Annexure – 1

REAL-TIME OBJECT DETECTION USING DEEP LEARNING

## A Project Report

Submitted in the partial fulfilment for the award of the degree of

# BACHELOR OF ENGINEERING

## IN

**CSE (Hons.) Specialization in Big Data Analytics**

## Submitted by: Tatikonda Chandrashekharachari

## 20BCS4428

**Under the Supervision of:**

**Ms. Tanvi Mam**

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING APEX INSTITUE OF TECHNOLOGY

## CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

**January 2024**

Annexure – 2

# DECLARATION

I, **‘Tatikonda Chandrashekhar (20BCS4428)**, student of **‘Bachelor of Engineering in CSE (Hons.) with Specialization in Big Data Analytics’**, **session: 2020-2024**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled ‘**Real Time Object Detection Using Deep Learning**’ is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

## Tatikonda Chandrashekharachari

## 20BCS4428

**Date: 01/01/2024**

**Place: Chandigarh University**

Annexure – 3

# Table of Contents

|  |  |
| --- | --- |
| Title Page  Declaration of the student Abstract Acknowledgement | Annexure - 1  Annexure - 2  Annexure - 2  Annexure - 1 |
| **1. INTRODUCTION** | **1** |
| 1.1 Problem Definition | 1 |
| 1.2 Project Overview/Specification\*(page-1 and 3) | 2 |
| 1.3 Hardware Specification | 2 |
| 1.4 Software Specification | 2 |
| **2. LITERATURE SURVEY** | **3** |
| 2.1 Existing System | 3 |
| 2.2 Proposed System | 3 |
| 2.3 Literature Review Summary | 3 |
| **3. DESIGN FLOW / PROCESS** | **9** |
| **4. RESULTS ANALYSIS AND VALIDATION** | **42** |
| **5. CONCLUSION AND FUTURE WORK** | **43** |
| **6. REFERENCES** | **45** |

**List of Figures**

1. Figure – 3.1 11

2. Figure – 3.2 13

3. Figure – 3.3 13

4. Figure – 3.4 17

5. Figure – 3.5 18

6. Figure – 3.6 22

7. Figure – 3.7 25

8. Figure – 3.8 27

9. Figure – 3.9 29

10. Figure – 3.10 32

11. Figure – 3.11 33

12. Figure – 3.12 33

13. Figure – 3.13 34

14. Figure – 3.14 34

15. Figure – 3.15 35

16. Figure – 3.16 35

17. Figure – 4.1 40

18. Figure – 4.2 41

# List of Tables

1. Table - 2.1 7

# INTRODUCTION Problem Definition:

While the human eye can instantly and accurately recognize a given image (its content, location, and nearby images) by interacting with it, robotic systems that support computer vision can sometimes be too slow and inaccurate. . Any development in this area will lead to improvements in efficiency and performance, and could pave the way for smarter, human-like systems. Therefore, systems like advanced technology that allow humans to perform tasks with little or no awareness will certainly make our lives much easier.

Identifying each object in an image or scene using computer/software is called object detection. Object detection is one of the most important problems in the field of computer vision or wireless networks. It is the basis for complex vision tasks such as object tracking and scene understanding, and is widely used in wireless networks. The task of object detection is to determine if there is an object belonging to a specified category in the image. If it exists, the next task is to identify its category and location information.

Traditional target detection algorithms are mainly aimed at detecting multiple targets, such as pedestrian detection and infrared target detection. Due to recent advances in deep learning techniques, especially after the emergence of deep convolution neural network (CNN) techniques, object detection algorithms have undergone a revolutionary development. Among these algorithms, the three main methods commonly used in this field are You Only Look Once (YOLO), Single Shot Multiple Box Detector (SSD) and Faster Region CNN (F-RCNN).

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

# PROBLEM FORMULATION

There has been a rapid and successful expansion of computer vision research in recent years. Part of this success can be attributed to the adoption and adaptation of machine learning methods, while others can be attributed to the development of new representations and models for specific computer vision problems, or the development of efficient solutions. Object detection is one area that has made significant progress. There are many objects in this world that humans have identified. So, this one is to make the machines recognize them. The current work provides an overview of object detection.

Object detection, given a set of object classes, consists in determining the location and scale of all object instances, if any, present in an image. Thus, the goal of an object detector is to find all object instances of one or more given object classes regardless of scale, location, pose camera view, partial occlusions, or lighting conditions.

# Project Overview:

However, with the emergence of 5G, the data characteristics of wireless networks such as big data, changing business, data diversification, and uneven temporal and spatial distribution of data pose serious challenges for detection. Targets in real-time environments. Additionally, real-time object detection should be performed on any device and in any environment. To meet the challenge, the project's object detection technology detects objects in real time using a model that can run on any device in any environment.

More specifically, our proposed method applies a convolution neural network to develop a model composed of several layers to classify a given object into several defined classes. Based on recent advancements in deep learning for image processing, the proposed scheme then uses multiple images and detects objects from these images and labels them with their respective class labels.

These images can come from videos fed into our prepared model and the model will be trained until the error rate is reduced to an acceptable level. To accelerate the computational performance of object detection techniques, we use an improved single-shot multi-box detector (SSD) algorithm and a faster region convolution neural network.

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.

# HARDWARE AND SOFTWARE REQUIREMENTS

**Hardware Specifications:**

* + CPU – Core i5 10Gen /Ryzen 5 or above
  + RAM – 8Gb or above
  + ROM - 500Mb or above
  + Web cam (either built-in or external)

# Software Specifications:

* + Python Compilers (Jupyter Notebook, Spyder, etc.)
  + Editors (Notepad, Visual Studio Code, Code Blocks, etc.)
  + Python packages for Image Recognition and Deep Learning

# LITERATURE REVIEW

## Existing System:

Among the many uses of object detection, safety is one of the most important. Of course, there are also many object detection applications using machine learning in the security field. To our knowledge, the subjects we selected for our literature review have not been the subject of publications.

To our knowledge, no one has conducted an in-depth study of the literature on the subject we have chosen. The two studies we found closest to the survey were either specific or covered specific methods rather than analyzing other articles.

## Proposed System:

Use deep learning models to achieve real-time object detection and recognition in dynamic environments from images and video captured by webcams. The main objective is to detect and recognize objects in real time. All things considered, we need rich data.

We have to observe different types of objects that move relative to the camera. This will help us observe and identify different cooperating and interacting objects. In this project, we focused on precision. Use deep learning models to achieve real-time object detection and recognition in images and video captured by webcams in dynamic environments.

The main objective is to detect and recognize objects in real time. All things considered, we need rich data. We have to observe different types of objects that move relative to the camera. This will help us observe and identify different cooperating and interacting objects. In this project, we focused on precision.

## Literature Review Summary:

1. A study on several techniques for object detection and tracking in video surveillance footage was done in one work by Murugan et al. The paper emphasizes how video surveillance has been a technology since the 1950s and reinforces the point made in the Introduction section about how watching security cameras can be exhausting and have a negative impact on a person's mental health. It also demonstrates how, by introducing a

sophisticated surveillance system, automating the procedure was the solution to that tedious chore. The primary goal of this paper is to describe the various techniques for object detection and tracking in videos.

Backdrop subtraction is the first technique described in the study. According to the report, backdrop removal is one of the most popular techniques for detecting moving objects. Background subtraction involves identifying and removing the background in order to show only the pixels of the moving object. The issue with it is that the results of the procedure are impacted if the background is not static and changes as a result of illumination or specific weather conditions. There are other algorithms for background subtraction as well.

1. A study by Flitton et al. compares different 3D interest point descriptors for CT images of baggage at airports. The main idea here is to find interesting things during baggage x-ray inspections. The paper compared five distinct methods: density, density histogram, density gradient histogram, rotation invariant feature transform, and scale invariant feature transform. Although the research was limited to a relatively narrow issue, the report does a good job of assessing each while providing key indicators. Both of the preceding studies do an excellent job of describing the various strategies, but they are too narrow in scope in comparison to our paper. Instead of focusing on machine learning and object detection separately, both publications provide a more in-depth explanation of the object detection techniques. The two articles cover fewer papers than our Systematic Literature Review does because of their narrower scopes.
2. Bezak, P. (September - 2016) suggests a deep learning approach for item recognition in historical architecture images in Trnava. For jobs requiring object recognition, it employs deep learning architectures based on CNN (Convolution Neural Network). Activation functions and a cascade of convolution layers are used to improve architecture. It is critical to determine the number of layers and the number of neurons in each layer. The TRNAVA LeNet 10 model was created and trained for this purpose. This model is based on 460 training images and 140 validation images, which is a 3:1 ratio. The images were color photographs that were 28x28 pixels in size and encoded in jpg format. In

the image of the Trnava historical building, the model correctly identified the correct item. The prediction accuracy of the proposed model increased to 98.88%.

1. Jung, H., Lee, S. et al (Jan – 2015), suggested that instead of using manually produced characteristics, deep learning techniques should be used to recognize face expressions. Convolution neural networks (CNN) and deep neural networks (DNN) are two types of deep networks used to solve recognizing problems. Deep networks were quickly built using deep learning toolkits that support CUDA, such as I and CudaConvnet2. They also used the OpenCV library to develop the Haar-like face detection technique. The photographs were cropped and reduced in size to 64 \* 64. The 327 face photos were then divided into ten groups, one for training and the other nine for testing. The recognition rates for six emotions were high, but the disgust label had a low recognition rate. Because the FER 2013 database contained only 547 training photos for the disgust label. Over-fitting is a possibility with the DNN.
2. Tenguria, R., Parkhedkar et al (April – 2017), Convolution neural networks have been replaced by more precise yet sophisticated approaches that can recognize things in real-time, according to the study. This paper has the potential to make significant advances in object recognition and tagging. However, progress in this area has been relatively slow. It plans to merge the fields of computer vision and robotics, with a focus on the implementation of image description applications on an embedded system platform. Depending on the data set used to train the model, a fixed number of items are allowed in the image. According to Shaoqing Ren et al., the development of the Region Proposal Network (RPN) allows for the network to share the entire image’s convolution characteristics, resulting in nearly free region proposals. In this case, the region suggestion technique directs the algorithm to find objects in the image. Second, the application of this technique in our system makes it computationally efficient and tailored to function on low-powered platforms.
3. Etemad, E., & Gao, Q. (Sep. – 2017): In order to improve the effectiveness of current object recognition algorithms, the research

presents an object localization method that makes use of image edge information as a cue to pinpoint the locations of the objects. The image’s Generic Edge Tokens (GETs) are extracted using the perceptual organization components of human vision. These edge tokens are parsed using the Best First Search method to precisely locate objects, with the detection score provided by the Deep Convolution Neural Network serving as the goal function. When the BFS is applied to the object localization and its search space, the search space is a collection of edge elements whose overlaps with the current candidate object are greater than zero. Real-time testing revealed that the model outperformed the RCNN, and there is still room for improvement by enhancing object localization by combining picture edge, color, and texture information, as well as the image’s learned properties.

