Computer Vision

CS308

Feng Zheng
SUSTech CS Vision Intelligence and Perception

Week 11

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Content

- Brief Review
- Two-stage Object Detection
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- One-stage Object Detection

Brief Review

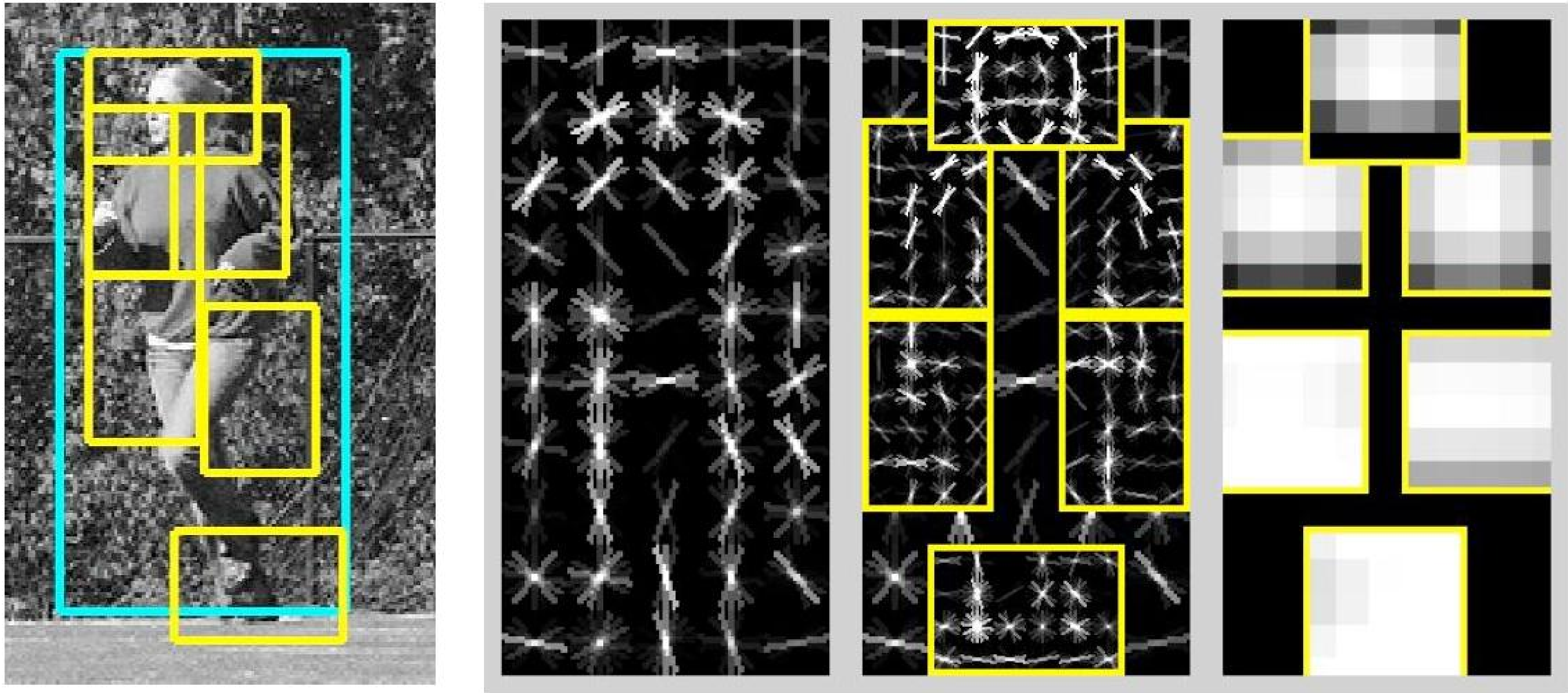
The Viola/Jones Face Detector

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

A Discriminatively Trained, Multiscale, Deformable Part Model



detection　root filter　part filters　deformation models

Pedro Felzenszwalb, David McAllester and Deva Ramanan

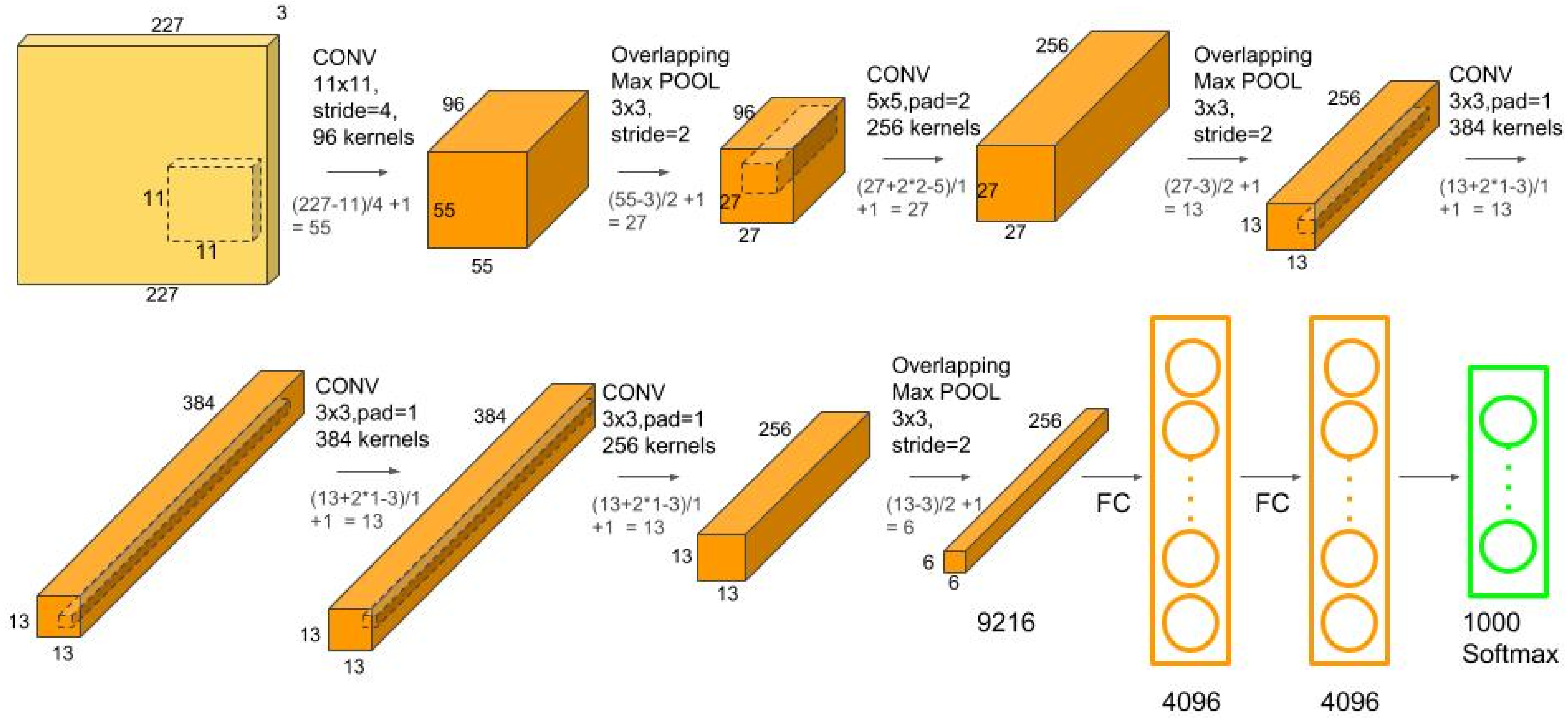
A Discriminatively Trained, Multiscale, Deformable Part Model. IEEE TPAMI, 2010.

Deep Convolutional Neural Networks-AlexNet

- The highlights of the paper
  - Use Relu instead of Tanh to add non-linearity. It accelerates the speed by 6 times at the same accuracy.
  - Use dropout instead of regularization to deal with overfitting. However the training time is doubled with the dropout rate of 0.5.
  - Overlap pooling to reduce the size of network. It reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively.
- Properties
  - It has 60 million parameters and 650,000 neurons and took five to six days to train on two GTX 580 3GB GPUs
  - It contains 5 convolutional layers and 3 fully connected layers.
  - The image size in the following architecture chart should be 227 * 227

https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Deep Convolutional Neural Networks



Two-stage Object Detection

Development of Object Detection

R-CNN 2013.11 → **OverFeat** ICLR' 14 → MultiBox CVPR' 14 → SPP-Net ECCV' 14 → MR-CNN ICCV' 15 → DeepBox ICCV' 15 → AttentionNet ICCV' 15 →

Fast R-CNN ICCV' 15 → DeepProposal ICCV' 15 → **Faster R-CNN** NIPS' 15 → **OHEM** CVPR' 16 → **YOLO v1** CVPR' 16 → G-CNN CVPR' 16 → AZNet CVPR' 16 →

Inside-OutsideNet(ION) CVPR' 16 → HyperNet CVPR' 16 → CRAFT CVPR' 16 → MultiPathNet(MPN) BMVC' 16 → **SSD** ECCV' 16 → GBDNet ECCV' 16 →

CPF ECCV' 16 → MS-CNN ECCV' 16 → **R-FCN** NIPS' 16 → PVANET NIPSW' 16 → DeepID-Net PAMI' 16 → NoC TPAMI' 16 → DSSD arXiv' 17 → TDM CVPR' 17 → **YOLO v2** CVPR' 17 →

Feature Pyramid Net(**FPN**) CVPR' 17 → RON CVPR' 17 → DCN ICCV' 17 → DeNet ICCV' 17 → CoupleNet ICCV' 17 → **RetinaNet** ICCV' 17 → DSOD ICCV' 17 →

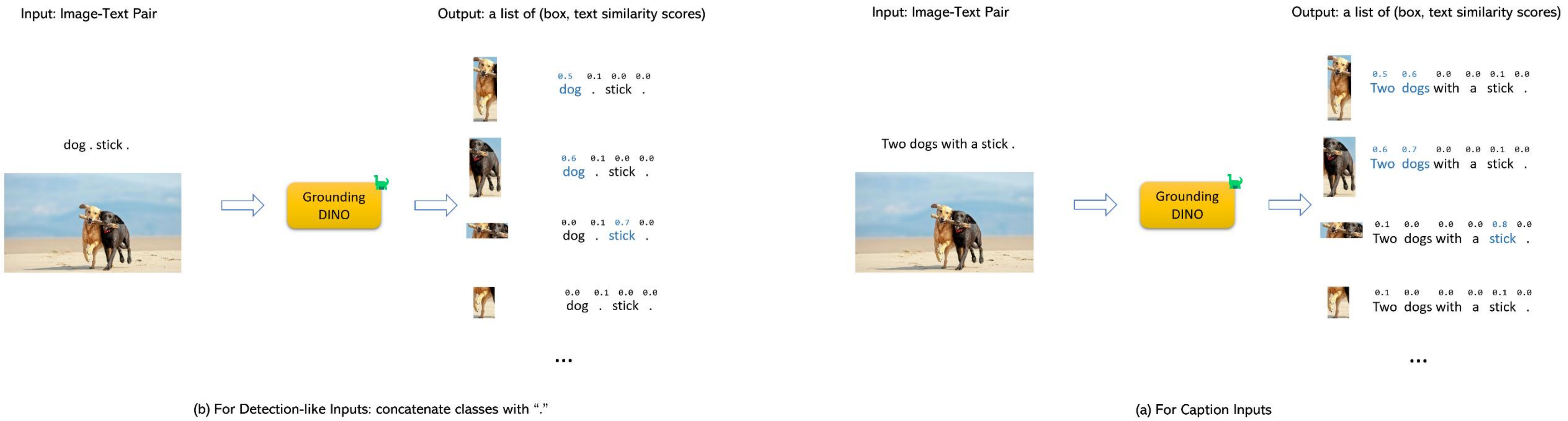
Mask R-CNN ICCV' 17 → SMN ICCV' 17 → **YOLO v3** arXiv' 18 → SIN CVPR' 18 → STDN CVPR' 18 → **RefineDet** CVPR' 18 → MLKP CVPR' 18 → Relation-Net CVPR' 18 →

