

An Internship Report on

# **3D Object Detection for Autonomous Driving**

Submitted By:

**Maripi Srinuvasarao (E&C-122/16)**

Under the guidance of:

**Dr. S.N Omkar**  
**Chief Research Scientist**  
**Aerospace Engineering Dept.**  
**IISc Bangalore**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**



**NATIONAL INSTITUTE OF TECHNOLOGY, SRINAGAR**  
**Jammu & Kashmir, India-190006**

June 2019 - September 2019

## **DECLARATION**

I hereby declare that the report presented is carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree at this or any other institution and GitHub link to my repository is <https://github.com/srinu6/Stereo-3D-Object-Detection-for-Autonomous-Driving>

Maripi Srinuvasarao

## **ACKNOWLEDGEMENT**

I wish to express my earnest gratitude to Aerospace Engineering Department, CINT lab project guides, research students for their valuable guidance and inspiration throughout the training period.

Maripi Srinuvasarao

## **Content**

Chapter 1	<b>Introduction</b>
Chapter 2	<b>Convolutional Neural Network(CNN)</b>
Chapter 3	<b>R-CNN (Region Convolutional Neural Network)</b>
Chapter 4	<b>R-CNN Evolution</b>
Chapter 5	<b>Stereo R-CNN</b>
Chapter 6	<b>Results and Future Scope</b>

# Chapter 1

## Introduction

Machine Learning is an idea to learn from examples and experience, without being explicitly programmed. Instead of writing code, you feed data to the generic algorithm, and it builds logic based on the data given.

### Examples of Machine Learning:

There are many examples of machine learning. Here are a few examples of classification problems where the goal is to categorize objects into a fixed set of categories.

**Face detection:** Identify faces in images (or indicate if a face is present).

**Email filtering:** Classify emails into spam and not-spam.

**Medical diagnosis:** Diagnose a patient as a sufferer or non-sufferer of some disease.

**Weather prediction:** Predict, for instance, whether or not it will rain tomorrow.

### Need of Machine Learning:

Machine Learning is a field which is raised out of Artificial Intelligence (AI). Applying AI, we wanted to build better and intelligent machines. But except for few mere tasks such as finding the shortest path between point A and B, we were unable to program more complex and constantly evolving challenges. There was a realisation that the only way to be able to achieve this task was to let machine learn from itself. This sounds similar to a child learning from itself. So machine learning was developed as a new capability for computers. And now machine learning is present in so many segments of technology, that we don't even realise it while using it. Finding patterns in data on planet earth is possible only for human brains. The data being very massive, the time taken to compute is increased, and this is where Machine Learning comes into action, to help people with large data in minimum time. If big data and cloud computing are gaining importance for their contributions, machine learning as technology helps analyse those big chunks of data, easing the task of data scientists in an automated process and gaining equal importance and recognition. The techniques we use for data mining have been around for many years, but they were not effective as they did not have the competitive power to run the algorithms. If you run deep learning with access to better data, the output we get will lead to dramatic breakthroughs which is machine learning.

### Kinds of Machine Learning:

There are three kinds of Machine Learning Algorithms.

- a. Supervised Learning

- b. Unsupervised Learning
- c. Reinforcement Learning

### **Supervised Learning:**

A majority of practical machine learning uses supervised learning.

In supervised learning, the system tries to learn from the previous examples that are given. (On the other hand, in unsupervised learning, the system attempts to find the patterns directly from the example given. Speaking mathematically, supervised learning is where you have both input variables (x) and output variables(Y) and can use an algorithm to derive the mapping function from the input to the output. The mapping function is expressed as  $Y = f(X)$ ).

Supervised learning problems can be further divided into two parts, namely classification, and regression.

**Classification:** A classification problem is when the output variable is a category or a group, such as “black” or “white” or “spam” and “no spam”.

**Regression:** A regression problem is when the output variable is a real value, such as “Rupees” or “height.”

### **Unsupervised Learning:**

In unsupervised learning, the algorithms are left to themselves to discover interesting structures in the data. Mathematically, unsupervised learning is when you only have input data (X) and no corresponding output variables. This is called unsupervised learning because unlike supervised learning above, there are no given correct answers and the machine itself finds the answers. Unsupervised learning problems can be further divided into association and clustering problems.

**Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as “people that buy X also tend to buy Y”.

**Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour.

### **Reinforcement Learning:**

A computer program will interact with a dynamic environment in which it must perform a particular goal (such as playing a game with an opponent or driving a car). The program is provided feedback in terms of rewards and punishments as it navigates its problem space. Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it continuously trains itself using trial and error method. Refer to fig.2 for block diagram.

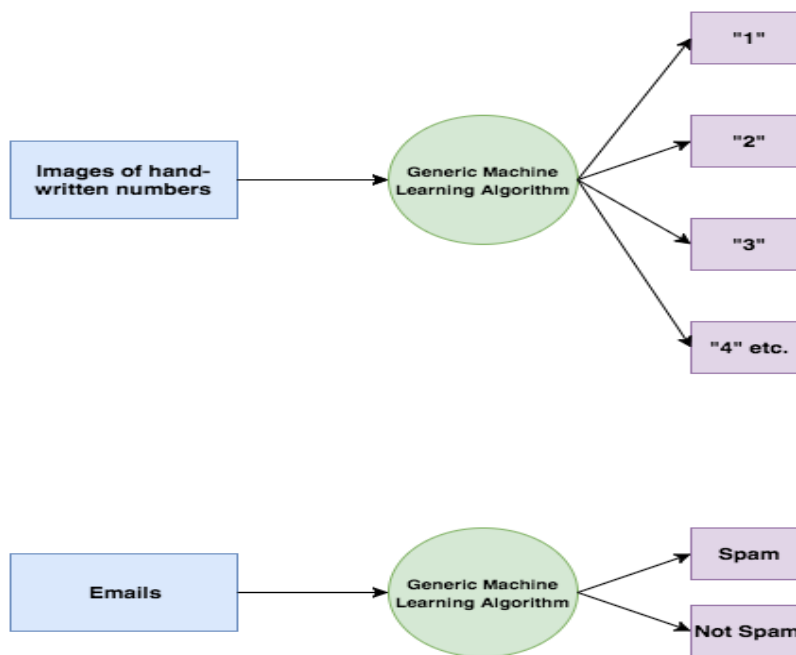


Figure 1 Working of Machine Learning

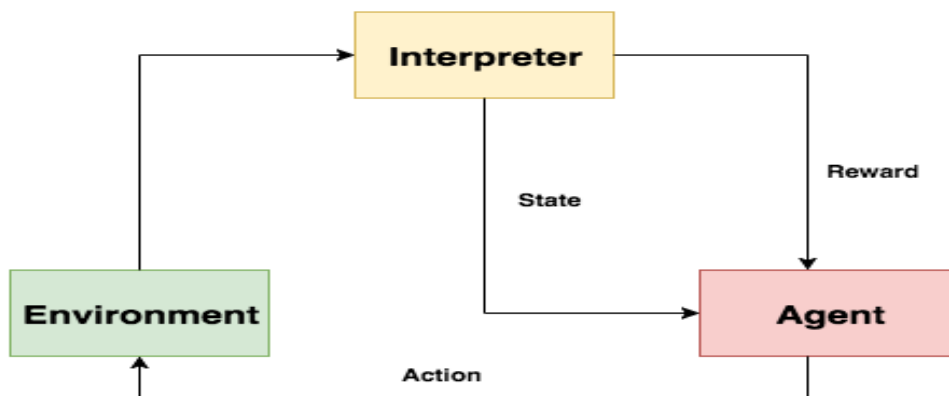


Figure 2 Reinforcement Learning

Here is the GitHub link to the PowerPoint Presentation file on Machine Learning and Deep Learning <https://github.com/srinu6/Machine-Learning-and-Deep-Learning-PPT> , me with my juniors prepared those slides on our own and presented those slides in Computational Intelligence Lab, Department of AeroSpace Engineering, IISc Bangalore. The content of those slides are extracted from various Learning sources like Coursera, Udemy, GeeksforGeeks.

