A Mini Project Report

on

Object Detection using Jetson Nano

Submitted in partial fulfillment of the

Requirements for the award of degree of

Bachelor of Technology

in

Computer Science and Engineering

by

A. Saiteja Goud
19H61A0501
A.Bhargavi
19H61A0504
A.Sumith
19H61A0506

Under the Guidance of

Dr. P. Srilatha M. Tech, Ph.D.

Asst. Professor, Department of CSE

Department of Computer Science and Engineering

ANURAG GROUP OF INSTITUTIONS

(Formerly CVSR College of Engineering)

(An Autonomous Institution, Approved by AICTE and NBA Accredited)
Venkatapur (V), Ghatkesar (M), Medchal(D)., T.S-500088

ANURAG
Engineering Engineers

Anurag Group of Institutions

An Autonomous Institution
(Formerly CVSR College of Engineering)
Venkatapur (V), Ghatkesar (M), Ranga Reddy (Dist.)
Ph: 08499953666, 08499963666. www.anurag.edu.in

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled **"OBJECT DETECTION USING JETSON NANO"** being submitted by **A. Saiteja Goud** bearing the Hall Ticket number **19H61A0501** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to **Anurag Group of Institutions (Formerly CVSR College of Engineering)** is a record of bonafide work carried out by them under my guidance and supervision from June 2022 to October 2022.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Internal Guide
Dr. P. Srilatha M. Tech, Ph.D.
Asst. Professor, Department of CSE

External Examiner

Dr.G.Vishnu Murthy,
Professor & Head, Dept. of CSE

ACKNOWLEDGEMENT

It is our privilege and pleasure to express profound sense of respect, gratitude and indebtedness to our guide **Dr. P. Srilatha,** Asst Professor, Dept. of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his/her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere gratitude to **Dr. G. Vishnu Murthy, Professor & Head**, Department of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his suggestions, motivations, and co-operation for the successful completion of the work.

We extend our sincere thanks to **Dr. V. Vijaya Kumar**, **Dean,** Research and Development, Anurag Group of Institutions, for his encouragement and constant help.

A.Saiteja Goud

19H61A0501

A.Bhargavi

19H61A0504

A.Sumith

19H61A0506

DECLARATION

We hereby declare that the project work entitled "**OBJECT DETECTION USING JETSON NANO**" submitted to the **Anurag Group of Institutions(Formerly CVSR College of Engineering)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology (B. Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Dr. P. Srilatha , Assistant Professor** and this project work have not been submitted to any other university for the award of any other degree or diploma.

A.Saiteja Goud
19H61A0501

A.Bhargavi
19H61A0504

A.Sumith
19H61A506

Date:

ABSTRACT

Object detection on embedded devices is becoming increasingly more popular in industrial use, as well as in individual use. Object detection can be used in many useful ways, such as security devices, quality assurance devices and many more. Object detection is a powerful technology to detect object in either images, videos or live video feed. This Project is introducing the reader in to the world of object detection using Nvidia Jetson Nano as a sole developing environment. The capabilities of Nvidia Jetson Nano to be used for such object detection application using live camera streaming. Introducing the tools and technologies needed to deploy object detection application on embedded device and also demonstrate of such deployment from ground up.

Keywords: Object detection, embedded device, Nvidia Jetson Nano

CONTENTS

1. INTRODUCTION

Object Detection is the process of finding and recognizing real-world object instances such as car, bike, TV, flowers, and humans out of an images or videos. An object detection technique lets you understand the details of an image or a video as it allows for the recognition, localization, and detection of multiple objects within an image. It is usually utilized in applications like image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

Object Detection is done through many ways:

- Feature Based Object Detection
- Viola Jones Object Detection
- SVM Classifications with HOG Features
- Deep Learning Object Detection

Object detection from a video in video surveillance applications is the major task these days. Object detection technique is used to identify required objects in video sequences and to cluster pixels of these objects. The detection of an object in video sequence plays a major role in several applications specifically as video surveillance applications. Object detection in a video stream can be done by processes like pre-processing, segmentation, foreground and background extraction, feature extraction. Humans can easily detect and identify objects present in an image. The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects with little conscious thought. With the availability of large amounts of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy.

Object detection is a computer vision technique that works to identify and locate objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene.

Image recognition assigns a label to an image. A picture of a dog receives the label “dog”. A picture of two dogs, still receives the label “dog”. Object detection, on the other hand, draws a box around each dog and labels the box “dog”. The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition.

1.1 MOTIVATION

The motivation behind this project is the number of benefits it provides when implemented in the Real world.

- **Traffic Monitoring and Road Maintenance**: Object detection can also be used to monitor traffic and road conditions in smart cities. Systems provide real-time data to transportation agencies about current traffic levels, potential hazards, and accidents. Object detection can be used to monitor different zones of a city.
- **People Counting** : People counting is also useful for stores and other businesses like museums to keep track of how many people enter and exit, monitoring specific areas to maintain desired capacity levels.
- **Parking Occupancy** : Object detection applications can identify open parking spaces in surface lots or parking garages. Data can be updated in real-time, alerting drivers of open spots with signs.

1.2 PROBLEM DEFINITION

Whenever you want to identify and localize objects in an image or whenever we want to do analysis on particular image or a video it is very important to detect the contents in that image and understand the class of an image. Detecting an object plays an important role in analysing any kind of image. This is used in many kind of applications such as identifying a group of people in crowded area, sensing the vehicles in traffic etc, Thus Object detection is used.

Object detection - to determine the locations of the objects and the classes to which they belong to. To accomplish this task, object detection method is used.

1.3 OBJECTIVES OF THE PROJECT

The motive of object detection is to recognize and locate all known objects in a scene. Preferably in 3D space, recovering pose of objects in 3D is very important for robotic control systems. Imparting intelligence to machines and making robots more and more autonomous and independent has been a sustaining technological dream for the mankind. It is our dream to let the robots take on tedious, boring, or dangerous work so that we can commit our time to more creative tasks. Unfortunately, the intelligent part seems to be still lagging behind. In real life, to achieve this goal, besides hardware development, we need the software that can enable robot the intelligence to do the work and act independently. One of the crucial components regarding this is vision, apart from other types of intelligences such as learning and cognitive thinking. A robot cannot be too intelligent if it cannot see and adapt to a dynamic environment. The searching or recognition process in real time scenario is very difficult. So far, no effective solution has been found for this problem. Despite a lot of research in this area, the methods developed so far are not efficient, require long training time, are not suitable for real time application, and are not scalable to large number of classes. Object detection is relatively simpler if the machine is looking for detecting one particular object. However, recognizing all the objects inherently requires the skill to differentiate one object from the other, though they may be of same type. Such problem is very difficult for machines, if they do not know about the various possibilities of objects.

- Detecting objects with clear boundaries.
- Detection objects ignoring occlusion.
- Accurate detection in live feed.

2. LITERATURE SURVEY

The research conducted so far for object detection and tracking objects in video surveillance system are disscussed in this chapter. The set of challenges outlined above span several domains of research and the majority of relevant work will be reviewed in the upcoming chapters. In this section, only the representative video surveillance systems are discussed for better understanding of the fundamental concept. Tracking is the process of object of interest within a sequence of frames, from its first appearance to its last. The type of object and its description within the system depends on the application. During the time that it is present in the scene it may be occluded by other objects of interest or fixed obstacles within the scene. A tracking system should be able to predict the position of any occluded objects. Object tracking systems are typically geared towards surveillance application where it is desired to monitor people or vehicles moving about an area. There are two district approaches to the tracking problem, top-down and another one is bottom-up. Top-down methods are goal oriented and the bulk of tracking systems are designed in this manner. These typically involve some sort of segmentation to locate region of interest, from which objects and features can be extracted for the tracking 40 Chapter 3 Literature Survey system. Bottom-up respond to stimulus and have according to observed changes. The top-down approach is most popular method for developing surveillance system. System has a common structure consisting of a segmentation step, a detection step, and a tracking step.

Table 2.1 Various other approaches

Sno	**Title**	**Author**	**Methodology Used**	**Datasets**	**Advantages**	**Limitations**
1.	Artificial Neural Networks for object Detection	J. Cruz	Ranking Prediction Algorithm used: IBGSA	CIFAR-10, Open images	IBGSA is useful in reducing extracting feature, which helps classifier for faster result.	It uses KNN which have low accuracy as a classifier

2.	Object Detector using YOLO	Joseph Redmon	Convolutional neural networking-Regression	PASCAL VOC dataset, INRIA	It makes less errors than R-CNN. YOLO sees the entire image and encodes some of the contextual information about the classes	It uses KNN which have low accuracy as a classifier.
3.	Object Detector using Mobile Net	G. Bradski	Deep learning-SSD	KITTI, COCO dataset	Computation time has been rapid due to use of faster RCNN along with VGG16	Enhancing time drastically diminishes performance
4.	Object Detector using global descriptor and sliding window	Nefi Alarcon	RANSAC – random sample consensus iterative method	DOTA , MS Coco, Image Net dataset	Its ability to do robust estimation of the model parameters.	BOF is robust to partial noises due to to that it don't produce spatial information

3. ANALYSIS

This chapter in the report gives a detailed analysis of the existing system and its drawbacks in the current world which will be followed by a brief explanation of the methodologies adopted in the proposed system along with its advantages. The software requirements specification section in the report discusses both hardware and software prerequisites needed for the project. It is further followed by the overall description of the flow of activities performed.

