“Object Detection using Deep Learning Approach”

Thesis Submitted in fulfilment of the Requirement for the degree

Of

Masters of Technology (M. Tech)
in
Department of Computer Science and Engineering

By

PAROMITA SAHA
(**Roll No.:** 502120021005 **Reg. No.:** 201430411210004)

Under the guidance of

Mr. Moloy Dhar
Assistant Professor

Guru Nanak Institute of Technology, Kolkata-700114

ACKNOWLEDGEMENT

Acknowledgement is not just for as near as formality but it is a genuine opportunity to express ineptness to all of those without whose active support and encouragement this project would be possible. Our highly acknowledged and great to many people for successful projects. First of all, we thank Guru Nanak Institute of Technology to work on the real-life platform. Biggest thanks to Mr. Moloy Dhar who is my project supervisor. For this biggest help of developing logic in different modules of the project. I am highly obliged to Mr. Moloy Dhar for the help, supreme co-operation, and spontaneous encouragement for the great success of this project.

Date: …/…/2022 PAROMITA SAHA
Place: Kolkata

GURU NANAK INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the thesis entitled, "**Object Detection using Deep Learning Approach**" which is going to be submitted by *Paromita Saha*, *Roll No.-502120021005*, for the award of the degree of partial fulfilment of Masters of Technology in the branch Computer Science and Engineering of Guru Nanak Institute of Technology, is a record of bonafide research work carried out under the supervision and guidance of the undersigned. Miss. Paromita Saha has worked for nearly one year on the aforementioned subject at the Department of Computer Science and Engineering, Guru Nanak Institute of Technology, Kolkata and this has reached the standard fulfilment of the requirements and the regulation relating to the degree.

The contents of this thesis, in full or part, have not been submitted to any other university or institution for the award of any degree.

Head of the department

Mrs. Sangeeta Bhattacharya
Head of the Department,
Department of Computer Science and
Engineering,
Guru Nanak Institute of Technology,
Kolkata-700114

Supervisor

Mr. Moloy Dhar
Assistant Professor,
Department of Computer Science and
Engineering,
Guru Nanak Institute of Technology,
Kolkata- 700114

CONTENTS

ABSTRACT

The most often utilized strategies for current deep learning models to accomplish a multitude of activities on devices are mobile networks and binary neural networks. In this research, we propose a method for identifying an object using the pre-trained deep learning model MobileNet for Single Shot Multi-Box Detector (SSD). This technique is utilized for real-time detection as well as webcams to detect the object in a video feed.

To construct the module, we use the MobileNet and SSD frameworks to provide a faster and effective deep learning-based object detection approach. Deep learning has evolved into a powerful machine learning technology that incorporates multiple layers of features or representations of data to get cutting-edge results. Deep learning has demonstrated outstanding performance in a variety of fields, including picture classification, segmentation, and object detection. Deep learning approaches have recently made significant progress in fine-grained picture categorization, which tries to differentiate subordinate-level categories.

The major goal of our study is to investigate the accuracy of an object identification method called SSD, as well as the significance of a pre-trained deep learning model called MobileNet. To perform this challenge of detecting an item in an image or video, we used OpenCV libraries, Python, and NumPy. This enhances the accuracy of behavior recognition at a processing speed required for real-time detection and daily monitoring requirements indoors and outdoors.

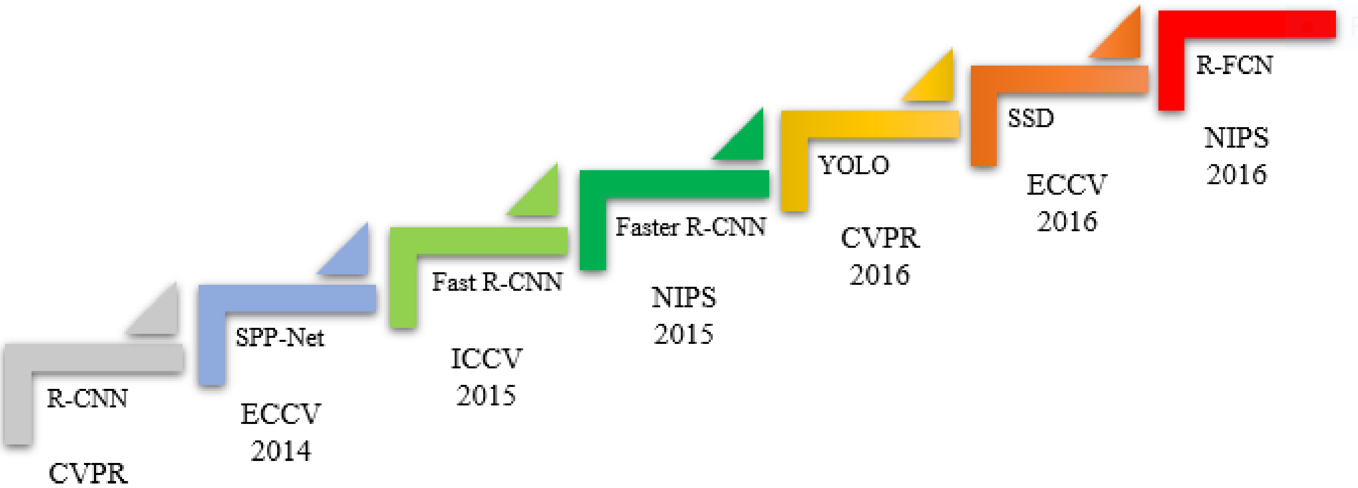
Keywords: *Open Source Computer Vision, Deep Learning, Deep Convolution Neural Networks, Single Shot Detector, MobileNet v3 architecture, Non Maximum Suppression, MS-COCO, Object Detection, Real time capture.*

INTRODUCTION

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. Surveillance cameras, pedestrian displays, self-driving cars, and facial recognition are just a few examples of where object detection is being used. An image, such as a picture, video, or webcam, is included in the Deep Learning sub-discipline of Object Detection [1]. Object recognition from video is a critical task in today's video surveillance applications.

The method of locating and classifying items using rectangular bounding boxes to identify and arrange them into categories is known as object detection. There are certain relationships between object detection and classification, as well as semantic segmentation and instance segmentation. Object detection includes face detection, text detection, pedestrian detection, logo detection, video detection, vehicle detection, and medical picture detection, among other things [2].

Deep Learning field, the sub-discipline which is called Object Detection, includes an image such as a photo, video, or webcam feed. Since R-CNN predicted in 2014, built on DNN, object detection has grown tremendously. Finally, a set of upgraded approaches based on R-CNN as SPP-NET, Faster RCNN, and R-FCN also appear.



Object Detectors using Deep Learning

Conventional approaches can be utilised to tackle this issue, however Convolutional Neural Networks (CNN) appear to be a feasible system choice. Because the SSD-MobileNet [3] paradigm cannot process live video on an embedded device, we looked into strategies that would allow us to reduce video processing times while maintaining accuracy in a system. One of my work's specific contributions is as follows: The accuracy, performance, and processing times of systems for a counting application are evaluated. SSD-MobileNet is the object detecting method. Natural images are included in the trained data set, also known as MS-COCO data set (91 classes).

This object detection model is a robust DNN model that can attain great precision. However, they required a significant number of computational and configuration variables, which are not ideal for embedded applications. The MobileNet was created in response to this difficulty, and it minimises the number of parameters and complexity of an algorithm to a number of resources required by embedded devices.

The use of a separate deep configuration is the MobileNet's most important impact. The width & resolution multiplier hyperparameters are two hyperparameters. Previously, certain strategies for object detection were used, such as employing the classic approach of computer vision to extract texture, shape, and colour.

As a result, a single network and faster performance are required. The Single Shot Multi-Box Detector is developed on MobileNet and includes extra layers like feature extraction layers which are specifically designed for real-time object detection, eliminating the need for a region proposal network and speeding up the process. To compensate for the loss of precision, SSD makes a number of changes to the multi-scale features and default boxes. The SSD object detection is divided into two parts: Extract feature maps first, then apply convolution filters to detect objects [4].

Mobilenet-v3 networks now offer a good compromise between processing speed and object identification accuracy. As a result, we picked the Mobilenet-v3 network, that is supported by a wide range of embedded systems, as the backbone network for building the suggested compact object detection model, keeping in mind the device's limitations [5].

MobileNet was developed to reduce the number of parameters and complexity of an algorithm to a number of resources demanded running by the embedded devices. With the approach of computing infrastructure, Deep Neural Networks have achieved massive success in the domain of Computer Vision to handle such massive amounts of data. The use of a separate deep configuration is the MobileNet's most important impact. The width and resolution multiplier hyperparameters are two hyperparameters. Previously, certain strategies for object detection were used, such as applying the standard computer vision methodology to encode texture.

One of the easiest methods in this approach is the SSD we're employing with the DL (deep learning) Pre-Trained Model MobileNet through CNN creation to overcome these drawbacks. As a result, the object can be easily achieved in complex video scenarios with complicated backgrounds. This technique is highly well suited to video conferencing and surveillance applications in which the background remains constant during the duration of the meeting or monitoring.

There are quite a wide range of CNN models available, ranging from the classic LeNet model to AlexNet, ZFNet, GoogleNet, VGGNet, ResNet, ResNeXt, SENet, DenseNet, Xception, PNAS/ENAS, MobileNet V1, MobileNet V2, MobileNet V3, through which we can achieve a massive success in object detection.

LITERATURE SURVEY

They employed RGB-depth videos, single and multiple viewpoints, and only single networks in this paper [6]. All of the datasets they utilized perform single activities, with one individual executing one task at a time. On the three types of datasets, this work gives a review of several state-of-the-art deep learning-based approaches proposed for human action recognition. However, these methods have a number of disadvantages, including the necessity to create big datasets, the fact that performance is dependent on the magnitude of the network weights, and the fact that hyper-parameter adjustment is difficult. Multiple networks from different streams are also necessary to recognise multiple human actions at the same time.

The methods employed in this project are Convolutional Neural Networks, which are used for identifying head posture, mouth motions, and face detection in this paper [7]. This research proposes a collection of techniques for online evaluation abnormal behavior monitoring based on image data. The monitoring of anomalous behavior such as turning heads and talking during the online test is done through the application of head pose estimation based on convolutional neural networks and threshold-based mouth state assessment, as well as the combination of specific decision rules. However, it detects all noticeable face motions, resulting in false reports and a low identification rate.

The researcher applied convolutional neural networks and deep neural networks, as well as a variety of dataset models, in this publication [8]. Object detection systems such as GoogleNet, AlexNet, ResNet, ResNeXt, SENet, DenseNet, and others are utilized. Animal detection, handed arms detection, human detection, and several other image object detections are also included. There are a few drawbacks to this project. Researchers can reduce the false positive rate by employing more models and evaluating the system, or by pre-processing the pictures and using videos.

Three separate pre-trained datasets were employed by the author. MobileNetv2, GoogleNet, and MobileNetv3 are the three. According to the results of this method present in this study [9], MobileNetV3 [4] is appropriate for handgun detection since it provides a perfect match between prediction speed and precision. The proposed model had a training accuracy rate of 96%. Real-time detection in live videos and photos was used in the tests. The system has an SMS capability that allows it to send an alarm message to the supervisor once a firearm is spotted. The proposed method can be utilised for a variety of purposes, including real-time identification of guns in supermarkets.

This review study [2] provides a thorough and in-depth examination of deep learning-based object detection algorithms. Backbone networks, detection designs, and loss functions are three parts of it. It also provides a thorough examination of the difficult issues. To offer a complete examination of complicated situations, the authors used Deep CNNs, Recurrent CNNs, and Support Vector Machines. More accurate detection frameworks can increase the real-time and accuracy of embedded detection applications, allowing object detection to be used in numerous applications. There is still a scarcity of research on object detection in 3D images and depth images (RGB-D), which necessitates additional attention. The current object detection algorithm is mostly intended for photos of small and medium size. In addition, the accuracy of detecting different scales of objects in high definition photographs is incredibly low.

The object detection algorithm in this paper [1] can recognise objects at up to 14 frames per second, therefore even low-quality cameras with any frame rate can yield good results. They just use a webcam with a frame rate of 6 frames per second. The SSD method demonstrated interior and outdoor input video frames through camera in our testing, but the placement of the objects differed between two consecutive frames. The video acquired by the webcam, as well as the algorithm, convert the size of a single frame to 300 x 300 pixels. The SSD can generate numerous anchor boxes for multiple categories with varying confidence levels by employing a greater proportion of default boxes, which can have a stronger effect, and distinct boxes for each position. The author is merely utilizing a webcam to detect items. Furthermore, the webcam is limited to 14 frames per second.

Author of this paper [5]employed the Pascal VOC [11] dataset in the experiments for network model training and testing in order to evaluate the detection performance of the proposed Mobilenet-SSDv2 detector. They also evaluated the new detector's object detection accuracy to that of the existing MobileNet-SSD detector. The network model's input picture format is a 512x512 RGB color image. They used a 200-epoch SGD method optimizer to train the proposed network. Based on the MobileNet-v2 backbone network, they suggested a lightweight network design with better feature extraction. However, the optimizer they utilized consumes a lot of memory, and the network's computation rate is extremely poor.

The article [3] describes a series of algorithms that use a Convolutional Neural Network to handle characteristics such as video resolution, bit rate, and extra hardware (VPU) for video processing. A comparison is made between an SSD-MobileNet model with self-trained and pre-trained training. The goal is to produce high performance metrics and fast execution time, which will allow a vehicle counting system to be implemented in a low-capacity embedded device. They could include enhancements to the automobile counting, notably in the tracking block, allowing for accurate vehicle tracking even as the number of skip frames grows.

They attempted to recognise an object that was shown in front of a webcam in this study [12]. The generated model was evaluated and trained using Google's TensorFlow Object Detection API framework. They concentrated on threading methodology to enhance fps, which resulted in a significant reduction in processing time. The detecting rate of the item decreases as the distance between the webcam and the object increases due to the 1.3mp web camera's inadequate pixel capacity. Furthermore, detecting a single object takes approximately three seconds, which is a significant constraint.

PROBLEM STATEMENT

Our research "***Object Detection Using Deep Learning Approach***" efficiently recognizes objects using the Single Shot Multi-Box Detector (SSD) methodology, which is applied to image and video data to identify objects. Our goal is to track and identify a common object in real time. In real life, we need a lot of information about our surroundings. We must understand how the objects move in respect to the camera. For example, as in case of a self-driving automobile, understanding how pedestrians interact will assist in accurately predicting pedestrian behaviour. This prediction will eventually aid the self-driving automobile in making wise decisions on a congested highway.

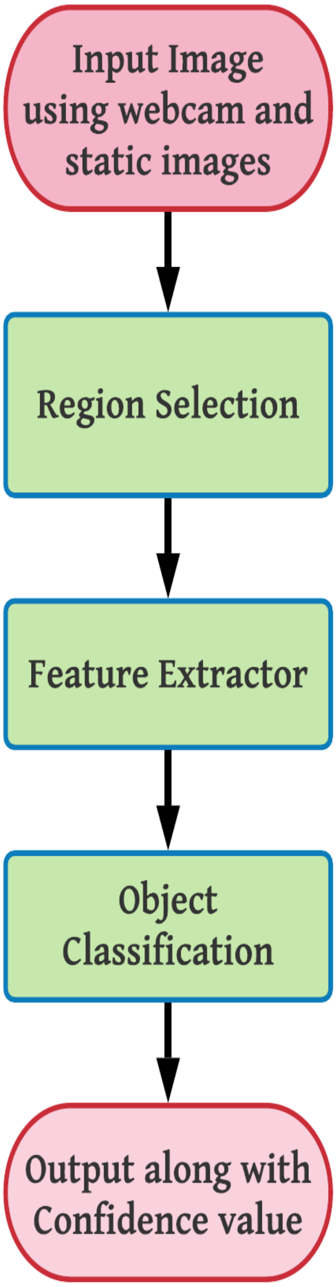

Input Image using webcam and static images
Region Selection
Feature Extractor
Object Classification
Output along with Confidence value

Fig 1. Problem statement

METHODOLOGY

A. Introduction to Object Detection:

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.

Object Detection is done through many ways:

- Feature Based Object Detection
- Viola Jones Object Detection
- SVM Classification with HOG Feature
- 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. It is usually utilized in applications like image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

B. Digital Image Processing:

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output.

