

DROWSINESS DETECTION USING OPENCV AND MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of Bachelor
of Engineering/Technology degree in BE,CSE

By

1.Konagalla Abhishek(38110255)

2.Olla Yaswanth Charan(38110667)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI - 600 119, TAMILNADU

MAY2020



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY (DEMED TO BE UNIVERSITY)

(Established under Section 3 of UGC Act, 1956)
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600119



www.sathyabama.ac.in

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **GADE YESWANTH ADITHYA REDDY (38110146)** and **KAKUMANU SAI MANIKANTA (38110489)** who carried out project entitled “Decentralized Voting System using Blockchain Technology” under my supervision from NOVEMBER 2021 to MARCH 2022

Internal Guide

Dr. S. JANCY, M.Tech., Ph.D.

Head of the Department

Dr. L. LAKSHMANAN., M.E., Ph.D.

Dr.S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I KONAGALLA ABHISHEK AND OLLA YASWANTH CHARAN hereby declare that the Project Report entitled DROWSINESS DETECTION USING OPEN CV AND MACHINE LEARNING done by us under the guidance of Ms.Aishwarya(Internal)and Dr.jayanthi(External)is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in BE, CSE.

DATE:

PLACE: SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala, M.E., Ph.D.**, Dean , School of computing and **Dr. L.Lakshmanan, M.E., Ph.D.**, , **Dr.S.Vigneshwari M.E., Ph.D.**, Heads of the Department of Computer science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. Jancy MCA., MBA., M.TECH., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer science and Engineering who were helpful in many ways for the completion of the project.

ABSTRACT:

More and more professions nowadays require concentration over the long term. Drivers must keep a near eye on the street, so they can respond to sudden occasions quickly. Driver fatigue typically becomes an instantaneous reason behind several traffic accidents. In this manner, there's a thought to create the frameworks that will distinguish and inform a driver of her/his terrible psychophysical condition, which may essentially diminish the number of fatigue-related car mishaps. Developing these devices, though, experiences several difficulties linked to swift and proper identification of the exhaustion signs of a rider. One of the specialized conceivable outcomes to execute driver tiredness discovery frameworks is to utilize the vision-based approach. Here we are identifying the driver drowsiness by evaluating vision framework .

CONTENTS

CHAPTER NO	TABLE OF CONTENTS	PAGE NO
ABSTRACT	V	
	LIST OF TABLES	
1	INTRODUCTION	
2	LITERATURE REVIEW	8
2.1	<u>Drowsiness Detection System Using Open CV</u>	8
	2.2 ACCIDENT MITIGATION SYSTEM WITH DROWSINESS	10
	2.3 Driver Drowsiness Detection System with OpenCV & TensorFlow	12
	2.4 Driver Drowsiness Detection System with OpenCV & Keras	13
3	SYSTEM DESIGN	16
4	MACHINE LEARNING	20
5	EYE ASPECT RATIO	33
6	FUTURE SCOPE	51

CHAPTER 1

INTRODUCTION

Driver tiredness discovery may be a car security innovation which avoids mishaps when the driver is getting lazy. Different considers have proposed that around 20% of all street mishaps are fatigue-related, up to 50% on certain streets. Driver weakness could be a critical calculate in a expansive number of vehicle mischances . Later measurements assess that yearly 1,200 passing and 76,000 wounds can be credited to weariness related crashes. The improvement of technologies for recognizing or avoiding laziness at the wheel could be a major challenge within the field of accident evasion frameworks. Since of the risk that laziness presents on the street, strategies got to be created for neutralizing its influences. Driver carelessness may be the result of a need of sharpness when driving due to driver tiredness and diversion. Driver diversion happens when an protest or occasion draws a person's consideration absent from the driving errand. Not at all like driver diversion, driver laziness includes no activating occasion but, instep, is characterized by a dynamic withdrawal of consideration from the street and activity requests. Both driver tiredness and diversion, in any case, might have the same impacts, i.e., diminished driving execution, longer response time, and an expanded hazard of crash inclusion. appears the piece chart of in general framework. Based on Procurement of video from the camera that's before driver perform real-time preparing of an approaching video stream in arrange to gather the driver's level of weariness on the off chance that the laziness is Estimated at that point it'll deliver the caution by detecting the eyes.

Accidents are inevitable with the growth of the population all around the globe. According to the World Health Organization (WHO), there are various reasons (Sleeping while driving, text while driving, over speed, and alcoholic consumption) for accidents happening despite the situations, according to the World Health Organization (WHO), Around 1.35 million deaths occurring during the road accidents annually. Around 22 countries have implemented and amended the laws to bring them into the best-practice line. There have been various researches conducted to mitigate the accidents to serve the lives and the values. A considerable amount of literature shows that the leading cause of accidents is drowsiness and fatigue. Various studies were conducted to identify issues with several areas, including IoT, Machine learning, and Mobile application. However, the accuracy and the efficiency used in the above researches are not significantly finding the target. To overcome the said problem, this study focuses on a hybrid approach to detect drowsiness and fatigues using deep learning; and IoT is used to avoid accidents effectively. Machine learning has numerous applications in the internet of things (IoT). One of these technological trends is E-care, which brings us information in real-time and continuous patient monitoring with predictive decision making.

CHAPTER 2

1. LITERATURE REVIEW

2.1 Drowsiness Detection System Using Open CV:

ABSTRACT:

More and more professions nowadays require concentration over the long term. Drivers must keep a near eye on the street, so they can respond to sudden occasions quickly. Driver fatigue typically becomes an instantaneous reason behind several traffic accidents. In this manner, there's a ought to create the frameworks that will distinguish and inform a driver of her/his terrible psychophysical condition, which may essentially diminish the number of fatigue-related car mishaps. Developing these devices, though, experiences several difficulties linked to swift and proper identification of the exhaustion signs of a rid er. One of the specialized conceivable outcomes to execute driver tiredness discovery frameworks is to utilize the vision-based approach. Here we are identifying the driver drowsiness by evaluating vision framework of him .

Methodology:

Index Terms - Drowsiness Detection, Image Processing, OpenCV. I. Driver tiredness discovery may be a car security innovation which avoids mishaps when the driver is getting lazy. Different considers have proposed that around 20% of all street mishaps are fatigue-related, up to 50% on certain streets. Driver weakness could be a critical calculate in a expansive number of vehicle mischances . Later measurements assess that yearly 1,200 passings and 76,000 wounds can be credited to weariness related crashes. The improvement of technologies for recognizing or avoiding laziness at the wheel could be a major challenge within the field of accident evasion frameworks. Since of the risk that laziness presents on the street, strategies got to be created for neutralizing its Driver carelessness may be the result of a need of sharpness when driving due to driver tiredness and diversion. Driver diversion happens when an protest or occasion draws a person's consideration absent from the driving errand. Not at all like driver diversion, driver laziness includes no activating occasion but, instep, is characterized by a dynamic withdrawal of consideration from the street and activity requests. Both driver tiredness and diversion, in any case, might have the same impacts, i.e., diminished driving execution, longer response time, and an expanded hazard of crash inclusion. appears the piece chart of in general framework. Based on Procurement of video from the camera that's before driver perform real-time preparing of an approaching video stream in arrange to gather the driver's level of weariness on the off chance that the laziness is Estimated at that point it'll deliver the caution by detecting the eyes.

In this paper, we have built a dowsy driver caution framework merely can execute in various ways. We used OpenCV to identify faces and eyes employing a haar cascade classifier and after

that we utilized a CNN show to foresee the status. This system will be expanded more and include plentiful safety highlights, for example, only a certain amount of people will come to or operate the car. Should an effort to steal the car would arise, the engine of the vehicle will not function or a warning tone can be detected. For an alleged robbery, an image of the burglar is sent to the vehicle's owner that will file a lawsuit against the vehicle's robber

2.2 ACCIDENT MITIGATION SYSTEM WITH DROWSINESS DETECTION: A MACHINE LEARNING AND IOT WITH HYBRID APPROACH

Abstract :

Accidents are unavoidable with population growth around the world. There have been numerous researches conducted to preserve both life and morals. Drowsiness and fatigue have been consistently identified as significant causes of accidents. Instead of relying on limited methods to detect drowsiness and tiredness, this study incorporates deep learning in conjunction with IoT. This study focuses on developing a prototype to minimize road accidents due to drowsiness, fatigue, carelessness, and other reasons. The CNN algorithm handled drowsiness detection; drivers will be notified as soon as they fall asleep. This study takes a novel approach by combining machine learning with drunk avoidance, direction control, speed control, and distance preservation. When paired with proper guidance, the said hybrid approach would produce the best solution to the accident issues without suspects.

Accidents are inevitable with the growth of the population all around the globe. According to the World Health Organization (WHO), there are various reasons (Sleeping while driving, text while driving, over speed, and alcoholic consumption) for accidents happening despite the situations, according to the World Health Organization (WHO), Around 1.35 million deaths occurring during the road accidents annually. Around 22 countries have implemented and amended the laws to bring them into the best-practice line. There have been various researches conducted to mitigate the accidents to serve the lives and the values. A

considerable amount of literature shows that the leading cause of accidents is drowsiness and fatigue. Various studies were conducted to identify issues with several areas, including IoT, Machine learning, and Mobile application. However, the accuracy and the efficiency used in the above researches are not significantly finding the target. To overcome the said problem, this study focuses on a hybrid approach to detect drowsiness and fatigues using deep learning; and IoT is used to avoid accidents effectively. Machine learning has numerous applications in the internet of things (IoT). One of these technological trends is E-care, which brings us information in real-time and continuous patient monitoring with predictive decision making.

