Deep Learning for Computer Vision

113-1/Fall 2024

https://cool.ntu.edu.tw/courses/41702 (NTU COOL)

http://vllab.ee.ntu.edu.tw/dlcv.html (Public website)

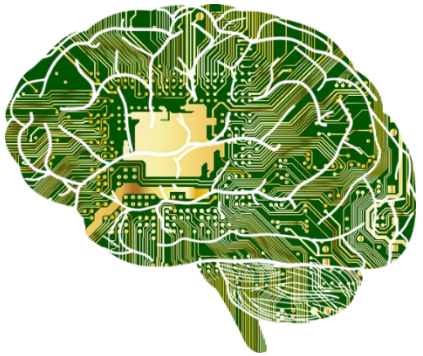
Yu-Chiang Frank Wang 王鈺強, Professor

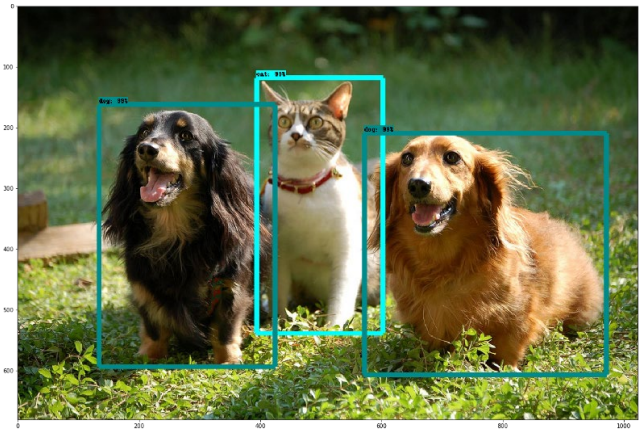
Dept. Electrical Engineering, National Taiwan University

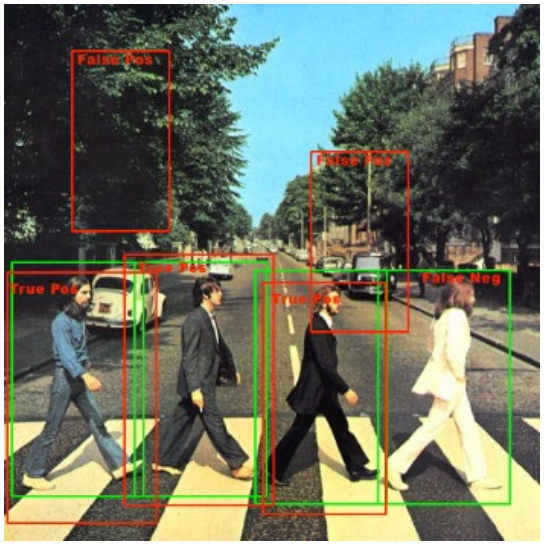
2024/11/26

What to Covered Today…

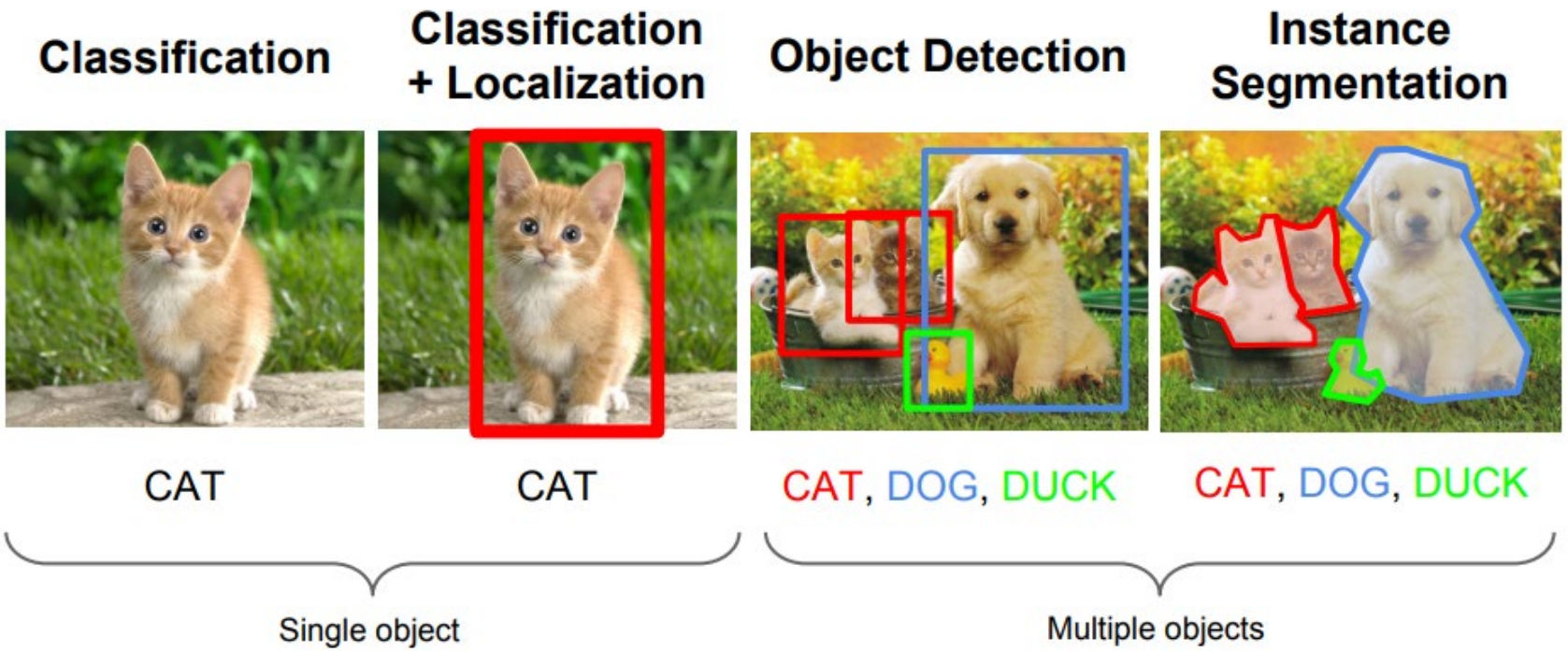
- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection & Grounding







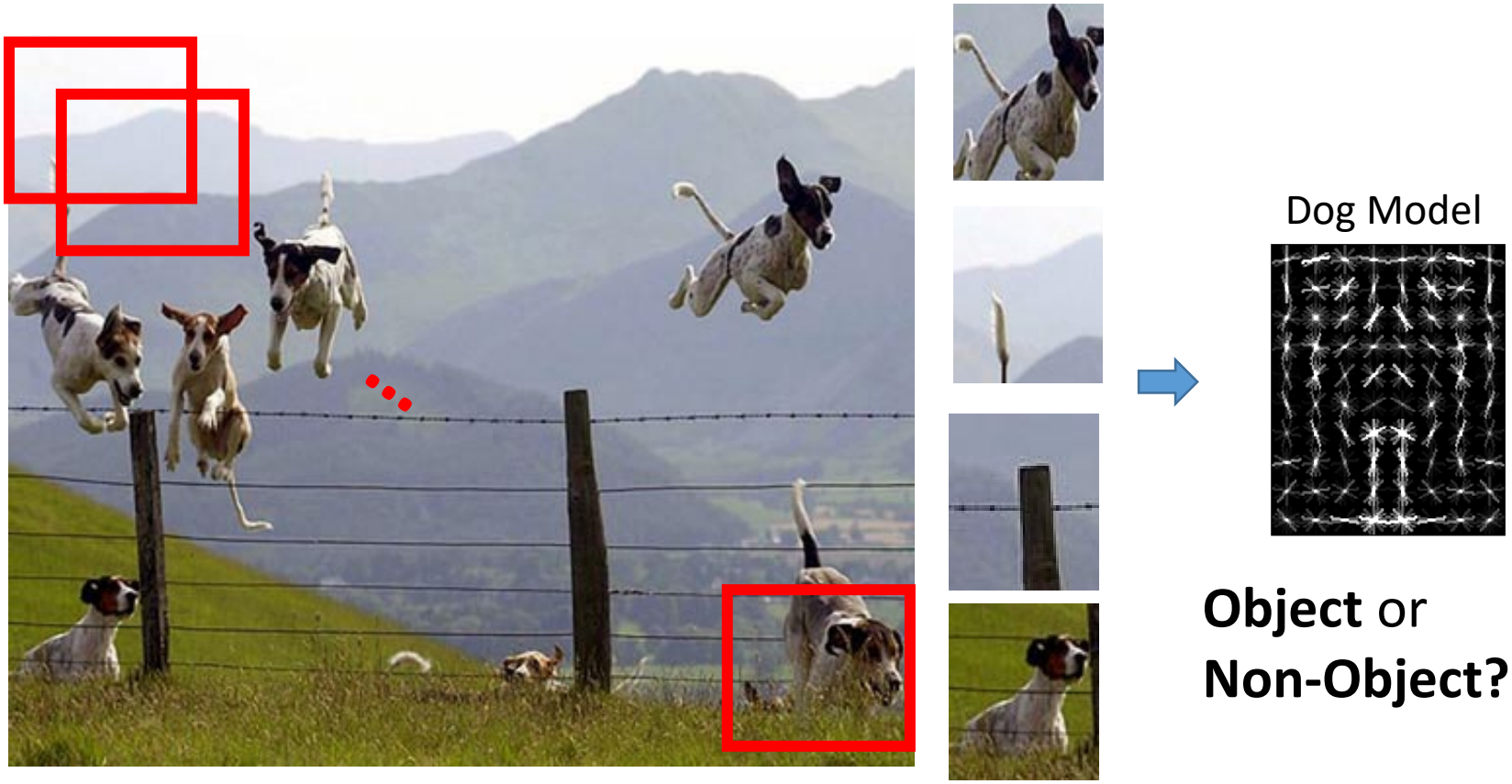
Roadmap



Slide from A. Karpathy

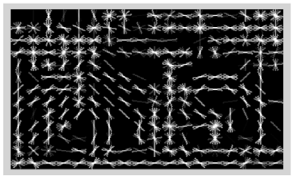
Object Detection

- Focus on object search: “Where is it?”
- Build templates that quickly differentiate object patch from background patch

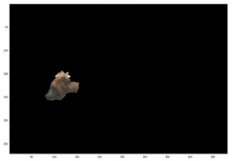


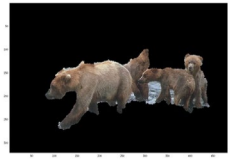
General Process of Object Recognition

Specify Object Model



Generate Hypotheses





Score Hypotheses

Gradient/region based or CNN features, usually based on summary representation with classification/voting results

Resolve Detection

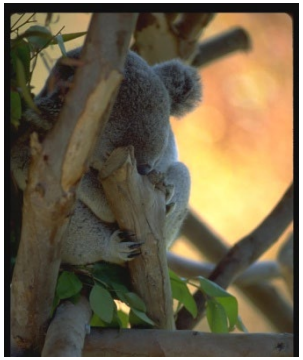
Rescore each proposed object based on the entire candidate set

Challenges in Modeling the Object Classes

Illumination

Object pose

Clutter



Occlusion

Intra-class appearance

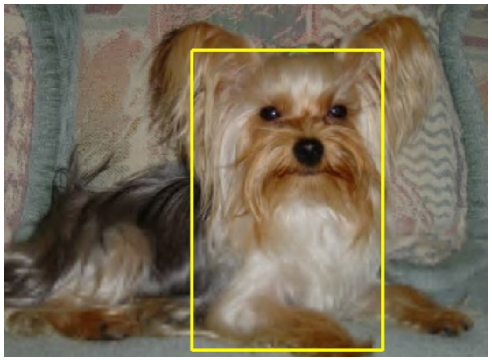
Viewpoint

Slide from K. Grauman, B. Leibe

Challenges in Modeling the Non-object Classes

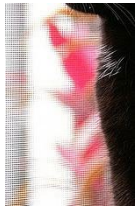
True Detection

Bad Localization

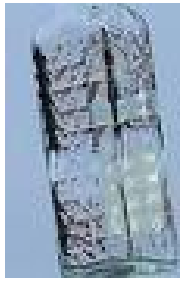


Confused with Similar Object

Misc. Background



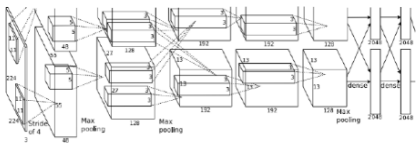
Confused with Dissimilar Objects



Detection by Classification

- CNN-based Methods

This image is CC0 public domain



Treat localization as a regression problem!

Vector: 4096

Fully Connected: 4096 to 1000

Fully Connected: 4096 to 4

"What"

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Weighted Sum

Loss

Box Coordinates (x, y, w, h)

L2 Loss

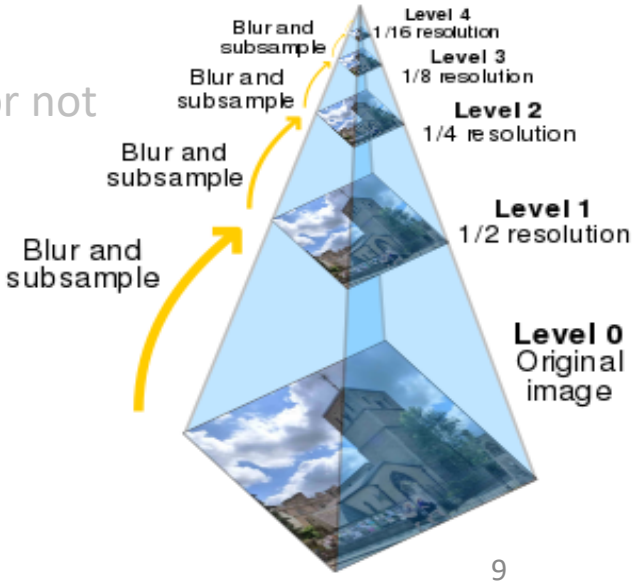
"Where"

Correct box: (x', y', w', h')

Slide credit: UMich EECS 498-007

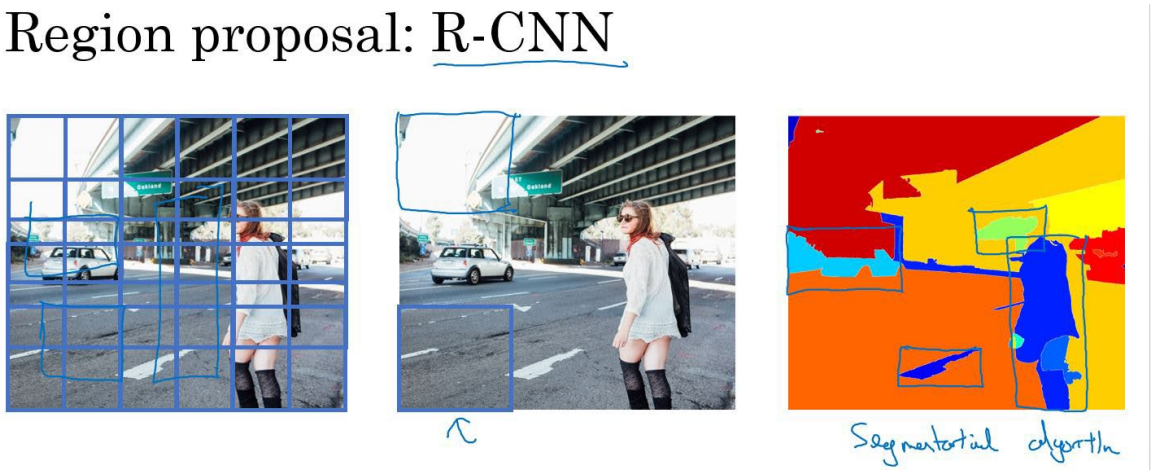
Detection by Classification (cont'd)

- Sliding Windows
  - "Slide" a box within the input image, or even across image scales (see below)
  - Classify each cropped image region inside the box and determine if it's an object of interest or not
  - E.g., HOG (person) detector by Dalal and Triggs (2005)
    Deformable part-based model by Felzenswalb et al. (2010)
    Real-time (face) detector by Viola and Jones (2001)
- Region (Object) Proposals
  - Generate region (object) proposals
  - Classify each proposal and determine it's an object or not



Detection by Classification (cont'd)

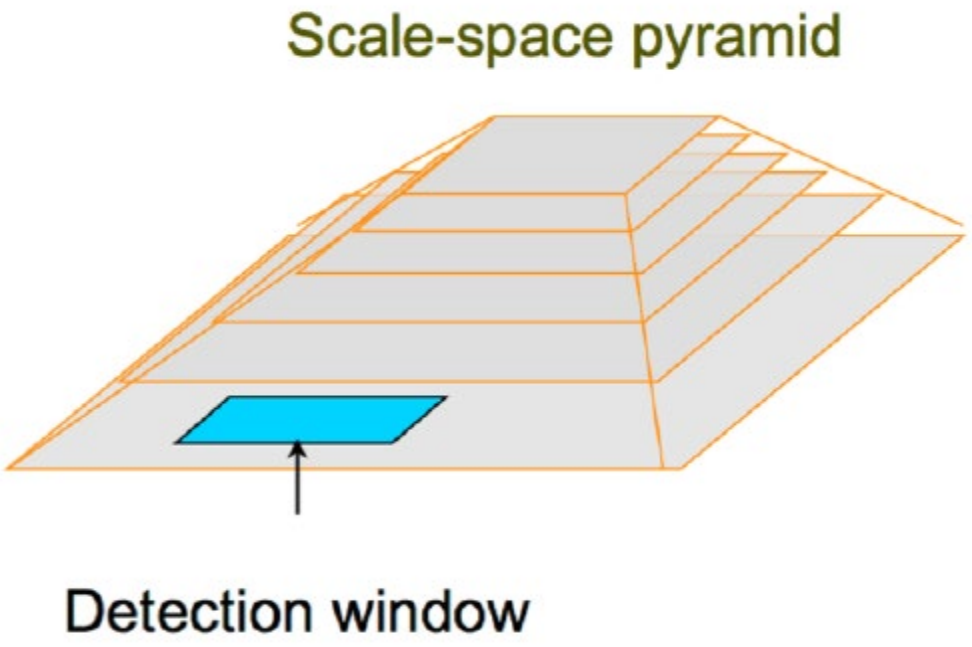
- Sliding Windows
  - "Slide" a box within the input image, or even across image scales (see below)
  - Classify each cropped image region inside the box and determine if it's an object of interest or not
- Region (Object) Proposals
  - Generate region (object) proposals -> how?
  - Classify each proposal and determine it's an object or not



Before the Resurgence of CNN: The HOG Detector

- Histogram of Oriented Gradients (HOG)
- Sliding window detectors finds the objects in 4 steps:
  - Inspect every window
  - Extract features in window
  - Classify & accept window if score > threshold
  - Clean-up (post-processing) stage

- Step 1: Inspect every window
  - Objects can vary in sizes, what to do?
  - Sliding window + image pyramid!



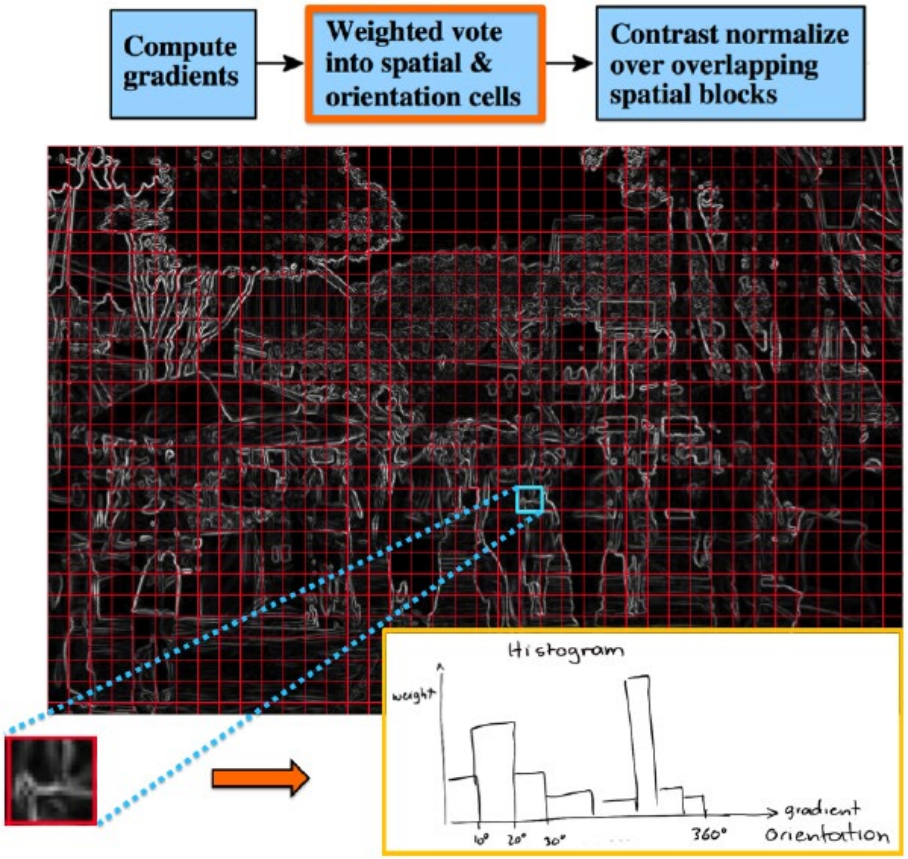
- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Ways to compute image gradients…

Mask Type	1D centered	1D uncentered	1D cubic-corrected	2x2 diagonal	3x3 Sobel
Operator	[-1, 0, 1]	[-1, 1]	[1, -8, 0, 8, -1]	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Miss rate at 10^{-4} FPPW	11%	12.5%	12%	12.5%	14%

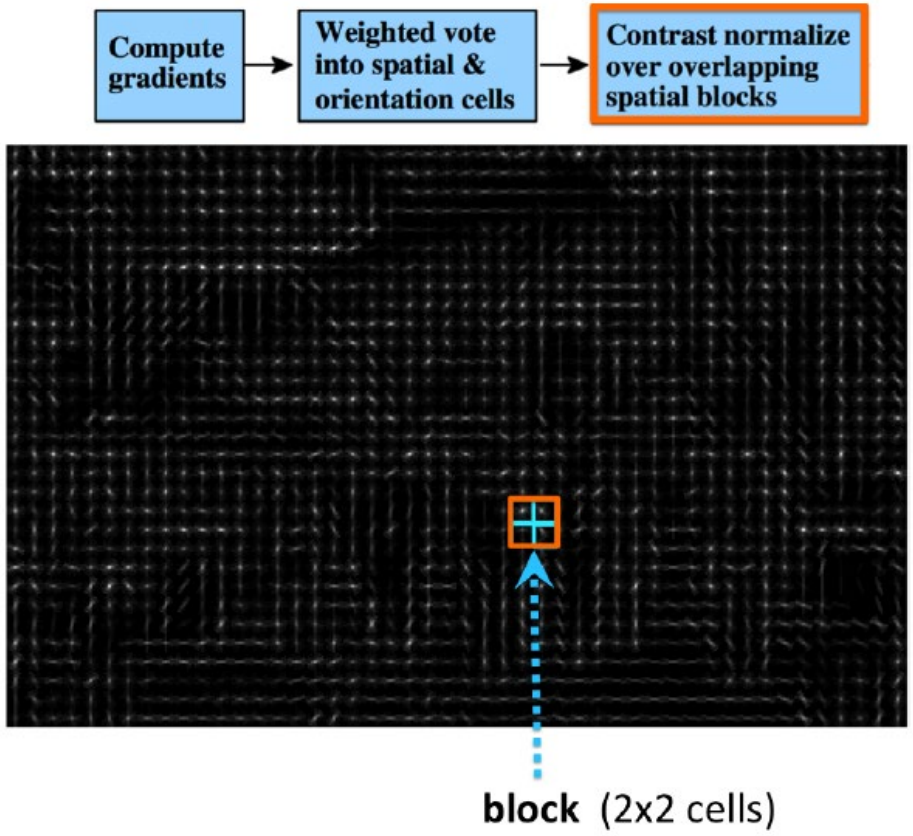
(Miss rate: smaller is better)

This gradient filter gives the best performance

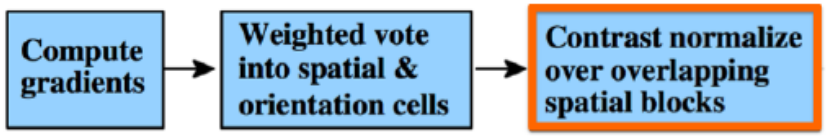
- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Divide the image into non-overlapping cells (grids) of 8 x 8 pixels
  - Compute a histogram of orientations in each cell, resulting in a 9-dimensional feature vector.

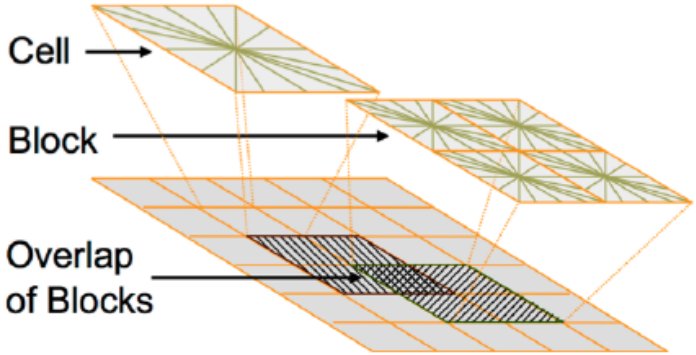


- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Divide the image into non-overlapping cells (grids) of 8 x 8 pixels
  - Compute a histogram of orientations in each cell (similar to SIFT), resulting in a 9-dimensional feature vector.
  - We now take blocks, where each has 2 x 2 cells, for HOG normalization.