A picture can be represented as a two-dimensional capacity f(x, y), where x and y are spatial directions, and the adequacy off in any combination of directions (x, y) is known as the picture's power or dark level. We call an image a computerised picture when x, y, and the abundance estimation of are all limited discrete amounts. The term DIP refers to the process of preparing enhanced images for a digital PC. A complex image is based on a limited number of components, each of which has its own area and value. Pixels are the components that make up a picture.

What Is an Image?

A signal in two dimensions is what an image is. It is defined by the mathematical function f(x,y), where x and y are the horizontal and vertical co-ordinates, respectively. The pixel value at any location in an image is determined by the value of f(x,y).

Figure 2. Digital Image

Pixel: **I**n digital imaging, a **pixel**, or **picture element** is the smallest addressable element in a raster image, or the smallest addressable element in an all points addressable display device; so it is the smallest controllable element of a picture represented on the screen. Each pixel represents a single value. Each pixel stores a value proportionate to the intensity of light at that specific spot. Pixels per inch or dots per inch are the units of measurement. Each pixel can be thought of as a single square point of coloured light.

Resolution: **I**mage resolution describes the image’s level of detail – higher resolution means more image detail. The resolution can be defined as pixel, spatial, temporal, and spectral resolution. In digital imaging, the resolution is often measured as a pixel count.

For example, the resolution of a picture can be defined as MxN if it has M rows and N columns. The higher the pixel resolution, the greater the image quality. An image's resolution can be divided into two categories: **(a) Low Resolution** and **(b) High Resolution.**

Color Image: **C**olour images are three band monochrome images in which, each band contains a different color and the actual information is stored in the digital image. The images are represented as red, green and blue (RGB images). And each color image has 24 bits/pixel means 8 bits for each of the three color band(RGB). 8-bit color is used for storing image information in a computer's memory or in a file of an image. In this format, each pixel represents one 8-bit byte. It has 0-255 range of colors, in which 0 is used for black, 255 for white.

Bit	7	6	5	4	3	2	1	0
Data	R	R	R	G	G	G	B	B

۞ RELATED TECHNOLOGY ۞

Deep Learning:

The field of artificial intelligence is essentially when machines can do tasks that typically require human intelligence. It encompasses machine learning, where machines can learn by experience and acquire skills without human involvement.

Deep learning [31] is a subset of machine learning in artificial intelligence, i.e., based upon artificial neural network and representation learning, as it is capable of implementing a function that is used to mimic the functionality of the brain by creating patterns and processing data. Deep Learning is also used for decision-making in fields like driverless cars (to detect pedestrians, street lights, other cars, etc.), speech recognition, image analysis(e.g., Identifying cancer in blood and tumors), smart TV, etc.

Deep learning [37] varies from typical machine learning techniques in that it can learn representations from data such as photos, video, or text without requiring hand-coded rules or domain knowledge from humans. Their highly adaptable systems may learn directly from raw data and improve forecast accuracy as more data is provided.

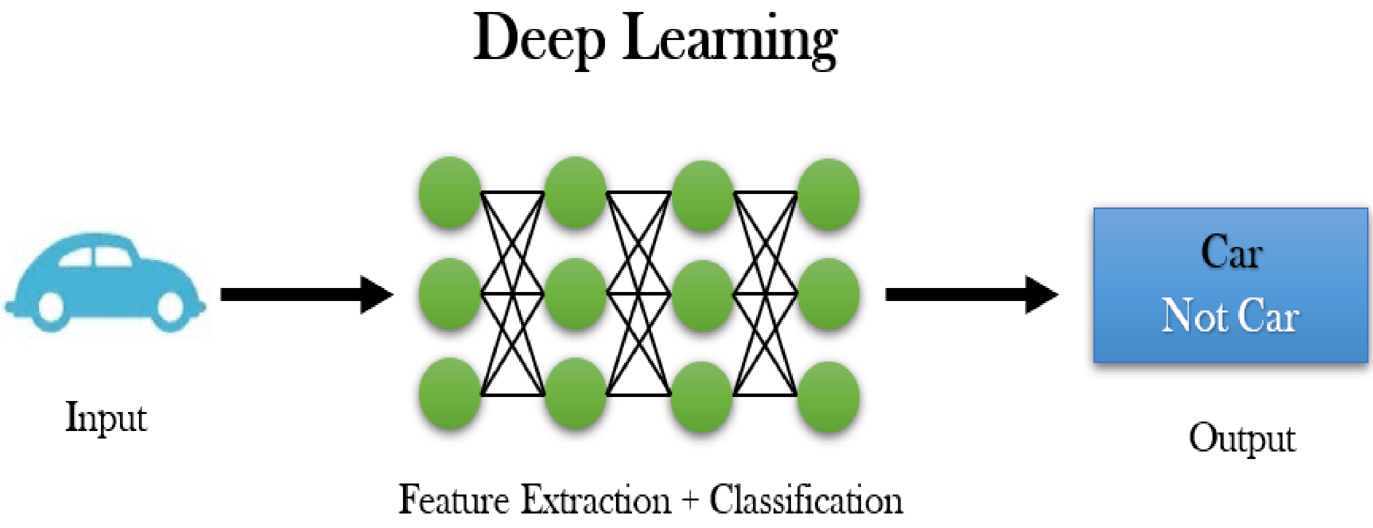


Fig 3. Deep Learning Diagram

In computer vision, conversational AI, and recommendation systems, deep learning is widely used. Deep learning is used in computer vision programmes to extract information from digital photos and videos. Conversational AI apps assist machines in comprehending and communicating in natural language.

Deep learning has paved the way to many recent AI advances, like Google DeepMind's AlphaGo, self-driving cars, intelligent voice assistants, and many others. Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected. The more deep learning algorithms learn, the better they perform.

Practical examples of Deep Learning:

- Virtual Assistants,
- Translations,
- Vision for driverless delivery trucks, drones and autonomous cars,
- Chatbots and service bots,
- Image Colorization,
- Facial Recognition,
- Medicine and pharmaceuticals,
- Personalized shopping and entertainment.

Architectural Methods for Deep Learning Algorithms—To build deep learning architectures, the following algorithms are used:

1. **Back Propagation:** In this algorithm, partial derivatives are calculated. Backpropagation is a mathematical weight function fine-tuning procedure used to increase the accuracy of an artificial neural network's outputs. To enhance outputs, it employs a technique called as chain rule. The technique conducts a backward pass through a network after each forward pass to modify the model's weights. The result is adjusted weights for neurons. It is usually characterized under supervised learning algorithm.

2. **Stochastic Gradient Descent:** In Gradient descent, the goal is to find global minima or optimum solutions. But to get that, we have to consider local minima solutions (not desirable) also. If the objective function is a convex function, it is easy to find the global minima. It's a quick and easy way to fit linear classifiers and regressors to convex loss functions like (linear) Support Vector Machines and Logistic Regression.

3. **Learning Rate:** The learning rate can reduce training time and increase performance. Too large changes are tackled by a higher learning rate and by slowly decreasing the learning rate later for fine-tuning. The learning rate is a hyperparameter that controls how much the model changes each time the model weights are changed in response to the predicted error.

4. **Batch Normalization:** Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This stabilises the learning process and significantly reduces the number of training epochs needed to create deep networks.

5. **Drop Out:** The dropout method drops random units during training by creating different 'thinned networks'. When testing these thinned networks' predictions are averaged, which helps to avoid overfitting.

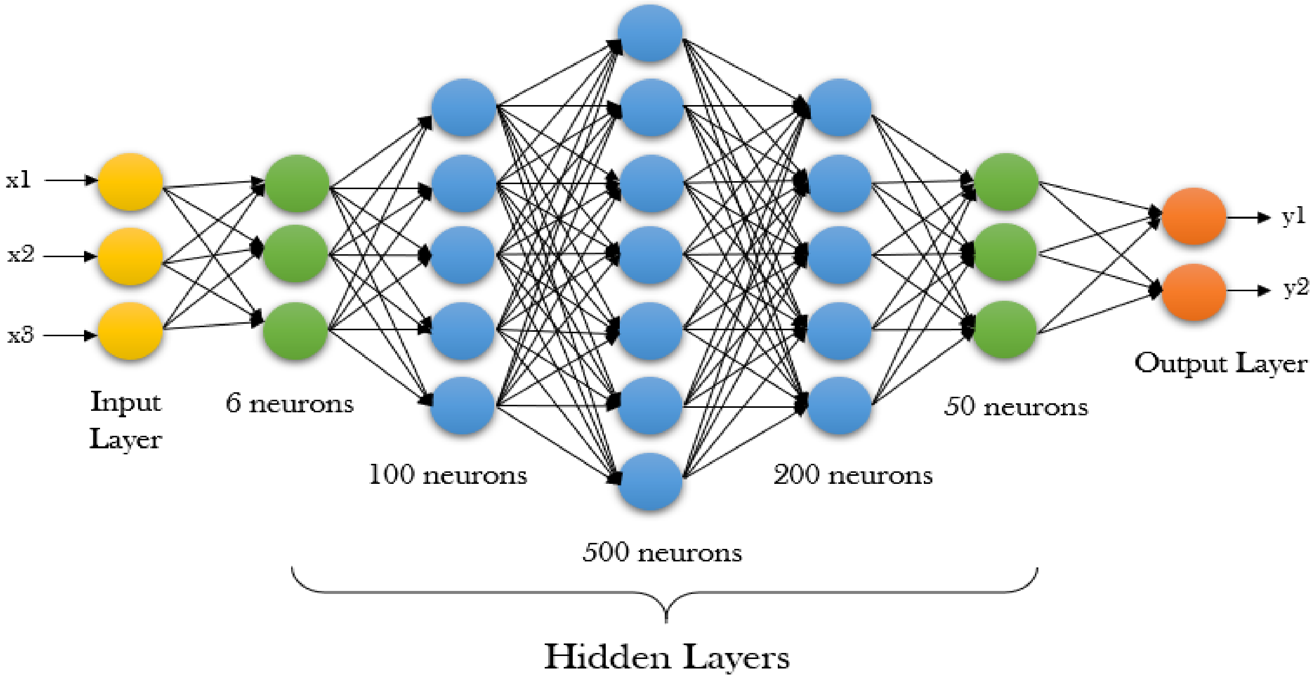
6. **Bag of Words:** A continuous bag of words is used to predict the next word. This is done by considering lots of sentences and for a specific word surrounding words that are captured. These specific words and surrounding words are fed to the neural network.

7. **LSTM:** Long Short-Term Memory networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. LSTM make small modifications to the information by multiplications and additions. LSTM has the edge over other techniques because it is able to consider previous data.

Deep Neural Network (DNN):

Deep neural networks have recently become the standard tool for solving a variety of computer vision problems. Nodes are little components of the system that function similarly to neurons in the human brain. When they are stimulated, a reaction occurs in these nodes. Some are connected and marked, while others are not, although nodes are generally grouped into layers. To complete a task, the system must process layers of data between the input and output. The deeper the network is evaluated, the more layers it must process to obtain the outcome.

An artificial neural network (ANN) having numerous layers between the input and output layers is known as a deep neural network [29] (DNN). Neural networks come in a variety of shapes and sizes, but they all include the same basic components: neurons, synapses, weights, biases, and functions. These components functioning similar to the human brains and can be trained like any other ML algorithm.



DNNs are capable of modelling non-linear relationships that are complex. Compositional models are generated by DNN architectures, in which the object is expressed as a layered composition of primitives. The extra layers allow for the compilation of characteristics from lower levels, potentially allowing for the modelling of complicated data with fewer units than a shallow network of equivalent performance.

DNNs are feedforward networks that transfer data from the input layer to the output layer without looping back. The DNN starts by creating a map of virtual neurons and assigning random integer values, or "weights," to their connections. The inputs and weights are multiplied, and the result is a value between 0 and 1. An algorithm would alter the weights if the network didn't recognise a pattern correctly.

Deep neural networks offer a lot of value to statisticians, particularly in increasing accuracy of a machine learning model. A neural network with some level of complexity, usually at least two layers, qualifies as a deep neural network (DNN), or deep net for short. Deep nets process data in complex ways by employing sophisticated math modelling.

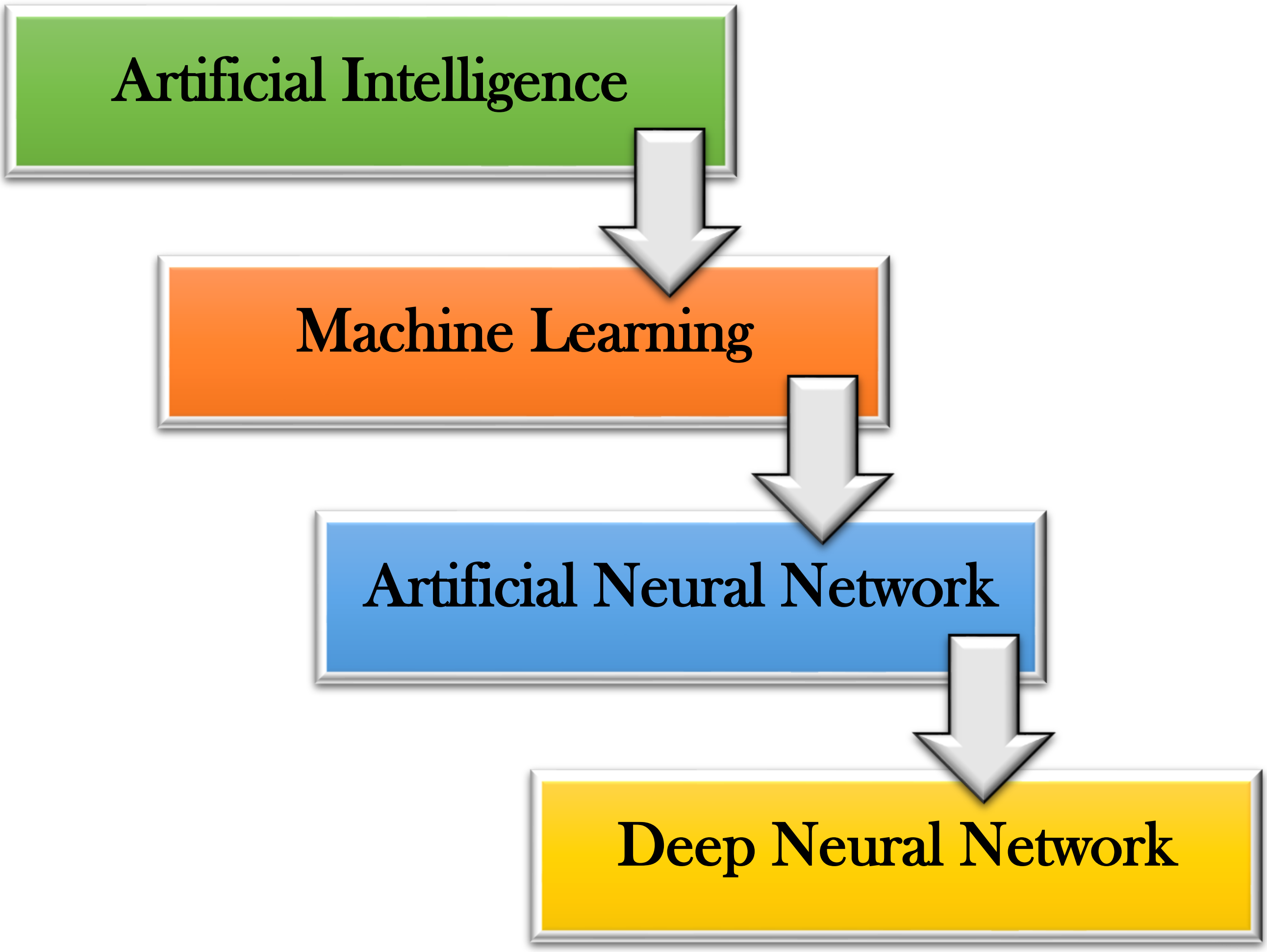


Fig 4. Evolution of Deep Neural Network

MobileNet:

Mobile models have been built on increasingly more efficient building blocks. TensorFlow's first mobile computer vision model, MobileNet, is designed for use in mobile applications. Depthwise separable convolutions are used by MobileNet. When compared to a network with regular convolutions of same depth in the nets, it substantially reduces the number of parameters. As a result, lightweight deep neural networks are created.

MobileNet is a CNN class that was open-sourced by Google, and it provides us with an ideal starting point for training our ultra-small and ultra-fast classifiers. The theory behind Depthwise Separable Convolution is that the depth and spatial dimensions of a filter can be separated, hence the name. The depthwise convolution is followed by the pointwise convolution in this convolution.