This study's primary idea is to mitigate road accidents with drowsiness detection using a machine learning approach. Besides, this study focuses on drunk avoidance, speed control, and collision avoidance with IoT. The drowsiness detection is used with Convolutional Neural Network (CNN) with Keras. While MQ13, MAX30105, and L298N sensors are used to detect drunk, speed, and collision, respectively. In addition to that, L298N is used for path direction control in case of emergency. When the driver is driving the vehicle, a camera placed in the vehicle continuously monitors the driver. The camera found the drowsiness with the trained model; it will give alarm instructions for a specific period. When the driver does not respond, then the vehicle will stop suddenly by IoT sensors. Also, suppose there is any collision situation found. In that case, the sensors measure the distance and control the direction based on the speed and other vehicle's movement. Figure 1 shows the flowchart of the whole system. The flowchart quickly shows that the system will implement based on the hybrid approach of machine learning and IoT.

Initially, a camera placed in the vehicle starts to capture the image of the driver. Once it is captured, then it detects the face. It identifies the eye and applies the classification algorithm whether the eye is open or closed—a deep learning model is applied to classify the status of the driver's eye. The Haar Cascades classifier is used to identify whether the eye is closed or not. Based on the classification threshold value, a convolution neural network (CNN) will identify the driver's drowsiness. The CNN is the most suitable deep learning algorithm for image-based pattern recognition tasks [9]–[12]. A CNN consists of an input layer, an output layer, and one or many hidden information layers. Also, it can capture the Spatial and Temporal dependencies in an image through the application of relevant filters Figure 2: Drowsiness Detection Steps Figure 1: System Flowchart 2021 International Conference on Information Technology (ICIT) 463 Authorized licensed use limited to: University of Glasgow. Downloaded on August 14,2021 at 16:32:47 UTC from IEEE Xplore. Restrictions apply. Figure 3 [13] shows the architecture of the CNN algorithm. It applies the feature learning concept and then uses the classification for objects and image classifications.

1.2 Driver Drowsiness Detection System with OpenCV & TensorFlow

This driver drowsiness detection project is created to prevent accidents. Drowsiness means sleepiness, so it prevents accidents that are caused by drivers who are feeling drowsy or we can say who fell asleep while driving.

So we are creating a Drowsiness detection system that will detect that the person's eyes are closed or open. And if a person's eyes are closed for a few seconds, the system will alert the person by ringing an alert sound.

Our approach to Create Drowsiness detection system:

In this project, for collecting images from webcam we will be using OpenCV and feed these images to our Deep learning model which will classify that the person's eyes is 'Open' or 'Closed'.

- We will take image input from the camera
- Detect face and eyes in the image.
- Create a Region of Interest(ROI), for both detected face and eyes.
- Feed this to our classifier(model), which will categorize whether eyes are open or closed.
- At last, we will calculate the time to check if the person is drowsy or not.

I have used keras to build models using Convolutional Neural Networks(CNN). CNN is a type of deep neural network which performs very accurately and efficiently for image classification. Convolutional Neural Network consists of input layer, output layer, and hidden layers. So this operation is performed on these hidden layers using a filter that performs 2D matrix multiplication on the layers.

The CNN model architecture I have created consist of following layers:

- Convolutional layer; 64 nodes, kernel size 3
- Convolutional layer; 64 nodes, kernel size 3
- Convolutional layer; 64 nodes, kernel size 3
- Fully connected layer; 128 nodes

The output layer is also a fully connected layer with 2 nodes. In all the layers, we use the activation function Relu except the output layer. In the output layer, we have used Soft max as an activation function.

You should have a webcam on your system through which images will be captured. You need to have installed the latest version of python or version (3.6 and above recommended). Now to run this project you are required to install:

OpenCV: to install run following command

2.4 Driver Drowsiness Detection System with OpenCV & Keras

With this Python project, we will be making a drowsiness detection system. 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.

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.

DataFlair has also published other *machine learning project ideas with source code*.

Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving.

The objective of this intermediate 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.

Driver Drowsiness Detection System

In this Python project, we will be using OpenCV for gathering the images from webcam and feed them into a **Deep Learning** model which will classify whether the person's eyes are 'Open' or 'Closed'. The approach we will be using for this Python project is as

follows :

Step 1 – Take image as input from a camera.

Step 2 – Detect the face in the image and create a Region of Interest (ROI).

Step 3 – Detect the eyes from ROI and feed it to the classifier.

Step 4 – Classifier will categorize whether eyes are open or closed.

Step 5 – Calculate score to check whether the person is drowsy.

Driver Drowsiness Detection Dataset

The dataset used for this model is created by us. To create the dataset, we wrote a script that captures eyes from a camera and stores in our local disk. We separated them into their respective labels ‘Open’ or ‘Closed’. The data was manually cleaned by removing the unwanted images which were not necessary for building the model. The data comprises around 7000 images of people’s eyes under different lighting conditions. After training the model on our dataset, we have attached the final weights and model architecture file “models/cnnCat2.h5”.

Now, you can use this model to classify if a person’s eye is open or closed.

Alternatively, if you want to build and train your own model, you can download the dataset: **Driver Drowsiness Dataset**

The Model Architecture

The model we used is built with Keras using **Convolutional Neural Networks (CNN)**. A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter.

The CNN model architecture consists of the following layers:

- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 64 nodes, kernel size 3
- Fully connected layer; 128 nodes

The final layer is also a fully connected layer with 2 nodes. A Relu activation function is used in all the layers except the output layer in which we used Soft max.

Chapter 3

3 System design

EXISTING SYSTEM:

- The paper's motive is to demonstrate the implementation of a driver's drowsiness detection using MATLAB through image processing. Studies suggest that about one-fourth of all serious road accidents occur because of the vehicle drivers' drowsiness where they need to take, confirming that drowsiness gives rise to more road accidents than accidents occur through Drink and Drive.
- Drowsiness Detection System is designed by employing vision-based concepts. The system's main component is a small camera pointing towards the driver's face scan and monitoring the driver's eyes to detect drowsiness
- There are various methods like detecting objects which are near to vehicle and front and rear cameras for detecting vehicles approaching near to vehicle and airbag system which can save lives after an accident is accorded

PROBLEM STATEMENT

- Most of the existing systems use external factors and inform the user about the problem and save users after an accident is accord but from research most of the accidents are due to faults in users like drowsiness, sleeping while driving.
- These methods can't able to detect the facial expressions, yawning head nods, and majorly on eye blink frequencies.
- Accuracy of the existing method is not good when compared to the proposed model

PROPOSED SYSTEM:

- Most of them in a few ways relate to highlights of the eye (ordinarily reflections from the eye) inside a video picture of the driver.
- The first point of this venture was to utilize the retinal reflection as a implies to finding the eyes on the confront, and after that utilizing the nonappearance of this reflection as a way of identifying when the eyes are closed.
- Applying this calculation on sequential video outlines may help within the calculation of eye closure period.
- Eye closure period for lazy drivers are longer than ordinary blinking. It is additionally exceptionally small longer time may result in extreme crash. So, we'll caution the driver immediately as closed eye is identified.

ADVANTAGES:

- This method will detect a problem before any problem accord and inform the driver and other passengers by raising an alert.
- In this OpenCV based machine learning techniques are used for automatic detection of drowsiness.
- Automatic driver mode is started.
- Alarm will ring.

Block diagram:

