International Institute of Information Technology, Bangalore

Software Production Engineering project

Drowsiness Detection live

Under the Guidance of
Prof. B. Thangaraju

Team Members:

1. A.V.Karthik Reddy – IMT2018011
2. gopal svs – IMT2018083
3. G Vamsi - IMT2018506

Table of Contents

1.Abstract:

Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving. A countless number of people drive on the highway day and night. Taxi drivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep. Due to which it becomes very dangerous to drive when feeling sleepy. This can be an important safety implementation as studies suggest that accidents due to drivers getting drowsy or sleepy account for around 20% of all accidents and on certain long journey roads it's up to 50%. It is a serious issue and most people that have driven for long hours at night can relate to the fact that fatigue and slight brief state of unconsciousness can happen to anyone and everyone.

There has been an increase in safety systems in cars & other vehicles and many are now mandatory in vehicles, but all of them cannot help if a driver falls asleep behind the wheel even for a brief moment. The majority of accidents happen due to the drowsiness of the driver. So, to prevent these accidents we will build a system using Python, OpenCV, and Keras which will alert the driver when he feels sleepy.

The objective of this Python project is to build a drowsiness detection system that will detect that a person's eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected.

The architecture of our project demands two layers.

- Front end.
- Back end.

The front end of the project is handled by "html" and the back end are swiftly handled by "Python", using "Flask" for interaction between ends. There are 3 main steps in drowsiness detection: Human detection, face detection and drowsiness detection. In this project Human detection was done using YOLO algorithm. Whereas Face detection was done by VIOLA JONES and Drowsiness detection was done with the help of face coordinates.

2 Introduction

2.1. OVERVIEW

‘Drowsiness detector live’ provides solution to all car companies around the world, who wants the driver safety and implement drowsiness detection that helps the driver to be active while driving which indeed prevents many road accidents. This system will make sound(alarm) when driver is drowsy. With help of this he can be active most of the times.

2.2 FEATURES

Live Drowsiness Detector.

2.3. WHY DEVOPS?

Our whole approach of the project was modular, we wanted to make different sets of development modules and wanted to deploy them with every new release without any hindrance. Wanted to test the changed code with continuous integration and then continuously deploying it. So, what all fills all these blanks was a culture, a philosophy DevOps. DevOps provides all the tools to increase the capability to complete above set goals within minimum time and less trouble for developers. DevOps tools consist of configuration management, test and build systems, application deployment, version control and monitoring tools. Continuous integration, continuous delivery and continuous deployment require different tools.

2.3.1. Devops Features

Improve deployment frequency

- Achieve faster time to market with lower failure rate
- More stable operating environments
- Improve communication and collaboration among teams

3. System Configuration

3.1. Operating system
Ubuntu 18.04.04 and it should have access to system camera or web cam.

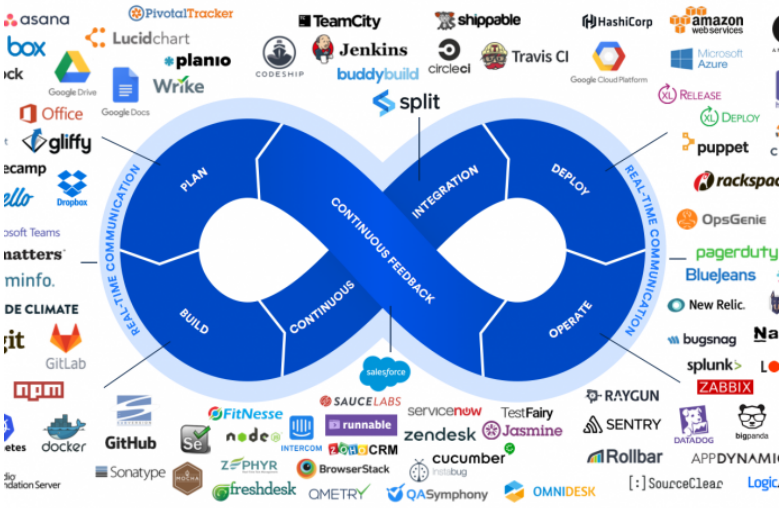
3.2. CPU and RAM
4 core processor and RAM 8 GB

3.3 KERNEL VERSION
Linux machine 5.8.0-44-generic

3.4. Language
python3, Flask framework.

3.5. DevOps Tools

- Source Control Management - GitHub
- Continuous Integration – Github actions
- Containerization - Docker
- Continuous deployment – Github actions, Heroku.
- Monitoring - ELK Stack (Elastic Search, Logstash, Kibana)



4. Software Development life cycle

4.1. INSTALLATION

4.1.1 Python Setup

The entire ML code is written in python and the web server is written using flask, a library in python. To install all those libraries required we use pip.

python3 -m pip freeze > requirements.txt- This command then lists all the version of libraries installed and directs them to file "requirements.txt"

python3 -m pip install -r requirements.txt- This command then installs the version of libraries into the environment using requirements.txt

106 lines (106 sloc) | 1.83 KB

```
absl-py==0.10.0
argon2-cffi==20.1.0
astunparse==1.6.3
async-generator==1.10
attrs==20.2.0
backcall==0.2.0
bleach==3.2.1
cachetools==4.1.1
certifi==2020.6.20
cffi==1.14.3
chardet==3.0.4
click==7.1.2
colorama==0.4.3
cycler==0.10.0
decorator==4.4.2
defusedxml==0.6.0
entrypoints==0.3
Flask==1.1.2
Flask-SocketIO==4.3.1
Flask-WTF==0.14.3
gast==0.3.3
google-auth==1.22.1
google-auth-oauthlib==0.4.1
google-pasta==0.2.0
grpcio==1.32.0
gunicorn==20.0.4
```

fig1 : requirements.txt looks like this.

4.2. SOURCE CONTROL MANAGEMENT

A Source Code Management (SCM) is a software tool used by programmers to manage the source codes. For our project every team member would clone the repository from git hub, create a different branch locally on their system and then merge it with master. The cycle of pulling the latest code from git, resolving any conflicts and then pushing the changes to git keeps happening for every update.

- **git clone** - This command copies the entire data on the git URL
- **git checkout -b <branch_name>** This command creates a new branch with the name as in 'branch_name'
- **git add** - This command adds changes in the working directory to the staging area
- **git commit -m "message while committing"**-This command is used to save your changes to the local repository with -m used to provide a concise description that helps your teammates (and yourself) understand what happened.
- **git checkout master**- This command switches to master branch
- **git pull**-This command is used to update the local version of a repository from a remote.
- **git merge** -This command is used to integrate changes from another branch.
- **git push**-This command will push all the latest code to the repository.

