CHALMERS
UNIVERSITY OF TECHNOLOGY

Detection of secondary task engagement in naturalistic driving data

Master's thesis in Automotive Engineering

SRIRANGA HULUKUNTE GOPINATH

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020:83

Detection of secondary task engagement in naturalistic driving data

SRIRANGA HULUKUNTE GOPINATH

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology
Gothenburg, Sweden 2020

Detection of secondary task engagement in naturalistic driving data
SRIRANGA HULUKUNTE GOPINATH

Master's Thesis 2020:83
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Detection of secondary task engagement in naturalistic driving data
SRIRANNGA HULUKUNTE GOPINATH
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Fatalities related to road traffic accidents are up to 25,000 in the EU annually. In most cases, these accidents occur due to human error in judgement or action. Driving requires undivided attention, but studies show that drivers often engage in secondary tasks which result in distraction causing accidents. To analyse this behaviour of drivers, naturalistic driving studies provide video data of drivers indulging in this behaviour.
Meanwhile, progress in computer vision and machine learning has led to algorithms capable of automatically detecting objects in images or videos. Convolutional neural networks (CNNs) are the most common artificial neural network used for such applications. This thesis work focuses on using the latest object detection algorithm named YOLO (You Only Look Once) to detect secondary tasks in images from naturalistic driving data. The algorithm is capable of detecting custom objects provided it is trained for them. The distractions caused due to engagement in secondary tasks were categorised and manually labelled. The data was categorised into 9 types of distractions. The labelled data was trained on a cloud virtual machine.
The results were noted for three different trials, and each trial varied in data size and classes of secondary tasks trained. Trial 3 had the best results with an average detection rate per class of around 88%. This trial was most comprehensive, and it was an iterated improvement based on the observations from previous trials. To make the algorithm robust, it needs to be trained with different datasets to arrive at a generalised model. This work aims to reduce the effort of manually annotating secondary tasks in huge naturalistic driving data.

Keywords: Secondary tasks, CNN, YOLO

Acknowledgements

This work would not have been possible without the constant support and guidance from my supervisor, Thomas Streubel. I would also like to thank the examinar, Marco Dozza for his insights on this work. I would like to extend my regards to division of Vehicle and Traffic safety, Department of Mechanics and Maritime sciences at Chalmers for providing this wonderful opportunity. I would like to extend my gratitude to Machine Intelligence group American University of Cairo (MI-AUC) for providing the data set for this work.

Sriranga hulukunte gopinath, Gothenburg, November 2020

Abbreviations

- ADAS - Advanced Driver Assistance System
- LDW - Lane Departure Warning
- FCW - Frontal Collision Warning
- AEB - Automatic Emergency Braking
- NHTSA - National Highway Traffic Safety Administration
- NDS - Naturalistic Driving Studies
- FOT - Field Operational Test
- CNN - Convolutional Neural Network
- YOLO - You Only Look Once
- GPU - Graphical Processing Unit
- RCNN - Regional Convolutional Neural Network
- SSD - Single Shot Detector

Contents

1
Introduction

The activity of driving has become strenuous, especially in the urban areas considering the increase in the number of vehicles and large traffic densities. For these reasons, the activity of driving requires complete attention. Studies show that in most cases, accidents are caused due to human error in judgement or action [1]. The number of deaths in traffic accidents amounts to around 25,000 annually in the EU alone [2]. Apart from the tragedy to the family of the deceased, it also has an economical impact on the country or state. The cost for accidents in the EU are estimated to 0.4-4.1% of the GDP [3]. These costs can be consolidated into categories starting from medical costs like hospitalization & medical treatments, property damages like vehicle or infrastructure and administrative costs which includes the involvement of police officers, insurance and legal costs.
To reduce accidents or fatal crashes, plenty of measures can be taken. The road and infrastructure design can be improved by mitigating fatalities. Governments or other responsible agencies can promote awareness on road safety and pass regulations promoting the safety of traffic participants. The manufacturers of vehicles play a crucial role as well by developing safer vehicles. Automotive safety is grouped into two major categories focusing on different functions. Passive safety focuses on preventing humans from getting injured in an accident while active safety systems target to avoid or at least mitigate the severity of accidents [4]. Various passive safety systems have helped to prevent injuries in crashes like seat belts, airbags and some advanced restraining systems [5]. An active safety system can be categorized into systems which issue a warning or systems which enable dynamic control of vehicles laterally and longitudinally [5]. Two examples of the former are Lane Departure Warning (LDW) and Frontal Collision Warning (FCW). In a further escalation, some active safety systems intervene when drivers fail to act. One such system is Automatic Emergency Braking (AEB) which detects a possible frontal collision and initiates a full brake to prevent or mitigate the crash. A culmination of all such advanced intervention systems is included in the automated vehicles available today [6]. Developments in machine learning, computer vision and sensor fusion have enabled the automotive industry to apply these technologies in developing systems improving safety [7]. However, a study from AAA Foundation for traffic safety reported that drivers who are familiar with using ADAS (Advanced driver assistance systems) are twice as likely to engage in secondary tasks leading to distracted driving [8].
According to the NHTSA (National Highway Traffic Safety Administration), distracted driving due to engagement in secondary tasks is one of the main reasons for accidents [9]. It defines distraction as any activity where the driver, whose primary task is to drive the vehicle, is involved in secondary tasks like using a mobile phone, eating or drinking and interaction with the infotainment system. The first step in addressing this problem is to raise awareness through campaigns to educate about the risks involved with distracted driving. This can be observed when the number of cell phone users dropped in the USA from 3.3% in 2016 to 2.9% in 2017 [10].

The NHTSA has reported that out of 6,734,000 motor vehicle crashes in 2018, the number of crashes due to distracted driving were 400,000 [9][11]. Meanwhile, the ETSC (European Transport Safety Council) has passed laws which penalize people for using cell phones while driving in many European countries [12]. For improving safe driving, it is of interest for researchers to investigate factors which make drivers indulge in these activities. Some questions like duration of time on cell phones, interaction with infotainment systems and which part of the journey where this behaviour is prevalent like highways or urban roads add more value to the reported numbers of cases. It is important to have driving data where people are showing natural behaviour while driving. This approach provides a more conclusive cause of crashes while indicating the interaction of driver with vehicle and environment.

There are several ways of collecting driving data, e.g. in a Natural Driving Study (NDS), a Field Operational Test (FOT) or a pilot study. This driving data contains a large pool of videos which is relevant to observe the drivers engagement in other activities besides driving. In a NDS, vehicles are equipped with loggers & cameras and provided to study participants to use in their everyday life without any restrictions. A major difference between FOT and NDS is that the latter is observational while the former disclose relations between traffic events, driver behavior and crash causation [13]. FOT data gives valuable information on natural interaction between drivers and systems since drivers show normal behaviour after a short adaptation phase.

Meanwhile, on the technological front, advancements have taken place in computer vision which have enabled technologies like facial recognition, classification and detection of objects. Deep learning or deep neural networks are instrumental in accelerating the above-mentioned technologies. Although deep learning has existed for a long time, they are now capable of performing complex operations due to the computational capabilities of machines today and the amount of data available [14]. Convolutional neural networks (CNNs) are capable of identifying objects with supervised and unsupervised training. There are several open-source algorithms which use different approaches to identify custom objects after training. Among a plethora of object detection algorithms like RCNN (Regional Convolutional neural network) and SSD (Single Shot Detector), YOLO (You only look once) is one such algorithm which is latest to be added to the list. It is unique in its approach to object detection, unlike others mentioned above. It follows a regression-based approach for classification while being computationally light [15].

With the help of object detection algorithms, custom objects in an image can be detected. For CNNs to detect custom objects in an image, it has to train on those objects so that the algorithm identifies a pattern in the image over the course of training. The crucial aspect of training for custom objects is that, this pattern of similarity can also be a behaviour. Therefore, a pattern in different secondary tasks will qualify it to be treated as custom objects. For example, if drinking from a cup has to be detected, then all the images used for training will contain identical posture of drinking. If the data of secondary tasks can be categorized and used for training, the algorithm can detect them. Two sets of data are available for this work. The first set was available on request from Machine Intelligence group at American University of Cairo (MI-AMC) [16]. The second dataset available at SAFER – the Vehicle and Traffic Safety Centre at Chalmers - contains annotated naturalistic driving data of individuals engaged in secondary tasks while driving. This work is intended to utilize the data sets available to train the YOLO algorithm detecting secondary tasks.

1.1 Background

The total number of fatalities due to road traffic accidents in the EU from 2000 to 2016 has decreased by 53 percent [17]. One possible explanation could be due to the improvement in safety features of vehicles sold currently. Another reason could be due to the laws passed by the EU to ensure vehicles are equipped with compulsory safety features like ABS (Anti-locking braking system) among others [18]. Although this downward trend of accidents is promising, distracted driving due to secondary task engagement has claimed the lives of 2,841 people in the USA in 2018. The EU has also identified driving distraction as a major source for accidents [19]. The number of fatalities caused by distracted driving in EU, however, is unavailable for now.

1.1.1 Distracted driving due to secondary task engagement

After understanding the consequences of distracted driving in terms of accidents and fatalities, it is important to clarify the term distraction in the context of driving. Going through well-documented resources on distracted driving, NHTSA's definition provided clarity and classified various categories of distraction [9]. This understanding is important while preparing the dataset. Distractions are categorized into different types such as visual, cognitive, auditory and physical. Any data collected for distracted driving will fall under these categories. A major issue in terms of distraction over the years has been mobile phones. It is interesting to note that using a mobile phone causes distractions of all categories stated above. A report shows a high number of cases of distracted driving as a result of mobile usage [20]. As the possession of mobile phones increased from 250 million units in 2007 to more than 1.5 billion units in 2020, the surge in cases of distracted driving may be a result of this increase [21]. Although possession of phone may not be equivalent to usage in the vehicle, it is likely to be used in the vehicle. Therefore, the dataset which will be prepared should be categorized for all forms of cell phone usage like texting and talking.

1.1.2 Feature extraction and object detection

Before dwelling on identifying distractions from videos or images, it is important to understand some fundamentals of image analysis. The main purpose of image analysis is to identify objects or features in an image. Different applications require different degrees of accuracy, and based on this, the feature is recognized and segmentation is necessary for accurate delineation. After extraction of features/objects, it is the work of a classifier to categorize them. Linear classifiers are the simplest but usually less powerful. Images are treated as vectors, and a linear classifier performs a dot product of the image vector along with a weighted vector so that the resulting dot product if greater than a set threshold value will be classified as the desired feature. This concept can also be applied to an image which contains a lot of features. The classifier will be moved in a sliding window operator across the image to extract features.
Another powerful technique to classify images is the process matching features in images irrespective of scale and orientation called SIFT (Scale Invariant Feature Transform) descriptor [22]. To extract features, the SIFT descriptor identifies blobs or interest points in images. The position and scale of these interest points are determined to match similarities and further classify as features between images. After the evolution

of classifiers which could learn on supervision, the SIFT descriptor is usually used for a specific purpose, and it performs poorly under bad lighting.

1.1.3 Driver distractions identified with an ensemble of CNNs

The Machine Intelligence group at American University of Cairo (AUC) had one of the earliest work in identifying driver distractions, making it a benchmark for future researchers [23]. Their research identified the need to address this problem due to the growing number of people using phones, availability of sophisticated infotainment systems in vehicles and interaction of people with these devices. Subsequently, following this was the progress in object detection using deep learning and convolutional neural networks. The work follows the principle of using several CNNs in tandem to identify distractions precisely. The data used for their research is a custom made dataset inspired by the predefined classes of driving distractions by State Farm's Dataset competition on Kaggle [24]. The dataset contains 10 distractions in a fixed pose which was made publicly available for research on request. The images are segmented to various parts of the body viz; face, hands, skin devices and fed to various CNNs like VGG, AlexNet and Inception. The weights learnt from all the networks are concatenated to an ensemble of classifiers which help decide for the distraction identified in an image. The results of their work showed a classification accuracy of over 90 percent and have provided a baseline accuracy for future researchers to benchmark against. The future work for this research included using a state of the art algorithm at that time named R-CNN (Regional based Convolutional neural network). Since their model was heavy to run in a real-time environment, it would be interesting for future researchers to build models which could handle the real-time implementation.

1.1.4 Driver distractions using machine learning and fuzzy logic

There is a different approach to identify and evaluate distractions [25]. A system is proposed, which has a model for normal driving and secondary task or distraction engagement. The driver performance is evaluated under various circumstances like driving without any distraction and driving with engagement in a secondary task. The vehicle data in both these instances serve as a metric to identify a possible distraction. To evaluate the distraction identified, they use fuzzy logic. To verify the working of the presented approach, a driver in the loop experiment was carried out. The proposed method starts with noting the driver's performance on a known segment of road. Variables like speed and ability to stay on the lane are also calculated. Following this, the driver is engaged in a distraction while driving on the same segment of road. The same variables which were evaluated for normal driving are compared with distracted driving and relation is established between the two. Finally, the fuzzy logic evaluator normalizes these variables and designates a percentage of distraction identified. This approach is different compared to other methods, as other methods require a visual aid to identify distractions and collecting and labelling such data may be extensive and expensive. The approach followed by the researchers here eliminates all such cost and have shown the feasibility of their model to be applied in real-time. However, this work comes with few limitations as mentioned in the paper. It only focuses on one distraction. It also missed a statistical analysis of different categories of drivers based on age, gender and driving experience. To have a robust system, all these limitations have to be addressed.