1. Mazumdar, M., Sarasvathi, V. and Kumar, A. (Aug. – 2017), suggested a technique for creating an interactive application to identify things from films; upon user input, it is also capable of identifying the specific object now displayed on the screen. A sequential frame extraction technique for films, as well as a deep learning strategy utilizing Convolution and Fully Connected Neural Networks, is used for this challenge, which has a 77% accuracy rate. Even when the item is slightly warped, translated, rotated, or partially obscured from view and can still be easily spotted by humans, computer vision work remains difficult. Using the fact that videos are made up of frames synced with some playback audio, the video can be analyzed in much greater detail by looking at the objects present in the frame images themselves, running the classifier to obtain probabilities for various classes, and then classifying the genre as well as identifying any objects in the video. By adding more datasets and optimizing the hardware setup, this model’s operational accuracy is increased,15nableing faster and more accurate item categorization over a wider variety of classes.
2. Sujana, S. R., Abisheck, S. S., Ahmed, A. T., & Chandran, K. S. (2017), suggests a method for using convolution neural networks and the idea of deep learning to identify things. Using the input video, it generates an output with a collection of recognized objects. The convolution neural network computes a confidence score for each object. It makes use of the

Single Shot Multi-box Detector, which has a high accuracy rate and uses convolution networks to identify multiple items at the same time. It employs Hard Negative Mining and Non-Maximum Suppression to increase the object’s confidence score and produce only one detection for each object, respectively. Thus, combining neural networks with deep residual networks improves the computing efficiency and accuracy of item identification.

YOLO V3 is a detector of objects which makes use of features learned by a deep convolutional neural network for detecting object in real time. It consists of 75 convolutional layers with up-sampling layers and skips connections for the complete image one neural network being applied. Regions of the image are made. Later bounding boxes are displayed along with probabilities. The most noticeable feature of YOLO V3 is that the detections at three different scales can be done with the help of it. But the speed has been traded off for boosts in accuracy in YOLO v3, and it does not perform well with small objects that appear in groups.

Faster R-CNN consists of two networks: a framework for object detection based on these concepts and a region proposal network (RPN) for producing zone suggestions. The main distinction between this approach and Fast R- CNN is that it generates region suggestions via selective search. When RPN shares the majority of its computations with the object identification structure, area recommendations are produced in a lot less time than they would in targeted screening. RPN ranks the area boxes, also referred to as anchors, and recommends the ones that are most likely to contain items. Two rapid RCNN algorithms are used by the Region Proposal Network to create regions and identify objects. The first method uses the suggested regions after making recommendations for them. A limitation of faster R-CNN is that it has a difficult training process and a poor processing speed.

Based on the results obtained from the literature review, the following conclusions are drawn.

* From the performance results of various object detection models on the MS COCO dataset, it can be concluded that SSD and R-FCN models are faster than Faster R-CNN.
* But if accuracy is more important than speed, Faster R-CNN outperforms SSD and R-FCN models.
* Faster R-CNN is the most accurate model when using Inception ResNet, operates at 1 frame per second and meets the minimum requirements to perform real-time object detection and recognition.
* Compared to other object detection models, SSD is faster but has difficulty detecting small objects.
* The speed of Faster R-CNN increases as the number of proposals decreases, which also reduces the accuracy of the model.
* According to Redmond et al, YOLOv3 detects 10 times faster than state- of-the-art methods. Therefore, YOLOv3 and its variant Tiny-YOLOv3 were chosen for the experiments.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **TITLE** | **YEAR** | **AUTHOR** | **TECHNIQUES** |
| 1 | Application of deep learning in object detection | 978-1-5090-5507-  4/17/$31.00 ©2017 IEEE ICIS 2017, May 24-26,  2017, Wuhan, China  2017 | Xinyi Zhou, Wei Gong, WenLong Fu, Fengtong Du | **DATASETS:**   * ImageNet * PASCALVOC * COCO   **METHODOLOGY:**   * R-CNN * SPP-net * Fast R-CNN * Faster R-CNN |
| 2 | An Efficient approach for object detection and object tracking | 2017 Third International Conference on Science Technology Engineering and Management | B. Maga | **MODULE:**   * Kernel Method and Training * Feature method * Template Generation   **TECHNIQUE:**   * Template Matching * Later Based Tracking |
| 3 | Object detection based on deep learning of small samples | 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI) March 2018, Xiamen,  China | Ce Li, Yachao Zhang | **KEY STEPS:**   * Foreground objects extraction * Background selection and fusion processing * Object Sem |
| 4 | A Learning algorithm | 2011,8th International | Chen | **KEY STEPS:** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | for model-based object detection | Conference on Ubiquitous Robots and Ambient Intelligence | Guodong, Zeyang Xia, Rongchuan Sun, Zhenhua Wang, Zhiwu Ren and Lining Sun | * Object detection * Shape Matching * Image Segmentation Shape Fragment   **PROPERTIES:**   * Rotation invariance * Scale invariance * Noise robustness |
| 5 | Object detection and tracking | 2015 INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCDE AND COMMUNICATION NETWORK | K. Rasool Reddy, K. Hari Priya, N. Neelima | **KEY STEPS:**   * Incremental Multiple principle component analysis * Frag Track * HOG - LBP Detector Generative and Discriminative Trackers Semi Supervised Support Vector   Machines |
| 6 | Modelling from an object and multi- object tracking system | 2016, Global Summit on Computers and Information Technology | Afef SALHI, Yacine MORESLY,  Fahmi GHOZZI,  Ameni YENGUI, and Ahmed FAKHFAKH | **KEY STEPS:**   * Block-matching * KLT algorithm (Kanade Lucas Tomasi) * Meanshift algorithm (MA) Camshift Algorithm (CA) |
| 7 | Object detection in sports video | MIPRO 2018, May 21-  25, 2018, Opatija Croatia | M. Buric, M. Pobar, M.IvasicKos | **METHODOLOGY :**   * Mask R-CNN * YOLO object detector Mixture of Gaussians method |
| 8 | Object Tracking Camera | IJSRD - International Journal for Scientific Research & Development| Vol. 3,  Issue 03, 2015 | ISSN  (online): 2321- 0613 | Priyanka Pacharne, Sanket Kotkar, Neha Darekar | **KEY STEPS :**  g  ion Background Subtraction |
| 9 | A Survey on Object Tracking in Video | 2017, IJSRD -  International Journal for Scientific Research & Development | Snehlata Raisagar, Ashish Tiwari | **KEY STEPS :**   * Video Sequence * Object Detection * Object Recognition Tracking |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | Detection and Tracking of Moving Object in Video - A Survey | 2016, || National Conference on Technological Advancement and Automatization in Engineering | Dhaval Deshpande, Nikhil Aatkare, Prof.Reena Somani | **STATISTICAL METHODS :**   * Background Subtraction * Temporal Differencing Correspondence Based Matching * Algorithm Kernel Tracking |

**Table – 2.1**

# DESIGN FLOW/PROCESS

**Deep learning:** A technique used in artificial intelligence (AI) called deep learning teaches computers to interpret data in a manner modelled after the human brain. Deep learning models can identify intricate patterns in images, text, audio, and other types of data to generate precise analyses and forecasts.

**Convolution neural networks:** A CNN is a particular type of network design for deep learning algorithms that is utilised for tasks like image recognition and pixel data processing. Although there are other kinds of neural networks in deep learning, CNNs are the preferred network architecture for identifying and recognising objects.

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single 26 perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply.

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data.

Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28 × 28. With this dataset a single neuron in the first hidden layer will contain 784 weights (28×28×1 where 1 bare in mind

that MNIST is normalised to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64 × 64, the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalised MNIST digits, then you will understand the drawbacks of using such models. Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non- linearity.

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single 26 perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply.

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data.

Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28 × 28. With this dataset a single neuron in the first hidden layer will contain 784 weights (28×28×1 where 1 bare in mind that MNIST is normalised to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64 × 64, the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to

classify colour normalised MNIST digits, then you will understand the drawbacks of using such models.

**Imutils:** Basic image processing tasks including translation, rotation, scaling, skeletonization, Matplotlib picture presentation, sorting contours, edge detection, and more are made simpler with OpenCV and both Python 2.7 and Python 3.

**Numpy:** Numeric Python, sometimes known as NumPy, is a Python library for computing and processing multidimensional and linear array elements. Numerous robust data structures, including multi-dimensional arrays and matrices, are available in NumPy. The best calculations for arrays and matrices are done using these data structures.

**Argparse:** The command line parameters can be handled easily with the help of the Argparse module. It shows the program's general usage, assistance, and errors. The Argument Parses object's parameter is usage. The string it displays will contain details about the programme and its arguments.

**OpenCV:** A machine learning and computer vision software library is available for free under the name OpenCV. Open-Source Computer Vision Library is how OpenCV is formally referred to. It was developed to speed up the incorporation of machine perception into consumer goods and to offer a standardised infrastructure for computer vision applications. Companies can easily use and modify the code thanks to OpenCV, a BSD-licensed programme.

A Python package called OpenCV makes it possible to carry out image processing and computer vision tasks. It offers a variety of capabilities, such as tracking, face recognition, and object detection.

**Deep Learning:** Deep learning is a machine learning technique. It teaches a computer to filter inputs through layers to learn how to predict and classify information. Observations can be in the form of images, text, or sound. The inspiration for deep learning is the way that the human brain filters information. Its purpose is to mimic how the human brain works to create some real magic.

In the human brain, there are about 100 billion neurons. Each neuron connects to about 100,000 of its neighbors. We’re kind of recreating that, but in a way and at a level that works for machines. In our brains, a neuron has a body, dendrites, and an axon. The signal from one neuron travels down the axon and transfers to the dendrites of the next neuron. That connection where the signal

passes is called a synapse. Neurons by themselves are kind of useless. But when you have lots of them, they work together to create some serious magic.

That’s the idea behind a deep learning algorithm! You get input from observation, and you put your input into one layer. That layer creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal! The neuron (node) gets a signal or signals (input values), which pass through the neuron. That neuron delivers the output signal.

## Single Shot Multi Box detector:

**Single Shot:** this means that the tasks of object localization and classification are done in a *single forward pass* of the network

**Multi Box:** this is the name of a technique for bounding box regression developed by Szegedy et al.

**Detector:** The network is an object detector that also classifies those detected objects.

Real-time object detection is a feature of SSD. Faster R-CNN uses boundary boxes that are generated by a region proposal network to classify things. The entire process moves at a speed of 7 frames per second, yet it is thought to be state-of-the-art in precision. Below the requirements for real-time processing. By doing away with the necessity for the regional proposal network, SSD accelerates the process. SSD implements a few improvements, such as multi-scale features and default boxes, to make up for the loss in accuracy. These enhancements enable SSD to match the Faster R-CNN's accuracy while employing images of lesser quality, significantly increasing speed. The following comparison shows that it surpasses the accuracy of the Faster R-CNN and even achieves real-time processing performance.

