Single Run Multibox Object Detection Using A Single Deep Neural Network

CS18L1 Project

12150821 CSU14214 Bastin Saju
12150825 CSU14218 Emmanuel Lopez
12150858 CSU14246 Sajith A Rahim
12150860 CSU14247 Sebastian Thomas
12150869 CSU14259 Vishnu VS
B. Tech Computer Science & Engineering

Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

April 2018

Model Engineering College Thrikkakara
Department of Computer Engineering

C E R T I F I C A T E

This is to certify that, this report titled ***Single Run Multibox Object Detection Using A Single Deep Neural Network*** is a bonafide record of the work done by

12150821 CSU14214 Bastin Saju
12150825 CSU14218 Emmanuel Lopez
12150858 CSU14246 Sajith A Rahim
12150860 CSU14247 Sebastian Thomas
12150869 CSU14259 Vishnu VS

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CS18L1 Project**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **Cochin University of Science & Technology**.

Guide

Santhini K A
Assistant Professor
Computer Engineering

Coordinator

Dr. Priya S
Professor
Computer Engineering

Head of the Department

Manilal D L
Associate Professor
Computer Engineering

September 15, 2021

Acknowledgements

This project would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to all of them.

First of all, We would like to thank our esteemed Principal, Prof. (Dr.) V.P Devassia, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

We am highly indebted to our Project Coordinator, Dr. Priya S, Professor and Head of the Department, Dr. Manilal D L, Associate Professor for their guidance, support and constant supervision throughout the duration of the work as well as for providing all the necessary information and facilities that this work demanded.

We would like to thank our Project Guide, Santhini K A, Assistant Professor for her support and valuable insights and also for helping me out in correcting any mistakes that were made during the course of the work.

We offer our sincere gratitude to all our friends and peers for their support and encouragement that helped us get through the tough phases during the course of this work.

Last but not the least, we thank the Almighty God for guiding us through and enabling us to complete the work within the specified time.

Bastin Saju

Emmanuel Lopez

Sajith A Rahim

Sebastian Thomas

Vishnu V.S

Abstract

In this era image processing is not a big deal, but the time consumed for computation and accuracy are the two things which cannot be compromised.We present a method for detecting objects in images using a single deep neural network.Single-Run Multibox detector, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Our model is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stage and encapsulates all computation in a single network thereby reducing the latency for applications which require real time monitoring.

Contents

List of Figures

Chapter 1

Introduction

Object detection has many practical applications and is currently being used in many promising and well publicized technologies that have caught the public imagination including but not limited to self-driving cars , and detecting people for purposes of security. Recognition and detection are also useful in diagnosis using medical imaging, and various scientific imaging applications. The task of finding objects belonging to classes of interest in images has long been a focus of Computer Vision research. The ability to localize objects is useful in many applications: from self-driving cars, where it allows the car to detect pedestrians, bicyclists, road signs, and other vehicles, to security, where intruding persons can be detected. Though a lot of progress has been made since the conception of the eld of Computer Vision more than ve decades ago, as always, there is scope for further improvement. This is especially true in the case of object detection where a myriad of factors including variation in object instances through pose and appearance, along with other environmental factors such as the degree of occlusion, and lighting tend to cause failures.

1.1 Proposed Project

1.1.1 Problem Statement

Current state-of-the-art object detection systems are variants of the following approach: hypothesize bounding boxes, resample pixels or features for each box, and apply a high quality classifier. This pipeline has prevailed on detection benchmarks since the Selective Search work through the current leading results on PASCAL VOC, COCO, and ILSVRC detection all based on Faster R-CNN. The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage.

Our improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.This results in a significant improvement in speed for high-accuracy detection 59 FPS ,when compared to Faster R-CNN 7 FPS and 45 FPS with YOLO.

1.1.2 Proposed Solution

The SSD detector differs form other single shot detectors due to the usage of multiple layers that provide a finer accuracy on objects with different scales.SSD dosen't resample pixels iteratively within the window. It is faster than traditional pipeline based detectors and it has a real time applicability of upto 60 fps. It also has a same level accuracy with greater speeds.Single-Run Multibox detector, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Our model is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stage and encapsulates all computation in a single network thereby reducing the latency for applications which require real time monitoring. Advantages:

- Scale based Object Detection
- Fast Classification - fast enough for real time applications
- Easily Integratable Module
- Scalability

Disadvantages:

- Library Dependencies make the module inviable for embedded applications.
- Hardware Requirements are quite high for embedded applications.
- Training Of the Net is cumbersome and requires large processing time.
- Mean Averege Precision is below 80%.

Chapter 2

System Study Report

2.1 Literature Survey

2.1.1 Approaches

Scale based Object Detection

The first stage of Scale based detection is to identify the locations and scales of the keypoint that can be repeatedly detected under different views of the same object. As the keypoint can be repeatedly detected, we call it stable features. Detecting stable features that are invariant to locations is achieved by searching for most of the locations over the image. To extend its invariance to scales, all possible scales of the image are searched instead of one scale only.

Region based Object Detection

Region-based object recognition systems have the advantage of greater generality and more easily trainable from visual data. View-based approach is generally a useful technique. However, since matching is done by comparing the entire objects, some methods are more sensitive to background clutter and occlusion. Some methods solve this problem by applying image segmentation on the entire objects so as to divide the image representations into smaller pieces for matching separately. Some other methods avoid using segmentation and solve the problem by employing voting techniques, like Hough transform methods. This technique allows evidence from disconnected parts to be effectively combined.

2.1.2 Existing Methods

DPM(Deformable Part Model)

In very high level, DPM assumes an object is constructed by its parts. Thus, the detector will first found a match of its whole, and then using its part models to fine-tune the result. DPM is known for its ability to identify difficult objects.

DPM is not known for its speed. Its ability to identify difficult objects, is the selling point. However, this implementation tries to optimize for speed as well. For a 640x480 photo, this implementation will be done in about one second, without multi-thread support.

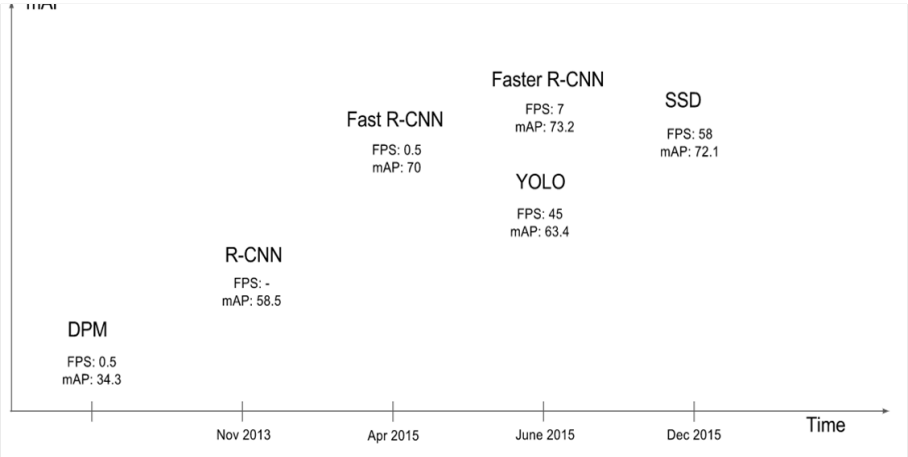


Figure 2.1: Timeline

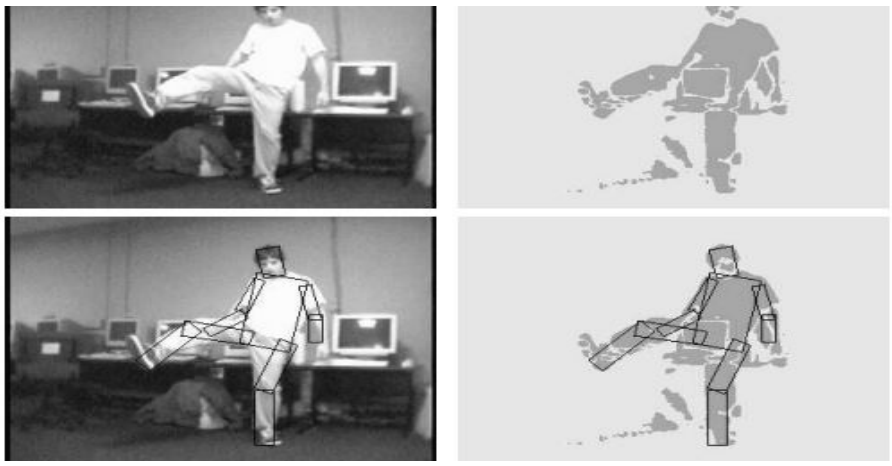


Figure 2.2: DPM

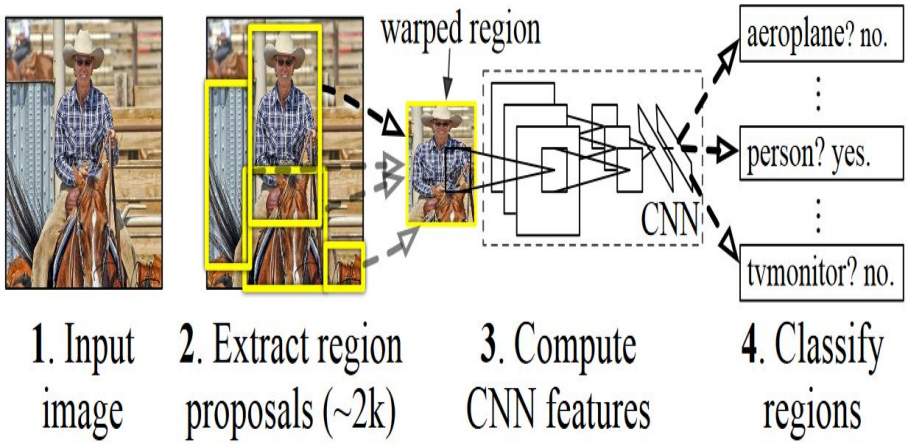


Figure 2.3: RCNN

Accuracy-wise:

There are two off-the-shelf implementations. One is the DPM in Matlab from the inventor, the other is the HOG detector from OpenCV.

RCNN

Region based CNN is a simple and scalable detection algorithm that improves mAP 30% relative to the previous best result on VOC 2012 achieving a mAP of 53.3%. This approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.
Advantages:

- RCNN has an mAP of 53.3%.

Disadvantages:

- It requires a forward pass of the CNN for every single region proposal for every single image(thats around 2000 forward passes per image).

- It has to train three different models separately. The CNN to generate image features, the classifier that predict class and the regression model to tighten the bounding boxes.This makes it extremely train to hard.

Fast RCNN

For the forward pass of CNN, for each image, a lot of proposed regions for the image invariably overlapped causing us to run same CNN computation again and again.Fast RCNN avoids this overlapping scenario using a technique known as RoIPool (Region of Interest Pooling). At its core,

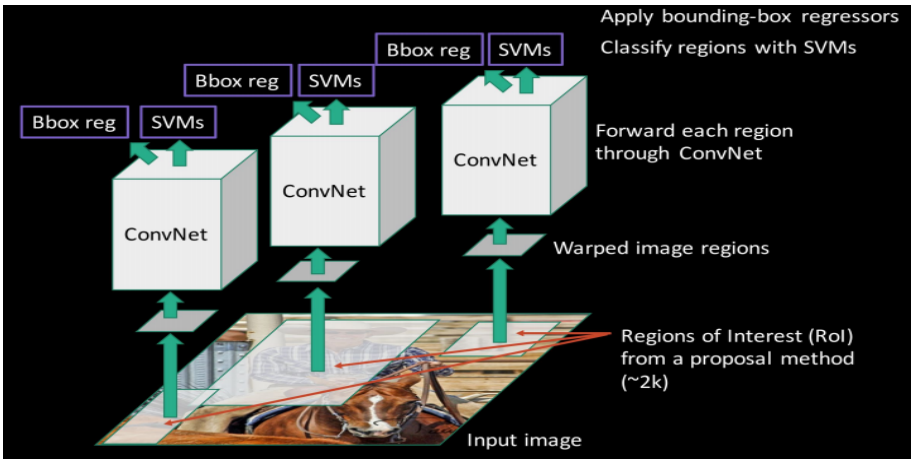


Figure 2.4: Fast-RCNN

RoIPool shares the forward pass of a CNN for an image across its subregions. In the image above, notice how the CNN features for each region are obtained by selecting a corresponding region from the CNNs feature map. Then, the features in each region are pooled (usually using max pooling). So all it takes us is one pass of the original image as opposed to 2000!.The second insight of Fast R-CNN is to jointly train the CNN, classifier, and bounding box regressor in a single model. Where earlier we had different models to extract image features (CNN), classify (SVM), and tighten bounding boxes (regressor), Fast RCNN instead used a single network to compute all the three.

