İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Hastalıklı Yaprak Tespiti

Yazılım Mühendisliği Ana Bilim Dalı

Tezsiz Yüksek Lisans / Bitirme Projesinin Raporu

Oğuz Kuru

Y210234084

Proje Danışmanı: Doç. Dr. Aytuğ Onan

Haziran 2023

Yazarlık Beyanı

Ben, **Oğuz Kuru**, başlığı **Hastalıklı Yaprak Tespiti** olan bu raporun ve raporun içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu raporun herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 11.06.2023

Hastalıklı Yaprak Tespiti

Öz

Bu çalışmada CNN model kullanılarak veri setinde bulunan bitki türleri üzerindeki hastalıklar ve hastalık türleri tespit edilmeye çalışılmıştır. Aynı zamanda hastalıklı yaprak ve sağlıklı yaprak fark edilmeye çalışılmıştır. Derin öğrenme modeli ve görüntülerin işlenmesi için Python programlama dili kullanılmıştır.

Veri seti https://www.kaggle.com/datasets/emmarex/plantdisease adresinden alınmıştır. Bu veri 54305 adet resimden oluşmaktadır. Resimler 14 bitki türü ve bitki türleri üzerindeki hastalıklarla beraber 38 sınıftan oluşmaktadır.

Resimler ilk olarak Python dili kullanılarak belli ön işlemlerden geçirilip yapay zekâ eğitimi için hazırlanmıştır. Daha sonra hazırlanan veriler https://tfhub.dev/google/cropnet/feature_vector/cassava_disease_V1/1 adresinden alının hazır CNN modeli üzerinde transfer learning tekniği kullanılarak eğitilmeye çalışılmıştır.

En sonun da çıkan sonuçlar yapay zekâ değerlendirme teknikleri ile değerlendirilerek modelin başarı değeri elde edilmiştir.

Anahtar Sözcükler: Derin Öğrenme, CNN, Python, Görüntü İşleme, Transfer Learning, Yaprak Hastalıkları

Teşekkür

Bu projeyi hazırlamamda yapay zekâ kısmında bana yardımcı olan Sayın Doç. Dr. Aytuğ Onan hocama ve bitki haslıkları kısmında, proje konusu seçiminde yardımcı olan Sayın Prof. Dr. Arif Behiç Tekin hocama teşekkür ederim.

İçindekiler

Şekiller Listesi

Kısaltmalar Listesi

CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
CPU	Central Processing Unit

Bölüm 1

Giriş

Bu "**Hastalıklı Yaprak Tespiti"** projesinin amacı, tarım ürünlerinde meydana gelen hastalıkların, kameralar kullanılarak kişi veya kişilere ihtiyaç duymadan makineler tarafından tespitini sağlayabilmektir. Proje şu şekilde çalışmaktadır; bitki hastalıkları ve sağlıklı yaprak görüntülerinden oluşan resimler ile CNN modeli eğitilerek yapay zekanın hastalıkları ve sağlıklı yaprakları tespit edilmesi sağlanabilmektedir. Kullanılan derin öğrenme modeli bitki hastalıkları için geliştirilmiş özel bir TensorFlow Hub modelidir. Bu model bu tarz bitki görüntülerinden oluşan verilerde başarılı olurken diğer veri yapılarında başarısı düşmektedir.

Modelin eğitimi gerçekleştirilmeden önce verilerin eğitime hazırlanabilmesi için belli başlı ön işlemlerden geçmesi gereklidir. İlk olarak resim dosyalarının okunup bilgisayarın anlayabileceği sayısal formata yani matris formatına çevrilmesi gereklidir. Daha sonra projede kullanılan CNN modelini kullanabilmek için verilerin derin öğrenme modeline uygun yapıda olması gereklidir. Örneğin projede kullanılan resim 256x256 piksel boyutundayken kullanılan model için gerekli olan resim boyutu 224x224 piksel boyutlarındadır. Bunun için veri setindeki resimler ilk olarak 224x224 piksel boyutlarına getirilmiştir.

Bir diğer konu ise bu tarz resimlerden oluşan büyük verilerin eğitiminde CPU kullanıldığında eğitimler uzun sürmekte ya da eğitimler başarısız olmaktadır. Onun için bu tarz büyük verilerinin eğitiminde GPU tercih edilmedir. Bu projede bu nedenle modelin eğitimi sırasında GPU kullanabilmek için Google Colab üzerinde bulunan A100 model GPU üzerinde eğitimi sağlanmıştır.

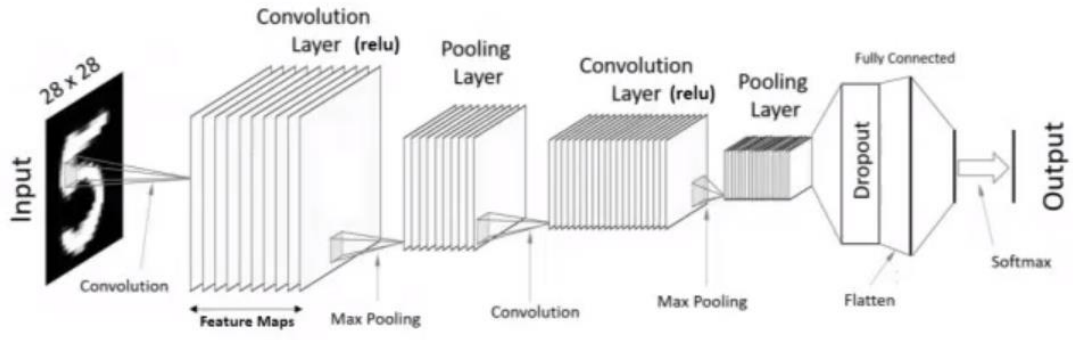
Bölüm 2

CNN ve Derin Öğrenme Parametreleri

Bu bölümde, yapay zeka yapılarından olan CNN' den ve yapay zeka eğitim tekniği olan transfer learning tekniğinden bahsedilmiştir.

