KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

MAKİNE ÖĞRENMESİ PROJESİ

CHEST X-RAY

Mehtap ÖKLÜ – 17110131052

Banu KÖSE – 18110131011

İÇİNDEKİLER

1-GİRİŞ

Zatürre, Tüberküloz rahatsızlıkları günümüzde halen var olan çok ciddi hastalıklardır. Zatürre, akciğer dokusunun iltihaplanmasıyla oluşan bir hastalıktır. Tüberküloz, diğer adıyla verem hastalığı bir bireyden diğerine yayılan bulaşıcı bir akciğer hastalığıdır. Bu hastalıkların tespiti için kişinin göğüs x-ray görüntüleri kullanılmaktadır.

2-PROJENİN AMACI

Chest X-Ray projesi, insan göğsünün X-ray görüntüleri kullanılarak kişinin zatürre, tüberküloz olup olmadığını bize veren bir projedir. Bu proje ile tanı koyan doktorların işi daha kolay hale gelecek olup, vakit kayıpları daha aza indirgemek amaçlanmıştır.

3-UYGULAMA GELİŞTİRME SÜRECİ

Bu süreci 3 aşamada değerlendirebiliriz.

3.1-KULANILACAK OLAN VERİLERİN BELİRLENMESİ

Bu aşamada projede kullanacağımız veri seti belirlemek için Kaggle'dan veri seti araştırması yaptık. Kaggle, hazır veri setleri barındıran, veri bilimci ve makine öğrenmesi uygulayıcıların bulunduğu bir web sitesidir. Araştırılan veri setlerinin kullanılabilir olup olmadığı tespit edildi.

Kullanılan veri setinin linki: https://www.kaggle.com/jtiptj/chest-xray-pneumoniacovid19tuberculosis

3.2-VERİ SETİ DÜZENLEME SÜRECİ

Bu aşama seçtiğimiz veri setinin ön hazırlık aşamasından geçmesidir. Yazdığımız ön hazırlık kodları ile elimizdeki veride bulunan gereksiz ya da gürültülü verileri temizleyerek veri setimizi kullanılabilir hale getirdik.

3.3-KODLARIN YAZILMASI VE ÇALIŞTIRILMASI

Hazır hale getirdiğimiz veri seti için kodların hazırlandığı aşamadır. Hazır olan veri setinin modelinin oluşturulması ve öğretilmesi için yapay zeka kodları yazıldı. Daha sonra elde edilen doğruluk verileri ile grafikler oluşturuldu.

4-KULLANILAN YAZILIMLAR

Proje için Spyder arayüzü ve Python dili kullandı.

5-PROJENİN ÇALIŞMA MANTIĞI

Kaggle'dan elde ettiğimiz veri setini doğru sonuç almak için ön hazırlık aşamasına soktuk. Bu ön hazırlık aşamasında düzenlemek istediğimiz veri setinin yolu kullanılarak veriler koda yüklendi. Yüklenen verilerin boyutları ayarlandı. Boyutları düzenlenen verilerin sırasıyla; gürültüsü silindi, grayscale olmayan veriler grayscale hale getirildi, segmente edildi. Segmente edilen veriler background ve foreground olarak ikiye ayrıldı. Background kısmında ciğerin tamamını görsel olarak kaydedip arka planı silinirken ardından foreground kısmında ciğerin gereksiz kısımlarını kaydeder ve arka planı silindi. Ardından ciğerin resimlerinden ciğerin gereksiz kısımlarını sildik ve elimizde bizim için gerekli olan kısım kaldı. Veri setinin ön hazırlık aşaması tamamlanmış oldu.

Ön hazırlıktan geçmiş ve hazır olan veri setimizi eğitim ve test olarak ikiye ayırdık. Bu ayırma işleminden sonra yapay zekayı eğitmede kullanacağımız modeli oluşturduk. Modeli oluştururken CNN algoritması kullandık. CNN algoritması, girdi olarak resim alan ve farklı katmanlardan oluşan bir derin öğrenme algoritmasıdır. Oluşturulan modeli compile ederek eğitime hazır hale getirdik. Ardından modelimizi fit ederek yapay zeka eğitimini yapmış olduk.

Modeli oluşturup yapay zekayı eğittikten sonra elde ettiğimiz verileri grafik oluşturmak için değişkenlere atadık. Daha sonra bu değişkenleri kullanarak grafikler oluşturduk.

Yaptığımız projenin doğru çalışıp çalışmadığını anlamak için bir fonksiyon yazdık. Bu fonksiyonu konsoldan çağırarak bir test dosyası içerisindeki görselde zatürre olup olmadığını hem yazılı hem de görsel olarak alabildik.

Tüm bu işlemlerden sonra projenin çalışma süresini ekrana yazacak kod bloğunu ve bir verinin üzerinde çalışırken diğer verinin bellekte hazır olmasını sağlayan kod bloğunu ekledik.

Son olarak run.py adında bir python dosyası daha oluşturduk. Bu yeni kod dosyası ile projeyi her çalıştırdığımızda çıktıları bir .txt dosyasında tutmak istedik. Amacımız eski epoch çıktılarını ve yeni epoch çıktılarının değerlerini karşılaştırmak istememiz.

6-ÖN HAZIRLIK AŞAMASI KODU

```
PATH = r"C:\Users\Mehtap\Desktop\preprocess\preprocessed_data\train"

HEIGHT = 512
WIDTH = 512
IMG_SIZE = (HEIGHT, WIDTH)
```

Şekil1 Yol Belirtme ve Boyutlandırma

Yukarıda verilen kod bloğunda PATH'e preprocess yapılacak olan verinin dosya yolu ve fotoğrafların boyutu verildi.

```
def load_images(folder_name, extension):
    image_files = []


    for file in os.listdir(PATH + "\\" + folder_name):
        if file.endswith("." + extension):
            image_files.append(os.path.join(PATH, folder_name, file))

    return image_files
```

Şekil2 Yol Belirtme ve Boyutlandırma

Fonksiyonu çağırırken gönderdiğimiz parameterelere göre, konumdaki belli uzantıya sahip görselleri import eden fonksiyon. İçerisindeki for döngüsü ile process yapılacak olan konumda, belirtilen klasöre (folder_name) girerek içindeki görselleri tek tek döndeririz. For içindeki if bloğu ile belli uzantılardaki (extension) dosyaları seçer, görselleri list'e kaydederiz. Son olarak seçilmiş verileri return ederiz.

