SELF-DRIVING CARS
PERCEPTION

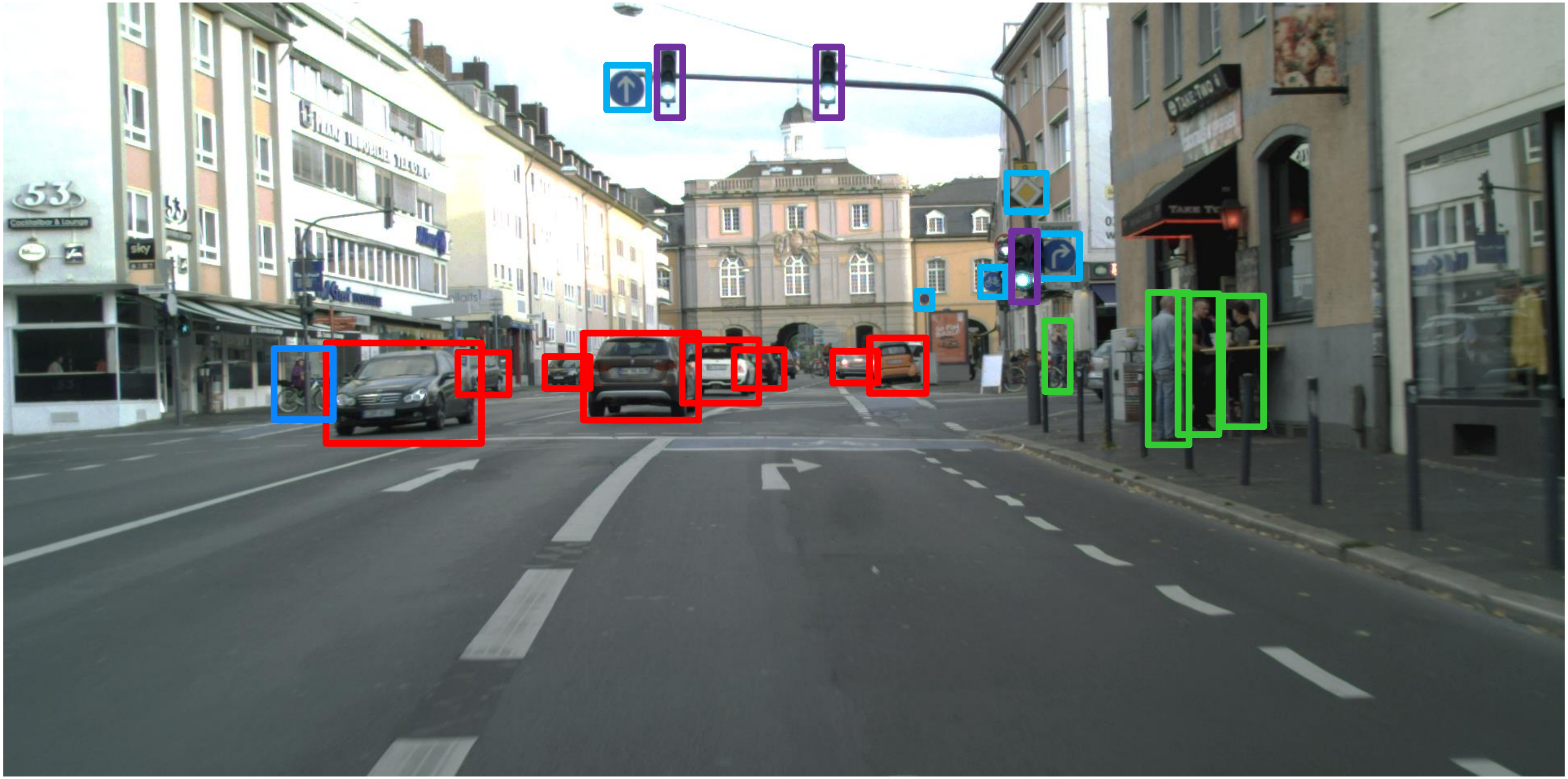
Photogrammetry & Robotics Lab

Perception for Self-Driving Cars
Vision-based Approaches

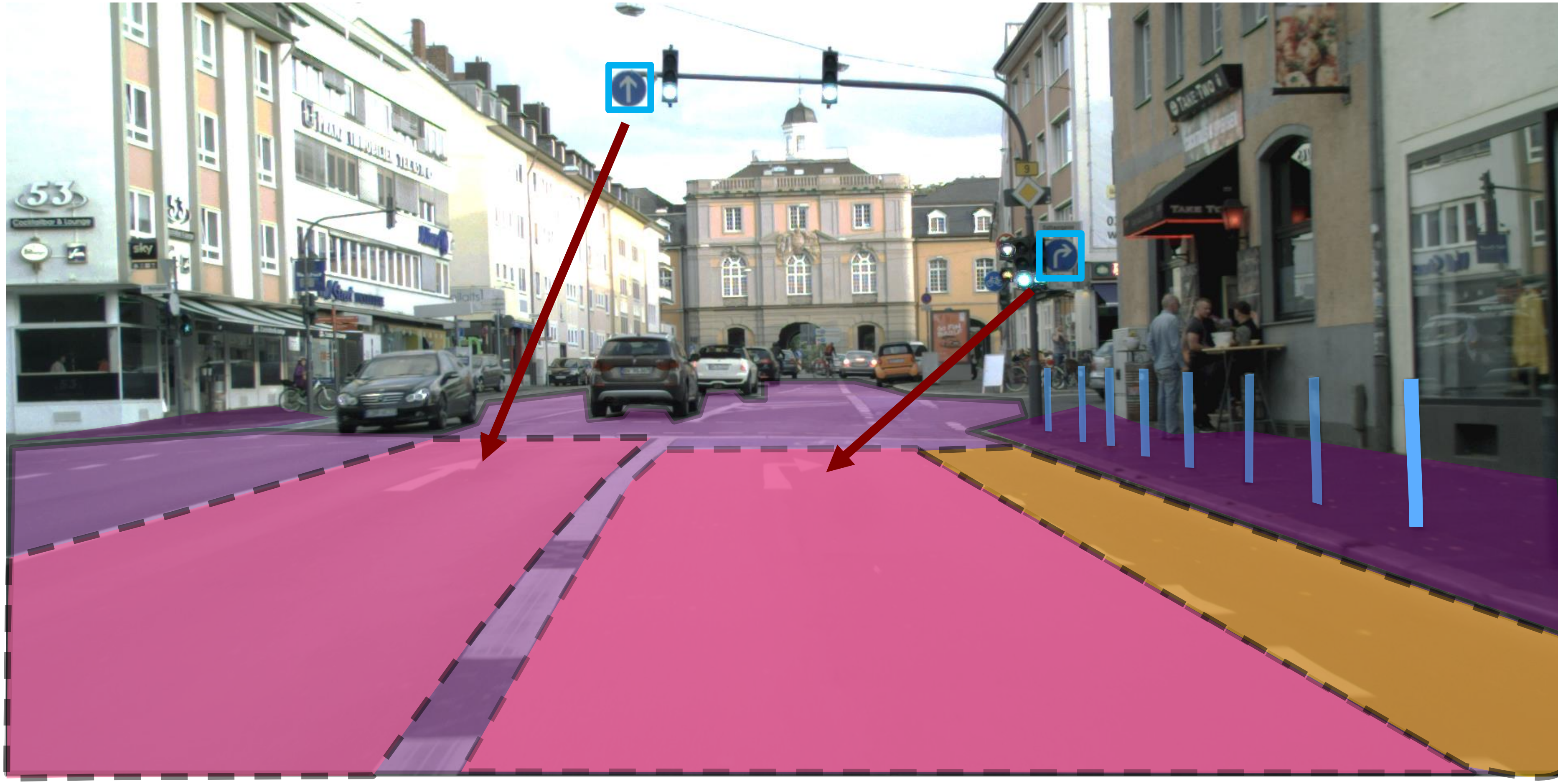
Jens Behley

Part of the Course: Techniques for Self-Driving Cars by
C. Stachniss, J. Behley, N. Chebrolu, B. Mersch, L. Peters, I. Bogoslavskyi

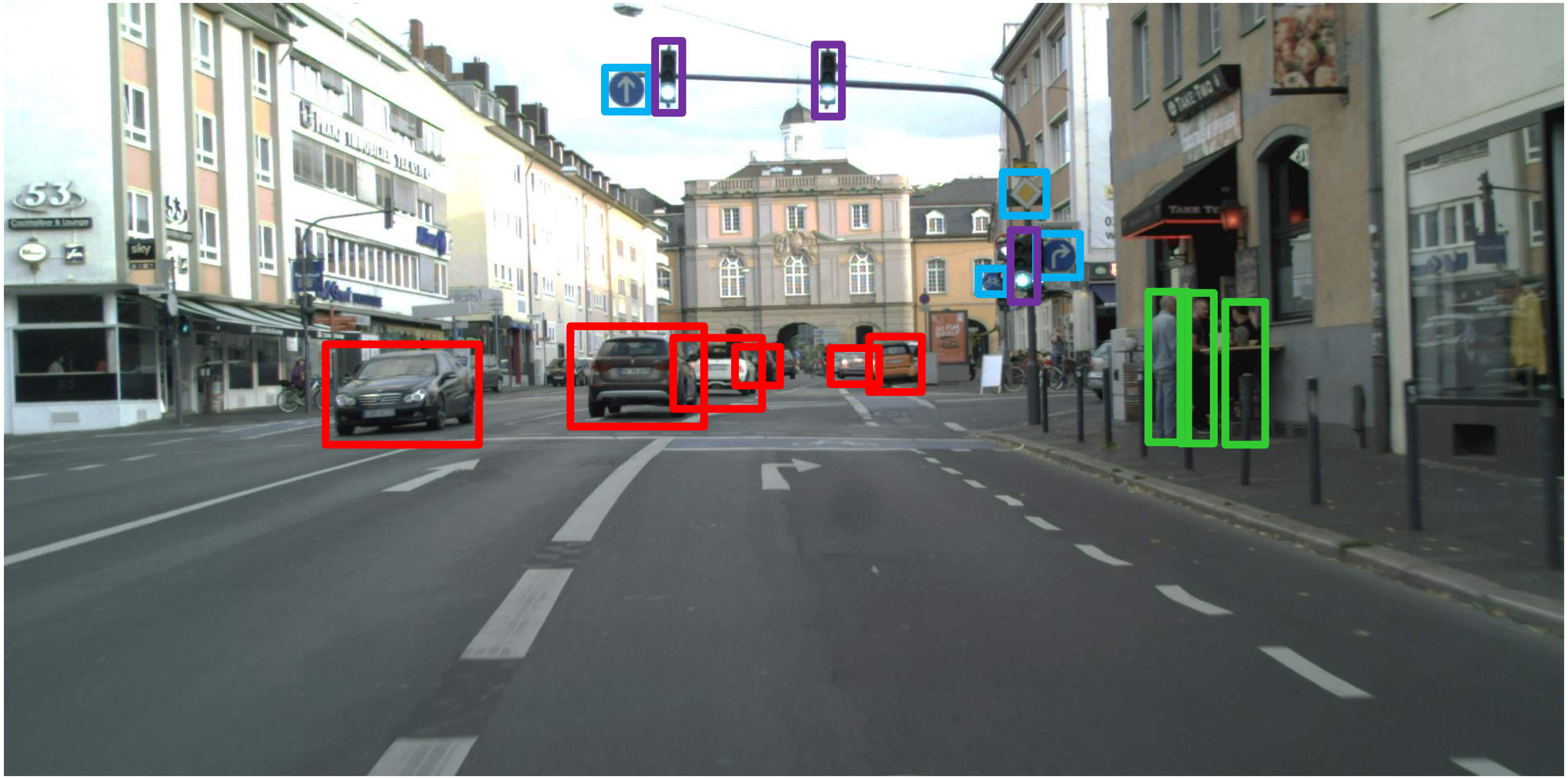
Which information is needed?



Which information is needed?



Content of this lecture



- Overview of perception stack & tasks
- Dive into camera-based perception

Perception Suite

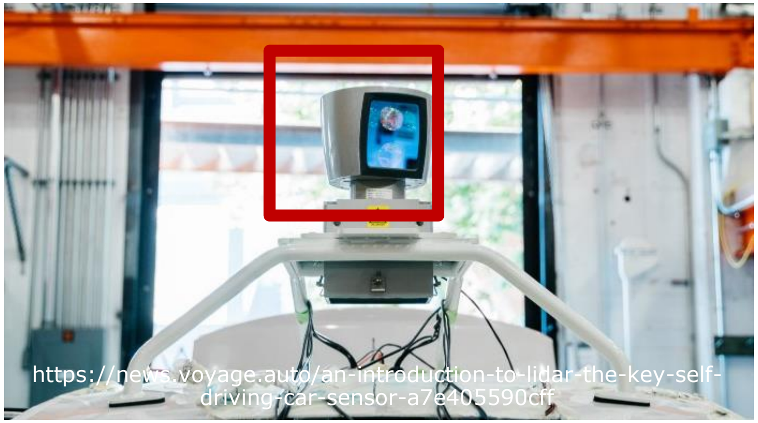
https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff

- Common Perception Suite
  - (Stereo) Camera
  - LiDAR
  - Radar

(Stereo) Camera

- Pro: cheap, high resolution, color
- Con: strongly affected by illumination, needs additional light at night

LiDAR Sensors



- Pro: Independent of illumination, Precise distance measurements
- Con: Expensive, mid resolution

Radar

- Pro: Position + velocity information, Matured technology
- Con: low resolution/sparse

Other Sensors

- Ultrasonic (near range)
- GPS
- Inertial Measurement Unit (IMU)
- Odometer

Complementing Modalities

- Many sensors are already build into cars (camera, RADAR, ultra-sound sensors, …)
- Not a single sensor will enable self-driving, but combination of sensors
- **But:** **Sensor fusion** essential to integrate information from multiple sensors

Perception Suite

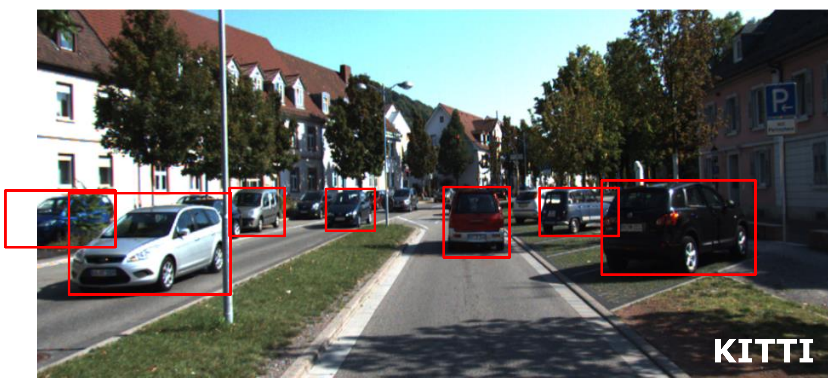
https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff

- Common Perception Suite
  - (Stereo) Camera
  - LiDAR
  - Radar
  - GPS+IMU

Perception Tasks

Classification

Semantic Segmentation

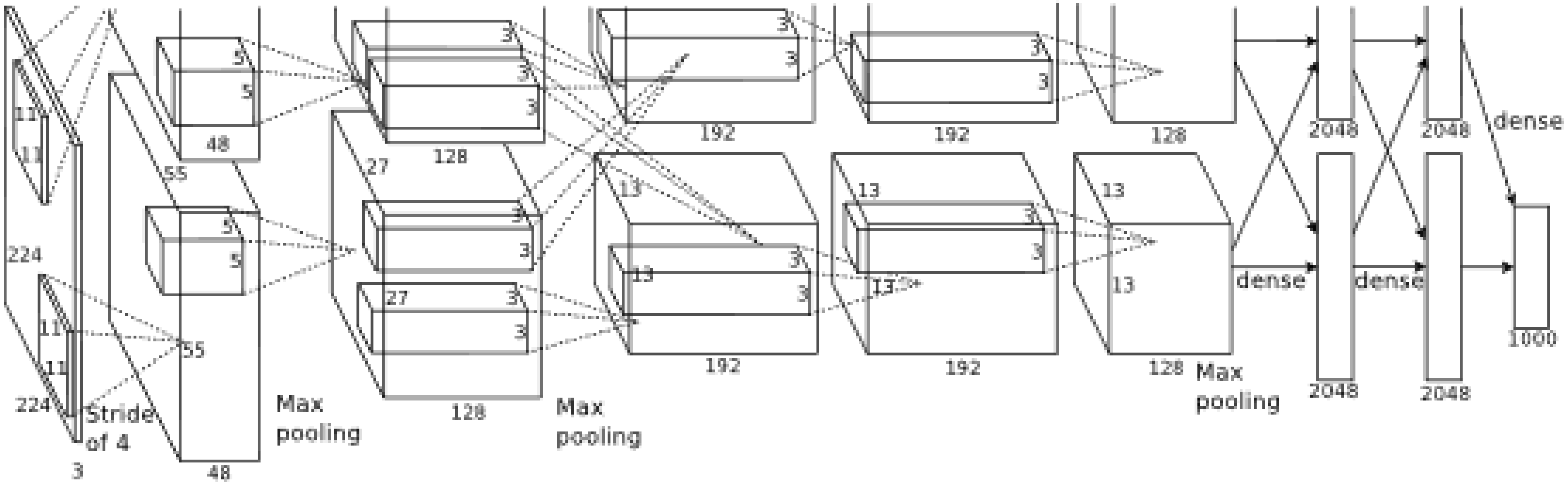


Detection

Panoptic Segmentation

[Geiger, 2012] [Cordts, 2014]

Convolution Neural Network



- Since 2012 success in ImageNet Challenge, the basis for most image-based perception tasks nowadays
- Here: high-level overview of CNNs

[Krizhevsky, 2012]

Why are CNNs successful now?