A single-stage object detection technique called SSD discretizes the output space of bounding boxes into a collection of default boxes at various aspect ratios and scales for each feature map position. The network creates scores at prediction time for the existence of each object type in each default box and modifies the box to better fit the shape of the object. In order to naturally manage objects of diverse sizes, the network also incorporates predictions from numerous feature maps with different resolutions.

Eliminating bounding box proposals and the ensuing pixel or feature resampling stage results in a fundamental gain in speed. The use of separate predictors (filters) for different aspect ratio detections, the use of small convolutional filters to predict object categories and offsets in bounding box locations, and the application of these filters to multiple feature maps from a network's later stages to perform detection at various scales are improvements over competing single-stage methods.

An SSD consists of two parts: an SSD head and a backbone model. A pre- trained image classification network serves as the backbone model's feature extractor in most cases. Usually, a network like ResNet trained on ImageNet has been modified by removing the final fully linked classification layer. Thus, what is left is a deep neural network that can preserve the spatial structure of the image at a lesser resolution while still extracting semantic meaning from the input image. The backbone of ResNet34 produces 256 7x7 feature maps for an input image. Later on, we will define features and feature maps.

The suggested method makes advantage of an upgraded SSD algorithm for faster real-time detection with increased precision. However, because it ignores the background from outside the boxes, the SSD technique is not suitable for detecting small objects. The suggested technique employs depth-wise separable convolution and spatial separable convolutions in their convolutional layers to address this problem. In particular, our suggested method combines a multilayer convolutional neural network with a new design. There are two phases to the algorithm. By utilising a resolution multiplier, it first decreases the extraction of

spatial dimensions from feature maps. Second, it is built with the use of tiny convolutional filters that apply the best aspect ratio values for object detection. During training, the main goal is to achieve a high confidence score.

A region proposal network is used by faster R-CNN to construct boundary boxes, which are subsequently used to categorise objects. The entire process operates at 7 frames per second, which is significantly below the criteria of real-time processing even though it is thought to be cutting-edge in terms of precision. By eliminating the need for the area proposal network, SSD speeds up the procedure. To compensate for the accuracy decline, SSD adds a few enhancements including default boxes and multi-scale functionality. With the help of these improvements, SSD can now equal the accuracy of the Faster R-CNN while working with pictures of lesser quality, which accelerates the process.

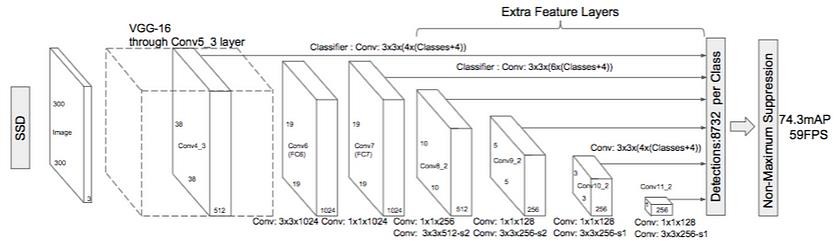
Single Shot Detector is a great deal quicker and accurate than previous approaches. Using feature maps of various dimensions, we generate forecasts on various scales, and then, in order to achieve high accuracy, we separate the forecasts based on ratio of aspect.

High accuracy is achieved even with input photos of low quality because to these properties.

Other algorithms frequently employ the object proposal methodology, in which they devise a mechanism to divide the image into segments and provide suggestions about where those segments may potentially be objects.

These algorithms forfeit accuracy. A notion known as "ground truth" separates actual or empirical evidence from assumed evidence. If certain boxes are absent, we cannot just train the algorithm; we must first identify them throughout the training process.

**Architecture**



**Figure - 3.1**

The architecture of SSD is based on the time-tested VGG-16 architecture, as seen in the diagram above, but does away with the completely connected layers. VGG-16 was chosen as the basic network due to its good performance in jobs requiring the categorization of high-quality images and its widespread application in issues where transfer learning can aid to improve outcomes. A number of auxiliary convolutional layers (starting with conv6) were added in place of the initial VGG fully connected layers, allowing for the extraction of features at various scales and a gradually smaller input size for each succeeding layers.

The architecture of SSD (Single Shot MultiBox Detector) in deep learning is a network composed of several layers that perform object detection. Here's a high-level overview of the SSD architecture:

1. Base Convolutional Layers: The SSD network typically starts with a base convolutional network, such as VGG or ResNet, which is pre-trained on a large-scale image classification task like ImageNet. These layers serve as feature extractors and capture high-level representations from input images.
2. Multi-scale Feature Maps: SSD utilizes feature maps at different scales to detect objects of various sizes. Several convolutional layers are added on top of the base network to generate feature maps at different resolutions. These feature maps capture both low-level and high-level information.
3. Convolutional Predictor Layers: At each feature map scale, a set of convolutional layers is added to predict the presence of objects at different locations and scales. These layers use a combination of

convolutional filters with different kernel sizes to detect objects with varying sizes.

1. Default Boxes/Anchors: SSD uses a predefined set of default boxes, also known as anchors, at each feature map scale. These default boxes are responsible for capturing objects of different aspect ratios and sizes. The number of default boxes per location may vary across feature maps to handle objects at different scales.
2. Confidence Scores and Localization: The convolutional predictor layers produce two sets of outputs for each default box: confidence scores and localization offsets. The confidence scores represent the probability of an object belonging to different classes, while the localization offsets refine the position and size of the default boxes to better align with the ground truth objects.
3. Non-maximum Suppression (NMS): To eliminate redundant detections, non-maximum suppression is applied to the predicted bounding boxes based on their confidence scores. It keeps the most confident detection for each object and removes overlapping or redundant detections.
4. Loss Function: The SSD network is trained using a combination of classification loss and localization loss. The classification loss measures the accuracy of predicted class probabilities, while the localization loss computes the discrepancy between predicted bounding boxes and ground truth boxes.

Overall, the SSD architecture enables efficient and accurate object detection by leveraging multi-scale feature maps and predefined default boxes. It is widely used in applications requiring real-time object detection, such as autonomous driving, video surveillance, and robotics.

In order to interpret the role of SSD algorithm, we first formally denote the following concepts.

Single shot: This means that the tasks of the thing localization and classification are exhausted one passing play of the network.

Multi-box: Ground truth box and predicted box are the boxes in multi-box. This is introduced by Szegedy.

Detector: The network is an associate degree object detector that conjointly classifies those detected objects.

Default the size of the boxes: The selection of boxes is based on the minimum value of convolution layer and maximum values of change in intensity. The first algorithm represents the procedure of producing specified feature maps F(m).

Truth boxes: After finding the size of boxes, the next phase is matching of the boxes with the corresponding truth boxes. A specific given picture to identify the truth boxes is explained in the second algorithm.

Loss function: The loss function is unbelievably simple, and it is a methodology of evaluating how well your role models your dataset. If your predictions are entirely of your loss function, it can operate next range. If the output range is less, it means that the model is good. The main objective is to minimize loss function. The loss function is also depending upon the sum of weighted localization and classification loss functions.

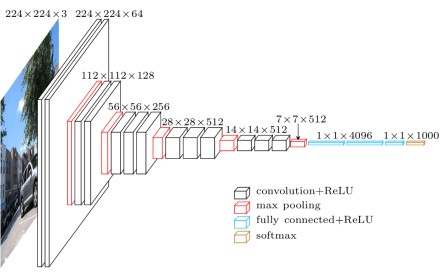
When a color image is fed into the input layer, SSD does the following.

Step 1: Image is passed through large number of convolutional layers extracting feature maps at different points.

Step 2: Every location in each of those feature maps uses a 4x4 filter to judge a tiny low default box.

Step 3: Predict the bounding box offset for each box. Step 4: Predict the class probabilities for each box.

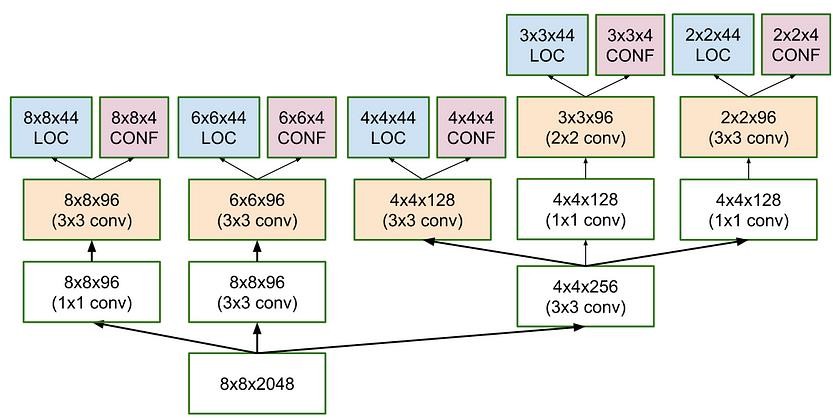
Step 5: Based on IOU, the truth boxes are matched with the predicted boxes. Step6: Instead of exploiting all the negative examples, the result exploits the best-assured loss for every default box.



**Figure - 3.2**

**MultiBox**

Szegedy's work on MultiBox, a technique for quick class-independent bounding box coordinate proposals, served as an inspiration for SSD's bounding box regression algorithm. It's interesting to note that a convolutional network modelled after Inception is employed in the MultiBox research. The 1x1 convolutions you can see below aid in dimensionality reduction since they reduce the number of dimensions while maintaining the same "width" and "height".



**Figure - 3.3**

Additionally, MultiBox loss function combined two essential elements that found their way into SSD:

**Confidence Loss:** This gauges the network's trust in the computed bounding box's objectivity. To compute this loss, categorical cross-entropy is employed.

**Location Loss:** It is a metric used to express how far the network's projected bounding boxes are from the training set's actual ones. Here, L2-Norm is applied.

The equation for the loss, which expresses how far off our prediction "landed," is as follows. I won't go too mathematical here; read the paper if you're inquisitive and want a more exact notation:

***Multibox\_Loss = Confidence\_Loss + Alpha \* Location\_Loss***

We can balance the contribution of the location loss with the aid of the alpha term. As is customary in deep learning, the objective is to identify the parameter values that minimise the loss function and increase the accuracy of our predictions.

In the context of object detection algorithms like SSD (Single Shot MultiBox Detector), MultiBox priors and IoU (Intersection over Union) are key concepts used for generating and evaluating bounding box predictions.

## MultiBox Priors (Default Boxes/Anchors):

MultiBox priors, also known as default boxes or anchors, are a set of predefined bounding boxes that are placed at various positions and scales on feature maps. These priors act as reference boxes for predicting object locations and sizes. Each MultiBox prior is associated with a specific aspect ratio and scale.

By placing default boxes at multiple positions and scales on different feature maps, SSD can handle objects of various sizes. The number of default boxes at each position may differ depending on the network architecture and feature map resolution. These default boxes serve as templates for predicting the locations and sizes of objects during inference.