Şekil3 Örnek Görsel (Orijinal Veri)

```
def display_one(img, title1 = "Orjinal"):

    plt.imshow(img), plt.title(title1)
    plt.xticks([]), plt.yticks([])
    plt.show()
```

Şekil4 Tek Görseli Ekrana Bastırma

Kendisine parametre olarak gönderilen tek görseli ekrana çıkartan fonksiyon. plt.imshow içerisindeki img'yi ekrana çıkartılacak görsel olarak belirler ve başlık olarak title1'i ayarlar.

```
def display_two(img1, img2, title1 = "Orjinal", title2 = "Düzenlenmiş"):

    plt.subplot(121), plt.imshow(img1), plt.title(title1)
    plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(img2), plt.title(title2)
    plt.xticks([]), plt.yticks([])
    plt.show()
```

Şekil5 İki Görseli Ekrana Bastırma

display_two fonksiyonu kendisine parametre olarak gönderilen 2 adet görseli ekrana çıkartır. İmg1 ve img2'yi ekrana görsel olarak verecek başlıklarını sırasıyla title1 ve title2 yapacak.

```
def preprocessing(data):
    images = []
    resized_images = []
    no_noise = []
    segmented = []
    unknown = []
```

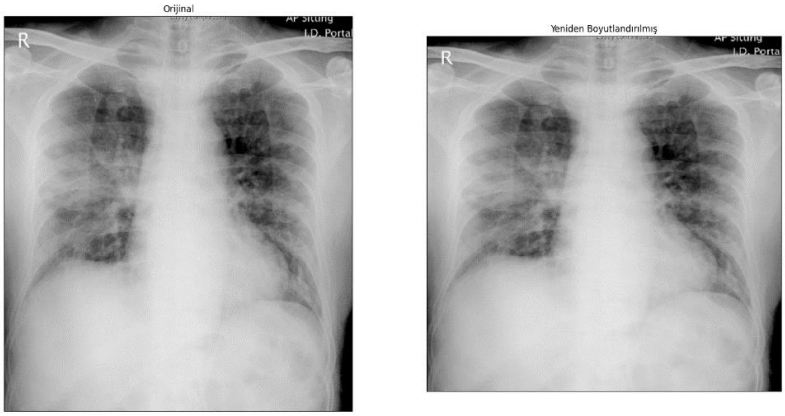
Şekil6 Process İşlemini Yönetme

Preprocess işlemini yöneten fonksiyon.

```
for img in data:
        images.append(cv2.imread(img, cv2.IMREAD_UNCHANGED))

    for i in range(len(images)):
        res_img = cv2.resize(images[i], IMG_SIZE, interpolation=cv2.INTER_LINEAR)
        resized_images.append(res_img)
```

Şekil7 Resized İmages

İlk for döngüsü ile görselleri datadaki konum listesinden alır ve images içine kaydeder. İkinci for döngüsü ile images list'in içindeki görselleri yeniden boyutlandırır , kırpar ve resized_images içine kaydeder.

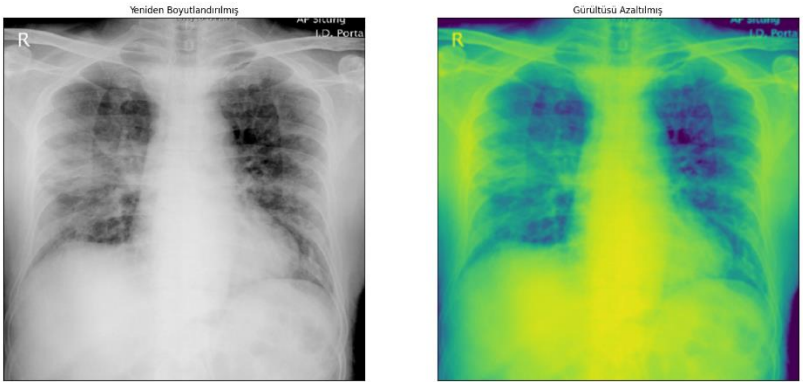


Şekil8 Boyutlandırılması Yapılmış Görsel

```
for i in range(len(resized_images)):
    blur = cv2.GaussianBlur(resized_images[i], (5, 5), 0)
    no_noise.append(blur)
```

Şekil9 Gürültü Silme

Resized_images içerisindeki görsellerin gürültüsünü silerek no_noise içerisine kaydeder.



Şekil10 Gürültüsü Silinen Görsel

```
for i in range(len(images)):
    if images[i].shape != IMG_SIZE:
        try:
            no_noise[i] = cv2.cvtColor(no_noise[i], cv2.COLOR_RGB2GRAY)
        except:
            continue
```

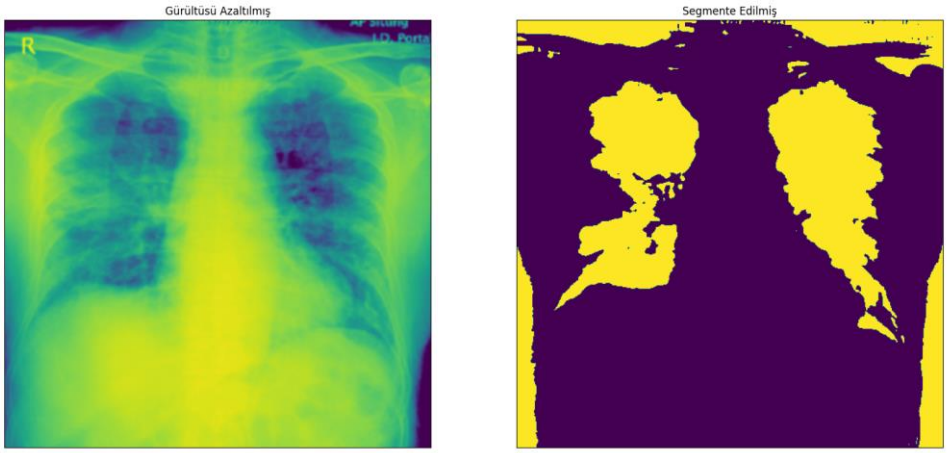
Şekil11 Grayscale Moduna Çevirme

no_noise içindeki grayscale olmayan görselleri seçip grayscale moduna çevirir.

```
for i in range(len(no_noise)):
    ret, thresh = cv2.threshold(no_noise[i], 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
    segmented.append(thresh)
```

Şekil12 Segment Etme

no_noise içindeki verileri segmente edip 'segmented' içine kaydeder.



