A survey of Object Classification and Detection based on 2D/3D data

Xiaoke Shen

The Graduate Center, City University of New York

Abstract

Recently, by using deep neural network based algorithms, object classification, detection and semantic segmentation solutions are significantly improved. However, one challenge for 2D image-based systems is that they cannot provide accurate 3D location information. This is critical for location sensitive applications such as autonomous driving and robot navigation. On the other hand, 3D methods, such as RGB-D and RGB-LiDAR based systems, can provide solutions that significantly improve the RGB only approaches. That is why this is an interesting research area for both industry and academia.

Compared with 2D image-based systems, 3D-based systems are more complicated due to the following five reasons: 1) Data representation itself is more complicated. 3D images can be represented by point clouds, meshes, volumes. 2D images have pixel grid representations. 2) The computation and memory resource requirement is higher as an extra dimension is added. 3) Different distribution of the objects and difference in scene areas between indoor and outdoor make one unified framework hard to achieve. 4) 3D data, especially for the outdoor scenario, is sparse compared with the dense 2D images which makes the detection task more challenging. Finally, large size labelled datasets, which are extremely important for supervised based algorithms, are still under construction compared with well-built 2D datasets such as ImageNet.

Based on those challenges listed above, the described systems are organized by application scenarios, data representation methods and main tasks addressed. At the same time, critical 2D based systems which greatly influence the 3D ones are also introduced to show the connection between them.

1 Introduction

Currently, great achievements have been shown in 2D-based systems by using deep neural networks. Actually, the neural network is not a new construct. It was first introduced in 1950s. In 1958, Rosenblatt [1] created the perceptron, an algorithm for pattern classification. The perceptron algorithm's first implementation, in custom hardware, was one of the first artificial neural networks to be produced. Although the perceptron initially seemed promising, Neural network research stagnated after machine learning research by Minsky and Papert (1969) [2], who discovered two key issues with the computational machines that processed neural networks. The first was that basic perceptrons were incapable of processing the exclusive-or circuit. The second was that computers did not have enough processing power to effectively handle the work required by large neural networks [3]. The first issue was solved by introducing more layers of networks and the second issue by both reducing the complexity of the algorithms and introducing more powerful computing hardware such as GPUs.

By using deep neural network based on algorithms, especially convolutional neural networks based algorithms, computer vision systems based on 2D images have been making great achievements in image classification, object detection and semantic segmentation since the year 2012. For some scenarios, deep neural network based algorithms can achieve a similar or even better performance on 2D image classification/detection and semantic segmentation than the human expert, but 3D-based systems are not as developed yet.

The survey is organized as follows: in the second section, 2D-based systems are introduced and in the third section the 3D-based systems are introduced.

2 2D-image based systems

2D-image based systems are introduced in this section based on the high-level tasks addressed.

2.1 High-level main tasks

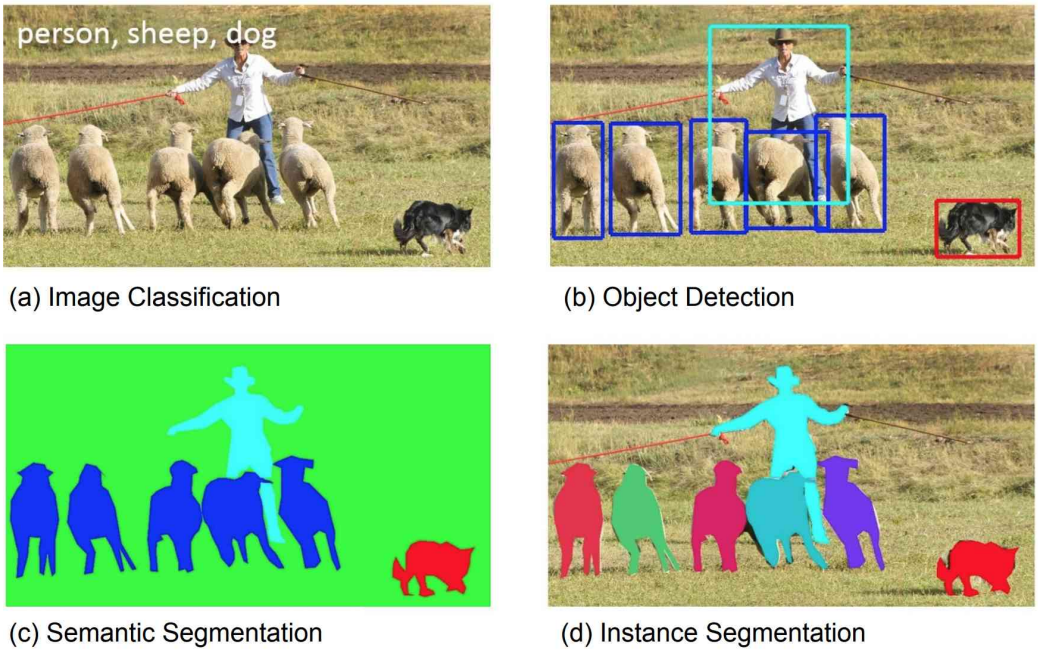


Figure 1: One example of different 2D computer vision tasks: a) Image classification b) Object Detection c) Semantic Segmentation d) Instance Segmentation. Figure is adjusted from [4].

Figure 1 from [4] demonstrates the tasks of Image classification, Object Detection, Semantic Segmentation and Instance Segmentation. Image classification is recognizing the interesting objects in each image and output the objects categories. Object Detection will not only output the objects categories but also the location of those objects with bounding boxes. Both semantic segmentation and instance segmentation will output the pixel level location of each object. The difference between those two tasks is that semantic segmentation does not distinguish the instance in the same category while the instance segmentation does.

2.2 Main Networks used for Image Classification

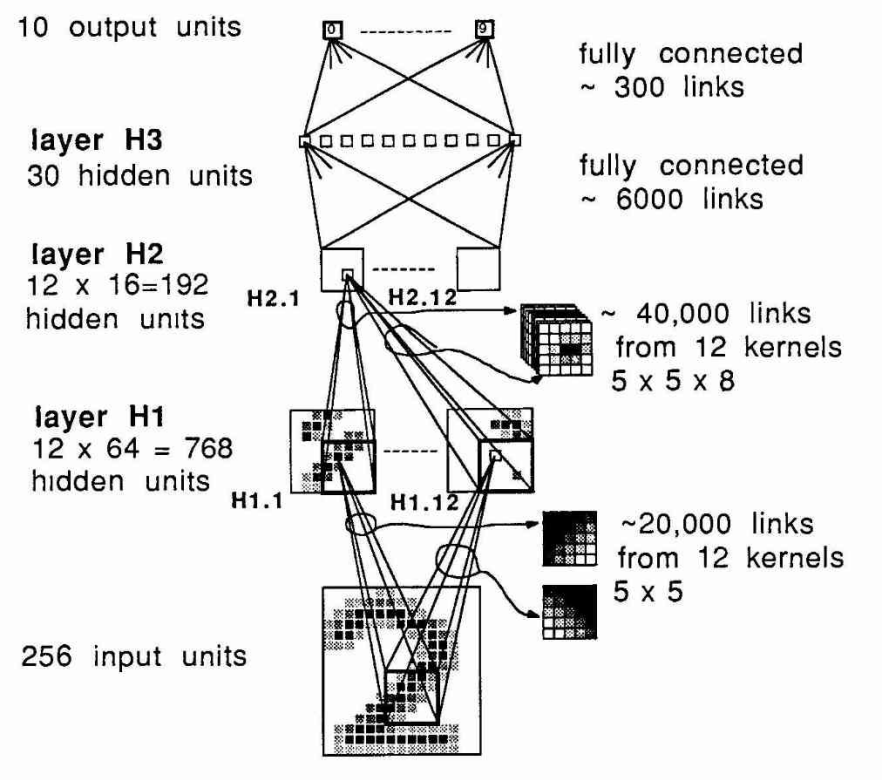


Figure 2: An early Neural Network used in [5].

Some traditional algorithms used for the image classification are nearest neighbor and SVM. The features are the flattened pixel values. In the year 1989, the first important application [5] of using BP(Back Propagation) appeared. From this paper, a basic structure is shown in Figure 2. Similar structures are used in the modern neural networks such as AlexNet [6], VGG 16 [7] and ResNet [8]. The basic idea of the CNN was also introduced in [5].

One important reason of the success of deep learning algorithms in the computer vision area is the invention of CNN. Another important reason is the availabIlity of large labelled datasets. As we know, in the machine learning research area, two kinds of learning approaches can be done: supervised learning and unsupervised learning. For the supervised learning algorithms, the labeled data is required to train the algorithm. So the availability of the labeled data is very important to the development of the supervised learning based algorithms. The ImageNet dataset [9] provides 1.2 million

high-resolution labeled images of 1000 categories. This dataset became one of the most important datasets related to the object classification.

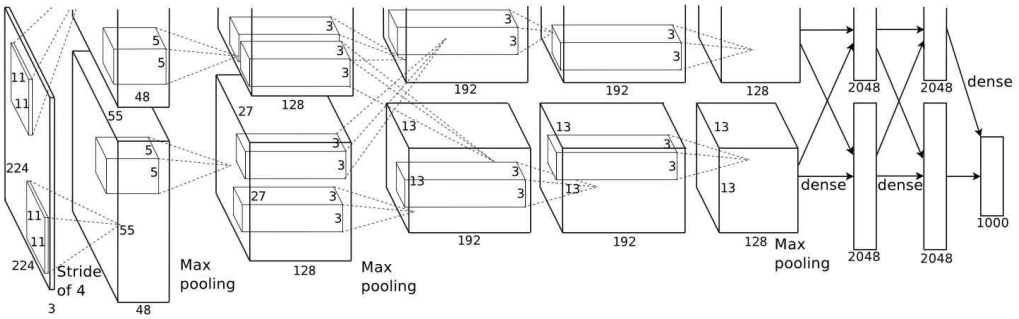


Figure 3: An illustration of the architecture of AlexNet [6]. Figure is from [6].

In the ILSVRC-2012 competition, the method of [6] achieved a top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry. The outstanding performance of deep neural network used in this paper brought the focus back to the neural network research again. the CNN network used in [6] is shown in Figure 3. Two GPUs were used to speed up the calculation. Dropout [10] was used here and was proved to be effective to reduce the overfitting problem. This structure is called AlexNet to emphasize the unique contribution of the author of this paper. Negatives of this network are it is a bit complicated and the structure is not so elegant. This was addressed by the future works.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4: An illustration of the VGG network structure. Figure is from [6].

Two years after the AlexNet, a new network called VGG was proposed in [6]. The structure of this network is very tidy and elegant. In contrast to the AlexNet [6] that used different size of the convolutional kernels, in the VGG CNN network [7] , only 3 by 3 kernels were used for the whole network. At the same time, the trained network weights based on the ImageNet [9] dataset were shared with the public. The negative of this model is that there are too many parameters and thus the computation is slow.

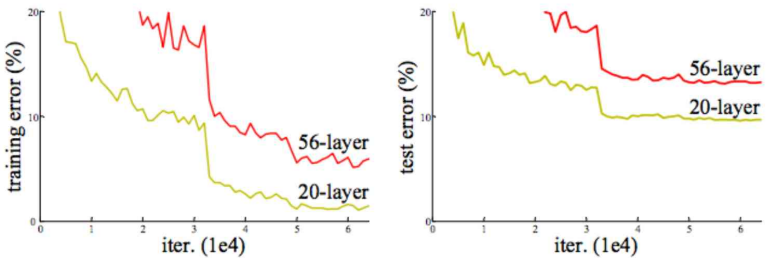


Figure 5: An illustration of deep neural networks fails by using the traditional network structure. Figure is from [8].

In the year of 2015, a new structure called ResNet [8] was proposed and this new structure increased the layers of the network to more than 100 with less parameters than the VGG16 model and better performance. The motivation of this work was to try to find a deeper neural network structure to achieve better performance as the deepest network prior this work was around 20 to 30 layers. The authors had a basic assumption that if more layers are added, the performance should be at least the same as the smaller networks. However, the traditional network's performance decreases when the layers increase [8] as shown in Figure 5. In order to address this problem, the authors of [8] designed a deep residual network by adding a short cut between every other layer and finally show that this network can achieve a better performance. In [8], layers of the neural network goes up to 101 layer and it has a better performance than the previous state-of-the-art neural network such as VGG16 [7], which only has 16 layers. However, as mentioned in [8], the parameters used for the 101-layer ResNet are even less than the 16-layer VGG16 network. This seems amazing. Indeed, the main contribution of the reduction of the parameters was using the convolutional network layer instead of the fully connected layer.

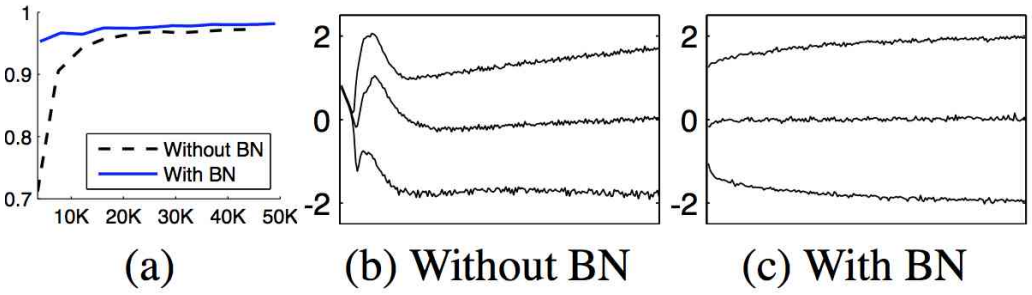


Figure 6: (a) The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy [11]. (b, c) The evolution of input distributions to a typical sigmoid, over the course of training, shown as 15, 50, 85th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift [11]. Figure and Caption are from original paper.

Alongside of the changing of the structure of deep neural networks [8], another algorithm called BN(Batch Normalization) [11] was proposed to speedup the convergence rate of the training process. The main contribution of this paper as shown in Figure 6 is that it greatly reduced the convergence time for the training process. This is why BN became one of the standard training steps for deep neural networks.

There are lots of other new deep neural network models introduced to the community in the past few years. In [12], an analysis of deep neural network models are given based on the ImageNet classification challenge [6]. The network modes analyzed in [12] include: AlexNet [6] , batch normalised AlexNet [13], batch normalised Network In Network (NIN) [14], ENet [15], GoogLeNet [16], VGG-16 and -19 [7], ResNet-18, -34, -50, -101 and -152 [8], Inception-v3 [17] and Inception-v4 [18] .

Top-1 validation accuracies[1] for top scoring single-model architectures is shown in Figure 7. Top1 validation accuracies vs. operations and also size

[1]For top-1 validation, if the top predicted class is the same as the target label, then the prediction is correct. In the case of top-5 validation, if the target label is one of the top 5 predictions, then the prediction is correct. In most work for the ImageNet classification challenge [6], the top-5 validation accuracies were provided. In [12], the comparison is done based on top-1 validation accuracies for all works to make sure it is fair.

of the parameters is shown in Figure 8. The accuracy vs. inferences per second and the parameter size are shown in Figure 9. From those visualization, we can have a clear picture about the parameter size and the performance from both the accuracy and inference time's perspective. Meanwhile, we can see the number of operations is a reliable estimate of the inference time [12] which is consistent with the intuition.

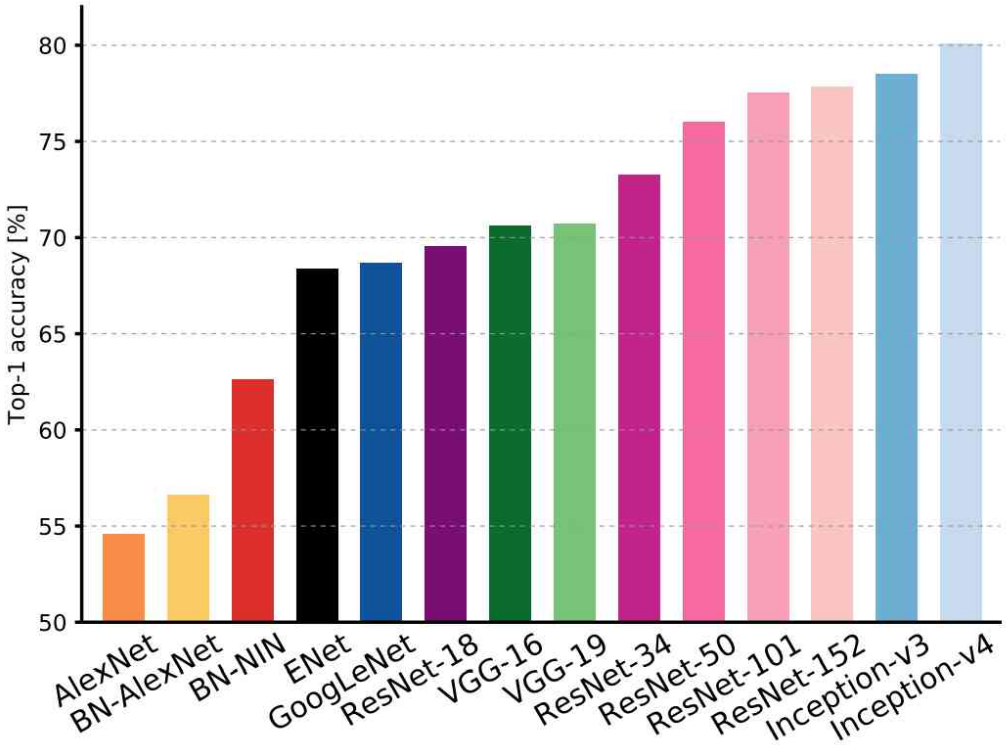


Figure 7: **Top1 vs. network**. Top-1 validation accuracies for top scoring single-model architectures. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.Figure and Caption are adjusted from [12]

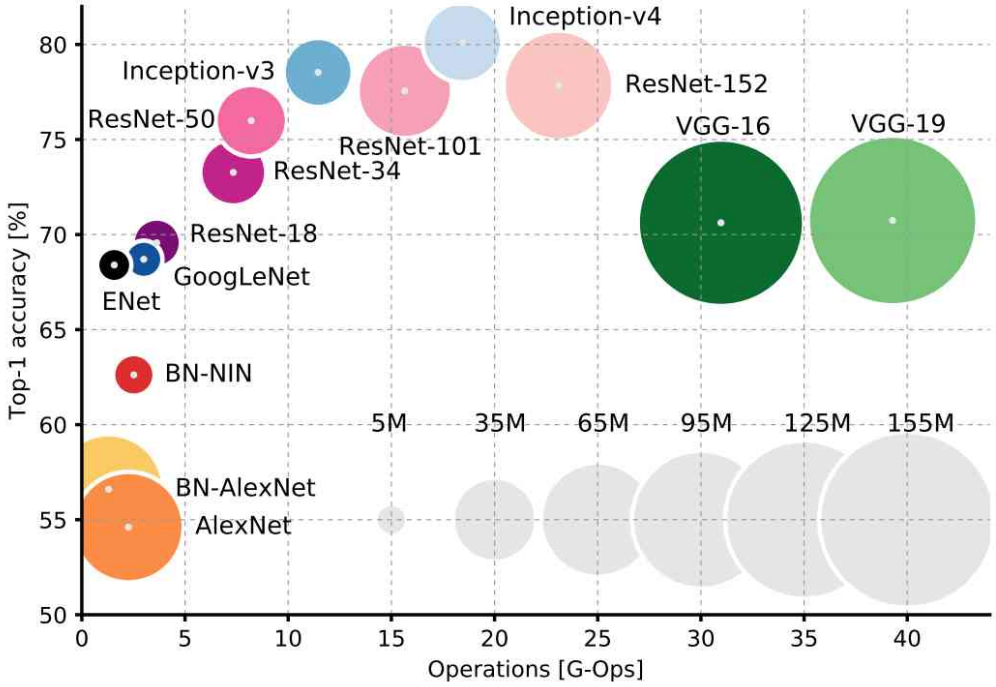


Figure 8: **Top1 vs. operations, size $\propto$ parameters**. Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5106 to 155106 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs. Figure and Caption are from [12]

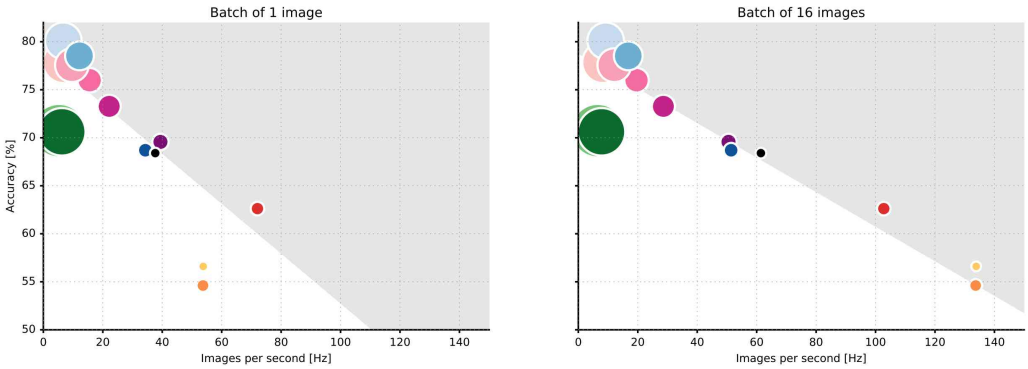


Figure 9: **Accuracy vs. inferences per second, size $\propto$ operations**. Non trivial linear upper bound is shown in these scatter plots, illustrating the relationship between prediction accuracy and throughput of all examined architectures. These are the first charts in which the area of the blobs is proportional to the amount of operations, instead of the parameters count. We can notice that larger blobs are concentrated on the left side of the charts, in correspondence of low throughput, i.e. longer inference times. Most of the architectures lay on the linear interface between the grey and white areas. If a network falls in the shaded area, it means it achieves exceptional accuracy or inference speed. The white area indicates a suboptimal region. E.g. both AlexNet architectures improve processing speed as larger batches are adopted, gaining 80 Hz. Figure and Caption are from [12].