- Several reasons made progress possible:
  1. Availability of large-scale data (ImageNet, etc.)
  2. Availability of compute capabilities (GPUs)
  3. **Availability of code (and frameworks)!**

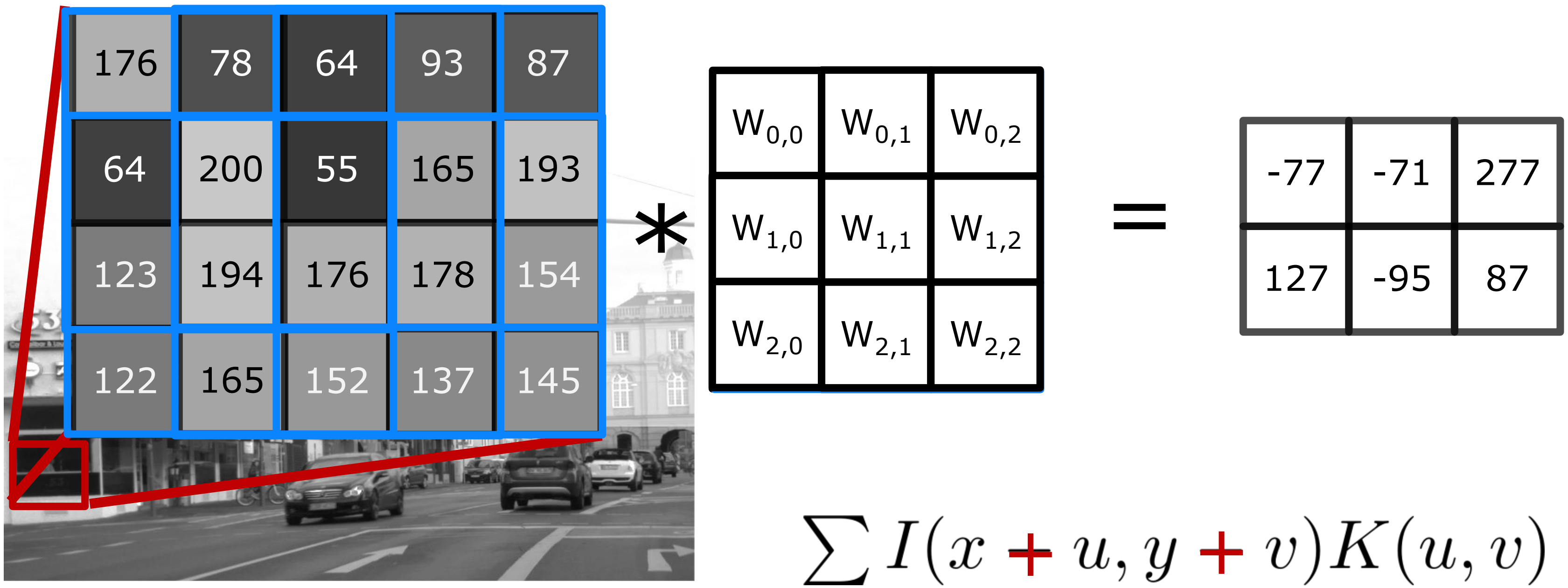
- Implementation for most paper available
- Many frameworks made it simple to build and train networks (Caffe, Theano, Torch, etc.)

Deep Learning Frameworks

- All operations must be implemented using GPU
- DL Frameworks available implementing the aforementioned operations (and many more)

TensorFlow PyTorch

Convolution Neural Network



$$\sum I(x + u, y + v)K(u, v)$$

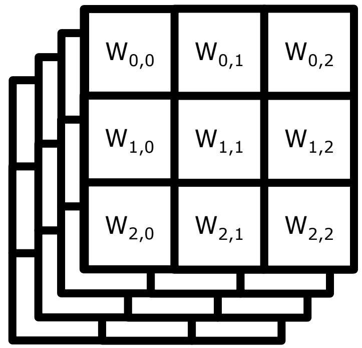
- Convolution “slides” kernel/filter K over image I
- Trivia: Most DL frameworks use cross-correlation instead

Convolution Neural Network

W

H

$H \times W$



$3 \times 3 \times$ **4**

$H \times W \times$ **1**
(with zero padding)

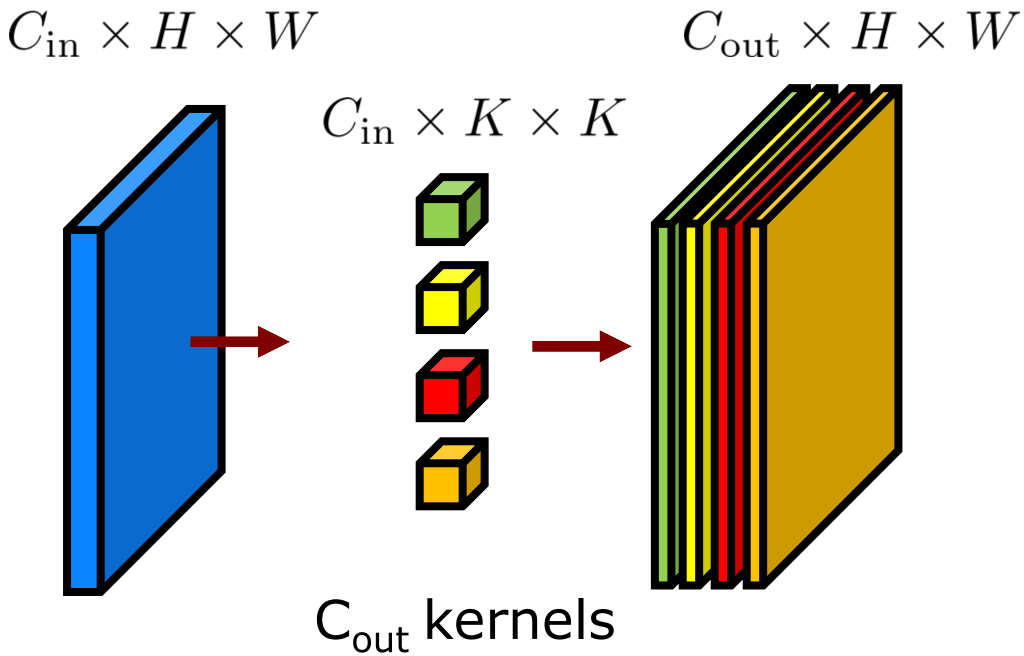
- For multi-channel input convolutional kernel has also as many channels
- Produces still one activation map

Channel-first vs. Channel-last

- Organizing tensors in different ways possible
- Main conventions:
  - Channel-first (PyTorch): $C \times H \times W$
  - Channel-last (Tensorflow): $H \times W \times C$

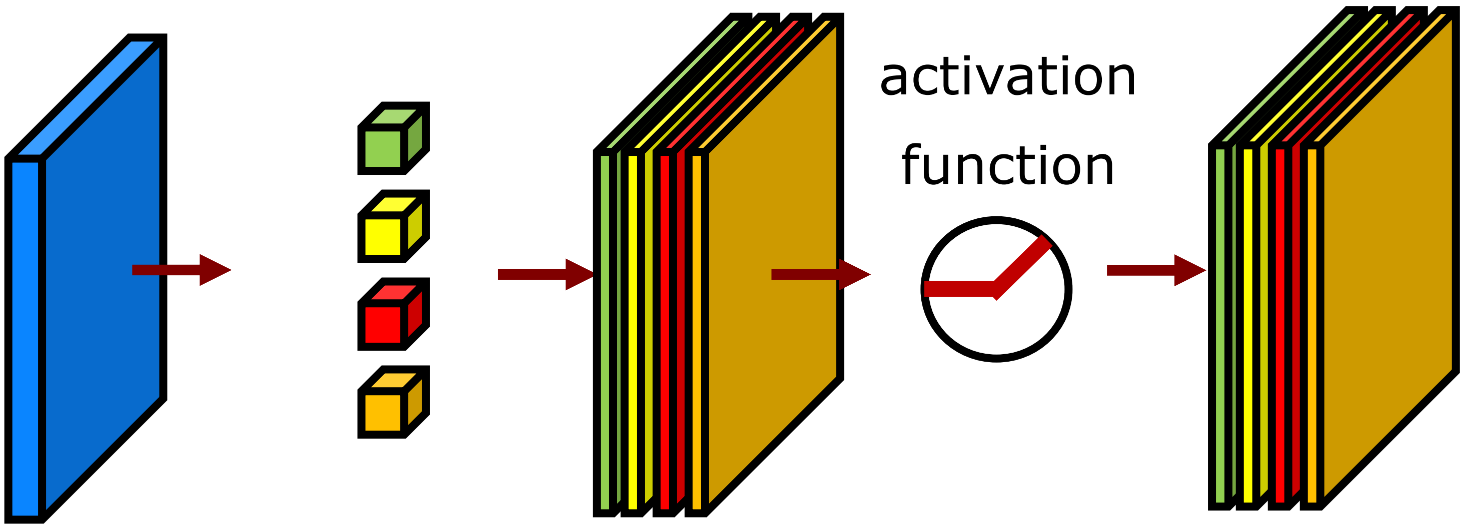
- We stick now to the channel-first convention…

Convolutional Layer



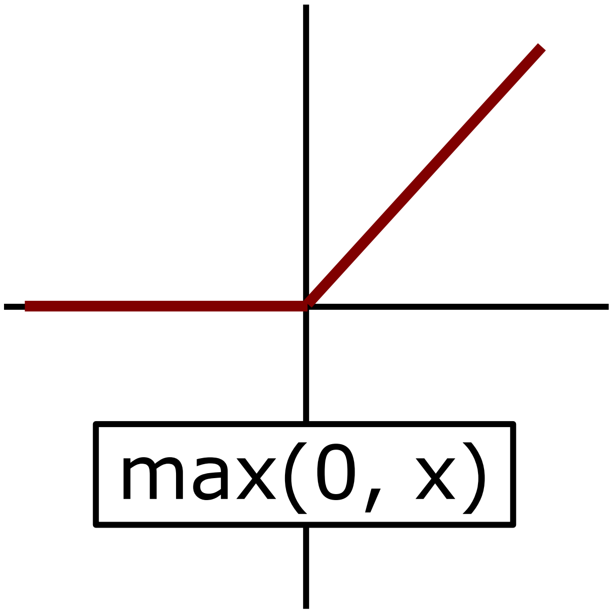
- Use multiple kernels to produce C_{out} maps

ConvLayer + Activation Function

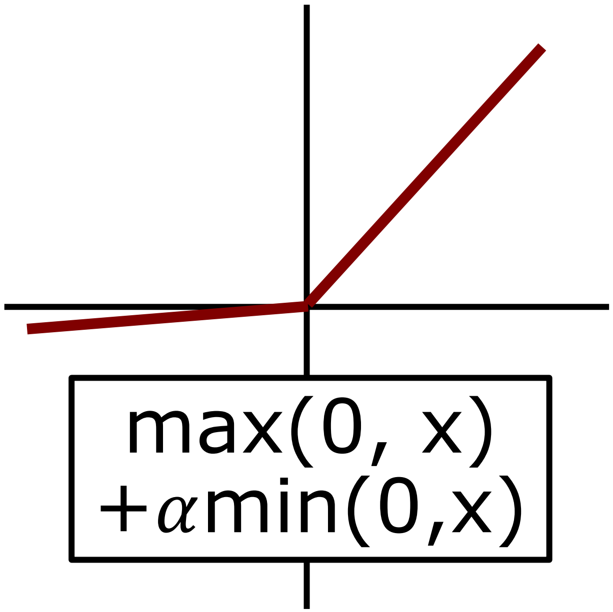


- Activation function (such as ReLU) applied after each convolutional layer
- Usually only implicit in the graphical representation

Activation Function



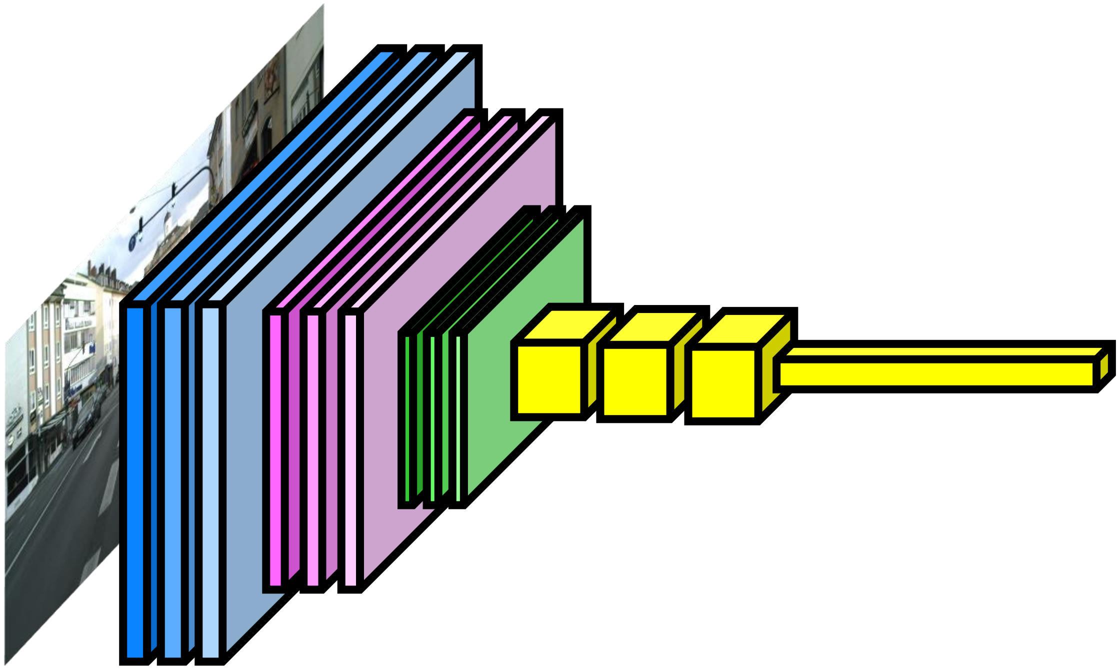
ReLU



Leaky ReLU

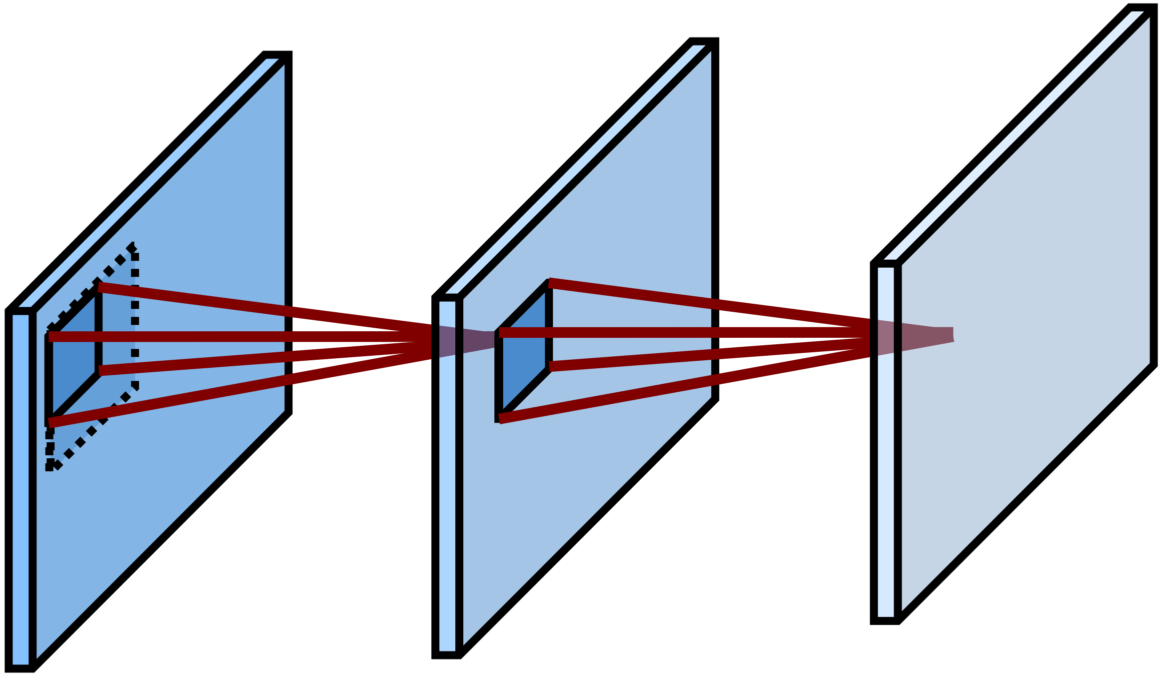
- Non-linear Activation functions
- Popular: **Rectified Linear Unit (ReLU)**
  + variants, e.g., Leaky ReLU