For Drowsiness Detector live GitHub link :Github Link for the project

4.3 BUILDING

PYTHON:

To build the python project we use requirements.txt file, and using pip we install all the dependent libraries for the project. We can find requirements file from the above given github repository link.

Command:$ pip install -r requirements.txt

the above command will install all necessary requirements

4.4 TEST

For testing the Drowsiness Detectoin live, we use unit test. Test cases are written in keeping a view to cover all possible cases.

Steps to use unittest.

1. Import unittest from the standard library
2. Create a class called TestSum that inherits from the TestCase class
3. Convert the test functions into methods by adding self as the first argument
4. Change the assertions to use the self.assertEqual() method on the TestCase class
5. Change the command-line entry point to call unittest.main()

source > test_drowsiness.py

```python
class TestModel(unittest.TestCase):
    def test1(self):
        rpred=[99]
        lpred=[99]
        i = 0
        thresh_time =0
        cum_score=[]
        result = []
        cap = cv2.VideoCapture('test_videos/test1.mp4')
        while(True):
            success,frame = cap.read()
            if(success == 0):
                break
            frame_processed,i,cum_score,result = main_fun(frame,i,rpred,lpred,thresh_time,cum_score,result)

        if 1 in result:
            result = 1
        else:
            result = 0
        print("video 1 result:- " , " Predicted Status : " , dict[result], " , Actual Status: " , dict[1])
        logger.info("video 1 result:- %d , Predicted Status : %s , Actual Status : %s",result,dict[result],dict[1])

        self.assertEqual(result, 1)
        self.assertNotEqual(result, 0)


    def test2(self):
        rpred=[99]
        lpred=[99]
        i = 0
        result = []
        thresh_time =0
        cum_score=[]
        cap = cv2.VideoCapture('test_videos/test2.mp4')
        while(True):
            success,frame = cap.read()
            if(success == 0):
                break
            frame_processed,i,cum_score,result = main_fun(frame,i,rpred,lpred,thresh_time,cum_score,result)

        if 1 in result:
            result = 1
        else:
            result = 0

        print("video 1 result:- ", " Predicted Status : " , dict[result], " , Actual Status: " , dict[1])
        logger.info("video 1 result:- %d , Predicted Status : %s , Actual Status : %s",result,dict[result],dict[1])
        self.assertEqual(result, 1)
```

fig 3: unit test cases

```
Terminal
(MLenv) lenovo@karthik:~/Downloads/SPE_PROJ-main$ python3 source/test_drowsiness.py
2022-05-12 14:17:43.523587: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
 shared object file: No such file or directory; LD_LIBRARY_PATH: /home/lenovo/miniconda3/envs/MLenv/lib/python3.8/site-packages/cv2/../../lib64:
2022-05-12 14:17:43.523676: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-05-12 14:17:56.784579: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared ob
ject file: No such file or directory; LD_LIBRARY_PATH: /home/lenovo/miniconda3/envs/MLenv/lib/python3.8/site-packages/cv2/../../lib64:
2022-05-12 14:17:56.784682: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-05-12 14:17:56.784754: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (karthik): /proc/driver/nvidia/version
does not exist
2022-05-12 14:17:56.797843: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the follow
ing CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
------------Testing on sample videos-----------
video 1 result:-   Predicted Status :  Drowsiness  , Actual Status:  Drowsiness
.video 1 result:-   Predicted Status :  Drowsiness  , Actual Status:  Drowsiness
.video 1 result:-   Predicted Status :  No Drowsiness  , Actual Status:  No Drowsiness
.video 1 result:-   Predicted Status :  No Drowsiness  , Actual Status:  No Drowsiness
.
----------------------------------------------------------------------
Ran 4 tests in 138.517s

OK
(MLenv) lenovo@karthik:~/Downloads/SPE_PROJ-main$
```

Fig 4: Running our model on test videos, which gives the actual status and predicted status of the person in the video.

4.4.1 CI workflow(Github actions):

GitHub Actions help you automate tasks within your software development life cycle. GitHub Actions are event-driven, meaning that you can run a series of commands after a specified event has occurred. For example, every time someone creates a pull request for a repository, you can automatically run a command that executes a software testing script.

This diagram demonstrates how you can use GitHub Actions to automatically run your software testing scripts. An event automatically triggers the workflow, which contains a job. The job then uses steps to control the order in which actions are run. These actions are the commands that automate your software testing.

The components of GitHub actions:
Below is a list of the multiple GitHub Actions components that work together to run jobs. You can see how these components interact with each other.

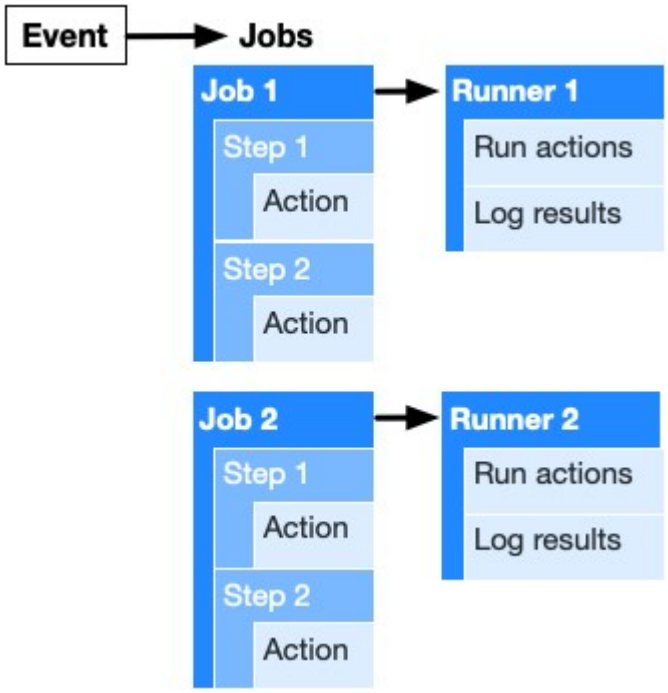


fig 5: Structure of GitHub actions components.

Workflows

The workflow is an automated procedure that you add to your repository. Workflows are made up of one or more jobs and can be scheduled or triggered by an event. The workflow can be used to build, test, package, release, or deploy a project on GitHub.

Events

An event is a specific activity that triggers a workflow. For example, activity can originate from GitHub when someone pushes a commit to a repository or when an issue or pull request is created.

Jobs

A job is a set of steps that execute on the same runner. By default, a workflow with multiple jobs will run those jobs in parallel. You can also configure a workflow to run jobs sequentially.

Steps

A step is an individual task that can run commands in a job. A step can be either an *action* or a shell command. Each step in a job executes on the same runner, allowing the actions in that job to share data with each other.

