

CNN  

			Tasks						
			ADAS						
			Self Driving						
			Localizati on	Perception	Planning/ Control	Driver state	Vehicle Diagnosis	Smart factory	
Methods	Traditional	Non-machine Learning		GPS, SLAM		Optimal control			
		Machine-Learning based method	Supervised	SVM MLP		Pedestrian detection (HOG+SVM)			
	CNN				Detection/ Segmentat ion/Classif ication	End-to- end Learning			
	RNN (LSTM)				Dry/wet road classificati on	End-to- end Learning	Behavior Prediction/ Driver identificati on		*
	DNN							*	*
	Reinforcement				*				
	Unsupervised							*	

Classification/Detection/Segmentation

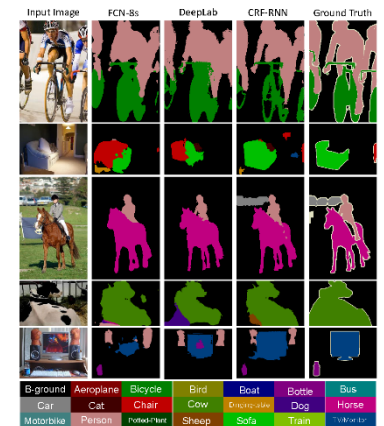
Image
Classification



Object
Detection



Semantic
Segmentation

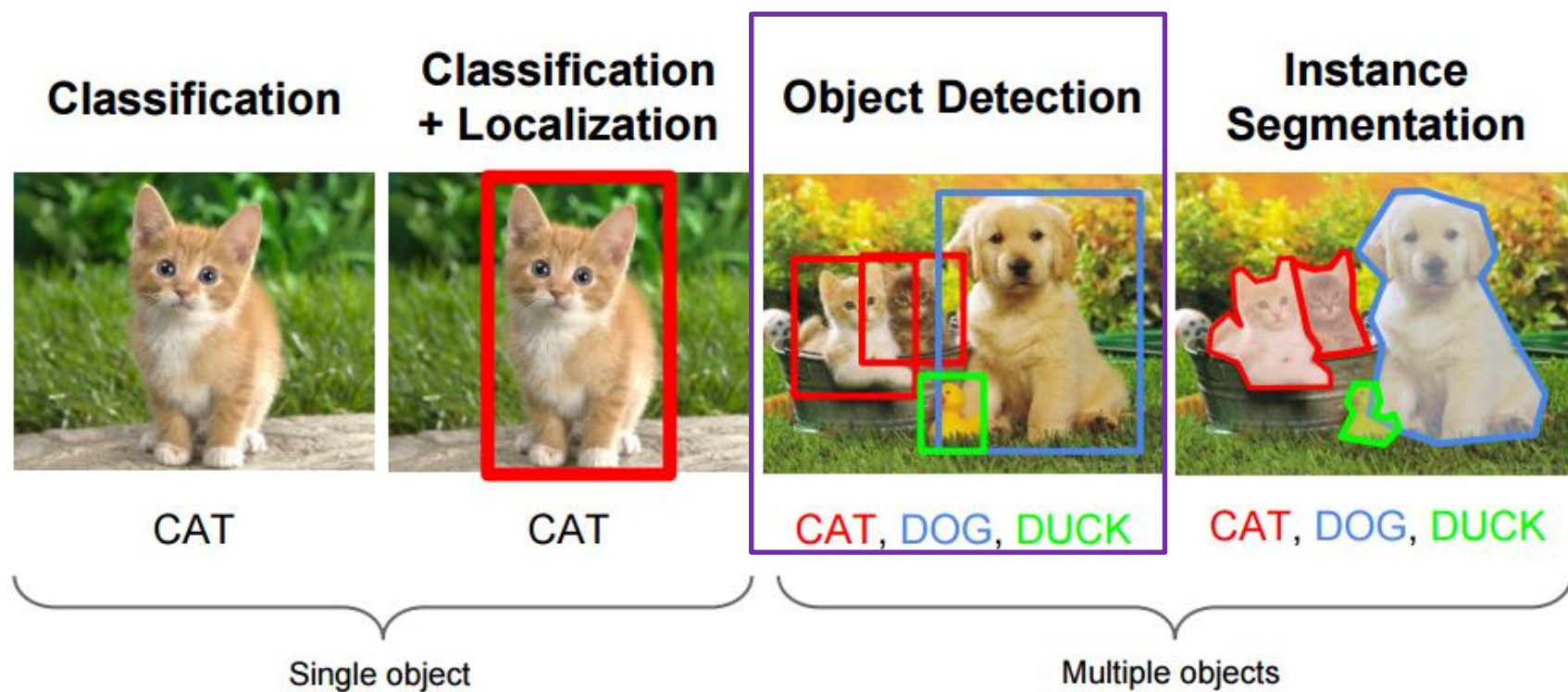


Detection/Segmentation



OBJECT DETECTION

Computer Vision Tasks



Detection as (classification+regression)?



DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers

Detection as (classification+regression)?

- Need variable sized outputs



CAT, (x, y, w, h)

CAT, (x, y, w, h)

....

CAT (x, y, w, h)

= many numbers

Detection as classification

- Detection as classification (e.g., sliding windows)
 - Problem: Need to test many positions and scales
 - Solution: If your classifier is fast enough, just do it!



CAT? NO
DOG? NO



CAT? YES!
DOG? NO



CAT? NO
DOG? NO

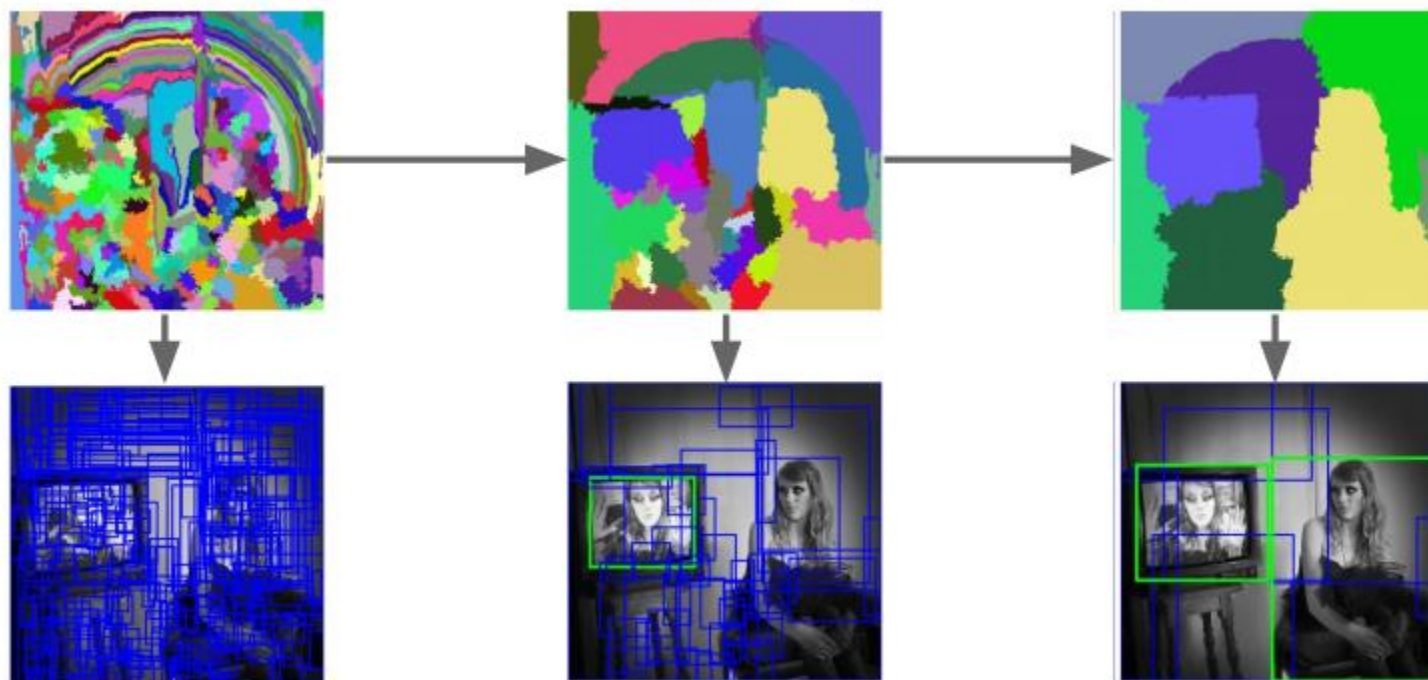
- τ

- Detection with a CNN classifier
 - Problem: Need to test many positions and scales, and use a computationally demanding classifier
 - Solution: Only look at a tiny subset of possible positions

Region Proposals

Bottom-up segmentation, merging regions at multiple scales

Convert
regions
to boxes



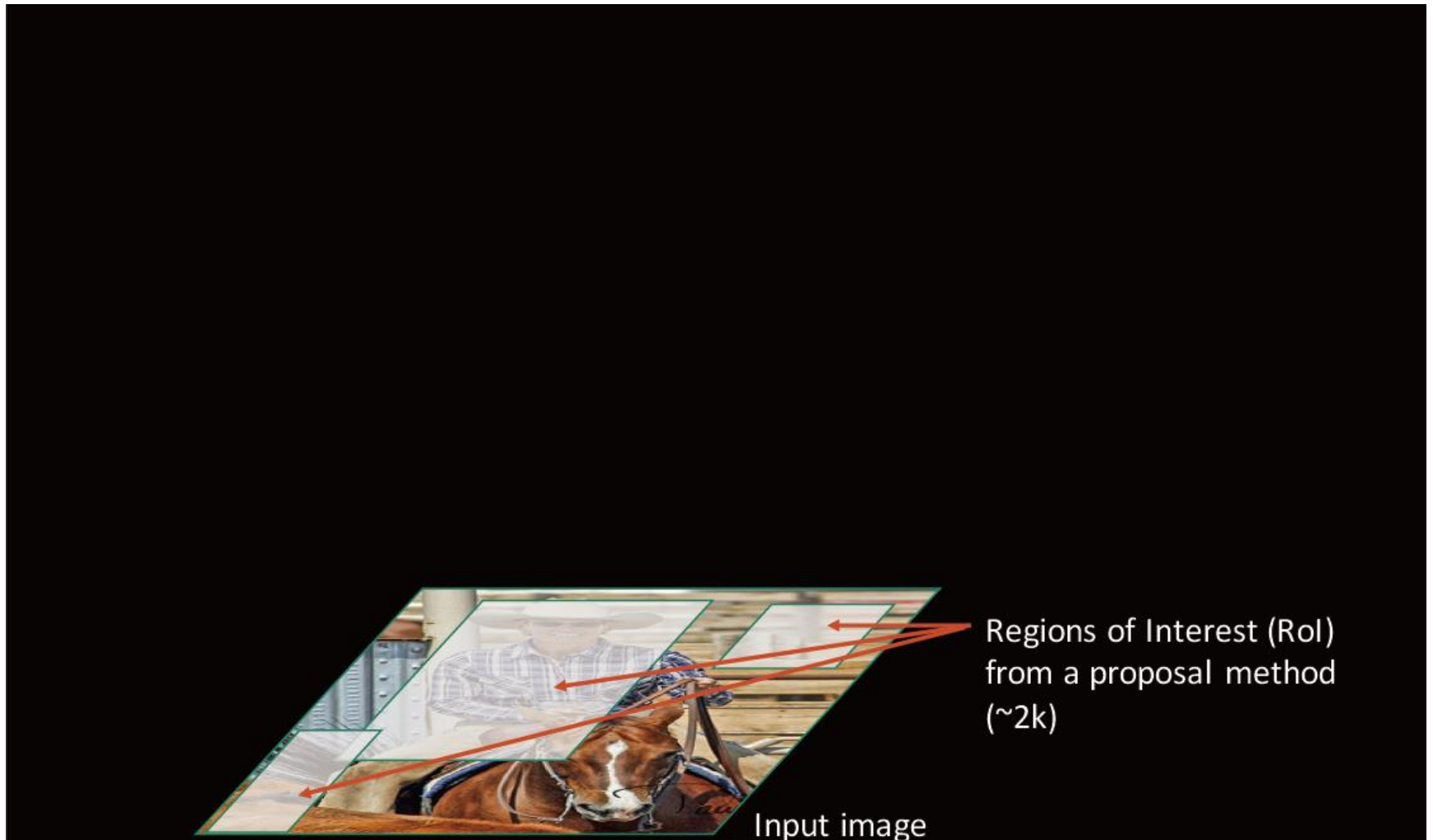
R-CNN (REGIONS WITH CNN FEATURES)

