Week 10
- Object Detection

2019.07.20

Solaris

(http://solarisailab.com)

Week 9 복습 – Semantic Image segmentation

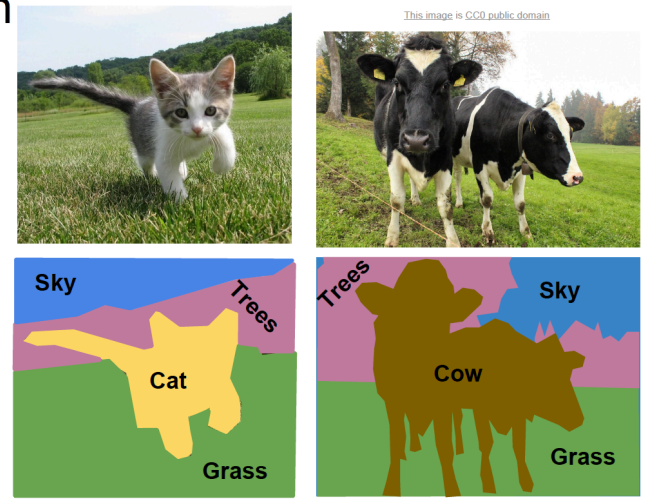
- **Sematic Image Segmentation** : 전체 이미지 pixel에서 의미있는 부분끼리 묶어서 prediction하는 기법 (Dense prediction)

Input Segmentation [9]

Semantic Segmentation

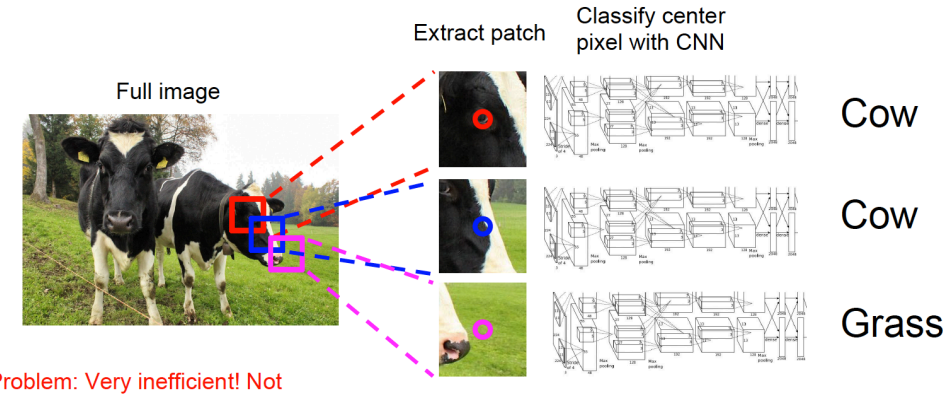
Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



Week 9 복습 – Semantic Image segmentation

Semantic Segmentation Idea: Sliding Window

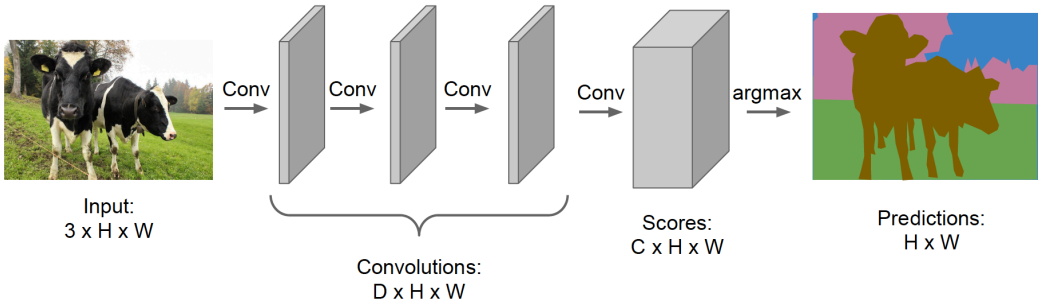


Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

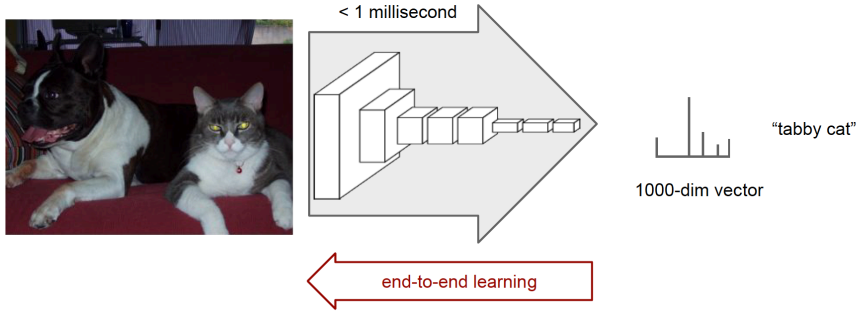
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

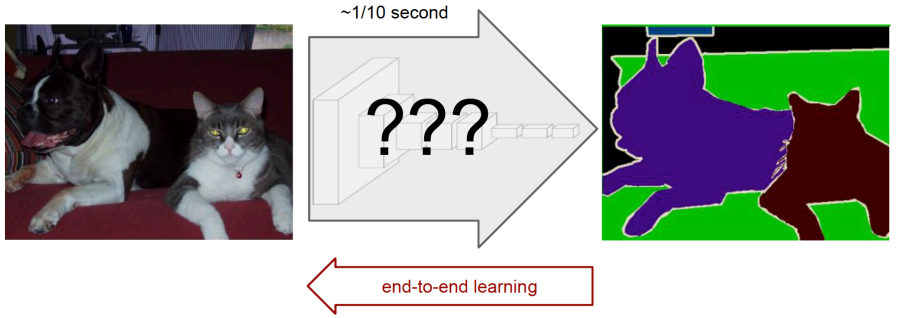


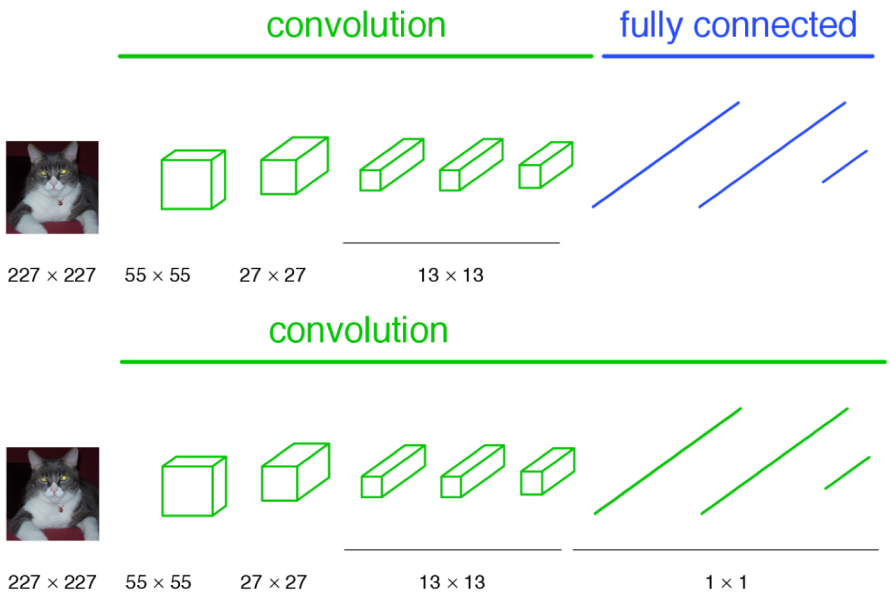
Week 9 복습 – Fully Convolutional Networks for Semantic Segmentation(FCN)

convnets perform classification



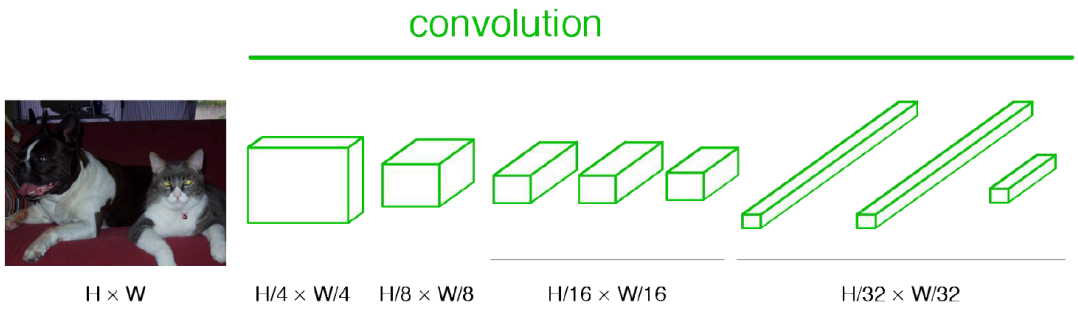
lots of pixels, little time?





Week 9 복습 – Fully Convolutional Networks for Semantic Segmentation(FCN)

becoming fully convolutional



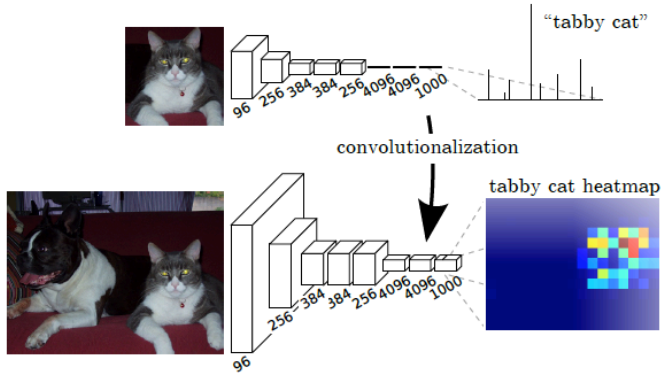
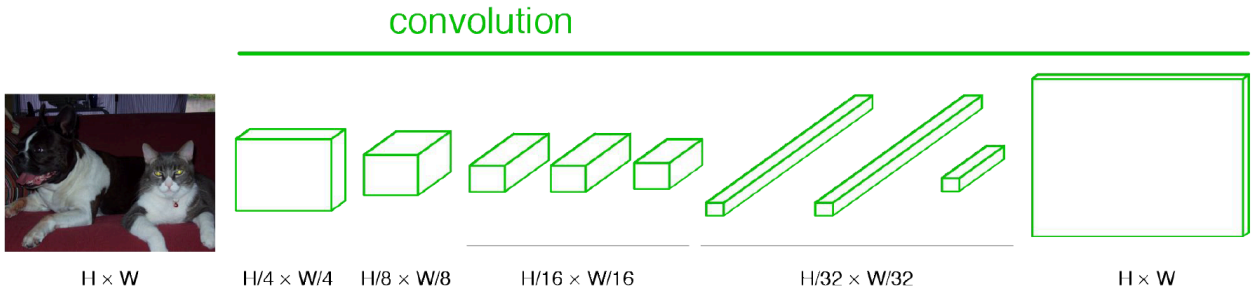


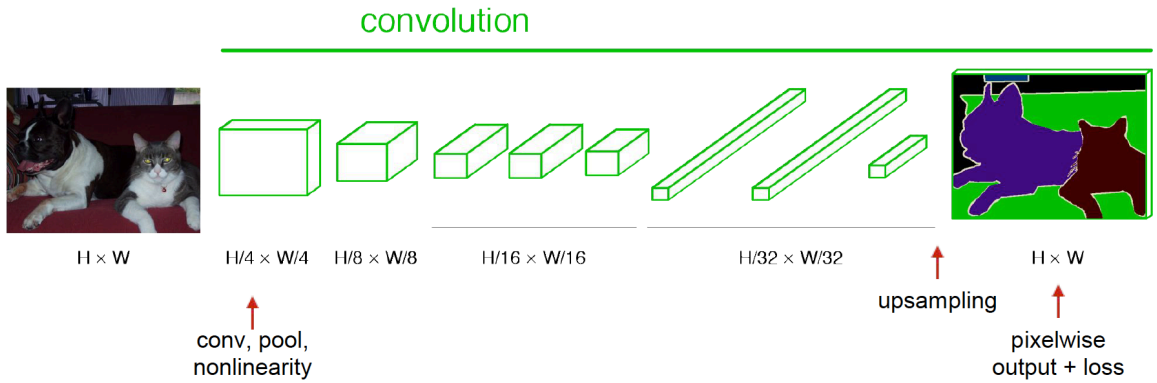
Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Week 9 복습 – Fully Convolutional Networks for Semantic Segmentation(FCN)

upsampling output



end-to-end, pixels-to-pixels network



Week 9 복습 – Skip Layers

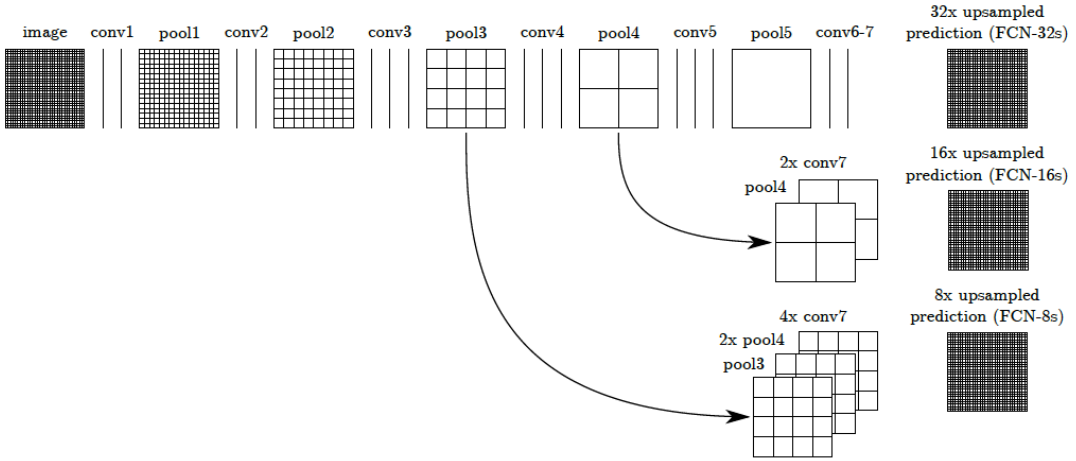
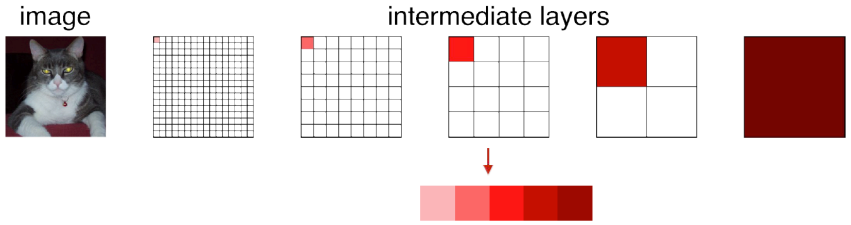


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

spectrum of deep features

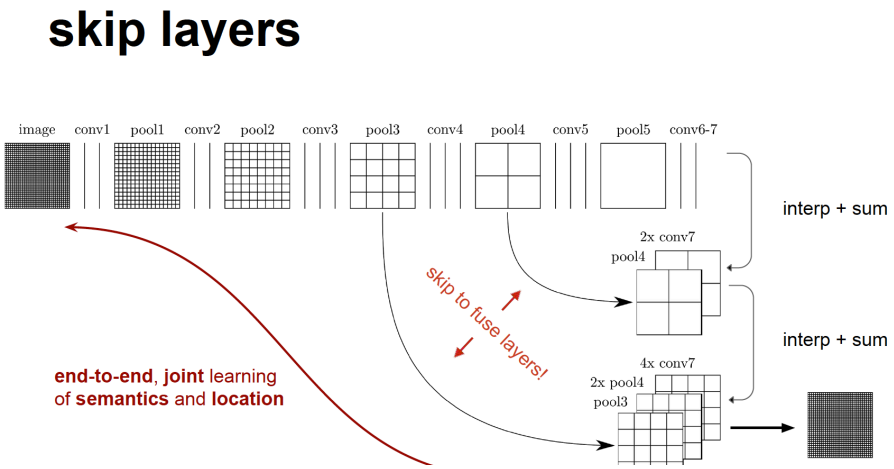
combine *where* (local, shallow) with *what* (global, deep)

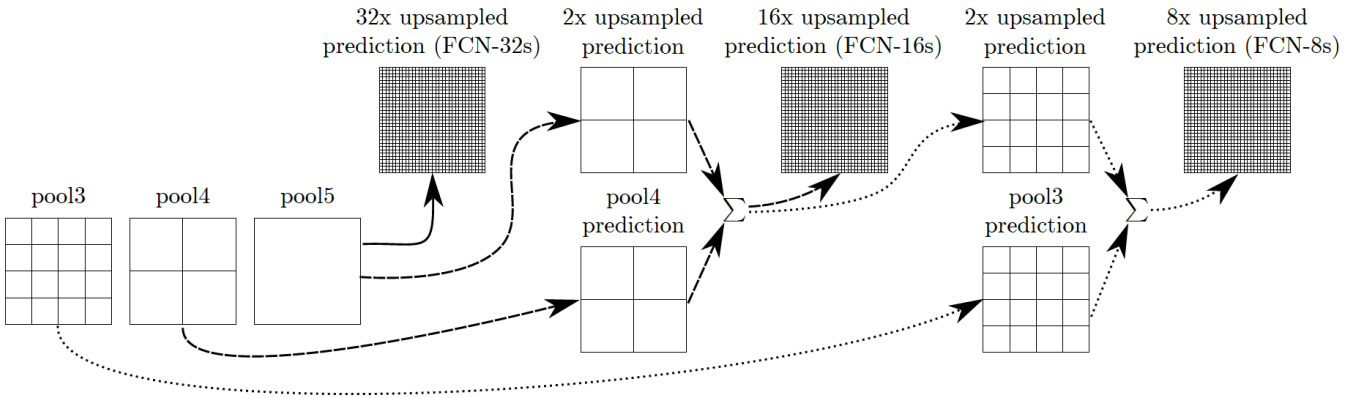


fuse features into **deep jet**

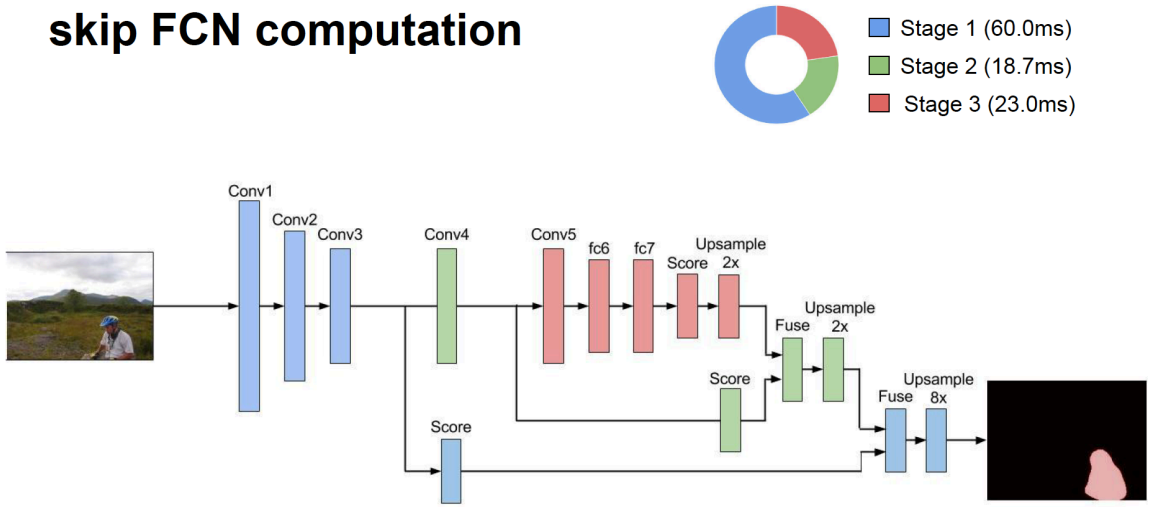
(cf. Hariharan et al. CVPR15 "hypercolumn")

Week 9 복습 – Skip Layers

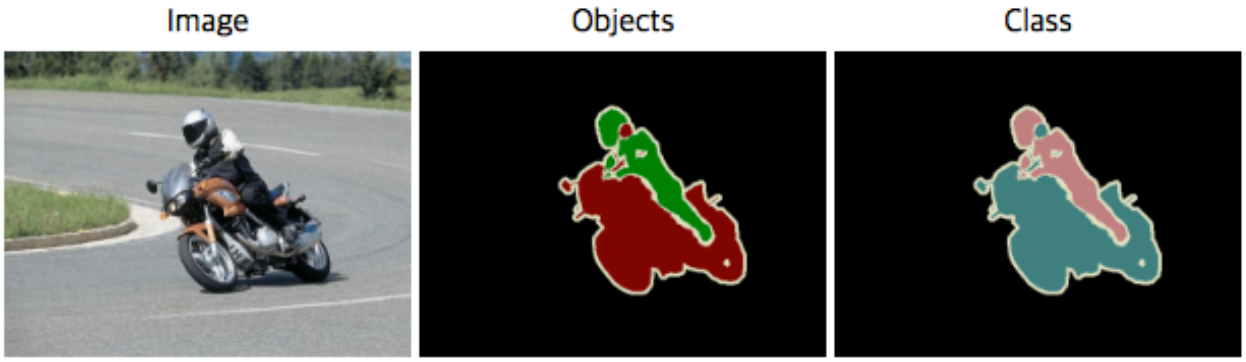




Week 9 복습 – Skip Layers & PASCAL VOC Dataset

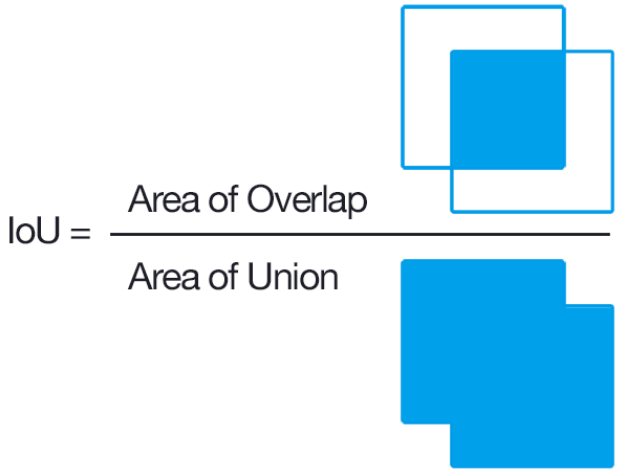


A multi-stream network that fuses features/predictions across layers



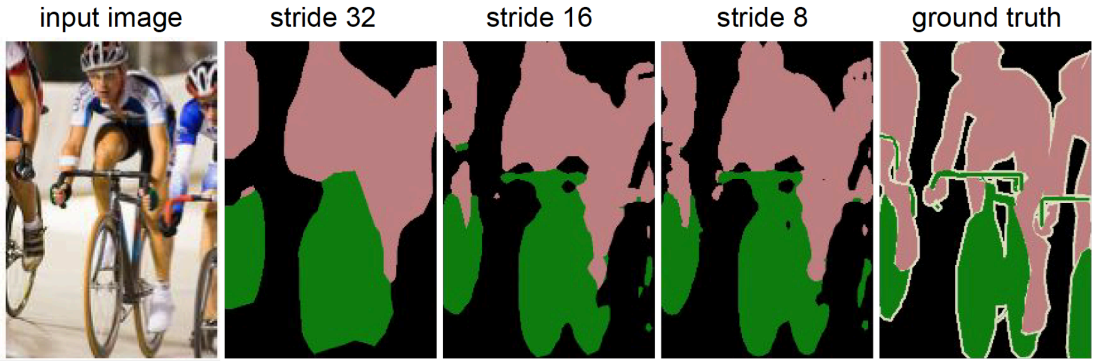
Week 9 복습 – Intersection over Union(IoU) Metric

- pixel accuracy: $\sum_i n_{ii} / \sum_i t_i$
- mean accuraccy: $(1/n_{\text{cl}}) \sum_i n_{ii}/t_i$
- mean IU: $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
- frequency weighted IU: $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$



Week 9 복습 – Skip Layers의 효과

skip layer refinement



13

Table 2. Comparison of skip FCNs on a subset[7] of PASCAL VOC 2011 segval. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Week 9 복습 – 성능비교

Table 1. We adapt and extend three classification convnets. We compare performance by mean intersection over union on the validation set of PASCAL VOC 2011 and by inference time (averaged over 20 trials for a 500 × 500 input on an NVIDIA Tesla K40c). We detail the architecture of the adapted nets with regard to dense prediction: number of parameter layers, receptive field size of output units, and the coarsest stride within the net. (These numbers give the best performance obtained at a fixed learning rate, not best performance possible.)

