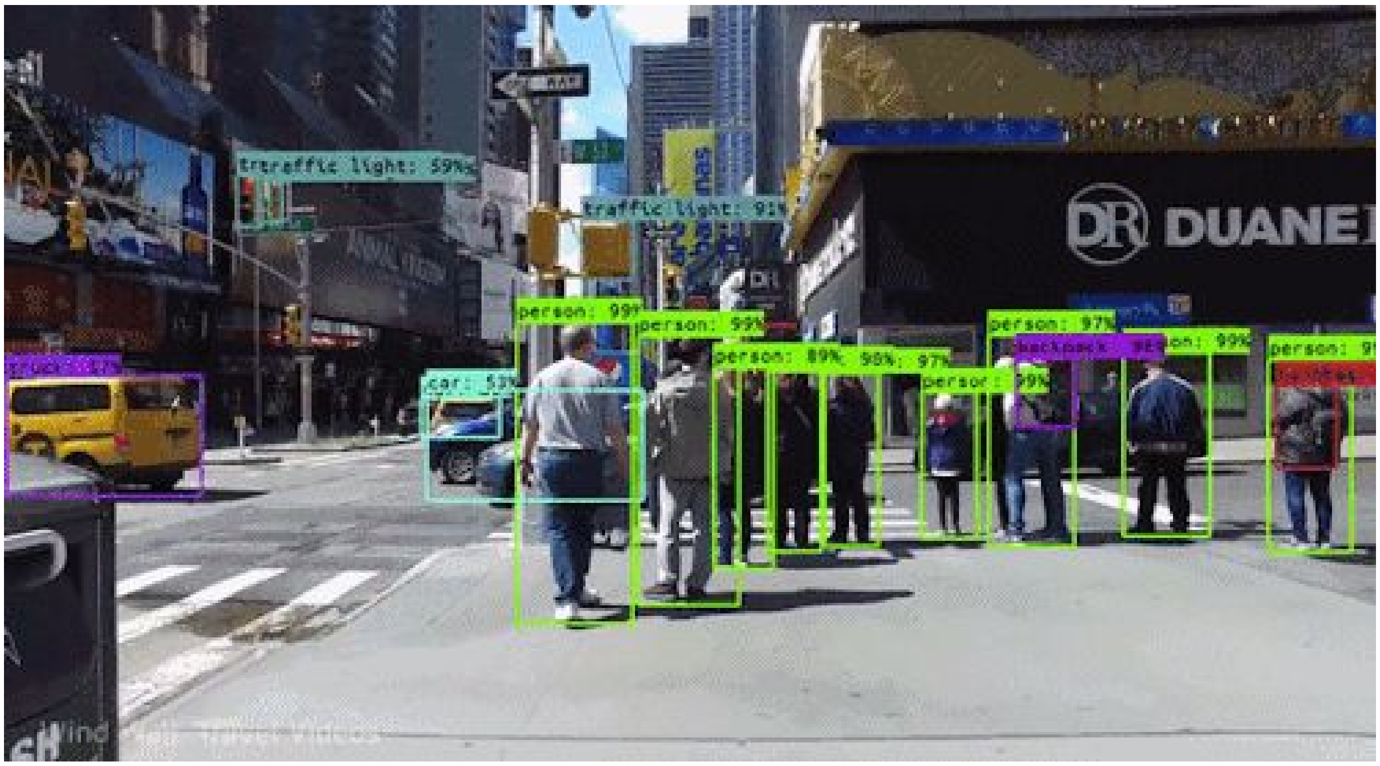
Tutorial On Object Detection

(From Base To Retinanet Algorithm)

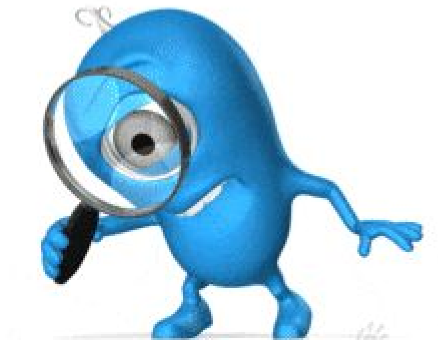
(Focus Only on Deep Learning Algorithms)



Before we start the tutorial ......

Having so many questions about Object Detection???

Confused with so many tutorials and concepts of Object Detection Algorithm???



No problem!! This tutorial will try to give you a clear and easy view of Object Detection Concepts.

This tutorial provides the intuitive ideas of Deep Learning Approach to solve such kind of problems and focuses on the latest method of Object Detection i.e.RetinaNet.

What Can We Learn From The Tutorial ??......

Through this tutorial, I will try to clear all your doubts about object detection. Here, I will try to give an easy and simple visualization for the concepts of object detection. Everything will be explained in very short and simple way with some visualization, so that even a person who is going to start his study about this topic can understand and get an intuitive idea about it.
This tutorial will try to answer questions like what is object detection, what is the procedure to do this task, why deep learning gives us good method to do it, why convolutional neural network gives us good results for such kind of task and so on...In short, our tutorial will consist of following concepts :

- **What is object detection?**
- **Applications of object detection**
- **Why deep learning and Convolutional Neural Network (CNN) ?**
    - **Components of CNN layers as convolutional layer, Relu activation function and max pooling layer, some loss functions for training.**
- **Object Detection as a classifier**
- **Two Stage Object Detection Algorithm (Some famous recent algorithms of object detection)like**
    - **RCNN,**
    - **Fast RCNN,**
    - **Faster RCNN etc.**
- **Single stage Object Detection Algorithms like**
    - **YOLO,**
    - **SSD and**
        - **NMS (Non Max Suppression)**
        - **IoU(intersection of Union)**
    - **Retinanet Object Detector**
        - **Architecture of RetinaNet**
        - **ResNet**
        - **Feature Pyramidal Network**
        - **Focal LOSS**
- **Links to download the papers of above described Algorithms**
- **Links For object Detection Dataset**
- **Links for available code of the algorithms for RetinaNet and SSD**
- **Links for the downloading the and installing the dependencies and framework for deep learning**

- **Links of Some Basic ideas for coding through Keras and Tensorflow as backend**
- **Results of the Algorithms RetinaNet and SSD**
  - **Problems and Solutions in training of RetinaNet**
  - **The new results after training the SSD and SSD + focal loss for udacity dataset**
  - **Comparative results of SSD512 and SSD 300 after testing with the provided trained weights.**
- **Conclusion**
- **Future Scope**
- **References**

Basically, I will try to build up some concepts related to object detection.

I will try that we don't need any audio clips to explain the concepts described here, and will try to explain all concepts with the help of some visualization.

So Let's Start…….

What Is Object Detection?

Before we start defining object detection concept, we should get some intuitive idea about object classification, object localization and how they are different from object detection .

Object Classification

In object classification task, a classification algorithm takes an input image and if the image contains some object, then it predicts the class of that object. To be more specific, if the algorithm is made for classification of cat,dog or none of them (3 classes) then after giving an image to the algorithm, it will predict whether the image contains cat or dog or none of them. We can see this through the given example: in which image contains cat and an algorithm is predicting the probability of the object to be a cat i.e. 0.99.(Figure 1)

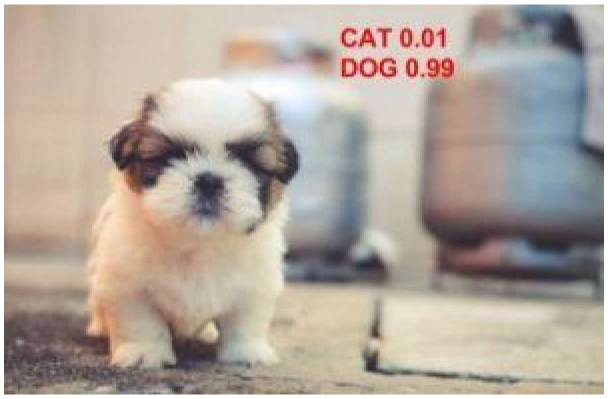


Figure 1

Image with cat and dog in a same image? What to do? (Figure 2)

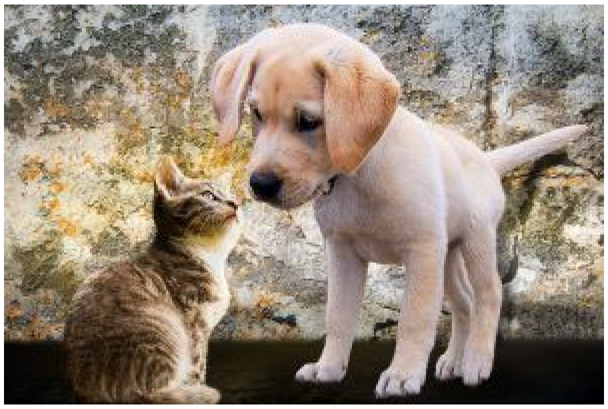


Figure 2

If there are more than one object in a single image, then to solve this problem we can train a multi label classifier which can predict other classes of objects at the same time.

Object localization

In object classification, we just came to know the class of the object in the given image, but still we don't know where the object is located in the

image. So to identifying the position of the object; when a class is determined by the classification algorithm, in the given image is called as object localization. So if the class of the object is not known then we have to predict the location of the object as well as the class of the object. This process is done by object detection algorithm.(Figure 3)

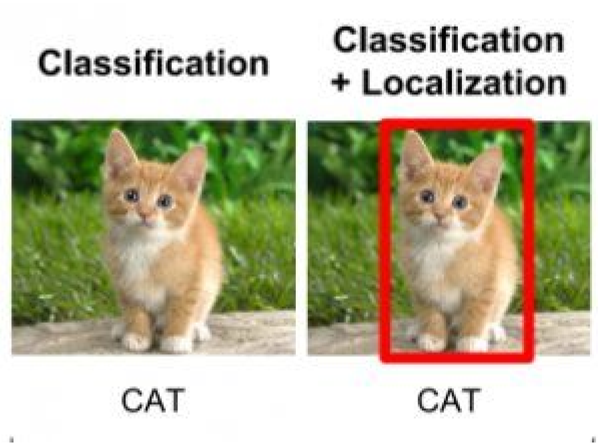


Figure 3

Object Detection

To predict the location and as well as the class of the object in a given image is called as the object detection. So like multiple object classification problem, we can detect multiple objects in a single image. So classifying and locating multiple objects in a single image is done by a single algorithm is called as "Object Detection". We can locate the object in the image by drawing the rectangular bounding box around the object. So now our prediction for each instance of the object in the image contains five variables such as class_name,
bounding_box_top_left_x_coordinate,bounding_box_top_left_y_coordinate,bounding_box_width and bounding_box_height.

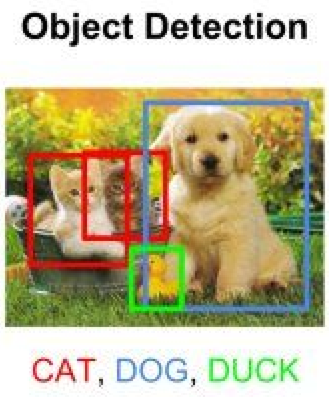


Figure 4

Now, we got the intuitive idea of object classification and object detection and difference between them. So let's move on to the next section of the tutorial that is why do we need deep learning and CNN for this task, but before that we should have idea about the applications of the object detection task.

Applications of Object Detection

Why do we need object detection :

1. **Video Compression**
2. **Video surveillance**
3. **Vision based control**
4. **HCI**
5. **Medical Imaging**
6. **Augmented Reality**
7. **Robotics**
8. **Self Driving Cars etc.**

Figure 5

Why do we need Deep Learning and CNN?

Before deep learning concepts, we were needed, some hand crafted features to classify the objects like edge detection,corner detection, color histogram etc.; which were not so robust and accurate to classify an object. Then after getting a great accuracy in Visual Recognition task through deep learning everybody started analyzing and focusing on the deep learning concepts due to its robustness,accuracy and easiness etc.. As we can see from below graph (Figure 6).

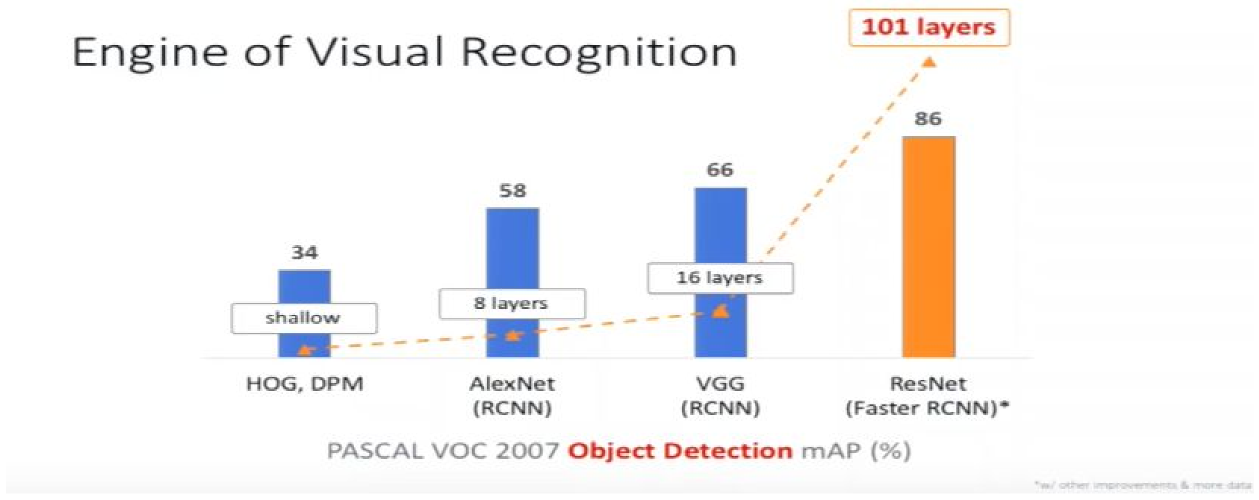


Figure 6

Why Convolutional Neural Network (CNN) ?

But before answering this question, let's see what is CNN and how it works?

Components of CNN

Usually a single layer of CNN consists of three components as below and deep neural network for object detection is designed as a combination of many layers of CNNs :

- **Convolutional Operation on input image**
- **Relu as a activation function**
- **Max Pooling**

Convolutional Operation on input image

Let's take an input image of size 6*6*1 and filter with 3*3 as shown in following visualization, then Convolution operation will give an output as below :

As we know discrete convolution is defined with the below formula

$$\begin{aligned}(f * g)[n] &= \sum_{m=-\infty}^{\infty} f[m]g[n-m] \\ &= \sum_{m=-\infty}^{\infty} f[n-m]g[m].\end{aligned}$$

(commutativity)

Therefore, in this part of CNN we do element wise multiplication of the filter with the elements of the same size of the block of the input image, then the result will be the sum of those multiplication. As we can see from the visualization : (Figure 7)

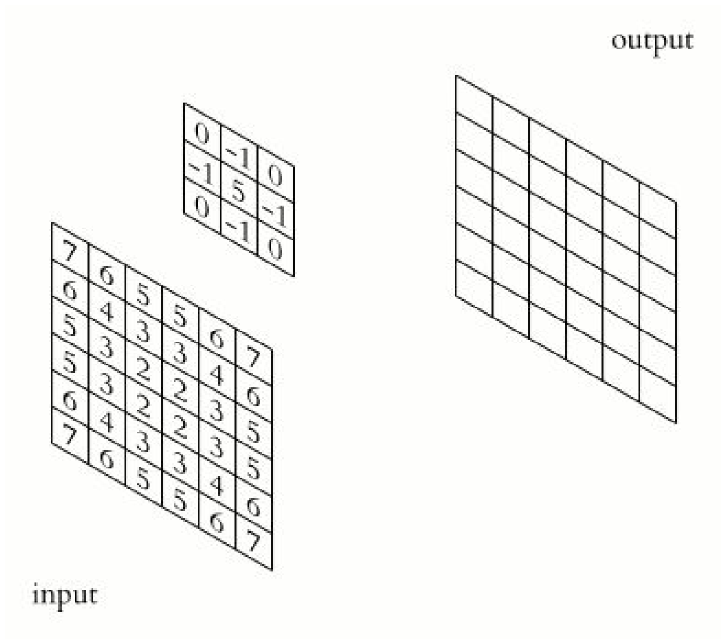


Figure 7

How to describe a convolutional operation in RGB channel image?