1. Generation of MultiBox Priors: MultiBox priors are typically generated by considering a set of predefined aspect ratios and scales. For each location on a feature map, multiple default boxes are created by combining different aspect ratios and scales. The aspect ratios control the width-to-height ratio of the boxes, while the scales determine their overall size.
2. Handling Objects at Different Scales: By using default boxes with different scales, the SSD network can detect objects of various sizes. Smaller default boxes are suitable for detecting small objects, while larger boxes are more appropriate for larger objects. The combination of multiple scales allows the network to handle objects at different scales effectively.
3. Default Box Variations: Some SSD implementations use variations of default boxes, such as allowing the aspect ratios and scales to be learned during training instead of being fixed. This allows the network to adapt to the specific characteristics of the dataset.

## Intersection over Union (IoU):

Intersection over Union (IoU) is a measure of overlap between two bounding boxes. It is used to evaluate the accuracy of predicted bounding boxes in comparison to ground truth boxes.

IoU is calculated by dividing the area of intersection between two bounding boxes by the area of their union. It quantifies the extent to which the predicted box aligns with the ground truth box. A higher IoU indicates a better alignment between the predicted box and the ground truth box.

During the training phase of object detection algorithms like SSD, the predicted bounding boxes are compared with the ground truth boxes using IoU. If the IoU between a predicted box and a ground truth box exceeds a predefined threshold (e.g., 0.5), the predicted box is considered a positive detection and contributes to the loss computation. If the IoU falls below the threshold, the predicted box is considered a negative detection and does not contribute to the loss.

IoU is also used during the post-processing step, where non-maximum suppression (NMS) is applied to remove redundant bounding box predictions. NMS selects the bounding box with the highest confidence score among a group of overlapping boxes based on their IoU values.

Overall, MultiBox priors provide a set of reference bounding boxes for object detection, and IoU is a measure used to evaluate the accuracy of predicted boxes relative to ground truth boxes.

Contrary to what I previously indicated, the rationale underlying the production of bounding boxes is considerably more intricate. But do not worry; it is still attainable.

Priors, also known as anchors in Faster-R-CNN parlance, are pre-computed, fixed-size bounding boxes that are constructed by the researchers in MultiBox and closely resemble the distribution of the initial ground truth boxes. In actuality, such priors are chosen so that their Intersection over Union ratio (also known as IoU and occasionally as the Jaccard index) is greater than 0.5. As you can see from the figure below, an IoU of 0.5 still falls short of ideal, but it does give the bounding box regression technique a firm starting point. This is a much better approach than having the predictions begin with random coordinates! MultiBox attempts to regress towards the true bounding boxes by starting with the priors as predictions.

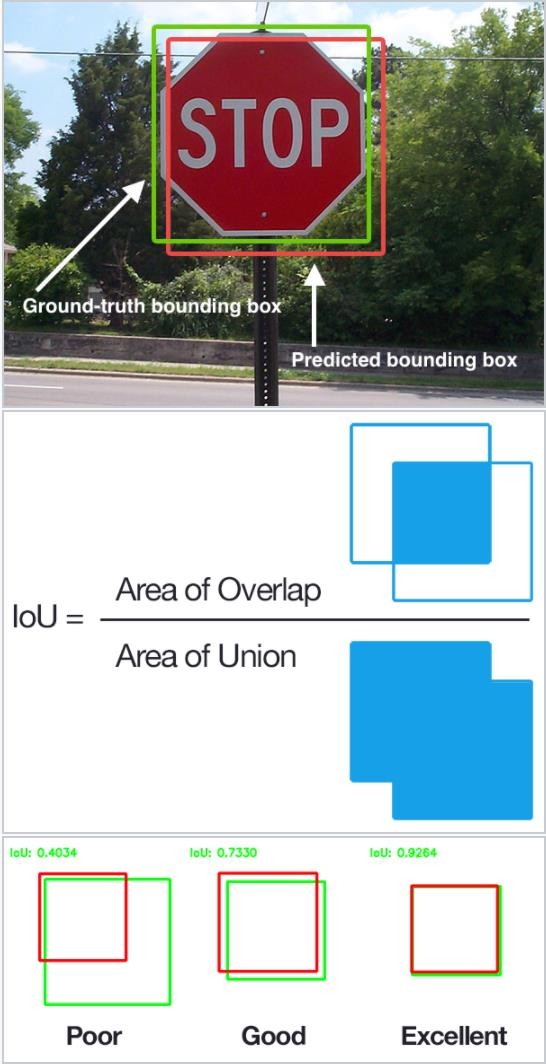
1. Calculation of IoU: IoU is computed by dividing the area of overlap between two bounding boxes by the area of their union. The formula for IoU is:

*IoU = (Area of Overlap) / (Area of Union)*

The area of overlap is the intersection between the two boxes, while the area of union represents the total area covered by both boxes.

1. IoU Threshold for Positive/Negative Detection: During training, a predefined IoU threshold is used to determine whether a predicted bounding box is considered a positive or negative detection. If the IoU between a predicted box and a ground truth box exceeds the threshold, it is classified as a positive detection. Otherwise, it is treated as a negative detection. The threshold value, often set to 0.5, can be adjusted depending on the specific requirements of the application.
2. Non-Maximum Suppression (NMS) and IoU: IoU plays a crucial role in the post-processing step of object detection, particularly in non-maximum suppression (NMS). NMS is applied to eliminate redundant bounding box predictions and select the most confident and accurate detections. It compares the IoU values between different predicted boxes and selects the one with the highest confidence score among the overlapping boxes.

IoU serves as a fundamental measure for evaluating the accuracy and quality of object detections. It helps in training the model by assigning positive or negative labels based on the degree of overlap with ground truth boxes. Additionally, it aids in post-processing steps like NMS to filter out redundant detections and select the most accurate bounding boxes.



**Figure - 3.4**

The resulting architecture, which has 11 priors per feature map cell (8x8, 6x6, 4x4, 3x3, and 2x2) and just one on the 1x1 feature map, yields a total of 1420 priors per image and allows for robust coverage of input images at various scales, allowing for the detection of objects of different sizes.

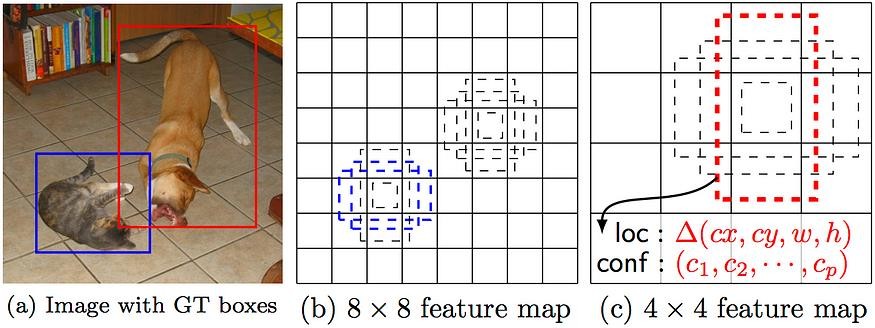
MultiBox only keeps the top K predictions that have the fewest location (LOC) and confidence (CONF) losses at the end.

## SSD Improvements :

A few modifications were made back on SSD to improve the network's ability to locate and categorise things.

Fixed Priors: Unlike MultiBox, each feature map cell has a set of standard bounding boxes with varying sizes and aspect ratios associated with it. In contrast to MultiBox, where they were selected because their IoU with respect to the ground truth was greater than 0.5, these priors were manually (but carefully) chosen. As a result, SSD should theoretically be able to generalise to any form of input without the need for a prior generation pre-training step.

For instance, if we set up 2 diagonally opposed points (x1, y1) and (x2, y2) for each b default bounding box per feature map cell and c classes to classify, SSD would calculate f \* b \* (4 + c) values for this feature map. This feature map has a size of f = m \* n.



**Figure - 3.5**

**Location Loss:** SSD computes the location loss using smooth L1-Norm. Since it does not attempt to be "pixel perfect" in its bounding box prediction—a change of a few pixels would hardly be noticed for most of us—it is nonetheless quite successful even though it is not as precise as L2-Norm.

**Classification:** SSD performs object classification, whereas MultiBox does not. As a result, a set of c class predictions are produced for each predicted bounding box, accounting for every potential class in the dataset.

# TRAINING & RUNNING SSD

Training an SSD (Single Shot MultiBox Detector) involves several steps, including preparing the dataset, configuring the network, defining the loss function, optimizing the model, and fine-tuning if necessary. Here's an overview of the training process for SSD and the datasets commonly used:

1. Dataset Preparation:
   * Annotated Dataset: You need a dataset with annotated images where each object of interest is labeled with its bounding box coordinates and corresponding class labels. Common datasets for object detection include COCO (Common Objects in Context), Pascal VOC (Visual Object Classes), and Open Images.
   * Data Augmentation: To enhance model generalization and increase the diversity of training samples, data augmentation techniques such as random scaling, cropping, flipping, rotation, and color jittering can be applied to the dataset.
2. Network Configuration:
   * Base Network: Select a base convolutional network architecture, such as VGG (Visual Geometry Group), ResNet (Residual Network), or MobileNet, which will serve as the feature extractor. The network should be pre-trained on a large- scale image classification task like ImageNet.
   * Feature Pyramid: Modify the base network to create a feature pyramid by adding extra convolutional layers on top of the base network. These layers extract features at different scales to handle objects of varying sizes.
3. Default Boxes (Anchors):
   * Define the aspect ratios and scales for the default boxes based on the characteristics of the objects in your dataset. These default boxes act as priors and serve as references for predicting object locations and sizes.
   * Assign Ground Truth Boxes: Match the ground truth boxes from the annotated dataset to the default boxes based on their IoU. Determine positive and negative matches, and assign class labels and localization offsets accordingly.
4. Loss Function:
   * The SSD loss function consists of two components: classification loss and localization loss.
   * Classification Loss: Typically, the SoftMax loss or focal loss is used to optimize the predicted class probabilities for each default box.
   * Localization Loss: Smooth L1 loss or IoU loss is used to optimize the predicted bounding box coordinates and refine the position and size of the default boxes.
5. Optimization:
   * Optimize the network using backpropagation and gradient descent techniques. Common optimization algorithms include Stochastic Gradient Descent (SGD) or adaptive methods like Adam or RMSprop.
   * Learning Rate Schedule: Utilize a learning rate schedule, such as decreasing the learning rate over time or employing learning rate warm-up techniques, to stabilize and improve training.
6. Fine-tuning (Optional):
   * After the initial training, you can perform fine-tuning by training the SSD on your specific dataset to improve performance on your target objects or domain. This step can be beneficial when the pre-trained model does not cover the specific objects or environments you are interested in.

It's important to note that the specific implementation details and choices may vary depending on the framework or library you are using (such as TensorFlow, PyTorch, or Keras) and the SSD variant you are employing (SSD300, SSD512, etc.). These steps provide a general outline of the training process for SSD in object detection.

