KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ

MÜHENDİSLİK ve MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

MAKİNE ÖĞRENMESİ

SİNDİRİM SİSTEMİ KANSER TESPİTİ

ENES ESEN – 18110131033

SAMED ZIRHLIOĞLU – 18110131037

İÇİNDEKİLER

1-GİRİŞ

Sindirim sistemi hastalıklarında kanser oranı günümüzde artmaya başladı. Bu kanserlerin artma sebeplerinden biride genetik aktarımdır. Genetik aktarım ile kanser olan insanlarda hastalığın genetik ile mi aktarıp(MSI) aktarılmadığı(MSS) kanser bölgesindeki dokunun tipi ve özellikleriyle alakalıdır.

Genetikle aktarılan hastalılarda uygulanan farklı tedaviler mevcuttur. Bu tedaviyi belirlemek için gereken süreç masraflı ve uzundur. Bu süreci en iyi değerlendirmek için Sindirim Sistemi Kanser Tespit projesini kuıllanabiliriz.

Bu masraflı ve uzun süreci en aza indirmek için Makine öğrenmesi yöntemi ile belirli hastalardan alınaan MSI ve MSS doku örnekleriyle

2-PROJENİN AMACI

Sindirim Sistemi Kanser Tespit projesi, hastalardan alınan ve daha önce belirlenmiş MSI ve MSS doku örnekleri ile eğitilen projemiz girdisi yapılan dokuyu test eder. Test sonucunda dokunun MSI mı MSS mi olduğu belirlenir. Bu şekilde masraftan ve eski yöntemle oluşan uzun sürecin önüne geçilebilir ve hasta insanların doğru bir tedavi süreci geçekleştirmeye olanak sağlar.

3-UYGULAMA VE GELİŞTİRME SÜRECİ

3.1-KULLANILACAK OLAN VERİLERİN BELİRLENMESİ

Projenin birinci aşamasında kullacağımız veri seti için kaggle'dan araştırma yaptık. Kaggle, kullanıcıların hazır veri kümeleri bulmasına ve yayınlamasına olanak sağlayan bir web sitesidir. Araştırma sonucunda belirlenen veri setini projede kullanılabilir olduğu tespit ettik. Elde ettiğimiz veri setinde MSS ve MSI klasörlere ayrılmış şekilde bulunmaktadır.

Kullanılan veri setinin linki: https://www.kaggle.com/purpleberrie/train-tcga-coad-msi-mss

3.2-VERİ SETİ DÜZENLEME SÜRECİ

Projenin ikinci aşaması, eğitim sürecinin verimli gerçekleşmesi için ön hazırlığın yapıldığı aşamadır. Ön işleme kodlarıyla veri setimizde bulunan gürültülü verileri temizledik. Ardından verileri birkaç aşamadan daha geçirerek verileri kullanılabilir hale getirdik.

3.3-KODLARIN YAZILMASI VE ÇALIŞTIRILMASI

Projenin üçüncü aşaması kodlamadır. Ön işleme ile hazırladığımız veri setini modelledik. Ardından modelimiz ile yapay zekayı eğittik. Eğitim sonucunda elde ettiğimiz veriler ile grafikler oluşturduk.

4-KULLANILAN YAZILIMLAR

Proje için Python dili ve Visual Studio Code arayüzü kullandık.

5.PROJENİN ÇALIŞMA MANTIĞI

Veri setimizden doğru sonuç almak için ön işleme aşamasından geçirdik. Bu aşamada istenilen uzantıdaki verileri grayscale olarak alarak import ettik. Daha sonra;

- Verilerin gürültüsünü sildik.
- Verileri segmente ettik.
- Verileri ön yüz ve arka yüz olarak ikiye ayırdık.
- Gereksiz olan ön yüzü, arka yüzün üzerinden kaldırıp istediğimiz veriyi elde ettik.

Yukarıdaki işlemler sırasında verileri görselleştirmek için bir fonksiyon yazdık.

Ön hazırlık ile hazır hale getirdiğimiz verileri import ettik. Ardından import ettiğimiz verileri modelledik. Verileri modellerken CNN kodu kullandık. Daha sonra modelimizi compile ederek eğitime hazır hale getirdik. Yapay zekayı eğittik.