1.2 Aim and Scope

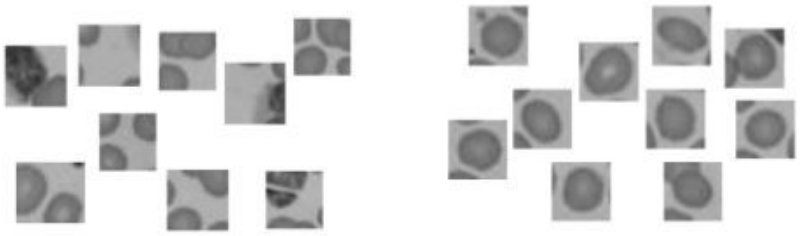
Researchers of traffic safety need to analyze secondary task engagement from the large naturalistic driving database to understand this behaviour. This data contains videos which exceed thousands of hours and manually annotating secondary tasks can be cumbersome. The thesis is intended to utilize the capabilities of image processing and feature extraction of open source deep learning algorithms to identify secondary tasks causing driver distraction automatically. The dataset containing images should be categorized to various secondary tasks like drinking, texting and talking on the phone and labelled accordingly. Through the course of this work, an open-source object detection algorithm is trained on the available dataset. It is of interest to determine the prediction rate of the algorithm considering the level of training it undergoes. The output of training may depend on the quality of data used, duration of training and amount of data. The algorithm is expected to be easy to use and should also accommodate features/functions which could be added for future work.

2

Theory of object recognition

2.1 Linear classifiers

As mentioned in section 1.1.2 that linear classifiers can be used for object detection or feature extraction. It is important to understand that images are vectors of pixels with each pixel indicating the intensity [26]. Since the images are treated as vectors, arithmetic operations can be performed, and this creates the fundamental concept of a linear classifier. For example, if we have to identify between cells in an image or its background, a linear classifier can be used to categorize an input image into one of the classes.



patches of background patches of cell

Figure 2.1: Images of cells [27]

An image is broken down into patches, and a classifier usually moves in a sliding window manner to identify the class. In Fig. 2.1, the patches of cells and background/non-cells are shown. To design a linear classifier which can make this categorization, additional vector ω, which when multiplied by Image (I) should result in a product greater than threshold τ. If the product is lesser than τ, then the patch can be classified as background.

$$I \cdot \omega > \tau \tag{2.1}$$

This technique can be applied to an image of bigger size, wherein the image can be divided into small patches, and weight vector ω can be moved in a sliding window to identify all the patches containing required features. Later, all the output can be combined to obtain an output similar to input image but with all identified class.

2.2 Learning of a classifier

The linear classifier formulated above can also be formulated in an alternative way by introducing an unknown constant instead of the threshold τ :

$$I \cdot \omega + \omega_0 > 0 \tag{2.2}$$

This classifier works by determining good values for ω and ω_0. It can be considered as a precursor to neural networks. Determining best values depends on the detection and classification of data.

2.2.1 Evaluating a classifier

Performance of a classifier can be evaluated based on false classification it makes [28]. Consider the example of cells and non-cells (Fig. 2.1). If the classifier is trained to detect only cells, it will classify everything fed to it as cells even if there are non-cells. Therefore, having good metrics for evaluation is vital. The concepts of precision and recall depict this behaviour.

$$\text{Precision for class A} = \frac{\text{number of examples correctly classified as A}}{\text{number of examples classified as class A}} \tag{2.3}$$

$$\text{Recall for class A} = \frac{\text{number of examples correctly classified as A}}{\text{number of examples from class A}} \tag{2.4}$$

Precision and recall together can be a better evaluator for a classifier while having only one metric can be misleading.

2.2.2 Negative log likelihood

Consider two classifiers shown in Fig. 2.2, where both of them succeed in classifying positive (green dot) and negative (red dot) examples but the classifier on top will probably perform better on a new dataset. This limitation of precision can be addressed by logistic regression. Through logistic regression, we can assign class probabilities to each example, and likelihood can be used to evaluate the performance of the classifier as a whole. A sigmoid function can be one possible way to solve logistic regression.

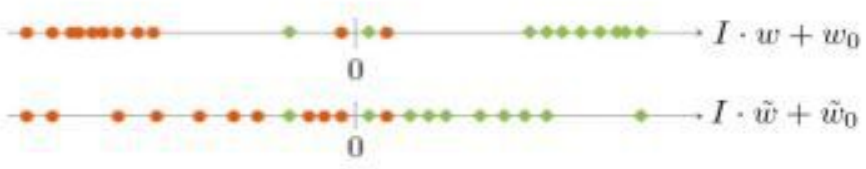


Figure 2.2: Output of two filters [27]

$$p = \frac{e^y}{1 + e^y} with, \mathrm{y} = I \cdot \omega + \omega_0 < 0 \tag{2.5}$$

2.2.3 Softmax function

When there is multiple classes to be detected, the classifier is made in such a way that the class probability for a patch is the summation of all the classes equal to 1. Softmax is

a generalization of logistic regression which helps to achieve a probability of classes to sum to 1. For a set of numbers y_c, the probability p_c such that

$$\sum_j p_j = 1 and y_c > y_j \Leftrightarrow p_c > p_j \tag{2.6}$$

by setting,

$$p_c = \frac{e_c^y}{\sum_j e_j^y} \tag{2.7}$$

The equation 2.7 represents the generalization of logistic regression.

2.2.4 Overfitting

Overfitting occurs when there are large number of parameters to tune, and in case a higher-order polynomial is used to fit all the measurements as shown in Fig. 2.3. In object detection, overfitting occurs when the amount of model parameters is higher when compared to the amount of data available. Hence, to avoid overfitting, more data needs to be included.

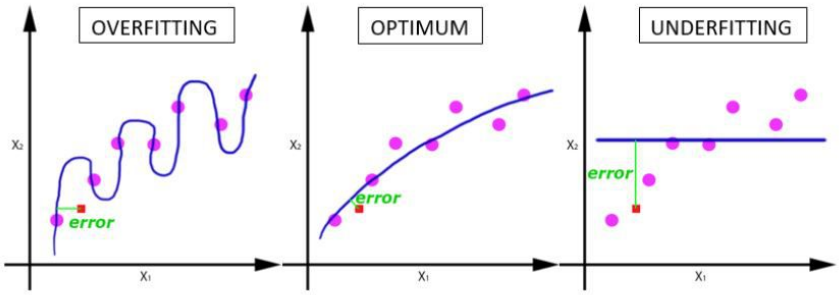


Figure 2.3: Overfitting, optimal fitting and under fitting of measurements[29]

2.2.5 Data expansion/ augmentation

It is essential to have a large labelled dataset for training to avoid overfitting. Labelling data for training can be a painstaking process as it has to be done manually. Therefore, a smarter way to create more data would be to modify the existing data by transformations and thus creating a larger dataset. The data modifications can be done in the following ways.

- Rotational or reflective transform of data
- Moderate change in brightness and scale
- Adding pixel noise

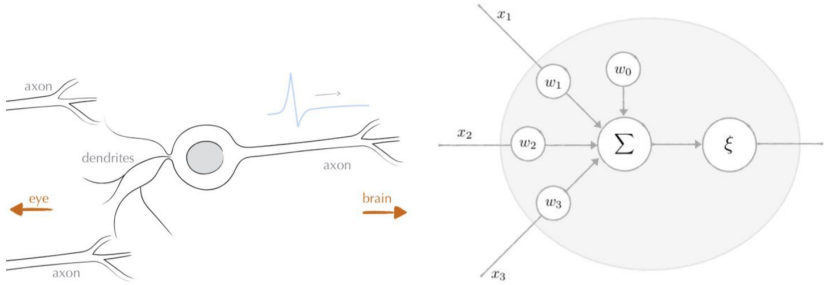
2.2.6 Training, validation and test set

A classifier with large ω value will have a low loss (miss classification rate) [30]. When we have to choose between different classifiers to address the problem, looking at loss functions alone may not be thoroughly justified for its choice. A smaller classifier with a lower

value of ω can still perform better on a new dataset. To avoid this, we create a different validation dataset with positive and negative examples. Based on the performance on a validation set, a classifier can be chosen accordingly.

2.3 Neural network

Neurons are the building blocks of the brain and nervous system. The human visual system is part of the nervous system. It is fast, accurate and precise in classifying various objects. This serves as an inspiration to the system which is based on artificial image analysis. The progress in science and technology has allowed studying the working of neurons. Neural networks use mathematical models which are inspired by them.



A neuron mathematical model of neuron

Figure 2.4: Biological neuron and its mathematical replica [27]

A simple mathematical model is shown in 2.4. The model is fed with inputs x_k while being multiplied by weights w_k. Few mathematical activation functions similar to biological activation include logistic sigmoid, tanh and rectified linear unit(Relu). However, in the model shown in 2.4, it is represented by ξ for simplicity.

2.3.1 General model of a Neural Network

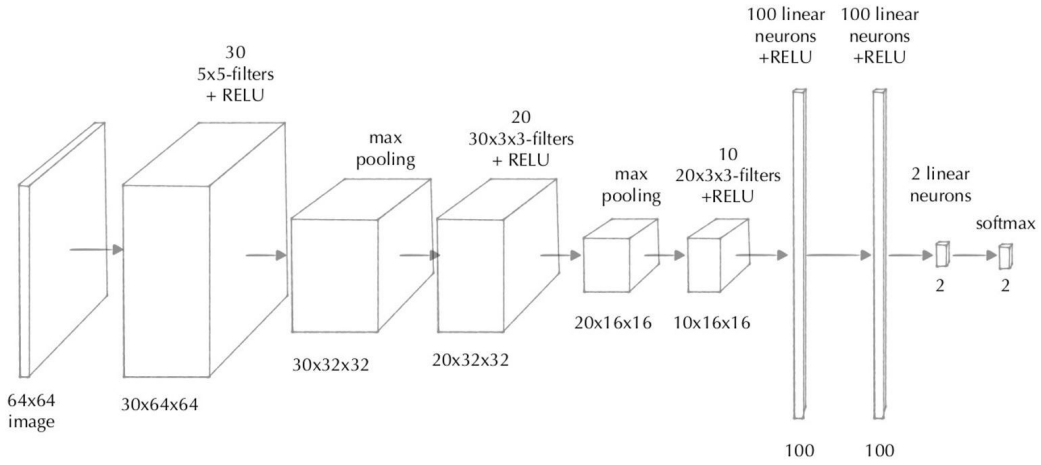


Figure 2.5: Schematic representation of a neural network [27]

A generic model of a neural network is shown in Fig 2.5. The input image is fed to the network with dimensions 64*64. The first layer of the network consists of 30 5*5 filters and a rectified unit layer (ReLU) as an activation function which returns zero if it receives a negative value. The output of this layer consists of 30 channels of 64*64 response maps. This is similar to RGB channels in terms of different channels stacked together, The next layer is called max pooling, where each of the response maps from the previous layer is divided into 2*2 regions which only keep the maximum value of the pixel value as shown in fig 2.6. The output from this layer, however, still consists of 30 channels, but the dimensions are reduced by half.

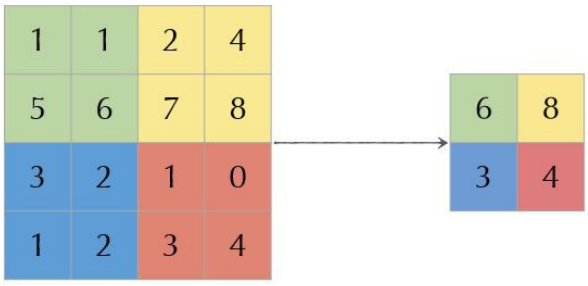


Figure 2.6: Max pooling [27]

The output from max pooling is fed to the third layer, which consists of 20 filters (vectors similar to images) with 3*3 dimensions and a ReLU. It is important to note that each filter colludes with 30 channels from the previous layer. Therefore, the output of this layer has to be known thoroughly. Since every filter f deals with 30 channels, each filter has a tensor of 30*3*3. The k-th filter f_k works on 30 channels, and if $X_k(x, y)$ is the k-th channel in the output from the previous layer, then the filter response will be,

$$\sum_{j}^{30} X_k * f_k \tag{2.8}$$

Moving on, layer 4 and 5 are similar in operation to the layers mentioned previously. But layer 6 takes a fully connected layers approach where 100 neurons compute a linear combination of outputs 10*16*16 from the previous layer. These fully connected layers are the penultimate layers before going to softmax function, where probabilities are estimated to assign a class for detection.

