**PROJECT REPORT**

**SAFETY HELMET DETECTION**

**With Deep Learning**

**Submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Electronics and Communication Engineering of School of Engineering, CUSAT.**

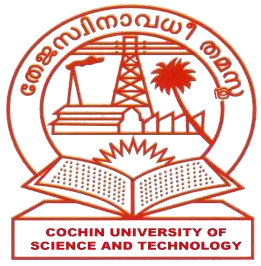
**By**

**ADISH K P 20318008**

**ADARSH K RAMESH 20318007**

**AJITH SADANANDAN 20318011**

**AYISHA NAZNEEM 20318030**

**IRIS JOHN 20318047**

**MAY 2022**

**DIVISION OF ELECTRONICS AND COMMUNICATION** **ENGINEERING**   
**SCHOOL OF ENGINEERING**

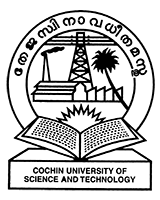
**KOCHI-682022**

DIVISION OF ELECTRONICS ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI-682022



**CERTIFICATE**

*Certified that the Project report entitled* ***“SAFETY HELMET DETECTION with Deep Learning”*** *is a bonafide work of team consisting of Adish K.P, Ajith Sadanandan, Adarsh k Ramesh, Ayisha Nazneeem, and Iris John towards the partial fulfillment for the award of the degree of B.Tech in Electronics and Communication of Cochin University of Science and Technology, Kochi-682022.*

**Project Guide Head of the Division**

Dr. Babita Roslind Jose Dr. Anju Pradeep

# ACKNOWLEDGEMENT

Rome is not built on a day, ie, dreams never turn to reality unless a lot of effort and hard work is put into it. And no effort bears fruit in the absence of support and guidance. It takes a lot of effort to work, the way to achieve this goal, and having someone to guide and help is always a blessing. We would like to take this opportunity to thank a few, who have been instrumental in reaching the finishing point of our project.

For every endeavor in our life, the guiding light of the Lord Almighty has always followed. We have yet again experienced his loving, kindness while doing our research on this project.

In the successful presentation of this project, we express our gratitude to Dr. Anju Pradeep, Head of the Division of Electronics and Engineering. We would like to thank our project guide Ms. Babita Jose for guiding us in our project and providing us with adequate facilities to complete the task we had undertaken.

Our sincere gratitude is expressed and extended to our lab staff and other faculty members of the Electronics and Communication Division for their assistance and unwavering support. We proudly thank all our friends who helped us in various stages of this endeavor. An excellent group of teaching and non-teaching staff helped us with this project. We owe much for the assistance they gave us while doing the project.

# ABSTRACT

The U.S. construction industry suffers from the highest number of fatalities among all industries, i.e., one in ﬁve worker deaths in private industry were in construction. Tremendous loss has occurred to the workers’ families, the industry, and the nation. Considering the large and increasing number of construction projects that are being conducted in the U.S., there is a growing necessity of developing innovative methods to automatically monitor the safety for the workers at construction sites. Based on the collected images, we ﬁrst detect the object of interest (i.e., construction worker) and further analyse whether the worker wears the helmet or not, by using computer vision and machine learning techniques.

Since the head is the most critical area of a human body and is the most vulnerable to an impact that could cause serious injury or death, the use of a protective helmet in construction work is needed. The use of Personal Protective Equipment (PPE) based on ICTs reduces the risk of accidents in the workplace, thanks to the capacity of the equipment to make decisions on the basis of environmental factors. Paradigms such as the Internet of Things (IoT) and Artificial Intelligence (AI) make it possible to generate PPE models feasibly and create devices with more advanced characteristics such as monitoring, sensing the environment and risk detection between others. The working environment is monitored continuously by these models, and they notify the employees and their supervisors of any anomalies and threats. This project presents a smart helmet prototype that monitors the conditions in the workers’ environment and performs a near real-time evaluation of risks.

With this project, we aim to automatically detect the uses of construction helmets (e.g., whether the construction worker wears the helmet or not) by analysing the construction surveillance images. Based on the collected images, we ﬁrst detect the object of interest (i.e., construction worker) and further analyse whether the worker wears the helmet or not, by using computer vision and machine learning techniques.

# CONTENT

[List of Figures iv](#_Toc104804466)

[Abbreviation v](#_Toc104804467)

[CHAPTER 1 INTRODUCTION 1](#_Toc104804468)

[1.1 About the Project 1](#_Toc104804469)

[CHAPTER 2 Object Detection 3](#_Toc104804470)

[2.1 Convolutional Neural Networks 3](#_Toc104804471)

[2.2 Object Detection Algorithms- 5](#_Toc104804472)

[2.2.1 YOLO Algorithm 5](#_Toc104804473)

[2.3 Other Algorithm For Object Detection 8](#_Toc104804474)

[2.3.1 SSD Algorithm 8](#_Toc104804475)

[2.3.2 Results from SSD Implementation 8](#_Toc104804475)

[2.4 SSD vs YOLO: 9](#_Toc104804476)

[CHAPTER 3 Face Recognition 10](#_Toc104804477)

[CHAPTER 4 Packages and Libraries used 13](#_Toc104804478)

[4.1 Open CV 13](#_Toc104804479)

[4.2 TensorFlow 14](#_Toc104804480)

[4.3 Google Colab 15](#_Toc104804481)

[4.4 Dashboard 16](#_Toc104804482)

[CHAPTER 5 WORKING 17](#_Toc104804483)

[CHAPTER 6 RESULTS 19](#_Toc104804484)

[CHAPTER 7 APPLICATIONS & CHALLENGES 20](#_Toc104804485)

[CHAPTER 8 CONCLUSION AND FUTURE ENHANCEMENTS 21](#_Toc104804487)

[REFERENCES 22](#_Toc104804488)

## List of Figures

[Figure 1 Block Diagram of Deep Learning Model 2](file:///C:\Users\aysha\OneDrive\Documents\Helmet%20Detection%20Phase%202.docx#_Toc104805006)

[Figure 2 CNN Model for Object Detection 3](#_Toc104805007)

[Figure 3 Yolo Architecture 5](#_Toc104805008)

[Figure 4 Object Detections of 80 Classes 6](file:///C:\Users\aysha\OneDrive\Documents\Helmet%20Detection%20Phase%202.docx#_Toc104805009)

[Figure 5 Intersection of Unions of bounding boxes 7](file:///C:\Users\aysha\OneDrive\Documents\Helmet%20Detection%20Phase%202.docx#_Toc104805010)

[Figure 6 SSD Architecture 8](#_Toc104805011)

[Figure 7 Annotating on LabelIMG 11](file:///C:\Users\aysha\OneDrive\Documents\Helmet%20Detection%20Phase%202.docx#_Toc104805012)

[Figure 8 Face Recognition working 12](file:///C:\Users\aysha\OneDrive\Documents\Helmet%20Detection%20Phase%202.docx#_Toc104805013)

[Figure 9 Streamlit dashboard 16](#_Toc104805014)

[Figure 10 Flowchart of Working of the Deep Learning Model 17](#_Toc104805015)

## List of Tables

[Table 1 SSD v/s YOLO 9](#_Toc104804971)

## Abbreviation

|  |  |
| --- | --- |
| IoT | Internet of Things |
| HSV | Hue, Saturation, Value |
| CNN | Convolutional Neural Networks |
| DCNN | Deep Convolutional Neural Networks |
| DSDM | Dynamic System Development Model |
| YOLO | You Only Look Once |
| SSD | Single Shot Detector |
| OpenCV | Open Source Computer Vision |
| CHT | Circular Hough Transform |
| HOG | Histogram of Oriented Gradients |

# CHAPTER 1 INTRODUCTION

In safety management at the construction site, it is essential to supervise the safety protective equipment wearing condition of the construction workers. Safety helmets can bear and disperse the hit of falling objects and alleviate the damage of workers falling from heights.

