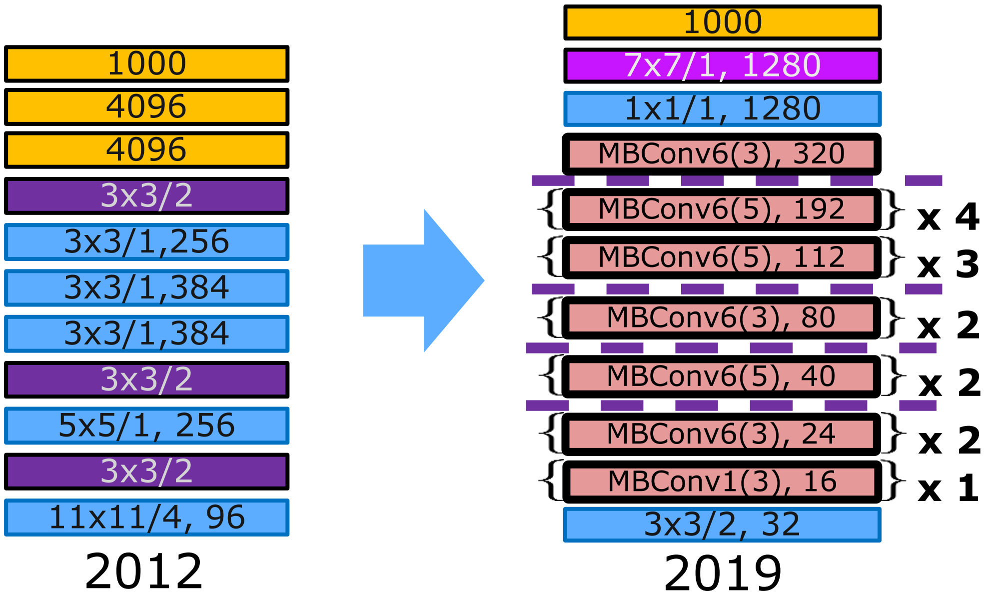
Photogrammetry & Robotics Lab

Machine Learning for Robotics and Computer Vision

Object Detection with CNNs

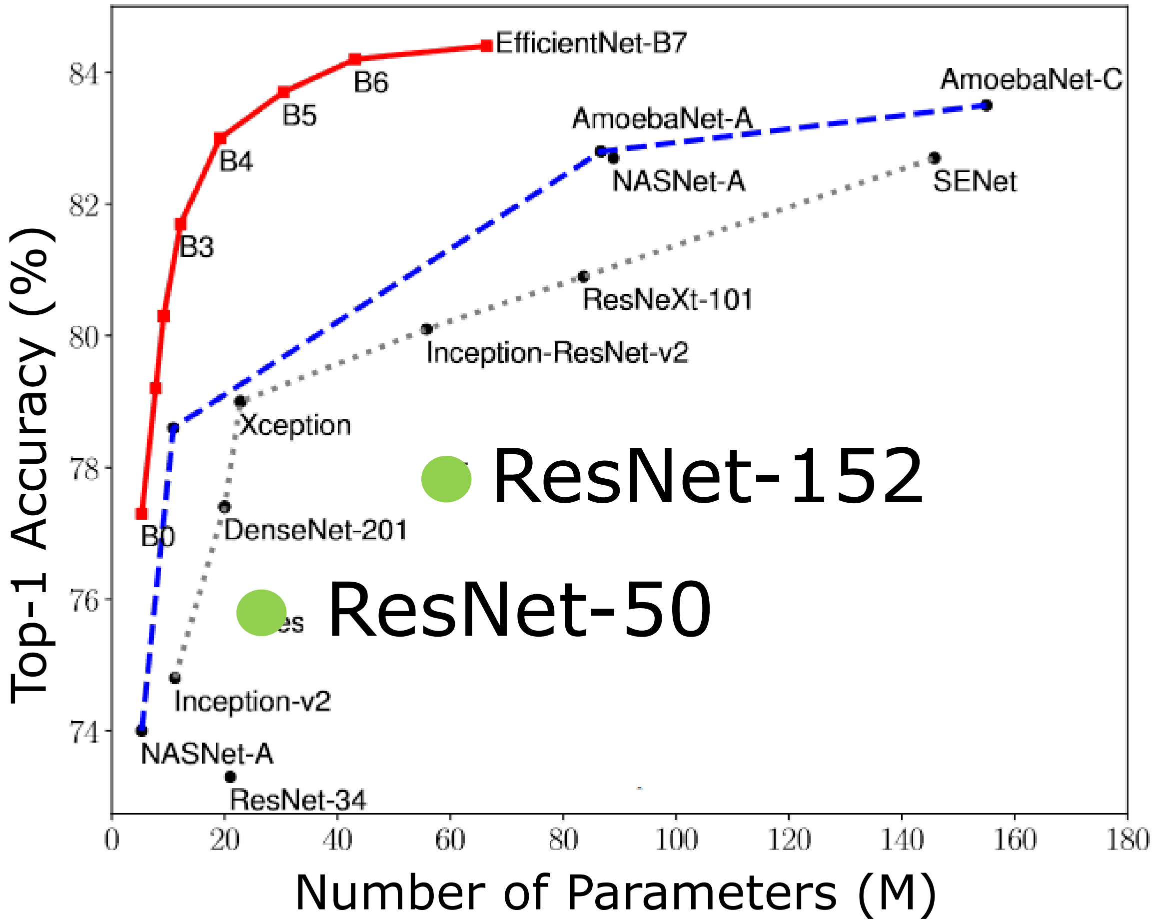
Jens Behley

Last Lecture: Building CNNs



- Popular and significant architectures and changes
  - global avg pooling → skip connection → efficiency
  - VGG → GoogleLeNet → ResNet → MobileNetV2 → EfficientNet
- Looked at common ways to close the gap between training and test performance (generalization gap)

Recap: Scaling EfficientNet



- Based on EfficientNet-B0, compound scaling (B1-B7) shows superior performance with smaller number of parameters

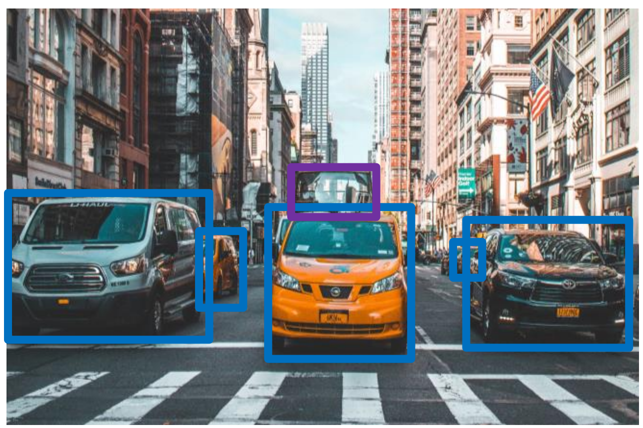
[Tan, 2019]

Figure from [Tan., 2019]

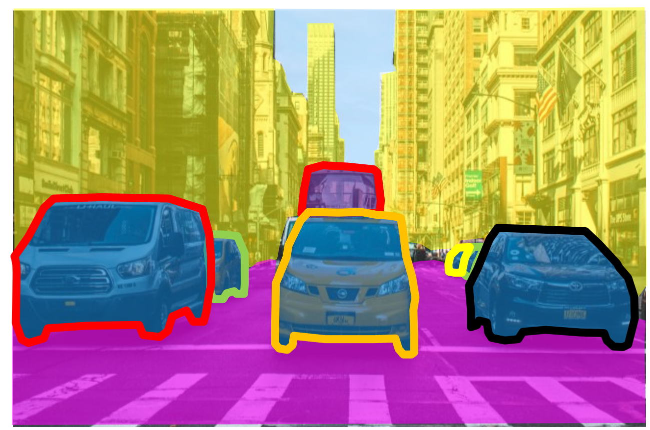
Perception Tasks

Classification

Semantic Segmentation



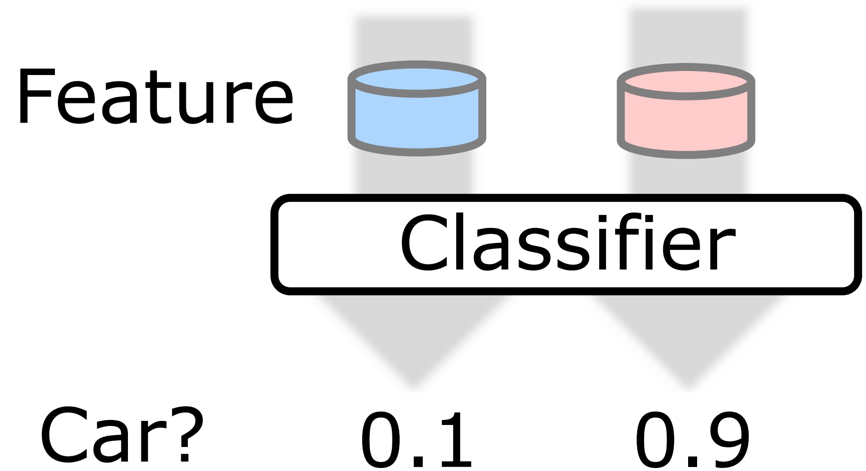
Object Detection



Panoptic Segmentation

Photo by Marisol Benitez on Unsplash

Anatomy of an Object Detector

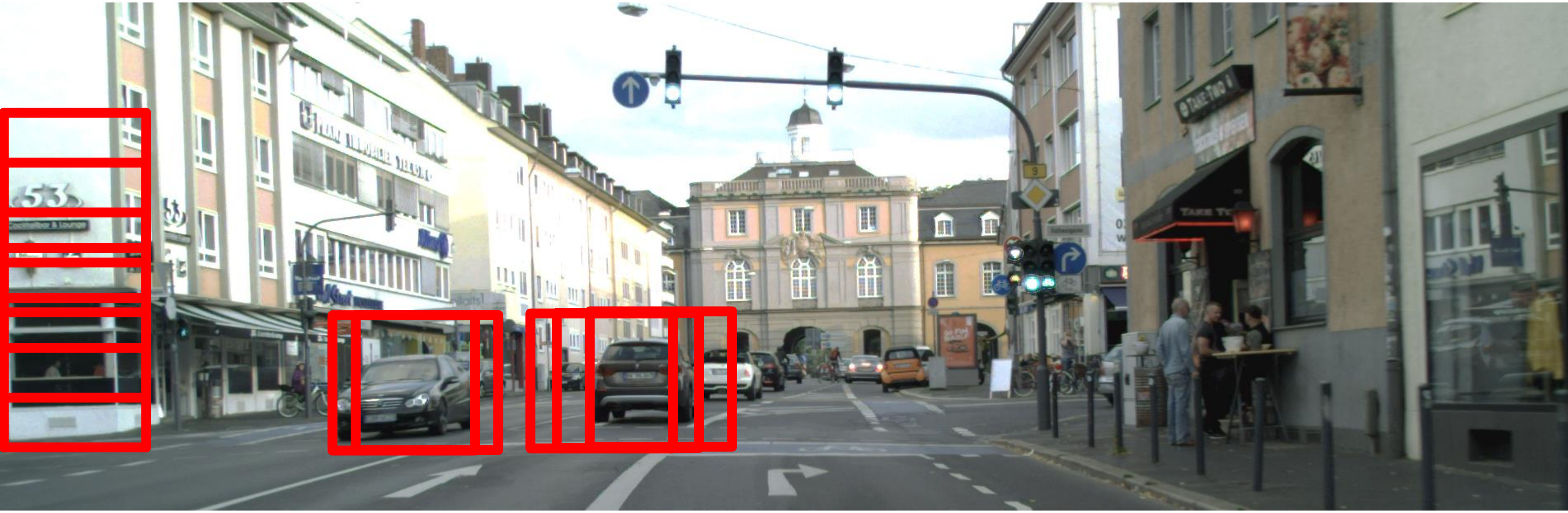


General Approach

1. Extract regions
2. Classify and score regions
3. Keep high scoring regions

Image from Cityscapes Dataset

Sliding Window Approaches

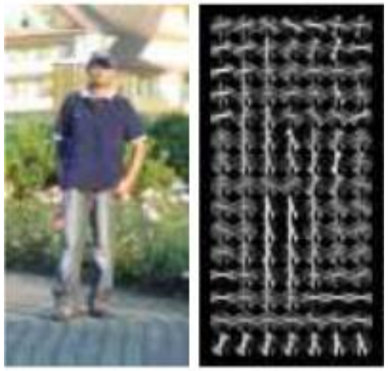


- Densely sample regions from image
- Classify image features extracted from the region

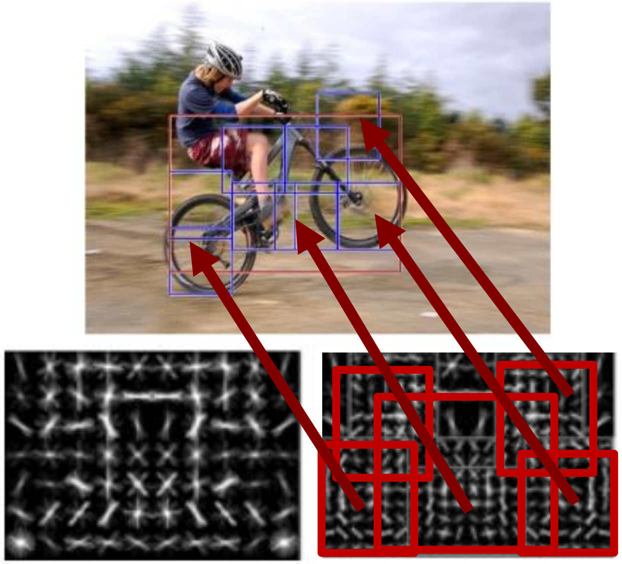
Image from Cityscapes Dataset

Traditional Object Detectors

Haar Features



Histogram of Gradients (HOG)