Advantages:

- Fast RCNN avoids redundant computation by employing a technique called ROI.

Disadvantages:

- In Fast RCNN a bunch of region proposals is created using selective search, a slow process that it takes a lot of time.

Faster RCNN

The insight of Faster R-CNN was that region proposals depended on features of the image that were already calculated with the forward pass of the CNN (first step of classification).Indeed, this is just what the Faster R-CNN team achieved.This way, only one CNN needs to be trained and we get region proposals almost for free.Here are the inputs of their model: 1. Inputs: Images(Notice how region proposals are not needed). 2. Outputs: Classifications and bounding box coordinates of the objects in the images.

Advantages:

- Faster than Fast RCNN

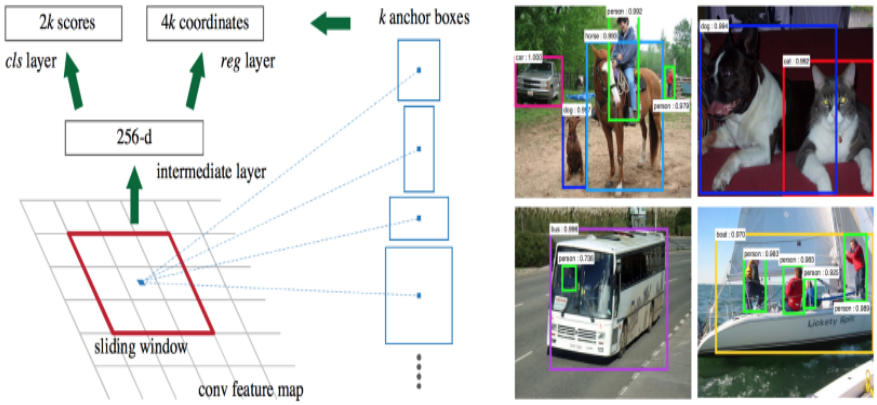


Figure 2.5: Faster RCNN

- Less Computation overhead
- Not significantly faster
- Depends on an external system for Region Proposals
- Depends on an external system for Region Proposals
- Not suitable for real-time applications.

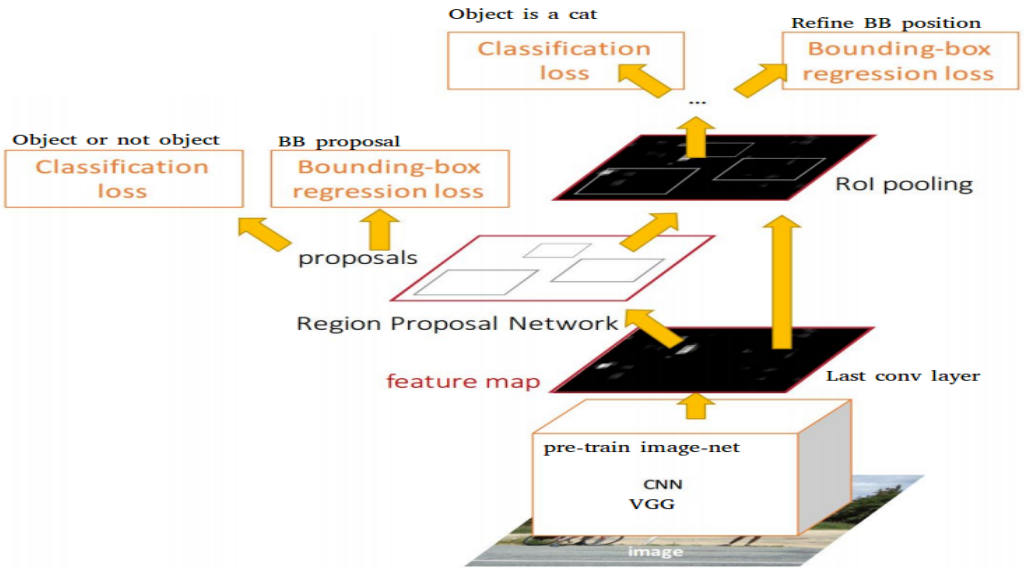


Figure 2.6: Faster RCNN

YOLO(You Only Look Once)

YOLO a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection.Object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Unified architecture of YOLO is extremely fast.

YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is far less likely to predict false detections where nothing exists.

Finally, YOLO learns very general representations of objects. It outperforms all other detection methods, including DPM and RCNN, by a wide margin when generalizing from natural images to artwork on both the Picasso Dataset and the PeopleArt Dataset.

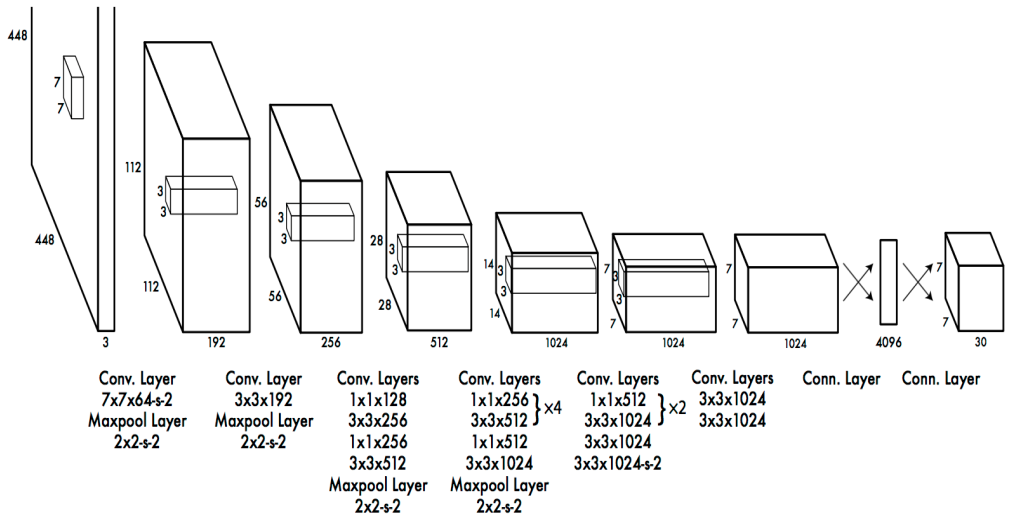


Figure 2.7: YOLO - Design

2.2 SSD : Single Run Mutibox Object Detector

2.2.1 Proposal

Current state-of-the-art object detection systems are variants of the following approach: hypothesize bounding boxes, re-sample pixels or features for each box, and apply a high quality classifier. This pipeline has prevailed on detection benchmarks since the Selective Search work through the current leading results on PASCAL VOC, COCO, and ILSVRC detection all based on Faster R-CNN.

The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature re-sampling stage. Our improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.

SSDframework.

(a)SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 88 and 44 in (b) and (c)). For each default box, we predict both the shape offsets and the condences for all object categories $((c_1, c_2, , c_p))$. At training time, we rst match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and condence loss (e.g. Softmax).

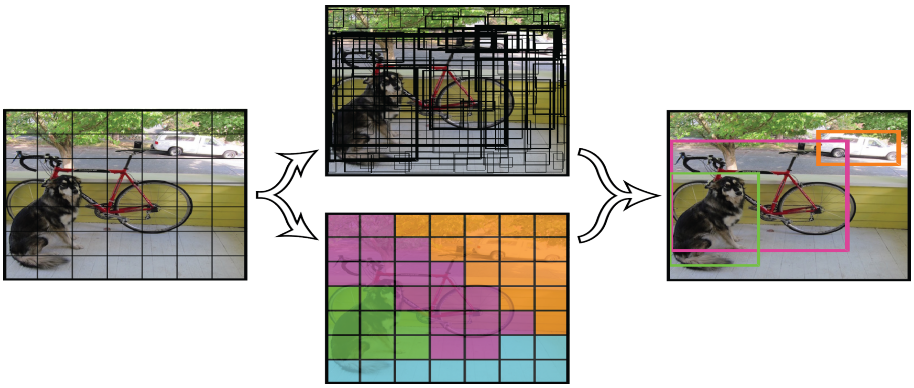


Figure 2.8: YOLO

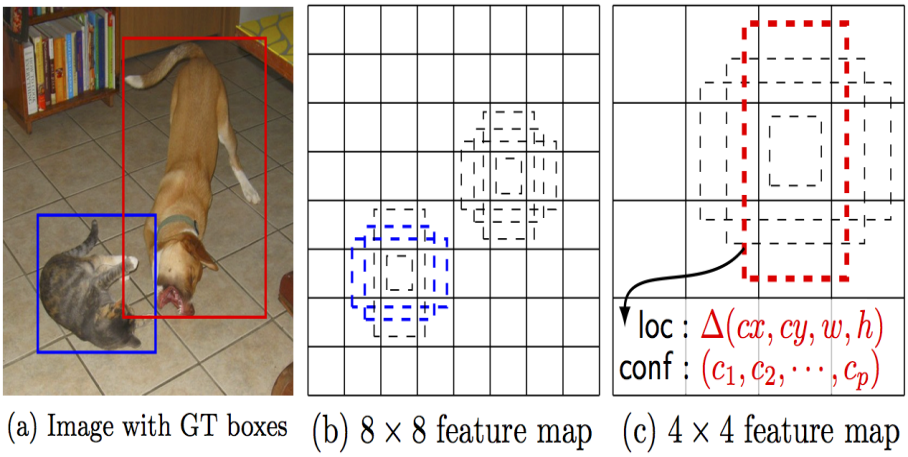


Figure 2.9: SSD

Features:

- SSD, a single-shot detector for multiple categories that is faster than the previous state-of-the-art for single shot detectors (YOLO), and signicantly more accurate, in fact as accurate as slower techniques that perform explicit region proposals and pooling (including Faster R-CNN).
- The core of SSD is predicting category scores and box offsets for a xed set of default bounding boxes using small convolutional lters applied to feature maps
- To achieve high detection accuracy we produce predictions of different scales from feature maps of different scales, and explicitly separate predictions by aspect ratio.
- These design features lead to simple end-to-end training and high accuracy, even on low resolution input images, further improving the speed vs accuracy trade-off.

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

The product described in this software requirement specification is single run multibox object detection using a single deep neural network which can be used for real time object detection.

3.1.2 Document Conventions

This document follows MLA Format. Bold-faced text has been used to emphasize section and sub-section headings. Highlighting is to point out words in the glossary and italicized text is used to label and recognize diagrams.

3.1.3 Intended Audience and Reading Suggestions

This document is to be read by the development team, the project guide, system testers and documentation writers. College faculty may review the document to learn about the project and to understand the requirements. The SRS has been organized approximately in order of increasing specificity. The developers need to become intimately familiar with the SRS.

3.1.4 Project Scope

Object detection has many practical applications and is currently being used in many promising and well publicized technologies that have caught the public imagination including but not limited to self-driving cars , and detecting people for purposes of security. Recognition and detection are also useful in diagnosis using medical imaging, and various scientific imaging applications.

Current state-of-the-art object detection systems are variants of the following approach: hypothesize bounding boxes, resample pixels or features for each box, and apply a high quality classifier. This pipeline has prevailed on detection benchmarks since the Selective Search work through the current leading results on PASCAL VOC, COCO, and ILSVRC detection all based on Faster R-CNN. The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage. Our improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate

predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.

3.1.5 Overview of Developers Responsibilities

- Deliver a fully function SSD Module
- Improvise Detection using additional layers in Deep Convolutional Network
- Provide Scalablility to possible extends.
- Provide Guidelines for Embedded Implementations.

.