Şekil13 Segmente Edilmiş Resim

```
for i in range(len(segmented)):
    kernel = np.ones((3, 3), np.uint8)
    opening = cv2.morphologyEx(segmented[i], cv2.MORPH_OPEN, kernel, iterations=2)
    sure_bg = cv2.dilate(opening, kernel, iterations=3)

    dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
    ret, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)
    sure_fg = np.uint8(sure_fg)

    unkn = cv2.subtract(sure_bg, sure_fg)
    unknown.append(unkn)
```

Şekil14 Background ve Foreground

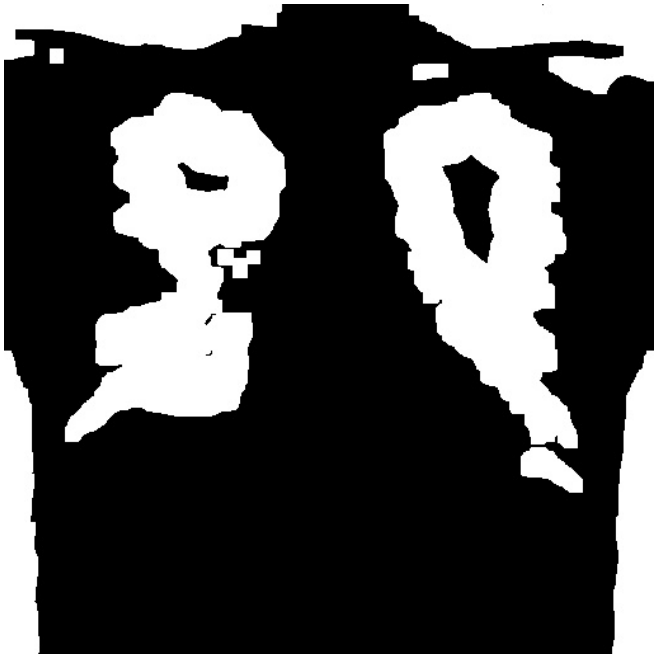
Segmented içindeki verileri background ve foreground olarak ikiye ayırır. Segmente verilen background'u kaldırarak foreground kısmını unknown içine kaydeder. Background ciğerin tamamını görsel olarak kaydeder, arka planı siler. Sadece ciğer kalır. Foreground, ciğerin gereksiz kısımlarını görsel olarak kaydeder, arka planı siler. Substract ile ciğerin gereksiz kısımları çıkartılır. Geriye sadece gerekli kısım kalır.

Şekil15 Background ve Foreground İşlemi Sonrası

```
for i in range(len(images)):
    cv2.imwrite(data[i], unknown[i])
```

Şekil16 Üzerine Yazarak Kaydetme

Verilerin düzenlenmiş halini, üstüne yazarak kaydeder.



Şekil17 Final

```
preprocessing(load_images("COVID19", "png"))
preprocessing(load_images("COVID19", "jpg"))
preprocessing(load_images("COVID19", "jpeg"))
```

Şekil18 Farklı Uzantı Formatları

Veri seti içerisindeki farklı fotoğraf uzantıları göz önüne alınarak işlem yapılır.

7- MODELİ OLUŞTURMA EĞİTİM GRAFİK OLUŞUMU VE KONTROL KODLARI

```
def start_time(name):
    print(name + " islemi basladi.")
    return time()

def stop_time(start_time, name):
    print(name + " islemi bitti. ({:.2f} saniye)".format(round((time() - start_time), 2)))
```

Şekil19 Zaman Öğrenme Fonksiyonu

İşlem kaç saniyede çalışmış onu öğrenmek için yazılan bir kod bloğu.

```
40/50 [=======================>......] - ETA: 45s - loss: 0.3795 - accuracy: 0.9047
41/50 [=======================>......] - ETA: 41s - loss: 0.3713 - accuracy: 0.9066
42/50 [========================>.....] - ETA: 36s - loss: 0.3629 - accuracy: 0.9089
43/50 [========================>.....] - ETA: 32s - loss: 0.3554 - accuracy: 0.9106
44/50 [=========================>....] - ETA: 27s - loss: 0.3483 - accuracy: 0.9123
45/50 [=========================>...] - ETA: 22s - loss: 0.3416 - accuracy: 0.9132
46/50 [=========================>...] - ETA: 18s - loss: 0.3346 - accuracy: 0.9151
47/50 [==========================>..] - ETA: 13s - loss: 0.3287 - accuracy: 0.9166
48/50 [==========================>..] - ETA: 9s - loss: 0.3222 - accuracy: 0.9183
49/50 [==========================>.] - ETA: 4s - loss: 0.3172 - accuracy: 0.9196
50/50 [==============================] - ETA: 0s - loss: 0.3150 - accuracy: 0.9201
50/50 [==============================] - 229s 5s/step - loss: 0.3150 - accuracy: 0.9201
Model egitme(fit) islemi bitti. (229.84 saniye)
```

Şekil20 Zaman Öğrenme Fonksiyonunun Çıktısı

```
TRAIN = r"train"
VALID = r"val"
TEST  = r"test"
```

Şekil21 Yol Belirtme

Ön işleme sonrası veri setinde bulunan train, test, validation klasörlerinin yolu.

```
batch_size = 64
img_height = 512
img_widht = 512
noepochs = 10
epochs_range=range(noepochs)
```