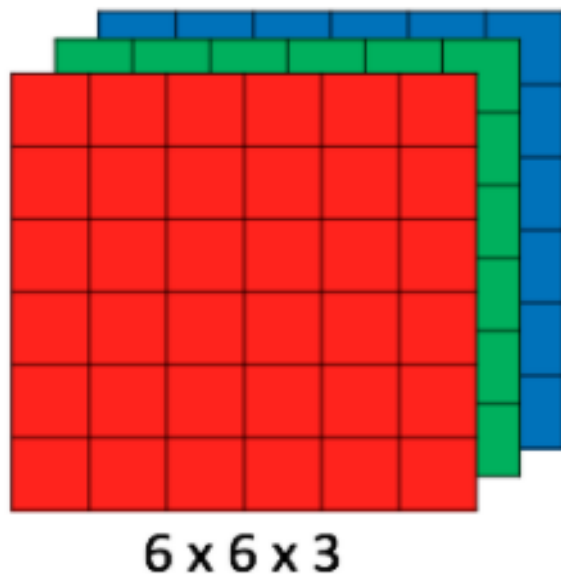
## Chapter 2

### Convolutional Neural Network

#### Understanding of Convolutional Neural Network (CNN) — Deep Learning

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

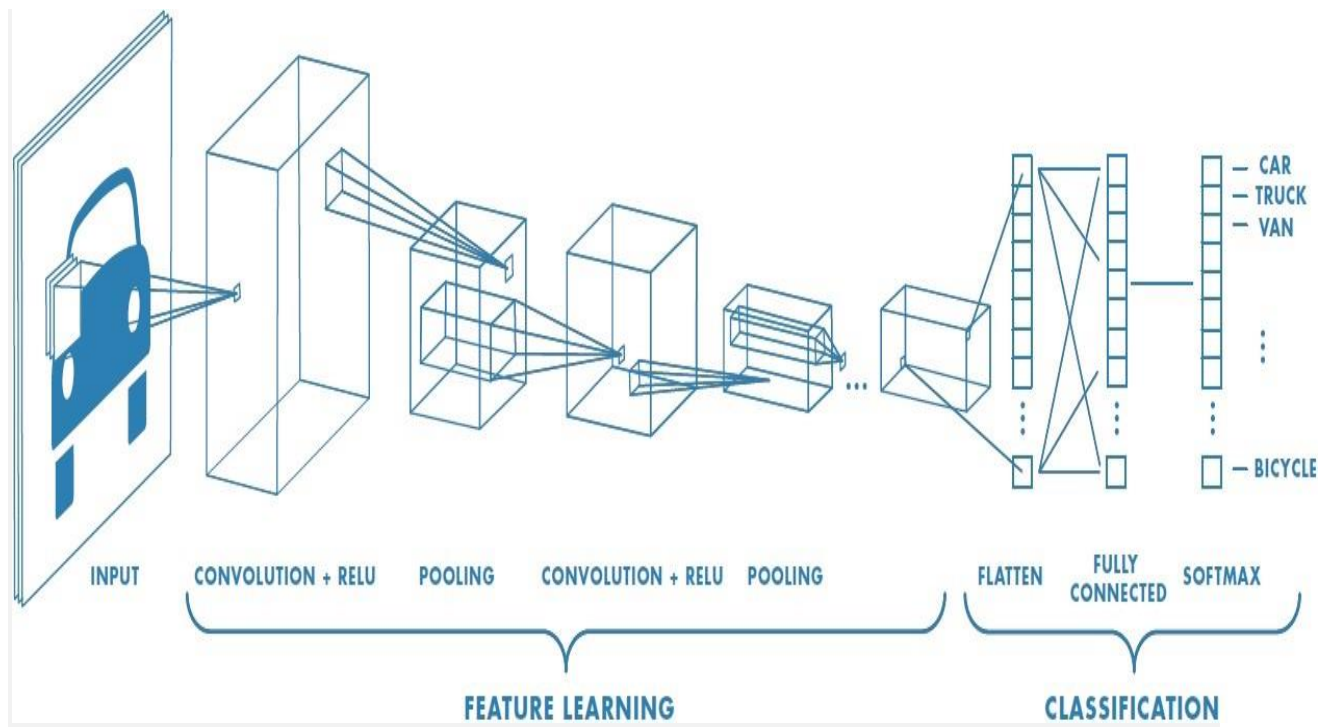
CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$  (  $h$  = Height,  $w$  = Width,  $d$  = Dimension ). Eg., An image of  $6 \times 6 \times 3$  array of matrix of RGB (3 refers to RGB values) and an image of  $4 \times 4 \times 1$  array of matrix of grayscale image.



**Figure 1 : Array of RGB Matrix**

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

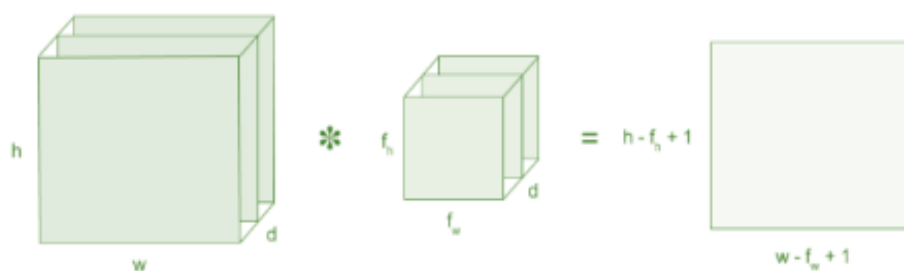




**Figure 2 : Neural network with many convolutional layers**

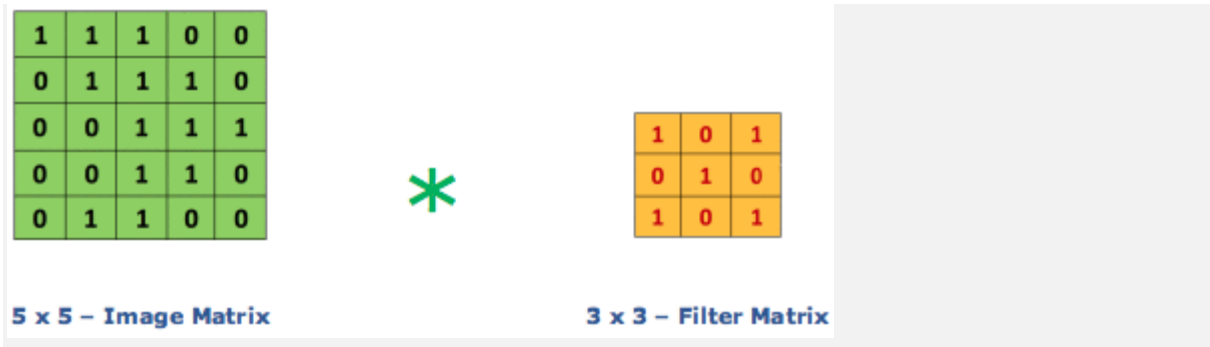
**Convolution Layer:** Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image and a filter or kernel matrix.

- An image matrix (volume) of dimension  **$(h \times w \times d)$**
- A filter  **$(f_h \times f_w \times d)$**
- Outputs a volume dimension  **$(h - f_h + 1) \times (w - f_w + 1) \times 1$**



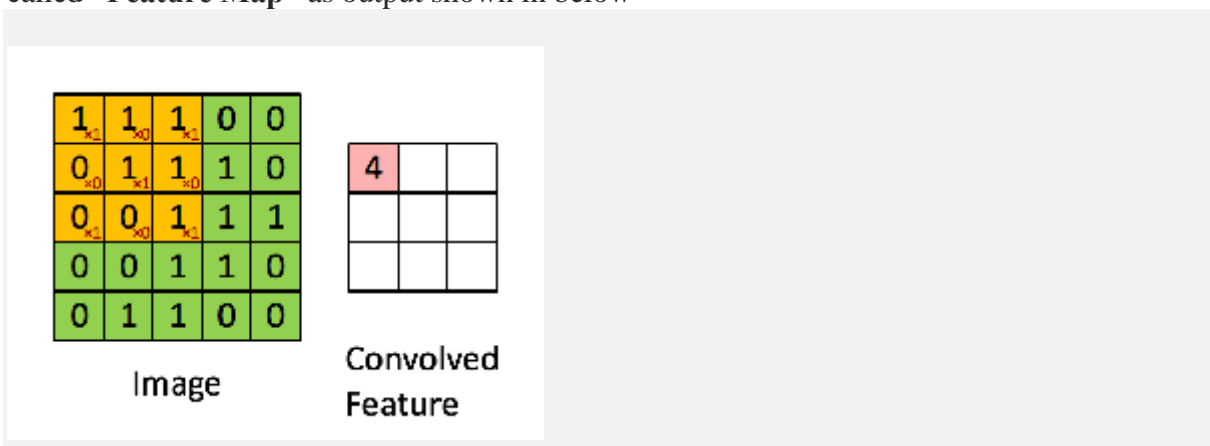
**Figure 3: Image matrix multiplies kernel or filter matrix**

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below



**Figure 4: Image matrix multiplies kernel or filter matrix**

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “**Feature Map**” as output shown in below



**Figure 5: 3 x 3 Output matrix**

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels).








Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 7 : Some common filters

## Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

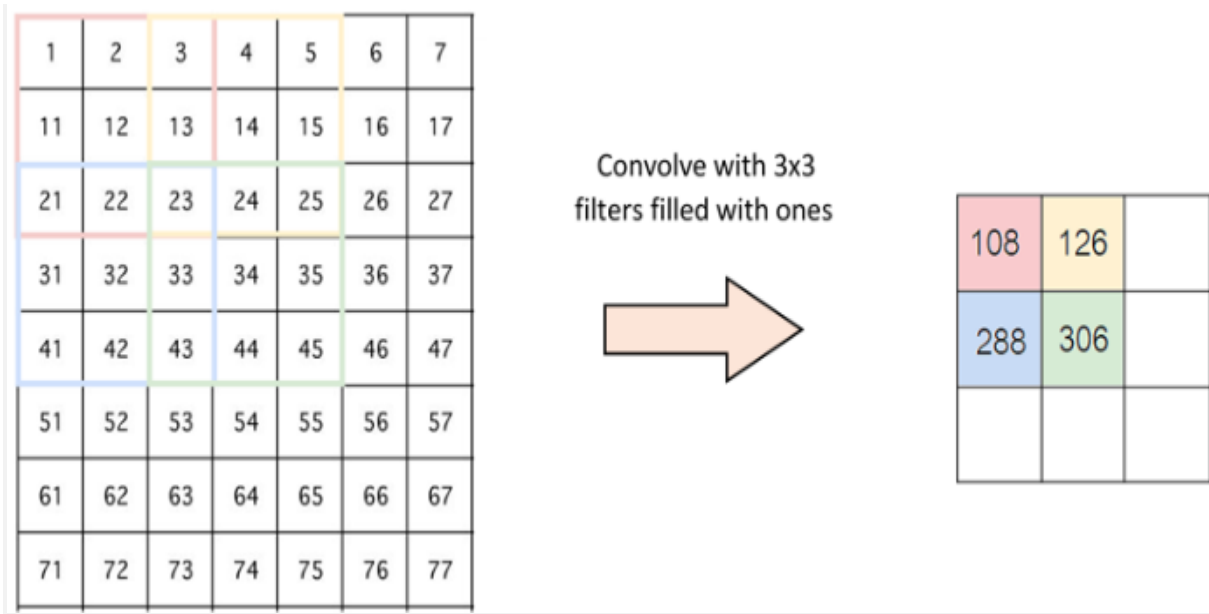


Figure 6 : Stride of 2 pixels

## Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

## Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is  $f(x) = \max(0, x)$ .