3.2 Overall Description

3.2.1 Product Perspective

The software module being developed is for a new portable stand-alone module based on PyTorch and Tensorflow which functions as a detector and classifier for real-time Object Detection.

Our approach, discretizes the output space of bounding boxes into a set of default boxes over various aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. Single-run is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes it easy to train and straightforward to integrate into systems that require a detection component.

Current state-of-the-art object detection systems are variants of the following approach: hypothesize bounding boxes, resample pixels or features for each box, and apply a high quality classifier. This pipeline has prevailed on detection benchmarks since the Selective Search work through the current leading results on PASCAL VOC, COCO, and ILSVRC detection all based on Faster R-CNN. The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage. Our improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales

3.2.2 Product Functions

Our improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.

This results in a significant improvement in speed for high-accuracy detection 59 FPS ,when compared to Faster R-CNN 7 FPS and 45 FPS with YOLO.

- Scale based Object Detection
- Fast Classification - enough for real time applications
- Easy Integration
- Provide Scalability

3.2.3 User Classes and Characteristics

Face Detection

Popular applications include face detection and people counting. Have you ever noticed how facebook detects your face when you upload a photo? This is a simple application of object detection that we see in our daily life.

People Counting

Object detection can be also used for people counting, it is used for analysing store performance or crowd statistics during festivals. These tend to be more difficult as people move out of the frame quickly (also because people are non rigid objects).

Vehicle Detection

Similarly when the object is a vehicle such as a bicycle or car, object detection with tracking can prove effective in estimating the speed of the object. The type of ship entering a port can be determined by object detection(depending on shape, size etc). This system for detecting ships are currently in development in some European countries.

3.2.4 Manufacturing Industry

Object detection is also used in industrial processes to identify products. Say you want your machine to only detect circular objects. Hough circle detection transform can be used for detection.

3.2.5 Online Images

Apart from these object detection can be used for classifying images found online. Obscene images are usually filtered out using object detection.

3.2.6 Security

In the future we might be able to use object detection to identify anomalies in a scene such as bombs or explosives (by making use of a quadcopter).Many such devices are under test-phase currently and are showing promise.

3.2.7 Video Surveilence and Tracking

UAVs equipped with Object Detectors are used to detect people,vehicles and other critical objects in Borders and WarZones. Modern UAVs have better hardwire and Object Detectors which allows self navigation and stall resolution.

3.2.8 Self Driving Cars

Self Driving Cars use object detection for Navigating through roads .Googles LIDAR system is great for generating an accurate map of the cars surroundings, but its not ideal for monitoring the speed of other cars in real time. Thats why the front and back bumper of the driverless car includes radar.The stream is fed into object detector which detects and classifies obstructions.

3.3 Design and Implementation Constraints

3.3.1 Library Dependancy

Uses PyTorch - Only Available for Windows and Unix based OS and Arduino. PyTorch is a Python package with a different way of constructing the neural network. It provides Tensors and has the ability to enhance computation speed. At a basic level, it is a library comprises of the different components such as torch that support strong GPU support, torch.util, torch.autograd which supports all tensor operation and various other components. The framework is used as a replacement for numpy to utilise the power of GPU or a research platform which enhances flexibility and speed.

3.3.2 GPU

Requires High Power GPU for Training.

NVIDIAs standard libraries made it very easy to establish the first deep learning libraries in CUDA, while there were no such powerful standard libraries for AMDs OpenCL. Right now, there are just no good deep learning libraries for AMD cards so NVIDIA it is. Even if some OpenCL libraries would be available in the future I would stick with NVIDIA: The thing is that the GPU computing or GPGPU community is very large for CUDA and rather small for OpenCL.

3.3.3 Language requirements

Classifier will support only these languages: English, French, German, Spanish, Mandarin, Japanese, Arabic and Hindi.

3.4 User Documentation

- https://docs.anaconda.com/
- pytorch.org/tutorials/

3.5 General Constraints

The accuracy of the proposed system is less than that of existing system.

3.6 Assumptions and Dependencies

Development of this module is based on the assumption that Long Term Support will be provided for PyTorch and Adaption States won't change in the near future.Any such change must be refelected on the Implemetation if required in future.

3.7 External Interface Requirements

3.7.1 User Interfaces

Detection and Classification Result will be obatined via a seprate Live Stream Window.
No Specific Requiremnts. There isn't an input interface.Input is provided via commandline. Output is obtained as augmented video in a PythonQt window.

3.7.2 Hardware Interfaces

SSD Module can be integrated to a software or can be used as standalone. Hardware Interface is provide by the OS hence (No Specific Requirement)

3.7.3 Software Interfaces

SSD implementation based on PyTorch supports Windows(Anaconda) and any other Unix Operating System.There are no specific Software interface except for IPC handled by OS.

3.7.4 Communication Interfaces

Not Applicable

3.8 Hardware and Software Requirements

3.8.1 Hardware Requirements

- Processor: 2GHZ Dual Core (Core 2 Duo 2.4GHZ or Athlon x2 2.7GHZ)
- Memory: 2GB RAM
- Graphics card AMD: DirextX 10.1 compatible with 512MB RAM (ATI Radion 3000, 4000, 5000 or 6000 series, with ATI Radion 3870 or higher performance)
- Graphics card Nvidia: DirectX 10.0 compatible with 512MB RAM (Nvidia GeForce 8, 9, 200, 300, 400 or 500 series with Nvidia GeForce 8800 GT or higher performance)

3.8.2 Software Requirements

Implementation Platform

Any Unix kernal base OS like Ubuntu,Manjaro,ArchLinux etc

3.8.3 Reason for choosing Unix:

- With Unix you have in general the option of using either command-lines (more control and flexibility) or GUIs (easier).
- Unix is more flexible and can be installed on many different types of machines, including mainframe computers, supercomputers, and micro-computers.
- Unix is more stable and does not go down as often as other OS, therefore requires less administration and maintenance.
- Unix has greater built-in security and permissions features.
- Unix possesses much greater processing power.
- The mostly free or inexpensive open-source operating systems, such as Linux and BSD, with their flexibility and control

PyTorch - Deep Learning Framework

PyTorch is a Python package with a different way of constructing the neural network. It provides Tensors and has the ability to enhance computation speed. At a basic level, it is a library comprises of the different components such as torch that support strong GPU support, torch.util, torch.autograd which supports all tensor operation and various other components. The framework is used as a replacement for numpy to utilise the power of GPU or a research platform which enhances flexibility and speed. It provides you with a feature to improve the way network behaves with zero lag or overhead and the technique is called Reverse-mode auto-differentiation. Intel MKL and NVIDIA are libraries used to maximize speed.

PyTorch has a unique way of building neural networks: using and replaying a tape recorder.

Most frameworks such as TensorFlow, Theano, Caffe and CNTK have a static view of the world. One has to build a neural network, and reuse the same structure again and again. Changing the way the network behaves means that one has to start from scratch.

With PyTorch, we use a technique called Reverse-mode auto-differentiation, which allows you to change the way your network behaves arbitrarily with zero lag or overhead. Our inspiration comes from several research papers on this topic, as well as current and past work such as autograd, autograd, Chainer, etc.

PyTorch has minimal framework overhead. We integrate acceleration libraries such as Intel MKL and NVIDIA (CuDNN, NCCL) to maximize speed. At the core, its CPU and GPU Tensor and Neural Network backends (TH, THC, THNN, THCUNN) are written as independent libraries with a C99 API. They are mature and have been tested for years.

Hence, PyTorch is quite fast whether you run small or large neural networks.

The memory usage in PyTorch is extremely efficient compared to Torch or some of the alternatives. Weve written custom memory allocators for the GPU to make sure that your deep learning models are maximally memory efficient. This enables you to train bigger deep learning models than before.

A network written in PyTorch is a Dynamic Computational Graph (DCG). It allows you to do any thing.

- Dynamic data structures inside the network. You can have any number of inputs at any given point of training in PyTorch. Lists? Stacks? No problem.
- Networks are modular. This is my favorite feature, actually. Each part is implemented separately, and you can debug it separately, unlike a monolithic TF construction.

3.9 Functional Requirements

The main functional requirements are the following:

3.9.1 Dataset

One of the hardests problems to solve in deep learning has nothing to do with neural nets: its the problem of getting the right data in the right format.

Deep learning, needs a good training set to work properly. Collecting and constructing the training set a sizable body of known data takes time and domain-specific knowledge of where and how to gather relevant information. The training set acts as the benchmark against which deep-learning nets are trained.

Training

Running a training set through a neural network teaches the net how to weigh different features, assigning them coefficients according to their likelihood of minimizing errors in your results.

Those coefficients, also known as meta-data, will be contained in vectors, one for each each layer of your net. They are the most important results you will obtain from training a neural network.

Testing

It functions as a seal of approval, and you dont use it until the end. After youve trained and optimized your data, you test your neural net against this final random sampling. The results it produces should validate that your net accurately recognizes images, or recognizes them at least x percentage of them.

If you dont get accurate predictions, go back to the training set, look at the hyper-parameters you used to tune the network, as well as the quality of your data and look at your pre-processing techniques.

Dataset Recommended

Microsoft COCO: Common Objects in Context Dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labeled instances in 328k images, the creation of our dataset drew upon extensive crowd worker involvement via novel user interfaces for category detection, instance spotting and instance segmentation. Objects are labeled using per-instance segmentations to aid in precise object localization.

PASCAL VOC - Visual Object Class This dataset is a set of additional annotations for PASCAL VOC 2010. It goes beyond the original PASCAL semantic segmentation task by providing annotations for the whole scene. The statistics section has a full list of 400+ labels.Since the dataset is an annotation of PASCAL VOC 2010, it has the same statistics as those of the original dataset. Training and validation contains 10,103 images while testing contains 9,637 images.

3.9.2 Video Pre-processing

The input video from the source (Disk/Webcam) is processed and converted into frames inorder to pass to the Deep Nueral Net. The input requirements of the net are same as that of VGG16 ImageNet i.e a size of 299 X 299.

OpenCV Library

OpenCV (Open Source Computer Vision) is a library of programming functions for realtime computer vision. It uses a BSD license and hence its free for both academic and commercial use. It has C++, C, Python and Java (Android) interfaces and supports Windows, Linux, Android, iOS and Mac OS. It has more than 3000 optimized algorithms. Adopted all around the world, OpenCV has more than 15 million downloads growing by nearly 280K/month. Usage ranges from interactive art, to mines inspection, stitching maps on the web on through advanced robotics.

Frame Extraction

Frames can be extracted from the input video feed using OpenCV function ;

video_to_frames(input_loc, output_loc) Function to extract frames from input video file and save them as separate frames in an output directory.
Args:
input_loc: Input video file.
output_loc: Output directory to save the frames.

Resizing Frame Extracted

Frames extracted can be resized into VGG 16 input requirement of 299x299 using OpenCV function;

cv.Resize(src, dst, interpolation=CV_INTER_LINEAR)
Args:
src - input image.
dst - output image;

3.10 Non-functional Requirements

3.10.1 Performance Requirements

Performance is one of the two major issues for our system. The system should have a high performance such that the user should be able to see a the augmented view of the area streaming with 25 fps.

Accuracy

Accuracy of the system is important for the user. For this reason, under the given assumptions, the system should have 100% accuracy, meaning that it should recognize all cars correctly. SSD classifies a success if probability is greater than 0.5 else goes undetected.

Simplicity

Regarding user interface, an important design goal is keeping it simple. User interface should not be so complicated keeping in mind that the module is to be integrated.Hence only code-based PythonQt Windows are used for Output and shell commands are used for input.

3.10.2 Safety Requirements

Information transmission should be securely transmitted to server without any changes in information. If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

3.10.3 Security Requirements

Since the system will be used offline and by a single user, *security is not an issue.* There are no disk or thread modification operations done by the Neural Net process hence is isolated from the rest of the process threads..

3.10.4 Software Quality Attributes

Availability

System will be available as long as the computer is running on is available.The thread can only be terminated by the OS or by kill command and hence availabilty constraits are null.

3.10.5 Maintainability

Our system is specified for one task. There will not be so much change in the future. Therefore, maintainability does not have more importance than it has in a normal software.

