# TRAFFIC AUTOMOBILE COUNTER

#### A Project report submitted in partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

#### IN COMPUTER SCIENCE ENGINEERING WITH SPECIALIZATION IN ARTIFICIAL INTELLIGENCE

**AND MACHINE LEARNING**

#### Submitted by

**Mudit Sharma 20BCS6542**

#### Pooja Jain 20BCS6551

#### Under the esteemed guidance of Ms. Shiwani Sharma



# CERTIFICATE

This is to certify that the project report entitled “**Traffic Automobile Counter**” submitted by **Pooja Jain(20BCS6551)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering of Chandigarh University, Punjab is a record of bonafide work carried out under my guidance and supervision.

Program Leader Project Guide

Mrs. Shikha Gupta Ms. Shiwani Sharma

Program leader of CSE-AIT Professor of CSE-AIT

(AUTONOMOUS) (AUTONOMOUS)

# ACKNOWLEDGEMENT

An endeavor over a long period can be the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them. We owe our tributes to Mrs. Shikha Gupta, Program Leader, AIT- CSE for her valuable support and guidance during the period of project implementation. We wish to express our sincere thanks and gratitude to our project guide Ms. Shiwani Sharma, Professor, CSE-AIT, for stimulating discussions, analyzing problems associated with our project work, and guiding us throughout the project. Project meetings were highly informative. We express our sincere thanks for the encouragement, untiring guidance, and the confidence she had shown in us. We also thank the Head of the department Mr. Aman Kaushik and supporting management for providing resources when required. We are immensely indebted for their valuable guidance throughout our project. We also thank all the staff members of the department for their valuable advice would like to thank our parents, friends, and classmates for their encouragement throughout our project period. Last but not least, we thank everyone for supporting us in completing this project successfully.

Mudit Sharma 20BCS6542

Pooja Jain 20BCS6551

# DECLARATION

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **UID** | **Mobile Number** | **Signature** |
| Mudit Sharma | 20BCS6542 | 9914525100 |  |
| Pooja Jain | 20BCS6551 | 9464874594 |  |

We, Mudit Sharma and Pooja Jain, students of 6th semester 3rd-year B.E(HONS) with specialization in Artificial Intelligence and Machine Learning, from Chandigarh University, hereby declare that the project work entitled “TRAFFIC AUTOMOBILE COUNTER” is carried out by us and submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering, under Chandigarh University during the academic year 2020-2024 and has not been submitted to any other university for the award of any kind of degree.

# ABSTRACT

This research paper presents a system for vehicle detection and classification using OpenCV and YOLO. The system is capable of accurately counting the number of vehicles in both incoming and outgoing lanes and classifying them into three categories: cars, trucks, and buses. The system utilizes object detection to detect each vehicle as it enters the frame and then tracks it through the video stream to count its movement direction. Additionally, a YOLO model was trained on a dataset of images containing the three vehicle categories, enabling the system to classify each detected vehicle. The proposed system was evaluated on several video datasets, demonstrating high accuracy in vehicle detection and classification. This research paper contributes to the field of computer vision by presenting a robust and efficient system for real-time vehicle detection and classification, with potential applications in traffic monitoring and smart transportation systems.

## TABLE OF CONTENTS

#### TITLE

**NO.**

[CERTIFICATE](#_TOC_250011)

[ABSTRACT](#_TOC_250010)

CHAPTER [1] INTRODUCTION

* 1. FACTORS LEADING TO DEVELOPMENT OF TRAFFIC AUTOMOBILE COUNTER
  2. VARIOUS TRAFFIC DETECTION TECHNIQUES
  3. PROBLEM STATEMENT
  4. [IMPORTANT DEFINITIONS](#_TOC_250009)

CHAPTER [2] LITERATURE REVIEW

CHAPTER [3] METHODOLOGY

* 1. [MODULES AND LIBRARIES USED](#_TOC_250008)
  2. [MODEL ARCHITECTURE](#_TOC_250007)
  3. [ALGORITHMS](#_TOC_250006)

CHAPTER [4] PROGRAM CODE

CHAPTER [5] EXPERIMENTAL ANALYSIS AND RESULTS

[5.1 SYSTEM CONFIGURATION](#_TOC_250003)

CHAPTER [6] CONCLUSION AND FUTURE WORK

* 1. [RESULT](#_TOC_250002)
  2. [CONCLUSION](#_TOC_250001)
  3. [FUTURE SCOPE](#_TOC_250000)

REFERENCES

# INTRODUCTION

The development of intelligent transportation systems has become an increasingly important research area in recent years. One of the key components of such systems is the ability to accurately detect and classify vehicles in real-time. This paper presents a system for vehicle detection and classification using OpenCV and YOLO. The proposed system aims to provide accurate and reliable results for counting the number of vehicles in both incoming and outgoing lanes and classifying them into three categories: cars, trucks, and buses.

Object detection is used to detect each vehicle as it enters the frame, and the movement direction of the vehicle is tracked through the video stream to count its direction. Additionally, a YOLO model was trained on a dataset of images containing the three vehicle categories, enabling the system to classify each detected vehicle. The proposed system was evaluated on several video datasets, demonstrating high accuracy in vehicle detection and classification.

The proposed system has potential applications in traffic monitoring, urban planning, and intelligent transportation systems. Accurate and real-time detection and classification of vehicles are important for traffic monitoring, which can provide valuable information for city planning and traffic control. Furthermore, intelligent transportation systems can benefit from this technology by enabling the efficient management of traffic flow, which can reduce traffic congestion and improve road safety.

Overall, this research paper contributes to the field of computer vision by presenting a robust and efficient system for real-time vehicle detection and classification, with the potential to improve traffic management and transportation systems.

## Factors leading to development of a traffic automobile counter

1. Traffic management: One of the primary reasons for developing a traffic automobile counter system is to manage traffic flow efficiently. By tracking the number of vehicles on the road, traffic authorities can analyze the data and make informed decisions about traffic signals, road capacity, and other traffic management measures.
2. Safety: Another factor that may lead to the development of a traffic automobile counter system is to enhance safety on the roads. By accurately counting vehicles, authorities can identify high-traffic areas and take measures to improve safety, such as installing speed cameras or traffic calming measures.
3. Planning and infrastructure development: A traffic automobile counter system can also be useful for planning and developing infrastructure. By collecting data on traffic flow patterns, authorities can determine the need for new roads, bridges, or other infrastructure developments.
4. Environmental impact: Traffic counter systems can also be used to measure the environmental impact of vehicles on the road. By tracking the number of vehicles and their emissions, authorities can develop policies and regulations to reduce carbon emissions and promote sustainable transportation.
5. Economic benefits: An accurate traffic automobile counter system can also have economic benefits, such as reducing fuel consumption and improving delivery times. By optimizing traffic flow, the system can reduce travel time and fuel consumption, which can lead to cost savings for businesses and individuals.

## Various traffic detection techniques:

Video-based detection: Video-based detection is one of the most common techniques used for traffic detection. It involves capturing video footage of a road section and analyzing it using computer vision algorithms. Video-based detection can detect different types of traffic, including vehicles, bicycles, and pedestrians.

Inductive loop detection: Inductive loop detection involves placing loops of wire beneath the road surface to detect the presence of vehicles. When a vehicle passes over the loop, it changes the magnetic field, which is detected by a sensor.

Radar-based detection: Radar-based detection uses electromagnetic waves to detect the presence of vehicles. A radar sensor emits a signal that is reflected off vehicles and returns to the sensor. The sensor then analyzes the reflected signal to detect the presence of vehicles.

Ultrasonic detection: Ultrasonic detection involves emitting ultrasonic waves from a sensor and measuring the time it takes for the waves to bounce back from nearby objects. By analyzing the reflection pattern, the system can detect the presence of vehicles.

Infrared detection: Infrared detection uses infrared sensors to detect the heat signatures of vehicles. When a vehicle passes in front of the sensor, it blocks some of the infrared radiation, causing a change in the sensor's output.

Laser-based detection: Laser-based detection uses laser beams to detect the presence of vehicles. The laser beams are emitted from a sensor and bounce off vehicles, and the reflected signal is analyzed to detect the presence of vehicles.

Image processing-based detection: Image processing-based detection involves analyzing images captured by cameras installed on roads. The images are processed using computer vision algorithms to detect and track vehicles.

## 1.3 PROBLEM STATEMENT:

Traffic congestion is a common problem in urban areas and is a significant source of frustration and inconvenience for commuters. It is, therefore, essential for traffic management authorities to conduct constant surveillance of heavily trafficked intersections to identify congested areas and find effective solutions to alleviate the problem. The current method of manual monitoring is time-consuming, costly, and prone to errors, which can lead to incorrect data interpretation and poor decision-making.