The causes of the construction site fatalities include falls, slips, being struck by objects, electrocution, and being caught in/between objects. And falls to a lower level are leading hazards that have caused construction fatalities, accounting for one-third of work deaths on construction sites. In most of the fall incidents, the workers fall from heights and hit their heads on hard floors. In one study that investigated the number of construction fatalities and the use of safety equipment, the results showed that 47.3% of fatally injured victims either had not used safety equipment (e.g., helmet, guard rails, etc.) or had not used them properly.

Construction workers tend to ignore safety helmets because of weak safety awareness. At the construction site, workers that wear safety helmets improperly are much more likely to be injured. In this project on Computer vision-based object detection, we develop a Machine learning-based method for the real-time detection of safety helmets at the construction site. The machine is trained to detect workers wearing and not wearing helmets. It further translates to image recognition of the person not wearing the helmet and displays the number of violations by an individual.

Based on this the list of violators along with their names are generated which can be viewed by the supervisor in the custom dashboard created using Streamlit. The project provides a reliable and convenient system that allows monitoring and safety of workers at same time. It can be implemented at any workplace with minimal costs and hardware requirements.

1.1 About the Project

In the process of construction, Working personnel must be equipped with a complete set of safety protection tools when carrying out relevant operations. Safety helmet is an important safety tool to effectively prevent the head injury of construction personnel, but only by wearing the safety helmet correctly can the head safety protection function of construction personnel be realized. The project aims to reduce the risk of serious head injury due to the impact of force and collision on the head. The traditional recognition method based on HSV model can recognize the wearing of a safety helmet by matching the color, shape and other features of the helmet. However, this method is too dependent on the template, the actual recognition rate is not stable, and it is easy to be disturbed by the surrounding environment. Artificial intelligence technology is widely used in the field of image process. We aim to automatically detect the uses of construction helmets (e.g., whether the construction worker wears the helmet or not) by analyzing the construction surveillance images. Based on the collected images, we ﬁrst detect the object of interest (i.e., construction worker) and further analyze whether the worker wears the helmet or not, by using computer vision and machine learning techniques. In the ﬁrst step, we incorporate the image with a popular human detection algorithm for construction worker detection; in the second step, the feature extraction techniques is applied to detect helmet uses for the construction working and analysis, which can effectively improve the efficiency and accuracy of image processing. We divide the project into different phases which are object detection, Face detection and finally the Face recognition.   
  
**1.2 BLOCK DIAGRAM:**

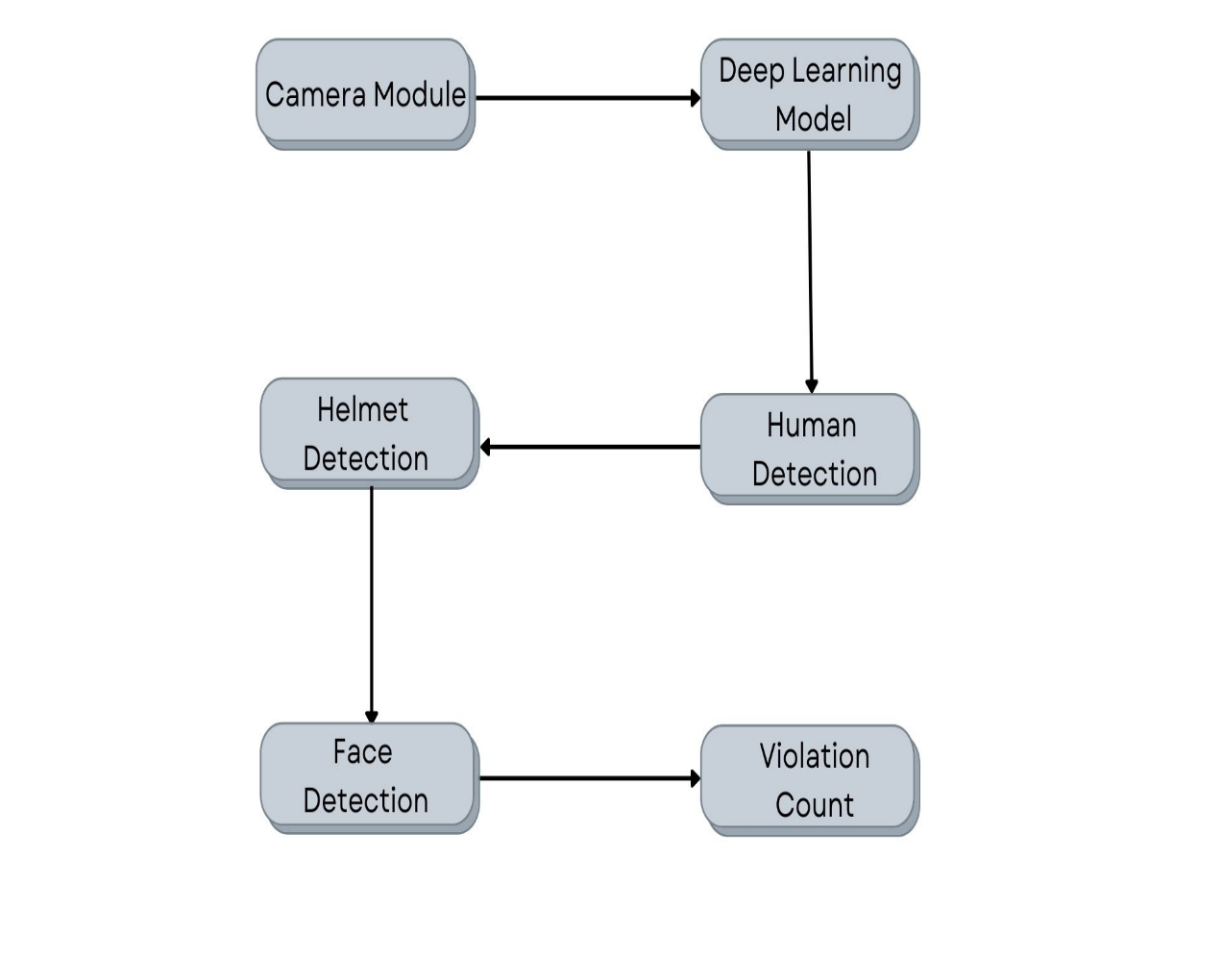


Figure 1 Block Diagram of Deep Learning Model

# CHAPTER 2 Object Detection

Object detection is a computer vision technique for locating instances of objects in images or videos. The fields of helmet detection and face detection comes under object detection as it also relates to a particular class. Object detection is done using deep neural networks and one of the most common one is Convolutional Neural Networks (CNN). CNN’s have been extensively used to classify images. But to detect an object in an image and to draw bounding boxes around them is a tough problem to solve.