R-CNN

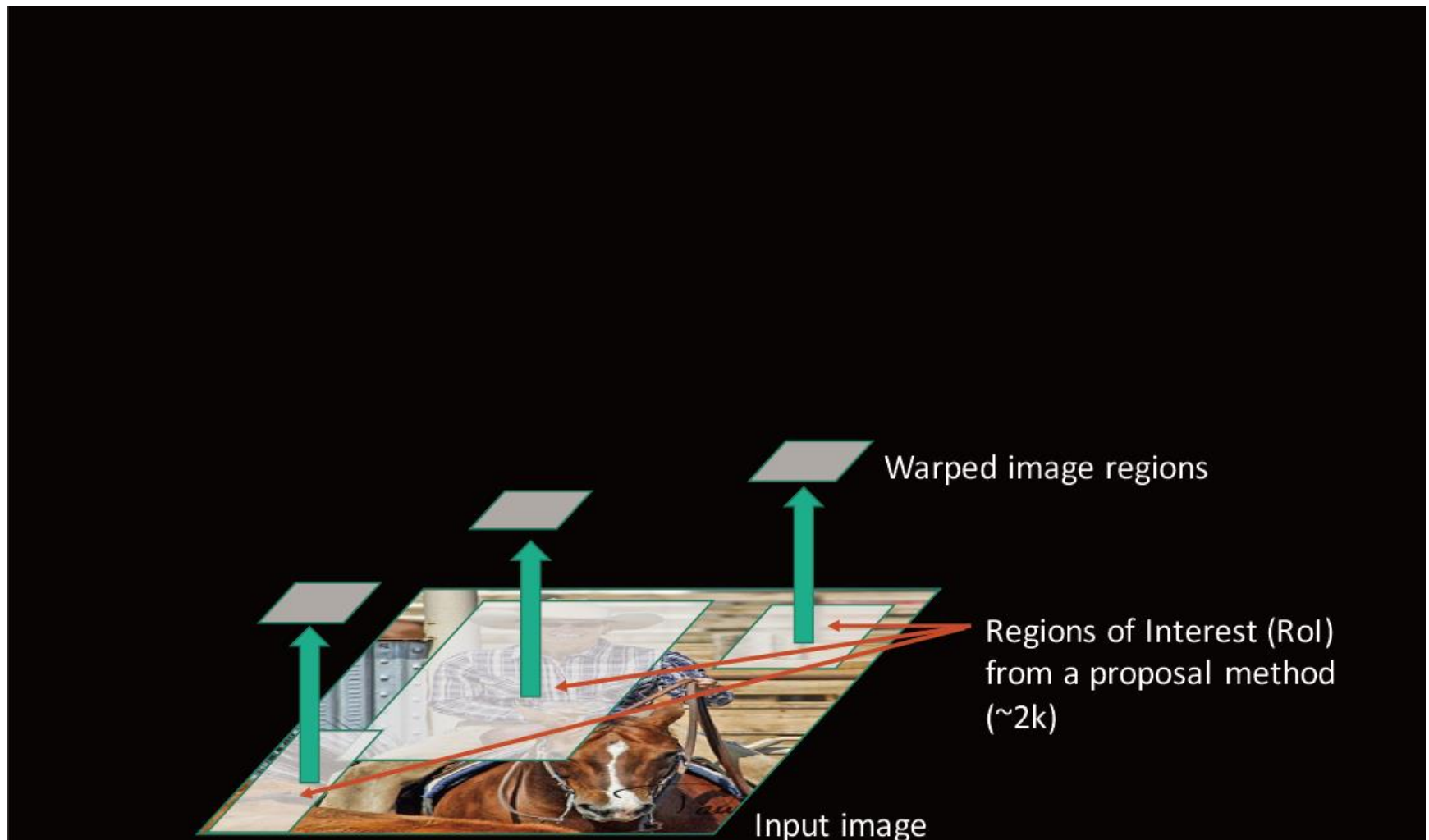


Input image

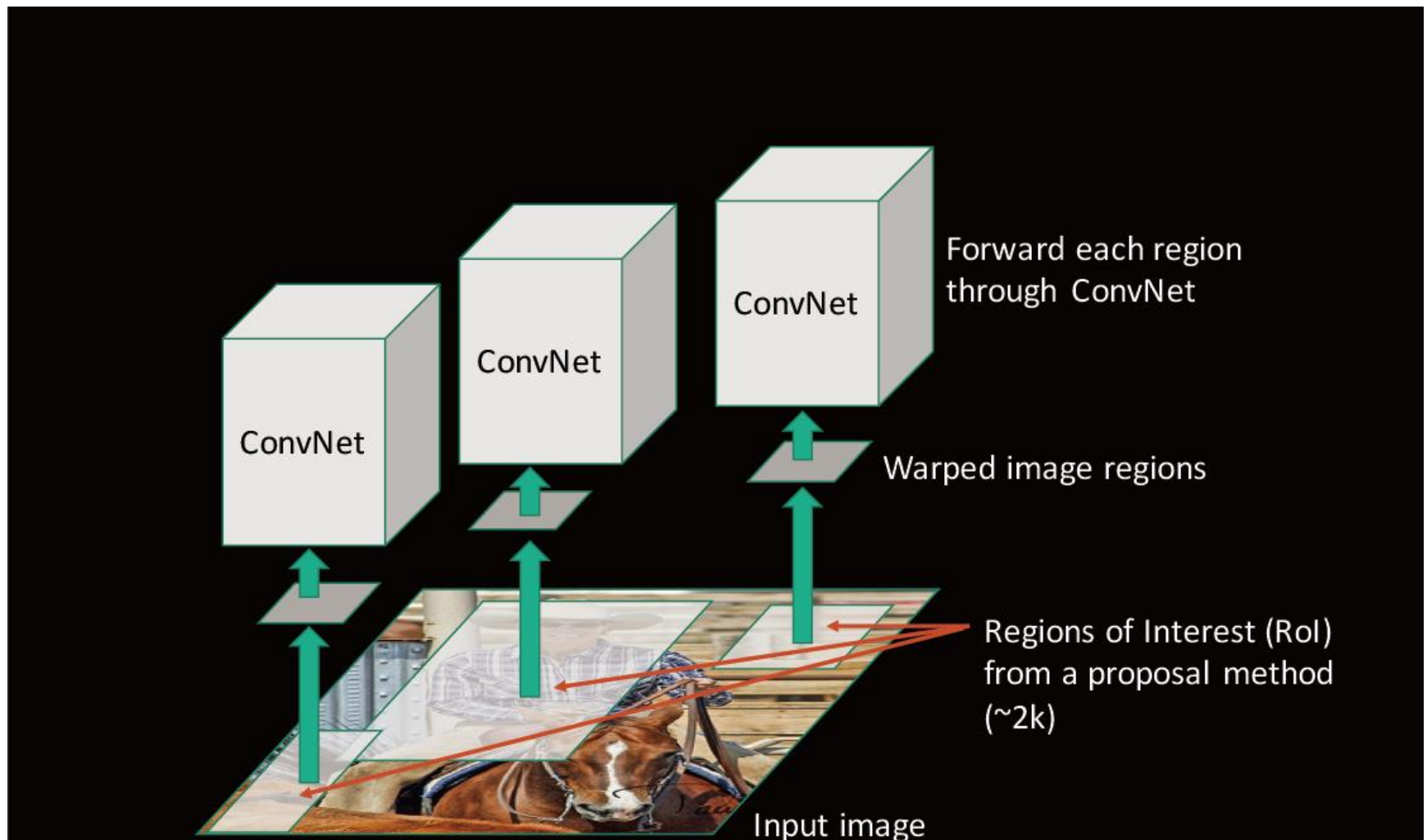
R-CNN



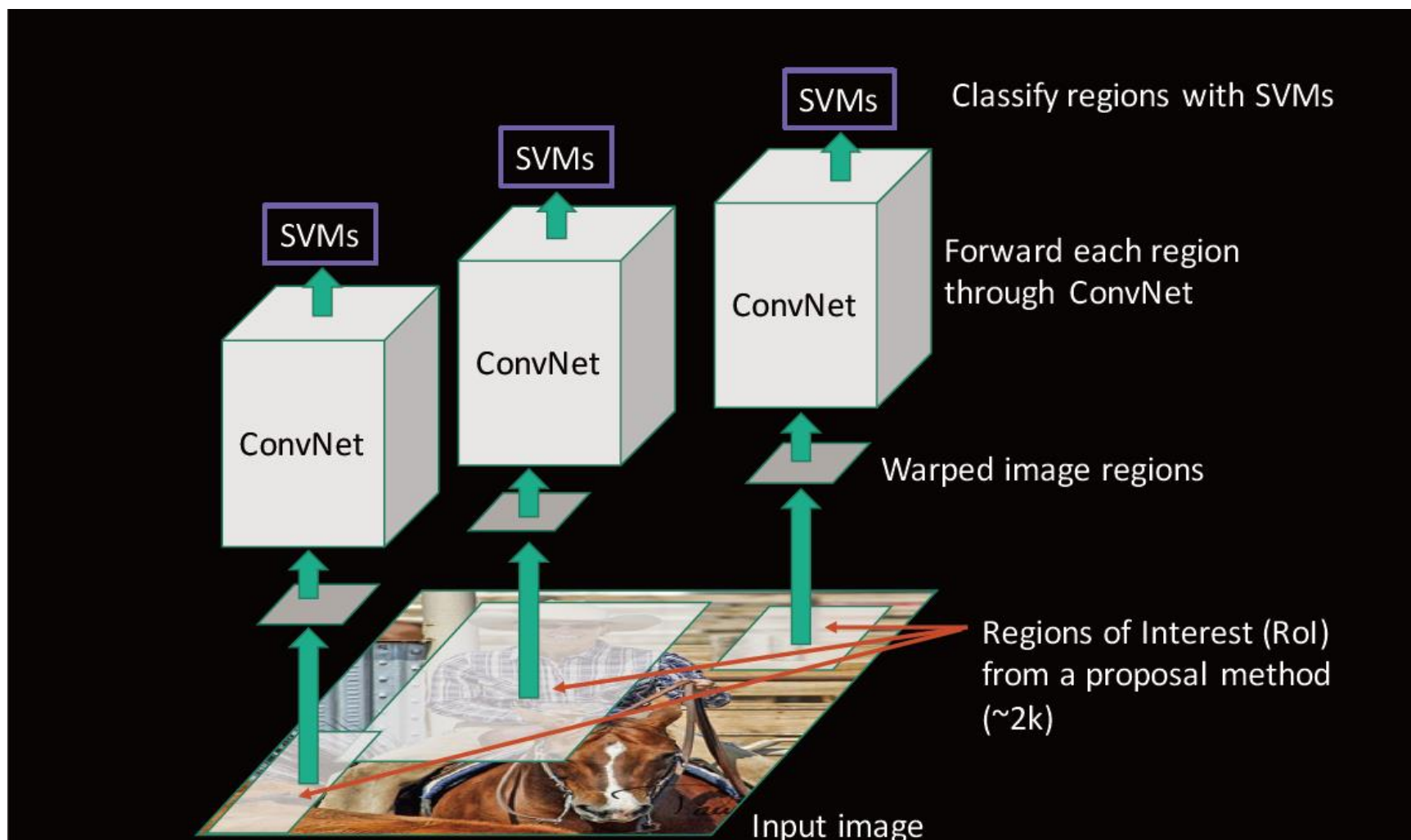
R-CNN



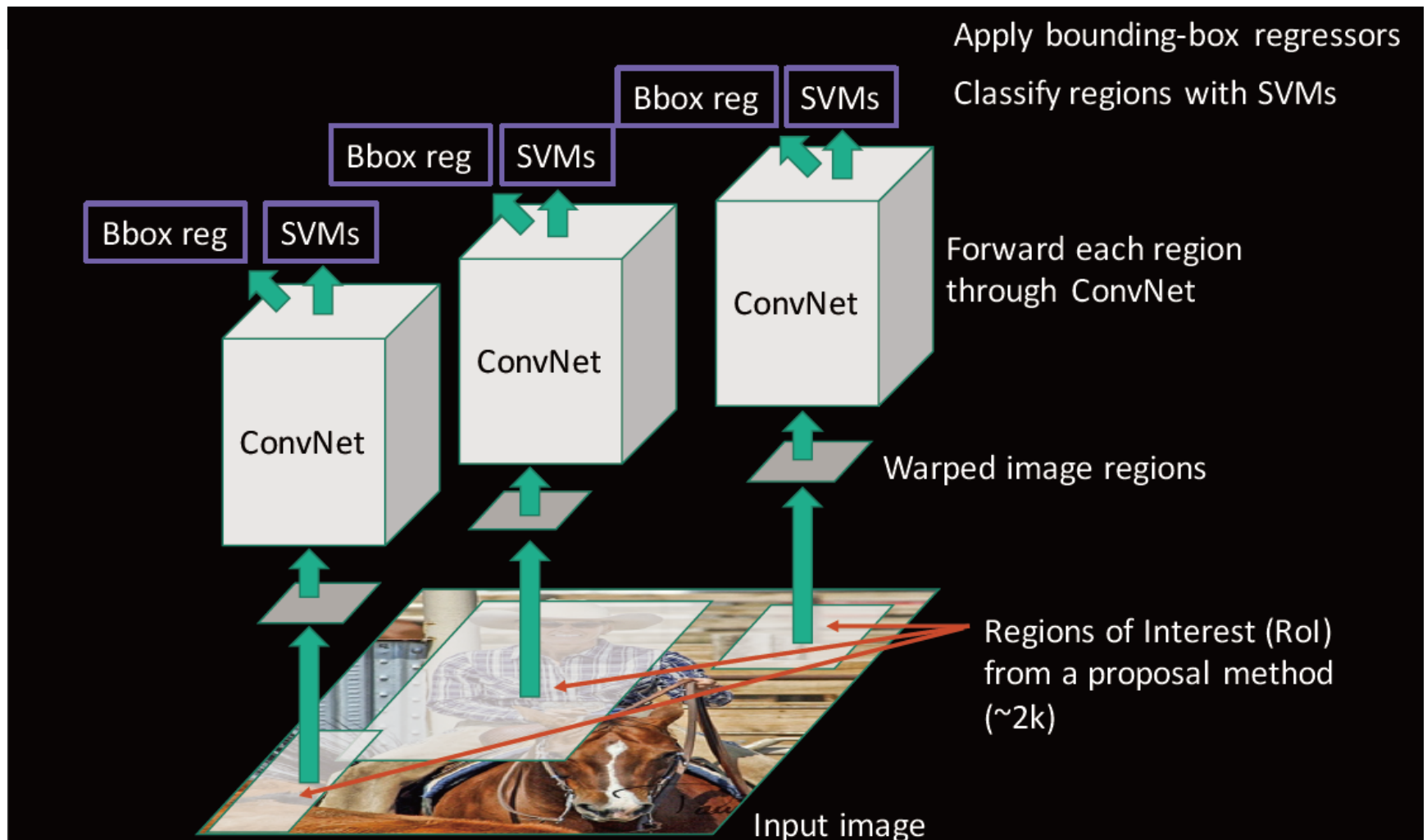
R-CNN



R-CNN



R-CNN

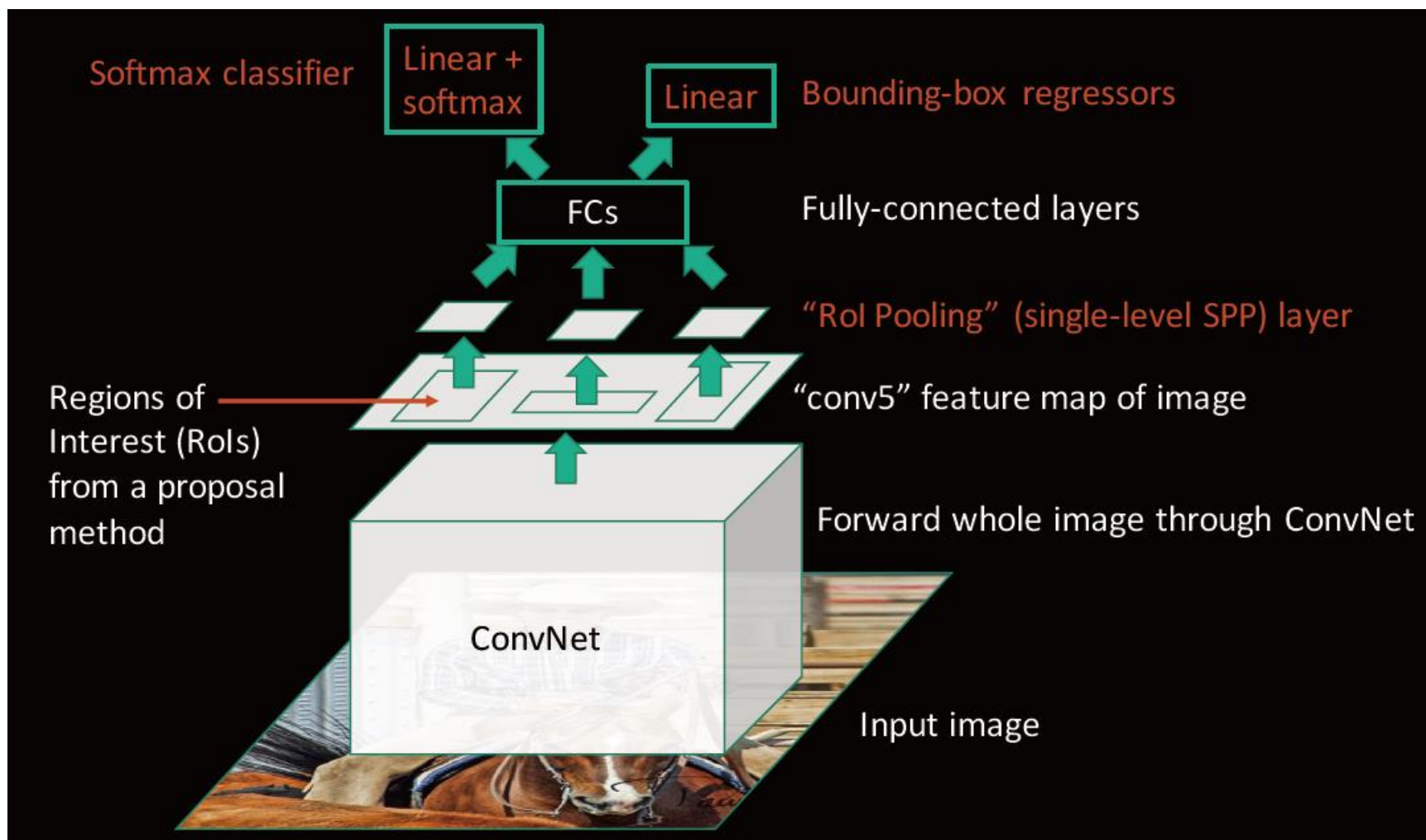


Limitations of R-CNN

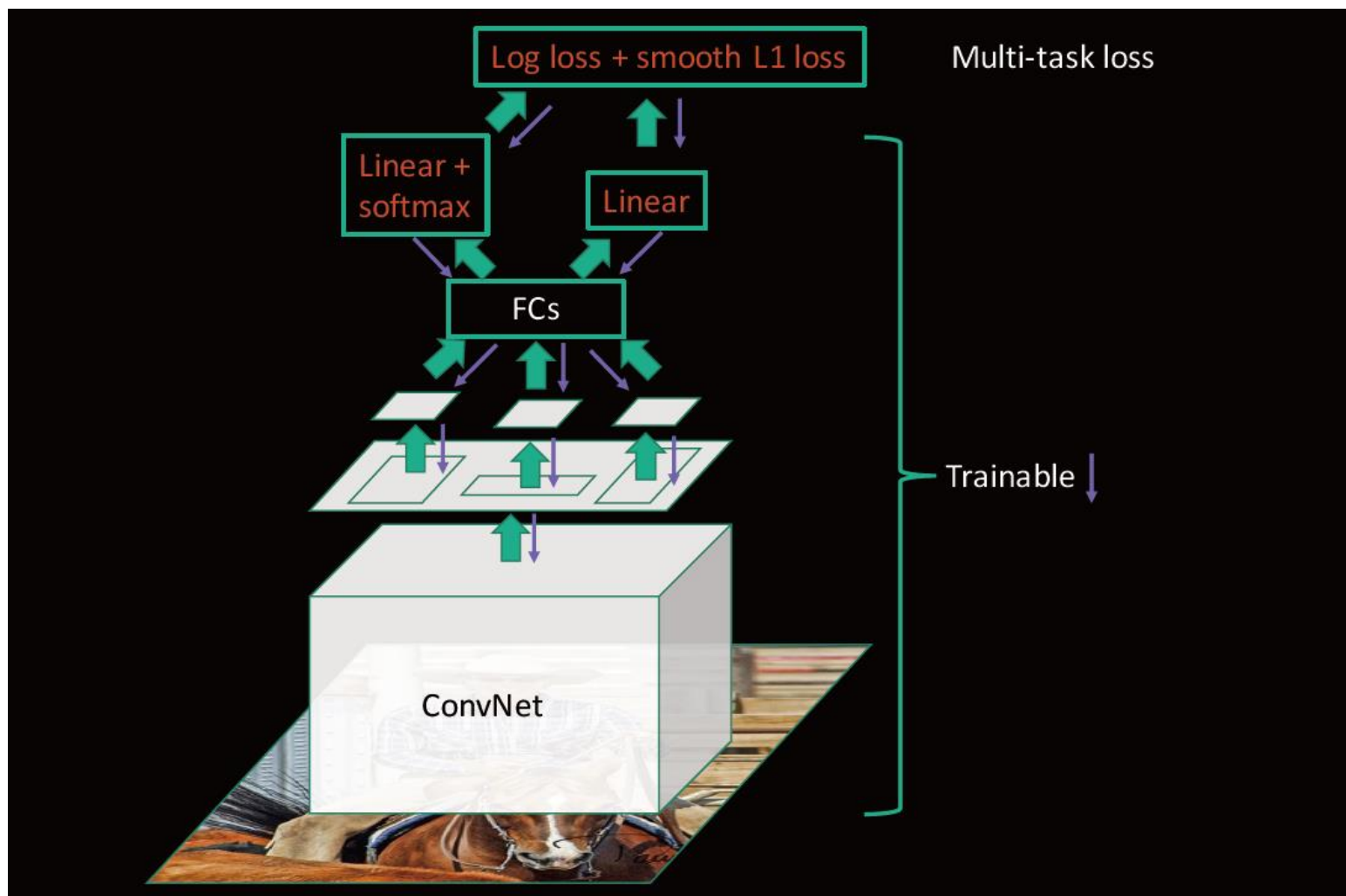
- Ad hoc training objectives
 - Fine tune network with softmax classifier (log loss)
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
 - 47s / image with VGG16

FAST R-CNN

Fast R-CNN (test time)



Fast R-CNN (training time)



Comparison

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
Speedup	8.8x	1x
Test time/image	0.32s	47.0s
Test speedup	146x	1x
mAP	66.9	66.0

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

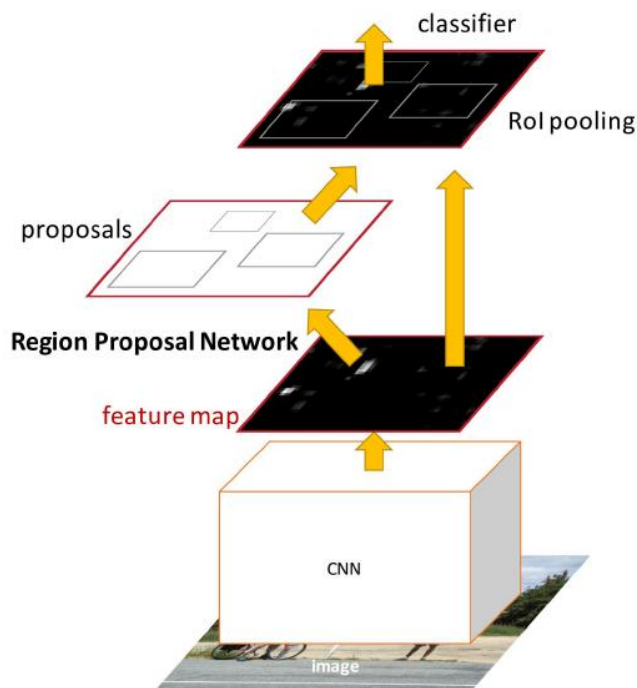
