**1. INTRODUCTION:**

Traffic & Accident management in India is a complex activity given the number of riders & automobiles we have. Using the existing video feeds that come through the cameras positioned on highways, traffic signal & busy roads, detect the following incidents/events 1) Bike riders who are riding without helmets 2) Count of vehicles that are passing through a signal/road at any given point in time. It should allow report generation on various parameters. 3) Ability to read vehicle no & track & search.

We trained a Deep Learning model to recognize helmet by creating our own dataset using YOLO V3 network. The dataset contained images obtained from internet and images taken from various traffic cctv footage. To detect motorcyclists without helmet we run this trained classifier on that particular frame and if it return false then that person is not wearing helmet . To extract motorcyclist from a frame we used pre trained YOLO V3 network trained on COCO dataset which recognizes motorcyclist and persons along with other vehicles and many other objects. We used intersection over union concept to get the motorcyclist and person sitting on it in a single image.

Once we recognize that the person is not wearing helmet we pass that image to OPEN ALPR to recognize the number plate. The number plate hence recognized is stored in database for further investigation .

We store the number plate data of every vehicle passing through a signal by first extracting the vehicle through YOLO V3 model then passing the extracted image to OPEN ALPR for number plate detection . We stored this data in MYSQL database.

For counting the number of vehicles passing through a particular signal so that this data could be used for dynamic signaling we first recognized all the vehicles using the pre trained YOLO V3 model and then developed an algorithm which can recognize if two vehicles in different frame are same or not.

1. **PROJECT MODEL:**

**2.1 DATASET:**

For the first step of detecting persons and bikes we used a pre trained YOLO model which was trained on COCO data set .COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has over 330k images and 1.5 million object instances and 80 object categories. We modified to detect only 2 objects person and bike. For helmet detection we used around 10k images 2000 of which were available online and the rest of the images we annotated ourselves by using MICROSOFT VOTT annotator. The training set had 8k images and test consisted of 2k images. Video dataset was collected from youtube and RCOEM college cctv footage.

**2.2 BIKE DETECTION:**

We used a pre-trained YOLOV3 model which could detect a motorcycle and person separately. It could also create bounding boxes around them. We used the concept of iou(intersection over union)to calculate overlapping between different boxes of persons and bikes. If the overlap value is above a threshold(0.3), we consider that the person is sitting on that particular bike. Using the bounding box coordinate values of the bike and person, we calculate bounding box coordinates for combined picture of motorcycle and persons sitting on it. We passed these cropped images to our next model which detects helmet.

**2.3 HELMET DETECTION:**

For helmet detection we trained Full YOLO model from scratch with our dataset for 50 epochs and its accuracy was found to be 83% .It has 22 layers with 20 of 2D convolution layers and 2 layers of Resnet. The model has batch size 2 and learning rate as 1e-4.We used NVIDIA QUAD K1200 - 4GB GPU and Intel Zeon E51607 3.1GHZ CPU on Windows 7 pro with 16 gb RAM for training our model. We used KERAS library to build our CNN model. It detects the helmet and marks the anchor box around it.

