Maria Mumtaz
Schwarzenburgstrasse 217
3097 Bern
maria.asad17@gmail.com
maria.mumtaz@unibe.ch

Data Science Project

Object Detection and Localization for Autonomous Cars

Conceptual Design Report

30 October 2021

Abstract

Research and development of self-driving vehicles has been gaining momentum due to new machine learning techniques and has seen significant advancement especially over the last decade. According to a recent technical report by the National Highway Traffic Safety Administration (NHTSA) [7], human factors account for around 94% of road accidents. Against this backdrop, autonomous driving systems are being developed with the promise of making roads safer, reducing emissions, transporting the mobility-impaired and improving fuel economics. This project aims to design a machine learning algorithm for object detection and localization for self-driving vehicles. The machine learning classification model would predict if a road image has a car, a pedestrian or nothing (background) and localize it by generating a bounding box around its whereabouts in that image. Convolutional Neural Networks and other deep learning methods will be used to detect and classify the object, whereas the state of the art algorithms such as YOLO would be employed to localize and generate the bounding box.

Table of Contents

Table of Contents	**1**
1 Project Objectives	**2**
2 Methods	**3**
3 Data	**7**
4 Metadata	**8**
5 Data Quality	**8**
6 Data Flow	**8**
7 Data Model	**9**
8 Risks	**9**
9 Preliminary Studies	**10**
10 Conclusions	**14**
Acknowledgements	**15**
References and Bibliography	**15**

1 Project Objectives

The core functions of a modular object detection for autonomous driving systems can be summarized as:

- Perception and assessment
- localization and mapping
- planning and decision making
- vehicle control and human-machine interface.

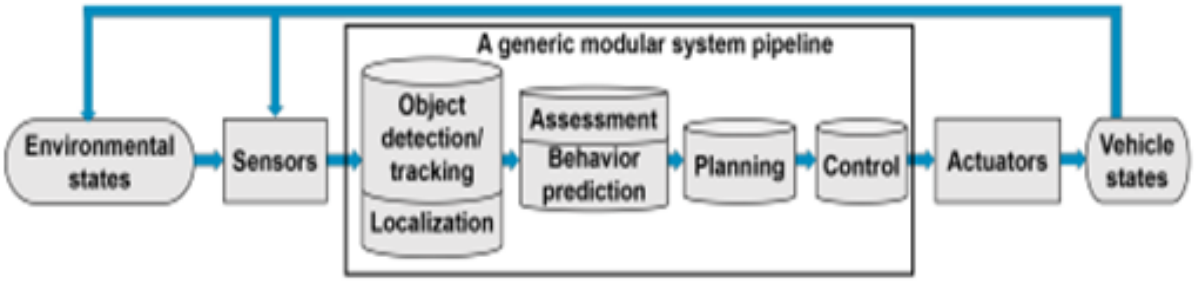


Fig 1: A high level classification of Automated driving system architecture

The goal of this project is to detect and categorize objects in real-time based on object classification and localization. A forward facing camera mounted on a car would output an image stream to the detection system while the car is driving. The problem is formulated as multi-label classification with no, one or many objects present in the captured image. Supervised machine learning is used for training the classifier to output not just a class label (object detection) but also parameters to generate a bounding box around the detected object.

For image classification, a multi layered Convolutional Neural Network will be trained. The result is fed to a Softmax unit that outputs the predicted class. In addition to this output, the ConvNet would also output the coordinates of the middle point of the detected object along with the width and length of the bounding box. The bounding box prediction will be done using the 'You-Only-Look-Once' (YOLO) algorithm.

2 Methods

The object detection system will be designed in python and employ many libraries, mainly:

Library name	Functionality
Numpy	For image representation as multi-dimensional array of pixel intensities
Matplotlib	For visualization of charts and graphs
Tensorflow	For implementing and training of deep learning models
Keras	For training of ML models and their evaluation
OpenCV2	For image and video processing

Table 2: List of libraries to use for image classification and localization

The training dataset would use static car images to learn vehicle heuristics, whereas the testing would be done on a video stream. For the sake of reproducibility, the test data would be splitted into test and validation sets.