Eğittiğimiz veriyi test etmek için ayrıyeten bir fonksiyon yazdık. Bu fonkiyonu konsoldan çağırarak bir girdi vericeğiz. Girdinin mss mi yoksa msi mı olduğunu böylece öğrenebiliriz.

İstediğimiz aralıktaki kod bloklarının ne kadar sürede çalıştığını öğrenmek için kronometre görevi gören fonksiyonları yazdık.

6-ÖN HAZIRLIK AŞAMASI

```
7   DATASET = r"E:\\makine_ogrenmesi\\data"
8   TEST = DATASET + "\\test"
9   TRAIN = DATASET + "\\train"
10  VALIDATION = DATASET + "\\val"
```

Şekil1-Yol belirtme

Ön işleme yapacağımız verinin dosya yolu verildi.

```
19  def import_images(folder_path, target_folder, extension=".jpg"):
20      paths = []
21      images = []
22      for file in os.listdir(folder_path + "\\" + target_folder):
23          if file.endswith(extension):
24              image_path = os.path.join(folder_path, target_folder, file)
25              images.append(cv2.imread(image_path, cv2.IMREAD_GRAYSCALE))
26              paths.append(image_path)
27
28      return images, paths
```

Şekil2-Görsel import etme

Parametre olarak aldığı veri seti konumu, sınıf ismi ve dosya uzantısını kullanarak istenilen görselleri grayscale modunda import edip; bu görselleri ve dosya yollarını return eden fonksiyon.

```python
def process(dataset, paths):
    images = []

    for image in dataset:
        blured_image = cv2.GaussianBlur(image, (5, 5), 0)
        ret, segmented_image = cv2.threshold(blured_image, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

        ones = np.ones((3, 3), np.uint8)
        morph = cv2.morphologyEx(segmented_image, cv2.MORPH_OPEN, ones, iterations=2)
        background = cv2.dilate(morph, ones, iterations=3)

        d_trans = cv2.distanceTransform(morph, cv2.DIST_L2, 5)
        ret, foreground = cv2.threshold(d_trans, 0.7 * d_trans.max(), 255, 0)

        foreground = np.uint8(foreground)
        images.append(cv2.subtract(background, foreground))

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

Şekil3-Ön işleme

Görselleri ve dosya yollarını parametre olarak alan, bu görsellere ön işleme yapan ve işlenmiş görselleri kaydeden fonsiyondur. Bu fonksiyonun yaptığı ön işleme aşamaları;

- Verilerin gürültüsünü sildik.
- Verileri segmente ettik.
- Verileri ön yüz ve arka yüz olarak ikiye ayırdık.
- Gereksiz olan ön yüzü, arka yüzün üzerinden kaldırıp istediğimiz veriyi elde ettik.

```python
def main():
    start_time = start_timer("Validation MSI")
    images, paths = import_images(VALIDATION, "MSI")
    calculate_time(start_time, "Validation MSI")
    process(images, paths)
```

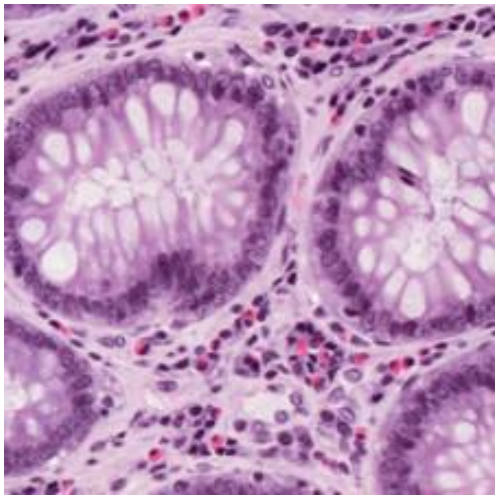
Şekil4-Ana fonksiyon

Ön hazırlık dosyasında oluşturduğumuz fonksiyonları ana fonksiyonda çağırdık.

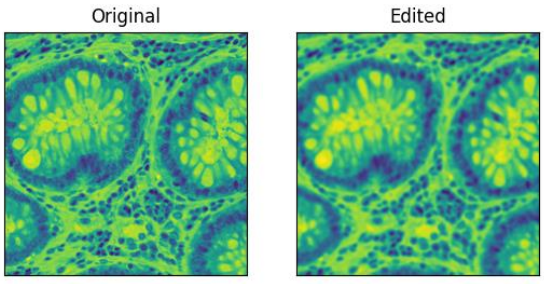
```
30  def show_images(image_1, image_2, header_1="Original", header_2="Edited"):
31      plt.subplot(121)
32      plt.imshow(image_1)
33      plt.title(header_1)
34      plt.xticks([])
35      plt.yticks([])
36      plt.subplot(122)
37      plt.imshow(image_2)
38      plt.title(header_2)
39      plt.xticks([])
40      plt.yticks([])
41      plt.show()
```

Şekil5-Ön işleme görselleştirme

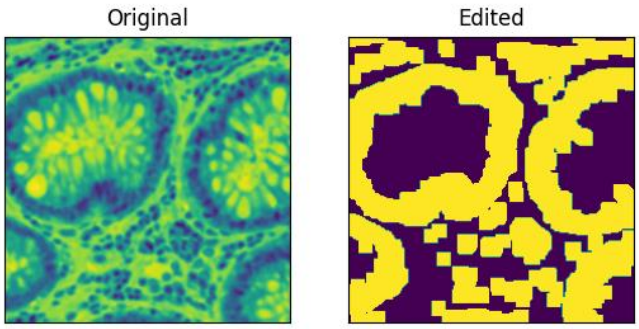
Ön işleme aşamasındaki çıktıları görselleştirmek için yazılan fonksiyon.



Şekil6-Orijinal veri



Şekil7-Gürültü silme işlemi



Şekil8-Segmentasyon işlemi

Şekil9-Düzenlenmiş veri

7-MODELİ OLUŞTURMA EĞİTİM GRAFİK OLUŞUMU VE KONTROL KODLARI

