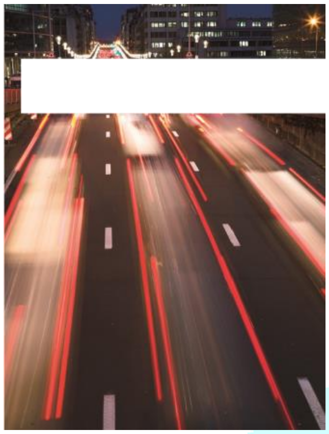
IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal



INTELLIGENT TRAFFIC SYSTEM INTELLIGENT TRAFFIC MANAGEMENT SYSTEM USING YOLO MACHINE LEARING MODEL

GUIDED BY : D.V.RAJ KUMAR

EDITED BY : A.PRADEEPKUMAR

SAMIDURAI D, Paavai Engineering College

SETHUPATHI C, Paavai Engineering College

PRADEEPKUMAR A, Paavai Engineering College

ABSTRACT People in today's era usually have the tendency of using their own private vehicles for commutation rather than using public transit and this result in large number of private vehicles on road. It leads to traffic congestion at every roads. In such scenario one cannot restrict individual to limit the usage of their private vehicles but we can able to manage traffic flow in a way that it doesn't alleviate congestion issues. The traditional traffic management approach works efficiently only if the traffic is less, but if the density of vehicles on a particular side of road increases on one side than other side, this approach fails. Hence, we aim to redesign the traffic signal system from static switching to dynamic signal switching, which can perform instant-time signal monitoring and handling. There are many projects emerging in order to convert the current transport system of cities to 'Smart system', by introducing Intelligent Transport System. Many initiatives are taken to design a system that can perform instant monitoring of traffic signals i.e., the traffic signal switching time will depend on the count of vehicles on each side of the road instead of predefined switching time. The switching time of signal will be decided based on vehicle detection in day-to-day traffic scenarios with good accuracy. This practice can prove its effectiveness in releasing the congested traffic at an efficient and faster rate.

INTRODUCTION

1.1 TRAFFIC MANAGEMENT SYSTEM

The history of Traffic Management System started in 1972 to centrally control the freeway system in the Twin Cities metro area. The Traffic Management System aims to provide motorists with a faster, safer trip on metro area freeways by optimizing the use of available freeway capacity, efficiently managing incidents and special events, providing traveller information, and providing incentives for ride sharing. Cities and traffic have developed hand-inhand since the earliest large human settlements. The same forces that draw inhabitants to congregate in large urban areas also lead to sometimes intolerable levels of traffic congestion on urban streets. Cities are the powerhouses of economic growth for any country. Transportation system provides the way for movements and medium for reaching destinations. Inadequate transportation system hampers economic activities and creates hindrances for development.

In most of the developing countries which are overburdened by rising population and extreme poverty, increasing economic activities and opportunities in the cities result in rapid increase in urban population and consequent need for transportation facilities. Authorities in these countries often fail to cope with the pressure of increasing population growth and economic activities in the cities, causing uncontrolled expansion of the cities, urban sprawl, traffic congestion and environmental degradation. Transportation and property are important in physical and economic development of towns and cities all over the world. Property and land values tend to increase in areas with expanding transportation networks, and increase less rapidly in areas without such improvements. Rapid and continued rise in housing and land prices are expected in cities with transportation improvements, rapid economic and population growth. The world would be severely limited in development without transportation, which is a key factor for physical and economic growth.

Developments of various transportation modes have become pivotal to physical and economic developments. Such modes include human porterage, railways, ropeways and cableways, pipelines, inland waterways, sea, air, and roads. Transport is critical to economic development, both low volume/rural roads and major arterials, and there is a direct relationship between a country's economic prosperity audiometers of paved roads. Since, traffic congestion creates quite an obstruction for smooth functioning of public transports, thus leading to avoidance of those by common people. According to a survey, an average person spends about 300 hours every year, waiting on traffic signals, which boils down approximately to an hour daily. Thus, traffic is an important part of our lives, not only having an impact on our transportation but also have a significant effect on our urban environment. Thus, it is a necessity to have a better functioning traffic control system implementation. Generally, the traffic system is controlled by three signal lights- green, red and yellow. The reason for the traffic congestion (commonly termed as traffic jams) is increasing number of vehicles and poor management of traffic algorithms.

1.2 Machine Learning :

Machine learning (ML) means that the computer can figure out a solution without being specifically programmed. That is, machines are able to continuously learn and deal with huge datasets using classifier algorithms. Classifiers, which categorize observations, are considered the backbone of ML. Meanwhile, other ML algorithms are built models of behaviors and use those models as a basis for making future predictions based on new input data. The power of machine learning tools lies in detecting and analyzing network attacks without having to accurately describe them as previously defined. Machine learning can aid in solving the most common tasks including regression, prediction, and classification in the era of extremely large amount of data and cybersecurity talent shortage. Machine-learning techniques have been applied in many aspects of network operation and management, where the system performance can be optimized and resources can be better utilized. Moreover, clustering and classification extract patterns out of data packets which can be used in many

The transportation route is part of distinct development pattern or road network and mostly described by regular street patterns as an indispensable factor of human existence, development and civilization. The route network coupled with increased transport investment result in changed levels of accessibility reflected through Cost-benefit analysis, savings in travel time and other benefits. These benefits are noticeable in increased catchment areas for services and facilities like shops, schools, offices, banks, and leisure activities. Road networks are observed in terms of its components of accessibility, connectivity and Traffic density, level of service, compactness and density of particular roads. Level of service is a measure by which the quality of service on transportation devices or infrastructure is determined, and it is a holistic approach considering several factors regarded as measures of traffic density and congestion rather than overall speed of the journey. Access to major roads provides relative advantages consequent upon which commercial users locate to enjoy the advantages. Modern businesses, industries, trades and general activities depend on transport and transport infrastructure, with movement of goods and services from place to place becoming vital and inseparable aspects of global and urban economic survival.

There is no fixed infrastructure for every junction, street and road which lead to loopholes in construction of fixed timing algorithms. Previously, human administrated or automated offline software were used for computation of time slots given to each signal at traffic signals. But these timings used to fail at specific times of the day or particular days (festivals etc.), which led to the development of self-automated online system in our project that continuously sense the environment and compute the timings to be given to traffic signal at a particular instant. The purpose of Traffic Management System is to improve transport operations and transport services profitability, reduce traffic jams and fatalities, provide sufficient driving, training, maintain road infrastructure, and maintain traffic law enforcement using the help of Machine Learning.

There are four steps of machine learning model that is useful for prediction for the prediction process.

1. Identify classes from training data.
2. Create a model using the training dataset that is being trained by ML algorithm.
3. During test phase, use the trained model to classify the unknown data and makes a prediction.
4. The prediction is evaluated for accuracy.

If the accuracy is not acceptable, the Machine Learning algorithm is trained repeatedly with an augmented training data set.

There are two main types of ML approaches, which are supervised and unsupervised.

applications such as security analysis and user profiling. Furthermore, there are many applications for analyzing traffic based on the ML algorithms such as identifying anomalies through discovery-based workbooks or features that describe user behavior.

