Comparative Analysis of Deep Learning Image Detection Algorithms

Shrey Srivastava (shrey.srivastava2019a@vitstudent.ac.in)
V.I.T. University, Chennai
Amit Vishvas Divekar
VIT University - Chennai Campus
Chandu Anilkumar
VIT University - Chennai Campus
Ishika Naik
VIT University - Chennai Campus
Ved Kulkarni
VIT University - Chennai Campus
Pattabiraman V.
V.I.T. University - Chennai Campus

Research

Keywords: Object Detection, FRCNN, YOLO-v3, SSD, COCO dataset

Posted Date: December 23rd, 2020

DOI: https://doi.org/10.21203/rs.3.rs-132774/v1

Comparative Analysis of Deep Learning Image Detection Algorithms

Authors: Shrey Srivastava (shrey.srivastava2019a@vitstudent.ac.in), Amit Vishvas Divekar, Chandu Anilkumar , Ishika Naik, Ved Kulkarni, Pattabiraman V.
Institute: Vellore Institute of Technology (Chennai Campus), Kelambakkam - Vandalur Rd, Rajan Nagar, Chennai, Tamil Nadu 600127.

Abstract
As humans, we do not have to strain ourselves when we interpret our surroundings through our visual senses. From the moment we begin to observe, we unconsciously train ourselves with the same set of images. Hence, distinguishing entities is not a difficult task for us. On the contrary, computer views all kinds of visual media as an array of numerical values. Due to this contrast in approach, they require image processing algorithms to examine the contents of images. This project presents a comparative analysis of 3 major image processing algorithms: SSD, Faster R-CNN, and YOLO. In this analysis, we have chosen the COCO dataset. With the help of the COCO dataset, we have evaluated the performance and accuracy of the three algorithms and analysed their strengths and weaknesses. Using the results obtained from our implementations, we determine the differences between how each algorithm runs and suitable applications for each. The parameters for evaluation are accuracy, precision, F1 score.

Keywords: Object Detection, FRCNN, YOLO-v3, SSD, COCO dataset

I. Introduction

In recent times, the industrial revolution makes use of computer vision for their work. Automation industries, robotics, medical field, and surveillance sectors make extensive use of deep learning.[1] Deep learning has become the most talked-about technology owing to its results which are mainly acquired in the realm of image classification, object detection, and language processing. The market forecast predicts outstanding growth around the coming years. The main reasons cited for this are primarily the availability of a large number of datasets and the powerful Graphics Processing Units.[1] In recent times, both these requirements of a large number of datasets and powerful Graphics Processing Units are easily available. [1]

Image classification and detection are the most important pillars of object detection. There is a plethora of datasets available. Microsoft COCO is one such widely used image classification domain. It is a benchmark dataset for object detection. It introduces a large-scale dataset that is available for image detection and classification.[2]

In this review article, we aim to make a comparative analysis of and SSD, Faster-RCNN, and YOLO. The first algorithm that we are comparing in the current work is SSD (Single Shot Detection) which adds layers of several features to the end network and facilitates ease of detection. [3] The Faster R-CNN is a unified, faster, and accurate method of object detection that uses a convolutional neural network. While YOLO (You Only Look Once) was developed by Joseph Redmon that offers end to end network.[3]

II. Literature Survey

Object detection has been an important topic of research in recent times. With powerful learning tools available deeper features can be easily detected and studied. We attempted to compile information on various object detection tools and algorithms used by different researchers so that a comparative analysis can be done and meaningful conclusions can be drawn to apply them in the field of object detection. For this purpose, we carried out literature survey to get an insight regarding our work.

The work done by Ross Girshick has introduced the Fast R-CNN model as a method of object detection.[3] It makes use of the CNN method in the target detection field. The novelty of the method proposed by Girshick has proposed a window extraction algorithm instead of a conventional sliding window extraction procedure in the R-CNN model, there is separate training for the deep convolution network for feature isolation and the support vector machines for categorization.[4] In the fast R- CNN method they have combined feature extraction with

classification into a classification framework. [3] The training time is nine times faster in Fast R-CNN than in R-CNN. Whereas in the faster R-CNN method the proposal isolation region and bit of Fast R-CNN are put into a network template referred to as region proposal network (RPN). The accuracy of Fast R-CNN and Faster R-CNN is the same. The research concludes that the method is a unified, deep learning-based object detection system that works at 5-7 fps. [4]

Another research work done by Kim et al is discussed here. This research work uses CNN with background subtraction to build a framework that detects and recognizes moving objects using CCTV cameras. It is based on the application of the background subtraction algorithm applied to each frame. [5]

Another detection network is YOLO. Joseph Redmon et al have proposed You Only Look Once (YOLO).a one-time convolutional neural network for the prediction of the frame position and classification of multiple candidates is offered by YOLO. end-to-end target detection can be achieved this way. It uses a regression problem to solve object detection. A single end-toend system completes the process of putting the output obtained from the original image to the category and position. [6]

Tanvir Ahmed et al have proposed a modified method that uses an advanced YOLO v1 network model which optimizes the loss of function in YOLO v1, it has a new inception model structure, has a specialized pooling pyramid layer, and has better performance. The advanced application of YOLO is taken from this research paper. It is also an end-to-end process that carries out an extensive experiment on a PASCAL VOC dataset. The network is an improved version and also shows high effectiveness. [7]

Wei Liu et al came up with a new method of detecting objects in images using a single deep neural network. They named this procedure the Single Shot MultiBox Detector SSD. According to the team, SSD is a simple method and requires an object proposal as it is based on the complete elimination of the process that generates a proposal. It also eliminates the subsequent pixel and resampling stages. So, it combines everything into a single step. SSD is also very easy to train and is very straightforward when it comes to integrating it into the system. This makes detection easier. The primary feature of SSD is using multiscale convolutional bounding box outputs that are attached to several feature maps. [8]

Another paper is based on an advanced type of SSD. In his paper, the authors have proposed their research work to introduce Tiny SSD, a single shot detection deep convolutional neural network. TINY SSD aimed to ease real-time embedded object detection. It comprises of highly enhanced layers comprising of non-uniform Fire subnetwork and a stack of non-uniform subnetwork of SSD based auxiliary convolutional feature layers. The best feature of Tiny SSD is its size of 2.3 MB which is even smaller than Tiny YOLO. The results of this work have shown that Tiny SSD is well suited for embedded detections.[9]
We also referred to this review by AR Pathak et al. [1] This paper describes the role of deep learning technique by using CNN for object detection. The paper also accesses some deep learning techniques for object detection systems. The current paper states that deep CNNs work on the principle of weight sharing. It gives us information about some crucial points in CNN. These features of CNN depicted in this paper are: [1]

a. CNN is integration and involves the multiplication of two overlapping functions.
b. Features maps are abstracted to reduce their complexity in terms of space
c. Repetition of the process is done to produce the feature maps using filters.

d. CNN utilizes different types of pooling layers.

III. Existing methodologies

SSD

Other object detection models such as YOLO or Faster R-CNN perform their operations at a much lesser speed as compared to SSD, making a much more favorable object detection method.

Before the development of SSD, several attempts had been made to design a faster detector by modifying each stage of the detection pipeline. However, any significant increase in speed by such modifications only resulted in a decrease in the detection's accuracy and hence researchers concluded that rather than altering an existing model, they would have to come up with a fundamentally different object detection model, and hence, the creation of the SSD model. [8]

SSD does not resample pixels or features for bounding box hypotheses and is as accurate as models that do. In addition to this, it is quite straightforward compared to methods that require object proposals because it completely removes proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD simpler to train and straightforward to integrate into systems that require a detection component. [8]

It's architecture heavily depends on the generation of bounding boxes and the extraction of feature maps, which are also known as default bounding boxes. The network calculates loss by comparing the offsets of the predicted classes and the default bounding boxes with the ground truth values of the training samples, using different filters for every iteration. Using the backpropagation algorithm and the calculated loss value, all the parameters are updated. This way, SSD is able to learn the most optimal filter structures that can accurately identify the object features and generalize the given training samples in order to reduce the loss value, resulting in high accuracy during the evaluation phase. [10]

Analysis of the Functions

SSD is based on a feed-forward complex network that builds a fixed-size collection of bounding boxes and scores corresponding to the presence of object instances in those boxes, followed by non-maximum suppression to generate the final detection results. The preliminary network layers are based on a standard architecture utilized for high quality image classification (and truncated before any classification layers), which is a VGG-16 network. An auxiliary structure is added to the truncated base network such as convo6 to produce detections.

1. **Extracting Feature Maps:**
   SSD uses the VGG-16 architecture to extract feature maps because it shows very good performance for the classification of images with high quality. The reason for using auxiliary layers is because they allow us to extract the required features at multiple scales as well as reduce the size of our input with each layer that we traverse through. [8] For each cell in the image, the layer makes a certain number of predications. Each prediction consists of a boundary box and the box generates scores for all the classes it detects in this box including a score for no object at all. We can think of it as an algorithm making a 'guess' as to what is in the boundary box by choosing the class with the highest score. These scores a called 'confidence scores' and making such predictions is called 'MultiBox'. Figure 1 depicts the SSD model with the extra feature layers.

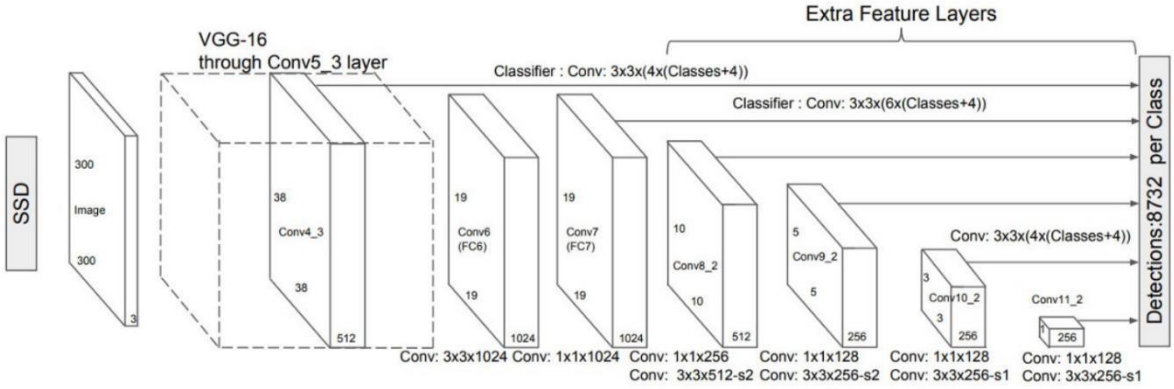


Figure1: SSD model [8]

2. **Convolutional predictors for object detection:**

   Every feature layer produces a fixed number of predictions by utilising convolutional filters. For every feature layer of size m × n having p channels, the basic component for generating prediction variables of a potential detection result is a 3 × 3 × p small kernel that creates a confidence score for every class, or a shape offset calculated with respect to the default grounding box coordinates which are provided by the COCO Dataset at every single one of the 'm x n' locations. [8]

3. **Default boxes and aspect ratios:**

   By now, you may be able to infer that each feature map cell is associated with a corresponding default bounding box for multiple feature maps in the network. The default boxes decide the feature map in a complex manner so that the placement of each box concerning its corresponding cell is fixed. At each feature map cell, we speculate the offsets concerning the default box shapes in the cell and the scores for each class which tells us about the class of object present inside the bounding box. Going into further detail, for every box out of k at a particular given location, c class scores are calculated and its 4 offsets relative to the primal default box shape. This computation results in a total of (c + 4) k filters that are applied around each location in the feature map, yielding (c + 4) kmn outputs for a m × n feature map. [8]

<u>SSD Training Process</u>

1. **Matching Process:**

   All SSD predictions are divided into two types; positive matches or negative matches. Positive matches are only used by SSD to calculate the localization cost which is the mismatch of the boundary box with the default box. The match is positive only if the corresponding default boundary box has an IoU greater than 0.5 with the ground truth. In any other case, it is negative. IoU stands for the 'intersection over the union'. It is the ratio between the intersected area over the joined area for two regions. IoU is also referred to as the Jaccard index and using this condition makes the learning process much easier.

