RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	19CSC312A
Course Name	Artificial Intelligence
Programme	B.Tech.
Department	Computer Science and Engineering
Faculty	FET

Name of the Student	Tanishq Porwar
Reg. No	18ETCS002131
Semester/Year	6TH / 2018
Course Leader/s	

Declaration Sheet				
Student Name	Tanishq Porwar			
Reg. No	18ETCS002131			
Programme	B.Tech.		Semester/Year	6th /2018
Course Code	19CSC312A			
Course Title	Artificial Intelligence			
Course Date		to		
Course Leader				

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Contents

Faculty of Engineering and Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	Programme	B. Tech.
Semester/Batch	6th/2018		
Course Code	19CSC312A	Course Title	Artificial Intelligence
Course Leader(s)	Dr. Subarna Chatterjee, Gp. Capt. Rath and Santoshi Kumari		

Assignment-1						
Register No		18ETCS002131	Name of Student		Tanishq Porwar	
Question	Marking Scheme			Marks		
				Max Marks	First Examiner Marks	Moderator
1						
	1	Executive summary		**03**		
	2	Background and Objectives		**04**		
	3	Comparative analysis of state-of-the-art methods		**10**		
	4	Conclusion and Recommendations		**03**		
	5	Presentation		**05**		
	Max Marks			**25**		
Total Assignment Marks				25		

Course Marks Tabulation				
Question	**First Examiner**	**Remarks**	**Moderator**	**Remarks**
1				
Marks (Max 25)				

Signature of First Examiner **Signature of Moderator**

Question No. 1

Solution to Question No. 1:

1.1 Executive Summary

The problem taken by me for this assignment is that of real-time object detection. Object detection is one of the most important computer vision problem, with a multitude of applications, it has been one of the primary areas of research in the computer vision domain, solutions based on deep learning such as Region-Based Convolutional Neural Network(R-CNN), have been proposed.

Object detection is a computer vision technique that works to identify and locate objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene. Object detection is inextricably linked to other similar computer vision techniques like image recognition and image segmentation, in that it helps us understand and analyze scenes in images or video.

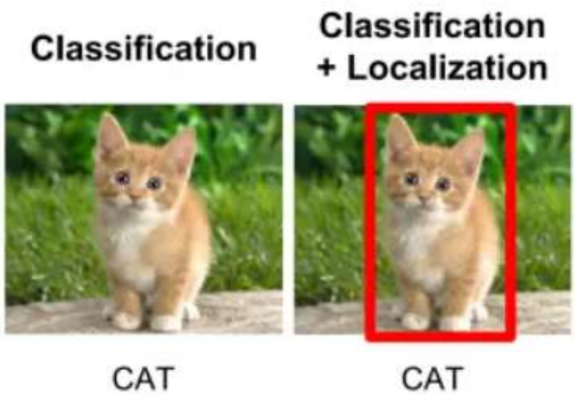

Figure 1: object classification vs object detection

But there are important differences. Image recognition only outputs a class label for an identified object, and image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability to *locate objects within an image or video*. This then allows us to count and then track those objects.
Given these key distinctions and object detection's unique capabilities, we can see how it can be applied in a number of ways:

- Crowd counting
- Self-driving cars
- Video surveillance
- Face detection
- Anomaly detection
- Human pose estimation

The challenges faced while designing object detection algorithms are:

- Speed
- Memory efficiency
- Accuracy

In this assignment we are going to find the best possible combination of speed and accuracy while comparing different state-of-the-art object detection algorithms to perform object detection. Models which are usually

computationally expensive, achieve the best accuracy, but cannot be deployed in a simple setting with limited resources, whereas, faster models fail to achieve similar results compared to their bigger and more memory intensive counterparts.

1.2 Background and Objectives

Object detection methods are broadly classified into two type:

- machine learning-based approaches
- deep learning-based approaches

1. **Machine learning-based approaches**

   Computer vision is an integral part of many robotic applications. In the 90s, there was a rise of feature descriptors (SIFT, SURF) as the primary technique used to solve a host of computer vision problems (image classification, object detection, face recognition). Often these feature descriptors are combined with traditional machine learning classification algorithms such as Support Vector Machines and K-Nearest Neighbors to solve these computer vision problems.
   Computer vision techniques are used to look at various features of an image, such as the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into a regression model that predicts the location of the object along with its label.

