CNN
Applications and Architectures

Chetan Arora

IIIT Delhi

Adapted from the slides of

Fei Fei Li, Andrej Karpathy, Justin Johnson, A. Torralba,

Svetlana Lazebnik, Ross Girshick, Kaiming He and Rob Fergus

Visual Recognition Tasks

Visual Recognition Tasks

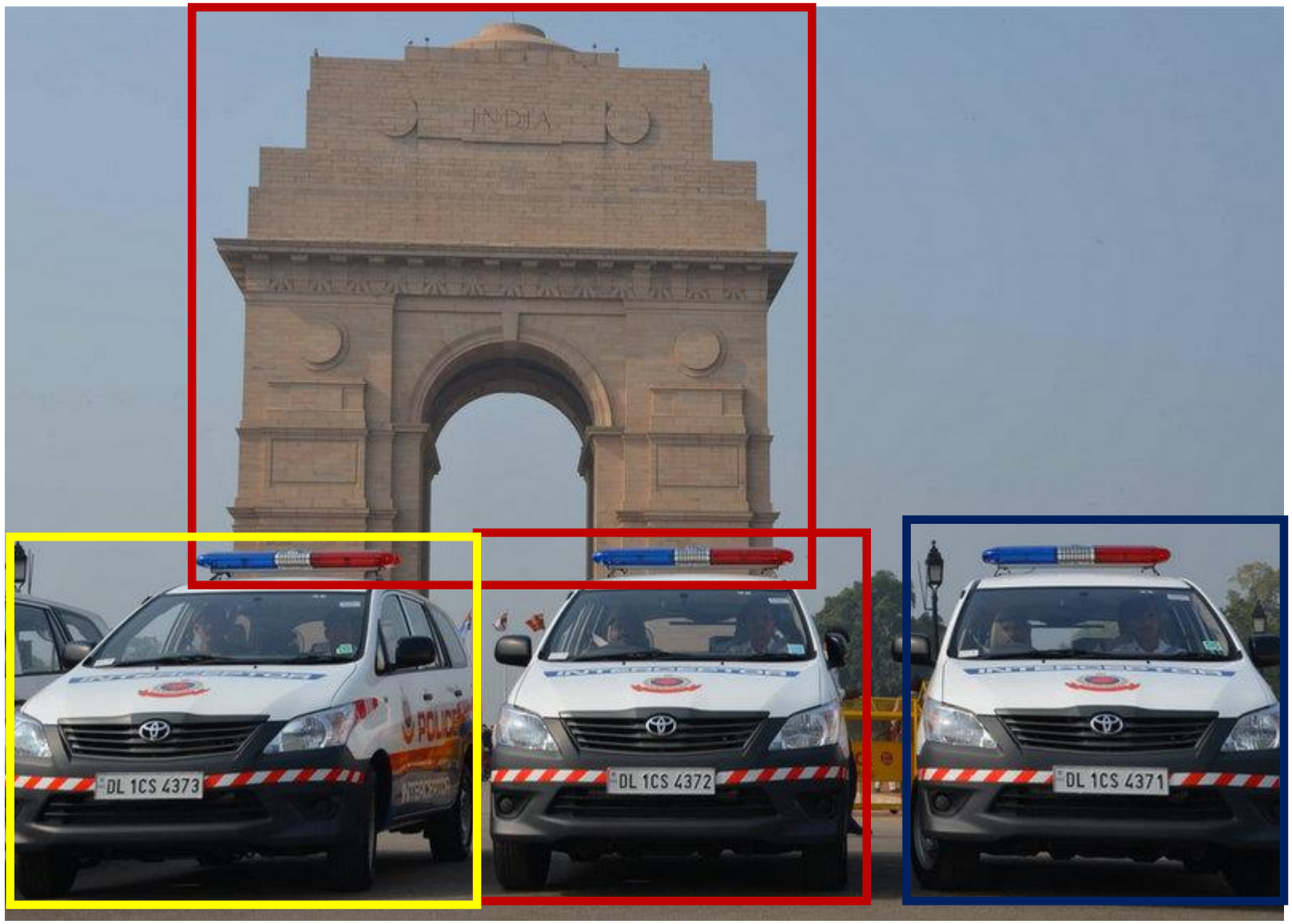
Image Classification

Is it a natural or man made scene

Is it a forest or a beach?

Does this image contains a building?

Visual Recognition Tasks



Object Detection

Does this image contain a building? [where]

Which objects does this image contains?

Visual Recognition Tasks

Object Detection [pixel wise localization]

Which pixels are building?

Visual Recognition Tasks

Semantic Segmentation

Classifying each pixel in the image

Applications: Computational Photography

Face Detection

Dynamic Range Enhancement

Applications: Object Attributes

Building:
42 m height
100 m away

Car:
Police Car
Frontal View

Autonomous and Assistive Driving

Applications: Instance Recognition

Does this image contains “India Gate”?

Recognizing landmarks in images

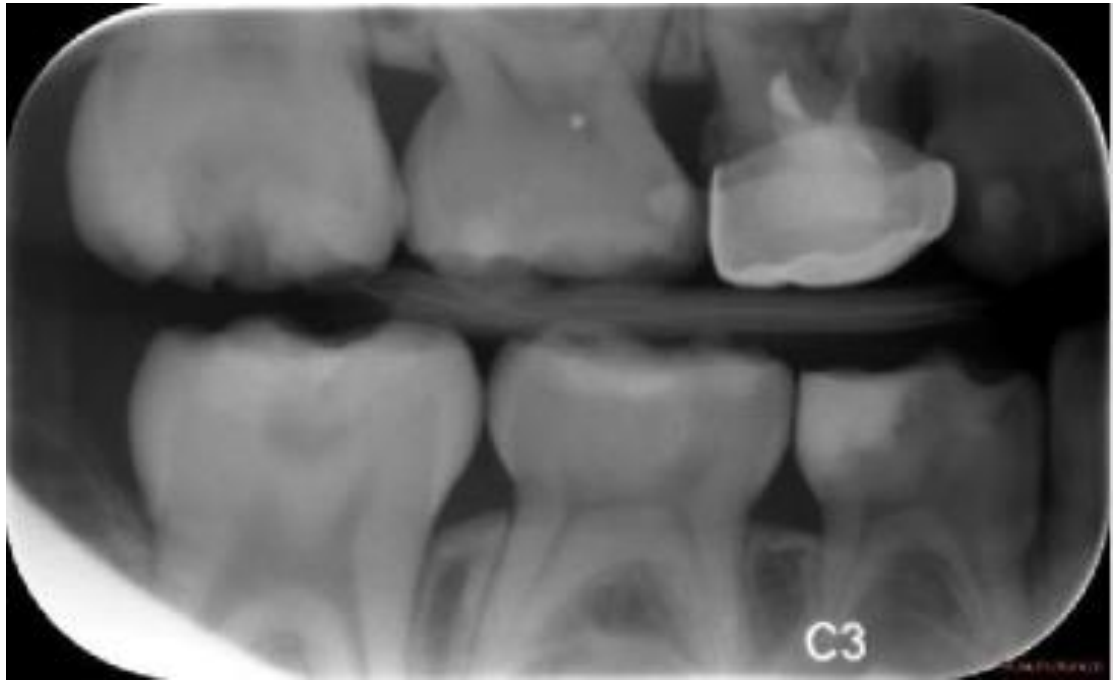
Recognizing products in super market

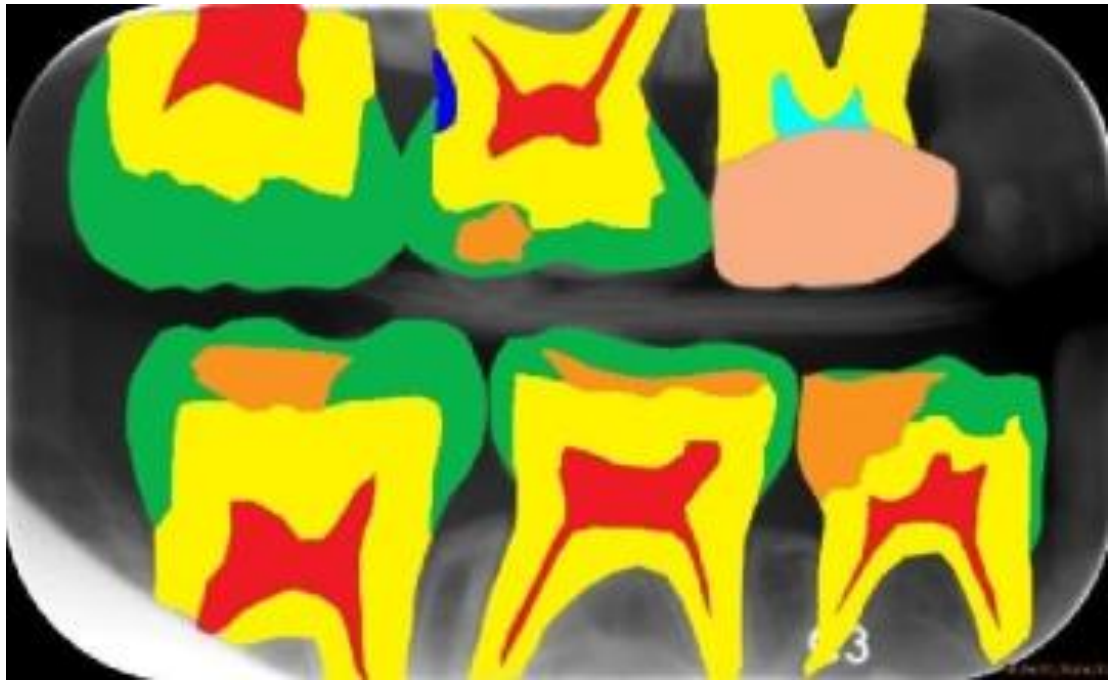
Applications: Assistive Vision

Applications: Security and Surveillance

Applications: Activity Recognition

Applications: Medical

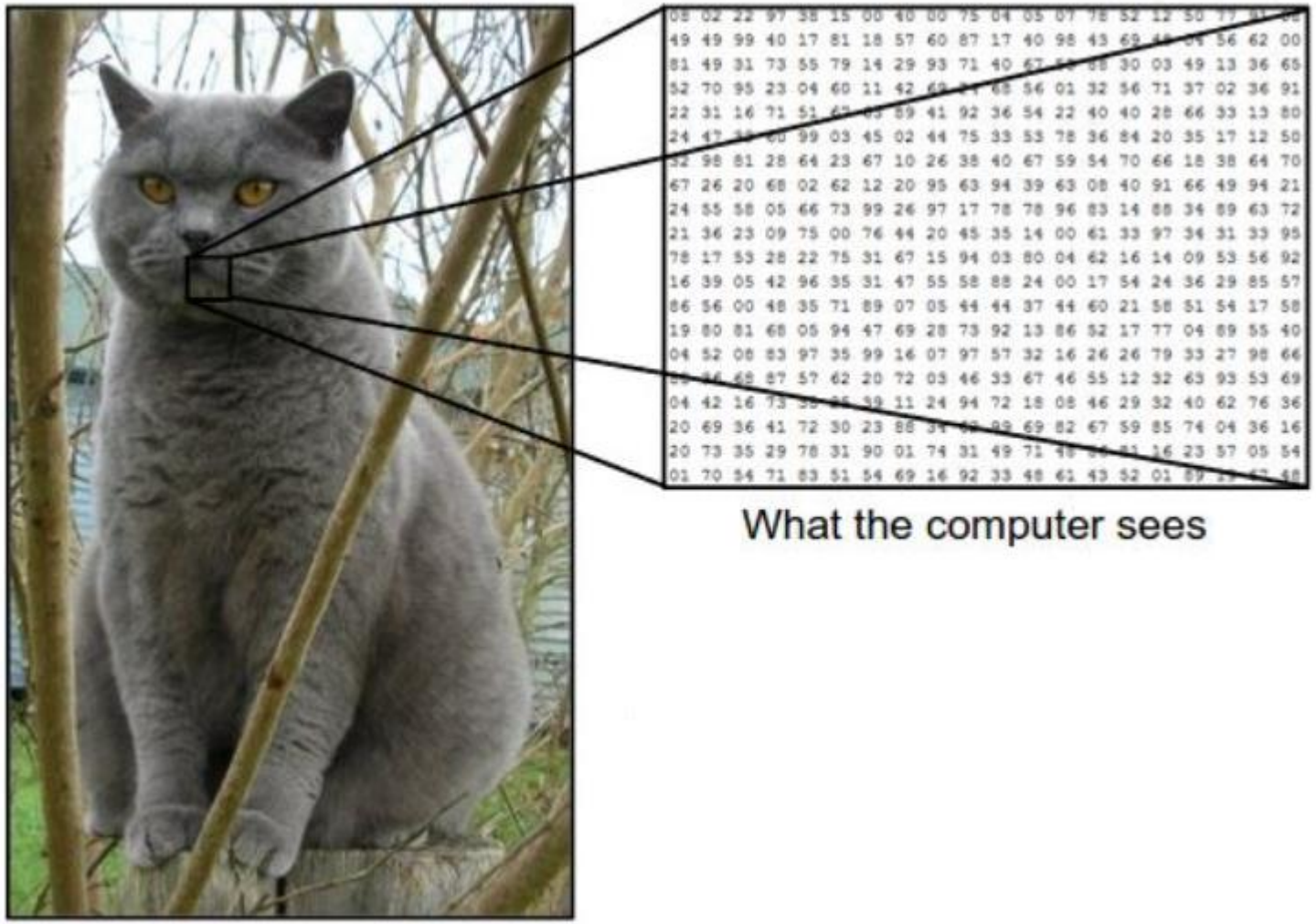




Applications: ADAS, Autonomous Driving

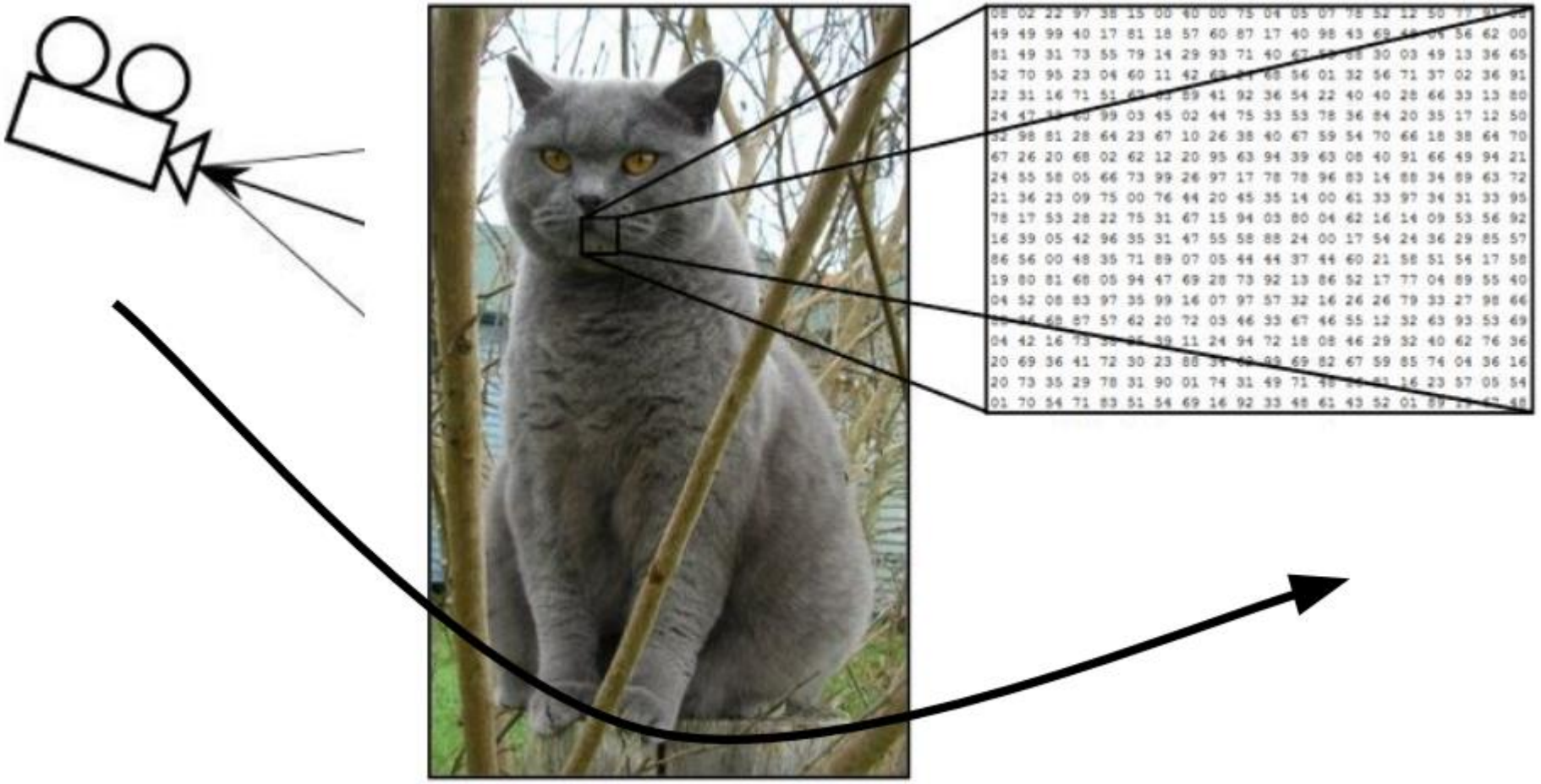
The Problem: Semantic Gap

- Images are represented as 3D arrays of numbers, with integers between [0, 255].
- E.g. 300 x 100 x 3 (3 for 3 color channels RGB)



What the computer sees

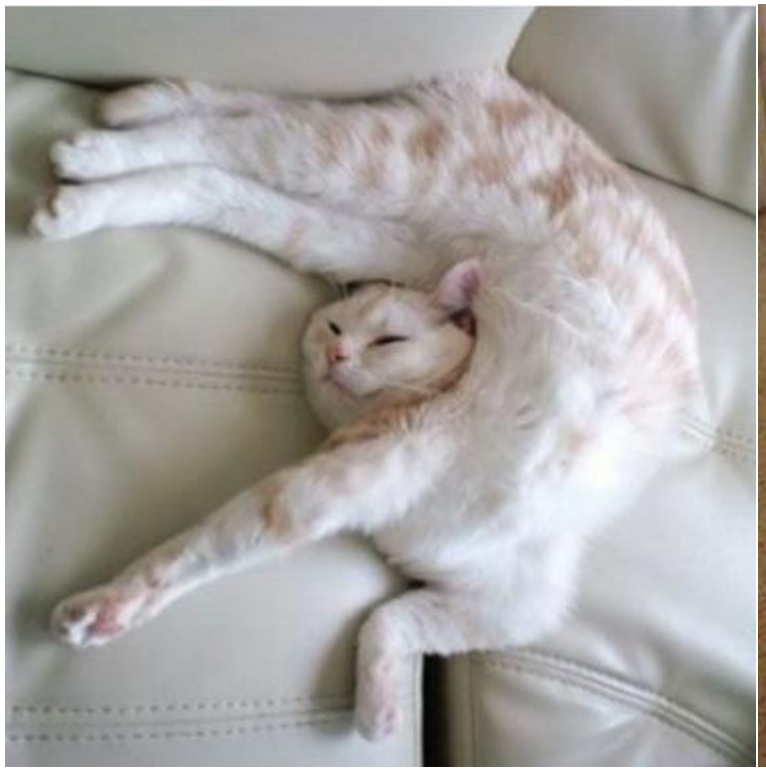
Challenges: View Point Variation



Challenges: Illumination Variation

Challenges: Scale

Challenges: Deformation




Challenges: Occlusion

Challenges: Background Clutter

Challenges: Interclass Variation

Image Classification

Data Driven Approach

- Collect a dataset of images and labels
- Use Machine Learning to train an image classifier
- Evaluate the classifier on a withheld set of test images

```
def train(train_images, train_labels):
    # build a model for images -> labels…
    return model
def predict(model, test_images):
    # predict test_labels using the model…
    return test_labels
```

Example Training Set

First classifier: Nearest Neighbor Classifier

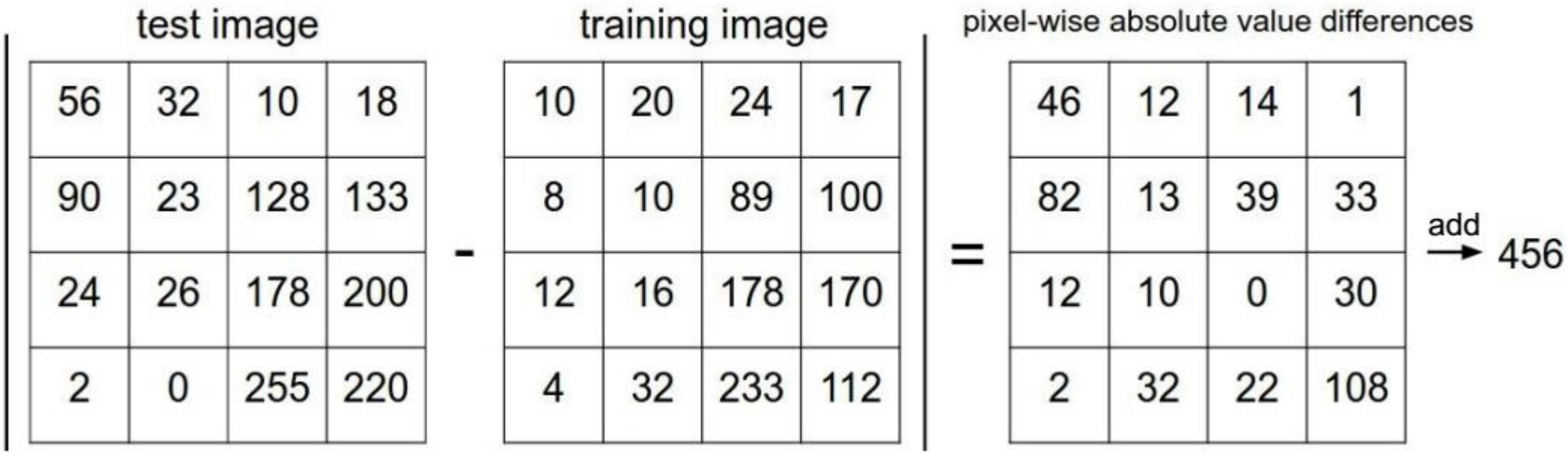
```
def train(train_images, train_labels):
    # build a model for images -> labels…
    return model
def predict(model, test_images):
    # predict test_labels using the model…
    return test_labels
```

Remember all training images and their labels

Predict the label of the most similar training image

How do we compare the images?

L1 Distance: $\mathrm{dist}(\mathrm{I}_1, \mathrm{I}_2) = \sum_p |I_1^p - I_2^p|$



Dataset

- Example dataset: CIFAR-10
- 10 labels
- 50,000 training images
- Each image is tiny: 32x32
- 10,000 test images

Output: Nearest Neighbor Classifier on CIFAR

For every test image (first column), examples of nearest neighbors in rows

Generalization

- Terrible performance in real life
- Distance metrics on level of whole images can be very unintuitive

Original

Shifted

Occluded

Darkened

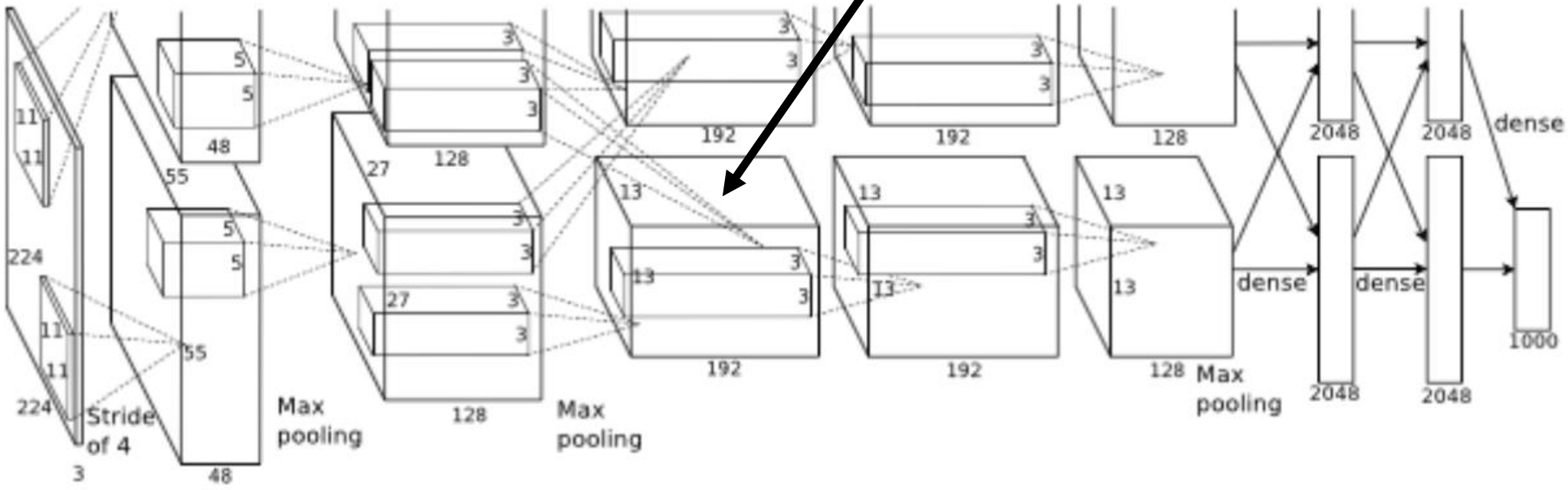
ImageNet Challenge

- ImageNet (image-net.org)
  - 15M high resolution images
  - over 22K categories
  - labeled by Mechanical Turk workers
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)
- 2010-2015
- 2010 competition
  - 1.2 M training images, 1000 categories (general and specific)
  - 200K test images
  - output a list of 5 object categories in descending order of confidence
  - two error rates: top-1 and top-5

AlexNet (2012)

- 5 convolutional layers
- 3 fully connected layers
- 1000-way softmax output layer

Trained on GTX 580 with 3 GB of RAM. Network (feature maps spread across two GPUs)



AlexNet (2012): Tricks of the Trade

- Downsampled images
  - shorter dimension 256 pixels, longer dimension cropped about center to 256 pixels
  - R, G, B channels
- Mean subtraction from inputs
- ReLU instead of logistic or tanh units
- DropOut

AlexNet (2012): Tricks of the Trade

- Data set augmentation
  - Generate image translations by selecting random 224 x 224 sub-images
  - Horizontal reflections (standard trick in computer vision)
  - When testing, extract 10 distinct 224x224 sub-images and average predictions

- More data set augmentation
  - Vary intensity and color of the illumination from epoch to epoch

Visualization

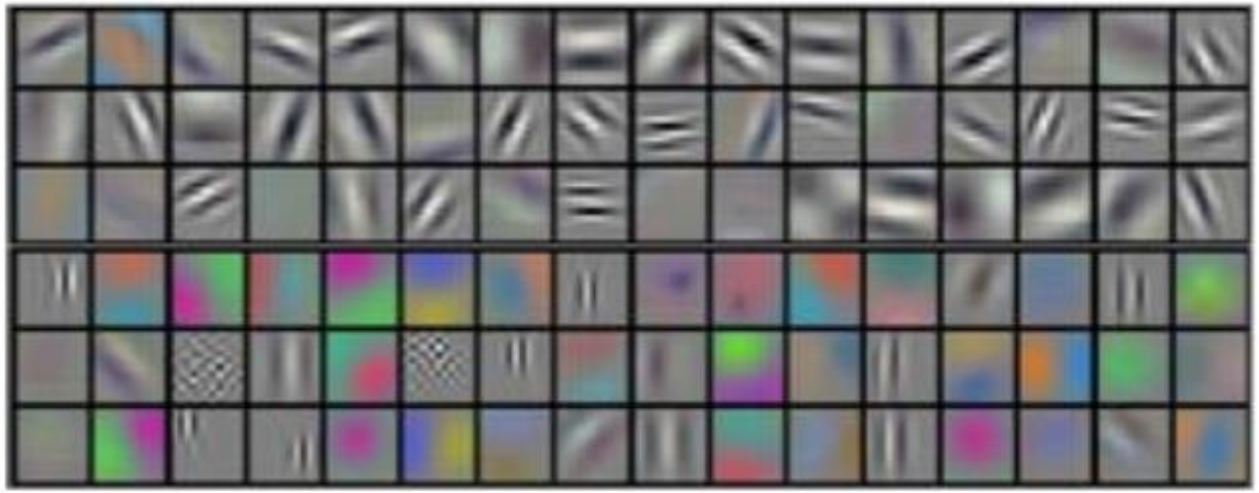


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

Results


mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Results

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

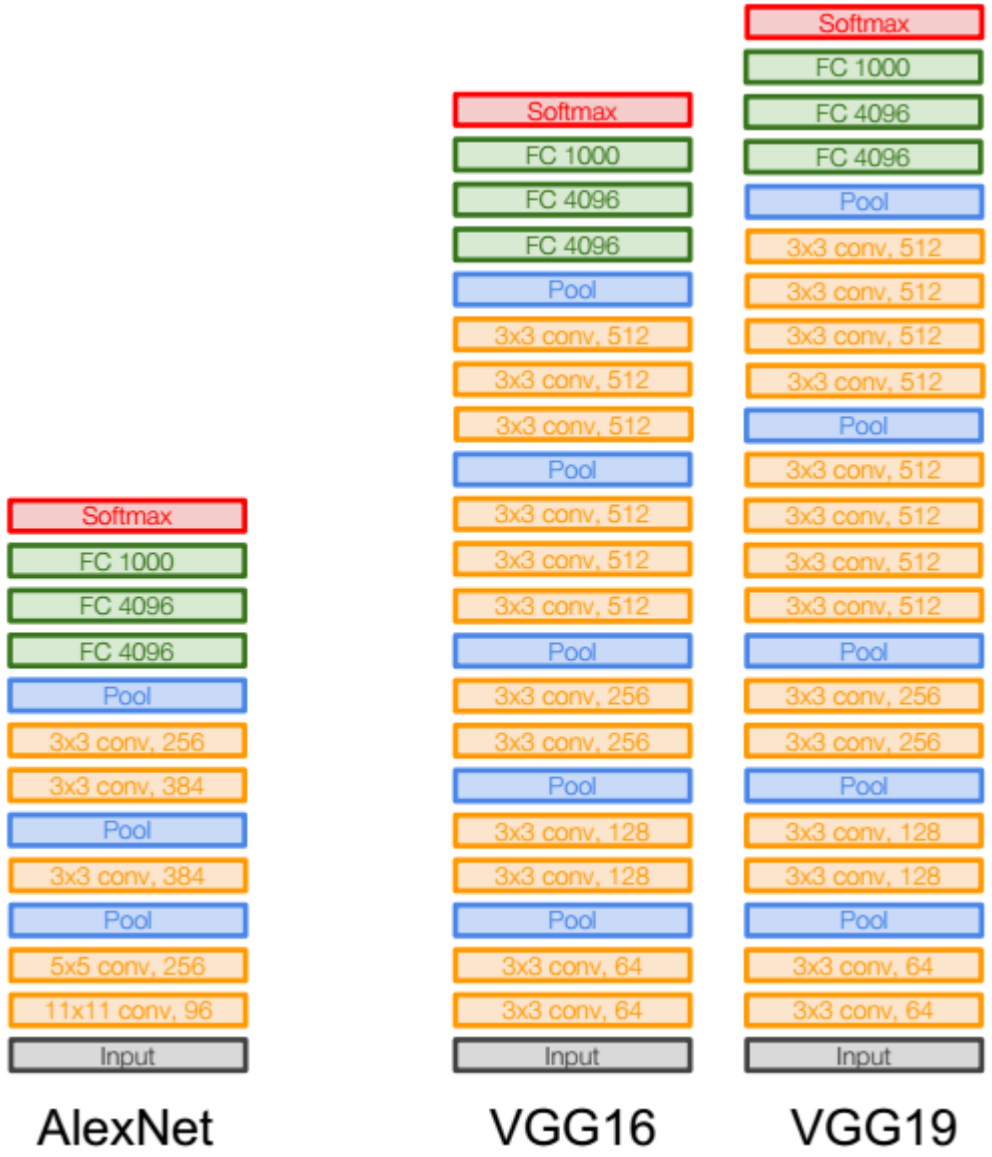
Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