Then the procedure will be as follows, just add the outputs from all the channels to give the output for that particular location. Consider visualization : (Figure 8)

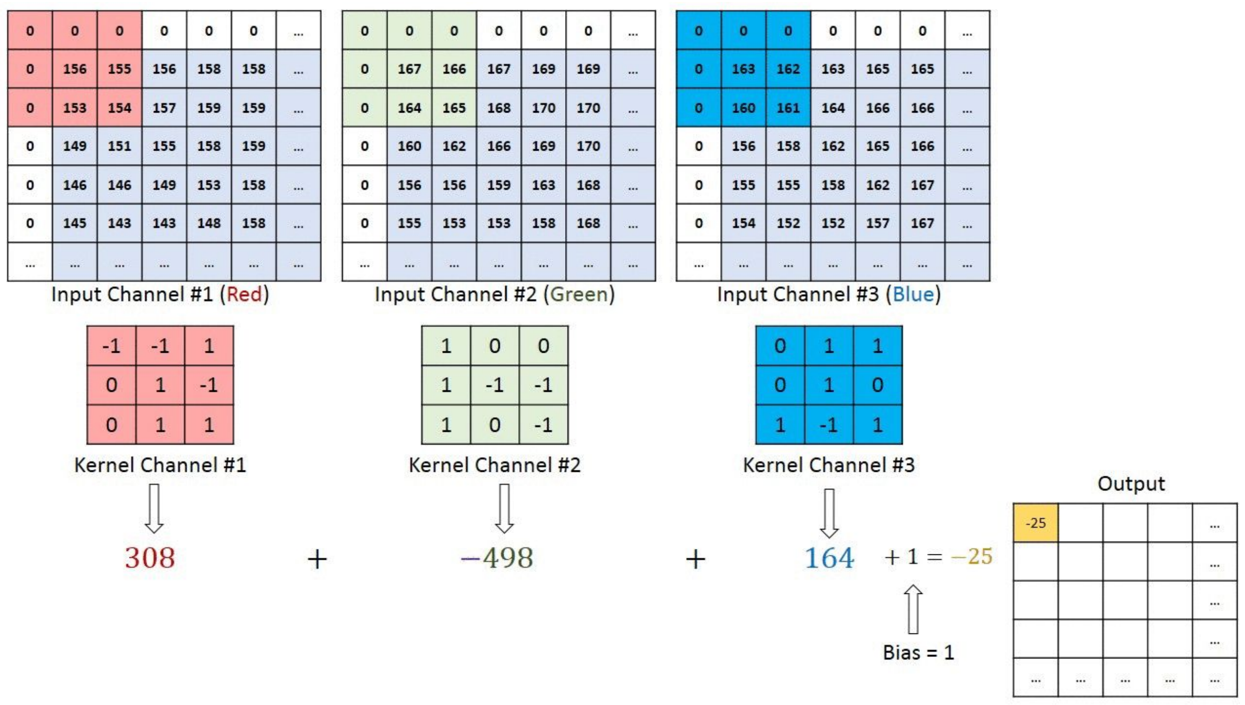
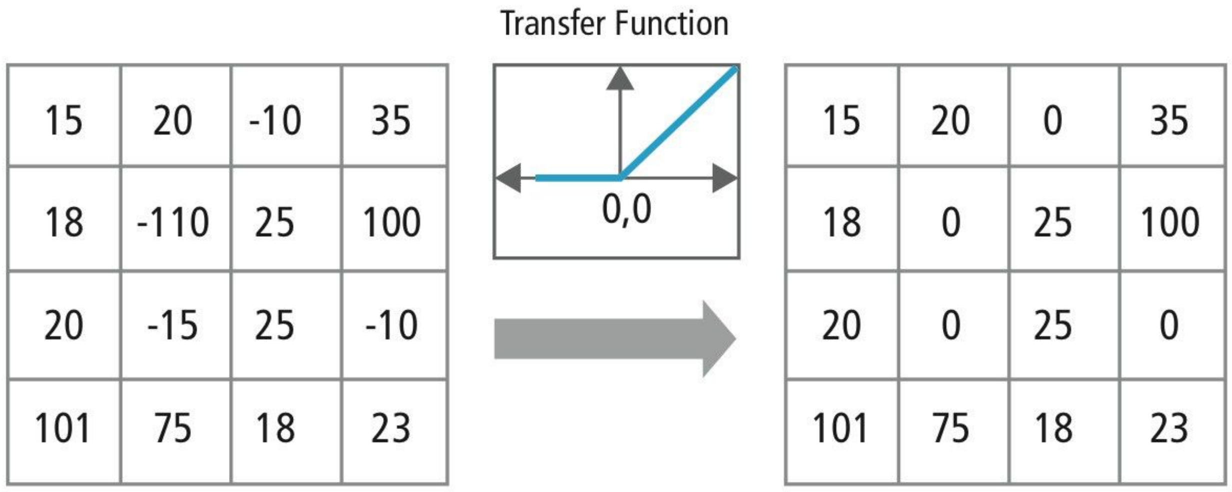


Figure 8

We can get multiple features of the same image by defining the multiple filters (kernel).

ReLu as a activation function

Relu stands for rectified linear unit. It introduces the nonlinearity in the network, because most of the complex functions are nonlinear so we need to introduce some non linearity in the network with the help of the activation function. There are other nonlinear functions as well like softmax, tanh etc., but now a days relu is used widely because of its unique functionality. Therefore, here I have given the intuitive idea of ReLu activation function. Consider visualization : (Figure 9)



15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Figure 9

Max Pooling layer

To downsample the feature maps we need to reduce the size of the image so we required the max pool or average pool layer in the CNN. We can get an intuitive idea of the max pooling with the following visualization ; (Figure 10)

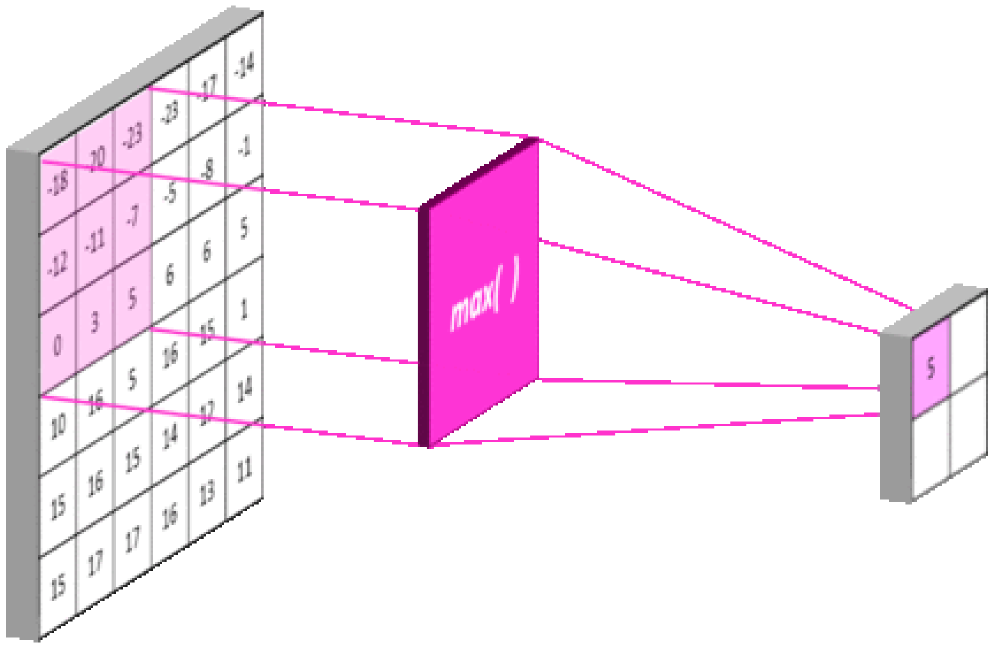


Figure 10

So above defined components are the basic building blocks of the CNN layer. A deep CNN is the combination of such single layer. So the output of the CNN layer is called as feature maps since its giving the feature of the input image.

The main key feature to use CNN in class of image (2D and 3D signal) is the retaining of the local features of the image.In CNN we are not losing the local feature of the image since filters are also in 2D or 3D form , but we can lose this information when we make all 2D or 3D image into a single vector form. Moreover, we required very less number of parameters to make a CNN network as compared to fully connected layer in which all neurons are connected to the each input element. So less memory required due to shareability feature of the parameters in CNN. As we can visualize it from following figures.(Figure 11 and Figure 12)

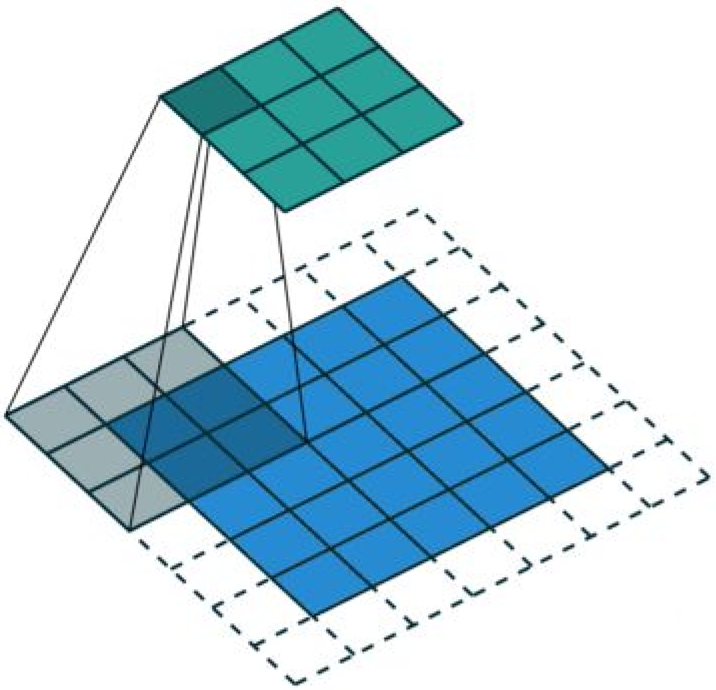


Figure 11

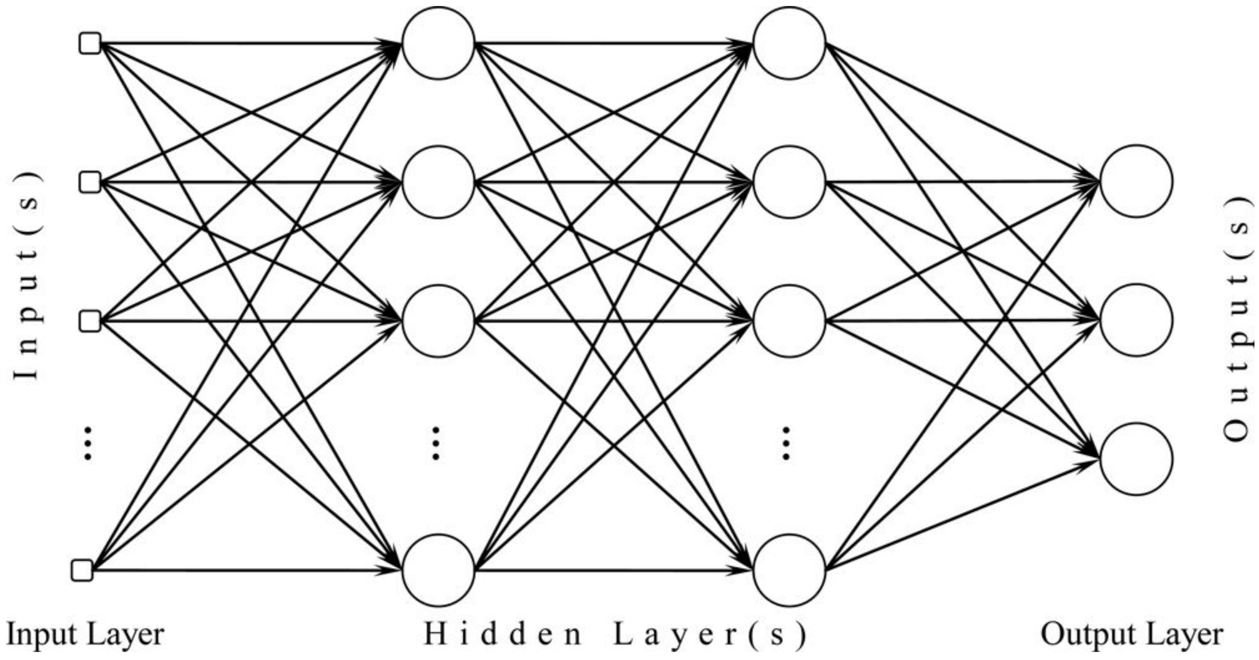


Figure 12

As we can see from figure 11, that only 3*3=9 parameters are required in the one layer of the CNN network for any size of the input image, but in the case of the Fully connected layer for such small image like 5*5=25, we need at least 26 parameters if there is only one neuron in the hidden layer as we can see from figure 12.

We can see one simple visualization for the image classification task of cat and dog with the help of CNN as below : (Figure 13)

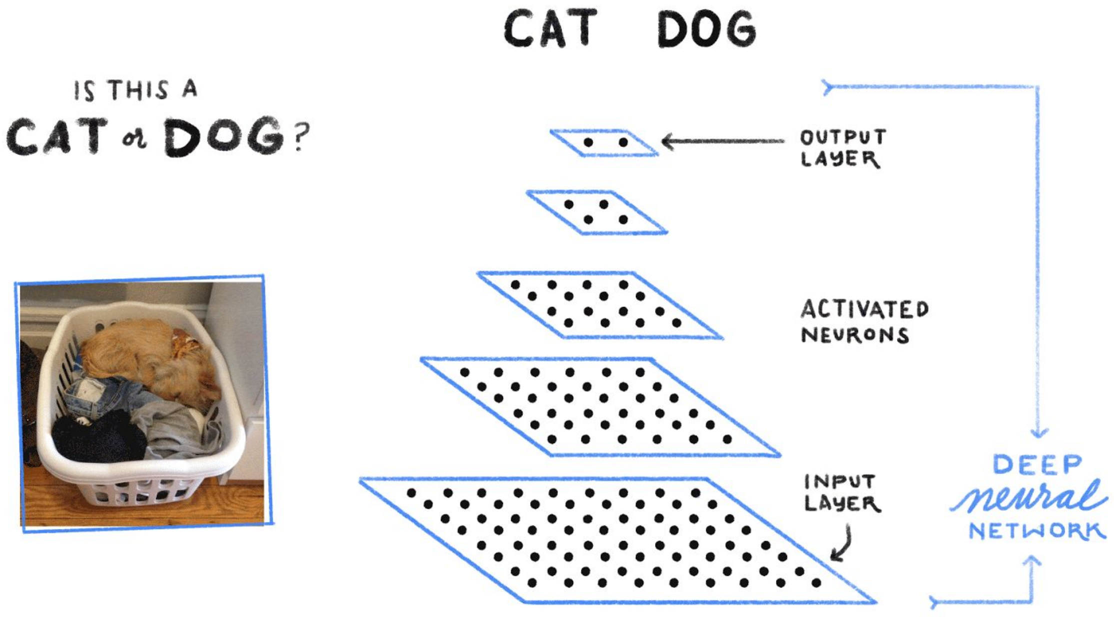


Figure 13

Some Loss Functions for Training

After forward pass, we need to train the network. Training means to model the network such that it can give a right prediction for some new unknown images. We need to make learn the neural network for this task and we need to modify the parameters accordingly . In this task we need some

technique that is backpropagation which will back propagate the error for the current model and then do the modification in parameters accordingly. Loss is defined as the difference in between predicted output from the network and known output of the image.

There are different kind of errors defined and for back propagating the error in the network, we have different kind of methods like SGD ,MOMENTUM,NAG,ADADELTA etc..but we can't discuss all methods here, Please consider the following paper for it .

Katarzyna Janocha 1 , Wojciech Marian Czarnecki "*On Loss Functions for Deep Neural Networks in Classification*". *[1]*

Object detection as a classifier

In this method, we just take windows of the fixed size from the input images at all the possible locations and feed these patches to an image classifier.