## **2.1 Convolutional Neural Networks**

Deep convolutional neural networks (CNN or DCNN) are the type most commonly used to identify patterns in images and video. DCNNs have evolved from traditional artificial neural networks, using a three-dimensional neural pattern inspired by the visual cortex of animals. Deep convolutional neural networks (DCNNs) uses a three-dimensional neural network to process the Red, Green, and Blue elements of the image at the same time. This considerably reduces the number of artificial neurons required to process an image, compared to traditional feed forward neural networks. Deep convolutional neural networks receive images as an input and use them to train a classifier. The network employs a special mathematical operation called a “convolution” instead of matrix multiplication. The architecture of a convolutional network typically consists of four types of layers: convolution, pooling, activation, and fully connected.

Diagram, engineering drawing

Description automatically generated

Figure 2 CNN Model for Object Detection

 It consist of 4 major layers:

**1) Convolutional Layer**: Applies a convolution filter to the image to detect features of the image. Here is how this process works:

·       A convolution: takes a set of weights and multiplies them with inputs from the neural network.

·       Kernels or filters: during the multiplication process, a kernel (applied for 2D arrays of weights) or a filter (applied for 3D structures) passes over an image multiple times. To cover the entire image, the filter is applied from right to left and from top to bottom.

·       Dot or scalar product: a mathematical process performed during the convolution. Each filter multiplies the weights with different input values. The total inputs are summed, providing a unique value for each filter position.

**2) ReLU Activation Layer:** The convolution maps are passed through a nonlinear activation layer, such as Rectified Linear Unit (ReLu), which replaces negative numbers of the filtered images with zeros.

**3) Pooling Layer**: The pooling layers gradually reduce the size of the image, keeping only the most important information. For example, for each group of 4 pixels, the pixel having the maximum value is retained (this is called max pooling), or only the average is retained (average pooling).,Pooling layers help control overfitting by reducing the number of calculations and parameters in the network. After several iterations of convolution and pooling layers (in some deep convolutional neural network architectures this may happen thousands of times), at the end of the network there is a traditional multi layer perceptron or “fully connected” neural network.

**4) Fully Connected Layer:** In many CNN architectures, there are multiple fully connected layers, with activation and pooling layers in between them. Fully connected layers receive an input vector containing the flattened pixels of the image, which have been filtered, corrected and reduced by convolution and pooling layers. The softmax function is applied at the end to the outputs of the fully connected layers, giving the probability of a class the image belongs to – for example, is it a car, a boat or an airplane.

## 2.2 Object Detection Algorithms-

### 2.2.1 YOLO Algorithm

YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in one run of the Algorithm. OLO doesn’t search for interested regions in the input image that could contain an object, instead it splits the image into cells, typically 19x19 grid. Each cell is then responsible for predicting K bounding boxes.

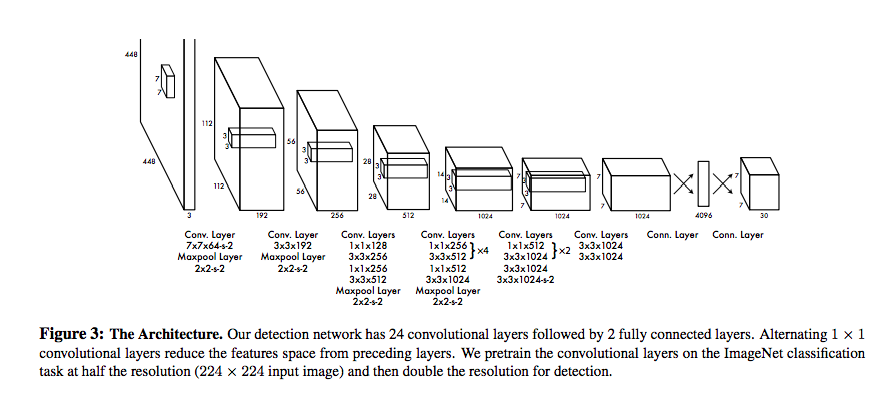
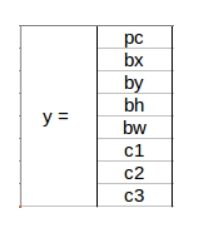


Figure 3 Yolo Architecture

An Object is considered to lie in a specific cell only if the center co-ordinates of the anchor box lie in that cell. Due to this property the center co-ordinates are always calculated relative to the cell whereas the height and width are calculated relative to the whole Image size. During the one pass of forwards propagation, YOLO determines the probability that the cell contains a certain class. The class with the maximum probability is chosen and assigned to that particular grid cell. Similar process happens for all the grid cells present in the image. The algorithm finally outputs the required vector showing the details of the bounding box of the respective class.

To understand the YOLO algorithm, it is necessary to establish what is actually being predicted. Ultimately, we aim to predict a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:



1. Center of a bounding box (bxby)
2. Width (bw)
3. Height (bh)
4. Value cis corresponding to a class of an object (e.g., car, traffic lights, etc.)

In addition, we have to predict the pc value, which is the probability that there is an object in the bounding box.when working with the YOLO algorithm we are not searching for interesting regions in our image that could potentially contain an object. Instead, we are splitting our image into cells, typically using a 19×19 grid. Each cell is responsible for predicting 5 bounding boxes (in case there is more than one object in this cell). Therefore, we arrive at a large number of 1805 bounding boxes for one image..Most of these cells and bounding boxes will not contain an object. Therefore, we predict the value pc, which serves to remove boxes with low object probability and bounding boxes with the highest shared area in a process called non-max suppression.

**Intersection Over Union (IoU)**

When two or more bounding box exist for a single object then YOLO algorithm use IoU to decide the good predicton.

IoU = Area of the intersection / Area of the union,

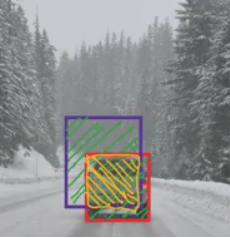
IoU = Area of yellow box / Area of green box

Figure Intersection of Unions of bounding

If IoU is greater than 0.5, we can say that the prediction is good enough. 0.5 is an arbitrary threshold we have taken here, but it can be changed according to your specific problem. Intuitively, the more you increase the threshold, the better the predictions becomes.

