Deep Learning for Computer Vision

TDAI Deep Learning Summer School, 2022

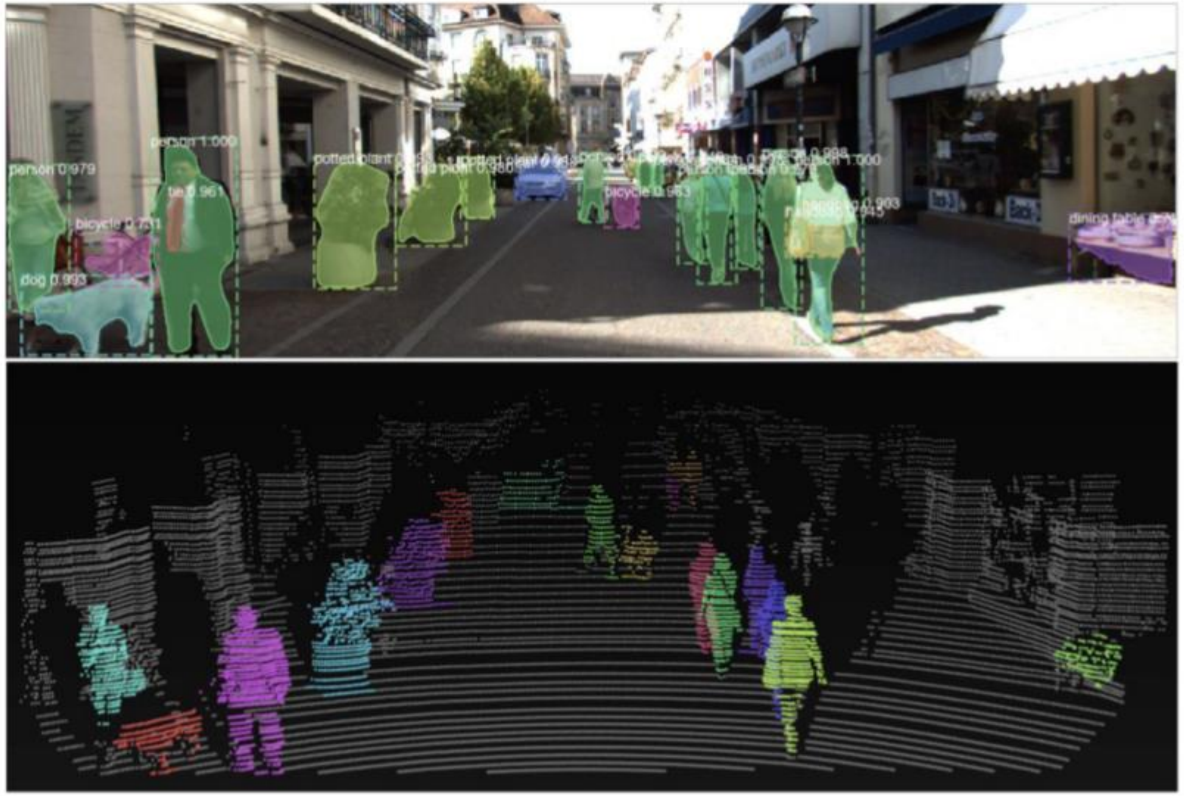
June 2nd, 2022

Wei-Lun (Harry) Chao

The Ohio State University

Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



Recap:
Machine learning and deep learning

Machine learning recap

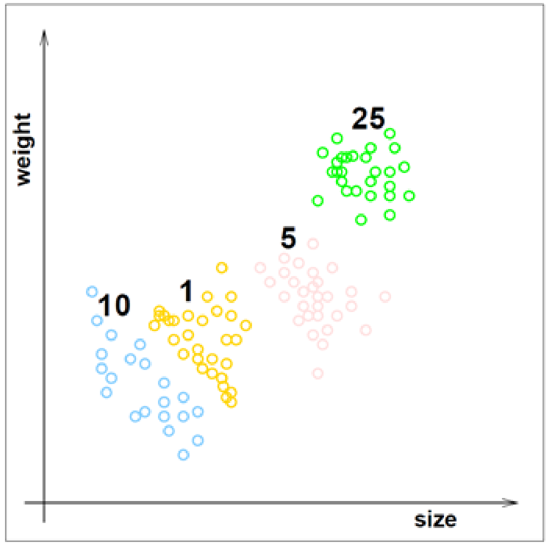
Learning from Data

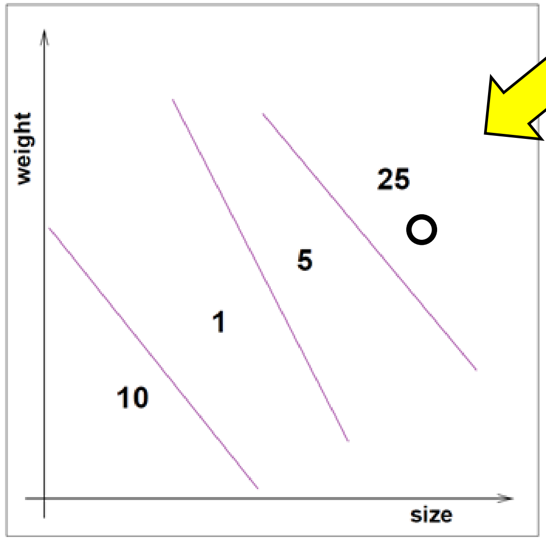
Algorithm, model **Evaluation, loss** **Training data**

Example: coin classifier

Training data → Machine learning algorithms → **Learned models & patterns**

Test data

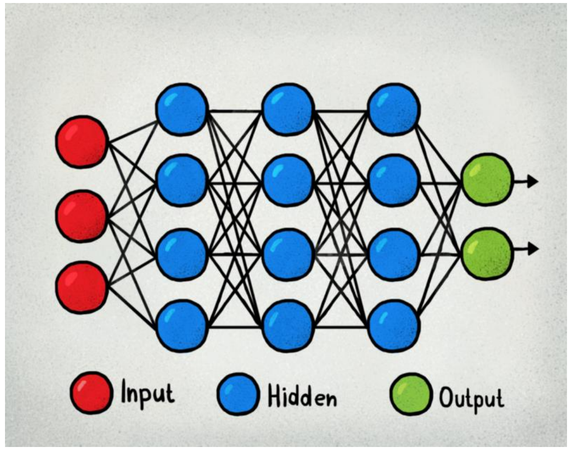




[Figure credit: Y. Abu-Mostafa, M. Magdon-Ismail, H-T Lin. Learning from data.]

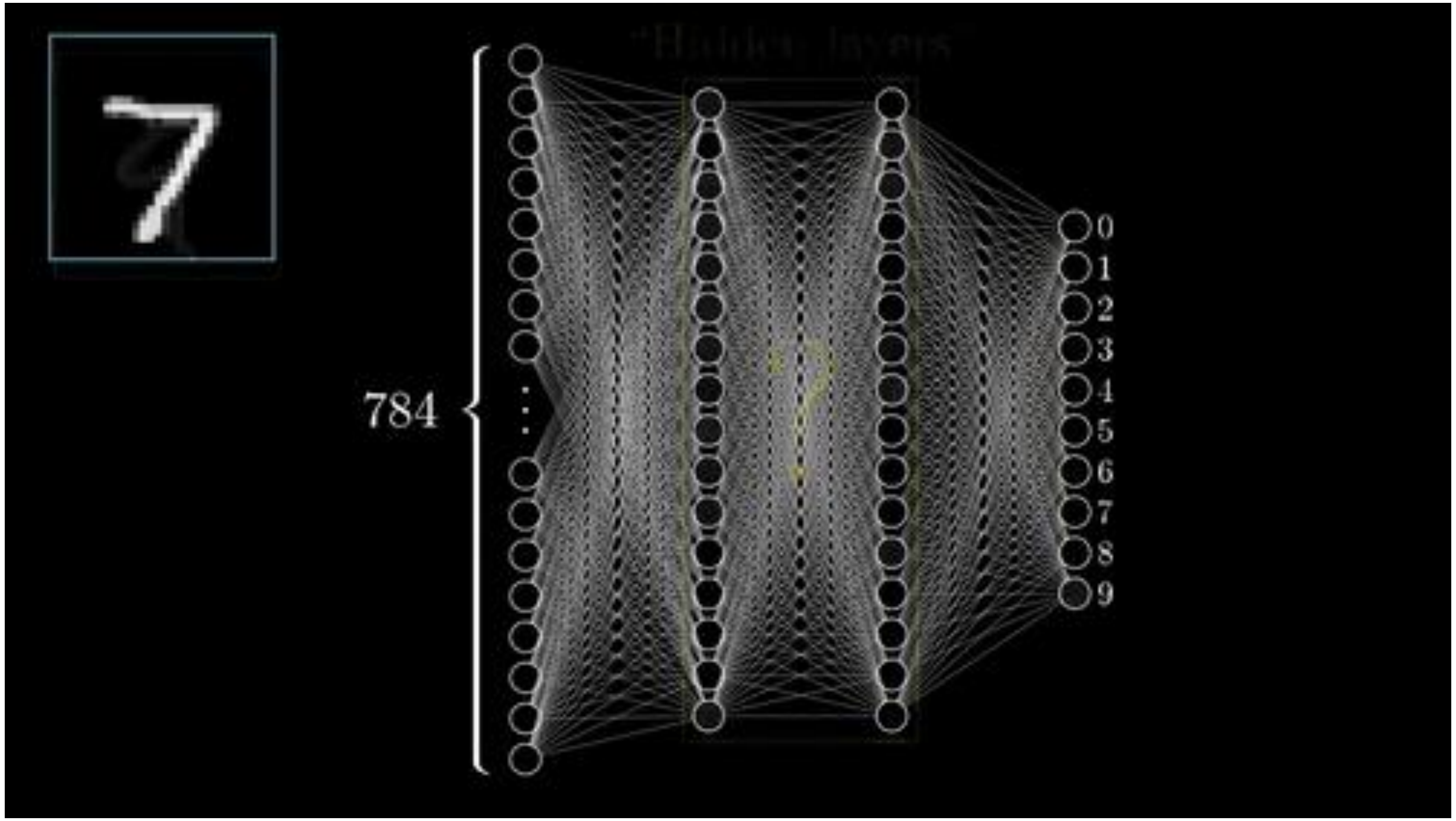
Deep learning recap

→ Classifier → Label: 1, 5, 10, 25, ...



A sequence of "learnable" computation!

Example: image classification

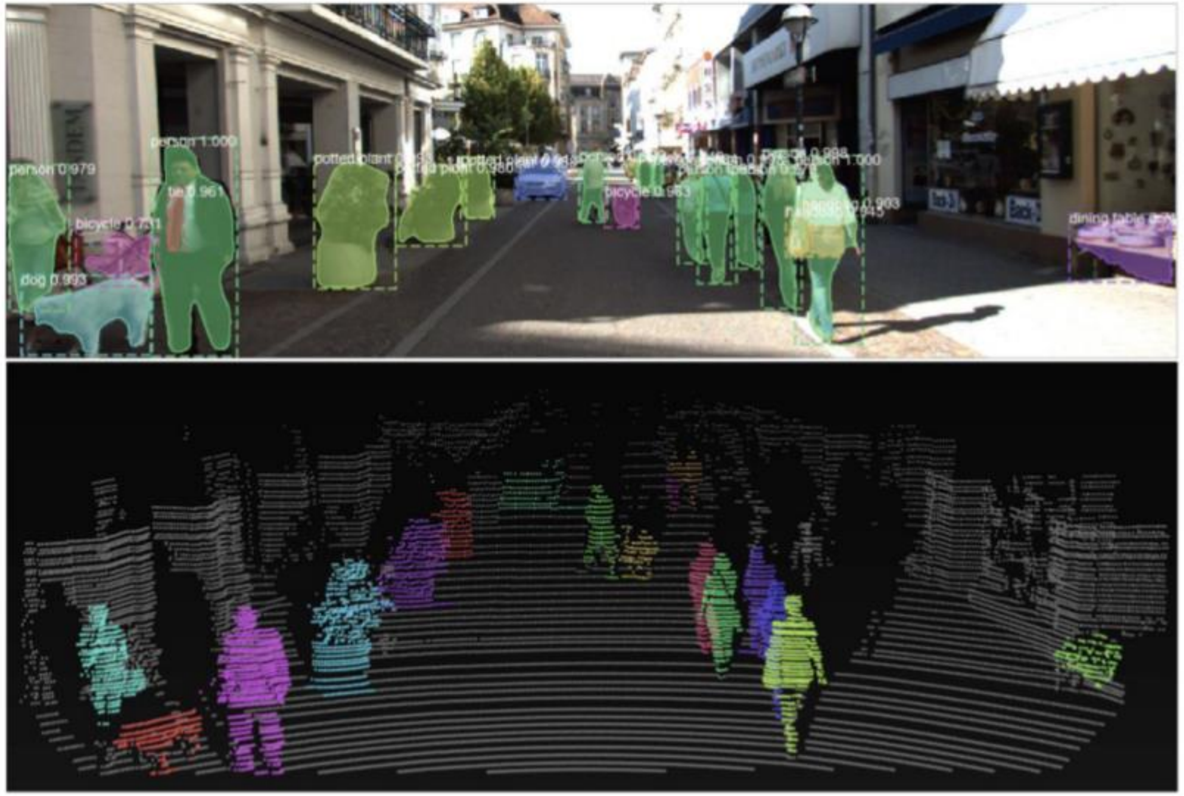


A sequence of "learnable" computation!

[Gif credits: Gradient descent 3Blue1Brown series S3 E2]

Outline

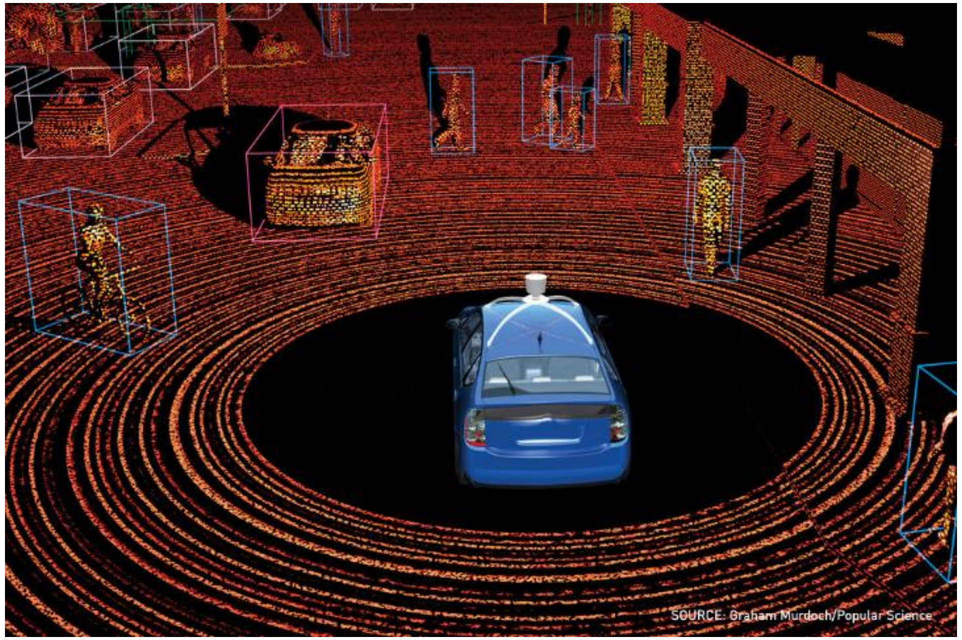
- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
    - Convolutional neural nets
    - Visual transformers
- Applications:
    - 2D Recognition
    - 3D Perception for autonomous driving
    - 2D Generation
- Practical problems:
    - Insufficient (labeled) data
    - Domain shifts



Computer vision

- When the data are about images, videos, or signals captured by 3D sensors …

[Source: Detectron2]

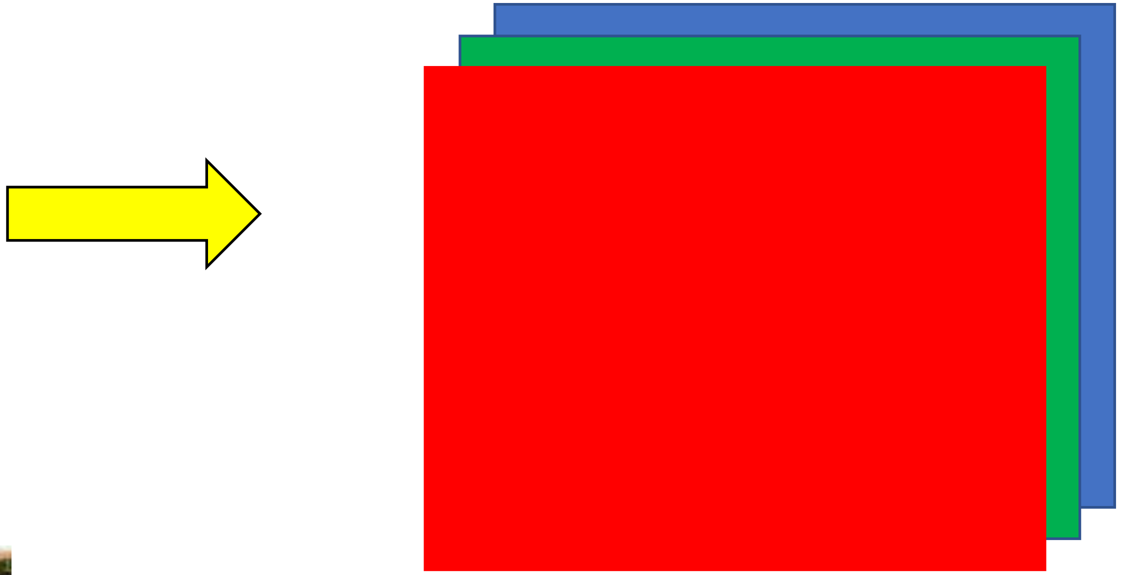


[Source: Graham Murdoch/Popular Science]

Computer vision: data

Image (s)

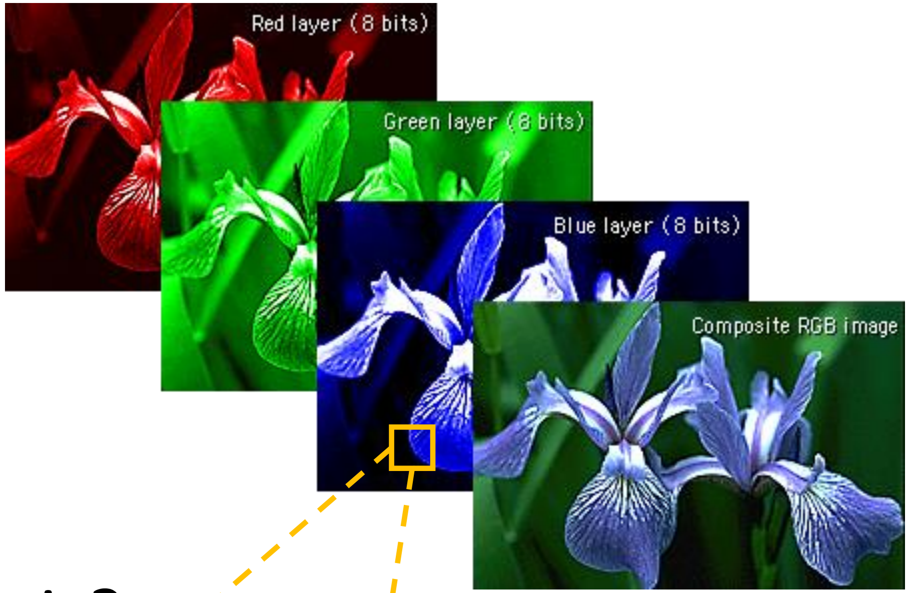
Video (s) = sequence of images



RGB image (s): Three matrices

Computer vision: data

- RGB images:



- What is inside each matrix?
  - {0,1,……,255}
  - Interval: [0, 1]

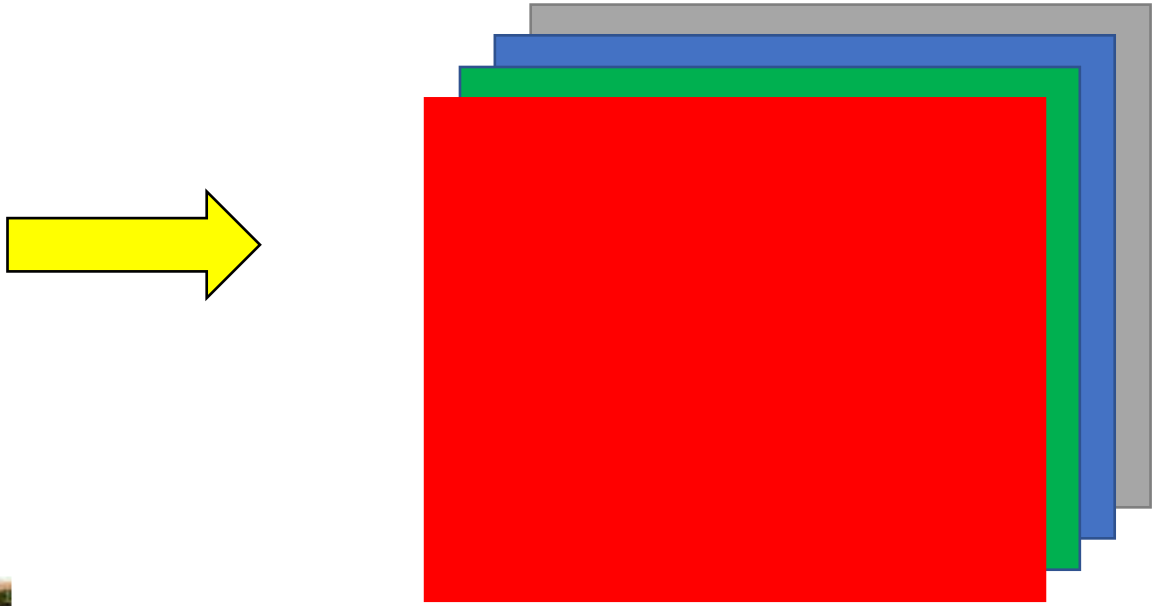
0	0	124	255	125
0	0	125	126	60
0	0	126	60	126
0	0	0	127	60
0	0	0	0	128

0	0	124	255	125
0	0	125	126	60
0	0	126	60	126
0	0	0	127	60
0	0	0	0	128

0	0	124	255	125
0	0	125	126	60
0	0	126	60	126
0	0	0	127	60
0	0	0	0	128

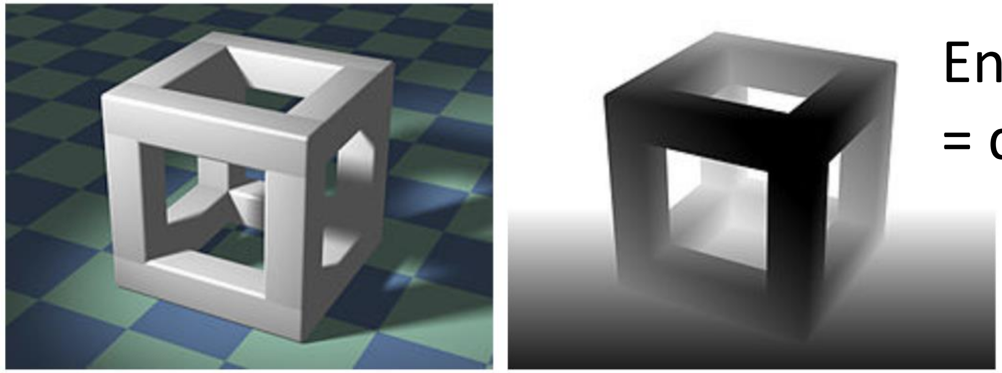
Computer vision: data

Image (s)



Video (s) = sequence of images

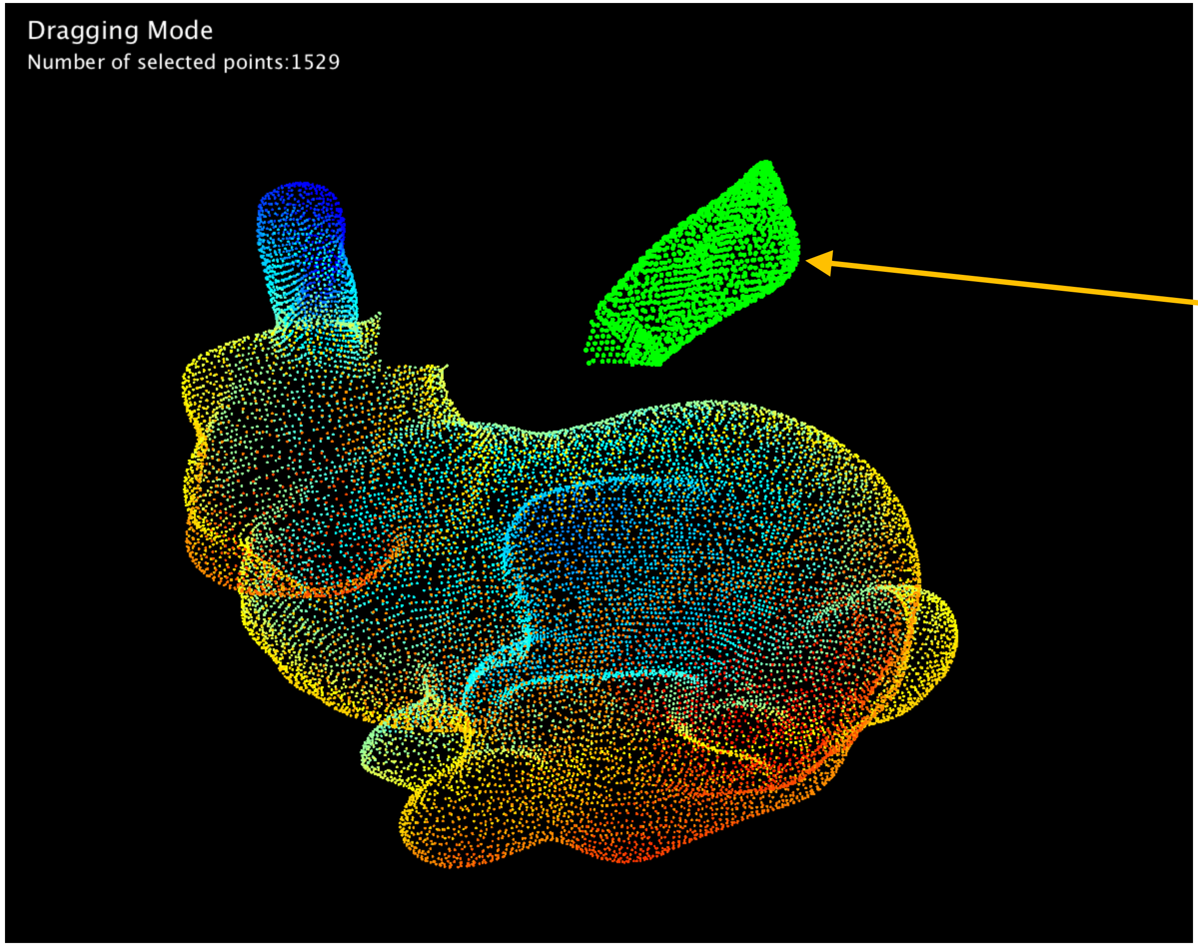
RGBD image (s): Four matrices



Entry value = depth

Computer vision: data

Point cloud



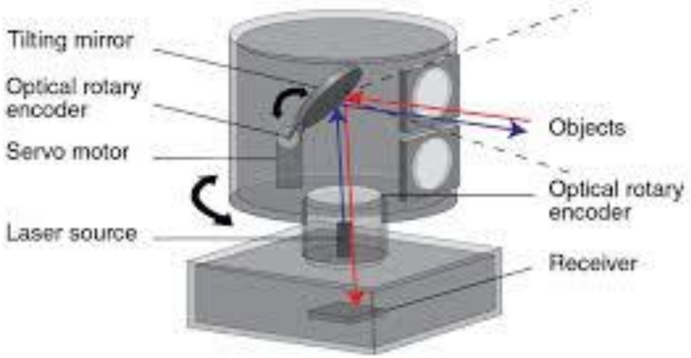
A collection of 3D (or 4D) points

x coordinate
y coordinate
z coordinate
reflectance

N points = 3-by-N or 4-by-N matrix
(should not be processed by convolutional neural nets directly!)

Computer vision: data

Image aligned with point cloud

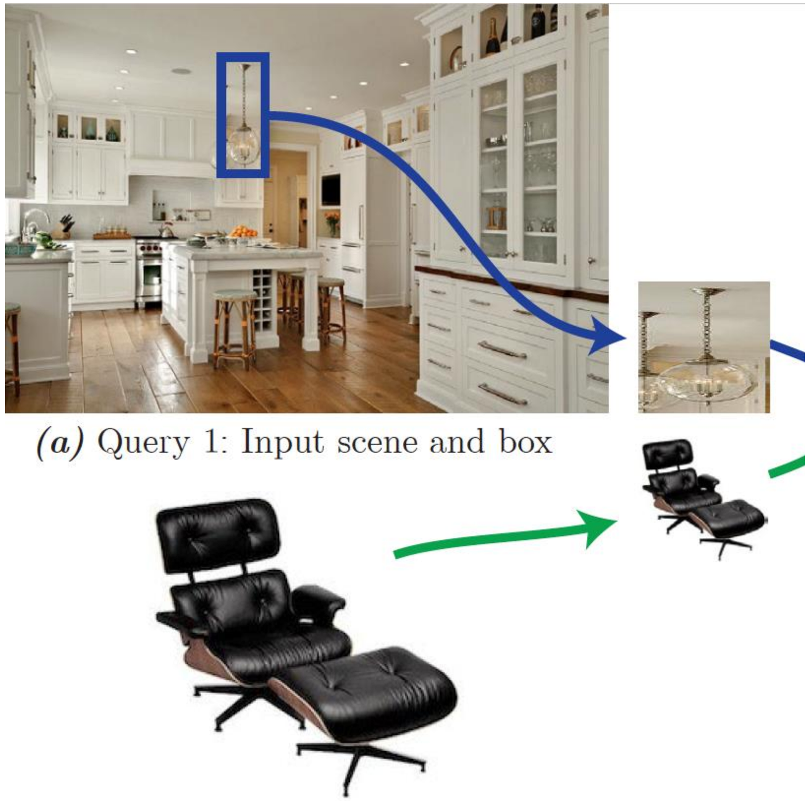


Questions?

Computer vision: representative tasks

Image Recognition

Computer vision: representative tasks

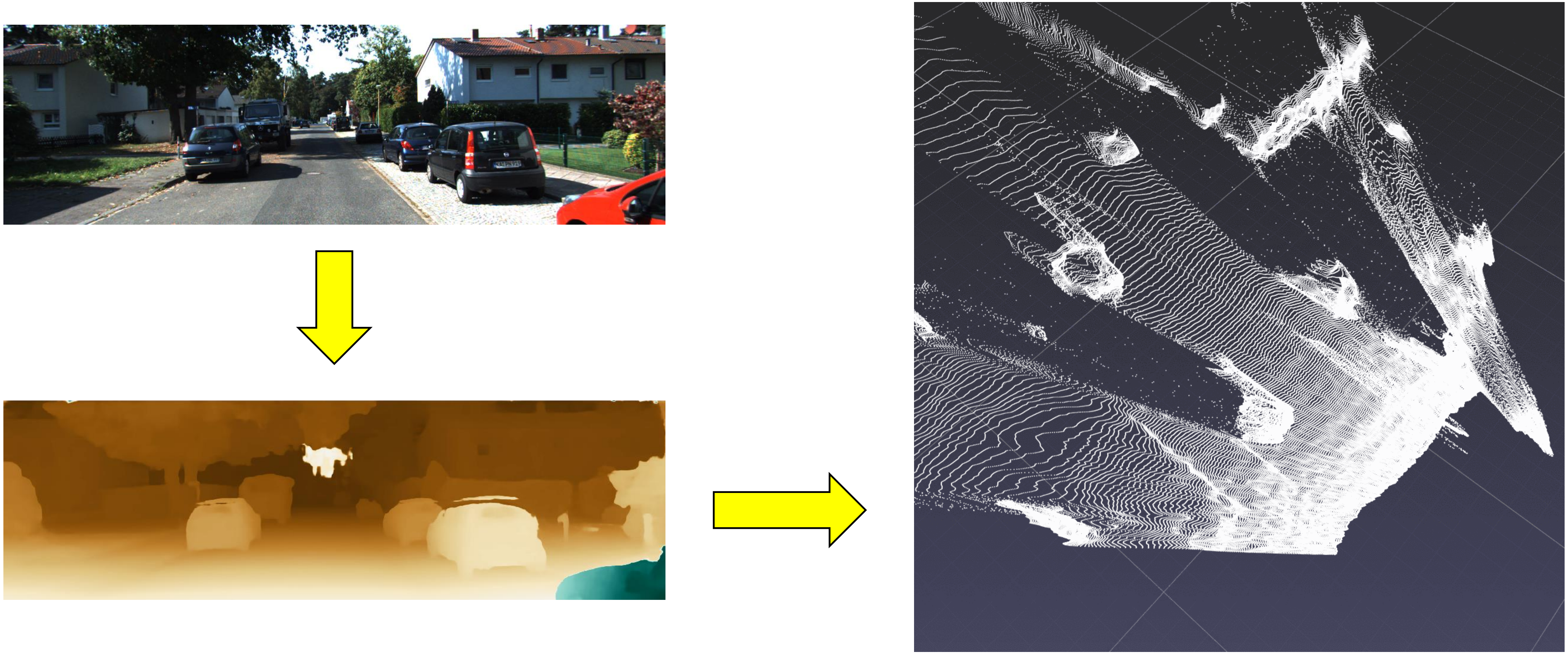


(a) Query 1: Input scene and box

(a) Query 2: Product

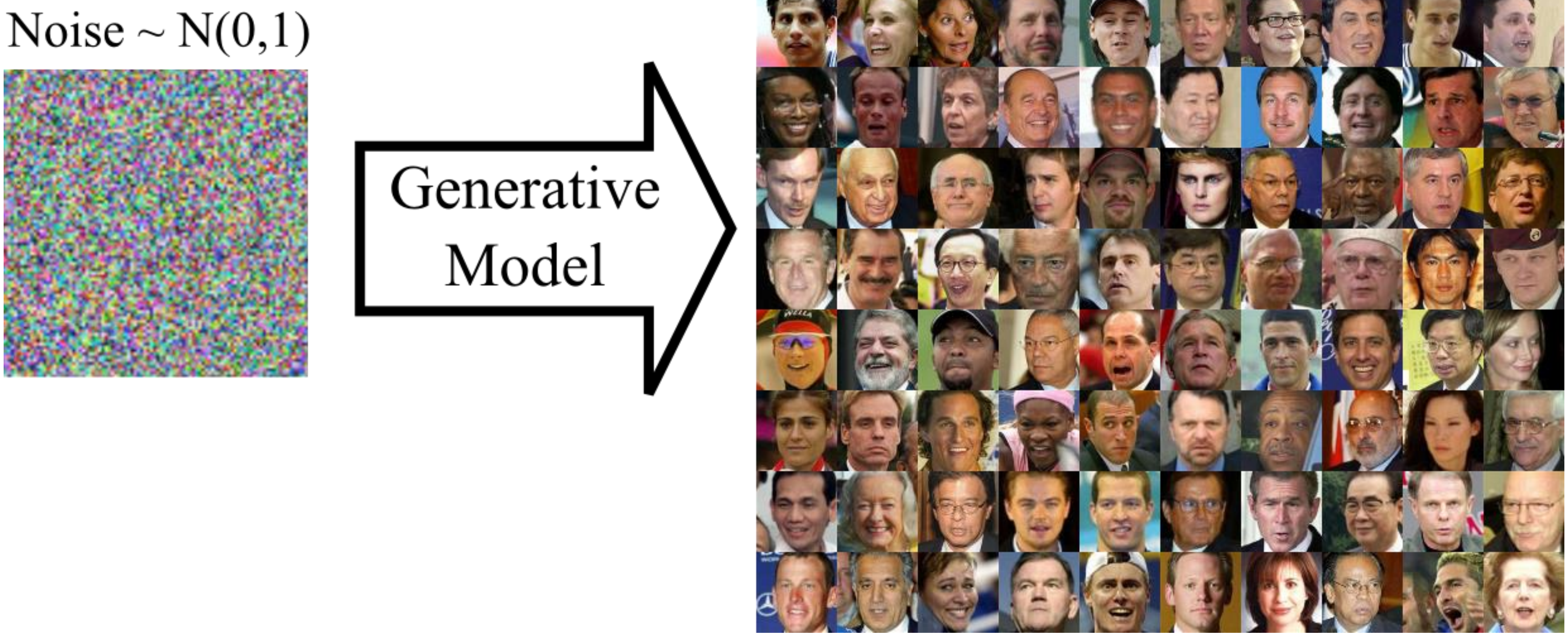
Retrieval, image-to-image search

Computer vision: representative tasks



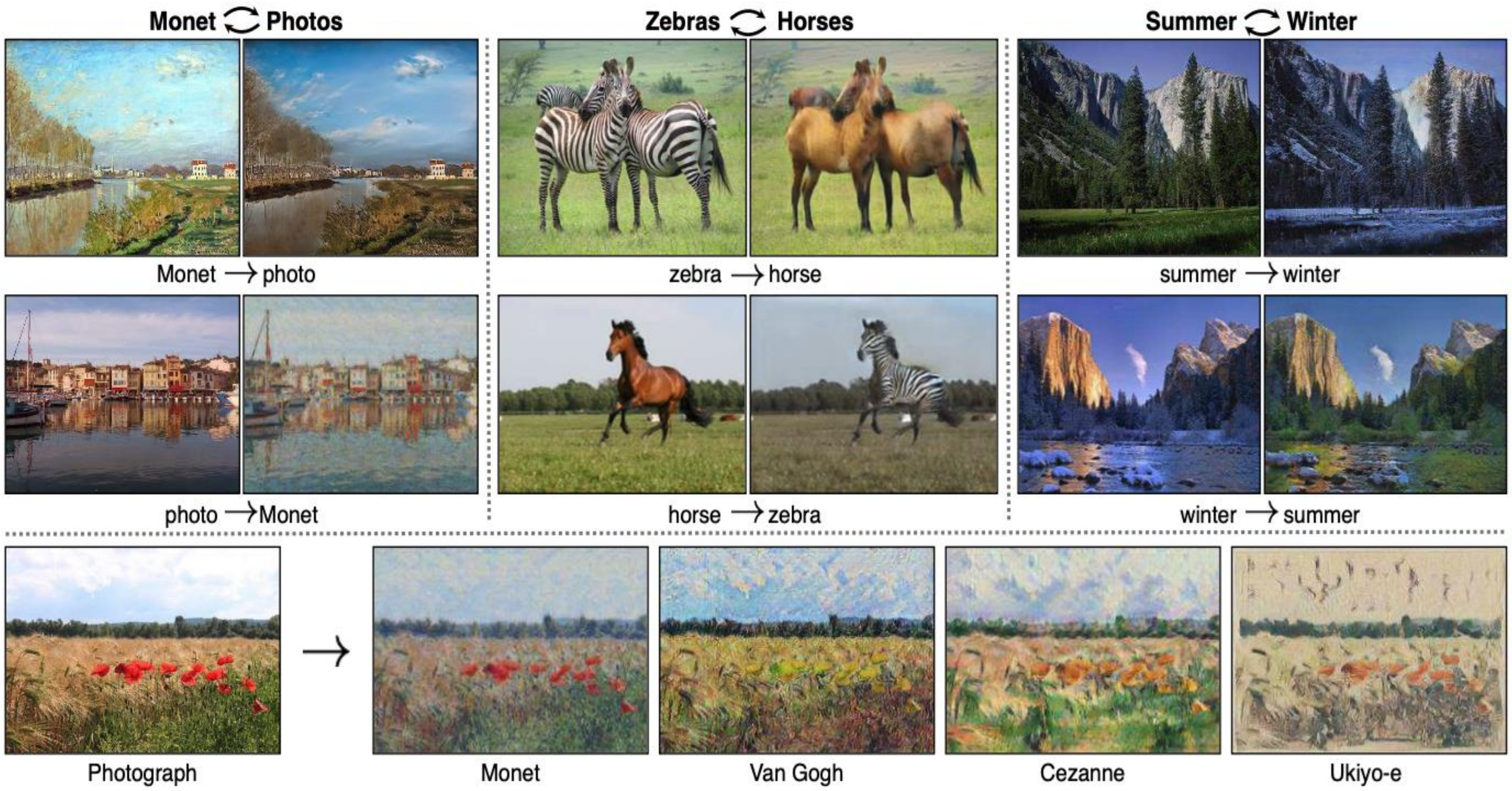
Depth estimation and 3D reconstruction

Computer vision: representative tasks



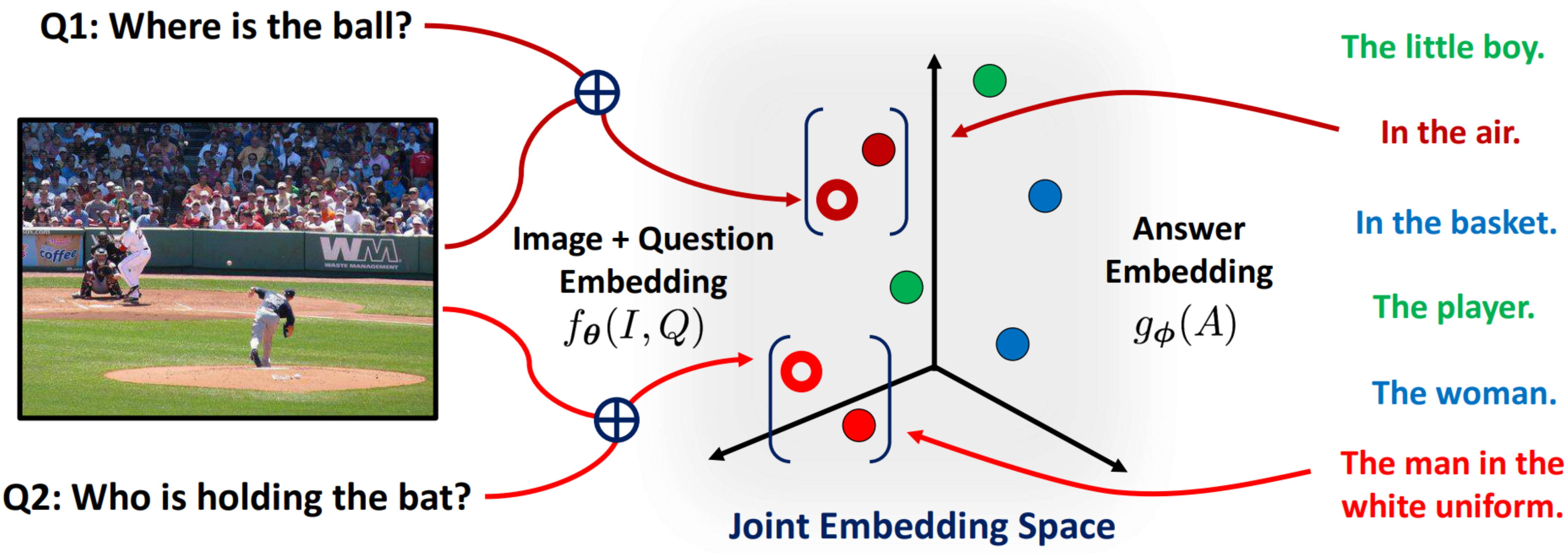
Computer vision: representative tasks

Style transfer



[Figure credit: CycleGAN, ICCV 2017]

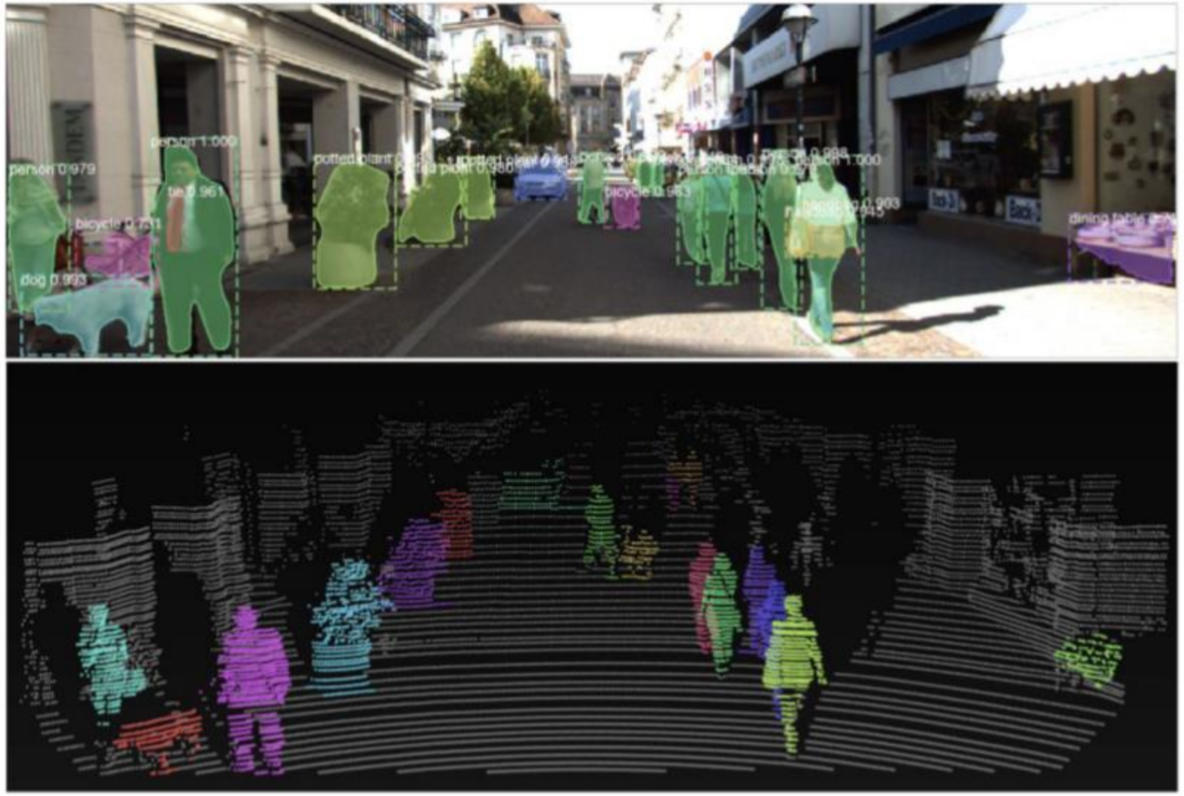
Computer vision: representative tasks



Vision & language; e.g., visual question answering

Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



Object-centric vs. scene-centric images

Object-centric images:

- contain a **single** class of objects
- The object size is usually large
- The background is simple

Scene-centric images:

- contain **multiple** classes of objects
- The object sizes can vary
- The background is challenging
- Objects may be occluded

Classification on object-centric images

car

elephant

- Single object class (not multi-label cases)

- **Properties to capture:**

➤ Translation invariant

➤ Scale invariant

The progress of deep learning for classification

ImageNet-1K (ILSVRC)

- 1,000 object classes
- 1,000 training images/class
- Each image contains just one class of object!

Metric: Top-k accuracy

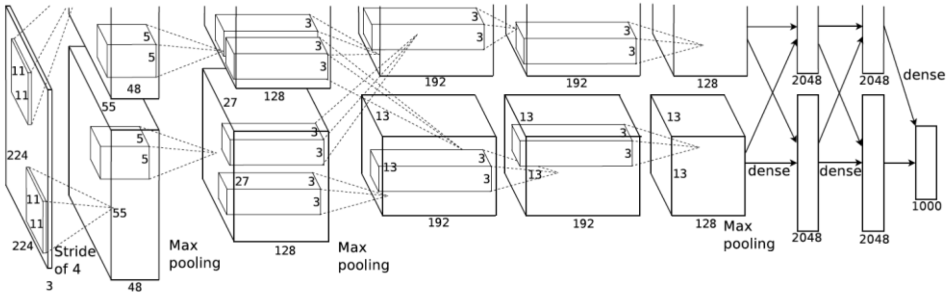
- For each image, return a list of top-k possible classes
- If the true class is within the list, the classification is correct

The progress of deep learning for classification

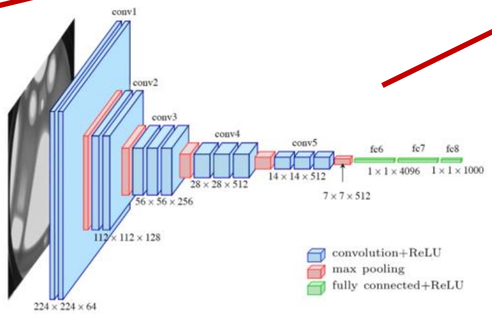
Top-5 error rate

0.05

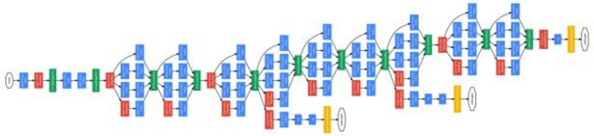
human



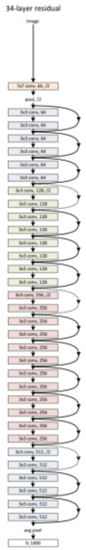
[Krizhevsky et al., 2012]



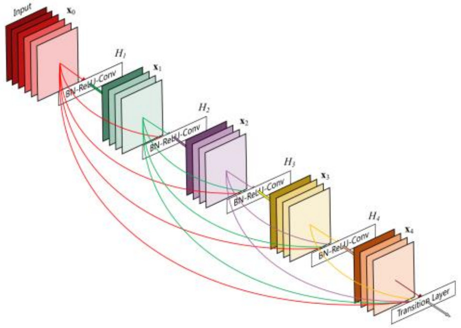
[Simonyan et al., 2015]



[Szegedy et al., 2015]



[He et al., 2016]

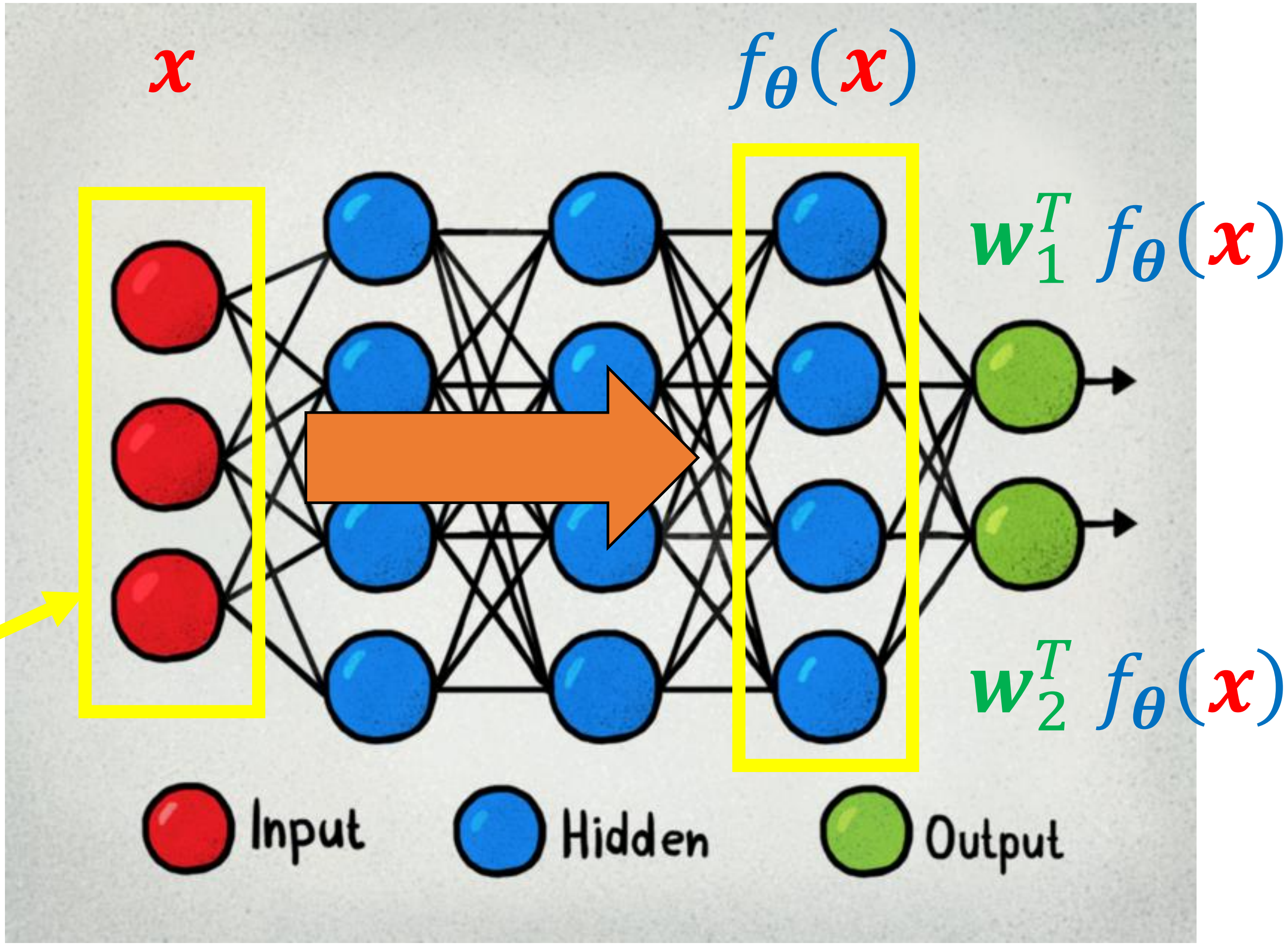


[Huang et al., 2017]

General formulation for all these variants

$$\text{Prediction} = \underset{c}{\operatorname{argmax}}\, \boldsymbol{w}_c^T f_{\boldsymbol{\theta}}(\boldsymbol{x})$$

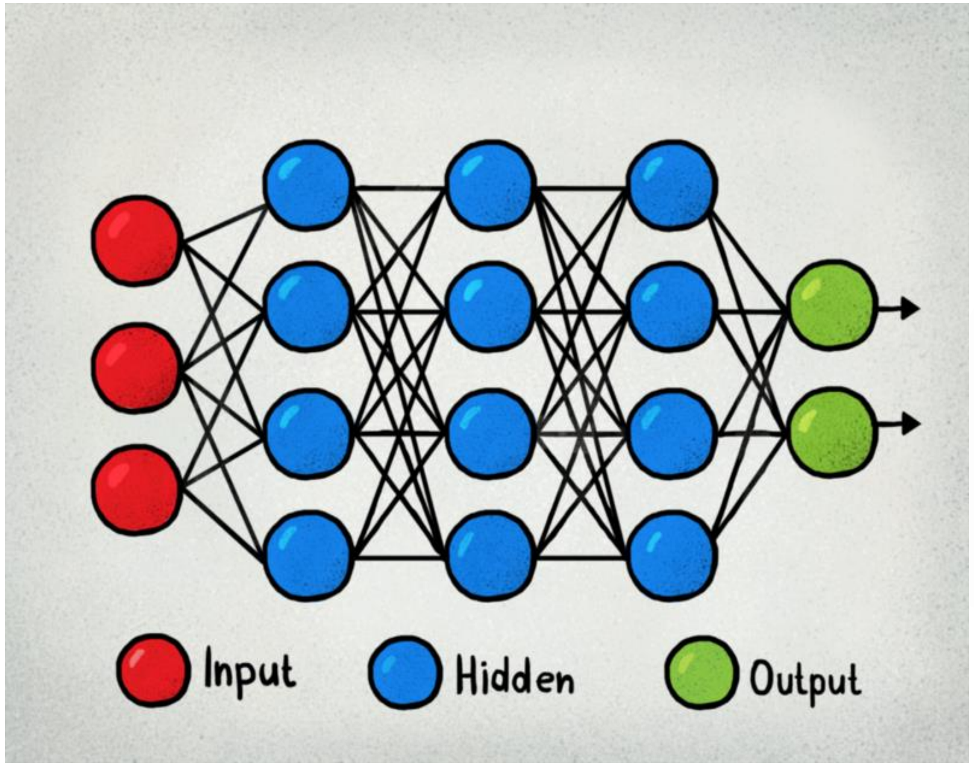
Image (pixels)

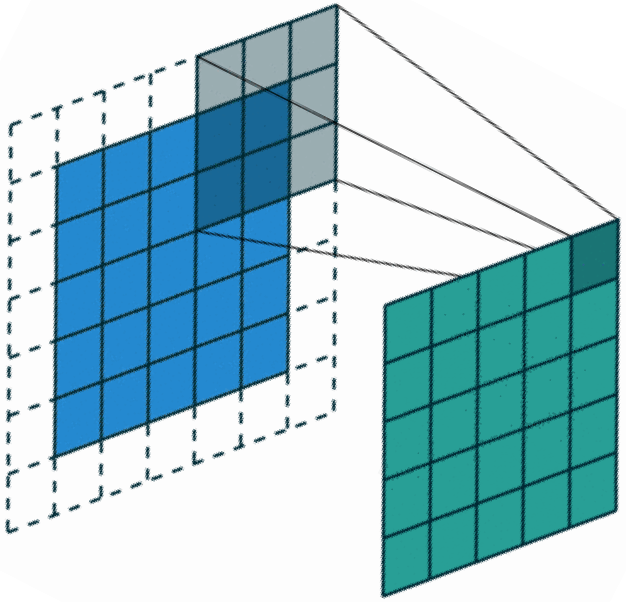


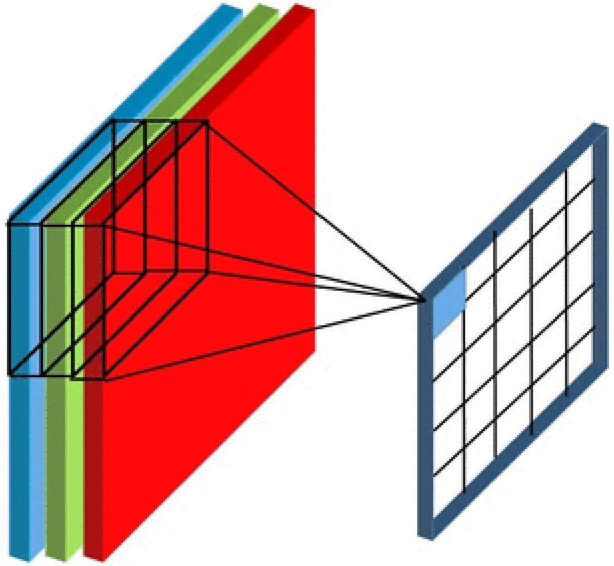
Convolution

A special computation between layers

- A current node is not directly affected by “all nodes in the previous layer”
- The network “weights” on the edges can be “re-used”