Why ReLU is important : ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

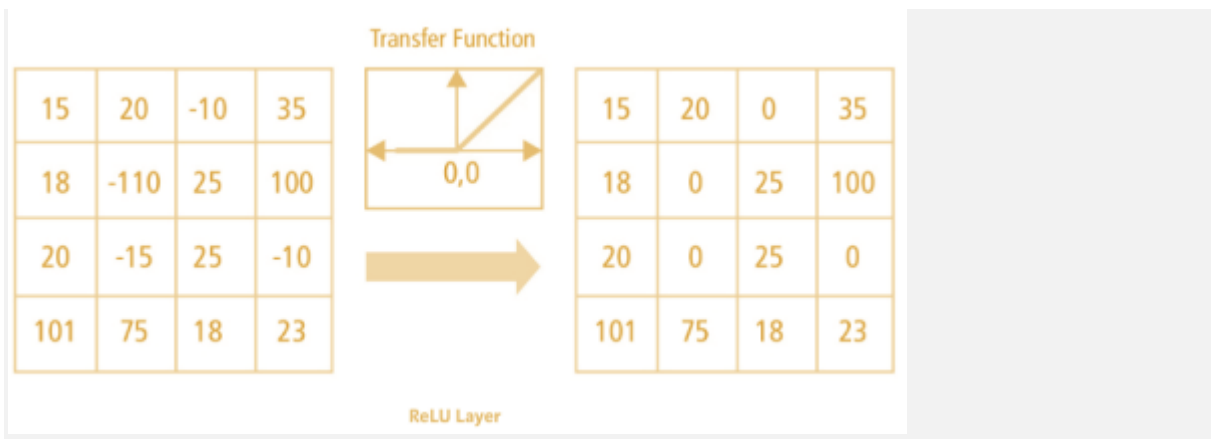


Figure 7 : ReLU operation

There are other non linear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two.

**Pooling Layer:** Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

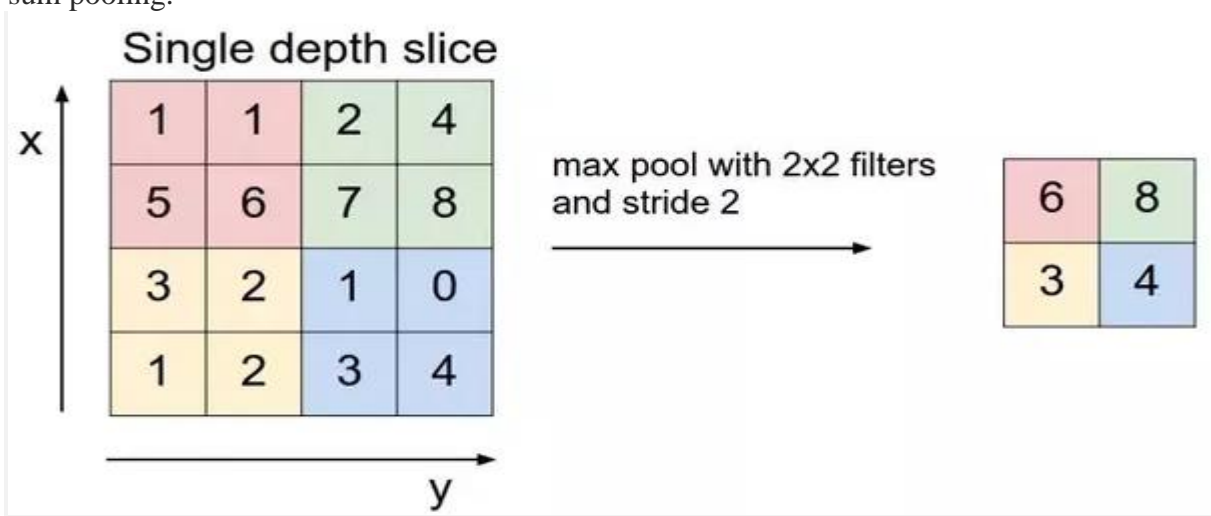


Figure 8 : Max Pooling

## Fully Connected Layer

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

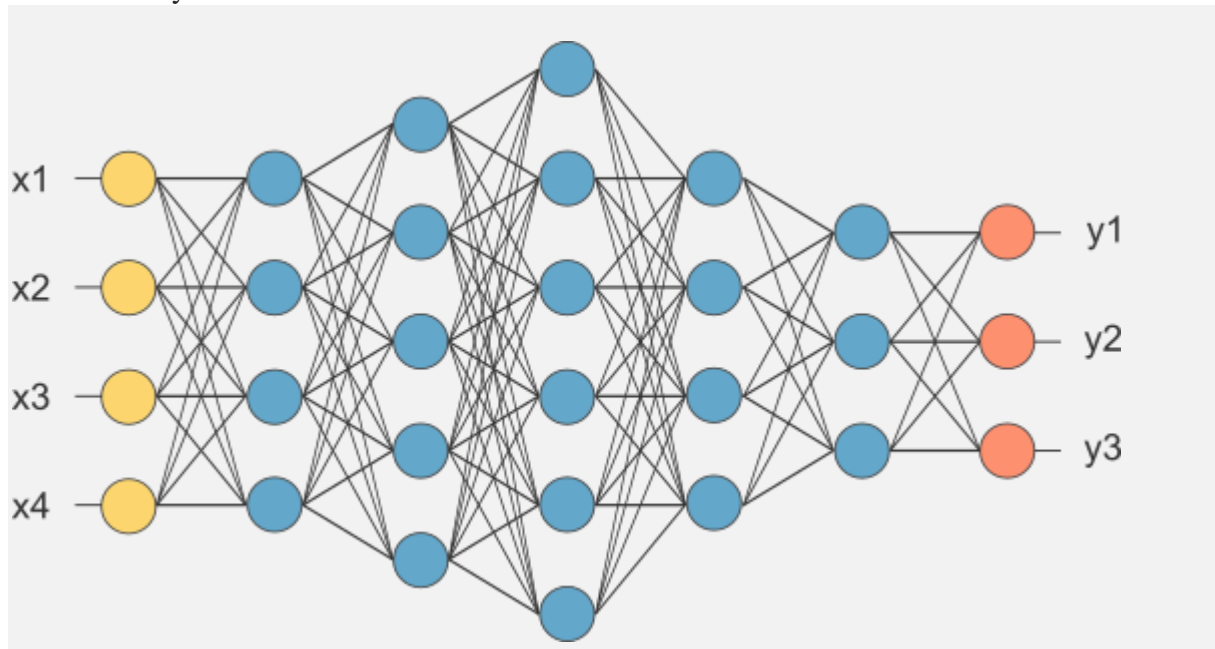


Figure 9 : After pooling layer, flattened as FC layer

In the above diagram, the feature map matrix will be converted as vector ( $x_1, x_2, x_3, \dots$ ). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

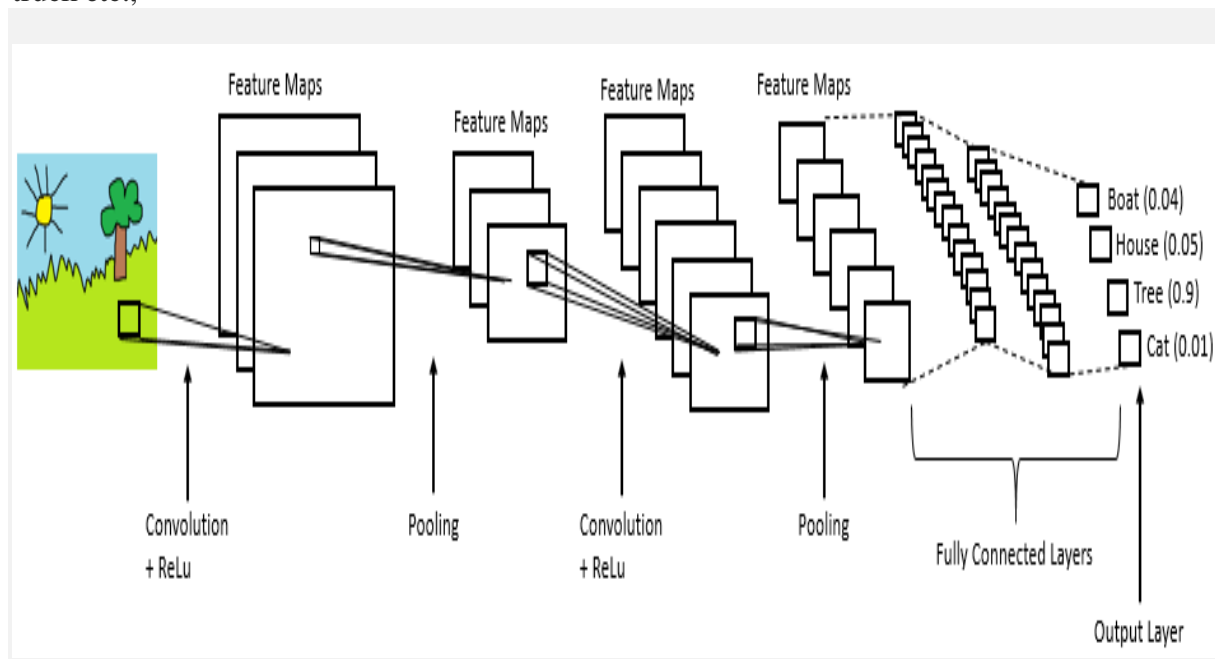


Figure 10 : Complete CNN architecture

## Chapter 3

### Region Convolutional Neural Network (R-CNN)

Region with Convolutional Neural Network (R-CNN) is proposed by Girshick et al. in 2013. It changed the object detection field fundamentally. By leveraging selective search, CNN and SVM, Girshick et al. achieved a very good result in VOC 2012.