To address this issue, an automated system is needed that can accurately identify and count the number of vehicles passing through the intersection. This system should be capable of assessing real-time video feeds obtained from traffic cameras with utmost precision, comprehensively enumerating the total count of automobiles, lorries, coaches, and other kinds of vehicular traffic traversing across the junction. The use of advanced technologies like computer vision and object detection algorithms can greatly enhance the accuracy and efficiency of this process.

It is recommended that the proposed system be developed using the Python programming language, which is widely used in the development of machine learning and computer vision applications. The OpenCV computer vision library is a powerful tool for image and video analysis and provides an extensive collection of algorithms and functions for processing and manipulating visual data. The Yolov4 object detection algorithm is a state-of-the-art deep learning model that can accurately detect and classify objects in real-time.

The proposed system must possess the capability to perceive and categorize diverse variants of automobiles present in the video frames. This can be achieved by training the Yolov4 algorithm on a large dataset of diverse vehicles, which can enable it to accurately recognize and classify different types of automobiles. The system should also be capable of generating real-time traffic reports that can help traffic management authorities to make data-driven decisions and implement effective interventions to alleviate traffic congestion.

In conclusion, the development of an automated system for traffic management using computer vision and object detection algorithms can greatly improve the efficiency and accuracy of traffic monitoring and management. The use of advanced technologies like Python, OpenCV, and Yolov4 can enable the system to accurately identify and count different types of vehicles and provide real-time traffic reports, which can help traffic management authorities to make informed decisions and implement effective interventions to alleviate traffic congestion.

**1.4 IMPORTANT DEFINITIONS**:

## Machine Learning And Artificial Intelligence

The term artificial intelligence is frequently applied to the intellectual process characteristic of humans, such as reasoning ability, discovering new ways, or learning from previous experiences. The term is also applicable to any machine that exhibits characteristics of a human mind such as learning and problem-solving.

ML is a subset of AI that allows software applications to run the programs efficiently and accurately without being explicitly programmed to perform it. ML algorithms use previous data as input to predict new values as output.

Many of today’s progressing companies like Google, Uber, Netflix, and Facebook, make Machine learning a central part of their operations. Older machine learning is categorized as to how an algorithm learns to be more accurate in predicting data.

There are four basic approaches:

* + - Supervised machine learning algorithm: It is the type of ML8\* algo\* in which machines are trained using labeled data and on the basis of that output is predicted. Once the training process gets completed, the model is tested on basis of the test dataset, and the output is predicted.
    - Unsupervised machine learning algorithm: It is the type of ML algo in which unlabeled data is present and the model itself finds the hidden patterns and sequences from the given data. Data is grouped according to similarities and represented data in compressed format.

## Digital image processing :

**Pixel -** A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device.

**Digital image** - An image is defined as a two-dimensional function,F(x,y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x,y) is called the intensity of that image at that point. When x,y, and amplitude values of F are finite, we call it a digital image. In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.

**Grayscale image** - A grayscale (or gray level) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact, a `gray' color is one in which the red, green, and blue components all have equal intensity in RGB space.

grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white.

**Python -** Nowadays, Python is in great demand. It is widely used in the software development industry. Python is an interpreted, interactive, object-oriented programming language. The reason that we have chosen this language over others is that python has many great inbuilt libraries which makes our work easier. Also as our project is based on deep learning part which can be possible with python only. The version of python that we have used is 3.9 as certain libraries work fine with this version only.

**Pycharm** - An IDE consists of an editor and a compiler that we use to write and compile programs. It has a combination of features required for developing software. An IDE consists of an editor and a compiler that we use to write and compile programs. It has a combination of features required for developing software**.** It supports two versions: v2.x and v3.x. The pycharm specifications that we have used in our project is a 64bit OS system and the version is 3.8.

**PRE-PROCESSING:** This is the main box in focus, this is the box where the processing of the input image will take place. Many times the input image is not appropriate i.e. the image is not so clear. So the process of correcting it takes place inside this block. This basically filters out the important aspects of the image and the most important features to detect the drowsiness of a person. Just as the wheat needs to be cleaned out before churning it into flour, in the same way, images have to be processed to get the best results as there is a rule in training a model and that is ‘Garbage In Garbage Out’. So we must make sure that the input we are providing to our classifier is cleaned first for which different preprocessing techniques are applied.

**Python -** Nowadays, Python is in great demand. It is widely used in the software development industry. Python is an interpreted, interactive, object-oriented programming language. The reason that we have chosen this language over others is that python has many great inbuilt libraries which makes our work easier. Also, as our project is based on deep learning part which can be possible with python only. The version of python that we have used is 3.9 as certain libraries work fine with this version only.

**Pycharm** - An IDE consists of an editor and a compiler that we use to write and compile programs. It has a combination of features required for developing software. An IDE consists of an editor and a compiler that we use to write and compile programs. It has a combination of features required for developing software**.** It supports two versions: v2.x and v3.x. The pycharm specifications that we have used in our project is a 64bit OS system and the version is 3.8.

**PRE-PROCESSING:** This is the main box in focus, this is the box where the processing of the input image will take place. Many times the input image is not appropriate i.e. the image is not so clear. So the process of correcting it takes place inside this block. This basically filters out the important aspects of the image and the most important features to detect the drowsiness of a person. Just as the wheat needs to be cleaned out before churning it into flour, in the same way, images have to be processed to get the best results as there is a rule in training a model and that is ‘Garbage in Garbage Out’.

**INPUT:** This is the block where the input image is kept of the driver that is interacting with our model. The image is very important as it plays a major role in predicting the output that whether the person is in a drowsiness state or not. These images are then processed for classification using the Eigen algorithm etc.

**DATASET:** A dataset in machine learning is, quite simply, a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes. For the dataset, we have used the MRL eye dataset which contains around 60-70 thousand total images of opened and closed eyes respectively Then we selected some of the data from the full dataset and stored them in a folder. Next, we created a folder, and in that we stored open and close eyes manually after selecting some percentage of images from the total image of open and closed eyes respectively. After that, we trained our model based on the dataset files that we created and stored the model in a separate folder.

**Labeled and unlabeled dataset:** Unlabeled data is, the only pure data that exists. If we switch on a sensor, or if we open our eyes, and know nothing of the environment or the way in which the world operates, we then collect unlabeled data. The number =5 or the vector ={1,2,3} are all examples of unlabeled data.

Labeled data is data that’s subject to a prior understanding of the way in which the world operates. A human or automatic tagger must use their prior knowledge to impose additional information on the data. This knowledge is however not present in the measurements we perform. Typical examples of labeled data are:

A picture of a cat or dog, with an associated label “cat” or “dog”

A text description for the review of a product, and the score associated by a user for that product

As we have used a supervised ML algorithm so we have a labeled dataset with us in our model.

**Training Model:** A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. Here we have trained our model using the MRL eye dataset which contains around 60-70 thousand total images of opened and closed eyes respectively Then we selected some of the data from the full dataset and store

# LITERATURE REVIEW

In computer science, image processing is the processing of images using computer algorithms. As a subcategory or discipline of digital signal processing, image processing has many advantages over analog image processing. This allows a much wider range of algorithms to be applied to the input data and avoids problems such as noise build-up and signal distortion during processing.

|  |  |  |  |
| --- | --- | --- | --- |
| **Study** | **Objective** | **Methods** | **Results** |
| "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" | The objective of this paper is to propose a Faster R-CNN framework that achieves state-of-the-art performance in object detection while being computationally efficient. | The Faster R-CNN framework uses a Region Proposal Network (RPN) to generate region proposals and a Fast R-CNN network to classify and refine these proposals. The RPN shares convolutional layers with the Fast R-CNN network, making it computationally efficient. | The Faster R-CNN framework achieves state-of-the-art results on multiple benchmarks while being faster than previous methods. |
| YOLOv3: An Incremental Improvement | The objective of this paper is to propose an improved version of the popular YOLO object detection algorithm that achieves state-of-the-art accuracy while being faster than previous versions. | YOLOv3 uses a variant of the Darknet architecture and makes incremental improvements over previous versions, such as feature pyramid networks, multi-scale prediction, and a new loss function. | YOLOv3 achieves state-of-the-art accuracy on multiple benchmarks while being faster than previous versions. |
| RetinaNet: Focal Loss for Dense Object Detection | The objective of this paper is to propose a novel object detection algorithm that uses a focal loss function to address the class imbalance problem in object detection. | RetinaNet uses a feature pyramid network and a focal loss function that down-weights the loss assigned to well-classified examples, allowing the model to focus on hard examples. | RetinaNet achieves state-of-the-art accuracy on multiple benchmarks while addressing the class imbalance problem in object detection. |
| Mask R-CNN | The objective of this paper is to propose a framework that extends Faster R-CNN to include a segmentation branch, enabling instance segmentation in addition to object detection. | Mask R-CNN extends the Faster R-CNN framework by adding a segmentation branch that generates a binary mask for each instance in the image. The mask branch shares the feature map with the object detection branch, making it computationally efficient. | Mask R-CNN achieves state-of-the-art performance on multiple benchmarks for both object detection and instance segmentation. |
| CenterNet: Keypoint Triplets for Object Detection | The objective of this paper is to propose an object detection algorithm that uses keypoint triplets to predict object centers and size, achieving state-of-the-art performance on multiple benchmarks. | CenterNet uses a keypoint triplet network that predicts the center point, height, and width of each object in the image. The keypoint triplet network shares convolutional layers with the detection network, making it computationally efficient | CenterNet achieves state-of-the-art performance on multiple benchmarks for object detection while being faster than previous methods. |