MobileNets introduces two new global hyperparameters (width multiplier and resolution multiplier) which let model developers trade off latency or accuracy for speed and compact size, depending on their needs. Convolution layers are depthwise separable in MobileNets. A depthwise convolution and a pointwise convolution make up each depthwise separable convolution layer. A MobileNet contains 28 layers when depthwise and pointwise convolutions are counted separately.

The main objective of their model was to increase network efficiency by reducing the number of parameters while not compromising on performance. Two operations are used to create a depthwise separable convolution.

- **Depthwise convolution**: The channel-wise DKxDK spatial convolution is known as depthwise convolution.
- **Pointwise convolution**: Pointwise convolution is the 1×1 convolution to change the dimension. This produces a feature map.

MobileNet V1:

MobileNet V1 was released in 2017 [32], it essentially launched a new branch of deep learning research in computer vision: developing models that can run on embedded platforms. As an effective substitute for standard convolution layers, MobileNetV1 introduces depth-wise separable convolutions. By separating spatial filtering from the feature generating method, depthwise separable convolutions efficiently factorise classical convolution. Two layers define depthwise separable convolutions: a light-weight depthwise convolution for spatial filtering and a heavier 1x1 pointwise convolution for feature creation. MobileNet V1 is a variant of MobileNet model which is specially designed for edge devices.

In the spring of 2017, the first version of MobileNets was published. The main goal was to present a set of TensorFlow-based computer vision models that maximise accuracy while taking into consideration the limited resources available for on-device or embedded applications. MobileNetV1 are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embeddings and segmentation. The core architecture of MobileNetV1 is based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks.

In order to construct mobile-first computer vision models, MobileNetV1 is attempting to achieve two key goals:

(i) **Smaller Model Size**: Fewer number of parameters

(ii) **Smaller Complexity:** Fewer Multiplications and Additions

MobileNet V2:

By utilising the low rank character of the problem, MobileNetV2 [33] developed the linear bottleneck and inverted residual structure in order to construct even more efficient layer structures. Figure 5 depicts the structure, which is defined by a 1x1 expansion convolution, depthwise convolutions, and a 1x1 projection layer. If and only if the input and output have the same number of channels, they are connected using a residual connection. This structure keeps a compact representation at the input and output while expanding to a higher-dimensional feature space internally to make nonlinear per-channel transformations more expressive.

MobileNet-v2 is a convolutional neural network that is 53 layers deep. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. MobileNetV2 is quite identical to the original MobileNet, except that it utilizes inverted residual blocks with bottlenecking features. It has a significantly fewer number of parameters than the original MobileNet. Any image size bigger than 32 × 32 is supported by MobileNets, with larger image sizes providing better performance.

A few significant changes were made to the MobileNetV1 architecture that resulted in a considerable increase in the accuracy of the model. The main changes made to the architecture were the introduction of inverted residual blocks and linear bottlenecks and the use of the ReLU6 activation function in place of ReLU. MobileNetV2 introduced two new features to the architecture: (i) linear bottlenecks between the layers and (ii) shortcut connections between the bottlenecks1.

MobileNetV2's central idea is that the bottlenecks contain the model's intermediate inputs and outputs, while the inner layer encodes the model's ability to switch from lower-level concepts like pixels to higher-level descriptors like image categories. Finally, shortcuts, like conventional residual connections, allow for speedier training and improved accuracy.

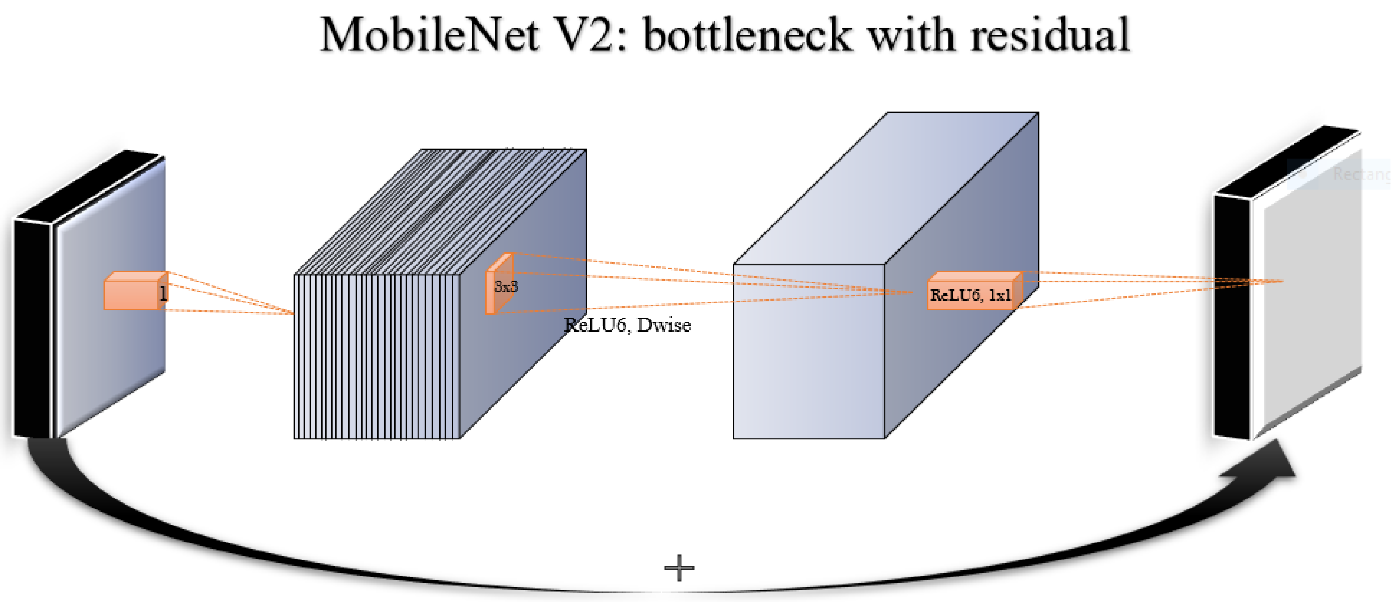


Fig 5. MobileNet V2 Block Diagram

MobileNet V3:

Unlike the previous version of MobileNet, which was hand-crafted, MobileNetV3 [34] uses AutoML to identify the best possible architecture in a search space that is conducive to mobile computer vision tasks. In order to develop the most effective models for MobileNetV3 [4], we employ a combination of these layers as building blocks. Modified swish nonlinearities are also added to the layers. The sigmoid is used in both squeezing and excitation, as well as the swish nonlinearity, but it is inefficient to compute and difficult to maintain precision in fixed point arithmetic, so we use the hard sigmoid instead.

The MobileNetV3 [35] search space is based on a number of current architectural design innovations that we have adapted for the mobile environment. First, we introduce the hard-swish (h-swish) activation function, which is based on the Swish nonlinearity function. The Swish function has a significant flaw: it is extremely wasteful to calculate on mobile devices. As a result, we adopt an approximation that can be written efficiently as the product of two piecewise linear functions.

MobileNetV3 leverages two AutoML techniques: MnasNet and NetAdapt. MnasNet, which employs reinforcement learning to determine the best configuration from a discrete set of options, is used by MobileNetV3 to find a coarse architecture initially. After that, the model uses NetAdapt, a complimentary technique that trims under-utilized activation channels in tiny increments, to fine-tune the design.

The addition of a squeeze-and-excitation block to MobileNetV3's core architecture is another innovative idea. The squeeze-and-excitation blocks' main objective is to increase the quality of a network's representations by explicitly modelling the interdependencies between its convolutional features' channels.

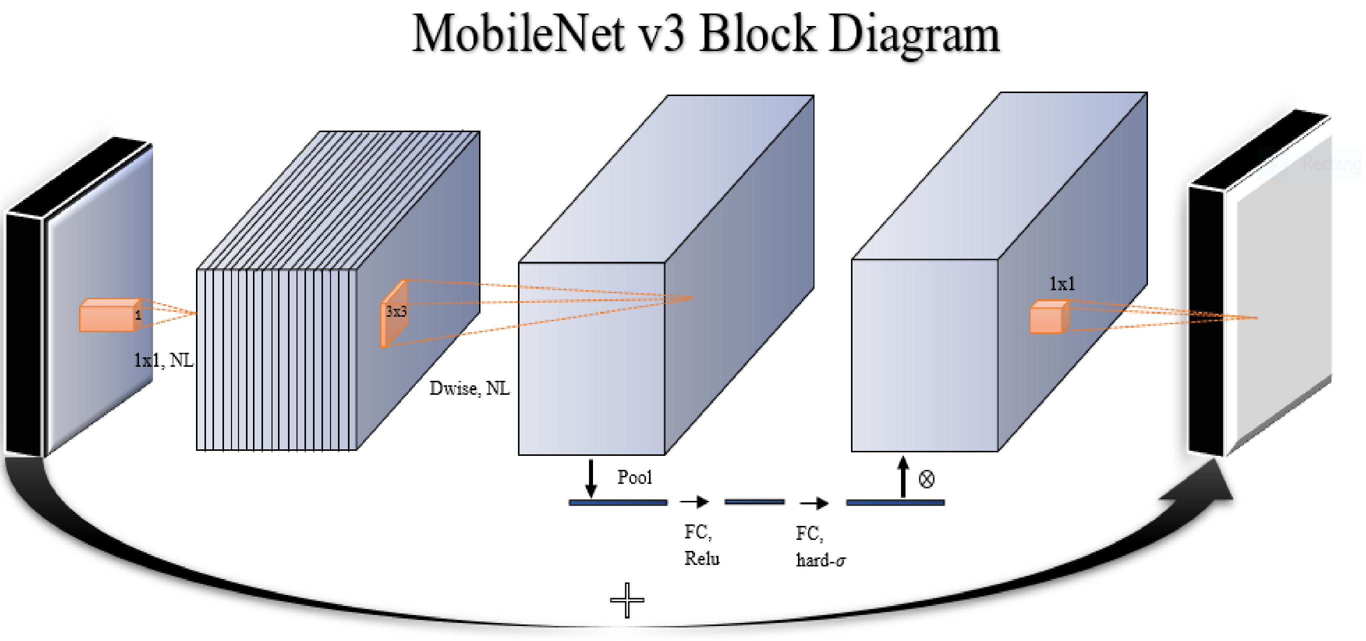


Fig 6. MobileNet V3 Block Diagram

Some other types of backbone network similar to MobileNet:

1. ShuffleNet: Megvii Inc (a.k.a Face++) introduced ShuffleNet, which they claim to be an extremely computation efficient CNN architecture, designed for mobile devices with computing power of 10–150 MFLOPs. The ShuffleNet utilizes pointwise group convolution and channel shuffle to reduce computation cost while maintaining accuracy. It manages to obtain lower top-1 error than the MobileNet system on ImageNet classification, and achieves ~13x actual speedup over AlexNet while maintaining comparable accuracy.

2. CondenseNet: CondenseNet is a novel, computationally efficient convolutional network architecture. It combines dense connectivity between layers with a mechanism to remove unused connections. The dense connectivity facilitates feature re-use in the network, whereas learned group convolutions remove connections between layers for which this feature re-use is superfluous. At test time, the model can be implemented using standard grouped convolutions —– allowing for efficient computation in practice.

3. EffNet: EffNet uses spatial separable convolutions. It is very similar to MobileNet's depthwise separable convolutions. It is made of depthwise convolution with a line kernel (1x3), followed by a separable pooling, and finished by a depthwise convolution with a column kernel (3x1). EffNet uses spatial separable convolution, which is simply a depthwise convolution splitted along the x and y axis with a separable pooling between them.

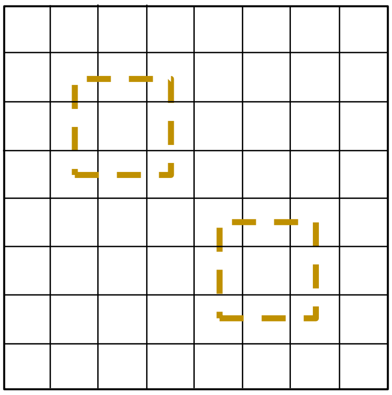
4. **ResNet:** ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

Single Shot Detector (SSD):

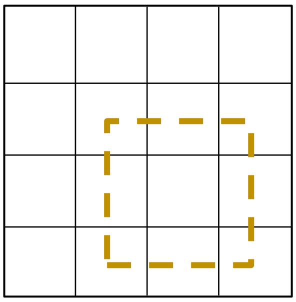
SSD [26] is a well-known one-stage detector, which only need to take one single shot to detect multiple objects within the image. Using a single deep neural network, the approach finds objects in images by dividing the output space of bounding boxes into a series of default boxes with varied aspect ratios 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.

This SSD detector is easy to train and integrate into software systems that need to detect objects. SSD offers much better accuracy than other single-stage algorithms, especially with smaller input image sizes. SSD Object Detection generates a subset of features from a CNN-based deep learning model and then implements convolution filters to recognise things. With object detection in SSD [25], detection rate accuracy is accomplished by utilising numerous boxes or filters of various sizes and aspect ratios. These filters are also applied to various extracted features from a network's subsequent phases. This facilitates detection at various scales. When compared to two-shot RPN-based techniques, SSD is substantially faster.

1. **MultiBox Detector:** SSD detects objects from a single layer. Actually, it uses multiple layers (multi-scale feature maps) to detect objects independently. SSD uses lower resolution layers to detect larger scale objects. For example, the 4× 4 feature maps are used for larger scale object. SSD adds 6 more auxiliary convolution layers after the MobileNetv3. Five of them will be added for object detection. In three of those layers, SSD make 6 predictions instead of 4. In total, SSD makes 8732 predictions using 6 layers. Multi-scale feature maps improve accuracy significantly.

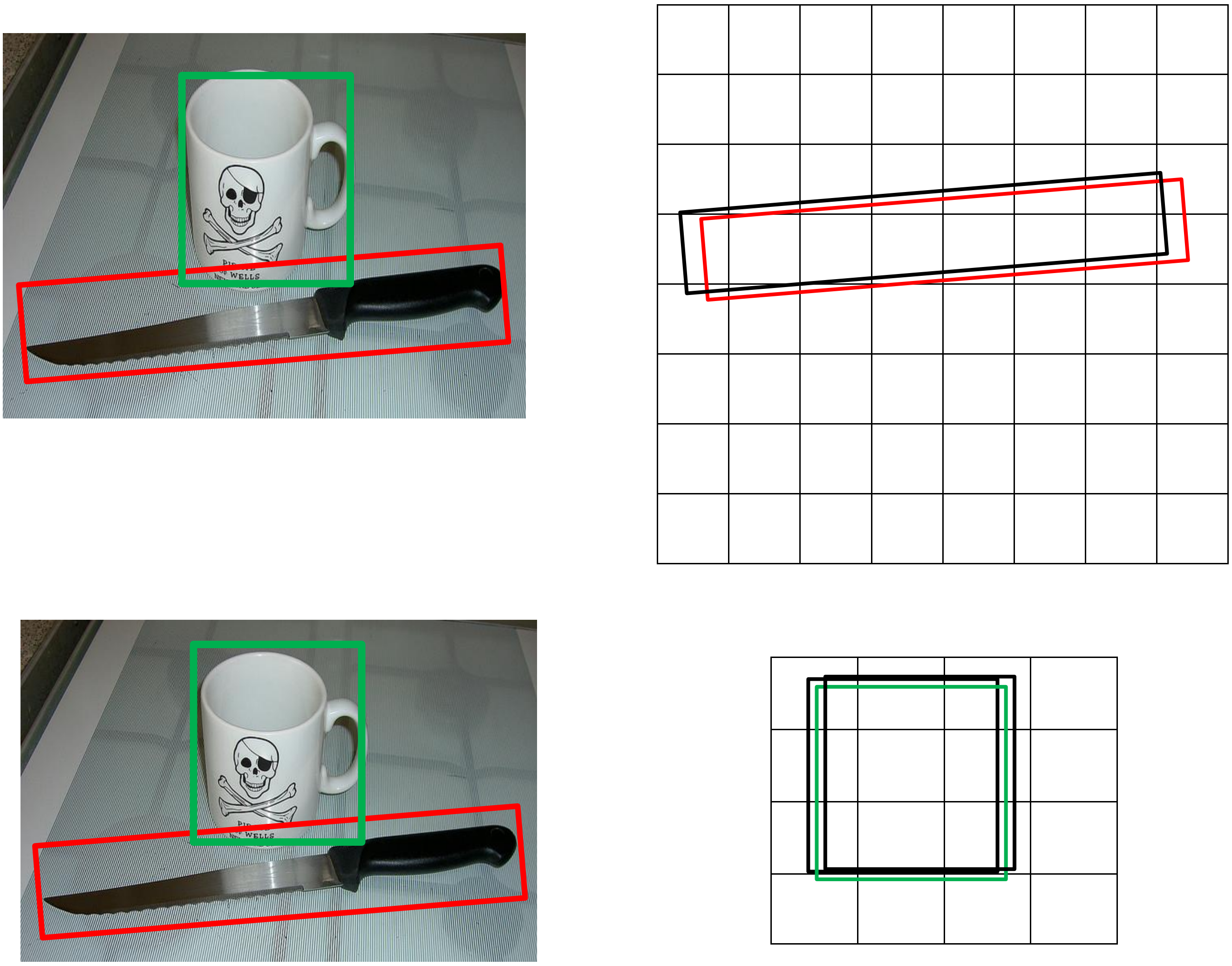


8x8 Feature Map



4x4 Feature Map

2. **Default Boundary Box:** Default boxes are bounding boxes that have been carefully picked based on their sizes, aspect ratios, and locations throughout the image. There are 8732 default boxes on SSD. The model's purpose is to choose which of the default boxes to use for a specific image, then predict errors from those default boxes to arrive at the final prediction.