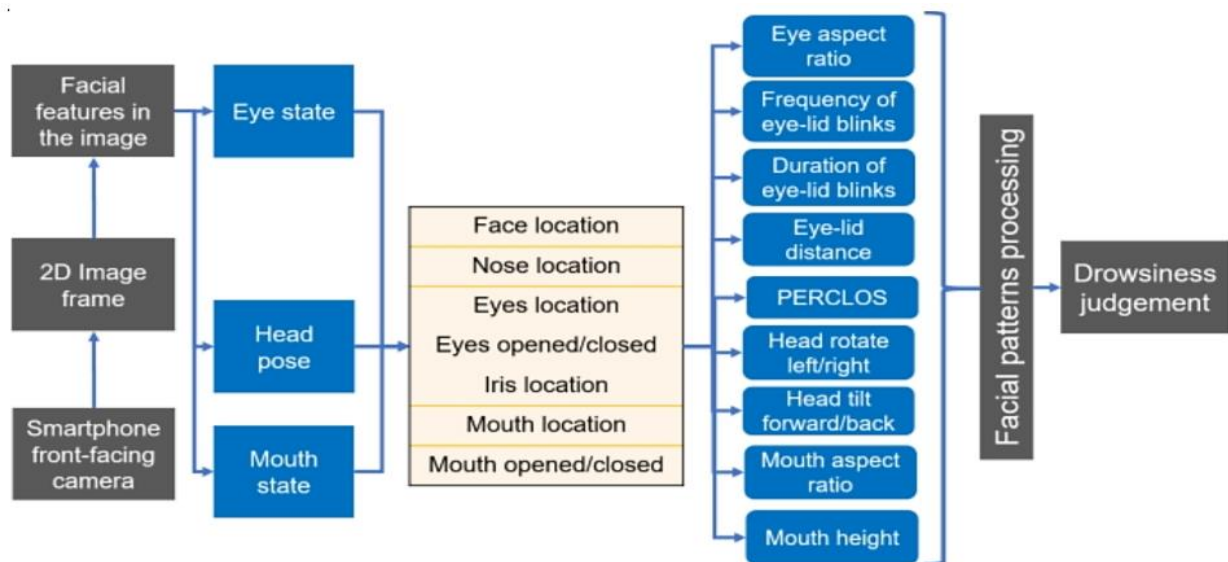


Fig. 1. General scheme for drowsiness state recognition

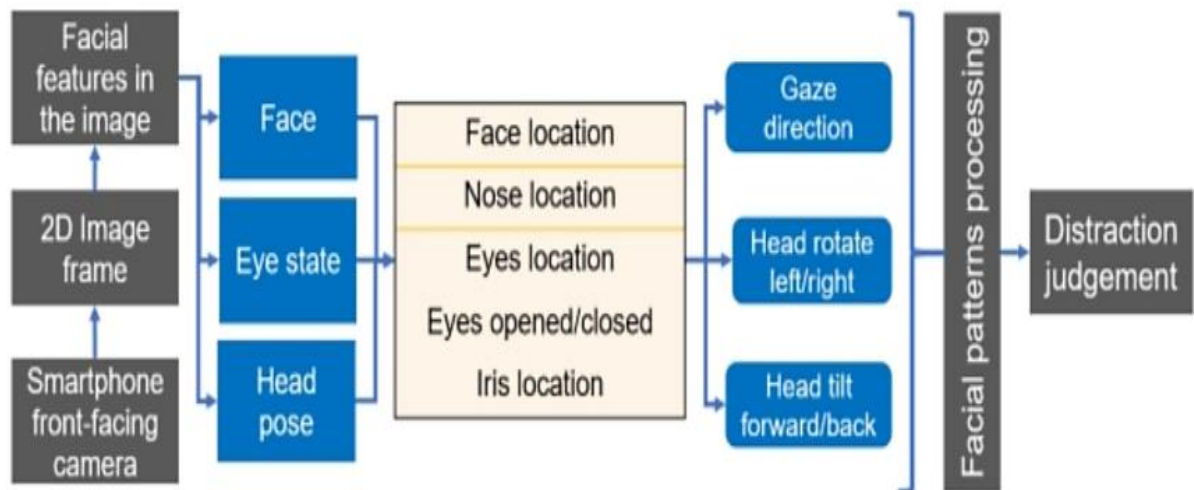
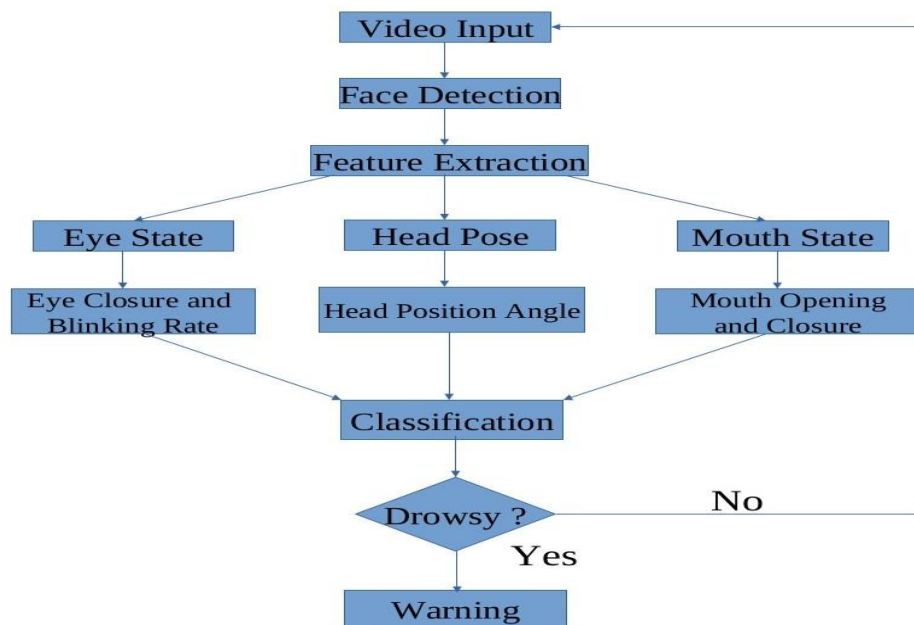


Fig. 2. General scheme for distraction state recognition

Flow diagram:



WORKING PRINCIPLE:

- There are various approaches to improve the street security for a vehicle driver. It ought to be noticed that one of the gigantic mainstream approaches introduced in past logical explores depends in the improvement of cutting-edge driver help frameworks.
- These wellbeing frameworks permit to diminish street mishaps and furnish better collaboration and commitment with a driver. Some normal instances of driver security advancements for this sort of frameworks are vehicle impact shirking framework, path keep partner, driver laziness and interruption observing what's more, cautioning.
- General utilization of such frameworks can be depicted as a certain arrangement of sequential orders along these lines: Observing driver conduct, condition of the vehicle or street circumstance by utilizing distinctive inherent assistant gadgets, including short and long range radars, lasers, lidars, video stream cameras to see the environmental factors; ceaseless investigation of readings from sensors and deciding risky circumstances while driving; cautioning driver about perceived perilous in-lodge and street circumstances; and taking control of the vehicle if driver response isn't adequate or missing.
- Right now, driver security frameworks vigorously depend on information gathered from various in-vehicle sensors.

SOFTWARE REQUIREMENTS

- Operating system: Windows 7.
- Coding Language: python
- Tool: pycham
- Libraries: OpenCV, argparse, imutils, math, dlib, pygam

HARDWARE REQUIREMENTS

- Processor: Pentium processor of 400 MHz or higher.
- RAM: minimum 64MB primary memory
- Hard disk: minimum 1 GB hard disk space.
- Monitor : preferable color monitor (16 bit color) and above.
- Web camara
- Compact disk drive
- A keybord and mouse.

CHAPTER 4

Machine learning

What are the 7 steps of machine learning?

7 Steps of Machine Learning

- Step 1: Gathering Data. ...
- Step 2: Preparing that Data. ...
- Step 3: Choosing a Model. ...
- Step 4: Training. ...
- Step 5: Evaluation. ...
- Step 6: Hyper parameter Tuning. ...
- Step 7: Prediction.

Introduction

In this blog, we will discuss the workflow of a Machine learning project this includes all the steps required to build the proper machine learning project from scratch. We will also go over data pre-processing, data cleaning, feature exploration and feature engineering and show the impact that it has on Machine Learning Model Performance. We will also cover a couple of the pre-modelling steps that can help to improve the model performance.

Python Libraries that would be need to achieve the task:

1. Numpy
2. Pandas
3. Sci-kit Learn
4. Matplotlib

Understanding the machine learning workflow

We can define the machine learning workflow in 3 stages.

1. Gathering data
2. Data pre-processing
3. Researching the model that will be best for the type of data
4. Training and testing the model
5. Evaluation

Okay but first let's start from the basics

What is the machine learning Model?

The machine learning model is nothing but a piece of code; an engineer or data scientist makes it smart through training with data. So, if you give garbage to the model, you will get garbage in return, i.e. the trained model will provide false or wrong predictions.

1Gathering Data

The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the

analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.

We can also use some free data sets which are present on the internet. Kaggle and UCI Machine learning Repository are the repositories that are used the most for making Machine learning models. Kaggle is one of the most visited websites that is used for practicing machine learning algorithms, they also host competitions in which people can participate and get to test their knowledge of machine learning.

2. Data pre-processing

Data pre-processing is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately. In machine learning, there is an 80/20 rule. Every data scientist should spend 80% time for data pre-processing and 20% time to actually perform the analysis.

What is data pre-processing?

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

Why do we need it?

As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we definitely need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects.

Most of the real-world data is messy, some of these types of data are:

1. **Missing data:** Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).
2. **Noisy data:** This type of data is also called outliers, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.
3. **Inconsistent data:** This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

Three Types of Data

1. Numeric e.g. income, age
2. Categorical e.g. gender, nationality
3. Ordinal e.g. low/medium/high

How can data pre-processing be performed?

These are some of the basic pre — processing techniques that can be used to convert raw data.

1. **Conversion of data:** As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.

2. **Ignoring the missing values:** Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.

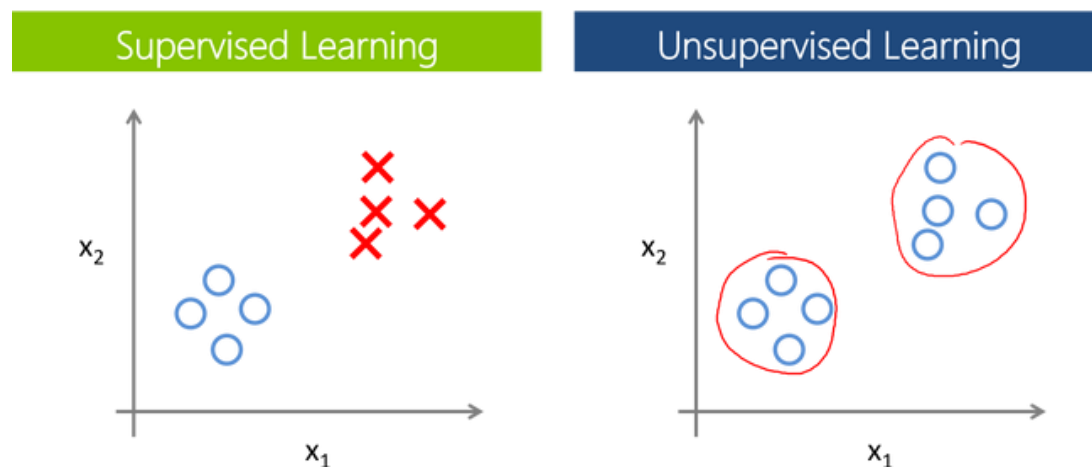
3. **Filling the missing values:** Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.

4. **Machine learning:** If we have some missing data then we can predict what data shall be present at the empty position by using the existing data.

5. **Outliers detection:** There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800 Kg; due to mistyping of extra 0]

Researching the model that will be best for the type of data

Our main goal is to train the best performing model possible, using the pre-processed data.



