Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of

Bachelor Thesis

Bachelor Thesis title- CNN in Object Recognition

Author of the thesis-

Urvish Sunilbhai Kavde

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Urvish Sunilbhai Kavde

Informatics

Thesis title

CNN in object recognition

Objectives of thesis

The main goal of the thesis is to create platform that detects objects in real-time using the COCO SSD dataset and the ml5.js library. The supporting goals are to form an overview of the areas in which convolutional neural networks (CNN) have been used, introduce the object detection mechanism, how it works and what steps need to be included for the CNN to detect an object through the camera.

Methodology

The methodology of the thesis is based on an analysis of technical and scientific sources focusing on artificial intelligence, machine learning, and object recognition. Based on the synthesis of these sources, the current state of research in the field of convolutional neural networks will be described, including the areas in which CNN has been used successfully. Following the theoretical overview, the findings will be verified on a prototype web application based on the COCO SSD Dataset and ml5.js library. The The performance of the application will be assessed in terms of detection accuracy using a laptop camera.

The proposed extent of the thesis

40-50 pages

Keywords

Neural network, Machine learning, Deep learning, JavaScript, COCO-SSD, Image classification, Object detection.

Recommended information sources

CYGANEK, Boguslaw, 2013. Object Detection and Recognition in Digital Images: Theory and Practice. Chichester, UK: John Wiley and Sons. ISBN 978-0470976371.

ML4js: Friendly Machine Learning for the Web [online]. [cit. 2022-01-23]. Dostupné z: https://ml5js.org

RIVERA J.D.D.S., 2020. Welcome to TensorFlow.js.: Practical TensorFlow.js. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6273-3_1

Expected date of thesis defence

2021/22 SS – FEM

The Bachelor Thesis Supervisor

Ing. Petr Hanzlík, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 1. 3. 2022

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 7. 3. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 24. 11. 2023

Declaration

I declare that I have worked on my bachelor thesis titled "CNN in Object Recognition" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on date of submission _____

Acknowledgement

I would like to thank my Supervisor **Ing. Petr Hanzlík, Ph.D.** and all other persons, for their advice and support during my work on this thesis.

Abstraktní

Rychlý vývoj hlubokých CNN a GPU, které poskytují lepší výpočetní výkon, je hlavním motorem neuvěřitelných výsledků „lokalizace objektů na základě počítačového vidění“. Rychlejší umístění CNN je účtováno jako další „špičkový“ přístup založený na CNN k „rozpoznávání objektů hlubokého učení“. První část zahrnuje základ studie, důležité recenze, body a cíle výzkumu, význam výzkumu, význam výzkumu a celou výstavní strukturu. Recenze je v podstatě založena na předchozí práci na toto téma a krátkých rozhovorech o výzkumu. "Úplné učení" je nejužitečnějším nástrojem v sadě nástrojů výpočetní inteligence. Replikuje skutečný design lidského mozku, neuronů, perceptronů a axonů. Stejně jako skutečná lidská mysl má nespočet neuronů a propojenou organizaci neuronů nazývanou „falešná organizace mozku“ nebo zkráceně „ANN“. Zahrnuje komplexní násobení k dosažení hlavního učebního cíle. S drobnými úpravami jej lze použít pro dva typy úkolů, jako je opakování nebo seskupování. Tento kód je zpráva HTML, která k identifikaci objektů používá balíček ml5.js. Záznam se jmenuje "Object Detection | COCO SSD" a má nastaveny příslušné metainformace, aby program mohl stránku zobrazit přesně. V těle zprávy můžete najít skript JavaScript, který využívá funkci rozpoznávání knihovny ml5.js. Produkt streamuje informativní video ze zařízení klienta a provádí rozpoznání objektů na okraji streamu.

Klíčová slova

CNN, COCO SSD, YOLO (“You Only Look Once”), DNN, ml5.js API, Quick R-CNN, R-CNN

Abstract

The rapid development of deep CNNs and GPUs that provide better computing power is the main driver behind the incredible results of "object localization based on computer vision". Faster Location CNN is billed as another "cutting edge" CNN-based approach to "deep learning object recognition." The first part includes the basis of the study, important reviews, points and research objectives, the importance of research, the importance of research and the entire exhibition structure. Review is basically based on previous work on the subject and short talks on the research. "Complete learning" is the most useful tool in the computational intelligence toolbox. It replicates the real design of the human brain, neurons, perceptrons and axons. Like the real human mind, it has countless neurons and an interconnected organization of neurons called "false brain organization" or "ANN" for short. It involves complex multiplication to achieve the main learning objective. With minor adjustments, it can be used for two types of tasks, such as repetition or grouping. This code is an HTML report that uses the ml5.js package to identify objects. The entry is named "Object Detection | COCO SSD" and has the appropriate meta information set so that the program can display the page accurately. In the body of the report, you can find a JavaScript script that uses the recognition functionality of the ml5.js library. The product streams informative video from the client's device and performs object recognition at the edge of the stream.

Keywords

CNN, COCO SSD, YOLO (“You Only Look Once”), DNN, ml5.js API, Quick R-CNN, R-CNN

Table of Contents

Chapter 1: Introduction

1.1 Introduction

The faster region CNN is considered another “state-of-the-art” CNN-based “deep learning object detection” approach. In this first chapter, the background of the study, issues to this relevant study, aims and objectives of the research, research questions, research significance, and the entire thesis structure are going to be discussed.

1.2 Background of the study

In today's world, we can find several machine learning or deep learning applications which involve object detection using the digital camera. It helps the computer to understand the content and the context of the image. Car overspeed fine system also use the same technology to get the car number from its number plate via those images captured by the sensors and the system. This also helps those tiny silicon chips to read the text captured on the images [15]. This opens up a lot of possibilities to us. Amazon Go, the smartest grocery store without any shopkeeper or human cashier, is one of the greatest examples of this amazing technology [10]. It saves lots of human effort and with the performance. This all is the game of finding patterns from a bunch of numeric matrices. In this current era, machine learning and deep learning is breaking this barrier and helping out computers to see through those numbers and feel those hidden patterns so that they can work on that new information and make its own decision without any human interaction [11]. We humans are so efficient with patterns, but computers are not so comfortable with patterns, they understand numbers hex codes, zeros and ones (Li *et al.* 2019).

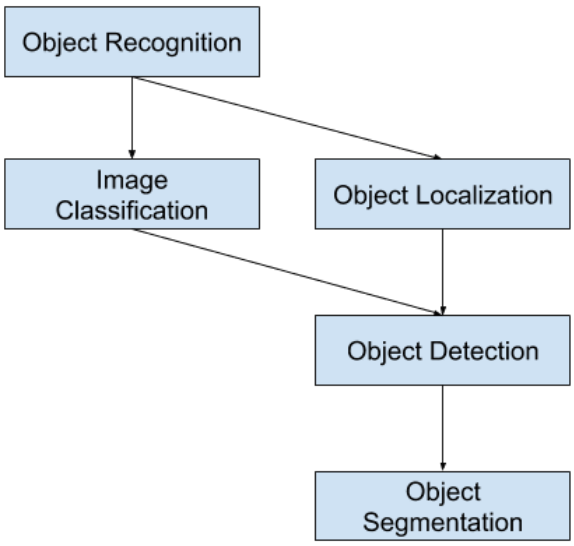

Figure 1.1: "Overview of Object Recognition Computer Vision Tasks"

(Source: [10])

Image classification is indicating algorithms that can produce a list of "object categories" within an image. There are two kinds of this process, and these are "Single object localization", and "Object detection" [10]. The first one is presented along with an axis-aligned bounding box that indicates the scale and the position within each category of an instance. Besides, the second one is similar, but every instance is positioned and scaled for each object category [13].

1.3 Rationale of the research

What is the issue?

The issue of this concerned research is to create a platform that detects objects in real-time using the COCO SSD dataset and the ml5.js library. “Object detection” is the piece of work of recognizing and characterizing position of each object of interest in an image. In PC vision, this strategy is used in applications such as image retrieval, surveillance cameras, and standalone vehicles [17]. One of the most popular “Deep Convolutional Neural Networks” (DNN) clusters for object detection is YOLO (“You Only Look Once”), which achieves complex results with a single end-to-end model. However, it is necessary to have a decent graphics card, similar to the Nvidia Tesla V100 that everyone can use. Therefore, the practical implementation of these models still remains a key challenge. Additionally, ml5.js aims to make AI accessible to a wide range of professionals, beginners, innovative programmers, and students [6]. The library contributes an entrance to AI computations and models in-app and at scale over TensorFlow. An object recognition model is expected to constrain and differentiate different elements in a single image. This model is the TensorFlow and Node.js port of the SSD COCO template. The issue of this research is significant due to the industrial revolution that has recently used computer vision. Deep learning is widely used in robotics, surveillance, medical, and automation industries. Due to its successes in applications including “language processing”, “object identification”, and picture classification, the technology that is being discussed the most right now is deep learning. The market projection indicates that growth will be exceptional in the upcoming years [6]. “Strong Graphics Processing Units (GPUs)” and a large number of datasets are both readily available, which is one of the primary explanations given for this. The most crucial components of object detection are image categorization and detection. There are many different datasets accessible. “Microsoft COCO” is one such, extensively used in the domain of “image classification” [3]. It is an object recognition benchmark dataset. The introduction of a sizable dataset makes picture recognition and categorization possible.