Convolution

Inner product
Element-wise multiplication and sum

0	0	0	0	1
0	0	0	1	1
0	0	1	1	1
0	1	1	1	1
1	1	1	1	1

Feature map (nodes) at layer *t*

0	0	1
0	1	1
1	1	1

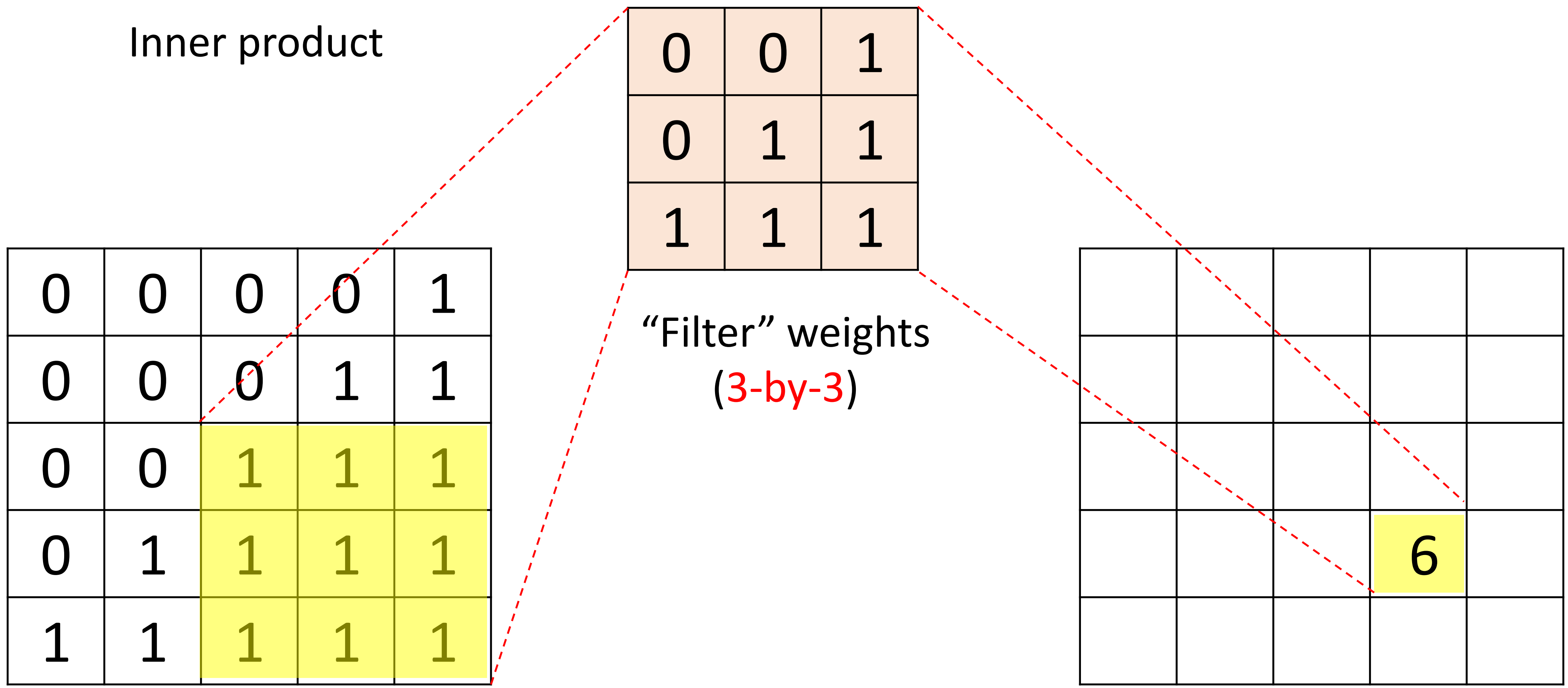
“Filter” weights
(3-by-3)

	1			

Feature map at layer *t*+1

Convolution

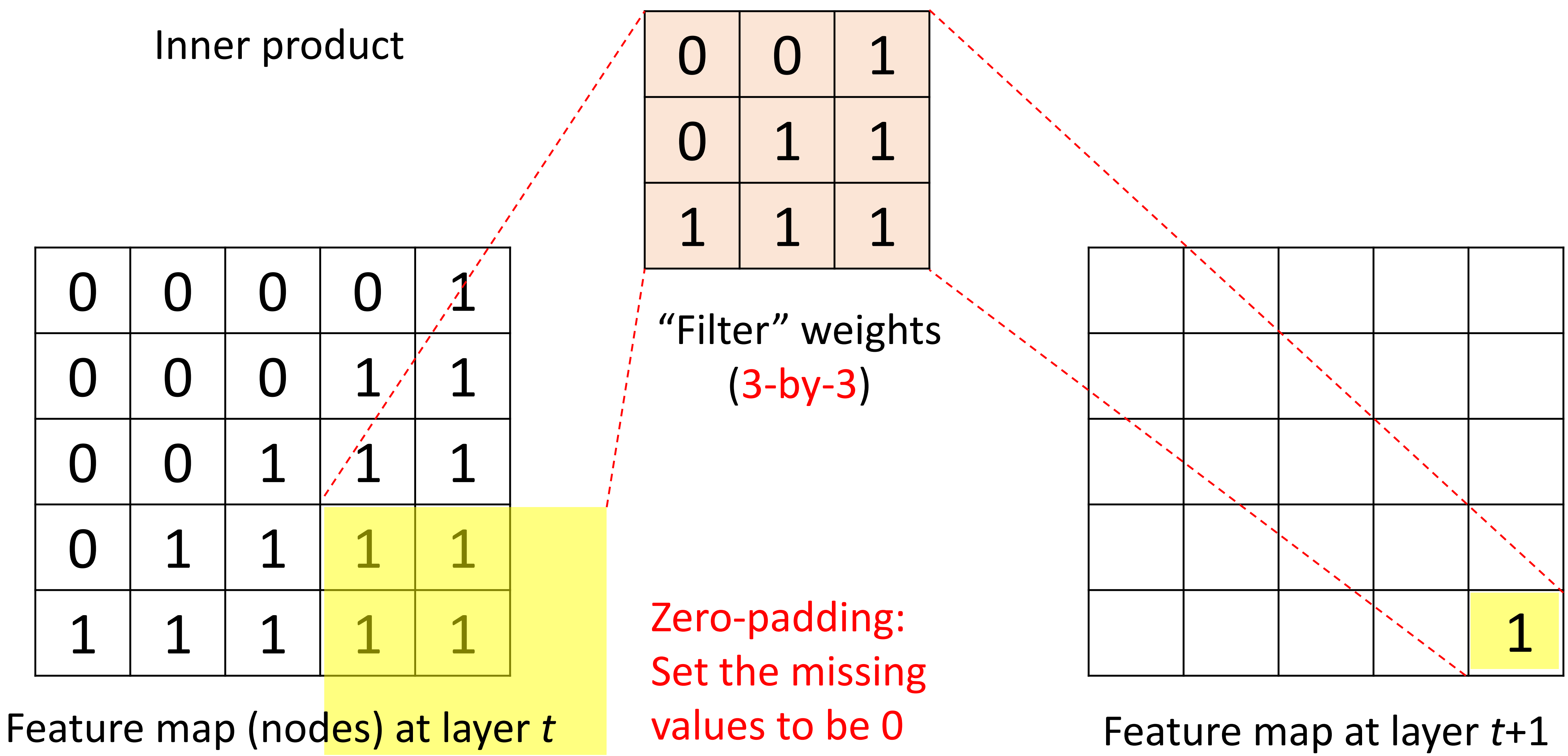
Inner product



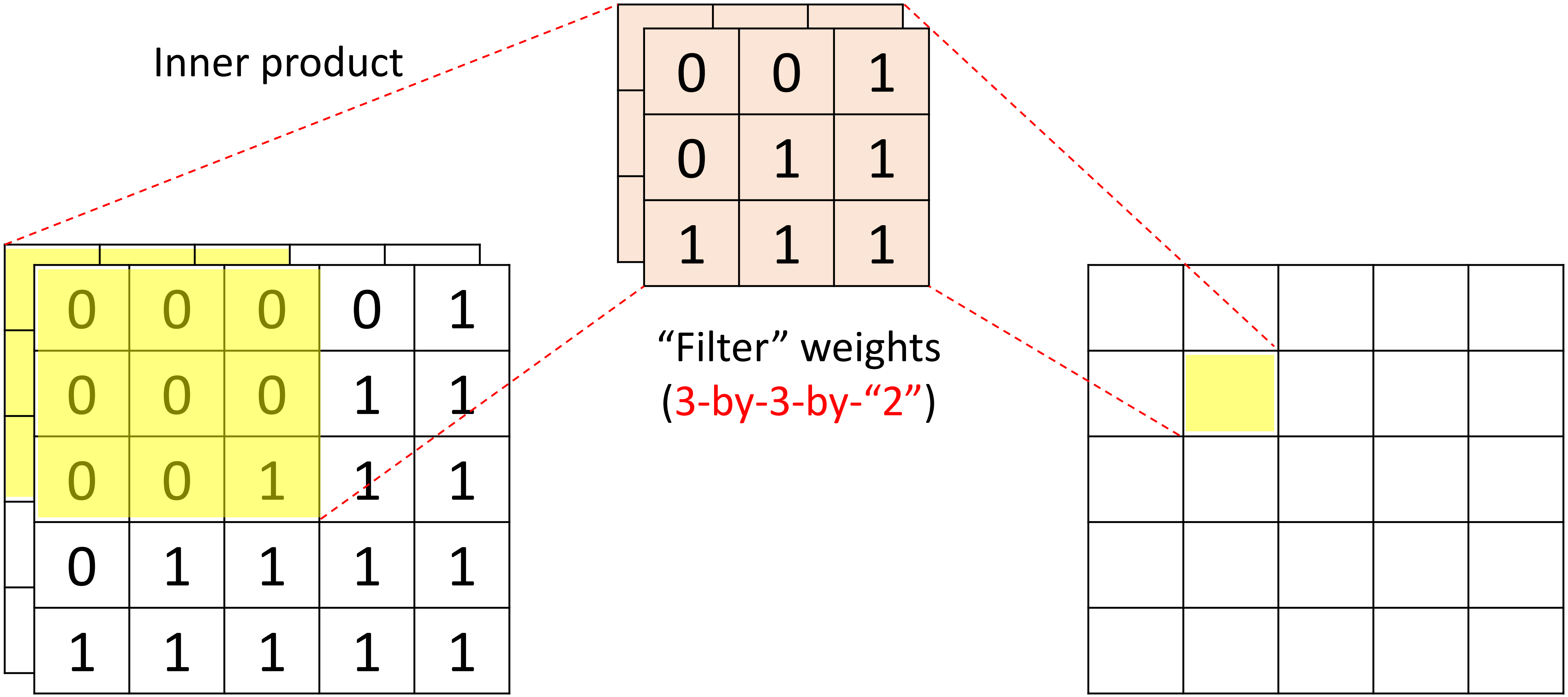
Feature map (nodes) at layer *t*

Feature map at layer *t*+1

Convolution



Convolution

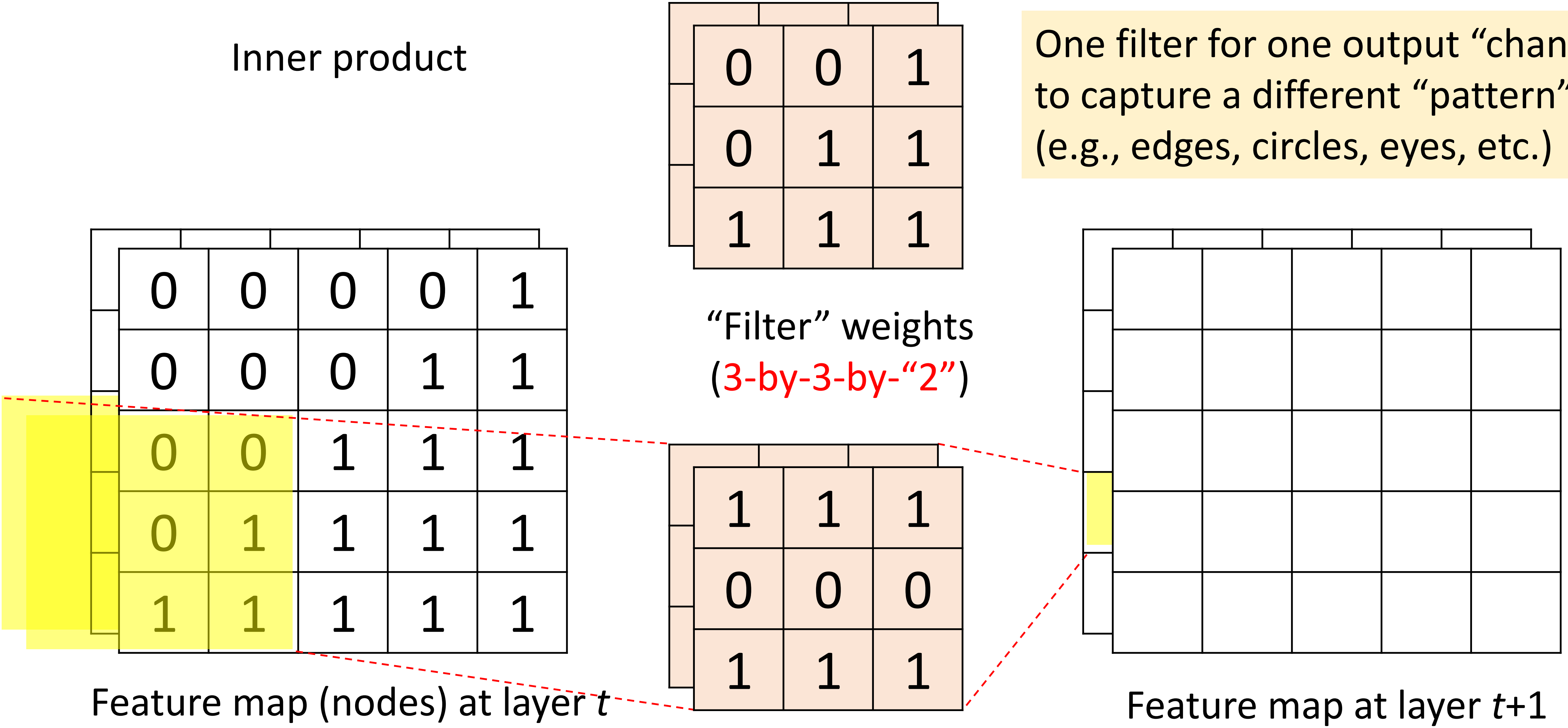


Feature map (nodes) at layer *t*

Feature map at layer *t*+1

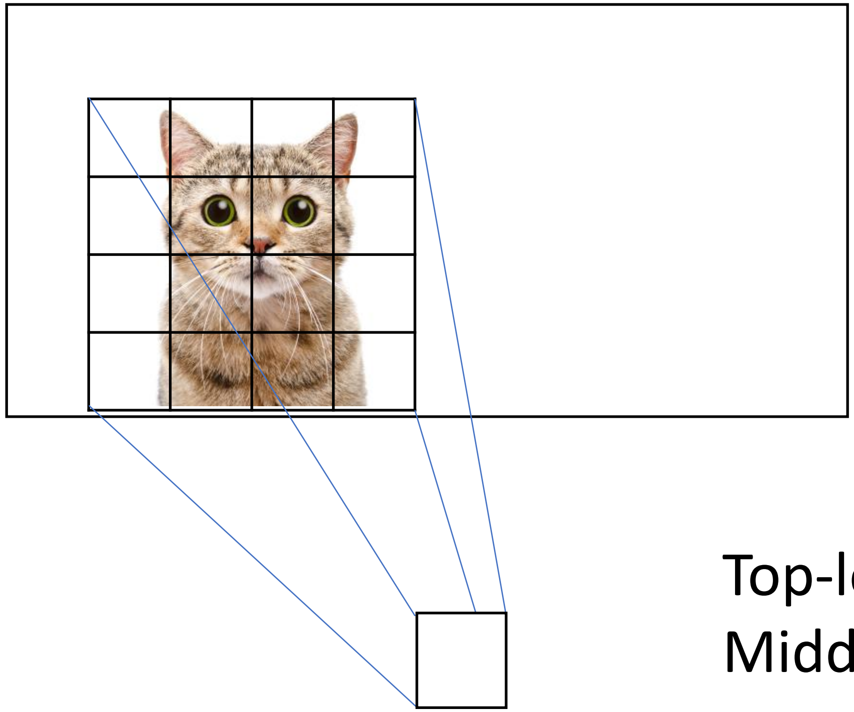
Convolution

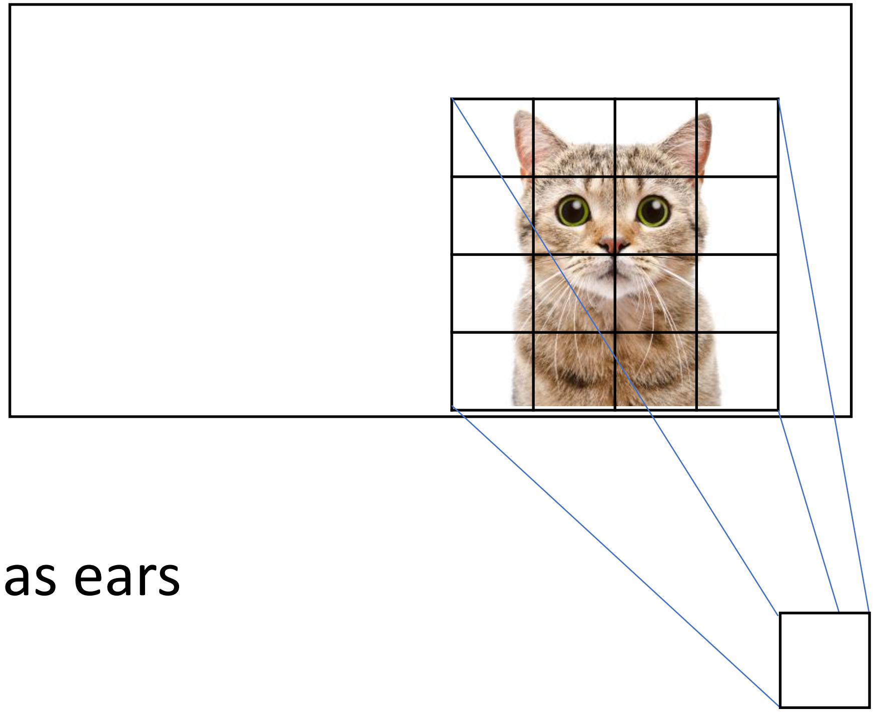
One filter for one output "channel" to capture a different "pattern" (e.g., edges, circles, eyes, etc.)



Convolution: properties

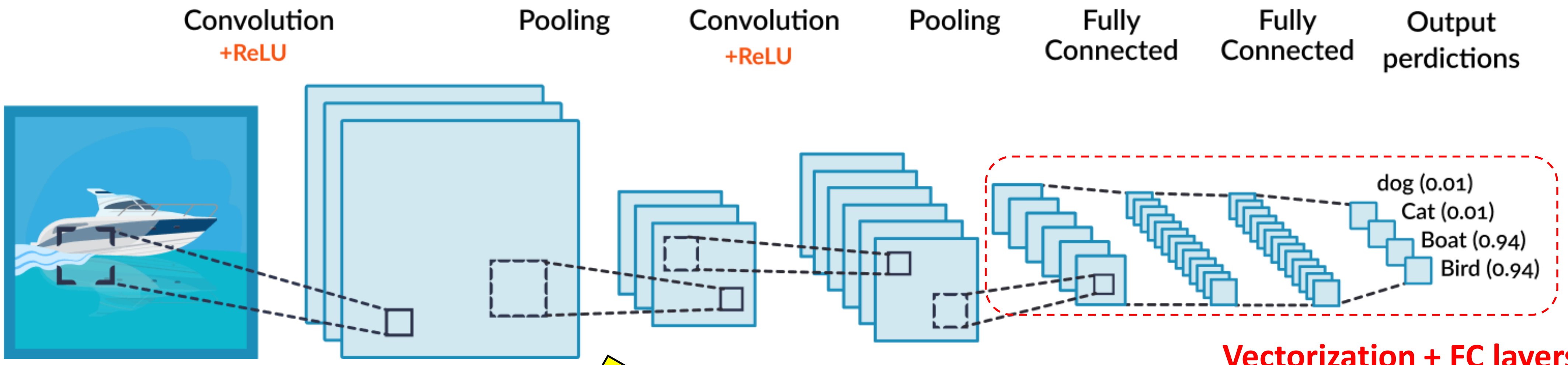
- Process nearby pixels together
- Translation invariant: “local patterns” can show up at different pixel locations
- Can process arbitrary-size images



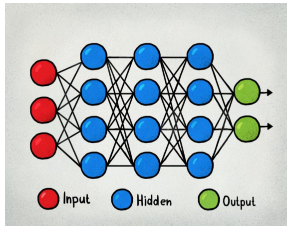


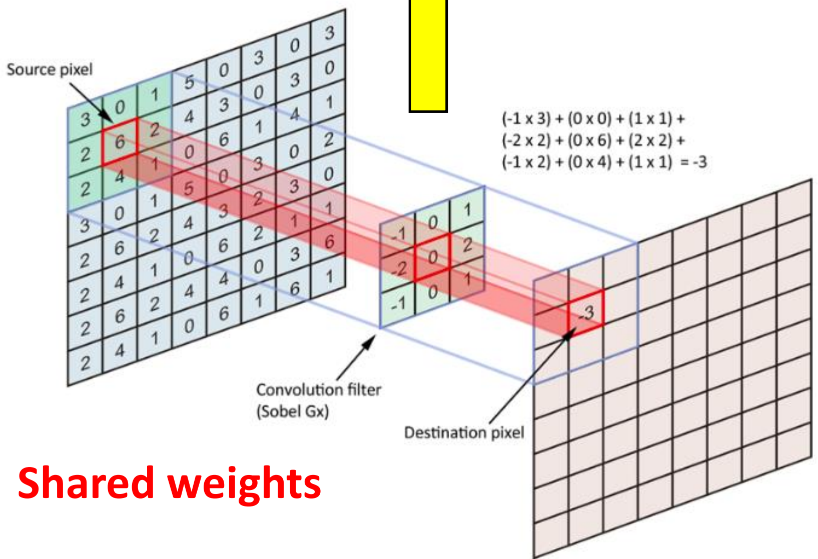
Top-left, Top right: has ears
Middle: has eyes

Convolutional neural networks (CNN)



Vectorization + FC layers





Shared weights

Max pooling + down-sampling

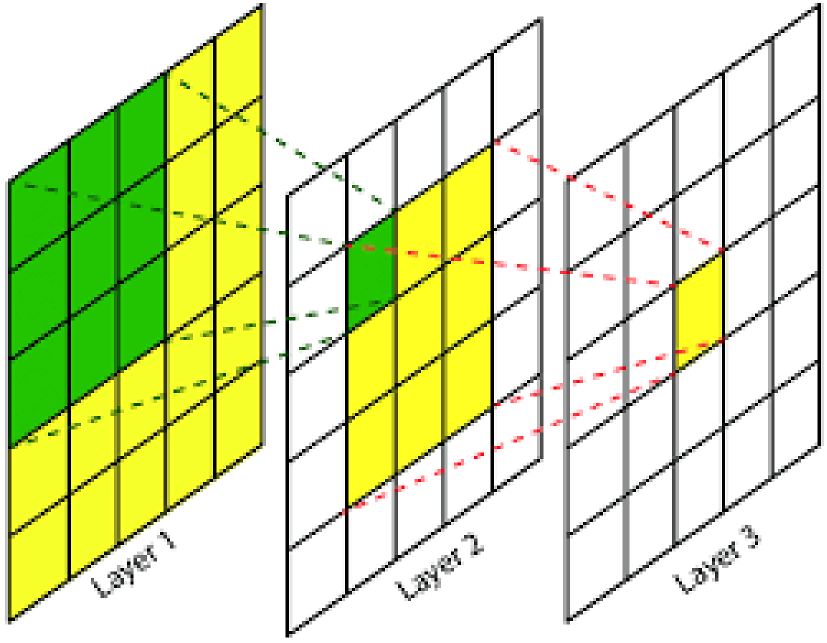
12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

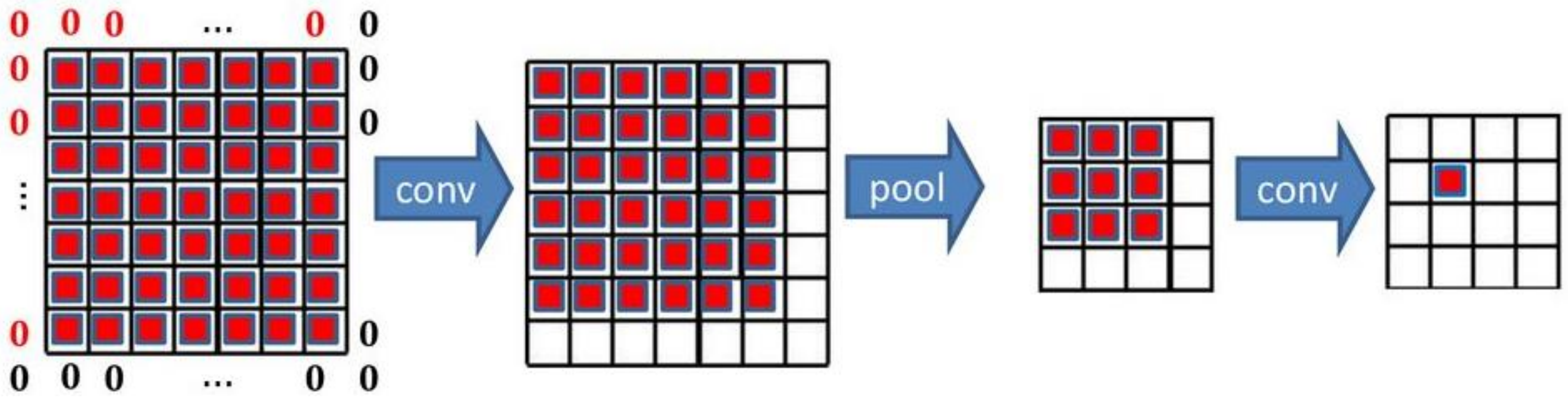
2×2 Max-Pool $\longrightarrow$

20	30
112	37

- Remove redundancy
- Translation-invariant
- Enlarge receptive filed

Receptive field

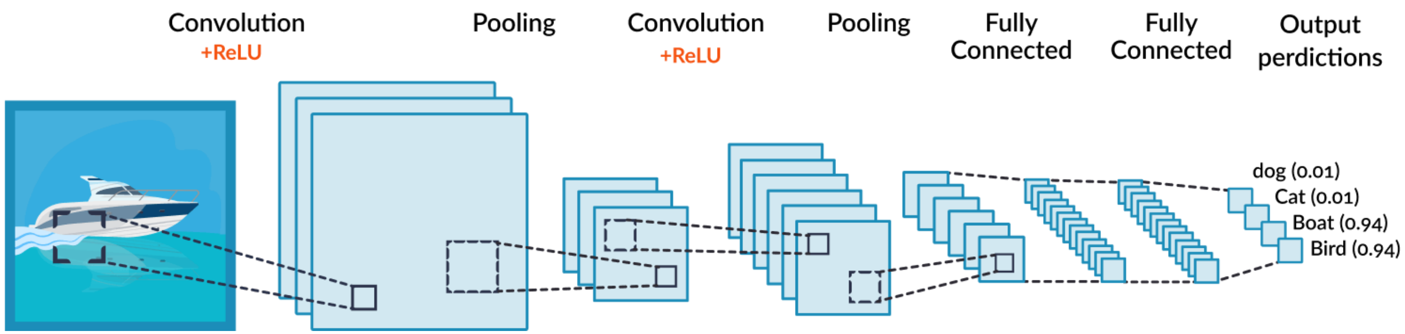




Linear receptive field

Exponential receptive field
(with pooling + down-sampling)

Layers of feature maps (representations)



What does a large response at each layer/channel mean?

First Layer Representation Second Layer Representation Third Layer Representation

Representative CNN networks

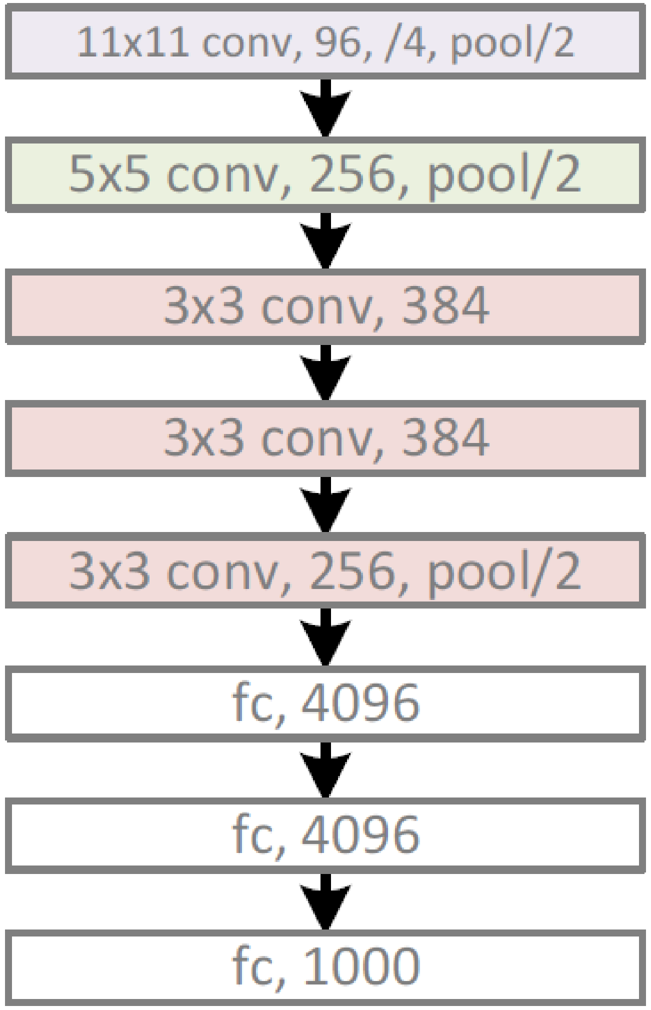
- AlexNet

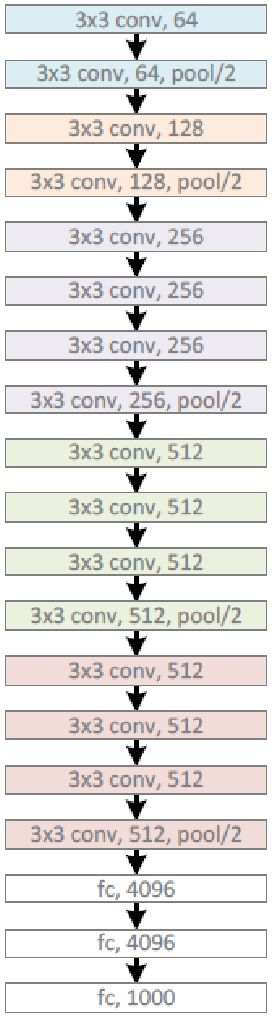
[Krizhevsky et al., 2012]

- VGGnet

[Simonyan et al., 2015]

- A block: computation
- Edge: nodes/tensors

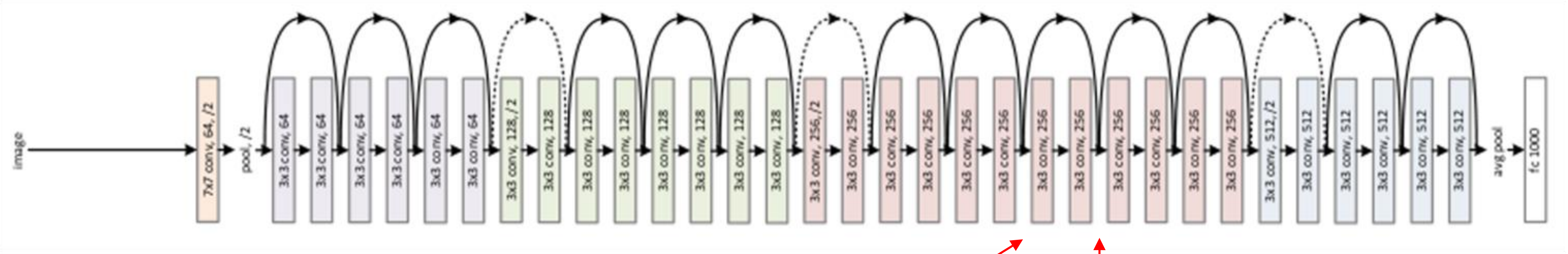




Representative CNN networks

- ResNet

[He et al, 2016]

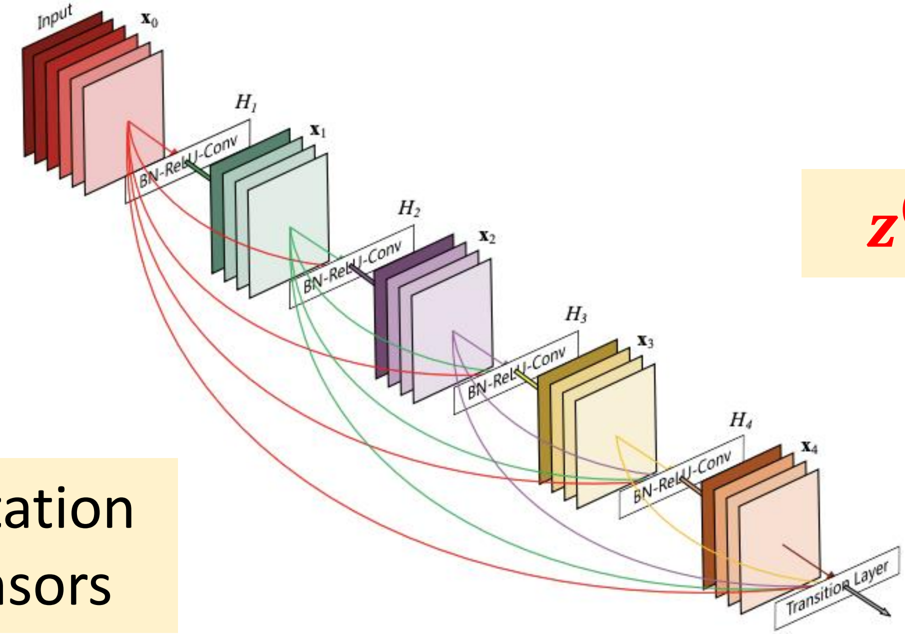


$\boldsymbol{z}^{(t-2)}$ $\boldsymbol{z}^{(t)}$

$$\boldsymbol{z}^{(t)} = \boldsymbol{z}^{(t-2)} + \mathrm{conv}\left(\boldsymbol{z}^{(t-2)}\right)$$

- DenseNet

[Huang et al, 2017]



- A block: computation
- Edge: nodes/tensors

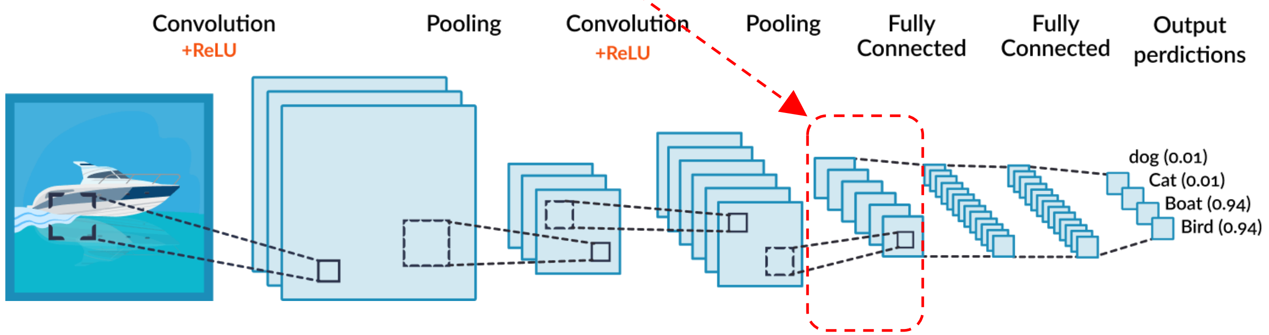
Advantages:

- Optimization
- Collect more information

Representative CNN networks

A general architecture involves

- Multiple layers of convolutions + ReLU (nonlinearity) + pooling + striding
- These result in a (final) feature map
    - Positions on the map correspond to the image
- The map goes through FC layers (MLP)
- Usually, we keep the network till the feature map
    - For feature extraction
    - For down-stream tasks
    - For image-to-image search



Training a CNN for classification

- Model: $\operatorname*{argmax}_{c} \boldsymbol{w}_c^T f_{\boldsymbol{\theta}}(\boldsymbol{x})$
  - What to learn: weights of convolution filters $\boldsymbol{\theta}$, and $\{\boldsymbol{w}_c\}$

- Training data: $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$

100: elephant

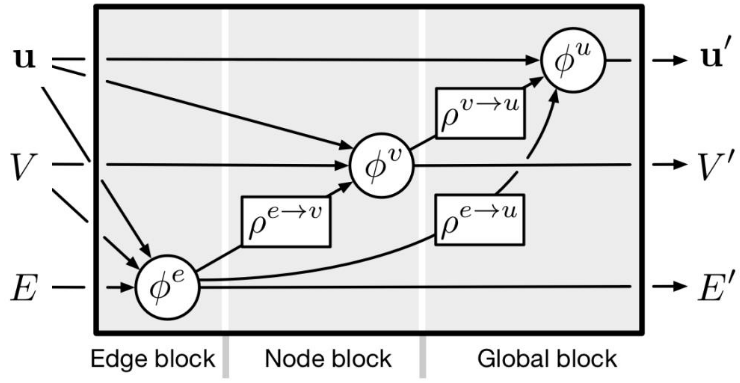
Minimize the empirical risk

$$\min \sum_{n=1}^{N} \ell(\boldsymbol{x}_n, y_n; \boldsymbol{\theta}, \{\boldsymbol{w}_c\})$$

- Objective/loss function: $\ell(\boldsymbol{x}, y; \boldsymbol{\theta}, \{\boldsymbol{w}_c\})$
  - For example, $\mathbf{1}\left[\operatorname*{argmax}_{c} \boldsymbol{w}_c^T f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq y\right]$, softmax

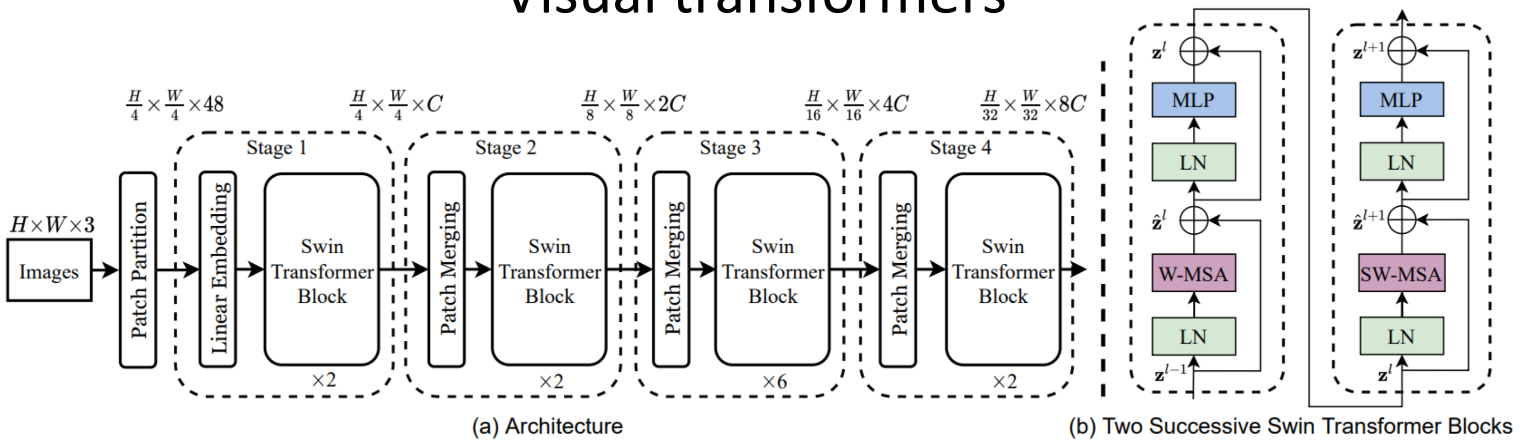
The diversity of deep learning models

Graph neural networks



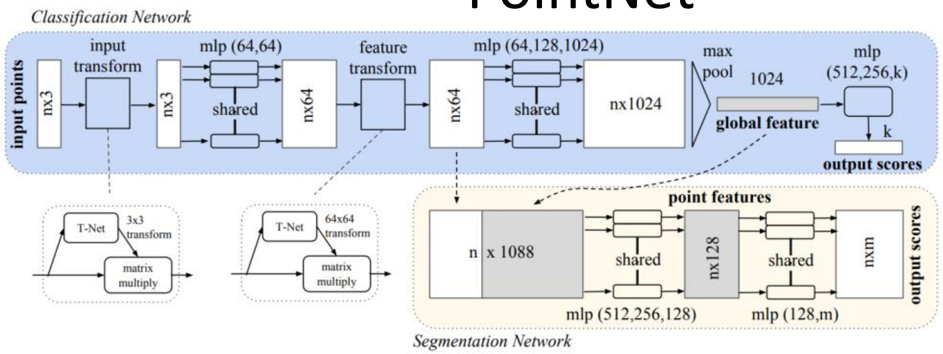
[Battaglia et al., 2018]

Visual transformers



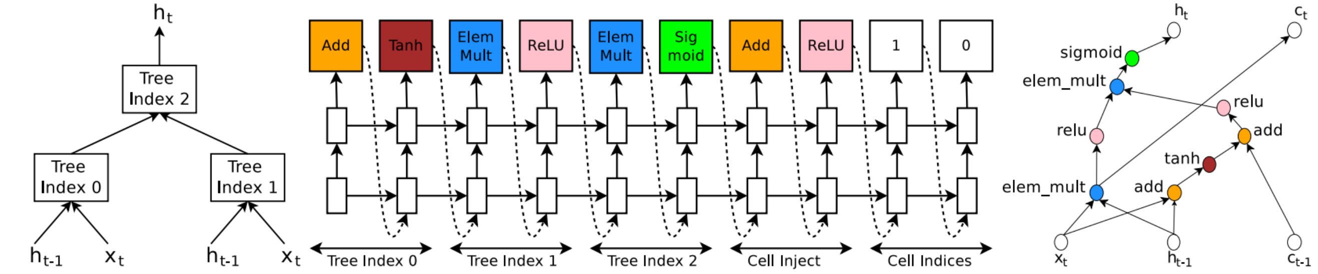
[Liu et al., 2021]

PointNet



[Qi et al., 2017]

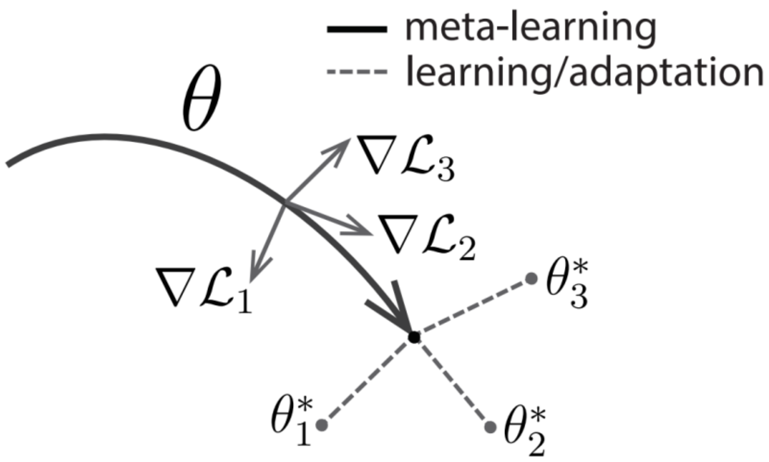
Neural architecture search



[Zoph et al., 2017]

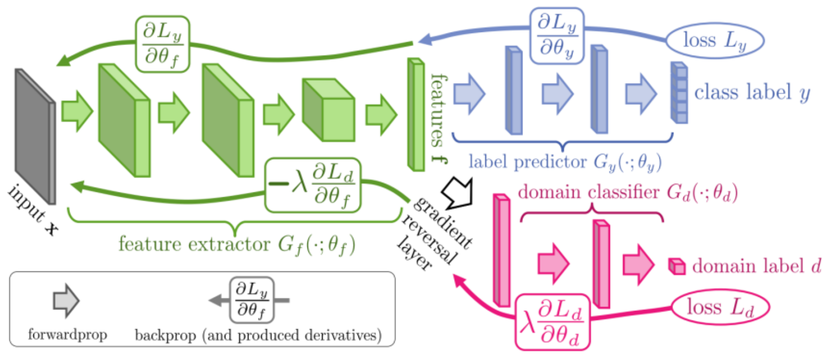
The diversity of deep learning algorithms

Meta-learning



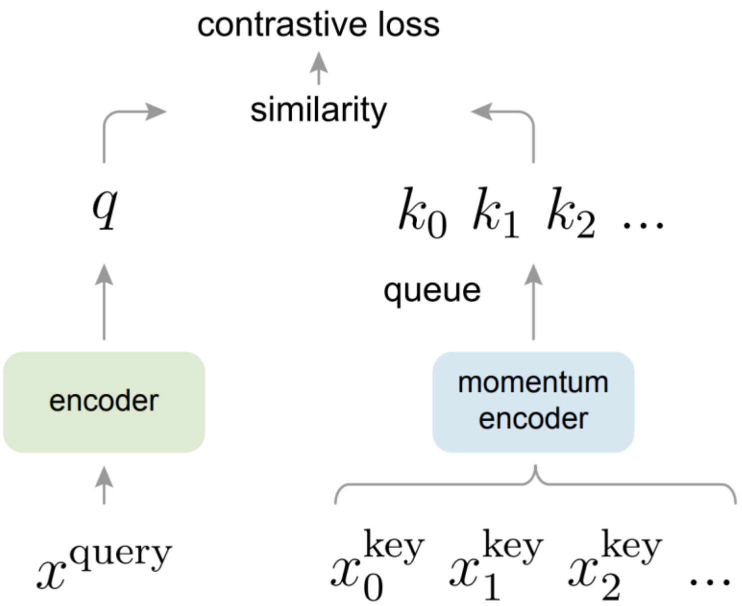
[Finn et al., 2017]

Adversarial learning



[Ganin et al., 2016]

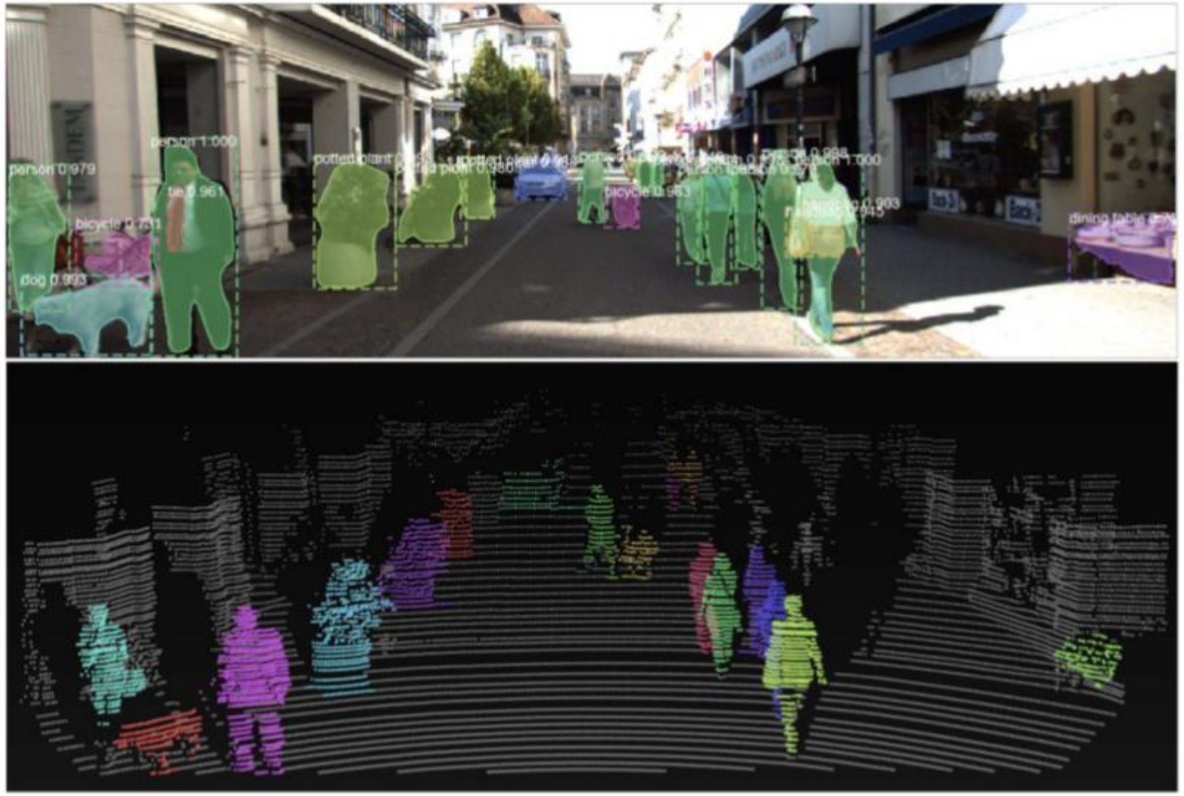
Contrastive learning



[He et al., 2020]

Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



Visual transformer

- A newly emerging way to generate the (final) feature map $f_{\boldsymbol{\theta}}(\boldsymbol{x})$
  - Inspired by the transformer blocks in NLP

$\boldsymbol{x}$ $\qquad$ $f_{\boldsymbol{\theta}}(\boldsymbol{x})$

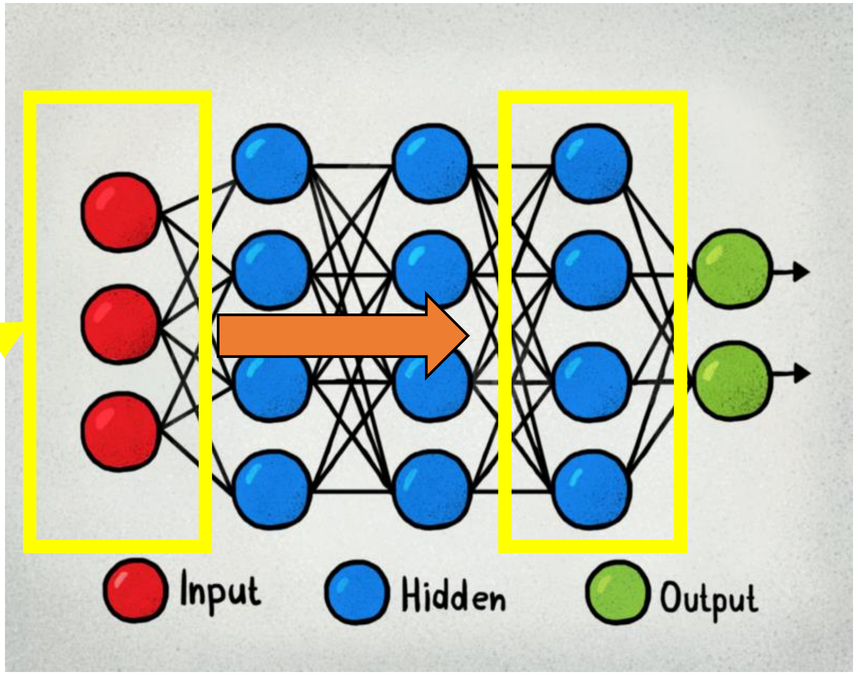
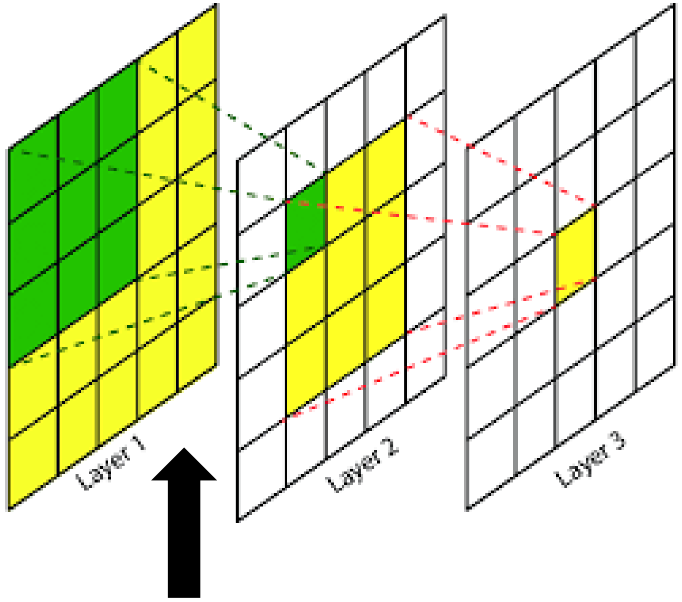


Image (pixels)

$$\text{Prediction} = \underset{c}{\operatorname{argmax}}\, \boldsymbol{w}_c^T\, f_{\boldsymbol{\theta}}(\boldsymbol{x})$$