Meanwhile, Figure 10 from [19] we can see the revolution of the depth of DNN on the ImageNet classification.

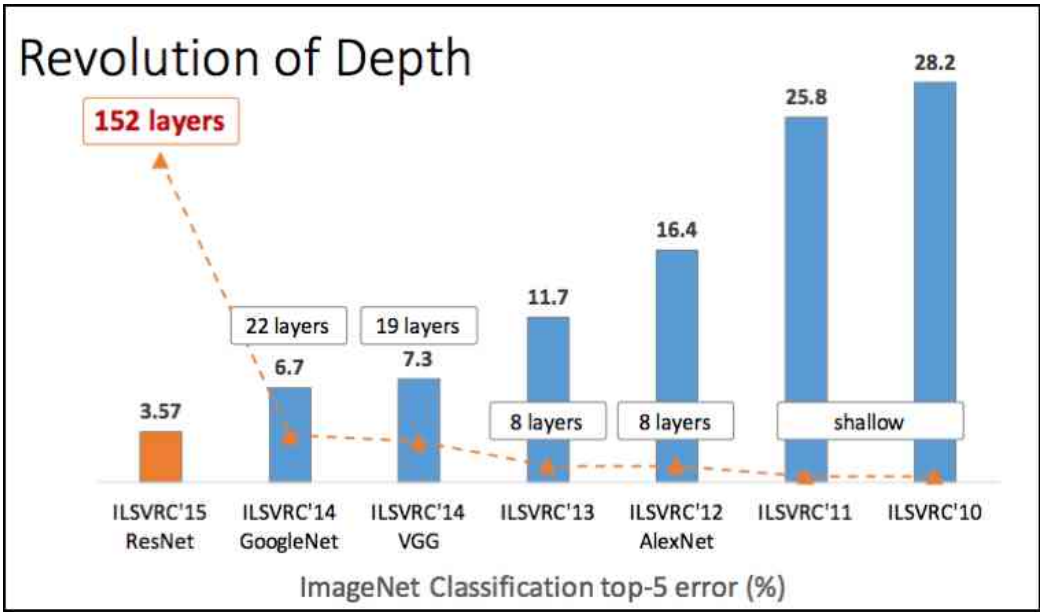


Figure 10: Revolution of Depth of DNN on ImageNet Classification. Figure is from [19]

2.3 Object detection

In the object detection research area, just like in the image classification where the ImageNet dataset is available to train the algorithm, in the object detection, COCO(Common Objects in Context) [4] and VOC dataset can be used to train the algorithms. For the COCO dataset, the bounding box and the mask of the objects are provided. The first important contribution of using deep neural networks to solve the object detection problem is the R-CNN [20]. The region of interest(ROI) of an image is proposed by the selective search(SS) [21] algorithm, and the ROI is cropped to feed a CNN to do the detection. However, as every proposed interesting area will be calculated to predict whether that specified region is an object, the speed of this algorithm is very slow. In order to address this problem, a fast R-CNN algorithm is proposed in [22] by improving the feature map generation efficiency. In another paper which is short for faster RCNN [23], instead of using the SS algorithm to generate ROI, the ROI is proposed by using deep neural network structure. By the combination, the performance of the algorithm can also be improved itself.

2.3.1 Bounding box encoding method

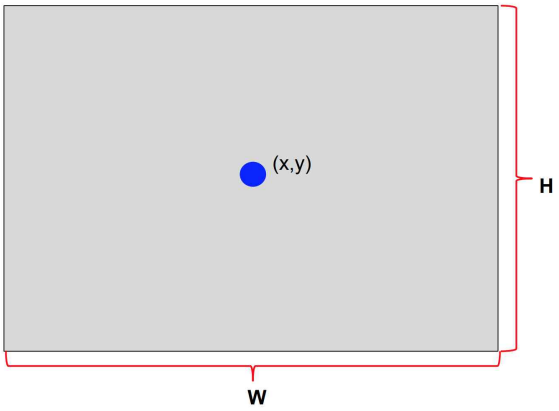


Figure 11: The 2D center offset box encoding method.

In the object detection task, one import part is estimating the bounding box of the object. The 2D bounding box estimation only focus on axis-aligned boxes. The box encoding method is simply based on a simple center coordinates of the bounding box (x, y) and the offset of the bounding box: width: W and height: H. We are calling this method as 2D center offset box encoding method and it is shown in Figure 11. The RCNN, Fast RCNN, Faster RCNN, YOLO, YOLOv2 and Mask R-CNN are using this kind of bounding box encoding with a slightly difference on the loss calculation.

2.3.2 Two-stage systems: RCNN [20], Fast RCNN [22] and Faster RCNN

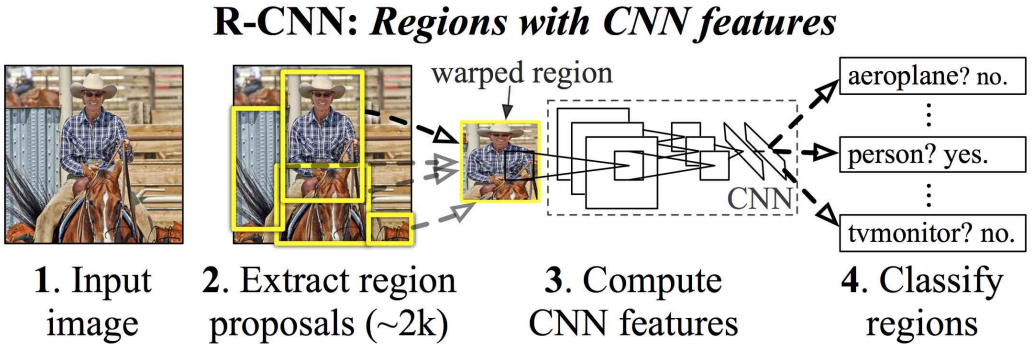


Figure 12: The RCNN [20] system object detection system overview: (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. Figure and Caption are from original paper.

RCNN is an important framework in 2D image detection task which mainly uses the CNN to extract the features for each cropped interesting region. After that the extracted features are fed to a SVM to do the classification and a bounding box regression is followed to improve the bounding box prediction based on the method from [24]. It mainly has two stages: region proposal and detection. As the detection process is computation expensive, the region proposals can make the detection step mainly focus on limited interesting regions (about 2000 regions for a typical image) which greatly reduces the complexity of the whole system and achieves a good performance on the 2D image detection task. This two-stage detection framework is becoming a classical model in both 2D image based object detection and 3D image based object systems. The frame work of RCNN is shown in Figure 12.

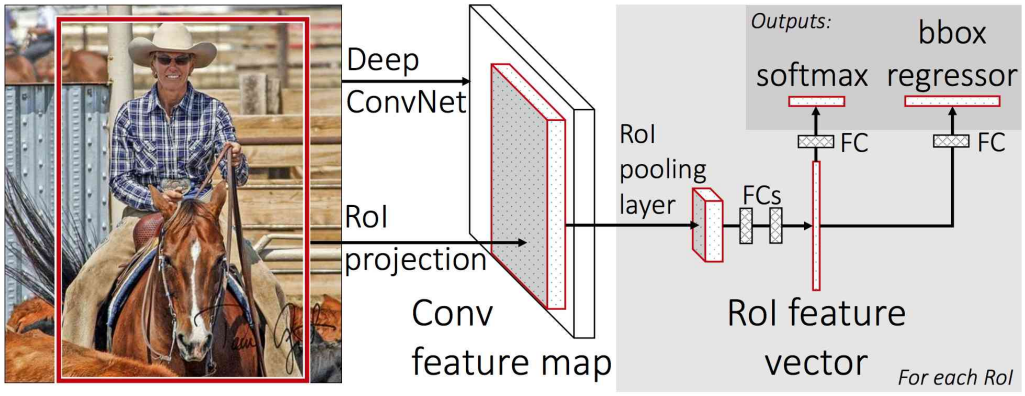


Figure 13: Fast R-CNN [22] architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss. Figure and Caption are from original paper.

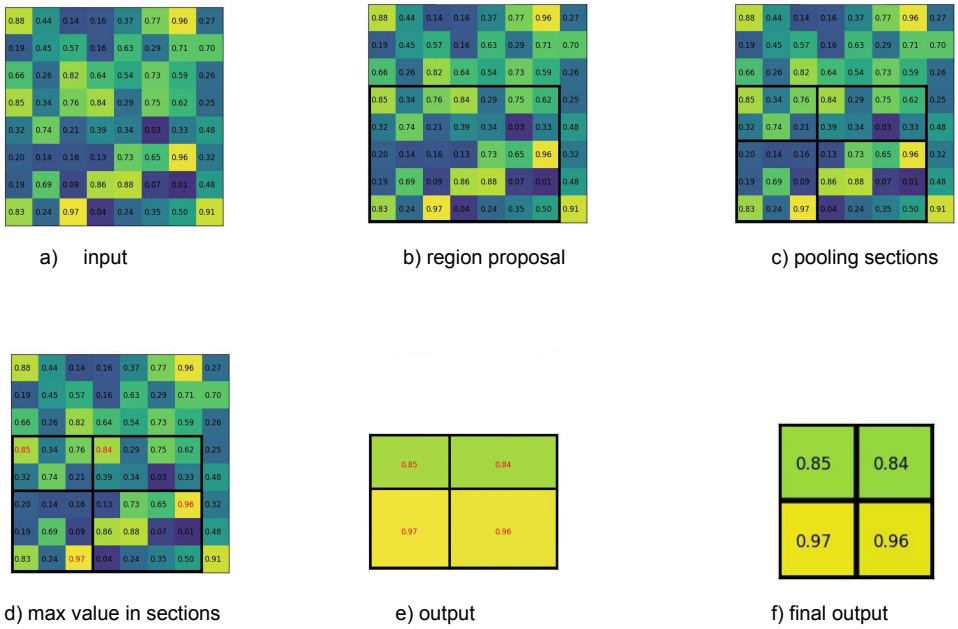


Figure 14: An example of the ROI pooling operation. The max pooling is used. ROI pooling is based on a single 88 feature map, one region of interest and an output size of 2 by 2. The figure is adjusted from [25].

Fast R-CNN improves the RCNN mainly with respect to three aspects: First, instead of doing convolution operations separately for each proposed region, the Fast RCNN does the convolution operations for the whole image firstly and then uses region proposals from the feature map directly to do the further detection. The feature map level region proposals are projected from the region proposals based on the original image. Second, using the softmax layer to replace the SVM classifier to make the detection under one deep learning framework. Finally, Fast R-CNN is using the Multi-task loss to do the object classification and the bounding box regression.
The Fast RCNN framework is shown in Figure 13. In order to have a same size of feature vectors from different size proposed regions, the ROI pooling is used and the ROI pooling is demonstrated in Figure 14.

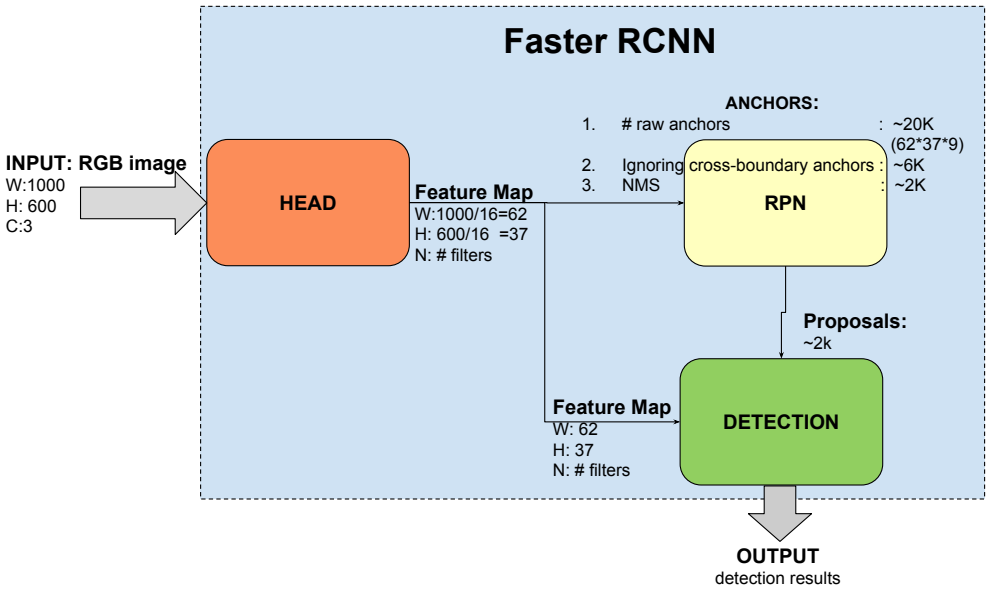


Figure 15: The three main steps for the Faster RCNN [23], system: head(backbone network), RPN and detection network.

The main contribution of the Faster RCNN is introducing the region proposal network(RPN) under the deep learning framework. Before the RPN, the regional proposal is done by traditional method such as SS which is used in both RCNN and Fast RCNN. Since traditional methods such as SS and Edge Box are calculated by CPU, the speed is slow. At the same time, RPN can be calculated by GPU, also the convolutional layer of the RPN and the detection network can be shared, the detection speed for the whole framework of Faster RCNN improved a lot. The framework of Faster RCNN is shown in Figure 15. It mainly contains three steps: the head is used to extract the features by using CNN, then the RPN is used to get the region proposal. Finally, it is using the detection network to do the object detection.

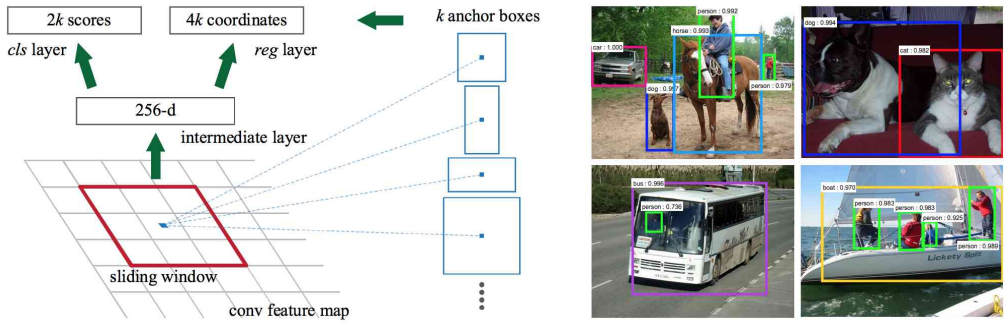


Figure 16: Left: Region Proposal Network (RPN) [23]. Right: Example detections using RPN proposals on PASCAL VOC 2007 test. Figure and Caption are from original paper.

As shown in Figure 16, 1) the RPN is done based on the feature map instead of the original image by using a sliding window method. The size of feature map is smaller than the original image by the pooling layer of the CNN. For example, when the original image size is 1000×600, if the 4 pooling layers are used, then the size of the feature map will be 62×37. The smaller size of feature map makes the regional proposals much faster. The proposal is done by using a $k \times k$ sliding window (the k in 3 in Faster RCNN), and different size and ratio anchors are used to get more accurate proposals. For each anchor, the RPN network will output 1) two scores: foreground score and background score 2) proposal bounding box: by using 2D center offset encoding, it will output 4 values. In Faster RCNN, 3 size and 3 ratio are selected. The comparison of the RPN and SS is shown in Table 1. From the result, we can see, the RPN is faster and the performance is better than SS.

method	# proposals	data	mAP
SS	2000	07	66.9
SS	2000	07+12	70.0
RPN+VGG,unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	73.2

Table 1: Detection results on PASCAL VOC 2007 test set. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. Figure and Caption are from [23],

2.3.3 One-stage systems: YOLO [26] and YOLO v2 [27]

The YOLO and YOLO v2 systems are one-stage detection systems. They do not have a separately region proposal stage. Instead they divide the original image roughly into a $S \times S$ grid and then based on those grid cells will be used as a rough region to do the further processing.

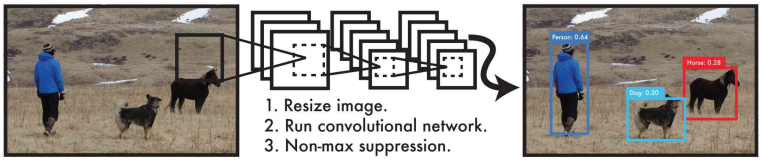


Figure 17: The YOLO [26] Detection System. Processing images with YOLO [26] is simple and straightforward. YOLO (1) resizes the input image to 448 x 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. Figure and Caption are from [26].

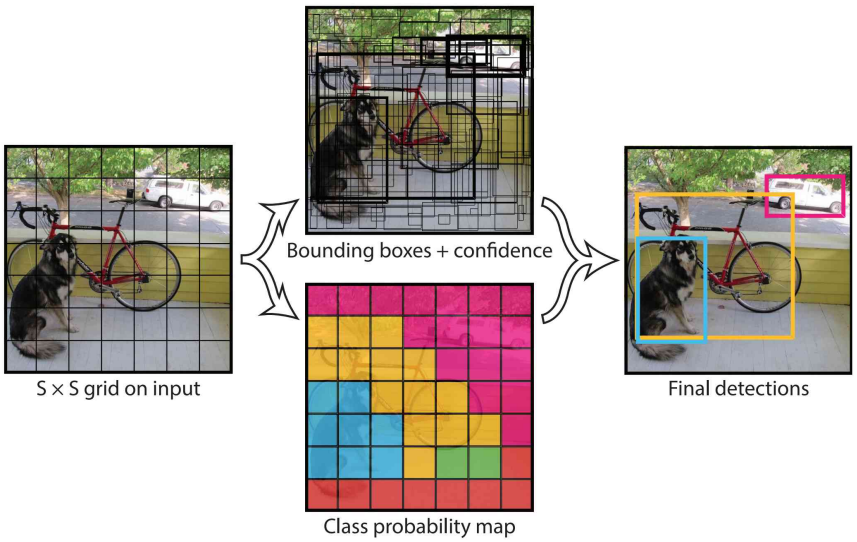


Figure 18: The Model. YOLO [26] models detection as a regression problem. It divides the image into an $S * S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S * S * (B * 5 + C)$ tensor. Figure and Caption are from [26].

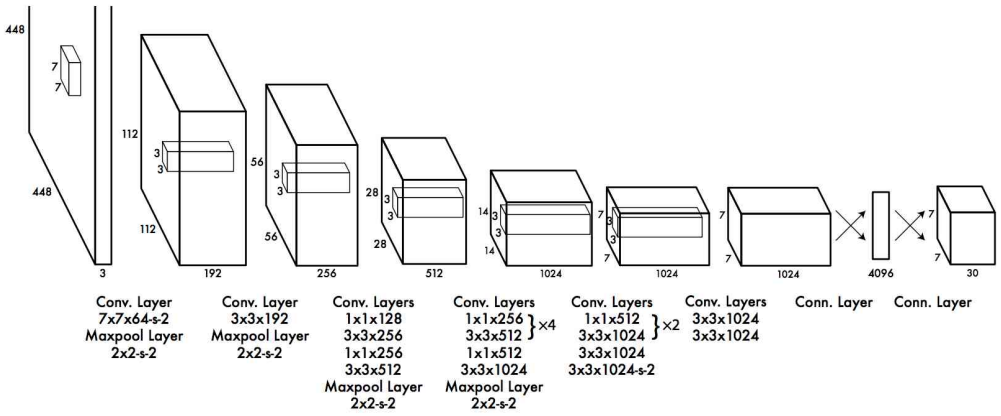


Figure 19: The Darknet Architecture. It has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. YOLO pretrains the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection. Figure and Caption are from [26].

method	network	box encoding	train data	mAP	FPS
Faster RCNN [23]	AlexNet	2D center offset encoding	07+12	62.1	18
Faster RCNN [23]	VGG16	2D center offset encoding	07+12	73.2	7
YOLO [26]	DarkNet	2D center offset encoding	07+12	63.4	45
YOLO [26]	VGG16	2D center offset encoding	07+12	66.4	21

Table 2: Comparison of the network, box encoding and performance for Faster RCNN and YOLO based on the detection results on PASCAL VOC 2007 test set. Training data: "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. The performance of both the Faster RCNN and YOLO can be improved by using more powerful network and other techniques such as batch normalization. In order to have a fair comparison, the improved results for both are not compared here. The improved results of the Faster RCNN can be found from the ResNet [8] and YOLO can be found from YOLO v2 [27]

method	input	resize	feature map or grid size	reception field for each proposal	#proposals for each reception field	total proposals
Faster RCNN [23]	224×224	446×446	27.9×27.9	48×48	9	7056
YOLO [26]	224×224	446×446	7×7	63.7×63.7	2	98

Table 3: Comparison of the number of candidate proposals for the Faster RCNN and YOLO. In the Faster RCNN, the VOC image size is resized with the shorter length as 600 which can generate about 20K raw proposals. Here in order to make a fair comparison, it is resized by using the same dimension as YOLO.