- Step 2: Extract Features in Window
  - Compute a histogram of orientations in each cell (similar to SIFT), resulting in a 9-dimensional feature vector.
  - We now take blocks, where each has 2 x 2 cells, for HOG normalization
  - Normalize each feature vector, such that each block has unit norm. This does not change the dim of the feature, just the magnitude.

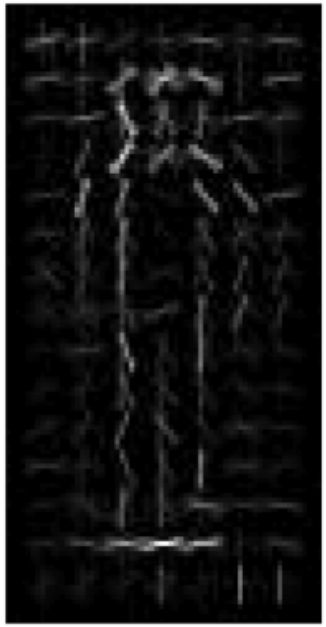




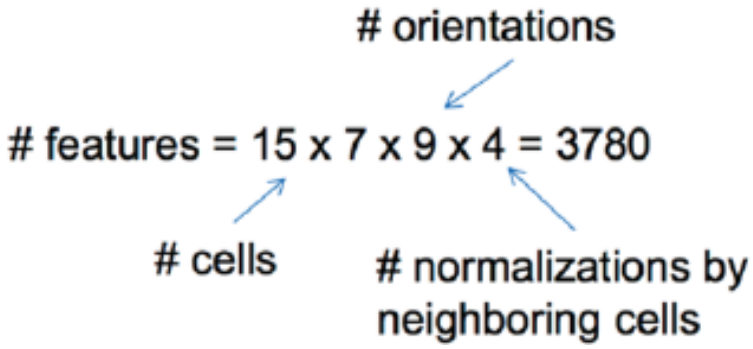
Feature vector *f* = [..., ..., ...]

L2 normalization in each block: $\mathbf{f} = \frac{\mathbf{f}}{\sqrt{||\mathbf{f}||_2^2 + \epsilon^2}}$

- Step 2: Extract Features in Window
  - Normalize each feature vector, such that each block has unit norm. This does not change the dim of the feature, just the magnitude.
  - For each class of *person*, window is 15 x 7 HOG cells.
  - Each cell is in 4 blocks thus has 4 different normalizations; we make each as a feature representation (in 3780 dimensions).
  - We vectorize each feature matrix in each window.



Final descriptor for window
(**person class in this case**)

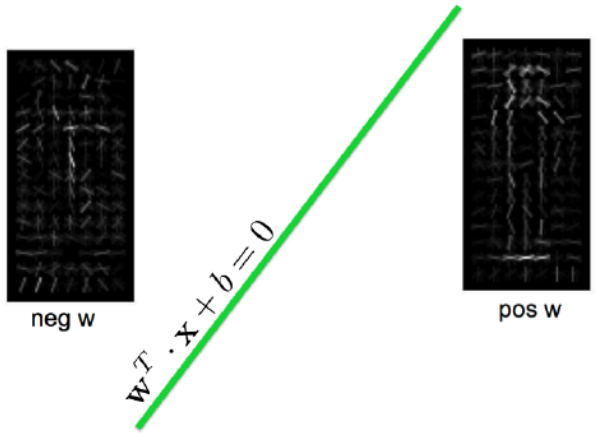


- Step 3: Detection (classify & accept window if score > threshold)
  - Training:

    learn a linear/nonlinear classifier to predict
    the presence of object class in each window

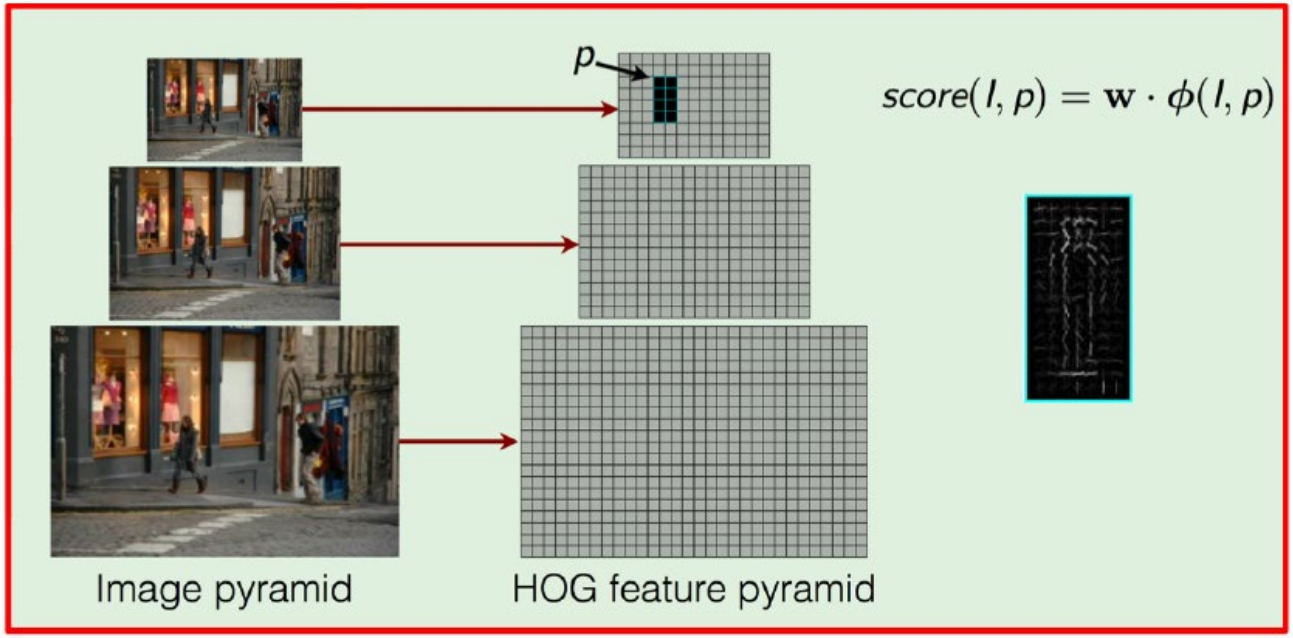
positive training examples

negative training examples

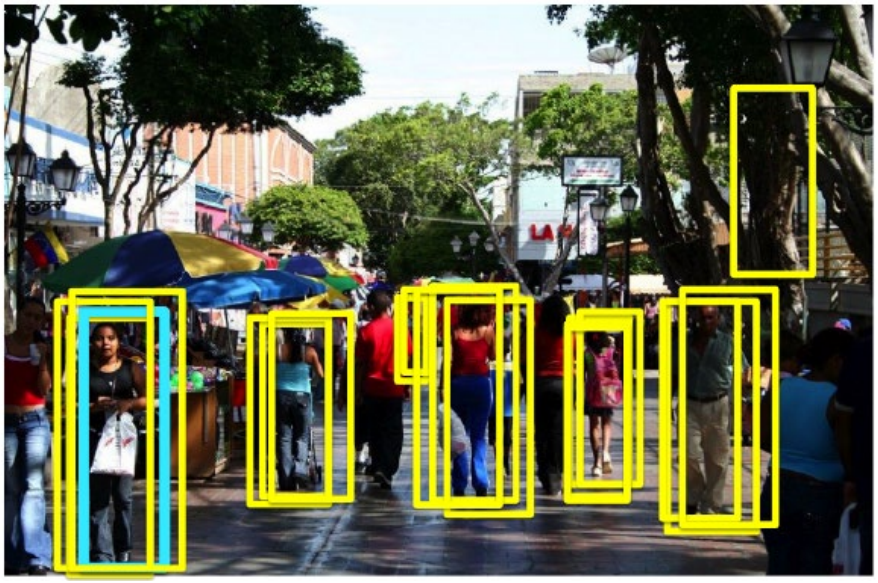


Train classifier. SVM (Support Vector Machines) is typically used.

- Step 3: Detection (Classify & accept window if score > threshold)
  - Training
  - Testing:
    Compute the score $\mathbf{w}^T\mathbf{x}+\mathbf{b}$ in each location, which can be viewed as performing cross-correlation (or convolution) with template **w** (and add bias **b**).



- Step 4: **Cleaning-Up**
  - Perform a greedy algorithm of **non-maxima suppression (NMS)** to pick the bounding box with highest score



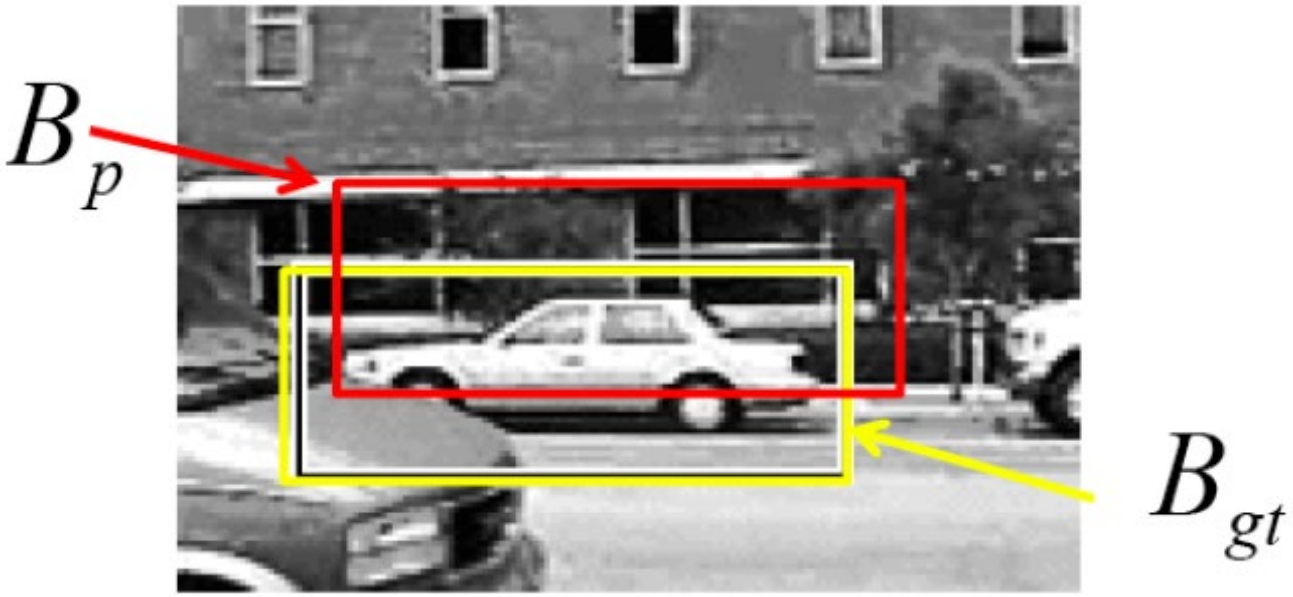
Non-maxima suppression (NMS)

$$\text{overlap} = \frac{\text{area}(box_1 \cap box_2)}{\text{area}(box_1 \cup box_2)} > 0.5$$

- Remove all boxes that overlap more than XX (typically 50%) with the chosen box

- **Evaluation**
  - IoU (intersection over union)
    - E.g, detection is correct if IoU between bounding box and ground truth > 50%

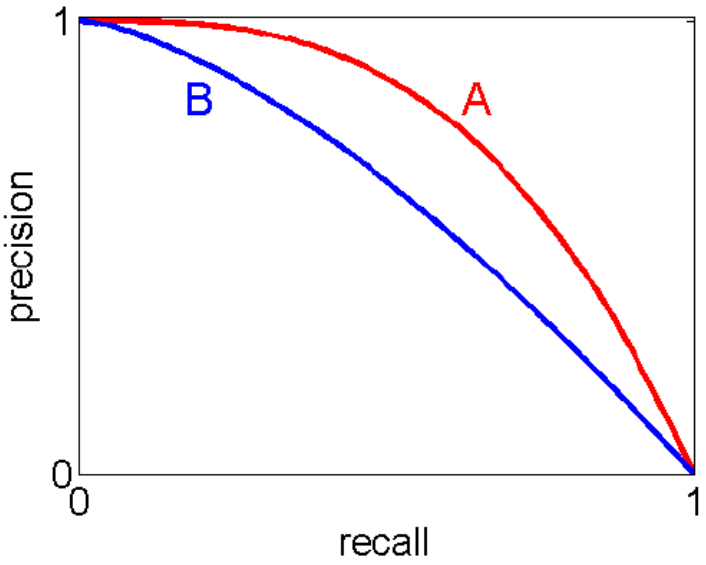
$$a_0 = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}$$



- **Evaluation**
  - IOU (intersection over union)
    - Mean IOU (mIOU): average IOU across classes
  - Precision & Recall
    - Sort all the predicted boxes according to scores, in a descending order
    - For each location in the sorted list,
      we compute precision and recall obtained when using top k boxes in the list.

$$\text{recall} = \frac{\#\text{correct boxes}}{\#\text{ground-truth boxes}}$$

$$\text{precision} = \frac{\#\text{correct boxes}}{\#\text{all predicted boxes}}$$

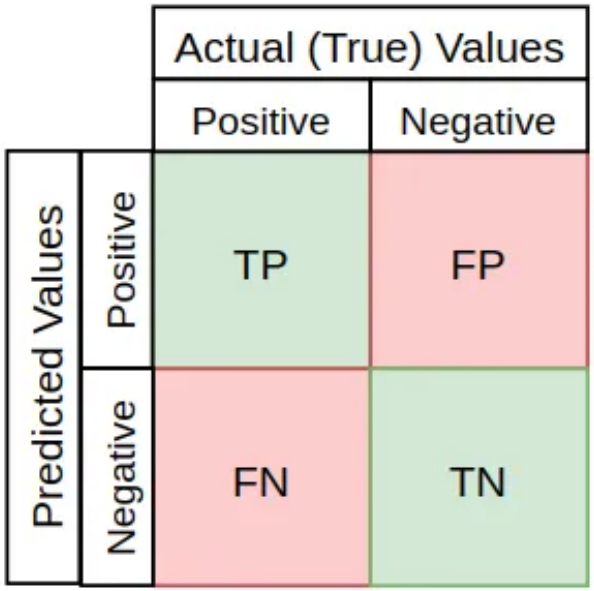


- Evaluation
  - IOU (intersection over union)
  - Precision and Recall

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

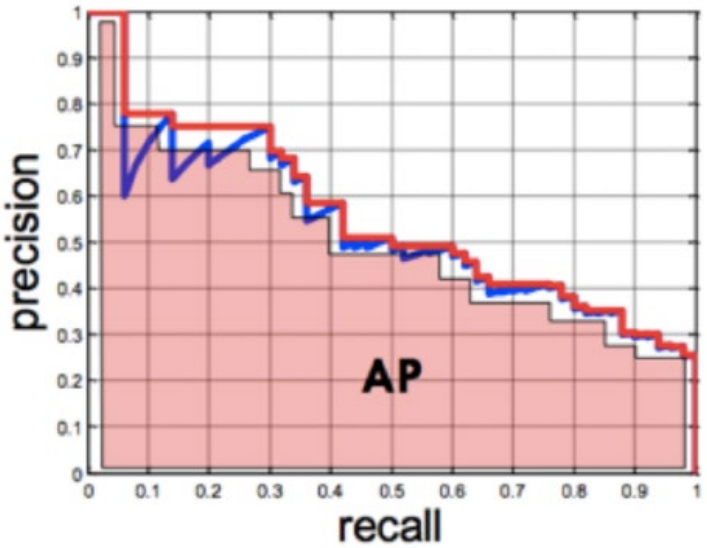
$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



  - F1-score: harmonic mean of P & R, i.e.,

$$2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}}$$

- Evaluation (cont'd)
  - IoU (intersection over union)
  - Precision, Recall, F1-score
  - Average Precision (AP):
    - Compute the area under P-R curve
    - mean Average Precision (mAP): average of AP across classes



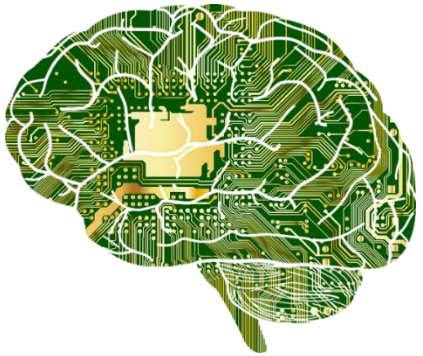
Something to Think About…

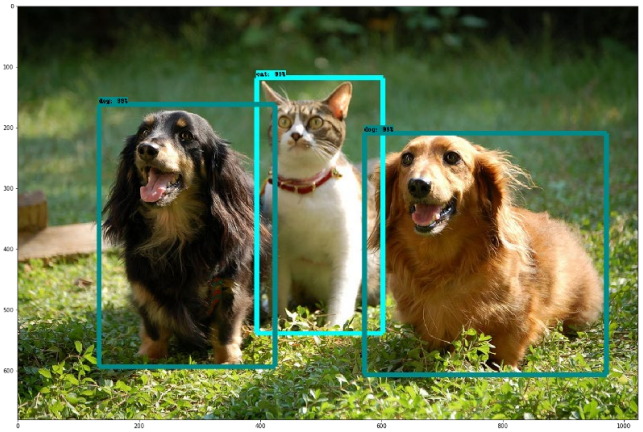
- Sliding window detectors work
  - *very well* for faces
  - *fairly well* for cars and pedestrians
  - *badly* for cats and dogs

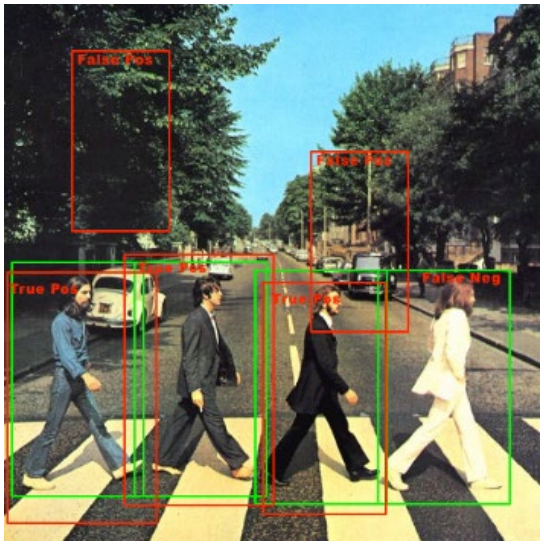
- Why are some classes easier than others?

What to Covered Today…

- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection

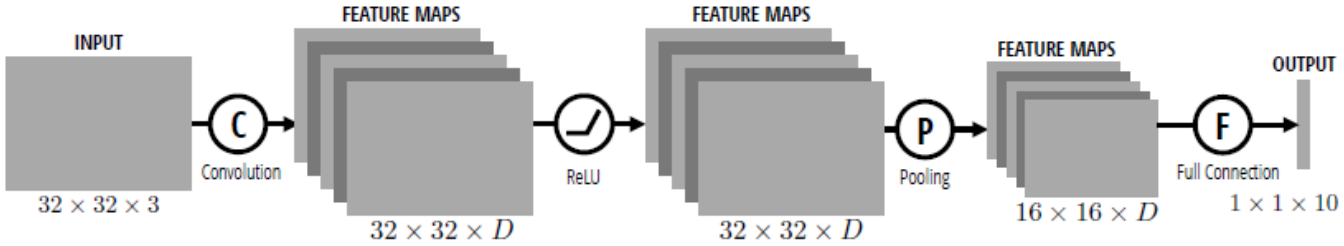


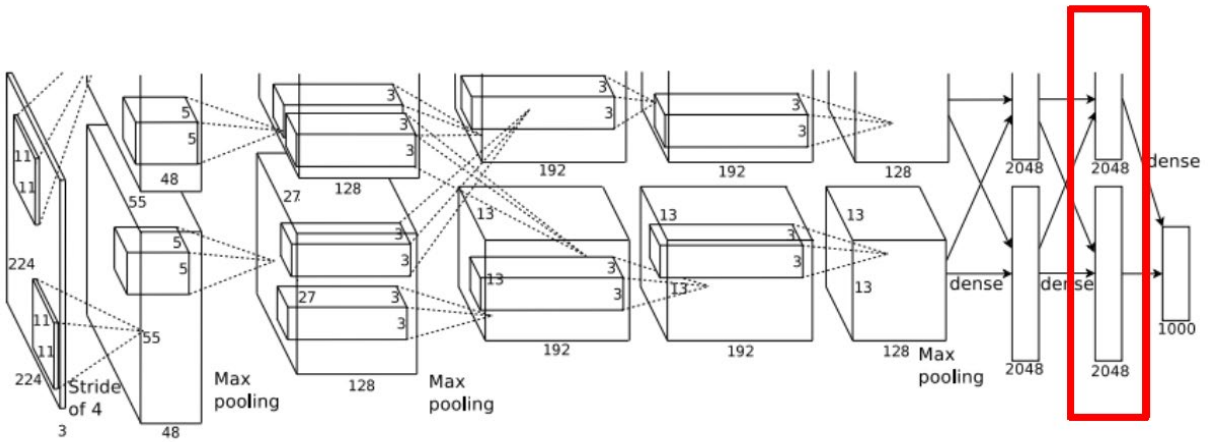




Recall that

- Visual Features derived by Convolutional Neural Networks



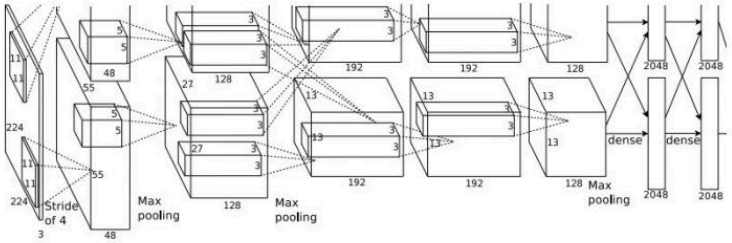


CNN as Feature Extractor

Image credit: Justin Johnson

CNN as Feature Extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

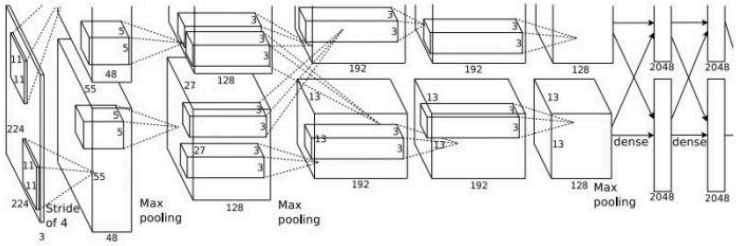


Dog? YES
Cat? NO
Background? NO

Slides by Justin Johnson

CNN as Feature Extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

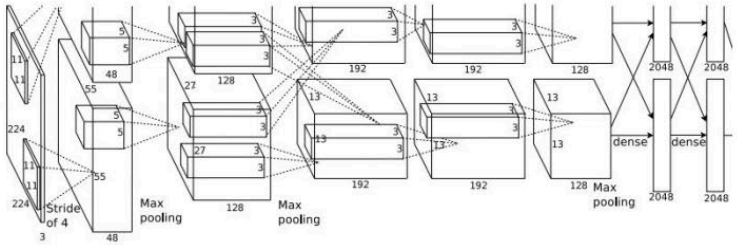


Dog? YES
Cat? NO
Background? NO

Slides by Justin Johnson

CNN as Feature Extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

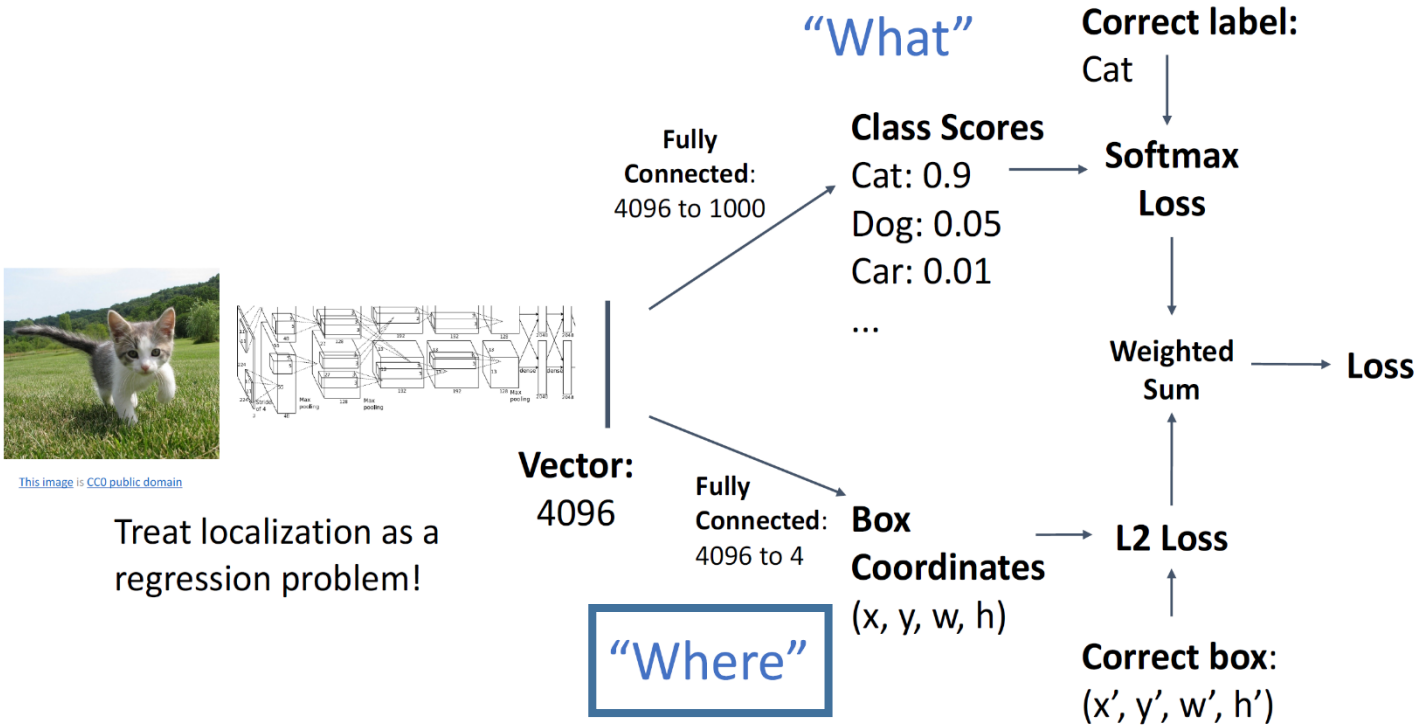
Slides by Justin Johnson

CNN as Feature Extractor

- What could be the problems?
  - Suppose we have an image of 600 x 600 pixels.
    If sliding window size is 20 x 20,
    then have (600-20+1) x (600-20+1) = ~330,000 windows to compute.
  - What if more accurate results are needed,
    need to perform multi-scale detection by
    - Resize image
    - Multi-scale/shape sliding windows
  - For each image, we need to forward pass image regions through CNN for at least ~330,000 times. -> **Slow!!!**

Recap: CNN for Object Detection

- Need to deal with more than one object
  - How?



