TRAFFIC SIGN CLASSIFICATION

Dissertation submitted to

Shri Ramdeobaba College of Engineering & Management, Nagpur

in partial fulfillment of requirement for the award of

Bachelor of Engineering

In

SOFTWARE TECHNOLOGY

By

Akshay Mishra (33)

Amey Bhutada (35)

Anand Sharma (37)

Ananya (39)

Nakul Tibdewal (50)

Sarthak Pande (86)

Guide

Prof. Swati Hira

Computer Science and Engineering

Shri Ramdeobaba College of Engineering & Management, Nagpur 440 013

(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

November, 2020

SHRI RAMDEOBABA COLLEGE OF ENGINEERING & MANAGEMENT, NAGPUR

(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

Department of Computer Science Engineering

CERTIFICATE

This is to certify that the Thesis on "**Traffic Sign Classification**" is a bonafide work of **Akshay Mishra, Amey Bhutada, Anand Sharma, Ananya, Nakul Tibdewal and Sarthak Pande,** submitted to the Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur in partial fulfilment of the award of a Degree of Bachelor of Engineering, in SOFTWARE TECHNOLOGY(CSP 415). It has been carried out at the Department of Computer Science Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur during the academic year 2020.

Date: 11th November, 2020

Place: Nagpur

Prof. Swati Hira
Project guide
Department of Computer Science Engineering

Dr. Manoj B. Chandak
H.O.D
Department of Computer Science Engineering

DECLARATION

We, hereby declare that the thesis titled "**Traffic Sign Classification**" submitted herein, has been carried out in the Department of Computer Science Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree / diploma at this or any other institution / University.

Date: 11 November, 2020

Place: Nagpur

[illegible]	Bhutada	[illegible]
Akshay Mishra	Amey Bhutada	Anand Sharma
(33)	(35)	(37)
[illegible]	[illegible]	[illegible]
Ananya	Nakul Tibdewal	Sarthak Pande
(39)	(50)	(86)

APPROVAL SHEET

This report entitled "**Traffic Sign Classification**" by Akshay Mishra, Amey Bhutada, Anand Sharma, Ananya, Nakul Tibdewal and Sarthak Pande is approved for the degree of Bachelor of Engineering.

Project Guide	External Examiner
Prof. Swati Hira	

HOD

Dr. Manoj B. Chandak

Date: 11 November, 2020

Place: Nagpur

ACKNOWLEDGEMENT

It is a matter of great pleasure to present this report on a project titled "**Traffic Sign Classification**". We are thankful to Shri Ramdeobaba College of Engineering and Management, Nagpur for providing us this great opportunity to be working on this project.

We would like to thank Dr. Manoj B. Chandak, Head of Department, Computer Science and Engineering, RCOEM and our project guide Prof. Swati Hira for providing us with the facilities for completing this project.

Group Members:

Akshay Mishra

Amey Bhutada

Anand Sharma

Ananya

Nakul Tibdewal

Sarthak Pande

Department of Computer Science Engineering

Shri Ramdeobaba College of Engineering & Management, Nagpur

ABSTRACT

Road traffic constitutes a major part in the problem of society. As the road traffic is increasing day by day there is a necessity of following the traffic rules with proper discipline. Traffic rules consist of traffic sign boards and traffic signals which are meant to be followed by everyone in the society. The signboards are captured using cameras installed in the vehicle. This gives the driver a sort of assistance which alerts the driver and reduces the work of the driver. The main goals of this project are detection, and recognition.

Millions of people are injured annually in vehicle accidents. Most of the traffic accidents are the result of carelessness, ignorance of the rules and neglecting traffic signboards, both at the individual level by the drivers and the society at large. The magnitude of road accidents in India is alarming. This is evident from the fact that every hour there are about 56 accidents taking place similarly, every hour more than 14 deaths occur due to road accidents. When someone neglects to obey traffic signs, they are putting themselves at risk as well as other drivers, their passengers and pedestrians. All the signs and signals help keep order in traffic and they also are designed to reduce the number and severity of traffic accidents. Some drivers believe that some traffic signs are simply not necessary.

All road signs are placed in specific areas to ensure the safety of all drivers. These markers let drivers know how fast to drive. They help to create order on the roadways and are employed to provide essential information to drivers. Traffic signs include many useful environmental information which can help drivers learn about the change of the road ahead and the driving requirements. Signs which are taken out of specific places or not visible as a result of wear and tear can pose undesirable risks to drivers. They also tell drivers when and where to turn or not to turn. In order to be a terrific driver, you need to have an understanding of what the signs mean. Road signs are designed to make sure that every driver is kept safe.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I : INTRODUCTION

Nowadays, there is a lot of attention being given to the ability of the car to drive itself. One of the many important aspects for a self driving car is the ability for it to detect traffic signs in order to provide safety and security for the people not only inside the car but also outside of it. The traffic environment consists of different aspects whose main purpose is to regulate flow of traffic, make sure each driver is adhering to the rules so as to provide a safe and secure environment to all the parties concerned. The problem we are trying to solve has some advantages such as traffic signs being unique thereby resulting in object variations being small and traffic signs are clearly visible to the driver/system. The other side of the coin is that we have to contend with lighting and weather conditions.The main objective of our project is to design and construct a computer based system which can automatically detect the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions. The proposed approach consists of building a model using convolutional neural networks by extracting traffic signs from an image using color information.

Traffic signs are an integral part of our road infrastructure. They provide critical information, sometimes compelling recommendations, for road users, which in turn requires them to adjust their driving behaviour to make sure they adhere with whatever road regulation currently enforced. Without such useful signs, we would most likely be faced with more accidents, as drivers would not be given critical feedback on how fast they could safely go, or informed about road works, sharp turns, or school crossings ahead. In our modern age, around 1.3M people die on roads each year. This number would be much higher without our road signs.

Naturally, autonomous vehicles must also abide by road legislation and therefore recognize and understand traffic signs.

