iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** XI **Month of publication:** November 2022

DOI: https://doi.org/10.22214/ijraset.2022.47281

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XI Nov 2022- Available at www.ijraset.com

Real Time Smart Object Detection using Machine Learning

Madapati Asha Jyothi[1], Mr. M. Kalidas[2]
[2]*Asst. Professor*
Department Of Master Of Computer Applications, Chaitanya Bharathi Institute of Technology (A)

Abstract: Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance.
A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance. In this project, we use a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the most challenging publicly available data-set, on which a object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

I. INTRODUCTION

Object detection is a well-known computer technology connected with computer vision and mage processing. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. It focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital mages and videos. There are various applications including face detection, character recognition, and vehicle calculator.

A. Problem Statement

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of mage classification, which is defined as predicting the class of the mage. A slightly complicated problem is that of mage localization, where the mage contains a single object and the system should predict the class of the location of the object in the mage (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. in this case, the input to the system will be a mage, and the output will be a bounding box corresponding to all the objects in the mage, along with the class of object in each box.

B. Literature Survey

1) Robust object tracking with online multiple instance learning

Authors: Babenko, B., Yang, M., Belongie, S

Abstract:: In this paper we address the problem of tracking an object in a video given its location in the first frame and no other information.

Recently, a class of tracking techniques called “tracking by detection” has been shown to give promising results at real-time speeds. These methods train a discriminative classifier in an online manner to separate the object from the background. This classifier bootstraps itself by using the current tracker state to extract positive and negative examples from the current frame. Slight inaccuracies in the tracker can therefore lead to incorrectly labeled training examples, which degrade the classifier and can cause drift. in this paper we show that using Multiple instance Learning (MIL), instead of traditional supervised learning, avoids these problems and can therefore lead to a more robust tracker with fewer parameter tweaks. We propose a novel online MIL algorithm for object tracking that achieves superior results with real-time performance. We present thorough experimental results (both qualitative and quantitative) on a number of challenging video clips.

©IJRASET: All Rights are Reserved

2) A review and comparison of measures for automatic video surveillance systems.

Authors: Baumann, A., Bolt z, M., Ebling, J., Koenig, M., Loos, H.S., Merkel, M., Niem, W., Warzelhan, J.K., Yu, J

Abstract: Today's video surveillance systems are increasingly equipped with video content analysis for a great variety of applications. However, reliability and robustness of video content analysis algorithms remain an issue.They have to be measured against ground truth data in order to quantify the performance and advancements of new algorithms. Therefore, a variety of measures have been proposed in the literature, but there has neither been a systematic overview nor an evaluation of measures for specific video analysis tasks yet. This paper provides a systematic review of measures and compares their effectiveness for specific aspects, such as segmentation, tracking, and event detection. Focus is drawn on details like normalization issues, robustness, and representatives. A software framework is introduced for continuously evaluating and documenting the performance of video surveillance systems. Based on many years of experience, a new set of representative measures is proposed as a fundamental part of an evaluation framework.

3) Handcrafted and Deep Trackers: Recent Visual Object Tracking Approaches and Trends

Authors: Mustansar Fiaz

Abstract: in recent years visual object tracking has become a very active research area. An increasing number of tracking algorithms are being proposed each year. t is because tracking has wide applications in various real world problems such as human-computer interaction, autonomous vehicles, robotics, surveillance and security just to name a few. in the current study, we review latest trends and advances in the tracking area and evaluate the robustness of different trackers based on the feature extraction methods. The first part of this work comprises a comprehensive survey of the recently proposed trackers. We broadly categorize trackers into Correlation Filter based Trackers (CFTs) and Non-CFTs. Each category is further classified into various types based on the architecture and the tracking mechanism. in the second part, we experimentally evaluated 24 recent trackers for robustness, and compared handcrafted and deep feature based trackers. We observe that trackers using deep features performed better, though in some cases a fusion of both increased performance significantly. In order to overcome the drawbacks of the existing benchmarks, a new benchmark Object Tracking and Temple Color (OTTC) has also been proposed and used in the evaluation of different algorithms. We analyze the performance of trackers over eleven different challenges in OTTC, and three other benchmarks. Our study concludes that Discriminative Correlation Filter (DCF) based trackers perform better than the others. Our study also reveals that inclusion of different types of regularization s over DCF often results in boosted tracking performance. Finally, we sum up our study by pointing out some insights and indicating future trends in visual object tracking field.

II. SYSTEM ANALYSIS

A. Existing system:

There has been a lot of work in object detection using traditional computer vision techniques (sliding windows, deform-able part models). However, they lack the accuracy of deep learning based techniques. Among the deep learning based techniques, two broad class of methods are prevalent: two stage detection (RCNN, Fast RCNN, Faster RCNN) and unified detection (Yolo, SSD).

Disadvantages of Existing system

1) Less prediction rate

2) less accuracy

B. Proposed System

Here we are proposed YOLOV3 and YOLOV3-TINY models, One of the important fields of Artificial Intelligence is Computer Vision the science of computers and software systems that can recognize and understand images and scenes. Computer Vision is also composed of various aspects such as image recognition, object detection, image generation, image super-resolution and more. Object detection is probably the most profound aspect of computer vision due the number of practical use cases.

Object detection refers to the capability of software systems to locate objects in an image/scene and identify each object. It has been widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and driver less cars. There are many ways object detection can be used as well in many fields of practice. Like every other computer technology, a wide range of creative and amazing uses of object detection will definitely come from the efforts of computer programmers and software developers. Getting to use modern object detection methods in applications and systems, as well as building new applications based on these methods is not a straight forward task.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XI Nov 2022- Available at www.ijraset.com

Early implementations of object detection involved the use of classical algorithms, the popular computer vision library. However, these classical algorithms could not achieve enough performance to work under different conditions.

Advantages of proposed system

1) High Accuracy
2) Very Effective Models

C. System Requirements

1) Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.
The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.
Python del 3.7 version (or)
Anaconda 3.7 (or)
Jupiter (or)
Google colab

2) Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given En thought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.
Operating system : windows, Linux
Processor : minimum Intel 3
Ram : minimum 4 gb
Hard disk : minimum 250gb.

D. System Study

Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

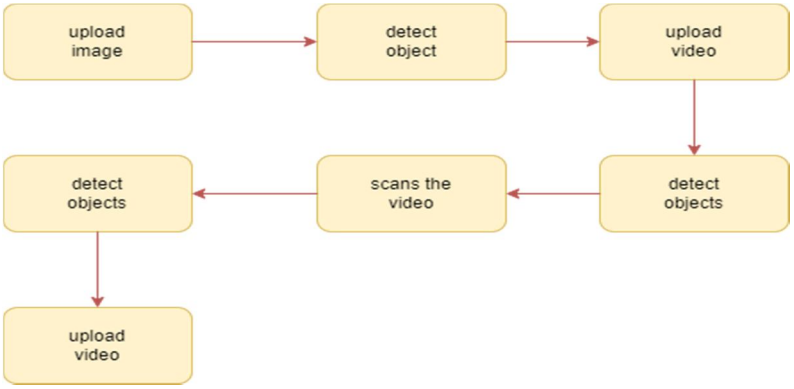
1) *Economical Feasibility:* This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.
2) *Technical Feasibility:* This study is carried out to check the technical feasibility, that s, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.
3) *Social Feasibility:* The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept t as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with t. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XI Nov 2022- Available at www.ijraset.com

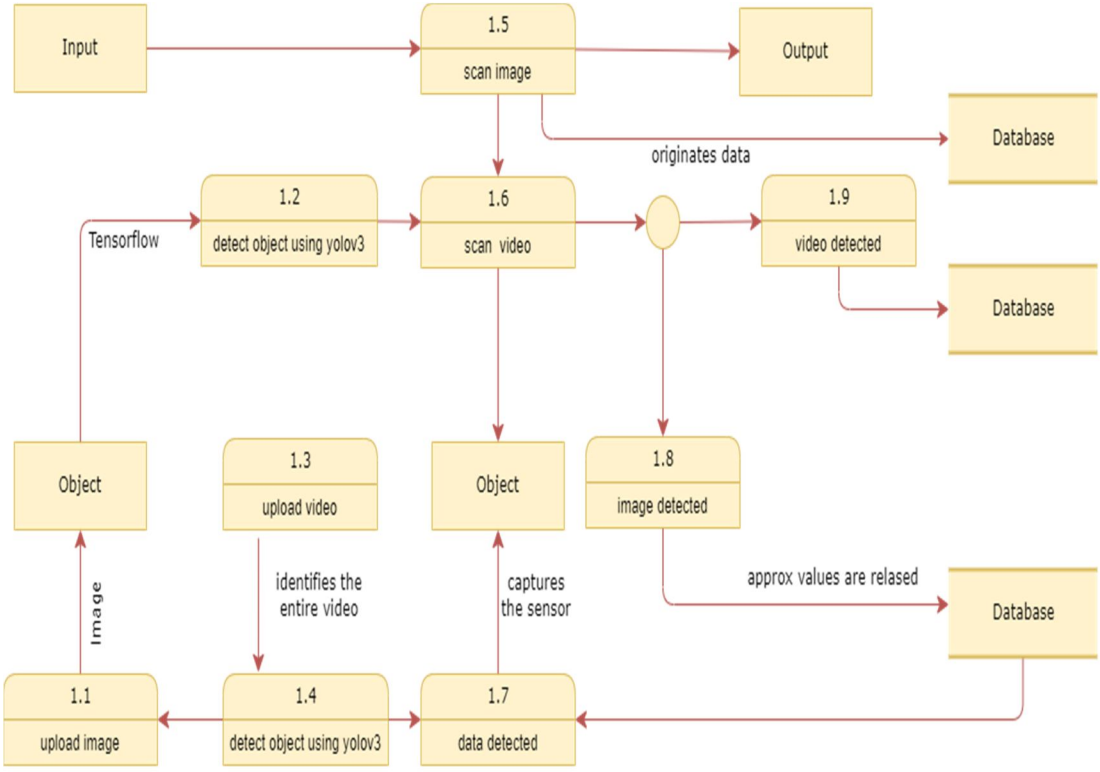
III. SYSTEM DESIGN

A. *System Architecture*



YOLOv2 improves the performance by using more anchor boxes and a new bounding box regression method.
YOLOv3 is an enhanced version of the v2 variant with a deeper feature detector network and minor representational changes. YOLOv3 has relatively speedy inference times with it taking roughly 30ms per inference.
YOLOv4 (YOLOv3 upgrade) works by breaking the object detection task into two pieces, regression to identify object positioning via bounding boxes and classification to determine the object's class. YOLO V4 and its successors are technically the product of a different set of researchers than versions 1-3.