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

Table 3. Our fully convolutional net gives a 20% relative improvement over the state-of-the-art on the PASCAL VOC 2011 and 2012 test sets and reduces inference time.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [12]	47.9	-	-
SDS [17]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	**~ 175 ms**

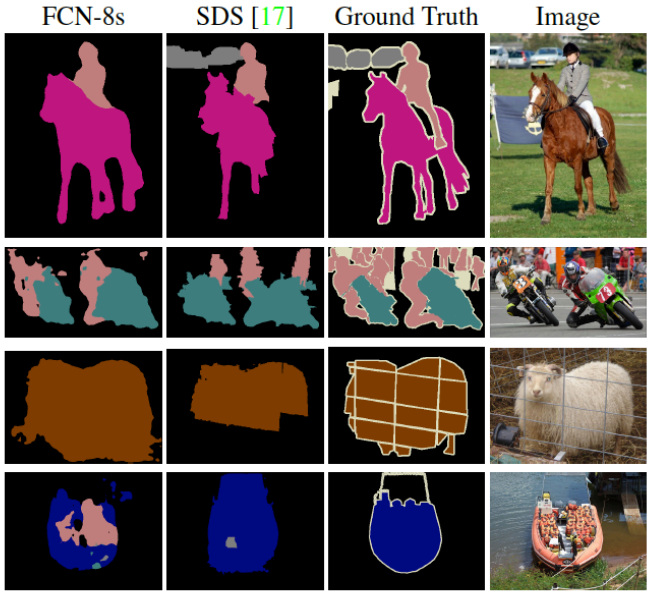
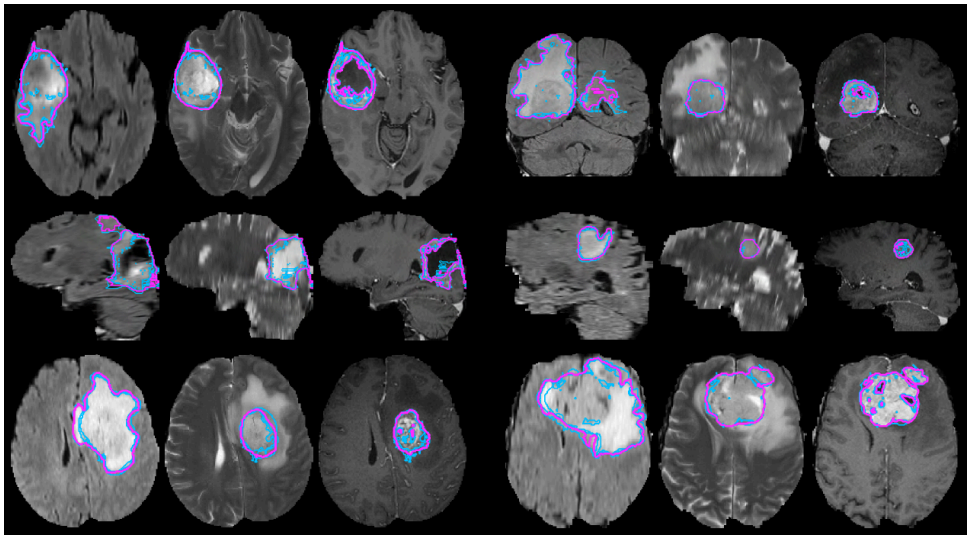
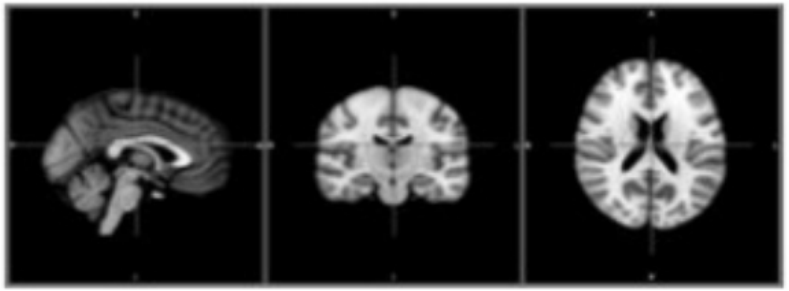


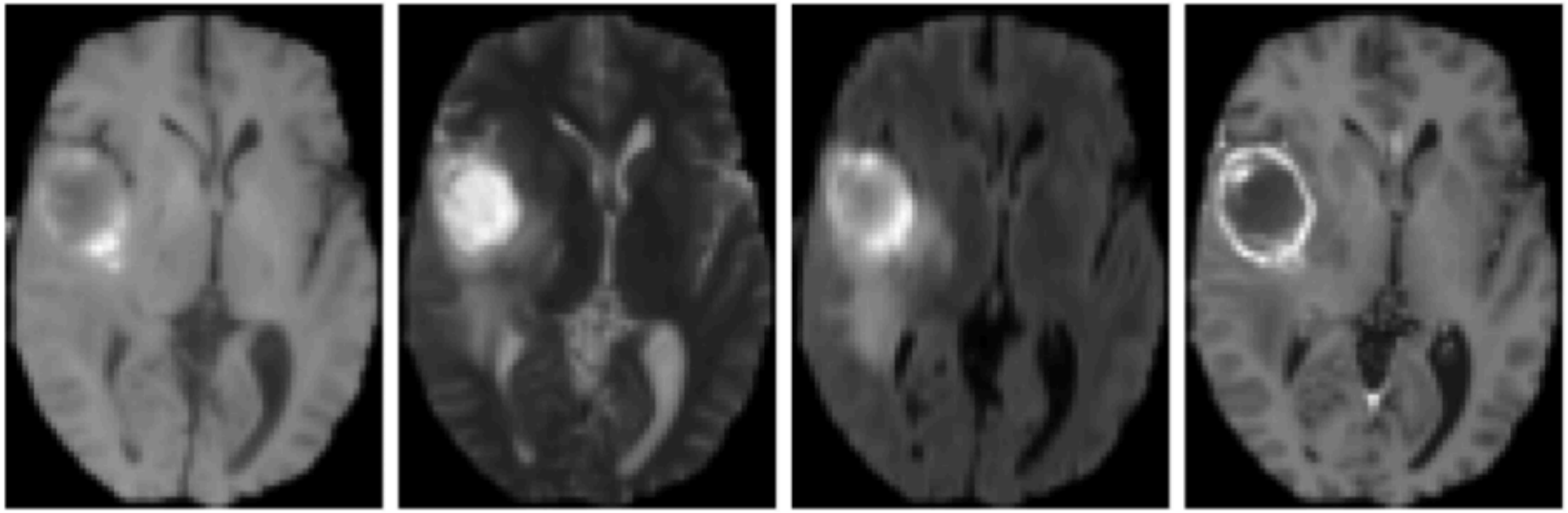
Figure 6. Fully convolutional segmentation nets produce state-of-the-art performance on PASCAL. The left column shows the output of our highest performing net, FCN-8s. The second shows the segmentations produced by the previous state-of-the-art system by Hariharan *et al*. [17]. Notice the fine structures recovered (first row), ability to separate closely interacting objects (second row), and robustness to occluders (third row). The fourth row shows a failure case: the net sees lifejackets in a boat as people.

Week 9 복습 – BraTS Database





Week 9 복습 – BraTS Database



(a) T1 (b) T2 (c) FLAIR (d) T1c

1 for necrosis
2 for edema
3 for non-enhancing tumor
4 for enhancing tumor
0 for everything else

Week 9 복습 – Maxout

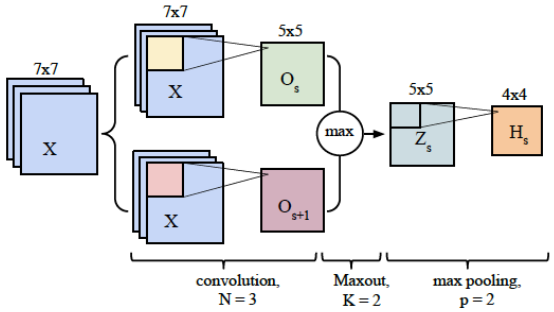


Figure 1: A single convolution layer block showing computations for a single feature map. The input patch (here 7×7), is convolved with series of kernels (here 3×3) followed by Maxout and max-pooling.

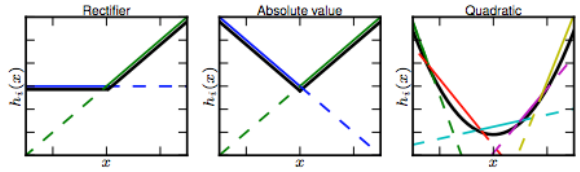
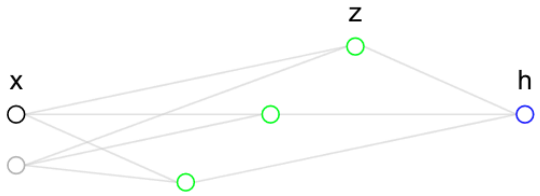
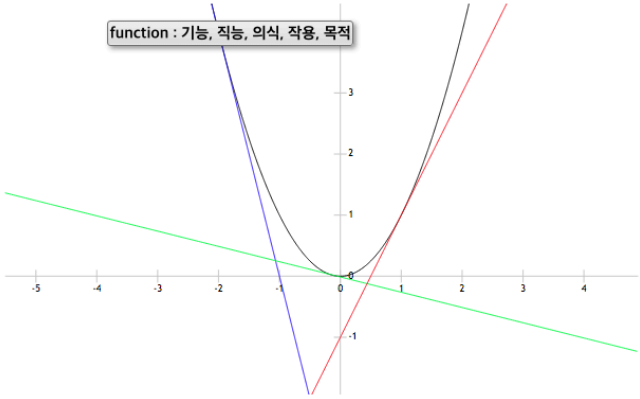


Figure 1. Graphical depiction of how the maxout activation function can implement the rectified linear, absolute value rectifier, and approximate the quadratic activation function. This diagram is 2D and only shows how maxout behaves with a 1D input, but in multiple dimensions a maxout unit can approximate arbitrary convex functions.



Input Units
Fully-connected Units
Max-pooling Units
Bias



Week 9 복습 – Brain Tumor Segmentation with Deep Neural Networks

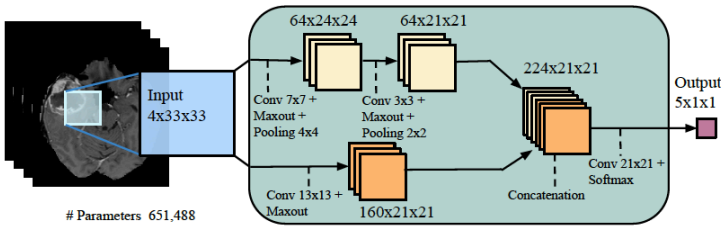
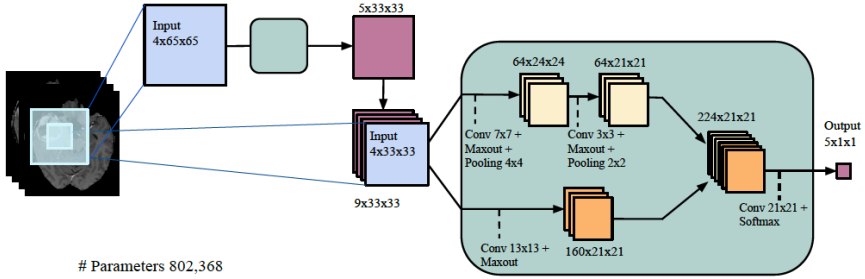
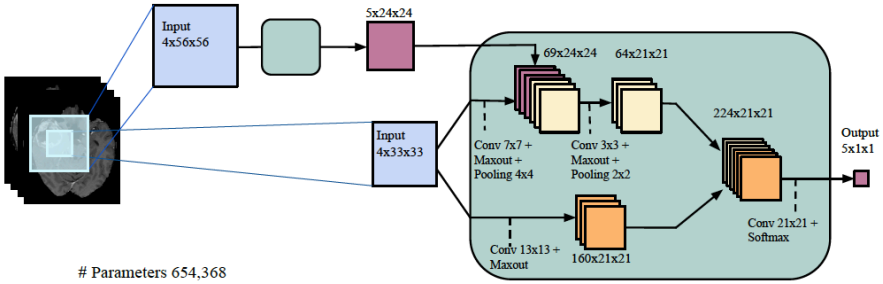


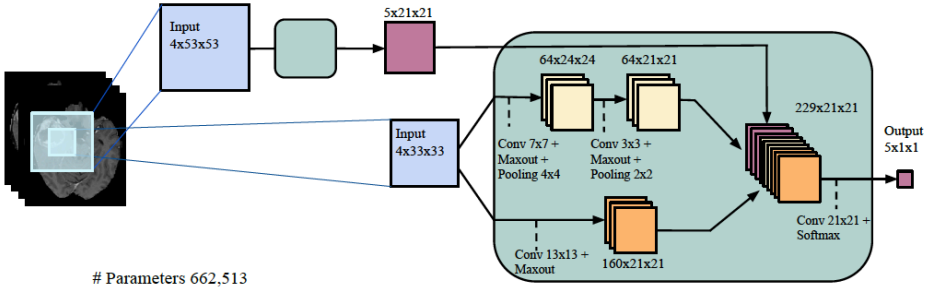
Figure 2: Two-pathway CNN architecture (TwoPathCNN). The figure shows the input patch going through two paths of convolutional operations. The feature-maps in the local and global paths are shown in yellow and orange respectively. The convolutional layers used to produce these feature-maps are indicated by dashed lines in the figure. The green box embodies the whole model which in later architectures will be used to indicate the TwoPathCNN.



(a) Cascaded architecture, using input concatenation (InputCascadeCNN).



(b) Cascaded architecture, using local pathway concatenation (LocalCascadeCNN).



(c) Cascaded architecture, using pre-output concatenation, which is an architecture with properties similar to that of learning using a limited number of mean-field inference iterations in a CRF (MFCascadeCNN).

Week 9 복습 – Brain Tumor Segmentation with Deep Neural Networks

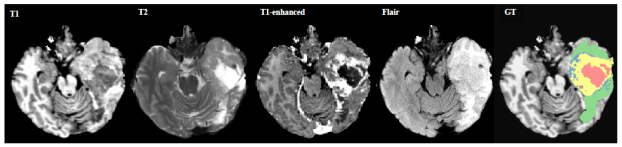


Figure 4: The first four images from left to right show the MRI modalities used as input channels to various CNN models and the fifth image shows the ground truth labels where edema, enhanced tumor, necrosis, non-enhanced tumor.

Table 1: Performance of the TwoPathCNN model and variations. The second phase training is noted by appending '*' to the architecture name. The 'Rank' column represents the ranking of each method in the online score board at the time of submission.

Rank	Method	Dice			Specificity			Sensitivity		
		Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
4	TwoPathCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
9	LocalPathCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
10	AverageCNN*	0.84	0.75	0.70	0.95	0.83	0.73	0.77	0.74	0.73
14	GlobalPathCNN*	0.82	0.73	0.68	0.93	0.81	0.70	0.75	0.65	0.70
14	TwoPathCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
15	LocalPathCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80

Table 2: Performance of the cascaded architectures. The reported results are from the second phase training. The 'Rank' column shows the ranking of each method in the online score board at the time of submission.

Rank	Method	Dice			Specificity			Sensitivity		
		Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
2	InputCascadeCNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
4-a	MFCascadeCNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
4-c	LocalCascadeCNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75

Week 9 복습 – Brain Tumor Segmentation with Deep Neural Networks

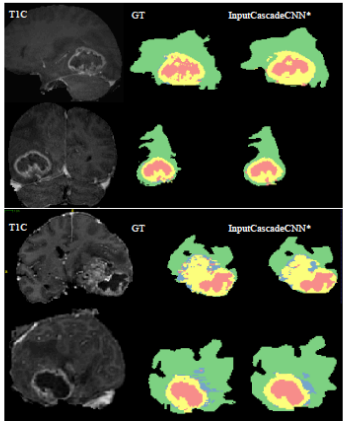


Figure 8: Visual results from our top performing model, InputCascadeCNN* on Coronal and Sagittal views. The subjects are the same as in Figure 7. In every sub-figure, the top row represents the Sagital view and the bottom row represents the Coronal view. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

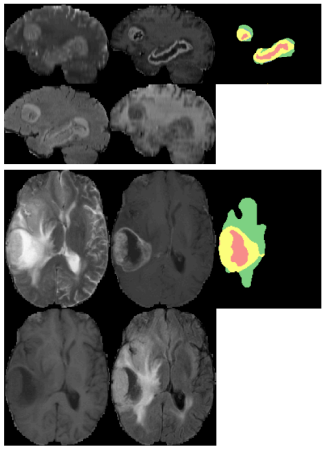


Figure 9: Visual segmentation results from our top performing model, InputCascadeCNN*, on examples of the BRATS2013 test dataset in Saggital (top) and Axial (bottom) views. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

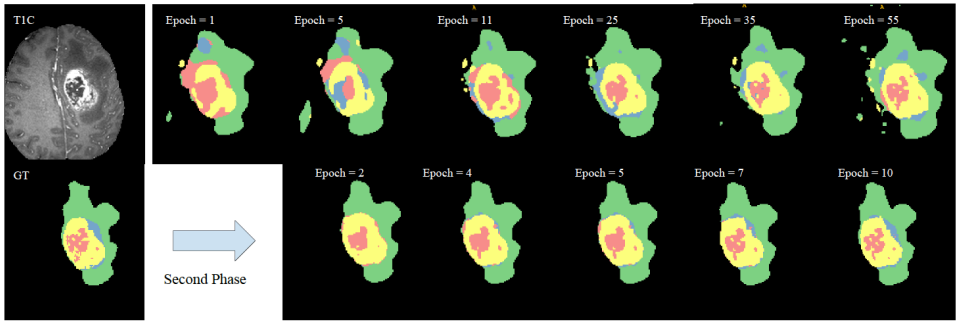


Figure 6: Progression of learning in InputCascadeCNN*. The stream of figures on the first row from left to right show the learning process during the first phase. As the model learns better features, it can better distinguish boundaries between tumor sub-classes. This is made possible due to uniform label distribution of patches during the first phase training which makes the model believe all classes are equiprobable and causes some false positives. This drawback is alleviated by training a second phase (shown in second row from left to right) on a distribution closer to the true distribution of labels. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

Week 9 복습 – Brain Tumor Segmentation with Deep Neural Networks

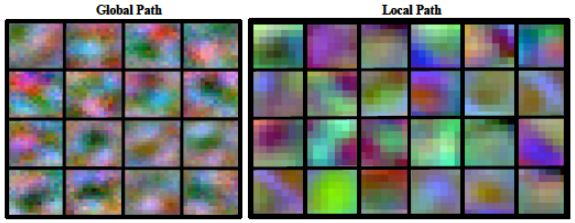


Figure 5: Randomly selected filters from the first layer of the model. From left to right the figure shows visualization of features from the first layer of the global and local path respectively. Features in the local path include more edge detectors while the global path contains more localized features.

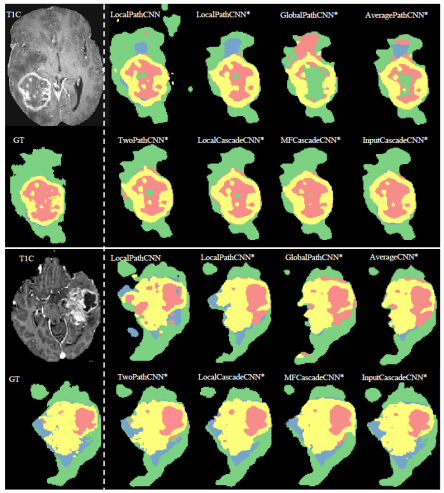


Figure 7: Visual results from our CNN architectures from the Axial view. For each sub-figure, the top row from left to right shows T1C modality, the conventional one path CNN, the Conventional CNN with two training phases, and the TwoPathCNN model. The second row from left to right shows the ground truth, LocalCascadeCNN model, the MFCascadeCNN model and the InputCascadeCNN. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

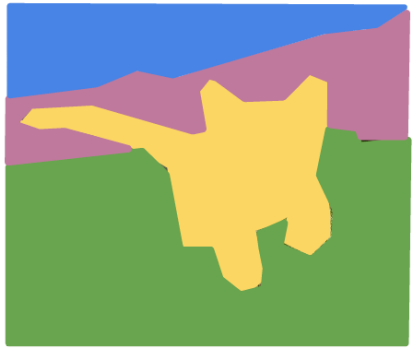
Table 3: Comparison of our implemented architectures with the state-of-the-art methods on the BRATS-2013 test set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
InputCascadeCNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
Tustison	0.87	0.78	0.74	0.85	0.74	0.69	0.89	0.88	0.83
MFCascadeCNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
TwoPathCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
LocalCascadeCNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75
LocalPathCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
Meier	0.82	0.73	0.69	0.76	0.78	0.71	0.92	0.72	0.73
Reza	0.83	0.72	0.72	0.82	0.81	0.70	0.86	0.69	0.76
Zhao	0.84	0.70	0.65	0.80	0.67	0.65	0.89	0.79	0.70
Cordier	0.84	0.68	0.65	0.88	0.63	0.68	0.81	0.82	0.66
TwoPathCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
LocalPathCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80
Festa	0.72	0.66	0.67	0.77	0.77	0.70	0.72	0.60	0.70
Doyle	0.71	0.46	0.52	0.66	0.38	0.58	0.87	0.70	0.55

Object Detection

Other Computer Vision Tasks

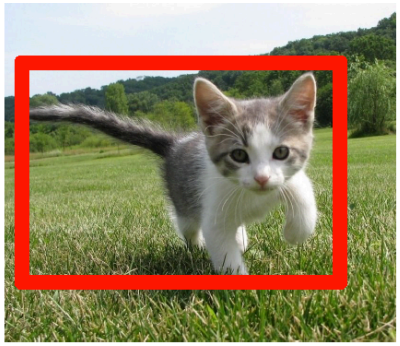
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

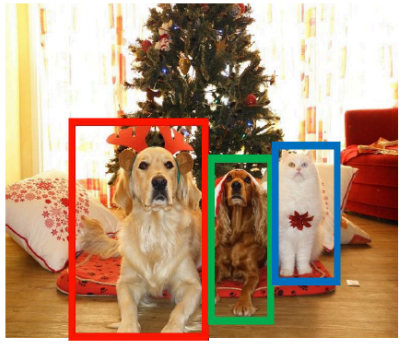
Classification + Localization



CAT

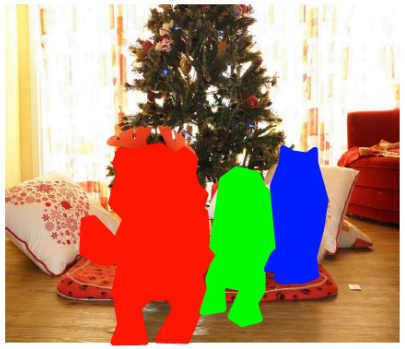
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

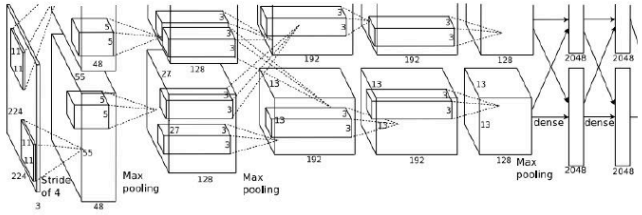
Multiple Object

Slide credit: CS231n (https://goo.gl/QSA36t)

Object Detection

Object Detection as Classification: Sliding Window

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



Dog? NO
Cat? YES
Background? NO

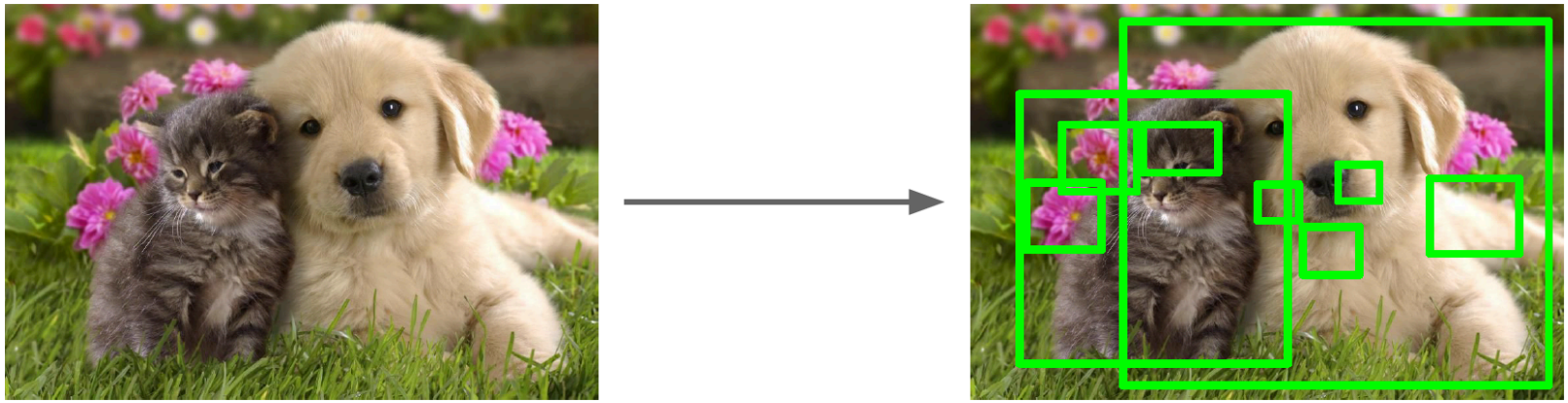
Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

Slide credit: CS231n (https://goo.gl/QSA36t)

Region Proposal

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU

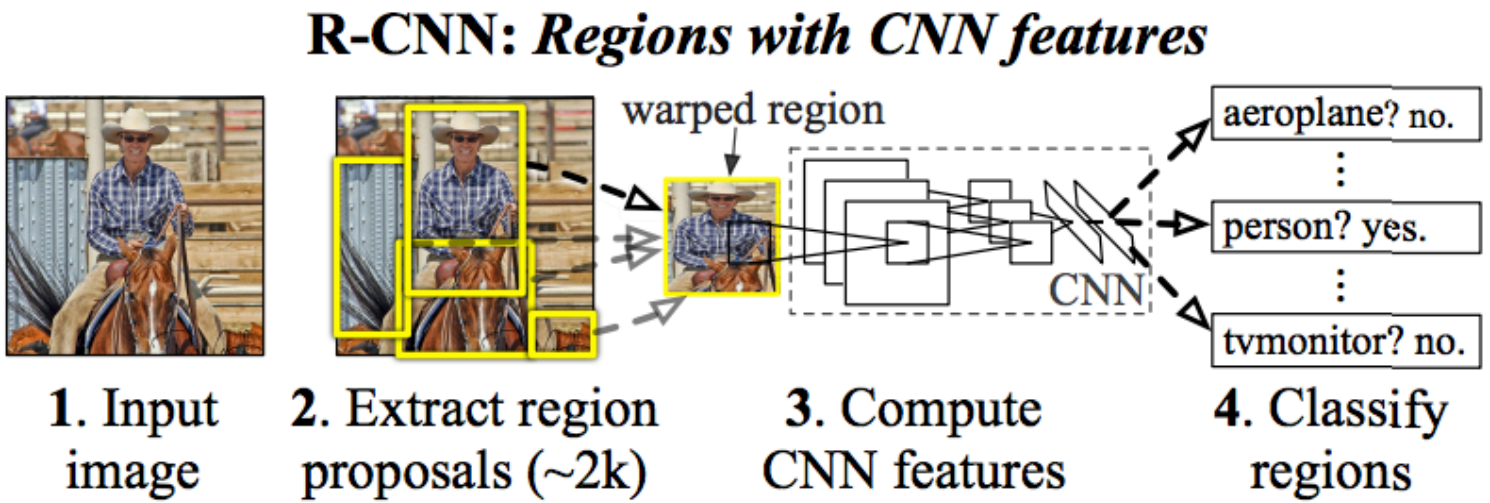


Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Slide credit: CS231n (https://goo.gl/QSA36t)

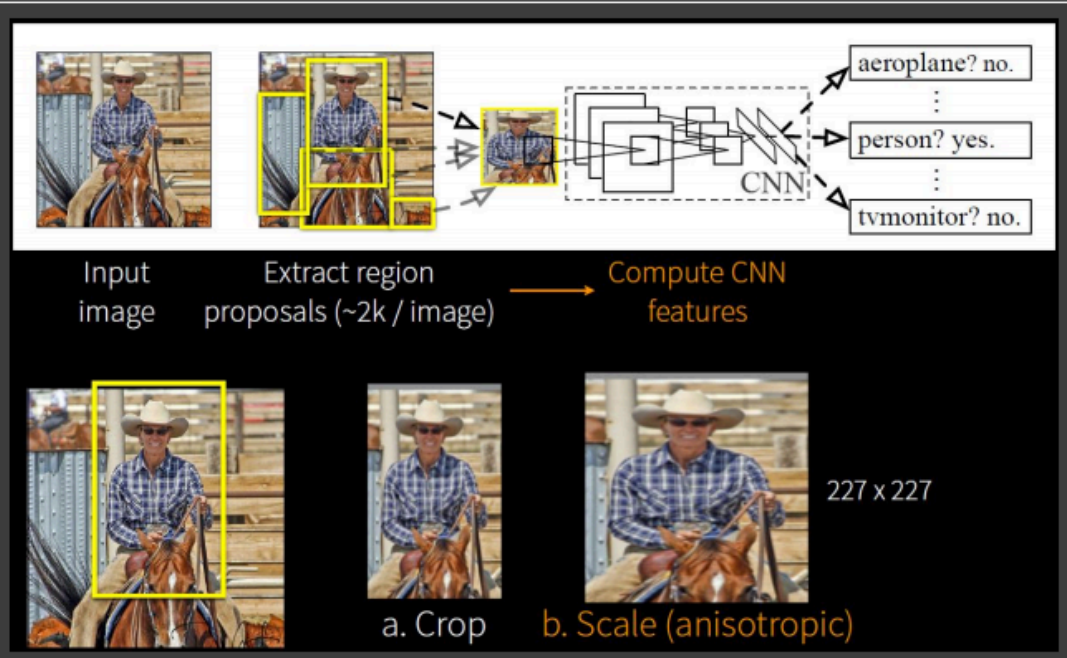
R-CNN(Regions with CNN features)

- Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." CVPR. 2014.
- https://arxiv.org/pdf/1311.2524.pdf



R-CNN(Regions with CNN features)

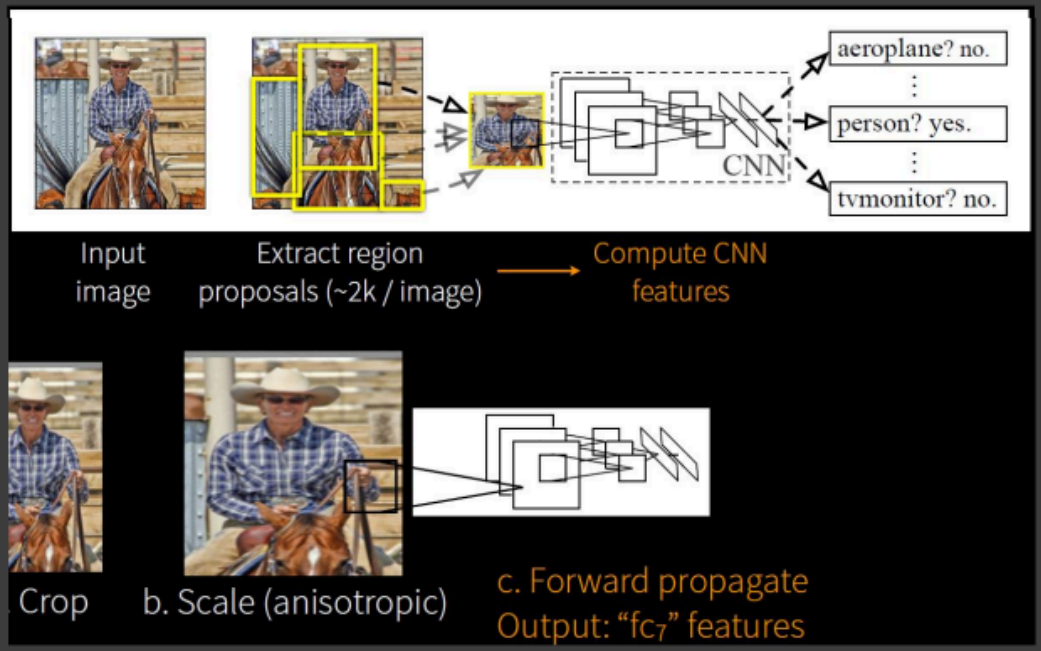
R-CNN: Step 2



Slide credit: COLLIN MCCARTHY
(https://goo.gl/AZB14j)

R-CNN(Regions with CNN features)

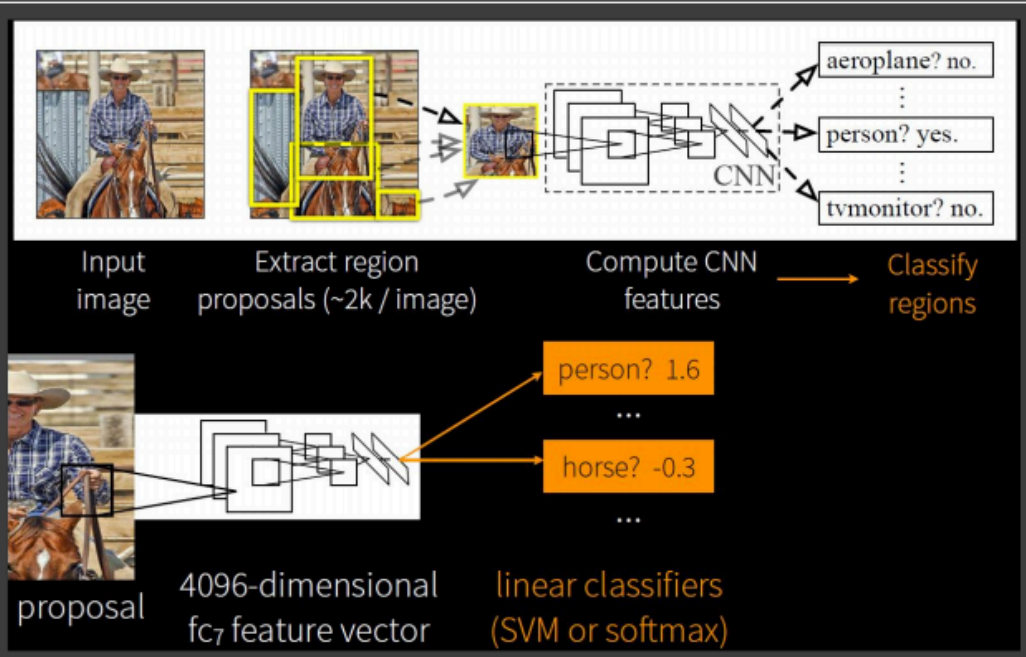
R-CNN: Step 2



Slide credit: COLLIN MCCARTHY (https://goo.gl/AZB14j)

R-CNN(Regions with CNN features)

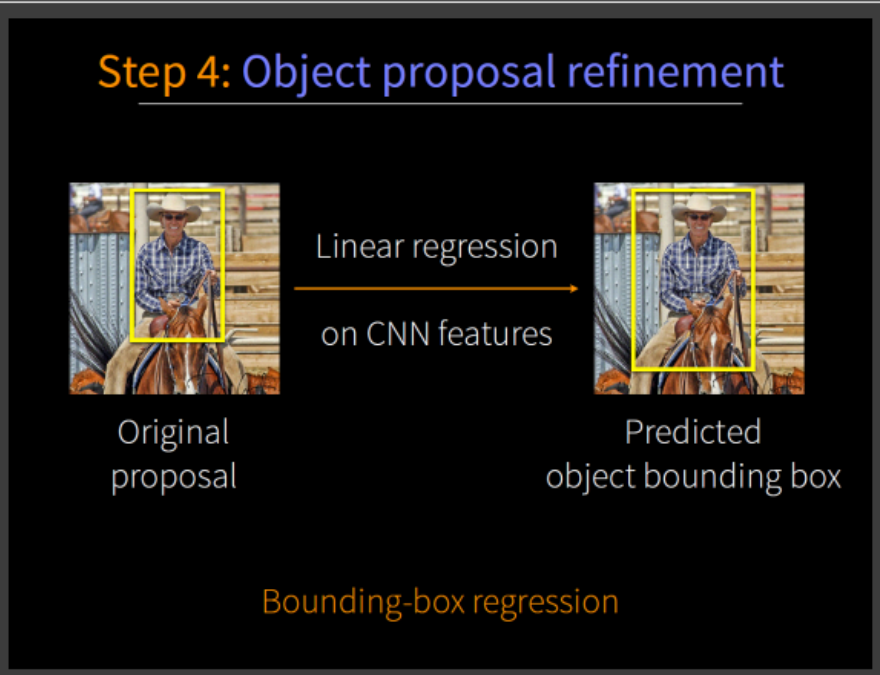
R-CNN: Step 3



Slide credit: COLLIN MCCARTHY (https://goo.gl/AZB14j)

R-CNN(Regions with CNN features)

R-CNN: Step 4



Slide credit: COLLIN MCCARTHY (https://goo.gl/AZB14j)

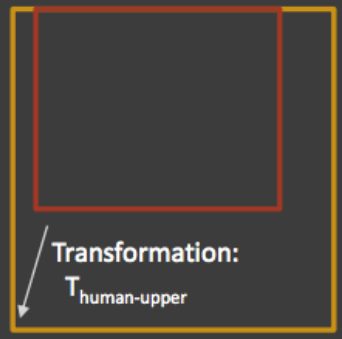
Linear Regression

R-CNN: Step 4

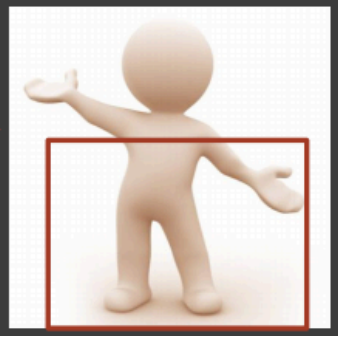
Predicting Object Bounding Box

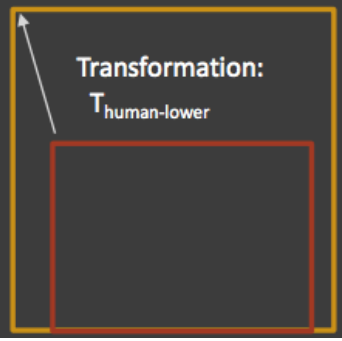
Ground Truth Bounding Box

Features: $F_{human\text{-}upper}$



Features: $F_{human\text{-}lower}$



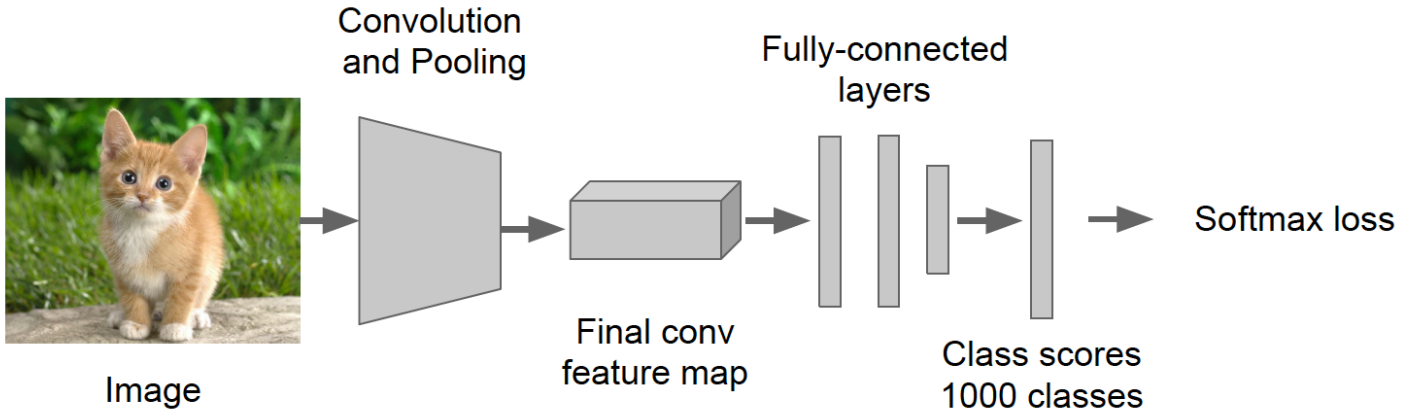


Slide credit: COLLIN MCCARTHY (https://goo.gl/AZB14j)

R-CNN

R-CNN Training

Step 1: Train (or download) a classification model for ImageNet (AlexNet)



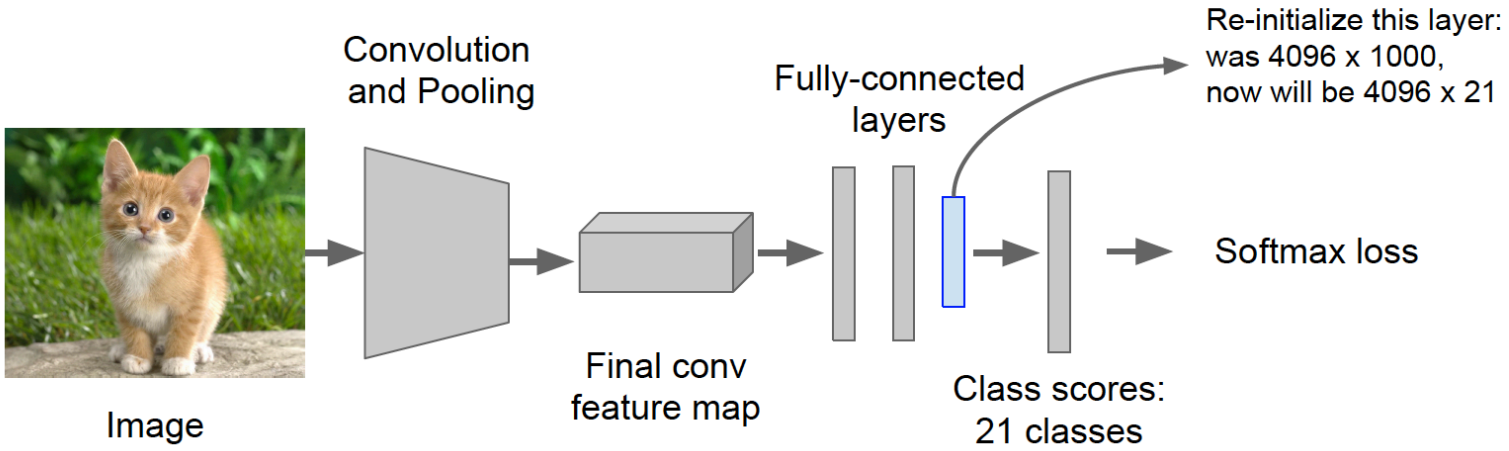
Slide credit: CS231n
(https://goo.gl/QSA36t)

R-CNN

R-CNN Training

Step 2: Fine-tune model for detection

- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



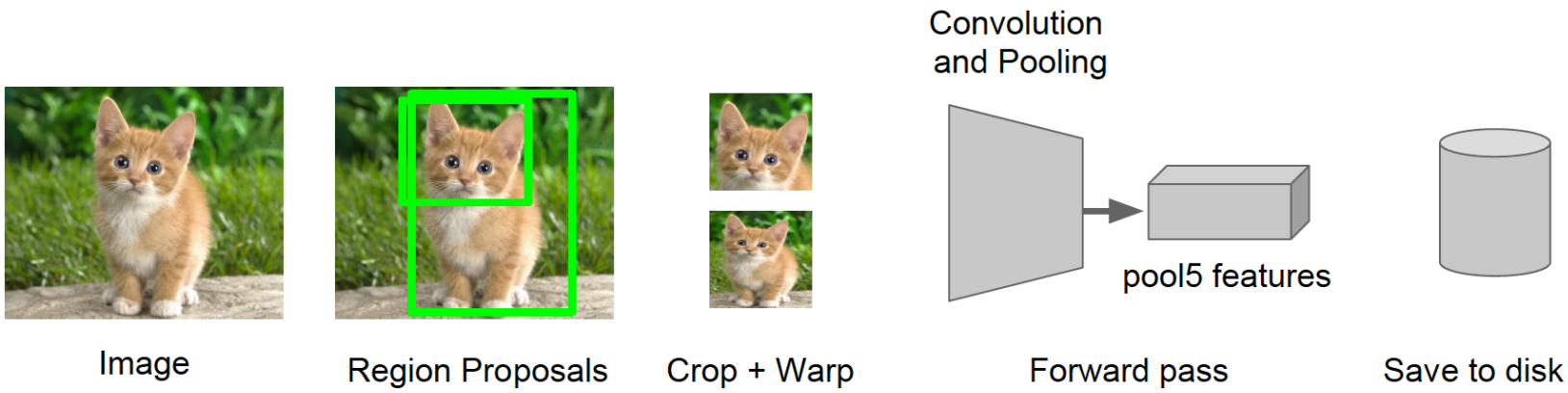
Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN

R-CNN Training

Step 3: Extract features

- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!



Slide credit: CS231n (https://goo.gl/QSA36t)

Training Mini-Batch (CNN Fine-Tuning)

- We treat all region proposals with ≥ **0.5 IoU overlap with a ground-truth box as positives** for that box's class and the **rest as negatives**.
- In each SGD iteration, we uniformly sample **32 positive windows** (over all classes) and **96 background windows** to construct a **mini-batch of size 128.**
- We bias the sampling towards positive windows because they are extremely rare compared to background.

R-CNN

R-CNN Training

Step 5 (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals

Training image regions



Cached region features



Regression targets
(dx, dy, dw, dh)
Normalized coordinates

(0, 0, 0, 0)
Proposal is good

(.25, 0, 0, 0)
Proposal too far to left

(0, 0, -0.125, 0)
Proposal too wide

Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN

Object Detection: Datasets

	PASCAL VOC (2010)	ImageNet Detection (ILSVRC 2014)	MS-COCO (2014)
Number of classes	20	**200**	80
Number of images (train + val)	~20k	**~470k**	~120k
Mean objects per image	2.4	1.1	**7.2**

Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN

Object Detection: Evaluation

We use a metric called “mean average precision” (mAP)

Compute average precision (AP) separately for each class, then average over classes

A detection is a true positive if it has IoU with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)

Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve

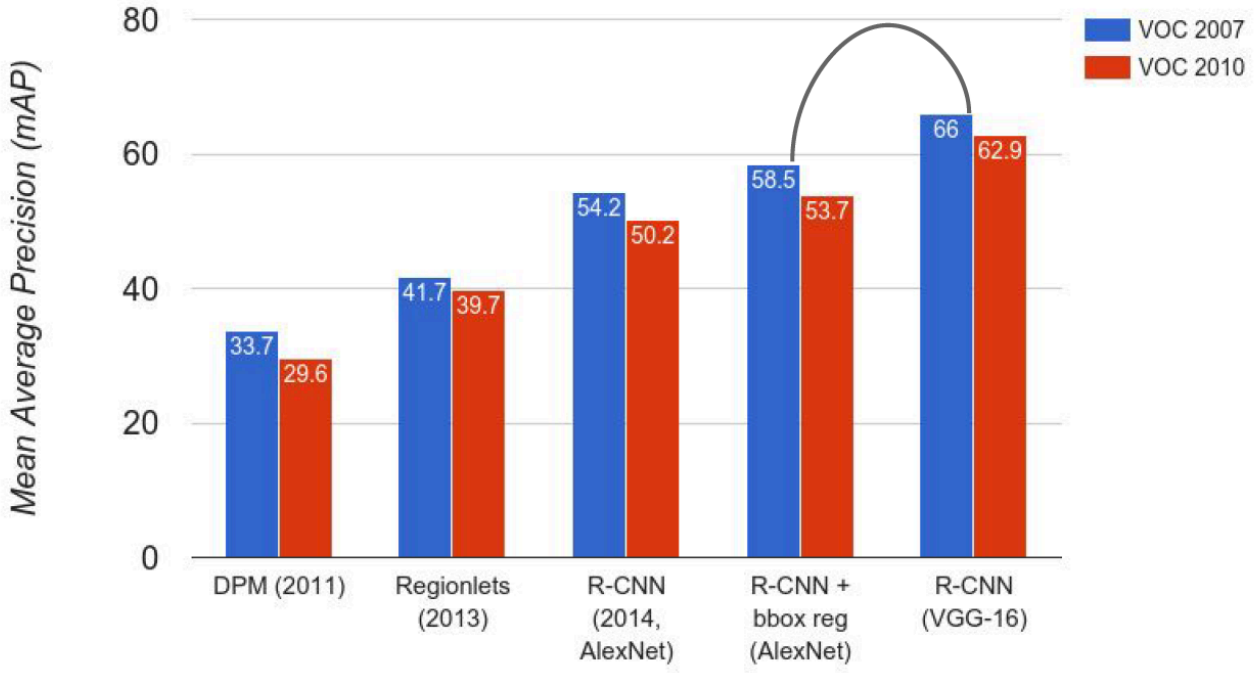
TL;DR mAP is a number from 0 to 100; high is good

Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN

R-CNN Results

Features from a deeper network help a lot



Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN

R-CNN Problems

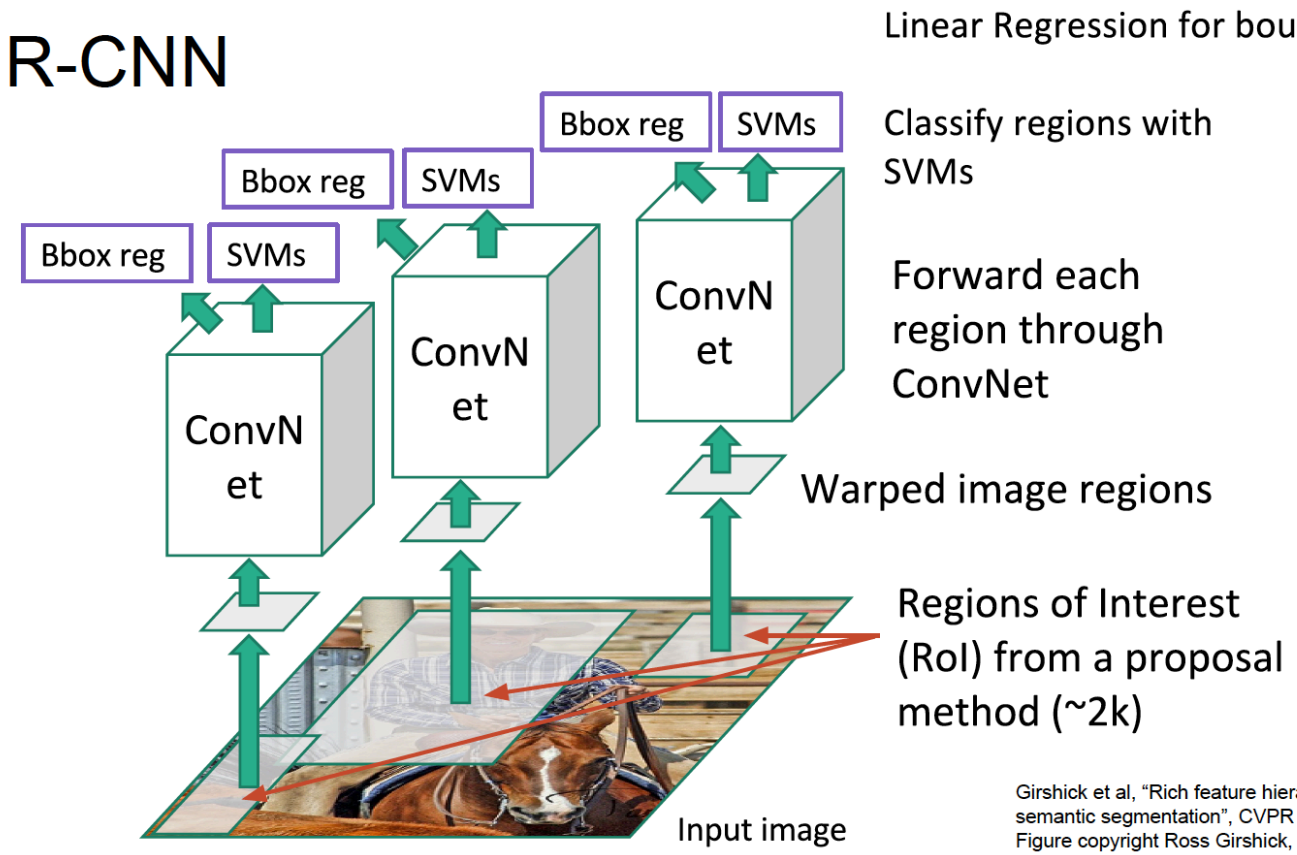
1. Slow at test-time: need to run full forward pass of CNN for each region proposal

2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline

Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN



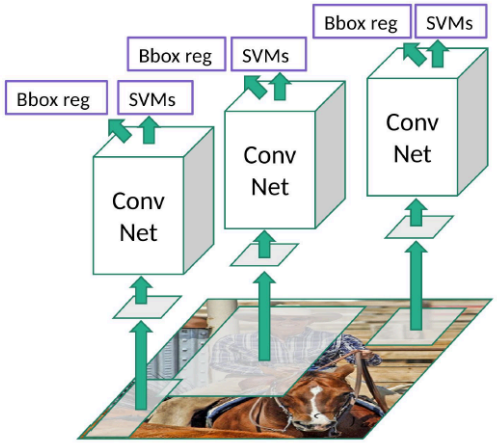
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Slide credit: CS231n (https://goo.gl/QSA36t)

R-CNN

R-CNN: Problems

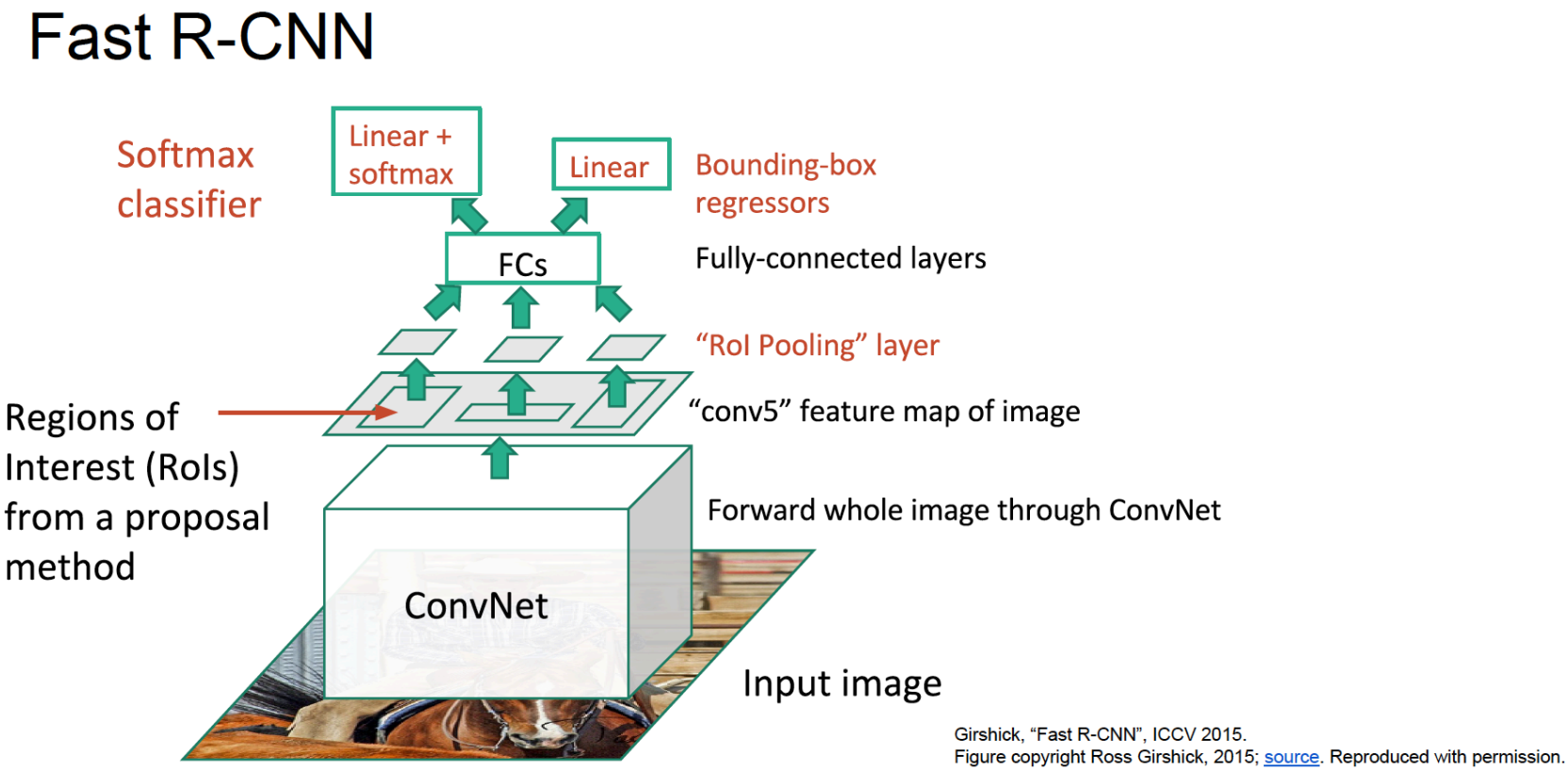
- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