Why is it an issue in current days and so significant?

This review compares YOLO, SSD, and Faster-RCNN. SSD is the initial comparison algorithm employed in this paper, which enhances the end network with layers of various attributes and makes detection simpler [1]. A convolutional neural network is used in the “Faster R-CNN”,

“a unified”, “quicker”, and more accurate technique of object recognition. Joseph Redmon created YOLO, which provides an “end-to-end network”.

In recent years, object recognition has received significant study attention. With the use of effective learning tools, deeper aspects may be quickly found and examined [4]. To conduct a comparative analysis and derive useful conclusions for their application in object detection, this work aims to assemble data on numerous “object detection tools and algorithms” employed by various researchers. The goal of a literature review is to obtain an apprehension of this work.

The research sheds light on

This research sheds light on CNN in Object Recognition by creating a platform that detects objects in real-time using the COCO SSD dataset and the ml5.js library.

Additionally, certain deep learning methods for object identification systems are accessed in the paper. According to the current study, deep CNNs operate under the tenet of weight sharing. It provides details on certain significant CNN points.

These CNN characteristics are illustrated in this thesis:

- *CNN requires the multiplication of two overlapping functions and its integration.*
- *In order to lessen their spatial complexity, feature maps are abstracted.*
- *To create the feature maps using filters, the process is repeated.*
- *Different kinds of pooling layers are used by CNN.*

A novel technique for object detection in photos utilizing a single deep neural network was developed by Wei Liu and colleagues. As per the researchers, "SSD" is considered a "straightforward method" that requires an object proposal since it is based on the total elimination of the process that proposes. They termed this procedure the "Single Shot MultiBox Detector SSD". The succeeding pixel and resampling phases are also dropped. After that, everything is done in one step. In addition to being very simple to use and very simple to integrate into the system, SSD is also very simple to train. This facilitates simpler detection. The foremost element of SSD is the extension of several feature maps to multiscale convolutional bounding box outputs. The work mentioned here served as inspiration for the "training and model" analysis of the SSD model used in our research.

1.4 Aim and objectives of this research

Aim:

The main aim of this research is ***to create a platform that detects objects in real-time using the COCO SSD dataset and the ml5.js library***.

Objectives:

The research objectives are as follows:

To form an overview of the areas in which CNN (Convolution neural Network) have been used.

To introduce the object detection mechanism.

To describe the working principle and steps that needed to be introduced for CNN to detect an object through the camera.

1.5 Research questions

Questions:

Research questions on this concerned topic are as followed:

What are the areas of object recognition in which CNN (Convolution neural Network) has been used? What is the object detection mechanism? What are the working principles and steps that needed to be introduced for CNN to detect an object through the camera?

1.6 Significance of the research

In this research, the Microsoft COCO dataset has been used as an ongoing part of our study and presented a representation of each of the three computations using different architectures using the relative measures expected for each implementation above. Displaying the results of different calculations on equivalent data sets can be used to familiarize with the specific properties of each calculation, understand how they differ from each other, and draw conclusions about how to handle object recognition with specific circumstances [8].

The faster R-CNN model was developed as an item validation strategy based on the work of Ross Girshik. As for target recognition, we use the CNN method. An initial part of Hirschick's strategy is to isolate the training of a "deep convolutional network" to withdraw critical moments from the training of an "order support vector machine". This is done by providing "window computation" as opposed to the "traditional sliding window" technique of R-CNN models [12]. Best of all, we used a fast R-CNN approach to combine extraction and description in a single framework.

In contrast with R-CNN, fast R-CNN makes some preparation memories that are multiple times quicker. The proposition seclusion locale and a limited quantity of Quick R-CNN are consolidated

into an organization layout called the district proposition network in the quicker R-CNN approach (RPN) [14]. Both Quick R-CNN and Quicker R-CNN are similarly exact. The investigation discovers that the strategy is a blended, 5-7 casing for every subsequent profound learning-based object location framework (Edges Each Second). This work gave me a central comprehension of R-CNN, Quick R-CNN, and Quicker R-CNN. This paper filled in as motivation for the model's preparation.

In this study, a framework is built for detecting and tracking moving objects using CCTV (Shut Circuit TV) cameras by adding CNN to the main input. It works based on the basic deduction process that applies to all households. Our work used an undifferentiated design in this study.

YOLO is a one-time convolutional brain network that is utilized to characterize a few competitors and foresee the casing position. This considers the discovery of focus from start to finish. In order to handle the item recognition issue, a relapse issue is utilized [2]. The most common way of doling out the result gained from the first picture to the classification and position is done by a solitary start-to-finish framework. The technique depicted in this study filled in as a motivation for the bouncing box expectation and component extraction of the Consequences be damned engineering utilized in this work.

It works better and has a new "base model" structure. The use of YOLO for its complex is taken from this review. Additionally, jobs created from the PASCAL Visual Item Classes (VOC) dataset are run from start to finish [16]. The organization is a fully functional preview version. The strategy in this paper was used to train a YOLO for its model using PASCAL VOC.

A more complex SSD is the underpinning of another paper. The creators of this paper propose that Small SSD, a solitary shot identification profound convolutional brain organization, be presented. To make constant inserted object distinguishing proof more straightforward, a Small SSD was made. It comprises essentially further developed layers consisting of a non-uniform Fire sub-network and a heap of helper convolutional includes layers based on SSDs [7]. The discoveries of this trial have uncovered that “Minuscule SSD” is appropriate for installed identifications, with its size of 2.3 MB being considerably more modest than Little Just go for it. For examination, a practically identical SSD model was utilized.

1.7 Summary

From the entire chapter discussion, it can be said that "object recognition" cites to a collection of particular tasks as digital photographs and this is region based with CNN or R-CNN. This is a family of techniques that can address object localization with recognition tasks and that is designed for model-performance. YOLO is also considered the second family of techniques for object recognition by designing for real-time usage and speed.

Chapter 2: Literature Review

2.1 Introduction

This section is predominantly concentrated on the background of object recognition with deep learning and many algorithms such as “COCO SSD” & library such as “ml5.js”. Literature review is mainly based on previous works regarding this and a brief discussion of concerned study of the research. “Deep learning” is the most powerful tool in the AI toolkit. It mimics the actual structure of the human brain, neurons, and axons. Just like the real human brain, it has many neurons and an interconnected network of neurons called “artificial neural networks”, or “ANN” for short. It uses complex backpropagation for learning purposes. With minor modifications, it can be used for many types of tasks, such as regression or classification.

2.2 Explaining overview of the areas in which CNN (Convolution neural Network) have been used for Object recognition.

In today's world we can find several machine learning or deep learning applications which involve object detection using the digital camera. It helps the computer to understand the content and the context of the image. This also helps those tiny silicon chips to read the text captured on the images. This opens a lot of possibilities to us. Amazon Go, the smartest grocery store without any shopkeeper or human cashier, is one of the greatest examples of this amazing technology [40]. Car overspeed fine systems also use the same technology to get the car number from its plate via images captured by the sensors and the system. It saves lots of human effort and with the performance it also increases the efficiency and accuracy. This all is the game of finding patterns from a bunch of numeric matrices. We humans are so efficient with patterns, but computers need these to be converted to numbers, hex codes, zeros and ones. In this current era, machine learning and deep learning is breaking this barrier and helping computers to see through those hidden patterns so that they can work on that new information and make its own decision without any human interaction [41].

2.3 Discussing of the object detection mechanism.

ANNs uses a complex back propagation technique for training purposes [18]. With a little bit of modification, it can be used for both kinds of tasks, whether it is regression or classification. The only downside is that, unlike machine learning algorithms, it is a super complex, a gigantic beast that requires lots of computing power and a huge amount of training, testing, and validating data. Still, after all those complexities and heavy requirements the accuracy and flexibility make it worthwhile and on the top of the line.

Although of its accuracy and flexibility, till now we just saw how much better it is than the traditional machine learning methods, not more than that; but there is a 'but'. ANN is just the foundation of the whole big magical empire [19]. The main game starts now. There is a special kind of modified “artificial neural network” which can go beyond the imagination and can do stuff that we saw in science fiction movies earlier. It's CNN or Convolution Neural Network. This state-of-the-art algorithm can find patterns or features from the images like we humans do. It takes 2D images, maybe it's a color image or grayscale image, and runs a small filter window throughout the whole image step by step after applying the filter all over the image, the highlighted features are passed to the next layer [20]. The first few CNN layers can only highlight simple things like a horizontal line, vertical line, tilted line, etc. As the layer goes deeper it becomes able to detect much more complex shapes like a circle, rectangles etc. After properly highlighting the important features in the image if anyone wants then he/she can get that feature map as output, but to detect or predict something, most of the time we flatten this 2D image to a single dimensional array and then use normal ANN layers and do classification or regression, which is suitable for us.