3.11 Other Requirements

3.11.1 Output Interface

The classified Object classes along with the bounding boxes augmented onto the input frame are displayed to the user as output through a Window using PythonQt.

PythonQt

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British firm Riverbank Computing. It is available under similar terms to Qt versions older than 4.5; this means a variety of licenses including GNU General Public License (GPL) and commercial license, but not the GNU Lesser General Public License (LGPL).PyQt supports Microsoft Windows as well as various flavours of Unix, including Linux and macOS.

3.11.2 Object Recognition

- There must be 5 meters between Object for camera.
- There should not be any obstacles between camera and the Object.
- Weather effects like fog will reduce the accuracy of the recognition.

3.11.3 Object State

- The Object must be in detection field for atleast 10 seconds.
- Object mustn't be occluded.
- Object must not emit any light
- Object must not be heavily deformed.

3.11.4 Output Augementation

- Object recognized is augmented by placing a border around it, the color of the border is dependent on the class of Object.
- System user/watcher can save a frame from the video stream.

Chapter 4

System Design

4.1 Introduction

Object detection has many practical applications and is currently being used in many promising and well publicized technologies that have caught the public imagination including but not limited to self-driving cars , and detecting people for purposes of security. Recognition and detection are also useful in diagnosis using medical imaging, and various scientific imaging applications.

4.2 Purpose

The purpose of this document is to present a detailed description of the system for Single-run Multi-Box Object Detector Using Single Deep Neural Net.The task of nding objects belonging to classes of interest in images has long been a focus of Computer Vision research. The ability to localize objects is useful in many applications: from self-driving cars, where it allows the car to detect pedestrians, bicyclists, road signs, and other vehicles, to security, where intruding persons can be detected. Though a lot of progress has been made since the conception of the eld of Computer Vision more than ve decades ago, as always, there is scope for further improvement. This is especially true in the case of object detection where a myriad of factors including variation in object instances through pose and appearance, along with other environmental factors such as the degree of occlusion, and lighting tend to cause failures.

In this work we focus on improving object detection through the use of more representative features and better models. We propose new features that are not only more powerful, but also more robust and capture more information than the currently popular features.Our approach, discretizes the output space of bounding boxes into a set of default boxes over various aspect ratios and scales per feature maplocation. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with dierent resolutions to naturally handle objects of various sizes. Single-run is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes it easy to train and straightforward to integrate into systems that require a detection component.

4.3 Overview

This document is the design document for the project Single-run Multi-Box Object Detector Using Single Deep Neural Net.

In the chapter System Architecture, a detailed architecture of the proposed system is shown. It includes the architecture diagram as well as explanation of the various modules of the system such as Video Preprocessing, Feature extraction, Classification. The chapter Data Description includes Data ow Diagrams and Use case Diagrams. The chapter Algorithms include the various Algorithms used in the Project.

Chapter 5

System Architecture

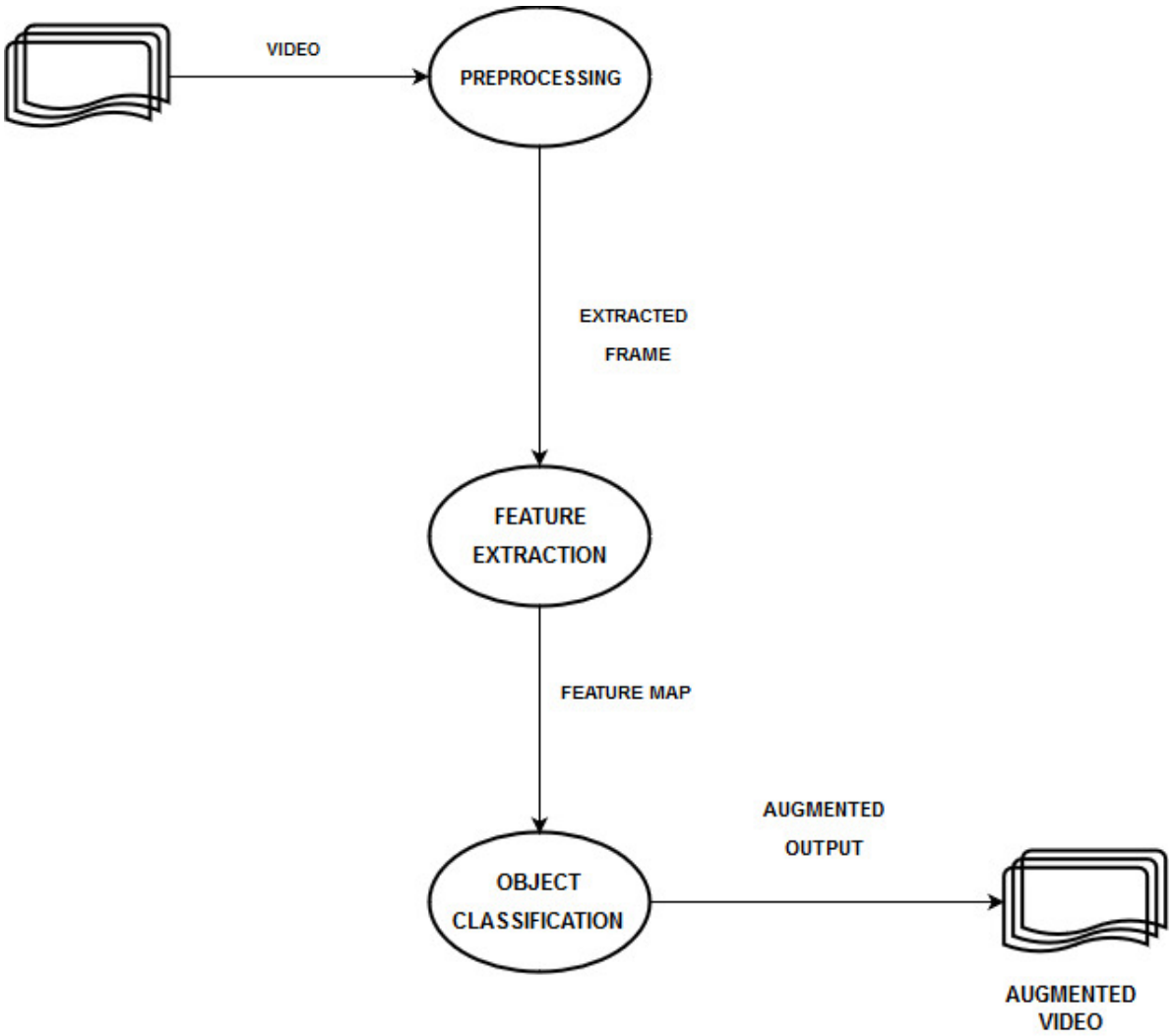


Figure 5.1: System Architecture

5.1 Detailed System Architecture

System has 3 Components:

- Video Pre-processing
- Feature Extraction
- Object Classification

5.2 Video Pre-processing

An image in a photograph is called a raw image, in order to extract useful information from it, it must be converted into certain form that meets the Neural Nets Requirements. The first step in preparing the image for object detection is called preprocessing. The input video from the source (Disk/Webcam) is processed and converted into frames inorder to pass to the Deep Neural Net. The input requirements of the net are same as that of VGG16 ImageNet i.e a size of 299 X 299.

5.2.1 Frame Extraction

The Input video is converted into frames and stored in a predefined buffer .This is done using OpenCV.

Input: Video Stream [Max Resolution :1280 x 760 @68fps]
Process: Conversion of Video to Frames using OpenCV
Output: Video Frames

5.2.2 Frame Resizing

Converted frames are then resized to 299 x 299 size to meet the input requirement of VGG Net.

Input: Video Frames
Process: Resizing of Frame to Required Dimensions.
Output: Resized frames.

5.3 Feature Extraction

In image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

Input: Frames [299 x 299]
Process: Feature Generation Over Convolution
Output : Feature Map [Vectors]

5.4 Object Classification

In feature extraction an image of fixed size is converted to a feature vector of fixed size. Classification algorithm takes this feature vector as input and outputs a class label.

Input: Feature Map Vectors
Process: Classification into Classes
Output: Object Classes

5.5 VGG 16 Net Architecture

VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day is it still considered to be an excellent vision model, although it has been somewhat outperformed by more revent advances such as Inception and ResNet.

5.6 SSD Module Architecture

SSD is a Single Deep Convolutional Nueral Net which include a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.It is built on top of VGG16 Net.