3.1 EXISTING SYSTEM

In an image with multiple objects, it is a challenging task to determine the location of all the individual objects (detection) and then recognize them, due to several reasons:

- There can be a possible overlap between multiple objects causing occlusions for one or all.
- Objects in the image can have varying orientations.
- The objects could only be partially present in the image.
- Images from low fps video stream can be blurry and distort the features of the object..

3.2 PROPOSED SYSTEM

JETSON NANO

Jetson Nano is a GPU-enabled edge computing platform for AI and deep learning applications. The GPU-powered platform is capable of training models and deploying online learning models but is most suited for deploying pre-trained AI models for real-time high-performance inference. Using additional Nvidia tools, deployed standard AI models can have enhanced fidelity performance**.**

3.3 SOFTWARE AND HARDWARE REQUIREMENTS

The Jetson Nano developer kit needs some packages and tools to implement the object detection and recognition task. All installations will be made for Python3.

Software Requirements

- Idle-python
- Windows 10

Hardware Requirements

- NVIDIA Jetson Nano Developer KIT
- 32GB Sandisk Ultra SD card
- Microsoft USB LifeCam 1080p HD (Webcam)

3.3.1 PURPOSE

- The main purpose of object detection is to identify and locate one or more effective targets from still image or video data. It comprehensively includes deep learning.
- To detect two dimensional and three dimensional images or moving objects.

3.3.2 SCOPE

Today, object recognition is the core of most vision-based AI software and programs. Object detection plays an important role in scene understanding, which is popular in security, transportation, medical, and military use cases.

- **Object detection in Retail**: Strategically placed people counting systems throughout multiple retail stores are used to gather information about how customers spend their time and customer footfall. AI-based customer analysis to detect and track customers with cameras helps to gain an understanding of customer interaction and customer experience, optimize the store layout, and make operations more efficient. A popular use case is the detection of queues to reduce waiting time in retail stores.
- **Autonomous Driving:** Self-driving cars depend on object detection to recognize pedestrians, traffic signs, other vehicles, and more. For example, Tesla's Autopilot AI heavily utilizes object detection to perceive environmental and surrounding threats such as oncoming vehicles or obstacles.
- **Animal detection in Agriculture**: Object detection is used in agriculture for tasks such as counting, animal monitoring, and evaluation of the quality of agricultural products. Damaged produce can be detected while it is in processing using machine learning algorithms.

- **People detection in Security:** A wide range of security applications in video surveillance are based on object detection, for example, to detect people in restricted or dangerous areas, suicide prevention, or automating inspection tasks in remote locations with computer vision.

3.3.3 OVERALL DESCRIPTION

Object detection becoming more and more popular on everyday usage. Object detection can be used for variety of purposes, for example safety critical and precise applications, but using it for more day-to-day applications that helps the community is becoming more popular and can be carried out on lighter devices. Working with neural networks used to be a very sophisticated field, but technologies are moving forward with very fast pace. Nowadays, there is many powerful pretrained neural networks models. That are available to everyone. The focus of the thesis is to investigate the elements needed for deploying object detection application on embedded device called Nvidia Jetson Nano and its capabilities to carry out object detection application. Such elements what we are going to examine are Nvidia Jetson Nano itself and its operating system JetPack. Jetson-inference, which is Nvidia's docker container consisting variety of neural network examples and tools to help guide a way to build a custom application. Take a look at programming language which is used heavily in the field of machine learning and neural networks, Python. Brief look at heuristics of SSDMobilenet. Deeper functionalities of neural networks are out of the scope of this thesis. Also going over transfer learning and model conversion.

4. DESIGN

ARCHITECTURE DIAGRAM

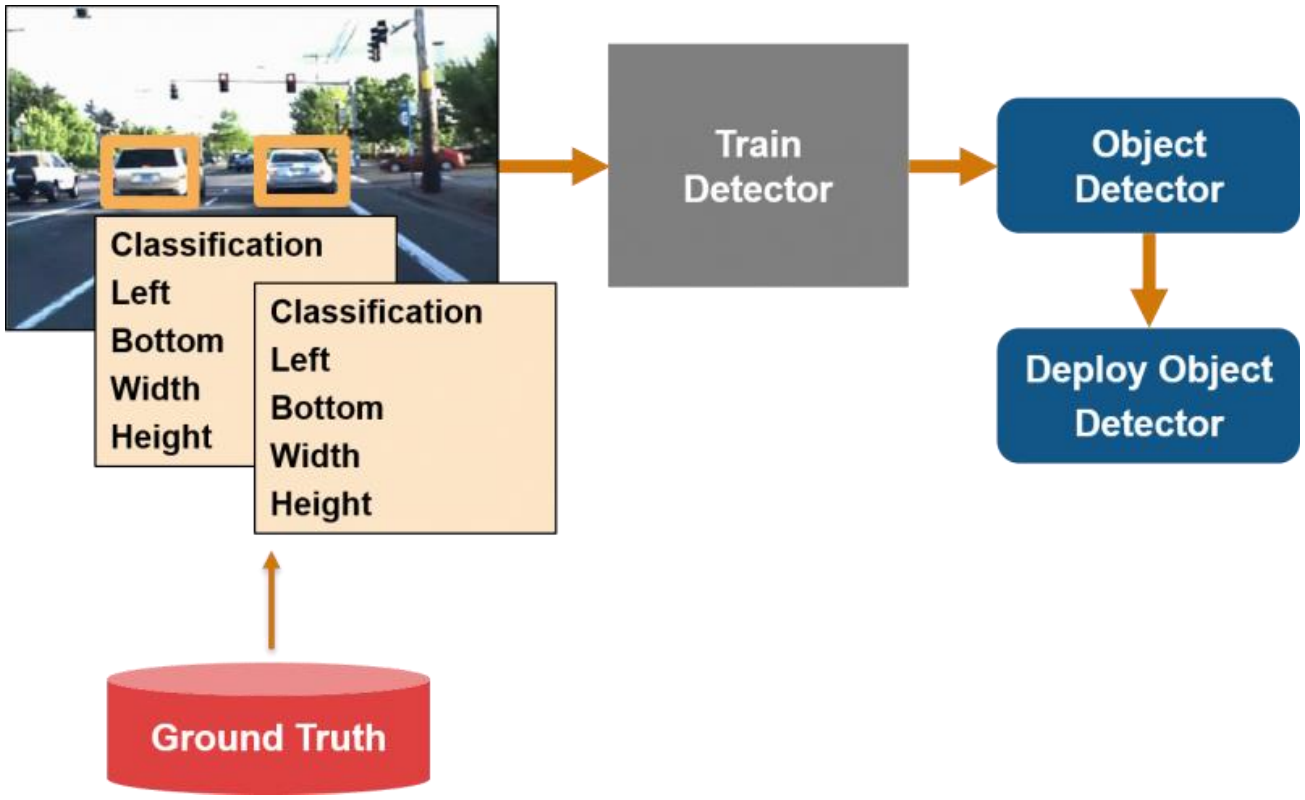


Figure 4.1 Architectural diagram for proposed model

In this model where a video or image is taken as the input and the objects in the images are classified and the required parameters are taken for training the edge device and the objects detector comes into the action for detecting the objects in input. Here input maybe preloaded images or live camera can also be used.

4.1 UML DIAGRAMS

We have represented the design of the system with the help of UML diagrams. The following UML diagrams visually represent the system along with its main actors, roles, actions, and classes. They help in easily understanding the overall system.