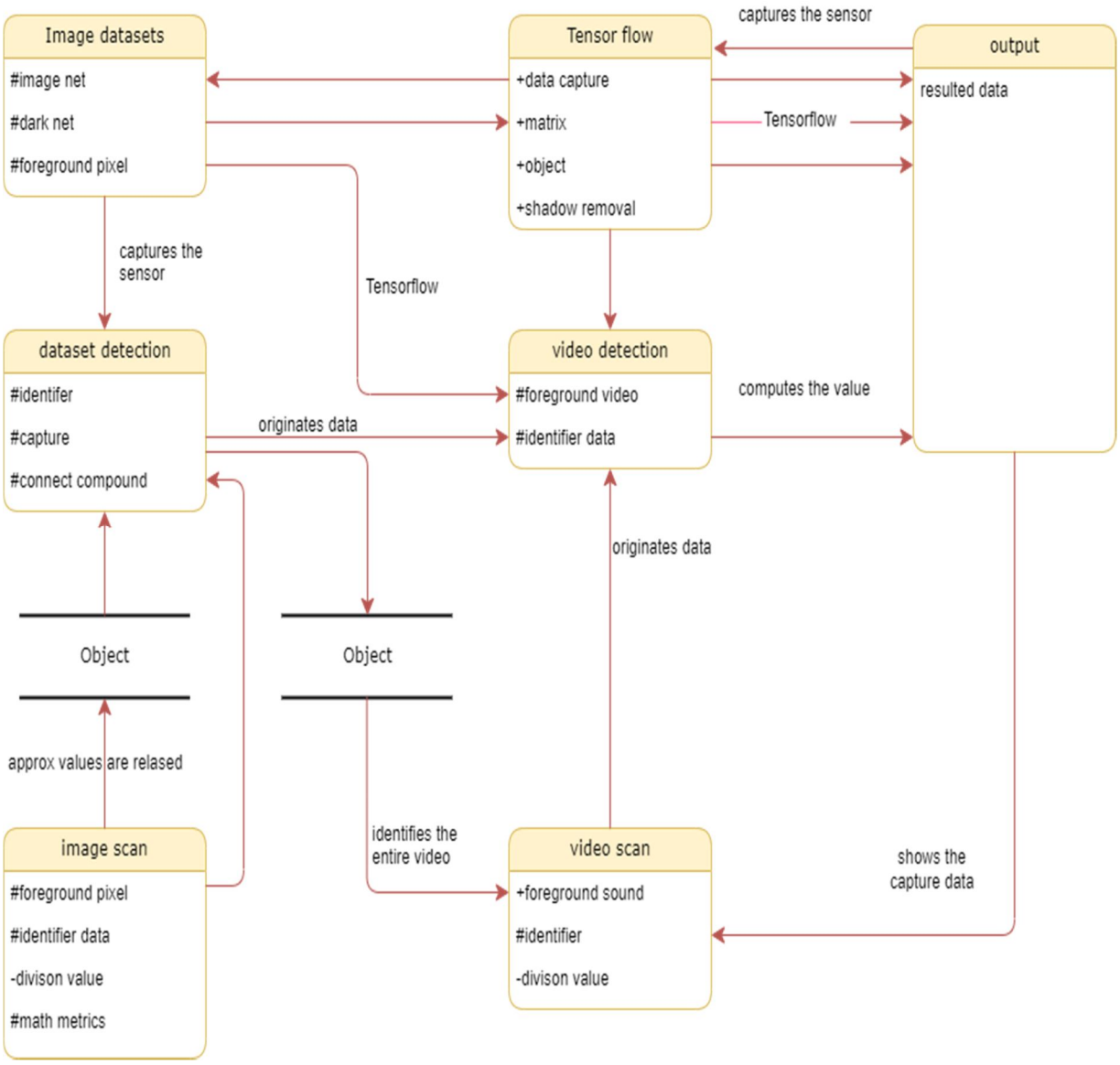
B. *DFD Level-O*



The DFD is also called as bubble chart. t is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
The data flow diagram (DFD) is one of the most important modeling tools. t is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

©IJRASET: All Rights are Reserved

C. DFD Level-1



DFD shows how the information moves through the system and how t is modified by a series of transformations. t is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

D. UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.
The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.
The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

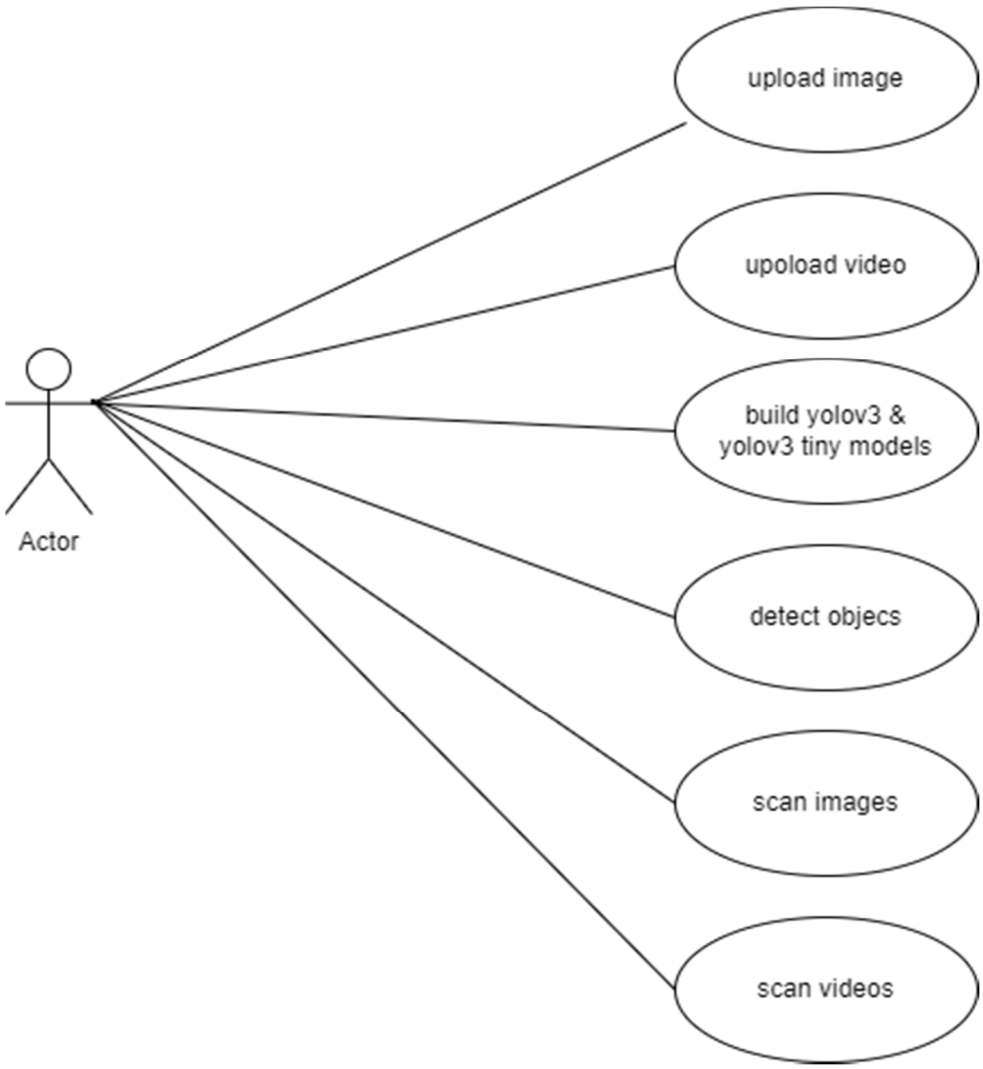
The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

1) Goals

The Primary goals in the design of the UML are as follows:

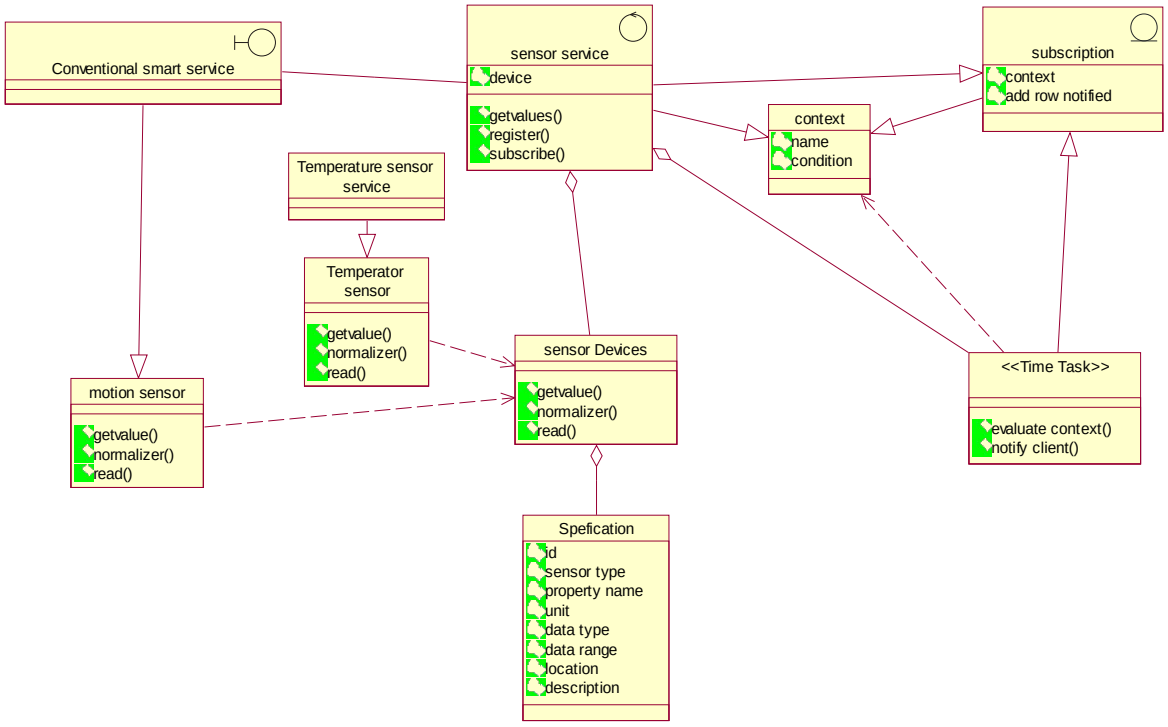
a) Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
b) Provide extensibility and specialization mechanisms to extend the core concepts.
c) Be independent of particular programming languages and development process.
d) Provide a formal basis for understanding the modeling language.
e) Encourage the growth of OO tools market.
f) Support higher level development concepts such as collaborations, frameworks, patterns and components.
g) Integrate best practices.