**Different Versions of YOLO**

### YOLOv1

The first YOLO version was announced in 2015 by Joseph Redmon, [Santosh Divvala](https://arxiv.org/search/cs?searchtype=author&query=Divvala%2C+S), [Ross Girshick](https://arxiv.org/search/cs?searchtype=author&query=Girshick%2C+R), and [Ali Farhadi](https://arxiv.org/search/cs?searchtype=author&query=Farhadi%2C+A) in the article [“You Only Look Once: Unified, Real-Time Object Detection”](https://arxiv.org/pdf/1506.02640.pdf). Not long after, YOLO dominated the object-detection field and became the most popular algorithm used, because of its speed, accuracy, and learning ability.Instead of treating object detection as a classification problem, the authors thought about it as a regression task concerning spatially separated bounding boxes and associated class probabilities, using a single neural network. The YOLOv1 processed images in real-time at 45 frames per second, while a smaller version – Fast YOLO – reached 155 frames per second and still achieved double the mAP of other real-time detectors.

### YOLOv2

YOLOv2 (sometimes called YOLO9000), was released a year later, in 2016 also by Joseph Redmon and Ali Farhadi in the article [“YOLO9000: Better, Faster, Stronger”](https://arxiv.org/pdf/1612.08242v1.pdf). The name with number 9000 was given, because of the model’s ability to predict even 9000 different objects categories and still run in real-time. The novel model version was not only trained simultaneously on object detection and classification datasets but also gained Darknet-19 as the new baseline model.Since YOLOv2 was also a huge success and became the next state-of-the-art object detection model, more and more engineers began to experiment with this algorithm and create their own, diverse YOLO versions. Some of them will be mentioned throughout the article.

### YOLOv3

The [new version](https://pjreddie.com/darknet/yolo/) of the algorithm was released in 2018 by Joseph Redmon and Ali Farhadi in the article [“YOLOv3: An Incremental Improvement”](https://arxiv.org/pdf/1804.02767.pdf). It was based on the Darknet-53 architecture.In YOLOv3, the softmax activation function was replaced with independent logistic classifiers. During training, the binary cross-entropy loss was used. The Darknet-19 architecture was improved and changed into Darknet-53, with 53 convolutional layers. Besides that, the predictions were made on three different scales that improved YOLOv3’s AP in predicting small objects.

Advantages:

* Speed (45 frames per second — better than realtime)
* Network understands generalized object representation (This allowed them to train the network on real world images and predictions on artwork was still fairly accurate).

faster version (with smaller architecture) — 155 frames per sec but is less accurate.

## YOLOv4

YoloV4 is an important improvement of YoloV3, the implementation of a new architecture in the Backbone and the modifications in the Neck have improved the mAP(mean Average Precision) by 10% and the number of FPS(Frame per Second) by 12%. In addition, it has become easier to train this neural network on a single GPU.

Backbone of YOLOv4 consist of a deep neural network composed mainly of convolution layers. The main objective of the backbone is to extract the essential features, the selection of the backbone is a key step it will improve the performance of object detection. Often pre-trained neural networks are used to train the backbone.

The YoloV4 backbone architecture is composed of three parts:

• Bag of freebies

• Bag of specials

• CSPDarknet53

**Output from the Object Detection using YOLO**

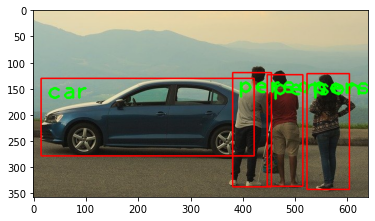
****

Figure 4 Object Detections of 80 Classes

## 2.3 Other Algorithm for Object Detection

### 2.3.1 SSD Algorithm

SSD presents an object detection model using a single deep neural network combining regional proposals and feature extraction. SSD is designed for object detection in real-time. Faster R-CNN, another object detection method, uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. It runs at 7 frames per second, far below what real-time processing needs. SSD speeds up the process by eliminating the need for the region proposal network. To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN’s accuracy using lower resolution images, which further pushes the speed higher.

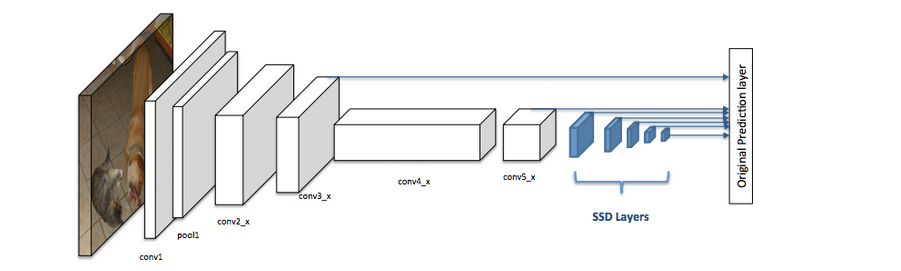


Figure 6 SSD Architecture

SSD divides the image using a grid and has each grid cell be responsible for detecting objects in that region of the image. Detection of objects simply means predicting the class and location of an object within that region. If no object is present, we consider it as the background class and the location is ignored. if there are multiple objects in one grid cell or we need to detect multiple objects of different shapes we use anchor boxes and receptive fields. Each grid cell in SSD can be assigned with multiple anchors/prior boxes. These anchor boxes are predefined and each one is responsible for size and shape within a grid cell. Essentially, the anchor box with the highest degree of overlap with an object is responsible for predicting that object’s class and its location. This property is used for training the network and for predicting the detected objects and their locations once the network has been trained. In practice, each anchor box is specified by an aspect ratio and a zoom level.

The SSD architecture allows pre-defined aspect ratios of the anchor boxes to account for this. The ratios parameter can be used to specify the different aspect ratios of the anchor boxes associated with each grid cell at each zoom/scale level. The zooms parameter is used to specify how much the anchor boxes need to be scaled up or down with respect to each grid cell.

Next is the receptive field. It is defined as the region in the input space that a particular CNN’s feature is looking at. We will use "feature" and "activation" interchangeably here and treat them as the linear combination (sometimes applying an activation function after that to increase non-linearity) of the previous layer at the corresponding location. Because of the convolution operation, features at different layers represent different sizes of the region in the input image.

Feature maps refer to a set of features created by applying the same feature extractor at different locations of the input map in a sliding window fashion. Features in the same feature map have the same receptive field and look for the same pattern but at different locations. This creates the spatial invariance of ConvNet.

So it happens in two steps-feature map extractions and application of convolutional filters to detect objects. The extra step taken by SSD is that it applies more convolutional layers to the backbone feature map and has each of these convolution layers output object detection results. As earlier layers bearing smaller receptive fields can represent smaller-sized objects, predictions from earlier layers help in dealing with smaller-sized objects. Because of this, SSD allows us to define a hierarchy of grid cells at different layers. For example, we could use a 4x4 grid to find smaller objects, a 2x2 grid to find mid-sized objects, and a 1x1 grid to find objects that cover the entire image.