4.1.1 USECASE DIAGRAM

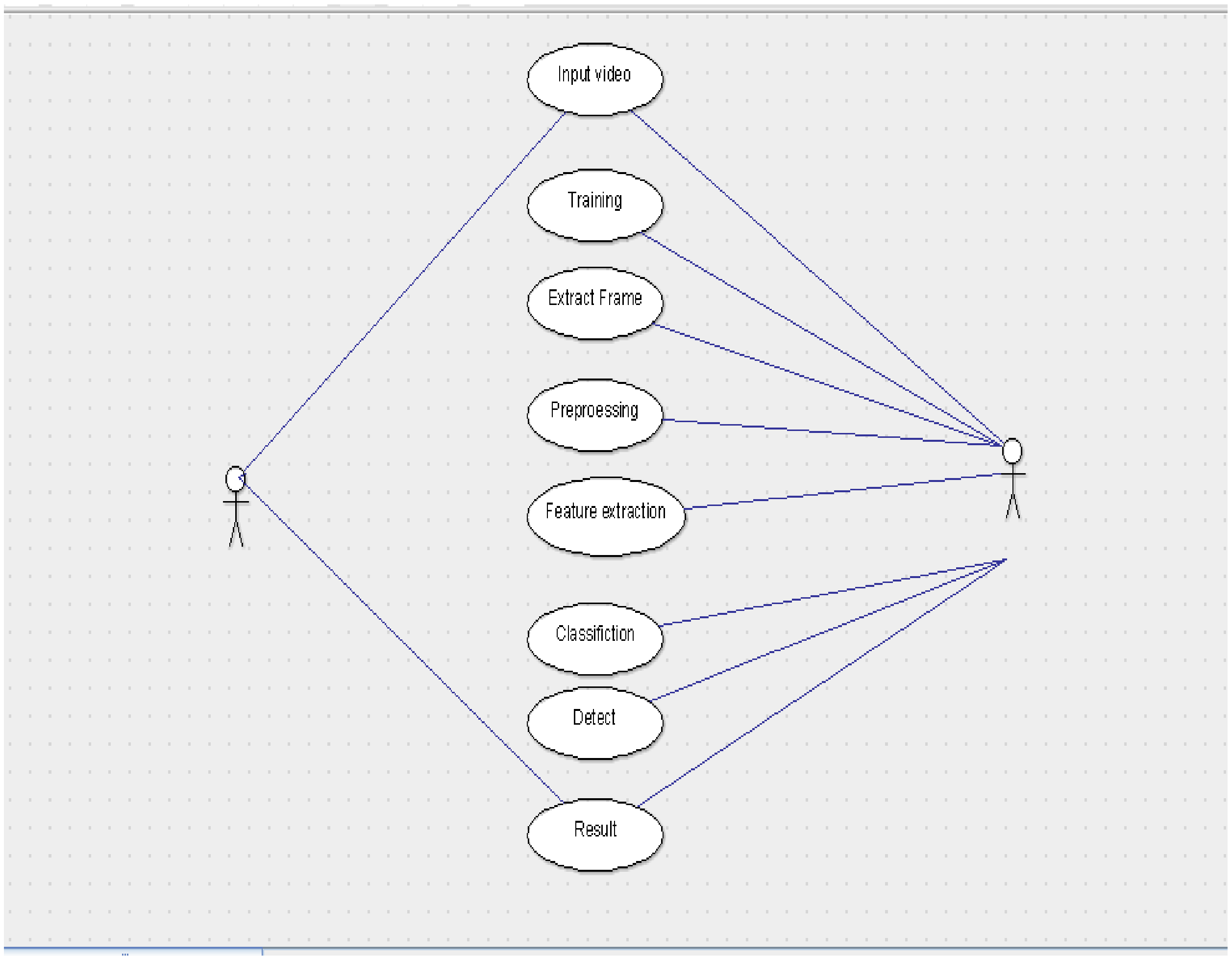


Figure 4.1 Use Case Diagram for Object Detection

4.1.2 CLASS DIAGRAM

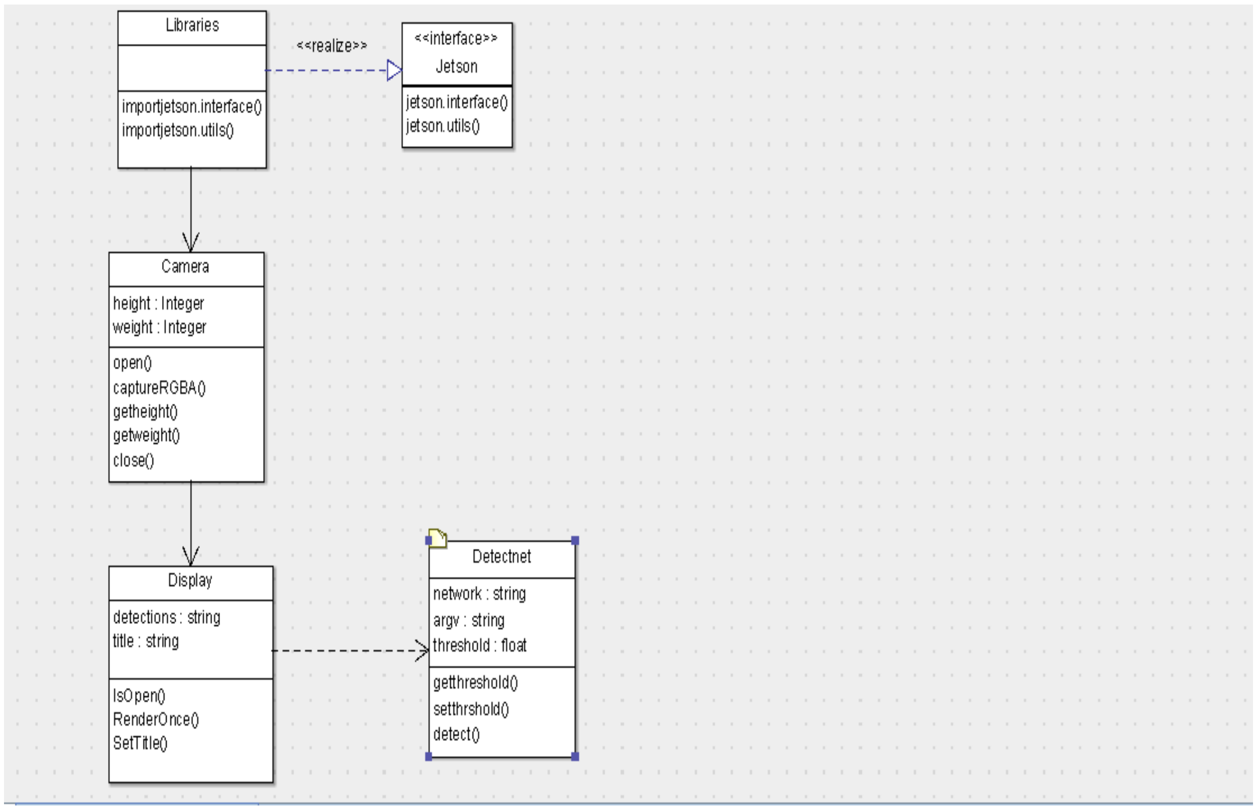


Figure 4.2 Class diagram for Object Detection

4.1.3 ACTIVITY DIAGRAM

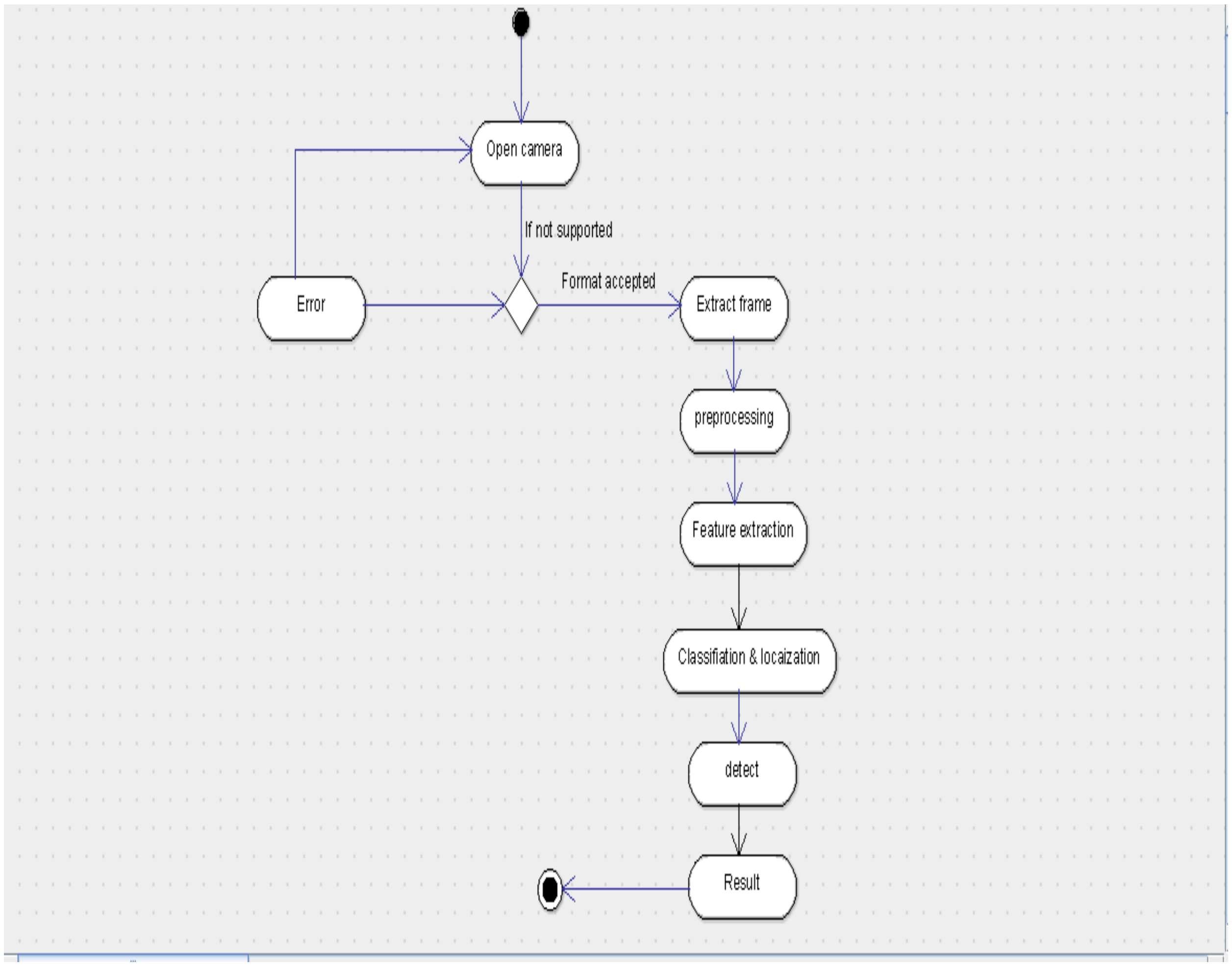


Figure 4.3 Activity Diagram for Object Detection

4.1.3 SEQUENCE DIAGRAM

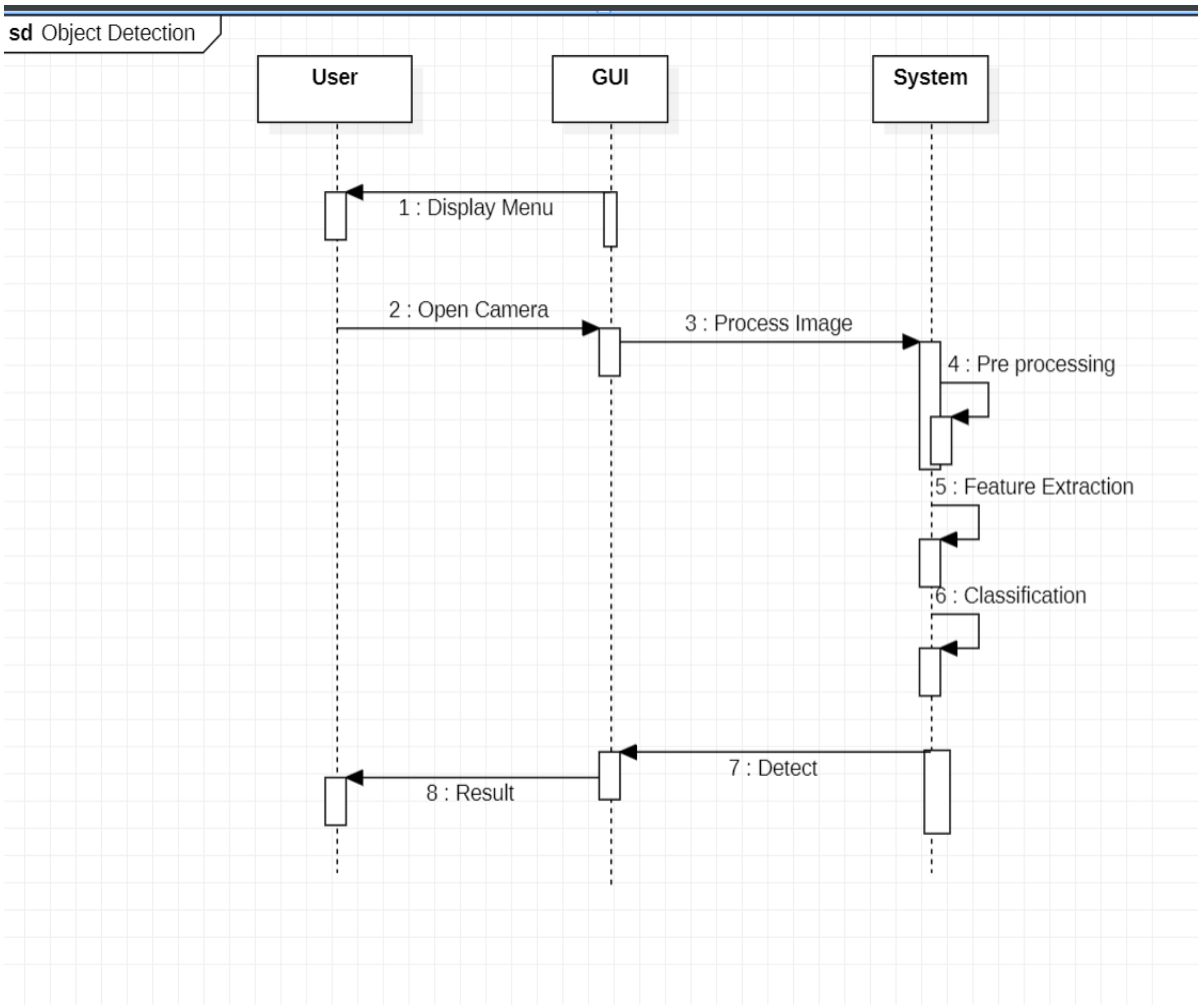