1.3 Intelligent Traffic Management System using Machine Learning:

With the highly rising traffic congestion all around the world, and it's management by traditional approach are not efficient for smooth commutation purpose. Hence, there is a need to come up with a solution which can be globally accepted and would lead for the better management of traffic. In today's traditional approach the signal switches at its predefined regular interval, but the density of vehicles of the road at every signal doesn't remains the same, hence the static approach fails. Under such scenario, if the signal remains the same to switch at its regular interval then the side of road which is densely populated will always remain completely packed. As mentioned in above systems, till date they are to getting vehicle count only, so that comparative study and analysis of traffic can be done.

There are many projects emerging in order to convert the current transport system of cities to 'Smart system' and there are many initiatives under this, one of this is Intelligent Transport System. Many initiatives were taken to design a system that can perform real-time monitoring of traffic signals i.e., the traffic signal switching time will not be predefined one, instead the switching time will depend on the count of vehicles on each side of the road. This process of getting the count of vehicle on the road can be achieved using various detection techniques. Techniques like Vehicle detection using sensors may fail at circumstances when the traffic gets denser at peak timings.

Our aim is to design and develop a miniature to depict the current road situation along with monitoring and handling the traffic issues. Hence to proceed with this project we are using a pre-trained YOLO Machine Learning Model to perform the task of object detection.

YOLO uses OpenCV for object detection along with multiple foreground and background subtraction and removal of noise from the input image. The CCTV cameras that are being used for surveillance purpose can be made use to capturing the footage of the road, this image will be passed to the pretrained model as input image. To do so each side of the road will be divided into particular frames of same height and width for capturing the image. The count obtained from the image is passed into a pre-defined Python program. As per the count obtained, switching time will be assigned for each side of road. The program will initially check if the count of vehicle in all lanes and then the signal switching will happen dynamically where the lane with higher vehicle count will be opened first. , YOLO architecture is more like FCNN (fully convolutional neural network) and passes the image size NxN once through the FCNN and output size is MxM prediction. This architecture is splitting the input image size as MxM grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes is done.

SYSTEM ANALYSIS AND SPECIFICATION

- Supervised learning:

It is a classification method, which trains the labeled data set to produce new prediction outputs, given input variables and output variables.
In Supervised learning, learning continues until the algorithm reaches an acceptable level of performance. The algorithm constantly predicts outcomes based on training data, and it is constantly corrected.

- Unsupervised Learning:

This technique is called clustering method, where dataset does not need to be labeled; only input data will be given. The aim of unsupervised learning is to learn more about data by modeling infrastructure or basic distribution of data.

In our project, the supervised learning approach is used for traffic analysis purpose. We can create labeled data set and pre-train the model to produce the prediction outputs.

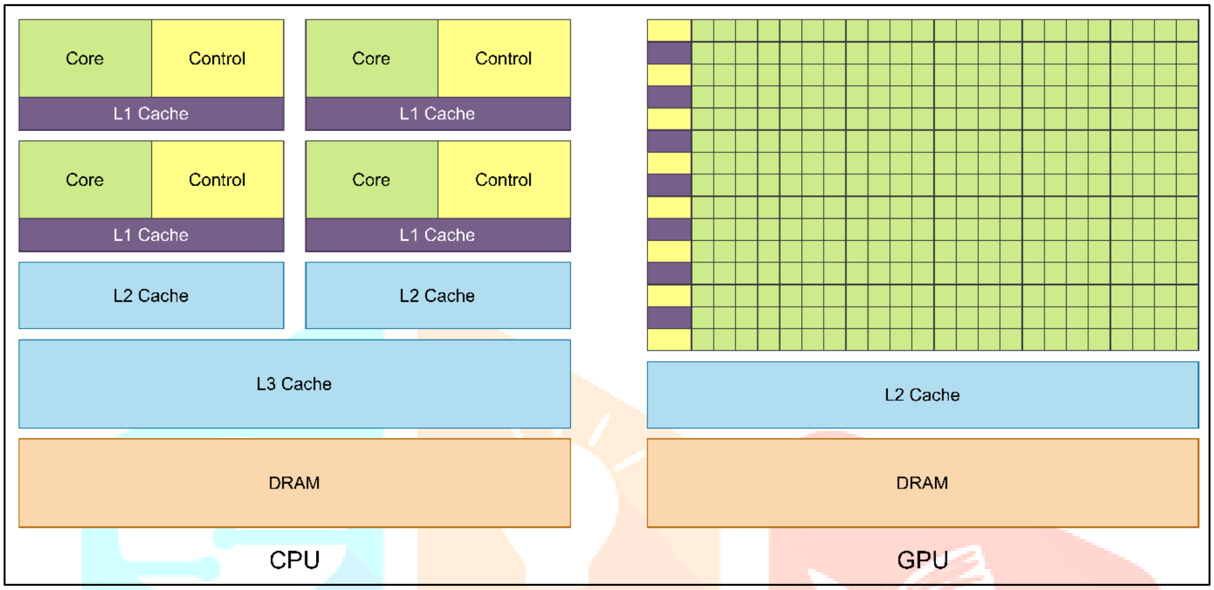
YOLO (You Only Look Once), is a network for object detection. It is the one of the most powerful pretrained model to give utmost accuracy. Yolo is a combined version of RCNN (Region-based Convolutional Neural Networks) and SSD (Single Shot Detector), both make YOLO much faster, efficient and powerful algorithm. By applying object detection algorithm in YOLO, one will not only be able to determine what is in an image, but also where a given object is placed i.e., the location. Also, the model is trained using huge dataset hence it can detect image placed in any random manner i.e., it can detect object even if they are rotated in 360 degree. YOLO is an efficient model by distinguishing between two very closely placed objects. Unlike traditional approach of applying classifier on each image and making prediction, YOLO look at the image once and but in a clever way. It divides the image into N numbers of partitions and into MxM grid. Now YOLO applies its algorithm one by one in partitions and predict confidence score/ Confidence score is the score that tells us whether object is present or not. On the basis of the confidence score, YOLO detects an object.

YOLO can process many frames with less execution time as compared to other pretrained models. YOLO computes its prediction in terms of precision and recall, precision measures how accurate the predictions are and recall measures how good we find all the positives i.e., how correctly the objects are classified. To increase its performance factor YOLO uses IoU, Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. IoU defines how two closely place objects can be easily detected without hampering the accuracy of the model. YOLO consist of two core components. One of the YOLO's component R_CNN uses selective search algorithm and proposes accurate bounding box that definitely contains objects whereas the other component SSD that helps to speed up the processing of an image. Compared to other region proposal classification networks (fast RCNN) which