2.1 CNN (Convolutional Neural Network)

CNN resim sınıflandırma projelerinde kullanılan, belli katmanlardan oluşan nesne tanımla temelli derin öğren modelidir.

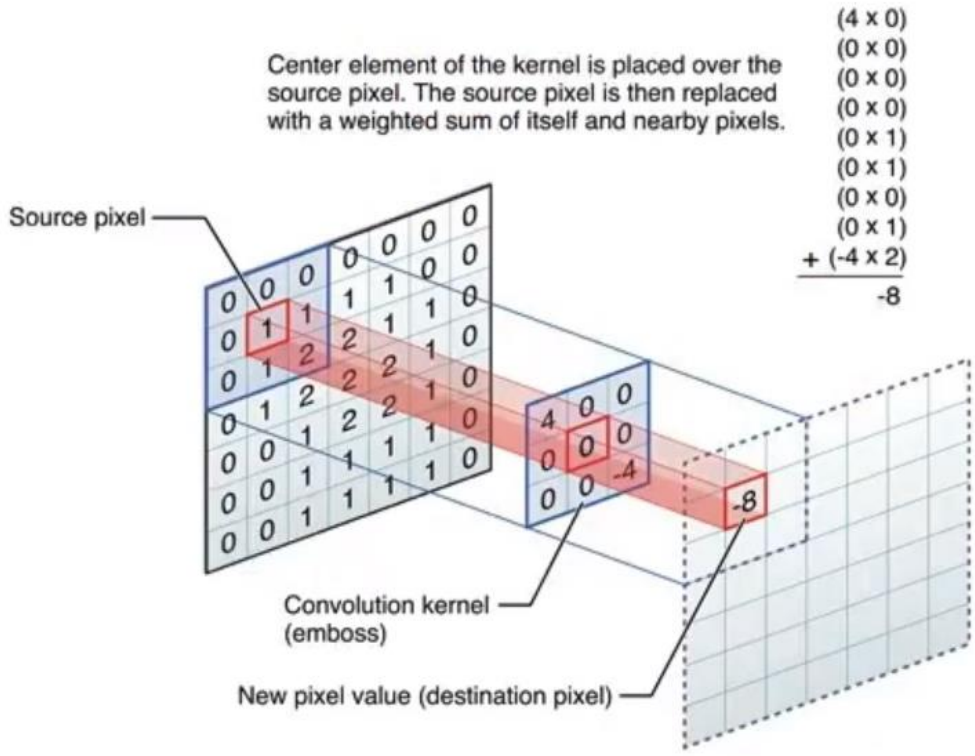


Şekil 2.1: CNN genel modeli

2.1.1 Convolutional Layer

CNN, input kısmında bulunan resmi filtrelerden geçirerek o resme ait bulunan belli özellikleri çıkartarak o resmin tanınmasına dayanan çalışma mantığını kullanır. Bu belli başlı özellikler resme ait köşeler, karmaşık şekiller olabilir. Örneğin yukarıdaki resimdeki 5 sayısının üst düz çizgisi veya alt kısmındaki yuvarlak kısmı gibi. Input

kısmındaki resimler matris formatında olduğu için bilgisayar bu özellikleri matematiksel formda tutar. Bu işlem yani resme ait belli başlı özelliklerin çıkarılması convolutional layer kısmında meydana gelir.

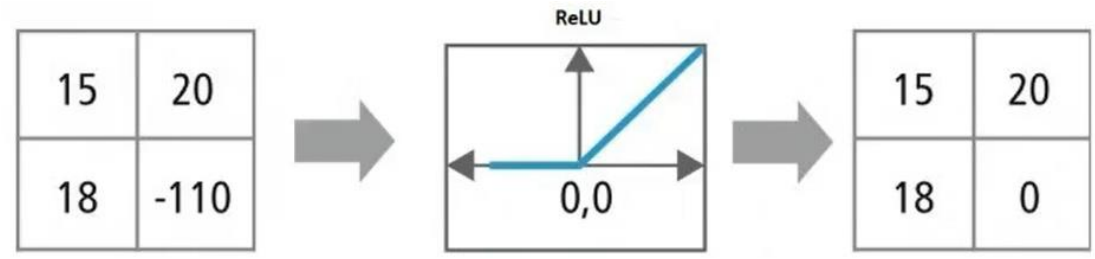


Şekil 2.2: Convolutional işlemi

Yukarıdaki resimdeki gibi resme ait matris değerleri üzerinde filtre (convolution kernel) uygulanarak matematiksel işlemler gerçekleştirilir ve resme ait şekiller saklanır. Bu işlem sırasında matrisin boyutu küçültülür.

2.1.2 ReLU Aktivasyon Fonksiyonu

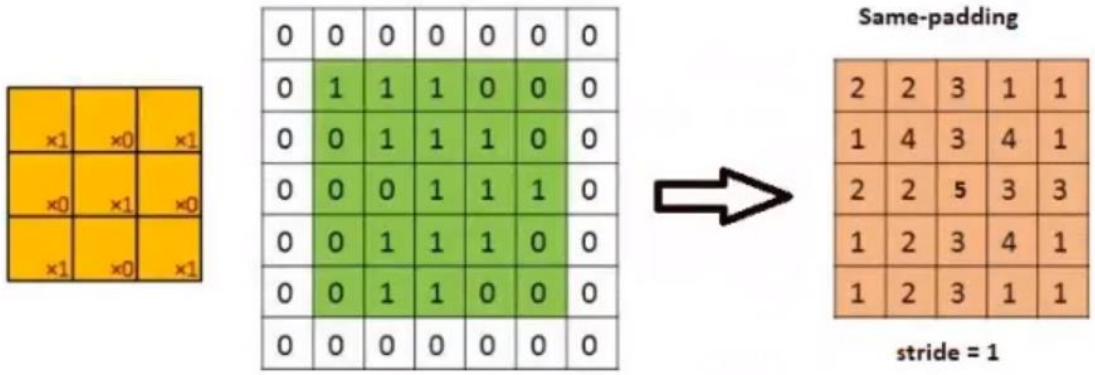
Convolutional işleminden sonra feature map çıktıktan sonra bu feature mape **ReLU** aktivasyon fonksiyonu uygulanarak nonlinearity arttırılır.