Cascade R-CNN CVPR' 18 → RFBNet ECCV' 18 → CornerNet ECCV' 18 → PFPNet ECCV' 18 → Pelee NIPS' 18 → HKRM NIPS' 18 → R-DAD AAAI' 19 → **M2Det** AAAI' 19 ···

https://github.com/ultralytics/yolov5

Grounding DINO

- Framework



[GitHub - IDEA-Research/GroundingDINO: The official implementation of "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection"]

R-CNN

- Object detection
  - ➢ The process of finding and classifying objects in an image.
- Deep learning approach: regions with convolutional neural networks (R-CNN)
  - ➢ Combine rectangular region proposals with convolutional neural network features
- R-CNN is a two-stage detection algorithm
  - ➢ The first stage identifies a subset of regions in an image that might contain an object
  - ➢ The second stage classifies the object in each region

R-CNN

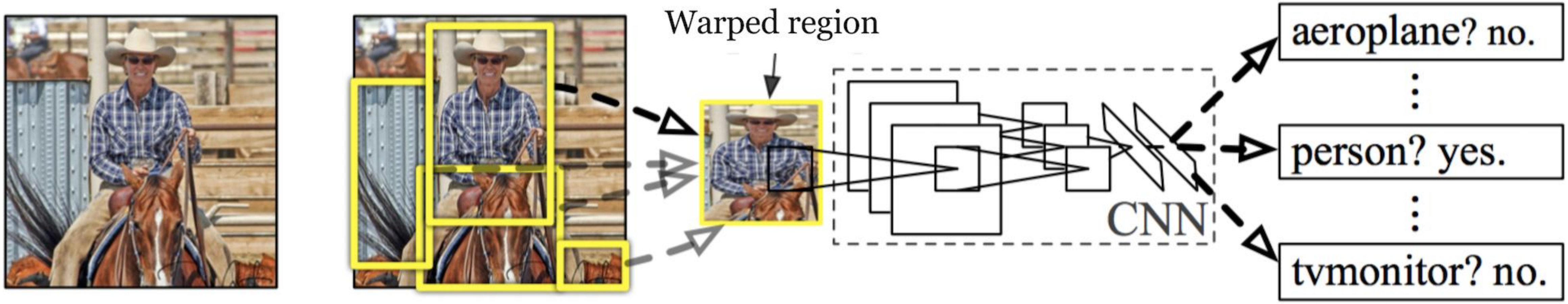
- **Object Detection Using R-CNN Algorithms**
  - Find regions in the image that might contain an object. These regions are called *region proposals*.
  - Extract CNN features from the region proposals.
  - Classify the objects using the extracted features.
- Three variants: each variant attempts to optimize, speed up, or enhance the results of one or more of these processes.
  - R-cnn
  - Fast-rcnn
  - Faster-rcnn

[1] Girshick, R., J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *CVPR '14 Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. Pages 580-587. 2014

[2] Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE International Conference on Computer Vision*. 2015

[3] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *Advances in Neural Information Processing Systems* . Vol. 28, 2015.

R-CNN



1. Input images

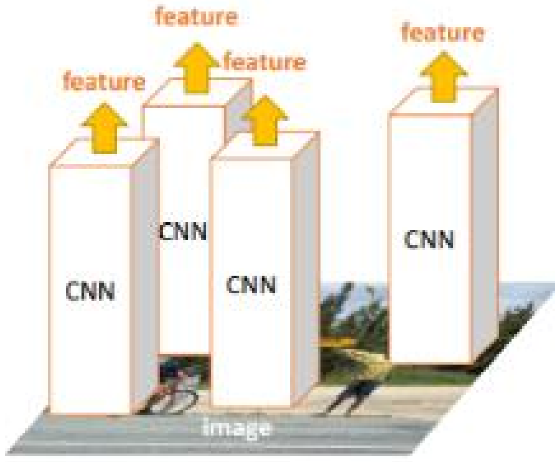
2. Extract region proposals (~2k)

3. Compute CNN features

4. Classify regions

R-CNN

- **Pre-train** a CNN network:
  - ➢ AlexNet, ResNet, VGG, GoogLeNet
- Propose class-independent regions of interest by selective search (~2k candidates per image).
  - ➢ **Warp** to have a fixed size as required by CNN.
- Finetune CNN on warped proposal regions for K + 1 classes
  - ➢ One class refers to the background
  - ➢ Use much smaller learning rate
  - ➢ Oversample the positive cases



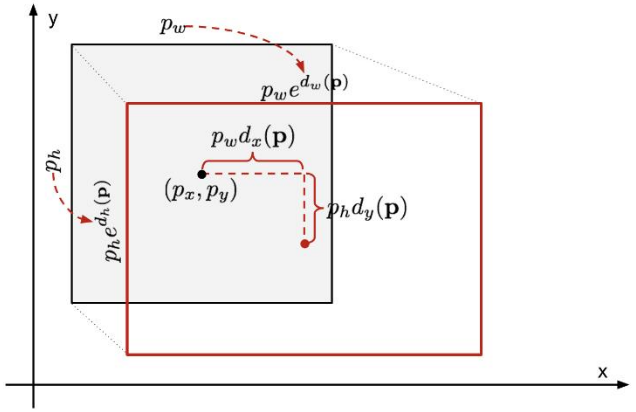
R-CNN

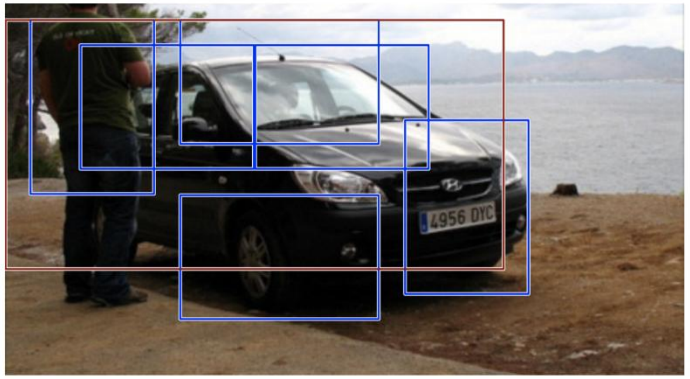
- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features

[1] Zitnick, C. Lawrence, and P. Dollar. "Edge boxes: Locating object proposals from edges." *Computer Vision-ECCV*. Springer International Publishing. Pages 391-4050. 2014.

R-CNN

- Create features from the image proposals
  - One SVM for each object class
  - Fully train the CNN before train the SVM
  - The positive sample: IoU overlap threshold >= 0.3
- A regression model is trained
  - Correct the predicted detection window on bounding box correction offset using CNN features.
- Non-max suppression





Before non-max suppression

After non-max suppression

R-CNN

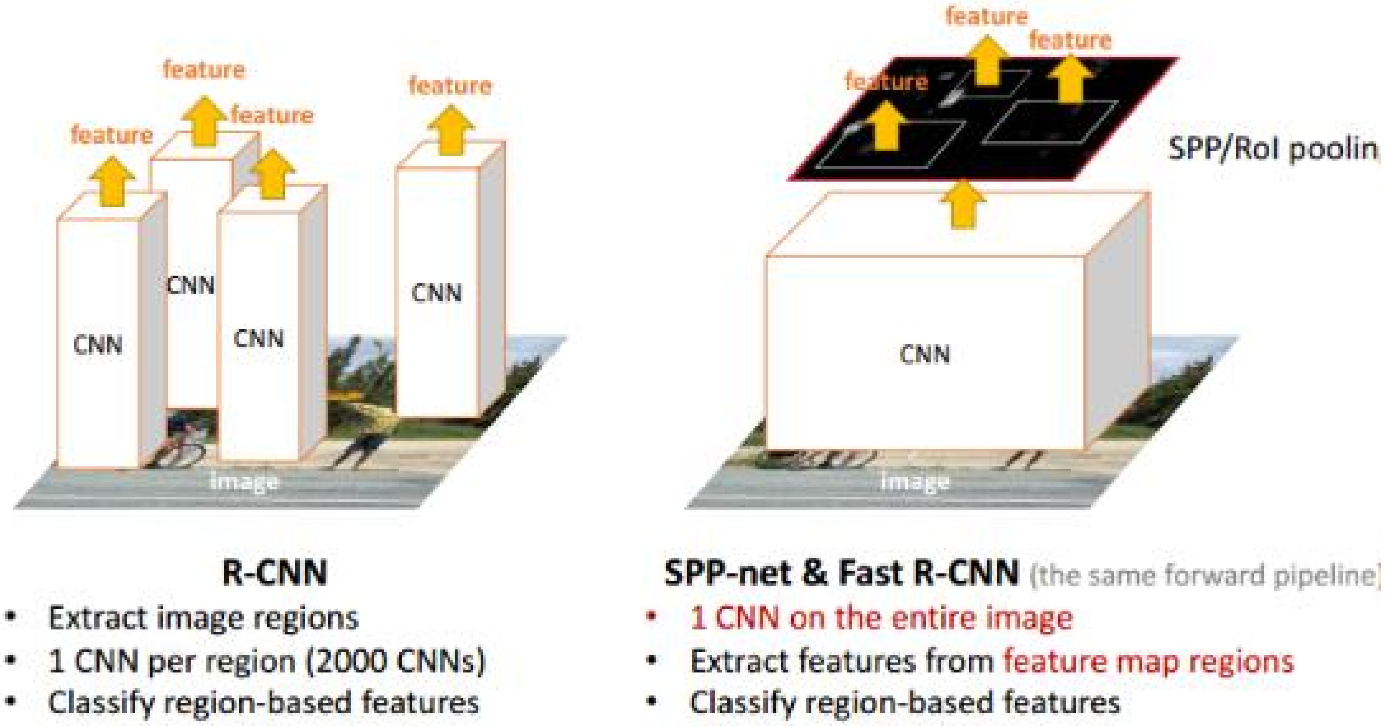
- Problems with R-CNN
  - It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
  - It cannot be implemented real time as it takes around 47 seconds for each test image.
  - The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

Fast R-CNN

- Comparison

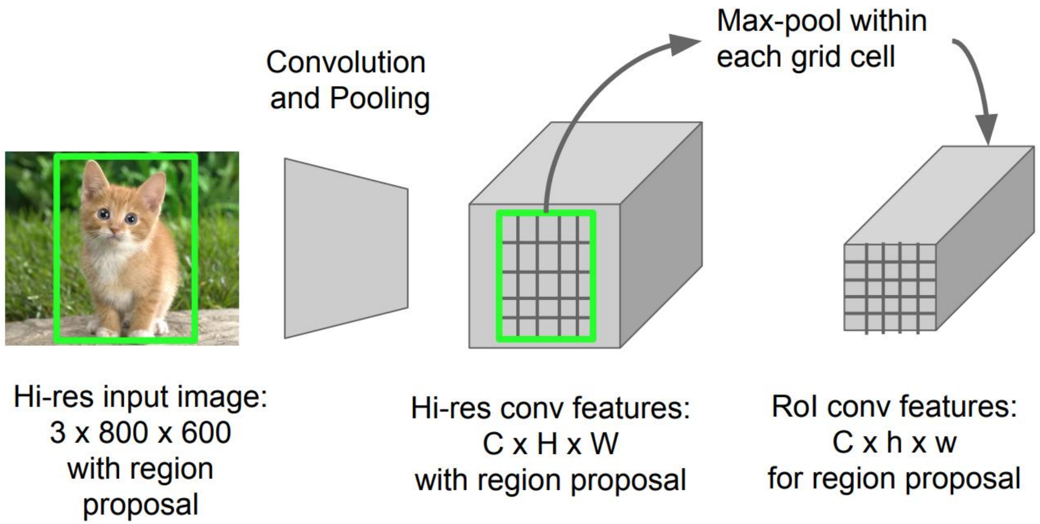
image → crop / warp → conv layers → fc layers → output

image → conv layers → spatial pyramid pooling → fc layers → output



Fast R-CNN

- RoI Pooling to align features
  - ➢ Max pooling: convert features in the projected region of the image of any size, h x w, into a small fixed window, **H x W**
    - ✓ Input region is divided into H x W grids
    - ✓ Every subwindow of size h/H x w/W
    - ✓ Apply max-pooling in each grid