E. Use Case Diagram



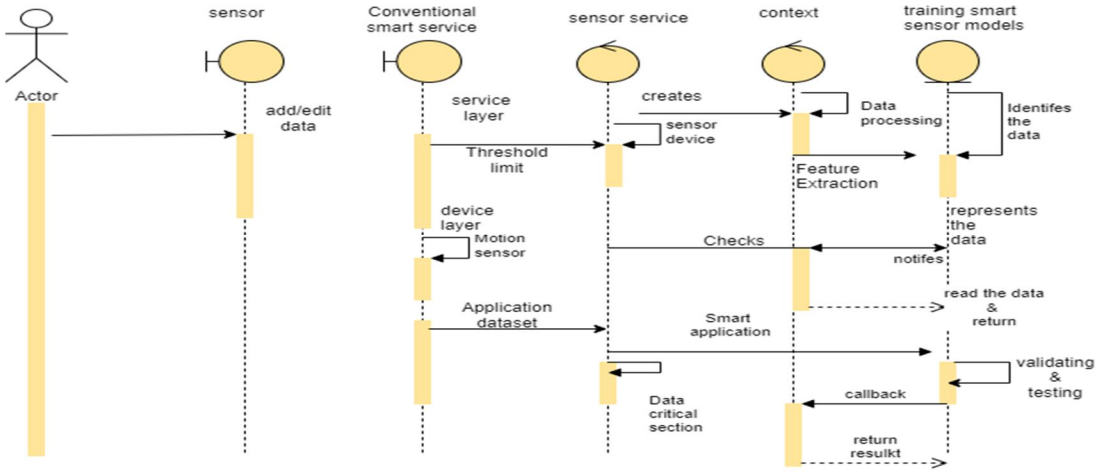
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

F. Class Diagram



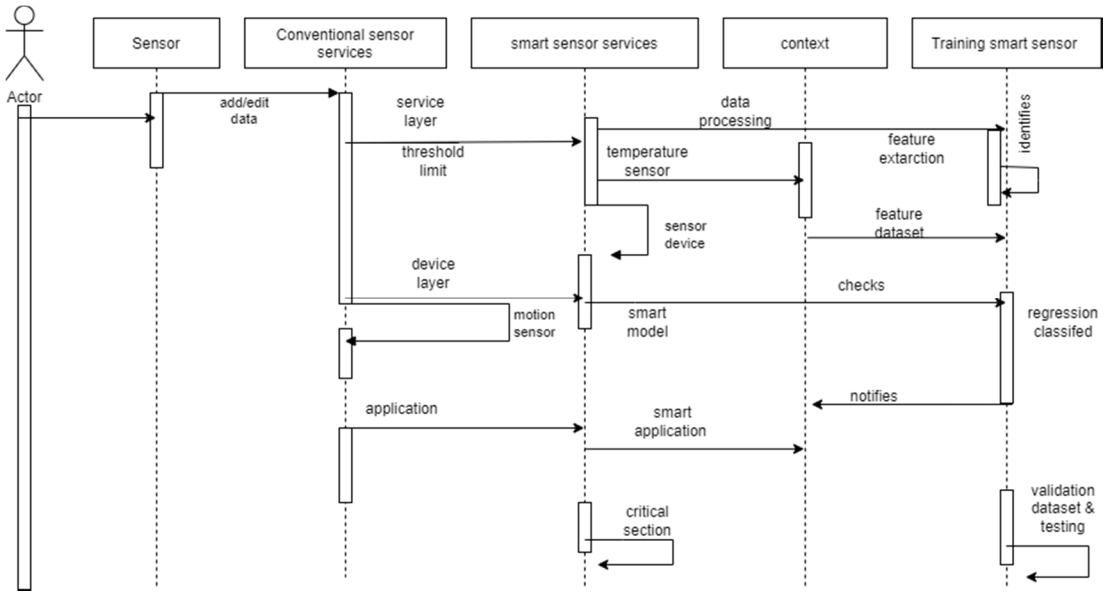
The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

G. Sequence Diagram



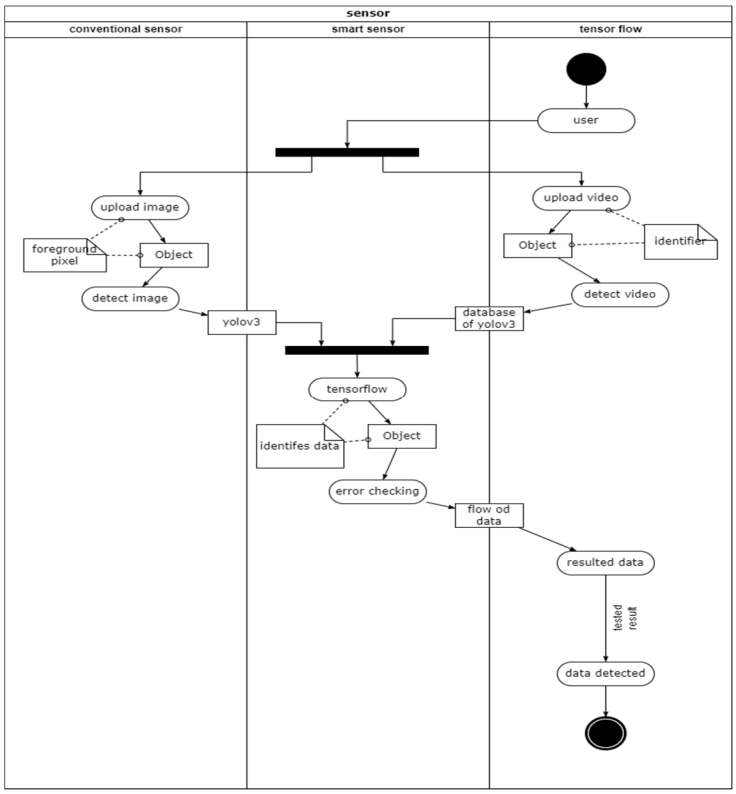
A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

H. Sequence Diagram



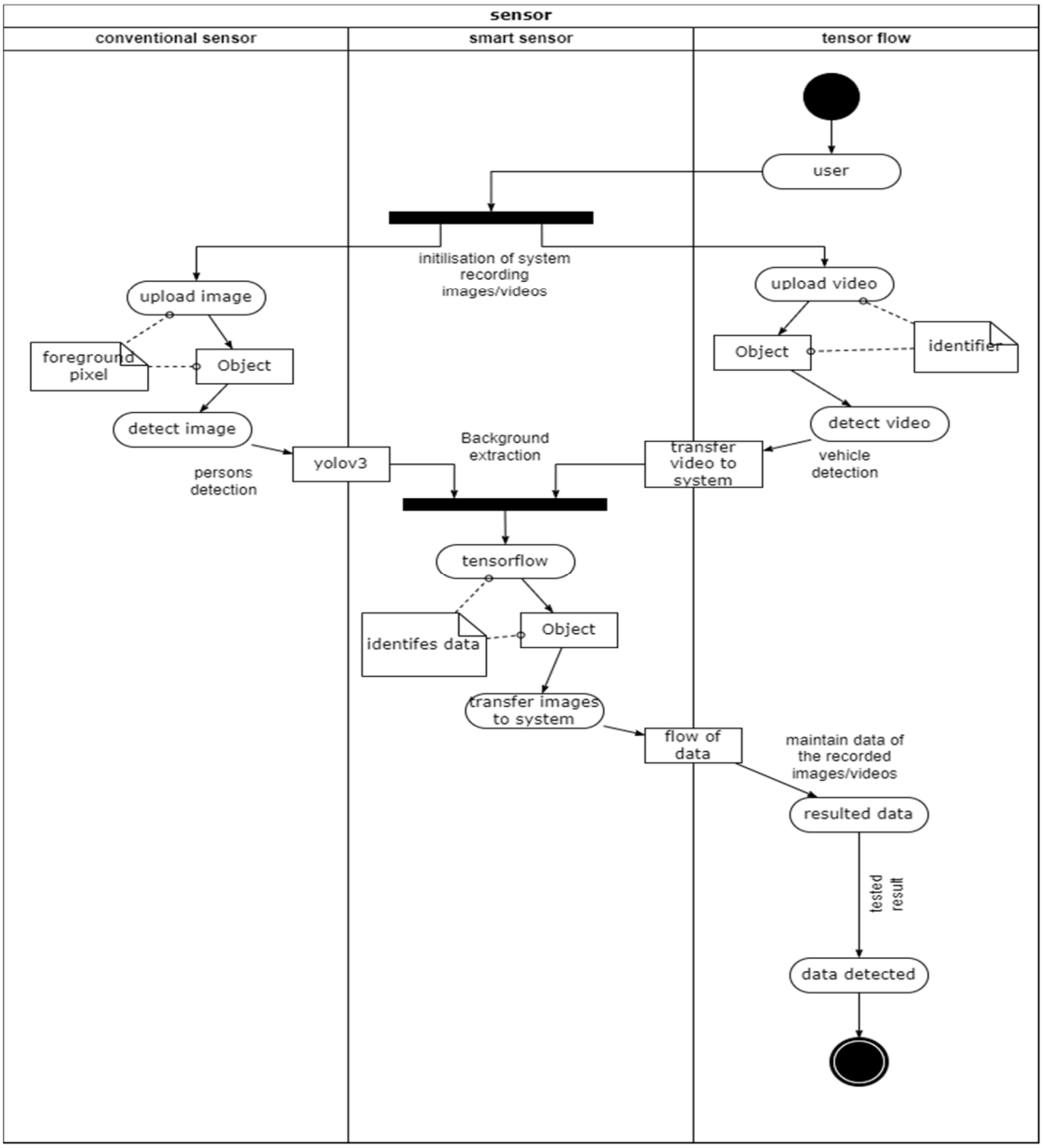
A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. Object detection is a difficult task because of illumination changes in environment, variations in target appearance, similar non-target objects in background, and occlusions

I. Activity Diagram



An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram.