Fast R-CNN

- Pros
 - End-to-end training of deep ConvNets for detection
 - Fast training times
- Cons
 - Out-of-network region proposals
 - Selective search: 2s/image
- Solution
 - Test-time speeds don't include region proposals
 - Just make the CNN do region proposals too!

FASTER R-CNN

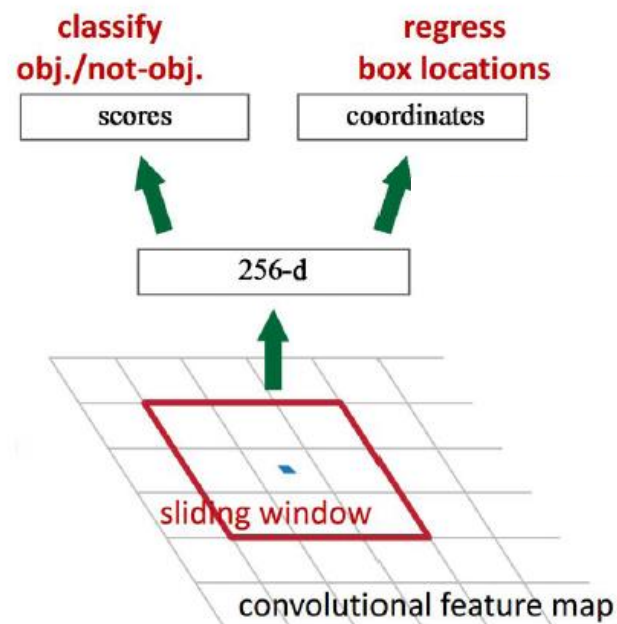
Faster RCNN

- Insert a **Region Proposal Network (RPN)** after the last convolutional layer
- RPN trained to produce region proposals directly
 - no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN



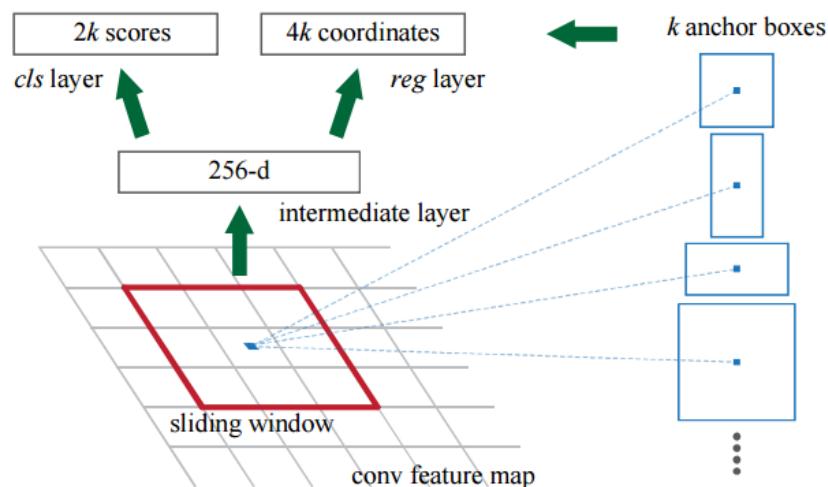
Faster R-CNN: RPN

- Slide a small window on the feature map
- Build a small network for:
 - classifying object or not-object, and
 - regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window