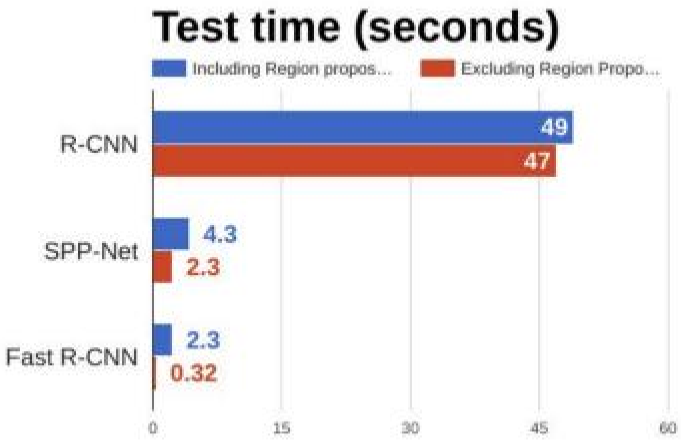


Fast R-CNN

- Pre-train a CNN and propose regions (~2k candidates)
- Alter the pre-trained CNN:
  - Replace the last max pooling layer of the pre-trained CNN with a RoI pooling layer
  - Replace the last fully connected layer and the last softmax layer (K classes) with a fully connected layer and softmax over K + 1 classes
- Two output layers:
  - A softmax estimator of K + 1 classes outputs a discrete probability distribution per RoI
  - A bounding-box regression model predicts offsets relative to the original RoI for each of K classes

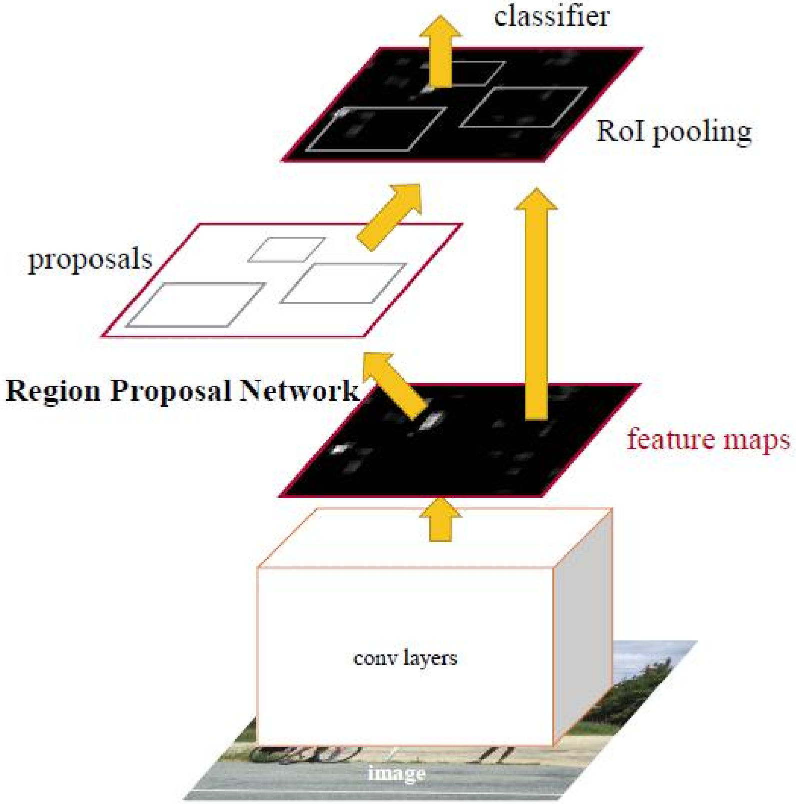
Fast R-CNN

- Fast
  - ➢ Instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map



- Drawback
  - ➢ The region proposals are generated separately by another model and that is very expensive

Faster R-CNN



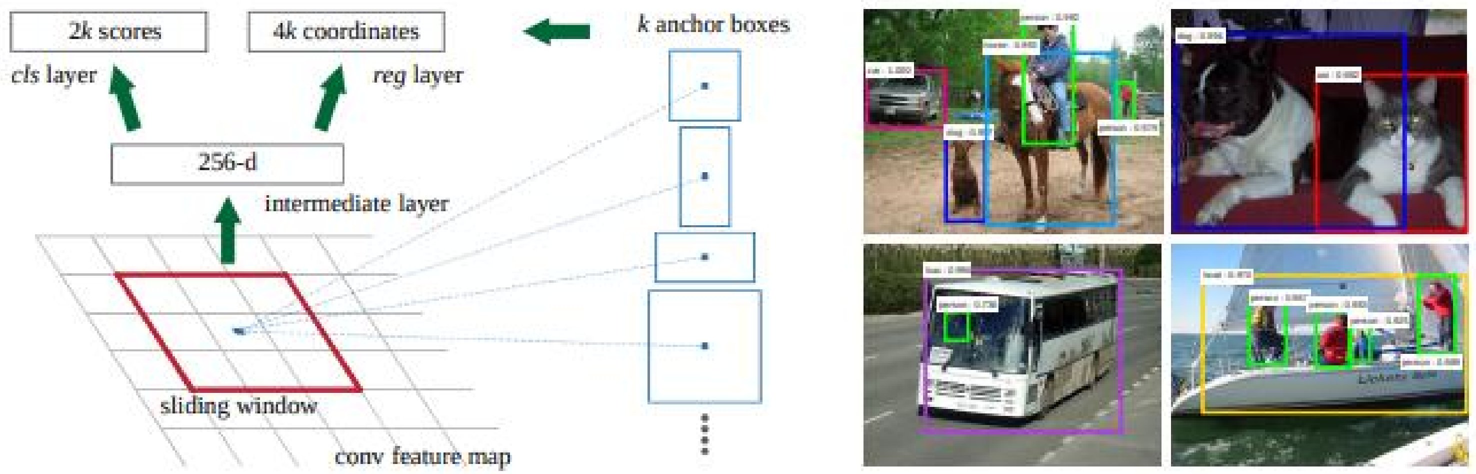
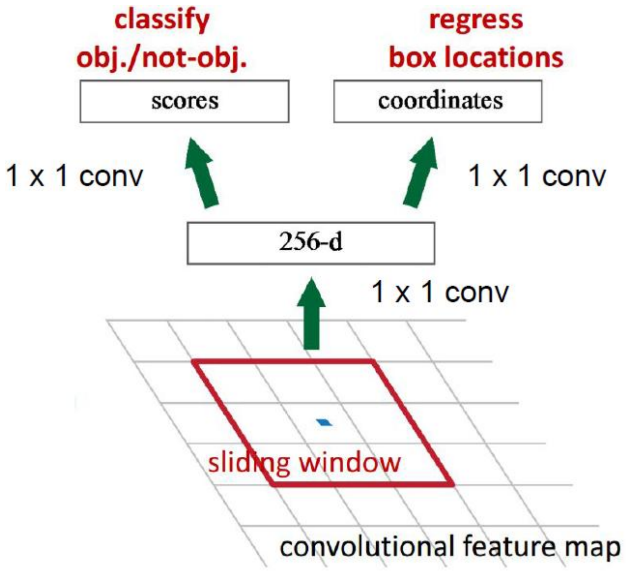


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

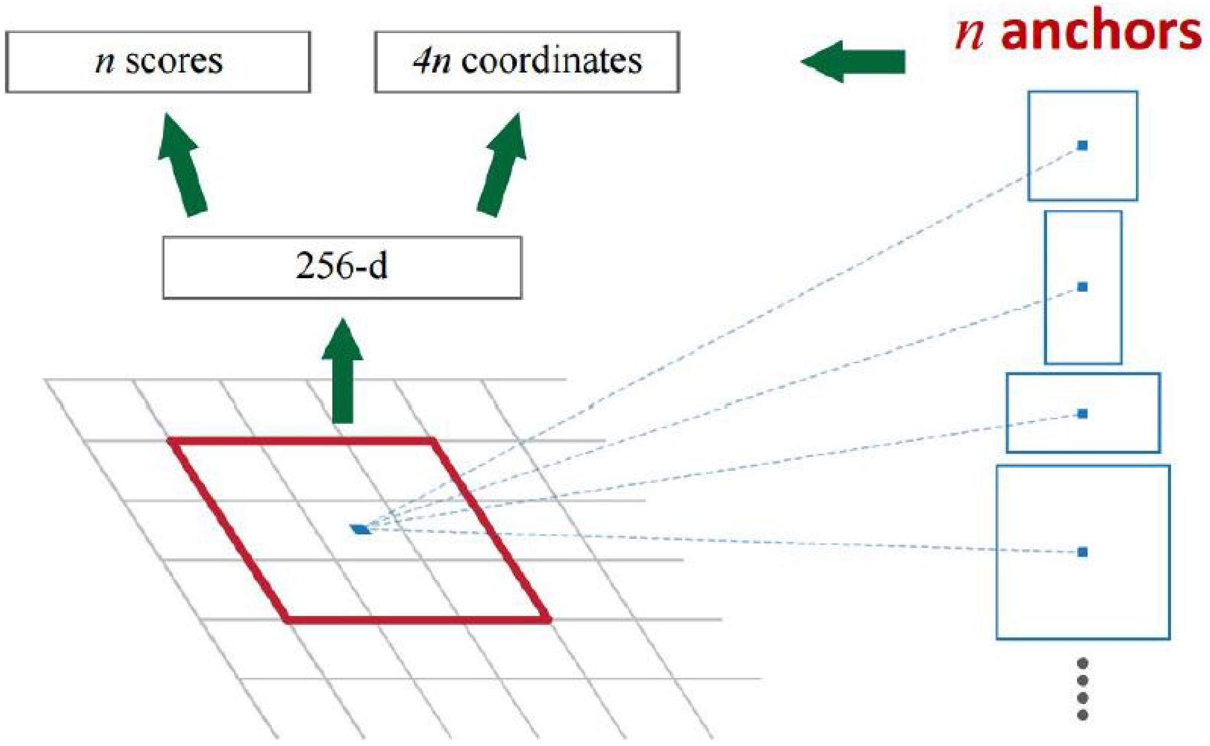
Faster R-CNN

- Region Proposal Network (RPN)
  - RPN trained to produce region proposals directly
  - RoI Pooling, upstream classifier and bbox regressor (Fast R-CNN)
- Build a small network for:
  - Classifying object or not-object
  - Regressing bbox locations
  - Position of the sliding window: provide localization information with reference to the image
  - Box regression: provide finer localization information with reference to this sliding window