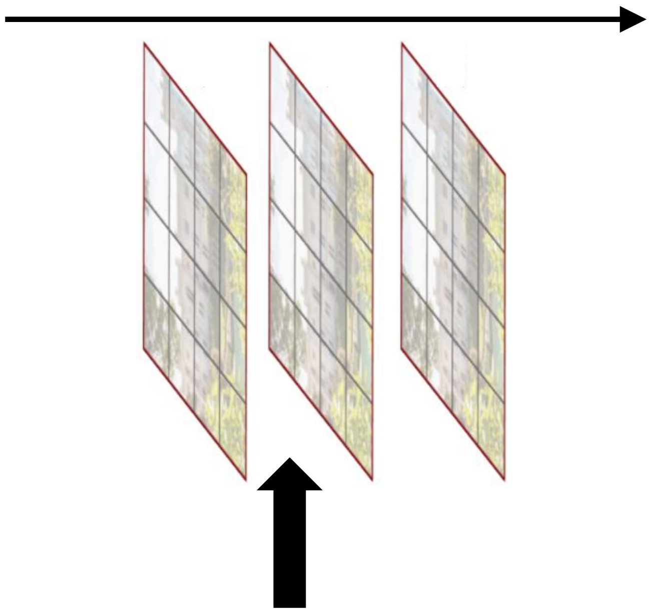
CNN vs. Visual transformer

CNN



Convolution

Visual transformer

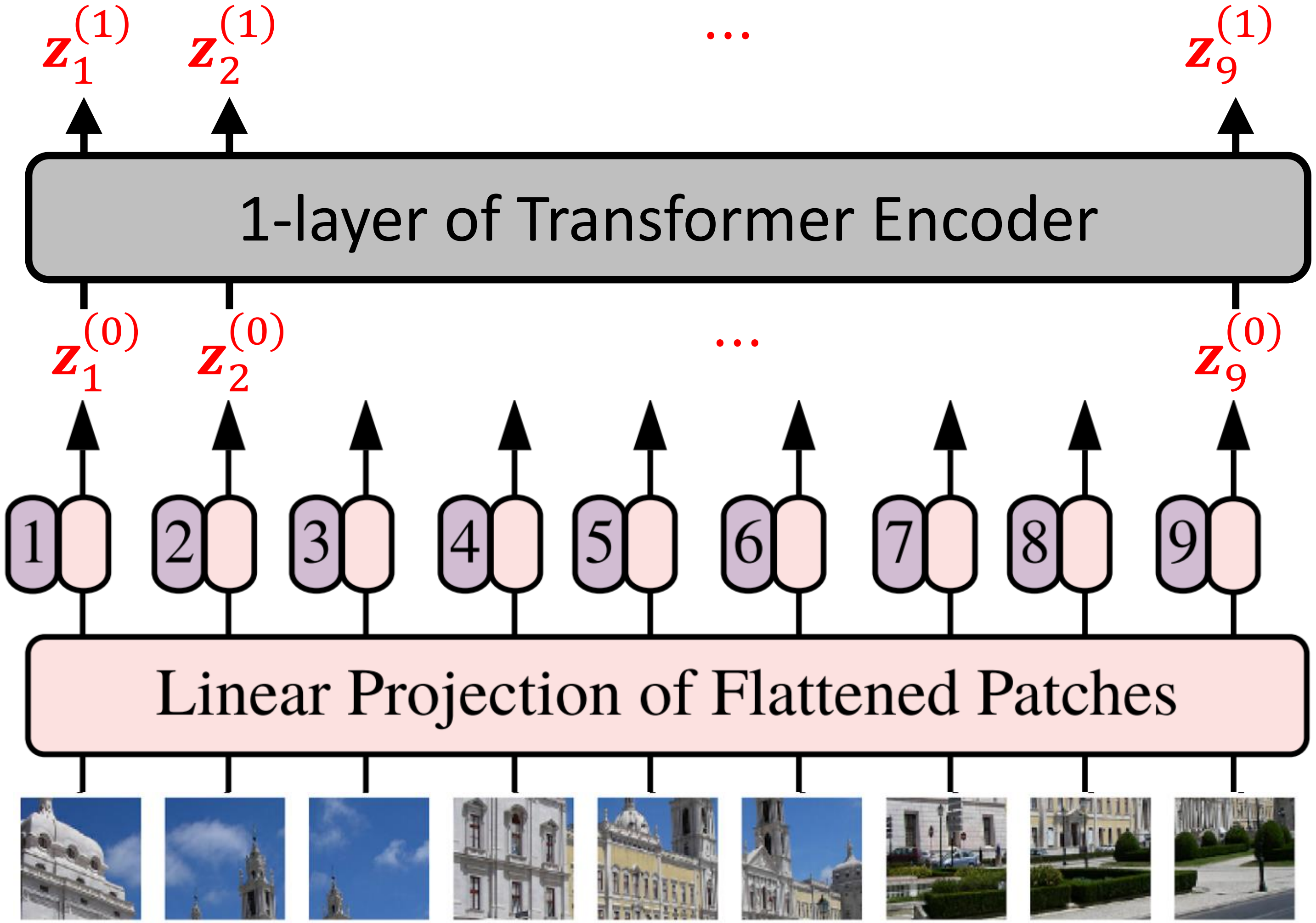


Transformer

Visual transformer

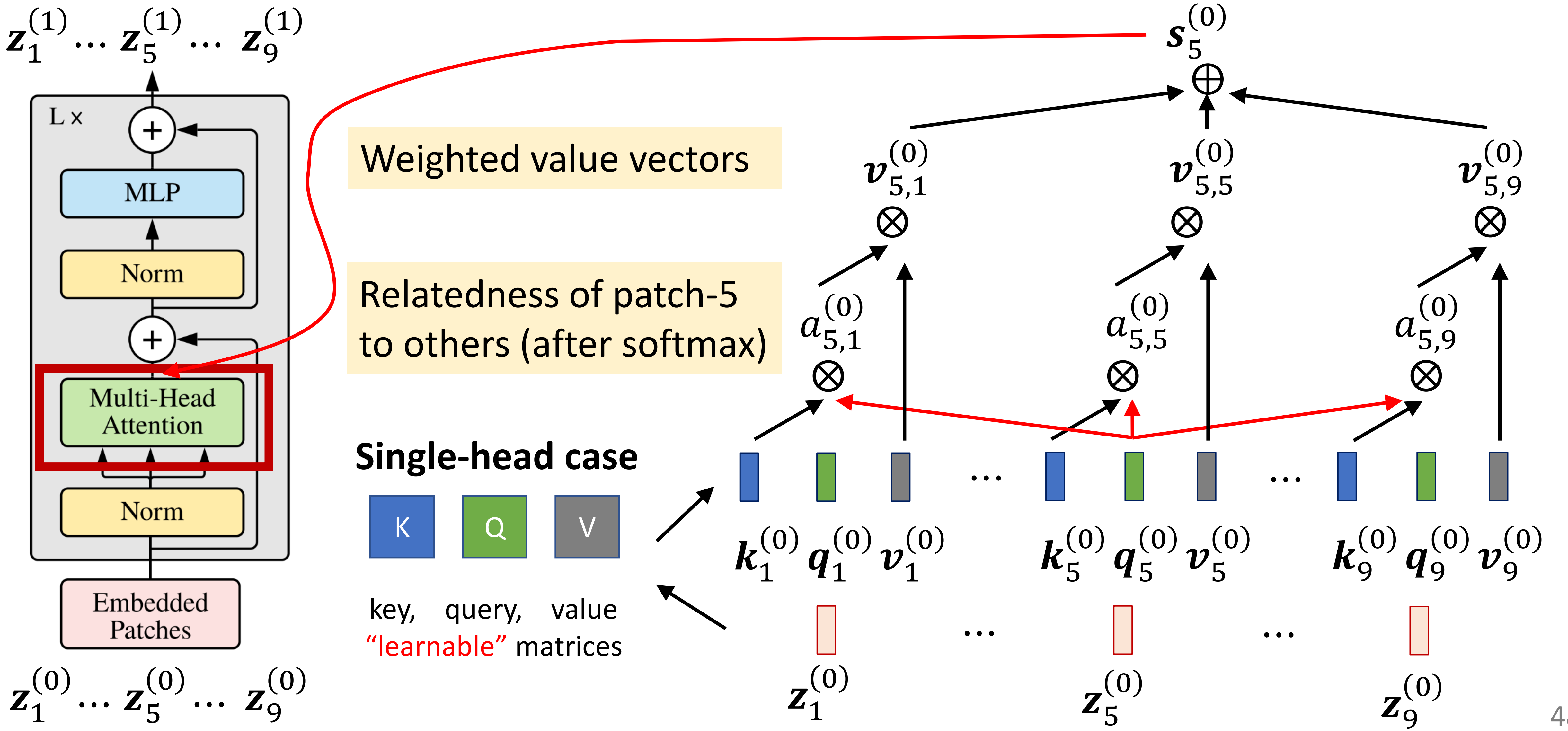
[Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2021]

(2) Vectorize each of them
+ encode each with a shared MLP
+ “spatial” encoding



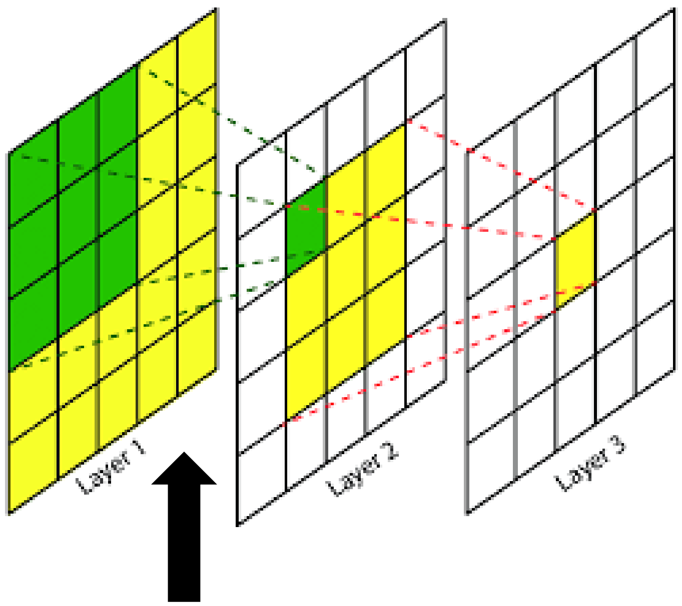
(1) Split an image into patches

1-layer of transformer encoder



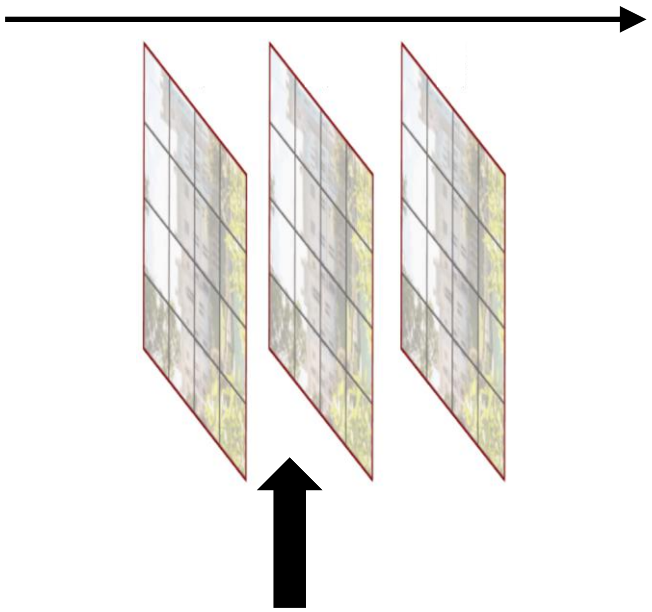
CNN vs. Visual transformer

CNN



Convolutions

Visual transformer



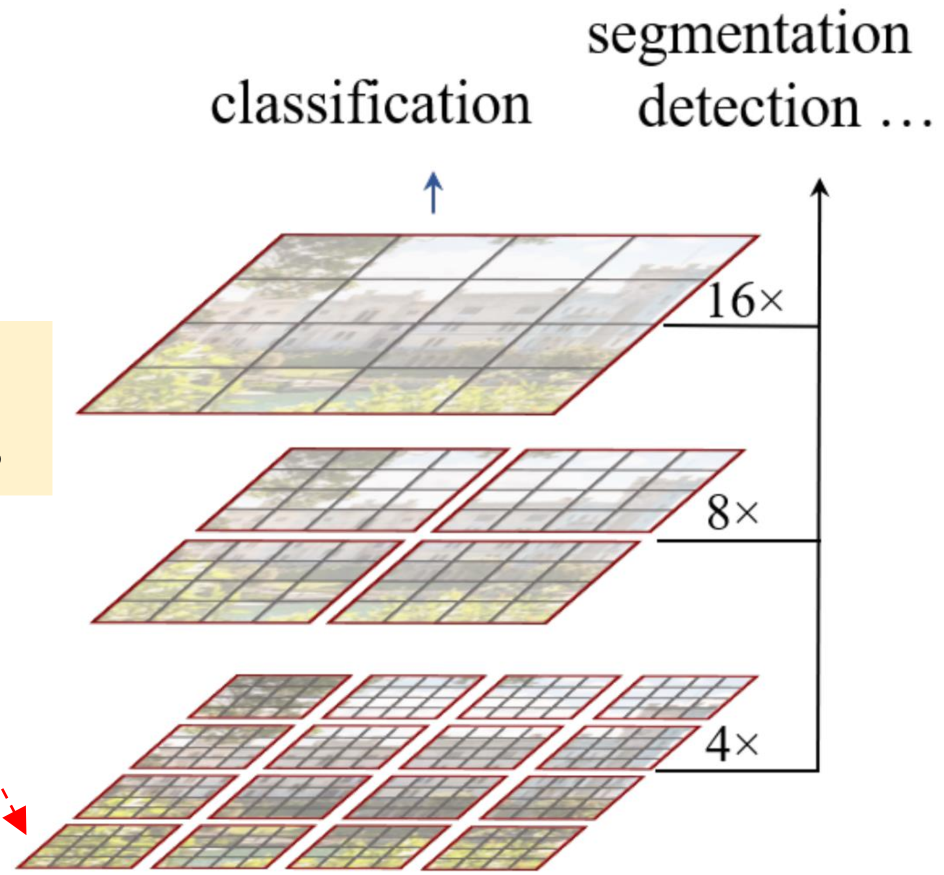
Transformer

- Enable large receptive filed and long-distance relationship
- Enable different local relationships (based on $\boldsymbol{k}_i^{(0)} \otimes \boldsymbol{q}_j^{(0)}$)

Swin transformer

[Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021]

- Consider smaller patches and local “transformer”
- Produce feature maps of different resolutions, like CNNs



ImageNet classification accuracy

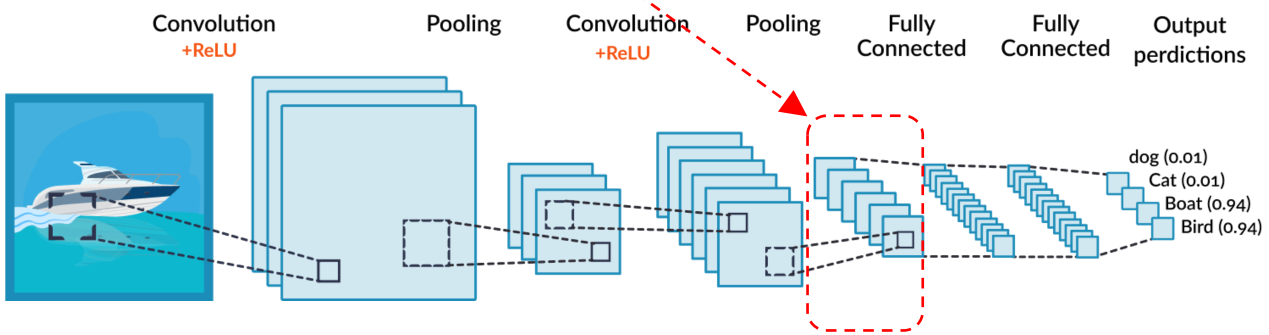
[Liu et al., 2021]

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

Short summary

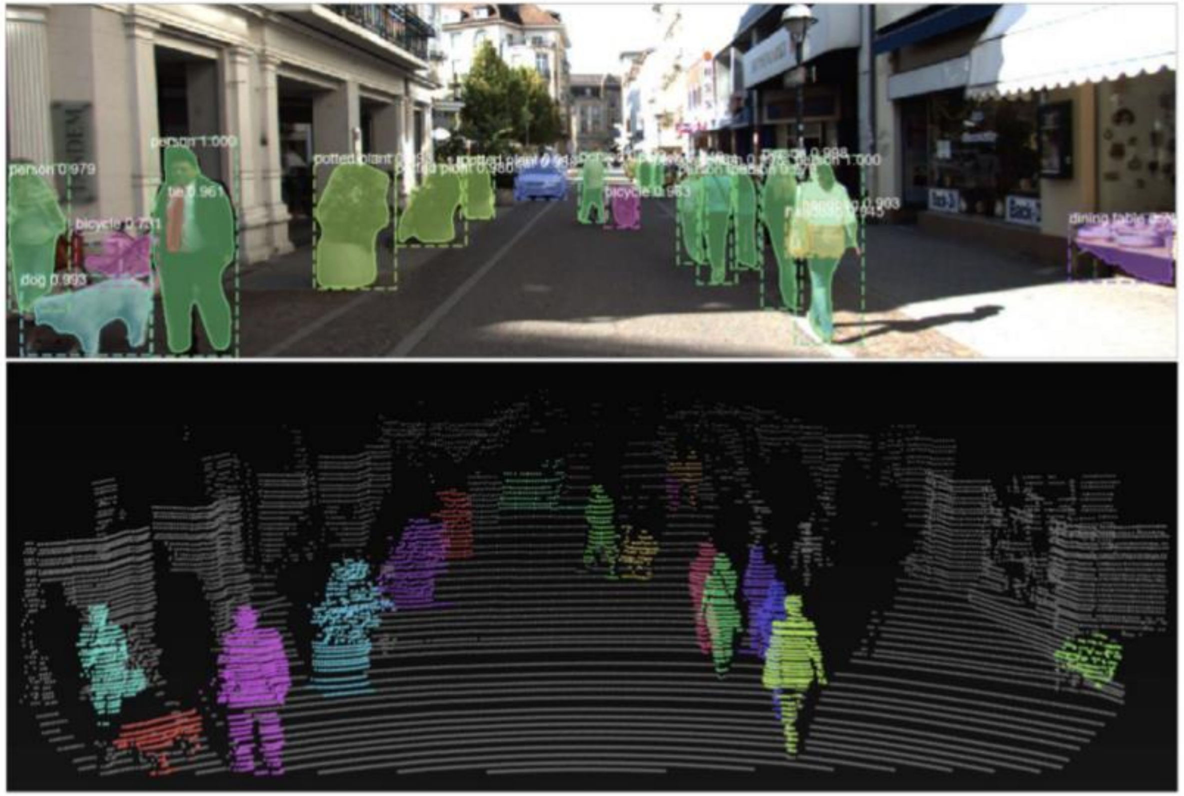
A general architecture of CNN or visual transformers involves

- Multiple layers of computations + nonlinearity + (pooling + striding)
- These result in a (final) feature map
- The map goes through FC layers (MLP)
- Usually, we keep the network till the feature map



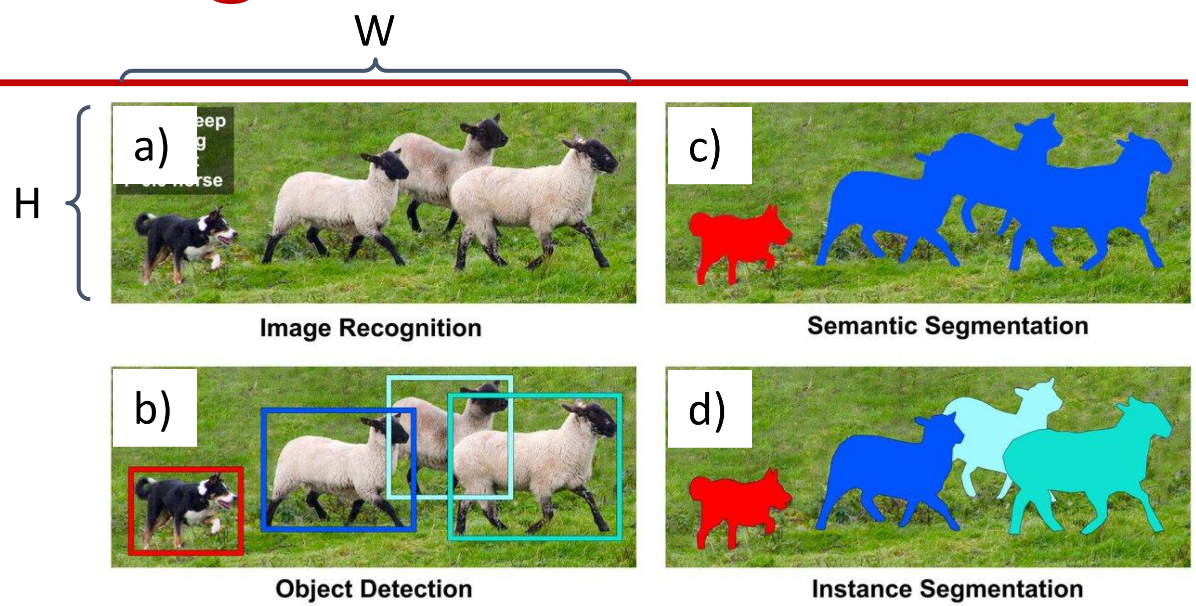
Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



Representative 2D recognition tasks

- **"Same" input: images**



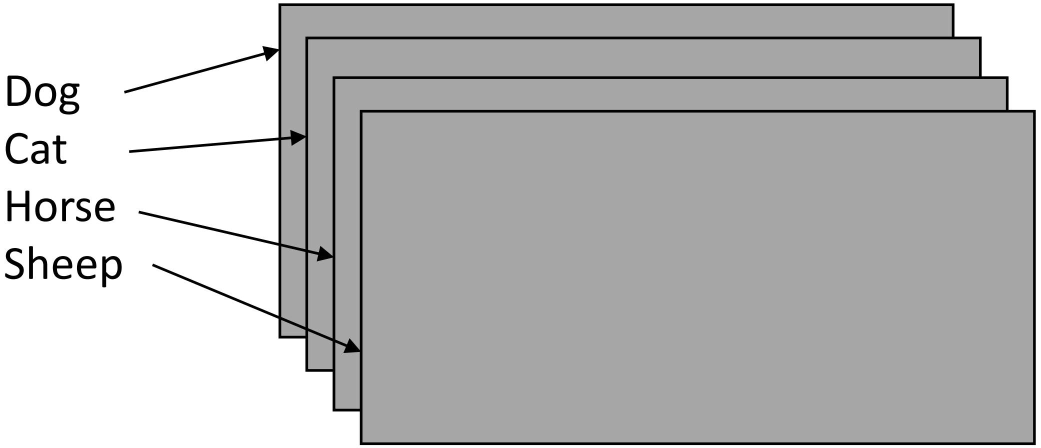
- **"Different" outputs:**

a) A C-dim class probability vector

b) A set of bounding boxes, each with box location and class probability

c) An W x H x C feature map

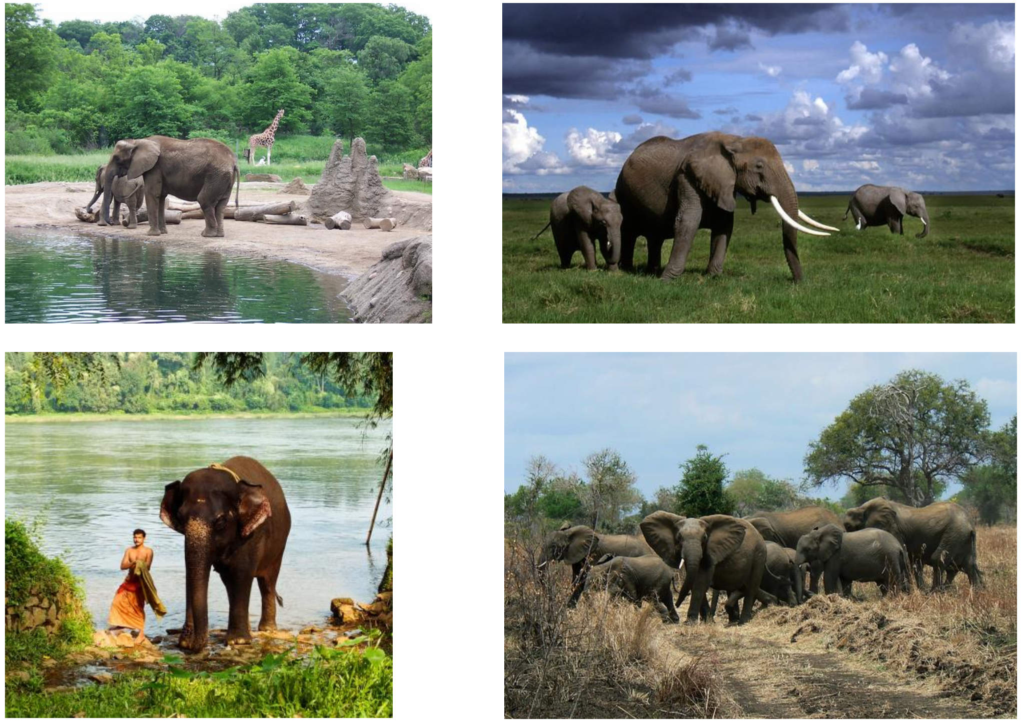
d) A combination of b) and c)



- **"Different" labeled training data**

Object- vs. scene centric images

ImageNet [object-centric]:
- Image-level class label
- 1K classes (~1M images)
- 21K classes (~14M images)



MSCOCO [scene-centric]:
- Instance-level label
- 82 classes (~0.3M images)

Object- vs. scene centric images






- Object-centric images usually contain a **single** class of objects.
- **Object frequency and semantic cues** in different kinds of images can be different!

Semantic segmentation

Semantic segmentation

- Every "pixel" to have a class label

- **Properties:**

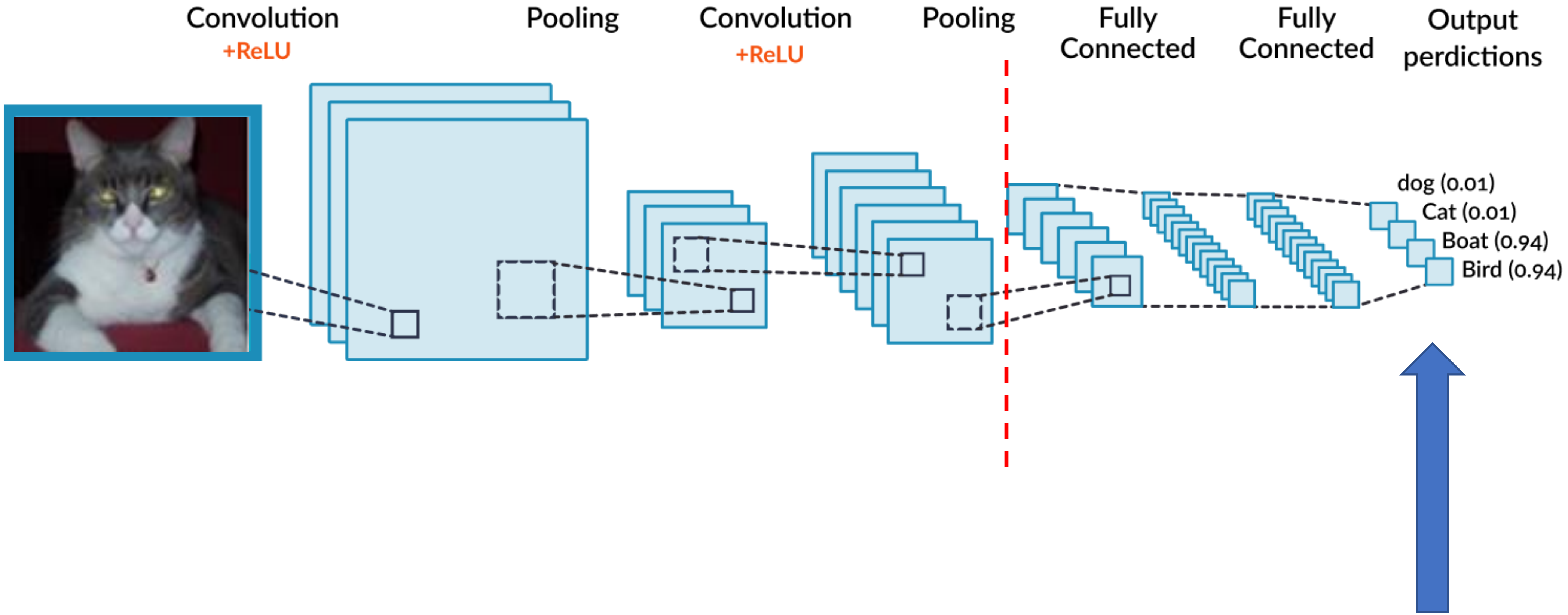
➢High-resolution output

➢Context

➢Localization

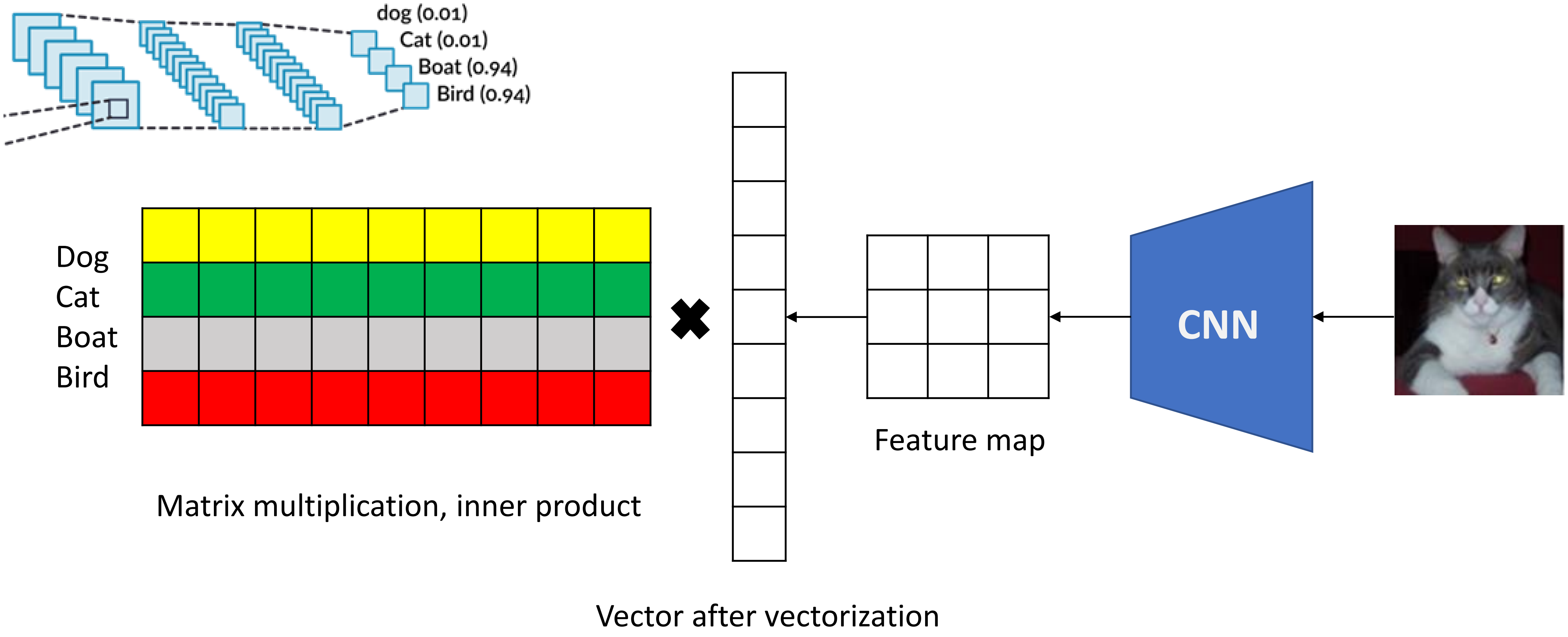
➢Multi-scale

New architecture?

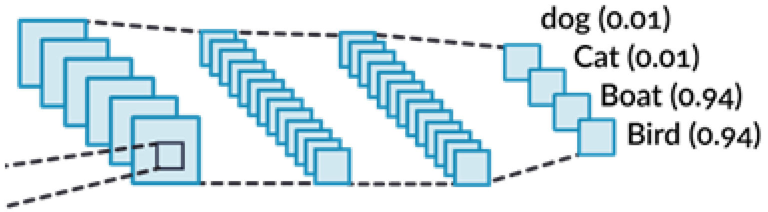


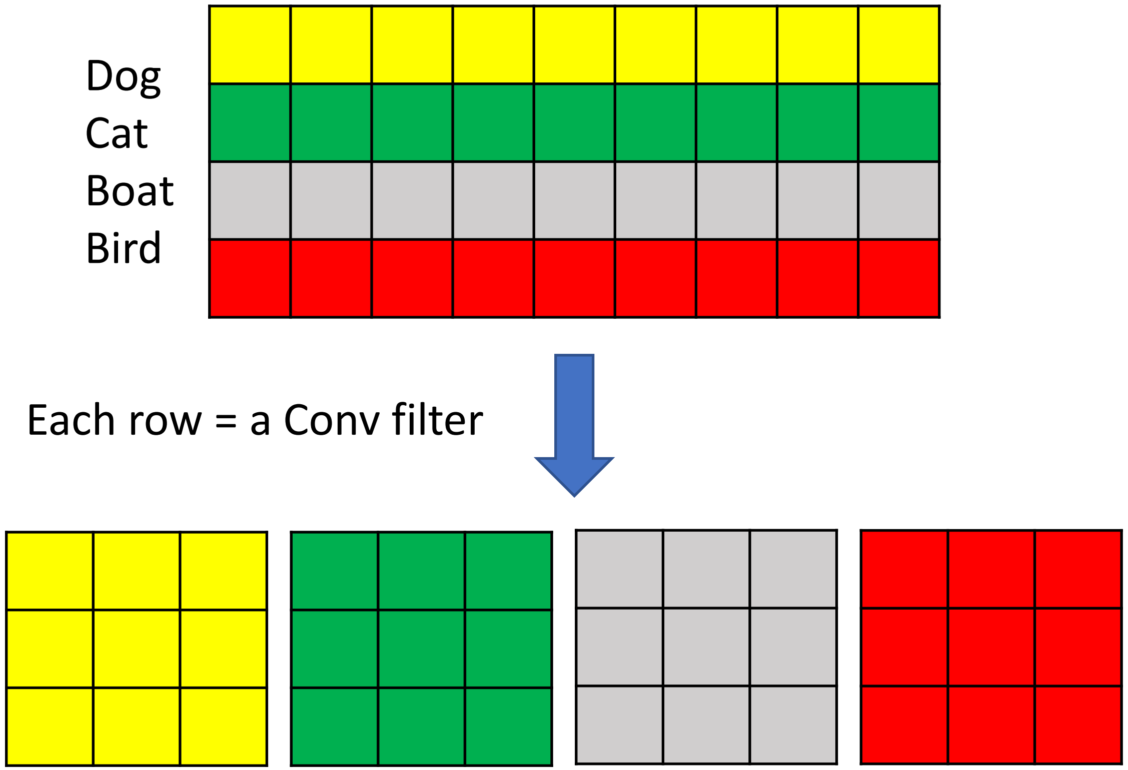
Single spatial output!

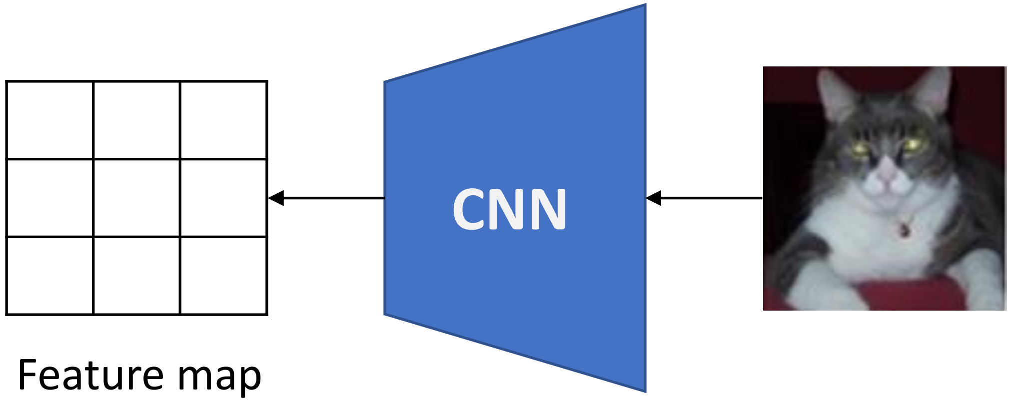
Fully-convolutional network (FCN)



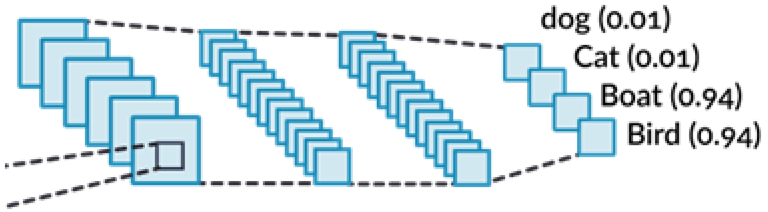
Fully-convolutional network (FCN)

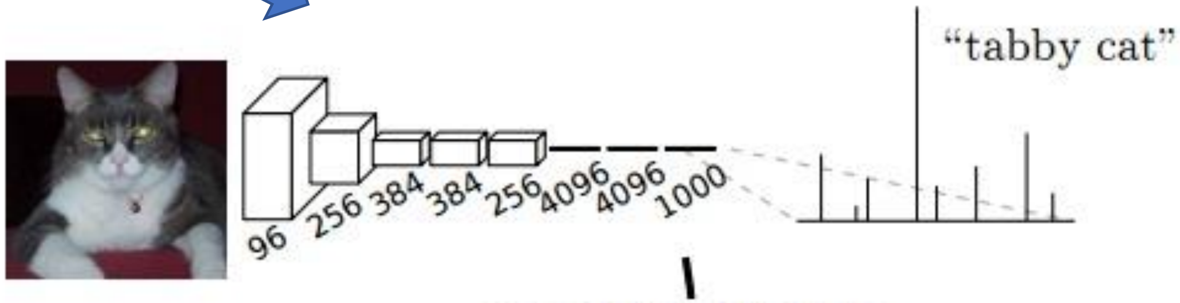


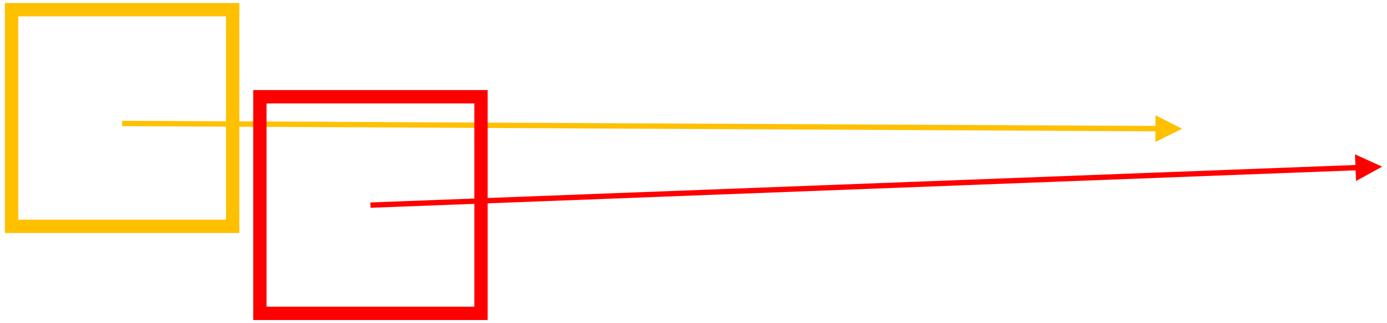




Fully-convolutional network (FCN)



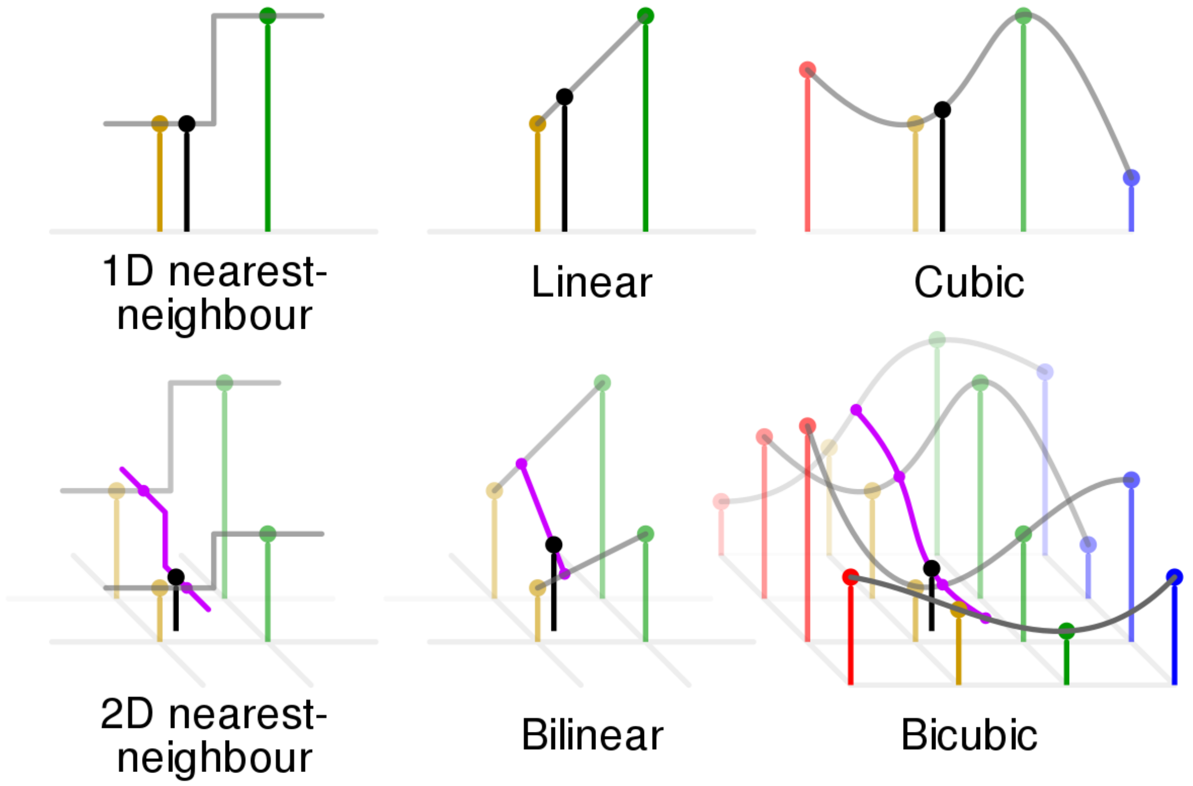




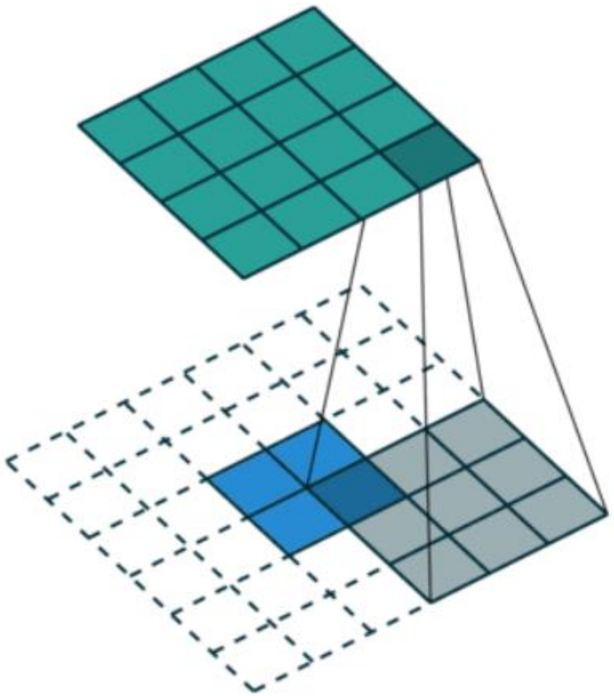
[Long et al., Fully Convolutional Networks for Semantic Segmentation, CVPR 2015]

Up-sampling

Interpolation



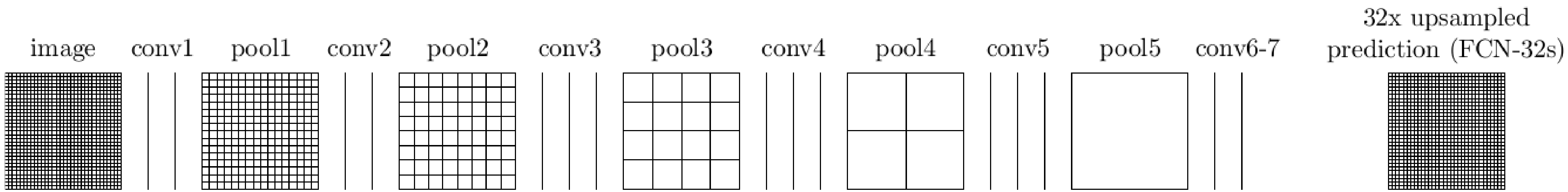
Deconvolution



Upsampling Via Deconvolution (Blue: Input, Green: Output)

Fully-convolutional network (FCN)

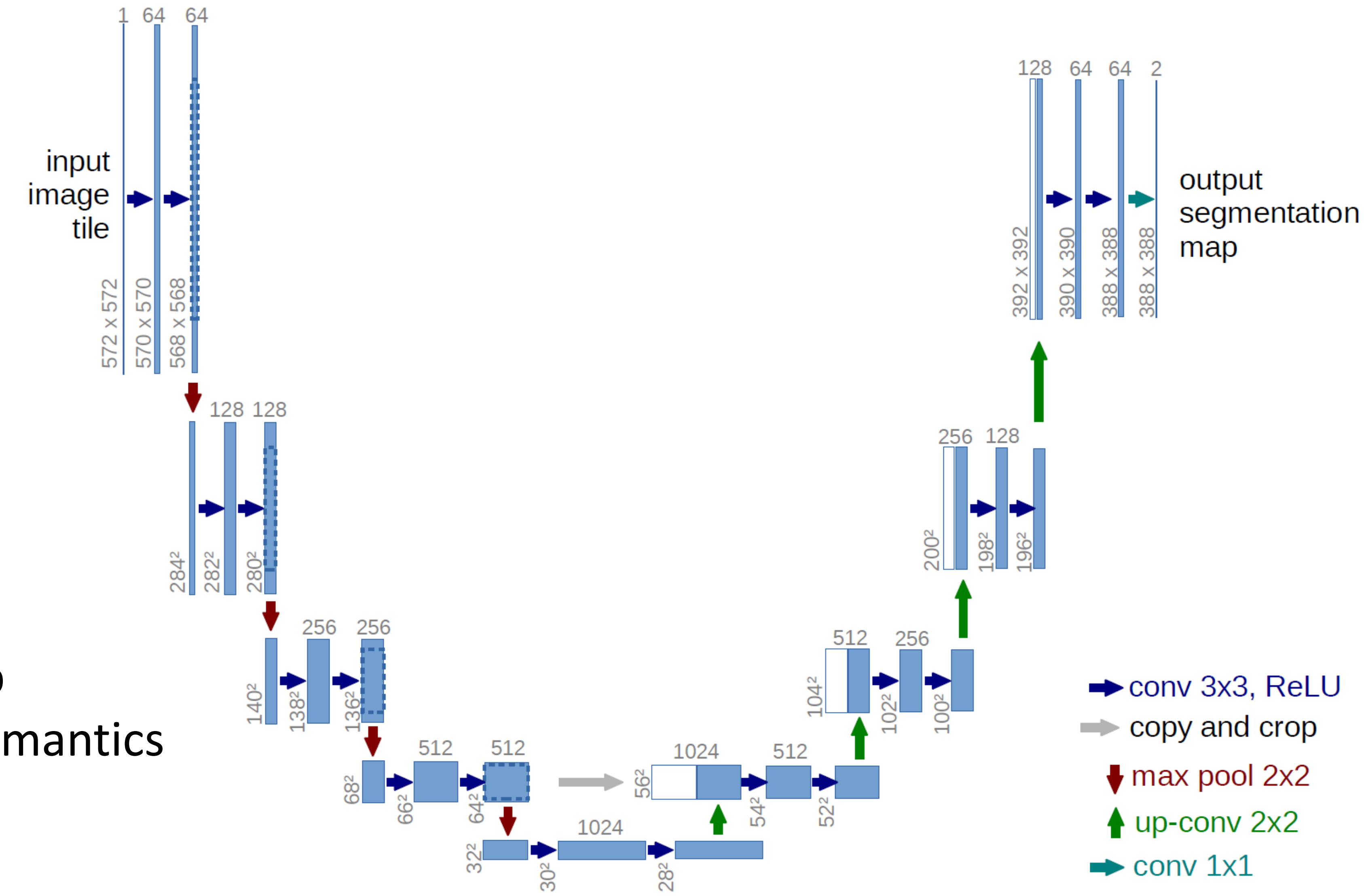
Help
context + semantics



[Long et al., Fully Convolutional Networks for Semantic Segmentation, CVPR 2015]

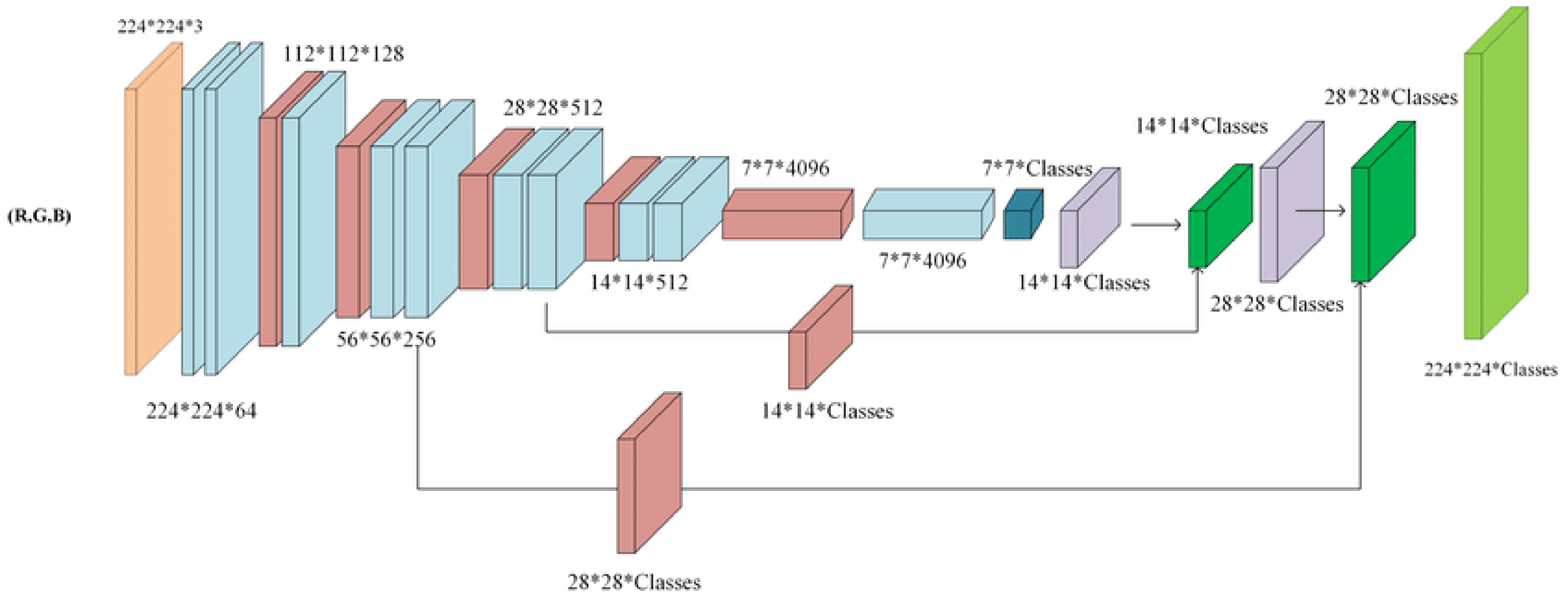
U-Net

[Ronneberger et al., U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015]



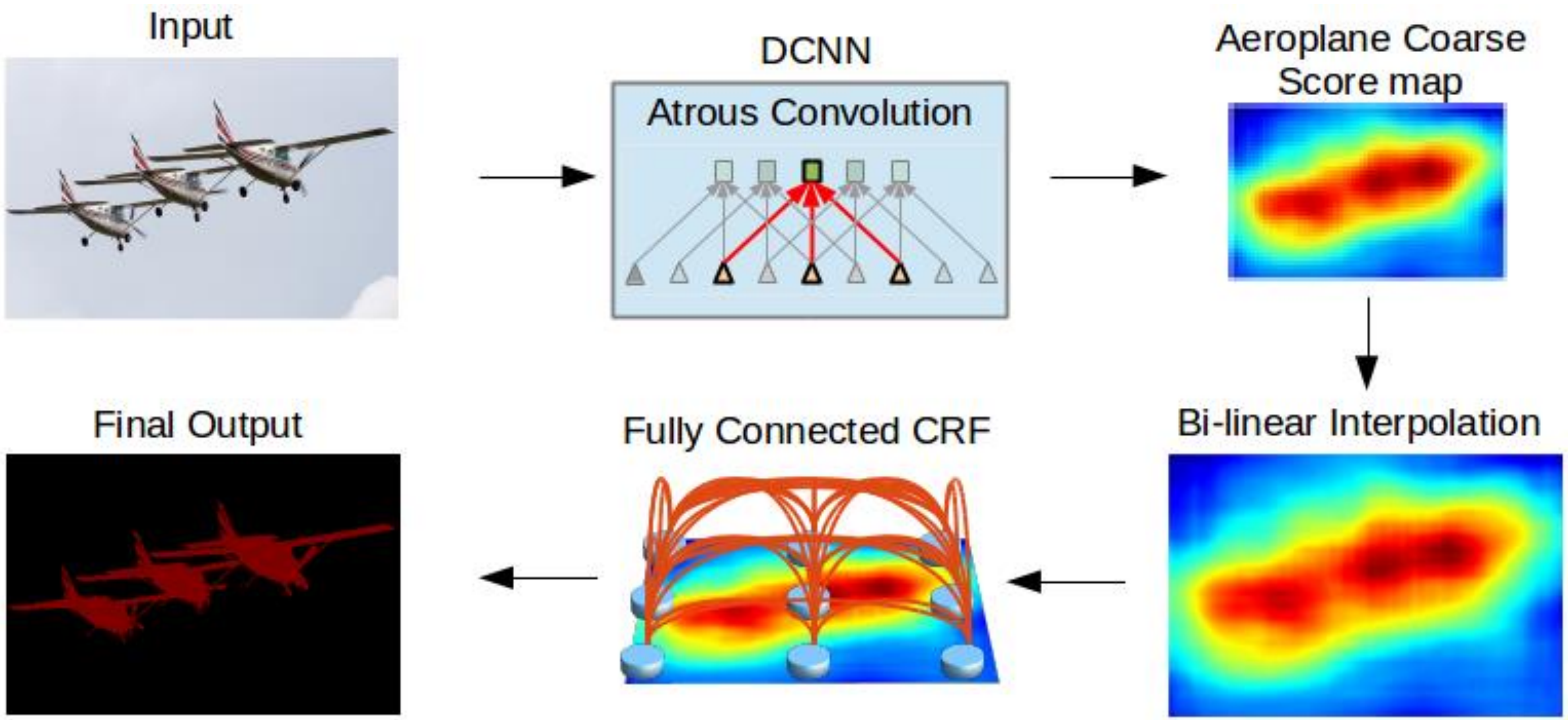
Help
context + semantics

U-Net (aka, Hourglass network)



CRF to improve localization

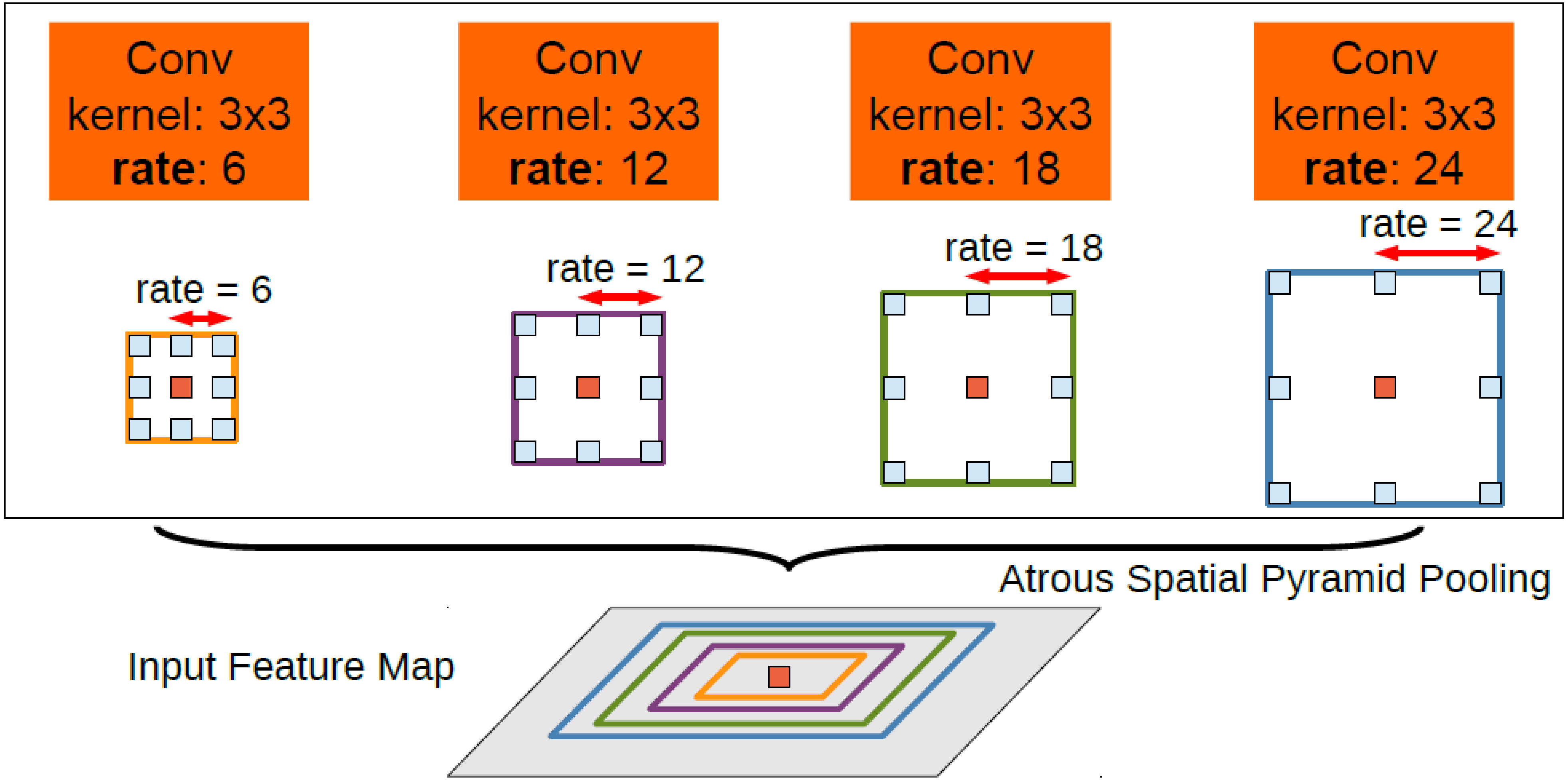
- DeepLab



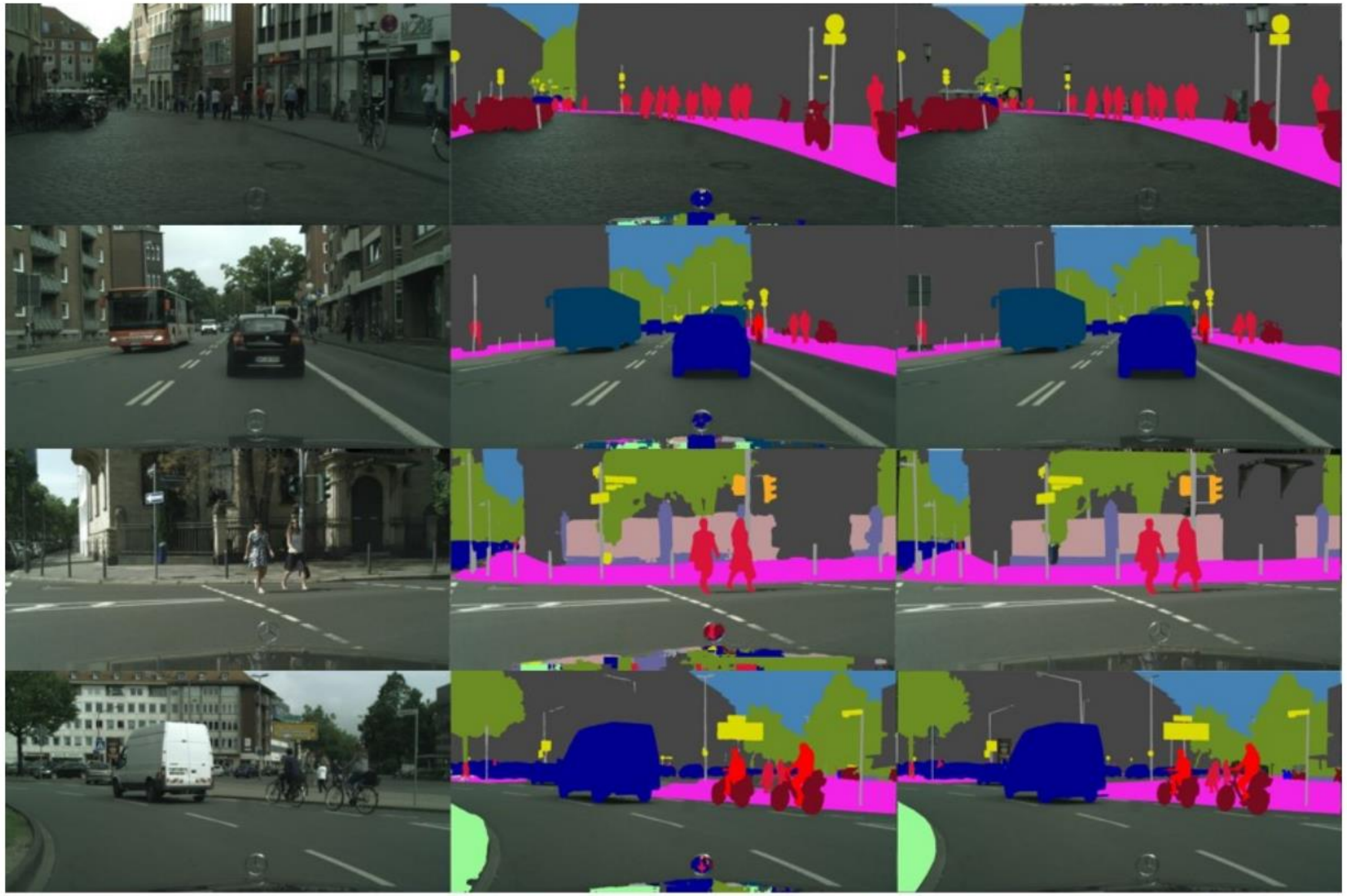
CRF: similar and nearby pixels have the same class label