Şekil22 Boyutlandırma ve Eğitim Değişkenleri

Görsellerin boyutları ve eğitim değişkenleri. Noepochs, eğitim adımlarının tamamıdır. Batch_size ise bir seferde yapay sinir ağını eğitmek için kullanılacak örnek sayısını belirtir.

```
base_time = start_time("Training dataset import")
train_ds = image_dataset_from_directory(
    TRAIN,
    validation_split = 0.3,
    subset = 'training',
    seed = 123,
    image_size = (img_height, img_widht),
    batch_size = batch_size
)
stop_time(base_time, "Training dataset import")
```

```
base_time = start_time("Validation dataset import")
valid_ds = image_dataset_from_directory(
    VALID,
    validation_split = 0.3,
    subset = 'validation',
    seed = 123,
    image_size = (img_height, img_widht),
    batch_size = batch_size
)
stop_time(base_time, "Validation dataset import")
```

Şekil23 Parçalama İşlemleri

Train için yazılmış kod bloğu. validation_split ve subset ile parçalama işlemi yaptık. Seed argümanı ile karıştırma ve dönüşüm için isteğe bağlı işlem yaptık. İmg_size resim boyutlarımızı barındırıyor.

```
test_dir = []
base_time = start_time("Test dataset import")
for img in os.listdir(TEST):
    test_dir.append(os.path.join(TEST, img))
stop_time(base_time, "Test dataset import")
```

Şekil24 Konumlarını Kaydetme

Yukarıdaki kod bloğu ile test konumundaki tüm fotoğrafların konumlarını listeye kaydediyoruz.

```
class_names = train_ds.class_names
num_classes = len(class_names)
print(class_names)
```

Şekil25 Sınıf Bilgisi

Sınıflar hakkında bilgi verildi.

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
valid_ds = valid_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Şekil26

AUTOTUNE ile bir veri üzerinde çalışırken, aynı anda sıradaki verinin bellekte hazır hale getirilmesini sağlarız.

```
base_time = start_time("Model olusturma")
model = Sequential([
    layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_widht, 3)),
    layers.Conv2D(16,3,padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32,3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64,3,padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(128,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(256,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.3),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dense(num_classes)
])
stop_time(base_time, "Model olusturma")
```

Şekil27 CNN Algoritmasının Uygulanması

Makine öğrenmesi algoritmalarından olan CNN algoritmasını yukarıdaki kod parçacığı ile projemize ekledik. Görsele uyguladığımız boyutlardan dolayı çıktı orijinal görselden daha küçük olur. Bunu önlemek için padding isayesinde dolgulama işlemi yaptık. Activation=’relu’ sayesinde negatif değerlerden kurtulduk.

```
base_time = start_time("Model derleme(compile)")
model.compile(
    optimizer = 'adam',
    metrics=['accuracy'],
    loss=SparseCategoricalCrossentropy(from_logits=True)
)
stop_time(base_time, "Model derleme(compile)")
```

Şekil28 Model Compile Etme

Modeli compile ederek eğitim işlemine hazır hale getirdik.

```
base_time = start_time("Model egitme(fit)")
my_model=model.fit(
    train_ds,
    epochs = noepochs,
    validation_data = valid_ds
)
stop_time(base_time, "Model egitme(fit)")
```

Şekil29 Modeli Eğitme

Modeli eğitim işlemini yaptık

```
acc = my_model.history['accuracy']
val_acc = my_model.history['val_accuracy']
loss = my_model.history['loss']
val_loss = my_model.history['val_loss']
```