2.1 Convolutional Neural Network:

Convolutional neural network or ConvNet is a specialized type of linear operation and can be seen as the simple application of a filter matrix to a determined input. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in the input by tweaking the parameters of the convolution. The network can adjust itself to reduce the error and therefore learn the best parameters to extract relevant information on the database.

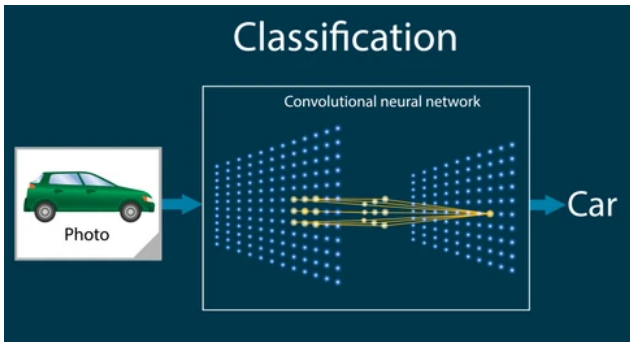


Fig 2.1.1: Representation of filter multiplication for a ConvNet

The first part of the project will be designing the ConvNet. The key component of the ConvNet is the convolutional layer itself. The convolution is based on the filter matrix and this matrix is then convolved with a local region of the input image and can be defined as:

$$y_j = \sum w_{ij} * x + b_j,$$

Where, the operator * represents the convolution operation, w is the filter matrix and b is the bias, x is the input and y is the output.

During the training process, the filter matrix will constantly update itself to get the weight that minimizes loss. Relu activation will be used because it converges faster. The output will then be modified by the max-pooling layer which keeps more information about the input image, like background and texture. See Figure 2.1.2.

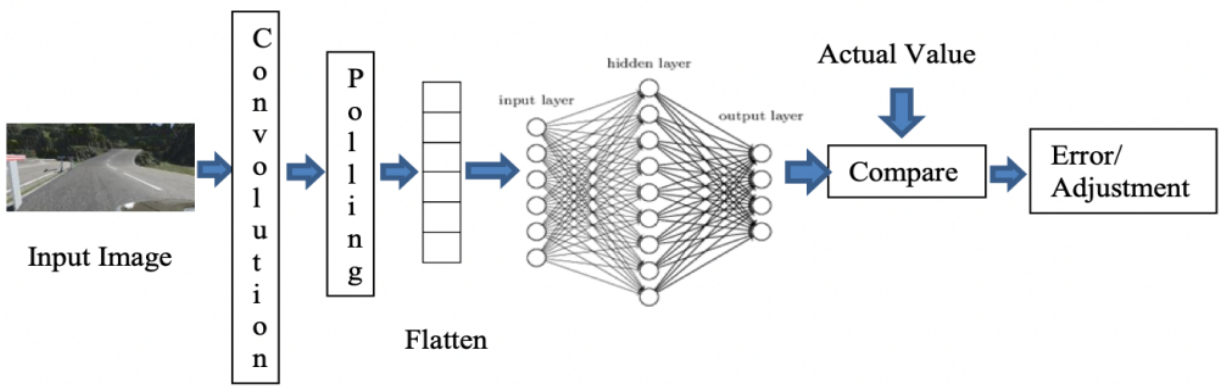


Fig 2.1.2: Image classification using ConvNet, Max pooling and fully connected layer

The output of the ConvNet will be an eight dimensional array yhat = (Pc, bx, by, bh, bw, c1, c2, c3). Where, Pc indicates the presence/absence of the object, bx and by are the coordinates of the middle point of the detected object, bh and bw are the height and width of the bounding box, c1, c2 and c3 are class labels for car,pedestrian,nothing respectively. The loss function would be:

Loss = (y1 - yhat)^2 + (y2 - yhat)^2 + ...+ (y8 - yhat)^2

2.2 Bounding Box Prediction

The object detection can be done using a sliding window algorithm, starting by picking the window size and then feeding this to the input image. The window is then shifted a little over and then fed again to the image. A big step size will reduce the number of windows but the coarse granularity will affect performance. Whereas a fine granularity (meaning small step size) will result

in high computational cost. To overcome this problem, the sliding window can be implemented convolutionally where all the windows share the computation at once. Instead of running the computation on the subsets of the image independently, it combines all the propagation into one form of computation, convolutionally making all the predictions at the same time. This is called the YOLO Algorithm.