[Chen et al., DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, PAMI 2017]

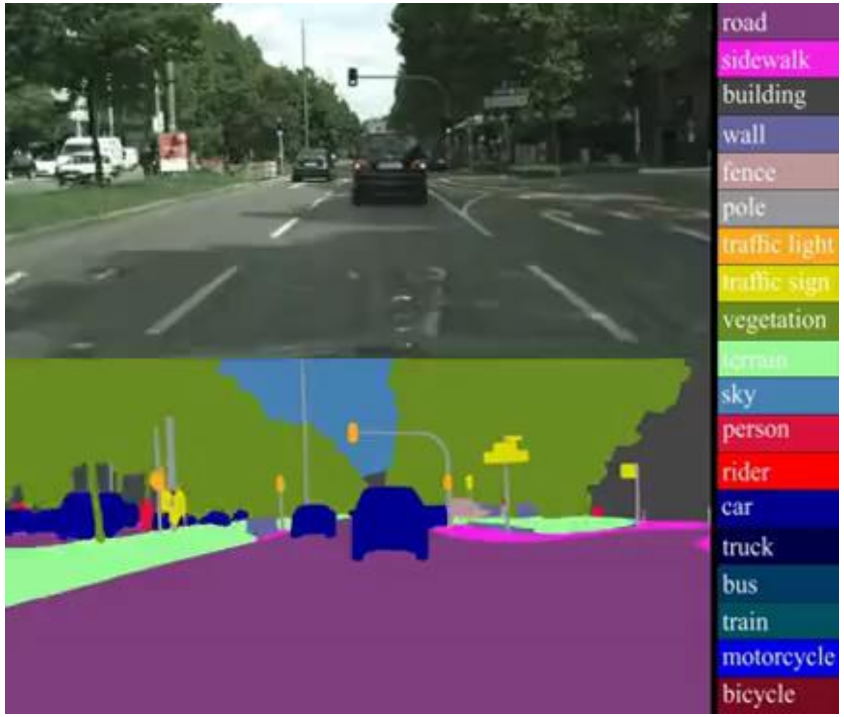
Atrous Spatial Pyramid Pooling (ASPP) for multi-scale features



Example results



[Nirkin et al., HyperSeg, 2021] Ground truth



[Zhao et al., Pyramid scene parsing network, 2017]

Object detection + instance segmentation

Object detection

- **Properties:**

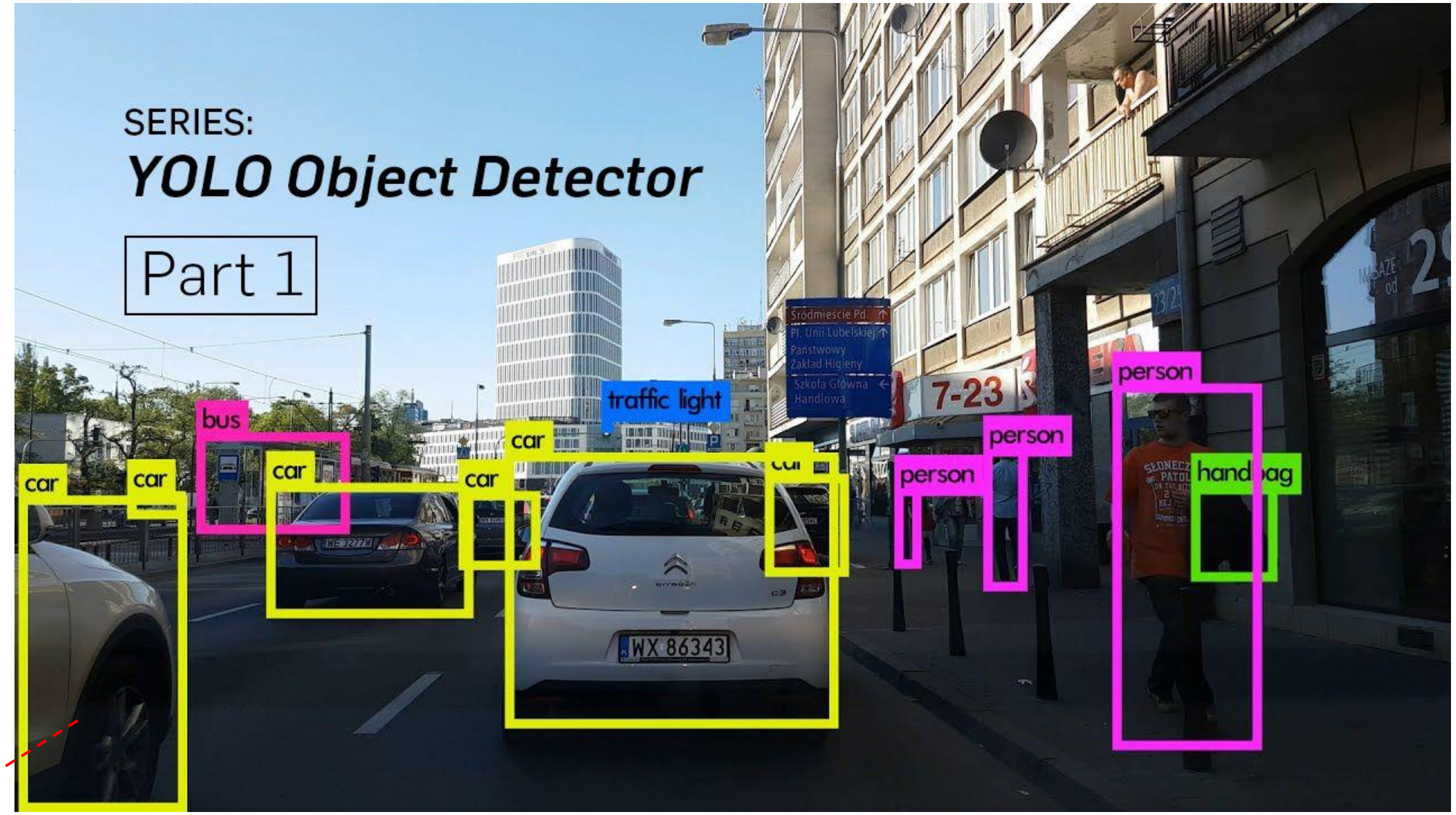
➢ Labels + bounding boxes

➢ Localization

➢ Multi-scale

➢ Context

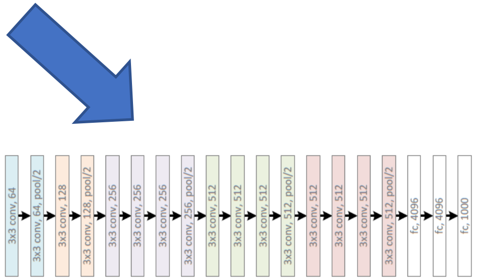
➢ “Undetermined” numbers



[class, u-center, v-center, width, height]

Naïve way

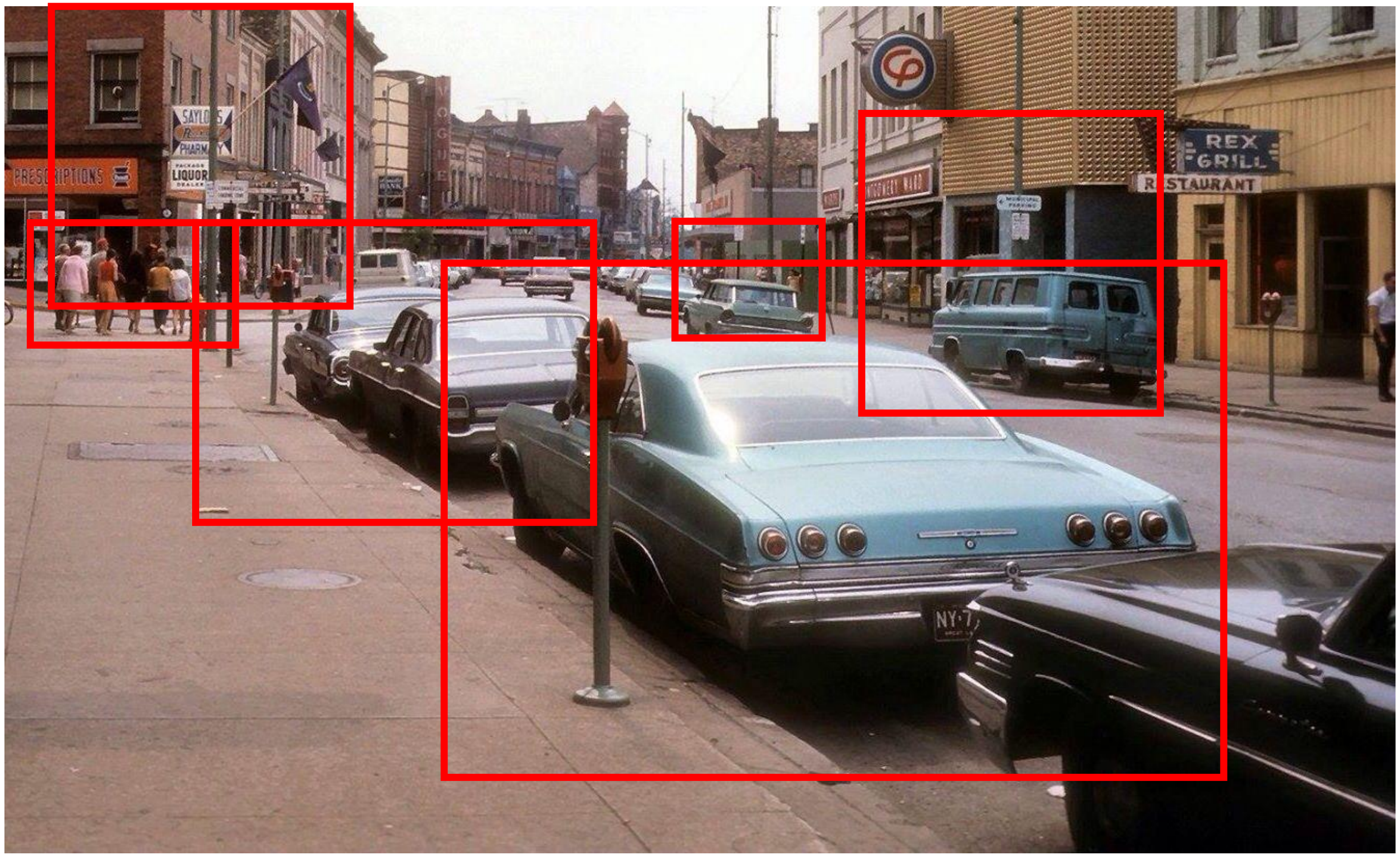
- Sliding window
  - ➢Time consuming
  - ➢What size?



ResNet classifier

R-CNN

- Objectness proposal
- CNN classifier
- Box refinement



[Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014]

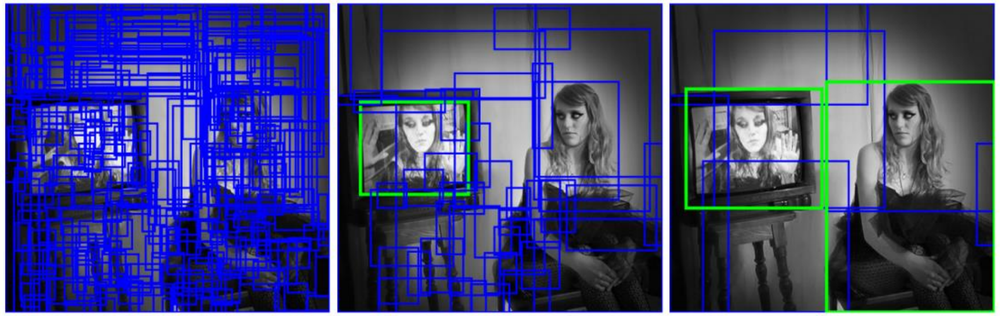
Selective search for proposal generation

- Step 1:
  - Not deep learning
  - **super-pixel-based** segmentation
- Step 2:
  - Recursively combine similar regions into larger ones
- Step 3:
  - Boxes fitting

Input Image

Initial Segmentation

Input Image

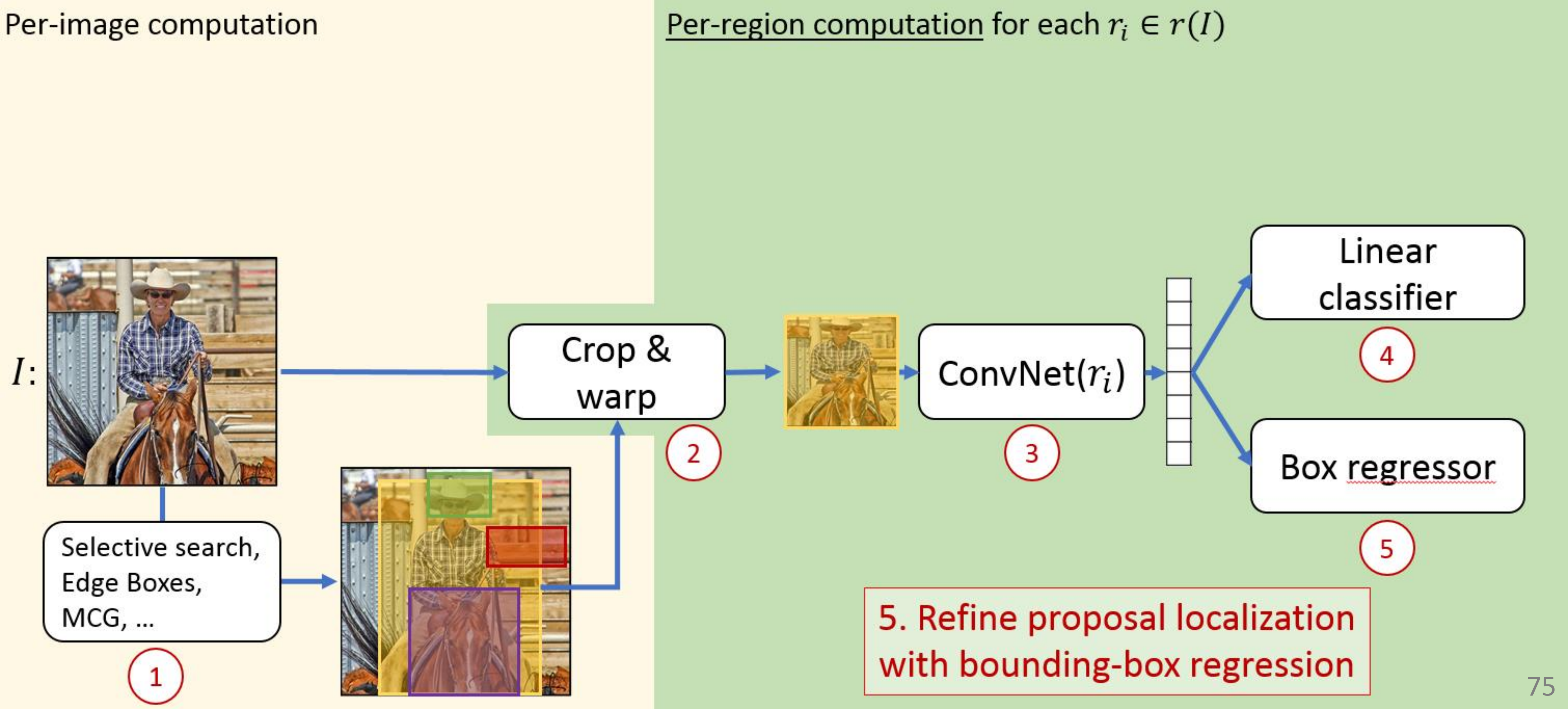


[Stanford CS 231b]

R-CNN

[Girshick et al., Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014]

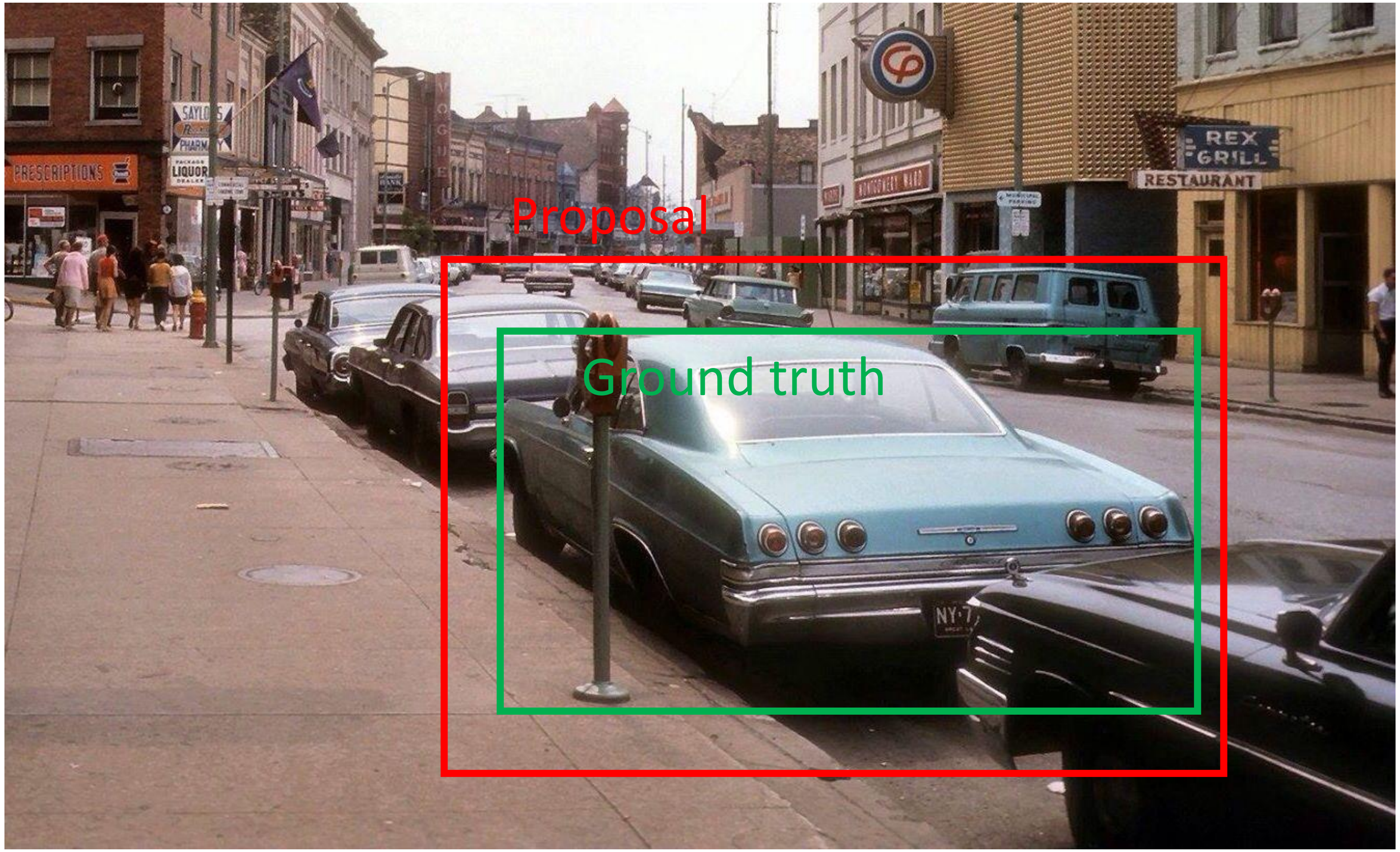
[Girshick, CVPR 2019 tutorial]



R-CNN

- Box regression:
  - ➢(du, dv)
  - ➢(dw, dh)

By offset = MLP(feature)



R-CNN

- Problems:

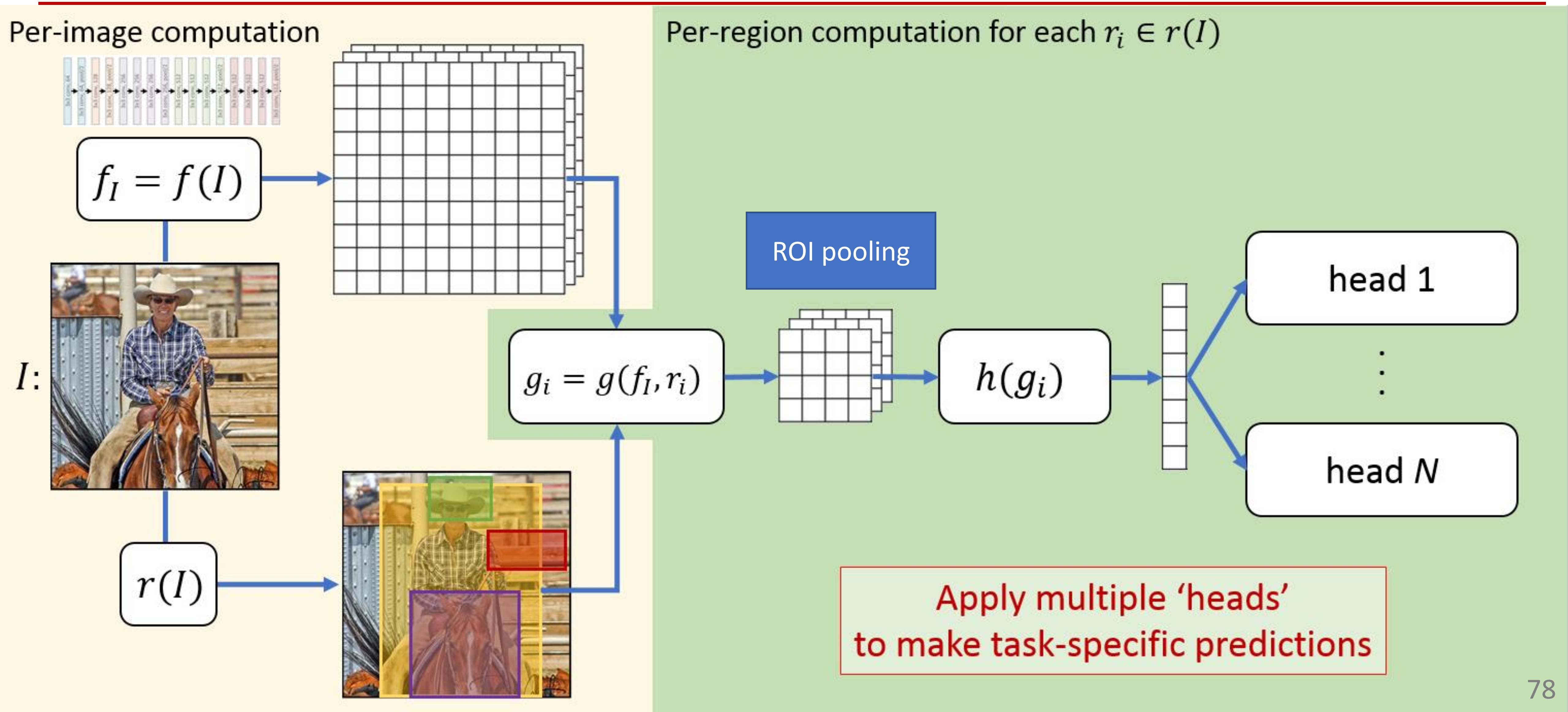
➢ Slow: every proposal needs to go through a “full” CNN

➢ Mis-detection: the proposal algorithm is not trained together

Fast R-CNN

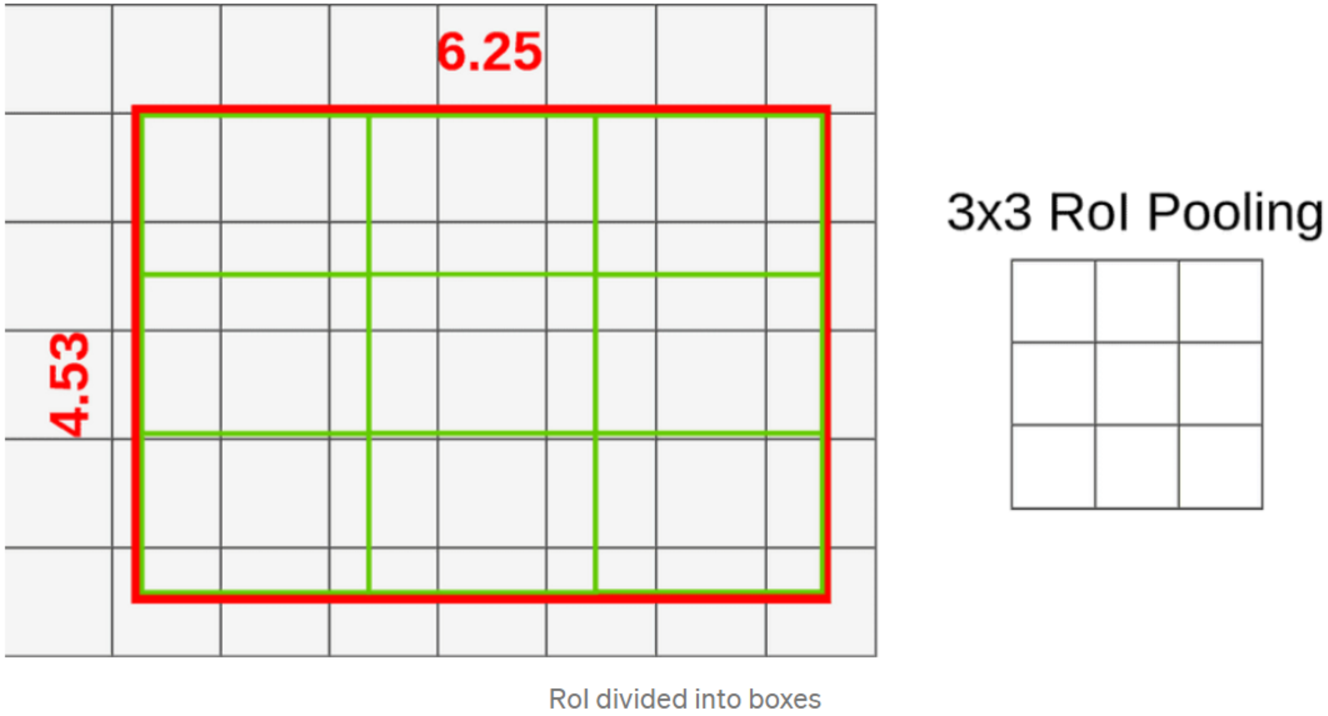
[Girshick, Fast R-CNN, ICCV 2015]

[Girshick, CVPR 2019 tutorial]

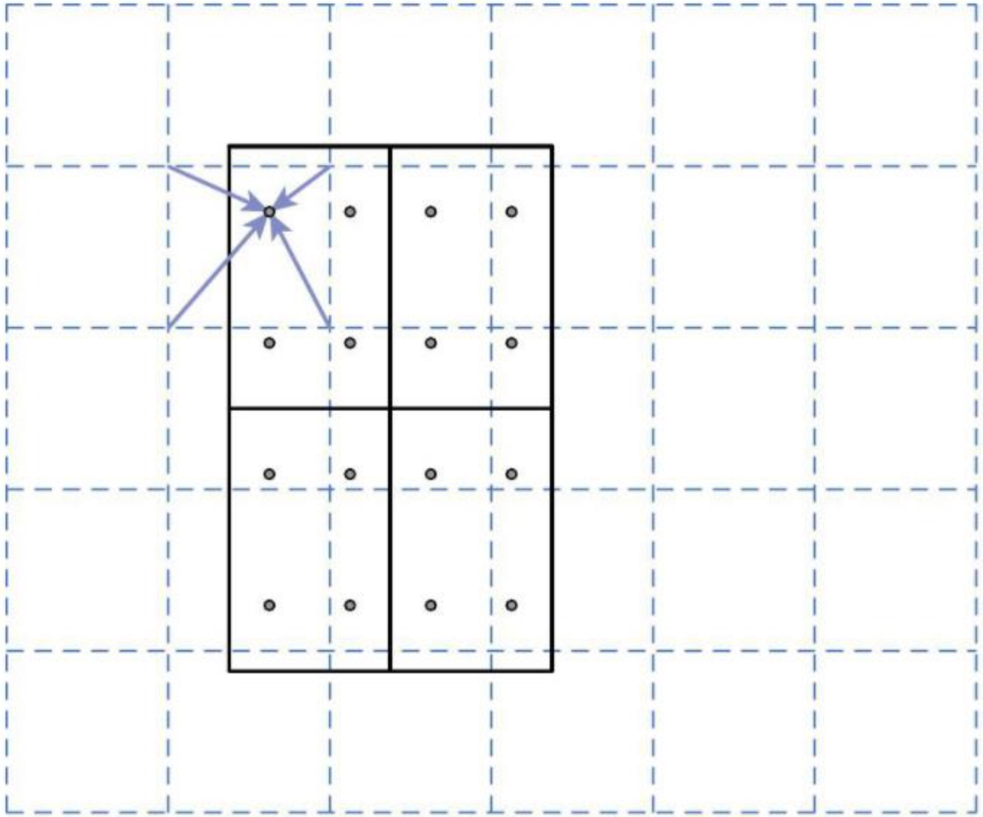


ROI pooling vs. ROI align

Making features extracted from different proposals the same size!



ROI Pooling

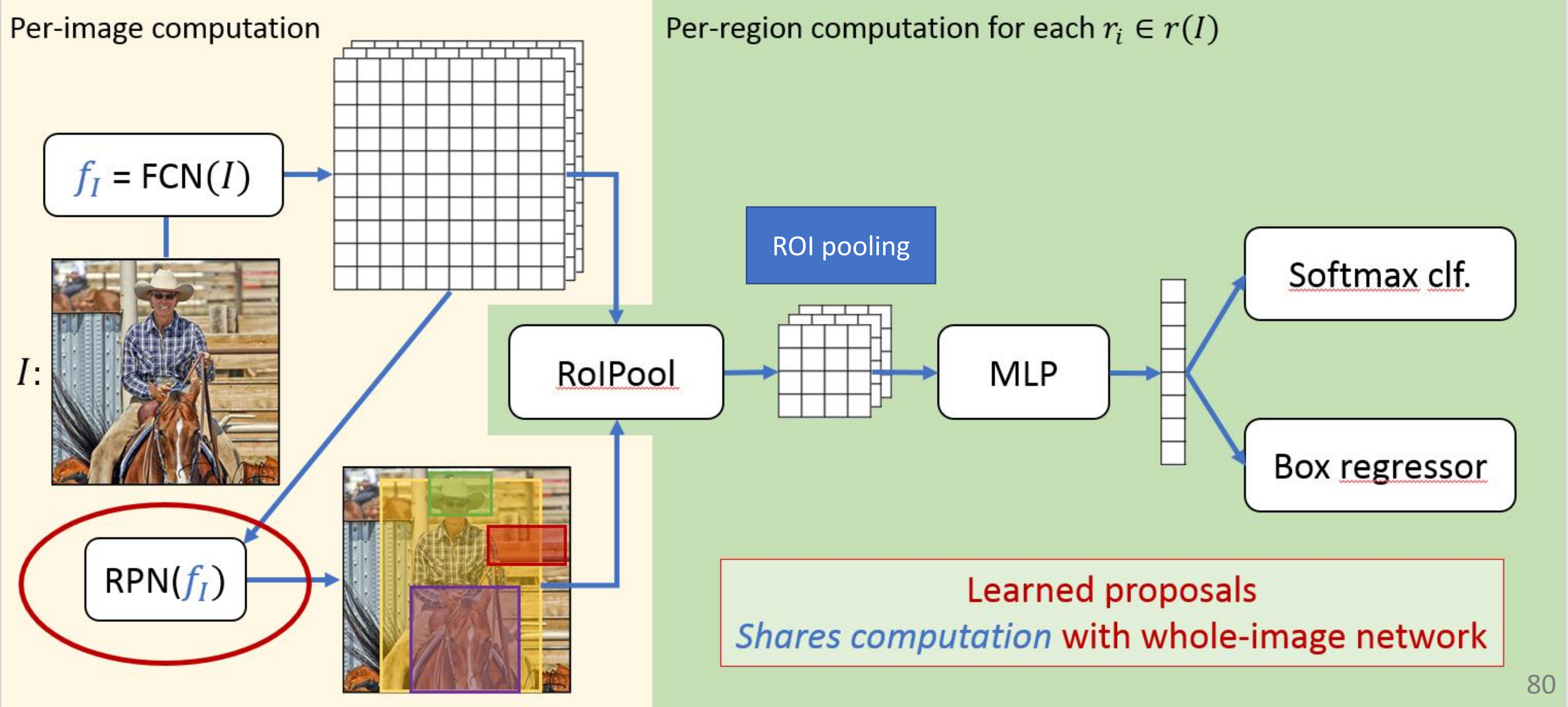


ROI Align

Faster R-CNN

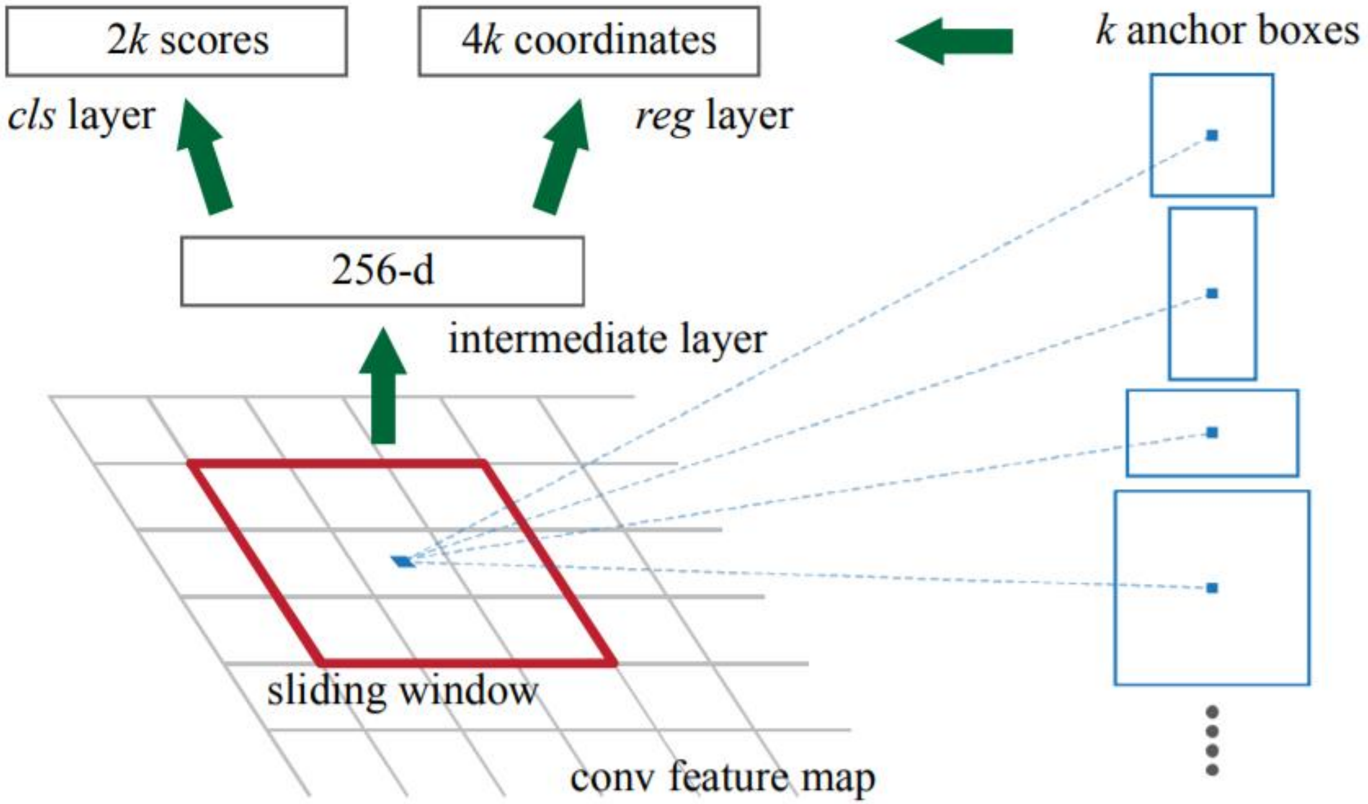
[Ren et al., Faster r-cnn: Towards real-time object detection with region proposal networks, NIPS 2015]

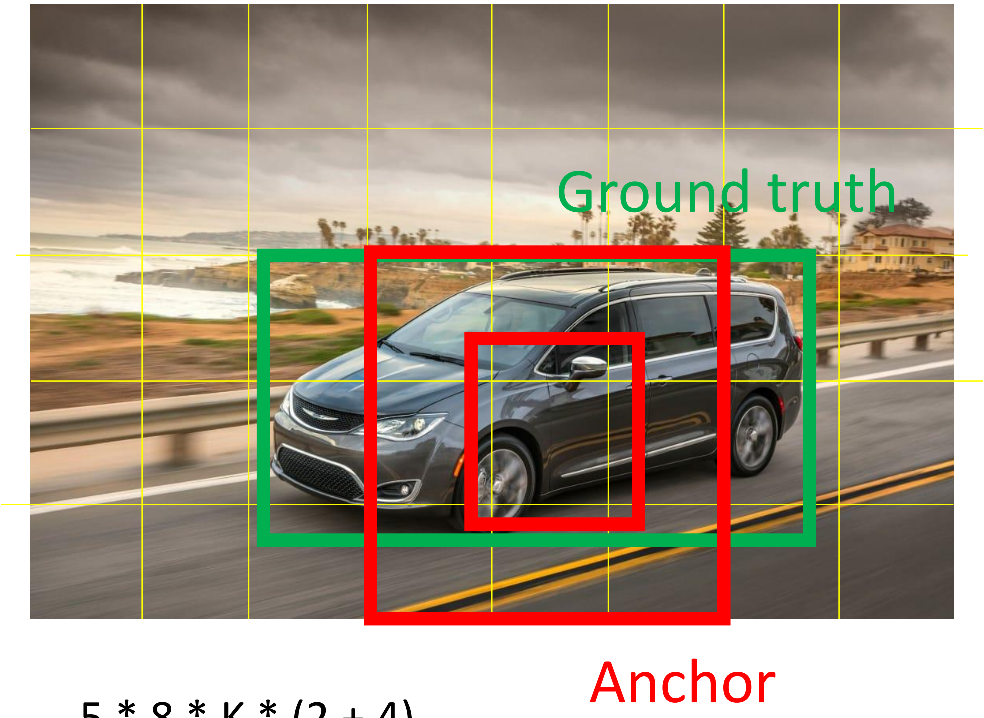
[Girshick, CVPR 2019 tutorial]



How to develop RPN (region proposal network)?

[Ren et al., 2015]



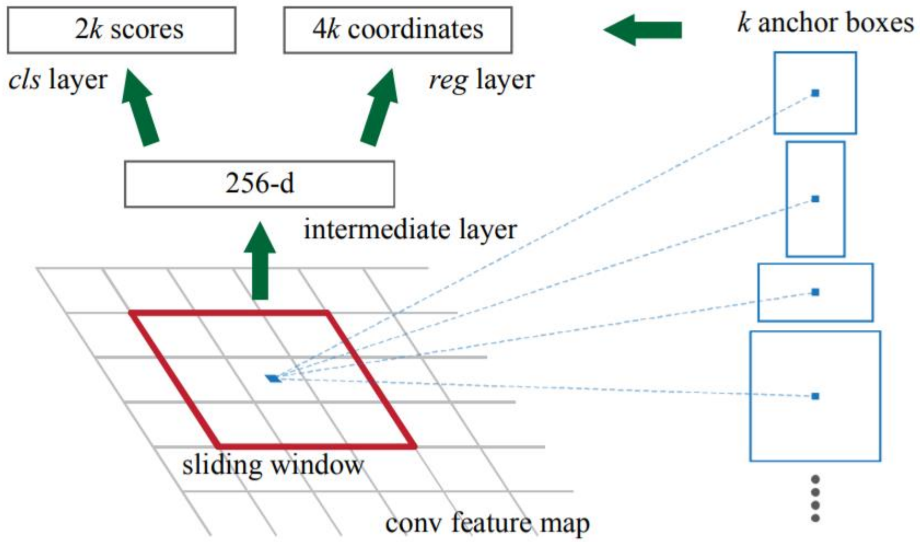


5 * 8 * K * (2 + 4)

What do we learn from RPN?

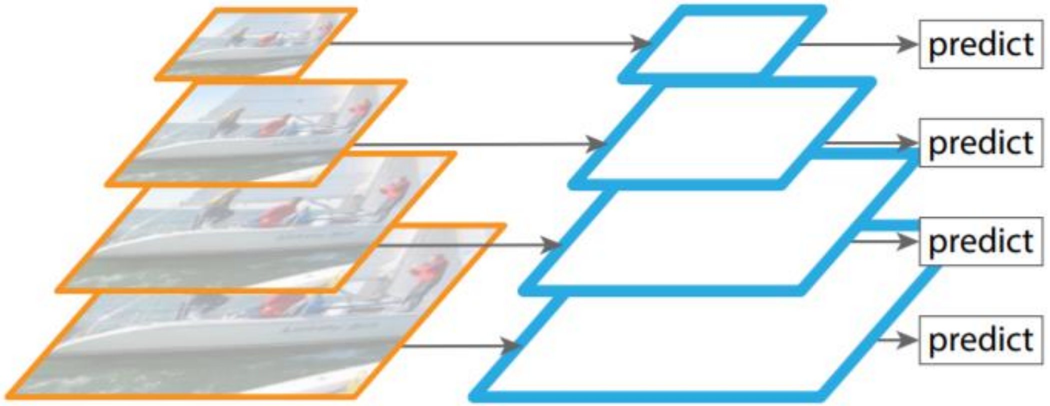
- *"How to encode your labeled data so that your CNN can learn from them"* is important!

- Inference: predict these "values" and accordingly transfer them to bounding boxes!

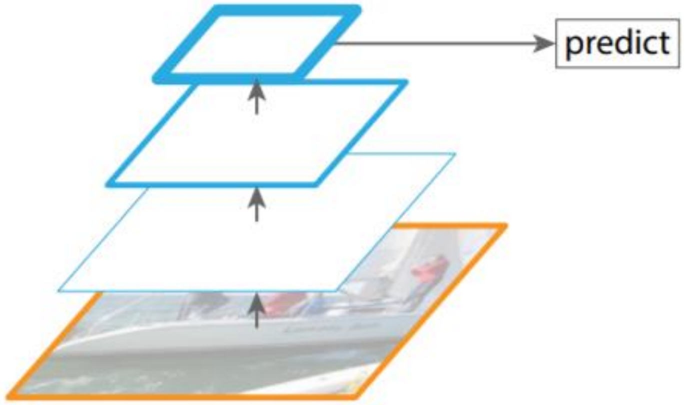


Questions?

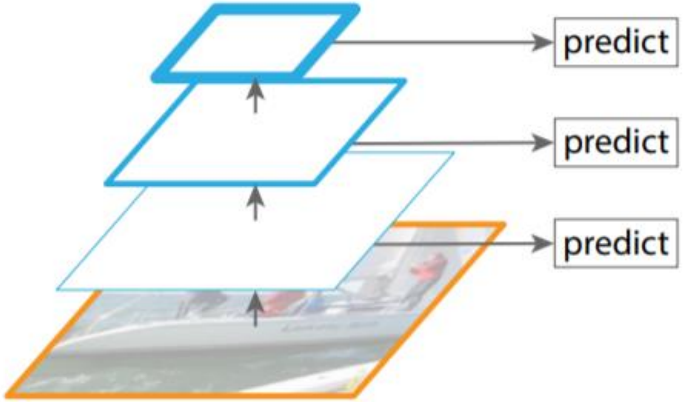
How to deal with object sizes?



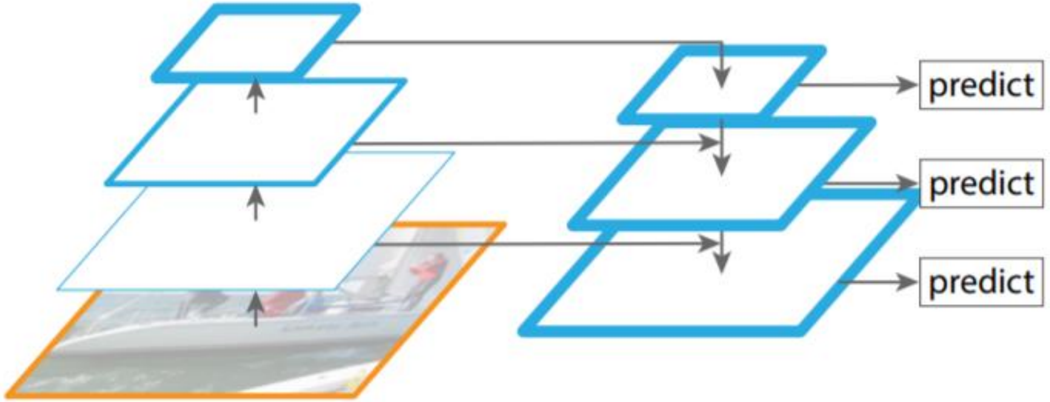
(a) Featurized image pyramid



(b) Single feature map



(c) Pyramidal feature hierarchy

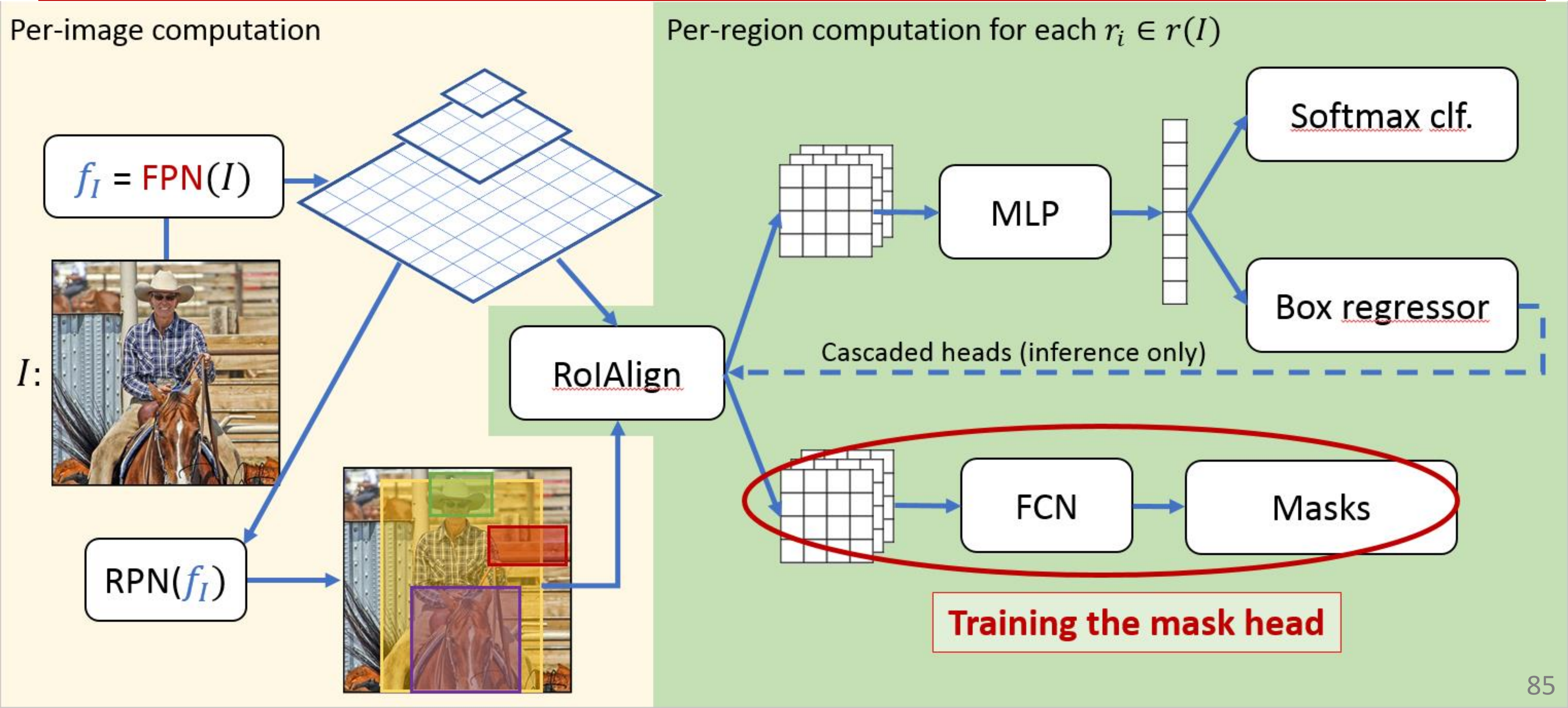


(d) Feature Pyramid Network

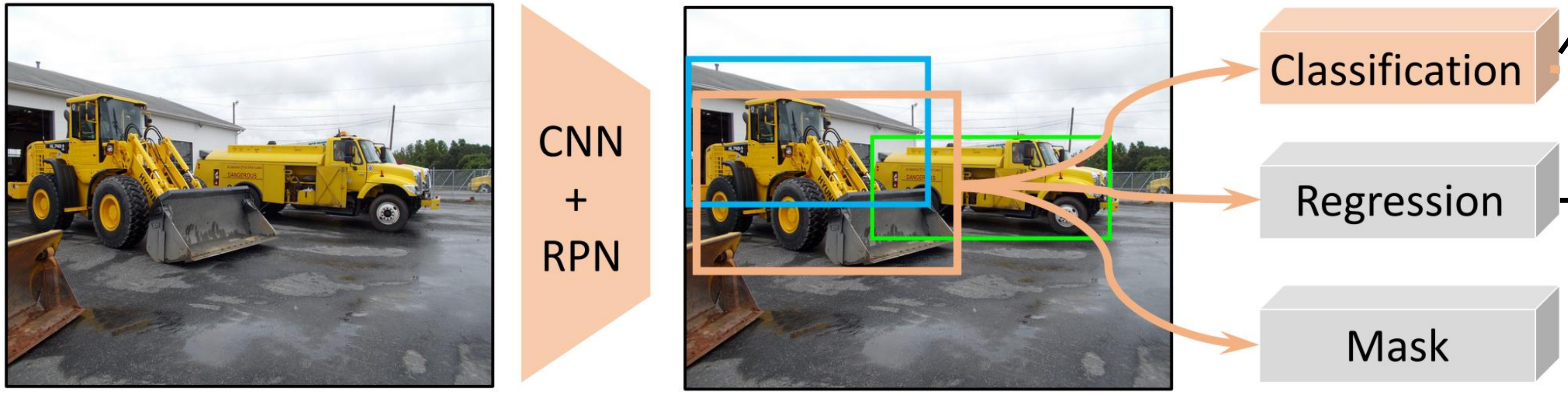
[Lin et al., Feature Pyramid Networks for Object Detection, CVPR 2017]

Mask R-CNN

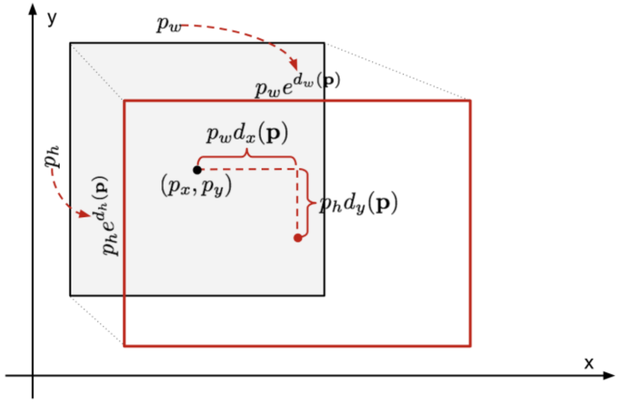
[He et al., Mask r-cnn, ICCV 2017] [Girshick, CVPR 2019 tutorial]



Mask R-CNN: for instance segmentation



Bulldozer: 80%
Bus: 15%
Motorcycle: 5%

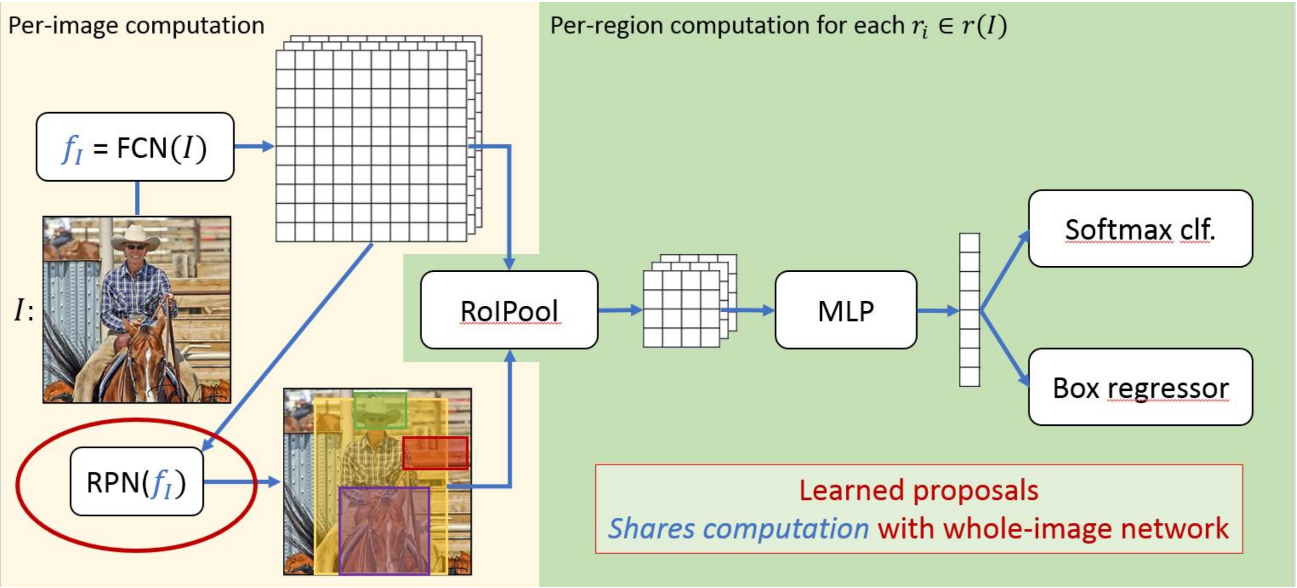


CNN: convolutional neural network
RPN: region proposal network

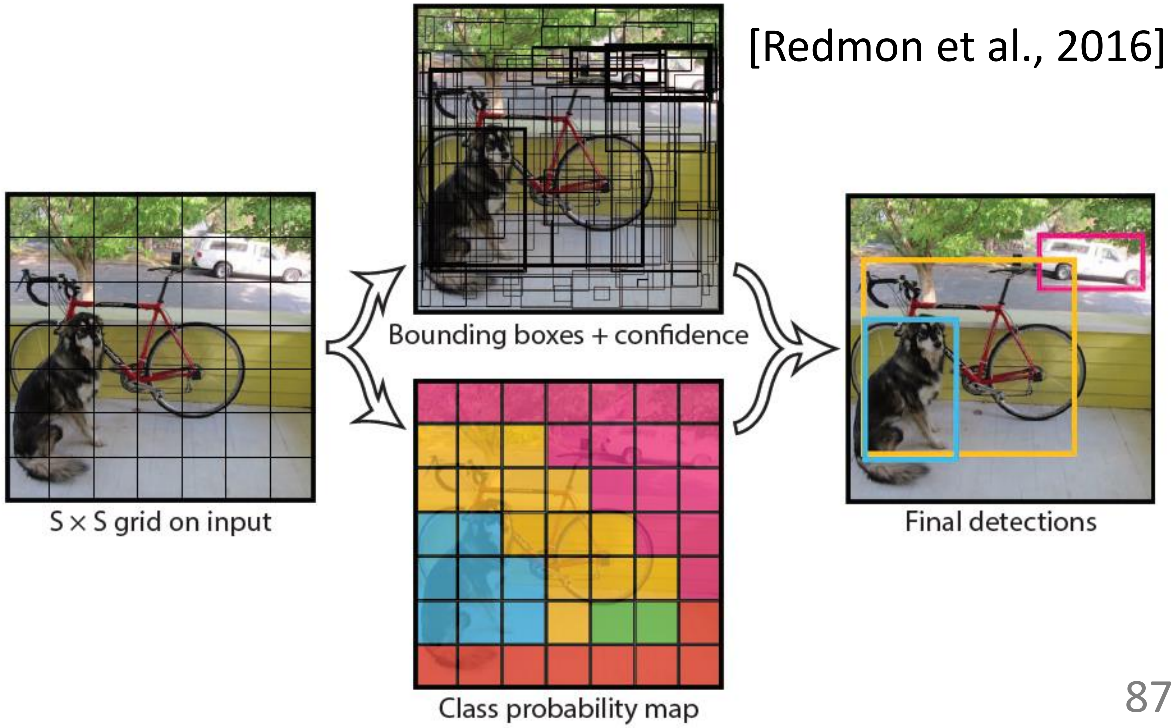
2-stage vs. 1-stage detectors

- Other names: single-shot, single-pass, … (e.g., YOLO, SSD)
- Difference: no ROI pooling/align

