COMPARATIVE ANALYSIS OF OBJECT DETECTION ALGORITHMS

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology in

Computer Science and Engineering

By

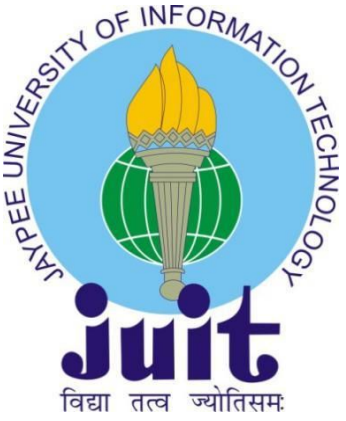
Ashutosh Singh - 151302

Rishabh Mehta - 151301

Under the supervision of

Mr. Nitin Kumar

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Certificate

Candidate's Declaration

We hereby declare that the work presented in this report titled "**Comparative Analysis of Object Detection Algorithms**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from August 2018 to May 2019 under the supervision of Mr. Nitin Kumar (Assistant Professor, Department of Computer Science and Engineering and Information Technology). The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Rishabh Mehta
151301

Ashutosh Singh
151302

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mr. Nitin Kumar
Assistant Professor

Dated:

Acknowledgement

This project would not have been possible without the professional guidance of our supervisor Mr. Nitin Kumar. We are indebted to him for investing his invaluable time throughout this project. His constant feedback has helped in polishing the work we have put forth. As our mentor, he has taught us more than we could ever give him credit for here. He has shown us, by example, what a good professional should be.

We are also grateful to all of those with whom we have had the pleasure to work during this project.

Table Of Contents

Abstract

The field of image processing has attracted a lot of attention during the last decade. Object detection algorithms have seen rapid development from conventional architectures to more sophisticated architectures which rely on the neural networks for cognitive pattern recognition. Sophisticated machine learning algorithms and faster GPUs have rendered us with a plethora of algorithms for object classification as well as object detection, the most prominent ones have been discussed in this report. Our main objective is to compare object classification and object detection models. From the number of proposed models over the years, this work picks the best, "state of the art" object detection models for comparison, namely You Only Look Once and Single Shot Multibox Detector. Moreover, this work also compares the underlying backbone architecture of these models and how well they fare off against each other.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Humans, without any forethought, can perceive, deduce and extract valuable information with relative ease. When it comes to images, humans can not only detect multiple objects with accuracy and precision, but are also able to decipher the context. Holistic understanding of images will provide computers with leeway to become operational in vast applications, ranging from military to purely academic. The benefits of advancement in this field will benefit industries where data is stored in the form of videos or images like security and surveillance, advanced driver assistance systems (ADAS) and autonomous driving, facial recognition, drone services etc, but not limiting to them. Automation in factories and offices have also been boosted by the work done in the field of optical character recognition, industrial line inspection systems, manufacturing defects systems.

1.2 Problem Statement

One of the most fundamental and longstanding problem of computer vision is object detection, which refers to finding instances of real world objects such as bikes, cars, faces in videos or images. Object detection involves both classification and localization of an object in the image. The former relates to identifying the class of the image, whereas the latter refers to detecting the location of the object in the image i.e, placing a bounding box around the object in the image. Seemingly simple problem at first, becomes increasingly difficult when you account for different viewpoints, illumination, deformation, occlusion.

With ever changing landscape of computer vision powered by newly published research everyday claiming to get the best results, this work endeavors to find whether or not can the research results be replicated, if so, to what measure and draw comparisons with other contemporary object detection models based on deep neural networks.

1.3 Methodology

To draw meaningful comparisons between the different classifiers and object detection models, they are trained on a standard datasets. Once the technical concepts of classifiers and object detection models have been discussed, this report will finally begin comparing the different classifiers and object detection models with the test data from the standard datasets, which will provide us with valuable information about the accuracy of each model. Metrics such as confusion matrix, accuracy, recall, precision, mAp scores have been utilized in order to analyze the obtained results.

1.4 Organization

First and foremost, a brief but comprehensive literature survey on past practices and seminal works which laid the foundation for modern frameworks has been presented, which discusses object classification and object detection techniques prior to the emergence of deep neural networks, such as KNN classifier, SVM classifier, Softmax classifier, Viola Jones [4], HOG [5], and other machine learning based algorithms. This report begins with the very basic problem of object detection and what it entails by breaking the problem into 3 stages-- information region selection, feature extraction, object classification. Then it sheds some light on challenges faced by object detection models. It discusses some popular classification algorithms. Then it focuses on the history of deep learning and its influence in the field of object detection. It takes a brief look at modern day object detectors and their working. Experimental results of YOLO[1] and SSD [2] have been put forth to draw some meaningful conclusions. Finally, it addresses several problems which the aforementioned object detectors fail to address and new promising directions for better object detection models.

1.5 Conclusion

This chapter provides the reader with a little insight into the project. It discussed the basic problem of object detection and how it is going to be tackled in this report. It also provides a brief overview about how the report has been structured.

CHAPTER 2
LITERATURE SURVEY

This chapter focuses on some of the seminal works which have laid the foundation for modern frameworks, it also discusses popular classification and object detection techniques prior to the emergence of deep neural networks. The likes of KNN classifier, SVM classifier, Softmax classifier, Viola Jones [4], HOG [5], and other machine learning based algorithms have been outlined in this chapter. It begins with the very basic problem of object detection and what it entails by breaking the problem into 3 stages-- information region selection, feature extraction, object classification. It also sheds some light on challenges faced by object detection models. We discuss some popular classification algorithms. Then it focuses on the history of deep learning and its influence in the field of object detection. Towards the end of this chapter, it takes brief look at modern day object detectors and their working.

2.1 Object Detection

One of the most fundamental and longstanding problem of computer vision is object detection, which refers to finding instances of real world objects such as bikes, cars, faces in videos or images. Object detection involves both classification and localization of an object in the image. The former relates to identifying the class of the image, whereas the latter refers to detecting the location of the object in the image i.e, placing a bounding box around the object in the image. Seemingly simple problem at first, becomes increasingly difficult when you account for different viewpoints, illumination, deformation, occlusion.

But, before we concern ourselves with various object detection models, first we must understand the problem and the challenges concerning it. We can divide the process of object detection models broadly into 3 categories as given in [6], informative region selection, feature extraction, classification.

Informative Region Selection [6] This stage tries to identify what all objects are present in the image and where they're located. Needless to say, there a plethora of

ways to do this, but only a very few number of ways which are computationally economic and produce satisfactory region selections. A select few have been discussed in this paper, spanning from the very naive divide and conquer approach (which partitions the image into a number of segments, which are then fed to a classifier for classification of objects present in that segment), to exhaustive approach such as scanning the whole image with a multi-scale sliding window, to sophisticated deep learning techniques for feature selection.

Feature Extraction [6] To identify a multitude of different objects, feature extraction algorithms need to be powerful in their indexing capabilities, in other words certain attributes which are characteristic to that object and differentiate them from others, need to be associated or mapped to those objects. Feature extraction is essential for every image classification and object recognition model. Formally, as defined in [7], mapping image pixels into feature space is known as feature extraction. Several models are based on a simple architecture, which make use of feature extraction algorithm in conjunction with a powerful classifier for object detection.

Classification [6] Determine which of multitude of possible categories are present in that information region selection. Promising new classifiers such as SVM are ergonomic and can even be run on personal computers within reasonable time frames. A lot of research work has been done on image classification and considerable gains have been made by image classification algorithms as outlined by Chum and Zisserman [8] during the last two decades. AdaBoost [9], Deformable Part-Based Model [10], Support Vector Machine [11] are some of the most heavily used robust classifiers. We will discuss some of the most popular ones in this report, namely, KNN-classifier, SVM classifier and Softmax classifier.

2.2 Classification

In this section the problem of Image Classification and popular classifiers has been discussed. Image Classification involves the the task of determining an input image one label from a fixed set of categories which may or may not be mutually exclusive. This is a longstanding fundamental problem in the area of Computer Vision, which, despite its simplicity, powers many practical applications. Other Computer Vision problems such as object detection and image segmentation can be reduced to image classification.

2.2.1 K-Nearest Neighbor

The kNN algorithm is the most simple image classification algorithm. Training a kNN classifier simply consists of storing the images and the labels of the training images. A new image is classified using a similarity index which can be computed using a distance function-- Manhattan distance(d_1), Euclidean distance(d_2).

Manhattan Distance L1	$d_1(I_1, I_2) = \sum_p \left\lvert I_1^p - I_2^p \right\rvert$
Euclidean Distance L2	$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$

Table 1: Similarity index metrics

The Manhattan distance compares the images pixel by pixel and adds up all the differences for all the pixels.

The Euclidean distance, simply computes the pixel wise difference, which is squared, and then it sums those values for each pixel to finally take the square root, which gives us the desired L2 distance.

Once the k closest images to the given image using one of L1/L2 distance are found, the image can be assigned the label which is dominant amongst its k-neighbors. This makes it more resistant to outliers.

2.3.2 Support Vector Machine

An SVM classifier is a linear classifier which uses the concept of decision planes to divide different data points into non overlapping classes. The decision planes defines the decision boundaries which separate the data points that belong to mutually exclusive classes. That being said, it may still not result in perfect classification, when such a situation arises, the SVM classifier finds a hyperplane that maximizes the margin and minimizes the misclassifications.

A linear classifier defines a linear mapping from images to label scores. Formally:

$$f(x_i,\ W,\ b) = Wx_i + b$$

The above equation, maps the raw image pixels to label scores for $\boldsymbol{K}$ labels. Where, $\boldsymbol{x}_i$ is the image vector of shape [D x 1] associate with a label $\boldsymbol{y}_i$. The weight matrix, $\boldsymbol{W}$ is of size [K x D], and finally the bias vector $\boldsymbol{b}$ is of size [K x 1]. The eventual output gives us a matrix of size [K x 1] which constitutes of scores for each class.

Every row of the weight vector, $\boldsymbol{W}$, corresponds to a template for one of the classes. Therefore, this implies that the linear classifier is effectively matching templates, where the templates are learned. This weight vector parametrizes our score function and thus, can be altered to closely align the predicted scores with the ground truth labels in the training data.