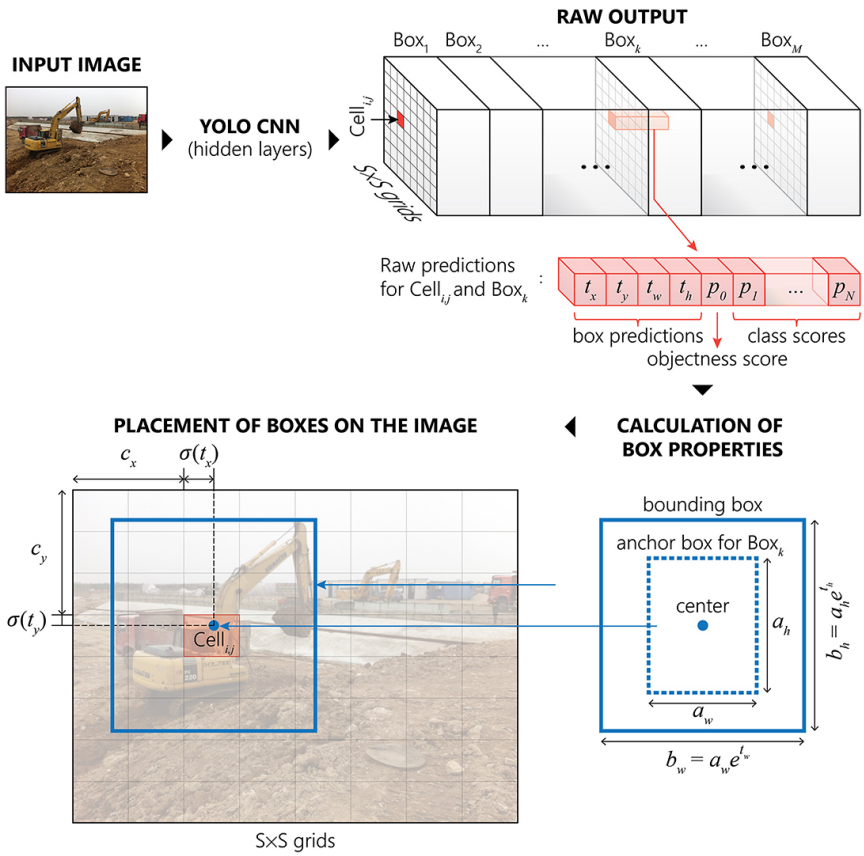


Fig 2.2: SxS gird applied convolutionally to the image for bounding box prediction

2.3 The YOLO Algorithm

The 'You Only Look Once' or YOLO algorithm sees the entire image and requires only a single forward propagation to detect objects on run-time. In this algorithm, a grid is placed on the image and then a single neural network is applied to the entire image. This outputs the image classification and localization for each cell of the grid. Hence for each grid cell, there is a Y label that is an eight dimensional vector.

If a grid cell has two objects, the yolo algorithm takes the midpoint of each of the object and assigns the object to the grid cell containing the midpoint. A finer grid has less chances of having more than one object in one grid cell.