Faster R-CNN

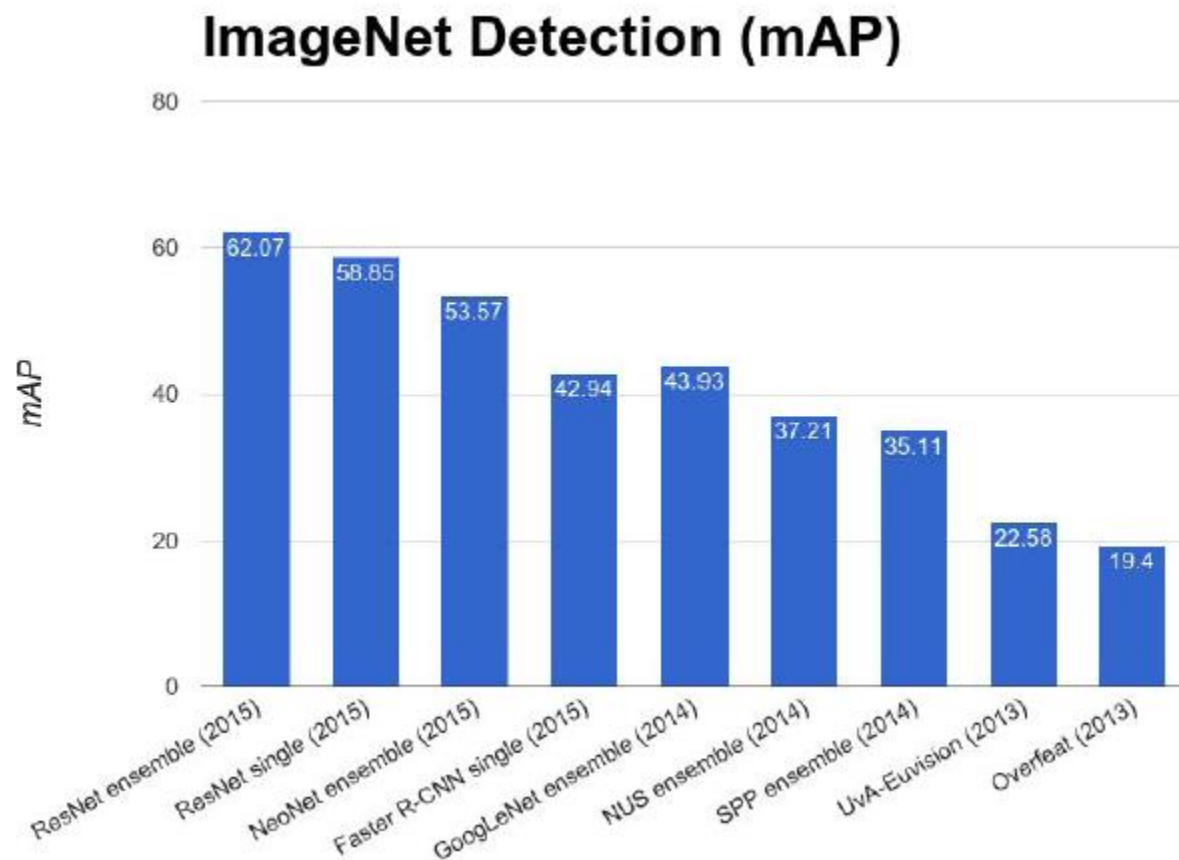
- Use k ($=9$) **anchor boxes** at each location
- Anchors are translation invariant: use the same ones at every location
- Regression gives offsets from anchor boxes
- Classification gives the probability that each (regressed) anchor shows an object



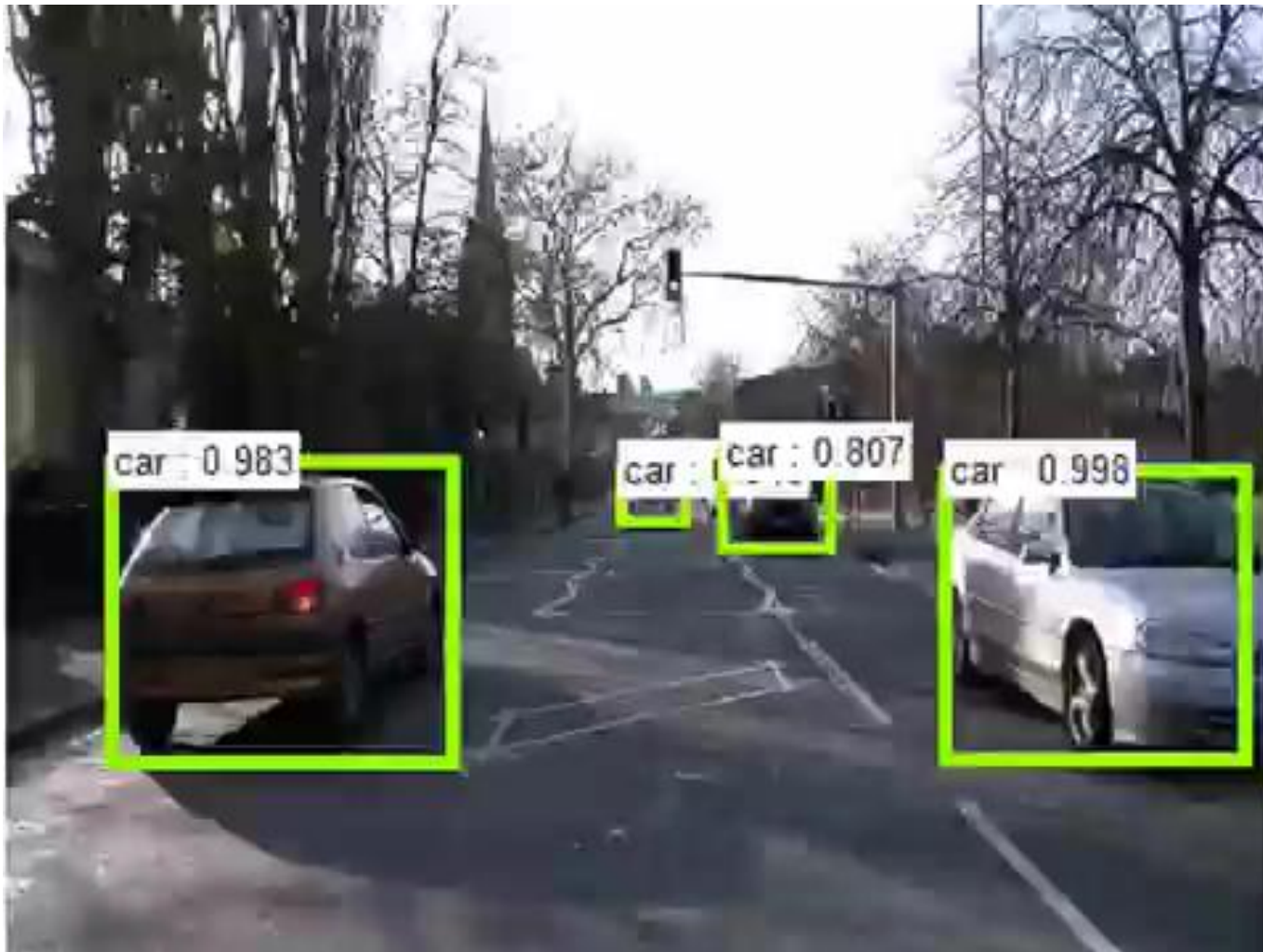
Results

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
Speedup	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9

ImageNet Detection 2013 - 2015



Object detection in the wild by Faster R-CNN + ResNet



Code links

- R-CNN
 - Caffe + Matlab (<https://github.com/rbgirshick/rcnn>)
- Faster R-CNN
 - Caffe + Matlab (<https://github.com/rbgirshick/fast-rcnn>)
- Faster R-CNN
 - Caffe + Matlab (https://github.com/ShaoqingRen/faster_rcnn)
 - Caffe + Python (<https://github.com/rbgirshick/py-faster-rcnn>)
- YOLO
 - <http://pjreddie.com/darknet/yolo/>

OBJECT SEGMENTATION

Computer Vision Tasks

Classification



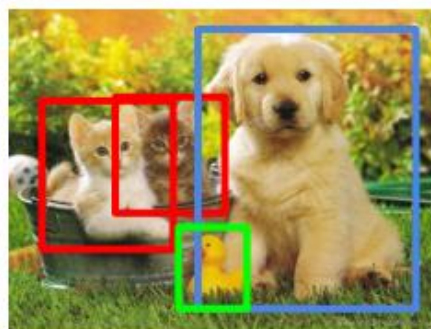
CAT

**Classification
+ Localization**



CAT

Object Detection



CAT, DOG, DUCK

Segmentation



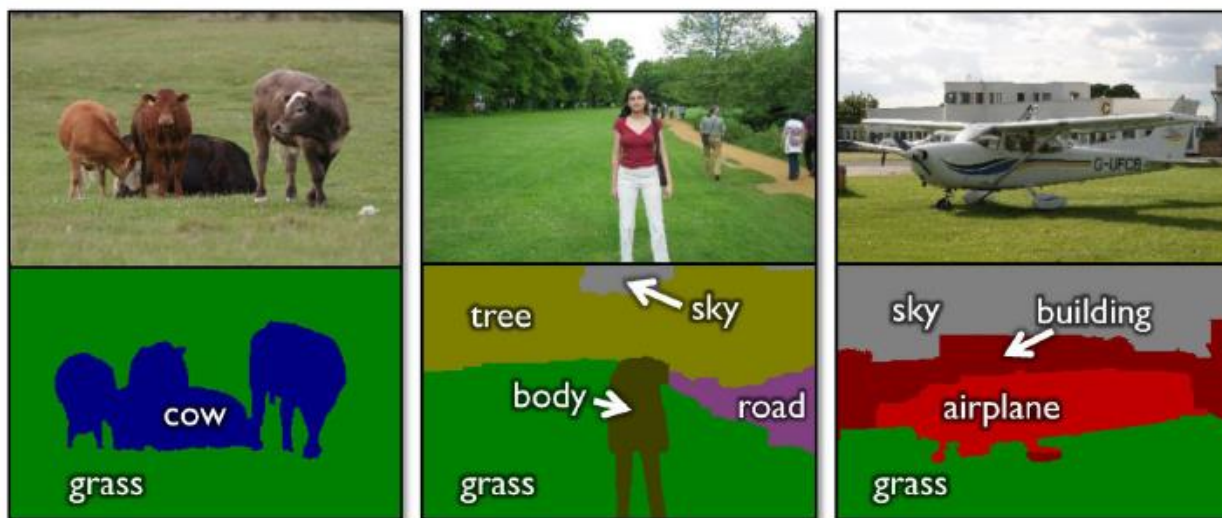
CAT, DOG, DUCK

Single object

Multiple objects

Semantic segmentation

- Label every pixel
- Don't differentiate instances

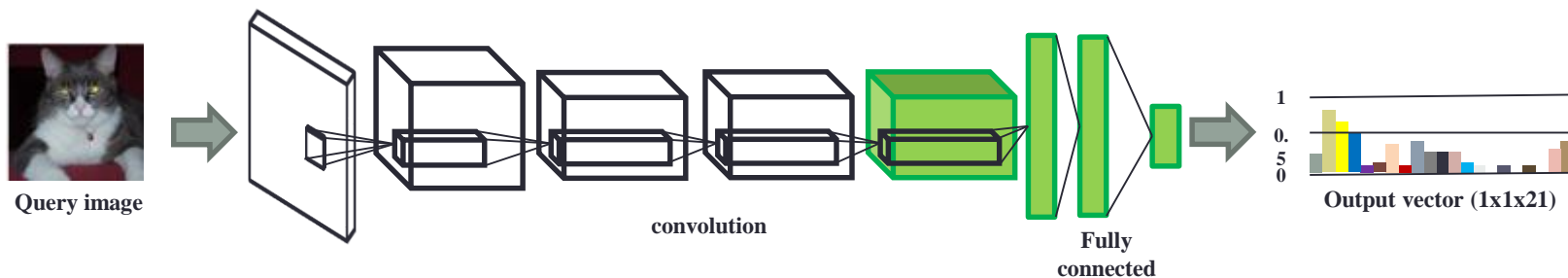


object classes	building	grass	tree	cow	sheep	sky	airplane	water	face	car	
	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat

FULLY CONNECTED LAYERS AS CONVOLUTION LAYERS

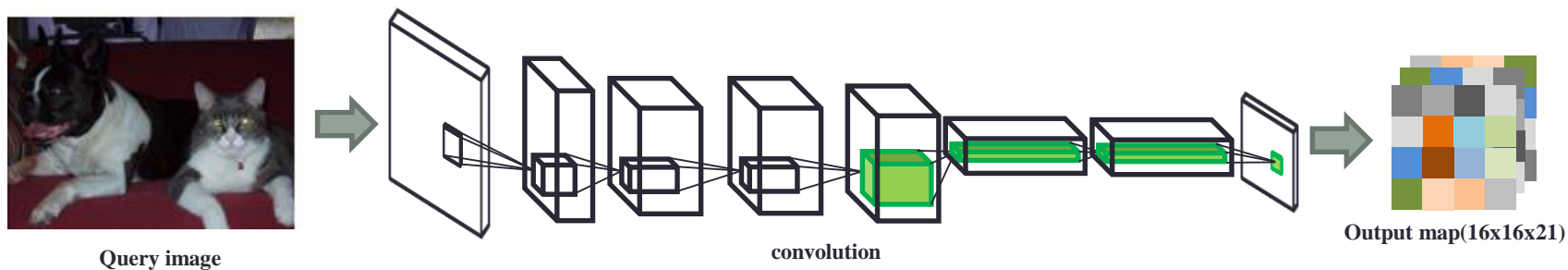
FC layer as Conv layer

- Image classification



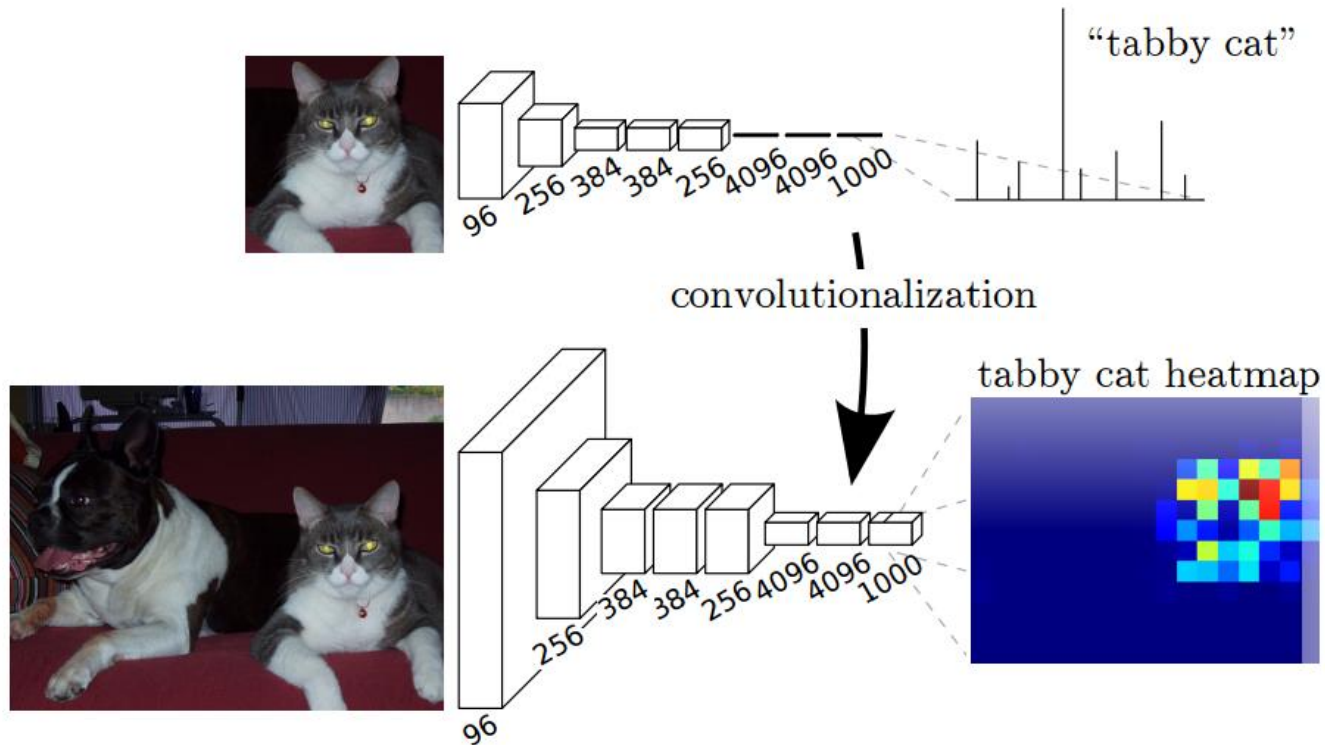
- Semantic segmentation

- Given an input image, obtain pixel-wise segmentation mask using a deep Convolutional Neural Network (CNN)