2. **Hard negative mining:**

   After the matching step, almost all of the default boxes are negatives, largely when the count of possible default boxes is high. This causes a large imbalance between the positive and negative training examples. Rather than using up all the negative examples, SSD sorts them by their greatest confidence loss for each default box, the highest ones

so that the ratio of the negatives and positives is at most 3:1 at any given moment were picked. This leads to faster optimization and more stable training. [8]

3. **Data augmentation:**
   This is crucial for increasing accuracy. Data can be augmented with flipping, cropping, and color distortion. To handle variants in various object sizes and shapes, each training image is randomly sampled by one of the following options: [8]

   → Use the original,

   → Sample a patch with IoU of 0.1, 0.3, 0.5, 0.7 or 0.9, →
   Randomly sample a patch.

4. **Final detection**: The results are generated by performing NMS on multi-scale refined bounding boxes. Using the above-mentioned methods such as hard negative mining, data augmentation, and a larger number of other methods, SSD significantly outperforms the Faster R-CNN in terms of accuracy on PASCAL VOC and COCO, while being three times faster.[10] The SSD300 ,where the input image size is 300_300, runs at 59 FPS, which is more accurate and efficient than YOLO. However, SSD is not as efficient at detection for small objects, which can be solved by having a better feature extractor backbone (e.g. ResNet101), adding deconvolution layers with skip connections to introduce additional large-scale context, and designing a better network structure. [11]

ALGORITHM: [8]

1. We began by importing all relevant modules into our google colab notebook such as os, pathlib, matplotlib, matplotlib.pyplot, io, scipy.misc, numpy ,PIL , TensorFlow and tensorflow_hub.
2. 'tf.get_logger(). setLevel('ERROR')' prevents TensorFlow from logging any unnecessary debugging information on the terminal
3. Next, we defined a function called as 'load_image_into_numpy_array' that takes a test image url as an argument. If the url is valid, the image is converted to a numpy array and returned.
4. The array is reshaped to a three-channel image input tensor of type 'tf.uint8' with shape [1, height, width, 3] since our model will only accept this as input.
5. To properly visualize the images with detected boxes, key points, and segmentation, the TensorFlow Object Detection API was used. To install it, the repository 'https://github.com/tensorflow/models' was cloned.
6. After installing the Object detection API and importing the remaining dependencies, the next step was done.
7. Label maps correspond index numbers to the category names so that when our complex network predicts a certain number, we know what label it corresponds to. Here we used internal utility functions, where we loaded the 'mscoco_label_map.txt' file from the repository that we loaded the Object Detection API code from before.

   Using 'label_map_util.create_category_index_from_labelmap', we used the path to this text file to create a list called 'category index'.
8. For our project, we used the 'SSD MobileNet v2 320x320' object detection model. The SSD with Mobilenet v2 is initialized from the Imagenet classification checkpoint and

trained on COCO 2017 dataset (images scaled to 320x320 resolution). The model is load using 'hub.load(model)'

9. Since this model is pretrained on the COCO dataset, we directly began testing it with the images that we converted to tensors earlier on in the code.
10. Calling the 'load_image_into_numpy_array' function with the test image's url path as an argument, the returned numpy array is passed through the hub_model() function which returns a dictionary of properties of the image:
11. To visualize the results, we used the TensorFlow Object Detection API to show the squares from the inference step (and the key points when available).
    If the dictionary returns keypoints, we include them in the result. The detection boxes,classes,their respective scores,the maximum number of boxes, the minimum score threshold are assigned in the
    'viz_utils.visualize_boxes_and_labels_on_image_array( )' function
12. Finally, these results are visualized by displaying the image with all its detection boxes, their labels, and respective scores.

Faster R-CNN

R-CNN stands for Region-based Convolutional Neural Networks. This method combines region proposals for object segmentation and high capacity CNNs for object detection. [12]

The algorithm of the original R-CNN technique is as follows: [12]

1. Using a Selective Search Algorithm, several candidate region proposals are extracted from the input image. In this algorithm, numerous candidate regions are generated in initial sub-segmentation. Then, regions which are similar are combined to form bigger regions using a greedy algorithm. These regions make up the final region proposals.
2. The CNN component warps the proposals and extracts distinct features as a vector output.
3. The features which are extracted are fed into an SVM for recognizing objects of interest in the proposal.

Figure 2 given below explains the features and working of R-CNN.

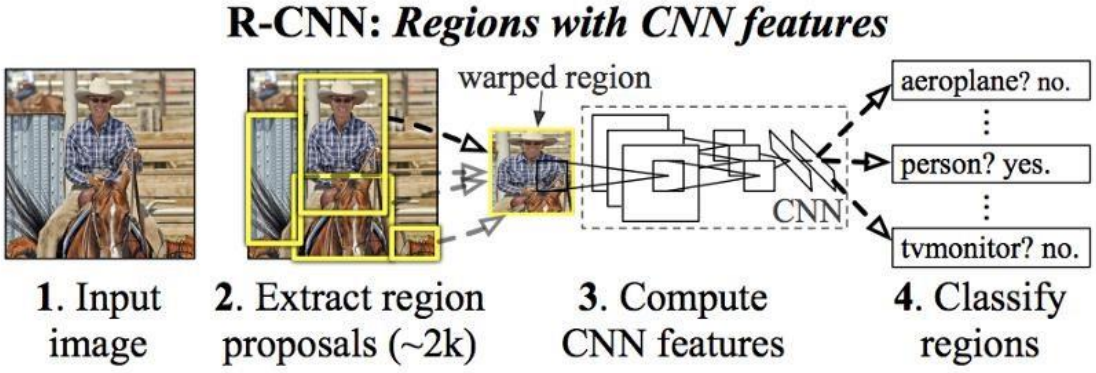


Figure 2: R-CNN model [12]

This technique was plagued by a lot of drawbacks. The requirement to classify ~2000 region proposals makes the training of the CNN a very time-consuming process. This makes real-time implementation impossible as each test image would take close to 47 seconds for execution.

Furthermore, machine learning could not take place as the Selective Search Algorithm is a fixed algorithm. This could result in non-ideal candidate region proposals being generated. [12]

Fast R-CNN is an algorithm for object detection that solves some of the drawbacks of R-CNN. It uses an approach like R-CNN, but instead of region proposals, the CNN uses the image itself for generating a convolutional feature map, from which region proposals are identified and warped. An RoI pooling layer is used to reshape the warped squares into a predefined size for a fully connected layer to accept them. A SoftMax layer then predicts the class of the region from the RoI vector. [13]

Fast R-CNN is faster than its predecessor because feeding ~2000 proposals as input to the CNN per execution is not required. The convolution operation is done to generate a feature map only once per image. [13] The figure 3 given below describes the features and working of Fast RCNN.

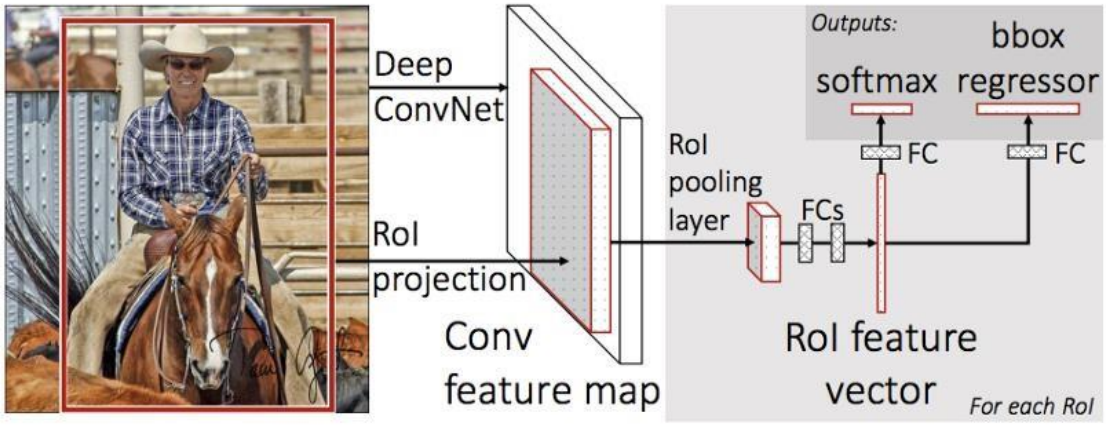


Figure 3: Fast R-CNN [13]

Fast R-CNN shows a significant reduction in training and testing time required over R-CNN. But it was noticed that including region proposals significantly bottlenecks the algorithm, reducing its performance. [3]

R-CNN and Fast R-CNN used Selective Search as the algorithm to figure out region proposals. This being a very time-consuming algorithm, Faster R-CNN eliminated the need for its implementation and instead let the proposals be learned by the network. Just as in the case of Fast R-CNN, a convolutional map is obtained from the image. But a separate network replaces the Selective Search algorithm to predict proposals. These proposals are then reshaped and classified using RoI pooling. Refer to the figure 4 for the working of Faster R-CNN.

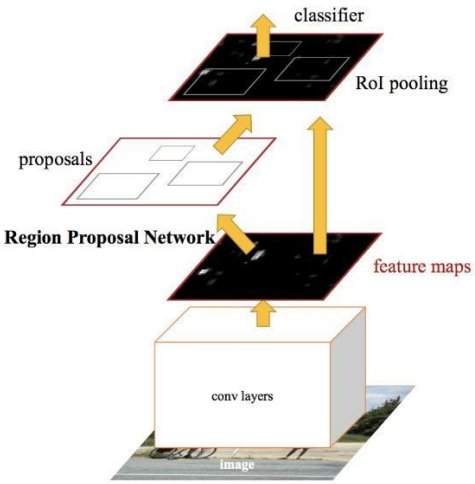


Figure 4: Faster R-CNN [3]

Faster R-CNN offers an improvement over its predecessors so significant that it is now capable of being implemented for real-time object detection.

