Drowsiness Detection System

COMPUTER VISION INTERMEDIATE LIBRARIES PYTHON

This article was published as a part of the Data Science Blogathon.

Introduction on Drowsiness Detection System

Nowadays drowsiness of drivers is one of the main reasons behind road accidents. It is natural for the drivers who take long drives to doze off behind the steering wheel. In this article, we will build a drowsiness detection system that will alert the driver as soon as he fell asleep.

Drowsiness is identified by using vision-based techniques like eyes detection, yawning, and nodding. When it comes to yawning and nodding some people can sleep without yawning and nodding.

One more method is by using physiological sensors like biosensors. Here the disadvantages are like the driver may hesitate to wear them or he may forget to wear them. Detecting drowsiness through eye detection is best compared to the remaining techniques.

Let's get started.

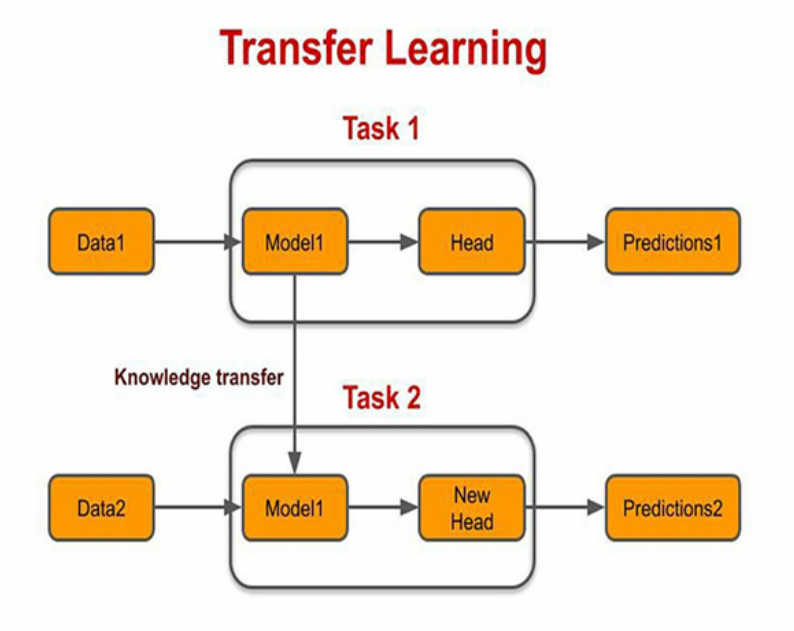
Requirements

●Python
●OpenCV: OpenCV is a great tool for image processing and performing many computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, object tracking, and many more tasks.
● TensorFlow: Tensorflow is a free and open-source library, developed by the Google Brain team for machine learning and artificial intelligence. Tensorflow has a particular focus on the training and inference of deep neural networks.
● Keras: Keras is an open-source software library and it provides a Python interface for artificial neural networks. Keras is more user-friendly because it is an inbuilt python library.

Transfer Learning

In this project, we will use transfer learning to build the model. Transfer Learning is a machine learning method where we use a pre-trained model for a new model with the related problem statement.

For example, a model that is used for the recognization of cars can be used for the recognization of trucks.