Traditionally, standard computer vision methods were employed to detect and classify traffic signs, but these required considerable and time-consuming manual work to handcraft important features in images. Instead, by applying deep learning to this problem, we create a model that reliably classifies traffic signs, learning to identify the most appropriate features for this problem by itself. Here, we have created a deep learning architecture that can identify traffic signs with close to 98% accuracy on the test set.

Literature Review:

To get the proper idea it was important to review some previous work of other researchers in the field of Traffic Sign Detection and Classification. As mentioned in paper [2], there are two tasks in an overall traffic sign recognition system: finding the locations and sizes of traffic signs in natural scene images (**traffic sign detection**) and classifying the detected traffic signs to their specific subclasses (**traffic sign classification**). Traffic signs are designed with regular shapes and conspicuous colors which are different from natural objects to attract human drivers' attention. Meanwhile, the inner contents and diagrams denote the specific meanings of traffic signs. Therefore, traffic signs can be easily captured by human drivers due to their well-defined appearances. However, there are many difficulties for identifying traffic signs by computer, such as illumination changes, color deterioration, motion blur, cluttered background and partial occlusion.

After reviewing one of the research papers[2], which helped in identifying and recognizing traffic sign boards in various backgrounds and lighting conditions from static digital images. In the proposed method, the input image is preprocessed at the first time and then traffic sign location is specified by edge detection and morphology operation and finally traffic sign location is extracted and their characters are recognized. The methodology used for the Pre Processing part were ColorThresholding , Gray Scale Conversion , Use of Median Filter , Edge Detection , Mathematical Morphology and for Recognition and classification part used the techniques such as Sobel edge detector , Canny Edge detector , Artificial Neural Networks(ANN) and Support Vector Machines(SVM).Most of the researchers have designed the model for traffic sign detection using different algorithms but have the same idea of processing the image captured by the camera installed in vehicles and detect the sign on traffic sign boards and notify the user. But in the research paper of IRJET they have suggested one interesting idea of speed control along with sign detection and recognition. The idea was, speed will be controlled automatically according to signs recognized by the system.

In paper[3], an IRJET research paper written by Anju Manjooran, Annmariya Seby, Anphy Varghese, Krishnadas J, there idea was to provide a comprehensive assistance to the driver for following the traffic signs, Traffic Sign Board Detection and Voice Alert System along with Speed Control. The signboards are captured using cameras installed in the vehicle. The captured image will undergo image processing by SURF algorithm in MATLAB and identify the signboard. Technologies used in their project were Image Processing, CNN(Convolutional Neural Network) and SVM(Support Vector Machines).

Some of the gaps in Traffic Signal Detection System:-

1. In dim light or in the absence of light the accuracy of the system will decrease, as the image of the sign board will not be fetched accurately.

2. If the traffic sign boards are installed at very small differences then the frames generated by the system will merge with the other frame so it will be difficult to notify the user, the correct result in such cases.

In one more proposed method in paper[1], the input image is preprocessed at the first time and then traffic sign location is specified by edge detection and morphology operation and finally traffic sign location is extracted and their characters are recognized.

Preprocessing of the dataset was using methods like Color Thresholding, Gray Scale Conversion, Use of Median Filter, Edge Detection and Mathematical Morphology. The Traffic Sign Recognition part was carried out using Sobel edge detection and Canny Edge detection. Many Image Classification Techniques such as Artificial Neural Networks(ANN), Support Vector Machines(SVM), Genetic Algorithms (GA) and Fuzzy support Vector Machines (FSVM) were used.
The paper deals with object detection in outdoor environments which are useful for Driver Support systems and Intelligent Autonomous Vehicles to make some decisions about their speed, trajectory and send a warning signal indicating over speed, warn or limit illegal ma-oeuvres. It works in identifying traffic sign boards detection and recognition.

After understanding the concepts which were mentioned in various literatures it was quite evident that to Recognise and classify the Traffic Signs in Image as well as videos in real time there was a need of an algorithm that is fast and gives a better accuracy than the previously existing algorithms . Yolo-V3 provided the solutions which the problem statement demanded for.

CHAPTER II : DESIGN

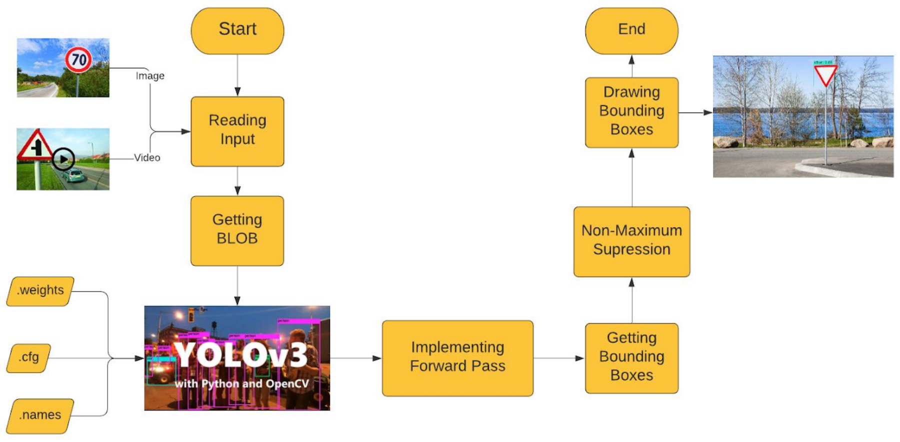


Figure 1: Architectural Design

The need is to design a system that accepts images as input and then detects and classifies traffic signs present in them with appreciable precision. We have broadly divided the whole process into 3 stages (as shown in figure 1):

1. Initial Stage : Here we will be taking an image as input. Also we can take video frames and then split them into images using a suitable threshold FPS(Frames per second). After that the image will be processed into a numpy array for further manipulations and calculations.
2. Intermediate Stage : Important files like .txt file (which contains train and test data path), .names file(names of classes) and configuration file(information regarding structure of custom created model) will be used in loading the YOLO-V3 network using the openCV library and Forward pass will be implemented.
3. Final Stage : Now we have the bounding boxes as our preliminary result. Using Non-Maximum Suppression (a technique that selects a single entity among many overlapping entities, in this case, bounding boxes) , we will eliminate the weak predictions and then finally draw bounding boxes on the objects detected.