```
20  # ZAMAN KONTROL FONKSİYONLARI
21  start_time = 0
22  def start_timer(process_name='Process'):
23      print(process_name + ' Started')
24      start_time = time()
25
26  def stop_timer(process_name='Process'):
27      print(process_name + " Finished ({:.2f} seconds)".format(round((time() - start_time), 2)))
```

Şekil10-Zaman kontrol fonksiyonları

İstediğimiz aralıktaki kod bloklarının ne kadar sürede çalıştığını öğrenmek için kronometre görevi gören fonksiyonlar. Mevcut zamanı start_timer() fonksiyonu sayesinde start_time değişkenine atıyoruz. Daha sonra ilk zamandan şimdiki zamana kadar geçen süreyi hesaplamak için şimdiki zamandan, ilk kaydettiğimiz zamanı çıkartarak hesaplıyoruz.

```
38  # VERİSETİ KONUMU
39  DATASET_DIR = r"E:\makine_ogrenmesi\preprocessed_data"
40  TEST_DIR = DATASET_DIR + "\\test\\"
41  TRAIN_DIR = DATASET_DIR + "\\train\\"
42  VALIDATION_DIR = DATASET_DIR + "\\val\\"
43
44  start_timer('Dataset Importing')
45  # TRAIN DATA IMPORT
46  train_data_generator = ImageDataGenerator(
47      validation_split = 0.2,
48      preprocessing_function = preprocess_input
49  )
50  train_data = train_data_generator.flow_from_directory(
51      TRAIN_DIR,
52      target_size = IMG_SIZE,
53      shuffle = True,
54      seed = SEED,
55      class_mode = 'categorical',
56      color_mode = 'grayscale',
57      batch_size = BATCH_SIZE,
58      subset='training'
59  )
```

Şekil11-Veri yolu belirtilmesi ve data importu

Veri setinin yolunu ve alt klasörlerini define yöntemiyle tanımladık. Daha sonra tensorflow'a ait keras kütüphanesinin preprocess fonksiyonlarından birini kullanarak, veri setimizin %20'sinin validation verisi olacağını belirttik. Sonrasında veri setimizin yolunu, görsel boyutunu ve renk modu gibi çeşitli parametreleri belirterek verimizin import edilmesini sağladık.