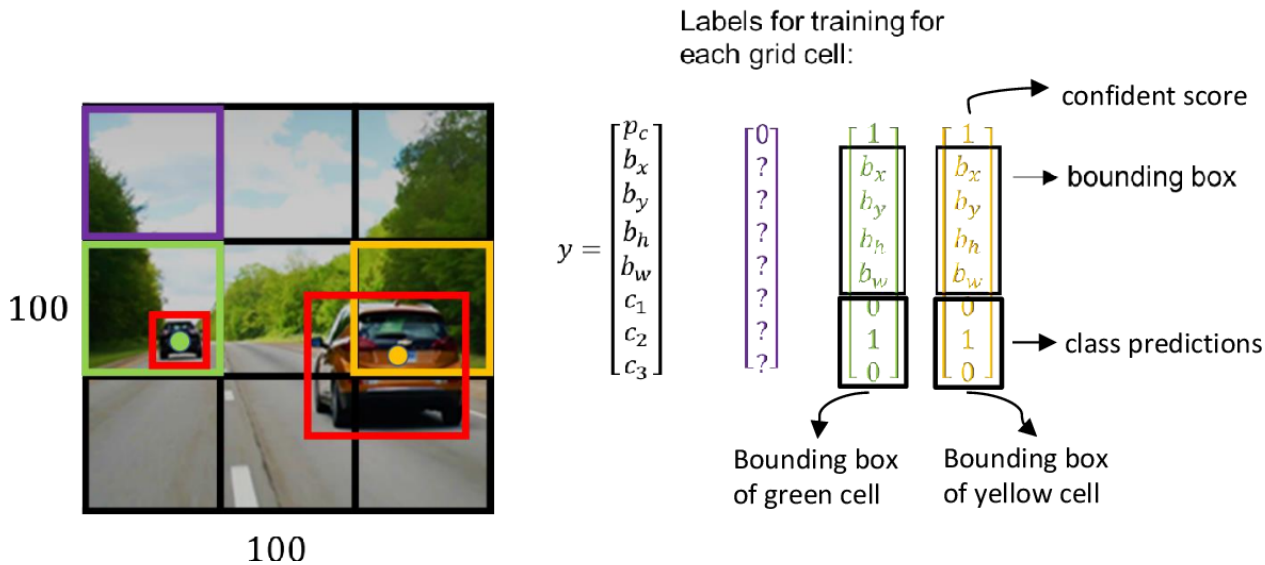


Fig 2.3: Bounding box prediction using YOLO algorithm

2.4 Intersection over Union

Intersection over Union function is an evaluation metric used to measure the accuracy of the bounding box. It is used as a function of the ground-truth bounding box and the predicted bounding box. If the predicted and the ground-truth bounding boxes overlap perfectly, the IoU will be 1. For better prediction, IoU should be greater than 0.5.

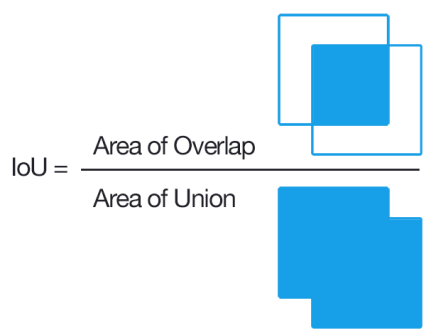


Fig 2.4: Computing IoU by dividing the area of overlap between the bounding boxes by the area of union

2.5 Non-max suppression

With the Yolo algorithm, it is possible that more than one grid cell identify the centre of the single object. This would result in multiple detections of the same object. Non-max suppression is a way to make sure that the algorithm detects each object only once. It compares the probabilities

associated with each of these detections and takes the largest probability, discarding all others. Next, it discards any bounding box with lower IoU. For our project of three different object detections, it independently carries out non max-suppression thrice for each category of the object detection.

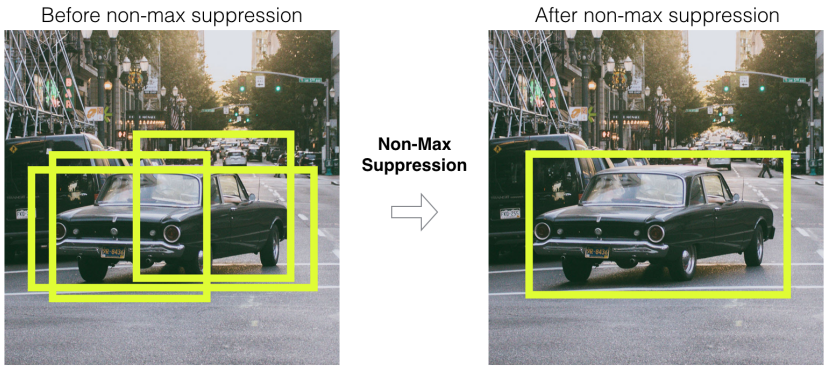


Fig 2.5: Before and after of non-max suppression

2.6 Hardware

For this project, due to the large volume of data in the format of images, a lot of computational power is required in order to achieve acceptable runtimes and timely results including:

1. Local workstation with 2.8GHz i7 vPro and 8 GB of RAM
2. Local workstation with 1x Nvidia Quadro M220 4GB and 128 GB of RAM

3 Data

Developing and improving autonomous car systems require massive real-world data from the high-resolution cameras. Obtaining this data requires computing solutions capable of storing and recording data in real-time with speed and accuracy.

The data is gathered by mounting a camera on the hood of the car. Video streams from dash cams from in the city of Palma de Mallorca, Spain and Bernese-Oberland region are collected. These videos are then processed to output different frames of images and their respective labels on presence or absence of objects in them using OpenCV2 library. In addition, Stanford Car dataset [10] would also be used to train the Neural Network on car features. The images are then divided into two groups, one with cars/pedestrians and other with nothing/background.

All images would be rescaled to 360 * 240 pixel dimension. Data augmentation techniques (flipping, rotating, cropping etc.) may be applied to improve accuracy.

4 Metadata

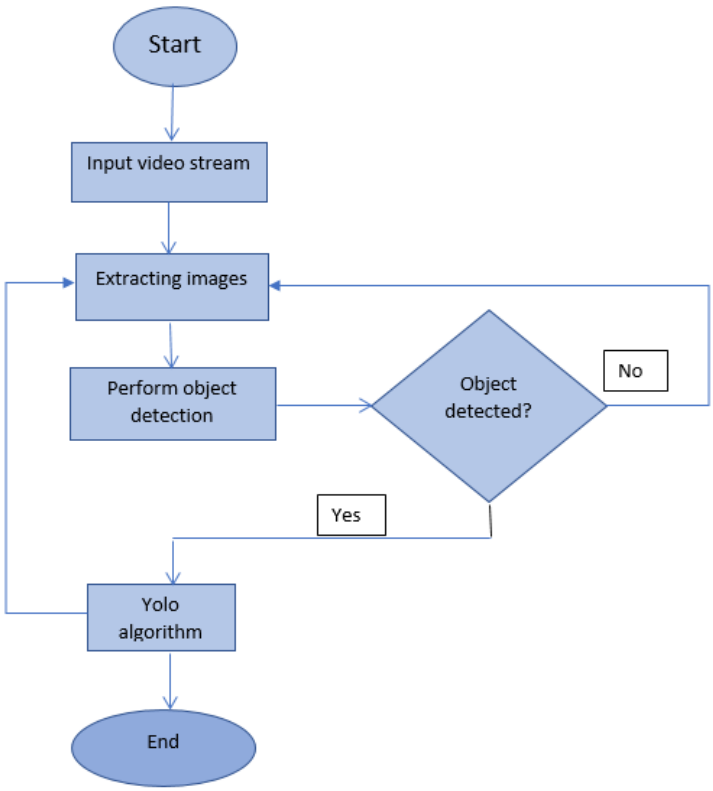
Since training is dome on images, no specific metadata would be needed. The processed images would be represented as an array of [360,240,3] for a 3-channel colored image of 360x240 dimension. The class label of training images would be 0 or 1 indicating presence or absence of a car in it.

5 Data Quality

As machine learning algorithms are prevailing for automated driving, an enormous amount of driving data is in demand for training and testing purposes, making data collection on public roads a valuable activity. The datasets differ from each other in various aspects such as data size, sensor setup and data format. This raises a key pertinent question: Which datasets are suitable for what self-driving algorithms?

A most comprehensive survey on publicly available datasets for self-driving vehicles algorithms can be found in [2].

6 Data Flow



7 Data Model

Since the project uses video streams and images for training and testing, there is no structured data model present for the input data. Having said that, the data may still be classified into images as a set of numerical arrays with values between 0 and 255, and associated numerical labels such as 0 or 1 indicating presence or absence of predicted class.

