T.C.
DOKUZ EYLÜL ÜNİVERSİTESİ
YÖNETİM BİLİŞİM SİSTEMLERİ

SLEEP DETECTOR

YASİN SEVEN

2017469044

MUHAMMET OĞUZHAN YILMAZ

2018469063

OĞULCAN GALATA

2017469021

YAPAY ZEKA

DR. KUTAN KORUYAN

OCAK 2022

PROJENİN AMACI

Bu projenin amacı, yapay zeka teknikleri kullanılarak kameradan gelen görüntüdeki insanın gözünün açık ya da kapalı olduğunu tespit eden sistemi geliştirmektir. Program, ders esnasında veya online sınav esnasında kullanıcının kim olduğunu tanımlayan ve gözünün kapalı olup olmadığını tespit ederek, bir metin dosyasına geçmiş kayıtları tutmak için tasarlanmıştır.

PROJENİN ÖZETİ

Bu proje temel olarak iki kısımdan oluşmaktadır. İlk kısımda Kaggle'dan alınmış yaklaşık 1400 açık ve kapalı göz resmine sahip bir veri seti, kişinin gözlerinin açık veya kapalı olup olmadığını sınıflandıracak bir derin öğrenme modelini beslemek için kullanılmıştır. Bu model projenin ikinci kısmında kullanılmak üzere kaydedilmiştir. İkinci kısımda ise bir kamera kaydı başlatılarak görüntüdeki kişilerin daha önce tanımlanmış kişilerden olup olmadığı, kişinin yüzünün ve gözlerinin konumunun anlık olarak tespit edilmesi daha sonrasında ise kişinin gözlerinin açık ya da kapalı olup olmadığı tespit edilerek, program sonunda metin dosyasına aktarılması sağlanmıştır.

Kaynak kodu: https://github.com/svnyasin/sleep_detector

PROJEDE KULLANILAN TEKNOLOJİLER

Kullanılan Programlar: Microsoft Visual Studio Code

Kullanılan Programlama Dilleri: Python

Kullanılan Kütüphaneler:

- **face_recognation:** Yüz tanıma işlemini basitleştirmek için geliştirilmiş bir python kütüphanesi.
- **Keras:** Neredeyse her tür derin öğrenme modelini tanımlamak ve eğitmek için uygun bir yol sağlayan Python için bir derin öğrenme kütüphanesidir. Keras, Tensorflow , Theano ve CNTK üzerinde çalışabilen Python ile yazılmış bir üst düzey sinir ağları API'sıdır.
- **OpenCV:** OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir.
- **NumPy:** Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kitaplıktır.

PROJE YAPIM AŞAMALARI

KISIM 1

Adım 1: Dataset %80 train, %20 test olarak tanımlanır. Ve sınıflar tespit edilir.

```
def generator(dir, gen=image.ImageDataGenerator(rescale=1./255), shuffle=True,batch_size=1,target_size=(24,24),class_mode='categorical' ):

    return gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,color_mode='grayscale',class_mode=class_mode,target_size=target_size)



# CNN YAPAY SİNİR AĞLARININ GÖRÜNTÜ İŞLEME İÇİN KULLANILAN ÖZEL BİR DALIDIR.

# DATASET TANIMLANIR
BS= 32
TS=(24,24)
train_batch= generator('data/train',shuffle=True, batch_size=BS,target_size=TS)
valid_batch= generator('data/test',shuffle=True, batch_size=BS,target_size=TS)
SPE= len(train_batch.classes)//BS
VS = len(valid_batch.classes)//BS
print(SPE,VS)
```

Adım 2: Kullandığımız model, Evrişimli Sinir Ağları (CNN) kullanılarak Keras ile oluşturulmuştur. Bir evrişimli sinir ağı, görüntü sınıflandırma amaçları için son derece iyi performans gösteren özel bir derin sinir ağı türüdür. Bir CNN temel olarak bir girdi katmanı, bir çıktı katmanı ve birden çok katmana sahip olabilen bir gizli katmandan oluşur. Katman ve filtre üzerinde 2B matris çarpımı gerçekleştiren bir filtre kullanılarak bu katmanlar üzerinde bir evrişim işlemi gerçekleştirilir. CNN model mimarisi aşağıdaki katmanlardan oluşur:

Evrişimsel katman; 32 düğüm, çekirdek boyutu 3

- Evrişimsel katman; 32 düğüm, çekirdek boyutu 3
- Evrişimsel katman; 64 düğüm, çekirdek boyutu 3
- Tam bağlantılı katman; 128 düğüm

Son katman ayrıca 2 düğümlü tam bağlantılı bir katmandır. Softmax kullandığımız çıktı katmanı dışındaki tüm katmanlarda bir Relu aktivasyon işlevi kullanılır.

```
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
    MaxPooling2D(pool_size=(1,1)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#32 convolution filters used each of size 3x3
#again
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1,1)),

#64 convolution filters used each of size 3x3
#choose the best features via pooling

#randomly turn neurons on and off to improve convergence
    Dropout(0.25),
#flatten since too many dimensions, we only want a classification output
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
#one more dropout for convergence' sake :)
    Dropout(0.5),
#output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])
```

Adım 3: Model derlenir ve eğitim işlemi gerçekleştirilir. Ardından ikinci kısımda kullanılmak üzere cnnCat2.h5 dosyası olarak kaydedilir.

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

print(model)

model.fit_generator(train_batch, validation_data=valid_batch,epochs=25,steps_per_epoch=SPE ,validation_steps=VS)

model.save('models/cnnCat2.h5', overwrite=True)
```

KISIM 2

Adım 1: Programın açılış saati loglama işlemi için bir değişkene atılır.

```
now = datetime.now()
start_time_str = now.strftime("%H:%M:%S")
start_time = datetime.strptime(start_time_str, "%H:%M:%S")
```

Adım 2: Sağ ve sol gözün tespit edilebilmesi için OpenCV tarafından daha önceden hazırlanmış XML dosyaları tanımlanır.

```
#SAĞ VE SOL GÖZÜ TESPİT EDEN OPENCV XML DOSYALARI
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')

lbl=['Close','Open']
```

Adım 3: Kısım 1'de kaydettiğimiz cnncat2.h5 modelimizi kullanmak üzere tanımlıyoruz. Ve diğer gerekli değişkenleri tanımlıyoruz

```
#MODEL YÜKLEMESİ
model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]
name = "Unknown"
```

Adım 4: Tanınan yüzlerin gelen görüntüde tespit edilebilmesi için face_recognation kütüphanesi ile kişi resimleri eklenir.

```
#TANINAN YÜZLERİN EKLENMESİ
yasin_seven_image = face_recognition.load_image_file("yasin_seven.jpg")
yasin_seven_face_encoding = face_recognition.face_encodings(yasin_seven_image)[0]

ogulcan_galata_image = face_recognition.load_image_file("ogulcan_galata.jpeg")
ogulcan_galata_face_encoding = face_recognition.face_encodings(ogulcan_galata_image)[0]

oguzhan_yilmaz_image = face_recognition.load_image_file("oguzhan_yilmaz.jpeg")
oguzhan_yilmaz_face_encoding = face_recognition.face_encodings(oguzhan_yilmaz_image)[0]

#TANINAN YÜZLERİN LİSTESİ
known_face_encodings = [
    yasin_seven_face_encoding,
    ogulcan_galata_face_encoding,
    oguzhan_yilmaz_face_encoding

]

#TANINAN YÜZLERİN TEXTLERİ
known_face_names = [
    "Yasin Seven",
    "Ogulcan Galata",
    "Oguzhan Yilmaz"
]
```

Adım 5: Sürekli çalışacak bir döngü başlatılarak, kameradan gelen her frame için döngü içindeki kodlar çalıştırılır. Gelen framede ilk olarak yüzün ve gözlerin konumu tespit edilir ve tespit edilen yüzler tanınan yüzlerle karşılaştırılır eğer tanınmıyorsa 'unknown' olarak tanımlanır.