Actions

Actions are standalone commands that are combined into *steps* to create a *job*. Actions are the smallest portable building block of a workflow. You can create your own actions, or use actions created by the GitHub community. To use an action in a workflow, you must include it as a step.

Runners

A runner is a server that has the GitHub actions runner application installed. You can use a runner hosted by GitHub, or you can host your own. A runner listens for available jobs, runs one job at a time, and reports the progress, logs, and results back to GitHub.

GitHub Actions uses YAML syntax to define the events, jobs, and steps. These YAML files are stored in your code repository, in a directory called .github/workflows.

Steps to be followed for creating a workflow:

1. In your repository, create the .github/workflows/ directory to store your workflow files.
2. In the .github/workflows/ directory create a new YAML file and add the code.
3. Commit those changes and push them to your GitHub repository.
   In the Github actions workflow we write a script to first set up the environment to run the test cases written for drowsiness-detection-iiitb. And then we call to test various test cases.

4.5. DOCKER ARTIFACT

Docker is a software platform for building applications based on containers which are small and lightweight execution environments that make shared use of the operating system kernel but otherwise run-in isolation from one another. Docker images of the module is created and then pushed to docker hub, from where we can pull those images and run it at our end. Docker file is created for the project that will run when we build these docker images. So install docker on your deployment and development machines.

```
Dockerfile
1   FROM python:3.6
2   MAINTAINER gopalsvs venkatasaigopal01@gmail.com
3   WORKDIR ./
4   COPY requirements.txt requirements.txt
5   RUN pip install -r requirements.txt
6   COPY . .
7   EXPOSE 8000
8   CMD ["gunicorn", "drowsiness_detection:app"]
9
```

fig6: docker file, used to build docker image.

Command to build docker image in terminal:

$ **docker build -t <docker user id>/<repo name> .**

4.5.1 CI Workflow

First We need to add credentials for our docker hub and some others if necessary. To do that

1.) go to settings in github.

2.) on the left side, bottom third one, select secrets.

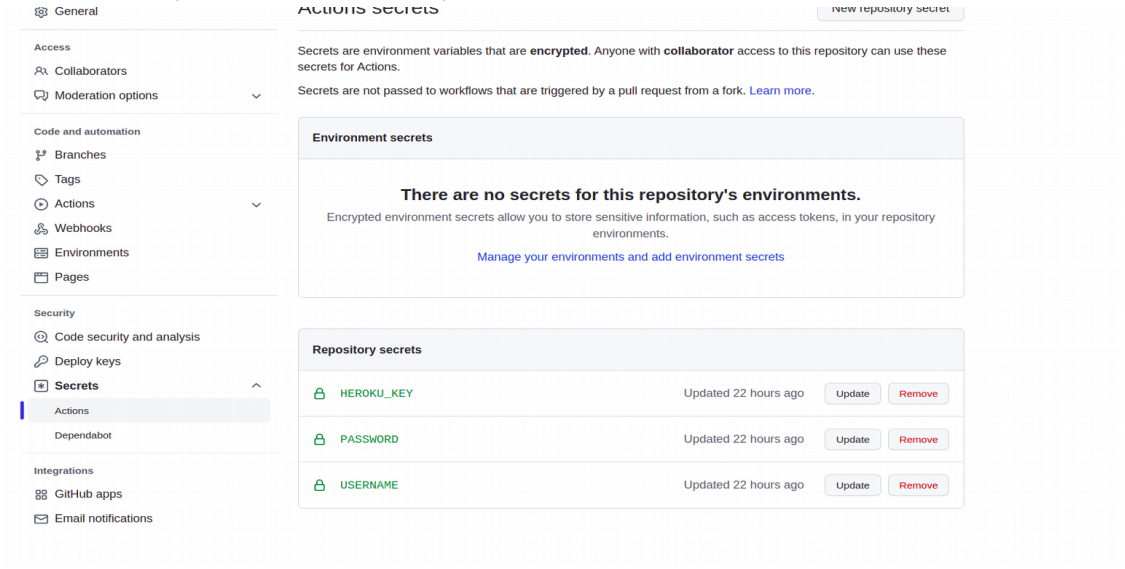


Fig 7: way to add credentials.

3. Here select new repository secret and add the credentials. Note: for every thing we need to give a name to it and then its value eg: to add docker hub id, give something called as

DOCKERHUB_USERNAME as name and our id pavanperuru as value. Same logic for every addition of credential

- Now add a related workflow script to build the docker image and push it to docker hub.
- Make sure the docker credential here in script matches your docker credential id you set earlier.

```yaml
- name: Docker Image Building and Pushing
  uses: mr-smithers-excellent/docker-build-push@v4
  with:
    image: teeyagundi/spe_project
    tag: latest
    registry: docker.io
    username: ${{ secrets.USERNAME }}
    password: ${{ secrets.PASSWORD }}
```

Fig 8: Building and deploying drowsiness-detection-iiitb image on docker hub in github actions workflow.

- After successful deployment of docker images we can see the result onour docker-hub account.

4.6 Heroku:

Heroku is a cloud Platform as a Service (PaaS) that we can use that to deploy, manage and to scale apps.

```yaml
- name: Deploy the application to heroku cloud platform
  uses: akhileshns/heroku-deploy@v3.12.12 # This is the action
  with:
    heroku_api_key: ${{secrets.HEROKU_KEY}}
    heroku_app_name: "drowsiness-detection-iiitb"
    heroku_email: "karthikavreddy@gmail.com"
    usedocker: true
```

Fig 9: deployment workflow of drowsiness-detection-iiitb.

After deployment if we use this following link we can run our application. https://drowsiness-detection-iiitb.herokuapp.com/