   One such algorithm is Object Detection using Hog Features, in a groundbreaking paper in the history of computer vision, Navneet Dalal and Bill Triggs introduced Histogram of Oriented Gradients (HOG) features in 2005. Hog features are computationally inexpensive and are good for many real-world problems. On each window obtained from running the sliding window on the pyramid, we calculate Hog Features which are fed to an SVM (Support vector machine) to create classifiers. We were able to run this in real time on videos for pedestrian detection, face detection, and so many other object-detection use-cases.

2. **Deep learning-based approaches**

   Since we had modeled object detection into a classification problem, success depends on the accuracy of classification. After the rise of deep learning, the obvious idea was to replace HOG based classifiers with a more accurate convolutional neural network-based classifier. Deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, in which features don't need to be defined and extracted separately.

   Because deep learning methods have become the state-of-the-art approaches to object detection, we will be focusing on deep-learning based algorithms in this assignment.

Basic structure of Deep learning-based approaches

Deep learning-based object detection models typically have two parts. An **encoder** takes an image as input and runs it through a series of blocks and layers that learn to extract statistical features used to locate and label objects. Outputs from the encoder are then passed to a **decoder**, which predicts bounding boxes and labels for each object.

The simplest decoder is a pure regressor. The regressor is connected to the output of the encoder and predicts the location and size of each bounding box directly. The output of the model is the X, Y coordinate pair for the object and its extent in the image. Though simple, this type of model is limited. You need to specify the number of boxes ahead of time. If your image has two dogs, but your model was only designed

to detect a single object, one will go unlabelled. However, if you know the number of objects you need to predict in each image ahead of time, pure regressor-based models may be a good option.

The basic idea is that we will take a raw RGB image and perform a series of transformations on the image. On each transformation, we learn a denser representation of the image. We then take this denser representation, apply the transformation and learn an even denser representation. It turns out by using this procedure, we learn more and more abstract features with which to represent the original image. At the end, we can use these abstract features to predict using some traditional classification method

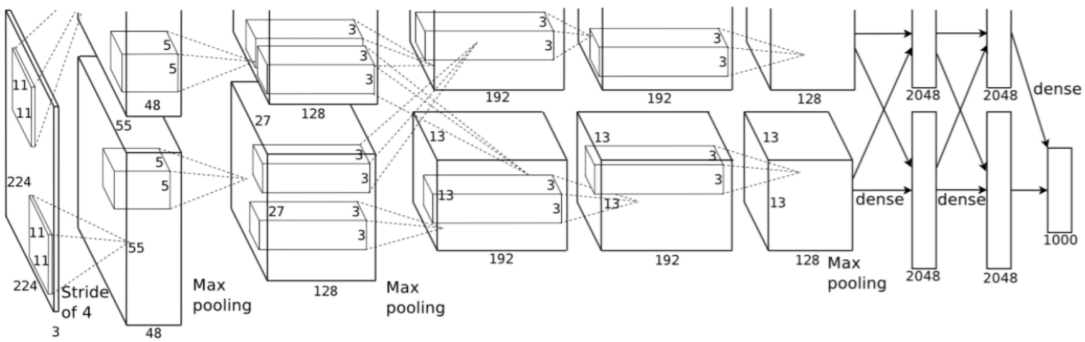

Figure 2: working of CNN

An extension of the regressor approach is a **region proposal network**. In this decoder, the model proposes regions of an image where it believes an object might reside. The pixels belonging to these regions are then fed into a classification subnetwork to determine a label (or reject the proposal). It then runs the pixels containing those regions through a classification network. The benefit of this method is a more accurate, flexible model that can propose arbitrary numbers of regions that may contain a bounding box. The added accuracy, though, comes at the cost of computational efficiency.