J. Activity Diagram



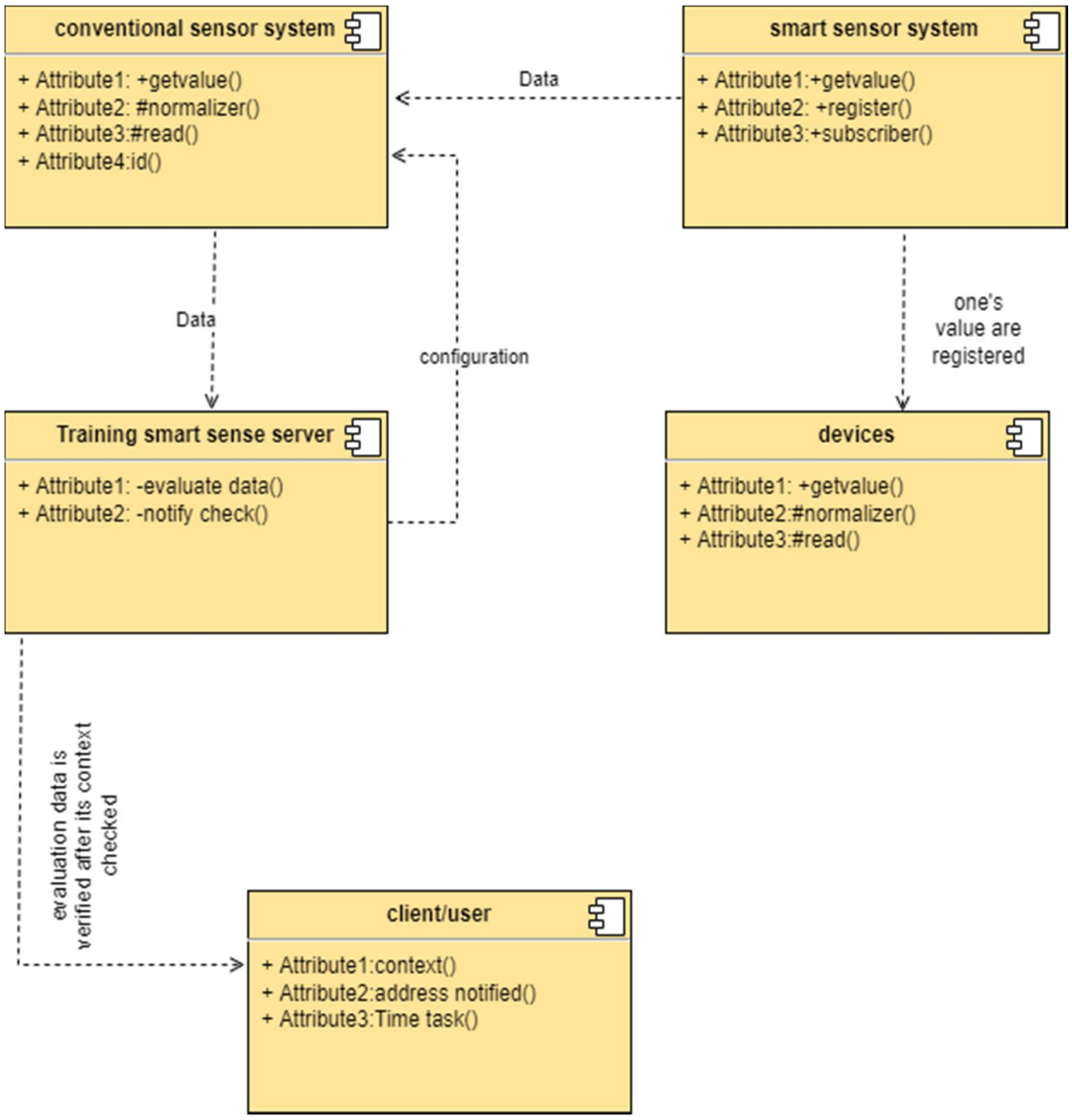
Activity diagram of object detection module The process of object detection begins by dividing the points into smaller sections of data called chunks. The chunks are created by projecting all the points into the XY plane and then dividing the plane into small regions. The regions in the X direction (azimuth) are separated by the angle from the center line and the regions in the Y direction (range) are separated by their range from the sensor. There are five regions in azimuth that span evenly across the field of view and fifteen regions in range that span 25 feet at close ranges and 100-200 feet at the farther ranges. For the points in each chunk, a plane is created that best fits the data using singular value decomposition (SVD). It is assumed that the majority of the points contained within each chunk are returns from the terrain. Thus, the fitted plane can be used to classify the points within each chunk. A standard deviation value representing the uncertainty of the radar beam width is applied above and below the fitted plane. Points that fall within three standard deviations of the plane are classified as terrain. Three standard deviations was selected because it represents 99.7% of the area of the radar beam. The points more than three standard deviations above the plane are classified as obstacles and the points more than three standard deviations below the plane are classified as below terrain. It should be noted that for radar applications, it is not unusual to see points located beneath a surface due to multi-path effects of the radar beam.

These points are usually mirrored returns from strong targets. Multi-path radar effects have been studied for over 50 years and are still being studied today . After point classification, the obstacle points are clustered using DBSCAN.

The resulting clusters are validated by comparing them with their surrounding regions in the 2D intensity image generated earlier by this module. Validation occurs by verifying a contrast in intensity of the object itself versus the background of the object. If there is a significant contrast, meaning that the cluster has a higher intensity than its surrounding area, the cluster is validated. Validated clusters are then tracked with time and can be used to determine if the object is a hazard to the vehicle.

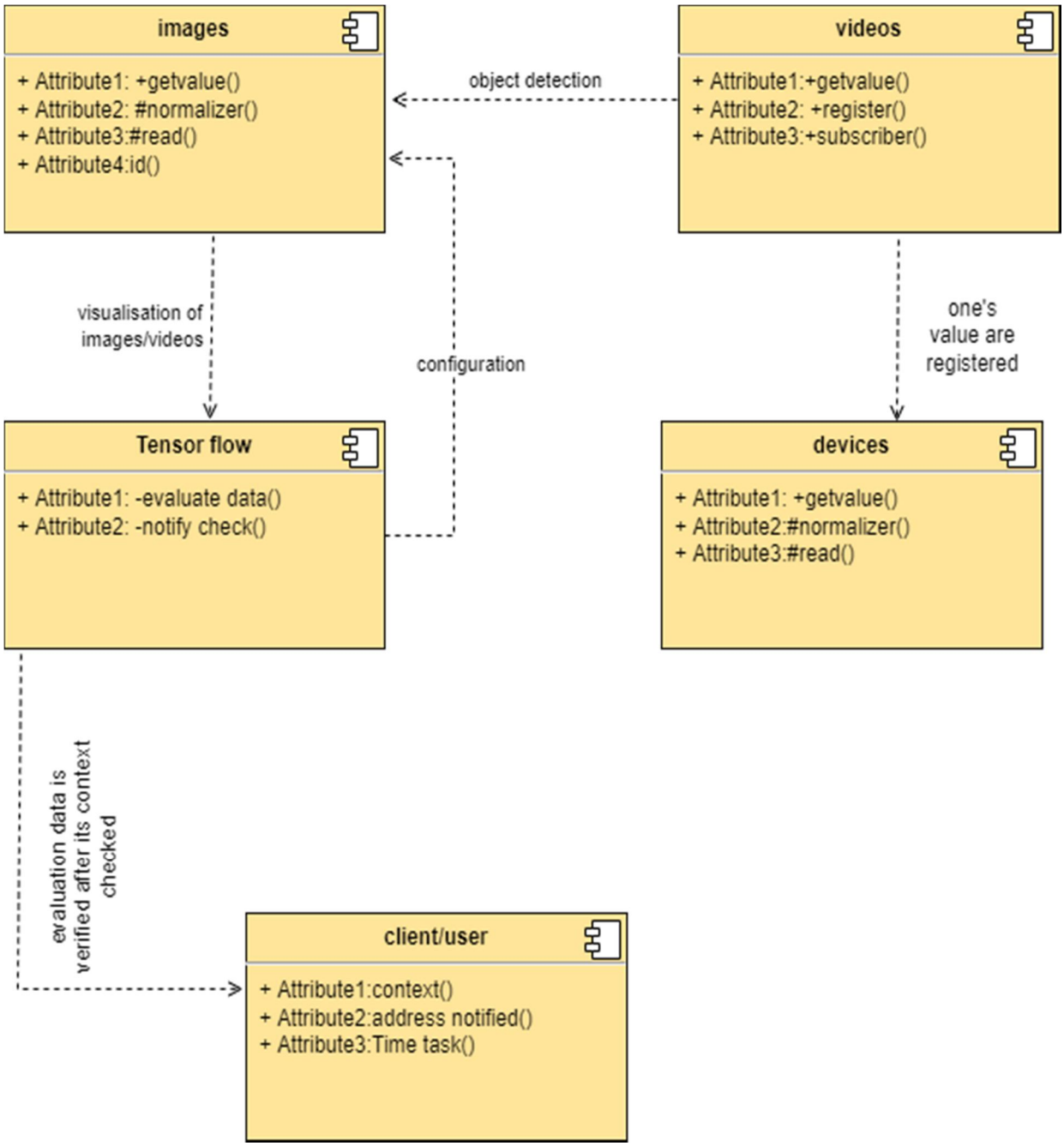
Mapping: It is not feasible to store all of the points in memory over periods of time longer than 10 seconds. The radar can theoretically generate 5,581,500 points in each 0.5 second frame. Thus, in order to maintain a history of the environment for future use, the data must be sampled into a database capable of storing data over a large physical area.