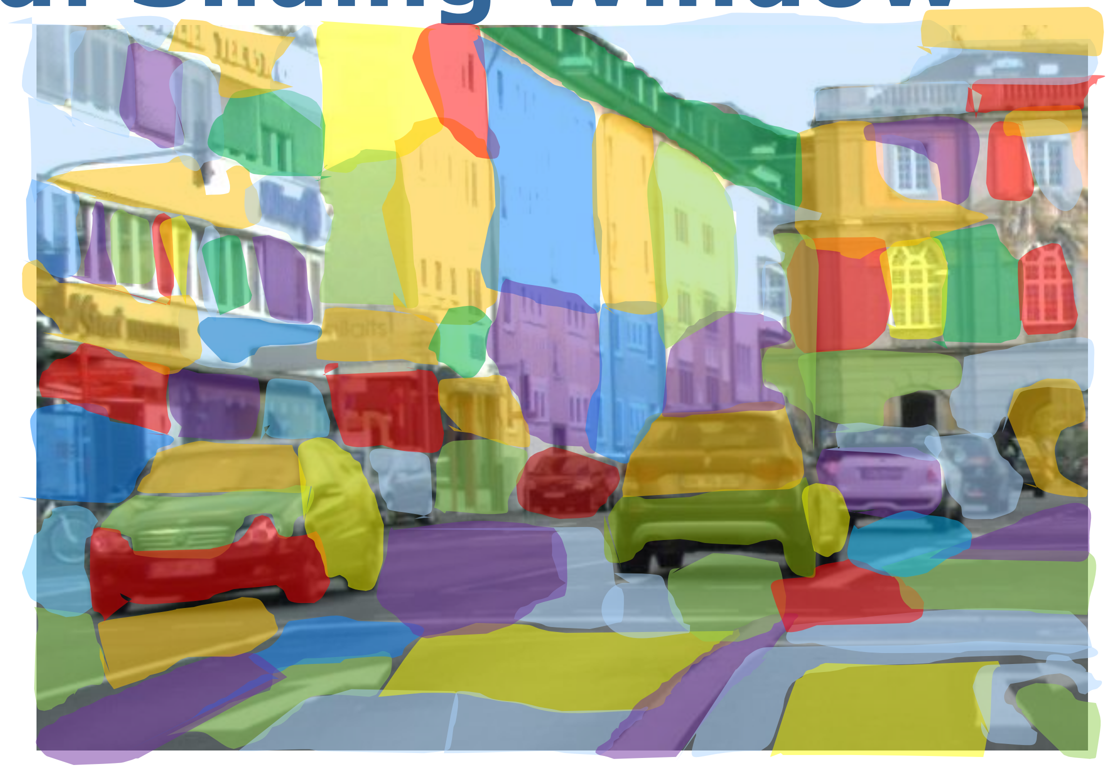


Deformable Part Model (DPM)

- Use aforementioned receipt: Sliding Window, Feature extraction, Classification of features
- Main innovations: better features

[Viola, 2001] [Dalal, 2005] [Felzenszwalb, 2008]

Beyond: Sliding Window



- Fine-to-coarse aggregation of super-pixel regions

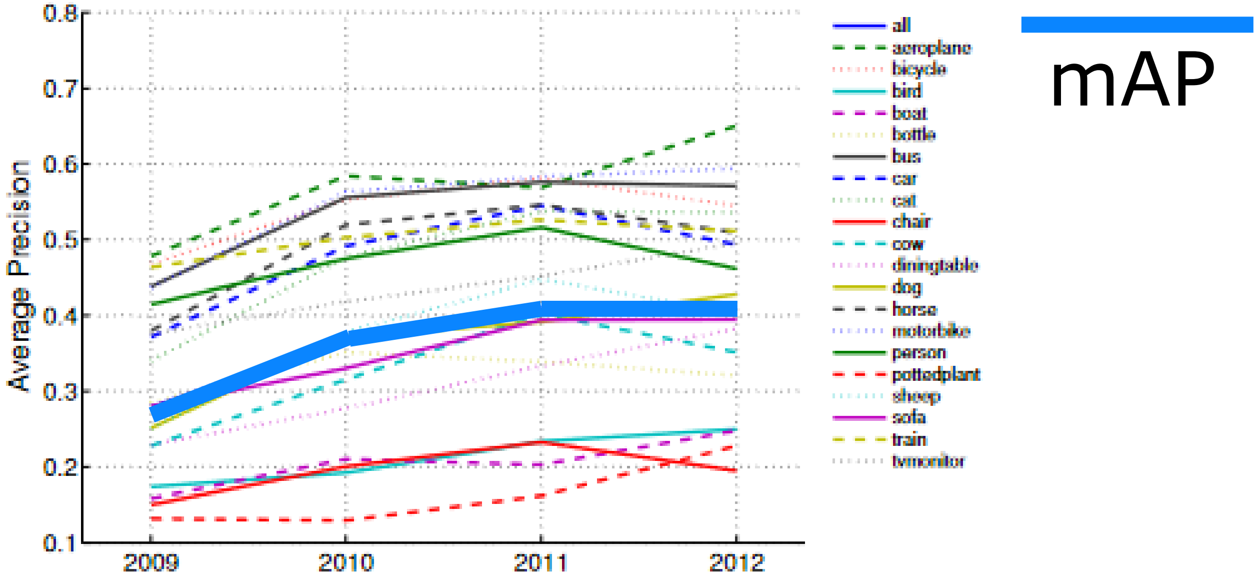
[van de Sande, 2011]

Selective Search

- Fine-to-coarse aggregation of super-pixel regions
- Far less **proposals** then sliding window
- Includes different scales

[van de Sande, 2011]

Pascal VOC Detection

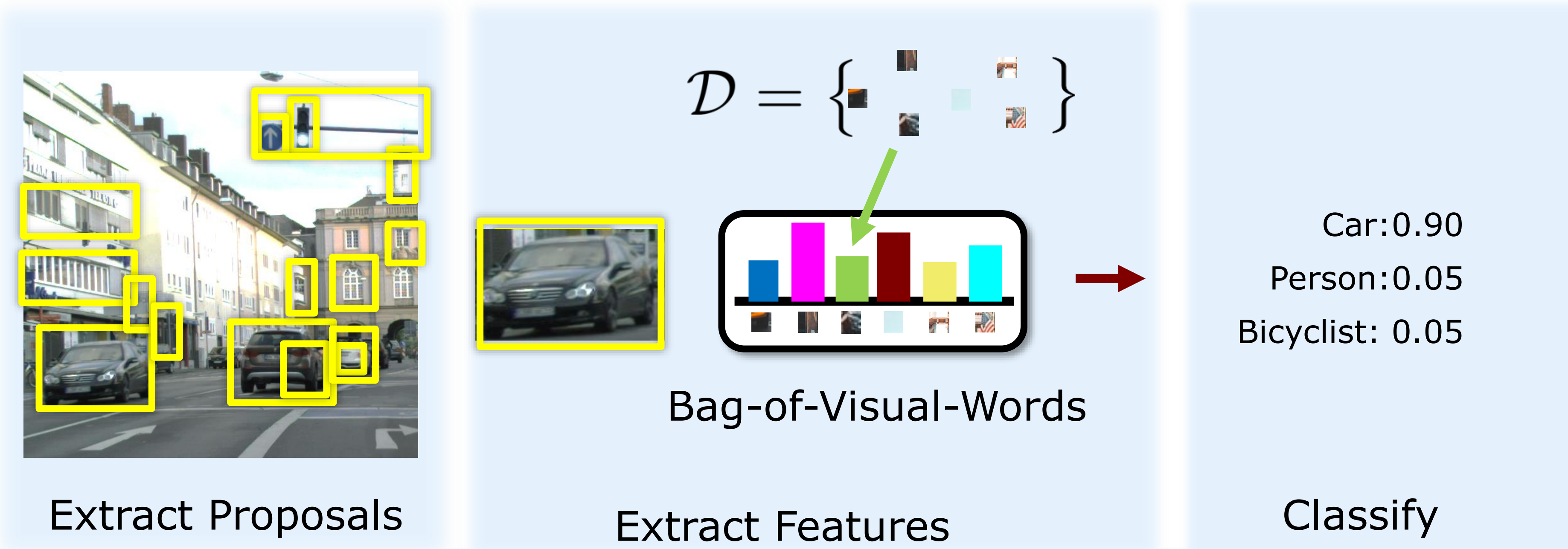


Challenge year

- 2008-2011 dominated by DPM-based methods: other features, re-scoring.
- 2012: Selective search with improved features

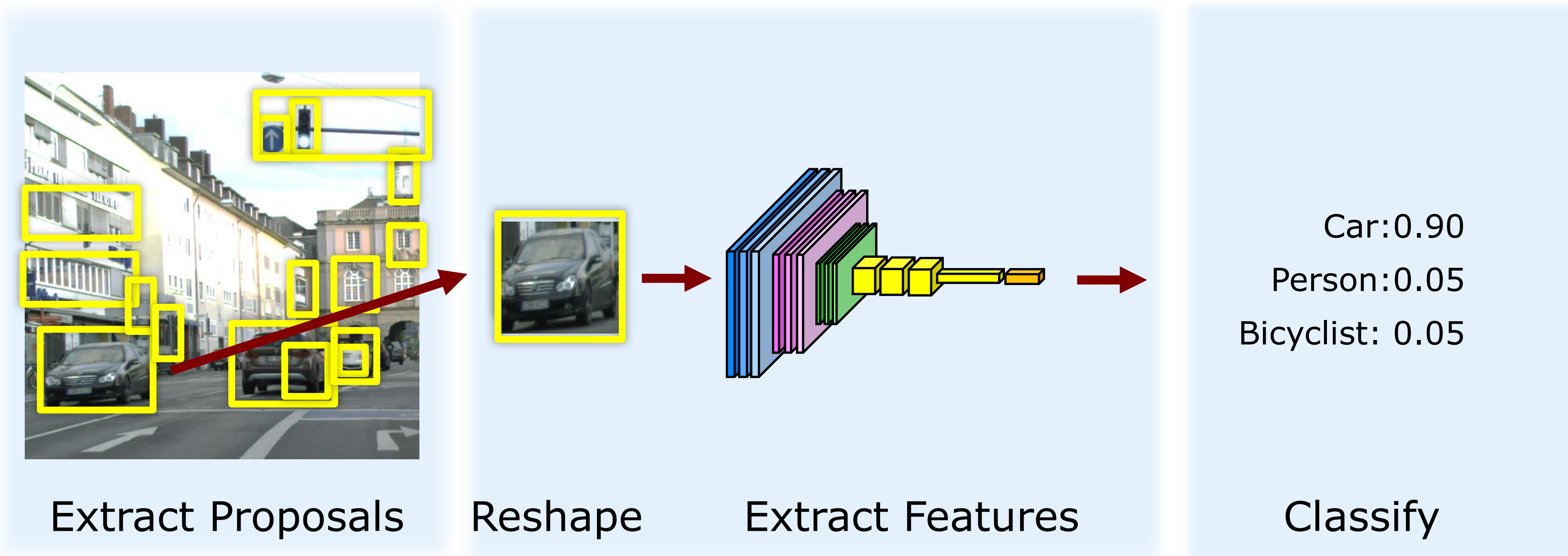
[van de Sande, 2011]

Traditional approach (~2012)



- Proposals → Features → Classification
- Obvious way to use to put CNNs to work?

R-CNN (2014)



- Selective Search for region proposals
- Compute feature using **CNN** (AlexNet/VGG-16)
- Class-wise linear SVM on features for classification
- Bounding box regression to refine classified bounding boxes

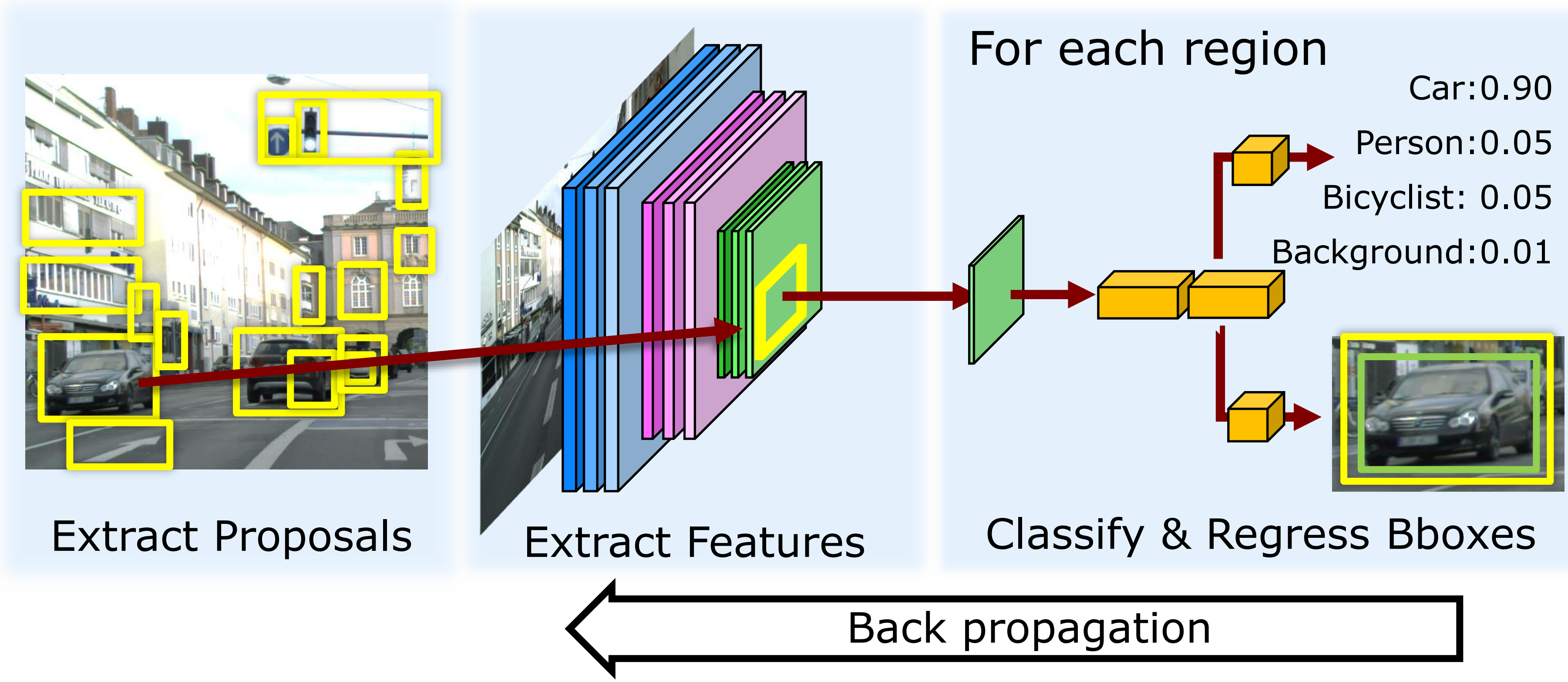
[Girshick, 2014]

Drawbacks of R-CNN

- Two separate training steps needed: first train CNN, then SVM on-top
- Training expensive: SVM on CPU needs features in amendable format
- Object detection is slow: 47 s / image