From this we will get to know the class as well as the location of the object with the help of the fixed class window. (Figure 14)

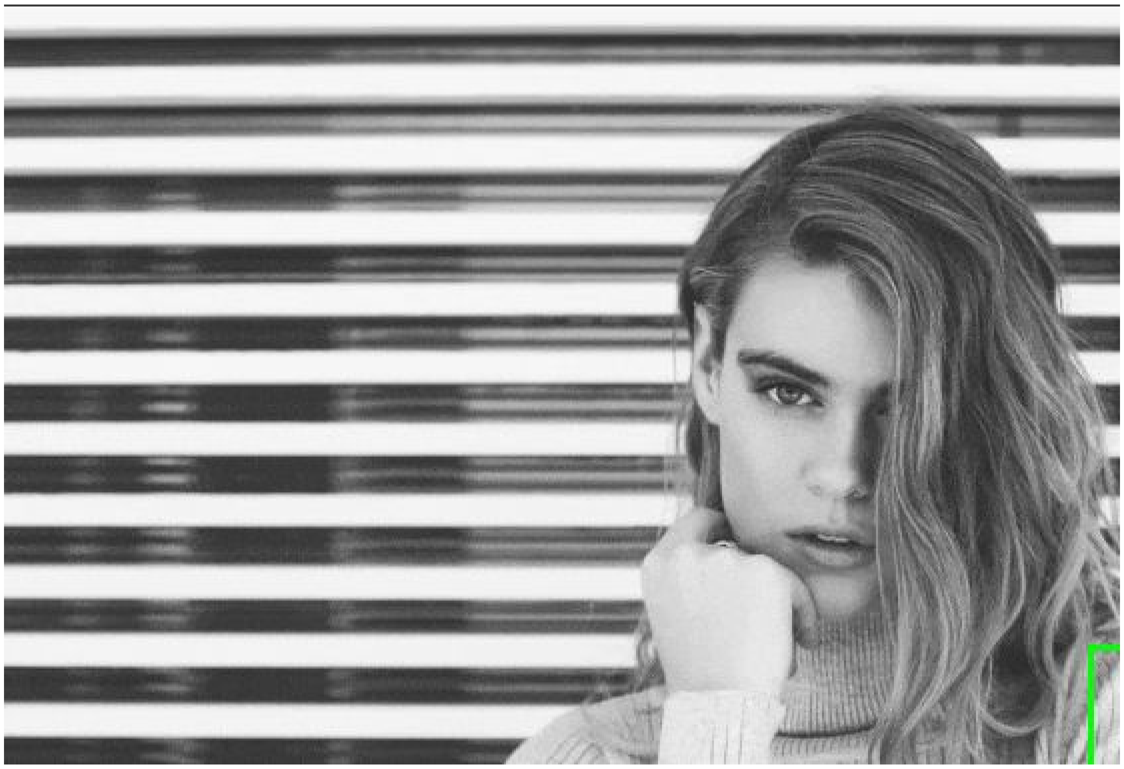


Figure 14

Simple right??

But problem with this method is as below:

Image scalability

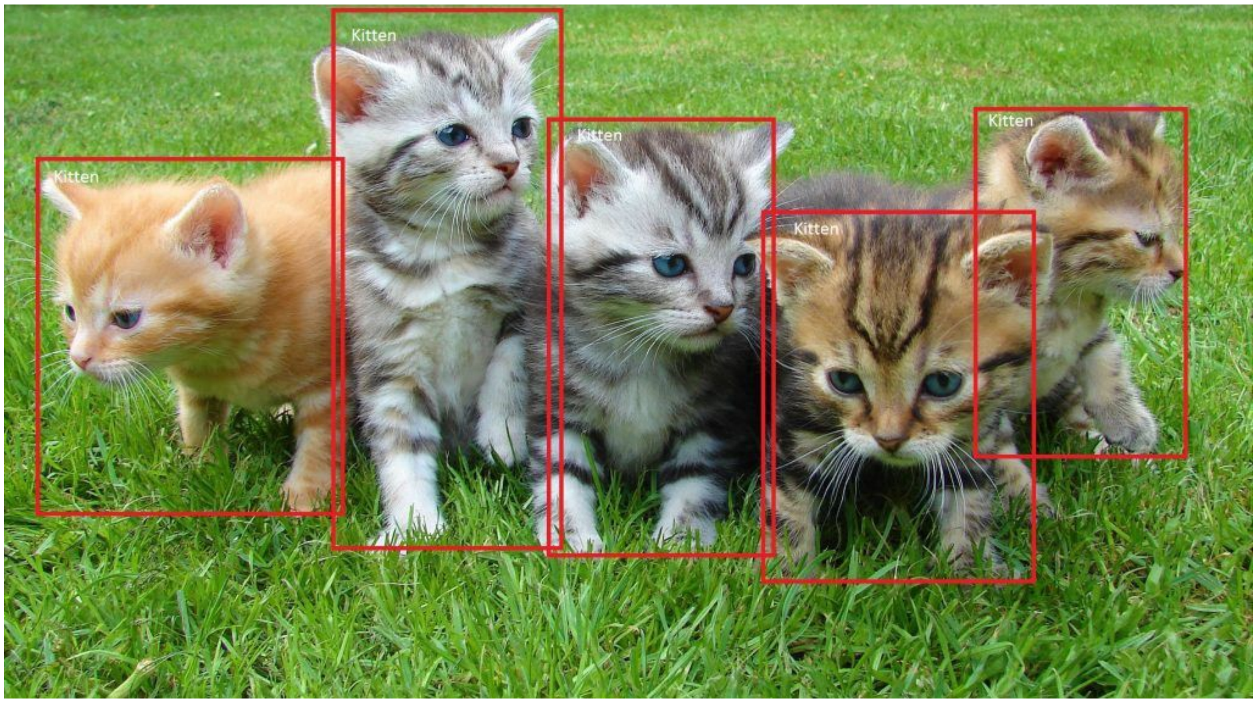


Figure 15 (Same object with different scales.)

So fixed rectangle box will not work in this case (figure 15). To overcome from this issue, we can resize the image at multiple scales and we count on the fact that our chosen window size will completely contain the object in one of the resized images.

Most commonly, resizing means to downsample the image (size is reduced). (figure 16)

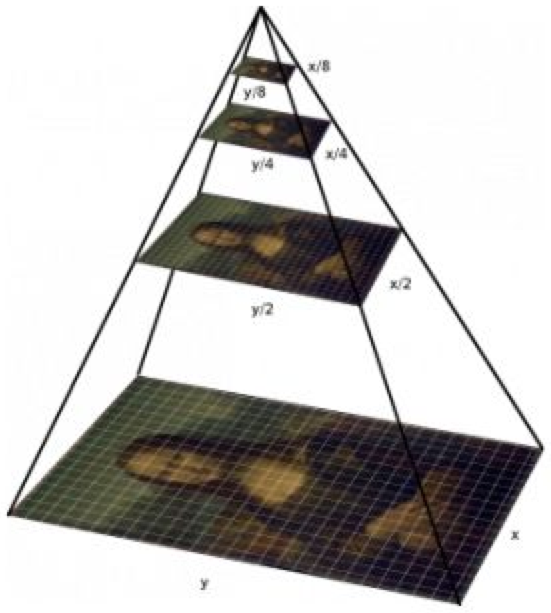


Figure 16

Example of this method is given in figure 17 . On each of the scaled images fixed size window is run. So all these windows are fed to the classifier algorithm to detect the object of interest. (This is kind of one shot object detection algorithm)

Figure 17

There are some other issues with this method as well, like its too slow and there is an aspect ratio problem (object can be in a different situation like standing mode or sitting position etc.)

To solve such issues other famous algorithm was also defined in deep learning concept. Which has been described in brief as below.

Note : To get an idea about Retinanet we have to begin with some famous object detection methods:

RCNN ,FAST RCNN, FASTER RCNN, YOLO ,SSD etc..

Two stage object detection algorithms

RCNN [2]

We basically modeled the problem of object detection as object classifier, but there was a big significant problem. CNN was too slow and computationally so expensive .It was not a great idea to run a CNN classifier on each of the scaled image patches generated by the sliding window detector. RCNN tried to solve this issue. RCCN basically stands for region based CNN object detector.

It proposed an algorithm in which an algorithm called selective search is run to find out the region of interest, which reduced the number of patches which needed to go through the cnn classifier block to detect the object of interest. We can get approx 200 bounding boxes for CNN classifier and it was much less in number as compared to an algorithm of CNN. Selective search uses local cues like texture, intensity, color and/or a measure of insideness, etc. to generate all the possible locations of the object. Now, we can feed these boxes to our CNN based classifier.

This can be clearly illustrated through the figure 18 as shown below. (Figure 18)

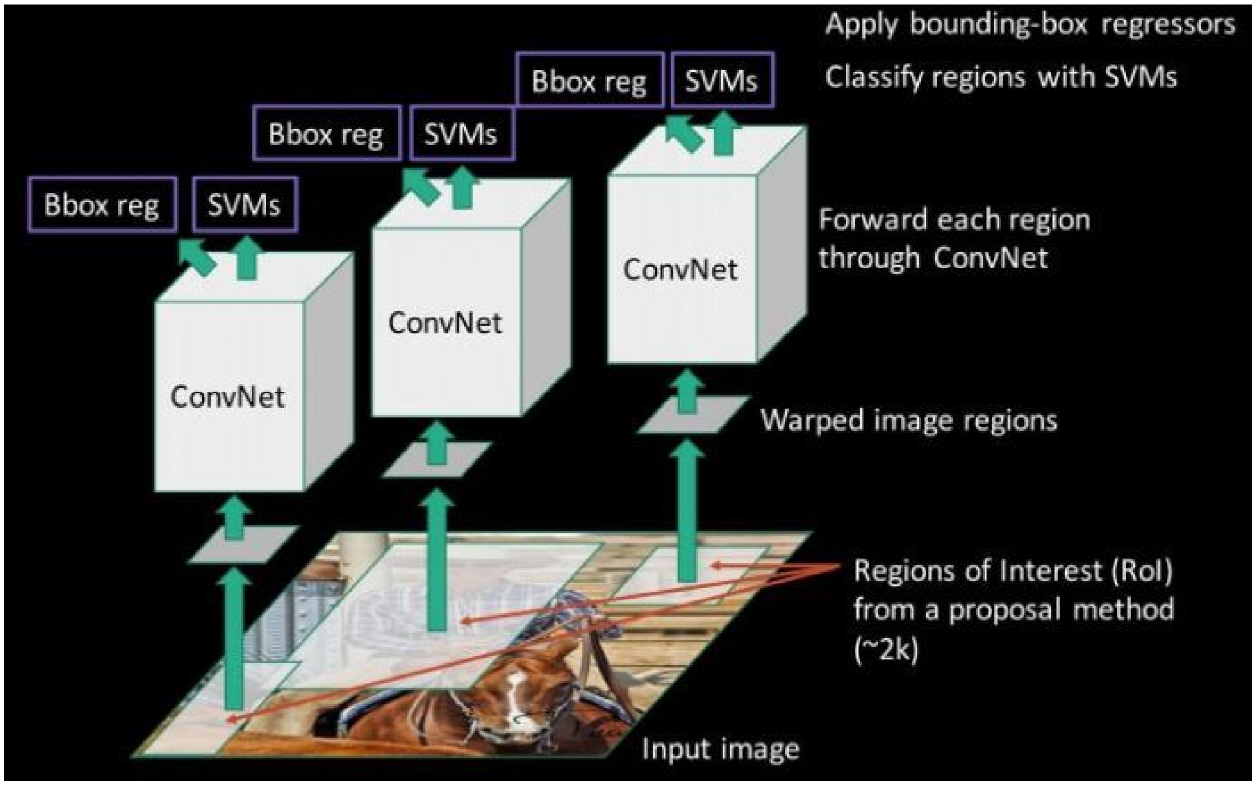

Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Figure 18

As we know, fully connected part of CNN takes a fixed size input so, we resize all the generated boxes to a fixed size (224×224 for VGG) and feed into the CNN part. Hence, RCNN can be described with the help of three steps .

1. **Run Selective Search to the given input image to get the region of interest.**
2. **Feed these patches to CNN, followed by SVM to predict the class of each patch.**
3. **Then optimization is done by bounding box regression separately.**

It is called two stage object detector because it need to determine the area of interest in first stage, then classification and bounding box regression is done in the second stage.

FAST RCNN [3]

After improving the deep CNN method to RCNN still the speed of the algorithm was too slow and to solve this problem again one more modified version of RCNN was introduced as fast RCNN, in which instead of finding out features for all region of interest object we first find out the features of the given input image, then impose the selective search of the input image on the finally achieved feature map by the deep CNN then apply bounding box regression algorithm and classifier algorithm on the selected region of interest which we got from selective search method. Again Fast RCNN method can also be defined into 3 steps as below which can be visualized by the given image as well : (Figure 19)

1. Performing feature extraction over the image before proposing regions with the help of deep CNN, thus only running one deep CNN over the entire image instead of 2000 CNN's over 2000 overlapping regions as in RCNN)
2. Then apply the region proposal method on it.
3. SVM is replaced by a softmax layer, thus extending the neural network for predictions instead of creating a new model.

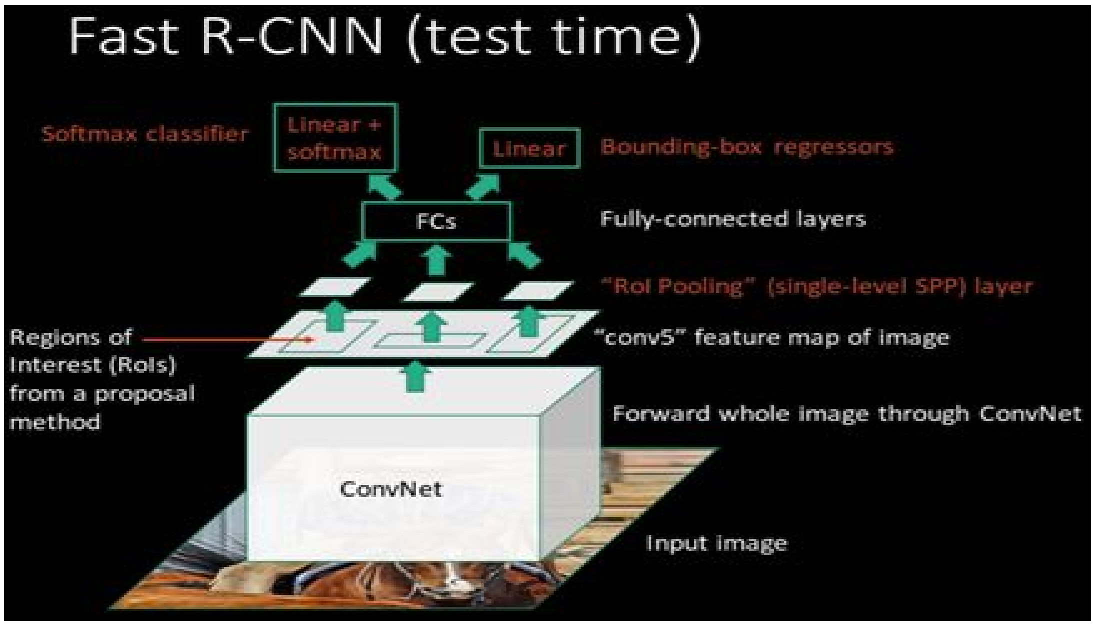
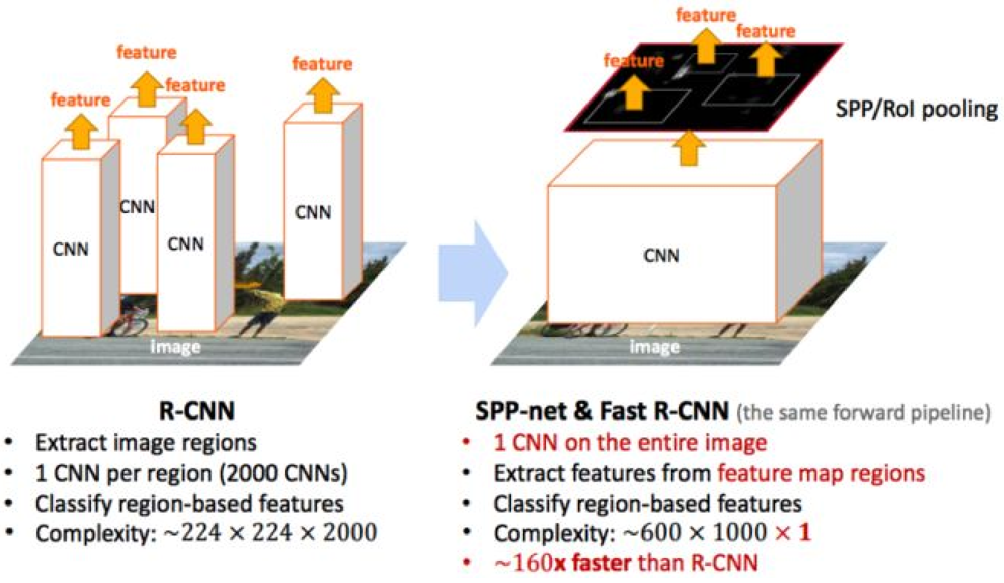

Fast R-CNN (test time)
Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Figure 19

Modification from RCNN to faster RCNN :

Softmax function is used to classify two or more than two objects from the image, it will provide the probability of the each object to which the object can be belonged to. (Figure 20) [3]



feature
feature
feature
feature
CNN
CNN
CNN
CNN
image
feature
feature
feature
SPP/RoI pooling
CNN
image
R-CNN
• Extract image regions
• 1 CNN per region (2000 CNNs)
• Classify region-based features
• Complexity: ~224 × 224 × 2000
SPP-net & Fast R-CNN (the same forward pipeline)
• 1 CNN on the entire image
• Extract features from feature map regions
• Classify region-based features
• Complexity: ~600 × 1000 × 1
• ~160x faster than R-CNN

Figure 20

Faster RCNN *[4]*

There is a little bit modification is Faster RCNN was done to achieve Faster RCNN and that was instead of using selective search for getting the region of interest author has applied region proposed network (RPN) to get the region of interest to make the fast rcnn to more faster. Something we can visualize as below from the figure 21 :

Here's how the RPN worked:

- **To handle the variations in aspect ratio and scale of objects, Faster R-CNN introduces the idea of anchor boxes. At each location, the original paper uses 3 kinds of anchor boxes of scale 128x 128, 256×256** and 512×512. Similarly, for aspect ratio, it uses three aspect ratios 1:1, 2:1 and 1:2. So, In total, at each location, we have 9 boxes on which RPN predicts the probability of it being background or foreground.