Supervised Learning:

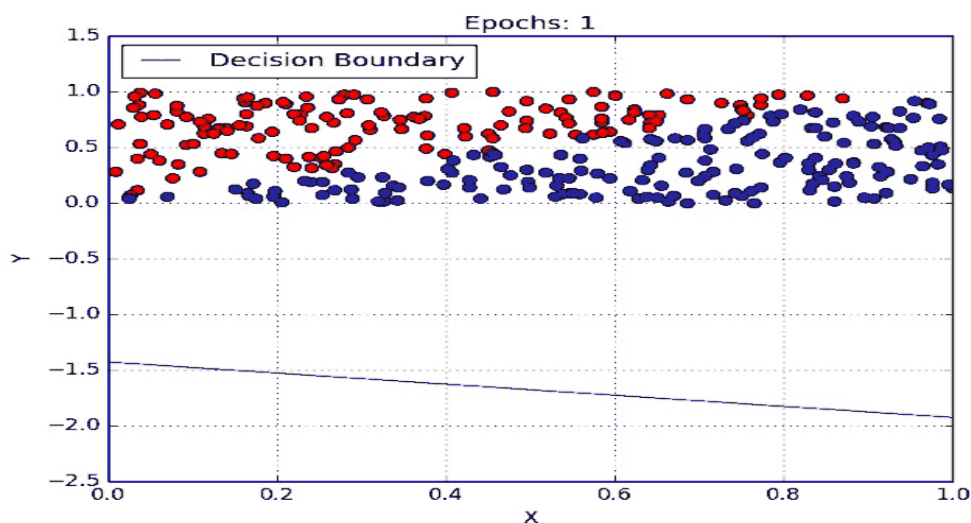
In Supervised learning, an AI system is presented with data which is labelled, which means that each data tagged with the correct label.

The supervised learning is categorized into 2 other categories which are “**Classification**” and “**Regression**”.

Classification:

Classification problem is when the target variable is **categorical** (i.e. the output could be classified into classes — it belongs to either Class A or B or something else).

A classification problem is when the output variable is a category, such as “red” or “blue” , “disease” or “no disease” or “spam” or “not spam”.



As shown in the above representation, we have 2 classes which are plotted on the graph i.e. red and blue which can be represented as ‘setosa flower’ and ‘versicolor flower’, we can image the X-

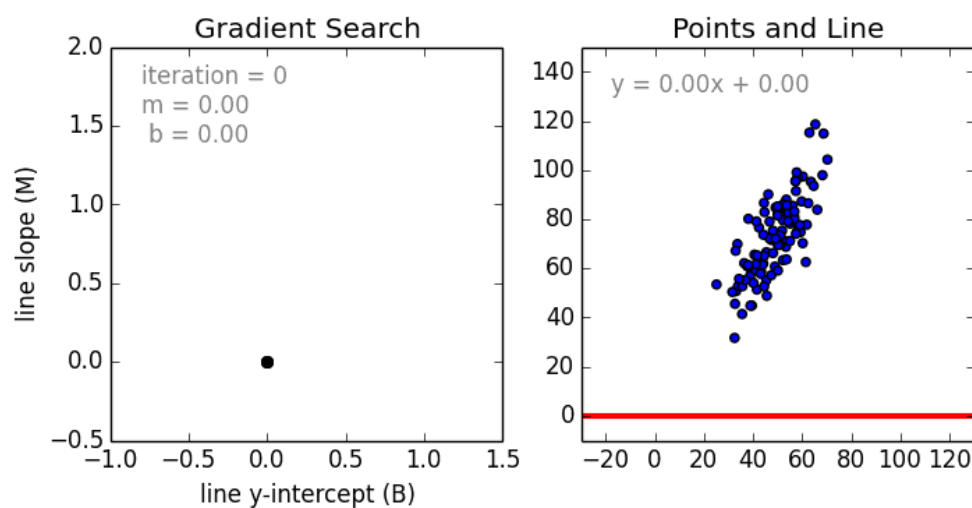
axis as the 'Sepal Width' and the Y-axis as the 'Sepal Length', so we try to create the best fit line that separates both classes of flowers.

These some most used classification algorithms.

- K-Nearest Neighbor
- Naive Bayes
- Decision Trees/Random Forest
- Support Vector Machine
- Logistic Regression

Regression:

While a **Regression** problem is when the target variable is **continuous** (i.e. the output is numeric).

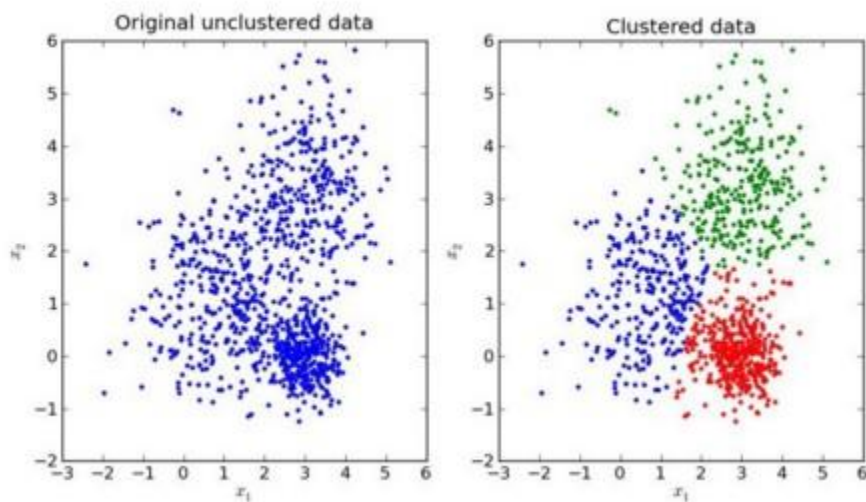


As shown in the above representation, we can imagine that the graph's X-axis is the 'Test scores' and the Y-axis represents 'IQ'. So we try to create the best fit line in the given graph so that we can use that line to predict any approximate IQ that isn't present in the given data.

These some most used regression algorithms.

- Linear Regression
- Support Vector Regression
- Decision Tress/Random Forest
- Gaussian Progresses Regression
- Ensemble Methods

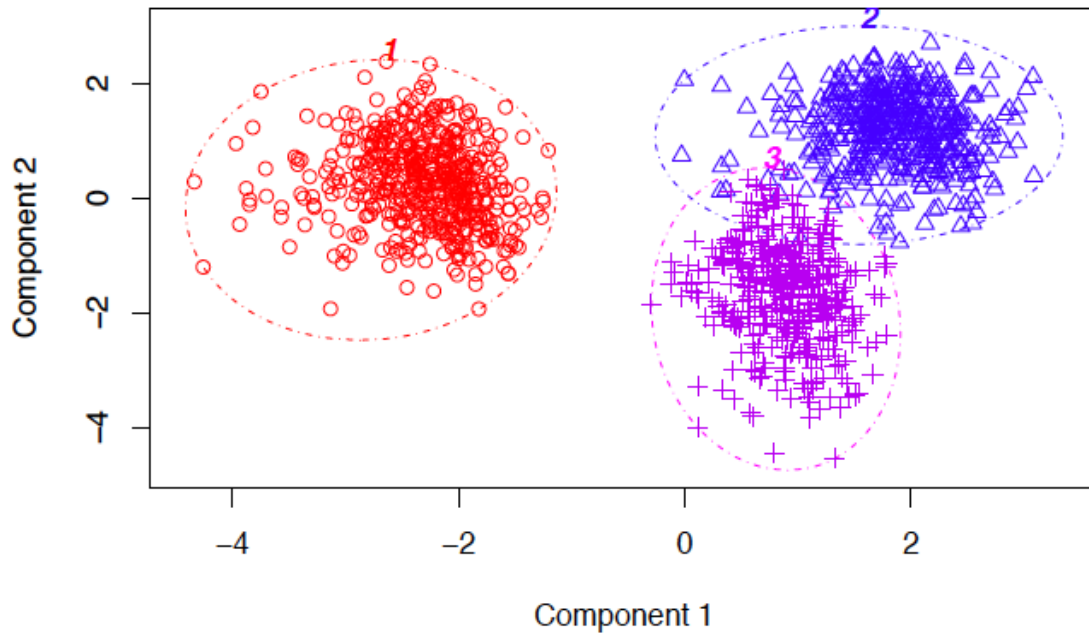
Unsupervised Learning:



The unsupervised learning is categorized into 2 other categories which are “**Clustering**” and “**Association**”.

Clustering:

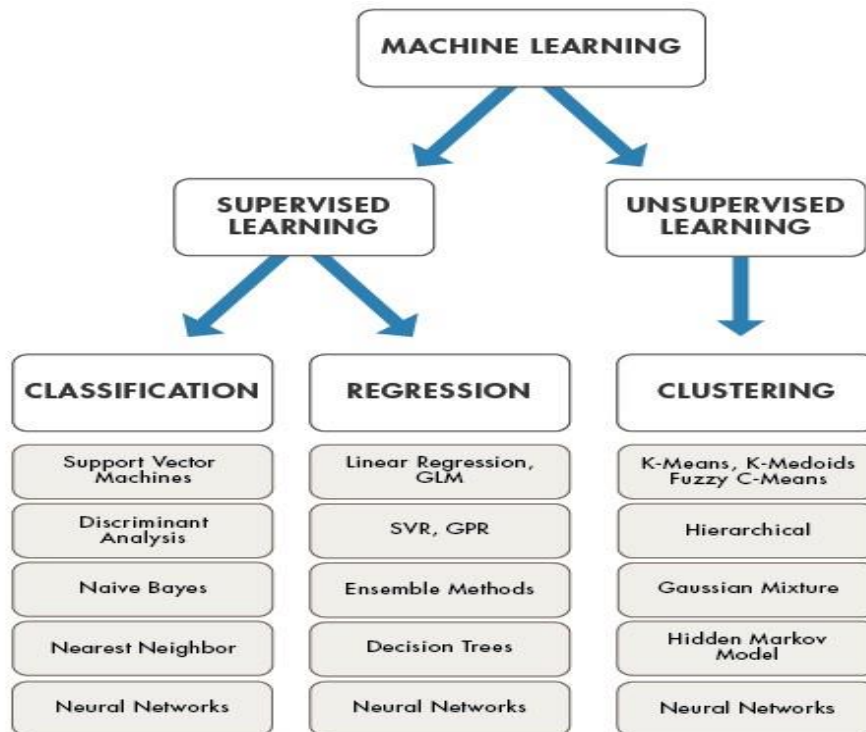
A set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.



