3D object detection on KITTI Benchmark

An introduction to deep learning

Could a single layer perceptron be sufficient? Sufficient for what? What actually do these networks do in mathematical terms?

What is the famous XOR problem, how is it related to understand neural nets, or any classification problem?

What is a hidden layer and why is it required?

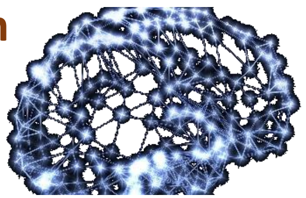
- Can’t we calculate end to end directly? Do we actually need more than one hidden layer?

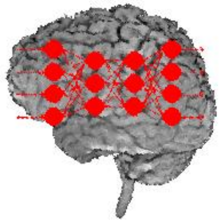
Why is deep learning not deeper for the last two years?

Why was convolution operation integrated into neural networks? What does it actually calculate?

Connectionism

Perceptrons…





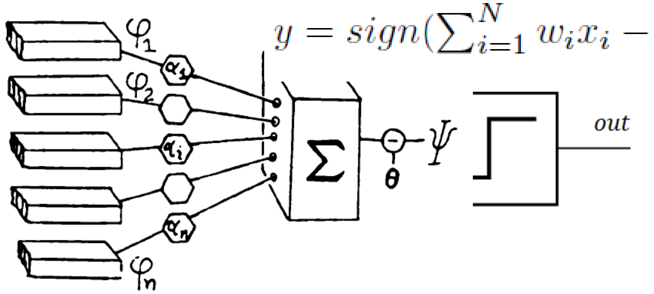
A one hidden layer perceptron is a universal approximator!

Hornik (1991)

Single Layer Perceptron

Rosenblatt (1957)

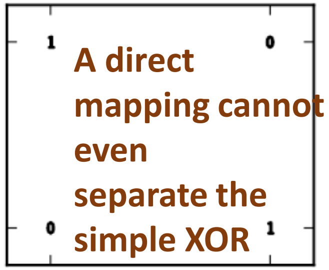
$$y = sign(\sum_{i=1}^{N} w_i x_i - \theta)$$



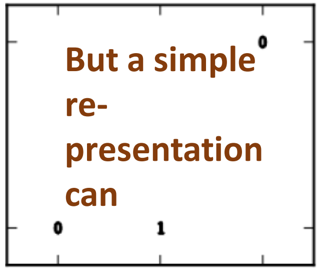
A perceptron is a simple mapping function

The famous XOR problem

Minsky and Papert (1969)

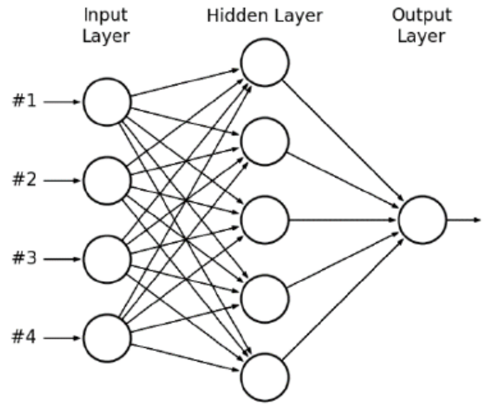


A direct mapping cannot even separate the simple XOR variables,



But a simple re-presentation can

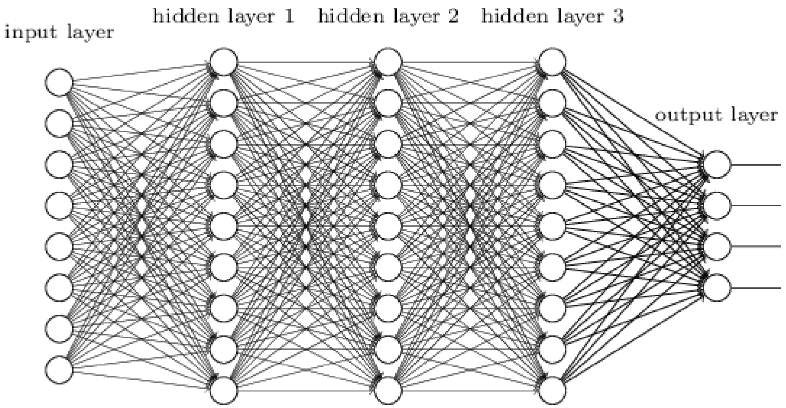
Multi Layer Perceptron



No need for deep perceptron in theory

Deep Perceptron

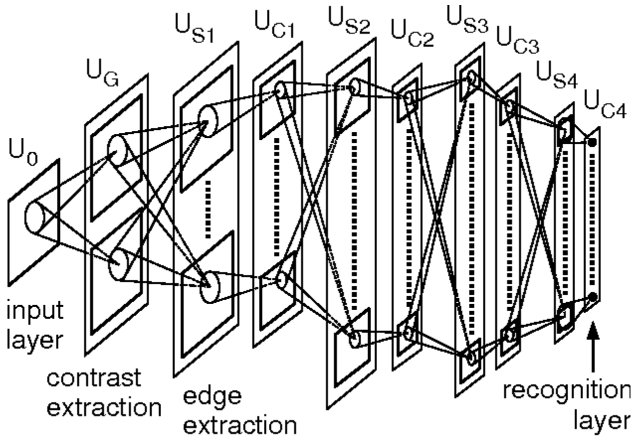
Tough to optimize



Vanishing gradient in Backpropagation

Neocognitron

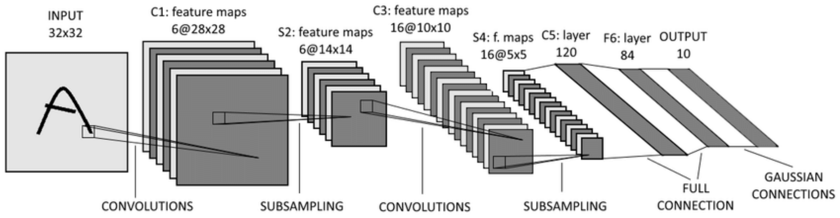
Fukushima (1980)



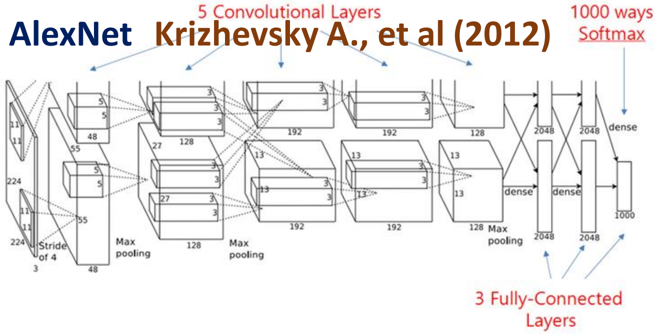
Local features instead of full connections

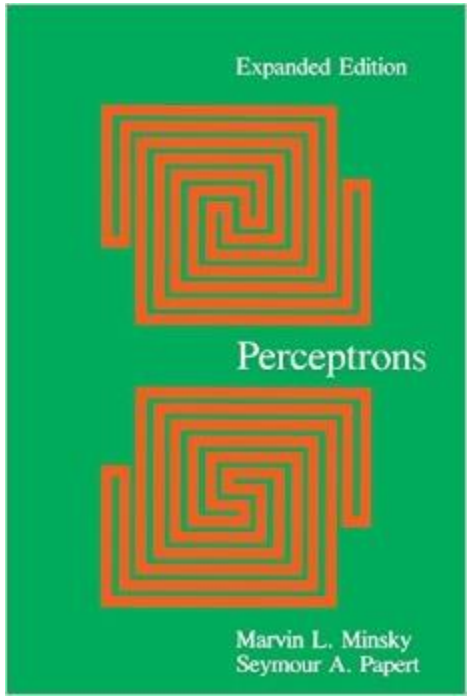
Convolutional Neural Network

LeNet-5 LeCun (1998)



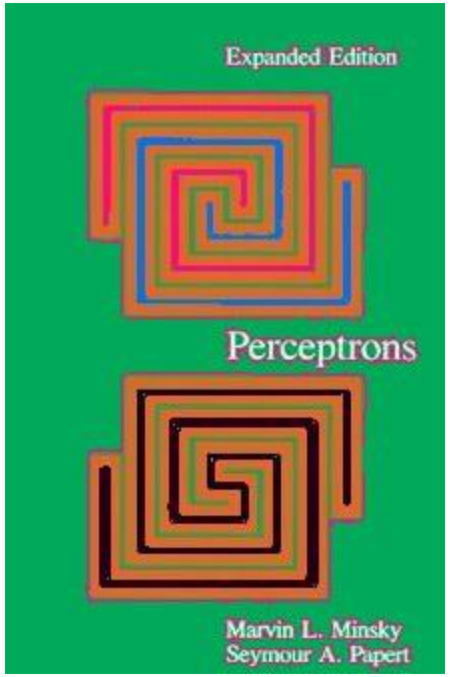
AlexNet Krizhevsky A., et al (2012)