- Extracted features are pre-determined and not adapted

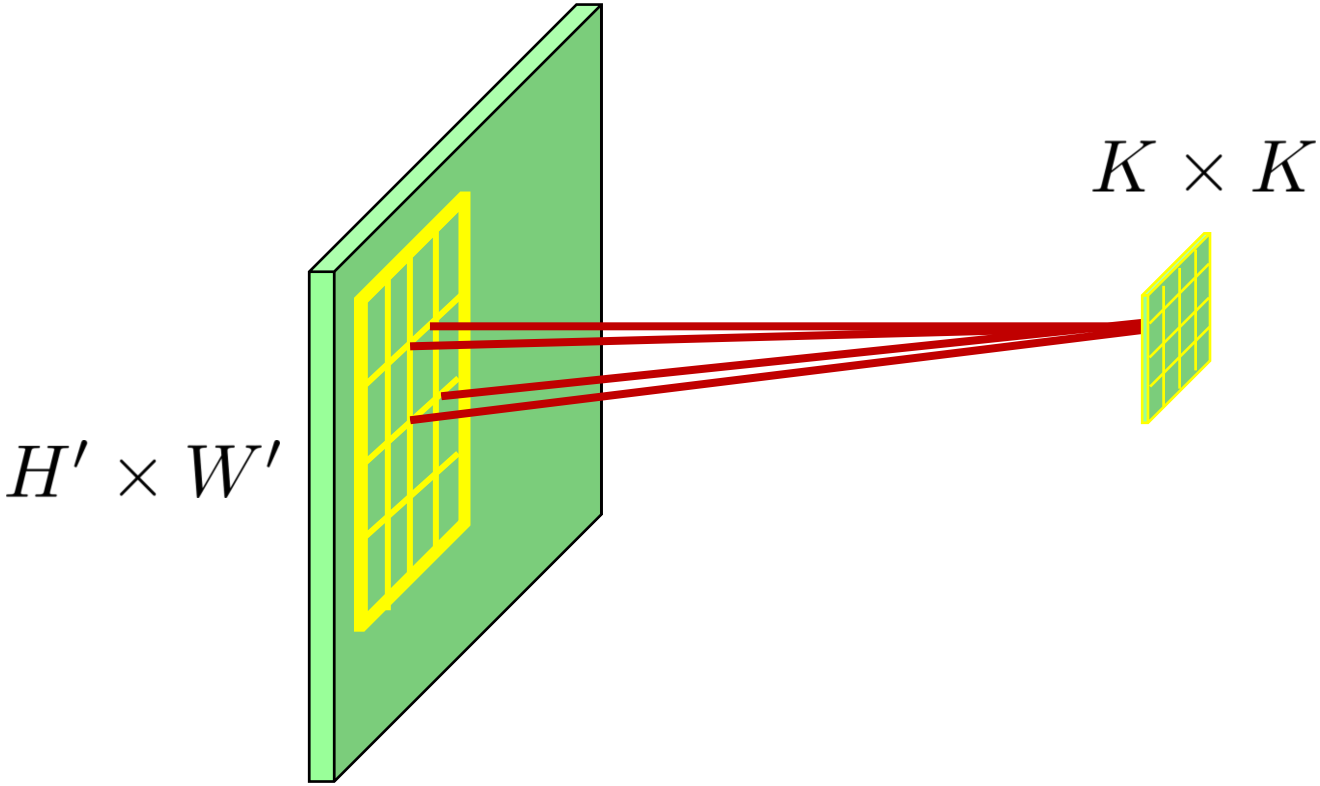
Fast R-CNN (2015)



- **Idea:** Extract features directly from feature map of complete image
- Needs only single forward pass through the CNN (e.g., AlexNet, VGG-16)!

[Girshick,2015]

Region-of-Interest(RoI) Pooling



- Adaptive max pooling brings extracted feature maps into appropriate size for RoI Network, e.g., K=7

[Girshick,2015]

Training: Loss

- Multi-task loss: Cross-entropy for class L_{cls} and the bounding box smoothed L1 loss L_{loc}:

$$\mathcal{L} = L_{\text{cls}}(p, p^*) + \lambda L_{\text{loc}}(v, v^*)$$

- L_{loc} only evaluated for positive examples (non-background):

$$L_{\text{loc}} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(v_i - v_i^*)$$

with

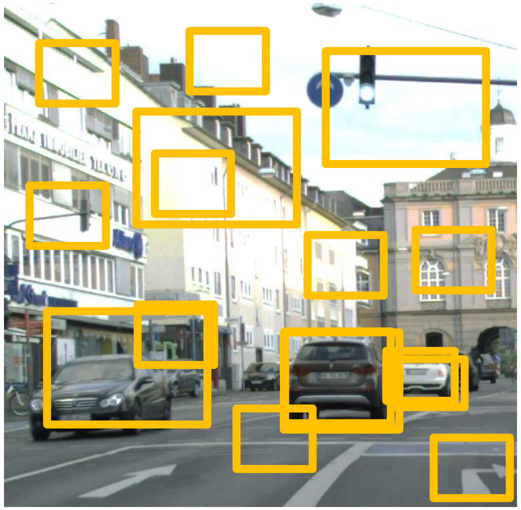
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & , if |x| < 1 \\ |x| - 0.5 & , \text{otherwise.} \end{cases}$$

Stage-wise Batch Sampling

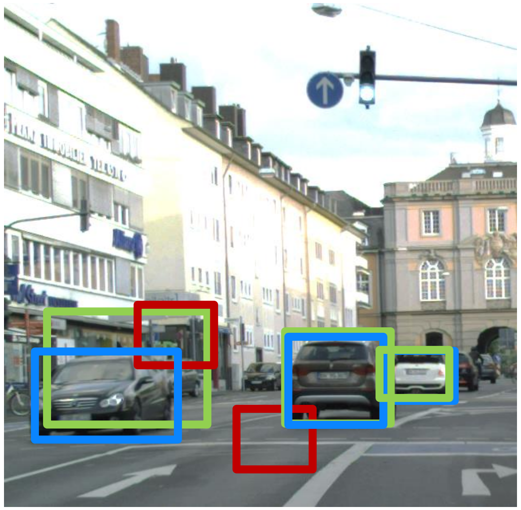
- Each image contains large number of proposals → impractical to use all!

- Sample proposals more efficiently:
  1. Select random images (e.g., N = 2)
  2. Sample R/N proposals from each image (e.g., R = 128) → batch size

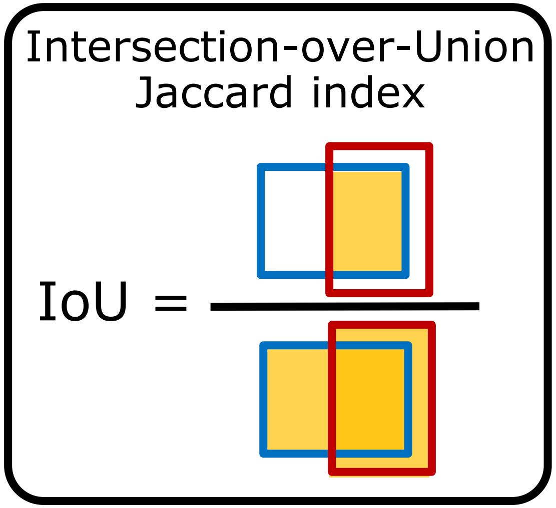
Stage-wise Batch Sampling (2)



Proposals

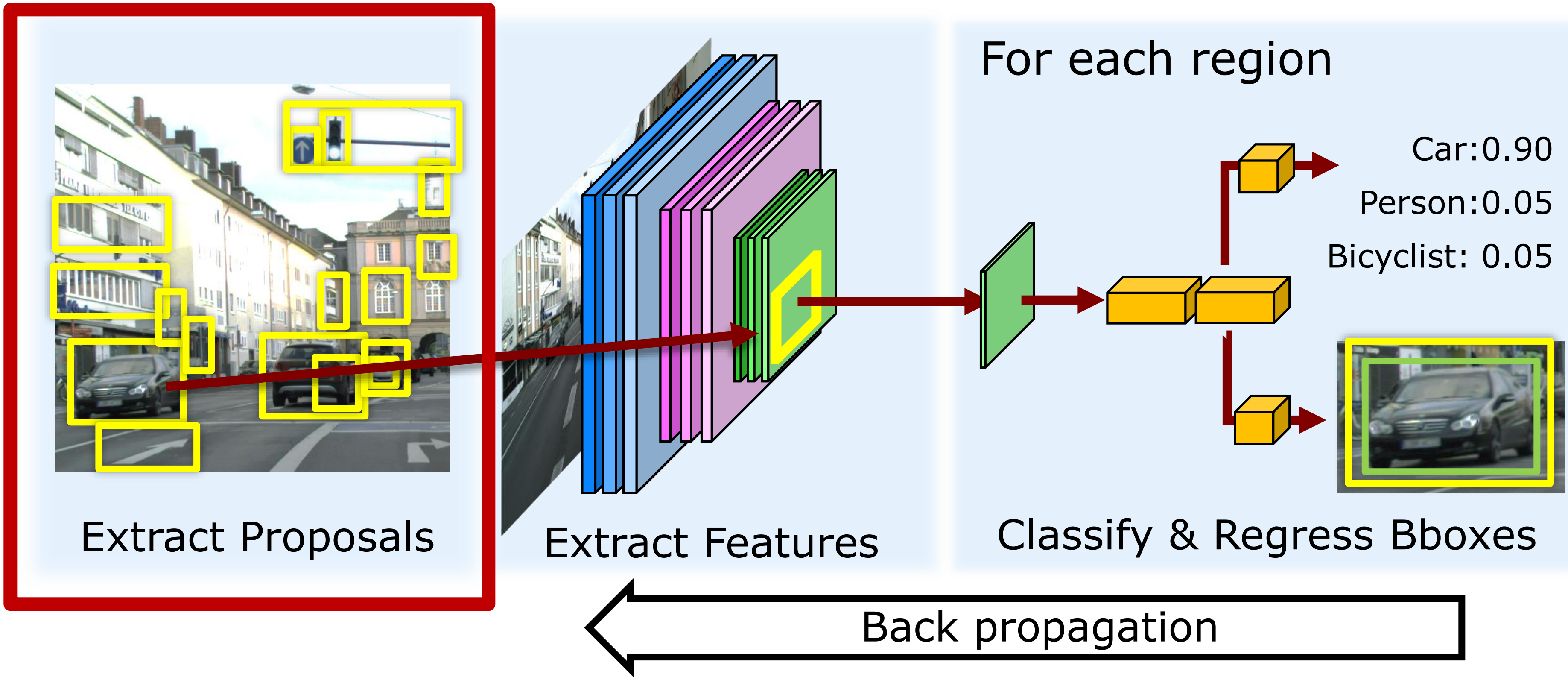


Selected Proposals



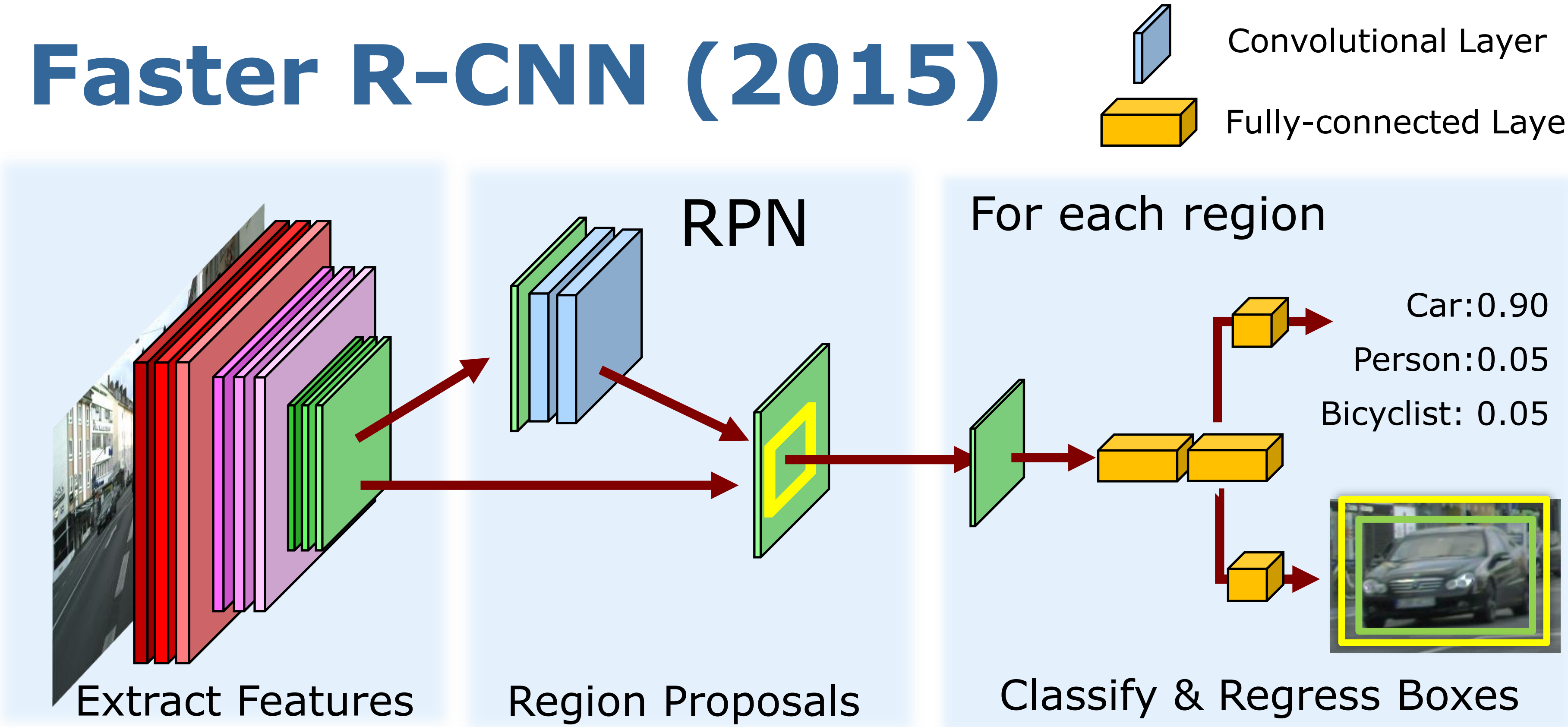
- Still more likely to have proposals that are background
- **Solution: stratified sampling**
  - 50% samples are **positive** (IoU > 0.5)
  - 50% samples are **background** (0.5 > IoU > 0.1)

Drawbacks of Fast R-CNN



- Region proposal still something externally generated ➔ computational bottleneck
- Still too slow for real-time applications