YOLO

In modern times YOLO (You Only Look Once) is one of the most precise and accurate object detection algorithms available. It has been made on the basis of a newly altered and customized architecture named Darknet [14]. The first version was inspired by Google Net, which used tensor to sample down the image and predicted it with the maximum accuracy. The tensor is generated on the basis of a similar procedure and structure which is also seen in the Region of Interest that is pooled and compiled {to decrease the number of individual computations and make the analysis swifter) that is used in the Faster R-CNN network. The following generation utilized an architecture with just 30 convolutional layers, that in turn consisted of 19 layers from DarkNet-19 and an extra 11 for detection of natural objects or objects in natural context as the way we have used by having COCO as dataset and metrics. It provided more precise detection and with good speed, although it struggled with pictures of small objects and small pixels. But version 3 has been the greatest and most accurate version of YOLO which has been used widely because of its high precision. Also, the architecture with multiple layers has made the detection more precise. [10]

YOLOv3 makes use of the latest darknet features like 53 layers and it has undergone training with one of the most reliable datasets called ImageNet. The layers used are from an architecture Darnnet-53 which is convolutional in nature. For detection, the aforementioned 53 layers were supplemented instead of the pre-existing 19 and this enhanced architecture was trained and instructed with PASCAL VOC. After so many additional layers the architecture maintains one of the best response times with the accuracy offered. It also is very helpful in analyzing live video feed because of its swift data unsampling and object detection techniques. One can notice that this version is the best enhancements in ML using neural networks. The previous version did not work well with the images of small pixels but the recent updates in v3 have made it very useful in analyzing satellite imaging even for defense departments of some countries. The architecture performs in 3 different layers which makes it more efficient but the process is a little slower yet state-of-the-art. For understanding, the framework refers to the figure 5 given below.

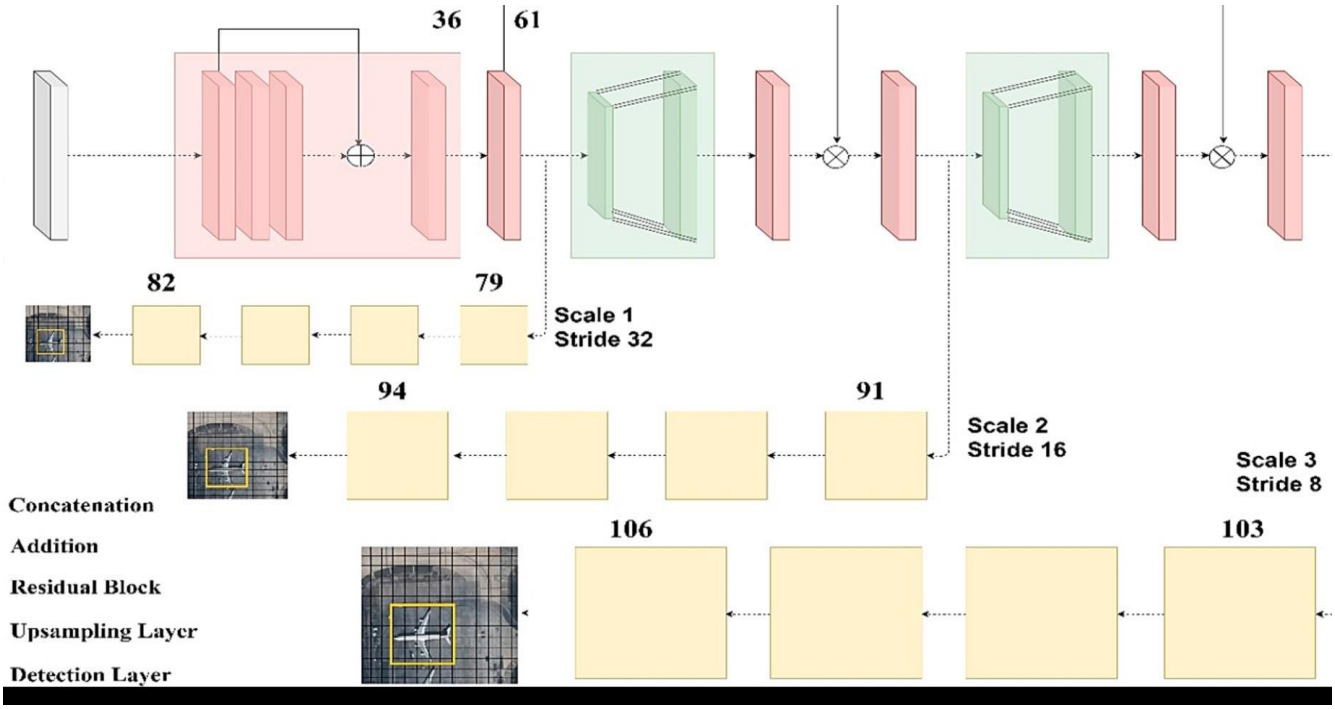


Figure 5: YOLO Architecture [10] **Analysis**

Of The Functioning [15]:

1. Forecasting:

❾ This model utilizes packages of different lengths and breadths to produce the weights and frames that establish a strong foundation. This technique is an individual where the network determines the objectivity and allocation independently. The logical regression is used by YOLOv3 where it foresees the objectivity score. It is projected over the selection frame initially on the object that has been established to be the fundamental truth in the picture by pre-training models.[16] This gives a singular bounding box and any kind of fallacy in this part would cause mistakes in both allocation of these boxes and their accuracy and also in the detection arrear. The bounding box forecasting is depicted in the equation given in figure 6.

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
\sigma(x) &= 1/(1+e^{-x})
\end{aligned}
$$

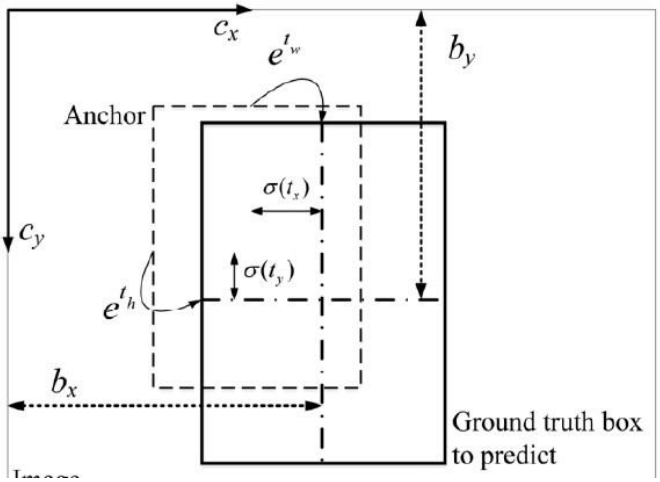


Figure 6: Bounding box forecasting [15]

2. Class Prediction:

YOLOv3 executes a soft-max function to alter the scores to an understandable format for the code. The format are 1. YOLOv3 uses multiple classifications by tag. These tags are custom and non-exclusive. For eg. 'man' and 'woman' are not exclusive. The architecture modifies the function with individualistic logistic classifiers. YOLOv3 uses binary loss function initially. It uses the soft-max function after that. This leads to a reduction in complexity by avoiding it for the first implementation. [17]

3. Predictions:

3 distinct orders and dimensions are used for pre-determining the bounding boxes. These are in combination with the function extractor, DarkNet-53. The last levels include detection and categorization into object classes. 3 takes is what is taken on each scale of the COCO dataset. That leads to more than 70 class predictions as an o/p tensor. These features are a classic coder-decoder design introduced in Single-Shot-Detector. The grouping of k-means is also used for finding the best bounding boxes. Finally, in the COCO dataset dimensions like 10 x 13, 62 x 45 and others are used. In total there are 9 distinct dimensions including the aforementioned.

4. DarkNet-53 - The feature Extractor:

YOLOv2 had the implementation of DarkNet-19 but in the recently modified model of YOLO Darknet-53 is being used where the 53 is 53 convolutional levels. DarkNet-53 offers better performance and is 1.5 times faster. Compared to ResNet-152, DarkNet-53 has almost the same performance in terms of accuracy and precision but it is twice as fast. [18] The following figure 7 shows the YOLO model.

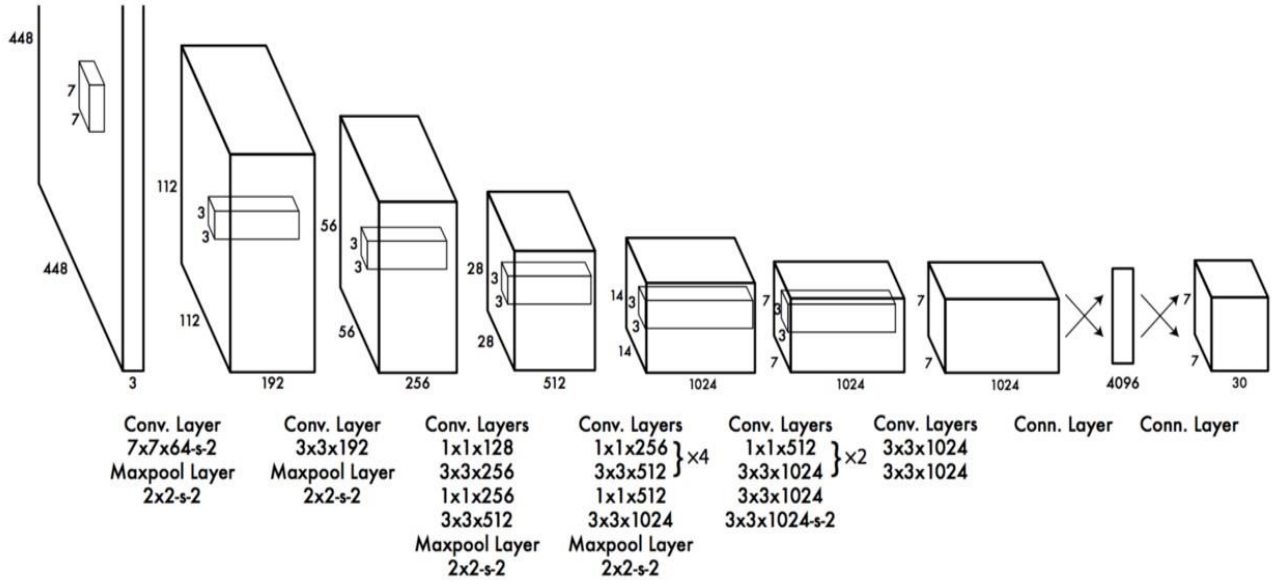


Figure 7: YOLO Model ConvNet [18]

IV. Dataset:

MICROSOFT COCO

In recent times for the search of a perfect combination of algorithm and data set, contenders have used the top and highly rated deep learning architectures and data sets. They are used for arriving at the best possible precision and accuracy. The most commonly used data sets are PASCAL VOC and Microsoft COCO. For our review analysis, we have used COCO as a dataset as well as an Evaluation Metric. They applied different ways of analysis, tweaking and calibrating the base networks and adjusting the software; that leads to better precision but also for improving accuracy, speed, and local split performance. [10]

For Object detection, the use of computationally costly architectures and algorithms such as RCNN, SPP-NET the use of smart data sets having varied objects and images which also have various objects and are of different dimensions have become a necessity. Not to forget the extreme scope in live video feed monitoring the cost of detection becomes too high. Recently the advancement in deep learning architectures has lead algorithms like YOLO and SSD networks to detect objects by the access to a singular NN (neural network). The introduction of latest architectures has increased the competition between various techniques. [10] But recently COCO has emerged as the most used data set for training and classification. Also, more developments have made it alterable for adding classes. [2].