Methods used for clustering are:

- Gaussian mixtures
- K-Means Clustering
- Boosting
- Hierarchical Clustering
- K-Means Clustering
- Spectral Clustering

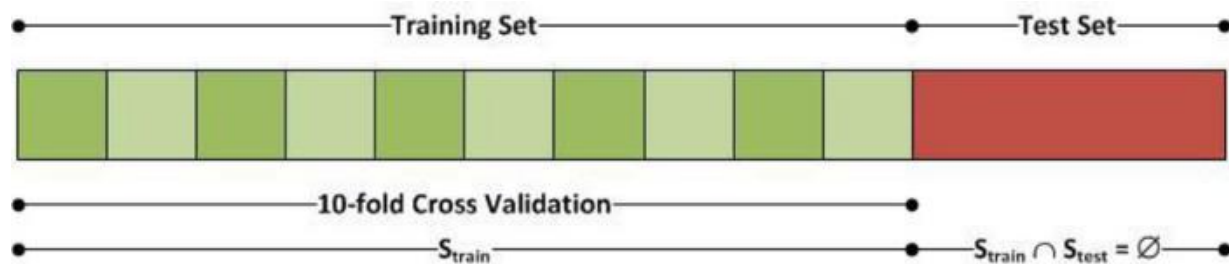
Overview of models under categories:



4. Training and testing the model on data

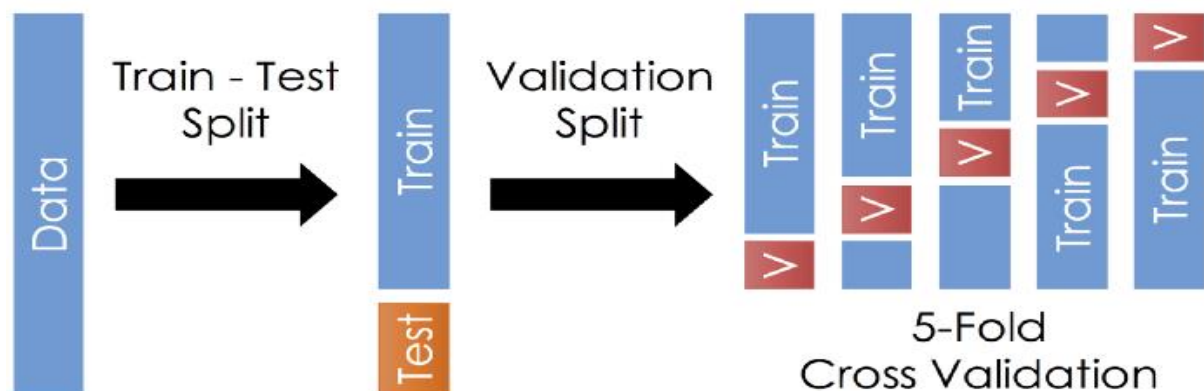
For training a model we initially split the model into 3 three sections which are ‘**Training data**’, ‘**Validation data**’ and ‘**Testing data**’.

You train the classifier using ‘**training data set**’, tune the parameters using ‘**validation set**’ and then test the performance of your classifier on unseen ‘**test data set**’. An important point to note is that during training the classifier only the training and/or validation set is available. The test data set must not be used during training the classifier. The test set will only be available during testing the classifier.



Training set: The training set is the material through which the computer learns how to process information. Machine learning uses algorithms to perform the training part. A set of data used for learning, that is to fit the parameters of the classifier.

Validation set: Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. A set of unseen data is used from the training data to tune the parameters of a classifier.



Once the data is divided into the 3 given segments we can start the training process.

In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use 'test set')

instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

The model uses any one of the models that we had chosen in step 3/ point 3. Once the model is trained we can use the same trained model to predict using the testing data i.e. the unseen data. Once this is done we can develop a confusion matrix, this tells us how well our model is trained. A confusion matrix has 4 parameters, which are '**True positives**', '**True Negatives**', '**False Positives**' and '**False Negative**'. We prefer that we get more values in the True negatives and true positives to get a more accurate model. The size of the Confusion matrix completely depends upon the number of classes.

n=165	Predicted: NO	Predicted: YES
	Actual: NO	Actual: YES
	50	10
	5	100

- **True positives** : These are cases in which we predicted TRUE and our predicted output is correct.
- **True negatives** : We predicted FALSE and our predicted output is correct.
- **False positives** : We predicted TRUE, but the actual predicted output is FALSE.
- **False negatives** : We predicted FALSE, but the actual predicted output is TRUE.

We can also find out the accuracy of the model using the confusion matrix.

$$\text{Accuracy} = (\text{True Positives} + \text{True Negatives}) / (\text{Total number of classes})$$

i.e. for the above example:

$$\text{Accuracy} = (100 + 50) / 165 = 0.9090 \text{ (90.9\% accuracy)}$$

5. Evaluation

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.

To improve the model we might tune the hyper-parameters of the model and try to improve the accuracy and also looking at the confusion matrix to try to increase the number of true positives and true negatives.

Conclusion

In this blog, we have discussed the workflow a Machine learning project and gives us a basic idea of how a should the problem be tackled.

CHAPTER 4

EYE ASPECT RATIO:

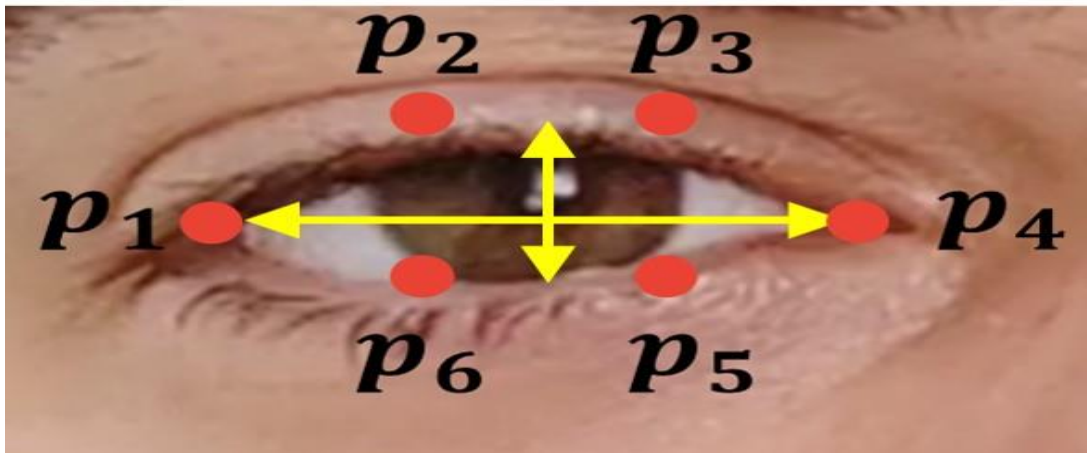
The Eye Aspect Ratio is an estimate of the eye opening state.

the eye aspect ratio can be defined by the below equation. ... A program can determine if a person's eyes are closed if the Eye Aspect Ratio falls below a certain threshold.

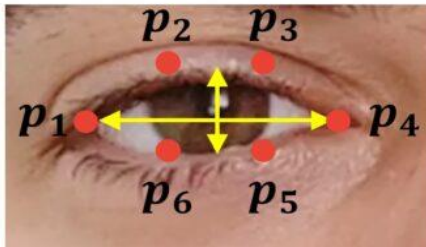
Clmtrackr is another facial landmark plotter FORMULA

FORMULA

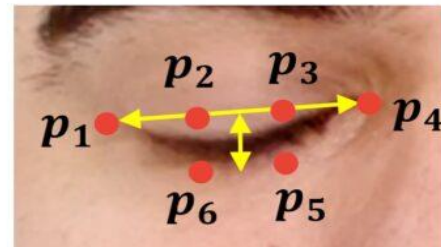
$$EAR = \frac{\|P2-P6\| + \|P3-P5\|}{2\|P1-P4\|}$$



Eye aspect ratio will be larger and relatively constant over time when eye is open



Eye aspect ratio will be almost equal to zero when a blink occurs



Facial Landmark:

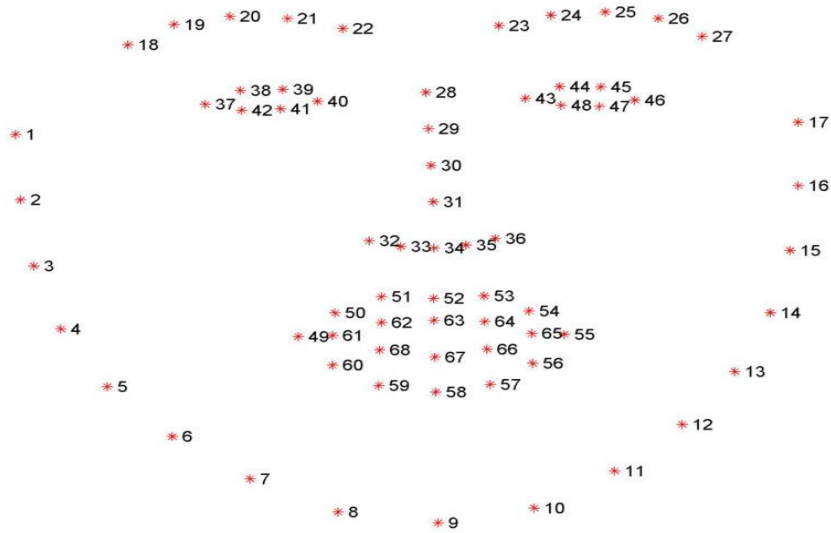
Facial Landmark- It is a inbuilt HOG SVM classifier used to determine the position of 68(x, y) coordinates that map to facial structures on the face

It is mainly used for image or video processing and also analysis including object detection, face detection, etc. Facial landmarks are used to localize and represent important regions of the face, such as: · Mouth. · Eyes. · Eyebrows.

