Project Report

on

Object Detection (Modified YOLO)

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

Session 2018-19
in

Computer Science And Engineering

By
Eshan Pandey
1503210080

Deepak Kapil
1503210071

Under the guidance of

Prof. (Dr.) Shailesh Tiwari

ABES ENGINEERING COLLEGE, GHAZIABAD

AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)

Project Report

on

Object Detection (Modified YOLO)

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

Session 2018-19
in

Computer Science And Engineering

By
Eshan Pandey
1503210080

Deepak Kapil
1503210071

Under the guidance of

Prof. (Dr.) Shailesh Tiwari

ABES ENGINEERING COLLEGE, GHAZIABAD

AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)

STUDENT'S DECLARATION

We hereby declare that the work being presented in this report entitled "OBJECT DETECTION (MODIFIED YOLO)" is an authentic record of our own work carried out under the supervision of Dr. SHAILESH TIWARI
The matter embodied in this report has not been submitted by us for the award of any other degree.

Dated: **Signature of students(s)**

(Name(s).....)

Department:

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Signature of HOD
Prof. (Dr.) Shailesh Tiwari
Computer Science & Engineering Department
Date.....

Signature of Supervisor
Prof. (Dr.) Shailesh Tiwari
Head of Department
Computer Science & Engineering Department

CERTIFICATE

This is to certify that Project Report entitled “OBJECT DETECTION (MODIFIED YOLO)” which is submitted by Eshan Pandey, Deepak Kapil and Abhishek Agrawal in partial fulfillment of the requirements for the award of degree B. Tech. in Department of Computer Science and Engineering of U. P. Technical University, is a record of the candidate's own work carried out by them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Supervisor

Date

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe a special debt of gratitude to Professor (Dr.) Shailesh Tiwari, Department of Computer Science & Engineering, ABES Engineering College, Ghaziabad for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant effort that our endeavors have seen the light of the day.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:

Name :

Roll No.:

Date :

Signature:

Name :

Roll No.:

Date :

ABSTRACT

We take inspiration from convolution operation and present neighbourhood block subtraction technique. Convolution is the weighted summation of neighbourhood, used to extract certain specific features in image (depending the convolutional kernel), our neighbourhood block subtraction enhances the corner and borders in the image while suppressing the background noise in the image.

We train YOLOv3 as Modified YOLOv3 with neighbourhood block subtraction on PASCAL VOC 2007 train and val data and 2012 train data. We test our model on PASCAL VOC 2007 test data. Our model archives 73% mAP with 0.25 threshold IOU and 64.02% mAP with threshold 0.5 IOU. When compared with the original implementation of YOLOv3 trained on MS COCO our modified YOLOv3 archives 3.4% higher mAP with negligible difference in fps. Our modified YOLOv3 out performs almost all the entries of the Pascal VOC 2007 challenge. In 12 of the 20 classes our model exceeds the best entry the 2007 challenge.

We believe that neighbourhood block subtraction has the potential to be used as a feature detector in classification related tasks.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF SYMBOLS

$\sum$	Summation
$\int$	Integration
=	Equality
*	Convolution Operation
x	Multiplication

LIST OF ABBREVIATIONS

NN	Neural Network
CNN	Convolutional Neural Networks
YOLO	You Only Look Once
SSD	Single Shot Multibox Detector
GPU	Graphics Processing Unit
CPU	Central Processing Unit
VOC	Visual Object Challenge
BB	Bounding Box
IOU	Intersection Over Union
mAP	mean average precision
OCR	Optical Character Recognition

CHAPTER 1

INTRODUCTION

Vision and perception comes naturally to us humans. We never think how hard can visual intelligence be. It was up until mid 20th century that vision and perception was not considered to be an intelligent task. Vision and perception was considered an easy approach as compared to reasoning and planning. Back than artificial intelligence back than was mostly a theory. With further advancements in artificial intelligence, when attempts were made, in mid 20th century to build AI with reasoning and planning and AI with vision and perception, it was realised that vision and perception can be often harder than reasoning (may be not planning!). We have a very limited understanding of human/animal visual perception abilities. It is so, we believe the artificial visual intelligence is very hard. A more in depth understanding of human/visual perception system may help us build more efficient systems with much more ease. ANN was an approach in this direction. Invented by Geoffrey Hinton it was build to loosley mimic the human brain. ANN was a very successful approach in artificial intelligence. With backpropagation (learning, error and correction method) [15], ANN produced very impressive results to every aspect of artificial intelligence. However it was not considered a correct path to artificial general intelligence. The primary reason was hardware limitations. It was in mid 2000s that the problem of hardware was overcome with very powerful GPUs. Since then we have seen the fastest growth in the field of AI. CNN was invented by the student of Geoffrey Hinton, Yann Lecun. But, can artificial general intelligence be solved with neural networks? May be, but we still do not understand a lot about intelligence and intelligent models, so may be not. It is very difficult to answer this question precision. So why this approach of neural networks? Because this produces results. We have seen results with CNN that were not possible to imagine a decade ago. Facial recognition, image classification, a single model to classify and label upto 1000, synthetic data generation with GANs [14]. As of now we believe this a direction of study which needs to be extensively explored.

Our B.Tech. project and report is one such attempt to explore this vast and extensively growing field of computational intelligence. We had some prior experience in this field and wanted to explore this even further. We primarily studied image classification and object detection. We present our findings and results in this report.

1.1 Problem Introduction

Computer Vision is a field in the Computer Science that deals with the question that how a machine can be made to understand a scene (a digital image). By understanding it is meant to extract various meaningful features from image. These meaningful features can be used to extract more complex objects, like car, dog, cat etc. The purpose of Computer Vision is to automate the work that human visual system can perform and more. Vision in the computer system is the ability to perceive, interpret, understand, analyze and draw conclusions from the digital image/optical data. In the early 1970s attempts were being made in the field of computer vision. Back than vision was considered an easy task as compared to reasoning and planning. However that was not the case for so long. It is important to understand why Computer Vision is difficult as compared to classical reasoning based algorithms. While we try to mimic the human visual perception ability, our methods are mostly different. Human approach to vision is primarily cognitive but that of a machine (computer) is highly mathematical. In this project we will be dealing with deep learning based models for object detection. Object detection is a computer technology that deals with recognition and localization of the object class in an image.

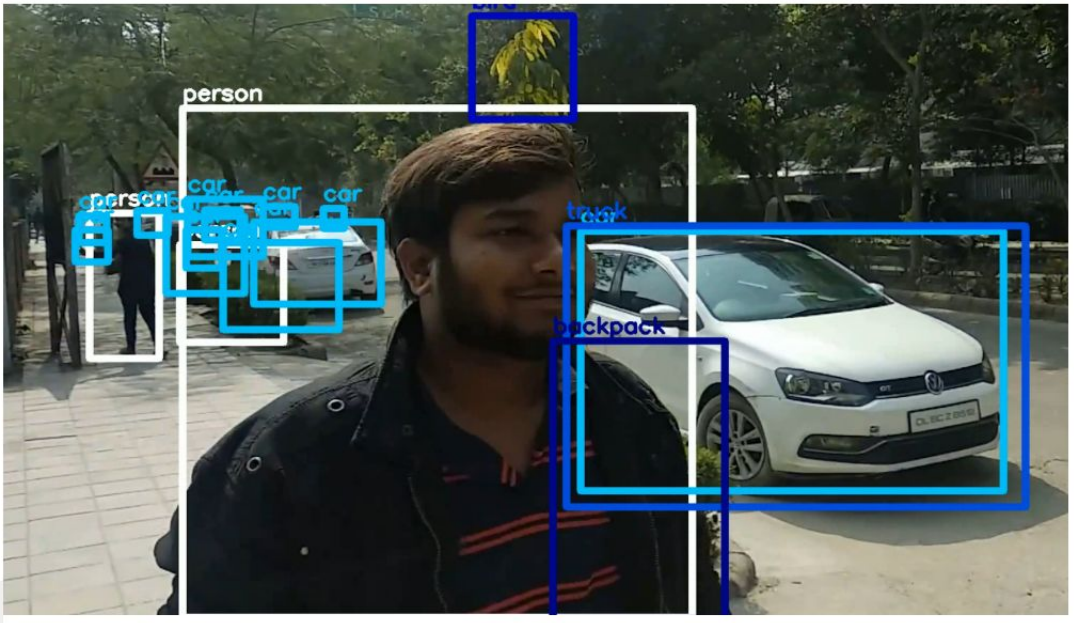

Fig 1.1: Example of object detection. The detection model recognises person and car in the image.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for

squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

1.1.1 Motivation

A huge amount of data today is in the form of images and videos. These data can provide a lot of insight about almost any real life situation. However this is not yet a possibility. Classical approaches like Image Processing methods can not be scaled and generalised, they are mostly context specific, simple to execute but exceptionally difficult to derive. Contemporary methods of Computer Vision especially Deep Learning models are held back due to lack of hardware requirements. All this points to the fact that something needs to be done in order to achieve the automation of visual tasks. Thus we work in the direction to make visual automation more feasible for real life applications.

1.1.2 Project Objective

Our objective is to improve the single unified CNN architecture for object detection. Our proposed model improve the stability of the state of the art object detection architectures like and YOLO[2,3,4]. It has been observed that very accurate models tends to slow down and therefore cannot be used in real time. On the other hand very fast model are prone to errors. Our objective is to provide a better mAP without significant decrease in speed, so that object detection can be carried out in real time with a considerable degree of accuracy.

1.1.3 Scope of the Project

Automation of visual application is a challenging task, however the rewards are significant. Following are some of the applications of visual automation

1. Image/Video data analysis.
2. Assistance of visually challenged people.
3. In development of level 3 autonomous cars.
4. In development of level 4 & 5 autonomous cars with additional technologies.
5. Face recognition.
6. Image pattern recognition.
7. Detection and regulation of hate content in image/video data over the internet.

8. Image to text data (extracting texts from images).
9. Image/Feature reconstruction.
10. Image synthesis.

1.2 Related Previous Work

People have made efforts towards real time computer vision algorithms from early 1990s. However most of the approaches were not very general in their approaches, each situation required a specific set of feature extractor which just could not be used in other situations. These set of feature extractors required years of human effort. It was from late 2000s that the computer vision community started to be very general in the approach, the set of feature extractors were no longer hand picked, instead the models learned. VGG16 and VGG19 are the two very popular feature extractors, used in a wide variety of image classification problems (Visual objects classification, document classification, etc.) The original implementation of VGG models were trained the imagenet dataset for thousand class classification [11,14], LeNet, U-Net, AlexNet, RetinaNet, ResNet, ResNeXt are some popular image classification architectures, generally used as feature extractor.

Some real time approaches for object detection were made after the development in classification models like VGG16. YOLO [2] and SSD [1] are the earliest and best unified real time detection architectures.

1.3 Organisation of the Report

In chapter 2 we talk about literature survey. We study various state of the algorithms for object detection and image classification. To name some, VGG16 and VGG19, R-CNN, SSD and YOLO.

In chapter 3 we discuss system design and methodology, all the changes that we made to the original implementation of YOLO. We discuss neighbourhood block subtraction technique for feature extraction and background noise suppression.

In chapter 4 we discuss implementation and results. All that we tried, and how did we do it. And how the results can be reproduced. (We also added the weight file of the trained network in the CD to exactly reproduce the results)

And finally we talk about what else we wanted to try but could not due to time constraints. Those are our future plans and direction of work.

CHAPTER 2

LITERATURE SURVEY

2.1 Digital Image

A digital image a numeric representation of the image. Bitmap is unarguably the most simple digital storage mechanism. Bitmap is a very simple black and white image stored as a two dimensional array of bits(ones and zeros). Each bit represent one pixel of the image. Bit set as zero represents black and that set to one represents white.

With enough space allocation in the memory to store a real number the pixels of the image could be arbitrary grey tones. The arbitrary levels of grey could be represented as a two dimensional array of real numbers, say between 0 and 1, with pixel color varying smoothly from black at 0.0 through mid-grey at 0.5 to white at 1.0. But this mechanism was considered highly inefficient. The floating point number would take up 32 times more storage than simple bitmap image.

Thus, an alternative efficient mechanism was developed, Pixmap. Instead of using floating point number for grey level representation which would take up 32bits for each pixel, integer values were used. The integer values would range from 0 to 255, taking up 8 bits. Now, pixel color would varying smoothly from black at 0 through mid-grey at 127 or 128 to white at 255.

The grey scale image was not enough. With loss of color a lot of visual information was dropped. Thus RGB scheme was introduced. In the early nineteenth century Thomas Young and Hermann Helmholtz proposed Youbg_helmhotz theory of trichromatic color vision.Most of the color (almost all) that a human eye can perceive can be reconstructed with the combination of the primary colors, RED, GREEN and BLUE. The level of intensity of each color in the combination determines the color we see. The RGB image is a 3D data structure with shape (Height X Width X Channels). Each pixel represented as (1 X 1 X Channels). Each pixel in the image is combination of the integer values representing the three colors and is scaled between 0 to 255.

2.2 Haar Based Cascade Classifiers

(Viola and Jones,2001) Introduced a breakthrough concept of Haar feature based cascade classifiers [5]. Object Detection using Haar feature-based cascade classifiers is an effective object detection technique. The cascade function is trained on huge data set of positive and negative images. The technique can be used to detect any class. However, the paper describes detection technique for

human faces. One of the drawbacks of the approach is that cascade function can not be generalised, meaning that the function needs to be trained for accordingly for every situation. Once trained cascade function can only be used for the same class.

The algorithm requires many positive images (images with the target class, like human face) and many negative images(images without the target class) Than features are extracted from the training data set.

2.3 Convolution Operation

Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel.
Convolution is a mathematical operation on two real valued functions, defined as

$$s(t) = \int x(a)w(t - a)da. \quad (2.1)$$

The first argument x is referred to as input, and the second argument is referred to as kernel and the result is referred to as feature map.

The convolution operation is generally denoted with $*$,

$$s(t) = (x*w)(t) \quad (2.2)$$

When dealing with data on the computer it more realistic to assume that the data is provided in discrete time intervals, and t can take only integer values. Assuming that *x* and *w* are only defined over integer values of *t*, the discrete convolution can be defined as

$$s(t) = (x * w)\ (t) = \sum_{a=-\infty}^{\infty} x(a)\ w(t-a) \quad (2.3)$$

In computer vision the input data (grey scale image) is two dimensional (Note that an RGB image has three channels and each channel may be represented separately or as average of RGB intensities, as the need may be, and the resulting image will be two dimensional). Thus, for two dimensional image, two dimensional kernel is used as

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)\ K(i - m, j - n) \quad (2.4)$$

One of the most important properties of convolution is its commutative nature.

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)\, K(i - m, j - n)$$

$$= (K * I)(i, j) = \sum_m \sum_n K(i - m, j - n)\, I(m, n) \quad (2.5)$$

Convolution kernels are used as feature extractors in the convolution neural networks.