2-stage detector



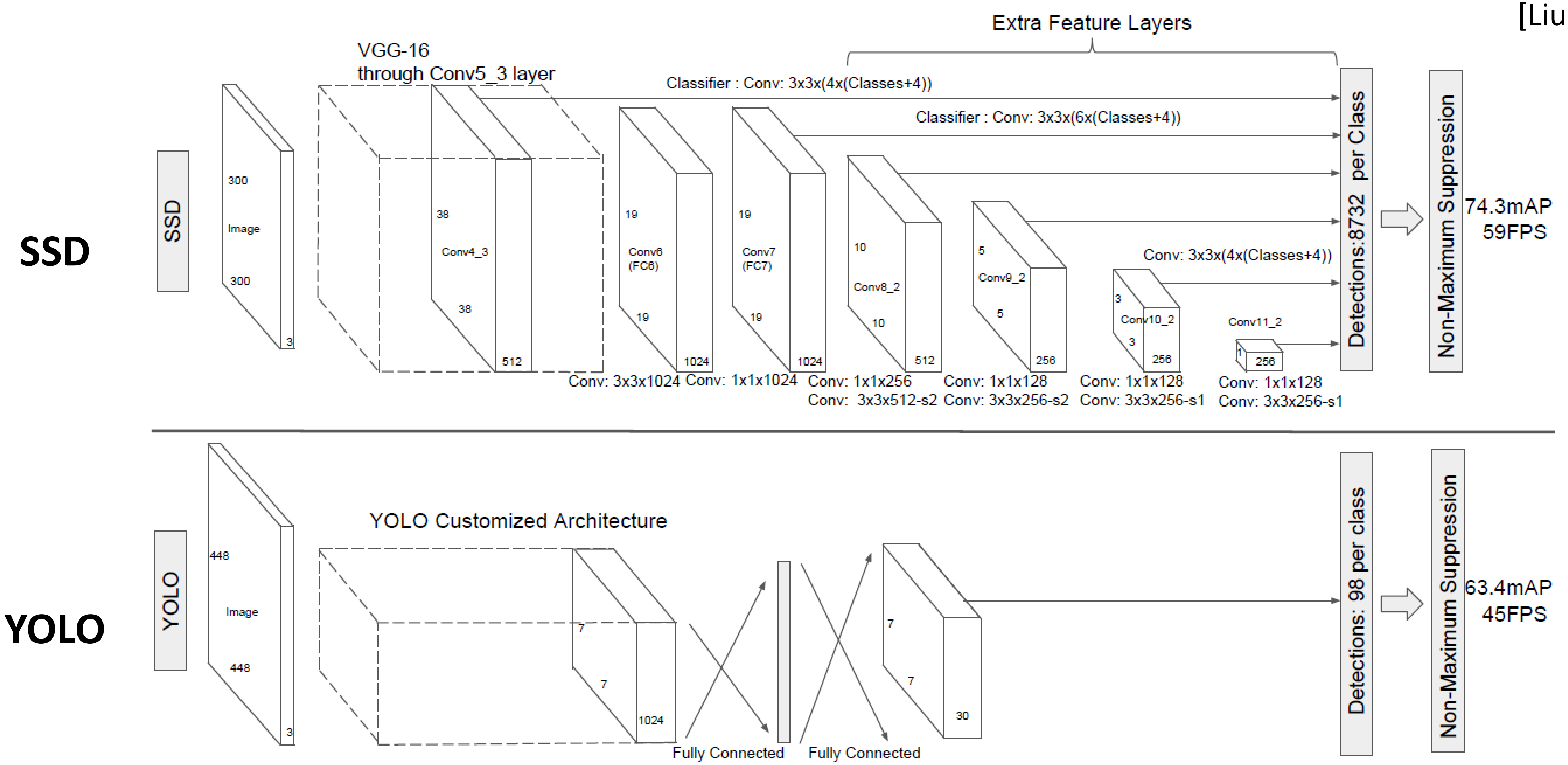
1-stage detector



[Redmon et al., 2016]

Exemplar 1-stage detectors

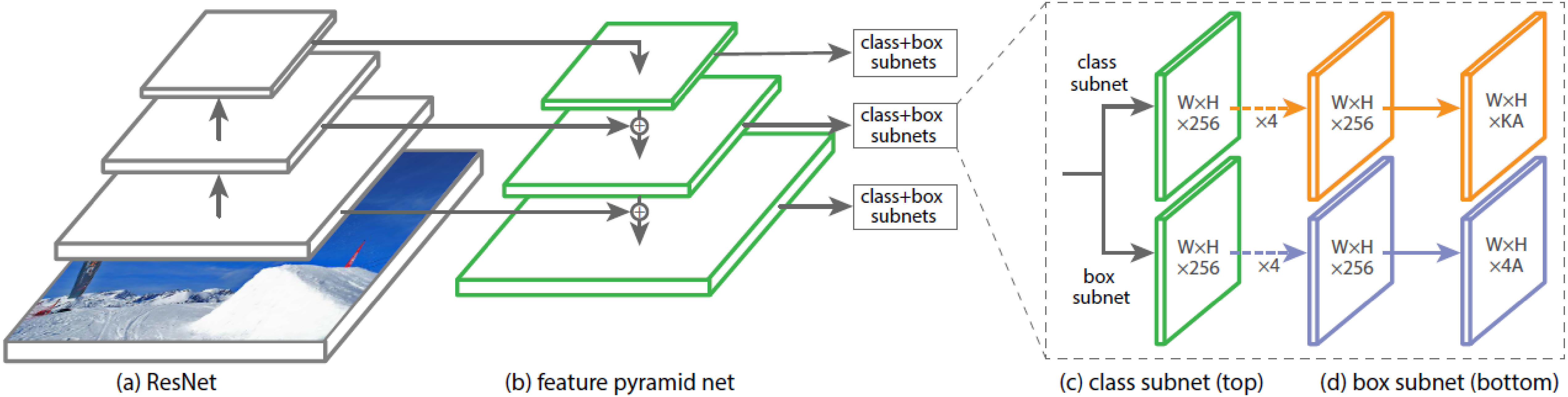
[Liu et al., 2016]



[Redmon et al., 2016]

Exemplar 1-stage detectors (Retina Net)

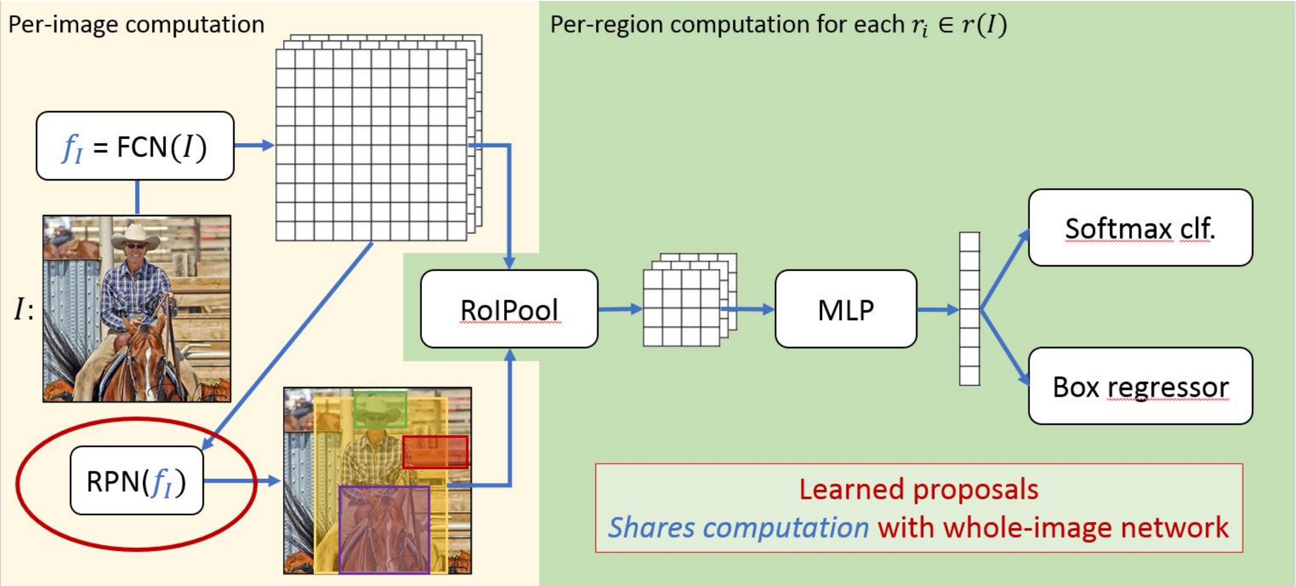
[Lin et al., 2017]



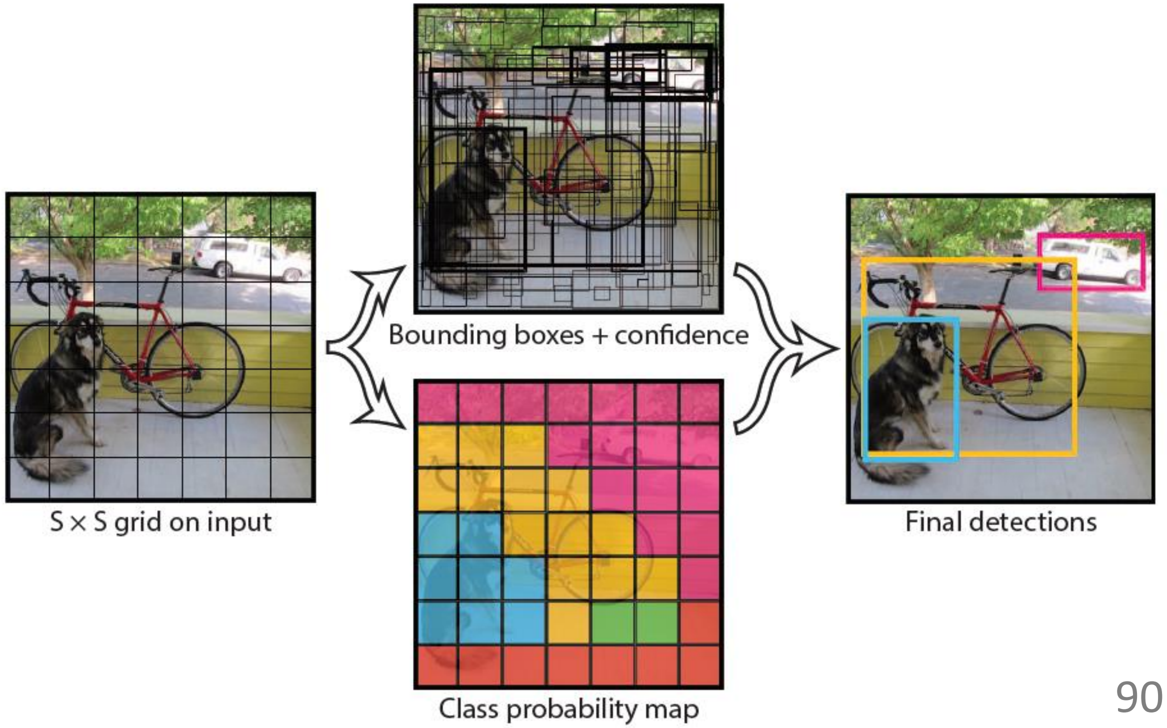
2-stage vs. 1-stage detectors

- Pros for 1-stage:
  - Faster!

- Cons for 1-stage:
  - Too many negative locations; scale



[Redmon et al., 2016]

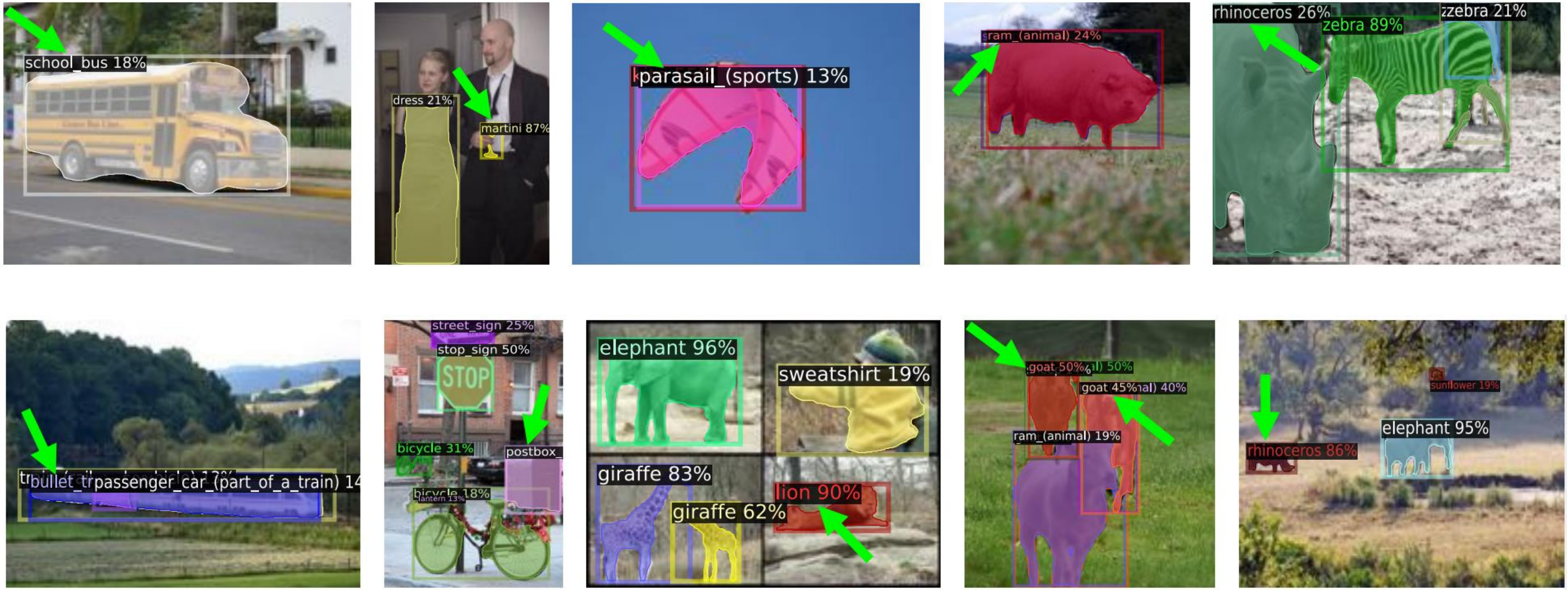


Inference: choose few from many

- Non-Maximum Suppression (NMS)

[Pictures from “towards data science” post]

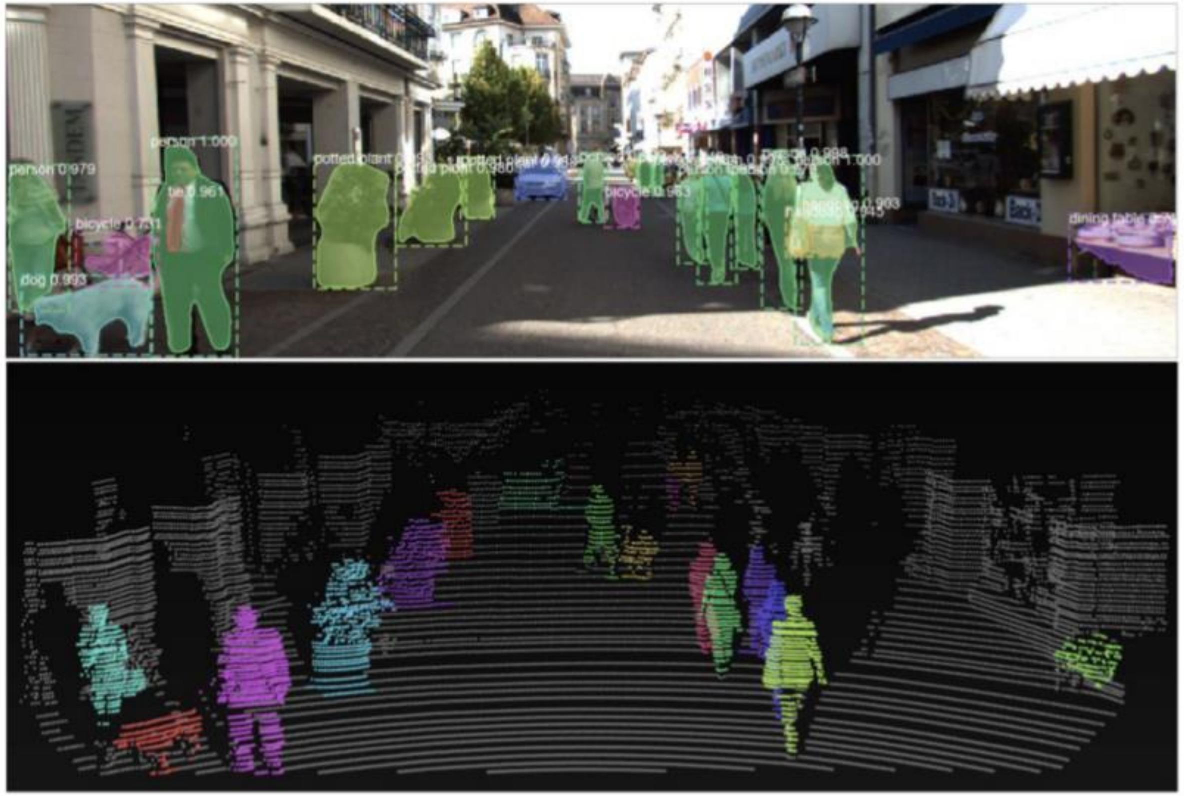
Example results



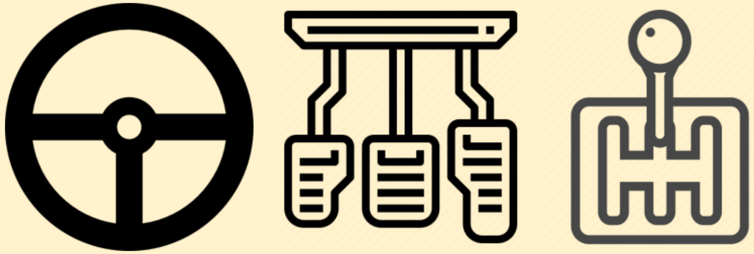
[Zhang, et al., 2021]

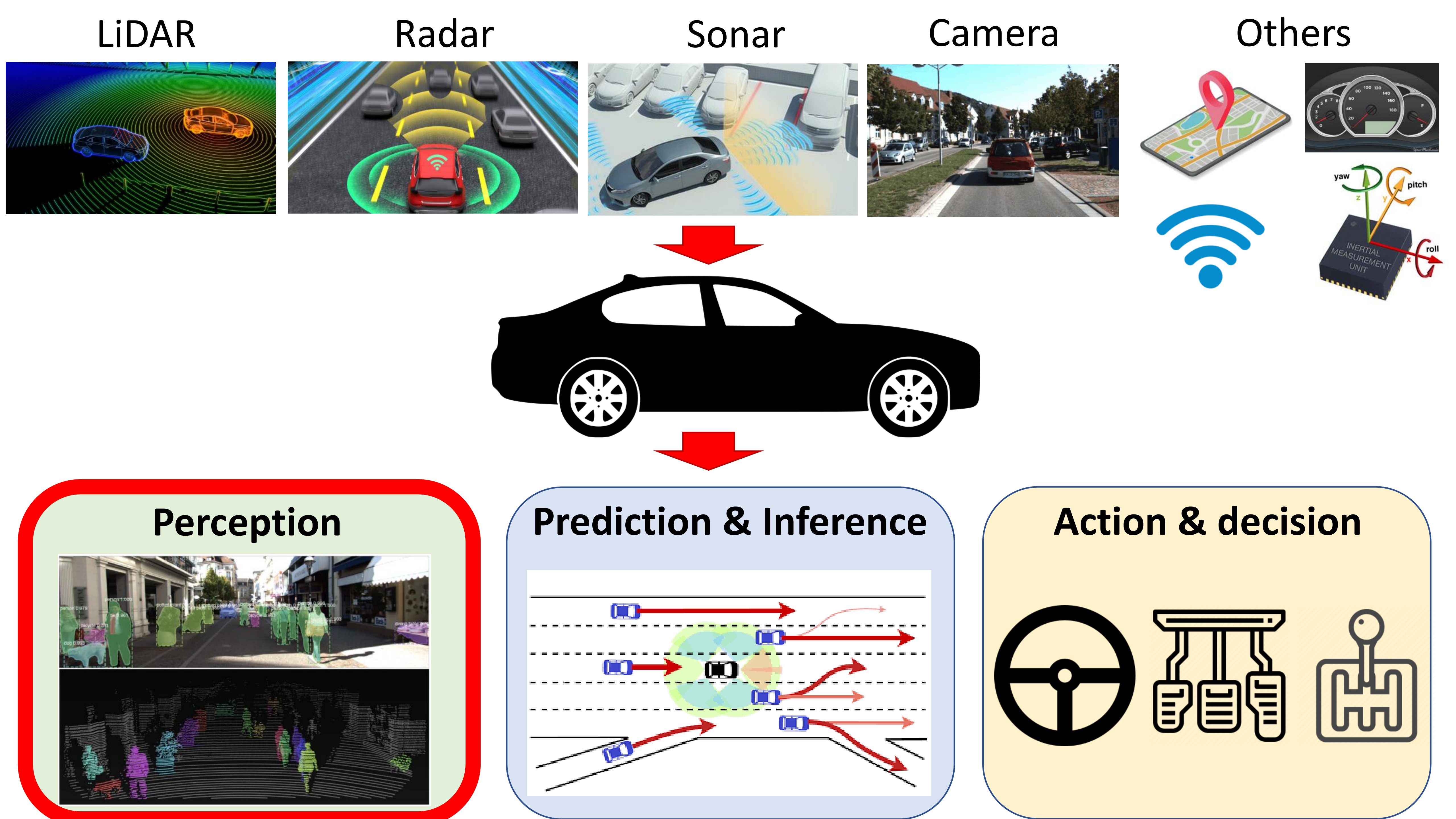
Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts

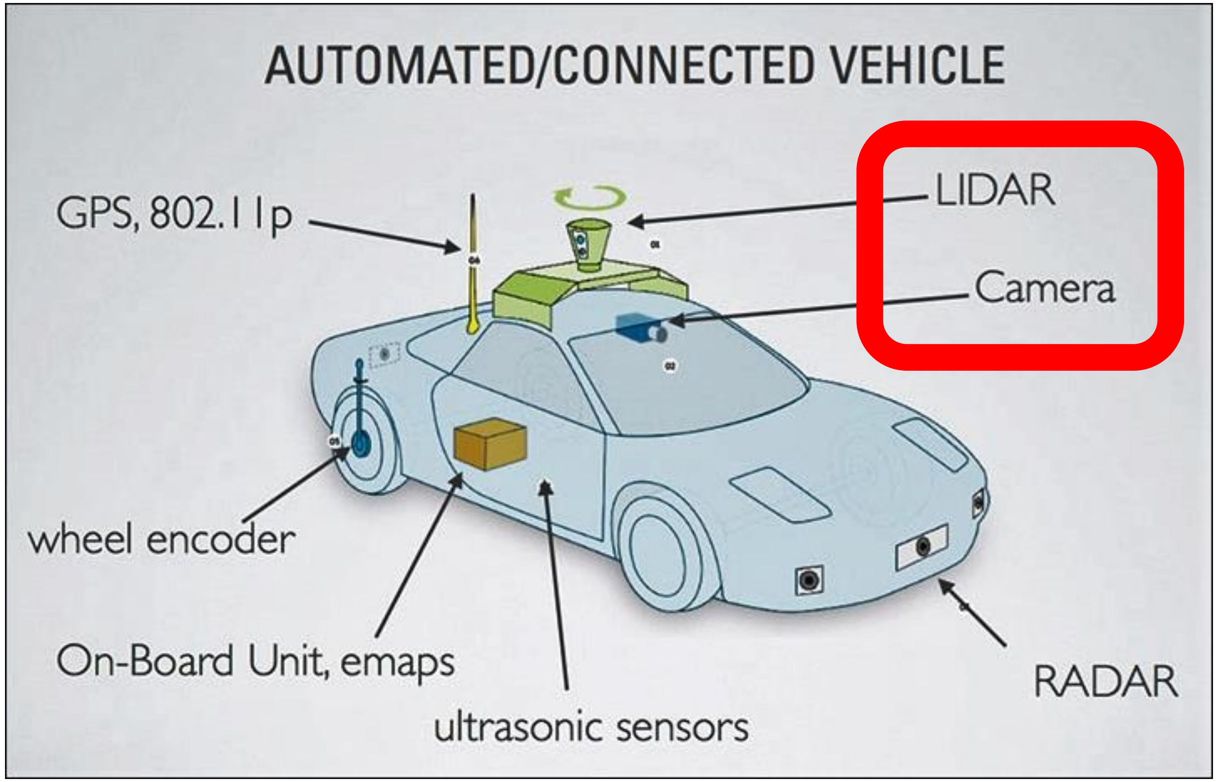


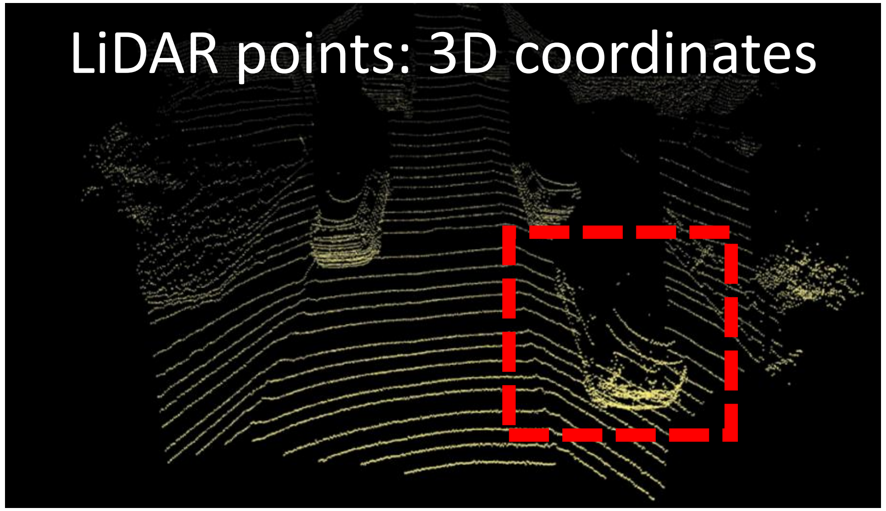
Action & decision



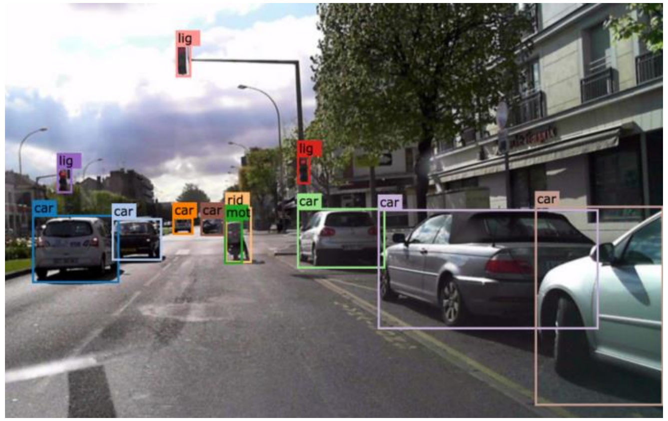


LiDAR
Radar
Sonar
Camera
Others
yaw
pitch
roll
INERTIAL MEASUREMENT UNIT
Perception
Prediction & Inference
Action & decision

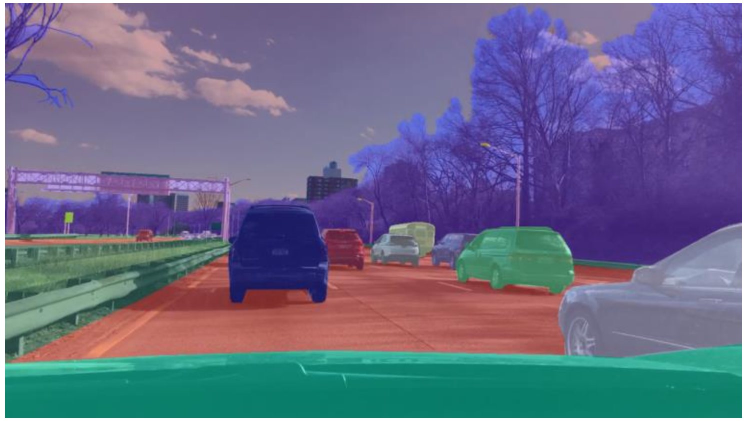




2D object detection

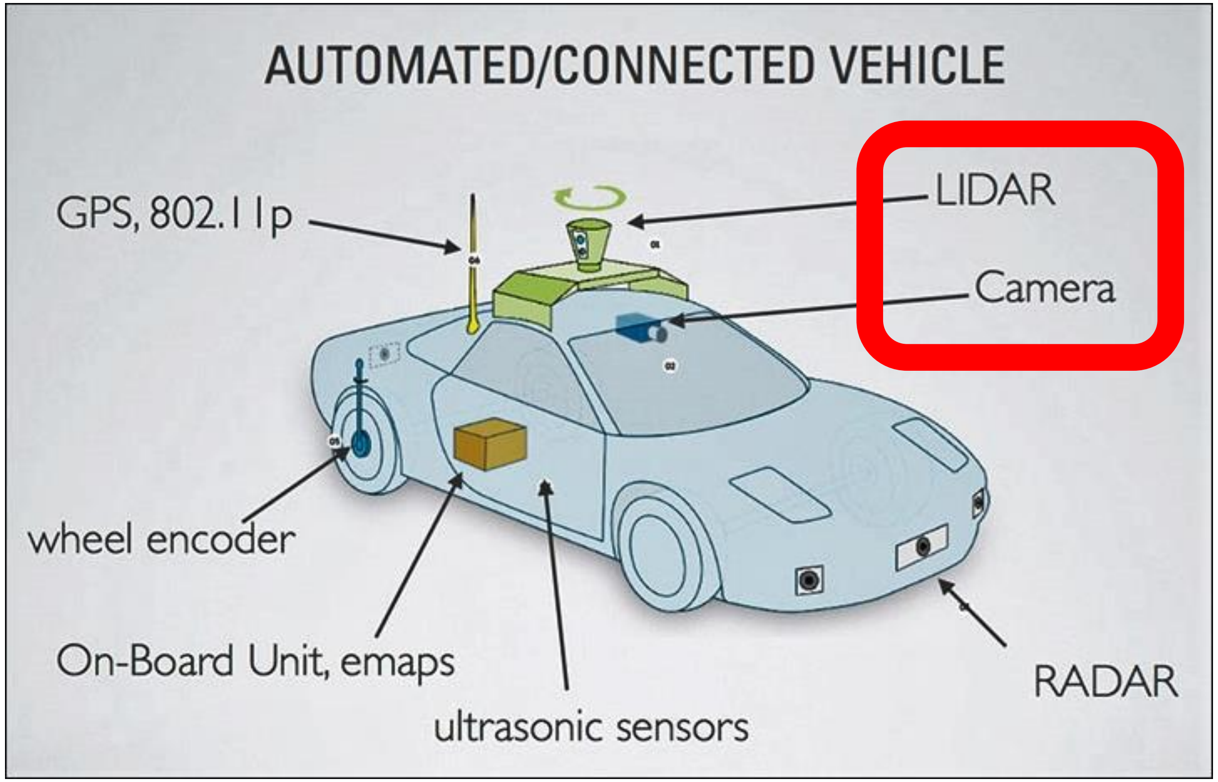


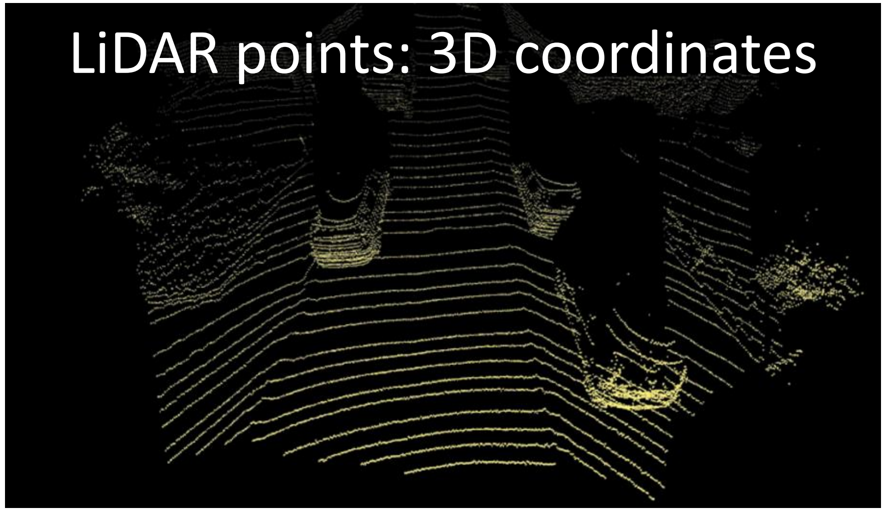
2D instance segmentation



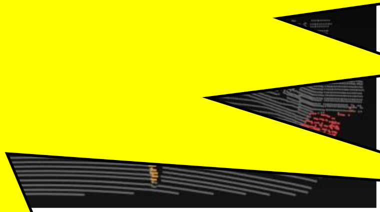
Long-standing computer vision tasks

[Source: BDD100K dataset, CVPR 2016]





3D object detectio… 3D … segmentation



How “far” are the detected objects?

Identify ob… 3D locations in (x, y, z), not 2D locations in image pixels!

LiDAR-based 3D perception

LiDAR-based 3D perception

[Source: Graham Murdoch/Popular Science]

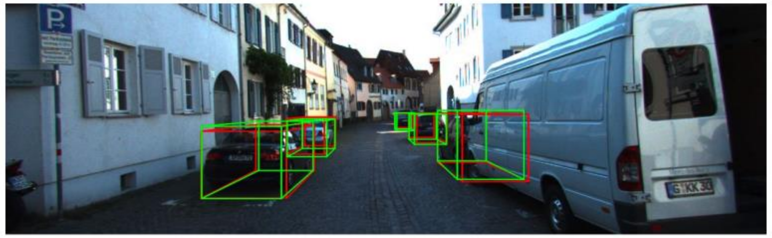
LiDAR:

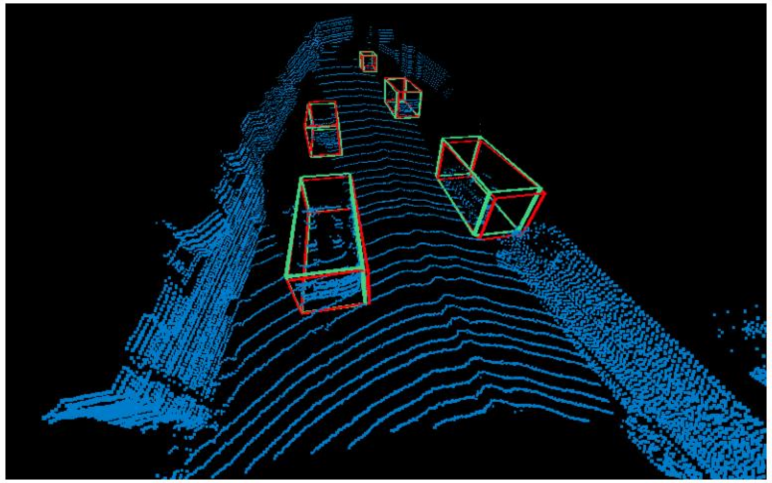
- Light Detection and Ranging sensor
- accurate 3D point clouds of the environment

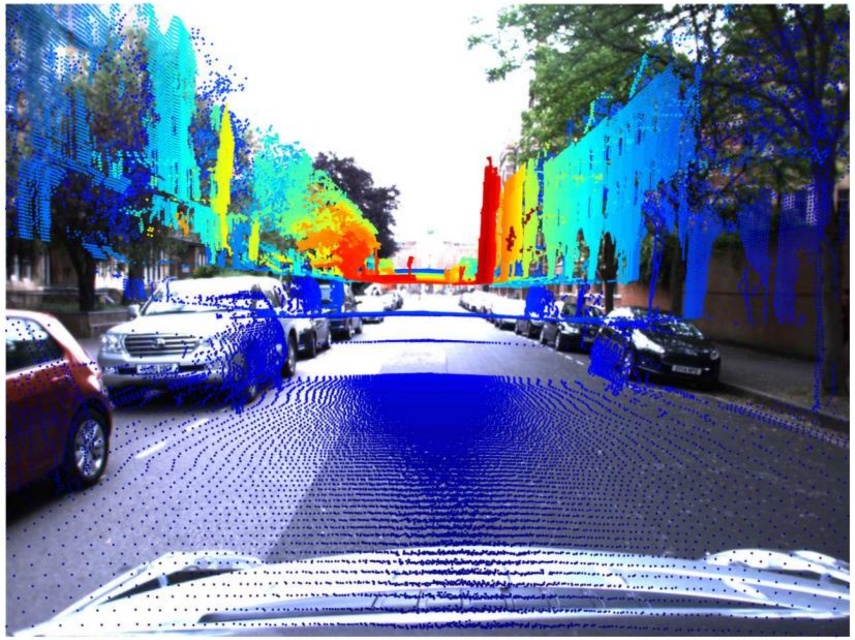
LiDAR-based 3D perception

You can view the LiDAR point clouds from different angles

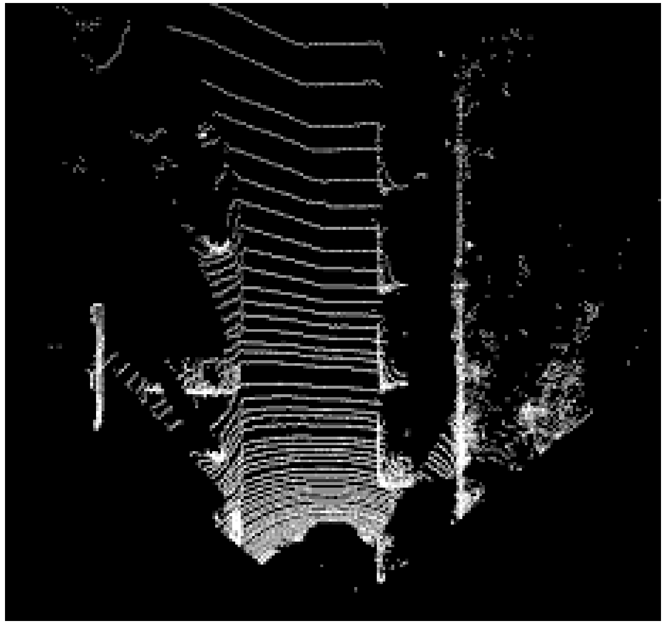
Frontal view





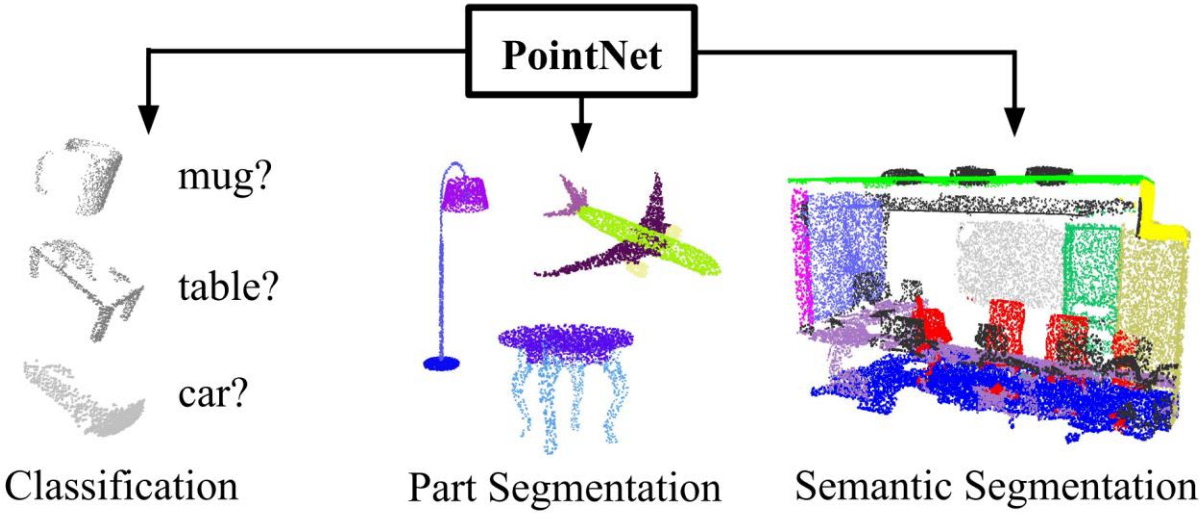


Bird's-eye view (BEV)

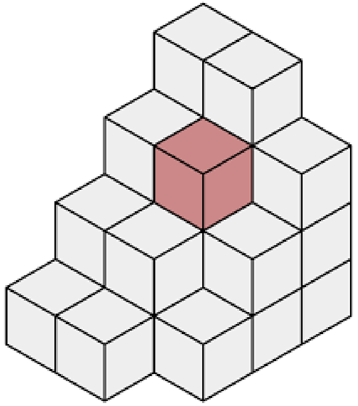


Two major ways to process LiDAR point clouds

- **Point-wise** processing
  - PointNet [Qi et al., 2017]
  - PointNet++ [Qi et al., 2017]
  - PointRCNN [Shi et al., 2019]
  - …

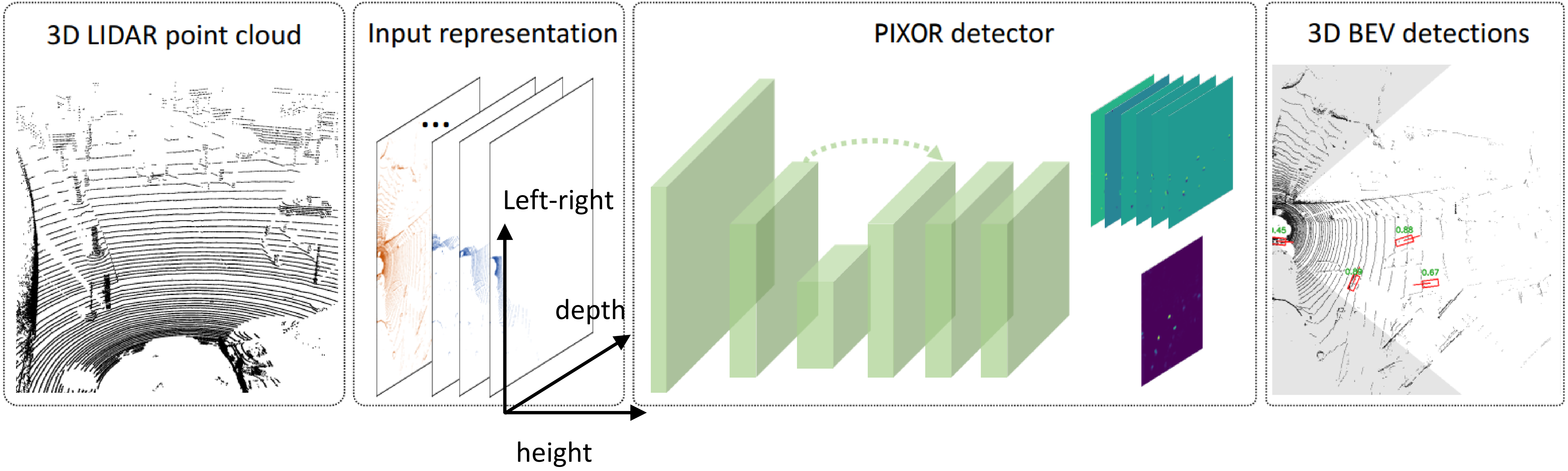


- **Voxel-based** processing: turn points into a tensor (e.g., W x D x H x F)
  - PointPillars [Lang et al., 2019]
  - VoxelNet [Zhou et al., 2017]
  - PIXOR [Liang et al., 2018]
  - …



Voxel-based processing + 3D object detectors

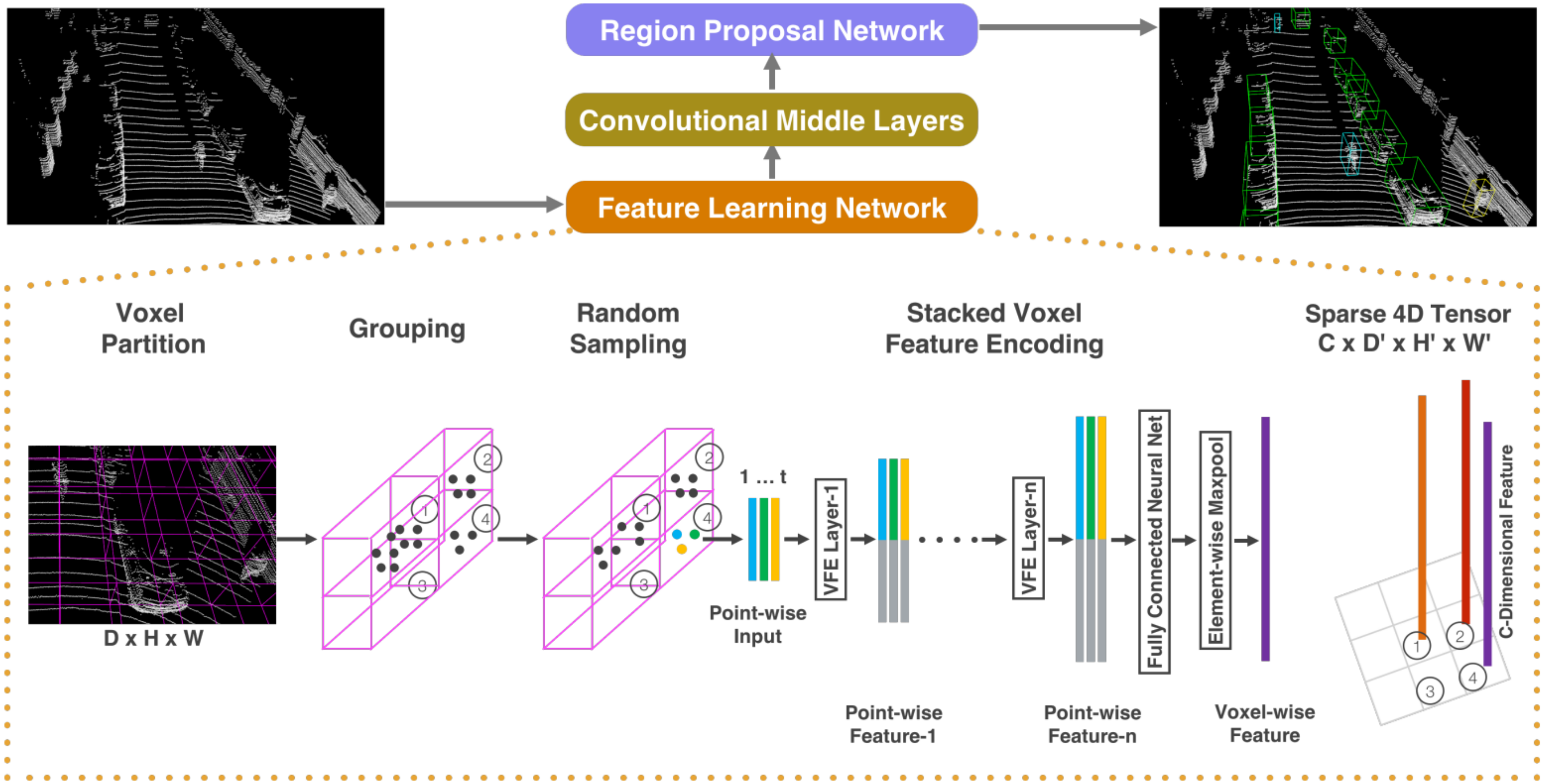
- Occupation (PIXOR): 3D points as a 3D occupation tensor from bird's-eye-view



[Yang et al., PIXOR: Real-time 3D Object Detection from Point Clouds, 2019]

Voxel-based processing + 3D object detectors

- VoxelNet (4D tensors with voxel grid feature: W x D x H x F)

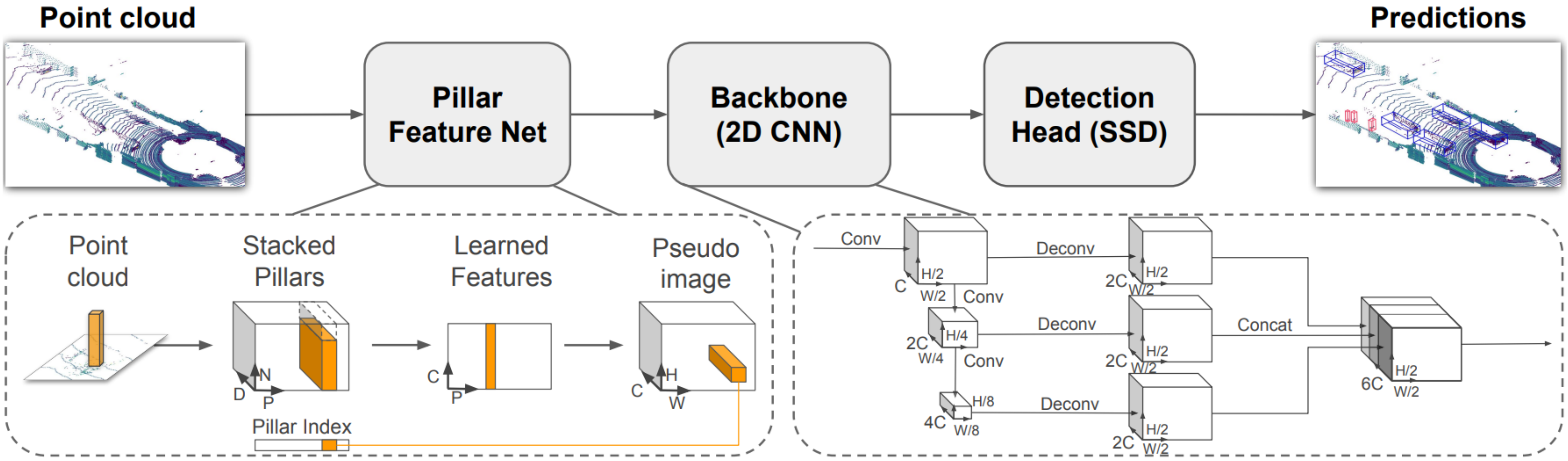


More complicated 4D tensors with voxel grid feature: W x D x H x F

[Zhou et al., VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, 2017]

Voxel-based processing + 3D object detectors

- PointPillars (3D tensors with voxel grid feature: W x D x (H x F))



[Lang et al., PointPillars: Fast Encoders for Object Detection from Point Clouds, 2019]

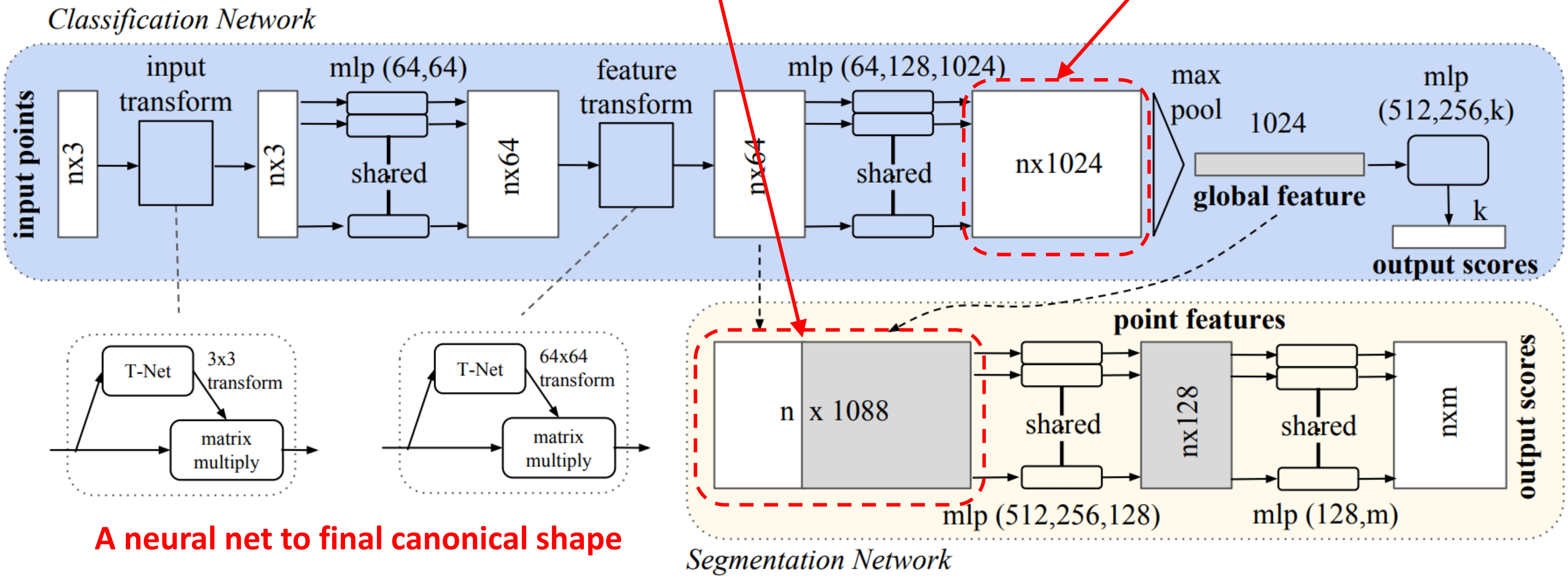
Questions?