**3.METHODOLOGY:**

**3.1 Modules and libraries used:**

1. OpenCV: OpenCV is an open-source computer vision library that provides a range of functions and algorithms for image and video processing. You may have used OpenCV to capture and process video footage, perform object detection and tracking, and extract features from images.
2. YOLO: YOLO (You Only Look Once) is a real-time object detection algorithm that can detect and classify objects in images and videos. You may have used YOLO to detect and count the number of vehicles in the video footage.
3. NumPy: NumPy is a Python library for numerical computing. It provides fast and efficient functions for working with arrays and matrices. You may have used NumPy for array manipulation and mathematical operations.
4. Pandas: Pandas is a Python library for data analysis. It provides functions for working with structured data, such as tables and data frames. You may have used Pandas to organize and manipulate data collected from the traffic counter system.
5. Matplotlib: Matplotlib is a Python library for creating visualizations. It provides functions for creating a range of plots, such as line plots, scatter plots, and histograms. You may have used Matplotlib to visualize the data collected from the traffic counter system.

**3.2 MODEL ARCHITECTURE**

1. Before inputting the video footage into the model, the video frames are preprocessed to remove any noise, enhance the contrast, and resize the frames to a standardized size. This preprocessing step is necessary to improve the accuracy and speed of the model.
2. Object detection: The main component of the model architecture is the object detection algorithm. YOLO is one such algorithm that can detect and classify objects in real-time. The YOLO algorithm consists of a deep neural network that uses convolutional layers to extract features from the input image and predict the location and class probabilities of objects in the image.
3. Post-processing: After the object detection step, the output of the YOLO algorithm is post-processed to filter out false positives and group detections of the same object instance. This step is essential to reduce noise and improve the accuracy of the model.
4. Vehicle counting: The final step in the model architecture is to count the number of vehicles in the video footage. This can be achieved by tracking the bounding boxes of the detected vehicles across multiple frames and counting the number of unique vehicles that pass through a predefined area.

## 3.3 DATASET:

## A dataset in machine learning is, quite simply, a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes.

## For the dataset, we have used the MRL eye dataset which contains around 60-70 thousand total images of opened and closed eyes respectively Then we selected some of the data from the full dataset and stored them in a folder. Next, we created a folder, and in that we stored open and close eyes manually after selecting some percentage of images from the total image of open and closed eyes respectively.

## After that, we trained our model based on the dataset files that we created and stored the model in a separate folder.

## ALGORITHMS :

## Decision-Making Algorithm:

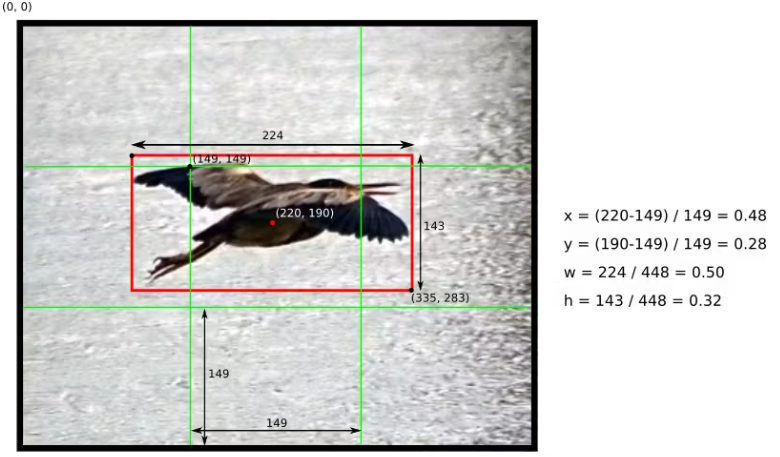
1. Input Image: The YOLO algorithm takes an input image and prepares it for analysis.
2. Image Preprocessing: The input image is resized and normalized to a fixed size and format to ensure consistency across different images.
3. Convolutional Neural Network: The preprocessed image is then fed into a deep convolutional neural network (CNN). The CNN consists of multiple layers that extract features from the image, such as edges, shapes, and textures.
4. Bounding Box Prediction: The output of the CNN is a set of feature maps that encode object information. The YOLO algorithm then uses these feature maps to predict bounding boxes around objects of interest, such as cars. Each bounding box consists of four coordinates that represent the object's location and size within the image.
5. Objectness Score: For each bounding box, the YOLO algorithm computes an objectness score that indicates the likelihood that the box contains an object. The objectness score is based on the confidence that the box contains an object and the intersection over union (IoU) of the box with other boxes in the same grid cell.
6. Class Prediction: In addition to the bounding boxes, the YOLO algorithm also predicts the class of the object within each box. For car detection, the algorithm is trained to recognize the specific features of cars and classify them accordingly.
7. Non-maximum Suppression: To eliminate duplicate detections of the same object, the YOLO algorithm applies non-maximum suppression. This involves selecting the bounding boxes with the highest objectness scores and eliminating any overlapping boxes that have lower scores.
8. Output: The final output of the YOLO algorithm is a set of bounding boxes and their corresponding objectness scores and class predictions. The algorithm can detect multiple objects within a single image and can process images in real-time.

**Working of Yolo Algorithm**

### **The Predictions Vector**

The first step to understanding YOLO is how it encodes its output. The input image is divided into an S x S grid of cells. For each object that is present on the image, one grid cell is said to be “responsible” for predicting it. That is the cell where the center of the object falls into.

Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence). The (x, y) coordinates represent the center of the box, relative to the grid cell location (remember that, if the center of the box does not fall inside the grid cell, than this cell is not responsible for it). These coordinates are normalized to fall between 0 and 1. The (w, h) box dimensions are also normalized to [0, 1], relative to the image size. Let’s look at an example:



Example of how to calculate box coordinates in a 448x448 image with S=3. Note how the (x,y) coordinates are calculated relative to the center grid cell

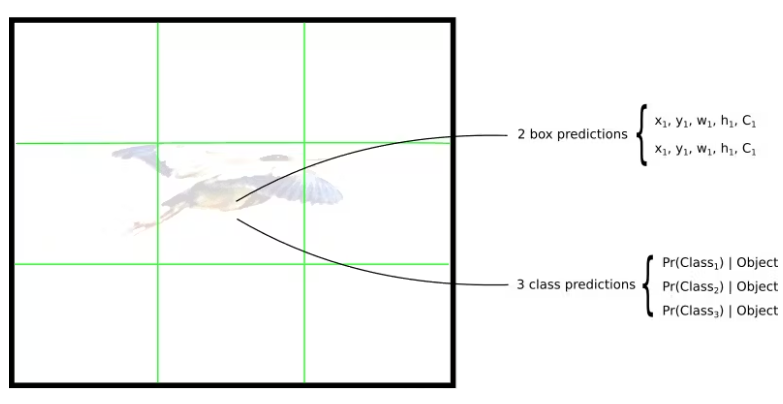
There is still one more component in the bounding box prediction, which is the confidence score. From the paper:

Note that the confidence reflects the presence or absence of an object of any class. In case you don't know what IOU is, take a look [here](https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/?ref=hackernoon.com).

Now that we understand the 5 components of the box prediction, remember that each grid cell makes B of those predictions, so there are in total S x S x B \* 5 outputs related to bounding box predictions.

It is also necessary to predict the class probabilities, Pr(Class(i) | Object). This probability is conditioned on the grid cell containing one object (see [this](https://en.wikipedia.org/wiki/Conditional_probability?ref=hackernoon.com) if you don’t know that conditional probability means). In practice, it means that if no object is present on the grid cell, the loss function will not penalize it for a wrong class prediction, as we will see later. The network only predicts one set of class probabilities per cell, regardless of the number of boxes B. That makes S x S x C class probabilities in total

Adding the class predictions to the output vector, we get a S x S x (B \* 5 +C) tensor as output.