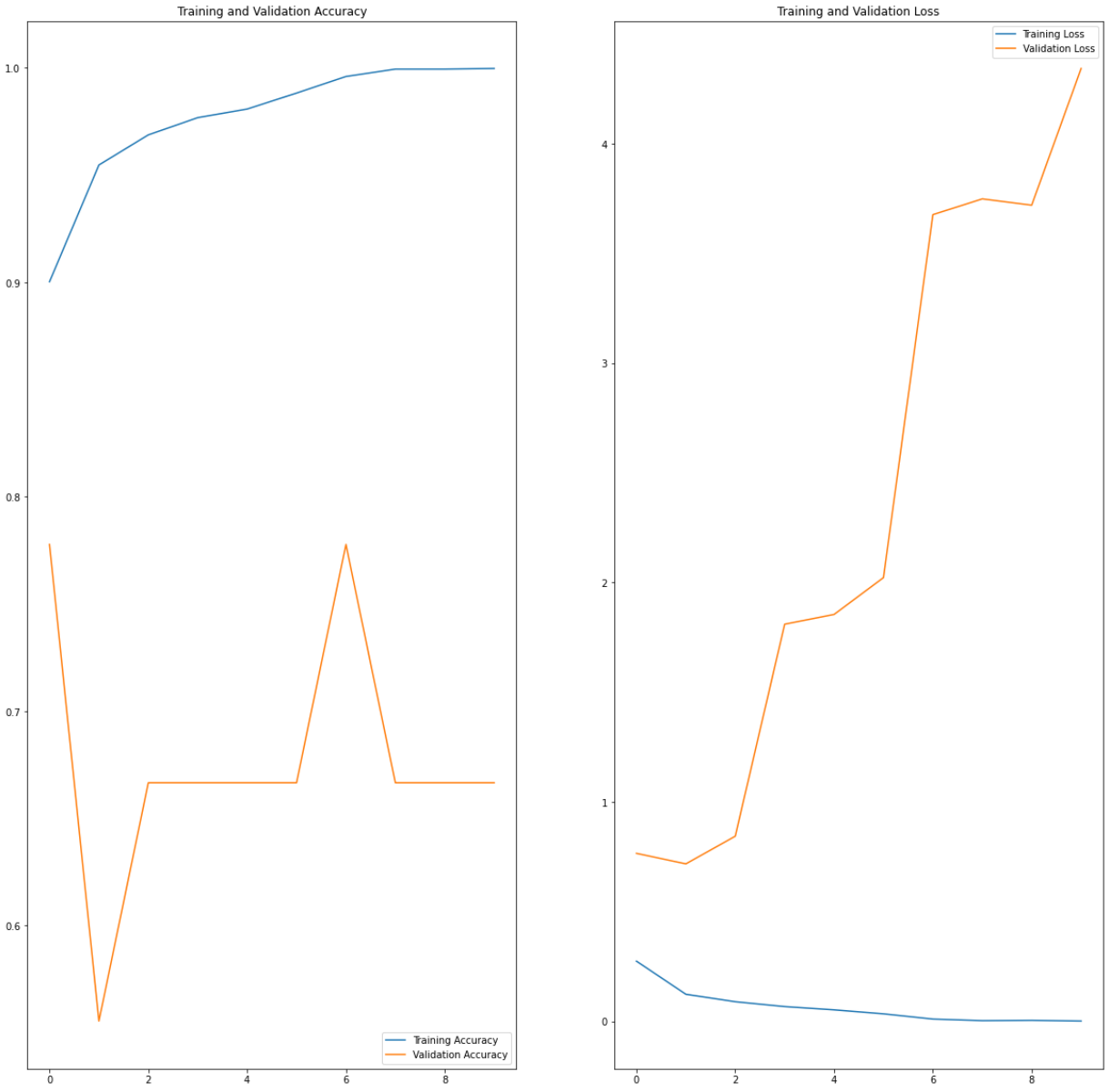
Şekil30 Çıkan Değerleri Tutma

Eğitim sonrasında çıkan verileri değişkenlere atadık.

```
plt.figure(figsize=(20, 20))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range,val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range,loss,label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Şekil31 Grafiğe Dökme

Son hale getirilen veririn, işleme sırasında oluşan değerlerini tutarak grafik haline getirdik.



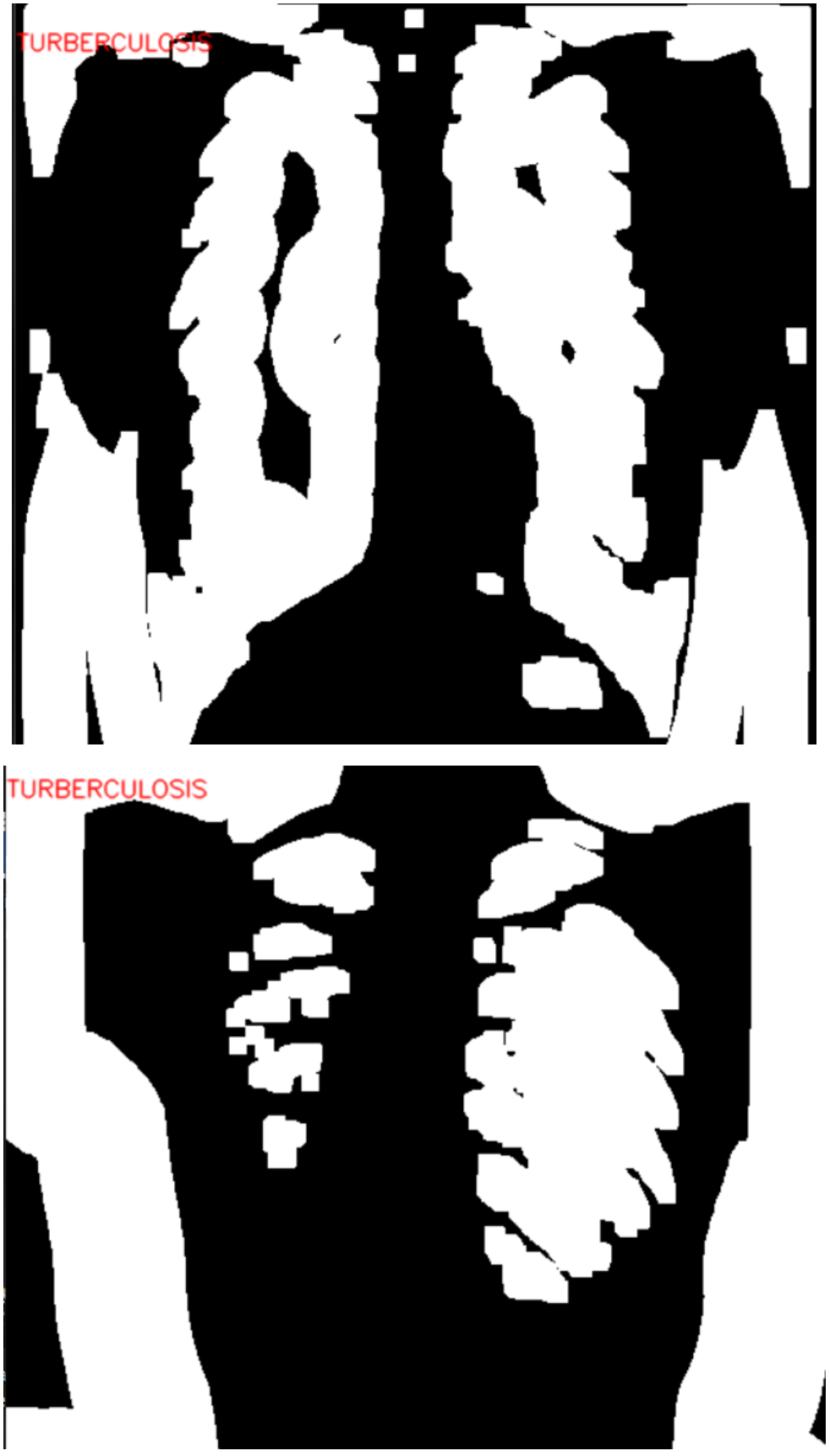
Şekil32 Grafik

```
def write_console():
    root = tk.Tk()
    root.withdraw()
    for img_path in test_dir:
        img = load_img(img_path, target_size=(img_height, img_widht))
        img_array = img_to_array(img)
        img_array = tf.expand_dims(img_array, 0)
        predictions = model.predict(img_array)
        score = tf.nn.softmax(predictions[0])
        print(img_path)
        print("Bu fotograf: {}. OLASILIK:  {:.2f}"
            .format(class_names[np.argmax(score)], 100 * np.max(score)))

        font = cv2.FONT_HERSHEY_SIMPLEX
        org = (00, 40)
        fontScale = 0.5
        color = (0, 0, 255)
        thickness = 1
        image = cv2.putText(
            cv2.imread(img_path),
            class_names[np.argmax(score)],
            org, font, fontScale, color, thickness, cv2.LINE_AA, False
        )
        cv2.imshow(img_path, image)
        cv2.waitKey()

write_console()
```

Şekil33 Konsola ve Görsel Çıkartma

Son olarak bu kod bloğunu kullanarak ekranda hangi resmin hangi hastalığa karşılık gelindiği elde edildi. Bazı örnekleri aşağıda verilmiştir.

Şekil34 PNEUMONIA Teşhisi



Şekil35 TURBERCULOSIS Teşhisi

8- GEÇMİŞ ÇALIŞTIRMALARIN ÇIKTILARINI ELDE TUTMAK İÇİN YAZILAN KOD BLOĞU