K. Component Diagram



The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

L. Component Diagram

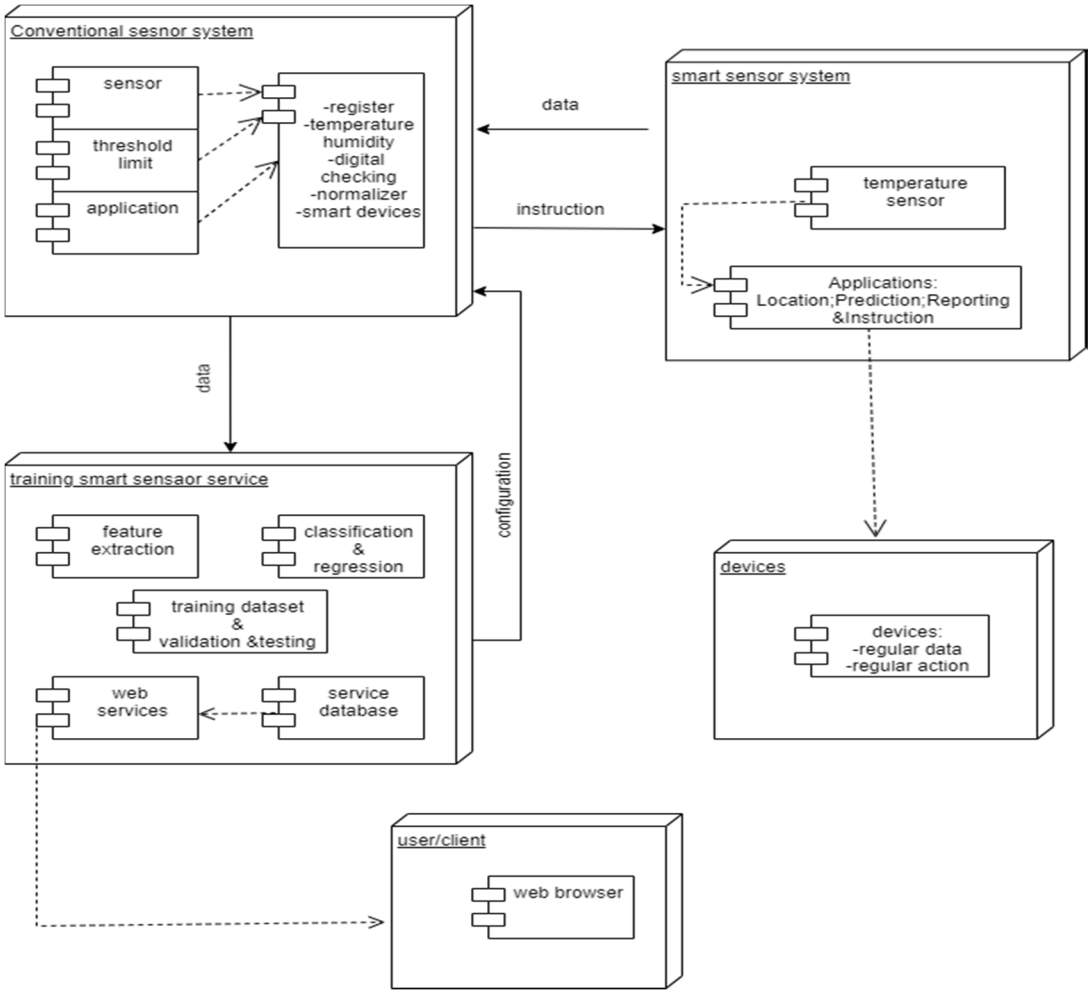


The data (account and inspection ID) flows into the component via the port on the right-hand side and is converted into a format the internal components can use. The interfaces on the right are known as required interfaces, which represents the services the component needed in order to carry out its duty.

The data then passes to and through several other components via various connections before it is output at the ports on the left. Those interfaces on the left are known as provided interface, which represents the services to deliver by the exhibiting component.

It is important to note that the internal components are surrounded by a large 'box' which can be the overall system itself (in which case there would not be a component symbol in the top right corner) or a subsystem or component of the overall system (in this case the 'box' is a component itself).

M. Deployment Diagram



A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. A line that indicates a message or other type of communication between nodes. Component: A rectangle with two tabs that indicates a software element. Dependency: A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.

IV. DESIGN PHASE

A. Modules and Functional Requirement

1) *Tensor Flow:* TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. t is a symbolic math library, and is also used for machine learning applications such as neural networks. t is used for both research and production at Google. TensorFlow was developed by the Google Brain team for Internal Google use. t was released under the Apache 2.0 open-source license on November 9, 2015. TensorFlow object detection is a computer vision technique that detects, locates, and traces an object from a still image or video. The method allows us to recognize how the models work and provides a fuller understanding of the image or video by detecting objects.
2) *Fast R-CNN (Region-based Convolutional Neural Network):* Object detection is the process of finding and classifying objects in an image. One deep learning approach, regions with Convolutional neural networks (R-CNN), combines rectangular region proposals with Convolutional neural network features. R-CNN is a two-stage detection algorithm.
3) *Single Shot Detector (SSD):* Single Shot detector like YOLO takes only one shot to detect multiple objects present in an image using multi box. It is significantly faster in speed and high-accuracy object detection algorithm. A quick comparison between speed and accuracy of different object detection models on VOC2007.

4) *YOLO (You Only Look Once):* YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.
5) *Region-based Fully Convolutional Network (R-FCN):* To achieve this, R-FCN utilities position-sensitive score maps to address a dilemma between translation-in variance in image classification and translation-variance in object detection. This R-CNN architecture uses the selective search algorithm that generates approximately 2000 region proposals. These 2000 region proposals are then provided to CNN architecture that computes CNN features. These features are then passed in an SVM model to classify the object present in the region proposal.
6) *Spatial Pyramid Pooling (SPP-net):* Spatial Pyramid Pooling (SPP) is a pooling layer that removes the fixed-size constraint of the network, i.e. a CNN does not require a fixed-size input image. Specifically, we add an SPP layer on top of the last Convolutional layer. A Pyramid Pooling Module is a module for semantic segmentation which acts as an effective global contextual prior.

B. Functional Requirements

1) Data Collection

a) *ImageNet:* Image Net data set consists of around 14 million images in total for 21,841 different categories of objects (*data as of 12th Feb 2020*). Some of the popular categories of objects in Image Net are Animal (fish, bird, mammal, invertebrate), Plant (tree, flower, vegetable) and Activity (sport).
b) *Common Objects in Context (COCO):* COCO is a large-scale object detection, segmentation, and captioning data set. It contains around 330,000 images out of which 200,000 are la-belled for 80 different object categories.
c) *Google's Open Images:* Open Images is a data set of around 9M images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. It contains a total of 16M bounding boxes for 600 object classes on 1.9M images, making it the largest existing dataset with object location annotations.

2) Data Pre-processing

a) *Read Image:* In this step, we store the path to our image data set into a variable then we created a function to load folders containing images into arrays. But first, we need to import the libraries that we are going to use for this tutorial first.
b) *Resize Image:* In this step in order to visualize the change, we are going to create two functions to display the images the first being a one to display one image and the second for two images. After that, we then create a function called processing that just receives the images as a parameter.

3) Training & Testing

a) *Step 1:* Annotate some images. During this step, you will find/take pictures and annotate objects' bounding boxes. ...
b) *Step 3:* Configuring a Training Pipeline. ...
c) *Step 4:* Train the model. ...
d) *Step 5:* Exporting and download a Trained model.

4) Modeling

Given an image or a video stream, an object detection model can identify which of a known set of objects might be present and provide information about their positions within the image.

C. Non Functional Requirements

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000.

Description of non-functional requirements is just as critical as a functional requirement.

1) Usability requirement.
2) Serviceability requirement.
3) Manageability requirement.
4) Recover-ability requirement.
5) Security requirement.
6) Data integrity requirement.
7) Capacity requirement.
8) Availability requirement.
9) Scalability requirement.
10) Interoperability requirement.
11) Reliability requirement.
12) Maintainability requirement.
13) Regulatory requirement.

D. Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. t is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

1) Emotion analysis
2) Sentiment analysis
3) Error detection and prevention
4) Weather forecasting and prediction
5) Stock market analysis and forecasting
6) Speech synthesis
7) Speech recognition
8) Customer segmentation
9) Object recognition
10) Fraud detection
11) Fraud prevention
12) Recommendation of products to customer in online shopping

a) Terminologies of Machine Learning

- *Model:* A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- *Feature:* A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- *Target (Label):* A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- *Training:* The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- *Prediction:* Once our model is ready, t can be fed a set of nputs to which t will provide a predicted output(label).

b) Types of Machine Learning

- *Supervised Learning:* This involves learning from a training data set with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.

- *Unsupervised Learning:* This involves using unlabeled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- *Semi-supervised Learning:* This involves using unlabeled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- *Reinforcement Learning:* This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

c) Advantages of Machine learning

- *Easily Identifies Trends And Patterns:* Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, t serves to understand the browsing behaviors and purchase histories of ts users to help cater to the right products, deals, and reminders relevant to them. t uses the results to reveal relevant advertisements to them.
- *No Human Intervention Needed (Automation):* With ML, you don't need to babysit your project every step of the way. Since t means giving machines the ability to learn, t lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.
- *Continuous Improvement:* As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.
- *Handling multi-dimensional and multi-variety Data:* Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.
- *Wide Applications:* You could be an e-trailer or a healthcare provider and make ML work for you. Where t does apply, t holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

d) Disadvantages of Machine Learning

- *Data Acquisition:* Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.
- *Time and Resources:* ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. t also needs massive resources to function. This can mean additional requirements of computer power for you.
- *Interpretation of Results:* Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

e) Modules Used in Project

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. t is a symbolic math library, and is also used for machine learning applications such as neural networks. t is used for both research and production at Google.

E. Algorithms

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. YOLO uses features learned by a deep Convolutional neural network to detect an object.

1) Step 1 : Download the models. We will start by downloading the models using the script file get Models. from command line.
2) Step 2 : Initialize the parameters. ...
3) Step 3 : Load the model and classes. ...
4) Step 4 : Read the input. ...
5) Step 5 : Process each frame.

The improved Tiny YOLOv3 uses K-means clustering to estimate the size of the anchor boxes for data set. The pooling and convolution layers are added in the network to strengthen feature fusion and reduce parameters. The network structure increases up sampling and down sampling to enhance multi-scale fusion.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XI Nov 2022- Available at www.ijraset.com

Tiny-yolov3 is a simplified version of YOLOv3, which has a much smaller number of convolution layers than YOLOv3, which means that tiny-yolov3 does not need to occupy a large amount of memory, reducing the need for hardware. And it also greatly speeds up detection, but lost some of the detection accuracy.