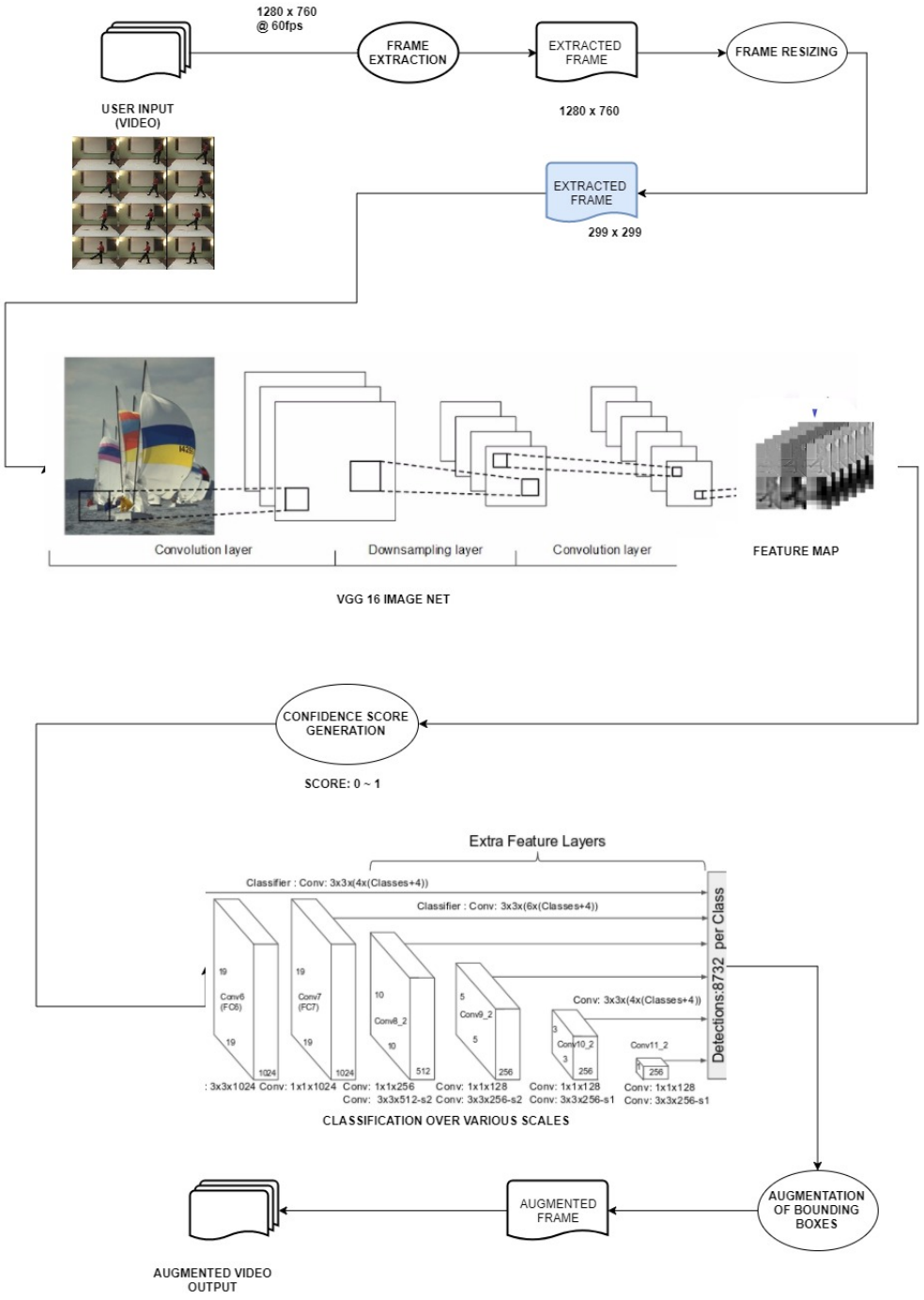


Figure 5.2: Detailed System Architecture

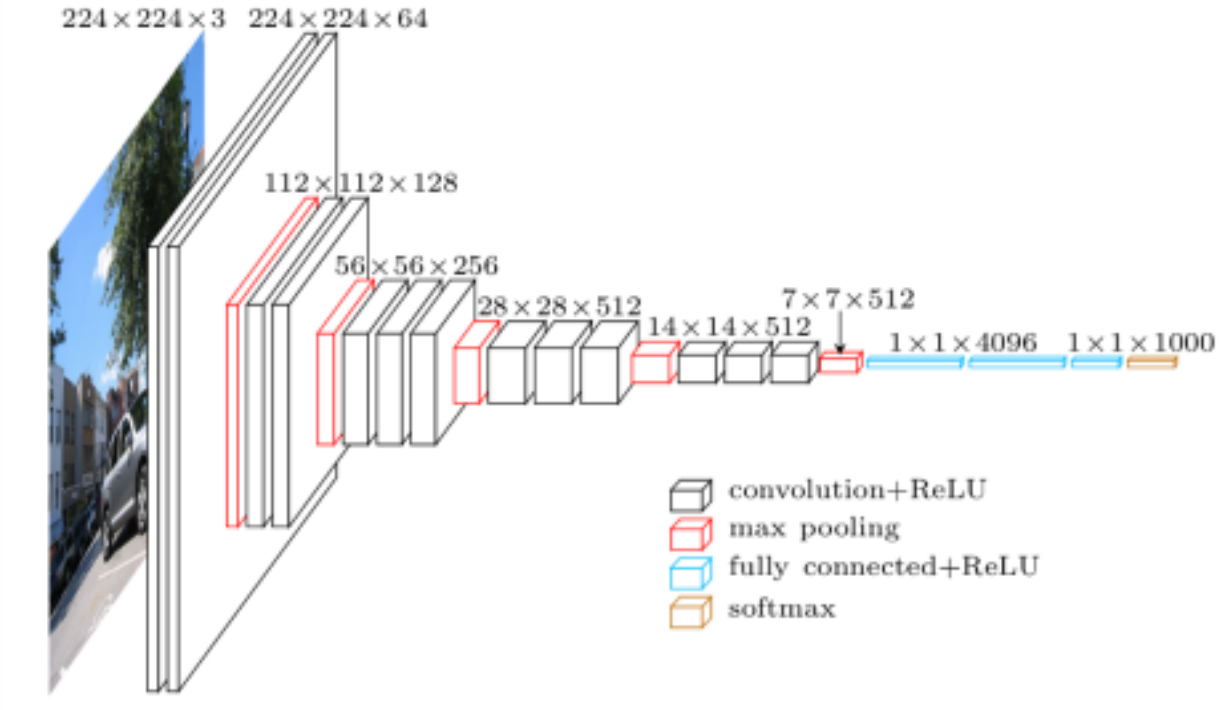


Figure 5.3: VGG 16 Net

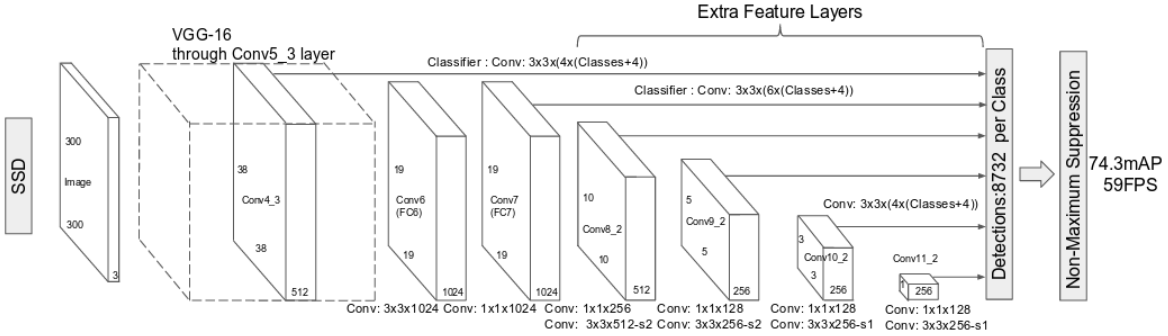


Figure 5.4: SSD

Chapter 6

Data Description

A Data Flow Diagram is a graphical representation of the ow of data through an information System, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can be later elaborated. DFD's can also be used for the visualization of data processing.

6.1 Data Flow Diagram

6.1.1 Level 0 DFD

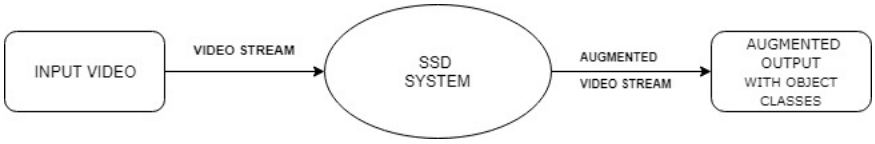


Figure 6.1: Level 0 DFD

6.1.2 Level 1 DFD

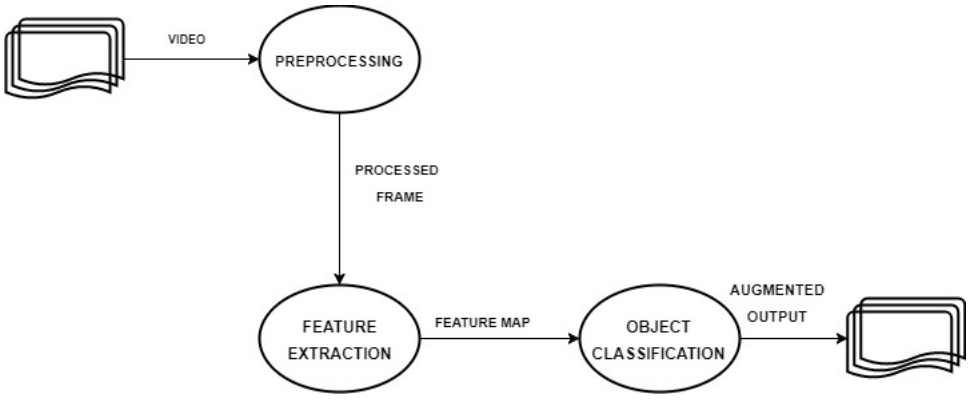


Figure 6.2: Level 1 DFD

6.1.3 Level 2 DFD

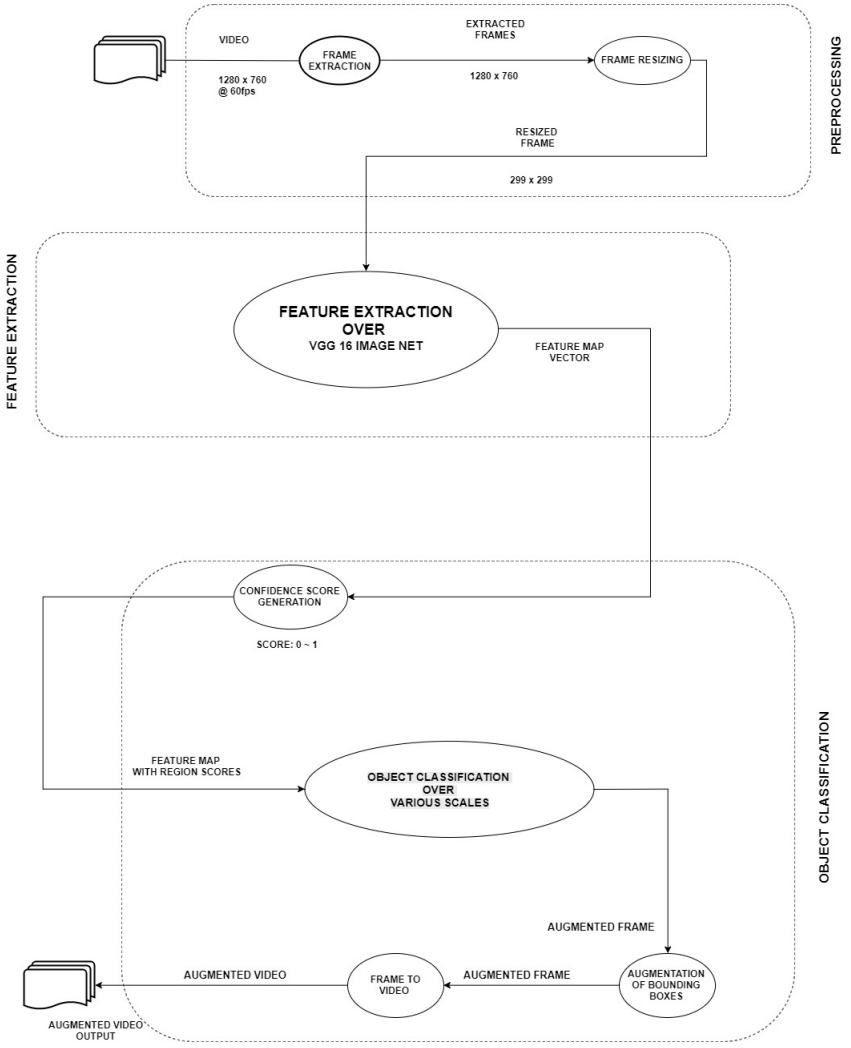


Figure 6.3: Level 2 DFD

6.2 Use Case Diagram

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. The use cases are nothing but the system functionalities written in an Organised manner.