Single shot detectors (SSDs) seek a middle ground. Rather than using a subnetwork to propose regions, SSDs rely on a set of predetermined regions. A grid of anchor points is laid over the input image, and at each anchor point, boxes of multiple shapes and sizes serve as regions. For each box at each anchor point, the model outputs a prediction of whether or not an object exists within the region and modifications to the box's location and size to make it fit the object more closely. Because there are multiple boxes at each anchor point and anchor points may be close together, SSDs produce many potential detections that overlap. Post-processing must be applied to SSD outputs in order to prune away most of these predictions and pick the best one. The most popular post-processing technique is known as **non-maximum suppression**.

Why a standard convolutional neural network isn't used?
The major reason why we cannot proceed with this problem by building a standard convolutional network followed by a fully connected layer is that, the length of the output layer is variable — not constant, this is because the number of occurrences of the objects of interest is not fixed. A naive approach to solve this problem would be to take different regions of interest from the image, and use a CNN to classify the presence of the object within that region. The problem with this approach is that the objects of interest might have different spatial locations within the image and different aspect ratios. Hence, you would have to select a huge number of regions and this could computationally blow up. Therefore, algorithms like R-CNN, YOLO etc. have been developed to find these occurrences and find them fast.

1.3 Comparative analysis of state-of-the-art methods

In this assignment we will consider models which are quite similar in their architecture on a high level overview. The models that are considered are:

- Region based Convolutional Neural Network (R-CNN)
- Region based Fully Convolutional networks (R-FCN)
- Spatial Pyramid Pooling (SPP-net)
- Single Shot Detector (SSD)
- You Only Look Once (YOLO)

Evaluation metric

- mAP (mean Average Precision)
- memory requirement
- testing time

1. R-CNN

   The R-CNN model was one of the first models to use convolutional neural networks for object detection. The goal of R-CNN is to take in an image, and correctly identify where the main objects (via a bounding box) in the image. But how do we find out where these bounding boxes are? R-CNN does what we might intuitively do as well -propose a bunch of boxes in the image and see if any of them actually correspond to an object. R-CNN creates these bounding boxes, or region proposals using a process called Selective Search. Selective search, looks at the image through windows of different sizes, and for each size tries to group together adjacent pixels by texture, color, or intensity to identify objects. Once the proposals are created, R-CNN warps the region to a standard square size and passes it through to a feature extractor or image classifier, which is a CNN. On the final layer of the CNN, RCNN adds a Support Vector Machine (SVM) that simply classifies whether this is an object, and if yes, which object is it.

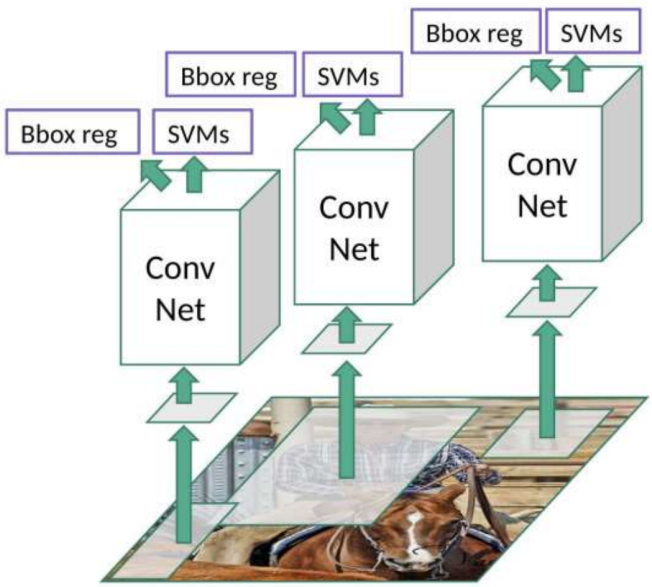

Figure 3: working of R-CNN

Problems with R-CNN

- It still takes a huge amount of time to train the network as you would have to classify a large number of region proposals per image.

- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

2. SPP-net

Still, RCNN was very slow. Because running CNN on 2000 region proposals generated by Selective search takes a lot of time. SPP-Net tried to fix this. With SPP-net, we calculate the CNN representation for entire image only once and can use that to calculate the CNN representation for each patch generated by Selective Search. This can be done by performing a pooling type of operation on JUST that section of the feature maps of last conv layer that corresponds to the region. The rectangular section of conv layer corresponding to a region can be calculated by projecting the region on conv layer by taking into account the downsampling happening in the intermediate layers(simply dividing the coordinates by 16 in case of VGG).

There was one more challenge: we need to generate the fixed size of input for the fully connected layers of the CNN so, SPP introduces one more trick. It uses spatial pooling after the last convolutional layer as opposed to traditionally used max-pooling. SPP layer divides a region of any arbitrary size into a constant number of bins and max pool is performed on each of the bins. Since the number of bins remains the same, a constant size vector is produced as demonstrated in the figure below.

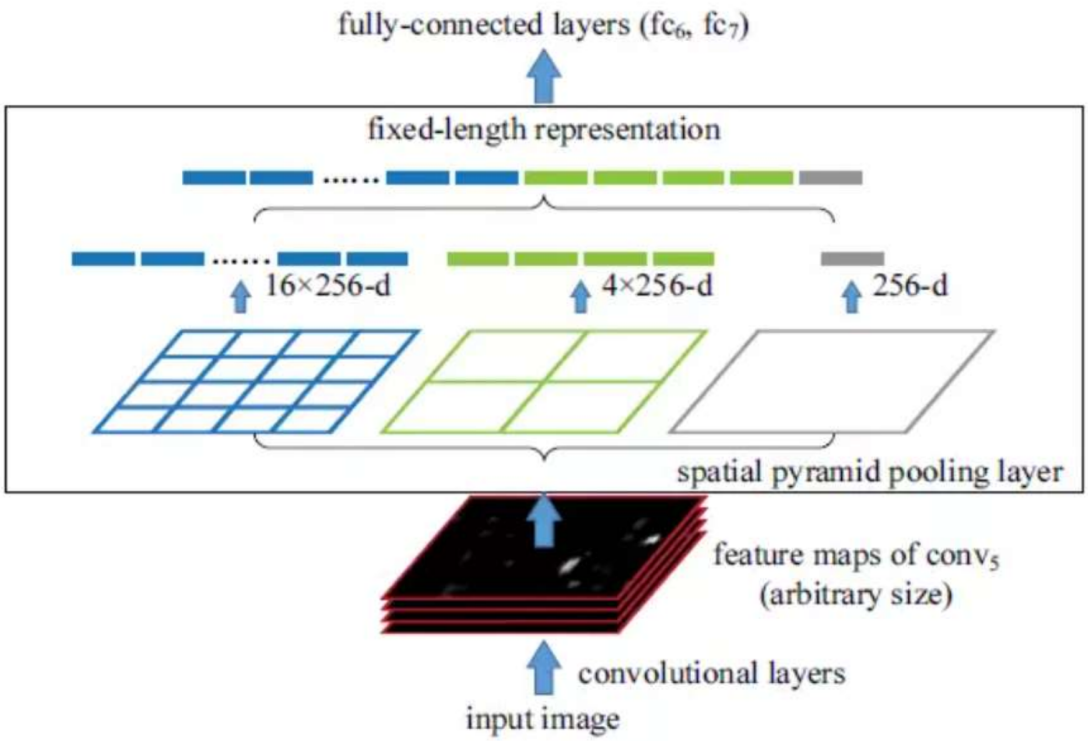

Figure 4: working of SPP-net

Drawbacks:

However, there was one big drawback with SPP net, it was not trivial to perform back-propagation through spatial pooling layer. Hence, the network only fine-tuned the fully connected part of the network.