Faster R-CNN

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

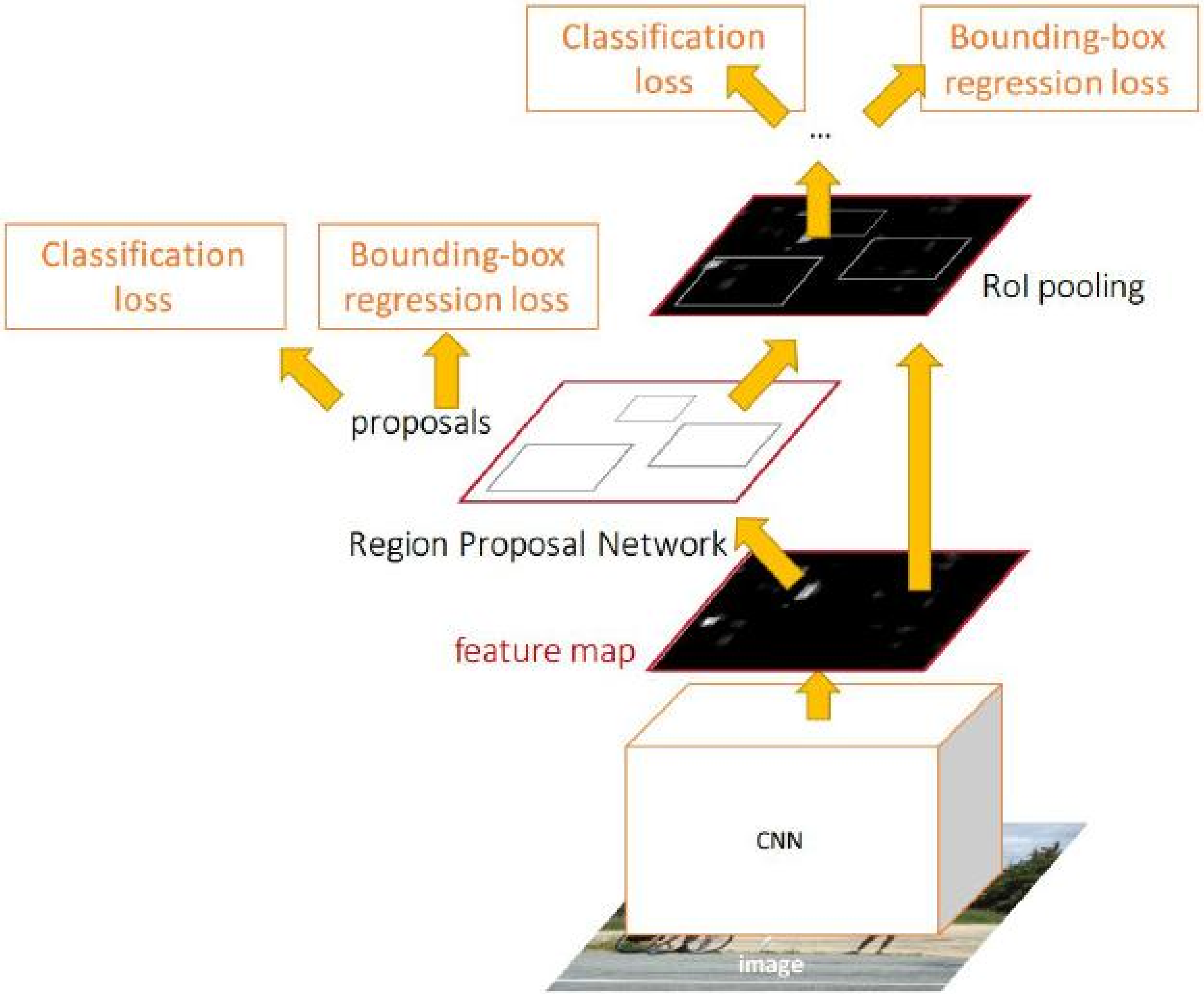


Faster R-CNN

- Pre-train a CNN network
- Fine-tune the RPN
  - Initialization: Positive samples have IoU (intersection-over-union) > 0.7, while negative samples have IoU < 0.3
  - Slide a small n x n spatial window
  - Predict multiple regions (3 scales + 3 ratios => k=9 anchors at each sliding position)
- Train a Fast R-CNN object detection model using the proposals generated by the current RPN
- Use the Fast R-CNN network to initialize RPN training
- Fine-tune the RPN-specific layers
- Fine-tune the unique layers of Fast R-CNN
- Above three steps can be repeated if needed

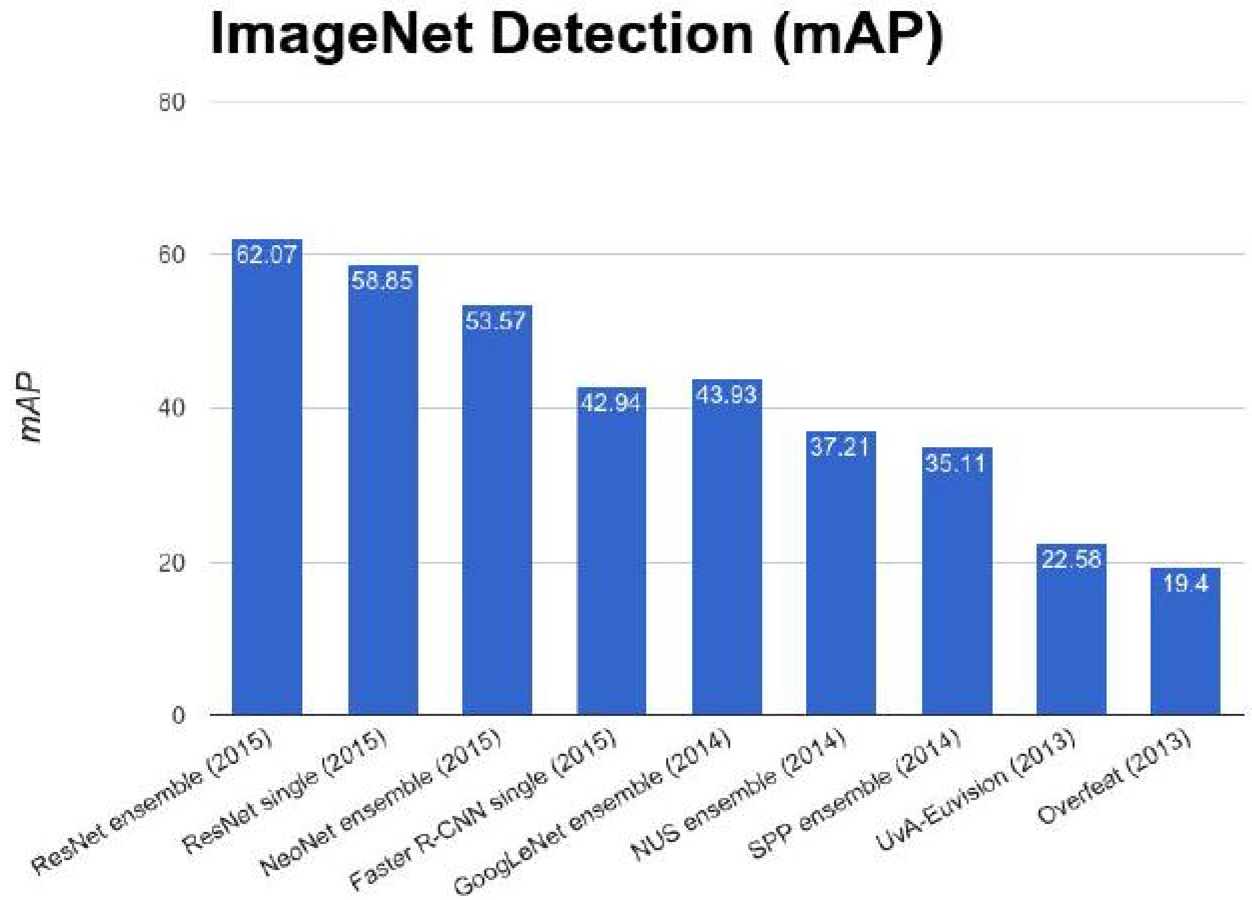
Faster R-CNN

- Four losses
  - RPN classification (anchor good / bad)
  - RPN regression (anchor -> proposal)
  - Fast R-CNN classification (over classes)
  - Fast R-CNN regression (proposal -> box)



Faster R-CNN

- ImageNet Detection 2013 - 2015



You Only Look Once: Unified, Real-Time One-stage Object Detection

Framework of YOLOv1

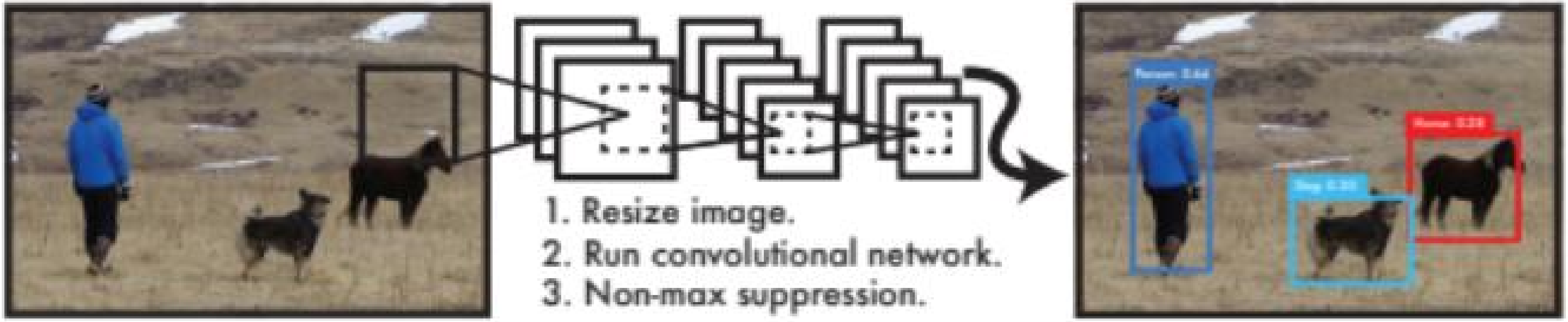


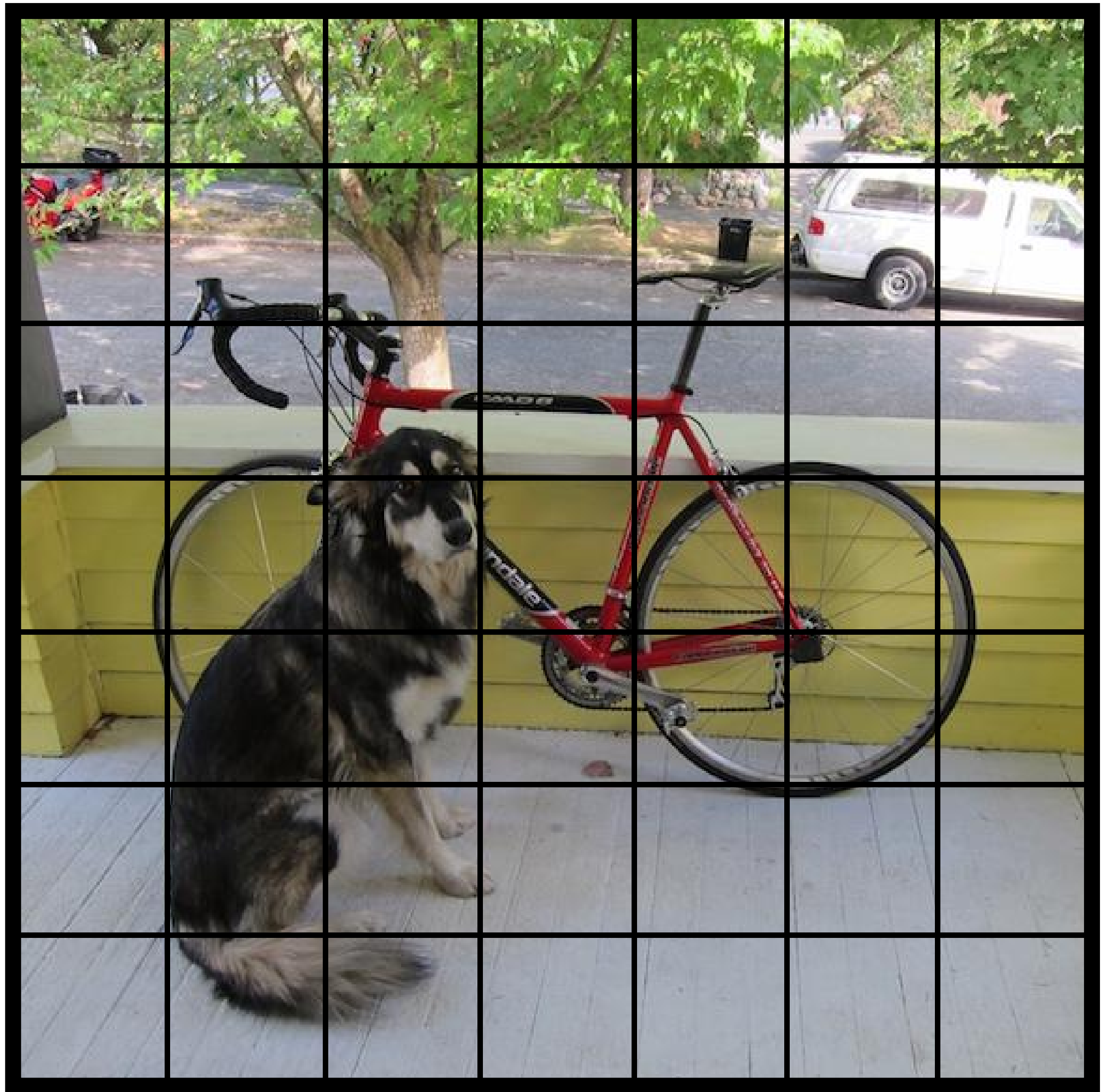
Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Idea