Convolution Neural Network



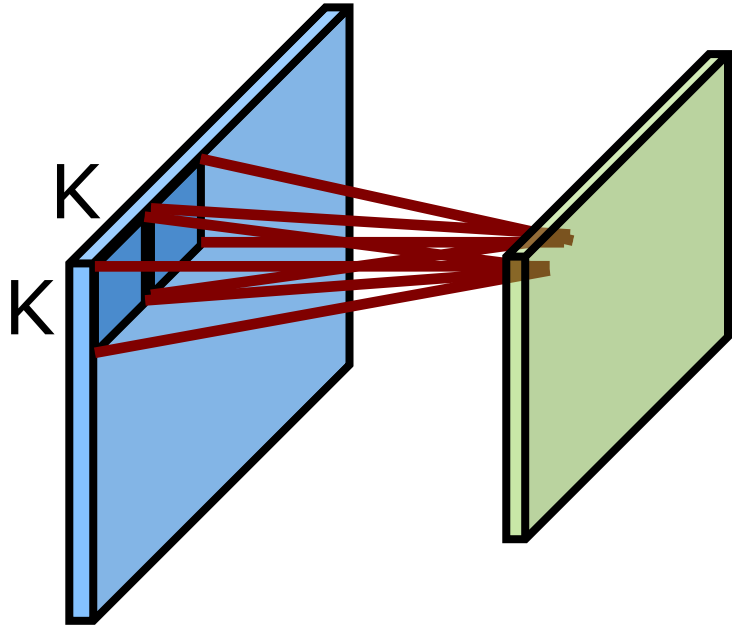
- Stack of convolutional layers
- Pooling layer to increase receptive field of layers

Receptive field



- Location in deeper layers take inputs of window of earlier layers
- Deeper layers “see” more from earlier layers

Pooling Layer



- Pooling layers increase the receptive field & aggregates information
- Translation invariance to small shifts
- Common: max pooling, average pooling

Example: Max Pooling

12	14	1	4	4	1
3	4	5	2	2	3
8	9	12	3	4	7
8	3	4	3	3	4

14	5	4
9	12	7

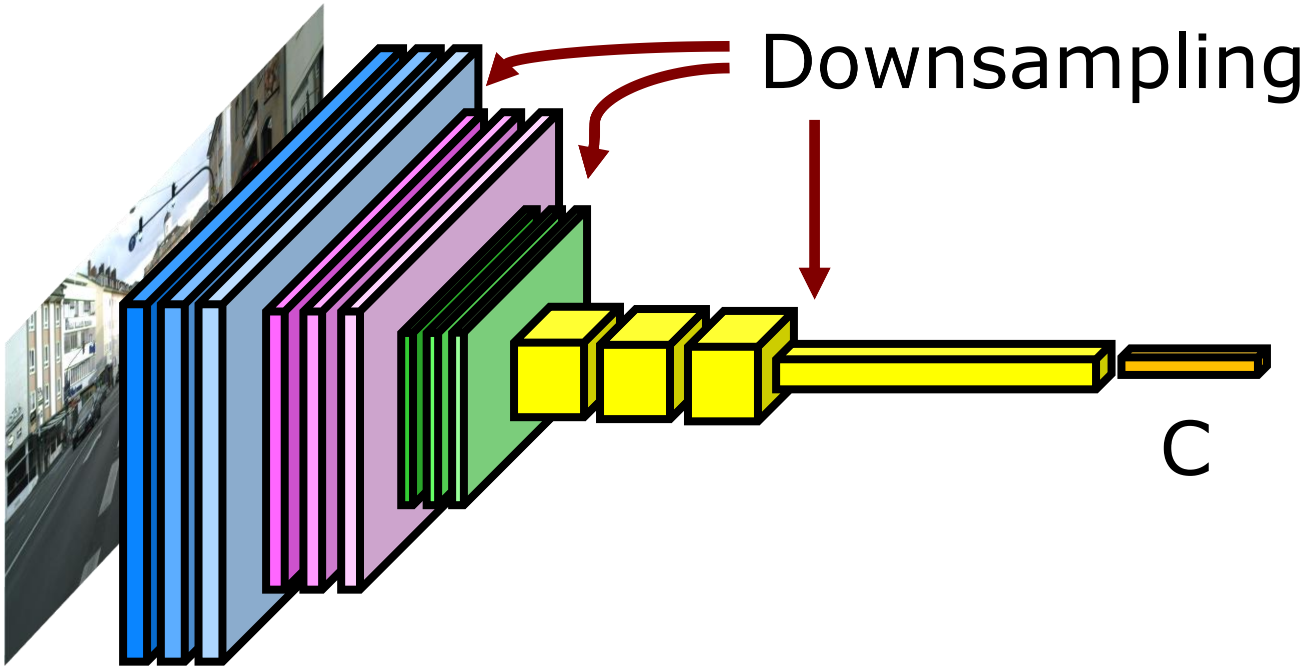
2×2 max pooling, stride 2

- Compute maximum in each region

Strided Convolution

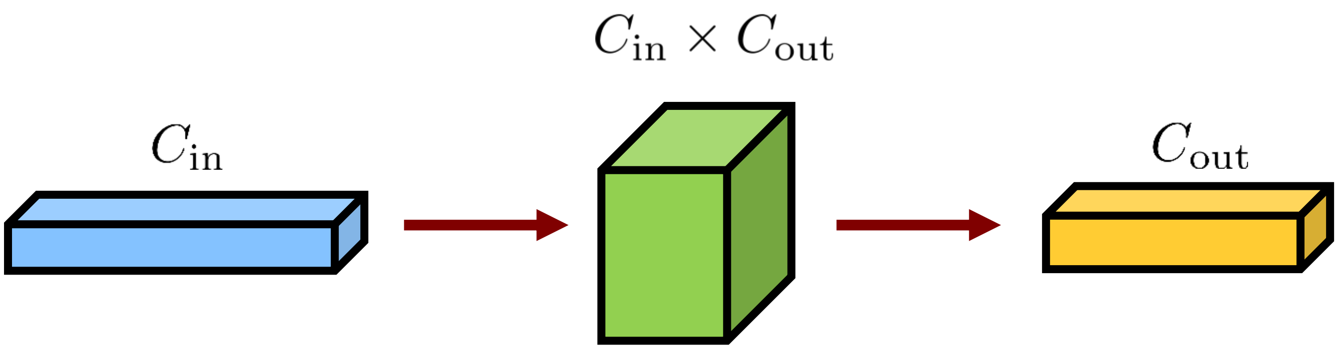
- Common: Use **strided convolution** for downsampling to increase the receptive field
- Stride > 1 reduces size of feature map

Convolution Neural Network



- Output usually appropriately shaped tensor
- Example: C classes ➔ C logit values for softmax

Fully connected (FC) layer



- Each value of the input is used to produce output value: $\mathbf{y} = \mathbf{W}\mathbf{x}$
- Common: flattening of $C \times H \times W$ tensor to vector $C \cdot H \cdot W$ before FC layer
- Also called **linear layer**

Learning

- Neural network is basically just a rather complex function:

$$f(\mathbf{x}_i; \theta) = L_3(L_2(L_1(\mathbf{x}_i; \mathbf{w}_1); \mathbf{w}_2); \mathbf{w}_3)$$

- **Loss function** is the objective we want to minimize; determines what network structure should learn

Common Loss Functions

- Loss $\ell(y_i, f(\mathbf{x}_i;\theta)) \in \mathbb{R}$ determines difference between prediction $f(\mathbf{x}_i;\theta)$ and target y_i

- Examples:
  - **L2 loss** for regression tasks

$$\ell(y_i, f(\mathbf{x}_i;\theta)) = (y_i - f(\mathbf{x}_i;\theta))^2$$

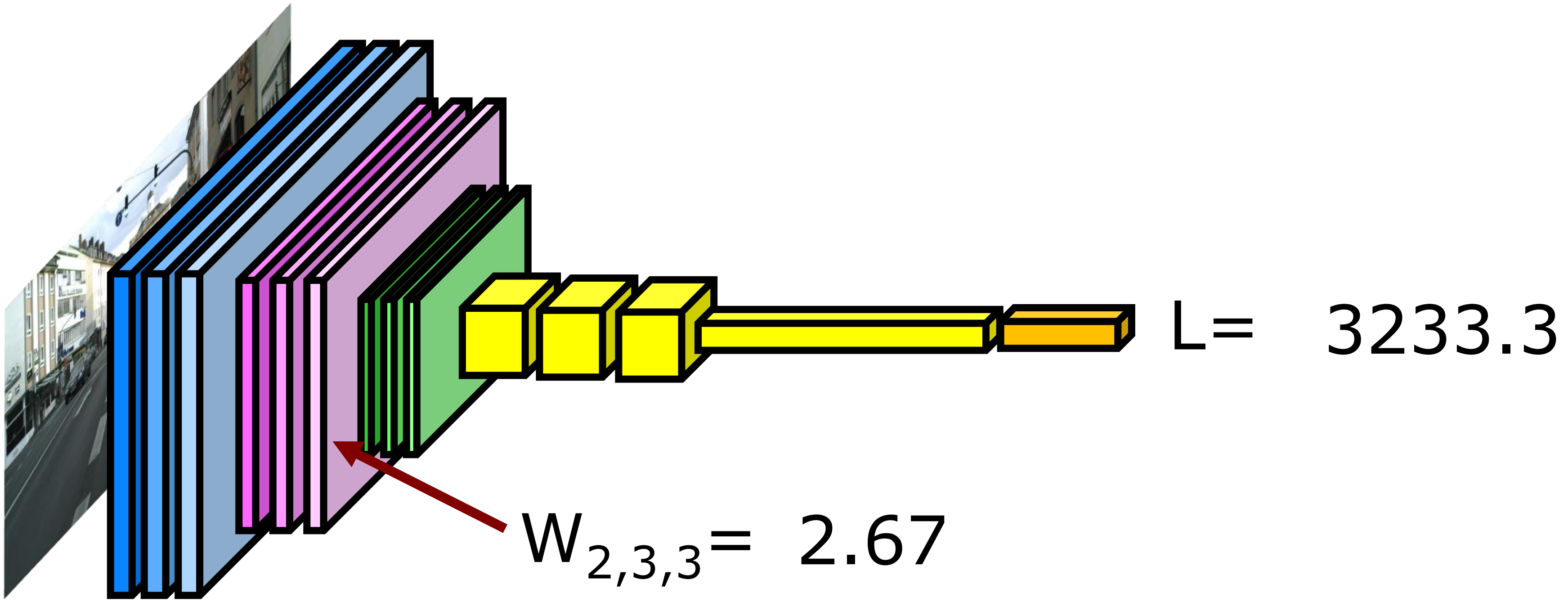
  - **Cross entropy** loss for classification

Softmax

$$\ell(j, f(\mathbf{x})) = -\log \frac{\exp(f_j(\mathbf{x}))}{\sum_k \exp(f_k(\mathbf{x}))}$$

$$= -f_j(\mathbf{x}) + \log(\textstyle\sum_k \exp(f_k(\mathbf{x})))$$

Learning via gradient descent



- **Idea:** Determine how to change parameters in a layer to reduce loss
- Parameter updates efficiently computed via **back propagation**

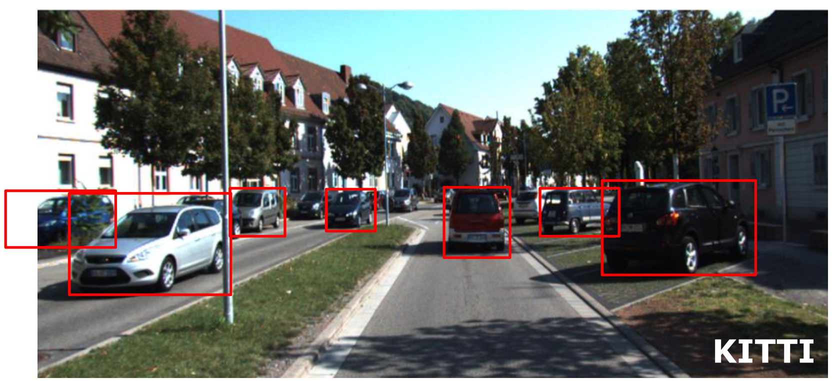
More details on CNNs

- We touched only parts needed to understand papers in the seminar
- Much more theory, building blocks, best practices
- For more details on CNNs, see the links in the description of the video.

Perception Tasks

Classification

Semantic Segmentation

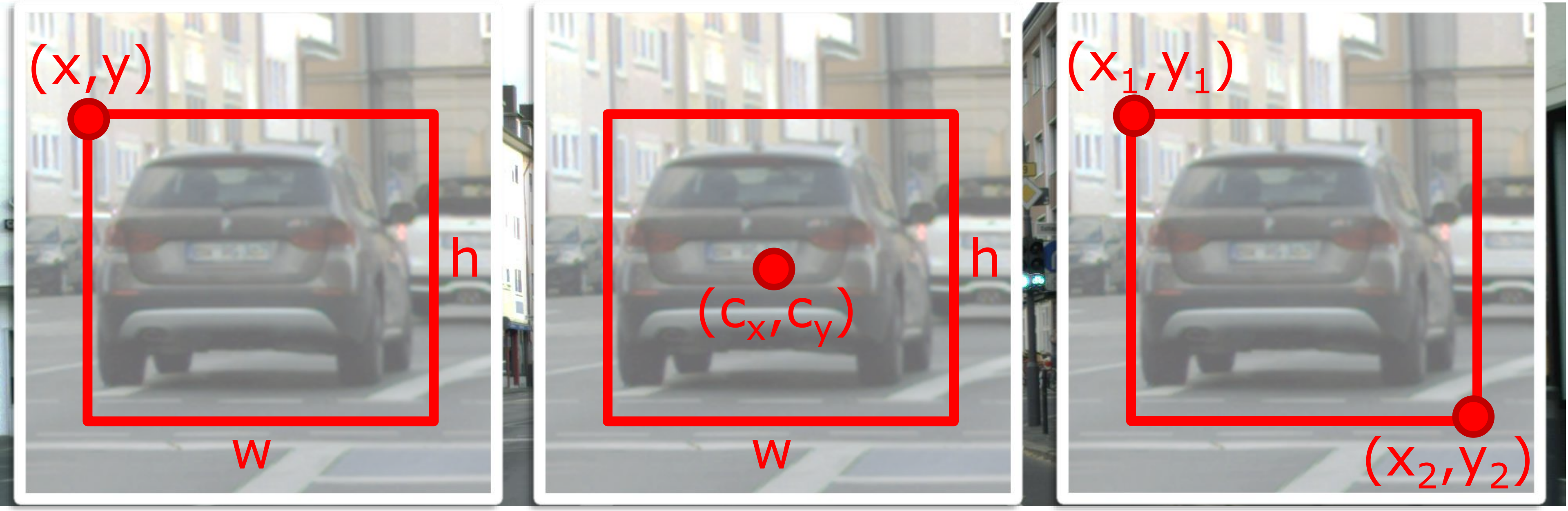


Detection

Panoptic Segmentation

[Geiger, 2012] [Cordts, 2014]