2.3.1.1 Transfer learning

The idea of transfer learning as the name suggests is to use the things learnt from one network in solving a problem and implement on a different network to solve a different problem. Sometimes due to the lack of large datasets, this technique can be used. There are several image databases like ImageNet, which contain a plethora of images that can be utilized for training.

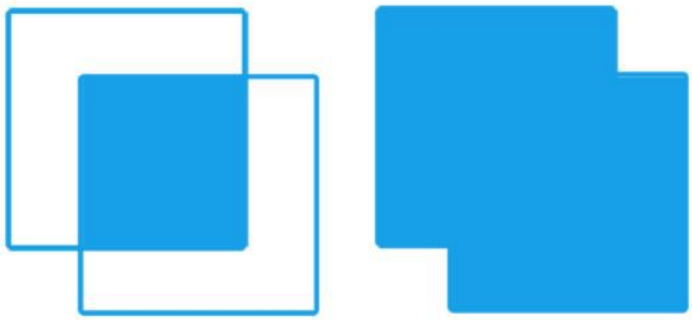
2.3.1.2 Intersection Over Union (IOU)

Objects detected by CNN in an image is usually localized by enclosing a bounding box. IOU is used as an evaluating metric to measure the closeness of this bounding box with

the ground truth, which is the actual location of the object in the image [31].

$$\text{IOU} = \frac{\text{Area of overlap}}{\text{Area of union}} \tag{2.9}$$

IOU is calculated as the area of overlap over the area of union shown in Fig. 2.7. The area of overlap measures the area under intersection between the predicted box and ground truth. Area of union is the total area enclosed by predicted box and ground truth. The ground truth is obtained from labelling the class during training. IOU is expressed in percentage and higher the percentage, the closer is the predicted box to the ground truth.



Area of overlap Area of union

Figure 2.7: Pictorial representation of area of union and area of overlap [32]

2.3.1.3 Batch size, Iterations and epochs

Most of CNNs used for prediction arrive at an optimal solution after going through several iterations. Over the course of training, the algorithms try to minimize the loss. Loss is the error in prediction calculated for every iteration against the actual label in steps. The loss function is controlled by a parameter called the learning rate. Learning rate determines the steps in which the loss function is minimized for each iteration.

- Batch size: It will be difficult for the algorithm to process all the images at once. Therefore, the images are divided into batches and fed accordingly. The batch size refers to the number of images fed to the algorithm at a time. Batch size depends on the memory available for computation. More images can be fit in a batch if the memory is more. The memory here can be from CPU and GPU. Several batches together are termed as number of batches. For example, if there are 100 images in total, which has 10 images in a batch, the number of batches is 10.
- Epochs: One epoch means all the images are passed forward and backward through all the layers of the network. For a large database with several thousands of images, having only one epoch will result in under fitting shown in Fig. 2.3. As the number of epochs increases, the weight is changed, and this results in under fitting to overfitting the data. There is no set number of epochs to have the optimal fitting of data as it depends on the problem.
- Iterations : Iterations depend on the above-mentioned terms. The number of batches required to complete one epoch. Assume we have a dataset containing 1000 images and number of batches is 100 which means each batch contains 10 images. Therefore,

it takes 10 iterations to complete one epoch. To further simplify, the formula mentioned below will help in putting all the terms in context.

$$\text{Number of batches (N)} = \frac{\text{Total number of images}}{\text{Batch size}} \tag{2.10}$$

$$\text{Iterations (I)} = \frac{\text{Total number of images}}{\text{Number of batches(N)}} \tag{2.11}$$

$$1 \text{ epoch} = \text{Number of batches (N)} * \text{Iterations (I)} \tag{2.12}$$

2.3.2 Training on CPU (Central Processing Unit) vs GPU (Graphical Processing Unit)

Both GPU and CPU are microprocessing units designed for performing computations. CPU is capable of performing different types of computations sequentially, whereas GPU is capable of performing similar tasks parallelly [33]. It is difficult to conclude if GPUs are better than CPUs as they work on different principles. But for machine learning tasks, especially for image processing, GPUs proved to be more suitable. This is because GPUs are capable of performing matrix multiplications of image vectors parallel faster, better memory bandwidth and having large and fast registers capable of processing faster than CPU [33]. The Fig. 2.8 shows the performance of GPUs for deep learning tasks over the last 3 years. For approximation, an AMD CPU takes 30 times more time than an NVIDIA GPU for training a CNN.

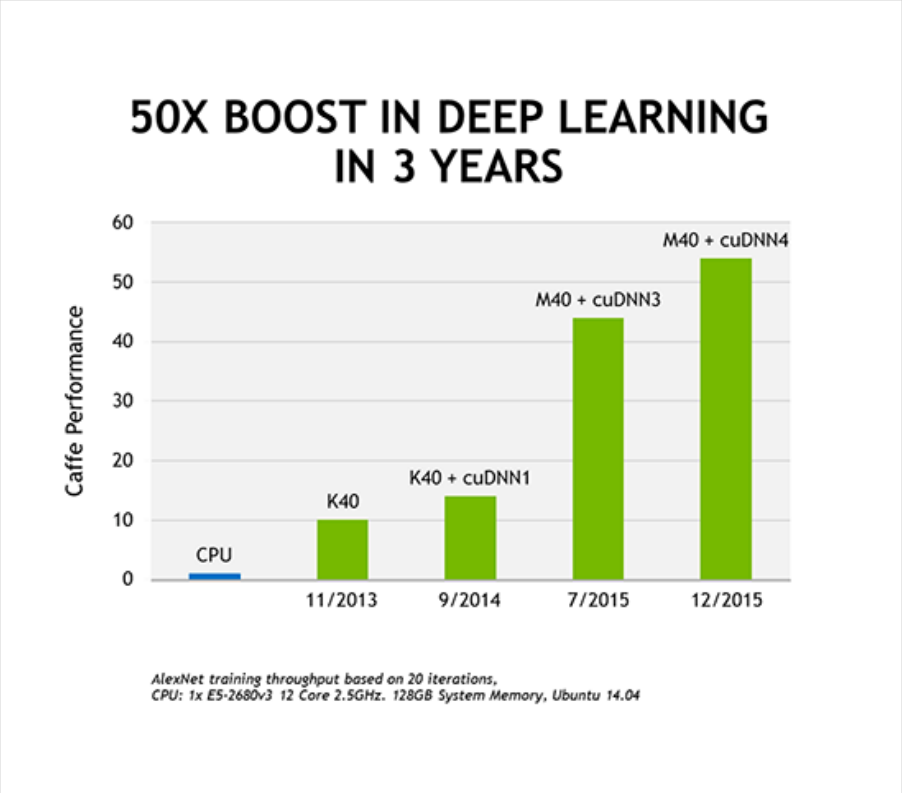


Figure 2.8: Performance of GPU compared to CPU [34]

2.4 Object detection history

2.4.1 Viola Jones algorithm

In 2001, Paul Viola and Michael Jones created the first efficient facial detection algorithm [35]. This algorithm worked on the principle of hand-coding features of faces and feeding them into a classifier. This algorithm was the first important breakthrough in computer vision. It was operational on a webcam which gave it an edge over other facial detection available at that time.
For every image fed to the algorithm, it would downsample them to different hand-coded features. These features extracted from various filters were later concatenated to fed to a linear classifier which would categorize the input image into a binary classification. The main drawback of this algorithm was its inability to identify different poses. Even though object detection improved after this, the idea was groundbreaking at that time and was implemented into OpenCV as well.

2.4.2 Histogram of Oriented Gradients

Another technique of object detection came in the year 2005 and is called histograms of oriented gradients. This technique still involved hand-coding features but with a different approach. The idea was to measure the intensity of each pixel to the surrounding pixels. Later on, the gradient is pointed in the direction of the surrounding pixel with the highest intensity. For each pixel, the gradients are calculated and placed in different bins of varied intensity, as shown in Fig. 2.9. After obtaining histogram bins for each pixel, it is replaced by a single gradient direction which is more prevalent than all other gradient directions. Finally, the image with pixels is replaced by gradients for the entire image. After obtaining this, it can be compared with images to measure similarity and to help in classification.

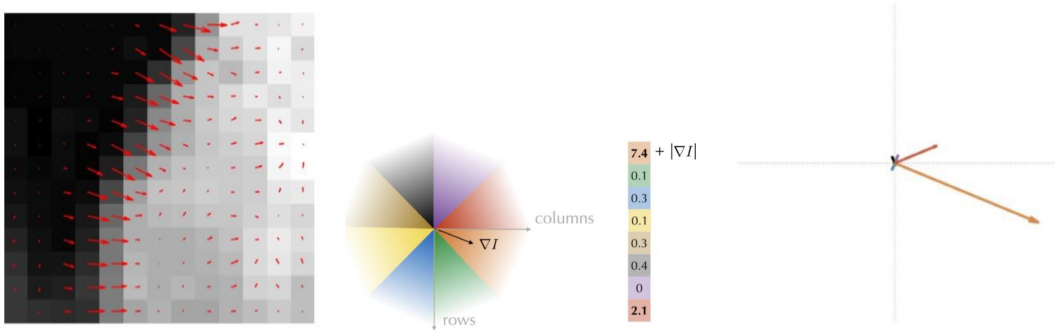


Image with gradients histogram bins Resulting histogram

Figure 2.9: Example of histogram of gradient descent [27]

2.4.3 Beginning of deep learning era

ImageNet organizes yearly competitions for researchers to check their detection algorithms to see which is the best performing algorithm for every year. During 2012, ImageNet's

large scale visual recognition challenge held its annual competition where Alex Krizhevsky, along with his team, tested their convolutional neural network. It outperformed every other network in the competition. This was one of the major turning points in the era of deep learning. Even though convolutional neural networks have existed for decades, their usage was limited for two major reasons. The implementation of CNNs was restricted by the hardware available for computation and amount of data available for training [36]. The advancements in the development of hardware, especially with the graphical processing units (GPUs) has increased the computational capabilities. This was also followed by the availability of large amounts of images online, which helped in the pursuit of collecting images in a large scale hierarchical databases like ImageNet [37].

2.4.4 Convolutional neural networks for detection

CNNs, along with other linear classifiers, can be used to classify features or objects in an image. But, just a binary classification of whether a particular object is present in an image is insufficient unless it is localized. It would be great if the object is detected in an image and also a bounding box is placed enclosing the object. Fortunately, research on this area came to the conclusion that object detection is very much possible using CNNs. The idea behind using CNNs for object detection started off with utilizing existing convolutional neural networks like VGG (Visual Geometry Group) or Inception which have already been trained on several images to localize objects in an image. If a CNN has to be used to find objects in images as seen in Fig. 2.10, the simplest approach would be to divide the image into smaller patches and slide a small window across the whole image to detect various classes of objects. At each step, we can predict the class of object detected inside the window. Although this process seems easy, it can be computational heavy. Therefore, a smarter way to detect objects is needed to reduce the computational load.

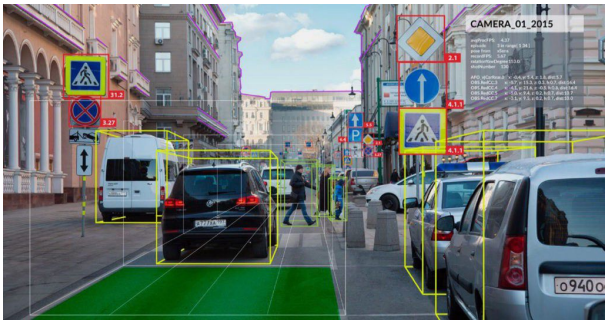


Figure 2.10: Detection of objects using CNN [38]

2.4.5 R-CNN

A better approach for object detection was found in 2015, which used a regional-based localization of objects in an image called R-CNN [39]. R-CNN uses a process called selective search or region selection where an image before being fed to a CNN is segmented for various features or objects. The selective search or regional search looks for objects with an image by placing various bounding boxes of different size and shape. For each of these boxes placed, the adjacent pixels are compared in terms of intensity and colour to find objects. Fig. 2.11 below shows the process involved in detection using R-CNN.

After the image is broken down into selective regions where there is a high probability of objects, these segments of images are fed to a CNN. The output of CNN is further fed to a classifier to detect an object. Finally, these objects are checked by a linear regression model to output tighter co-ordinates into the bounding boxes of the objects detected. This approach at the time proved to be the most efficient way to solve the object detection conundrum, and several improvements were made to this approach. Improvements of the same approach like Fast R-CNN, Faster R-CNN and Masked R-CNN were released the following years, and all of them score high precision scores on various large datasets.