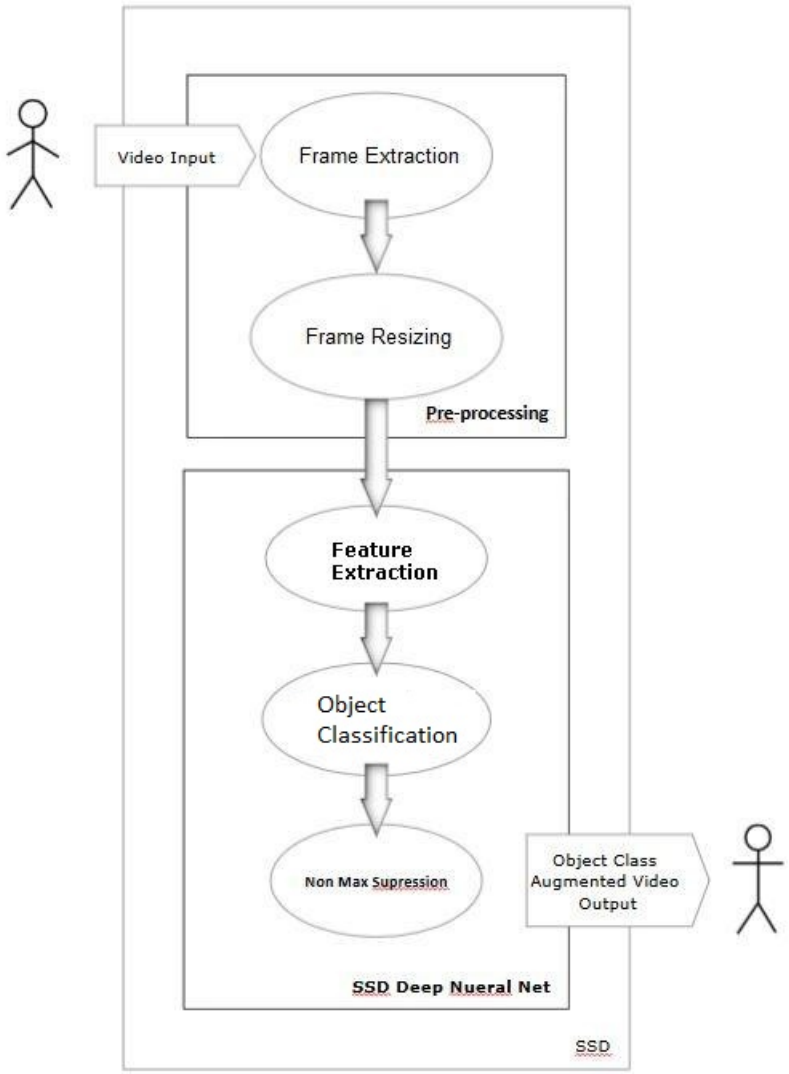


Figure 6.4: Use case diagram

6.3 Class diagram

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

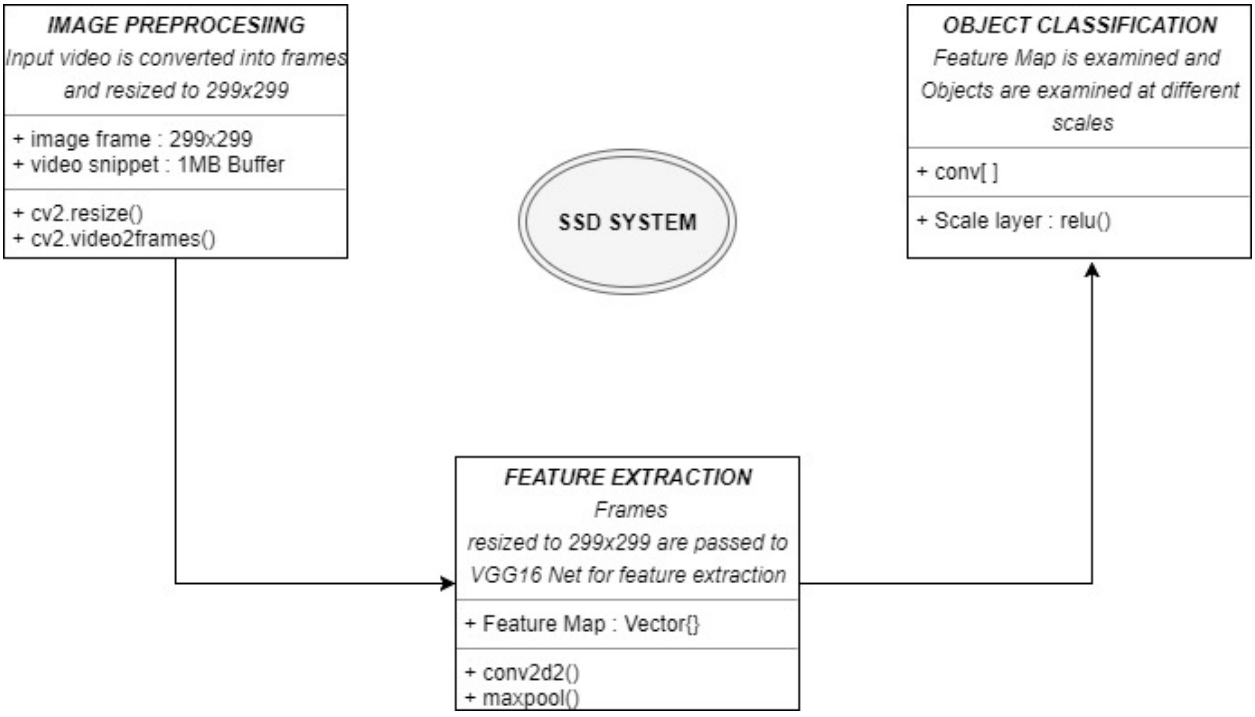
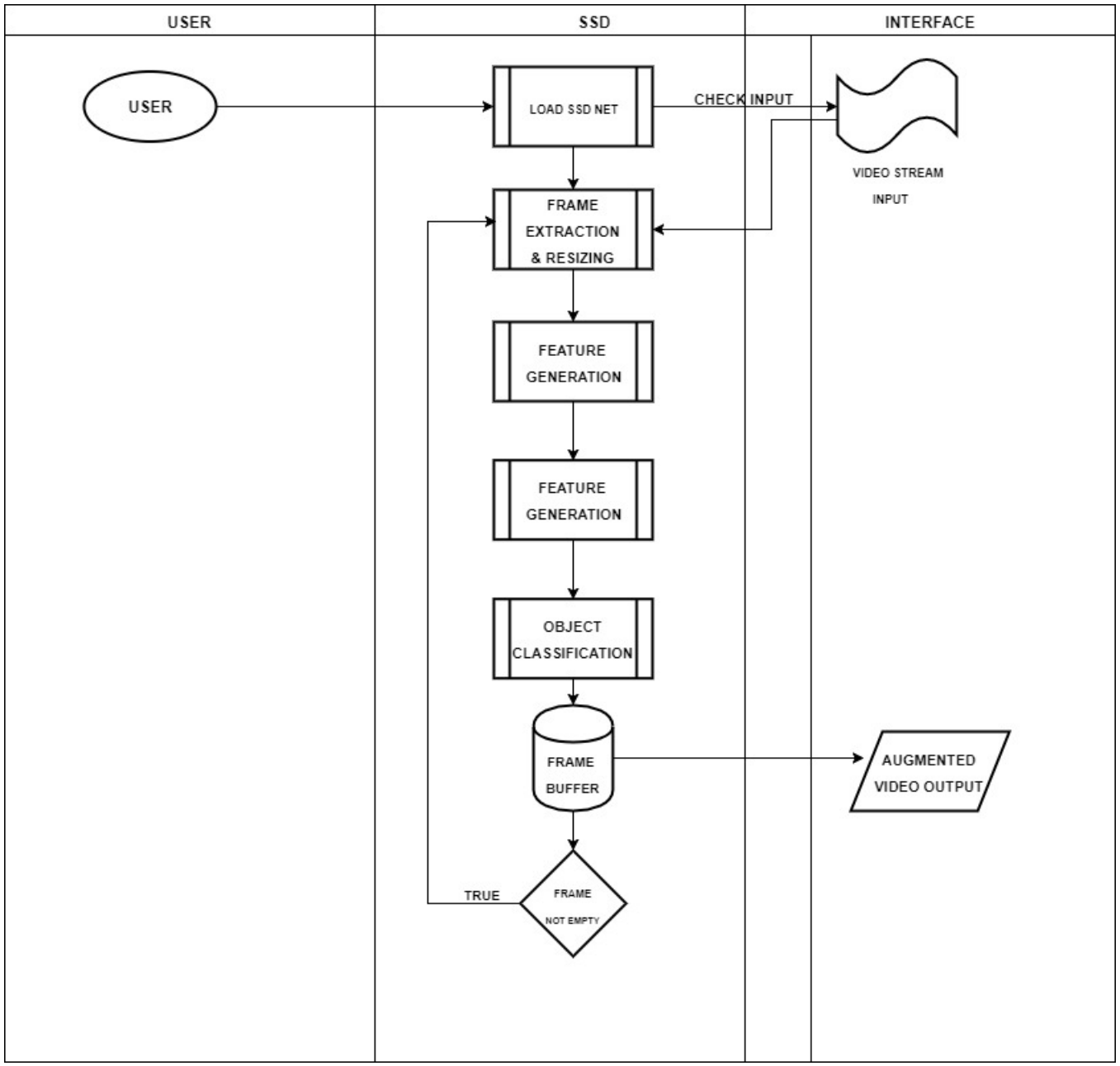


Figure 6.5: Class Diagram

6.4 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.


USER
SSD
INTERFACE
USER
LOAD SSD NET
CHECK INPUT
VIDEO STREAM INPUT
FRAME EXTRACTION & RESIZING
FEATURE GENERATION
FEATURE GENERATION
OBJECT CLASSIFICATION
FRAME BUFFER
AUGMENTED VIDEO OUTPUT
TRUE
FRAME NOT EMPTY

6.5 Dataset Design

Datasets play a very important (and sometimes underrated) role in research.One of the hardests problems to solve in deep learning has nothing to do with neural nets: its the problem of getting the right data in the right format.

DATASET ITEMS - Variables:

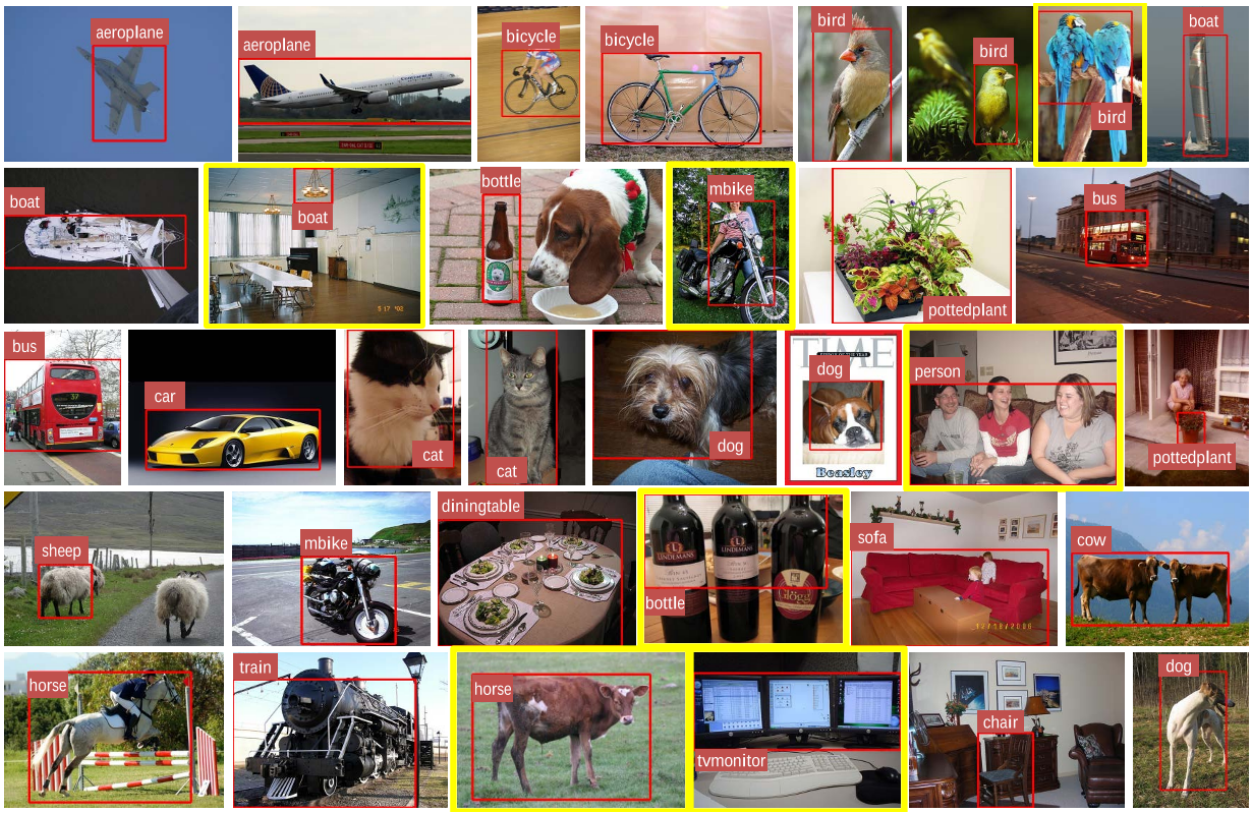
- Texture
- Color
- Contours
- Specific Features

6.5.1 Dataset Used - PASCAL VOC2012 :Visual Object Class

This dataset is a set of additional annotations for PASCAL VOC 2010. It goes beyond the original PASCAL semantic segmentation task by providing annotations for the whole scene. The statistics section has a full list of 400+ labels.Since the dataset is an annotation of PASCAL VOC 2010, it has the same statistics as those of the original dataset. Training and validation contains 10,103 images while testing contains 9,637 images.

We use VOC2012 trainval and VOC2007 trainval and test (21503 images) for training, and test on VOC2012 test (10991 images). We train the models with 1023 learning rate for 60k iterations, then 1354 for 20k iterations.

.


aeroplane
aeroplane
bicycle
bicycle
bird
bird
bird
boat
boat
boat
bottle
mbike
pottedplant
bus
bus
car
cat
cat
dog
dog
person
pottedplant
sheep
mbike
diningtable
bottle
sofa
cow
horse
train
horse
tvmonitor
chair
dog

.

