

THE OBJECTIVE OF THIS PROJECT:

lack of vigilance by drivers is one of the main causes of road accidents, leading to a troubling number of injuries and deaths every year. It is natural for drivers on long journeys to doze off behind the wheel. Even stress and lack of sleep can make drivers feel drowsy while driving. This project aims to prevent and reduce these accidents by creating a drowsiness detection system. Here, we need to create a system that can detect drivers' closed eyes and alert them if they fall asleep while driving. Even if the driver's eyes are closed for a few seconds, this system will immediately inform the driver, thus avoiding terrible traffic accidents. The system classify the driver's eyes as "open" or "closed".

The recognition of EYES can be done with different methods but the most interesting is the machine learning with a neural network and that what will be developed during the following coming steps:

- 1. Collecting data**
- 2. Preparing the Dataset**
- 3. training our custom YOLOv4-Tiny object detector**

TAKING PHOTOS AND ANNOTATE

Probably one of the most famous quotes defending the power of data is that of Google's Research Director Peter Norvig claiming that "We don't have better algorithms. We just have more data". For that we need a relatively large data set for optimum performance simply because they are able to "learn" more features (almost like gaining experience).so we took 1000 images using the camera of the other one in different positions and either in different distances and backgrounds to be more precise under the training of our Dataset.

In order for our detector to learn to detect objects in images, it needs to be fed with labeled training data.so we have the following:

1. Install Microsoft's Visual Object Tagging Tool (VoTT).
2. Annotate the objects inside the photos inserted.
3. Once you have labeled enough images press "CTRL+E" to export the project. You should now see a folder called "vott-csv-export". Within that folder, you should see a "*.csv file" called "Annotations-export.csv" which contains file names and bounding box coordinates.

CREATING DATASET

"<https://app.roboflow.com/datasets>" Using this app we will be able to create our own Dataset which will be used in training our detector.

1. Importing our images being annotated before in addition to "Annotations-export.csv" file.
 2. Applying the resize preprocessing option --> scratch to 416 x 4163.
- UploadTrain/Test Split the dataset and choose "Train/Test Split" of "70% 20% 10%"4. Generate our Dataset


vectorCompleteDataset

Last Upload
3 days ago
303 images added

Dataset Size
303 images
Augmentation Disabled

Annotations
vectorCompleteDataset
Object Detection

Images




[View all images \(303\)](#)


☒ Train/Test Split
Your images are split at upload time. [Learn more.](#)

Train
365

Valid
105

Test
52



 Augmentation Options
Randomly applied to images in your training set

+ Add Augmentation Step

Augmentations create new training examples for your model to learn from. [Learn more on our blog.](#)

TRAINING OUR DATASET

"<https://colab.research.google.com/drive/1pwowg038eognddf6sxdg5asc8picae-g#scrollto=gnavu7eu9cqj3>" Using this site we then be able to use "YOLOv4-tiny-Darknet-Roboflow.ipynb" where in this notebook, we implement the tiny version of YOLOv4 for training on your own dataset, YOLOv4 tiny. Just follow the tutorial in this notebook but taking into consideration to change: --> %env compute capability=60 (replace here 60 by 30: -->compute capability=30)And also in --> !sed -i "s/ARCH= -gencode arch=compute_60,code=sm_60/ARCH= -gencode arch=compute_\${compute capability},code=sm_\${compute capability}/g" Make file (replace here 60 by 30: compute_30,code=sm_30)In the part of "SET UP CUSTOM DATASET FOR YOLOV4" you have to put the link of the export of your dataset.

%%%%% NOTE: YOLO Darknet format

Once training is done we download:

- 1. The detector file "custom-yolov4-tiny-detector.cfg" in the cfg directory**
- 2. The weight file "custom-yolov4-tiny-detector_best.weights" in the backup directory**

OBJECT DETECTION USING OPENCV WITH PYTHON

These steps to install and activate this environment to be able to use python codes directly and also the openCV module:

→ `pip -V`

→ --> `sudo apt install python-pip`

→ --> `pip install --user virtualenv`

```
import cv2
```

```
import numpy as np
```