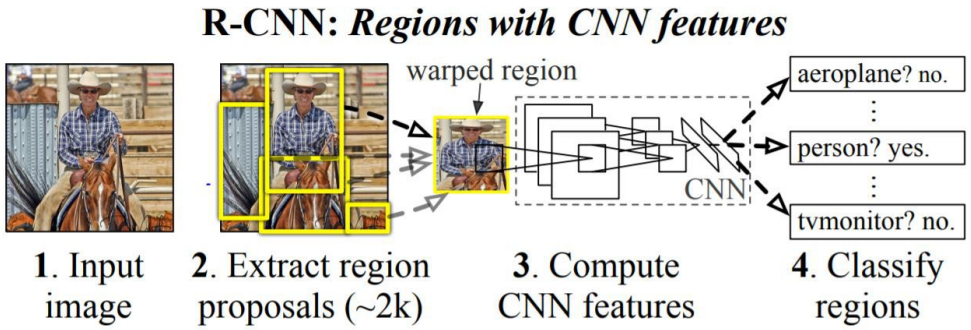


Figure 2.11: Detection of objects using R-CNN [39]

3
Methodology

To recall, the identification of driving distraction as result of secondary task engagement using a convolutional neural network is possible considering the level of accuracy and computational capabilities of hardware available at present to handle the data. This work focuses on identifying the distraction as a holistic based approach where it is labelled manually and trained for the same using the YOLO algorithm. To simplify, distractions are treated as custom objects to be detected in an image. However, this work does not follow a segmented approach, where distractions are broken down to gestures and objects like phone and cups for prediction. But, the framework provided here can be extended to accommodate a segmented approach for identifying distractions. Another important aspect is to implement a state of the art image detection algorithm to solve our problem, which has not been done for this problem yet. Its creators [40] have well documented the prowess of YOLO in terms of its speed and accuracy. Some distractions have a distinct pattern which makes the detection of it as a whole more suitable. For example, all the drinking distractions involve the same gesture where the driver has one hand on the steering wheel while the other hand is occupied in holding the drink to the mouth.
Python contains several open source libraries available for computer vision and deep learning. Some of the main libraries which are crucial in executing this project are OpenCV, Tensorflow and Numpy. However, some of libraries come built-in with Python like math, which contains basic operations involving constants.
The OpenCV library consists of functions capable of working with images or videos in terms of feature extraction, image transformation and image filtering [41]. It is an open source item, which makes the ease of usage hassle free and has lot of tutorials available online to get familiar with functions. Some of the vital functions which is crucial for this work may include, reading the neural network, extracting features and applying bounding boxes and adjusting the shape of the image.
Tensorflow is an open source library which basically is capable of machine learning application like neural network. It is originally developed as part of Google brain project [42]. Numpy is a library capable of handling high performance multidimensional arrays and functions capable of operating these arrays [43]. Matplotlib is a plotting library which also an extension of Numpy [44].

3.1 Data Procurement

The fundamental idea of using CNN is to recognise a pattern from data instead of tedious programming. Therefore, for the algorithm to perform well, we need a lot of training data. The dataset used for the algorithm was obtained on request from [23]. The pattern observed by the algorithm here for the distractions could be transferred when training for a new dataset. The data created by American University in Cairo (AUC) was inspired by the StateFarm's dataset, which was one of the earliest available datasets to classify various distraction features. However, this dataset could be used for competition purpose the only [24]. Hence the new dataset created by AUC had all the enlisted classes of distraction used in the StateFarm competition. This dataset was provided from AUC for academical use. There are some limitations to their data wherein naturalistic behaviour is not followed. Some of the data collected are premeditated as the subjects involved in the study are told of distractions to perform while being recorded.

3.1.1 Ethics of assessment

Since the project involves personal identity of participants and their behaviour collected as part of naturalistic driving study, it is the responsibility of researchers to protect these data and ensure the identity of participants is kept confidential. The data containing videos is property of SAFER. Only authorised individuals have access to the data. The FOT rooms do not have access to internet and thus ensures data is not shared online for any purpose. The researchers at SAFER have pledged to the keep the privacy of participants.

3.1.2 Labelling

The algorithm needs labelled data for training. The labels in images represent the class of the object which in this case is the distraction and allows the algorithm to learn those features. To learn from the images, the position or coordinates are marked. There are several software tools available to perform labelling. The labelled images with coordinates are saved as a text file along with the image dataset. This is done so that algorithm will only learn from specified coordinates corresponding to the image. LabelImg was used to perform this task. It is simple to use and available for all operating systems. This software allows to label the data for both YOLO and pascal VOC data formats. Fig.3.1 shows the interface of the tool in which the labelling is done. The distraction is enclosed in a green box, and subsequently, the class is labelled as drinking. This procedure can be tedious as the user should label all the classes of images used for training. However, Google has labelled images available for objects in the COCO (Common Objects in Context) dataset, and this contains around 80 classes of general objects. Since our dataset requires labelling of distractions, this has to be done manually. As the labelling is done manually, it can be cumbersome for large datasets. It will be interesting to have methods which could make the process of customised labelling automatic.

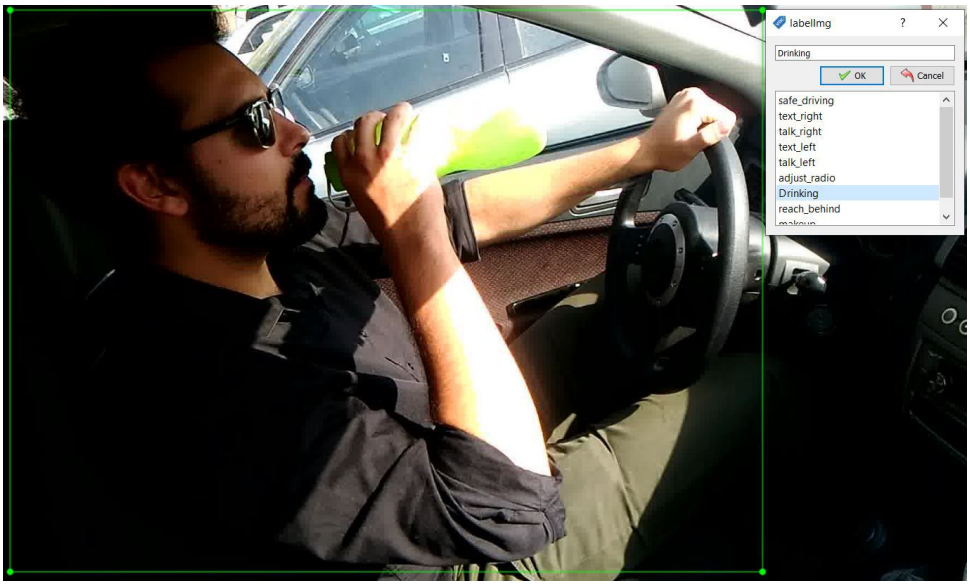


Figure 3.1: Labelling distraction in the tool

3.1.3 Data cleaning and segregation

The distracted driver dataset contains mixture of images which has to be arranged according to the various types of distractions. The data available can be classified as following distractions viz; safe driving, text left, text right, talk left, talk right, adjust radio, reach behind, drinking, hair and make up.

Table 3.1: Classes of distractions

Classes	Description
c0	safe driving
c1	talk left (phone)
c2	talk right (phone)
c3	text left (phone)
c4	text right (phone)
c5	adjust radio
c6	reach behind
c7	hair and makeup
c8	drinking

The Table 4.3 shows different classes of distractions captured in images available for training. These classes were defined in StateFarm dataset for the Kaggle competition [24]. Safe driving here means when both hands were on the steering wheel, whereas every other class involves driver having only one hand on the steering wheel and the other hand is used for secondary task engagement. The same classes were used for the preparation of the dataset except for one class. This class contained images of talking to passengers. The quality of the image was poor and conflicted with safe driving due to its similarities in posture. This needed a segmentation of the distraction instead of holistically labelling as talking to passengers. For these reasons, it is excluded from training.

3.2 YOLO (You Only Look Once) algorithm

As the name suggests, the YOLO algorithm scans through the image once and generates a bunch of confidence scores for different classes of objects. YOLO is written on a darknet framework which is a neural network written in C and CUDA. Darknet is known for its simplicity in installation and usage. It is also made to support both CPU and GPU for computation [45].

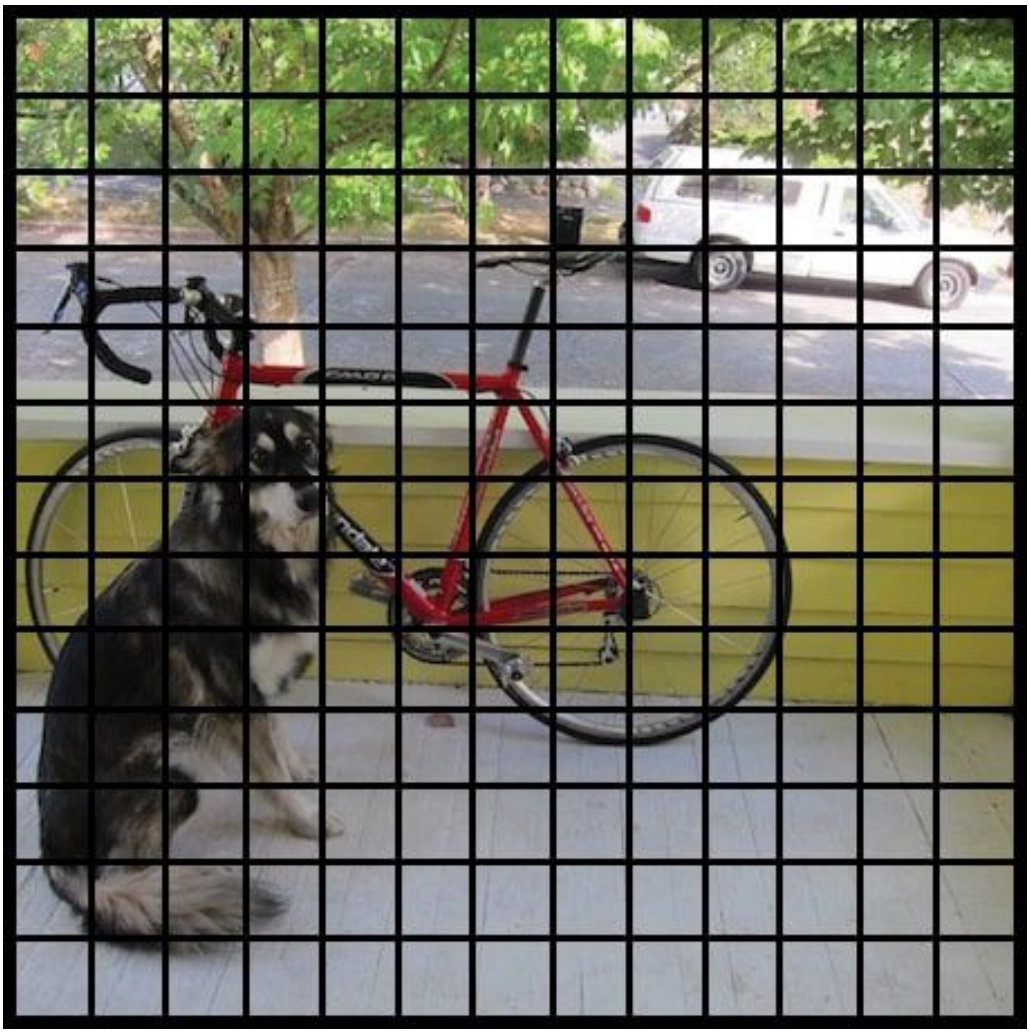


Figure 3.2: Image divided into 13*13 grid by YOLO [40]

YOLO performs differently compared to other object detection systems. It does not follow the traditional approach of training a classifier to scan through the image in a sliding window fashion to detect different classes of object. Instead, YOLO divides an image into small patches or grids, as shown in Fig. 3.2. In each of these grids, the algorithm applies bounding boxes which basically enclose an object found in the image. The bounding boxes are of various size, considering that objects come in different shape and aspect ratio. The bounding boxes, however, are not limited to the size of the grid. After all the bounding boxes are applied, it will look similar to Fig. 3.4. While it draws five bounding boxes for each of these grids, it simultaneously also predicts the confidence score of an object being found in that box. These objects are further associated with different classes of objects. The novelty with YOLO is that all the above-mentioned tasks are performed

simultaneously by a single convolutional neural network which has not been done yet.