Point-wise processing

- PointNet

From n x 3 to n x 1088

From n x 3 to n x 1024

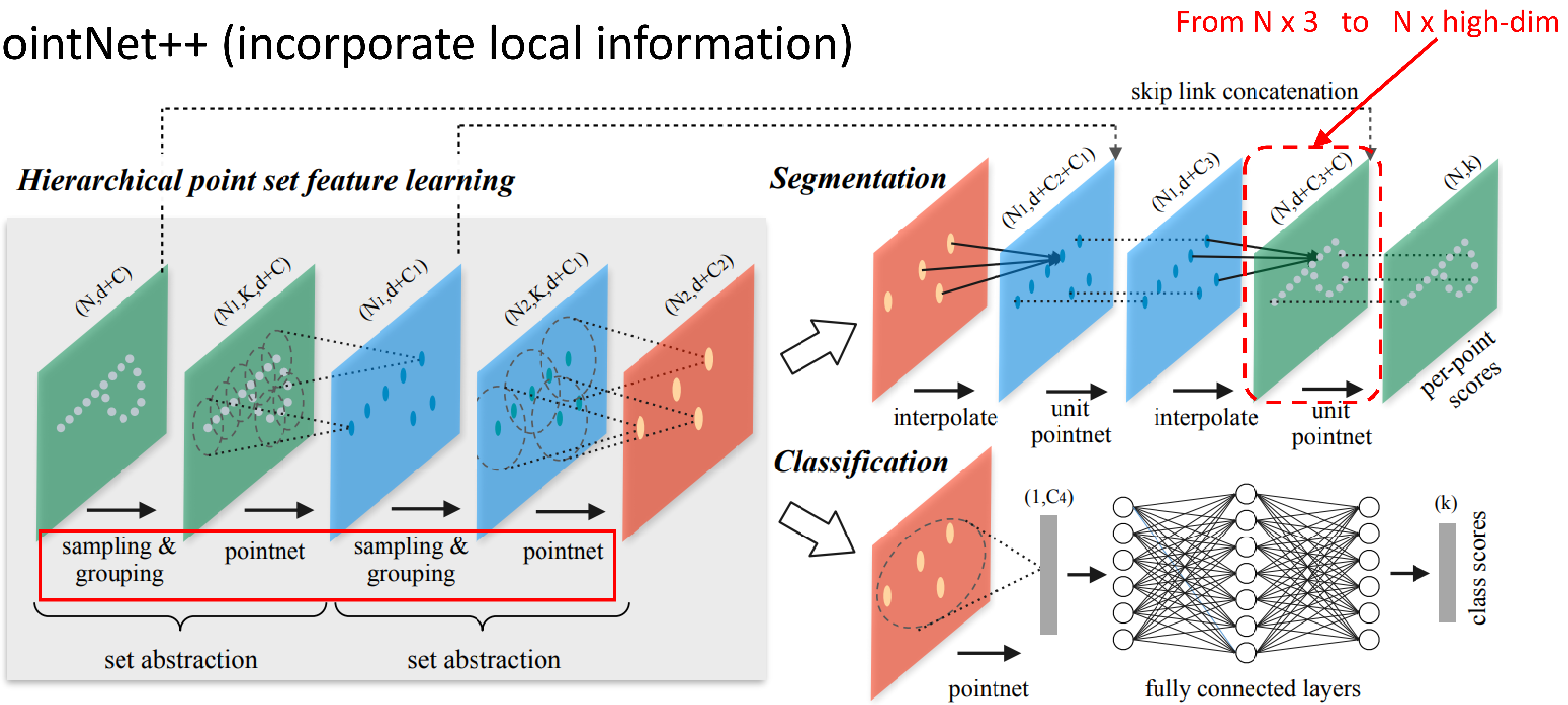


A neural net to final canonical shape

[Qi et al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, 2017]

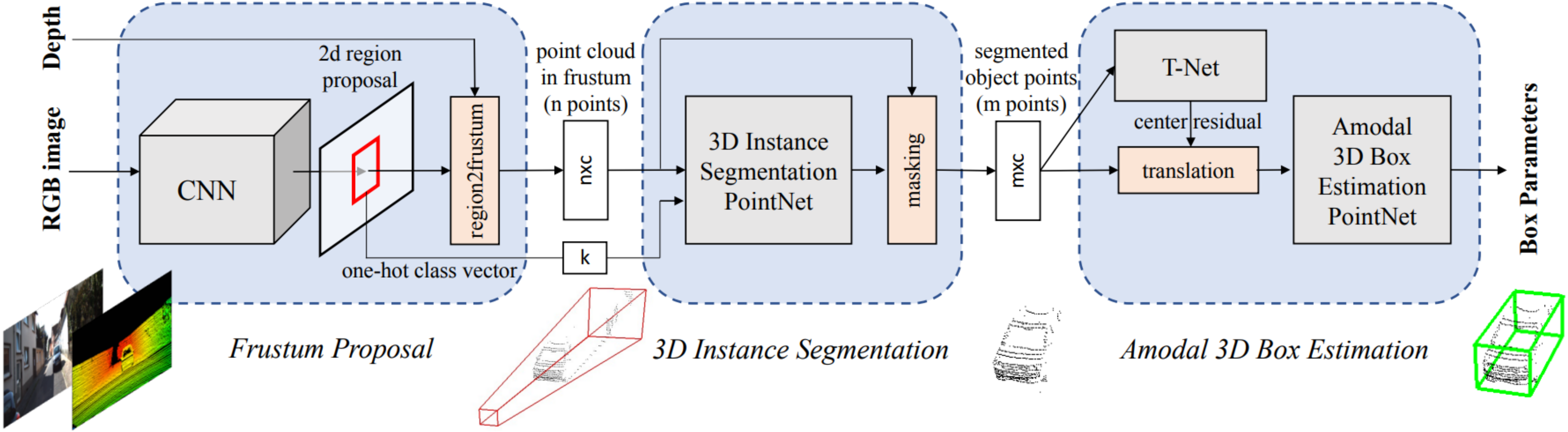
Point-wise processing

- PointNet++ (incorporate local information)



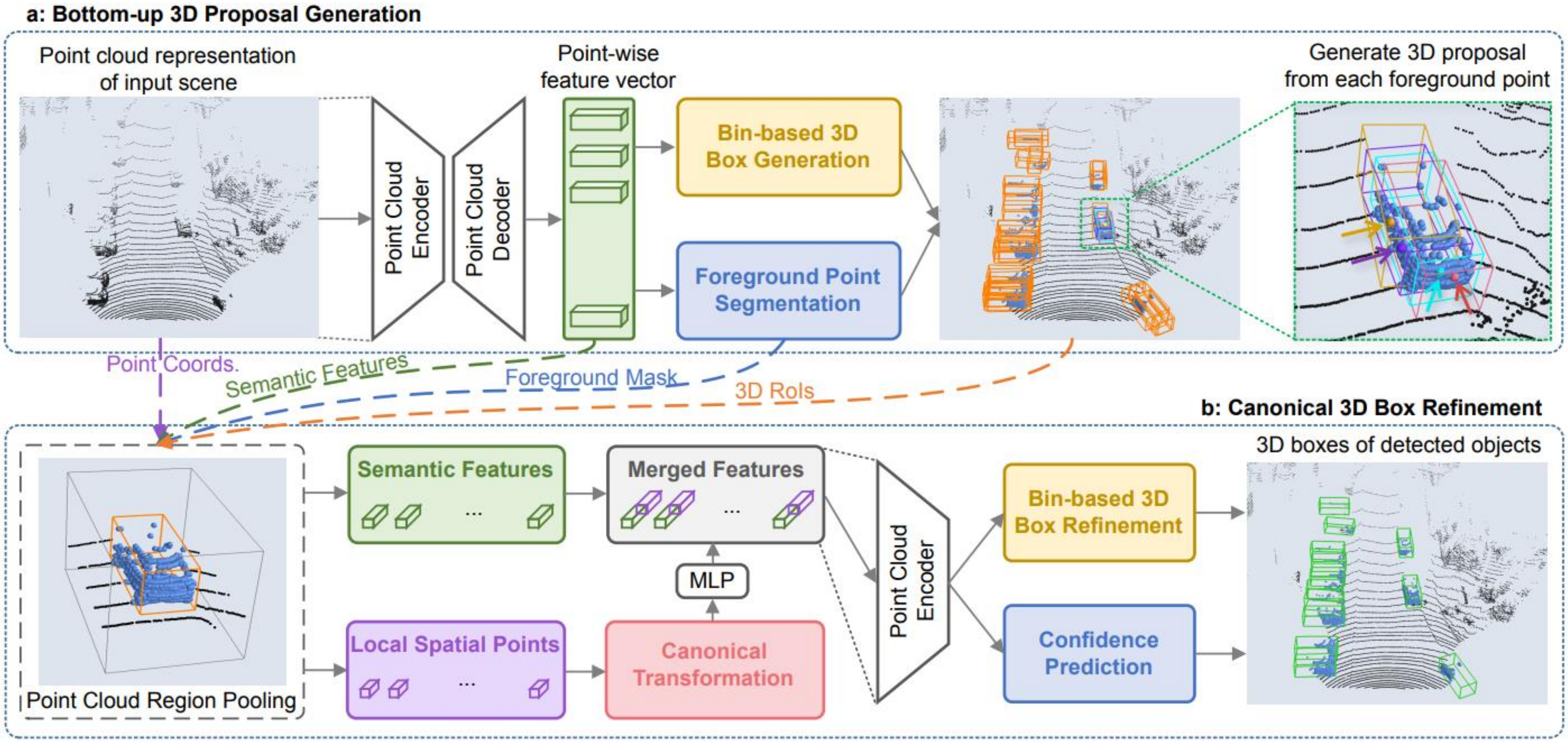
[Qi et al., PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, 2017]

Point-wise 3D object detectors



[Qi et al., Frustum PointNets for 3D Object Detection from RGB-D Data, 2018]

Point-wise 3D object detectors



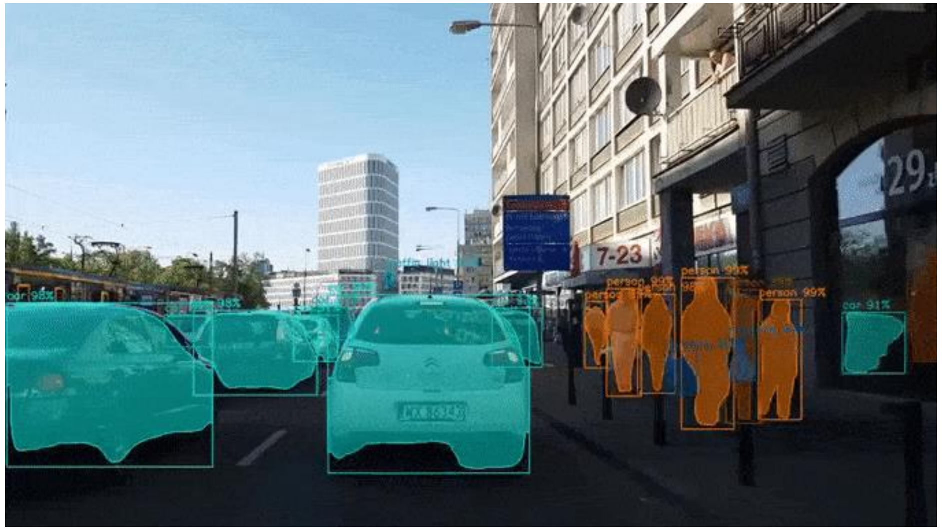
[Shi et al., PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, 2019]

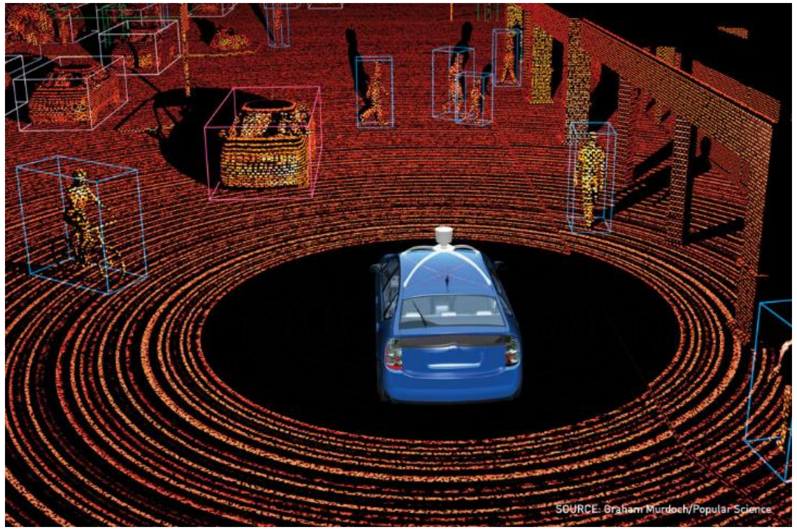
Example results

[Chen et al., CVPR 2017]

Image-based 3D perception

Affordability and reliability

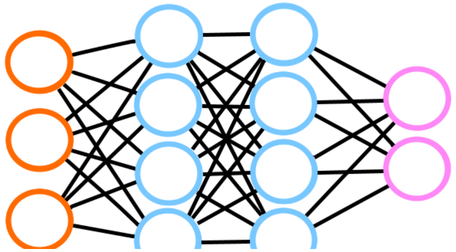


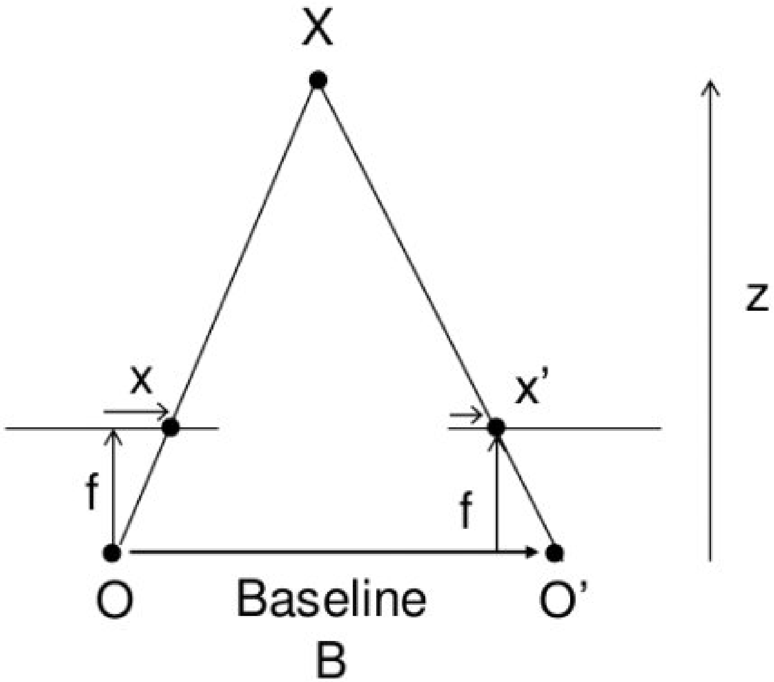


Images provide no depth

LiDAR is expensive (> $10K)
Over-reliance on LiDAR is risky

Stereo depth estimation





I_l

I_r

D

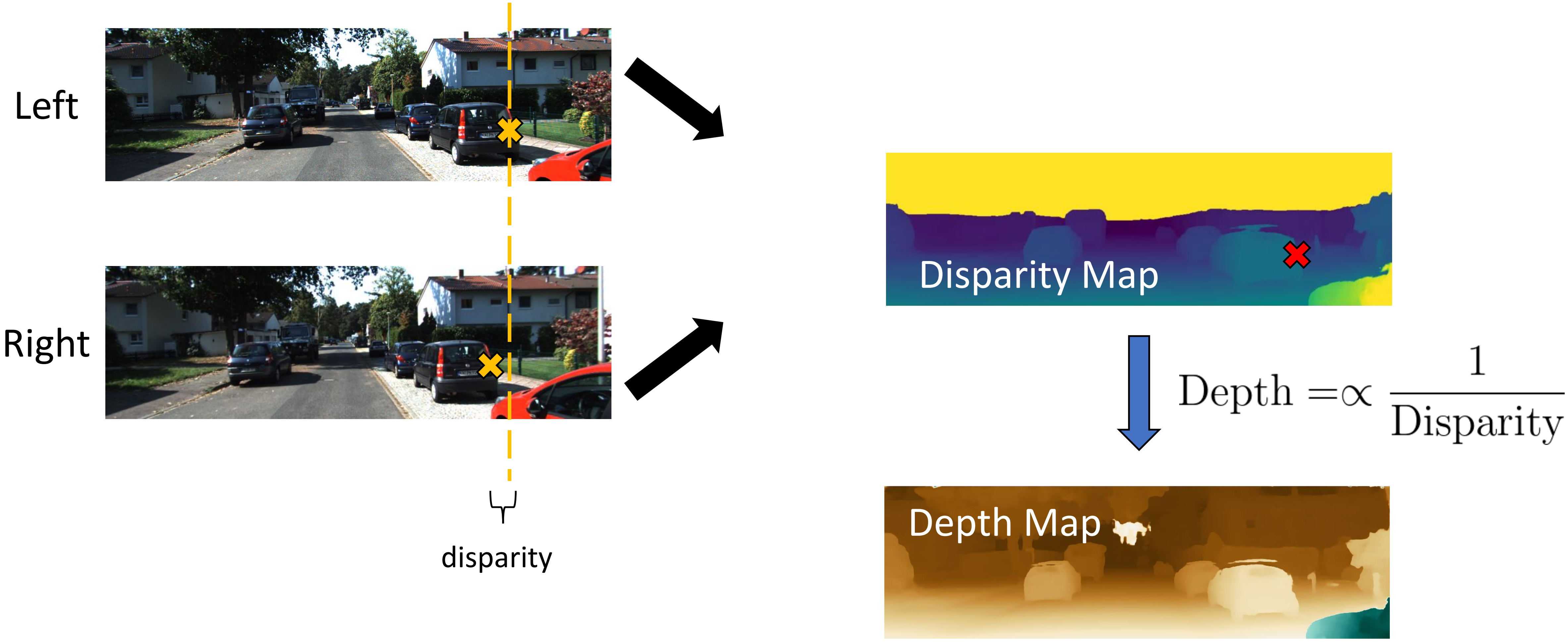
$$I_l(u,v) = I_r(u, v - D(u,v))$$

Focal length Baseline

$$Z(u,v) = \frac{f_U \times b}{D(u,v)}$$

Z

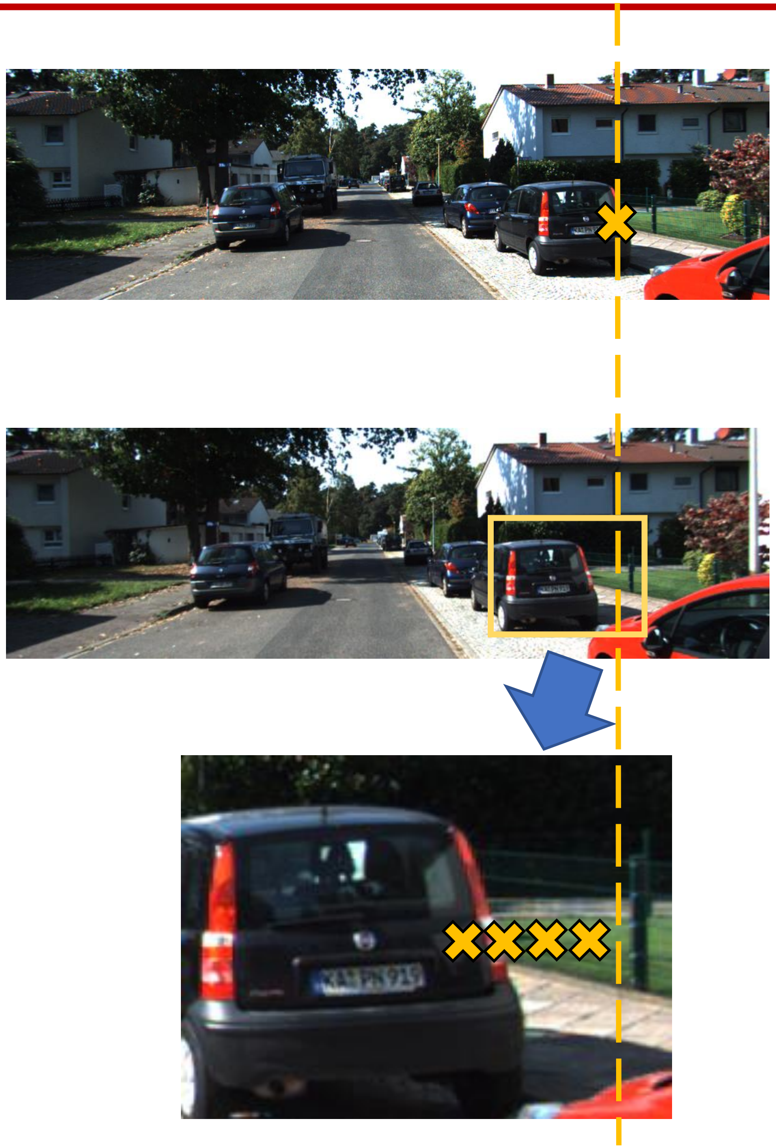
Stereo depth estimation

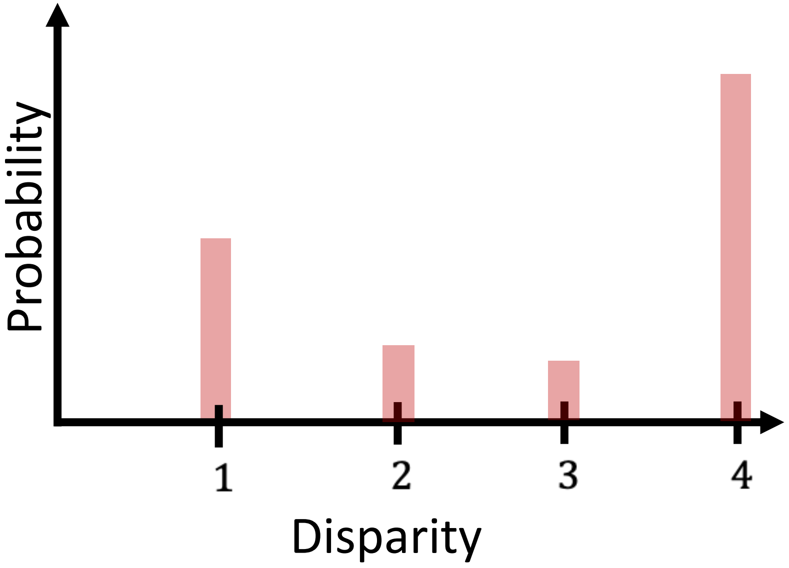


Stereo depth estimation

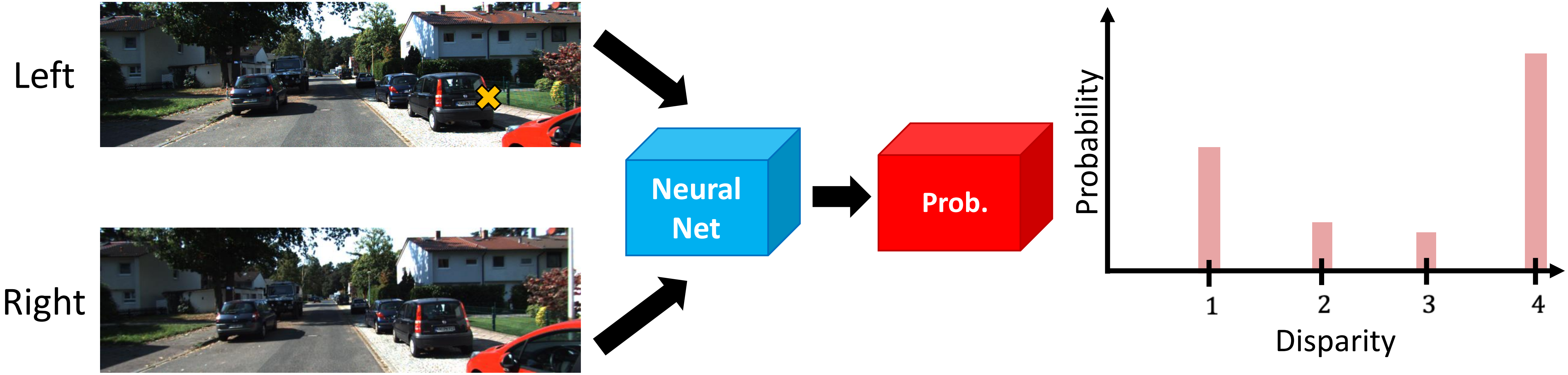
Left

Right

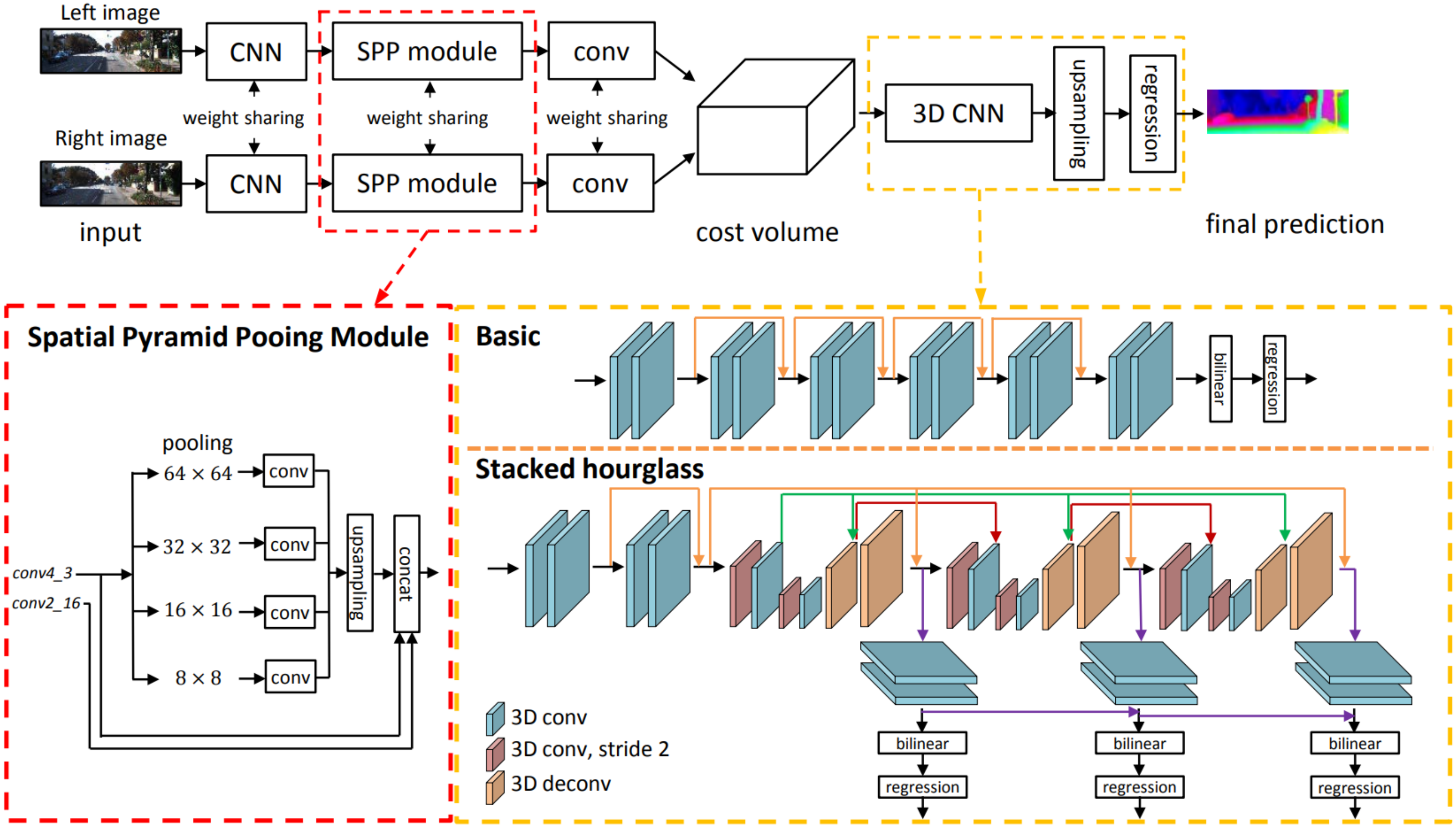




Stereo depth estimation

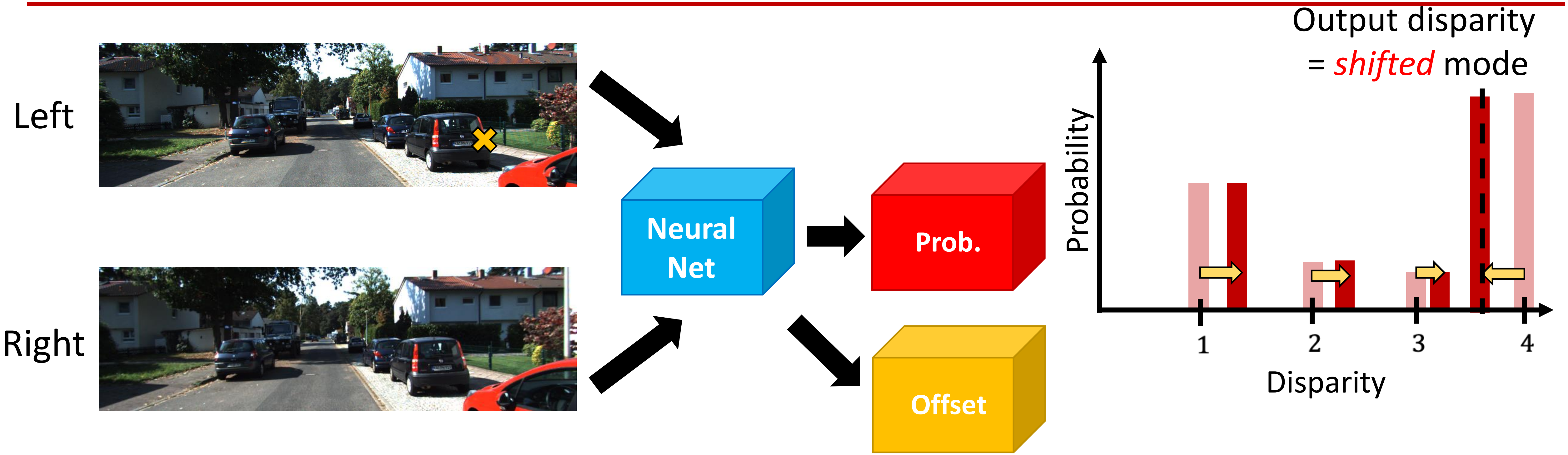


Pyramid stereo matching network (PSMNet)

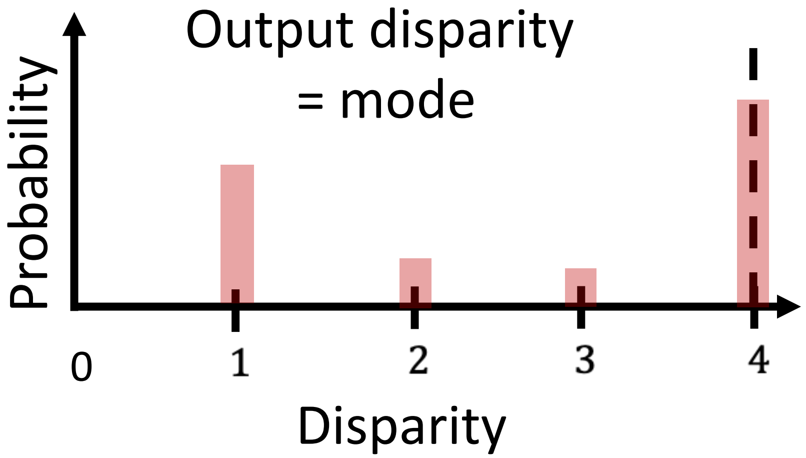


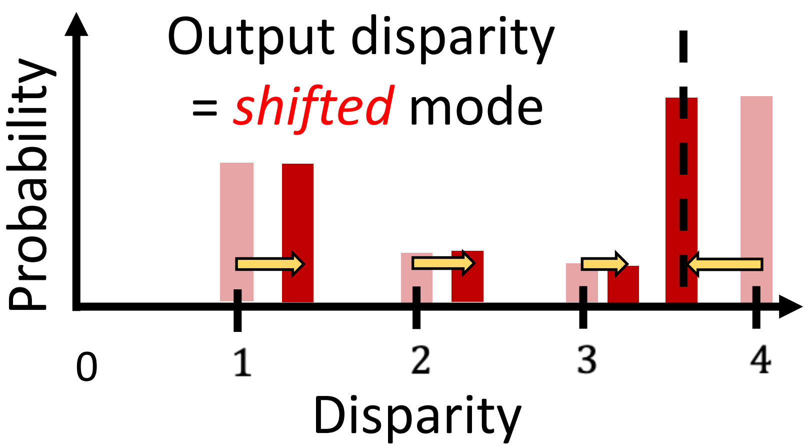
[Chang et al., Pyramid stereo matching network, 2018]

Continuous Disparity Network

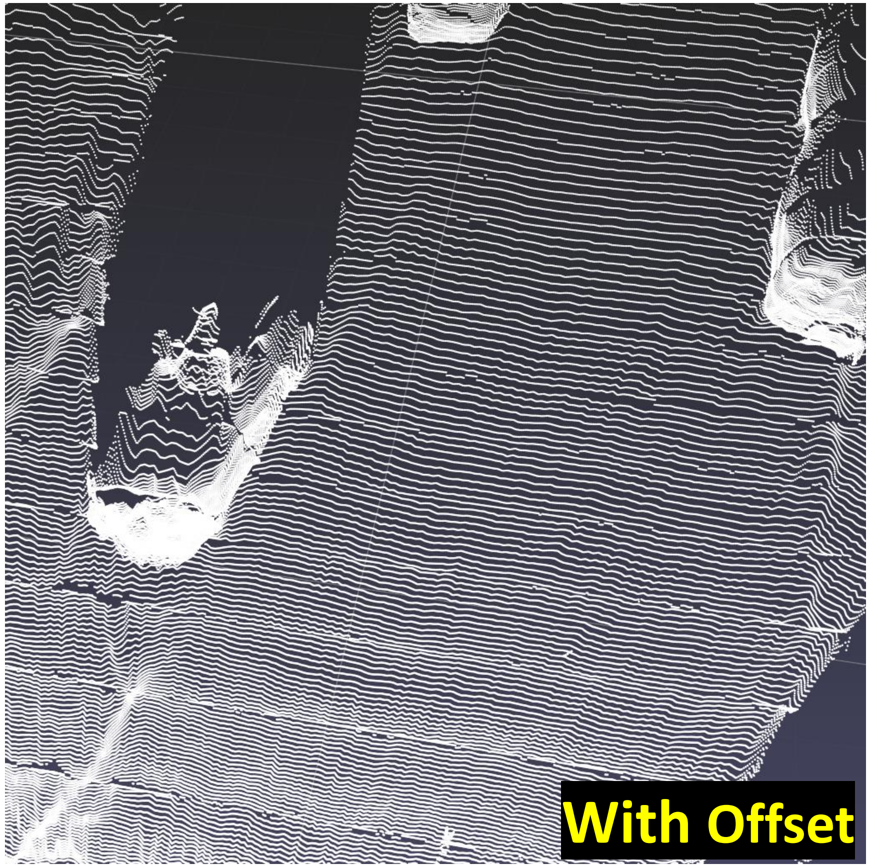


[Garg et al., Wasserstein Distances for Stereo Disparity Estimation, 2020]

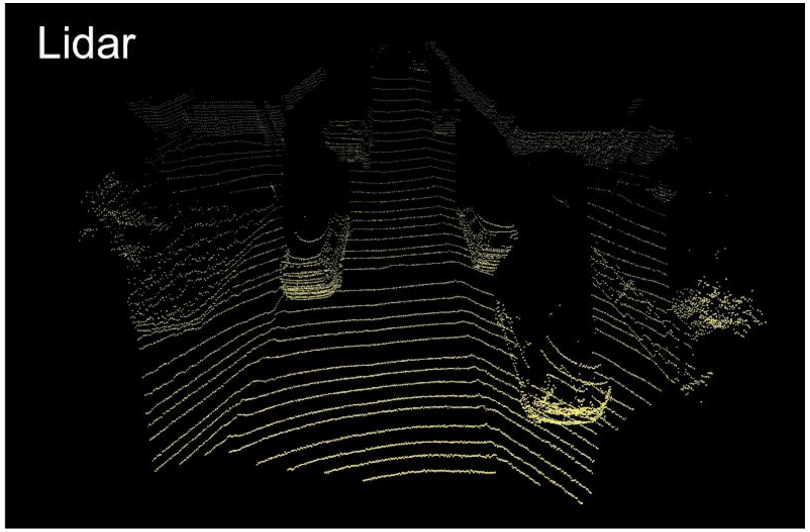

Output disparity
= mode
Probability
Disparity
0
1
2
3
4


Output disparity
= *shifted* mode
Probability
Disparity
0
1
2
3
4

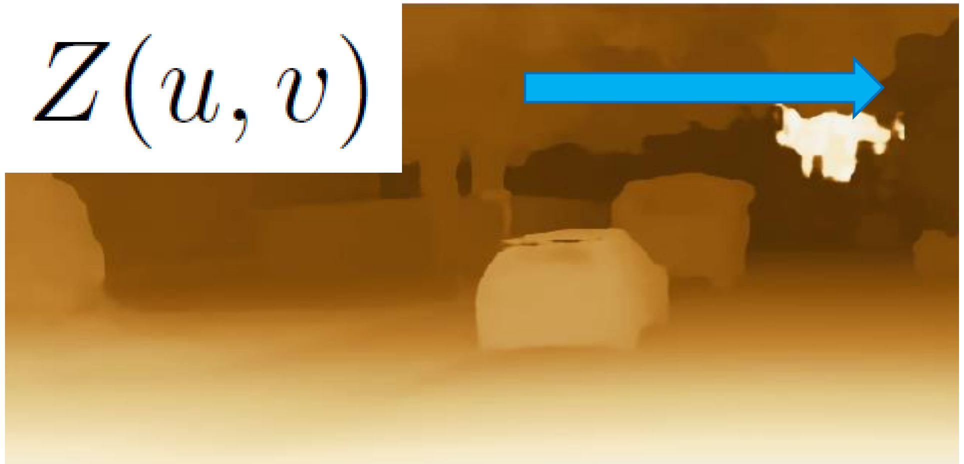

Without Offset


With Offset

LiDAR vs. camera-based depth



Camera-based depth estimation

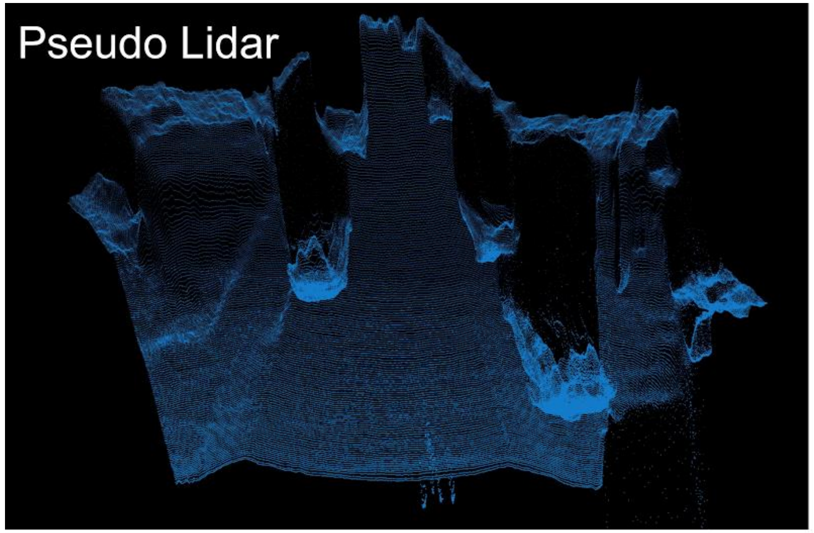


depth: $z = Z(u, v)$,

width: $x = \dfrac{(u - c_U) \times z}{f_U}$,

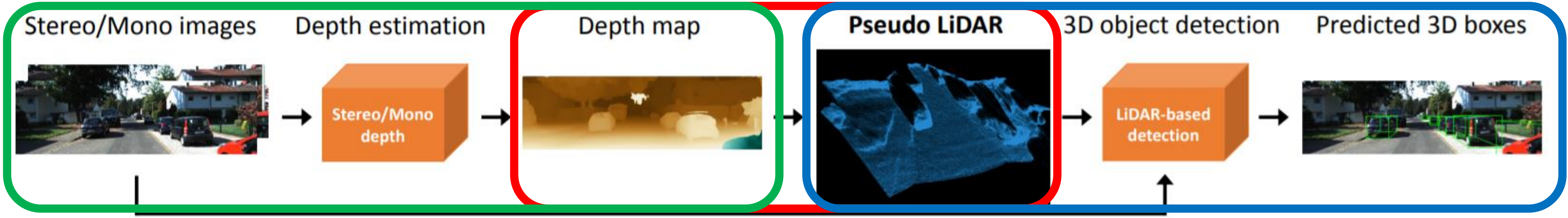
height: $y = \dfrac{(v - c_V) \times z}{f_V}$,

Pseudo-LiDAR representation



Pseudo-LiDAR framework

- **Pseudo-LiDAR representation:** gluing depth estimation + LiDAR-based detection



- SOTA camera-based depth estimators and LiDAR-based object detectors can seamlessly be incorporated!

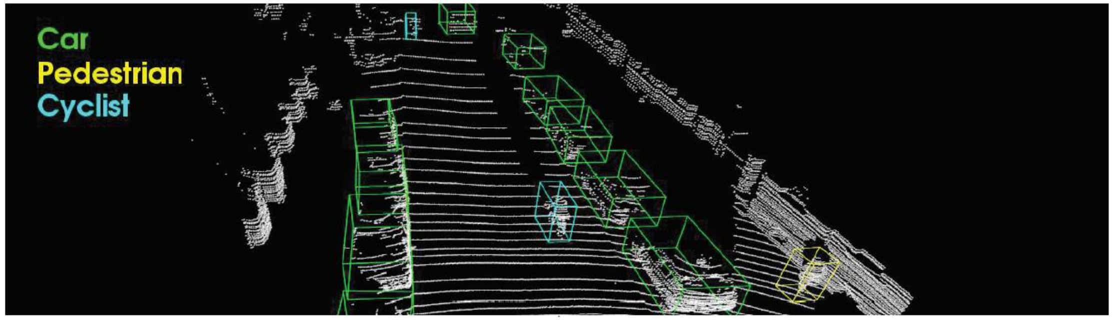
Depth estimation models
- DORN (CVPR18)
- MC-CNN (JMLR 16)
- GC-Net (ICCV 17)
- PSMNet (CVPR18)
- ......

3D object detection models
- PIXOR (CVPR 18)
- F-PointNet (CVPR 18)
- AVOD (IROS 18)
- PointRCNN (CVPR 19)
- ......

Yan Wang, **Wei-Lun Chao**, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger, "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," CVPR, 2019

Data representation is important!

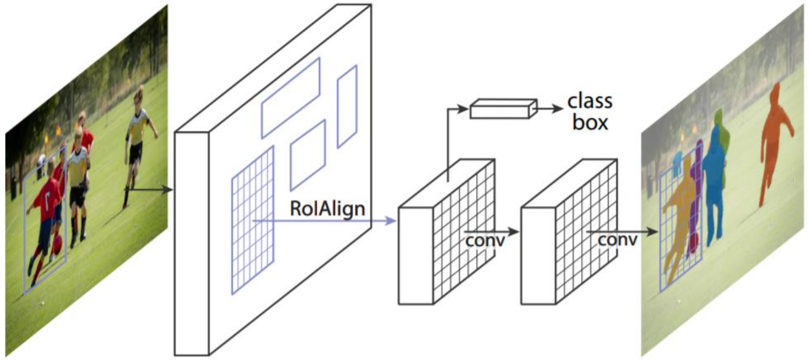
- **LiDAR-based:** 3D point clouds



[Source: VoxelNet, CVPR 2018]

- **Camera-based:** 2D depth maps

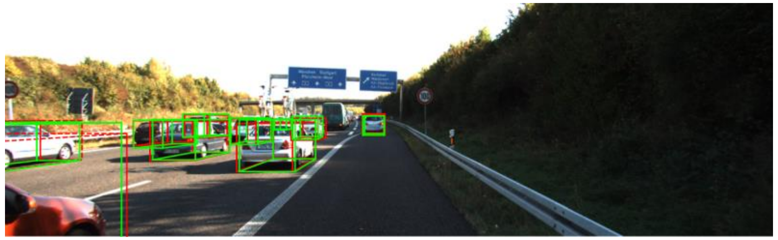
[Source: Mask R-CNN, ICCV 2017]

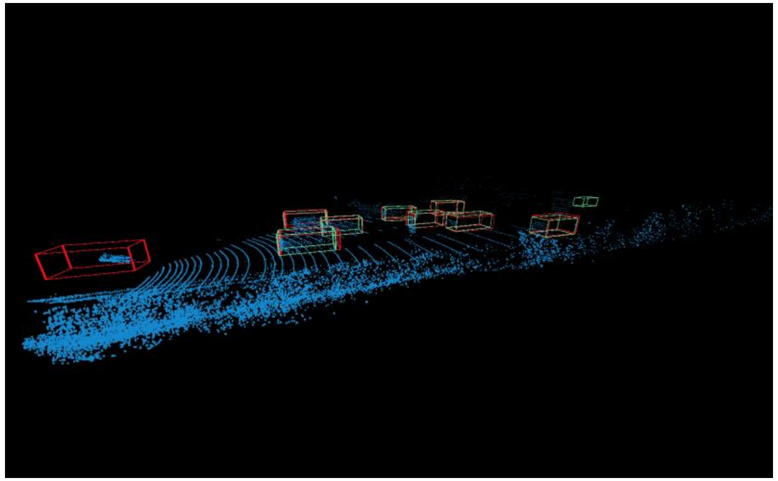


  - Processing depth as an image leads to large distortion!

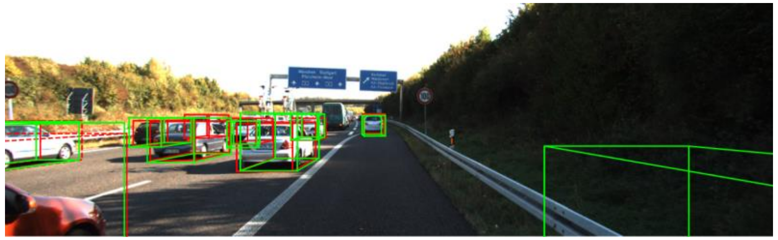
Example results

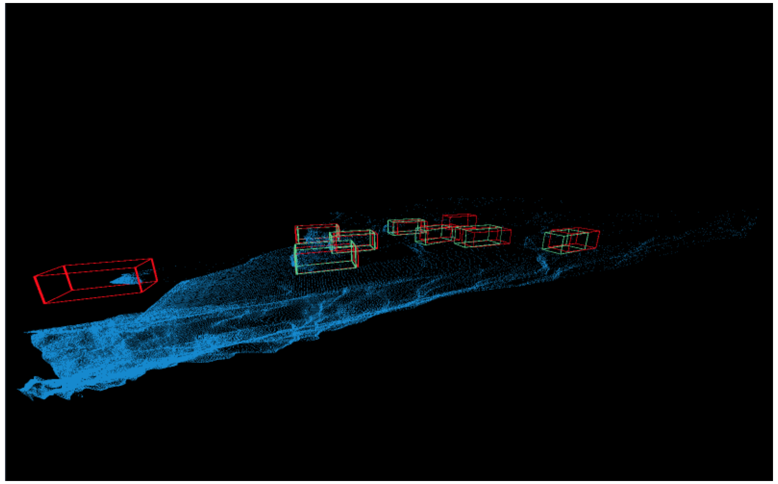
LiDAR



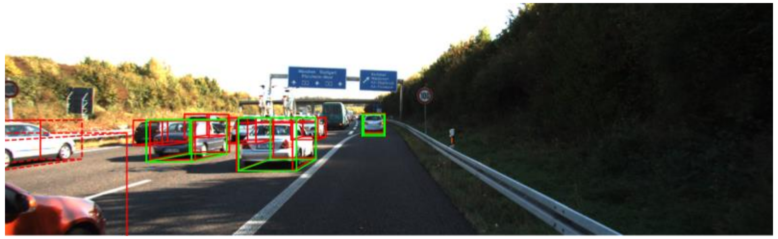


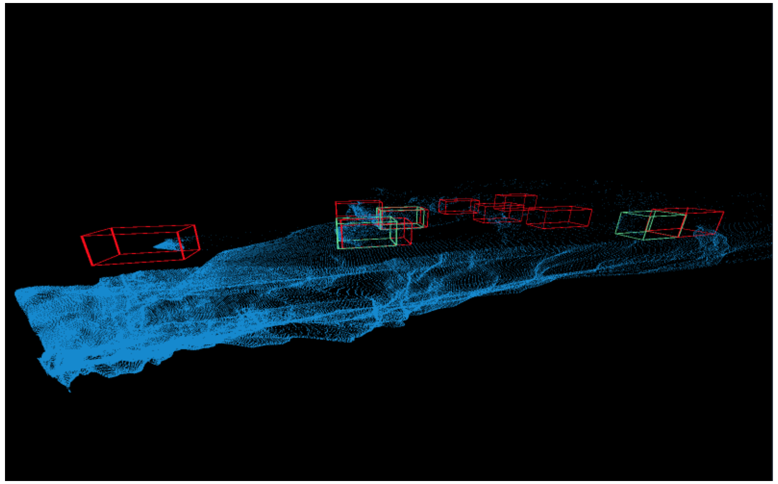
pseudo-LiDAR

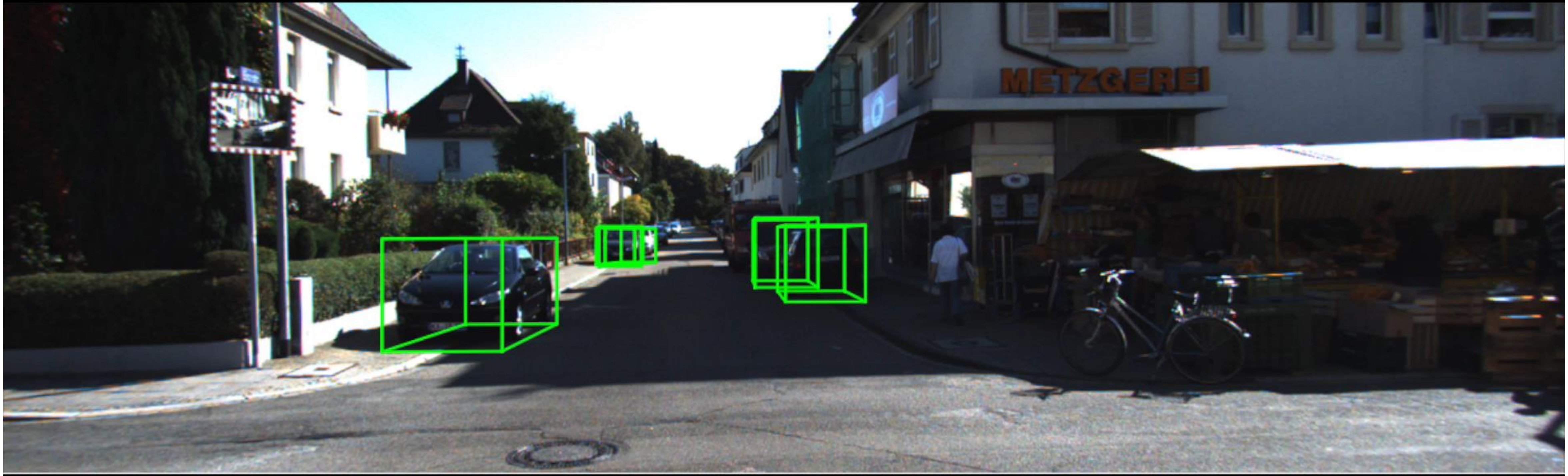




Depth-map

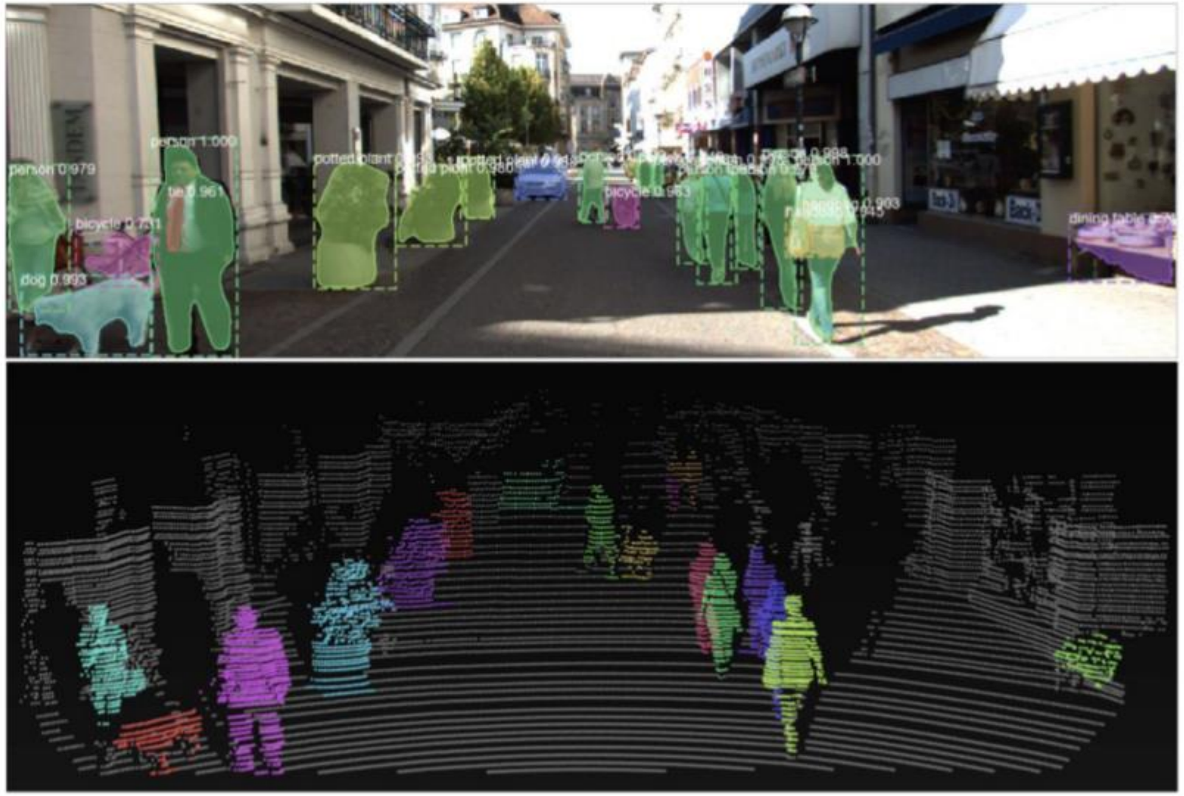




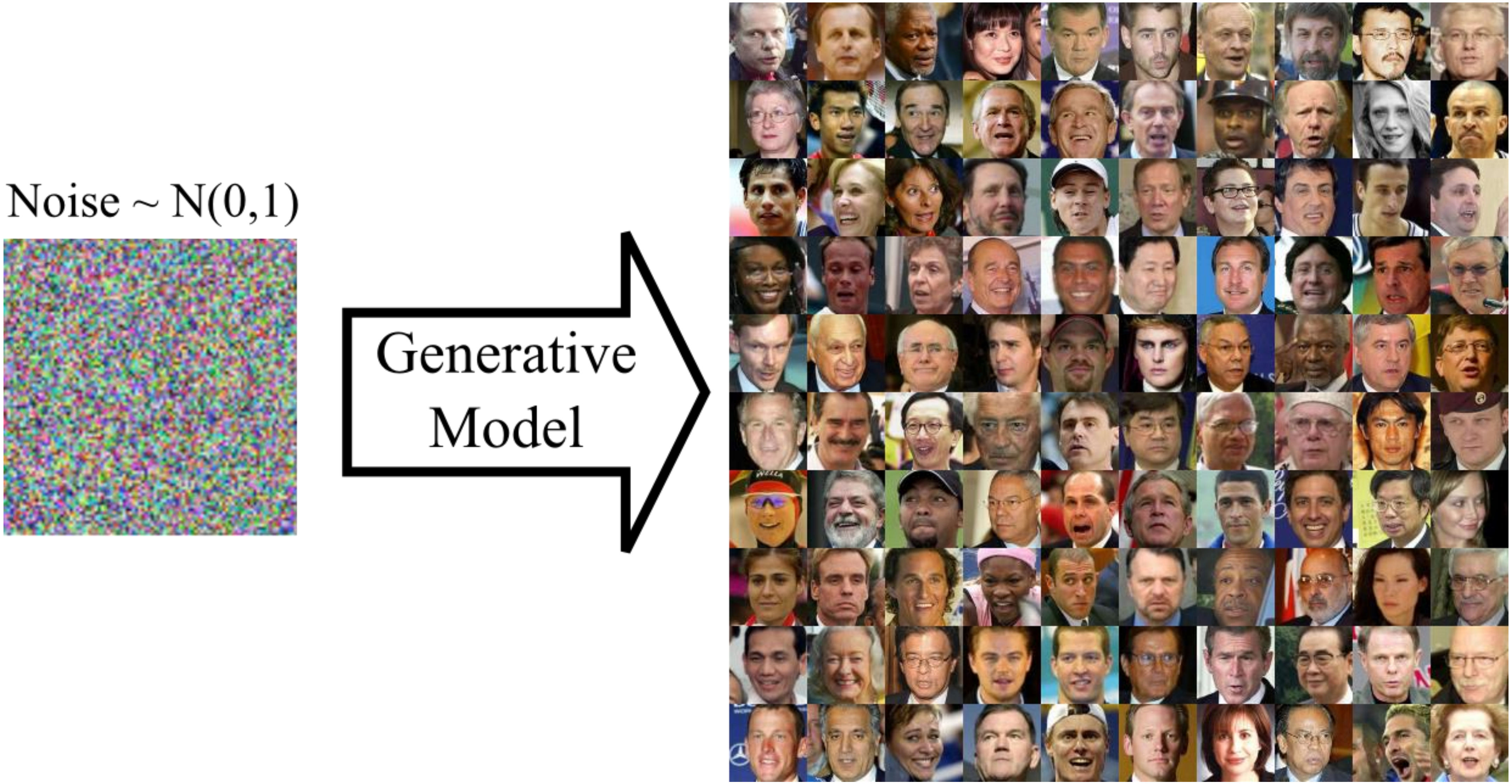

METZGEREI

Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



Generative models

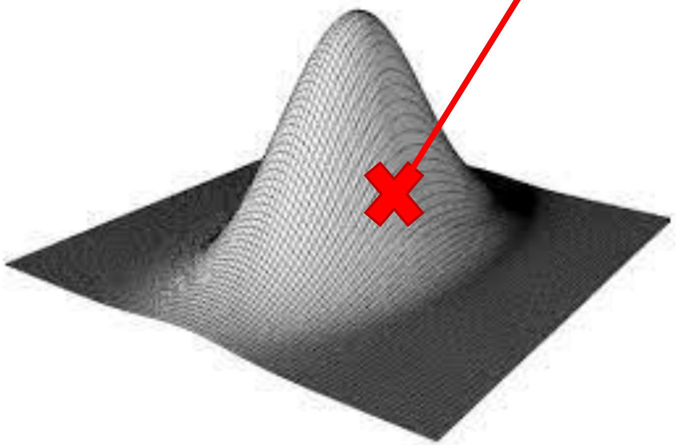


Generative models

image

Easily samplable distribution

$z \sim \mathcal{N}(0, \mathbf{I})$



$g: Z \mapsto X$

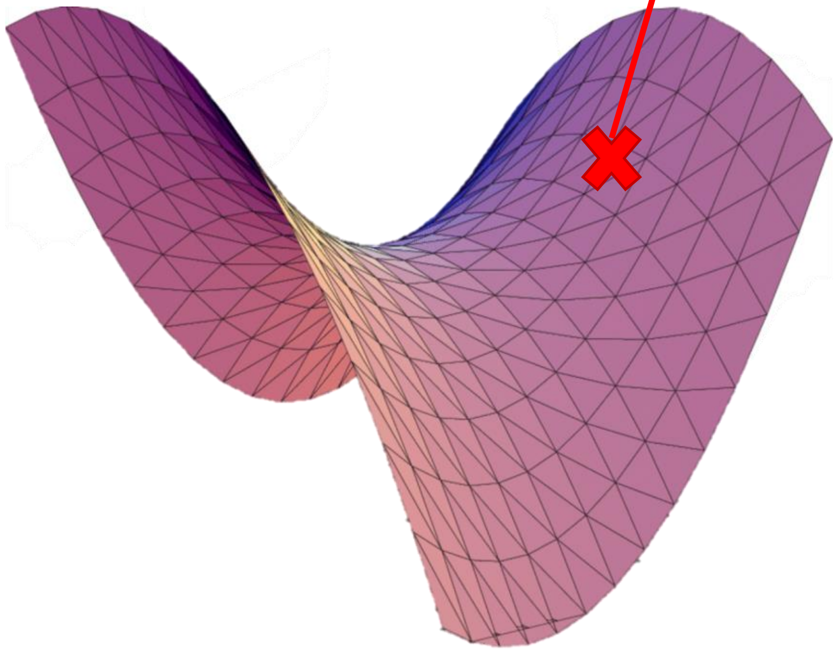


image manifold: $p(\boldsymbol{x})$

[Credits: Tutorial on Diffusion Models]

What and how to learn?