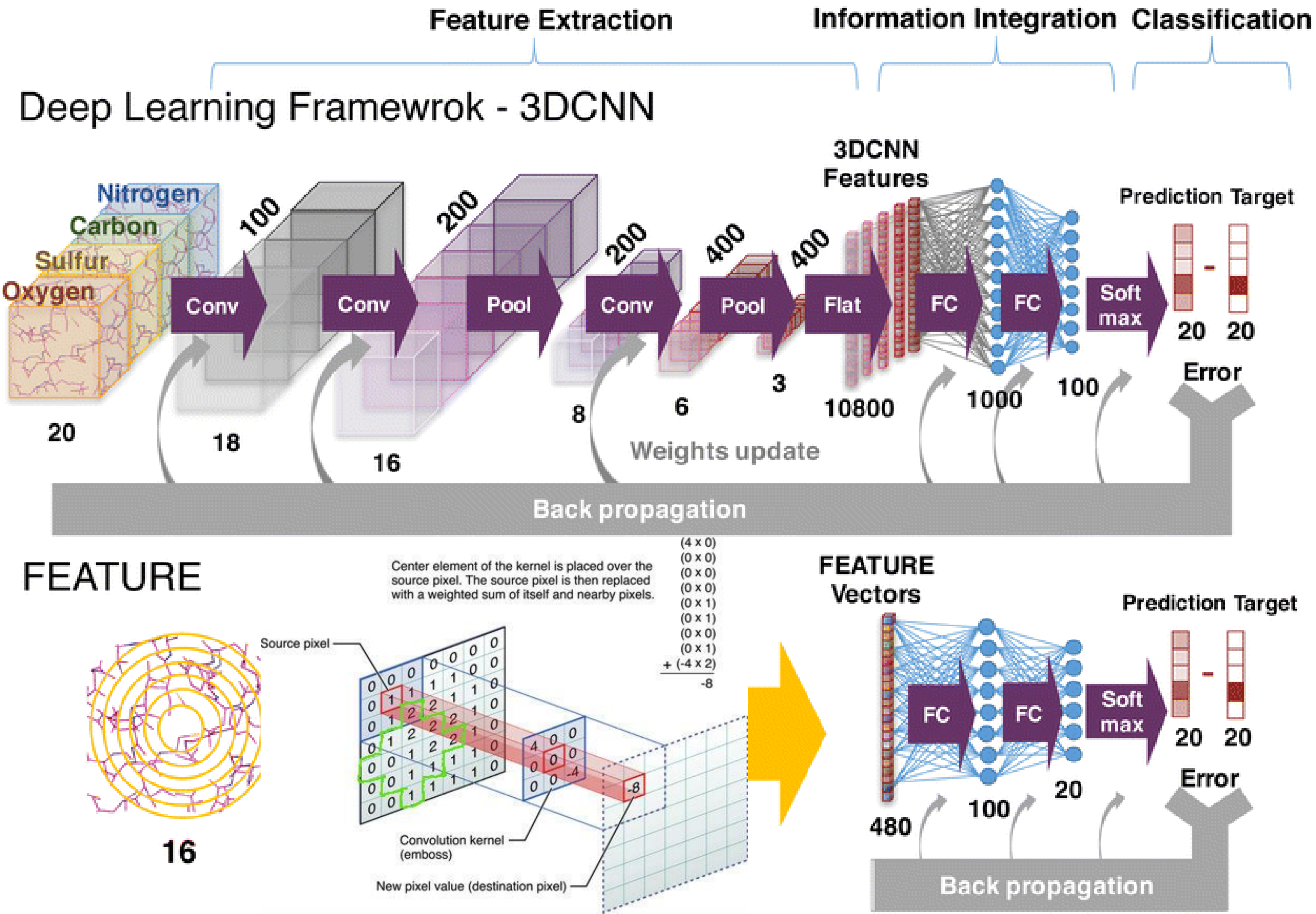




When perceptron-like machines came on the scene, we found that in order to understand their capabilities we needed some new ideas. It was not enough simply to examine the machines themselves or the procedures used to make them learn. **Instead, we had to find new ways to understand the problems they would be asked to solve.** This is why our book turned out to be concerned less with perceptrons per se than with concepts that could help us see the relation between patterns and the types of parallel-machine architectures that might or might not be able to recognize them.

Minsky and Papert (1988)





Wen Torng and Russ B. Altman (2017) 3D deep convolutional neural networks for amino acid environment similarity analysis. BMC Bioinformatics 2017 18:302

Object detection recognition segmentation

Object Detection

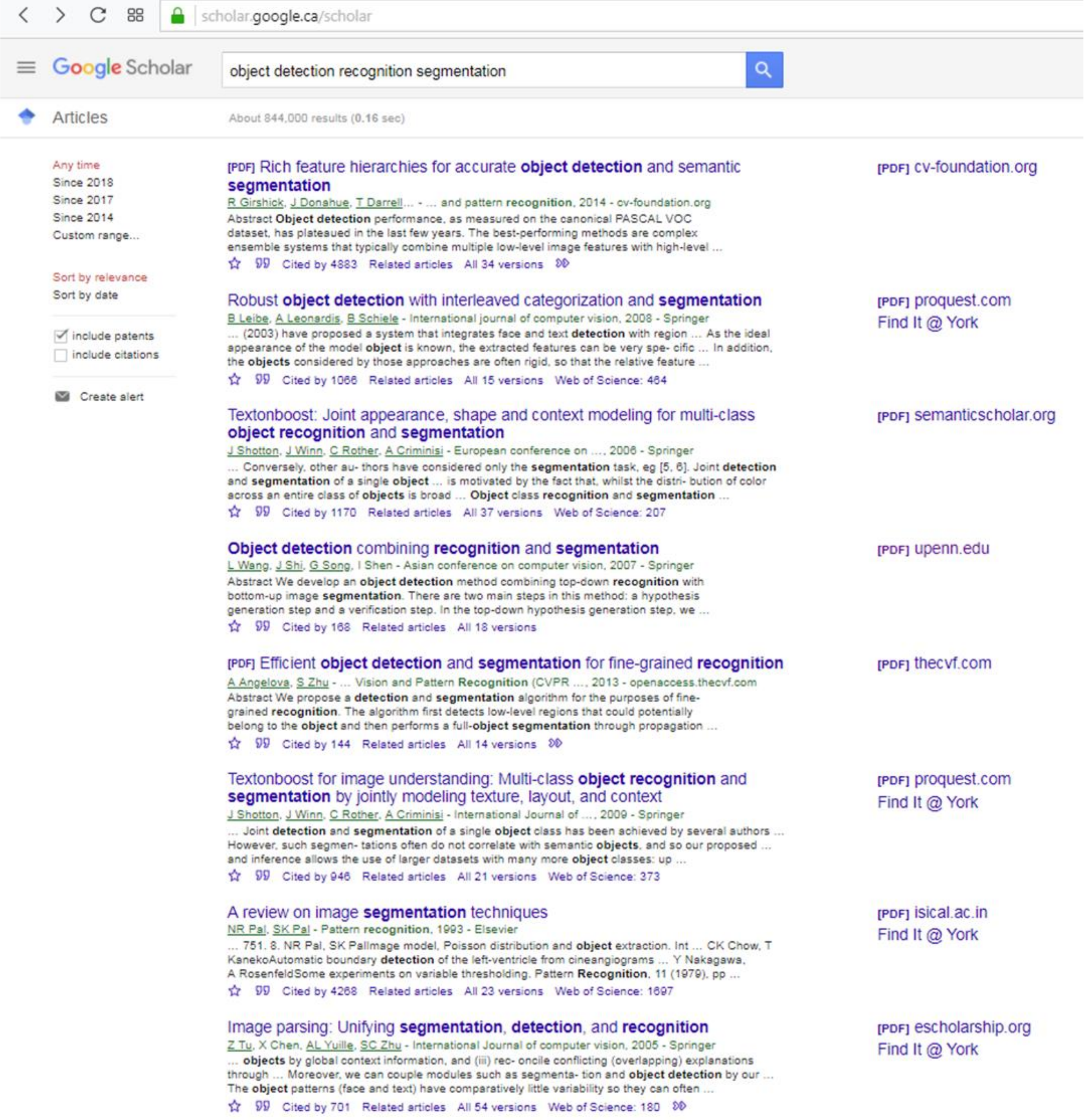
- Generate a bounding box

Object Recognition
Object Segmentation

Localization
Regression
Classification
Labeling
Semantic Segmentation

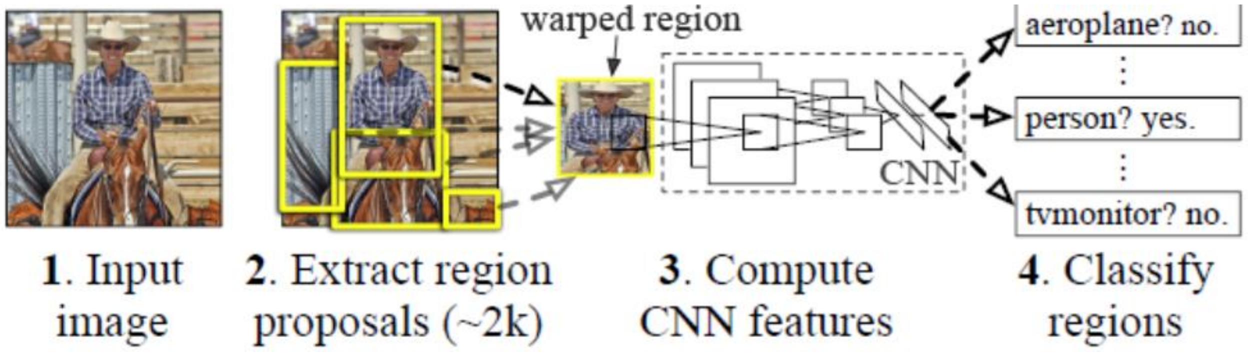
Multidimensional Labeling
Categorization