3. Fast R-CNN

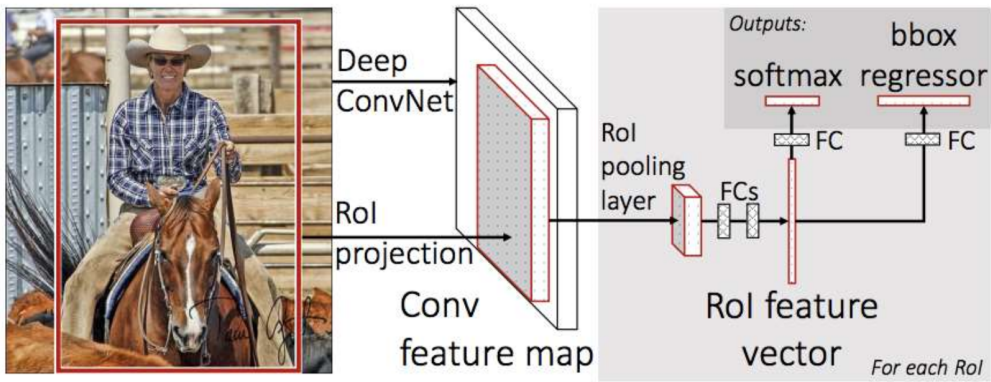

Figure 5: working of fast R-CNN

Fast RCNN uses the ideas from SPP-net and RCNN and fixes the key problems. The problems of R-CNN were solved in Fast R-CNN by the creator of R-CNN himself, the approach is similar to the R-CNN algorithm. He realized that for each image, a lot of proposed regions for the image invariably overlapped causing us to run the same CNN computation again and again. His insight was simple — Why not run the CNN just once per image and then find a way to share that computation across the proposals? This is exactly what Fast R-CNN does, using a technique known as RoI Pool (Region of Interest Pooling). At its core, RoI Pool shares the forward pass of a CNN for an image across its subregions. In the image above, notice how the CNN features for each region are obtained by selecting a corresponding region from the CNN's feature map. Then, the features in each region are pooled (usually using max pooling). So, all it takes us is one pass of the original image. The second insight of Fast RCNN is to jointly train the CNN, classifier, and bounding box regressor in a single model. Where earlier we had different models to extract image features (CNN), classify (SVM), and tighten bounding boxes (regressor), Fast RCNN instead used a single network to compute all three. Fast R-CNN replaced the SVM classifier with a softmax layer on top of the CNN to output a classification. It also added a linear regression layer parallel to the softmax layer to output bounding box coordinates. In this way, all the outputs needed came from one single network

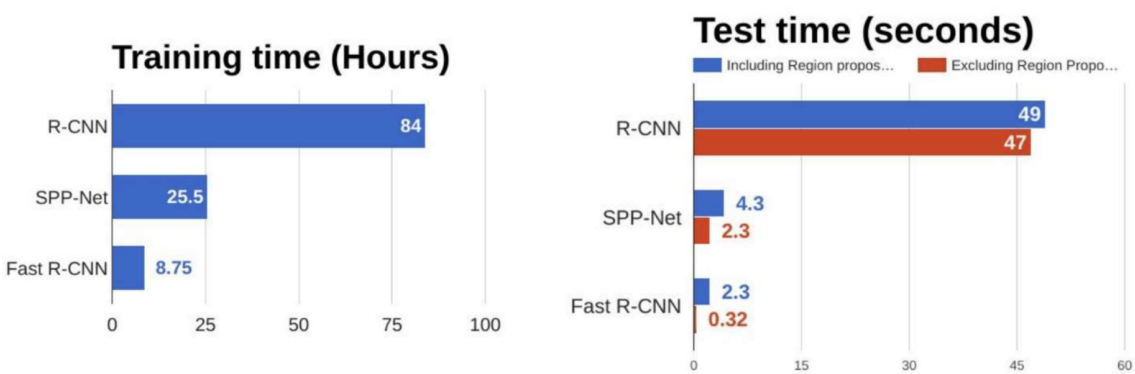

Figure 6: Speed comparison of R-CNN, Fast R-CNN and SPP-net

Drawbacks:

There was still one bottleneck with Fast R-CNN which had to be sorted, that was the region proposer. The very first step to detecting the locations of objects is generating a bunch of potential bounding boxes or regions of interest to test. In Fast R-CNN, these proposals were created using Selective Search, a fairly slow process that was found to be the bottleneck of the overall process