Object detection task



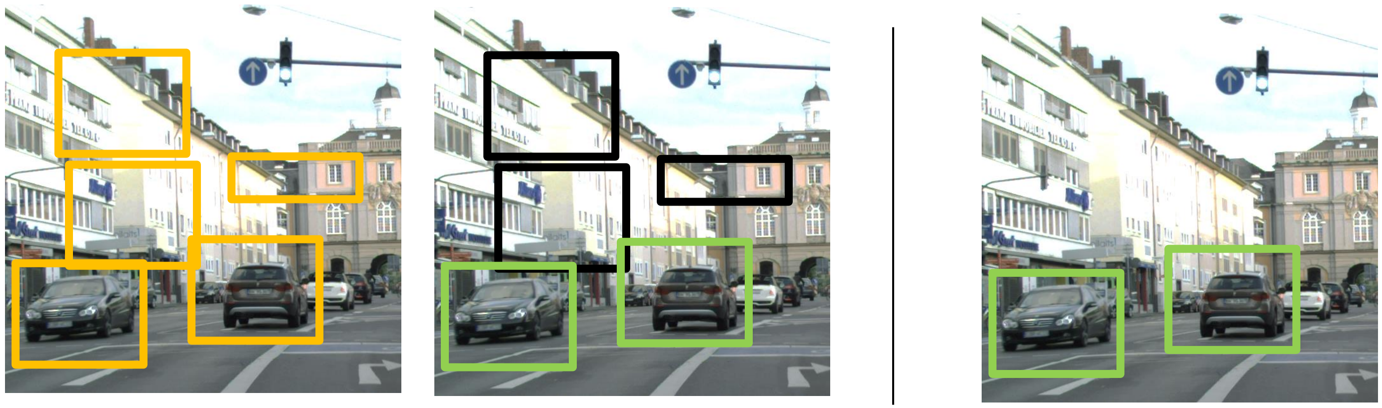
- Input: RGB Image
- Output:
  - **bounding boxes** defined by (x, y, w, h) or (c_x, c_y, w, h) or (x_1,y_1, x_2,y_2)
  - **confidence scores** in [0,1]

Modern Object Detectors

- Rely mainly on Convolution Neural Networks (CNN)

- Two main paradigms:
  - **Anchor-based** approaches
  - **Anchor-free** approaches

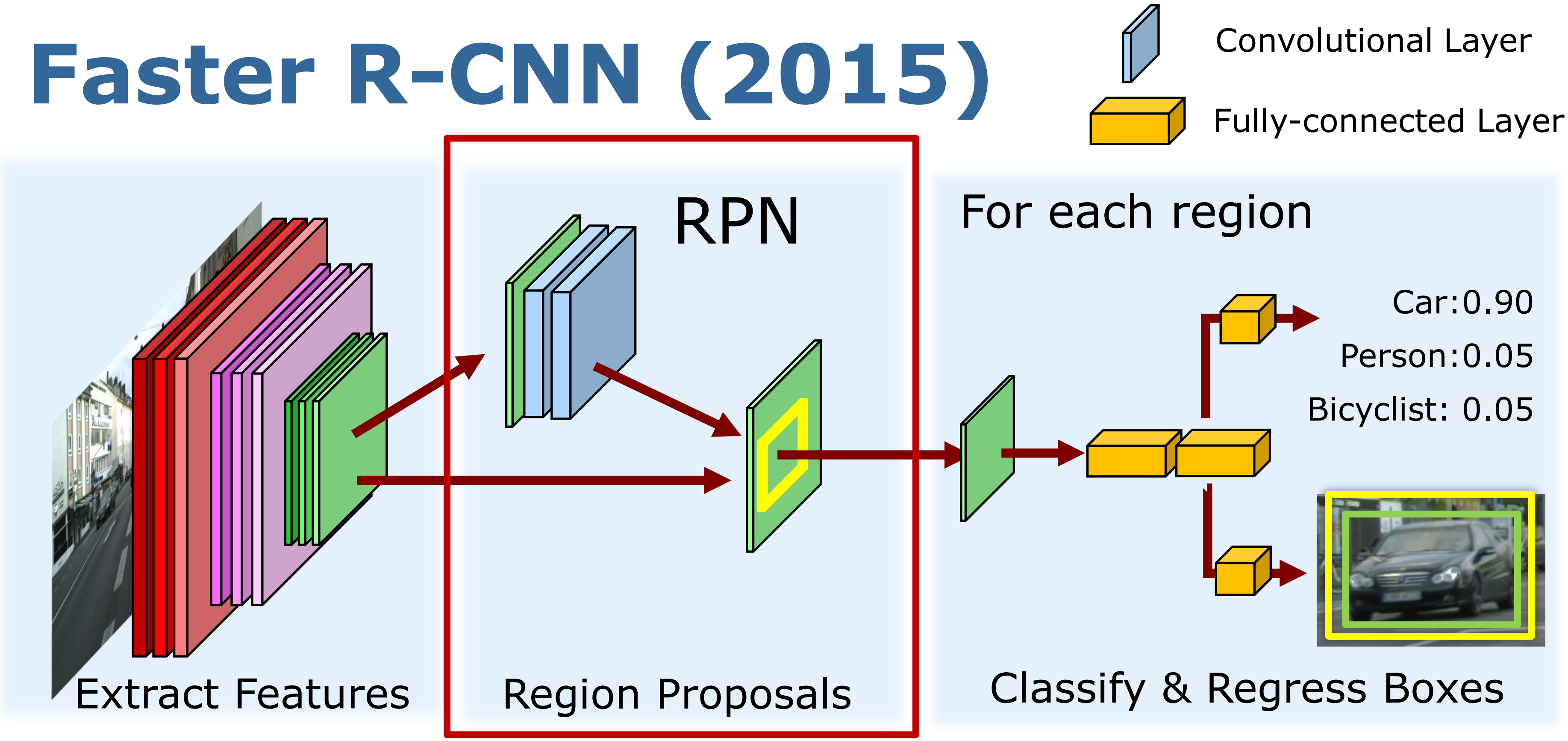
Anchor-based Approaches



Two-stage Single-stage

- **Two-stage** approaches:
  - First stage: Generate proposals based on anchors
  - Second stage: Refine and classify proposals
- **Single-stage** approaches:
  - Refine & classify anchors in single pass

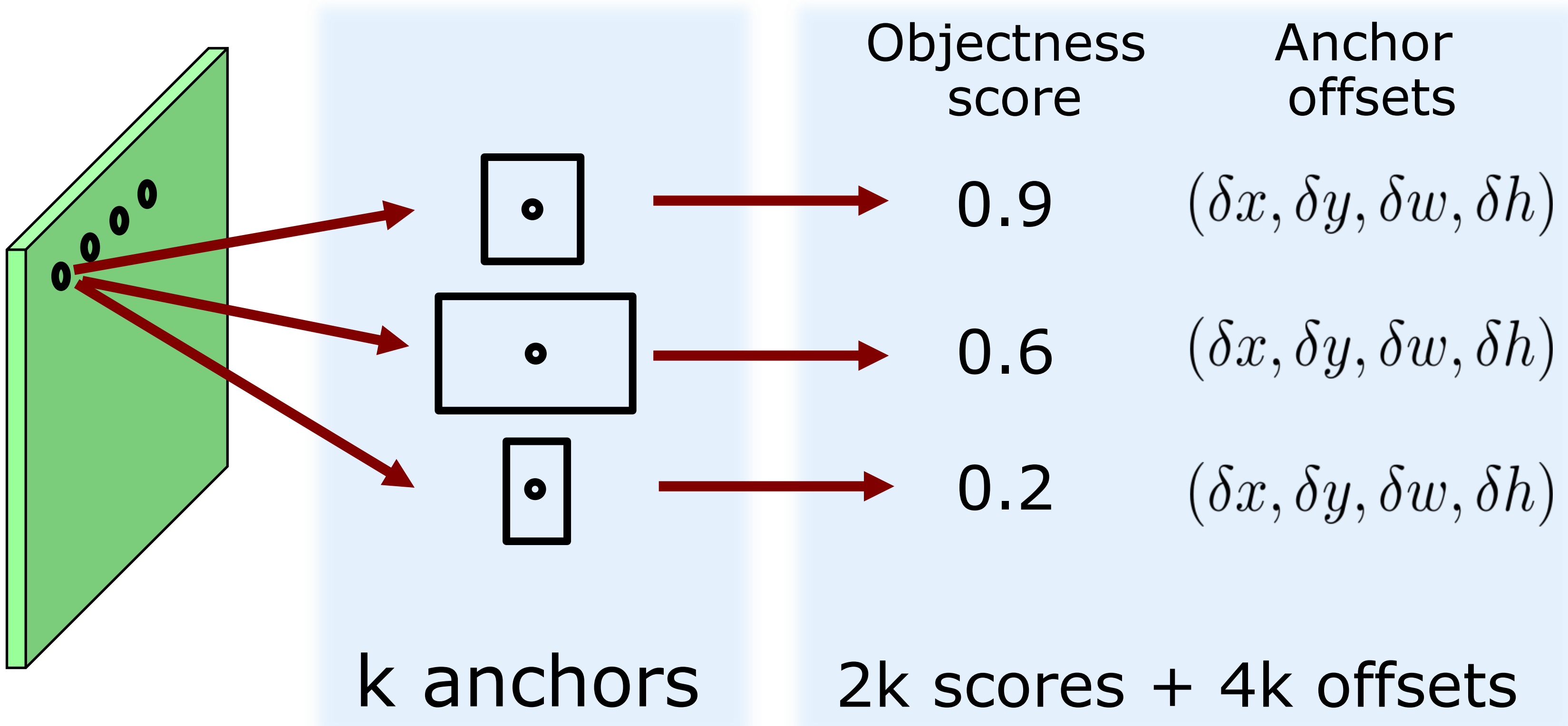
Faster R-CNN (2015)



- **Region Proposal Network** produces proposals
- **Region-wise classification network** uses same features as input as RPN (shared features)

[Ren, 2015]

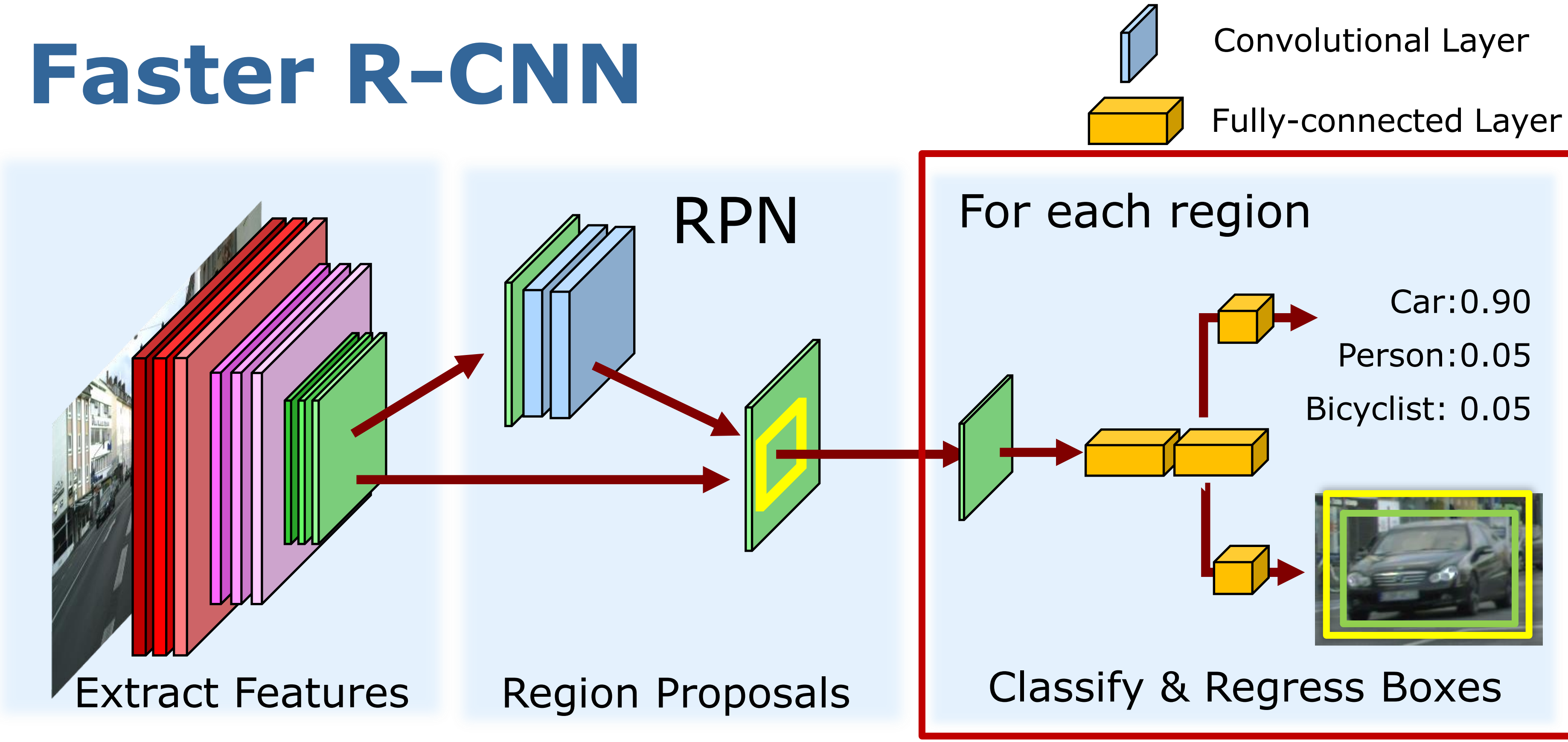
Region Proposal Network (RPN)



- Scores set of anchors with fixed initial sizes
- Produces objectness and anchor offsets (for each anchor)
- Keep **N-top scored anchors** as RoI for classification in second stage

[Ren, 2015]

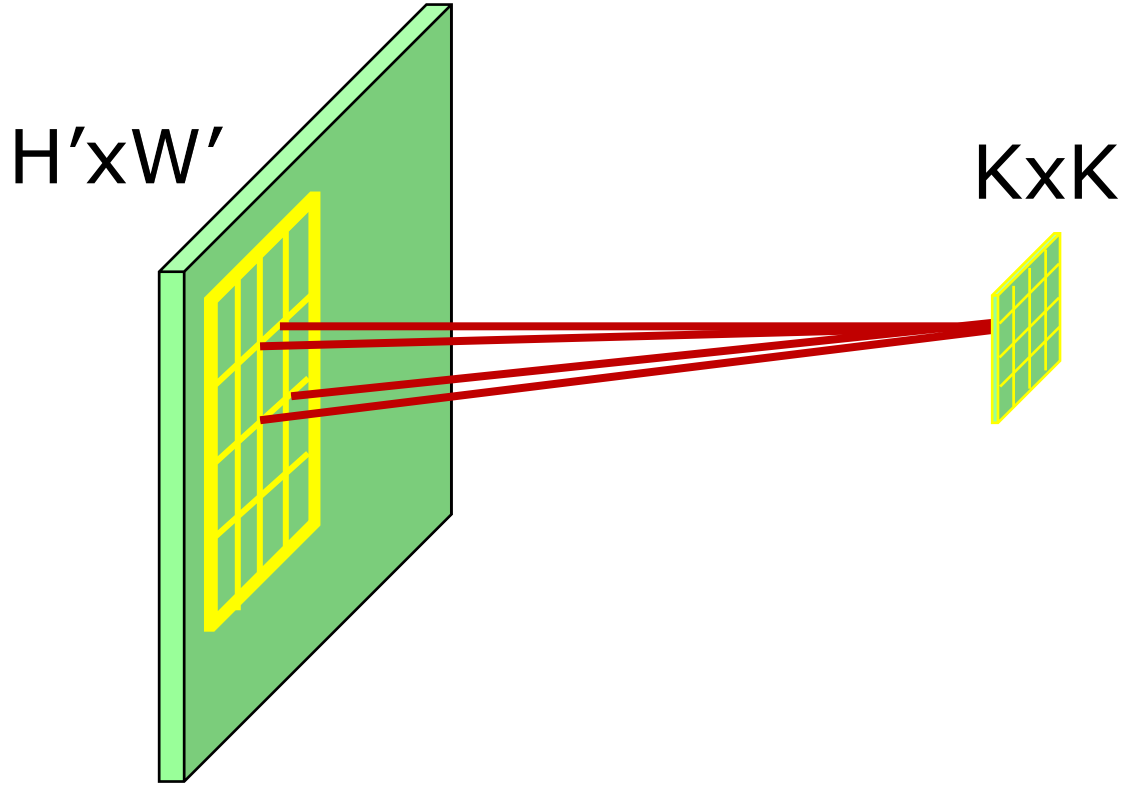
Faster R-CNN



- **Region-wise classification network** classifies and refined bounding boxes (regression)
- But, how to extract proposal-specific features?