Furthermore, COCO is better than other popular widely used data sets as per some research papers. [2] They are namely Pattern Analysis, Statistical Modelling and Computational Learning Visual Object Classes, ImageNet & SUN. The above-mentioned data sets vary hugely based on size, categories, and types. ImageNet was made to target a wider category where the number of different categories but they were fine-grained. SUN focused on more of a modular approach where the regions of interest were based on the frequency of them occurring in the data set. Finally, PASCAL VOC's was similar yet different in approach to COCO. It used a wide range of images taken from the environment and nature. Microsoft Common Objects in

Context is made for the detection and classification of the objects in their classic natural context. [2].

Annotation pipeline [2]

As seen in the following figure 8 an annotation pipeline explains the identification and categorization of a particular image.

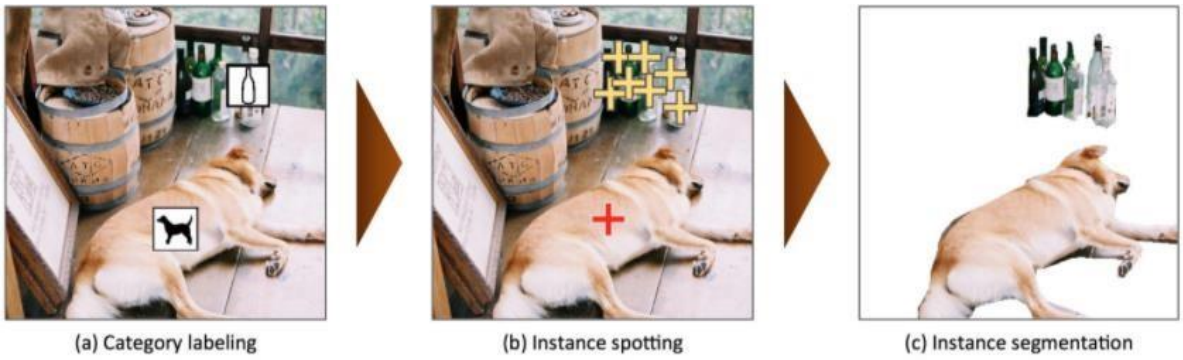


Figure 8: Annotation pipeline [2]

This type of annotation pipeline gives a better perspective to object detection algorithms. Training algorithms using these diverse images and advanced concepts like crowd scheduling and visual segmentation. Following figure 9 gives the detailed categories that are available in MS COCO. The 11 super-categories are Person and Accessories, Animal, Vehicle, Outdoor Objects, Sports, Kitchenware, Food, Furniture, Appliance, Electronics, and Indoor Objects. [19]

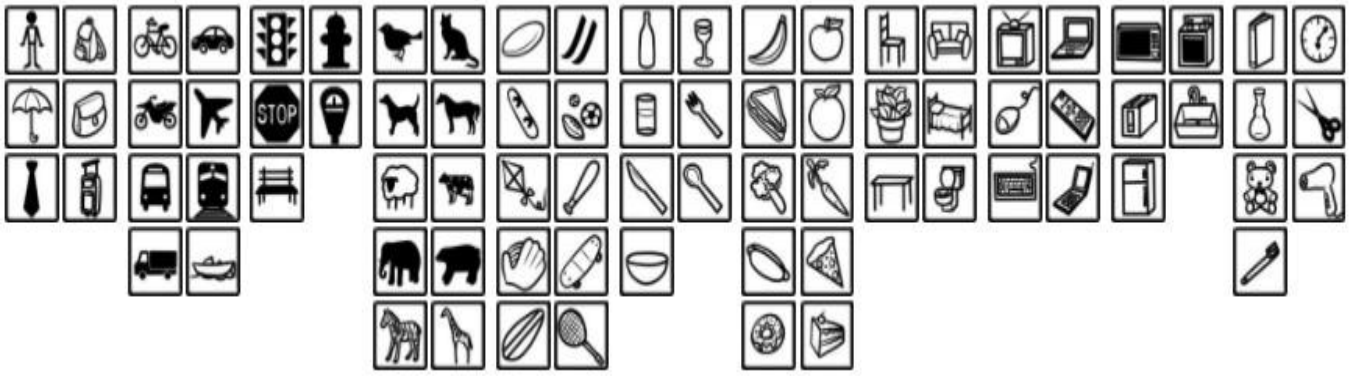


Figure 9: Categories of images [19]

V. RESULTS AND DISCUSSIONS:

Two performance metrics are applied to object detecting models for testing. These are 'Average Precision' and an F1 score. The predicted bounding boxes are compared with the ground truth bounding boxes by the detector according to IOU (Intersection Over Union). The 'True Positive', 'False Negative', and 'False Positive' are defined and then used for the calculation of precision and recall which in turn are used for calculating the F1 score. The Formulae for these are as follows:

Precision = TP/(TP +FP')

Recall = TP/(TP + FN')

And using these, F1 score = 2*Precision*Recall/(Precision + Recall)

Apart from these two, the performance of the models is also measured using the metrics given by the COCO metrics API. Using all these, the outcomes came out to be as follows:

Figure 8: Graph for SSD [10]

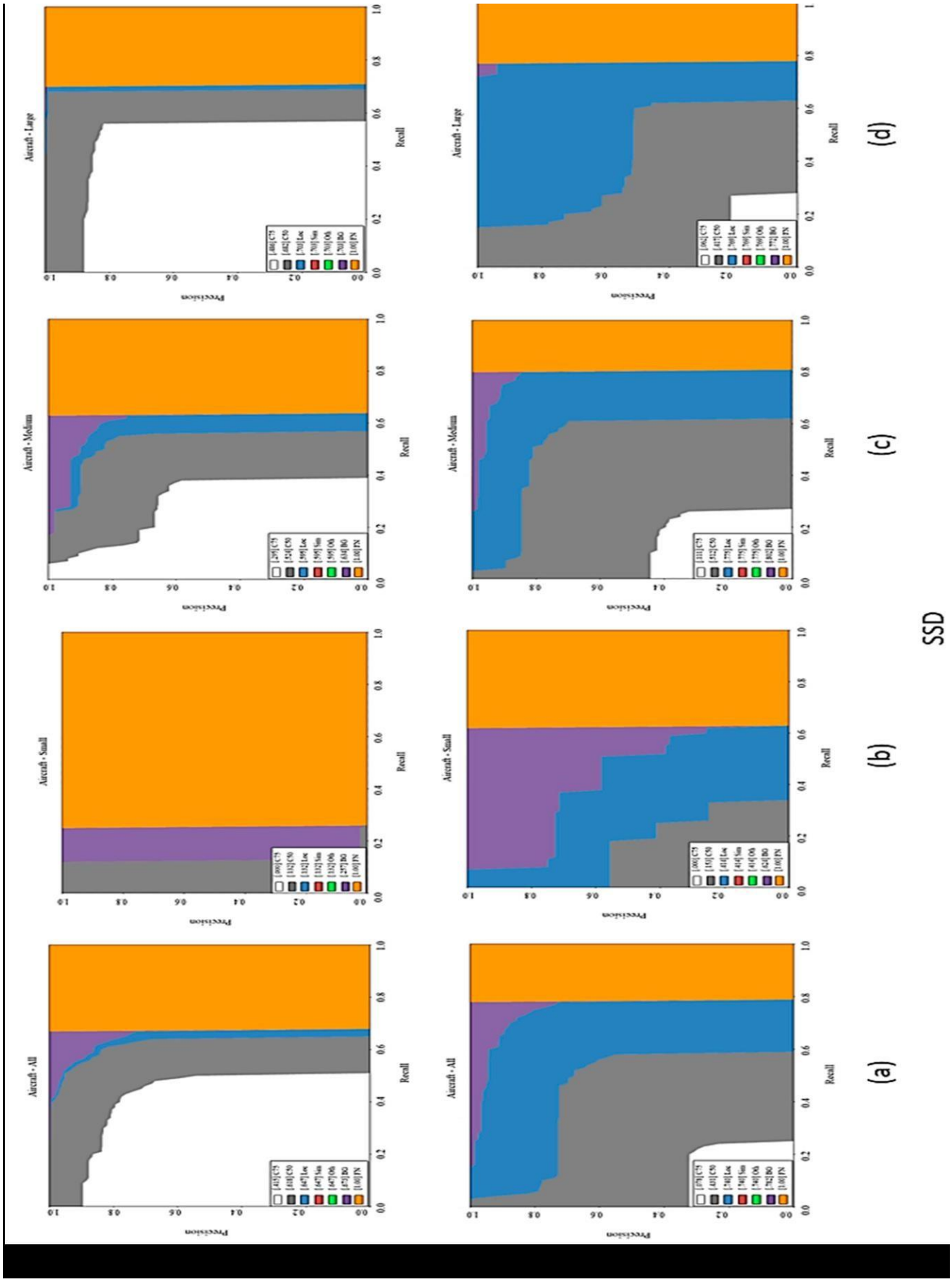


Figure 9: Graph for Faster RCNN [10]