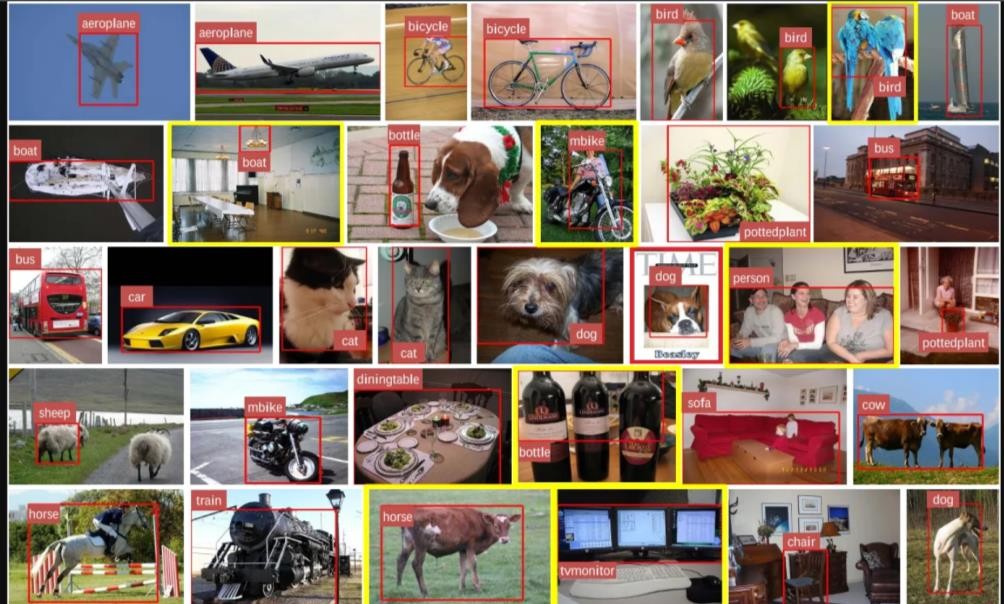
## Datasets

You will require training and test datasets with assigned class labels (just one per bounding box) and ground truth bounding boxes. The COCO and VOC datasets from Pascal are fantastic places to start.

Datasets play a crucial role in training and evaluating object detection models like SSD (Single Shot MultiBox Detector). Datasets for SSD should contain annotated images with bounding box coordinates and corresponding class labels. Here are some commonly used datasets for training and evaluating SSD:

1. COCO (Common Objects in Context):
   * COCO is a widely used large-scale dataset for object detection, segmentation, and captioning.
   * It contains over 200,000 labelled images with more than 80 object categories.
   * COCO provides a diverse range of object instances, occlusions, and complex scenes.
   * The dataset is split into training, validation, and test sets.
2. Pascal VOC (Visual Object Classes):
   * Pascal VOC is a popular dataset for object detection, recognition, and segmentation tasks.
   * It consists of images from 20 different object categories, including animals, vehicles, and common objects.
   * The dataset offers annotations for bounding boxes, object labels, and segmentation masks.
   * Pascal VOC provides standardized evaluation protocols, including predefined train-val splits.
3. Open Images:
   * Open Images is a large-scale dataset created by Google that features diverse and challenging object detection scenarios.
   * It contains over 9 million images with annotations for 600 object classes.
   * The dataset offers a wide variety of object instances, including fine- grained categories.
   * Open Images is well-suited for training models on a large-scale and handling complex real-world scenarios.
4. ImageNet:
   * Although ImageNet is primarily an image classification dataset, its vast collection of labelled images can be used to pre-train the base network of SSD.
   * ImageNet contains millions of images across thousands of object categories.
   * Pre-training on ImageNet enables the base network to learn generic features and improves the performance of SSD on specific object detection tasks.

These datasets are commonly used for training and evaluating SSD models. However, it's worth noting that depending on your specific application or domain, you may need to collect and annotate your own dataset or explore other specialized datasets that cater to your needs. Additionally, it's important to ensure that the dataset you choose aligns with your target objects, environments, and performance requirements.



**Figure - 3.6**

## Default Bounding Boxes

Default bounding boxes, also known as default boxes or anchors, are an essential component of object detection algorithms like SSD (Single Shot MultiBox Detector). These default boxes serve as reference templates for predicting the locations and sizes of objects within an image.

Here are some key points about default bounding boxes:

* Purpose: The primary purpose of default bounding boxes is to provide a set of prior knowledge about the expected locations and scales of objects in an image. They act as a starting point for the network to make predictions.
* Spatial Coverage: Default boxes are placed at different positions and scales across multiple feature maps of the SSD network. They cover various areas of the image, enabling the model to handle objects of different sizes and aspect ratios.
* Aspect Ratios and Scales: Default boxes are defined based on a range of aspect ratios and scales. The aspect ratio determines the width-to-height ratio of the boxes, allowing them to capture objects with different proportions. The scale determines the overall size of the boxes, enabling detection at various object sizes.
* Generation Process: Default boxes are typically generated by considering a predefined set of aspect ratios and scales. At each position in the feature maps, a set of default boxes is created by combining different aspect ratios and scales. The number of default boxes at each position may vary, depending on the architecture and configuration of the SSD network.
* Matching Ground Truth: During training, the default boxes are matched with the ground truth bounding boxes from the annotated dataset. This matching process involves calculating the Intersection over Union (IoU) between the default boxes and the ground truth boxes. The default boxes with the highest IoU overlap are assigned as positive matches for the corresponding object class.
* Localization Predictions: SSD predicts both the class labels and the offsets for each default bounding box. The localization offsets adjust the position and size of the default boxes to align them with the ground truth objects more accurately. These offsets help refine the locations and dimensions of the predicted bounding boxes.

By using default bounding boxes, SSD avoids exhaustive sliding window searches across the entire image and focuses on predicting objects at specific locations and scales. The default boxes provide a flexible and efficient framework for detecting objects of various sizes and aspect ratios within an image.

To ensure that the majority of objects could be caught, it is advised to define a diversified collection of default bounding boxes, of different scales and aspect ratios. Each feature map cell in the SSD article includes about 6 bounding boxes.

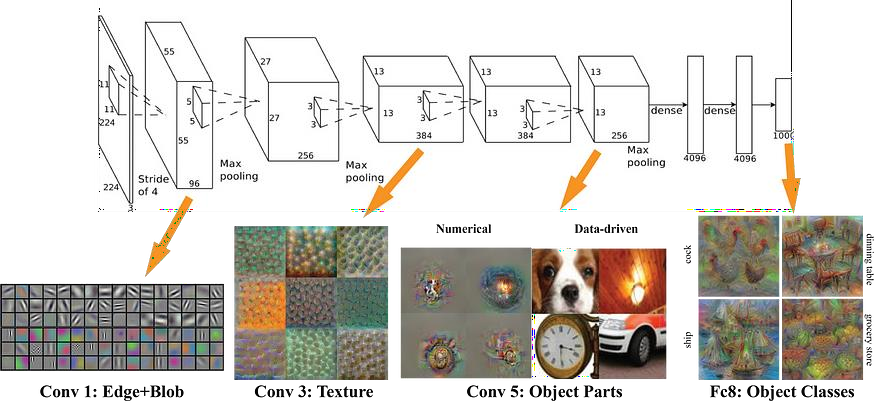
## Feature Maps

Feature maps are intermediate representations extracted from the input image by convolutional layers in a deep neural network, such as SSD (Single Shot MultiBox Detector). These feature maps capture different levels of spatial and semantic information, enabling the network to detect objects at various scales and abstraction levels. Here's some more information about feature maps:

* Extraction of Feature Maps: Feature maps are generated by passing the input image through the convolutional layers of the network. Each convolutional layer applies a set of learnable filters to the input image, performing local feature extraction. The output of each convolutional layer is a feature map.
* Spatial Hierarchy: In deep neural networks, the convolutional layers are typically arranged in a hierarchical manner. As the network gets deeper, the size of the feature maps decreases while the receptive field increases. This hierarchy allows the network to capture both local and global contextual information.
* Semantic Representation: Feature maps encode higher-level semantic information as the network goes deeper. Early layers capture low-level features like edges, textures, and colours, while deeper layers capture more complex and abstract features related to objects and their arrangements.
* Multi-scale Feature Maps: In object detection algorithms like SSD, feature maps at multiple scales are used to handle objects of different sizes. These feature maps are obtained by introducing additional convolutional layers on top of the base network. Each feature map is associated with a specific scale and captures features at a different level of detail.
* Spatial Resolution: Feature maps have reduced spatial resolution compared to the input image. This reduction is typically achieved by using pooling or strided convolutions, which reduce the size of the feature maps while increasing their receptive fields. The reduced resolution helps to reduce computation and focuses on capturing more abstract and higher-level features.
* Object Localization: Feature maps are crucial for object localization in SSD. Each feature map position is associated with a set of default bounding boxes (anchors). The network predicts the offsets and confidences for these default boxes at each feature map location. The combination of feature maps at different scales allows SSD to detect objects across a wide range of sizes and aspect ratios.
* Feature Fusion: In some architectures, such as Feature Pyramid Networks (FPN), feature maps from multiple resolutions are fused together to create a feature pyramid. This fusion helps capture multi-scale information and enhance object detection performance.

Overall, feature maps are intermediate representations in deep neural networks that encode spatial and semantic information. They play a crucial role in object detection algorithms like SSD by providing a hierarchical and multi-scale representation of the input image, enabling the network to detect objects with varying scales and complexities.

Running MultiBox on multiple feature maps increases the likelihood that any object, big or small, will eventually be detected, localised, and correctly classified. Features maps, or the results of the convolutional blocks, are a representation of the dominant features of the image at different scales. How the network "sees" a particular image across its feature maps is depicted in the graphic below:



## Hard Negative Mining

**Figure - 3.7**

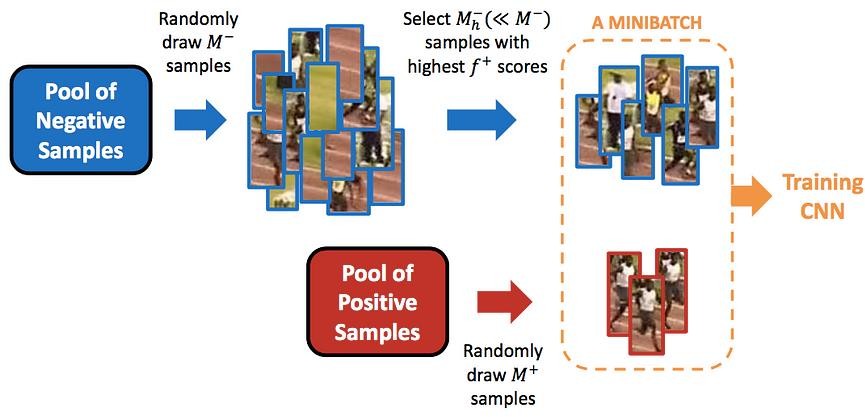
Hard negative mining is a technique commonly used in object detection algorithms, including SSD (Single Shot MultiBox Detector), to improve the training process by focusing on challenging negative samples. It involves selecting hard negative examples during the training phase to enhance the

model's ability to discriminate between positive and negative detections. Here's an overview of how hard negative mining works:

1. Positive and Negative Samples:
   * During training, the dataset is annotated with bounding boxes and class labels.
   * Positive samples refer to instances of objects that are present in the image and have corresponding ground truth annotations.
   * Negative samples, on the other hand, represent areas of the image that do not contain any object of interest.
2. Initial Training:
   * In the initial stages of training an object detection model like SSD, the number of negative samples typically exceeds the number of positive samples by a large margin.
   * This imbalance can lead to a bias towards negative samples and hinder the model's ability to learn accurate object detection.
3. Hard Negative Mining:
   * Hard negative mining aims to address the imbalance between positive and negative samples by focusing on challenging negative samples.
   * During training, the model produces a large number of candidate bounding box predictions that need to be evaluated.
   * Hard negative mining involves selecting negative samples that are difficult to classify correctly and have high confidence scores.
   * The model's confidence scores for negative samples are computed based on their predicted class probabilities or the SoftMax scores.
4. Mining Criteria:
   * Hard negative mining is typically performed based on certain criteria, such as selecting the samples with the highest confidence scores or those that have the highest loss values.
   * The specific criteria depend on the implementation and the loss function used.
5. Training with Hard Negative Samples:
   * The selected hard negative samples are then used to update the model's parameters during the backpropagation process.
   * By focusing on challenging negative samples, the model learns to improve its ability to discriminate between positive and negative detections, leading to better performance.
6. Iterative Process:
   * Hard negative mining is often an iterative process performed during each training epoch or mini-batch.
   * As the model gets trained, it becomes better at handling easy negative samples, making it necessary to focus on harder negatives to continue improving the model's performance.