VGG Net (2014)

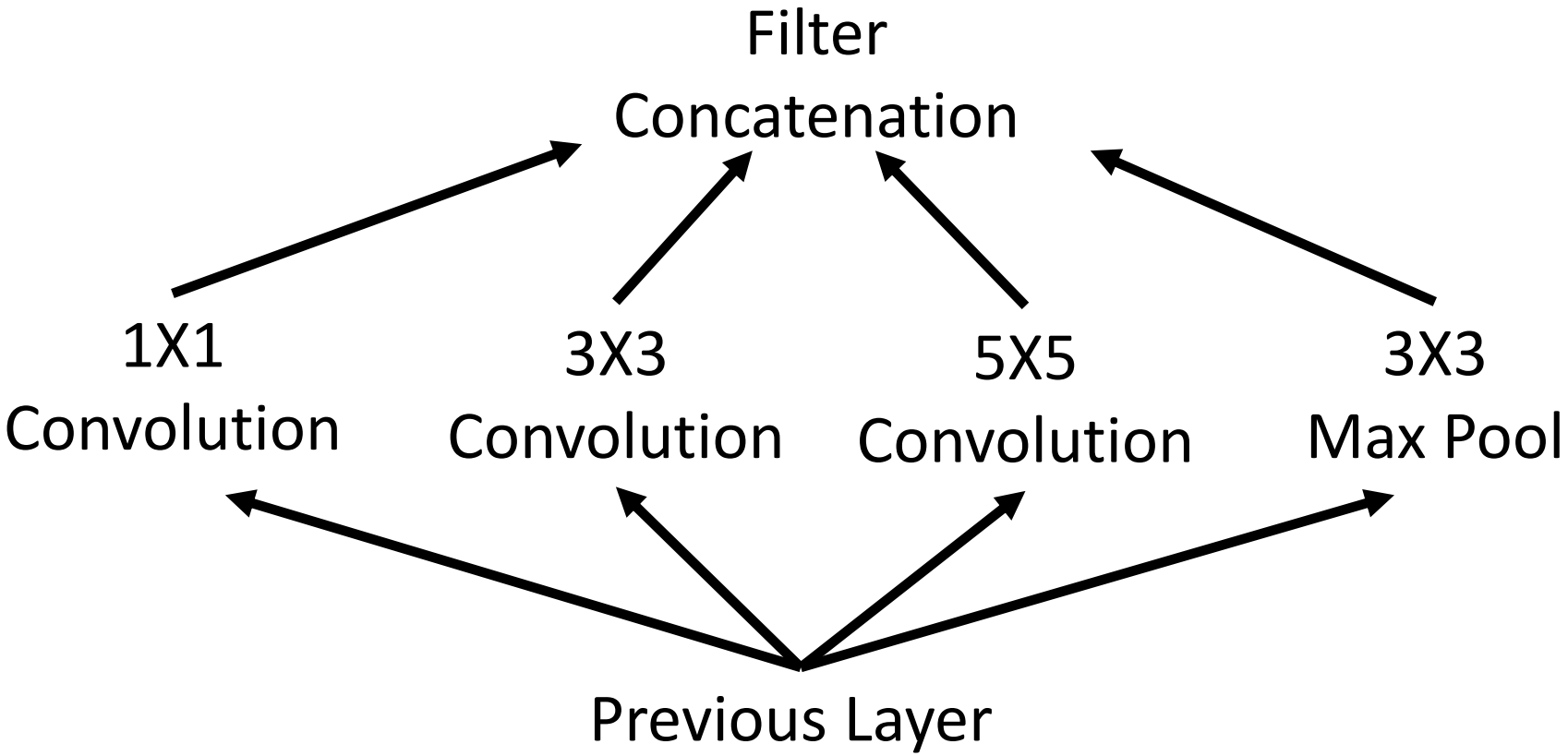
- Smaller filters, deeper networks.

Why?

- Stack of three 3x3 conv (stride 1) layers has same effective receptive field as one 7x7 conv layer
- But deeper, more non-linearities
- And fewer parameters: $3 * 3^2C^2$ vs. 7^2C^2 for C channels per layer



GoogLeNet (2014): Inception Module



GoogLeNet

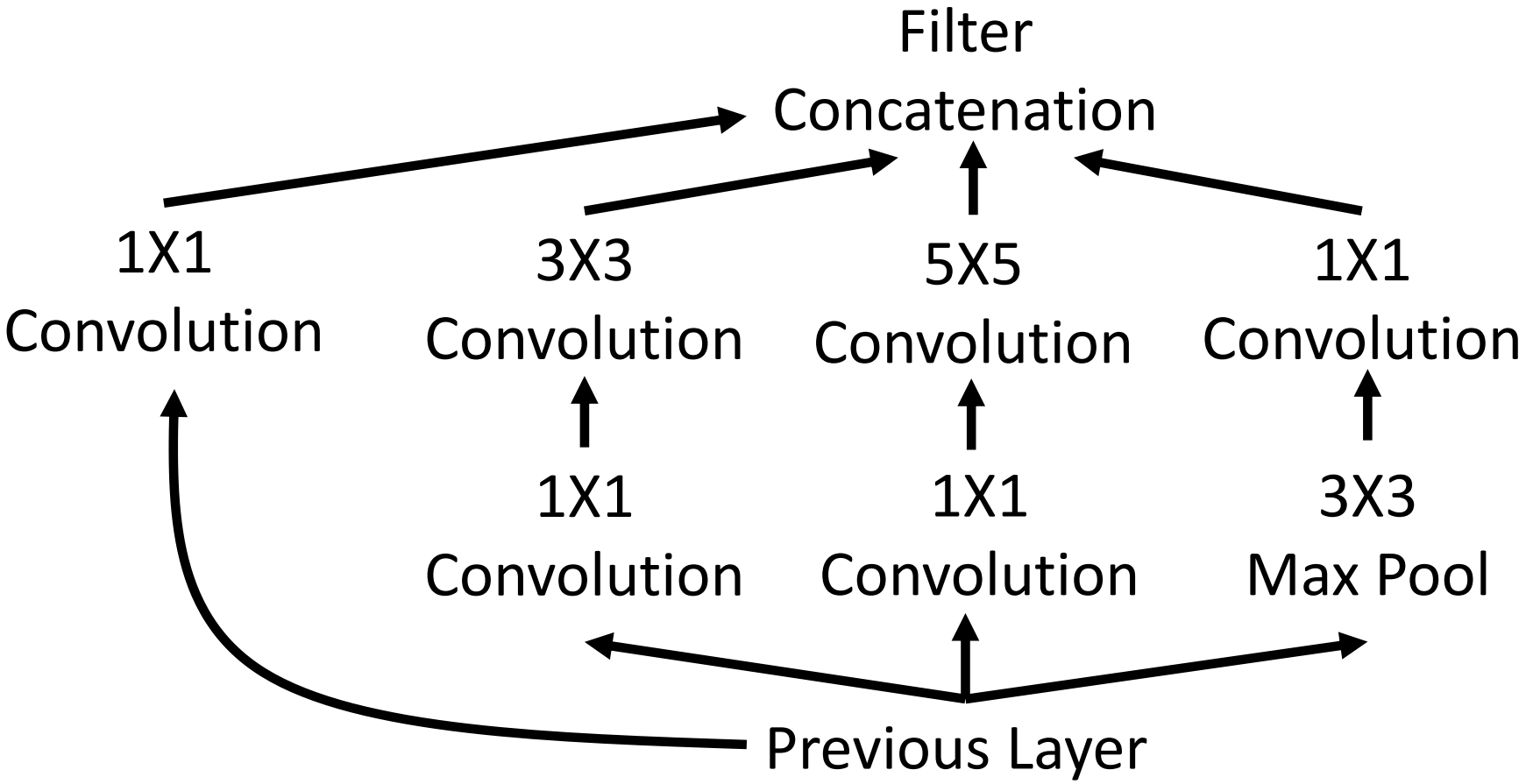
Problem?

- Heavy Compute: Too many feature maps because of feature cancatenation

Solution?

- 1X1 Convolution: preserves spatial dimensions, reduces depth

GoogLeNet: Inception Module



ResNet (2015)

- What happens when we continue stacking deeper layers on a “plain” convolutional neural network?
- Hypothesis: the problem is an optimization problem, deeper models are harder to optimize
- The deeper model should be able to perform at least as well as the shallower model.
- A solution by construction is copying the learned layers from the hallower model and setting additional layers to identity mapping

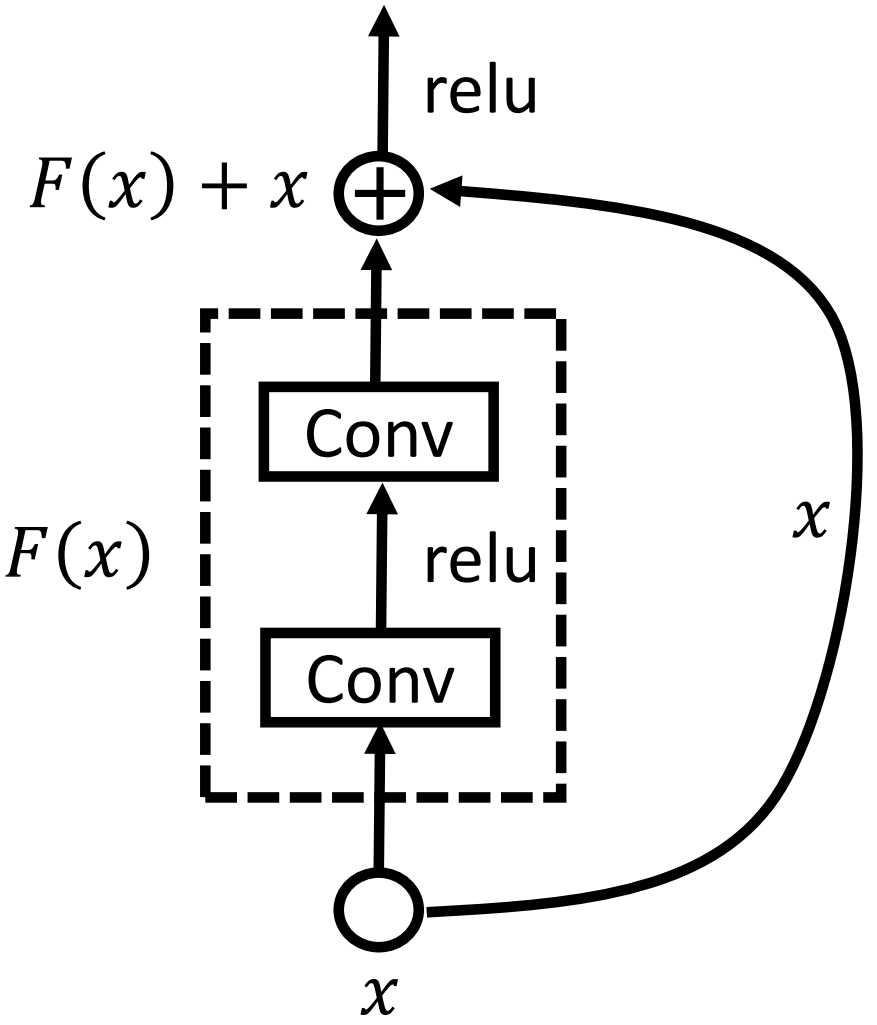


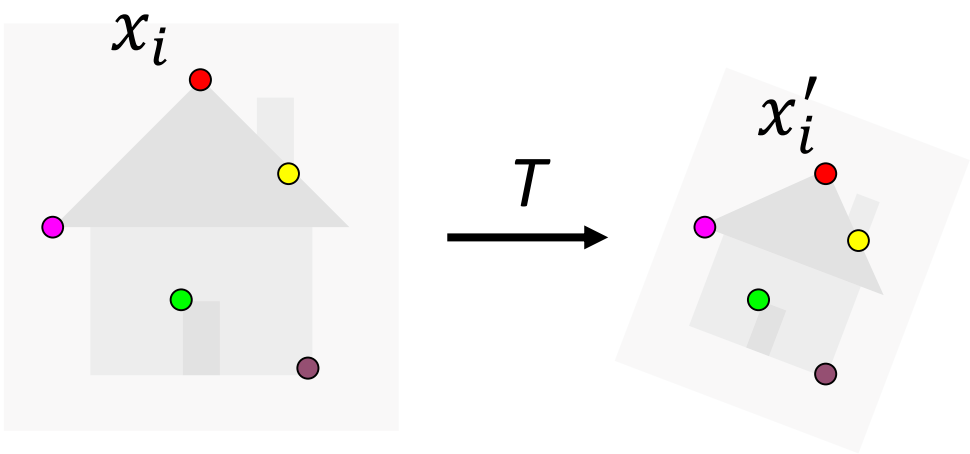
Image Classification Architectures: Summary

ILSVRC 2012	AlexNet		
ILSVRC 2013	ZFNet	Proposed by Zeiler-Fergus	improvement on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers.
ILSVRC 2014	GoogLe Net	Szegedy et al, Google	Development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large amount of parameters that do not seem to matter much.
ILSVRC 2014 Runner-up	VGGNet	Simonyan Zisserman	Showed that network depth is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. Outperformed GoogLeNet in multiple transfer learning tasks.
ILSVRC 2015	ResNets	He et al	Special skip connections and features heavy use of batch normalization. Also missing fully connected layers at the end of the network.

Object Detection

Object Detection Ideas: Geometrical Era

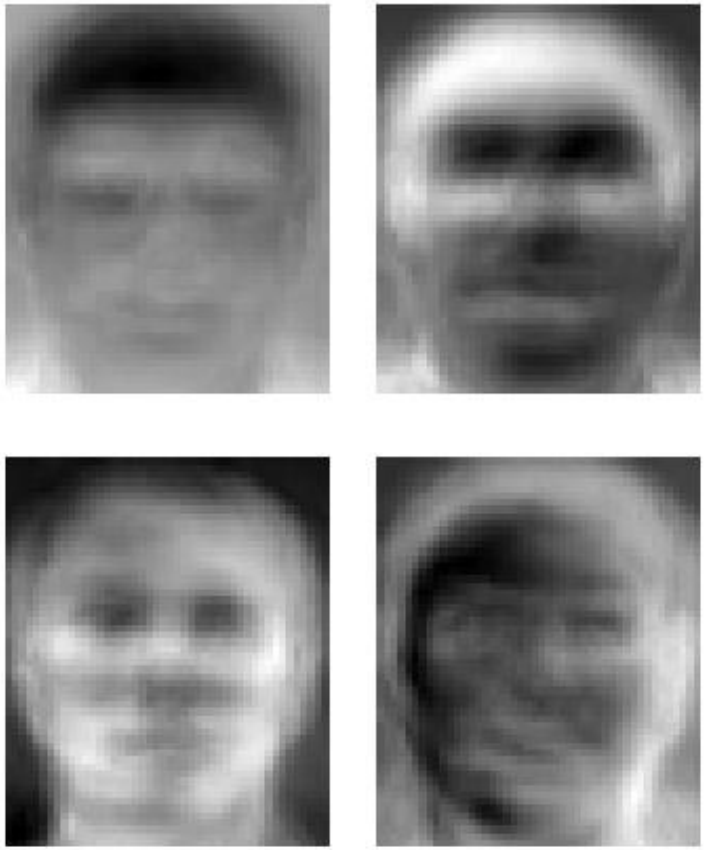
- Fit a model to a transformation between pairs of features (*matches*) in two images



Find transformation T that minimizes:

$$\sum_i residual(T(x_i), x_i')$$

Object Detection Ideas: Appearance Models

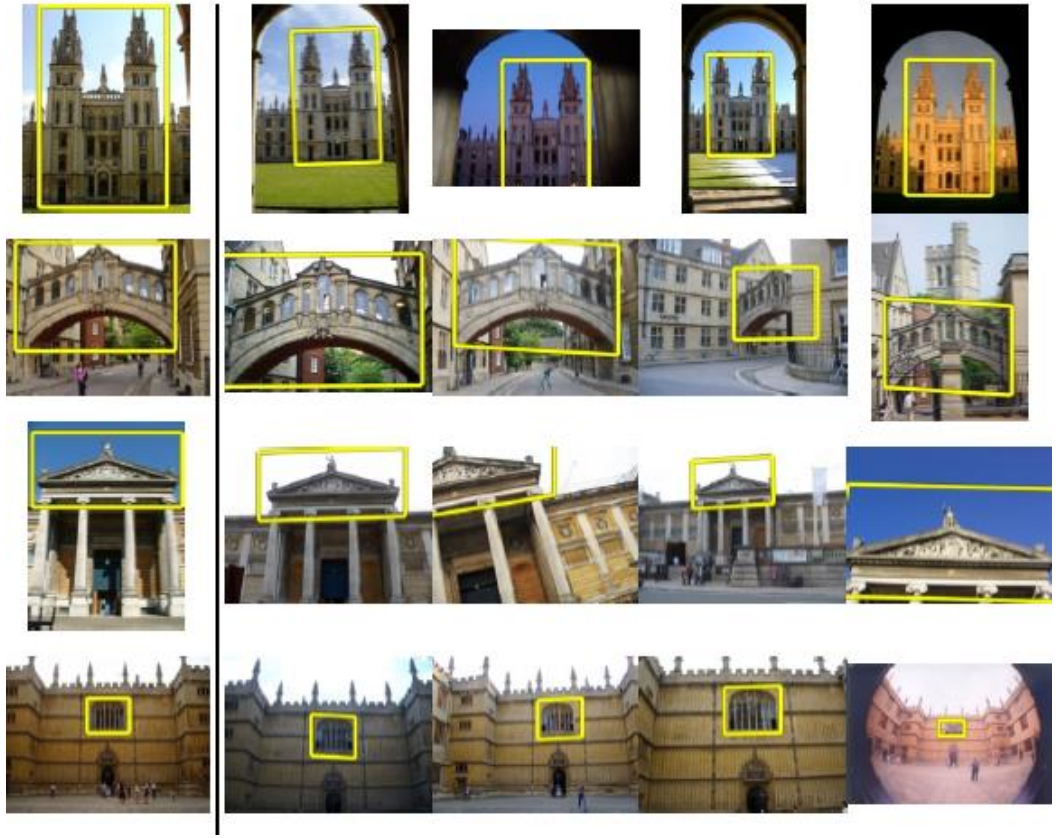


Eigenfaces (Turk and Pentland, 1991)

- Other global models based upon color histograms, appearance manifolds.
- Requires global registration of patterns. Not robust to clutter, occlusion, geometric transformations

Object Detection Ideas: Local Features

SIFT, D. Lowe (1999, 2004)



Large-scale image search, combining local features, indexing, and spatial constraints
Philbin et al. ‘07

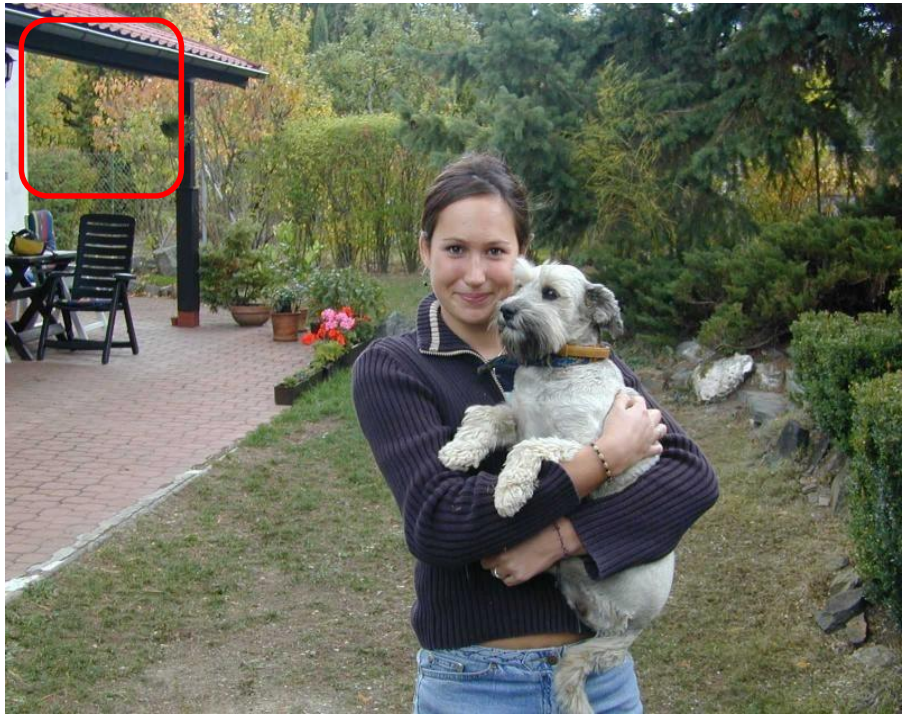
Constellation Model

- Globally different objects, but have portions in common

Bag of Features Models

Object → **Bag of ‘words’**

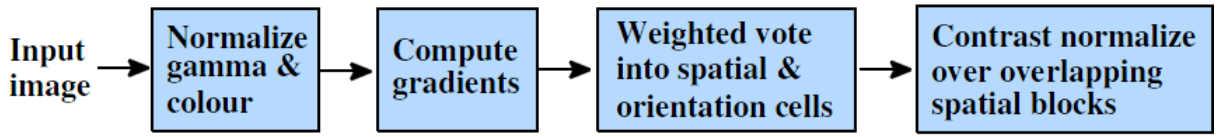
Object Detection Ideas: Sliding Window

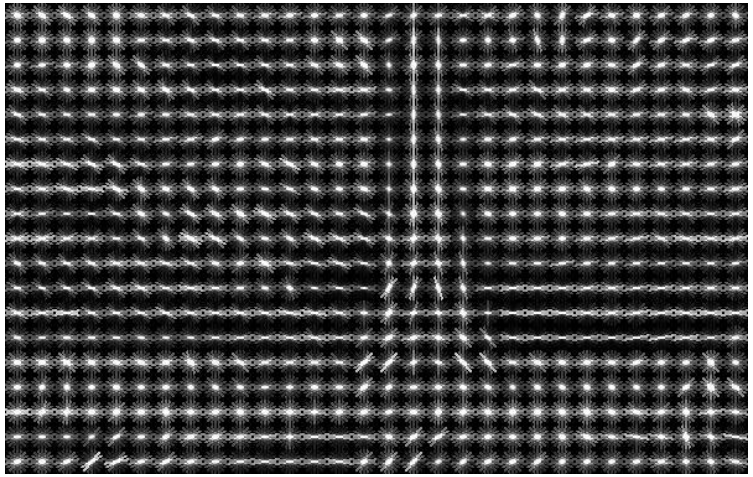


- Turk and Pentland, 1991
- Belhumeur et al., 1997
- Schneiderman & Kanade 2004
- Viola and Jones, 2000
- Schneiderman & Kanade, 2004
- Agrawal and Roth, 2002

Histograms of oriented gradients (HOG)

- Partition image into blocks and compute histogram of gradient orientations in each block





N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Image credit: N. Snavely

Pedestrian detection with HOG

- Train a pedestrian template using a linear support vector machine

positive training examples

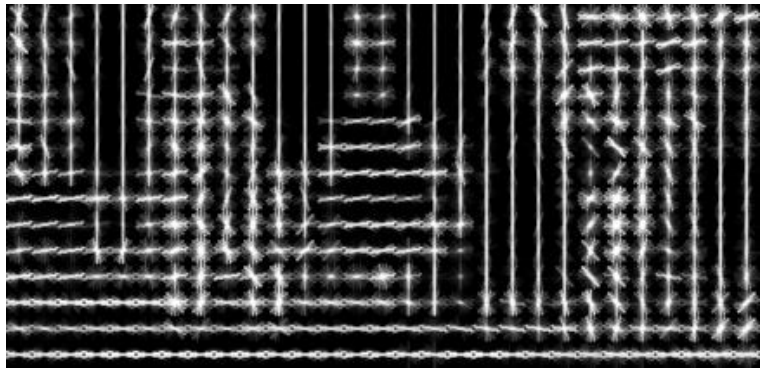
negative training examples

N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

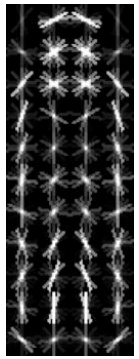
Pedestrian detection with HOG

- Train a pedestrian template using a linear support vector machine
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG *pyramid*

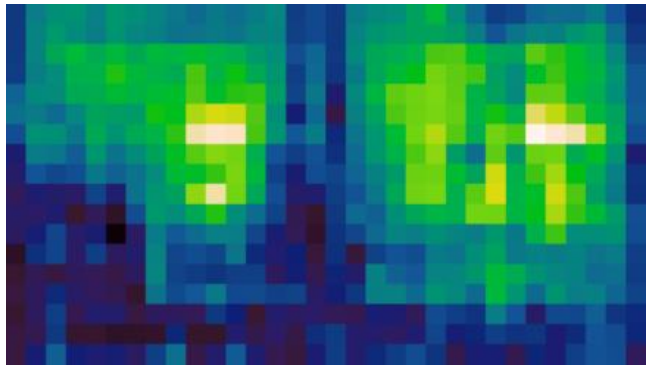
HOG feature map



Template



Detector response map

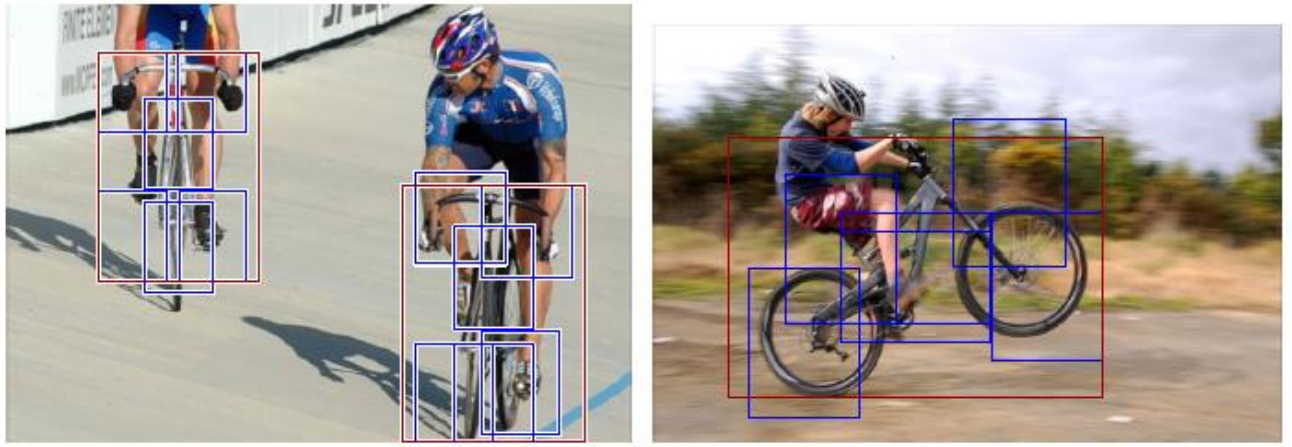


N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Example detections

[Dalal and Triggs, CVPR 2005]

Discriminatively trained part-based models



P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, PAMI 2009, “Object Detection with Discriminatively Trained Part-Based Models”

Object Detection with Deep Neural Networks

HoG by Convolutional Layers

Steps of computing HOG:

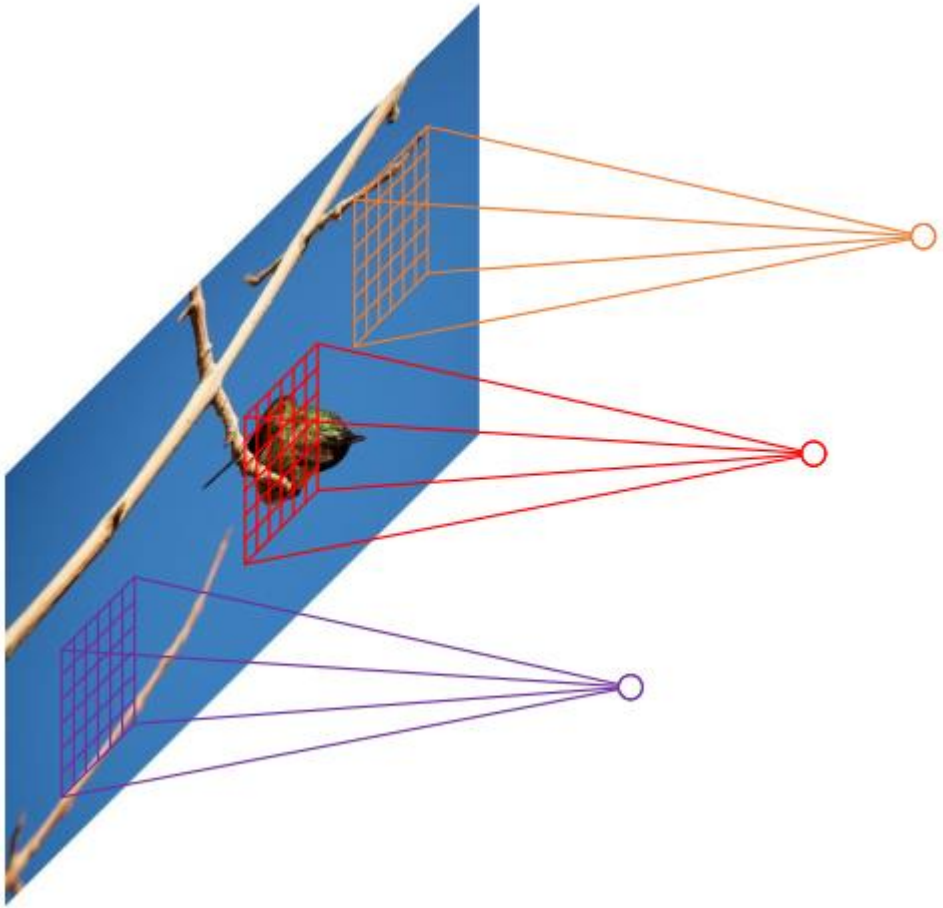
- Computing image gradients
- Binning gradients into 18 directions
- Computing cell histograms
- Normalizing cell histograms

Convolutional perspectives:

- Horizontal/vertical edge filters
- Directional filters + gating (non-linearity)
- Sum/average pooling
- Local response normalization (LRN)

HOG, dense SIFT, and many other "hand-engineered" features are convolutional feature maps.