Real image: $\boldsymbol{x} \sim q(\boldsymbol{x})$

Generative model: $\boldsymbol{x}' \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}')$

$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$

$\boldsymbol{x}' \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}' | \mathbf{z})$ or $\boldsymbol{x}' = g_{\boldsymbol{\theta}}(\mathbf{z})$

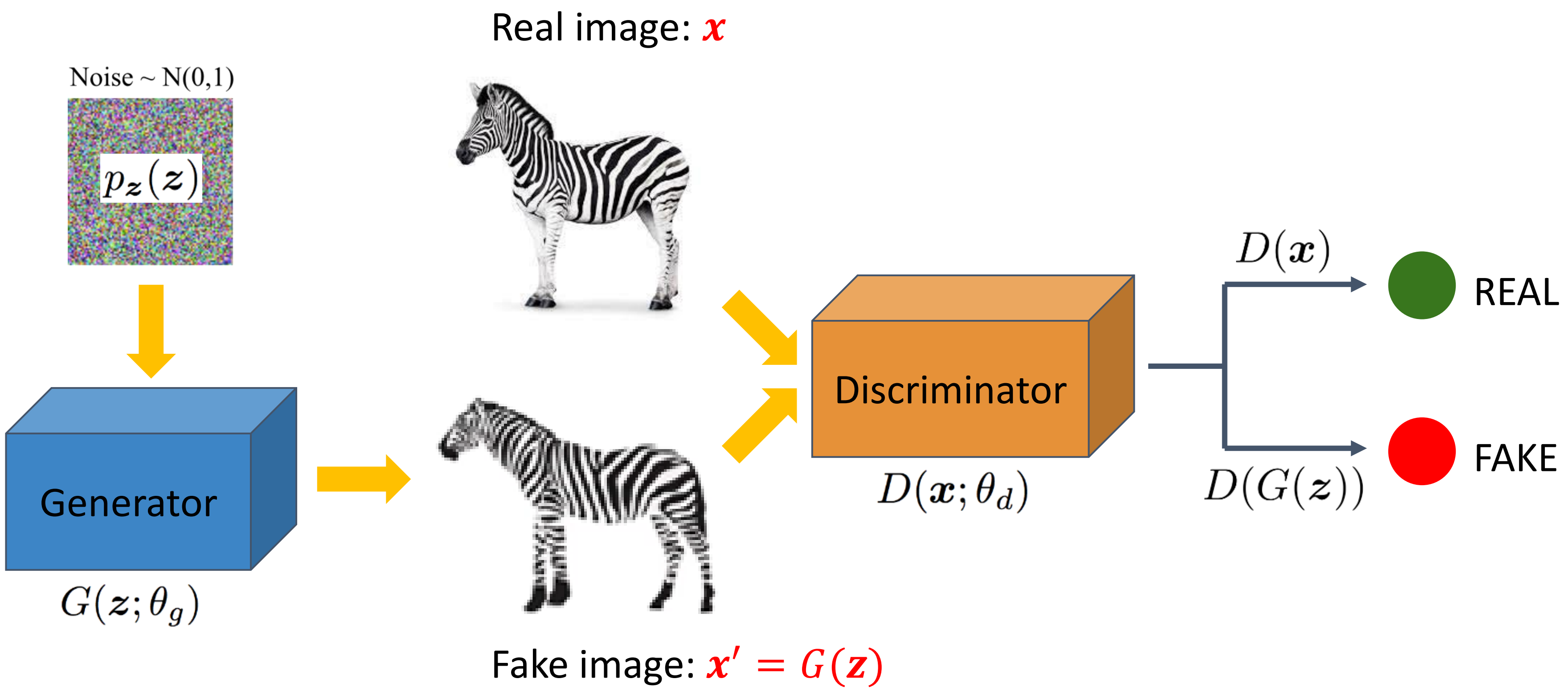
Objective 1

- $p_{\boldsymbol{\theta}}$ can explain real images $\boldsymbol{x} \sim q(\boldsymbol{x})$
- Maximize $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{x} \sim q(\boldsymbol{x})$
- Example: variational auto-encoder (VAE)

Objective 2

- $\boldsymbol{x} \sim q(\boldsymbol{x})$ and $\boldsymbol{x}' \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}')$ are indistinguishable
- Example: generative adversarial net (GAN)

Generative adversarial net (GAN)

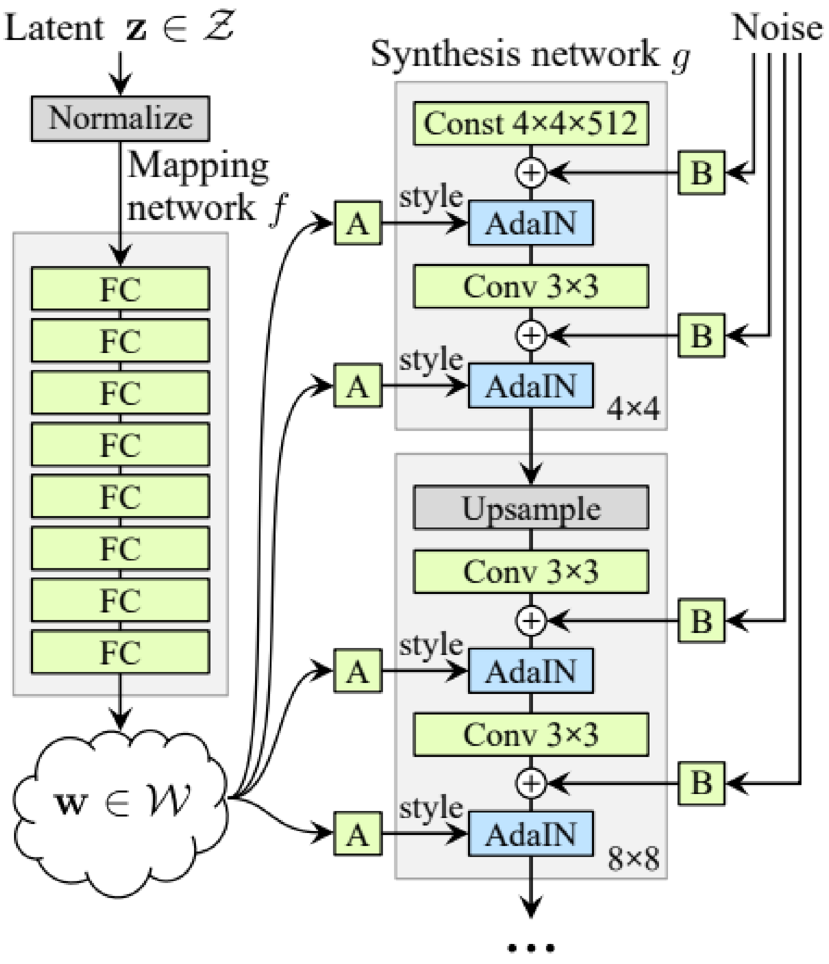


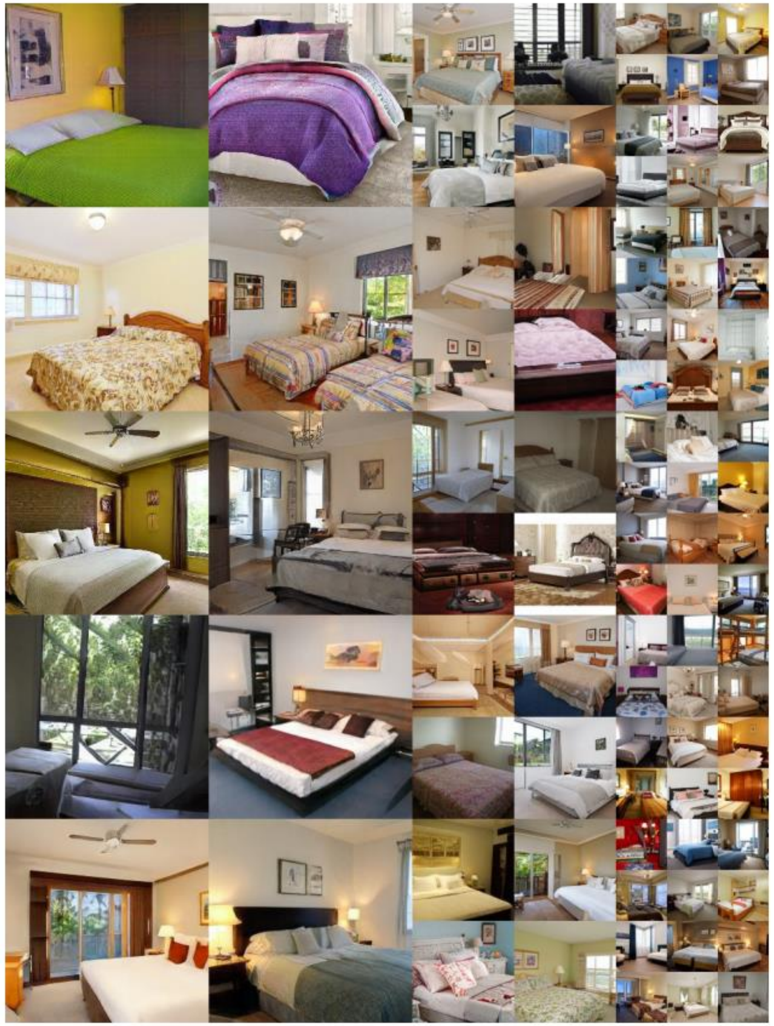
Iterate between:

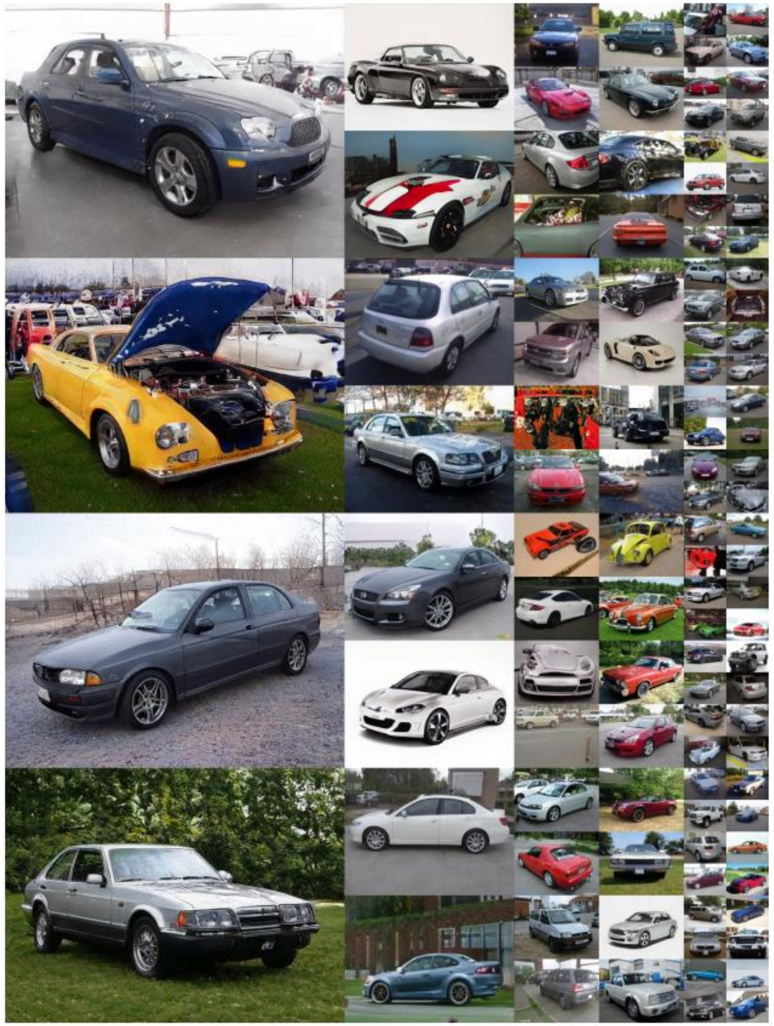
- Update *D* to distinguish between $\boldsymbol{x} \sim q(\boldsymbol{x})$ and $\boldsymbol{x}' = G(\boldsymbol{z}), \boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$
- Update *G* such that *D* cannot distinguish between $\boldsymbol{x} \sim q(\boldsymbol{x})$ and $\boldsymbol{x}' = G(\boldsymbol{z}), \boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$

[Credits: Mengdi Fan and Xinyu Zhou, CSE 5539 course presentation]

Example results (by Style-GAN)







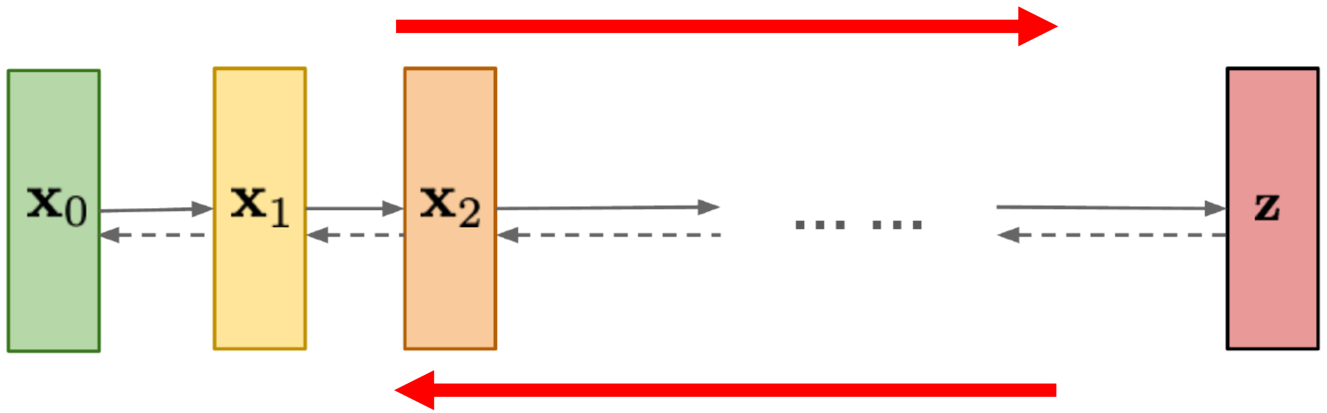
[A Style-Based Generator Architecture for Generative Adversarial Networks, CVPR 2019]

Other generative models

- Denoising Diffusion Probabilistic Models (PPDM)
  - Learn to inverse the diffusion process
  - Can generate very high-quality images

Diffusion by simple Gaussian

Diffusion models:
Gradually add Gaussian
noise and then reverse



Denoising by neural networks (each step by a U-net!)

[Denoising Diffusion Probabilistic Models, NeurIPS 2020]

Other generative models

- Denoising Diffusion Probabilistic Models (PPDM)

[Denoising Diffusion Probabilistic Models, NeurIPS 2020]

Other generative models

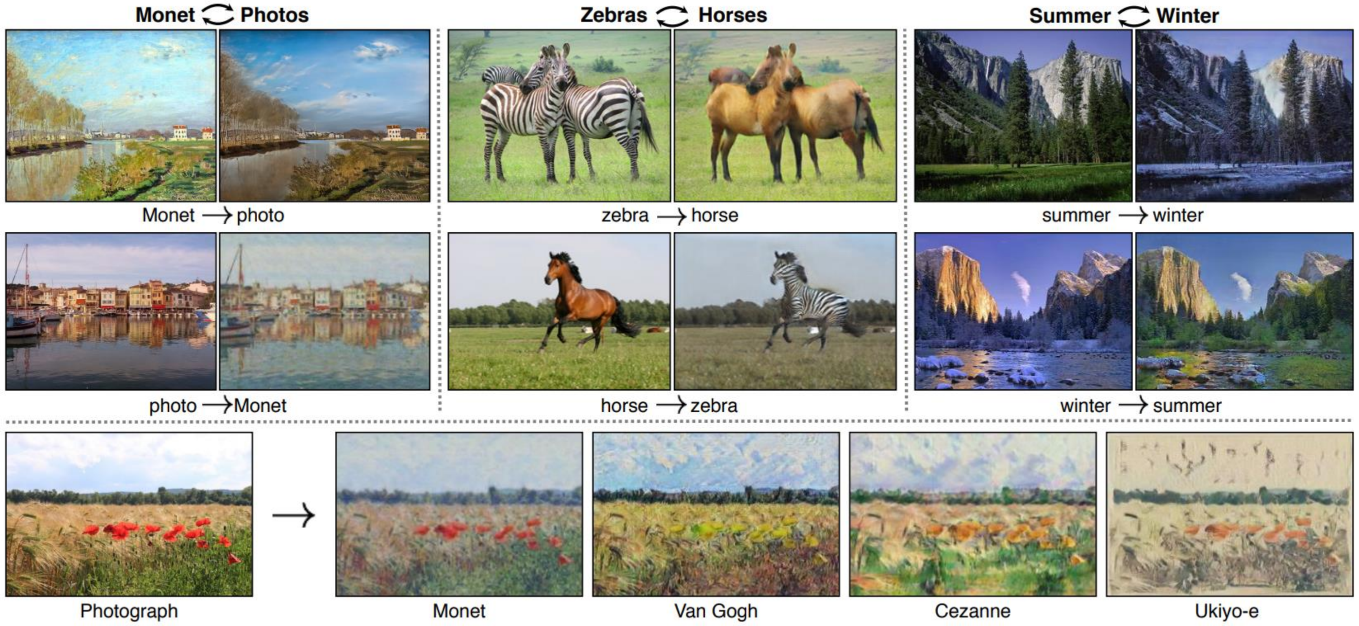
Big-GAN Diffusion models Real

[Diffusion Models Beat GANs on Image Synthesis, NeurIPS 2021]

Conditional image generation

Replace $z \sim \mathcal{N}(0, \mathbf{I})$ by

- Real image: image translation from domain A to domain B
- Can learn with "unpaired" data or "paired" data



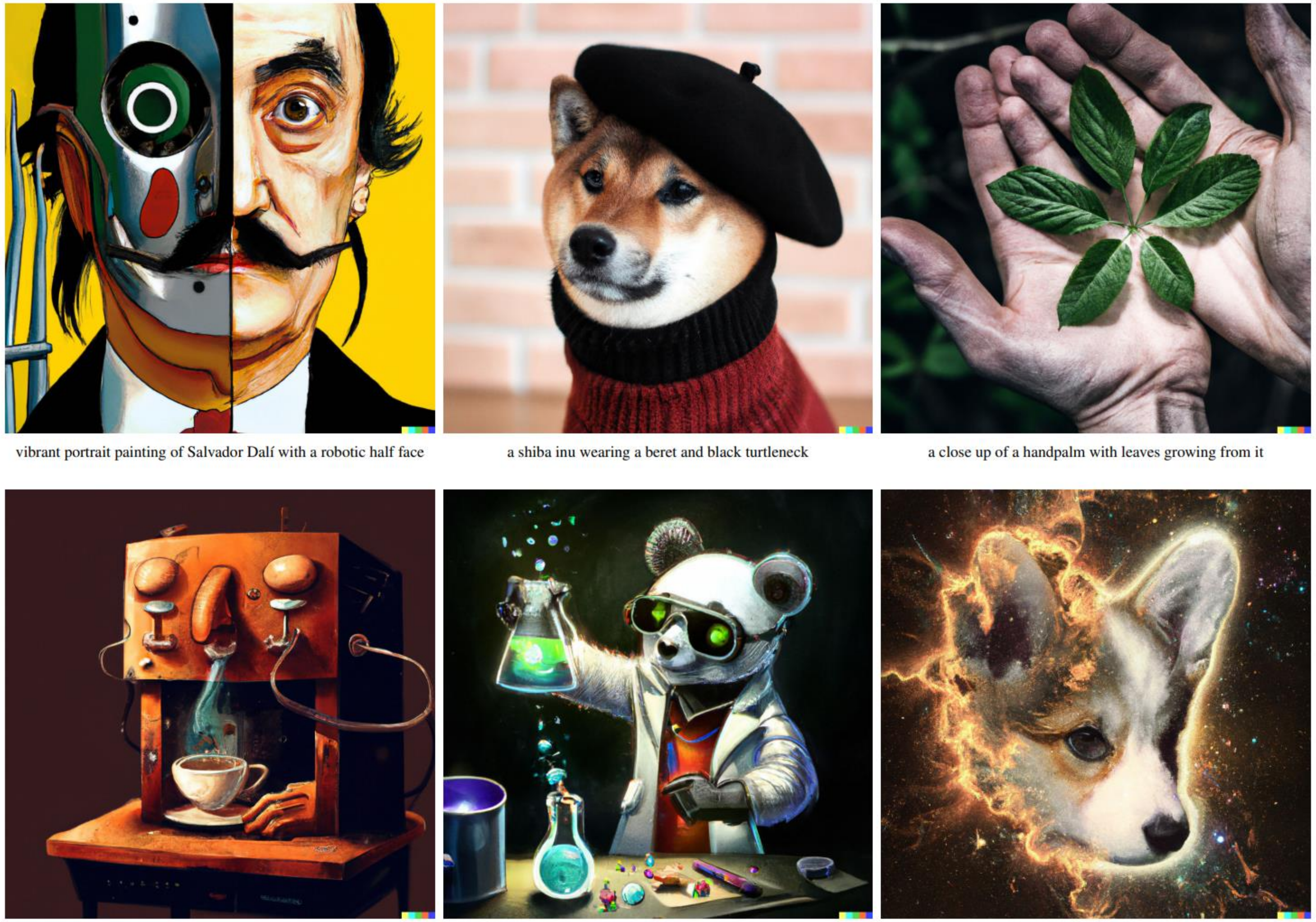
[Zhu et al., 2017]

[Wang et al., 2018]

Conditional image generation

Replace $z \sim \mathcal{N}(0, \mathbf{I})$ by

- Text: text-conditioned image generation
- Usually need to learn with "paired" data



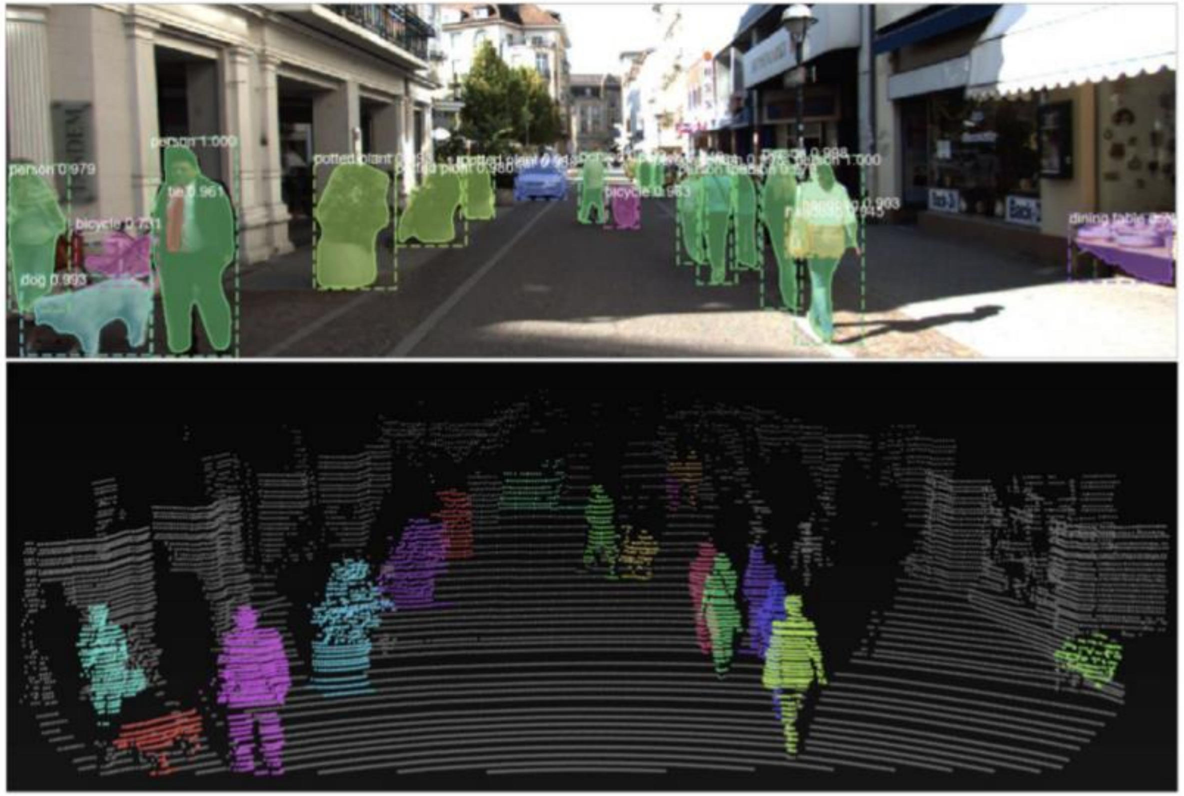
vibrant portrait painting of Salvador Dalí with a robotic half face | a shiba inu wearing a beret and black turtleneck | a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artstation | panda mad scientist mixing sparkling chemicals, artstation | a corgi's head depicted as an explosion of a nebula

[Hierarchical Text-Conditional Image Generation with CLIP Latents, arXiv 2022]

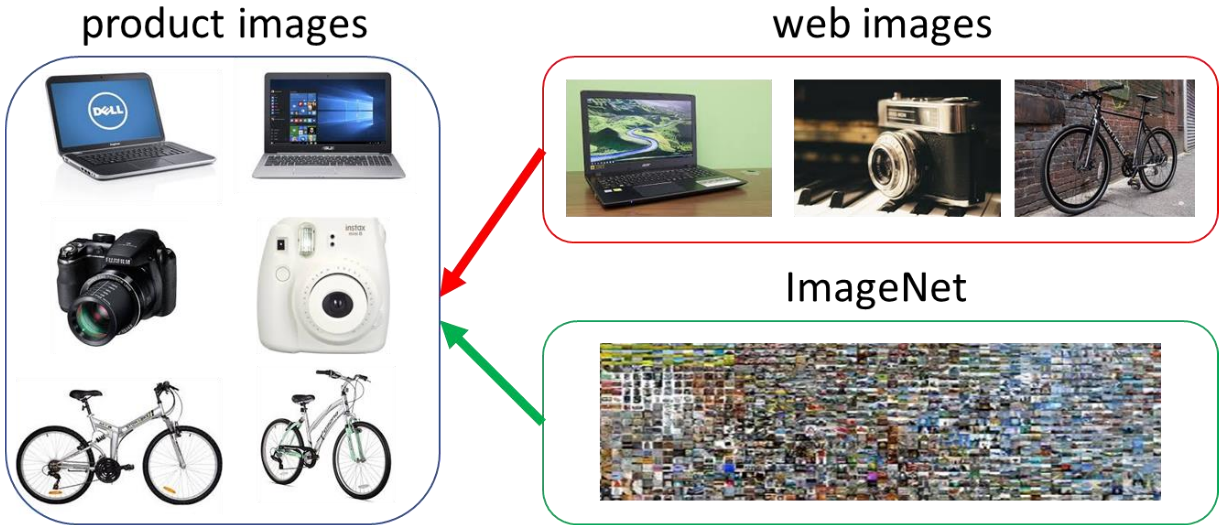
Outline

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



“Some” challenges in DL for CV

- Deep neural networks are “labeled” data hungry
- Mismatch between training and test data



Challenge-1: Insufficient (labeled) data

- Example: ImageNet-1K (ILSVRC)
  - 1,000 object classes
  - 1,000 training images per class

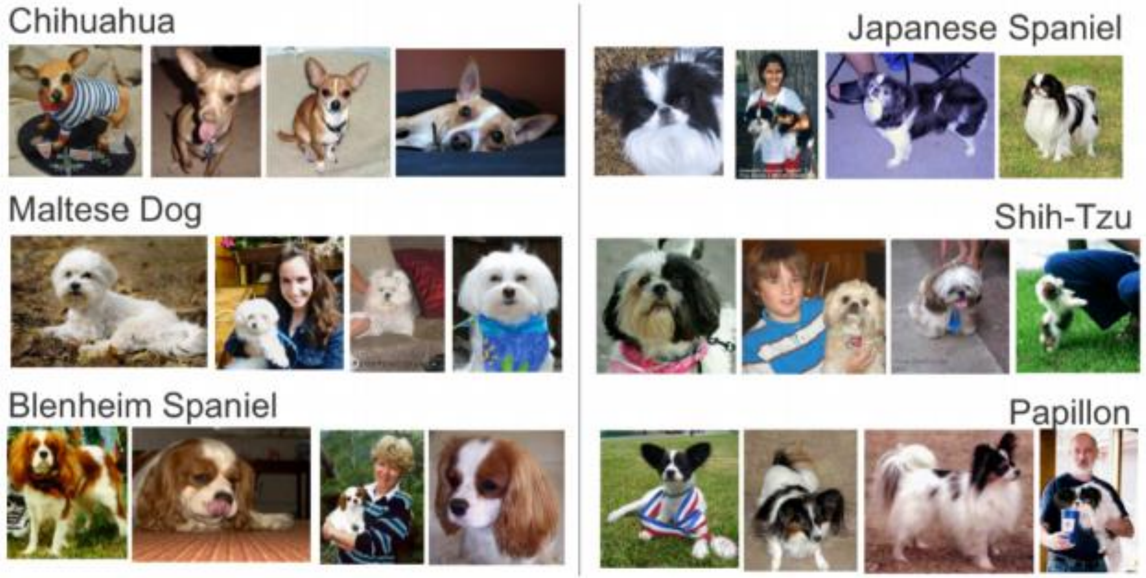
It’s hard to collect labeled data

- Collecting and annotating data is time-consuming and expensive

- Crowdsourcing can be noisy and may not be feasible for certain problems
  - e.g., medical images and applications

- For some applications, even “unlabeled” data can be hard to collect
  - e.g., fine-grained classes, long-tailed problems, data privacy and protection

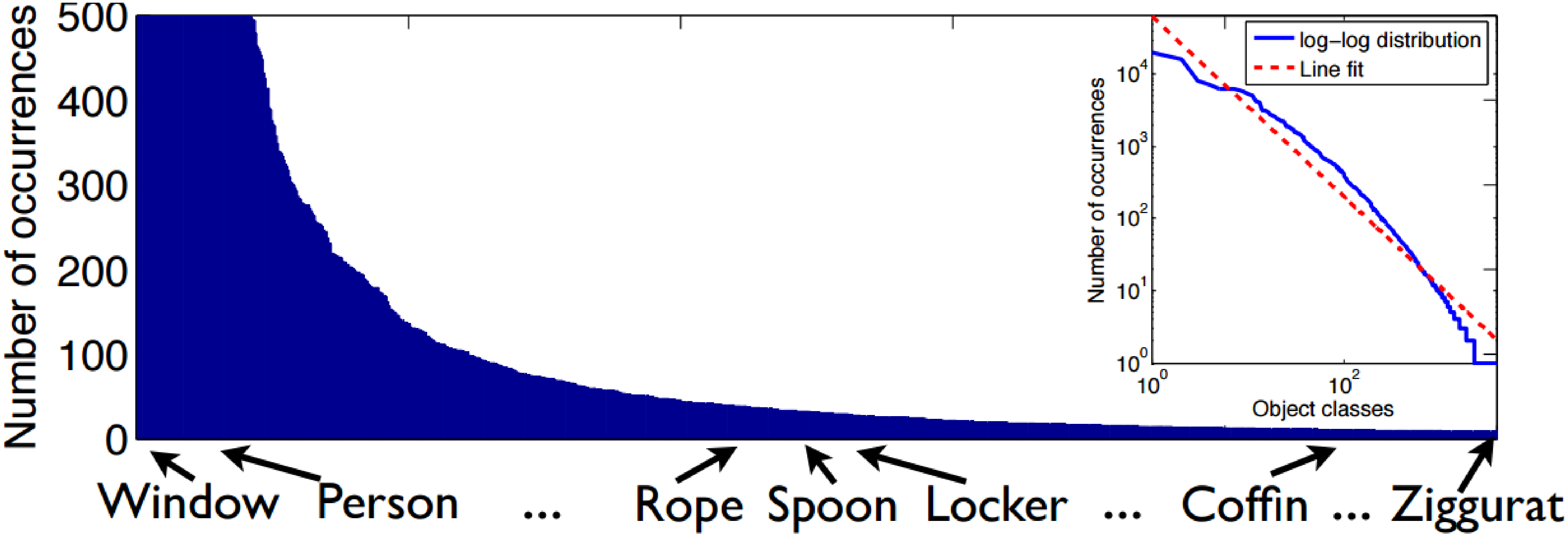
Fine-grained classes



[Credits: Rogerio Feris, ICCV-2019 slides]

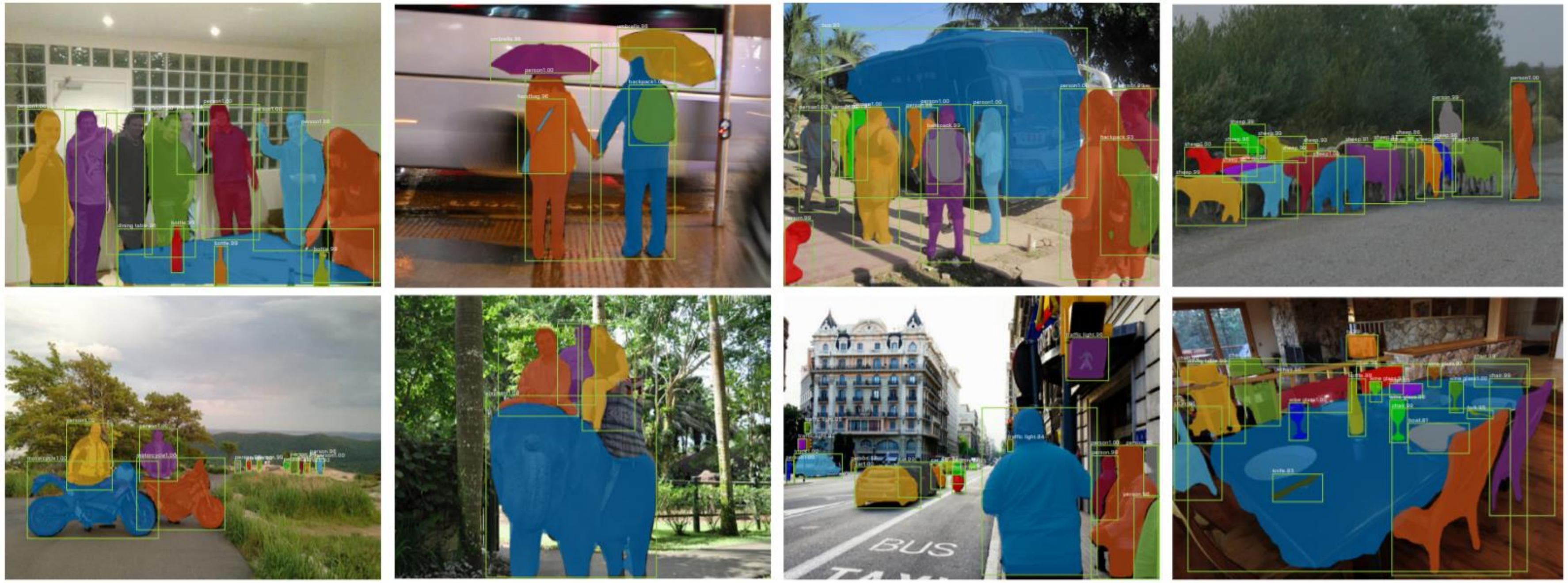
Long-tailed distribution

Objects in SUN datasets



Collecting dense labels is even harder

- Images with detailed instance segmentation labels

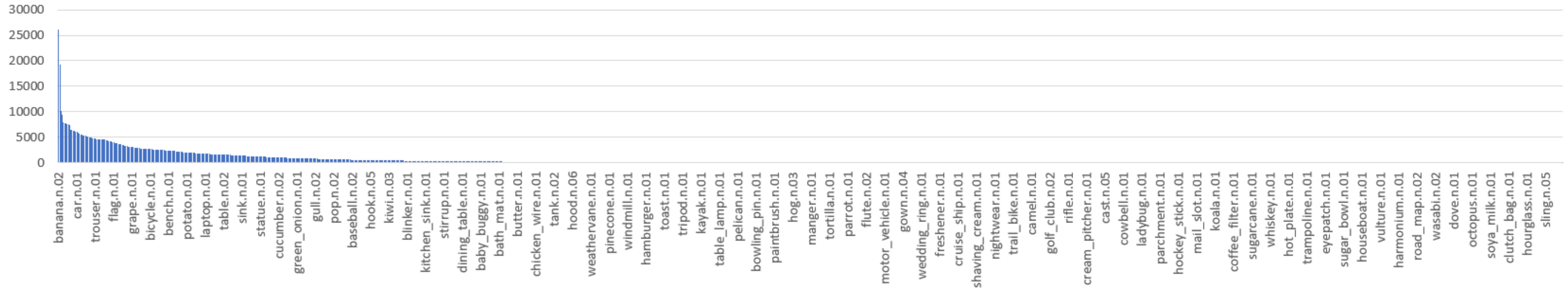


[He et al, 2017]

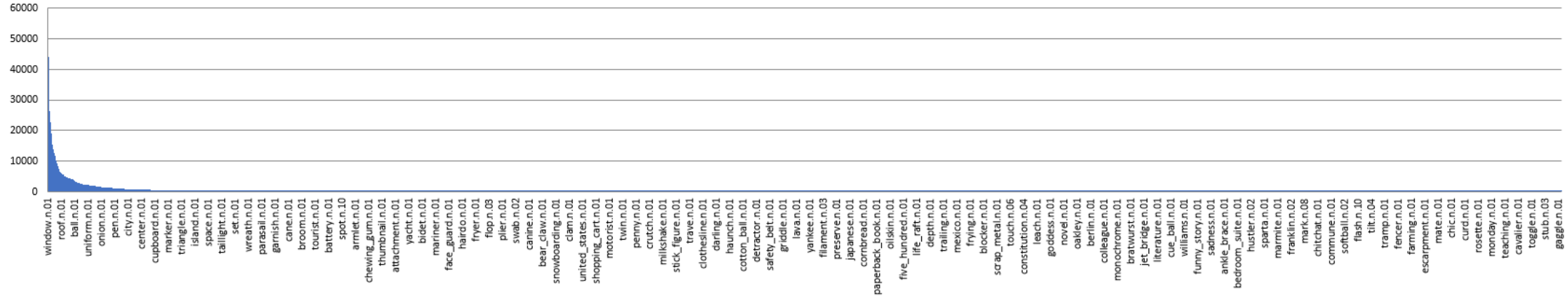
- MSCOCO: ~100 classes from 328K images
- Complex tasks, however, have fewer labeled data …

Long-tailed distribution on densely-labeled data

LVIS



Visual Genome



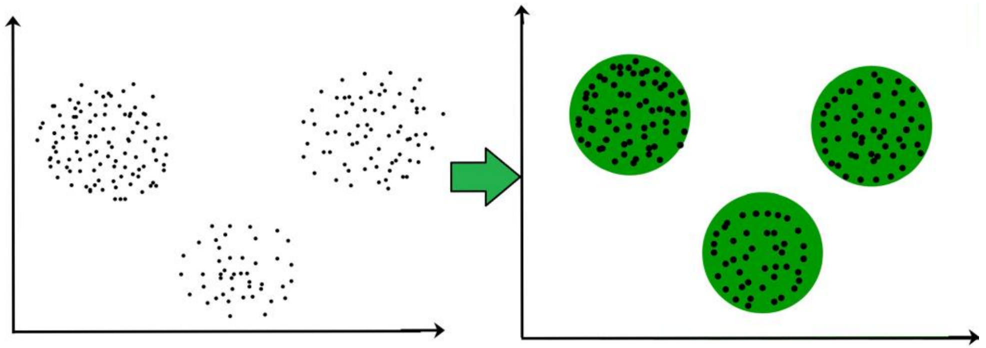
Self-supervised learning

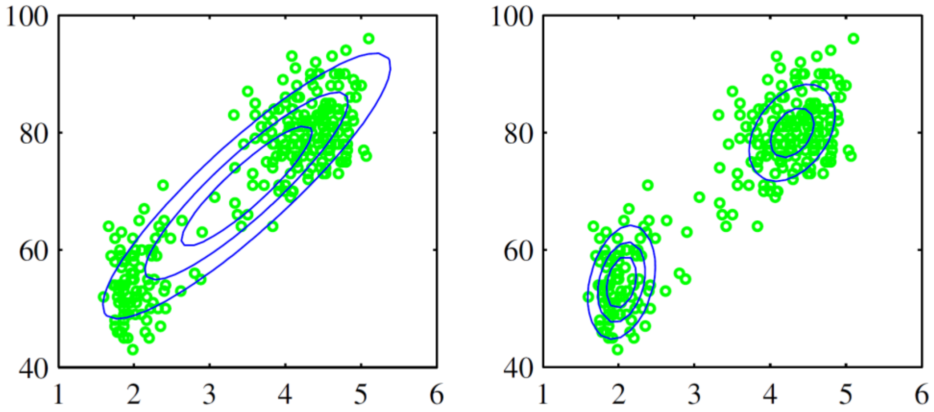
Can we learn a neural network from unlabeled data?

- What to learn?
- How to learn?

Traditional paradigms

- Unsupervised learning
  - Discover the structure (e.g., clusters, groups, or classes) of the data instances
  - Estimate the distribution/density p(x) of the data instances
  - Generative models





- Assumption: "similar" data should be grouped together

Deep self-supervised learning paradigms

- One more purpose:
  - Unsupervised feature or neural network learning

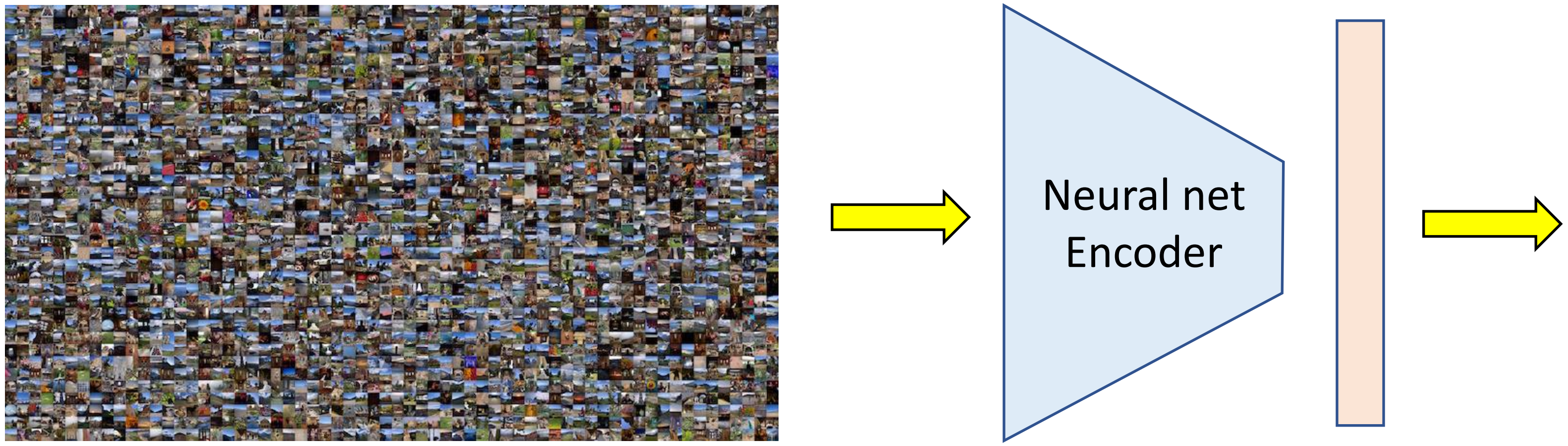
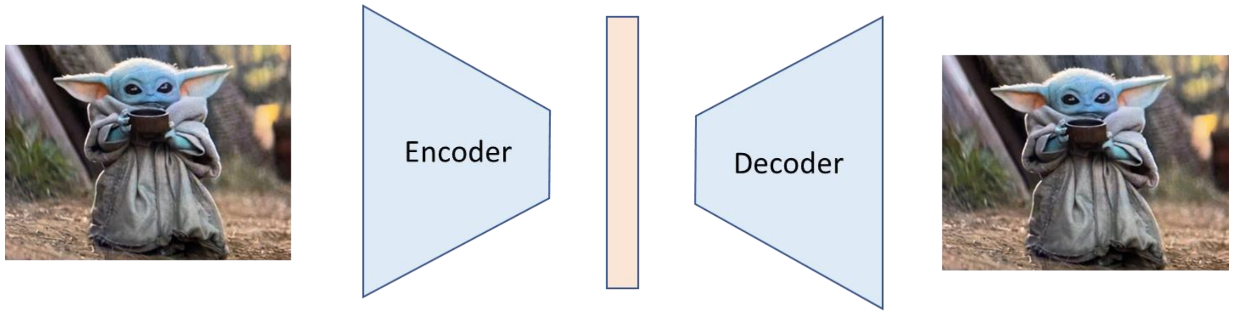


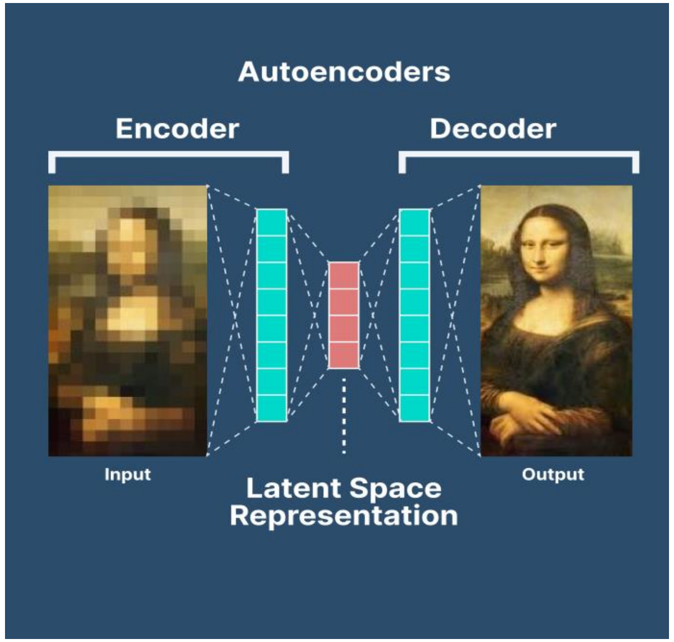
Image-to-image search

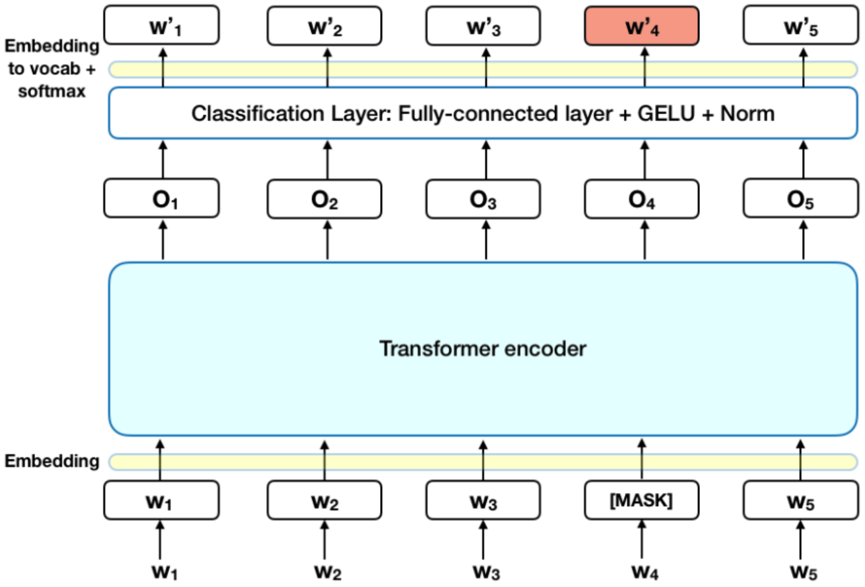
Initialize “downstream” tasks like for long-tailed, few-shot classification

Deep self-supervised learning paradigms

- Generative:
  - AE (auto-encoder), VAE
  - Masked or auto-regressive training
  - …

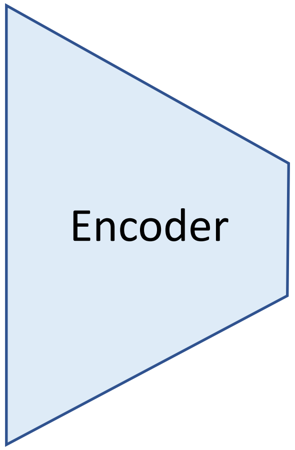




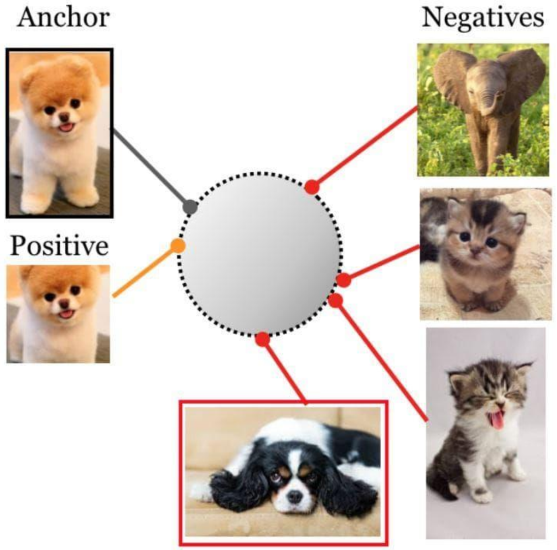


Deep self-supervised learning paradigms

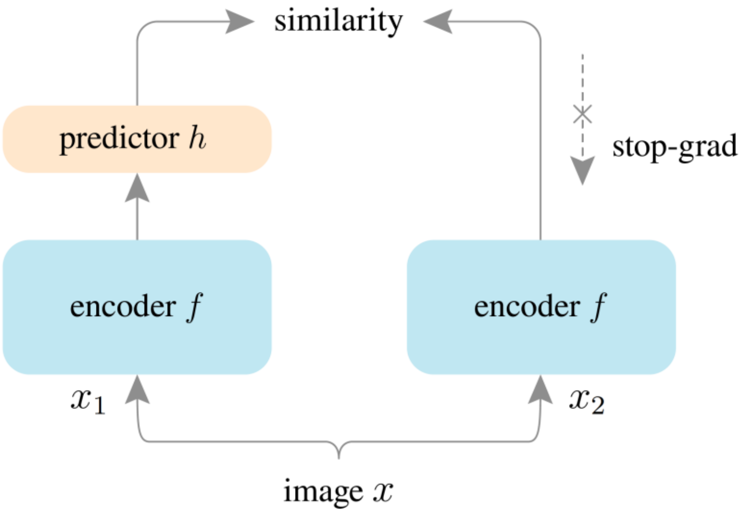
- Discriminative:
  - Contrastive learning
  - Similarity learning
  - Clustering based learning
  - …



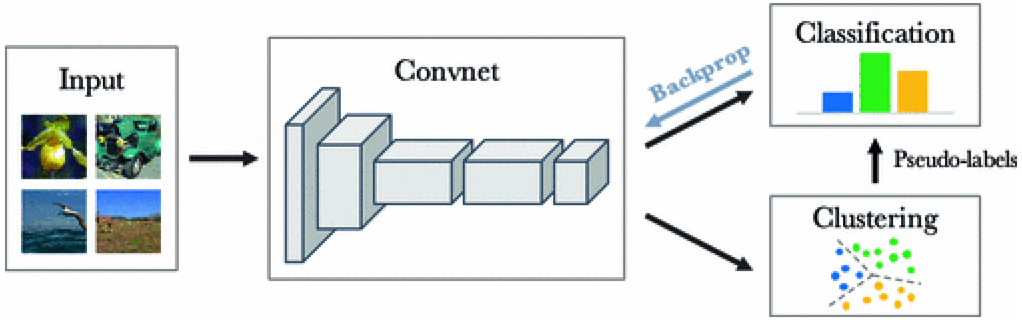
[Noroozi et al., 2016]



[Khosla et al., 2020]



[Chen et al., 2020]

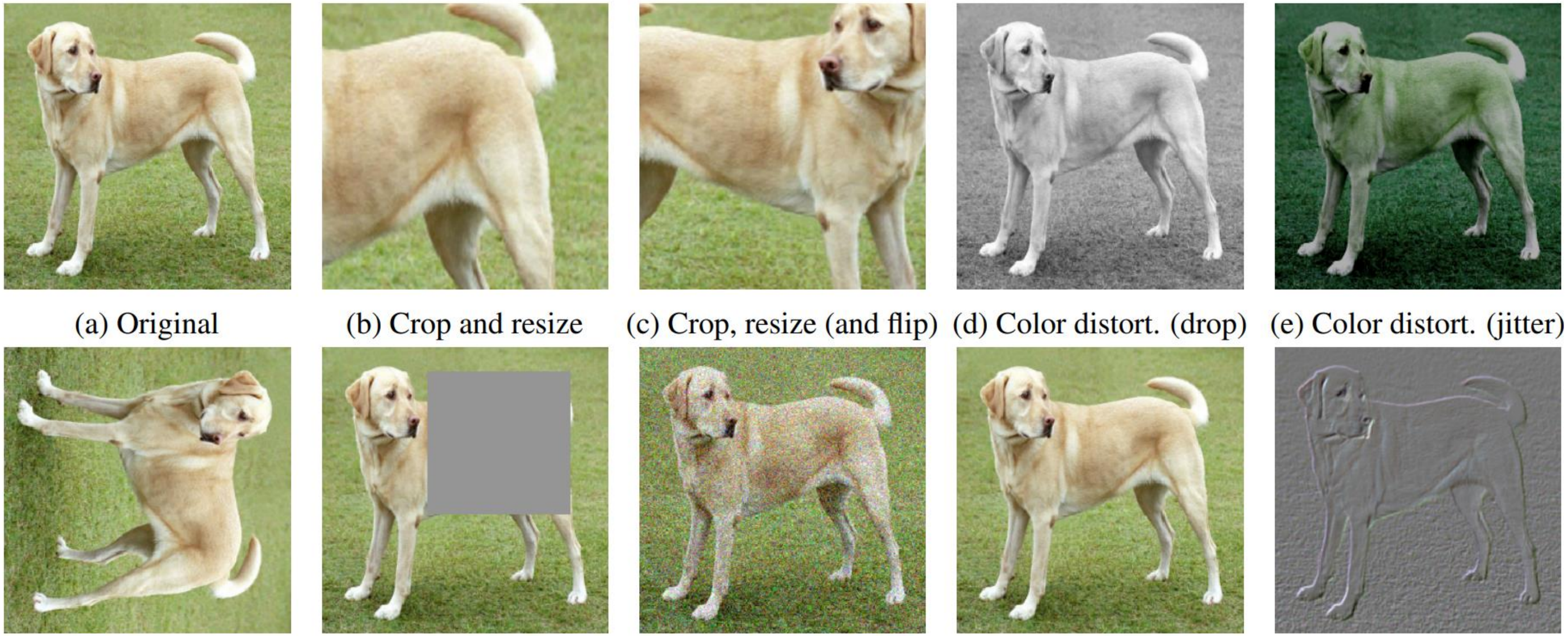


[Caron et al., 2018]

Contrastive learning

[Google AI blog, Advancing Self-Supervised and Semi-Supervised Learning with SimCLR]

Data augmentation



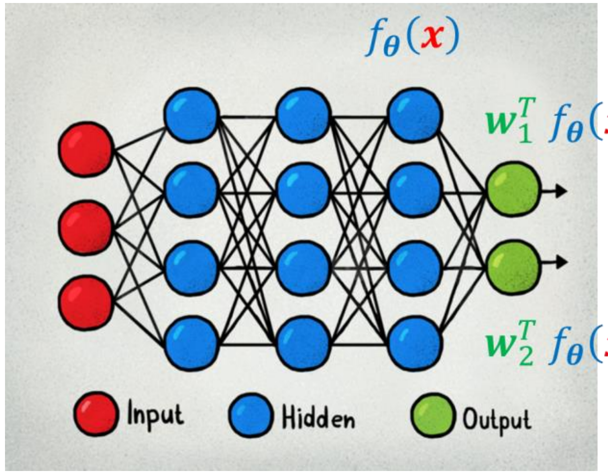
(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

[Chen et al., A Simple Framework for Contrastive Learning of Visual Representations, ICML 2020]

Results

Method	Architecture	ImageNet (Self-supervised) Top-1	Top-5
Supervised	ResNet50	76.5	-
CPC [38]	ResNet v2 101	48.7	73.6
InstDisc [12]	ResNet50	56.5	-
LA [50]	ResNet50	60.2	-
MoCo [14]	ResNet50	60.6	-
BigBiGAN [51]	ResNet50 (4x)	61.3	81.9
PCL [52]	ResNet50	61.5	-
SeLa [53]	ResNet50	61.5	84.0
PIRL [17]	ResNet50	63.6	-
CPCv2 [38]	ResNet50	63.8	85.3
PCLv2 [52]	ResNet50	67.6	-
SimCLR [15]	ResNet50	69.3	89.0
MoCov2 [47]	ResNet50	71.1	-
InfoMin Aug [19]	ResNet50	73.0	91.1
SwAV [13]	ResNet50	**75.3**	-

Linear evaluation:
- Freeze the feature extractor
- Only learn the last FC layer (i.e., linear classifier)

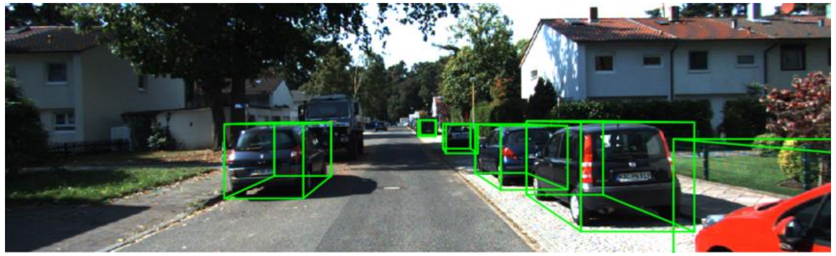


[Jaiswal et al., A Survey on Contrastive Self-supervised Learning, 2021]

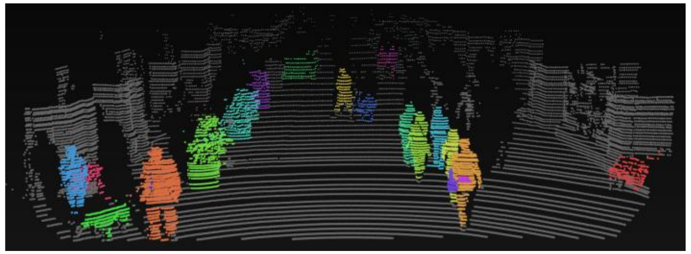
Unsupervised 3D object detection

Can we learn a 3D object detector from unlabeled data?