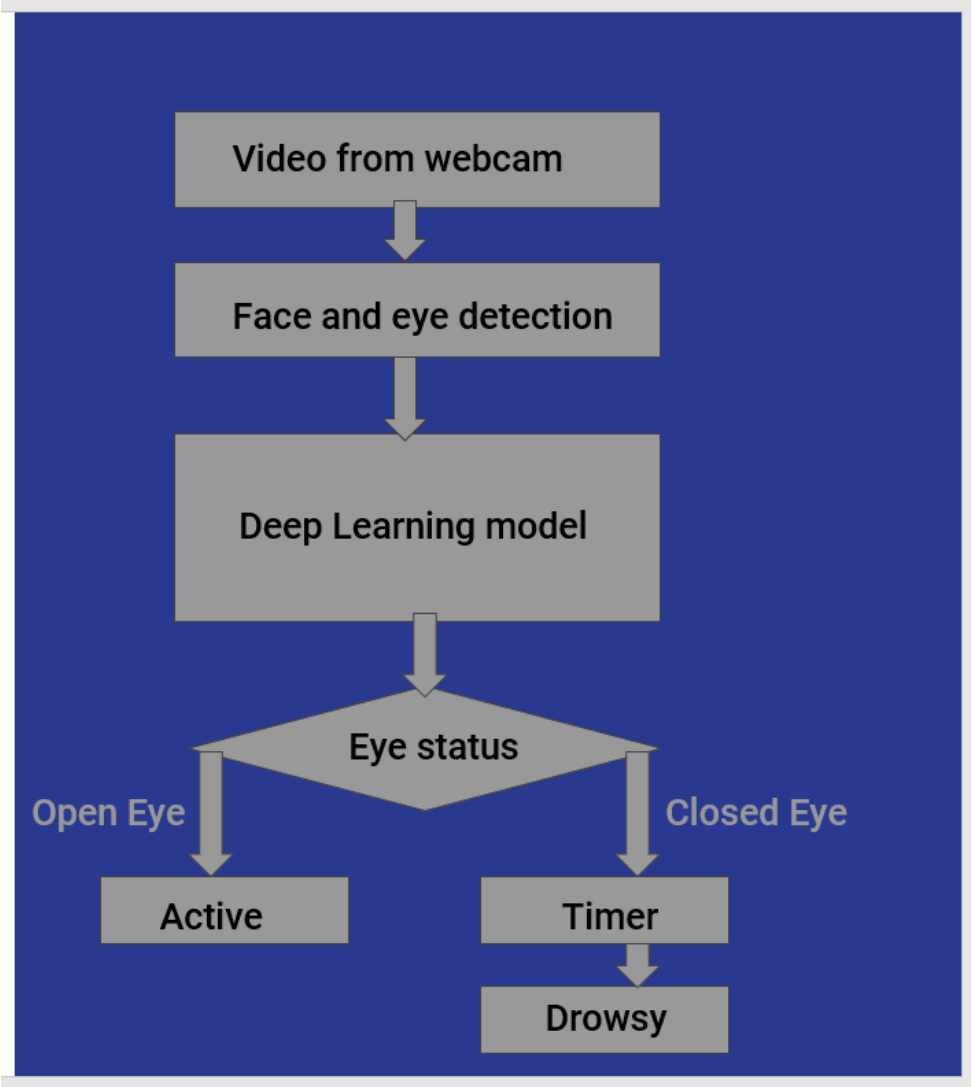


Here mainly it focuses on the knowledge that is gained while solving one problem and it applies to a different but related problem.

Methodology

The methodology of this project is the first video is captured using a webcam and from the video first face is detected using the Harcascade algorithm and then the eyes are detected. Then we use our deep learning model which is built using transfer learning to know the status of the eye. If it is an open eye then it will say Active and if it is a closed eye then it will check for a few seconds and then it will say the driver is drowsy and will beep an alarm.

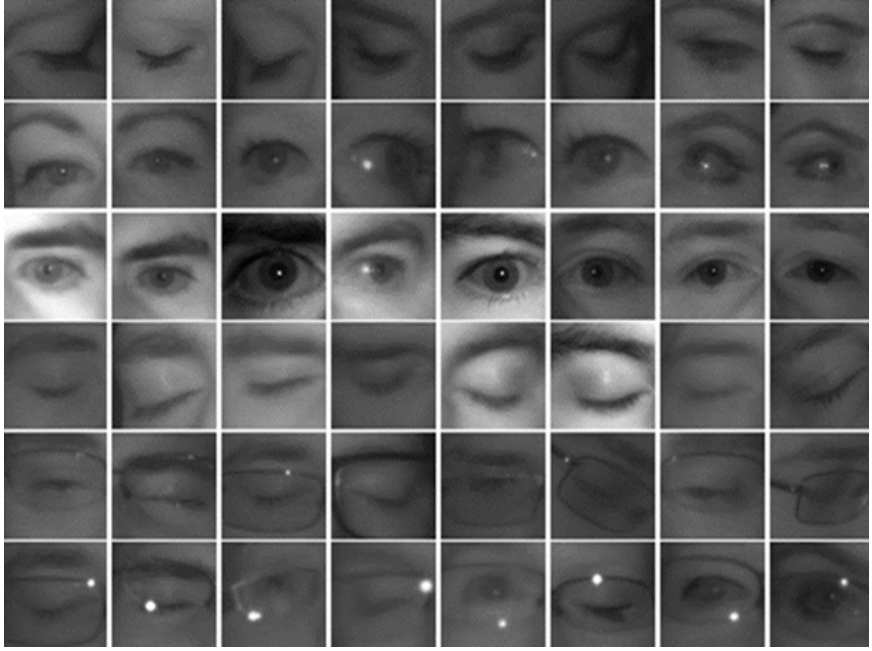
We will use Python, OpenCV, TensorFlow, and Keras to build a system that can detect the closed eyes of drivers and alert them if ever they fall asleep while driving. If the driver's eyes are closed, this system will immediately inform the driver. OpenCV that we are going to use now will monitor and collect the driver's images via a webcam that was attached and feed them into the deep learning model and then the model will classify the driver's eyes as 'open' or 'closed.'