7.1 Conceptual Data Model

On a high conceptual level, the labeled video stream may be represented as a time series of multi channel image frames with class label as depicted in the following illustration:

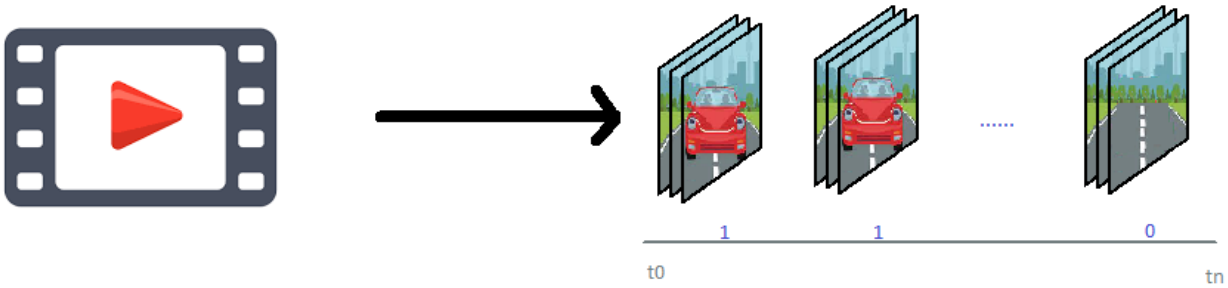


Fig 7: Conceptual data model

7.2 Logical & Physical Data Model

Video streams or images do not have a logical or physical data model of related entities.

8 Risks

Demonstrating the safety and risk factors of a system which is running deep learning techniques depends heavily on the type of technique and the application context. Thus, reasoning about the safety of such algorithms require:

- Understanding the impact of the possible failures
- Understanding the context within the wider system
- Defining the assumption regarding the system context and the environment in which it will likely be used
- Defining what a safe behavior means, including non-functional constraints

In [3], an example is mapped on the above requirements with respect to a deep learning component. The problem space for the component is pedestrian detection with convolutional neural networks. The top level task of the system is to locate an object of class from a distance of 100 meters, with a lateral accuracy of +/- 20 cm, a false negative rate of 1% and false positive rate of 5%. The assumption is that the braking distance and speed are sufficient to react when detecting persons which are 100 meters ahead of the planned trajectory of the vehicle. Alternative sensing methods can be used in order to reduce the overall false negative and false positive rates of the system.

In reality, a machine learning system only encounters a finite number of test samples and an actual operational risk is an empirical quantity on the test set. The operational risk may be much larger than the actual risk for small cardinality test sets. This uncertainty caused by the instantiation of the test set can have large safety implications on individual test examples.

Another safety aspect could be the stopping of autonomous control software once a failure is detected [6]. Specific fault types and failures could be found in [5]

9 Preliminary Studies

Using OpenCV2, the images were extracted from the video stream. With some pre-processing and resizing the image data is prepared to be fed to ConvNet.

The preliminary study was performed on video taken at Mallorca during module 3. This video was manually processed and labelled for detection of cars in OpenCV2 python library. The video was converted to images at every 300 frames for labelling.

```
In [ ]: for file in os.listdir("vids/"):
            if file.endswith(".MP4"):
                capture = cv.VideoCapture('vids/'+str(file))
                cont = capture.isOpened()

                while cont:

                    isTrue, frame = capture.read()
                    if not isTrue:
                        break

                    cv.namedWindow("image")

                    if fno%300==0 and fno !=0:
                        cv.imshow('image', frame)
                        cv.imwrite('data/c/'+str(cc)+'.jpg', frame)
            if not cont:
                break

        capture.release
        cv.destroyAllWindows()
```

The first thing observed was that the dash camera was tilted and not completely vertical, resulting in non-horizontally aligned images with respect to ground.