V. CODING PHASE

A. Sample Code

```
#include <fstream>
#include <sstream>
#include <iostream>
#include <opencv2/dnn.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
const char* keys =
"{help h usage ? | | Usage examples: \n\t\t./object_detection_yolo.out --image=dog.jpg \n\t\t./object_detection_yolo.out --video=run_sm.mp4}"
"{image i        |<none>| input image   }"
"{video v       |<none>| input video   }"
"{device d       |<cpu>| input device   }";
using namespace cv;
using namespace dnn;
using namespace std;
// Initialize the parameters
float confThreshold = 0.5; // Confidence threshold
float nmsThreshold = 0.4;  // Non-maximum suppression threshold
int inpWidth = 416;  // Width of network's input image
int inpHeight = 416; // Height of network's input image

vector<string> classes;
// Remove the bounding boxes with low confidence using non-maxima suppression
void postprocess(Mat& frame, const vector<Mat>& out);
// Draw the predicted bounding box
void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame);
// Get the names of the output layers
vector<String> getOutputsNames(const Net& net);
int main(int argc, char** argv)
{
Command Line Parser parser(argc, argv, keys);
parser.about("Use this script to run object detection using YOLO3 in OpenCV.");
if (parser.has("help"))
{
parser.printMessage();
return 0;
}
// Load names of classes
string classesFile = "coco.names";
ifstream ifs(classesFile.c_str());
string line;
while (getline(ifs, line)) classes.push_back(line);

string device = "cpu";
```

```
device = parser.get<String>("device");
// Give the configuration and weight files for the model
String modelConfiguration = "yolov3.cfg";
String modelWeights = "yolov3.weights";
// Load the network
Net net = readNetFromDarknet(modelConfiguration, modelWeights);
if (device == "cpu")
{
cout << "Using CPU device" << endl;
net.setPreferableBackend(DNN_TARGET_CPU);
}
else if (device == "gpu")
{
cout << "Using GPU device" << endl;
net.setPreferableBackend(DNN_BACKEND_CUDA);
net.setPreferableTarget(DNN_TARGET_CUDA);
}
// Open a video file or an image file or a camera stream.
string str, outputFile;
VideoCapture cap;
VideoWriter video;

Mat frame, blob;
try {
outputFile = "yolo_out_cpp.avi";
if (parser.has("image"))
{
// Open the image file
str = parser.get<String>("image");
ifstream ifile(str);
if (!ifile) throw("error");
cap.open(str);
str.replace(str.end()-4, str.end(), "_yolo_out_cpp.jpg");
outputFile = str;
}
else if (parser.has("video"))
{
// Open the video file
str = parser.get<String>("video");
ifstream ifile(str);
if (!ifile) throw("error");
cap.open(str);
str.replace(str.end()-4, str.end(), "_yolo_out_cpp.avi");

outputFile = str;
}
// Open the webcaom
else cap.open(parser.get<int>("device"));
}
catch(...) {
```

```
cout << "Could not open the input image/video stream" << endl;
return 0;
}
// Get the video writer initialized to save the output video
if (!parser.has("image")) {
video.open(outputFile,VideoWriter::fourcc('M','J','P','G'),28,                    Size(cap.get(CAP_PROP_FRAME_WIDTH),
cap.get(CAP_PROP_FRAME_HEIGHT)));
}
// Create a window
static const string kWinName = "Deep learning object detection in OpenCV";
namedWindow(kWinName, WINDOW_NORMAL);
// Process frames.
while (waitKey(1) < 0)
{
// get frame from the video

cap >> frame;
// Stop the program if reached end of video
if (frame.empty()) {
cout << "Done processing !!!" << endl;
cout << "Output file is stored as " << outputFile << endl;
waitKey(3000);
break;
}
// Create a 4D blob from a frame.
blobFromImage(frame, blob, 1/255.0, cv::Size(inpWidth, inpHeight), Scalar(0,0,0), true, false);
//Sets the input to the network
net.setInput(blob);
// Runs the forward pass to get output of the output layers
vector<Mat> outs;
net.forward(outs, getOutputsNames(net));
// Remove the bounding boxes with low confidence
postprocess(frame, outs);
// Put efficiency information. The function getPerfProfile returns the overall time for inference(t) and the timings for each of the
layers(in layersTimes)
vector<double> layersTimes;
double freq = getTickFrequency() / 1000;


double t = net.getPerfProfile(layersTimes) / freq;
string label = format("Inference time for a frame : %.2f ms", t);
putText(frame, label, Point(0, 15), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(0, 0, 255));
// Write the frame with the detection boxes
Mat detectedFrame;
frame.convertTo(detectedFrame, CV_8U);
if (parser.has("image")) imwrite(outputFile, detectedFrame);
else video.write(detectedFrame);
imshow(kWinName, frame);
}
cap.release();
```

```
if (!parser.has("image")) video.release();
return 0;
}
// Remove the bounding boxes with low confidence using non-maxima suppression
void postprocess(Mat& frame, const vector<Mat>& outs)
{
vector<int> classIds;
vector<float> confidences;
vector<Rect> boxes;
for (size_t i = 0; i < outs.size(); ++i)

{
// Scan through all the bounding boxes output from the network and keep only the
// ones with high confidence scores. Assign the box's class label as the class
// with the highest score for the box.
float* data = (float*)outs[i].data;
for (int j = 0; j < outs[i].rows; ++j, data += outs[i].cols)
{
Mat scores = outs[i].row(j).colRange(5, outs[i].cols);
Point classIdPoint;
double confidence;
// Get the value and location of the maximum score
minMaxLoc(scores, 0, &confidence, 0, &classIdPoint);
if (confidence > confThreshold)
{
int centerX = (int)(data[0] * frame.cols);
int centerY = (int)(data[1] * frame.rows);
int width = (int)(data[2] * frame.cols);
int height = (int)(data[3] * frame.rows);
int left = centerX - width / 2;
int top = centerY - height / 2;
classIds.push_back(classIdPoint.x);
confidences.push_back((float)confidence);

boxes.push_back(Rect(left, top, width, height));
}
}
}
// Perform non maximum suppression to eliminate redundant overlapping boxes with
// lower confidences
vector<int> indices;
NMSBoxes(boxes, confidences, confThreshold, nmsThreshold, indices);
for (size_t i = 0; i < indices.size(); ++i)
{
int idx = indices[i];
Rect box = boxes[idx];
drawPred(classIds[idx], confidences[idx], box.x, box.y,box.x + box.width, box.y + box.height, frame);
}
}
// Draw the predicted bounding box
```

```
void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame)
{
//Draw a rectangle displaying the bounding box
rectangle(frame, Point(left, top), Point(right, bottom), Scalar(255, 178, 50), 3);

//Get the label for the class name and its confidence
string label = format("%.2f", conf);

if (!classes.empty())
{
CV_Assert(classId < (int)classes.size());
label = classes[classId] + ":" + label;
}
//Display the label at the top of the bounding box
int baseLine;
Size labelSize = getTextSize(label, FONT_HERSHEY_SIMPLEX, 0.5, 1, &baseLine);
top = max(top, labelSize.height);
rectangle(frame, Point(left, top - round(1.5*labelSize.height)), Point(left + round(1.5*labelSize.width), top + baseLine), Scalar(255,
255, 255), FILLED);
putText(frame, label, Point(left, top), FONT_HERSHEY_SIMPLEX, 0.75, Scalar(0,0,0),1);
}
// Get the names of the output layers
vector<String> getOutputsNames(const Net& net)
{
static vector<String> names;
if (names.empty())

{
//Get the indices of the output layers, i.e. the layers with unconnected outputs
vector<int> outLayers = net.getUnconnectedOutLayers();

//get the names of all the layers in the network
vector<String> layersNames = net.getLayerNames();
// Get the names of the output layers in names
names.resize(outLayers.size());for (size_t i = 0; i < outLayers.size(); ++i)
names[i] = layersNames[outLayers[i] - 1];
}
return names;
}
```

B. Output

A single image:

./build/object_detection_yolo --image=bird.jpg --device=cpu

To run on Windows system, change syntax accordingly:

.\build\Release\object_detection_yolo --image=run.mp4 --device=gpu

A video file:

./build/object_detection_yolo --video=run.mp4 --device=cpu

To run on Windows system, change syntax accordingly:

.\build\Release\object_detection_yolo --video=run.mp4 --device=gpu

Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors

are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

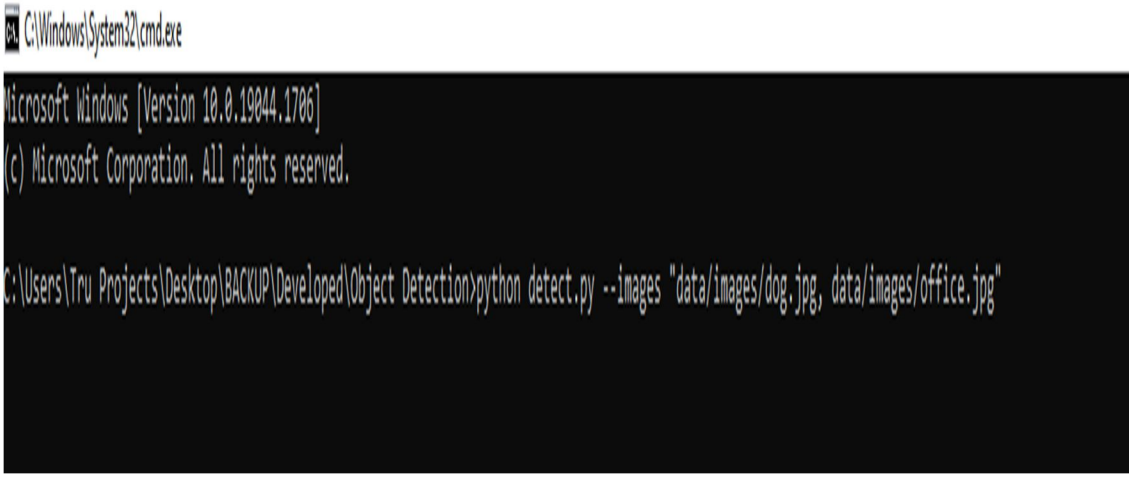
Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

Prediction – Once our model is ready, t can be fed a set of inputs to which t will provide a predicted output(label).