Şekil 2.3: ReLU aktivasyon fonksiyonu uygulanan matris değerlerindeki değişim.

2.1.3 Same Padding

Filtreleme işlemi sırasında verinin hacimsel boyutunda azalma meydana gelir ve bilgi kaybetmeye başlarız. Bu veri kaybını önlemek için filtreleme işleminden sonra veriye **same padding** işlemi uygulanır.

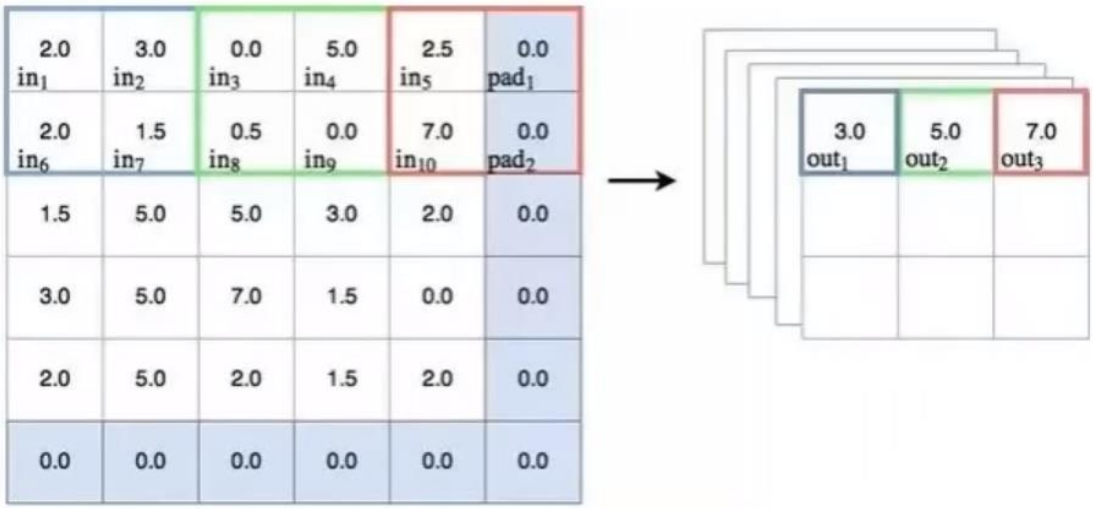


Şekil 2.4: Same Padding işlemi

Same padding işlemi sırasında kareler üzerinde 1 kare veya 2 kare gibi değişken sayılarda ilerletilebilir. Filtrenin bu kareler üzerinde ilerletilmesi işlemine stride denir ve ilerleme sayısına eşit olur.

2.1.4 Max Pooling

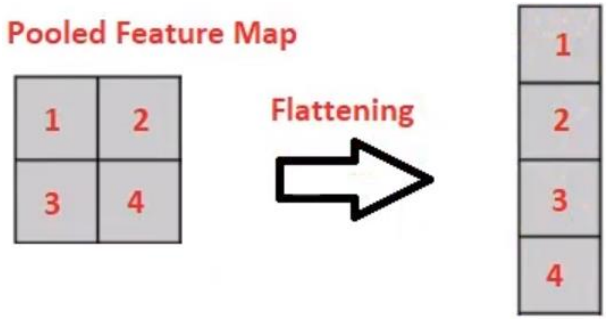
Same padding işleminden sonra resim boyutunu küçültmek ve overfittingi önlemek için Max Pooling işlemi gerçekleştirilir. Farklı pooling işlemleri (min pooling, avarege pooling) de vardır ancak bu projede ve genelde en çok max pooling yöntemi tercih edilir.



Şekil 2.5: Max Pooling işlemi

2.1.5 Flattening

En son işlem olan Flatten katmanında matris formatındaki veriler düz bir blok haline getirilir. Veri düz blok haline getirilirken overfittingi önlemek için dropout yöntemiyle bazı özellikler elenir. En son Flatten katmanından çıkan veriye sınıf sayısına göre softmax ya da sigmoid aktivasyon fonksiyonu uygulanır. Eğer sınıflandırma yapılacak veri 2 sınıftan oluşuyorsa sigmoid aktivasyon fonksiyonu, ikiden fazla sınıfa sahip ise softmax aktivasyon fonksiyonu uygulanır ve çıktı elde edilir.



Şekil 2.6: Flattening işlemi

2.2 Derin Öğrenme Parametreleri

Derin öğrenme algoritmalarında çalışmadan çalışmaya değişebilecek bazı parametreler vardır. Bu parametrelerin mutlak doğru değerleri yoktur. En iyi değerler, deneme iyileştirme stratejisi ile elde edilmeye çalışılır. Bu parametrelere hyper parametreler denir. Bu bölümde bazı hyper parametrelere değinilmiştir.

2.2.1 Batch Size

Derin öğrenme uygulamalarında, veri setinde bulunan tüm verilerin aynı anda aktarılması bellek ve zaman açısından zor bir iştir. Bu problemin çözülebilmesi için; veri seti küçük gruplara ayrılmaktadır. Bu şekilde birden fazla girdinin parçalar halinde işlenmesi **batch size** olarak adlandırılmaktadır. Model tasarlanırken batch-size parametresi olarak belirlenen değer; modelin aynı anda kaç veriyi işleyeceği anlamına gelmektedir.