After getting output from the FPN again we can follow same procedure as we did in the fast RCNN. We add a pooling layer, some fully-connected layers, and finally a softmax classification layer and bounding box regressor. In a sense, Faster RCNN = RPN + Fast RCNN. (Figure 22)

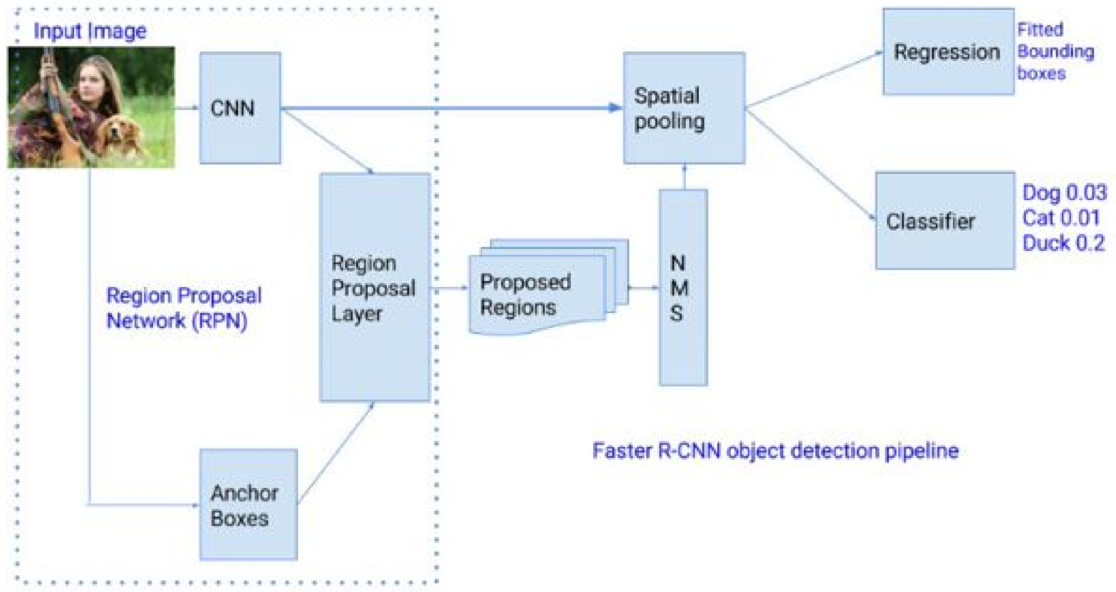


Figure 21

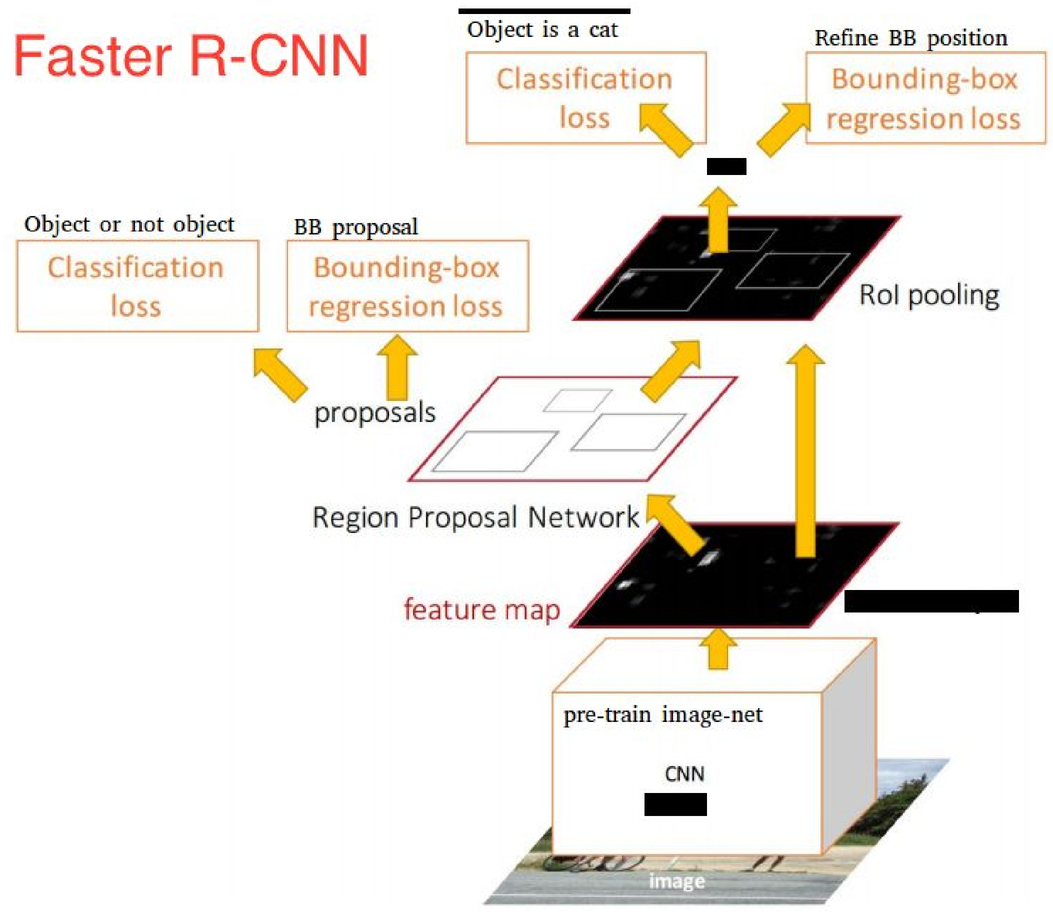


Figure 22

Let's' have a quick comparison of the performances of the above described two stage object detectors.

	R-CNN	Fast R-CNN	Faster R-CNN
Test Time per Image	50 Seconds	2 Seconds	0.2 Seconds
Speed Up	1x	25x	250x

Table 1

Like this, Faster R-CNN achieved a much better speeds and gives us a state-of-the-art accuracy in above described two stage object detection algorithm . Still more to do with object detection algorithm because still the algorithm is too slow. Even though it gives us most accurate object detector.

One stage object detection algorithms

YOLO *[5]*

It's time to increase the speed of the detector. So one stage object detector is defined here to increase the speed of the model. This algorithm gives us a significant increase in the speed of the model, but accuracy of the model get decreased as compared to two stage detector. (Figure 23)

With each iteration a classifier makes a prediction of what type of object is in the window. It performs thousands of predictions per image. Because of this sliding process it works pretty slowly.

It uses the concept of the bounding box as described in the Faster RCNN with different size and aspect ratios. The biggest advantage of the YOLO model is how it got its name—You Only Look Once.

In this algorithm, the input image is divided into a grid of cells.Then, each cell makes an attempt to predict the bounding boxes with confidence scores for those boxes and class probabilities. Then the individual bounding box confidence score multiplies by a class probability map to get a final class detection score.We can see the illustration from Figure 23.

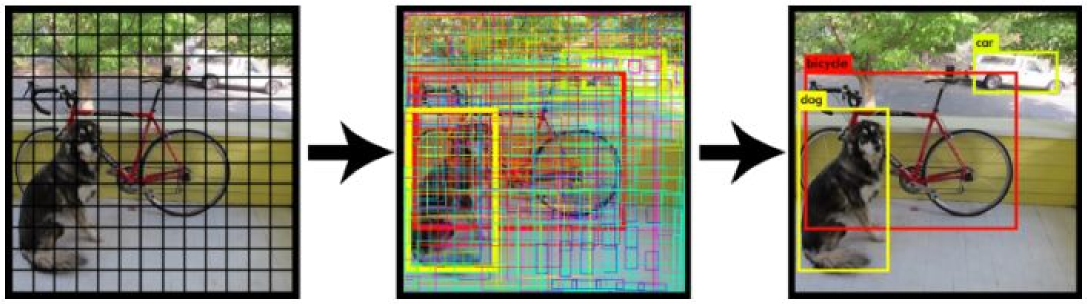


Figure 23

After this other versions of YOLO came such as YOLO2[6] and YOLO3[7] to increase the accuracy of the model along with the speed of the detector. YOLO2 [6] can recognise 80 classes. To run this you need to install some additional dependencies that are required for demonstration purposes (the model interface requires only TensorFlow). For more intuition about this algorithm I will provide some links to follow up if you are interested to read more about it. It uses the concept of NMS (Non Max Suppression), which is described in the next section of detector, i.e. SSD because SSD also uses the same concept to suppress so many bounding boxes from the single object.

SSD [8]

Another single shot detection method is SSD, which stands for Single-Shot Detector. The functionality of this detector is almost same as the YOLO and both introduced almost at the same time.Like R-FCN, it provides enormous speed gains over Faster R-CNN, but does so in a markedly different manner.

Faster RCNN was doing detection in two stage as it used a region proposal network to generate regions of interest; then in next step next, it used fully-connected layers convolutional layers to classify those regions. SSD

does the same task in a single stage that's why it is called as single shot "single shot,"which is simultaneously predicting the bounding box and the class of the object of the input image.

Concretely, given an input image and a set of ground truth labels, SSD does the following:

1. **Pass the image through a series of convolutional layers, yielding several sets of feature maps at different scales (e.g. 10x10, then 6x6, then 3x3, etc.) It uses the VGGnet (which is just deep CNN with relu activation function see Figure (24))**
2. **For each location in *each* of these feature maps, use a 3x3 convolutional filter to evaluate a small set of default bounding boxes. These default bounding boxes are essentially equivalent to Faster R-CNN anchor boxes.**
3. **For each box, simultaneously predict a) the bounding box offset and b) the class probabilities**
4. **During training, match the ground truth box with these predicted boxes based on IoU (NMS). The best predicted box will be labeled a "positive," along with all other boxes that have an IoU with the truth >0.5.**

To fix this imbalance, SSD does two things.

1. **Firstly, it uses non-maximum suppression (NMS); which is described in figures 25 and 26. In which to get the single bounding box around an object we will calculate the IoU(Intersection of union) for all available bounding boxes for that object, then group together highly-overlapping boxes into a single box according to the box confidence obtained by the IoU. In other words, if four boxes of similar shapes, sizes, etc. contain**

the same dog, NMS would keep the one with the highest confidence and discard the rest. (Consider figure 25 and 26)

2. **Secondly, the SSD model has used a technique called hard negative mining to balance classes during training. In hard negative mining, only a subset of the negative examples with the highest training loss (i.e. false positives) are used at each iteration of training. SSD keeps a 3:1 ratio of negatives to positives.**

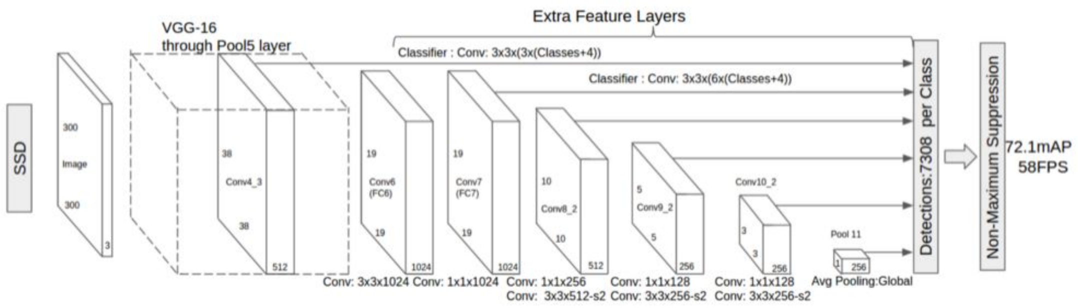


Figure 24

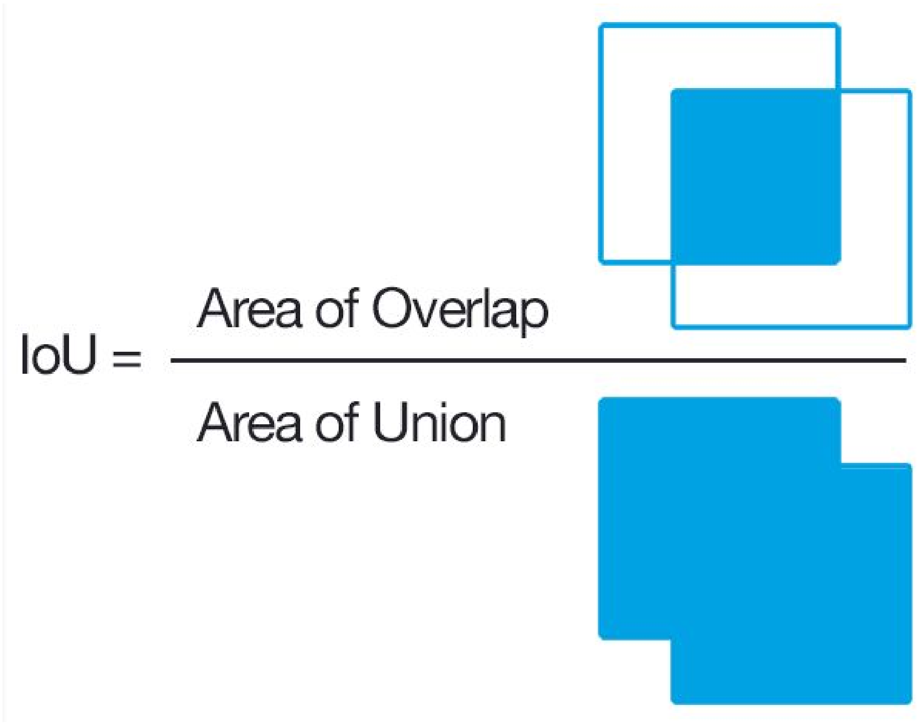


Figure 25

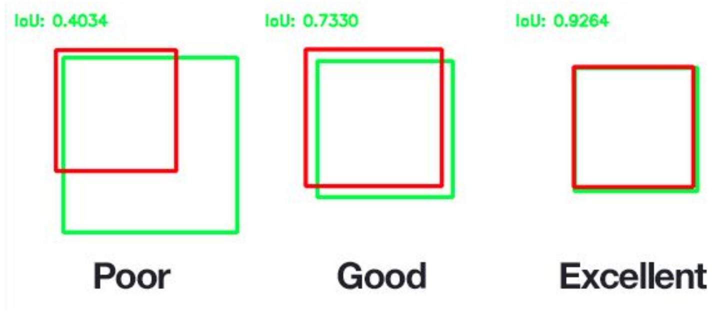


Figure 26

So the procedure of SSD can be described and visualize with the help of Figure 27.