Facial Landmark:

Facial Landmark- It is a inbuilt HOG SVM classifier used to determine the position of 68(x, y) coordinates that map to facial structures on the face

It is mainly used for image or video processing and also analysis including object detection, face detection, etc. Facial landmarks are used to localize and represent important regions of the face, such as: · Mouth. · Eyes. · Eyebrows.

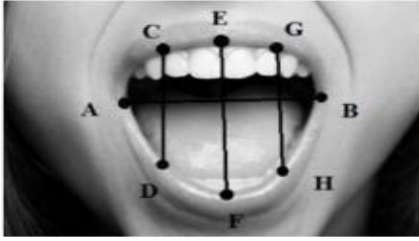


MAR (MOUTH ASPECT RATIO)

Using this concept, we have calculated the Mouth Aspect Ratio: Representing the face with 68-(x,y) coordinates. As we see that the mouth is represented by a set of 20-(x,y) coordinates. So, we have used coordinates 62, 64, 66, and 68 to calculate the distance between them in the same way as EAR Calculation.

$$\text{MAR} = \frac{|CD| + |EF| + |GH|}{3 * |AB|}$$

$$3 * |AB|$$



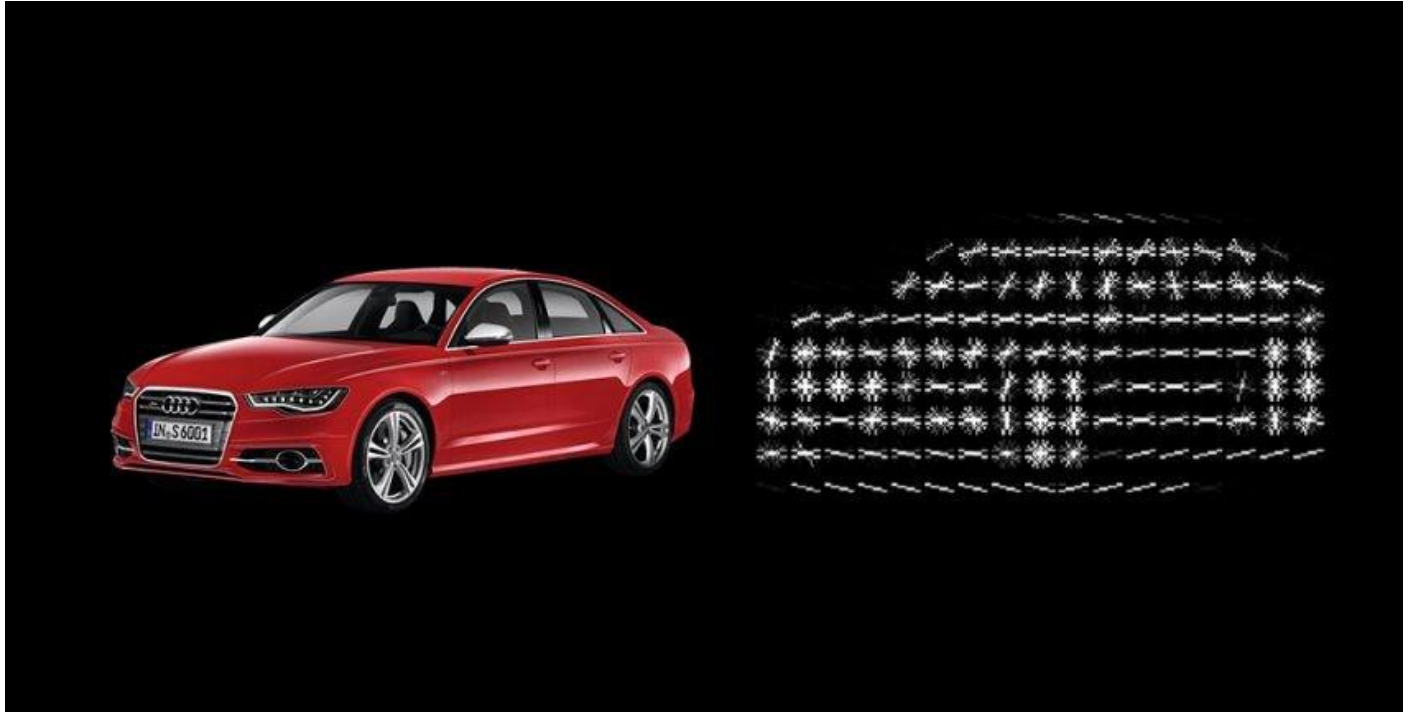
$$\text{MAR} = \frac{|\text{EF}|}{|\text{AB}|}$$

Introduction

Feature engineering is a game-changer in the world of machine learning algorithms. It's actually one of my favorite aspects of being a data scientist! This is where we get to experiment the most – to engineer new features from existing ones and improve our model's performance.

Some of the top data scientists in the world rely on feature engineering to boost their leaderboard score in hackathons. I'm sure you would even have used various feature engineering techniques on structured data.

Can we extend this technique to unstructured data, such as images? It's an intriguing riddle for [computer vision](#) enthusiasts and one we will solve in this article. Get ready to perform feature engineering in the form of feature extraction on image data!



here are actually multiple techniques for feature extraction. I covered three basic ones in my previous article which you should read before proceeding:

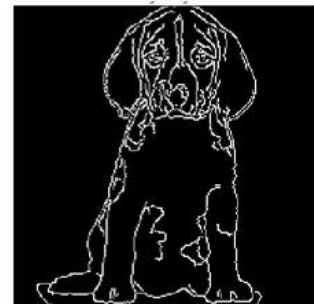
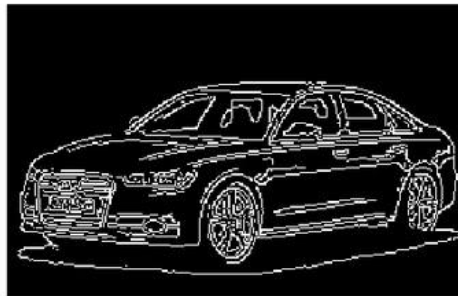
What is a Feature Descriptor?

You might have had this question since you read the heading. So let's clear that up first before we jump into the HOG part of the article.

Take a look at the two images shown below. Can you differentiate between the objects in the image?



We can clearly see that the right image here has a dog and the left image has a car. Now, let me make this task slightly more complicated – identify the objects shown in the image below:



Still easy, right? Can you guess what was the difference between the first and the second case? The first pair of images had a lot of information, like the shape of the object, its color, the edges, background, etc.

On the other hand, the second pair had much less information (only the shape and the edges) but it was still enough to differentiate the two images.

Do you see where I am going with this? We were easily able to differentiate the objects in the second case because it had the necessary information we would need to identify the object. And that is exactly what a feature descriptor does:

There are a number of feature descriptors out there. Here are a few of the most popular ones:

- HOG: Histogram of Oriented Gradients
- SIFT: Scale Invariant Feature Transform
- SURF: Speeded-Up Robust Feature

In this article, we are going to focus on the HOG feature descriptor and how it works. Let's get started!

introduction to the HOG Feature Descriptor

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. It is widely used in computer vision tasks for object detection.

Let's look at some important aspects of HOG that makes it different from other feature descriptors:

- The HOG descriptor focuses on the structure or the shape of an object. Now you might ask, how is this different from the edge features we extract for images? In the case of edge features, we only identify if the pixel is an edge or not. HOG is able to provide the edge direction as well. This is done by extracting the **gradient and orientation** (or you can say magnitude and direction) of the edges
- Additionally, these orientations are calculated in **'localized' portions**. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated. We will discuss this in much more detail in the upcoming sections
- Finally the HOG would generate a **Histogram** for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'

To put a formal definition to this:

Implementing HOG using tools like OpenCV is extremely simple. It's just a few lines of code since we have a predefined function called ***hog*** in the ***skimage.feature*** library. Our focus in this article, however, is on how these features are actually calculated.

Process of Calculating the Histogram of Oriented Gradients (HOG)

We should now have a basic idea of what a HOG feature descriptor is. It's time to delve into the core idea behind this article. Let's discuss the step-by-step process to calculate HOG.

Consider the below image of size (180 x 280). Let us take a detailed look at how the HOG features will be created for this image:



Step 1: Preprocess the Data (64 x 128)

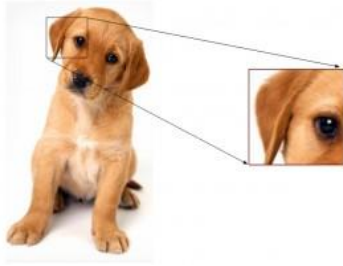
This is a step most of you will be pretty familiar with. Preprocessing data is a crucial step in any machine learning project and that's no different when working with images.

We need to preprocess the image and bring down the width to height ratio to 1:2. The image size should preferably be 64 x 128. This is because we will be dividing the image into 8*8 and 16*16 patches to extract the features. Having the specified size (64 x 128) will make all our calculations pretty simple. In fact, this is the exact value used in the original paper.