Chapter 7

Implementation

7.1 Algorithms

7.1.1 Video Preprocessing

Algorithm 1 : Video To Frame Conversion

Algorithm for Conversion Of Video to Frames
Input : Video Input
Output: Video Frames Datastructure: Prioriy Queue Abstract Data Type as Buffer

1. Start
2. Import OpenCV as cv2
3. Input Video : set String variable video with "path to video"
4. Initialize Integer variable count as 0 for frame count
5. while(sucess) - loop until conversion failure
   - 5.1 Convert to frames : Using video2frames function in OpenCV cv2.video2frames(video,frame %d, count)
   - 5.2 Increment the Frame Count -count
6. end while
7. Stop

Algorithm 2 : Frame Resizing

Resizing of Frames to required Dimensions i.e 299x299px
Input : Video Frames
Output: Resized Video Frames of dimensions 299x299

1. Start
2. Import OpenCV as cv2
3. Set Dimension : Variable dim with required value for dimemsion, here 299
4. Input Frame as 'im' variable of type image
5. while(frame is valid)
   5.1 Set image variable Output as Cv2.resize(im,dim,interarea) using OpenCV function resize
6. end while
7. Stop

7.1.2 Feature Extraction - VGG Net

Algorithm 3 : Loading VGGNet 16 Model

Input : Stored VGG Model
Output: VGG Runtime Enviroment

1. Declare an Argument parser
2. Set Model in PyTorch as MODELS = VGG16
3. Ensure a valid model was supplied via command line argument if not
   3.1 raise AssertionError
   3.2 Exit
4. Initialize the input image shape as (224x224 pixels) along with the pre-processing function (this might need to be changed based on which model we use to classify our image)
5. Set input shape as inputShape = (224, 224)
6. Use image net preprocessing from the path imagenet_utils.preprocess_input (OPTIONAL)
7. Load the Model from Disk (VGG16 IS 575MB in Size)
8. pre-process the image using the appropriate function based on the model that has been loaded ie subtraction scaling etc (OPTIONAL WITH STEP 6)
9. Run the model over input to obtain Feature Map

7.1.3 Object Classification

Algorithm 4 : Neural Net Modelling

Set Neural Net Layers for Classification

1. Set Layer Architecture: Initialize Layer as net1 = NeuralNet Type of Layer Arguments
   - 1.1 Layer 1 Name:'input' Function: InputLayer
   - 1.2 Layer 2 Name:'conv2d1' Function: Conv2DLayer
   - 1.3 Layer 3 Name:'maxpool1'Function: MaxPool2DLayer
   - 1.4 Layer 4:Name:'conv2d2' Function: Conv2DLayer
   - 1.5 Layer 5:Name:'maxpool2' Function:MaxPool2DLayer
   - 1.6 Layer 6:Name: 'dropout1'Function:DropoutLayer
   - 1.7 Layer 7:Name'dense'Function:DenseLayer
2. Define Input Layer : Set input layer parameters as input_shape=(None, 1, 28, 28) :Set parameters for kernal (Here None) and kernal dimensions
3. Define Hidden Layers:
   - 3.1 Layer 1 layer conv2di:¡filter,size¿ :Set filter and filter size
   - 3.2 Layer 2 layer maxpooli: maxpool1_pool_size=(2, 2) :Set pooling area size
   - 3.3 Layer 3 dropout1 :dropout1_p=0.5 : Set threshold
   - 3.4 Layer 4 dropouti : ¡dropouti¿
4. Define Output Layer
   - 4.1 Set layer output with parameters output_nonlinearity and output_num_units

Algorithm 5 : Augmentation

Augmentation Algorithm
Input : Object Classes along with Bounding Box Coordinates
Output: Augmented Video Frames

1. Get Bounding Box coordinates (center(x,y),width,height) from BBOX Utility
2. Draw Boxes accoeding to BBOX Coordinates
3. Augment over Frames
4. Recorrect augmented boxes per frame using Non-max Supression.

7.2 Development Tools

- **Sublime Text:**
  Sublime Text is a proprietary cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses.Sublime Text has a powerful, Python API that allows plugins to augment built-in functionality. Package Control can be installed via the command palette, providing simple access to thousands of packages built by the community.

  Some of the features of Sublime Text are

  - "Goto Anything,"quick navigation to files, symbols, or lines
  - "Command palette" uses adaptive matching for quick keyboard invocation of arbitrary commands
  - Simultaneous editing: simultaneously make the same interactive changes to multiple selected areas
  - Python-based plugin API
  - Project-specific preferences
  - Cross platform (Windows, macOS, and Linux)

- **QT Creator:**
  Qt Creator is a cross-platform C++, JavaScript and QML integrated development environment which is part of the SDK for the Qt GUI application development framework. It includes a visual debugger and an integrated GUI layout and forms designer. The editor's features include syntax highlighting and autocompletion. Qt Creator uses the C++ compiler from the GNU Compiler Collection on Linux and FreeBSD. On Windows it can use MinGW or MSVC with the default install and can also use Microsoft Console Debugger when compiled from source code. Clang is also supported.

  Qt Creator includes a code editor and integrates Qt Designer for designing and building graphical user interfaces (GUIs) from Qt widgets.

  The code editor in Qt Creator supports syntax highlighting for various languages. In addition to that, the code editor can parse code in C++ and QML languages and as a result code completion, context-sensitive help, semantic navigation are provided.

  Qt Designer is a tool for designing and building graphical user interfaces (GUIs) from Qt widgets. It is possible to compose and customize the widgets or dialogs and test them using different styles and resolutions directly in the editor. Widgets and forms created with Qt Designer are integrated with programmed code, using the Qt signals and slots mechanism.

  Qt Quick Designer is a tool for developing animations by using a declarative programming language QML.

- **PyCharm:**
  PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.
  PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition released under a proprietary license - this has extra features.
  Features include : Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes. Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages. Python refactoring: including rename, extract method, introduce variable, introduce constant, pull up, push down and others support for web frameworks: Django, web2py and Flask Integrated Python debugger, Integrated unit testing, with line-by-line code coverage. Google App Engine Python development. Version control integration : unified user interface for Mercurial, Git, Subversion, Perforce and CVS with changelists and merge.

- **Google Slides:**
  Google Slides is an online presentation app that lets you create and format presentations and work with other people. It is part of a free, web-based software office suite offered by Google within its Google Drive service. Edits are tracked by user with a revision history presenting changes.

  An editor's position is highlighted with an editor-specific color and cursor. A permissions system regulates what users can do. Updates have introduced features using machine learning, including "Explore", offering search results based on the contents of a document, answers based on natural language questions in a spreadsheet, and dynamic design suggestions based on contents of a slideshow, and "Action items", allowing users to assign tasks to other users.

  In order to view and edit documents, spreadsheets and presentations offline on a computer, users need to be using the Google Chrome web browser. A Chrome extension, Google Docs Offline, allows users to enable offline support for Docs, Sheets and Slides files on the Google Drive website. The Android and iOS apps natively support offline editing.

  Presentation files converted to .gslides Slides format cannot be larger than 100 MB. Images inserted cannot be larger than 50 MB, and must be in either .jpg, .png, or non-animated .gif formats.

- **Atom:**
  Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained. Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. It is written in CoffeeScript and Less. It can also

be used as an integrated development environment (IDE). Atom was released from beta, as version 1.0, on 25 June 2015. Its developers call it a "hackable text editor for the 21st Century".

Using the default plugins, the following programming languages are supported in some aspect as of v1.5.1:

C/C++, C, Clojure, CSS, CoffeeScript, GitHub Flavored Markdown, Go, Git, HTML, JavaScript, Java, JSON, Julia, Less, Make, Mustache, Objective-C, PHP, Perl, Property List (Apple), Python, Ruby on Rails, Ruby, Sass, Shell script, Scala, SQL, TOML, XML, YAML

- **Spyder:**
  Spyder (formerly Pydee) is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates NumPy, SciPy, Matplotlib and IPython, as well as other open source software. It is released under the MIT license.

  Spyder is extensible with plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows with WinPython[9] and Python (x,y), on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.

  Spyder makes use of Qt either through the binding PyQt or PySide.This flexibility is reached through a small abstraction layer called QtPy.

Chapter 8

Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs, and to verify that the software product is fit for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs
- performs its functions within an acceptable time
- is sufficiently usable
- can be installed and run in its intended environments, and
- achieves the general result its stakeholders desire.

8.1 Testing Methodologies

Black Box Testing: also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors

- Initialization and termination errors

White Box Testing: is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential.

8.2 Unit Testing

In the unit testing phase, we tested the separate modules of the software. The white box testing was carried out where each module or component of the software was tested individually. The software systems that make up the system are the modules and the routines, which are assembled and integrated to perform a specific function in a large system. Many modules at different levels are needed. Unit testing, independent of one another focuses on modules to locate errors. This enables the tester to detect errors in coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided. Unit testing compromises of the set of tests performed prior to integration of the unit into the entire project.

8.2.1 Preprocessing

1. Test Case: Input a video.
2. Verdict: Resized frames are obtained.

8.2.2 Feature Extraction

1. Test Case: Image frames with only moving objects.
2. Verdict: Feature map of the objects are obtained.

8.2.3 Object Classification

1. Test Case: Input feature map.
2. Verdict: Objects classified and augmented using bounding boxes.

8.3 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing . Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. Here, we combined the various modules that have been unit tested and tested them as a whole. The system accepts a video as it's input and verifies the user according to the previous data present in the database.

1. Top Down Integration This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structuring either a depth first or breadth first manner.

2. Bottom Up Integration This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up processing required for modules subordinate to a given level is always available and the need for steps are eliminated.

8.4 System Testing

The implementation of a computer based system requires the test data to be prepared and that the system and its element be tested in a planned or structured manner. The computer program components is a major sub-system of the computer based information system and particular attention should be given to the testing of the system as it is developed.

Chapter 9

Graphical User Interface

9.1 GUI Overview

The user interface was made using Qt Creator.

9.2 Main GUI Components

The graphical user interface (GUI) was created using Qt Creator. Application contains two sections for displaying the input and the processed output simultaneously and has an option to input video from the users directory. It also has an execute button upon clicking which the process runs in background and after the completion of process the output is displayed.

Chapter 10

Results

Include screenshots of the project.