CHAPTER III: METHODOLOGY

BASIC IDEA :

The whole idea behind this project is to make a real time custom object detection system for different traffic signs using YOLO-V3. The dataset we will be using is the GTSDB (German Traffic Sign Detection Benchmark). It is a single-image detection assessment for researchers with interest in the field of computer vision, pattern recognition and image-based driver assistance. We can broadly segregate the work into three major parts:

1. Data Preprocessing (Converting Traffic signs dataset into YOLO format)
2. Training the files to make a custom model
3. Designing a GUI to classify inputs.

TECHNOLOGICAL STACK:

1. **Python programming language**

Python is an interpreted, high-level and general-purpose programming language.
It is used for web development, software development, scripting, etc.

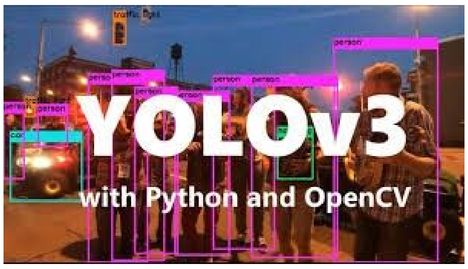
2. **PyQt5**

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt implements around 440 classes and over 6,000 functions and methods including: a substantial set of GUI widgets, classes for accessing SQL databases,etc.

3. **OpenCV**

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision.

4. **YOLO-V3**



YOLOv3 is the latest variant of a popular object detection algorithm YOLO – You Only Look Once. The published model recognizes 80 different objects in images and videos, but most importantly it is super fast and nearly as accurate as Single Shot MultiBox (SSD).

5. **Darknet**: Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.

Object Detection :

Object detection reduces human efforts in many fields. In our case, we are using YOLO V3 to detect an object. YOLO V3 has DARKNET-53, with these 53 layers; the model is more powerful to identify even the small objects from the image. YOLO V3 is able to identify more than 80 different objects in one image. YOLO V3 can bring down the error rate drastically. YOLO v3 uses a thin sized boundary box.

How does a YOLO work?

The YOLO (You Only Look Once) algorithm divides any given input image into an SxS grid system. Each grind on the input image is responsible for detection on the object. Now the grid cell predicts the number of boundary boxes for an object.

As shown in Figure 2, every boundary box has five elements (x, y, w, h, confidence score). X and y are the coordinates of the object in the input image, w and h are the width and height of the object respectively. Confidence score is the probability that the box contains an object and how accurate is the boundary box.

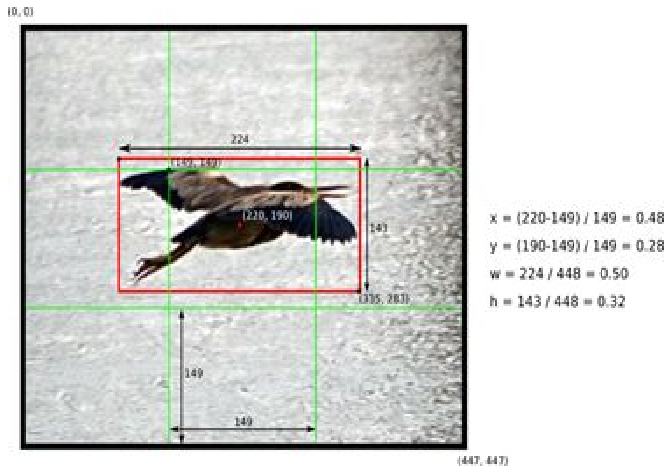


Figure 2: Bounding box in YOLO-V3

YOLO algorithm : It is based on regression where object detection and localization and classification of the object for the input image will take place in a single go. This type of algorithm is commonly used in real-time object detection(As shown in Figure 3).

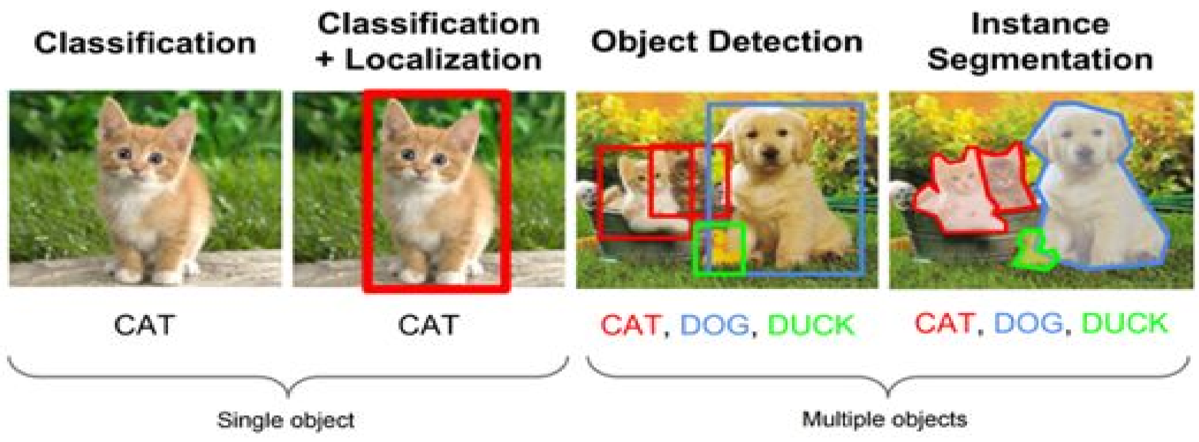


Figure 3: Application of YOLO on single and multiple objects

YOLO v1 :

Architecture :