3.1 SYSTEM ANALYSIS

3.1.1 Existing System With the highly rising traffic congestion all around the world, and it's management by traditional approach are not efficient for smooth commutation purpose hence there is a need to come up with a solution which can be globally accepted and would lead for the better management of traffic. In today's world where technology has transcended all barriers it has now become easy to solve most human problems and one of these problems include Traffic Congestion. Traffic congestion has increased drastically over the years and has had negative impacts that include road rage, accidents, air pollution, wastage of fuel and most importantly unnecessary delays. The fact that encouraged proposing new solution is that in many cities of the world, the traffic signal allocation is still based on timer. The Timer Approach has a drawback that even when there is a less traffic in one of the roads, green signal is still allocated to the road till its timer value falls to 0, whereas the traffic on another road is comparably more faces red signal at that time. This causes congestion and time loss to commutators. Most of the present systems are not automated and are prone to human errors. There are many projects emerging in order to convert the current transport system of cities to 'Smart system' and there are many initiatives under this, one of this is Intelligent Transport System. Many initiatives were taken to design a system that can perform real-time monitoring of traffic signals i.e., the traffic signal switching time will not be predefined one, instead the switching time will depend on the count of vehicles on each side of the road. **High-end Graphics Processing Unit:** Machine learning models require a lot of computational power to run on. For any neural network, the training phase of the deep learning model is the most resource-intensive task. Traditionally, the training phase of the deep learning pipeline takes the longest to achieve. This is not only a time-consuming process, but an expensive one. While training, a neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training and the model then spits out a prediction. Weights are adjusted to find patterns in order to make better predictions. To significantly reduce training time, we can use machine learning GPUs, which enable us to perform AI computing operations in parallel. GPUs are optimized for training artificial intelligence and deep learning models as they can process multiple computations simultaneously. GPUs are parallel processors designed to accelerate portions of a program, but not to replace CPU computing. The main program is executed on the CPU, but some code fragments, called kernels, are executed on the GPU. These Graphical processing units (GPUs) can reduce these costs, enabling us to run models with massive numbers of parameters quickly and efficiently. This is because GPUs enable us to parallelize the training tasks, distributing tasks over clusters of processors and performing compute operations simultaneously. GPUs are also optimized to perform target tasks, finishing computations faster than non-specialized hardware. These processors process the same tasks faster and free the CPUs for other tasks. This eliminates bottlenecks created by compute limitations. perform detection on various region proposals and thus end up performing prediction multiple times for various regions in an image

3.1.2 PROPOSED SYSTEM Our aim is to design and develop a machine learning model to handle the traffic signal switching by depicting the number of vehicles present in a road along with detection of different types of vehicles present in the road. The proposed system helps to develop a solution that analyses the presence of vehicles on the road and handles the traffic congestion issues, resulting in a better managed, more coordinated and smarter use of traffic networks. This can be done using the analysis of vehicle count data obtained from source like CCTV Cameras present in highways or in traffic signals, using a trained machine learning model called **YOLO**. YOLO is an OpenCV based machine learning model which does the Object Detection and counts the number of vehicles in a lane. The recorded data is then sent into the predefined python program where the machine learning model is already written and based on the obtained vehicle count data – we can dynamically switch the signal among the lanes. Thereby, round the clock safety and hassle-free traffic management can be obtained using Proposed Intelligent Traffic Management System. Implementation of our project will eliminate the need for traffic personnel at various junctions for regulating traffic. Thus, the use of this technology is valuable for the analysis and performance improvement of road traffic. Also, priority to emergency vehicles has been the topic of some research in the past which can be enabled with further training of our machine learning of our model.**Selecting the right GPU for our project:** Selecting the GPUs for the implementation has significant budget and performance implications. We need to select GPUs that can support the project in the long run and have the ability to scale through integration and clustering. For large-scale projects, this means selecting production-grade or data center GPUs. In the GPU market, there are two main players i.e AMD and Nvidia. Nvidia GPUs are widely used for machine learning because they have extensive support in the forum software, drivers, CUDA, and cuDNN. So, in terms of AI and machine learning, NVIDIA is the pioneer for a long time. NVIDIA GPUs are the best supported in terms of machine learning libraries and integration with common frameworks, such as PyTorch or TensorFlow. The NVIDIA CUDA toolkit includes GPU-accelerated libraries, a C and C++ compiler and runtime, and optimization and debugging tools. It enables us to get started right away without worrying about building custom integrations. **NVIDIA CUDA Powered GPUs:** CUDA stands for 'Compute Unified Device Architecture' which was launched in the year 2007, it's a way in which we can achieve parallel computing and yield most out of GPU power in an optimized way, which results in much better performance while executing tasks. The CUDA toolkit is a complete package that consists of a development environment that is used to build applications that make use of GPUs. Also, the CUDA runtime has its drivers so that it can communicate with the GPU. cuDNN is a neural network library that is GPU optimized and can take full advantage of Nvidia GPU. This library consists of the implementation of convolution, forward and backward propagation, activation functions, and pooling. It is a must library without which we cannot use GPU for training neural networks. The main difference between GPUs and CPUs is that GPUs devote proportionally more transistors to arithmetic logic units and fewer to caches and flow control as compared to CPUs. A GPU is smaller than a CPU but tends to have more logical cores (arithmetic logic units, control units and memory cache) than the latter.



Google Colab Cloud Environment: Google Colaboratory is an online cloud-based platform based on the Jupyter Notebook framework, designed mainly for use in machine learning operations. There are many distinguishing features that set it apart from any other coding environment.

Reason for choosing Google Colab over PC: One of the main benefits of using Colab is that it has most of the common libraries that are needed for machine learning like TensorFlow, Keras, Scikit Learn, OpenCV, numpy, pandas, etc. pre-installed. Having all of these dependencies means that we can just open a notebook and start coding without having to set up anything at all. Any libraries that are not pre-installed can also be installed using standard terminal commands. While the syntax for executing terminal commands remains the same, one must add an exclamation mark (!) at the start of the command so that the compiler can identify it as a terminal command. Another feature is that the Colab environment is independent of the computing power of the computer itself. Since it is a cloud-based system, as long as there is an internet connectivity, even heavy machine learning operations can be run from a relatively old computer that ordinarily wouldn't be able to handle the load of executin those operations locally. Additionally, Google also offers a GPU (Graphics Processing Unit) and a TPU (Tensor Processing Unit) for free. These hardware accelerators

3.2.2 Software Requirements: Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. Some of the software requirements for our project are,

1. Operating System
2. Python (Programming Language)
3. Darknet (Neural-network framework)

Operating System An operating system is system software that manages computer hardware, software resources and provides common services for computer programs. Since we are using Google Colab Cloud Environment for our project, we will be using the services of Debian Linux Operating System.Reason for Debian to be the default operating system in Google Colab is because the Debian Family of Linux has official support for CUDA, Kubernetes, TensorFlow and Keras by default. So, it becomes easy for user work around our cloud environment with basic Linux commands.

Python Our machine learning model is written with Python programming language as it is the most preferred language for developing machine learning can run heavy machine learning operations on large datasets much faster than any local environment. While Colab allows uploading local files onto the runtime each time it is loaded, uploading and re-uploading large training datasets each time the runtime is restarted can be frustrating. Colab also offers data versatility, a simple alternative 'mount' Google Drive onto the Colab notebook. This operation requires just two lines of code that Colab inserts a file with the click of a button and this enables access to read files that is uploaded into Google Drive. This means that we don't have to reupload local files after every runtime restart. Simply uploading them once and access them simply by mounting the Google Drive solves the issue. Like the rest of Google's online document editing platforms like Google docs, Google Slides, Google Sheets, etc., Colab too offers similar sharing options allows to seamlessly collaborate with others on joint coding projects. One thing to keep in mind is that when a notebook is shared, other users cannot see the output and results from code that one has executed. Also, if one uploads some files from their computer to the notebook, other collaborators will not be able to see them so it is better to upload those files to Google Drive and then access them from there so everyone can see and use the files.