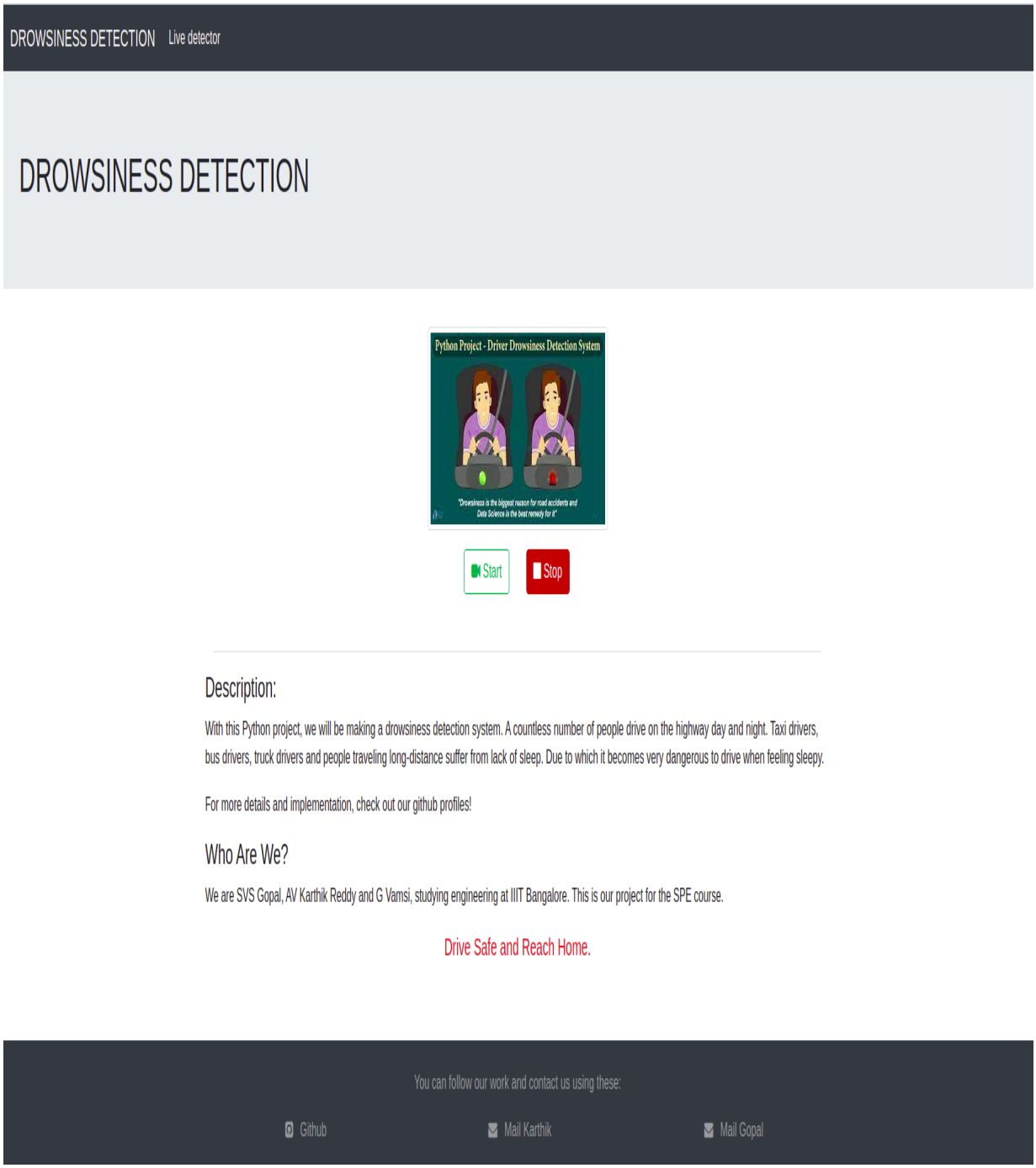


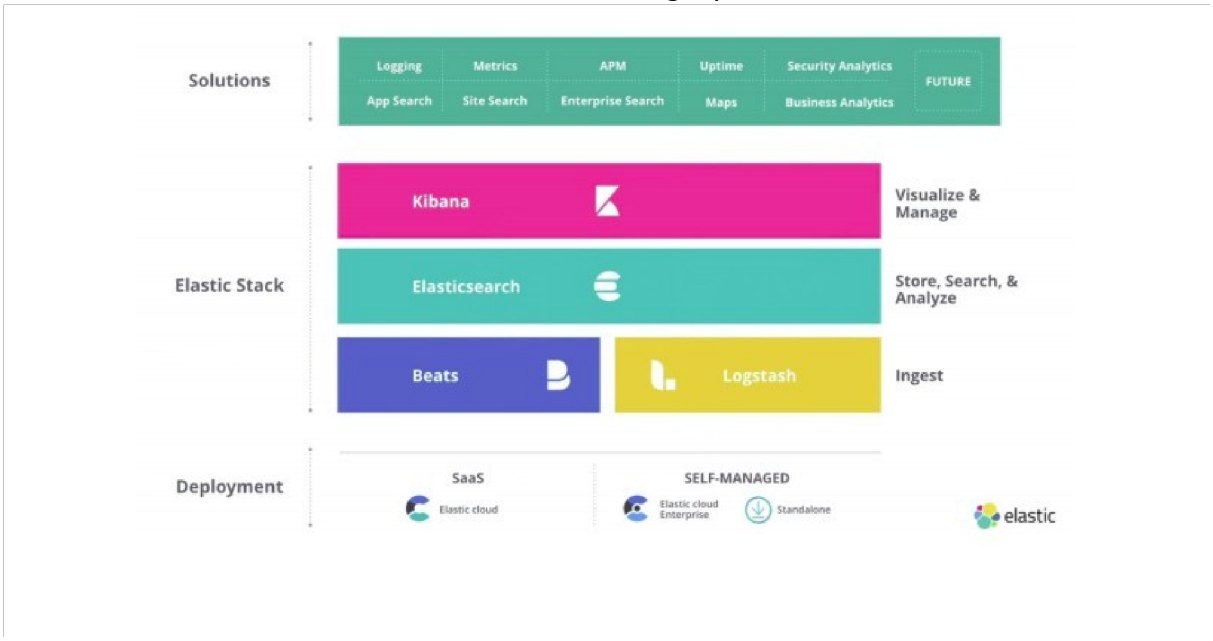
fig10: This is how the application interface looks.

4.7 Monitoring using ELK

Continuous monitoring provides near immediate feedback and insight into performance and interactions across the network.

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana .

- Elasticsearch is a search and analytics engine.
- Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch.
- Kibana lets users visualize data with charts and graphs in Elasticsearch.



4.7.1 Logstash, configuration file:

Downloading Logstash:
https://www.elastic.co/downloads/logstash
Download appropriate version for your OS. (Ex. LinuxX86_64)
After the tar.gz file is downloaded, use this command to extract $tar -xvf path/to/file/tar.gz

Configuration file:

The bare structure of the configuration file of logstash is input
{ plugin {
settings
}
}
filter { plugin
{
settings
} }
output
{ plugin{
settings
}
}
To run logstash, we have to input a configuration file on how to parse the data of a given log file.
Go into the extracted logstach directory and execute this command to run.
$./bin/logstash -f /path/to/configuration/file **Command and the Output :**