Here we will introduce R-CNN while later chapters will cover Fast R-CNN, Faster R-CNN and Mask R-CNN that is introduced by Girshick and other team members as well. Besides, there are other objection detection approaches such as Single-Shot Object Detector (SSD) and You Only Look Once (YOLO).

The objective of image classification is classifying the category of the whole image. On the other hand, object detection includes not only classification but also localization which was treated as a regression problem by Girshick et al..

This story will discuss [R-CNN](#) (Girshick et al., 2013) and the following will be covered:

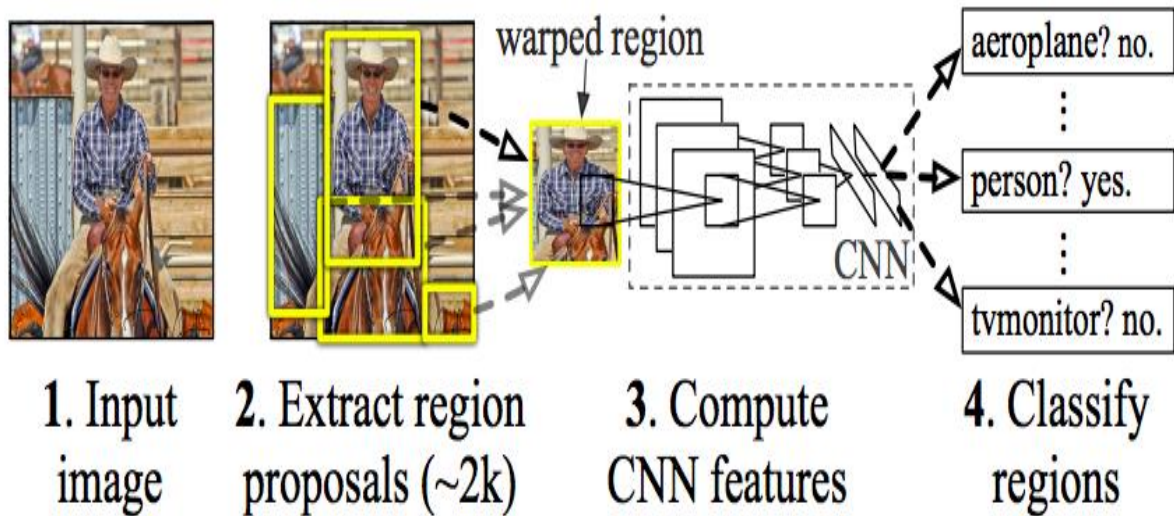
- Architecture of R-CNN
- Selective Search
- Feature Extraction
- Classification

#### Architecture of R-CNN

Given an image, R-CNN use selective search to generate around 2000 region proposals to compute features by using a convolutional neural network (CNN). Region proposals are regions that include the potential object. It will be wrapped as 227 x 227 RGB to fit into CNN. Feature extraction will be done in CNN layers and passing to multiple binary classifiers to figure out the class of particular regions.

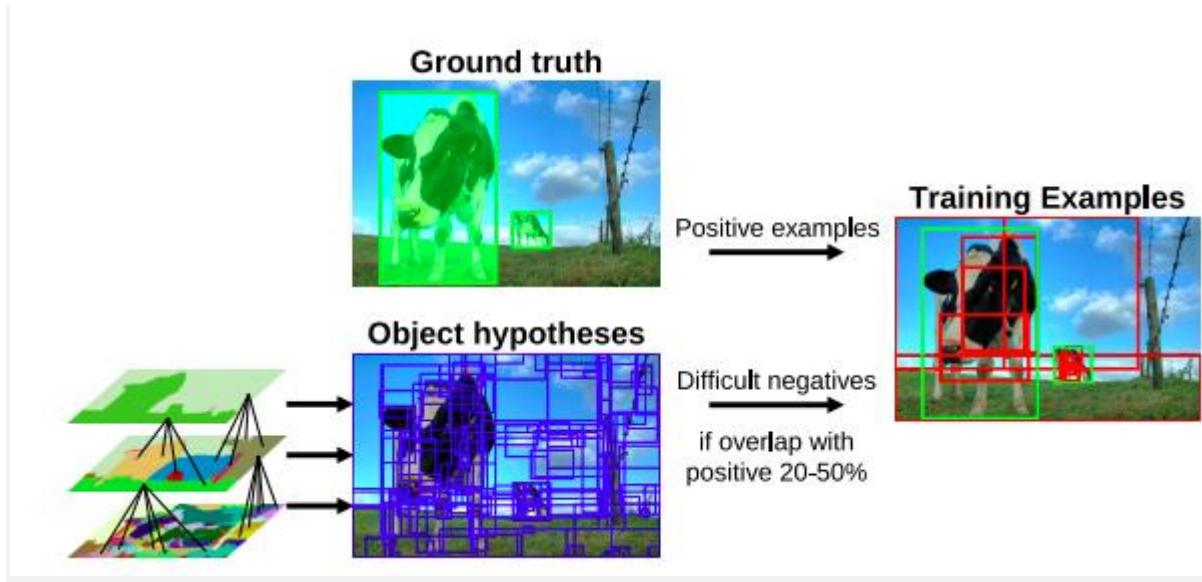


## R-CNN: *Regions with CNN features*



The architecture of R-CNN (Girshick et al., 2013)

## Selective Search



The architecture of Selective Search (Uijlings et al., 2012)

In R-CNN, selective Search approach is applied to find region proposals for classification. selective Search can capture any possible scales and less computational complexity. 2000 regions are selected from selective search.



The design considerations of selective search are:

- Capture all scales
- Diversification
- Fast to compute

To achieve that, the Hierarchical Grouping Algorithm is chosen to group those similar regions in a bottom-up approach. Feature for calculating similarity includes color, texture, region size, region filling,

#### Hierarchical Grouping Algorithm

It is greedy-search to find the region proposals. The procedures are:

- Obtain initialized regions by segment
- Calculating the similarity of neighboring regions
- Grouping similar region to the same bucket
- Coming to the most similar region within the same bucket. (Repeat this step)

---

**Algorithm 1: Hierarchical Grouping Algorithm**

---

**Input:** (colour) image

**Output:** Set of object location hypotheses  $L$

Obtain initial regions  $R = \{r_1, \dots, r_n\}$  using [13]

Initialise similarity set  $S = \emptyset$

**foreach** *Neighbouring region pair*  $(r_i, r_j)$  **do**

    Calculate similarity  $s(r_i, r_j)$

$S = S \cup s(r_i, r_j)$

**while**  $S \neq \emptyset$  **do**

    Get highest similarity  $s(r_i, r_j) = \max(S)$

    Merge corresponding regions  $r_t = r_i \cup r_j$

    Remove similarities regarding  $r_i : S = S \setminus s(r_i, r_*)$

    Remove similarities regarding  $r_j : S = S \setminus s(r_*, r_j)$

    Calculate similarity set  $S_t$  between  $r_t$  and its neighbours

$S = S \cup S_t$

$R = R \cup r_t$

Extract object location boxes  $L$  from all regions in  $R$

---

Hierarchical Grouping Algorithm (Uijlings et al., 2012)

## Diversification Strategies

Calculating different attributes to find the similarity. Possible attributes can be light intensity, shading, size, etc.

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),$$

Diversification Strategies Similarity Formula (Uijlings et al., 2012)

## Feature Extraction

CNN is chosen to perform the feature extraction. As mentioned before, selective search targets to capture all regions which imply that there are a different scale and ratio images. In order to fit into CNN, all regions are wrapped as a 227 x 227 RGB image. After that 4k dimensional feature via 5 convolutional layers and 2 fully connected layers.