Some major algorithms for object detection similar to SSD:

1. **RCNN:** R-CNN is a progressive visual object recognition system that combines bottom-up region proposals with convolution neural network-generated rich possibilities. Region-based convolutional neural networks or regions with CNN features (R-CNNs) are pioneering approaches that apply deep models to object detection. R-CNN models first select several proposed regions from an image and then label their categories and bounding boxes.

2. **YOLO:** Convolutional neural networks (CNN) are used in the YOLO method to recognise objects in real time. The algorithm has continued to evolve ever since its initial release in 2016. To detect objects, the approach just takes a single forward propagation through a neural network, as the name

suggests. A single neural network is used to detect objects in YOLO. It performs classification and bounding box regression in one step as a single-stage detector.

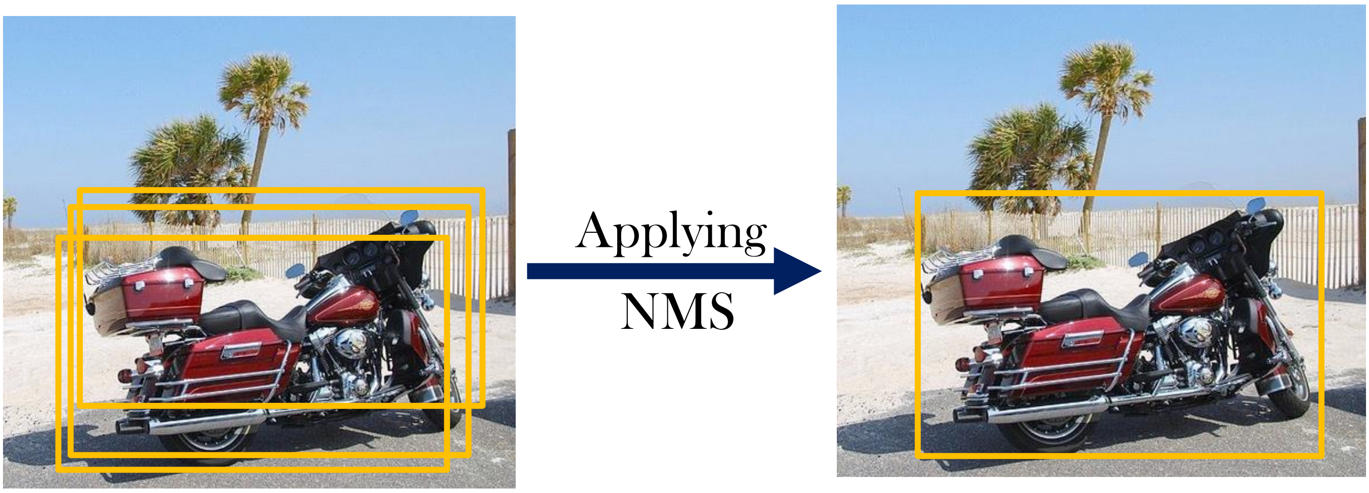
3. SqueezeDet: SqueezeDet is the name of a deep neural network that was released in 2016 for computer vision. It was developed mainly for autonomous driving, where it uses computer vision algorithms to detect objects. It's a single-shot detection algorithm, like YOLO. Convolutional layers are not only utilised to extract feature maps in SqueezeDet, but they are also employed as the output layer to compute bounding boxes and class probabilities. This models have only one forward pass of neural networks in their detection process, allowing them to be incredibly quick.

4. Mask R-CNN: Mask RCNN, is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation and instance segmentation. Mask R-CNN added a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. It is simple to train and adds only a small overhead to Faster R-CNN; it can run at 5 fps. Semantic and Instance segmentation are 2 main image segmentation types fall under Mask R-CNN.

Non-Maximum Suppression (NMS):

Non-Maximum Suppression is a computer vision approach that is utilised in a variety of tasks. It's a set of algorithms for selecting one entity (e.g., bounding boxes) out of several other overlapping entities. To get the desired results, we can choose the selection criteria. Commonly, some sort of probability number and some form of overlap measure are used as criteria (e.g. Intersection over Union). NMS [20] is used to whittle down many detected bounding boxes to only a few. Windowing is used by most object detectors at the most basic level. Hundreds of thousands of windows (anchors) of varying sizes and shapes are created.

These windows are said to contain only one object, and each class is assigned a probability/score by a classifier. After the detector generates a huge number of bounding boxes, the best ones must be filtered away. The most often used algorithm for this task is NMS.



The objects in the image can be diverse sizes and forms, therefore the object detection algorithms generate numerous bounding boxes to catch each of them precisely. (Image on the left) We should ideally have a single bounding box for each object in the image. Something along the lines of the illustration on the right. These object detection methods use non-max suppression to choose the best bounding box from a set of many predicted bounding boxes. This method is used to "suppress" the less likely bounding boxes and only maintain the best.

How does Non-Max Suppression work?

The goal of non-max suppression is to choose the best bounding box for an object while rejecting or "suppressing" any others. The NMS [28] takes two factors into consideration:

1. The objectiveness score is given by the model
2. The overlap or IOU of the bounding boxes

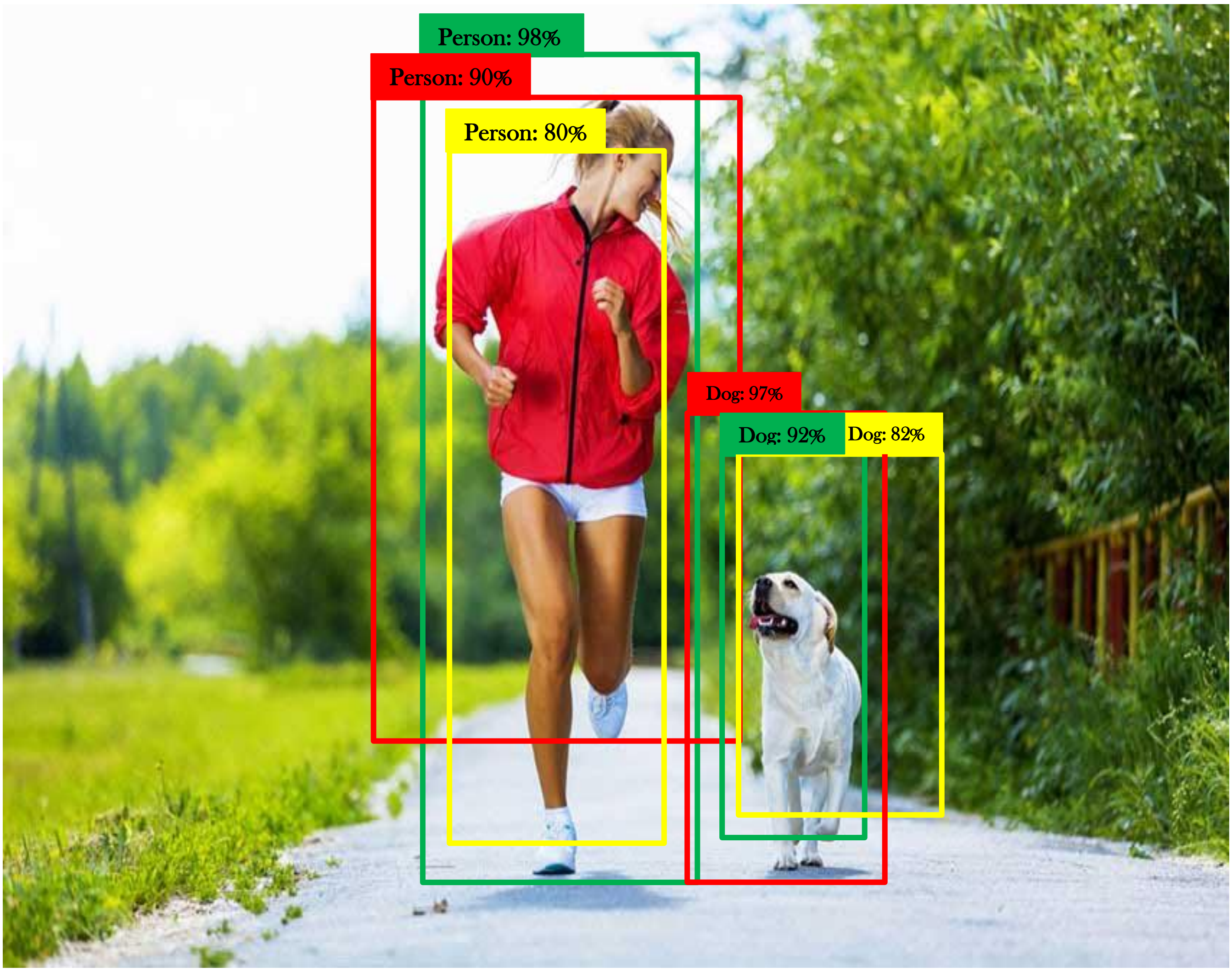


Fig 7. Example of Non-Maximum Suppression

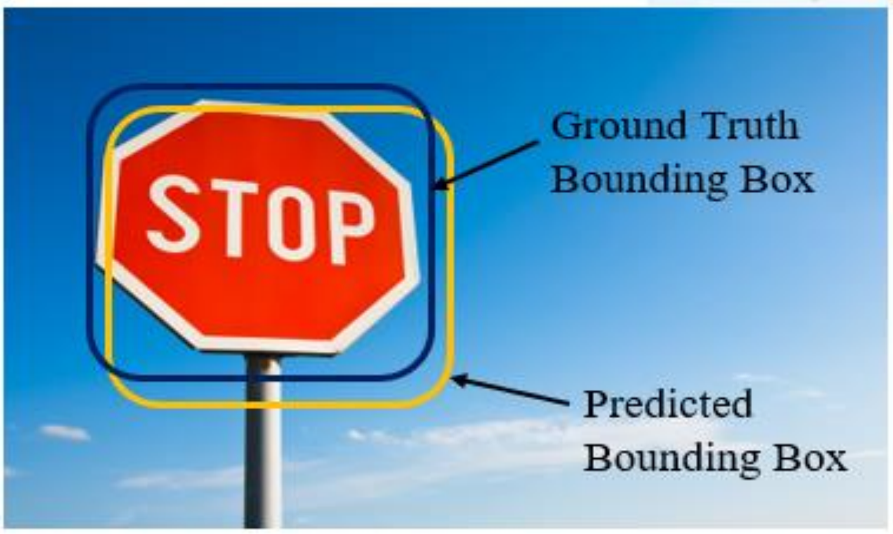
Intersection Over Union (IoU): The Jaccard index, also known as the Intersection over Union (IoU) metric, is a method for quantifying the percent overlap between the ground truth BBox (Bounding Box) and the forecast BBox. In NMS, however, we find IoU between two predicted BBoxes. IOU calculation is actually used to measure the overlap between two proposals. Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. IoU can be used to evaluate any method that produces anticipated bounding boxes as an output.

In order to apply Intersection over Union to evaluate an (arbitrary) object detector we need:

(i) The ground-truth bounding boxes (i.e., the hand labelled bounding boxes from the testing set that specify where in the image our object is).

(ii) The predicted bounding boxes from our model.

We can use Intersection over Union as long as we have these two sets of bounding boxes.



Application of Object Detection: There are many applications similar to object detection. Some of the major applications are:

1. **Pedestrian Counting:** Pedestrian counting is a type of object detection that can be used for a variety of applications, such as locating a person or a criminal, analysing store performance, or calculating crowd figures during gatherings. This is a difficult task because people move out of the frame rapidly. 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 to automate inspection tasks on remote locations with computer vision.

2. **Self-driving Cars:** Self-driving cars are the most promising technology for the future, but their operation is complicated because they use a number of techniques to observe their surroundings, such as radar, laser light, GPS, odometer, and computer vision. Sensory data is interpreted by advanced control systems to allow navigation methods, as well as obstacles and it, to work. This is a significant step toward

driverless cars because it occurs at a rapid rate. Object detection is used by self-driving automobiles to distinguish pedestrians, traffic signs, other vehicles, and other objects. For example, Tesla's Autopilot AI mainly relies on object detection to detect risks in the surroundings, such as oncoming vehicles or barriers.

3. **Facial Recognition:** "Deep Face" is a facial recognition system that uses deep learning to recognise human faces in digital images. A group of Facebook researchers designed and built the app. In Google Photos, Google has its own facial recognition algorithm, which automatically separates all of the photos based on the person in the image. Facial Recognition involves several components, or as the authors put it, it concentrates on various elements such as the eyes, nose, mouth, and brows in order to recognise a face.

4. **Security:** Object detection is important in the field of security; it's employed in significant fields like Apple's Face ID and the retina scan seen in all sci-fi movies. Government also makes extensive use of this programme to access security feeds and compare them to their existing databases in order to identify criminals or detect things such as car numbers implicated in criminal activity. The possibilities are endless.

5. **Medical Feature Detection:** Object detection has allowed for many breakthroughs in the medical community. Because medical diagnostics rely heavily on the study of images, scans, and photographs, object detection involving CT and MRI scans has become extremely useful for diagnosing diseases, for example with ML algorithms for tumor detection.

6. **Animal Detection in Agriculture:** Counting, animal monitoring, and evaluating the quality of agricultural goods are all tasks that object detection is employed for in agriculture. Machine learning algorithms can detect damaged produce while it is being processed. Animal detection using surveillance cameras' can help farmers, living near forests to protect themselves and their agriculture from wild animals.

7. **Vehicle detection with AI in Transportation:** Object recognition is used to detect and count vehicles for traffic analysis and to detect cars that stop in dangerous areas, for example, on crossroads or highways. Using CCTV cameras, we can also detect the vehicle plate number, who will exceed the speed limit in crowded roads, to prevent roadside accidents in the upcoming eras.

8. **Industrial Quality Check:** Object detection is also useful in industrial processes for identifying and recognising products. Finding a specific object through visual examination is a basic activity that is used in a variety of industrial operations, including sorting, inventory management, machining, quality control, and packaging. Inventory management is difficult since goods are difficult to track in real time. Improved inventory accuracy is possible thanks to automatic object counting and localization.

LIBRARIES AND PACKAGES

In this project, we used PyCharm to implement Object Detection using Deep Neural Network.

PyCharm:

In this project we used PyCharm emulator. PyCharm [22] is an integrated Development Environment (IDE) used in computer programming which is specifically for the python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). In PyCharm there are three editions:

(1) Professional Edition,

(2) Community Edition,

(3) Edu Edition.

Community Edition are used for smart and intelligent Python development, including code assistance, refactorings, visual debugging, and version control integration. Professional Edition includes features provided in Community Edition. It is also used for professional Python, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools. Edu Edition consists learning programming languages and related technologies with integrated educational tools.

Here, we used Professional Edition of the PyCharm 2021.3. PyCharm is a python IDE. It provides code analysis, a graphical debugger, macOS and Linux versions. The community Edition was released under the Apache License, and there is also a Professional Edition with extra features, which was released under a Proprietary License.

PyCharm provides an API, so that developers can write their own plugins with an extended PyCharm features. There are many plugins which are compatible with PyCharm.