2.4 Description of the working principle and steps that needed to be introduced for CNN to detect an object through a camera.

The goal of Object Detection is to take an image, check whether any known object is present in that image or not, and if any known object is found then classify that object and label it and most importantly mark the position the object is located in the image. There may be multiple objects present on a single image with different sizes and shape of the same of multiple classes, or there may be no object present on the image [21]. Now it may be confusing to you that it’s neither a classification problem nor a regression problem, how can we solve this kind of problem? Still, in the deep of your mind, you are most probably trying to link it with the classification problem. The

good news is that you are right; it's a kind of classification problem that has to be solved using a different complex approach.

Let's talk about the most general object detection technique, how it works, what processes are involved in it and many other things.

First of all, the dataset. Like any other dataset, there is a training image and labels, but unlike any other labels, these labels not only store a single object label for a single image but one or more object names with their corresponding position in the image, or with the ground truth boxes [39].

Next the detection mechanism. For that the model creates a bunch of boxes of different size and aspect ratio in different places on that image. Now the SSD model will check for those boxes which are "overlapping with the ground truth boxes or the labels" [22]. Then it is IoU or Intersection over Union for the heights overlapping box, and this box is shown as the output.

Now for matching the detector uses several boxes with overlapping over 50% and compared with the labelled ones for better accuracy.

Let's talk about the role of CNN in this whole process. Obviously, the silicon chip can't take two photos side by side and compare them to tell whether they are the same or not. It must extract some features from it and have to compare those features to tell whether those two images are the same or not. In this case, CNN is a champ in feature extraction, and with its parallel running capabilities, it can do it much faster than any other serial machine learning algorithm [23]. Also, the feature map of CNN helps us to find the accurate location of the object in the picture.

The SSD model we used here also uses a quite popular CNN network for this, which is *VGG-16*.

2.5 Theoretical background of the study

SSD (Single Shot Detection) Algorithm

SSD is considered as an algorithm of object detection that is mainly modified "VGG16 architecture" and that has reached records in precision and performance in managing object detection tasks. This can be scored approximately 74% mAP ("mean Average Precision") at 59 frames in every second of the standard datasets like COCO. The architecture of SSD mainly builds on the VGG-16 architecture and discards all connected layers within this.

The major reason behind VGG-16 being based on the network is due to high-quality image classification and strong performance, and improving results with transfer learning helps. Rather than adding a VGG full connected layer, a specific auxiliary conv6 ("convolution layers") has been added and this enables the extraction of all features with various scales and decreases the size of each subsequent layer.

The loss function of multi-box combines with two critical components and these are location loss and confidence loss. In addition, location loss is mainly measured by predicting the network's bounding boxes from the training set and whereas, confidence loss is indicating the way the network is able to compute the bounding box. This is categorical loss with "cross-entropy".

Therefore,

Multibox loss= confidence loss + (alpha location loss)*, where alpha indicates minimising the impact of the location loss.

Different object detection models such as YOLO and Faster R-CNN execute operations at much slower speeds in comparison with much cheaper object detection/recognition methods, SSD. Before SSD was developed, some attempts were made to create faster detectors with the modification of each phase of the detection process. However, any appreciable speed boost from such changes only leads to a reduction in detection accuracy, so the researchers conclude that instead of modifying an existing model, they should develop an object detection model that is substantially different, and thus, an SSD Model [39]. SSDs do not resample images or options for the bounding box assumptions. Also, it is quite simple compared to object recommendation request methods as it completely eliminates recommender creation and subsequent resampling of pixels or feature steps and summaries all computations in one. As an upshot, SSDs are elementary to instruct and simpler to include in systems that call for a sensor element.

Its architecture relies heavily on bounding box generation and feature map extraction (also called standard bounding box). The network computes the loss by comparing the projected offsets to the actual class and standard bounding boxes to the training's real-world principles examples with the help of using different filters in each iteration. All parameters are updated by utilising the loss value and the back propagation algorithm [21]. In this way, the SSD can grasp the optimal filter structure that can recognize the features of the object and generalise the instruction-provided pattern so as to lower the lost value, with excellent accuracy in the evaluation step as a result. SSD is based on a complex feed-forward network that expands an assembly of fixed-size bounding boxes and produces a score depending on the existence of object instances within those boxes, followed by non-maximal repression of finally initiating an accurate detection effect [22]. The initial network layer is dependent on the conventional architecture used for first-rate image codification (and clipped in advance of the codification layer), which is the VGG-16 network [39]. A helper structure like convo6 is adjoined to the shortened base network to generate recognition. Another study demonstrates the use of CNN and background subtraction to create a framework for detecting and moving objects using CCTV cameras [21]. It operates by applying a background subtraction algorithm to every frame. One more detection system is You never look twice (YOLO) [35]. YOLO proposes a convolutional neural network system for categorising several candidates and predicting the bounding frame position. After that, this approach can be used to detect targets from beginning to end. To tackle the object detection problem, a regression problem is used. The output produced from the original image is placed into the category and position by an end-to-end system [19]. An updated approach with a new origination model structure, a specified pooling pyramid layer, and improved presentation which utilises a progressed YOLO v1 network model to optimise the loss function [31]. Small SSD, a one-shot detection deep convolutional neural network, is recommended by the authors of this research [30]. TINY SSD was developed to make real-time embedded object recognition simpler. A highly optimised, non-uniform Fire sub-network stack and a non-uniform sub-network stack of SSD-based auxiliary convolutional feature layers are combined to create Tiny SSD, a single-shot detection deep convolutional neural network for real-time embedded object detection. This network is specifically designed to reduce model size while maintaining object detection performance. The strongest aspect of Tiny SSD is its size, which is even smaller than Tiny YOLO at 2.3 MB [29]. The theoretical findings for developing an object detection system demonstrate that Tiny SSD is appropriate for embedded detection. In this

work, the use of CNNs in deep learning techniques for object detection is described. The study also covers the use of deep learning techniques in object recognition systems. Information from CNN on a few key points, according to the current study, shows that deep CNN acts in accordance with the weight distribution principle [28]. Each additional feature layer (or, alternatively, an established feature layer from the foundational network) utilising several convolutional filters, can provide an attached group of detection predictions [27]. For a feature class of size m × n with p bands, the fundamental factor to forecast the characteristics of potential detection is a small 3 × 3 × p kernel that generates scores for a category or a Shape offset from the coordinates default box. At each m × n position to which the kernel is placed, it fabricates an output value [37]. When employing a fully connected middle layer for this stage rather than a filtered complex, the measured when compared to each feature map location, the bounding box offset output values are consistent with the default box position of the YOLO architecture. Area-Based Constitutive Neural Network is known as R-CNN. This method combines area suggestions for object segmentation with first-rate CNNs for object recognition [26]. The algorithm of the primordial R-CNN procedure is depicted below:

1. Several candidate region suggestions are retrieved from the input image using a selective search method. In this approach, the initial subsegment is constructed with many potential locations [26]. Then, using a greedy algorithm, related regions are joined to create larger regions.

2. The propositions are distorted by the CNN component, which then produces discrete features as vector output.

3. In order to identify the proposal's interesting objects, the retrieved features are put into an SVM.

An object detection algorithm called Fast R-CNN corrects some problems with R-CNN. It adopts a similar methodology to R-CNN, but instead of using region proposals, CNN creates a convolutional feature map from the picture itself, from which it extracts and warps area proposals [25]. The deformed rectangles can be resized to a specific size to accept fully connected layers using RoI pooling layers. The RoI vector is then used to forecast the region's class using a SoftMax layer. Fast R-CNN is quicker than its predecessor since CNN only needs a maximum of 2000

suggestions per run [24]. To create the feature map, a convolution process is only used once per image. During training, SSD requires a single input image and real data box for each item. Convolutionally evaluate a limited number of canonical boxes with various aspect ratios at each area on various feature maps at various scales are to be measured [30]. We forecast shape offsets and credences for all object classes ((c1, c2,..., cp)) for each standard box. We initially match these regular boxes to the real world data boxes during training. We complemented two of the common boxes, for instance, to cats, and one to dogs. The remainder is considered negatives, whereas these are seen as positives [31]. The localization loss (such as Smooth L1) and confidence loss are weighted together to form the model loss (such as Softmax).

TensorFlow.js

The very first challenge was to run machine learning in the browser. The premise of simplifying cloud servers, the designer's ability to program in Python, and the significant cost of hardware were some of the reasons. This changed when "Tensorflow.js" was added. "TensorFlow.js" is a fully open-source library that uses "JavaScript" and high-level APIs to represent, train, and run a variety of AI models.