Representation

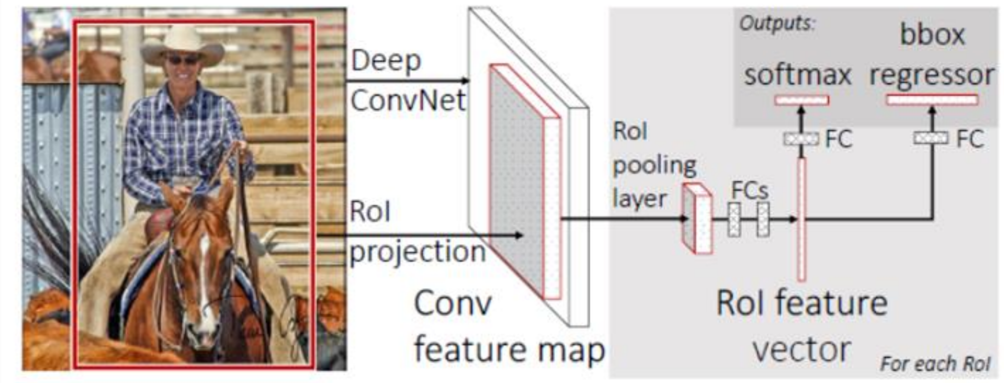


From slower to faster RCNN

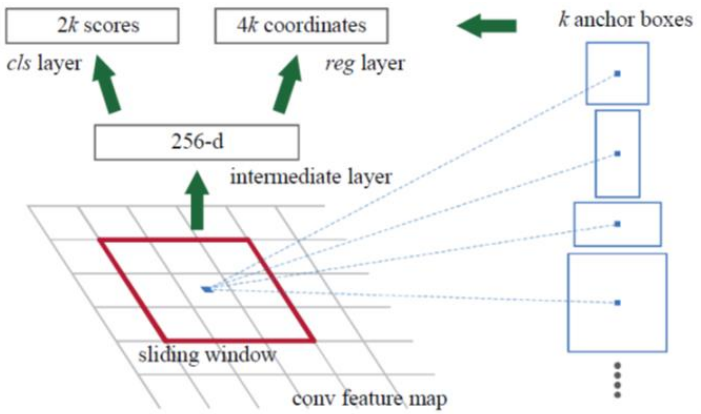
Run CNN for each proposal

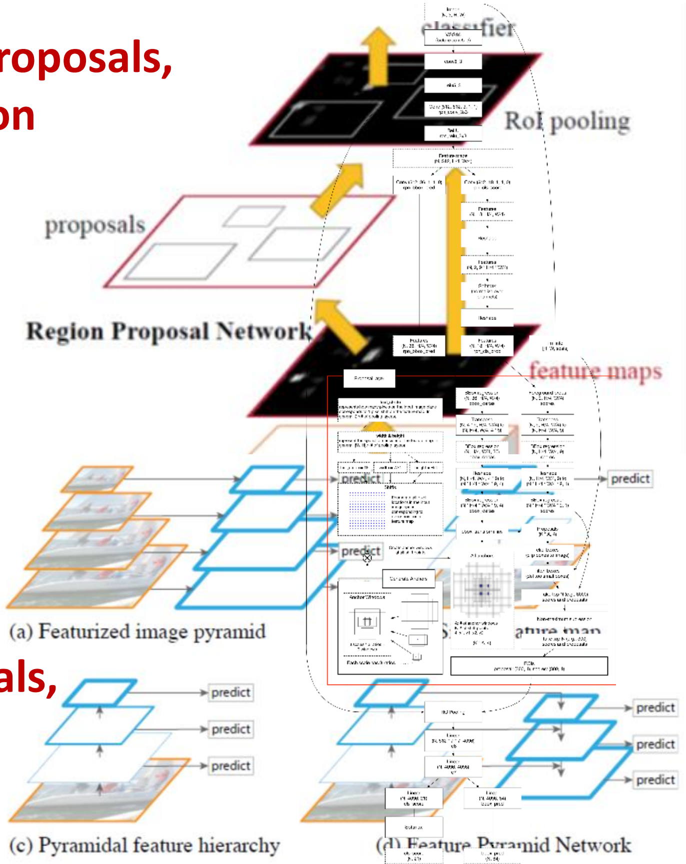


Run CNN for image once, project the proposals



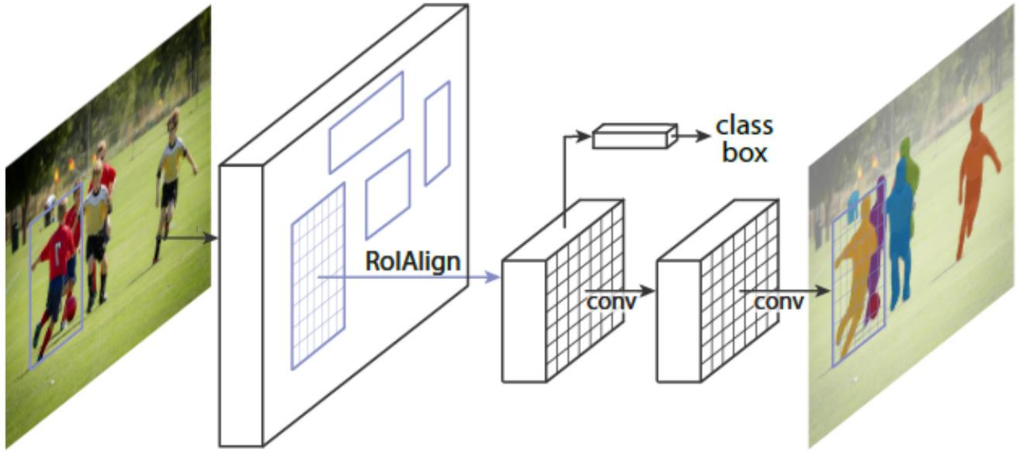
Run CNN before making proposals, propose in lower dimension





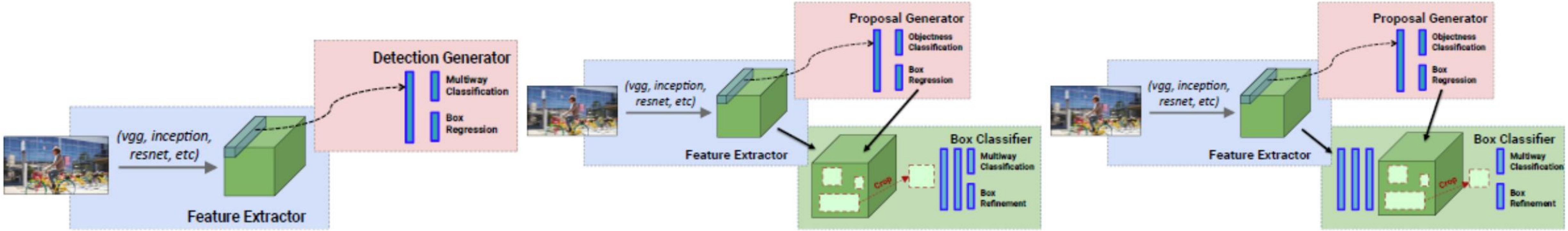
Run CNN before making proposals, propose in multiple scales