We define another function to measure the effectiveness of the learned templates, called the SVM loss. For a given class j [1..K], with score $\boldsymbol{s_j = f(x_i, W)}$, the SVM loss for i-th image $\boldsymbol{(x_i, y_i)}$ is given as:

$$L_i = \sum_{j \neq i} max(0,\ s_j - s_{y_i} + \Delta)$$

In other words, the SVM loss wants the score for the correct class to be higher than the score for incorrect classes by some fixed margin Δ. If any class, has a score which

cannot overcome this margin, then it will incur some loss, otherwise the loss will be zero. The objective of SVM loss function is to help in finding the weights that not only satisfy this constraint, but also minimize the total loss given by:

$$L = \frac{1}{N}\sum_i L_i$$

2.3.3 Softmax Classifier

Apart from k-NN and SVM classifier, the other most popular classifier is the Softmax classifier which has a different loss function. The Softmax classifier is generalization of a binary logistic classifier for multiple classes. The main difference between the Softmax and the SVM classifier arises when you look at the scores for interpretation. While the scores of the SVM classifier provide little to no insight for which class score should be given higher advantage, the softmax classifier instead computes probabilities which can easily be interpreted into confidence scores for each class.

The scores are interpreted as the unnormalized log probabilities for each class and replace we compute the loss using the **cross-entropy loss**:

$$L_i = -f_{y_i} + log\sum_j e^{f_j}$$

where $\boldsymbol{f_j}$ represents the score for the j-th class. The full loss for the dataset is computed by taking the mean of $\boldsymbol{L_i}$ for all $\boldsymbol{x_i}$. The function:

$$P(y_i \mid x_i;\ W) = f_j(z) = \frac{e^{z_j}}{\sum_j e^{f_j}}$$

is called the softmax function. It takes the vector of consisting of arbitrary real scores and squashes the values between zero and one that sum to one.

2.3 Factors affecting Object Detection

Superficially simple problem of object detection becomes increasingly complex when we take into account the factors discussed below. However, object detection in images with appropriate illumination, for relatively small database of objects is comparatively straightforward and deemed solved by the computer vision community. Occlusion and variable illumination present a challenge for the established models.

2.3.1 Positioning

Objects might be positioned differently in an image than the one used in the training data, and such cases must also be handled by the model.

2.3.2 Scale

Change in size of the object can arise from different viewpoints, regardless, the object recognition model must still be able to correctly recognize the object.

2.3.3 Illumination

Different lighting conditions, which may arise from different time of the day, different weather conditions may deem the object unrecognizable in the image. This can be problematic if your model depends upon color-based recognition model.

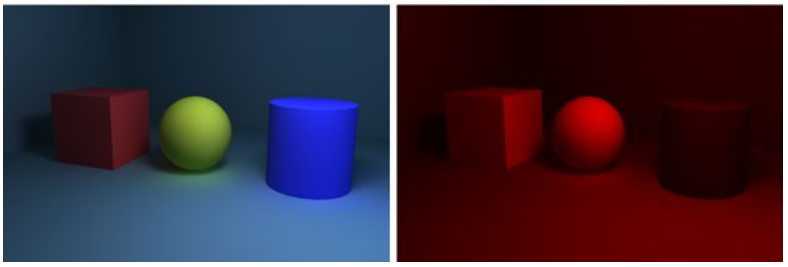


Fig 1: Objects with different illumination

2.3.4 Rotation

Different orientations of the same object correspond to the same object, and thus must be detected by the model. Orientation of the object must not hinder with the recognition of the object in the image.

Fig 2: Different orientations of an object

2.3.5 Occlusion

Simply put, a part of the object may be hidden due to some other obstructive object or the object itself, which might make it difficult to detect or classify the object. As outlined in [12], occlusion can be categorized broadly under three categories, i.e., inter-object occlusion, self-occlusion, or by background scene structure.

Fig 3: Occluded cars

2.4 History of Deep Neural Networks

The main requisite for any architecture based on core premises of artificial intelligence is cognition because it seeks to emulate the human mind. Cognitive computing is based on the model of the human brain, which aspires to identify a real-world concrete entity and its extent and intent. There are two approaches of performing the process of object detection on a digital image or video. Machine Learning based and deep learning based.

Prior to breakthroughs in the field of deep neural networks, machine learning based object detectors were handcrafted or engineered for object detection. This approach embeds the features that are distinct and characteristic to the object in the algorithm and then a classification algorithm is applied along with this handcrafted algorithm to

detect the class of the object. This seminal work laid the foundation for modern, state of the art, object detection models based on deep neural networks.

Viola-Jones Algorithm was the first machine learning based algorithm that was used to detect objects in an image or video at it lies at the foundation of OpenCV library. Algorithm was developed by Paul Viola and Michael Jones in 2001. Viola-Jones [4] algorithm uses haar-like features such as edge-features, line features, and four rectangles. The algorithm transforms the original image into a grayscale image as it reduces data to be processed and increases the speed of the algorithm. Next, it moves a sub-window from top-left corner to bottom-right corner in very small strides and it tries to find the haar-like features in the sub-window. The size of the sub-window varies as the object size in the image varies. Algorithm increases its efficiency by generating an integral image, with the help of integral images haar extractors can be calculated by only four arithmetic operations.

Cascading is used to further increase the efficiency of the algorithm. Cascading detects a particular number of strong features in the sub-window and if these strong features are not found in the sub-window it moves the sub-window without searching for the weak features in the sub-window hence saving computation power while increasing the rate of detection.

Another popular machine learning method is Histogram of Gradient. Histogram of Gradient HOG [5] is an object descriptor and performs surprisingly well in human detection in still images as well as videos. Features for all location in the image are extracted in this algorithm. Robert K.McConnell first described the concept of HOG [5] but it went famous after a supplementary work was done on it by Navneet Dalal and Bill Triggs.

An object in an image can be detected by capturing information about the gradient. We segment the image into multiple cells and for each pixel within the cells, we compile a histogram gradient. We get a descriptor when we combine these histograms. Contrast-normalization is applied on the local histograms. The

normalization value depends on intensity of the image block. These features are sent to SVM [11] for detecting the class of objects.

Deep learning-based approaches are able to perform the process of object detection without specifically defining features. Methods in this approach easily outperform machine learning based approaches. Deep learning approaches are generally based on CNN.

Convolutional neural networks are inspired by the visual cortex of human brain. Different neurons are activated when humans see an object, each responsible for detecting different feature such as lines, edges.

Emulation of this conduct of visual cortex can be achieved with the assistance of convolution, max pooling, and fully connected layer. The convolutional layer, applies convolution on the image and extract features from the input image, ReLU is applied on the after convolutional to increase non-linearity, max pooling is used to make it spatial invariant and fully connected layer is an artificial neural network, different neuron in different layer are activated when a particular type of feature is detected, it identifies which object belongs to which class.

CNN requires high computation power when detecting objects from the image and can be slow. Several powerful improvements have been made on CNN to make it faster even on a low-end devices. The likes of Single Shot MultiBox Detectors [2] and You Only Look Once [1] are the two object detection models which outperform other competitors when it comes to detection in real time.

2.5 Artificial Neural Networks

An artificial neural network is a system that mimics the human brain and nervous system. An artificial neural network consists of multiple layers. Layer from which input is provided is called the input layer and layer from which outputs are received are called output layer. The layers that are between the input and output layer are

referred to as the hidden layers. The layer consists of nodes called neurons, each neuron is connected to neurons in other layers by synapse. Each neuron is responsible for detecting the presence of a particular feature. Each neuron has an activation function which helps them to learn, figure out the pattern in the data and determine the presence of the particular neural feature. Synapse contain weight which determines whether the neuron will be activated or not. If for a particular neuron the input is 'x' and weight of the synapse is 'w' then,

$$z = w * x + b$$

where 'b' is the bias, it is added because when 'x' is zero for a specific neuron then that neuron will not be activated during the course of training. z is passed to the activation function, which detects when a distinct feature is present or not. Then it propagates our input data from the input layer, till it reaches the output layer via hidden layers between them.

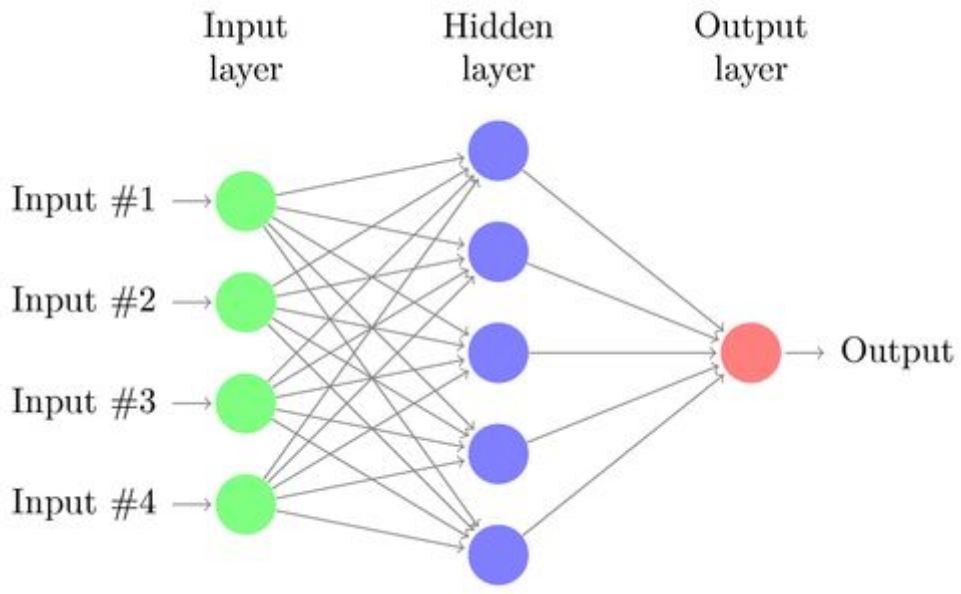


Fig 4: Architecture of Artificial Neural Network

2.4.1 Cost Function

A cost function is used to measure how well the neuron's predicted values are from the actual value. Cost function plays a pivotal role in the training of the model. If 'y' represents the actual value and 'a' is the predicted value of the neurons i.e.,

$$z = w * x + b$$

$$\sigma(z) = a$$

where 'σ' is activation function.