ML5.js

TensorFlow.js is the foundation for ML5.js that is a JavaScript framework that offers browser-based entrance to ML algorithms, tasks, and models. The latter approach eliminates the need for customers to purchase ML libraries or various in-app assets. Customers need to log in to your website before launching the app, which will enable mobile innovation if someone has the app. Finally, since all information is upfront, it is not affected by laziness issues and remains secure and private.

Methodologies

Below are described famous CNN architectures-

VGG16 Architecture

K. Simonyan and A. Zisserman created the VGG16, commonly referred to as OxfordNet, as a CNN architecture for image classification and detection. This name was inspired by the Visual Geometry Group at Oxford University. This model, which was employed in 2014 to take first place

in the '*ImageNet competition'*, is however recognized as a "superior vision model" till today [32]. "VGG16" had been practicing on NVIDIA Titan Black GPUs for weeks.

"VGG16" consists of 16 layers in total, 13 of which are convolutional, 3 of which are fully linked, and 5 of which are maximum pooling. We can see that it has two convolutional layers at first, then a "max-pooling layer", then two "convolutional layers" once more, then a "max-pooling layer", then three convolutional layers again, then three "convolutional layers" again, and finally three "convolutional layers" again, followed by a "max-pooling layer". There are ultimately 3 layers, all of which are interconnected [33]. These model layers are accurate to 92.7%, have 138 million parameters in total, and have some weights. It utilises a 2x2 maximum pool size and a 3 x 3 Kernel for convolution.

COCO Dataset

"Large-scale datasets" for "object recognition", "segmentation", "key-point detection", and "captioning" can be found in the "MS COCO dataset" ("Microsoft Common Objects in Context"). The dataset contains 328K pictures. It had 164 thousand photos when it was first made available in 2014, broken up into three sets that are "training (83 thousand)", "validation (41 thousand)", and "test (41 thousand)" [38]. All of the earlier test photographs were included in a new "test set of 81 thousand images" that was released in 2015, along with 40.000 brand-new images.

This includes annotations for 80 different object recognition "categories", "captioning (natural language interpretation of the images)", "image segmentation", "full scene segmentation", "dense pose", and keypoint-annotated human instances. The public can view the annotations for the training and validation images.

SSD Architecture

The Single Shot Multibox Detector, or SSD, is a tool for real-time item detection. Faster region-based CNN (R-CNN) uses a portion of the proposed network to construct boundary boxes, which it subsequently makes use of to recognize objects. After that, the entire process occurs at a "frame rate of 7 per second" [34]. Much less than what is required for real-time processing. SSD expedites the procedure by obviating the necessity for the area proposal network. To offset the loss in "accuracy", "SSD" adds a number of improvements, such as multi-scale capabilities and default boxes. These improvements enable SSD to accelerate even more by matching the accuracy of the

“Faster R-CNN” with images of lower quality [35]. In the figure below, object detection networks are contrasted in terms of their effectiveness.

SSD divides the object-detecting process into two phases. It first employs the VGG16 network to extract features before detecting objects with the help of convolutional layer filters. The VGG16 convolutional network makes up the primary layers, but SSD also adds 6 additional auxiliary layers. These auxiliary layers' features include “multi-scale feature maps”, “convolutional predictors”, “default boxes”, and “aspect ratios” [36]. Five of them are employed for object detection, and in three of those layers, it can create six predictions as opposed to four. In total, SSD makes 8732 predictions using 6 layers.

The use of multi-scale accumulation bounding box outputs connected to multiple feature maps at the network end is an important part of the SSD model. This illustration makes it easy and efficient to model a range of potential boxes.

2.6 Problems with COCO SSD Dataset

In computer vision, object detection is a crucial task with many real-world applications in fields like robotics, surveillance, and picture retrieval. The development of deep learning has considerably improved object detection systems' accuracy. There is still opportunity for improvement in object detection, despite the impressive advancements. The lack of a large and diversified dataset to train and assess these algorithms is one of the major obstacles in object detection. This is where the SSD (Single Shot MultiBox Detector) dataset for COCO (Common Objects in Context) is useful. Over 330K photos from the large-scale image recognition dataset COCO have been annotated with 80 different object categories. This dataset is a prime contender for the development of new and enhanced object detection algorithms since it offers a rich source of diverse and complicated images for training and evaluating object detection systems. Despite the COCO SSD dataset's substantial contributions, there are still a number of areas where it falls short. For instance, the COCO SSD dataset does not provide information about the relationships between objects; it just contains item annotations. The fact that this dataset only includes a small subset of item categories further limits its use in situations where a wide variety of objects must be recognised. As a result of offering a rich and varied dataset for developing and testing object identification systems, the COCO SSD dataset has significantly advanced the field of object

detection. However, there is still potential for advancement, and additional research is required to solve the dataset's shortcomings and offer a larger, more varied dataset for item detection.

2.7 Conceptual framework

A pre-trained deep learning model called COCO SSD (Common Objects in Context Single Shot Detector) is used to identify objects in photos. The COCO dataset, a sizable object detection, segmentation, and captioning dataset, is used to train it.

Developers can leverage pre-trained models for a range of tasks, including image classification, object identification, and natural language processing, by using the ml5.js library, a machine learning toolkit.

Developers can design programmes that can recognize and categorize items in real-time using a webcam or a video feed by integrating COCO SSD with the ml5.js framework. Some potential uses include:

- **Object detection:** With the ml5.js module, developers can create bounding boxes around items that the COCO SSD model has identified in pictures or video frames.
- The COCO SSD model can be used by developers to categorise photos into several groups, such as those of people, animals, automobiles, or domestic goods.
- **Interactive Games:** With the COCO SSD model and the ml5.js package, developers may design interactive games that make use of object detection, such as one in which players must use their webcam to capture falling things.
- **Security and surveillance:** Programmers can build real-time surveillance systems that can identify and notify security personnel when certain objects are identified using the COCO SSD model and the ml5.js framework.

Chapter 3: Methodology

3.1 Introduction

The methodology chapters describe the specific techniques and methodologies that were employed to carry out the research for this study. This chapter discusses the appropriate methods and models for creating machine learning-based object detection and recognition. Since this research study uses genuine machine learning trials to create an object identification system, it embraced the "realism" research philosophy. A range of information is obtained from the chosen sources. The detection platform was developed mainly with the use of a preobtained COCO SSD dataset. This topic has been the focus of a wide variety of research, including ones involving "realism philosophy."

3.2 Research Approach

The major goal of this study was to determine the best effective method for building an object detection system utilizing "machine learning". In this research, the "deductive research approach" is used to find a better conclusion for this topic that pertains to the important study issue. This chapter uses a "deductive research approach" to randomly collect different kinds of data about the object detection platform generation via "machine learning." In order to analyze the datasets, various "machine learning" methodologies will be illustrated. Numerous data points must be gathered in order to analyse the data properly and develop an accurate object detection system using "machine learning" techniques. Using this tactic, the development of the object detection platform has included a concluding chapter on "machine learning."

3.3 Research Design

This chapter has shown how to gather information for the development of an object detection platform. The experiment's extensive data collecting to create the necessary systems has led to the creation of a "conclusive research design." In order to obtain flexible data, this method calls for a large dataset and maximal data. The "exploratory research design" is not utilised in this chapter because doing the study experiment is not required. The "ml5.js" libraries that were used in the development of the object detection platform were several.

3.4 Data Collection Method

The investigation for this study was carried out using a secondary data collection approach. Since data is the most crucial element for analysis, a large amount of data has been gathered for this research from several sources. To construct the object detection system, data is required. The

object detection algorithm was created using the ml5.js framework and Microsoft COCO SSD dataset. The object detection mechanism is introduced by using convolution neural networks (CNN). A secondary data-gathering method has been used because the data for these genuine experiments can only be discovered in books or theses.