Hard negative mining helps address the class imbalance issue and ensures that the model dedicates more attention to challenging negative samples during training. By selectively mining hard negatives, the model can better learn to distinguish between positive and negative detections, leading to improved accuracy and reduced false positives.

We risk having an excessive number of negative examples in our training set since the majority of the bounding boxes will have low IoU during training and will consequently be interpreted as negative training examples. Therefore, it is advised to maintain a ratio of negative to positive examples of about 3:1 rather than using only negative predictions. You must maintain negative samples since the network needs to understand what makes an inaccurate detection and be explicitly told what does not.



**Figure- 3.8**

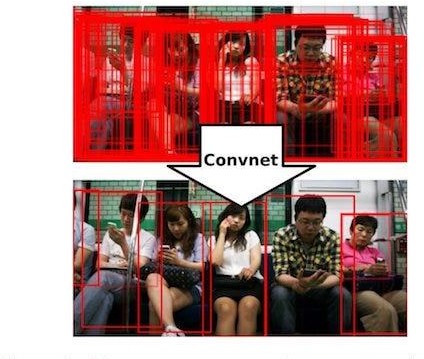
## Non-Maximum Suppression (NMS)

Non-maximum suppression (NMS) is a post-processing technique commonly used in object detection algorithms, including SSD (Single Shot MultiBox Detector), to eliminate redundant bounding box predictions and select the most confident and accurate detections. NMS helps ensure that only the most relevant and non-overlapping bounding boxes are retained. Here's how non-maximum suppression works:

1. Bounding Box Predictions:
   * During the inference phase, an object detection model like SSD generates multiple bounding box predictions for various objects within an image.
   * Each bounding box prediction consists of coordinates (x, y) for the top- left corner and (width, height) representing the size of the box. Additionally, it is associated with a class label and a confidence score indicating the model's confidence in the prediction.
2. Sorting by Confidence Scores:
   * The first step in non-maximum suppression is to sort the bounding box predictions based on their confidence scores.
   * The predictions with higher confidence scores are considered more likely to be accurate detections.
3. Iterative Suppression:
   * Starting with the prediction that has the highest confidence score, non- maximum suppression iterates through the sorted list of predictions.
4. Overlap Calculation:
   * For each prediction, the overlap or Intersection over Union (IoU) is calculated with all the remaining predictions in the list.
   * IoU is the ratio of the intersection area between two bounding boxes to their union area. It measures the extent of overlap between two boxes.
5. Thresholding:
   * A predefined IoU threshold is set to determine the level of overlap at which two bounding boxes are considered redundant.
   * If the IoU between the current prediction and any of the remaining predictions exceeds the threshold, the latter bounding box is suppressed, as it is considered a duplicate or highly overlapping detection.
6. Non-Maximum Suppression:
   * Bounding box predictions that survive the thresholding process are considered as non-maximum detections.
   * These non-maximum detections are retained as the final set of bounding box predictions for objects of interest.
7. Per-Class NMS:
   * In multi-class object detection scenarios, non-maximum suppression is typically applied separately to each class's bounding box predictions.
   * This ensures that the suppression process is class-specific and avoids suppressing relevant detections from different classes.

Non-maximum suppression significantly reduces the number of redundant and overlapping bounding box predictions, leading to a more concise and accurate set of detections. By selecting the most confident and non-overlapping detections, NMS helps improve the precision and clarity of object detection results.

Given the large number of boxes produced during a forward pass of SSD at inference time, it is crucial to prune the majority of the bounding box using a method known as non-maximum suppression. Only the top N predictions are kept, and boxes with a confidence loss threshold less than ct (for example, 0.01) and IoU less than It (for example, 0.45) are kept. This makes sure that the network only keeps the most likely predictions and discards the noisier ones.



## Data Augmentation

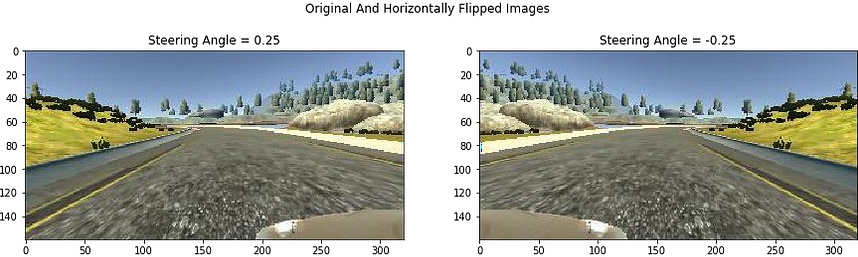
**Figure - 3.9**

Data augmentation is a technique used in deep learning to artificially increase the size and diversity of training datasets by applying various transformations to the existing data. By augmenting the data, the model learns to generalize better and becomes more robust to variations in the input. Data augmentation is commonly employed in various computer vision tasks, including object detection with SSD (Single Shot MultiBox Detector). Here are some commonly used data augmentation techniques:

1. Image Flipping:
   * Images can be horizontally flipped, meaning they are mirrored along the vertical axis.
   * This augmentation technique is useful when the object's orientation does not affect its class label, such as in object detection tasks.
2. Random Cropping and Resizing:
   * Randomly cropping and resizing images helps introduce spatial variations and changes in scale.
   * Cropping involves selecting a smaller region of the image, while resizing changes the image's dimensions.
   * This augmentation technique is beneficial for handling objects at different scales and aspect ratios.
3. Rotation and Shearing:
   * Rotation involves rotating the image by a random angle, introducing variations in the object's orientation.
   * Shearing applies a shear transformation to the image, distorting its shape along a given axis.
   * These augmentations help the model handle objects with different orientations and angles.
4. Color Jittering:
   * Color jittering modifies the image's color distribution by adding small random perturbations to its pixel values.
   * This augmentation technique helps the model become more invariant to changes in lighting conditions, contrasts, and color variations.
5. Adding Noise:
   * Adding random noise to the image can enhance the model's ability to handle noisy or low-quality input data.
   * Common types of noise include Gaussian noise, salt-and-pepper noise, and speckle noise.
6. Elastic Transformations:
   * Elastic transformations deform the image by locally distorting it based on random displacement fields.
   * This augmentation technique introduces spatial deformations, mimicking real-world deformations and variations.
7. Occlusion:
   * Occluding parts of the image by adding random patches or objects can help the model learn to handle occlusions and partial object visibility.

These are just a few examples of data augmentation techniques used in deep learning. The specific choice and combination of augmentation methods depend on the task, dataset, and domain. By applying data augmentation, models can learn from a more diverse range of training examples, leading to improved generalization and performance on unseen data.

According to the authors of SSD, data augmentation is essential for teaching the network to become more resilient to different object sizes in the input, just like in many other deep learning applications. In order to achieve this, they created extra training examples that included both random patches and patches from the original image that were cut out at various IoU ratios (e.g., 0.1, 0.3, 0.5, etc.). Additionally, each image is randomly horizontally flipped with a chance of 0.5, ensuring that prospective items are equally likely to appear on the left and the right.



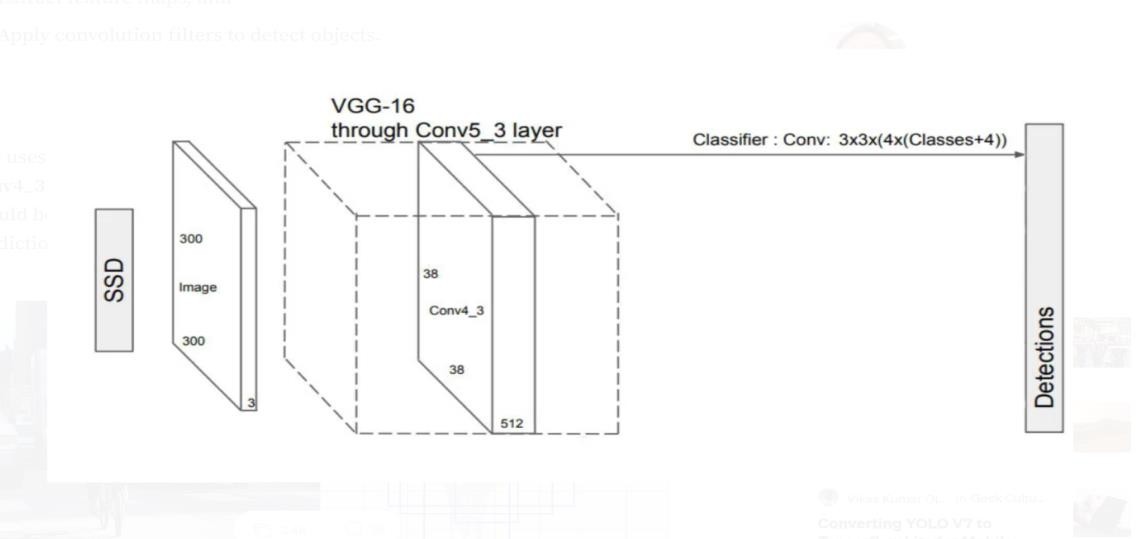
**Figure- 3.10**

## Working

Real-time object detection is a feature of SSD. Faster R-CNN uses boundary boxes that are generated by a region proposal network to classify things. The entire process moves at a speed of 7 frames per second, yet it is thought to be state-of-the-art in precision. Below the requirements for real-time processing.