**EXAMPLE:**

**A] RIDER WITH HELMET:**



Image of Rider wearing helmet

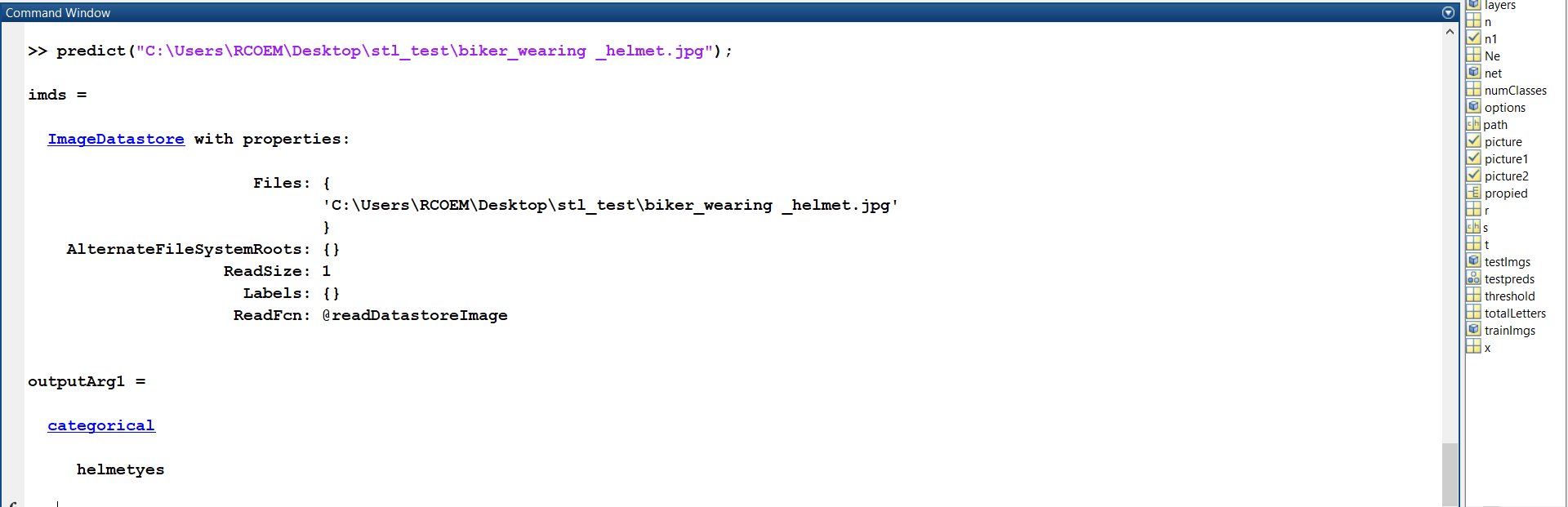


Image of Output where above image is input[o/p : helmetyes]

**B] RIDER WITHOUT HELMET:**



Image of Rider not wearing helmet

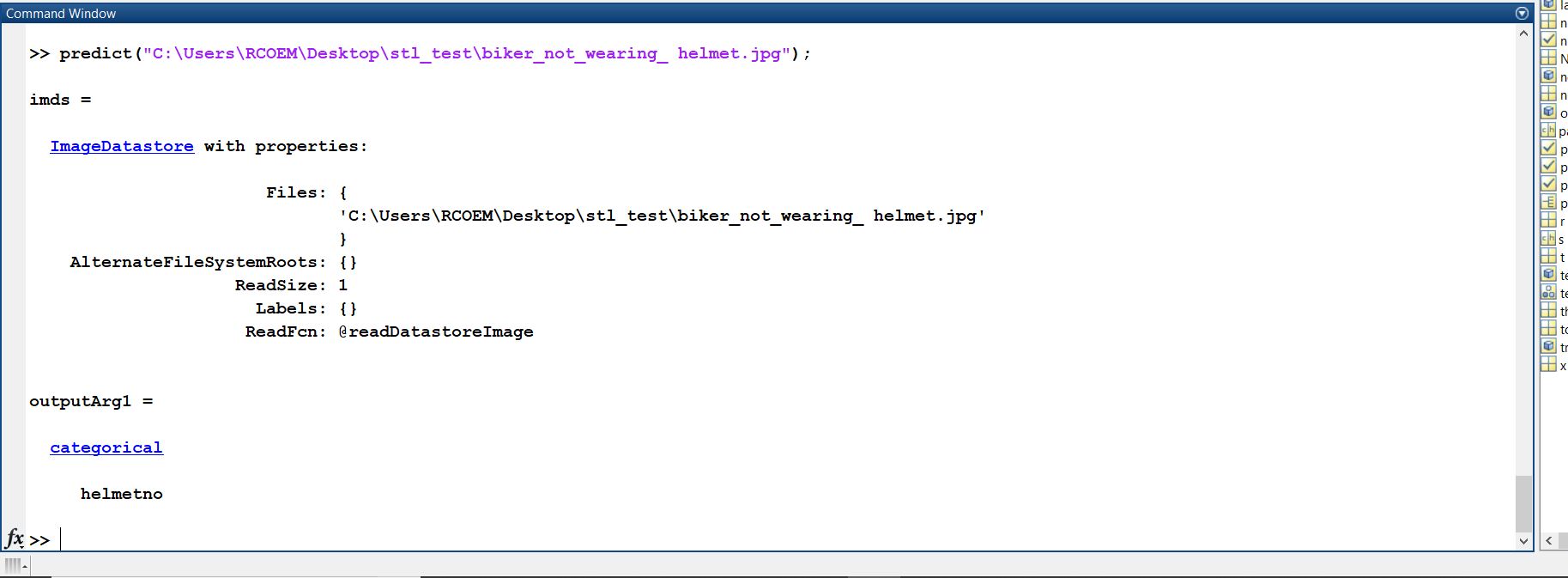


Image of Output where above image is input[o/p : helmetno]