Another important paper for the object detection is YOLO [26] [27]. For the YOLO algorithm, the speed of detection is faster than the RCNN approach, however, the performance is slightly reduced. The Darknet structure which is used for YOLO is shown in Figure 19. The Network, the bounding box encoding and the performance comparison is given in Table 2. Meanwhile, the comparison of the proposal candidates numbers generated by the two method are compared in Table 3. From the comparison shown in Table 3 we can explain that YOLO is faster because fewer proposals are considered by YOLO. However, this introduces worse performance and unsuccessful detection of small objects.

2.3.4 Summary of 2D-based methods

	method
two-stage methods	R-CNN [20]
	fast/er R-CNN [22] [23]
	mask R-CNN [28]
	Light-Head R-CNN [29]
	R-FCN [30]
one-stage methods	YOLO [26], YOLO v2 [27]
	SSD [31]
	DSSD [32]
	RetinaNet [33]

Table 4: Some common object detection frame work by stages: two-stage methods and one-stage methods

one-stage method	# candidate objects
YOLO v1 [26]	98
YOLO v2 [27]	∼1k
OverFeat [34]	∼1-2k
SSD [31]	∼8-26k(hard-example mining)
RetinaNet [33]	∼100-200k("soft" hard-example mining)

Table 5: Number of candidate objects for different single-stage object detection methods.

From the description above we can have a general understanding that for the object detection, there are two kinds of methods: 1) Two-stage methods Detector 2) One-stage methods. Part of those detection algorithms are organized by those two different stage methods and are shown in Table 4. Actually, there is a competition between the two-stage methods and one-stage methods. Generally, the one-stage method is improved by introducing more proposals as shown in Table 5 and by introducing new loss functions such as Focal loss [33] to get rid of the unbalance between the positive proposals and negative proposals. Meanwhile, two-stage methods are also trying to improve the speed by introducing the lighter head, for example Light Head R-CNN [29]. The comparison of those different stage method's performance is shown in Figure 20 and 21 based on the COCO dataset.

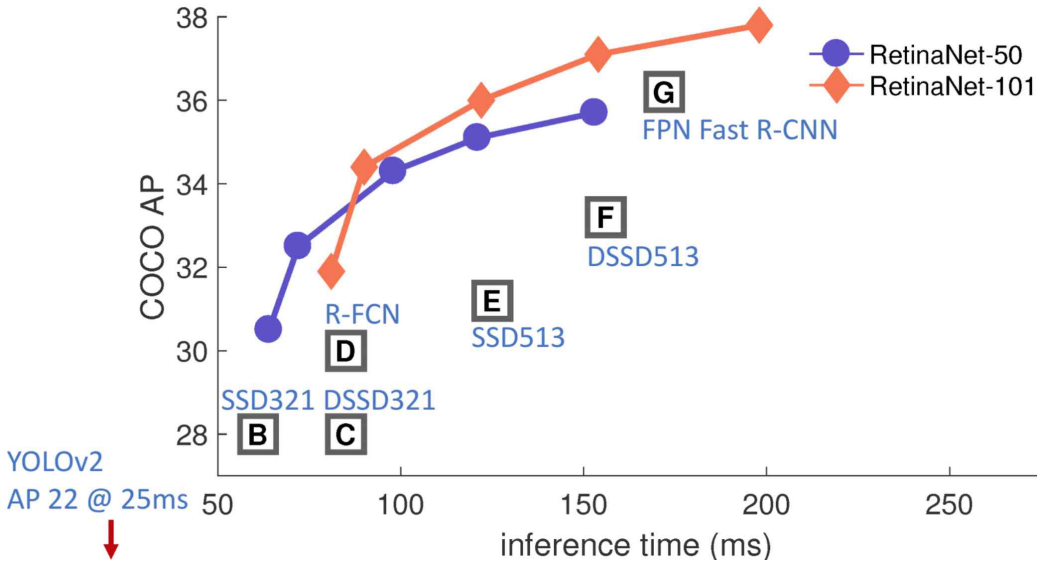


Figure 20: Speed (ms) versus accuracy (AP) on COCO test-dev. Enabled by the focal loss, RetinaNet detector outperforms all previous one-stage and two-stage detectors, including the best reported Faster R-CNN [23] system from [35]. Variants of RetinaNet with ResNet-50-FPN (blue circles) and ResNet-101-FPN (orange diamonds) are shown at five scales (400-800 pixels). Ignoring the low-accuracy regime ($AP < 25$), RetinaNet forms an upper envelope of all current detectors, and an improved variant (not shown) achieves 40.8 AP. Figure and Caption are adjusted from [33]

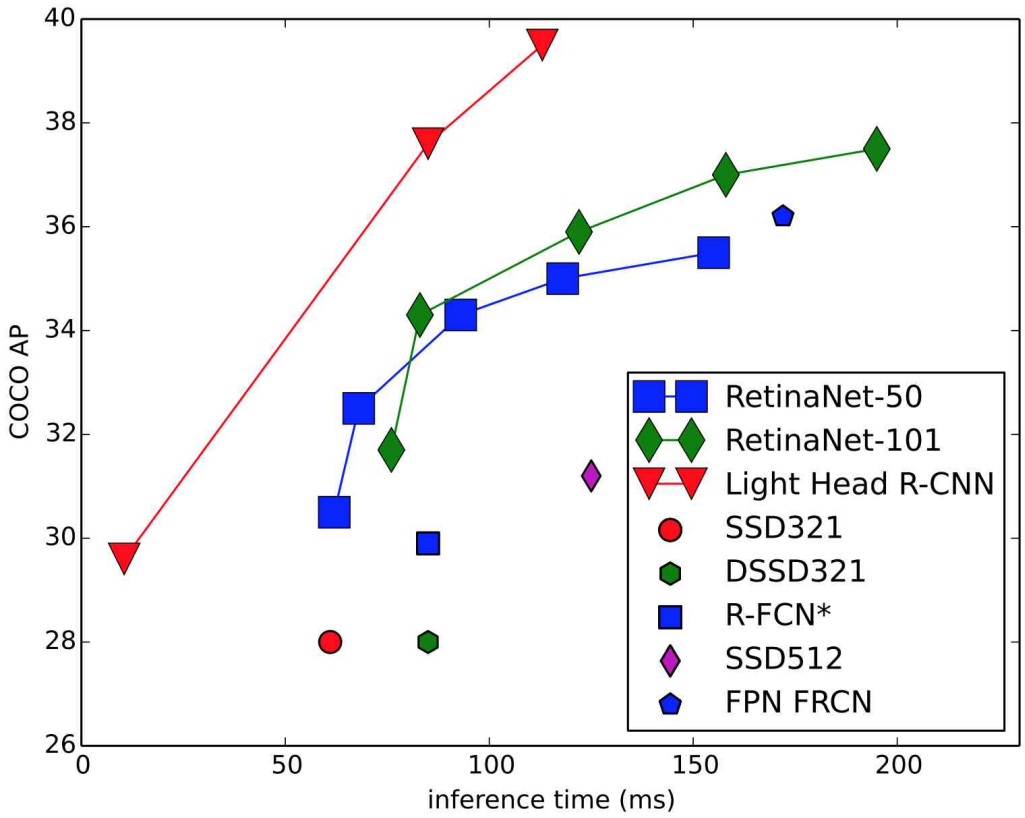


Figure 21: Comparisons of some one-stage and two-stage methods. Figure is from [29].

2.4 Semantic segmentation

For semantic segmentation, a pixel level detection of an object is provided. One important paper in this area is Fully Convolutional Network(FCN) [36]. It upsamples the feature map to make sure that a more accurate location information can be preserved. In addition, the data in the previous layers are combined with the deeper layer to preserve more information and thus improve the accuracy of the semantic segmentation. After this paper, FCN became mainstream in semantic segmentation. DeepLab [37], FCIS [38] and mask-RCNN [28] are using FCN. For the DeepLab [37], the CRF is used to further improve the result by benefiting of the redundant information of nearby pixels. The CRF approach is firstly introduced in [39]. For FCIS [38], location sensitive feature maps are generated to improve the pixel level prediction. The FCIS [38] is the improved version of [30]. So far, the FCN and

the CRF approach became the standard method in the semantic segmentation area. Besides those papers, [40] [41] are also using deep neural network approaches to improve the performance of the semantic segmentation.

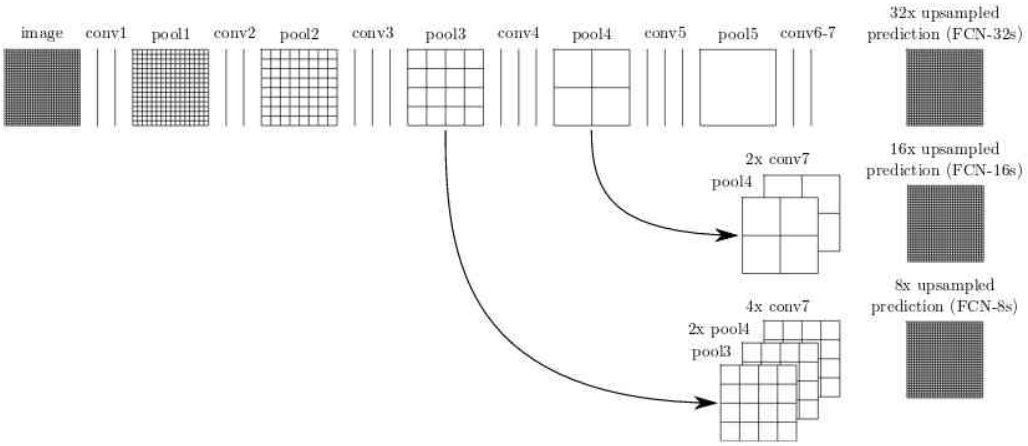


Figure 22: The FCN [36] structure. Figure is from the original paper.

2.4.1 Mask R-CNN [28]

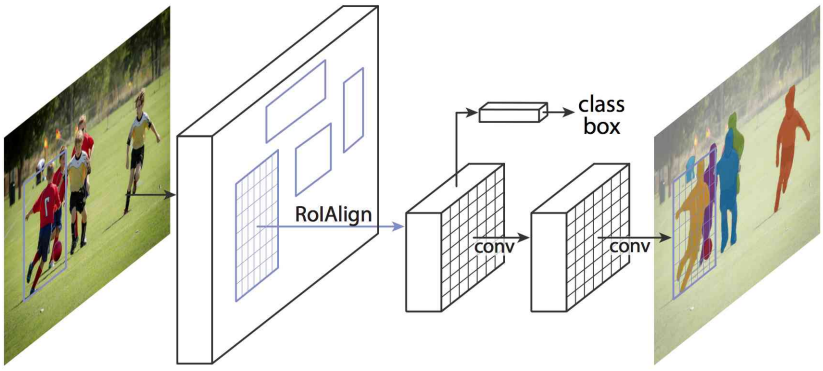


Figure 23: The Mask R-CNN [28] framework for instance segmentation. Figure is from original paper.

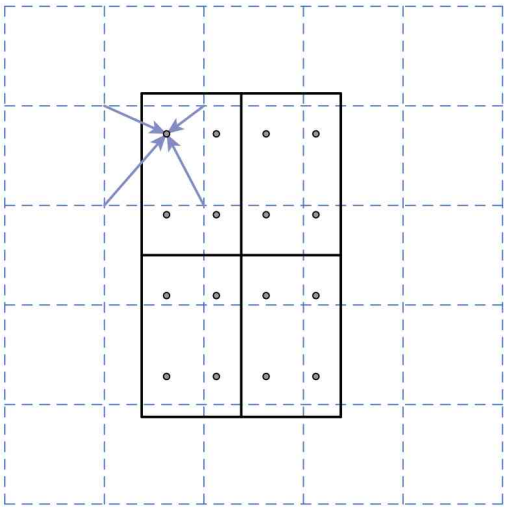


Figure 24: RoIAlign: The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points. Figure and Caption are from [28]

net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

Figure 25: Backbone Architecture: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet. Figure and Caption are from [28]

	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

Figure 26: RoIAlign (ResNet-50-C5, stride 32): Mask-level and box-level AP using large-stride features. Misalignments are more severe than with stride-16 features , resulting in big accuracy gaps. Figure and Caption are from [28]

The Mask R-CNN framework is shown in Figure 23. One important contribution of Mask R-CNN is RoIAlign as shown in Figure 24. The alignment of the ROI is not so sensitive in the object detection, however, in the semantic segmentation it is sensitive as it has to do the detection in pixel level and by employing the alignment, the performance is improved a lot. Another important fact for the improvement of the performance of the Mask R-CNN is using the more powerful backbone networks as observed from the results shown in Figure 25. The performance of RoIAlign is shown in Figure 26.

3 3D-image based systems

In this section, 3D-image based systems will be introduced. The main high-level tasks in 3D vision, 3D image data representation and methods used to do the 3D classification and object detection are briefly described. Several import papers in this area are surveyed.

3.1 Main high-level tasks

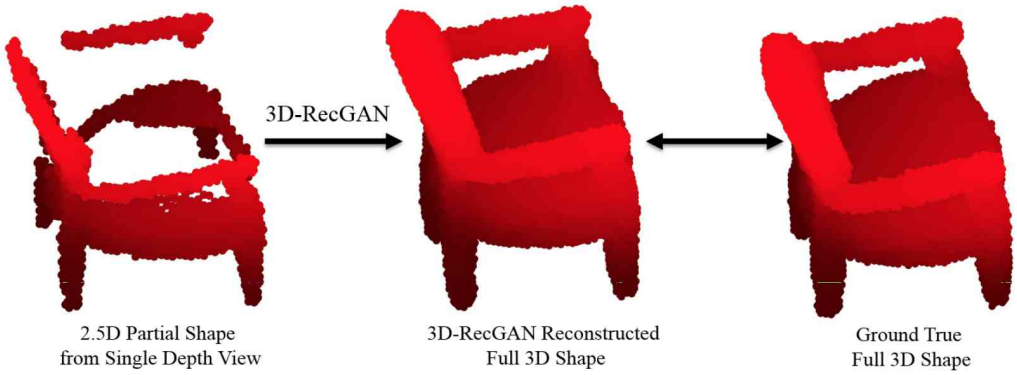


Figure 27: A 3D shape reconstruction example: 3D-RecGAN reconstructs a full 3D shape from a single 2.5D depth view [42]

Similar to the 2D systems, the main tasks also include 3D object classification, 3D object detection, 3D semantic segmentation and 3D instance segmentation. Also, as 3D images commonly suffer from occlusion such as self occlusion and inter object occlusion. In order to address this issue, a 3D shape reconstruction task is also actively researched in the 3D vision understanding community. This part is important, however, it will not be surveyed. An example of 3D shape reconstruction is shown in Figure 27

3.2 3D image data representation

3D images can have multiple data representations, such as multi-view RGB-D images, volumetric images, polygonal meshes and point clouds.

Figure 28: Output from the RGB camera (left) and depth camera (right). Missing values in the depth image are a result of (a) shadows caused by the disparity between the infrared emitter and camera or (b) missing or spurious values caused by specular or low albedo surfaces.

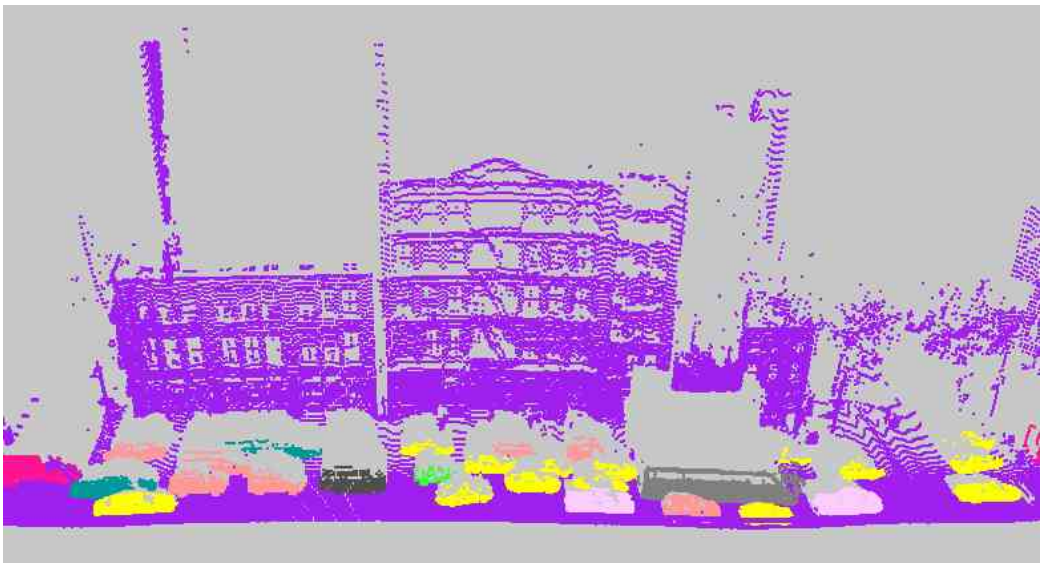


Figure 29: An example of the 3D data from the outdoor urban LiDAR scans. The figure is provided by the Computer Vision Laboratory of Hunter College [43].

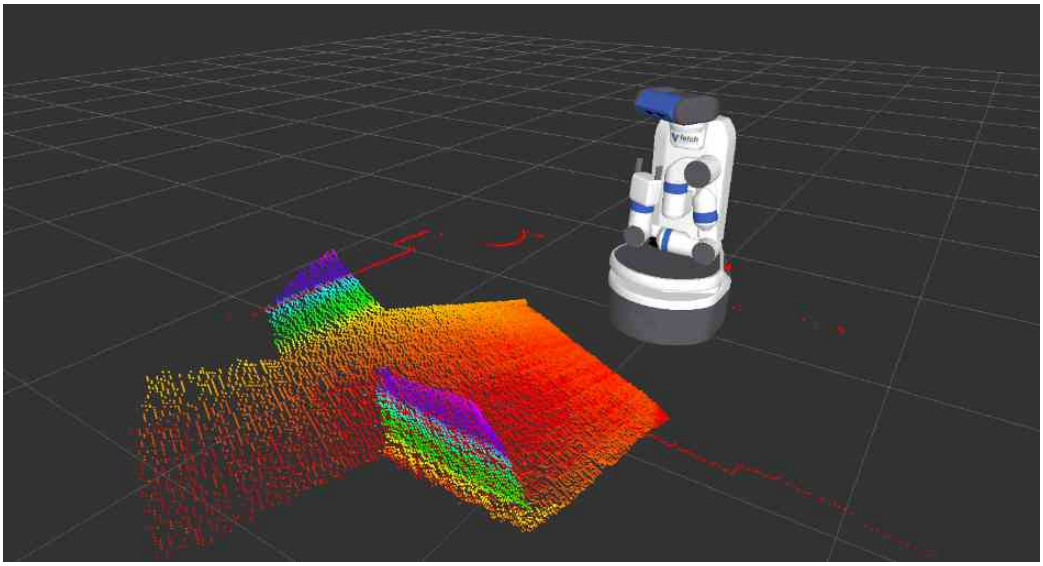


Figure 30: The 3D view from the robot's perspective.

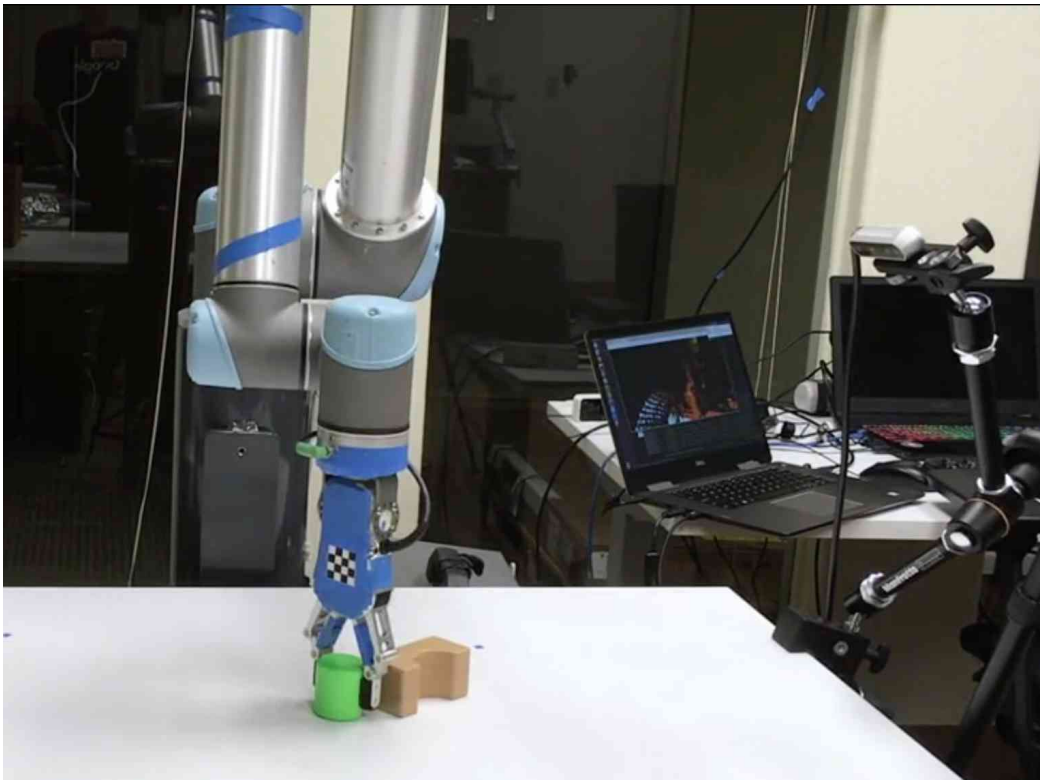


Figure 31: A robot is picking up an object with the help of the RGB-D camera based object detection system.