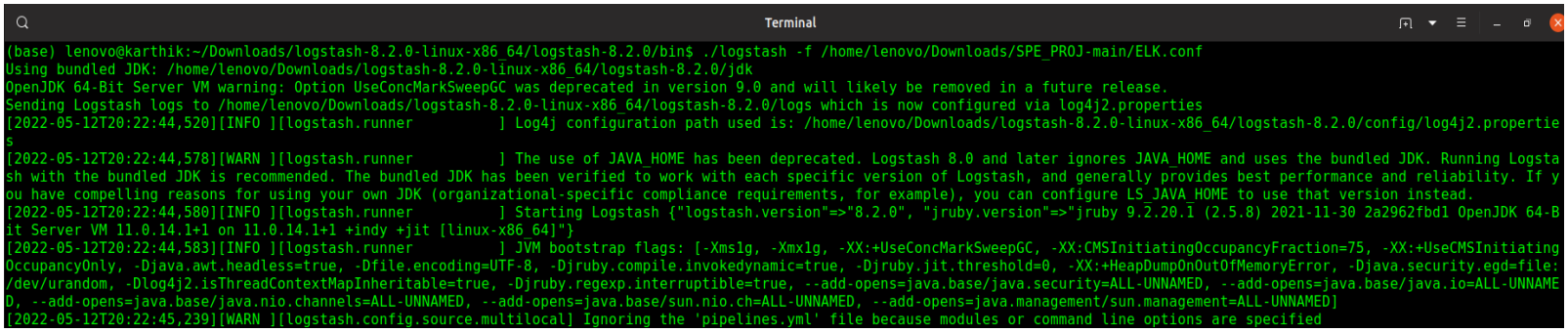


fig 11:command to run the logstash.

4.7.2 Elasticsearch, Kibana Visualizations:

Download the Elasticsearch archive for your OS https://www.elastic.co/downloads/elasticsearchDownload the Kibana archive for your OS, https://www.elastic.co/downloads/kibana Or use cloud service for the above. But the configuration file we have to be changed accordingly. Kibana is an open source browser based visualization tool mainly used to analyze large volumes of logs in the form of line graph, bar graph, pie charts, heat maps, region maps, coordinate maps, gauge, goals, timelion etc.

The visualization makes it easy to predict or to see the changes in trends of errors or other significant events of the input source. With the index pattern we tell kibana what Elasticsearch index to analyze.

Follow the following steps to set up the index pattern:
In Kibana, go to Management → Stack Management Look for Kibana → Index Patterns → Create Index Pattern , set your index pattern based on the index pattern provided in logstash configuration file followed by *. And in the next step select @timestamp as your Time field. Hit Create index pattern , and you are ready to analyze the data. Explore discover and Dashboard options.

4.8. BUILDING WORKFLOW

From the above sections we have seen how we have integrated docker and other necessary tools with github actions. We were able to build docker images with github actions. Here’s the reference figure to the entire workflow and its final full stage view after a successful run of github workflow..

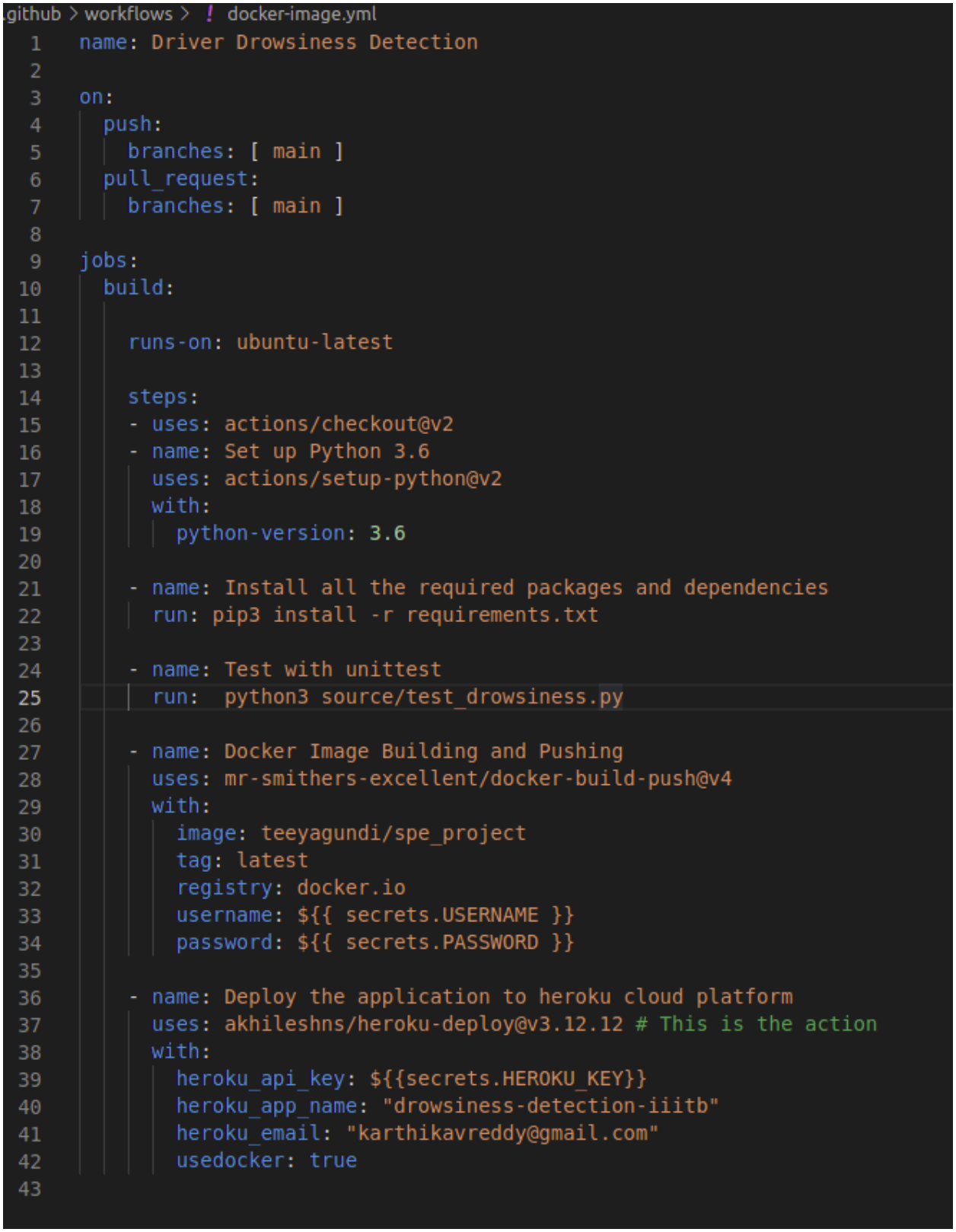


Fig 14: complete workflow of github actions.