Slide credit: CS231n (https://goo.gl/QSA36t)

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Slide credit: CS231n (https://goo.gl/QSA36t)

Fast R-CNN

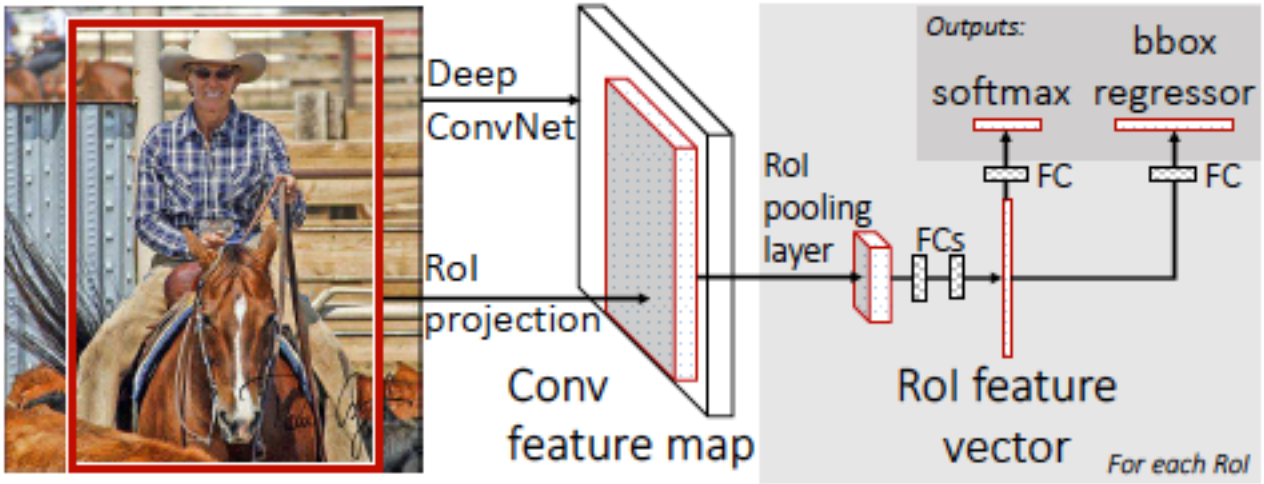
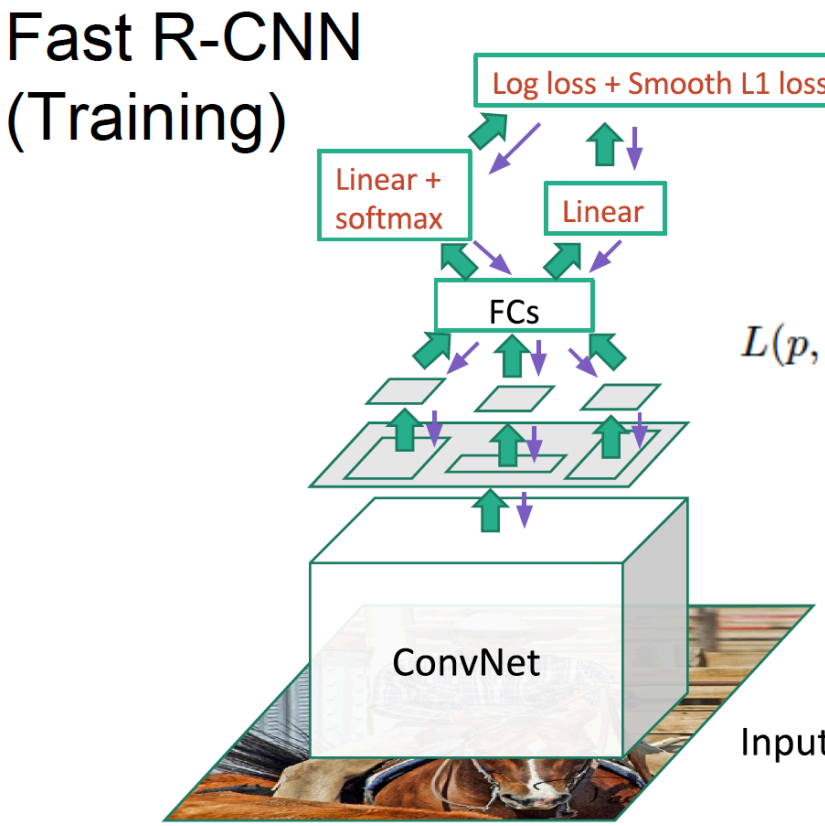


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Fast R-CNN



Multi-Task Loss

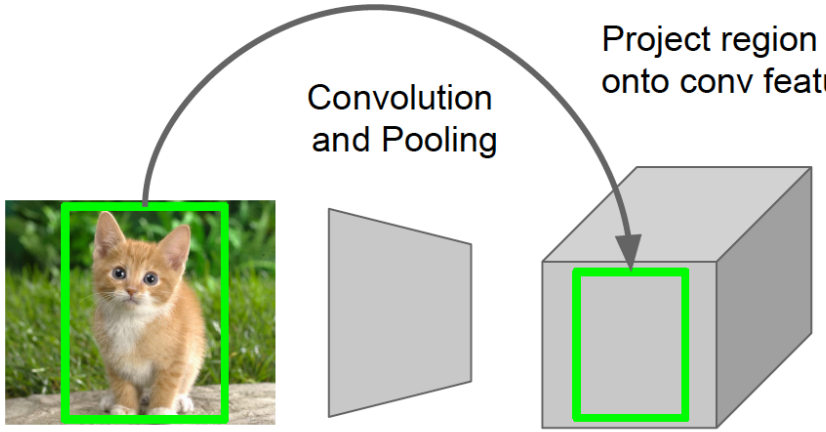
$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \quad (1)$$

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

Slide credit: CS231n (https://goo.gl/QSA36t)

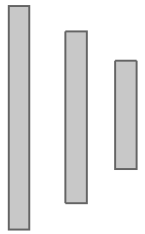
Fast R-CNN

Fast R-CNN: Region of Interest Pooling



Hi-res input image:
3 x 800 x 600
with region
proposal

Hi-res conv features:
C x H x W
with region proposal



Problem: Fully-connected layers expect low-res conv features: C x h x w

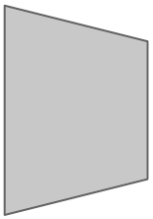
Slide credit: CS231n (https://goo.gl/QSA36t)

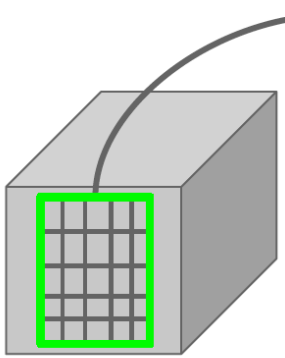
Fast R-CNN

Fast R-CNN: Region of Interest Pooling

Hi-res input image:
3 x 800 x 600
with region
proposal

Convolution
and Pooling

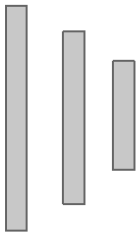




Divide projected
region into h x w grid

Hi-res conv features:
C x H x W
with region proposal

Fully-connected
layers

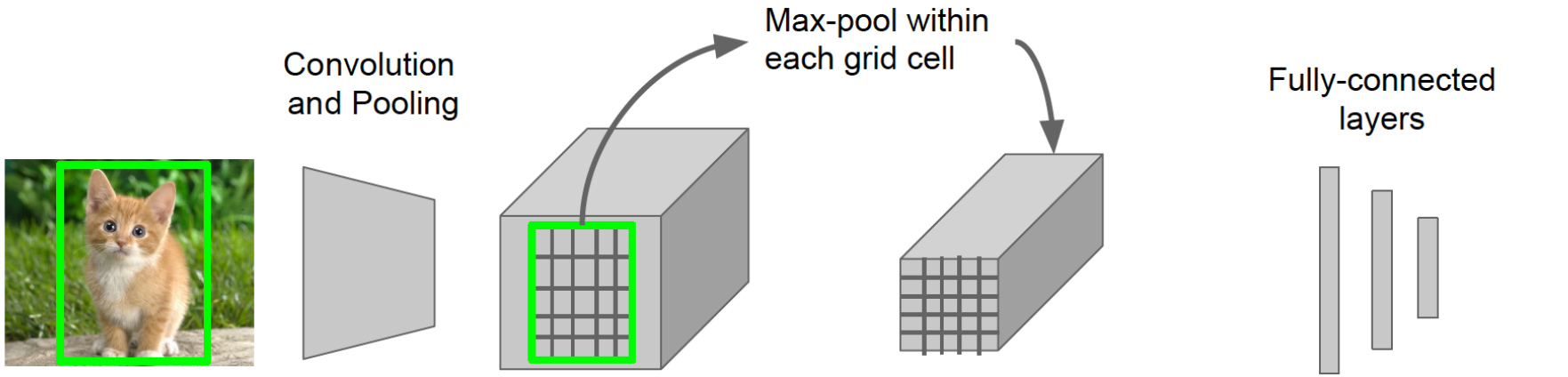


Problem: Fully-connected
layers expect low-res conv
features: C x h x w

Slide credit: CS231n
(https://goo.gl/QSA36t)

Fast R-CNN

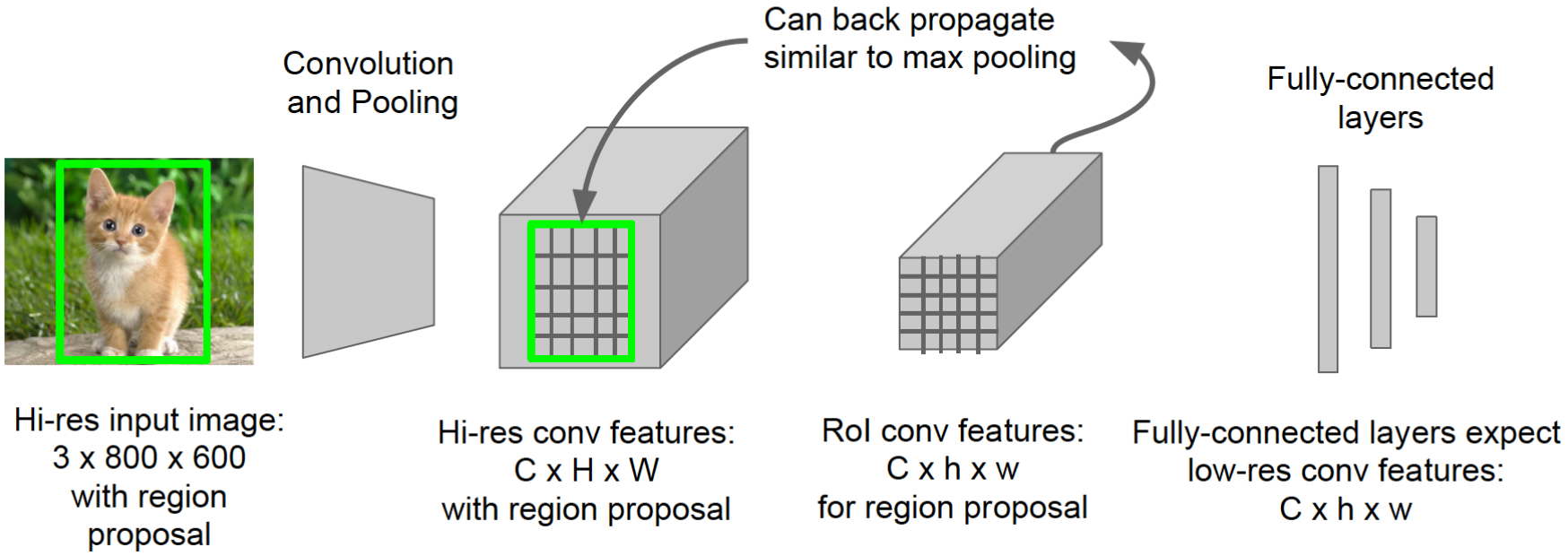
Fast R-CNN: Region of Interest Pooling



Slide credit: CS231n
(https://goo.gl/QSA36t)

Fast R-CNN

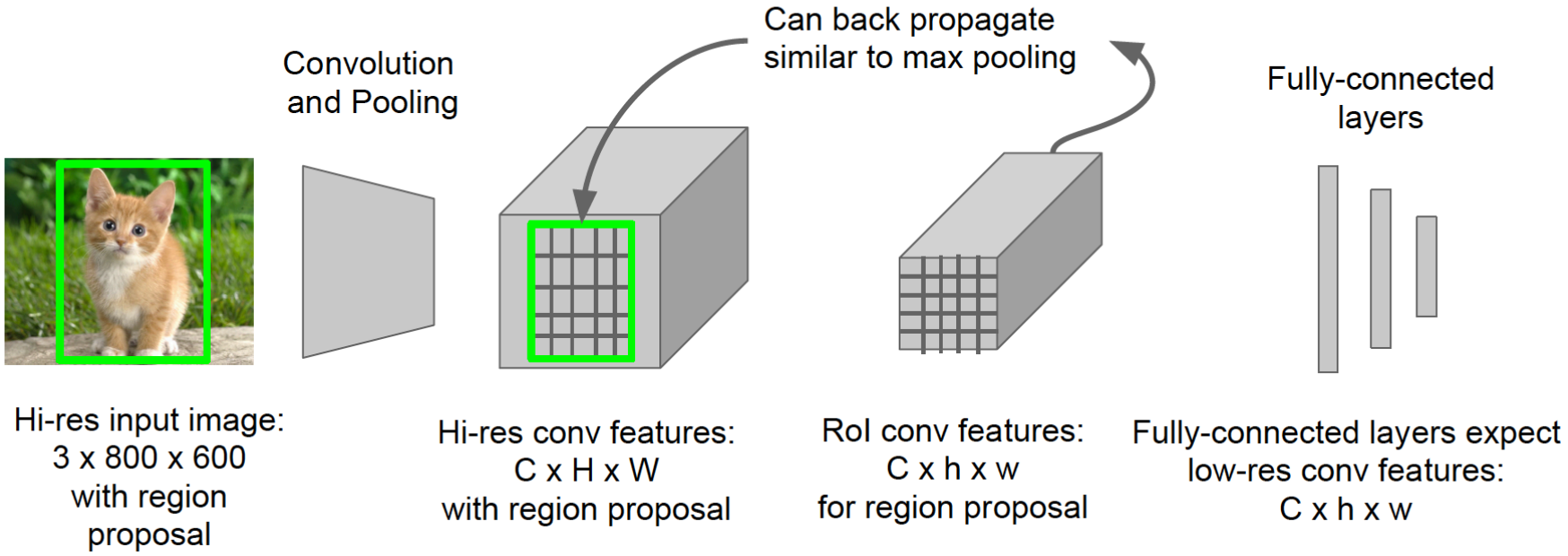
Fast R-CNN: Region of Interest Pooling



Slide credit: CS231n
(https://goo.gl/QSA36t)

Fast R-CNN

Fast R-CNN: Region of Interest Pooling



Slide credit: CS231n
(https://goo.gl/QSA36t)

Fast R-CNN

Fast R-CNN Results

	R-CNN	Fast R-CNN
Training Time:	84 hours	**9.5 hours**
(Speedup)	1x	**8.8x**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
mAP (VOC 2007)	66.0	**66.9**

Faster!

FASTER!

Better!

Using VGG-16 CNN on Pascal VOC 2007 dataset

Slide credit: CS231n (https://goo.gl/QSA36t)

Fast R-CNN Training Mini-batch

- During fine-tuning, each SGD mini-batch is constructed from **N = 2 images**, chosen uniformly at random.
- We use mini-batches of size **R = 128**, sampling **64 RoIs from each image**.
- As in [9], we take **25% of the RoIs** from object proposals that have intersection over union (**IoU) overlap with groundtruth bounding box of at least 0.5**. These RoIs comprise the examples labeled with a **foreground object class**, i.e. $u \geq 1$.
- The **remaining RoIs** are sampled from object proposals that have a maximum **IoU with ground truth in the interval [0.1, 0.5)**, following [11]. These are the **background examples** and are labeled with $u = 0$.

Fast R-CNN

Fast R-CNN Problem:

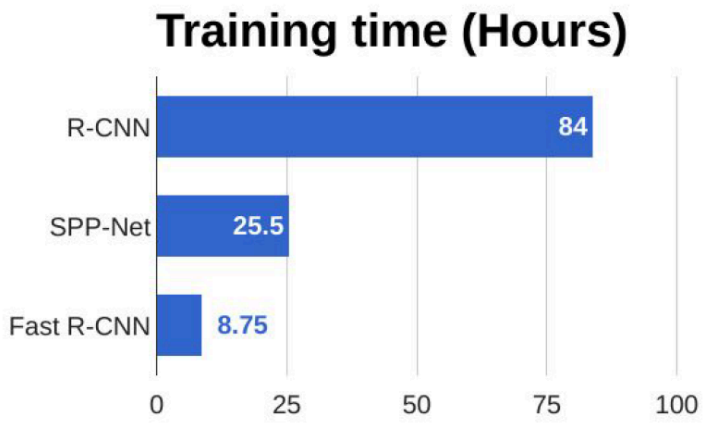
Test-time speeds don't include region proposals

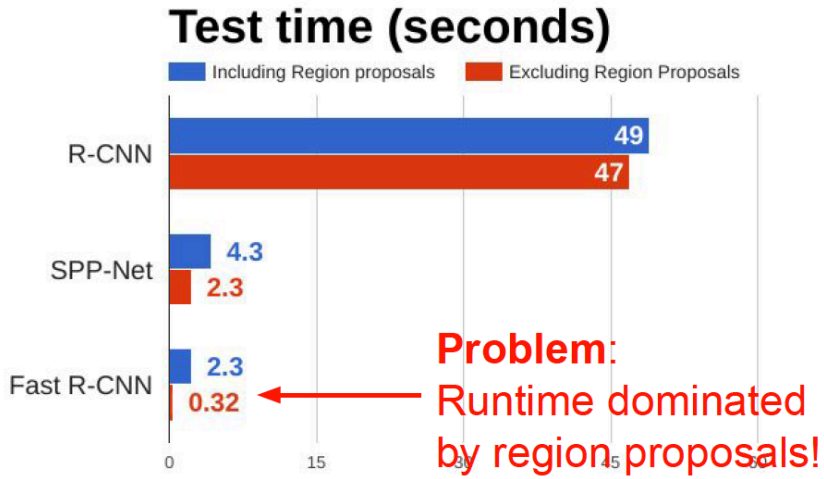
	R-CNN	**Fast R-CNN**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

Slide credit: CS231n (https://goo.gl/QSA36t)

Fast R-CNN

R-CNN vs SPP vs Fast R-CNN



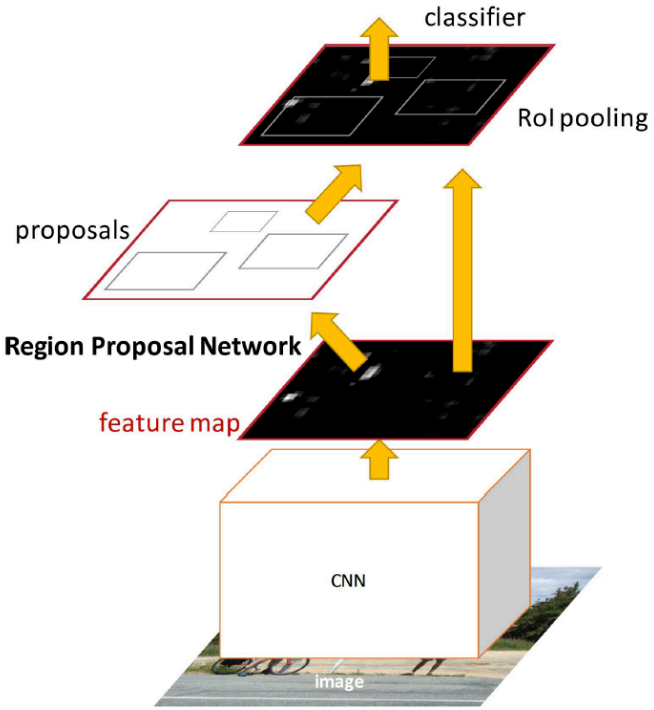


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Slide credit: CS231n
(https://goo.gl/QSA36t)

Faster R-CNN

Faster R-CNN:



Insert a **Region Proposal Network (RPN)** after the last convolutional layer

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Slide credit: Ross Girschick

Slide credit: CS231n (https://goo.gl/QSA36t)

RPN

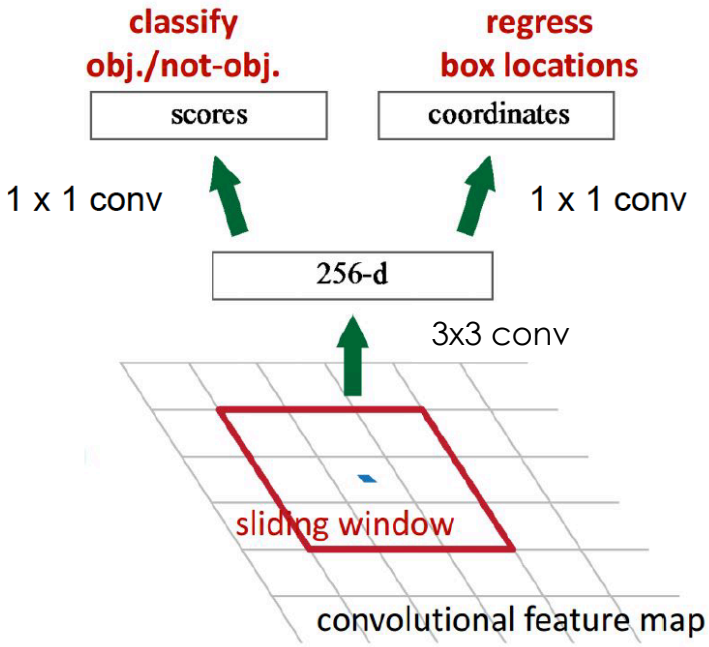
Faster R-CNN: Region Proposal Network

Slide a small window on the feature map

Build a small network for:
- classifying object or not-object, and
- regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



Slide credit: Kaiming He

Slide credit: CS231n
(https://goo.gl/QSA36t)

RPN

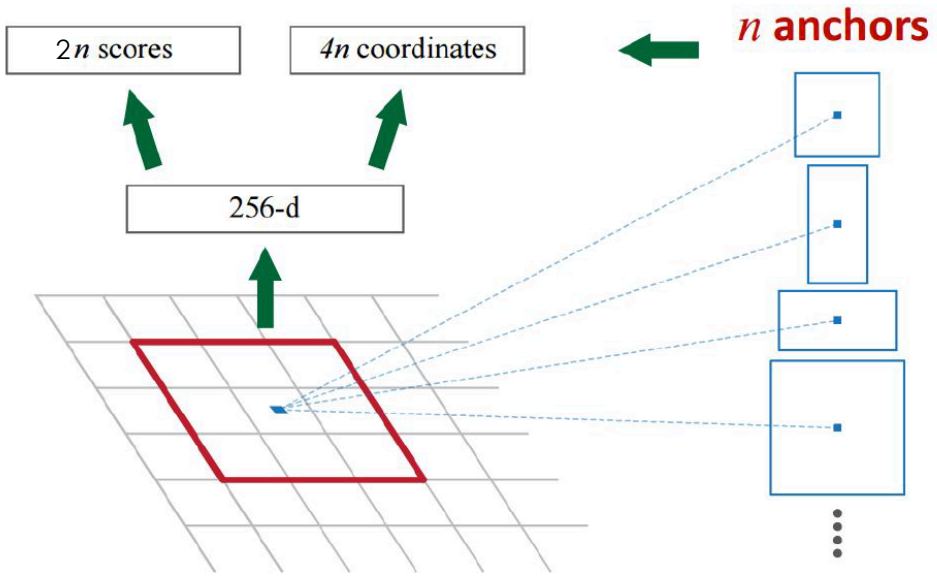
Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

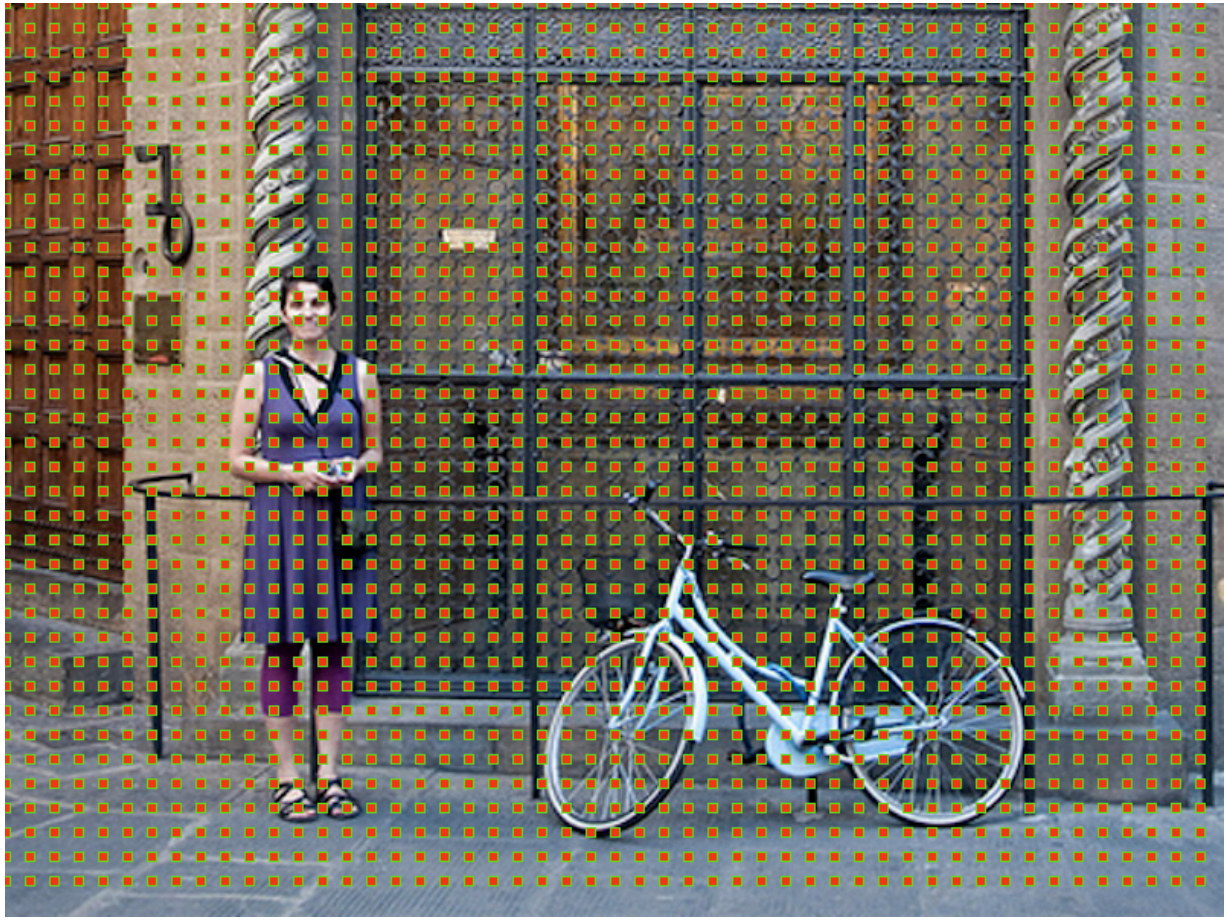
Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object