In addition, run mask segmentation in parallel with classification



Run CNN before making proposals, propose in paired (fused) multiple scales

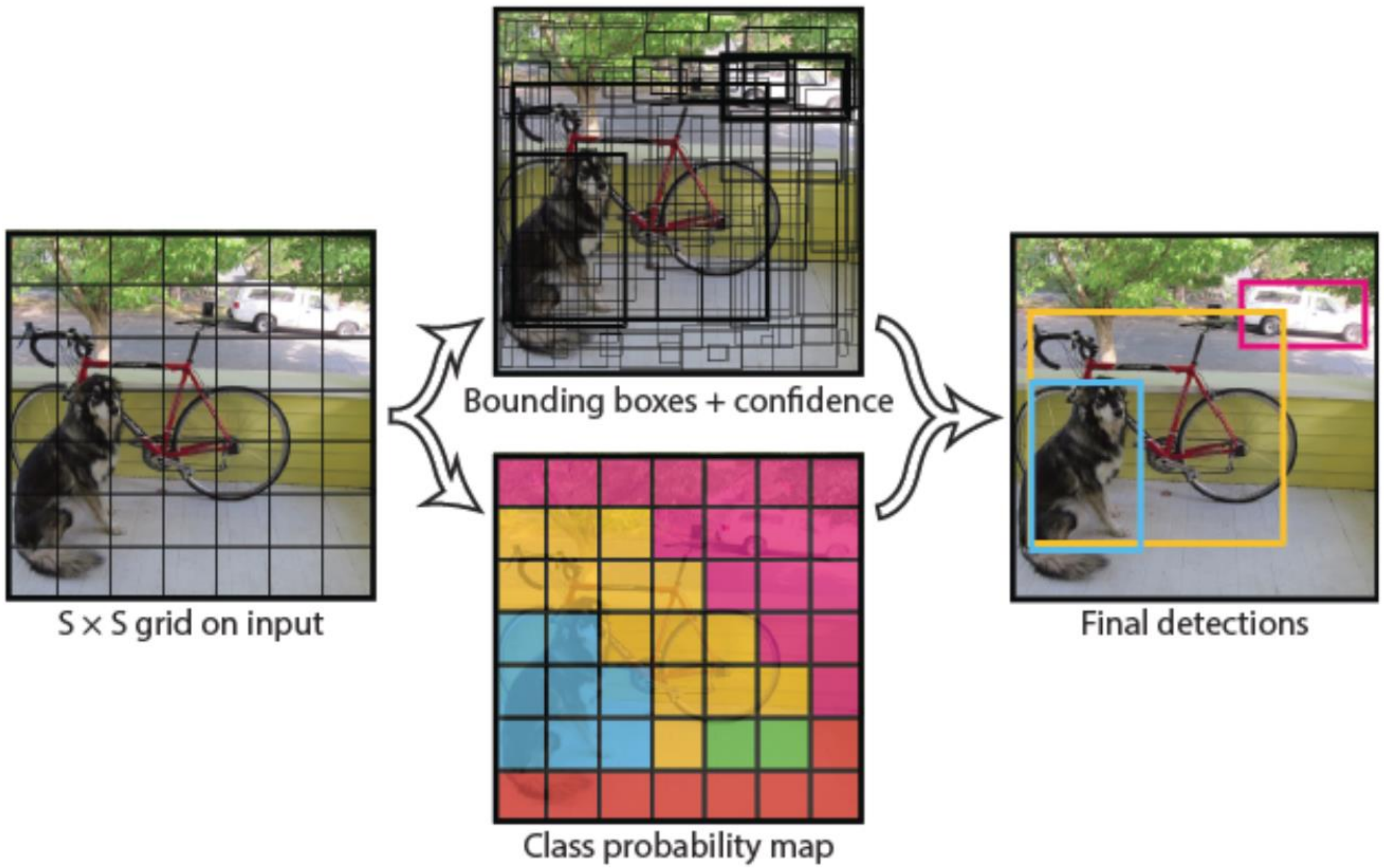
What could have been a fastest RCNN?



(a) SSD. (b) Faster RCNN. (c) R-FCN.

Figure 1: High level diagrams of the detection meta-architectures compared in this paper.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017, July). Speed/accuracy trade-offs for modern convolutional object detectors. In IEEE CVPR.



"Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design."

KITTI 3D Results (updated every week)

Car

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment
1	AVOD-FPN		code	**71.88 %**	**81.94 %**	**66.38 %**	0.1 s	Titan X (Pascal)
	J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. arXiv preprint arXiv:1712.02294 2017.							
2	F-PointNet			70.39 %	81.20 %	62.19 %	0.17 s	GPU @ 3.0 Ghz (Python)
3	DF-PC_CNN			66.22 %	80.28 %	58.94 %	0.5 s	GPU @ 3.0 Ghz (Matlab + C/C++)
4	AVOD		code	65.78 %	73.59 %	58.38 %	0.08 s	Titan X (pascal)
	J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. arXiv preprint arXiv:1712.02294 2017.							
5	VxNet(LiDAR)			65.11 %	77.47 %	57.73 %	0.03 s	GPU @ 2.5 Ghz (Python + C/C++)
6	MV3D			62.35 %	71.09 %	55.12 %	0.36 s	GPU @ 2.5 Ghz (Python + C/C++)
	X. Chen, H. Ma, J. Wan, B. Li and T. Xia: Multi-View 3D Object Detection Network for Autonomous Driving. CVPR 2017.							
7	MV3D (LIDAR)			52.73 %	66.77 %	51.31 %	0.24 s	GPU @ 2.5 Ghz (Python + C/C++)
	X. Chen, H. Ma, J. Wan, B. Li and T. Xia: Multi-View 3D Object Detection Network for Autonomous Driving. CVPR 2017.							
8	F-PC_CNN			48.07 %	60.06 %	45.22 %	0.5 s	GPU @ 3.0 Ghz (Matlab + C/C++)
	X. Du, M. Jr., S. Karaman and D. Rus: A General Pipeline for 3D Detection of Vehicles. 2018.							
9	SDN			24.08 %	37.87 %	22.01 %	0.096 s	GPU @ 1.7 Ghz (Python)
10	LiCar			21.92 %	23.90 %	20.31 %	0.09 s	GPU @ 2.5 Ghz (Python)
11	LMnet			15.67 %	19.20 %	15.83 %	0.013 s	GPU @ 1.1 Ghz (Python + C/C++)
12	LMNetV2			15.24 %	14.75 %	12.85 %	0.02 s	GPU @ 2.5 Ghz (C/C++)
13	3dSSD			14.97 %	14.71 %	19.43 %	0.03 s	GPU @ 2.5 Ghz (Python + C/C++)
14	LMnetV1.1			11.51 %	20.52 %	11.53 %	0.01 s	GPU @ 1.0 Ghz (Python + C/C++)
15	M3D			7.81 %	10.25 %	6.54 %	0.4 s	GPU @ 2.5 Ghz (Python + C/C++)
16	DoBEM			6.95 %	7.42 %	13.45 %	0.6 s	GPU @ 2.5 Ghz (Python + C/C++)
	S. Yu, T. Westfechtel, R. Hamada, K. Ohno and S. Tadokoro: Vehicle Detection and Localization on Bird's Eye View Elevation Images Using Convolutional Neural Network. IEEE Inte Safety, Security, and Rescue Robotics (SSRR) 2017.							
17	CSoR			6.79 %	6.76 %	6.14 %	3.5 s	4 cores @ >3.5 Ghz (Python + C/C++)
	L. Plotkin: PyDriver: Entwicklung eines Frameworks für räumliche Detektion und Klassifikation von Objekten in Fahrzeugumgebung. 2015.							
18	MonoFusion			5.18 %	7.08 %	4.68 %	0.12 s	TITAN X GPU
19	3D-SSMFCNN		code	2.28 %	2.39 %	1.52 %	0.1 s	GPU @ 1.5 Ghz (C/C++)
	L. Novak: Vehicle Detection and Pose Estimation for Autonomous Driving. 2017.							
20	SPC			0.52 %	0.68 %	0.60 %	0.4 s	4 cores @ 2.5 Ghz (Python)
21	LidarNet			0.02 %	0.01 %	0.03 %	0.007 s	GPU @ 2.5 Ghz (C/C++)
22	mBoW			0.00 %	0.00 %	0.00 %	10 s	1 core @ 2.5 Ghz (C/C++)
	J. Behley, V. Steinhage and A. Cremers: Laser-based Segment Classification Using a Mixture of Bag-of-Words. Proc. of the IEEE/RSJ International Conference on Intelligent Robots							

Table as LaTeX | Only published Methods

Pedestrian

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment
1	F-PointNet			**44.89 %**	**51.21 %**	**40.23 %**	0.17 s	GPU @ 3.0 Ghz (Python)
2	AVOD-FPN		code	39.00 %	46.35 %	36.58 %	0.1 s	Titan X (Pascal)
	J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. arXiv preprint arXiv:1712.02294 2017.							
3	VxNet(LiDAR)			33.69 %	39.48 %	31.51 %	0.03 s	GPU @ 2.5 Ghz (Python + C/C++)
4	AVOD		code	31.51 %	38.28 %	26.98 %	0.08 s	Titan X (pascal)
	J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. arXiv preprint arXiv:1712.02294 2017.							
5	3dSSD			17.35 %	20.22 %	17.20 %	0.03 s	GPU @ 2.5 Ghz (Python + C/C++)
6	LMNetV2			11.46 %	13.64 %	11.57 %	0.02 s	GPU @ 2.5 Ghz (C/C++)
7	LMnet			0.70 %	0.66 %	0.70 %	0.013 s	GPU @ 1.1 Ghz (Python + C/C++)
8	LMnetV1.1			0.69 %	0.64 %	0.74 %	0.01 s	GPU @ 1.0 Ghz (Python + C/C++)
9	mBoW			0.00 %	0.00 %	0.00 %	10 s	1 core @ 2.5 Ghz (C/C++)
	J. Behley, V. Steinhage and A. Cremers: Laser-based Segment Classification Using a Mixture of Bag-of-Words. Proc. of the IEEE/RSJ International Conference on Intelligent Robots a							

Table as LaTeX | Only published Methods

Cyclist

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment
1	F-PointNet			**56.77 %**	**71.96 %**	**50.39 %**	0.17 s	GPU @ 3.0 Ghz (Python)
2	VxNet(LiDAR)			48.36 %	61.22 %	44.37 %	0.03 s	GPU @ 2.5 Ghz (Python + C/C++)
3	AVOD-FPN		code	46.12 %	59.97 %	42.36 %	0.1 s	Titan X (Pascal)
	J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. arXiv preprint arXiv:1712.02294 2017.							
4	AVOD		code	44.90 %	60.11 %	38.80 %	0.08 s	Titan X (pascal)
	J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. arXiv preprint arXiv:1712.02294 2017.							
5	LMNetV2			3.23 %	2.84 %	3.28 %	0.02 s	GPU @ 2.5 Ghz (C/C++)
6	LMnetV1.1			0.31 %	0.49 %	0.55 %	0.01 s	GPU @ 1.0 Ghz (Python + C/C++)
7	LMnet			0.29 %	0.55 %	0.36 %	0.013 s	GPU @ 1.1 Ghz (Python + C/C++)
8	3dSSD			0.24 %	0.25 %	0.25 %	0.03 s	GPU @ 2.5 Ghz (Python + C/C++)
9	mBoW			0.00 %	0.00 %	0.00 %	10 s	1 core @ 2.5 Ghz (C/C++)
	J. Behley, V. Steinhage and A. Cremers: Laser-based Segment Classification Using a Mixture of Bag-of-Words. Proc. of the IEEE/RSJ International Conference on Intelligent Robots a							

Table as LaTeX | Only published Methods

AVOD

Key idea?