## Classification

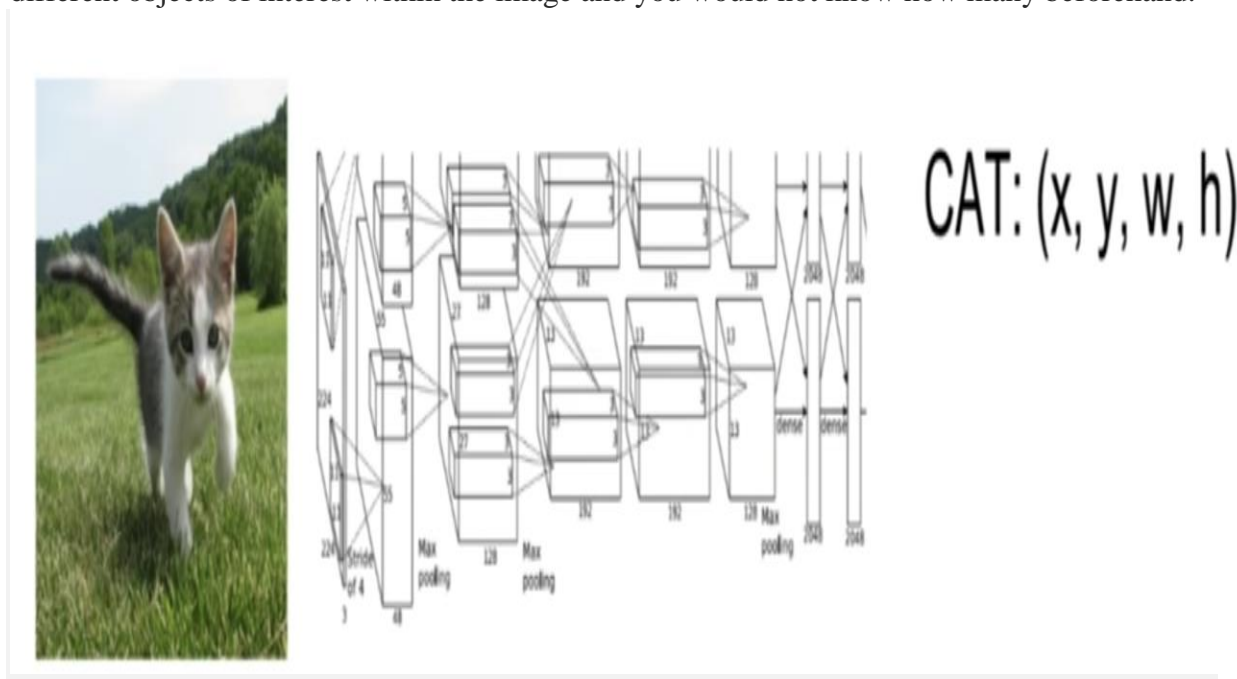
Every region is classified by multiple SVM binary classifiers. By applying greedy non-maximum suppression, a high intersection-over-union (IoU) overlap with a higher score region will be rejected for each class.

## Chapter 4

### R-CNN Evolution

#### R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms

Computer vision is an interdisciplinary field that has been gaining huge amounts of traction in the recent years (since CNN) and self-driving cars have taken center stage. Another integral part of computer vision is object detection. Object detection aids in pose estimation, vehicle detection, surveillance etc. The difference between object detection algorithms and classification algorithms is that in detection algorithms, we try to draw a bounding box around the object of interest to locate it within the image. Also, you might not necessarily draw just one bounding box in an object detection case, there could be many bounding boxes representing different objects of interest within the image and you would not know how many beforehand.



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

#### R-CNN

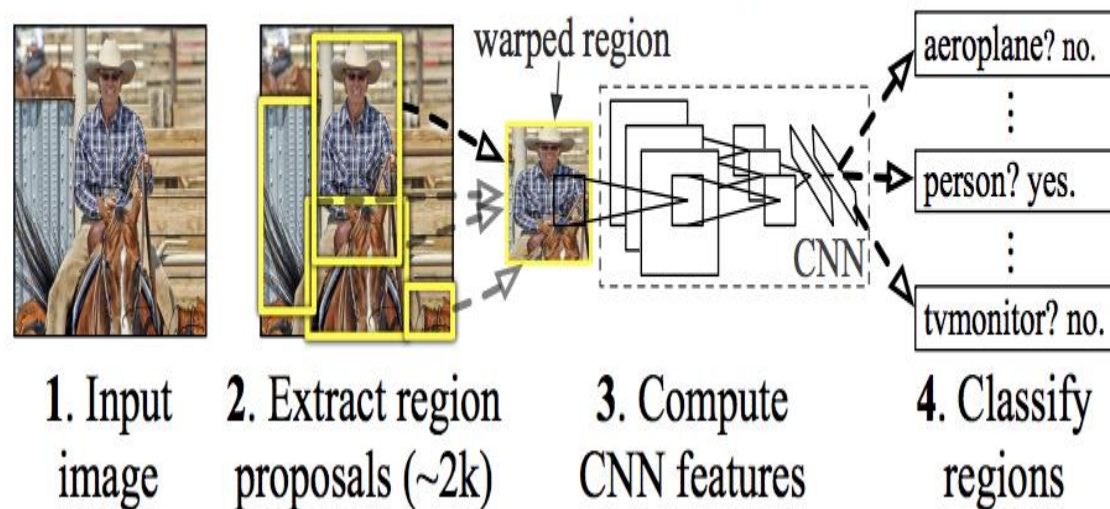
To bypass the problem of selecting a huge number of regions, [Ross Girshick et al](#) proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions,

you can just work with 2000 regions. These 2000 region proposals are generated using the selective search algorithm which is written below.

Selective Search:

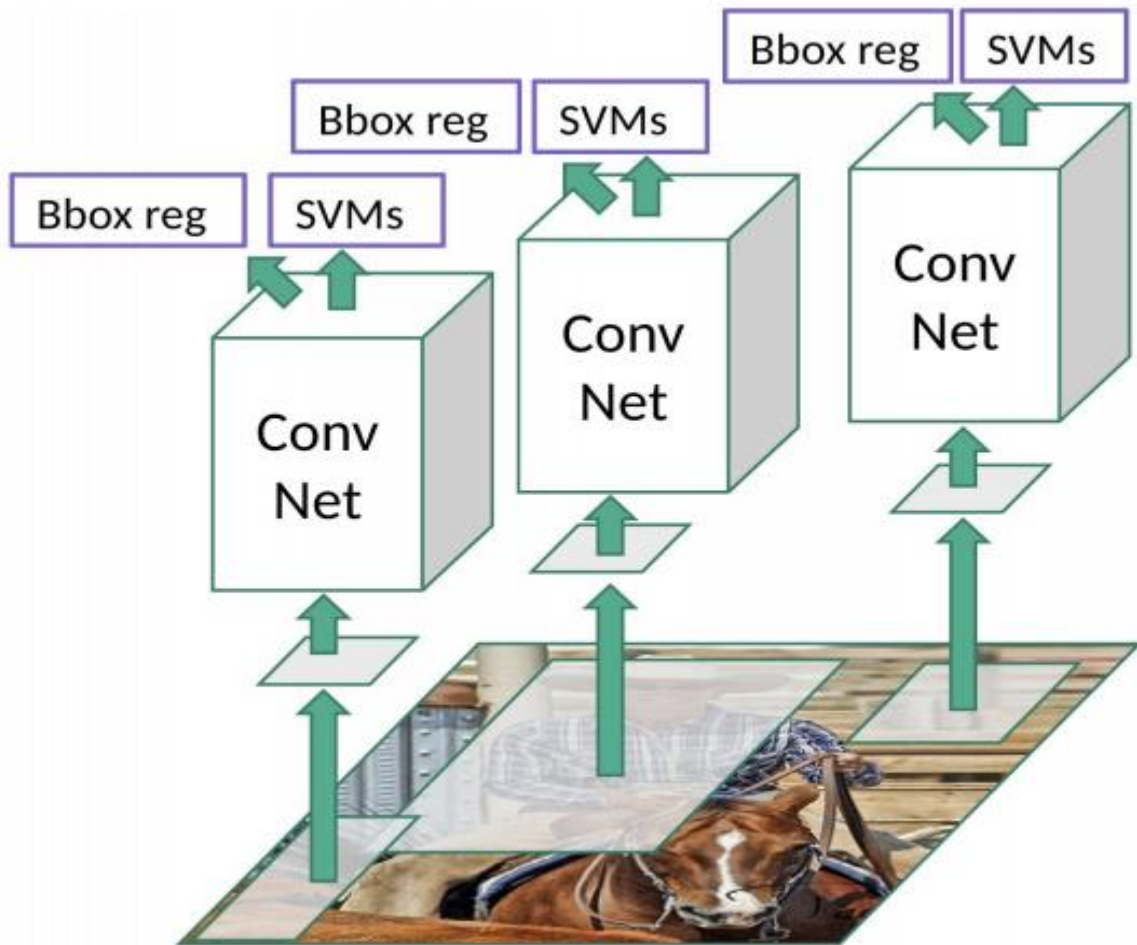
1. Generate initial sub-segmentation, we generate many candidate regions
2. Use greedy algorithm to recursively combine similar regions into larger ones
3. Use the generated regions to produce the final candidate region proposals

## R-CNN: *Regions with CNN features*



### R-CNN

To know more about the selective search algorithm, follow this [link](#). These 2000 candidate region proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an [SVM](#) to classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values to increase the precision of the bounding box. For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could've been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.