To fix the alignment, the video was rotated by an angle of 5 degrees (anticlockwise) using python match and openCV2 libraries.

```
def rotateImage(image, angle):
    h, w = image.shape[:2]
    img_c = (w / 2, h / 2)

    rot = cv.getRotationMatrix2D(img_c, angle, 1)

    rad = math.radians(angle)
    sin = math.sin(rad)
    cos = math.cos(rad)
    b_w = int((h * abs(sin)) + (w * abs(cos)))
    b_h = int((h * abs(cos)) + (w * abs(sin)))

    rot[0, 2] += ((b_w / 2) - img_c[0])
    rot[1, 2] += ((b_h / 2) - img_c[1])

    outImg = cv.warpAffine(image, rot, (b_w, b_h), flags=cv.INTER_LINEAR)
    return outImg
```

This resulted in some black areas around the corners.

To remove the black areas, 20% of images from each edge was cropped using python math and numpy libraries.

The original images were in 4K resolution. This image size is too big to be processed with a neural network in reasonable performance given the processing resources. Therefore the images

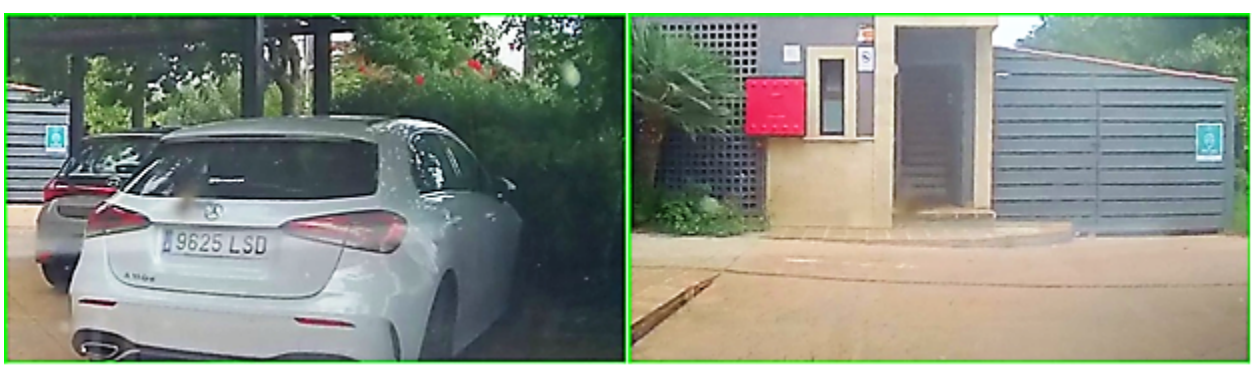
were converted first to 1025 * 600 pixels for labelling and later saved as 360 * 240 pixel dimension for training.

```
def crop_img(img, scale=1.0):
    center_x, center_y = img.shape[1] / 2, img.shape[0] / 2
    width_scaled, height_scaled = img.shape[1] * scale, img.shape[0] * scale
    left_x, right_x = center_x - width_scaled / 2, center_x + width_scaled / 2
    top_y, bottom_y = center_y - height_scaled / 2, center_y + height_scaled / 2
    img_cropped = img[int(top_y):int(bottom_y)-200, int(left_x):int(right_x)]
    return img_cropped
```