Benefit from fusion before region proposal

- Two stage proposal pipeline
- Initial proposals in lower dimension feature space
- Fusion is before proposals, in low dimension
- Backproject proposals to higher resolution
- Another fusion, this time with less proposals
- Final proposals with fully connected layers

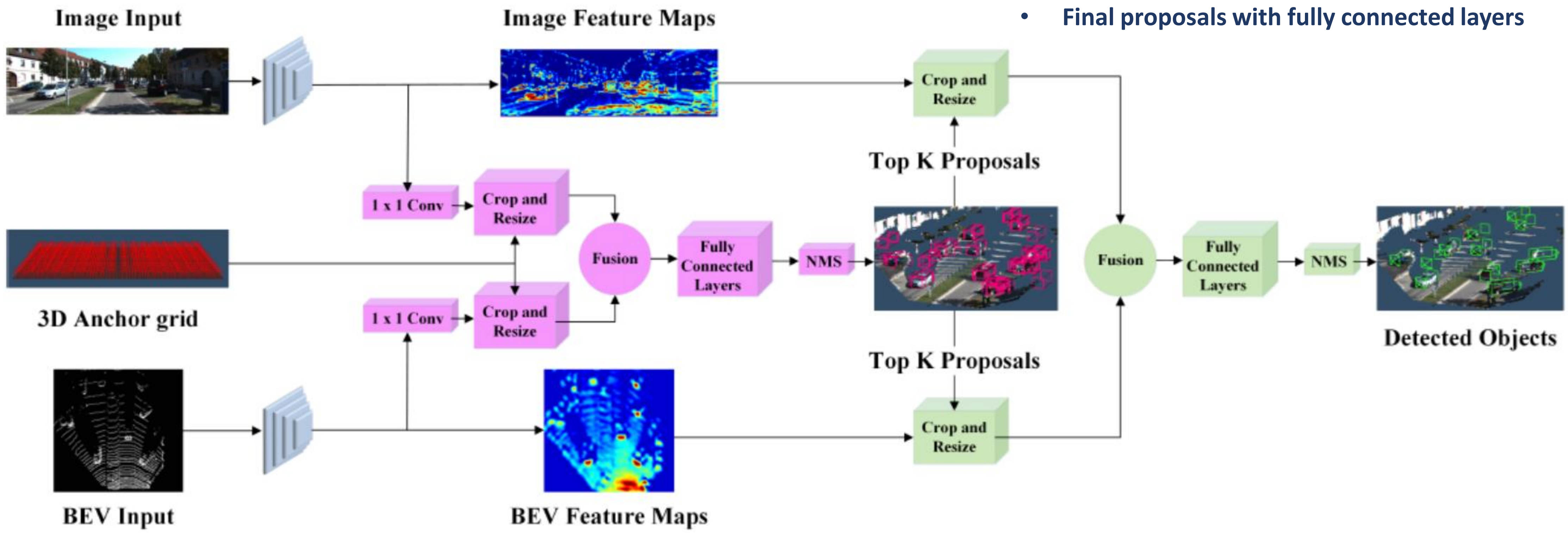
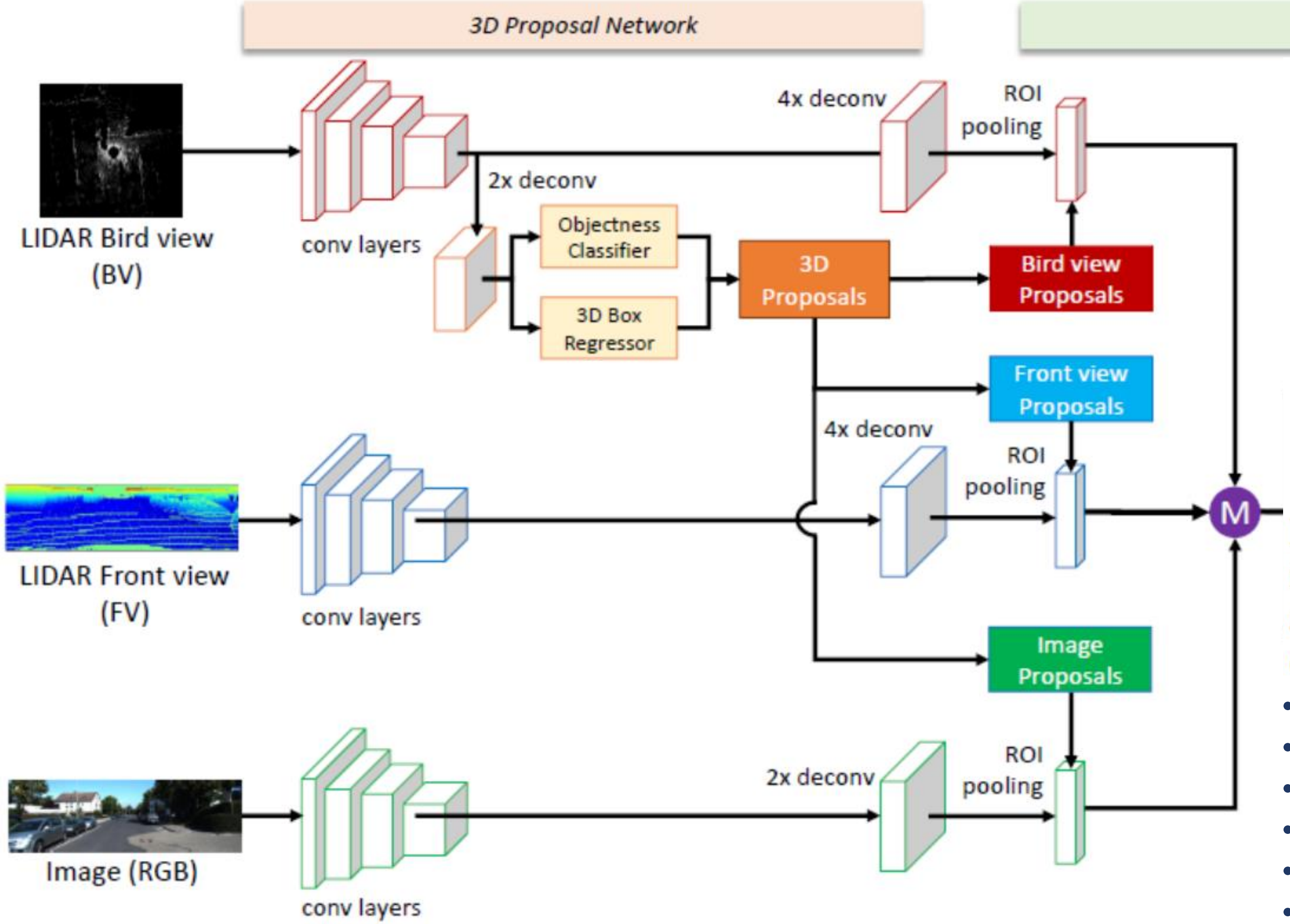


Fig. 2: The proposed method's architectural diagram. The feature extractors are shown in **blue**, the region proposal network in **pink**, and the second stage detection network in **green**.

MV3D



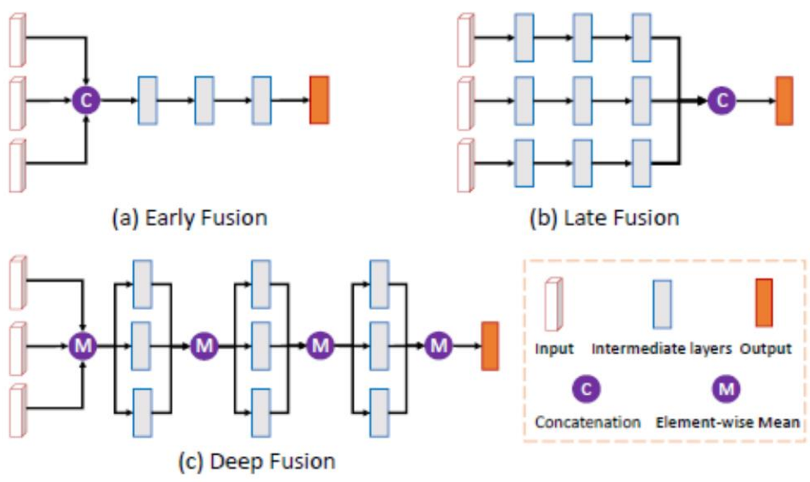


Figure 3: **Architectures of different fusion schemes:** We instantiate the join nodes in early/late fusion with *concatenation* operation, and deep fusion with *element-wise mean* operation.