**2.3.2 Results from implementation of SSD**

We initially used the SSD algorithm to detect 80 classes of objects using a pre-trained dataset. The objects in the frame was detected and later on using the dataset for helmet, we have identified people wearing helmet from images and videos.

**a) Detection of objects from images**

Input image:

****

SSD Output:

****

**b) Detection of helmets from image**

Input Image:

SSD Output:

****

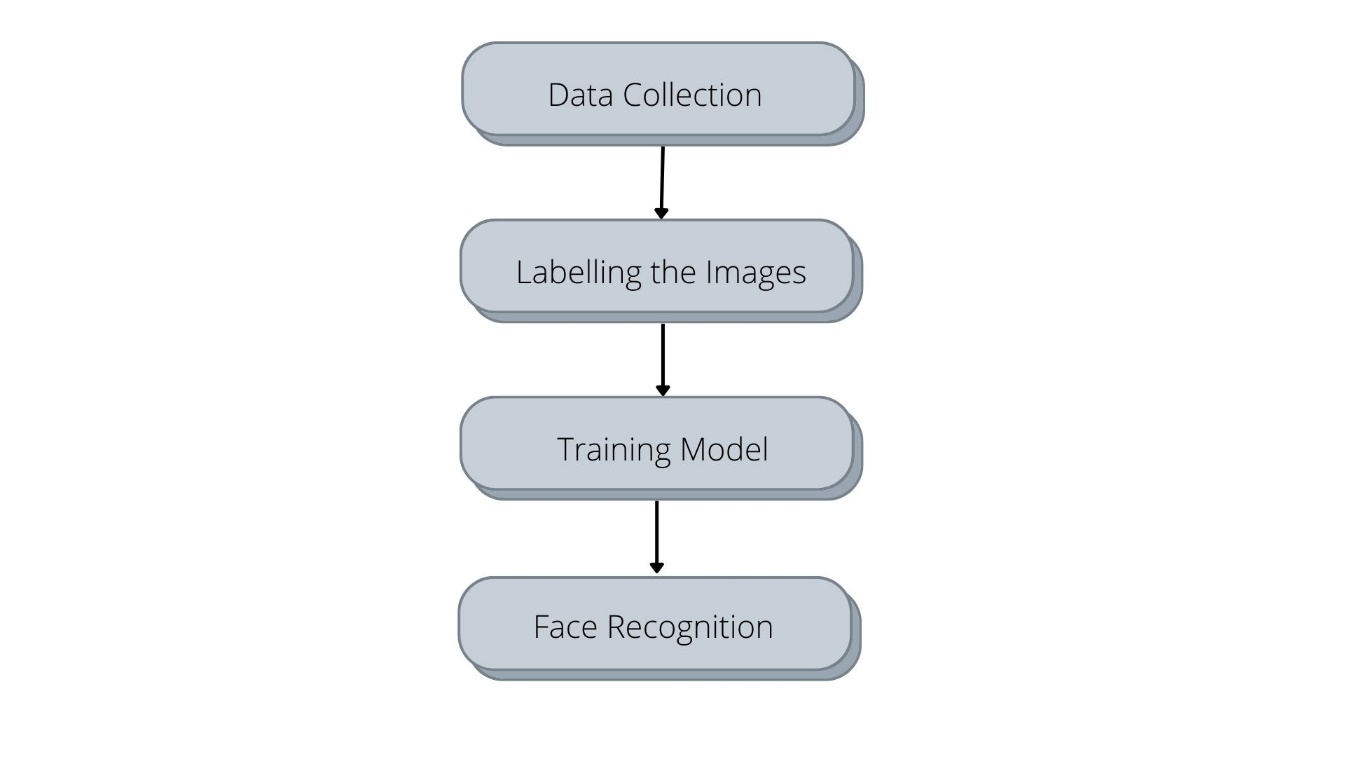
## 2.4 SSD vs YOLO

Table 1 SSD v/s YOLO

|  |  |
| --- | --- |
| **SSD** | **YOLO** |
| Single Shot Detector | You Only Look Once |
| Runs a convolutional network on input images at just one time and computes a feature map. | The open-source technique of object detection which will acknowledge objects in pictures and videos fleetly |
| SSD could be a higher choice as we have a tendency to square measure able to run it on a video and therefore the truth trade-off is extremely modest. | YOLO is a better option when  exactness is not too much of disquiet but you want to go super quick |
| When the object size is tiny, the performance dips a touch. | YOLO could be a higher choice even when the object size is small. |
| Runs a convolutional network on input image just one time and computes a feature map | Can be enforced for applications as well as artificial intelligence, self-driving cars, and cancer recognition approaches. |

# CHAPTER 3 Face Recognition

Face-recognition algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.



**a) Data Collection –**

We collected 200 images of each individual in the group with helmet and without helmet.

**b) Labelling the images –**

The data(images) has to be labelled with the name tags of the individuals before loading to the model. In this project we use ‘Labelimg’ for labelling the images in YOLO.

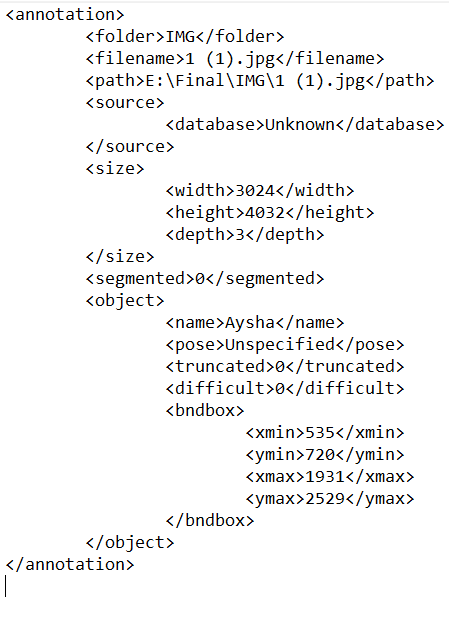
**LabelImg :** It is a graphical image annotation tool, which is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format. It also supports YOLO and CreateML formats. Labels are used to help identify components in a data which you want to train your model to identify in datasets that are not labeled. High quality datasets are essential for computer vision and building a highly performant model. Computer vision models follows the garbage in, garbage out philosophy which means labeling images carefully and accurately is important. We've created a guide to modelling to help make sure your training dataset is high quality.

Graphical user interface, application

Description automatically generated

Figure 7 Annotating on LabelIMG

**c) Training Model –**

After collecting these images, the images are annotated and added the feature extraction. After detecting faces in an image, we crop the faces and feed them to a Feature Extraction Algorithm, which creates face embedding- a multi-dimensional (mostly 128 or 512 dimensional) vector representing features of the face.