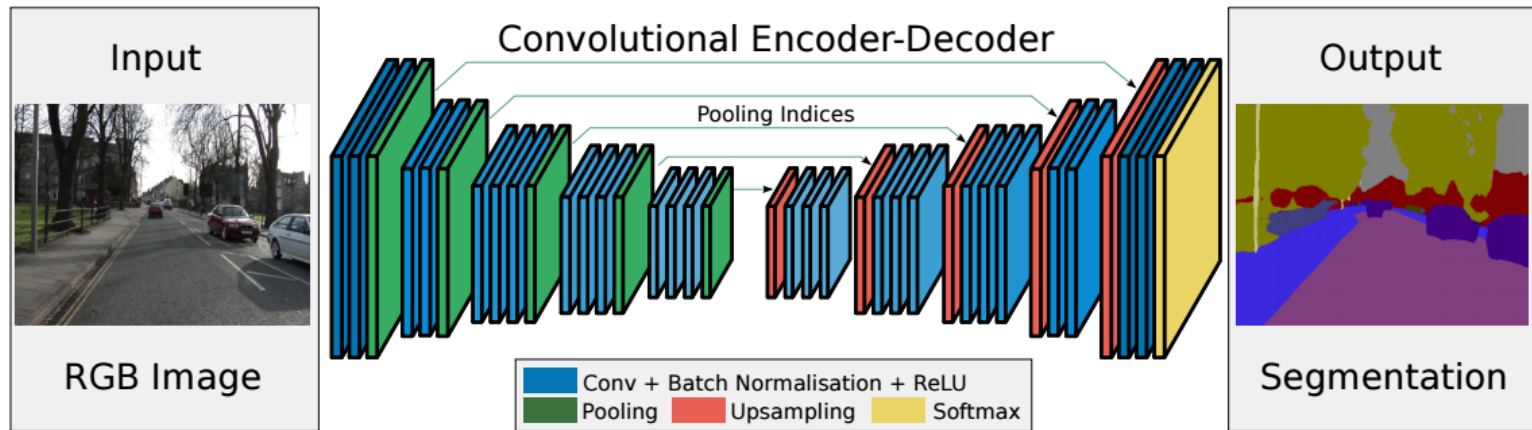


FC layer as Conv layer

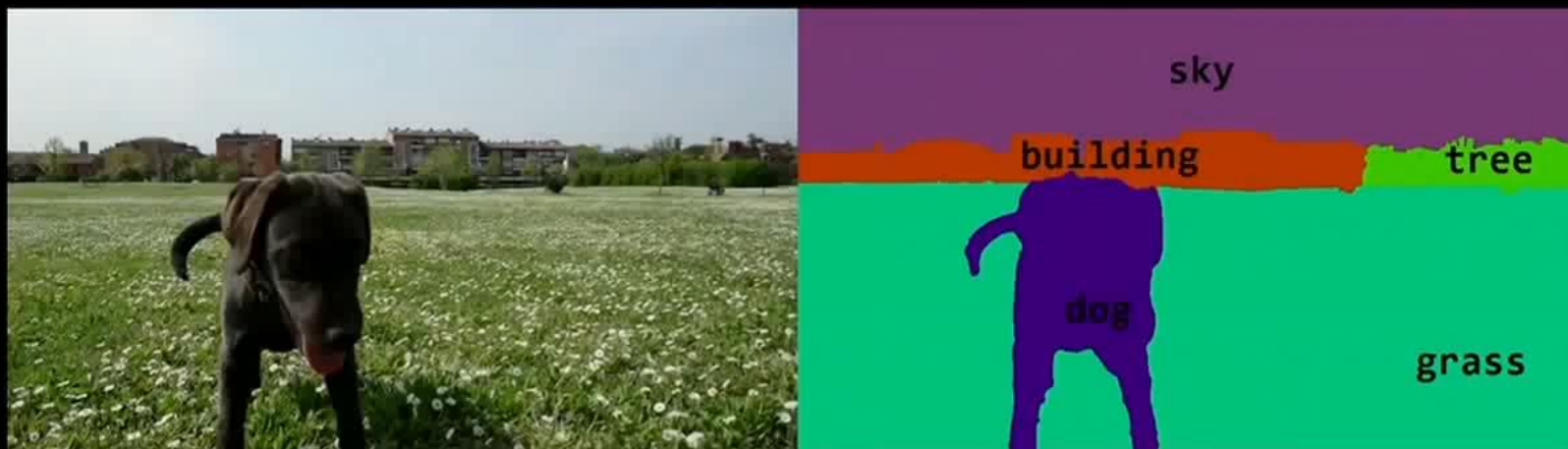
- Transforming fully connected layers into convolution layers enables a classification net to output a heatmap



Encoder/Decoder



Microsoft Deep Learning Semantic Image Segmentation



Today

Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes

Tobias Pohlen, Alexander Hermans,
Markus Mathias, Bastian Leibe

Visual Computing Institute, Computer Vision Group
RWTH Aachen University



MORE APPLICATIONS

BEHAVIOR REFLEX APPROACH

			Tasks						
			ADAS						
			Self Driving						
			Localizati on	Perception	Planning/ Control	Driver state	Vehicle Diagnosis	Smart factory	
Methods	Traditional	Non-machine Learning		GPS, SLAM		Optimal control			
		Machine-Learning based method	Supervised	SVM MLP		Pedestrian detection (HOG+SVM)			
	CNN				3D object Detection	End-to- end Learning			
	RNN (LSTM)				Dry/wet road classificati on	End-to- end Learning	Behavior Prediction/ Driver identificati on		*
	DNN							*	*
	Reinforcement				*				
	Unsupervised							*	

DAVE-1

DAVE-1

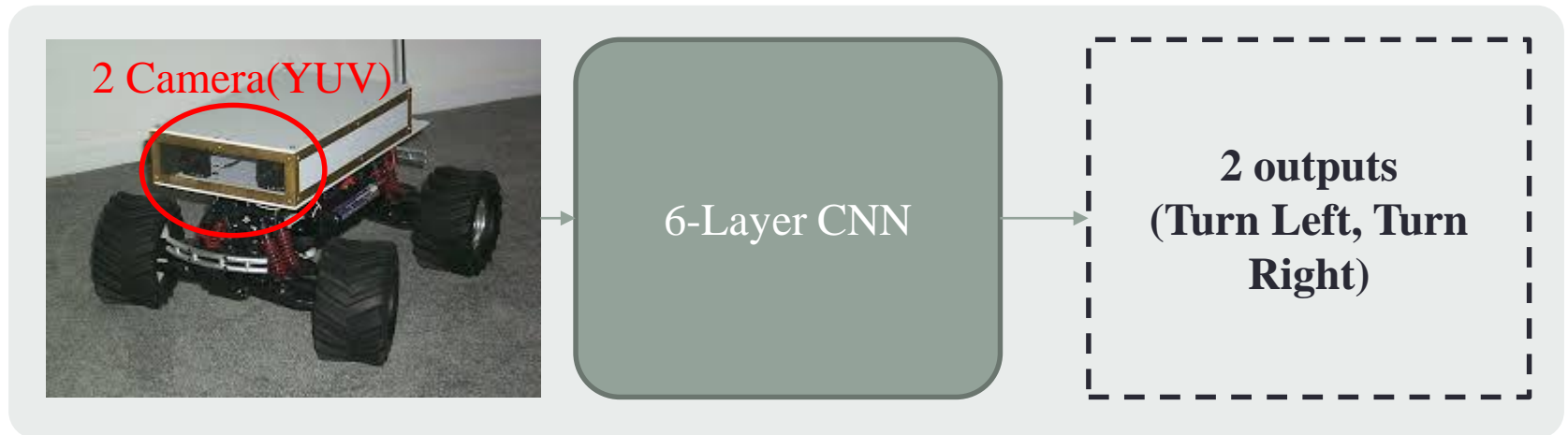
- Development period : 2003-2004
 - CNN structure is different with today's usual CNN structure
 - Input : two camera (YUV channel)
 - 320 x 240 input image are resized to 149 x 58 using LPF
 - Learning handle angle from remote control
 - Learning 4 days using Intel Xeon 3.0GHz CPU



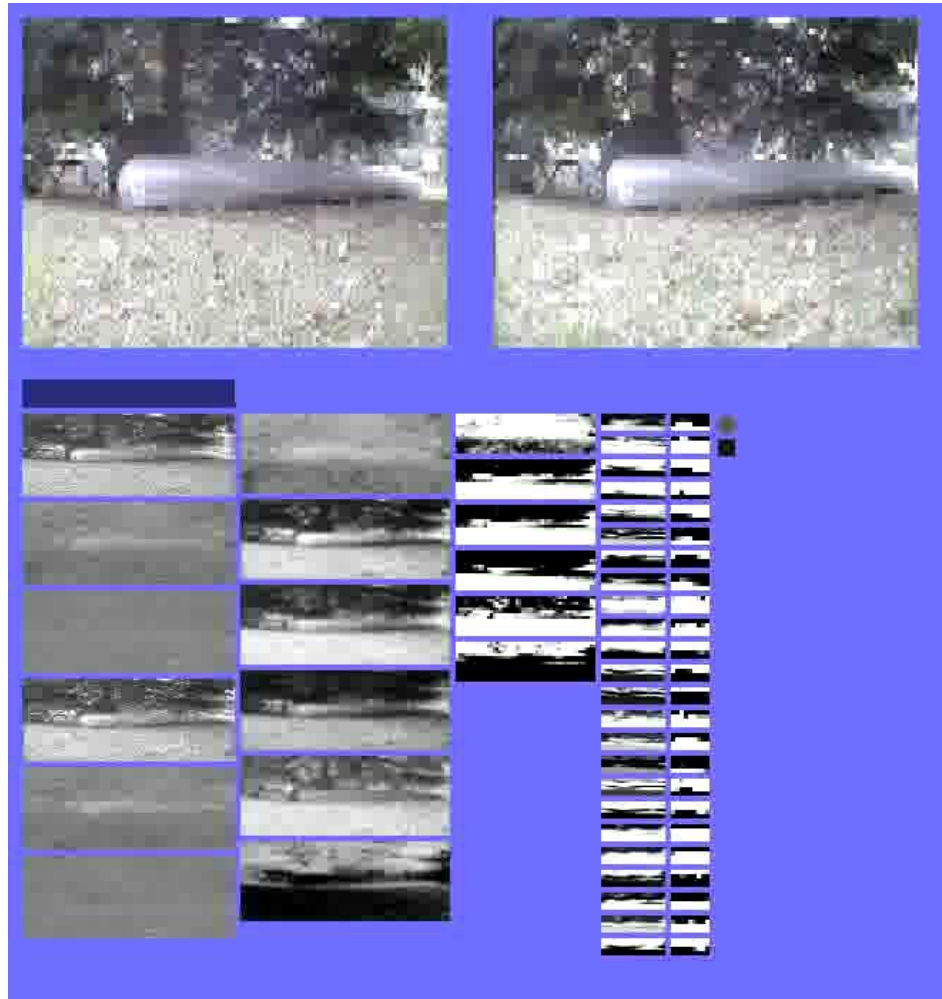
DAVE robot

- Reference
 - Muller, Urs, et al. "Off-road obstacle avoidance through end-to-end learning." *Advances in neural information processing systems*. 2005.