- **Two stage proposal pipeline**
- **Initial proposals in lower dimension feature space**
- **Only LIDAR BEV deployed for proposals, fusion is afterwards,**
- **Backproject proposals to higher resolution**
- **Another fusion, this time with less proposals**
- **Final proposals with auxiliary losses**

Figure 1: **Multi-View 3D object detection network (MV3D):** 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 project them to three views. 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.

AVOD

Benefit from fusion before region proposal

- Two stage proposal pipeline
- Initial proposals in lower dimension feature space
- Fusion is before proposals, in low dimension
- Backproject proposals to higher resolution
- Another fusion, this time with less proposals
- Final proposals with fully connected layers

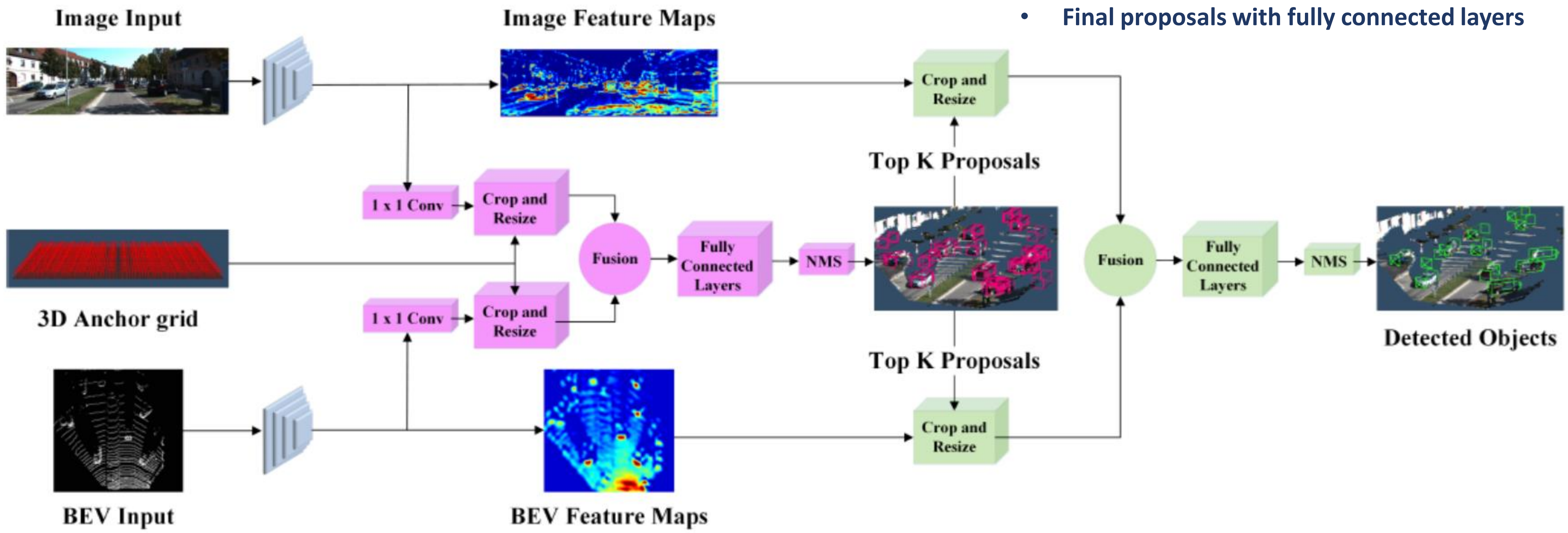


Fig. 2: The proposed method's architectural diagram. The feature extractors are shown in **blue**, the region proposal network in **pink**, and the second stage detection network in **green**.

Pipeline	Single / Two-stage / Multistage
Sensors	Separately / Together
Fusion	Early / Late
Domain	General / Specific

Assumptions Simplification

- Only certain initial box ratios

Implicit Assumptions

- BEV doesn't work in indoor

Added Constraints

- Only certain height values

Detection	Box provide a simple structure to work with
Segmentation	Roughly know potential detection area
Classification	Always a key information, smartest constraint
Fusion	Hence perform a single operation on richer
Regression	To find a rough box and refine is easier
Proposal	To limit the search space is mandatory
Resolution	Not all the operations need full resolution
Representation	Inevitable if you operate. Design so to get best
Modality	More information, from more resources
Domain	Where to operate and take advantage best
Assumption	Explicit or implicit, beware.

The design of the pipeline starts from an initial idea, and expands with the necessities to make it a full framework

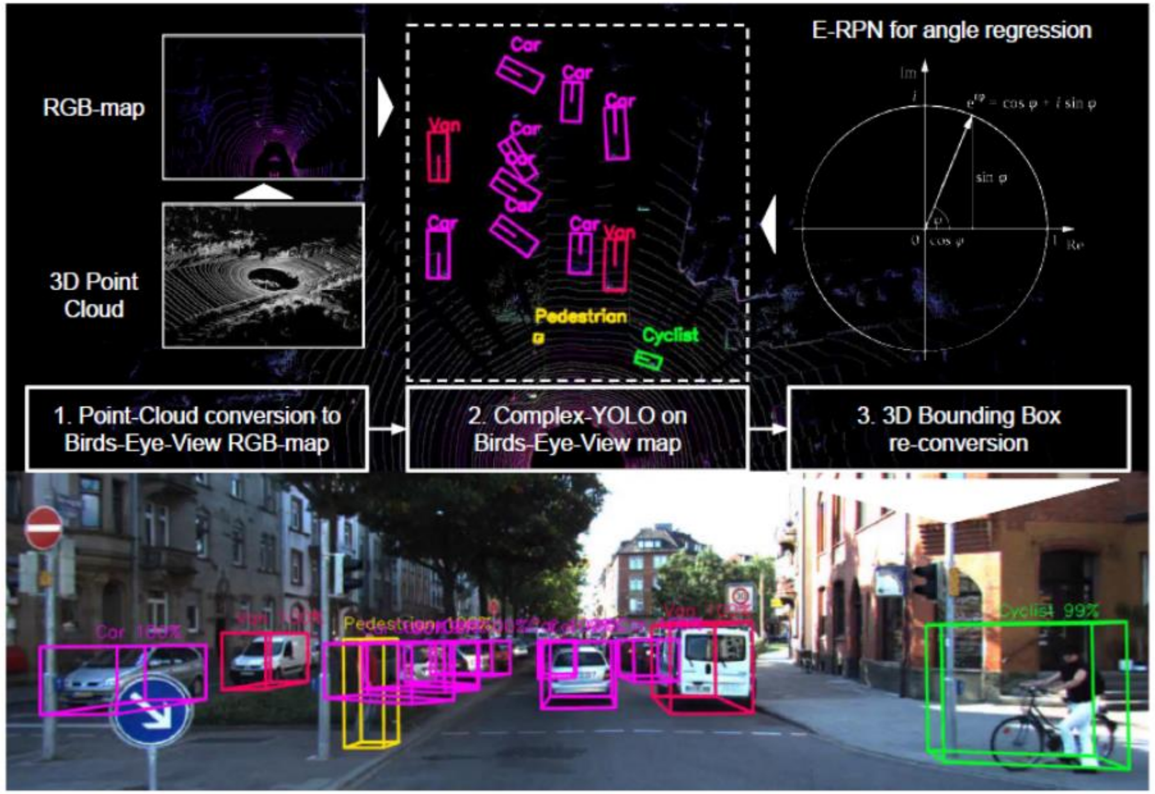


Fig. 1. Complex-YOLO is a very efficient model that directly operates on Lidar only based birds-eye-view RGB-maps to estimate and localize accurate 3D multiclass bounding boxes. The upper part of the figure shows a bird view based on a Velodyne HDL64 point cloud (Geiger et al. [1]) such as the predicted objects. The lower one outlines the re-projection of the 3D boxes into image space. Note: Complex-YOLO needs no camera image as input, it is Lidar based only.

- One stage proposal pipeline (fast!)
- Proposals in lower dimension feature space
- Only LIDAR BEV deployed for proposals, no fusion! (fast)
- Limit the search space by grid, 5 box per cell
- Strong priors on anchor size and direction (one size per class)
- Euler region proposal (complex angle regression)

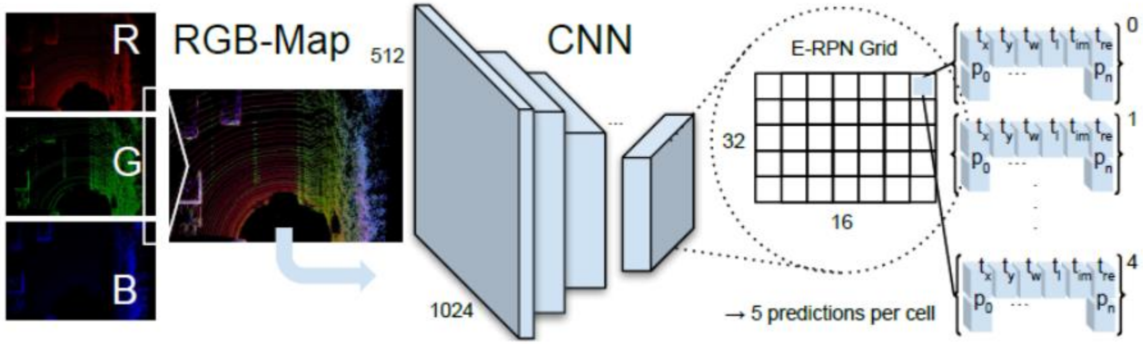


Fig. 2. Complex-YOLO Pipeline. We present a slim pipeline for fast and accurate 3D box estimations on point clouds. The RGB-map is fed into the CNN (see Tab. 1). The E-RPN grid runs simultaneously on the last feature map and predicts five boxes per grid cell. Each box prediction is composed by the regression parameters t (see Fig. 3) and object scores p with a general probability p_0 and n class scores $p_1 \ldots p_n$.