Figure 4.4 Sequence Diagram for Object Detection

4.1.4 COLLABORATION DIAGRAM

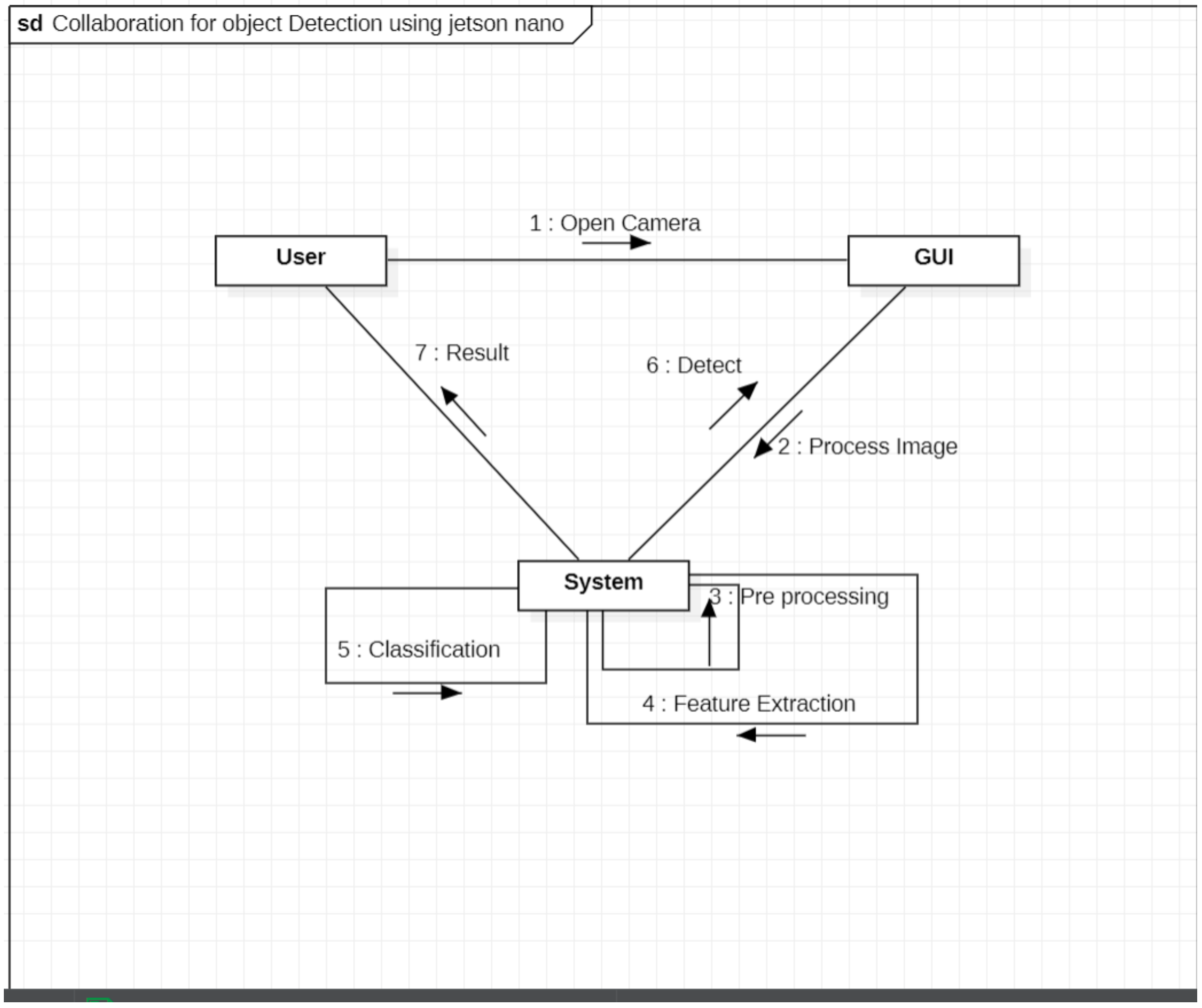


Figure 4.5 Collaboration Diagram for Object Detection

5. IMPLEMENTATION

The implementation of the project has been carried out in a step-by-step manner. A detailed description of each module is given below and it is followed by an introduction to the technologies used in implementing the project.