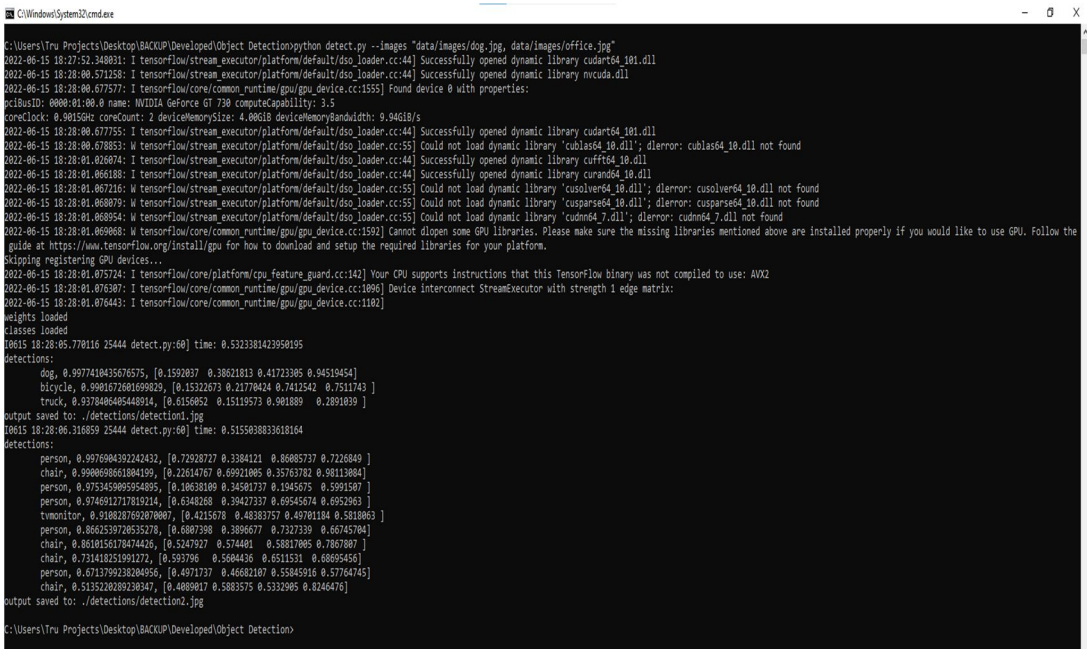
VI. SCREENSHOTS

A. For Image Detection

./build/object_detection_Yolo --image=bird.jpg --device=cpu

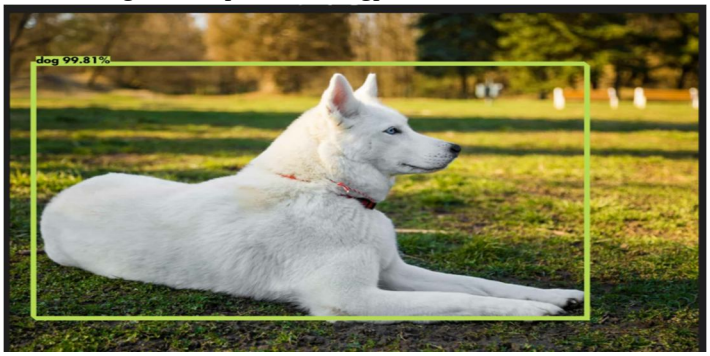


Users/project/desktop/BACKUP/developed/object detection/python detect.py --images "data/images.py



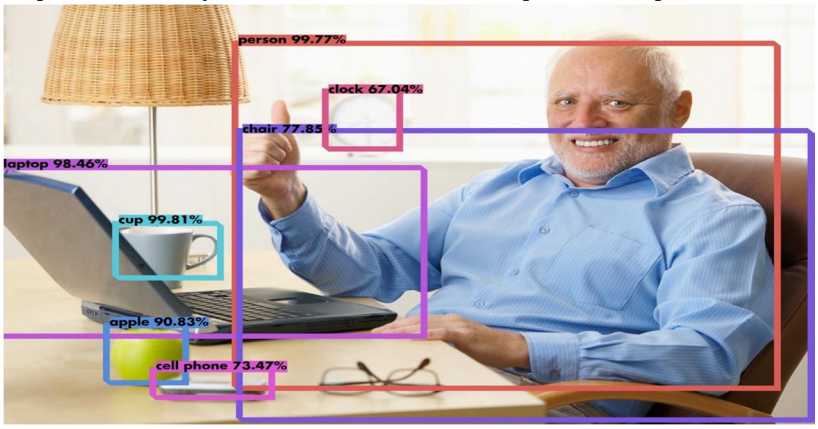
B. To Run Image

.\build\Release\object_detection_Yolo --image=run.mp4 --device=gpu



It captures the accuracy of an image as it appears on the screen and verifies while sensing and shows the value of an accurate given by the image as dog with accuracy of 99.81% depends the captured data in the image.

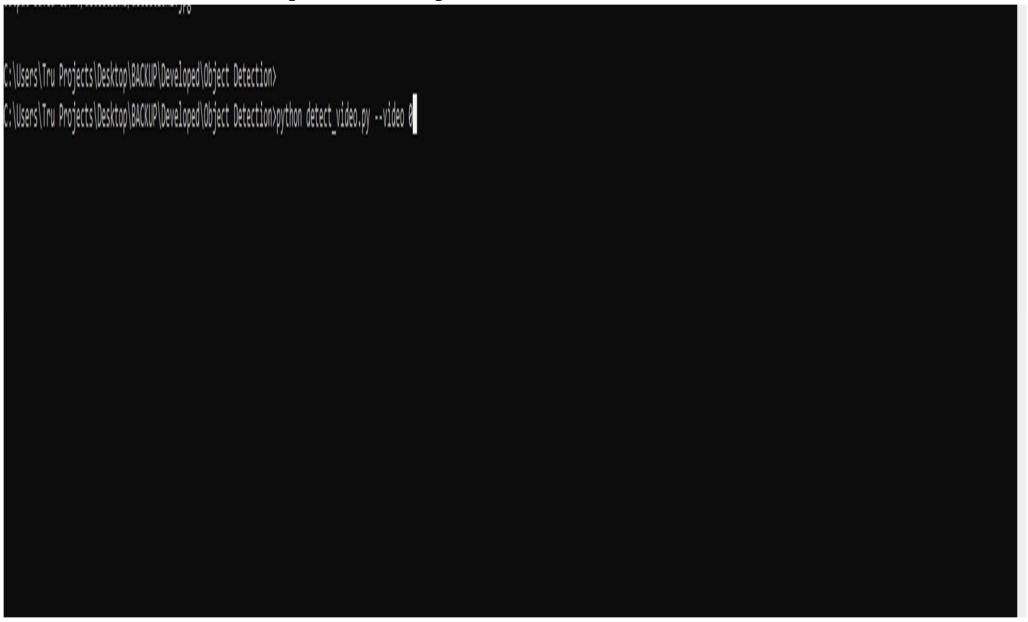
It captures the accuracy of an image as it appears on the screen and verifies while sensing and shows the value of an accurate given by the image as persons,cars,caps with accuracy of 99.81%, 85%, 93.59% depends the captured data.

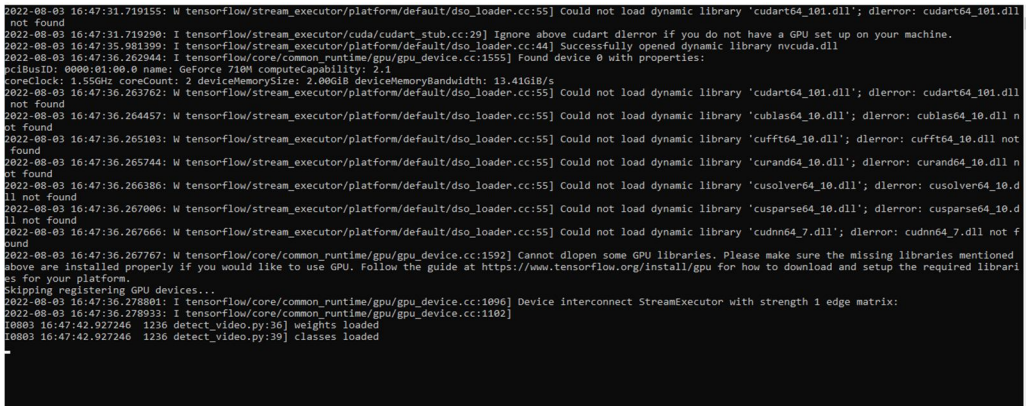


It captures the accuracy of an image as it appears on the screen and verifies while sensing and shows the value of an accurate given by the image which are saved in one folder as persons,cup,apple,cellphone,laptop,clockwith accuracy of 99.81%, 85%, 93.59% depends the captured data.

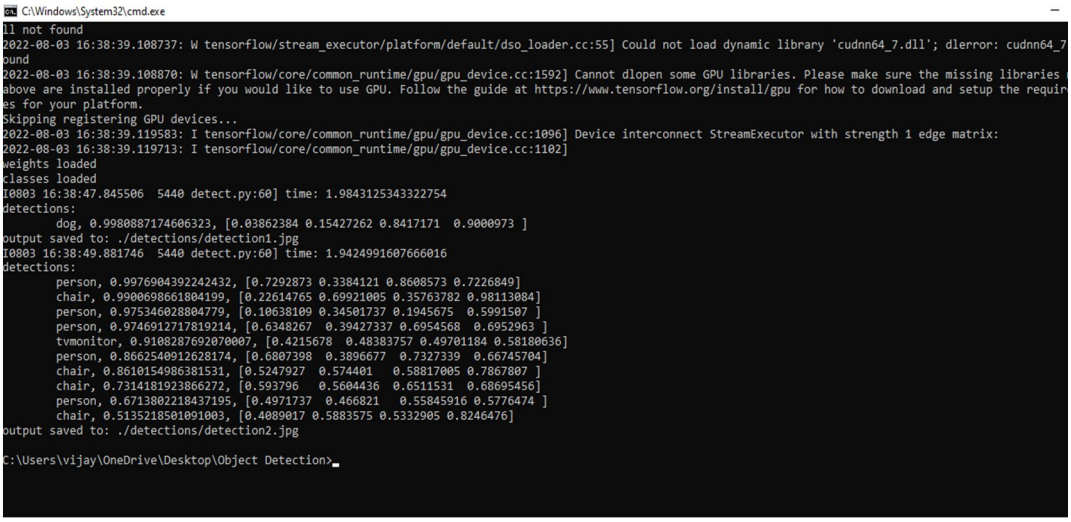
C. For Live Video

./build/object_detection_Yolo --video=run.mp4 --device=cpu



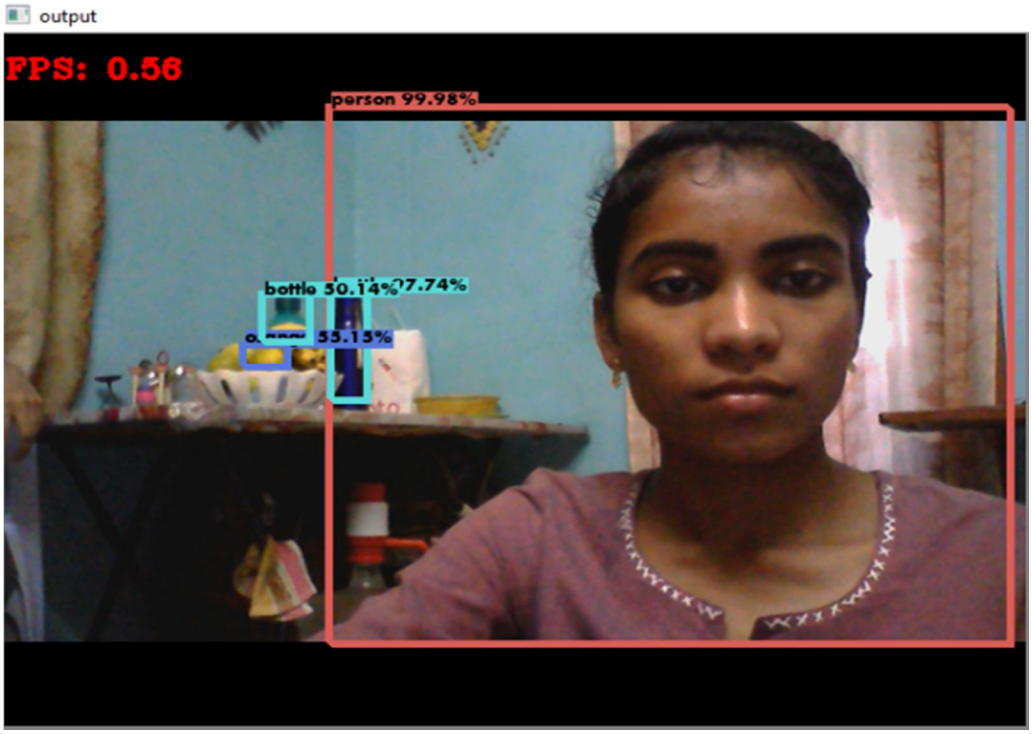


Users/project/desktop/BACKUP/developed/object detection/python detect.py --images “data/images.py