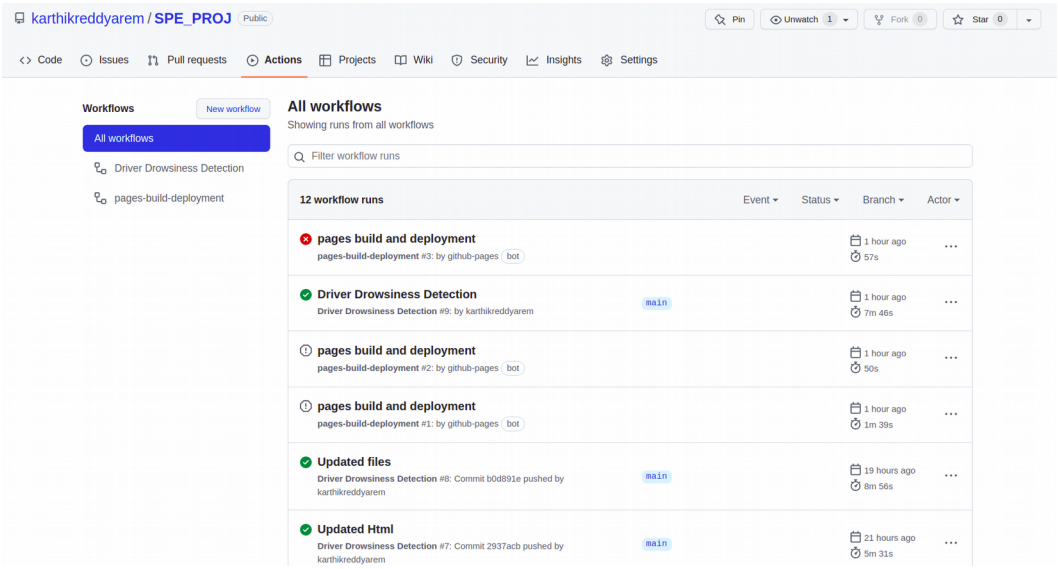


Fig 15: actions page

After completion of building workflow Click on the .yml file and then press build that's it after successful build the following image will show up. If its a green tick then it means that our build is correct.

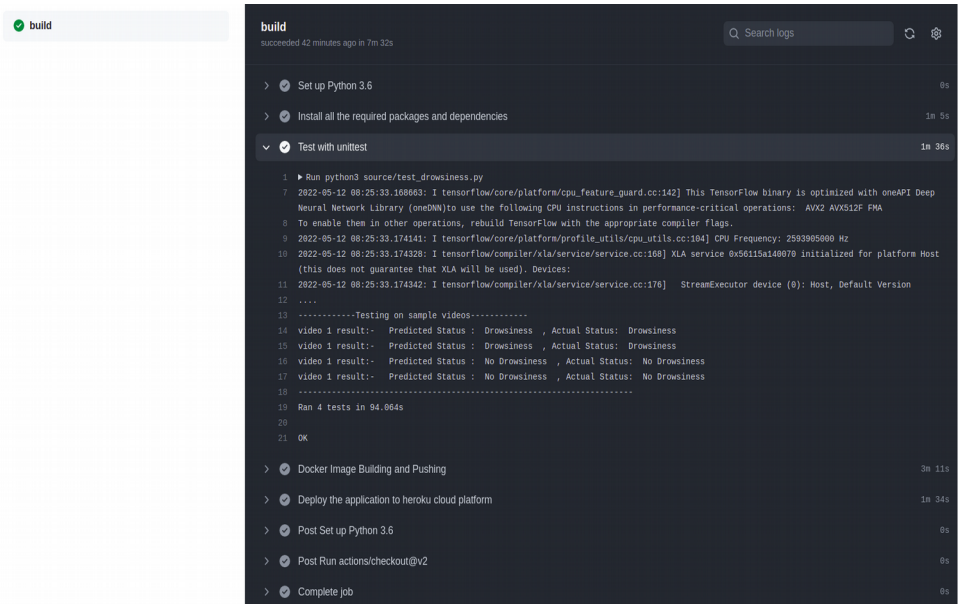


Fig 16: a successful build.

5. Model:

As the input is video we read each frame and used YOLOV4 to detect the person on each frame. If it detects person then we used VIOLA JONES algorithm to get the face Region of Interest(ROI) and gave this ROI as input to the drowsiness classifier model.

A. YOLO

We used Open CV dnn module with per-trained YOLOV4 model to detect objects. We will get the class id, confidence and bounding box corners of all the detection, then we onlytake the detections that are of person(class id = 0) with confidence value greater than 0.9. Even though we ignored weak detections(confidence value < 0.6), there will be lot of duplicate detections with overlapping bounding boxes. Non-max suppression removes boxes with high overlapping. We took non max threshold as 0.45.Finally from YOLO we are identifying the person from the frame i.e. driver.

B. VIOLA JONES

Only after identifying the person in the frame, we are using VIOLA JONES algorithm to get eye landmarks. In Open CV, we have several trained Haar Cascade models whichare saved as XML files. After using the function in Open CV, detectMultiScale() with loaded model object we will get all Region Of Interest(eyes bounding box corners) in the frame. Finally from VIOLA JONES algorithm we will get the eyes landmarks.

C. DROWSINESS DETECTION

The output from VIOLA JONES is reshaped to 22 x 22 x 3 image. We used this reshaped image for detecting thedrowsiness or no drowsiness by giving input to the CNNmodel.

1) Data set and Preprocessing: Data set has been taken from kaggle which contains 4K images divided 2K open eyes and 2K closed eyes. We augmented the data set and then converted the shape of each image to 22x22x3.

2) Architecture: we used 3 convolutional layers and 1 fully connected layer(dense). We used Adam optimization algorithm as an optimizer and cross entropy loss as an loss function. We used 50 epochs to train the model and accuracy, loss as metrics for validating. We maintain a sliding window of size 10 frames to keep track of the previous state of the eyes to detect drowsiness of a person. So for each frame(except inital 9 frames) we will get 10 values which tells the status of the eyes whether closed(0) or open(1). Here we are using past values to say whether the driver is drowsy or not. We can calculate the percentage of of closed eyes from the 10 values((no of times eyes closed*100)/10)We experimented with percentages while driver in drowsy state and in normal state. From those experiments we got the if percentage is above 60 then the driver is drowsy (out of 10, 6 times his eyes are closed). We kept threshold as 60 percent and above that we can detect the drowsiness. This sliding window will be moving for every new frame and updates the score accordingly. By using this method it helps us to identify frequent blinks of eyes to detect it as drowsiness.

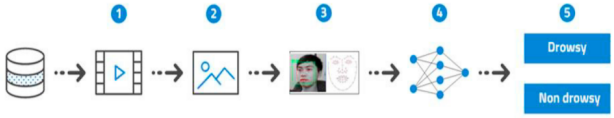


Fig. 2. Flow Diagram:We will take input from camera and then see at each frame , obtain region of intrest and use neural network to classify.