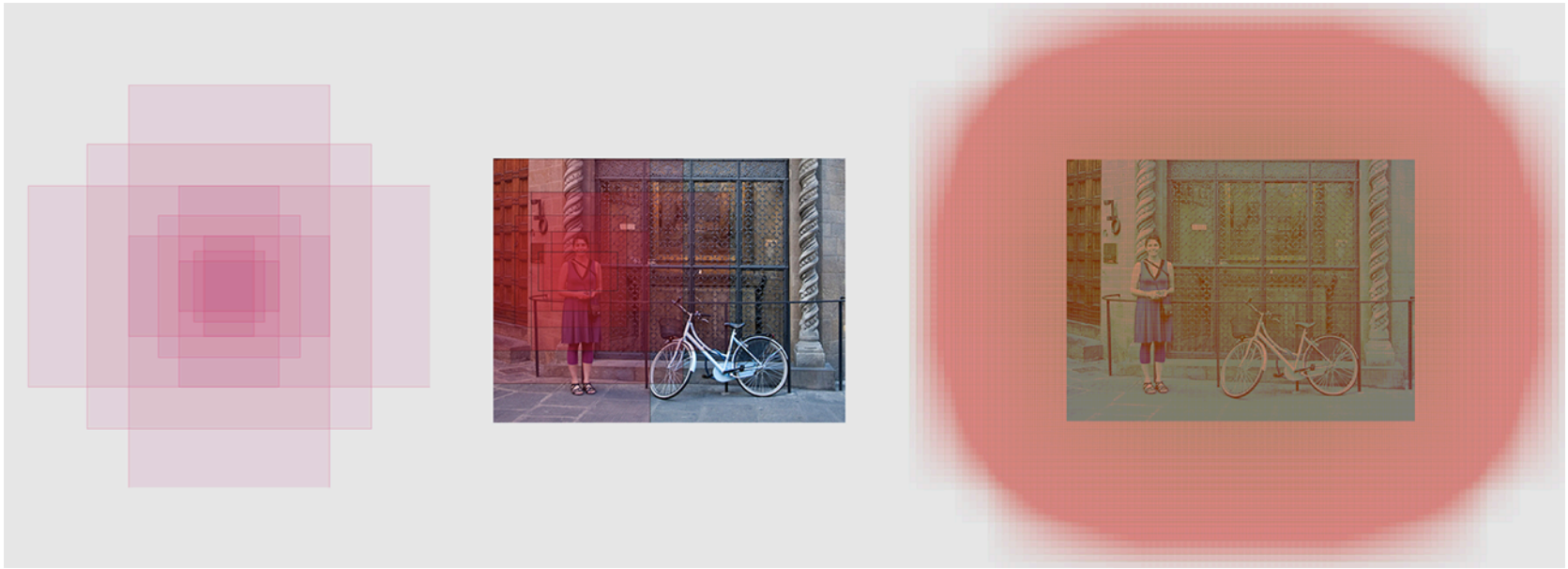
Slide credit: CS231n (https://goo.gl/QSA36t)

RPN – Anchor center



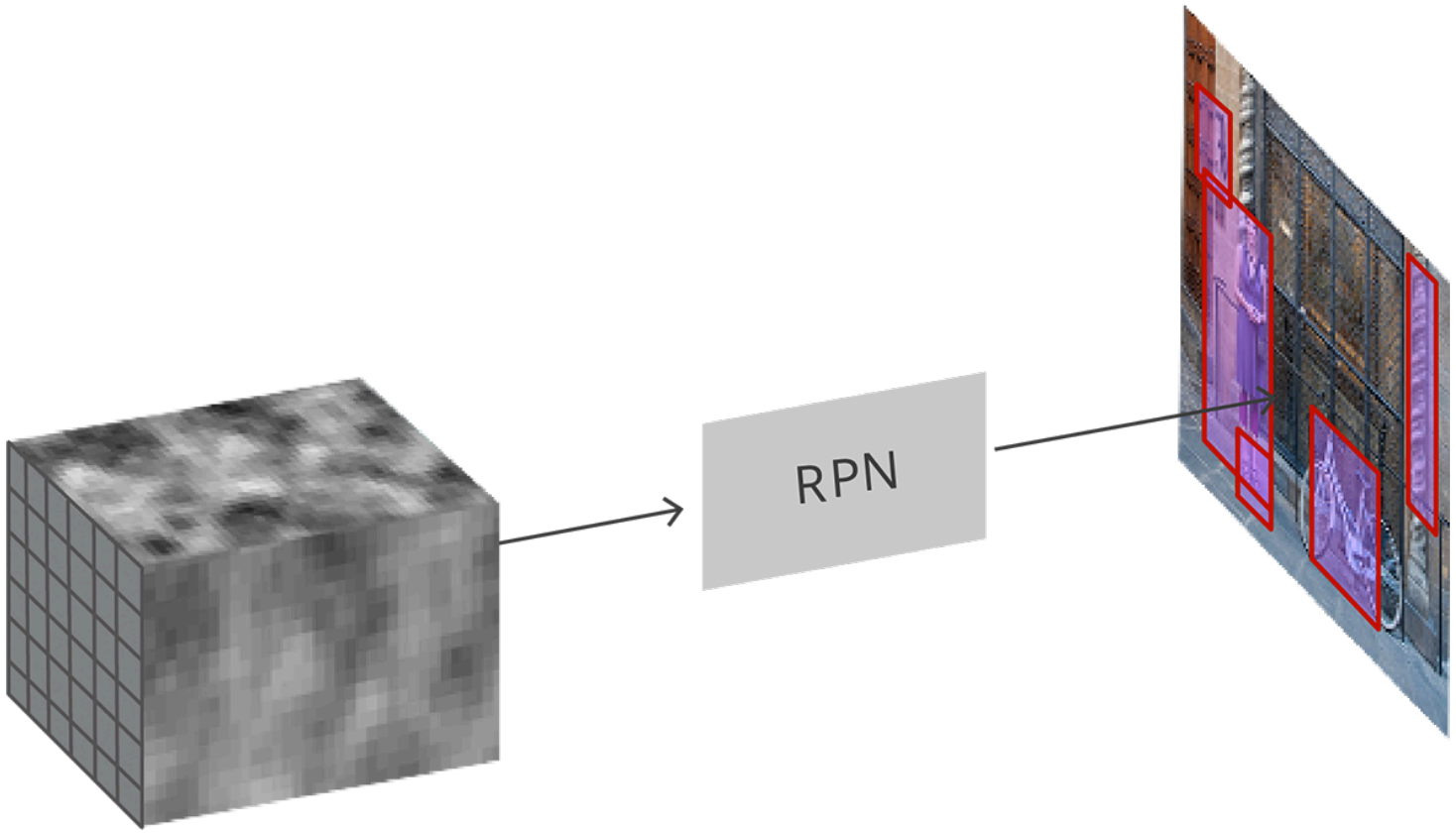
Slide credit: tryo labs
(https://goo.gl/xMty4w)

RPN – Anchors



Slide credit: tryo labs (https://goo.gl/xMty4w)

RPN

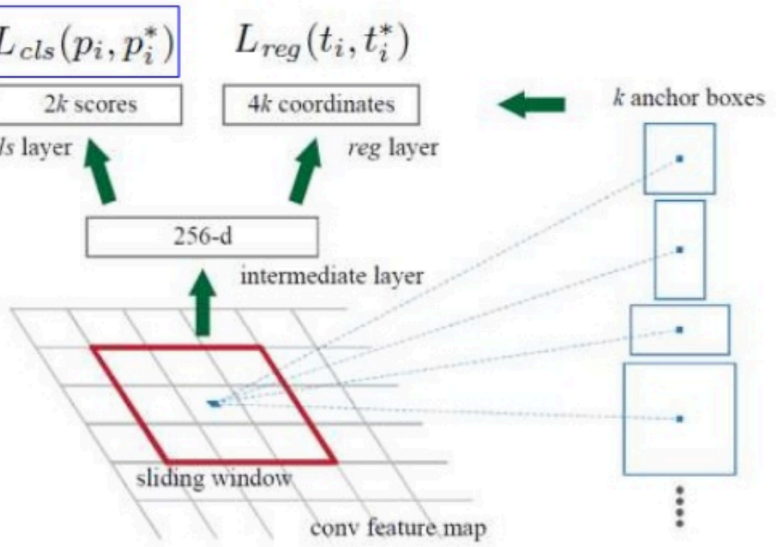


Slide credit: tryo labs
(https://goo.gl/xMty4w)

Faster R-CNN

RPN: Loss Function

- 2 class Softmax cross entropy loss
- Discriminative training:
  - pi* = 1 if IoU > 0.7
  - pi* = 0 if IoU < 0.3
  - otherwise, do not contribute to loss



$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Faster R-CNN

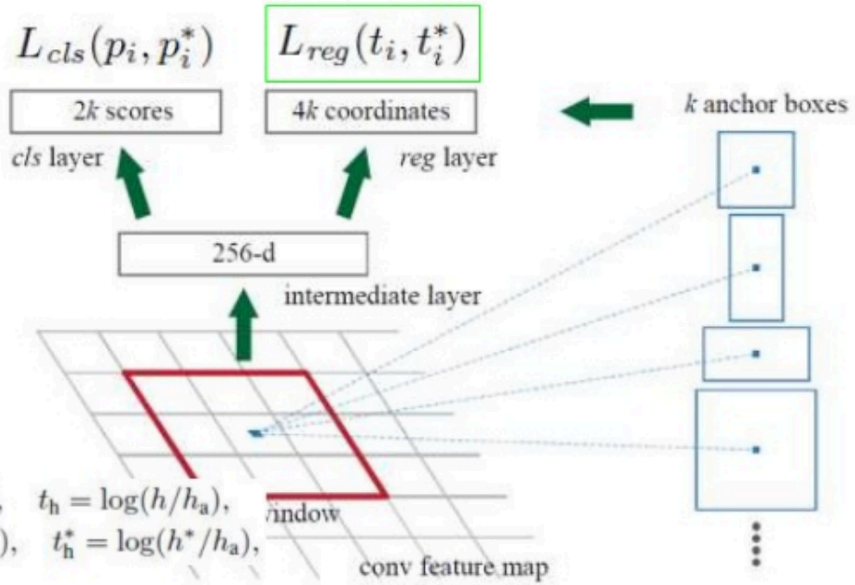
RPN: Loss Function

$$L_{\text{loc}}(t, t^*) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i, t_i^*), \quad (2)$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \quad (3)$$

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$



$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

only positive sample contribute to reg. Loss

Training RPN Mini-batch

- We assign a **positive label** to two kinds of anchors:
- (i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box,
- (ii) an anchor that has an IoU overlap higher than **0.7** with any ground-truth box.
- We assign a **negative label** to a non-positive anchor if its IoU ratio is lower than **0.3** for all ground-truth boxes.
- Anchors that are **neither positive nor negative** do not contribute to the training objective.

Training RPN Mini-batch

- It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate.
- Instead, we randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled **positive and negative anchors have a ratio of up to 1:1**.
- If there are fewer than 128 positive samples in an image, we pad the mini-batch with negative ones.

Faster R-CNN

How to train faster R-CNN ?

1. train RPN with ImageNet pre-trained model
2. train fast R-CNN using proposal generated by 1. [no params. sharing]
3. use conv trained by 2. to initialize model, fix shared conv, update RPN
4. fix shared conv, fine-tune fc

Faster R-CNN

Faster R-CNN: Training

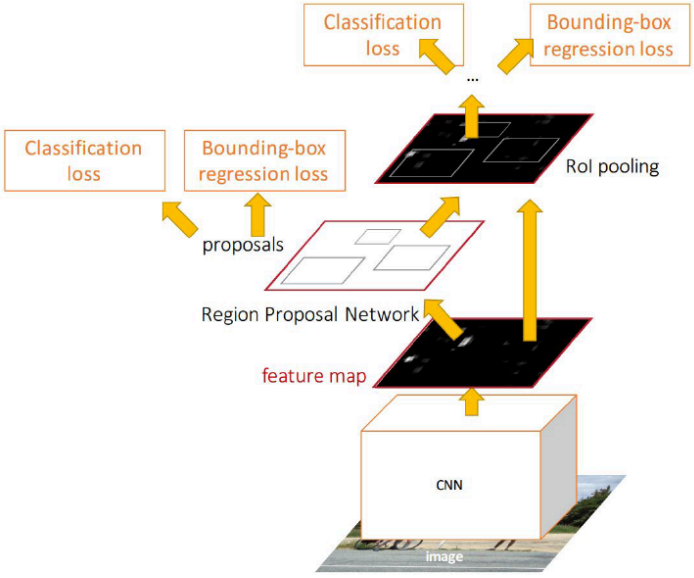
In the paper: Ugly pipeline

- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!

One network, four losses

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



Slide credit: Ross Girschick

Slide credit: CS231n (https://goo.gl/QSA36t)

Faster R-CNN

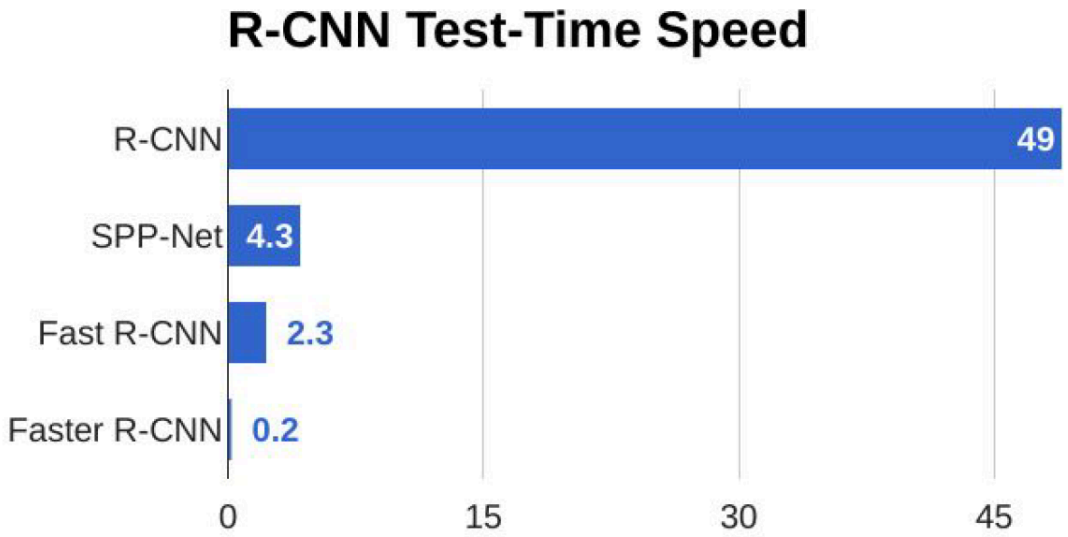
Faster R-CNN: Results

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Slide credit: CS231n (https://goo.gl/QSA36t)

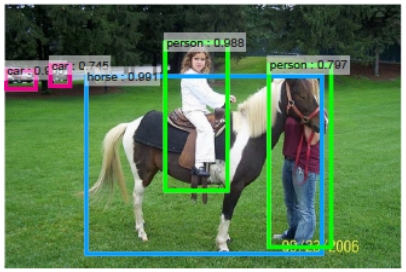
Faster R-CNN

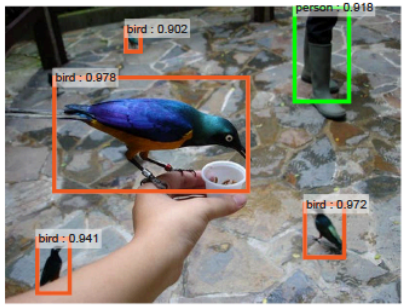
Fast**er** R-CNN:
Make CNN do proposals!

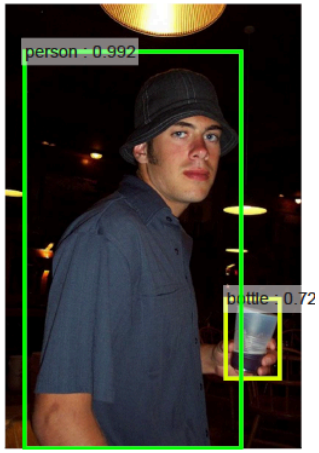


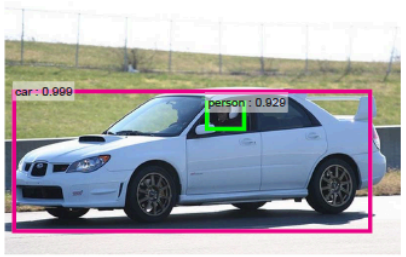
Slide credit: CS231n
(https://goo.gl/QSA36t)