```
while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()

    height,width = frame.shape[:2]


    left_eye = leye.detectMultiScale(frame)
    right_eye =  reye.detectMultiScale(frame)

    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown"

            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    process_this_frame = not process_this_frame
```

Adım 6: Tespit edilen yüzlerin ve gözlerin etrafında dikdörtgenler çizilir ve yüzün altına kişinin kim olduğu yazılır. Yeri tespit edilen gözlerin kısım 1'de oluşturulan model ile açık mı yoksa kapalı mı olduğunun tahminlemesi yapılır. Açık ise "1", kapalı ise "0" çıktısı alınır.

```
# Display the results
for (top, right, bottom, left), name in zip(face_locations, face_names):
    # Scale back up face locations since the frame we detected in was scaled to 1/4 size
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    # Draw a box around the face
    cv2.rectangle(frame, (left, top), (right, bottom ), (0, 0, 255), 2)

    # Draw a label with a name below the face
    cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, name, (left + 6, bottom - 6), font, 0.7, (255, 255, 255), 1)

for (x,y,w,h) in right_eye:
    cv2.rectangle(frame, (x,y) , (x+w,y+h) , (0,255,0) , 2 )

    r_eye=frame[y:y+h,x:x+w]
    count=count+1
    r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
    r_eye = cv2.resize(r_eye,(24,24))
    r_eye= r_eye/255
    r_eye=  r_eye.reshape(24,24,-1)
    r_eye = np.expand_dims(r_eye,axis=0)
    rpred_x = model.predict(r_eye)
    rpred=np.argmax(rpred_x,axis=1)
    if(rpred[0]==1):
        lbl='Open'
    if(rpred[0]==0):
        lbl='Closed'
    break
```

Adım 7: Kişinin gözü kapalı ise “score” isimli değişken arttırılmaya başlanır. Açık ise azaltılır. Eğer “score” 24’ü geçer ise kullanıcının uyuduğu ya da kopya çektiği varsayılır ve kanıt için resim çekilir. Burada eşik skorunun 24 verilmesinin sebebi kullanıcıya hata payı bırakmaktır.

```
if(rpred[0]==0 and lpred[0]==0):
    if (score <= 30):
        score=score+1
    print(name, " ", str(score))
else:
    if(score<=0):
        score=0
        print(name, " ", str(score))

    else:
        score=score-2
        print(name, " ", str(score))
```

```
if(score>24):

    if(isFirst):
        top_uyuma_sayisi=top_uyuma_sayisi+1
        print(top_uyuma_sayisi)
        isFirst=False

        #KANIT İÇİN RESİM ÇEK
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            print("uyuma")


        except:
            pass
```

Adım 8: Son olarak program kapatılırken sistemin açık kalma süresinin hesaplanabilmesi için kapanış saati kaydedilir ve gerekli hesaplamalar yapıldıktan sonra elde edilen bilgiler log dosyasına yazılarak program sonlandırılır.

```
    # Display the resulting image
    cv2.imshow('Video', frame)

    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        now = datetime.now()
        finish_time = now.strftime("%H:%M:%S")
        finish_time = datetime.strptime(finish_time, "%H:%M:%S")
        sure = str(finish_time-start_time)
        f = open("log.txt", "a")
        f.write("\n-----------------------------------------------

        f.write("\nIsim :" + name + "\nSistem baslangic zamani "+
        print(sure)
        break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```

SONUÇ

Sonuç olarak bu projeyi yaparken bir yapay zeka projesinin bütünüyle nasıl geliştirilebileceğini bir projenin nasıl planlanabileceğini ve takım olarak nasıl çalışılabileceğini tecrübe etmiş olduk. Bu proje eğer gerekli geliştirmeler ve entegrasyon yapılır ise canlı sınav sistemlerinde kopyayı önlemek için, online ya da yüz yüze olan derslerde öğrencilerin öğretmeni ne kadar dikkatli dinlediğini ölçmek için veya uzun yolculuklarda şoförün uyuklama durumunu tespit etmek için kullanılabilir.