Figure 3.3: Bounding boxes for predicted class [40]

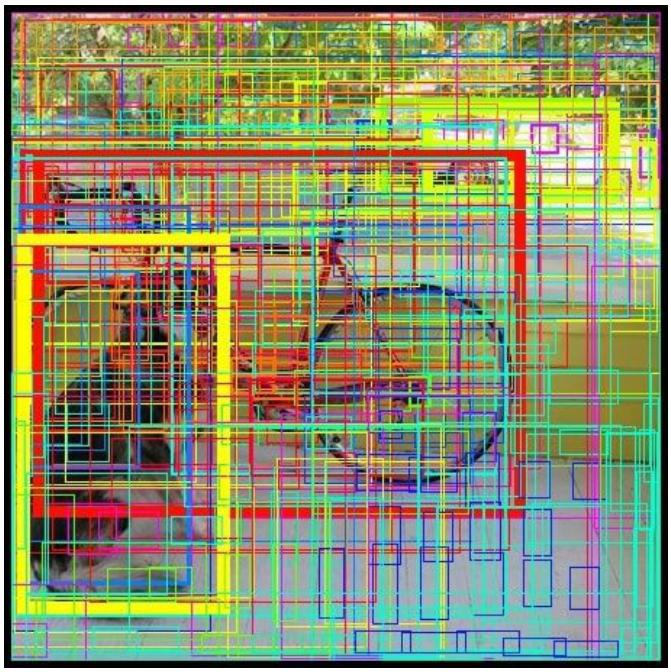


Figure 3.4: confidence scores for various classes detected [40]

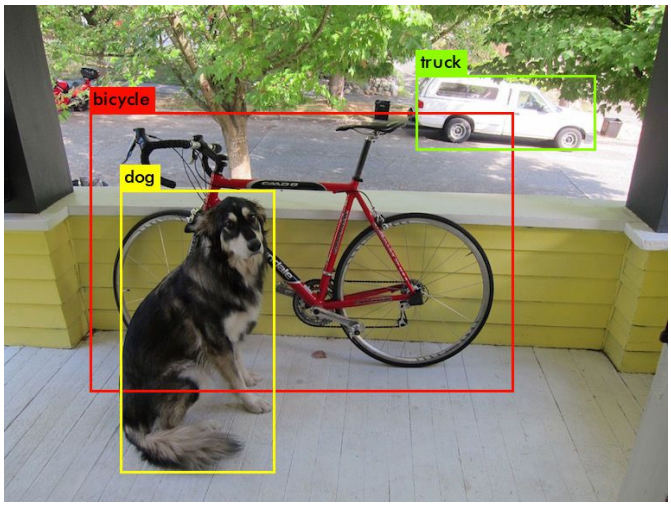


Figure 3.5: Final detection of objects by YOLO [40]

The thicker the border of the box, stronger is the confidence score for an object. YOLO was trained on PASCAL VOC (Visual Object Classes) dataset which contained around 20 different classes of data, namely car, dog, person, cat etc. As seen in Fig. 3.3, the thickest box has the best probability of finding an object. The final detection is based on combining the confidence score of the most significant bounding box and the class probability of an object. As referred from Fig. 3.2, there are 13*13 grids which means that the image has 169 grids in total. Since YOLO drops 5 bounding boxes per grid, a total of 845 bounding boxes are predicted for the whole image. Most of the boxes probably will have low confidence score as seen from thin boxes in the bottom right corner of the Fig. 3.4, only the ones with a confidence score of 30 percent or more are kept. The confidence score is the product of the probability of object found and IOU (Intersection Over Union). The confidence score will be zero for boxes which do not enclose any object. For possible objects in the image, the confidence score depends on the IOU. The final detection for the image is shown in Fig. 3.5.

$$\text{Confidence score} = \Pr(\text{object}) * \text{IOU (predicted)} \tag{3.1}$$

3.2.1 YOLO Architecture

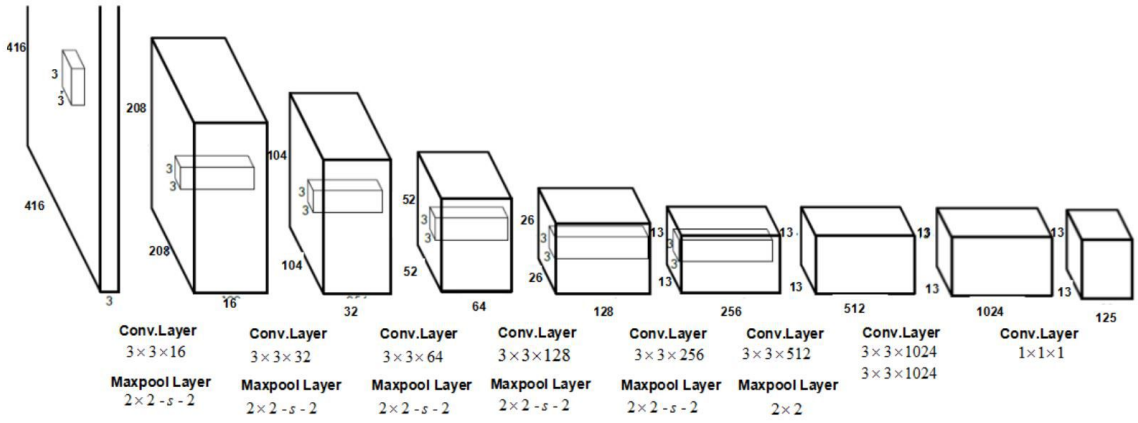


Figure 3.6: A graphical representation of architecture of YOLO [40]

The YOLO algorithm is a convolutional neural network. The creators have released several versions of the same algorithm with incremental changes in the architecture, although keeping the fundamental approach of treating detection as a regression problem [15]. The architecture of YOLO as seen from the Fig. 3.7 consists of series of convolutional layers and max pooling. The kernels are filters which are responsible for extracting features in every convolutional layer. It can be seen from Fig. 3.6 that the downsampling of images occurs after passing through several convolutional layers. The strides indicate the shift in a filter to pixels. When stride is 1, the filter is shifted by 1 pixel, and it is shifted by 2 pixels for stride value of 2. Filters also facilitate feature extracting which is max pooled for dominant gradients.

It can be seen from Fig. 3.6 that the input image has the size of 416*416*3. Here, the height and width of the image are 416, and 3 channels represent RGB (Red, green and blue). Consider that there are 20 classes of objects to be detected. The bounding boxes after the initial scan from the algorithm, as seen in Fig. 3.3 is fed to the algorithm. The output after going through several convolutions is 13*13*125. For simplicity, it can be understood as 125 stacks of 13*13 size extracted features. This output shape is for every grid or patch. The number of channels at the output layer depends on the number of bounding boxes and the data collected for each of the bounding boxes. The data elements for each box is represented in the table below. It consists of the position of the box in terms of co-ordinates x & y, width and height of the box, object score (confidence score) and class probabilities for all the classes in the dataset. Here, the number of classes of the dataset is 20, but this can change for every problem. This amounts to 25 data elements for each bounding box. Since YOLO registers 5 boxes per grid, the output tensor is 13*13*5*25 which can be re-written as 13*13*125.

Box pos.(x)	Box pos.(y)	height of box(h)	width of box(w)	object score	class 1	..	class N

```
Layer         kernel  stride  output shape
-------------------------------------------
Input                         (416, 416, 3)
Convolution    3×3      1     (416, 416, 16)
MaxPooling     2×2      2     (208, 208, 16)
Convolution    3×3      1     (208, 208, 32)
MaxPooling     2×2      2     (104, 104, 32)
Convolution    3×3      1     (104, 104, 64)
MaxPooling     2×2      2     (52, 52, 64)
Convolution    3×3      1     (52, 52, 128)
MaxPooling     2×2      2     (26, 26, 128)
Convolution    3×3      1     (26, 26, 256)
MaxPooling     2×2      2     (13, 13, 256)
Convolution    3×3      1     (13, 13, 512)
MaxPooling     2×2      1     (13, 13, 512)
Convolution    3×3      1     (13, 13, 1024)
Convolution    3×3      1     (13, 13, 1024)
Convolution    1×1      1     (13, 13, 125)
-------------------------------------------
```

Figure 3.7: Architecture of YOLO [40]

3.2.2 YOLO improvements

YOLO versions			
YOLOv1	YOLOv2	YOLOv3	YOLOv4
Object detection is solved by the approach of treating it as a regression problem instead of classification problem. It looks at entire image globally instead of locally using a CNN	YOLOv2 matched mean average precision (MAP) with SSD and Faster RCNN	Incremental inputs with respect to bounding boxes, better MAP scores compared to RetinaNet and 3.8 times faster [15]	YOLOv4 boasts of improved speed and accuracy compared to YOLOv3. The MAP & FPS have increased 10 percent and 12 percent respectively [46].

3.3 Training dataset on cloud virtual machine

Google Colab offers a free cloud service for machine learning and artificial intelligence research. To get started, it is necessary to have a Google account to have access to a GPU. The training is explained in steps starting from cloning the necessary repositories to training a custom dataset.

Step 1 : Enable GPU acceleration with notebook

An IPYNB file is a notebook document which enables to code with python language and its data [47]. The notebook document is uploaded to the Google Colab. Before using, the GPU should be enabled, which allows the user to connect to the cloud server of Google,

thereby allowing the usage of cloud GPU.

Step 2: Cloning and building Darknet

The darknet is cloned from AlexyAB's repository [45]. The YOLO algorithm is built on Darknet. Hence to run the training, the necessary dependencies of darknet should be enabled. OpenCV and CUDA are the main libraries which can be sourced directly from Git.

Step 3: Download YOLO pre-trained weights

Convolutional neural networks are made of several layers of neurons. The initial layers are capable of identifying primitive features like lines and circles, and as the layers go deeper, the networks extract more complex features [48] [49]. The final layers of the network after training are responsible for the detection of various classes of data like a cat, dog, person and more. The purpose of using pre-trained weights is to avoid training from the beginning for every problem but instead, enable transfer learning which is an approach in machine learning where a new problem is solved using the knowledge gained from previous ones. As distractions are treated as custom objects here, throughout the training process, the final layers will be updated with the pattern observed from the distractions. YOLO is pre-trained for 80 classes on the COCO dataset [40]. Among these, classes like a person, mobile phone and cups or bottles will from pre-trained weights will aid in training. The YOLO pre-trained weights are available on the official site. It can also be pulled from git.

Step 4: Connect Google Drive to Google Colab

The advantage of using Google cloud service is the connectivity provided among all the Google applications. The images which are required for training are uploaded to the google drive, which can be accessed by the Google Colab on request. However, the images can also be directly uploaded from the local machine onto the Google Colab.

Step 5: Gathering and labelling a custom dataset

As mentioned previously, the images are manually labelled using any third party labelling software available. The labelling is done based on an interesting point. In this case, the labelling is done for particular distractions as mentioned in 4.3. The coordinates of the bounding box are saved as a text file in the same folder as the images. The folder should also contain a text file named "classes.txt" indicating all the classes of images labelled.

Step 6: Moving the dataset to cloud virtual machine

All the images, along with the text files indicating the coordinates for the labelled features, are uploaded to the cloud virtual machine. The data is compressed beforehand to reduce the time needed for the upload. Later on, the files are decompressed to work with on the cloud virtual machine.

Step 7: Configuration files for training

To have flexibility and reusability in solving machine learning problems, it is necessary to have a configuration file which can make the work easier. Configuration files are necessary to run the model with the settings user needs to achieve the solution. The configuration file for YOLO can again be picked from the official site. This file contains the information of architecture of the algorithm and certain parameters which can be changed based on the user inputs.

Step 7.1 : Parameters to change in the configuration file

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches =18000
policy=steps
steps=14400,16200
scales=.1,.1
```

The configuration file contains a large set of parameters as shown above and begins with batch and subdivisions. Batch refers to the number of images used in each iteration of training. The batch size is set to 64 in the highlighted text code above, which means that each iteration of training uses 64 images for learning. Batches are further divided mini-batches or subdivisions which are processed parallelly by the GPU. For the above file, the subdivision is set to 16, which implies that 4 cycles of subdivision complete a batch of images. If the GPUs have higher memory, the subdivisions can be set to lower values. The batches and subdivisions are set to the highlighted values during training, and while testing, both of them are set to 1 as only 1 image can be tested at a time. In case of issues during training which often pertains to the lack of GPU memory, the batch can be set to 32 or 16.
The width and height of the image as input is 416*416, but any other size of the image used will be resized by the algorithm automatically for all the 3 channels (Red, green and blue). Momentum and decay are optimizers. Momentum refers to the factor by which the successive weights are updated. Decay is set low to minimize the weights of features which are less significant. The angle, saturation, hue and exposure are related to augmentation of the images. The images are rotated randomly during training. The algorithm also changes the saturation, exposure and hue of the image randomly over the course of training. These parameters ensure generalization of data. Learning rate is the response to the error for change in weights. Keeping the learning rate low increases the training process but ensures better training. The parameter burn in corresponds to a slow increase in the learning rate for the initial set of batches until the set learning rate. Here, the learning rate is set as 0.001, and burn in ensures that for 1000 batches, the learning is

lower than the set value. The maximum batches are the product of 2000*(number of classes), and training will be processed for this number of iterations. In contrast, steps are 80 and 90 percent of maximum batches respectively to adjust learning rate.

Step 7.2 : Changes in YOLO and convolutional layers

```
[yolo]
mask = 6,7,8
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90
classes=9
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

The YOLO layers are final detection layers in the algorithm. Mask refers to the indexes of bounding boxes and anchors refer to the various size of bounding boxes referred in this chapter which are adjusted throughout the training. The classes are set to 9 which represents the number of distractions. Jitter crops and resizes images specifically for the YOLO layer. Meanwhile ignore thresh and truth thresh adjusts duplicate detections. There are 3 YOLO layers present in the architecture and the highlighted changes have to be made for all of them.

```
[convolutional]
size=1
stride=1
pad=1
filters=42
activation=linear
```

All the convolutional layers of the algorithm which contains the data of filters have to be changed. YOLO algorithm has multiple convolutional layers, refer Fig. 3.7 The number of filters have to be changed according to the number of classes. The formula for number of filters goes by..

$$\text{Filters} = (\text{number of classes} + 5) * 3.$$

Step 7.3 : Additional configuration files

```
safe_driving
text_right
talk_right
text_left
talk_left
adjust_radio
Drinking
reach_behind
makeup
```

Two more text files have to be created. A (.names) text file as shown above should contain all the classes of data similar to the text file in the folder of images. A (.data) text file as shown below should contain the number of classes and backup path where the trained weights are stored after every thousand iterations. One last step before training is to create a text file which holds the relative path to all the training images. This completes all the necessary setup to start the training process.

```
classes =9
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/yolov3/backup/
```

Step 8 : Downloading some more pre-trained weights for the convolutional layers

This step is not mandatory, but doing so will definitely help the training and model to converge while being accurate. Google colab allows using the GPU continuously for about 12 hours. For every 3000 iterations, the weights are saved in Google Drive. The learning rate can also be seen as the training progresses.