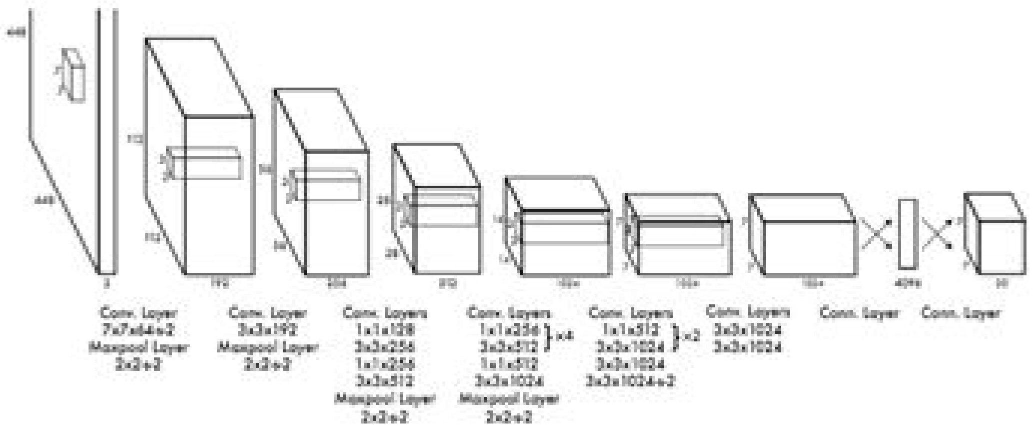


Figure 4 : Architecture of YOLO-v1

As shown in Figure 4, YOLO v1 uses the Darknet framework which is trained on ImageNet-1000 dataset. This works as mentioned above but has many limitations because of it the use of the YOLO v1 is restricted. It could not find small objects if they appeared as a cluster. This architecture found difficulty in generalisation of objects if the image is of other dimensions different from the trained image. The major issue is localization of objects in the input image.

Problem with YOLO v1:

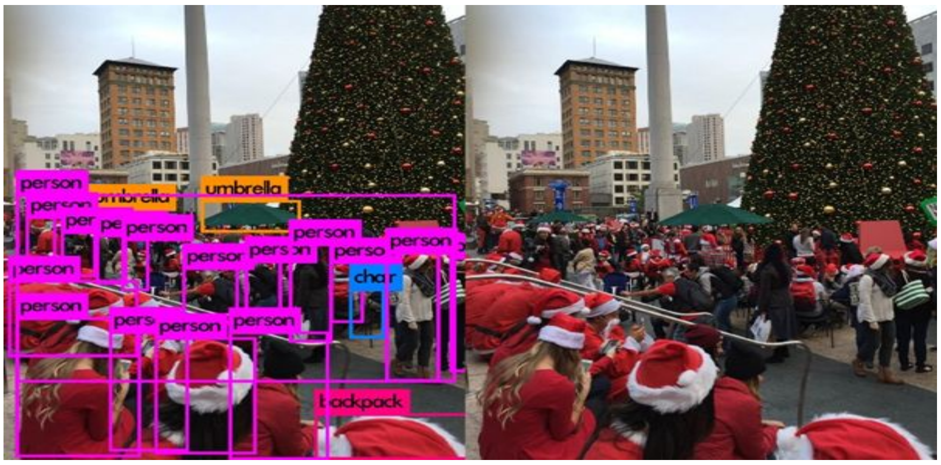


Figure 5: Problem with YOLO-v1

From the above image(Figure 5), we can see that YOLO version1 has limitations based upon the closeness of the object. As we can see, YOLO is detecting only 5 santa's from the lower left corner, but there are 9 santas.

YOLO v2

The second version of the YOLO is named as YOLO9000 which has been published by Joseph Redmon and Ali Farhadi at the end of 2016. The major improvements of this version are better, faster and more advanced to meet the Faster R-CNN which is also an object detection algorithm which uses a Region Proposal Network to identify the objects from the image input and SSD(Single Shot Multibox Detector).

The changes from YOLO v1 to Yolo v2 :

Batch Normalization: it normalises the input layer by altering slightly and scaling the activations. Batch normalization decreases the shift in unit value in the hidden layer and by doing so it improves the stability of the neural network. By adding batch normalization to convolutional layers in the architecture mAP (mean average precision) has been improved by 2% . It also helped the model regularise and overfitting has been reduced overall.

Higher Resolution Classifier: the input size in YOLO v2 has been increased from 224*224 to 448*448. The increase in the input size of the image has improved the MAP (mean average precision) upto 4%. This increase in input size has been applied while training the YOLO v2 architecture DarkNet 19 on ImageNet dataset.

Anchor Boxes: one of the most notable changes which can be visible in YOLO v2 is the introduction of the anchor boxes. YOLO v2 does classification and prediction in a single framework. These anchor boxes are responsible for predicting bounding boxes and these anchor boxes are designed for a given dataset by using clustering(k-means clustering).

Fine-Grained Features: one of the main issues that has to be addressed in the YOLO v1 is the detection of smaller objects on the image. This has been resolved in the YOLO v2 divides the image into 13*13 grid cells which are smaller when compared to its previous version. This enables the yolo v2 to identify or localize the smaller objects in the image and also be effective with the larger objects.

Multi-Scale Training: on YOLO v1 has a weakness detecting objects with different input sizes which says that if YOLO is trained with small images of a particular object it has issues detecting the same object on an image of bigger size. This has been resolved to a great extent in YOLO v2 where it is trained with random images with different dimensions range between 320*320 to 608*608. This allows the network to learn and predict the objects from various input dimensions with accuracy.

Why YOLO v3?

The previous version has been improved for an incremental improvement which is now called YOLO v3. As many object detection algorithms have been there for a while now the competition is all about how accurate and quickly objects are detected. YOLO v3 has all we need for object detection in real-time with accurately and classifying the objects. The authors named this as an incremental improvement

Improvements in YOLO v3 :

Bounding Box Predictions: In YOLO v3 gives the score for the objects for each bounding box. It uses logistic regression to predict the objectiveness score.

Class Predictions: In YOLO v3 it uses logistic classifiers for every class instead of softmax which has been used in the previous YOLO v2. By doing so in YOLO v3 we can have multi-label classification. With a softmax layer if the network is trained for both a person and man, it gives the probability between person and man let's say 0.4 and 0.47. With the independent classifier gives the probability for each class of objects. For example if the network is trained for a person and a man it would give the probability of 0.85 to person and 0.8 for the man and label the object in the picture as both man and person.