Limitations of Google Colab: Google Colab has a 'maximum lifetime' limit of running notebooks that is 12 hours with the browser open, and the 'Idle' notebook instance is interrupted after 90 minutes. In addition, a Google Account on Colab can run a maximum of 2 notebooks simultaneously. GPUs and TPUs are sometimes prioritized for users who use Colab interactively rather than for long-running computations, or for users who have recently used3 less resources in Colab. As a result, users who use Colab for long-running computations, or users who have recently used more resources in Colab, are more likely to run into usage limits and have their access to GPUs and TPUs temporarily restricted Resources present in a Colab session's Storage will be automatically deleted once the session gets restarted, so we cannot access the files that are stored during the previous session after recycling it.

Darknet Framework:

Darknet is an open-source neural network framework like Keras, PyTorch and TensorFlow. Darknet is written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. Darknet is installed with only two optional dependencies: OpenCV if users want a wider variety of supported image types or CUDA if they want GPU computation.

models. Python is a very useful programming language that has an easy-to-read syntax, and allows programmers to use fewer lines of code than would be possible in languages such as assembly, C, or Java.

Reasons for choosing Python : One of the key reasons for using Python for Machine Learning is its great library ecosystem. A library is a module or a group of modules published by different sources like 'PyPi' which include a pre-written piece of code that allows users to reach some functionality or perform different actions. Python libraries provide base level items so developers don't have to code them from the very beginning every time. Machine Learning requires continuous data processing, and Python's libraries let users access, handle and transform data. These are some of the most widespread libraries namely,

- Scikit-learn : For handling basic ML algorithms like clustering, linear and logistic regressions, regression, classification, and others.
- Pandas : For high-level data structures and analysis. It allows merging and filtering of data, as well as gathering it from other external sources like Excel, for instance.
- Keras : It allows fast calculations and prototyping, as it uses the GPU in addition to the CPU of the computer.
- TensorFlow : For working with deep learning by setting up, training, and utilizing artificial neural networks with massive datasets..

Python for machine learning is a great choice, as it is very flexible. The flexibility factor decreases the possibility of errors, as programmers have a chance to take the situation under control and work in a comfortable environment.

- It offers an option to choose either to use OOPs or scripting.
- There's also no need to recompile the source code, developers can implement any changes and quickly see the results.
- Programmers can combine Python and other languages to reach their goals.

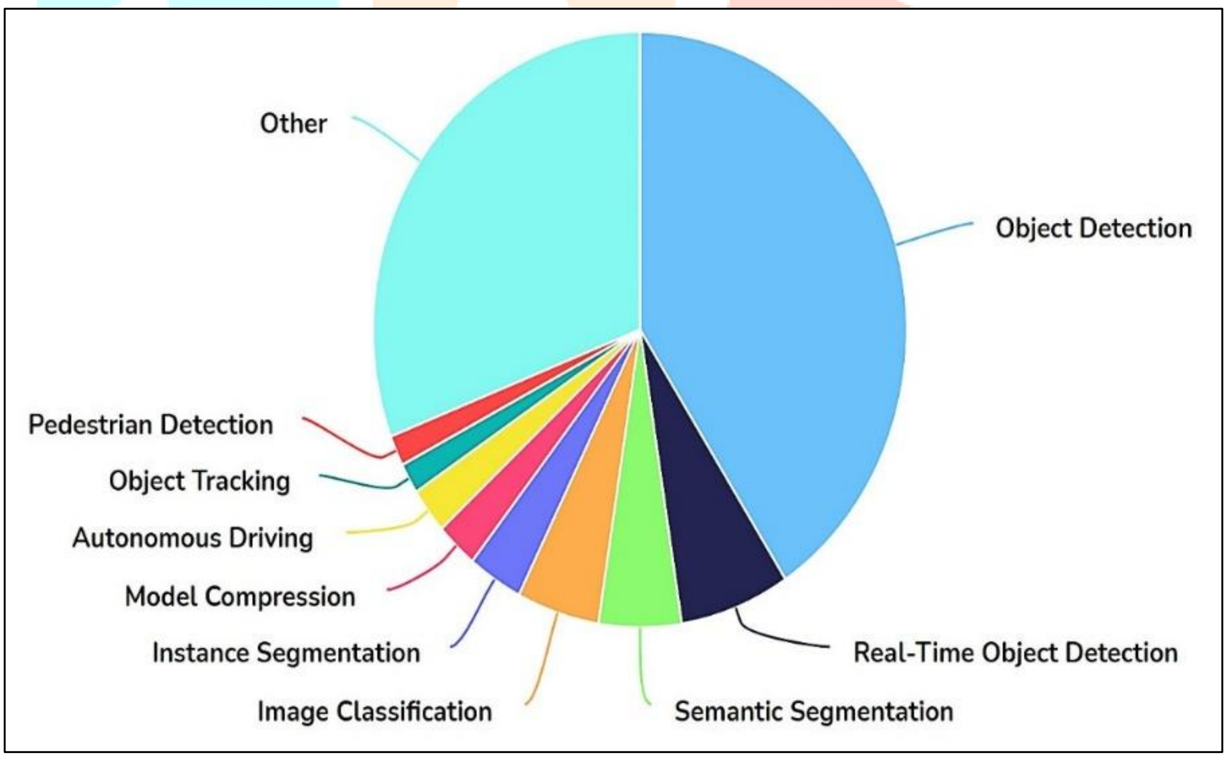
Moreover, its flexibility allows developers to choose the programming styles which they are fully comfortable with or even combine these styles to

The framework features You Only Look Once (YOLO) Machine Learning

Algorithm, a state-of-the-art, real-time object detection system. On a Titan X it processes images at 40-90 FPS and has a mAP on VOC 2007 of 78.6% and a mAP of 44.0% on COCO test-dev. Darknet displays information as it loads the config file and weights then it can be enabled to classify the image and print the top-10 classes for the image. Moreover, the framework can be enabled to run neural networks backward in a feature appropriately named Darknet Nightmare.
Recurrent neural networks are powerful models for representing data that changes over time and Darknet can handle them without making use of CUDA or OpenCV. The framework also allows its users to venture into game-playing neural networks. It features a neural network that predicts the most likely next moves in a game of Go. Users can play along with professional games and see what moves are likely to happen next, make it play itself, or try to play against it. solve different types of problems in the most efficient way.

.

Reason for choosing Darknet over the rest:
Darknet is mainly for Object Detection, and have different architecture, features than other deep learning frameworks. It is faster than many other NN architectures and approaches like FasterRCNN etc. One have to be in C if one needs speed, and most of the deep NN frameworks are written in c. TensorFlow has a broader scope in Machine Learning, but Darknet architecture & YOLO is a specialized framework, and it is in top of its game in speed and accuracy. YOLO can run on CPU but one can get 500 times more speed on GPU as it leverages CUDA and cuDNN



Principle of YOLO :

YOLO is refreshingly simple, single convolutional network model that simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline. We simply run our neural network on a new

4. MODULE DESCRIPTION

Our proposed system consists of 4 modules and divided into 2 phases. The modules are namely,

1. Machine Learning Model Setup and Development
2. YOLO Machine Learning Model Training & Weight Creation

image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in an image, Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image of size NxN once through the FCNN and output of size MxM prediction. YOLO architecture is splitting the input image in MxM grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since we frame detection as a regression problem, we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