5.1 MODULES

1.Jetson.inference

2.Jetson.utils

5.2 MODULE DESCRIPTION

5.2.1 Jetson.inference

- Detectnet object is used for loading of the images.
- Tensor networks are useful constructs for efficiently representing and manipulating correlated data.

DETECTNET

The **NVIDIA Deep Learning GPU Training System** (DIGITS) puts the power of **deep learning** in the hands of data scientists and researchers. Using DIGITS you can perform common deep learning tasks such as managing data, defining networks, training several models in parallel, monitoring training performance in real time, and choosing the best model from the results browser. DIGITS is completely interactive so that you can focus on designing and training networks rather than programming and debugging.DIGITS 4 introduces a new object detection workflow that allows you to train networks to detect objects (such as faces, vehicles, or pedestrians) in images and define bounding boxes around them. See the post **Deep Learning for Object Detection with DIGITS** for a walk-through of how to use this new functionality.

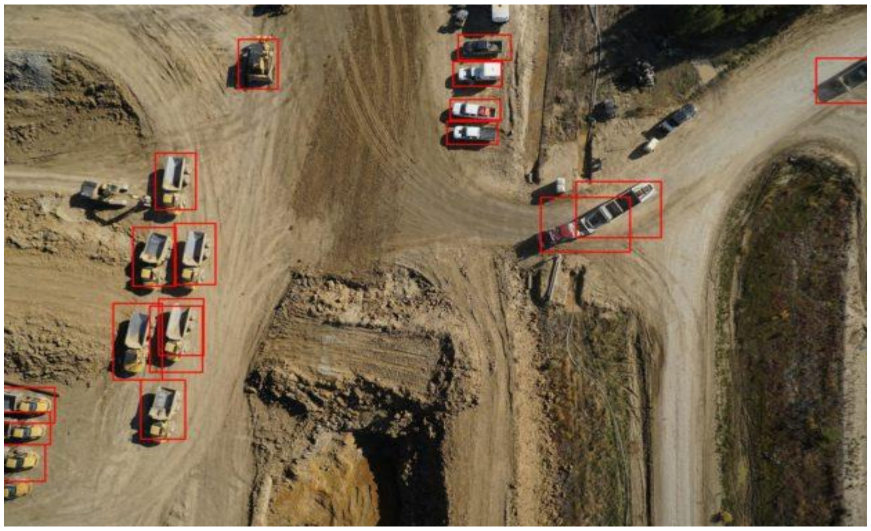


Figure 5.2.1(a)

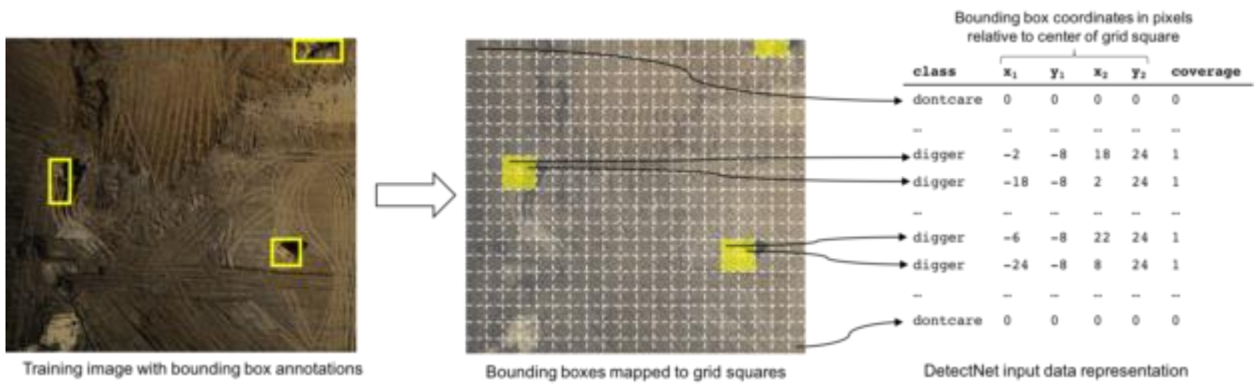
In order to get you up and running as fast as possible with this new workflow, DIGITS now includes a new example **neural network** model architecture called DetectNet. Figure 5.2.1(a) shows an example of the output of DetectNet when trained to detect vehicles in aerial imagery. DetectNet is provided as a standard model definition in DIGITS 4 and is trained using the Caffe deep learning framework. In this post we will explore the structure of DetectNet and show you how it is trained to perform object detection.

DETECTNET DATA FORMAT

Image classification training data samples are simply images (usually a small image or patch containing a single object) labeled by class (typically integer class ID or a string class name). Object detection, on the other hand, requires more information for training. DetectNet training data samples are larger images that contain multiple objects. For each object in the image the training label must capture not only the class of the object but also the coordinates of the corners of its bounding box. Because the number of objects can vary between training images, a naive choice of label format with varying length and dimensionality would make defining a loss function difficult.

DetectNet solves this key problem by introducing a fixed 3-dimensional label format that enables DetectNet to ingest images of any size with a variable number of objects present. The DetectNet data representation is inspired by the representation used by Redmon .

Figure 5.1.1(b) shows the process that DIGITS uses to ingest annotated training images for training DetectNet. DIGITS overlays the image with a regular grid with spacing slightly smaller than the smallest object we wish to detect. Each grid square is labeled with two key pieces of information: the class of object present in the grid square and the pixel coordinates of the corners of the bounding box of that object relative to the center of the grid square. In the case where no object is present in the grid square a special “dontcare” class is used so that the data representation maintains a fixed size. A coverage value of 0 or 1 is also provided to indicate whether an object is present within the grid square. In the case where multiple objects are present in the same grid square DetectNet selects the object that occupies the most pixels within the grid square. In the case of a tie the object with the bounding box with the lowest y-value is used. This choice is arbitrary for aerial imagery but is beneficial for datasets with a ground-plane, such as automotive cameras, where a bounding box with lower y-value indicates an object that is closer to the camera.


Training image with bounding box annotations
Bounding boxes mapped to grid squares
DetectNet input data representation

Bounding box coordinates in pixels relative to center of grid square

class	x_1	y_1	x_2	y_2	coverage
dontcare	0	0	0	0	0
--	--	--	--	--	--
digger	-2	-8	18	24	1
digger	-18	-8	2	24	1
--	--	--	--	--	--
digger	-6	-8	22	24	1
digger	-24	-8	8	24	1
--	--	--	--	--	--
dontcare	0	0	0	0	0

Figure 5.2.1(b)

TENSORNET

TensorRT is a library developed by NVIDIA for faster inference on NVIDIA graphics processing units (GPUs). TensorRT is built on CUDA, NVIDIA's parallel programming model. It can give around 4 to 5 times faster inference on many real-time services and embedded applications. While as per the documentation, It does give 40 times faster inference as compared to CPU only performance.