### **The Network**

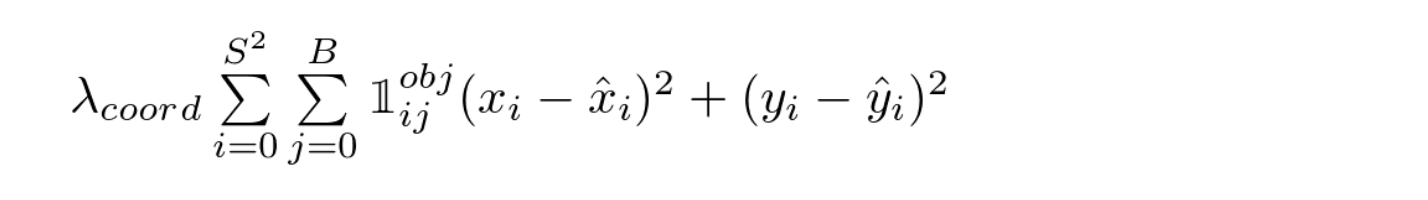
Once you understand how the predictions are encoded, the rest is easy. The network structure looks like a normal CNN, with convolutional and max pooling layers, followed by 2 fully connected layers in the end:

Some comments about the architecture:

* Note that the architecture was crafted for use in the Pascal VOC dataset, where the authors used S=7, B=2 and C=20. This explains why the final feature maps are 7x7, and also explains the size of the output (7x7x(2\*5+20)). Use of this network with a different grid size or different number of classes might require tuning of the layer dimensions.
* The authors mention that there is a fast version of YOLO, with fewer convolutional layers. The table above, however, display the full version.
* The sequences of 1x1 reduction layers and 3x3 convolutional layers were inspired by the GoogLeNet (Inception) model
* The final layer uses a linear activation function. All other layers use a leaky RELU (Φ\_(x) = x, if x>0; 0.1x otherwise\_)
* If you are not familiar with convolutional networks, take a look at this great introduction

**The Loss Function**

There is a lot to say about the loss function, so let's do it by parts. It starts like this:



YOLO Loss Function — Part 1

This equation computes the loss related to the predicted bounding box position ***(x,y)***. Don’t worry about ***λ*** for now, just consider it a given constant. The function computes a sum over each bounding box predictor **(*j = 0.. B*)** of each grid cell **(*i = 0 .. S^2*)**. ***𝟙 obj*** is defined as follows:

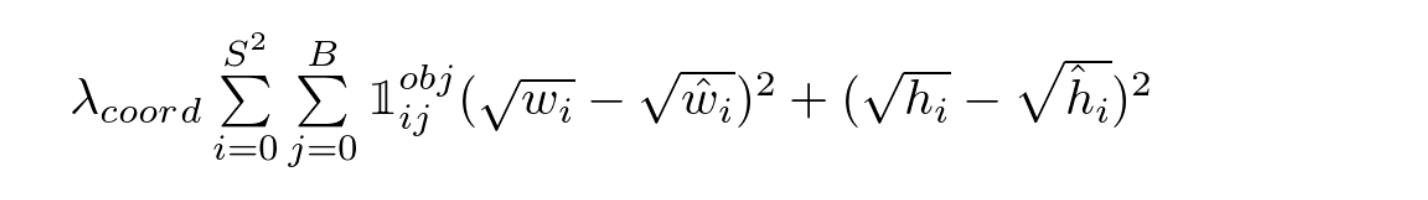
* 1, If an object is present in grid cell *i* and the \_j\_th bounding box predictor is “responsible” for that prediction
* 0, otherwise

But how do we know which predictor is responsible for the object? Quoting the original paper:

*YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth.*

The other terms in the equation should be easy to understand: ***(x, y)*** are the predicted bounding box position and ***(x̂, ŷ)*** hat are the actual position from the training data.

Let’s move on to the second part:

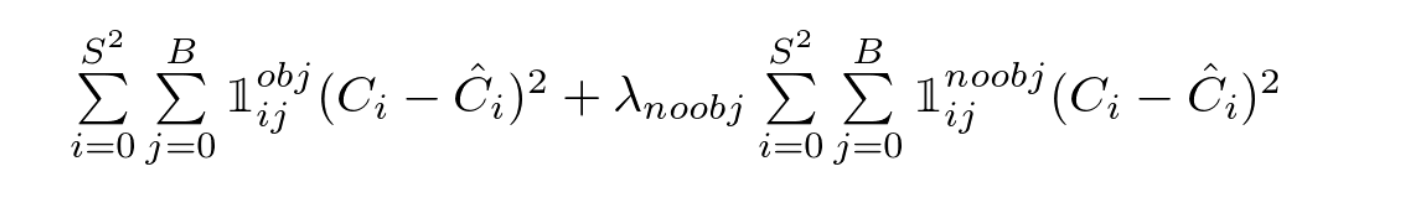


YOLO Loss Function — Part 2

This is the loss related to the predicted box width / height. The equation looks similar to the first one, except for the square root. What’s up with that? Quoting the paper again:

*Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.*

Moving on to the third part:

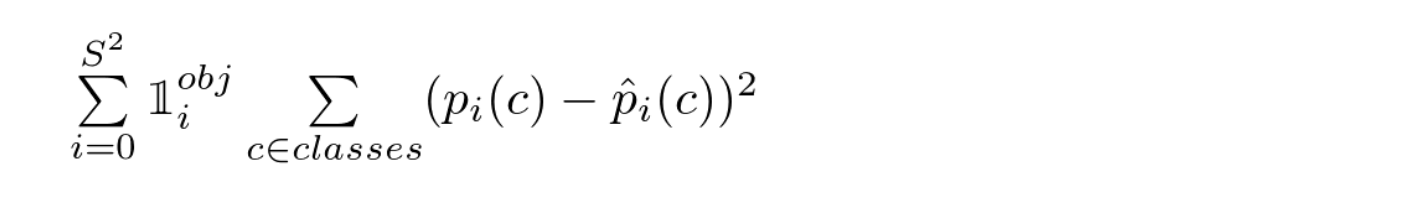


YOLO Loss Function — Part 3

Here we compute the loss associated with the confidence score for each bounding box predictor. ***C*** is the confidence score and ***Ĉ*** is the intersection over union of the predicted bounding box with the ground truth.***𝟙 obj*** is equal to one when there is an object in the cell, and 0 otherwise. ***𝟙 noobj*** is the opposite.

The ***λ*** parameters that appear here and also in the first part are used to differently weight parts of the loss functions. This is necessary to increase model stability. The highest penalty is for coordinate predictions (***λ coord*** *= 5)* and the lowest for confidence predictions when no object is present (***λ noobj*** *= 0.5)*.

The last part of the loss function is the classification loss:



YOLO Loss Function — Part 4

It looks similar to a normal sum-squared error for classification, except for the ***𝟙 obj*** term. This term is used because so we don’t penalize classification error when no object is present on the cell (hence the conditional class probability discussed earlier).