Dataset

For this project, we are going to use the MRL Eye dataset. MRL Eye dataset is a large-scale dataset that contains human eye images. Download the dataset using this link. http://mrl.cs.vsb.cz/eyedataset

This dataset contains 84898 eye images of 37 different persons of which 33 are men and 4 are women. It contains all types of images like open, closed, with glasses, without glasses, and in different lighting conditions.



After downloading separate all images into closed and open eyes into two separate folders.

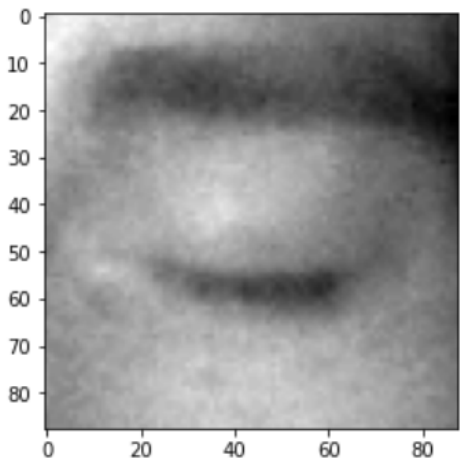
Implementation

First of all import required libraries.

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

Reading all the images in the dataset

```
Datadirectory = "mrlEyeDataset2/" Classes = ["Closed_Eyes", "Open_Eyes"] for category in Classes: path = os.path.join(Datadirectory, category) for img in os.listdir(path): img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE) backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB) plt.imshow(img_array, cmap="gray") plt.show() break break
```



Resizing the Image

Resize all the images into 224 x 224 for better feature extraction.

```
img_size = 224 new_array = cv2.resize(backtorgb, (img_size, img_size)) plt.imshow(new_array, cmap="gray") plt.show()
```

Creating training data

```
training_Data = [] def create_training_Data(): for category in Classes: path = os.path.join(Datadirectory, category) class_num = Classes.index(category) # 0 1, for img in os.listdir(path): try: img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE) backtorgb = cv2.cvtColor(img_array,cv2.COLOR_GRAY2RGB) new_array = cv2.resize(backtorgb, (img_size, img_size)) training_Data.append([new_array,class_num]) except Exception as e: pass create_training_Data()
```

Random shuffling to avoid overfitting.

```
import random random.shuffle(training_Data)
```

Creating arrays to store features and labels

```
X = [] y = [] for features,label in training_Data: X.append(features) y.append(label) X = np.array(X).reshape(-1, img_size, img_size, 3)
```

Import libraries that are required to build the model. Import the model and change the last fully connected(fc) layers and build the new model. Then compile the model using binary cross-entropy loss function and adam optimizer and then train the model for 5 epochs and then save the model so that no need to train again and again.

```
import tensorflow as tf from tensorflow import keras from tensorflow.keras import layers
```

```
model = tf.keras.applications.mobilenet.MobileNet() base_input = model.layers[0].input ##input base_output = model.layers[-4].output Flat_layers = layers.Flatten()(base_output) final_output = layers.Dense(1)(Flat_layers) final_output = layers.Activation('sigmoid')(final_output) new_model = keras.Model(inputs = base_input, outputs = final_output) new_model.compile(loss="binary_crossentropy", optimizer= "adam", metrics= ["accuracy"]) new_model.fit(X,Y, epochs = 5, validation_split = 0.2) ##training new_model.save('my_model.h5')
```

```
img_array = cv2.imread('mrlEyeDataset/Closed_Eyes/s0001_00080_0_0_0_0_0_01.png', cv2.IMREAD_GRAYSCALE) backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB) new_array = cv2.resize(backtorgb, (img_size, img_size)) X_input = np.array(new_array).reshape(1, img_size, img_size, 3) X_input = X_input/255.0 #normalizing data prediction = new_model.predict(X_input)
```