Overview



Result



Input images and CNN Layer results



Actual driving result

END TO END LEARNING FOR SELF-DRIVING CARS

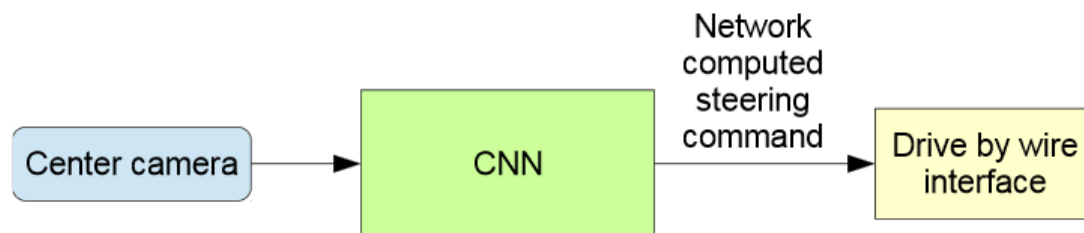
2016 Apr, arXiv, NVIDIA New Jersey



- The Project DAVE 2
- NVIDIA started the Project @2015

Main Idea

- Single Camera input
- Steering command output through a trained CNN model



Primary Motivation

- The driving system **avoid**:
 - the need to recognize human-designated features like lane markings, guard rails..
 - having a collection of “if, then, else” rules

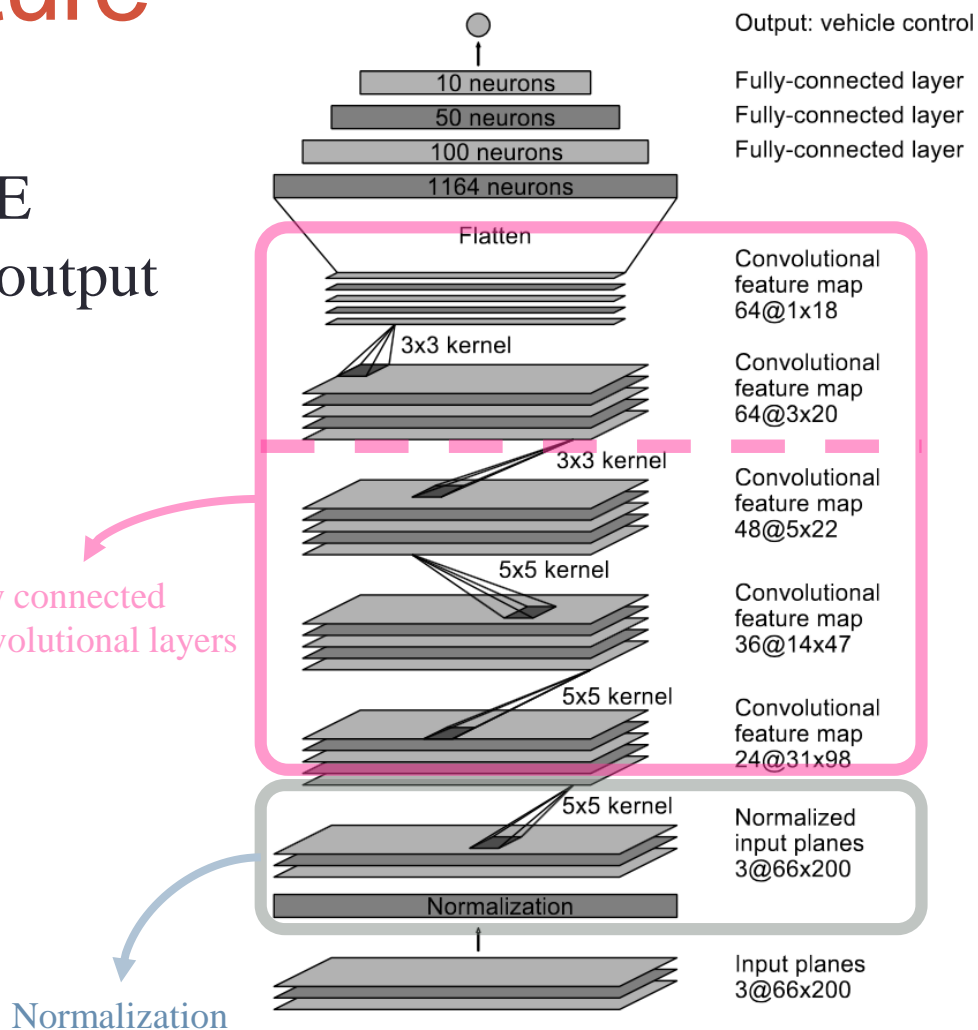
DAVE-1(2004) vs DAVE-2(2016)

- The small car → The real car
- Advance performance of training (GPGPU)
- Advanced CNN model
- Newly collected training data

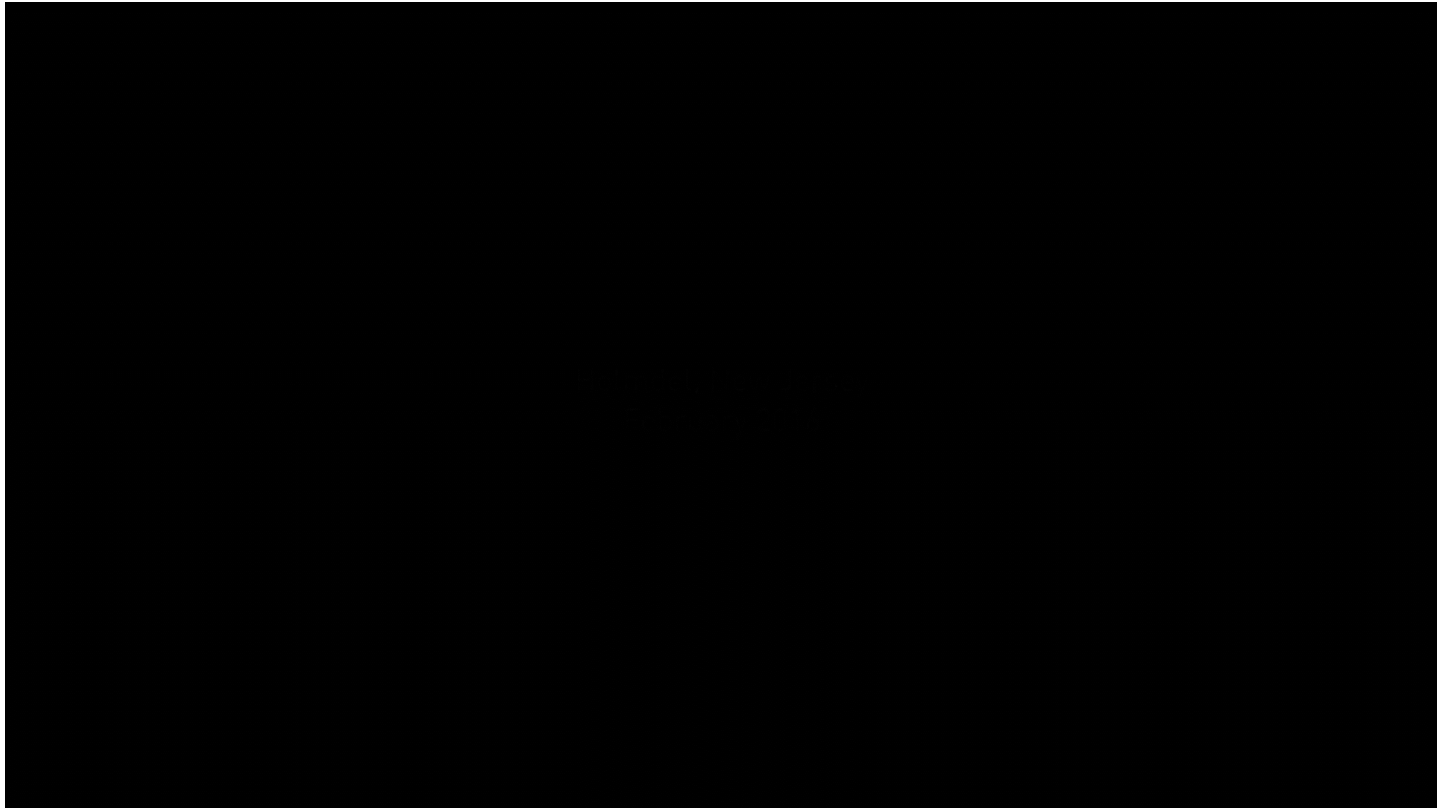
Network Architecture

➤ In order to minimize the MSE
(between steering command output
& human driver)

- YUV input
- The first layer:
hard-coded normalization
- Convolutional layers:
- [TensorFlow codes](#)



NVIDIA autonomous car driving video

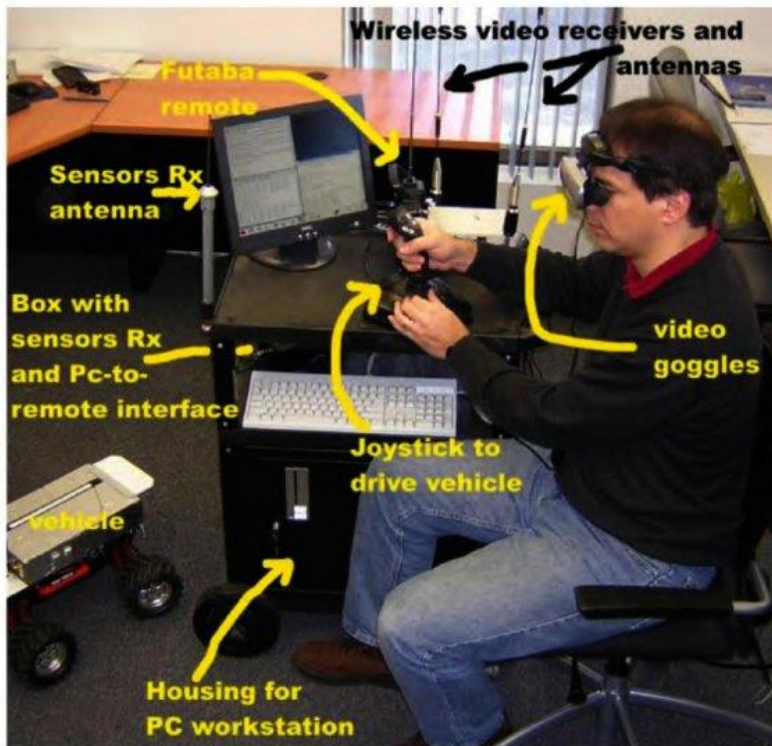


<https://youtu.be/qhUvQiKec2U>

- On road test:
For 10 miles driving, autonomous 98% of the time.

Trivia, Yann LeCun's post on fb

- Mr. LeCun participated in DAVE-1 project, 2004(below)



Yann LeCun

4월 26일 · 🌐

Our friends at NVIDIA in New Jersey have posted a paper on ArXiv about "DAVE-2", the ConvNet-based self-driving car system trained end to end that was demonstrated by NVIDIA CEO Jensen Huang at GTC 2016.

Interesting facts:

- The ConvNet maps a single image to a steering angle
- The ConvNet architecture is relatively small so as to run in real time on the Drive-PX embedded computer.
- The system is trained with Torch7
- The real-time system that drives the car is written in Torch7 and runs at 30 frames per second on the Drive-PX.
- The training dataset consists of about 72 hours of video with recorded steering angle provided by a human driver.
- The car has 3 cameras in left, center and right positions so as to simulate the view if the car were not in the center of the lane.

Paper: <http://arxiv.org/abs/1604.07316>

Video: <https://drive.google.com/.../0B9raQzOpizn1TkRla241ZnBEcjQ/view>

UPDATE: Slashdotted <https://hardware.slashdot.org/.../nvidia-gpu-powered-autonomo...>

	<p>[1604.07316] End to End Learning for Self-Driving Cars</p> <p>ARXIV.ORG</p>
--	--

좋아요 579개 댓글 19개 공유 160회

공유하기

DIRECT PERCEPTION APPROACH

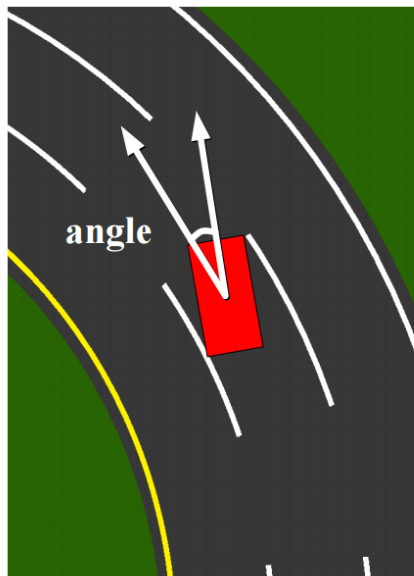
C. Chen et al. ICCV 2015

			Tasks						
			ADAS						
			Self Driving						
			Localizati on	Perception	Planning/ Control	Driver state	Vehicle Diagnosis	Smart factory	
Methods	Traditional	Non-machine Learning		GPS, SLAM		Optimal control			
		Machine-Learning based method	Supervised	SVM MLP		Pedestrian detection (HOG+SVM)			
	CNN				Affordance estimation	End-to- end Learning			
	RNN (LSTM)				Dry/wet road classificati on	End-to- end Learning	Behavior Prediction/ Driver identificati on	*	
	DNN							*	*
	Reinforcement				*				
	Unsupervised						*		

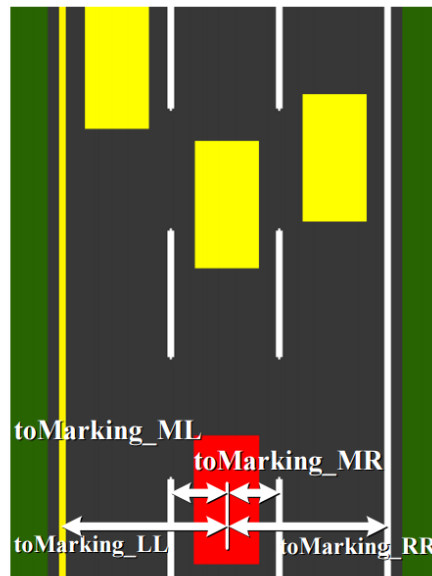
DeepDriving

- Direct perception
 - **Estimate the affordance for driving** instead of visually parsing the entire scene or blindly mapping an image to controls
 - Mapping an input image to **a small number of key perception indicators** that directly related to the affordance of a road/traffic state
- Approach
 - Built upon deep convolution neural network
 - Trained and tested on TORCS (The Open Racing Car Simulator)
 - Automatically learn image features for estimating affordance related to autonomous driving
 - **Much simpler structure than the typical mediated perception approach**
 - **More interpretable than the typical behavior reflex approach**