```
93   classes = list(train_data.class_indices.keys())
94   num_classes = len(classes)
95
96   model = Sequential()
97   model.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=IMG_SHAPE))
98   model.add(MaxPooling2D(pool_size=(2, 2)))
99   model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
100  model.add(MaxPooling2D(pool_size=(2, 2)))
101  model.add(Dropout(0.2))
102  model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
103  model.add(MaxPooling2D(pool_size=(2, 2)))
104  model.add(Flatten())
105  model.add(Dense(100, activation='relu'))
106  model.add(Dropout(0.2))
107  model.add(Dense(num_classes, activation='softmax'))
108  model.summary()
109
110  model.compile(
111      loss = 'categorical_crossentropy',
112      optimizer = Adam(),
113      metrics = ['accuracy']
114  )
```

Şekil12-CNN algoritması ve compile işlemi

Veri setimizdeki sınıfların isimleri ve sayısını değişkenlere atadık. Ardından modelimizi oluşturmak için CNN algoritması kullandık. Daha sonra oluşturduğumuz modeli adam optimizasyon yöntemi kullanarak derledik.

```
124  start_timer('Training')
125  hist = model.fit(
126      train_data,
127      steps_per_epoch = train_data.samples // BATCH_SIZE,
128      epochs = EPOCHS,
129      validation_data = valid_data,
130      verbose = 1,
131      validation_steps = valid_data.samples // BATCH_SIZE
132  )
133  stop_timer('Training')
```

Şekil13-Eğitim

Train ve valid verilerimizi belirterek modelimiz eğittik. Bu eğitimin boyutunu belirtmek için mevcut veri sayımızı daha önceden belirlediğimiz BATCH_SIZE değerine böldük. Tüm bu eğitim sürecinin ne kadar sürdüğünü öğrenmek için start_timer() fonksiyonunu kullandık.

```
Epoch 1/5
138/138 [==============================] - 209s 2s/step - loss: 0.7988 - accuracy: 0.5013 - val_loss: 0.6952 - val_accuracy: 0.3906
Epoch 2/5
138/138 [==============================] - 202s 1s/step - loss: 0.6928 - accuracy: 0.5131 - val_loss: 0.6974 - val_accuracy: 0.3906
Epoch 3/5
138/138 [==============================] - 205s 1s/step - loss: 0.6926 - accuracy: 0.5127 - val_loss: 0.7011 - val_accuracy: 0.3906
Epoch 4/5
138/138 [==============================] - 204s 1s/step - loss: 0.6920 - accuracy: 0.5137 - val_loss: 0.7007 - val_accuracy: 0.4688
Epoch 5/5
138/138 [==============================] - 208s 2s/step - loss: 0.6898 - accuracy: 0.5261 - val_loss: 0.6942 - val_accuracy: 0.5143
```

Şekil14-Eğitim 2

```python
def recogout():
    root=tk.Tk()
    root.withdraw()
    #img_path = filedialog.askopenfilename()
    image_paths = os.listdir(TEST_DIR)
    for image_path in image_paths:
        img=load_img(os.path.join(TEST_DIR, image_path), color_mode = 'grayscale', target_size=IMG_SIZE)
        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(image_path + " This image most likely belongs to {}"
            .format(classes[np.argmax(score)]))
```

Şekil15-Test fonksiyonu

Veri setimizdeki test klasörünün içinde yer alan görselleri, eğittiğimiz modelimize tâbi tutarak hangi sınıfa ait olabileceğini tahmin etmesini sağlana fonksiyon.

```
MSI (10).jpg This image most likely belongs to MSI
MSI (11).jpg This image most likely belongs to MSI
MSI (12).jpg This image most likely belongs to MSI
MSI (13).jpg This image most likely belongs to MSI
MSI (14).jpg This image most likely belongs to MSI
MSI (15).jpg This image most likely belongs to MSI
```

Şekil16-Test fonksiyonu 2

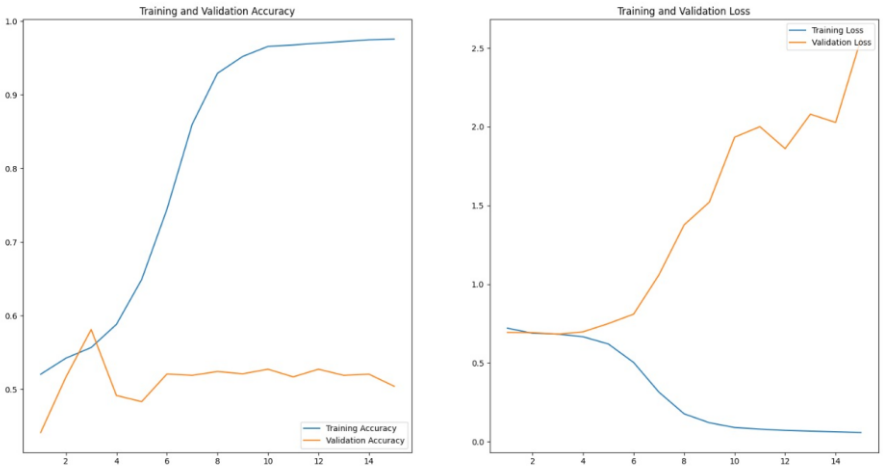
```python
accuracy = np.array(hist.history['accuracy'])
val_accuracy = np.array(hist.history['val_accuracy'])
loss = np.array(hist.history['loss'])
val_loss = np.array(hist.history['val_loss'])