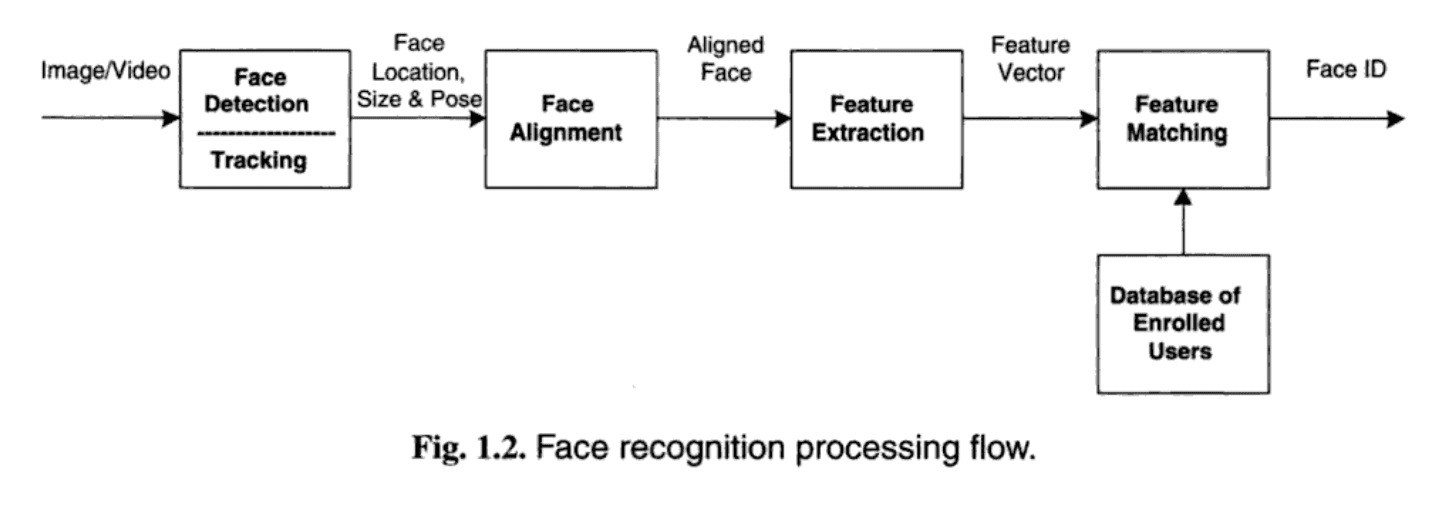
**d) Face Recognition –**

Figure 8 Face Recognition working

To train a functional face detection algorithm the following steps are required:

**Step 1:** The possible human eye regions are detected by testing all the valley regions in the gray-level image.

**Step 2:** The genetic algorithm is used to generate all the possible face regions which include the eyebrows, the iris, the nostril and the mouth corners.

**Step 3:** Each possible face candidate is normalized to reduce both the lighting effect, which is caused by uneven illumination and the shirring effect, which is due to head movement. The fitness value of each candidate is measured based on its projection on the eigen-faces.

**Step 4:** After a number of iterations, all the face candidates with a high fitness value are selected for further verification. At this stage, the face symmetry is measured, and the existence of the different facial features is verified for each face candidate.

# CHAPTER 4 Packages and Libraries used

## 4.1 Open CV

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.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database and etc. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

**4.1.1 Applications of OpenCV:** There are lots of applications that are solved using OpenCV, some of them are listed below

* face recognition
* Automated inspection and surveillance
* number of people – count (foot traffic in a mall, etc)
* Vehicle counting on highways along with their speeds
* Interactive art installations
* Anamoly (defect) detection in the manufacturing process (the odd defective products)
* Street view image stitching
* Video/image search and retrieval
* Robot and driver-less car navigation and control
* object recognition

**4.1.2 OpenCV Functionality**

* Image/video I/O, processing, display (core, imgproc, highgui)
* Object/feature detection (objdetect, features2d, nonfree)
* Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
* Computational photography (photo, video, superres)
* Machine learning & clustering (ml, flann)
* CUDA acceleration (gpu)

## 4.2 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

TensorFlow is a framework composed of two core building blocks — a library for defining computational graphs and a runtime for executing such graphs on a variety of different hardware. A computational graph has many advantages but more on that in just a moment. Building the computational graph.A computational graph is nothing but a series of TensorFlow operations arranged into a graph of nodes.

Running the computational graph.To actually evaluate the nodes, we must run the computational graph within a session. A session encapsulates the control and state of the TensorFlow runtime.

**4.2.1 Advantages of TensorFlow**

1.Open-source platform

It is an open-source platform that makes it available to all the users around and ready for the development of any system on it.

2. Data visualization

TensorFlow provides a better way of visualizing data with its graphical approach. It also allows easy debugging of nodes with the help of TensorBoard. This reduces the effort of visiting the whole code and effectively resolves the neural network.

3. Keras friendly

TensorFlow has compatibility with Keras, which allows its users to code some high-level functionality sections in it. Keras provides system-specific functionality to TensorFlow, such as pipelining, estimators, and eager execution.

1. Scalable

Almost every operation can be performed using this platform. With its characteristic of being deployed on every machine and graphical representation of a model allows its users to develop any kind of system using TensorFlow.

1. Compatible

It is compatible with many languages such as C++, JavaScript, Python, C#, Ruby, and Swift. This allows a user to work in an environment they are comfortable in.

## 4.3 Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.

Colab offers following functionalities:

* Write and execute code in Python
* Document your code that supports mathematical equations
* Create/Upload/Share notebooks
* Import/Save notebooks from/to Google Drive
* Import/Publish notebooks from GitHub
* Import external datasets e.g. from Kaggle
* Integrate PyTorch, TensorFlow, Keras, OpenCV
* Free Cloud service with free GPU

## 4.4 Dashboard

A real-time live dashboard is a web app used to display Key Performance Indicators (KPIs).

The data containing the violator’s details are sent over to be displayed in a dashboard created using Streamlit. The Dashboard provides a quick and simple visualization of data.

**Streamlit:** Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. It is a very easy library to create a perfect dashboard by spending a little amount of time. It also comes with the inbuilt webserver and lets you deploy in the docker container. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows to write an app the same way we write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