3D images are becoming more and more important and are widely used in reconstructing architectural models of buildings, navigation of self-driving cars, detection face (such as face ID for iPhone X), preservation of at-risk historical sites, and recreation of virtual environments for film and video game industries.
Mainly, there are two basic kinds of hardware available for the 3D data generation in outdoor and indoor environments. For outdoors, one typical hardware is LiDAR(Light Detection and Ranging). The coverage of this equipment can achieve to hundreds and even thousands meters. Most self-driving cars use a LiDAR scanner. For indoors, in recent years the availability of low-cost sensors such as the Microsoft Kinect have enabled the acquisition of short-range indoor 3D data at the consumer level. Meanwhile, smart phone such as iPhone X a equipped will a depth camera. In Figure 29, one example of the 3D data collected from the outdoor urban LiDAR scanner is shown. In Figure 28, depth map generated by a Kinect camera is provided. Capturing a 3D environment by a robot is shown in Figure 30. A RGB-D

camera based object detection system is used to help a robot to grasp objects as shown in Figure 31.

Figure 32: Shown here are Velodyne's HDL-64E LiDAR sensor (left) and the company's VLS-128 sensor (right). [44]

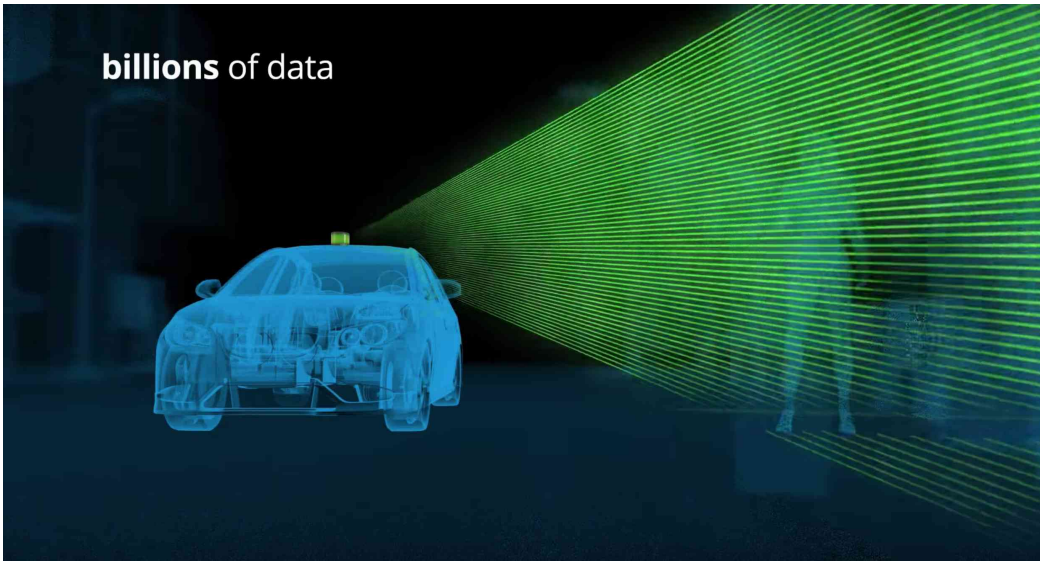


Figure 33: The vertical scan range of LiDAR sensor for Self-driving cars. The photo is collected from [45]

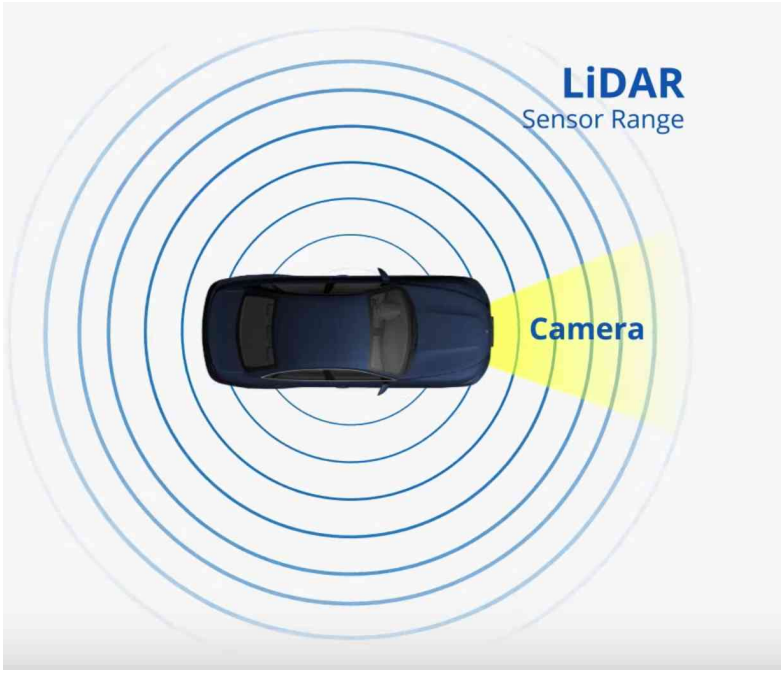


Figure 34: The horizontal scan range of LiDAR sensor for Self-driving cars. The photo is collected from [45]

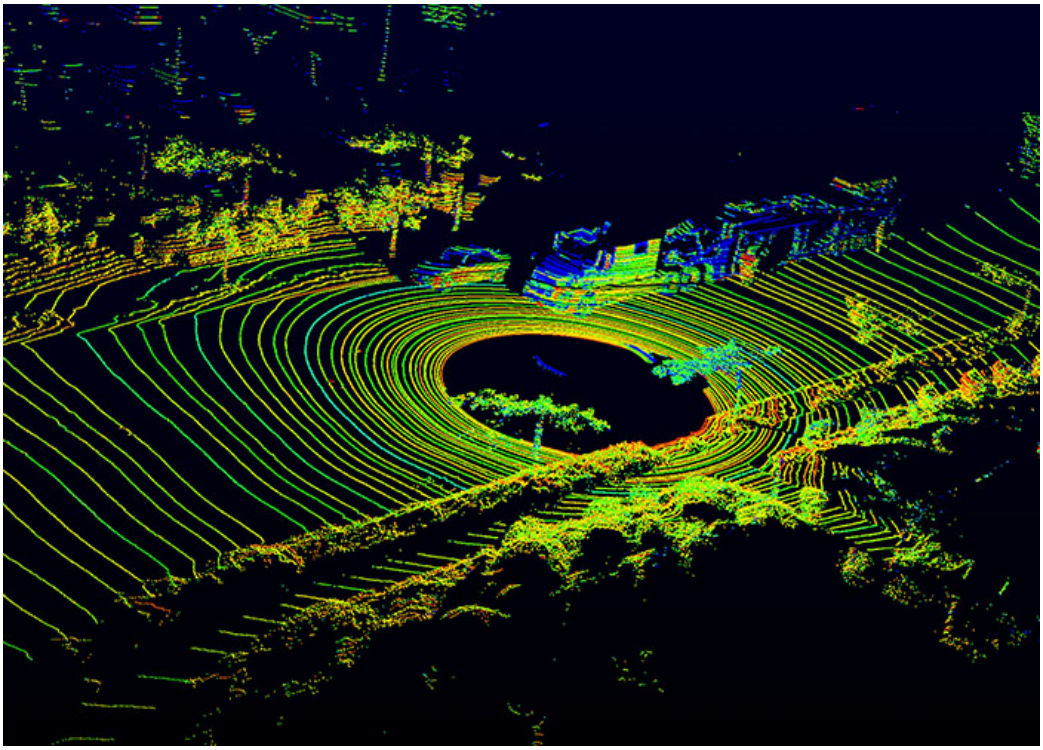


Figure 35: Raw data from Velodyne LiDAR HDL-64E. The photo is collected from Velodyne's official website [46].

Velodyne's HDL-64E LiDAR sensor is commonly used for Self-driving cars. There is a new LiDAR sensor with 128 laser beams released from Velodyne in 2017. The photos of both the 64 beams sensor and the 128 beams sensors is given in Figure 32. The sensor's vertical scan range is shown in Figure 33 while the horizontal scan range is shown in Figure 34. Raw data collected by Velodyne's HDL-64E LiDAR sensor is shown in Figure 35.

3.3 Classification

The classification in 3D is mainly based on the CAD(Computer-aided design) model. One import dataset is ModelNet [47]. ModelNet has 127915 3D CAD models from 662 categories. ModelNet10 and ModelNet40 are two subsets of the ModelNet. ModelNet10 has 4899 models from 10 categories and ModelNet40 has12311 models from 40 categories. For the classification task, the papers will be discussed based on the leaderboard of the ModelNet40. The

leaderboard is given next.

3.3.1 ModelNet40 leaderboard

Algorithm	ModelNet40 Classification (Accuracy)
RotationNet [48]	97.37%
PANORAMA-ENN [49]	95.56%
VRN Ensemble [50]	95.54%
Wang et al. [51]	93.80%
SO-Net [52]	93.40%
Kd-Net [53]	91.80%
3DmFV-Net [54]	91.60%
MVCNN-MultiRes [55]	91.40%
FusionNet [56]	90.80%
PANORAMA-NN [57]	90.70%
Pairwise [58]	90.70%
3D-A-Nets [59]	90.50%
MVCNN [60]	90.10%
Set-convolution [61]	90%
Minto et al. [62]	89.30%
PointNet [48]	89.20%
3DShapeNets [47]	77%

Table 6: ModelNet40 leaderboard. Top 16 as of April 4, 2018 and the baseline 3DShapeNets are shown here. For more results please visit [63]

3.3.2 Inputs for CAD classification

Algorithm	Accuracy	input	#views	others
RotationNet [48]	97.37	multiple-view	20	w/o upright orientation. Max result based on 11 trials. The average accuracy is 94.68
PANORAMA-ENN [49]	95.56	multiple-view	3	
VRN Ensemble [50]	95.54	volumetric	1	
Wang et al. [51]	93.80	volumetric	12	
SO-Net [52]	93.40	points	1	
Kd-Net [53]	91.80	points	1	
3DmFV-Net [54]	91.60	3D Modified Fisher Vectors	1	
MVCNN-MultiRes [55]	91.40	both volumetric and multiple-view		
FusionNet [56]	90.80	both volumetric and multiple-view		
PANORAMA-NN [57]	90.70	PANORAMA	1	
Pairwise [58]	90.70	multiple-view	12	decomposing an image sequence into a set of image pairs
3D-A-Nets [59]	90.50	volumetric	1	
MVCNN [60]	90.10	multiple-view RGB	80	with upright orientation
Set-convolution [61]	90	points	1	
Minto et al. [62]	89.30	both volumetric and multiple-view		
PointNet [48]	89.20	points	1	
3DShapeNets [47]	77	volumetric	1	

Table 7: ModelNet40 leaderboard. Top 16 as of April 4, 2018 and the baseline 3DShapeNets are shown here. For more results please visit [63]

Inputs for each system are list in Table 7. Mainly three kind of inputs are used to do the classification: multiple-view images, 3D voxel grids and point clouds. The multiple-view images and 3D voxel grids are regular input data formats which can be easily fed to a 2D or 3D CNN network. The point clouds are not in a regular format. However, recently, several works are based on this irregular format and achieve impressive results. We will introduce several of those works next.

3.3.3 MVCNN [60]

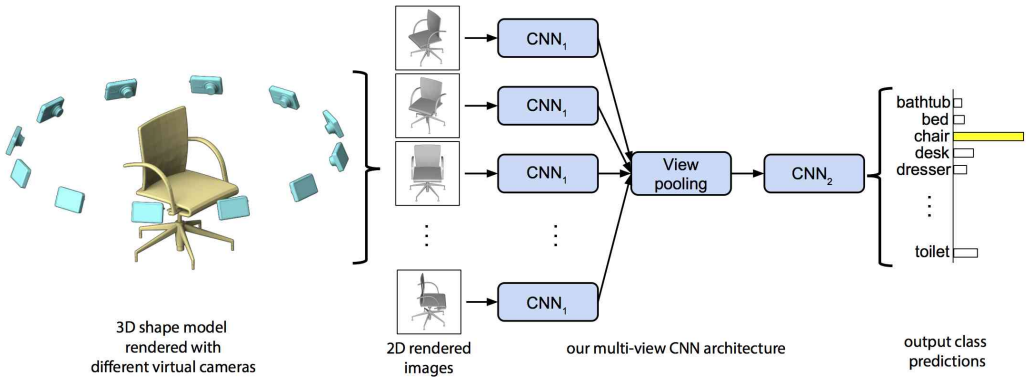


Figure 36: Multi-view CNN [60] for 3D shape classification. Figure is from original paper.

MVCNN is a powerful model regarding the performance on 3D shape classification. It is using the multiple rendered 2D images from the 3D CAD model. Features are extracted by those rendered 2D images by using a 2D CNN and the view pooling layer. Finally, based on these extracted features, it can achieve good classification results. From the results of ModelNet40 leaderboard, we can see the power of this multiple view approach.

3.3.4 RotationNet [64]

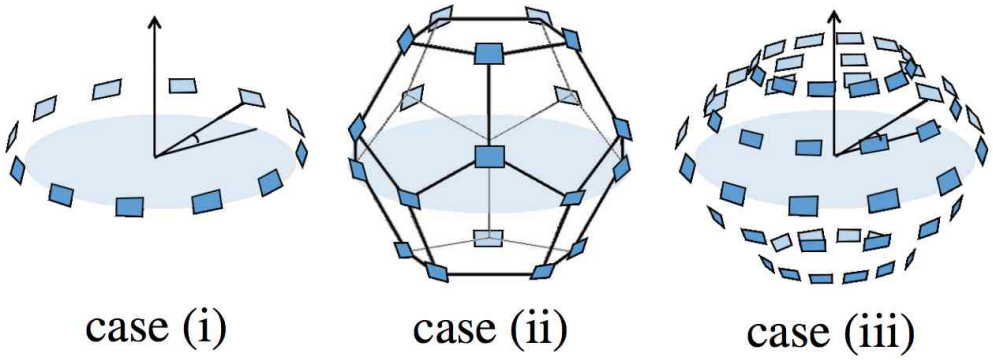


Figure 37: Illustration of three viewpoint setups considered in RotationNet [64]. A target object is placed on the center of each circle. Figure is from original paper.

Novel view point models

RotationNet is an extension of MVCNN [60]. In this paper, multiple views from different angles are explored. Three models of camera views are proposed as shown in Figure 37. The performance of case(i) (the same view points model as MVCNN [60]) and case(ii) are compared. Case (ii) achieves a better performance based on the ModelNet40 task. For the ModelNet40, the case(iii) model is not used.

Unsupervised view point estimation

RotationNet is using an unsupervised approach when using the view point information: the view point is not provided directly during the training process (the view point are also not available in most dataset such as ModelNet40) but inferred during the training process. The authors of RotationNet use the following optimization method to find the possible viewpoints.
RotationNet is defined as a differentiable multi-layer neural network $R()$. The final layer of RotationNet is the concatenation of M softmax layers, each of which outputs the category likelihood $P(\hat{y}_i, \mathbf{x_i}, v_i = j)$ where $j \in \{1, ..., M\}$ for each image $\mathbf{x_i}$. Here, $\hat{y}_i$ denotes an estimate of the object category label for $\mathbf{x_i}$. For the training of RotationNet, the set of images $\mathbf{x_i}_{i=1}^{M}$ are used

simultaneously and the following optimization problem is solved to get the estimated viewpoints [64]:

$$\max_{R,\{V_i\}_{i=1}^{M}} \prod_{i=1}^{M} P(\hat{y}_i = y|\mathbf{x_i}, v_i) \tag{1}$$

The parameters of R and latent variables $\{V_i\}_{i=1}^{M}$ are optimized to output the highest probability of y for the input of multi-view images $\mathbf{x_i}_{i=1}^{M}$.

RotationNet framework and the training process

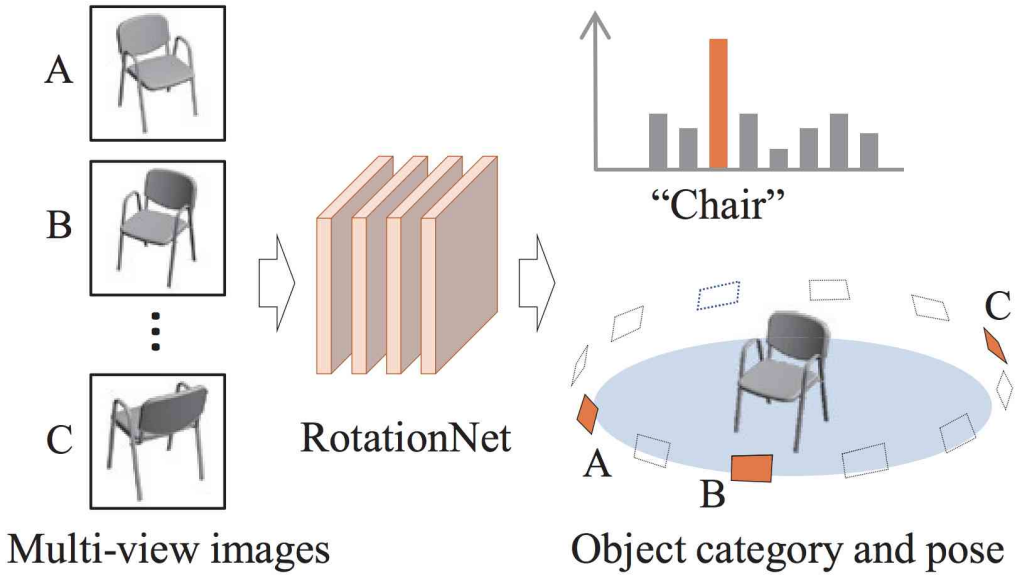


Figure 38: Illustration of the proposed method RotationNet. RotationNet takes a partial set ($\geq$ 1 images) of the full multi-view images of an object as input and predicts its object category by rotation, where the best pose is selected to maximize the object category likelihood. Here, viewpoints from which the images are observed are jointly estimated to predict the pose of the object. Figure and Caption are from original paper.

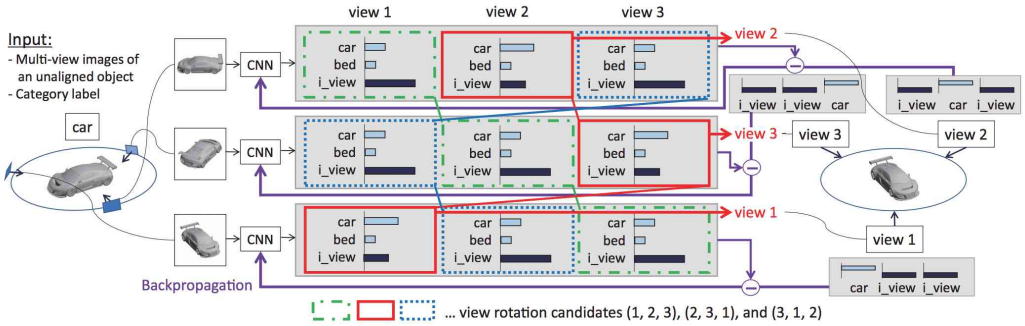


Figure 39: Illustration of the training process of RotationNet, where the number of views M is 3 and the number of categories N is 2. A training sample consists of M images of an unaligned object and its category label y. For each input image, the CNN (RotationNet) outputs M histograms with $N + 1$ bins whose norm is 1. The last bin of each histogram represents the "incorrect view" class, which serves as a weight of how likely the histogram does not correspond to each viewpoint variable. According to the histogram values, they decide which image corresponds to views 1, 2, and 3. There are three candidates for view rotation: $(1, 2, 3), (2, 3, 1)$, and $(3, 1, 2)$. For each candidate, they calculate the score for the ground-truth category ("car" in this case) by multiplying the histograms and selecting the best choice: $(2, 3, 1)$ in this case. Finally, they update the CNN parameters in a standard back-propagation manner with the estimated viewpoint variables. Note that it is the same CNN that is being used. Figure and Caption are from original paper.

The general RotationNet framework is shown in Figure 38. The training process of RotationNet is illustrated in Figure 39 where three views are provided to give the illustration. In order to obtain a stable estimation of the viewpoint by using deep neural networks, the "incorrect view" class is introduced in the training process of the RotationNet. The "incorrect view" will play the similar role as the "background" class in object detection task. The RotationNet calculates $P(\hat{y}_i = y|\mathbf{x_i}, v_i)$ by applying softmax functions to the $(N+1)$- dimensional outputs where N is the number of the categories.