The Quadratic cost function will be given by,

$$c = (\Sigma(y - a)^2) / n$$

Our model will be effective when the value of the cost function is low because the difference b/w the predicted and actual value will be less. From the parameter 'w', 'x' and 'b' only 'w' can be changed as 'b' being the bias should be constant for every neuron and 'x' is the input previous neuron (from first layer it will be input value) can't be changed.

So, the weight of the neuron(synapse) 'w', should be adjusted in such a way that it minimizes the cost function to make our model more effective and predict an accurate result. The calculated value of the cost function is fed in the neural network so that the neurons can adjust their weights.

2.4.2 Gradient Descent

This is an optimization algorithm which is used to minimize the cost of a function. To find local minimum, we take steps proportional to the negative of the gradient. In essence, it checks the slope of the cost function at a to determine whether the slope is positive or negative.

If gradient descent is applied to a cost function which is not convex, it can choose local minimum instead of the global minimum. To get rid of this problem, Stochastic Gradient descent is used which updates the weights after processing every input value, unlike gradient descent which updates the weights after going through all the

input values. SGD (Stochastic gradient descent) is able to get rid of above problem because weight updates are large and have large fluctuation.

2.4.3 Back-Propagation

Back-Propagation [13] is used to quickly adjust the weights across the entire network. We calculate the error contribution of each neuron after of data is processed. It relies heavily on the chain rule to go back through the network and calculate the weights.

The network is fed the training data multiple times because it is impossible to train the model by going through the data only once. Every time our network goes through the training data, it is referred to as an epoch. We should always redo more epochs to make our model learn.

2.5 Convolutional Neural Network

Conventional neural systems are not perfect for image processing and should be encouraged images in decreased resolution pieces.

A convolutional neural system (CNN) is a sort of special neural network utilized in image processing and recognition that is particularly intended to process pixel information. A convolution neural network consists of two parts feature learning and classification. Feature learning constitutes of Convolution, ReLU and Pool and classification have flatten, fully connected layer and softmax.

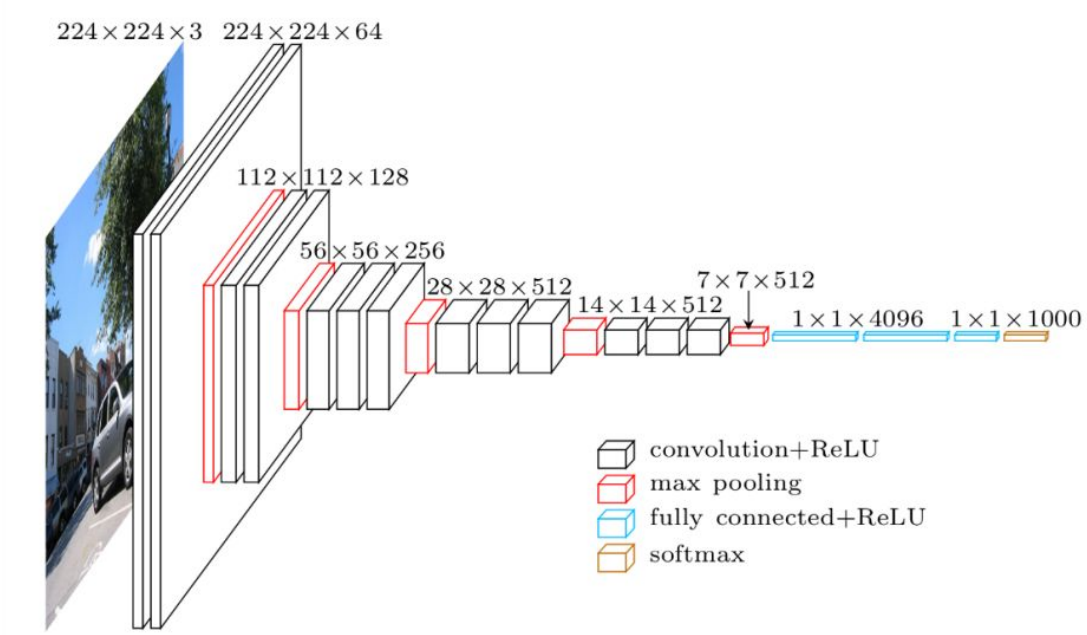


Fig 5: Architecture of Convolutional Neural Network

2.5.1 Convolution

The goal of a convolution layer is to extract features of the input image. Convolution preservers the relationship between the pixels by learning image features utilizing small squares of input data. After applying convolution on the input image size of the input image is reduced which is important as it reduces the number of input neurons in the input layer. Convolution is a mathematical function which takes input as image matrix and kernel.

Filter matrix is moved over the input image with a fixed stride and it multiplies the corresponding values of the filter matrix and input image. Operation like edge detection, blur and sharpen can be performed using convolution on the image using different filter map.

An image matrix of dimension: $(h * w * d)$

A filter: $fh * fw * d$

Outputs a volume dimension: $(h - f + 1) * (w - fw * 1) * 1$

Stride is defined as the number of shifts over the input matrix. Making a stride of one means we move the filter one pixel at a time. There are times when filter doesn't fit the image, we have two options to eradicate this problem, pad the zeros to the pictures so that it fits or drop the part of the image where the filter did not fit, the latter one is called valid padding because it keeps a valid part of the image and the former one is called zero-padding.

2.5.2 Non-Linearity (ReLU)

ReLU[13] is an acronym for Rectified Linear Unit, defined as:

$$f(x) = max(0, x)$$

Introduction of non-linearity is ReLU's [13] motivation. ReLU [13] is preferred over other non-linear function such as tanh, threshold or sigmoid because the performance of ReLU[13] is better. Element-Wise activation function max(0,x) is applied in this layer which transforms negative values to zeros.

2.5.3 Pooling Layer

Pooling layers are used to decrease the number of parameters. Spatial pooling is a technique used to shrink the dimensions of feature map while conserving the important details.

Spatial pooling can be done in 3 ways:
Max Pooling: Largest value is taken from rectified feature map.
Average Pooling: Average value is taken from the feature map
Sum Pooling: It is the aggregate of all elements in the feature map.

Max pooling filter of size 2x2 with a stride of 2 is generally used in pooling layer. For the pooling layer we have:

$$Input: W1 * H1 * D1$$

$$Stride: S$$

$$Filter: F$$

Then output will be W2 * H2 * D2 where,

$$W2 = (W1 - F)/S + 1$$

$$H2 = (H1 - F)/S + 1$$

$$D2 = D1$$

2.5.4 Fully Connected Layer

An artificial neural network in which every neuron from the previous node is connected with another layer forms the fully connected layer. As the input of the pooling is a matrix, the matrix is flattened into a vector so that it can be fed in the fully connected layer.

2.5.5 Softmax

Softmax is an activation function which is used by the last layer of the fully connected layer to classify generated features of the input image into different classes.

2.5.6 Convolutional neural network and object detection

A conventional convolution neural network cannot be used as an object detector because the number of occurrences of the objects are not fixed. To solve this problem, we can take a different approach, which is, to select a different segment from the image, and utilize a convolutional neural network to detect objects within that segment alone. The main problem with this methodology is that objects can have distinctive spatial area and aspect ratio within the picture.

2.6 Region Based Convolutional Neural Network (R-CNN)

Problem of object detection can be solved using region based convolutional neural network. Two thousand regions called region proposals are extracted from the image using selective search.

Following are the steps of selective search: high number of candidate regions are generated, similar regions are combined into larger method using greedy algorithm, final candidate region proposals are produced using the generated algorithm.

These two thousand region proposals are fed into a convolutional neural network after being wrapped into a square which produces 4096 dimensional features vector as output. The convolutional neural network act as a feature extractor and the dense output layer contains the features extracted from the image. The extracted features are then fed in Support Vector Machine which classifies the presence of the object within that candidate region proposal. Apart from detecting the object and its class the algorithm also gives four values to increase the precision of the bounding box.

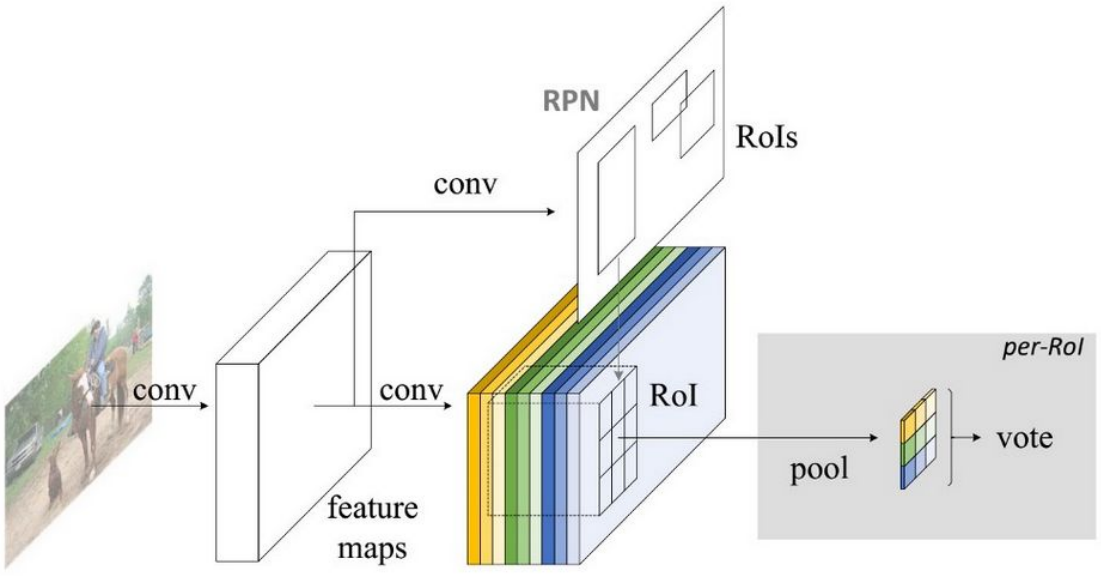


Fig 6: R-CNN Architecture

2.6.1 Issues with R-CNN

As no learning is occurring during selective search algorithm it could generate bad candidate region proposal. In addition to above it takes huge amount of time to train the network because of large number of region proposals. Also, it can't be implemented in real time.