**2.4 COUNTING NUMBER OF VEHICLES:**

The concept is to count only the vehicles which are moving and contribute to the traffic density of that area. The problem here is that when we apply object detection on a video it first converts the video into frames with the rate of 30 frames per second. There are rarely

any changes in many consecutive frames but our model detects the same object repeatedly hence giving us wrong interpretation of density at that moment. Therefore we designed our own algorithm which first detects the object in the frames and stores coordinates of its anchor box into database and before adding it to the total count of vehicles it first compares the coordinates of current object with the coordinates of all objects that are already counted. If they are found to be comparable the algorithm discards it and continues with next object that is detected otherwise the count is increased and the coordinates are stored in the list.

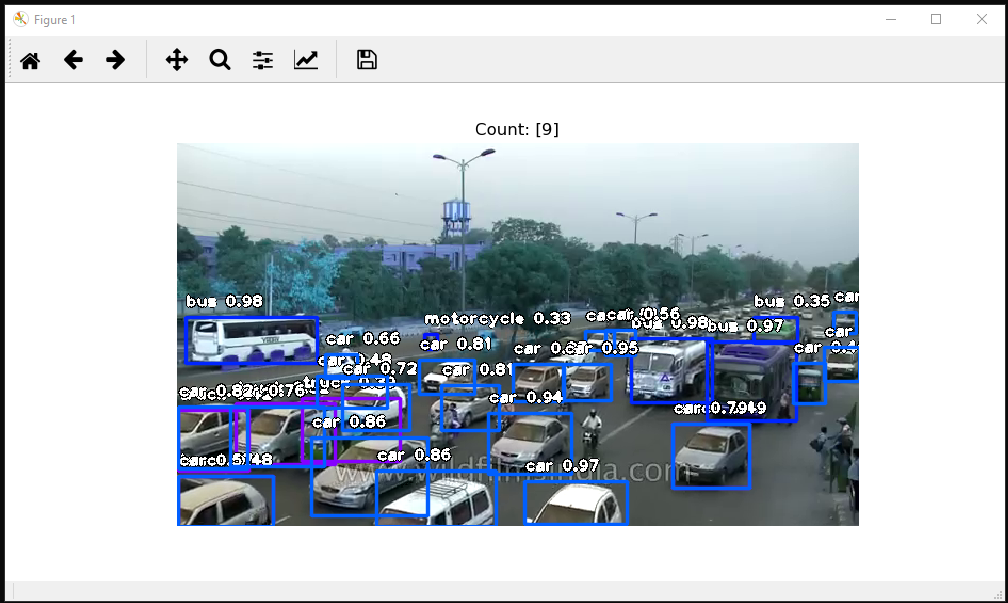


Image of System Counting No. Of Vehicles Passing at a time

**2.5 DETECTING NUMBER PLATE:**

For detection of number plates, we have used OpenALPR. OpenALPR stands for Automatic License Plate Recognition.

**2.5.1 About OpenALPR Cloud API:**

The OpenALPR Cloud API is a web-based service that analyzes images for license plates as well as vehicle information such as make, model, and color. The Cloud API service is easy to integrate into your application via a web-based REST service. When an image data is sent to the OpenALPR API, it processes that data and return JSON data describing the license plate and vehicle.

**EXAMPLE:**



Input Image for Number Plate detection

C:\Users\yash\Downloads\NUMPL.PNG

Image of Output for the above input image[o/p : TN70T4132]

**2.6 GUI DESIGING:**

Python provides various options for developing graphical user interfaces (GUIs). **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python Python when combined with Tkinter provides a fast and easyway to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task.