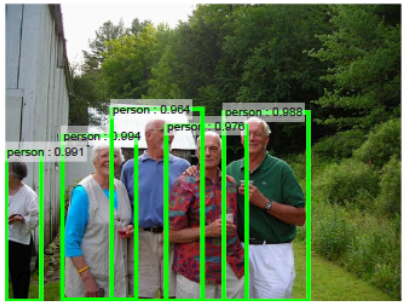
PASCAL VOC 2007 test results

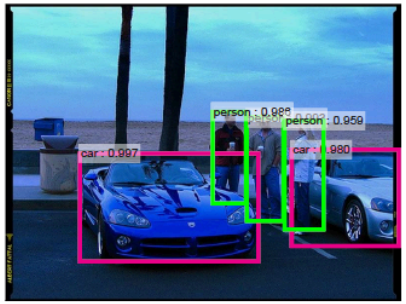


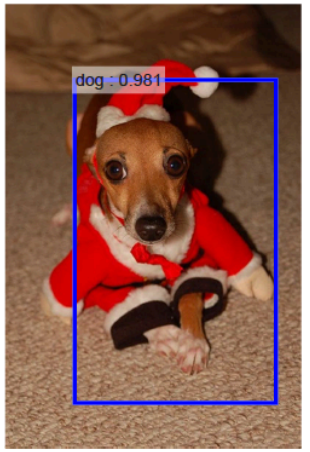


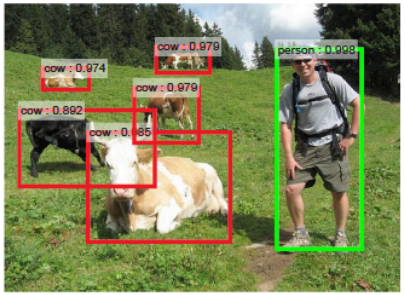




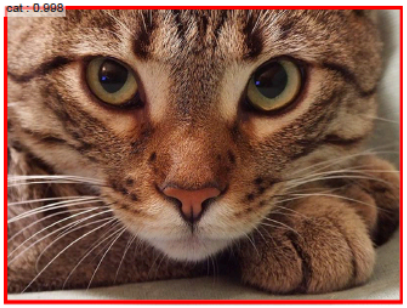


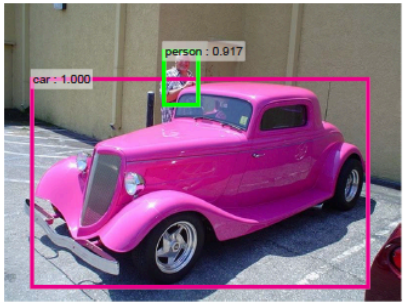


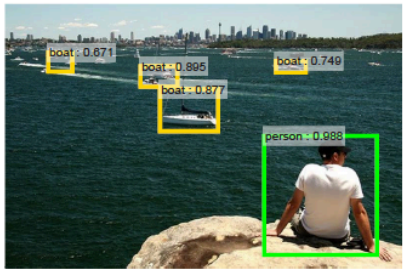


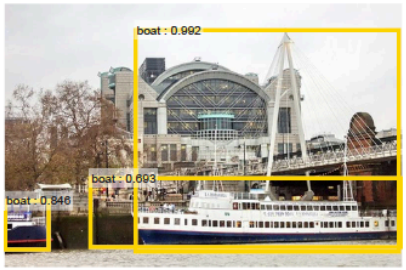


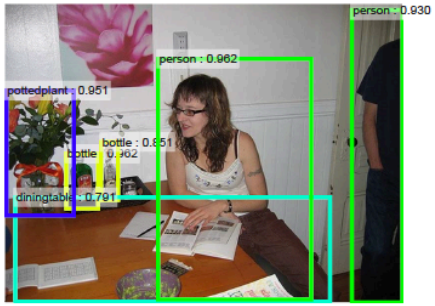
PASCAL VOC 2007 test results

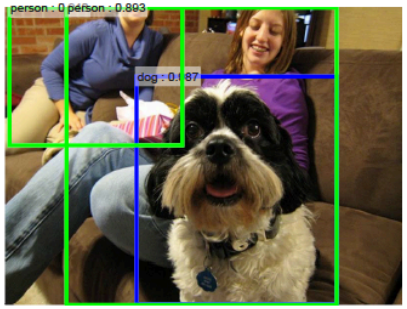


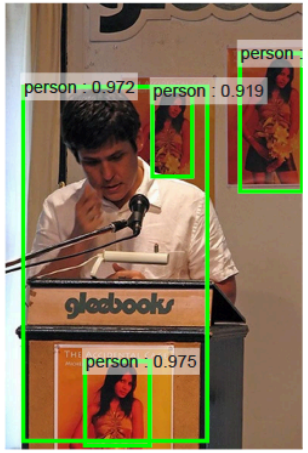


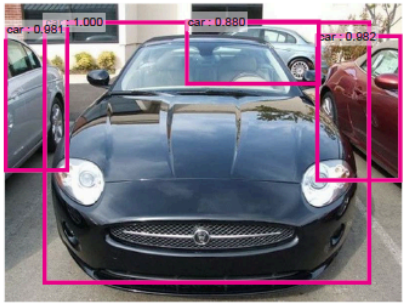


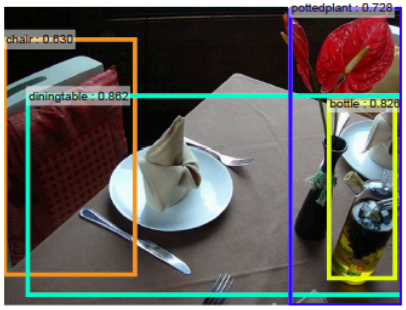


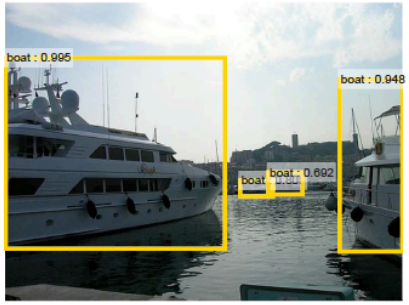




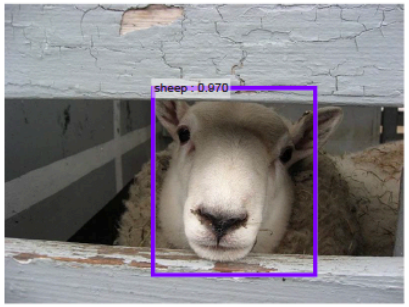


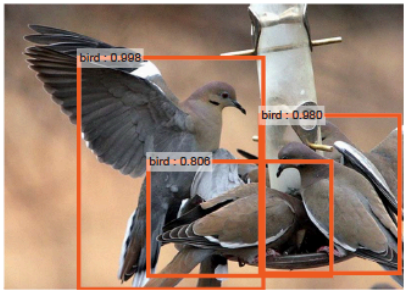


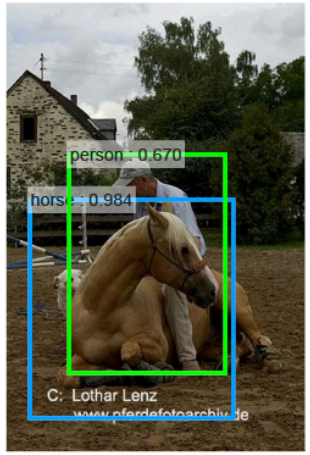


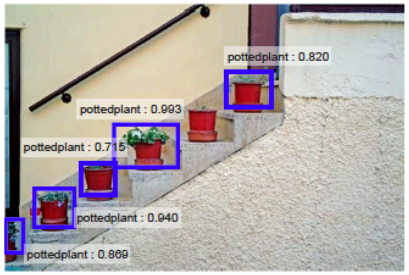


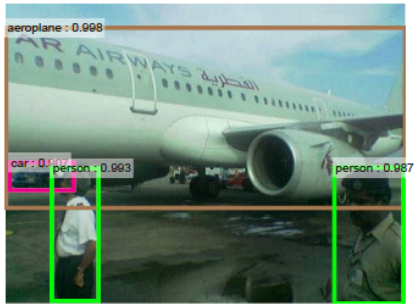
PASCAL VOC 2007 test results

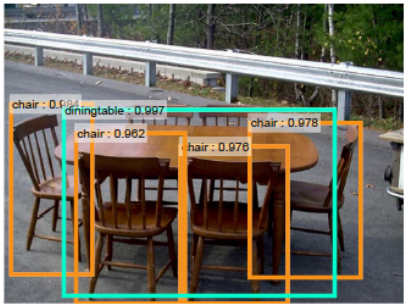


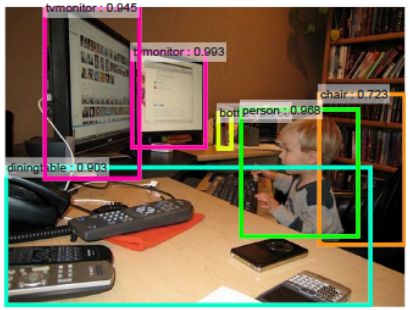


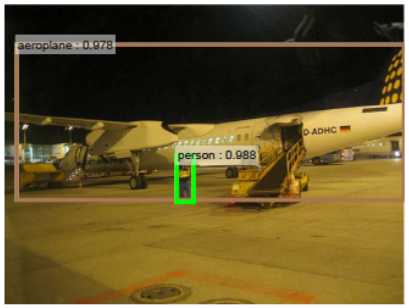


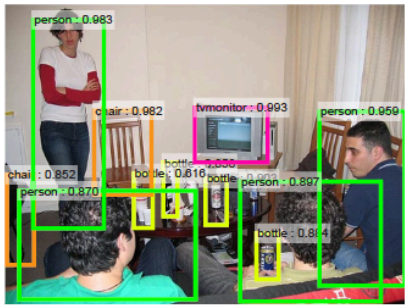




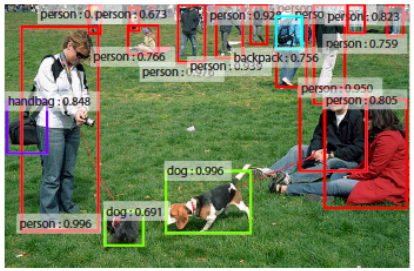


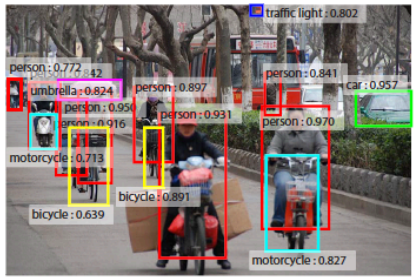


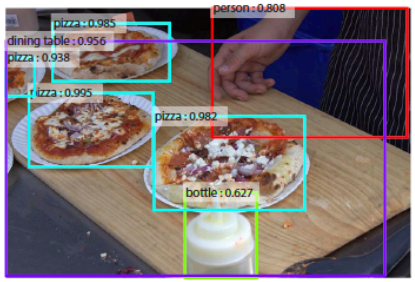


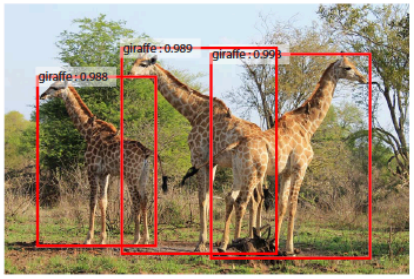


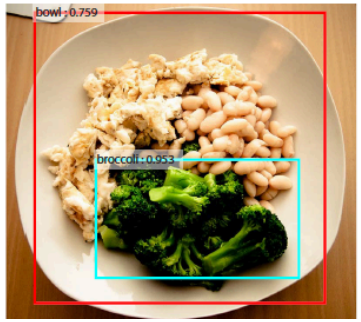
MS COCO test-dev results

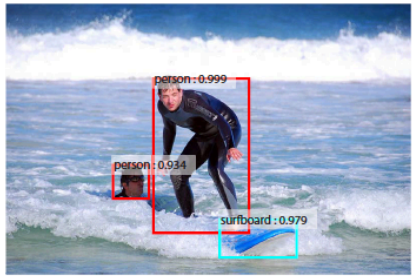




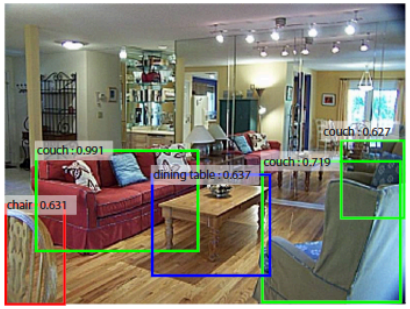


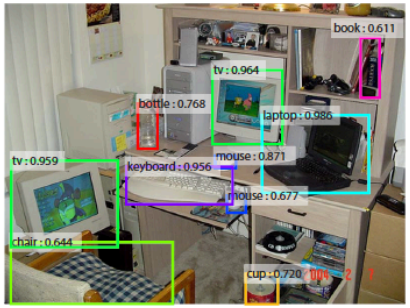


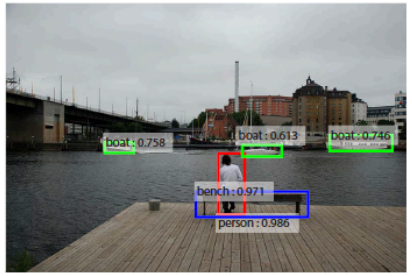


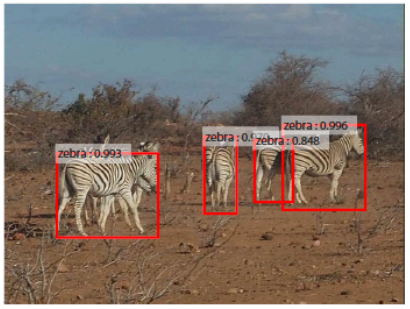


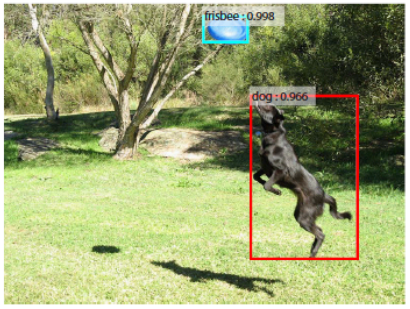
MS COCO test-dev results

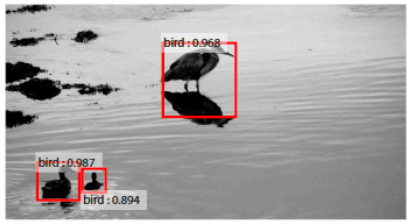


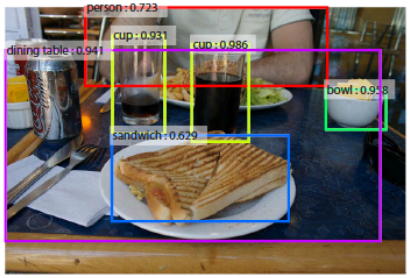




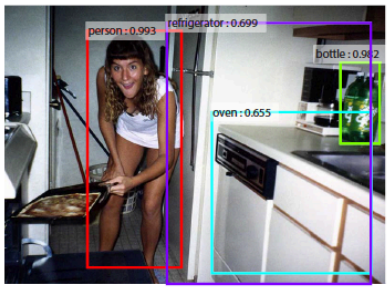


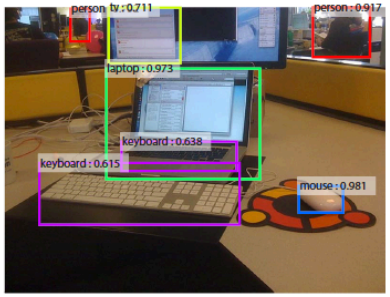


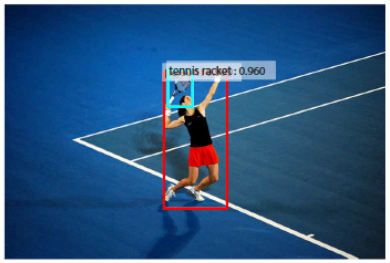


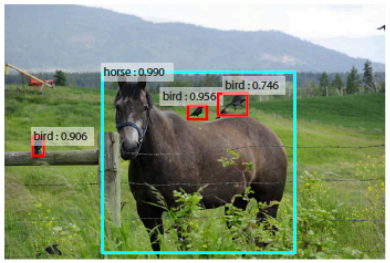


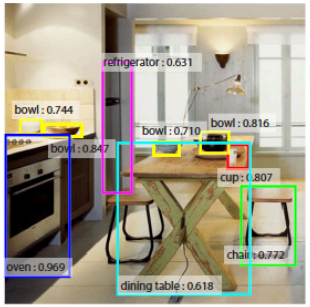
MS COCO test-dev results

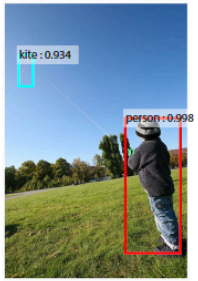


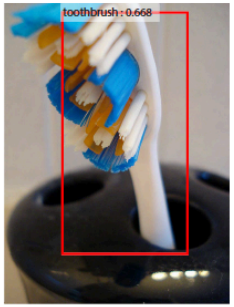










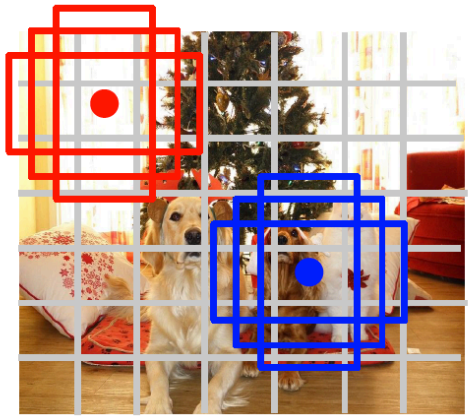


YOLO

Detection without Proposals: YOLO / SSD

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:

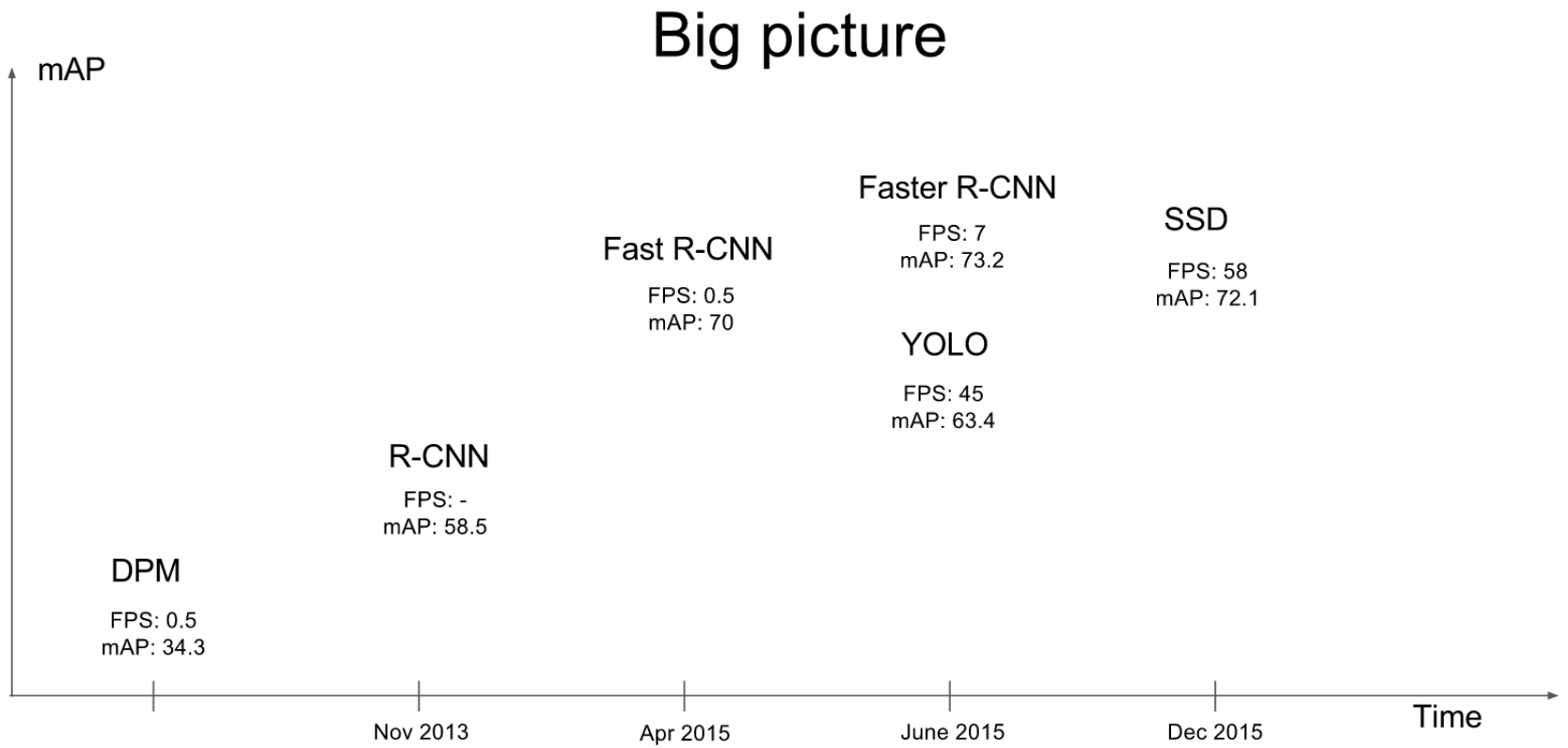
- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

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

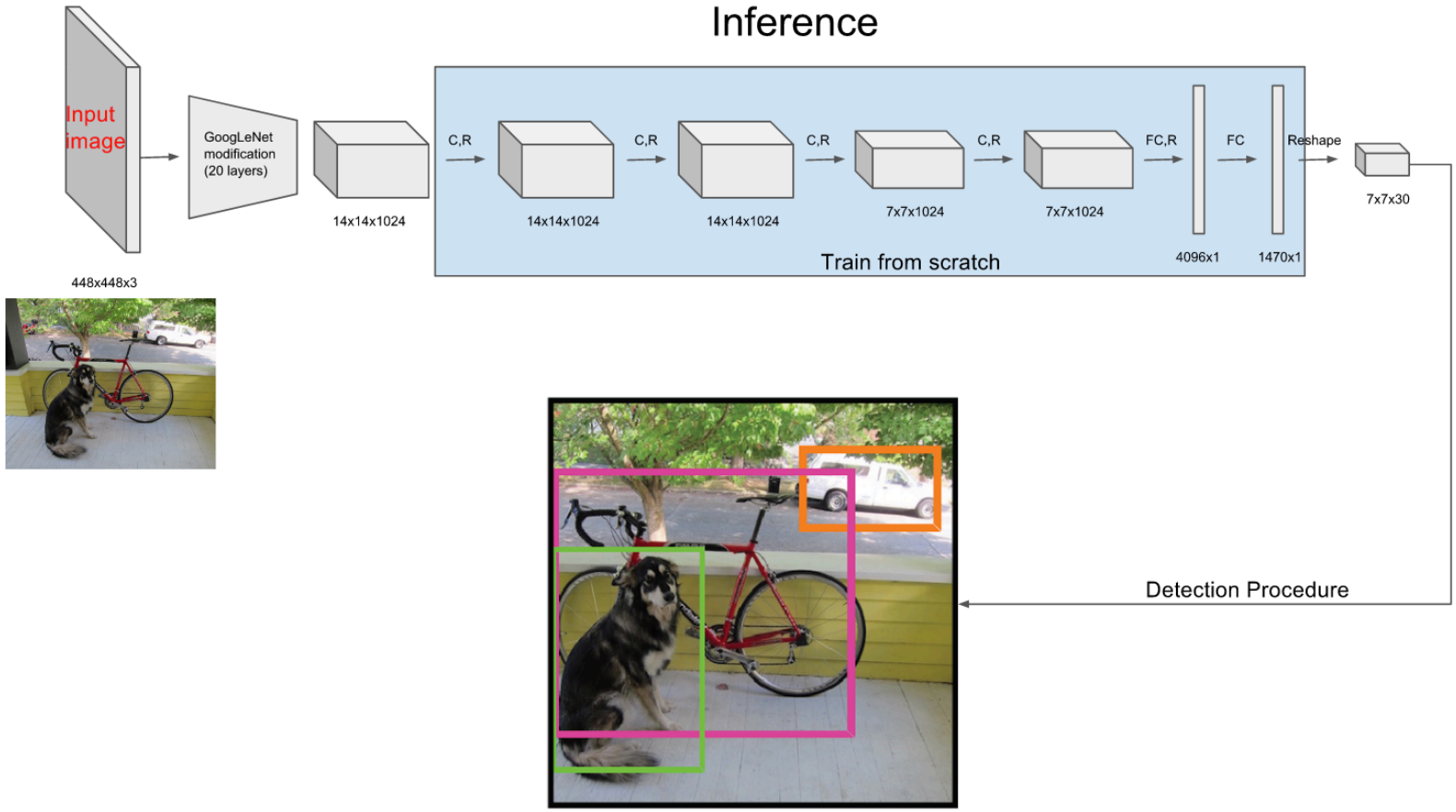
Slide credit: CS231n (https://goo.gl/QSA36t)

YOLO



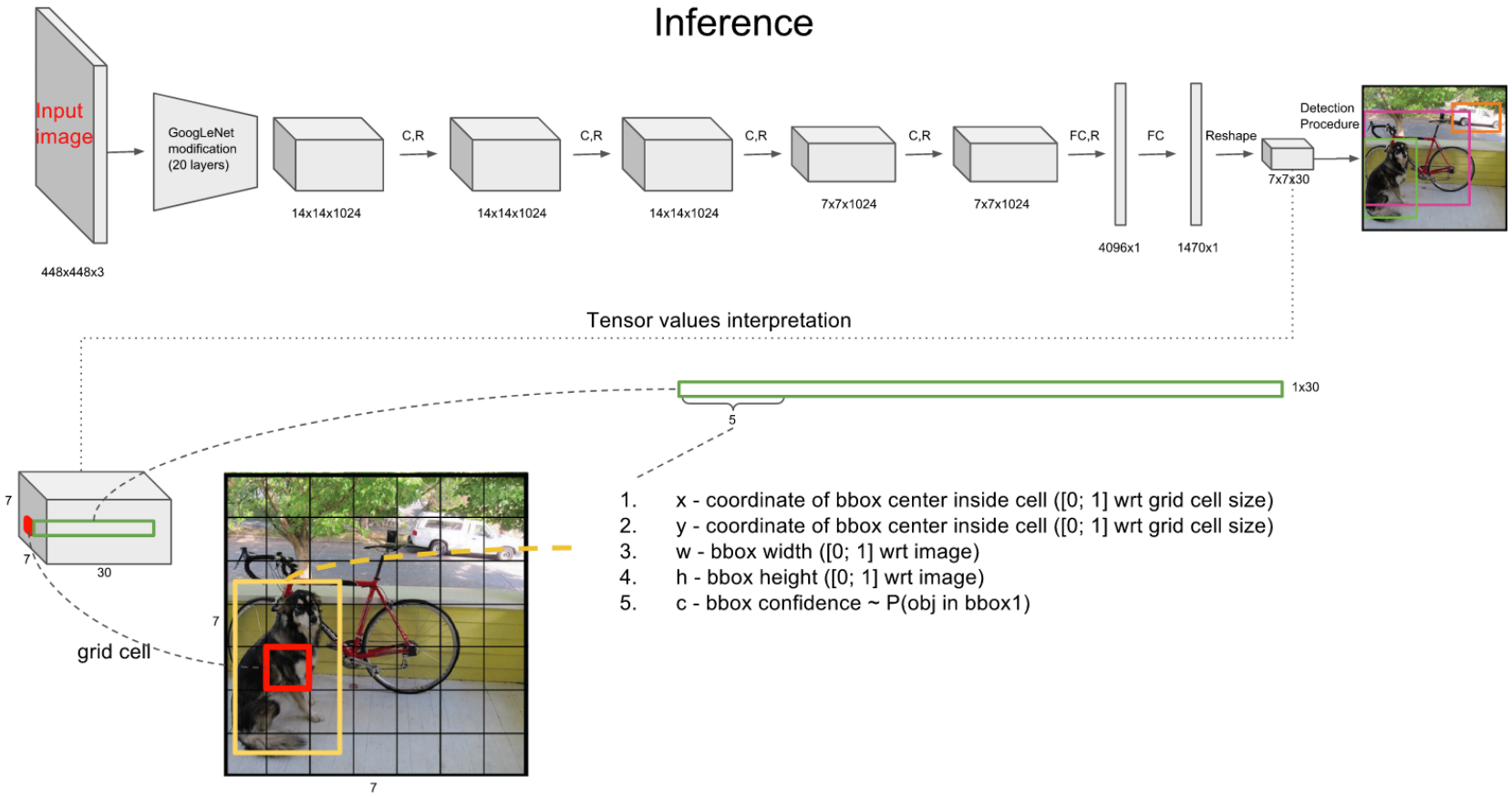
Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO



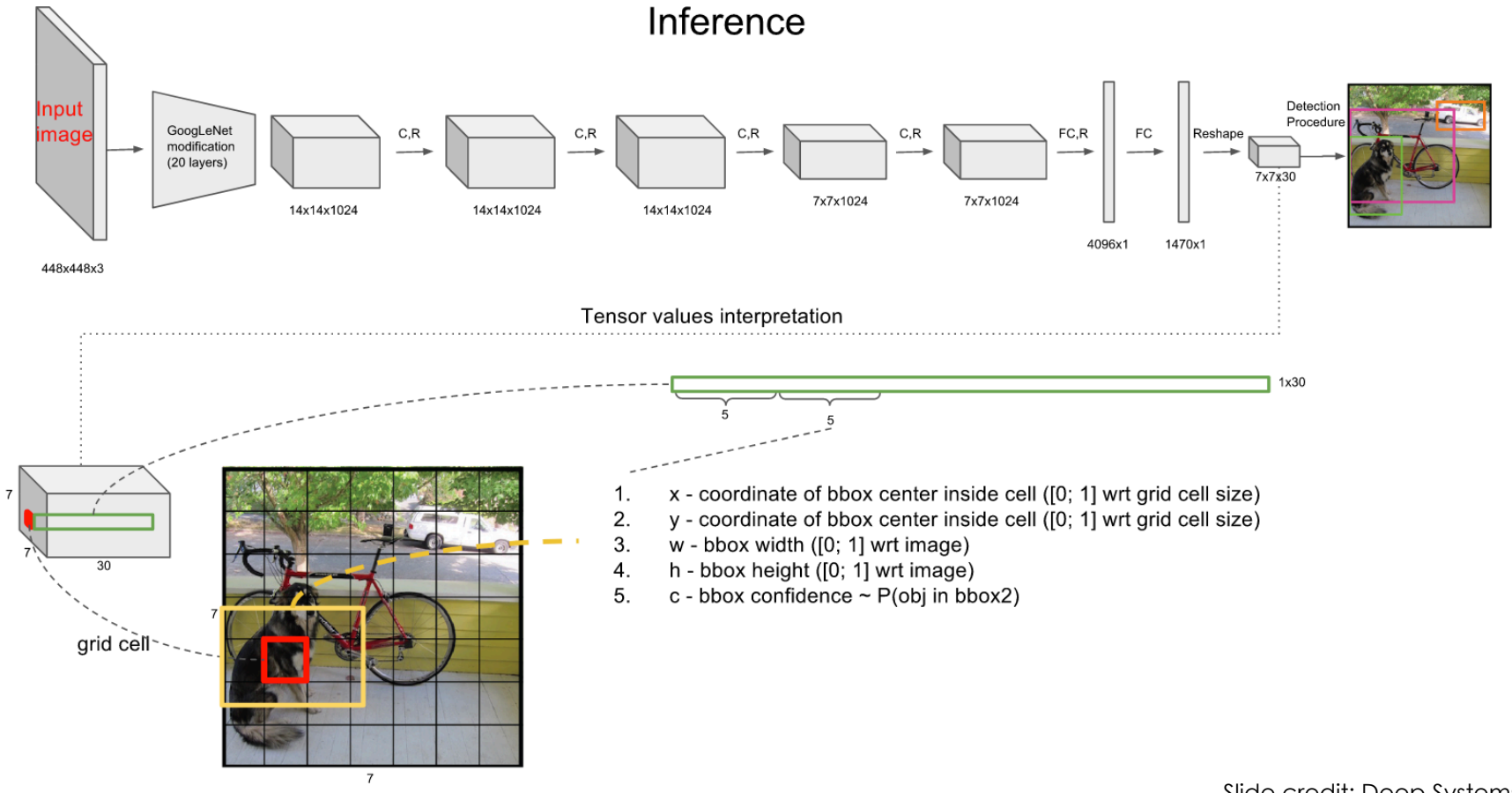
Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO



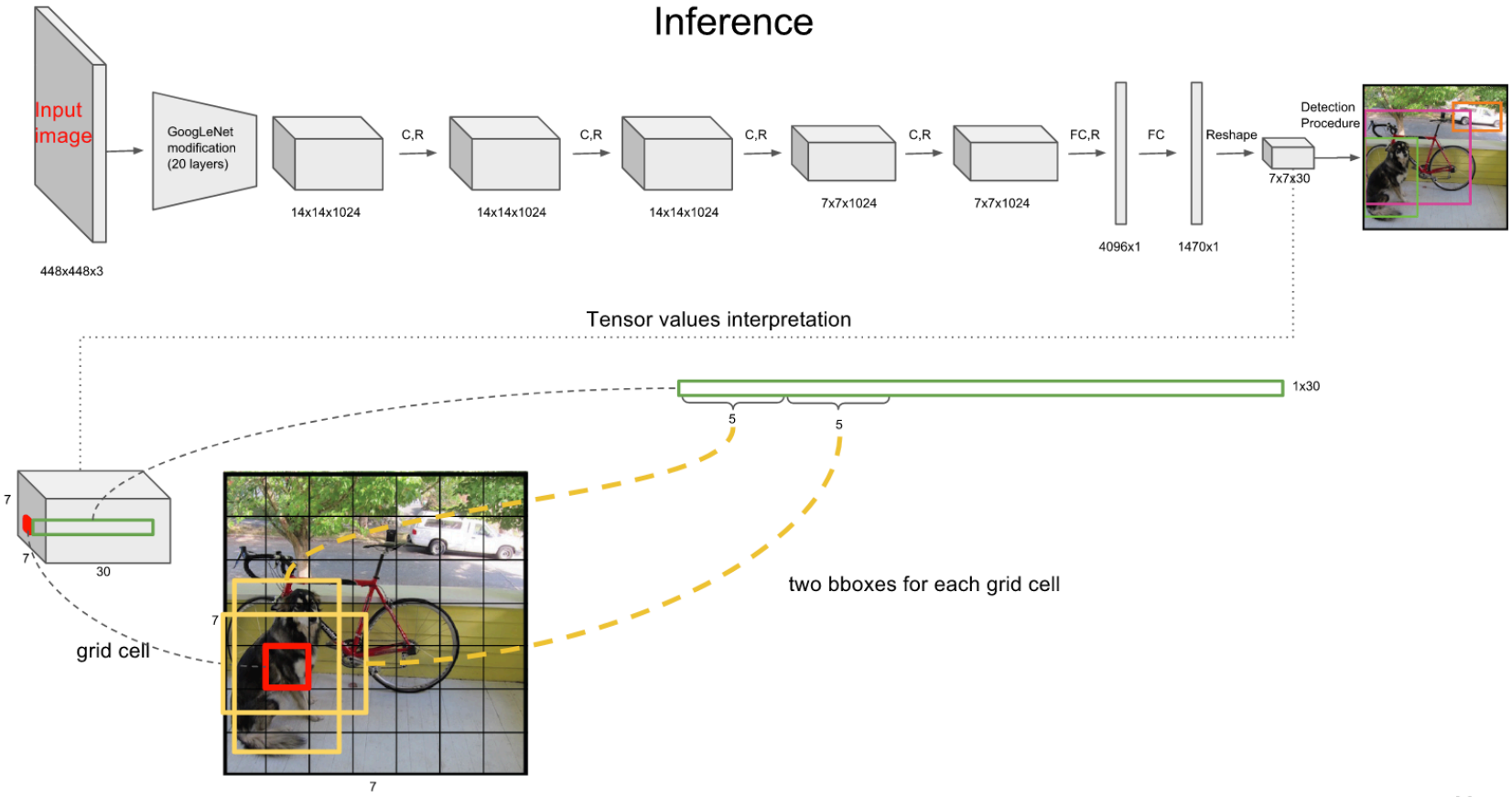
Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO



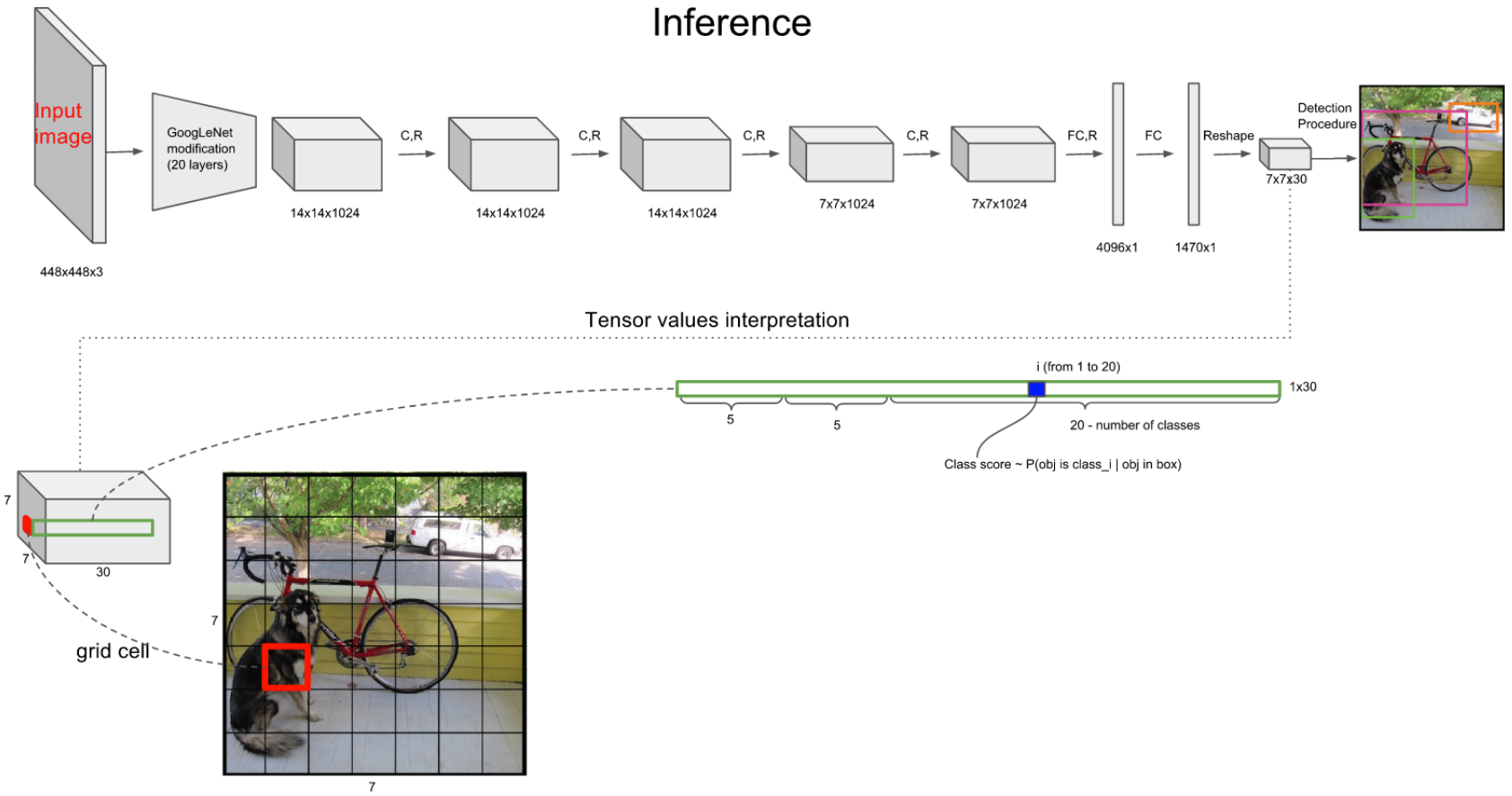
Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO



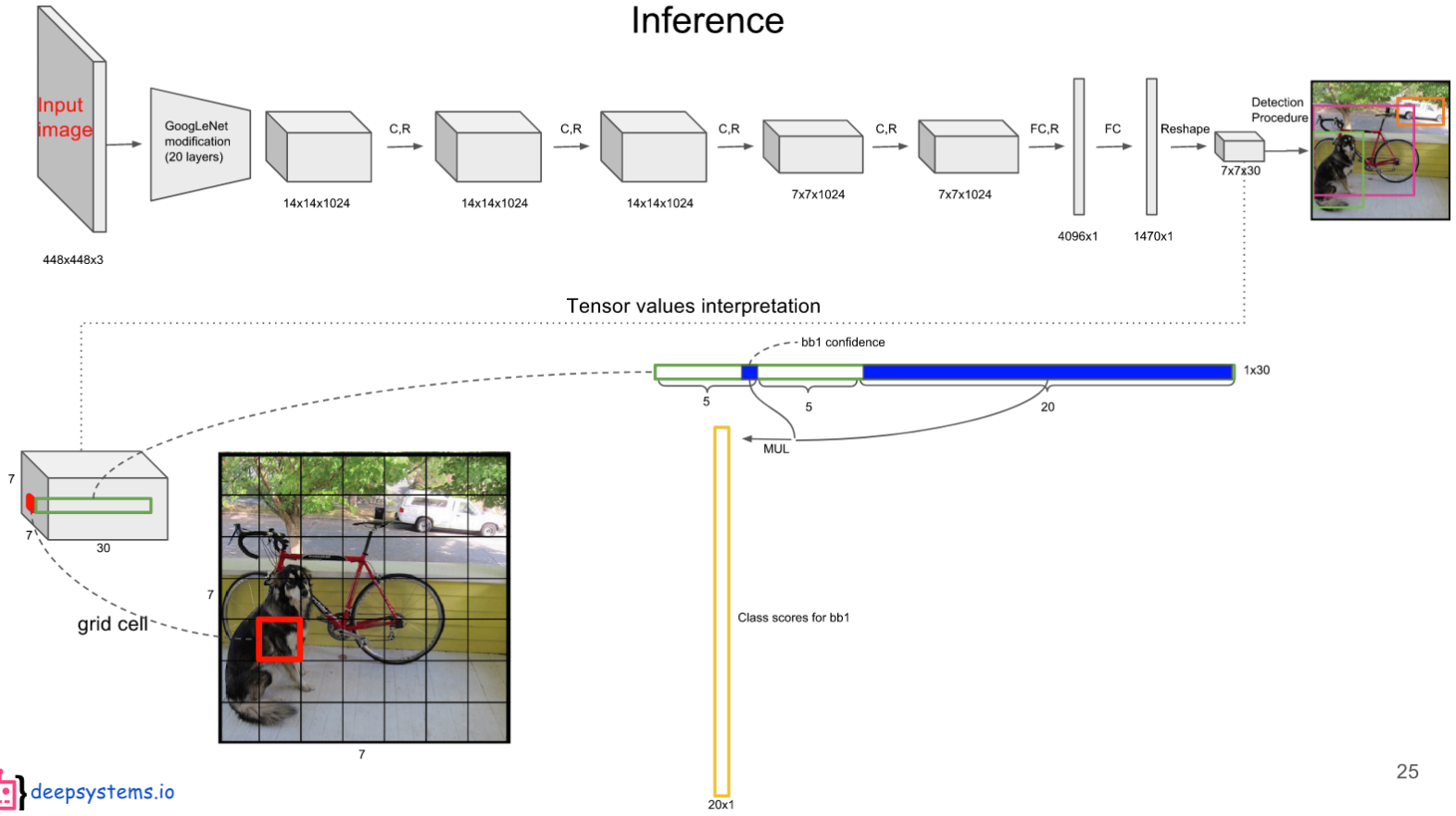
Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO



Slide credit: Deep Systems (https://goo.gl/Am4vQB)

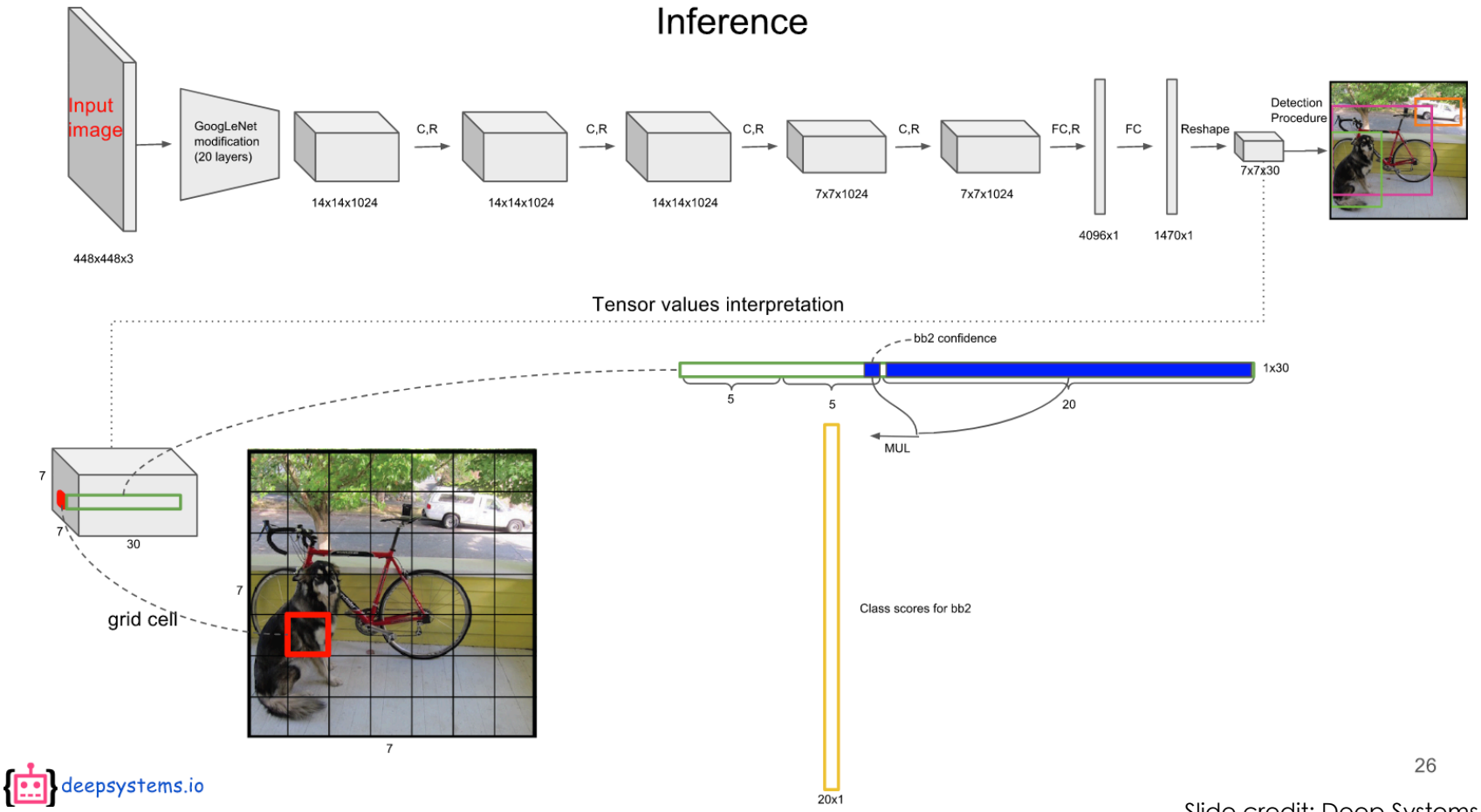
YOLO



deepsystems.io

Slide credit: Deep Systems (https://goo.gl/Am4vQB)

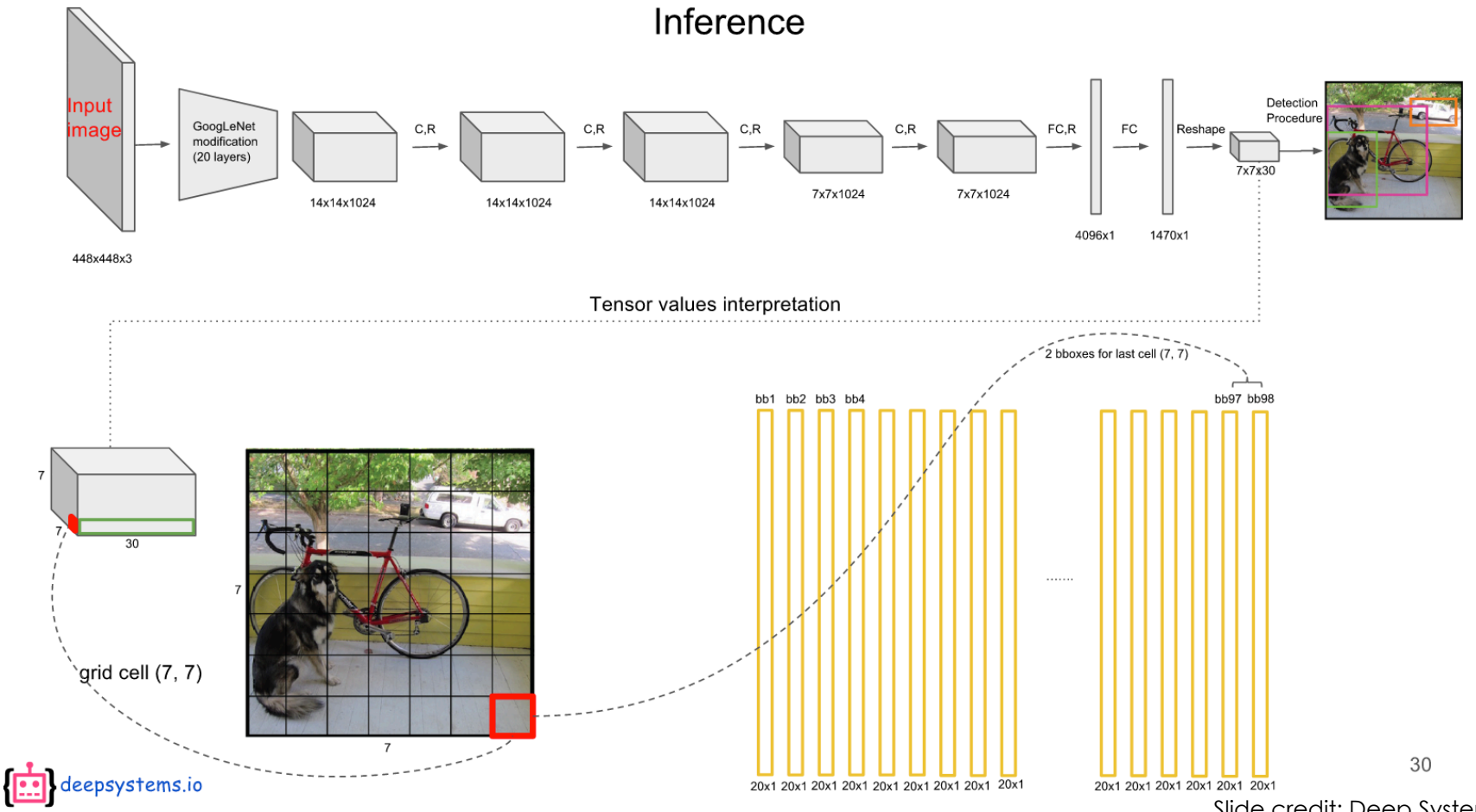
YOLO



deepsystems.io

Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO

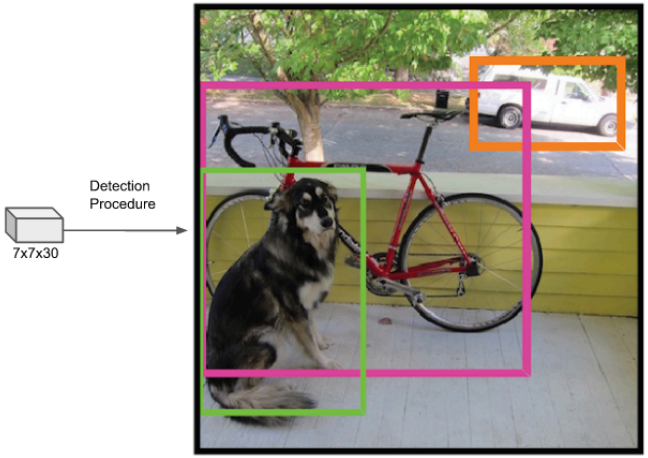


30

Slide credit: Deep Systems (https://goo.gl/Am4vQB)

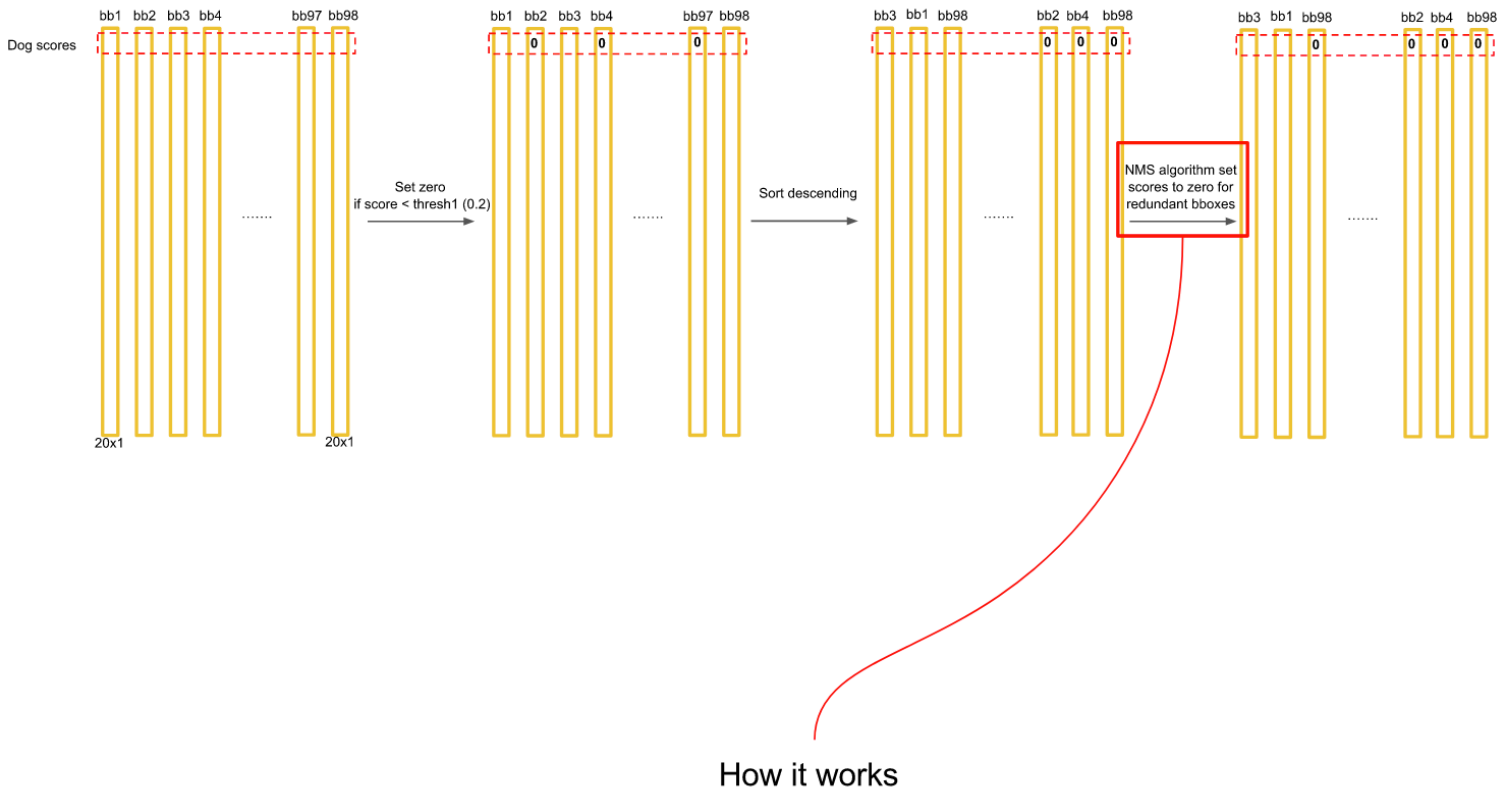
YOLO

Look at detection procedure



Slide credit: Deep Systems
(https://goo.gl/Am4vQB)

YOLO



Slide credit: Deep Systems
(https://goo.gl/Am4vQB)

YOLO

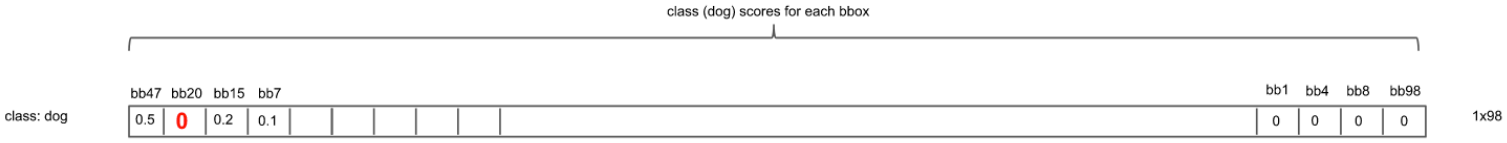
Non-Maximum Suppression: intuition

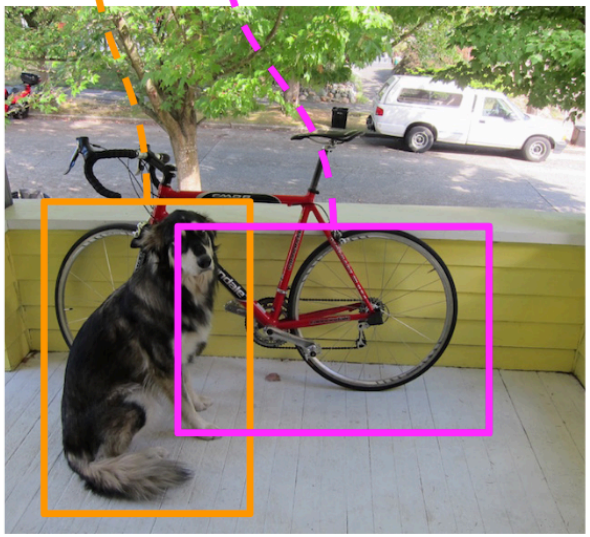
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO

Non-Maximum Suppression: intuition





Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

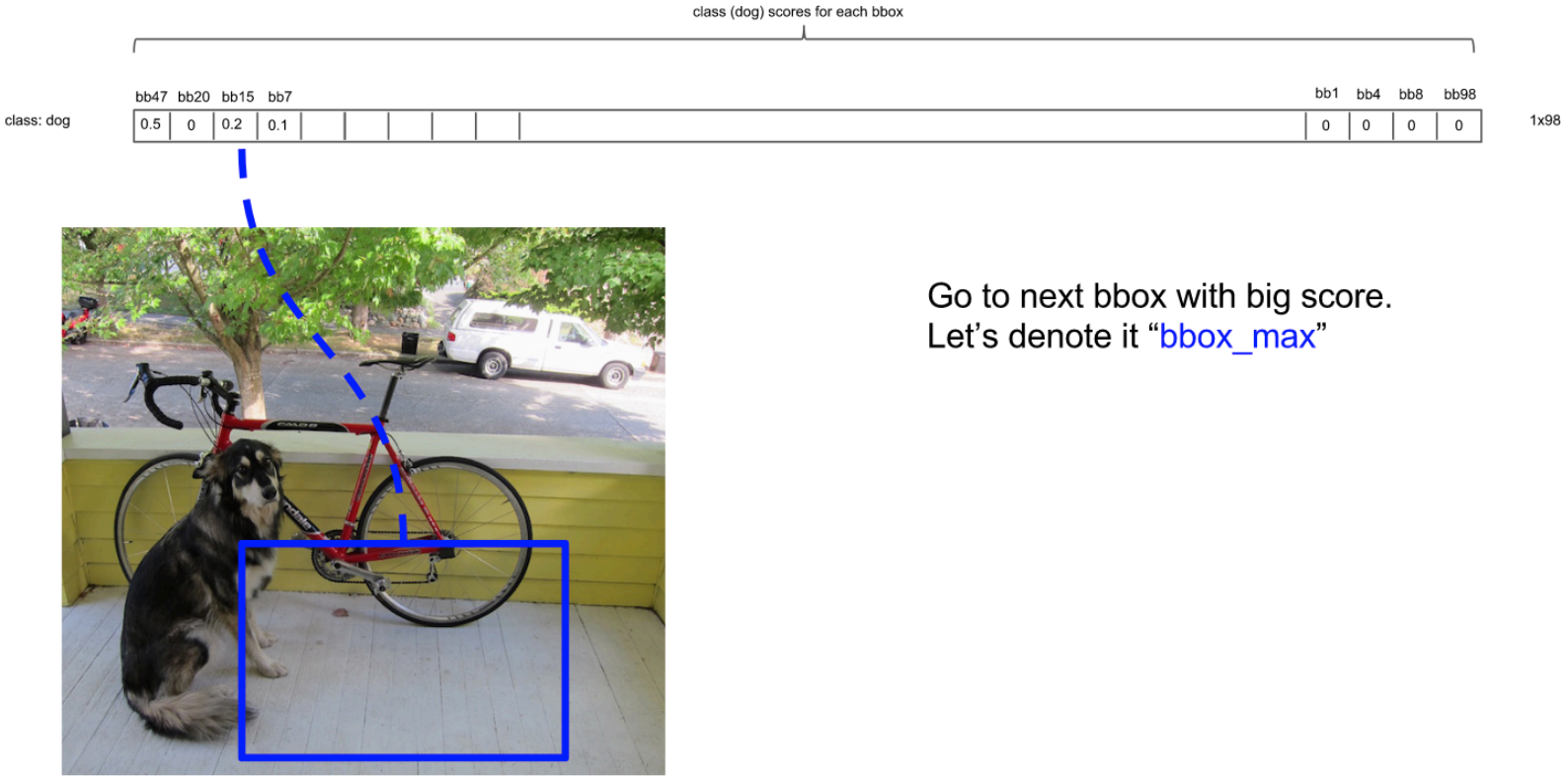
In this case: continue.