Fig 17: Flow Diagram : We will take input from camera and then see at each frame, obtain region of interest and use neural network to classify.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 22, 22, 32)	320
max_pooling2d_3 (MaxPooling2	(None, 22, 22, 32)	0
conv2d_4 (Conv2D)	(None, 20, 20, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 20, 20, 32)	0
conv2d_5 (Conv2D)	(None, 18, 18, 64)	18496
max_pooling2d_5 (MaxPooling2	(None, 18, 18, 64)	0
dropout_2 (Dropout)	(None, 18, 18, 64)	0
flatten_1 (Flatten)	(None, 20736)	0
dense_2 (Dense)	(None, 128)	2654336
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 2)	258

Total params: 2,682,658
Trainable params: 2,682,658
Non-trainable params: 0

Fig 18: Model Architecture

V. Results

At 50^{th} epoch

	accuracy	loss
train	0.9462	0.1649
validation	0.9534	0.1550e−04

TABLE I
PERFORMANCE

Fig 19: accuracy and loss at 50^{th} epoch.

These are the plots for the Drowsiness Detection model for 50 epochs and some sample outputs of two frames along with annotations:

Fig 20 : Epochs vs Accuracy

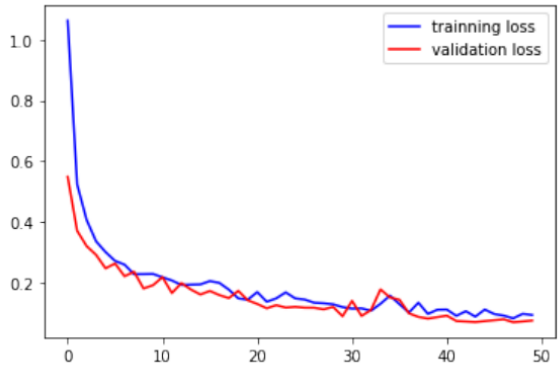


Fig 21: Epochs vs Loss

	Avg time to process one frame
Module	Time (in seconds)
Object-detection	0.542
Eyes-detection	0.027
Drowsiness-detection	0.078
object + Eyes detection	0.553
object + Eyes + drowsiness detection	0.575

Above table gives us the real-time performance and accuracy of each module. Average frames per second obtained without object detection is 12fps. But implementing object detection reduced the number of frames per second to avg of 5 fps. This values are obtained when run on a local machine with CPU. Overall the model can perform much better with high processing system and further reduce the processing time for each frame.

6: Results and discussion:

To run it in the local server we should go the project directory and we should use the following command.

$ **Flask run**

```
(MLenv) lenovo@karthik:~/Downloads/SPE_PROJ-main$ flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
2022-05-12 14:48:32.951462: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
 shared object file: No such file or directory; LD_LIBRARY_PATH: /home/lenovo/miniconda3/envs/MLenv/lib/python3.8/site-packages/cv2/../../lib64:
2022-05-12 14:48:32.951607: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-05-12 14:48:44.826640: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared ob
ject file: No such file or directory; LD_LIBRARY_PATH: /home/lenovo/miniconda3/envs/MLenv/lib/python3.8/site-packages/cv2/../../lib64:
2022-05-12 14:48:44.826745: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-05-12 14:48:44.826824: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (karthik): /proc/driver/nvidia/version
does not exist
2022-05-12 14:48:44.840281: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the follow
ing CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Starting camera thread.
Stopping camera thread due to inactivity.
```

Fig 22: Running the code in our local machine.

We can see that in the above figure there is a line saying “**Running on http://127.0.0.1:5000/**”. If we open that link our drowsiness detection local page will be visible in our machines.

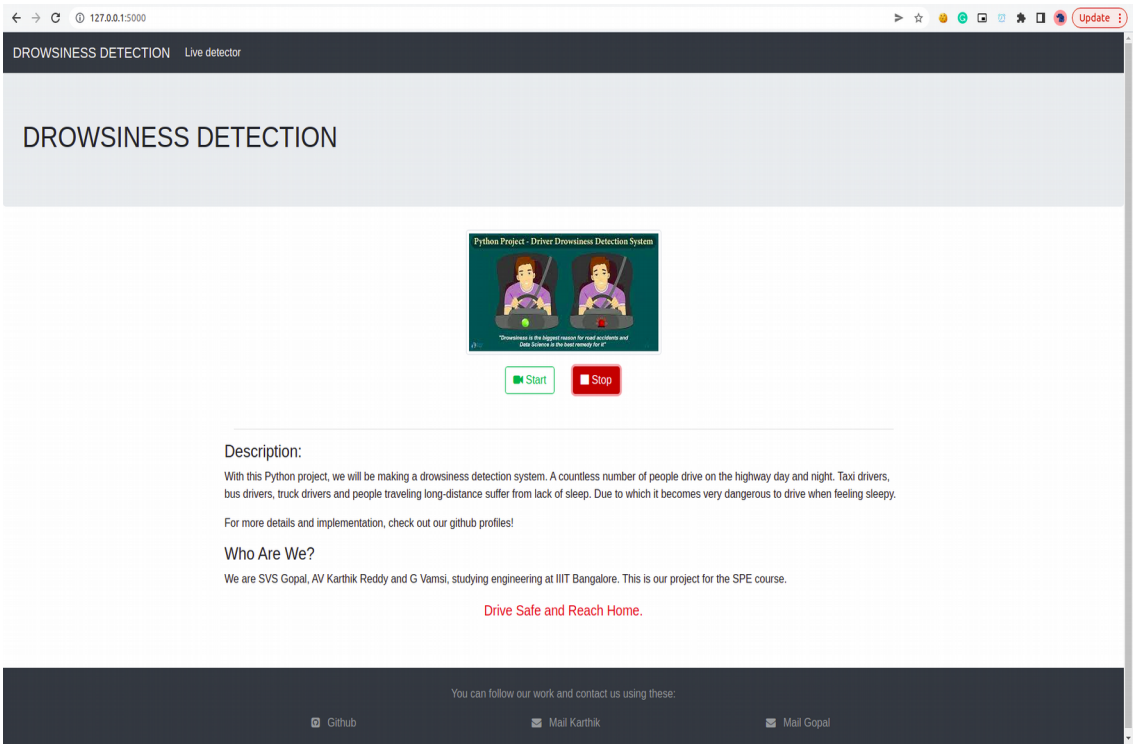


Figure 23: This is how it looks when we open the link in our local machine.

•Then press START for live drowsiness detection.

Around the human subject two boxes going to show up

1 The bigger box gives the bounding coordinates for the face

2 The smaller boxes gives us bounding boxes for the two eyes.

Reference images.