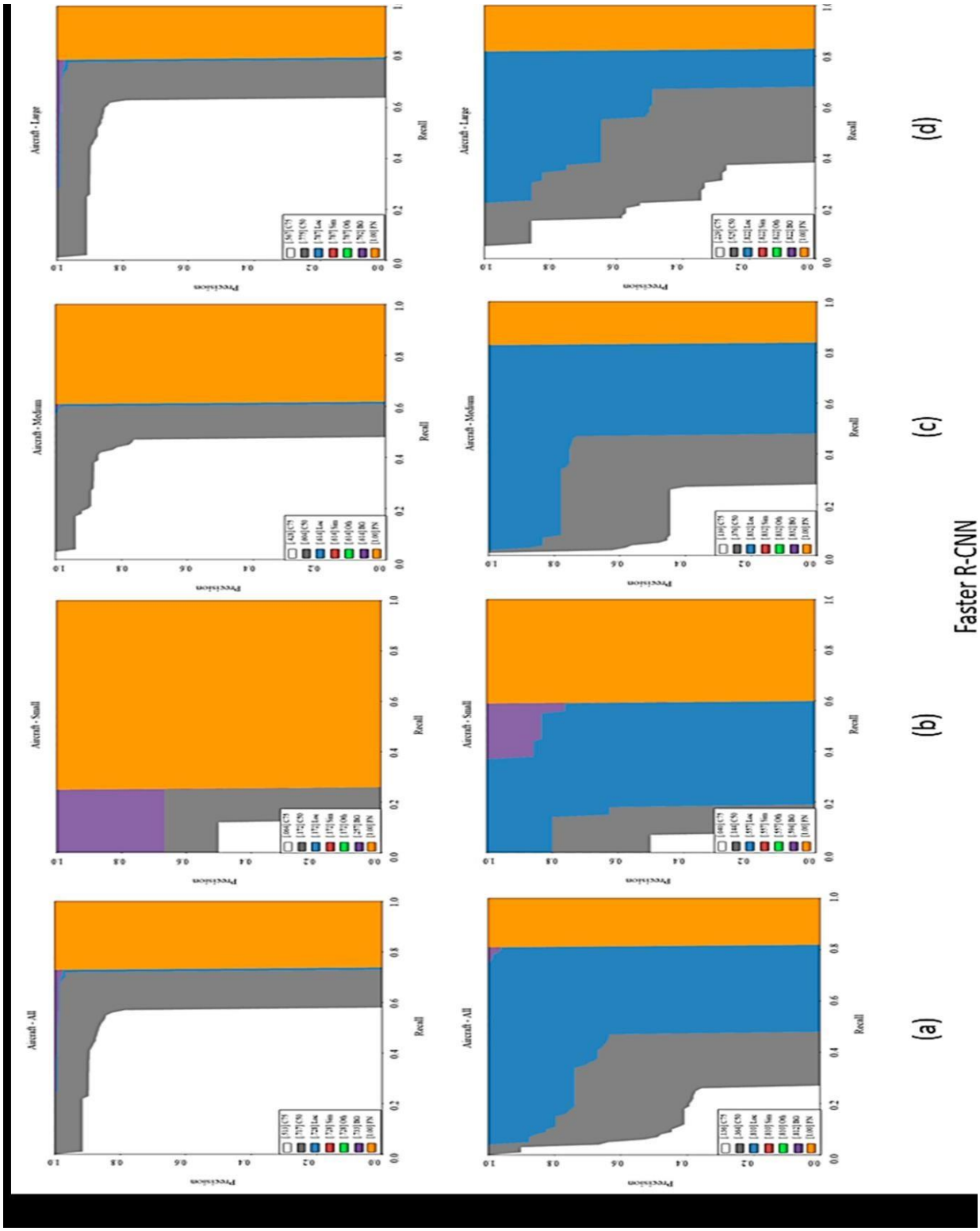
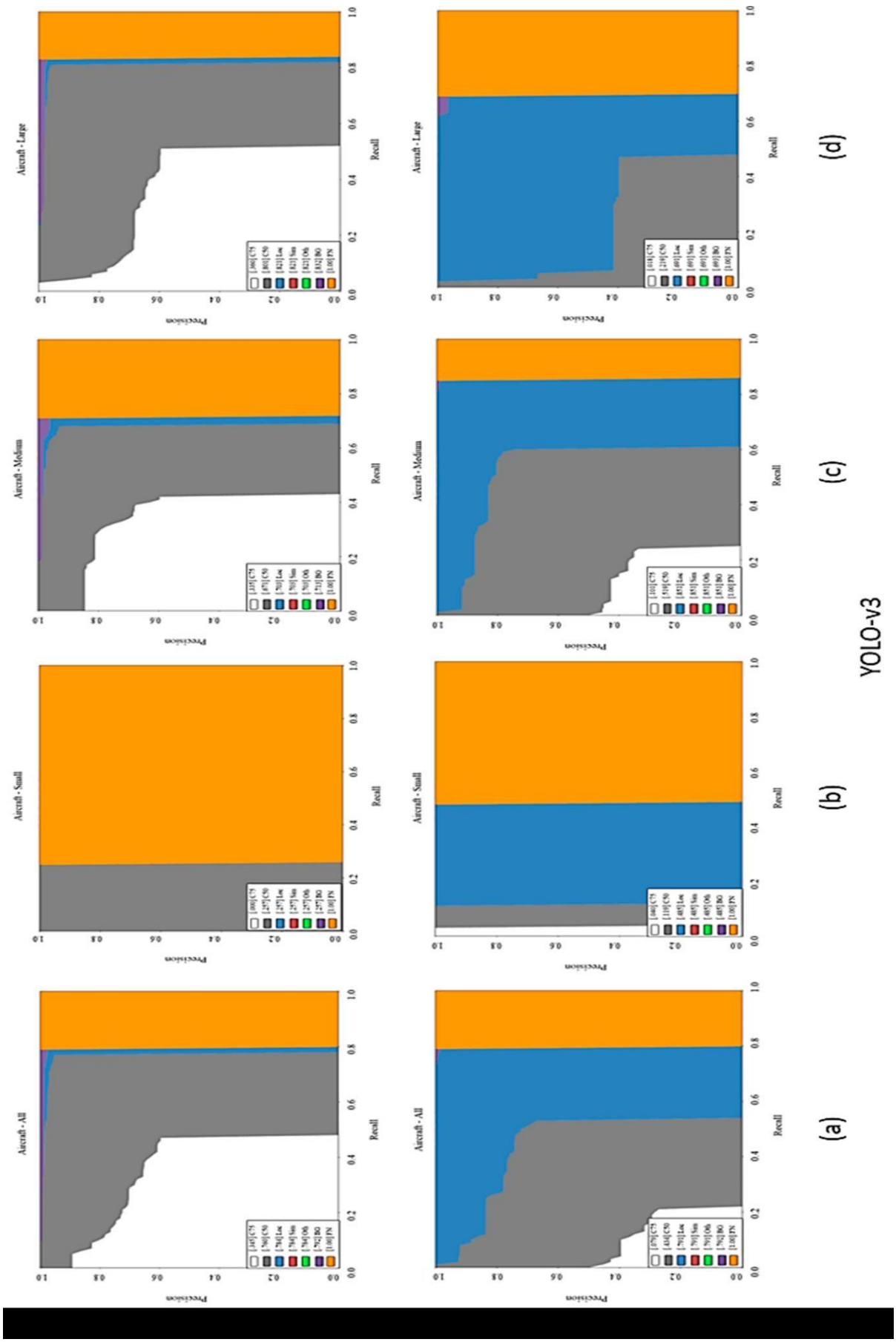


Figure 10 : Graph for YOLO [10]


Aircraft - All
Aircraft - Small
Aircraft - Medium
Aircraft - Large
Precision
Recall
(a)
(b)
(c)
(d)
YOLO-v3

Results Comparison:

Nº	IoU	area	maxDets	Average Precision			Average Recall		
				SSD	YOLO	FRCNN	SSD	YOLO	FRCNN
1	0.50:0.95	all	100	0.247	0.337	0.716	0.232	0.279	0.782
2	0.50	all	100	0.424	0.568	0.873	0.341	0.432	0.754
3	0.75	all	100	0.253	0.350	0.851	0.362	0.460	0.792
4	0.50:0.95	small	100/1*	0.059	0.152	0.331	0.102	0.257	0.567
5	0.50:0.95	med	100	0.264	0.359	0.586	0.401	0.494	0.653
6	0.50:0.95	large	100	0.414	0.496	0.846	0.577	0.623	0.893

*on Nº 4, maxDets value is 100 for avg. precision and 1 for avg. recall

The result of all these tests shows that YOLO is the better algorithm for building an Object detection model for overall accuracy while Faster R-CNN is the fastest and SSD was the best when it came to localization of objects.

VI. Conclusion

This review article compared the latest and most advanced CNN-based object detection algorithms. Without object detection, it would be impossible to analyze the hundreds of thousands of images that are uploaded to the internet every day.[19] Technologies like selfdriving vehicles that depend on real-time analysis are also impossible to realize without object detection. All the networks were trained with the open-source COCO dataset by Microsoft, to ensure a homogeneous baseline. It was found that Yolo-v3 is the fastest with SSD following closely and Faster RCNN coming in the last place. However, in terms of accuracy, when speed is not a major concern, Faster RCNN gives the most accurate results. SSD provides good object localization, even though it fares the worst in overall detection. Yolo-v3 displays fast convergence capability. Both Yolo-v3 and SSD find it difficult to detect small objects, something Faster RCNN has no problem doing. Thus, we can say that the use case influences which algorithm is picked; if you are dealing with a relatively small dataset and don't need real-time results, it is best to go with Faster RCNN. Yolo-v3 is the one to pick if you need to analyze a live video feed. Meanwhile, SSD provides a good balance between speed and accuracy. Additionally, Yolo-v3 is the most recently released of the three and is actively being contributed to by the vast open-source community. Hence, in conclusion, out of the three Object Detection Convolutional Neural Networks we analyzed, Yolo-v3 shows the best overall performance. This result is similar to what some of the previous reports have obtained.

A great deal of work can still be done in the future in this field. Every year, either new algorithms or updates to existing ones are published. Also, each field – aviation, autonomous vehicles (aerial and terrestrial), industrial machinery, etc. are suited to different algorithms. These subjects can be explored in detail in the future.

VII. List of Abbreviations

a. FRCNN – Faster Region based Convolutional Neural Network
b. SSD – Single Shot Detector
c. YOLO-v3 – You Look Only Once version 3
d. COCO – Common Objects in Context
e. VGG16 – Visual Geometry Group 16

VIII. Declarations

a. Ethics approval and consent to participate

Not applicable

b. Consent for publication

Not applicable

c. Availability of data and materials

Coco dataset used in the paper is available from the website https://cocodataset.org/#explore

d. Competing interests

The authors declare that they have no competing interests

e. Funding

Not applicable

f. Author's information

i. Amit Vishvas Divekar
Registration Number: 19BAI1002
School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600 127

ii. Chandu Anilkumar
Registration Number: 19BAI1003
School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600 127

iii. Shrey Srivastava
Registration Number: 19BAI1013
School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600 127

iv. Ishika Naik
Registration Number: 19BAI1079
School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600 127

v. Ved Kulkarni
Registration Number: 19BAI1092
School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600 127

vi. Pattabiraman V.
Employee ID: 50168
School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600

g. Affiliation

School of Computer Science and Engineering [SCOPE], Vellore Institute of Technology [VIT] Chennai Campus, Chennai, Tamil Nadu, India-600 127

h. Acknowledgment

Not Applicable

i. Contributions

i. Shrey Srivastava

- Research and Implementation of YOLO Algorithm.
- Comparative Analysis.

ii. Amit Vishvas Divekar

- Research and Implementation of Faster RCNN Algorithm.
- Comparative Analysis.

iii. Chandu Anilkumar

- Research and Implementation on Faster RCNN Algorithm.
- Comparative Analysis.

iv. Ishika Naik

- Research and Implementation of SSD Algorithm.
- Comparative Analysis.

v. Ved Kulkarni

- Research and Implementation on SSD Algorithm.
- Comparative Analysis

vi. Pattabiraman V.

- Verification of results obtained through implementations. • Approval of final manuscript.

IX. References

1. Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of deep learning for object detection. Procedia computer science, 132, 1706-1717.

2. Palop, J. J., Mucke, L., & Roberson, E. D. (2010). Quantifying biomarkers of cognitive dysfunction and neuronal network hyperexcitability in mouse models of Alzheimer's disease: depletion of calcium-dependent proteins and inhibitory hippocampal remodeling. In Alzheimer's Disease and Frontotemporal Dementia (pp. 245-262). Humana Press, Totowa, NJ.

3. Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137-1149.

4. Ding, S., & Zhao, K. (2018, March). Research on daily objects detection based on deep neural network. In IOP Conference Series: Materials Science and Engineering (Vol. 322, No. 6, p. 062024).

5. Kim, C., Lee, J., Han, T., & Kim, Y. M. (2018). A hybrid framework combining background subtraction and deep neural networks for rapid person detection. Journal of Big Data, 5(1), 22.

6. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

7. Ahmad, T., Ma, Y., Yahya, M., Ahmad, B., & Nazir, S. (2020). Object Detection through Modified YOLO Neural Network. Scientific Programming, 2020.

8. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

9. Womg, A., Shafiee, M. J., Li, F., & Chwyl, B. (2018, May). Tiny SSD: A tiny singleshot detection deep convolutional neural network for real-time embedded object detection. In 2018 15th Conference on Computer and Robot Vision (CRV) (pp. 95101). IEEE.

10. Alganci, U., Soydas, M., & Sertel, E. (2020). Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. Remote Sensing, 12(3), 458.

11. Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11), 32123232.

12. Shen, X., & Wu, Y. (2012, June). A unified approach to salient object detection via low rank matrix recovery. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 853-860). IEEE.

13. Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

14. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

15. Xu, D., & Wu, Y. (2020). Improved YOLO-V3 with DenseNet for Multi-Scale Remote Sensing Target Detection. Sensors, 20(15), 4276.

16. Butt, U. A., Mehmood, M., Shah, S. B. H., Amin, R., Shaukat, M. W., Raza, S. M., ... & Piran, M. (2020). A Review of Machine Learning Algorithms for Cloud Computing Security. Electronics, 9(9), 1379.

17. Ketkar, N., & Santana, E. (2017). Deep Learning with Python (Vol. 1). Berkeley, CA: Apress.

18. Jiang, R., Lin, Q., & Qu, S. (2016). Let blind people see: real-time visual recognition with results converted to 3D audio. Report No. 218, Standord University, Stanford, USA.

19. COCO. [Internet]. Available at https://cocodataset.org/#explore. Accessed on Oct 28 2020.

Figures

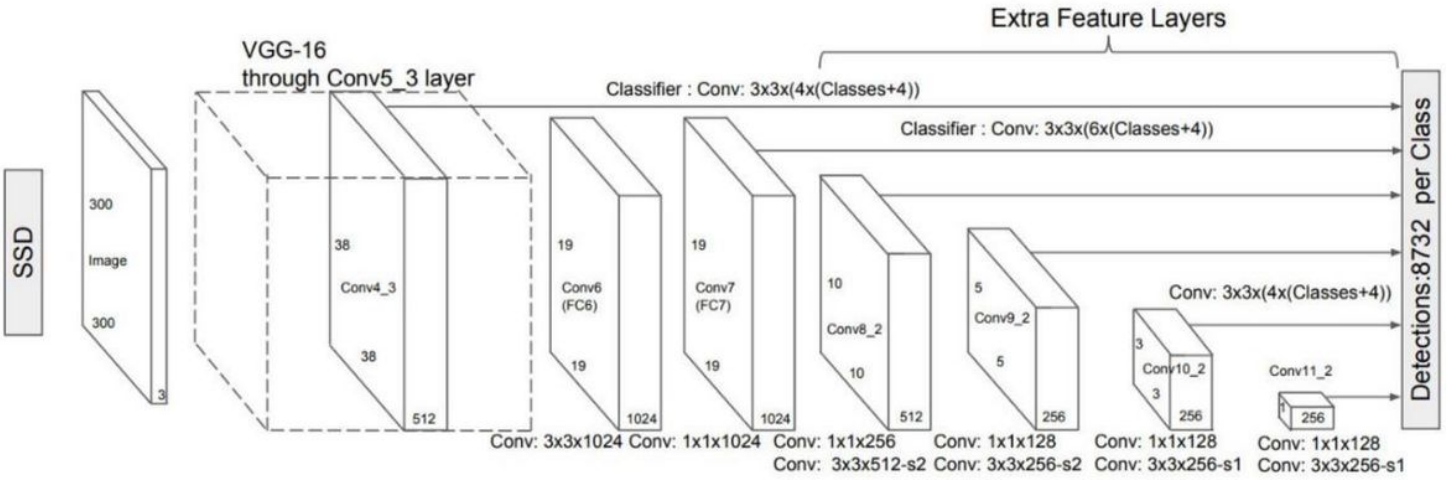


Figure 1

SSD model [8]

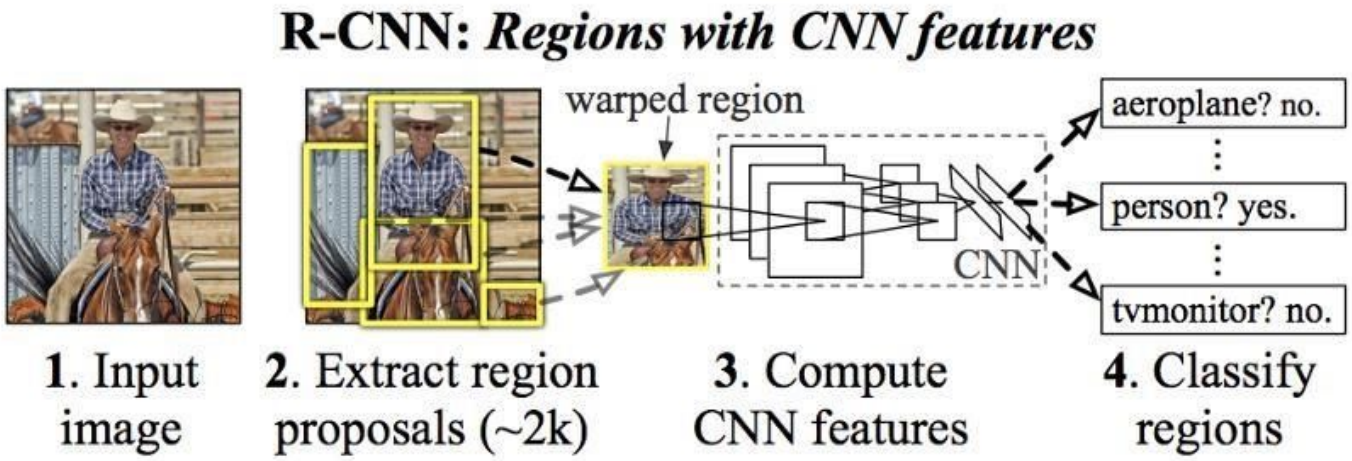


Figure 2

R-CNN model [12]

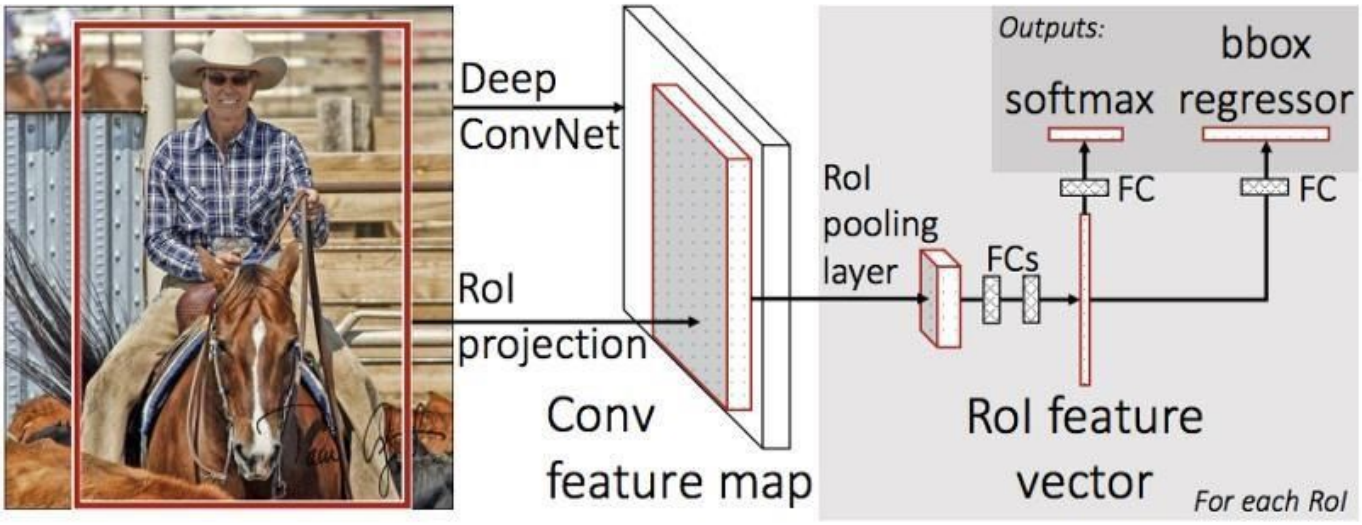


Figure 3

Fast R-CNN [13]

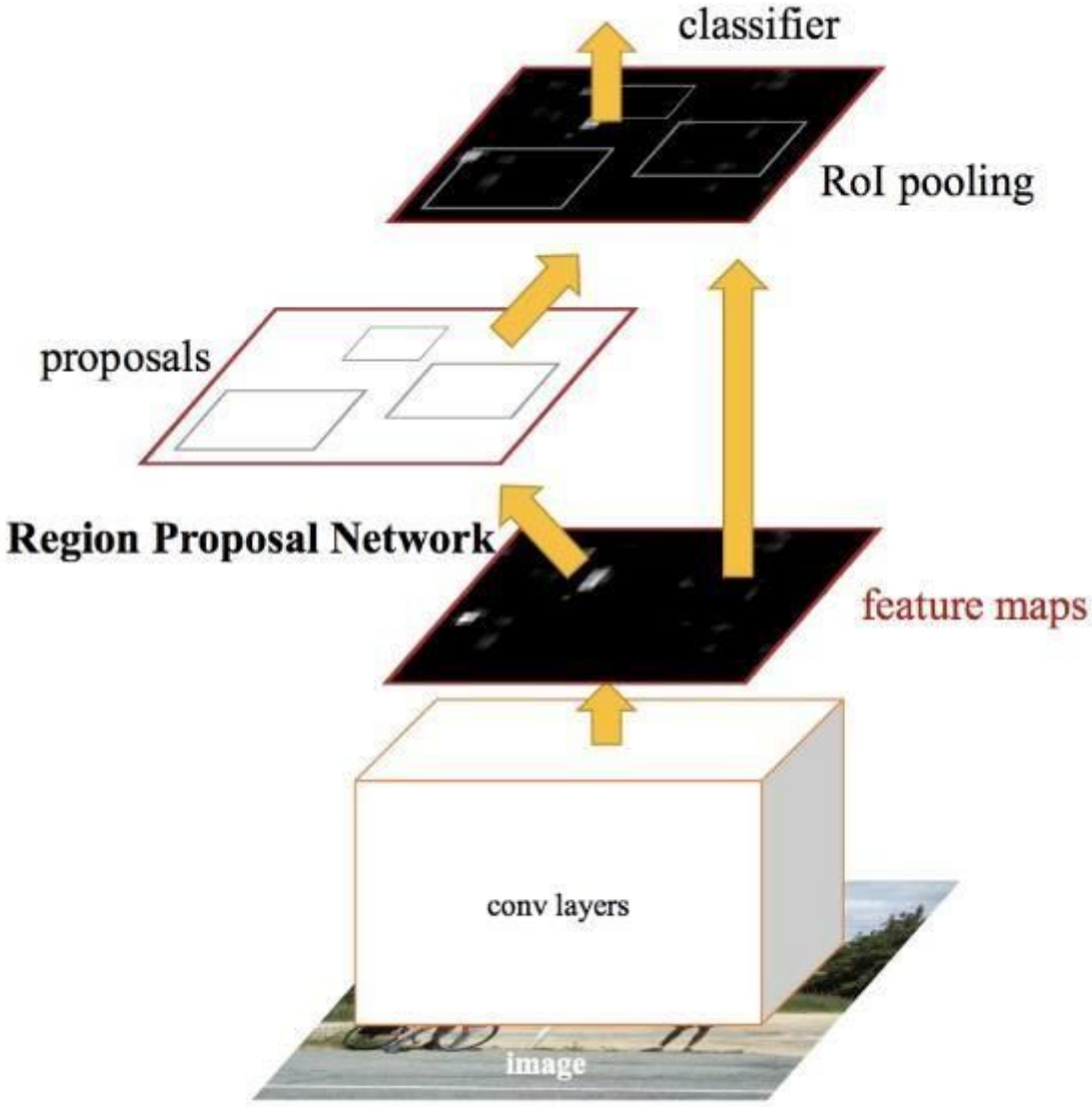


Figure 4

Faster R-CNN [3]