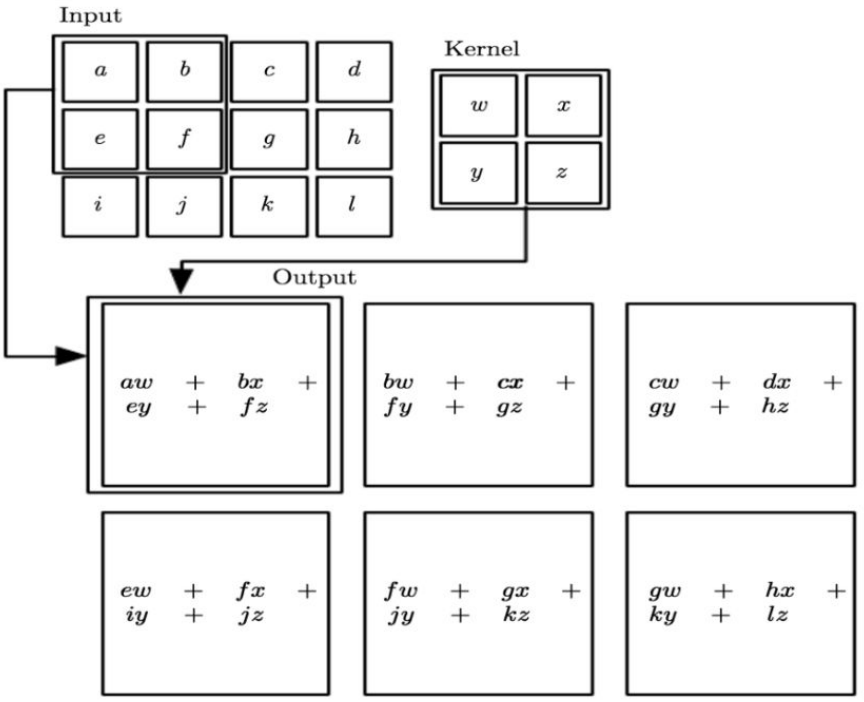

Figure 2.1: [5] The figure demonstrates the convolution operation. The convolution operation can also be understood as a weighted sum of the neighbourhood. [9]

2.4 Classical Approaches

Edge detection is one of the classical approaches in image processing and computer vision. Edge detection is very widely used in feature detection and feature extraction [8]. Edges in a digital image can be detected by detecting the set of points where image brightness changes sharply. The sharpe change in brightness can be referred to as discontinuity. Detection of this discontinuity is edge detection. Discontinuity in the depth, surface orientation, change in material property and variation in scene illumination may result in discontinues in the image brightness.[6][7].

Other classical approaches include corner detection, blob detection, ridge detection, affine invariant feature detection and KNN for classification.

2.5 Deep Neural Network and Convolutional Neural Network

2.5.1 Feed Forward Network

Feedforward Neural Network also known as multilayer perceptron model are the backbone of deep learning models. The goal of the feedforward network is to determine some f* which maps input x to output y. A feedforward network defines a mapping $y = f(x;\theta)$ and learns the value of the parameter θ to that result in the best function approximation. [9]

Feedforward neural network is called feedforward because of the direction of the information flow, i.e, the direction of flow of information is in the forward direction. Feedforward neural network is called network because it is a composition of many functions. The chain structure is very common in the feedforward network and can be generalized as

$f(x) = f^{n}(f^{n-1}(f^{n-2}(...f^{1}(x))))$, f^{1} is the first layer, f^{2} is the second layer and so on. The length of the chain is the depth of the model, thus the name deep learning. The first layer of the network is the input layer, the final layer of the model is the output layer, the layers in between are the hidden layer. Feed forward neural network is called neural because the concept is loosely inspired by neuroscience.

2.5.2 Convolutional Neural Networks (CNNs)

A convolution layer is a set of convolution kernels. A convolution layer takes the input and perform the convolution operations on the input. The number of output signal (in our case single channel 2 dimensional image) is equal to the number of convolutional kernels. Any neural network with at least one convolutional layer is called the convolutional neural network. Convolutional neural networks was popularised by Lecun,1995 [16] . when he open source the MNIST handwritten digits dataset [10] [15] .

It has been proven that convolution layer can successfully replace the dense layer in the network. This result is very import as the convolution layer may have anywhere between (3x3x128) to (3x3x1024) parameters but a fully connected layer may have 1000x1000 parameters. This difference is quite noticeable in the execution time(while training and while predicting). An architecture that heavily relies on multiple convolutional layers and less on dense (fully connected) layer is

multiple times faster the architecture that heavily relies on the dense layer. Include VGG16 as example. Compare the parameters in 14 convolution layer and two dense layers, 14 CNN layers have 4 Million parameter and 2 dense layer have over 100+ Million parameters. Almost all of the state of the art classification and detection architecture heavily rely on convolutional layer. The initial deep learning models however made use of dense layer(YOLOv1). But the more recent models completely eliminate the the dense layer (SSD [1] , YOLOv2 [3] and YOLOv3 [4]). SSD, despite of having VGG16 as the backbone for the classification, convert the last two layers from fully connected to convolutional, making the entire model consisting of convolution, pooling and activation layer. (No dense/fully connected layer)

2.5.3 Image Classification Using Deep Learning

One of the tasks that deep neural networks, especially deep convolutional neural networks are best at is image classification. CNN modes have outperformed almost all traditional approaches to image classification, segmentation and object detection. While the traditional approaches heavily rely on manual feature selection and detection.

The deep learning models despite being considerably large are much faster than the traditional approaches [10][15]. This is largely due to the fact, the deep learning models are optimized to run on a GPU. The modern day Pscal, Maxwell and Volta architecture GPU have more than 3000 specialised cores as compared to CPU with general purpose 2 to 16 cores. By minimising the communication between CPU and GPU and utilizing the parallel processing capability of GPU the contemporary deep learning models outperform the traditional approaches in terms of speed as well as accuracy.

Several deep learning architecture have been proposed since 2005 Mostly after Yann LeCun's CNN model for MNIST. CNN models can be used for a variety of classification tasks(we restricted only to image classification), like general purpose image classification, document classification, OCR(optical character recognition). Some of the most popular architectures for general purpose image classification are VGG16, VGG19, InceptionNet, MobileNet.

2.6 VGG19

VGG19 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5

test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPUs

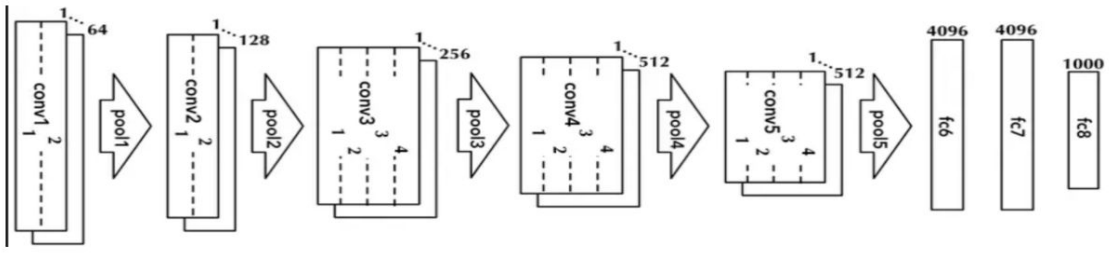

Fig 2.2: VGG 19 Architecture. Consists of five blocks of CNN layers, each block followed by a pooling layer. The five CNN blocks are followed by two fully connected layer followed by output layer.

The first CNN block, Block1 consists of 2 convolution layer with 64 convolution kernels in each layer. Block1 is followed by a pooling layer. Block2 consists of 2 convolution layer with 128 convolution kernels in each layer. Block2 is followed by a pooling layer. Block3 consists of 4 convolution layer with 256 convolution kernels in each layer. Block3 is followed by a pooling layer. Block4 consists of 4 convolution layer with 512 convolution kernels in each layer. Block4 is followed by a pooling layer. Block5 consists of 4 convolution layer with 512 convolution kernels in each layer. Block5 is followed by a pooling layer. The convolution architecture is followed by two fully connected layer of size 4096 each followed by the output layer of size 1000.