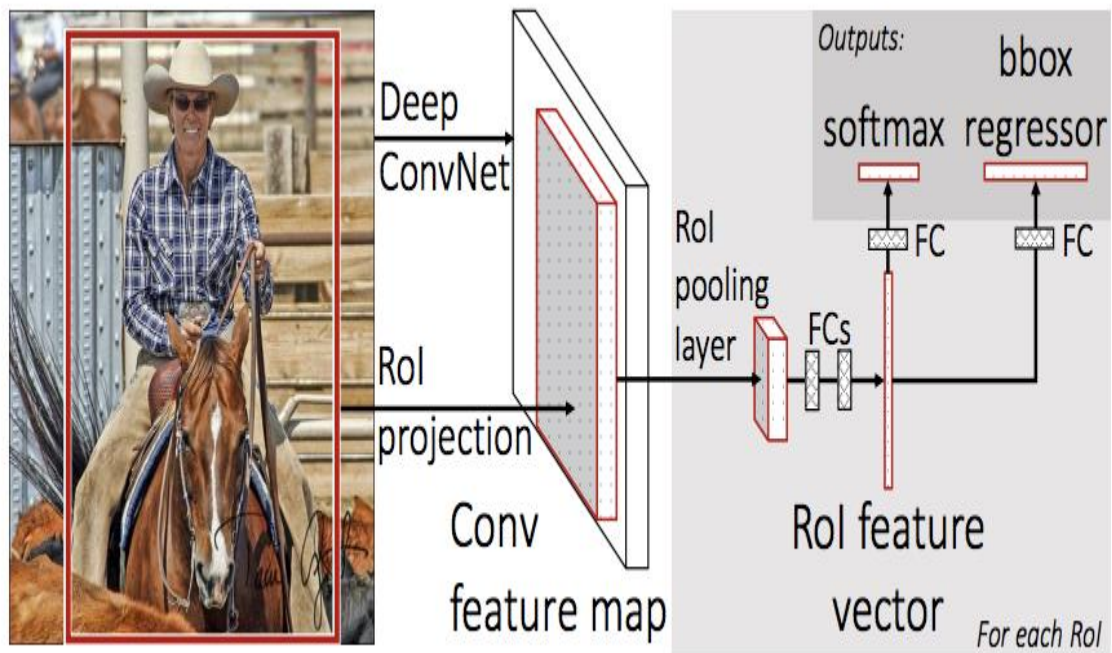
## R-CNN

### Problems with R-CNN

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.



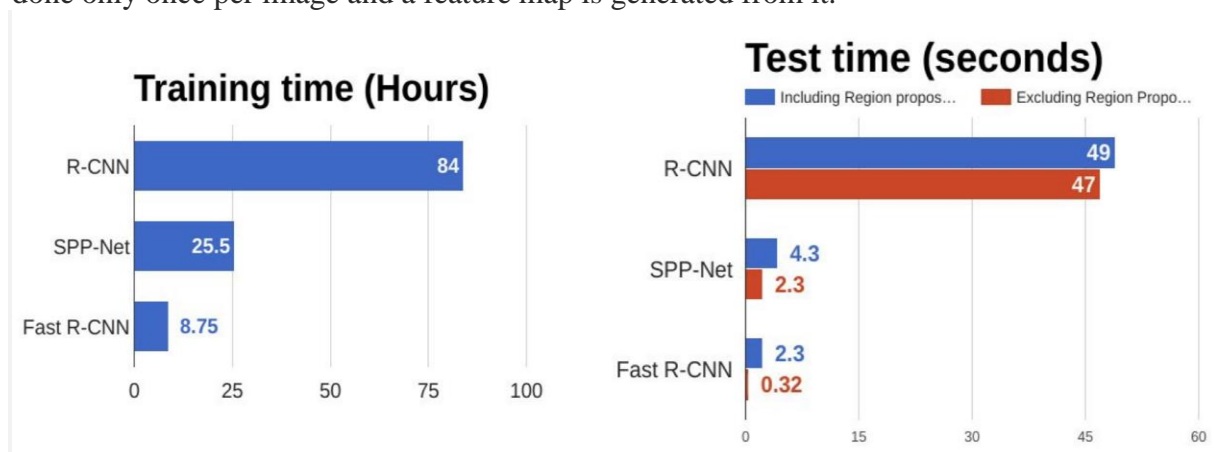
## Fast R-CNN



## Fast R-CNN

The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

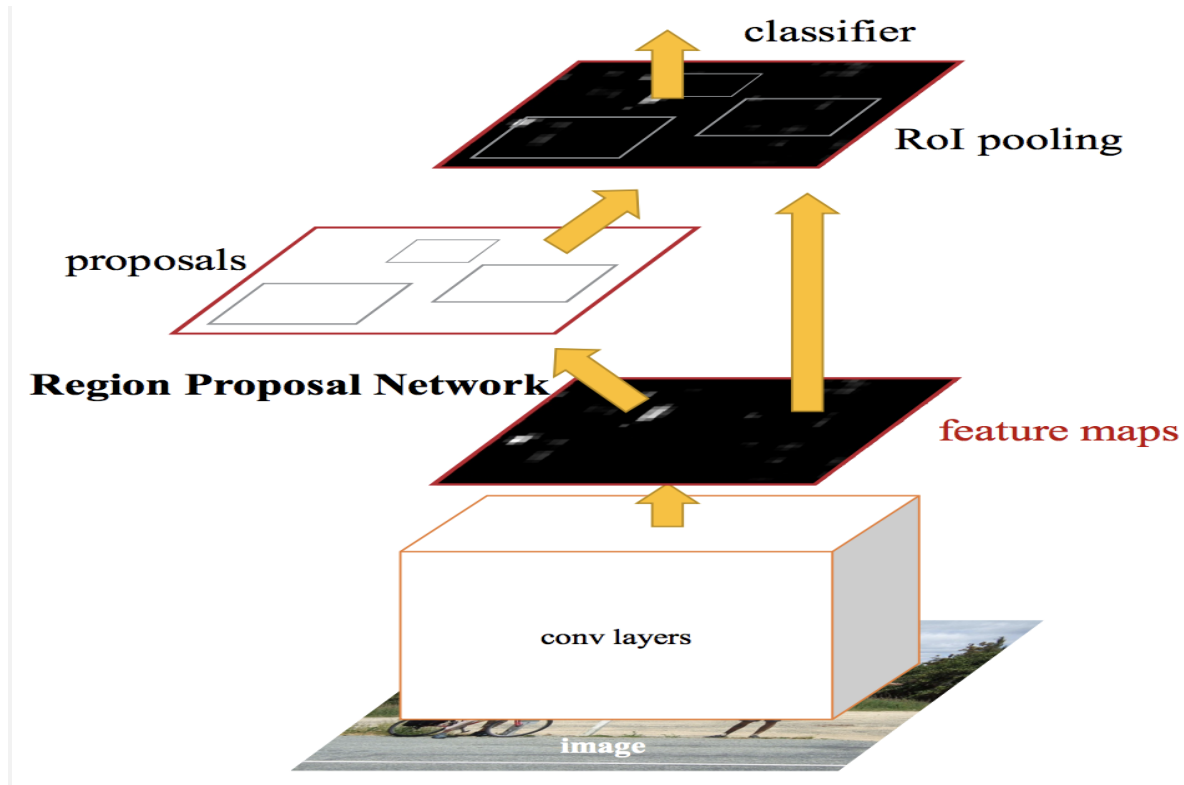
The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.



## Comparison of object detection algorithms

From the above graphs, you can infer that Fast R-CNN is significantly faster in training and testing sessions over R-CNN. When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