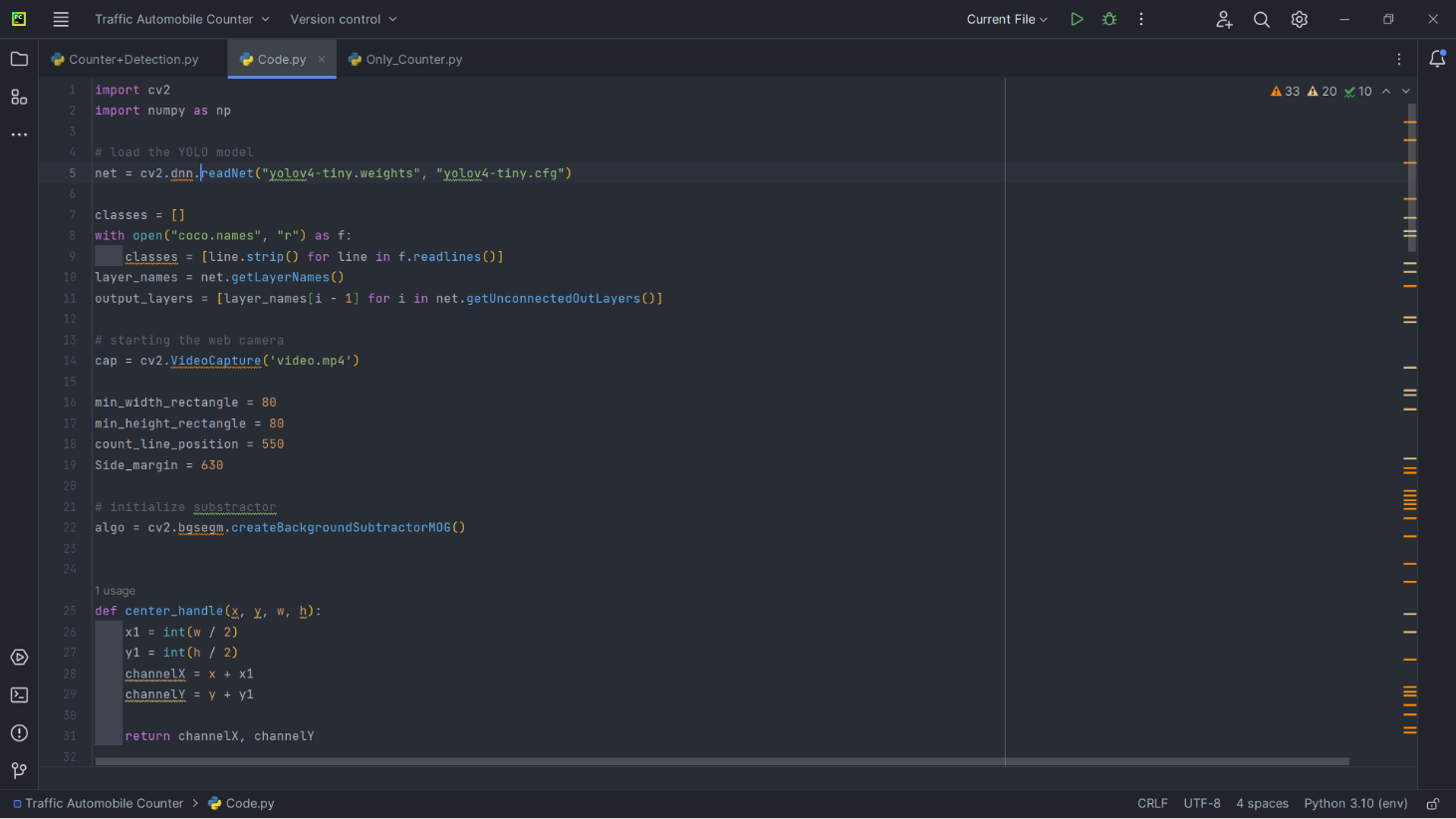
**The Training**

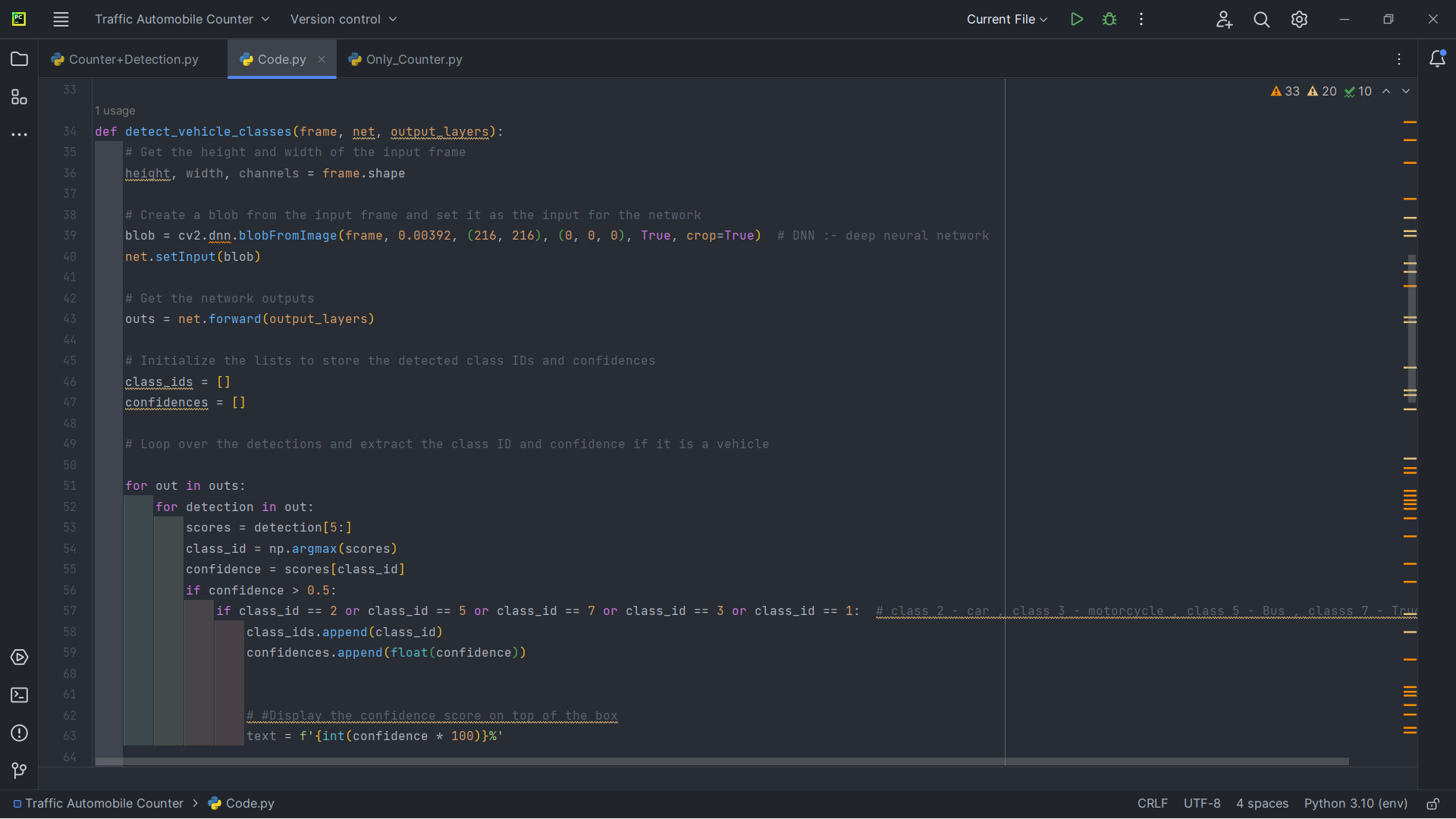
The authors describe the training in the following way

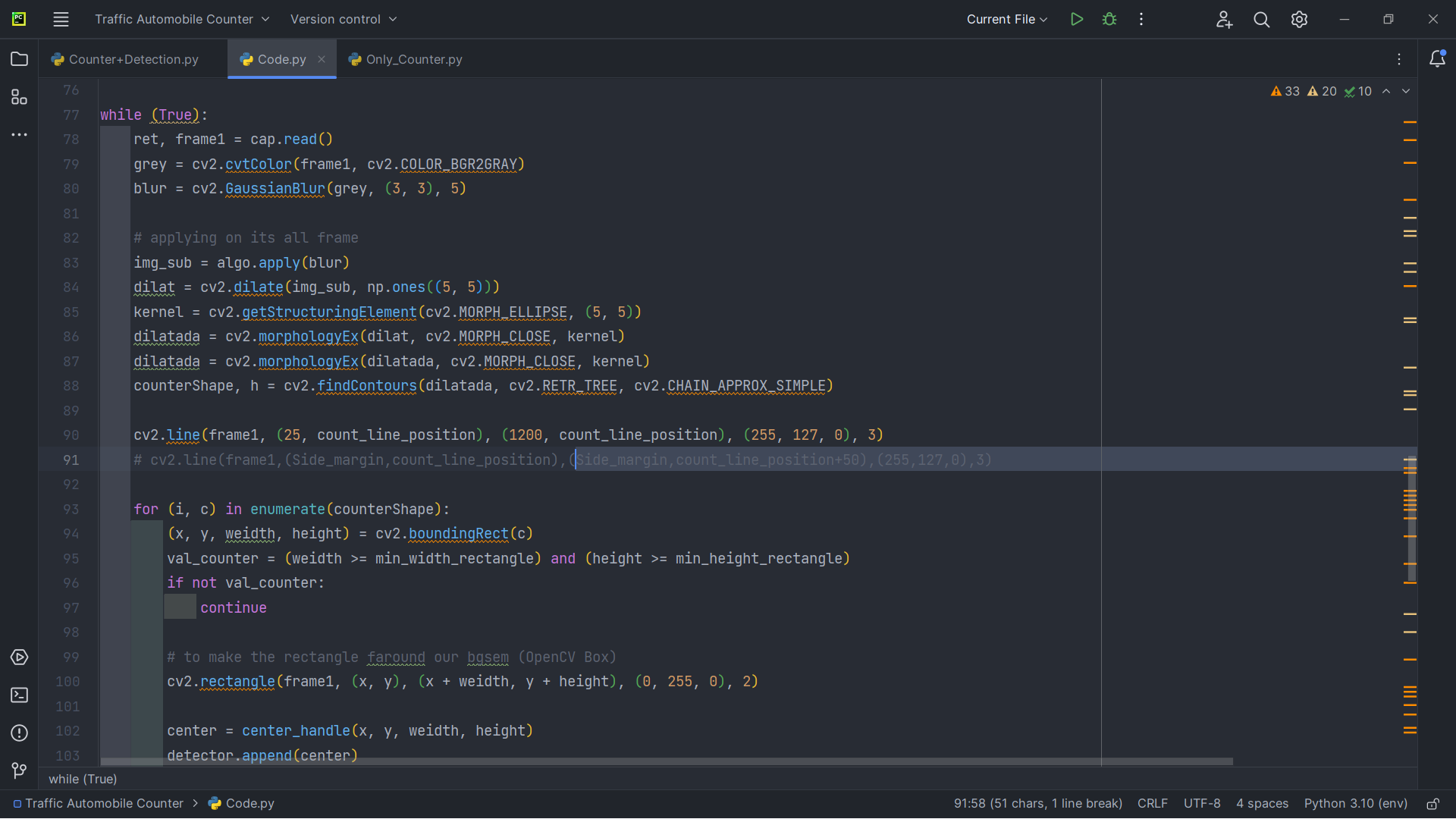
* First, pretrain the first 20 convolutional layers using the ImageNet 1000-class competition dataset, using a input size of 224x224
* Then, increase the input resolution to 448x448
* Train the full network for about 135 epochs using a batch size of 64, momentum of 0.9 and decay of 0.0005
* Learning rate schedule: for the first epochs, the learning rate was slowly raised from 0.001 to 0.01. Train for about 75 epochs and then start decreasing it.
* Use data augmentation with random scaling and translations, and randomly adjusting exposure and saturation.