Do this procedure for other “bbox_cur”. After that ...

Slide credit: Deep Systems (https://goo.gl/Am4vQB)

YOLO

Non-Maximum Suppression: intuition

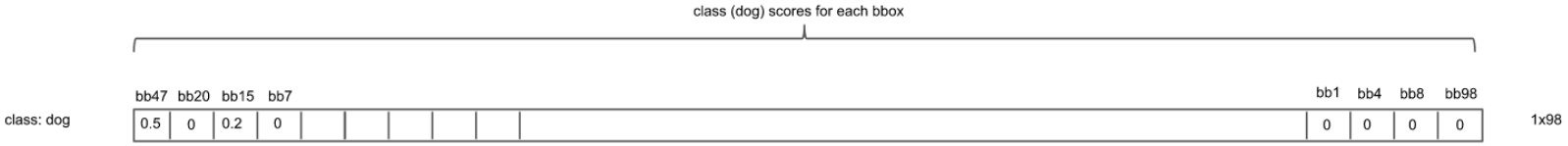


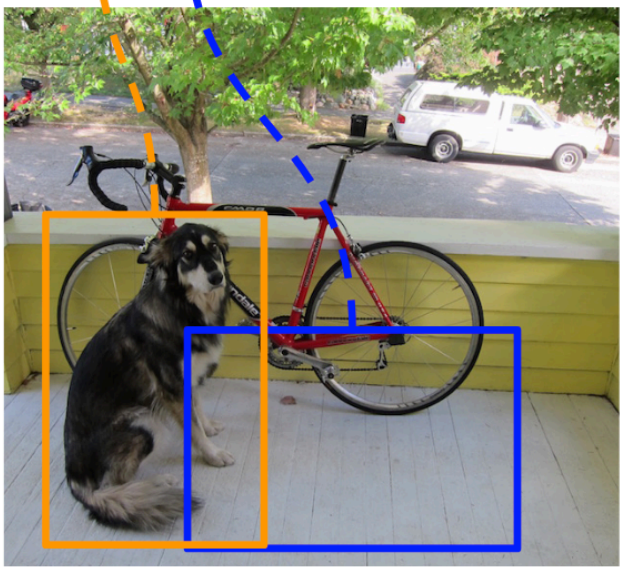
Go to next bbox with big score.
Let's denote it "bbox_max"

Slide credit: Deep Systems
(https://goo.gl/Am4vQB)

YOLO

Non-Maximum Suppression: intuition



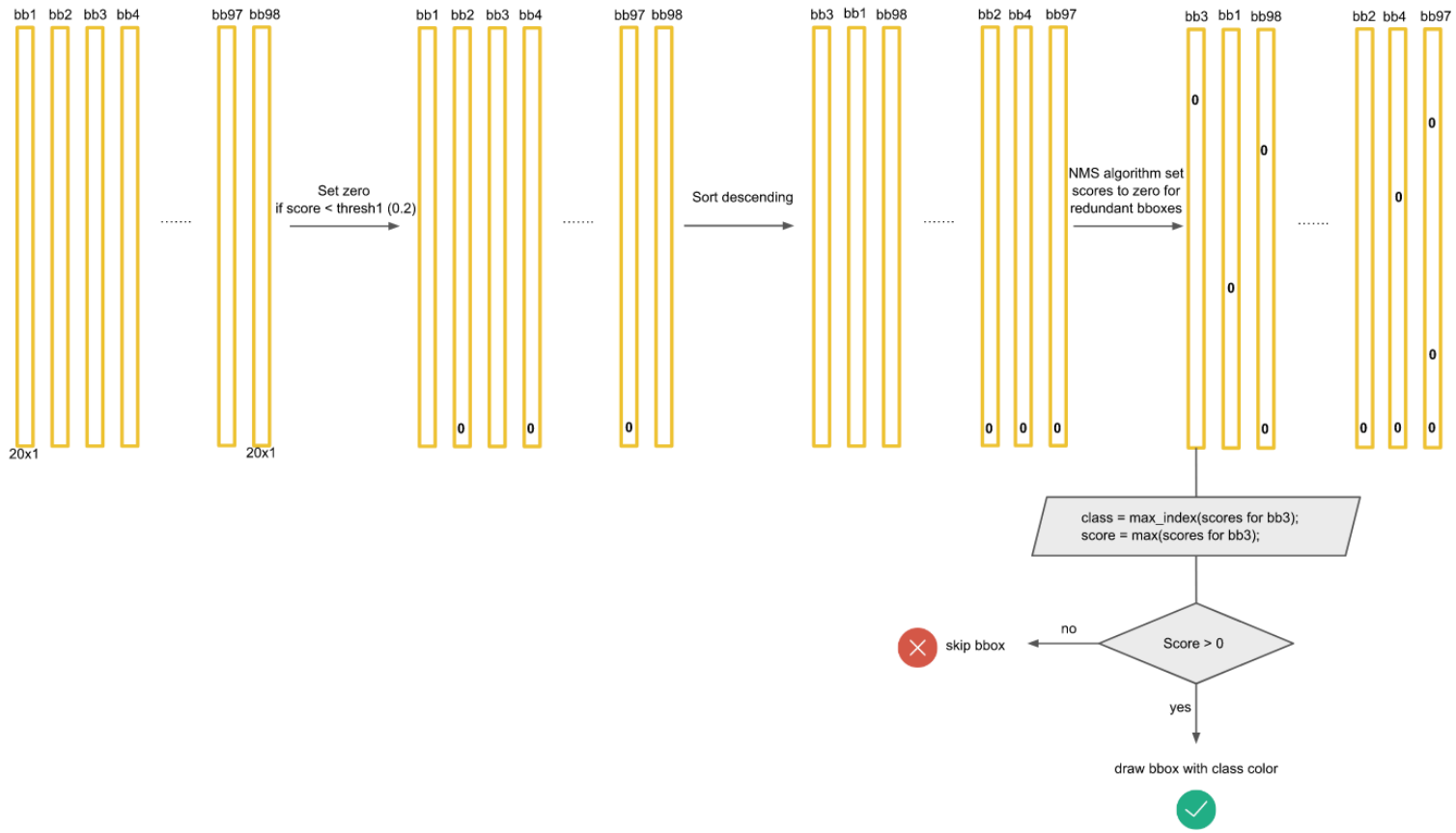


After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.

Slide credit: Deep Systems (https://goo.gl/Am4vQB)

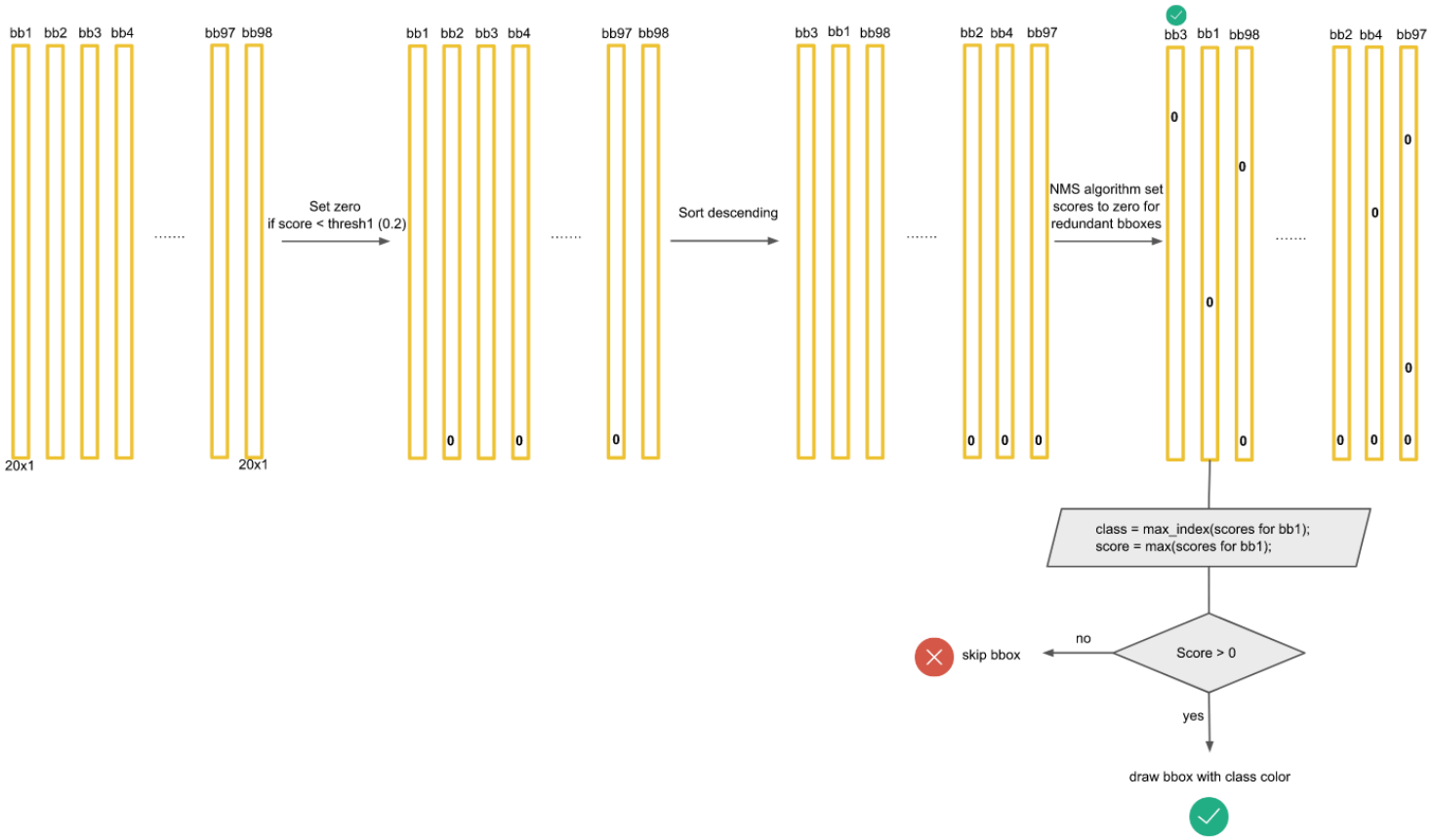
Non-Maximum Suppression

YOLO



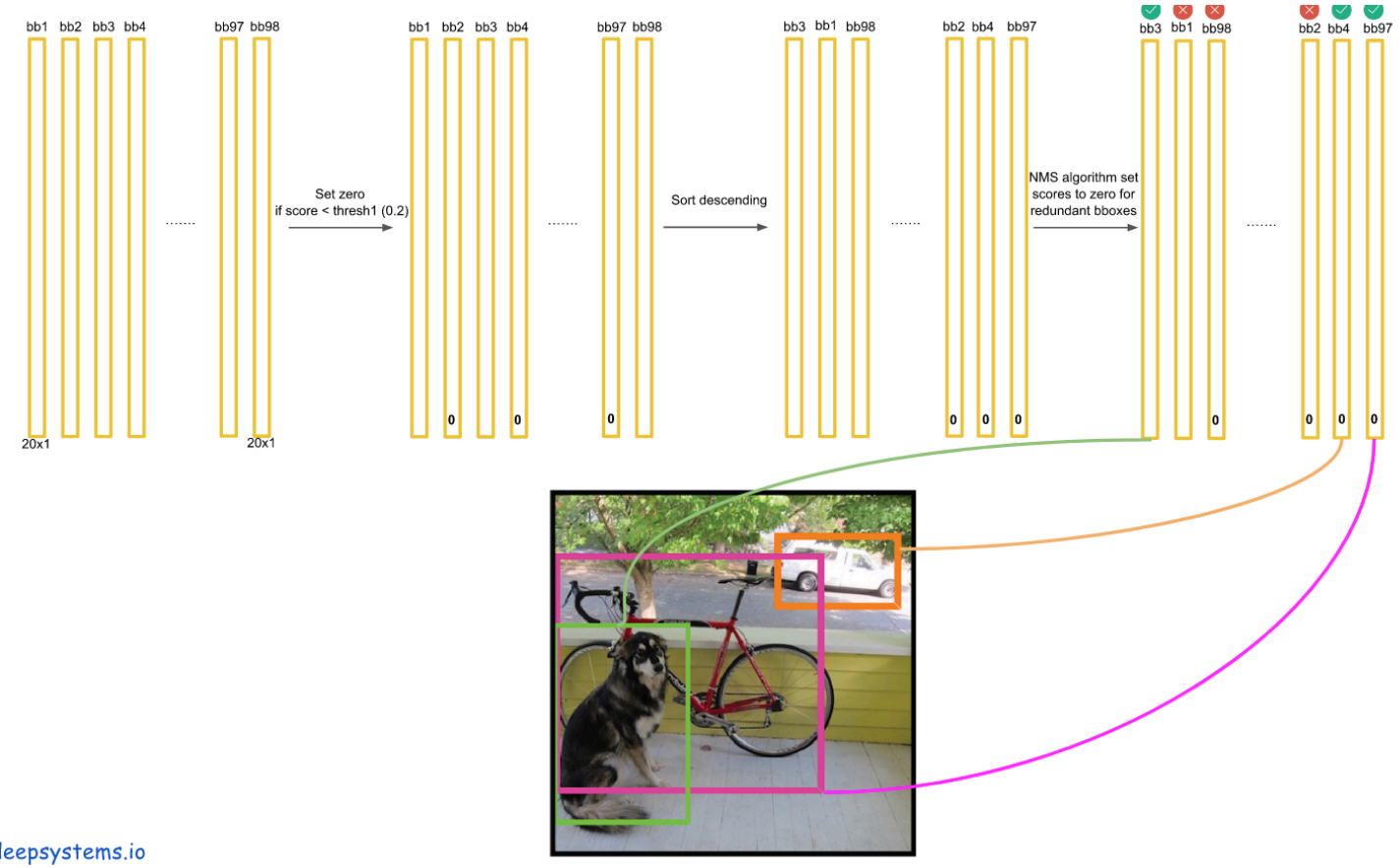
Slide credit: Deep Systems
(https://goo.gl/Am4vQB)

YOLO



Slide credit: Deep Systems
(https://goo.gl/Am4vQB)

YOLO



deepsystems.io

Slide credit: Deep Systems
(https://goo.gl/Am4vQB)

YOLO – Loss Function

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

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

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \tag{3}$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \tag{4}$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \tag{5}$$

더 공부할 자료 소개

- TensorFlow Models – 구글에서 공식적으로 관리하는 TensorFlow로 최신 논문을 구현한 코드들을 모아놓은 repository

https://github.com/tensorflow/models

- 딥러닝 트렌드를 이끌어가는 연구기관들에서 출판한 논문들

DeepMind : https://deepmind.com/research/publications/

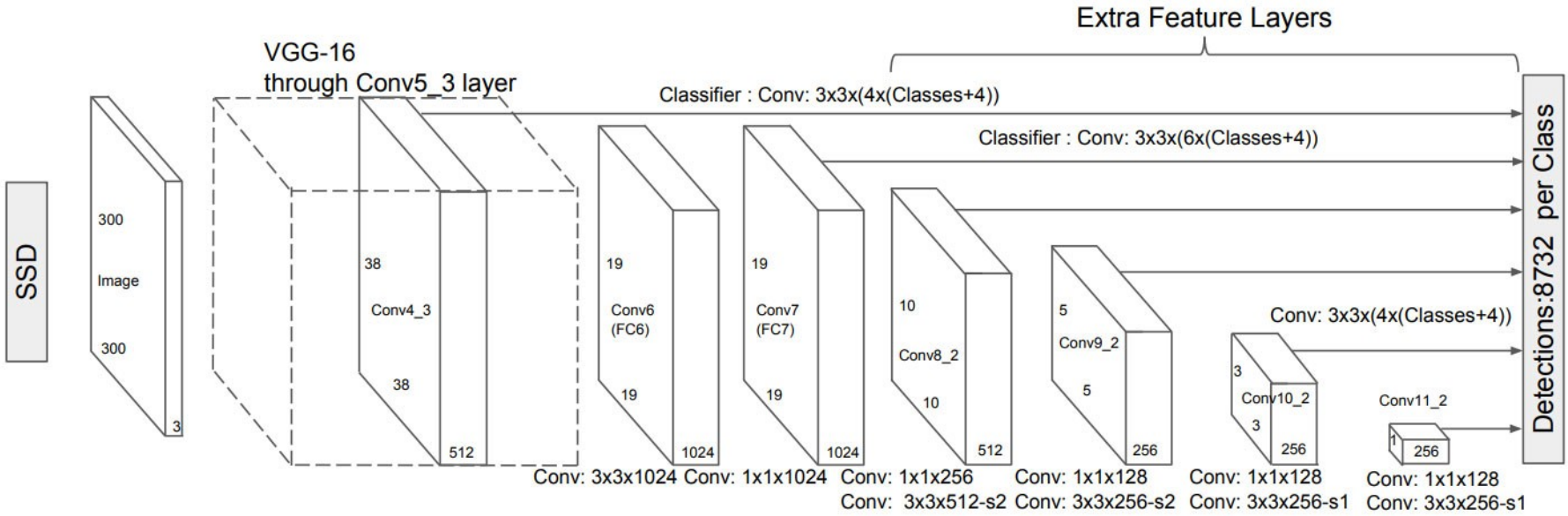
OpenAI : https://openai.com/research/#publications

FAIR : https://research.fb.com/publications/

MILA : http://www.iro.umontreal.ca/~lisa/publications2/index.php/publications

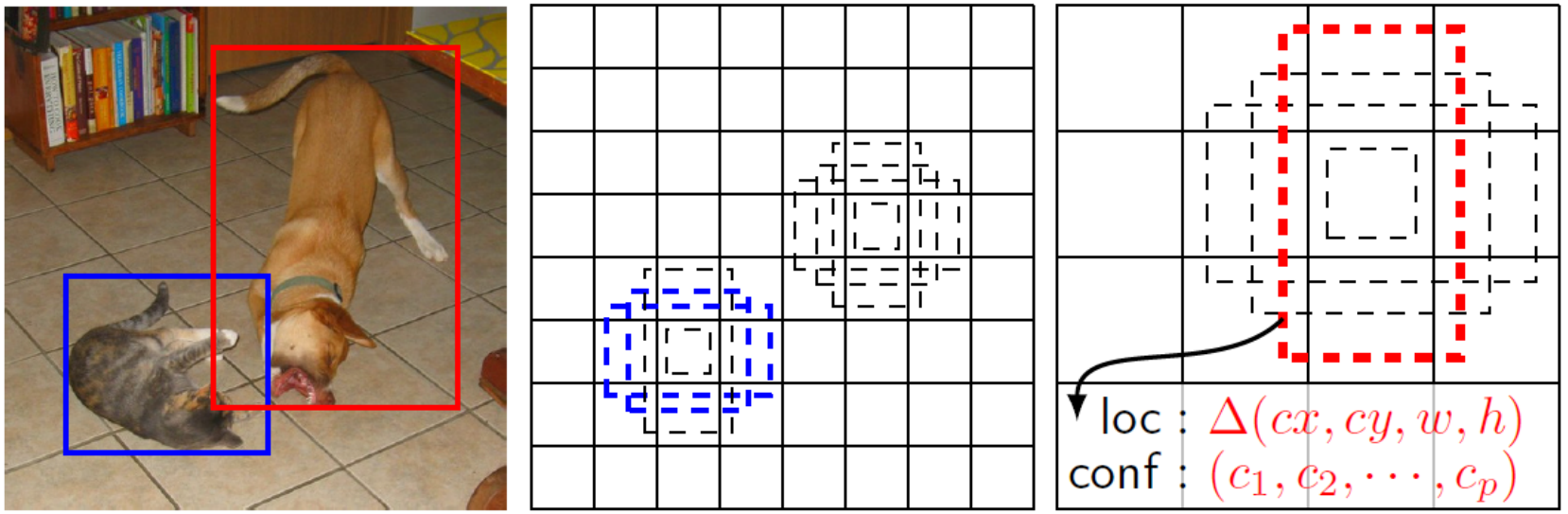
Appendix A - Ssd: Single shot multibox detector

- Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.



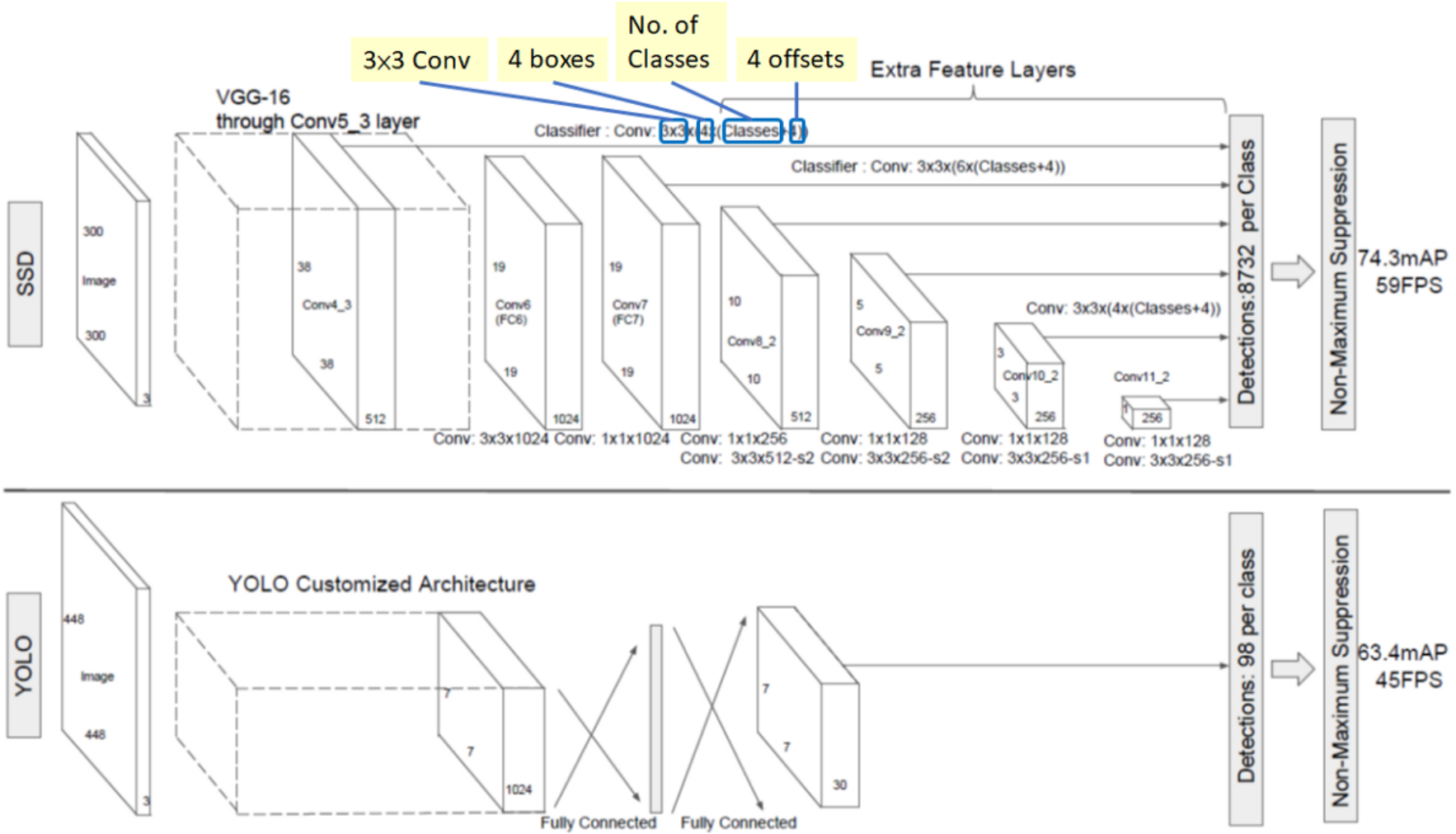
Evaluate with different scales

- we evaluate a small set (e.g. 4) of **default boxes of different aspect ratios** at **each location in several feature maps** with **different scales** (e.g. 8 x 8 and 4 x 4 in (b) and (c)).



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

SSD Architecture



SSD Architecture

- Say for example, at Conv4_3, it is of size $38 \times 38 \times 512$. 3×3 conv is applied. And there are **4 bounding boxes** and **each bounding box will have (classes + 4) outputs.** Thus, at Conv4_3, the output is **$38 \times 38 \times 4 \times (c+4)$.** Suppose there are 20 object classes plus one background class, the output is $38 \times 38 \times 4 \times (21+4) = 144{,}400$. **In terms of number of bounding boxes, there are $38 \times 38 \times 4 = 5776$** bounding boxes.

- Conv7: $19 \times 19 \times 6 = 2166$ boxes (6 boxes for each location)

- Conv8_2: $10 \times 10 \times 6 = 600$ boxes (6 boxes for each location)

- Conv9_2: $5 \times 5 \times 6 = 150$ boxes (6 boxes for each location)

- Conv10_2: $3 \times 3 \times 4 = 36$ boxes (4 boxes for each location)

- Conv11_2: $1 \times 1 \times 4 = 4$ boxes (4 boxes for each location)

- If we sum them up, we got 5776 + 2166 + 600 + 150 + 36 +4 = **8732 boxes in total**. If we remember YOLO, there are 7×7 locations at the end with 2 bounding boxes for each location. YOLO only got **$7 \times 7 \times 2 = 98$ boxes**. Hence, SSD has 8732 bounding boxes which is more than that of YOLO.

Loss Function

- Classification + Bounding-box Regression

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Default Box Scales & Aspect Ratios

- Suppose we have m feature maps for prediction, we can calculate Sk for the k-th feature map. Smin is 0.2, Smax is 0.9. That means the scale at the **lowest layer is 0.2** and the scale at the **highest layer is 0.9.** All layers in between is regularly spaced.

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m]$$

- For each scale, sk, we have 5 non-square aspect ratios:

$$a_r \in \{1, 2, 3, \tfrac{1}{2}, \tfrac{1}{3}\} \quad (w_k^a = s_k\sqrt{a_r}) \quad (h_k^a = s_k/\sqrt{a_r})$$

- For aspect ratio of 1:1, we got sk':

$$s'_k = \sqrt{s_k s_{k+1}}$$

- Therefore, **we can have at most 6 bounding boxes** in total with different aspect ratios. **For layers with only 4 bounding boxes, ar = 1/3 and 3 are omitted.**

Training Tricks

- **Hard Negative Mining** : Instead of using all the negative examples, we sort them **using the highest confidence loss for each default box** and pick the top ones so that the **ratio between the negatives and positives is at most 3:1**.This can lead to faster optimization and a more stable training.

- **Data Augmentation** : Each training image is randomly sampled by:

1. entire original input image
2. Sample a patch so that the overlap with objects is 0.1, 0.3, 0.5, 0.7 or 0.9.
3. Randomly sample a patch

- The size of each **sampled patch is [0.1, 1]** or original image size, and **aspect ratio from 1/2 to 2** After the above steps, **each sampled patch will be resized to fixed size and maybe horizontally flipped with probability of 0.5**, in addition to some **photo-metric distortions** [14].

Questions & Answers

Thank You!