Bazı özellikleri:

- Belirlenen batch değerinin GPU belleğine sığması gerektiği için 2' nin katları şeklinde belirlenir. 2, 4, 8, 16, 32 …
- CNN batch değerine karşı hassastır. Bu nedenle batch değerinde küçük değişiklikler başarımda büyük etkiler oluşturabilir.
- Batch size küçük olması iyileştirme (regularization) etkisi yaratmaktadır. Modele veri büyük gruplar halinde verildiğinde ezberleme daha fazla meydana gelir. [1]

2.2.2 Epoch

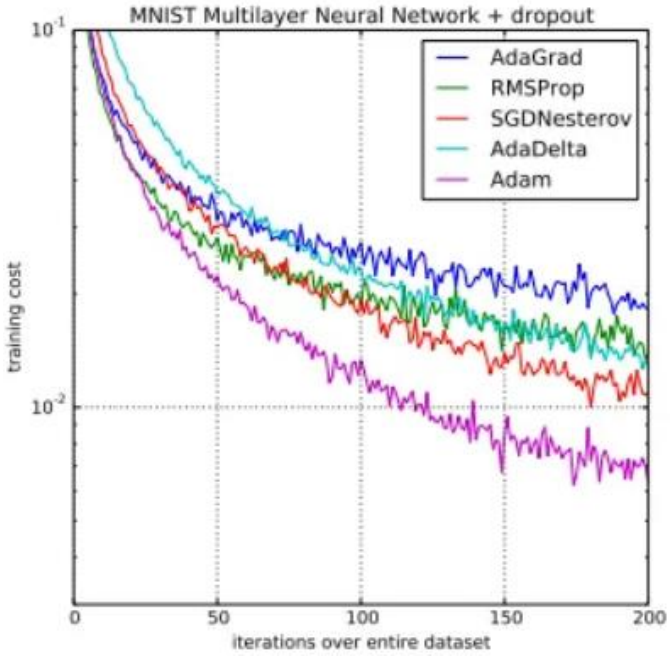
Model eğitilirken verilerin tamamı aynı anda eğitime katılamaz demiştik. Belli sayıda parçalar halinde eğitimde yer alırlar. İlk parça eğitilir, modelin başarımı test edilir, başarıma göre geriye yayılım ile ağırlıklar güncellenir. Daha sonra yeni eğitim kümesi

ile model tekrar eğitilip ağırlıklar tekrar güncellenir. Bu işlem her bir eğitim adımında tekrarlanarak model için en uygun ağırlık değerleri hesaplanmaya çalışılır. Bu eğitim adımlarının her birine “epoch” denilmektedir.

Derin öğrenmede problemi çözecek en uygun ağırlık değerleri adım adım hesaplandığı için ilk epoch’larda başarım düşük olacak, epoch sayısı arttıkça başarım da artacaktır. Bununla birlikte belli bir adımdan sonra modelimizin öğrenme durumu oldukça azalacaktır. Başarım belli bir epoch’dan sonra çok küçük birimlerde artış göstereceği için bu noktalarda eğitim sonlandırılabilir. [2]

2.2.3 Optimizer

Derin öğrenme uygulamalarında öğrenme işleminin temelde bir optimizasyon problemi olduğu için optimum değeri bulmak için optimizasyon yöntemleri kullanılmaktadır. Derin öğrenme uygulamalarında yaygın olarak stochastic gradient descent, adagrad, adadelta, adam, adamax gibi optimizasyon algoritmları kullanılmaktadır. [3] Bu çalışmada bilgisayarlı görü alanında iyi olduğu için ve learning rate gibi parametreleri kendi ayarladığı için **adam** optimizer seçilmiştir.



Şekil 2.7: Bazı optimizasyon algoritmalarının MNIST verisi üzerindeki çalışma zaman grafiği

Bölüm 3

Veri Seti ve Kodlar

Bu bölümde, veri setinin içeriğinden bahsedilip projede yer alan kodlar açıklanmıştır.

3.1 Veri Seti

Projede yer alan veri seti https://www.kaggle.com/datasets/emmarex/plantdisease web adresinden alınmıştır. Veri setinde toplam 54305 hastalıklı ve sağlıklı bitki yaprağı resmi bulunmaktadır. Resimler 256x256 boyutunda renkli resimlerden oluşmaktadır.

Verinin içindeki sınıflar ise şu şekildedir;

- Apple Scab
- Apple Rot
- Apple Rust
- Apple Healthy
- Bueberry Healthy
- Cherry Healthy
- Cherry Powdery Mildew
- Corn (Maize) Cercospora Leaf Spot Gray Leaf Spot
- Corn (Maize) Common Rust
- Corn (Maize) Healthy
- Corn (Maize) Northern Leaf Blight
- Grape Black Rot
- Grape Esca (Black Measles)
- Grape Healthy
- Grape Leaf Blight (Isariopsis Leaf Spot)

- Orange Haunglongbing (Citrus_Greening)
- Peach Bacterial Spot
- Peach Healthy
- Pepper Bell Bacterial Spot
- Pepper Bell Healthy
- Potato Early Blight
- Potato Healthy
- Potato Late Blight
- Raspberry Healthy
- Soybean Healthy
- Squash Powdery Mildew
- Strawberry Healthy
- Strawberry Leaf Scorch
- Tomato Bacterial Spot
- Tomato Early Blight
- Tomato Septoria Leaf Spot
- Tomato Spider Mites Two Spotted Spider Mite
- Tomato Target Spot
- Tomato, Tomato Mosaic Virus
- Tomato, Tomato Yellow Leaf Curl Virus

Şekil 3.1: Veri setindeki örnek resimler

3.2 Proje Kodları

Veri seti çok fazla sayıda renkli resimden oluştuğu için model eğitimi sırasında çok fazla işlem gücüne ihtiyaç vardır. Bu işlem gücü bilgisayarlarımızdaki CPU'lardan elde edeceğimiz işlem gücünden bile fazladır. Ekstra bir işlem gücüne yani GPU' ya gereksinim vardır. Bu yüzden bu projede yer alan kodlar Google Colab üzerinde sanal bir bilgisayar üzerinde ve GPU kullanılarak gerçekleştirilmiştir.