Some of PyCharm features are:

1) Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
2) Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes
3) Integrated Python debugger
4) Integrated unit testing, with line-by-line code coverage
5) Support for scientific tools like matplotlib, numpy and scipy [professional edition only]

OpenCV:

OpenCV (Open Source Computer Vision Library) [23] is a free software library for computer vision and machine learning. OpenCV was developed to provide a common infrastructure for computer vision applications and to assist commercial products integrate machine perception more efficiently. OpenCV is a BSD-licensed product, also it is easier for businesses to use and alter the code. The library has used more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms in the platform.

These algorithms is used for various functions such as discover and acknowledging faces. Identify objects classify human actions. In videos, track camera movements, track moving objects. Extract 3D models of objects, manufacture 3D purpose clouds from stereo cameras, sew pictures along to provide a high-resolution image of a complete scene, find similar pictures from a picture information, remove red eyes from images that are clicked with the flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms.

OpenCV's application areas include:

- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object detection
- Segmentation and recognition
- Motion tracking
- Augmented reality
- Structure from Motion (SFM)

Numpy:

NumPy [24] is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. NumPy will always be 100% open source software, free for all to use and released under the liberal terms of the modified BSD license.

In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

A new package called *Numarray* was written as a more flexible replacement for Numeric. Like Numeric, it too is now deprecated. Numarray had faster operations for large arrays, but was slower than Numeric on small ones, so for a time both packages were used in parallel for different use cases.

The last version of Numeric (v24.2) was released on 11 November 2005, while the last version of numarray (v1.5.2) was released on 24 August 2006.

NumPy applications are:

1. Maintains minimal memory
2. Multi-dimensional Arrays
3. Mathematical Operations

Numpy Array applications are:

1. Shape Manipulations
2. Array Generation
3. Array Dimensions

PROPOSED SYSTEM

We will detect objects in real time using the Mobilenet-SSD architecture in a quick and effective manner. We'll use OpenCV version 4.5.4.60 to construct a Python code for object detection using a deep neural network.

The system operates as the user gives the input. via real-time webcam and static images. MobileNet architecture optimizes the images and builds light-weight deep neural networks. MobileNet layers pre-processes the images. MobileNet layers receives the input images, and separates into frames. Each feature value is computed by reducing the pixel intensity of brighter region from darker region. Each feature value is computed by reducing the pixel intensity of brighter region from darker region.

The layers then converts the pixels from the selected regions of the input image into highlights that describes the image's contents. The input image is divided into [SxS] grid. For each cell it predicts bounding boxes (bbox). SSD detects many objects per grid cell regardless of the number bounding boxes. It predicts conditional class probabilities (conf). If no objects exists then confidence score is zero, else confidence score should be greater or equal to threshold value.

SSD then draws bounding box around the detected objects and predicts the feature class to which the object belongs. After calculating the confidence, Non-Maximum Suppression (NMS) is applied to predict the maximum confidence scored bounding box, so as to remove overlapping of too many bounding boxes. While detecting the 2-D images and via webcam, the classIds, class names (i.e. COCO names) and the confidence score is printed in the run terminal.

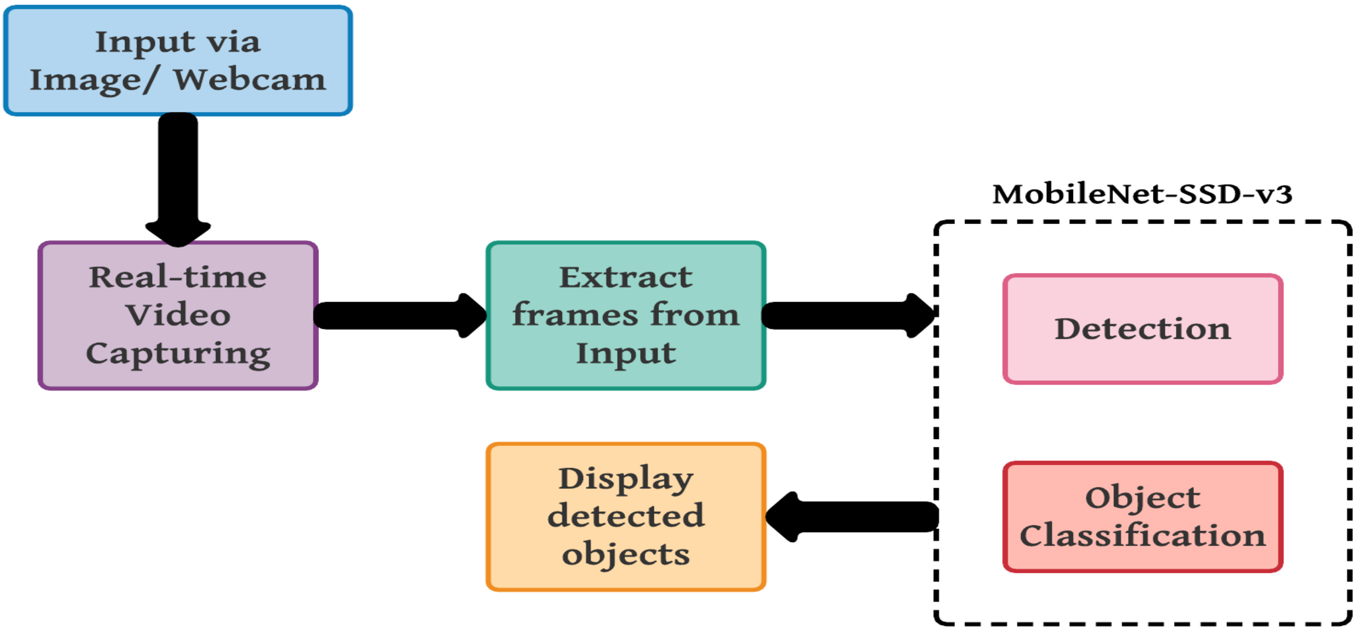


Fig 8. Proposed System Architecture Diagram.

SYSTEM MODEL

a. Software & Hardware Requirement Specifications

Python Compiler:	PyCharm, Python IDLE

- **Minimum System Requirements:**

Operating System:	Windows 8/8.1/10 (64-bit)
Complier:	Python IDLE
RAM:	4GB
Processor:	Intel i3 9th Generation
Disk Space:	120GB and another 2GB for caches
GPU:	2GB DDR4
Monitor Resolution:	1024x768

- **Recommended System Requirements:**

Operating System:	Windows 10 (64-bit)
Compiler:	PyCharm, VS, Anaconda, Eclipse
RAM:	8GB
Processor:	Intel i5 10th generation
Disk Space:	1TB WD
GPU:	8GB DDR5
Monitor Resolution:	1920x1080
SSD:	SSD drive with at least 5GB of free space

b. Planning

We followed the below steps to develop our project:

- Planning and analysing our problem statement.
- Specified requirements gathering needed to build our project.
- Analysing the requirements.
- Feasibility study analysis.
- Designing and creating the general layout.
- Doing a literature survey based on the previous related work similar to our topic.
- Selecting the algorithm for developing our model.
- Analysing the advantages and disadvantages.
- Started developing our project according to the planning.
- Installed the “**PyCharm Professional Edition 2021.3**” version.
- Checking and analysing the developed algorithm by project guide.
- Converting the design to machine-readable language.
- Coding the developed algorithm and the design using “**Python 3.10**”.
- Testing the functionality of the entire system.

According to our planning of project, the agile model below is drawn.

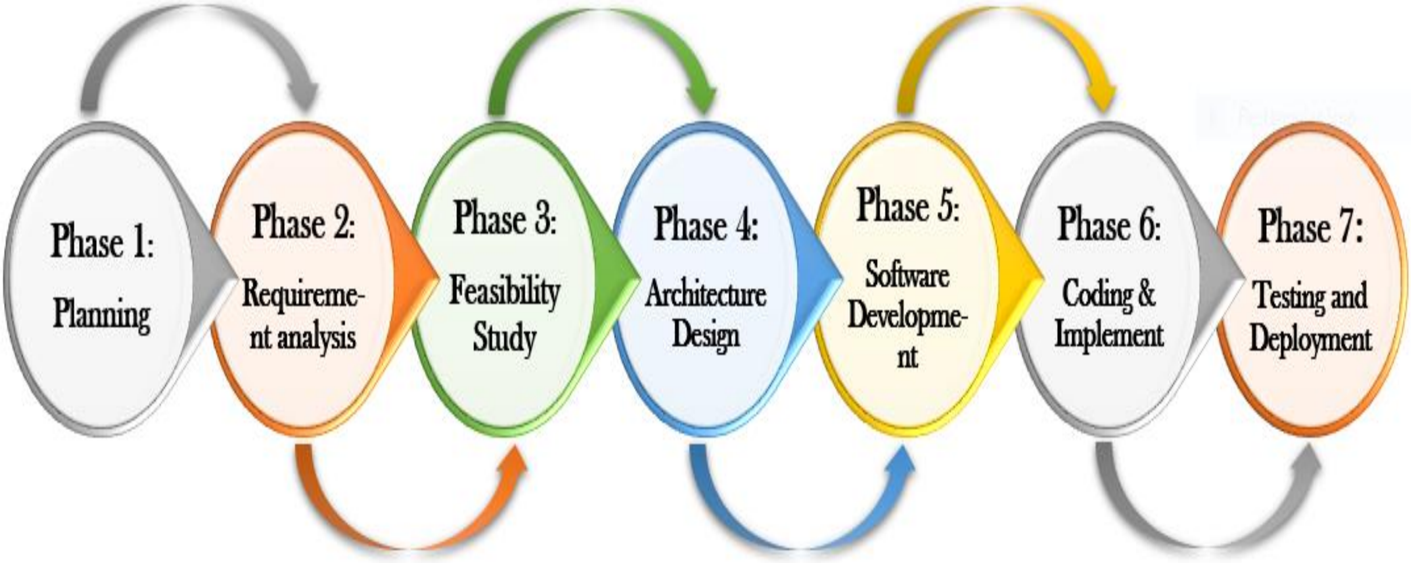


Fig 9. Agile Model

c. Design

FLOWCHART

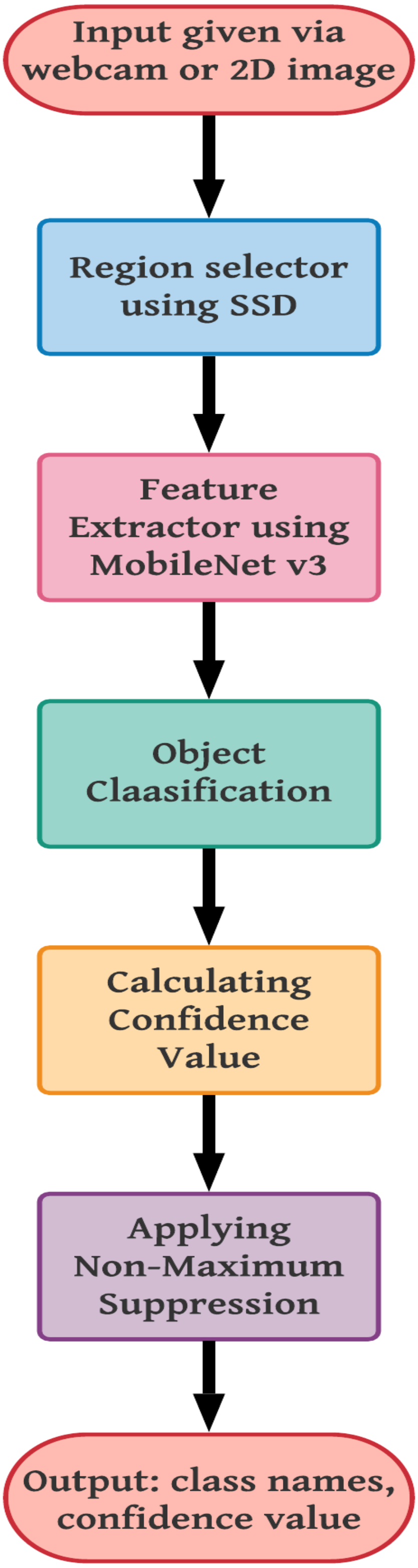

Input given via webcam or 2D image
Region selector using SSD
Feature Extractor using MobileNet v3
Object Claasification
Calculating Confidence Value
Applying Non-Maximum Suppression
Output: class names, confidence value

Fig 10. Flow Chart of Object Detection

DATA FLOW DIAGRAM

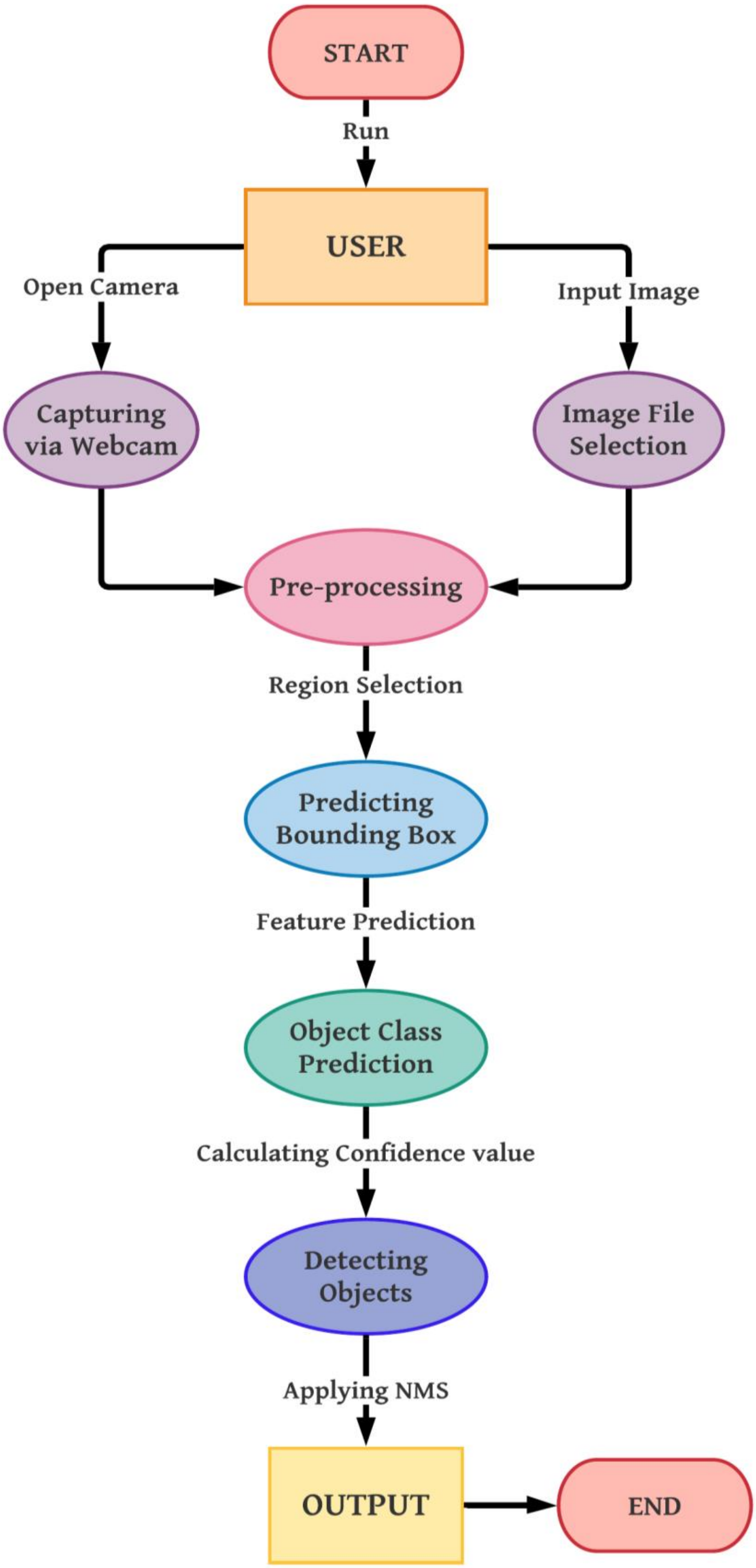


Fig 11. Data Flow Diagram of Object Detection System.

IMPLEMETATION

A. Modules

Algorithm for Object Detection System:

1. The input image is divided into **[SxS] grid**.

2. For each cell it predicts **bounding boxes (bbox)**.

3. Each bounding box contains five elements: **(x, y, w, h)** and a box **confidence score**.

4. **SSD** detects many objects per grid cell regardless of the number bounding boxes.

5. It predicts conditional **class probabilities (conf)**.

6. If no objects exists then confidence score is zero. Else confidence score should be greater or equal to **threshold value**.