3.3.5 PointNet [48]

Applications of PointNet

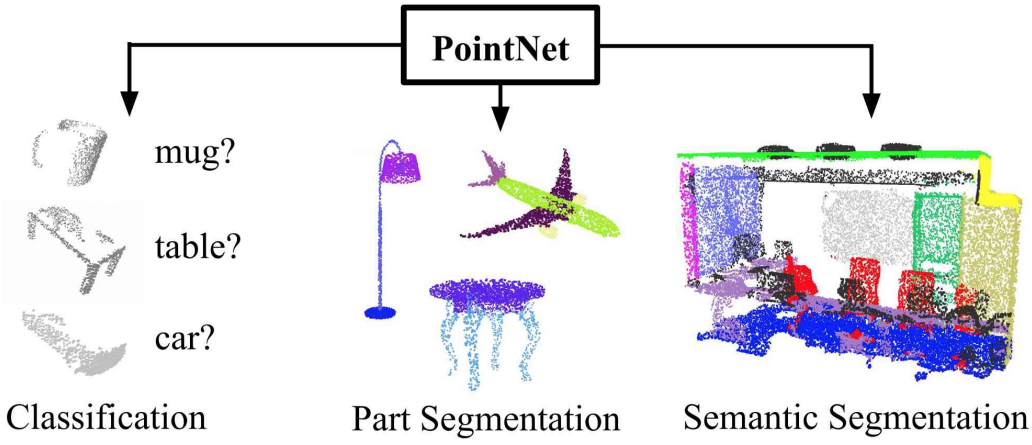


Figure 40: Applications of PointNet. Figure is from [48].

PointNet is a new work which is performing 3D vision understanding directly on the raw cloud point data. Applications of PointNet are shown in Figure 40. PointNet consumes raw point clouds (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D classification tasks.

PointNet Architecture

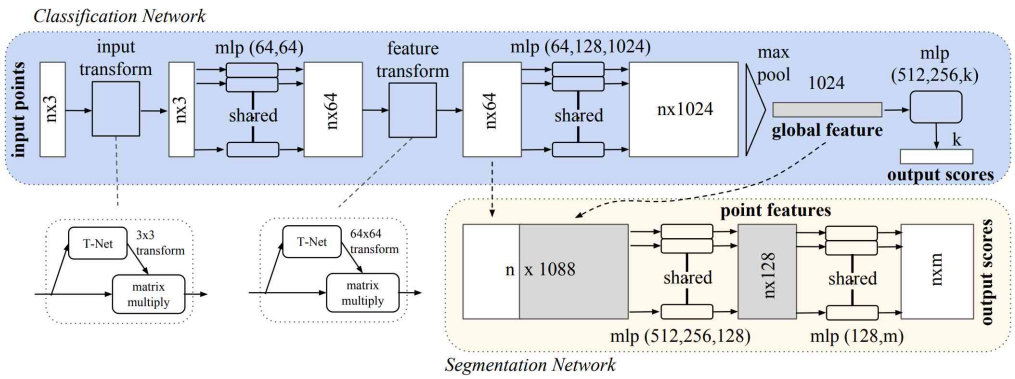


Figure 41: PointNet Architecture. The figure is from [48]. Here the "mlp" means multi-layer perceptron.

PointNet Architecture is shown in Figure 41. The classification network

takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point segmentation result.

The structure of classification network is summarized as follows.

1) takes n points as input. Each point has x, y, z three coordinate values. An input transform will be done based on the $n \times 3$ input data:
   a) T-Net provides a 3×3 transformation matrix
   b) matrix multiplication between the $n \times 3$ input data and the 3×3 transformation matrix is done to finish the **input transform**
2) Transformed $n \times 3$ data is passed to a 64×64 multi-layer perceptron(MLP) network to output a $n \times 64$ tensor
3) The $n \times 64$ tensor is fed to a **feature transform** net:
   (a) T-Net provides a 64×64 transformation matrix
   (b) matrix multiplication between the $n \times 64$ input data and the 64×64 transformation matrix is done to finish the feature transform
4) Transformed $n \times 64$ data is passed to a $64 \times 128 \times 1024$ MLP to output a $n \times 1024$ tensor
5) Global feature is generated by an element wise max pooling layer and output a 1×1024 tensor
6) Feeding the 1×1024 global feature to another $512 \times 256 \times k$ MLP to generate a classification vector with k values(corresponding to k class labels).

The segment network structure is as follows.

1) the 1×1024 global features are concatenated to every 1×64 point features. As the system totally has n points, the concatenation generates a $n \times 1088$ tensor.
2) the $n \times 1088$tensor is fed to a $512 \times 256 \times 128$ MLP to further generate a $n \times 128$ tensor.

3) the $n \times 128$ tensor is then fed to a $128 \times m$ to generate a $n \times m$ tensor (corresponding to m segmentation labels for each point)

Batch Normalization [11] is used for all layers with ReLU [65]. Dropout layers are used for the last MLP in the classification network.

T-Net in PointNet

For the semantic labelling task for a point cloud, the output is supposed to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. In order to achieve a transformation invariant learning representation for the raw cloud point, a T-Net is introduced in PointNet as shown in Figure 41. The main functionality of this T-Net is predicting an affine transformation matrix.

Performance of PointNet

The results of the PointNet on ModelNet40 are shown in Table 7.

More results of PointNet

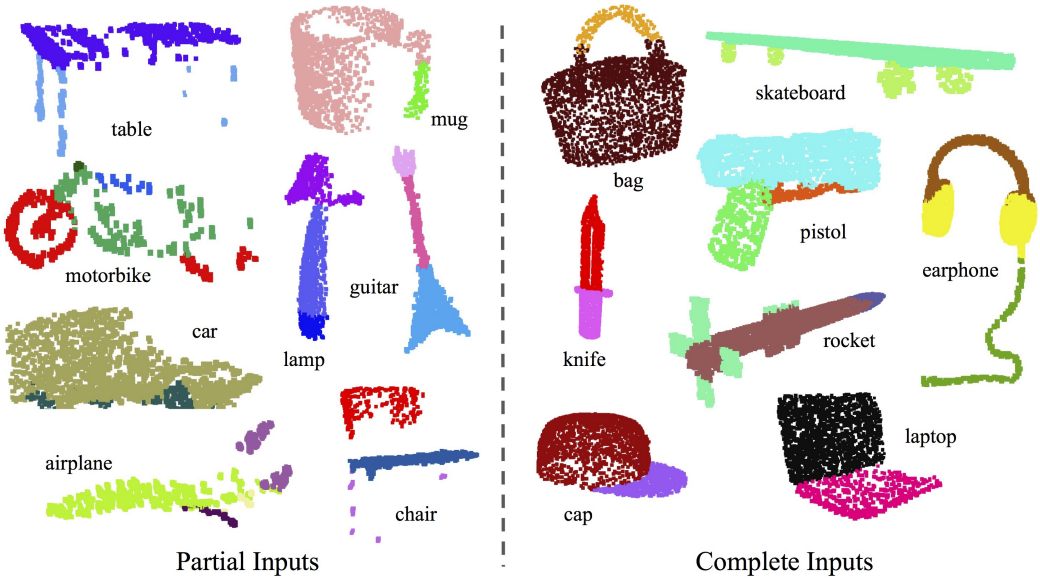


Figure 42: Part Segmentation Results [48]. The CAD part segmentation results are visualized across all 16 object categories. Both results for partial simulated Kinect scans (left block) and complete ShapeNet CAD models (right block) are shown. Figure and Caption are adjusted from original paper.

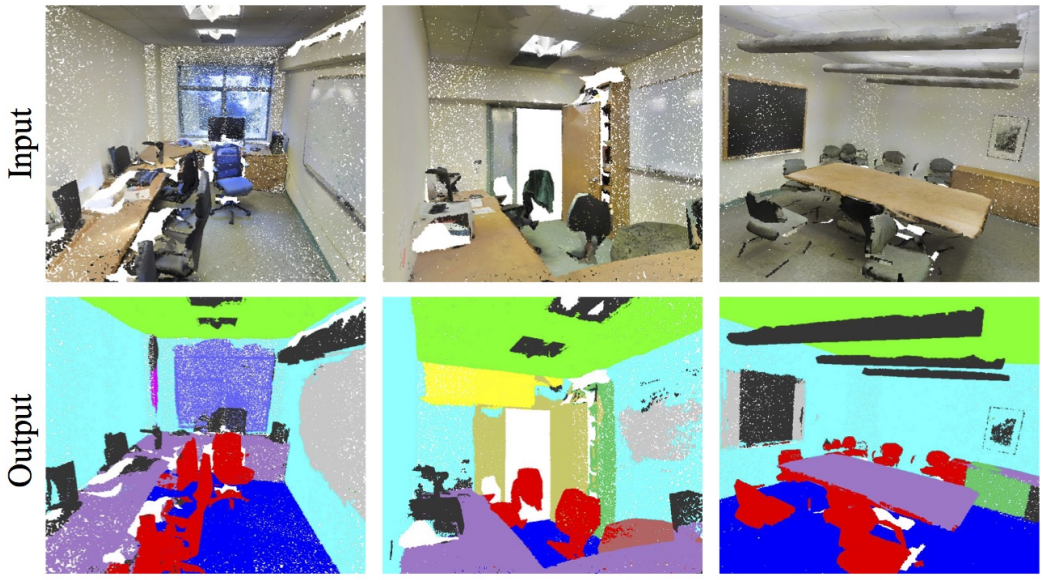


Figure 43: Semantic Segmentation Results. Top row is input point cloud with color. Bottom row is output semantic segmentation result (on points) displayed in the same camera viewpoint as input. Figure and Caption are from [48].

Part segmentation and semantic segmentation results are shown in Figure 42 and 43.

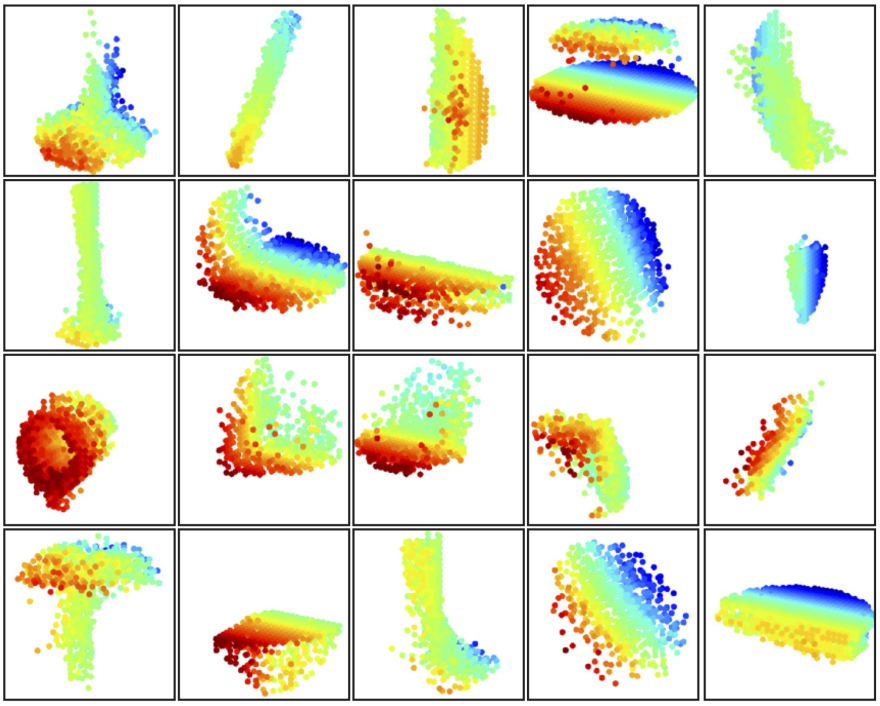


Figure 44: 3D point cloud patterns learned from the first layer kernels. The model is trained for ModelNet40 shape classification(20 out of the 128 kernels are randomly selected). Color indicates point depth (red is near, blue is far). Figure and Caption are from [66].

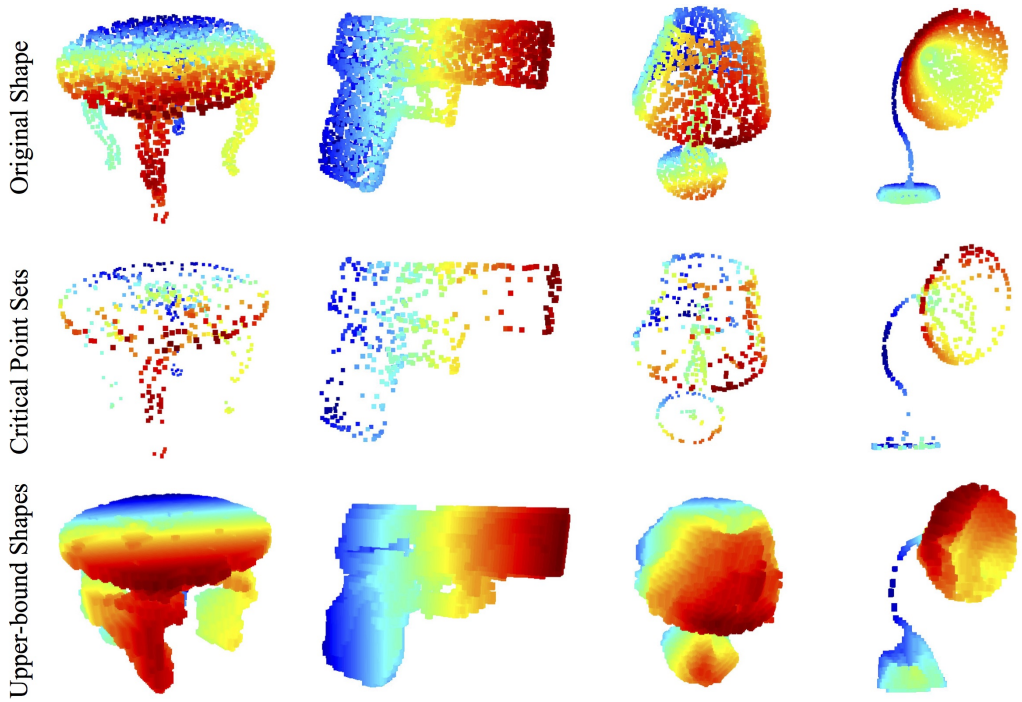


Figure 45: Visualizing Critical Points and Shape Upper-bound. The first row shows the input point clouds. The second row show the critical points picked by PointNet. The third row shows the upper-bound shape for the input – any input point sets that falls between the critical point set and the upper-bound set will result in the same classification result. Figure and Caption are from [48].

3D point cloud patterns learned from the PointNet are shown in Figure 44. This result is from [66], an improved version of PointNet.
The critical points learned by the PointNet and Shape Upper-bound of some objects are shown in Figure 45.

3.4 Detection

Similar to 2D-image based object systems, most 3D systems are also using the two-stage methods to do the 3d object detection: first, generate proposals and then do detection. At the same time, the unique properties of the 3D systems, such as different data representation and the availability of both 2D and 3D images, make the 3D detection framework more complicated and more interesting. We will discuss the datasets used for detection and main

works in indoor and outdoor scenarios next.

3.4.1 Datasets used for 3D object detection

Figure 46: Example images with annotation from the SUN-RGBD dataset [67].

Several datasets are used as the test bed of different detection algorithms. **NYU Depth Dataset V2** [**68**] has 1449 densely labeled pairs of aligned RGB and depth images from Kinect video sequences for a variety of indoor scenes.
SUN-RGBD dataset [**67**] has 19 object categories for predicting a 3D bounding box in real world dimension. It has 10,355 RGB-D images for training set and 2860 for testing set. The whole dataset is densely annotated and includes 146,617 2D polygons and 64,595 3D bounding boxes with accurate object orientations. Several examples of the SUN-RGBD dataset is shown in Figure 46.
KITTI [**69**] 3D object detection dataset consists of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80,256 labeled objects. The result of 3D object detection performance is evaluated by using the PASCAL criteria. Far objects are thus filtered based on their bounding box height in the image plane. Since only objects appearing on the image plane are labeled, objects in non car areas do not count as false positives. For cars a 3D bounding box overlap of 70% is required, while for pedestrians and cyclists the requirement is 50%. Difficulties are defined

as follows:

Easy: minimum bounding box height: 40 Pixel, Maximum occlusion level: Fully visible, Maximum truncation: 15 %
Moderate: minimum bounding box height: 25 Pixel, Maximum occlusion level: Partly occluded, Maximum truncation: 30 %
Hard: minimum bounding box height: 25 Pixel, Maximum occlusion level: Difficult to see, Maximum truncation: 50 %

Examples of the labelled object instances from the training set of different difficulties are shown in Figure 47

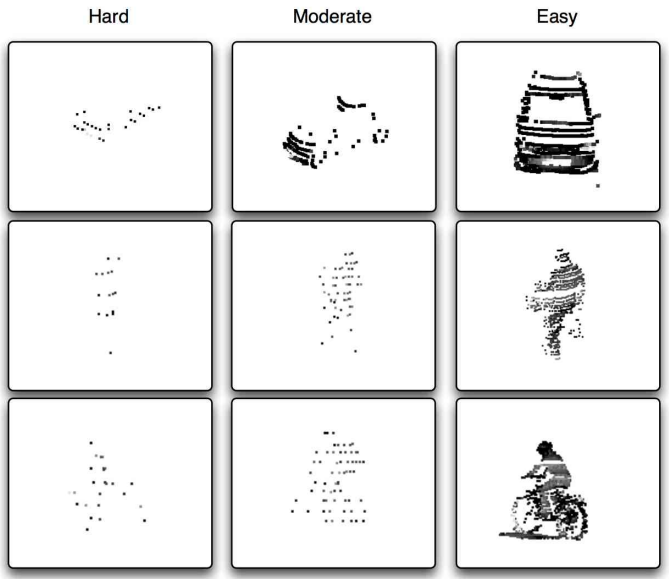


Figure 47: Examples of labelled object instances from the training set of different difficulties. Left column: hard. Middle column: moderate. Right column: easy. The figure is from [70].

3.4.2 Detection outputs

Type	output
Class	label confidence value
Bounding Box	Image plane BBOX BEV BBOX 3D BBOX
	orientation

Table 8: Different detection outputs based on 3D image data

Figure 48: Examples for the bounding box for each detection task: left is the detection bounding box for the image plane, middle is the bbox for the BEV and right is 3D detection bounding box.

For 3D understanding, the detection output is more complicated than the 2D case. It contains the output of the class and the bounding box. The class output is similar to the 2D detection task. For the bounding box detection, it can output the image plane bounding box, bird's eye view(BEV) bounding box and 3D bounding box. Meanwhile, for the 2D object detection, the bounding boxes are axis aligned, however, the BEV and 3D bounding boxes are not axis aligned. Finally, the orientation of BEV and 3D bounding box will also be detected in some tasks. The detection outputs are summarized in Table 8. Examples of each output are shown in Figure 48. In this survey, we mainly focus on the 3D bounding box detection.

3.4.3 Comparison by number of stages used

# of stage	System
Two-stage	Deep Sliding Shape [71]
	MV3D [72]
	AVOD [73]
	VoxelNet [74]
Three-stage	F-PointNet [75]

Table 9: Number of stages used for the different detection frameworks.

Most frameworks are based on two-stage methods where the proposals are firstly generated and then the detection will be done based on the proposals. F-PointNet [75] is a special case. It has three stages: first, get a frustum proposal from the detected 2D bounding based on the RGB image. The difference of the frustum proposal with the proposal from two-stage methods is it has a class label. second, 3D Instance segmentation is done based on the proposal. Finally, a 3D box is estimated based on the segmentation results.

3.4.4 The 3D bounding box encoding methods

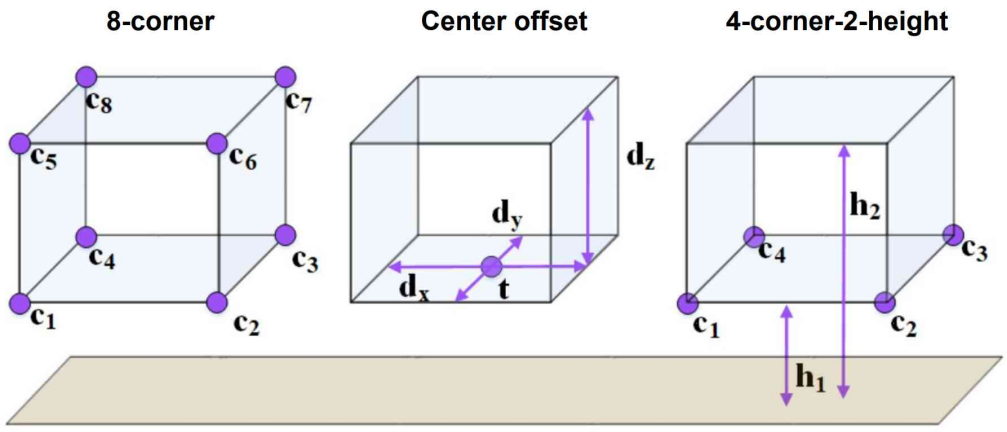


Figure 49: Different 3D bounding box encoding methods. The figure is adjusted from [73]

In order to represent a 3D bounding box, different methods are proposed, such as 8-corner method, 3D center offset method and 4-corner-2-height

method as shown in Figure 49.