Feature Pyramid Networks (FPN): YOLO v3 makes predictions similar to the FPN where 3 predictions are made for every location the input image and features are extracted from each prediction. By doing so YOLO v3 has the better ability at different scales. As explained from the paper, each prediction is composed of a boundary box, objectness and 80 class scores. Doing upsampling from previous layers allows getting meaningful semantic information and finer-grained information from earlier feature maps. Now, adding a few more convolutional layers to process improves the output.

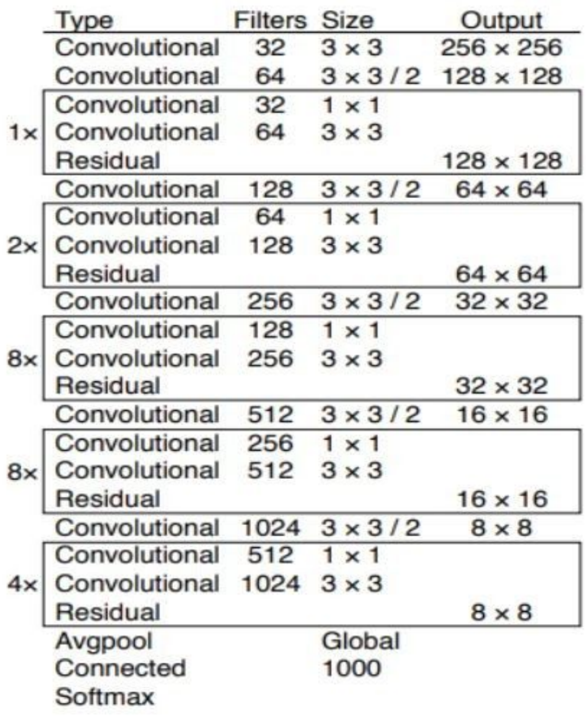
Darknet-53: the predecessor YOLO v2 used Darknet-19 as feature extractor and YOLO v3 uses the Darknet-53 network for feature extractor which has 53 convolutional layers. It is much deeper than the YOL v2 and also has shortcut connections. Darknet-53 consists mainly of 3x3 and 1x1 filters with shortcut connections.

YOLO-V3

YOLO stands for "You Only Look Once" and uses convolutional neural networks (CNN) for object detection.

When YOLO works it predicts classes' labels and detects locations of objects at the same time. That is why, YOLO can detect multiple objects in one image. The name of the algorithm means that a single network just once is applied to the whole image. YOLO divides image into regions, predicts bounding boxes and probabilities for every such region. YOLO also predicts confidence for every bounding box showing information that this particular bounding box actually includes object, and probability of included object in the bounding box being a particular class. Then, bounding boxes are filtered with a technique called non-maximum suppression that excludes some of them if confidence is low or there is another bounding box for this region with higher confidence.

YOLO-3 is the latest version that uses successive 3x3 and 1x1 convolutional layers. In total it has 53 convolutional layers with architecture as shown on the Figure 6 below. Every layer is followed by batch normalization and Leaky ReLU activation.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 6:YOLO V-3 Architecture

PART A - DATA PREPROCESSING

The dataset features:

- a single-image detection problem
- 900 images (divided in 600 training images and 300 evaluation images)
- These images are divided into 42 classes and these classes are divided into 4 categories.
- This 4 categories are:
  - Prohibitory
  - Danger
  - Mandatory
  - Other

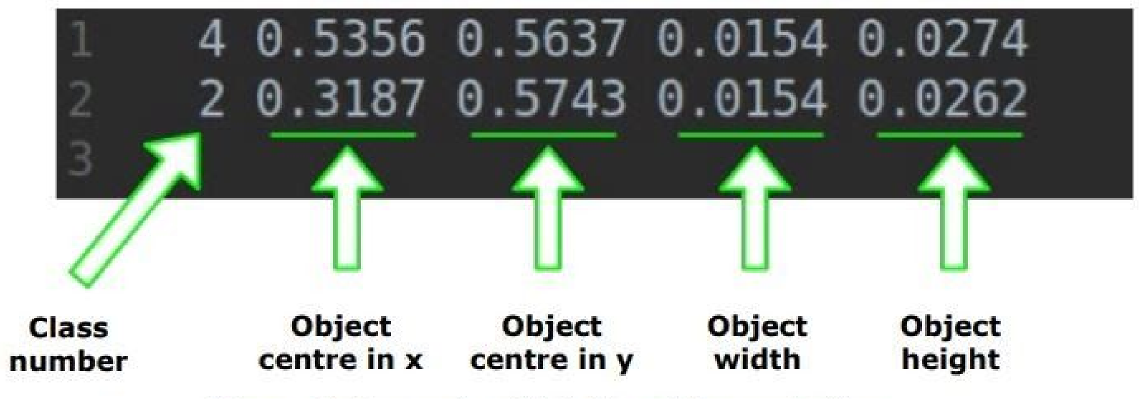
Image Format:

- Images are stored in PPM format.
- The sizes of the traffic signs in the images vary from 16x16 to 128x128.
- Traffic signs may appear in every perspective and under every lighting condition.

Annotation format:

Every image has to have text file with the same name as image file has:

Inside the .txt file every line describes only one object and consists of class number, object centre in x, object centre in y, object width and object height (as shown in Figure 7).



```
4 0.5356 0.5637 0.0154 0.0274
2 0.3187 0.5743 0.0154 0.0262
```

Figure 7 : Annotations file

Numbers for centre point in x, centre point in y, object width and object height need to be in range from 0 to 1. They are normalized by real image width and real image height consequently.

- Annotations are provided in CSV files. Fields are separated by a semicolon (;). They contain the following information:
  - Filename: Filename of the image the annotations apply for
  - Traffic sign's region of interest (ROI) in the image
    - leftmost image column of the ROI
    - upmost image row of the ROI
    - rightmost image column of the ROI
    - downmost image row of the ROI
  - ID providing the traffic sign's class
- **Annotations of bounding boxes' coordinates in txt file are as following:**
  XMin, YMin, XMax, YMax
- **but YOLO needs following:**
  [centre in x] [centre in y] [width] [height]
- This annotation has to be converted into a single .txt file for every image and having a filename same as the image.