[Ren, 2015]

Region-of-Interest(RoI) Pooling



- Adaptive max pooling brings extracted feature maps into appropriate size for RoI Network

[Girshick,2015]

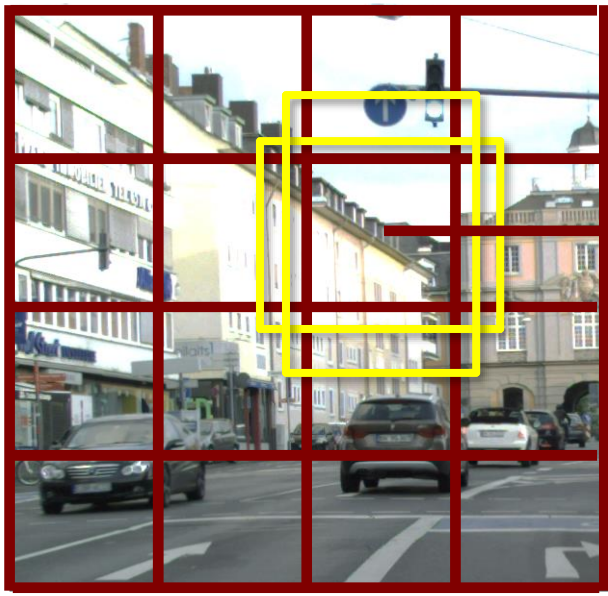
Faster R-CNN Summary

- Jointly trainable RPN and RoI classifier → sharing of features possible (alternated training)
- Fast enough for near real-time operation (~10 Hz)

- RPN already provides object bounding boxes → second stage needed?

You Only Look Once (YOLO)

S = 4



Per anchor scores & offsets:

$$(O_1, \delta x_1, \delta y_1, \delta w_1, \delta h_1)$$
$$(O_2, \delta x_2, \delta y_2, \delta w_2, \delta h_2)$$

Per cell class scores:

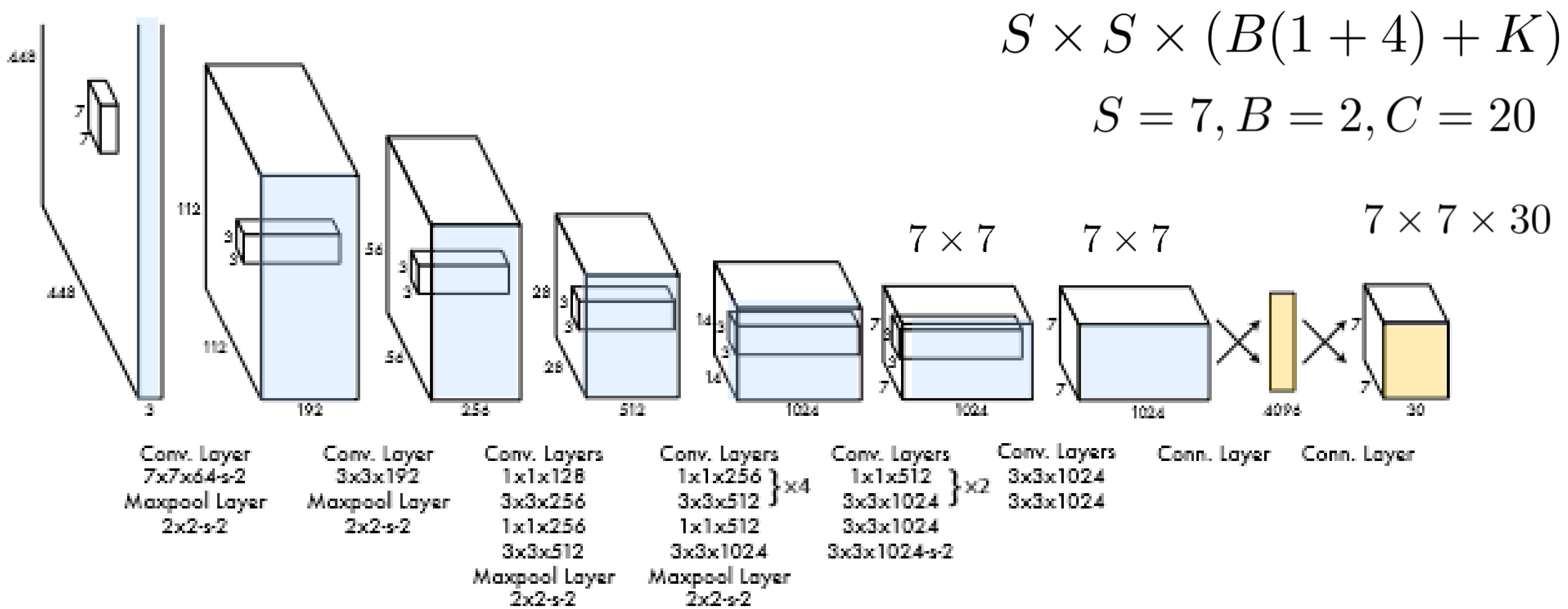
$$(C_1, C_2, \ldots, C_K)$$

Output: $S \times S \times (B(1+4)+K)$

- Predicts bounding boxes with single forward pass
- Each anchor gets objectness score O, bounding box offsets
- Objectness + class score determines outcome

[Redmon, 2015]

YOLO Architecture



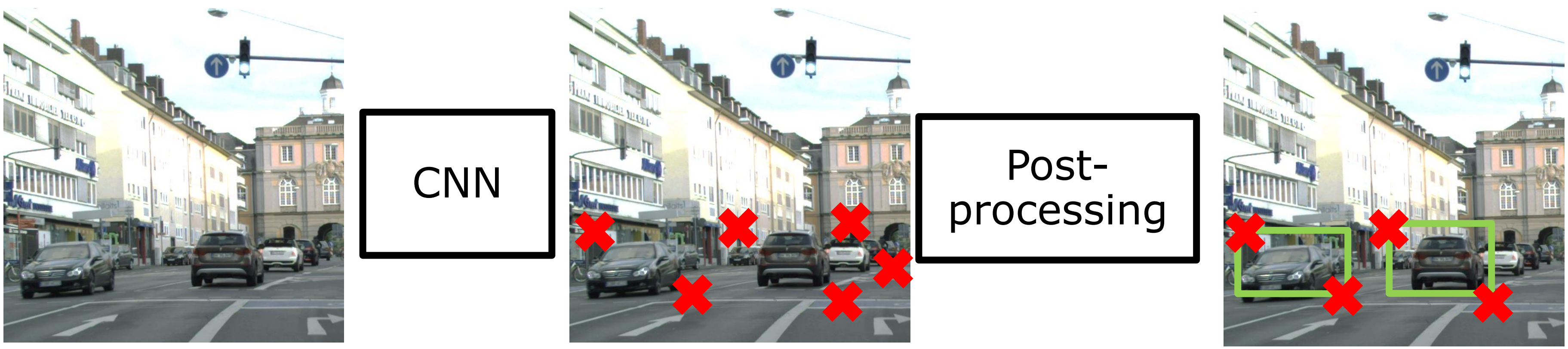
- YOLO uses a 24-layer convolutional network (DarkNet) with 2 fully connected layers, 2 anchors, 20 classes (Pascal VOC)

[Redmon, 2015]

YOLO Summary

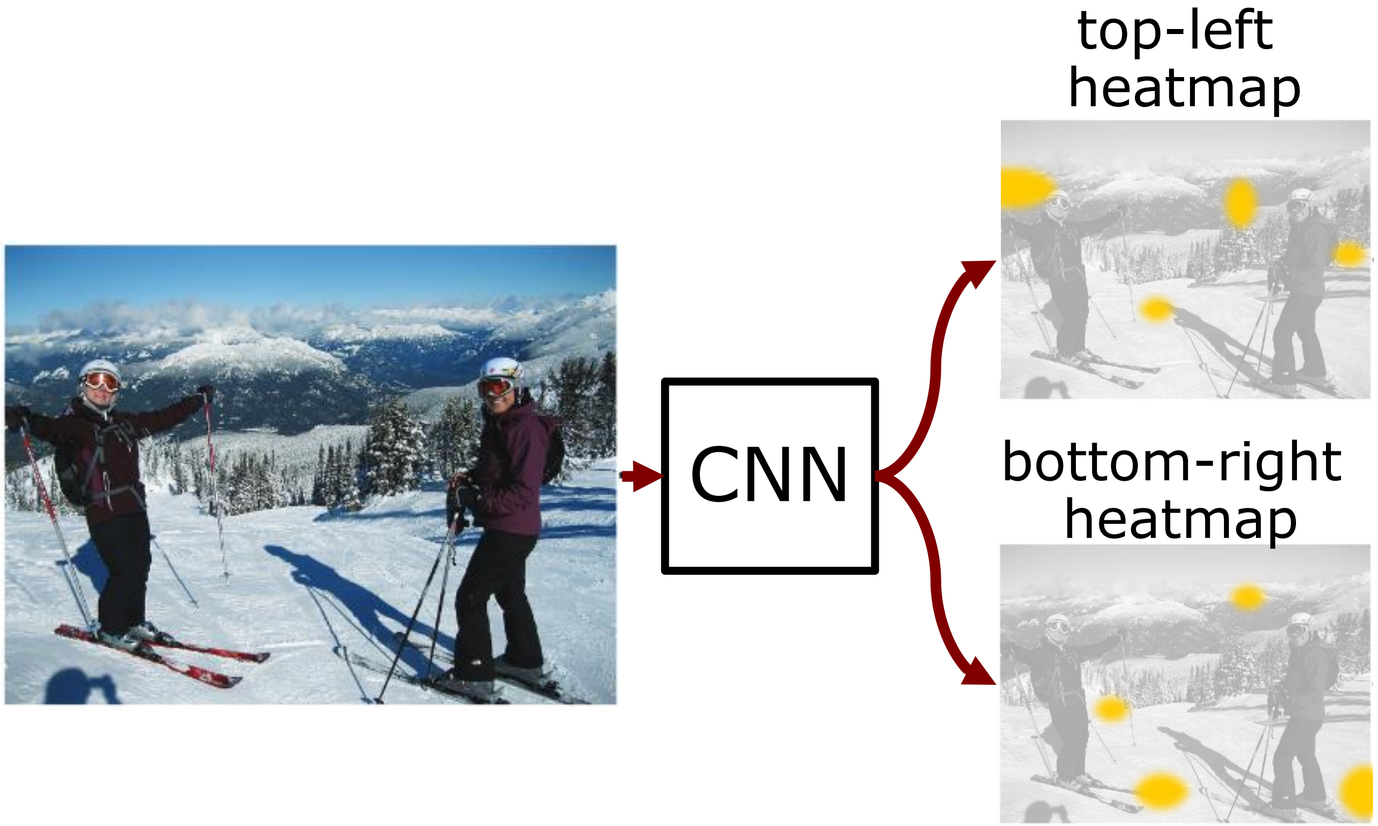
- Each grid cell produces at most 1 bounding box
- Only single pass though CNN needed
  → Blazingly fast (up to 144 Hz)
- But less proposals then Faster R-CNN
  → less accurate, misses too close objects

Anchor-free Approaches



- Determine keypoints (e.g., corners, centers of bounding boxes)
- Post-processing uses keypoints to produce bounding boxes

Bounding boxes from corners

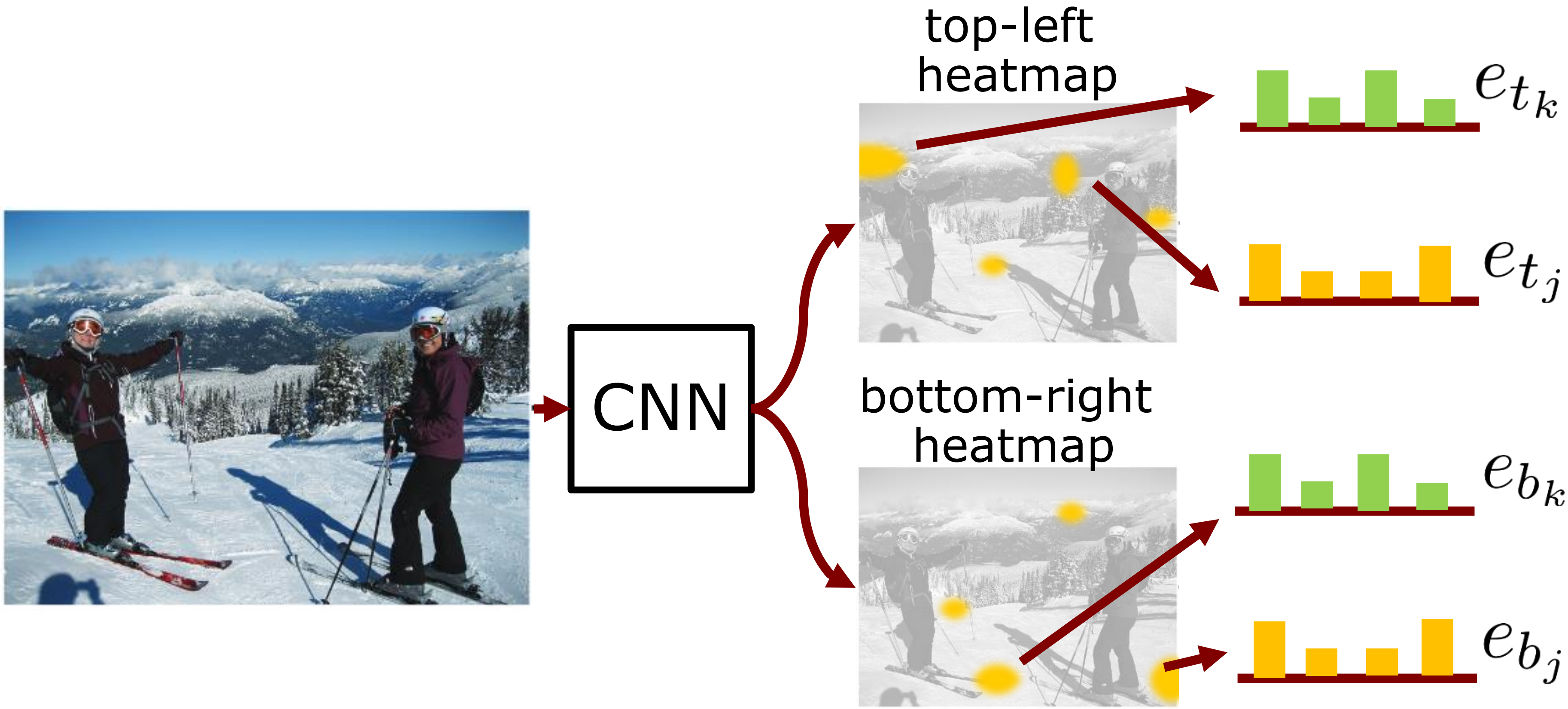


- Instead of scoring of anchors, **CornerNet** determines corners of bounding boxes
- Produce heatmaps of likelihood that at given pixel is upper-left or bottom-right corner

[Law, 2018]

Photo from [Law, 2018]

How to associate corners?