Our system divides
the input image
into a 7*7 grid

Each grid cell predicts
2 bounding boxes and
confidence scores for
those boxes

$$\begin{cases} p_{conf}, x, y, w, h \\ p_{conf}, x, y, w, h \\ p_{c_1}, p_{c_2}, \cdots, p_{c_{20}} \end{cases}$$



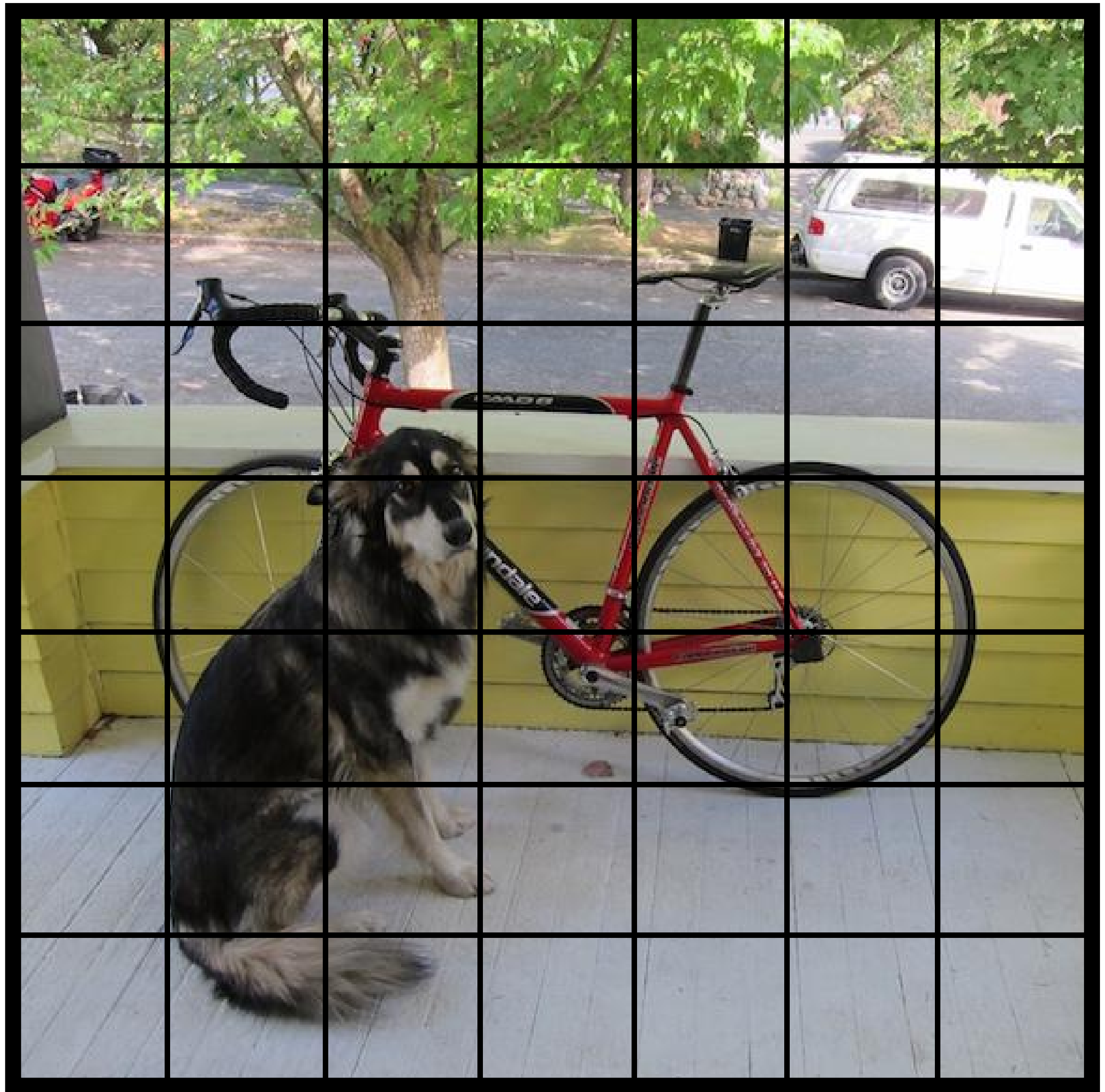
Idea

Our system divides the input image into a 7*7 grid

Each grid cell predicts 2 bounding boxes and confidence scores for those boxes

$$\begin{cases} \boxed{p_{conf}, x, y, w, h} & \text{predictor1} \\ \boxed{p_{conf}, x, y, w, h} & \text{predictor2} \\ p_{c_1}, p_{c_2}, \cdots, p_{c_{20}} \end{cases}$$

Two predictions have the shared class probabilities (20) and totally 30 values (10 for 2 boxes)



Idea

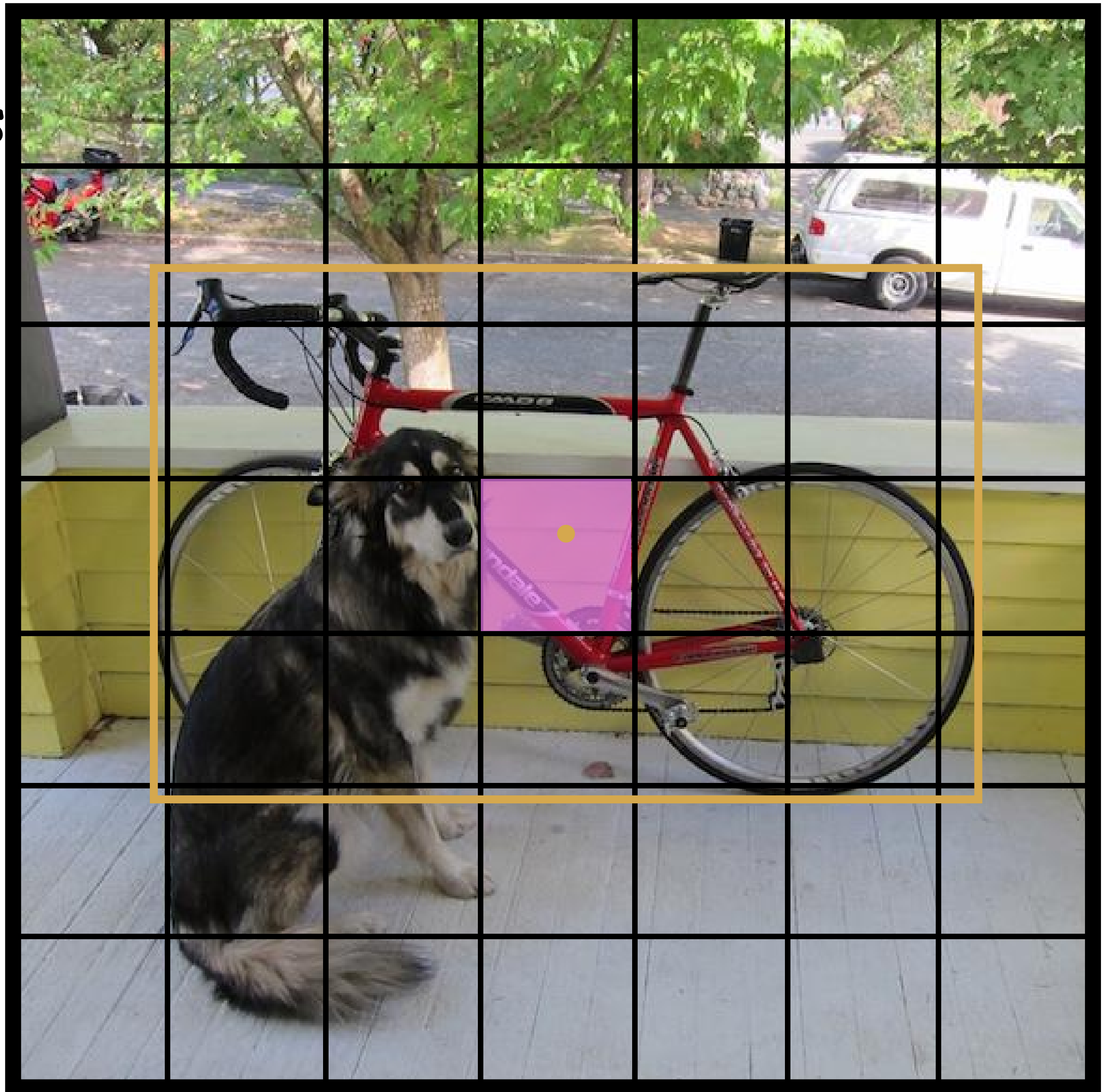
Our system divides the input image into a 7*7 grid

Each grid cell predicts 2 bounding boxes and confidence scores for those boxes

$$\begin{cases} p_{conf}, x, y, w, h \\ p_{conf}, x, y, w, h \\ p_{c_1}, p_{c_2}, \cdots, p_{c_{20}} \end{cases}$$

Ground Truth

1. Object belongs to the cell which the center located in



Idea

Our system divides the input image into a 7*7 grid

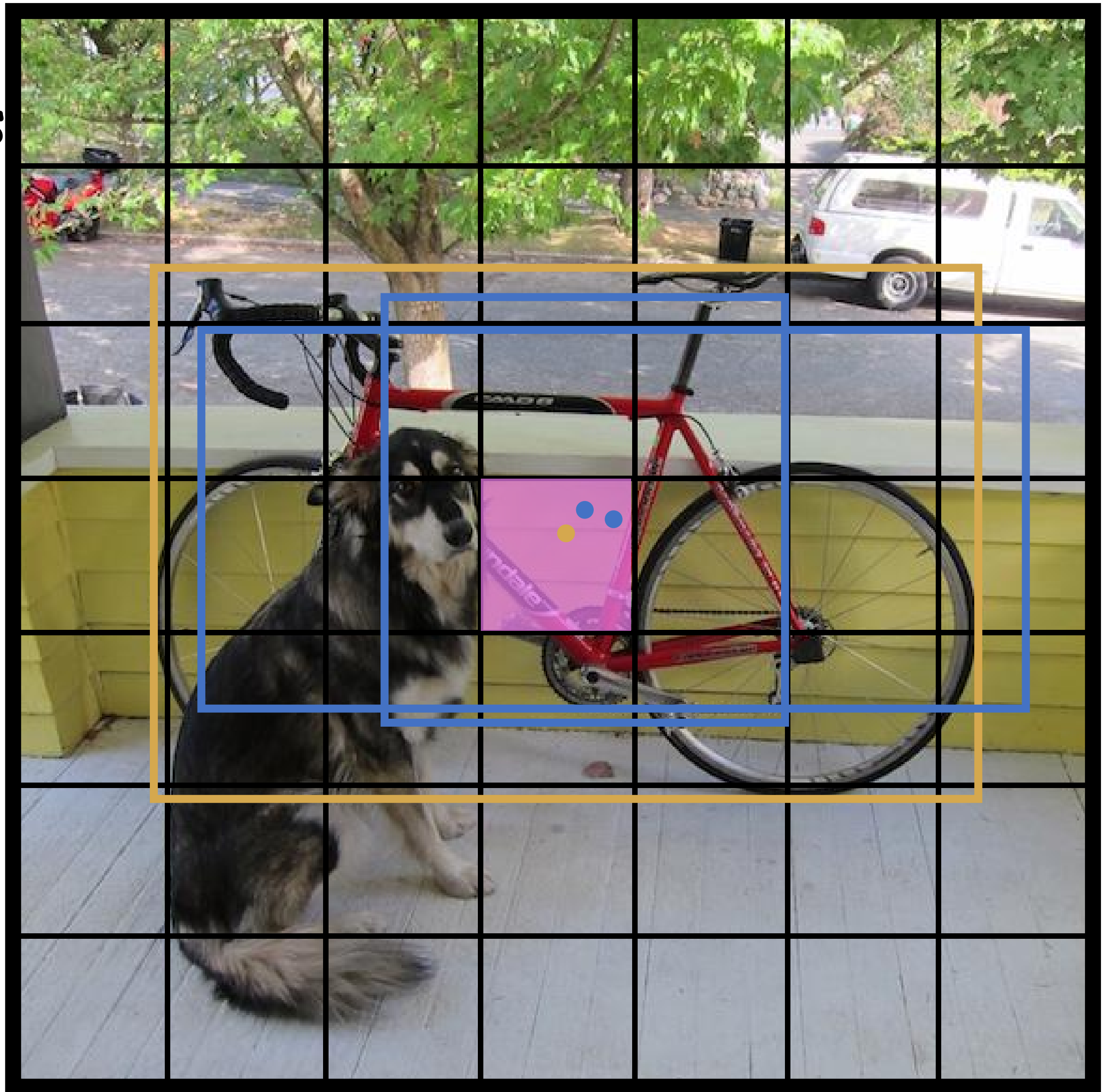
Each grid cell predicts 2 bounding boxes and confidence scores for those boxes

$$\begin{cases} p_{conf}, x, y, w, h \\ p_{conf}, x, y, w, h \\ p_{c_1}, p_{c_2}, \cdots, p_{c_{20}} \end{cases}$$