plt.figure(figsize=(20, 20))
plt.subplot(1, 2, 1)
plt.plot(RANGE_EPOCH, accuracy, label='Training Accuracy')
plt.plot(RANGE_EPOCH,val_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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

Şekil17-Grafik

Eğitim sonucunda çıkan değerleri belirli değişkenlere atadık. Bu değişkenleri kullanarak grafik oluşturduk.



Şekil18-Grafik sonucu

8-SONUÇ

Ön hazırlık ile verileri düzenledik. Ardından düzenlediğimiz veriler ile yapay zekayı eğittik. Yapay zekayı eğittikten sonra yazdığımız bir kod bloğu ile girdinin MSS mi MSI mı olduğunu tahmin ettik.

9-EKLER

9.1-ÖN HAZIRLIK AŞAMASI KODU

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


DATASET = r"E:\\data"
TEST = DATASET + "\\test"
TRAIN = DATASET + "\\train"
VALIDATION = DATASET + "\\val"


def start_timer(name):
    print(name + " Data Importing Started")
    return time.time()


def calculate_time(start_time, name):
    print(name + " Data Importing Finished ({:.2f} seconds)".format(round((time.time() - start_time), 2)))


def import_images(folder_path, target_folder, extension=".jpg"):
    paths = []
    images = []
    for file in os.listdir(folder_path + "\\" + target_folder):
        if file.endswith(extension):
            image_path = os.path.join(folder_path, target_folder, file)
            images.append(cv2.imread(image_path, cv2.IMREAD_GRAYSCALE))
            paths.append(image_path)
```

```
    return images, paths


def show_images(image_1, image_2, header_1="Original", header_2="Edited"):
    plt.subplot(121)
    plt.imshow(image_1)
    plt.title(header_1)
    plt.xticks([])
    plt.yticks([])
    plt.subplot(122)
    plt.imshow(image_2)
    plt.title(header_2)
    plt.xticks([])
    plt.yticks([])
    plt.show()


def process(dataset, paths):
    images = []


    for image in dataset:


        blured_image = cv2.GaussianBlur(image, (5, 5), 0)
        ret, segmented_image = cv2.threshold(blured_image, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
        show_images(image, blured_image)


        ones = np.ones((3, 3), np.uint8)
        morph = cv2.morphologyEx(segmented_image, cv2.MORPH_OPEN, ones, iterations=2)
        background = cv2.dilate(morph, ones, iterations=3)
        show_images(blured_image, background)


        d_trans = cv2.distanceTransform(morph, cv2.DIST_L2, 5)
        ret, foreground = cv2.threshold(d_trans, 0.7 * d_trans.max(), 255, 0)


        foreground = np.uint8(foreground)
        show_images(background, cv2.subtract(background, foreground))
```

```
        images.append(cv2.subtract(background, foreground))

    for i in range(len(dataset)):
        cv2.imwrite(paths[i], images[i])

def main():
    start_time = start_timer("Validation MSI")
    images, paths = import_images(VALIDATION, "MSI")
    calculate_time(start_time, "Validation MSI")
    process(images, paths)

    start_time = start_timer("Validation MSS")
    images, paths = import_images(VALIDATION, "MSS")
    calculate_time(start_time, "Validation MSS")
    process(images, paths)

    start_time = start_timer("Test MSI")
    images, paths = import_images(TEST, "MSI")
    calculate_time(start_time, "Test MSI")
    process(images, paths)

    start_time = start_timer("Test MSS")
    images, paths = import_images(TEST, "MSS")
    calculate_time(start_time, "Test MSS")
    process(images, paths)

    start_time = start_timer("Train MSI")
    images, paths = import_images(TRAIN, "MSI")
    calculate_time(start_time, "Train MSI")
    process(images, paths)

    start_time = start_timer("Train MSS")
    images, paths = import_images(TRAIN, "MSS")
```