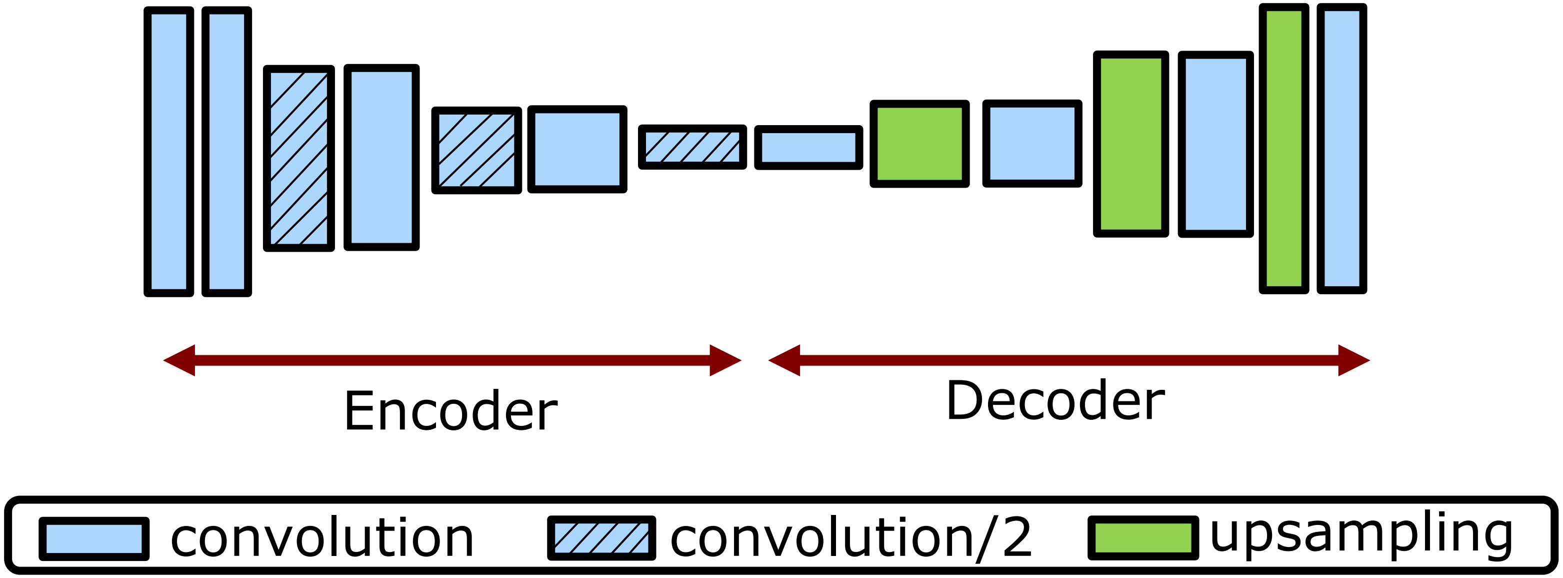


- To associate top-left and bottom-right corners, CornerNet determine embedding ("features")
- Similar embeddings correspond to the same object

[Law, 2018]

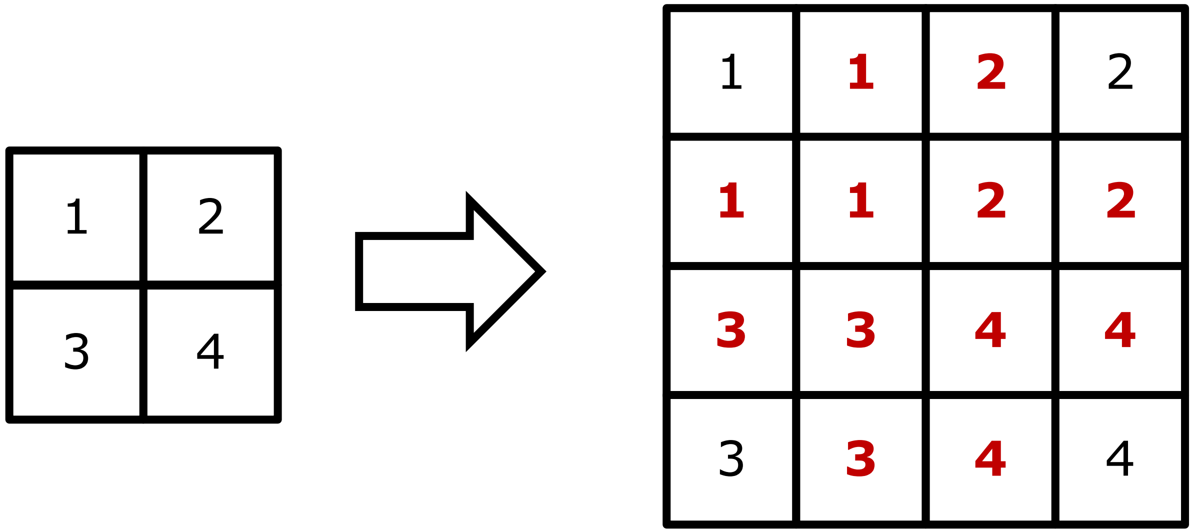
Photo from [Law, 2018]

Encoder Decoder Architecture



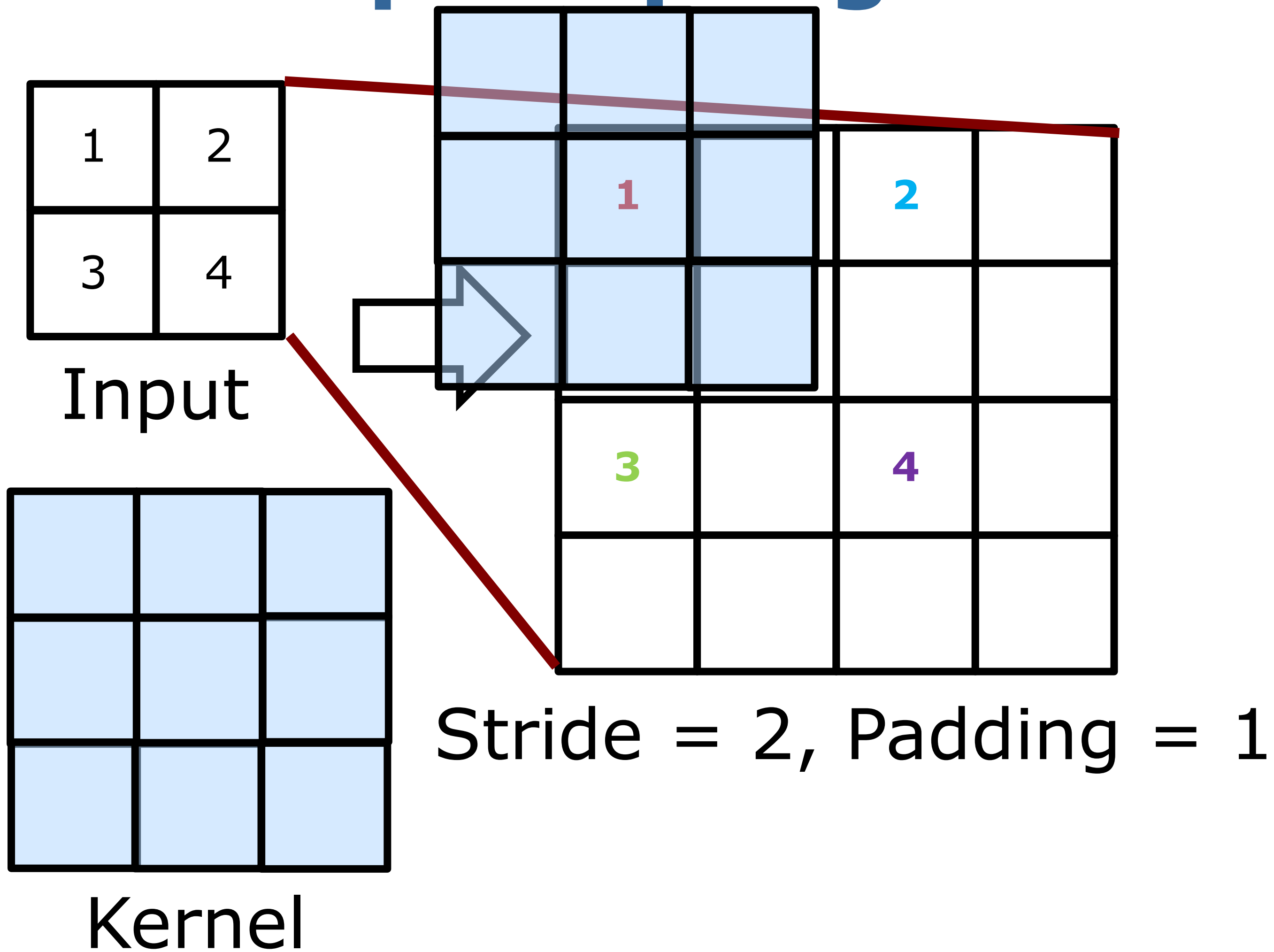
- We want to have **pixel-wise** features
- Encoder uses **strided convolutions** or **max pooling** to down-sample feature maps
- Decoder upsamples feature maps to original resolution using **upsampling** operations

Common Upsampling Methods



- **Nearest neighbor upsampling** just copies values from nearby pixels

Common Upsampling Methods



- Learnable weights for interpolation: Transpose Convolutions "inverts" convolution

CornerNet Summary

- **Anchor-free** approach
  - Corner locations (upper-left, bottom-right) with embedding vectors
  - Similarity between learned embedding vectors determines bounding box
- Encoder-Decoder architecture produces pixel-wise outputs

Perception Tasks

Classification

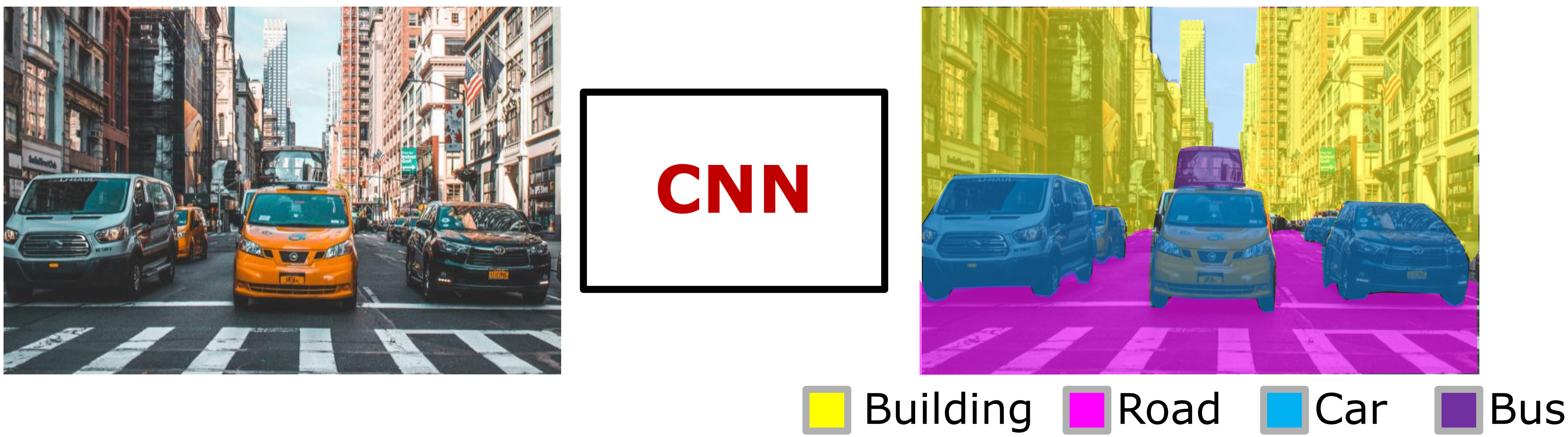
Semantic Segmentation

Detection

Panoptic Segmentation

[Geiger, 2012] [Cordts, 2014]

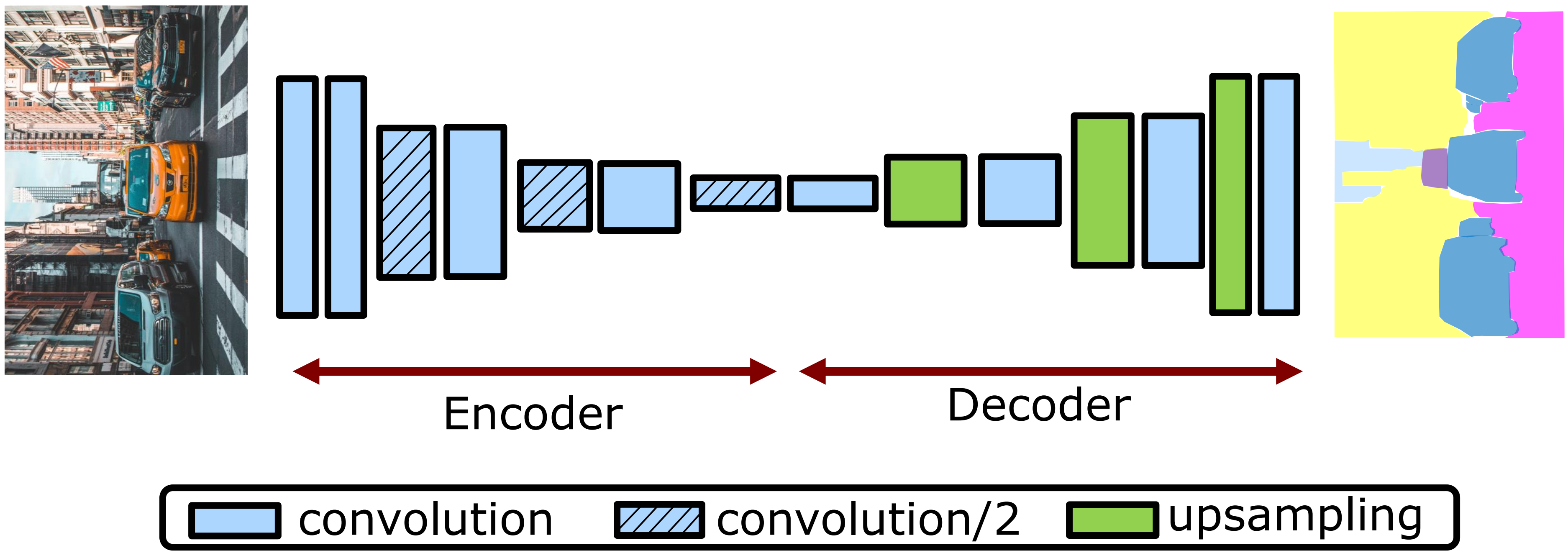
Semantic Segmentation



- **Goal:** Provide label $y_{i,j} \in \{1, \ldots, K\}$ for each pixel in the image.

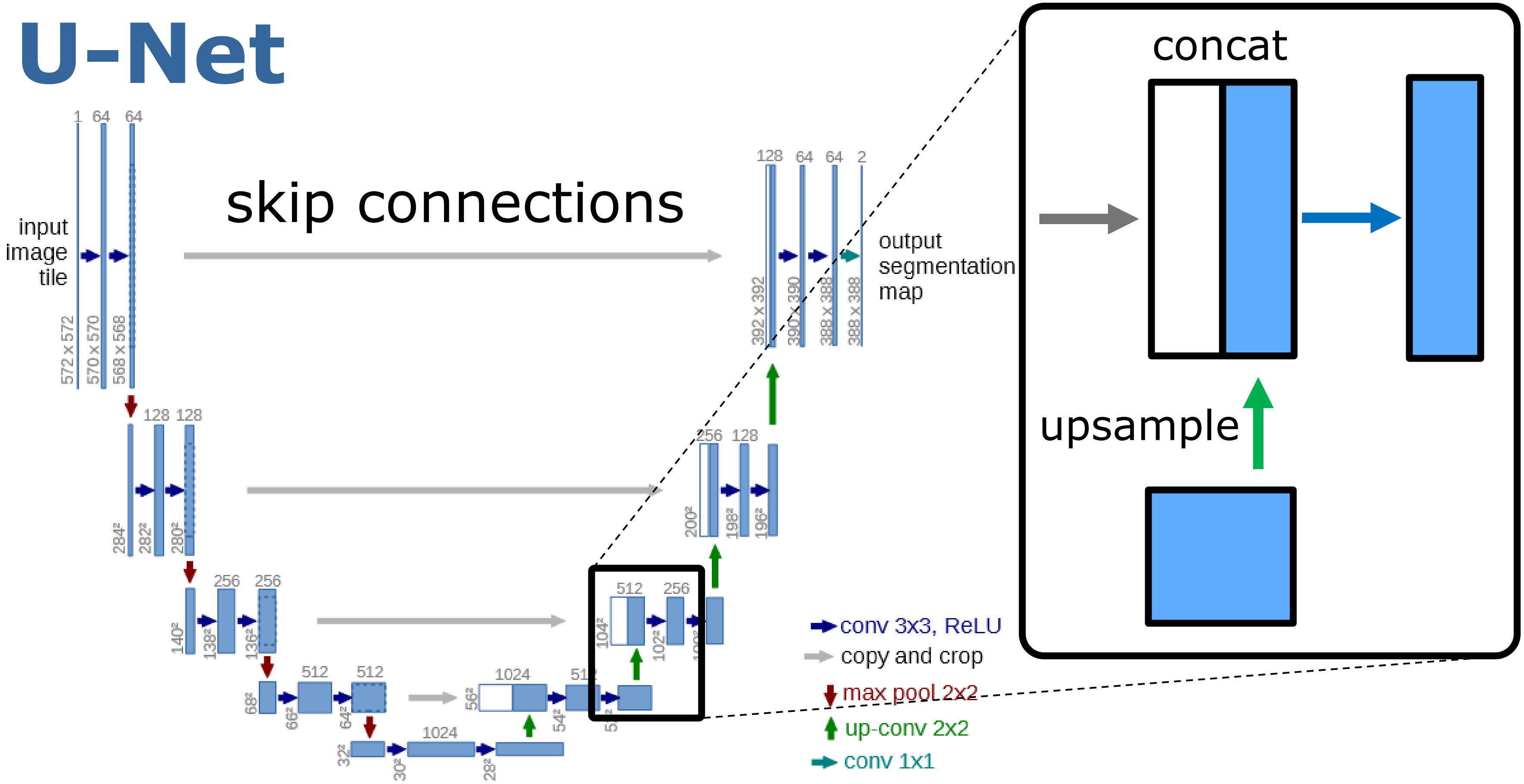
Photo by Marisol Benitez on Unsplash

Encoder-Decoder Architecture



- Combine up-sampling with convolutional layers to regain spatial resolution

U-Net



- **Skip connections** help to retain fine-grained results
- Concatenate feature volumes from encoder and upsampled feature volumes from decoder
- Convolve to reduce number of channels