2.7 SSD

SSD [1] is single shot multibox detector developed by W Liu. SSD is build over VGG16 classification model as feature extractor. They introduced the concept of anchor boxes in unified architecture for detection, which was fromarly used in R-CNN (not a unified architecture for object detection). After this approach YOLOv2 [3] was trained and build on both approaches, once with anchor boxed and once without achor boxes. However in YOLOv3 [4] anchor boxes were strictly used. The final two layers in the YOLOv1 [2] was fully connected layer, which made the detection network slow. SSD completely eliminated the concept of fully connected layer. By removing fully connected layer SSD was able to predict higher

number of detections with much fewer parameters. The default anchor boxes are chosen manually.

SSD extracts feature map with base VGG16. The first layer responsible for detection is Conv4_3 layer. It has 38 X 38 cells with four object prediction in each cell. Each prediction has four bounding box coordinates (x1, 2x, y1,y2) and twenty one class prediction scores. SSD reserves class 0 for no object. So Conv4_3 layer makes total 38 X 38 X 4 prediction with shape 38 X 38 X 4 X 25 for Pascal VOC data [13]. SSD adds six more layers after VGG16. Five of them are responsible for detecting objects in the image. Three out of those five layers predicts six predictions for the cell, not four. All in all SSD predicts 8732 predictions with the help of six convolution layer.

SSD defines a scale value for each feature map layer. Starting from the 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, 1/2 and 1/3.

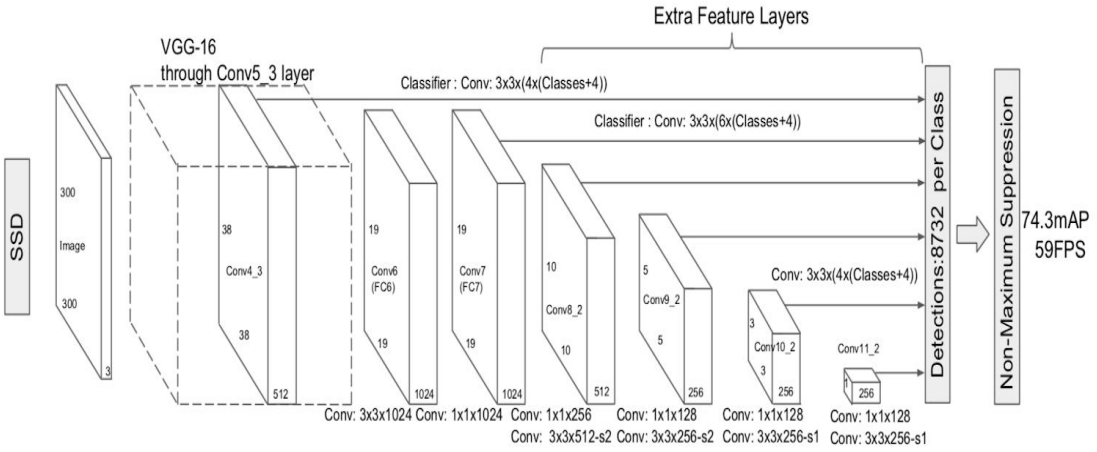

Fig 2.3: SSD model. Single Shot MultiBox Detector. A faster unified architecture for object detection.

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1 YOLO

YOLO [2,3,4] is You Only Look Once. One of the earliest approaches towards unified architecture for object detection that can run in realtime. A single model predicts the class, related probabilities for bounding box. YOLO version one treats detection as a regression problem. However the later versions used anchor boxes for detection. YOLO is trained on entire image (no region proposal like R-CNN), this has several advantages. First this makes YOLO very fast as compared to traditional modes for object detection. Second this eliminates a very complex pipeline of sliding window and region proposal for classifier, and so YOLO reasons globally for an image. OLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Third, YOLO learns generalizable representations of objects, this leads YOLO to be more robust and accurate when applied to new domains or unexpected inputs.

YOLO's unified design enables end-to-end training and testing, with real time speed at high mean average precision. YOLO divides the image into the grid of S*S. If the center of an object lies in the grid, the grid is responsible for the class, class probability and bounding box prediction. Each S*S grid predicts B class predictions, each B prediction has 4 bounding box coordinates(x, y, h and w, where x and y represents the centre of the image.) and 1 confidence score with C conditional Class probabilities. The confidence score measures how sure the model is that the grid lies in the cell. If there is no object with center in the grid the model produces zero confidence else the confidence is the IOU of the true positive dete So the output is encoded as S * S (B * 5 + C) tensor. For the Pascal VOC data they use S = 7, B = 2 and since Pascal VOC has 20 classes [13], so C = 20. The final output is of the shape 7 * 7 * 30.

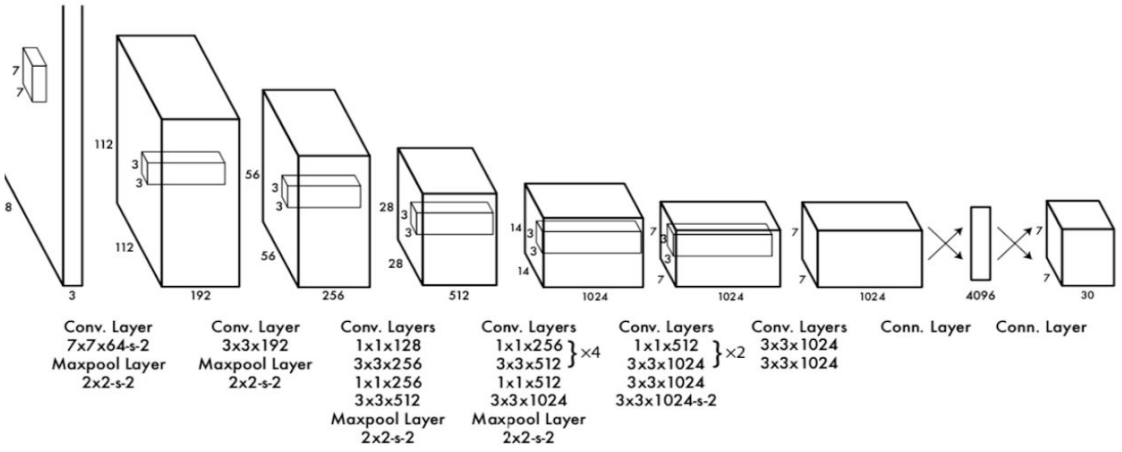

Fig 3.1: Original YOLO architecture. [2]

3.2 Modified YOLO

We present a modified version of YOLO. Keeping most of the YOLO architecture as it is we increase the mAP of YOLOv3 from 60.6% mAP to 64.02% mAP on our modified YOLOv3 on Pascal VOC 2007 test data. We use the concept of neighbourhood subtraction. This concept is inspired from convolution operation. Convolution operation is weighted sum of the neighbourhood. Depending on the kernel the weighted sum of the neighbour results in detection certain specific features extraction. A convolutional kernel extract a specific feature and suppresses other features in the image. We derive inspiration from this aspect of the result of convolution operation. Our approach is based on the following two reasoning, first most of the image has a noisy background, and there is a range of pixels which mark the border of the object and the background image. Subtraction of blocks from the neighbouring blocks results in following two results, the background is suppressed implying less chances of false positives, thus increasing mAP and enhancing of the borders (features/edges) of the objects. The size of the block is an important facture. We could not try many variations of the block size, but the large size of block (like 7X7, 8X8) will lead to complete image distortion.

Similar to convolution kernel the neighbourhood subtraction also has a kernel size (block size). However the movement of kernel is only either horizontal or vertical and also only in either direction (i.e, only either left or right in case of horizontal movement and only either up or down in vertical movement).