3.2.1 Veri Setinin Ortama Yüklenmesi

Veri seti Kaggle ortamından indirildikten sonra Google Colab ortamında kullanılabilmek için Google Drive kısmına zip dosyası olarak yüklenmiştir. Daha sonra zip dosyası Google Colab üzerinde kod yardımıyla çalışma ortamına açılmıştır.

```
!unzip "/content/drive/MyDrive/Colab Notebooks/PlantVillage/color.zip" -d "/content/drive/MyDrive/Colab Notebooks/PlantVillage"
```

Şekil 3.2: Zip dosyasını açan kod

3.2.2 Gerekli Kütüphaneler

- Tensorflow: Derin öğrenme modeli oluşturmak için gerekli kütüphane
- Tensorflow_hub: Trasfer Learning yöntemini kullanabilmek için gerekli kütüphane
- Pandas: Data frameler üzerinde işlem yapmak için gerekli kütüphane
- Matplotlib.pyplot: Verileri görselleştirmek için gerekli kütüphane
- IPython.display: Resim dostalarını görüntüleyebilmek için gerekli kütüphane
- Os: Klasörler içindeki dosyaları alabilmek için gerekli kütüphane
- Numpy: Array ve matris yapıdaki verilerle işlemler yapabilmek için gerekli olan kütüphane

3.2.3 Veri Setinin Okunması

Veri setini kodların içine aktarabilmek için resimlerin yüklü olduğu klasörün klasör yolunu os kütüphanesi kullanılarak bir değişkene atanmıştır.

```
images_path=os.listdir("/content/drive/MyDrive/Colab Notebooks/PlantVillage/color")
```

3.3: Resimlerin yüklü olduğu klasörlerin içe aktaran kod

Veri setimizdeki sınıfları gösteren bir csv dosyası olmadığı için sınıflar resimlerin dosya isimlerinden alınıp bir listede saklanmıştır.

```
[ ] images = []
    img_class = []
    for i in images_path:
      for j in os.listdir(i):
        images.append(i+"/"+j)
        a = i.split("/")
        img_class.append(a[-1])

[ ] images[0]

    'drive/MyDrive/Colab Notebooks/PlantVillage/color/Potato___Early_blight/001187a0-57ab-4329-baff-e7246a9edeb0___RS_Early.B 8178.JPG'

[ ] img_class[0]

    'Potato___Early_blight'
```

3.4: Sınıfların ve resim yollarının tutulduğu liste görüntüsü

38 sınıftan oluşan resimlerimiz listede string formda saklanmaktadır. Bilgisayarın bu sınıf etiketlerini daha iyi anlayabilmesi için sınıfların bir özellik ve sınıfa ait resimlerin o sınıf etiketi için True değerini, ait olmadığı sınıf etiketleri için de False etiketini alması gereklidir. Yani Sınıf özelliklerimiz boolean yapıda saklanmıdır. O da aşağıdaki resimdeki kodlar ile sağlanmaktadır.

```
[ ]  unique_class = np.unique(img_class)

[ ]  img_class= np.array(img_class)

[ ]  boolean_class = [i==unique_class for i in img_class]

[ ]  X=images
     y=boolean_class
```

Şekil 3.5: Sınıfları boolean forma dönüştüren kod görüntüsü

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False])
```

Şekil3.6: Boolean forma dönüştürüldükten sonra kodun çıktısı

3.2.4 Görüntü Verilerinin Model İçin Hazırlanması

Veri setindeki resimlerin modelde kullanılabilmesi için belli başlı ön hazırlıklardan geçerek modele uygun hale getirilmesi gereklidir. Örneğin veri setindeki resimler 256x256 formattayken model için gerekli olan yapı 224x224 pikseldir. Aynı zamanda şu an verilerimiz dosya yolu olarak bir listede saklanmaktadır. Bu dosya yolları okunup matris formatına çevrilmelidir. Bu ve bunun gibi işlemler bu kısımda gerçekleştirilmiştir.

İlk olarak verilerin bir kısmının train, bir kısmının validation ve kalan kısmının test verisi olarak ayrılması gerekir. Bu ayrılma oranı genellikle toplam verinin %80'i train için, %15'i test ve %5'i validation olarak gerçekleştirilir. Bu çalışmada veri setinde çok fazla görüntü olduğu için %20'si validation, %16sı test verisi için ayrılmıştır.

```
from sklearn.model_selection import train_test_split

X_train,X_valid,y_train,y_valid = train_test_split(X[:NUM_IMAGES],boolean_class[:NUM_IMAGES],test_size = 0.2, random_state=42)

X_train,X_test,y_train,y_test = train_test_split(X_train,y_train,test_size = 0.2,random_state = 42)
```

Şekil 3.7: Train,test ve validation split kodları

Bir sonraki aşama artık dosya yollarındaki resimlerin 224x224 ve matris formatını çevrilmesidir. Bunun için ilk dosya yollarının okunup decode edilerek bilgisayarın anlayabileceği yapıya çevrilmesi gerekir.