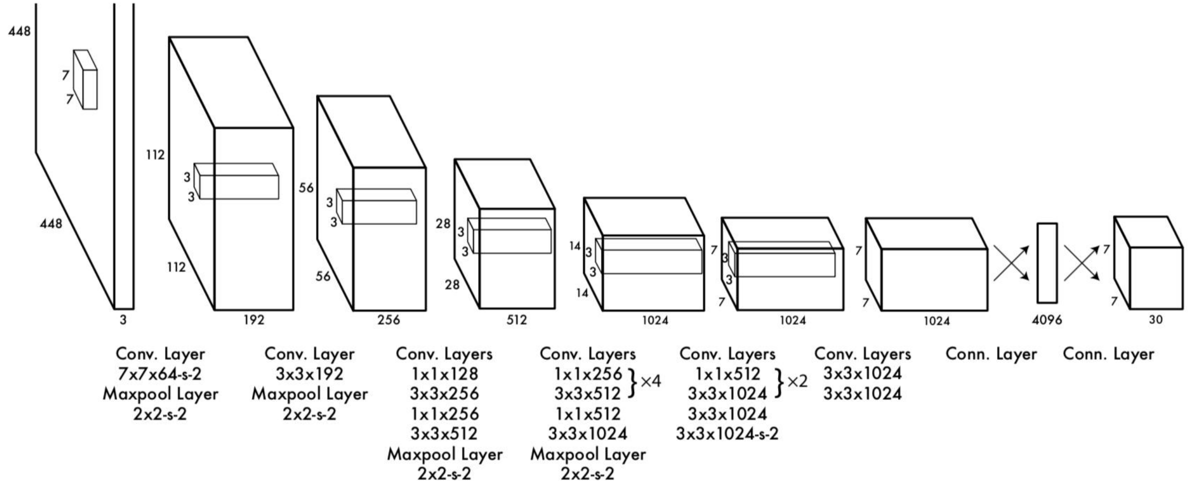
Ground Truth

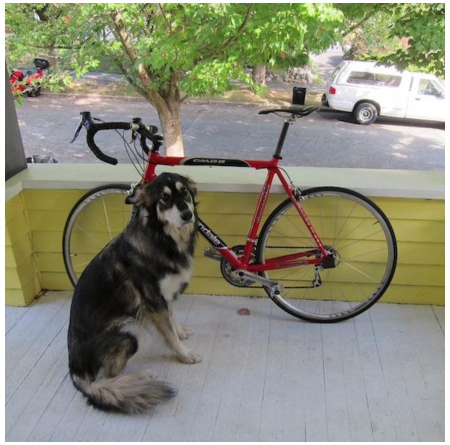
1. Object belongs to the cell which the center located in

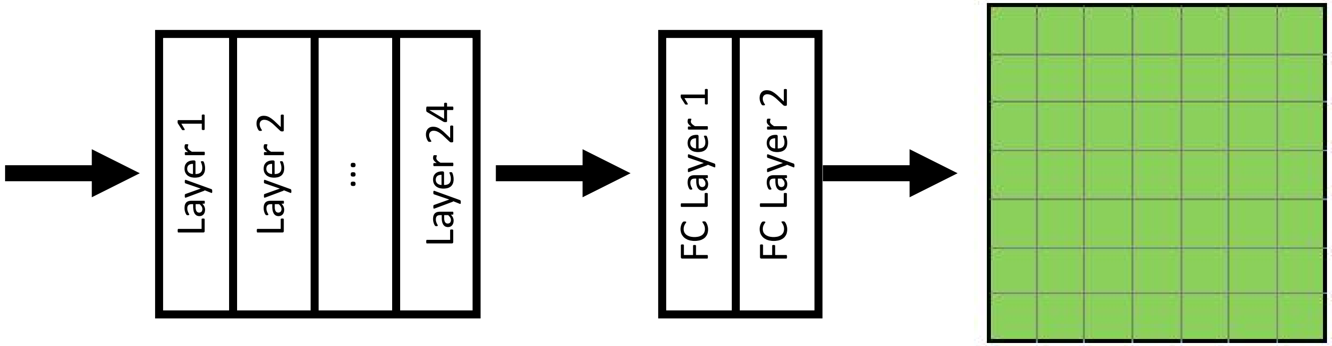
2. Object belongs to the predictor which has the IoU of highest score



Network







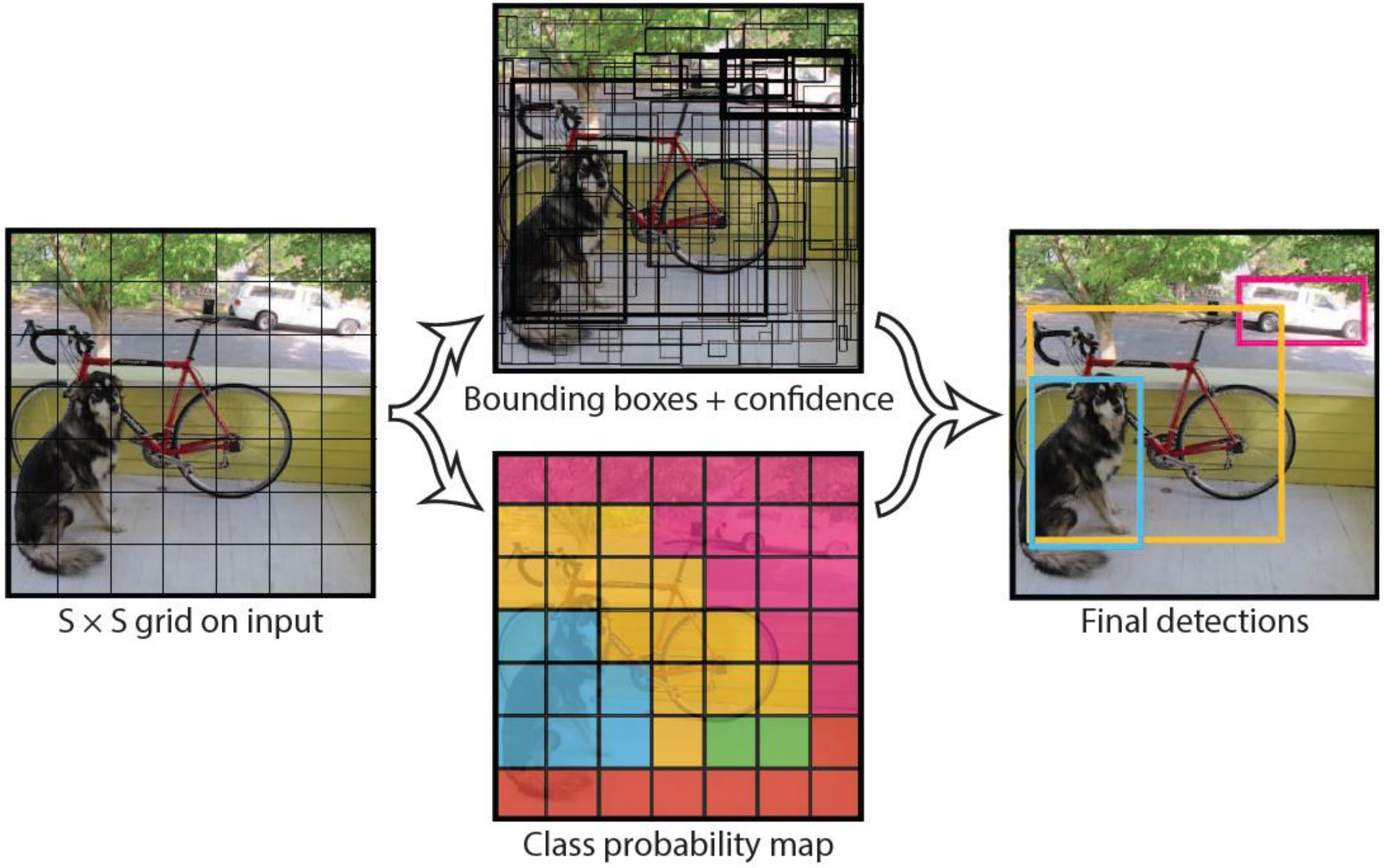
feature extractor network

(trained on ImageNet)

object detection layers

(trained on Pascal VOC)

Framework



$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Overall Loss

Location loss

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Confidence loss

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Class loss

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

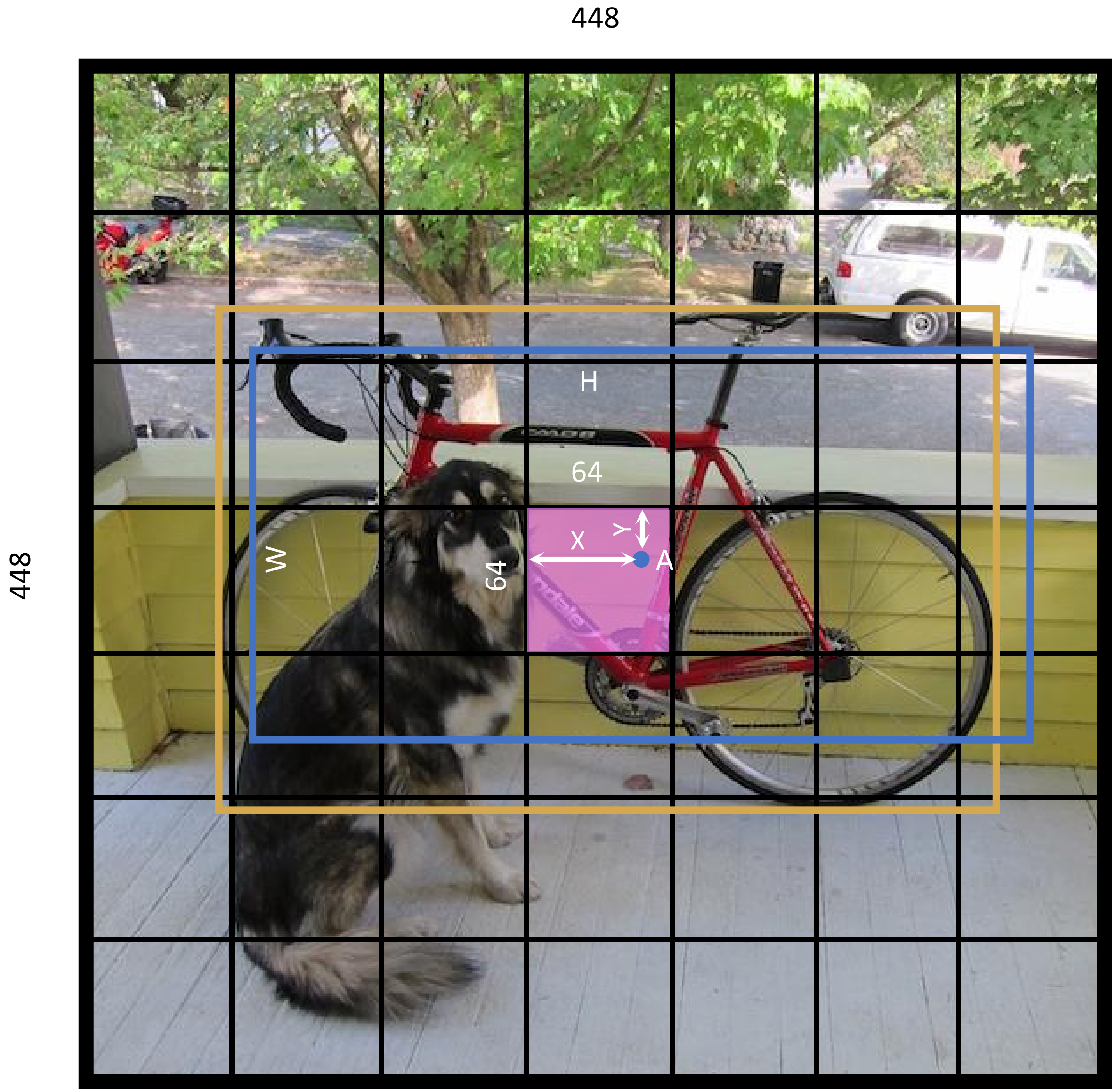
Location Loss

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

It only penalizes bounding box coordinate error if that predictor is “responsible” for the ground truth box