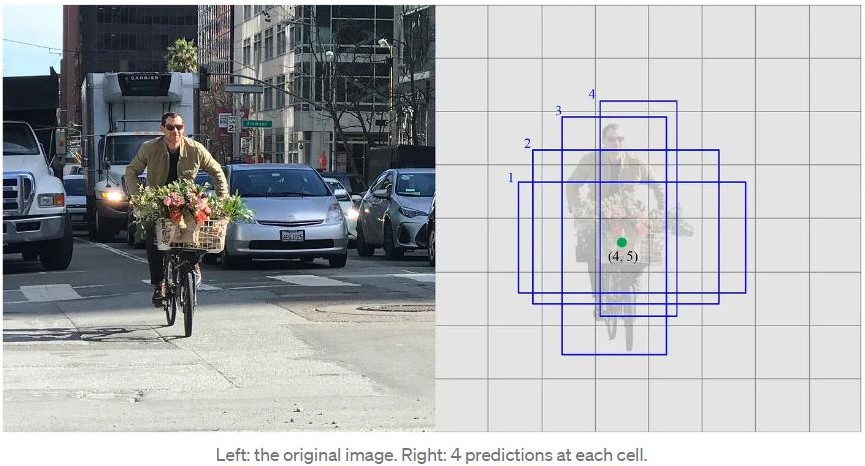
By doing away with the necessity for the regional proposal network, SSD accelerates the process. SSD implements a few improvements, such as multi- scale features and default boxes, to make up for the loss in accuracy. These enhancements enable SSD to match the Faster R-accuracy CNN’s while employing images of lesser quality, significantly increasing speed. The following comparison shows that it surpasses the accuracy of the Faster R- CNN and even achieves real-time processing performance.

The SSD object detection composes of 2 parts:

* Extract feature maps.
* Apply convolution filters to detect objects.

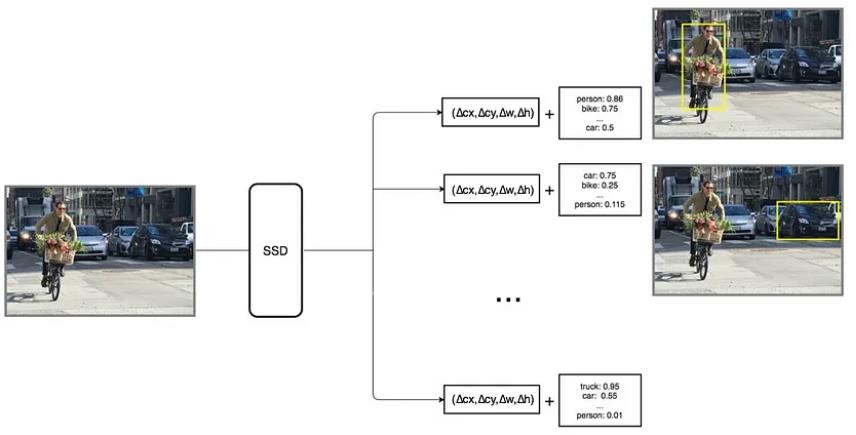
**Figure - 3.11**

SSD uses VGG16 to extract feature maps. Then it detects objects using the Conv4\_3 layer. For illustration, we draw the Conv4\_3 to be 8 × 8 spatially (it should be 38 × 38). For each cell (also called location), it makes 4 object predictions



**Figure - 3.12**

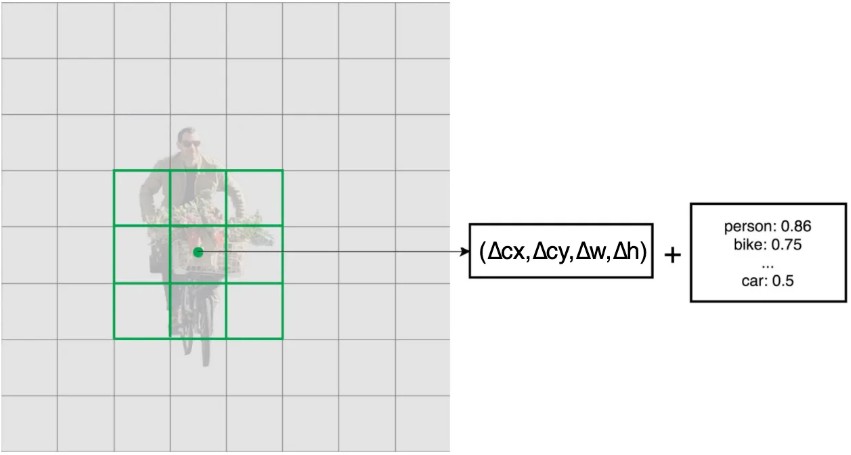
Each prediction composes of a boundary box and 21 scores for each class (one extra class for no object), and we pick the highest score as the class for the bounded object. Conv4\_3 makes a total of 38 × 38 × 4 predictions: four predictions per cell regardless of the depth of the feature maps. As expected, many predictions contain no object. SSD reserves a class “0” to indicate it has no objects.



**Figure - 3.13**

## Convolution predictors for object detection:

SSD does not use a delegated region proposal network. Instead, it resolves to a very simple method. It computes both the location and class scores using small convolution filters. After extracting the feature maps, SSD applies 3 × 3 convolution filters for each cell to make predictions. (These filters compute the results just like the regular CNN filters.) Each filter outputs 25 channels: 21 scores for each class plus one boundary box (detail on the boundary box later).



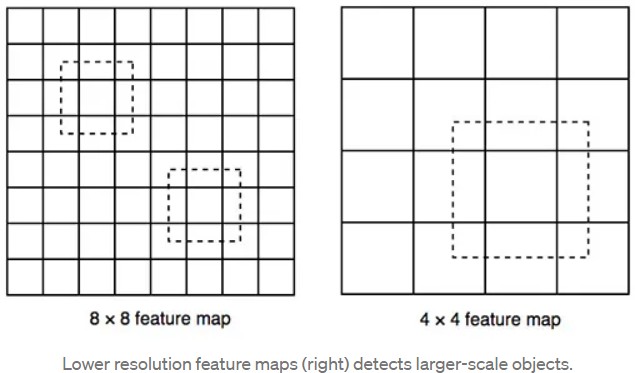
**Figure - 3.14**

For example, in Conv4\_3, we apply four 3 × 3 filters to map 512 input channels to 25 output channels.



## Multi-scale feature maps for detection:

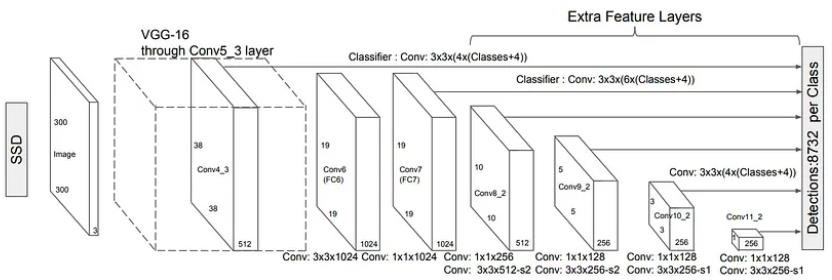
At first, we describe how SSD detects objects from a single layer. Actually, it uses multiple layers (multi-scale feature maps) to detect objects independently. As CNN reduces the spatial dimension gradually, the resolution of the feature maps also decrease. SSD uses lower resolution layers to detect larger scale objects. For example, the 4× 4 feature maps are used for larger scale object.



**Figure - 3.15**

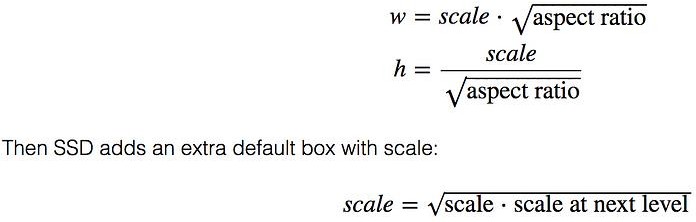
SSD adds 6 more auxiliary convolution layers after the VGG16. Five of them will be added for object detection. In three of those layers, we make 6 predictions instead of 4. In total, SSD makes 8732 predictions using 6 layers.

## Choosing default boundary boxes:



**Figure - 3.16**

Default boundary boxes are chosen manually. SSD defines a scale value for each feature map layer. Starting from the left, Conv4\_3 detects objects at the smallest scale 0.2 (or 0.1 sometimes), and then increases linearly to the rightmost layer at a scale of 0.9. Combining the scale value with the target aspect ratios, we compute the width and the height of the default boxes. For layers making 6 predictions, SSD starts with 5 target aspect ratios: 1, 2, 3, ½, and 1/3. Then the width and the height of the default boxes are calculated as:



And aspect ratio = 1.

## Matching Strategy:

SSD predictions are classified as positive matches or negative matches. SSD only uses positive matches in calculating the localization cost (the mismatch of the boundary box). If the corresponding default boundary box (not the predicted boundary box) has an IoU greater than 0.5 with the ground truth, the match is positive. Otherwise, it is negative. (IoU, the intersection over the union, is the ratio between the intersected areas over the joined area for two regions.)

## Additional SSD Notes

Due to the detector working on features at various resolutions, having more default boxes results in more accurate detection, albeit there is a speed cost. Having MultiBox on many layers also improves detection.

80% of the time is spent on the base VGG-16 network, thus SSD performance might be considerably better with a faster and more accurate network.

SSD makes mistakes when categorising items, such as animals. This is probable because different classes use the same places.

Using 512x512 input images, SSD-500 (the highest resolution variation) obtains the best mAP on Pascal VOC2007 at 76.8%, but at the cost of speed, as its frame rate drops to 22 fps. Thus, with 74.3, SSD-300 is a significantly better trade-off.

Smaller objects perform worse on SSD because some feature maps may not include them. While increasing the input image resolution helps to mitigate this issue, it does not entirely solve it.

For smaller objects, SSD performs worse than Faster R-CNN. Small items in SSD can only be detected in the leftmost, higher resolution layers. However, those layers include low-level features that are less useful for categorization, such as edges or colour patches.

With more default boundary boxes, accuracy improves at the expense of speed. Multi-scale feature maps help in object detection at various scales.

Accuracy will be improved by designing better default boundary boxes.

Objects in the COCO dataset are smaller. Use smaller default boxes (start with a smaller scale at 0.15 to enhance accuracy).

In comparison to R-CNN, SSD has lower localization error, but larger classification error when dealing with identical categories. We use the same border box to create numerous class predictions, which may be the reason of the greater classification mistakes.

SSD512 runs at 22 FPS rather than 59 FPS, but has greater precision (2.5%) than SSD300.

To improve accuracy, SSD can be educated from beginning to end. Location, scale, and aspect ratios are better covered by SSD, which also produces more

predictions. With the aforementioned enhancements, SSD may reduce the input image resolution to 300 300 while maintaining a comparable level of accuracy. The model can operate at real-time speed and still outperform the accuracy of the most advanced Faster R-CNN by eliminating the delegated region suggestion and using images of lesser resolution**The Drawbacks**

A neural network's shallow layers might not produce enough high-level characteristics to do prediction for small objects. Consequently, SSD performs less well for little things than for larger objects.