3.4.5 Comparison by input data, feature representation, BBOX encoding method and CNN kernel used

Input data	**System**
Monocular	Mono3D [76]
Stereo	3DOP [77]
Depth/LiDAR only	VoxelNet [74] MV3D [72]
Monocular+ Depth/LiDAR	Deep Sliding Shape [71] MV3D [72] AVOD [73] F-PointNet [75]
Stereo+ Depth/LiDAR	None

Table 10: Input data for the whole system of different 3D bounding box detection systems.

Different input data can be used to detect the 3D bounding box, such as the Monocular image, Stereo image and depth/LiDAR image. The detection system organized by the input data type is given in Table 10. Generally speaking, the 2D image only system including both the monocular and stereo image will perform worse than the 3D only or 2D+3D system. The comparison of the 2D image only and the 2D+3D system is provided in Figure 63. In the following section we will focus on the 3D only or 2D+3D detection systems.

3D BBOX Detection				**System**	**Proposals generation**			
Input data	features	BBOX encoding method	2D or 3D cnn		Input data	features	BBOX encoding method	2D or 3D cnn
Depth/LiDAR only	7-feature per point	3D center offset $+\theta$	3D +2D	VoxelNet [74]	LiDAR	7-feature	3D center offset $+\theta$	3D +2D
	FV+ BEV	8-corner	2D	MV3D [72]	LiDAR	FV+ BEV	8-corner	2D
Monocular+ Depth/LiDAR	voxel, + RGB	3D center offset	Depth: 3D image:2D	DeepSliding [71]	Monocular + Depth	voxel, +RGB projected to each cloud point	3D center offset	3D
	FV+ BEV +RGB	8-corner	2D	MV3D [72]	LiDAR	BEV	8-corner	2D
	BEV +RGB	4-corner-2-height	2D	AVOD [73]	LiDAR	BEV +RGB	4-corner-2-height	2D
	point+image	3D center offset $+\theta$	T-NET	F-PointNet [75]	Monocular	RGB	2D center offset	2D

Table 11: Input data for the whole system of different 3D bounding box detection systems.

A comprehensive comparison of the input data, the feature representation of input data and the bounding box encoding methods for both the proposals and the final 3D bounding box detections is provided in Table 11. DeepSliding [71] and VoxelNet [74] are using the 3D convolutional neural network to do the proposals generation and the bounding box detection. MV3D [72], AVOD [73] are projecting the depth or LiDAR data to a 2D similar images and are using a 2D CNN to do the proposal generation and bounding box detection. The relationship between MV3D and AVOD is explained later. F-PointNet [75] is using 2D RGB images to help generate the proposals and it is a special framework. At the same time, the different proposal generation method will have an influence on the application scenario of those frameworks, to be discussed later.

3.4.6 Comparison by performance

The performance comparison of the different systems is provided in Table 16 , 17 and 18 for the outdoor scenario based on the KITTI [69] dataset and in Table 12 for the indoor scenario based on the SUN-RGBD dataset [67].

	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	Runtime	mAP
DeepSliding [71]	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	19.55s	42.1
F-PointNet [75]	43.3	81.1	33.3	64.2	24.7	32.0	58.1	61.1	51.1	90.9	0.12s	54.0

Table 12: 3D object detection AP on SUN-RGBD val set. Evaluation metric is average precision with 3D IoU threshold 0.25 as proposed by [67].

3.4.7 Comparison by application scenario

Scenario	dataset	X(meter)	Y(meter)	Z(meter)
indoor	SUN RGB-D	5.2	6	2.5
outdoor	KITTI	70.4	80	4

Table 13: Indoor(data is collected from Deep Sliding Shape [71]) vs. outdoor(data is collected from VoxelNet [74])

3D object detection systems can be categorized by the supported application scenarios: indoor only, outdoor only or both. There are two main differences between indoor and outdoor scenarios: first, the range of indoor is small and

of outdoor is large. A comparison of the indoor range and outdoor range based on two typical datasets is shown in Table 13. Second, as the distribution of the outdoor objects is more sparse and the categories of interesting objects of the outdoor scenarios is less compared with indoor scenarios, the outdoor scenarios can use BEV to generate the proposals and then do the detection. However, the indoor scenarios, generation only based on BEV will get a bad performance since there might be multiple objects in the vertical direction.

BEV only proposal generation algorithms such as MV3D [72] do not performance well indoors. At the same time, as the space is too large for the outdoor scenario, some 3D CNN based algorithms, such as Deep Sliding Shape [71], can work well for indoors but may have a high possibility to fail for the outdoor scenario without adjustment.

Scenario	System
Indoor	Deep Sliding Shape [71]
Outdoor	MV3D [72]
	AVOD [73]
	VoxelNet [74]
Both	F-PointNet [75]

Table 14: Object detection system based on 3D image data for different application scenarios.

In the rest of this section we will introduce several of the latest papers organized by different application scenarios as shown in Table 14.

3.4.8 Indoor

Some 3D object detection systems can work well for the indoor scenarios are introduced in this part.

Deep Sliding Shapes [71]

The TSDF 3D Representation

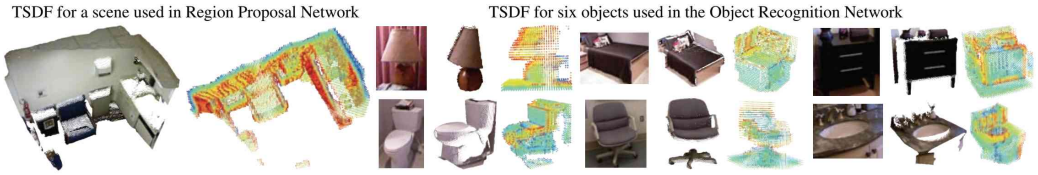


Figure 50: Visualization of TSDF Encoding. Only the TSDF values when close to the surface visualized . Red indicates the voxel is in front of surfaces and blue indicates the voxel is behind the surface. The resolution is $208 \times 208 \times 100$ for the Region Proposal Network, and $30 \times 30 \times 30$ for the Object Classification Network. Figure and Caption are from [71].

A directional Truncated Signed Distance Function(TSDF) is used to encode 3D shapes. The 3D space is divided into 3D voxel grid and the value in each voxel is defined to be the shortest distance between the voxel center and the surface from the input depth map. To encode the direction of the surface point, a directional TSDF stores a three-dimensional vector $[dx, dy, dz]$ in each voxel in order to record the distance in three directions to the closest surface point instead of a single distance value [71]. The example of the directional TSDF for the RPN and detection network is shown in Figure 50.

The 3D RPN

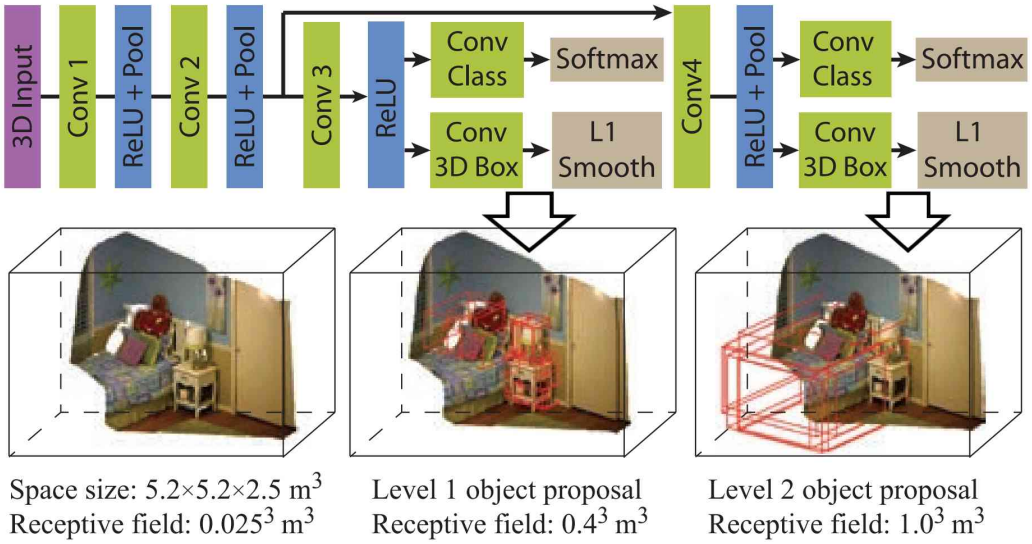
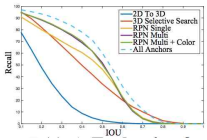


Figure 51: 3D Region Proposal Network: Taking a 3D volume from depth as input, a fully convolutional 3D network extracts 3D proposals at two scales with different receptive fields. Figure and Caption are from [71].

Deep Sliding Shapes [71] is inspired by the Faster RCNN [23] framework. The proposals are generated by a CNN based RPN. The RPN is shown in Figure 51. The network structure shown in Figure 51:

- is using 3D CNN to do the feature extraction.
- has two level region proposals since the variation of the physical size of the 3D object is large.



																				Recall	ABO	#Box
2D To 3D	41.7	53.5	37.9	22.0	26.9	46.2	42.2	11.8	47.3	33.9	41.8	12.5	45.8	20.7	49.4	55.8	54.1	15.2	50.0	34.4	0.210	2000
3D Selective Search	79.2	80.6	74.7	**66.0**	66.5	92.3	80.9	**53.9**	89.1	**89.8**	**83.6**	**45.8**	85.4	75.9	83.1	85.5	80.9	**69.7**	83.3	74.2	0.409	2000
RPN Single	87.5	98.7	70.1	15.6	95.0	100.0	93.0	20.6	94.5	49.2	49.1	12.5	100.0	34.2	81.8	94.9	93.3	57.6	96.7	75.2	0.425	2000
RPN Multi	100.0	**98.7**	**73.6**	42.6	94.7	**100.0**	92.5	21.6	**96.4**	78.0	69.1	37.5	**100.0**	75.2	97.4	97.1	96.4	66.7	100.0	84.4	0.460	2000
RPN Multi Color	**100.0**	98.1	72.4	42.6	**95.0**	**100.0**	**93.0**	19.6	**96.4**	79.7	76.4	37.5	**100.0**	**79.0**	**97.4**	**97.1**	**95.4**	57.6	**100.0**	**84.9**	**0.461**	2000
All Anchors	100.0	98.7	75.9	50.4	97.2	100.0	97.0	45.1	100.0	94.9	96.4	83.3	100.0	91.2	100.0	97.8	96.9	84.8	100.0	91.0	0.511	107674

Figure 52: Evaluation for 3D Object Proposal. "All Anchors" shows the performance upper bound when using all anchors. "RPN Multi" means two-level proposal RNP network. "RPN Multi Color" means the RGB color information is projected to each voxel. Figure and Caption are from [71].

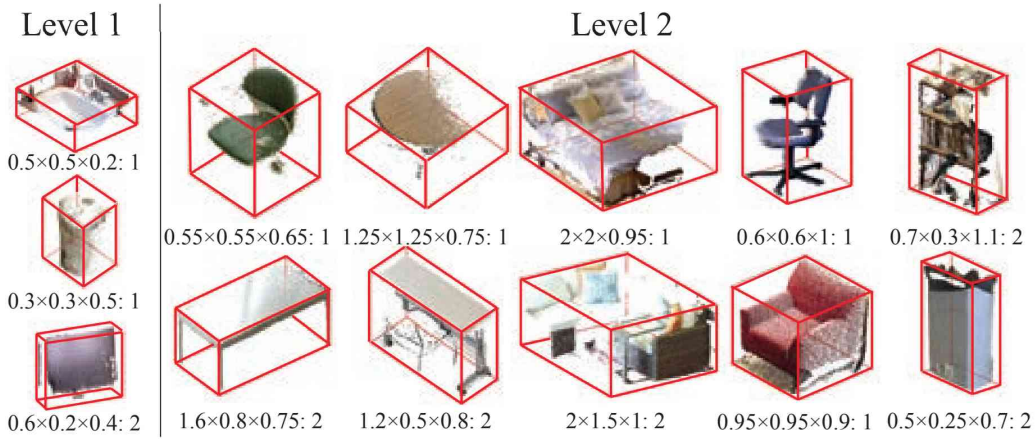


Figure 53: List of All Anchors Types. The subscripts show the $width*depth*height$ in meters, followed by the number of orientations for this anchor after the colon. Figure and Caption are from [71].

3D Selective Search(SS) is also proposed to compare the performance with the CNN based on RPN. The result is shown in Figure 52. From the result we can see similarities to the conclusion from Faster RCNN [23]: the CNN based RPN has a better performance compared with the traditional SS method for the 3D proposals generation. Anchors used for the RPN are shown in Figure 53.

Features fed to the RPN is from both the depth channel and the RGB image. A directional Truncated Signed Distance Function(TSDF) encoding is used to change the depth channel information to a 3D voxel grid data representation. Also the RGB color is projected to each voxel to improve the proposal results as shown in Figure 52.

The detection network

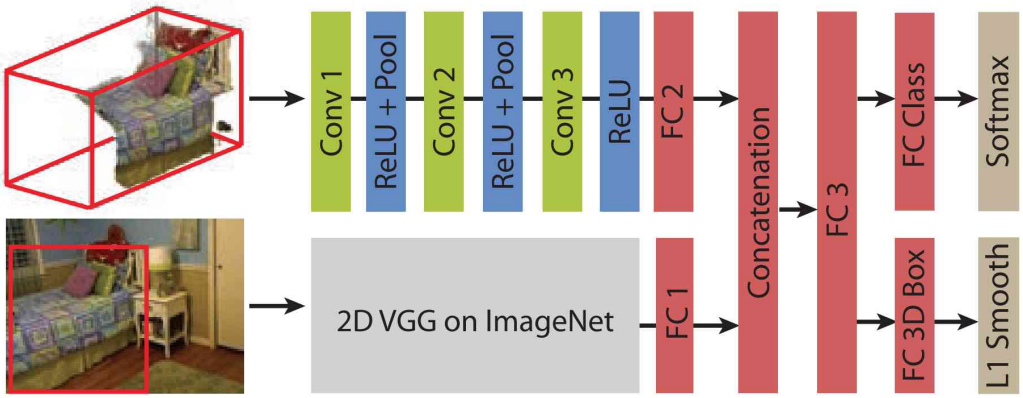


Figure 54: Joint Object Classification Network: For each 3D proposal, 3D volume from depth is fed to a 3D ConvNet, and feed the 2D color patch (2D projection of the 3D proposal) to a 2D ConvNet, to jointly learn object category and 3D box regression. Figure and Caption are from [71].

The detection network is shown in Figure 54. The detection network is using early fusing model to fuse the depth and RGB images together. The depth image is using the TSDF which is the same as RPN. The feature of 2D image is extracted based on the ImageNet pre-trained VGG network. The difference between the Faster RCNN [23] and Deep Sliding Shape [71] is: the RPN and detection network are trained separately in Deep Sliding Shape while in Faster RCNN the two networks share the convolutional layers.

Another important part for the Deep Sliding Shape is that it is using the different resolutions for the RPN and detection networks. The comparison of this difference is shown in Table 15.

network	resolution	3D CNN input data shape	physical size
RPN	$0.025 \times 0.025 \times 0.025$	$208 \times 208 \times 100$	$5.2 \times 6.0 \times 2.5$
Detection for bed	$0.067 \times 0.067 \times 0.032$	$30 \times 30 \times 30$	$2.0 \times 2.0 \times 0.95$
Detection for trash can	$0.010 \times 0.010 \times 0.012$	$30 \times 30 \times 30$	$0.3 \times 0.3 \times 0.5$

Table 15: Resolution and shape comparison between the RPN and detection network. The detection network has a fixed input shape which is $30 \times 30 \times 30$ and the resolution is decided by the proposed region size and the input shape. In this table, an anchor of the bed and an anchor of the trash can are used as examples of proposal's physical size to make the comparison.

The results

poposal	algorithm																					mAP
3D SS	dxdydz no bbreg	43.3	55.0	16.2	23.1	3.4	10.4	17.1	30.7	10.9	35.4	20.3	41.2	47.2	25.2	43.9	1.9	1.6	0.1	9.9	23.0
	dxdydz	52.1	60.5	19.0	30.9	2.2	15.4	23.1	36.4	19.7	36.2	18.9	52.5	53.7	32.7	56.9	1.9	0.5	0.3	8.1	27.4
RPN	dxdydz no bbreg	51.4	74.8	7.1	51.5	15.5	22.8	24.9	11.4	12.5	39.6	15.4	43.4	58.0	40.7	61.6	0.2	0.0	1.5	2.8	28.2
	dxdydz no size	59.9	78.9	12.0	51.5	15.6	24.6	27.7	12.5	18.6	42.3	15.1	59.4	59.6	44.7	62.5	0.3	0.0	1.1	12.9	31.5
	dxdydz	59.0	80.7	12.0	59.3	15.7	25.5	28.6	12.6	18.6	42.5	15.3	59.5	59.9	45.3	64.8	0.3	0.0	1.4	13.0	32.3
	tsdf dis	61.2	78.6	10.3	61.1	2.7	23.8	21.1	25.9	12.1	34.8	13.9	49.5	61.2	45.6	70.8	0.3	0.0	0.1	1.7	30.2
	dxdydz+rgb	58.3	79.3	9.9	57.2	8.3	27.0	22.7	4.8	18.8	46.5	14.4	51.6	56.7	45.3	65.1	0.2	0.0	4.2	0.9	30.1
	proj dxdydz+img	58.4	81.4	20.6	53.4	1.3	32.2	36.5	18.3	17.5	40.8	19.2	51.0	58.7	47.9	71.4	0.5	0.2	0.3	1.8	32.2
	dxdydz+img+hha	55.9	83.0	18.8	63.0	17.0	33.4	43.0	33.8	16.5	54.7	22.6	53.5	58.0	49.7	75.0	2.6	0.0	1.6	6.2	36.2
	dxdydz+img	62.8	82.5	20.1	60.1	11.9	29.2	38.6	31.4	23.7	49.6	21.9	58.5	60.3	49.7	76.1	4.2	0.0	0.5	9.7	36.4

Figure 55: Control Experiments on NYUv2 Test Set. Not working: box (too much variance), door (planar), monitor and tv (no depth). Figure and Caption are from [71].

The performance of 3D region proposal and object detection algorithm of Deep Sliding Shapes is evaluated on the standard NYUv2 dataset [78] and SUN RGB-D dataset [67]. The result of the NYUv2 Test Set is shown in Figure 55. From Row [RPN: dxdydz] to Row [RPN: dxdydz+img] in Figure 55, the performance of different feature encodings is compared and the main conclusions are given below: (1) TSDF with directions encoded is better than single TSDF distance ([dxdydz] vs. [tsdf dis]). (2)Directly encoding color on 3D voxels is not as good as using 2D image VGGnet ([dxdydz+rgb] vs. [dxdydz+img]), probably because the latter one can preserve high frequency signal from images. (3) Adding HHA[2] does not help, which indicates the depth information from HHA is already exploited by the 3D representation([dxdydz+img+hha] vs. [dxdydz+img]).

																			Recall	ABO	#Box
3D SS	78.8	87.2	72.8	72.2	65.5	86.1	75.1	65.0	70.0	87.1	67.5	53.1	68.1	82.8	86.8	84.4	85.0	69.2	94.0	72.0	0.394	2000
RPN	98.1	99.1	79.5	51.5	93.3	89.2	94.9	24.0	87.0	79.6	62.0	41.2	96.2	77.9	96.7	97.3	96.7	63.3	100.0	88.7	0.485	2000

Figure 56: Evaluation for region proposal generation on SUN RGB-D test set [71].

[2]HHA is proposed in [79]. In [79] the depth image is encoded with three channels at each pixel: **H**orizontal disparity, **H**eight above ground, and the **A**ngle the pixel's local surface normal makes with the inferred gravity direction.

		mAP
Sliding Shapes	- 42.09 - 33.42 - - - - - - - - - - 23.28 25.78 - 61.86	-
Deep Sliding Shapes	44.2 78.8 11.9 1.5 61.2 4.1 20.5 0.0 6.4 20.4 18.4 0.2 15.4 13.3 32.3 53.5 50.3 0.5 78.9	26.9

Figure 57: Evaluation for 3D object detection on SUN RGB-D test set [71].