[Ronneberger, 2015]

Figures from [Ronneberger, 2015]

U-Net Summary

- **Encoder-decoder** architecture to produce pixel-wise logits for classification
- **Skip connections** to use high-resolution information from encoder

Perception Tasks

Classification

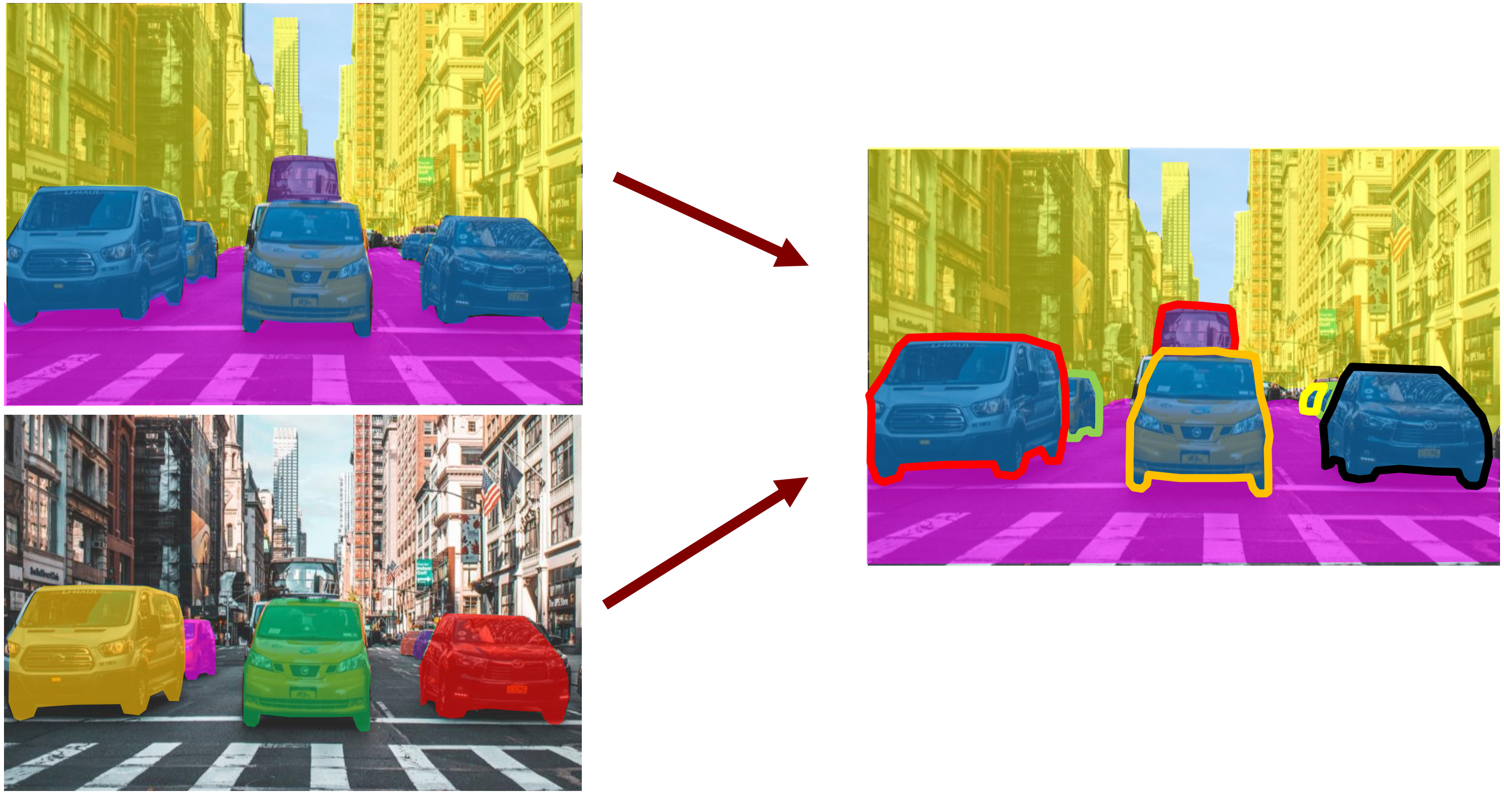
Semantic Segmentation

Detection

Panoptic Segmentation

[Geiger, 2012][Cordts, 2014]

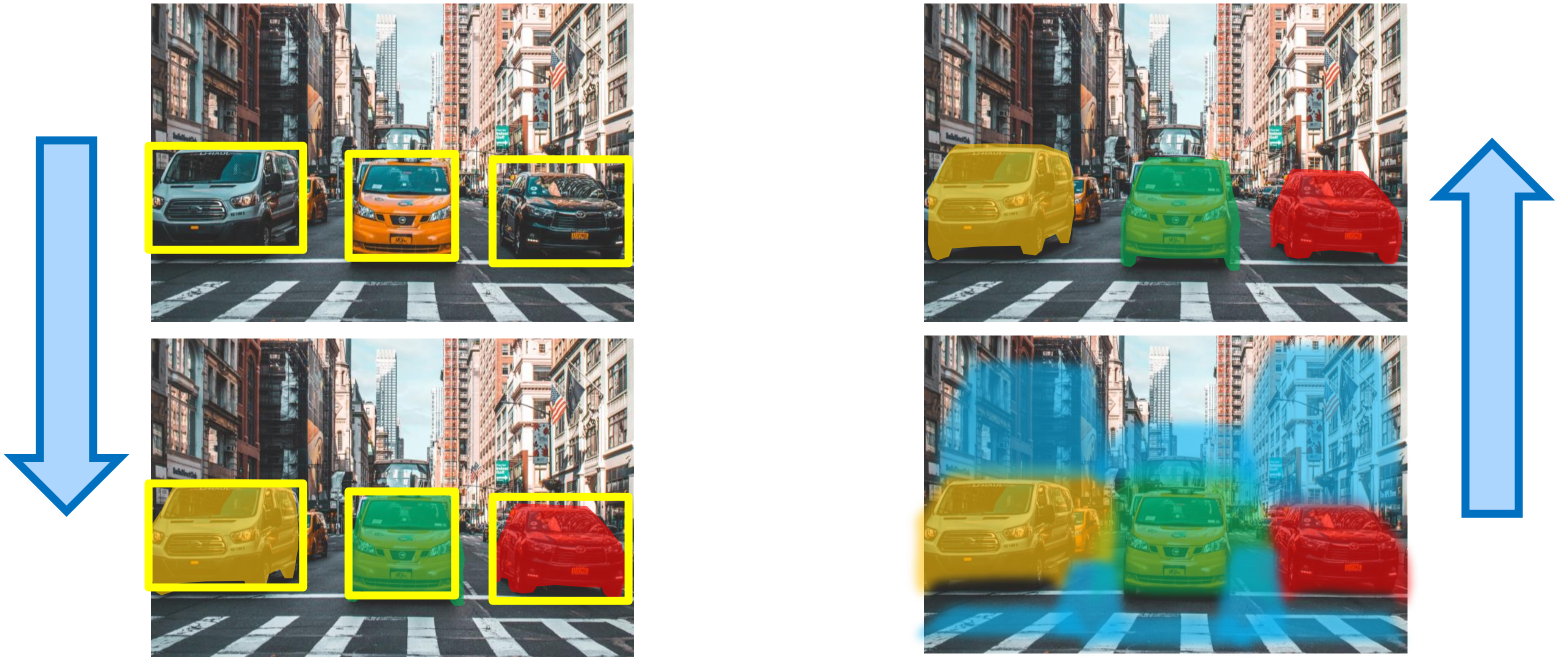
Panoptic Segmentation



- Panoptic Segmentation unifies **semantic** and **instance** segmentation
- Distinguish **stuff** (e.g., vegetation, road, …) and **thing** classes (e.g., car, pedestrian, …)

[Kirillov, 2019]

Top-down vs. Bottom-up Instance Segmentation

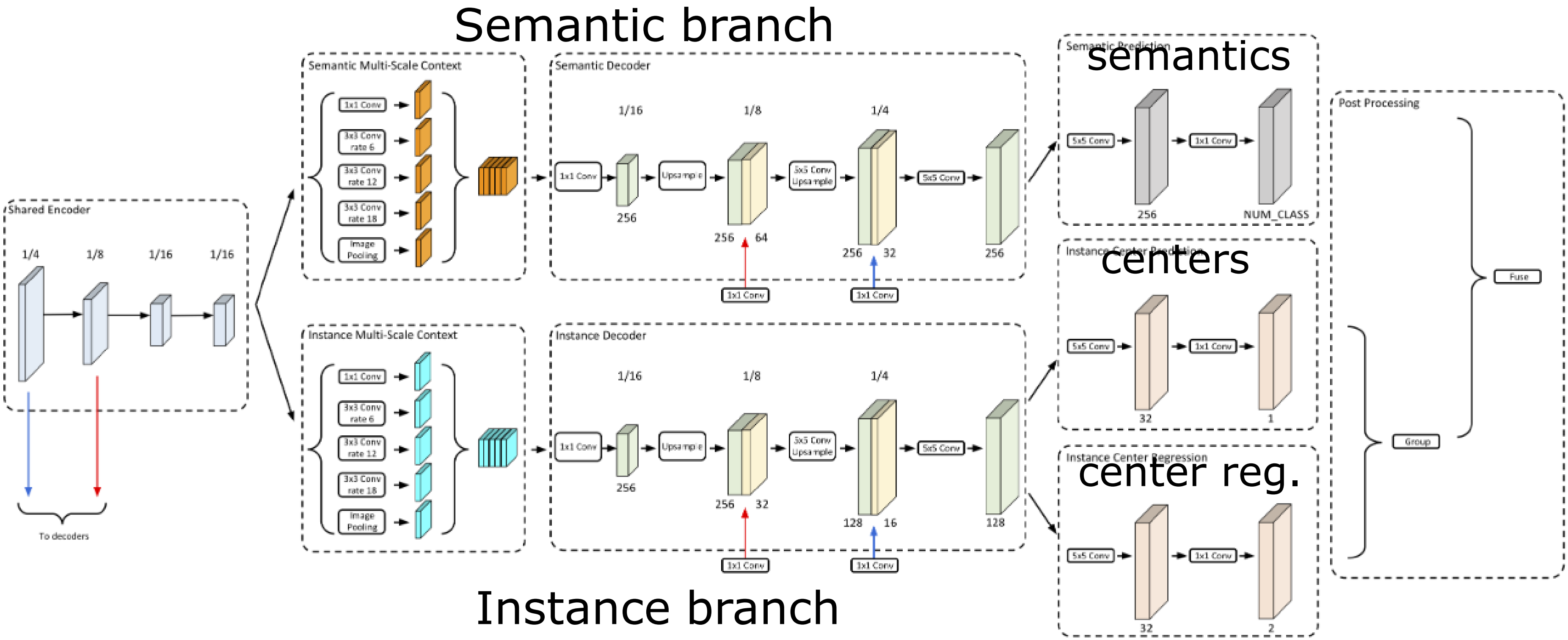


Top-down Approach

Bottom-up Approach

- **Top-down**: Instances are first determined and then foreground/background mask estimated
- **Bottom-up**: Determine per-pixel properties that are then used to cluster instances

Panoptic-DeepLab

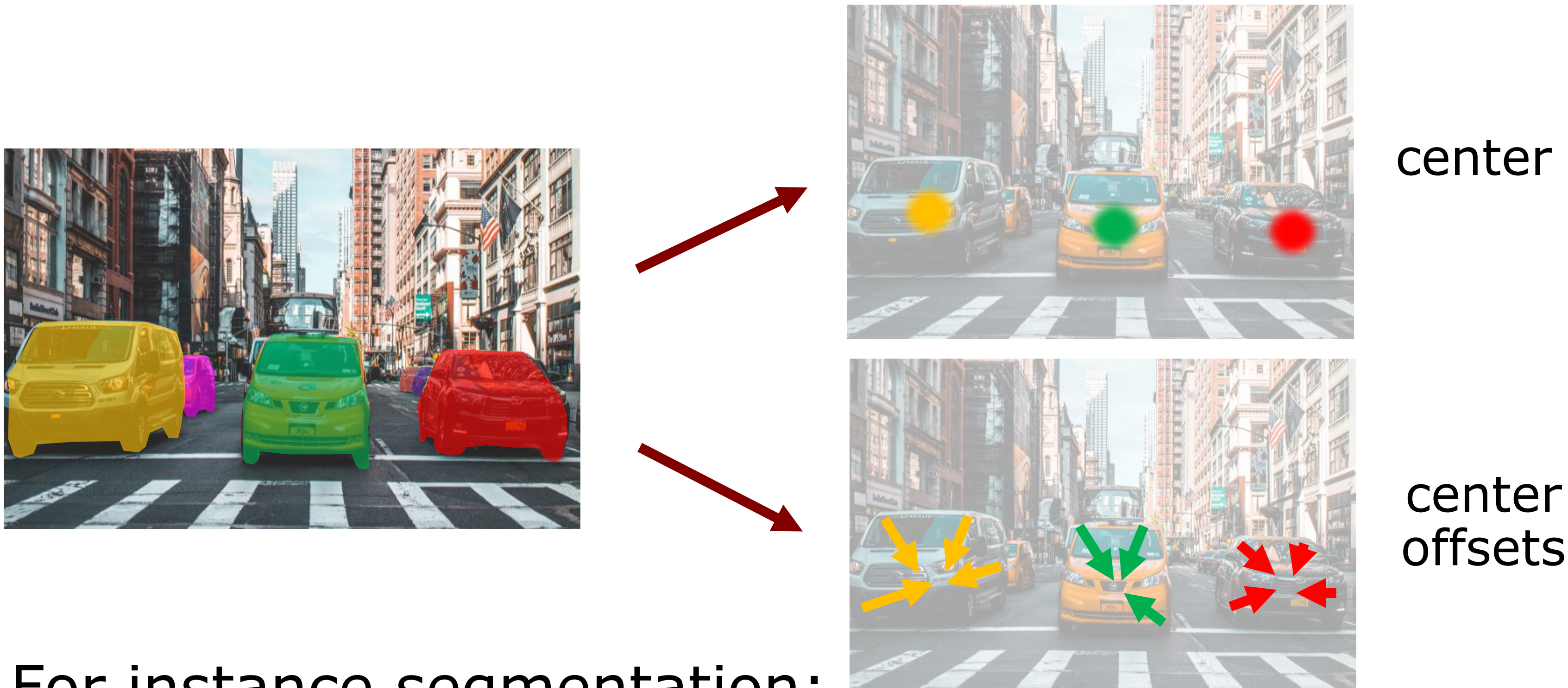


- **Bottom-up approach** using separate branches for semantic and instance segmentation
- Use semantic labels to filter instances & majority vote on instances to assign instance labels

[Cheng, 2020]

Figure from [Cheng, 2019]

Panoptic-DeepLab: Instances



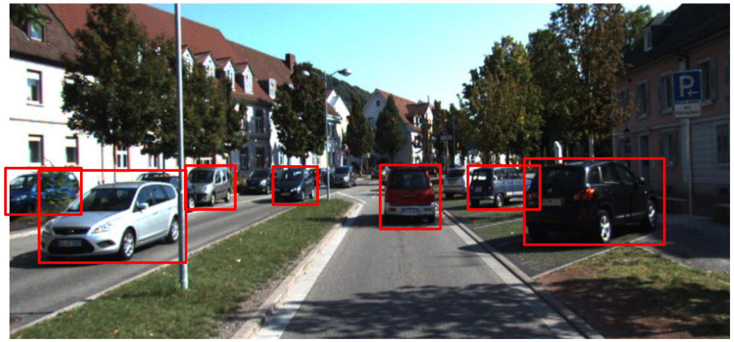
- For instance segmentation:
  - Instance center prediction (= center of mass)
  - For each instance pixel: estimate offset vector that points towards instance center
- At inference time: Use offsets to assign instance id of closest instance center.