```
IMG_SIZE = 224


#Create a function for preprocessing images

def process_image(image_path,img_size = IMG_SIZE):


  image = tf.io.read_file(image_path)
  image = tf.image.decode_jpeg(image,channels = 3)
  image = tf.image.convert_image_dtype(image,tf.float32)
  image = tf.image.resize(image,size = [IMG_SIZE,IMG_SIZE])
  return image
```

Şekil 3.8: Görüntü dönüşüm kodları

3.2.5 Verinin Batchlere Ayrılması

Yukarıda da bahsettiğimiz gibi verilerin hepsinin tek seferde aktarılması zaman ve bellek bakımından maliyetli olduğu için paketler halinde yüklenmesi gereklidir. Bu işlemler de bu kısımda gerçekleştirilmiştir.

```
BATCH_SIZE = 32

def create_data_batches(X,y = None,batch_size = BATCH_SIZE,valid_data = False, test_data=False):
  if test_data:

    print("Creating test data batches...")
    data = tf.data.Dataset.from_tensor_slices((tf.constant(X)))
    data_batch = data.map(process_image).batch(BATCH_SIZE)
    return data_batch

  elif valid_data:

    print("Creating validation data batches...")
    data = tf.data.Dataset.from_tensor_slices((tf.constant(X),
                                               tf.constant(y)))
    data = data.map(get_image_label)
    data_batch = data.batch(BATCH_SIZE)
    return data_batch

  else:

    print("Creating train data batches...")
    data = tf.data.Dataset.from_tensor_slices((tf.constant(X),
                                               tf.constant(y)))
    data = data.shuffle(buffer_size =(len(X)))
    data = data.map(get_image_label)
    data_batch = data.batch(BATCH_SIZE)
    return data_batch
```

Şekil 3.9: Verinin Batchlere ayıran kod görüntüsü

Yukarıdaki kodlarda train, validation ve test verisi için 3 ayrı yöntem kullanılmıştır. Bunun nedeni train verilerinde shuffle uygulanarak modelin karıştırılması sağlanarak belli sınıfları ezberlemesini önlemektir. Aynı zamanda test verilerinde sınıf etiketleri (y verisi) olmadığı için sadece görüntüler üzerinde batch gerçekleştirilmiştir.

3.2.6 Modelin Oluşturulması

Modeli kendimiz oluşturabileceğimiz gibi dışarıdan hazır modeller alınarak da çalışmayla ilgili modeller kullanılabilir. Modeli kendimizin oluşturması zaman bakımından maliyetli olduğu için daha önce bu konuyla ilgili benzer çalışmalardan hazır model almak hem başarı hem zaman için kazanç sağlar. Bu tarz dışarıdan hazır model alınıp kullanılması yöntemine transfer learning yöntemi denir.

Bu çalışma bitki hastalıkları ile ilgi olduğu için bu konuya en yakın çalışma olan Cassava disease çalışmasında oluşturulan model kullanılmıştır. Bu model manyok bitkisi üzerindeki hastalıkları tahminlemek için oluşturulan bir derin öğrenme modelidir. Model 224x224 renkli veriler üzerinde çalışır.

```
INPUT_SHAPE = [None,IMG_SIZE,IMG_SIZE,3]
OUTPUT_SHAPE = len(unique_class)
MODEL_URL = "https://tfhub.dev/google/cropnet/feature_vector/cassava_disease_V1/1"

def create_model(input_shape = INPUT_SHAPE,output_shape =OUTPUT_SHAPE,model_url = MODEL_URL ):
  print("Building model with:",MODEL_URL)
  model = tf.keras.Sequential([hub.KerasLayer(MODEL_URL),
                               tf.keras.layers.Dense(units = OUTPUT_SHAPE,activation = "softmax")])
  model.compile(loss = tf.keras.losses.CategoricalCrossentropy(),
                optimizer = "adam",
                metrics = ["accuracy"])
  model.build(INPUT_SHAPE)
  return model
```

Şekil 3.10: Model oluşturma kodları

Yukarıda resimde de görüldüğü gibi dense katmanında 2'den fazla sınıfımız olduğu için softmax kullanılmıştır. Optimizer olarak da adam optimizer seçilmiştir.

3.2.7 Eğitim Süreci

Modelde epoch sayısı 50 seçilmiş gibi görünse de modelde early stopping methodu kullanılmıştır. Bu metot 3 adımdan sonra modelde herhangi bir gelişme olmaması durumunda eğitimi durdurmaktadır. Adım sayısı bizim verdiğimiz göre değişebilir.

Değerlendirme metriği olarak da categorical crossentropy tercih edilmiştir. Categorical crossenropy tercih edilmesinin nedeni veriyi bölüp değerlendirmeyi verinin bölünmüş parçalarında tek tek gerçekleştirmesidir. Bu sayede verinin her kısmındaki başarı oranı görüntülenebilmektedir.

```
model = train_model()