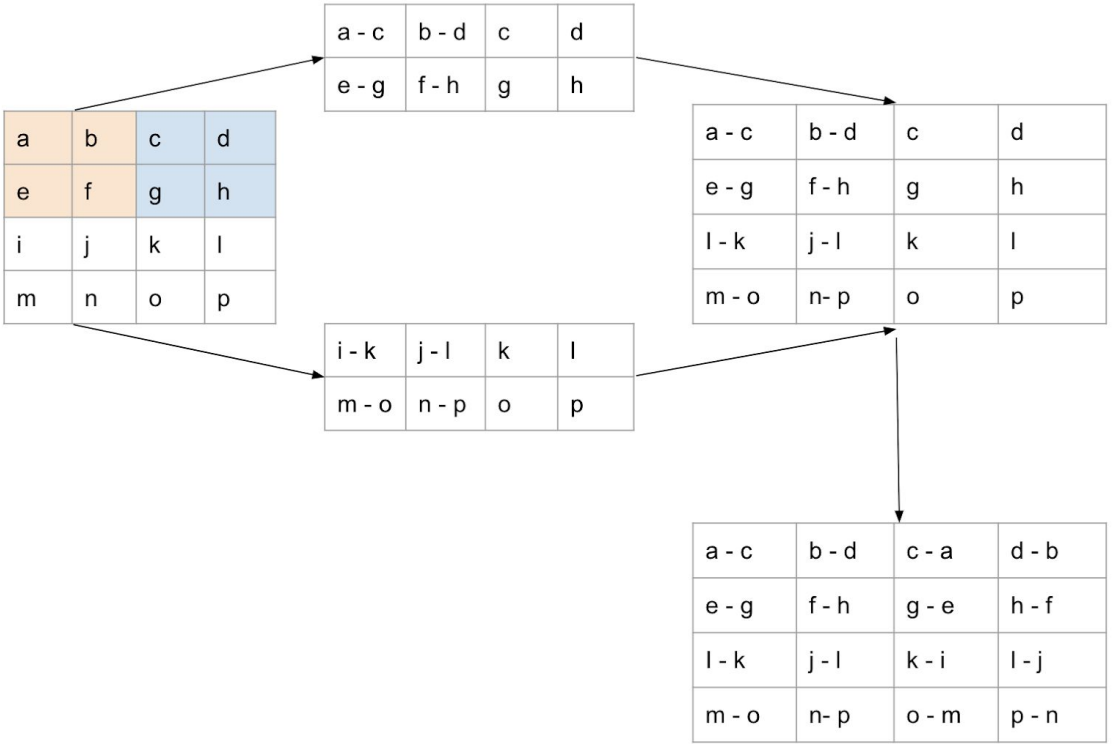

Fig 3.2: Horizontal neighbourhood subtraction with right movement of kernel with block size of 2X2.

An example of the neighbourhood block subtraction is demonstrated in the figure 3.2. The example shows horizontal subtraction with block of shape 2X2. The resulting image is entirely the difference of the neighbours. This enables to pick up the sharpe change in the image frequencies, borders, corners and edges, features those are more important than the others.

Fig: 3.3: The image on the left is from Pascal VOC 2007 train data. The image on the right is the result of neighbourhood block subtraction with block of size 4*4

The figure 3.3 shows the difference in the original image from Pascal VOC 2007 train data to the resultant image after neighbouring block subtraction with block size 4X4. A lot of irrelevant information in the image gets suppressed. Class determining features of the image are still maintained. This helps the convolution kernels to pick up and enhance the useful features in the image for class prediction and Bounding box detection.

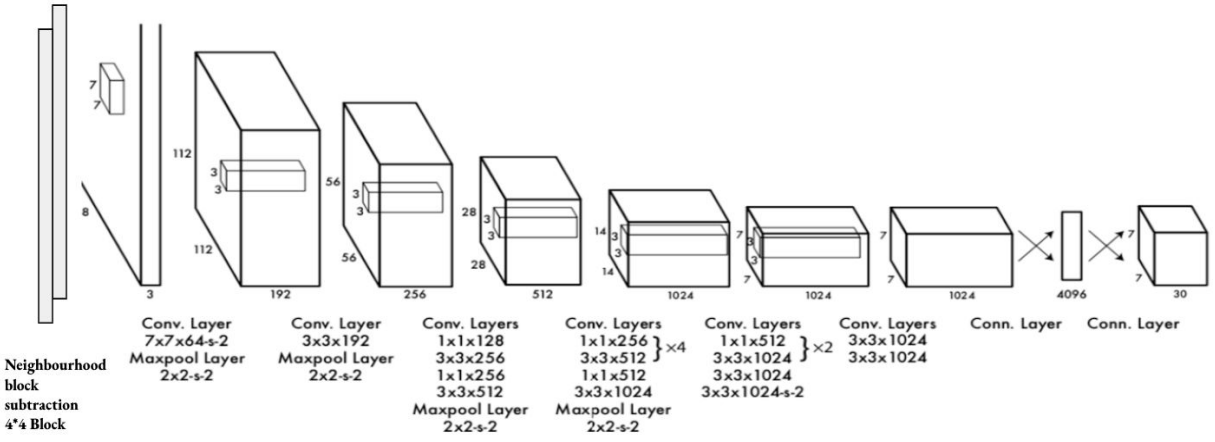

Fig 3.4 Modified YOLO architecture. Neighbourhood block subtraction at the beginning of the YOLO architecture. Block size 4 * 4.

We add block subtraction operation on just before the first convolutional layer as shown in Figure 3.4. We keep a copy of the original image and from the final predictions (the output tensor) we draw the bounding boxes and class labels on the copy of original image as visual output. However as of now the block subtraction is not the part of the network itself. We modified out training data (input image) before training and while prediction we do it outside of the network. We would like to keep out modified YOLO end to end trainable, thus we wish to implement this in the network architecture itself.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Software Used

1. Mac OS X - High Sierra 10.13.6
2. We build our Modified YOLO model on darkent framework by Joseph Redmon (inventor of yolo) [17]
3. Nvidia Graphic driver 396
4. CUDA Toolkit 10.1
5. cuDNN v7.6.0 for CUDA 10.1
6. Xcode 10.1 (10B61)
7. Apple LLVM 10.0.0
8. Python 3.x
9. Additional dependencies - numpy and opencv 4.x
10. Dataset - Pascal VOC 2007 test, train, val + Pascal VOC 2012 train, val [13]

4.2 Hardware Used

1. MacBook Pro 2017 model without touch bar (2 TB3 ports)
   i5-7360U CPU @ 2.30GHz
2. eGPU Enclosure: Sonnet eGFX Breakaway Box 550W (GPU-550W-TB3)
3. Video Card: GTX 1080 Ti
    a. VRAM 11GB
    b. CUDA cores count 3584 and
    c. CUDA compute capability 6.1.

```
[Eshans-MacBook-Pro:samples eshanpandey$ cd /Developer/NVIDIA/CUDA-10.1/samples/bin/x86_64/darwin/release
[Eshans-MacBook-Pro:release eshanpandey$ ./deviceQuery
 ./deviceQuery Starting...

  CUDA Device Query (Runtime API) version (CUDART static linking)

 Detected 1 CUDA Capable device(s)

 Device 0: "GeForce GTX 1080 Ti"
   CUDA Driver Version / Runtime Version          10.1 / 10.1
   CUDA Capability Major/Minor version number:    6.1
   Total amount of global memory:                 11264 MBytes (11810963456 bytes)
   (28) Multiprocessors, (128) CUDA Cores/MP:     3584 CUDA Cores
   GPU Max Clock rate:                            1620 MHz (1.62 GHz)
   Memory Clock rate:                             5505 Mhz
   Memory Bus Width:                              352-bit
   L2 Cache Size:                                 2883584 bytes
   Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
   Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
   Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
   Total amount of constant memory:               65536 bytes
   Total amount of shared memory per block:       49152 bytes
   Total number of registers available per block: 65536
   Warp size:                                     32
   Maximum number of threads per multiprocessor:  2048
   Maximum number of threads per block:           1024
   Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
   Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
   Maximum memory pitch:                          2147483647 bytes
   Texture alignment:                             512 bytes
   Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
   Run time limit on kernels:                     Yes
   Integrated GPU sharing Host Memory:            No
   Support host page-locked memory mapping:       Yes
   Alignment requirement for Surfaces:            Yes
   Device has ECC support:                        Disabled
   Device supports Unified Addressing (UVA):      Yes
   Device supports Compute Preemption:            Yes
   Supports Cooperative Kernel Launch:            Yes
   Supports MultiDevice Co-op Kernel Launch:      Yes
   Device PCI Domain ID / Bus ID / location ID:   0 / 133 / 0
   Compute Mode:
      < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

 deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.1, CUDA Runtime Version = 10.1, NumDevs = 1
 Result = PASS
 Eshans-MacBook-Pro:release eshanpandey$
```