Slide credit: UMich EECS 498-007

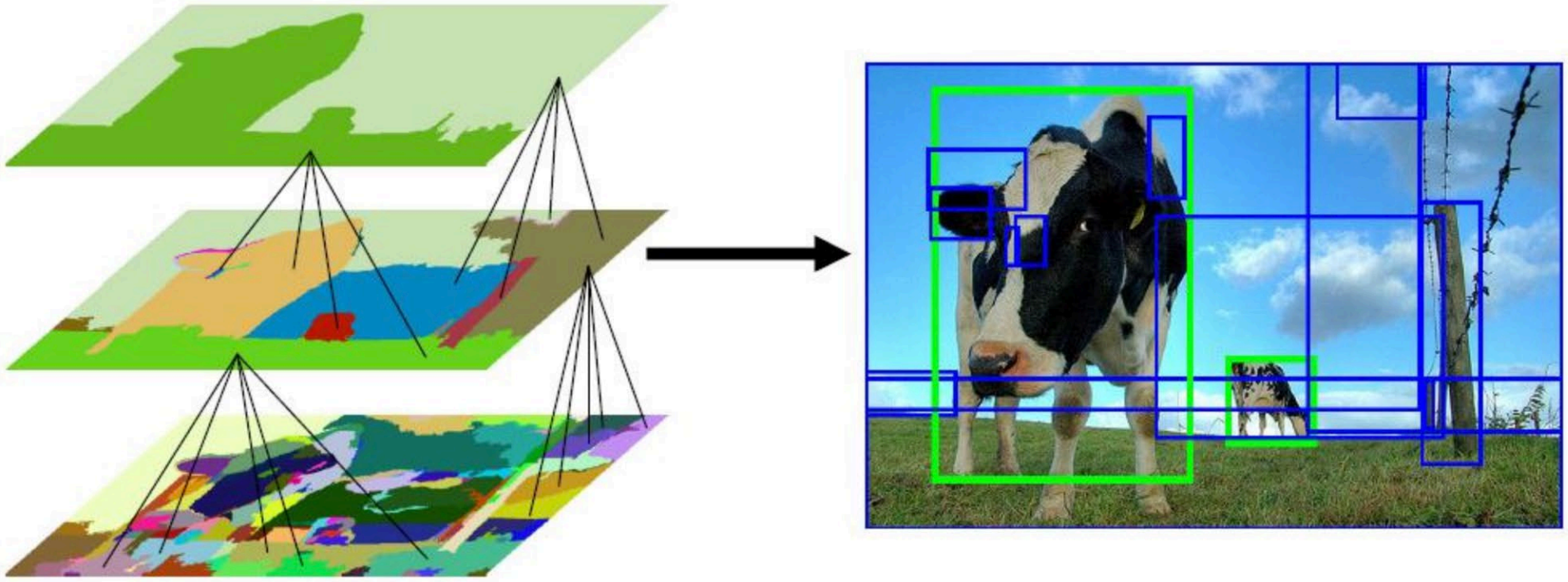
Two-Stage vs. One-Stage Object Detection

Methods

- Sliding Windows
- **Two−stage Frameworks**
  - R−CNN
  - Fast R−CNN
  - Mask R−CNN
  - ⋮
- **One−stage Frameworks**
  - YOLO
  - YOLOv2
  - YOLOv3
  - ⋮

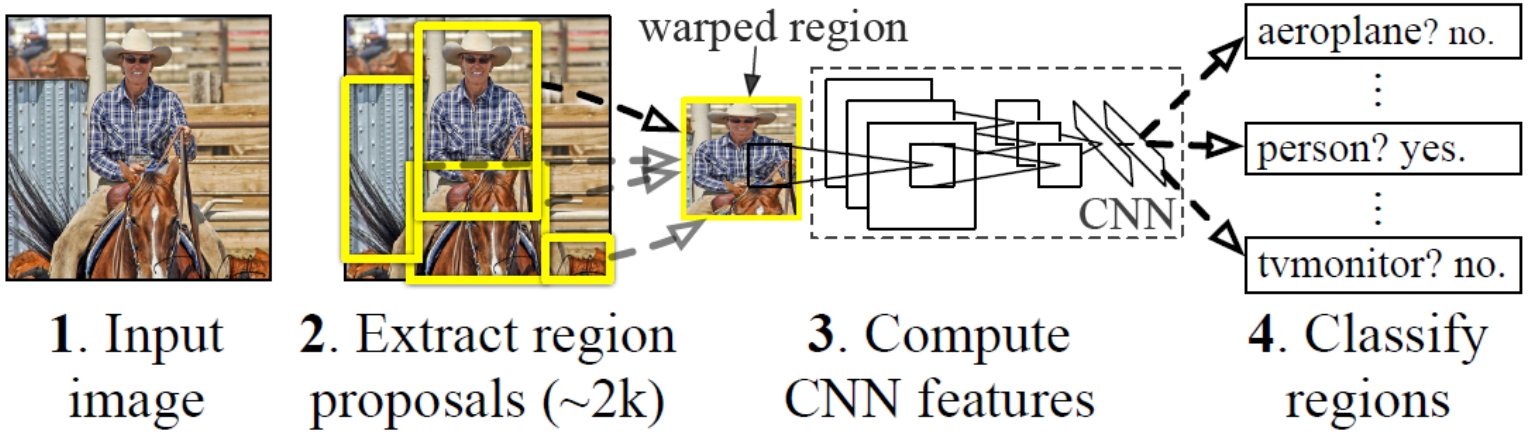
Region Proposal

- Solution
  - Use pre-processing algorithms to filter out some regions first, and feed the regions of interest (i.e., region proposals) into CNN
  - E.g., selective search



Uijilings et al. IJCV 2013

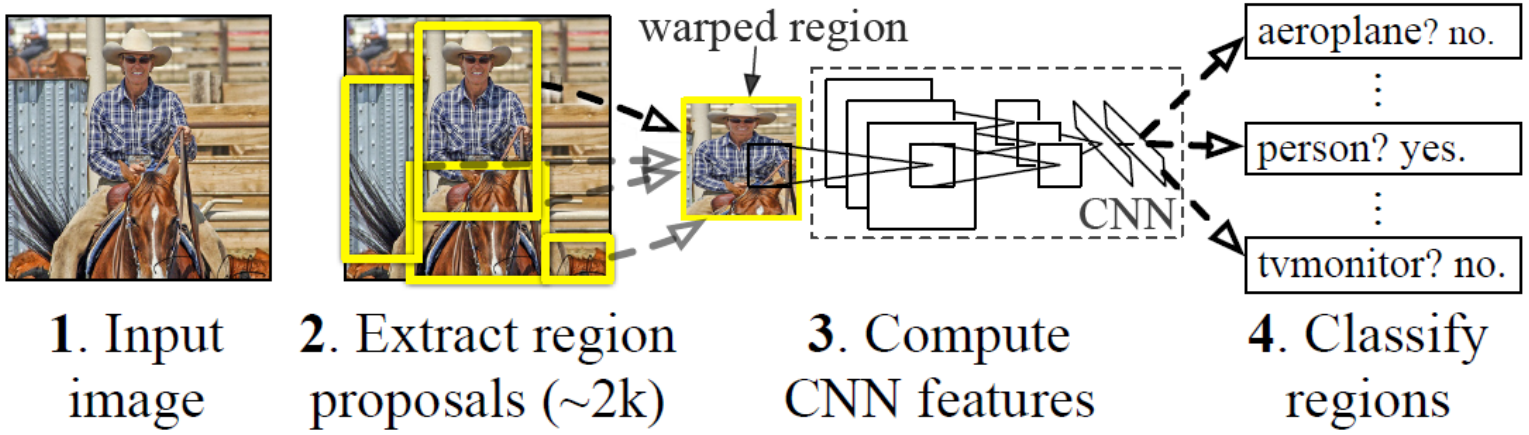
R-CNN (Girshick et al. CVPR 2014)



- Replace sliding windows with "selective search" region proposals (Uijilings et al. IJCV 2013)
- Extract rectangles around regions and resize to 227x227 pixels
- Extract features with fine-tuned CNN (e.g., initialized with network pre-trained on ImageNet)
- Classify last layer of network features with linear classifiers (e.g., SVM/MLP), and refine bounding box localization (bbox regression) simultaneously

http://arxiv.org/pdf/1311.2524.pdf

R-CNN (Girshick et al. CVPR 2014)

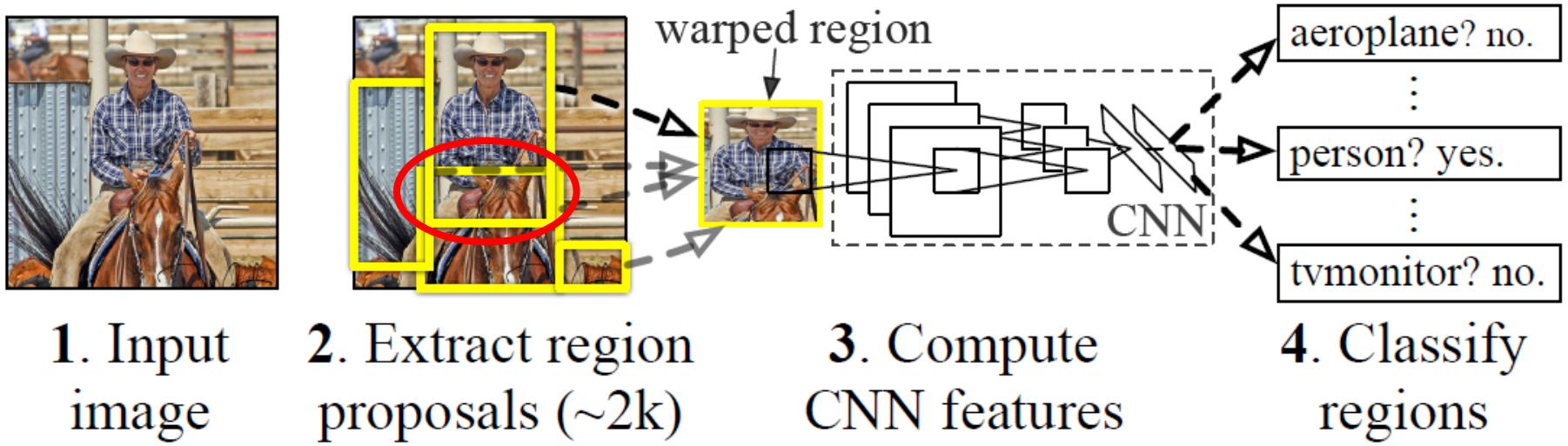


- Ad hoc training objectives:
  - Object class: Fine-tune network with softmax classifier (log loss)
  - Object class: Train post-hoc linear SVMs for each class (hinge loss)
  - Bbox location: Train post-hoc bounding-box regressors (least squares loss)
- Training is extremely slow with lots of disk space.
- Implementation/testing cannot be done in real time.

http://arxiv.org/pdf/1311.2524.pdf

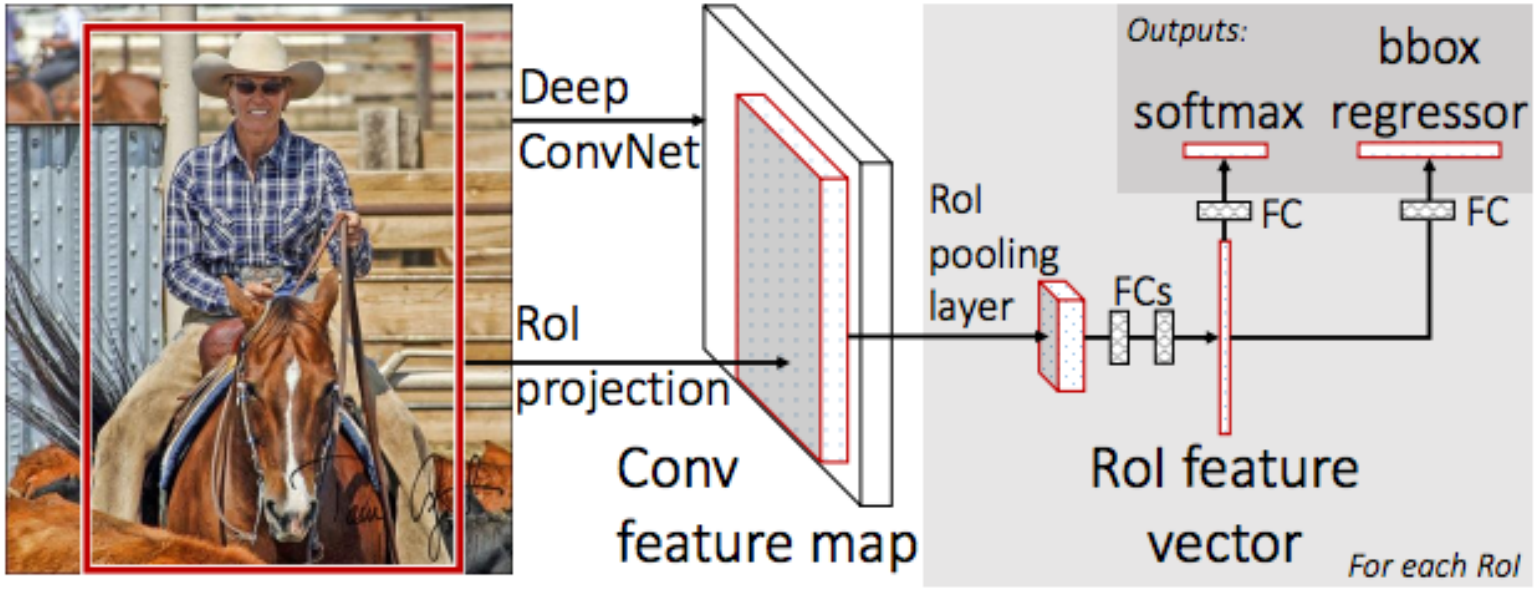
R-CNN (Girshick et al. CVPR 2014)

- What could be the problems?
  - Repetitive computation!
    For overlapping regions, we feed it multiple times into CNN



Fast R-CNN (Girshick ICCV 2015)

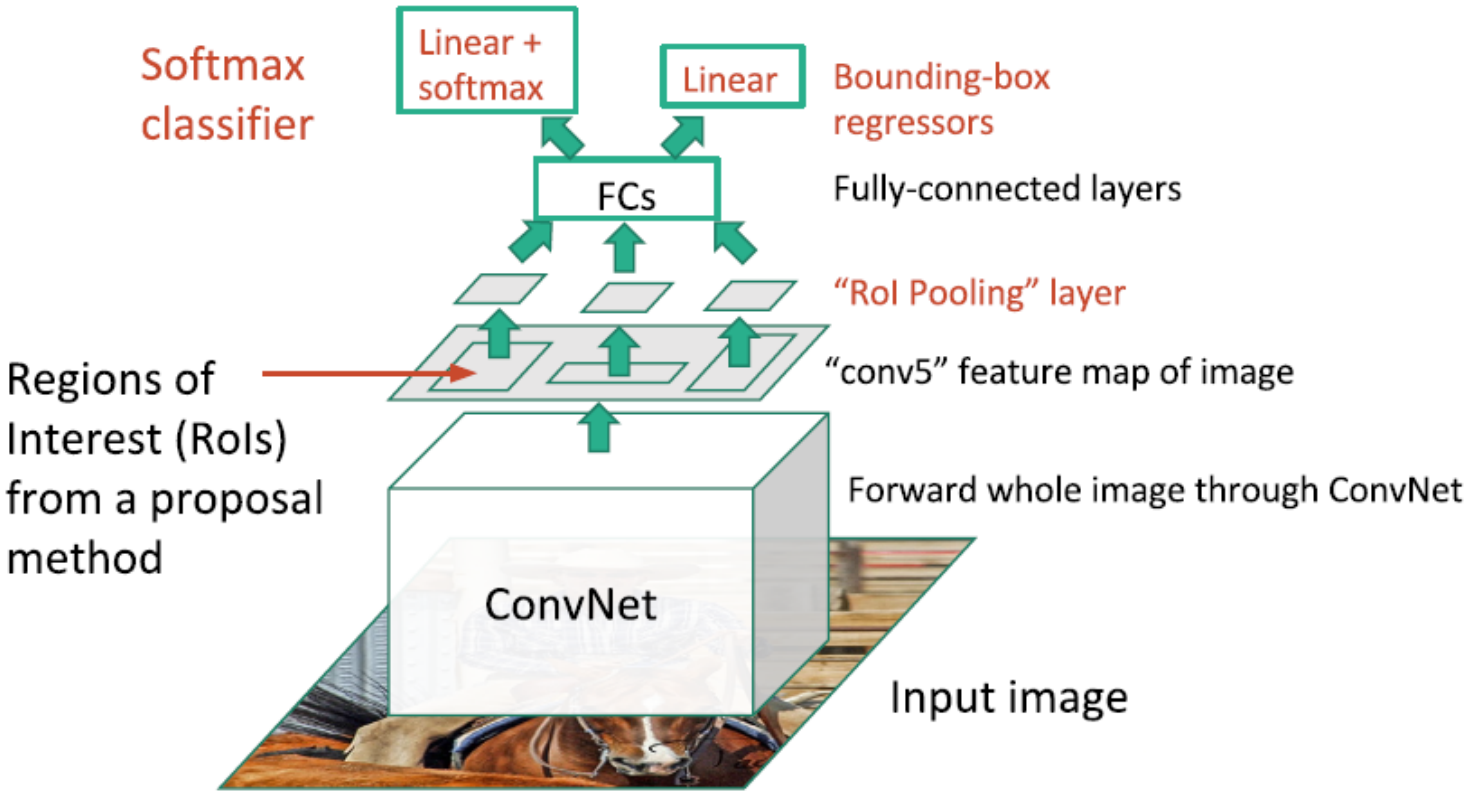
- Solution
  - Why not feed the whole image into CNN only once?
  - Then, crop the feature map instead of the image itself



https://arxiv.org/pdf/1504.08083.pdf

Fast R-CNN (Girshick ICCV 2015)

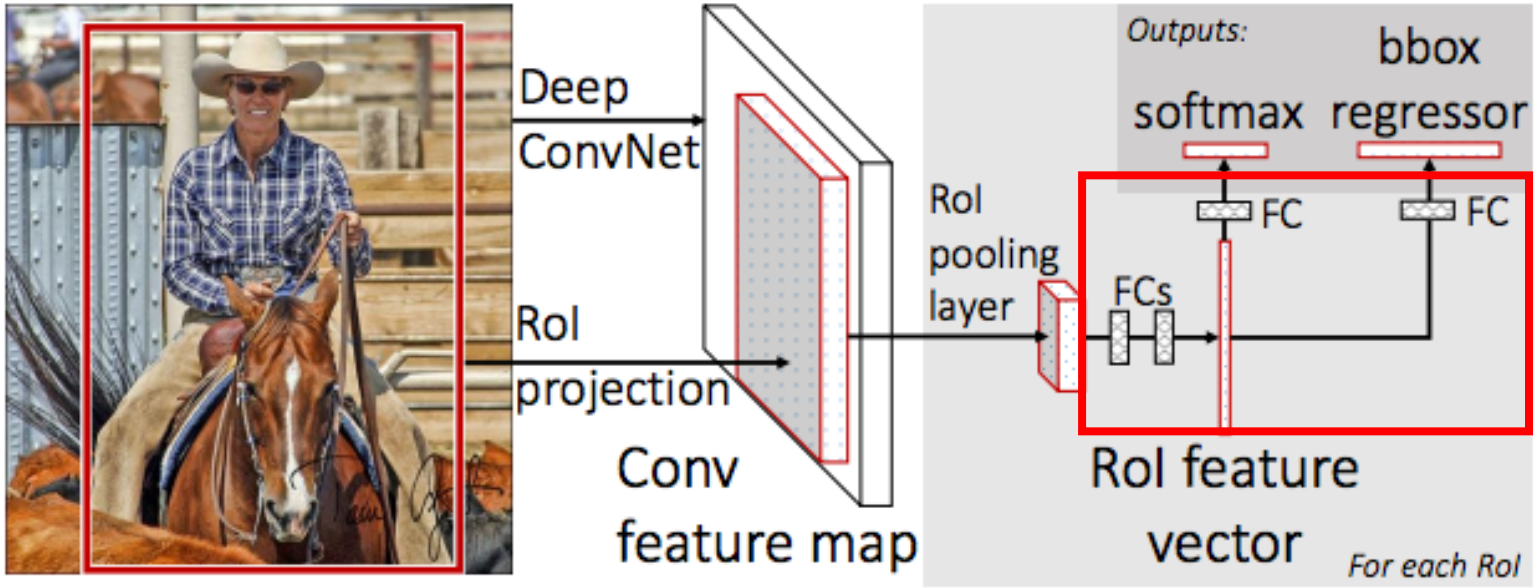
- Solution



https://arxiv.org/pdf/1504.08083.pdf

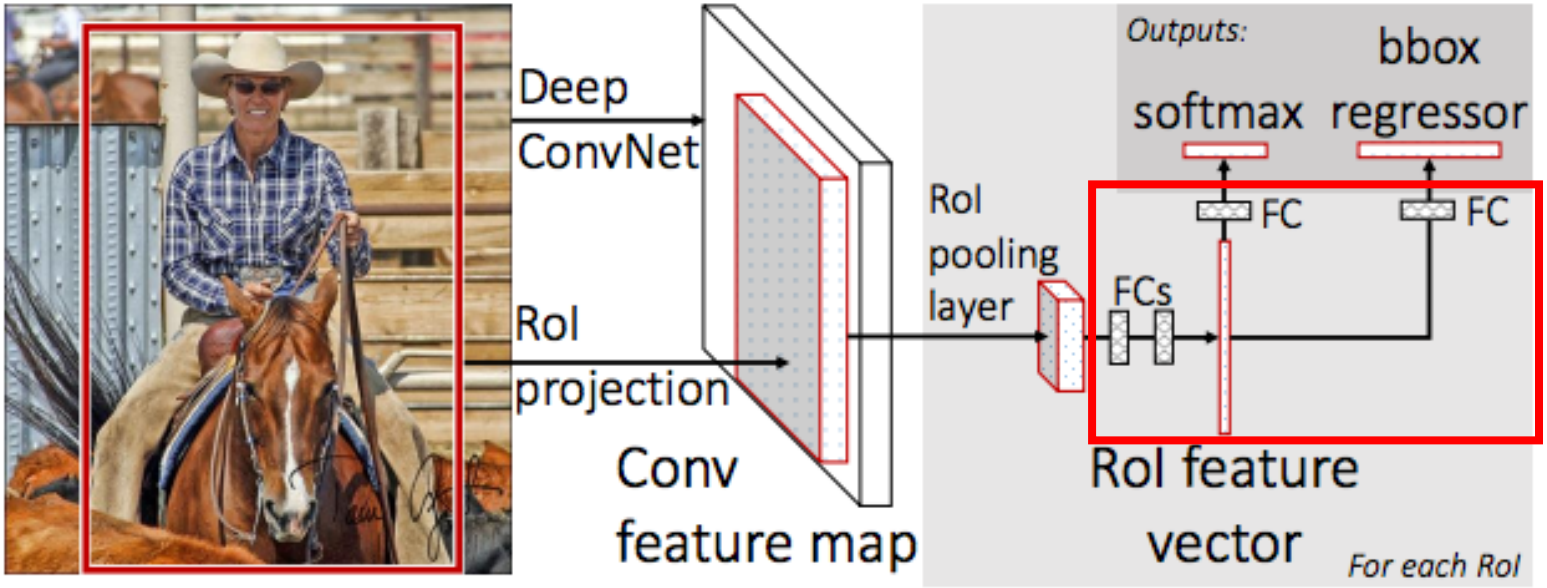
Fast R-CNN (Girshick ICCV 2015)

- Wait…what about the image feature to be processed?
  - Any potential problem?