2.6.2 Fast R-CNN

Problem of R-CNN regarding speed is solved by Fast R-CNN[15] sharing the computation of the convolutional layer between different proposals and swapping the order of generating regions proposals and running the CNN. Unlike R-CNN, images

are fed in the convolutional neural network first which generates a convolutional feature map.

From the convolutional feature map, region proposals are identified and then they are wrapped into squares. ROI pooling layer reshapes the squares so that they have a fixed size which can be fed to the fully connected layer. In RoI layer, features in region of interest are converted into a small feature map of HxW, both of the parameter H and W are tunable hyper-parameter.

From RoI feature vector, class of the proposed region is predicted using softmax layer, it also gives us the offset value for the bounding box.

Fast R-CNN [15] is outperforms R-CNN in terms of speed, because in R-CNN, two thousand region proposals are fed to the CNN. In Fast R-CNN [15], the convolution operations are performed only once and proposals are generated from the convolutional feature map.

2.6.3 Faster R-CNN

Fast R-CNN [15] makes use of the selective search algorithm to generate the region proposals which takes a lot of time as well as computational resources.

In Faster R-CNN [3], instead of selective search a Region Proposal Network is used for generating region proposals. Region Proposals are less compute intensive as compared to Fast R-CNN [15] and they share the most computation with the Object Detection Network. Region Proposal Network generates the anchor and ranked them and only the anchor who have high probability of containing object are sent to object detection network. Object Detection Network utilizes the region proposal to detect and predict the class of the object. Anchors are central point of the sliding window. In a convention Region Proposal Network there are 9 anchors at each position

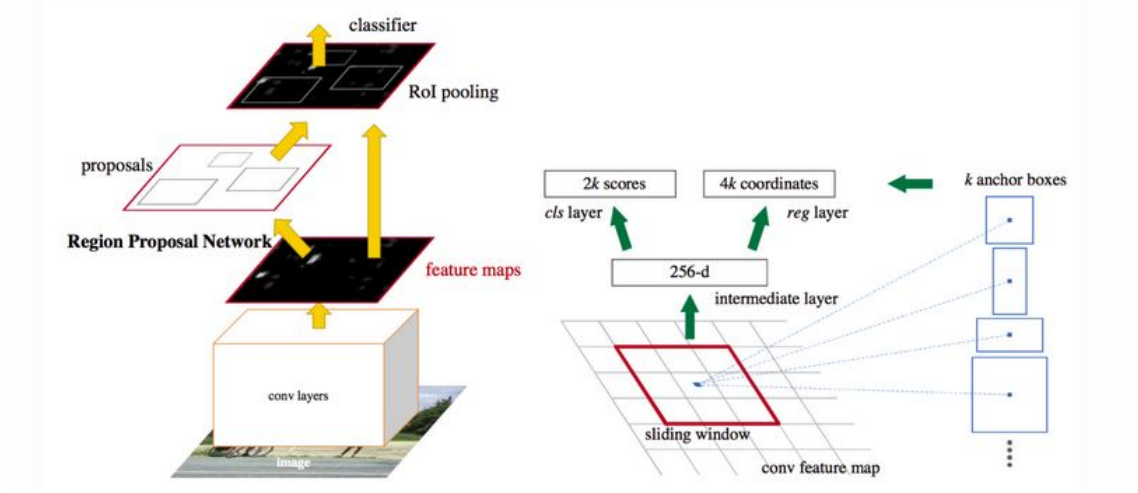


Fig 7: An illustration of Faster R-CNN model.

Region Proposal Network (RPN) produces number of proposals that will be inspected by regressor and classifier to check occurrence and presence of objects. It checks the probability of the anchors of being foreground or background. Probability of object having target object is determined by a classifier and a regression regresses the coordinates of the object in the proposal.

2.6.4 RoI Pooling

It is impossible to make a neural structure that can efficiently deal with features of different size and Region Proposal Network gives convolutional feature map of different size. Convolutional feature maps are reduced to same size with the help of Region of Interest Pooling.

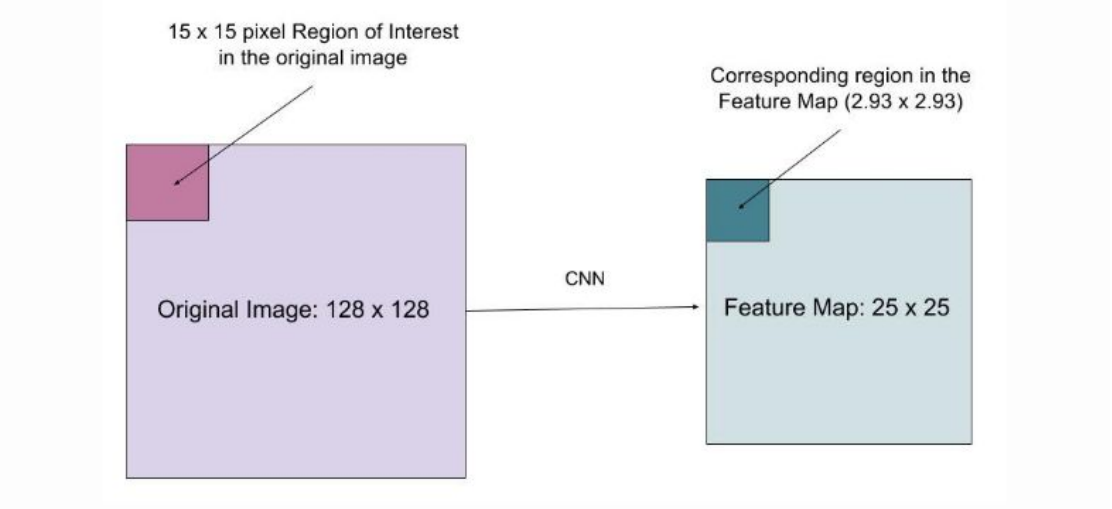


Fig 8: A region of interest is mapped precisely from the first integers onto the feature map without rounding off.

Max pooling is applied on Convolutional Feature map after they have been split by Region of Interest pooling into fixed number of equal regions.

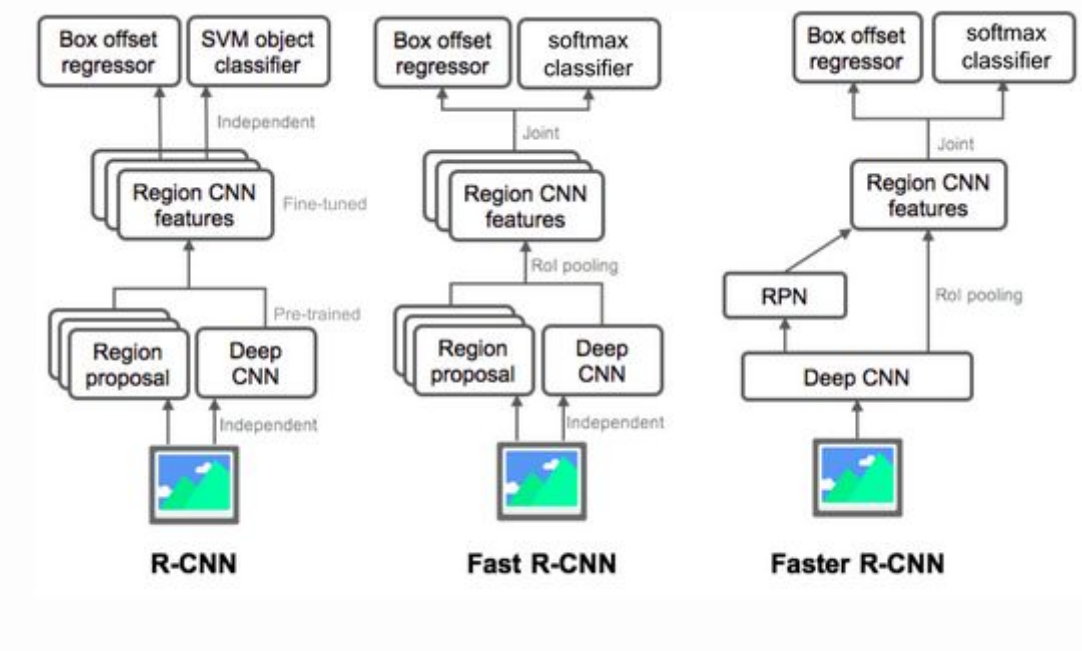


Fig 9: Summary of Models in the R-CNN

2.7 Single Shot MultiBox Detector

Single Shot MultiBox Detector [2] utilizes a single network to identify as well as classify the objects in an image. Unlike Faster R-CNN [3] object detector which utilizes Region Proposal for anticipating the presence of an object, Single Shot MultiBox Detector uses a bounding box and modifies that bounding box.

Various bounding box predictions are accomplished by the last couple of layers of the network accountable for predictions for the progressively tiny bounding box. Union of these predictions is the Ultimate prediction.

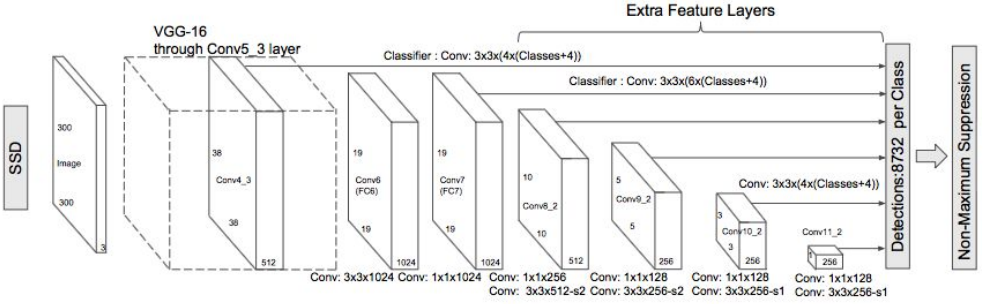


Fig 10: Architecture Of Single Shot MultiBox Detector

2.6.1 MultiBox

In Single Shot MultiBox Detector[2], MultiBox is the name of the technique for bounding box regression developed by Szegedy et al. MultiBox proposes coordinate for the bounding box. MultiBox gives two important elements:

1. Location Loss(l) This corresponds to the difference between the predicted bounding box and ground truth bounding box.
2. Confidence Loss(c) It is a measure of how sure the network is, of object belonging to a particular class.