7. **SSD** then draws bounding box around the detected objects and predicts the class to which the object belongs.

8. **Non-Maximum Suppression (NMS)** is used to predict the maximum confidence scored bounding box and to remove overlapping of too many bounding boxes.

9. If static images are taken as input then, the bounding box, predicted class names and the confidence score will be printed using **random colors**.

10. While detecting the 2-D images and via webcam, the classIds, class names (i.e. **COCO names**) and the confidence score is printed in the run terminal.

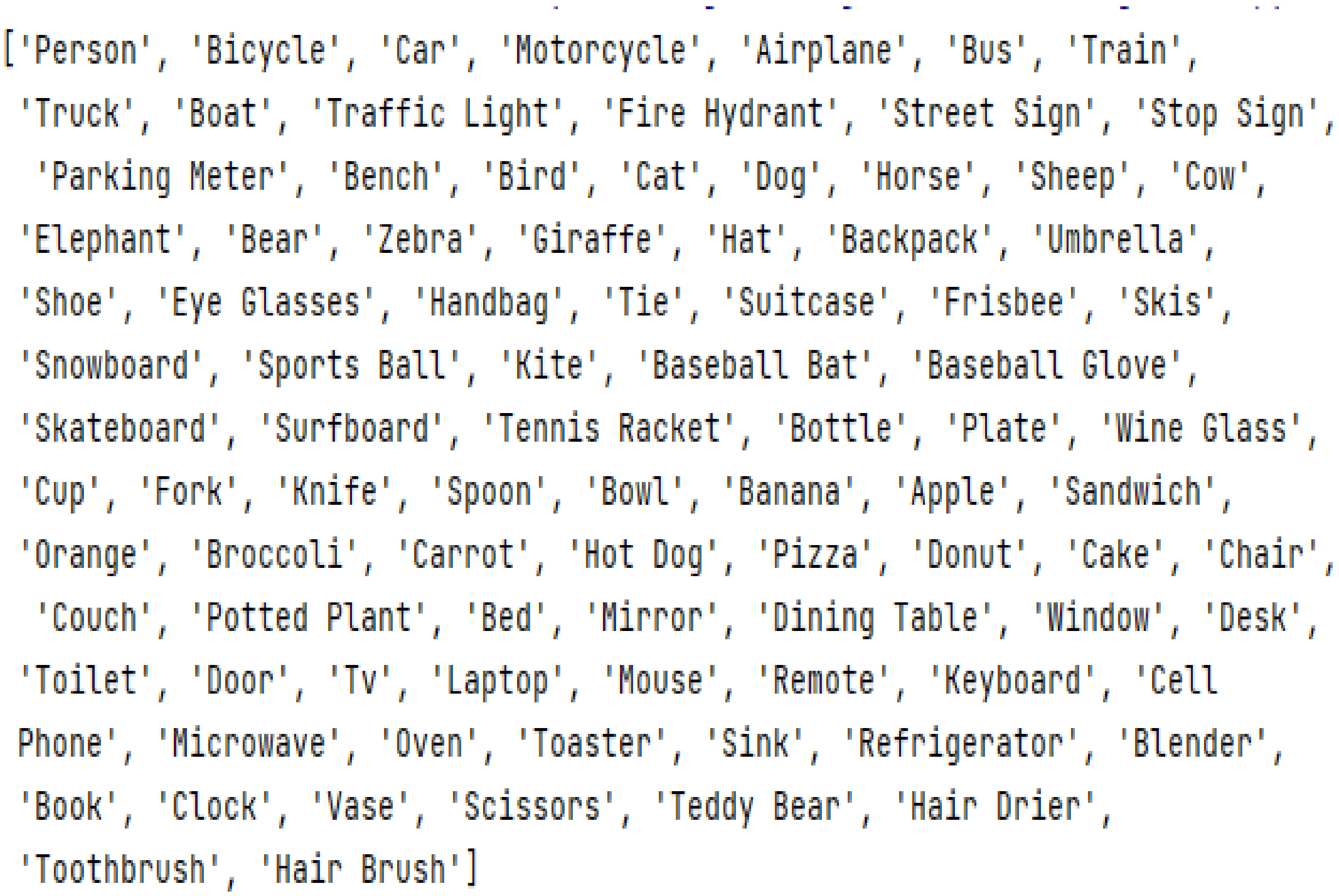
B. Module Description

1. Dataset Requirements:

We attempted to recognise items using the OpenCV library and a deep learning pre-trained model. Our model was pre-trained using the SSD method. We used Mobilenets' pre-trained models to build the SSD technique. This method may classify labels based on the learned model. We used the MS-COCO dataset [15] [21], which had 91 classifications, as shown in **fig 12**.

We take the input video and static images and transform them to a single frame input drop by scaling each frame to a size of 300x300 pixels. Two files are included with our pre-trained models: one for configuration and the other for weights. As a result, the model represents how neurons in a neural network are grouped in two ways: 1- Configure and 2- Weights.

In 2017, the training and validation split was changed from 83K/41,000 to 118K/51,000. In the new split, the same photographs and notes are used. The 41K images in the 2017 testing set are a variation of the 41K pictures in the 2015 testing set. The 2017 edition 7 includes a new unlabelled dataset with 123K pictures.



```
['Person', 'Bicycle', 'Car', 'Motorcycle', 'Airplane', 'Bus', 'Train',
 'Truck', 'Boat', 'Traffic Light', 'Fire Hydrant', 'Street Sign', 'Stop Sign',
  'Parking Meter', 'Bench', 'Bird', 'Cat', 'Dog', 'Horse', 'Sheep', 'Cow',
 'Elephant', 'Bear', 'Zebra', 'Giraffe', 'Hat', 'Backpack', 'Umbrella',
 'Shoe', 'Eye Glasses', 'Handbag', 'Tie', 'Suitcase', 'Frisbee', 'Skis',
 'Snowboard', 'Sports Ball', 'Kite', 'Baseball Bat', 'Baseball Glove',
 'Skateboard', 'Surfboard', 'Tennis Racket', 'Bottle', 'Plate', 'Wine Glass',
 'Cup', 'Fork', 'Knife', 'Spoon', 'Bowl', 'Banana', 'Apple', 'Sandwich',
 'Orange', 'Broccoli', 'Carrot', 'Hot Dog', 'Pizza', 'Donut', 'Cake', 'Chair',
  'Couch', 'Potted Plant', 'Bed', 'Mirror', 'Dining Table', 'Window', 'Desk',
 'Toilet', 'Door', 'Tv', 'Laptop', 'Mouse', 'Remote', 'Keyboard', 'Cell
 Phone', 'Microwave', 'Oven', 'Toaster', 'Sink', 'Refrigerator', 'Blender',
 'Book', 'Clock', 'Vase', 'Scissors', 'Teddy Bear', 'Hair Drier',
 'Toothbrush', 'Hair Brush']
```

Fig 12. MS-COCO large dataset (91 classes)

2. MobileNet-SSD-v3 Architecture:

To reduce computational complexity and increase the SSD detector's real-time performance, Google used the MobileNet network model [16]. The MobileNet network is the backbone network model for the SSD detector. The MobileNet-v3 [18] technique is utilised to improve the SSD algorithm and speed rating precision in real time. MobileNet is a type of convolutional neural network designed for mobile and embedded vision applications. MobileNetV3 is the third version of the architecture powering the image analysis capabilities of many popular mobile applications.

This method demands a single shot to detect many objects. The SSD [19] is a neural network architectural design for detection applications. This means that both categorization and localisation are happening at the same time. SSD reveals the limited output space of the basic box set. This network quickly scans a standard box for the presence of various object classes and then reassembles the box to fit what's inside. MobileNetV3 is the use of AutoML to find the best possible neural network architecture. In object detection tasks, MobileNetV3 operated with 25% less latency and the same accuracy of previous versions.

This network quickly scans a basic box for the presence of various object classes and then reconfigures the box to fit what's inside. This network can also handle a wide range of models with different natural bond sizes and resolutions. We also used Non-Maximum Suppression [20] to rule out the vast majority of these bounding boxes, either because their confidence is low or because they surround the same object as another bounding box with a high confidence score.

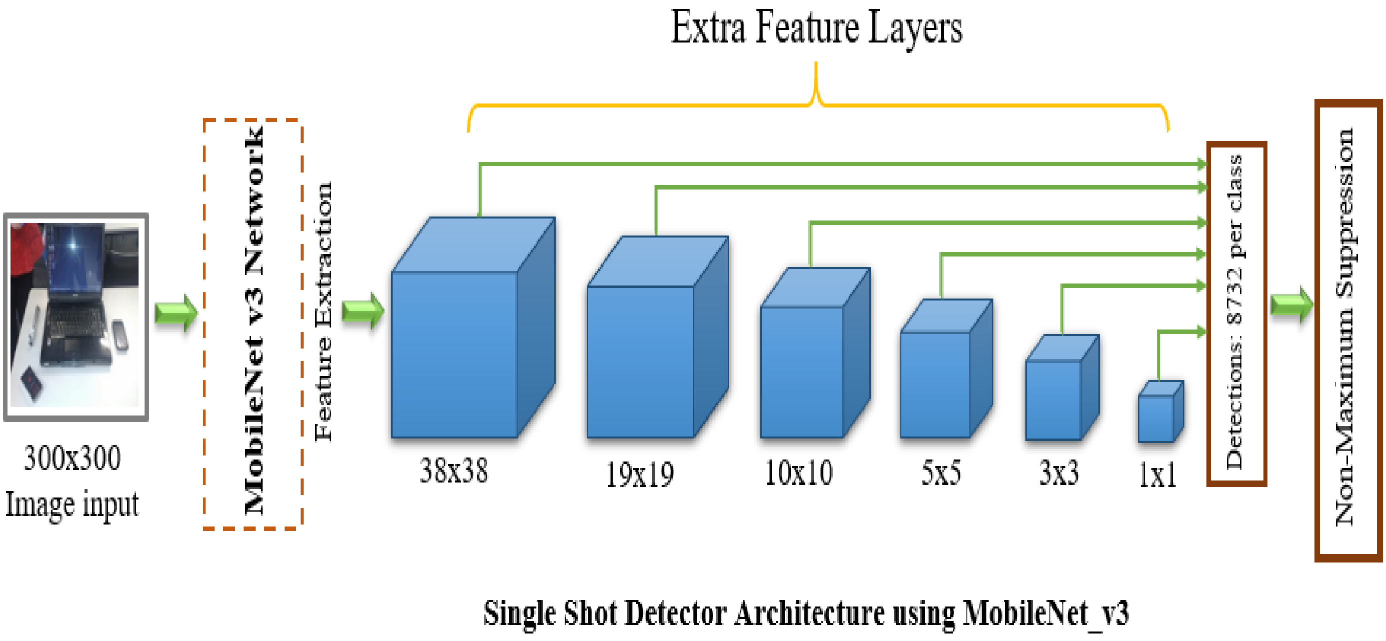
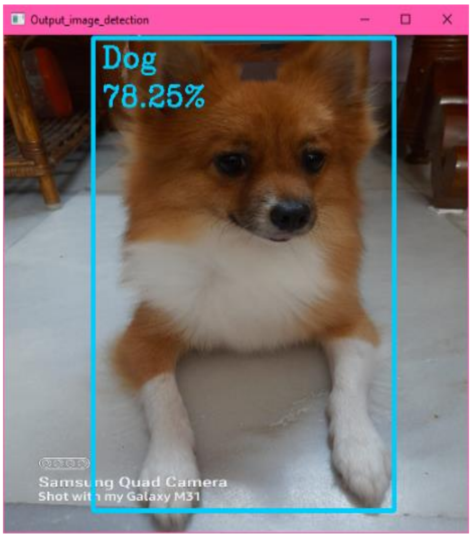
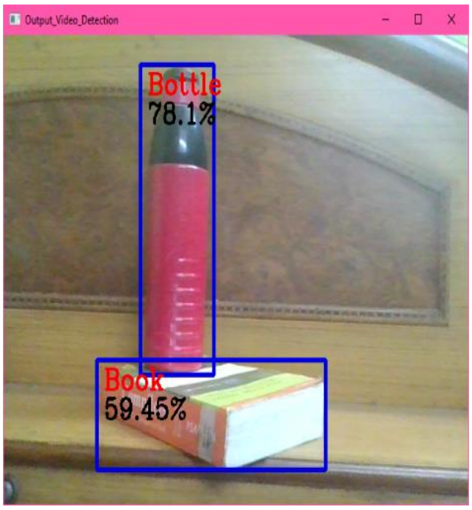
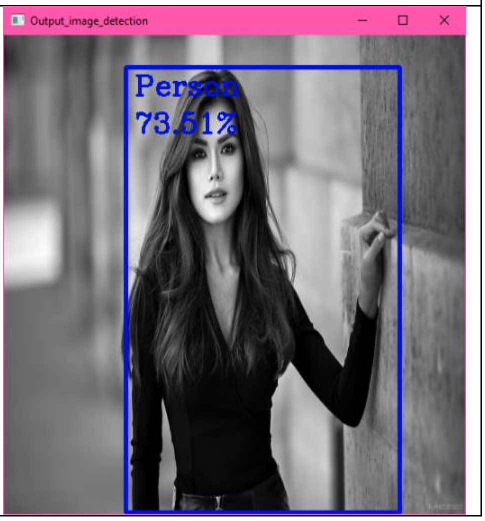
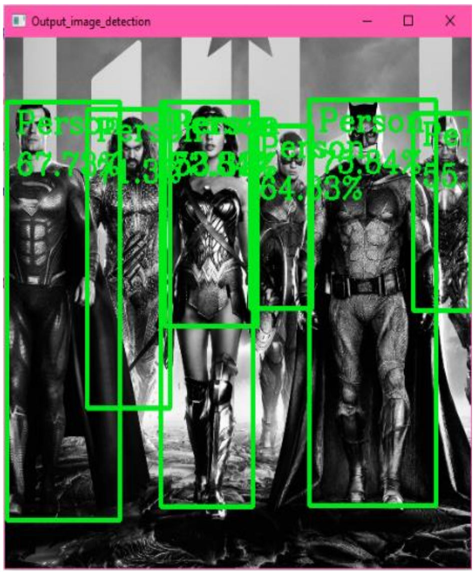
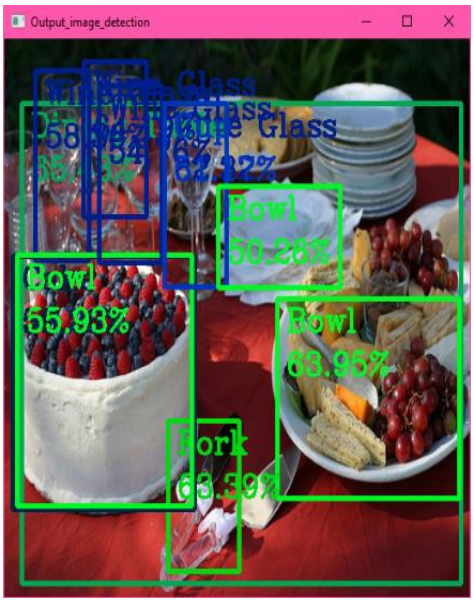
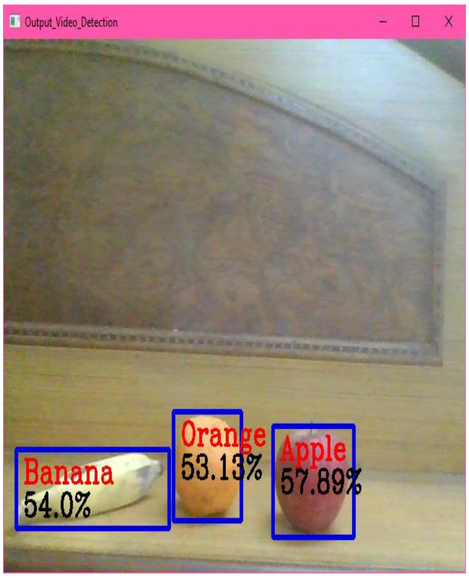


Fig 13. SSD Architecture using MobileNet_v3 as backbone.