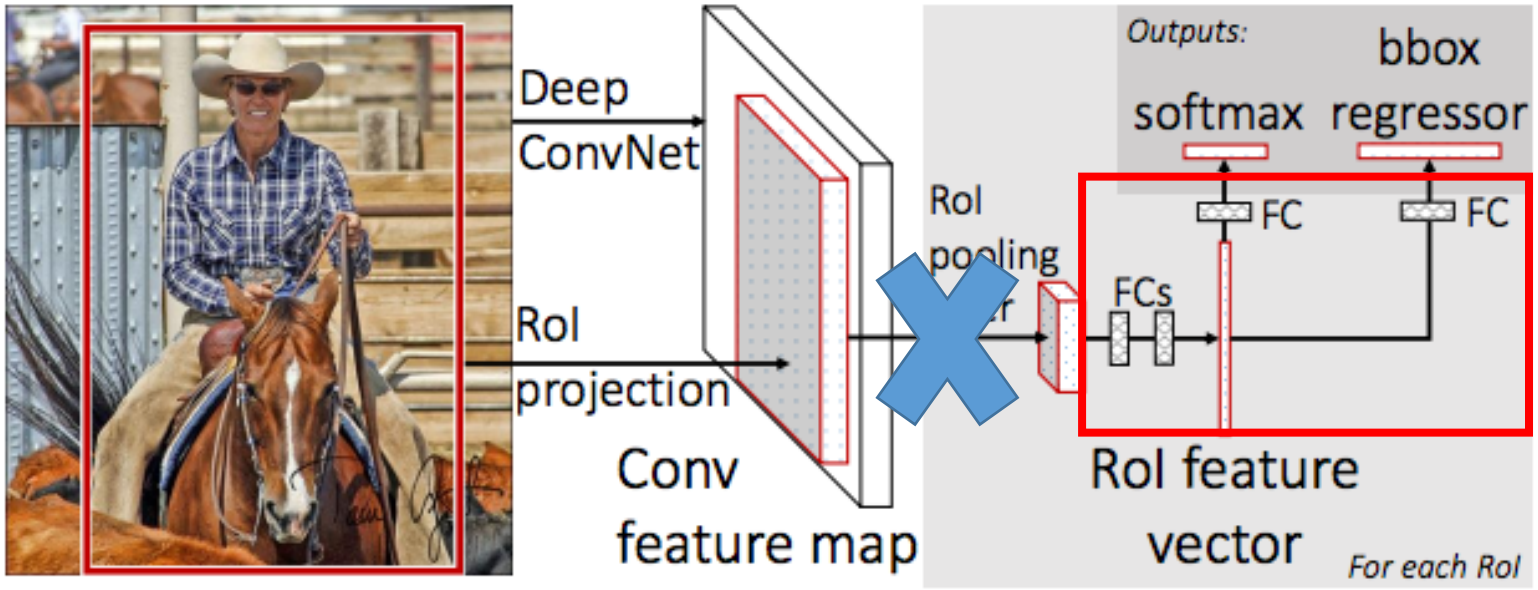
Fast R-CNN (Girshick ICCV 2015)

- Features need to be of the same size.
- How to crop features?
  - Since we have fully-connected layers, the size of feature map for each bounding box should be a fixed number



Fast R-CNN (Girshick ICCV 2015)

- How to crop features?
  - Since we have fully-connected layers, the size of feature map for each bounding box should be a fixed number
  - Resize/Interpolate the feature map as fixed size?
    - Not optimal. This operation is hard to backprop.
      -> we cannot train the conv layers in CNNs…

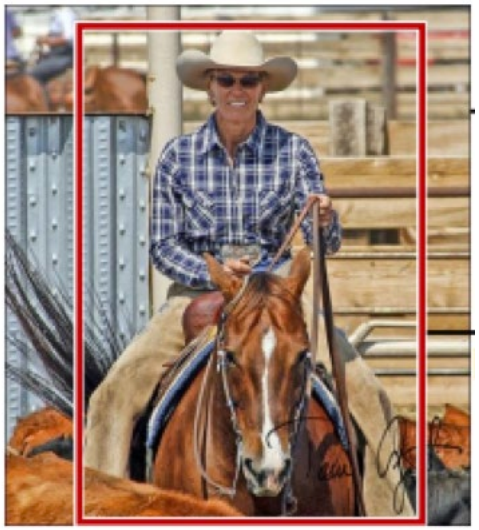


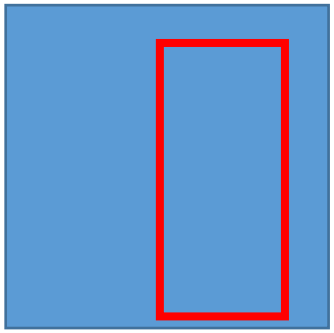
Fast R-CNN (Girshick ICCV 2015)

- How to crop features?
  - Since we have fully-connected layers, the size of feature map for each bounding box should be a fixed number
  - Resize/Interpolate the feature map as fixed size?
    - Not optimal. This operation is hard to backprop.
      -> we cannot train the conv layers for this problem…
- RoI Pooling
  - How to crop regions of various sizes but resulting in the same size?

RoI Pooling

- Step 1:
  Get bounding box for feature map from bounding box for image
  - Due to the (down)convolution/pooling operations,
    feature map would have a smaller size than the original image.





Feature map

RoI Pooling

- Step 2:
  Divide cropped feature map into fixed number of sub-regions
  - The last column and last row might be smaller

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Feature map
4 x 4 x 1

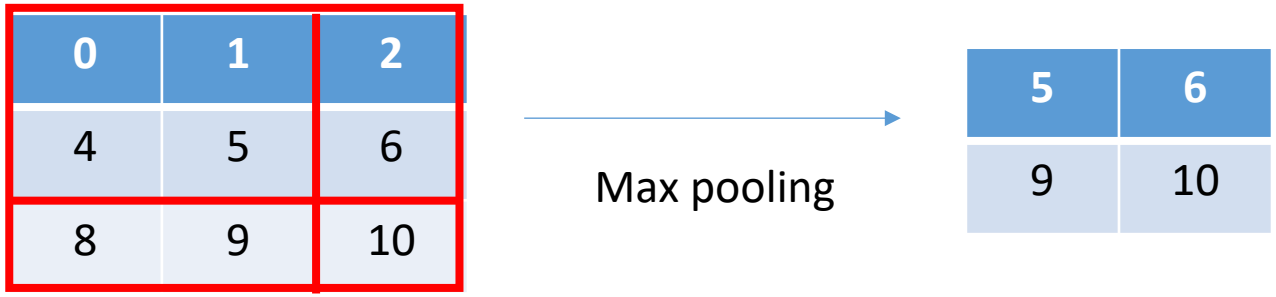
Make it as
2x2 grids

0	1	2
4	5	6
8	9	10

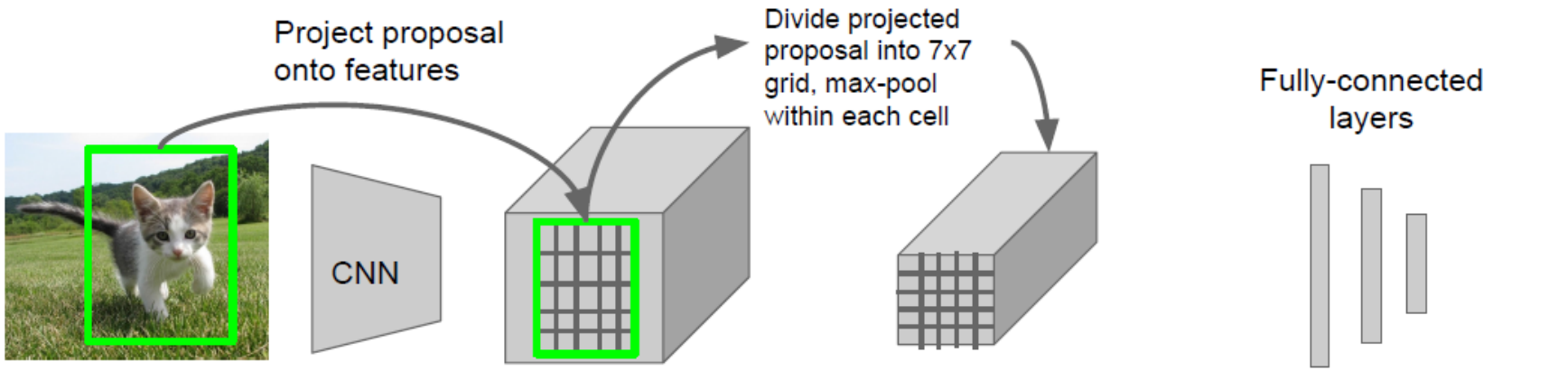
1	2	3
5	6	7
9	10	11
13	14	15

RoI Pooling

- Step 3:
  For each sub-region, perform max pooling (pick the max one)



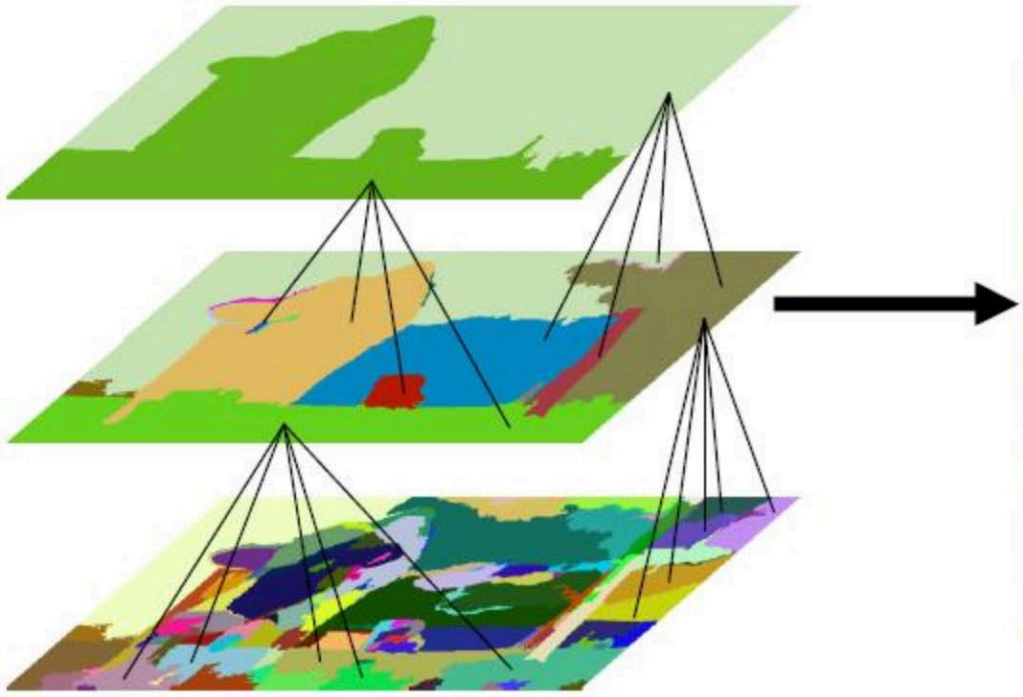
RoI Pooling

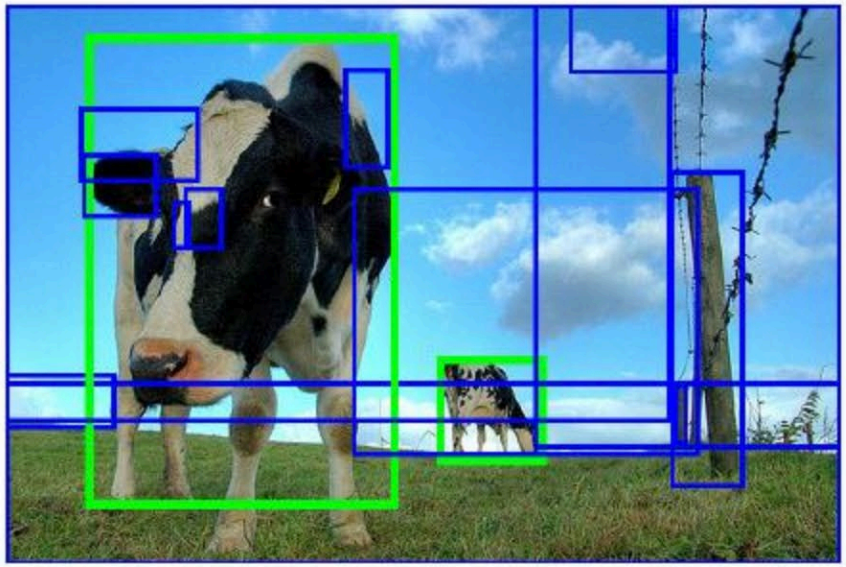


Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN (Girshick ICCV 2015)

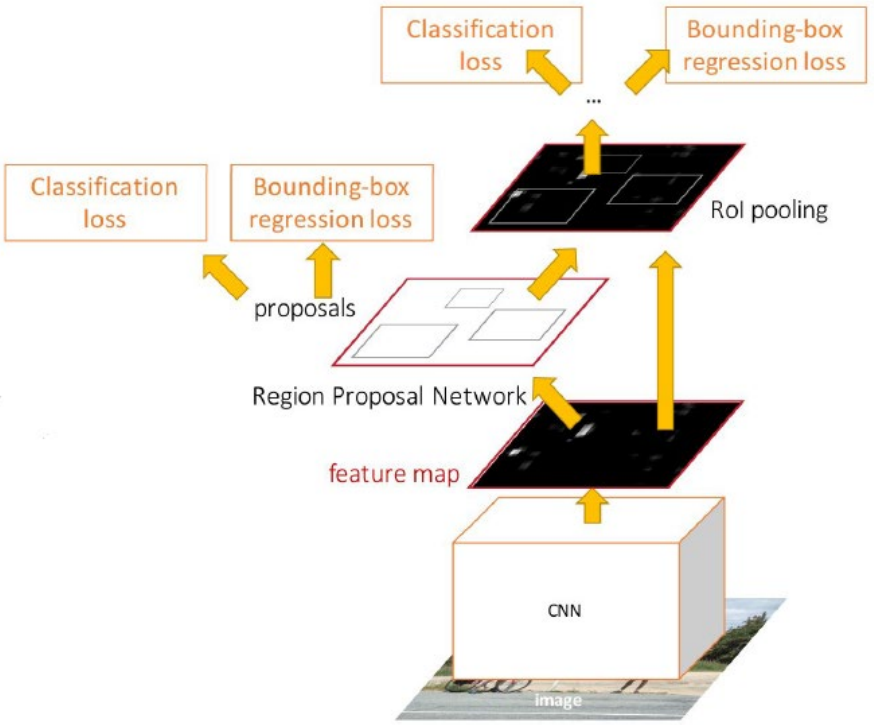
- What could be the problems?
  - We still need to collect the region proposals from a pre-processing step, which does not allow end-to-end learning.





Faster R-CNN (Ren et al. NIPS/NeurIPS 2015)

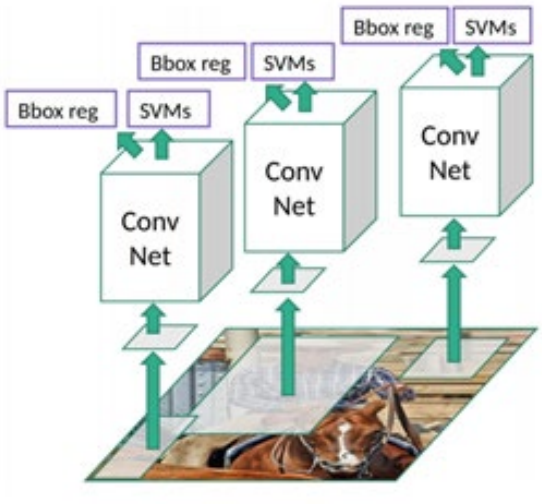
- Solution
  - Why not generate region proposals using CNN?
    -> Insert Region Proposal Network (RPN) to predict proposals from features
  - Jointly train with 4 losses:
    - RPN classification loss
    - RPN regress box coordinates
    - Final classification loss
    - Final box coordinates

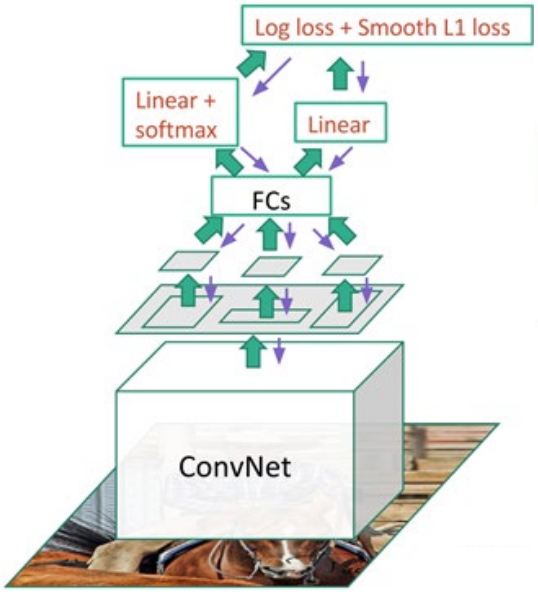


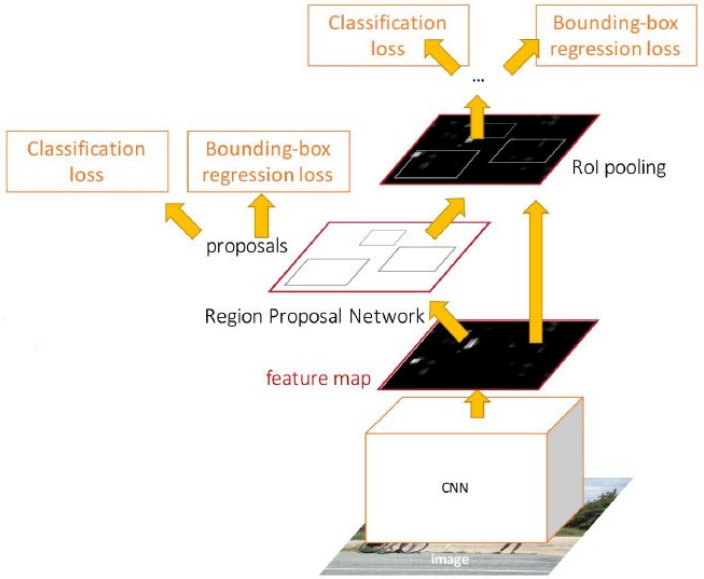
https://arxiv.org/pdf/1506.01497.pdf

Image credit: http://zh.gluon.ai/chapter_computer-vision/object-detection.html

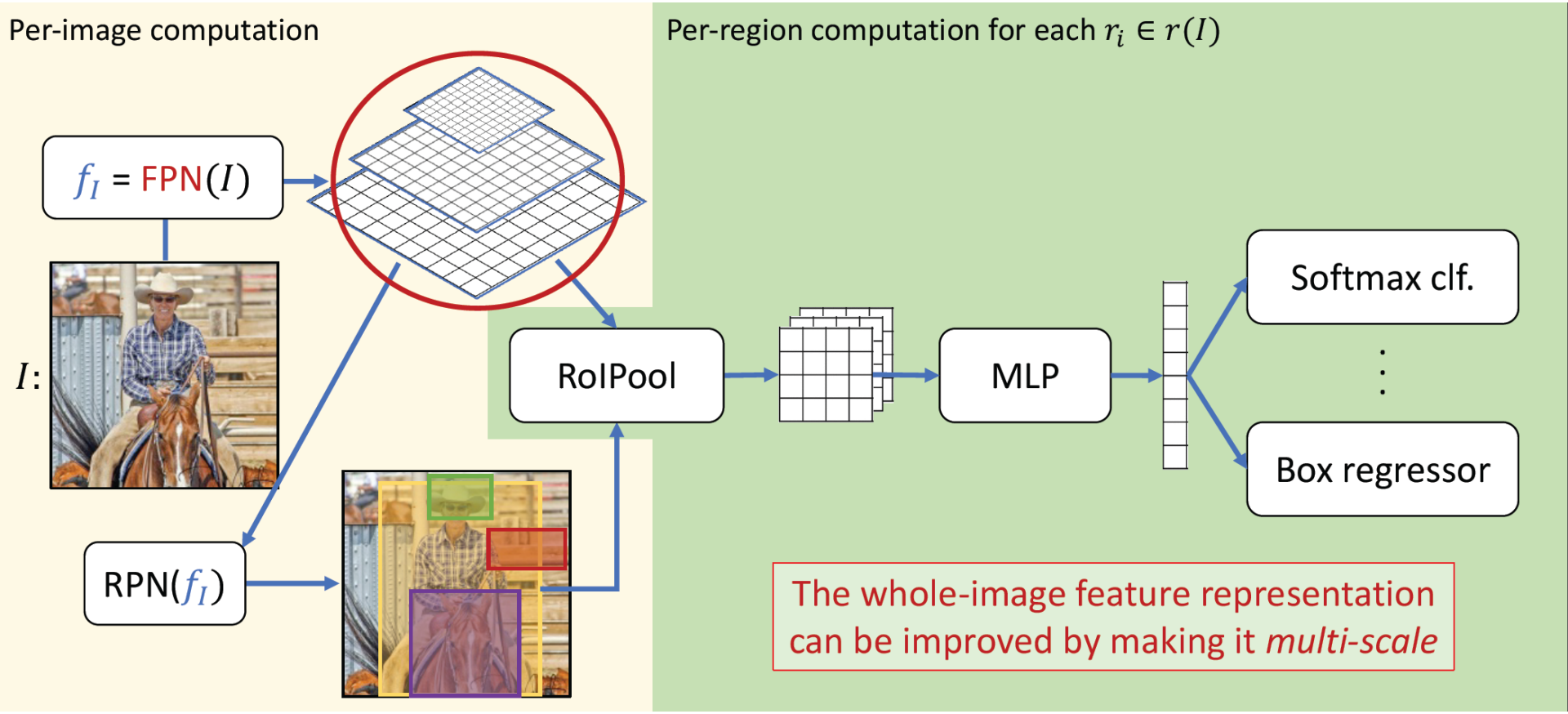
R-CNN, Fast R-CNN, & Faster R-CNN







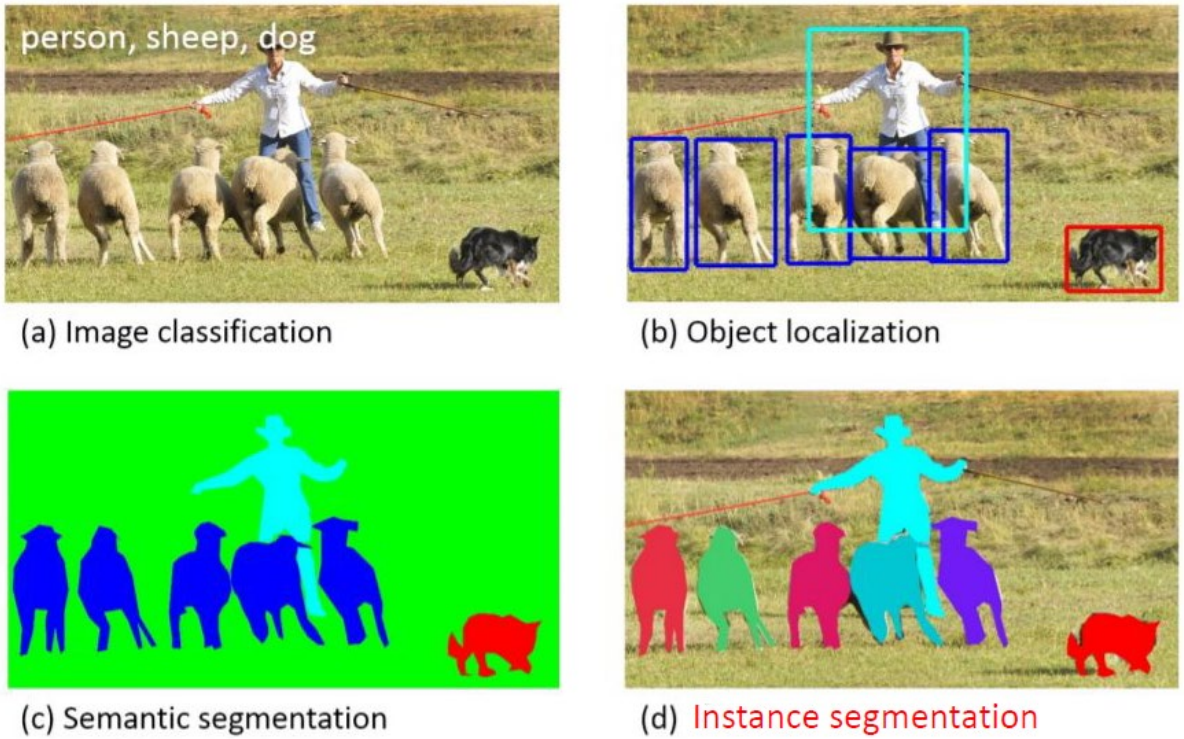
Faster R-CNN with Feature Pyramid Network



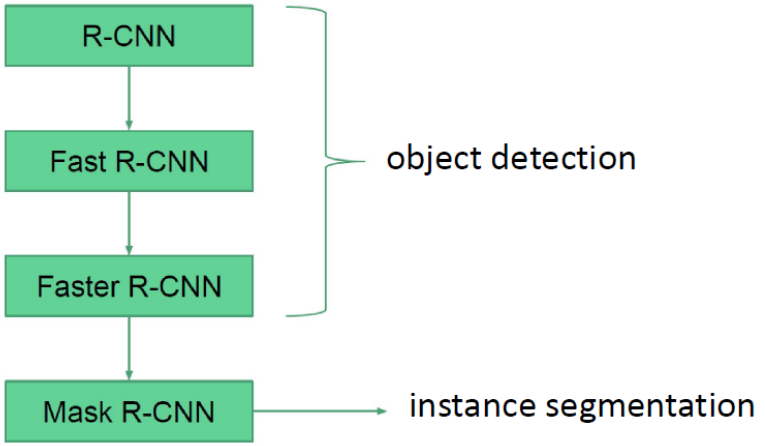
Slide credit: Ross Girshick

Faster R-CNN (Ren et al. NIPS 2015)

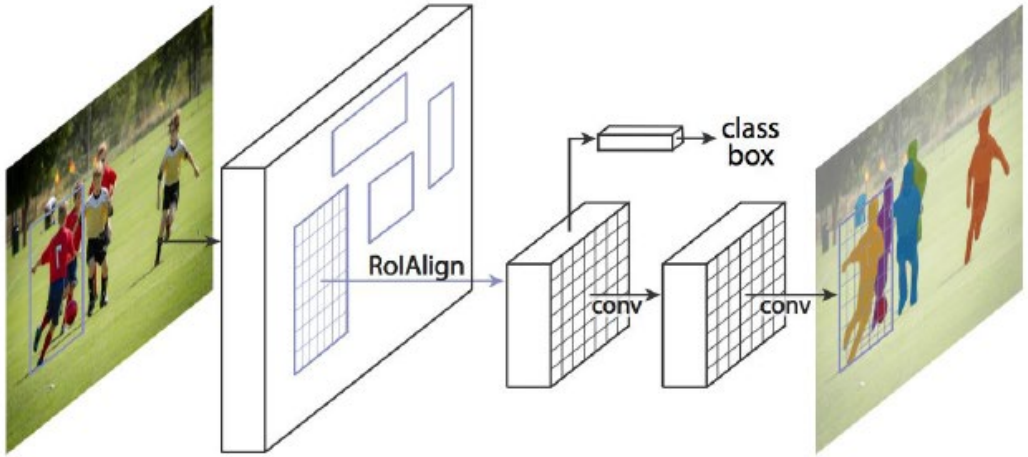
- What could be the problems
  - Two-stage detection pipeline is still too slow for real-time detection in videos…
  - What about instance-wise information?