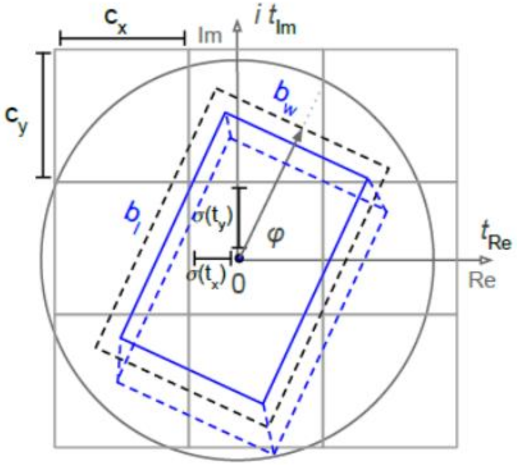


Fig. 3. 3D Bounding box regression. We predict oriented 3D bounding boxes based on the regression parameters shown in YOLOv2 [13], as well as a complex angle for box orientation. The transition from 2D to 3D is done by a predefined height based on each class.

VoxelNet

- Direct point based features, (no view projection)
- Proposals are end to end, voxel to anchor regression
- Only LIDAR voxel deployed for proposals
- Proposals are in highest voxel resolution
- Subsampling is on points, still needs efficient implementation
- Final proposals with fully connected layers

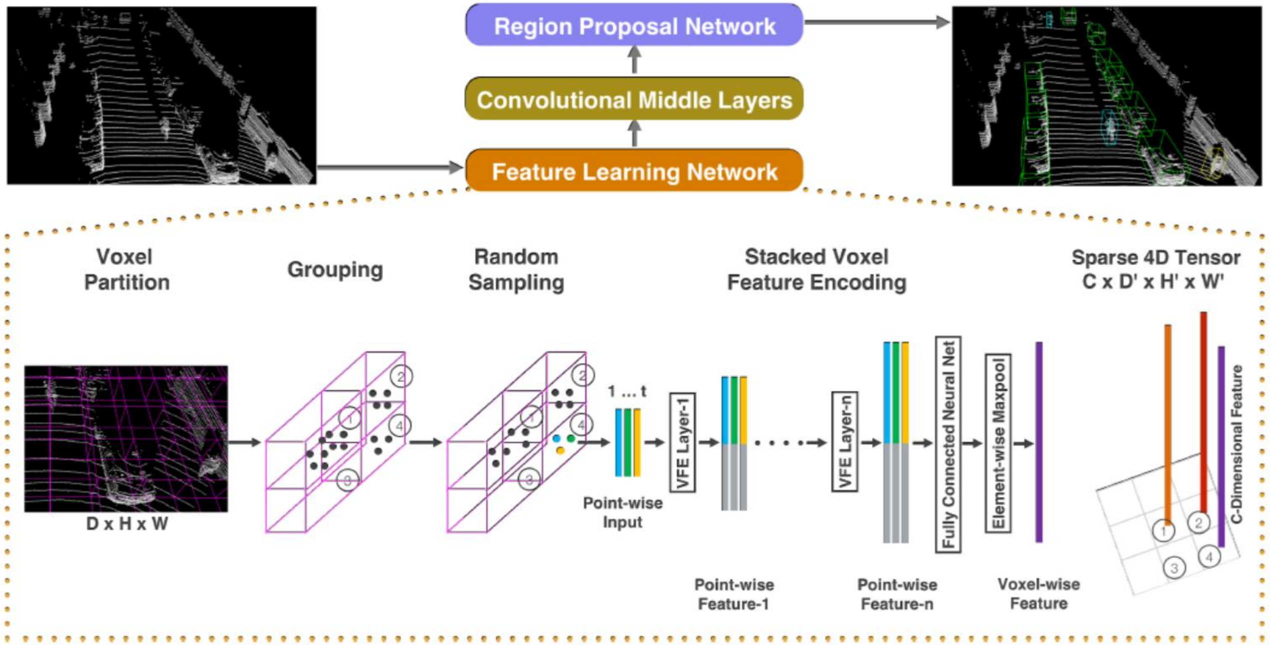


Figure 2. VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

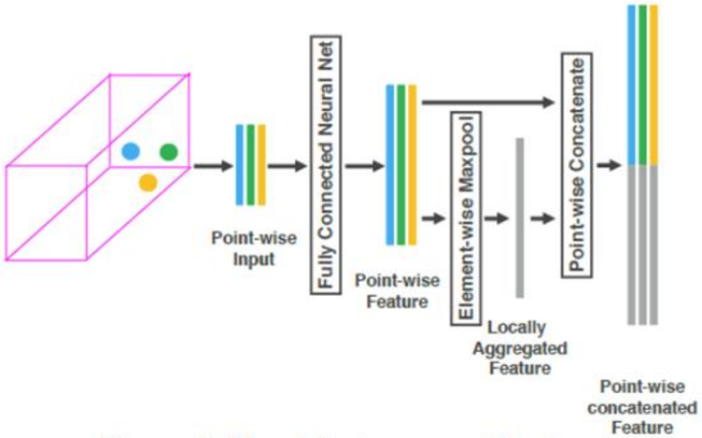


Figure 3. Voxel feature encoding layer.

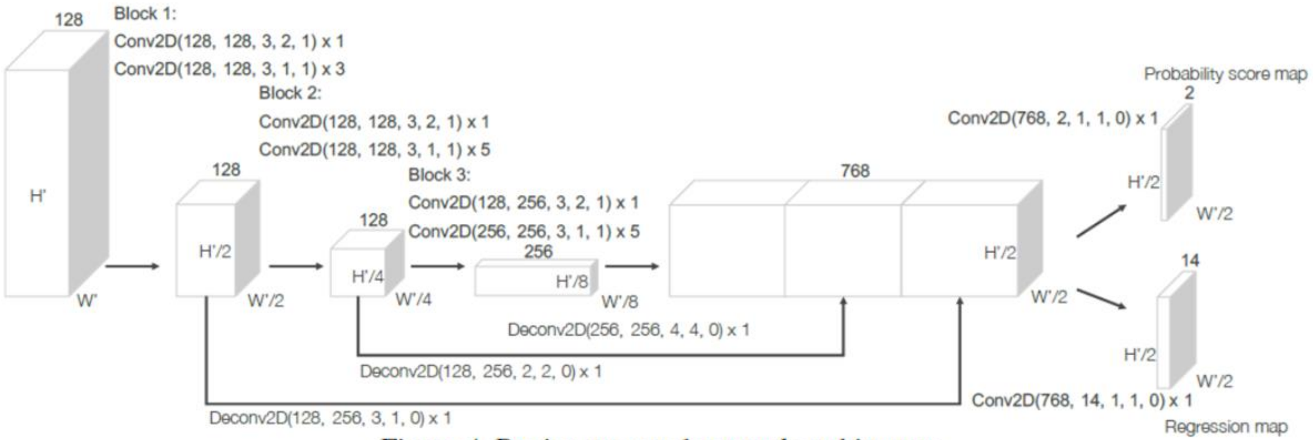


Figure 4. Region proposal network architecture.

PointFusion

- Proposal pipeline is connected to the end of FasterRCNN
- Proposals are the input
- Dense Fusion with corresponding point cloud features
- Final regression with fully connected layers

There is already a FasterRCNN somewhere here!