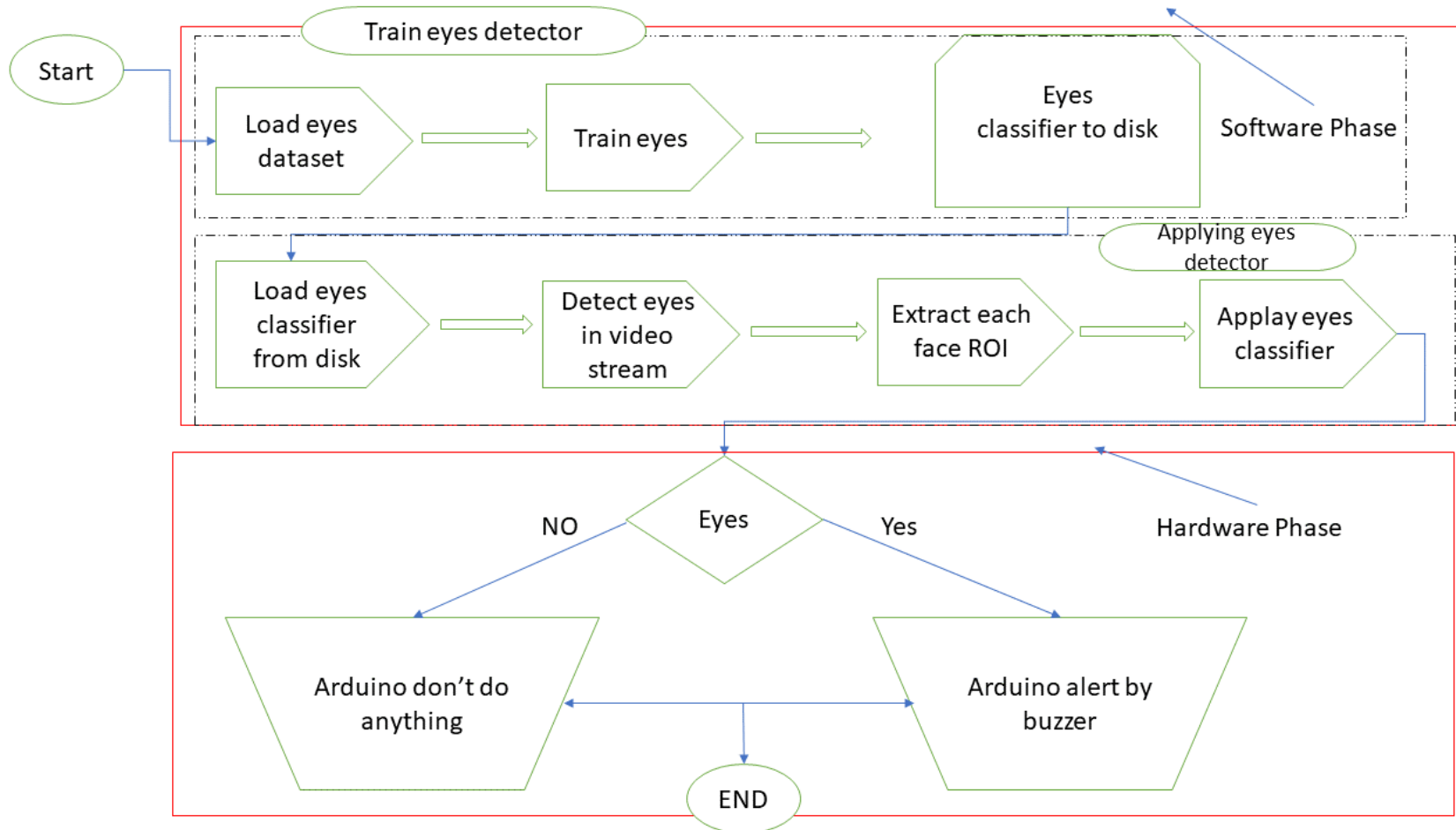
```
import time
```

```
import serial
```

Weight file: it's the trained model, the core of the algorithm to detect the objects.

Cfg file: it's the configuration file, where there are all the settings of the algorithm.

METHODOLOGY



CONCLUSION

- To study eyes detection and recognition techniques.
- To design a system that can detect and recognize open or closed eyes not in real time.
- To simulate and obtain results using OpenCV with Python.
- To implement the system on Arduino.
- The method attains accuracy up to 97.80%.