Multibox box Loss(m), which is how correct our overall prediction is given by:

$$m = c + \alpha l$$

where α is used to balance the contribution of location loss.

Anchors in Faster R-CNN are called priors here, and only those prior are selected who's Jaccard Index is greater than 0.5.

$$Jaccard\ Index\ = \frac{Area\ of\ Overlap\ of\ Bounding\ box}{Area\ of\ Union\ of\ Bounding\ Box}$$

MultiBox endeavors to relapse nearer to the ground truth, but detection begins from prior.

2.8 You Only Look Once

YOLO [1] is a general purpose object detector which adopts a new strategy to solve the problem of object detection. Unlike conventional object detectors YOLO [1], uses classifiers for detecting objects. It redefines the problem of object detection as a regression problem to spatially separated bounding boxes and associated class probabilities as given in [1]. Bounding boxes and class probabilities are predicted by using a single neural network in one pass, sparing a considerable measure of computational time and resources. A single neural network leaves room for further optimization, unlike Faster R-CNN [3] which can be hard to optimize.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig 11: Darknet-53 architecture

The new Darknet-53 architecture acts as the feature extractor. It is mainly comprised of [3 x 3] and [1 x 1] filters with skip connections. Though the revised architecture achieves same classification results, but it is twice as fast.

YOLO [1] makes predictions at three different scales, [13 x 13], [26 x 26] and [52 x 52]. Not only that, it also predicts 5 bounding boxes per grid cell. Therefore, the total bounding box predictions = 3 x (13 x 13 + 26 x 26 + 52 x 52) = 10647. It uses NMS suppression to suppress multiple detections of the same object.

Furthermore, it is extremely easy to implement due to its straightforward system design so it has some leeway over other contemporaries. Be that as it may, this fast object detector makes far too many localization errors than the likes of other detectors such as Faster-RCNN [3] but, the smooth tradeoff you get between speed and accuracy and the potential for optimization are the key factors which make YOLO [1] a really good, general purpose object detector.

2.9 Conclusion

This chapter focused on some of the seminal works which have laid the foundation for modern frameworks, it also discussed some classification and object detection techniques prior to the emergence of deep neural networks, such as KNN classifier, SVM classifier, Softmax classifier, Viola Jones [4], HOG [5], and other machine learning based algorithms. It also gave a brief outlook of modern day object detectors and insight into their working.

CHAPTER 3

SYSTEM DEVELOPMENT

This chapter places heavy emphasis on the process surrounding how the algorithms were implemented, the metrics used for comparison of different algorithms, the platform on which they were implemented, the programming languages in which they were written. The other open source libraries/frameworks that were used have also been enlisted here.

3.1 Software and Hardware Specifications

Below are the software and hardware required for building and running the algorithm.

3.1.1 Software Specifications

All of the algorithms have been written in python-3.6, be it in Jupyter notebooks (*.*ipnyb* files) as well as the *.*py* files. This was a conscious choice because of python's powerful interpreter-- which allows for partial code execution and easy debugging. Not only that, the minimal syntax ensures that the reader does not get bogged down by the syntactic details specific to a programming language.

Python-3.6 also ensures rapid prototyping and immediate execution within Jupyter notebooks without the need for compiling the program time and again with every minor change.

Moreover, the deep learning framework PyTorch, as its name suggests has also been written for python. Instead of just being another library written in some other language, PyTorch is designed for deep integration with Python code. Some advantages of PyTorch over other deep learning frameworks have been listed below:

1. PyTorch makes the most out of CPUs and the computation can also be accelerated by using graphical processing units(GPUs).
2. PyTorch is heavily optimized ad thus incurs minimal computation overhead.

3. The feature that edges the battle in favor of PyTorch is that it supports dynamic neural networks. This is essential when we need our network's behavior to change programmatically at run-time.
4. It is easy to learn and use for machine learning and deep learning algorithms.
5. Open source.

Other external open-source libraries used include--

1. numpy: It is a heavily used for, high-performance, multidimensional-array processing package.
2. matplotlib: This package is basically a plotting library which outputs high quality graphs, plots, figures in a variety of different formats including Jupyter notebooks.
3. tqdm: This package shows the progress of an executing loop as a progress bar, which can essential when training classifiers and object detectors to evaluate the progress.
4. torchvision: This python package provides the user with an interface to obtain and work with popular datasets, model architectures, image transformations in the field of computer vision.

These packages must be installed in the machine on which the algorithms are to be executed or tested. Without these packages, the programs may not lead to the desired behavior. They are absolutely essential for ensuring that the modules work as expected and described here.

On google collaboratory, these packages are already installed and the code can be readily executed.

3.1.2 Hardware Specifications

The algorithms that have been implemented are platform independent, they can run on all Windows/Ubuntu/Mac platforms. That being said, the computation power on these machines may not be enough to satisfy the needed RAM and GPU requirements. Therefore, it is strongly recommended that the algorithms or the Jupyter notebooks

provided should be executed on Google collaboratory, lest your system might crash due to insufficient memory requirements.

Google collaboratory provides every *gmail* user with a private cloud having 12GB RAM and Tesla K80 GPU for computing. However, all local data is periodically wiped out every 12 hours.

3.2 CIFAR-10 Dataset Preprocessing

The CIFAR-10 dataset consists of 60000 color images of 32x32 dimension. It contains 10 classes of objects with 6000 images each. There are 50,000 training images and 10,000 test images.

For tuning hyperparameters of k-NN, SVM and Softmax classifier, the best way is to split the training set into two-- a validation set and a test set. The validation set is slightly smaller as compared to the training set. For this purpose, the training set, which consists of 50,000 images, is split into-- a validation set of 1,000 images and training set of 49,000 images. This validation set is essentially used as a fake testset for tuning hyper-parameters.

3.3 COCO/PASCAL VOC Dataset Preprocessing

For many datasets, the bounding boxes follow a certain pattern. Let's assume the case of autonomous driving, the dataset will have two clusters-- one for bounding boxes for cars and one for bounding boxes of pedestrians Needless to say the former, will be heavily centered in the center of the image, while the latter will cluster on the fringes of the image.

This can lead to improved accuracy when it comes to object detection, for achieving this purpose k-means clustering is used to identify the top k-boundary boxes. These boundary-boxes are called anchors. Anchors are only used by the YOLO detector.

3.4 Training an Object Detector

Training the object detectors requires that our training data must contain the correct answer, so that the predicted output can be evaluated against the correct output to improve the model when it makes incorrect predictions. The following steps have been generalized to train an object detector.

1. Randomly initialize the weights of the neurons approximately close to zero.
2. Each input layer is then fed with the observations in the dataset .
3. Subsequent neurons are activated from the impact of preceding neurons according to their weights.
4. The values are propagated further until they reach to the output layer and get the prediction value of the network 'y'. After which the difference between predicted and actual value is calculated.
5. Backpropagation utilizes the chain rule to estimate the contribution of of each neuron in the produced error and it updates the weights accordingly to reduce it. Learning rate decides by how much weight should be updated.

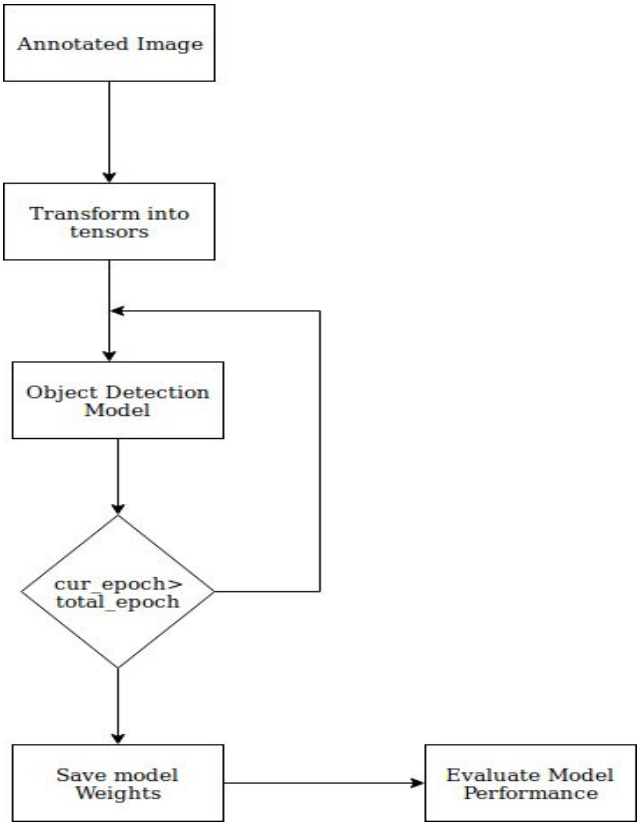


Fig 12: Generalised Pipeline of Object Detection

CHAPTER 4

PERFORMANCE ANALYSIS.

4.1 Datasets

Datasets such as CIFAR-10, COCO [17], PASCAL VOC [18] are standard datasets for image classification(CIFAR-10), object detection(PASCAL VOC, COCO), and image segmentation(COCO) empowering us to evaluate and compare the accuracy of different methods.

Nonetheless, they do not provide us with any useful information about the inference time of the different techniques employed.

4.1.1 CIFAR-10 Dataset

CIFAR-10 dataset consists of 60000 color images of 32x32 dimension. It contains 10 classes of objects with 6000 images each. There are 50000 training images and 10000 test images. The classes have been listed below:

plane	car	bird	cat	deer	dog	frog	horse	ship	truck

Table 2: Classes in CIFAR-10

Fig 13: Random Samples from CIFAR-10

All classes are mutually exclusive, i.e., it does not contain images that overlap between automobiles and trucks. The class "Truck" includes trucks that are big, while the class "Automobile" includes sedans, SUVs. Neither of the two classes include pickup trucks.

4.1.2 PASCAL VOC Dataset

Pascal VOC is natural datasets used image classification and object detection problems.