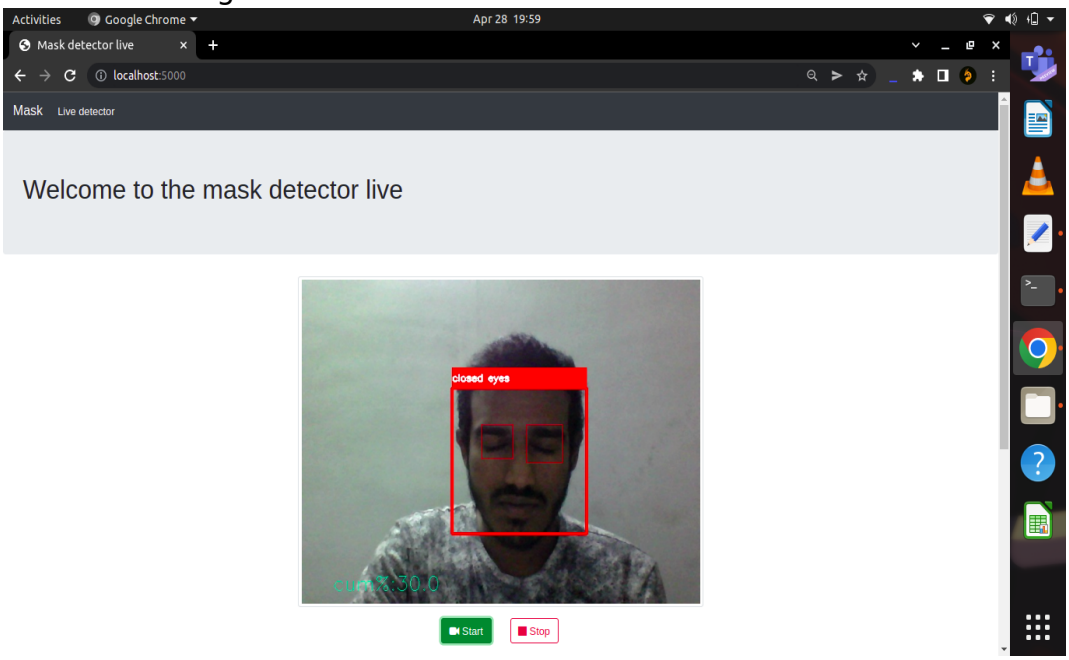


Figure 24: Person with closed eyes for more than 30% time.

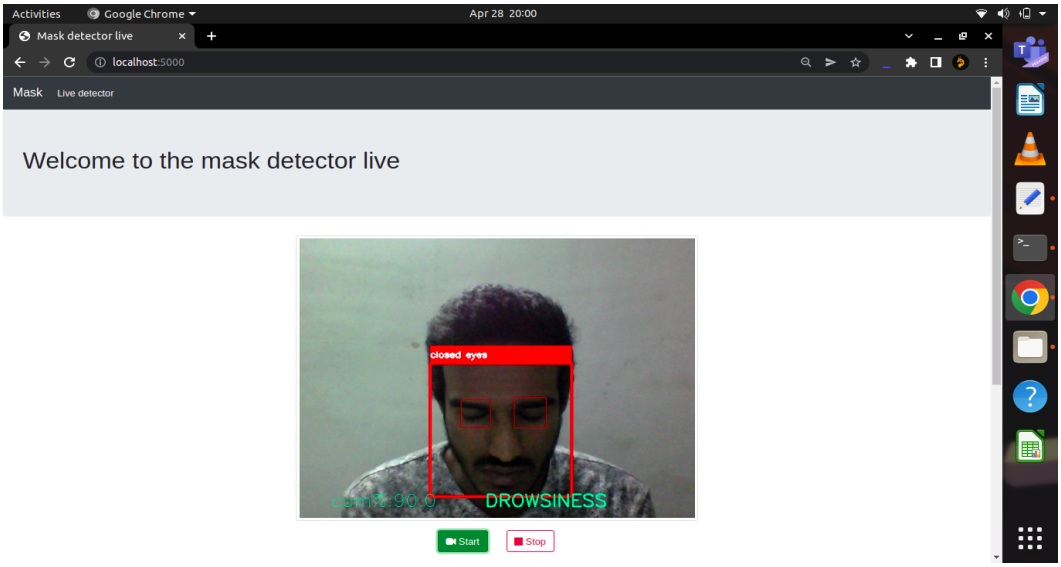


Figure 25: Drowsiness detected for closing eyes more than 90% of time.

7: FUTURE SCOPE AND CHALLENGES

When we want to minimize error we have to consider multiple methods to draw conclusions. Similarly we have to use physiological data of the person while driving to detect more accurately. But deploying the sensors on the body of a driver is not feasible for every trip. These physiological signals will help us to detect more accurately as they provide the data of our body functionality at real time. Some challenges that can occur in vision based techniques is that a driver can put on his cooling glasses which makes it difficult for the model to detect eyes. Hence alternate method of detecting drowsiness is necessary. So we could try to find a better solution to obtain the physiological signals of the driver more efficiently to improve the confidence of detection. For future works, we can try to focus on yawning analysis for drowsiness detection by landmark points of mouth. In addition, a different level of drowsiness can investigate without limitations of being in two situations because the boundaries among different levels of drowsiness are so narrow, and it would be a more challenging task.

8: CONCLUSION

The Devops methodology and life cycle tools prove to be better than the Agile methodology in terms of technical, cultural and business benefits. By minimizing friction between independent teams, DevOps enables a collaborative approach for enterprise software development and delivery that reflects the needs of the entire application life cycle for today's modern enterprises. Thus, we can develop, test, deploy and monitor the application easily.
The drowsiness detection plays a vital role in safe driving, and with the help of driver drowsiness detection system we can prevent accidents arising from drowsiness. Therefore, in the first step of the system, the system captures a stream of frames, and in the preprocessing unit, first object that is driver is detected by using YOLO algorithm then the driver fromthis bounding box of driver landmark points applied to access ROI, then eye region is selected, and the preprocessing unit selects the eye front of the camera. We used the output of this step as input to the neural network to eye state classification. If the network detects the eye is closed for more than 60 percent of times in the window size of 10, an alarm will be sent for the driver. Otherwise, it considers it as blinking. This is our complete mode for vision based detection of driver drowsiness at real time.

9 Links:

https://github.com/karthikreddyarem/SPE_PROJ

https://hub.docker.com/repository/docker/teeyagundi/spe_project

https://drowsiness-detection-iiitb.herokuapp.com/

10 REFERENCES:

[1] https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9
[2] https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg [3] https://towardsdatascience.com/review-mobilenetv2-light-weightmodel-image-classification-8febb490e61c