```
import os
from datetime import datetime

def get_time_string():
    now = datetime.now()
    return now.strftime("%d_%m_%Y__%H_%M_%S")


python_dir = "C:/Users/MEHTAP/AppData/Local/Programs/Python/Python39/python.exe"
current_dir = os.path.dirname(os.path.realpath(__file__))
code_dir = os.path.join(current_dir, "Mehtap.py")
log_dir = os.path.join(current_dir, get_time_string() + ".txt")
os.system(python_dir + " \"" + code_dir + "\" > \"" + log_dir)
```

Şekil36 Eski Çalıştırma Çıktılarını Tutma

Bu kod bloğu ile daha önce projeyi kaç defa çalıştırdıysak eski bilgilerini de tuttuk.

İlk fonksiyon ile mevcut zamanımızı string olarak return ettik. Mevcut zamanı nnow değişkeni içerisinde tuttuk. Strftime ile kaydettiğimiz zamanı gün, ay, yıl, saat, dakika, saniye ile return ettik.

Python_dir içerisine python derleyicisinin konumunu verdik, current_dir içerisindeki realpath ile derlenen bu dosyanın tam konumunu return ettik. Code_dir ile çalıştırılan dosaynın yolunu kaydettik, log_dir ile çalıştırma çıktılarının kaydedileceği dosyanın yolunu kaydettik. En son os.system ile kaydettiğimiz verileri kullanarak konsolu çalıştırdık.

Örneği aşağıda verilmiştir.

```
Training dataset import islemi basladi.
Found 4525 files belonging to 2 classes.
Using 3168 files for training.
Training dataset import islemi bitti. (0.21 saniye)

Validation dataset import islemi basladi.
Found 20 files belonging to 2 classes.
Using 6 files for validation.
Validation dataset import islemi bitti. (0.01 saniye)

Test dataset import islemi basladi.
Test dataset import islemi bitti. (0.00 saniye)

['PNEUMONIA', 'TURBERCULOSIS']

Model olusturma islemi basladi.
Model olusturma islemi bitti. (0.11 saniye)

Model derleme(compile) islemi basladi.
Model derleme(compile) islemi bitti. (0.00 saniye)

Model egitme(fit) islemi basladi.
 1/50 [..............................] - ETA: 5:34 - loss: 0.7233 - accuracy: 0.2969
 2/50 [>.............................] - ETA: 3:41 - loss: 2.6160 - accuracy: 0.5938
 3/50 [>.............................] - ETA: 3:37 - loss: 2.6070 - accuracy: 0.6615
 4/50 [=>............................] - ETA: 3:34 - loss: 2.0341 - accuracy: 0.7188
 5/50 [==>...........................] - ETA: 3:28 - loss: 1.7380 - accuracy: 0.7375
 6/50 [==>...........................] - ETA: 3:23 - loss: 1.5421 - accuracy: 0.7682
 7/50 [===>..........................] - ETA: 3:18 - loss: 1.3893 - accuracy: 0.7857
 8/50 [===>..........................] - ETA: 3:13 - loss: 1.2683 - accuracy: 0.7930
 9/50 [====>.........................] - ETA: 3:09 - loss: 1.1752 - accuracy: 0.7969
10/50 [=====>........................] - ETA: 3:04 - loss: 1.0776 - accuracy: 0.8109
11/50 [=====>........................] - ETA: 2:59 - loss: 0.9954 - accuracy: 0.8224
12/50 [======>.......................] - ETA: 2:55 - loss: 0.9345 - accuracy: 0.8268
13/50 [======>.......................] - ETA: 2:50 - loss: 0.8794 - accuracy: 0.8317
14/50 [=======>......................] - ETA: 2:45 - loss: 0.8424 - accuracy: 0.8281
15/50 [========>.....................] - ETA: 2:41 - loss: 0.8042 - accuracy: 0.8323
16/50 [========>.....................] - ETA: 2:36 - loss: 0.7679 - accuracy: 0.8389
17/50 [=========>....................] - ETA: 2:31 - loss: 0.7358 - accuracy: 0.8474
18/50 [=========>....................] - ETA: 2:26 - loss: 0.7015 - accuracy: 0.8550
19/50 [==========>...................] - ETA: 2:22 - loss: 0.6702 - accuracy: 0.8610
20/50 [==========>...................] - ETA: 2:17 - loss: 0.6440 - accuracy: 0.8641
21/50 [==========>...................] - ETA: 2:13 - loss: 0.6216 - accuracy: 0.8661
```

Şekil37 Eski Çalıştırma Çıktılarını Tutma

9-SONUÇ

Düzenlediğimiz veriler ile yapay zekayı eğittik. Yazdığımız fonksiyon ile verilen girdide zatürre ve tüberküloz olup olmadığını hem yazı hem de görsel ile öğrendik.

10-EKLER

KOD BLOKLARININ YAZILI HALİ AŞAĞIDA VERİLMİŞTİR

10.1ÖN HAZIRLIK AŞAMASI KODU

```
import os
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Prepocess yapılacak olan verinin dosya yolunu kaydettik.
PATH = r"C:\Users\Mehtap\Desktop\preprocess\preprocessed_data\train"

# Preprocess sonucunda kaydedilecek olan fotoğrafların boyutunu
kaydettik.
HEIGHT = 512
WIDTH = 512
IMG_SIZE = (HEIGHT, WIDTH)