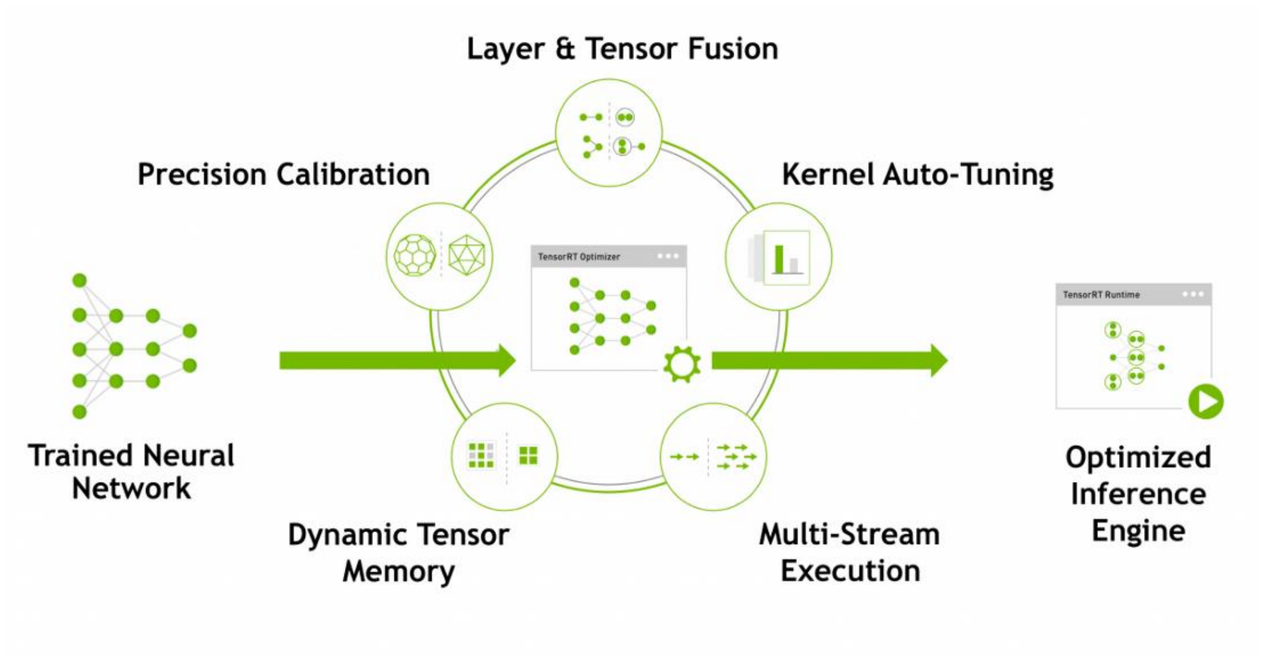


Figure 5.2.2 TensorFlow layers

5.2.2 JETSON.UTILS

Utils is used for capturing the images and detecting or locating the objects in the video or image.

- VideoSource()
- VideoOutput()

CAMERA ON JETSON

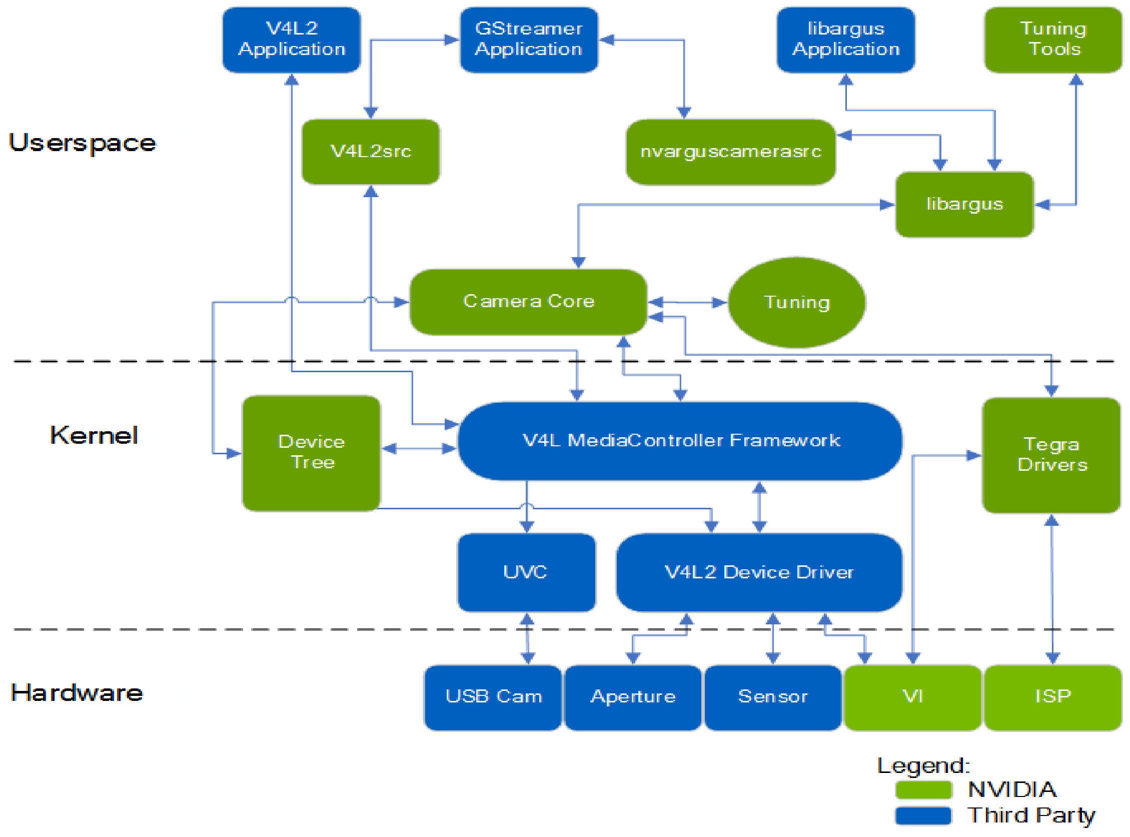
The NVIDIA Jetson Developer Kits support direct connect camera integration using two main methods. The first is to use a MIPI **C**amera **S**erial **I**nterface (**CSI**). The MIPI Alliance is the name of the industry group that develops technical specifications for the mobile ecosystem. On Jetsons like the Nano, this might be a sensor module like the familiar Raspberry Pi V2 camera based on the Sony IMX219 Image Sensor.

The second type of camera is a camera that connects via a USB port, for example a webcam. In this article, we will discuss USB cameras.

JETSON CAMERA ARCHITECTURE

At its core, the Jetsons use the Linux kernel module **V**ideo Four **L**inux (version **2**) (**V4L2**). V4L2 is part of Linux, and not unique to the Jetsons. V4L2 does a wide variety of tasks, here we concentrate on video capture from cameras.

Here is the Jetson Camera Architecture Stack:



5.3 INTRODUCTION TO TECHNOLOGIES USED

5.3.1 PYTORCH

TRANSFER LEARNING WITH PYTORCH

Transfer learning is a technique for re-training a DNN model on a new dataset, which takes less time than training a network from scratch. With transfer learning, the weights of a pre-trained model are fine-tuned to classify a customized dataset. In these examples, we'll be using the ResNet-18 and SSD-Mobilenet networks, although you can experiment with other networks too.

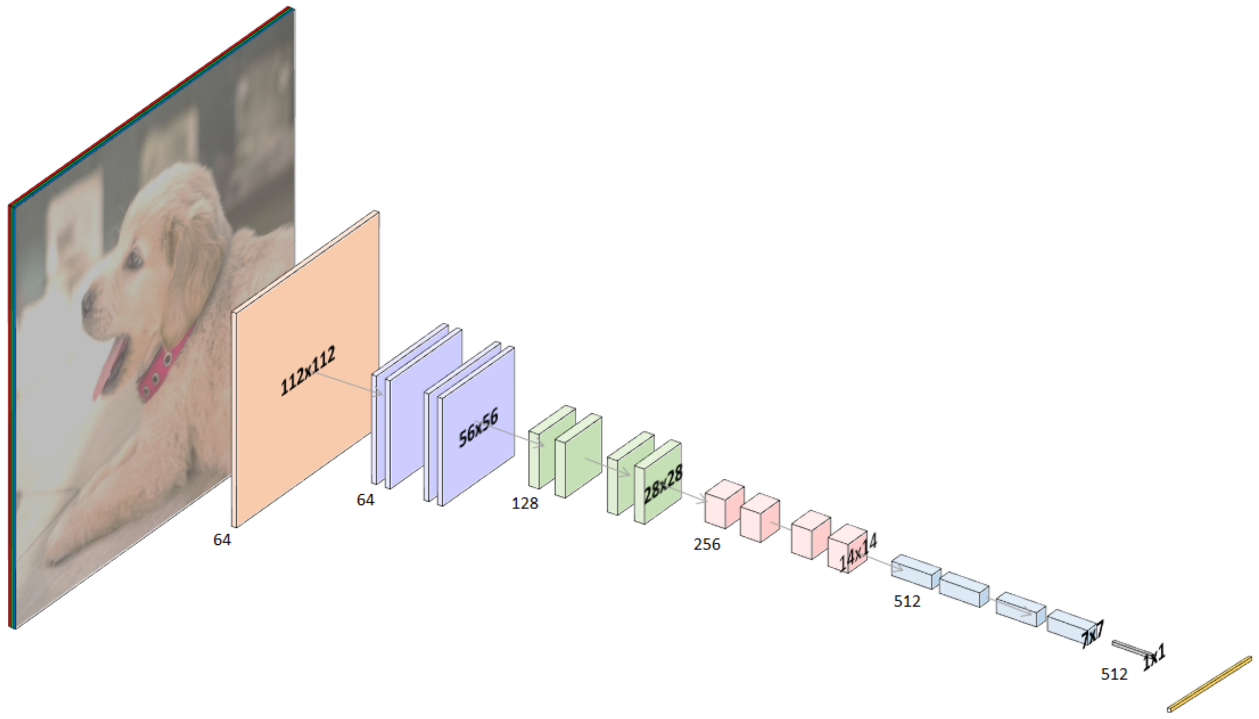


Figure 5.3 Transfer Learning

Although training is typically performed on a PC, server, or cloud instance with discrete GPU(s) due to the often large datasets used and the associated computational demands, by using transfer learning we're able to re-train various networks onboard Jetson to get started with training and deploying our own DNN models.