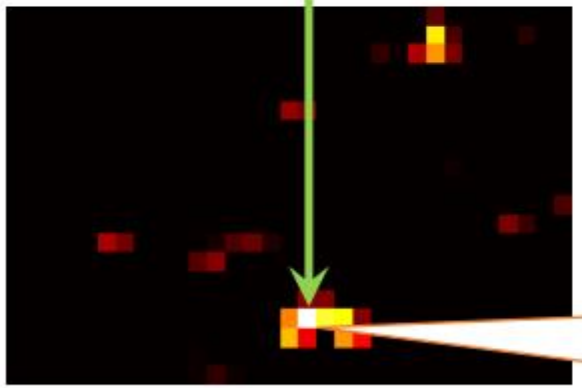
Convolutional Layers



Convolutional layers are locally connected

- a filter/kernel/window slides on the image or the previous map
- the position of the filter explicitly provides information for localizing
- local spatial information w.r.t. the window is encoded in the channels

Feature Maps = features and their locations



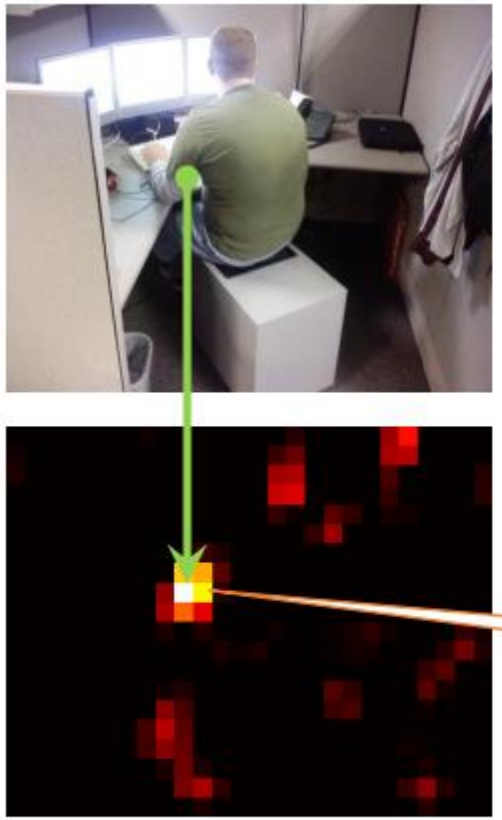
one feature map of $conv_5$
(#55 in 256 channels of a model trained on ImageNet)

ImageNet images with strongest responses of this channel

Intuition of *this* response:
There is a "circle-shaped" object (likely a tire) at this position.

What **Where**

Feature Maps = features and their locations



one feature map of $conv_5$
(#66 in 256 channels of a model trained on ImageNet)

ImageNet images with strongest responses of this channel

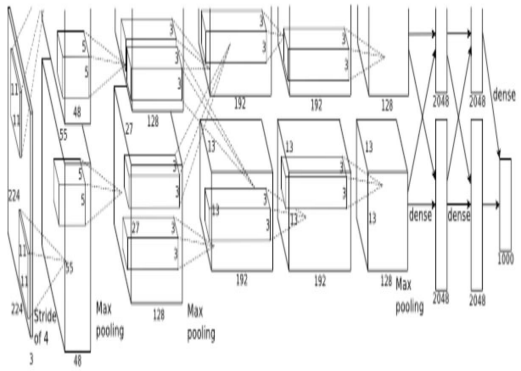
Intuition of *this* response:
There is a "λ-shaped" object (likely an underarm) at this position.

What **Where**

Classification + Localization

Cat

Localization as Regression Problem



Output of last fully connected layer: 4096 dim vector

4096 -> 1000

Class Scores

Cat: 0.9

Dog: 0.1

Car: 0.05

Correct Label

Softmax Loss

Total Loss

4096 -> 4

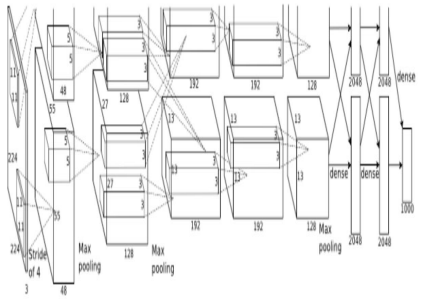
Box Coordinates

x, y, w, h

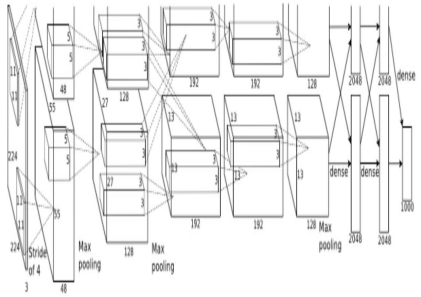
L2 Loss

Correct Box

Localization as Regression: Problem?

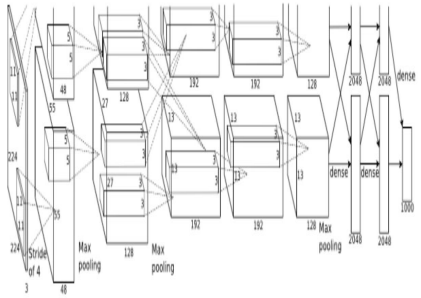


Cat: (x,y,w,h)



Dog: (x,y,w,h)
Cat: (x,y,w,h)

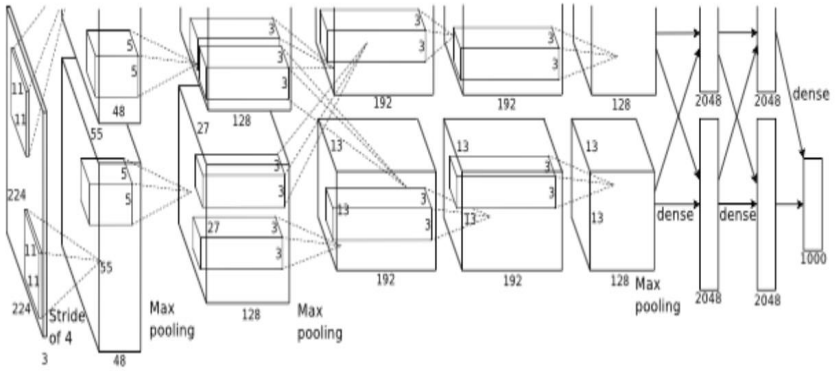
Each image needs a different number of outputs!



Duck: (x,y,w,h)
Duck: (x,y,w,h)
…

Object Detection as Classification

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



Dog: Yes

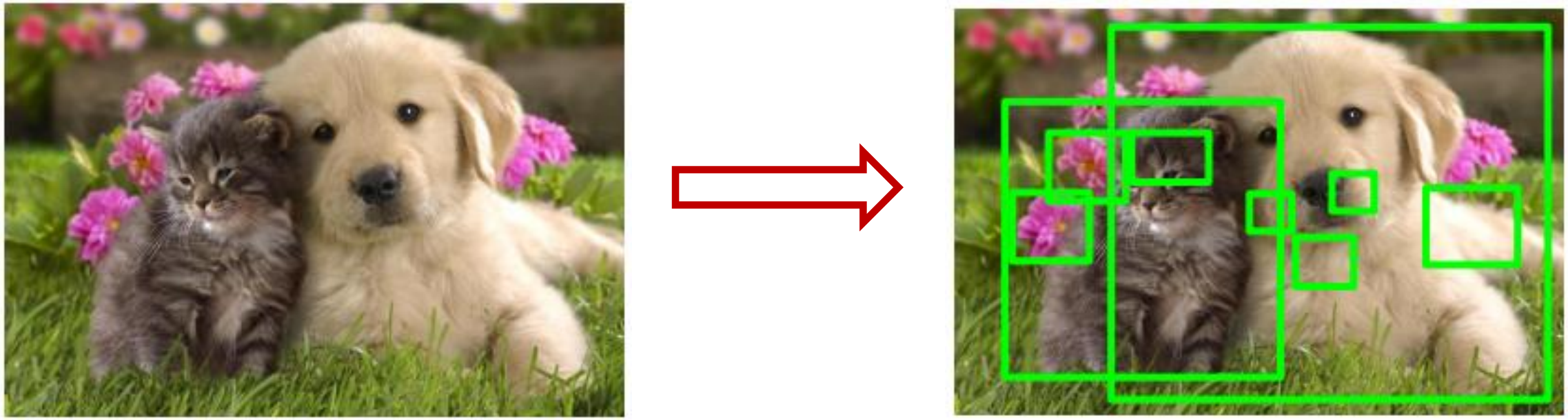
Cat: No

BG: Yes

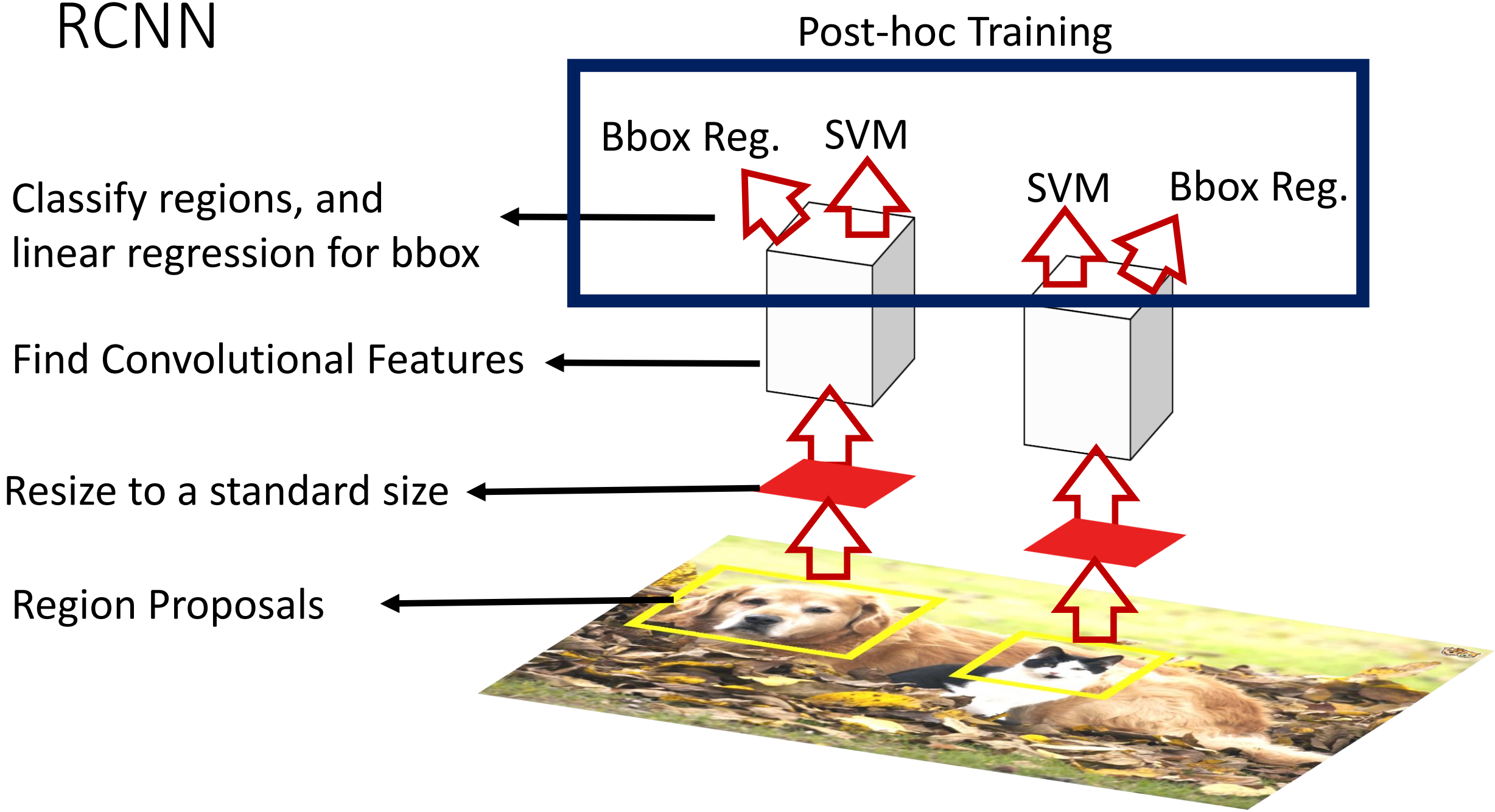
Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive.

Region Proposals: “Objectness” Detection

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



RCNN


Post-hoc Training
Bbox Reg.
SVM
SVM
Bbox Reg.
Classify regions, and
linear regression for bbox
Find Convolutional Features
Resize to a standard size
Region Proposals

RCNN: 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)

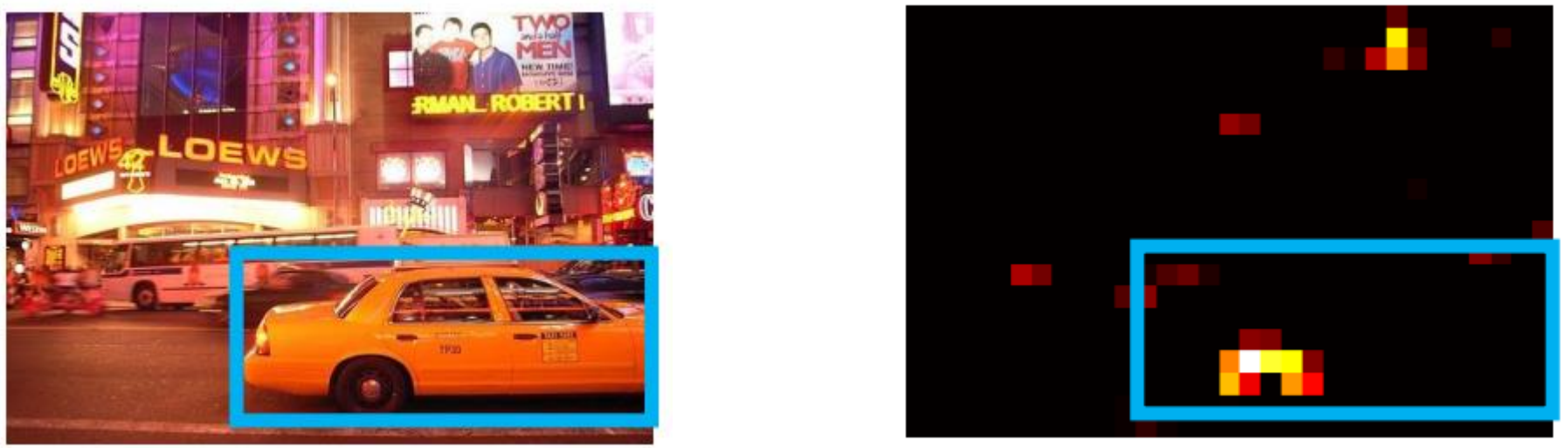
- Training is slow (84h), takes a lot of disk space

- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - ~2K proposals/passes through a Conv Net

Region-based CNN Features

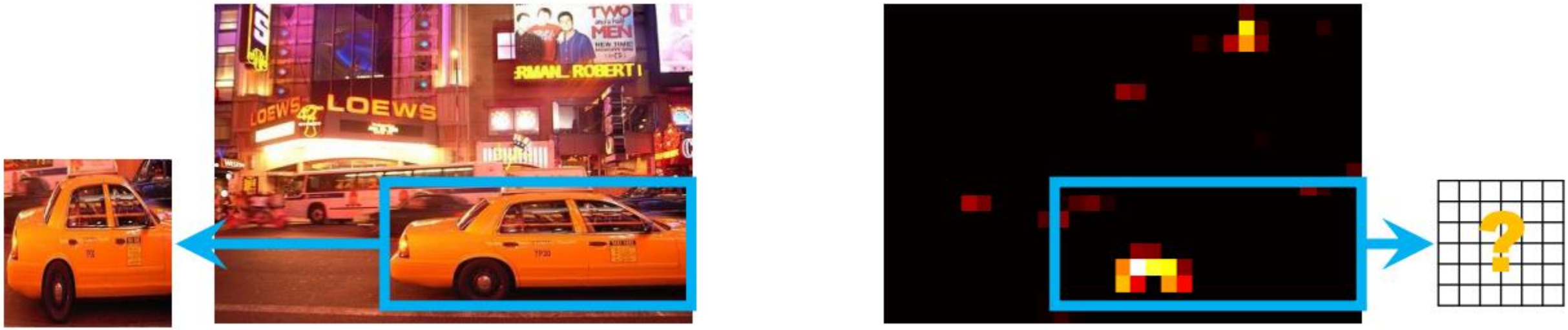
- Given proposal regions, what we need is a feature for each region
- R-CNN: cropping an image region + CNN on region, requires 2000 CNN computations
- What about cropping feature map regions?

Regions on Feature Maps



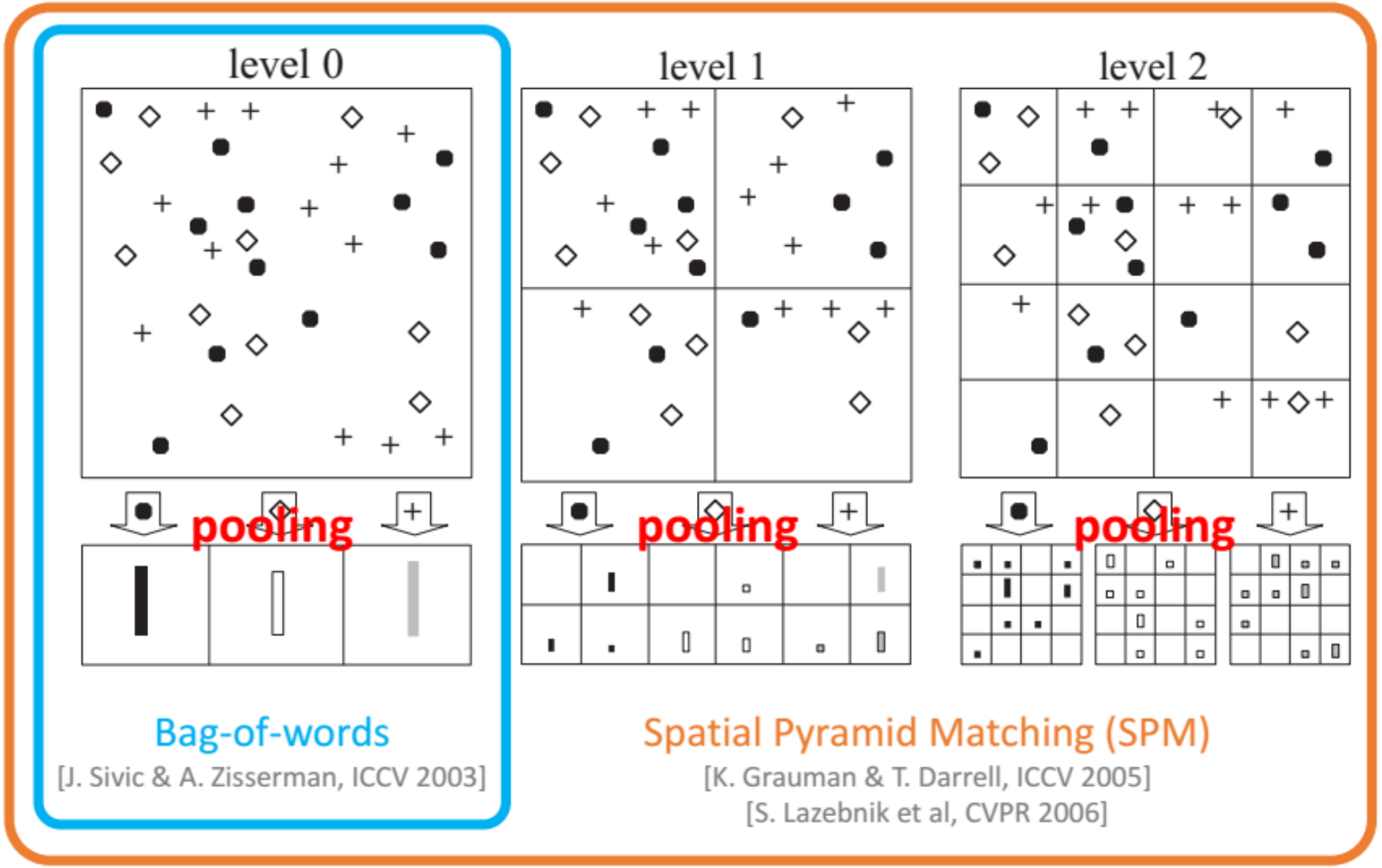
- Compute convolutional feature maps on the entire image only once.
- Project an image region to a feature map region (using correspondence of the receptive field center)
- Extract a region-based feature from the feature map region…

Regions on Feature Maps



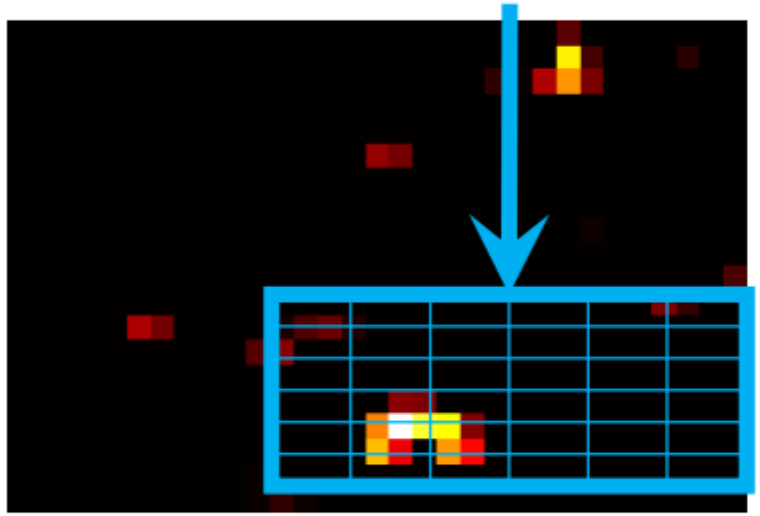
- Fixed-length features are required by fully-connected layers or SVM
- But how to produce a fixed-length feature from a feature map region?
- Solutions in traditional compute vision: Bag-of-words, SPM…

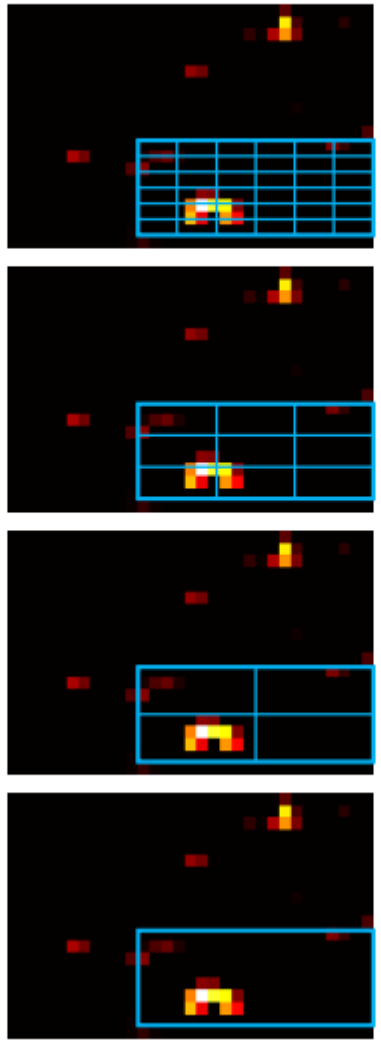
Bag-of-words & Spatial Pyramid Matching



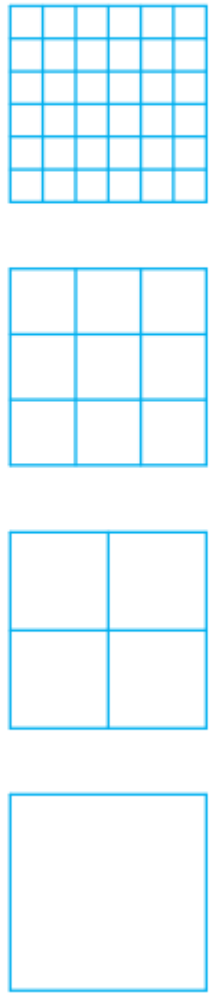
Spatial Pyramid Pooling (SPP) Layer

- fix the number of bins (instead of filter sizes)
- adaptively-sized bins





pooling



a finer level maintains explicit spatial information

concatenate, fc layers…

a coarser level removes explicit spatial information (bag-of-features)

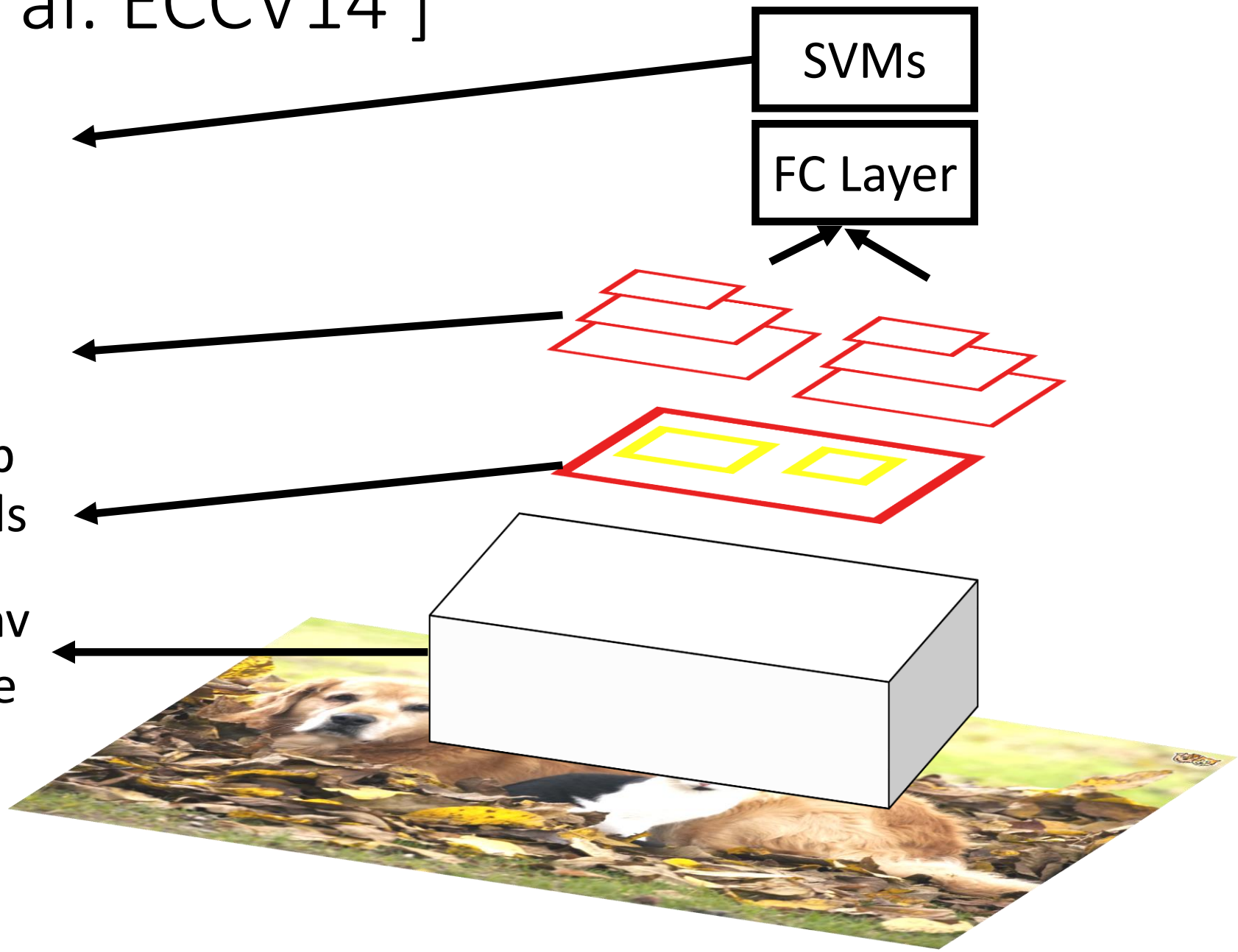
SPP Net [He et al. ECCV14]