Verify annotations by LabelIMG:

- After converting annotations into YOLO format, it is possible to check that calculations for bounding boxes were made correctly.
- Open folder with images and just created .txt files with annotations.
- Create one more txt file with name classes.txt (use any text editor like notepad or other) and in every separate line write categories' names that we used for Traffic Signs: prohibitory, danger, mandatory, other.
- Save changes and close the file classes.txt.
- Open Terminal (or Anaconda Prompt) and activate the environment in which you installed the LabelIMG tool.
- Launch LabelIMG by one of the following command (depending on the way you chose for installation):
- Click on the button Open Dir and navigate to the folder with images, annotations in txt files and just created file classes.txt.
- By using Next and Previous, check if bounding boxes cover regions with needed objects.

PART B - MODEL TRAINING

For Model Training, we first have to prepare the files, set up configuration files and then Run the training.

Prepare the files:

For every dataset, we must have a:

1. .data file (it contains information like number of classes, train location, valid location, names location and backup location.)
2. classes.names file(it contains names of all the classes)
3. train.txt(full path to training images)
4. test.txt(full path to testing images)

Configuration files:

A Configuration file consists of parameters of training like learning rate, angle, hue, saturation, exposure and structures of CNN like filters, activation, size and stride. It also has the last three YOLO layers structures.

Our task is to change the number of classes and filters in the last 3 YOLO layers and its above convolution layers. We also have to take care for certain other parameters for good functioning of our custom object detector.

- filters = (dataset + coordinates + 1) * masks

  So, in our customized data, filters = (4 + 4 + 1) * 3 = 27

- Max_batches : It is the total number of iterations.

  Steps : Number of iterations after which learning rate changes.

  So, we have, max_batches = #classes * 2000 (but not less than 4000) and

  steps = (80% of max_batches, 90% of max_batches)

- Batch : It is the number of images processed in a single iteration.

  Subdivision : It is the number of mini-batches.

  For Training file, batch = 32 and subdivision = 16

  For Validation file, batch = 1 and subdivision = 1

Running Training :

For Training purposes, we use the state-of-the-art Darknet architecture. It uses pre trained weights and applies transfer learning to output the new weights.

The command used is as follows:

```
./darknet detector train <DATA_FILE> <CFG FILE> <PRE-TRAINED WEIGHTS>
```

The trained weights are saved in the backup folder. Also, weights are saved every 100 and every 1000 iterations.

NOTE - When to stop Training?

Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. Overfitting is the phenomenon where we get high accuracy on training dataset and relatively lower accuracy on test dataset.

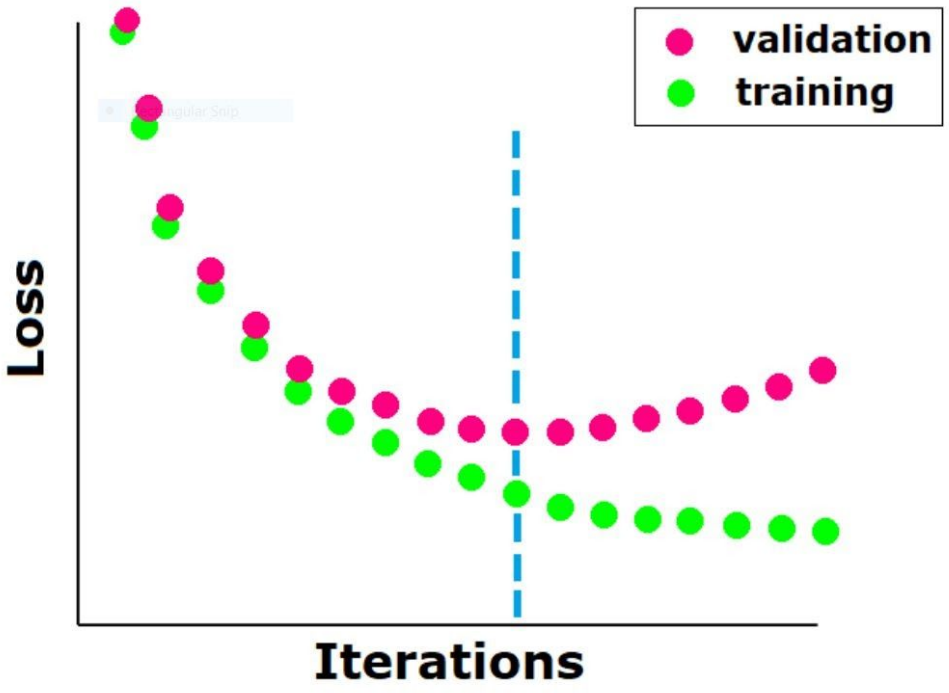


Figure 8 : Overfitting

To avoid this problem, we calculate mean average precision (mAP) for the trained weights and find out the maximum mAP.

In the training process, the weights get saved in every 100 and 1000 iterations. That means after we finish the training process, in our weights directory, we will be having the the following weights:

- traffic_signs_100.weights
- traffic_signs_200.weights
- traffic_signs_300.weights

and so on.

Now our task will be to find mAP for individual weights and select the one with the maximum value.

The command used is as follows:

```
./darknet detector map <DATA_FILE> <CFG FILE> <WEIGHTS FILE>
```

PART C - GUI DESIGN

We have designed a simple Graphical User Interface which takes input an image and outputs the bounding boxes and names of the classified traffic sign. We have used PyQt5 along with the designer tool to automate this process. We have made modules for UI creation and Object Detection.