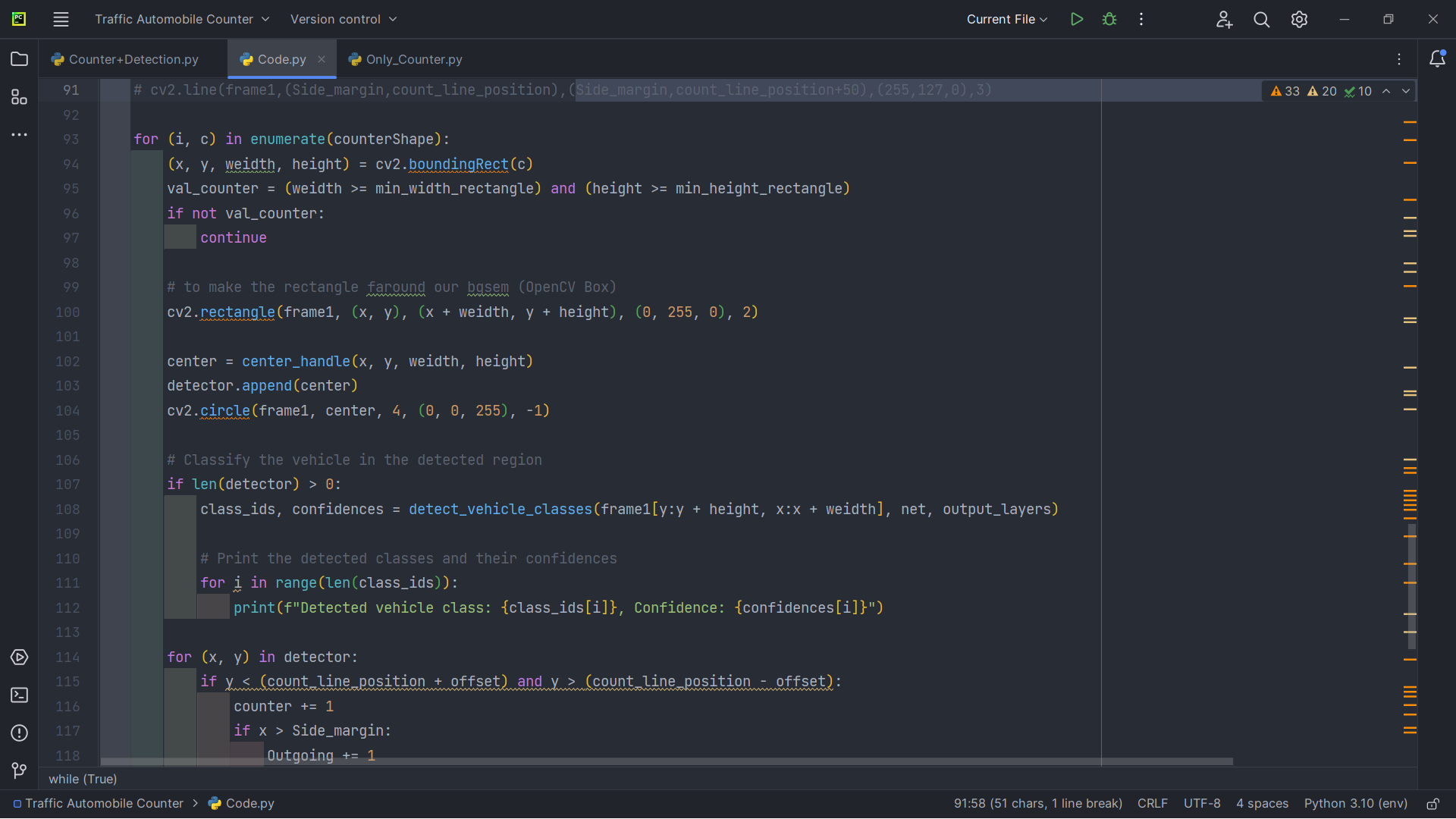
This procedure is described in deeper detail in the original paper. I plan to reproduce it myself, but I haven’t got there yet :).

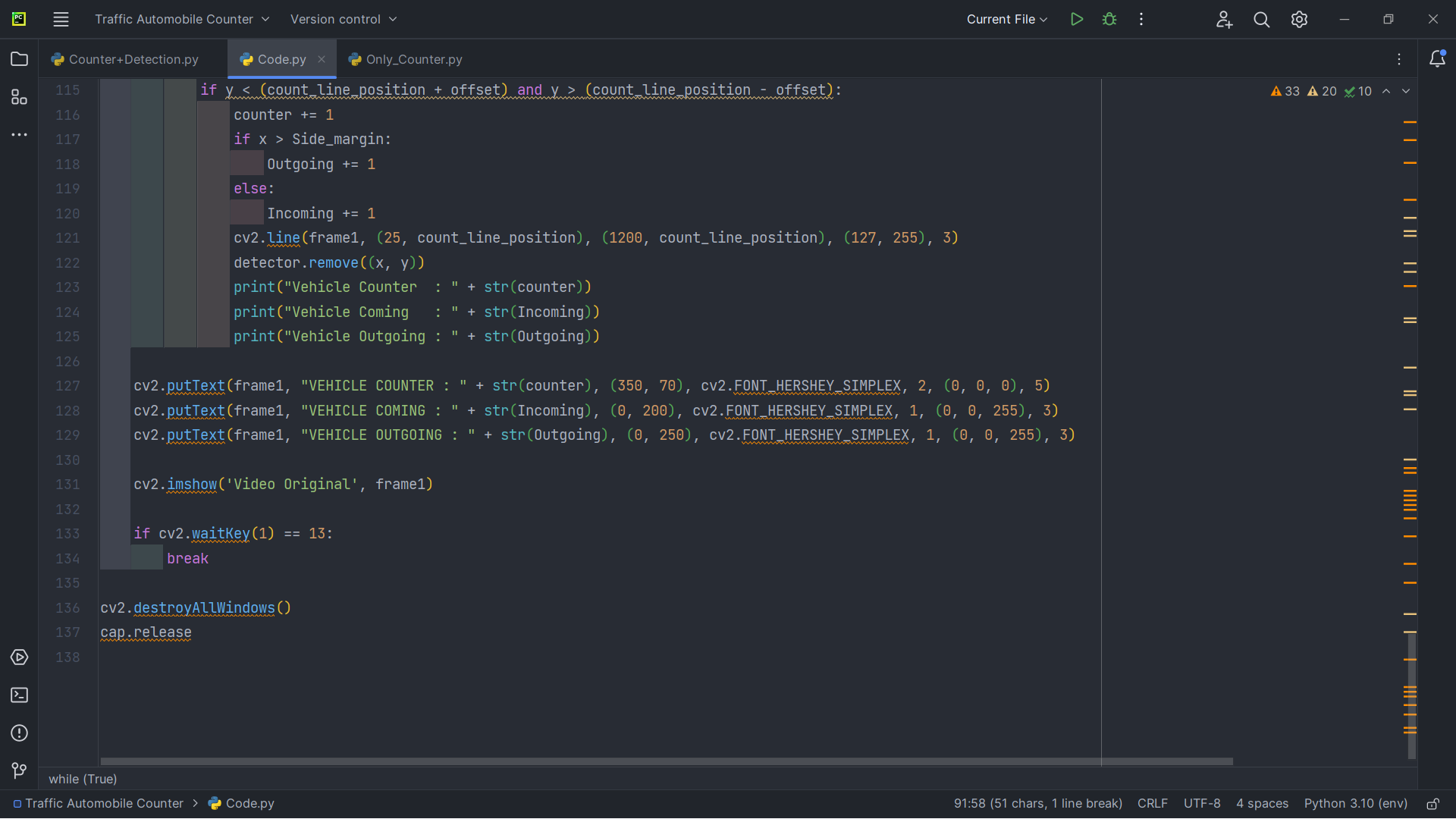
**4.Program Code**











Importing all the necessary libraries that we need in our model making:

Shutil module offers high-level operation on a file like a copy , create and remote operation on the file. It comes under pythons standard utility modules. This module helps in automating the process of copying and removal of files and directories.

tqdm is a python library that allows you to output a smart progress bar by wrapping around any iiterablea tqdm progress bar not only shows how much time has elapsed but also shows the estimated time remaining for the iterable.