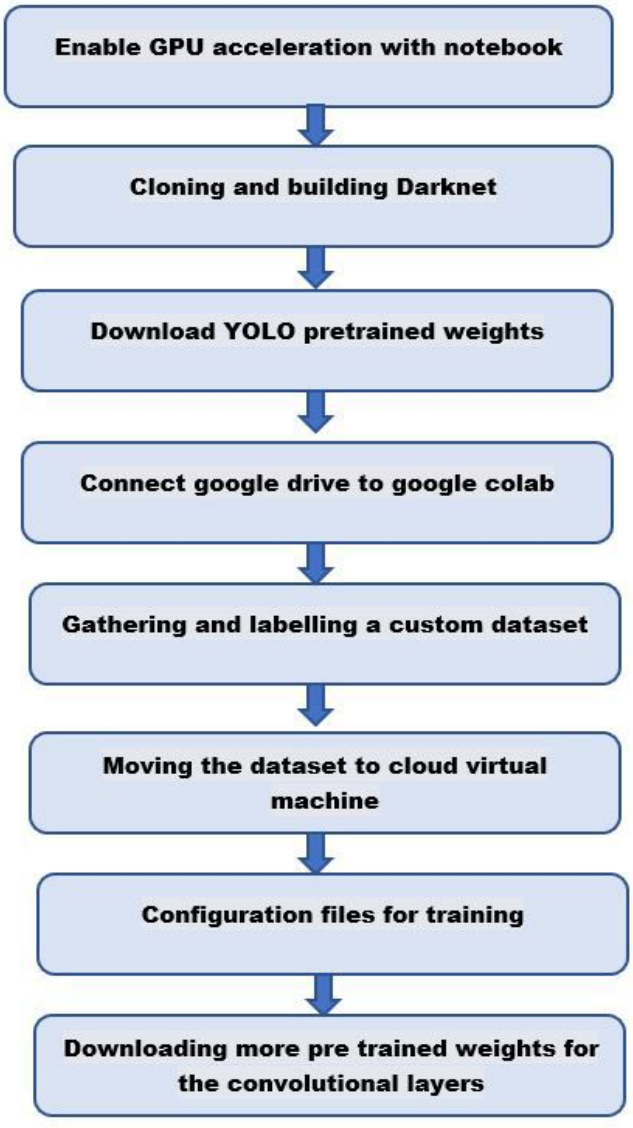


Figure 3.8: Flow chart of training process setup

3.4 Training dataset on local machine

The dataset can also be trained on a local machine. It is necessary to fulfil certain hardware and software requirements for a smooth operation. The steps for the training setup are methodically explained below.

Step 1: Procuring darknet YOLO is originally built on the framework of darknet [45]. Darknet provides YOLO with the capabilities to object detection. Therefore, it is necessary to install darknet on the local machine. Running it on Windows may not be the most optimal way as the creators have written in C. However, the creator named ALexyAB has modified darknet to work on Windows or Linux efficiently. This version of darknet can be pulled from git and should be installed on the local machine.

Step 2: CUDA and OPENCV installation CUDA (Compute Unified Device Architecture) enables the use of a GPU as a supplementary source in processing the computation needed for training and running YOLO algorithm. Before downloading CUDA, it is important to check if the local machine is enabled with GPU like Nvidia. The training can also be performed on the CPU, but having both CPU and GPU working together will accelerate the process and makes it faster. Download the correct version of CUDA for the GPU available on the local machine. OpenCV, as mentioned before, is a machine learning framework for computer vision applications and allows you to see the object detection real-time. Any version above 4.0 can be installed, and the exact version is given in the appendix.

Step 3: Downloading weights file The weight file allows for solving a new problem based on learning from the previous problem. YOLO is pre-trained for classes in COCO dataset [40]. The weight file can be downloaded from the official website of YOLO [50].

Step 4 : Configuration file Once the darknet is built on the machine, prepare the configuration file the same as shown in the previous section. YOLO detection can also be used on real-time detection.

Step 5: Training and grabbing final weights Initialize training after following all the above steps and download the trained weights file and configuration file after completion of training. Training can be stopped based on the desired loss. This is used to create object detection.

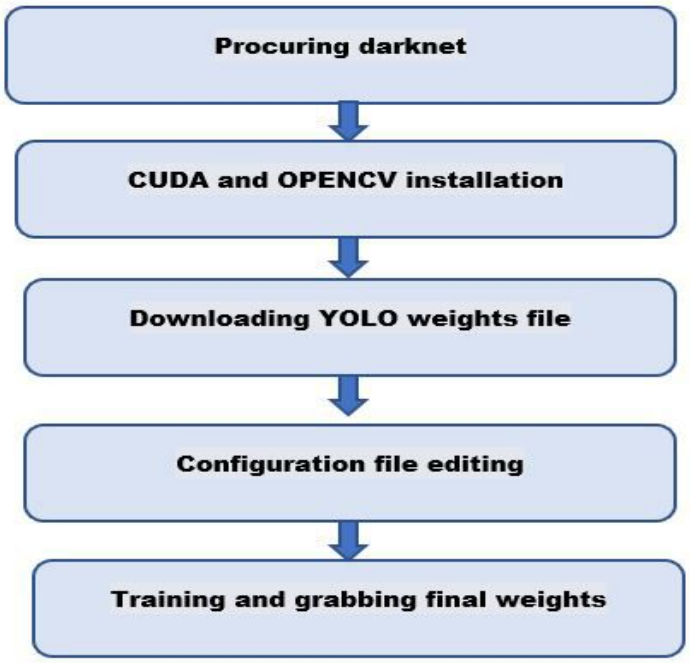


Figure 3.9: Flow chart of training process setup for local machine

4

Results

The possibility of having two training options was discussed in the previous section. The training was carried out on a cloud server, and the results are displayed as confusion matrices, performance metrics and images of predictions. The results also contain the class of distraction, size of data and other relevant information specific to each of the trails.

4.1 Trial 1: YOLOv3 tested for a single class of distraction

The confusion matrix for this trial is shown in Fig. 4.1. The columns represent the actual label of the distraction while the rows depict the label predicted by the algorithm. To make it easier to observe the data, the true labels are highlighted green in colour and false labels are red. The dataset was divided into training and test set respectively. The training set contained . The numbers shown in the confusion matrix are that of predictions for the test set. The class of distraction trained is drinking (c8). The top left box contains the number of detections which was correctly identified as drinking and the number was 197. The bottom left box signifies the cases which were drinking but predicted otherwise. There were 4 cases. The top right box contains predictions which were unrelated to drinking but were classified as drinking. Based on all these obtained values, few metrics which can aid in understanding the behaviour of the algorithm is tabulated in 4.1. These parameters are calculated based on the true and false predictions made by the algorithm. As seen from the table 4.1, the accuracy is around 88%. While some of the other measures like fall out, miss rate and specificity depict contradicting behaviour compared to high accuracy. To simplify, the algorithm detects every image as the same class. The results are discussed in detail in the following chapter.

- Class - c8 (drinking), refer to Table 4.3.
- Dataset size used for training - 910 images (more than 80% drinking).
- Algorithm - YOLOv3.
- Training set - 75% of dataset chosen randomly.
- Test set - 25% of dataset chosen randomly.

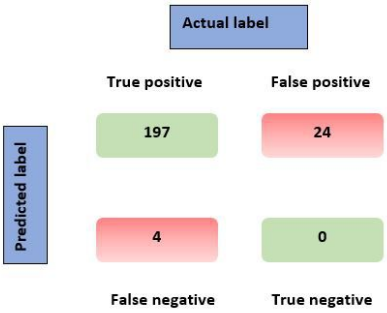


Figure 4.1: Confusion matrix for trial 1

Table 4.1: Metrics evaluated for trial 1

Parameters	Values(in %)
Accuracy	87.56
False positive rate or fallout	100
False negative rate or miss rate	1.9
Specificity	0

4.2 Trial 2: YOLOv3 tested for multiple classes of distraction

The trial involves testing the YOLOv3 on multiple classes of distractions. The description of each class is presented in the table 4.3. The dataset used contained a total of 912 images around 100 images per class. A total of 25% chosen randomly was kept as a test set. The confusion matrix is shown in Fig. 4.2. The values are the detection rate in percentage. The rows of the matrix represent the true/actual label of the image while the column represent the label detected by the algorithm. The main diagonal of the matrix, which is also highlighted, depicts the detection scores for each class, and these numbers are crucial among others in the matrix. The average detection rate per class for this trial is approximately 46%. The algorithm was trained for 4000 iterations while the average loss was under 2 at the end of the training process. It can observed from that the predicted label in each column does not sum to 100 percent. This is because of a parameter called confidence in the detection framework. The confidence defines the algorithm's certainty in identifying a particular class. The confidence value ranges from 0 to 1. When it is set to a higher value, the algorithm does not make any detection for some cases.
The distractions like c2 (talking on the right-hand side), c3 (texting on the left-hand side), c6 (reaching behind) and c8 (drinking) show higher detection rates over 50%. Whereas classes like c0 (safe driving), c4 (texting on right-hand side) and c5 (adjusting radio) recorded low scores of detection. It is also evident from the confusion matrix that, c6 (reaching behind) and c7 (hair and makeup) were the most miss-classified classes. The final detections for the different classes are shown in Fig. 4.3.

- Class - All classes of distraction(c0 to c8), refer table (4.3).
- Dataset size used for training - 912 images.
- Test set - 25% of dataset randomly picked (228 images)
  .
- Iterations of training - 4000 with avg. loss under 2.
- Single pose dataset.

Table 4.2: Classes of distractions

Classes	Description
c0	safe driving
c1	talk left (phone)
c2	talk right (phone)
c3	text left (phone)
c4	text right (phone)
c5	adjust radio
c6	reach behind
c7	hair and makeup
c8	drinking

Predicted label

True label	C0	C1	C2	C3	C4	C5	C6	C7	C8
C0	23.7	0	0	26.9	0	0	0	0	0
C1	0	46.15	0	0	0	0	0	0	0
C2	0	0	74.07	0	0	0	0	20.6	0
C3	0	0	0	50	0	0	0	0	0
C4	30.76	0	0	0	28	0	0	12	0
C5	0	0	0	0	0	25	0	0	0
C6	0	0	18.5	0	24	15.3	61.5	16	30.76
C7	46.15	0	0	0	36	50	23.07	40	0
C8	0	26.92	0	0	0	0	0	0	69.2

Figure 4.2: Confusion matrix for trial 2 showing prediction rate in %

Figure 4.3: Detections of distractions

4.3 Trial 3: YOLOv3 tested for multiple classes of distractions for a large dataset

After learning a few indicators from the two trials, namely the size of the dataset and multiple classes, this trial tries to consolidate the drawbacks from previous trials to achieve the best possible prediction rate. The trial contains all distractions as shown in 4.3. The size of the dataset used for training is 6317 images approximating to about 700 images per class. The test set contains around 1500 images. Another addition to this trial is multiple poses of distractions, as shown in Fig. 4.5. It becomes easier for the algorithm to generalize if tested on a completely different dataset, given that it was trained on different poses. Similar to trial 2, the predictions are depicted in the confusion matrix shown in Fig. 4.4. The rows indicate actual labels and columns indicate predictions by the algorithm. The highlighted boxes from top left to bottom right indicate the absolute class predictions for each distraction. After calculating an average of all the values highlighted, the average prediction rate of all the classes is approximately 88%. By increasing the data for training by approximately seven times, the average prediction was increased additionally by 42% roughly. The classes which had a prediction rate of over 90% were c2 (talking on left hand), c3 (texting with left hand), c5 (adjusting radio), c7 (hair and makeup) and c8 (drinking). Safe driving(c0) recorded the lowest prediction rate of 81.5%. Most of the miss classifications were under 10%, except for few. c1(talking with left hand) had 16.6% of the data miss classified as c7(hair and makeup). While c6(reach behind) had 12.4% recognized incorrectly as c7 (hair and makeup). The algorithm was trained for 9000 iterations with an average loss under 2 at the end of the training.