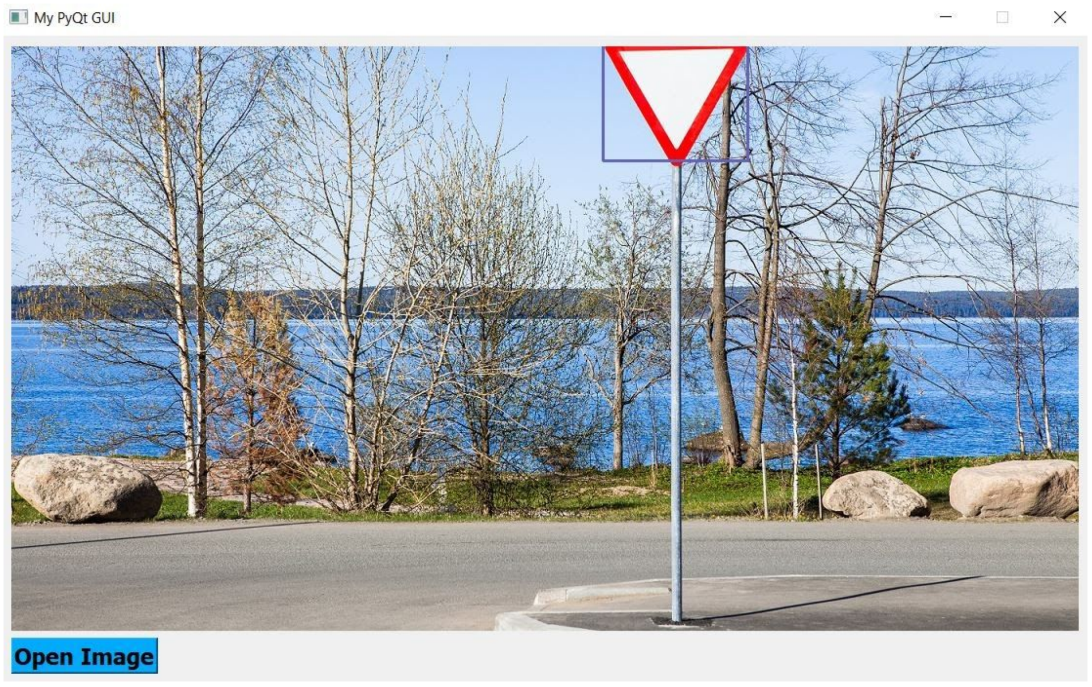


Figure 9 : User Interface

Using the GUI, the user can open any image and then click on ‘process’ to detect and localize all the traffic signs in that image. It will also display the type of that traffic sign and the accuracy with which it is determined using bounding boxes (as shown in figure 9).

CHAPTER IV : RESULTS

We have tested it on some sample images and the expected output has been achieved.

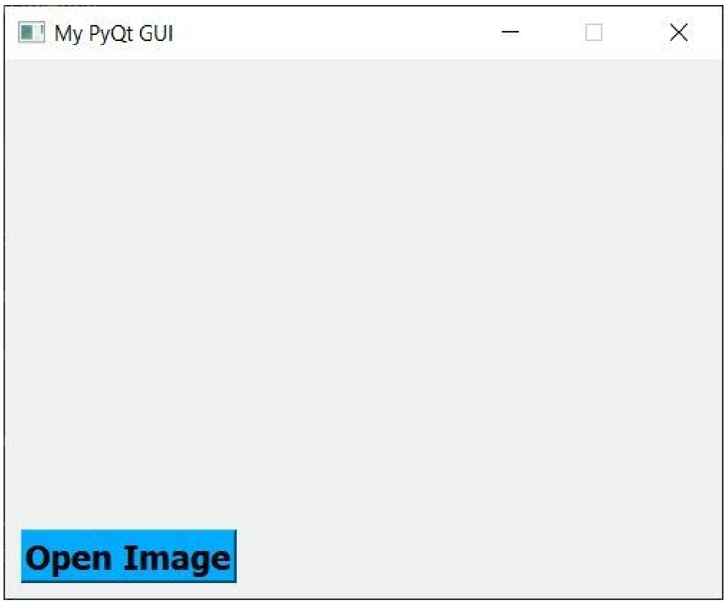


Figure 10 : Simple UI Prompt

Figure 10 shows a simple GUI which shows a OPEN IMAGE prompt which is used to select the input images.

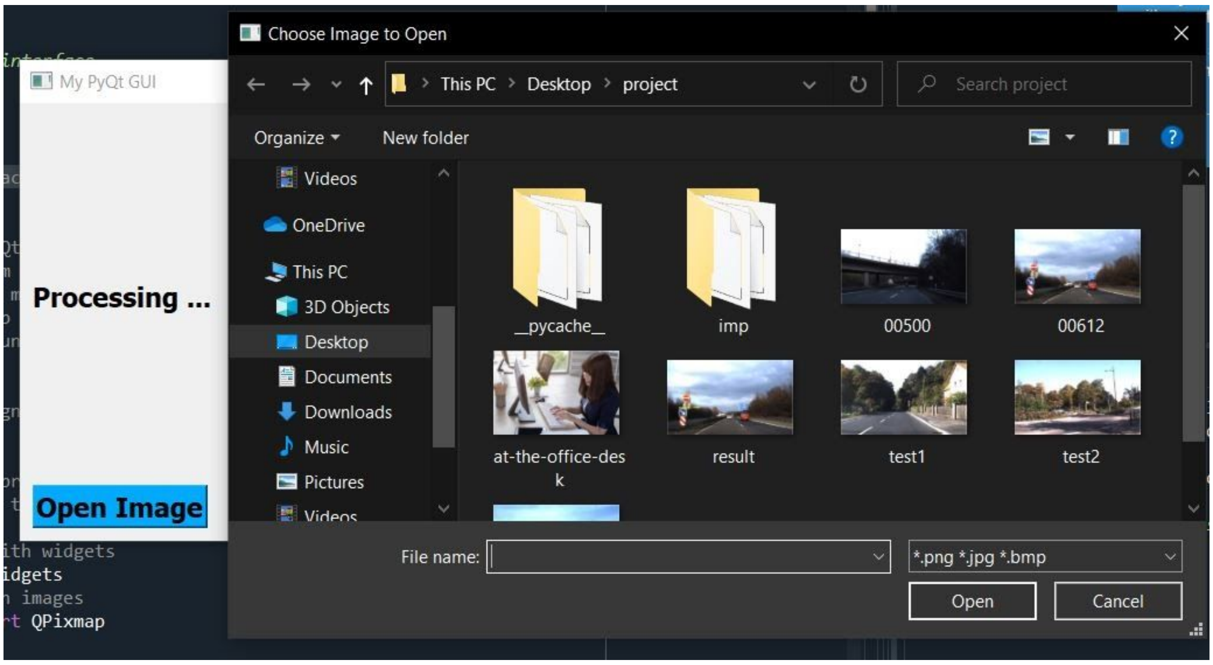


Figure 11 : FIle Selector

Figure 11 shows selection on input image for classification.