[Cheng, 2020]

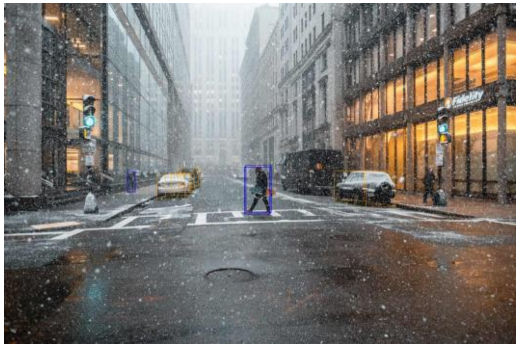
Panoptic DeepLab Summary

- **Shared encoder** with different decoders for semantics and instances
- **Center** and **center offsets** for bottom-up clustering of instances
- Aggregation of semantic and instance mask to produce instances

Need for labeled data



KITTI



NuScenes

Waymo Open Dataset

- Mostly supervised training
  → need for accurately **labeled data**
- Several research datasets available
- Companies have dedicated teams to annotate data (non-public dataset)

Automotive Dataset (Detection)

Name	Year	#Categories	#Images	Data
KITTI	2012	8	15k	B,S
BDD100K	2017	10	100k	B
ApolloScape	2018	8-35	144k	B
KAIST	2018	3	9k	B
Argoverse	2019	15	22k	B
Lyft L5	2019	9	46k	B
A2D2	2019	14	12k	B
nuScenes	2019	23	40k	B,S
Waymo Open	2019	4	200k	B

Bounding Box (B), Segmentation Masks (S)

Segmentation Datasets

name	#categories	#images	task
▪ MS COCO (2014)	80	118k	I,P
▪ Cityscapes (2016)	30(19)	5k	S,I,P
▪ Mapillary Vistas (2017)	66	25k	S,I
▪ ADE20K (2017)	2,693(150)	25k	S,I

S = Semantic Seg., I = Instance Seg., P = Panoptic Seg.

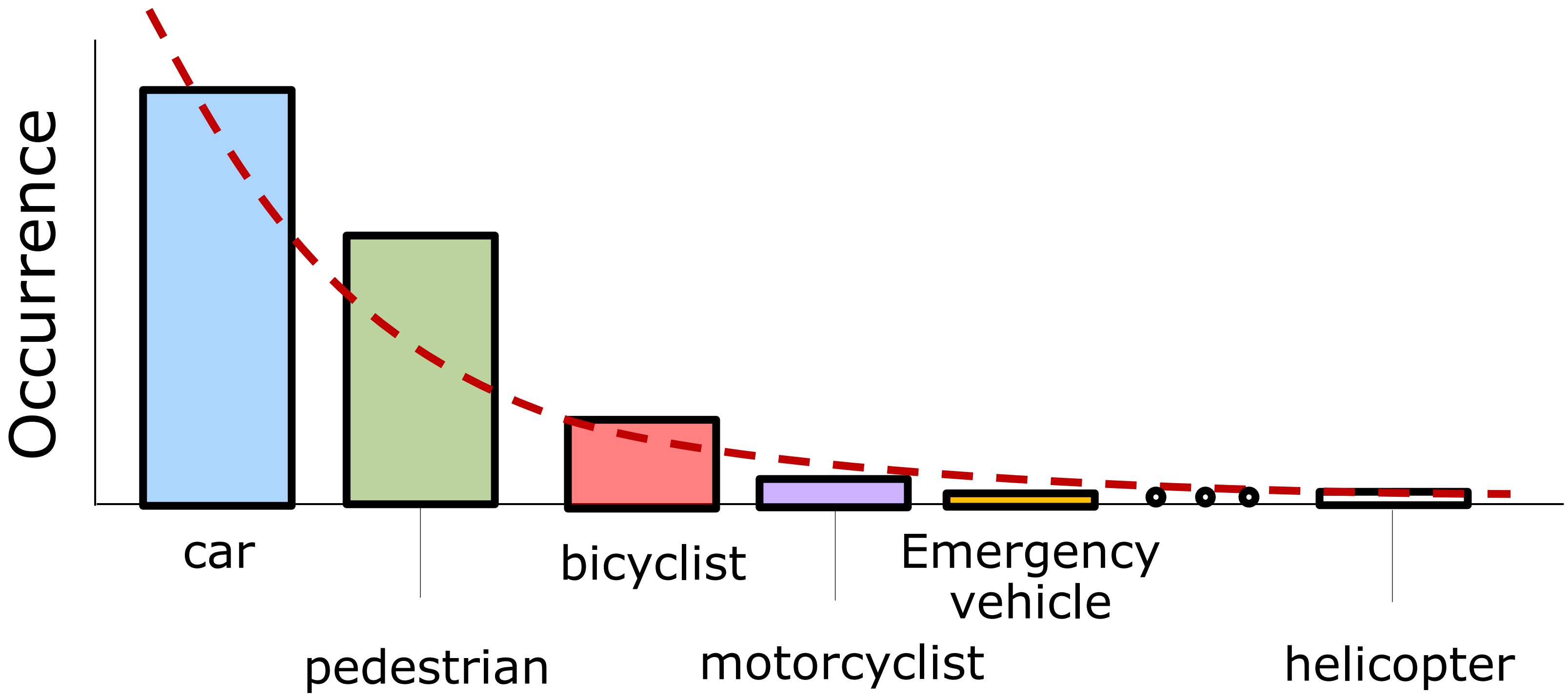
[Lin, 2014][Cordts, 2016][Neuhold, 2017][Zhou, 2017]

Labeling data is expensive

- Labeling data is tedious and expensive
- Examples
  - Cityscapes: ~1.5 h per image ➔ 7500 h/312 days for 5k images
  - Mapillary Vistas: ~1.5 h per image → 4.2 years for 25k images
  - MS COCO: 22k h (category labeling) + 10k h (instance spotting) + 26k h (instance segmentation*) → 6.6 years
- Not included: Validation of annotations!

* 22h per 1000 segments, ~1.2M instances for 80 classes

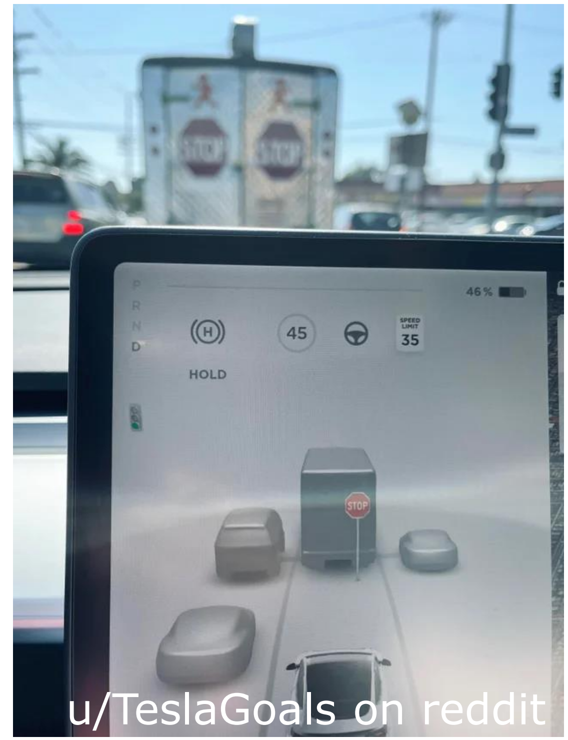
Long Tail Problem



- Class distribution is a long-tailed distribution
- Few classes (e.g., car, pedestrian) are abundantly observable
- Most classes appear in the long tail, e.g., construction vehicles, emergency vehicles, etc.

Rare Events & Situations

u/anonymousmice on reddit



u/TeslaGoals on reddit

- Rare events that should be properly handled
- At scale, rare events are not so rare anymore

Rare Events & Situations

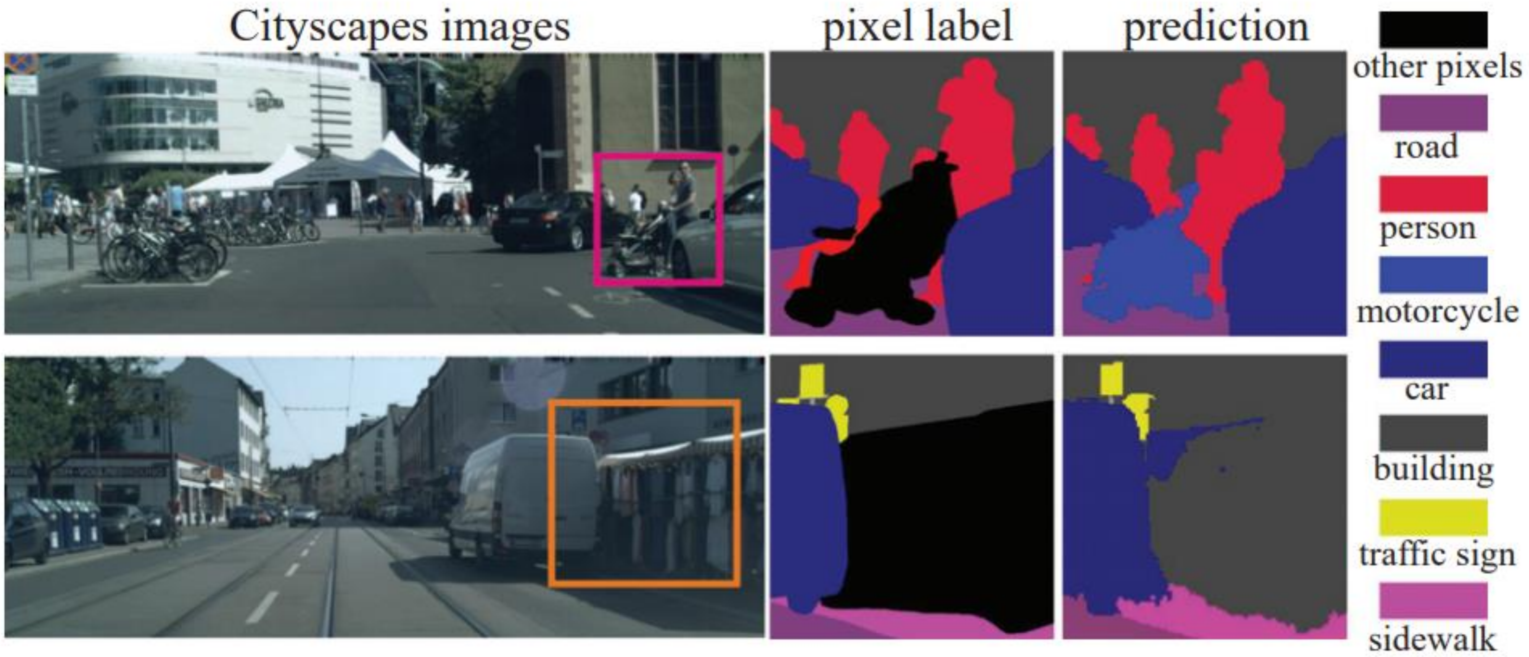

© Anil Polat


© dave gostisha


© Max Malsch


© Melanie Toomey

Open-Set Perception

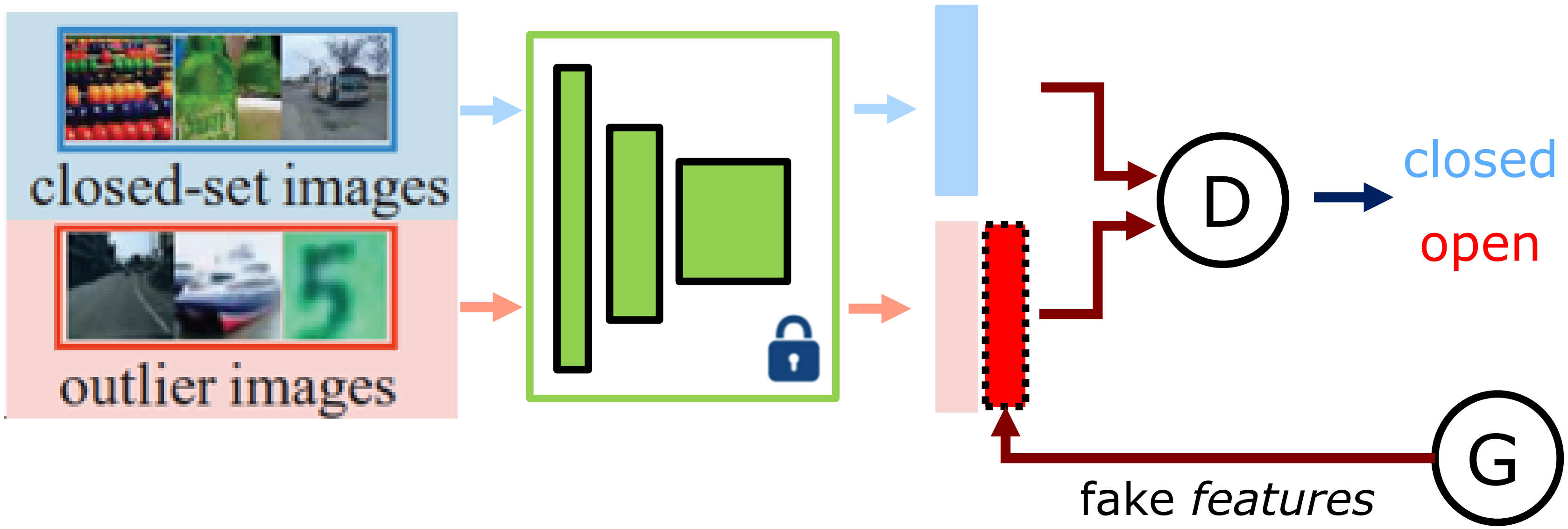


- Assume that not all relevant classes are known at **training time**
- **Aim:** Learn model to identify out-of-distribution (ODD) or anomalies

[Kong, 2021]

Image from [Kong, 2021]

OpenGAN



- Discriminator (D) distinguishes between known (closed) and unknown (open) examples
- Use generator (G) to generate additional open-set examples (besides some outlier images)

[Kong, 2021]

Image adopted from [Kong, 2021]

Open Challenges in Perception

- Many open challenges towards level 5 autonomous driving
  - Detection of rare events & situations
  - Challenging weather conditions
  - Adversarial attacks on perception systems

- How to deal with such situations gracefully & effectively?

Summary

- Looked at Perception stack & tasks
- Discussed CNNs as backbone of image-based perception
- Common approaches for detection, semantic segmentation and panoptic segmentation
- Challenges in visual perception and open problems

Thank you for your attention

References

[Cordts, 2016] Cordts et al. The Cityscapes Dataset for Semantic Urban Scene Understanding, CVPR, 2016.

[Cheng, 2020] Cheng et al. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-up Panoptic Segmentation, CVPR, 2020.

[Geiger,2012] Geiger et al. Are we ready for autonomous driving?, CVPR, 2012.

[Girshick, 2014] Girshick et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR, 2014.

[Kirillov, 2019] Kirillov et al. Panoptic Segmentation, CVPR, 2019.

[Kong, 2021] Kong et al. Open-Set Recognition via Open Data Generation, ICCV, 2021.

[Law, 2018] Law et al. CornerNet: Detecting Objects as Paired Keypoints, ECCV, 2018.

[Lin, 2014] Lin et al. Microsoft COCO: Common Objects in Context, ECCV, 2014.

[Krizhevsky, 2012] Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS, 2012.

[Redmon, 2016] Redmon et al. You Only Look Once: Unified, Real-Time Object Detection, CVPR, 2016.

[Ren, 2015] Ren et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NeurIPS, 2015.

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

[Zou, 2019] Zou et al. Object Detection in 20 Years: A survey, Arxiv, 2019.