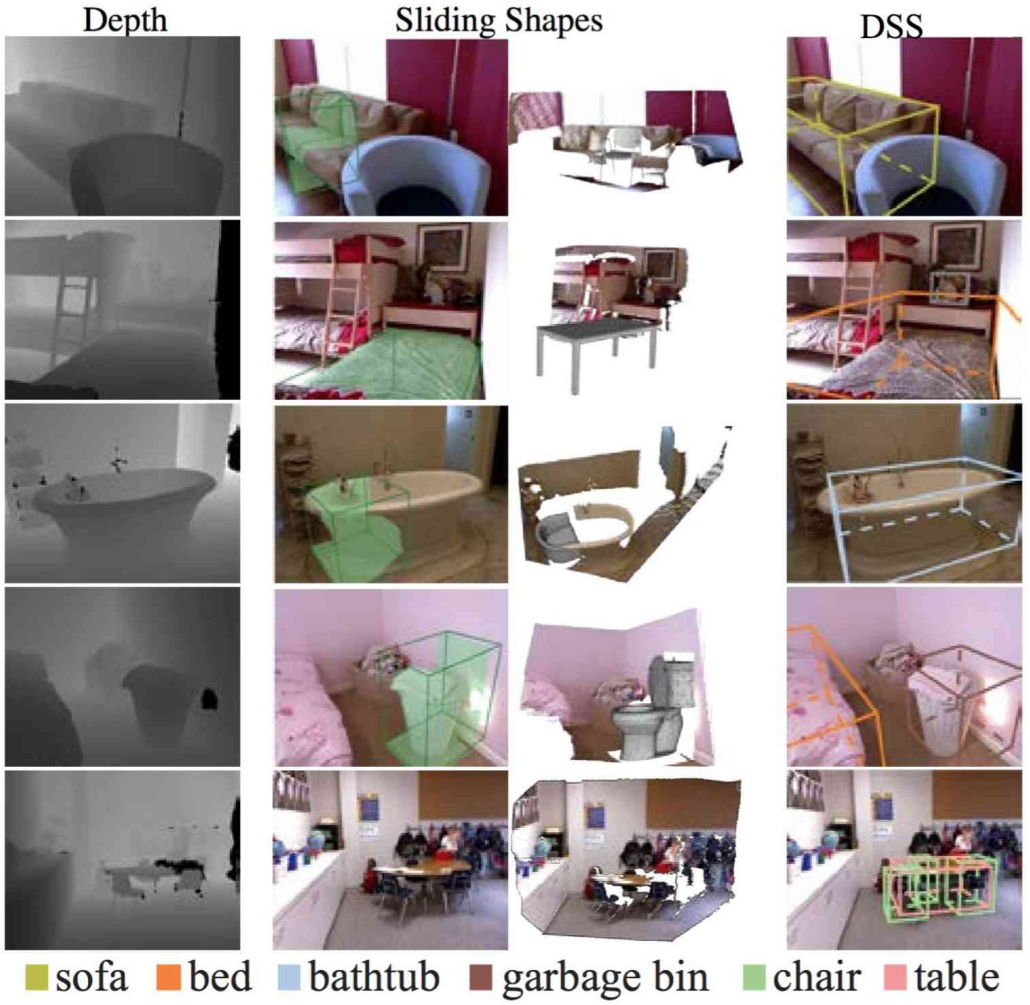


Figure 58: Comparison with Sliding Shapes. DSS algorithm is able to better use shape, color and contextual information to handle more object categories, solve the ambiguous cases, and detect objects with a typical size. Figure and Caption are adjusted from [71].

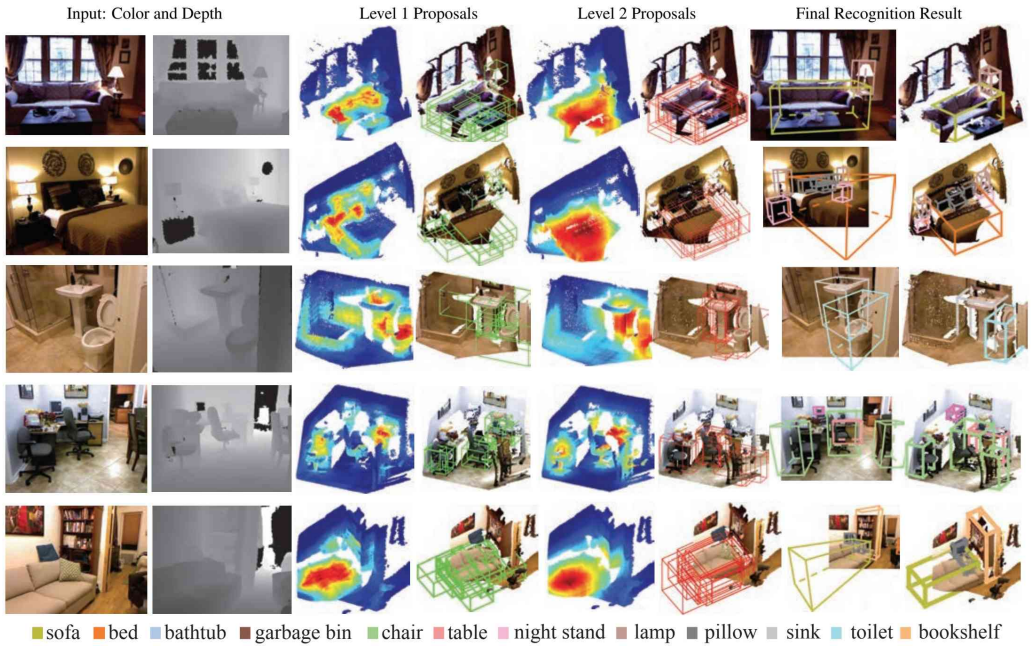


Figure 59: Examples for Detection Results. For the proposal results, The heat map for the distribution of the top proposals (red is the area with more concentration) and a few top boxes after NMS are shown. For the classification results, the 3D detection can estimate the full extent of 3D both vertically (e.g. bottom of a bed) and horizontally (e.g. full size sofa in the last row). Figure and Caption are from [71].

Results on the SUN RGB-D dataset are shown in Figure 57 and 58. The comparison of the detection results of DSS with the F-PointNet [75] is shown in Table12. From the results, we can see that RPN has a better performance than Selective Search. Deep sliding shape has a better performance than their previous work Sliding Shape [80]. Sliding Shapes [80] is a 3D object detector that runs sliding windows in 3D to directly classify each 3D window by using SVM. Figure 58 shows side-by-side comparisons to Sliding Shapes. Figure 59 shows some general example for the Deep Sliding Shape system.

3.4.9 Outdoor

KITTI [69] is an important outdoor dataset. For the outdoor scenario, the algorithms used to do the outdoor object detection will be studied by the leaderboard of the 3D object detection for the KITTI data set by April, 2018.

The leaderboard of 3D object detection for the KITTI dataset

Method	Moderate	Easy	Hard	Runtime	Environment
AVOD-FPN [73]	71.88 %	81.94 %	66.38 %	0.1 s	Titan X (Pascal)
F-PointNet [75]	70.39 %	81.20 %	62.19 %	0.17 s	GPU@3.0 Ghz (Python)
DF-PC_CNN [73]	66.22 %	80.28 %	58.94 %	0.5 s	GPU@3.0 Ghz (Matlab + C/C++)
AVOD [73]	65.78 %	73.59 %	58.38 %	0.08 s	Titan X (pascal)
VoxelNet [74]	65.11 %	77.47 %	57.73 %	0.03 s	GPU@2.5 Ghz (Python + C/C++)
MV3D [72]	62.35 %	71.09 %	55.12 %	0.36 s	GPU@2.5 Ghz (Python + C/C++)
MV3D (LiDAR) [72]	52.73 %	66.77 %	51.31 %	0.24 s	GPU@2.5 Ghz (Python + C/C++)
F-PC_CNN [73]	48.07 %	60.06 %	45.22 %	0.5 s	GPU@3.0 Ghz (Matlab + C/C++)

Table 16: 3D object detection evaluation result for **Car** based on the KITTI evaluation server. All methods are ranked based on the moderately difficult results. The table is adjusted from [81]. The top 8 are listed here.

Method	Moderate	Easy	Hard	Runtime	Environment
F-PointNet [75]	44.89 %	51.21 %	40.23 %	0.17 s	GPU@3.0 Ghz (Python)
AVOD-FPN [73]	39.00 %	46.35 %	36.58 %	0.1 s	Titan X (Pascal)
VoxelNet [74]	33.69 %	39.48 %	31.51 %	0.03 s	GPU@2.5 Ghz (Python + C/C++)
AVOD [73]	31.51 %	38.28 %	26.98 %	0.08 s	Titan X (pascal)

Table 17: 3D object detection evaluation result for **Pedestrian** based on the KITTI evaluation server. All methods are ranked based on the moderately difficult results. The table is adjusted from [81]. The top 4 are listed here.

Method	Moderate	Easy	Hard	Runtime	Environment
F-PointNet [75]	56.77 %	71.96 %	50.39 %	0.17 s	GPU@3.0 Ghz (Python)
VoxelNet(LiDAR) [74]	48.36 %	61.22 %	44.37 %	0.03 s	GPU@2.5 Ghz (Python + C/C++)
AVOD-FPN [73]	46.12 %	59.97 %	42.36 %	0.1 s	Titan X (Pascal)
AVOD [73]	44.90 %	60.11 %	38.80 %	0.08 s	Titan X (pascal)

Table 18: 3D object detection evaluation result for **Cyclist** based on the KITTI evaluation server. All methods are ranked based on the moderately difficult results. The table is adjusted from [81]. The top 4 are listed here.

MV3D [72] and AVOD [73]

The comparison of the input data and feature encoding for the RPN and detection part of MV3D [72] and AVOD [73] can be found in Table 10. Since MV3D is using the BEV only to propose the region candidates, it performances good on big objects such as cars. However, it will not have a good performance for small object detection such as pedestrians and also for the indoor scenario. AVOD [73] further improves the MV3D [72] by using BEV and RGB image to propose the region candidates.

Framework of MV3D

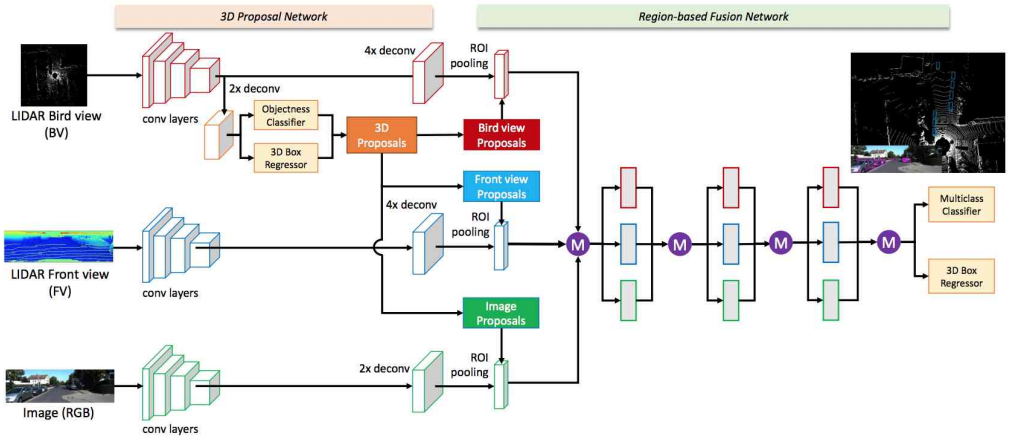


Figure 60: Multi-View 3D object detection network (MV3D) [72] .

The two-stage detection framework of the MV3D is shown in Figure 60

including the RPN and detection two stages. The network takes the bird's eye view and front view of LiDAR point cloud as well as an image as input. It first generates 3D object proposals from bird's eye view map and projects them to three views: BEV, FV from LiDAR and Image plane view. A deep fusion network is used to combine region-wise features obtained via ROI pooling for each view. The fused features are used to jointly predict object class and do oriented 3D box regression [72].

Input features overview for MV3D

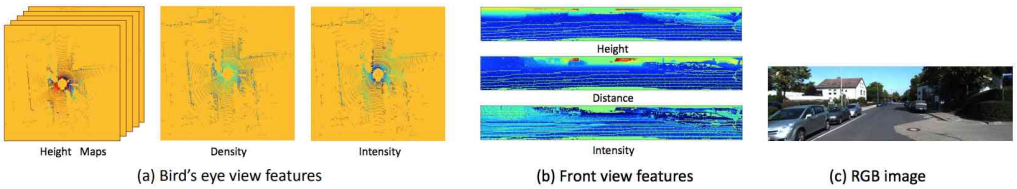


Figure 61: Input features of the MV3D network: BEV, FV and RGB. Figure is from original paper.

MV3D projects 3D point cloud to the bird's eye view and the front view. Figure 61 visualizes the point cloud representation and input image. The detail of the BEV and FV representation is given next.

BEV features for MV3D

The bird's eye view representation is encoded by height, intensity and density. The point cloud is projected into a 2D grid with resolution of $0.1m$. For each cell, the height feature is computed as the maximum height of the points in the cell. To encode more detailed height information, the point cloud is divided equally into M slices. A height map is computed for each slice. The intensity feature is the reflectance value of the point which has the maximum height in each cell. The point cloud density indicates the number of points in each cell. So it has $(M + 2)$ channel features [72] . MV3D uses point cloud in the range of $[0, 70.4] \times [-40, 40]$ meters in the X and Y dimensions. The size of the input features is $704 \times 800 \times (M + 2)$. The value of M is not provided in [72]. The length of Z dimension is also not provided. If we suppose that for the length of Z dimension is 2.5 meters as in AVOD [73] and the resolution of the Z dimension is also 0.1 meters, then the size of the

input feature for the BEV will be $704 \times 800 \times 27$.

FV features for MV3D

MV3D projects the FV into a cylinder plane to generate a dense front view map as in VeloFCN [82]. The front view map is encoded with three-channel features, which are height, distance and intensity as shown in Figure 61. Since KITTI uses a 64-beam Velodyne laser scanner, the size of map for the front view is 64×512.

Performance of MV3D

Data	AP_{3D} (IoU=0.5)			AP_{loc} (IoU=0.5)			AP_{2D} (IoU=0.7)		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
FV	67.6	56.30	49.98	74.02	62.18	57.61	75.61	61.60	54.29
RGB	73.68	68.86	61.94	77.30	71.68	64.58	83.80	76.45	73.42
BV	92.30	85.50	78.94	92.90	86.98	86.14	85.00	76.21	74.80
FV+RGB	77.41	71.63	64.30	82.57	75.19	66.96	86.34	77.47	74.59
FV+BV	95.19	87.65	80.11	95.74	88.57	88.13	88.41	78.97	78.16
BV+RGB	**96.09**	88.70	80.52	**96.45**	89.19	80.69	89.61	**87.76**	79.76
BV+FV+RGB	96.02	**89.05**	**88.38**	96.34	**89.39**	**88.67**	**95.01**	87.59	**79.90**

Figure 62: An ablation study of multi-view features: Performance are evaluated on KITTI validation set. Figure is from [72]

Method	Data	IoU=0.25			IoU=0.5			IoU=0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D	Mono	62.94	48.2	42.68	25.19	18.2	15.52	2.53	2.31	2.31
3DOP	Stereo	85.49	68.82	64.09	46.04	34.63	30.09	6.55	5.07	4.1
VeloFCN	LIDAR	89.04	81.06	75.93	67.92	57.57	52.56	15.20	13.66	15.98
MV3D(BV+FV)	LIDAR	96.03	88.85	88.39	95.19	87.65	80.11	71.19	56.60	55.30
MV3D(BV+FV+RGB)	LIDAR+Mono	**96.52**	**89.56**	**88.94**	**96.02**	**89.05**	**88.38**	**71.29**	**62.68**	**56.56**

Figure 63: The comparison of the 3D detection performance based on the KITTI validation set. The comparison is provided by [72]

The performance of MV3D is evaluated based on the outdoor KITTI dataset. The performance of 3D object detection based on the test set can be found from the leaderboard. The performance of 3D object detection based on validation dataset is shown in Figures 62 and 63. It only provides the car detection results. Detection results for the pedestrians and cyclists

are not provided.

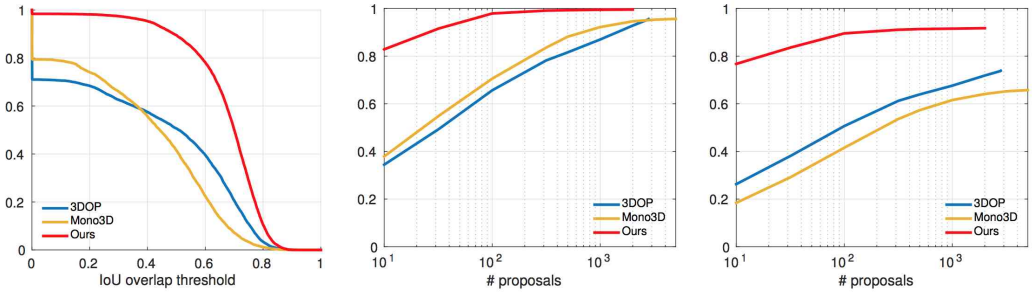


Figure 64: 3D bounding box Recall: From left to right: Recall vs IoU using 300 proposals, Recall vs #Proposals at IoU threshold of 0.25 and 0.5 respectively. Recall are evaluated on moderate data of KITTI validation set. Figure and Caption are from [72].

The performance of the RPN is shown in Figure 64

Framework of AVOD [73]

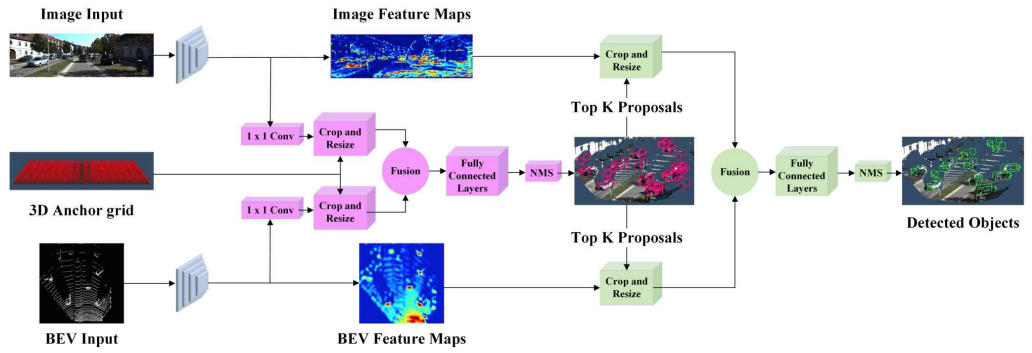


Figure 65: AVOD's architectural diagram. The feature extractors are shown in blue, the region proposal network in pink, and the second stage detection network in green. Figure and Caption are from [73].

The framework of AVOD is shown in Figure 65. AVOD is using the same encoding method as MV3D for the BEV. In AVOD, the value of M is set as 5 and the range of the LiDAR is $[0, 70] \times [-40, 40] \times [0, 2.5]$ meters. So the size of the input feature for the BEV is $700 \times 800 \times 7$. AVOD is using both the BEV and image to do the region proposals which is the main difference

to the MV3D work.

Performance of AVOD [73]

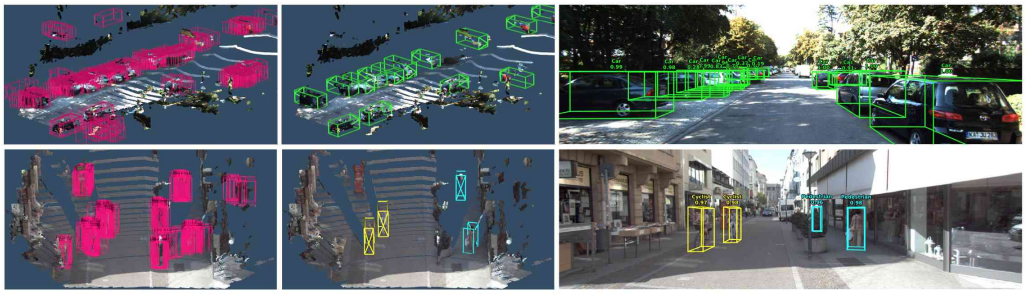


Figure 66: Qualitative results of AVOD [73] for cars (top) and pedestrians/cyclists (bottom). Left: 3D region proposal network output, Middle: 3D detection output, and Right: the projection of the detection output onto image space for all three classes. The 3D LiDAR point cloud has been colorized and interpolated for better visualization. Figure and Caption are from [73]

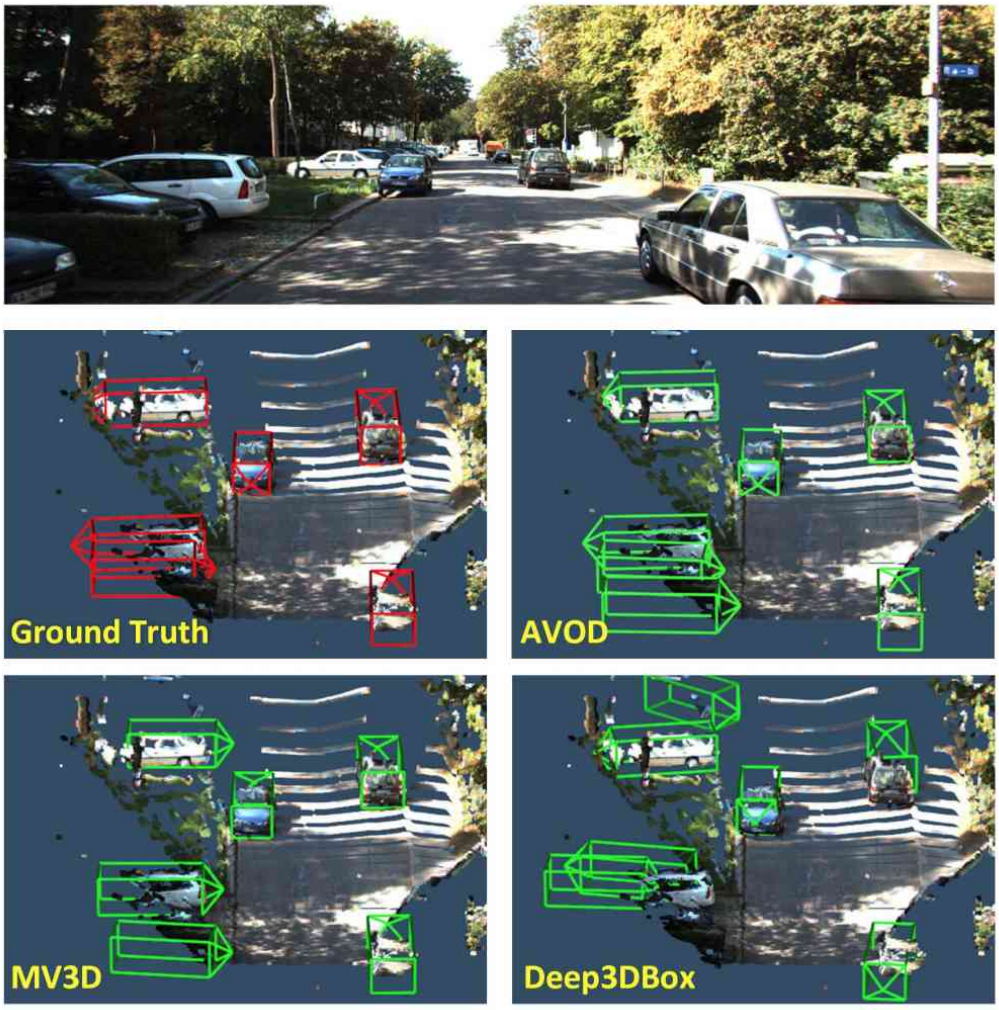


Figure 67: A qualitative comparison between MV3D, Deep3DBox [83], and AVOD architecture relative to KITTI's ground truth on a sample in the validation set. Figure is from [73]. We did not cover Deep3DBox in this survey.