**2.6.1 NEED OF GUI:**

(a) To Display the traffic status at particular time i.e numbervehicleat given time

(b) To Display vehicle details for entered vehiclenumber

Libraries to be imported: Tkinter

top = Tkinter.Tk()top.mainloop()

The above code will create a window.

root = Tk()

text = Text(root)

This code will create a text view where user will enter the number plate or time

def callback(): Fetch(text).

b = Button(master, text="fetch", command=callback)

This code will create button having id master, text fetch and pressing button it will move the pointer to function name call back Function callback the data from database

**2.7 DATABASE:**

Database tables are used here to store the information of vehicles and traffic density with respect to location, date and time. These stored data is used to fine the traffic rule’s violator and to manage the traffic in the city. **MySQL Workbench** is used for database management.

**2.7.1 Database table for Track And Search and Fine Management:**

Information of every vehicle recognized by a particular camera is stored in the database table, with the date, time and location of recognition. Here the location of recognition is according to the camera which recognizes the vehicle. The information of vehicle here means its registered vehicle number, its image captured at the time of recognition and the information that either any traffic rule was violated by that particular vehicle, and if violated what rule was violated by it. So the columns in this particular database table are: VEH\_NUM, CAM\_NUM, IMG\_NAME, DATE\_COL, TIME\_COL and VIOLATIONS. Every vehicle image is stored at a particular static location and using the entry in IMG\_NAME column the image can be retrieved easily. Using data from this table any vehicle can be tracked or searched, i.e. it can be found that at what date and time does the particular vehicle passed from particular location or at which location does the particular vehicle was recognized recently. Data from this table is also used to generate fine receipt (**e-chalan**) according to the violations committed by the particular vehicle.

**2.7.2 Database table for traffic management:**

Count of vehicles passing from a particular location every 5 minutes is stored with date and time in this particular database table. Similar data can be stored for different locations in the city, according to the camera (camera number) that is used to take footage (using which the count is calculated) of the traffic. Now using this data the density of vehicles, at different locations is found and traffic is managed accordingly. Here columns used were: CAM\_NUM, DATE\_COL, TIME\_COL and COUNT.

**2.7.3 MySQL Workbench Installation:**

(a) Installed **MySQL Installer 8.0.15**

(b) Using MySQL Installer, MySQL Workbench is installed.

**2.7.4 Connecting to Database, storing and retrieving from it using python:**

(a) User is created in MySQL Workbench while installing it.

(b) mysql.connector is imported.

(c) mysql.connector.connect() function connects to the user and returns MySQL Connection object.

(d) cursor() function returns an cursor object using which execute() function is executed to execute any sql queries.

(e) We defined different functions in a separate python file to: create database, create table, insert in table and retrieve data from table.

(f) fetchone() function it returns the object with data from the 1st row from the selected rows.

**3. TECHNOLOGY/TOOLS USED:**

**3.1 TENSOR FLOW:**

Tensor Flow (TF) [2] is an open source software library for machine learning written in Python and C++. The main reason behind it is that TF was developed by Google Brain Team. Google has already been using TF to improve some tasks on several products. These tasks include speech recognition in Google Now, search features in Google Photos, and the smart reply feature in Inbox by Gmail. Some design decision in TF have lead to this framework to be early adopted by a big community. One of them is the ease of going from prototype to production. There is no need to compile or to modify the code to use it on a product. Then, the framework is not only thought as a research tool, but as a production one. Another main design aspect is that there is no need to use dierent API when working on CPU or GPU. Moreover, the computations can be deployed over desktops, servers and mobile devices. A key component of the library is the data flow graph. The sense of expressing mathematical computations with nodes and edges is a TF trademark. Nodes are usually the mathematical operations, while edges define the input / output association between nodes. The information travels around the graph as a tensor, a multidimensional array. Finally, the nodes are allocated on devices where they are executed asynchronously or in parallel when all the resources are ready

.