The datasets that have been used here are VOC2012 and VOC2007 which consists of 20000 images having 20 different classes listed below:

person	table	cat	cow	dog	horse	sheep	plane	bus	cycle
boat	bottle	car	bike	train	bird	chair	plant	sofa	tv

Table 3: Classes in Pascal VOC

4.1.3 COCO Dataset

The COCO train, validation, and test sets, contain more than 200,000 images and 80 object classes. All object instances present in the image have annotations for validating the predictions. For object classification, most models make use of a softmax layer, which makes the assumption that all the classes are mutually exclusive. However, the labels from the COCO datasets are not mutually exclusive. Therefore, most models trained on the COCO dataset opt for logistic regression instead of using a softmax layer.

4.2 k-NN, SVM and Softmax Classifier

After training k-NN, SVM and Softmax classifier on CIFAR-10 dataset, following results were obtained when the classifiers were tested on the test dataset which contains 10,000 images.

Classifier	Accuracy
k-NN, k=10	28.79%
SVM	36.19%
Softmax	35.5%

Table 4: Classification results on CIFAR-10

k-NN classifier is extremely poor when it is tested on the training set, even after tuning the hyperparameters and using 5 fold cross-validation to get the best k=10 metric. Furthermore, it takes O(N) time to predict the label for any given image.

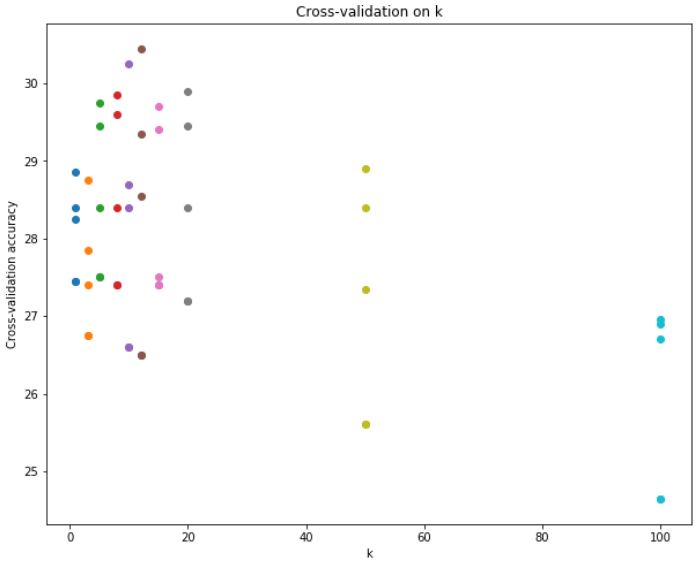


Fig 14 : k-NN Cross validation on k

SVM classifier does outperform the k-NN classifier, but the performance gain is very little. Moreover, the scores predicted by the SVM classifier do not tell us how strongly the image belongs to that a particular class.

Fig 15 : Templates learnt by SVM Classifier

Softmax classifier is very competitive with the SVM classifier, but marginally loses out. The advantage of using a softmax classifier over SVM is that the scores predicted by the softmax classifier are probability distribution for the given classes in the training dataset. That being said, softmax classifier cannot be used when the classes are not linearly separable.

Fig 16: Templates learnt by Softmax Classifier

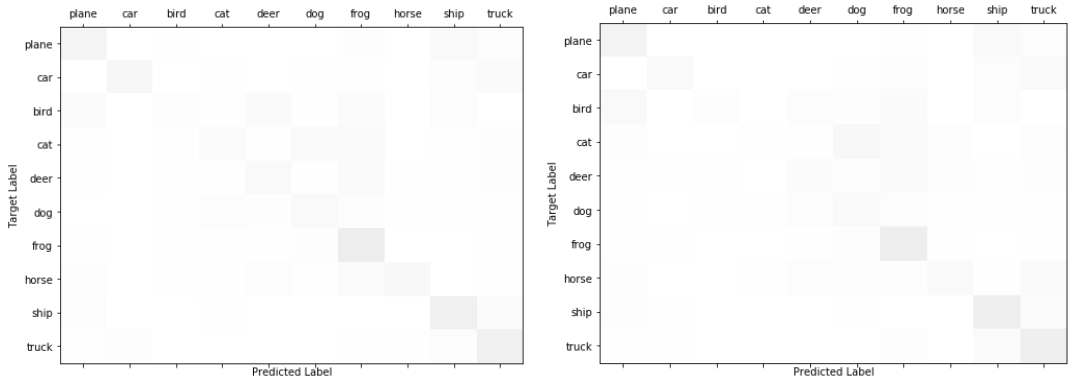


Fig 17: Confusion Matrix-SVM(left), Softmax(right)

4.3 VGG16 Classifier

Single shot multibox detector utilises slightly enhanced version of VGG16, its classification results on CIFAR-10 dataset will be adept in providing intuition about how SSD does the classification.

Following are the classification results obtained when the neural network architecture is trained on the CIFAR-10 dataset with a batch size of 512 for 20 epochs. The learning rate remains 0.1 for first 10 epochs and 0.01 for the next 10 epochs.

The learning rate is changed from 0.1 to 0.01 because after going through the dataset 10 times, our accuracy starts to stagnate.

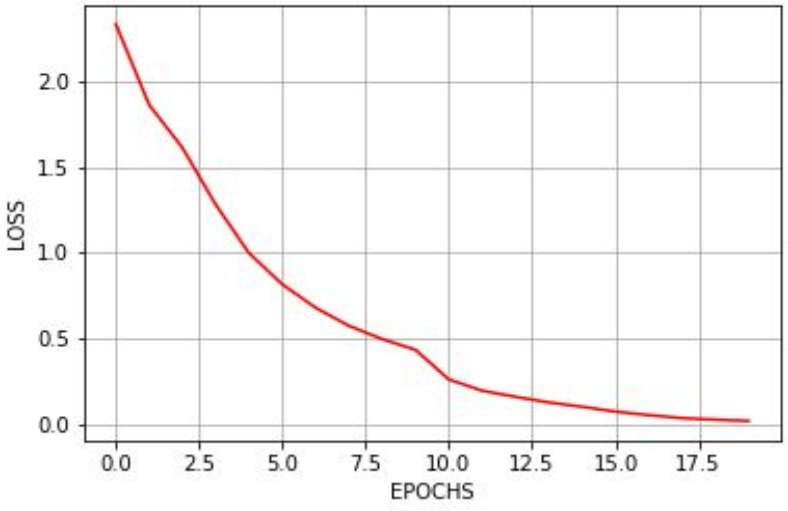


Fig 18 : VGG16 Classifier Epoch vs Loss Graph

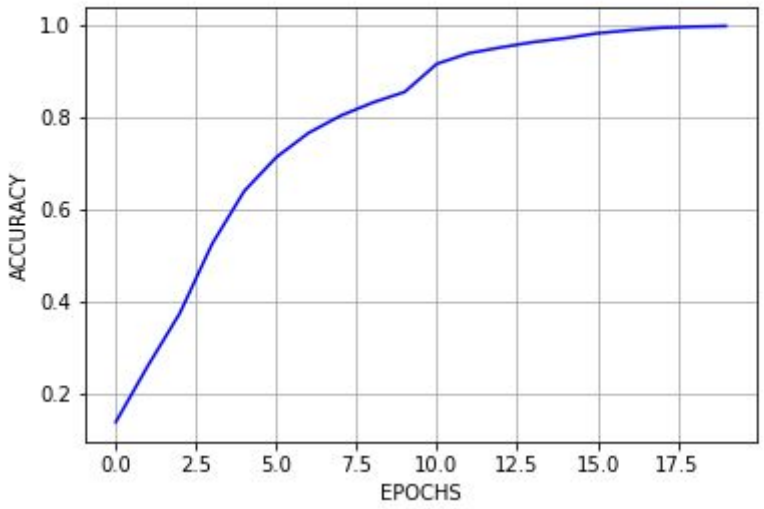


Fig 19: VGG16 Classifier Epoch vs Accuracy Graph

From the above graphs, it can concluded that as our classifier goes through the dataset multiple times our loss decreases and accuracy increases.

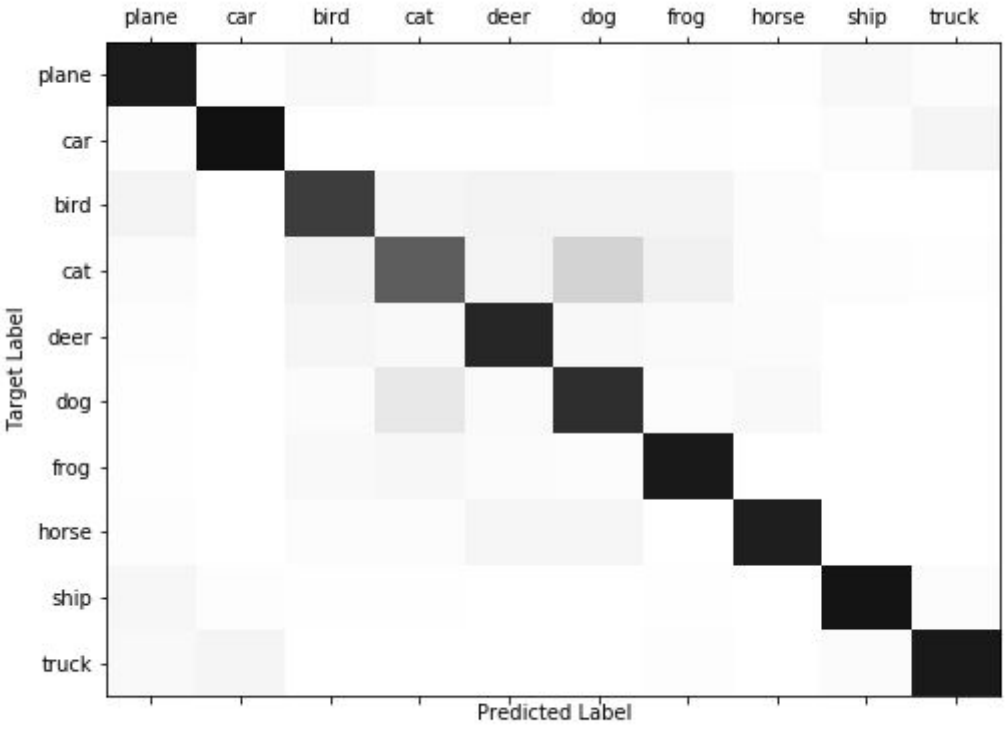


Fig 20: VGG16 Confusion Matrix on CIFAR-10

VGG16 classifier does well on most classes as compared to the previous classifiers, but it does label the images with dogs as cats and cats as dogs, which can be due to the fact that they

might appear visually similar to the classifier because of their fur coats. For the class plane, it also predicts the label as bird or ship-- which might be as a result of the blue background the images seem to share.