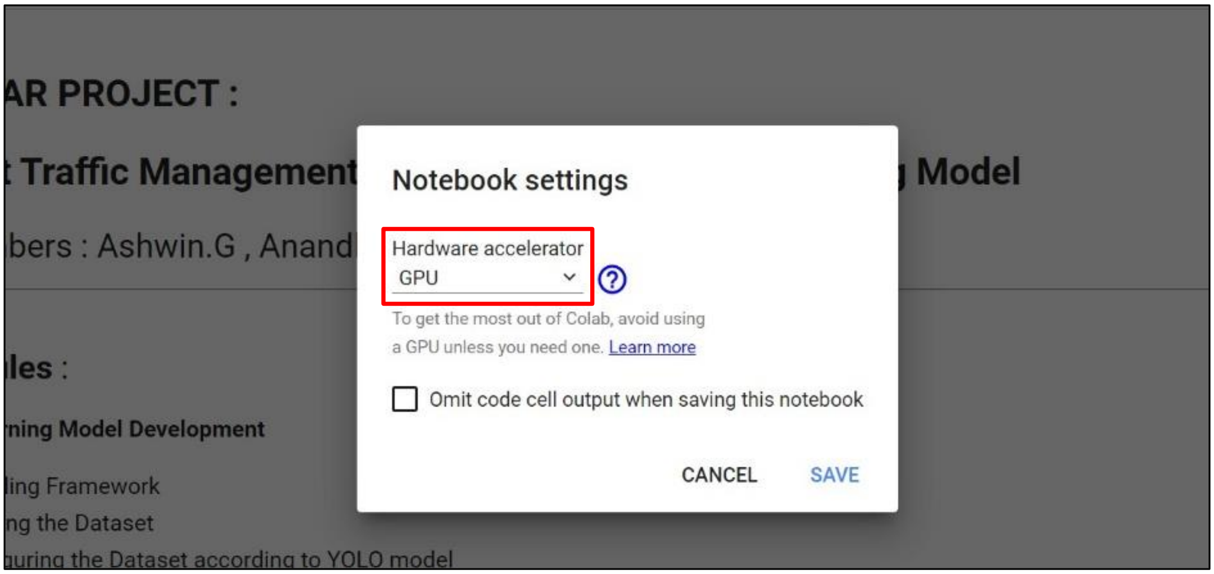
3. Vehicle Detection and Counting of Vehicles by YOLO Model
4. Dynamic Signal Switching

4.1 Machine Learning Model Setup and Development: Before starting to develop our machine learning model with our own dataset, we must prepare the suitable environment for our model to develop it in a faster way – because creation of a machine learning model is a tedious process and it takes huge computation power to develop it. So, we are developing our machine learning model in Google Colab, which will drastically save our computation time and helps us to develop comparably faster than what it might actuatake to develop in our personal computer. Google

Colab Environment's automatically provisioned computation power is arguably 100x faster than the computation capability of our local host.

4.1.1 Setting up the Google Colab Environment: As we knew already, Google Colab Cloud Environment will dynamically provision resources for each session based on our computations. Since we are need to train the dataset for developing a fully-functioning YOLO Machine Learning Model, we must change the Runtime Instance of our current Google Colab Session into GPU/TPU Runtime. Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.



Makefile is a program building tool which runs on Unix, Linux, and their flavors. It aids in simplifying building program executables that may need various modules. To determine how the modules need to be compiled or recompiled together, **make** takes the help of user-defined makefile. As we knew that NVIDIA Tesla K80 is the GPU present in our runtime environment, we can now install the darknet neural-network framework now by creating a folder called

"darknet" and cloning the GitHub repository of Darknet. Once after cloning the darknet framework into our environment, start creating the configuration file for our darknet framework inside the darknet folder. We must modify and overwrite the "make file" (configuration) for our Darknet framework which is compatible with the computations that will be required for training our machine learning model using this framework. Compute capability of a GPU determines its general specifications and available features. Since our GPU is NVIDIA Tesla K80, the compute capability of our GPU is 30, which needs to be set in the 'Make File' of Darknet Framework. Since the Colab's GPU dependencies shift from time to time automatically, we need to run the makefile after checking the actual dependency to which our Colab Notebook is currently connected to. Currently, our Colab Notebook is connected to NVIDIA Tesla K80 GPU. If the GPU is shifted to another GPU, we need to tweak the 'Make File' accordingly. For example, with the Cityscape dataset is one of the most widely adapted for developing the object detection algorithms, but for India, where traffic violations are rampant, these datasets can't be inculcated to ensure safer road travel.

1. Pixel counts for each label in the y axis.
2. The four-level label hierarchy and the label ids for intermediate levels (level 2, level 3).

4.1.3 Dataset Collection:

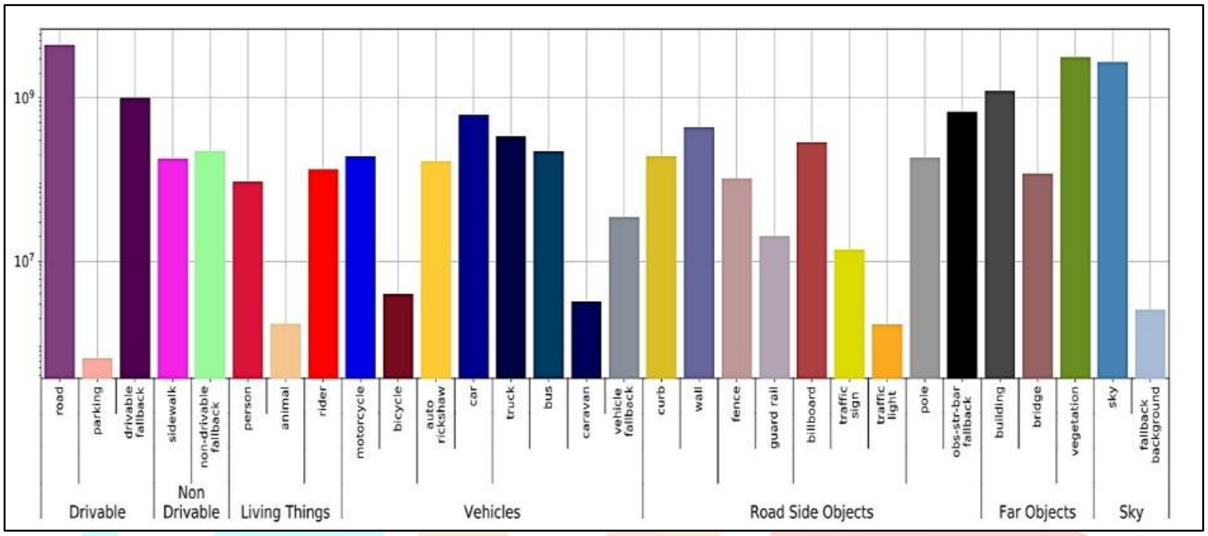
For developing a machine learning model, one need to create an image dataset first. A dataset assembles a collection of images that are labeled and used as references for objects that are used by the developers to test, train and evaluate the performance of their algorithms. Algorithms trained with larger datasets perform significantly better than those trained on smaller ones. With more data come more variations and the algorithm can learn from the myriads of differences of the visual world. The quality of the model depends on the quality of the data set input. Creating a dataset is not always a simple matter. We must collect, annotate, convert into model supported format and then insert the dataset into the model for training the data which might take hours to several days.