Classify each region with SVMs

Spatial Pyramid Representation (SPP)

“Conv 5” Feature Map with region proposals

Apply single pass conv net over whole image



SPP Net: Good and Bad

- Good
  - Fixes the inference speed issue with RCNN
- Bad
  - Inherits adhoc training objectives from RCNN
  - Training still slow.
- New Problem
  - Parameters below SPP layers can not be updated during training

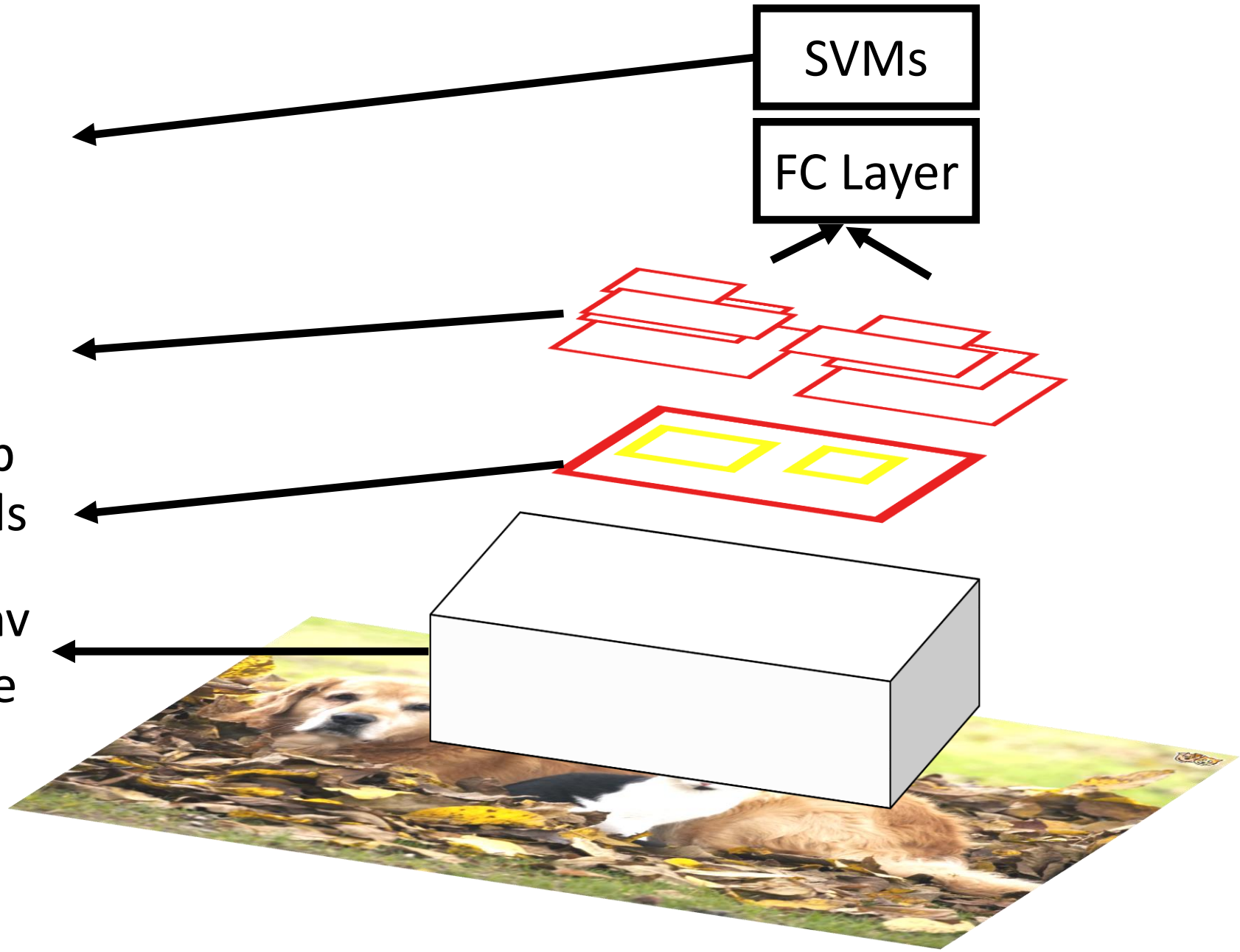
Fast RCNN

Classify each region with SVMs

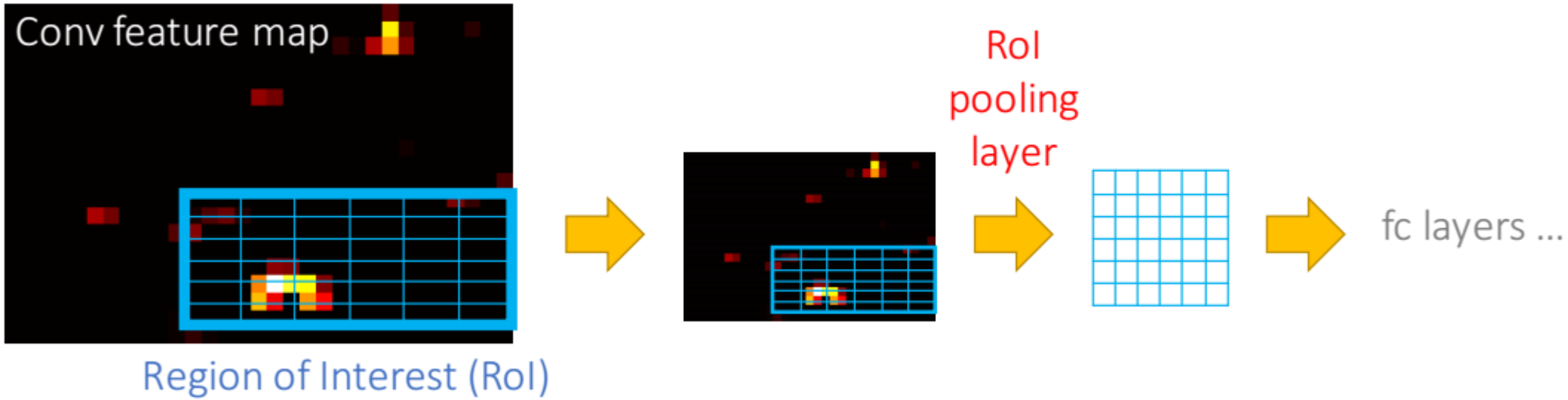
Spatial Pyramid Representation (SPP)

“Conv 5” Feature Map with region proposals

Apply single pass conv net over whole image



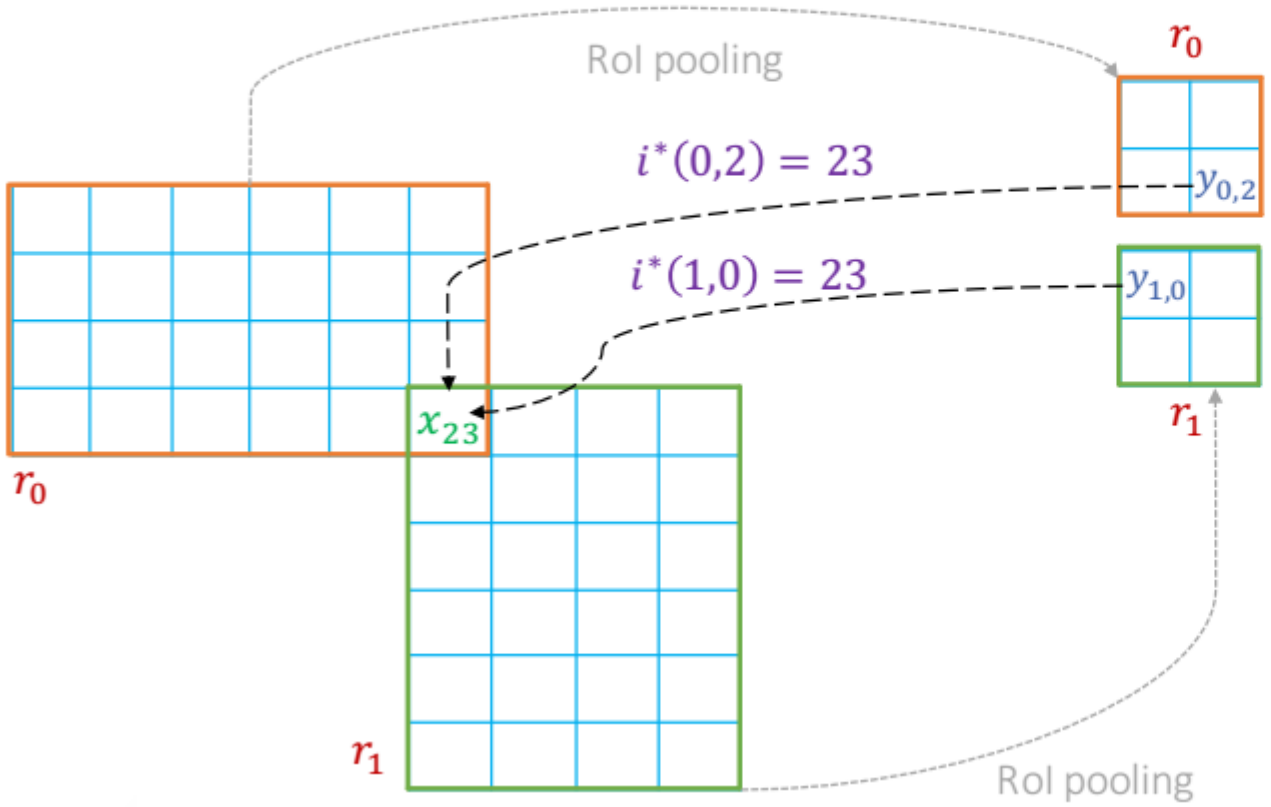
ROI Pooling Layer



Just a special case of the SPP layer with one pyramid level

Differentiable ROI Pooling

RoI pooling / SPP is just like max pooling, except that pooling regions overlap



$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}$$

1 if r, j "pooled" input i; 0 o/w

Partial for x_i

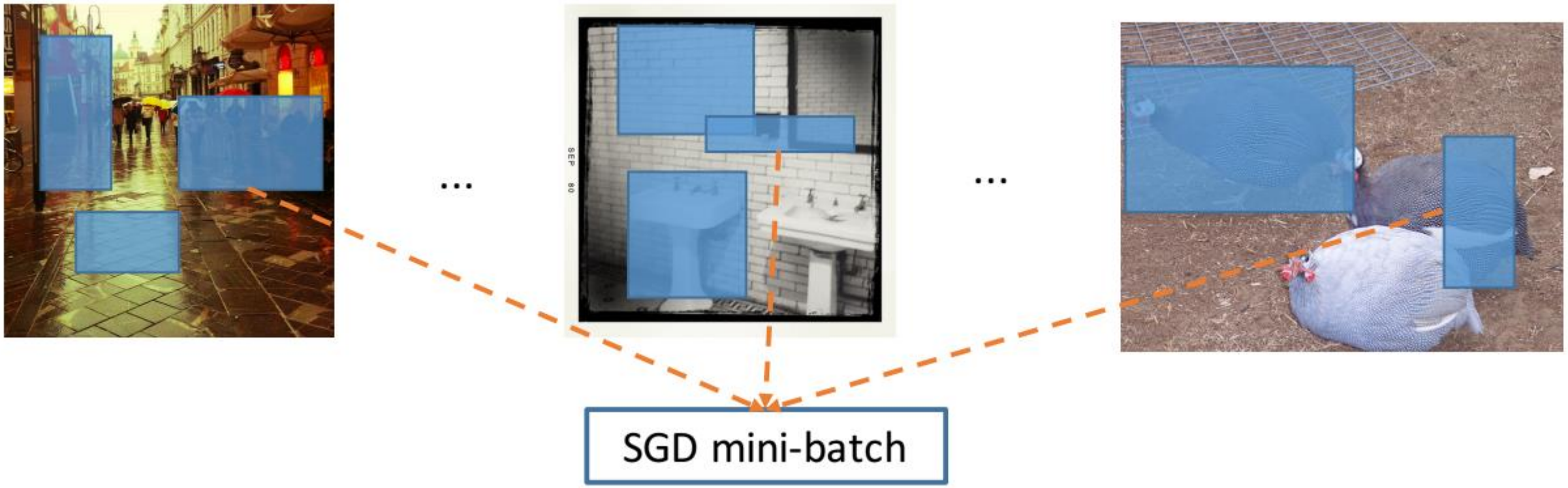
Over regions r, locations j

Partial from next layer

←– – – max pooling "switch" (*i.e.* argmax back-pointer)

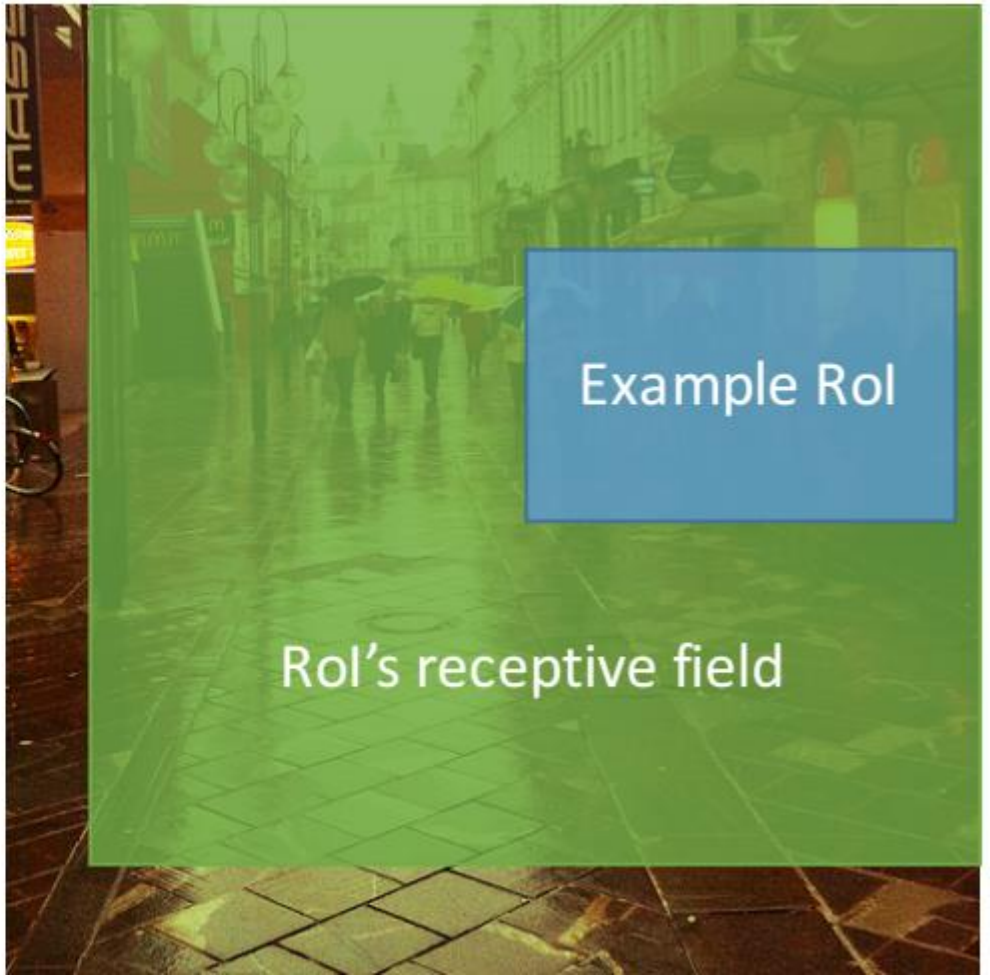
Efficient SGD

- R-CNN and SPP-net use region-wise sampling to make mini-batches
- Sample 128 example RoIs uniformly at random
- Examples will come from different images with high probability



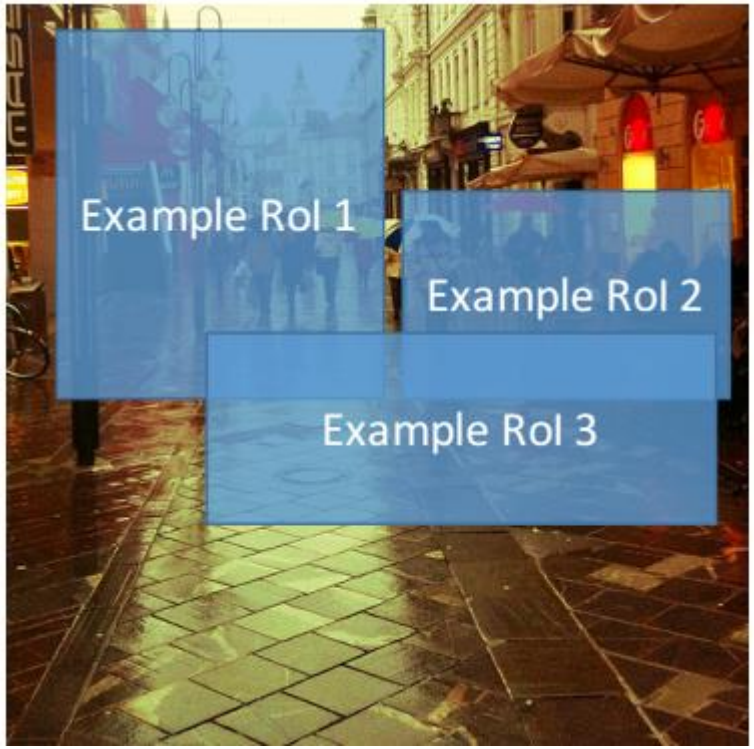
Efficient SGD

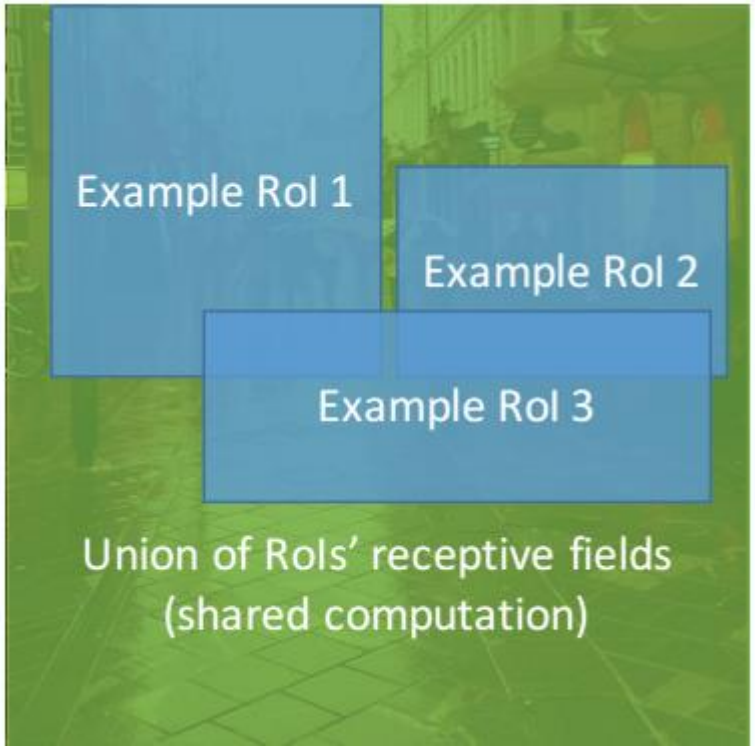
- The receptive field of one RoI could be very large (in worst case almost an whole image)



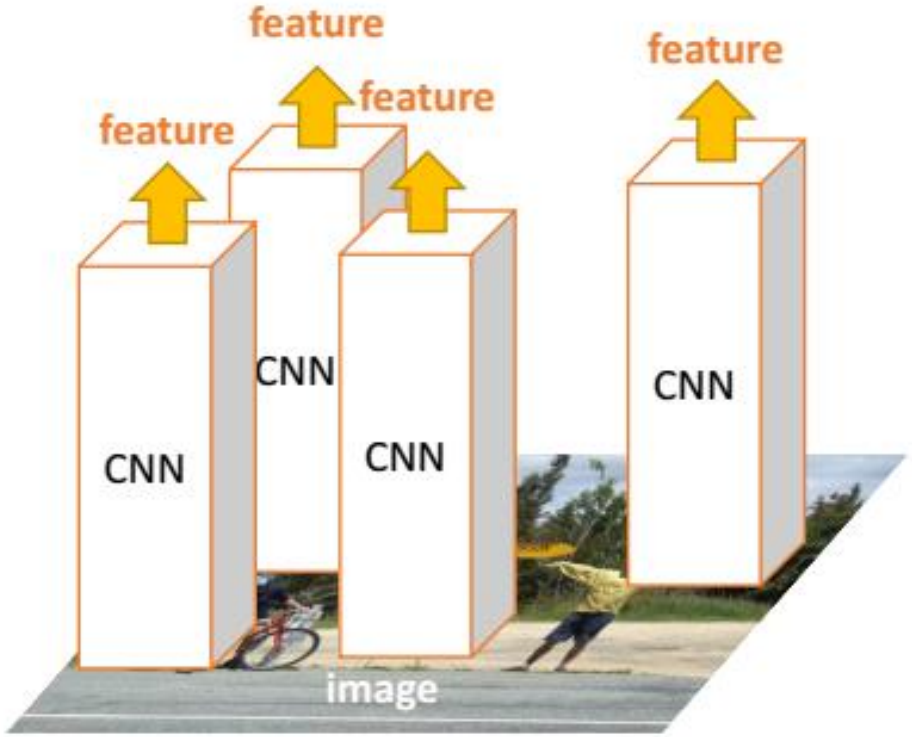
Efficient SGD

- Solution: use hierarchical sampling to build mini-batches
- Share computation between overlapping examples from the same image



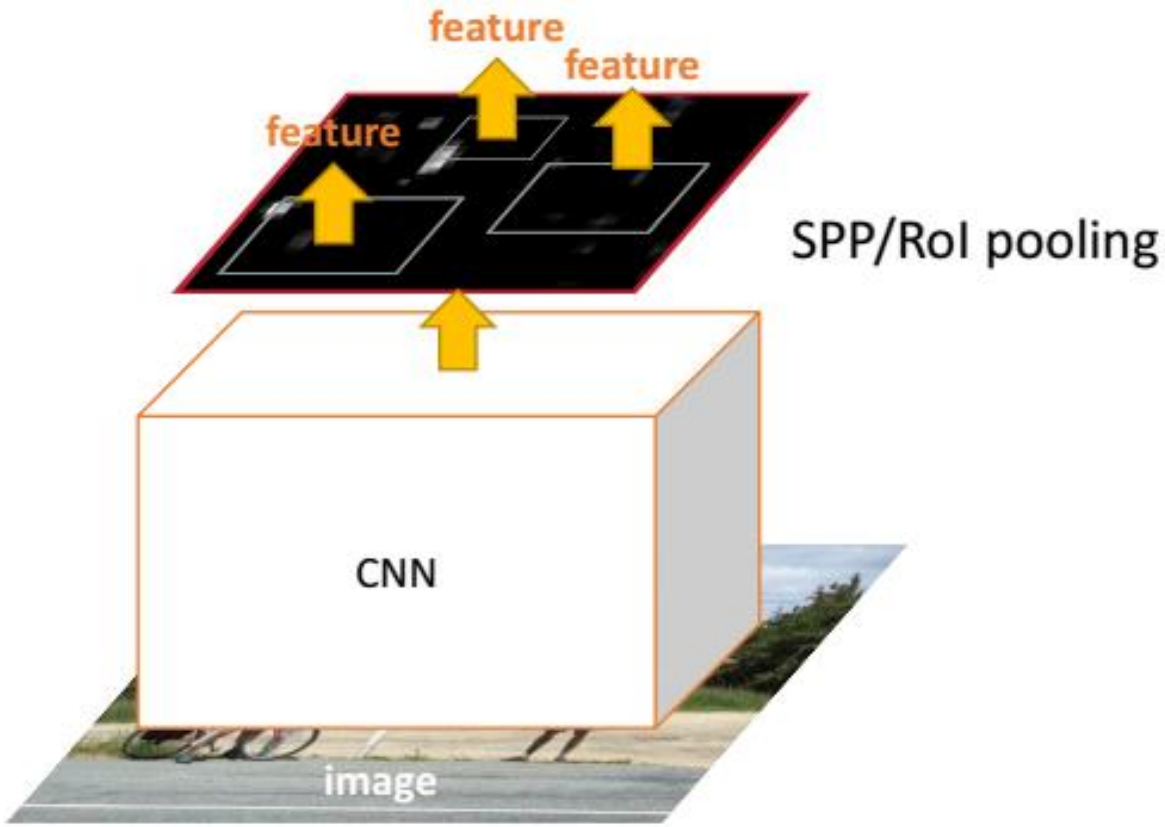


R-CNN vs. Fast R-CNN/SPP-Net



R-CNN

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



SPP-net & Fast R-CNN (the same forward pipeline)

- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features

Fast RCNN: Key Ideas

- Differentiable ROI Pooling
  - Improves accuracy
- Fast SGD by hierarchical sampling of ROIs
  - Improves speed
- What is still wrong
  - Out of network region proposals

Region Proposal from Feature Maps

- Object detection networks are fast (0.2s)…

- But what about region proposal?
  - Selective Search [Uijlings et al. ICCV 2011]: 2s per image
  - EdgeBoxes [Zitnick & Dollar. ECCV 2014]: 0.2s per image

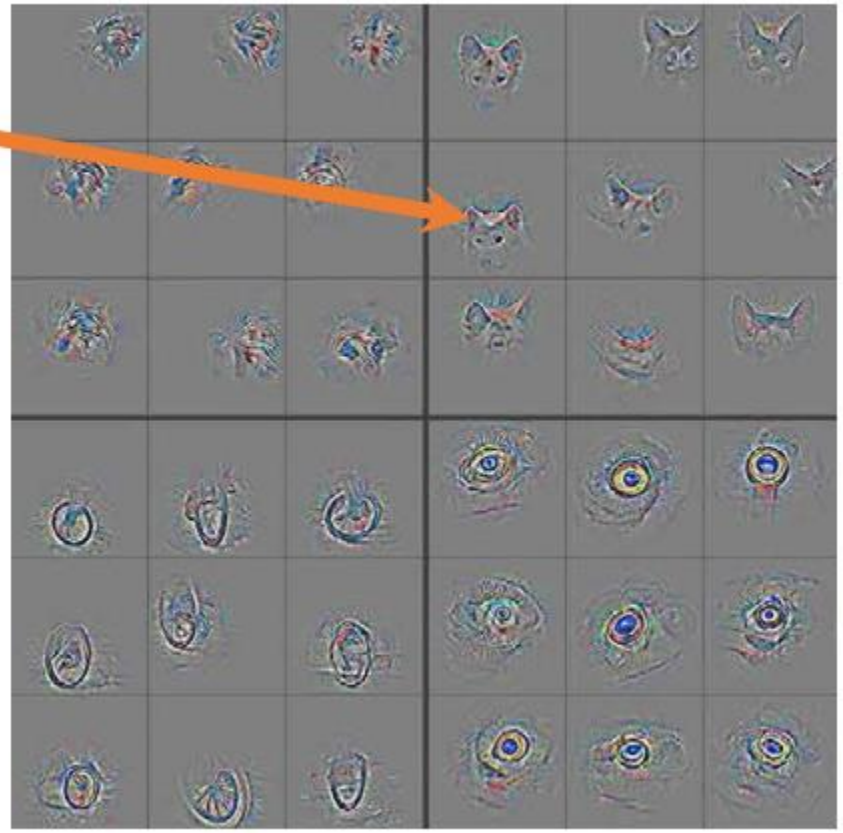
- Can we do region proposal on the same set of feature maps?

Region Proposal from Feature Maps

- By decoding one response at a single pixel, we can still roughly see the object outline*
- Finer localization information has been encoded in the channels of a convolutional feature response
- Extract this information for better localization…

* Zeiler & Fergus's method traces unpooling information so the visualization involves more than a single response. But other visualization methods reveal similar patterns.