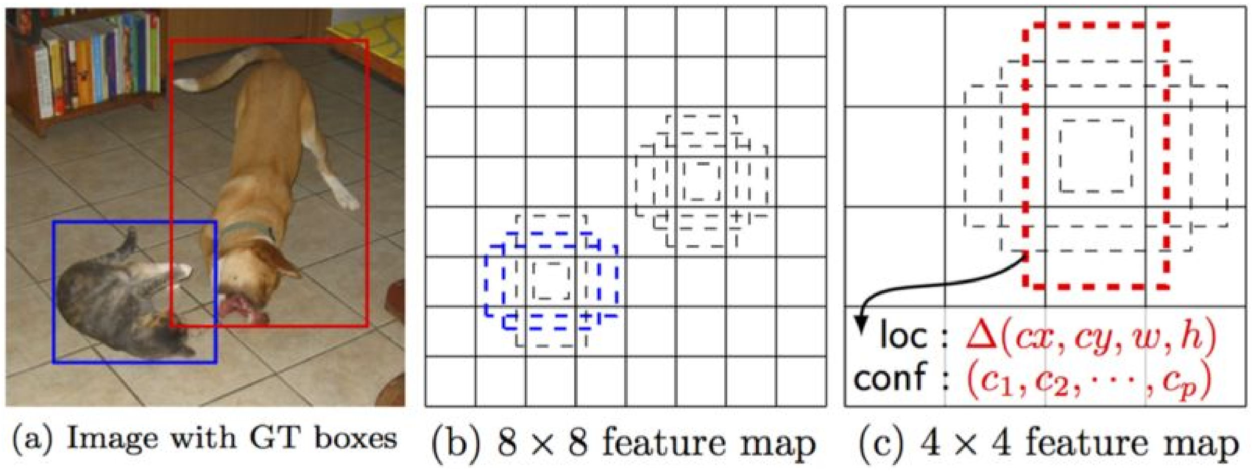


Figure 27

Same bounding boxes are used in each level of the feature maps.

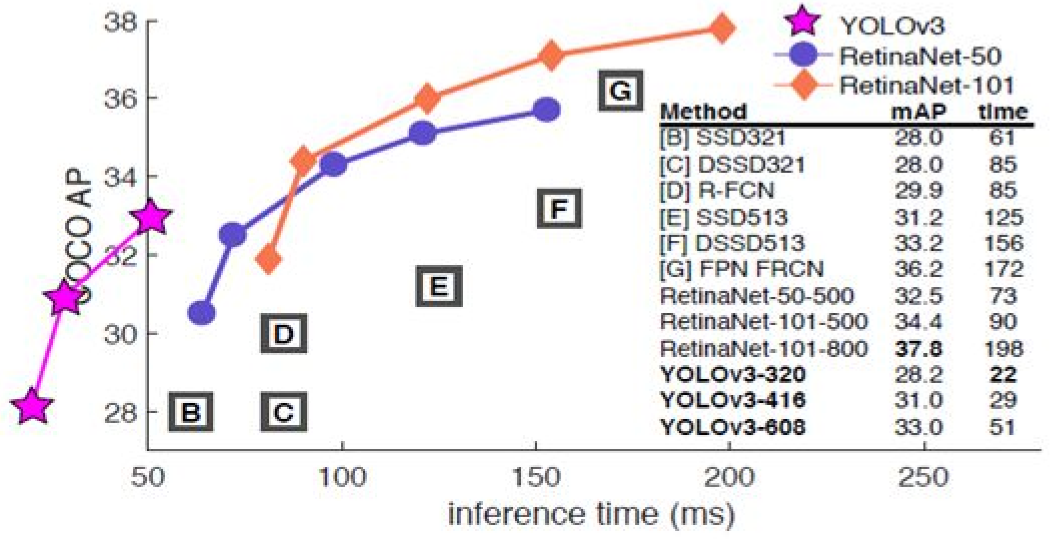
Ultimately, SSD is not so different from Fast RCNN. It simply skips the "region proposal" step, instead considering every single bounding box in every location of the image simultaneously with its classification. It gives a speedy and accurate model for the object detection.

We can find out other versions of the SSD as SSD v2 etc.

RetinaNet Object Detector *[9][12]*

This is the latest Object Detection method proposed by the research group of Facebook; which is also known as Facebook AI Research (FAIR), in Feb 2018. This paper is the second version of Retinanet introduced in 2017 during the ICCV 2017 conference. I would like to

give more focus on this work because of following reason :



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 28

Where, speed is in ms and AP is for accuracy

It is giving a great trade off between speed and accuracy.

To make object detection robust, accurate and as well as fast, above one shot detectors were introduced, but the problem with these object detection algorithm was that these methods cannot reach the accuracy of the two stage object detector. So the FAIR research group introduced a new loss function to make the detector robust and accurate for one stage detectors as two stage detector.

Architecture of RetinaNet

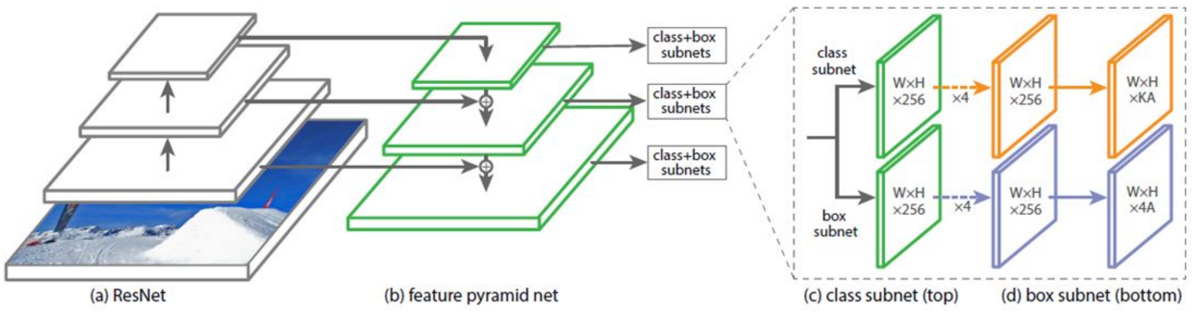


Figure 29 : RetinaNet Architecture

As we can see from the above figure 29, to fully understand the concept of Retinanet object detector, we have to understand following three concepts of Deep Learning.

1. **ResNet : RetinaNet uses ResNet a backbone architecture for extracting the features of the image.**
2. **Feature Pyramid network**
3. **Focal Loss for Dense Object Detection**

Let's discuss each concept one by one.

1. *ResNet* [10]

What is ResNet ? Why do we use it ?

The very deep neural network is difficult to train because of vanishing and exploding gradient types (in which multiplying too many small parameters will result in zero gradients or multiplying very large parameters will result in infinity gradient respectively). ResNet was introduced to solve this issue so that we can train very deep neural network.

In this will be skip some connections which allow us to take the activation from one layer and suddenly feed it into another layer even much deeper neural network.

It uses a block called residual block which is given in following figure 30 and 31:

Residual Block :

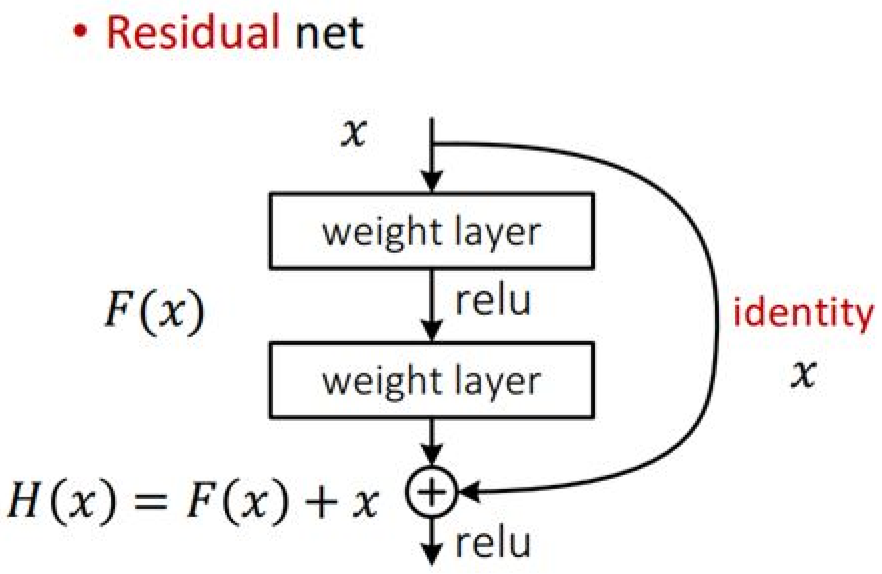


Figure 30

which can be shown as below :

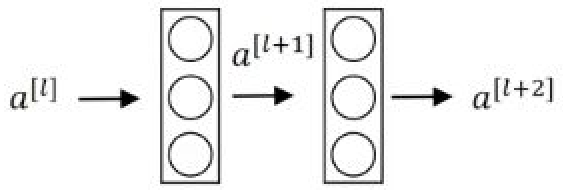


Figure 31

In which, we have a[l] =x; input for activation layer l, and a[l+2]=H(x) ;activation for layer l+2 , but we will give input to activation layer l+2 as a summation of a[l] and activation of x as F(x) i.e. H(x)=F(x)+x.

So that if somehow F(x) is zero, then H(x)=x ;as identity, and problem of vanishing gradient will be solved.

So we can obtain ResNet as a combination of such blocks in a deep neural network as below figure 32;

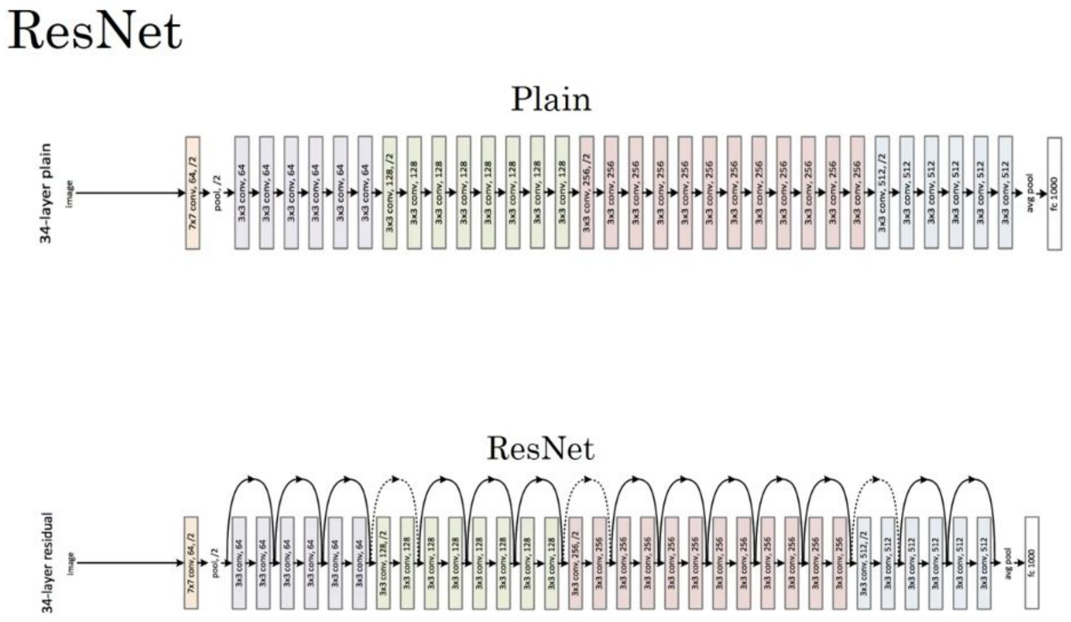


Figure 32

Which can give us below intuition for learning : We can see the importance of ResNet from the below graph (Figure 33). That's why for training deep Neural Network we should use ResNet for better performance.

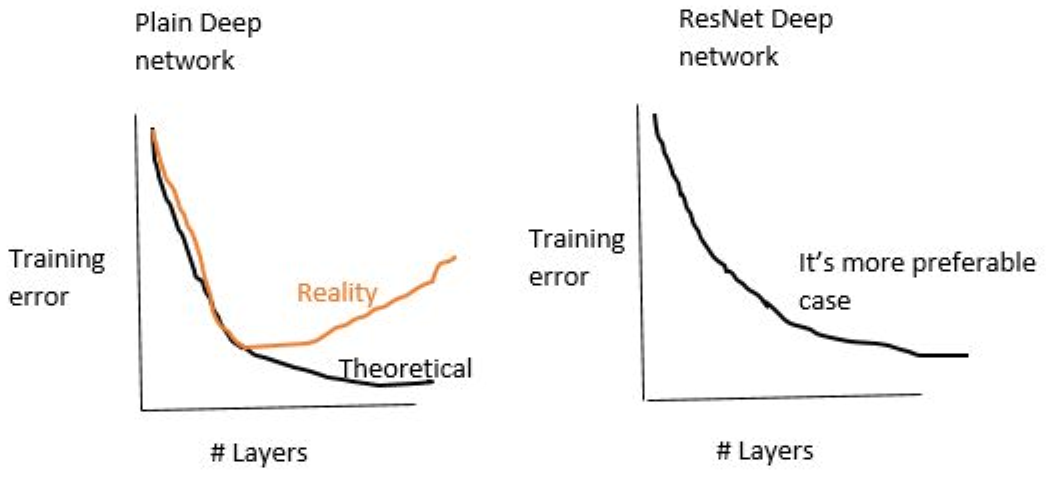


Figure 33

Since RetinaNet is using a very deep neural network that's why ResNet is used to train the network.

2 .Feature Pyramid network [11]

Now second block to understand is Feature Pyramid Network :

Earlier we were detecting objects in different scales for small objects, which was really challenging task. For this task we can make a pyramid of the same image at different scales as shown in Figure 34 .a .

Even though this was also time consuming to process all multiple scale images and needs more memory to be trained end-to-end simultaneously. Therefore, then pyramid of feature maps were introduced as shown in figure 34 b. In which, feature maps getting after each convolution were used for detection. But as we know, feature maps in lower layers, i.e. near to input images, composed of low-level features of the images and that can't give accurate object detection.

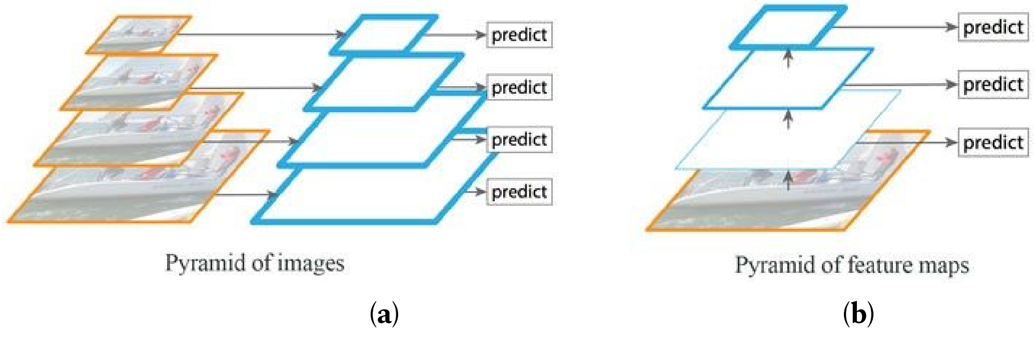


Figure 34

To resolve this problem Feature Pyramid Network (FPN) were introduced in which, most outer layer of the network, i.e. last feature map,

which is having a great feature quality, imposed on the lower level feature maps to get a better feature map at lower levels as well. This kind of feature extractor generates multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection.Which is shown in figure 35

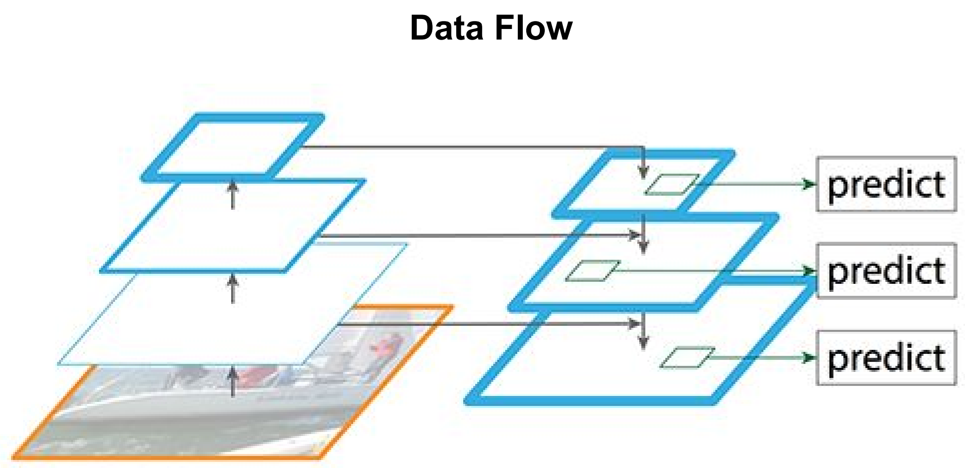


Figure 35

FPN composes of a bottom-up and a top-down pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the semantic value for each layer increases.