PyTorch is the machine learning framework that we'll be using, and example datasets along with training scripts are provided to use below, in addition to a camera-based tool for collecting and labeling your own training datasets.

5.3.2 SSD-MOBILENET

RE-TRAINING SSD-MOBILENET

SSD is a popular one-stage detector that can predict multiple classes. The method detects objects in images using a single deep neural network by discretizing the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. The object detector generates scores for the presence of each object category in each default box and adjusts the box to better fit the object shape. Also, the network combines predictions from multiple feature maps with different resolutions to handle objects of different sizes. The SSD detector is easy to train and integrate into software systems that require an object detection component. In comparison to other single-stage methods, SSD has much better accuracy, even with smaller input image sizes.

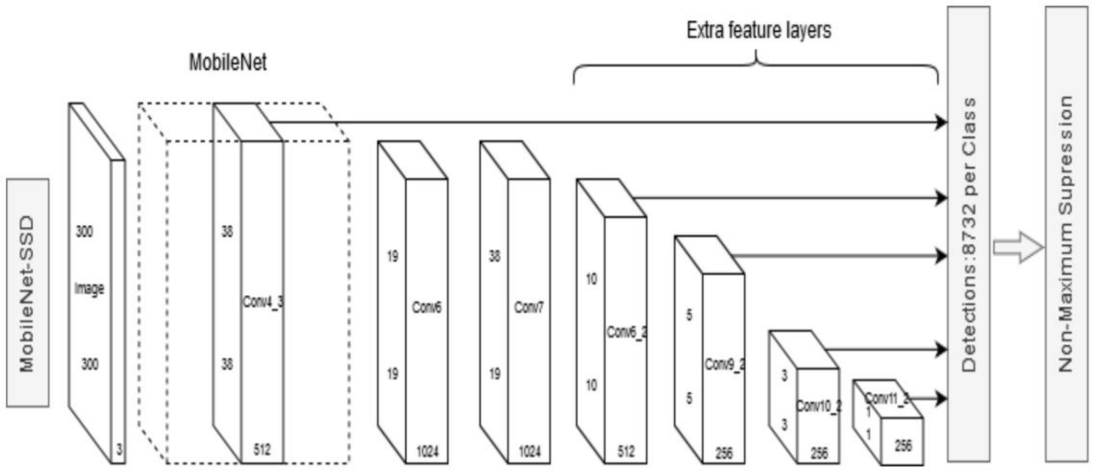


Figure 5.3.1 SSD Layers

5.3.3 IMAGENET

ImageNet is a large database or dataset of over 14 million images. It was designed by academics intended for computer vision research. It was the first of its kind in terms of scale. Images are organized and labelled in a hierarchy.

In Machine Learning and Deep Neural Networks, machines are trained on a vast dataset of various images. Machines are required to learn useful features from these training images. Once learned, they can use these features to classify images and perform many other tasks associated with computer vision. ImageNet gives researchers a common set of images to benchmark their models and algorithms.

It's fair to say that ImageNet has played an important role in the advancement of computer vision.

5.3.4 TENSORFLOW

Tensorflow and Keras are open-source libraries for numerical computation and large-scale machine learning that ease Google Brain TensorFlow, the process of acquiring data, training models, serving predictions, and refining future results.

- Tensorflow bundles together Machine Learning and Deep Learning models and algorithms.
- It uses Python as a convenient front-end and runs it efficiently in optimized C++.
- Tensorflow allows developers to create a graph of computations to perform.
- Each node in the graph represents a mathematical operation and each connection represents data. Hence, instead of dealing with low-details like figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application.

The deep learning artificial intelligence research team at Google, Google Brain, in the year 2015 developed TensorFlow for Google's internal use. This Open-Source Software library is used by the research team to perform several important tasks.

TensorFlow is at present the most popular software library. There are several real-world applications of deep learning that makes TensorFlow popular. Being an Open-Source library for deep learning and machine learning, TensorFlow finds a role to play in text-based applications, image recognition, voice search, and many more. DeepFace, Facebook's image recognition system, uses TensorFlow for image recognition. It is used by Apple's Siri for voice recognition. Every Google app that you use has made good use of TensorFlow to make your experience better.

5.4 SAMPLE CODE

My-detection.py

```
import jetson.inference
import jetson.utils

net=jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5)
camera=jetson.utils.gstCamera(1280,720,"/dev/video0")
display= jetson.utils.glDisplay()

while display.IsOpen():
        img,width,height=camera.CaptureRGBA()
        detections=net.Detect(img,width,height)
        display.RenderOnce(img,width,height)
        display.SetTitle("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
```

Detectnet-console.py

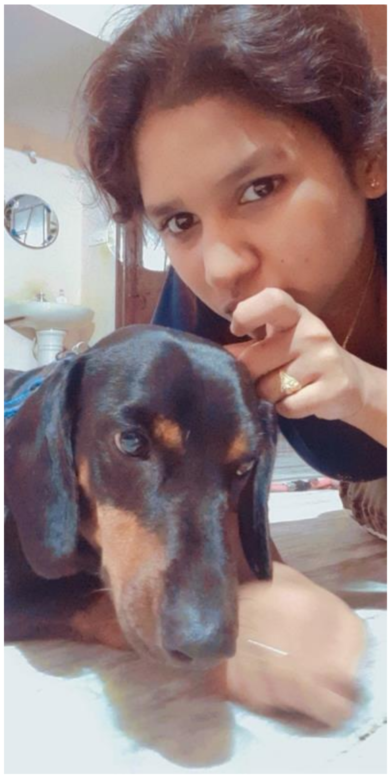
```
import jetson.inference
import jetson.utils
import argparse
import sys

parser = argparse.ArgumentParser(description="Locate objects in a live camera stream using an object detection 
DNN.", 
                              formatter_class=argparse.RawTextHelpFormatter, 
epilog=jetson.inference.detectNet.Usage() +
                              jetson.utils.videoSource.Usage() + jetson.utils.videoOutput.Usage() + 
jetson.utils.logUsage())

parser.add_argument("input_URI", type=str, default="", nargs='?', help="URI of the input stream")
parser.add_argument("output_URI", type=str, default="", nargs='?', help="URI of the output stream")
parser.add_argument("--network", type=str, default="ssd-mobilenet-v2", help="pre-trained model to load (see 
below for options)")
parser.add_argument("--overlay", type=str, default="box,labels,conf", help="detection overlay flags (e.g. --
overlay=box,labels,conf)\nvalid combinations are:  'box', 'labels', 'conf', 'none'")
parser.add_argument("--threshold", type=float, default=0.5, help="minimum detection threshold to use")
is_headless = ["--headless"] if sys.argv[0].find('console.py') != -1 else [""]
try:
        opt = parser.parse_known_args()[0]
except:
        print("")
        parser.print_help()
        sys.exit(0)
output = jetson.utils.videoOutput(opt.output_URI, argv=sys.argv+is_headless)
net = jetson.inference.detectNet(opt.network, sys.argv, opt.threshold)
input = jetson.utils.videoSource(opt.input_URI, argv=sys.argv
while True:
        img = input.Capture()
```

```
detections = net.Detect(img, overlay=opt.overlay)
print("detected {:d} objects in image".format(len(detections)))
for detection in detections:
    print(detection)
output.Render(img)
output.SetStatus("{:s} | Network {:.0f} FPS".format(opt.network, net.GetNetworkFPS()))
net.PrintProfilerTimes()
if not input.IsStreaming() or not output.IsStreaming():
    break
```

6. TEST CASES

TEST CASE 1:

TEST CASE 2:

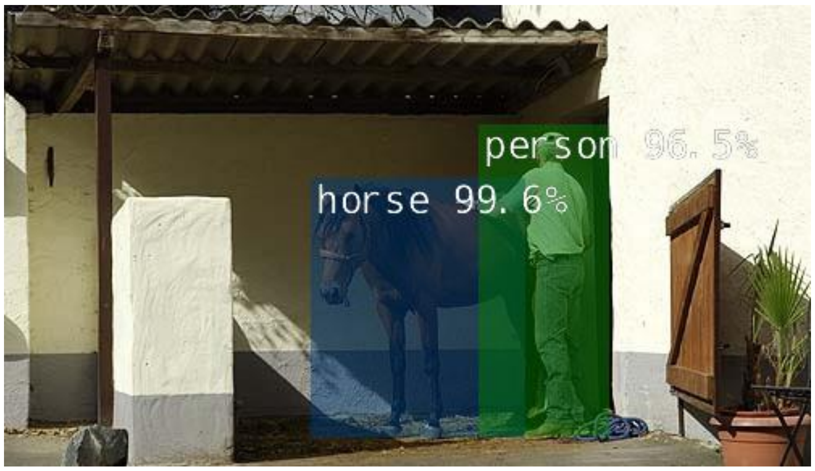


TEST CASE 3:

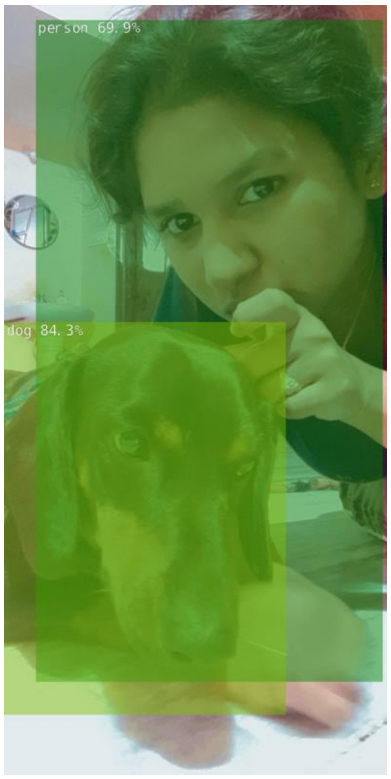
Detecting objects in live stream.