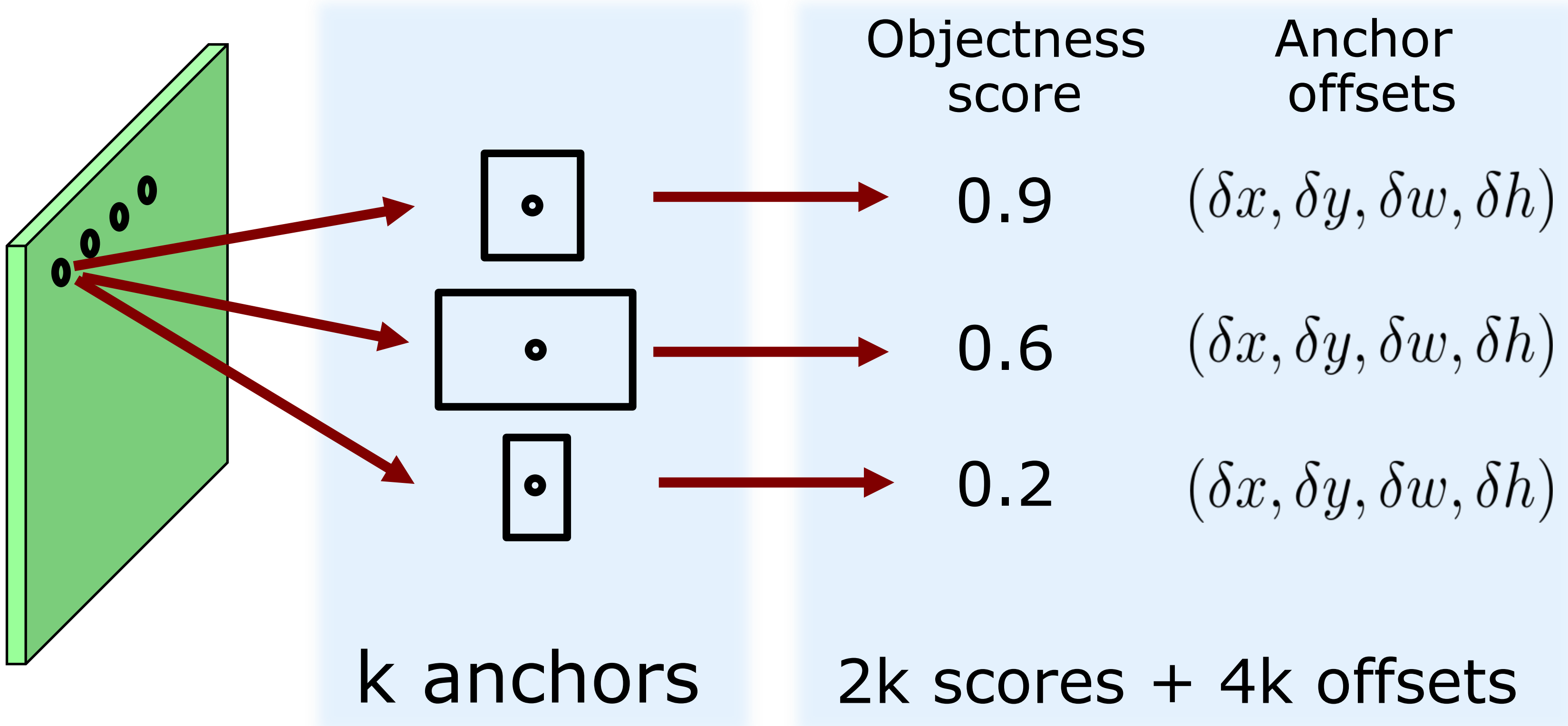
Faster R-CNN (2015)



- Replace proposal generation with CNN, the so called **Region Proposal Network** → faster
- Region-wise classification network uses same features as input as RPN (shared features)

[Ren, 2015]

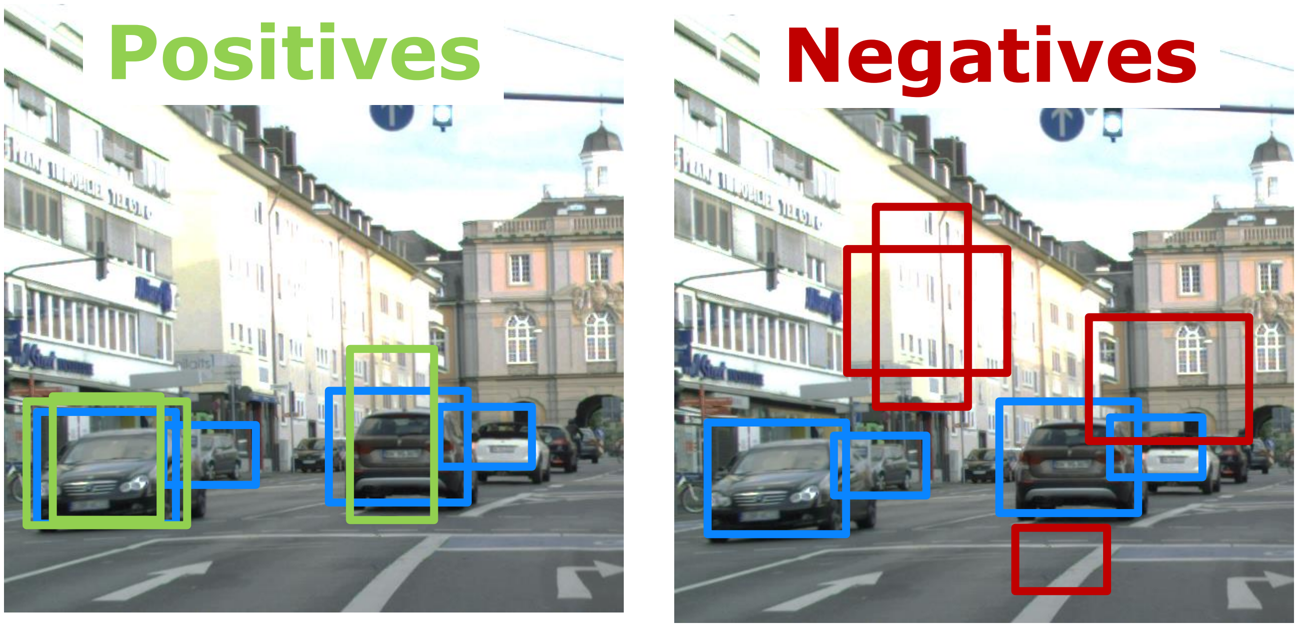
Region Proposal Network (RPN)



- Scores set of anchors with fixed initial sizes
- 3x3 conv + separate 1x1 conv for objectness and anchor offsets (for each anchor)
- Keep N-top scored anchors as RoI for Fast R-CNN

[Ren, 2015]

Anchor Assignment



- IoU-based assignment to determine positive vs. negative examples
  - Positive: highest or IoU > 0.7 with ground truth box
  - Negative: IoU < 0.3 for all ground truth boxes
- Usually far more negatives than positive boxes

Two-Stage Approach

- Note: There are still two stages involved!

- **First Stage:** Region Proposal Network (RPN) produces object/non-object scores with refined bounding box coordinates from anchors

- **Second Stage:** For N top-scored (objectness), the classification into different object classes and class-wise bounding-box regression is performed (Fast R-CNN RoI classifier)

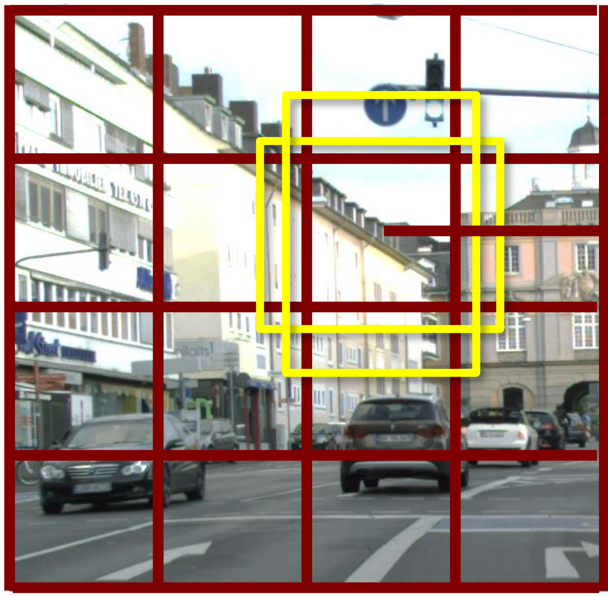
Faster R-CNN Summary

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

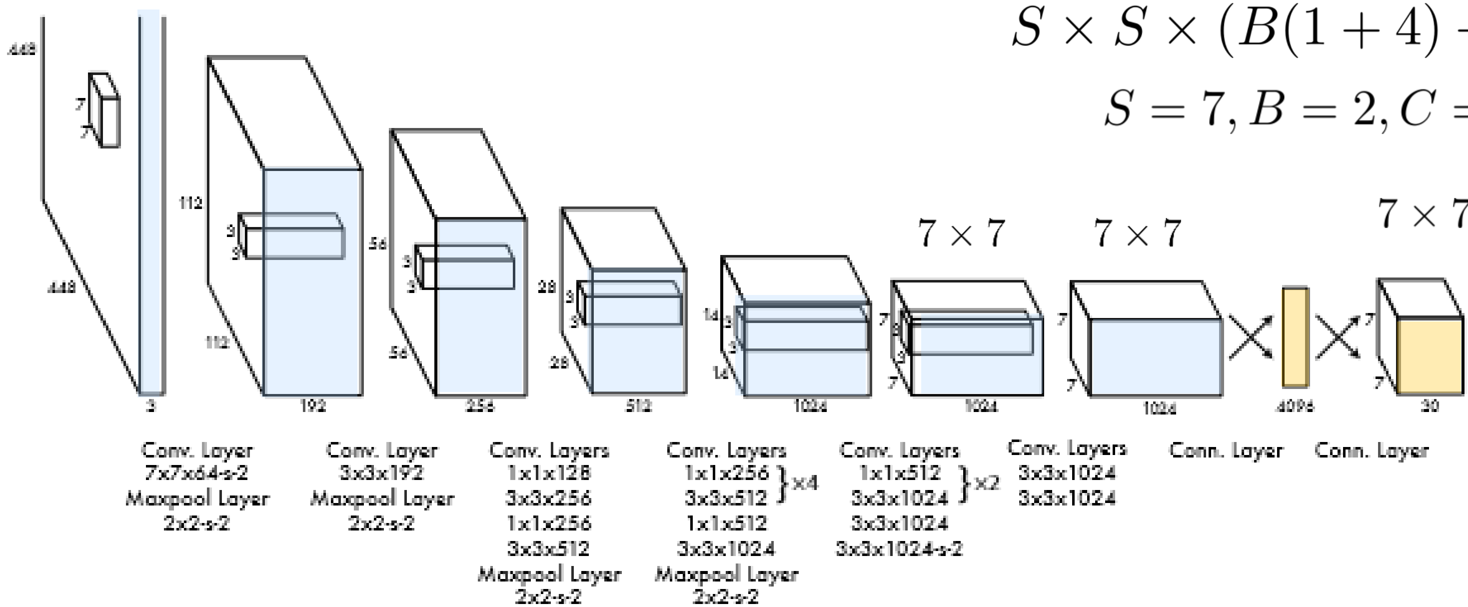
- Predicts bounding boxes with single forward pass
- Each anchor gets objectness score O, bounding box offsets
- Objectness + class score determines outcome

[Redmon, 2015]

YOLO Architecture

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

$$S = 7, B = 2, C = 20$$



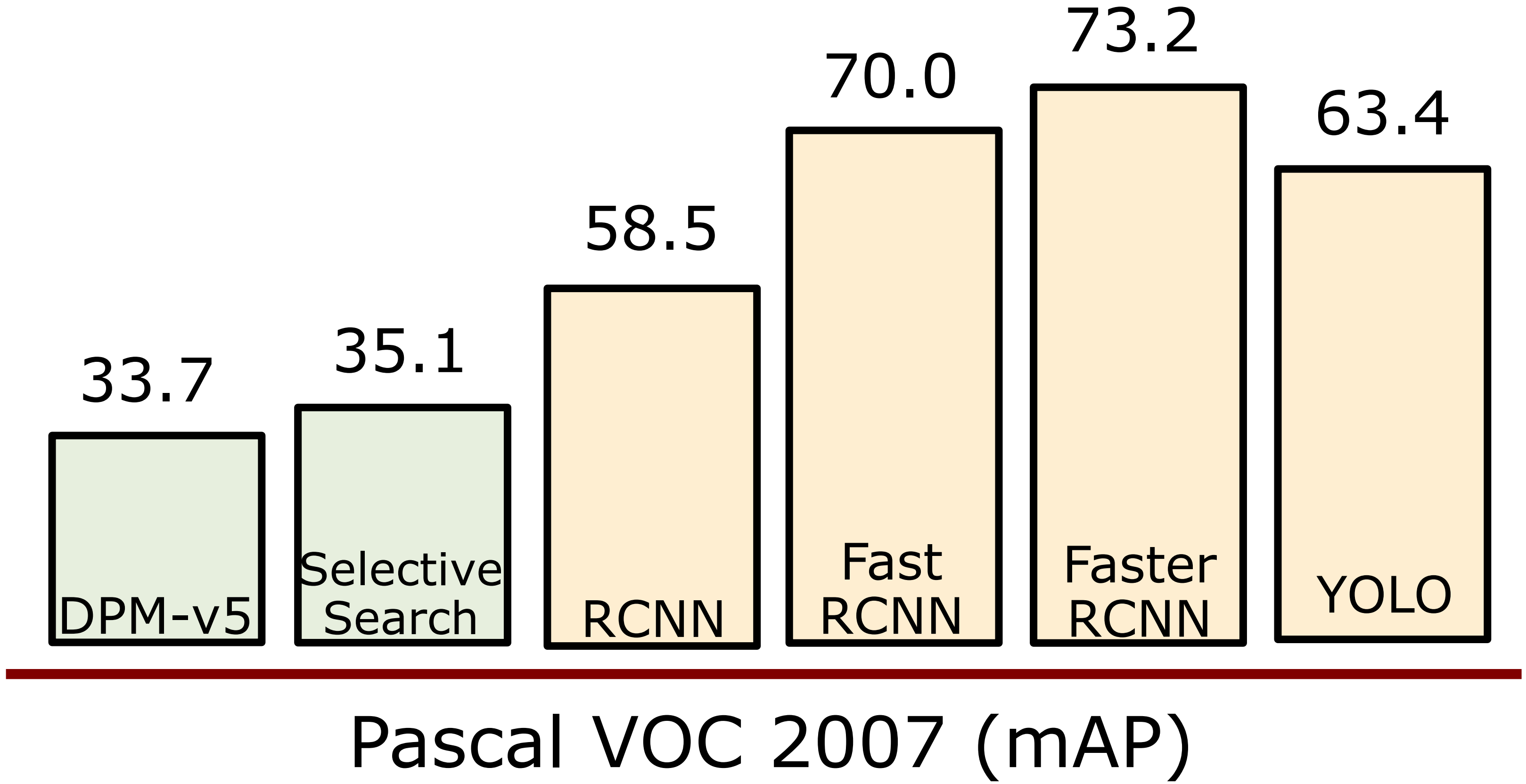
- YOLO uses a 24-layer convolutional network (DarkNet) with 2 fully connected layers 2 anchors, 20 classes (Pascal VOC)