D. To Run Video

.\build\Release\object_detection_Yolo --video=run.mp4 --device=gpu

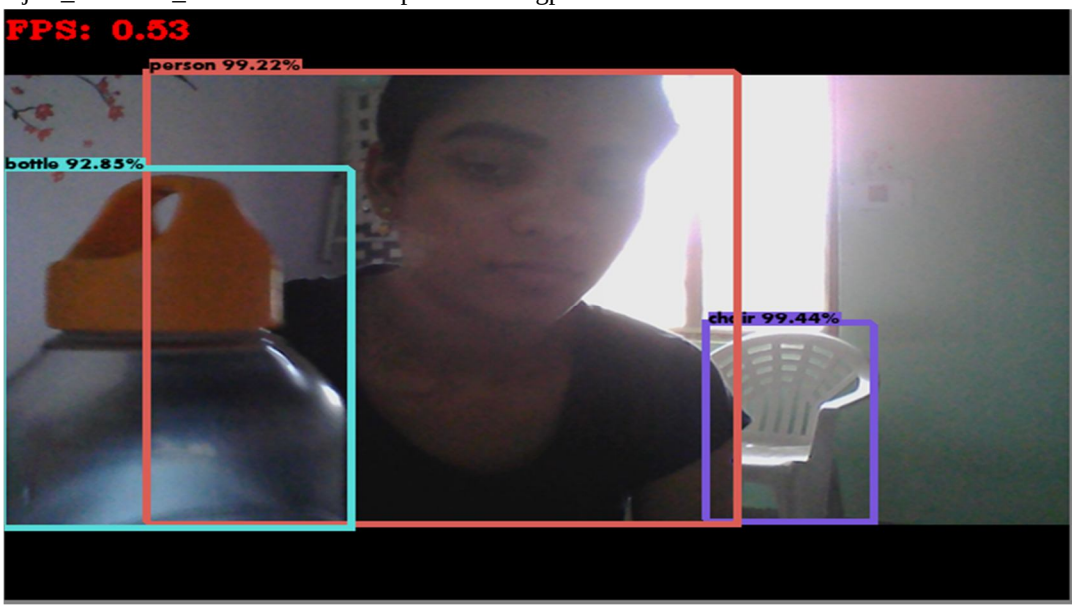


It captures the accuracy of a live video as it appears on the screen and verifies while sensing and shows the values of an accurate given by the images as displayed on the
video like persons,bottle,fruits(oranges) with accuracy of 99.81%, 85%, 93.59% depends the captured data

.

E. To Run Video

.\build\Release\object_detection_Yolo --video=run.mp4 --device=gpu



It captures the accuracy of a live video as it appears on the screen and verifies while sensing and shows the values of an accurate given by the images as displayed on the video like persons,bottle,chair with accuracy of 99.81%, 85%, 93.59% depends the captured data.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XI Nov 2022- Available at www.ijraset.com

VII. VALIDATIONS & TESTING PHASE

Tensor flow	Train the object detection model
Validation Data	Selected images verified
Expected Output	Object need to show in rectangular box
Observed output	Upload successful image in box
Remarks	The test result is verified

RCCN(Region-based Convolutional Neural Network)	(9 instead of 24) fewer filters in those layers
Validation Data	image should upload properly
Expected Output	upload successful
Observed output	Layers loaded successful
Remarks	The test result is verified

Dark-net53	foundation for object detection problems
Validation Data	Should open selected image
Expected Output	Selected image should load
Observed output	verified
Remarks	The test result is verified

Ssp (Spatial Pyramid Pooling)	removes the fixed-size constraint of the network
Validation Data	Should show exact result either in fixed size
Expected Output	result should be displayed
Observed output	Result is displayed
Remarks	The test result is verified

©IJRASET: All Rights are Reserved

VIII. CONCLUSION

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision and deep learning. Custom data set was created using labeling and the evaluation was consistent. This can be used in real-time applications which require object detection for Pre-processing in their pipeline. An important scope would be to train the system on a video sequence for usage in tracking applications. Addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.

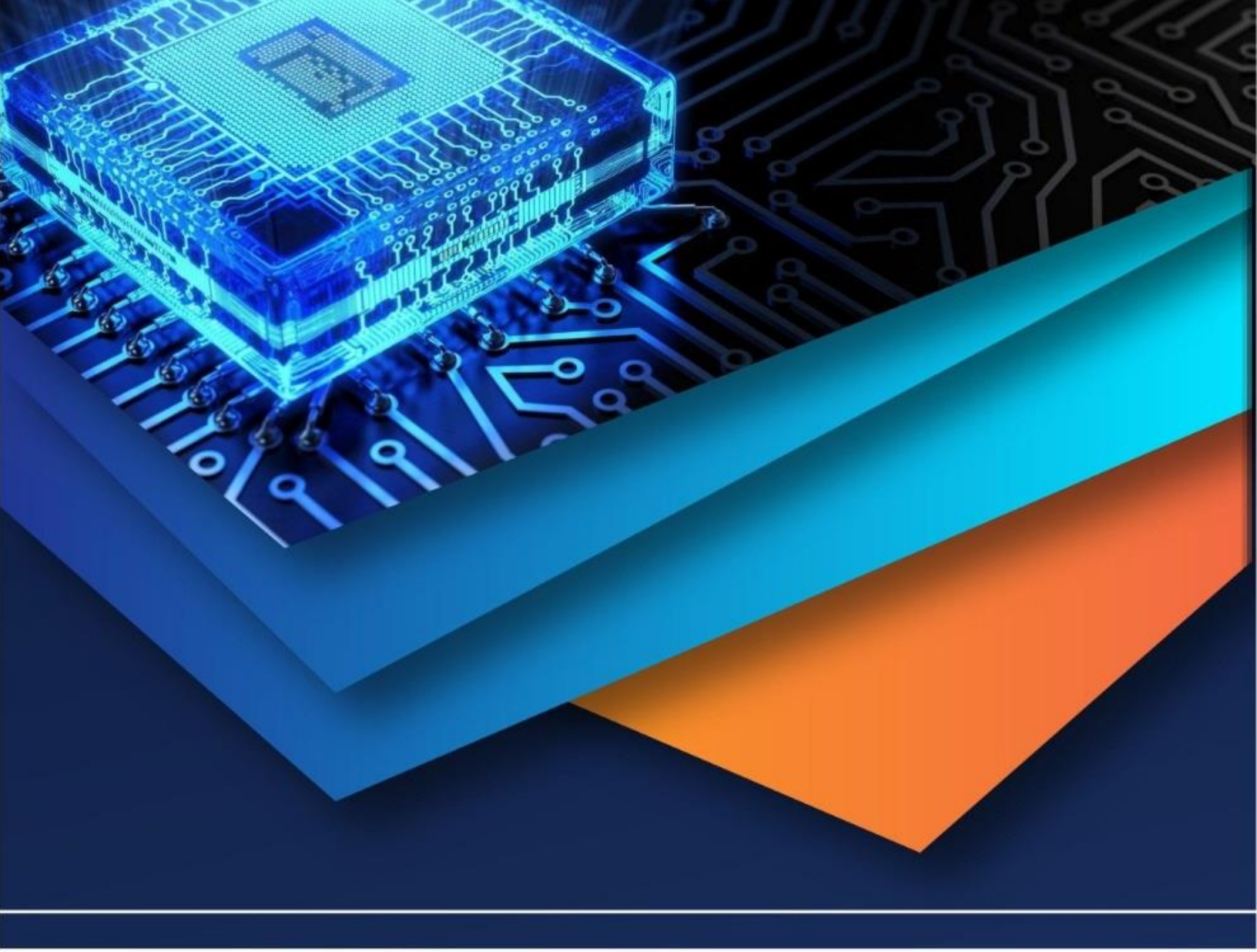
IX. FUTURE SCOPE

Computer vision is still a developing discipline, it has not been matured to that level where it can be applied directly to real life problems. After few years computer vision and particularly the object detection would not be any more futuristic and will be ubiquitous. For now, we can consider object detection as a sub-branch of machine learning.

REFERENCES

[1] Babenko, B., Yang, M., Belongie, S.: Robust object tracking with online multiple Instance learning. EEE Transactions on Pattern Analysis and Machine Intelligence 33(8), 1619– 1632 (2011) 2
[2] Baumann, A., Boltz, M., Ebling, J., Koenig, M., Loos, H.S., Merkel, M., Niem, W., Warzelhan, J.K., Yu, J.: A review and comparison of measures for automatic video surveillance systems. EURASIP Journal on mage and Video Processing 2008(824726), 1–30 (2008) 2
[3] Bhat, G., Johnander, J., Danelljan, M., Shahbaz Khan, F., Felsberg, M.: Unveiling the power of deep tracking. n: Proceedings of the European Conference on Computer Vision (ECCV), pp. 483–498 (2018) 13
[4] Cehovin, L., Leonardis, A., Kristan, M.: Visual object trac ˇ king performance measures revisited. EEE Transactions on mage Processing 25(3), 1261–1274 (2016) 2
[5] Fiaz, M., Mahmood, A., Javed, S., Jung, S.K.: Handcrafted and deep trackers: Recent visual object tracking approaches and trends. ACM Computing Surveys (CSUR) 52(2), 43 (2019) 1
[6] Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. EEE transactions on pattern analysis and machine ntelligence 37(3), 583–596 (2014) 13
[7] Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. EEE Transactions on Pattern Analysis and Machine ntelligence 37(3), 583–596 (2015) 2
[8] Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., Zhang, J.: Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics and protocol. EEE Transactions on Pattern Analysis and Machine ntelligence 31(2), 319–336 (2009)

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)