3D object detection



3D instance segmentation



- We focus on MOBLE traffic participants (e.g., cars, pedestrians, cyclists, etc.)
  - How can we discover mobile objects from unlabeled LiDAR data?

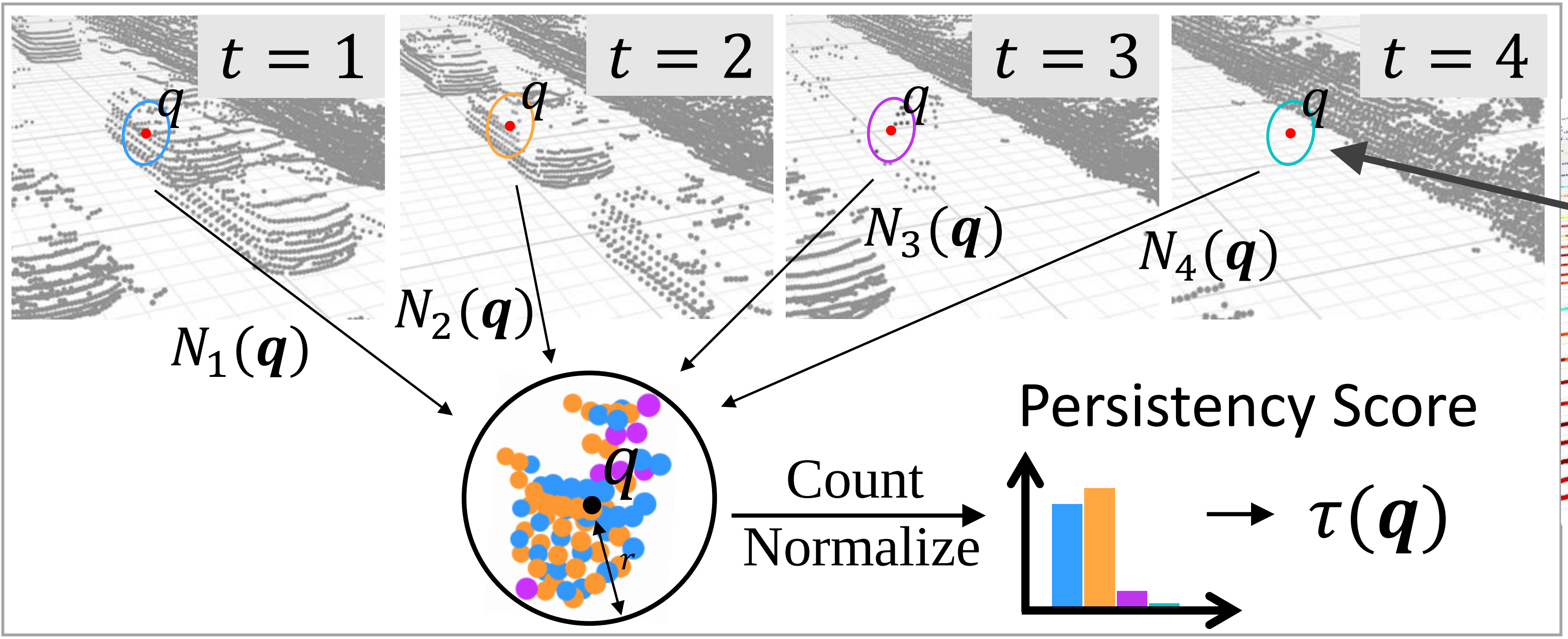
- **Simple heuristics:**
  - Mobile objects are unlikely to stay persistent at the same location over time.
  - We can easily collect multiple traversal data of repeated routes (many of us drive through the same routes every day).

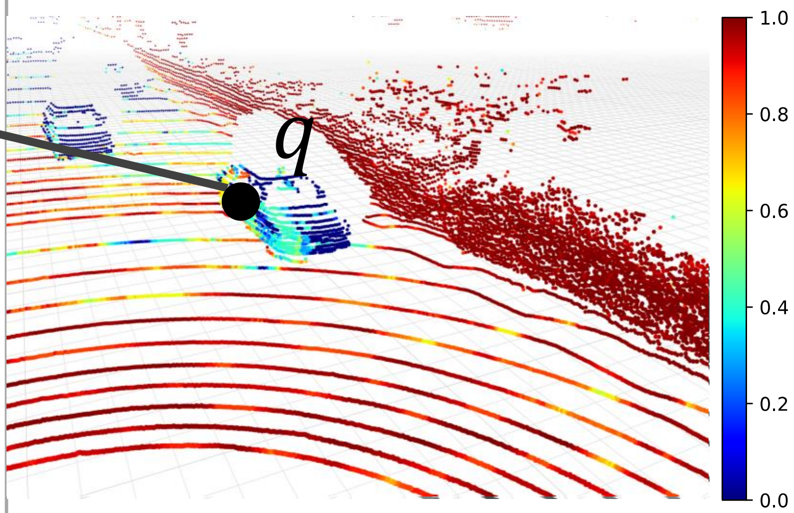
[Diaz-Ruiz et al., Ithaca365: Dataset and Driving Perception under Repeated and Challenging Weather Conditions, CVPR 2022]

Learning to Detect Mobile Objects Without Labels

Point clouds from past traversals

Current point cloud





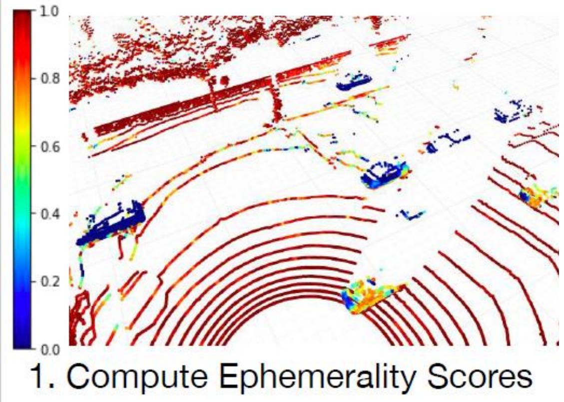
Entropy of the histogram quantifies the persistency!

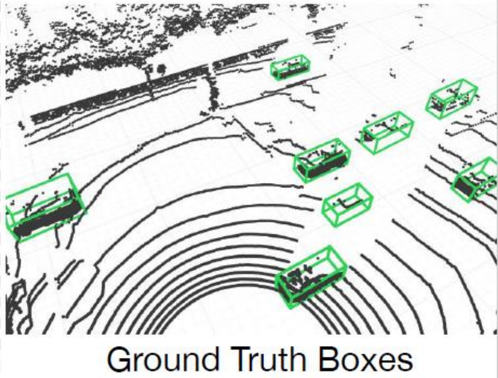
[You et al., Learning to Detect Mobile Objects from LiDAR Scans Without Labels, CVPR 2022]

Learning to Detect Mobile Objects Without Labels

- **Weak labels through simple heuristics:**
  - Clustering low persistent (ephemeral) objects
  - Fitting bounding boxes
  - Filtering out invalid bounding boxes (e.g., boxes under ground or flying)

- **Still not perfect:** (1) missing objects (2) incorrect boxes

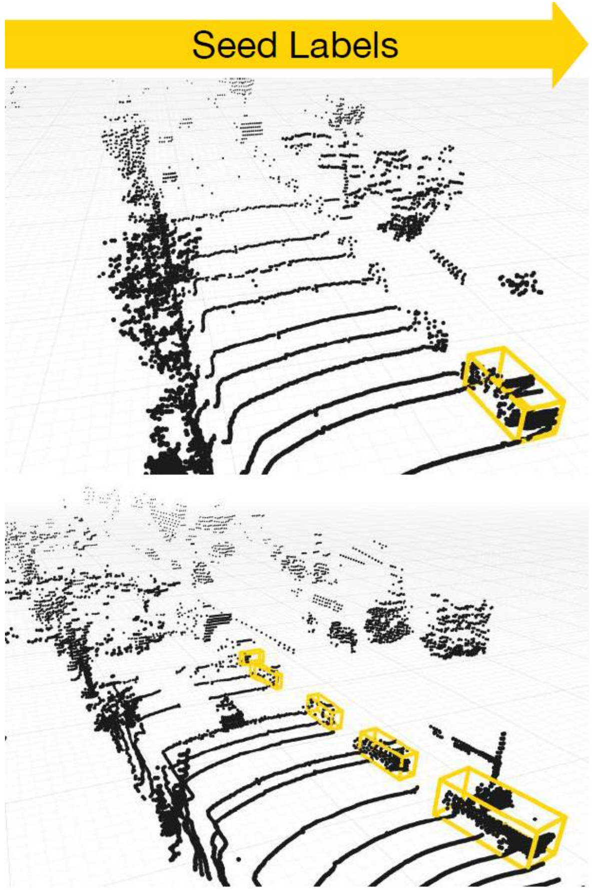


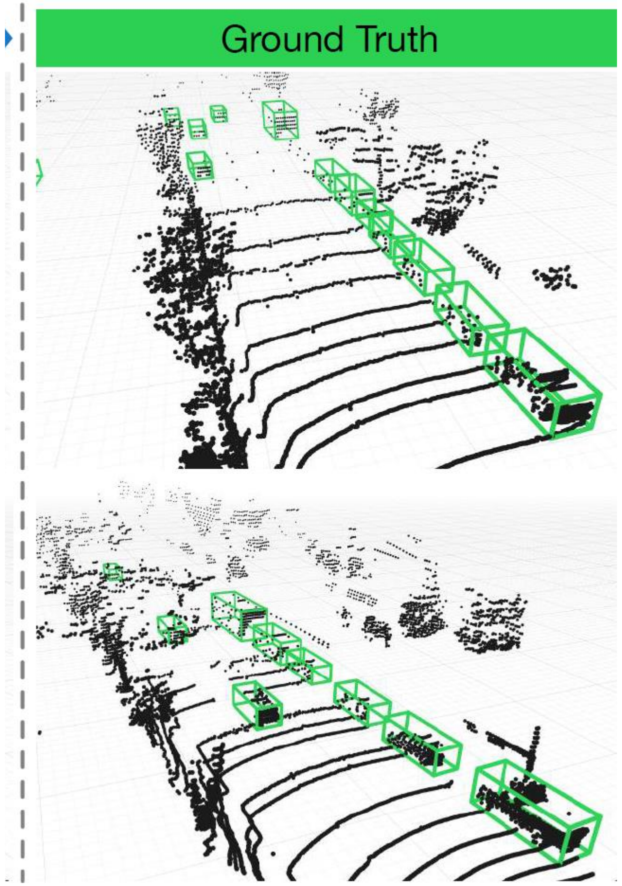


[You et al., Learning to Detect Mobile Objects from LiDAR Scans Without Labels, CVPR 2022]

Learning to Detect Mobile Objects Without Labels

Train a 3D detector and predict   Train another 3D detector and predict





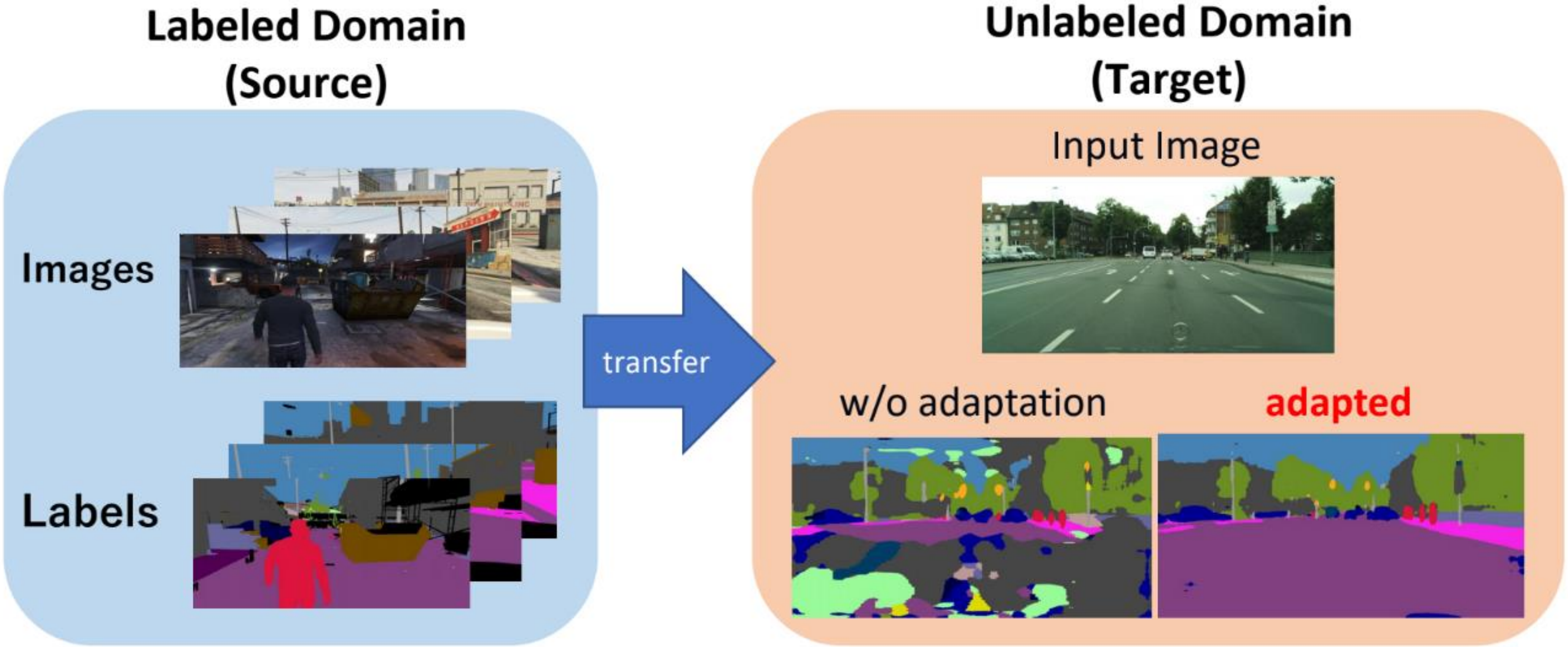
[You et al., Learning to Detect Mobile Objects from LiDAR Scans Without Labels, CVPR 2022]

Domain adaptation

Challenges-2: Domain Shifts

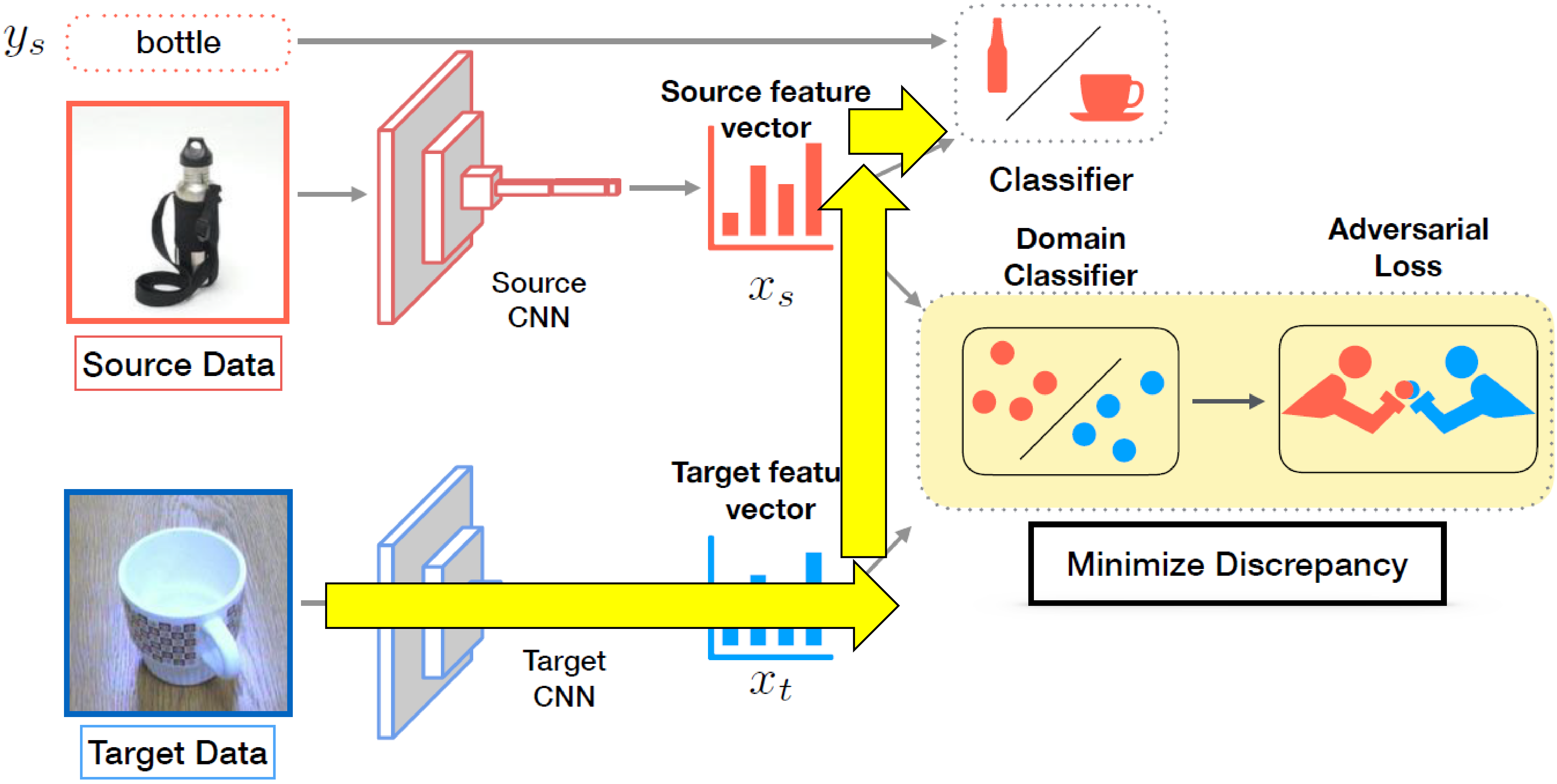
KITTI (Germany)	Argoverse (USA)	nuScenes (USA, Singapore)	Lyft (USA)	Waymo (USA)

Domain adaptation



[Saenko al., 2019]

Domain adversarial training



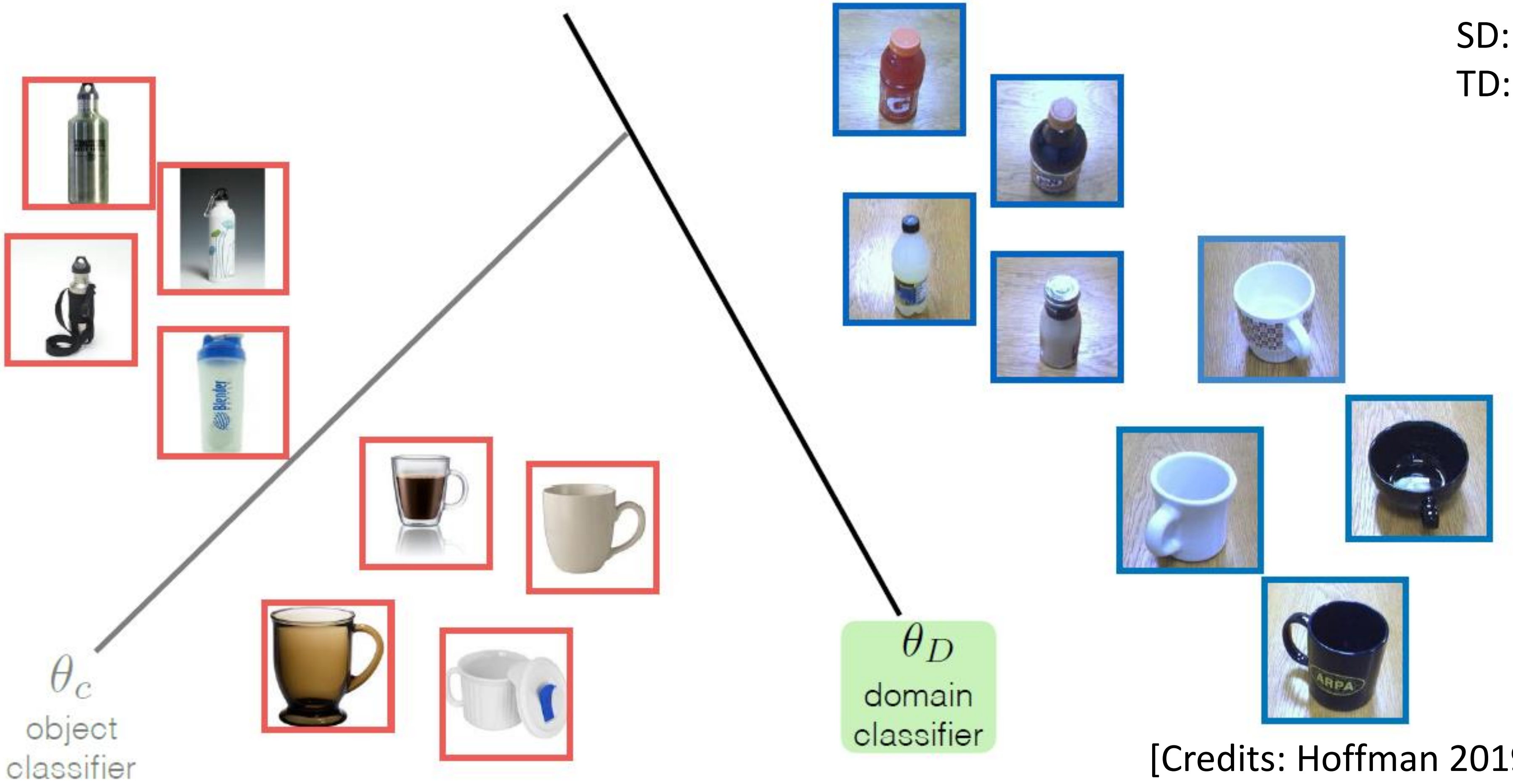
[Credits: Hoffman 2019 ICCV tutorial]

Domain adversarial training

$$\min_{\theta_D} \mathcal{L}_{\text{dom}}(\mathbf{X}_s, \mathbf{X}_t, \theta_R; \theta_D)$$

$$\min_{\theta_R} \mathcal{L}_{\text{rep}}(\mathbf{X}_s, \mathbf{X}_t, \theta_D; \theta_R)$$

Binary classification:
SD: +1
TD: -1

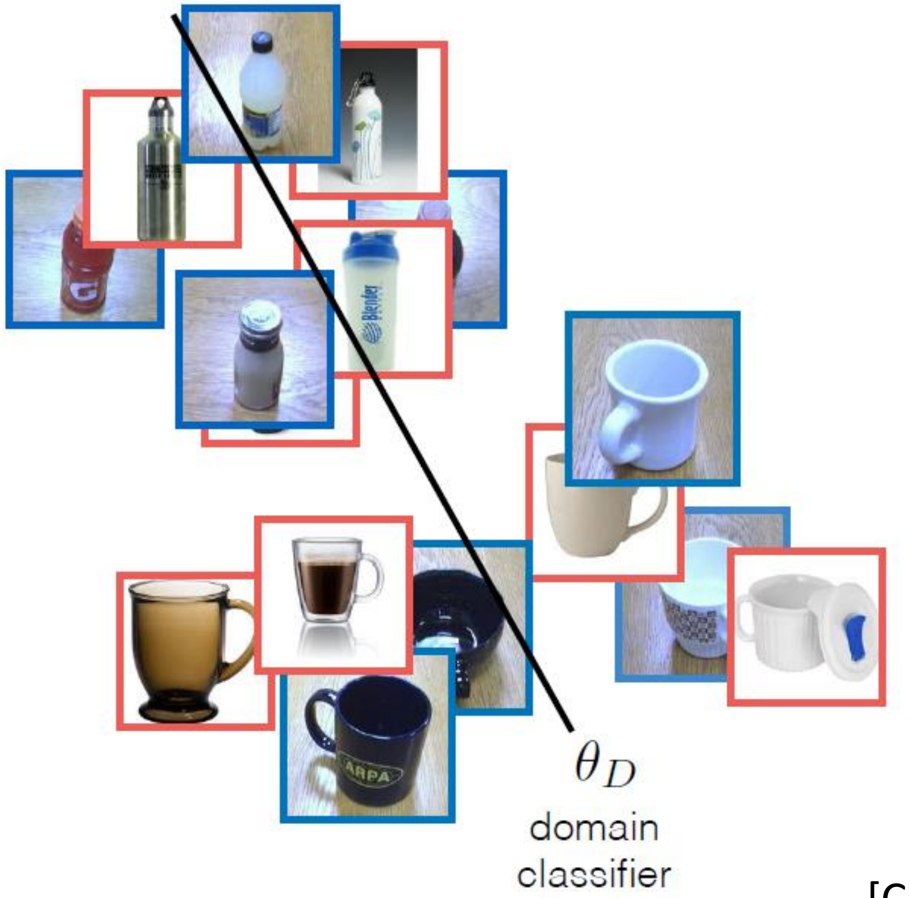


[Credits: Hoffman 2019 ICCV tutorial]

Domain adversarial training

$$\min_{\theta_D} \mathcal{L}_{\text{dom}}(\mathbf{X}_s, \mathbf{X}_t, \theta_R; \theta_D)$$

$$\min_{\theta_R} \mathcal{L}_{\text{rep}}(\mathbf{X}_s, \mathbf{X}_t, \theta_D; \theta_R)$$

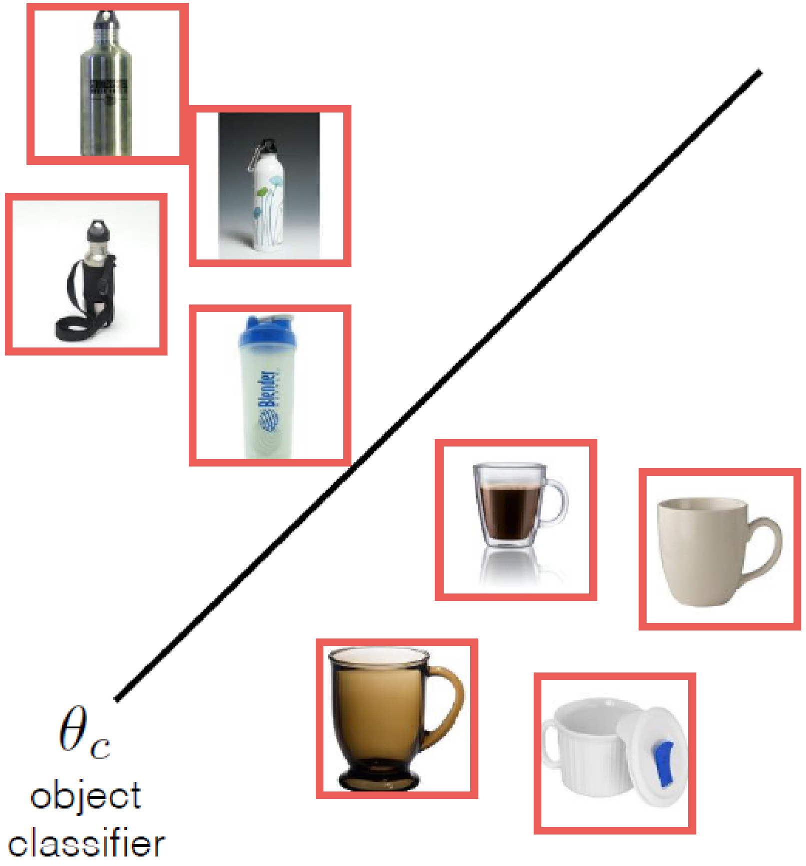


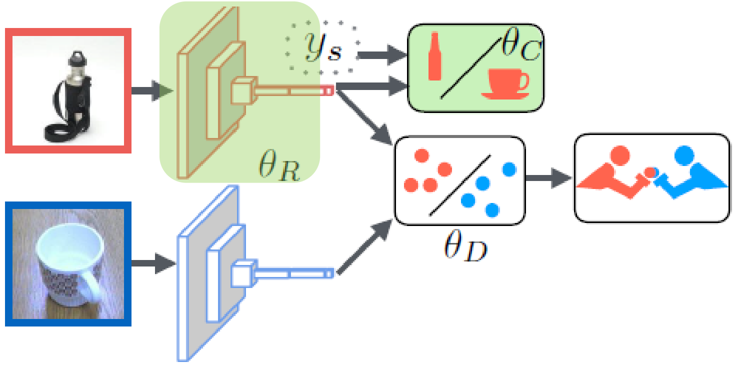
Binary classification:
SD: +1
TD: -1

[Credits: Hoffman 2019 ICCV tutorial]

Domain adversarial training

$$\min_{\theta_C,\theta_R} \mathcal{L}_{\text{cls}}(\mathbf{X}_s, \mathbf{Y}_s; \theta_C, \theta_R)$$

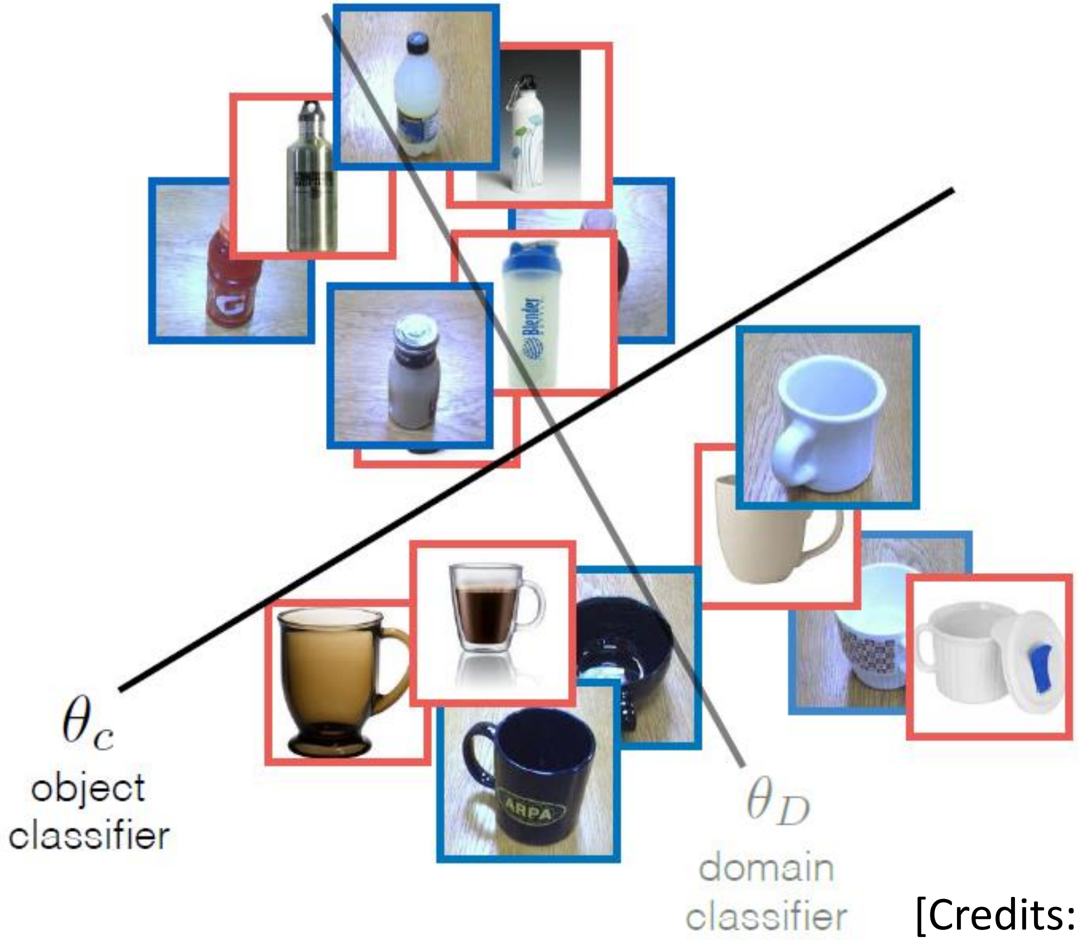




[Credits: Hoffman 2019 ICCV tutorial]

Domain adversarial training

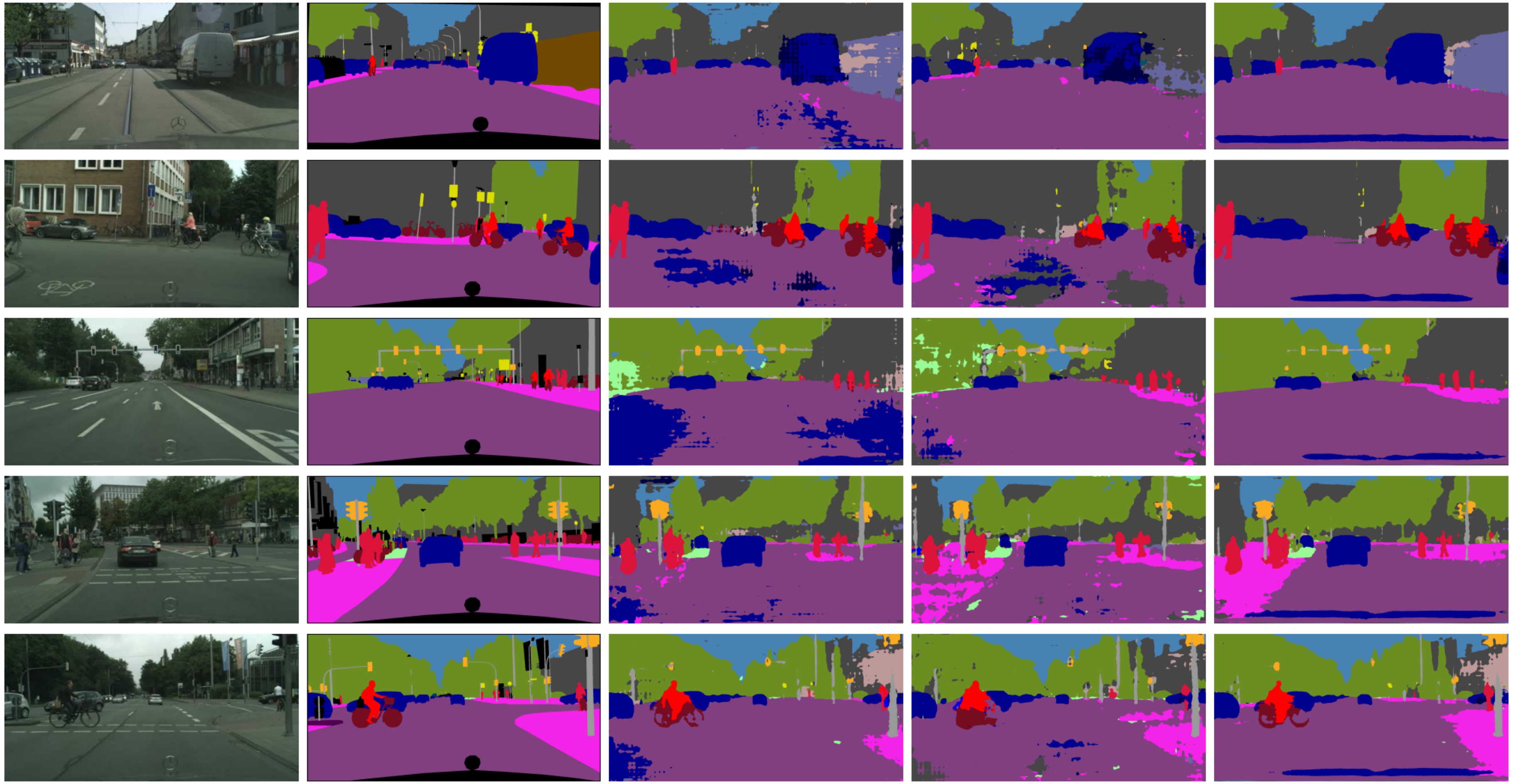
$$\min_{\theta_D} \mathcal{L}_{\text{dom}}(\mathbf{X}_s, \mathbf{X}_t, \theta_R; \theta_D) \qquad \min_{\theta_R} \mathcal{L}_{\text{rep}}(\mathbf{X}_s, \mathbf{X}_t, \theta_D; \theta_R) \qquad \min_{\theta_C, \theta_R} \mathcal{L}_{\text{cls}}(\mathbf{X}_s, \mathbf{Y}_s; \theta_C, \theta_R)$$



The devils in the details!
Be aware of trivial solutions
or poor convergence

[Credits: Hoffman 2019 ICCV tutorial]

Example results



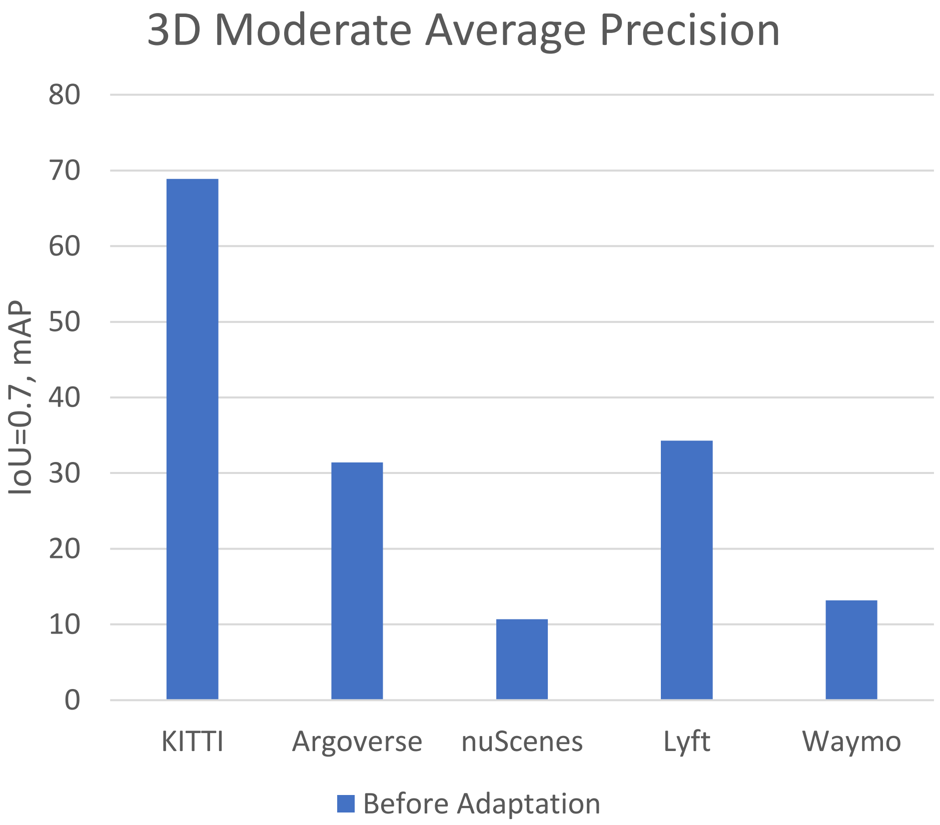
Target Image | Ground Truth | Before Adaptation | Feature Adaptation | Ours

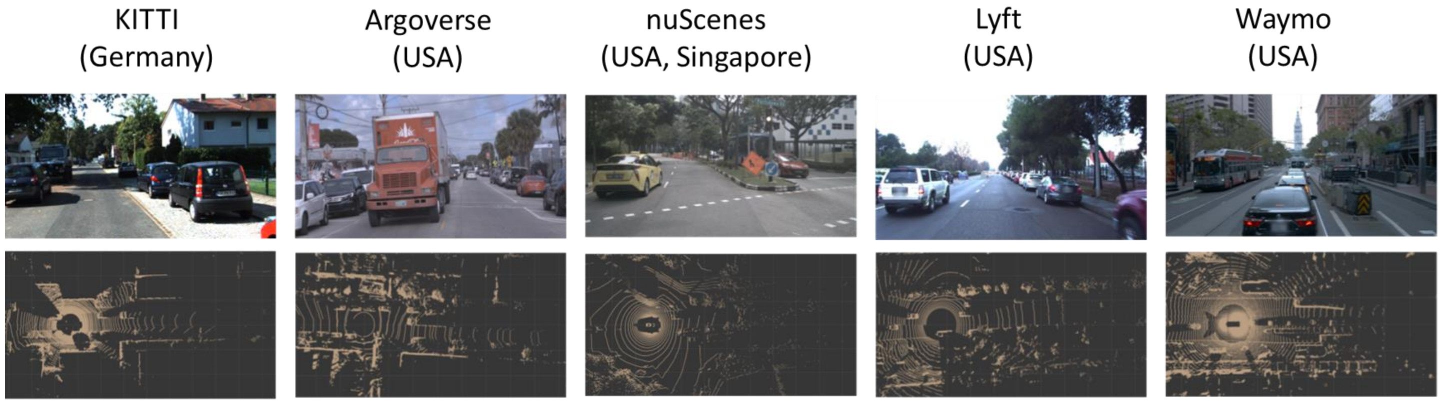
[Tsai et al., CVPR 2018]

3D domain adaptation

Domain adaptation of 3D perception

- Train 3D object detectors on different **source** datasets and test it on KITTI

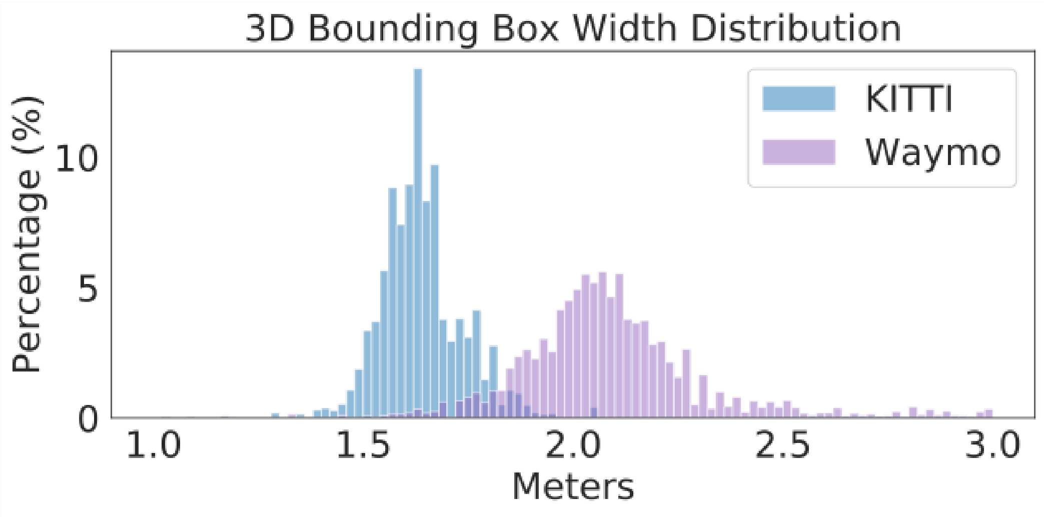




[Wang et al., Train in Germany, Test in The USA: Making 3D Object Detectors Generalize, CVPR 2020]

Domain adaptation of 3D perception

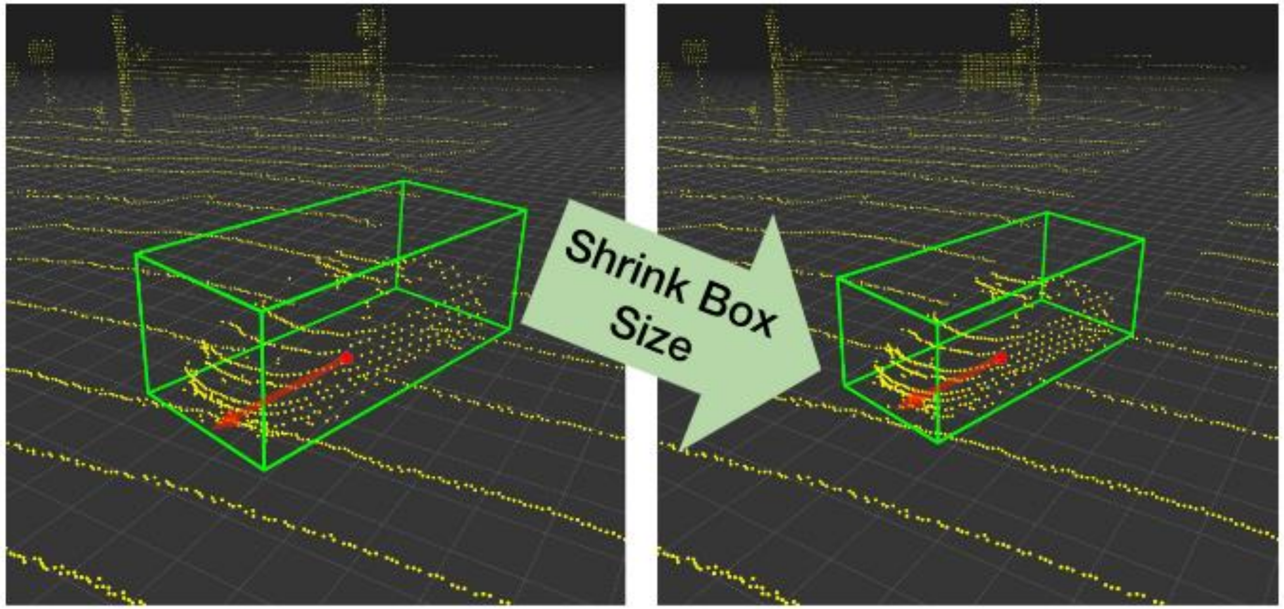
- Sizes of the cars are different at different geo-locations



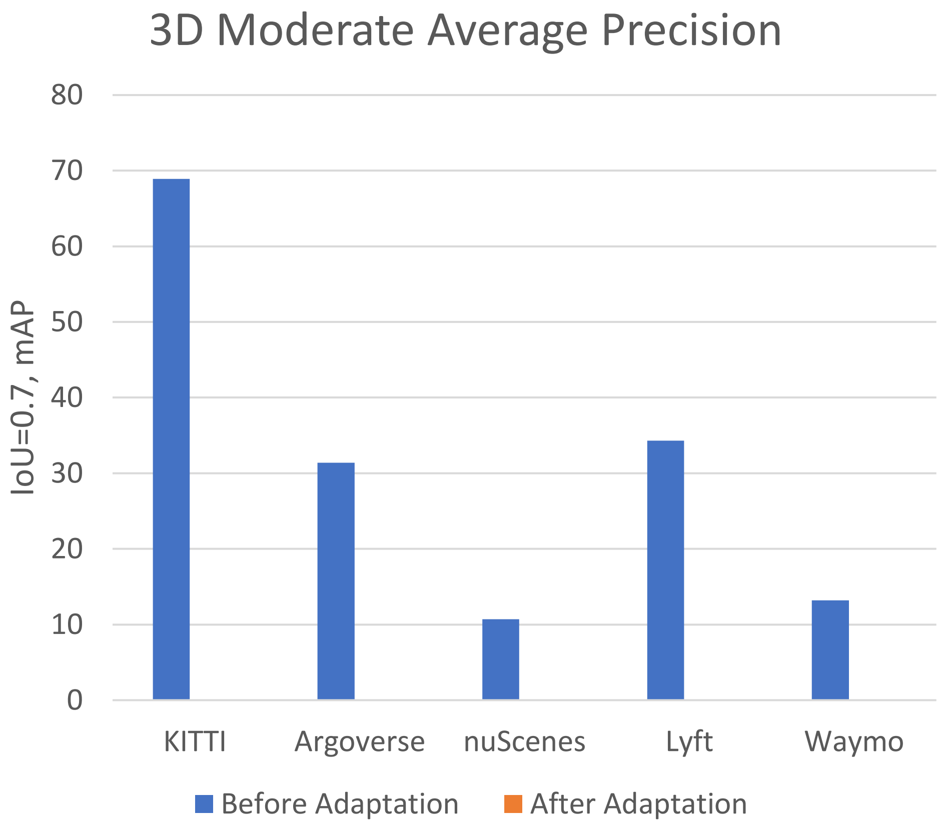
- Learned 3D object detectors will "memorize" the sizes!

[Wang et al., Train in Germany, Test in The USA: Making 3D Object Detectors Generalize, CVPR 2020]

Domain adaptation of 3D perception

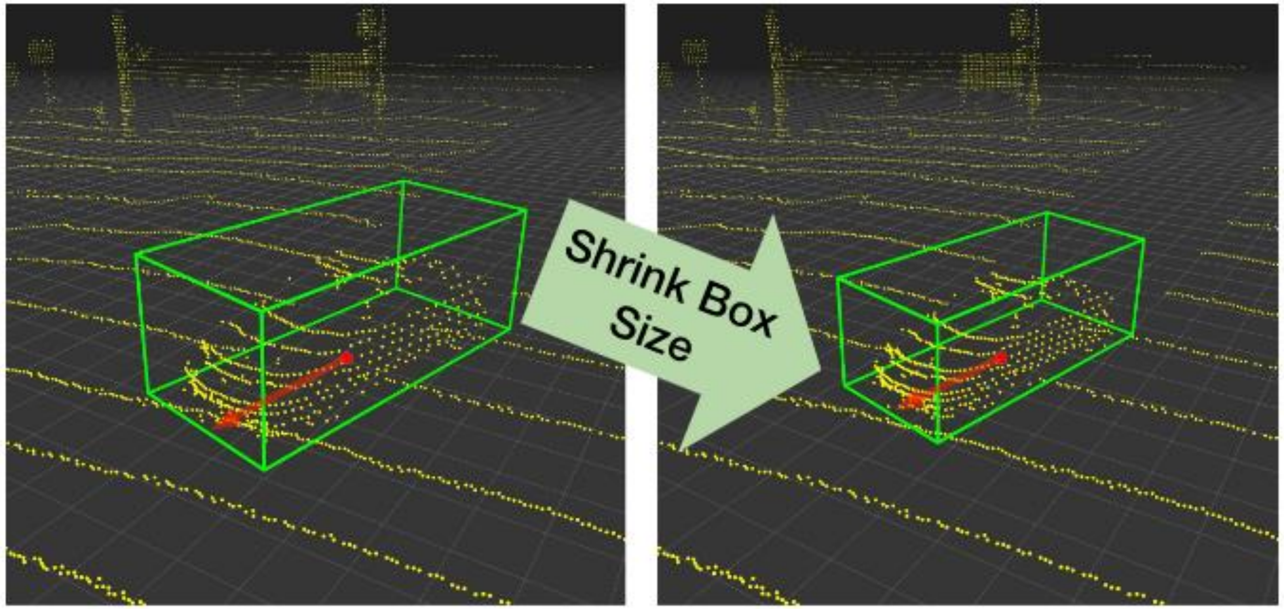


Resize points and labels in the source domain
+
Re-training



[Wang et al., Train in Germany, Test in The USA: Making 3D Object Detectors Generalize, CVPR 2020]

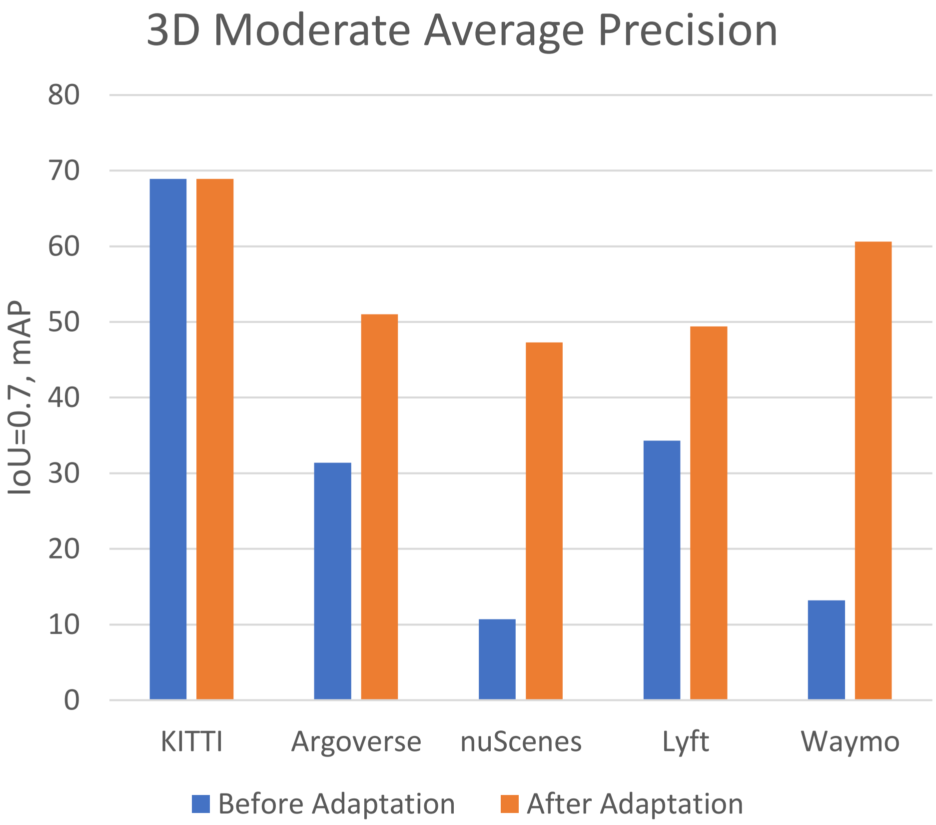
Domain adaptation of 3D perception



Resize points and labels in the source domain

+

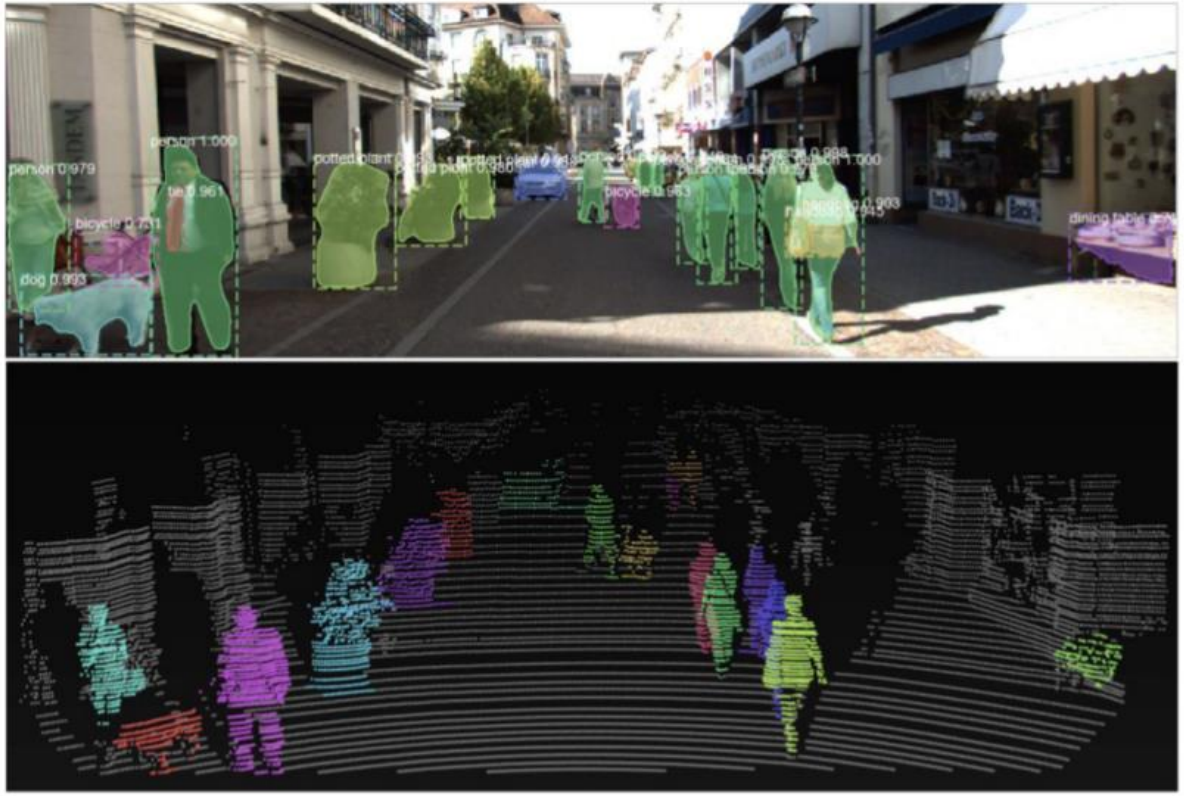
Re-training



[Wang et al., Train in Germany, Test in The USA: Making 3D Object Detectors Generalize, CVPR 2020]

Summary

- (Brief and narrow) introduction to computer vision
- Basic deep learning blocks for computer vision
  - Convolutional neural nets
  - Visual transformers
- Applications:
  - 2D Recognition
  - 3D Perception for autonomous driving
  - 2D Generation
- Practical problems:
  - Insufficient (labeled) data
  - Domain shifts



Take home

- Good tutorials online:
  - CVPR 2017-2022, ECCV 2018-2020, ICCV 2017-2021 [search tutorial or workshop]
  - ICML/NeurIPS/ICLR 2018-2021 [search tutorial or workshop]

- Good framework:
  - PyTorch: Torchvision
  - PyTorch: Detectron2

- Good source code:
  - Papers with code: https://paperswithcode.com/

Thank you!