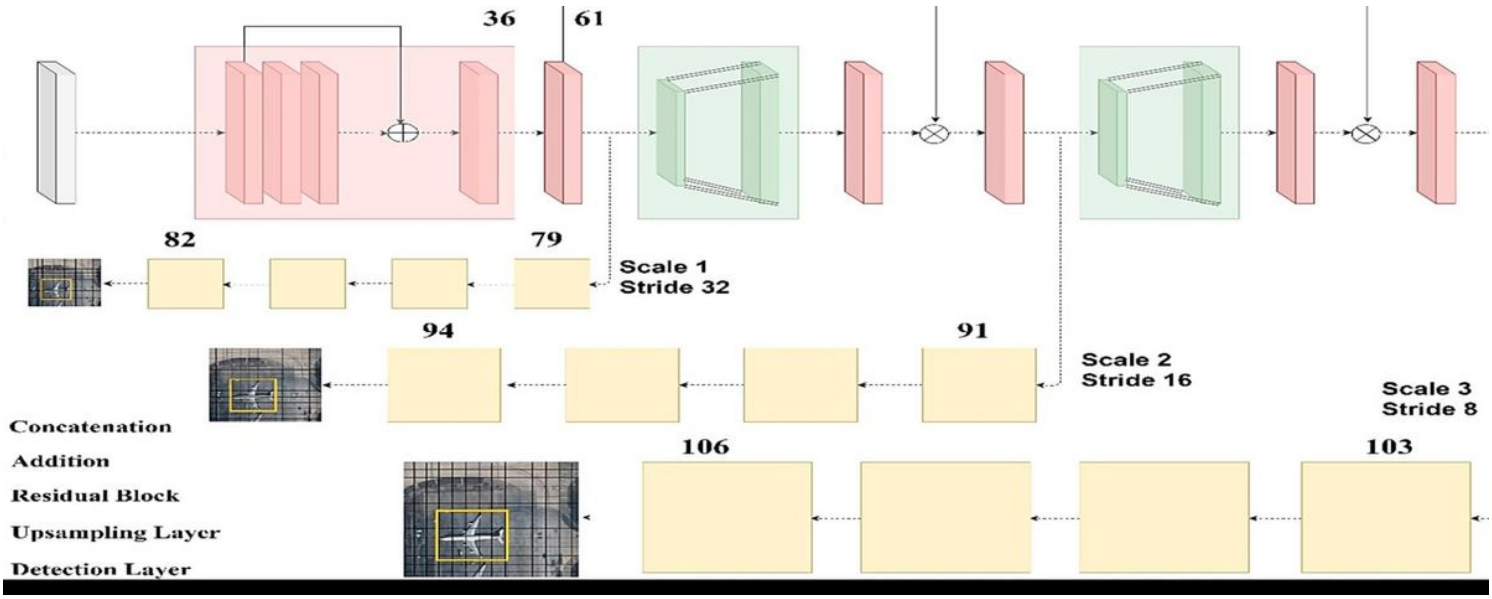


Figure 5

YOLO Architecture [10]

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
\sigma(x) &= 1/(1+e^{-x})
\end{aligned}
$$

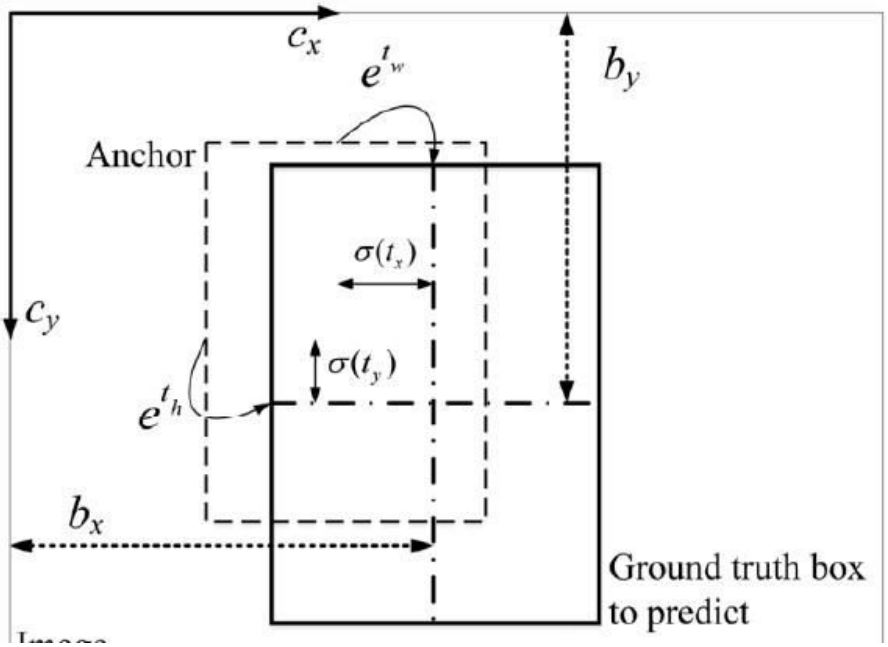


Figure 6

Bounding box forecasting [15]

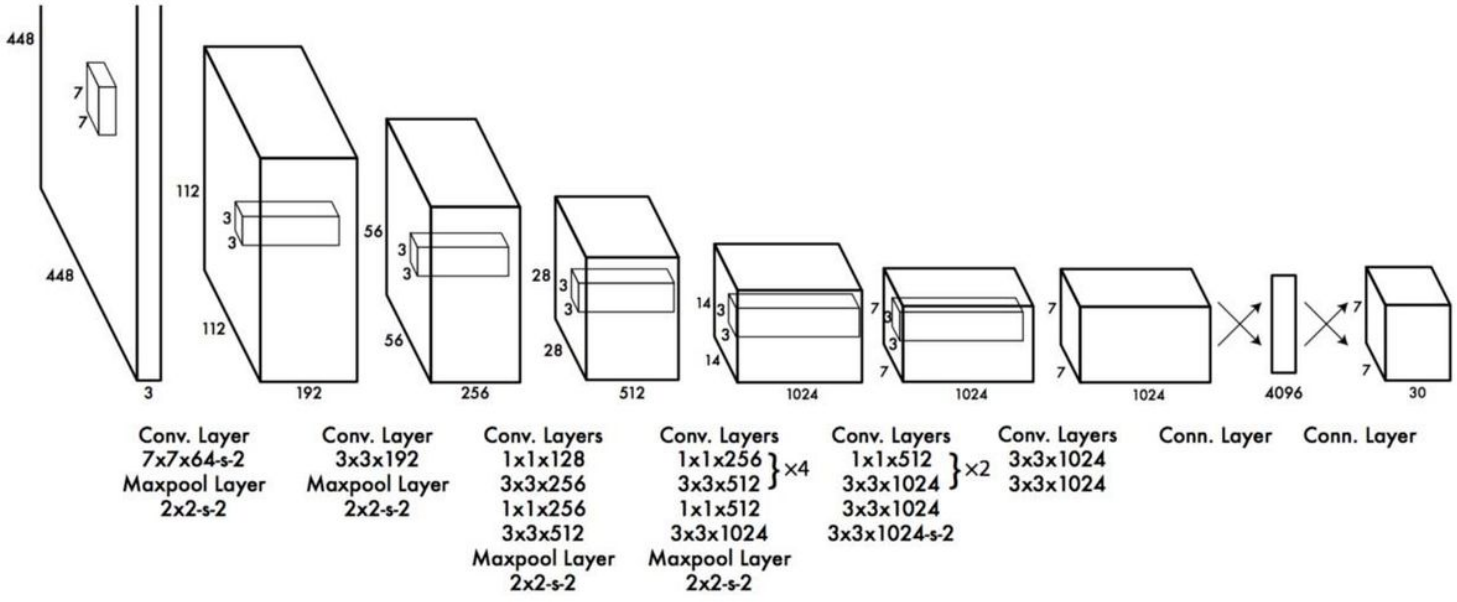


Figure 7

YOLO Model ConvNet [18]

Figure 8

Annotation pipeline [2]

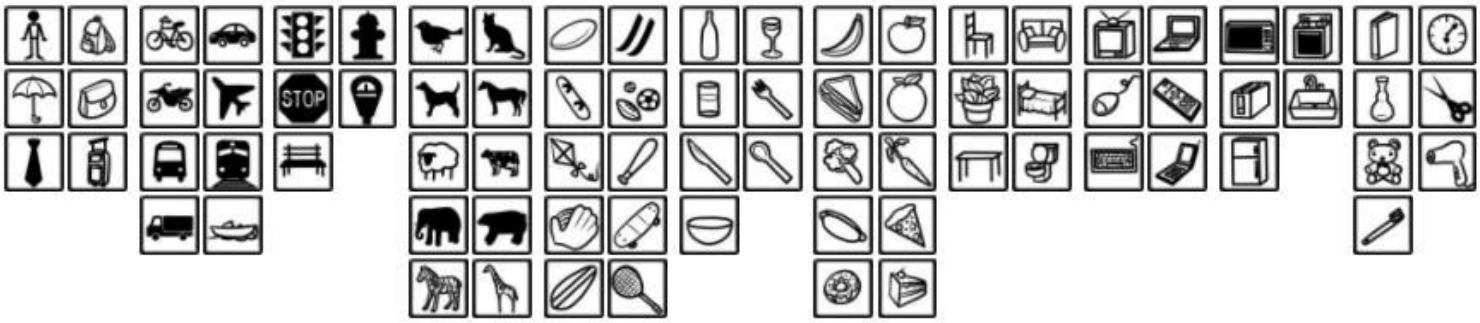


Figure 9

Categories of images [19]