# Fonksiyonu çağırırken gönderdiğimiz parametrelere göre, konumdaki
belli uzantıya sahip görselleri import eden fonksiyon
def load_images(folder_name, extension):
    image_files = []

    # Process yapılacak olan konumda, belirtilen klasöre(folder_name)
girerek içindeki görselleri tek tek dönen döngü
    for file in os.listdir(PATH + "\\" + folder_name):
        # Belli uzantılardaki(extension) dosyaları seçen if bloğu
        if file.endswith("." + extension):
            # Seçilmiş görselleri list'e kaydeden kod
            image_files.append(os.path.join(PATH, folder_name, file))

    # Seçilmiş verilerin return edilmesi
    return image_files

# Kendisine parametre olarak gönderilen görseli ekrana çıkartan
fonksiyon
def display_one(img, title1 = "Orjinal"):
    # 'img'yi ekrana çıkartılacak görsel olarak belirler ve başlık olarak
'title1' ayarlar
    plt.imshow(img), plt.title(title1)
```

```
    # Görseli x-y eksenine yerleştirir
    plt.xticks([]), plt.yticks([])
    # Görseli ekrana çıkartır
    plt.show()

# Kendisine parametre olarak gönderilen 2 görseli ekrana çıkartan
fonksiyon
def display_two(img1, img2, title1 = "Orjinal", title2 = "Düzenlenmiş"):
    # 'img1'i ekrana çıkartılacak görsel olarak belirler ve başlık olarak
'title1' ayarlar
    plt.subplot(121), plt.imshow(img1), plt.title(title1)
    # Görseli x-y eksenine yerleştirir
    plt.xticks([]), plt.yticks([])
    # 'img2'yi ekrana çıkartılacak görsel olarak belirler ve başlık olarak
'title2' ayarlar
    plt.subplot(122), plt.imshow(img2), plt.title(title2)
    # Görseli x-y eksenine yerleştirir
    plt.xticks([]), plt.yticks([])
    # Görseli ekrana çıkartır
    plt.show()

# Preprocess işlemini yöneten fonksiyon
def preprocessing(data):
    images = []
    resized_images = []
    no_noise = []
    segmented = []
    unknown = []

    #images
    # Görselleri, 'data'daki konum listesinden alır ve 'images'in içine
kaydeder
    for img in data:
        images.append(cv2.imread(img, cv2.IMREAD_UNCHANGED))

    #resized_images
    # 'images' list'inin içindeki görselleri istediğimiz boyuta göre kırpar ve
'resized_images' içine kaydeder
    for i in range(len(images)):
```

```
        # [interpolation] = Fotoğrafları her yönünden eşit olarak
boyutlandırmak için kullanılır
        res_img = cv2.resize(images[i], IMG_SIZE,
interpolation=cv2.INTER_LINEAR)
        resized_images.append(res_img)

    #no_noise
    # 'resized_images' içindeki görsellerin gürültüsünü silerek 'no_noise'
içine kaydeder
    for i in range(len(resized_images)):
        blur = cv2.GaussianBlur(resized_images[i], (5, 5), 0)
        no_noise.append(blur)

    #grayscale
    # 'no_noise' içindeki grayscale olmayan görselleri seçip grayscale
moduna çevirir
    for i in range(len(images)):
        if images[i].shape != IMG_SIZE:
            try:
                no_noise[i] = cv2.cvtColor(no_noise[i],
cv2.COLOR_RGB2GRAY)
            except:
                continue

    #segmented
    # 'no_noise' içindeki verileri segmente edip 'segmented' içine kaydeder
    for i in range(len(no_noise)):
        ret, thresh = cv2.threshold(no_noise[i], 0, 255,
cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
        segmented.append(thresh)

    #segment_bg & sure_fg
    # 'segmented' içindeki verileri; background ve foreground olarak ikiye
ayırır
    # Segmente veriden background'u kaldırarak foreground kısmını
'unknown' içine kaydeder
    for i in range(len(segmented)):
        kernel = np.ones((3, 3), np.uint8)
        opening = cv2.morphologyEx(segmented[i], cv2.MORPH_OPEN,
kernel, iterations=2)
```

```
        # [bg] = Background => Ciğerin tamamını görsel olarak kaydeder,
arkaplanı siler. Sadece ciğer kalır.
        sure_bg = cv2.dilate(opening, kernel, iterations=3)

        dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
        # [fg] = Foreground => Ciğerin gereksiz kısımlarını görsel olarak
kaydeder, arkaplanı siler. Sadece ciğerin gereksiz kısımları kalır.
        ret, sure_fg = cv2.threshold(dist_transform, 0.7 *
dist_transform.max(), 255, 0)
        sure_fg = np.uint8(sure_fg)

        # Ciğer kısmından, ciğerin gereksiz kısımlarını çıkartır. Geriye
sadece ciğerin gerekli kısmı kalmış olur.
        unkn = cv2.subtract(sure_bg, sure_fg)
        unknown.append(unkn)

    # Verilerin düzenlenmiş halini, üstüne yazarak kaydeder
    for i in range(len(images)):
        cv2.imwrite(data[i], unknown[i])


preprocessing(load_images("PNEUMONIA", "png"))
preprocessing(load_images("PNEUMONIA", "jpg"))
preprocessing(load_images("PNEUMONIA", "jpeg"))
```

10.2 MODELİ OLUŞTURMA EĞİTİM GRAFİK OLUŞUMU VE KONTROL KODLARI

```
import os
import cv2 as cv2
import numpy as np
import tkinter as tk
import tensorflow as tf
import matplotlib.pyplot as plt

from time import time
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.preprocessing import
image_dataset_from_directory
from tensorflow.keras.preprocessing.image import img_to_array,
load_img

def start_time(name):
    print(name + " islemi basladi.")
    return time()

def stop_time(start_time, name):
    print(name + " islemi bitti. ({:.2f} saniye)".format(round((time() -
start_time), 2)))


# Veri setinde bulunan train, test, validation klasörlerinin yolu
TRAIN = r"C:\Users\Mehtap\Desktop\machine learning\Mehtap Banu
Machine Learning\preprocessed_data\train"
VALID = r"C:\Users\Mehtap\Desktop\machine learning\Mehtap Banu
Machine Learning\preprocessed_data\val"
TEST  = r"C:\Users\Mehtap\Desktop\machine learning\Mehtap Banu
Machine Learning\preprocessed_data\test"