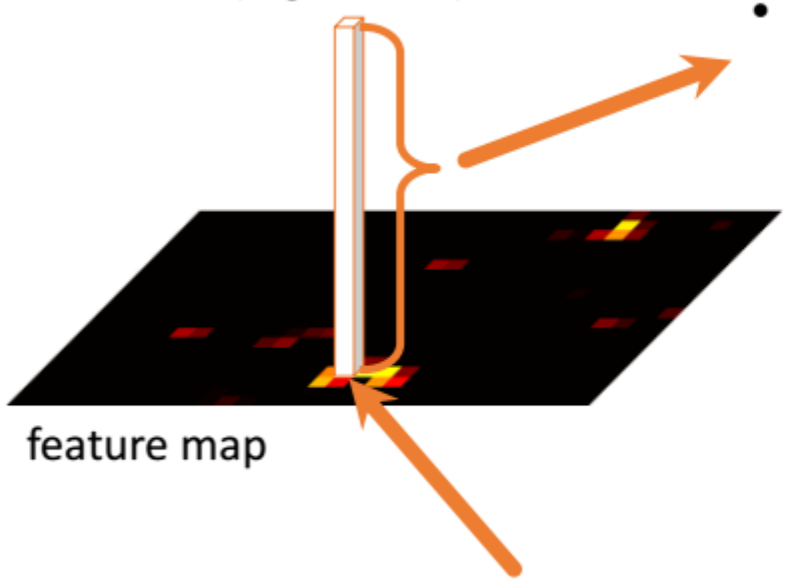
Revisiting visualizations from Zeiler & Fergus



Region Proposal from Feature Maps

image

a feature vector
(e.g., 256-d)



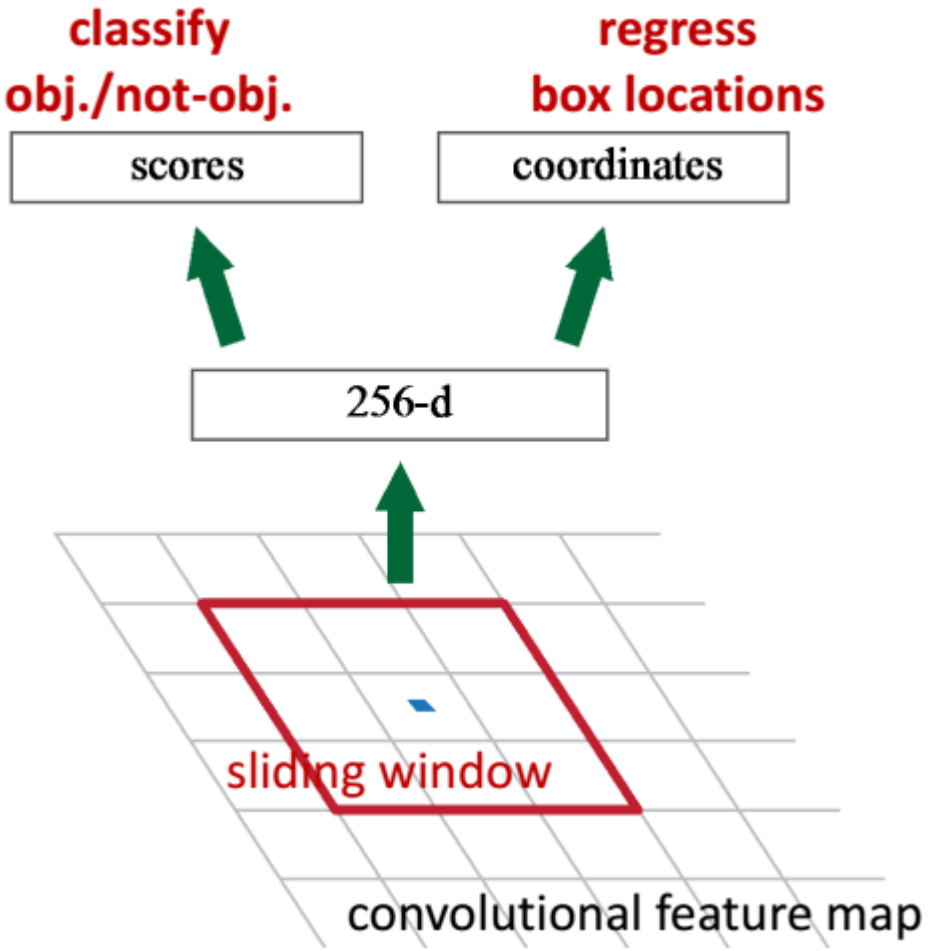
feature map

- The channels of this feature vector encodes finer localization information

- The spatial position of this feature provides coarse locations

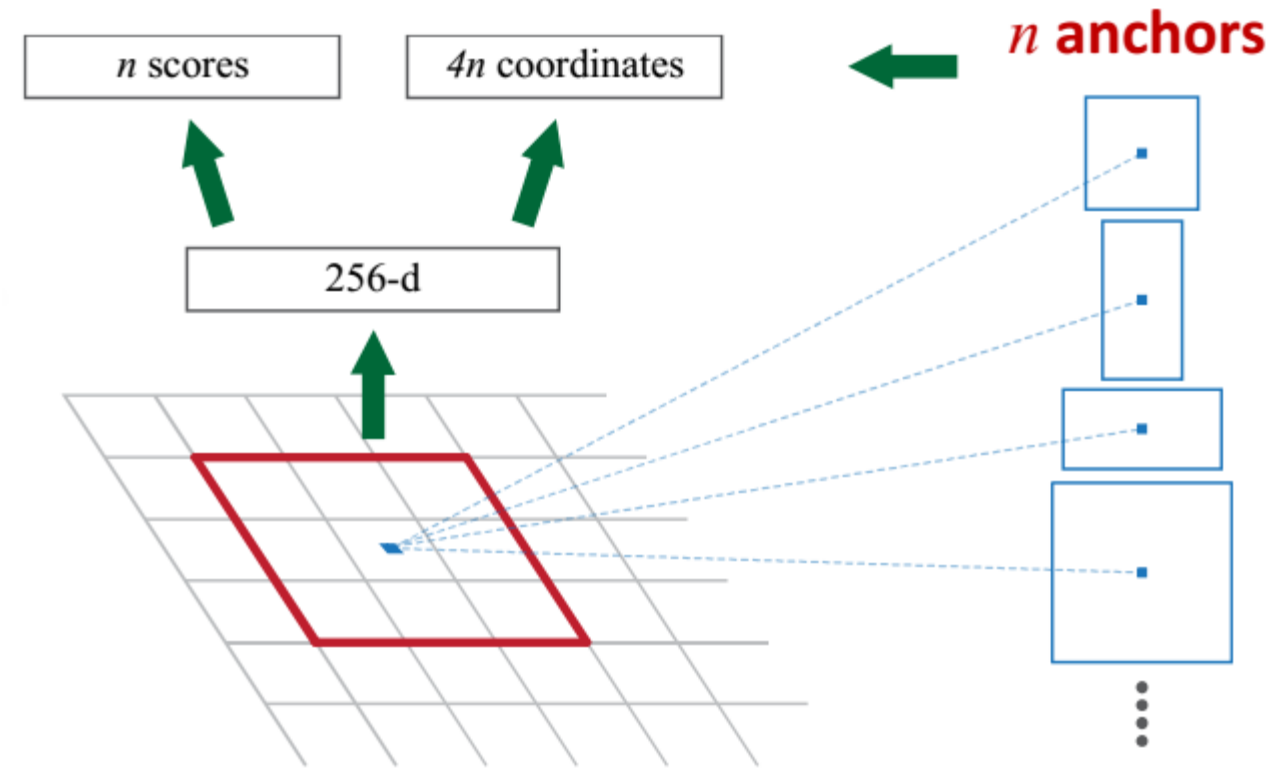
Region Proposal Network

- 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



Region Proposal Network

- **Anchors**: pre-defined reference boxes
- Box regression is with reference to anchors: regressing an anchor box to a ground-truth box
- Object probability is with reference to anchors, e.g.:
  - anchors as positive samples: if IoU > 0.7 or IoU is max
  - anchors as negative samples: if IoU < 0.3

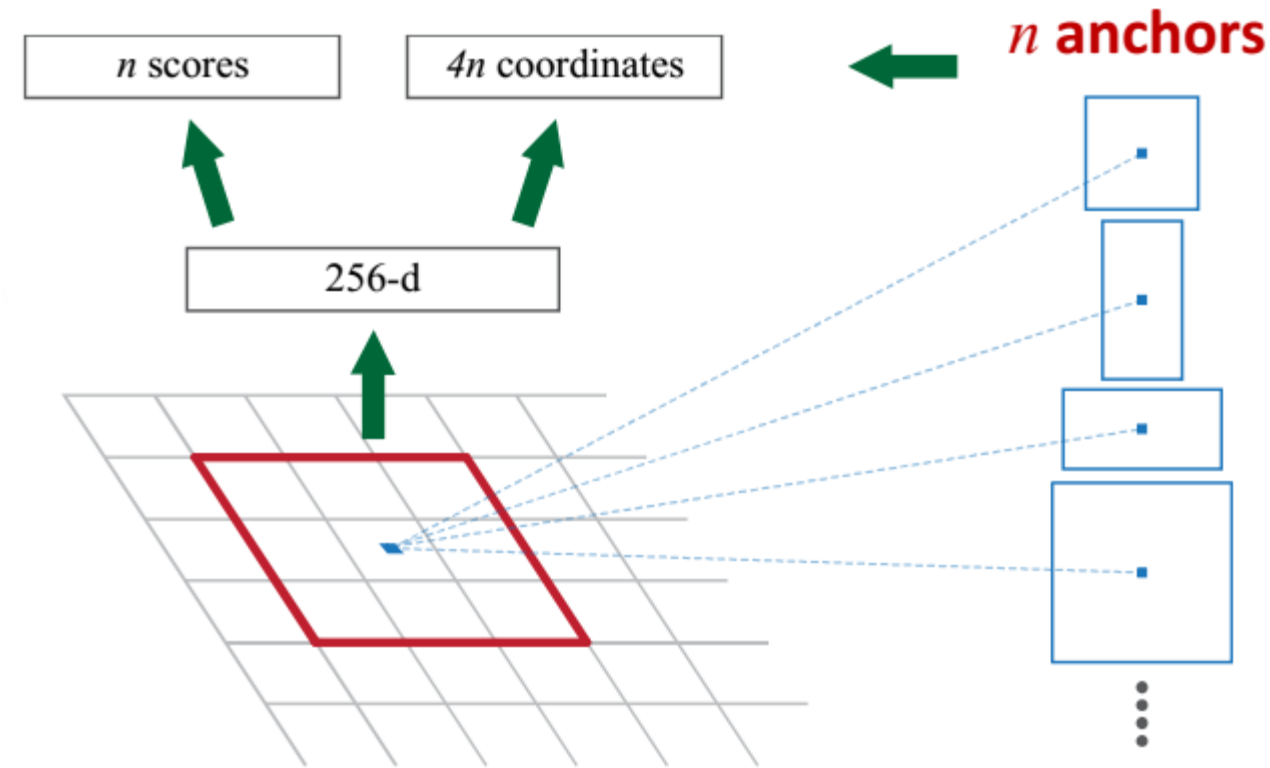


Region Proposal Network

- **Anchors**: pre-defined reference boxes

Translation-invariant anchors:

- the same set of anchors are used at each sliding position
- the same prediction functions (with reference to the sliding window) are used
- a translated object will have a translated prediction

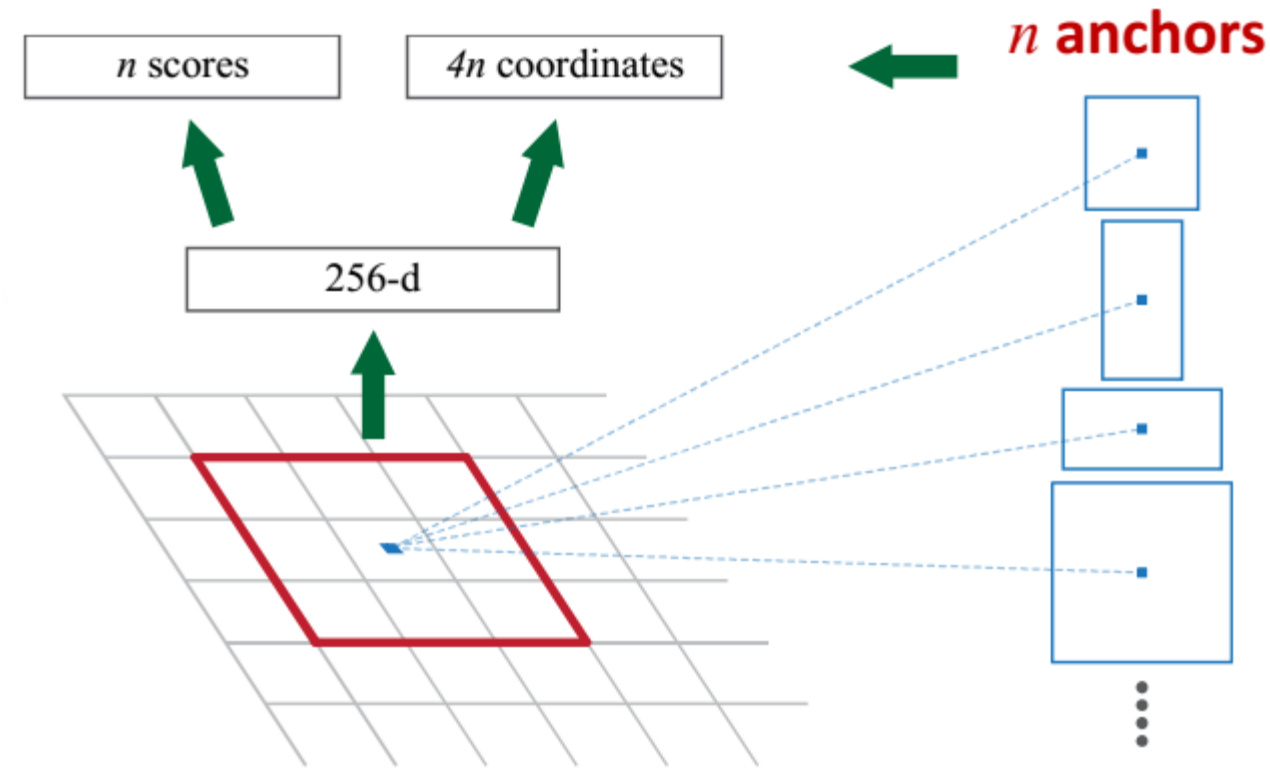


Region Proposal Network

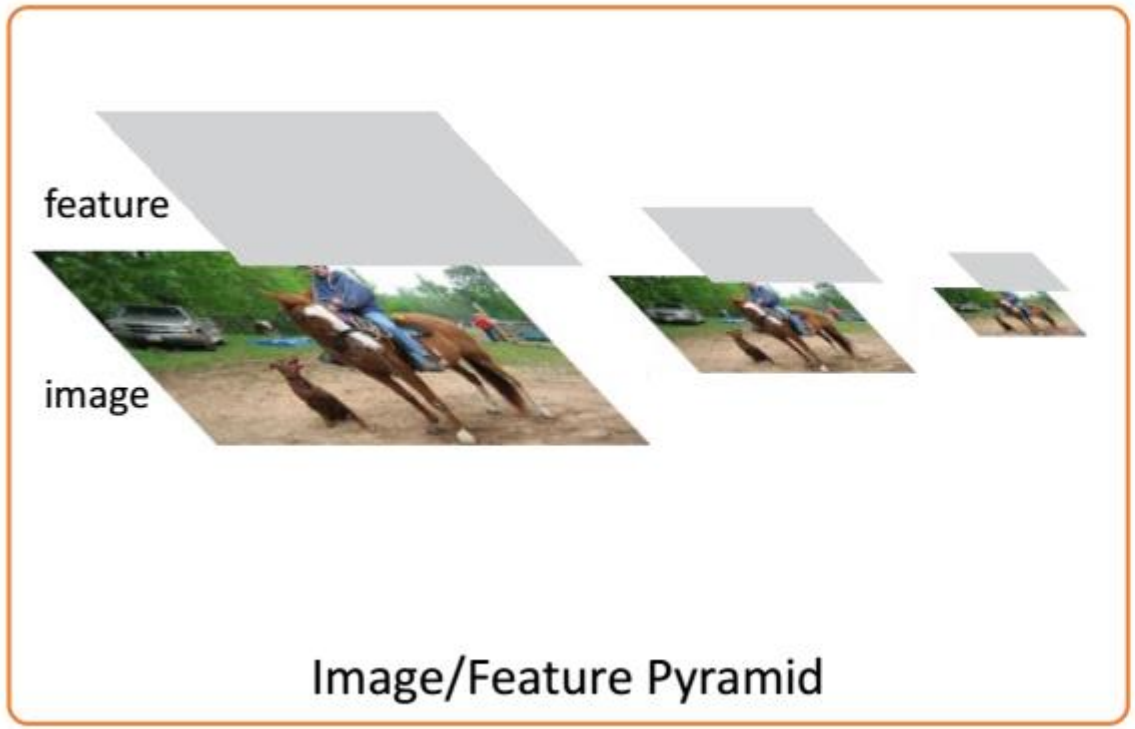
- **Anchors**: pre-defined reference boxes

Multi-scale/size anchors:

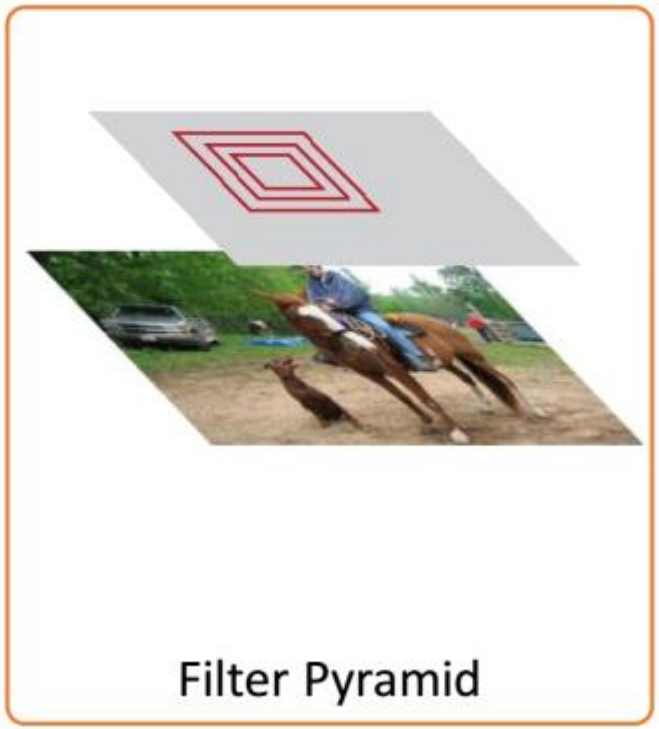
- multiple anchors are used at each position: e.g., 3 scales ($128^2, 256^2, 512^2$) and 3 aspect ratios (2:1, 1:1, 1:2) yield 9 anchors
- each anchor has its own prediction function
- single-scale features, multi-scale predictions



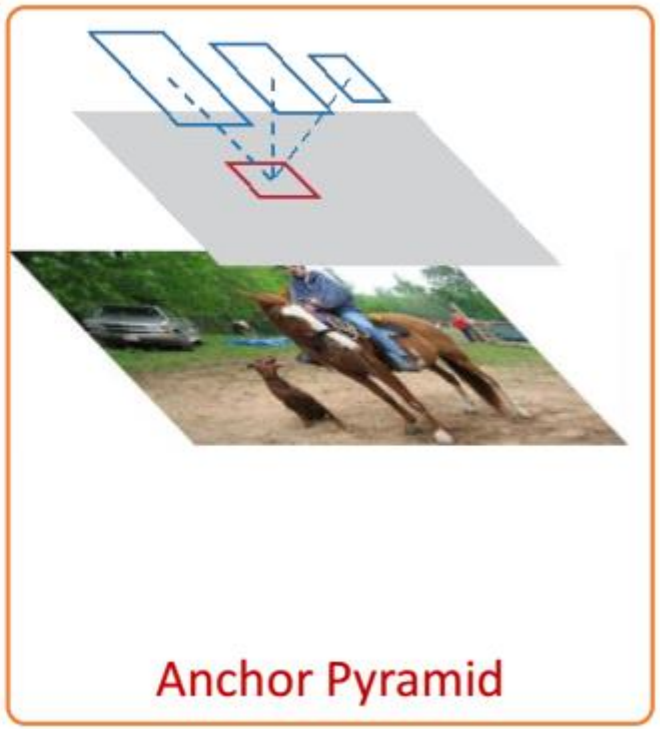
Comparisons of Multi Scale Strategies



Image/Feature Pyramid

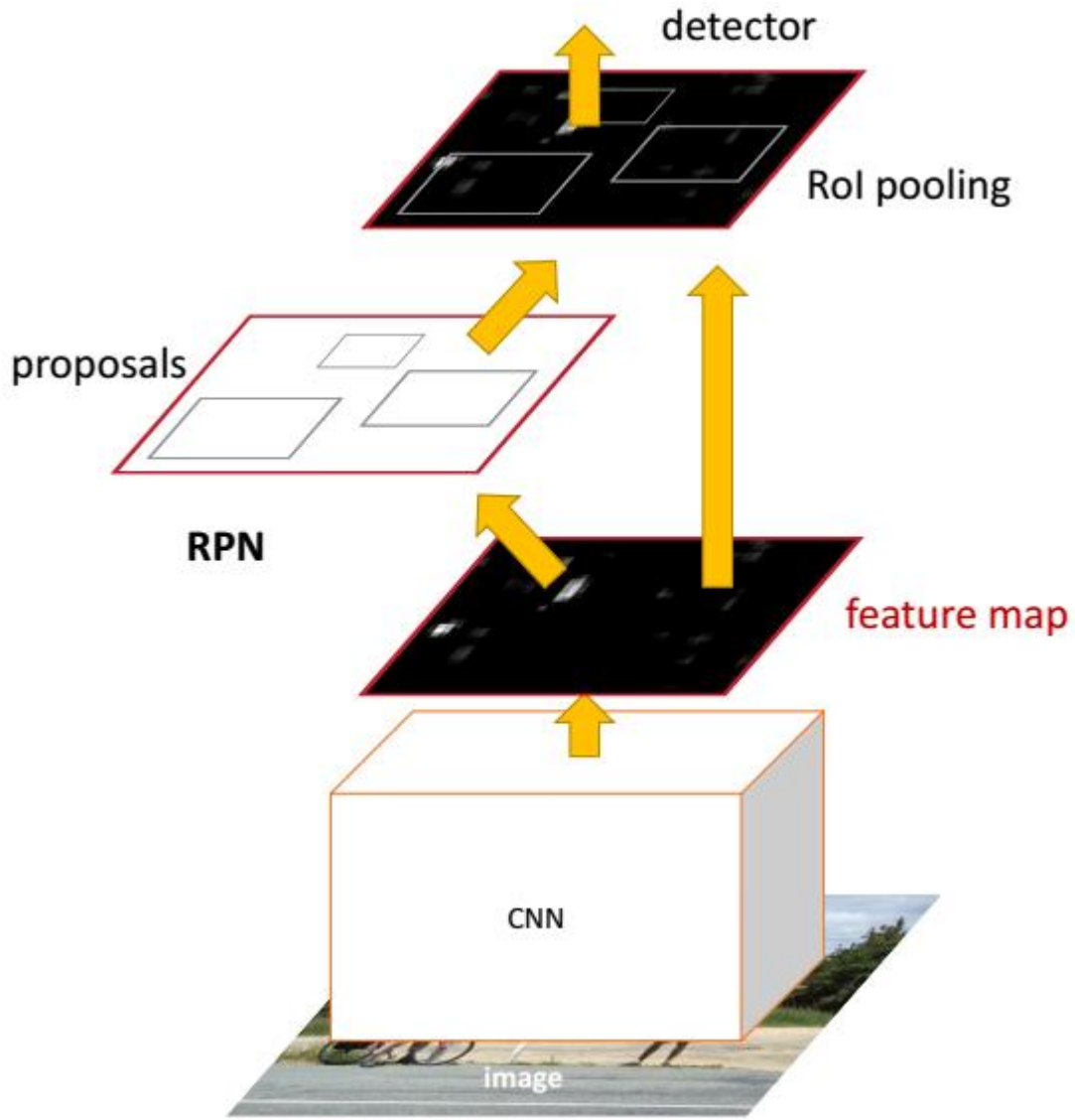


Filter Pyramid



Anchor Pyramid

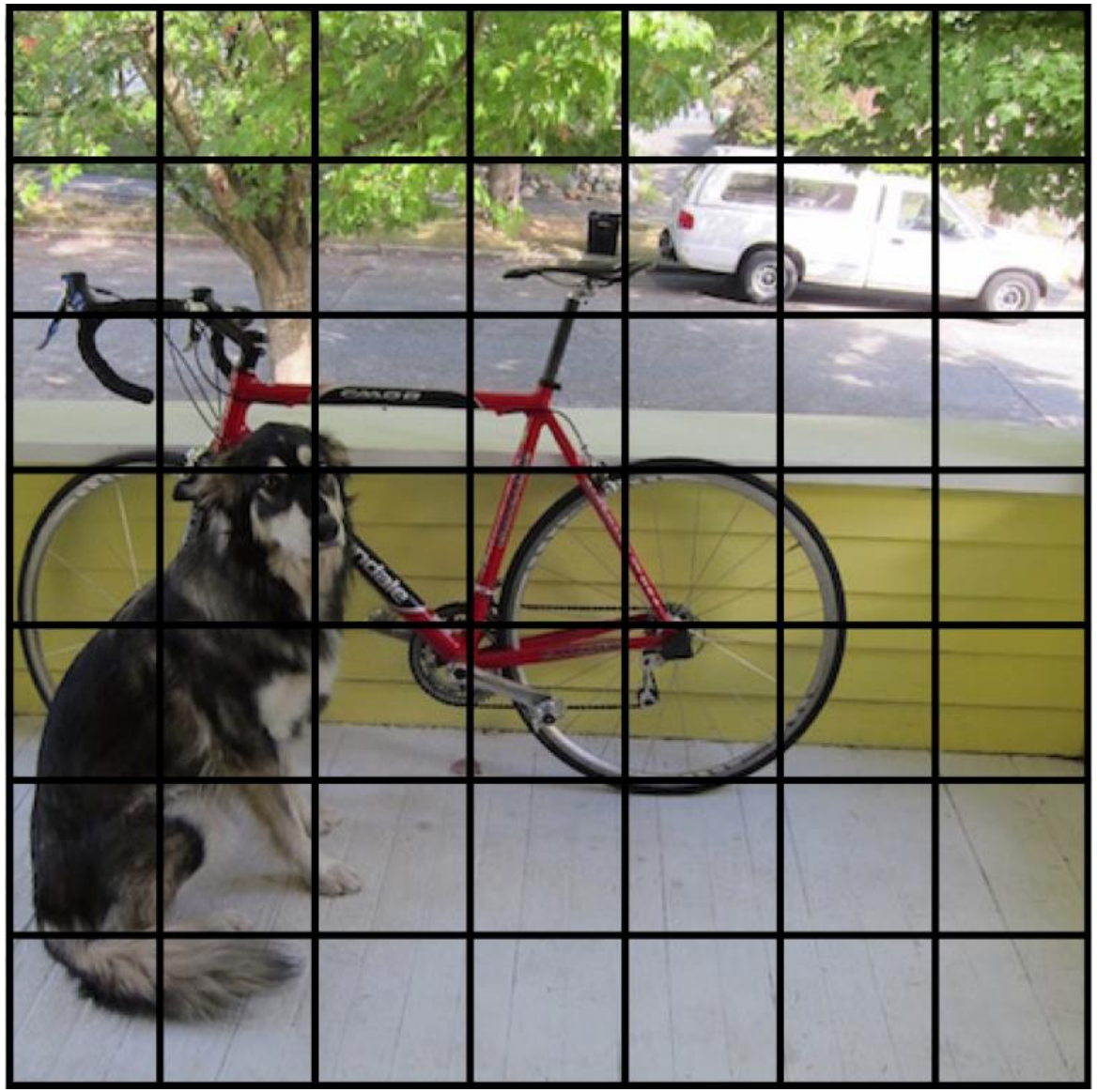
Faster RCNN



YOLO

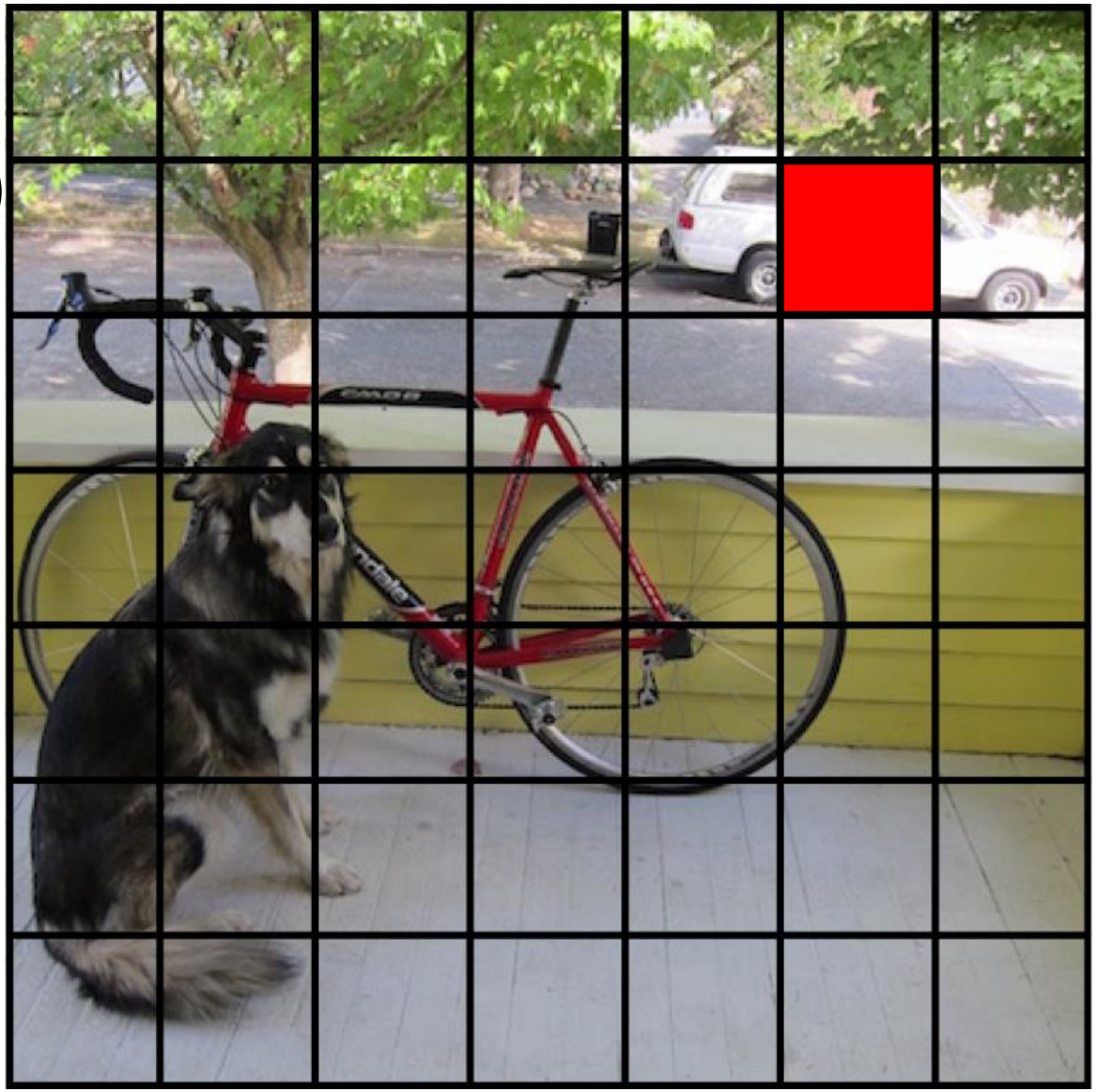
YOLO

Split the image into a grid



YOLO

Each cell predicts boxes and confidences: P(Object)