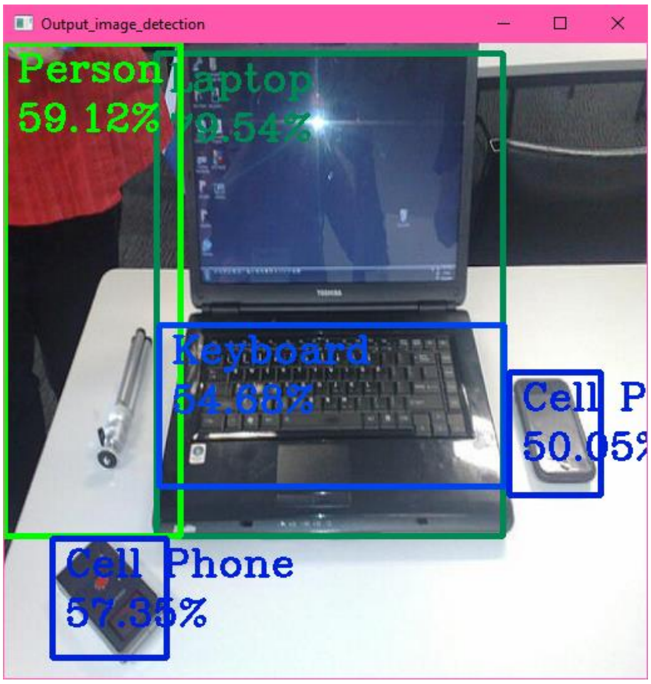
EXPERIMENTAL RESULTS

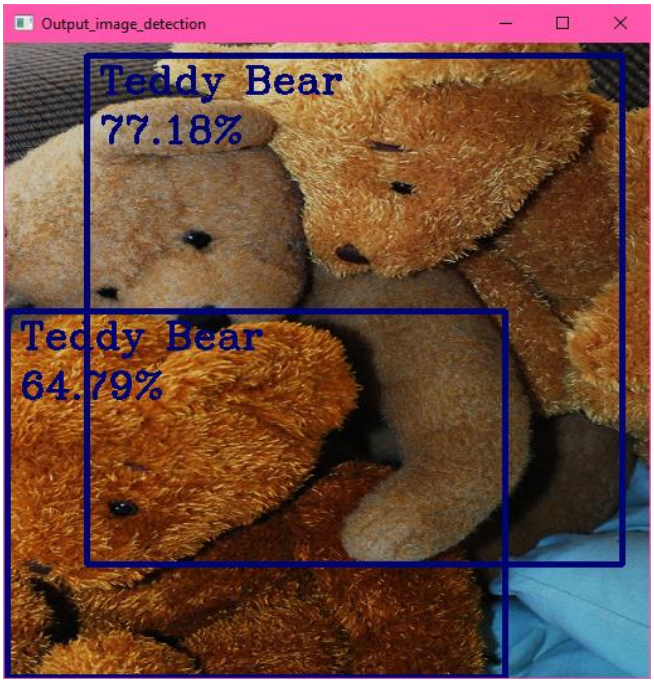
a. Test Results

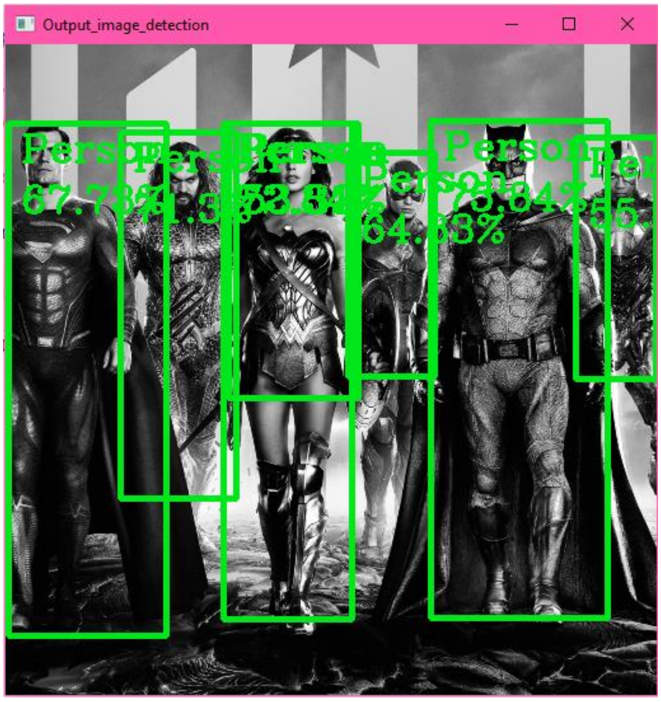
Test Condition	Expected Results	Test Results	Output Image
When input taken from static images	Image with random coloured bounding box around the objects, predicted class & Confidence value.	SUCCESSFUL	 Output_image_detection Dog 78.25% Samsung Quad Camera Shot with my Galaxy M31
When input taken from webcam	Image with bounding box around the objects, predicted class & Confidence value.	SUCCESSFUL	
When black and white image is taken as input	Image with random coloured bounding box around the objects, predicted class & Confidence value.	SUCCESSFUL	

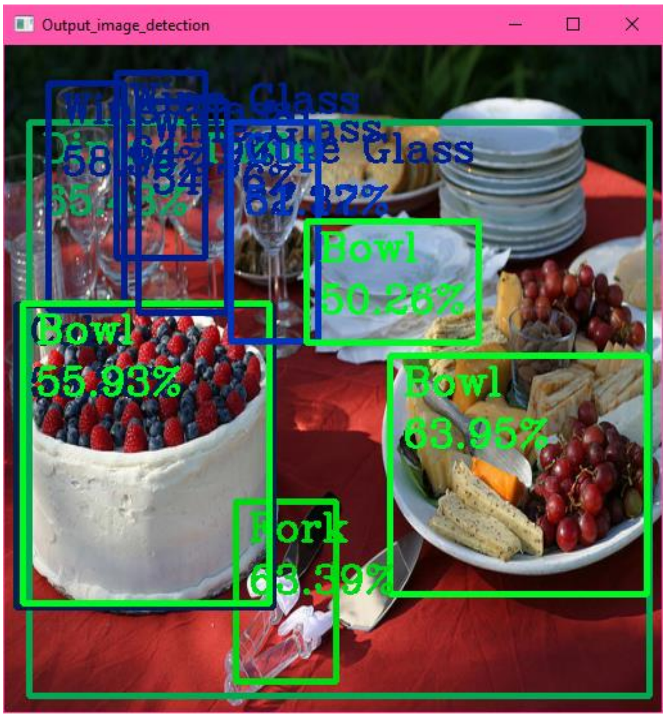
When black and white image with overlapping objects is taken as input	Image with random coloured bounding box around the objects, predicted class & Confidence value.	SUCCESSFUL	 Output_image_detection Person 67.79% Person 64.43% Person 73.84%
When colour image with overlapping objects is taken as input	Image with random coloured bounding box around the objects, predicted class & Confidence value.	SUCCESSFUL	 Output_image_detection Wine Glass 62.32% Bowl 50.28% Bowl 55.93% Bowl 63.95% Fork 63.39%
When colour image with overlapping objects is taken as input from webcam	Image with bounding box around the objects, predicted class & Confidence value.	SUCCESSFUL	 Output_Video_Detection Banana 54.0% Orange 53.13% Apple 57.89%

b. Screen Shots









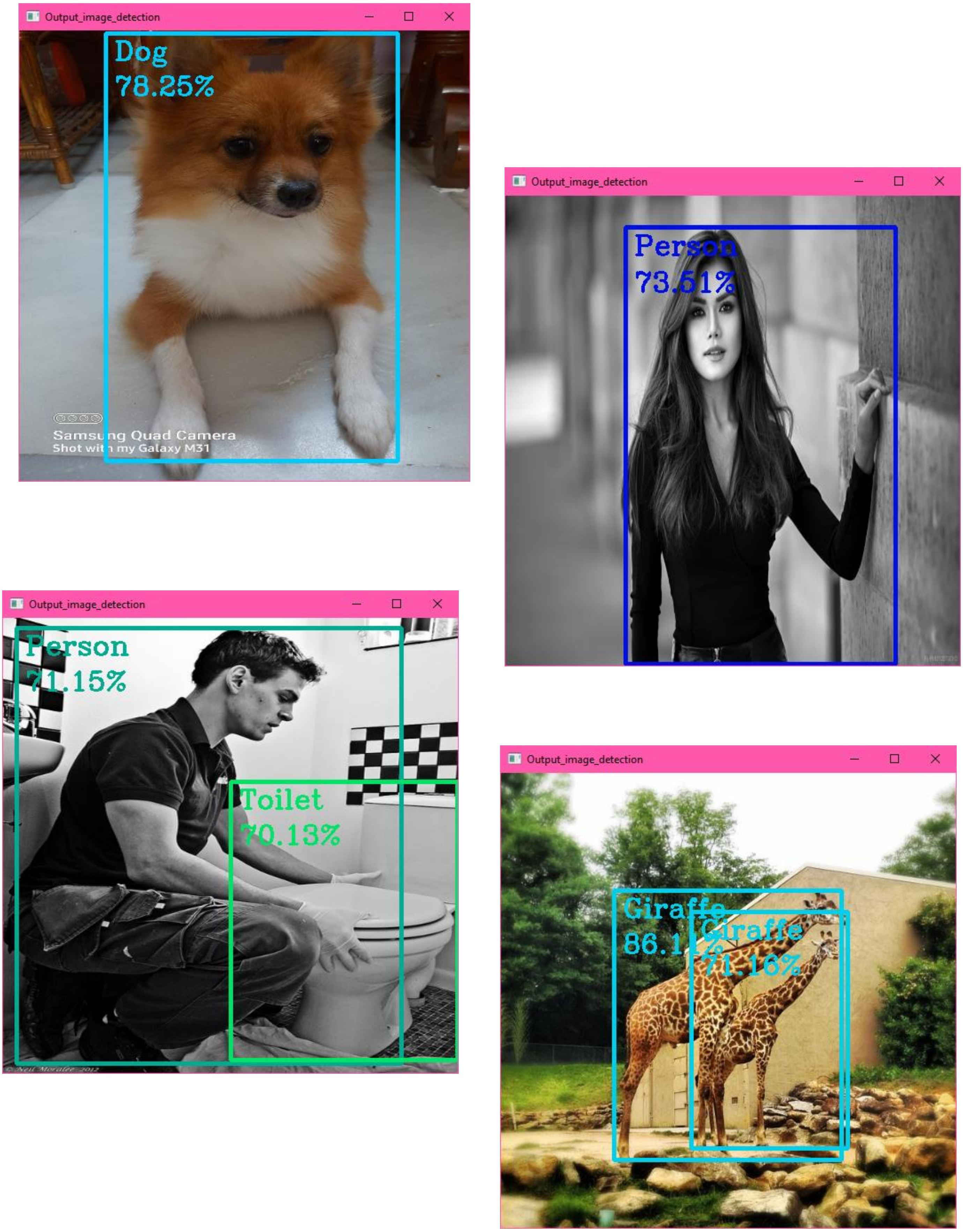
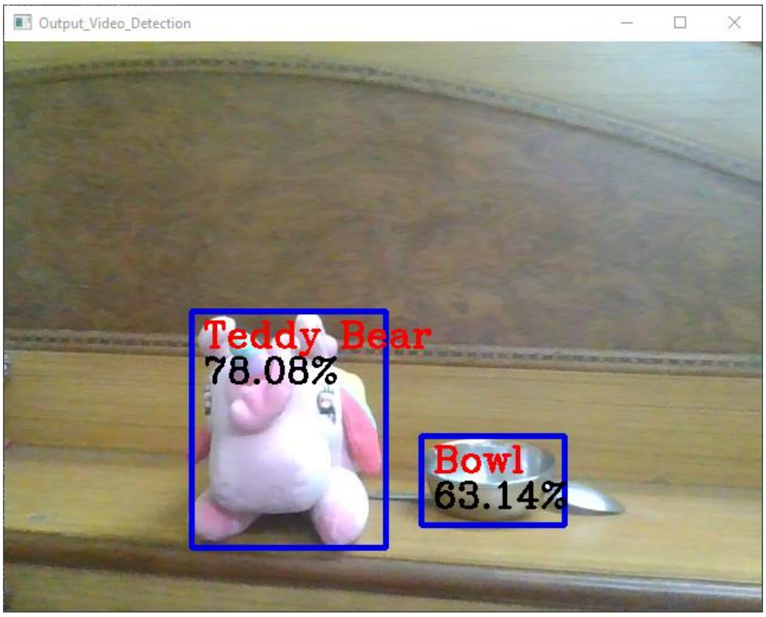
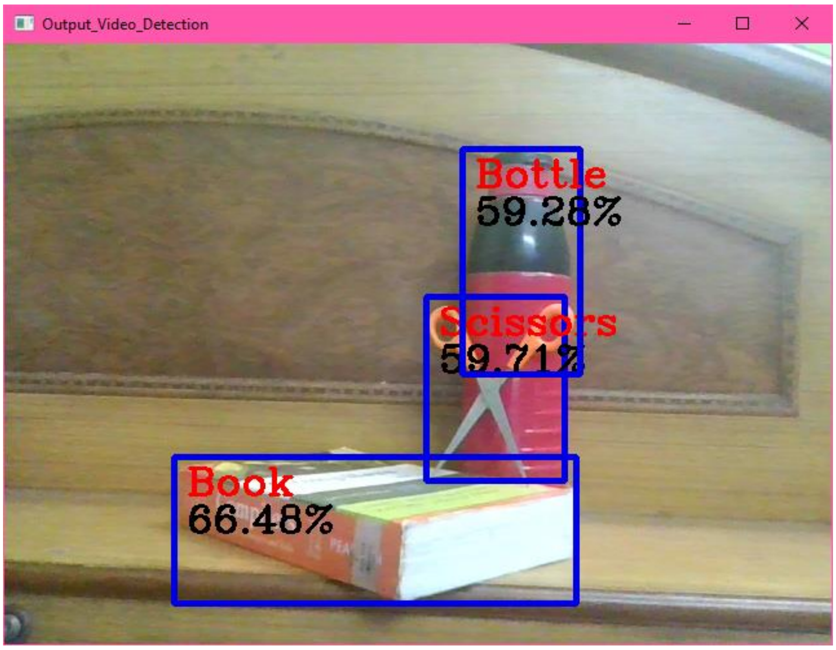
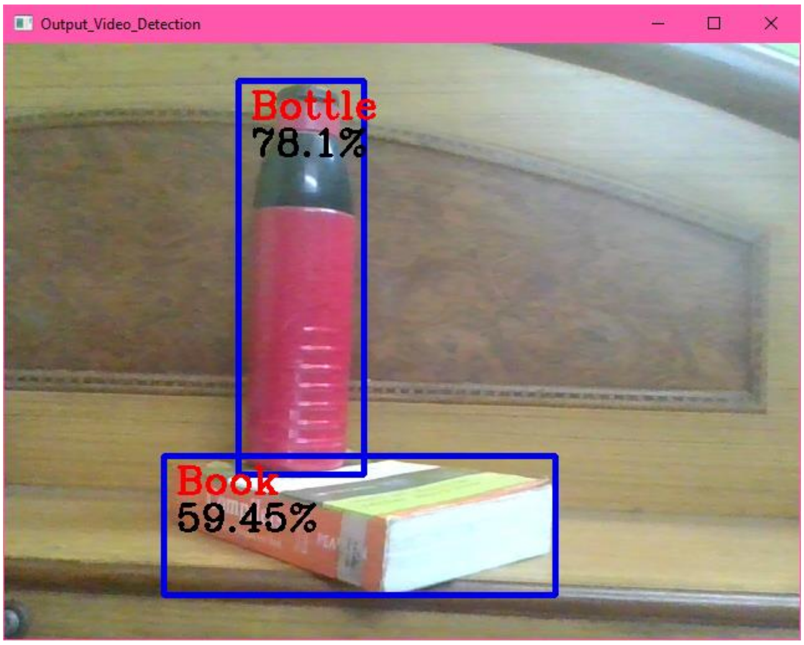
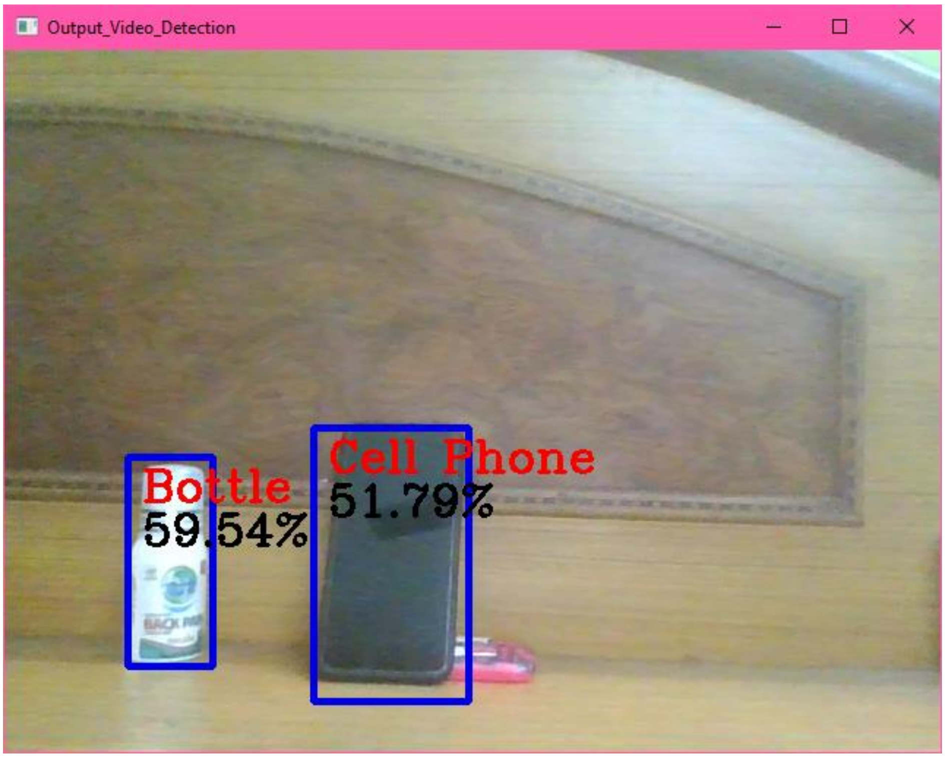


Fig 14. Outputs taken from 2D images.