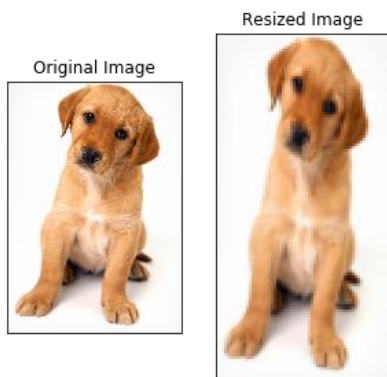
Coming back to the example we have, let us take the size 64 x 128 to be the standard image size for now. Here is the resized image:

Step 2: Calculating Gradients (direction x and y)

The next step is to calculate the gradient for every pixel in the image. **Gradients are the small change in the x and y directions.** Here, I am going to take a small patch from the image and calculate the gradients on that:



We will get the pixel values for this patch. Let's say we generate the below pixel matrix for the given patch (*the matrix shown here is merely used as an example and these are not the original pixel values for the given patch*):



121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Source: Applied Machine Learning Course

I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x-direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above the selected pixel.

Hence the resultant gradients in the x and y direction for this pixel are:

- Change in X direction(G_x) = $89 - 78 = 11$
- Change in Y direction(G_y) = $68 - 56 = 8$

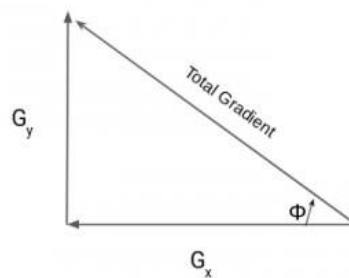
This process will give us two new matrices – one storing gradients in the x-direction and the other storing gradients in the y direction. This is similar to using a Sobel Kernel of size 1. **The magnitude would be higher when there is a sharp change in intensity, such as around the edges.**

We have calculated the gradients in both x and y direction separately. The same process is repeated for all the pixels in the image. The next step would be to find the magnitude and orientation using these values.

Step 3: Calculate the Magnitude and Orientation

Using the gradients we calculated in the last step, we will now determine the magnitude and direction for each pixel value. For this step, we will be using the Pythagoras theorem (yes, the same one which you studied back in school!).

Take a look at the image below:



The gradients are basically the base and perpendicular here. So, for the previous example, we had G_x and G_y as 11 and 8. Let's apply the Pythagoras theorem to calculate the total gradient magnitude:

$$\text{Total Gradient Magnitude} = \sqrt{[(G_x)^2 + (G_y)^2]}$$

$$\text{Total Gradient Magnitude} = \sqrt{[(11)^2 + (8)^2]} = 13.6$$

Next, calculate the orientation (or direction) for the same pixel. We know that we can write the tan for the angles:

$$\tan(\Phi) = G_y / G_x$$

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Magnitude		1							
Bin	0	20	40	60	80	100	120	140	160

Hence, the value of the angle would be:

$$\Phi = \text{atan}(G_y / G_x)$$

The orientation comes out to be 36 when we plug in the values. So now, for every pixel value, we have the total gradient (magnitude) and the orientation (direction). We need to generate the histogram using these gradients and orientations.

But hang on – we need to take a small break before we jump into how histograms are created in the HOG feature descriptor. Consider this a small step in the overall process. And we'll start this by discussing some simple methods of creating Histograms using the two values that we have – gradients and orientation.

Different Methods to Create Histograms using Gradients and Orientation

A histogram is a plot that shows the frequency distribution of a set of continuous data. We have the variable (in the form of bins) on the x-axis and the frequency on the y-axis. Here, we are going to take the angle or orientation on the x-axis and the frequency on the y-axis.

Method 1:

Let us start with the simplest way to generate histograms. We will take each pixel value, find the orientation of the pixel and update the frequency table.

ere is the process for the highlighted pixel (85). Since the orientation for this pixel is 36, we will add a number against angle value 36, denoting the frequency:

						121	10	78	96	125									
						48	152	68	125	111									
						145	78	85	89	65									
						154	214	56	200	66									
						214	87	45	102	45									

Frequency							1												
Angle	1	2	3	4 ...	35	36	37	38	39....		175	176	177	178	179	180			

Source: [Applied Machine Learning Course](#)

The same process is repeated for all the pixel values, and **we end up with a frequency table that denotes angles and the occurrence of these angles in the image**. This frequency table can be used to generate a histogram with angle values on the x-axis and the frequency on the y-axis.

That's one way to create a histogram. Note that here the bin value of the histogram is 1. Hence we get about 180 different buckets, each representing an orientation value. Another method is to create the histogram features for higher bin values.

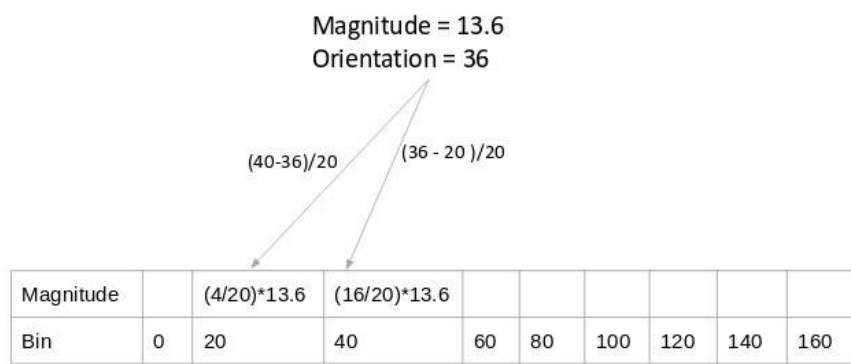
Method 2:

This method is similar to the previous method, except that here we have a bin size of 20. So, the number of buckets we would get here is 9.

Again, for each pixel, we will check the orientation, and store the frequency of the orientation values in the form of a 9 x 1 matrix. Plotting this would give us the histogram:

Source: [Applied Machine Learning Course](#)

Method 3:



The histograms created in the HOG feature descriptor are not generated for the whole image. Instead, the image is divided into 8×8 cells, and the histogram of oriented gradients is computed for each cell. Why do you think this happens?

By doing so, we get the features (or histogram) for the smaller patches which in turn represent the whole image. We can certainly change this value here from 8×8 to 16×16 or 32×32 .

If we divide the image into 8×8 cells and generate the histograms, we will get a 9×1 matrix for each cell. This matrix is generated using method 4 that we discussed in the previous section.



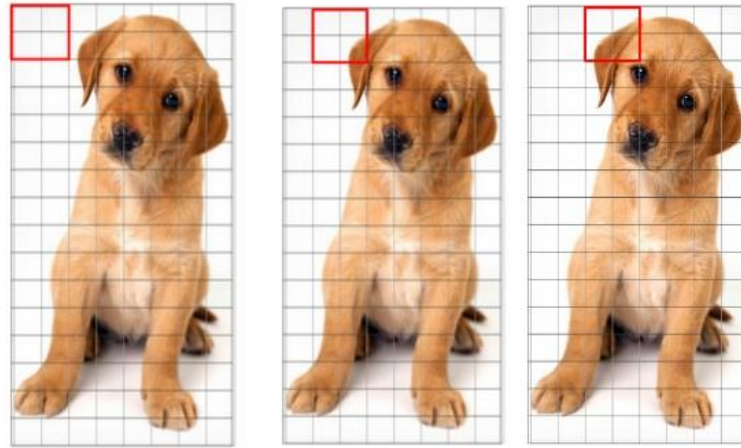
Once we have generated the HOG for the 8×8 patches in the image, the next step is to normalize the histogram.

Step 5: Normalize gradients in 16×16 cell (36×1)

Before we understand how this is done, it's important to understand why this is done in the first place.

Although we already have the HOG features created for the 8×8 cells of the image, the gradients of the image are sensitive to the overall lighting. This means that for a particular picture, some portion of the image would be very bright as compared to the other portions.

We cannot completely eliminate this from the image. But we can reduce this lighting variation by normalizing the gradients by taking 16×16 blocks. Here is an example that can explain how 16×16 blocks are created:



Here, we will be combining four 8×8 cells to create a 16×16 block. And we already know that each 8×8 cell has a 9×1 matrix for a histogram. So, we would have four 9×1 matrices or a single 36×1 matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector V :

$$V = [a_1, a_2, a_3, \dots, a_{36}]$$

We calculate the root of the sum of squares:

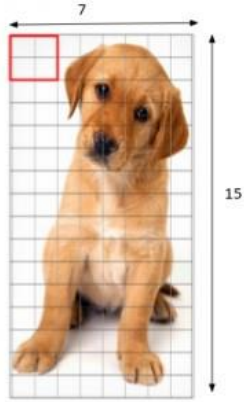
$$k = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2 + \dots + (a_{36})^2}$$

And divide all the values in the vector V with this value k :

$$\text{Normalised Vector} = \left(\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \dots, \frac{a_{36}}{k} \right)$$

The resultant would be a normalized vector of size 36×1 .

Step 6: Features for the complete image



We are now at the final step of generating HOG features for the image. So far, we have created features for 16×16 blocks of the image. Now, we will combine all these to get the features for the final image.

Can you guess what would be the total number of features that we will have for the given image? We would first need to find out how many such 16×16 blocks would we get for a single 64×128 image:

We would have 105 (7×15) blocks of 16×16 . Each of these 105 blocks has a vector of 36×1 as features. Hence, the total features for the image would be $105 \times 36 \times 1 = 3780$ features.

We will now generate HOG features for a single image and verify if we get the same number of features at the end.

Implementing HOG Feature Descriptor in Python

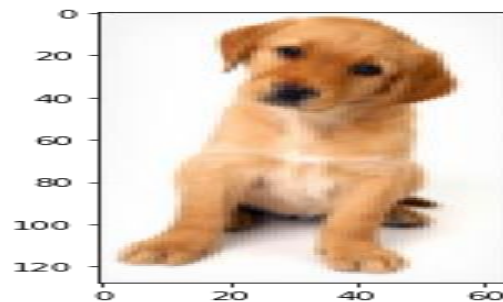
Time to fire up Python! This, I'm sure, is the most anticipated section of this article. So let's get rolling.