4. Faster R-CNN

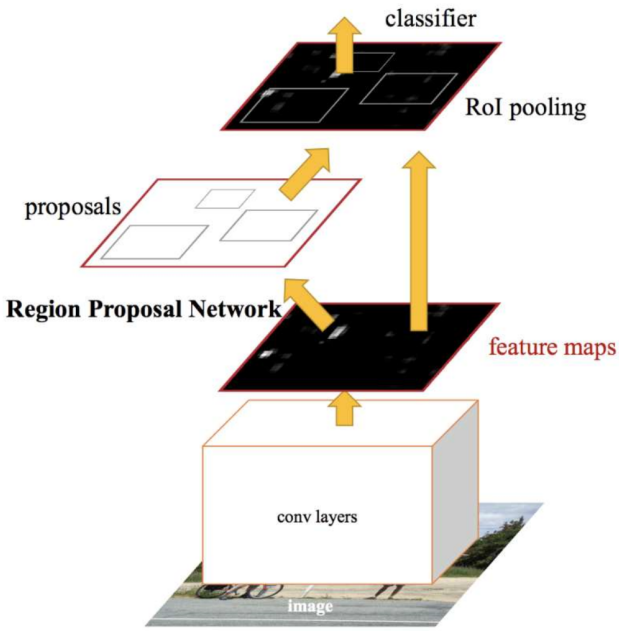

Figure 7: working of Faster R-CNN

Both R-CNN & Fast R-CNN uses selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, Shaoqing Ren et al. came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals.

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

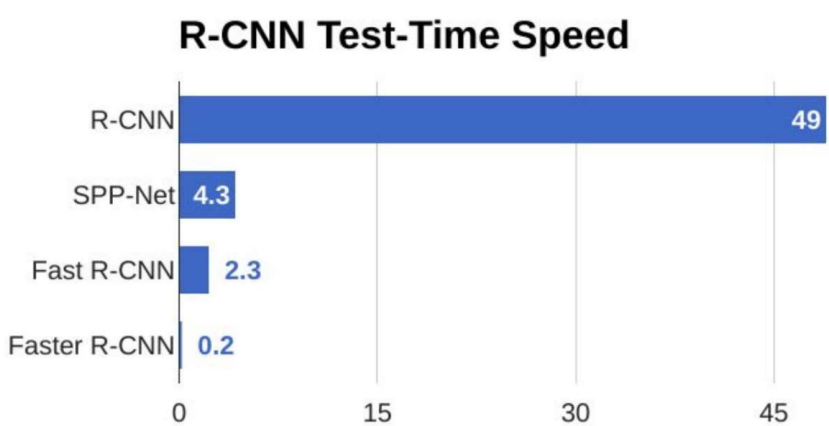

Figure 8: Speed comparison of R-CNN, Fast R-CNN, SPP-net, Faster R-CNN

From the above graph, you can see that Faster R-CNN is much faster than its predecessors. Therefore, it can even be used for real-time object detection.

5. R-FCN

With the increase in performance, Faster R-CNN was an order of magnitude swifter than its earlier counterpart fast R-CNN .But there was issue of applying the region specific component had to be applied several times in an image, this issue was resolved in R-FCN (Region based Fully Convolutional networks) where the computation required per image was reduced drastically, where instead of cropping features from the same layer where the crops are predicted, the crops are taken from last layer of features prior to predictions. The algorithm works faster than Faster R-CNN while achieving comparable accuracy.

6. SSD

Single Shot Detector achieves a good balance between speed and accuracy. SSD runs a convolutional network on input image only once and calculates a feature map. Now, we run a small 3×3 sized convolutional kernel on this feature map to predict the bounding boxes and classification probability. SSD also uses anchor boxes at various aspect ratio similar to Faster-RCNN and learns the off-set rather than learning the box. In order to handle the scale, SSD predicts bounding boxes after multiple convolutional layers. Since each convolutional layer operates at a different scale, it is able to detect objects of various scales.

That's a lot of algorithms. Which one should you use? Currently, Faster-RCNN is the choice if you are fanatic about the accuracy numbers. However, if you are strapped for computation (probably running it on Nvidia Jetsons), SSD is a better recommendation. Finally, if accuracy is not too much of a concern but you want to go super-fast, YOLO will be the way to go.