Output_Video_Detection
Teddy Bear
78.08%
Bowl
63.14%


Output_Video_Detection
Bottle
59.28%
Scissors
59.71%
Book
66.48%


Output_Video_Detection
Bottle
78.1%
Book
59.45%



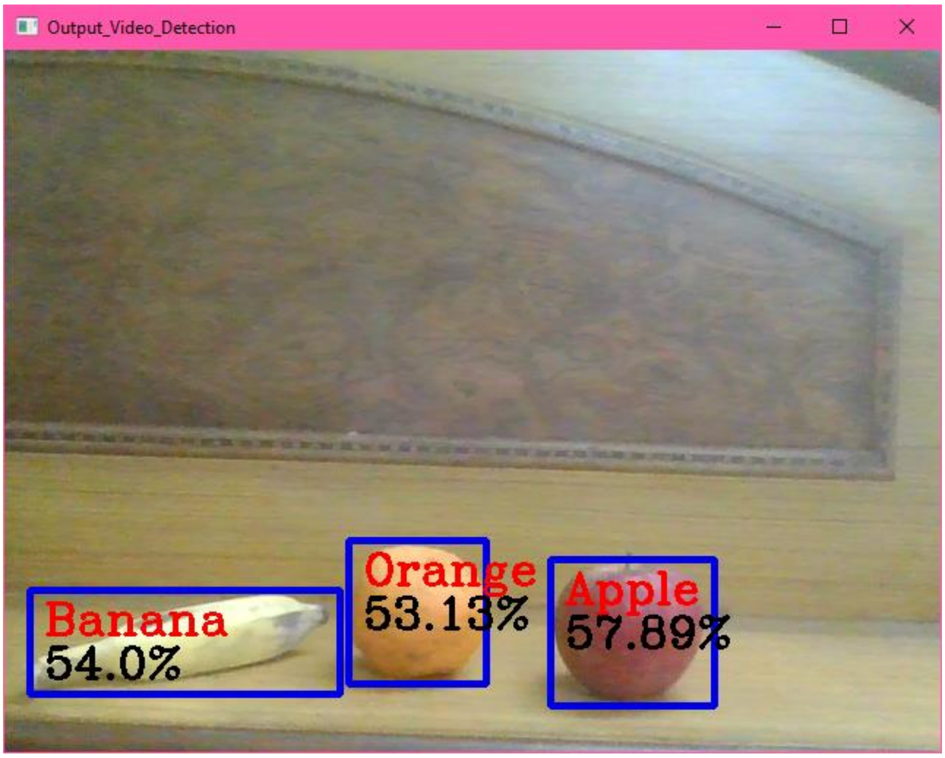


Fig 15. Outputs taken from real-time webcam.

CONCLUSION

Object detection using deep learning has become a research hotspot in recent years. We attempted to recognise an object that was exhibited in front of a webcam in this study. The obtained model was pre-trained using MobileNet and Single Shot Multi-Box Detector. Reading a frame from a web camera causes a slew of issues, thus a good frames per second solution is built to alleviate Input / Output worries.

We proposed a lightweight network design with superior feature extraction based on the Mobilenet-v3 backbone network. We combine the Mobilenet-v3 and SSD models to improve the feature map of the input image and the detection accuracy of the back-end detection network. With the unique placement of an object in the frame in the x, y axis, we are able to detect objects more precisely and identify them individually based on testing results.

This research also provides experimental results on multiple ways for item detection and identification, as well as a comparison of each approach's efficiency. This is a major gain for x86 hardware with limited resources.

Experiments show that the suggested Mobilenet-SSDv3 detector not only keeps the original Mobilenet-SSD detector's rapid processing benefit, but also improves detection performance significantly. This is achieved by combining two techniques: deep learning with OpenCV for object detection, and OpenCV for efficient, threaded video streaming. By using MobileNet and the SSD detector for object detection, a high precision object detection approach has been established, making it efficient to all cameras. A high accuracy object detection solution has been built combining MobileNet and the SSD detector, making it efficient for all cameras. The end result is a deep learning- based object detector that can process around 6-30 FPS.

Our system can recognise objects in its dataset, such as person, animals, fruits, vegetables, cars, a motorcycle, bottles, a couch, and so on. The goal of this study is to create an autonomous system in which object and scene recognition aids the community in making the system more engaging and appealing.

FUTURE STUDY

Our goal in this research is to create an Object Recognition system that can recognise 2-D and 3-D items in an image. Surveillance systems, face identification, fault detection, character recognition, and other applications can all benefit from the object recognition system. The characteristics used and the classifier used for recognition determine the performance of the object recognition system. In addition, the research endeavour aims to combine classical classifiers in order to recognise the item.

Color information from photos can be used to improve object recognition accuracy, which is important in robotics. Night vision mode could potentially be included as an option in tracking devices and CCTV cameras. Because of its extensive coverage and diverse viewing angles for the objects to be tracked, multi-view tracking can be performed utilising many cameras to make the systems entirely automatic and overcome the preceding restrictions.

We can also utilise real-time object detection to create self-driving automobiles that can identify items approaching from the front. We can use it for new era children's, which can help them to learn and identify new things at home digitally.

This research will be used primarily in the future to identify items with high functionality features in the external world. Future improvements can be concentrated by putting the project on a system with a GPU for faster results and more accuracy.

REFERENCES

[1] Younis, A., Shixin, L., Jn, S., & Hai, Z. (2020, January). **Real-time object detection using pre-trained deep learning models MobileNet-SSD.** In Proceedings of 2020 the 6th International Conference on Computing and Data Engineering (pp. 44-48). doi: https://doi.org/10.1145/3379247.3379264

[2] Xiao, Youzi, Zhiqiang Tian, Jiachen Yu, Yinshu Zhang, Shuai Liu, Shaoyi Du, and Xuguang Lan. "**A review of object detection based on deep learning**." Multimedia Tools and Applications 79, no. 33 (2020): 23729-23791.

[3] Heredia, A., & Barros-Gavilanes, G. (2019, June). **Video processing inside embedded devices using ssd-mobilenet to count mobility actors**. In 2019 IEEE Colombian Conference on Applications in Computational Intelligence (ColCACI) (pp. 1-6). IEEE. doi: 10.1109/ColCACI.2019.8781798

[4] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). **Searching for mobilenetv3**. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1314-1324).

[5] Chiu, Y. C., Tsai, C. Y., Ruan, M. D., Shen, G. Y., & Lee, T. T. (2020, August). **Mobilenet-SSDv2: An improved object detection model for embedded systems**. In 2020 International Conference on System Science and Engineering (ICSSE) (pp. 1-5). IEEE.

[6] D. Wu, N. Sharma and M. Blumenstein, "**Recent advances in video-based human action recognition using deep learning: A review,**" 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2865-2872, doi: 10.1109/IJCNN.2017.7966210.

[7] S. Hu, X. Jia and Y. Fu, "**Research on Abnormal Behavior Detection of Online Examination Based on Image Information**," 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2018, pp. 88-91, doi: 10.1109/IHMSC.2018.10127. https://ieeexplore.ieee.org/document/8530188

[8] Dhillon, Anamika, and Gyanendra K. Verma. "**Convolutional neural network: a review of models, methodologies and applications to object detection.**" Progress in Artificial Intelligence 9.2 (2020): 85-112, https://doi.org/10.1007/s13748-019-00203-0

[9] Ghazal, M., Waisi, N., & Abdullah, N. (2020). **The detection of handguns from live-video in real-time based on deep learning**. Telkomnika, 18(6), 3026-3032.

[10] Chiu, Y. C., Tsai, C. Y., Ruan, M. D., Shen, G. Y., & Lee, T. T. (2020, August). Mobilenet-SSDv2: **An improved object detection model for embedded systems**. In 2020 International Conference on System Science and Engineering (ICSSE) (pp. 1-5). IEEE. doi: https://doi.org/10.1109/ICSSE50014.2020.9219319

[11] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). **The pascal visual object classes challenge: A retrospective**. International journal of computer vision, 111(1), 98-136.

[12] Kanimozhi, S., Gayathri, G., & Mala, T. (2019, February). **Multiple Real-time object identification using Single shot Multi-Box detection**. In 2019 International Conference on Computational Intelligence in Data Science (ICCIDS) (pp. 1-5). IEEE. doi: https://doi.org/10.1109/ICCIDS.2019.8862041

[13] Luo, Y., Li, S., Sun, K., Renteria, R., & Choi, K. (2017, November). **Implementation of deep learning neural network for real-time object recognition in OpenCL framework**. In 2017 International SoC Design Conference (ISOCC) (pp. 298-299). IEEE.

[14] Harshal Honmote , Shreyas Gadekar , Pranav Katta , Madhavi Kulkarni, 2022, **Real Time Object Detection and Recognition using MobileNet-SSD with OpenCV**, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 01 (January 2022)

[15] Lin TY. et al. (2014) **Microsoft COCO: Common Objects in Context**. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham.

[16] **Introducing the Next Generation of On-Device Vision Models: MobileNetV3**, Available online: https://ai.googleblog.com/2019/11/introducing-next-generation-on-device.html

[17] Ren, Junda, Yongkun Du, Zhineng Chen, Fen Xiao, Caiyan Jia, and Hongyun Bao. "**An Empirical Study of Object Detectors and Its Verification on the Embedded Object Detection Model Competition**." In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 1-6. IEEE, 2020.

[18] Koonce, Brett. "**MobileNetV3**." In *Convolutional Neural Networks with Swift for Tensorflow*, pp. 125-144. Apress, Berkeley, CA, 2021.

[19] Ning, Chengcheng, Huajun Zhou, Yan Song, and Jinhui Tang. "**Inception single shot multibox detector for object detection**." In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 549-554. IEEE, 2017.

[20] Bodla, Navaneeth, Bharat Singh, Rama Chellappa, and Larry S. Davis. "**Soft-NMS--improving object detection with one line of code**." In *Proceedings of the IEEE international conference on computer vision*, pp. 5561-5569. 2017.

[21] Veit, Andreas, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. "**Coco-text: Dataset and benchmark for text detection and recognition in natural images**." *arXiv preprint arXiv:1601.07140* (2016).

[22] PyCharm. https://www.jetbrains.com/pycharm/

[23] OpenCV. https://opencv.org/

[24] Numpy. https://numpy.org/

[25] SSD: Single Shot Detector for object detection using Multi-Box. https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca

[26] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "**Ssd: Single shot multibox detector**." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.

[27] Vaishali, Shilpi Singh. "**Real-Time Object Detection System using Caffe Model.**" International Research Journal of Engineering and Technology (IRJET) Volume 6 (2019).

[28] NMS. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

[29] Miikkulainen, Risto, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju et al. "**Evolving deep neural networks**." In Artificial intelligence in the age of neural networks and brain computing, pp. 293-312. Academic Press, 2019.

[30] Szegedy, C., Toshev, A., & Erhan, D. (2013). **Deep neural networks for object detection**.

[31] Goodfellow, I., Bengio, Y., & Courville, A. (2016). **Deep learning**. MIT press.

[32] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "**Mobilenets: Efficient convolutional neural networks for mobile vision applications.**" arXiv preprint arXiv:1704.04861 (2017).

[33] Howard, Andrew, Andrey Zhmoginov, Liang-Chieh Chen, Mark Sandler, and Menglong Zhu. "**Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation**." (2018).

[34] Howard, Andrew, and Suyog Gupta. "**Introducing the Next Generation of On-Device Vision Models: MobileNetV3 and MobileNetEdgeTPU**." (2020).

[35] Koonce B. (2021) **MobileNetV3. In: Convolutional Neural Networks with Swift for Tensorflow**. Apress, Berkeley, CA.

[36] MobileNetv3. https://towardsdatascience.com/everything-you-need-to-know-about-mobilenetv3-and-its-comparison-with-previous-versions-a5d5e5a6eeaa

[37] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "**Deep learning.**" nature 521, no. 7553 (2015): 436-444.

APPENDIX

Sample Code

A. Image_main

```
#input taken as image
img = cv2.imread('images/elsa1' + '.jpeg')

#resizing the output player
resize_img = cv2.resize(img, (512, 512))

classNames = []
classFile = 'dataset\\coco.names'

#color mapping randomly.
colors = np.random.uniform(0, 255, size=(len(classNames), 2))
```

```
configPath = 'C:\\Users\\PAROMITA SAHA\\PycharmProjects\\ObjectDetector\\dataset\\ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'C:\\Users\\PAROMITA SAHA\\PycharmProjects\\ObjectDetector\\dataset\\frozen_inference_graph.pb'

net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0 / 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

```
#storing the confidence value and converting into %.
conf = str(round(confidence * 100, 2)) + "%"

#creating the bounding box around the object.
cv2.rectangle(resize_img, box, colors[classId - 1], thickness=4)
#printing the class names(coco_names) inside the bounding box.
cv2.putText(resize_img, classNames[classId-1], (box[0] + 10, box[1] + 30),
cv2.FONT_HERSHEY_COMPLEX, 1, colors[classId-1], 2)
#printing the confidence value inside the bounding box.
cv2.putText(resize_img, conf, (box[0] + 10, box[1] + 70),
cv2.FONT_HERSHEY_COMPLEX, 1, colors[classId-1], 2)

print(classIds, classNames[classId-1], conf)

cv2.imshow("Output_image_detection", resize_img)
```

B. Webcam_main

```
thres = 0.5                  # threshold to detect the objects
nms_thres = 0.2              #Non-Maximum Suppression

#opening of webcam.
cap = cv2.VideoCapture(0)
cap.set(3, 720)                   #width
cap.set(4, 640)                   #height
cap.set(10, 150)                  #brightness

while True:
    success, img = cap.read()
    classIds, confs, bbox = net.detect(img, confThreshold=thres)

    indices = cv2.dnn.NMSBoxes(bbox, confs, thres, nms_thres)
    print(indices)
```

```
#to stop detecting many bounding boxes
for i in range(len(bbox)):
    for i in indices:
        box = bbox[i]
        confidence = confs[i]
        x, y, w, h = box[0], box[1], box[2], box[3]
```

```
#storing the confidence value and converting into %.
conf = str(round(confidence * 100, 2)) + "%"

#creating the bounding box around the object.
cv2.rectangle(img, (x, y), (x+w, h+y), color=(230, 0, 0), thickness=3)
#printing the class names(coco_names) inside the bounding box.
cv2.putText(img, classNames[classIds[i]-1], (box[0] + 10, box[1] + 30),
cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 252), 2)
#printing the confidence value inside the bounding box.
cv2.putText(img, conf, (box[0] + 10, box[1] + 60), cv2.FONT_HERSHEY_COMPLEX, 1,
(0, 0, 0), 2)

print(classIds, classNames[classIds[i] - 1], conf)

#output player pop up window
cv2.imshow("Output_Video_Detection", img)
cv2.waitKey(1)
cv2.destroyAllWindows()
```