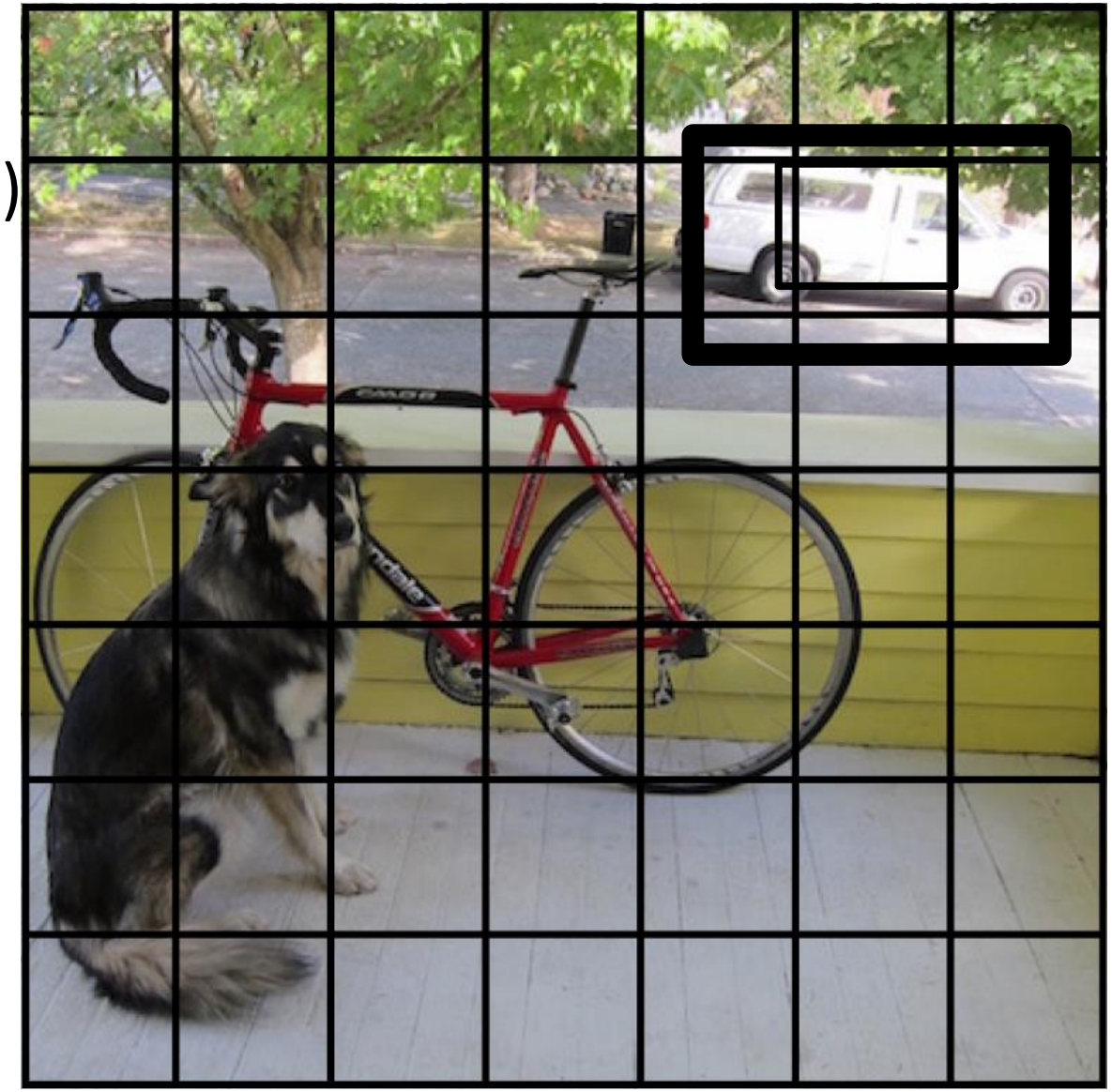


YOLO

Each cell predicts boxes and confidences: P(Object)

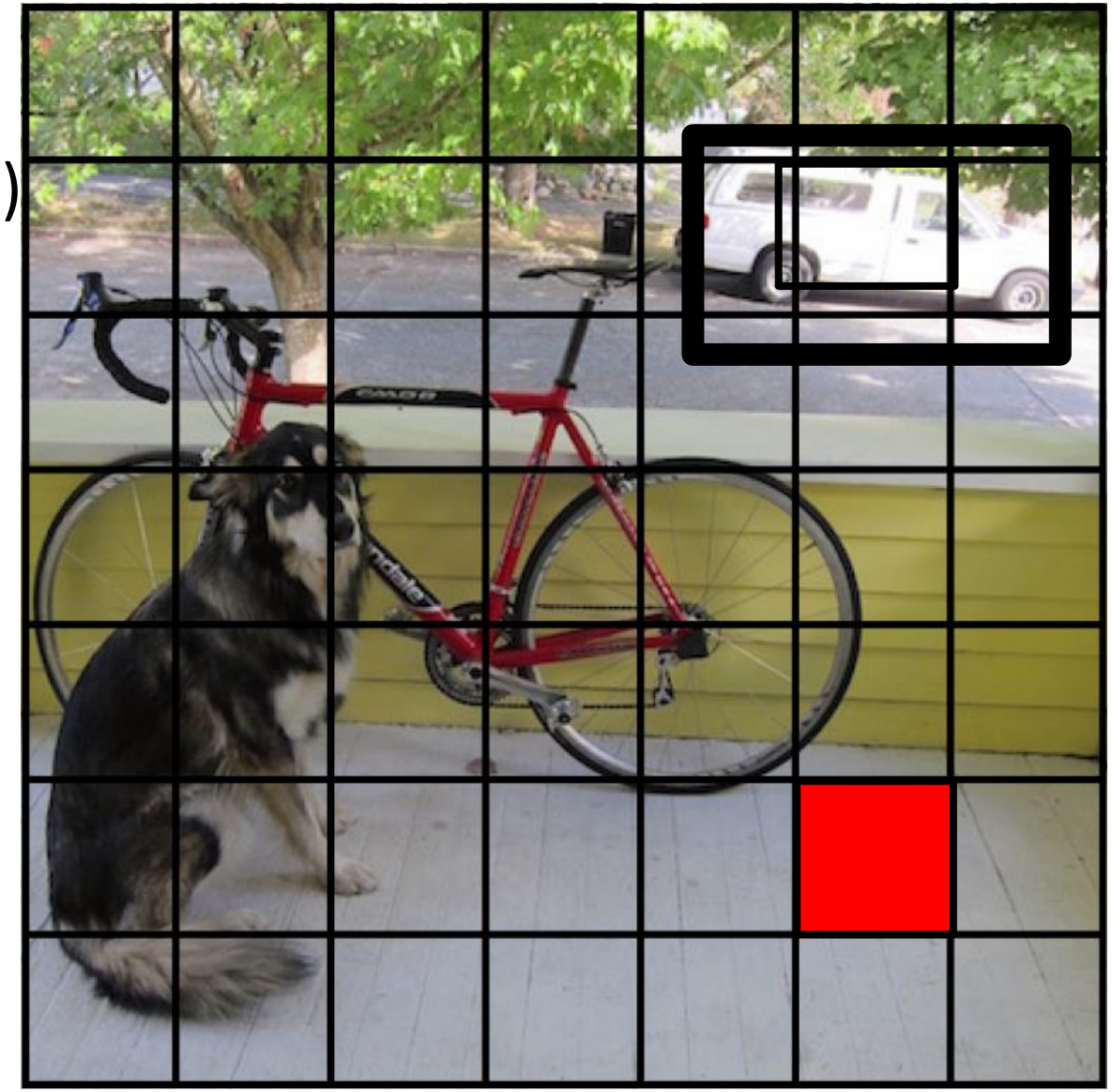
YOLO

Each cell predicts boxes and confidences: P(Object)



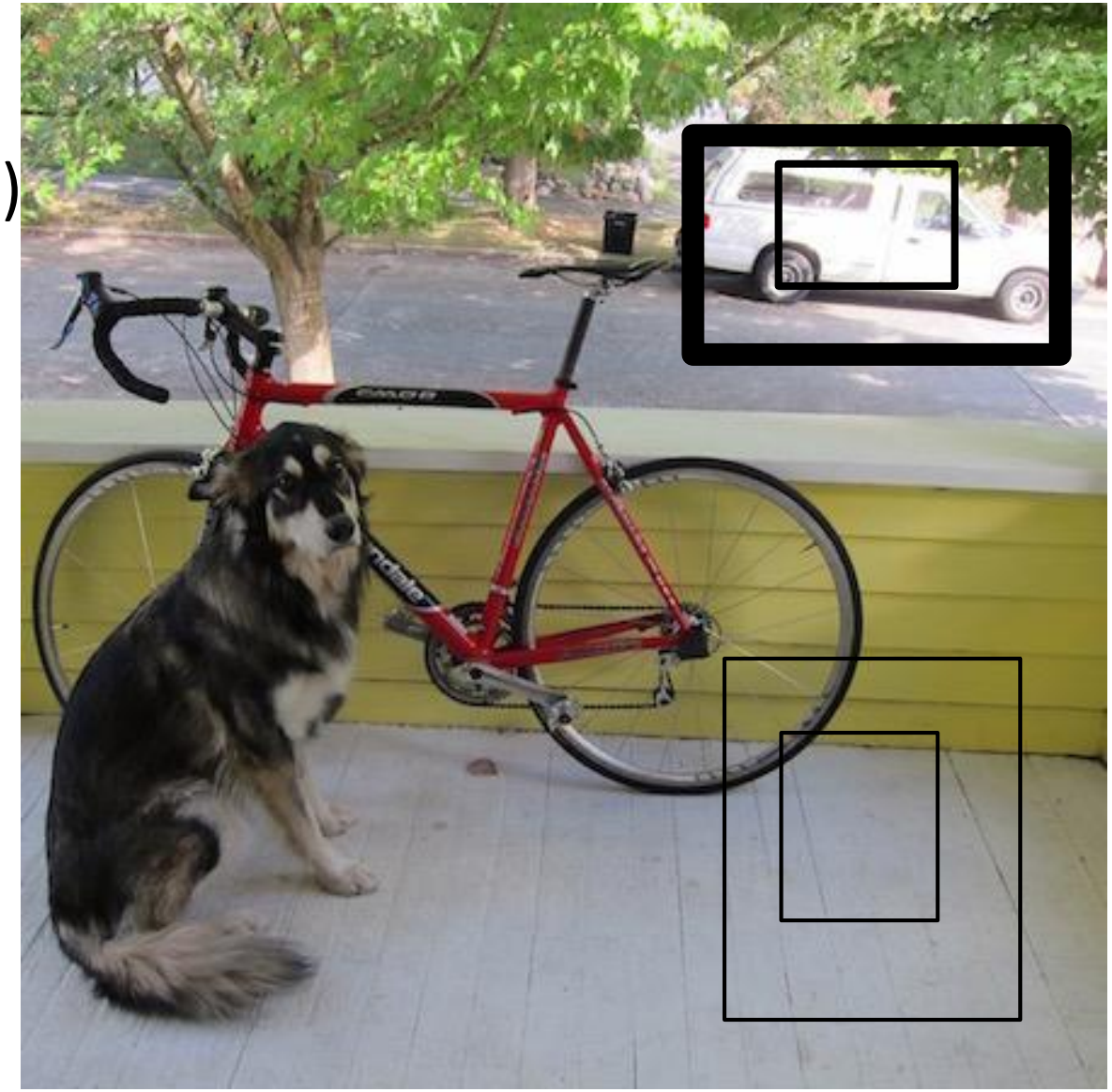
YOLO

Each cell predicts boxes and confidences: P(Object)



YOLO

Each cell predicts boxes and confidences: P(Object)

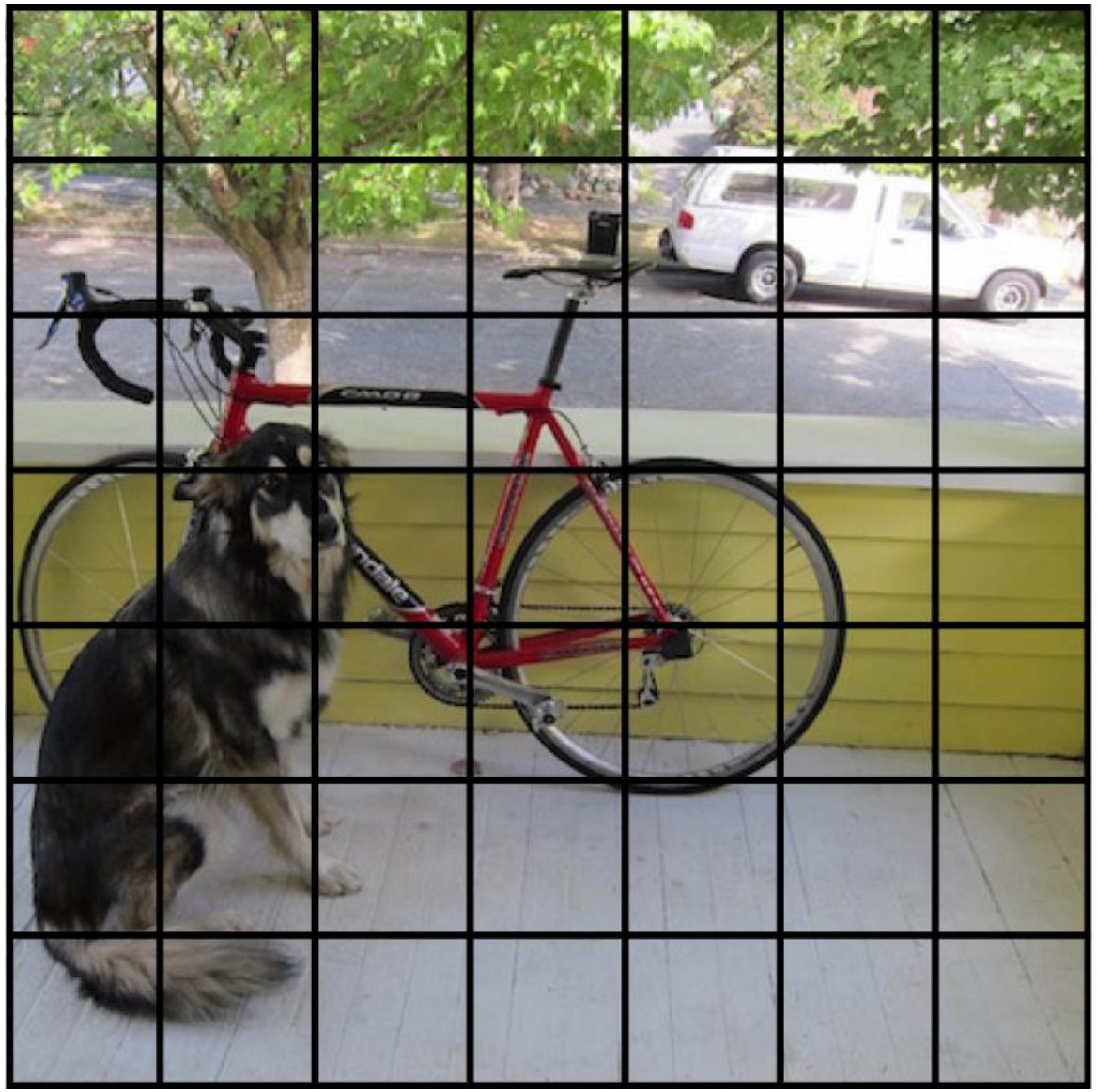


YOLO

Each cell predicts boxes and confidences: P(Object)

YOLO

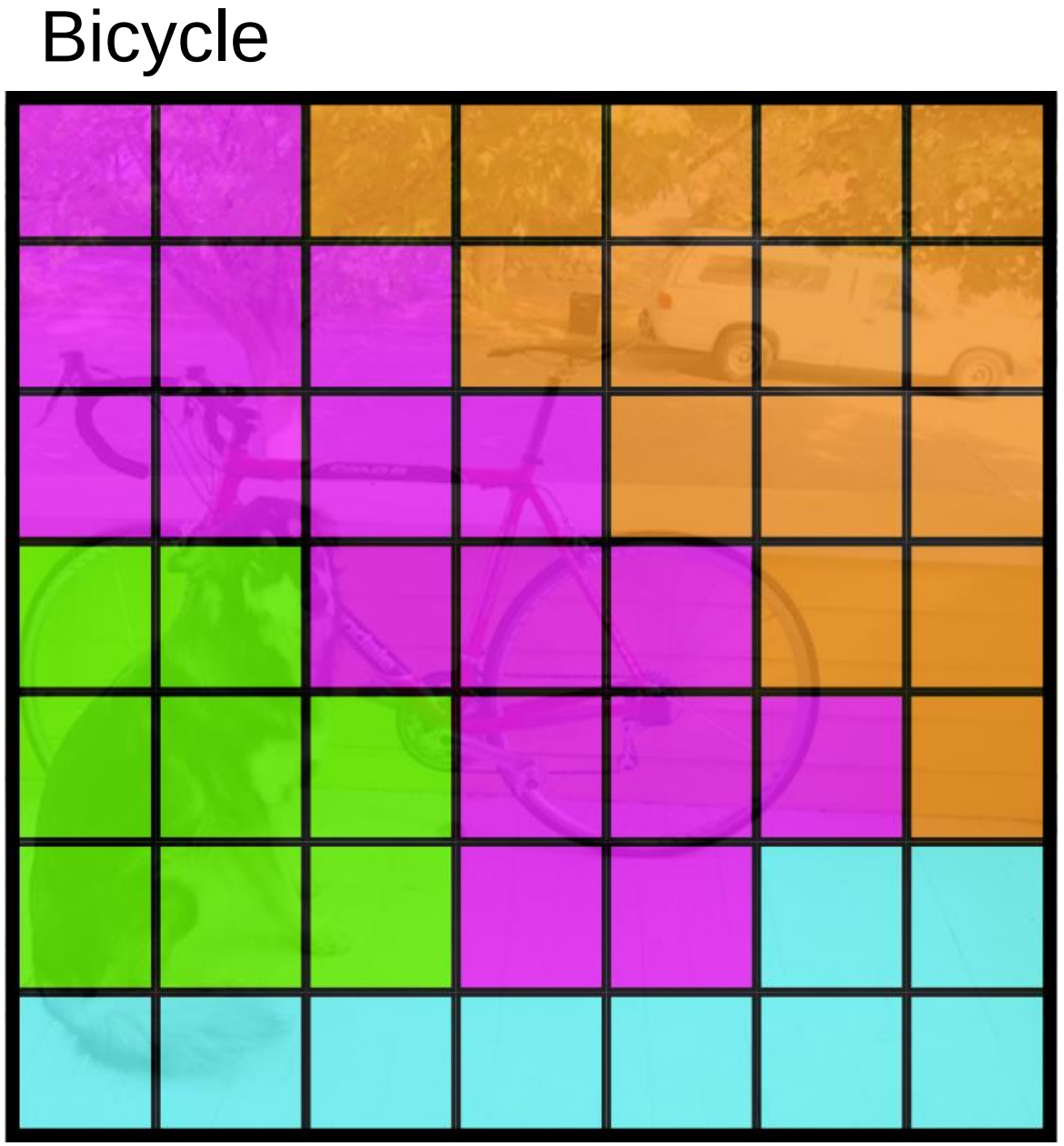
Each cell also predicts a class probability.



YOLO

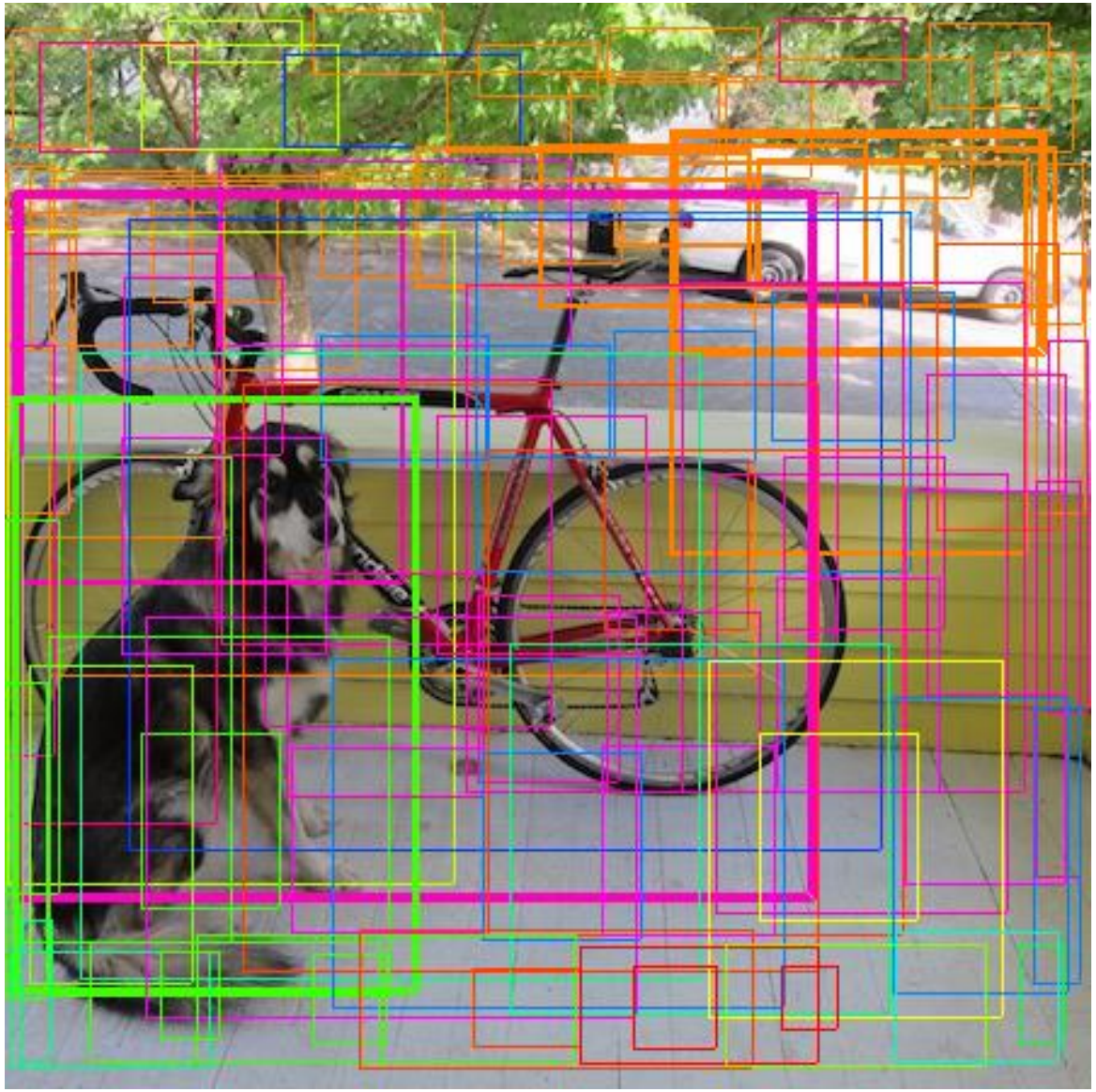
Each cell also predicts a class probability.

Conditioned on object: P(Car | Object)



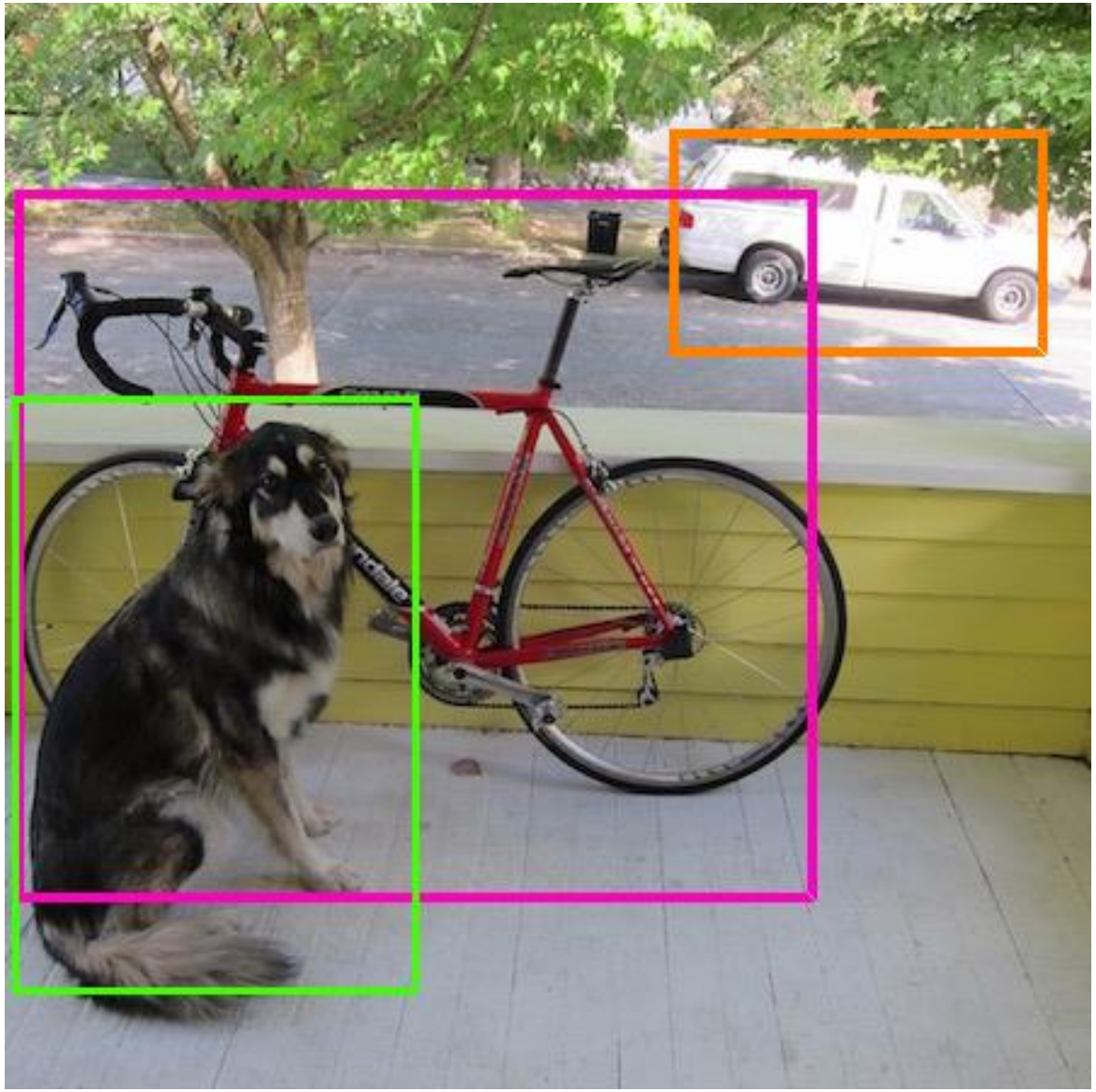
YOLO

Then combine the box and class predictions.