Criteria	Time/Epoch(s)
Maximum Time	29.94
Minimum Time	28.41
Average Time	29.09

Table 5: Time/Epoch for VGG16

The Time/Epoch values have been computed on google cloud which has 12GB RAM and Tesla K80 GPU.

4.4 Darknet Classifier

Since, Darknet is the backbone of the YOLO [1] detector, let's take a look at how well it fares off on the CIFAR-10 dataset. The classification results on CIFAR-10 dataset gives us a better insight about how well it classifies objects in actual YOLO [1] detector.

Following are the classification results obtained when the neural network architecture is trained on the CIFAR-10 dataset with a batch size of 512 for 20 epochs. The learning rate remains 0.1 for first 10 epochs and 0.01 for the next 10 epochs.

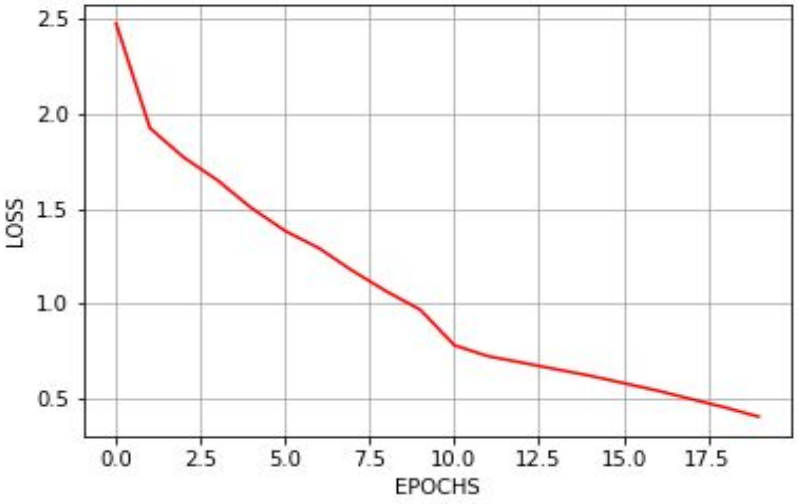


Fig 21 : Darknet Classifier Epoch vs Loss Graph

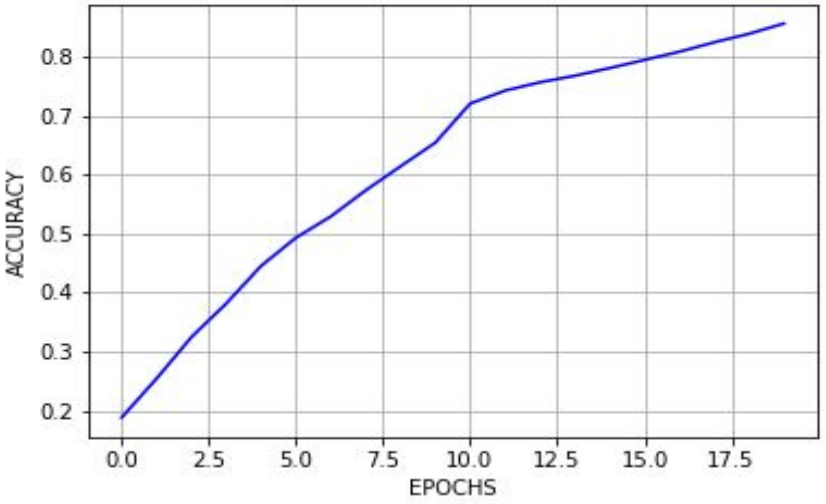


Fig 22: Darknet Classifier Epoch vs Accuracy

From the above graphs, it can be concluded that as our classifier goes through the dataset multiple times our loss decreases and accuracy increases.

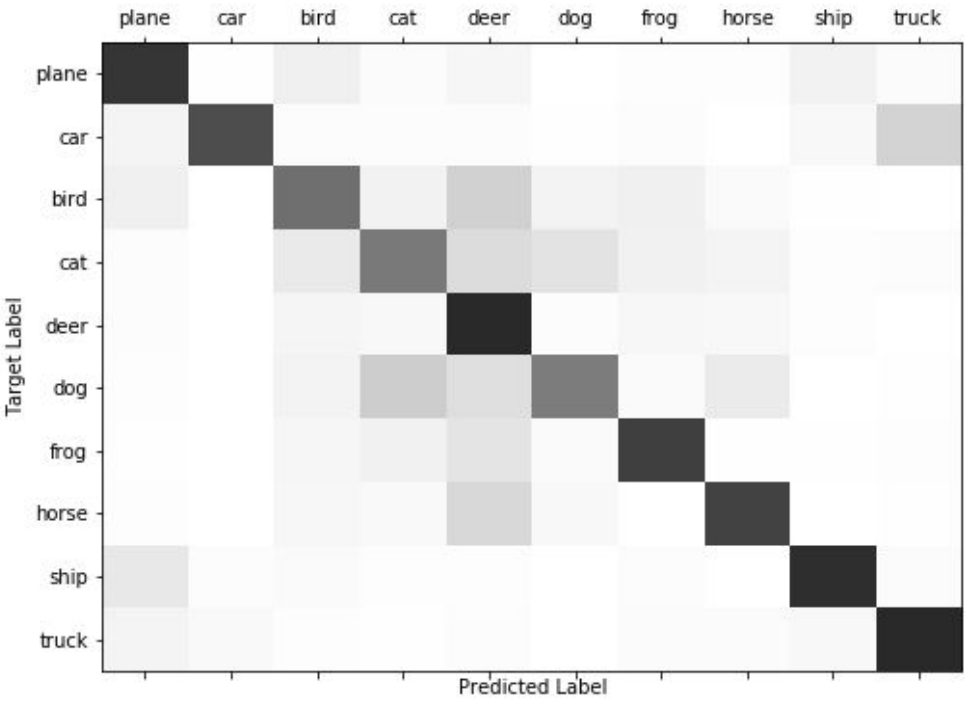


Fig 23: Darknet Confusion Matrix on CIFAR-10

Just like the VGG16, the Darknet classifier seems to do well on most classes except for bird, horse and the dog class. It confuses cats with dogs and vice-versa, horse with deer, presumably due to their brown coats. Car also gets misclassified as Trucks. However, it outperforms k-NN, SVM and Softmax by a wide margin quite easily, once it has been trained and is competitive with the results produced by VGG-16 classifier.

Criteria	Time/Epoch(s)
Max Time	354.75
Min Time	351.62
Avg Time	352.79

Table 6: Time/Epoch for Darknet

The Time/Epoch values have been computed on google cloud which has 12GB RAM and Tesla K80 GPU.

4.5 Comparative Analysis of Classifiers

The table given below compares the accuracy of different classifiers on the CIFAR-10 dataset for all the classes. As you can infer from the table, the classifiers which use CNN for feature extraction do much better than the ones which depend on template matching.

Classes	Darknet-19	VGG16	k-NN, k=10	SVM	Softmax
Plane	89.6	89.1	43.7	39.8	47.6
Car	87.7	92.8	10.1	39.3	29.2
Bird	67.9	75.5	42.0	14.0	9.0
Cat	70.1	63.3	16.5	18.4	5.8
Deer	80.4	85.1	40.0	26.7	21.1
Dog	60.1	82.0	11.6	27.9	30.2
Frog	90.4	89.8	25.0	65.2	61.6
Horse	82.5	88.2	15.7	29.4	24.5
Ship	88.4	91.9	71.7	56.6	59.4
Truck	89.4	89.7	8.3	55.0	60.6
Mean Accuracy	**80.65**	**84.47**	**28.79**	**38.0**	**35.8**

Table 7: Classifier accuracy per class

Criteria	VGG16 Time/Epoch(s)	Darknet Time/Epoch(s)
Max Time	29.94	354.75
Min Time	28.41	351.62
Avg Time	29.09	352.79

Table 8: Time/Epoch for VGG16 and Darknet

Darknet-19 classifier takes more time to go through the dataset as compared to the VGG16 classifier because it has more convolutional layers than VGG16 which only contains 16 convolutional layers as compared to Darknet-19's 38.

4.6 Result Analysis of SSD

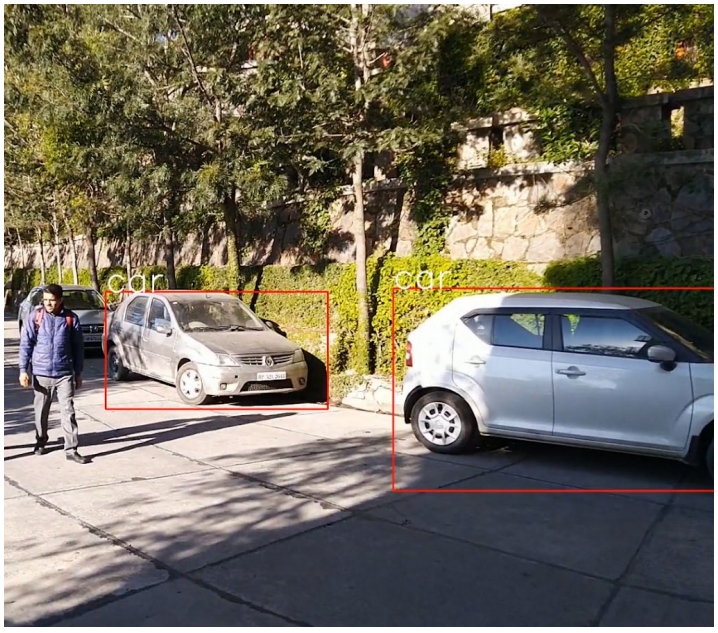
We have implemented the previously stated object detection model, namely, SSD[2] using the architecture of VGG_16. The results obtained from training the SSD object detector on custom data are shown below:

Input **Output**

Fig 24: Robust detection of occluded objects

As can be seen from the output images above, the system not only handles varied illumination, but also manages to detect objects even in slightly occluded state. The two cars, are under different degree of illumination, the front one is brightly illuminated while the one behind that, is occluded as well as under shade, but is still getting detected.