Building model with: https://tfhub.dev/google/cropnet/feature_vector/cassava_disease_V1/1
Epoch 1/50
600/600 [==============================] - 41s 57ms/step - loss: 0.4668 - accuracy: 0.8683 - val_loss: 0.2510 - val_accuracy: 0.9212
Epoch 2/50
600/600 [==============================] - 30s 49ms/step - loss: 0.1835 - accuracy: 0.9454 - val_loss: 0.1881 - val_accuracy: 0.9427
Epoch 3/50
600/600 [==============================] - 30s 50ms/step - loss: 0.1361 - accuracy: 0.9593 - val_loss: 0.1585 - val_accuracy: 0.9488
Epoch 4/50
600/600 [==============================] - 30s 50ms/step - loss: 0.1116 - accuracy: 0.9666 - val_loss: 0.1337 - val_accuracy: 0.9575
Epoch 5/50
600/600 [==============================] - 30s 50ms/step - loss: 0.0960 - accuracy: 0.9711 - val_loss: 0.1236 - val_accuracy: 0.9594
Epoch 6/50
600/600 [==============================] - 29s 49ms/step - loss: 0.0820 - accuracy: 0.9759 - val_loss: 0.1161 - val_accuracy: 0.9629
Epoch 7/50
600/600 [==============================] - 30s 50ms/step - loss: 0.0727 - accuracy: 0.9777 - val_loss: 0.1137 - val_accuracy: 0.9654
Epoch 8/50
600/600 [==============================] - 29s 49ms/step - loss: 0.0650 - accuracy: 0.9808 - val_loss: 0.1074 - val_accuracy: 0.9644
Epoch 9/50
600/600 [==============================] - 29s 49ms/step - loss: 0.0582 - accuracy: 0.9823 - val_loss: 0.1126 - val_accuracy: 0.9654
Epoch 10/50
600/600 [==============================] - 29s 48ms/step - loss: 0.0532 - accuracy: 0.9853 - val_loss: 0.1212 - val_accuracy: 0.9585
```

Şekil 3.11: Eğitim süreci

Bölüm 4

Proje Sonuçlarının Değerlendirilmesi

En son sonuçlarımızı değerlendirdiğimizde accuracy değerinin 0.9853, val_accuracy değerinin 0.9585 olduğunu görebiliyoruz. Değerler %95'in üzerinde yani gayet iyi bir model. Aynı zamanda değerler birbirine yakın olduğu için büyük ihtimal modelde ezberleme yok diyebiliriz. Bunu netleştirmek için model daha sonra test verileri üzerinde de test edilip sonuçları değerlendirilmelidir.

4.1 Test Verisinin Sonuçları

Yukarıda bahsedilen sonuçlar model hakkında biraz fikir verse de son olarak modeli test verileri üzerinde de test edip değerlendirmek model üzerinde emin olmamızı sağlar.

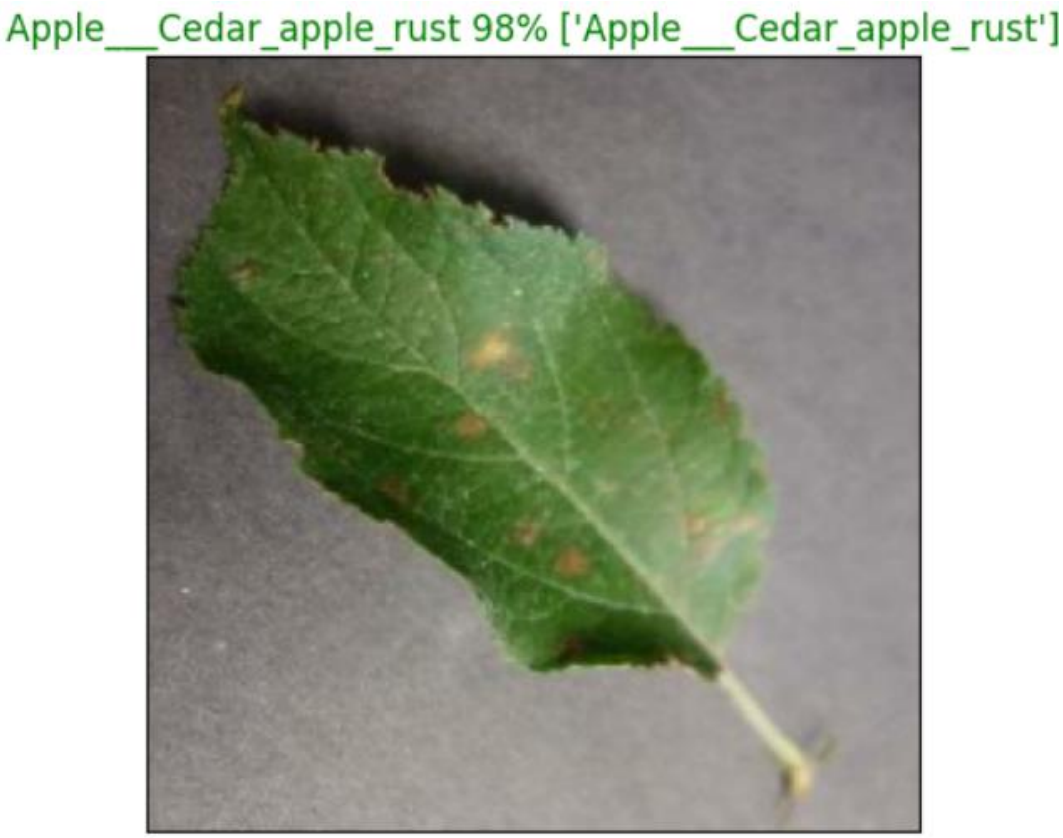
Test verileri modelin daha önce görmediği görüntülerden oluşur ve modele sadece görüntü (sınıf etiketleri verilmez) olarak verilir. Modele bu test verileri tahmin ettirilir daha sonra çıkan sonuçları biz kendimiz sınıf etiketlerine ve sonuçlara bakarak değerlendirebiliriz.