[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

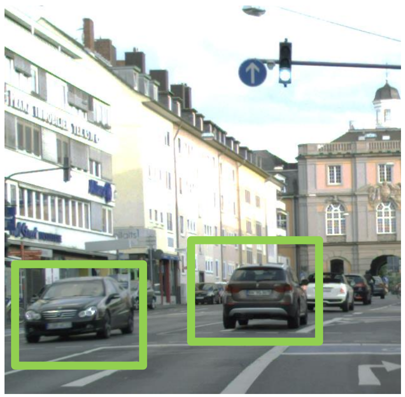
Comparison on Pascal VOC 2007



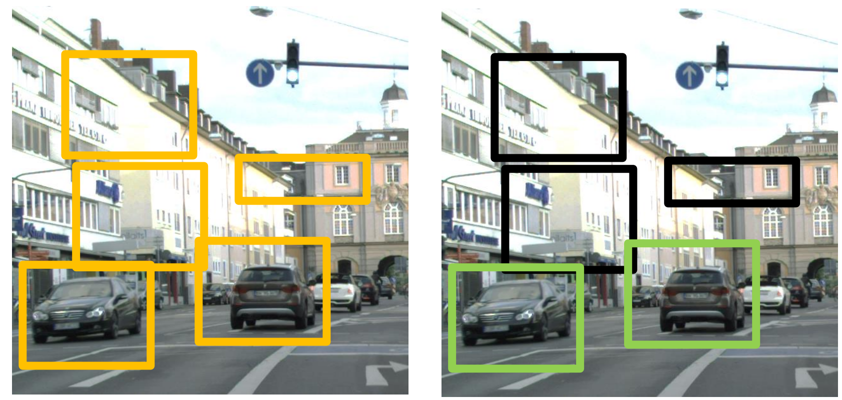
- Steep progress on Pascal VOC since usage of CNNs

Source: [Zou, 2019] [Redmon, 2015]

Single vs. Two-Stage Approaches



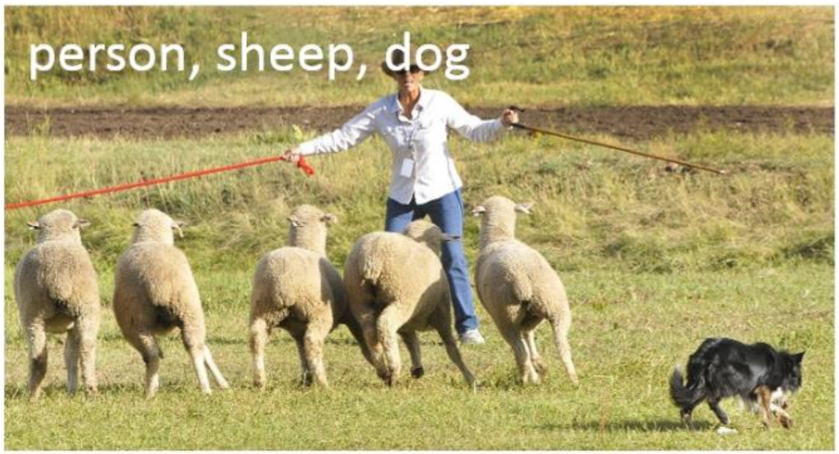
Single-stage

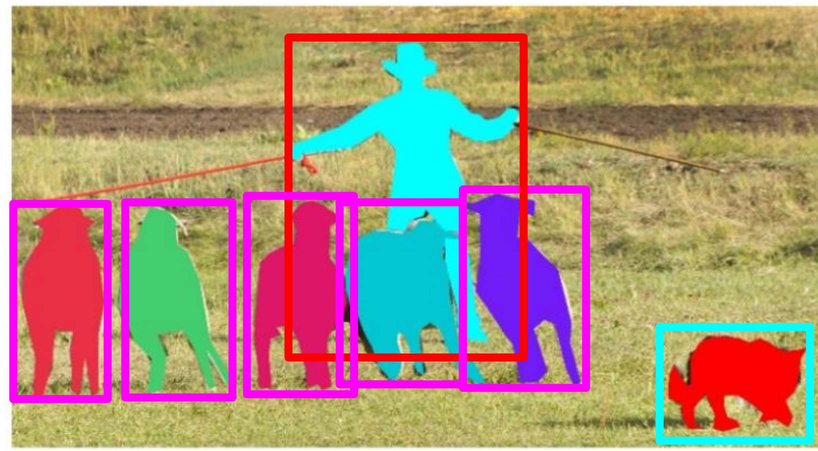


Two-stage

- Two paradigms for Object Detection:
  1. **Single-stage approaches**: Directly produces bounding boxes in single forward pass
  2. **Two-stage approaches:** First generates class-agnostic proposals and classifies only top N-proposals

MS Common Objects in Context (COCO)



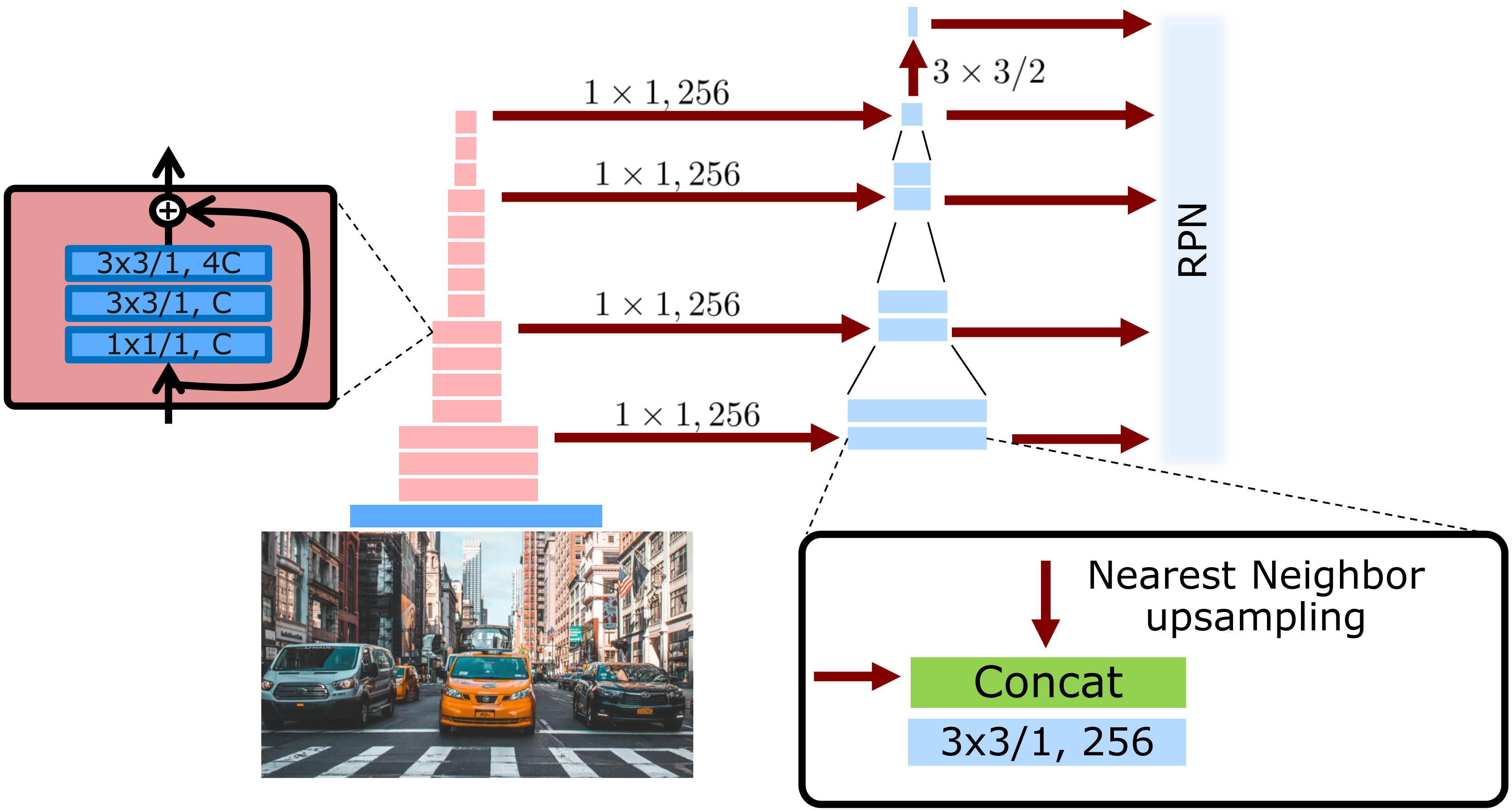


- Dense, i.e., pixel-wise annotation of objects
- 20 categories (Pascal) → 80 categories (COCO)
- Over 123,000 images, avg. 7.3 objects per image
- Replacing Pascal as standard dataset for detection

[Lin, 2014]

Figure from [Lin, 2014]

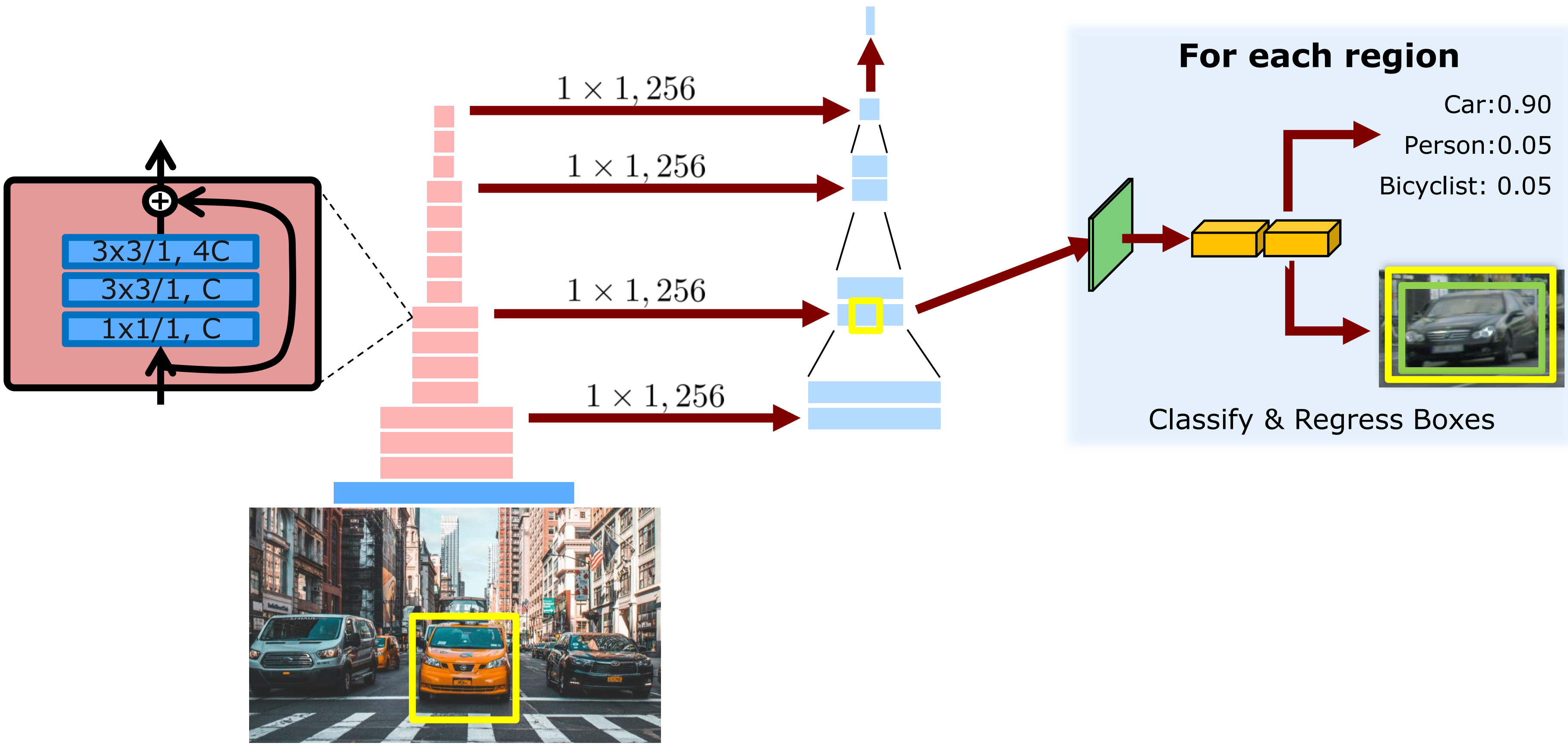
Feature Pyramid Networks (2017)



- Idea: Build feature pyramid from feature maps of last bottleneck feature maps and upsampled previous layers
- Apply RPN on each level of feature pyramid

[Lin, 2017]

Feature Pyramid Networks (2016)

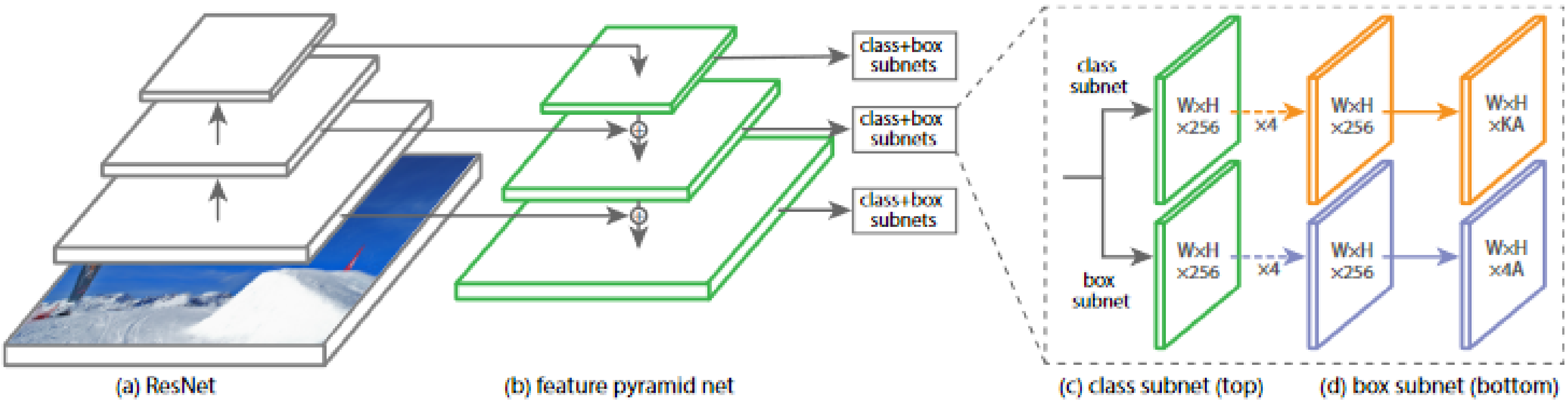


- Assign anchor to single feature map in pyramid
- As before: Apply Fast R-CNN classification on top-scoring region proposals