That being said, larger objects dominated when present along with small objects as found in Fig 14. This could be the reason for the average precision of smaller objects to be less when compared to larger objects.



(a) Small and large object in the image

(b) Only small object in the image

Fig 25: Domination of larger object in detection

4.7 Comparative Analysis of YOLO and SSD

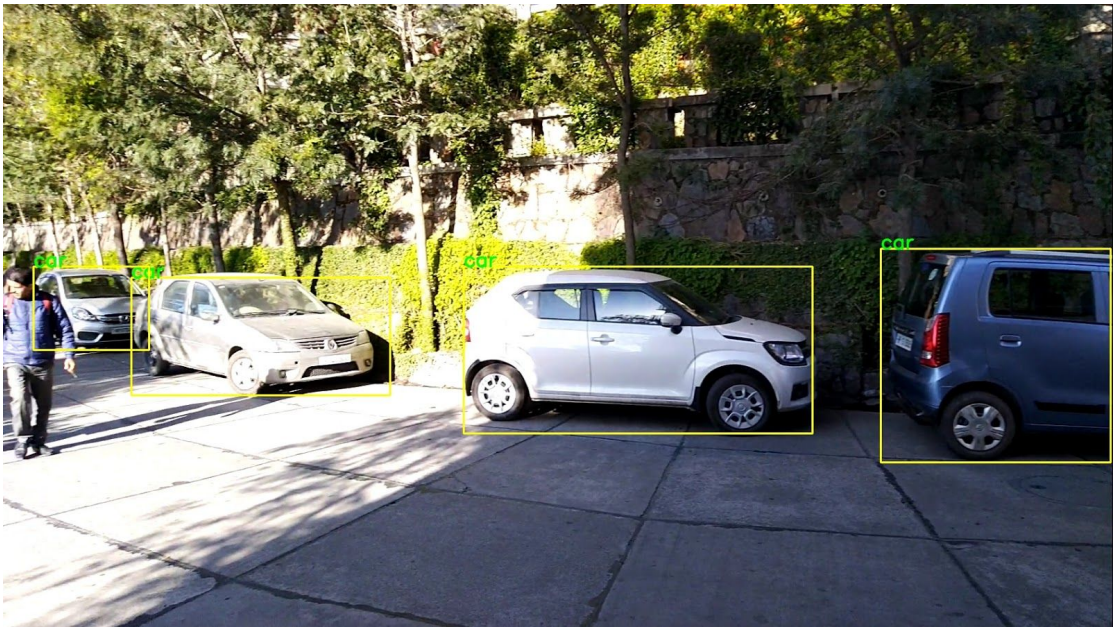


Fig 26: SSD output

Fig 27: YOLO output

From the above images it can be inferred that YOLO is more robust when it comes to detecting smaller objects when larger objects are dominant in that image.

Fig 28: SDD(left) and YOLO(on right) outputs side by side

It can be concluded from the above images that SSD is more flexible when object features are slightly altered as SSD is correctly detecting the bottle which is different from the ones it was trained on.

Fig 29: Input Image

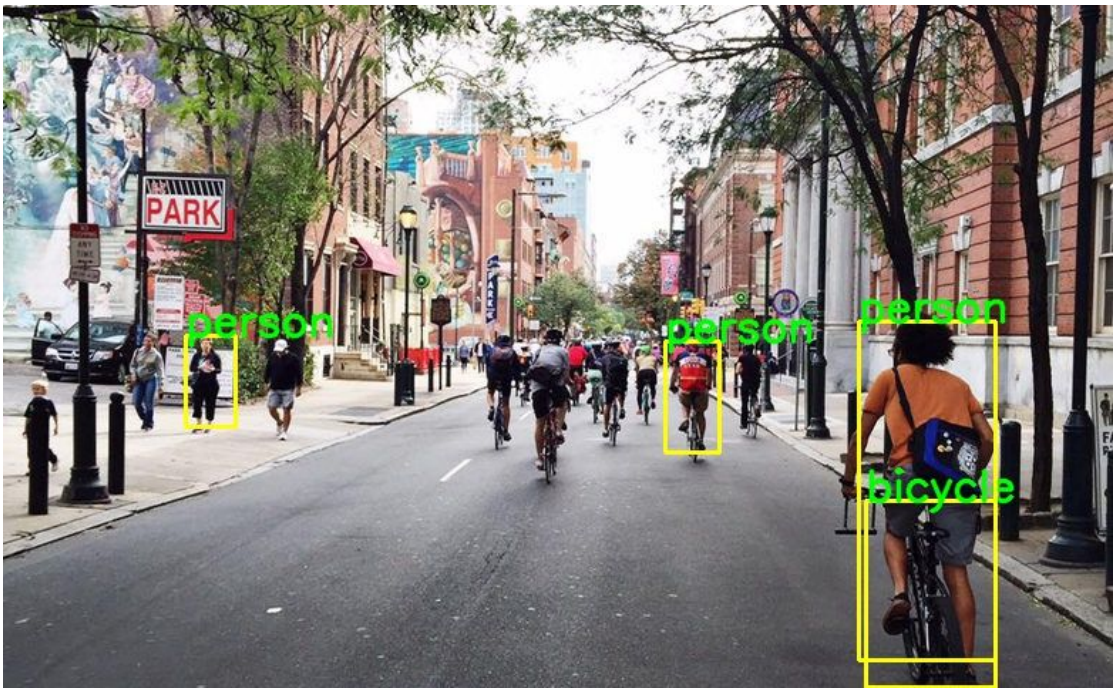


Fig 30: Single Shot Output

Fig 31: YOLO results

YOLO is more reliable when there is huge concentration of objects in the image. It is better suited for images with high occlusion as can be seen with the car- with 0.9 probability highlighted in green towards the right.

Criteria	SSD Time/Image(s)	YOLO Time/Image(s)
Max Time	2.028	1.98
Min Time	1.423	1.5
Avg Time	1.593	1.624

Table 9: SSD and YOLO Inference Time

From the above results, it can be inferred that YOLO is more robust when it comes to detecting smaller objects when larger objects are dominant in that image. It can also be concluded from the above results that while SSD is more flexible when object features are slightly altered, YOLO is more reliable when there is huge concentration of objects in the image. It is better suited for images with high occlusion.

Both are SSD and YOLO are good for real time object detection when provided with strong Graphical processing units but SSD is slightly faster than YOLO.

CHAPTER 5
CONCLUSION

We have trained k-NN, SVM and Softmax, darknet-19 and VGG16 classifier on CIFAR-10 dataset and as we had expected darknet-19 and VGG16 clearly outperform the other 3 by a wide margin.

k-NN classifier is extremely poor when it is tested on the training set, even after tuning the hyperparameters and using 5 fold cross-validation to get the best k=10 metric. Furthermore, it takes O(N) time to predict the label for any given image.

SVM classifier does outperform the k-NN classifier, but the performance gain is very little. Moreover, the scores predicted by the SVM classifier do not tell us how strongly the image belongs to that a particular class.

Softmax classifier is very competitive with the SVM classifier, but marginally loses out. The advantage of using a softmax classifier over SVM is that the scores predicted by the softmax classifier are probability distribution for the given classes in the training dataset. That being said, softmax classifier cannot be used when the classes are not linearly separable.

We can infer that YOLO is more robust when it comes to detecting smaller objects when larger objects are dominant in that image. It can also be concluded from the above results that while SSD is more flexible when object features are slightly altered, YOLO is more reliable when there is huge concentration of objects in the image. It is better suited for images with high occlusion.

Both are SSD and YOLO are good for real time object detection when provided with strong Graphical processing units but SSD is slightly faster than YOLO.

5.1 Future Scope

Object detection can be applied in many fields from autonomous driving-- where it allows for real-time video processing and aided-driving, to security surveillance using object tracking.

While the progress that has been made in the field of Computer Vision, it still has a long way to go. One of the key areas to address in future researches is to extract context from the image.

Fig 32: Barack Obama Jesting

For example, in the above image, even an extremely well trained object with a perfect mAP score will be unable to tell why this picture is so funny. Not only, Barack Obama was the United State's president at that time, he is also pulling the leg of his secret-service/staff by adding weight to the scale. The other people in the image seem to be enjoying at the other guy's expense.

More efforts need to be concentrated in this area because it would be truly remarkable to have a model that is able to decipher context from the input images.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015.

[3] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[4] P. Viola, M. Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, in: Computer Vision and Pattern Recognition, 2001.

[5] Dalal, N. and Triggs, B. Histograms of oriented gradients for pedestrian detection. In CVPR, 2005.

[6] Object Detection with Deep Learning: A Review Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE

[7] Bhuravarjula, Hari Hara Pavan Kumar, and VNS Vijaya Kumar. "A Novel Image Retrieval Method using color Moments."International Journal Of Electronics And Computer Science Engineering (Ijecse, ISSN: 2277-1956) 1.04 (2012): 2432-2438.

[8] O. Chum, A. Zisserman, An Exemplar Model for Learning Object Classes, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[9] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” J. of Comput. & Sys. Sci., vol. 13, no. 5, pp. 663–671, 1997.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan,“Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, pp. 1627–1645, 2010.

[11] C. Cortes and V. Vapnik, “Support vector machine,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995

[12] Beng Yong, Lee & Liew, Lee & S. Cheah, W & C. Wang, Y. (2014). Occlusion handling in videos object tracking: A survey. IOP Conference Series: Earth and Environmental Science. 18. 10.1088/1755-1315/18/1/012020.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representation by back-propagation of errors,” Nature, vol. 323, no. 323, pp. 533–536, 1986.

[14] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in ICML, 2010.

[15] R. Girshick, “Fast r-cnn,” in ICCV, 2015.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge International Journal of Computer Vision, 115(3):211–252, 2015.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ra-manan, P. Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In ECCV, 1 May 2014.

[18] The PASCAL Visual Object Classes Challenge: A Retrospective, Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. International Journal of Computer Vision, 111(1), 98-136, 2015

[19] Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop, number EPFL-CONF-192376, 2011