Loss

The meaning of x, y, w, h in predictors (blue)



Loss

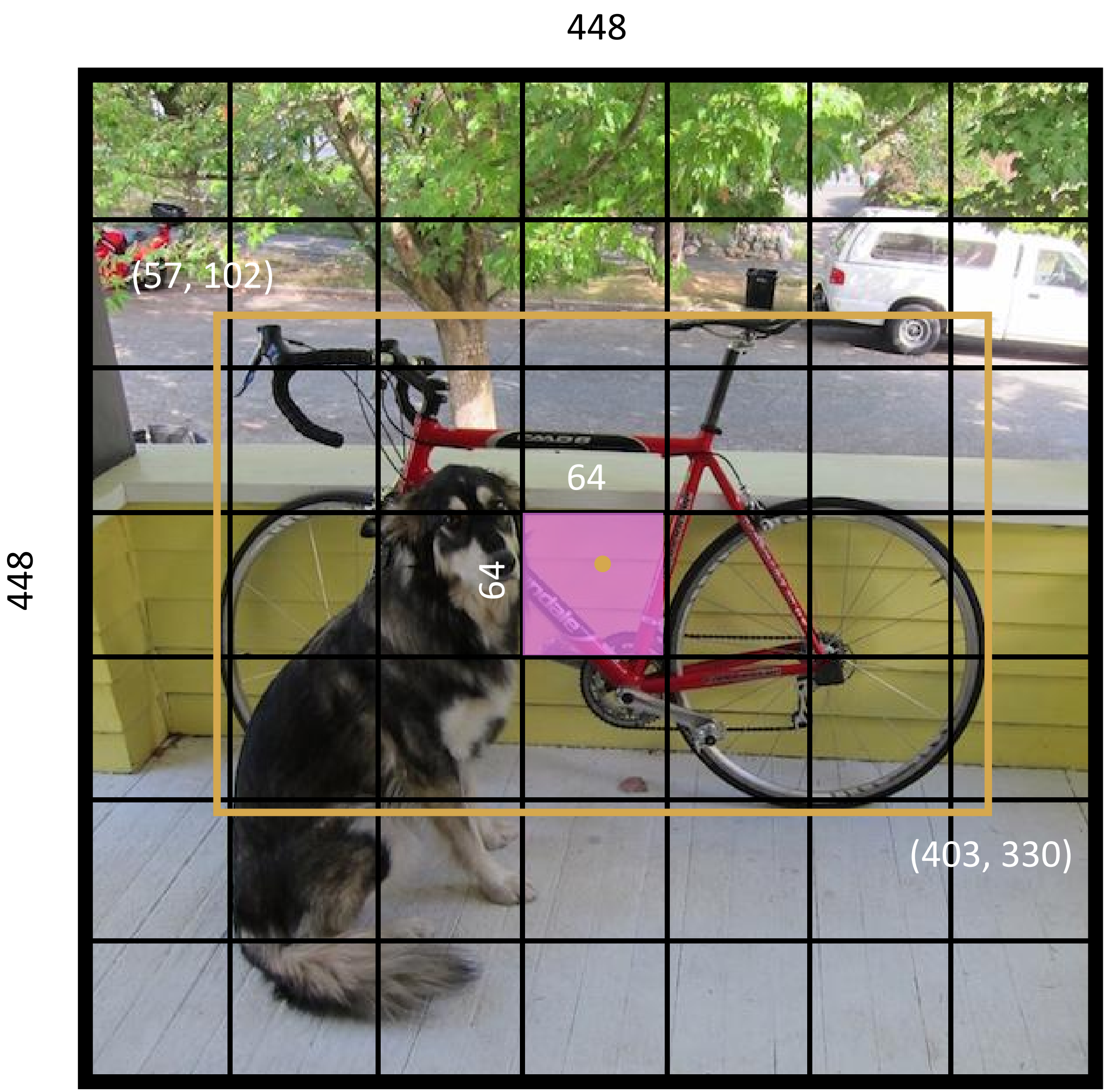
The index of grid is (3, 3).
The gt representation is calculated by

$\hat{x}_i$ = (center_x-64*3)/64

$\hat{y}_i$ = (center_y-64*3)/64

$\hat{w}_i$ = (403-57)/448

$\hat{h}_i$ = (430-102)/448



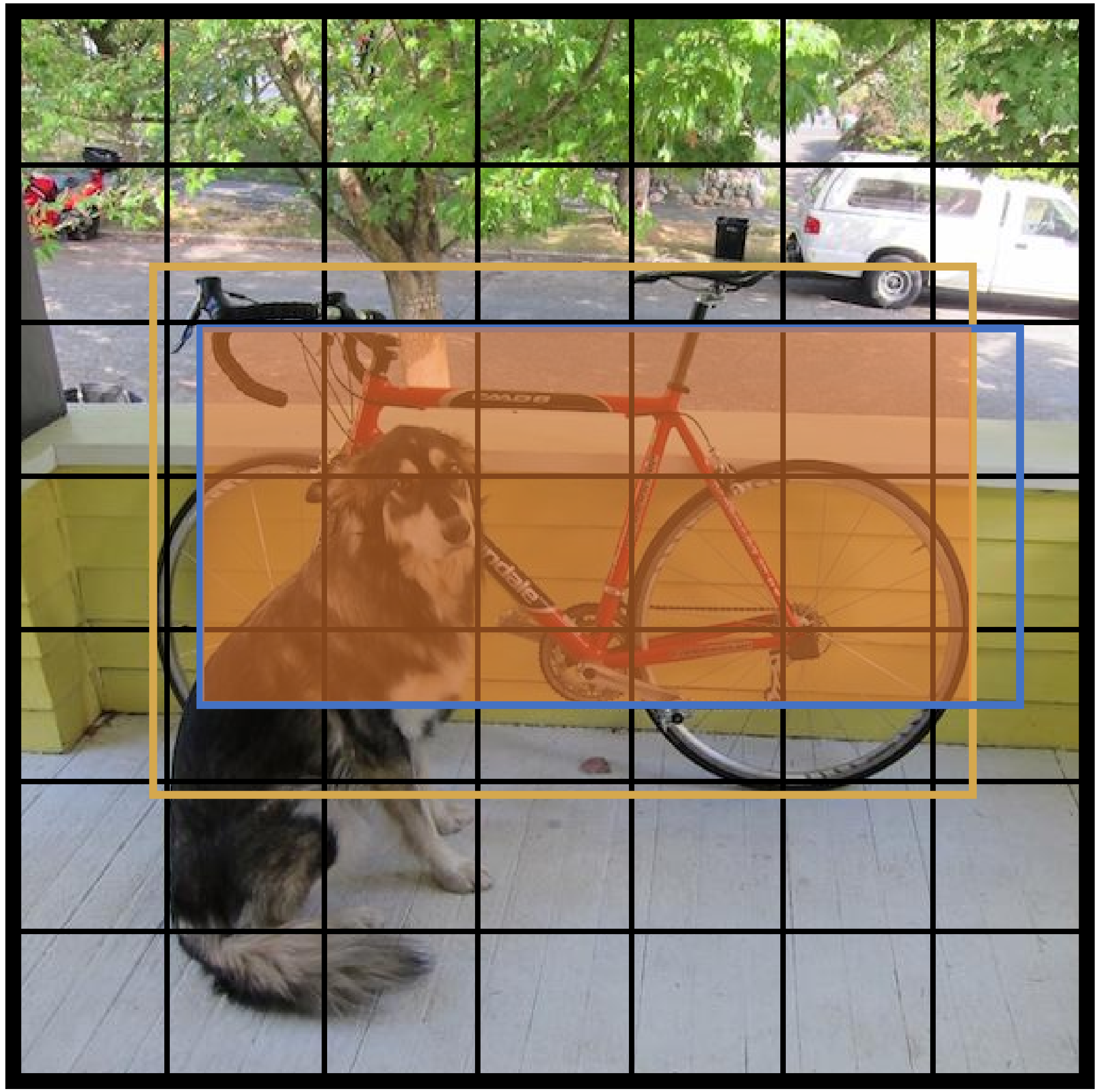
Loss

If has object:

$$\hat{C}_i = \mathrm{IOU}^{\mathrm{truth}}_{\mathrm{pred}}$$

If has no object:

$$\hat{C}_i = 0$$



Loss

$$= \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Confidence loss

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \boxed{\hat{C}_i} \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$= \Pr(\text{Class}_i | \text{Object})$$

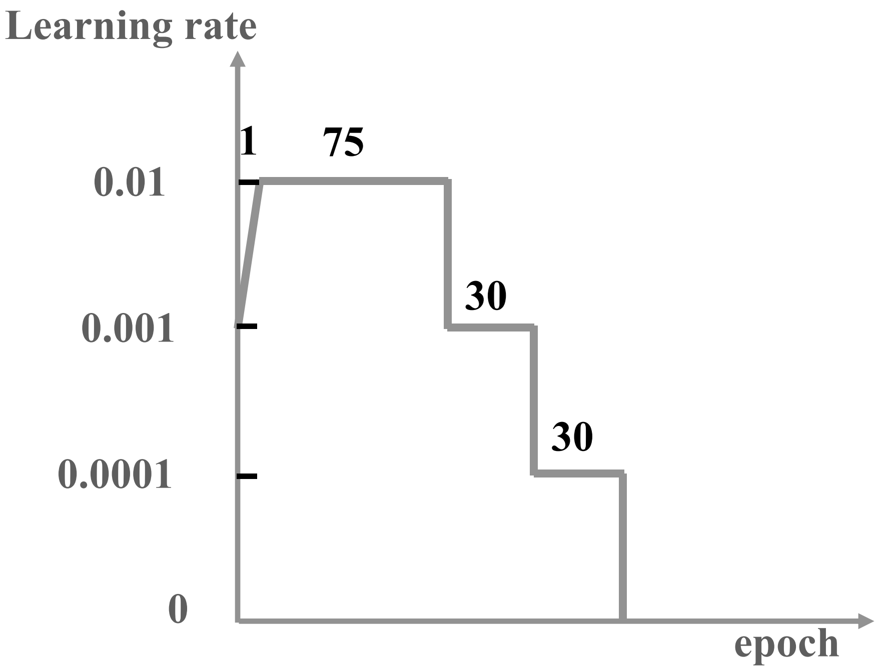
Class loss

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \boxed{\hat{p}_i(c)} \right)^2$$

$$\Pr(\text{Class}_i | \text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Training