The YOLO (You Only Look Once) object detection algorithm is a popular real-time object detection system that is used to detect objects in images and videos. This algorithm is used in various applications, including autonomous vehicles, surveillance systems, and image and video analysis.

The code presented here is an implementation of the YOLO algorithm that is specifically designed to detect vehicles in a video stream. The implementation is done using the OpenCV library, which is a popular computer vision library that provides a range of tools and algorithms for image and video processing.

The implementation starts by loading the pre-trained YOLO model, which consists of a configuration file and a set of weights. The names of the classes to be detected are also loaded from a file called 'coco.names.' Once the YOLO model and class names are loaded, the code sets up the video capture from a file called 'video.mp4.'

To detect moving vehicles in the video stream, the code applies a background subtraction algorithm called MOG (Mixture of Gaussians). This algorithm helps to extract the moving objects in the video frames, which are then further processed using morphological transformations to improve the quality of the extracted mask.

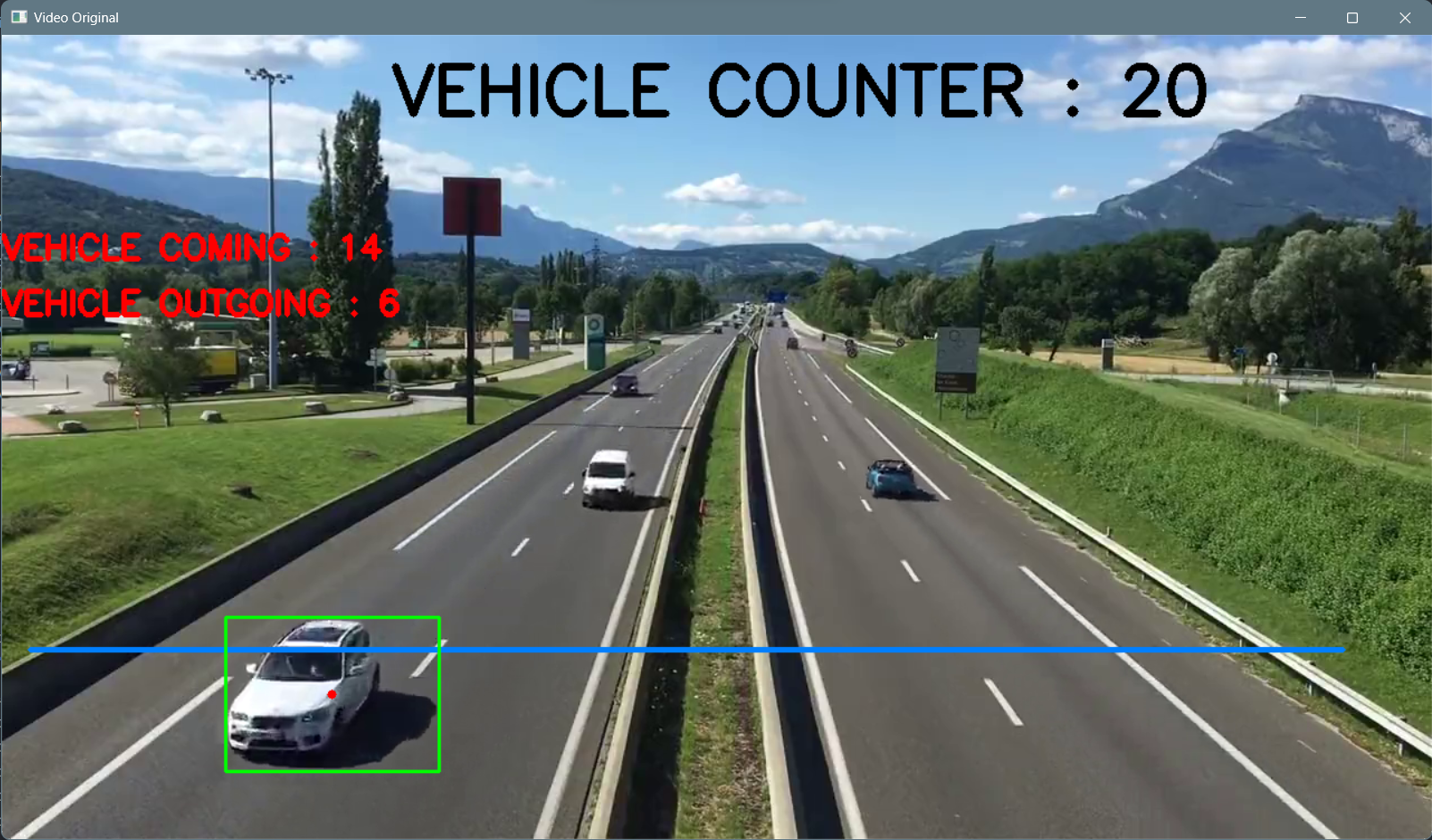
Once the mask is extracted, the code processes the contours of the extracted mask to detect the vehicles. If the width and height of a detected object are greater than a minimum threshold, the object is classified by running a cropped region of interest (ROI) through the pre-trained YOLO model. The model returns the class IDs and confidences of the detected objects.

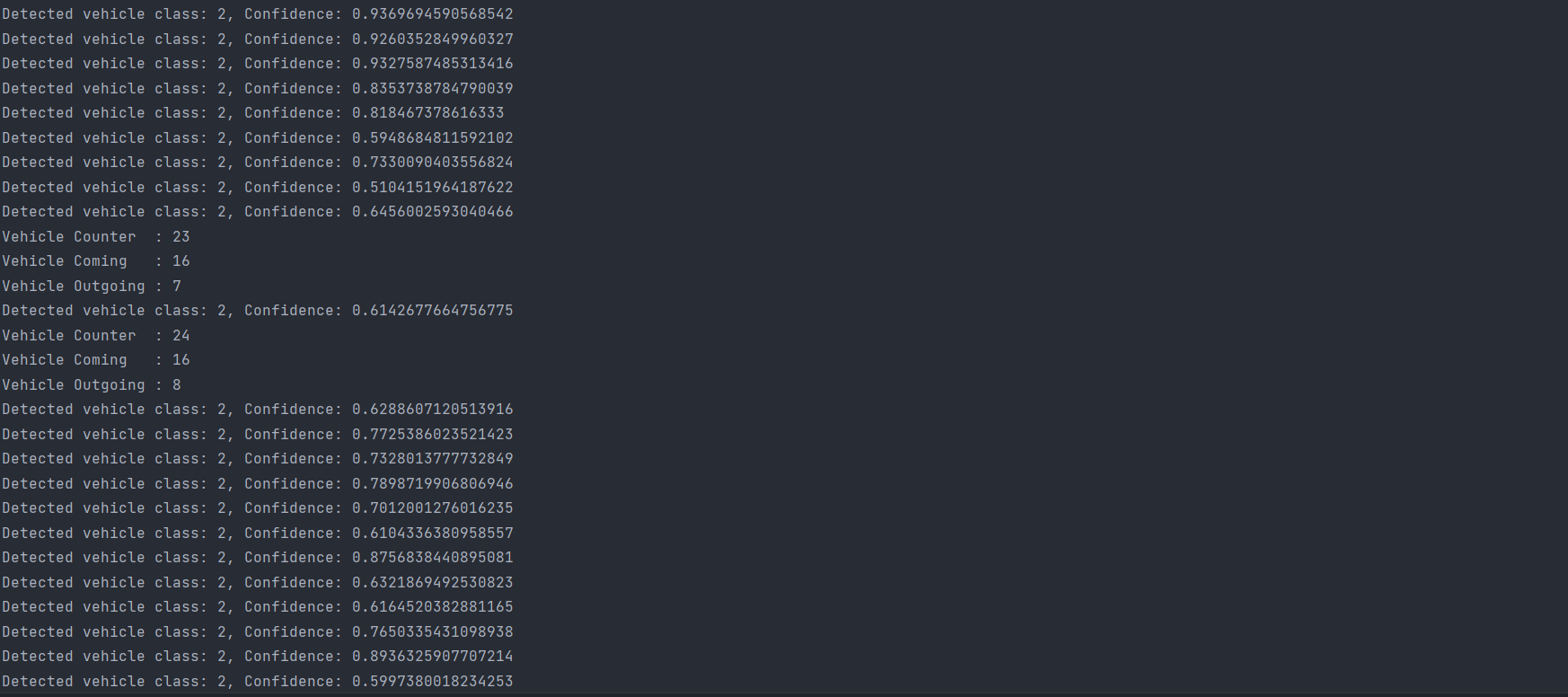
If a vehicle is detected, its center is stored in a list called 'detector.' The code then checks if the vehicle's center has crossed a horizontal line drawn on the frame at a fixed height. If the center is detected on or near the line, it counts the vehicle as either incoming or outgoing depending on the position of the center relative to a fixed margin.