Keeping this quality of FPN in mind, this is used in Retinanet network.

3. Focal loss for dense object detection

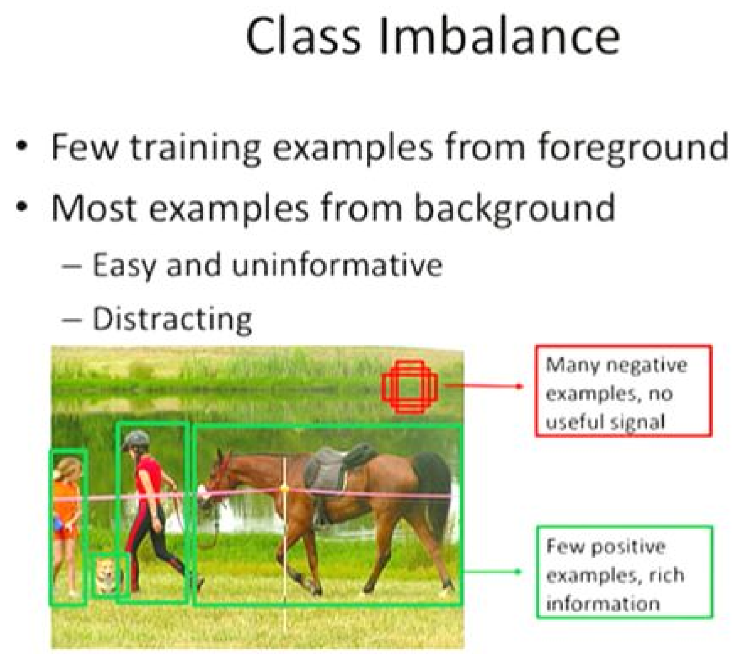
Third and last most important component of RetinaNet Object Detector is the new loss function defined by the FAIR group i.e. Focal Loss.
Let's have a look for this loss.

Focal Loss

Why do we need to define a new loss function for the object detection?

As we can see from below figure 36, there are more negative examples (background) in a given image than positive examples (which contains some object), its a huge class imbalance for the training of the network .

That is why one stage object detector (YOLO, SSD, etc.) were fast, but have trailed the accuracy of two-stage detectors because of extreme class imbalance encountered during training.



Class Imbalance

- Few training examples from foreground
- Most examples from background
  - Easy and uninformative
  - Distracting

Many negative examples, no useful signal
Few positive examples, rich information

Figure 36

Thus, the FAIR research group has defined a new loss function to deal with this problem.Focal loss is the reshaping of cross entropy loss, such that it's down-weights the loss assigned to well-classified examples. So like this the novel focal loss would be able to focus the training on a sparse set of hard examples rather than the vast number of negatives during training.

Let's have a look how it works and what does it mean.

Let's look at a binary cross entropy loss for single object classification which is defined as below: CE Loss

$$\begin{gathered} if\, y = 1 \\ CE(p, y) = -\log(p) \\ otherwise \\ CE(p, y) = -\log(1 - p) \end{gathered}$$

Binary cross entropy loss

$$\begin{gathered} if\, y = 1 \\ pt = p \\ otherwise \\ pt = 1 - p \\ CE(pt) = -\log(pt) \end{gathered}$$

Binary cross entropy loss

Therefore, to manage high class imbalance, we will add a weighting parameter; which is treated as hyper-parameter set of cross-validation, called as alpha.

$$CE(pt) = -\alpha_t * \log(pt)$$

Balancing parameters in Focal loss

As mentioned in the paper, easily classified negatives examples (background which is easily classified having classification probability more than 0.6) comprise the majority of the loss and dominate the gradient. While alpha balances the importance of positive/negative examples, it does not differentiate between easy/hard examples. So the FAIR group has reshaped the cross entropy function and introduced a new focal loss as mentioned below

$$FL(pt) = -(1 - p_t)^{\gamma} * \alpha_t * \log(pt)$$

Focal loss equation

Here, gamma is called the focusing parameter and alpha is called the balancing parameter.

Let's go deeper to understand why this loss is important in training.

Case 1: **Easy negative and positive examples (which are correctly classified)**

Let's consider that we have easily classified foreground object with p=0.9.Then cross entropy loss will be

CE(foreground) = -log(0.9) = 0.1053

Now, consider easily classified background object with p=0.1. So, cross entropy loss

CE(background) = -log(1–0.1) = 0.1053

Now, lets calculate the focal loss for both the cases above. We will use alpha=0.25 and gamma = 2 (given by the author)

FL(foreground) = -1 x 0.25 x (1–0.9)2 log(0.9) = 0.00026**

FL(background) = -1 x 0.25 x (1–(1–0.1))2 log(1–0.1) = 0.00026.**

Case 2: **Hard Examples (misclassified example)**

Let's consider we have misclassified foreground object with p=0.1. Now cross entropy loss

CE(foreground) = -log(0.1) = 2.3025

Now, consider misclassified background object with p=0.9. Now cross entropy loss

CE(background) = -log(1–0.9) = 2.3025

Now, lets calculate the focal loss for both the cases above. We will use alpha=0.25 and gamma = 2

FL(foreground) = -1 x 0.25 x (1–0.1)2 log(0.1) = 0.4667**

FL(background) = -1 x 0.25 x (1–(1–0.9))2 log(1–0.9) = 0.4667**

Case 3: **Easy examples (Very easily classified)**

Say we have an easily classified foreground object with p=0.99. Now cross entropy loss

CE(foreground) = -log(0.99) = 0.01

Now, consider easily classified background object with p=0.01. Now cross entropy loss

CE(background) = -log(1–0.01) = 0.1053

Now, consider focal loss for both the cases above. We will use alpha=0.25 and gamma = 2

FL(foreground) = -1 x 0.25 x (1–0.99)2 log(0.99) = 2.5*10^(-7)**

FL(background) = -1 x 0.25 x (1–(1–0.01))2 log(1–0.01) = 2.5*10^(-7)**

Conclusion:

Case 1: 0.1/0.00026 = 384 times smaller number

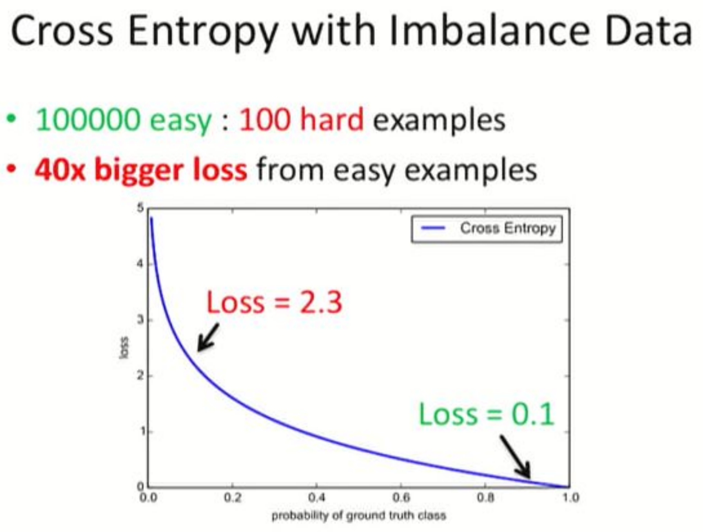
Case 2: 2.3/0.4667 = 5 times smaller number

Case 3: 0.01/0.00000025 = 40,000 times smaller number.

These three cases clearly show that Focal loss adds very less weight to well classified examples and large weight to miss-classified or hard classified examples.

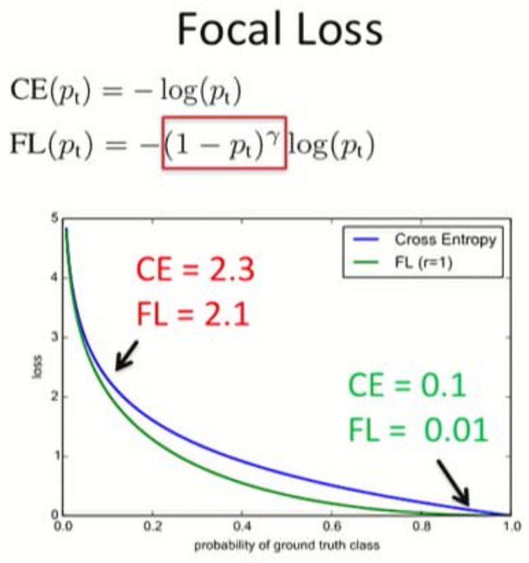
This is the basic intuition behind designing Focal loss. The authors have tested different values of alpha and gamma and finally settled with the above mentioned values.

This can be visualized with the help of following graph which was given in the paper as well : (Figure 37 and 38)


Cross Entropy with Imbalance Data

- 100000 easy : 100 hard examples
- **40x bigger loss** from easy examples

Figure 37


Focal Loss

$$\mathrm{CE}(p_t) = -\log(p_t)$$
$$\mathrm{FL}(p_t) = -\boxed{(1-p_t)^{\gamma}}\log(p_t)$$

Figure 38

So, like this we have described each important component of the RetinaNet architecture which was given in figure 29 .So the combination of these components makes Retinanet more robust, accurate and fast object detector which we have seen through the comparative graph in figure 30:

Concept of Anchor boxes is used as same as described in YOLO and SSD, but there are some Important points to note:

- When training for object detection, the focal loss is applied to all ~100k anchors in each sampled image.
- The total focal loss of an image is computed as the sum of the focal loss over all ~100k anchors, normalized by the number of anchors assigned to a ground-truth box.
- gamma =2 and alpha =0.25 works best and in general alpha should be decreased slightly as gamma is increased.

Inference on RetinaNet

- To improve speed, Decode box predictions from at most 1k top-scoring predictions per FPN level, after thresholding the detector confidence at 0.05.
- The top predictions from all levels are merged and non-maximum suppression with a threshold of 0.5 is applied to yield the final decisions.

https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d

Links to download the papers of above described Algorithms

Links to download all related paper is as below in decreasing order of the latest algorithms for object detection :

1. https://arxiv.org/pdf/1708.02002.pdf version2 (RetinaNet)
2. http://openaccess.thecvf.com/content_ICCV_2017/papers/Lin_Focal_Loss_for_ICCV_2017_paper.pdf version 1 (RetinaNet)

3. https://ieeexplore.ieee.org/document/8099589/ FPN paper
4. https://link.springer.com/chapter/10.1007%2F978-3-319-46448-0_2 SSD version5
5. https://www.cs.unc.edu/~wliu/papers/ssd.pdf SSD version 1
6. https://arxiv.org/abs/1512.02325 SSD paper at arxiv
7. https://pjreddie.com/media/files/papers/YOLOv3.pdf yolo version 3 paper
8. https://arxiv.org/pdf/1612.08242.pdf yolo version 2 paper
9. https://pjreddie.com/media/files/papers/yolo.pdf yolo version 1 paper
10. https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf faster RCNN Paper
11. https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf fast RCNN Paper
12. https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Liang_Recurrent_Convolutional_Neural_2015_CVPR_paper.pdf RCNN paper
13. https://pjreddie.com/darknet/yolo/ yolo tutorial

Links For object Detection Dataset

Here , I have given links to download the dataset for object detection, which can be used for training of the network.

1. http://host.robots.ox.ac.uk/pascal/VOC/voc2007/ PASCAL VOC dataset (with 20 classes)
2. http://cocodataset.org/#download Coco Dataset
3. https://motchallenge.net/data/2D_MOT_2015/ MOT Challenge Dataset

4. Robust Systems Lab Northeastern University,Boston, US, Person Re-identification Datasets
5. Udacity Dataset, specially for detecting objects for autonomous vehicle.

The dataset from MOT dataset, Udacity dataset and person re identification can be used to retrain the given network for the pedestrian dataset to make it more accurate for detecting pedestrian since this can be used in the application of computer vision such as , Video surveillance , autonomous vehicle driving ETC.

In fact, I have trained the given network for the dataset called Udacity dataset which is having roughly 20,000 images in total and 5 object categories.

I have trained the network, which was available online on github.

Links for available code of the algorithms for RetinaNet and SSD

These are links to download the available codes for RetinaNet and SSD algorithm which is most popular algorithm for object detection now a days.

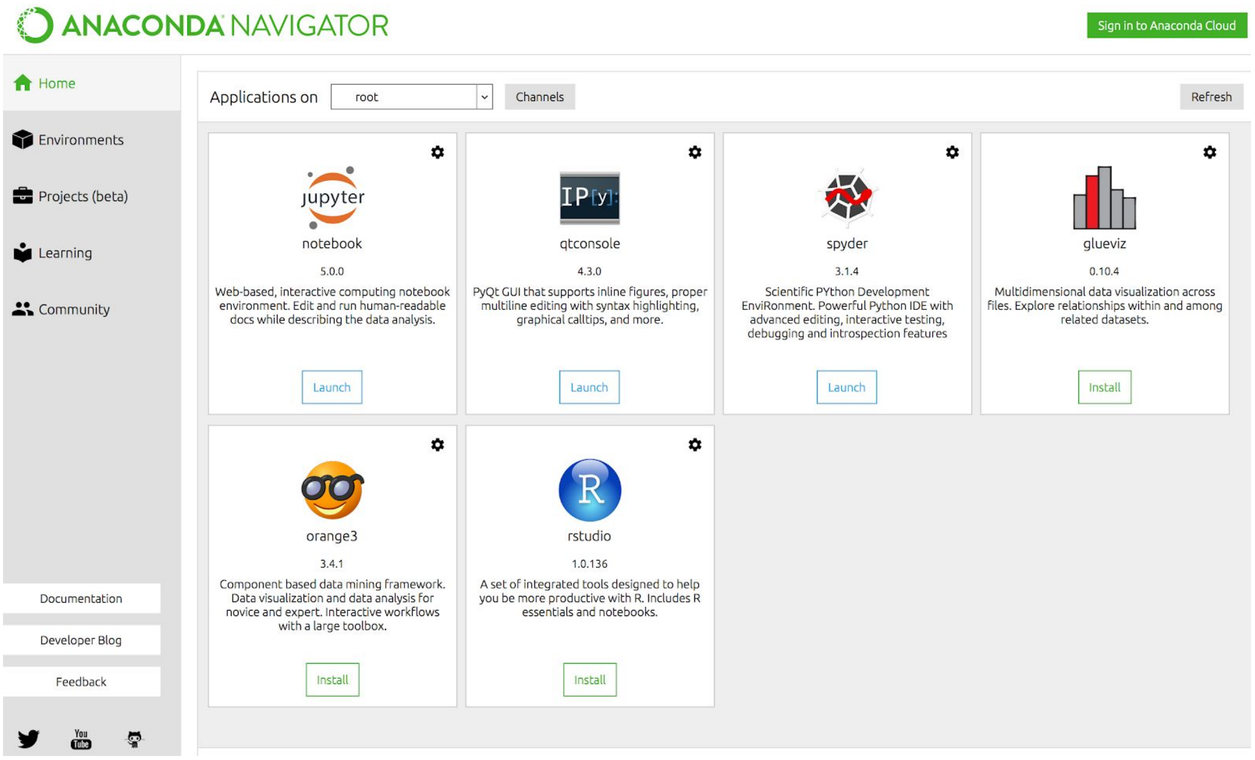
1. https://github.com/fizyr/keras-retinanet
2. https://github.com/pierluigiferrari/ssd_keras

These algorithms can be downloaded and can be retrained for your own dataset or can be tested with the help of given trained set of weights.

Links for the downloading the and installing the dependencies and framework for deep learning

We need to download some dependencies like Keras, Python, Tensorflow, NumPy and other frameworks to run the available codes. So here I am giving some links to download these frameworks and some tutorials to install it. Since all frameworks can be used with the help of Anaconda-navigator, so it's better to download and Install Anaconda-Navigator instead of downloading and installing all other dependencies.