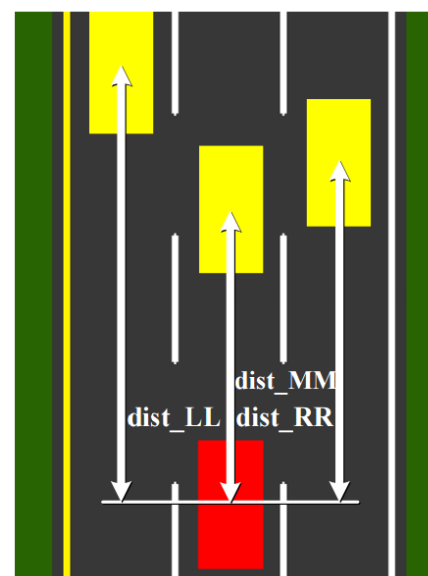
Affordance



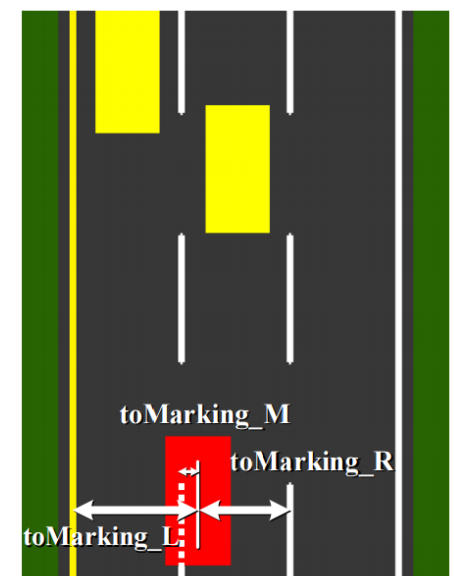
(a) angle



(b) in lane: toMarking



(c) in lane: dist



(d) on mark.: toMarking

Affordance indicators

always:

1) angle: angle between the car's heading and the tangent of the road

“in lane system”, when driving in the lane:

2) toMarking_LL: distance to the left lane marking of the left lane

3) toMarking_ML: distance to the left lane marking of the current lane

4) toMarking_MR: distance to the right lane marking of the current lane

5) toMarking_RR: distance to the right lane marking of the right lane

6) dist_LL: distance to the preceding car in the left lane

7) dist_MM: distance to the preceding car in the current lane

8) dist_RR: distance to the preceding car in the right lane

“on marking system”, when driving on the lane marking:

9) toMarking_L: distance to the left lane marking

10) toMarking_M: distance to the central lane marking

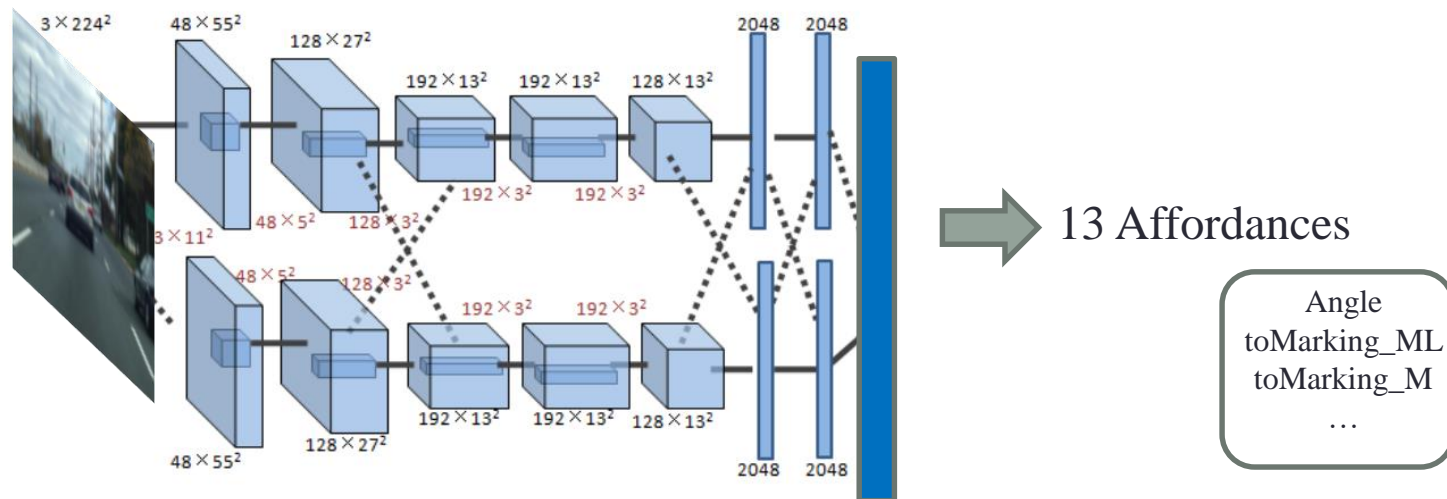
11) toMarking_R: distance to the right lane marking

12) dist_L: distance to the preceding car in the left lane

13) dist_R: distance to the preceding car in the right lane

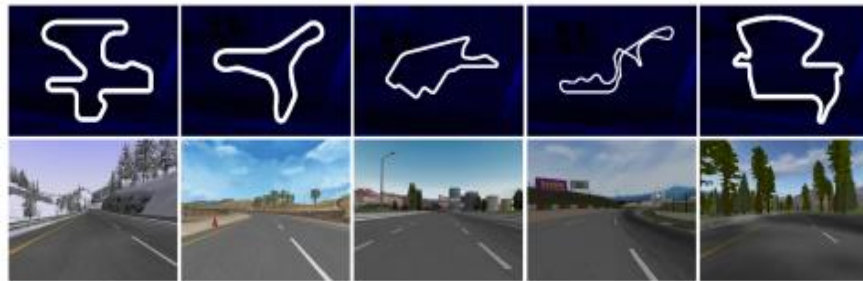
Affordance estimation - CNN Model Learning

- AlexNet
- Supervised Learning (484,815 Training images)



Training Dataset

- TORCS(The Open Racing Car Simulator)
- Collect indicators (human control data)
 - Speed of host car
 - Position of host car
 - Distance to the preceding car



Tracks (7 Tracks)



Cars (22 kinds)

Visualization of Learned models

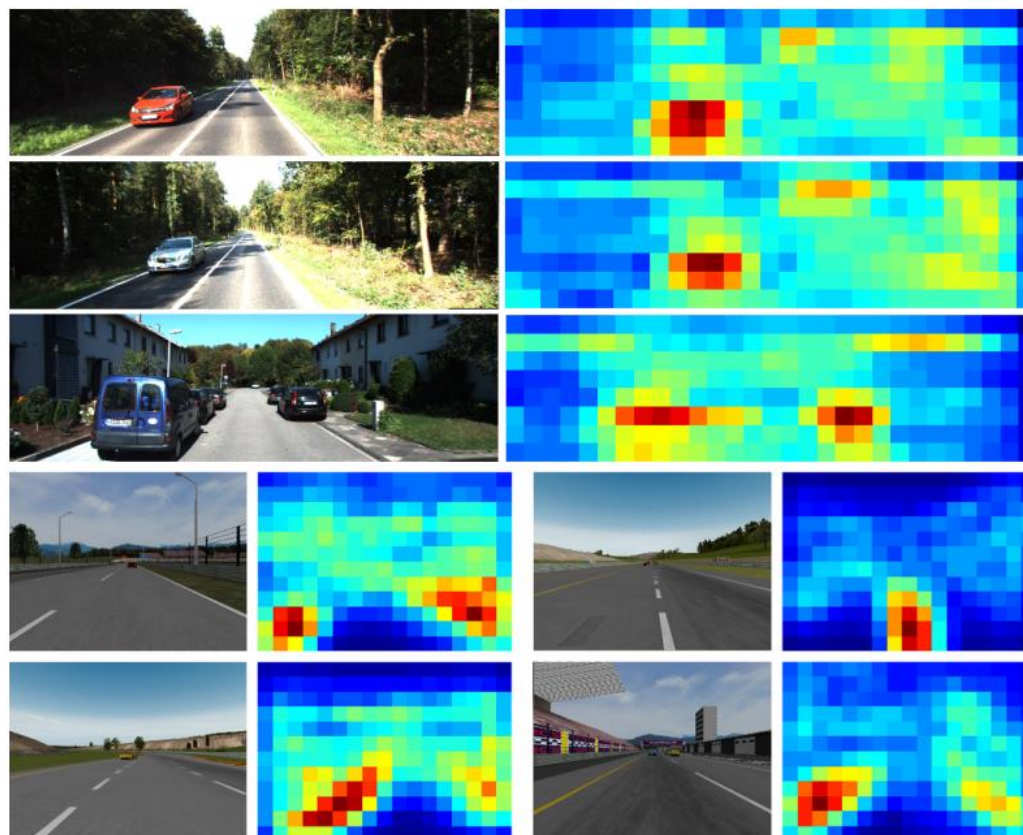


Figure 13: **Response map** of our KITTI-based (Row 1-3) and TORCS-based (Row 4-5) ConvNets. The ConvNets have strong responses over nearby cars and lane markings.

Result



Learning Affordance for Direct Perception in Autonomous Driving

Chenyi Chen Ari Seff Alain Kornhauser Jianxiong Xiao

Princeton University



BAKCUPS

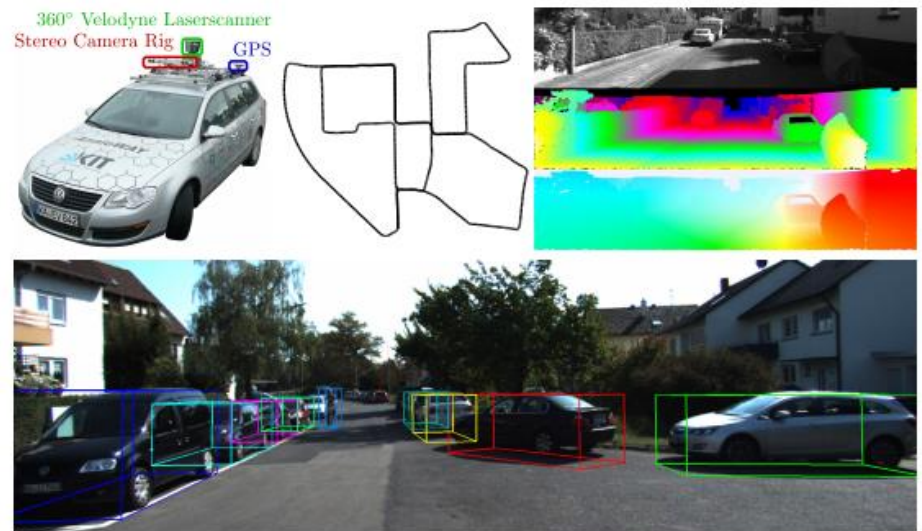
KITTI DATASET

Are we ready for Autonomous Driving?

The KITTI Vision Benchmark Suite (CVPR, 2012)

1. Introduction

Developing autonomous systems that are able to assist humans in everyday tasks is one of the grand challenges in modern computer science. One example are autonomous driving systems which can help decrease fatalities caused by traffic accidents. While a variety of novel sensors have been used in the past few years for tasks such as recognition, navigation and manipulation of objects, visual sensors are rarely exploited in robotics applications: Autonomous driving systems rely mostly on GPS, laser range finders, radar as well as very accurate maps of the environment.



An Empirical Evaluation of Deep Learning on Highway Driving

must keep their hands on the steering wheel and prepare to control the vehicle in the event of any unexpected obstacle or catastrophic incident. Financial considerations contribute to a substantial performance gap between commercially available auto-pilot systems and fully self-driving cars developed by Google and others. Namely, today's self-driving cars are equipped with expensive but critical sensors, such as LIDAR, radar and high-precision GPS coupled with highly detailed maps.

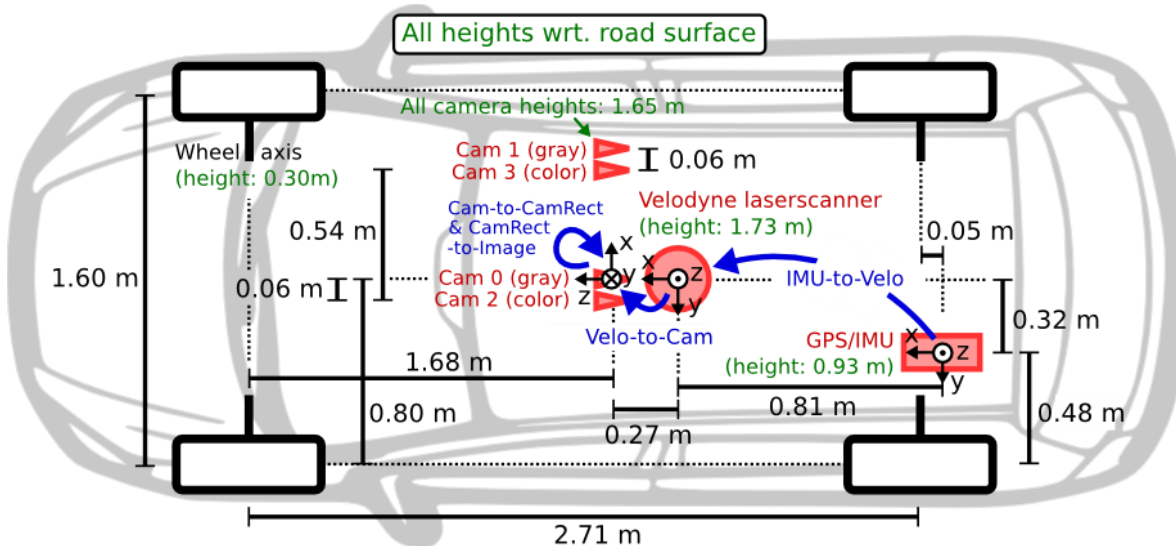
In today's production-grade autonomous vehicles, critical sensors include radar, sonar, and cameras. Long-range vehicle

richer set of features at a fraction of the cost. By advancing computer vision, cameras could serve as a reliable redundant sensor for autonomous driving. Despite its potential, computer vision has yet to assume a significant role in today's self-driving cars. Classic computer vision techniques simply have not provided the robustness required for production grade automotives; these techniques require intensive hand engineering, road modeling, and special case handling. Considering the seemingly infinite number of specific driving situations, environments, and unexpected obstacles, the task of scaling classic computer vision to robust, human-level performance would prove monumental and is likely to be unrealistic.