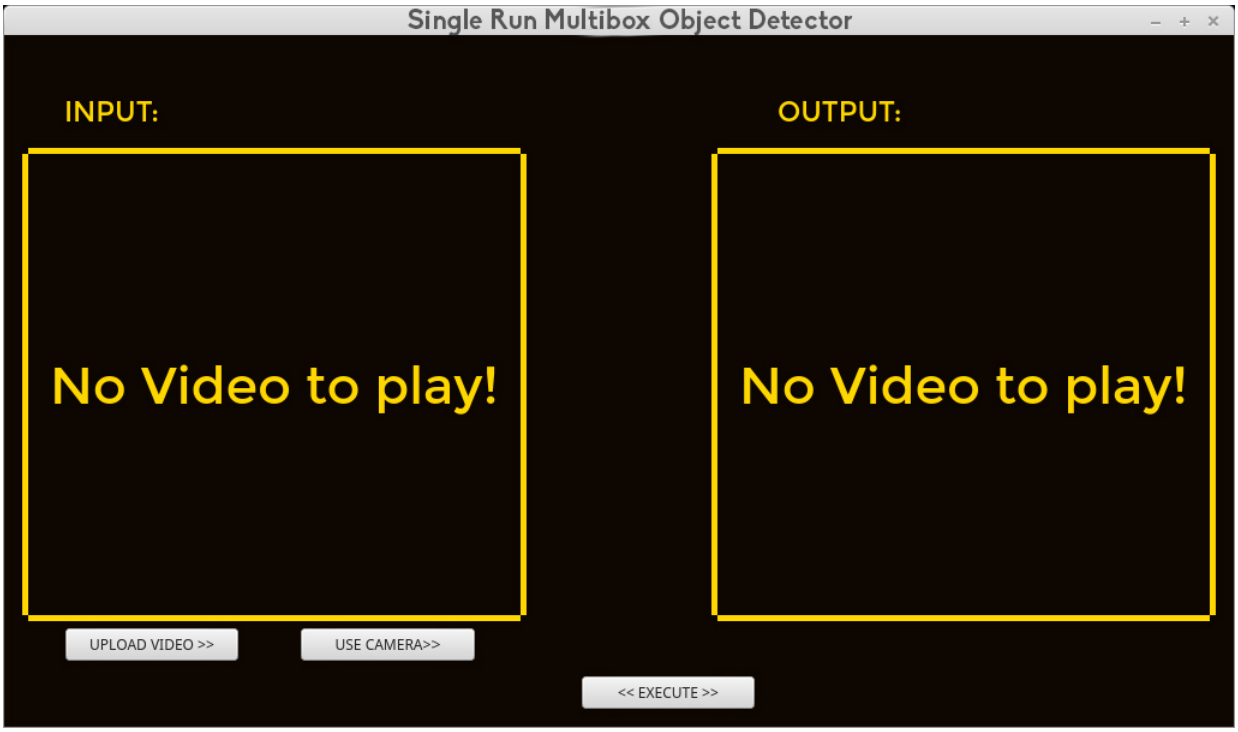


Figure 10.1: Screenshot 1

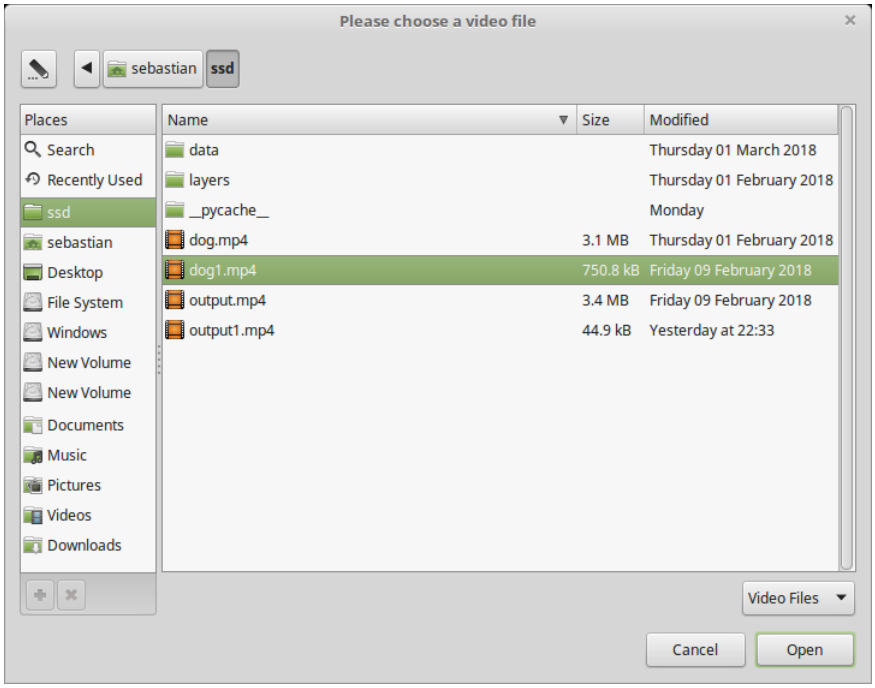


Figure 10.2: Screenshot 2

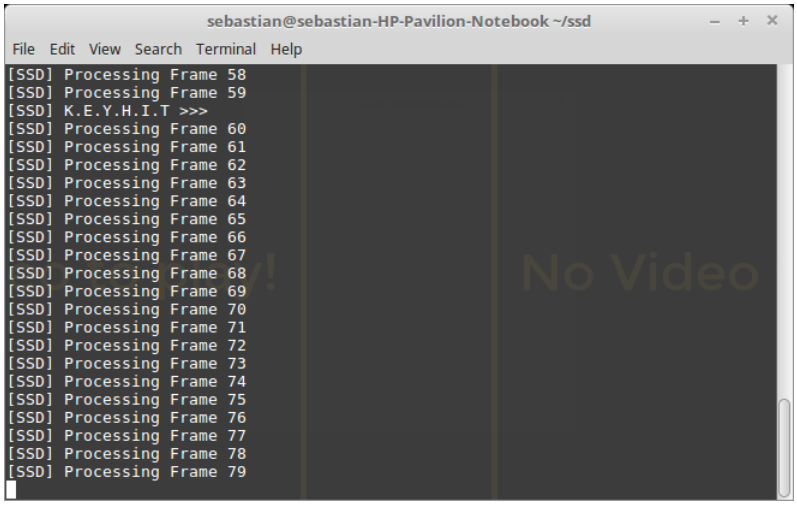


Figure 10.3: Screenshot 3

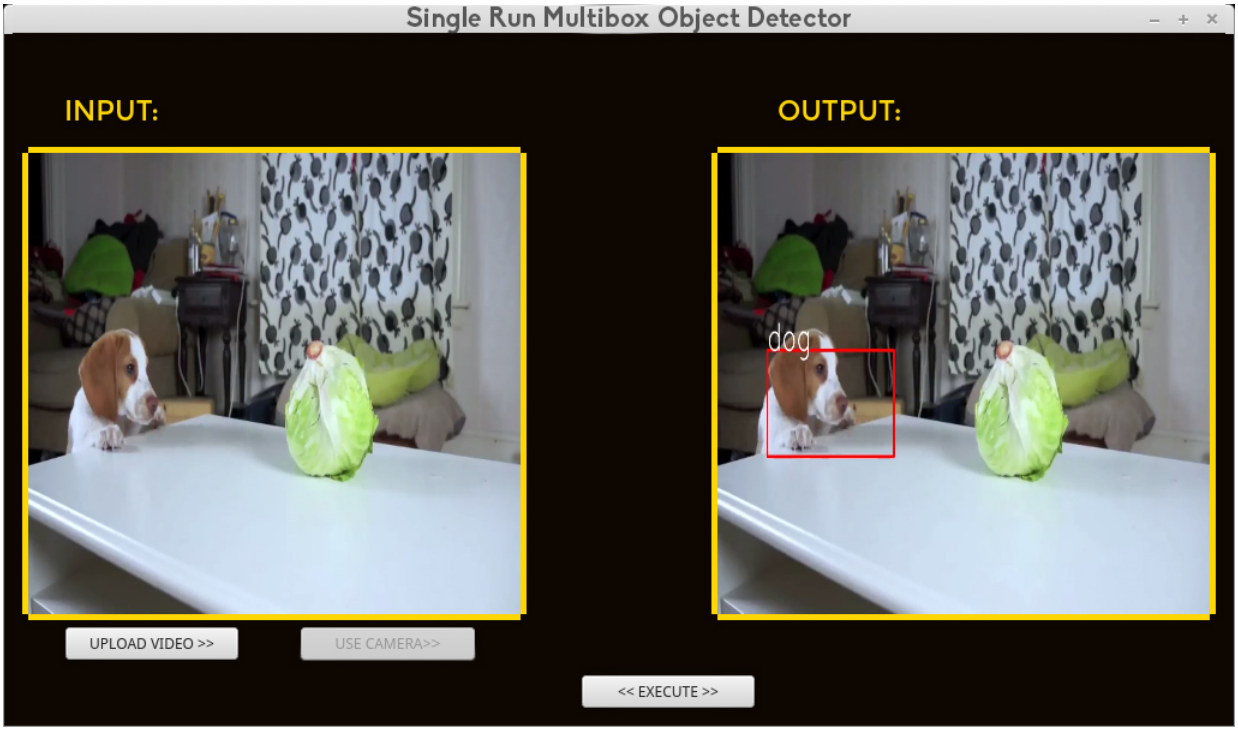


Figure 10.4: Screenshot 4

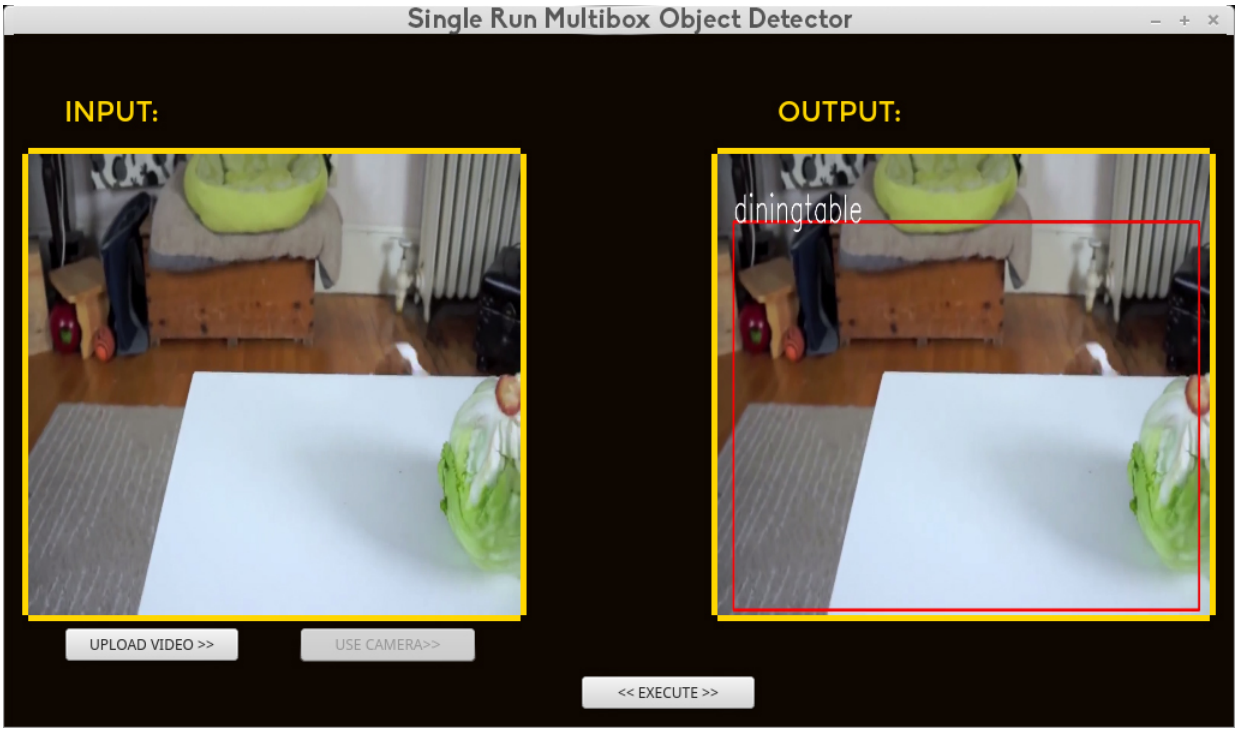


Figure 10.5: Screenshot 5

Chapter 11

Conclusion

Single Run Multi Box Object Detection Using a single deep neural network is a project capable of processing input video and classifies the objects in the input video real-time with reduced latency and more accuracy.Object detection has many practical applications and is currently being used in many promising and well publicized technologies that have caught the public imagination including but not limited to self-driving cars , and detecting people for purposes of security. Recognition and detection are also useful in diagnosis using medical imaging, and various scientific imaging applications Even though at present Library Dependencies make the module inviable for embedded applications and the Hardware Requirements are quite high, SDNN is able to provide fast detection with acceptable Mean Averege Precison.

We came to a conclusion that the output of our single run multi box object detector is acceptable self driving cars which needs faster processing of the video. But the major disadvantage of our technique is that it requires GPU with multiple CUDA cores and higher memory bandwidth.

Chapter 12

Future Scope

A Single Deep Nueral Net with Real-time Object Detection can be used in self driving cars. It can also be used for surveillance. It can be used in recognizing suspects at large. Deep learning generally benefits from big data training. Hence the proposed model can be improved by using a large training set that consists of 395,909 images from ILSVRC 2013 ImageNet. The system should be able to cope with different upscaling factors. Number of different CNNs used for comparison can be higher.

References

[1] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "Object detection using deep neural network"in International conference on Image processing,2014.

[2] Ji-jian Hou, Ran Ji, Cui Qin, Yu Yang, Chao-xin Wang and Zhe-long Wang, A System for object detection in International Conference on Image processing, 2012.

[3] SSD: Single Shot MultiBox Detector
Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. CCV 2016 - Computer Vision and Pattern Recognition (cs.CV)

[4] DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In Con-ference onter Vision and Pattern Recognition (CVPR).

[5] Neil: Never ending image learner, 2014.

[6] B. Berlin. Basic color terms: Their universality and evolution. Univ of California Press, 1991.

[7] R. A. Brooks, R. Creiner, and T. O. Binford. The acronym model-based vision system. In Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'79, pages 105113, San Francisco, CA, USA, 1979. Morgan Kaufmann Publishers Inc.