1. https://anaconda.org/anaconda/anaconda-navigator For windows and linux both
2. https://anaconda.org/conda-forge/keras keras for windows and linux both
3. https://conda.io/docs/user-guide/install/index.html installation guide for installing conda on windows and as well as on linux.



notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
3.1.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz
0.10.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

rstudio
1.0.136
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 39

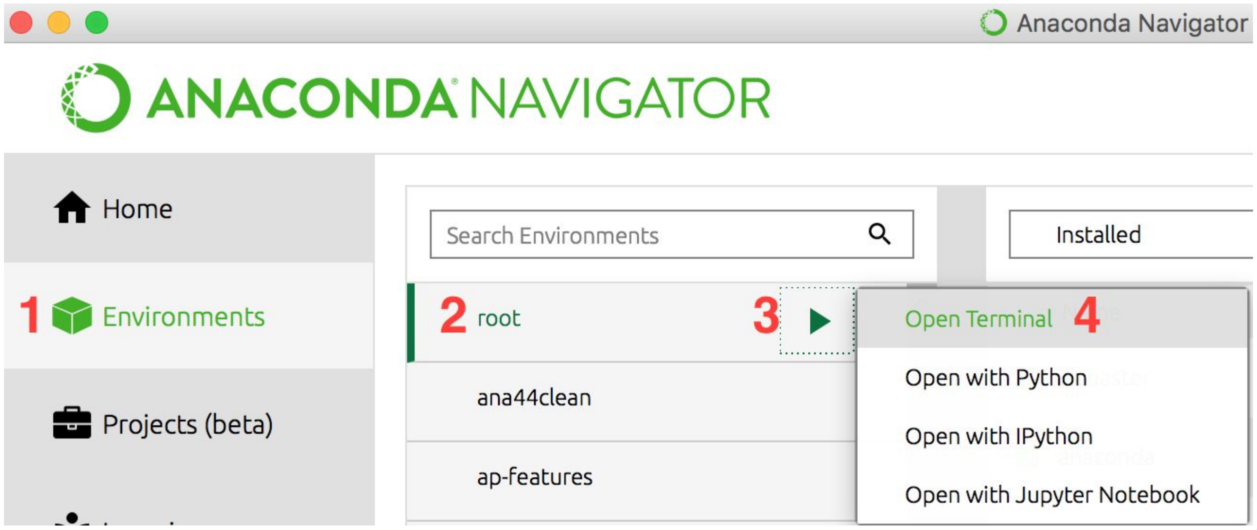


Figure 40

Links of Some Basic ideas for coding through Keras and Tensorflow as backend

Here, I can't describe all things about coding through Keras, so for getting some basic idea about coding you can follow up following links to learn coding. And you can also go through the basics of python of you don't have pre knowledge of it.

1. https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html This is for Learning Keras
2. https://www.python.org/about/gettingstarted/ This site is for learning basics of python

Results of the Algorithms RetinaNet and SSD

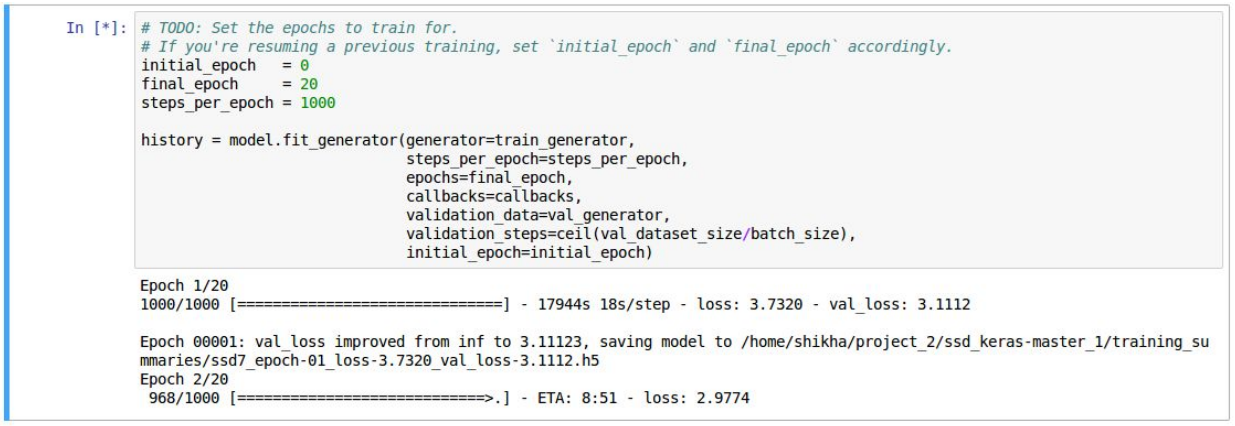
Problems and Solutions in training of RetinaNet

Retinanet has a very big architecture, so for training for our dataset we need a large amount of space in our system, so before train the available code first checkes your memory space and GPU size.

Solution :

If you don't have enough space in your system, then you can try to train SSD architecture with Focal loss from with the help of SSD available code in Keras, as I have provided the link for the code.

In Fact, I have also trained the SSD+Focal loss for the Udacity dataset, and SSD with Udacity dataset which you can check through the following figure 41 and 42.



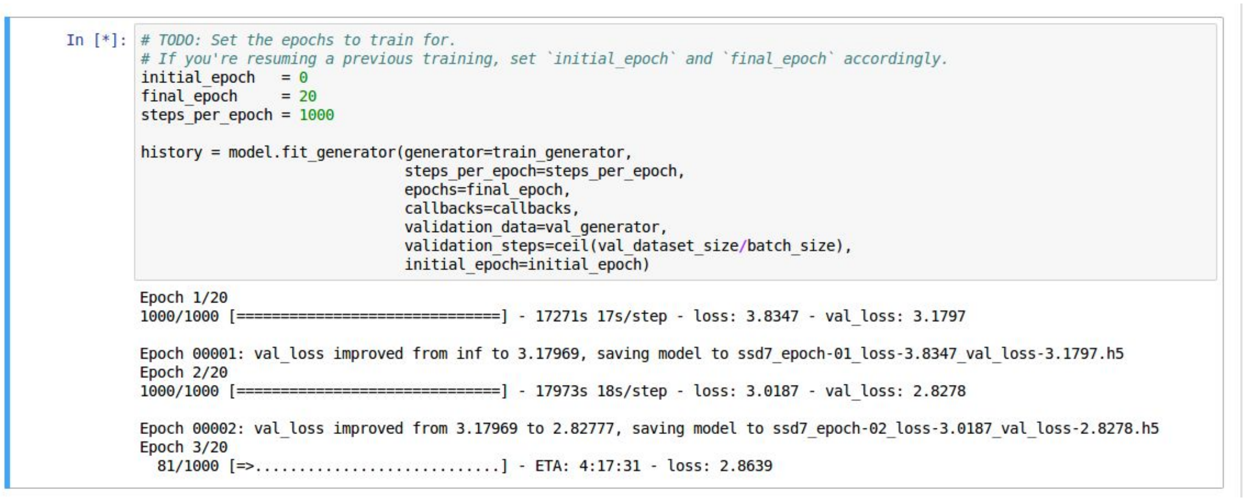
```
In [*]: # TODO: Set the epochs to train for.
        # If you're resuming a previous training, set `initial_epoch` and `final_epoch` accordingly.
        initial_epoch   = 0
        final_epoch     = 20
        steps_per_epoch = 1000

        history = model.fit_generator(generator=train_generator,
                                      steps_per_epoch=steps_per_epoch,
                                      epochs=final_epoch,
                                      callbacks=callbacks,
                                      validation_data=val_generator,
                                      validation_steps=ceil(val_dataset_size/batch_size),
                                      initial_epoch=initial_epoch)

        Epoch 1/20
        1000/1000 [==============================] - 17944s 18s/step - loss: 3.7320 - val_loss: 3.1112

        Epoch 00001: val_loss improved from inf to 3.11123, saving model to /home/shikha/project_2/ssd_keras-master_1/training_su
        mmaries/ssd7_epoch-01_loss-3.7320_val_loss-3.1112.h5
        Epoch 2/20
         968/1000 [============================>.] - ETA: 8:51 - loss: 2.9774
```

Figure 41



```
In [*]: # TODO: Set the epochs to train for.
        # If you're resuming a previous training, set `initial_epoch` and `final_epoch` accordingly.
        initial_epoch   = 0
        final_epoch     = 20
        steps_per_epoch = 1000

        history = model.fit_generator(generator=train_generator,
                                      steps_per_epoch=steps_per_epoch,
                                      epochs=final_epoch,
                                      callbacks=callbacks,
                                      validation_data=val_generator,
                                      validation_steps=ceil(val_dataset_size/batch_size),
                                      initial_epoch=initial_epoch)

        Epoch 1/20
        1000/1000 [==============================] - 17271s 17s/step - loss: 3.8347 - val_loss: 3.1797

        Epoch 00001: val_loss improved from inf to 3.17969, saving model to ssd7_epoch-01_loss-3.8347_val_loss-3.1797.h5
        Epoch 2/20
        1000/1000 [==============================] - 17973s 18s/step - loss: 3.0187 - val_loss: 2.8278

        Epoch 00002: val_loss improved from 3.17969 to 2.82777, saving model to ssd7_epoch-02_loss-3.0187_val_loss-2.8278.h5
        Epoch 3/20
          81/1000 [=>............................] - ETA: 4:17:31 - loss: 2.8639
```

Figure 42

Since, SSD uses VGG16 as a backend network, which is less dense than ResNet, so it can be trained on your system. But use focal loss instead of Cross Entropy loss as described in the paper of RetinaNet to get more accurate results of the detection.

In the next section, I will show some results with the help of training of these networks.

The new results after training the SSD and SSD + focal loss for udacity dataset

In this section, I will show some new results, which we got from the training the SSD with udacity dataset and SSD plus focal loss with udacity dataset.

I just train the network for 20 iterations with 1000 steps per epoch, visualization for training is as below :