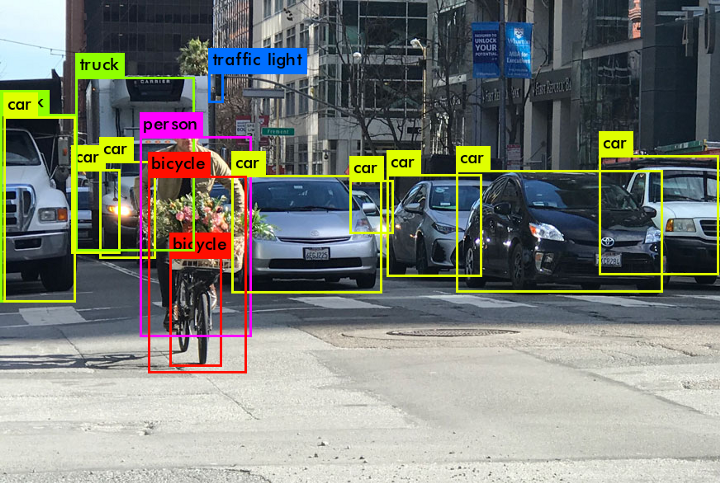
**3.2 KERAS LIBRARY:**

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

**4. ABOUT YOLO MODEL:**

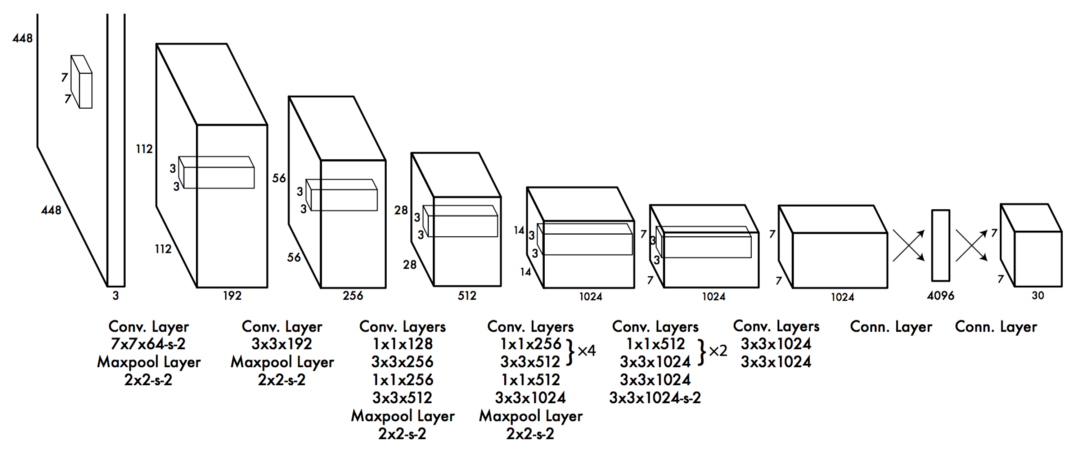


A demonstration from the YOLOv2.



The objects detected by YOLO:

**NETWORK DESIGN**



YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolution layers use 1 × 1 reduction layers alternatively to reduce the depth of the features maps. For the last convolution layer, it outputs a tensor with shape (7, 7, 1024). The tensor is then flattened. Using 2 fully connected layers as a form of linear regression, it outputs 7×7×30 parameters and then reshapes to (7, 7, 30), i.e. 2 boundary box predictions per location.

A faster but less accurate version of YOLO, called Fast YOLO, uses only 9 convolutional layers with shallower feature maps.

### **Loss function**

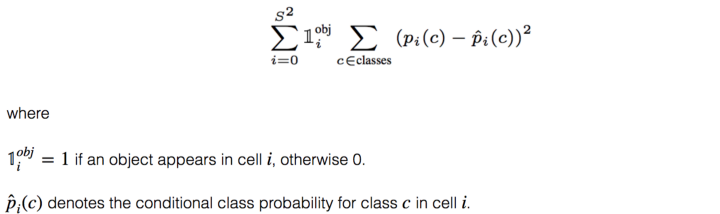
YOLO predicts multiple bounding boxes per grid cell. To compute the loss for the true positive, we only want one of them to be **responsible** for the object. For this purpose, we select the one with the highest IoU (intersection over union) with the ground truth. This strategy leads to specialization among the bounding box predictions. Each prediction gets better at predicting certain sizes and aspect ratios.

YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of:

* the **classification loss**.
* the **localization loss** (errors between the predicted boundary box and the ground truth).
* the **confidence loss** (the objectness of the box).