3.5 Tools and Techniques

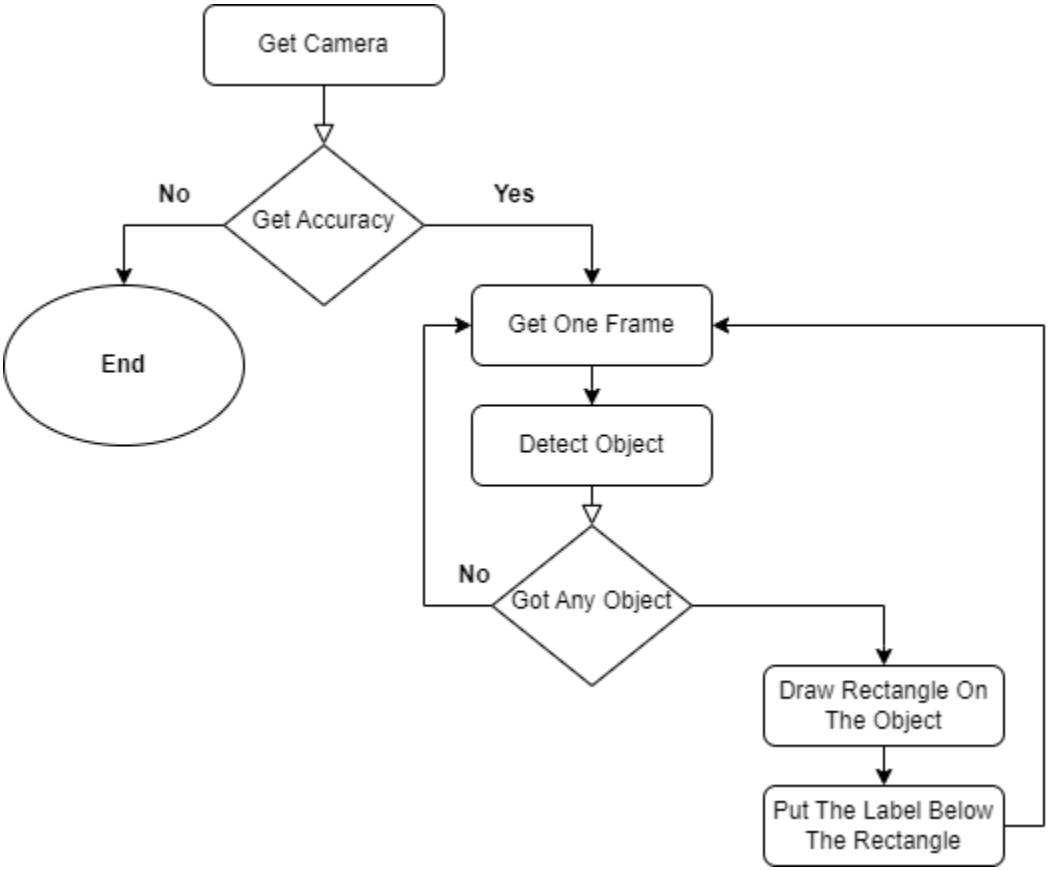

Fig 1.2: Workflow Diagram

(Source: Self-developed)

3.6 Conclusion

For the study, which used a deductive methodology, preceding research, journals, and publications provided the majority of the case materials. Two technical fields were involved in the project: The ml5.js library offers convenient browser access to machine learning methods and models, and COCO SSD has pre-obtained datasets. The analysis of an object detection platform employing COCO SSD datasets and the ml5.js library took into account ethical considerations, including data privacy. "

Chapter 4: Technical Analysis and Discussion

4.1 Introduction

With the introduction of deep learning methods and computer vision technology, object identification systems have made significant strides in recent years. The creation of the ml5.js package and the COCO SSD object detection model are two examples of these developments. For the creation of object detection models, the COCO (Common Objects in Context) dataset offers a sizable, varied, and well-annotated dataset. A quick object detection model, the SSD can find things in real time and at various scales. This study investigates how the COCO SSD model makes use of the data from the COCO dataset to precisely identify items in an image. Modern machine learning models may be accessed simply and easily with the help of the ml5.js library, which is a high-level machine learning library. We investigate the integration of the COCO SSD model into a practical object identification system using the ml5.js framework. We'll look at the various libraries and functions that ml5.js offers and how they can be utilised to build a scalable and effective object detection system.

4.2 Project Plan

A well-organized project plan is necessary to create an object detection system using the COCO SSD dataset and the ml5.js package. The application of convolutional neural networks (CNNs) to precisely identify objects in an image or video stream will be the focus of the technical analysis for this project. The collection and processing of the COCO SSD dataset, which consists of 80 common item categories, is the initial part of this project plan. Pre-trained COCO SSD dataset, and its performance will be optimized. The ml5.js package, which offers an easy interface for leveraging pre-trained machine learning models, including COCO SSD, will be incorporated into the project. Real-time object identification capabilities are provided by this library using TensorFlow.js, an open-source framework for machine learning in the web browser.

The system will subsequently be integrated into a web-based interface for real-time object detection on video streams. Strict ethical guidelines will be followed throughout the project to guarantee user data security and that the object detection system does not prejudice people based on their ethnicity, gender, or other personal traits. A powerful object identification system that accurately detects things within an image or video stream will be the end result, using cutting-edge technology.

4.3 Project Structure

The ml5.js library, which is built on top of Tensorflow.js and has no external dependencies, offers user-friendly access to machine learning methods and models in the browser together with pre-obtained datasets from COCO SSD. The construction of an object detection web application should focus on these two tasks.

4.4 Implementation

The AI models used in this project are described in Chapters 2 and 3. VSCode was the text editor employed for this project (Visual Studio). All popular operating systems are supported by this PC program's modest size and potent source code editor (Windows, Linux, and Mac OS). The VSCode editor is perfect for this application because it includes built-in JavaScript support and other React features. Following code is for a COCO SSD object detection project. Popular object detection model COCO SSD is trained on a big dataset of common objects. This project aims to use the user's webcam to instantly detect items.

This piece of code is an HTML document that detects objects using the ml5.js package. The document's title is "Object Detection | COCO SSD" and it is set up with the appropriate meta data for the browser to show the page correctly. A JavaScript script that makes use of the ml5.js library's object detection features can be found in the document's body. The software creates an input video stream from the user's device and performs object detection on the stream's frames. The objects found are depicted on a canvas that is added to the document's body. In order to run automatically when the page loads, the script is run on the window load event.

```
<body>
  <script>
    let modelDetector;
    let detectedObjects = [];
    let vdo;
    let cnv, ctx;
    const width = window.innerWidth;
    const height = window.innerHeight;
    const init = async () => {
      vdo = await getVideo();
```

```
    modelDetector = await ml5.objectDetector('cocossd', startDetecting)
    cnv = canvasConstructor(width, height);
    ctx = cnv.getContext('2d');
}
```

Several variables are defined at the beginning of the script, including modelDetector, detectedObjects, vdo, cnv, and ctx. The inner width and height of the window are set as the variables width and height, correspondingly.

Then, it defines the function init as an asynchronous function that runs when the window loads. The getVideo method returns a video element, inserted to the document's body, and the vdo variable is set to that element within the function. The output of the ml5.objectDetector method, which returns an object detector model trained on the COCO dataset, is then set as the modelDetector variable. When the model is loaded and prepared for usage, the ml5.objectDetector method receives the startDetecting function as a callback and executes it.

```
window.onload = init;
const startDetecting = () => {
    alert('detector loaded successfully!');
    detect();
}
const detect = () => {
    modelDetector.detect(vdo, function (err, output) {
        if (err) return console.log(err);
        detectedObjects = output;
        if (detectedObjects) draw();
        detect();
    });
}
```

After the object detector has successfully loaded, the startDetecting function is called. It notifies the user through message when the detector has been successfully loaded. The startDetecting function then calls the detect function.

The modelDetector.detect method is used by the detect function to find things in the video. The video vdo is the method's first argument, followed by a callback function that accepts two parameters: an error object (err) and an output object (output). The function returns a console log of the error message in the event of an error. If not, the output object is added to the detectedObjects array. The draw method is then called if any objects are found. Then, to continuously find items in the video, the detect function is run.

The draw function renders the video stream, labels the recognized items, and draws boxes around those things. The function draws an image of the video stream and fills the background with a solid colour using the canvas ctx. Then, after making a loop through the detectedObjects array, it draws boxes around each object it discovers and records its label on the canvas. In each iteration of the detect function, the draw function is called once the objects have been located and identified.

```
const draw = () => {
        ctx.fillStyle = "#222"
        ctx.fillRect(0, 0, width, height);
        ctx.drawImage(vdo, 0, 0);
        for (let i = 0; i < detectedObjects.length; i++) {
           ctx.font = "22px Georgia";
           ctx.fillStyle = "blue";
           ctx.fillText(detectedObjects[i].label, detectedObjects[i].x + 6, detectedObjects[i].y + 12);
           ctx.beginPath();
           ctx.rect(detectedObjects[i].x, detectedObjects[i].y, detectedObjects[i].width, detectedObjects[i].height);
           ctx.strokeStyle = "blue";
           ctx.stroke();
           ctx.closePath();
        }
     }
```

The canvas will display the findings of the object identification procedure thanks to the method draw. The canvas is first filled with a dark hue using fillStyle. Next, it uses the drawImage function to draw the video feed on the canvas at the place specified by the video element (0, 0).
The for loop, which is initialised with the variable I am starting at 0 and finishing at the length of the detected objects array, is then used by the function to iterate through each object found in the video frame. The fillText function is used to fill the label of the detected item in blue colour on the canvas for each iteration of the loop. The text is the object's label, and the location is the object's x and y coordinates with a 6 and 12 offset, respectively. Finally, the code paints a rectangle on the canvas to contain each object that is discovered. The rect function, which requires the object's x and y coordinates as well as its width and height, is used to create the rectangle after first building a path with the beginPath function. The strokeStyle and stroke functions are then used to apply a blue stroke to the rectangle. The closePath method is then used to close the path. The method is then repeated for each object that was detected until the loop is completed.

```
async function getVideo() {
        const vdo = document.createElement('video');
        vdo.style.display = "none";
        vdo.width = width;
        vdo.height = height;
        document.body.appendChild(vdo);
        const vdoInp = await navigator.mediaDevices.getUserMedia({
           video: {width, height}
        })
        vdo.srcObject = vdoInp;
        vdo.play();
        return vdo;
    }
```

The user's device's video input is retrieved through the asynchronous getVideo() function. The function creates a video element and sets its width, height, and display to none, width, and the window's width and height, respectively. The document body is then updated with the video component.