Mask R-CNN (ICCV2017)

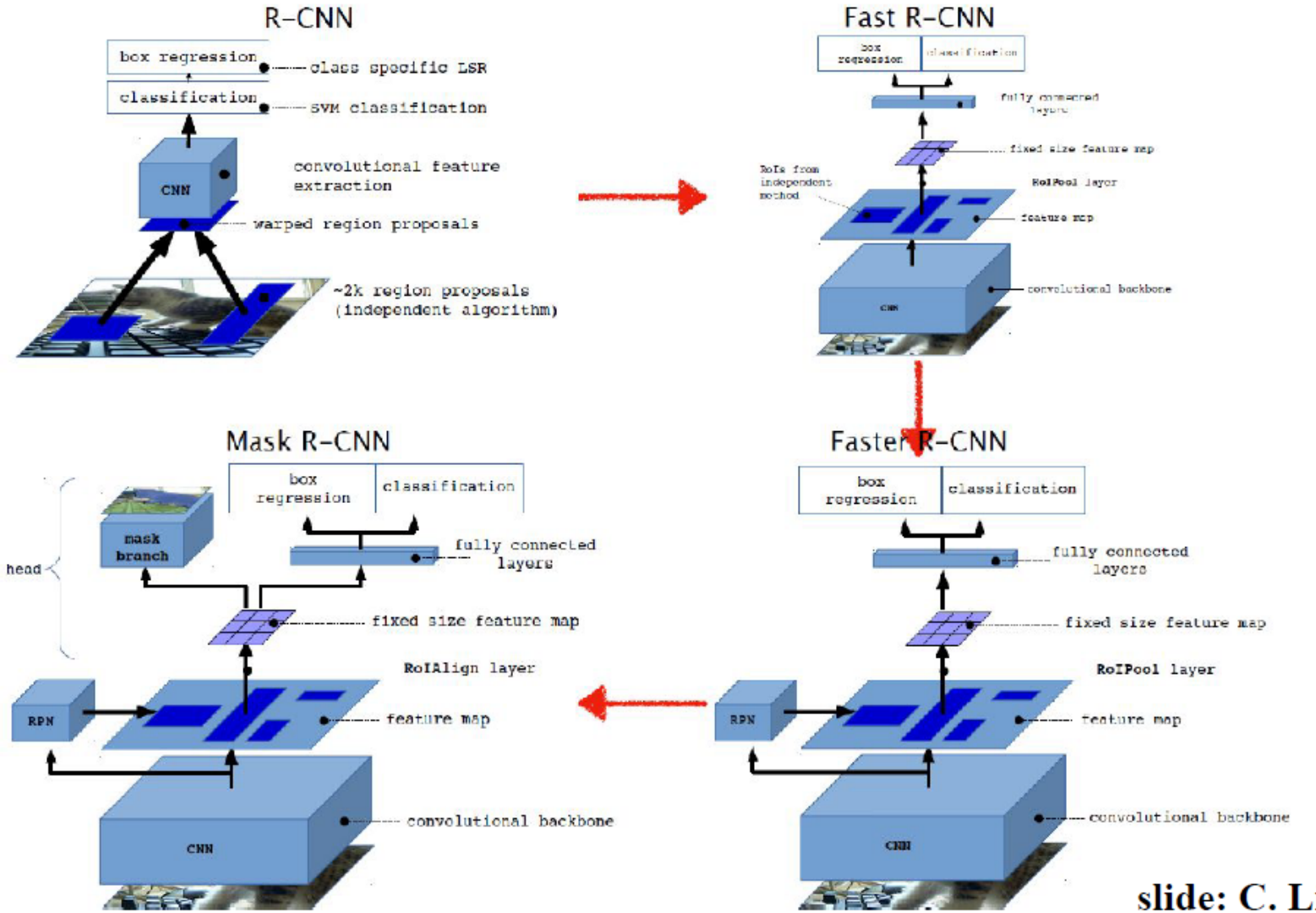


- Still a 2-stage detector
- Goals:
  - Refined detection + precise segmentation
  - Faster R-CNN + FCN
- Overall design:
  - Use of ResNet or feature pyramid net for feature extraction
  - Use RPN to produce proposals (w/ ROI align)
  - Use one detection branch for box classification + regression
  - Use one



He et al, "Mask R-CNN", arXiv 2017

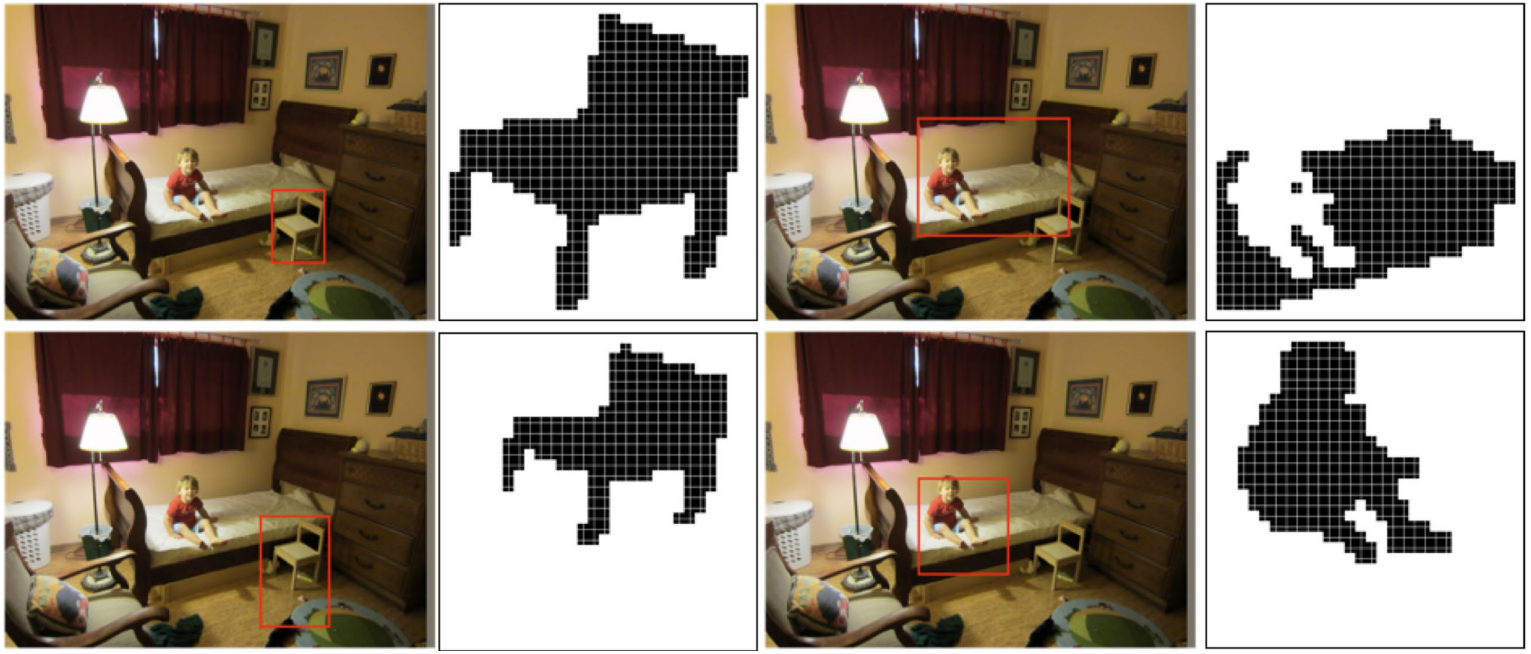
R-CNN Family



slide: C. Lim

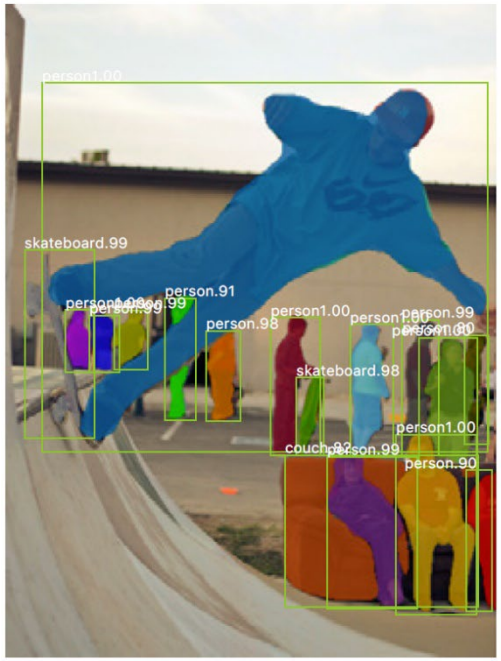
Mask R-CNN (cont'd)

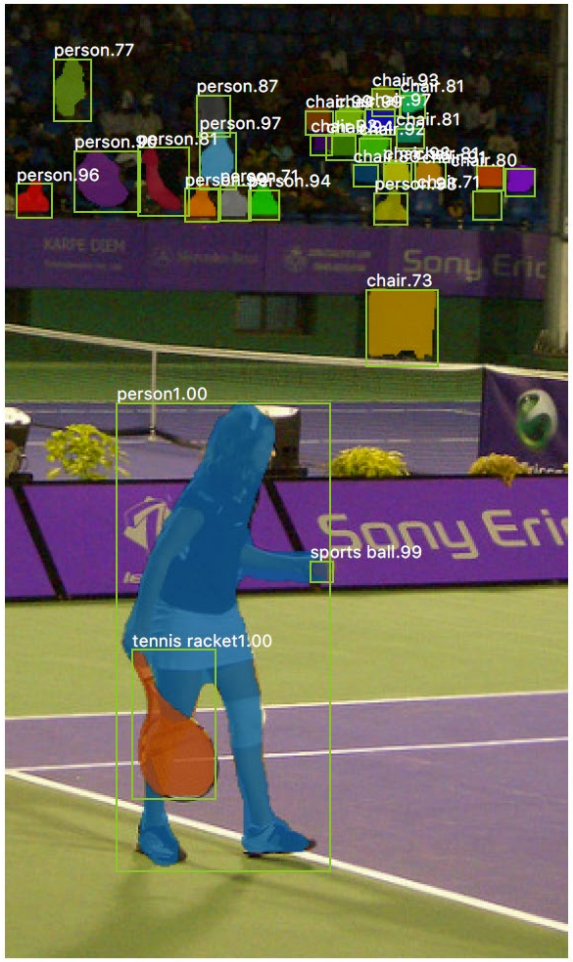
- Example Training Data (requires pixel-level labels)



Mask R-CNN

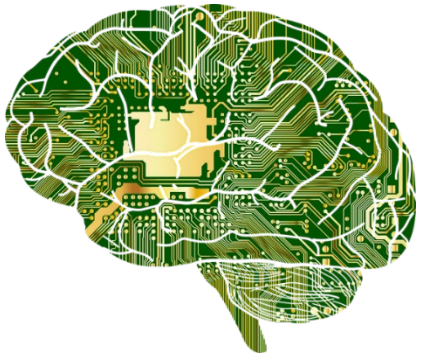
- Very good results!
  - running at 5fps though

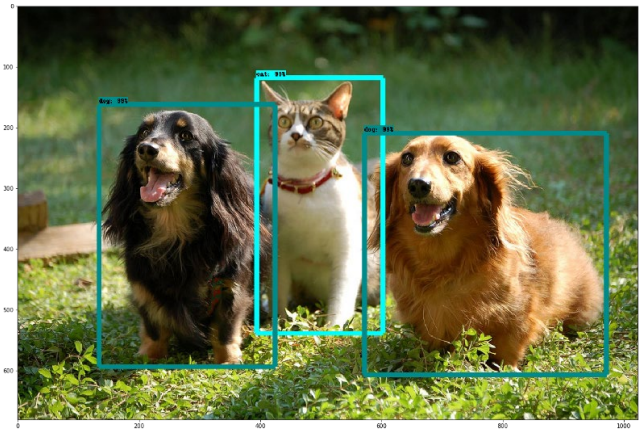


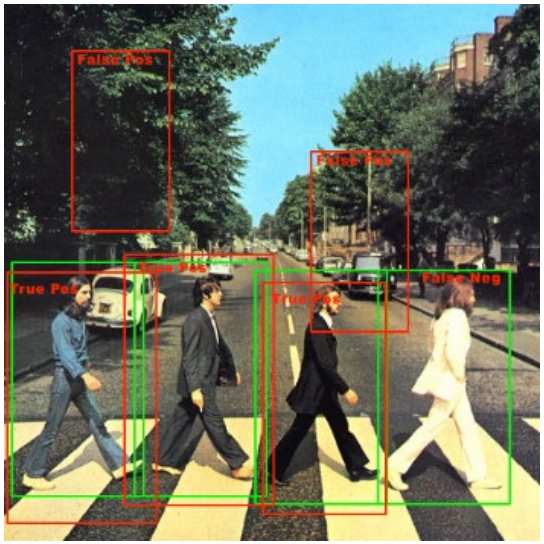


What to Covered Today…

- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection

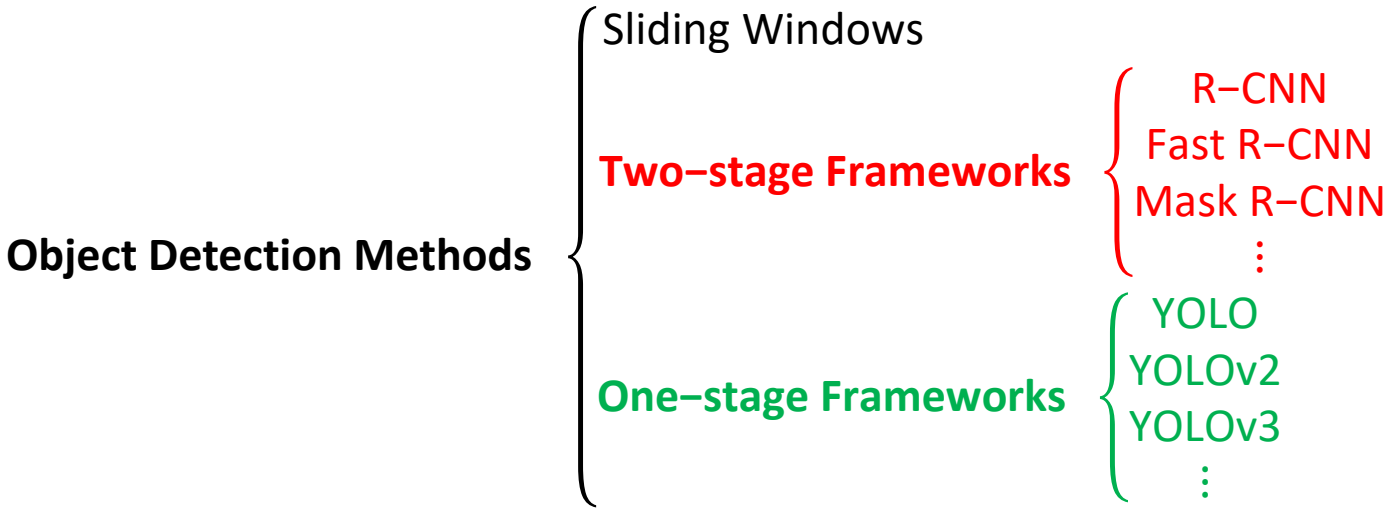






Recap

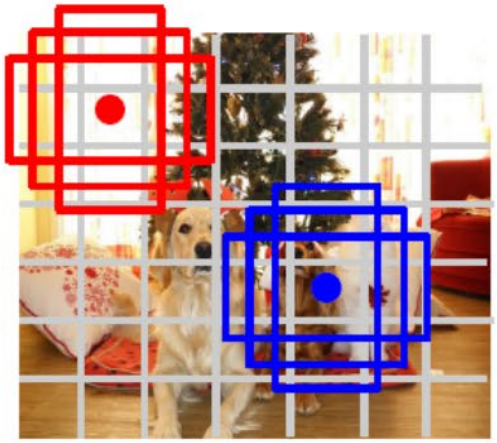
- So far, the introduced methods follow a **two-stage** framework.
  1. Region Proposal
  2. Per-Region Classification/Regression
- Can we make it faster by integrating the above two steps into one single network?



One-Stage Object Detection: Detection without Proposals

Go from input image to tensor of scores with one big convolutional network!

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

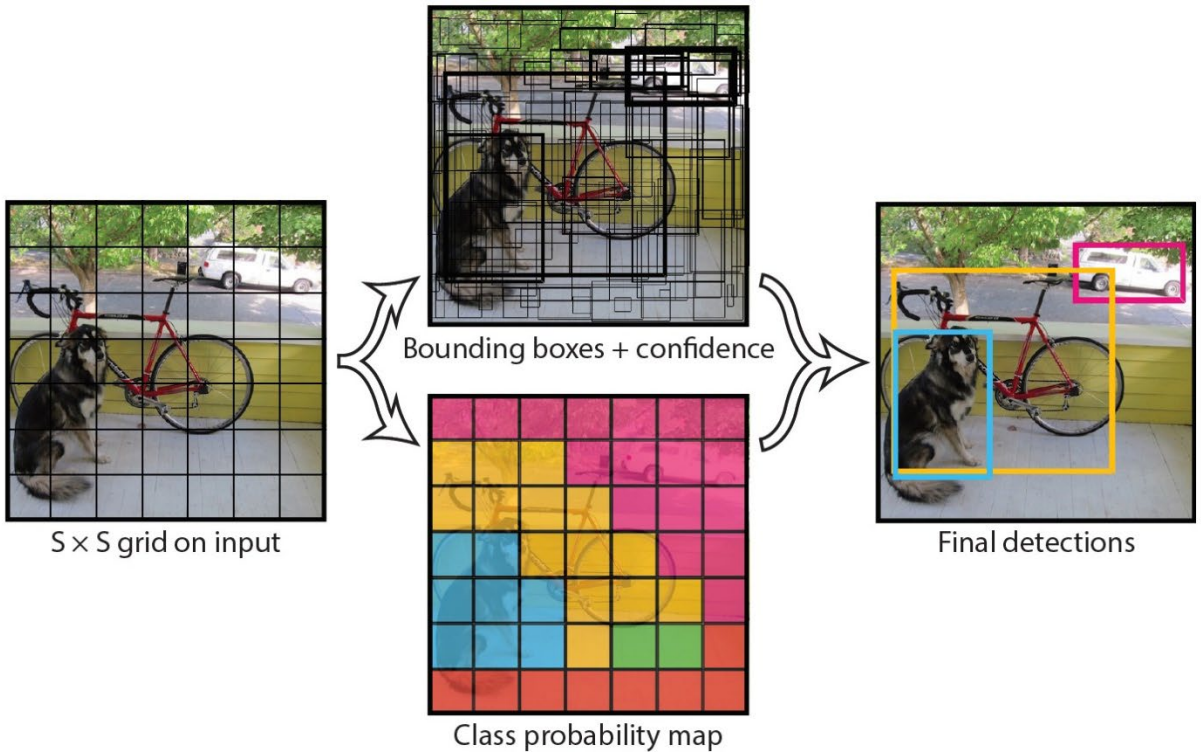
Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

You Only Look Once (YOLO)

Divide the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities.

These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.



You Only Look Once (YOLO)

class confidence score = box confidence score × conditional class probability

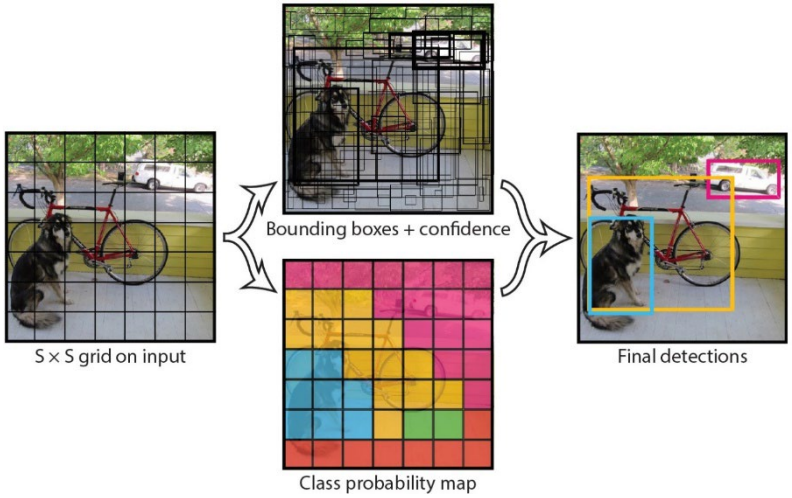
$$\begin{aligned}
\text{box confidence score} &\equiv P_r(object) \cdot IoU \\
\text{conditional class probability} &\equiv P_r(class_i|object) \\
\text{class confidence score} &\equiv P_r(class_i) \cdot IoU \\
&= \text{box confidence score} \times \text{conditional class probability}
\end{aligned}$$

where

$P_r(object)$ is the probability the box contains an object.
IoU is the IoU (intersection over union) between the predicted box and the ground truth.
$P_r(class_i|object)$ is the probability the object belongs to $class_i$ given an object is presence.
$P_r(class_i)$ is the probability the object belongs to $class_i$

You Only Look Once (YOLO)

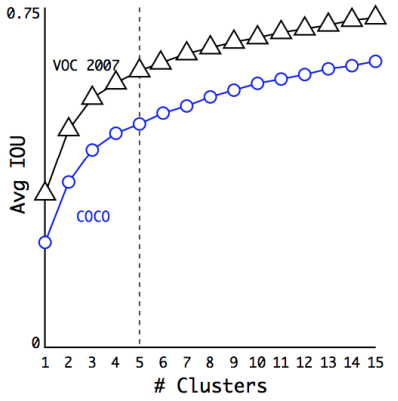
- **Fast**.
  Good for real-time processing.
- **End-to-end learnable**.
  Predictions (object locations and classes) are made from one **single network**.
- **Access to the entire image**.
  Region proposal methods limit the classifier to the specific region.
  YOLO **accesses to the whole image** in predicting boundaries.
  With additional context, result in fewer false positives in background areas.
- **Spatial diversity**.
  Detect one object per grid cell. It enforces spatial diversity in making predictions.