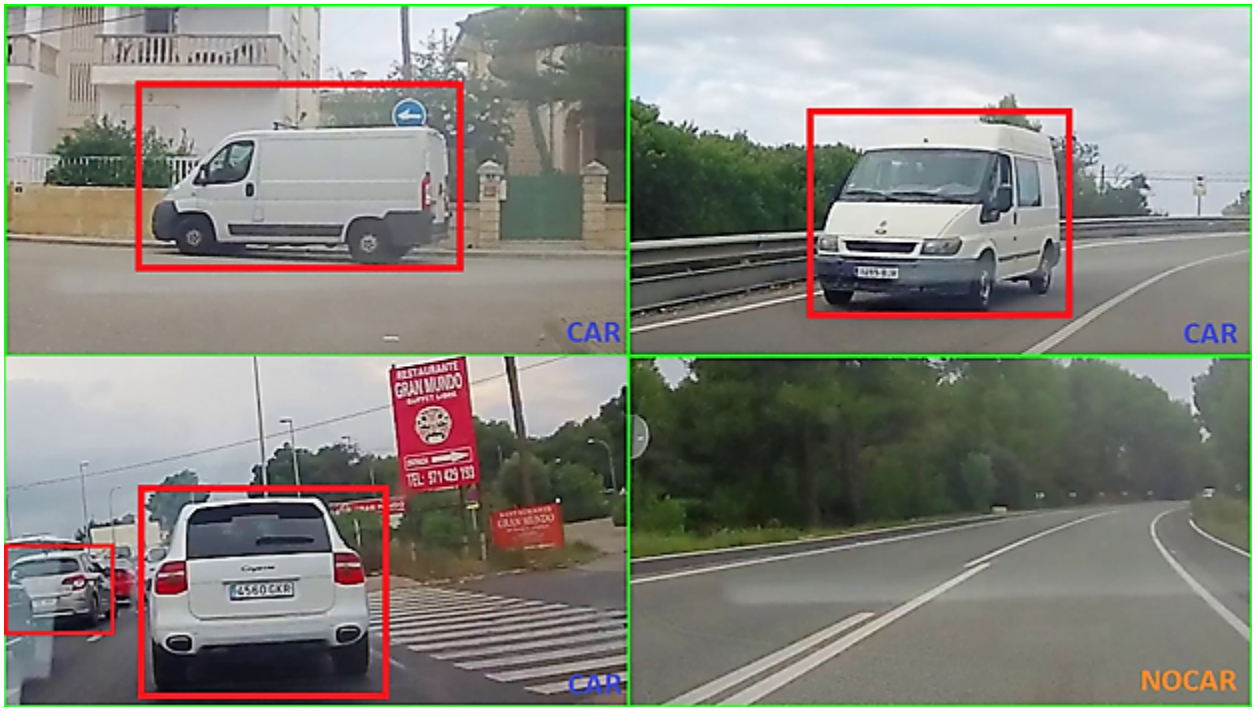
Wherever a car was detected, and bounding was generated to crop and save the image in appropriate dimensions.

Images with no cars were given label 0 while ones with cars were assigned label 1.

Below are examples of an image with label 1 and another with label 0.



Several of such images were labelled with various angles of cars facing the camera.



It is worthwhile mentioning that in addition to manually labelled image data, many car images from Stanford Car Image dataset would be used for training because of their versatility, quality and feature coverage.

10 Conclusions

Autonomous driving systems promise a safe, comfortable and efficient driving experience. However the full potential of such systems cannot be realized unless the robustness of state-of-the-art is improved further.

Acknowledgements

The contents of the report were inspired by the Deep learning specialization on Coursera by Andrew Ng.

References and Bibliography

[1] S. Grigorescu, B. Trasnea, T. Cocias, G. Macesanu, "Ä Survey of Deep Learning Techniques for Autonomous Driving"

[2] H. Yin and C. Berger, "When to Use What Data Set for Your Self-driving Car Algorithm: An Overview of Publicly Available Driving Datasets," in 2017 IEEE 20th Int. Conf. on Intelligent Transportation Systems (ITSC), Oct 2017, pp. 1–8.

[3]] H. C. Burton S., Gauerhof L., "Making the Case for Safety of Machine Learning in Highly Automated Driving," Lecture Notes in Computer Science, vol. 10489, 2017.

[4] K. R. Varshney and H. Alemzadeh, "On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products," Big data, vol. 5, 10 2016.

[5] P. Koopman, "Challenges in Autonomous Vehicle Validation: Keynote Presentation Abstract," in Proceedings of the 1st Int. Workshop on Safe Control of Connected and Autonomous Vehicles, 2017.

[6] Z. Kurd, T. Kelly, and J. Austin, "Developing Artificial Neural Networks for Safety Critical Systems," Neural Computing and Applications, vol. 16, no. 1, pp. 11–19, Jan 2007.

Links:

[7] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115

[8] https://towardsdatascience.com/deep-learning-for-self-driving-cars-7f198ef4cfa2

[9]https://github.com/TheClub4/car-detection-yolov2/blob/master/Car%20detection%20for%20Autonomous%20Driving/Autonomous_driving_application_Car_detection_v3a.ipynb

[10] http://ai.stanford.edu/~jkrause/cars/car_dataset.html