Fig 4.1: Snapshot of video card device query specifying various device specification. VRAM 11GB, total CUDA cores count 3584 and CUDA compute capability 6.1.

4.3 Implementation details

We start training our YOLO model darknet53conv74 weights. Hyper parameters settings are as follows: batch = 64, subdivisions = 16, height = 416, width = 416, channels = 3, momentum = 0.9, decay = 0.0005, saturation = 1.5, exposure = 1.5, hue = 0.1, learning rate = 0.001, maxbatches = 50200. We trained for 50200 iterations on Pascal VOC 2007 tarin and val, 2012 train data, this took us approximately 65 hours on Nvidia GTX 1080 Ti. We validated our model on Pascal VOC 2007 test set.

We build the darknet with GPU = 1, CUDNN = 1, OPENCV = 0, OPENMP = 0, DEBUG = 0.

Following is the YOLO model architecture

```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32 0.299 BF
```

1 conv 64 3 x 3 / 2 416 x 416 x 32 -> 208 x 208 x 64 1.595 BF

2 conv 32 1 x 1 / 1 208 x 208 x 64 -> 208 x 208 x 32 0.177 BF

3 conv 64 3 x 3 / 1 208 x 208 x 32 -> 208 x 208 x 64 1.595 BF

4 Shortcut Layer: 1

5 conv 128 3 x 3 / 2 208 x 208 x 64 -> 104 x 104 x 128 1.595 BF

6 conv 64 1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x 64 0.177 BF

7 conv 128 3 x 3 / 1 104 x 104 x 64 -> 104 x 104 x 128 1.595 BF

8 Shortcut Layer: 5

9 conv 64 1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x 64 0.177 BF

10 conv 128 3 x 3 / 1 104 x 104 x 64 -> 104 x 104 x 128 1.595 BF

11 Shortcut Layer: 8

12 conv 256 3 x 3 / 2 104 x 104 x 128 -> 52 x 52 x 256 1.595 BF

13 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

14 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

15 Shortcut Layer: 12

16 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

17 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

18 Shortcut Layer: 15

19 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

20 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

21 Shortcut Layer: 18

22 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

23 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

24 Shortcut Layer: 21

25 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

26 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

27 Shortcut Layer: 24

28 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

29 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

30 Shortcut Layer: 27

31 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

32 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

33 Shortcut Layer: 30

34 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

35 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

36 Shortcut Layer: 33

37 conv 512 3 x 3 / 2   52 x 52 x 256 ->   26 x 26 x 512 1.595 BF

38 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

39 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

40 Shortcut Layer: 37

41 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

42 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

43 Shortcut Layer: 40

44 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

45 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

46 Shortcut Layer: 43

47 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

48 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

49 Shortcut Layer: 46

50 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

51 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

52 Shortcut Layer: 49

53 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

54 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

55 Shortcut Layer: 52

56 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

57 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

58 Shortcut Layer: 55

59 conv 256 1 x 1 / 1   26 x 26 x 512 ->   26 x 26 x 256 0.177 BF

60 conv 512 3 x 3 / 1   26 x 26 x 256 ->   26 x 26 x 512 1.595 BF

61 Shortcut Layer: 58

62 conv 1024 3 x 3 / 2   26 x 26 x 512 ->   13 x 13 x1024 1.595 BF

63 conv 512 1 x 1 / 1   13 x 13 x1024 ->   13 x 13 x 512 0.177 BF

64 conv 1024 3 x 3 / 1   13 x 13 x 512 ->   13 x 13 x1024 1.595 BF

65 Shortcut Layer: 62

66 conv 512 1 x 1 / 1   13 x 13 x1024 ->   13 x 13 x 512 0.177 BF

67 conv 1024 3 x 3 / 1   13 x 13 x 512 ->   13 x 13 x1024 1.595 BF

68 Shortcut Layer: 65