Training graph for SSD with udacity dataset

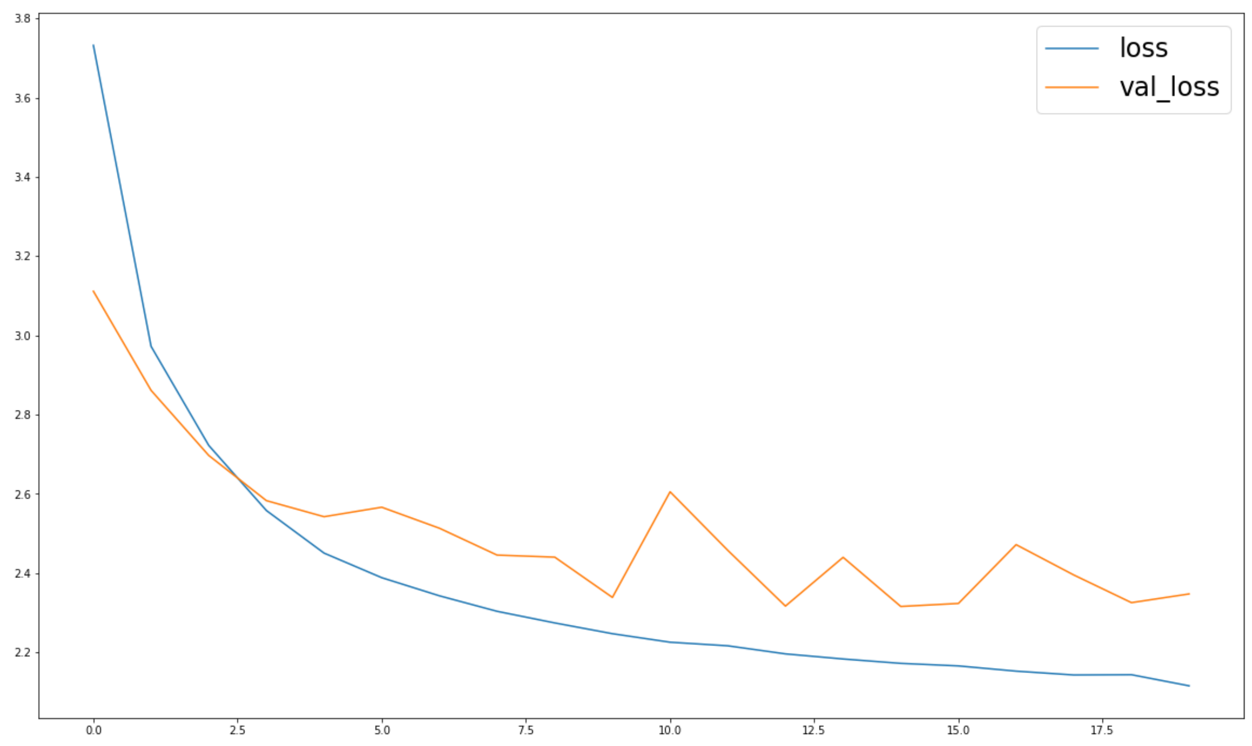


Figure 43

Testing result of the trained SSD with udacity dataset

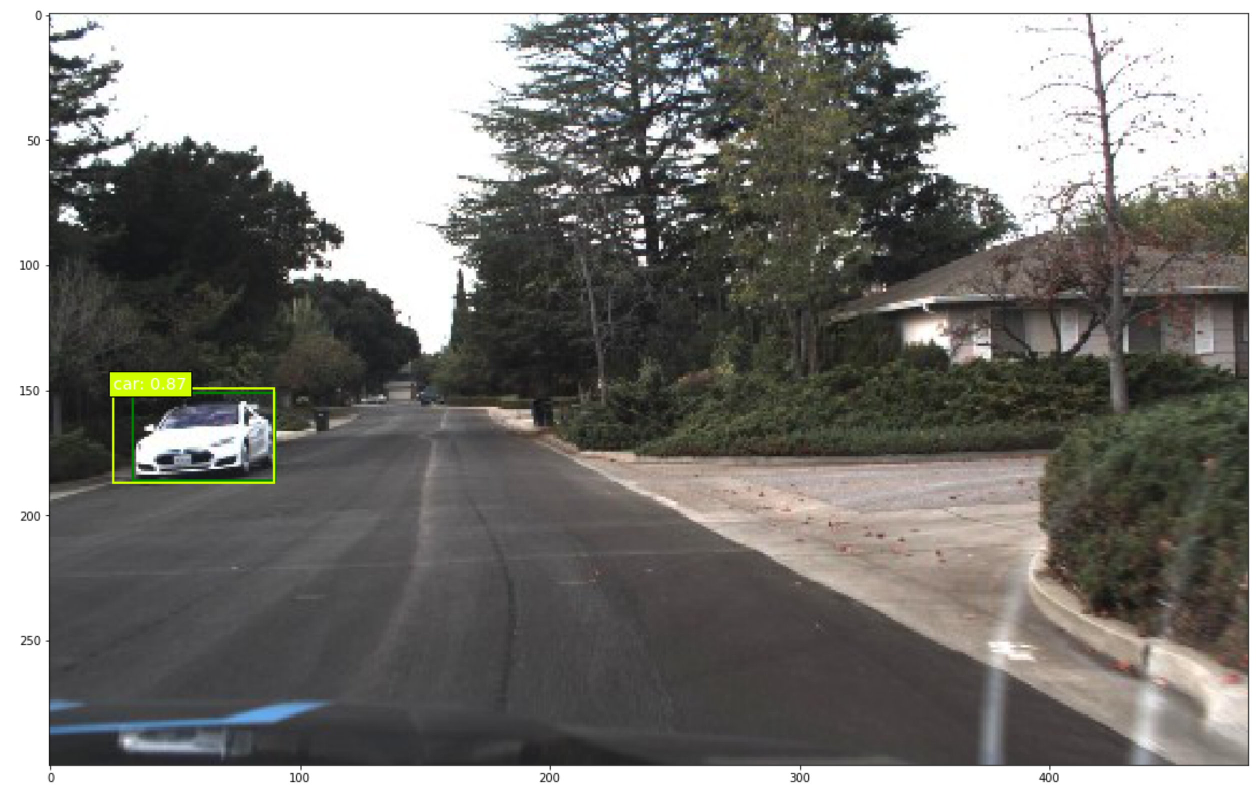


Figure 44

Training graph for SSD + focal loss with udacity dataset

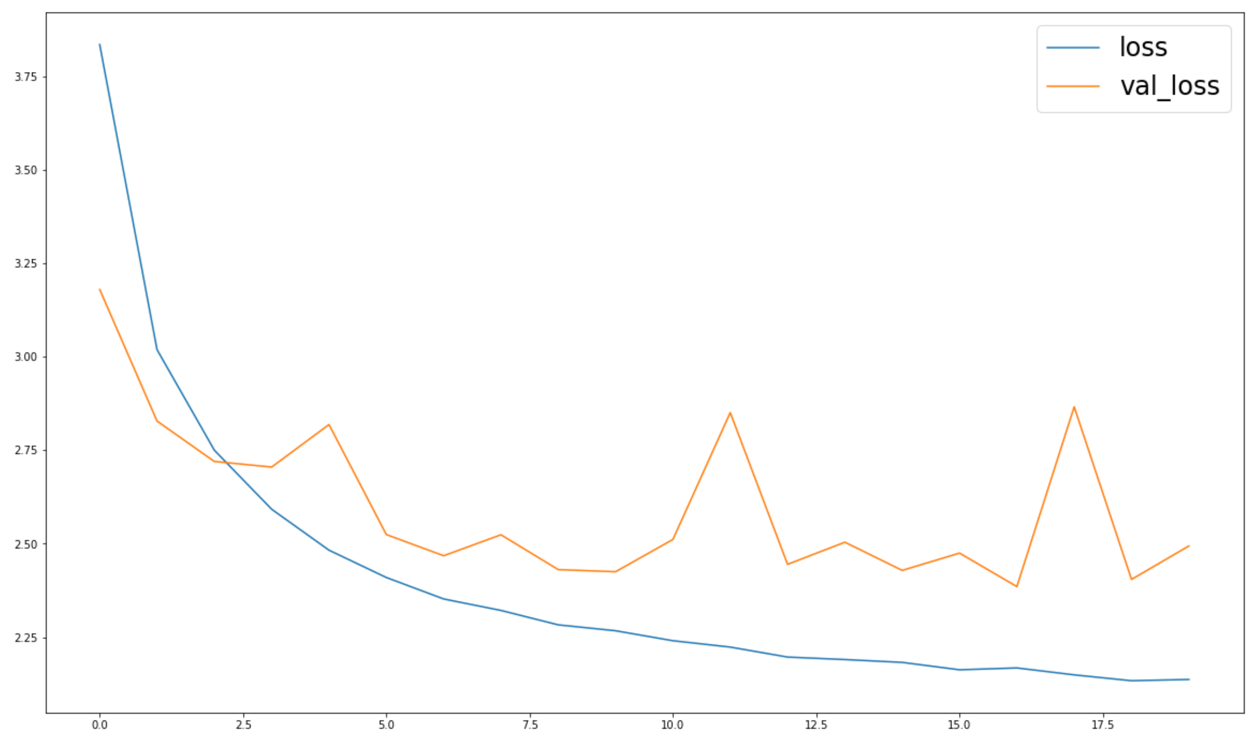


Figure 45

Testing result of the trained SSD +focal loss with udacity dataset

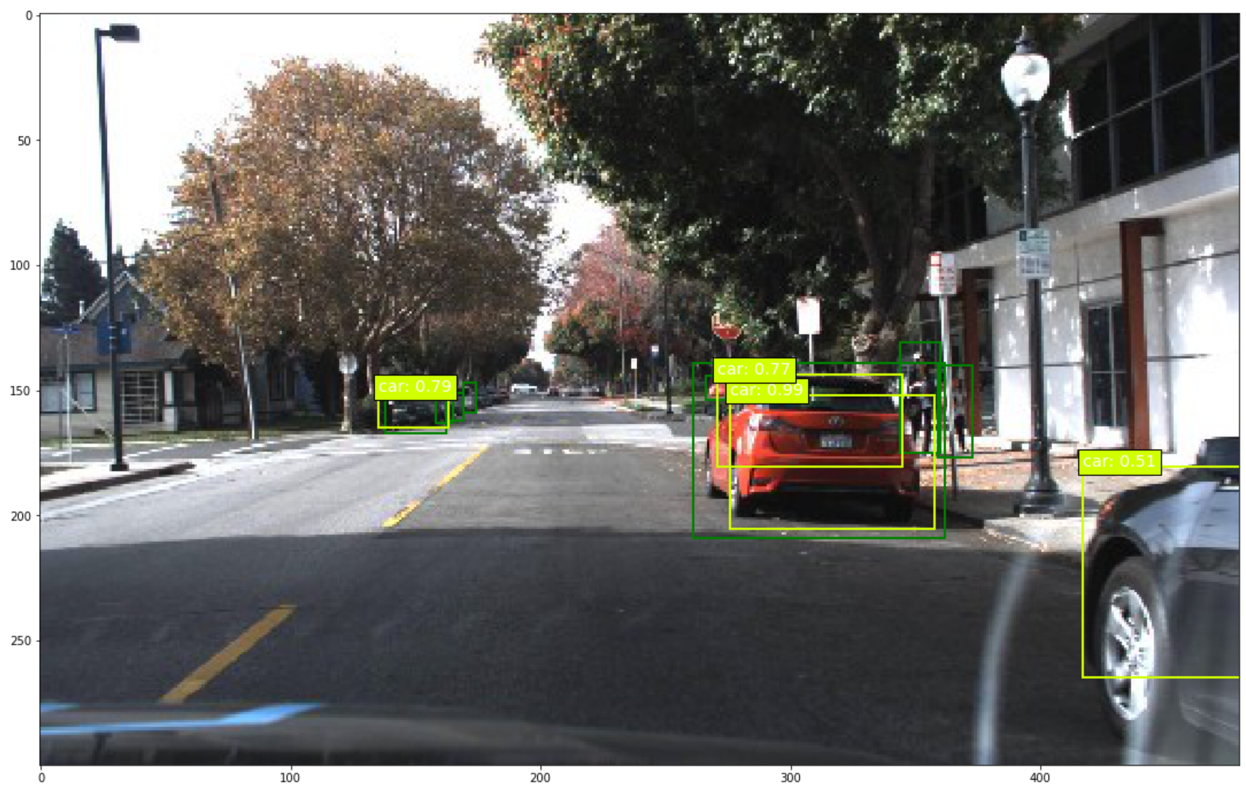
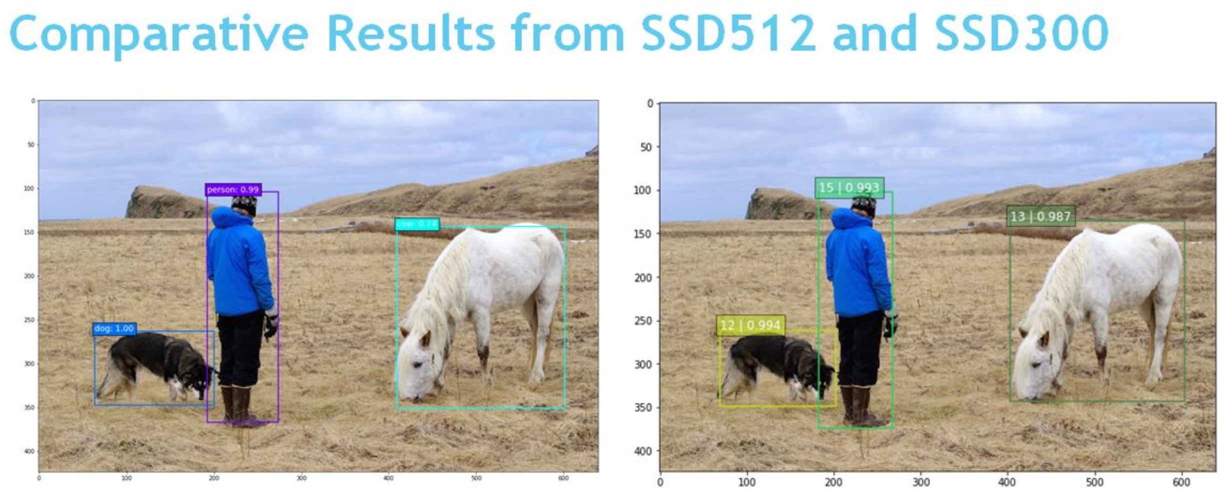


Figure 46

As, we can see the above figures, from figure 43 and 45 that training for just 20 epochs is not giving good validation error, especially in case of SSD+Focal loss, we can't compare exactly the results and graph obtained from such less number of epochs. We need to train algorithm at least 50 times for getting better results. Even though, our assumption that SSD+Focal loss will work better than Cross Entropy loss for validation set can't be proved in such small number of epochs. But, testing results from figure 44 and 46, are giving good results for detection, where green color is a true value, and yellow green color for the detection result of the trained model.

Comparative results of SSD512 and SSD 300 after testing with the provided trained weights.

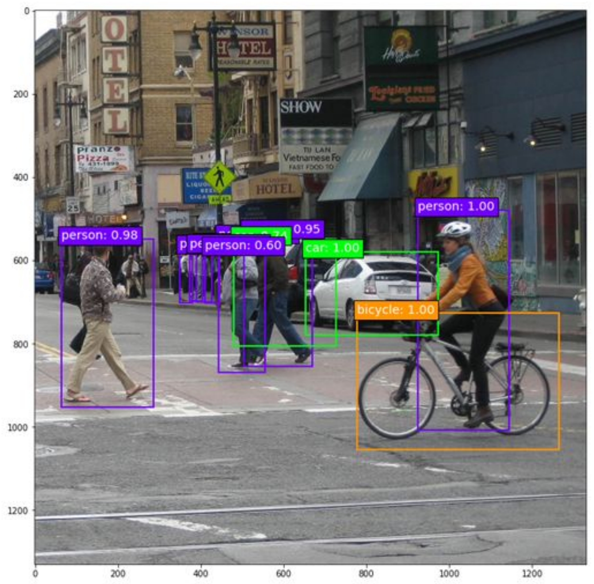
Here,SSD 512 and SSD 300 stands for the size of the input image which is given to the SSD algorithm. We have just tested the SSD 512 and 300 for available trained weights to get some intuition about the results of the algorithm.



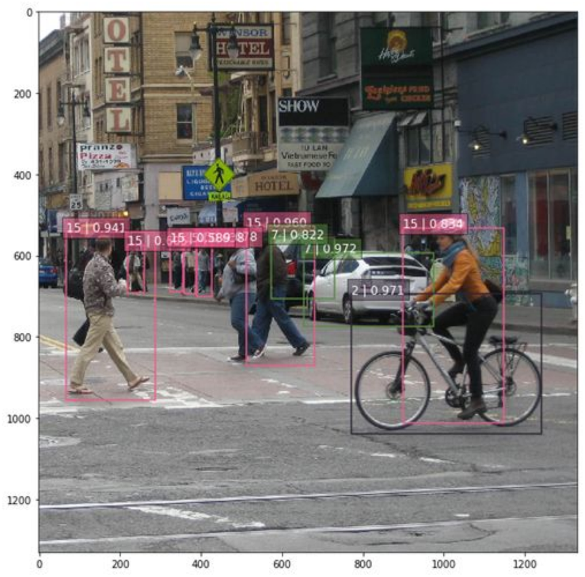
(a) **(b)**

Figure 47

For VOC DATASET 2007 (a) SSD 512 (b) SSD 300



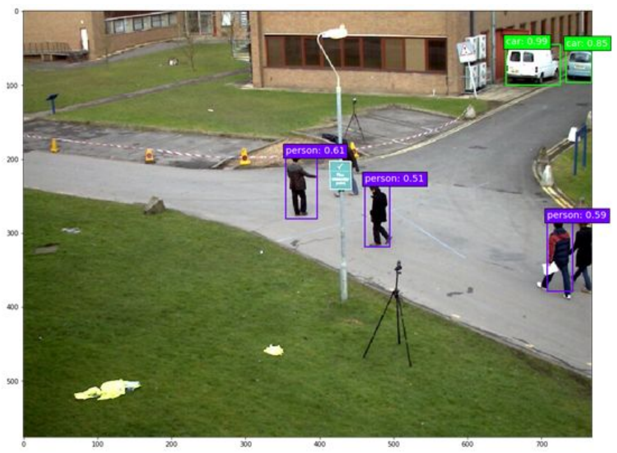
(a)



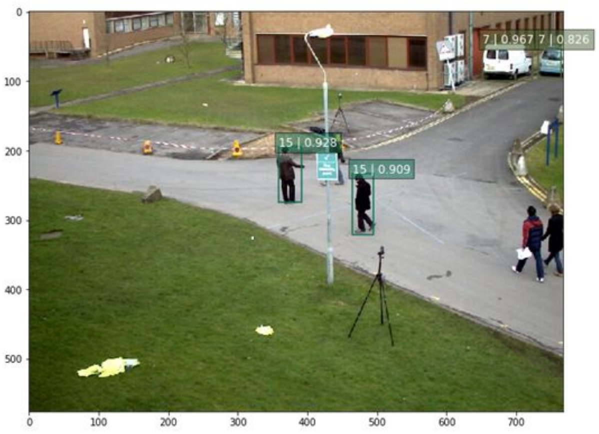
(b)

Figure 48

For VOC DATASET 2007 (a) SSD 512 (b) SSD 300



(a)



(b)

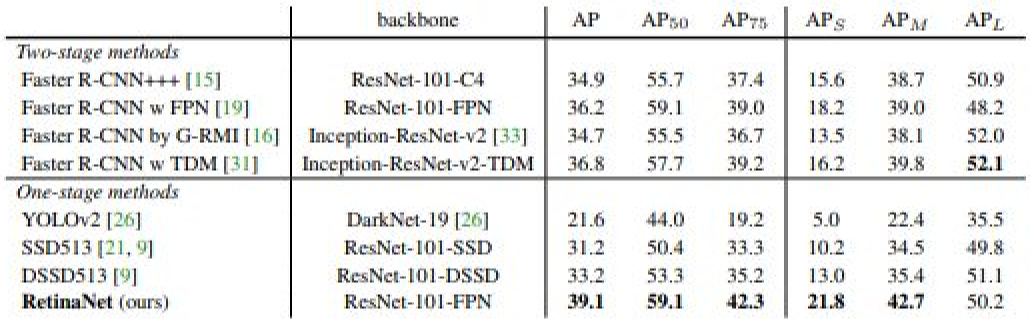
Figure 49

For MOT DATASET 2007 (a) SSD 512 (b) SSD 300

So, we can see if we will increase the size of the input image or increasing the resolution of the image will give us more accurate results. This is true for other algorithms as well. But it will be slower than the SSD 300 .

Conclusion

After comparing and learning all the recent algorithms we can say that RetinaNet is given good trade off in speed and accuracy as compare to other algorithm as we can see from the figure 28 which was given in the paper of the RetinaNet. But we can get more intuition from the below table 2 , which was also given in the paper of the RetinaNet.



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [19]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [16]	Inception-ResNet-v2 [33]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [26]	DarkNet-19 [26]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [21, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	**39.1**	**59.1**	**42.3**	**21.8**	**42.7**	50.2

Table 2

So we can say that Retinanet is giving very good trade off between accuracy as compare to the other fastest algorithm like YOLO and accurate algorithm like faster RCNN.

Future SCOPE

As we can see from the above conclusion and form table 2 still there is more to do in the field of the object detection since, we still didn't get an algorithm which can give us accuracy greater than 50 AP. So, from the tutorial we learned the basic concepts of all the latest object detection algorithm, so after getting the intuitive idea you all also can contribute in this field of computer vision. One idea can be like training the Object Detection algorithm with YOLO plus focal loss, we can check the accuracy of this detector and for other ideas we have to think further in this area and need to give more time to develop such kind of accurate and fast algorithm.

References

1. Katarzyna Janocha 1 , Wojciech Marian Czarnecki "*On Loss Functions for Deep Neural Networks in Classification".arXiv:1702.05659v1 [cs.LG] 18 Feb 2017*
2. R. Girshick, J. Donahue, T. Darrell, and J. Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation.* In CVPR, 2014.
3. Ross Girshick, *Fast R-CNN , ICCV 2015,* Microsoft Research.
4. S. Ren, K. He, R. Girshick, and J. Sun. *Faster R-CNN: Towards real-time object detection with region proposal networks.* In NIPS, 2015.
5. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. *You only look once: Unified, real-time object detection*. In CVPR, 2016.
6. J. Redmon and A. Farhadi. *YOLO9000: Better, faster, stronger*. In CVPR, 2017.
7. J. Redmon and A. Farhadi. *Yolov3: An incremental improvement*. arXiv, 2018.
8. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. *SSD: Single shot multibox detector.* In ECCV, 2016.
9. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar. *Focal loss for dense object detection*. arXiv preprint, arXiv:1708.02002v2, 2018.
10. K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In CVPR, 2016.
11. T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. *Feature pyramid networks for object detection.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017
12. R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollar, ´ and K. He. Detectron. *https://github.com/ facebookresearch/detectron,* 2018.

- **From Author**

Hopefully, you all have enjoyed the tutorial and got some intuitive idea of the object detection algorithms of deep learning . I hope this tutorial was helpful in your studies. So that you can contribute in the future work of this interesting and trending application of computer vision. Still more research is needed in this field.

For any query and suggestions contact me through mail. My email IDs are shikhad.bhu@gmail.com and shikha.d@gist.ac.kr

About Me

Shikha Dubey
Phd Student
Machine Learning and Vision Laboratory
School of Electrical Engineering and Computer Science
GIST, Rep. of Korea
Research area : Digital Image & Speech Processing, Machine Learning & Computer Vision

Gwangju Institute of Science and Technology

Thank You