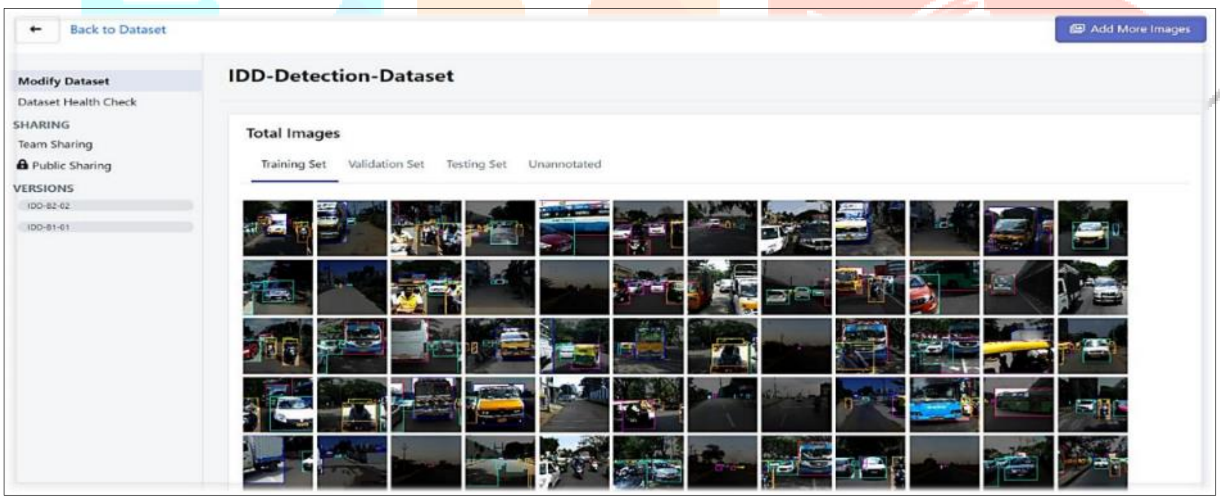
4.1.3.1 Dataset used in our model:

While several datasets are already available to develop machine learning models, they tend to focus on neatly structured driving environments. This usually corresponds to well-delineated infrastructure such as lanes, a small number of well-defined categories for traffic participants, low variation in object or background appearance and strict adherence to traffic rules. We chose **IDD – India Driving Dataset** as a main source of training images for our project. IDD is a novel dataset for road scene understanding in unstructured environments where the above assumptions are largely not satisfied. It consists of 10,004 images, finely annotated with 34 classes collected from 182 drive sequences on Indian roads. The label set is expanded in comparison to popular benchmarks such as Cityscapes, to account for new classes. It also reflects label distributions of road scenes significantly different from existing datasets, with most classes displaying greater within-class diversity.

Our dataset annotations have unique labels like billboard, auto-rickshaw, animal etc. We also focus on identifying probable safe driving areas beside the road. The labels for the dataset are organized as a 4-level hierarchy. Unique integer identifiers are given for each of these levels. The histogram bellow gives:

3. The color coding used for the prediction and ground truth masks are given to the corresponding masks.

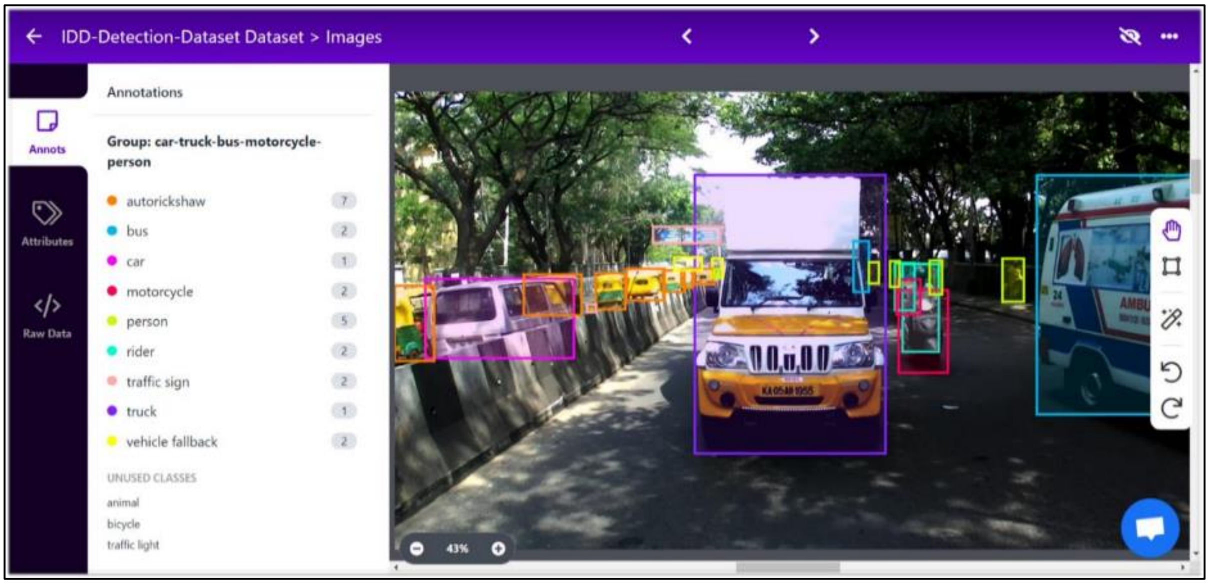




Roboflow accepts images along with its XML files where we have the details of our annotation. As IDD Dataset comes pre-annotated with the help of XML files with them, we can upload the image as well as it's XML file into Roboflow which largely conserves our preparation time for pre-annotated dataset images. In addition to that, we can use the existing dataset as a part of custom dataset that we are planning to create. Like in our case where we are creating a dataset with IDD Dataset as a main contributor and vehicle images from web search as part of our dataset. Here, only the images from IDD Dataset are already annotated and the files that we have uploaded from sources like Google Open-Images Dataset and Google Search might be missing the annotation for them. Roboflow even offers free annotation tool which filters the images in our custom dataset with missing annotation and helps us annotate those images.

During the Dataset Generation process in Roboflow, we can split the dataset as training , validation and testing set which might help us validate how well the model gets trained with this dataset. We can also upload images as batches and save different versions of same dataset in Roboflow.

Roboflow offers various other image pre-processing services like Auto-Orient Images, resizing all the dataset images, merging color channels to make our model faster and insensitive to subject color. Boosting contrasts based on the image's histogram to improve normalization and line detection in varying lighting conditions. Roboflow just like Google Colab is a freemium service where we have to pay for using it above limited service. So, we can opt for premium option if we wish to scale up the size of our dataset.



Dataset Extraction: Using the API Key obtained at the end of conversion of dataset by Roboflow, we can now the same to unzip the dataset in our Google Colab Cloud Environment Notebook. After unzipping process, we can now start setting up the directory path for our YOLO – Darknet Framework to detect the extracted data. Once the extraction process is done, we must write the configuration file for our model based on the number of classes available in our dataset used for training. We build the configuration file iteratively from the base configuration file available on the Darknet Framework's cloned open-source repository. Soon after setting up the configuration file's variables, we can now know that the file is written in our runtime with the help of output displayed at the end of execution. Now, the model can be started to train. The model runs on the configuration file that we have created moments ago. **4.2.3. Dataset Training & Weight Creation:** Now, we will be using the YOLO Darknet Detector to train the model. When the mAP for first 1000 iterations are done, now the mAP score is calculated and then mAP score will be calculated after further 100 iterations at 1100 iterations. At 1100 iterations - the mAP(mean Average Precision) Score is calculated and compared with the previous mAP score and the training continues for 1200 iterations. This process continues for hours/days according to the size of the dataset and the files that are present.