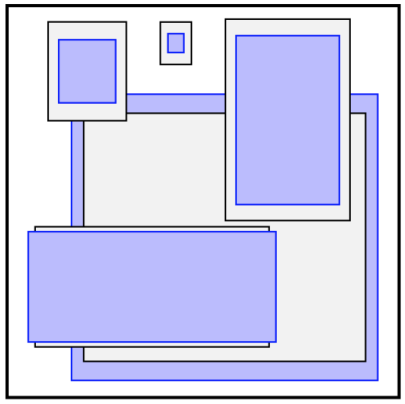


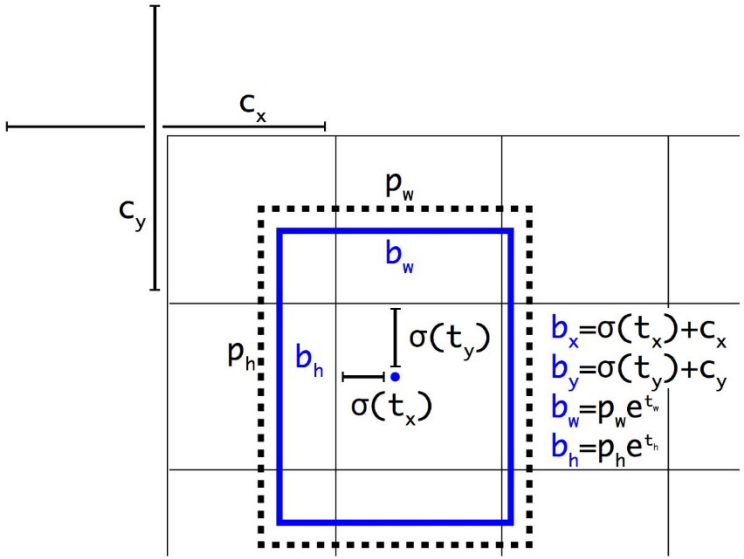
YOLOv2

- **Predetermined bounding box shape** (*anchor boxes*)
  Guesses that are common for real-life objects using *k*-means clustering
  $\Longrightarrow$ Predicts **offsets** rather than bounding boxes themselves
- **Move the class prediction from the cell level to the boundary box level**
  Each *bounding box* (instead of each *cell*) produces a class prediction

$$S \times S \times (B * 5 + C) \Longrightarrow S \times S \times (B * (5 + C))$$



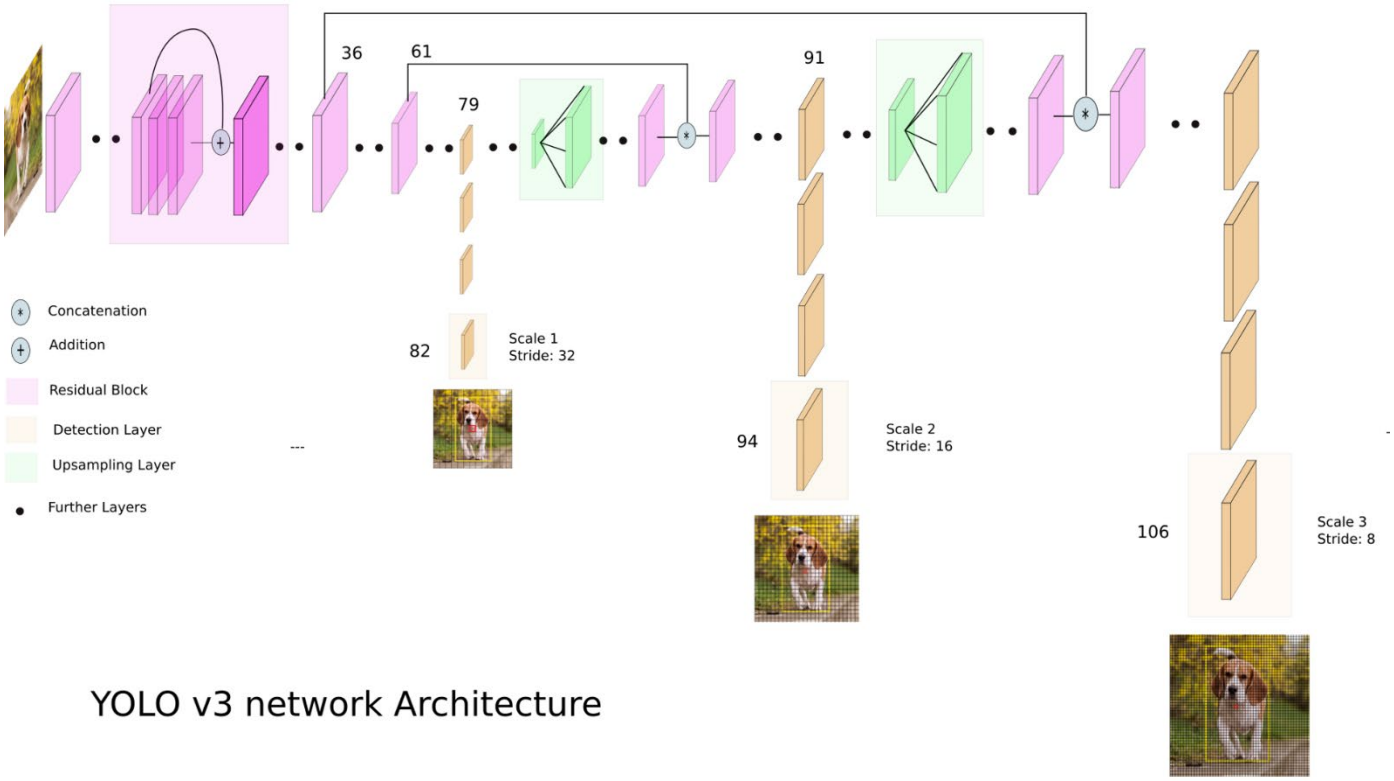




YOLOv3

- **Feature Pyramid Networks** (FPN) **like Feature Pyramid**
  YOLOv3 makes predictions at 3 different scales (similar to the FPN).

$$S \times S \times (3 * (5 + C))$$



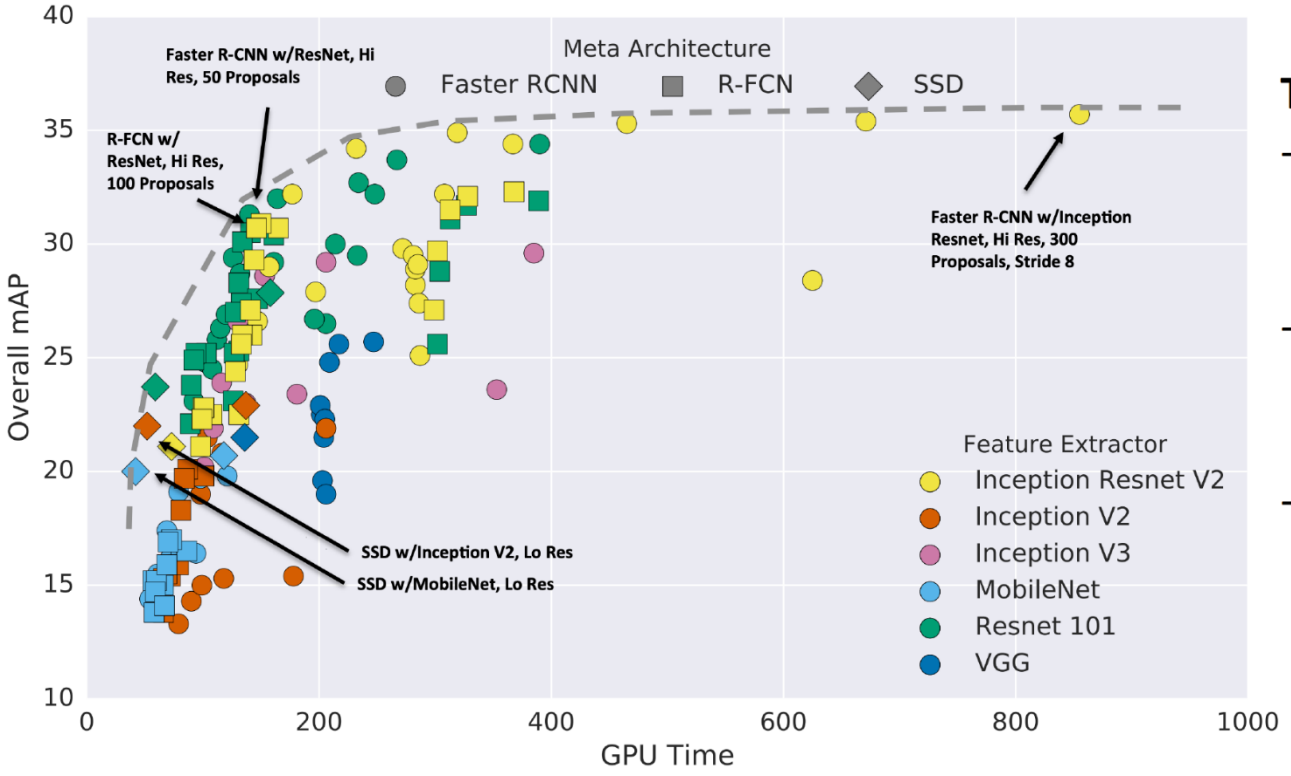
YOLO v3 network Architecture

Recap

Object Detection Methods

- Sliding Windows
- **Two−stage Frameworks** (High Accuracy, Slow)
  - R−CNN
  - Fast R−CNN
  - Mask R−CNN
  - ⋮
- **One−stage Frameworks** (Good Accuracy, Very Fast)
  - YOLO
  - YOLOv2
  - YOLOv3
  - ⋮

Remarks

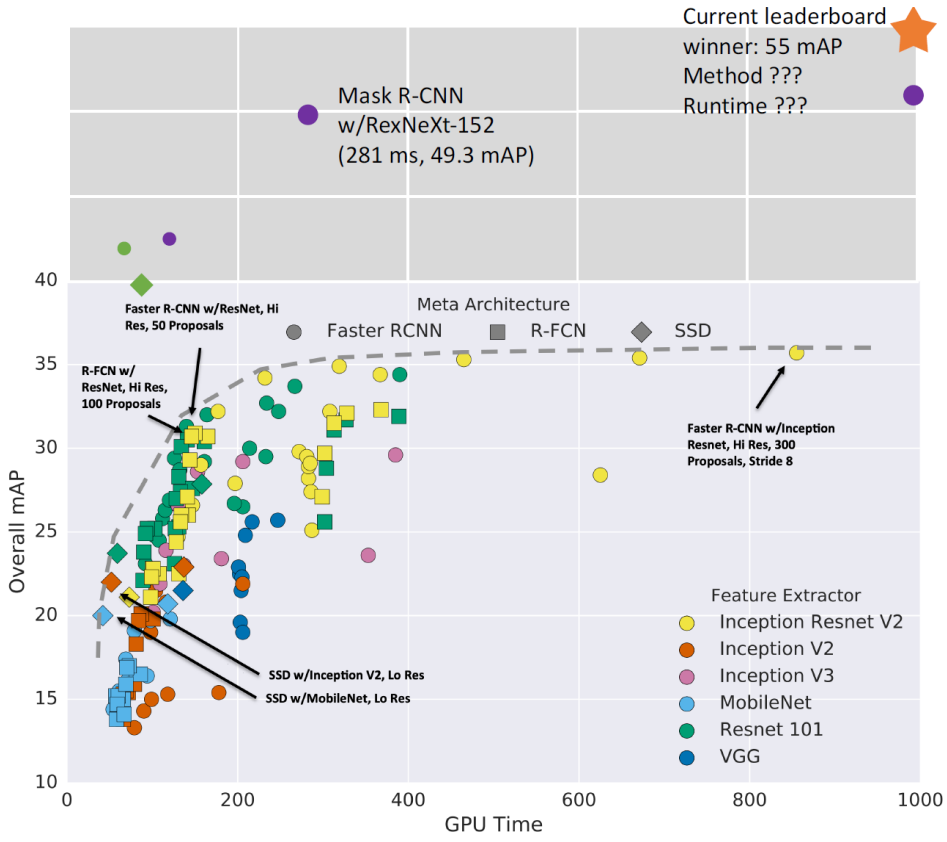


Takeaways:

- Two stage method (Faster R-CNN) get the best accuracy, but are slower
- Single-stage methods (SSD) are much faster, but don't perform as well
- Bigger backbones improve performance, but are slower

Slide credit: UMich EECS 498-007

Remarks (cont'd)



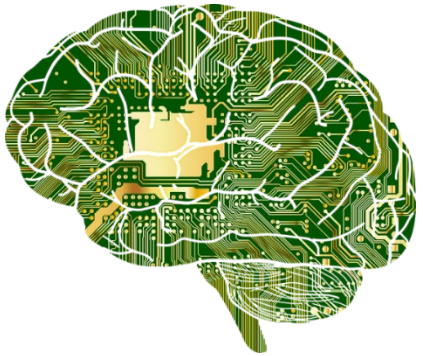
These results are a few years old … since then GPUs have gotten faster, and we've improved performance with many tricks:

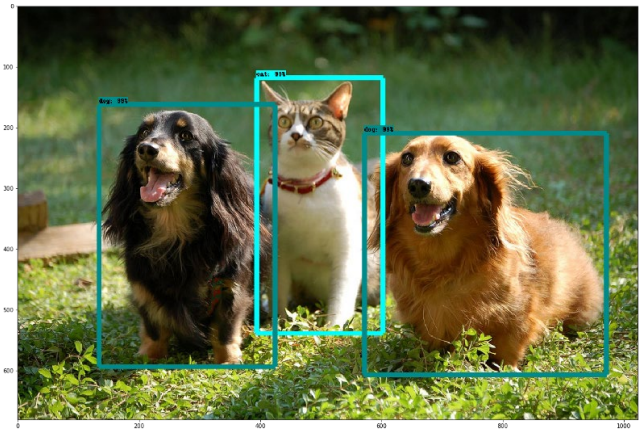
- Train longer!
- Multiscale backbone: Feature Pyramid Networks
- Better backbone: ResNeXt
- Single-Stage methods have improved
- Very big models work better
- Test-time augmentation pushes numbers up
- Big ensembles, more data, etc

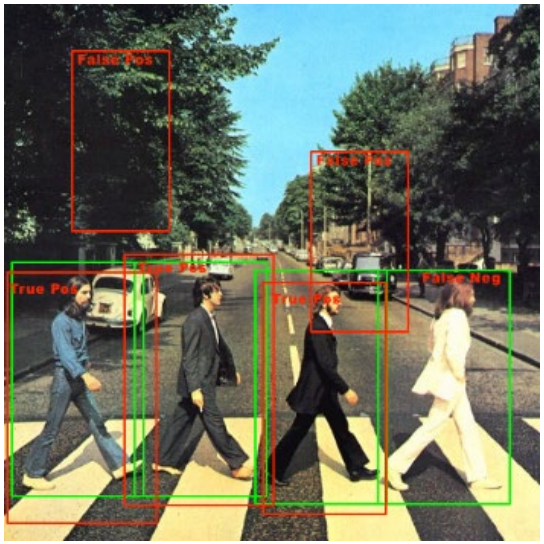
Slide credit: UMich EECS 498-007

What to Covered Today…

- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection

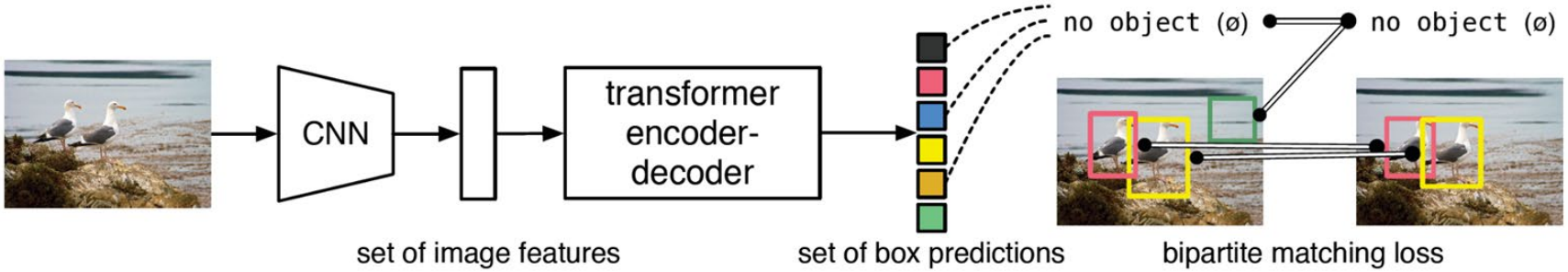






DEtection TRansformer (DETR) (ECCV'20)

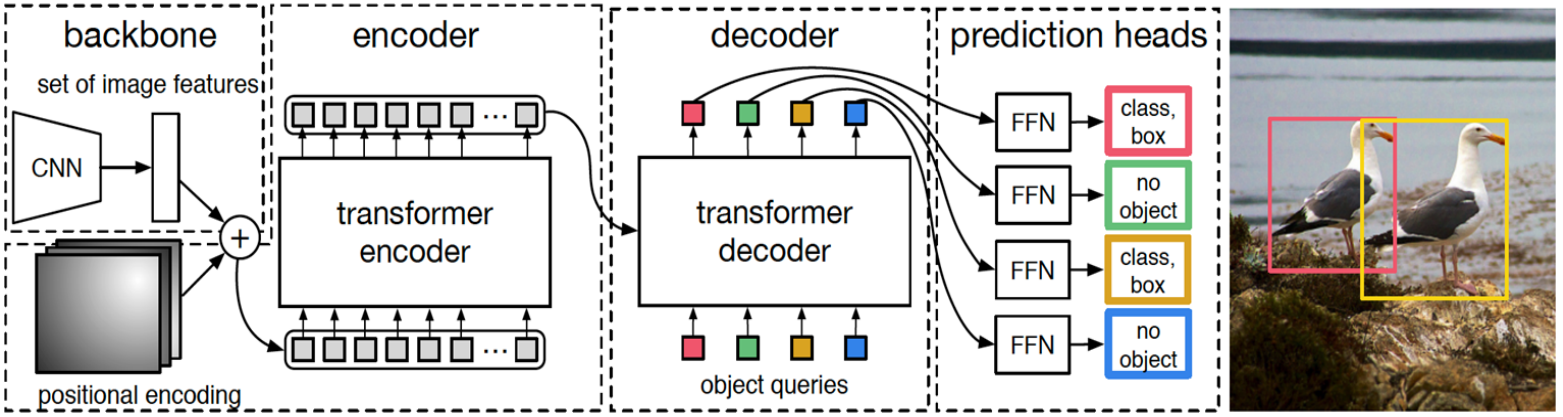
- Proposed by Facebook AI
- Previous methods rely on NMS to matching predictions to ground truth bboxes
  - Designing NMS involves lots of manual tuning
  - NMS can sometimes incorrectly suppress true positive detections
- DETR views object detection as direct set prediction problem => no more NMS



Carion et al., "End-to-End Object Detection with Transformers", ECCV 2020

Framework of DETR

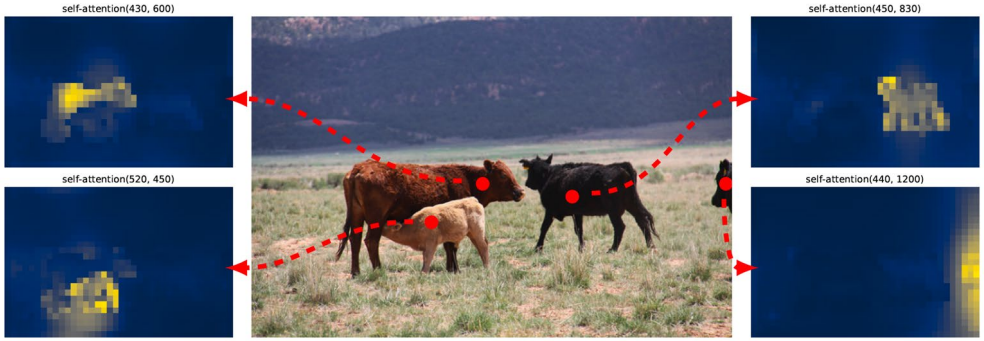
- CNN
  - Feature extraction with CNN
- Transformer
  - Generate N guesses
  - N is set to a relatively high number (~100)
- FFN
  - Predict class logits and bounding box



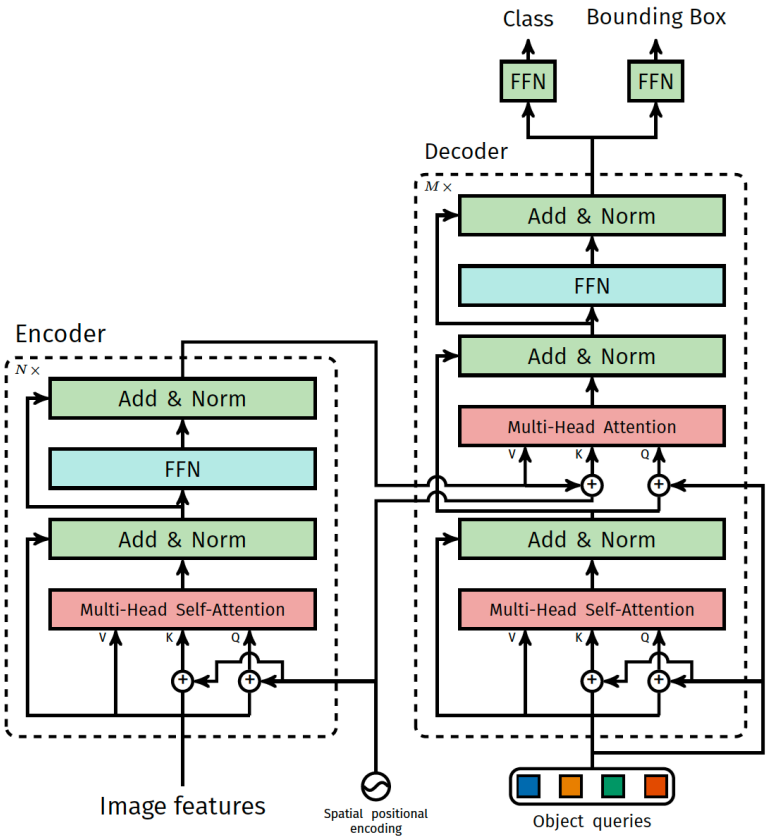
Carion et al., "End-to-End Object Detection with Transformers", ECCV 2020

Framework of DETR (cont'd)

- Transformer captures global semantics and disentangles objects simultaneously
- Decoder inputs: N learnable queries



Attention maps of self-attention layers in encoder



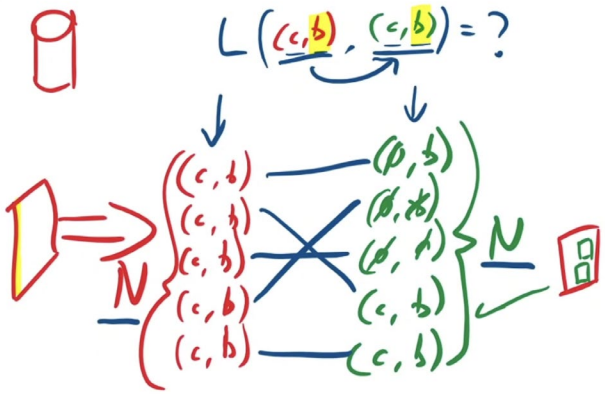
Carion et al., "End-to-End Object Detection with Transformers", ECCV 2020

Bipartite Matching Loss in DETR

- **Goal:**
  Assign each prediction to a ground truth object (or "no object" ∅)
  - Solve bipartite matching with **Hungarian algorithm**
- **Loss**
  Cross entropy loss + bounding box loss

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

$$\mathcal{L}_{\text{box}}(b_{\sigma(i)}, \hat{b}_i) = \lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) + \lambda_{\text{L1}} ||b_{\sigma(i)} - \hat{b}_i||_1$$



- Plays the same role as NMS
  - Force unique matching between prediction & GT; permutation invariant