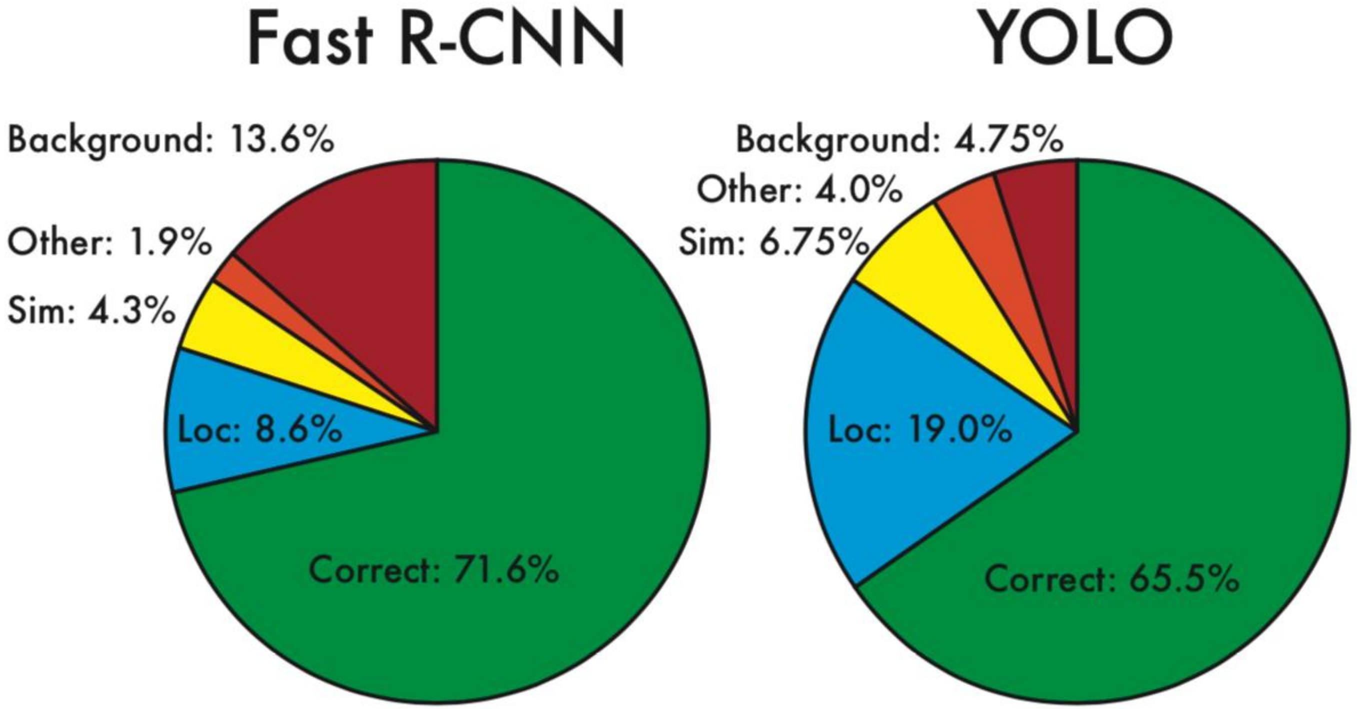
- Data augmentation: scale, translation, random adjust exposure and saturation
  - dropout rate: 0.5
  - momentum: 0.9
  - weight decay: 0.0005
  - batch size: 64
  - learning rate



Experiments

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Error Analysis

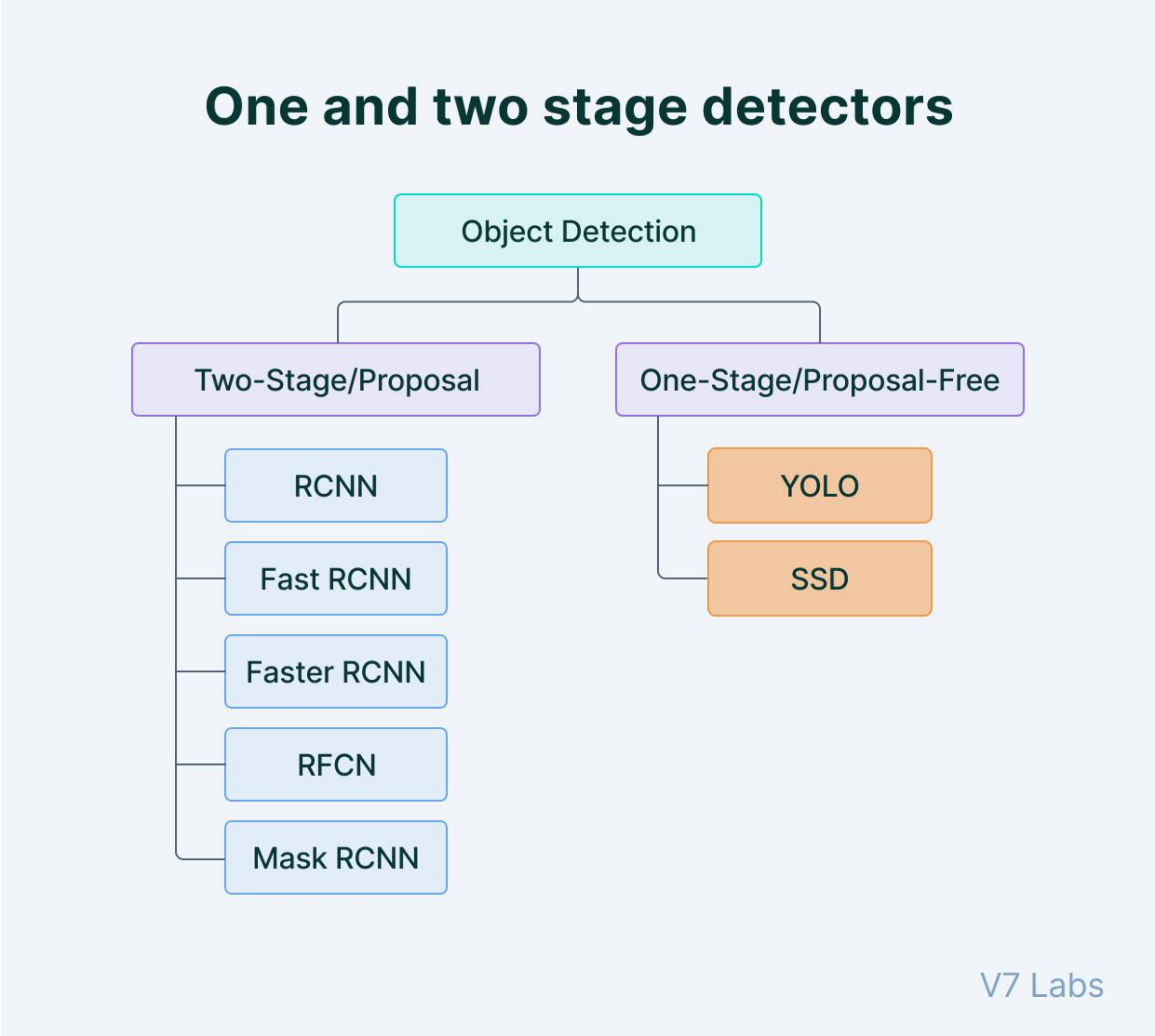


- Correct: correct class and IOU $> .5$
- Localization: correct class, $.1 < \text{IOU} < .5$
- Similar: class is similar, IOU $> .1$
- Other: class is wrong, IOU $> .1$
- Background: IOU $< .1$ for any object

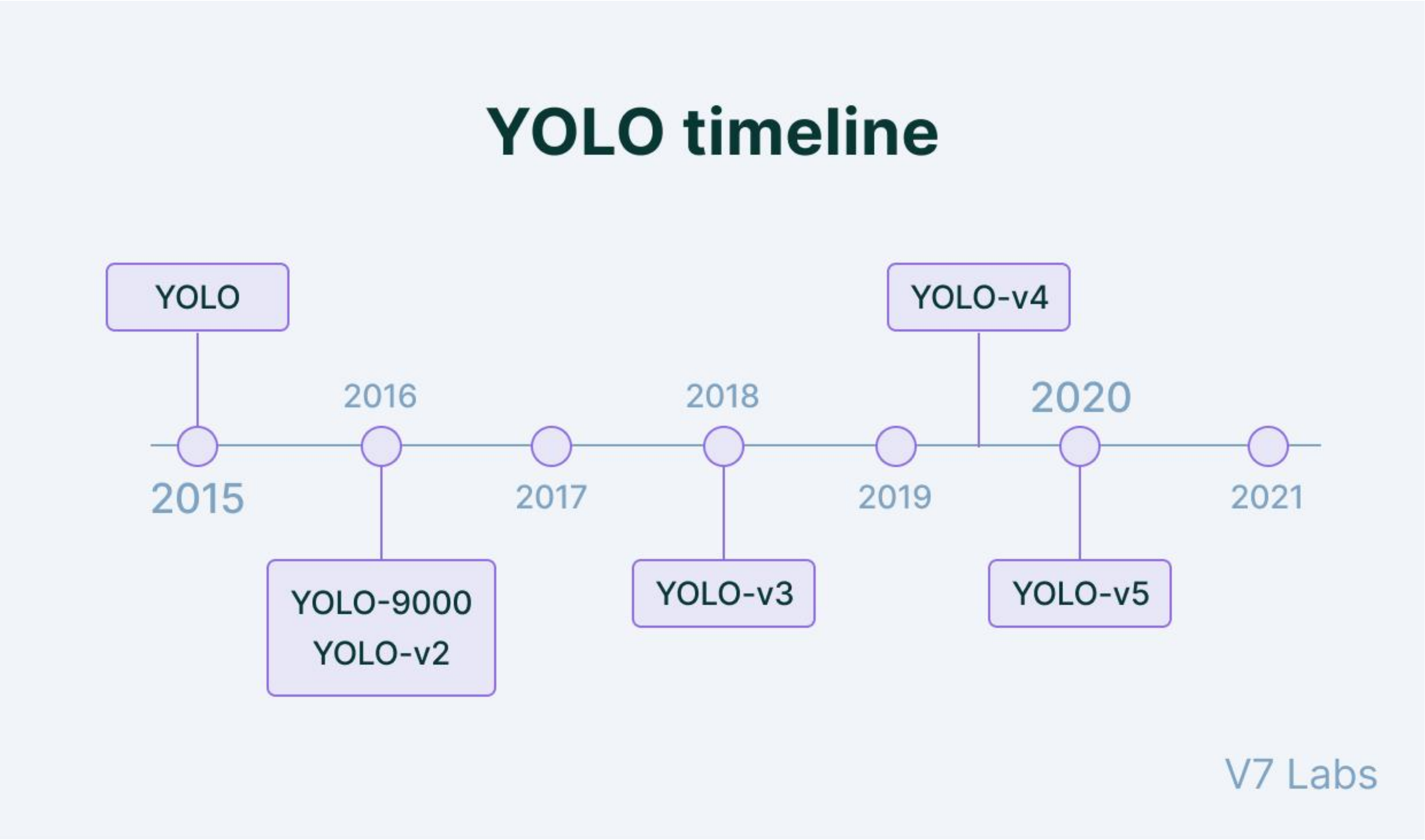
Conclusions

Conclusion-Classical Methods

- Two-stages:
  - ➢ Detecting possible object regions
  - ➢ Classifying the image in those regions into object classes

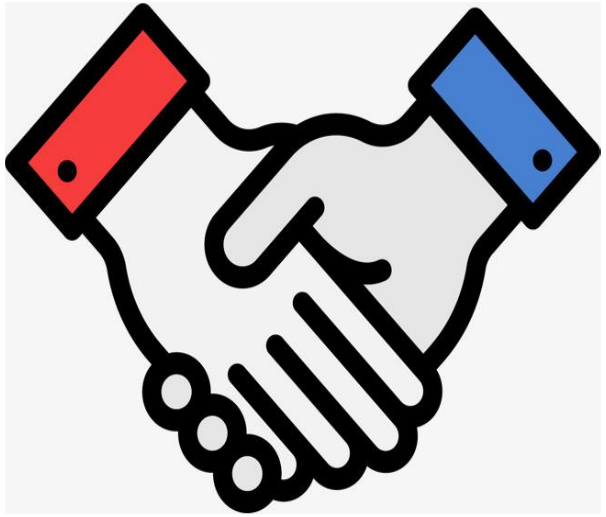


Conclusion-YOLO Development



Conclusion-YOLO Development

- YOLOv1:
  - 24 convolutional layers
  - 2 fully connected layers
- YOLOv2:
  - batch normalization
  - 5 anchor boxes
- YOLOv3
  - DarkNet-53: 106 layers
  - Predict at 82, 94, and 106 layers
  - Predict 3 boxes per cell
- YOLOv4:
  - Weighted Residual Connections
  - Cross Mini Batch Normalization
  - Cross Stage Partial Connections
  - Self Adversarial Training
  - Data augmentation



Thanks

zhengf@sustc.edu.cn