# Görsellerin boyutları ve eğitim değişkenleri
batch_size = 32
img_height = 512
```

```
img_widht = 512
noepochs = 10
epochs_range=range(noepochs)

# Train Dataset
base_time = start_time("Training dataset import")
train_ds = image_dataset_from_directory(
    TRAIN,
    validation_split = 0.3,
    subset = 'training',
    seed = 123,
    image_size = (img_height, img_widht),
    batch_size = batch_size
)
stop_time(base_time, "Training dataset import")

# Valid Dataset
base_time = start_time("Validation dataset import")
valid_ds = image_dataset_from_directory(
    VALID,
    validation_split = 0.3,
    subset = 'validation',
    seed = 123,
    image_size = (img_height, img_widht),
    batch_size = batch_size
)
stop_time(base_time, "Validation dataset import")

# Test Dataset
# Test konumundaki tüm fotoğrafların konumlarının listeye kaydedilmesi
test_dir = []
base_time = start_time("Test dataset import")
for img in os.listdir(TEST):
    test_dir.append(os.path.join(TEST, img))
stop_time(base_time, "Test dataset import")

# Sınıflar hakkında bilgi verildi
class_names = train_ds.class_names
num_classes = len(class_names)
print(class_names)
```

```
# Bir veri üzerinde çalışırken, aynı anda sıradaki verinin bellekte hazır
hale getirilmesini sağlayan kod parçacığı
AUTOTUNE = tf.data.experimental.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
valid_ds = valid_ds.cache().prefetch(buffer_size=AUTOTUNE)

# Modelin oluşturulması
# [CNN Algoritması]
base_time = start_time("Model olusturma")
model = Sequential([
   layers.experimental.preprocessing.Rescaling(1./255,
input_shape=(img_height, img_widht, 3)),
   layers.Conv2D(16,3,padding='same', activation='relu'),
   layers.MaxPooling2D(),
   layers.Conv2D(32,3, padding='same', activation='relu'),
   layers.MaxPooling2D(),
   layers.Conv2D(64,3,padding='same', activation='relu'),
   layers.MaxPooling2D(),
   layers.Conv2D(128,3,padding='same',activation='relu'),
   layers.MaxPooling2D(),
   layers.Conv2D(256,3,padding='same',activation='relu'),
   layers.MaxPooling2D(),
   layers.Dropout(0.3),
   layers.Flatten(),
   layers.Dense(256, activation='relu'),
   layers.Dense(num_classes)
])
stop_time(base_time, "Model olusturma")

# Modeli compile ederek eğitime hazır hale getirme
base_time = start_time("Model derleme(compile)")
model.compile(
   optimizer = 'adam',
   metrics=['accuracy'],
   loss=SparseCategoricalCrossentropy(from_logits=True)
)
stop_time(base_time, "Model derleme(compile)")

# Modelin fit edilmesi(eğitim/train)
```

```
base_time = start_time("Model egitme(fit)")
my_model=model.fit(
    train_ds,
    epochs = noepochs,
    validation_data = valid_ds
)
stop_time(base_time, "Model egitme(fit)")

# Eğitim sonrasında çıkan verinin değişkenlere atanması
acc = my_model.history['accuracy']
val_acc = my_model.history['val_accuracy']
loss = my_model.history['loss']
val_loss = my_model.history['val_loss']

# Son haline getirilen verinin, işleme sırasında oluşan değerlerinin
tablolaştırılması
plt.figure(figsize=(20, 20))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range,val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range,loss,label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

#write_console() fonksiyonunu konsola yazarak verdiğimiz klasördeki
resimlerin sonuçlarını hem yazılı hem görsel olarak görebiliriz.
def write_console():
    root = tk.Tk()
    root.withdraw()
    for img_path in test_dir:
        img = load_img(img_path, target_size=(img_height, img_widht))
        img_array = img_to_array(img)
        img_array = tf.expand_dims(img_array, 0)
        predictions = model.predict(img_array)
```

```
        score = tf.nn.softmax(predictions[0])
        print(img_path)
        print("Bu fotograf: {}. OLASILIK:  {:.2f}"
            .format(class_names[np.argmax(score)], 100 * np.max(score)))

        font = cv2.FONT_HERSHEY_SIMPLEX
        org = (00, 40)
        fontScale = 0.5
        color = (0, 0, 255)
        thickness = 1
        image = cv2.putText(
            cv2.imread(img_path),
            class_names[np.argmax(score)],
            org, font, fontScale, color, thickness, cv2.LINE_AA, False
        )
        cv2.imshow(img_path, image)
        cv2.waitKey()

write_console()
```

10.3 GEÇMİŞ ÇALIŞTIRMALARIN ÇIKTILARINI ELDE TUTMAK İÇİN YAZILAN KOD BLOĞU

```
import os
from datetime import datetime

# Mevcut zamanı string olarak return eden fonksiyon
def get_time_string():
    # Mevcut zamanı 'now' içine kaydettik
    now = datetime.now()
    # Kaydettiğimiz zamanı; gün_ay_yıl__saat_dakika_saniye olarak return ettik
    return now.strftime("%d_%m_%Y__%H_%M_%S")

# Python derleyicisinin konumunu kaydettik
python_dir =
"C:/Users/MEHTAP/AppData/Local/Programs/Python/Python39/python.exe"
# Mevcut konumumuzu kaydettik
# [realpath] = Derlenen bu dosyanın tam konumunu return eden sistem fonksiyonu
current_dir = os.path.dirname(os.path.realpath(__file__))
# Çalıştırılacak dosyanın yolunu kaydettik
code_dir = os.path.join(current_dir, "Mehtap.py")
# Çalıştırma çıktılarının kaydedileceği dosyanın yolunu kaydettik
log_dir = os.path.join(current_dir, get_time_string() + ".txt")

# Kaydettiğimiz verileri kullanarak konsolu çalıştırdık
os.system(python_dir + " \"" + code_dir + "\" > \"" + log_dir)
```