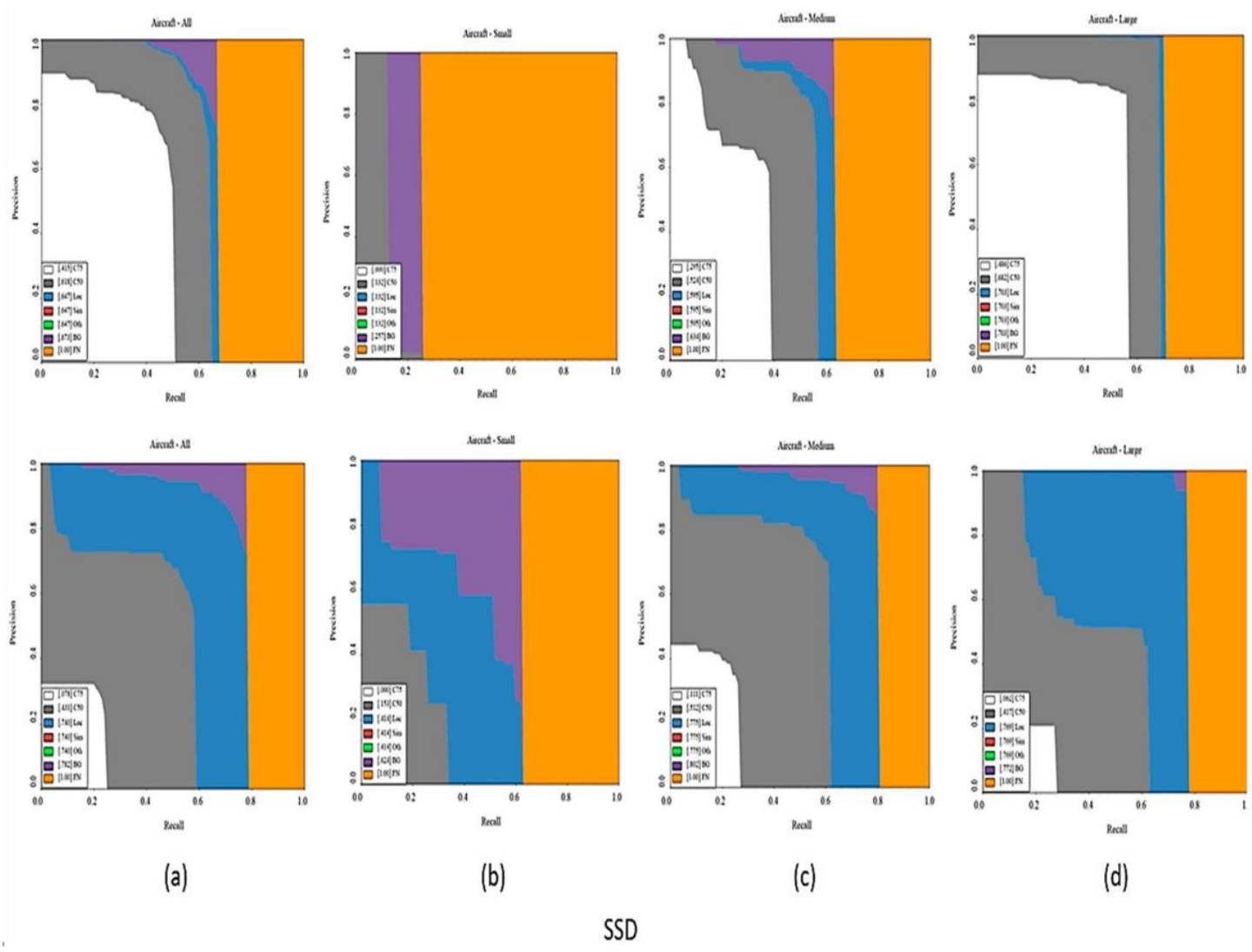


Figure 10

Graph for SSD [10]

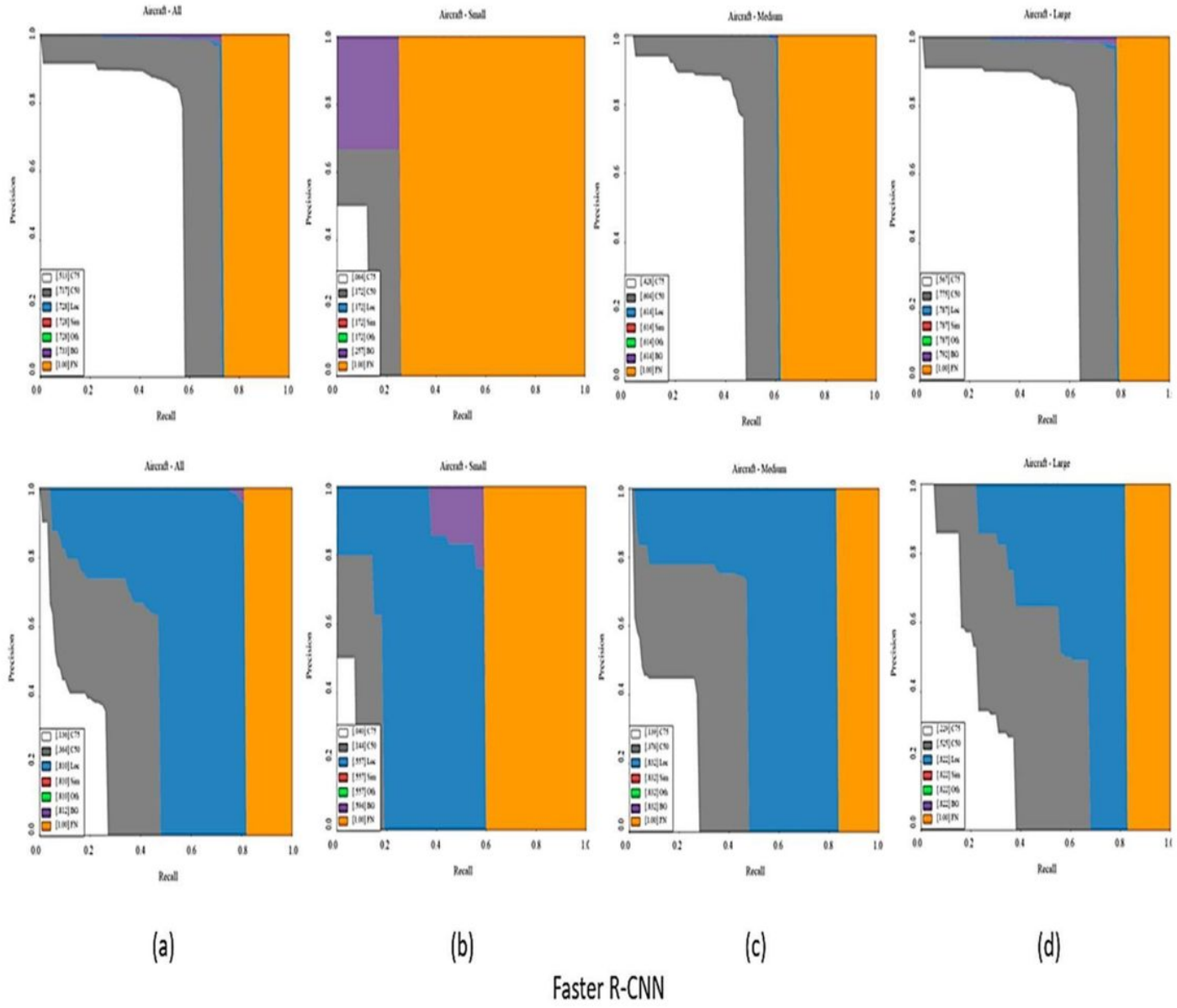


Figure 11

Graph for Faster RCNN [10]

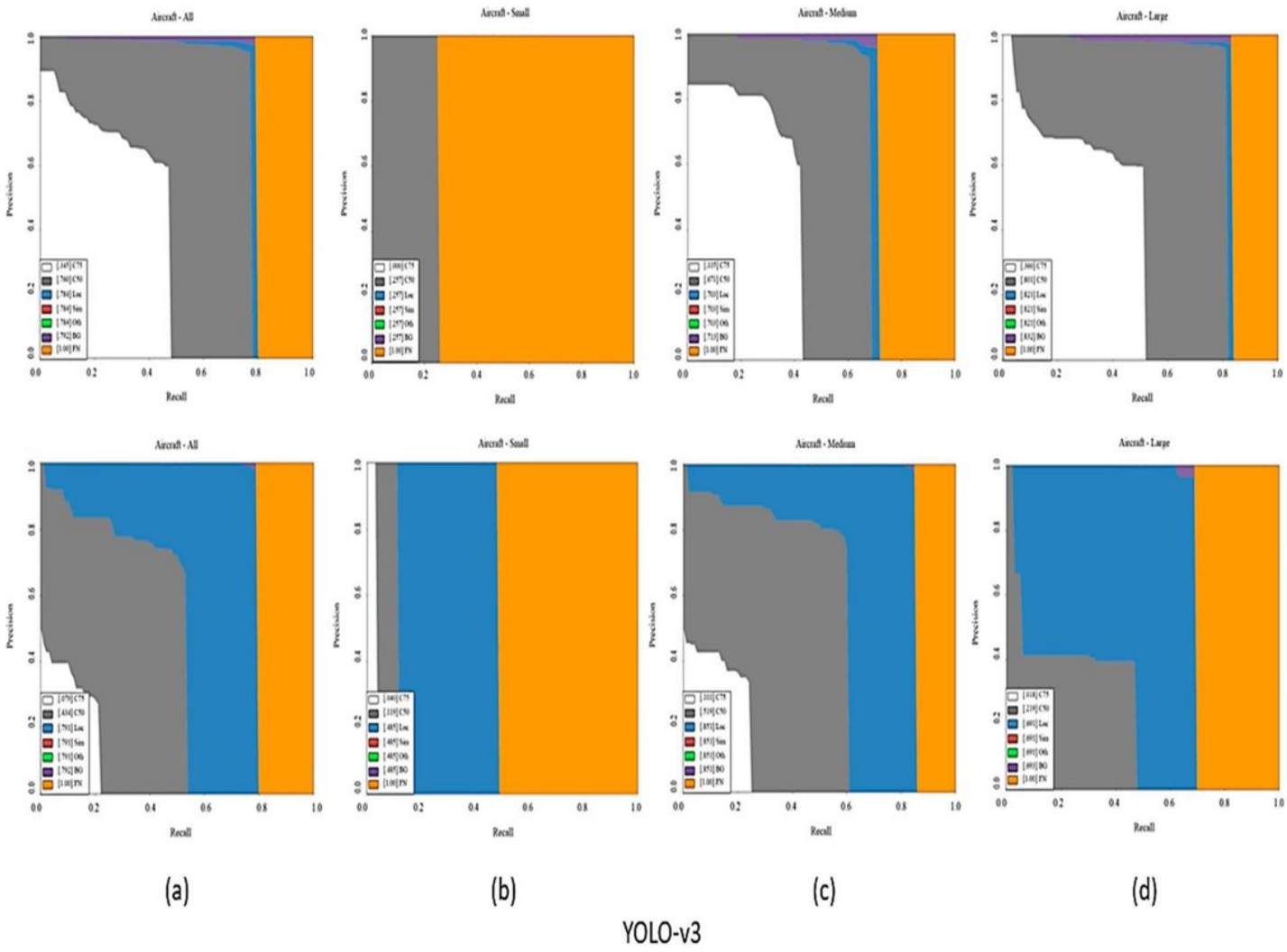


Figure 12

Graph for YOLO [10]