YOLO

Finally do NMS and threshold detections

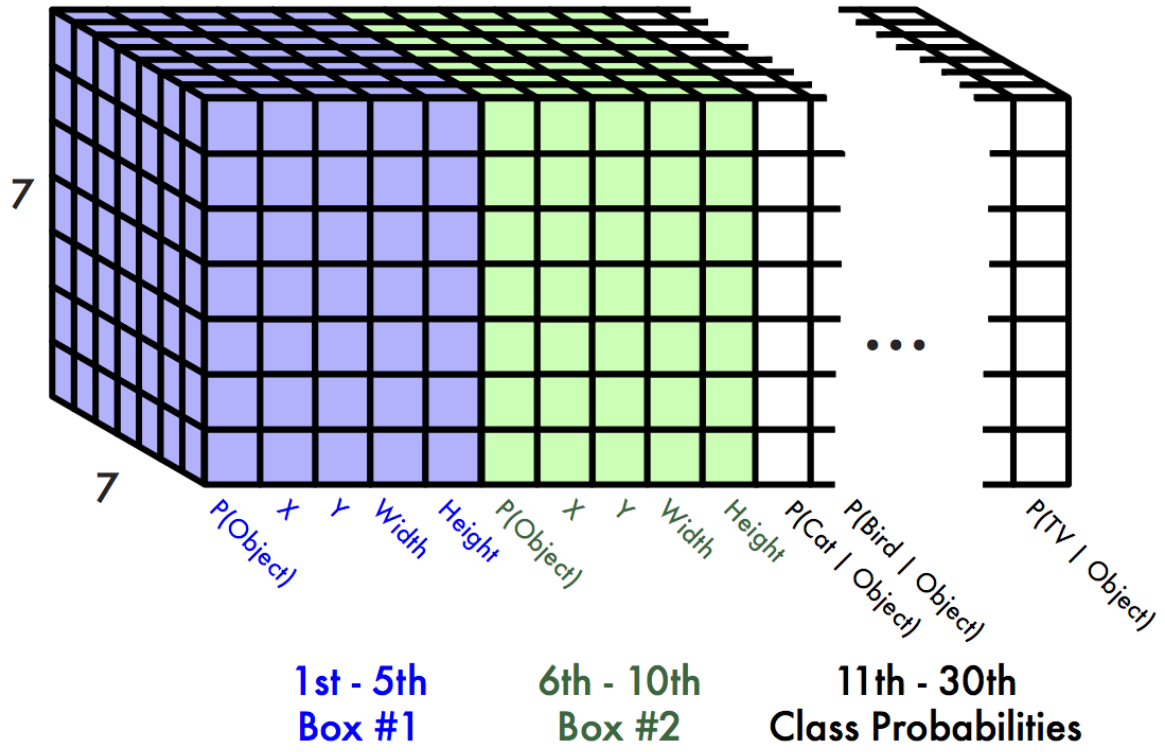


YOLO

This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



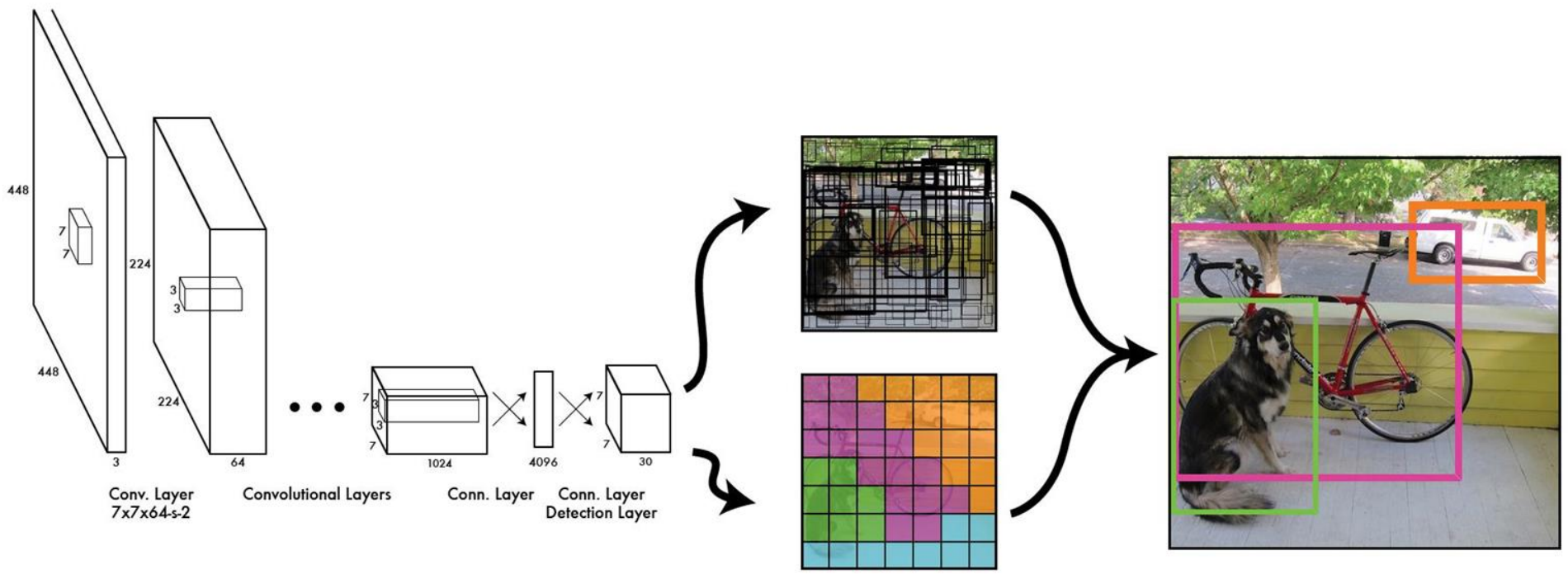
For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