4.3.1 YOLO Model - Network Architecture:

Our YOLO model has 24 convolutional layers followed by 2 fully connected layers. It uses 1 x 1 reduction layers followed by a 3 x 3 convolutional layer. The 7x7 layer(rightmost) is one of the many bounding boxes that is classified by our YOLO Model. Our model applies its algorithm in each of these many bounding boxes that our model has already classified. The entire process is explained below [5].

The system here divides the input image received, into an S x S grid. Each of these grid cells predicts B bounding boxes and confidence scores for these boxes.

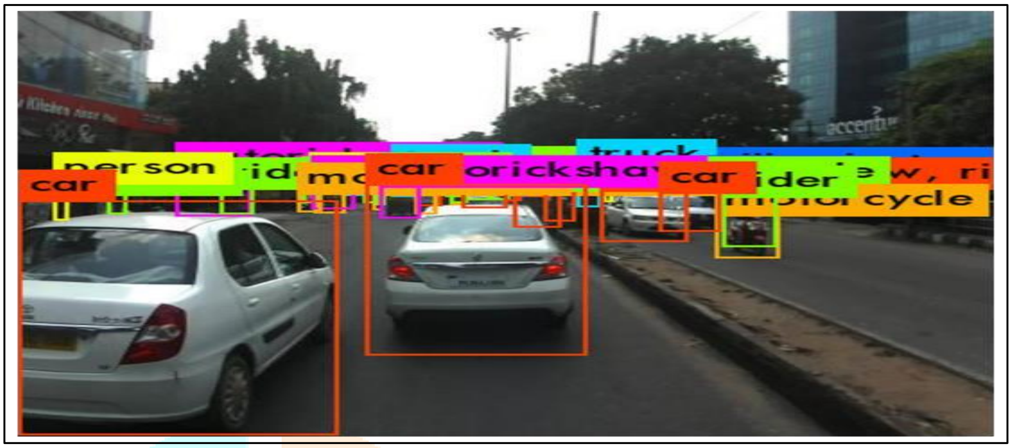
Currently, our project's training was done until 1406 iterations - which nearly took 5 hours with 82 hours of training left and 67488 images getting trained. Here the 67488 images are not the actual 67488 images, because a single image can be divided into N*N grid, so the actual set of images trained is unclear currently.

At the end of training, we will be able to notice the "weights" necessary for functioning of our machine learning model.

To test whether the trained weight is able to detect objects present in a sample image, we can pass that image into the detector present in Darknet framework. With our testing results, we conclude that our weights are good to go for Vehicle Detection with our YOLO Machine Learning Model. The Weights can be extended with further training of the model with additional training data and the weights will continue to with better accuracy in extensive training.

4.3 YOLO Machine Learning Model and Vehicle Detection:

The YOLO(YouLookOnlyOnce) model is a combined version of RCNN and SSD for object detection which gives utmost accuracy and also it is a much faster, efficient and powerful algorithm. The YOLO framework (You Only Look Once) takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. It outperforms other detection methods, including DPM (Deformable Parts Models) and R-CNN. YOLO reframes object detection as a single regression problem instead of a classification problem. This system only looks at the image once to detect what objects are present and where they are, hence the name YOLO(YouLookOnlyOnce). Also, the model can be trained using huge dataset hence it can detect image placed in any random manner. i.e., it can detect object even if they are rotated in 360 degree. Unlike traditional approach of applying classifier on each image and making prediction, YOLO first takes an input data, and then divides the input data grids. Image classification and localization are applied on each grid.



The confidence score indicates how sure the model is that the box contains an object and also how accurate it thinks the box is that predicts. The confidence score can be calculated using the formula:

*C = Pr(object) * IoU*

IoU: Intersection over Union between the predicted box and the ground truth. If no object exists in a cell, its confidence score should be zero.

YOLO then predicts the bounding boxes and their corresponding class probabilities for objects if present. Now YOLO applies its algorithm one by one in partitions and predict confidence score, confidence score is the scores that tells us whether object is present or not. On the basis of the confidence score YOLO detects an object.

Each grid cell also predicts C conditional class probabilities *Pr (Class i | Object)*. It only predicts one set of class probabilities per grid cell, regardless of the number of boxes B. During testing, these conditional class probabilities are multiplied by individual box confidence predictions which give class-specific confidence scores for each box. These scores show both the probability of that class and how well the box fits the object.

*Pr (Class i| Object) *Pr (Object)*IoU = Pr (Class i) *IoU.*

The final predictions of a confidence score are encoded as, **S x S x (B*5 + C).**

Intersection Over Union (IoU):

Usually, the threshold for IoU is kept as greater than 0.5. Although many researchers apply a much more stringent threshold like 0.6 or 0.7. If a bounding box has an IoU less than the specified threshold, that bounding box is not taken into consideration.

Looking at the boxes, someone may visually feel it is good enough to conclude that the model detected the car object. Someone else may feel the model is not yet accurate as the predicted box does not fit the ground-truth box well.

To objectively judge whether the model predicted the box location correctly or not, a threshold is used. If the model predicts a box with an IoU score greater than or equal to the threshold, then there is a high overlap between the predicted box and one of the ground-truth boxes. This means the model was able to detect an object successfully. The detected region is classified as Positive (i.e., contains an object). On the other hand, when the IoU score is smaller than the threshold, then the model made a bad prediction as the predicted box does not overlap with the ground-truth box. This means the detected region is classified as Negative (i.e., does not contain an object).

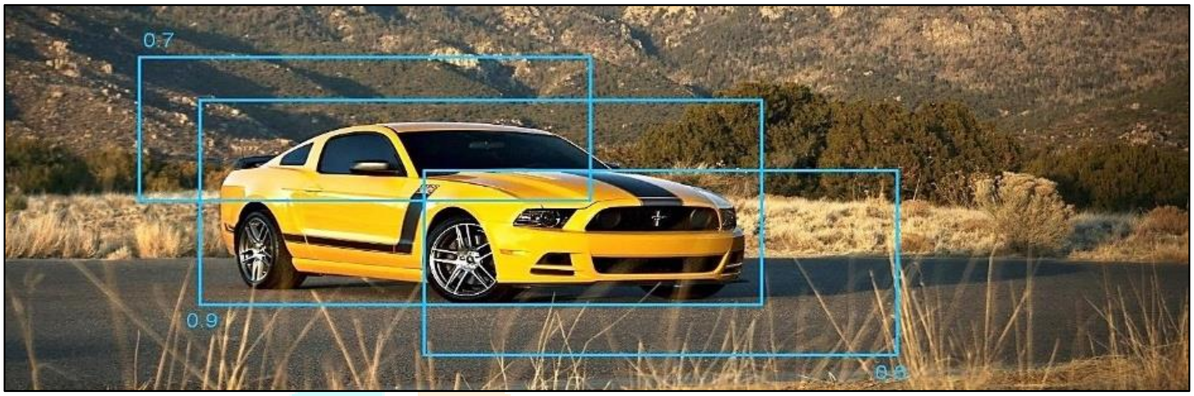
$$\boxed{class(IoU) = \begin{cases} Positive \rightarrow IoU \geq Threshold \\ Negative \rightarrow IoU < Threshold \end{cases}}$$

Fig:4.18 IoU and the use of Threshold Value

Non-Maximum Suppression:

The algorithm may find multiple detections of the same object. Non-max suppression is a technique by which the algorithm detects the object only once. Consider an example where the algorithm detected three bounding boxes for the same object.