The comparison of the AVOD and other methods is shown in the leaderboard. Some visualization results are shown in Figures 66 and 67.

VoxelNet [74]
Architecture of VoxelNet [74]

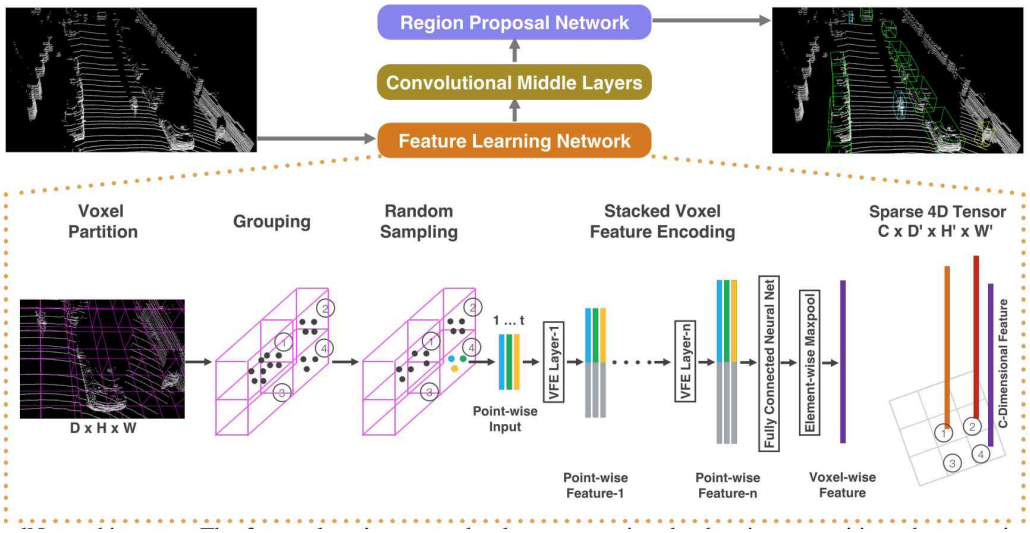


Figure 68: VoxelNet [74] architecture. Figure is from original paper.

VoxelNet architecture is shown in Figure 68. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

Feature Learning Network

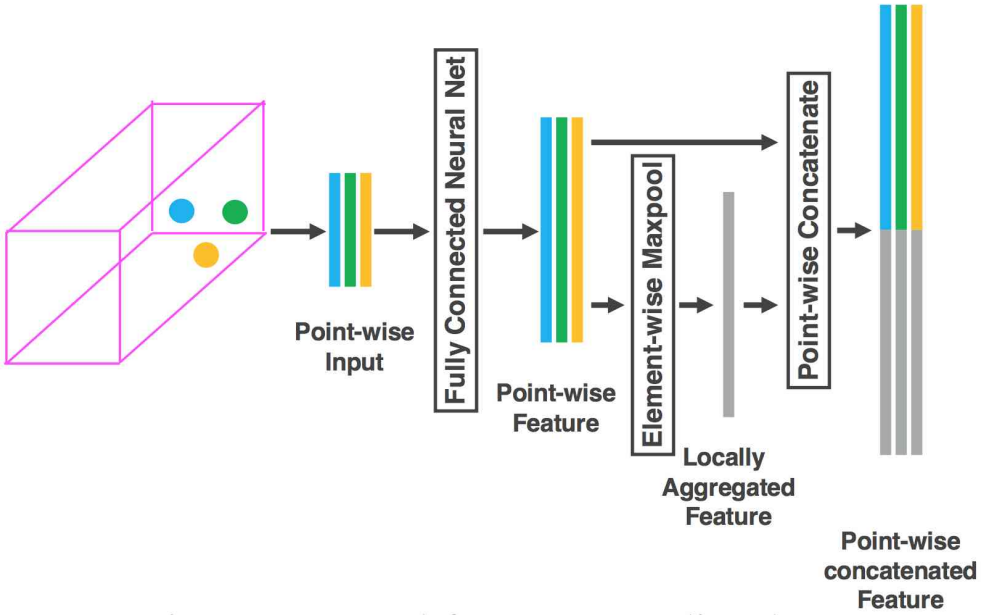


Figure 69: Voxel feature encoding layer. Figure is from [74]

The point cloud is subdivided into equally spaced voxels with size D', H', W' along the Z, Y, X axes respectively. The D', H' and W' for car detection is set as $D' = 10, H' = 400, W' = 352$ with the consideration of the range of LiDAR data as $[-3, 1] \times [-40, 40] \times [0, 70.4]$ meters and resolution of $0.4 \times 0.2 \times 0.2$ meters along Z, Y, X axis respectively.

A fixed number, T, of points from voxels containing more than T points are randomly sampled. For each point, a 7-feature is used which is $(x, y, z, r, x - v_x, y - v_y, z - v_z)$ where x, y, z are the XYZ coordinates for each point. r is the received reflectance and (v_x, v_y, v_z) is the centroid of points in the voxel. Voxel Feature Encoding is proposed in VoxelNet. The 7-feature for each point is fed into the Voxel feature encoding layer as shown in Figure 69. Fully connected networks are used in the VFE network with element-wise MaxPooling for each point and concatenation between each point and the element-wise MaxPooling output. The input of the VFE is $T \times 7$ and the output will be $T \times C$ where C depends on the FC layers of the VFE itself and depends on the whole VFE layers network used. Finally, an element-wise MaxPooling is used again and change the dimension of the output to $1 \times C$. Then for each voxel we have a one vector with C elements as shown in Figure 68. For the whole framework, we will have an input data with shape of $C \times D' \times H' \times W'$.

Convolutional Middle Layers

For car detection, $T = 35, C = 128$. A 3D CNN is used to further extract features. The input of this 3D CNN is $C \times D' \times H' \times W' = 128 \times 10 \times 400 \times 352$. The output is $64 \times 2 \times 400 \times 352$. After reshaping, the input to RPN is a feature map of size $128 \times 400 \times 352$.

Region Proposal Network

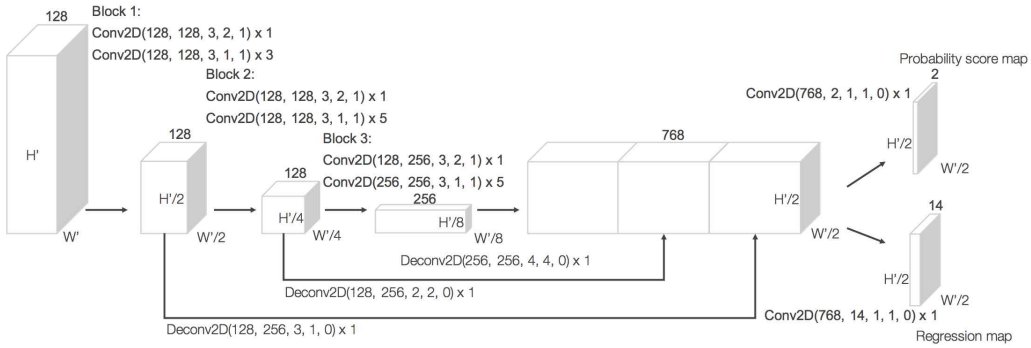


Figure 70: Region proposal network architecture. Figure is from [74].

Only one anchor size, $length = 3.9, width = 1.6, height = 1.56$ meters is used. It centered at $z = -1.0$ meters with two rotations, 0 and 90 degrees. The input to RPN is the feature map provided by the convolutional middle layers. The architecture of this network is illustrated in Figure 70.

Performance of VoxelNet

The comparison of the VoxelNet and other methods is shown in the leaderboard.

3.4.10 Outdoor & Indoor

Some 3D object detection systems can work well for both the indoor and outdoor scenarios are introduced in this part.

F-PointNet [75]

Framework of F-PointNet

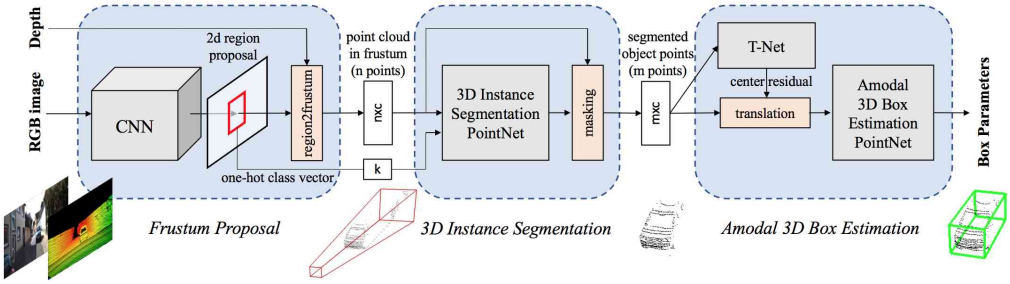


Figure 71: Frustum PointNets [75] for 3D object detection. Figure is from original paper.

The framework of the Frustum PointNets is given in Figure 71. It first leverages a 2D CNN object detector to propose 2D regions and classify their content. 2D regions are then lifted to 3D and thus become frustum proposals. Given a point cloud in a frustum ($n \times c$ with n points and c channels of XYZ, intensity etc. for each point), the object instance is segmented by binary classification of each point. Based on the segmented object point cloud ($m \times c$), a light-weight regression PointNet (T-Net) tries to align points by translation such that their centroid is close to amodal box center. At last the box estimation net estimates the amodal 3D bounding box for the object.

Performance of F-PointNet

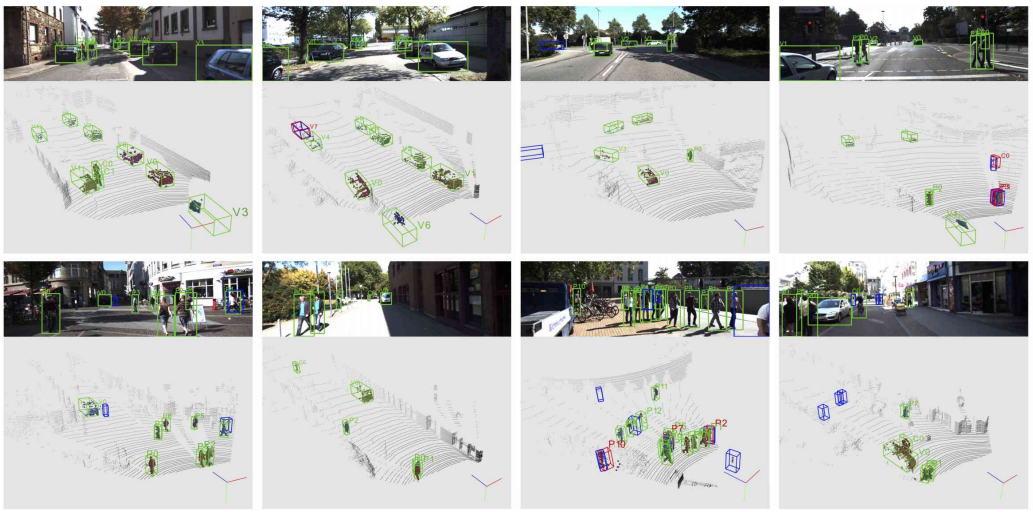


Figure 72: Visualizations of Frustum PointNet results on KITTI val set (best viewed in color with zoom in). These results are based on PointNet++ [66] models , running at 5 fps and achieving test set 3D AP of 70.39, 44.89 and 56.77 for car, pedestrian and cyclist, respectively. 3D instance masks on point cloud are shown in color. True positive detection boxes are in green, while false positive boxes are in red and ground truth boxes in blue are shown for false positive and false negative cases. Digit and letter beside each box denote instance id and semantic class, with v for cars, p for pedestrian and c for cyclist. Figure and Caption are from [75].

The comparison of the F-PointNet and other methods is shown in the leaderboard. Some detection results are demonstrated in Figure 72.

3.5 Data representation methods summary for 3D system

From the surveyed 3D systems, we can see the importance of the data representation to the performance. Here the pros and cons of different data representation methods to the classification and detection tasks is summarized:

- using **Projected multiple view RGB images**
  - **Pros** : It is a similar to the human being's recognization process for a 3D object by looking from different views. Since it can

encode the multiple view info into 2D RGB image, it can take the advantage of well developed 2D image recognization system such as 2D CNN to further classify or detect 3D objects.

  - **Cons** : Sometime, not all the desired multiple-view RGB images are available. In some papers which use multiple views such as RotationNet [48] and MVCNN [60], views from different angles are used to do the classification. This is possible when the whole object's CAD model is available. However, in the real application such as the autonomous cars scenario, from the self-driving cars' perspective, it can only take multiple-view of objects from one side during the driving process. The other side of the objects cannot be observed due to self-occultations. The partial availability of all views will reduce the performance of the algorithms based on all views.

- using **voxel**

  - **Pros**: It is a natural way to represent a 3D shape into voxel.
  - **Cons**: Corresponding to voxel representation, 3D CNN is commonly used based on this representation. The computation complexity of $\mathcal{O}(n^3)$ makes the system which uses 3D CNN based on voxel representation can only afford low resolution voxels. Low resolution will decrease the performance as not all the shape information will be preserved. Furthermore, the voxel representation will suffer from occultations.

- using **projected 2D similar images from depth image**

  - **Pros**: By projecting depth image to 2D similar images such as the 3 channels including the x, y and z values collecting from depth image can take the benefit of well developed 2D image system by using the state of the art technology such as 2D CNN.
  - **Cons**: Projecting depth info to 2D similar image will lose the geometry information and further decrease the classification or detection performance.

- using **raw point cloud**

  - **Pros**: properly using the point cloud info can well preserve the 3D geometry information. At the same time, the complexity of using point cloud is $\mathcal{O}(n^2)$ which is less expensive than using the voxel.

– **Cons**: The techniques of using point cloud is under development. Same to the voxel representation, point cloud will suffer from occultations.

4 Conclusion

In this survey, the main works in the 2D/3D image based object classification/detection system are introduced to help the researcher have an understanding about this area. Comprehensive comparisons are given to different methods based on the RPN network, detection network, feature encoding methods, bounding box encoding methods and also the application scenarios. Also the performance of different methods are compared based on some commonly used datasets.

References

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, 1958.

[2] M. Minsky, "Perceptrons: An introduction to computational geometry," *MIT Press. ISBN 0-262-63022-2*, 1969.

[3] `https://en.wikipedia.org/wiki/Artificial_neural_network`. Accessed: 2017-10-20.

[4] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[12] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016.

[13] S. Zagoruyko, "imagenet-validation.torch." `https://github.com/szagoruyko/imagenetvalidation.torch`, 2016.

[14] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[15] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *CoRR*, vol. abs/1606.02147, 2016.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[18] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[19] K. He, "Iccv15 talks." `http://image-net.org/challenges/talks/ilsvrc2015_deep_residual_learning_kaiminghe.pdf`, 2015.

[20] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[21] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vision*, vol. 104, pp. 154–171, Sept. 2013.

[22] R. B. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1440–1448, IEEE Computer Society, 2015.

[23] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 91–99, 2015.

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

[25] T. Grel, "Region of interest pooling explained." `http://modelnet.cs.princeton.edu/`, 2017.

[26] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[27] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[28] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[29] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head R-CNN: in defense of two-stage object detector," *CoRR*, vol. abs/1711.07264, 2017.

[30] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016.

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[32] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," *CoRR*, vol. abs/1701.06659, 2017.

[33] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017.

[34] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013.

[35] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[36] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[37] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016.

[38] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," *CoRR*, vol. abs/1611.07709, 2016.

[39] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," *CoRR*, vol. abs/1502.03240, 2015.

[40] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," *CoRR*, vol. abs/1411.5752, 2014.

[41] P. H. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," *CoRR*, vol. abs/1506.06204, 2015.

[42] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3d object reconstruction from a single depth view with adversarial learning," *CoRR*, vol. abs/1708.07969, 2017.

[43] C. V. Laboratory, "Computer vision laboratory, hunter college of cuny." http://www.cs.hunter.cuny.edu/~ioannis/Vision.htm, 2018.

[44] velodyne official website, "velodyne lidar." http://velodynelidar.com/blog/128-lasers-car-go-round-round-david-hall-velodynes-new-sensor/, 2017.

[45] VelodyneLiDAR, "Lidar 101." https://www.youtube.com/watch?v=RW5ONf7DtyY, 2017.

[46] V. official website, "Raw user data collected by velodyne hdl-64e 3d lidar." http://velodynelidar.com/hdl-64e.html, 2017.

[47] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, "3d shapenets for 2.5d object recognition and next-best-view prediction," *CoRR*, vol. abs/1406.5670, 2014.

[48] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016.

[49] K. Sfikas, I. Pratikakis, and T. Theoharis, "Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval," *Computers and Graphics*, vol. 71, pp. 208 – 218, 2018.

[50] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *CoRR*, vol. abs/1608.04236, 2016.

[51] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3d object recognition," 2017.

[52] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-Organizing Network for Point Cloud Analysis," *ArXiv e-prints*, Mar. 2018.

[53] R. Klokov and V. S. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," *CoRR*, vol. abs/1704.01222, 2017.

[54] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks," *CoRR*, vol. abs/1711.08241, 2017.

[55] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," *CoRR*, vol. abs/1604.03265, 2016.

[56] V. Hegde and R. Zadeh, "Fusionnet: 3d object classification using multiple data representations," *CoRR*, vol. abs/1607.05695, 2016.

[57] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the panorama representation for convolutional neural network classification and retrieval," 04 2017.

[58] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," *CoRR*, vol. abs/1605.08359, 2016.

[59] M. Ren, L. Niu, and Y. Fang, "3d-a-nets: 3d deep dense descriptor for volumetric shapes with adversarial networks," *CoRR*, vol. abs/1711.10108, 2017.

[60] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015.

[61] S. Ravanbakhsh, J. Schneider, and B. Poczos, "Deep Learning with Sets and Point Clouds," *ArXiv e-prints*, Nov. 2016.

[62] L. Minto, P. Zanuttigh, and G. Pagnutti, "Deep learning for 3d shape classification based on volumetric density and surface approximation clues," in *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pp. 317–324, INSTICC, SciTePress, 2018.

[63] ModelNet, "Modelnet." `http://modelnet.cs.princeton.edu/`, 2018.

[64] A. Kanezaki, "Rotationnet: Learning object classification using unsupervised viewpoint estimation," *CoRR*, vol. abs/1603.06208, 2016.

[65] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[66] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017.

[67] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[68] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[69] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[70] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015.

[71] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in RGB-D images," *CoRR*, vol. abs/1511.02300, 2015.

[72] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," *CoRR*, vol. abs/1611.07759, 2016.

[73] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," *ArXiv e-prints*, Dec. 2017.

[74] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *CoRR*, vol. abs/1711.06396, 2017.

[75] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from RGB-D data," *CoRR*, vol. abs/1711.08488, 2017.

[76] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *IEEE CVPR*, 2016.

[77] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D.

Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 424–432, Curran Associates, Inc., 2015.

[78] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, (Berlin, Heidelberg), pp. 746–760, Springer-Verlag, 2012.

[79] S. Gupta, R. B. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," *CoRR*, vol. abs/1407.5736, 2014.

[80] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," *ECCV*, 2014.

[81] KITTI, "Kitti 3d object detection leaderboards." `http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d`, 2017.

[82] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *CoRR*, vol. abs/1608.07916, 2016.

[83] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," *CoRR*, vol. abs/1612.00496, 2016.