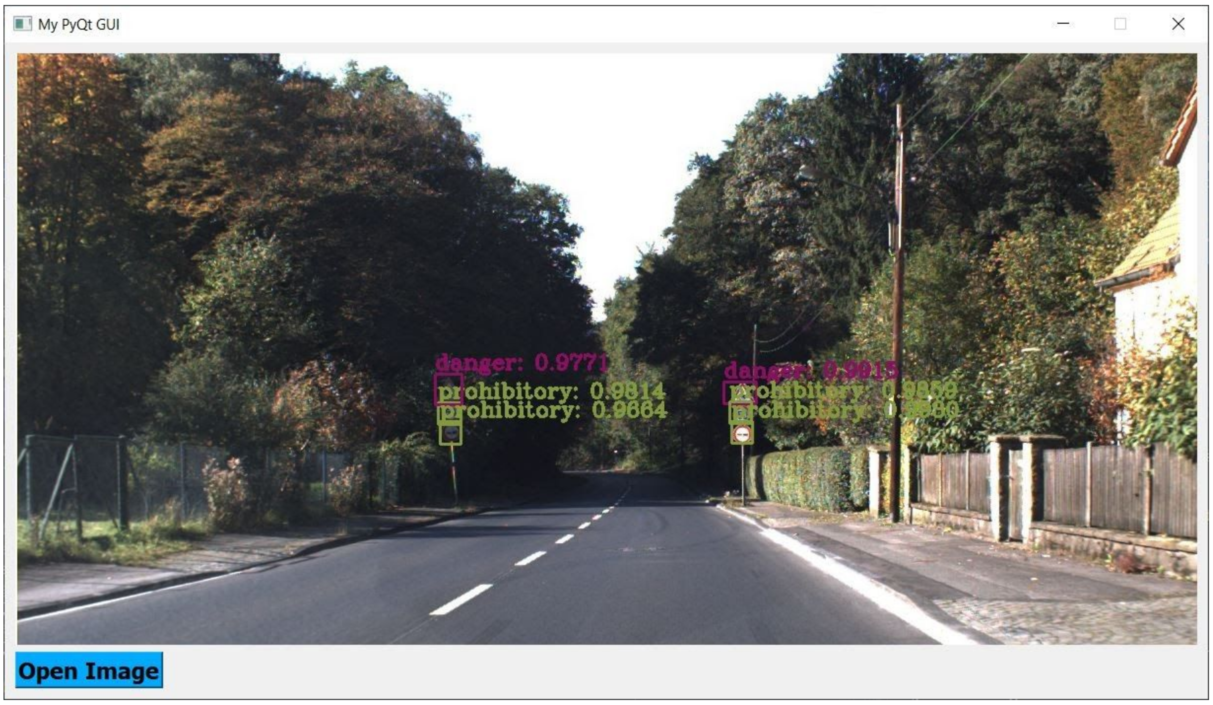


Figure 12 : Final Result

In Figure 12, we see the final result where traffic signs are detected and classified with an 97% precision.

CHAPTER V : CONCLUSION

We have successfully completed the classification of traffic signs by using YOLO-V3 algorithms and produced the expected results. Apart from images, we can also detect traffic signs in videos and real time footage.

To achieve the results, we have:

1. Converted our custom dataset into YOLO format.
2. Prepared files for training and validation.
3. Trained our custom object detector on the Darknet framework.
4. Developed a GUI to detect input images.

The classification of traffic signs into different categories is a very interesting topic where lots and lots of research can be done.

REFERENCES

[1] Alberto Broggi, Pietro Cerri, Paolo Medici, Pier Paolo Porta VisLab,”Real Time Road Signs Recognition,”2007 IEEE Intelligent Vehicles Symposium Istanbul, Turkey, June 13-15, 2007.

[2] Ying Sun, Pingshu Geˆ, Dequan Liu, “Traffic Sign Detection and Recognition Based on Convolutional Neural Network,”College of Mechanical and Electronic Engineering Dalian Minzu University Dalian, China.

[3] Aashrith Vennelakanti, Smriti Shreya, Resmi Rajendran, Debasis Sarkar, Deepak Muddegowda”Traffic Sign Detection and Recognition using a CNN Ensemble,”Affiliation: Qualcomm India Private Limited, Bangalore.

[4] Yi Yang, Hengliang Luo, Huarong Xu and Fuchao Wu ”Towards Real-Time Traffic Sign Detection and Classification ,”2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC) October 8-11, 2014. Qingdao, China.

[5] Xianyan Kuang, Wenbin Fu, Liu Yang Jiangxi ”Real-Time Detection and Recognition of Road Traffic Signs using MSER and Random Forests ,” University of Science and Technology, Ganzhou, China.

[6]https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2 - Architectural Design

[7]https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006 - How does YOLO work

[8]https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89 - YOLO v1

[9]https://towardsdatascience.com/evolution-of-yolo-yolo-version-1-afb8af302bd2 - Problems with YOLO v1

[10]https://www.geeksforgeeks.org/yolo-v2-object-detection/ - YOLO v2 and changes from YOLO v1.

[11]https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b - Why YOLO v3

[12]https://medium.com/analytics-vidhya/preparing-yolo-v3-custom-data-3a9fb35e9c6d - Annotations file for custom data

[13]https://github.com/tzutalin/labelImg - LabelImg tool

[14]https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ - Overfitting

[15]https://towardsdatascience.com/what-is-map-understanding-the-statistic-of-choice-for-comparing-object-detection-models-1ea4f67a9dbd - Mean Average Precision