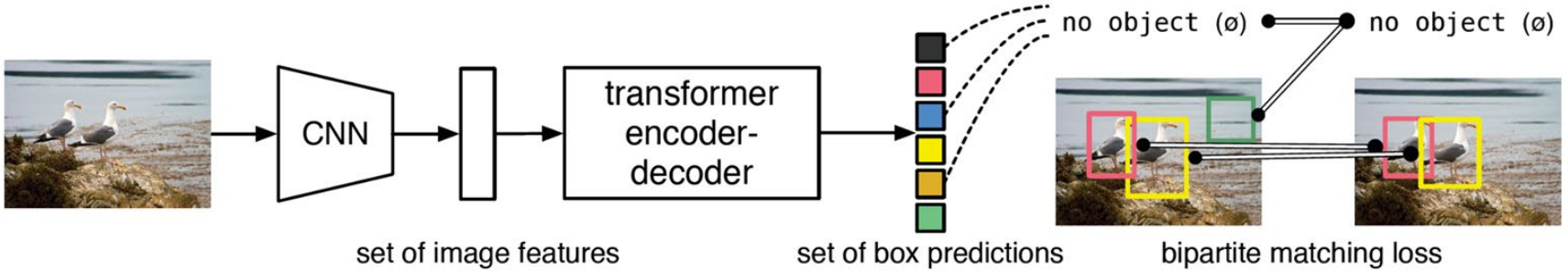
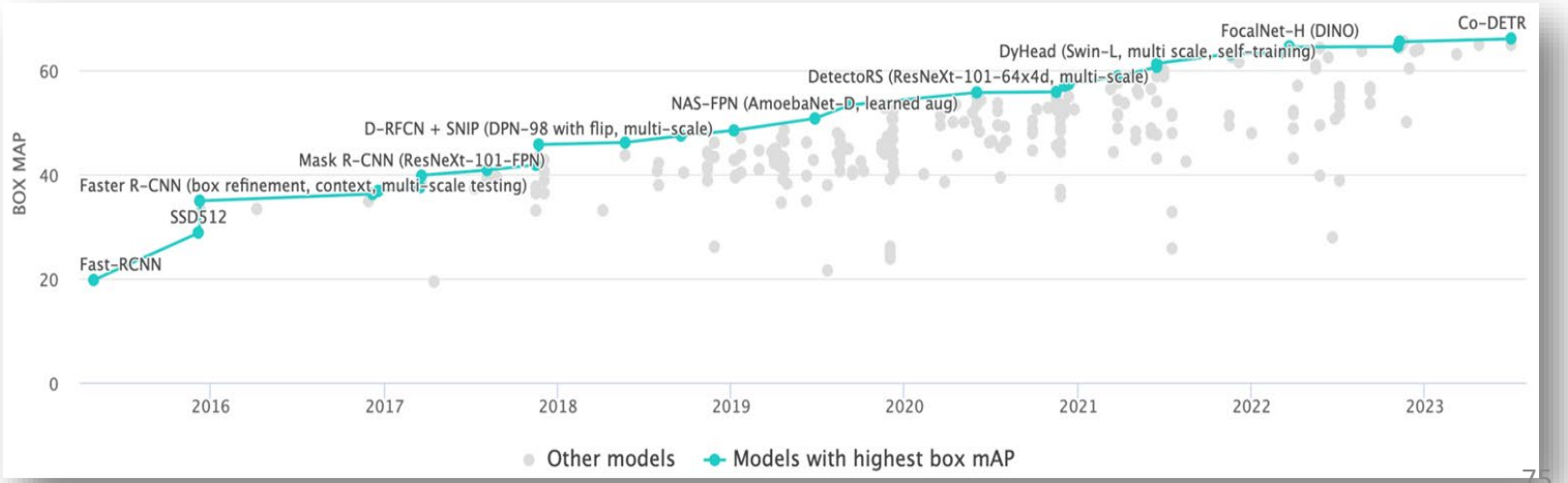


Image credit: "DETR: End-to-End Object Detection with Transformers (Paper Explained)"
https://youtu.be/T35ba_VXkMY?si=3om17kAQzN2q9R_M&t=701

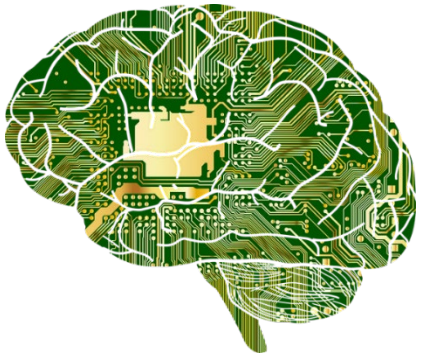
Final Remarks for DETR

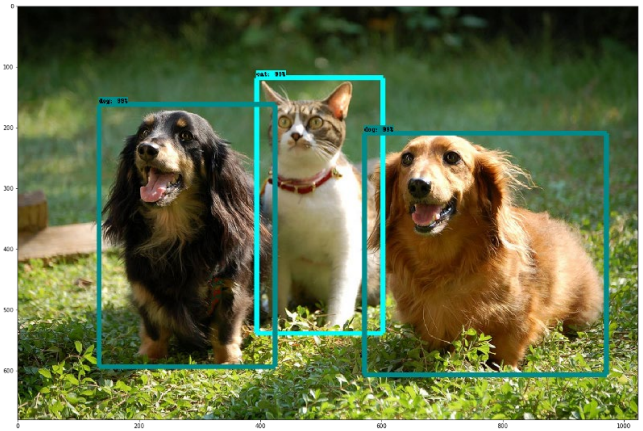
- **Simplicity**
  Simple to implement; less hand-designed components.
- **End-to-End learnable**
  No region proposal; no NMS for post-processing
- **Attends to the entire image**
  Transformer allows the model to capture global context of the image.
- **Current State-of-the-Art**
  Variants of DETR score very well on well-known datasets like MSCOCO

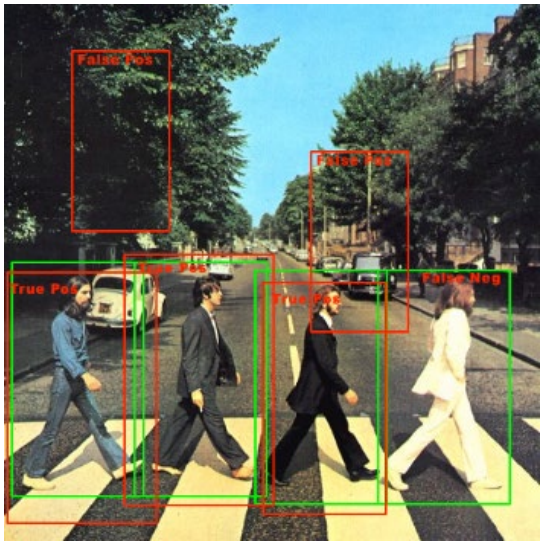


What to Covered Today…

- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection & Grounding

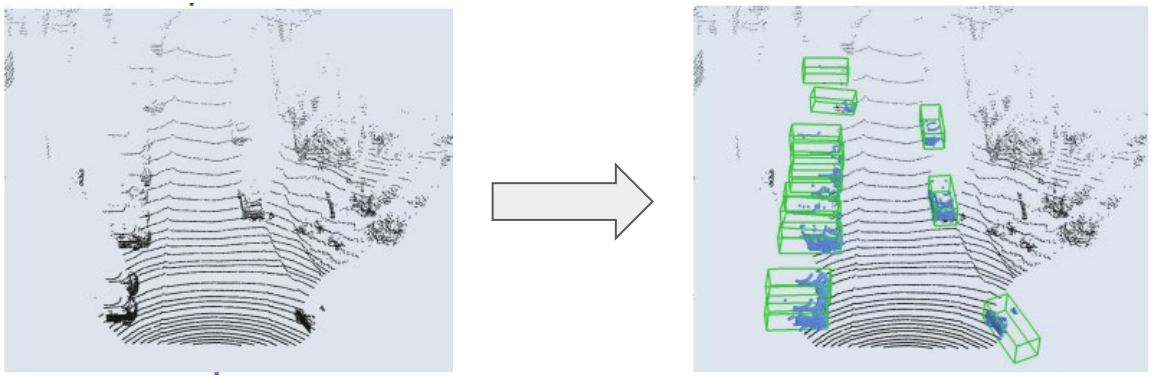






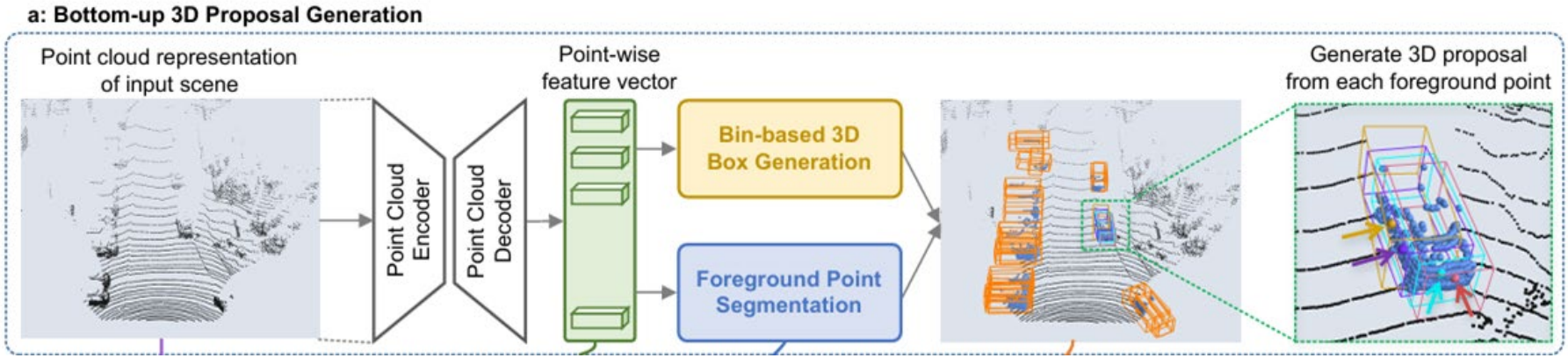
Intro of 3D Detection

- Produce 3D bounding boxes with the following inputs:
  - 3D point clouds
  - 2D images



Supervised 3D Detection

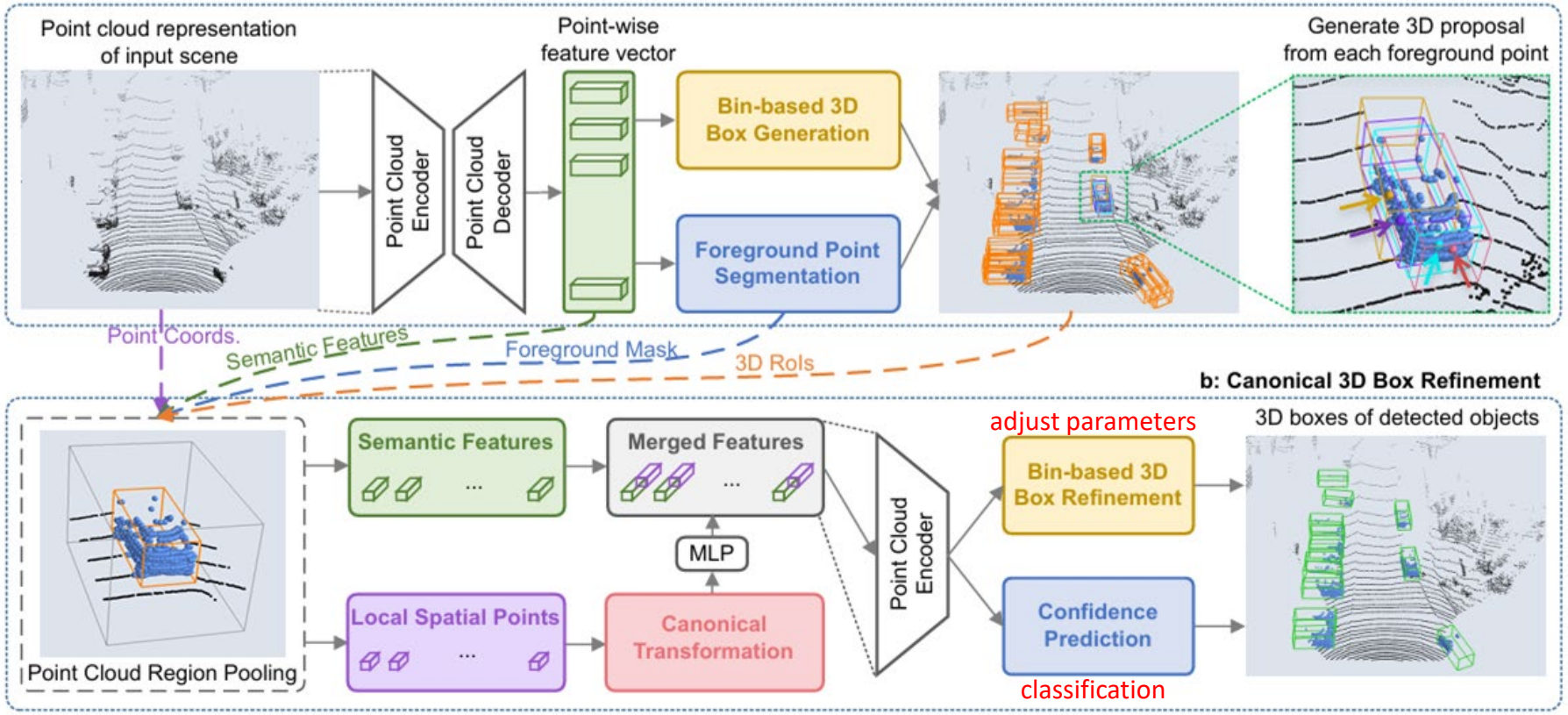
- PointRCNN (CVPR 2019)
  - segment foreground points
  - generate 3D bounding box proposals for each foreground point (i.e., center, box size, & orientation)



PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud

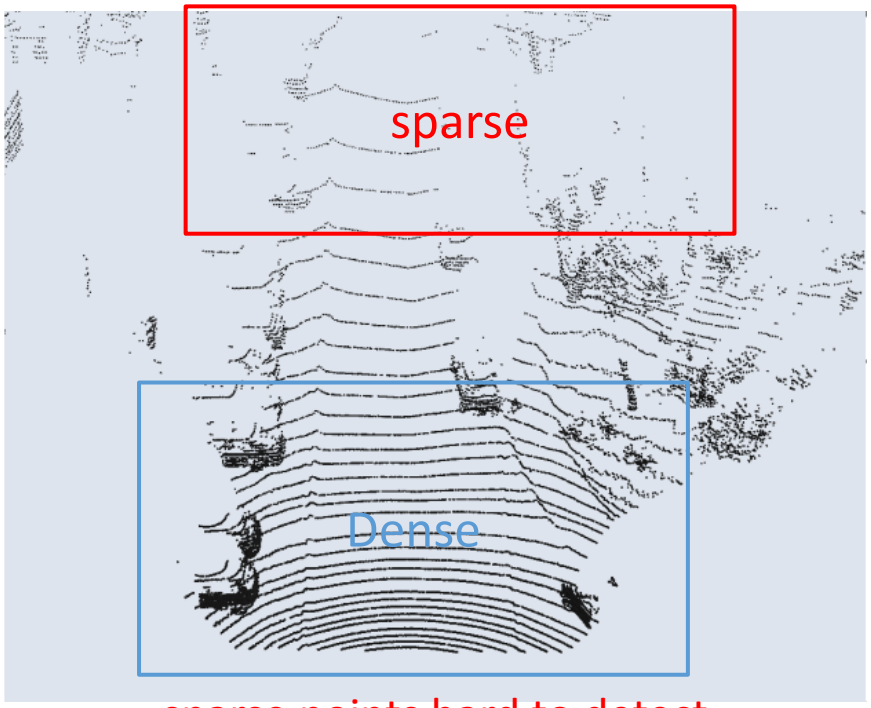
Supervised 3D Detection (cont'd)

- For each bounding box and points inside
  - transform point coordinate by treating *bounding box center* as origin
  - concate point **coordinate** (low-level) and **feature** (high-level) for all points and encode a global feature for each box
  - output classification score (confidence) for each box & preserve high-score box as final prediction

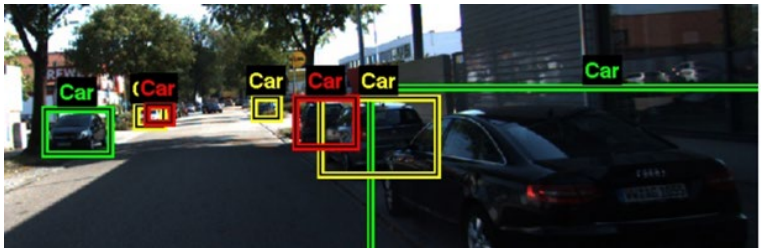


PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud

- Problem of PointRCNN:
  - Data collection -> need to reduce the need of labeled 3D bounding box
  - Points far from camera are too sparse -> need other information to help



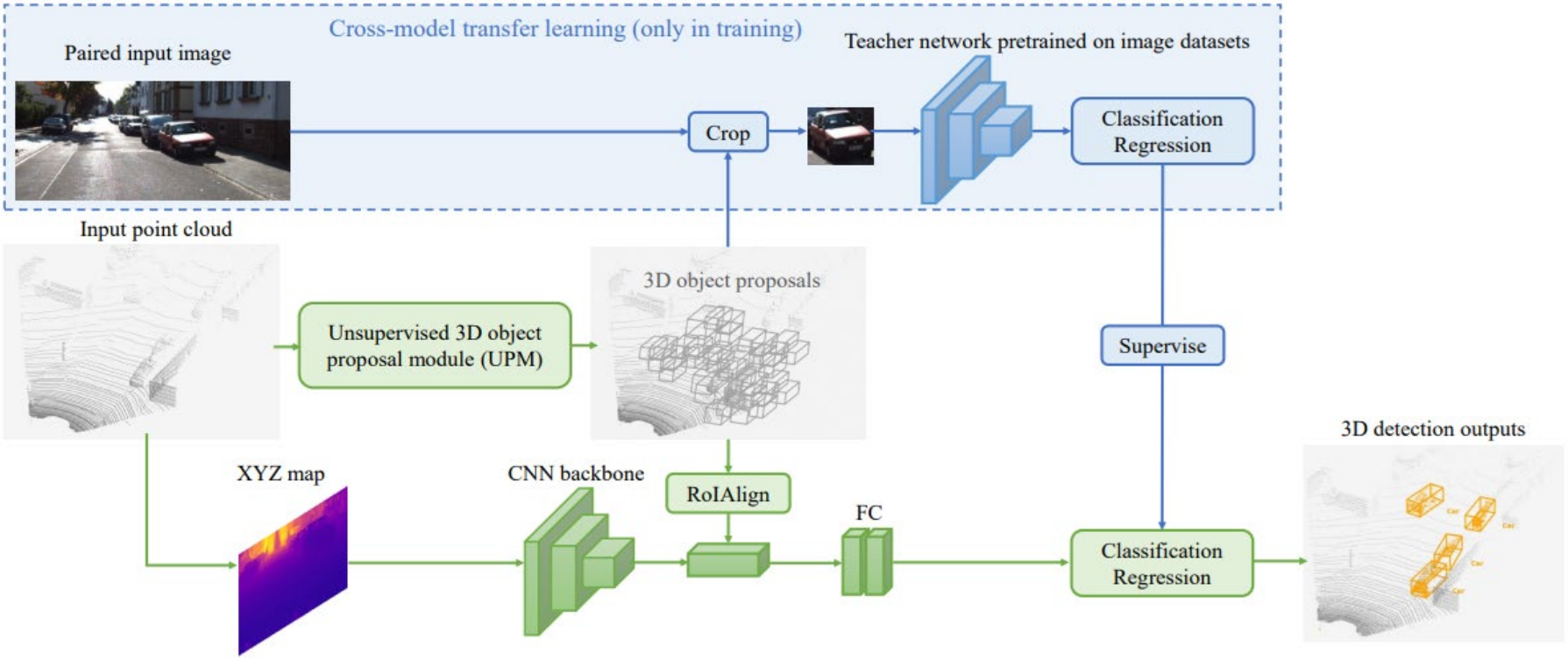
sparse points hard to detect



Can we leverage 2D boxes?

Weakly-Supervised 3D Detection

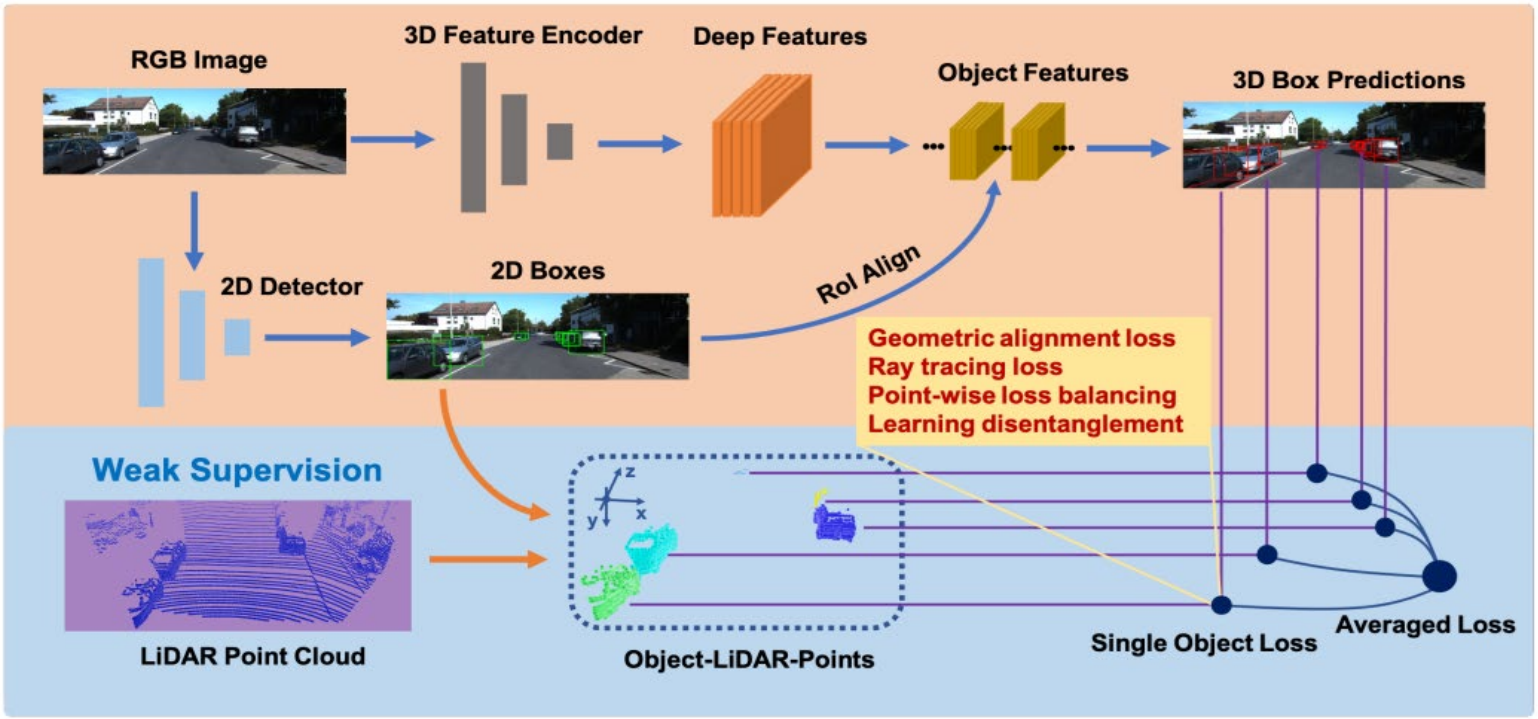
- VS3D (ACM MM 2021):
  - Use pre-trained 2D classifier and detector to guide 3D prediction
  - 2D projection of 3D bounding box should perfectly match a 2D bounding box



Weakly Supervised 3D Object Detection from Point Clouds

Weakly-Supervised 3D Detection

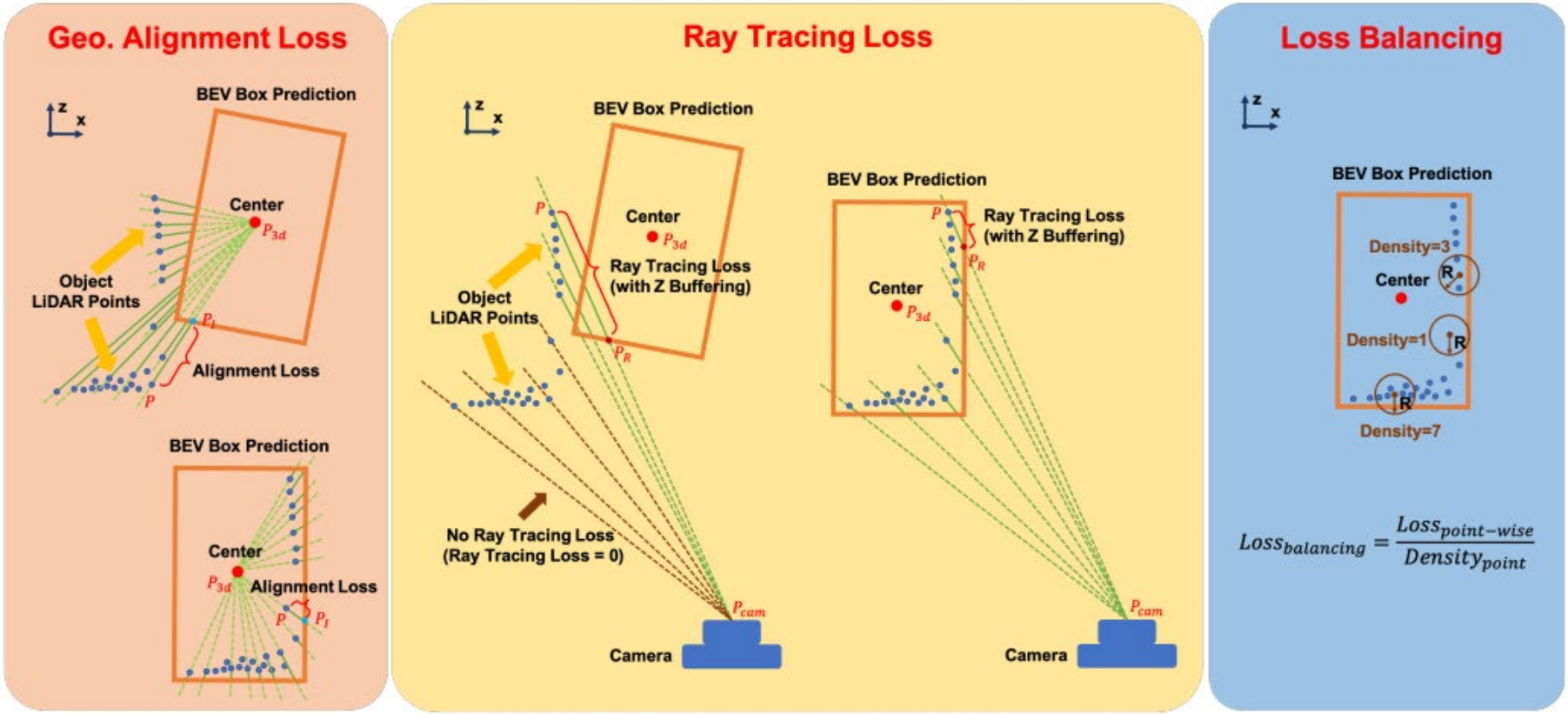
- WeakM3D (ICLR 2022):
  - 2D image + 3D point cloud as inputs
  - Also use 2D detector as strong prior
  - Introduce many loss for regularization