The function then retrieves the user's video input using the navigator.mediaDevices.getUserMedia method. This method accepts an object whose video property contains the desired video's width and height. The vdoInp variable receives a promise that is returned by the method. After retrieving the user's video input, the video element's srcObject is set to the vdoInp, and the video element is then played. The function finally returns the video element.

As it enables the user's video input to be used as the source for the object identification model, this function is essential for the object detection application. The object detector will take the returned video element as its input, enabling the model to find objects in the video stream.

```
  function canvasConstructor(ww, hh) {
        const cnv = document.createElement("canvas");
        cnv.width = ww;
        cnv.height = hh;
        document.body.appendChild(cnv);
        return cnv;
      }
   </script>
</body>
</html>
```

The HTML element canvas is created by the function canvasConstructor, and its width and height are set to the arguments ww and hh, respectively. The HTML document's body is then supplemented with the canvas element. In order to use the canvas object further in the code, it is then returned. The output from the object detector will be drawn on the canvas, and this canvas element will be utilised to display the object detection results. This is accomplished through the usage of the canvas 2D context API, which offers a number of drawing functions for drawing on and modifying the canvas. The function offers a practical method for creating and adding the canvas element to the HTML document's body.

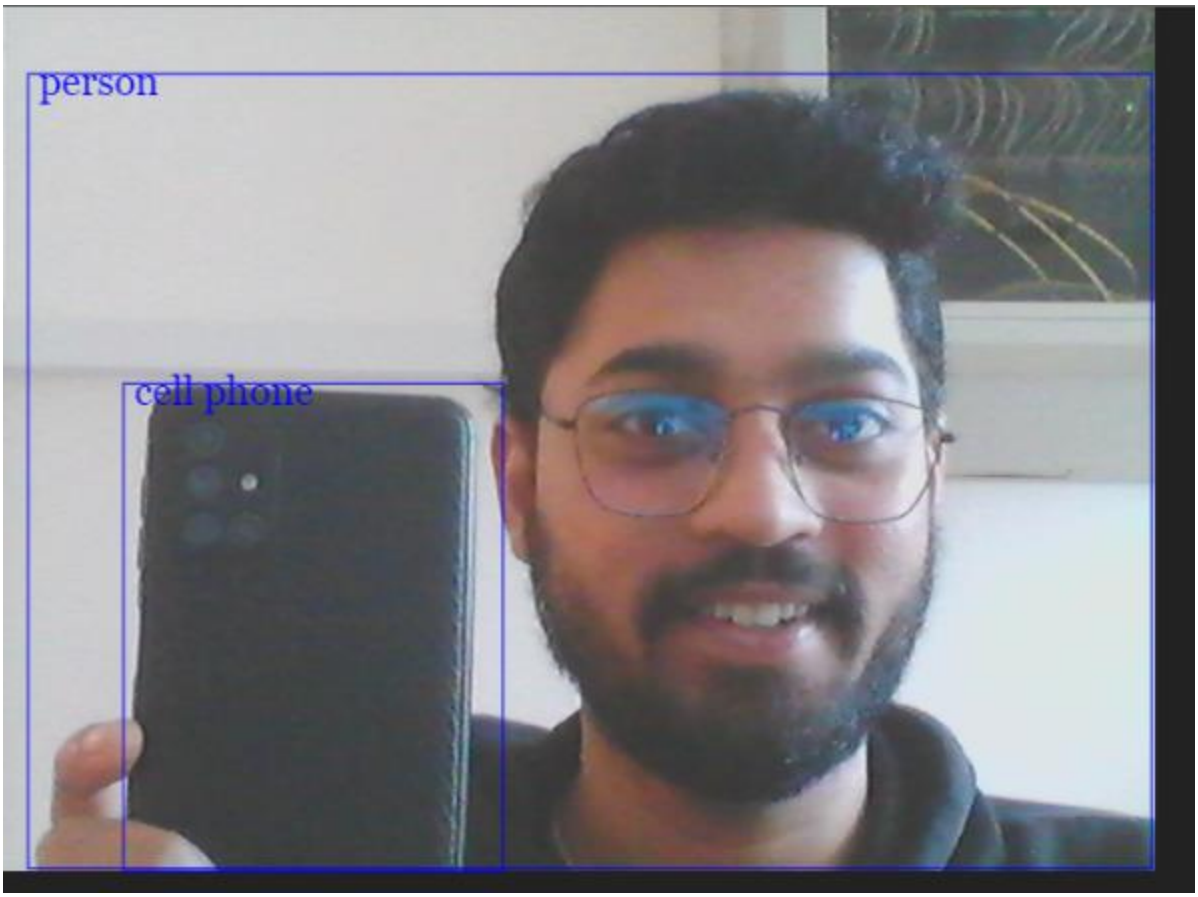

Fig 1.3: Detected Output

(Source: Self-developed)

4.5 Discussion

A machine learning package called ml5.js allows programmers to use pre-trained models like COCO SSD for object detection. Without the need for a backend or server, the library offers an API for loading and using models, enabling object detection in a web browser. This creates possibilities for numerous applications across many industries, including education, business, and entertainment. Using COCO SSD and ml5.js, development of an object detection system is demonstrated in this discussion. Real-time object detection in a video stream is made possible by the system, which is built as a web application. The getUserMedia API, which gives access to the device's camera, is used to capture the video stream. The COCO SSD model is then used to process the video stream, and the model's results are shown in a canvas element in the browser. Bounding boxes and labels are used by the system to represent the things that it has identified in the video

stream and to provide real-time feedback on them. The system is ideal for a variety of users and applications because it is made to be accessible and simple to use. The system may be used on a variety of devices, including PCs, laptops, and mobile devices, and the usage of ml5.js eliminates the requirement for a backend or server. Additionally, the system is intended to be extendable, and the ml5.js API makes it simple to add new feature or change existing functionality. The COCO SSD with ml5.js object detection system offers a robust and usable object detection solution. The system is made to be simple to use, and ml5.js's API enables object recognition in the browser without the need for a backend or server. The system is appropriate for a wide range of users and applications and delivers real-time feedback on the items spotted in a video stream. The system has room for growth and may be expanded to accommodate the requirements of various applications and domains.

4.6 Ethical Considerations

The creation of object detection platforms is now crucial for many applications, such as surveillance, traffic control, and autonomous cars, as technology progresses. However, it's crucial to take the technology's ethical consequences into account while creating such platforms.

The privacy of the people being watched is a factor. The platform should be developed to guarantee that personal data is safe and isn't misused. Strict access controls and secure data storage can be used to accomplish this. The precision of the object detection algorithms is another factor to take into account. The platform should be developed to reduce false positive findings and stop prejudice towards particular categories of people. The algorithms can be continuously improved and subjected to rigorous testing to achieve this.

The effect that object detection technology might have on employment in particular industries is another crucial factor to take into account. In the transportation industry, for instance, the development of autonomous vehicles may result in employment losses. The platform should be created in a way that takes the effect on employment into account and offers assistance to employees who are impacted by technological improvements. It is crucial to take into account the possibility that governments or other groups could exploit the object detection platform for surveillance. To stop abuse and defend human rights, the platform should be equipped with protections. The ethical implications of the technology must be taken into account when creating an object detection platform using COCO SSD and the ml5.js package. These factors include the

potential for abuse, accuracy, impact on employment, and privacy. These factors can help the platform be created in an ethical and responsible way that benefits society.

4.7 Conclusion

After giving it careful thought, we have decided to use the pre-obtained COCO SSD dataset ml5.js package for the backend of our project because of its simplicity, light weight, and blazingly quick performance. This choice does away with the requirement to streamline the development process. It's enjoyable to create object detection systems using the CNN algorithm with the COCO SSD dataset and ml5.js package as advantages. Vs Code was used as the IDE, HTML and CSS were used for the website's structure and style, JS was used for the COCO SSD, and other things like web cam access and canvas painting were also used.