[Lin, 2017]

RetinaNet (2017)

3x3 convolutions

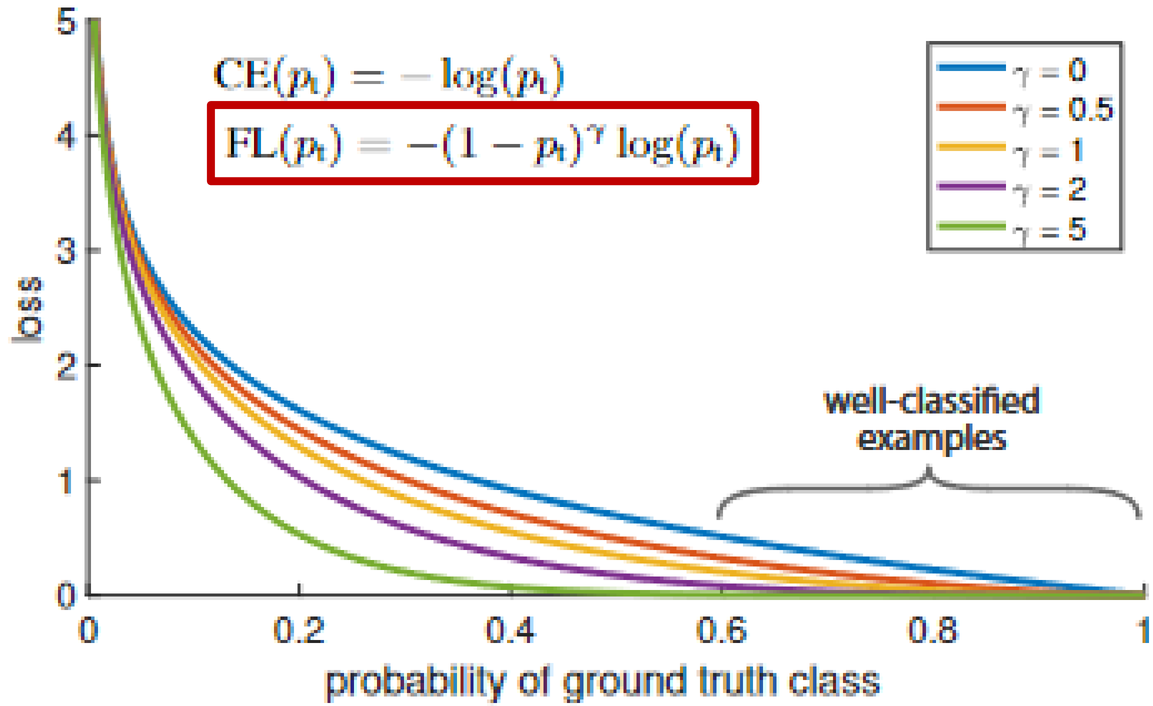


- Extension of FPNs to **single-stage approach**
- As with YOLO: apply directly classification + regression on feature pyramid level; here 9 anchors per level with level-dependent size
- **Problem:** many negative anchors, few positives

[Lin, 2017]

Figure from [Lin, 2017]

Focal Loss (2017)

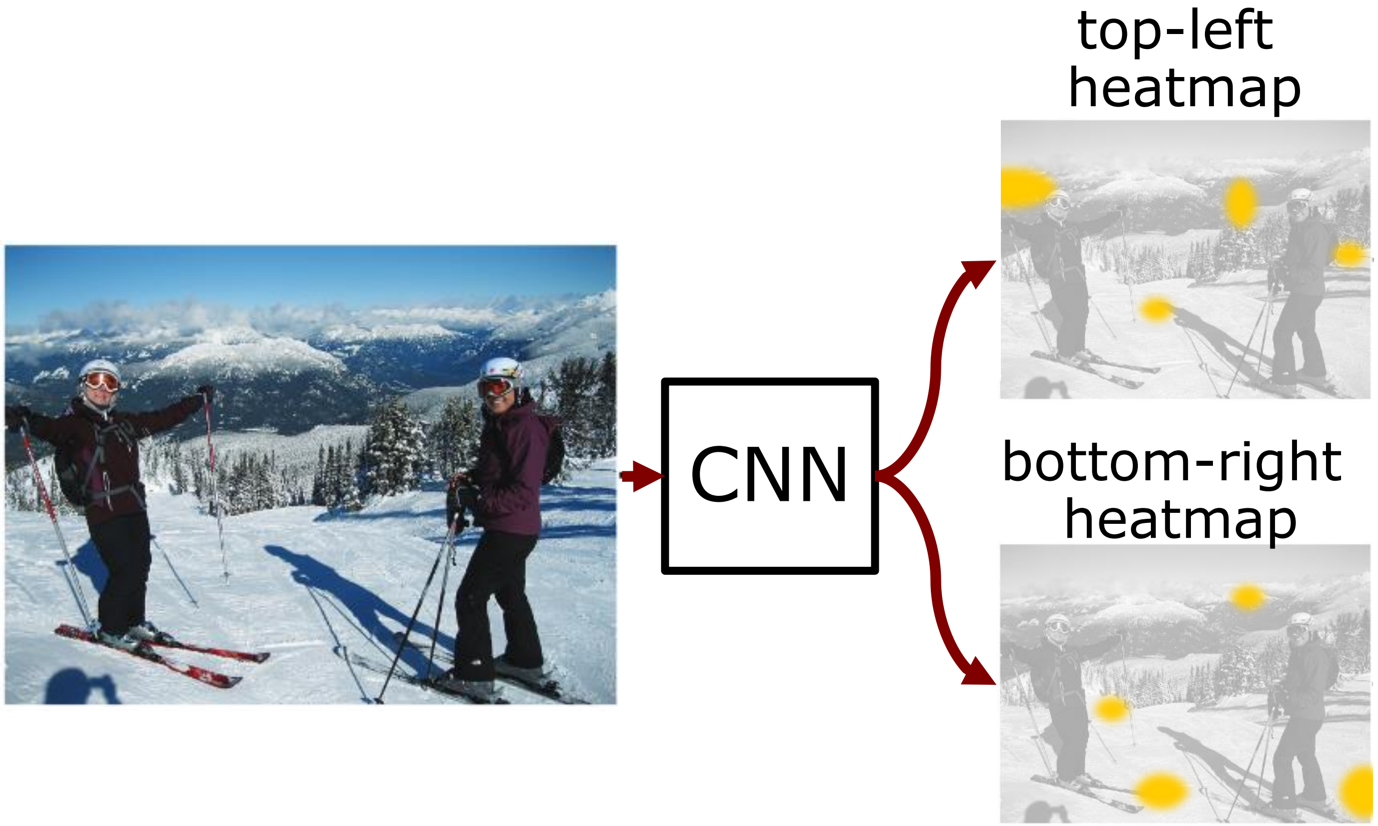


- This imbalance of positive and negatives reduces performance of detector
- **Focal loss** addresses this by modifying the cross-entropy loss by weighting with confidence

[Lin, 2017]

Figure from [Lin, 2017]

Bounding boxes from corners

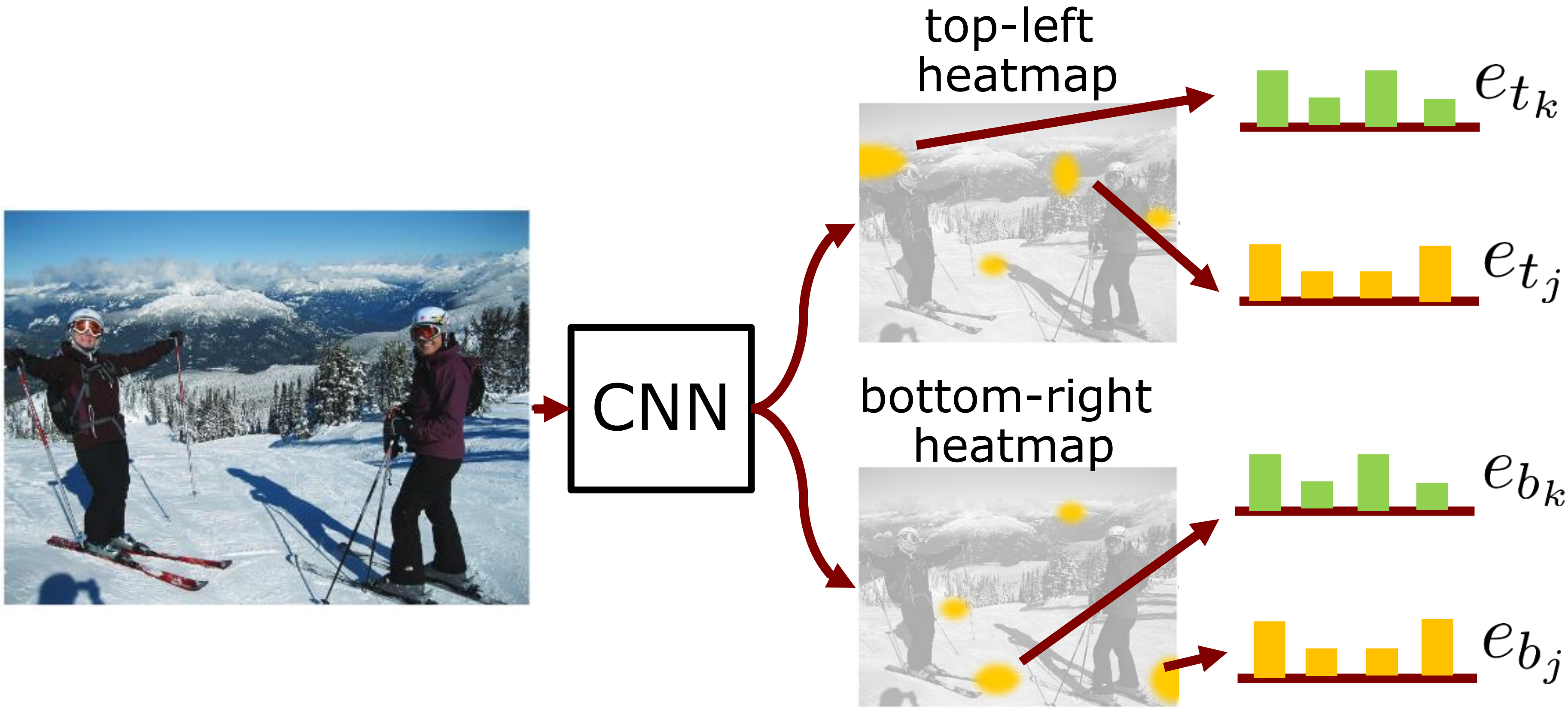


- Instead of scoring of anchors, CornerNet determines corners of bounding boxes
- Produce heatmaps of likelihood that at given pixel is upper-left or bottom-right corner

[Law, 2018]

Photo from [Law, 2018]

How to associate corners?

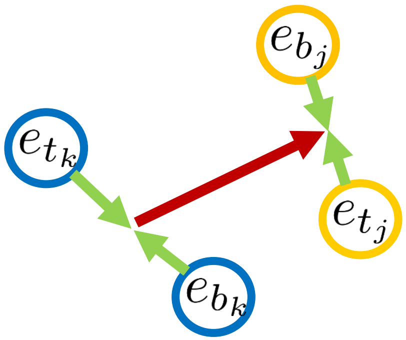


- To associate top-left and bottom-right corners, CornerNet determine embedding (“features”)
- Similar embeddings correspond to the same object

[Law, 2018]

Embedding Loss

$$L_{pull} = \frac{1}{N} \sum_{k=1}^{N} \left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2 \right],$$



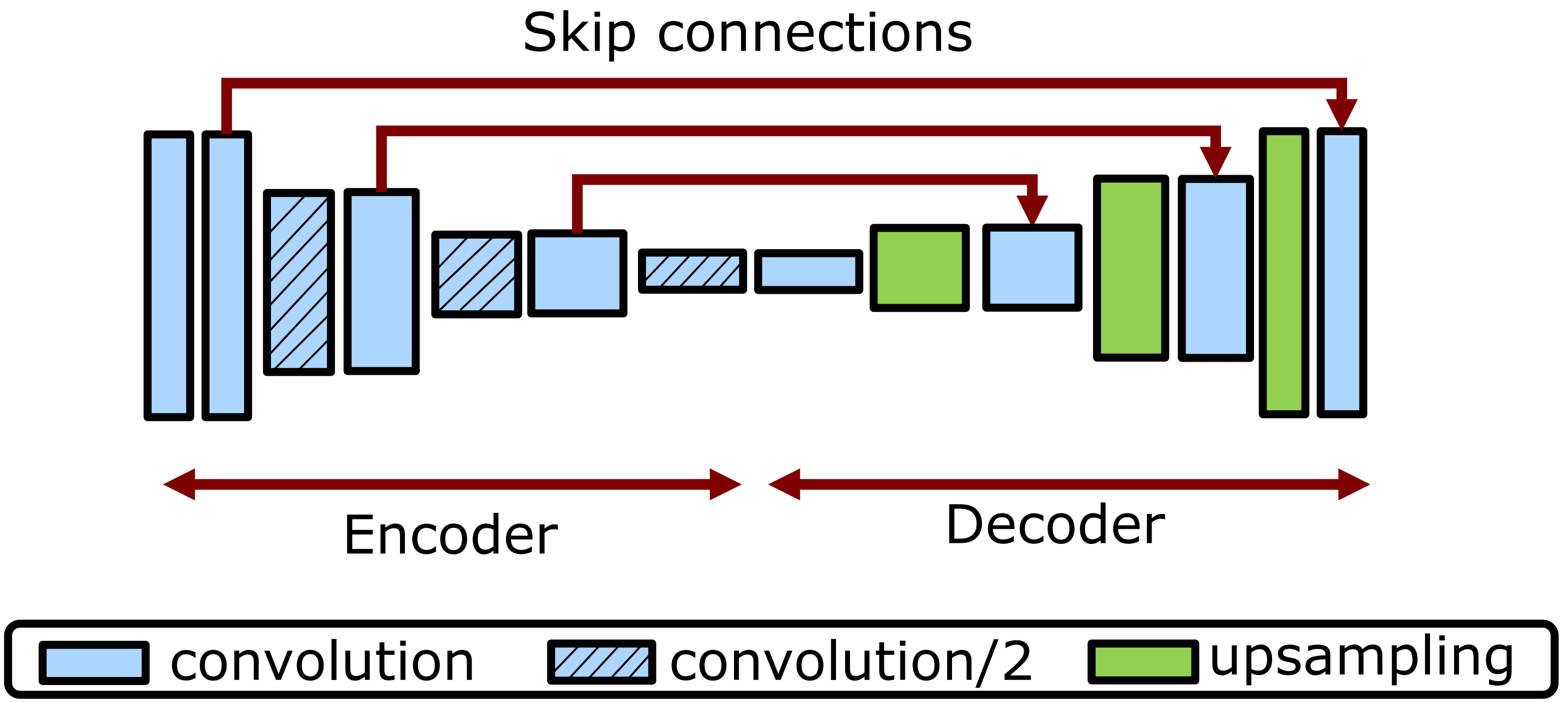
$$L_{push} = \frac{1}{N(N-1)} \sum_{k=1}^{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \max(0, \Delta - |e_k - e_j|),$$

$$e_k = \frac{e_{t_k} + e_{b_k}}{2}$$

- Embedding vectors for corresponding corners are pulled together
- Embedding vectors of all other corners should be pushed away
- Δ is a margin between dissimilar corners