The requirement for complicated data augmentation also implies that it requires a substantial amount of data to train. For instance, if the model is pretrained on the COCO dataset, SSD performs better for Pascal VOC. Therefore, before training your model on your own data, make sure it has been pretrained on large datasets such Pascal VOC, COCO, and Open Images. I suppose that's the easiest fruit to choose.

While neural networks have proven to be powerful and versatile models for various tasks, they do have some drawbacks. Here are a few key limitations and challenges associated with neural networks:

1. Need for Large Amounts of Labelled Data:
   * Neural networks typically require a large amount of labelled training data to achieve high performance. Collecting and annotating large datasets can be time-consuming and expensive, especially for tasks with specific domain requirements.
2. Computationally Intensive:
   * Training and inference with large neural networks can be computationally expensive, requiring substantial computational resources. Complex architectures and large datasets can lead to long training times and high memory usage, which may limit their practicality for resource- constrained environments.
3. Black Box Nature:
   * Neural networks are often considered as "black box" models, meaning it can be challenging to interpret and understand the internal workings and decision-making processes of the model. This lack of interpretability can

be problematic in scenarios where explain-ability is crucial, such as medical or legal applications.

1. Vulnerability to Adversarial Attacks:
   * Neural networks can be susceptible to adversarial attacks, where intentionally crafted inputs can cause the model to produce incorrect outputs. Small perturbations or imperceptible changes to the input can lead to significant misclassifications, posing security concerns in applications where reliability is critical.
2. Overfitting and Generalization:
   * Neural networks are prone to overfitting, where the model performs well on the training data but fails to generalize to unseen data. Regularization techniques and careful model selection are necessary to mitigate overfitting and ensure good generalization performance.
3. Lack of Causality and Contextual Understanding:
   * Neural networks often lack a deep understanding of causality and contextual relationships in the data. They learn statistical associations from the training data without necessarily grasping the underlying causal mechanisms, which can limit their ability to handle complex reasoning and decision-making tasks.
4. Data Bias and Ethical Concerns:
   * Neural networks are sensitive to biases present in the training data. If the training data is biased or not representative of the target population, the model can inherit and perpetuate these biases, leading to unfair or discriminatory outcomes. Addressing data biases and ensuring ethical use of neural networks is an ongoing challenge.

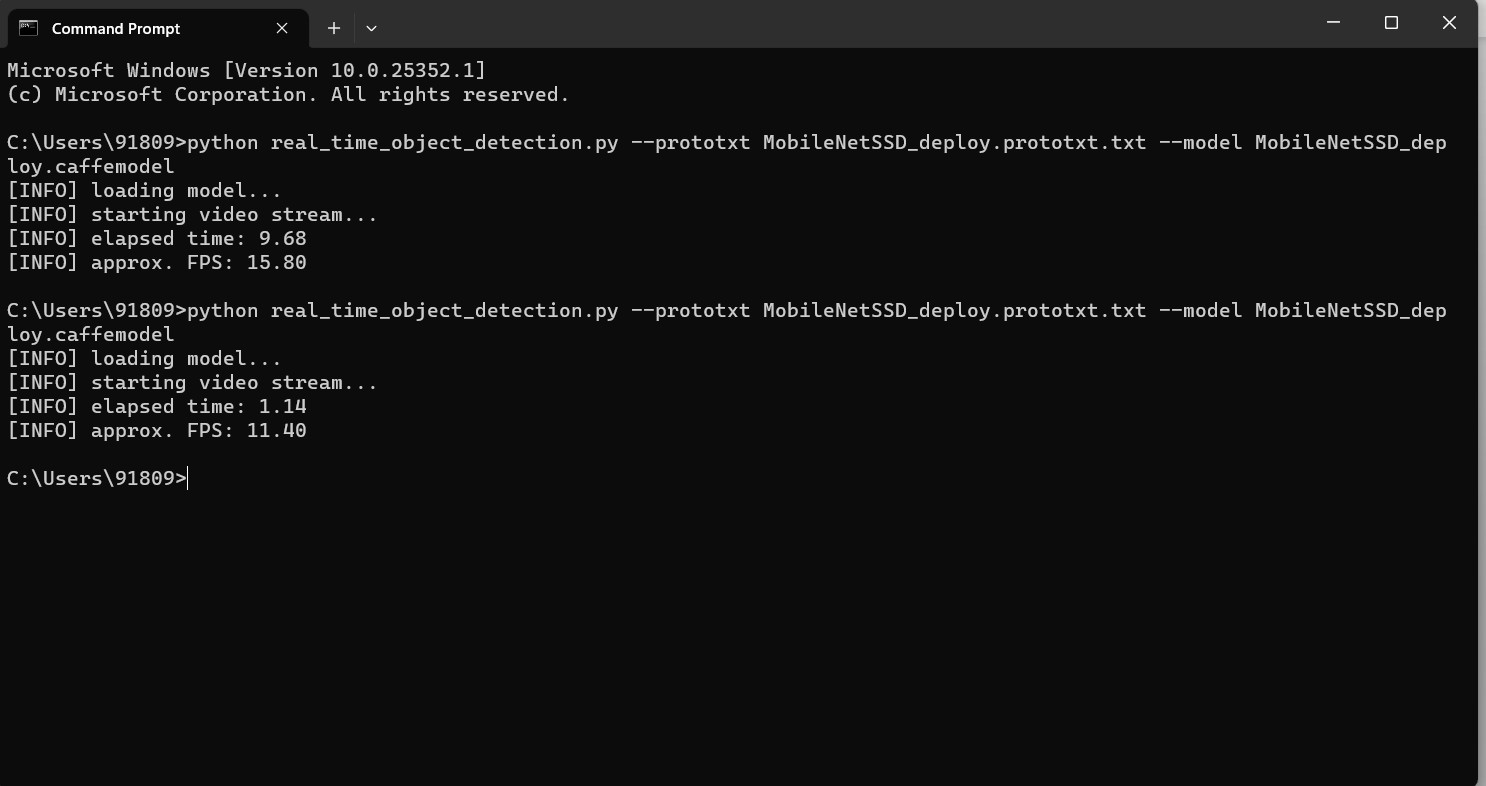
It is important to be aware of these drawbacks and limitations when working with neural networks and to carefully consider their implications for the specific application and domain. Researchers and practitioners are actively exploring

solutions and techniques to mitigate these challenges and enhance the capabilities and reliability of neural networks.

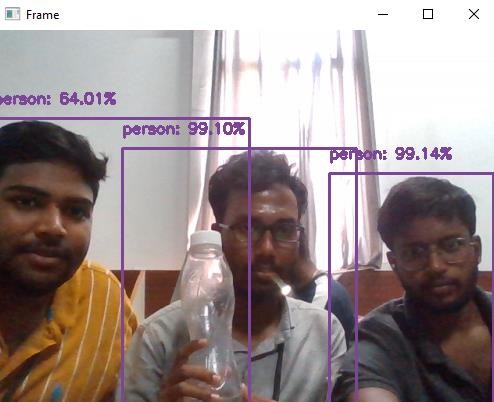
# RESULTS ANALYSIS AND VALIDATION

* + Open the command prompt and then enter the command as below :

## python real\_time\_object\_detection.py --prototxt MobileNetSSD\_deploy.prototxt.txt --model MobileNetSSD\_deploy.caffemodel



**Figure - 4.1**



**Figure - 4.2**

# CONCLUSION AND FUTURE WORK

One of the newest and most fascinating areas of deep learning is object recognition. Face detection is a popular use of object detection, found in almost all smartphone cameras. On multiple deep learning models capable of performing real-time object detection and recognition. By studying the performance of these algorithms on standard datasets, YOLOv3, Tiny-YOLOv3 and Faster R-CNN have been identified as the most suitable and efficient deep learning models to perform detection and recognition of real-time objects on scale engineering vehicles. It is concluded that Faster R-CNN performs on par with SSD and FCN models in terms of speed, while exhibiting better accuracy compared to these models.

As In the modern world, object detection and recognition can be regarded as one of the most difficult, complicated, and crucial tasks in the field of computer vision. As far as we know, this project was created with the fundamental goal of capturing real-time objects in images, videos, or web cams.

* Future innovations can be concentrated by putting the project on a system with GPU for quicker outcomes and greater accuracy. • It can be improved and innovated in the future by anyone without worrying about complexity.
* For instance, MS COCO does small object detection in several face detection tasks and applications. For better small object localization across partial barriers. In order to improve the network architecture, we will make some changes.
* To achieve accurate and efficient recognition of small objects, thereby reducing reliance on data networks.

Thus, it can be said in the end that in order to improve accuracy and performance, preprocessing techniques like edge detection and increasing image augmentation and contrast should be used.

# REFERENCES

1. S. Murugan, K. S. Devi (2018), A. Sivaranjani, and P. Srinivasan, “A study on various methods used for video summarization and moving object detection for video surveillance applications,” Multimer. Tools Appl., vol. 77, no. 18, pp. 23273– 23290.
2. G. Flitton, T. P. Breckon, and N. Megherbi, “A comparison of 3D interest point descriptors with application to airport baggage object detection in complex CT imagery,” Pattern Recognition., vol. 46, no. 9, pp. 2420–2436, 2013.
3. Bezak, P. (2016, September). Building recognition system based on deep learning. In 2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR) (pp. 1-5). IEEE.
4. Jung, H., Lee, S., Park, S., Kim, B., Kim, J., Lee, I., & Ahn, C. (2015, January). Development of deep learning-based facial expression recognition system. In 2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV) (pp. 1-4). IEEE.
5. Tenguria, R., Parkhedkar, S., Modak, N., Madan, R., & Tondwalkar, A. (2017, April). Design framework for general purpose object recognition on a robotic platform. In 2017 International Conference on Communication and Signal Processing (ICCSP) (pp. 2157-2160). IEEE.
6. Etemad, E., & GAO, Q. (2017, September). Object localization by optimizing convolutional neural network detection score using generic edge features. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 675-679). IEEE.
7. Mazumdar, M., Sarasvathi, V., & Kumar, A. (2017, August). Object recognition in videos by sequential frame extraction using convolutional neural networks and fully connected neural networks. In 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) (pp. 1485-1488). IEEE.

## LINKS:

[www.mindwise-groningen.nl/deep-learning-the-beautiful-mind/](http://www.mindwise-groningen.nl/deep-learning-the-beautiful-mind/)

https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-020-01826-x

https://[www.ijert.org/a-study-on-real-time-object-detection-using-deep-](http://www.ijert.org/a-study-on-real-time-object-detection-using-deep-) learning#:~:text=Object%20detection%20empowers%20recognizing%20instance,much%20li ke%20human%20vision%20works.

http://www.diva-portal.org/smash/get/diva2:1414033/FULLTEXT02.pdf https://[www.ijraset.com/research-paper/moving-object-detection-using-ml](http://www.ijraset.com/research-paper/moving-object-detection-using-ml) https://youtu.be/ATw1Dy4p1GU

https://youtu.be/hMFx1TXjAJc https://youtu.be/1P6K9pVKgVo