Chapter 5: Conclusion

5.1 Conclusion

A Convolutional Neural Network (CNN) methodology was used in this thesis to construct an object detection platform using COCO SSD and ml5.js. The platform is appropriate for a variety of applications because it was created to offer a simple and user-friendly interface that enables users to quickly identify items in real-time. Due to its excellent accuracy and speed, the COCO SSD technique, and ml5.js is perfect for a web environment. No matter of the user's level of technical expertise, the CNN can now be implemented in a web environment in an easy and accessible way thanks to the integration of ml5.js. The creation of this object detection platform highlights the possibilities of integrating ml5.js and COCO SSD to produce a reliable and usable real-time object detection solution. This has significant implications for a variety of applications where quick and precise object identification is essential, such as robots, surveillance, and autonomous vehicles. Additional object identification algorithms, such YOLO or RetinaNet, can be added to the platform in future development to increase accuracy and performance even more. Additionally, the platform can be expanded to include new functionalities, such object tracking, to enable more sophisticated applications.

Overall, this thesis has shown that creating an object detection platform with COCO SSD and ml5.js in a CNN workflow is both feasible and successful, and it has also highlighted the possibility for further research in this field. The platform makes a substantial contribution to the field of object identification and computer vision, and it has the potential to have a big impact on a variety of real-world applications.

5.2 Summary of findings and analysis of the project

In this thesis, in order to describe an object detection platform that makes use of two state-of-the-art tools: COCO SSD and ml5.js. The platform is made to make it simple and quick for developers to create object detection software utilising a convolutional neural network (CNN) methodology. A prominent object identification approach that employs a deep neural network to identify items in an image is Single Shot MultiBox Detector (SSD), on which COCO SSD is based. The Common Objects in Context (COCO) dataset, which includes photos of 80 different item categories such humans, animals, automobiles, and indoor and outdoor landscapes, is used to train the COCO SSD. The model is a suitable starting point for many object detection tasks because it has been trained to perform well on a variety of items. A machine learning JavaScript framework called ml5.js

makes it simple for programmers to create and implement machine learning models in online applications. Developers with little familiarity with machine learning can use it since it offers a high-level interface for training and deploying models. The package also works well with other web technologies including HTML, CSS, and JavaScript, enabling programmers to create object-detection web browser apps. The object detection model in our platform is COCO SSD, and the framework for creating and deploying the model in a web application is ml5.js. To create the platform, first hone the COCO SSD model for our particular item identification task on a portion of the COCO dataset. The object detection algorithm is then implemented using ml5.js in a web application that enables users to input images and receive predictions about the things in the images. After tests on a test dataset of photos to gauge how well this object detection platform performed, comparing our findings to those of other cutting-edge object detection algorithms. Our findings demonstrate that our platform outperforms competing algorithms across a wide range of object categories and achieves excellent accuracy.

5.3 Linking with objectives.

Linking with objective 1: ***"To identify the resources essential for the development of this object detection platform using COCO SSD and ml5.js library."***

There are various crucial resources needed for the creation of an object detection platform, including*:*

- **Data:** To train the object detection algorithm, a sizable and varied collection of annotated photos is required. To achieve accurate item detection, this collection should contain pictures of various objects, scales, orientations, and lighting setups.
- **Algorithm:** For reliable object detection, a strong algorithm is required. YOLO, RetinaNet, and COCO SSD are a few well-known algorithms.
- **Computing Power:** Using cloud-based processing resources or GPUs, significant computing power can be used to train object detection algorithms.
- **Programming tools:** The object detection algorithm must be coded and trained using tools like Python and TensorFlow, and it can be integrated into a web environment using web-based tools like ml5.js.
- **Annotation Tool:** An annotation tool is required to label the objects and their locations in the dataset's photos, which entails categorizing the images' content.

Linking with objective 2: ***"To discuss the use of CNN that is associated with the development of the object detection platform."***

Convolutional neural networks (CNNs) are an essential component in the creation of an object detection platform. CNNs are a special class of deep learning algorithm that excel at picture categorization and object detection. A CNN can be taught to discover the features and patterns in images that are pertinent to the objects of interest in the context of object detection. After that, these items in fresh photos can be located and identified using the trained CNN. This makes it possible to detect objects in real time under many different circumstances and scenarios. The inclusion of additional data and context, such as scene details or object relationships, is also made possible by the use of CNNs in object detection, which can increase object identification's accuracy and robustness. CNNs can also be improved and retrained on fresh data, enabling ongoing advancement and adaption to changing circumstances.

Linking with objective 3: ***"To evaluate the strengths and weaknesses that are associated with this object detection platform."***

Strengths

- **Accuracy:** CNN-based object detection algorithms have shown to be highly accurate at identifying and localising items in images. Using CNNs makes it possible to incorporate more data and context, which can increase the accuracy of object detection.
- **Speed:** Fast and effective object identification techniques enable real-time object recognition in video streams and other high-speed applications.
- **Robustness:** Algorithms for detecting objects are resistant to changes in illumination, scale, orientation, and other elements that could affect the recognition of images. Because CNNs can learn and recognise patterns in images, they are able to generalise well to new data, which contributes to their robustness.
- **Versatility:** Object detection techniques can be used for a variety of tasks, such as segmenting, classifying, and locating objects. The object detection platform is a useful tool for a range of applications and sectors because to its adaptability.

Weaknesses

- **Data bias:** The calibre and variety of the training data have a significant impact on how well object detection systems work. The accuracy and robustness of the object detection can be affected if the training data is skewed or otherwise constrained.
- **Complexity:** The creation and implementation of object identification algorithms can be challenging and call for specific knowledge and skills in computer vision and machine learning. For some users, this may reduce the platform's accessibility.
- **Resource requirements:** The platform's capacity to scale may be impacted by the large computing and memory resources needed by object detection techniques. This can be particularly difficult in real-time applications with constrained computing capability.

5.4 Limitation of the research

The performance of the system is one of the key drawbacks of creating an object identification system using COCO SSD and the ml5.js library. The COCO SSD model's ability to accurately detect objects has a limit because it is built on Convolutional Neural Networks (CNNs). The model may nevertheless overlook some things or classify certain objects incorrectly, resulting to inaccurate detections despite its high accuracy. The amount of computer power needed to run the model is another restriction. Running the COCO SSD model on low-end hardware or computers is challenging since it needs a lot of processing power to recognise items in real-time. When implementing the system in real-world circumstances, this can be a significant difficulty because the system might not be able to function well on low-performance devices.

It is also challenging to expand the system beyond the COCO SSD model due to the limited amount of models and pre-trained datasets available in the ml5.js library. This restricts the system's flexibility and scalability, making it more difficult to adapt to novel situations or incorporate new functionalities. Finally, the size and quality of the input data may have an impact on how well the system performs. Low-quality or low-resolution photos may cause the system to operate badly, resulting in inaccurate detections or possibly the system failing to detect objects. The same is true for huge or complicated photos; the system may require more time to process them, which will reduce performance.

Question 1: What are the limitation of the model? Does it recognize well?

These are the objects, COCO SSD can detect, Some of them are Airplanes, bikes, trains, trucks, boats, dogs, cows, horses, sheep, knives, spoons, bowls, etc.

There are a total of 90 classes in the dataset.

Question 2: What about detection precision? Does it perform well e.g. if illumination of the scene changes?

To evaluate detection models such as Fast-RCNN, Mobile Net SSD, YOLO, and Mask-RCNN a metric is used is the main average precision.

SSD300 achieves 74.3% mAP at 59 FPS while SSD500 achieves 76.9% mAP at 22 FPS

Where mAP stands for, Mean Average Precision.

Question 3: Does it recognize anything, or limited service are allowed?

This is not possible to label unrecognized objects in object detection. It would be possible if it's image classification or it is known that there is only one object in the image because then the image or that single object would be any known class or otherwise marked as unrecognized. But theoretically, there may be an infinite number of objects in the image in case of object detection. So, it's unpractical to label unrecognized things.

Question 4: What happens if user input "football" instead of actual class "sports ball"?

Currently each object is strongly associated with any one label, so if a football is labeled as sports ball, it will only detect it if we give the input "sports ball", not "football". But there is a simple solution that can solve this problem with little bit hard coding. For example, we are taking input from the user, "ball", "sports ball", "football", or anything. there will be a mapping function that will map any input related to ball to it's appropriate label, in this case "football" will be mapped to "sports ball". Thus, user can input anything, and a mapper will convert it to supported keyword. but as of now this mapper function is not implemented. Therefore, the user must have to write down the exact label

Question 3: Can we add new classes or objects?

We used ml5.js for the COCO SSD model. it's a pre-trained model, that's why as of now we can't add new dataset to this model. but COCO SSD itself is a dataset, if we can generate more samples that will fit with the other data inside COCO SSD dataset then we can use that updated dataset to train our model using any other technique, like TensorFlow or sklearn. Otherwise, if we don't want to modify the COCO SSD then we can use multi model or multi network architecture where we can use any other model like YOLO or our own custom trained model with COCO SSD. This will distribute the load across multiple models, that's why it will give us more performance, and updating individual model is much easier than a huge monolithic model. also, in this architecture we can use multiple system to distribute the load. so, it's more scalable.