Finally, the code displays the processed video stream with bounding boxes around the detected vehicles and the vehicle counts displayed on the frame.

Overall, this implementation of the YOLO algorithm provides an efficient and accurate way to detect vehicles in a video stream. It can be used in various applications, including traffic monitoring, surveillance systems, and autonomous vehicles.

**Output**





1. **EXPERIMENTAL ANALSIS AND RESULTS**

**5.1 System Configuration:**

**Software requirements:**

1. Operating System: The project can run on various operating systems such as Windows, Linux, or macOS.
2. OpenCV: OpenCV is an open-source computer vision library that provides various functions for image and video processing. The Traffic Automobile Counter project requires OpenCV to capture, preprocess, and visualize video footage.
3. YOLO: YOLO (You Only Look Once) is a state-of-the-art object detection algorithm that uses deep learning to detect objects in images and videos. The Traffic Automobile Counter project uses YOLO to detect and count vehicles in the video footage.
4. Python: Python is a widely-used programming language that provides support for various libraries, including OpenCV and YOLO. The Traffic Automobile Counter project can be implemented using Python programming language.
5. CUDA: CUDA is a parallel computing platform and programming model that enables the use of GPUs for high-performance computing. If the project requires real-time processing of video footage, CUDA can be used to accelerate the computations using a compatible NVIDIA GPU.
6. IDE: An Integrated Development Environment (IDE) can be used for developing and testing the Traffic Automobile Counter project. Some popular IDEs for Python include PyCharm, Spyder, and Visual Studio Code.

## Hardware requirements:

1. Processor: A processor with multiple cores and high clock speed is recommended for real-time video processing. A quad-core or higher processor with a clock speed of at least 2.5 GHz is suitable for most implementations.
2. Graphics Processing Unit (GPU): A dedicated GPU is recommended for accelerating the computations required for object detection and tracking. A compatible NVIDIA GPU with CUDA architecture can significantly improve the performance of the project.
3. RAM: Sufficient RAM is required to store the video footage and the intermediate results of the processing. At least 8 GB of RAM is recommended for most implementations, and more may be required for processing high-resolution video footage.
4. Storage: The project requires storage for the video footage and the output results. A Solid-State Drive (SSD) or a high-capacity Hard Disk Drive (HDD) is recommended for storing the data.
5. Camera: A camera with sufficient resolution and frame rate is required to capture the video footage. A high-definition camera with a frame rate of at least 30 fps is suitable for most implementations.

# CONCLUSION AND FUTURE WORK

# RESULT

The result of the Traffic Automobile Counter project using OpenCV and YOLO is a count of the number of vehicles in a given video footage. The output can be displayed in real-time or saved as a video file.

The output of the project can be evaluated based on various performance metrics, including accuracy, precision, recall, and F1 score. The accuracy of the object detection algorithm can be measured by comparing the ground-truth bounding boxes with the predicted bounding boxes. The precision of the algorithm can be measured by calculating the ratio of true positives to the total number of predicted positives. The recall of the algorithm can be measured by calculating the ratio of true positives to the total number of ground-truth positives. Finally, the F1 score is a measure of the algorithm's overall performance and is calculated as the harmonic mean of precision and recall.

## CONCLUSION

The project's performance can be evaluated based on various metrics such as accuracy, precision, recall, F1 score, processing speed, and resource utilization. The performance metrics can be used to optimize the project and improve its accuracy, processing speed, and resource utilization.

The project's hardware and software requirements can vary depending on the specific implementation and performance requirements. However, the project can be implemented using widely available hardware and software, such as a processor with multiple cores, a compatible GPU with CUDA architecture, sufficient RAM, storage, and a high-definition camera.

In conclusion, the Traffic Automobile Counter project using OpenCV and YOLO provides an efficient and accurate solution for counting the number of vehicles in a given video footage and can be applied in various real-world scenarios, such as traffic management, surveillance, and securit

#### FUTURE SCOPE:

The Traffic Automobile Counter project using OpenCV and YOLO has a wide range of future scope and potential for further development and improvement. Here are some possible future directions for the project:

1. Multi-object tracking: Currently, the project detects and counts each vehicle independently. However, in real-world scenarios, multiple vehicles can be in close proximity to each other, making it challenging to count them accurately. Developing a multi-object tracking algorithm can help improve the accuracy of the vehicle count in such scenarios.
2. Integration with traffic management systems: The project can be integrated with traffic management systems to provide real-time data on traffic flow and vehicle count. The data can be used to optimize traffic flow, reduce congestion, and improve road safety.
3. Detection of other types of vehicles: Currently, the project is designed to detect and count automobiles only. However, with the increasing use of other types of vehicles such as bicycles, motorcycles, and electric scooters, the project can be expanded to detect and count these vehicles as well.
4. Real-time analysis of traffic patterns: The project can be enhanced to provide real-time analysis of traffic patterns and identify trends and patterns in traffic flow. The data can be used to optimize traffic management and improve road safety.
5. Integration with other sensors: The project can be integrated with other sensors such as speed sensors, traffic light sensors, and weather sensors to provide a more comprehensive analysis of traffic flow and vehicle count.

# REFERENCES

1. Shah, Rahul D., et al. "A novel deep learning framework for vehicle detection and counting using Yolov4 and OpenCV." Journal of Ambient Intelligence and Humanized Computing 13.2 (2022)
2. Zhang, Yuan, et al. "Real-time vehicle detection and counting using a Yolov4-based model." 2022 12th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM). IEEE, 2022.
3. Boopathi, Thangavel, et al. "Real-time vehicle detection and counting system for traffic management." Journal of Ambient Intelligence and Humanized Computing 12.11 (2021)
4. Singh, Raman, and Arnav Kumar Jain. "Real-time traffic counting and classification using Yolov4." 2020 12th International Conference on Communication Systems & Networks (COMSNETS). IEEE, 2020.
5. OpenCV Documentation: <https://docs.opencv.org/master/d9/df8/tutorial_root.html>
6. Yolov4 Official Repository: <https://github.com/pjreddie/darknet/tree/master/cfg>
7. Redmon, Joseph, and Ali Farhadi. "Yolov4: An Incremental Improvement." arXiv preprint arXiv:1804.02767 (2018).
8. Rosebrock, Adrian. "Vehicle detection, tracking, and counting with OpenCV and Python." pyimagesearch (2018).
9. Balasubramanian, Vignesh, and Anandhakumar Balasubramanian. "Traffic density estimation using Yolov4." International Journal of Computer Applications 179.44 (2021)
10. Narayanan, Hariharan, and Prasanna SR. "Real-time vehicle detection using Yolov4-tiny and OpenCV." 2020 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). IEEE, 2020.
11. Chakraborty, Samrat, and Bishal Paul. "Real-time vehicle detection and classification using Yolov4 and OpenCV." 2019 4th International Conference on Computing, Communication and Security (ICCCS). IEEE, 2019.
12. Ramanathan, Rajesh, and C. Kavitha. "Real-time vehicle detection and tracking using Yolov4 and deep sort." 2019 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). IEEE, 2019.
13. Kuo, Yi-Chang, and Chih-Wei Chiu. "Vehicle detection and tracking using Yolov4 and extended Kalman filter." 2019 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). IEEE, 2019.

Liu, Zhaohui, et al. "Real-time vehicle detection based on Yolov4 and improved Faster R-CNN." 2018 International Conference on Intelligent Transportation, Big Data and Smart City (ICITBS). IEEE, 2018.