We will see how we can generate HOG features on a single image, and if the same can be applied on a larger dataset. We will first load the required libraries and the image for which we are going to create the HOG features:

(663, 459, 3)



We can see that the shape of the image is 663 x 459. We will have to resize this image into 64 x 128. Note that we are using *skimage* which takes the input as height x width.

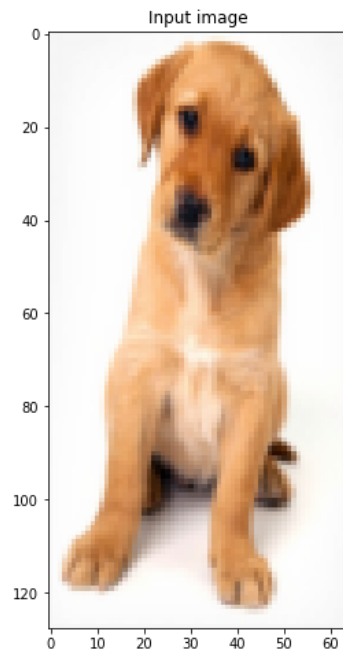


Here, I am going to use the `hog` function from *skimage.features* directly. So we don't have to calculate the gradients, magnitude (total gradient) and orientation individually. The `hog` function would internally calculate it and return the feature matrix.

Also, if you set the parameter '`visualize = True`', it will return an image of the HOG.

Before going ahead, let me give you a basic idea of what each of these hyperparameters represents. Alternatively, you can check the definitions from the official documentation [here](#).

- The *orientations* are the number of buckets we want to create. Since I want to have a 9 x 1 matrix, I will set the orientations to 9



- *pixels_per_cell* defines the size of the cell for which we create the histograms. In the example we covered in this article, we used 8 x 8 cells and here I will set the same value. As mentioned previously, you can choose to change this value
- We have another hyperparameter *cells_per_block* which is the size of the block over which we normalize the histogram. Here, we mention the cells per blocks and not the number of pixels. So, instead of writing 16 x 16, we will use 2 x 2 here

The feature matrix from the function is stored in the variable *fd*, and the image is stored in *hog_image*. Let us check the shape of the feature matrix:

As expected, we have 3,780 features for the image and this verifies the calculations we did in step 7 earlier. You can choose to change the values of the hyperparameters and that will give you a feature matrix of different sizes.

Let's finally look at the HOG image:

End Notes

The idea behind this article was to give you an understanding of what is actually happening behind the HOG feature descriptor and how the features are calculated. The complete process is broken down into 7 simple steps.

Opencv

What is OpenCV in machine learning?

OpenCV (**Open Source Computer Vision Library**) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products

Can OpenCV be used for machine learning?

OpenCV can also be used for doing some **machine learning tasks**. For example, you can train an SVM model, Logistic Regression model or Bag Of Visual Words model in OpenCV.

What is OpenCV and how does it work?

OpenCV is the huge **open-source library for the computer vision, machine learning, and image processing** and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human

s OpenCV a deep learning framework?

Two weeks ago OpenCV 3.3 was officially released, bringing with it a highly improved deep learning (dnn) module. This module now supports a number of deep learning frameworks, including Caffe, TensorFlow, and Torch/PyTorch.

OpenCV an algorithm?

Why use OpenCV for Computer Vision Tasks? OpenCV, or Open Source Computer Vision library, started out as a research project at Intel. It's currently the largest computer vision library in terms of the sheer number of functions it holds. **OpenCV contains implementations of more than 2500 algorithms**

What is OpenCV project?

OpenCV (Open Source Computer Vision Library) is **a library of programming functions mainly aimed at real-time computer vision**. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel).

Is OpenCV a Python library?

OpenCV-Python is a **library of Python bindings designed to solve computer vision problems**. ... This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++.

Is OpenCV easy?

The truth is that learning OpenCV used to **be quite challenging**. The documentation was hard to navigate. The tutorials were hard to follow and incomplete. And even some of the books were a bit tedious to work through.

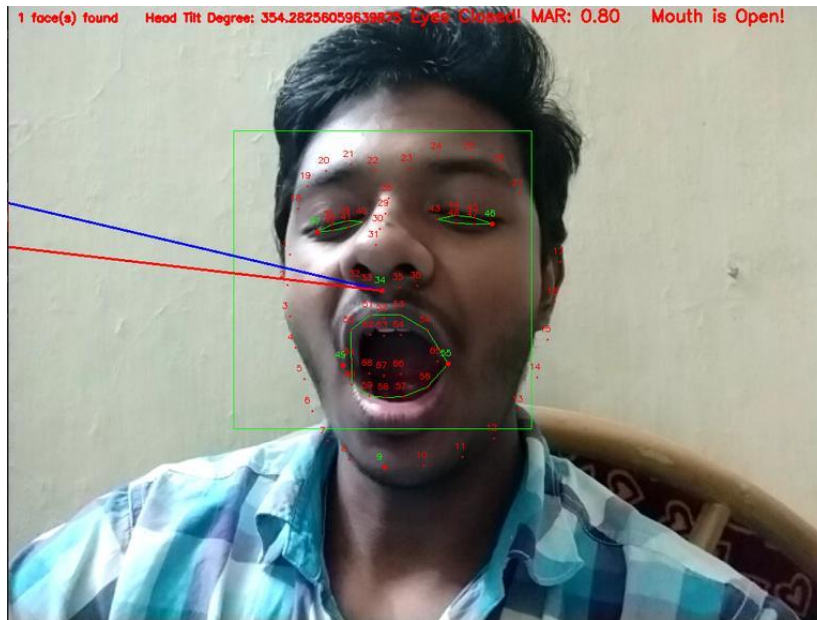
Chapter 6

Future scope:

- The model can be improved incrementally by using other parameter blinking rate ,state of the cars , etc.
- If all these parameter are used it can improve the accuracy by lot.
- We plan to future work on the project by adding a sensor to o track the heart rate in order to prevent the accident caused due to sudden heart attack to driver.
- Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when is asleep and stop the video accordingly .
- It can also be used in application that prevent user from sleeping

RESULTS AND DISCUSSION:

- The driver anomaly observing framework created is able of identifying laziness, intoxicated and careless practices of driver in a brief time.
- The Laziness Detecting Framework created based on eye closure of the driver can separate ordinary eye flicker and tiredness and distinguish the laziness while driving.
- The suggested device is able to avoid the incidents when driving due to sleepiness. The system works properly even in case of drivers sporting spectacles and even below low light stipulations if the digital camera offers higher output.
- Information about the head and eyes position is obtained through a range of self-developed photograph processing algorithms. During the monitoring, the system is able to figure out if the eyes are opened or closed.
- When the eyes have been closed for too long, a warning sign is issued. processing judges the driver's alertness level on the groundwork of continuous eye closures



CONCLUSION:

- The driver anomaly observing framework created is able of identifying laziness, intoxicated and careless practices of driver in a brief time. The Laziness Detecting Framework created based on eye closure of the driver can separate ordinary eye flicker and tiredness and distinguish the laziness while driving.
- The suggested device is able to avoid the incidents when driving due to sleepiness. The system works properly even in case of drivers sporting spectacles and even below low light stipulations if the digital camera offers higher output.
- Information about the head and eyes position is obtained through a range of self-developed photograph processing algorithms. During the monitoring, the system is able to figure out if the eyes are opened or closed. When the eyes have been closed for too long, a warning sign is issued.
- processing judges the driver's alertness level on the groundwork of continuous eye closures

REFERENCE PAPERS:

REFERENCES

[1] Manu, B. N. 2016. Facial features monitoring for real time drowsiness detection. 2016 12th International Conference on Innovations in Information Technology (IIT).

[2] Alshaqaqi, B., Baquhaizel, A. S., Amine Ouis, M. E., Boumehed, M., Ouamri, A., & Keche, M. 2013. Driver drowsiness detection

system. 2013 28th International Workshop on Systems, Signal Processing and Their Applications (WoSSPA).

[3] Baek, J. W., Han, B.-G., Kim, K.-J., Chung, Y.-S., & Lee, S.-I. 2018. Real-Time Drowsiness Detection Algorithm for Driver State

Monitoring Systems. 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN).

[4] You, F., Li, X., Gong, Y., Wang, H., & Li, H. 2019. A Real-time Driving Drowsiness Detection Algorithm with Individual

Differences Consideration. IEEE Access, vol. 7, pp. 179396-179408.

[5] Dasgupta, A., Rahman, D., & Routray, A. 2018. A Smartphone-Based Drowsiness Detection and Warning System for Automotive

Drivers. IEEE Transactions on Intelligent Transportation Systems, 1–10.

[6] Eraldo, B., Quispe, G., Chavez-Arias, H., Raymundo-Ibanez, C., & Dominguez, F. 2019. Design of a control and monitoring system

to reduce traffic accidents due to drowsiness through image processing. 2019 IEEE 39th Central America and Panama Convention

(CONCAPAN XXXIX).

[7] Lashkov, I., Kashevnik, A., Shilov, N., Parfenov, V., & Shabaev, A. 2019. Driver Dangerous State Detection Based on OpenCV &

Dlib Libraries Using Mobile Video Processing. 2019 IEEE International Conference on Computational Science and Engineering

(CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC).

[8] Zhang, S., & Wang, X. 2013. Human detection and object tracking based on Histograms of Oriented Gradients. 2013 Ninth

International Conference on Natural Computation (ICNC).

[9] Kamran, M., Mannan, M., & Jeong, Y. 2019. Drowsiness, Fatigue and Poor Sleep's Causes and Detection: A Comprehensive Study.

IEEE Access, vol. 7, pp. 167172-167186.

[10] Zhang, W., Cheng, B., & Lin, Y. 2012. Driver drowsiness recognition based on computer vision technology. Tsinghua Science and

Technology, vol. 17, no. 3, pp. 354-362.