7. SCREEN SHOTS

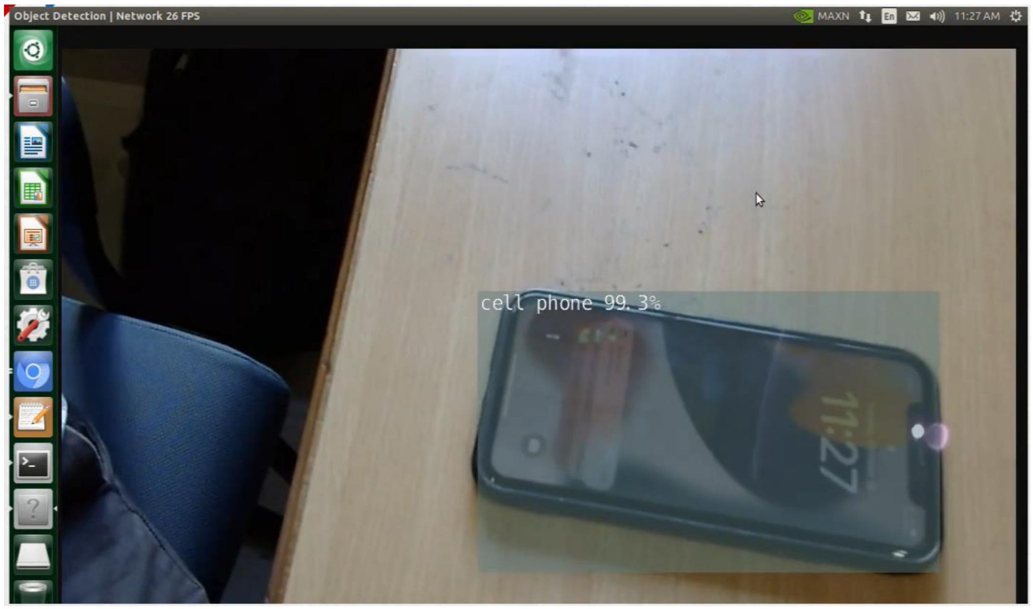
OUTPUT 1:

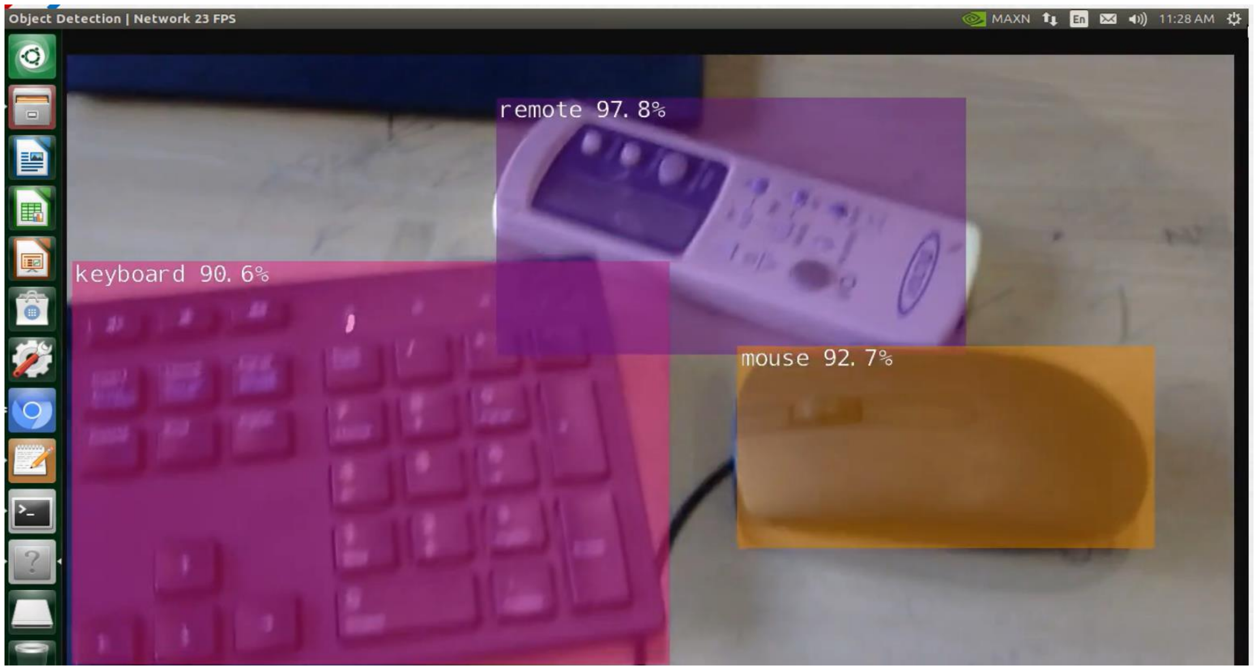


OUTPUT 2:



OUTPUTS OF LIVE CAMERA





8. CONCLUSION

Machine learning and object detection is becoming more and more popular within’ embedded devices. For industrial use, as an example for monitoring devices, security devices, quality assurance devices and many more, but also object detection with embedded devices for personal use is accessible. The ability to create such applications using object detection networks can help an individual or industry solve a variety of different problems and tasks. Purpose of this thesis was to introduce the reader on basic tools and technologies used in such applications and development on embedded device called Nvidia Jetson Nano. Also, to demonstrate deployment of object detection application from ground up. Even though program is not nowhere near perfect, thesis proved the capabilities of Nvidia Jetson Nano to be a sole environment of development for such application. For further improvements of application in question, would be furthering the data set, continue training for even better accuracy, but also add variety of bad habits. Object detection is not limited in any way on which object is decided to detect, thus it is powerful technique to improve day to day life. Nvidia Jetson Nano carried out the application with satisfaction, but it does come with limitations. Such as computing power. Handling even larger object detection applications can be troublesome, in example training large data sets.

Outcoming data of the example implementation application can be used in many different ways. Data can be used recognize the students in the college whether they are carrying ID cards are not and as well as to detect crowdy places in the college or for to go canteen or not, by extending it we can also monitor the students in class for attendance purpose.

9. FUTURE ENHANCEMENT

- For Night time visual tracking, night vision mode should be available as an inbuilt feature in the CCTV.
- To detect identity card for student as per the student database.
- Detection of fire and taking according actions (water sprinklers) to that.
- Detection of bio-degradable and Non bio-degradable product.

10. BIBLIOGRAPHY

[1] Nvidia. (n.d.). About Us. Nvidia.
Available from: https://www.nvidia.com/en-us/about-nvidia/

[2] Nvidia. (n.d.). Jetson Nano. Nvidia Developer.
Available from: https://developer.nvidia.com/embedded/jetson-nano

[3] Nvidia. (n.d.). JetPack SDK 5.0 Developer Preview. Nvidia Developer.
Available from: https://developer.nvidia.com/embedded/jetpack

[4] Franklin D. 2022. Dusty-nv/jetson-inference: Deploying Deep Learning. GitHub.
Available from: https://github.com/dustynv/jetson-inference

[5] Python. (n.d.). General Python FAQ. Python Docs.
Available from: https://docs.python.org/3/faq/general.html

[6] Python Institute. (n.d.). What is Python? Python Institute.
Available from: https://pythoninstitute.org/what-is-python/

[7] Hui J. 2018. SSD object detection: Single Shot MultiBox Detector for realtime processing.
Availablefrom:https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multiboxdetector-for-real-time-processing-9bd8deac0e06

[8] Jog C. 2020. The Two Benefits of the ONNX Library for ML models.
Available From: https://medium.com/truefaceai/two-benefits-of-the-onnx-library-for-ml-models-4b3e417df52e

[9] Brownlee J. 2017. A Gentle Introduction to Transfer Learning for Deep Learning. Machine Learning Mastery.
Available from: https://machinelearningmastery.com/transfer-learning-for-deep-learning/