69 conv 512 1 x 1 / 1   13 x 13 x1024 ->   13 x 13 x 512 0.177 BF

70 conv 1024 3 x 3 / 1   13 x 13 x 512 ->   13 x 13 x1024 1.595 BF

71 Shortcut Layer: 68

72 conv 512 1 x 1 / 1 13 x 13 x1024 -> 13 x 13 x 512 0.177 BF

73 conv 1024 3 x 3 / 1 13 x 13 x 512 -> 13 x 13 x1024 1.595 BF

74 Shortcut Layer: 71

75 conv 512 1 x 1 / 1 13 x 13 x1024 -> 13 x 13 x 512 0.177 BF

76 conv 1024 3 x 3 / 1 13 x 13 x 512 -> 13 x 13 x1024 1.595 BF

77 conv 512 1 x 1 / 1 13 x 13 x1024 -> 13 x 13 x 512 0.177 BF

78 conv 1024 3 x 3 / 1 13 x 13 x 512 -> 13 x 13 x1024 1.595 BF

79 conv 512 1 x 1 / 1 13 x 13 x1024 -> 13 x 13 x 512 0.177 BF

80 conv 1024 3 x 3 / 1 13 x 13 x 512 -> 13 x 13 x1024 1.595 BF

81 conv 75 1 x 1 / 1 13 x 13 x1024 -> 13 x 13 x 75 0.026 BF

82 yolo

83 route 79

84 conv 256 1 x 1 / 1 13 x 13 x 512 -> 13 x 13 x 256 0.044 BF

85 upsample 2x 13 x 13 x 256 -> 26 x 26 x 256

86 route 85 61

87 conv 256 1 x 1 / 1 26 x 26 x 768 -> 26 x 26 x 256 0.266 BF

88 conv 512 3 x 3 / 1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF

89 conv 256 1 x 1 / 1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF

90 conv 512 3 x 3 / 1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF

91 conv 256 1 x 1 / 1 26 x 26 x 512 -> 26 x 26 x 256 0.177 BF

92 conv 512 3 x 3 / 1 26 x 26 x 256 -> 26 x 26 x 512 1.595 BF

93 conv 75 1 x 1 / 1 26 x 26 x 512 -> 26 x 26 x 75 0.052 BF

94 yolo

95 route 91

96 conv 128 1 x 1 / 1 26 x 26 x 256 -> 26 x 26 x 128 0.044 BF

97 upsample 2x 26 x 26 x 128 -> 52 x 52 x 128

98 route 97 36

99 conv 128 1 x 1 / 1 52 x 52 x 384 -> 52 x 52 x 128 0.266 BF

100 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

101 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

102 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

103 conv 128 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 128 0.177 BF

104 conv 256 3 x 3 / 1 52 x 52 x 128 -> 52 x 52 x 256 1.595 BF

105 conv 75 1 x 1 / 1 52 x 52 x 256 -> 52 x 52 x 75 0.104 BF

4.4 Results

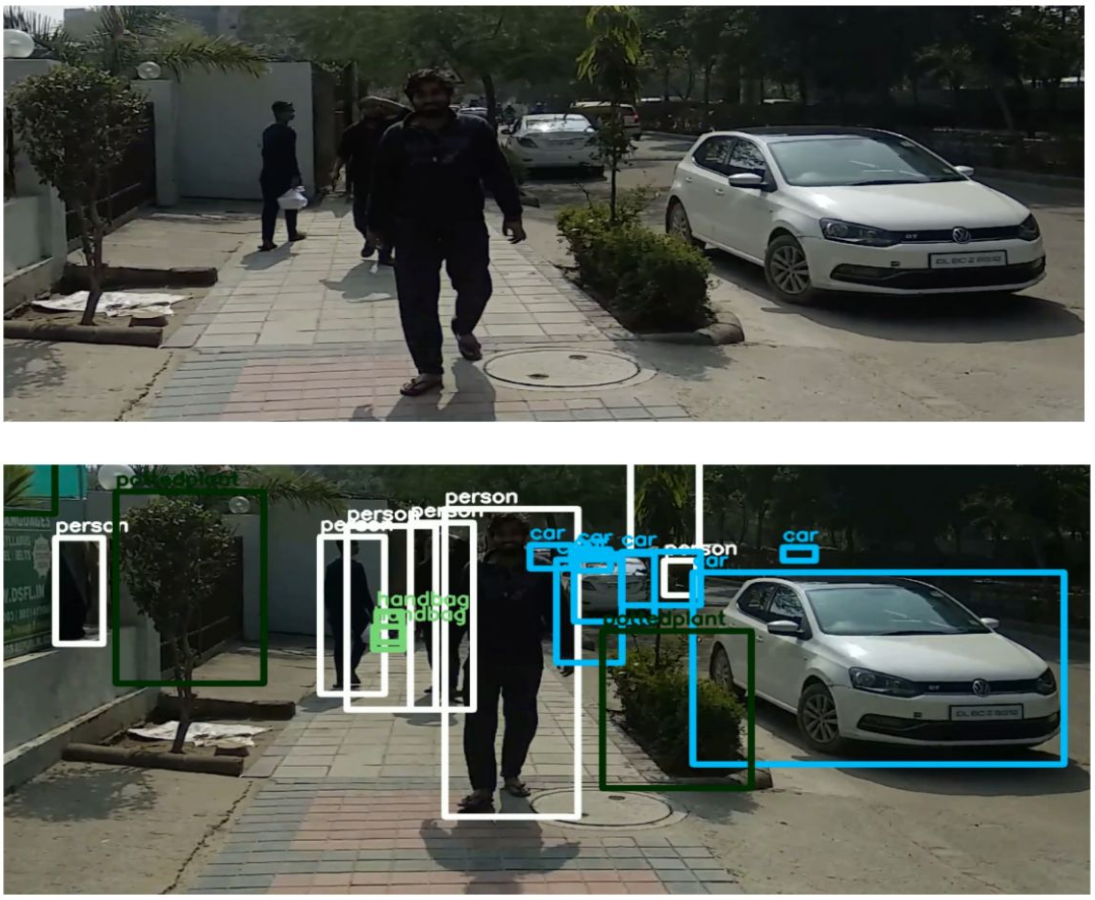

Fig 4.2: YOLO v3 model prediction on real life image

A sample prediction of the model is shown in the figure. When compared with the entries of the Pascal VOC 2007 entries out model performs exceptionally well. Our model exceeds in precision in 12 out of 20 classes with the best performing model of the 2007 competition.

CHAPTER 5

CONCLUSION

5.1. Performance Evaluation

On Pascal VOC 2007 test we achieve 64.02% mAP at 0.5 IOU thresh and 73% mAP with 0.25 IOU. Class wise mAP precision are as follows

detections_count = 22739, unique_truth_count = 12032
class_id = 0, name = aeroplane, ap = 75.32% (TP = 199, FP = 35)
class_id = 1, name = bicycle, ap = 75.19% (TP = 231, FP = 42)
class_id = 2, name = bird, ap = 55.15% (TP = 225, FP = 76)
class_id = 3, name = boat, ap = 50.42% (TP = 138, FP = 92)
class_id = 4, name = bottle, ap = 36.54% (TP = 154, FP = 88)
class_id = 5, name = bus, ap = 73.14% (TP = 154, FP = 70)
class_id = 6, name = car, ap = 74.03% (TP = 828, FP = 156)
class_id = 7, name = cat, ap = 77.94% (TP = 262, FP = 79)
class_id = 8, name = chair, ap = 45.78% (TP = 333, FP = 360)
class_id = 9, name = cow, ap = 66.06% (TP = 159, FP = 109)
class_id = 10, name = dining table, ap = 59.19% (TP = 116, FP = 71)
class_id = 11, name = dog, ap = 72.63% (TP = 345, FP = 171)
class_id = 12, name = horse, ap = 81.56% (TP = 269, FP = 102)
class_id = 13, name = motorbike, ap = 76.66% (TP = 227, FP = 77)
class_id = 14, name = person, ap = 70.90% (TP = 3031, FP = 752)
class_id = 15, name = potted plant, ap = 32.35% (TP = 153, FP = 104)
class_id = 16, name = sheep, ap = 56.96% (TP = 141, FP = 110)
class_id = 17, name = sofa, ap = 68.81% (TP = 153, FP = 88)
class_id = 18, name = train, ap = 73.14% (TP = 195, FP = 49)
class_id = 19, name = tv monitor, ap = 58.58% (TP = 164, FP = 65)

for thresh = 0.25, precision = 0.73, recall = 0.62, F1-score = 0.67
for thresh = 0.25, TP = 7477, FP = 2696, FN = 4555, average IoU = 56.73 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.640169, or 64.02 %

5.2. Comparison with existing State-of-the-Art Technologies

When compared with the entries of 2007 VOC Challenge our modified YOLO performed much better than any other entry. We compared the results in the table below.