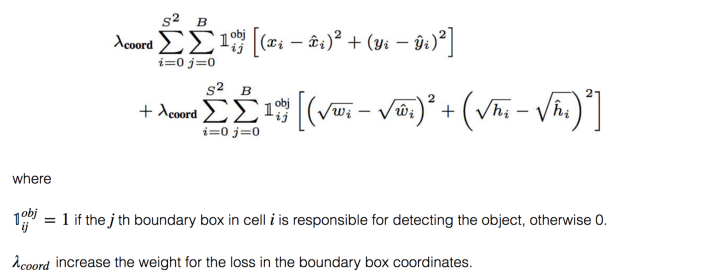
**Classification loss**

If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class:



**Localization loss:**

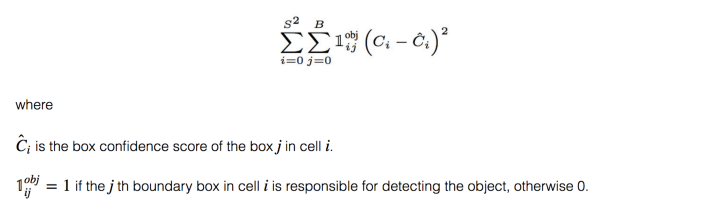
The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object.



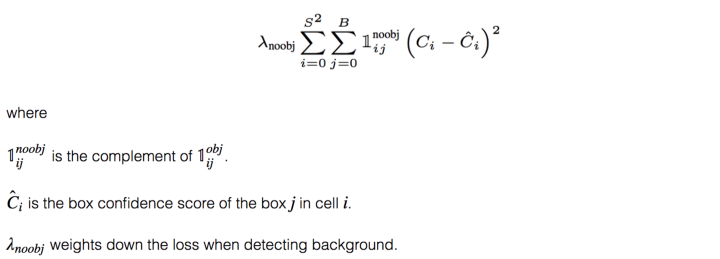
We do not want to weight absolute errors in large boxes and small boxes equally. i.e. a 2-pixel error in a large box is the same for a small box. To partially address this, YOLO predicts the square root of the bounding box width and height instead of the width and height. In addition, to put more emphasis on the boundary box accuracy, we multiply the loss by λcoord(default: 5).

**Confidence loss:**

If an object is detected in the box, the confidence loss (measuring the objectness of the box) is:

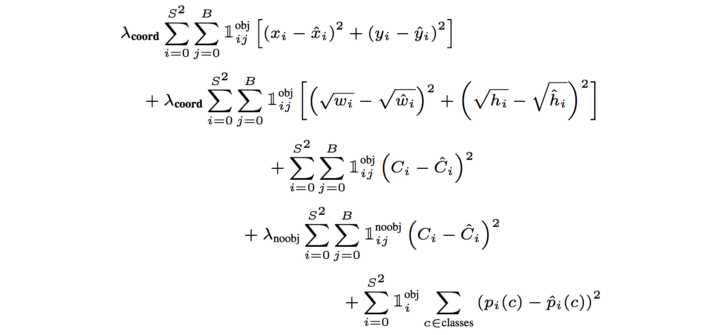


If an object is not detected in the box, the confidence loss is:



Most boxes do not contain any objects. This causes a class imbalance problem, i.e. we train the model to detect background more frequently than detecting objects. To remedy this, we weight this loss down by a factor λnoobj (default: 0.5).

**Loss:** The final loss adds localization, confidence and classification losses together.



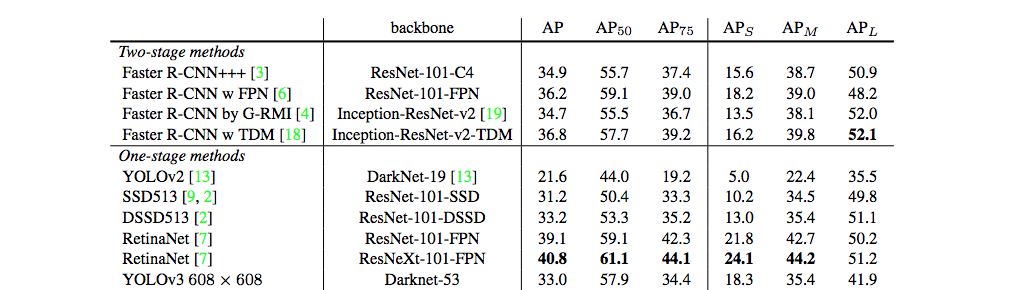
### Training

YOLO is trained with the ImageNet 1000 class classification dataset in 160 epochs: using stochastic gradient descent with a starting learning rate of 0.1, polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9. In the initial training, YOLO uses 224 × 224 images, and then retune it with 448× 448 images for 10 epochs at a 10−3 learning rate. After the training, the classifier achieves a top-1 accuracy of 76.5% and a top-5 accuracy of 93.3%.