### Faster R-CNN

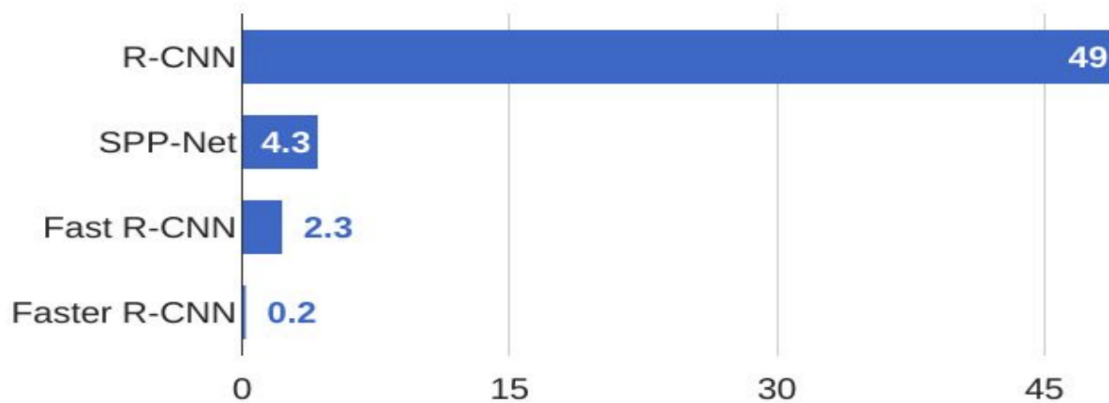


### Faster R-CNN

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

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

## R-CNN Test-Time Speed

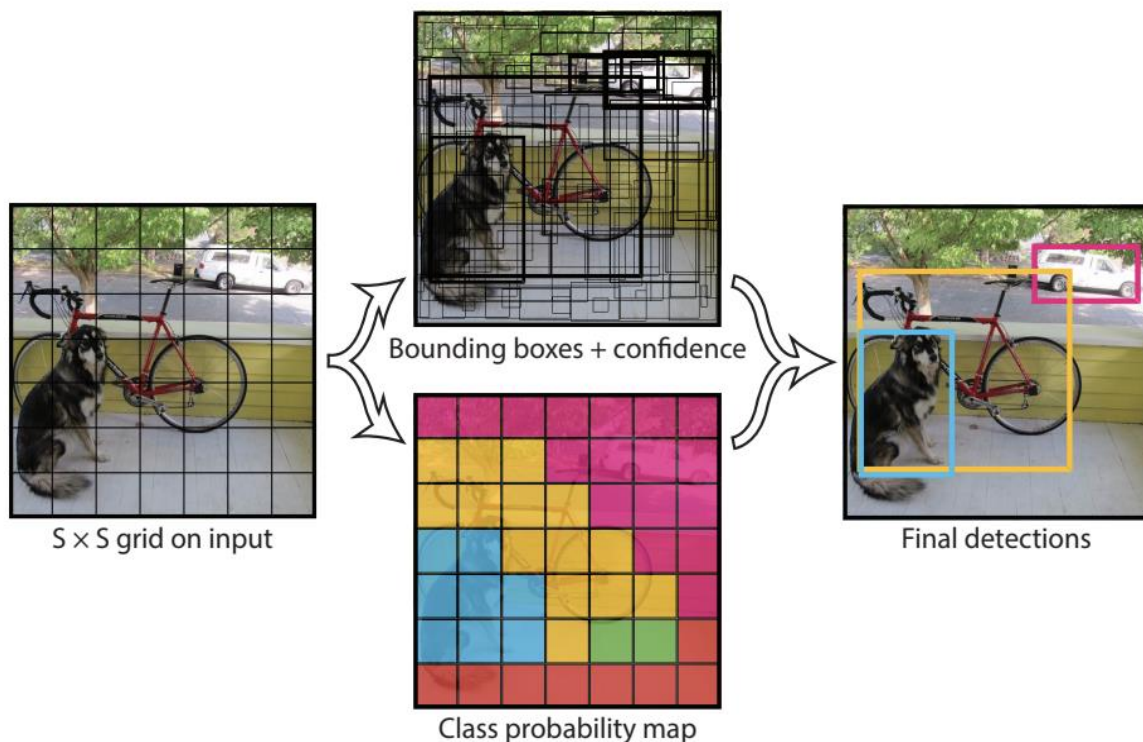


Comparison of test-time speed of object detection algorithms

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

## YOLO — You Only Look Once

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region based algorithms seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.





## YOLO

How YOLO works is that we take an image and split it into an  $S \times S$  grid, within each of the grid we take  $m$  bounding boxes. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

YOLO is orders of magnitude faster(45 frames per second) than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

## Chapter 5

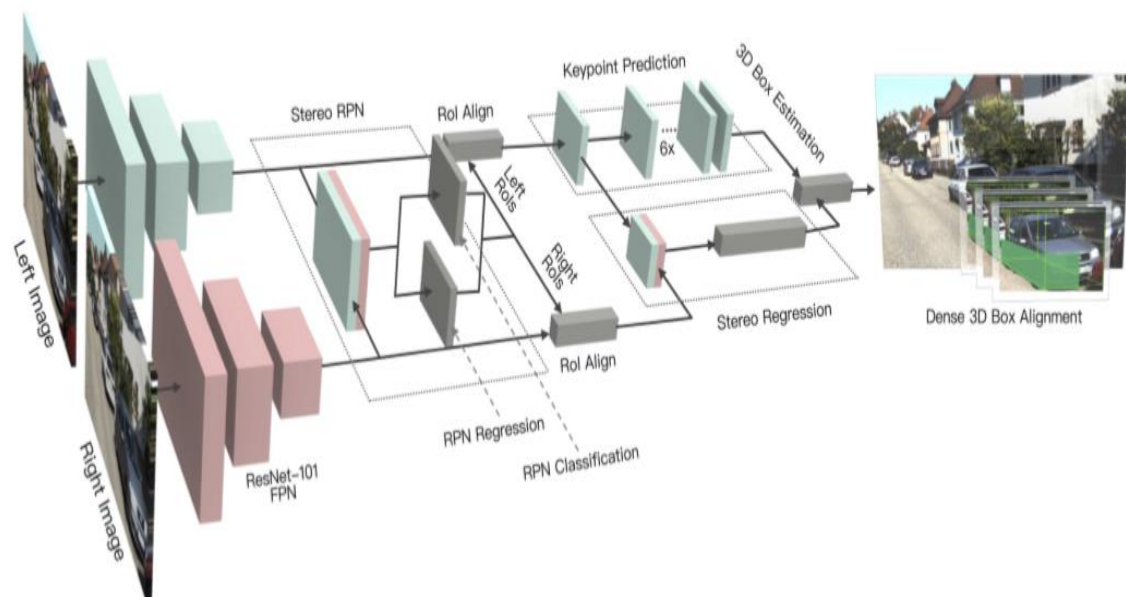
### Stereo R-CNN

#### • Stereo RCNN

With Stereo RCNN there are two other popular methods for 3D object detection but why stereo then?

1. LiDAR-based 3D Object Detection : Currently, most of the 3D object detection methods heavily rely on LiDAR data for providing accurate depth information in autonomous driving scenarios. However, LiDAR has the disadvantage of high cost, relatively short perception range ( $\sim 100$  m). Gaint companies like Tesla uses this method, this costs 50% of car's cost. But Stereo RCNN method costs only 5% of car's cost.
2. Monocular-based 3D Object Detection : monocular camera provides alternative low-cost solutions for 3D object detection. The depth information can be predicted by semantic properties in scenes and object size, etc. However, the inferred depth cannot guarantee the accuracy, especially for unseen scenes. It uses one camera.

Stereo RCNN based 3D Object Detection : Comparing with monocular camera, stereo camera provides more precise depth information by left-right photometric alignment. Comparing with LiDAR, stereo camera is low-cost while achieving comparable depth accuracy for objects with non-trivial disparities. The perception range of stereo camera depends on the focal length and the baseline. It uses two or more cameras. Its kind of seeing with one eye(monocular) and seeing with two or more eyes(stereo).



Main part we are going to discuss in this network are:

☐ Stereo R-CNN Network

☐ Stereo RPN

☐ Stereo R-CNN

1. Stereo Regression

2. Keypoint Prediction

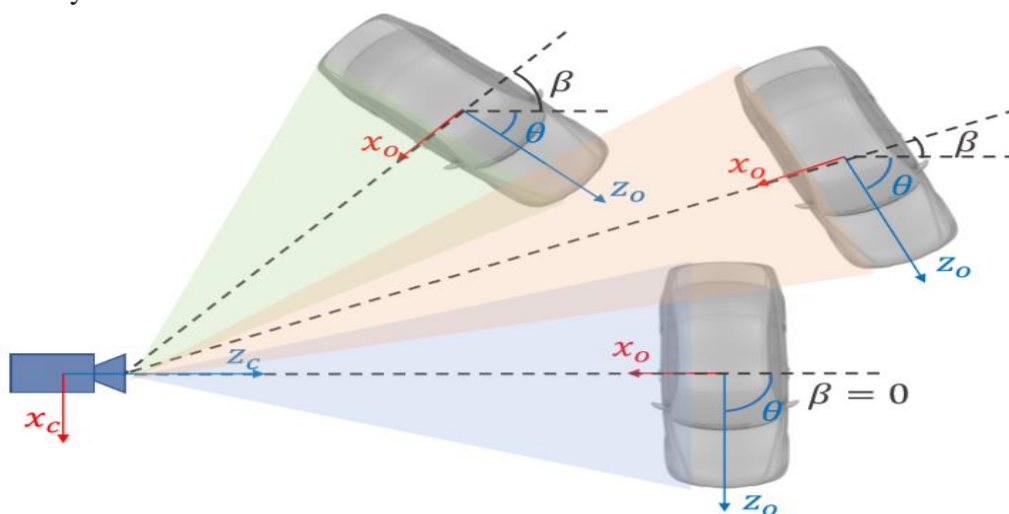
☐ 3D Box Estimation

☐ Dense 3D Box Alignment

**Stereo R-CNN Network** Stereo R-CNN can simultaneously detect and associate 2D bounding boxes for left and right images with minor modifications. We use weight-share ResNet-101 and FPN as our backbone network to extract consistent features on left and right images. It outputs feature maps. ResNet-101: Residual Network, which contains 101 layers. FPN: Feature Pyramid Network

**Stereo RPN** Region Proposal Network (RPN) is a slidingwindow based foreground detector. After feature extraction, a 3 X 3 convolution layer is utilized to reduce channel, followed by two sibling fully-connected layer to classify objectness and regress box offsets for each input location which is anchored with pre-define multiple-scale boxes. we concatenate left and right feature maps at each scale, then we feed the concatenated features into the stereo RPN network.