The probabilities of the boxes are 0.7, 0.9, and 0.6 respectively. To remove the duplicates, we are first going to select the box with the highest probability and output that as a prediction. Then eliminate any bounding box with IoU > 0.5 (or any threshold value) with the predicted output. The result will be:



Vehicle Detection:

After the development of the model, we must pass an input image into the model as an argument to the startup program. The python program will initially start importing necessary libraries required for running the program along with packages from other directories. After importing all necessary libraries and packages. With loading the input from the arguments passed into the model, the model loads the 'weights' file for the model which was saved by us earlier from the second module of our project. Along with the weights the model loads the directories of configuration file and the names file containing the dataset label names. Then, the model checks if a GPU is present in the local runtime to boost the speed of the machine learning model. Now the model enters the vehicle detection phase. The model once again checks for if the image is present from the passes input argument path. Now the image enters the image processing phase. The input image is now resized into 416*416 resolution, which helps in better performance of the model to detect inputs. This configuration is built-in already in YOLO's configuration file. The resized image is then returned as a tensor variable into the model. Now the model uses the neural network weights to detect the input with bounding boxes initially and then perform non-max suppression and the detected output is displayed as result.

Vehicle Count & Dynamic Signal Switching:

As we have developed the YOLO model to count the number of vehicles present from an input source, the model detects the vehicles from the image and then counts the number of vehicles present in the given source. The count obtained from the source can now be passed into the python program for determining the threshold value of each lane which we have predefined already. The python program now compares the count of vehicles from each lane and executes further steps in the next module. The obtained data is then sent to the computer system in which we have written a python program that processes the input information and we have already predefined a threshold value based on the count of vehicles. So that the system determines the priority of each lane to open the signal. If all model detects no vehicles or same number of vehicles on each lane, the model will automatically switch to static signal switching approach.

CONCLUSION:

The main objective of Intelligent Traffic Management system is founded to fix the problem of traffic which most of the cities in urban as well as rural areas are facing with the help of this project wherein the focus would be to minimize the vehicular congestion. The setup requires traffic data as input which will then be used with our machine learning model for efficient traffic flow without creating much chaos on the road. The model may take comparatively more training time but the response time will be less. The model is

FUTURE WORK

The timer-approach can be enabled to come into existence when the model fails to detect at crucial times like bad weather and low visibility initially. The system can be added with cloud computation support in the future so that the system can log the traffic of respective lanes with date and time which will be highly effective in analyzing the traffic data for further improvement of roads This scenario can be vastly minimized with extended use of our model, since the machine learning models can learn to adapt to different scenarios with continuous use. Our model is able to add even more custom-functions to the program like closing signal for pedestrians crossing, priority for lane with ambulance and vehicle monitoring etc,. prepared in such a way that it decides smart switching timing for the signal on all sides of the road so the no one has to wait for longer interval of time on the road and flow of traffic is smooth on the road. Since the system learns from time to time, the system can be updated in future to become fully-autonomous with training and learning.

REFERENCES

[1] J. Tiwari, A. Deshmukh, G. Godepure, U. Kolekar and K. Upadhyaya, "Real Time Traffic Management Using Machine Learning," *2020 International Conference on Emerging Trends in Information Technology and Engineering (icETITE)*, Vellore, India, 2020, pp. 1-5, doi: 10.1109/ic-ETITE47903.2020.462.

[2] Indrabayu, R. Y. Bakti, I. S. Areni and A. A. Prayogi, "Vehicle detection and tracking using Gaussian Mixture Model and Kalman Filter," *2016 International Conference on Computational Intelligence and Cybernetics, Makassar,*2016,pp.115-119,doi: 10.1109/CyberneticsCom.2016.7892577.

[3] S. Maqbool, M. Khan, J. Tahir, A. Jalil, A. Ali and J. Ahmad, "Vehicle Detection, Tracking and Counting", *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, Shenzhen, 2018, pp. 126-132, doi: 10.1109/SIPROCESS.2018.8600460.

[4] R. Krishnamoorthy and S. Manickam, "Automated Traffic Monitoring Using Image Vision", *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, Coimbatore, 2018, pp. 741-745, doi: 10.1109/ICICCT.2018.8473086.

[5] C. S. Asha and A. V. Narasimhadhan, "Vehicle Counting for Traffic Management System using YOLO and Correlation Filter," *2018 IEEE*

[8] C. Kumar B., R. Punitha and Mohana, "YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications," *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT),* Tirunelveli, India, 2020, pp. 1316-1321, doi: 10.1109/ICSSIT48917.2020.9214094.

[9] J. T. G. Nodado, H. C. P. Morales, M. A. P. Abugan, J. L. Olisea, A. C. Aralar and P. J. M. Loresco, "Intelligent Traffic Light System Using Computer Vision with Android Monitoring and Control", *TENCON 2018 - 2018 IEEE Region 10 Conference*, Jeju, Korea (South), 2018, pp. 2461-2466, doi: 10.1109/TENCON.2018.8650084.

[10] Dimililer, Kamil & Kirsal Ever, Yoney & Mustafa, Sipan. "Vehicle Detection and Tracking Using Machine Learning Techniques" *International Conference on Theory and Application of Soft Computing,* 2020, doi:10.1007/978-3-030-35249-3_48.

[11] A. Dubey, M. Lakhani, S. Dave and J. J. Patoliya, "Internet of Things based adaptive traffic management system as a part of Intelligent Transportation System (ITS)," *2017 International Conference on Soft Computing and its Engineering Applications (icSoftComp)*, Changa, 2017, pp. 1-6, doi: 10.1109/ICSOFTCOMP.2017.8280081.

[12] Y. M. Jagadeesh, G. M. Suba, S. Karthik and K. Yokesh, "Smart autonomous traffic light switching by traffic density measurement through

International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, 2018, pp. 1-6, doi: 10.1109/CONECCT.2018.8482380

[6] A. M. Ghoreyshi, A. AkhavanPour and A. Bossaghzadeh, "Simultaneous Vehicle Detection and Classification Model based on Deep YOLO Networks," *2020 International Conference on Machine Vision and Image Processing (MVIP)*, Iran, 2020, pp. 1-6, doi: 10.1109/MVIP49855.2020.9116922

[7] R. A. Asmara, B. Syahputro, D. Supriyanto and A. N. Handayani, "Prediction of Traffic Density Using YOLO Object Detection and Implemented in Raspberry Pi 3b + and Intel NCS 2," *2020 4th International Conference on Vocational Education and Training (ICOVET)*, Malang, Indonesia, 2020, pp. 391-395, doi: 10.1109/ICOVET50258.2020.9230145.

sensors," *2015 International Conference on Computers, Communications, and Systems (ICCCS)*, Kanyakumari, 2015, pp. 123-126, doi: 10.1109/CCOMS.2015.7562885.

[13] P. Soundarya Lahari, M. F. Mohammed, K. Lingaraju and K. Amulya, "Density Based Traffic Control with Emergency Override," *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, 2018, pp. 2094-2099, doi: 10.1109/RTEICT42901.2018.9012488.

[14] R. Nabati and H. Qi, "RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles," *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, pp. 3093-3097, doi: 10.1109/ICIP.2019.8803392.