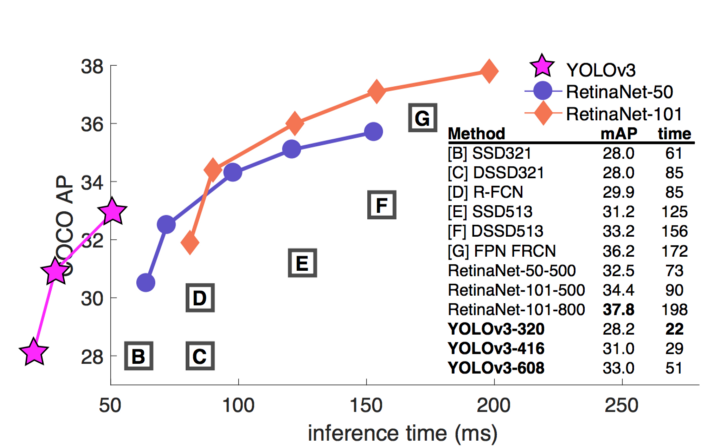
Then the fully connected layers and the last convolution layer is removed for a detector. YOLO adds three 3 × 3 convolutional layers with 1024 filters each followed by a final 1 × 1 convolutional layer with 125 output channels. (5 box predictions each with 25 parameters) YOLO also add a passthrough layer. YOLO trains the network for 160 epochs with a starting learning rate of 10−3 , dividing it by 10 at 60 and 90 epochs. YOLO uses a weight decay of 0.0005 and momentum of 0.9.

**YOLOv3 performance**

YOLOv3's COCO AP metric is on par with SSD but 3x faster. But YOLOv3’s AP is still behind RetinaNet. In particular, AP@IoU=.75 drops significantly comparing with RetinaNet which suggests YOLOv3 has higher localization error. YOLOv3 also shows significant improvement in detecting small objects.



YOLOv3 performs very well in the fast detector category when speed is important.



**5. REFERENCES:**

(a) Yogiraj Kulkarni, Shubhangi Bodhke, Amit Kamthe and Archana Patil, “Automatic Number Plate Recognition for Motorcyclists Riding Without Helmet”in Proceeding of 2018 IEEE International Conference on Current Trends toward Converging Technologies, Coimbatore, India.

(b) Jimit Mistry, Aashish K. Misraa, Meenu Agarwal, Ayushi Vyas, Vishal M. Chudasama, Kishor P. Upla, “An Automatic Detection of Helmeted and Non-helmeted Motorcyclist with License Plate Extraction using Convolutional Neural Network”,Electronics Engineering Department, Sardar Vallabhbhai National Institute of Technology (SVNIT), Surat, India.

(c)Dharma Raj KC1, Aphinya Chairat2, Vasan Timtong2, Matthew N. Dailey2, “Helmet Violation Processing Using Deep Learning”, Mongkol Ekpanyapong2 Asian Institute of Technology Khlong Luang, Pathum Thani 12120, Thailand.

(d)C. Vishnu, Dinesh Singh, C. Krishna Mohan and Sobhan Babu, “Detection of Motorcyclists without Helmet in Videos using Convolutional Neural Network”, Visual Intelligence and Learning Group (VIGIL), Department of Computer Science and Engineering Indian Institute of Technology Hyderabad, Kandi, Sangareddy-502285, India.

**6**. **ACKNOWLEDGEMENT**

The written word has an unfortunate tendency to convert genuine appreciation into an embossed formality. But this is solitary mode by which we can record our approaches everlastingly. Numerous individuals have contributed their time and energies in aiding us complete our disserting labor magnificently.

We would like to express our sincere gratefulness to our mentors, Dr. Ramchand Hablani, and Dr. Shailendra S Aote, Computer Science Department whose uninterrupted supervision and stimulation facilitated us to triumph grand altitudes.

We are also thankful to Dr. M. B. Chandak , Head, Computer Science and Engineering Department and Dr. R. S. Pande, Principal, RCOEM, for their constant support and help which encouraged us a lot in completing this project. And last but not the least we would like to acknowledge our parents, friends, contemporaries and all those people known or unknown individuals who have directly or indirectly helped us in carrying out our work successfully.