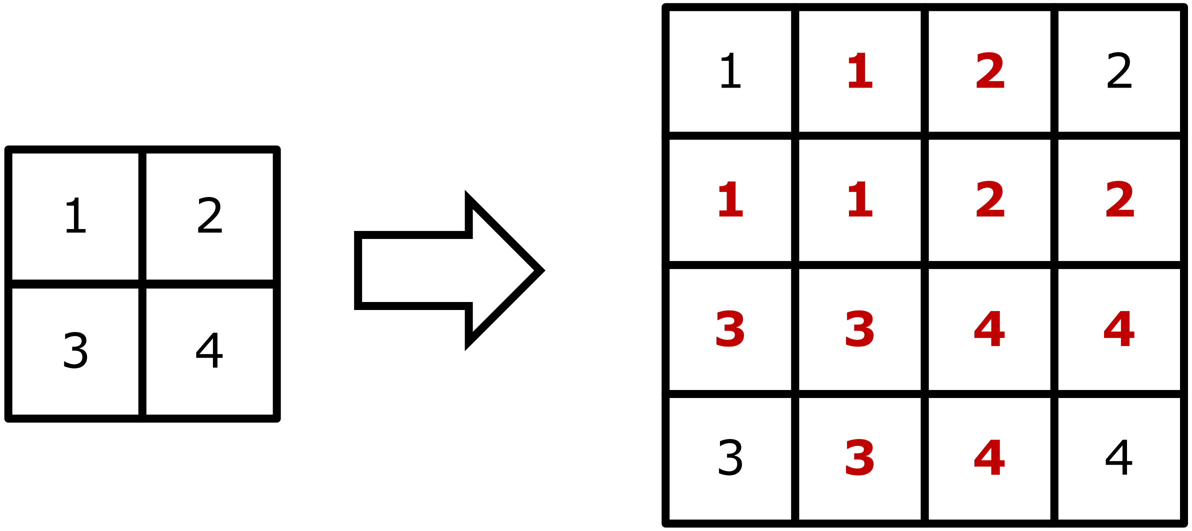
[Law, 2018]

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

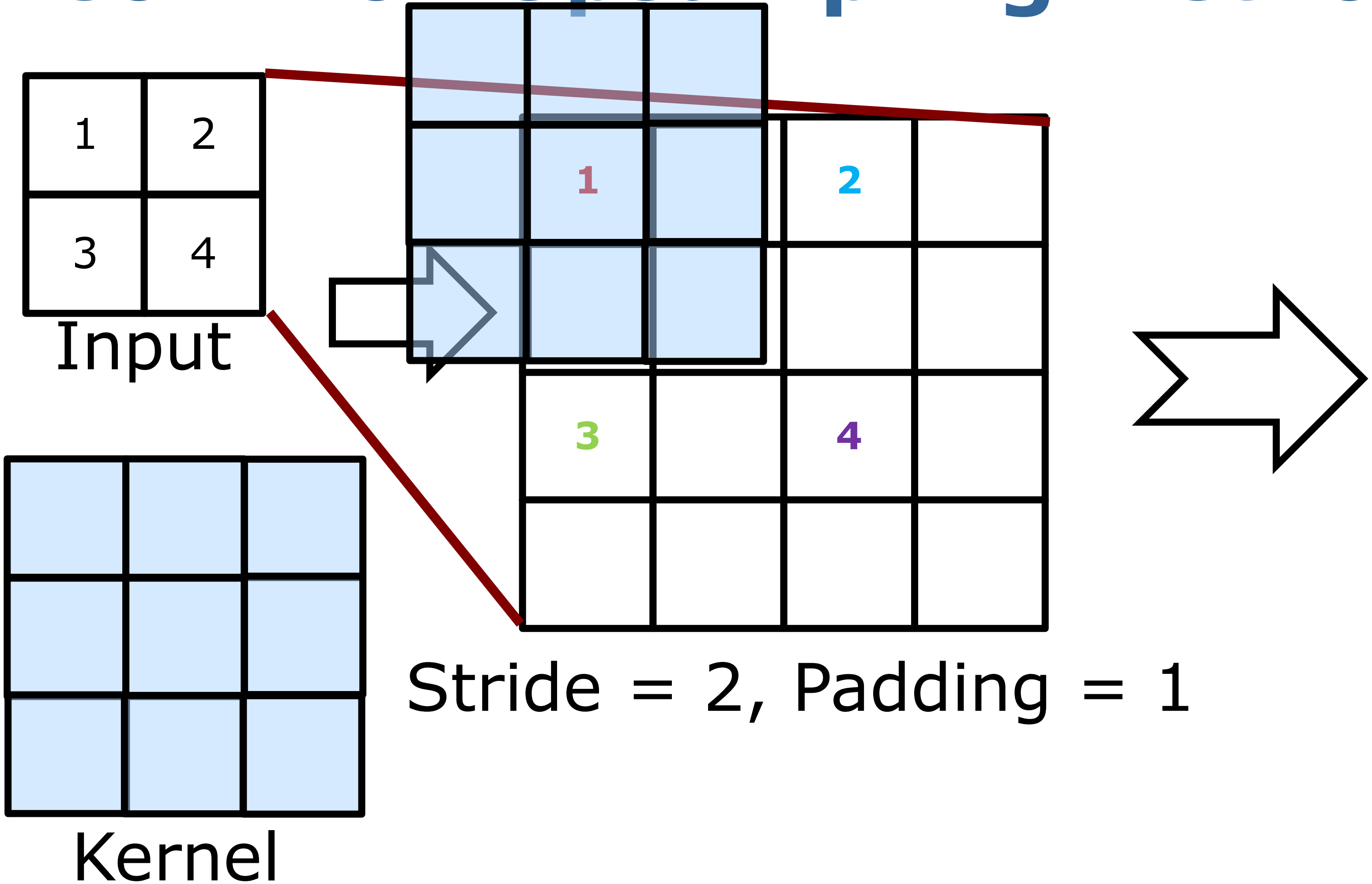
Common Upsampling Methods

1	2
3	4

1	**1.25**	**1.75**	2
1.5	**1.75**	**2.25**	**2.5**
2.5	**2.75**	**3.25**	**3.5**
3	**3.25**	**3.75**	4

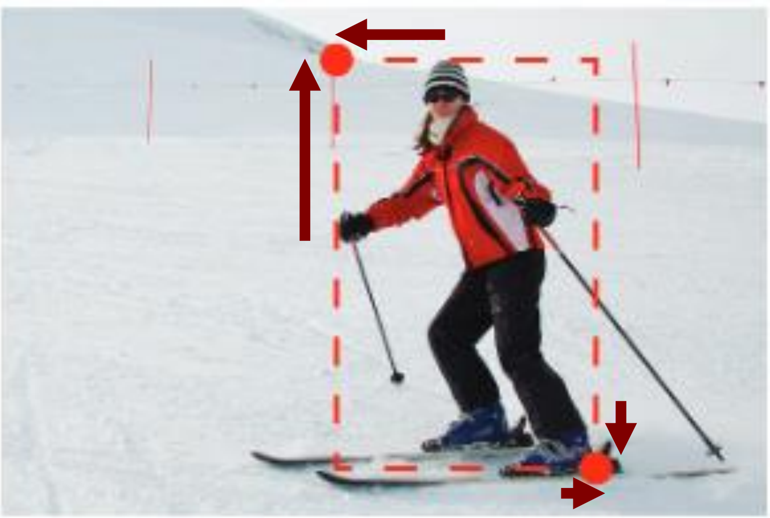
- **Bilinear upsampling** interpolates values in between using bilinear interpolation

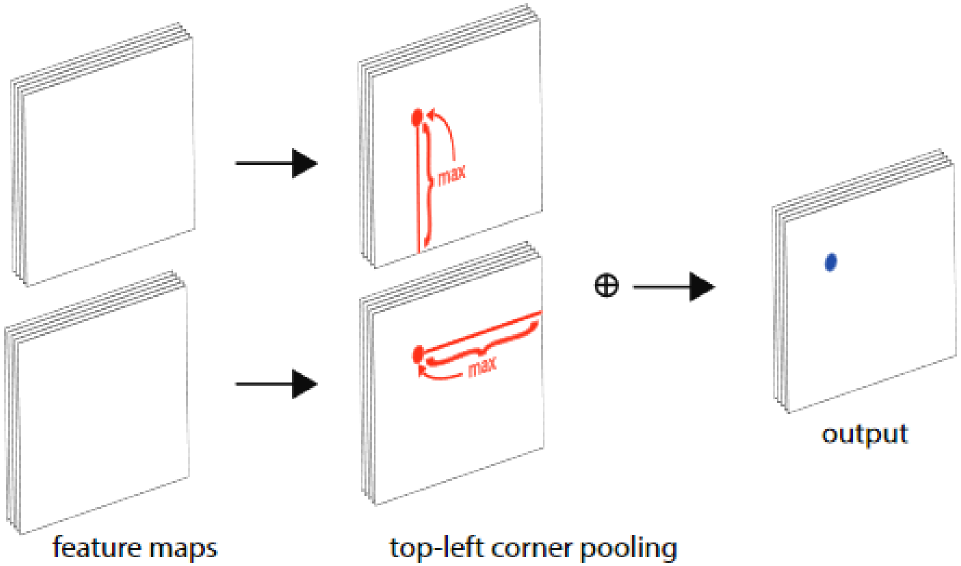
Common Upsampling Methods



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

Corner Pooling

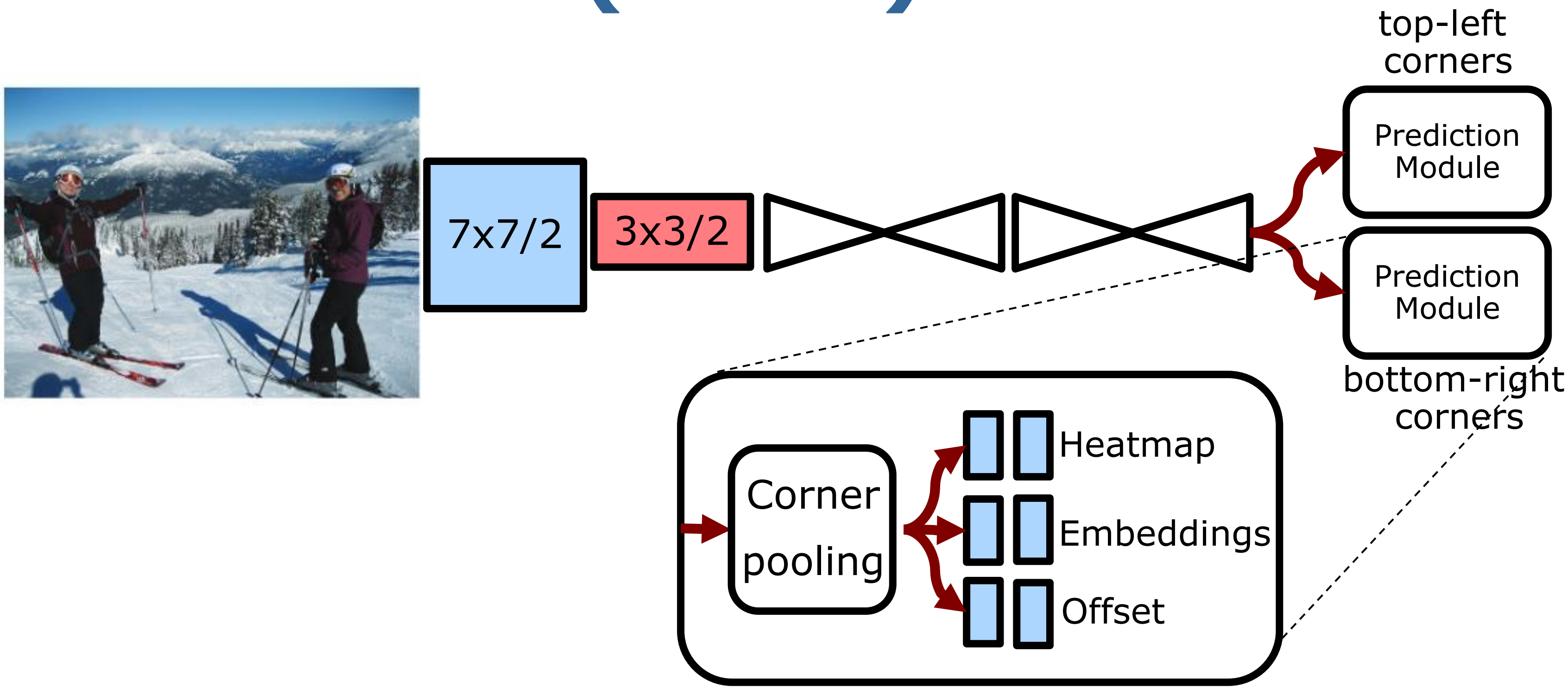




- Relevant features of object might be far away
- **Idea:** Aggregate features for top-left/bottom-right corner from bottom/right and top/bottom features
- **Max pooling** over features maps

Figures from [Law, 2018]

CornerNet (2018)