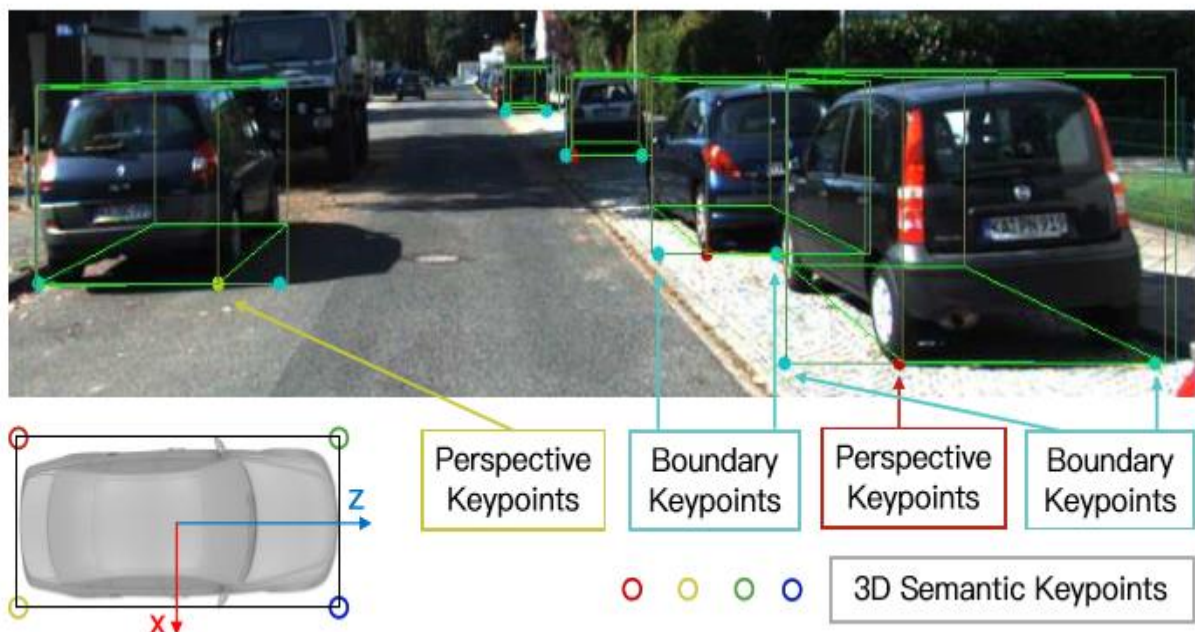
**Stereo R-CNN:** 1. Stereo Regression: After stereo RPN, we have corresponding left-right proposal pairs. We apply RoI Align on the left and right feature maps respectively at appropriate pyramid level. The left and right RoI features are concatenated and fed into two sequential fully-connected layers (each followed by a ReLU layer) to extract semantic information. We use four sub-branches to predict object class, stereo bounding boxes, dimension, and viewpoint angle respectively.



we use  $\theta$  to denote the vehicle orientation respecting to the camera frame, and  $\beta$  to denote the object azimuth respecting to the camera center. Three vehicles have different orientations, however, the projection of them are exactly the same on cropped RoI images. We therefore regress the viewpoint angle  $\alpha$  defined as:  $\alpha = \theta + \beta$ . To avoid the discontinuity, the training targets are  $[\sin\alpha, \cos\alpha]$  pair instead of the raw angle value.

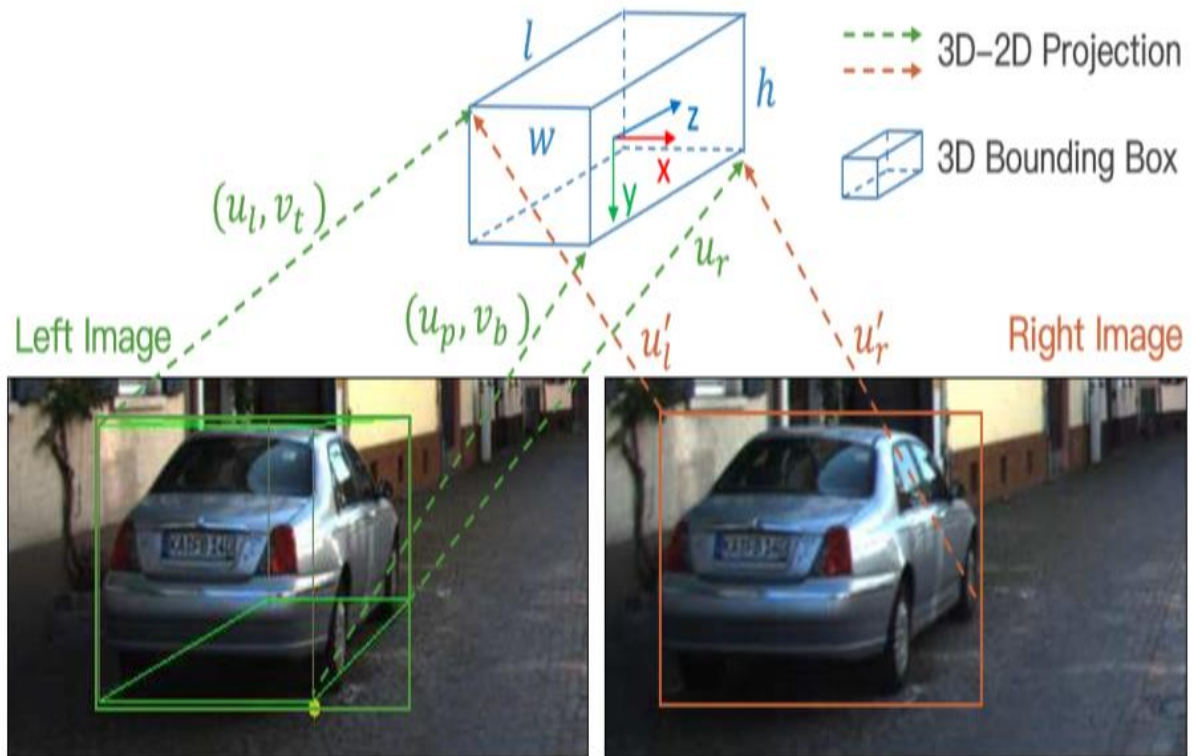
With stereo boxes and object dimension, the depth information can be recovered intuitively, and the vehicle orientation can also be solved by decoupling the relations between the viewpoint angle with the 3D position.

2. Keypoint Prediction: Besides stereo boxes and viewpoint angle, we notice that the 3D box corner which projected in the box middle can provide more rigorous constraints to the 3D box estimation.



we define four 3D semantic keypoints which indicate four corners at the bottom of the 3D bounding box. There is only one 3D semantic keypoint can be visibly projected to the box middle (instead of left or right edges). We define the projection of this semantic keypoint as perspective keypoint.

**3D Box Estimation** we solve a coarse 3D bounding box by utilizing the sparse keypoint and 2D box information. States of the 3D bounding box can be represented by  $x = \{x, y, z, \theta\}$ , which denotes the 3D center position and horizontal orientation respectively. Given the left-right 2D boxes, perspective keypoint, and regressed dimensions, the 3D box can be solved by minimize the reprojection error of 2D boxes and the keypoint.



we extract seven measurements from stereo boxes and perspective keypoints:  $z = \{u_l, v_t, u_r, v_b, u'_l, u'_r, u_p\}$ , which represent left, top, right, bottom edges of the left 2D box, left, right edges of the right 2D box, and the  $u$  coordinate of the perspective keypoint.

$$v_t = (y - h/2)/(z - w/2 \sin\theta - l/2 \cos\theta),$$

$$u_l = (x - w/2 \cos\theta - l/2 \sin\theta)/(z + w/2 \sin\theta - l/2 \cos\theta),$$

$$u_p = (x + w/2 \cos\theta - l/2 \sin\theta)/(z - w/2 \sin\theta - l/2 \cos\theta),$$

$$u'_r = (x - b + w/2 \cos\theta + l/2 \sin\theta)/(z - w/2 \sin\theta + l/2 \cos\theta).$$
 Truncated edges are dropped on above seven equations. These multivariate equations are solved via Gauss-Newton method.

This project contains the implementation of CVPR 2019 paper [arxiv](#).



## Chapter 6

### Results and Future Scope

#### Output of Stereo 3D Object Detection:

Here 3D bounding box across the cars are on the left, right camera images are on top and bottom respectively.



#### Future scope:

- This Stereo R-CNN is heavy for computation and we know YOLO (You Only Look Once) method is not that heavy for computation.
- By combining the YOLO model and Stereo R-CNN, we could even run out 3D Objection Model in our smart phones, as YOLO can be run in smart phones.
- It decreases cost also.
- No paper is published on these till now.

## References:

- [1] Stereo R-CNN based 3D Object Detection for Autonomous Driving, Li, Peiliang and Chen, Xiaozhi and Shen, Shaojie, CVPR, 2019
- [2] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR), pages 2040–2049, 2017.
- [3] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5410–5418, 2018.
- [4] X.Chen,K.Kundu,Z.Zhang,H.Ma,S.Fidler,andR.Urtasun. Monocular 3d object detection for autonomous driving. In European Conference on Computer Vision, pages 2147– 2156, 2016.
- [5] X.Chen,K.Kundu,Y.Zhu,H.Ma,S.Fidler,andR.Urtasun. 3d object proposals using stereo imagery for accurate object class detection. In TPAMI, 2017.
- [6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In IEEE CVPR, volume 1, page 3, 2017.
- [7] M.Engelcke,D.Rao,D.Z.Wang,C.H.Tong,andI.Posner. Vote3deep: Fast object detection in 3d point clouds using efficientconvolutionalneuralnetworks. InRobotics and Automation (ICRA), 2017 IEEE International Conference on, pages 1355–1361. IEEE, 2017.
- [8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomousdriving? thekittivisionbenchmarksuite. InComputer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3354–3361. IEEE, 2012.
- [9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In Computer Vision (ICCV), 2017 IEEE International Conference on, pages 2980–2988. IEEE, 2017.
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.