KITTI Benchmark suite – Sensor Setup

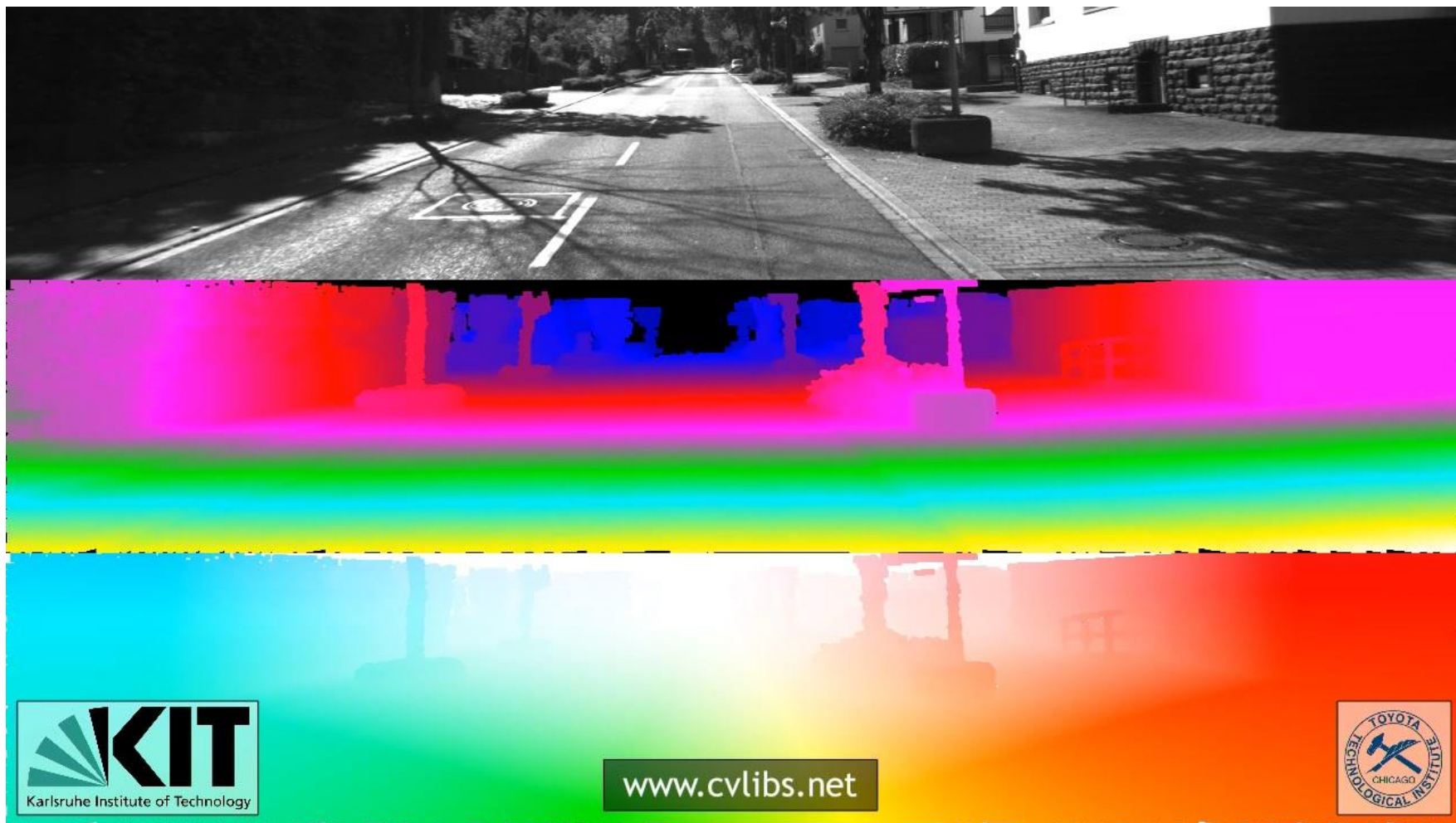
- 다양한 센서가 탑재된 자동차로 실제 주행을 통해 데이터 수집
 - 1 Navigation System(GPS/IMU) : OXTS RT 3003
 - 1 Laser scanner : Velodyne HDL-64E
 - 2 Grayscale cameras : Point Grey Flea 2 (FL2-14S3M-C)
 - 2 Color cameras : Point Grey Flea 2 (FL2-14S3C-C)
 - 4 Varifocal lenses : Edmund Optics NT59-917



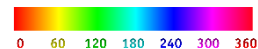
실제 데이터 수집을 위한 차량 및 센서 설계도

실제 차량 사진

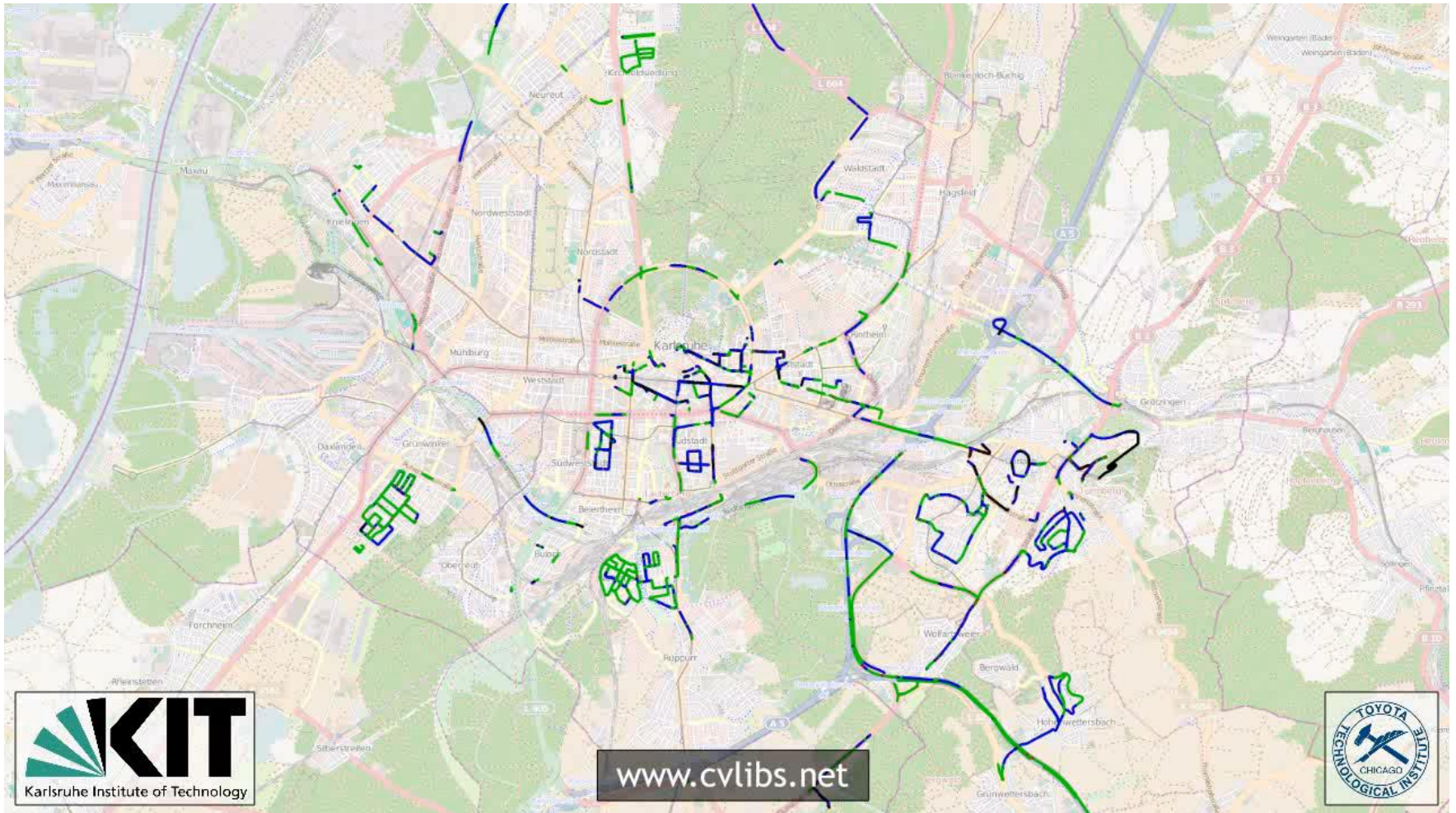
KITTI Benchmark suite – Ground truth(1/3)



Stereo & Optical Flow

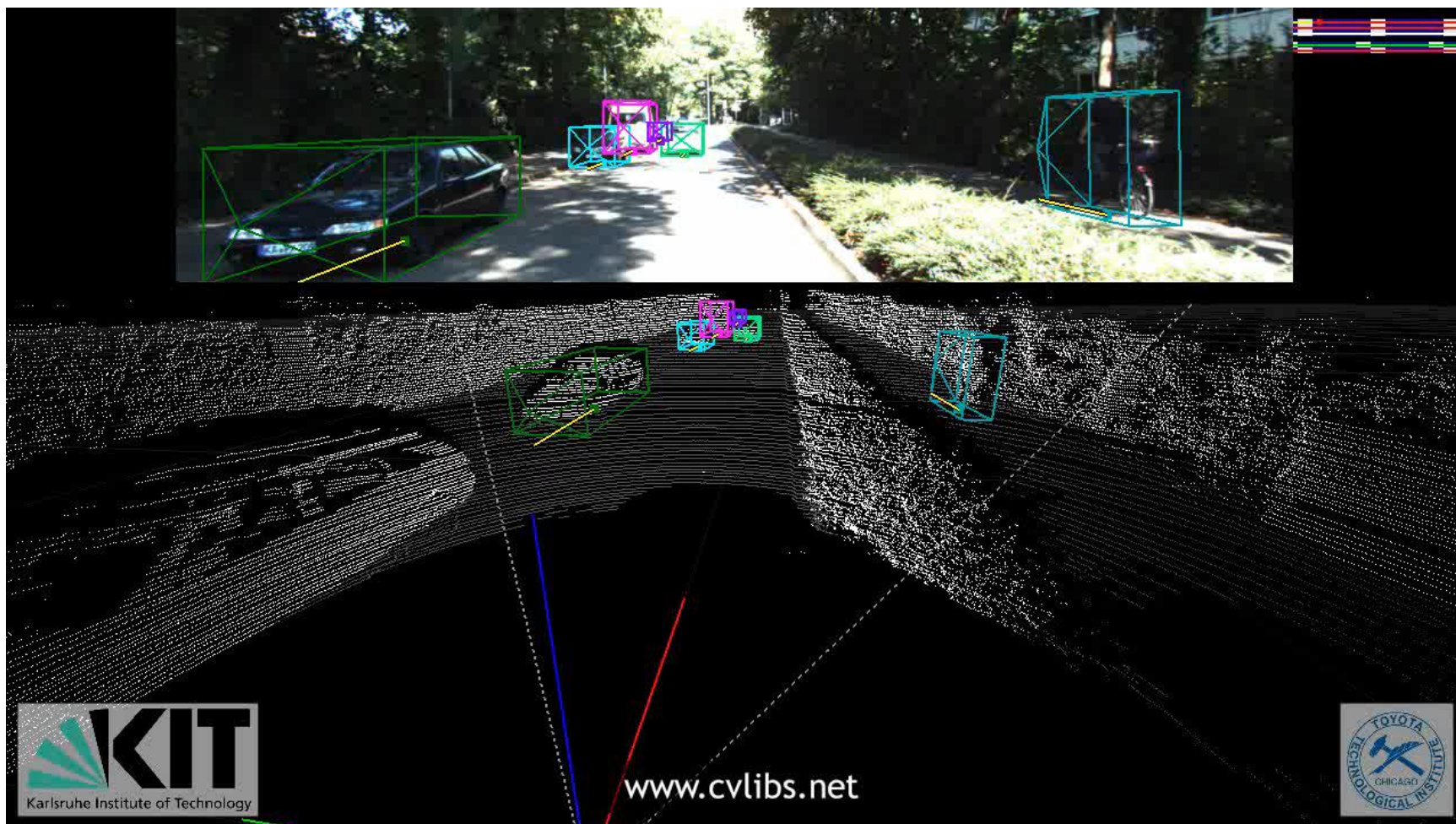


KITTI Benchmark suite – Ground truth(2/3)



Odometry

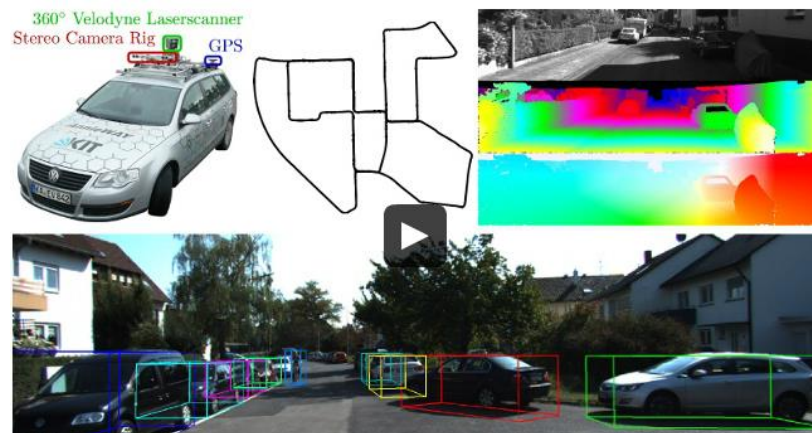
KITTI Benchmark suite – Ground truth(3/3)



KITTI Benchmark suite

- 다양한 비전 기술에 대한 Ground truth가 존재

- Stereo [1],[4]
- Optical Flow [1],[4]
- Odometry [1]
- Object detection [1]
- Tracking [1]
- Road/Lane detection [1]
- Raw data 제공[3]



KITTI Benchmark suite

- Reference

- [1] Geiger, Andreas, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012.
- [2] Geiger, Andreas, et al. "Vision meets robotics: The KITTI dataset." *The International Journal of Robotics Research* (2013): 0278364913491297.
- [3] Fritsch, Jannik, Tobias Kuehnl, and Andreas Geiger. "A new performance measure and evaluation benchmark for road detection algorithms." *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013).* IEEE, 2013.
- [4] Menze, Moritz, and Andreas Geiger. "Object scene flow for autonomous vehicles." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015.