```
predictions = model.predict(test_data,verbose = 1)
predictions
```

```
175/175 [==============================] - 8s 41ms/step
array([[8.8391796e-05, 2.7356868e-05, 6.3105610e-05, ..., 2.3607147e-07,
        2.5475030e-10, 7.3530043e-07],
       [1.7697035e-09, 5.0487806e-06, 3.6297676e-10, ..., 3.3918366e-07,
        1.4324135e-14, 4.3358799e-07],
       [1.8371628e-08, 2.8063976e-08, 2.5023488e-12, ..., 1.7087535e-08,
        7.1380532e-12, 2.1192413e-04],
       ...,
       [5.5320676e-10, 4.2172755e-11, 1.4710420e-10, ..., 3.7827484e-12,
        9.9352433e-15, 5.0878052e-13],
       [4.7027115e-27, 2.1695382e-28, 2.2001614e-33, ..., 1.0000000e+00,
        5.1132094e-32, 1.2622707e-26],
       [5.7555248e-16, 3.0199503e-15, 1.3847264e-08, ..., 2.1675367e-04,
        1.6180323e-27, 2.8841861e-15]], dtype=float32)
```

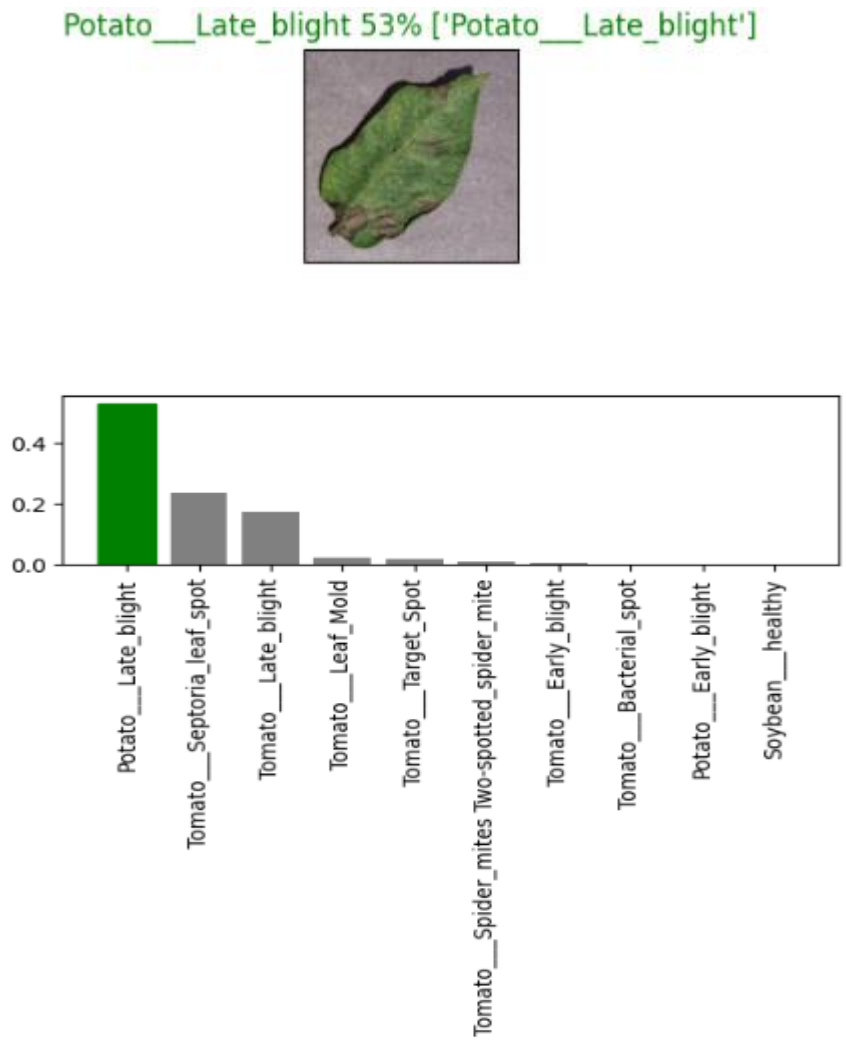
Şekil 4.1: Test verilerinin tahminleme işlemi

Yukarıda sonuçlar görüldüğü gibi matris formatındadır. Bu sayılar şunu gösteriyor, görüntünün 38 ayrı sınıf için o sınıfa ait olabilme olasılığı.



Şekil 4.2: Tahminlenen test verisinin tahminleme olasılığı ile görüntüsü

Yukarıda resimde de görüldüğü gibi model %98 olasılıkla görüntünün Apple Cedar Apple Rust sınıfına ait olduğunu tahminlemiş ve gerçekten de görüntü o sınıfa ait çıkmış.



Şekil 4.3: Görüntü için tahminlenen olma olasılığı en yüksek 10 sınıf

Yukarıdaki resimde de görüldüğü gibi model %53 olasılıkla en yüksek tahmin olarak Potato late blight tahminlenmiştir. İkinci en yüksek olasılıkla Tomato septoria leaf spot olarak tahminlemiştir. %53 olasılıkla tahminlenmesine rağmen tahminde başarılı olunmuştur.

Sonuç olarak model train,validation ve test verilerinde yüksek başarı oranına sahiptir. Model verilerde yer alan görüntüler için kullanılabilir.

Kaynaklar

Tek yazarlı doi numaralı dergi makalesi örneği:

[1] Neural Network Hyperparameters, http://colinraffel.com/wiki/neural_network/hypermarameters.

[2] Andrew Ng, 2018, Understanding mini-batch gradient descent.

[3] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, Benjamin Recht, 2017, "The Marginal Value of Adaptive Gradient Methods in Machine Learning", https://arxiv.org/pdf/1705.08292.pdf

T.C.
İzmir Kâtip Çelebi Üniversitesi
Fen Bilimleri Enstitüsü

HASTALIKLI YAPRAK TESPİTİ

Yazılım Mühendisliği Ana Bilim Dalı
Yüksek Lisans / Bitirme Proje Raporu

Oğuz Kuru
Y210234084

Tez Danışmanı: Doç. Dr. Aytuğ Onan

Haziran 2023

OĞUZ KURU	HASTALIKLI YAPRAK TESPİTİ RAPORU	2023	YÜKSEK LİSANS

Özgeçmiş

Adı Soyadı: Oğuz Kuru

Eğitim:

2015–2021 İzmir Kâtip Çelebi Üniversitesi, Makine Müh. Bölümü