```
    calculate_time(start_time, "Train MSS")
    process(images, paths)


main()
```

9.2-MODELİ OLUŞTURMA EĞİTİM GRAFİK OLUŞUMU VE KONTROL KODLARI

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense,  Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.applications.inception_resnet_v2 import preprocess_input
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
import tkinter as tk
import tensorflow as tf

import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from time import time
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Flatten,Dense
from keras.preprocessing.image import load_img, img_to_array
import os

# ZAMAN KONTROL FONKSİYONLARI
start_time = 0
def start_timer(process_name='Process'):
    print(process_name + ' Started')
    start_time = time()

def stop_timer(process_name='Process'):
```

```
    print(process_name + " Finished ({:.2f} seconds)".format(round((time() - start_time), 2)))

# DEFINE
SEED = 10
EPOCHS = 15
RANGE_EPOCH = range(1, EPOCHS + 1)

BATCH_SIZE = 64
IMG_HEIGHT = 224
IMG_WIDTH = 224
IMG_SIZE = (IMG_HEIGHT, IMG_WIDTH)
IMG_SHAPE = (IMG_HEIGHT, IMG_WIDTH, 1)

# VERİSETİ KONUMU
DATASET_DIR = r"E:\finished_projects\machine_learning\samed_enes\preprocessed_data"
TEST_DIR = DATASET_DIR + "\\test\\"
TRAIN_DIR = DATASET_DIR + "\\train\\"
VALIDATION_DIR = DATASET_DIR + "\\val\\"

start_timer('Dataset Importing')
# TRAIN DATA IMPORT
train_data_generator = ImageDataGenerator(
    validation_split = 0.2,
    preprocessing_function = preprocess_input
)
train_data = train_data_generator.flow_from_directory(
    TRAIN_DIR,
    target_size = IMG_SIZE,
    shuffle = True,
    seed = SEED,
    class_mode = 'categorical',
    color_mode = 'grayscale',
    batch_size = BATCH_SIZE,
    subset='training'
```

```
)

# TRAIN DATA IMPORT
val_data_generator = ImageDataGenerator(
    preprocessing_function = preprocess_input,
    validation_split = 0.2
)
valid_data = val_data_generator.flow_from_directory(
    VALIDATION_DIR,
    target_size = IMG_SIZE,
    shuffle = False,
    seed = SEED,
    class_mode = 'categorical',
    color_mode = 'grayscale',
    batch_size = BATCH_SIZE,
    subset = 'validation'
)

# TEST DATA IMPORT
test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
test_data = test_generator.flow_from_directory(
    TEST_DIR,
    target_size = IMG_SIZE,
    shuffle = False,
    seed = SEED,
    class_mode = 'categorical',
    color_mode = 'grayscale',
    batch_size = BATCH_SIZE
)
stop_timer('Dataset Importing')

classes = list(train_data.class_indices.keys())
num_classes = len(classes)
```

```
model = Sequential()
model.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=IMG_SHAPE))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

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

start_timer('Training')
hist = model.fit(
    train_data,
    steps_per_epoch = train_data.samples // BATCH_SIZE,
    epochs = EPOCHS,
    validation_data = valid_data,
    verbose = 1,
    validation_steps = valid_data.samples // BATCH_SIZE
)
stop_timer('Training')

accuracy = np.array(hist.history['accuracy'])
val_accuracy = np.array(hist.history['val_accuracy'])
loss = np.array(hist.history['loss'])
```

```
val_loss = np.array(hist.history['val_loss'])

plt.figure(figsize=(20, 20))
plt.subplot(1, 2, 1)
plt.plot(RANGE_EPOCH, accuracy, label='Training Accuracy')
plt.plot(RANGE_EPOCH,val_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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

def recogout():
    root=tk.Tk()
    root.withdraw()
    #img_path = filedialog.askopenfilename()
    image_paths = os.listdir(TEST_DIR)
    for image_path in image_paths:
        img=load_img(os.path.join(TEST_DIR, image_path), color_mode = 'grayscale', target_size=IMG_SIZE)
        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(image_path + " This image most likely belongs to {}"
            .format(classes[np.argmax(score)]))

recogout()
```