- Class - All classes of distraction(c0 to c8), refer table (4.3).
- Dataset size used for training - 6317 images.
- Test set - 25% of dataset randomly picked.
- Different pose of distractions for better generalization.
- Iterations of training - 9000 with avg. loss under 2.

Table 4.3: Classes of distractions

Classes	Description
c0	safe driving
c1	talk left (phone)
c2	talk right (phone)
c3	text left (phone)
c4	text right (phone)
c5	adjust radio
c6	reach behind
c7	hair and makeup
c8	drinking

Predicted label

True label	C0	C1	C2	C3	C4	C5	C6	C7	C8
C0	81.5	0	0	0	0	0	0	0	0
C1	0	83.3	0	0	0	0	0	0	0
C2	0	0	90.35	0	0	0	0	0	0
C3	0	0	0	93	0	0	0	0	0
C4	8.2	0	0	5	86.4	0	0	0	0
C5	0	0	0	0	0	92.2	0	0	0
C6	0	0	7.01	0	7.5	0	82.5	4.5	0
C7	0.63	16.6	0	0	6.01	2.1	12.4	90.9	3.03
C8	0	0	0	0	0	3.3	0	0	93.9

Figure 4.4: Confusion matrix for trial 3 showing prediction rate in %

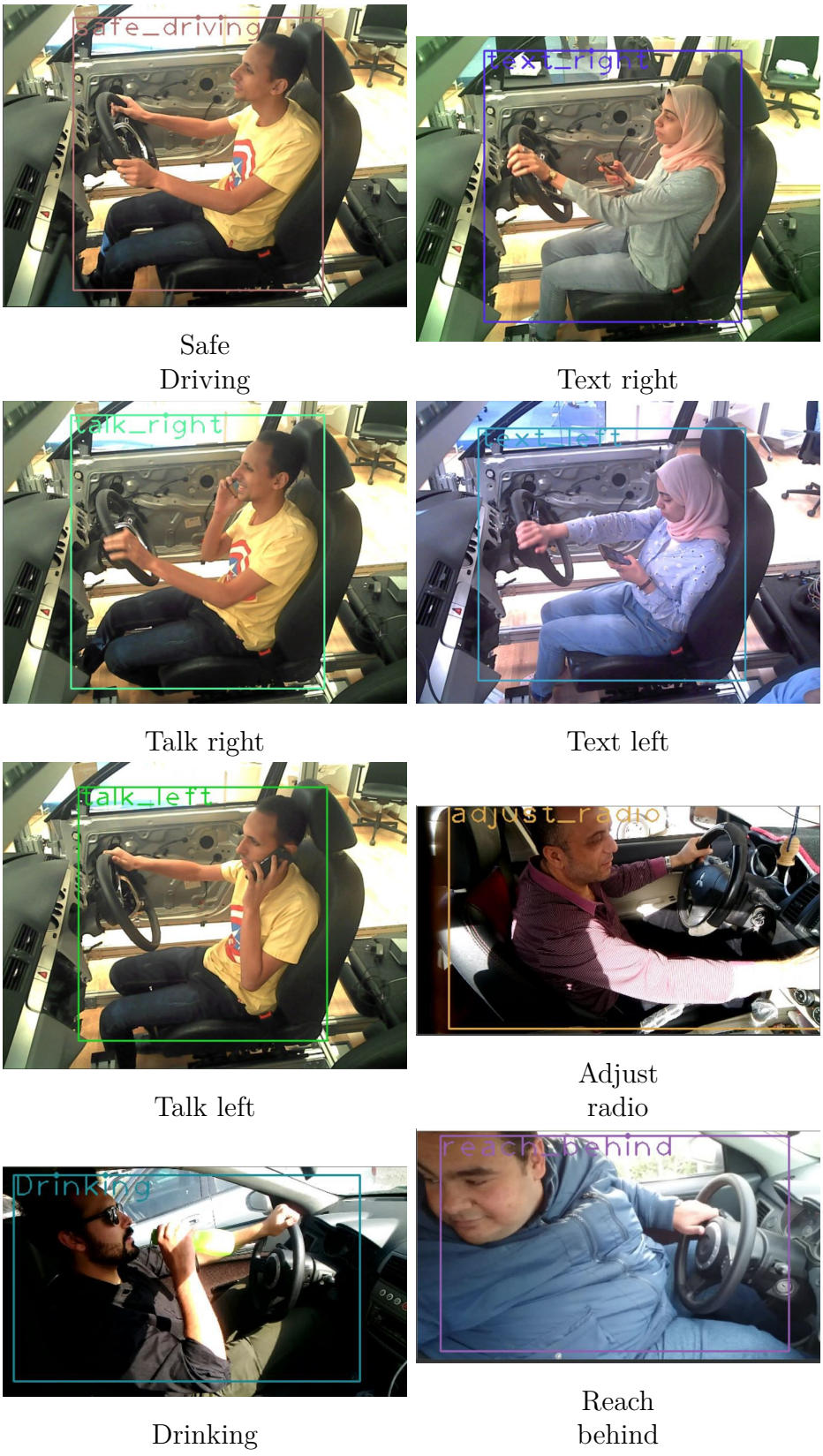


Figure 4.5: Detection of distractions (multiple pose)

4.4 All the trials in a nutshell

This section aims to compare some important aspects of all the trials. Each trial varied with class, data size and pose. This section also compares the performance of the algorithm in each trial.

Variables	Trial 1	Trial 2	Trial 3
Classes	1	9	9
Images trained	910	912	6317
Avg. prediction per class	87	44	88
Pose	single	single	multiple
Algorithm	YOLOv3	YOLOv3	YOLOv3

5

Discussion

The results obtained from all the trials are interesting and requires an examination to interpret the findings. Each trial was intended to check the behaviour of the algorithm considering variations in data size, classes and other hyper-parameters. Since the trials are different from each other, they will be discussed separately to bring them into context.

5.1 Trial 1: YOLOv3 tested for a single class of distraction

The trial was intended to get familiar with the working of the YOLO algorithm and identify one class of distraction correctly. Hence it was initially trained majorly for one class of distraction. The confusion matrix in Fig. 4.1 depicts the prediction rate for the class, drinking (c8). The test set contained a total of 225 images which approximately amounts to 25% of the training set. However, out of the test set, 24 images were not drinking, and the remaining 201 were drinking. This small sample of non-drinking images was included in the test set to verify if the algorithm not only succeeds in identifying drinking images but the non-drinking images as well.

The algorithm was successful in identifying almost all of the drinking images except for 4. But, the images which were unrelated to drinking were also classified as drinking. These values can be referred from Fig. 4.1. As seen from the table 4.1 the algorithm promises good values. Even though it is capable of identifying almost all the drinking images correctly, it is susceptible. It fails to classify non-drinking images correctly. Upon examination, it was found out that such results occur due to training imbalance [51]. Training imbalance occurs when data is not equally distributed in all the classes. Although few images were labelled for non-drinking and included in the training, this was insufficient.

The parameters evaluated from the confusion matrix is shown in 4.1. As discussed earlier, this classification is imbalanced, and since most of the images which were trained are drinking images, the algorithm boasts of high accuracy. But this may not be a good metric to evaluate this trial and often be misleading in binary classification. This algorithm failed to identify all non-drinking images, and this is reflected in the false positive rate or fallout. This means that the algorithm identifies everything that is fed to it as drinking. It is not ideal to have an algorithm which performs in such a manner as it is not robust. To make it more robust, it is necessary to balance the training data or have multiple classes of data.

5.2 Trial 2: YOLOv3 tested for multiple classes of distraction

Trial 2 is intended for predicting multiple classes of distractions and with a relatively lower number of the dataset for training. Around 100 images of every distraction were used which contained different individuals. The perspective of image for all individuals is the same and detections are shown in 4.3. This low number of the dataset is intended to observe the detections given the design of the architecture of YOLO. To recap, YOLO is designed based on the regression problem where the image pixels are transformed into bounding boxes along with confidence scores for classes. This makes it faster with mean average precision scores higher than most of the other object detection algorithms [40]. For these reasons, it was interesting to find the predictions for a low amount of data. Although it is a widely known fact that neural networks are image hungry for learning, there is no absolute rule on the number of images needed to achieve the desired accuracy. This is a trial and error method and often depends on the problem definition.
As mentioned in the results, the average prediction per class is approximately is 46%. Even though each class of data contained approximately the same amount of data, the prediction scores do not reflect the same. The classes which involve distinct actions or objects have relatively higher prediction scores. In the case of c1, c2, c3, c4 and c8, it is explicitly evident that mobile phone and water bottle were involved in this distraction. Since the labelling of images was accurately performed, the algorithm learnt to distinguish these classes better. However, there are some anomalies to be examined here. In the case of c0 (safe driving), where both the hands-on steering wheel defines the class, some detections suggested the driver was involved with using phone or reaching behind. The miss classification of safe driving for texting may be because, in some images, the hand holding the phone was close to the steering wheel. The same applies to being identified as reaching behind instead of safe driving where the individual was looking behind while both the hands were on the wheel. Similarly, for the class c7 (hair and makeup), the distraction is wrongly identified as talking on the right side of the phone. This is due to the similarity in the posture of talking on the phone and individuals using the right hand to adjust hair. In some cases, it is challenging to come to the reasoning for a particular miss classification. For c8 (drinking), it is identified as reach behind even though these postures are totally unrelated. The same can be observed in c4(texting on right hand) which is miss classified for reaching behind and hair & makeup.
Almost all of the miss-classifications can be attributed to a low number of training data even though the goal of the trial was not to achieve a specific level of prediction rate. It was interesting to find the learning capacity of the algorithm with fewer images. The algorithm was subjected to 4000 iterations of training and 280 epochs. The result of the trial makes it imperative to train with a larger dataset to see the difference in the prediction rate.

5.3 Trial 3: YOLOv3 tested for multiple classes of distraction

Trial 3 is comprehensive to the amount of data, multiple pose and modifications done to the configuration file to aid better training. Almost all of the data available was labelled for training while aiming for a high prediction rate. The dataset contains multiple poses of individuals engaged in distractions. This helps the algorithm to learn better, especially with the classes of distraction which involves using mobile phone with left and right hands. To recap the results for the trial, the average prediction rate of all the classes is around 88%. c8 (drinking) was the class with the best prediction score. A small percentage of around 3% of the drinking class was wrongly identified as c7 (hair and makeup). This was maybe due to the similarity in postures of drinking and reaching for hair. Similarly, c4(texting with right hand), c1(talking with left hand) also had miss classifications with c7 (hair and makeup) for the same possible reasons. But for other few classes, the wrong predictions are hard to clarify because of no possible relation with the posture. For example, c2 (talking on the right side) has no probable similarity to c7 (hair and makeup). In the case of c0 (safe driving), the miss classified predictions as c4 (texting with right hand) may be due to the closeness of the right hand to the steering wheel.
This trial showed that more data yields better result but fails to answer some anomalies with the miss classifications. Although the algorithm performs well on this dataset but having a forethought of a generalised algorithm should be put to the test on a different dataset to check the robustness.

5.4 Comparing results with related work

As mentioned previously, the machine intelligence group at AUC provided one of the earliest distraction detecting framework. The methodology used by them is different compared to this work as they segment each distraction distinctly and use a different CNN to train each of these segments. The final detection is based on the weights obtained from all the segments. Although this yielded an accuracy above 90 percent, it proved to be computationally heavy, and they proposed a lighter version which brought the accuracy to around 85 percent compared to prediction rate per class of 88 percent obtained here. The method proposed in this work follows a holistic approach of treating distractions as a whole and uses a more advanced algorithm which unifies the detection process by a single CNN. This made the training process easier and fast. This work provides an alternative for training on a cloud for those lacking competent hardware on a local machine. The intended application for AUC was to use it as a real-time system in semi-automated vehicles and hence demanded high accuracy. The intention of this work is different, where its use is to only help in the automatic annotation of secondary tasks in driving data.

5.5 Limitations

The limitations of this work are dependent on some of the performance constraints generally observed in CNNs, along with some of the limitations observed in this work.
CNNs perform well when its capabilities are utilised appropriately. But, some of their inevitable limitations are the failure in classification with varying pose or contrast. CNNs learn better for the pose they are trained with and usually fail to perform with a different pose, lighting or backgrounds. For example, the weight file of trial 2 would have failed to identify the change in the pose, as shown in figure 4.5. Hence, to have a generalised model, it needs to be trained across multiple poses and colour contrast (hue, saturation and exposure). This sometimes makes it expensive to get the data and annotate. Although a minor fix can be done with image transformations that may reduce the quality of data available in the image.
The work was initially intended to perform on the data available at SAFER. SAFER contained naturalistic driving data categorised and annotated into several classes of distractions. This data could not be used for training due to hardware limitations of the local machine and confidentiality in extracting the data out of the FOT rooms at SAFER. Hence, a different dataset was used, which was similar. As discussed above, variations in the data sets result in poor generalisation.
The training of CNNs requires powerful GPU which can be accelerated with CUDA. It requires a lot of time to train on the CPU. Hence, the training was performed on a cloud virtual machine. But this comes with some limitations as well. The cloud has limited time for training, and if the training goes idle for a while, then the model will be ejected out of the server. The training could approximately run for 10-12 hours on a cloud which was sufficient to arrive at the desired loss. But, when we use to start increasing the size of data for training, it will take longer time and training on cloud becomes difficult.