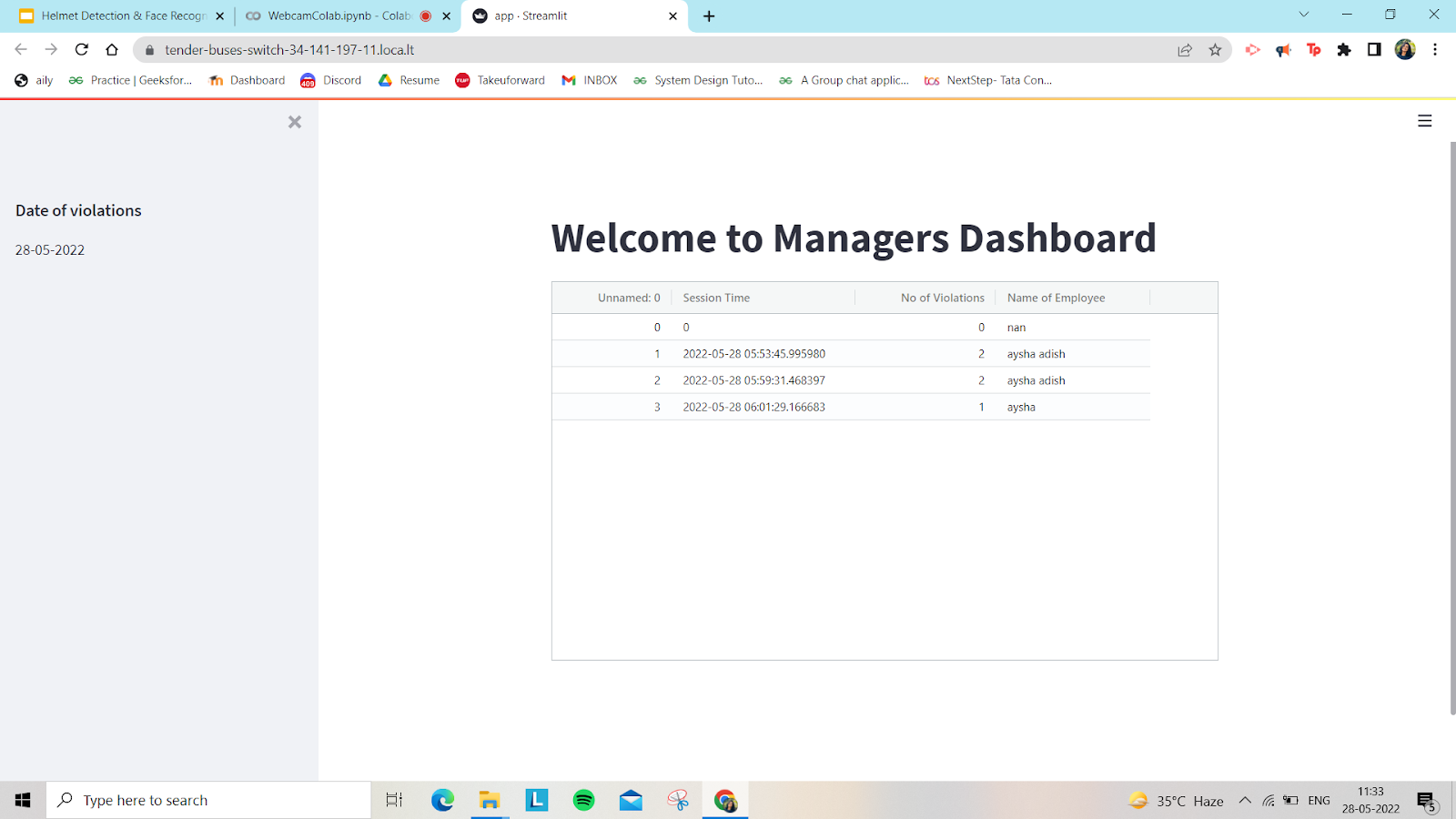


Figure 9 Streamlit dashboard

# CHAPTER 5 WORKING

Our aim is to build a system that detects persons not wearing helmets in a civil work site. The working can be divided into two stages.

1. Helmet Detection

2. Face Detection

In the first stage, we have to detect the persons and the helmet from the video captures. For this, the video input from the camera module is fed into the deep learning model . The model is a pertained machine learning model built using the YOLO Object detection algorithm and other python packages like TensorFlow, opencv, etc.

We use computer vision for deep learning to analyze the [different types of data sets](https://www.cogitotech.com/blog/what-are-the-various-types-of-data-sets-used-in-machine-learning/) through annotated images showing the object of interest in an image. It can recognize the patterns to understand the visual data feeding thousands or millions of images that have been labeled for supervised machine learning algorithms training. So, the model efficiently detects persons, helmets, and faces. Once the input is loaded, the model identifies the persons with and without helmets.

Bounding boxes appear on the head, showing labels: ‘Person with Helmet’ and ‘Person without Helmet ‘A count of number of people wearing the helmet is also maintained by the system. Once the persons are detected the same data is fed into the face detection model which is the second stage of the project.

In the second stage, face identification of the persons without helmets is being implemented using another deep learning model which is also a pre-trained one using 400 of our own images. This face detection comes useful in finding out the number of violations done by the same person. The data from the model is received in a CSV file which is to be uploaded to the dashboard and the data is displayed in a tabulated way including details of count, name of violators and time stamp.