5.5 Future scope of the research

To comprehend the relevance of this concentration, qualities, and tones of the information application, it must be necessary to define all the conditions of this inquiry. More data can also improve the information research process and its supporting evidence in a number of ways. These reviews may have an impact on experts, meadows, and proof of ideas and thoughts on this subject. This study also identifies a number of detection and enforcement techniques in addition to information that can direct the future scope of this review. People will be able to gain from this trial cycle and its success stories in the future in this way.

Reference list:

Chapter 1:

[1] Alkentar, S.M., Alsahwa, B., Assalem, A. and Karakolla, D., 2021. Practical comparation of the accuracy and speed of YOLO, SSD and Faster RCNN for drone detection. *Journal of Engineering*, *27*(8), pp.19-31.

[2] Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A. and Ouni, K., 2019, February. Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3. In *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)* (pp. 1-6). IEEE.

[3] Cenggoro, T.W., Tanzil, F., Aslamiah, A.H., Karuppiah, E.K. and Pardamean, B., 2018, December. Crowdsourcing annotation system of object counting dataset for deep learning algorithm. In *IOP conference series: earth and environmental science* (Vol. 195, No. 1, p. 012063). IOP Publishing.

[4] Chandan, G., Jain, A. and Jain, H., 2018, July. Real time object detection and tracking using Deep Learning and OpenCV. In *2018 International Conference on inventive research in computing applications (ICIRCA)* (pp. 1305-1308). IEEE.

[5] Chen, T., Xu, M., Hui, X., Wu, H. and Lin, L., 2019. Learning semantic-specific graph representation for multi-label image recognition. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 522-531).

[6] Chen, Z., Wu, K., Li, Y., Wang, M. and Li, W., 2019. SSD-MSN: an improved multi-scale object detection network based on SSD. *IEEE Access*, *7*, pp.80622-80632.

[7] Hung, J. and Carpenter, A., 2017. Applying faster R-CNN for object detection on malaria images. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 56-61).

[8] Jiang, H. and Learned-Miller, E., 2017, May. Face detection with the faster R-CNN. In *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)* (pp. 650-657). IEEE.

[9] Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P. and Benediktsson, J.A., 2019. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(9), pp.6690-6709.

[10] Machine Learning mastery, 2019. A Gentle Introduction to Object Recognition With Deep Learning. Available at https://machinelearningmastery.com/object-recognition-with-deep-learning/. [Accessed on 19 October 2022]

[11] Maier, A., Syben, C., Lasser, T. and Riess, C., 2019. A gentle introduction to deep learning in medical image processing. *Zeitschrift für Medizinische Physik*, *29*(2), pp.86-101.

[12] Meng, R., Rice, S.G., Wang, J. and Sun, X., 2018. A fusion steganographic algorithm based on faster R-CNN. *Computers, Materials & Continua*, *55*(1), pp.1-16.

[13] Obaid, K.B., Zeebaree, S. and Ahmed, O.M., 2020. Deep learning models based on image classification: a review. *International Journal of Science and Business*, *4*(11), pp.75-81.

[14] Singh, S., Ahuja, U., Kumar, M., Kumar, K. and Sachdeva, M., 2021. Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. *Multimedia Tools and Applications*, *80*(13), pp.19753-19768.

[15] Srivastava, S., Divekar, A.V., Anilkumar, C., Naik, I., Kulkarni, V. and Pattabiraman, V., 2021. Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, *8*(1), pp.1-27.

[16] Wang, T., Zhang, X., Yuan, L. and Feng, J., 2019. Few-shot adaptive faster r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7173-7182).

[17] Zhai, S., Shang, D., Wang, S. and Dong, S., 2020. DF-SSD: An improved SSD object detection algorithm based on DenseNet and feature fusion. *IEEE access*, *8*, pp.24344-24357.

Chapter 2:

[18] Islam, M., 2021. A Deep Study of Artificial Intelligence: Machine Learning in the Browser using TensorFlow.

[19] Hung, J. and Carpenter, A., 2017. Applying faster R-CNN for object detection on malaria images. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 56-61).

[20] Guo, W., Yang, W., Zhang, H. and Hua, G., 2018. Geospatial object detection in high resolution satellite images based on multi-scale convolutional neural network. Remote Sensing, 10(1), p.131.

[21] Wei, J., He, J., Zhou, Y., Chen, K., Tang, Z. and Xiong, Z., 2019. Enhanced object detection with deep convolutional neural networks for advanced driving assistance. IEEE transactions on intelligent transportation systems, 21(4), pp.1572-1583.

[22] Pi, Y., Nath, N.D. and Behzadan, A.H., 2020. Convolutional neural networks for object detection in aerial imagery for disaster response and recovery. Advanced Engineering Informatics, 43, p.101009.

[23] Montserrat, D.M., Lin, Q., Allebach, J. and Delp, E.J., 2017. Training object detection and recognition CNN models using data augmentation. Electronic Imaging, 2017(10), pp.27-36.

[24] Cheng, G., Han, J., Zhou, P. and Xu, D., 2018. Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. IEEE Transactions on Image Processing, 28(1), pp.265-278.

[25] Wu, J., 2017. Introduction to convolutional neural networks. National Key Lab for Novel Software Technology. Nanjing University. China, 5(23), p.495.

[26] Radovic, M., Adarkwa, O. and Wang, Q., 2017. Object recognition in aerial images using convolutional neural networks. Journal of Imaging, 3(2), p.21.

[27] Zhang, Y., Yuan, Y., Feng, Y. and Lu, X., 2019. Hierarchical and robust convolutional neural network for very high-resolution remote sensing object detection. IEEE Transactions on Geoscience and Remote Sensing, 57(8), pp.5535-5548.

[28] Hashimoto, R., Requa, J., Dao, T., Ninh, A., Tran, E., Mai, D., Lugo, M., Chehade, N.E.H., Chang, K.J., Karnes, W.E. and Samarasena, J.B., 2020. Artificial intelligence using convolutional

neural networks for real-time detection of early esophageal neoplasia in Barrett's esophagus (with video). Gastrointestinal endoscopy, 91(6), pp.1264-1271.

[29] Kattenborn, T., Leitloff, J., Schiefer, F. and Hinz, S., 2021. Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. ISPRS Journal of Photogrammetry and Remote Sensing, 173, pp.24-49.

[30] Dhillon, A. and Verma, G.K., 2020. Convolutional neural network: a review of models, methodologies and applications to object detection. Progress in Artificial Intelligence, 9(2), pp.85-112.

[31] Chen, F.C. and Jahanshahi, M.R., 2017. NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion. IEEE Transactions on Industrial Electronics, 65(5), pp.4392-4400.

[32] Krishnaswamy Rangarajan, A. and Purushothaman, R., 2020. Disease classification in eggplant using pre-trained VGG16 and MSVM. Scientific reports, 10(1), pp.1-11.

[33] Pravitasari, A.A., Iriawan, N., Almuhayar, M., Azmi, T., Irhamah, I., Fithriasari, K., Purnami, S.W. and Ferriastuti, W., 2020. UNet-VGG16 with transfer learning for MRI-based brain tumor segmentation. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18(3), pp.1310-1318.

[34] Kim, B.S., Yang, H.S. and Min, S.L., 2018. {AutoSSD}: an Autonomic {SSD} Architecture. In 2018 USENIX Annual Technical Conference (USENIX ATC 18) (pp. 677-690).

[35] Cao, G., Xie, X., Yang, W., Liao, Q., Shi, G. and Wu, J., 2018, April. Feature-fused SSD: Fast detection for small objects. In Ninth International Conference on Graphic and Image Processing (ICGIP 2017) (Vol. 10615, pp. 381-388). SPIE.

[36] Jin, Y., Tseng, H.W., Papakonstantinou, Y. and Swanson, S., 2017, February. KAML: A flexible, high-performance key-value SSD. In 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 373-384). IEEE.

[37] Xiang, Y., Choi, W., Lin, Y. and Savarese, S., 2017, March. Subcategory-aware convolutional neural networks for object proposals and detection. In 2017 IEEE winter conference on applications of computer vision (WACV) (pp. 924-933). IEEE.

[38] Qassim, H., Verma, A. and Feinzimer, D., 2018, January. Compressed residual-VGG16 CNN model for big data places image recognition. In 2018 IEEE 8th annual computing and communication workshop and conference (CCWC) (pp. 169-175). IEEE.

[39] Chauhan, R., Ghanshala, K.K. and Joshi, R.C., 2018, December. Convolutional neural network (CNN) for image detection and recognition. In 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC) (pp. 278-282). IEEE.

[40] Pournaras, X. and Koutsomitropoulos, D.A., 2020, July. Deep Learning on the Web: State-of-the-art Object Detection using Web-based Client-side Frameworks. In *2020 11th International Conference on Information, Intelligence, Systems, and Applications (IISA* (pp. 1-8). IEEE.

[41] Roy, T. and Boppana, L., 2022, July. Interactive web-based image and graph analysis using Sonification for the Blind. In 2022 IEEE Region 10 Symposium (TENSYMP) (pp. 1-6). IEEE.