5.6 Future work

The future work can start with working on the data available at SAFER. The data at SAFER include videos or images which are grayscale. The concept of transfer learning can be applied to this data, where we use the pre-trained weights on a new dataset. Apart from this, to make the algorithm more robust and generalised, more data is needed along with competent hardware making the training faster. An alternative methodology can be implemented where, instead of identifying distraction as a whole, each distraction can be broken down into segments. This segmented approach can help identify specific objects or gestures of a distraction which can help the algorithm learn better, and the problem of detection becomes easier. Since this work intends to make the manual annotation of secondary task engagement in naturalistic driving data automatic, it can be tested on the same after having a generalised model. However, this might require validation between manual and automatic annotation.

6
Conclusions

Distracted driving as a result of secondary task engagement is one of the reasons for fatal accidents. One of the factors influencing this behaviour is due to the emergence of mobile phones and interaction with the infotainment system in the vehicle. Advancements in computer vision have led to algorithms capable of detecting objects in images or videos. To automatically detect these distractions, a convolutional neural network named YOLO (You only look once) was used to identify distractions. The data was available from Machine intelligence group at the American University of Cairo (AUC) on request, and this data was based on distractions categorised by StateFarm's dataset used for an object detection competition on Kaggle. The distractions were subdivided into nine classes viz; safe driving, texting with a left hand, texting with a right hand, talking with a left hand, talking with a right hand, adjusting a radio, drinking, hair & makeup and reaching behind. These images were used to train the YOLO algorithm so that it can recognise the pattern of distractions and could be later validated with testing. The images which were used for training were labelled by specifying the coordinates in the image containing the distraction. The training was carried out on a cloud virtual machine. Alternatively, the possibility of training on the local machine was also considered, but it was not preferred for reasons related to hardware capabilities available. The weights file was obtained after training along with a configuration file. The configuration file helped in setting up or changing certain parameters which aided in the better training process. An object detection framework was built, which could identify the distractions in the images using the weights file available after training. Several trials were conducted to check the behaviour of the algorithm, and it was modified with each trial to get better prediction rates and efforts were made to make it robust.

Trial 1 was intended to identify only one distraction (drinking). Since it was trained majorly on one class (drinking), the training imbalance made the algorithm to identify everything given to it as drinking. Whereas trial 2 was trained for multiple classes of distractions, but the data used for training was comparatively low. This was done to understand the traditional attribute of convolutional neural networks requiring more data to achieve high accuracy. It was interesting to find out with an average of around 100 images per class; the algorithm achieved an average prediction rate of 44%. This trial can be used to extrapolate the amount of data required to achieve higher prediction rate. Following trial 2, the final trial showed a higher prediction rate of 88% per class when compared to trial 2. This was achieved by using a larger size of training data and to aid efficient learning, and it contained multiple poses of all the distractions. The algorithm can be used to detect secondary tasks from naturalistic driving data available at SAFER provided it is trained on them.

Bibliography

[1] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," 2018.

[2] (2016). "Over 25 000 victims of road accidents in the eu in 2016," [Online]. Available: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/EDN-20171119-1?inheritRedirect=true (visited on 09/25/2020).

[3] W. Wijnen, W. Weijermars, A. Schoeters, W. van den Berghe, R. Bauer, L. Carnis, R. Elvik, and H. Martensen, "An analysis of official road crash cost estimates in European countries," *Safety Science*, vol. 113, no. December 2018, pp. 318–327, 2019, ISSN: 18791042. DOI: 10.1016/j.ssci.2018.12.004. [Online]. Available: https://doi.org/10.1016/j.ssci.2018.12.004.

[4] J. Hu, C. A. Flannagan, S. Bao, R. W. McCoy, K. M. Siasoco, and S. Barbat, "Integration of Active and Passive Safety Technologies - A Method to Study and Estimate Field Capability," *SAE Technical Papers*, no. November, 2015, ISSN: 01487191. DOI: 10.4271/2015-22-0010.

[5] F. Gustafsson, "Automotive safety systems," *IEEE Signal Processing Magazine*, no. 4, pp. 32–47, 2009, ISSN: 1053-5888. DOI: 10.1109/msp.2009.932618.

[6] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles," *IEEE Consumer Electronics Magazine*, no. 5, pp. 18–25, 2018, ISSN: 21622256. DOI: 10.1109/MCE.2018.2828440.

[7] H. J. Vishnukumar, B. Butting, C. Muller, and E. Sax, "Machine learning and deep neural network - Artificial intelligence core for lab and real-world test and validation for ADAS and autonomous vehicles: AI for efficient and quality test and validation," *2017 Intelligent Systems Conference, IntelliSys 2017*, no. September, pp. 714–721, 2018. DOI: 10.1109/IntelliSys.2017.8324372.

[8] (2020). "Aaa: Distracted driving increases with adas technology," [Online]. Available: https://aashtojournal.org/2019/12/20/aaa-distracted-driving-increases-with-adas-technology/.

[9] (2018). "Distracted driving," [Online]. Available: https://www.nhtsa.gov/risky-driving/distracted-driving.

[10] D. O. T. Hs, "Driver electronic device use in 2010," *Annals of Emergency Medicine*, vol. 59, no. 6, pp. 494–495, 2012, ISSN: 01960644. DOI: 10.1016/j.annemergmed.2012.03.003.

[11] (2020). "Nhtsa report for total number of motor vehicle crashes in 2018," [Online]. Available: https://www.iii.org/table-archive/20446 (visited on 11/09/2020).

[12] (2016). "Distractions," [Online]. Available: https://etsc.eu/tag/distraction/.

[13] J. Bärgman, *Methods for Analysis of Naturalistic Driving Data in Driver Behavior Research*. 2016, ISBN: 978-91-7597-501-6.

[14] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, pp. 1–6, 2018. DOI: 10.1109/ICEngTechnol.2017.8308186.

[15] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. arXiv: 1804.02767. [Online]. Available: http://arxiv.org/abs/1804.02767.

[16] H. M. Eraqi, Y. Abouelnaga, M. H. Saad, and M. N. Moustafa, "Driver distraction identification with an ensemble of convolutional neural networks," *Journal of Advanced Transportation*, vol. 2019, 2019, ISSN: 20423195. DOI: 10.1155/2019/4125865. arXiv: 1901.09097.

[17] (2018). "Road safety statistics - characteristics at national and regional level," [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Road_safety_statistics_-_characteristics_at_national_and_regional_level&oldid=463733.

[18] A. Debyser, "Road safety in the EU," no. February, p. 10, 2017.

[19] (2018). "Mobility and transport," [Online]. Available: https://ec.europa.eu/transport/road_safety/users/young-people/distraction_en.

[20] Y. Artan, O. Bulan, R. P. Loce, and P. Paul, "Driver cell phone usage detection from HOV/HOT NIR images," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 225–230, 2014, ISSN: 21607516. DOI: 10.1109/CVPRW.2014.42.

[21] *Number of smartphones sold to end users worldwide from 2007 to 2020*, https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007, Accessed: 2020-08-17.

[22] D. G. Low, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, pp. 91–110, 2004. [Online]. Available: https://www.cs.ubc.ca/%7B~%7Dlowe/papers/ijcv04.pdf.

[23] Y. Abouelnaga, H. M. Eraqi, and M. N. Moustafa, "Real-time Distracted Driver Posture Classification," no. Nips, 2017. arXiv: 1706.09498. [Online]. Available: http://arxiv.org/abs/1706.09498.

[24] *State farm distracted driver detection*, https://www.kaggle.com/c/state-farm-distracted-driver-detection, Accessed: 2020-08-25.

[25] A. Aksjonov, P. Nedoma, V. Vodovozov, E. Petlenkov, and M. Herrmann, "Detection and Evaluation of Driver Distraction Using Machine Learning and Fuzzy Logic," *IEEE Transactions on Intelligent Transportation Systems*, no. 6, pp. 2048–2059, 2019, ISSN: 15249050. DOI: 10.1109/TITS.2018.2857222.

[26] R. S. Hegadi, "Image Processing : Research Opportunities and Challenges," *National Seminar on Research in Computers*, no. March, pp. 1–6, 2010.

[27] Torsten sattler, *Image Analysis*, https://urlzs.com/QKgYy, Online; accessed 12 October 2020.

[28] K. S. B, "Stapor K. (2018) Evaluating and Comparing Classifiers: Review, Some Recommendations and Limitations. In: Kurzynski M., Wozniak M., Burduk R. (eds) Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017. CORES 2017. Adv," vol. 1, no. May 2017, 2018, ISSN: 21945357. DOI: 10.1007/978-3-319-59162-9.

[29] (2020). "Image source for overfit," [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9 (visited on 10/21/2020).

[30] Y. Xu and R. Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating

the Generalization Performance of Supervised Learning," *Journal of Analysis and Testing*, no. 3, pp. 249–262, 2018, ISSN: 2096-241X. DOI: 10.1007/s41664-018-0068-2. [Online]. Available: https://doi.org/10.1007/s41664-018-0068-2.

[31] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 658–666, 2019, ISSN: 10636919. DOI: 10.1109/CVPR.2019.00075. arXiv: 1902.09630.

[32] (2020). "Image source for iou," [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visited on 10/20/2020).

[33] A. M. Kayid, "Performance of CPUs/GPUs for Deep Learning workloads," no. May, p. 25, 2018. DOI: 10.13140/RG.2.2.22603.54563.

[34] (2020). "Image source for gpu performance," [Online]. Available: https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/ (visited on 10/21/2020).

[35] Y.-Q. Wang, "An Analysis of the Viola-Jones Face Detection Algorithm," *Image Processing On Line*, vol. 4, pp. 128–148, 2014, ISSN: 2105-1232. DOI: 10.5201/ipol.2014.104.

[36] H. Chiroma, U. A. Abdullahi, S. M. Abdulhamid, A. Abdulsalam Alarood, L. A. Gabralla, N. Rana, L. Shuib, I. A. Targio Hashem, D. E. Gbenga, A. I. Abubakar, A. M. Zeki, and T. Herawan, "Progress on Artificial Neural Networks for Big Data Analytics: A Survey," *IEEE Access*, vol. 7, pp. 70 535–70 551, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2880694.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," 2009.

[38] (2018). "Region proposal network (rpn) — backbone of faster r-cnn," [Online]. Available: https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9 (visited on 10/12/2020).

[39] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014, ISSN: 10636919. DOI: 10.1109/CVPR.2014.81. arXiv: 1311.2524.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016, ISSN: 10636919. DOI: 10.1109/CVPR.2016.91. arXiv: arXiv:1506.02640v5.

[41] M. Naveenkumar, "OpenCV for Computer Vision Applications," no. March 2015, pp. 52–56, 2016.

[42] J. D. Dignam, P. L. Martin, B. S. Shastry, and R. G. Roeder, "Eukaryotic gene transcription with purified components," *Methods in Enzymology*, no. C, pp. 582–598, 1983, ISSN: 15577988. DOI: 10.1016/0076-6879(83)01039-3.

[43] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Computing in Science and Engineering*, no. 2, pp. 22–30, 2011, ISSN: 15219615. DOI: 10.1109/MCSE.2011.37. arXiv: 1102.1523.

[44] P. Barrett, J. Hunter, J. T. Miller, J.-C. Hsu, and P. Greenfield, "matplotlib – A Portable Python Plotting Package," *ASP Conference Series*, no. June, p. 91, 2005. [Online]. Available: http://adsabs.harvard.edu/abs/2005ASPC..347...91B.

[45] J. Redmon, *Darknet: Open source neural networks in c*, http://pjreddie.com/darknet/, 2013–2016.

[46] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020. arXiv: 2004.10934. [Online]. Available: http://arxiv.org/abs/2004.10934.

[47] *Jupiter note book*, https://fileinfo.com/extension/ipynb, Accessed: 2020-08-30.

[48] (2019). "How do pre-trained models work?" [Online]. Available: https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9 (visited on 10/19/2020).

[49] M. Zabir, N. Fazira, Z. Ibrahim, and N. Sabri, "Evaluation of pretrained Convolutional Neural Network models for object recognition," *International Journal of Engineering and Technology(UAE)*, no. 3, pp. 95–98, 2018, ISSN: 2227524X. DOI: 10.14419/ijet.v7i3.15.17509.

[50] *Official website of yolo*, https://pjreddie.com/darknet/, Accessed: 2020-09-01.

[51] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, pp. 249–259, 2018, ISSN: 18792782. DOI: 10.1016/j.neunet.2018.07.011. arXiv: 1710.05381.