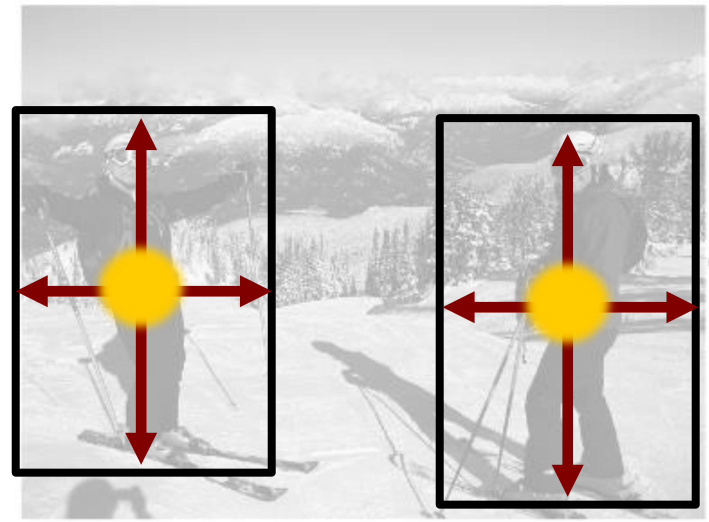


- CornerNet uses two encoder-decoder networks on 4 times reduced features maps
- Heatmaps, Embeddings, and offsets are computed by 3x3 Convs + 1x1 Convs (for K classes)

[Law, 2018]

Photo from [Law, 2018]

Objects as Points

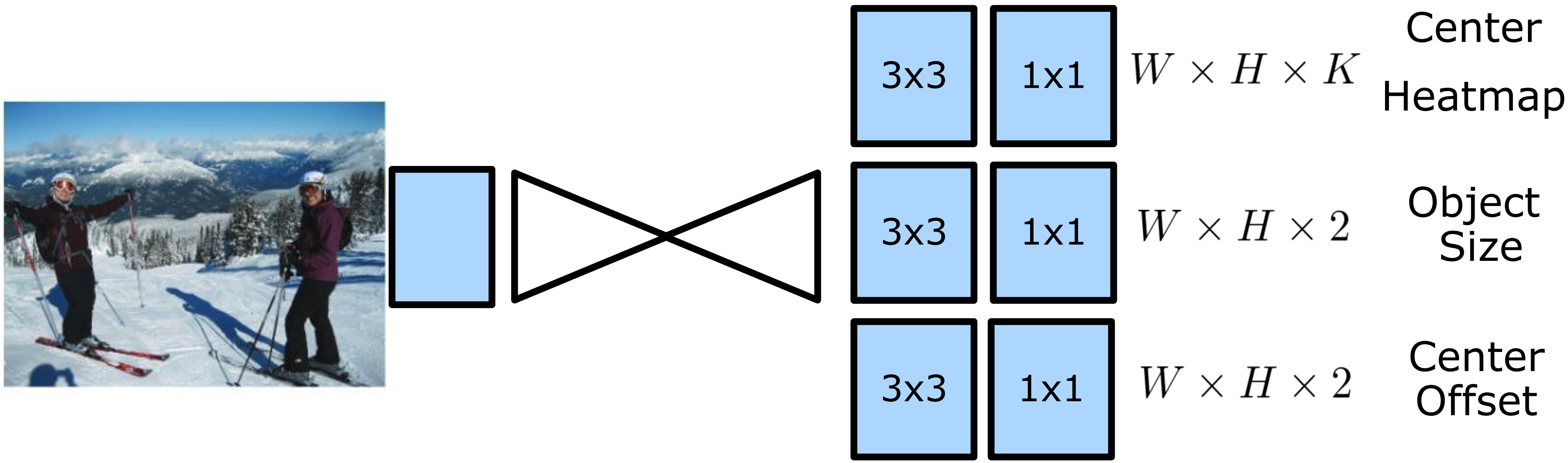


- Different parameterization for bounding box
- **Idea:** Determine only center location and most likely sizes/offsets of bounding boxes at center locations

[Zhou, 2019]

Photo from [Law, 2018]

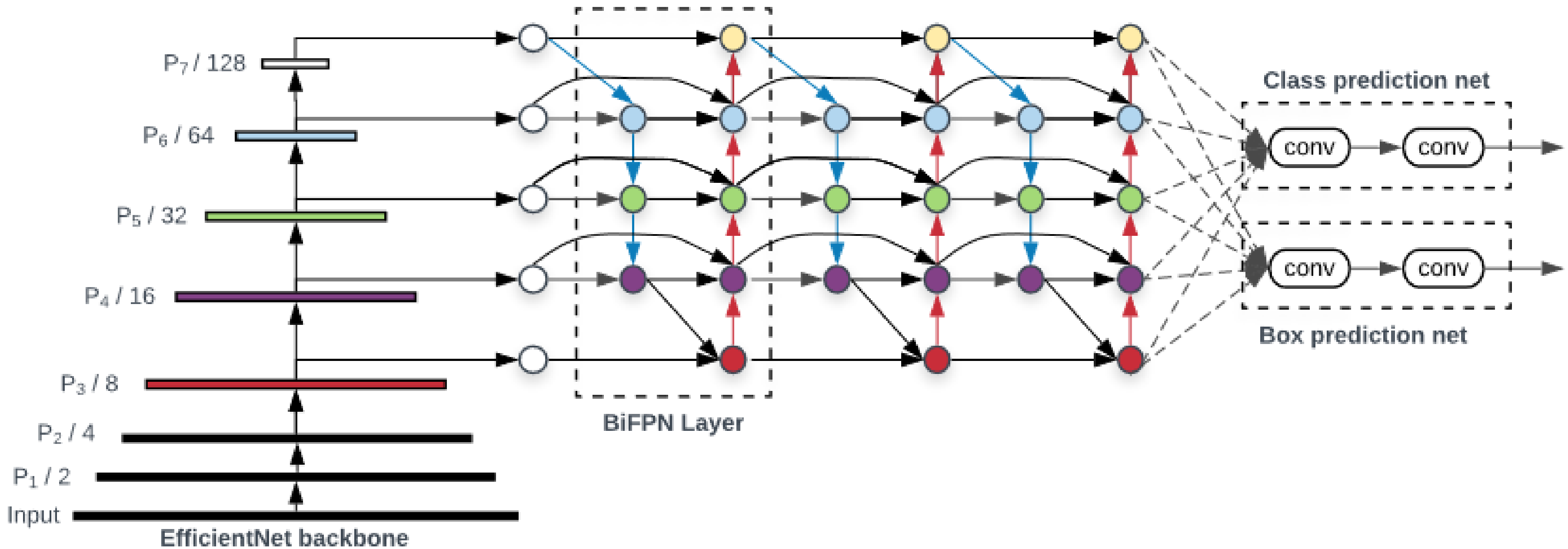
CenterNet (2019)



- CenterNet uses encoder-decoder architecture and produces center heatmaps for all classes
- Bounding box width/height is class-agnostic
- Center offsets account for down-sampled feature maps

[Zhou, 2019]

EfficientDet (2020)

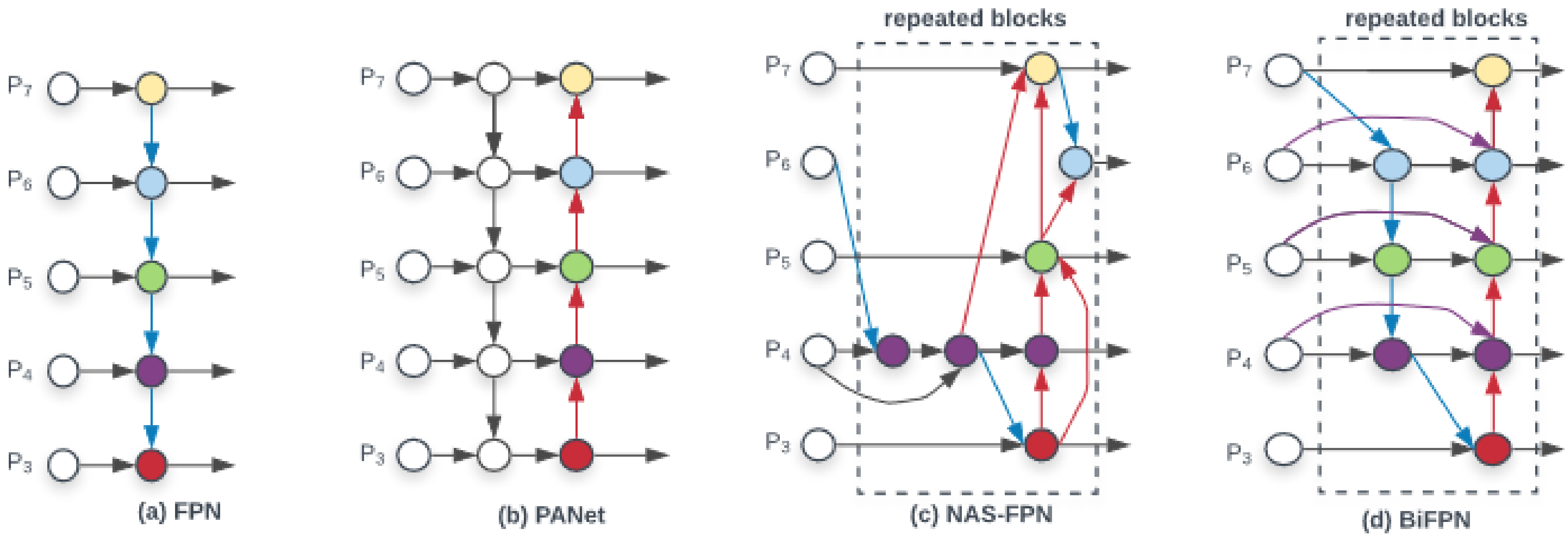


- Compound scaling for detection network, scaling backbone, BiFPN networks, class/box predictors, and resolution
- Again, highly efficient in terms of parameters

[Tan, 2020]

Figure from [Tan, 2020]

BiFPN



- Multi-path aggregation by up-/down-sampling and skip-connection
- Repeated generation of BiFPN blocks to get better features

[Tan, 2020]

Figure from [Tan, 2020]

Recent YOLO variants

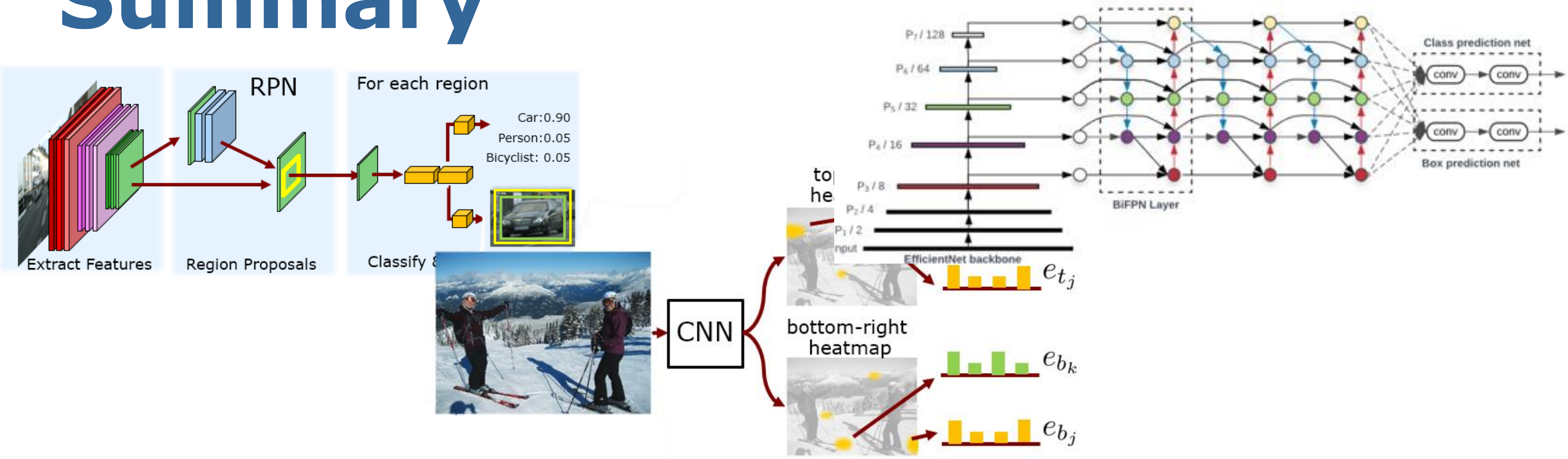
- Different variants improved performance of YOLO considerably:
  1. YOLO9000/YOLOv2: More classes, better backbone (Darknet-19), improved training
  2. YOLOv3: Darknet-53, multi-scale bounding boxes
  3. YOLOv4/YOLOv5 : improved backbone, better training strategies and improved data augmentation, FPN
  4. Scaled-YOLOv4: improved scaling

[Redmon, 2017] [Redmon, 2018] [Redmon, 2018] [Bochkovskiy, 2020]

Performance on MS COCO

Approach	mAP
▪ Fast R-CNN (2015)	19.7
▪ Faster R-CNN (2015)	21.9
▪ YOLOv2 (2016)	21.6
▪ YOLOv3 (2018)	33.0
▪ Feature Pyramid Network (2016)	36.2
▪ RetinaNet (2017)	40.8
▪ CornerNet (2018)	40.5
▪ CenterNet (2019)	42.1
▪ YOLOv4 (2020)	43.0
▪ YOLOv5 (2020)	55.0
▪ EfficientDet-D7x (2020)	55.1
▪ Scaled-YOLOv4 (2020)	55.5

Summary



- Discussed two-stage and single-stage detectors
- Nowadays, single-stage detectors on-par with two-stage detectors
- Anchor-based vs. keypoint-based detectors

See you next week!

References

- Bochkovskiy et al. YOLOv4: Optimal Speed and Accuracy of Object Detection, arxiv, 2020.
- Girshick et al. Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR, 2015.
- Girshick. Fast R-CNN, ICCV, 2015.
- Law et al. CornerNet: Detecting Objects as Paired Keypoint, ECCV, 2018.
- Lin et al. Feature Pyramid Networks for Object Detection, CVPR, 2017.
- Lin et al. Focal Loss for Dense Object Detection, ICCV, 2017.
- Lin et al. Microsoft COCO: Common Objects in Context, ECCV, 2014.
- Redmon et al. You Only Look Once: Unified, Real-Time Object Detection, CVPR, 2016.
- Redmon et al. YOLO9000: Better, Faster, Stronger, CVPR, 2017.
- Redmon et al. YOLOv3: An Incremental Improvement, Arxiv, 2018.
- Ren et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NeurIPS, 2015.
- Tan et al. EfficientDet: Scalable and Efficient Object Detection, CVPR, 2020.
- Wang et al. Scaled-YOLOv4: Scaling Cross Stage Partial Network, arxiv, 2020.
- Zhou et al. Objects as Points, arxiv, 2019.
- YOLOv5: See repository: https://github.com/ultralytics/yolov5