WeakM3D: Towards Weakly Supervised Monocular 3D Object Detection

Weakly-Supervised 3D Detection (cont'd)

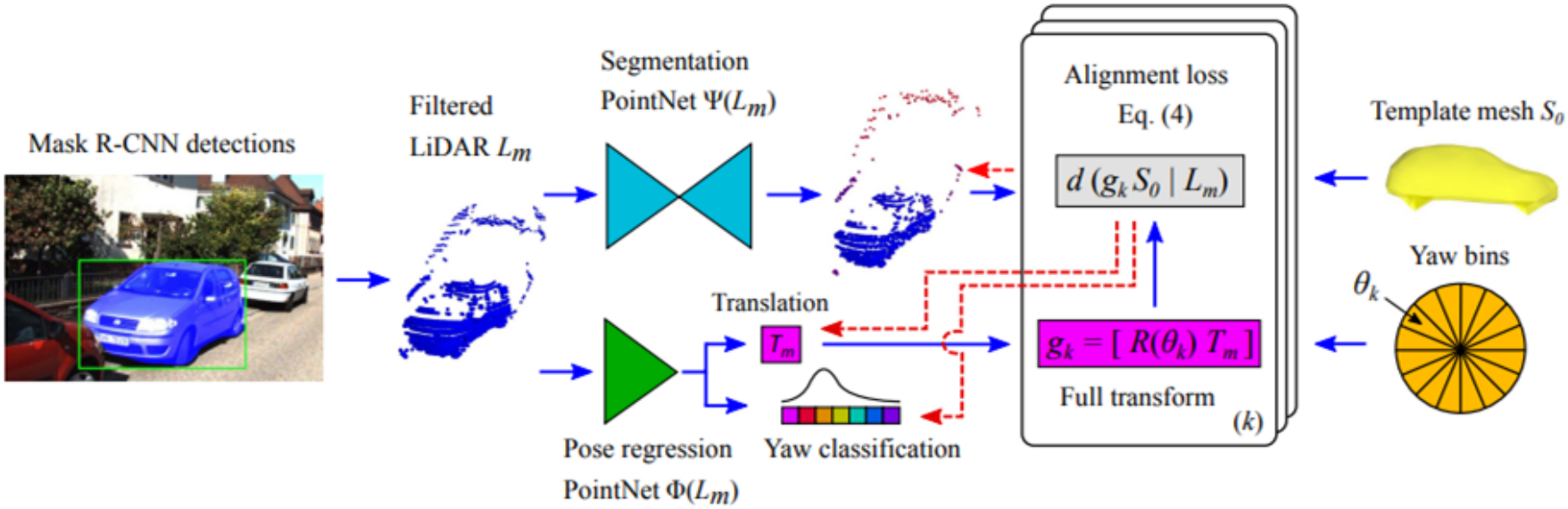
- WeakM3D (ICLR 2022) (cont'd):
  - Alignment loss ensure most points align to ***at most 2 sides*** of predicted bbox
  - Ray tracing loss ensure point-camera ray panetrates through bbox ***without*** touching any others
  - Loss balancing ensure that the sparse regions not neglected



WeakM3D: Towards Weakly Supervised Monocular 3D Object Detection

Weakly-Supervised 3D Detection (cont'd)

- McCraith et.al (ICRA 2022):
  - Use a human-designed mesh of car to find matching points in 3D space
  - Treat the bounding box of the aligned mesh as final predcition

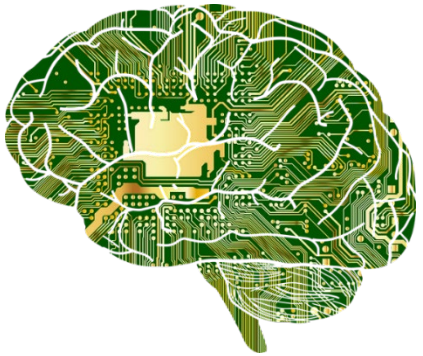


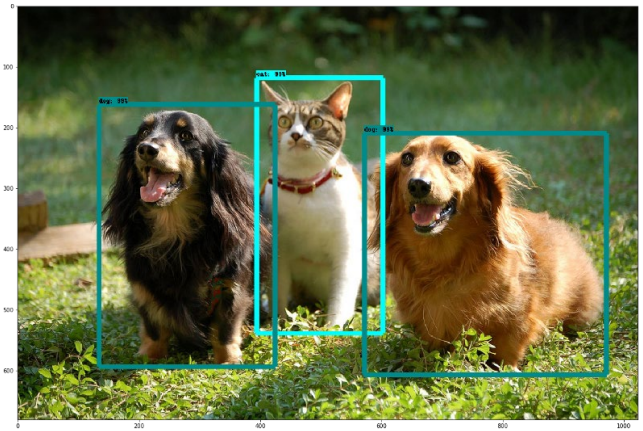
More references for 3D detection

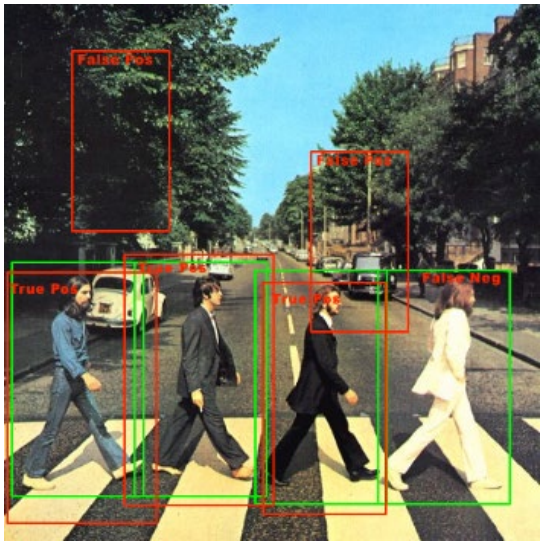
- Supervised
  - PatchNet: A Simple Face Anti-Spoofing Framework via Fine-Grained Patch Recognition (CVPR 2022)
  - PointPillars: Fast Encoders for Object Detection from Point Clouds (CVPR 2019)
  - Deep Hough Voting for 3D Object Detection in Point Clouds (ICCV 2019)
- Weakly-supervised
  - FGR: Frustum-Aware Geometric Reasoning for Weakly Supervised 3D Vehicle Detection (ICRA 2021)
  - Weakly Supervised 3D Object Detection from Lidar Point Cloud (ECCV 2020)
  - Weakly Supervised Monocular 3D Object Detection Using Multi-View Projection and Direction Consistency (CVPR 2023)
  - Weakly Supervised Monocular 3D Detection with a Single-View Image (CVPR 2024)

What to Covered Today…

- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection & Grounding

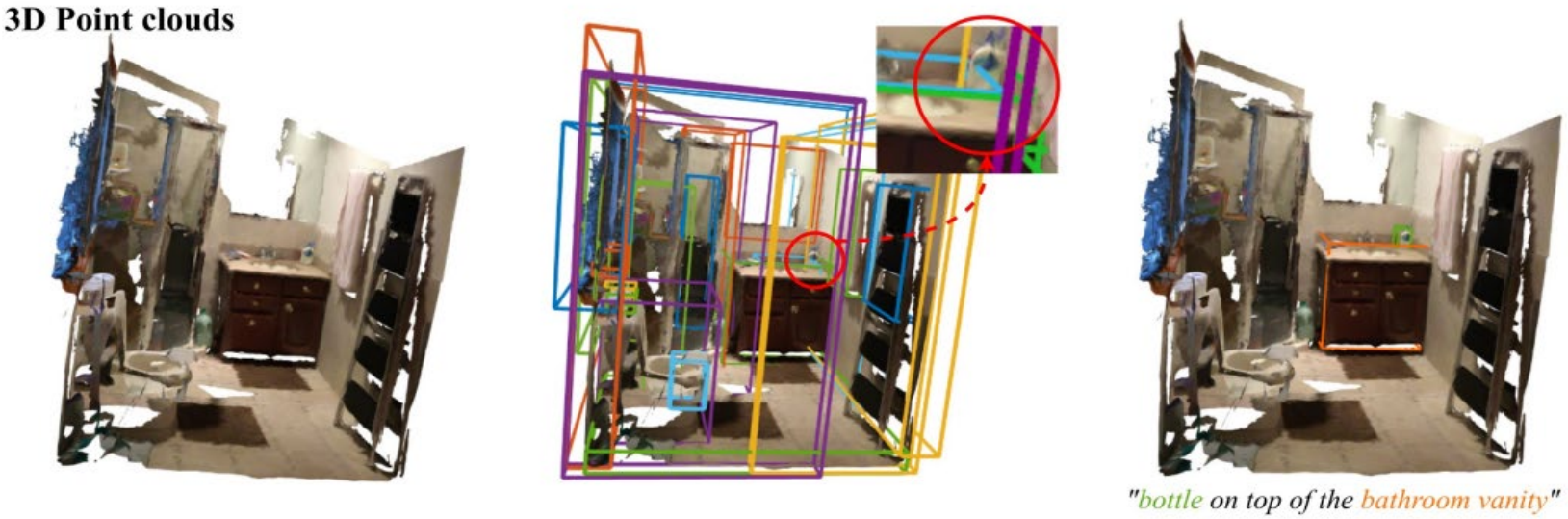






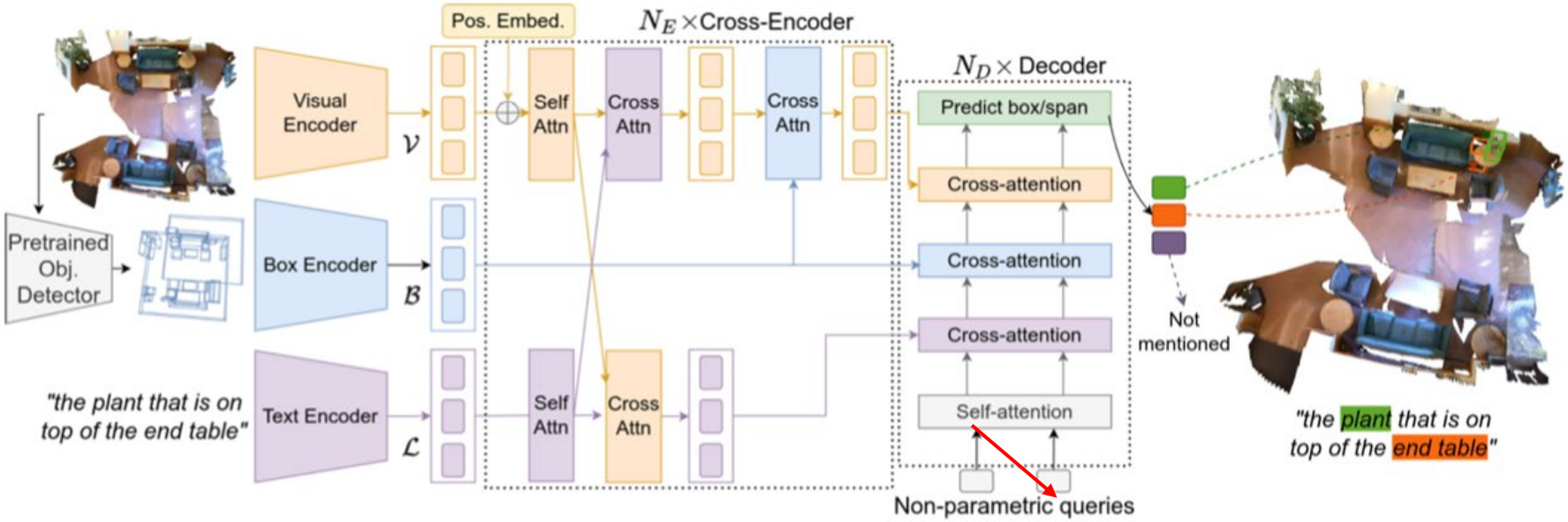
Intro of 3D Visual Grounding (3DVG)

- Find target object with text prompt as query
- Input prompt contains a target object and anchor objects
  - Target: Object that the query text want to find (only one)
  - Anchor: Objects that are mentioned in the text whichhave spatial relation with the target (can have many of them)
- In general 3VG setting:
  - 3D bbox proposal are given (directly provided by the dataset)
  - Only target object has ground-truth label (even for FS settings)



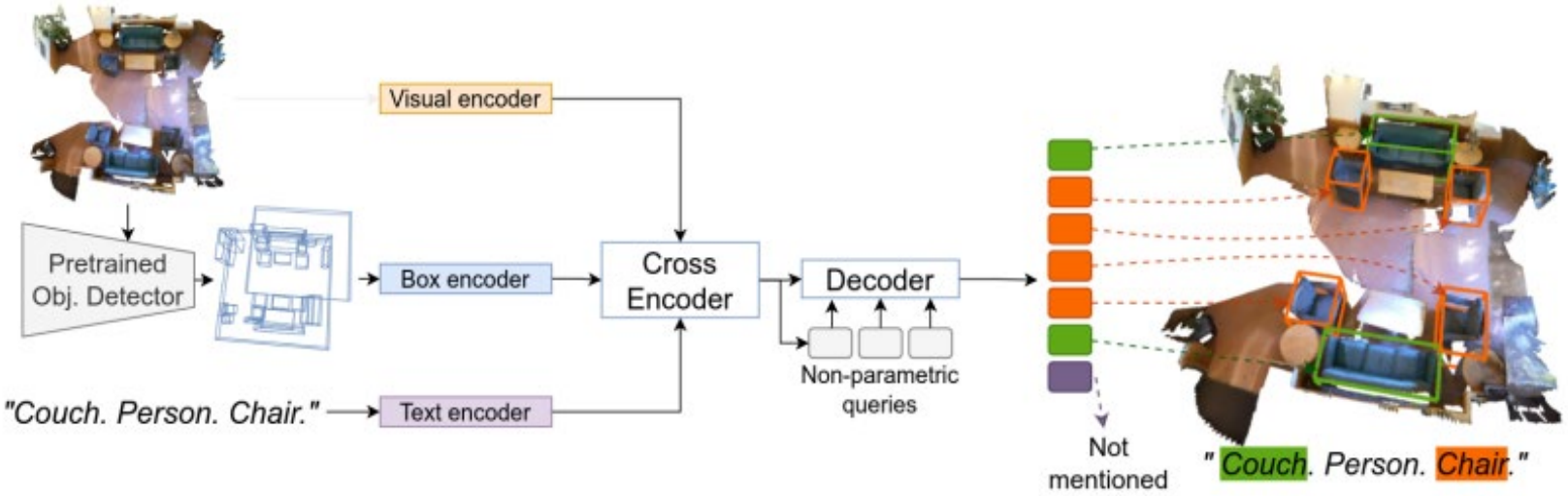
Fully-supervised 3DVG

- **BUTD-DETR (ECCV 2022)**
  - Use colored point cloud as input
  - obtain object proposal (and class pred.) from pre-trained 3D detector
  - Transfomer-based decoder to cross-attend visual-textual features
    - Non-parametric queries: treat each proposal as a token (global feature of each proposal as input)



encode points inside each proposal as input token feature

- BUTD-DETR (ECCV 2022) (cont'd)
  - Synthesize query texts for pre-training
    - direct combine random object classes
    - treat the last mentioned object as target, others as anchors

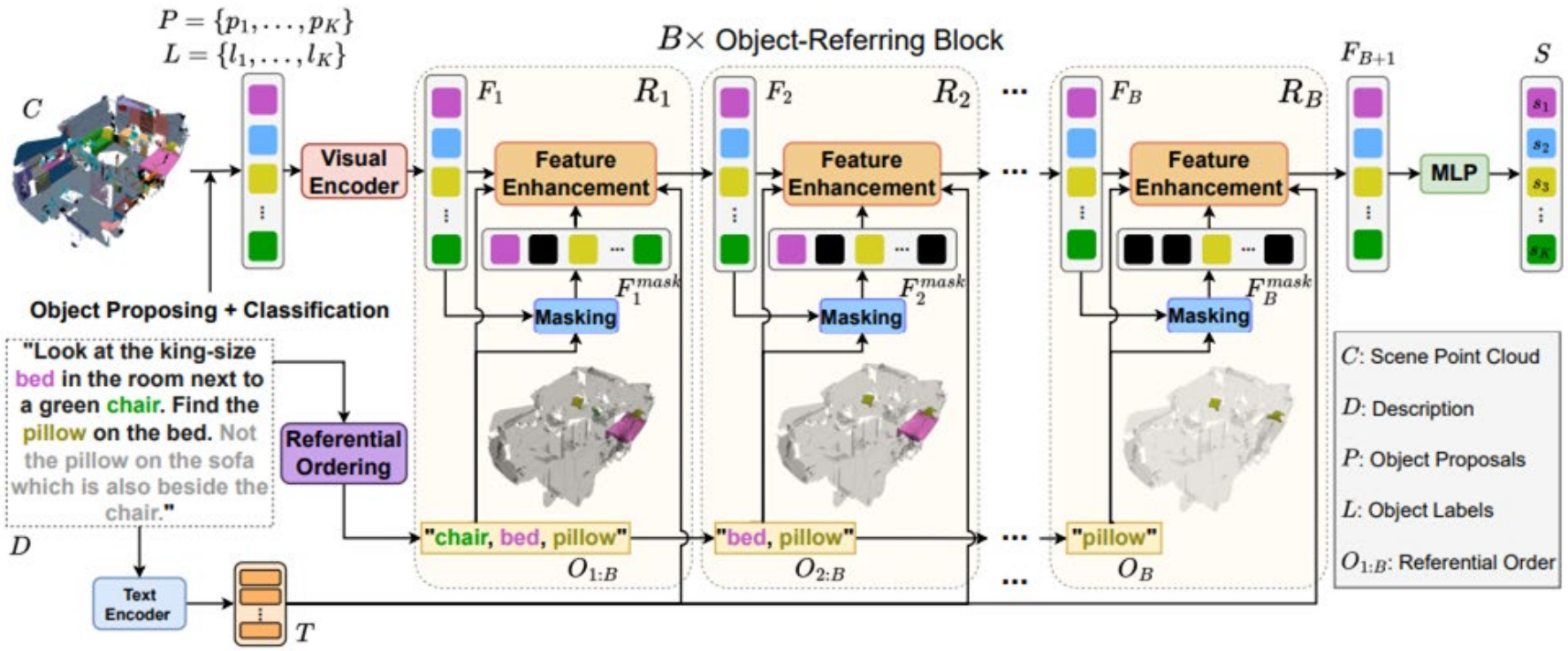


Data-Efficient 3DVG

- Data-Efficient 3D **Vi**sual **G**rounding via **O**rder-Aware **R**eferring (ViGOR), WACV 2025, VLL@NTU

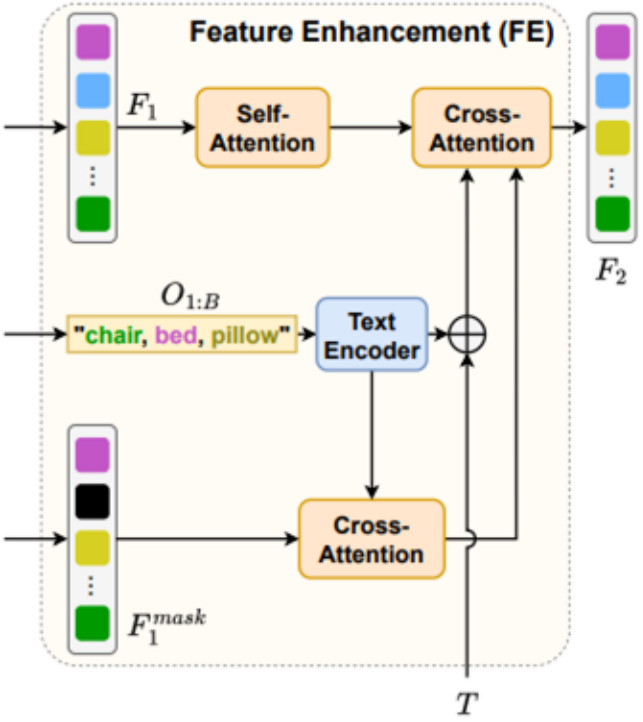
Data-Efficient 3DVG (cont'd)

- ViGOR, WACV 2025
  - Leverage LLM to produce the referring order for 3DVG (anchor -> target)
  - Apply several referring blocks to progressively find the target object
    - Each block contains feature enhancement to focus on anchor/target objects



Data-Efficient 3DVG (cont'd)

- Feature enhancement in each block
  - Upper branch: pass ***all*** proposal features
  - Middle branch: encode texts of the ***anchor/target objects*** from the referential order
  - Lower branch: mask features of other classes (not belong to anchor or target) and enhance anchor/target feature



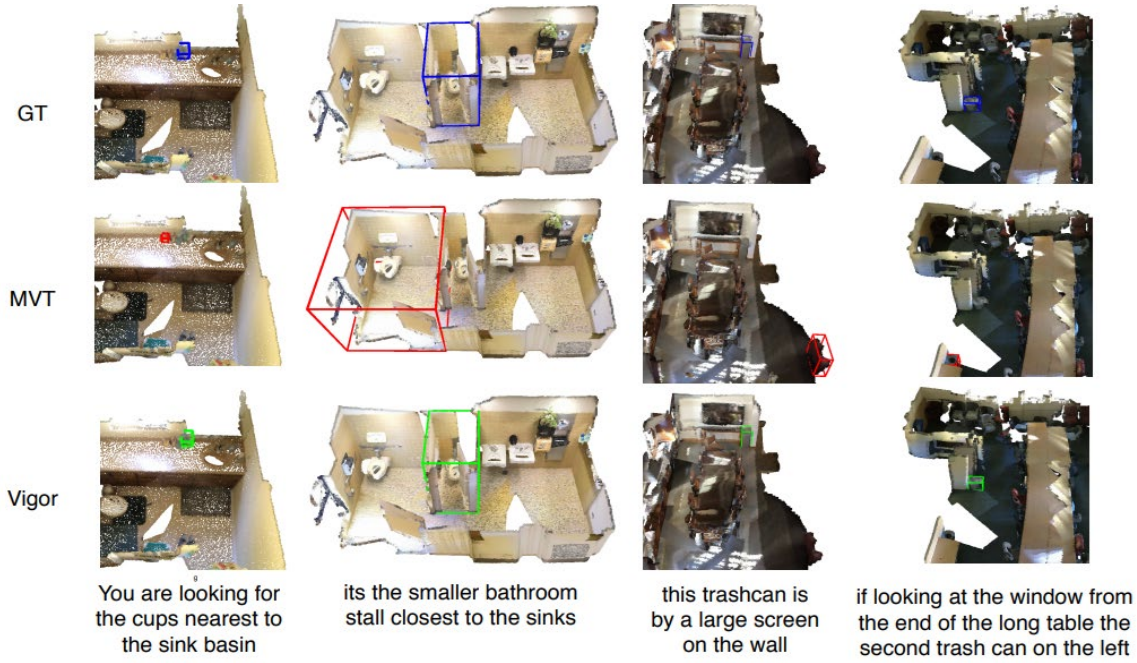
Results

- NR3D dataset
  - contains 707 indoor point cloud scenes from ScanNet
  - 28715/7485 descriptions in the training/testing set, each has a target object label
- Quantitative evaluation
  - Only train for 1~10 % of **description-target training pairs**
    - Unlabeled descriptions (and their corresponded scenes) in training set are ***not*** used
  - Our 1% better than many methods with 10% data

Method	Labeled Training Data			
	1%	2.5%	5%	10%
Referit3D [2]	4.4	13.6	20.3	23.3
TransRefer3D [17]	11.0	16.1	21.9	25.7
SAT [46]	11.6	16.0	21.4	25.0
BUTD-DETR [21]	24.2	28.6	31.2	33.3
MVT [20]	9.9	16.1	21.6	26.5
MVT + CoT3DRef [5]	9.4	17.3	26.5	38.2
ViL3DRel + CoT3DRef [5]	22.38	27.25	33.77	38.4
Vigor (Ours)	**33.5**	**36.1**	**41.5**	**46.0**

More references of 3DVG

- Traditional
  - Multi-View Transformer for 3D Visual Grounding (CVPR 2022)
  - Language Conditioned Spatial Relation Reasoning for 3D Object Grounding (NeurIPS 2022)
- Data-efficient
  - NS3D: Neuro-Symbolic Grounding of 3D Objects and Relations (CVPR)
  - CoT3DRef: Chain-of-Thoughts Data-Efficient 3D Visual Grounding (ICLR 2024)



What We've Covered Today…

- **Object Detection**
  - Detection via Sliding Windows
  - Two-Stage vs. Single-Stage Detectors
  - Transformer-based Detectors
  - 3D Detection & Grounding