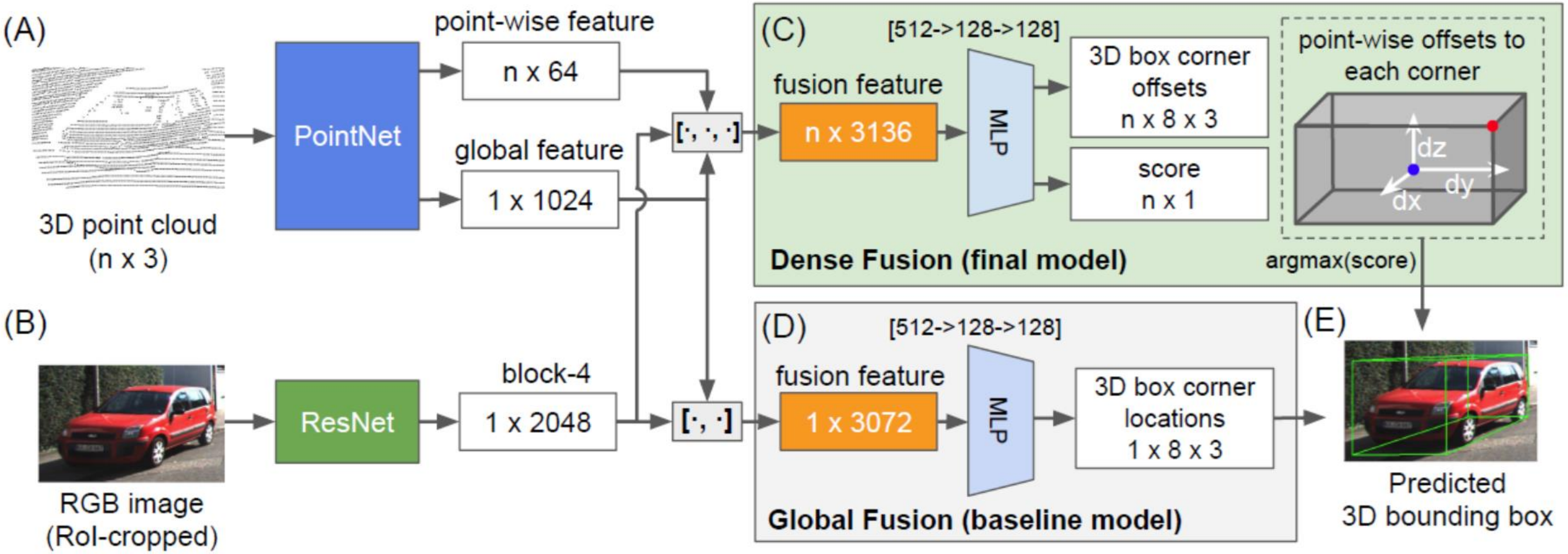


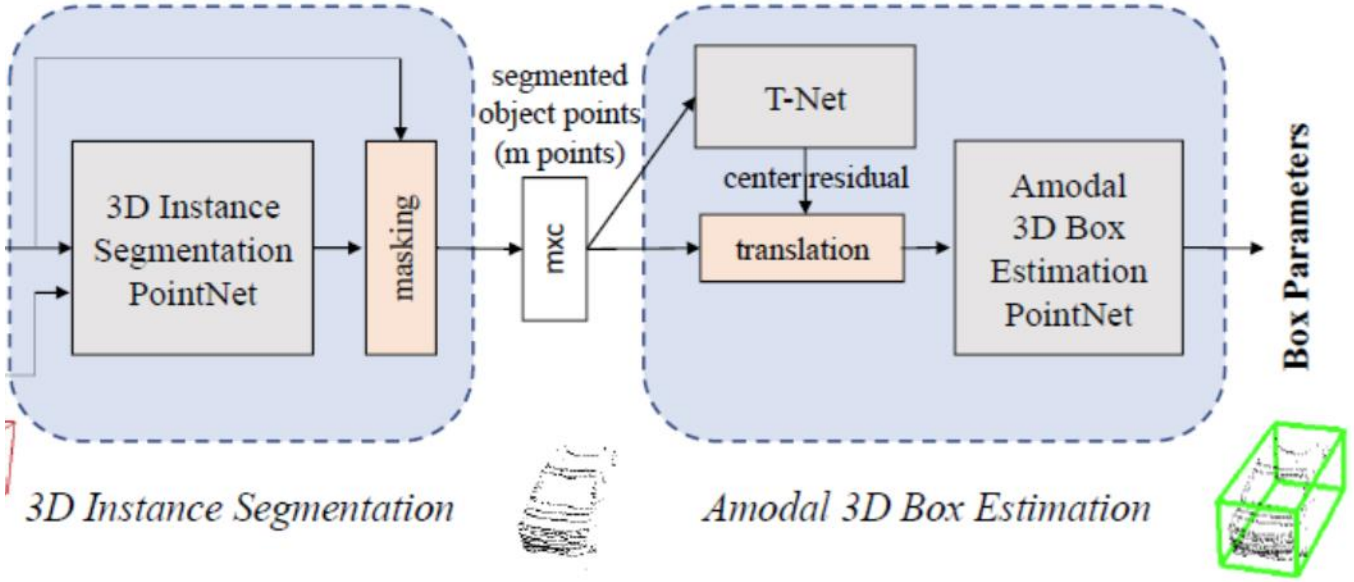
Figure 2. An overview of the dense PointFusion architecture. PointFusion has two feature extractors: a PointNet variant that processes raw point cloud data (A), and a CNN that extracts visual features from an input image (B). We present two fusion network formulations: a vanilla *global* architecture that directly regresses the box corner locations (D), and a novel *dense* architecture that predicts the spatial offset of each of the 8 corners relative to an input point, as illustrated in (C): for each input point, the network predicts the spatial offset (white arrows) from a corner (red dot) to the input point (blue), and selects the prediction with the highest score as the final prediction (E).

Frustrum PointNet

C. Details on RGB Detector (Sec 4.1)

For 2D RGB image detector, we use the encoder-decoder structure (e.g. DSSD [9], FPN [20]) to generate region proposals from multiple feature maps using focal loss [21] and use Fast R-CNN [12] to predict final 2D detection bounding boxes from the region proposals.

To make the detector faster, we take the reduced VGG [32] base network architecture from SSD [22], sample half of the channels per layer and change all max pooling layers to convolution layers with 3×3 kernel size and stride of 2. Then we fine-tune it on ImageNet CLS-LOC dataset



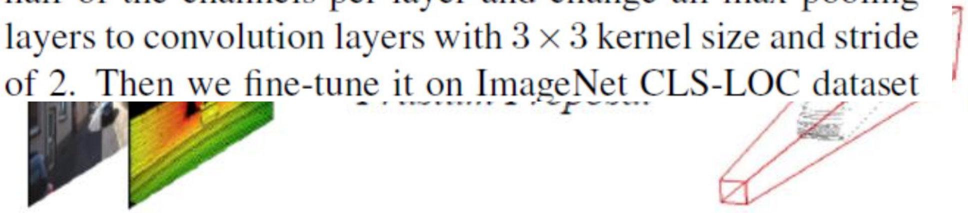
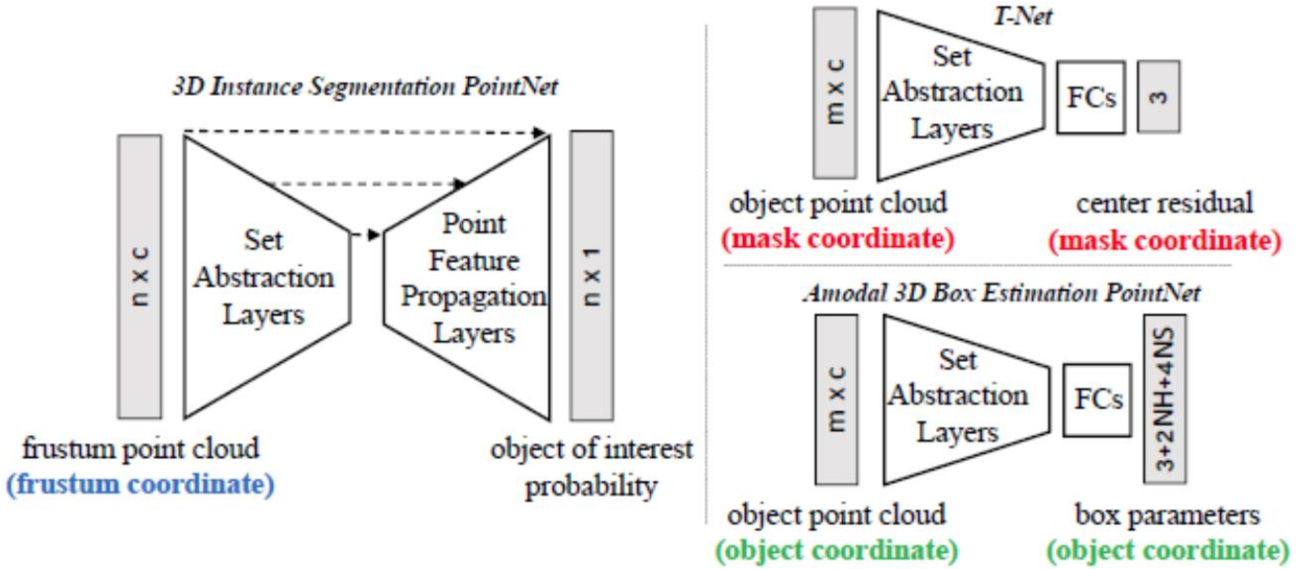


Figure 2. **Frustum PointNets for 3D object dete** … and classify
their content. 2D regions are then lifted to 3D and … vith n points
and c channels of XYZ, intensity etc. for each poi … Based on the
segmented object point cloud ($m \times c$), a light-weig … heir centroid
is close to amodal box center. At last the box esti … istrations on
coordinate systems involved and network input, ou



What next?

We have not mention anything about:

C. Details on RGB Detector (Sec 4.1)

For 2D RGB image detector, we use the encoder-decoder structure (e.g. DSSD [9], FPN [20]) to generate region proposals from multiple feature maps using focal loss [21] and use Fast R-CNN [12] to predict final 2D detection bounding boxes from the region proposals.

To make the detector faster, we take the reduced VGG [32] base network architecture from SSD [22], sample half of the channels per layer and change all max pooling layers to convolution layers with 3×3 kernel size and stride of 2. Then we fine-tune it on ImageNet CLS-LOC dataset

RetinaNET