Detection and Alerting

Now let us see the implementation for detection and alerting. Here we set the timer for 2 seconds and 5 frames per second. If the eyes were closed for 2 seconds continuously that means if the model captures continuous 10 closed eye frames then it will alert the driver by beeping the alarm sound.

```
import winsound frequency = 2500 # Set frequency to 2500 duration = 1500 # Set duration to 1500 ms == 1.5 sec import numpy as np import cv2 path = "haarcascade_frontalface_default.xml" faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml") cap = cv2.VideoCapture(1) #check if webcam is opened correctly if not cap.isOpened(): cap = cv2.VideoCapture(0) cap.set(cv2.CAP_PROP_FPS, 5) counter = 0 while True: ret,frame = cap.read() eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml') gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) eyes = eye_cascade.detectMultiScale(gray, 1.1, 4) for x,y,w,h in eyes: roi_gray = gray[y:y+h, x:x+w] roi_color = frame[y:y+h, x:x+w] cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
```

```
eyess = eye_cascade.detectMultiScale(roi_gray) if len(eyess) == 0: print("Eyes are not detected") else: for (ex, ey, ew, eh) in eyess: eyes_roi = roi_color[ey: ey+eh, ex: ex+ew] gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) if(faceCascade.empty()==False): print("detected") faces = faceCascade.detectMultiScale(gray, 1.1, 4) # Draw a rectangle around eyes for (x,y,w,h) in faces: cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2) font = cv2.FONT_HERSHEY_SIMPLEX final_image = cv2.resize(eyes_roi, (224,224)) final_image = np.expand_dims(final_image, axis=0) final_image = final_image/255.0 Predictions = new_model.predict(final_image) if (Predictions>=0.3): status = "Open Eyes" cv2.putText(frame, status, (150,150), font, 3, (0, 255, 0), 2, cv2.LINE_4) x1,y1,w1,h1 = 0,0,175,75 cv2.rectangle(frame, (x1, y1), (x1 + w1, y1 + h1), (0,0,0), -1) #Add text cv2.putText(frame, 'Active', (x1 + int(w1/10),y1 + int(h1/2)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,0), 2) elif Predictions<0.3: counter = counter + 1 status = "Closed Eyes" cv2.putText(frame, status, (150,150), font, 3, (0, 0, 255), 2, cv2.LINE_4) x1,y1,w1,h1 = 0,0,175,75 cv2.rectangle(frame, (x1,y1), (x1 + w1, y1 + h1), (0,0,255), 2) if counter > 10: x1,y1,w1,h1 = 0,0,175,75 #Draw black background rectangle cv2.rectangle(frame, (x1, y1), (x1 + w1, y1 + h1), (0,0,0), -1) #Add text cv2.putText(frame, "Sleep Alert !!!", (x1 + int(w1/10), y1 + int(h1/2)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255), 2) winsound.Beep(frequency, duration) counter = 0 cv2.imshow("Drowsiness Detection", frame) if cv2.waitKey(2) & 0xFF == ord('q'): break cap.release() cv2.destroyAllWindows()
```

Conclusion

When you run the code it will open the webcam and will capture the video and gives output based on your eyelid closure. This drowsiness detection system helps the drivers a lot and prevents many road accidents that are caused due to drowsiness. So far we have seen,

–> How to use transfer learning?

–> How to build the model, train the model?

–> How to change the layers according to our problem statement?

–> Finally implementation of the drowsiness detection system.

Hope you guys found it useful.

Article Url - https://www.analyticsvidhya.com/blog/2022/05/drowsiness-detection-system/

Amrutha K