Figure 9: Speed comparison of Faster R-CNN, Fast R-CNN, SSD and YOLO

SSD seems to be a good choice as we are able to run it on a video and the accuracy trade-off is very little. However, it may not be that simple, look at this chart that compares the performance of SSD, YOLO, and Faster-RCNN on various sized objects. At large sizes, SSD seems to perform similarly to Faster-RCNN. However, look at the accuracy numbers when the object size is small, the gap widens.

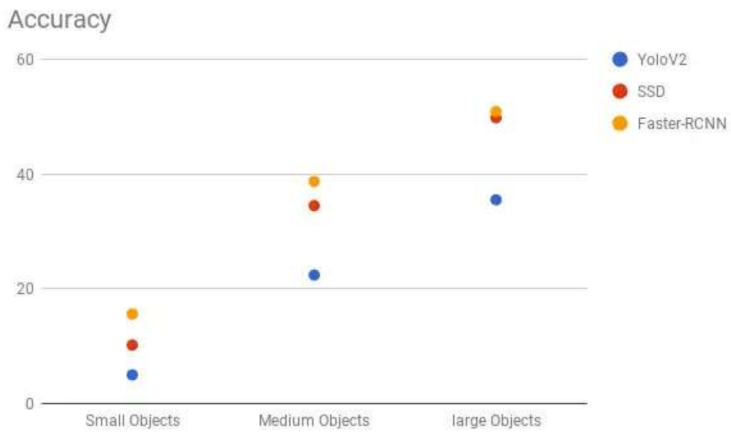

Figure 10: Accuracy comparison of Faster R-CNN, Fast R-CNN, SSD and YOLO

7. YOLO

For YOLO, detection is a simple regression problem which takes an input image and learns the class probabilities and bounding box coordinates.
YOLO divides each image into a grid of S x S and each grid predicts N bounding boxes and confidence. The confidence reflects the accuracy of the bounding box and whether the bounding box actually contains an object(regardless of class). YOLO also predicts the classification score for each box for every class in training. You can combine both the classes to calculate the probability of each class being present in a predicted box.
So, total SxSxN boxes are predicted. However, most of these boxes have low confidence scores and if we set a threshold say 30% confidence, we can remove most of them as shown in the example below.

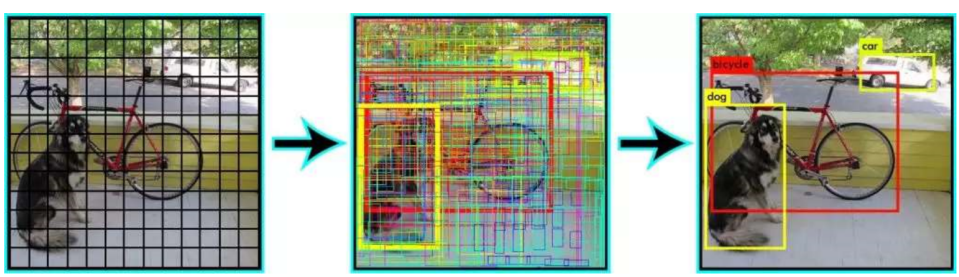

Figure 11: example working of YOLO

Notice that at runtime, we have run our image on CNN only once. Hence, YOLO is super-fast and can be run real time. Another key difference is that YOLO sees the complete image at once as opposed to looking at only a generated region proposal in the previous methods. So, this contextual information helps in avoiding false positives. However, one limitation for YOLO is that it only predicts 1 type of class in one grid hence, it struggles with very small objects.