Models/ Class	Airplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	Dining table	Dog	Horse	Motorbike	Person	Potted plant	Sheep	Sofa	Train	TVmonitor
INRIA Flat	74.8	62.5	51.2	69.4	29.2	60.4	76.3	57.6	53.1	41.1	54.4	42.8	76.5	62.3	84.5	35.3	41.3	50.1	77.6	49.3
INRIA Genetic	**77.5**	63.6	**56.1**	**71.9**	33.1	60.6	**78.0**	58.8	**53.5**	42.6	54.9	45.8	77.5	64.0	**85.9**	**36.3**	44.7	50.6	**79.2**	53.2
INRIA Larlus	62.6	54.0	32.8	47.5	17.8	46.4	69.6	44.2	44.6	26.0	38.1	34.0	66.0	55.1	77.2	13.1	29.1	36.7	62.7	43.3
MPI BOW	58.9	46.0	31.3	59.0	16.9	40.5	67.2	40.2	44.3	28.3	31.9	34.4	63.6	53.5	75.7	22.3	26.6	35.4	60.6	40.6
PRIPUVA	48.6	20.9	21.3	17.2	6.4	14.2	45.0	31.4	27.4	12.3	14.3	23.7	30.1	13.3	62.0	10.0	12.4	13.3	26.7	26.2
QMUL HSLS	70.6	54.8	35.7	64.5	27.8	51.1	71.4	54	46.6	36.6	34.4	39.9	71.5	55.4	80.6	15.8	35.8	41.5	73.1	45.5
QMUL LSPCH	71.6	55.0	41.1	65.5	27.2	51.1	72.2	55.1	47.4	35.9	37.4	41.5	71.5	57.9	80.8	15.6	33.3	41.9	76.5	45.9
TKK	71.4	51.7	48.5	63.4	27.3	49.9	70.1	51.2	51.7	32.3	46.3	41.5	72.6	60.2	82.2	31.7	30.1	39.2	71.1	41.0
ToshCam rdf	59.9	36.8	29.9	40.0	23.6	33.3	60.2	33.0	41.0	17.8	43.2	33.7	63.9	53.1	77.9	29.0	27.3	31.2	50.1	37.6
ToshCam svm	54.0	27.1	30.3	35.6	17.0	22.3	58.0	34.6	38.0	19.0	27.5	32.4	48	40.7	78.1	23.4	21.8	28.0	45.5	31.8
Tsinghua	62.9	42.2	33.9	49.7	23.7	40.7	62.0	35.2	42.7	21.0	38.9	34.7	65	48.1	76.9	16.9	30.8	32.8	58.9	33.1
UVA Bigrams	61.2	33.2	29.4	45.0	16.5	37.6	54.6	31.3	39.9	17.2	31.4	30.6	61.6	42.4	74.6	14.5	20.9	23.5	49.9	30.0
UVA FuseAll	67.1	48.1	43.4	58.1	19.9	46.3	61.8	41.9	48.4	27.8	41.9	38.5	69.8	51.4	79.4	32.5	31.9	36.0	66.2	40.3
UVA MCIP	66.5	47.9	41.0	58.0	16.8	44	61.2	40.5	48.5	27.8	41.7	37.1	66.4	50.1	78.6	31.2	32.3	31.9	66.6	40.3
UVA SFS	66.3	49.7	43.5	60.7	18.8	44.9	64.8	41.9	46.8	24.9	42.3	33.9	71.5	53.4	80.4	29.7	31.2	31.8	67.4	43.5
UVA WGT	59.7	33.7	39.4	44.5	22.2	32.9	55.9	36.3	36.8	20.6	25.2	34.7	65.1	40.1	74.2	26.4	26.9	25.1	50.7	29.5
XRCE	72.3	57.5	53.2	68.9	28.5	57.5	75.4	50.3	52.2	39.0	46.8	45.3	75.7	58.5	84	32.6	39.7	50.9	75.1	49.5
Modified YOLO	75.3	**75.2**	55.2	50.4	**36.5**	**73.1**	74.0	**77.9**	45.8	**66.0**	**59.1**	**72.6**	**81.6**	**76.7**	70.9	32.4	**57.0**	**68.8**	73.1	**58.6**

Table 5.1: Comparison of our modified model with entries of VOC 2007 challenge. The highlighted entries are our results. Bold and underlined entries represent the best result in class. Out of 20 classes in Pascal Voc, our model perform produced highest mAP for 12 classes.

When compared with YOLOv3 trained on MS COCO dataset [12] our modified YOLOv3 outperforms the original implementation by 3.42% mAP, with very minimal decrement in speed.

5.3. Future Directions

The neighbourhood block subtraction gives us hope for potential research direction. We would try out blocks of varying sizes and compare the results, this will give us an idea about the ideal size of the block. We also would like to try out consecutive neighbourhood block subtraction layer by layer. As of now we sequentially perform the block subtraction, however this process could be more efficiently realised (in terms of time) by parallel execution with cuda. One other implementation we would like to make is include the block subtraction mechanism in the model architecture itself. We would also test this approach with other detection architectures like SSD and fast R-CNN. More experimentation will help us to develop more reasoning for the process.

Appendix

Darknet

Darknet is deep learning framework in C, developed by Joseph Redmon [17]. The original implementation of all the YOLO versions was in the Darknet. We also use darknet in our project.

OpenCV

The project utilised many different API's. OpenCV, the main computer vision API, was originally developed at Intel corporations and is now maintained at Willow Garage (http://www.willowgarage.com) and provides implementations of many widely used computer vision algorithms. The OpenCV project has been in development since 1999.

PASCAL VOC

Pascal VOC is the visual object classification challenge. It provides data set for training testing and validation [13]. We train our model on Pascal VOC 2007 train and val data and 2012 train data. We test our data on Pascal VOC 2007 test data.

References

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

2. Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016.

3. Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

4. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).

5. (Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." In *null*, p. 511. IEEE, 2001.)

6. Barrow, Harry G., and Jay M. Tenenbaum. "Interpreting line drawings as three-dimensional surfaces." *Artificial intelligence*17, no. 1-3 (1981): 75-116

7. Lindeberg, Tony. "Edge detection and ridge detection with automatic scale selection." *International Journal of Computer Vision* 30, no. 2 (1998): 117-156.

8. [Umbaugh, Scott E. *Digital image processing and analysis: human and computer vision applications with CVIPtools*. CRC press, 2010.]

9. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. "Deep learning. Book in preparation for MIT Press." *URL¡ http://www. deeplearningbook. org* (2016).

10. LeCun, Yann, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits, 1998." *URL http://yann. lecun. com/exdb/mnist* 10 (1998): 34.

11. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

12. COCO: Common Objects in Context. http://mscoco.org/dataset/ #detections-leaderboard (2016) [Online; accessed 25-July-2016].

13. Everingham, Mark, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes challenge: A retrospective." *International journal of computer vision* 111, no. 1 (2015): 98-136.

14. Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672-2680. 2014.

15. LeCun, Yann, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. "Backpropagation applied to handwritten zip code recognition." *Neural computation* 1, no. 4 (1989): 541-551.

16. LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series." *The handbook of brain theory and neural networks* 3361, no. 10 (1995): 1995.

17. Redmon, Joseph. "Darknet: Open source neural networks in c. Pjreddie. com." (2016).

ORIGINALITY REPORT

20%	15%	9%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	en.wikipedia.org Internet Source	3%
2	people.cs.clemson.edu Internet Source	2%
3	neurohive.io Internet Source	2%
4	medium.com Internet Source	2%
5	www.deeplearningbook.org Internet Source	1%
6	Submitted to University of Macau Student Paper	1%
7	Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016 Publication	1%

opensource.com

8	Internet Source	1%
9	Submitted to University of Sunderland Student Paper	1%
10	appliedmachinelearning.blog Internet Source	1%
11	Submitted to King's College Student Paper	1%
12	shartoo.github.io Internet Source	1%
13	dspace.uah.es Internet Source	<1%
14	Submitted to BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA BIBLIOTECA Student Paper	<1%
15	depositonce.tu-berlin.de Internet Source	<1%
16	Submitted to Queen Mary and Westfield College Student Paper	<1%
17	Submitted to Career Education Corporation Online Student Paper	<1%
18	Submitted to University of St Andrews Student Paper	<1%

19	www.instructables.com Internet Source	<1%
20	machinelearningonline.blog Internet Source	<1%
21	Submitted to University of Technology, Sydney Student Paper	<1%
22	Submitted to University of Lugano Student Paper	<1%
23	Submitted to Southern Illinois University Student Paper	<1%
24	Submitted to Chungnam National University Student Paper	<1%
25	Submitted to University of Mauritius Student Paper	<1%
26	Submitted to University of Warwick Student Paper	<1%
27	Advances in Intelligent Systems and Computing, 2015. Publication	<1%
28	Submitted to Università degli Studi di Trieste Student Paper	<1%
29	Submitted to International Islamic University Malaysia Student Paper	<1%

30	Submitted to Asian Institute of Technology Student Paper	<1%
31	"Advances in Artificial Intelligence", Springer Science and Business Media LLC, 2019 Publication	<1%
32	towardsdatascience.com Internet Source	<1%
33	Submitted to University of Northumbria at Newcastle Student Paper	<1%
34	Submitted to University of Lancaster Student Paper	<1%
35	"Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications", Springer Nature, 2018 Publication	<1%
36	Submitted to Covenant University Student Paper	<1%
37	Submitted to Sim University Student Paper	<1%
38	Submitted to Coventry University Student Paper	<1%