**Code Flow**

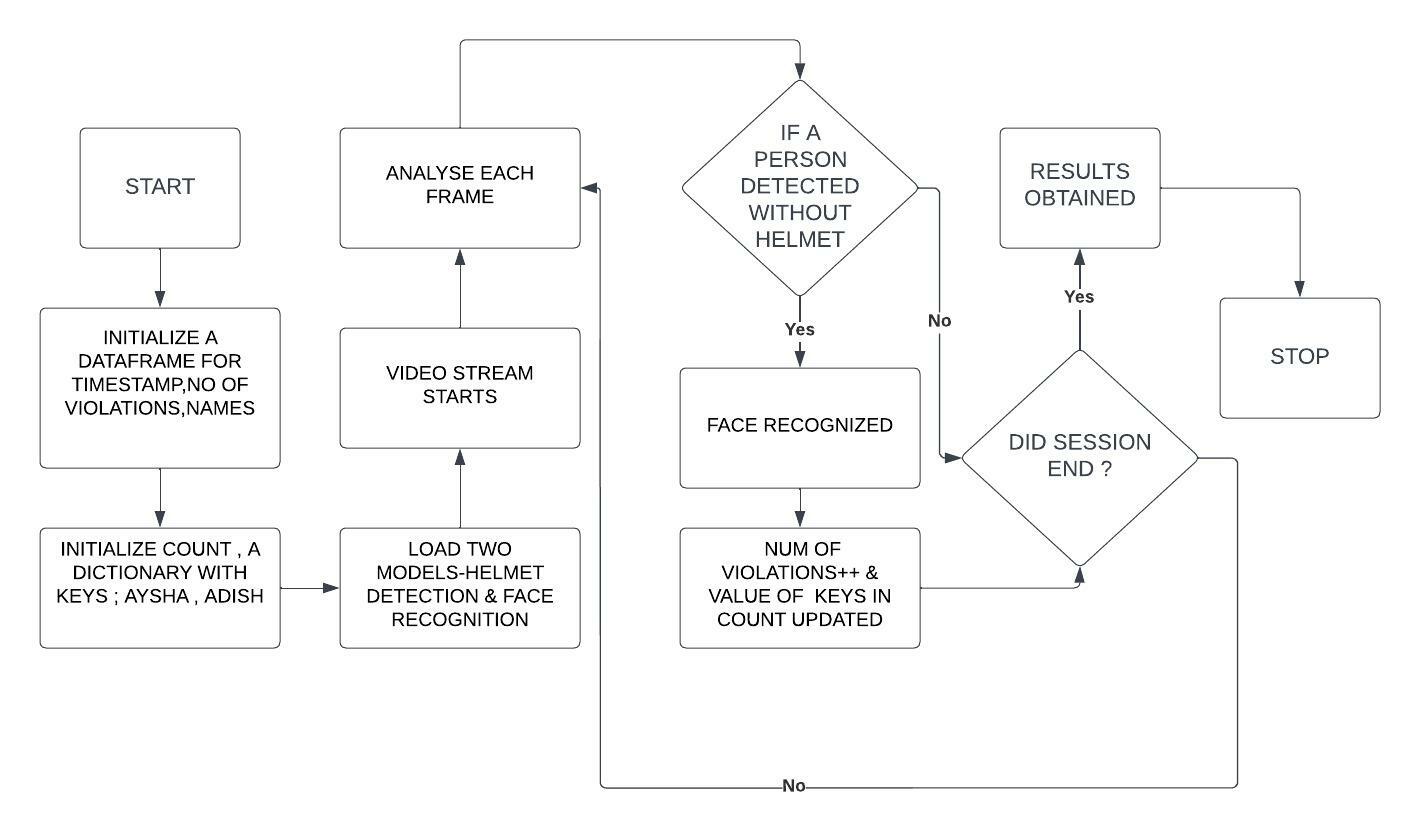


Figure 10 Flowchart of Working of the Deep Learning Model

The machine learning code along with the implementation code is written in python and run in Google colab. Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.Pandas and Numpy are two important python packages that we use to create data structures to store the recorded values.Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. First we initialize a data frame named df with columns - 'Time stamp', 'No of Violations', 'Names'. Time stamp stores the duration which violation was recorded. No of violations store the total number of violations occurred during the recording and names stores the registered workers name whose violations where recorded. We also require a dictionary called 'count' where in we have two keys: Aysha, Adish. These are the two classes we will identify through face recognition. Values of these keys are : 'with helmet' and 'without helmet'. The same will be updated whenever a class is found violating or keeping the rule.

Accessing our webcam within a Google Colab notebook cannot be done using direct python code. When you use a Google Colab Runtime you are connecting to a cloud VM hosted by Google. In order to utilize your local machine's webcam within the VM we can utilize JavaScript code.For this purpose we also make use of another library called opencv.OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations. Cv2. videocapture() starts video streaming. Each frame is analysed seperately. We will be able to identify two classes:Aysha, Adish over which bounding boxes will appear indicating whether they have worn helmets or not. Once a face is recognizes without any helmet on head, the same is updates in the dictionary called count. Ex: if Aysha is recognized with helmet and Adish is recognized without helmet then the updated dictionary becomes:

Count= { Aysha: with helmet, Adish:without helmet}

Let's consider this example for further explanation.Here Aysha and Adish are the two keys are their corresponding values are 'with helmet' and 'without helmet'. Frames are continously analysed untill the duration ends. Usually in a worksite the duration can be from 6-8 hours. For the demo purpose and considering limitations of google colab we have set the time for 2 minutes. It is to be noted that each persons last frame will be considered as the final result. That is even if a person wears helmet for some time and then takes it off for rest of them day, then whatever appears in the last frame will be evaluated. That is 'without helmet ' in this case. Once this count dictionary is finalized we evaluate the and update the dataframe. By iterating through each value of the dictionary. The time stamp which was collected during the video streaming is stored first and then we calculate number of violations. Here number of violations is 1 and correspondingly 'Adish ' will be added in the names column. Once the data frame is ready this will be converted to a csv file. To publish the results to the manager we have used streamlit.Streamlit is an open-source app framework for Machine Learning and Data Science teams.It creates beautiful web apps in minutes.The file is sent through streamlit functions and the streamlit web app is invoked directly from Google Colab

# CHAPTER 6 RESULTS

Result 4 Violations count

Result 3 With Helmet detection

Result 2 Without Helmet detection

Result 1 Image Recognition Output

# CHAPTER 7 APPLICATIONS

**Construction Site-** The proposed system can be installed in construction sites and hazardous workplaces where safety equipment like helmets is required. This can reduce accidents that happen due to the negligence of workers and supervisors. With detailed reports of violations, the management can take necessary steps including education of violators regarding the health hazards associated with not wearing helmets.

**Motorbike Helmet detection -** The same system can be used to ensure that motorbike users wear helmets while riding. With a slight modification, the system can provide details of the vehicle along with a picture of the rider so that fines and warnings can be provided to the user. This can ensure that people are following the law and thereby reduce fatalities associated with road accidents.

# CHALLENGES

* + 1. Lack of data - we require almost 10000 images, Currently we have trained the model with 400 images
    2. Hardware limitations of google colab - the limitations are in terms of ram, gpu ram and hbm, dependent on google colab hardware, at the moment is respectively ≈25gb, ≈12gb and ≈64gb. This will limit the dataset you can load in memory and the batch size in your training process
    3. Limited space & time - the google colab platform stores files in google drive with a free space of 15gb

# CHAPTER 8 CONCLUSION AND FUTURE ENHANCEMENTS

In this project, a novel approach is proposed for automatic detection of helmet uses for construction safety using computer vision and machine learning techniques. The proposed system has two major parts: one part incorporates frequency domain information of the image with a popular human detection algorithm HOG for human (i.e., construction worker) detection; the other part works for helmet use detection combining color information and Circle Hough Transform (CHT). Currently, our system can detect helmets composed of some particular colors, such as yellow, blue, red, and white. As an

extension of this work, we aim to make the system scalable to detect helmets with other colors. In future, the system will be made well capable of differentiating between normal cap and helmet, as the proposed system shows low performance in this case. Also, we aim to apply some deep learning techniques for

improving the overall accuracy of the system. Also, applying upper body searching algorithm instead of detecting whole human as object of interest can improve the helmet detection accuracy

In this project, a novel approach is proposed for automatic detection of helmet uses for construction safety using machine learning techniques. The experiment results demonstrate that the method can be used to detect the safety helmets worn by the construction workers at the construction site. The model uses the SSD-Mobile Net algorithm to detect safety helmets. The presented method offers an alternative solution to detect the safety helmets and improve the safety management of the construction workers at the construction site.

As a project phase-I we were able to identify persons and 79 other classes of objects as output from the Object Detection Model. Also successfully ran and tested dashboard consisting of homepage and result page. Further

# REFERENCES

1. [Deep Learning-Based Safety Helmet Detection in Engineering Management Based on Convolutional Neural Networks](https://www.hindawi.com/journals/ace/2020/9703560/)
2. [An improved helmet detection method for YOLOv3 on an unbalanced dataset](https://arxiv.org/pdf/2011.04214.pdf)
3. [Train your own custom model for Helmet Detection(object detection) using YOLO.](https://medium.com/@vijaysingh_60587/train-your-own-custom-model-for-helmet-detection-object-detection-using-yolo-f53a48066d7a)
4. [SAFETY HELMET DETECTION IN INDUSTRIAL ENVIRONMENT USING DEEP LEARNING](http://aircconline.com/csit/papers/vol10/csit100518.pdf)

`