Illustrating the speed of SSD and YOLO

So in most of the recent projects where real-time object recognition is used where speed is the main concern and accuracy can be compromised, YOLO and SSD are used. To show the speed of these models, I implemented an android application, that detects objects in real-time using the mobile phone camera

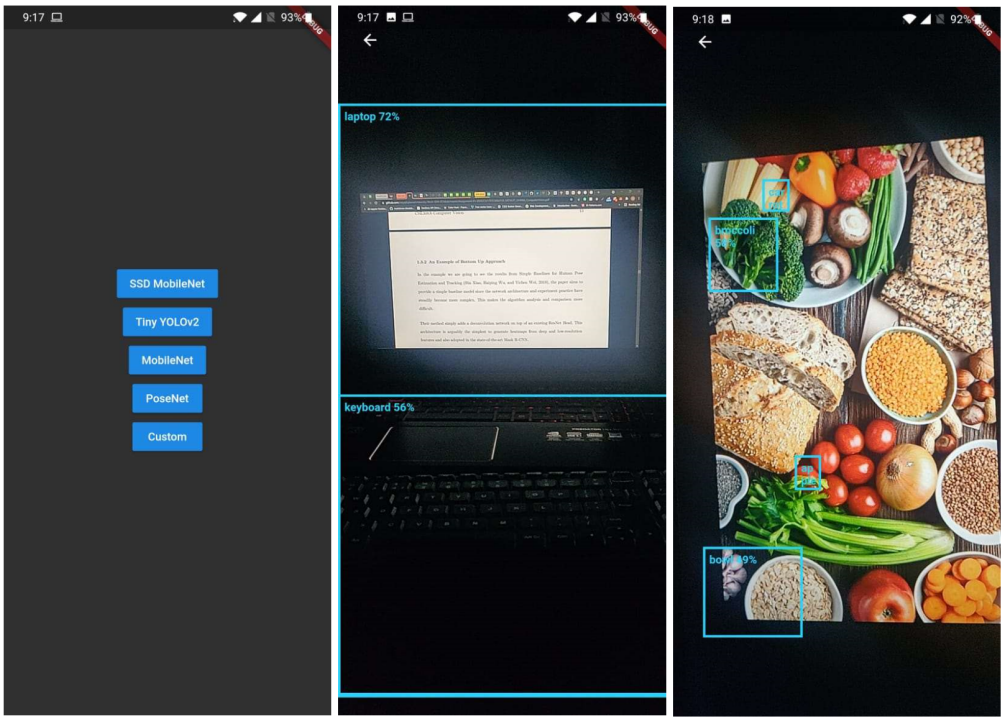

Figure 12: (A) android app for real-time object detection using different models

(B) YOLO for object detection

(C) SSD for object detection

1.4 Conclusion and Recommendations

Choice of a right object detection method is crucial and depends on the problem you are trying to solve and the set-up. Object Detection is the backbone of many practical applications of computer vision such as autonomous cars, security and surveillance, and many industrial applications.

Several object detection techniques like R-CNN, fast RCNN, faster R-CNN, single shot detector (SSD), YOLO v3 etc. are being discussed and compared. From the analysis, it is found that as the model was developed the speed and accuracy has been improved and increased. Fast R-CNN is improved than RCNN but Faster R-CNN is much improved than fast R-CNN. Also, single shot detector is better than faster R-CNN, while YOLO v3 is better than single shot detector. Earlier, till YOLO v3 was not developed SSD was the best. But now, the best technique found latest is YOLO v3 which is much better than SSD also and much faster than SSD. But there is a trade off in accuracy.

1. Baohua Qiang, Ruidong Chen, Mingliang Zhou, Yuanchao Pang, Yijie Zhai and Minghao Yang. Convolutional Neural Networks-Based Object Detection Algorithm by Jointing Semantic Segmentation for Images
2. Shrey Srivastava, Amit Vishvas Divekar, Chandu Anilkumar, Ishika Naik, Ved Kulkarni, Pattabiraman V. Comparative Analysis of Deep Learning Image Detection Algorithms.
3. Dr. Divyakant Meva. A Comparative Study of Various Object Detection Algorithms and Performance Analysis
4. https://arxiv.org/pdf/1504.08083.pdf
5. https://arxiv.org/pdf/1506.02640v5.pdf
6. https://arxiv.org/pdf/1311.2524.pdf
7. https://arxiv.org/pdf/1506.01497.pdf
8. Nikhil Yadav, Utkarsh Binay. Comparative Study of Object Detection Algorithms
9. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e