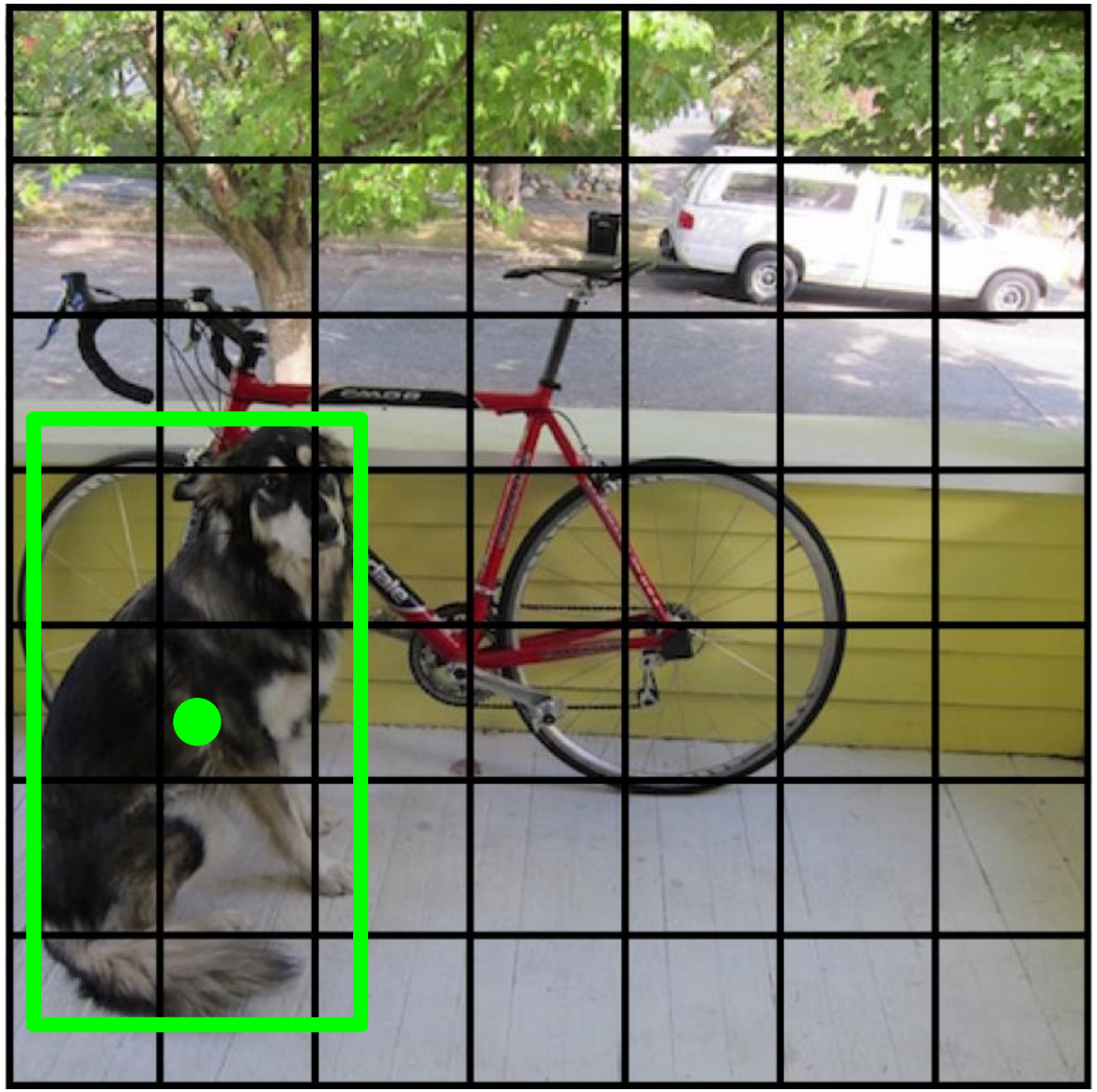
YOLO

Thus we can train one neural network to be a whole detection pipeline



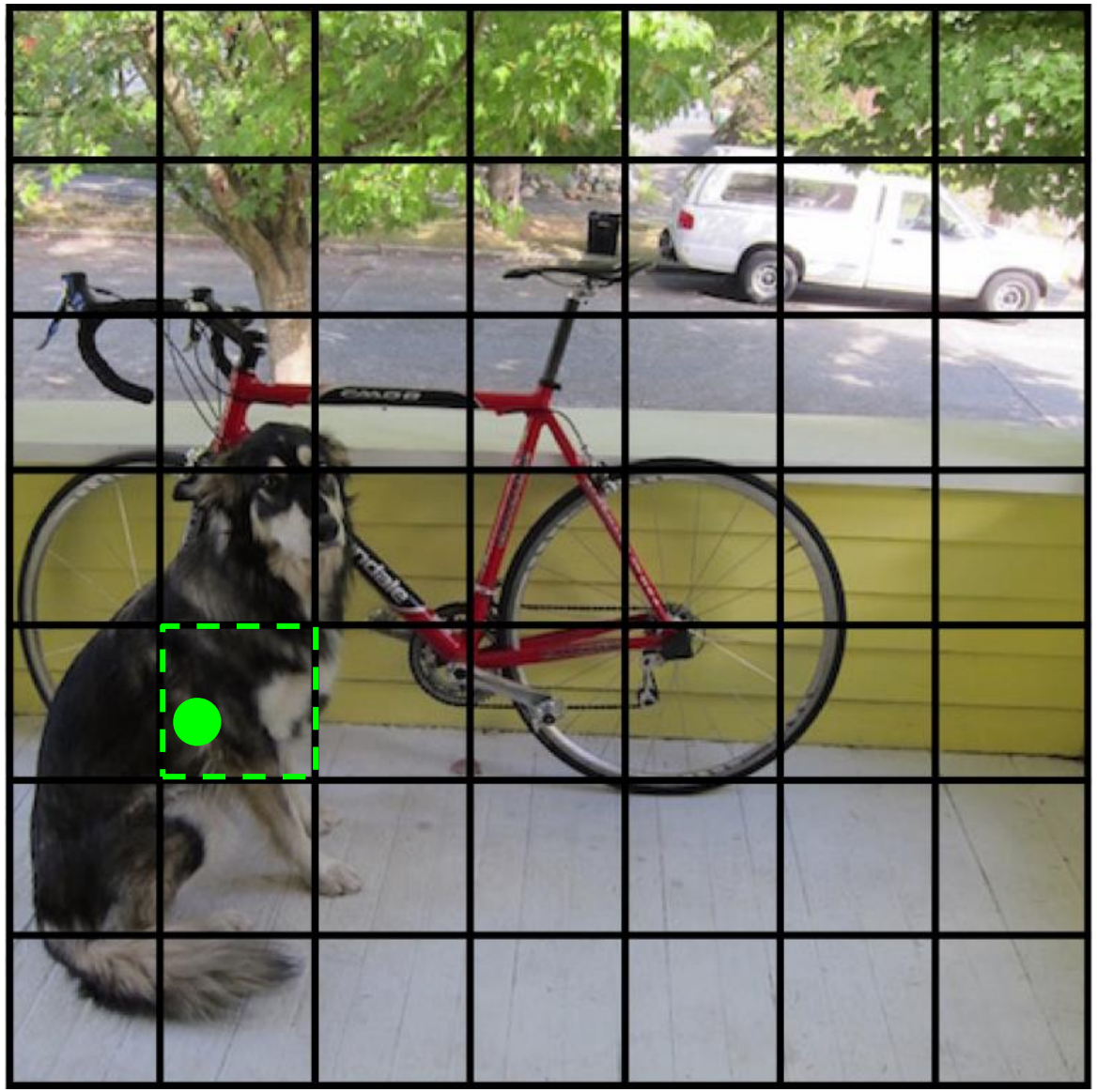
YOLO

During training, match example to the right cell



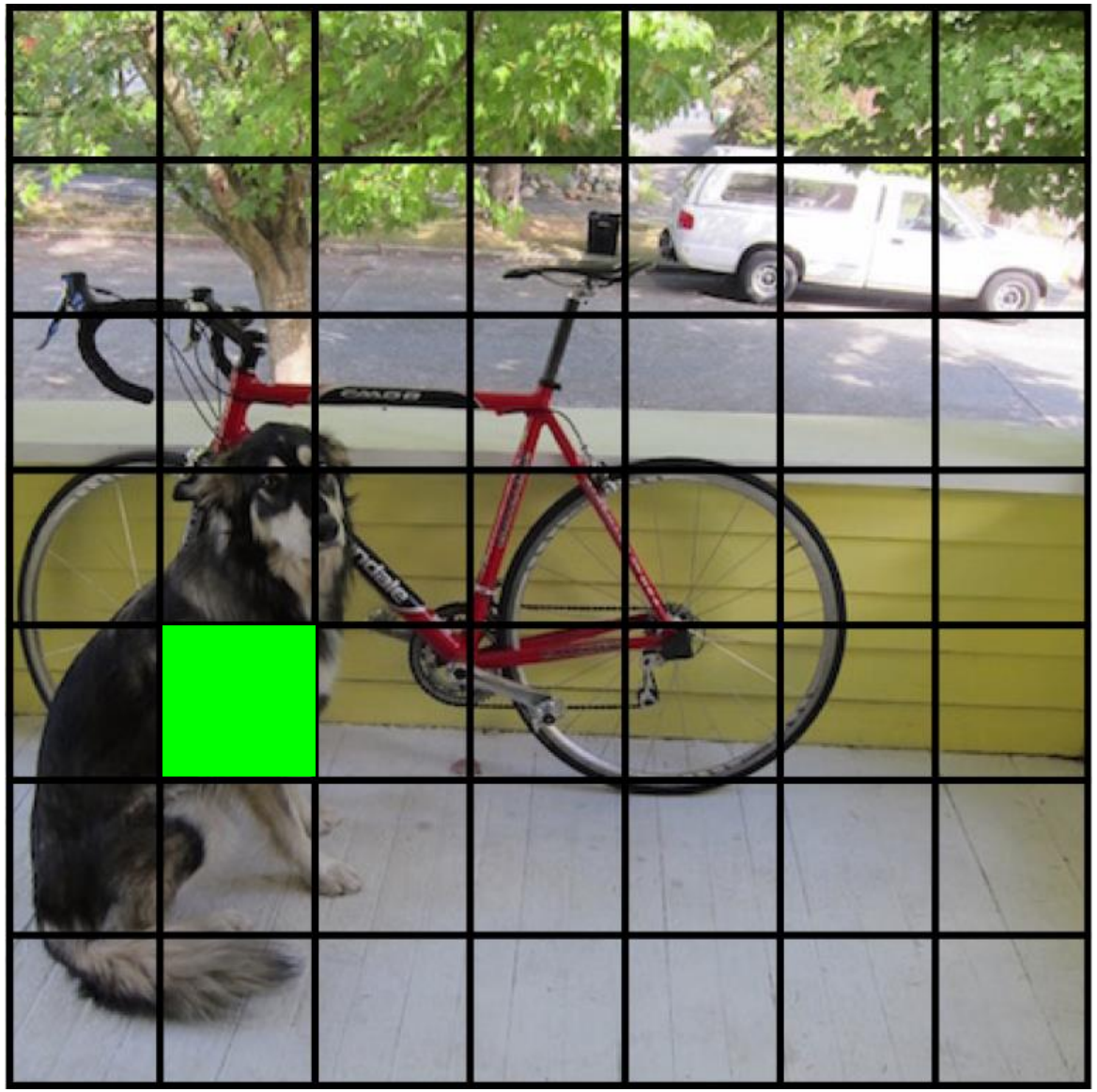
YOLO

During training, match example to the right cell



YOLO

Adjust that cell's class prediction



Dog = 1

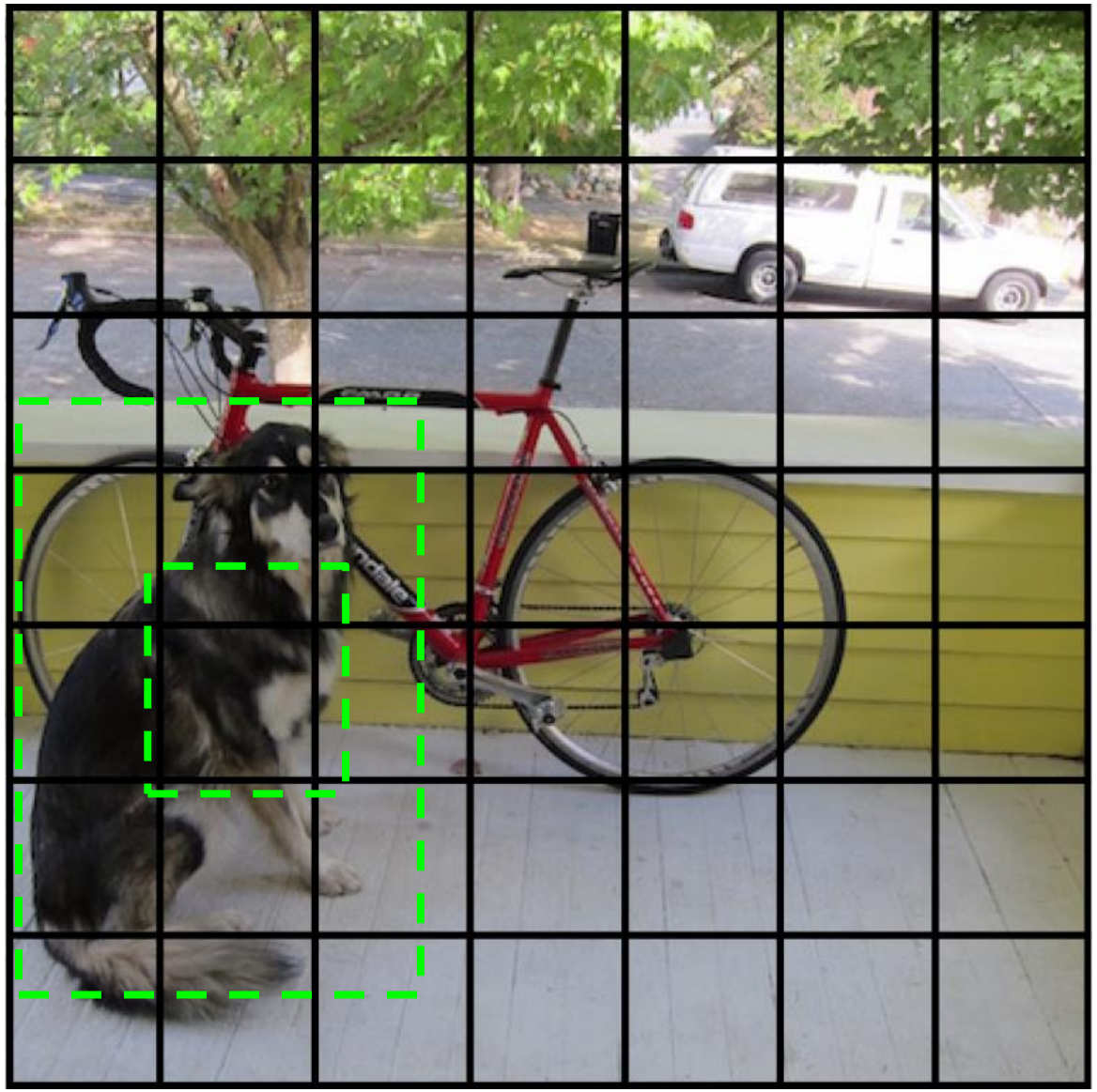
Cat = 0

Bike = 0

…

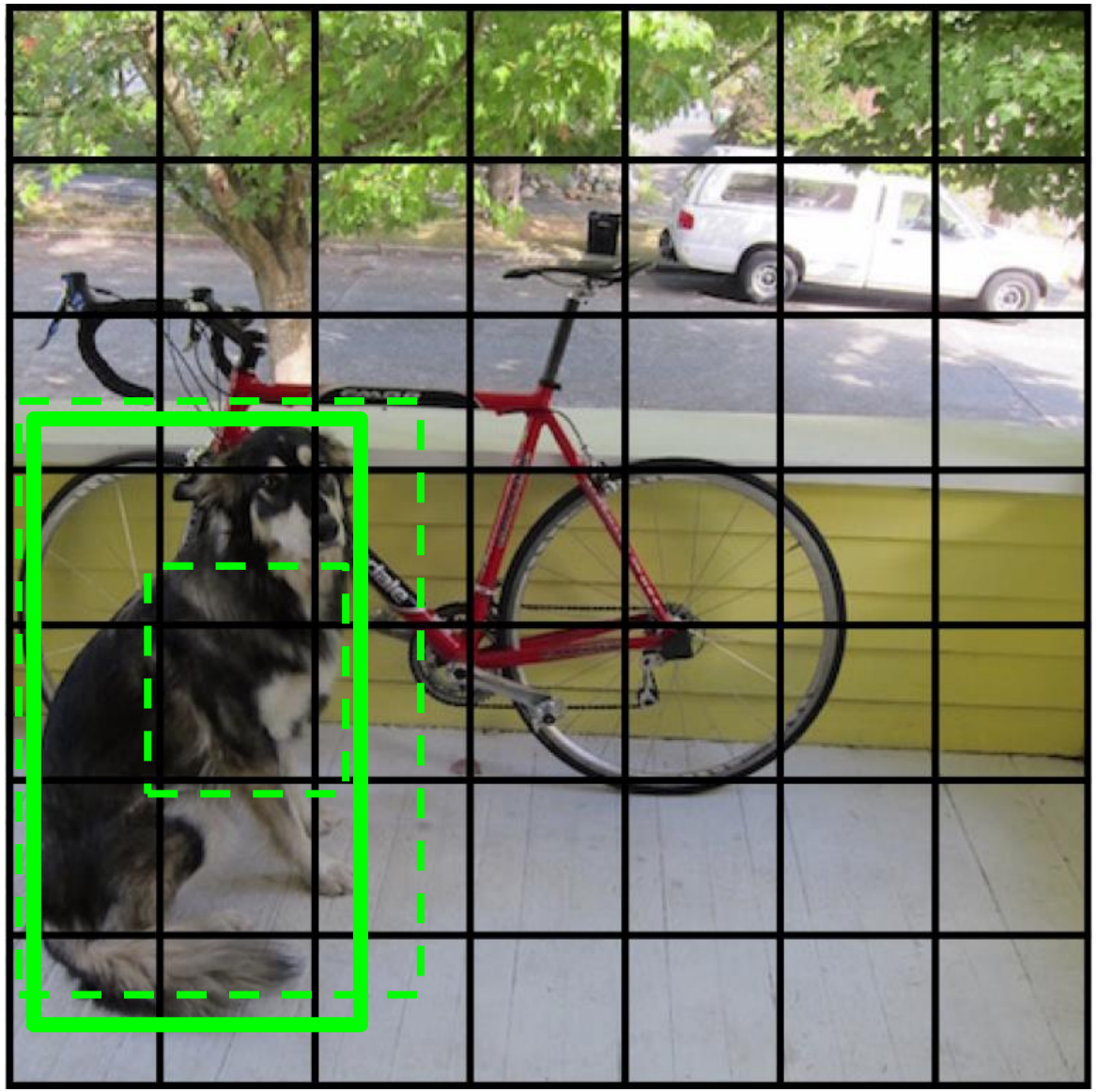
YOLO

Look at that cell's predicted boxes



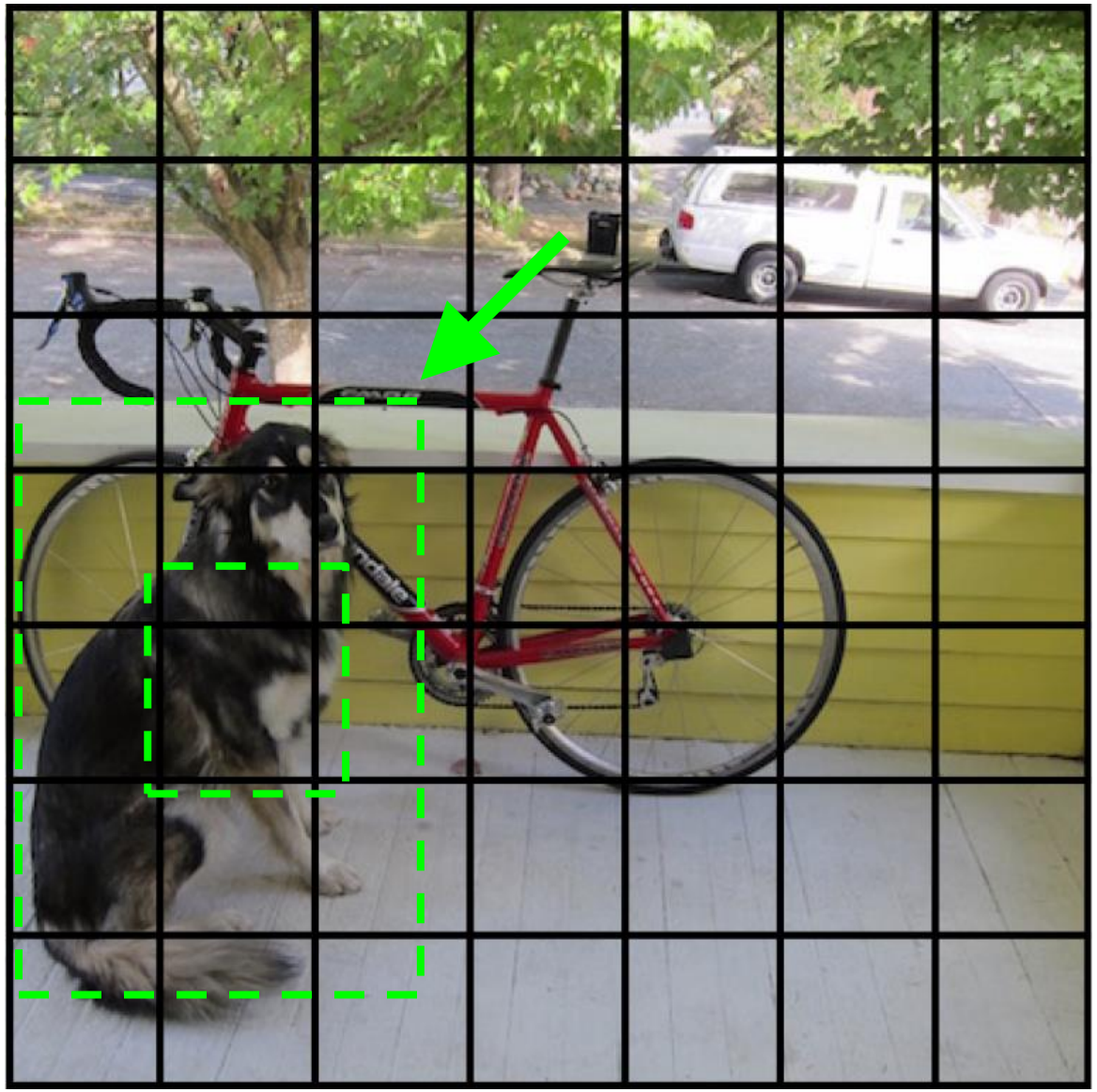
YOLO

Find the best one, adjust it, increase the confidence



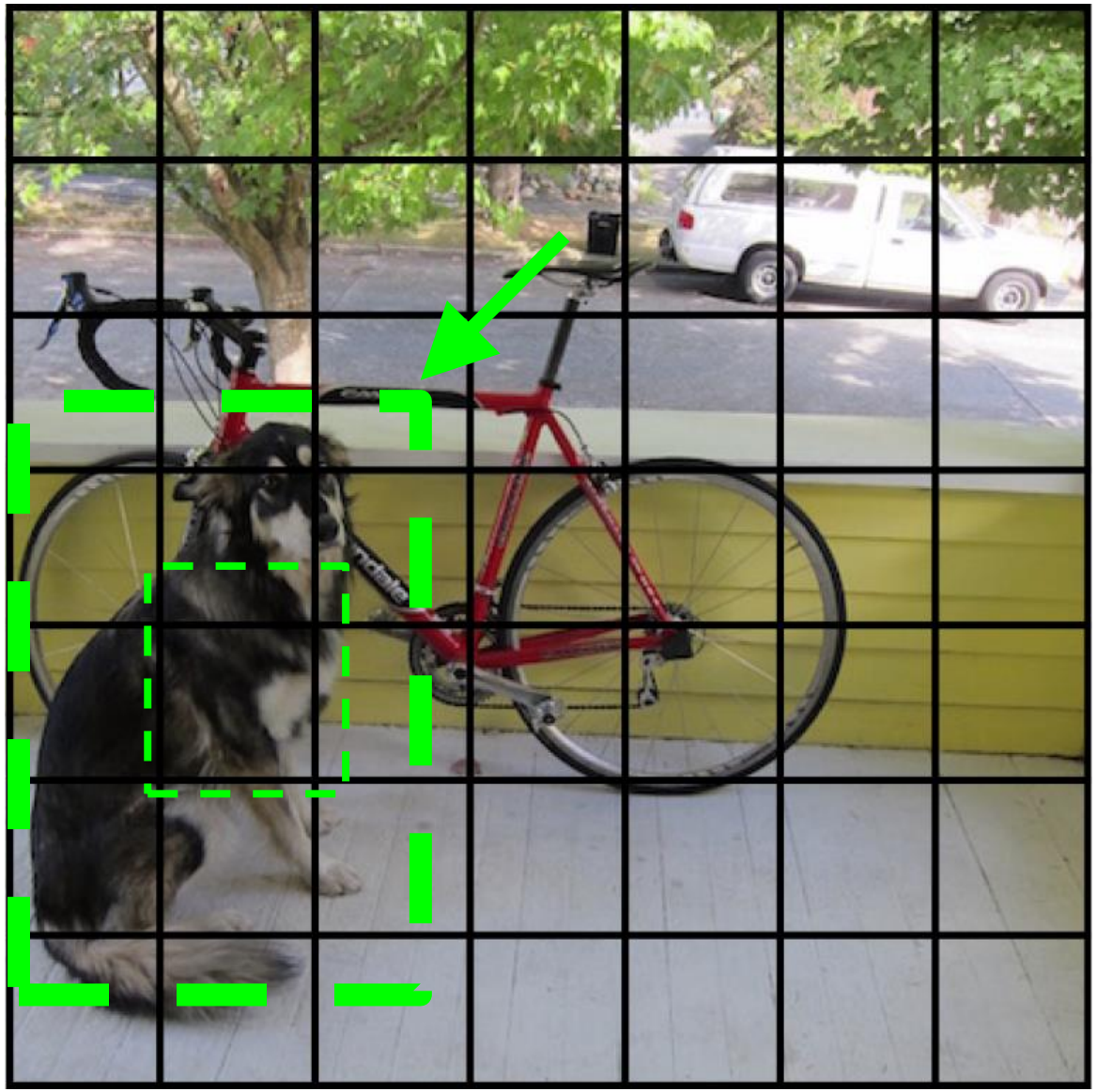
YOLO

Find the best one, adjust it, increase the confidence



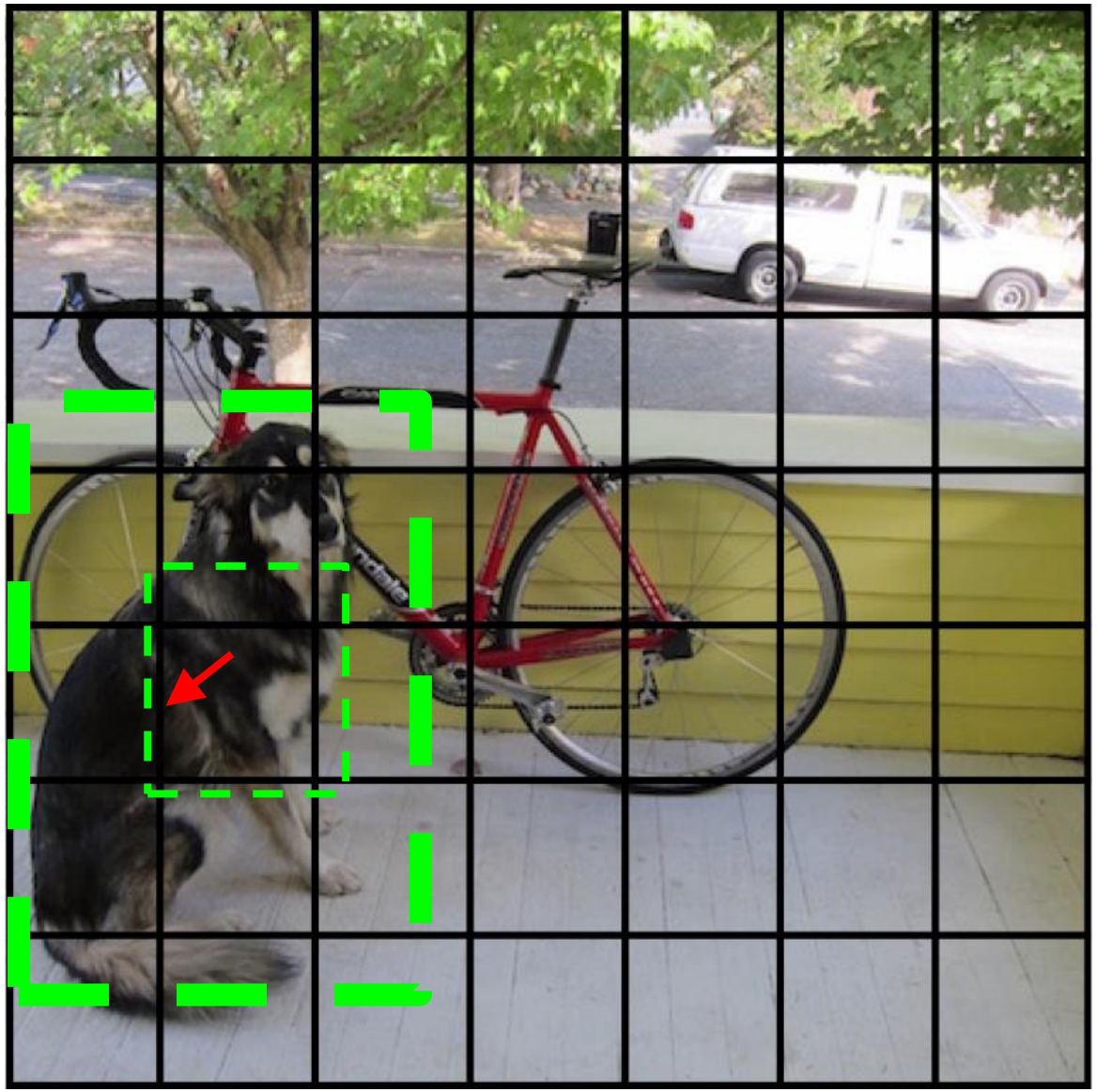
YOLO

Find the best one, adjust it, increase the confidence



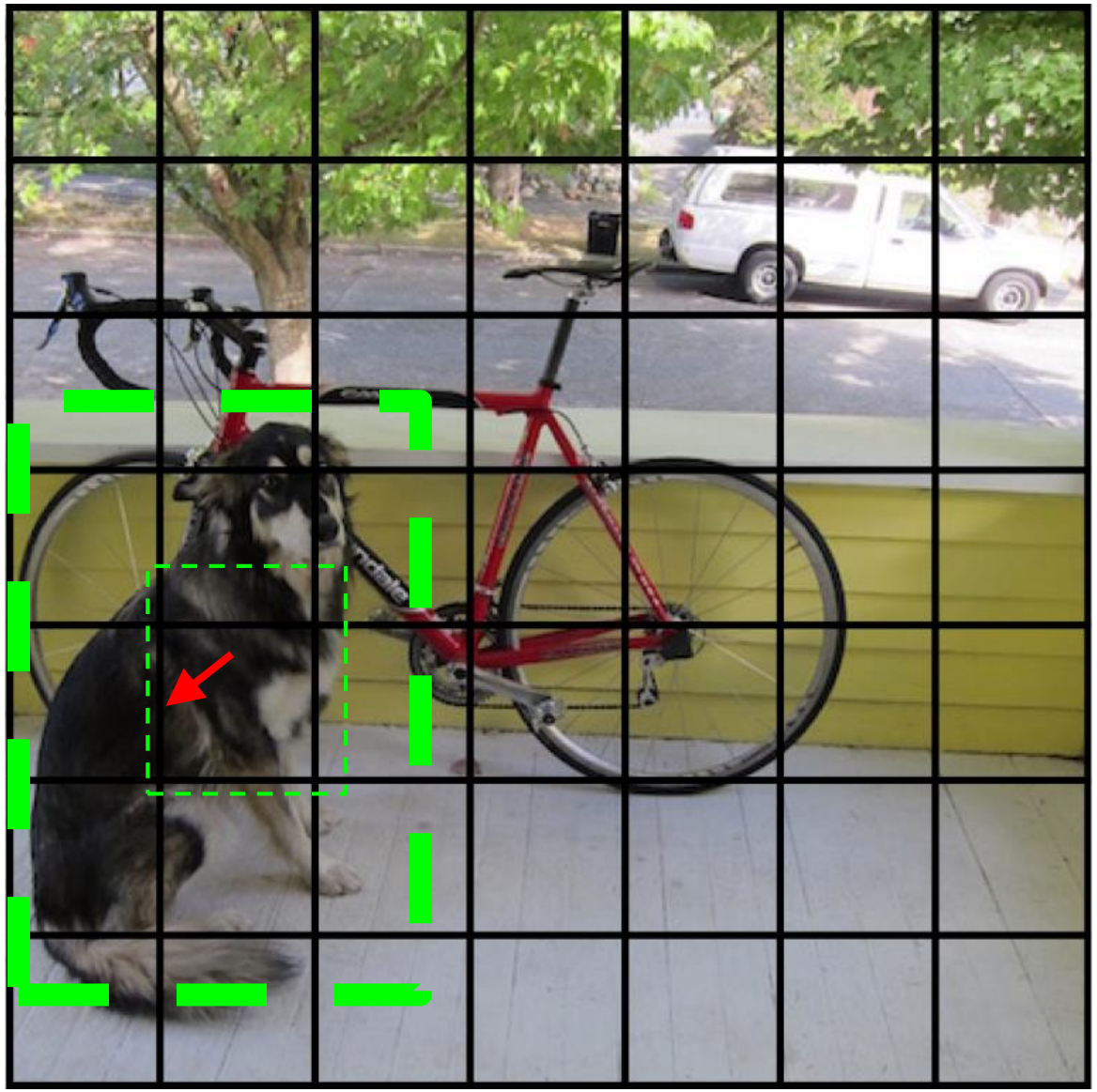
YOLO

Decrease the confidence of other boxes



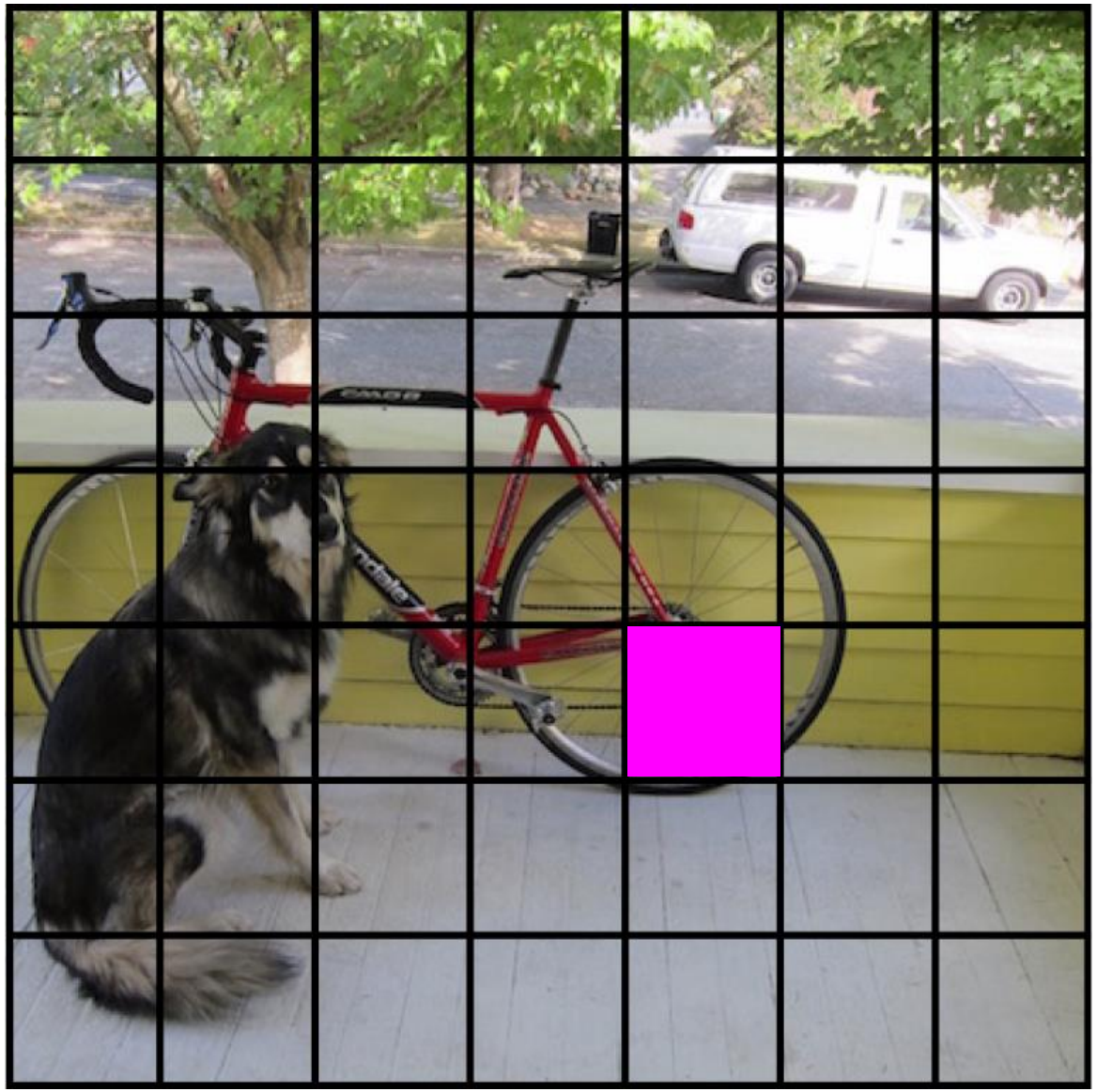
YOLO

Decrease the confidence of other boxes



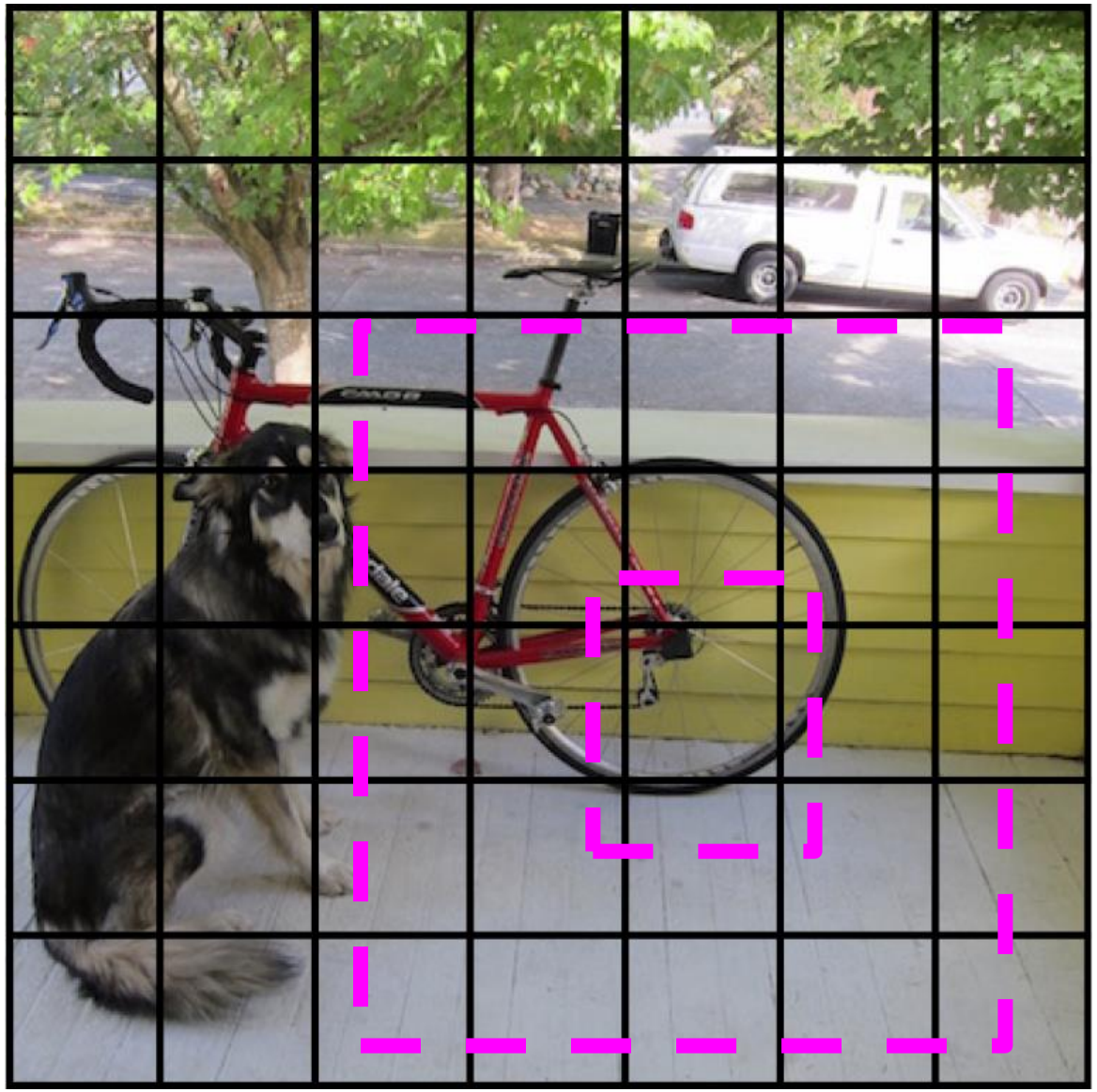
YOLO

Some cells don’t have any ground truth detections!



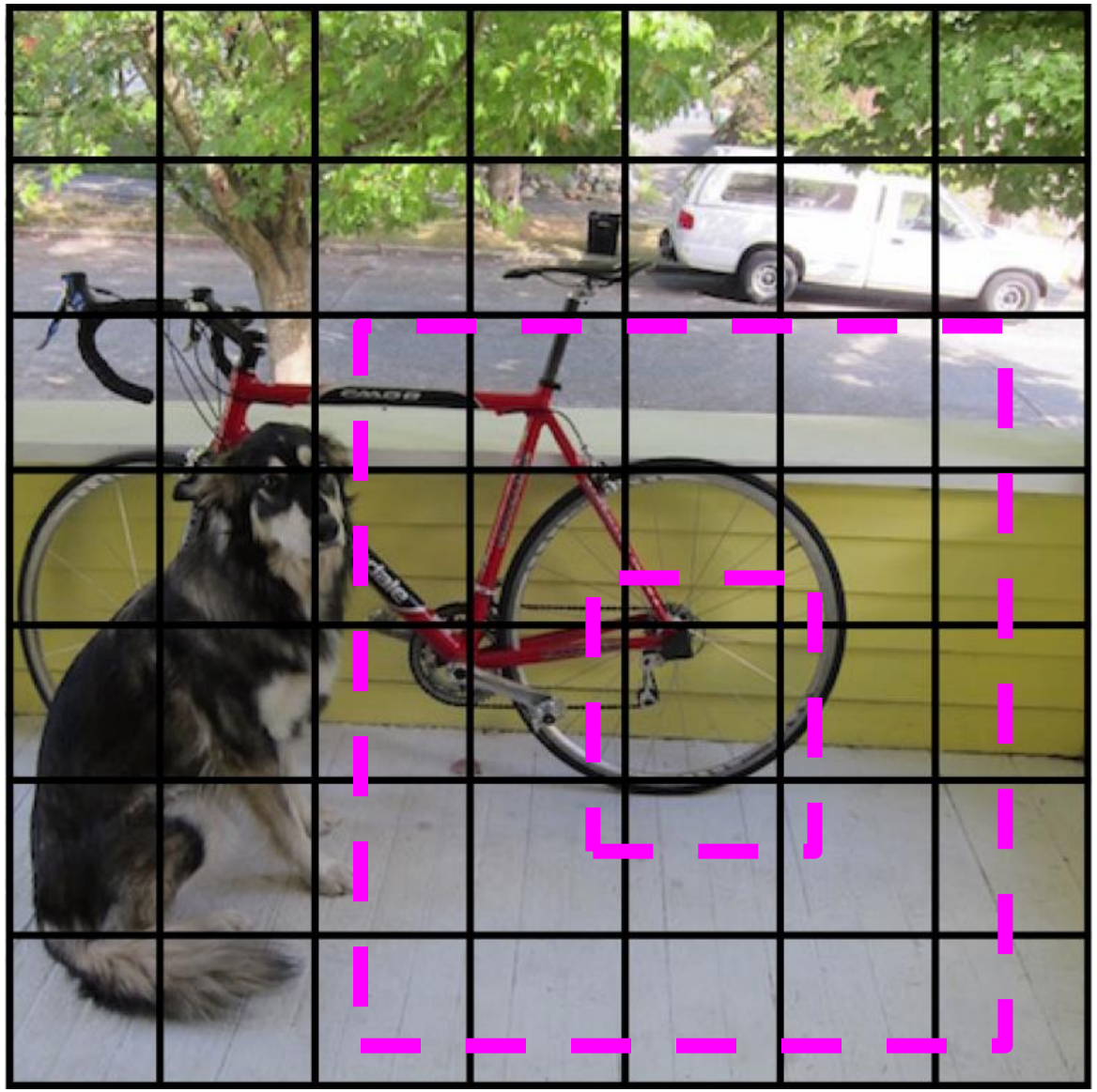
YOLO

Some cells don’t have any ground truth detections!



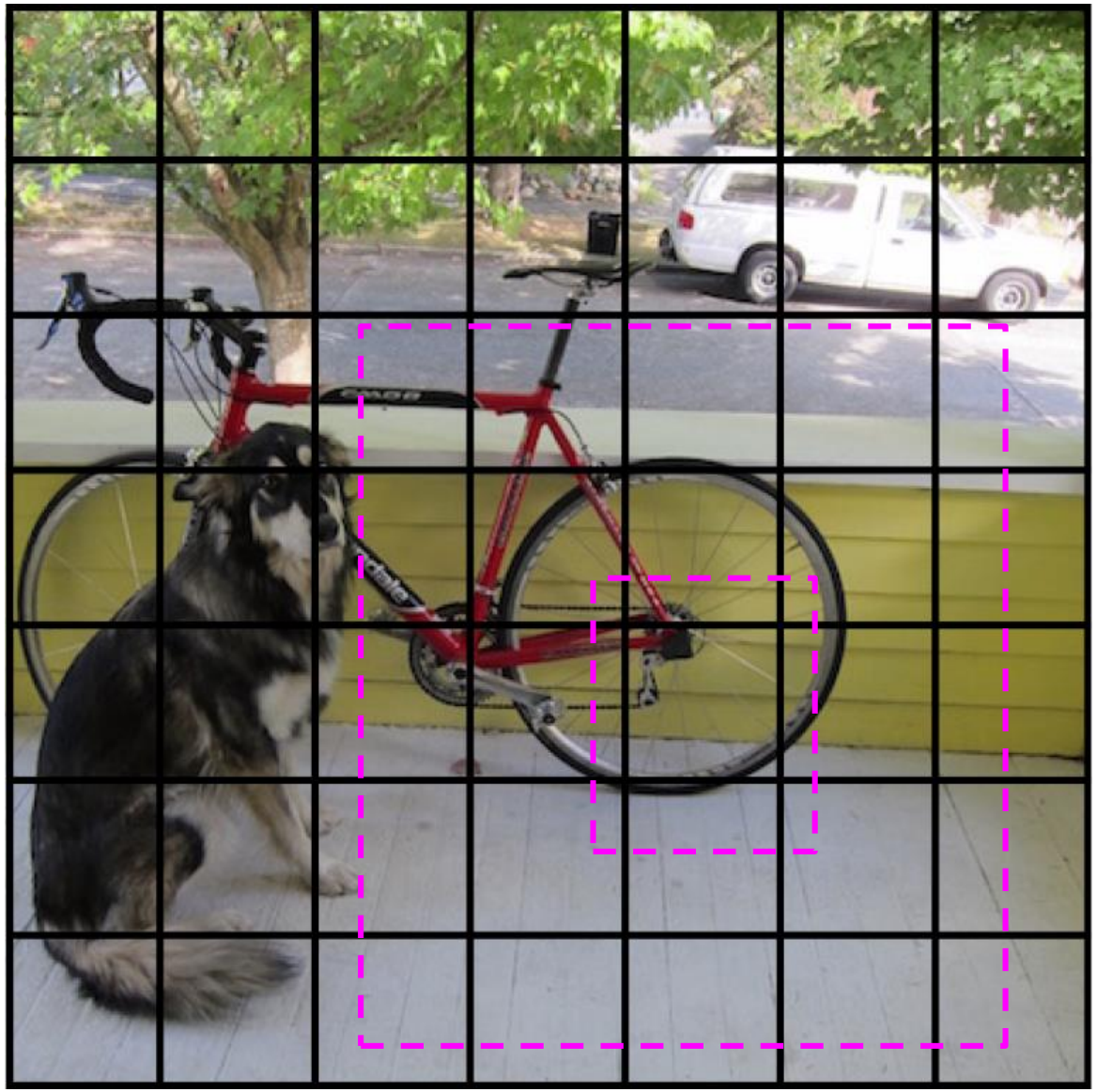
YOLO

Decrease the confidence of these boxes



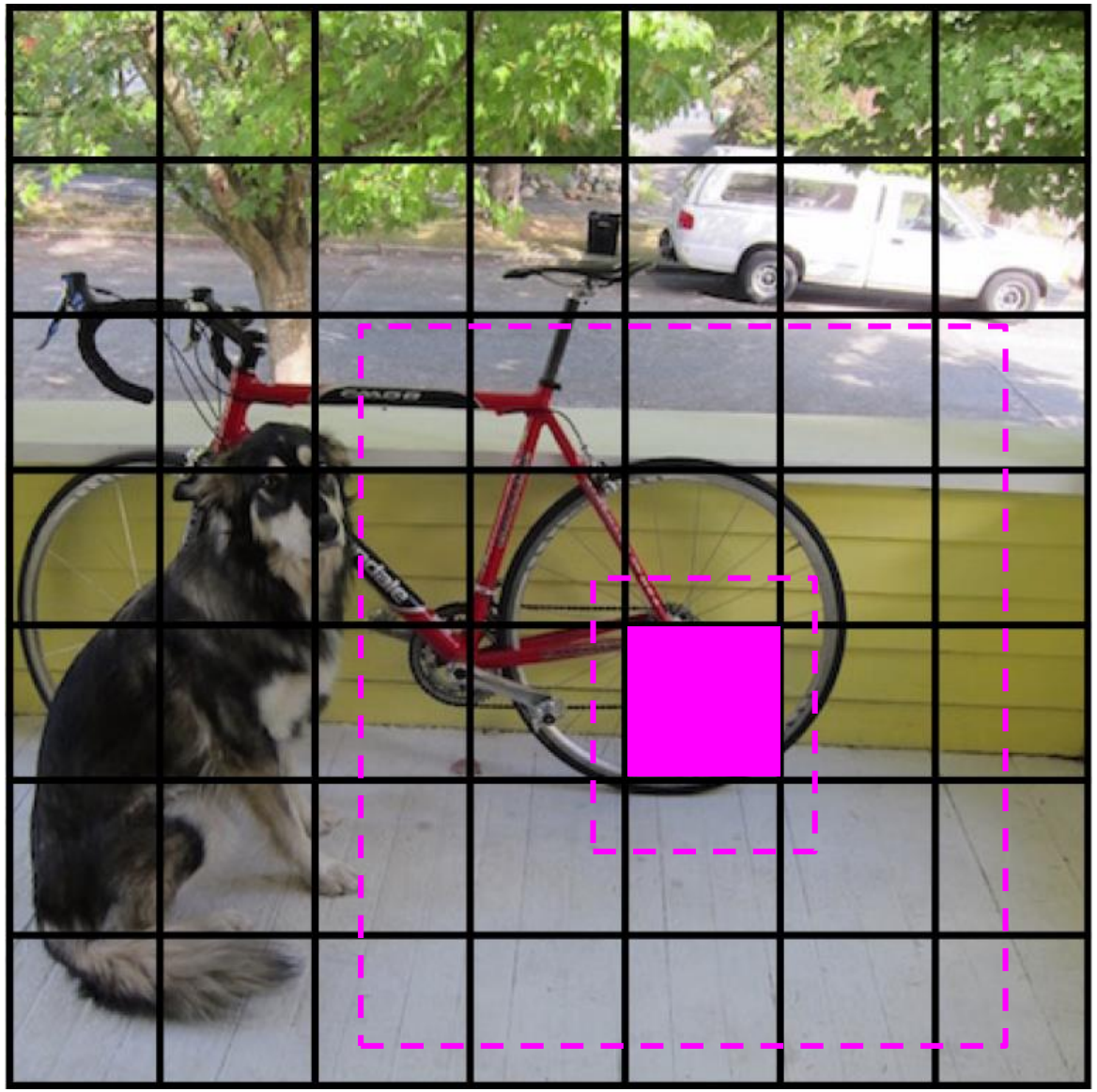
YOLO

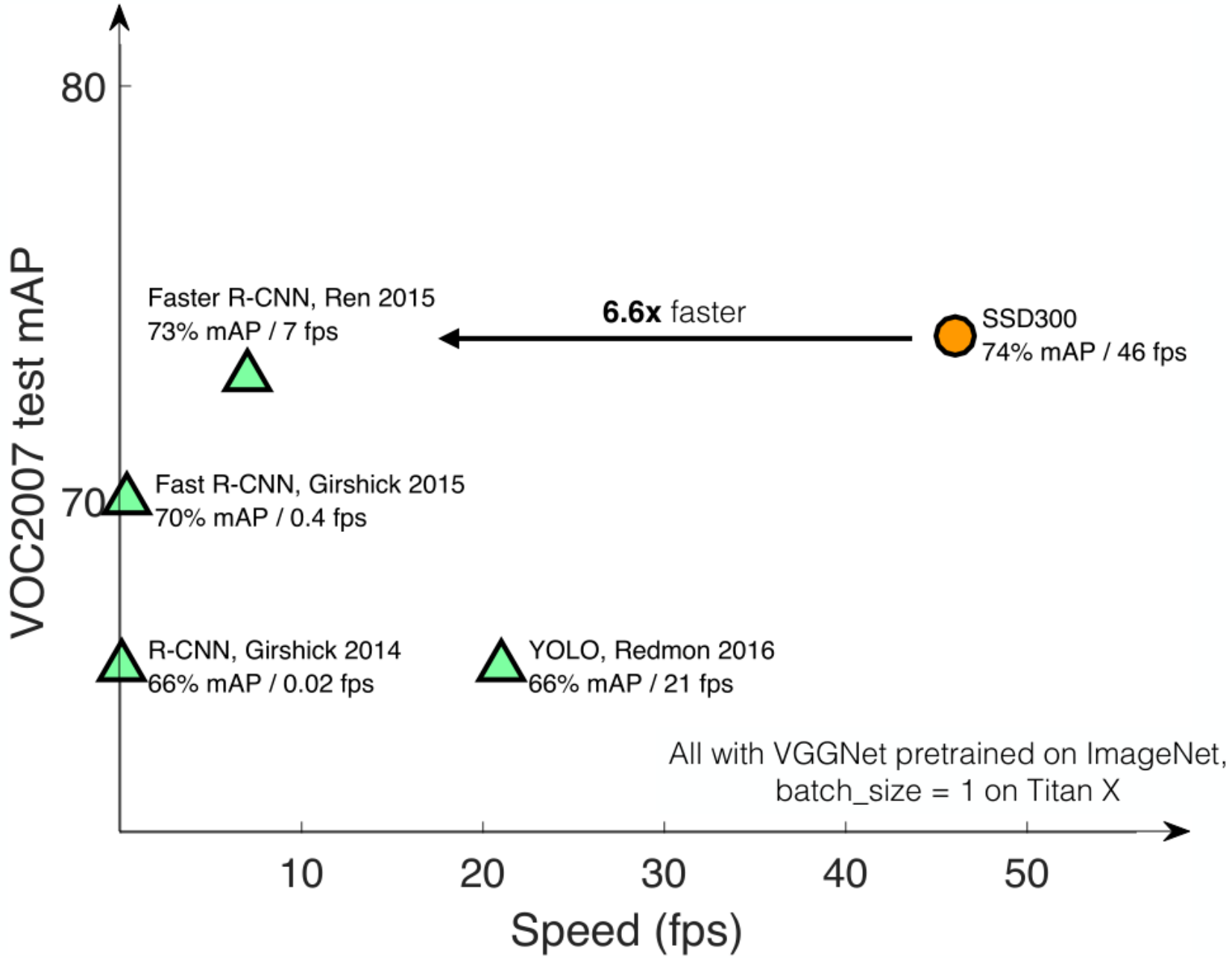
Decrease the confidence of these boxes



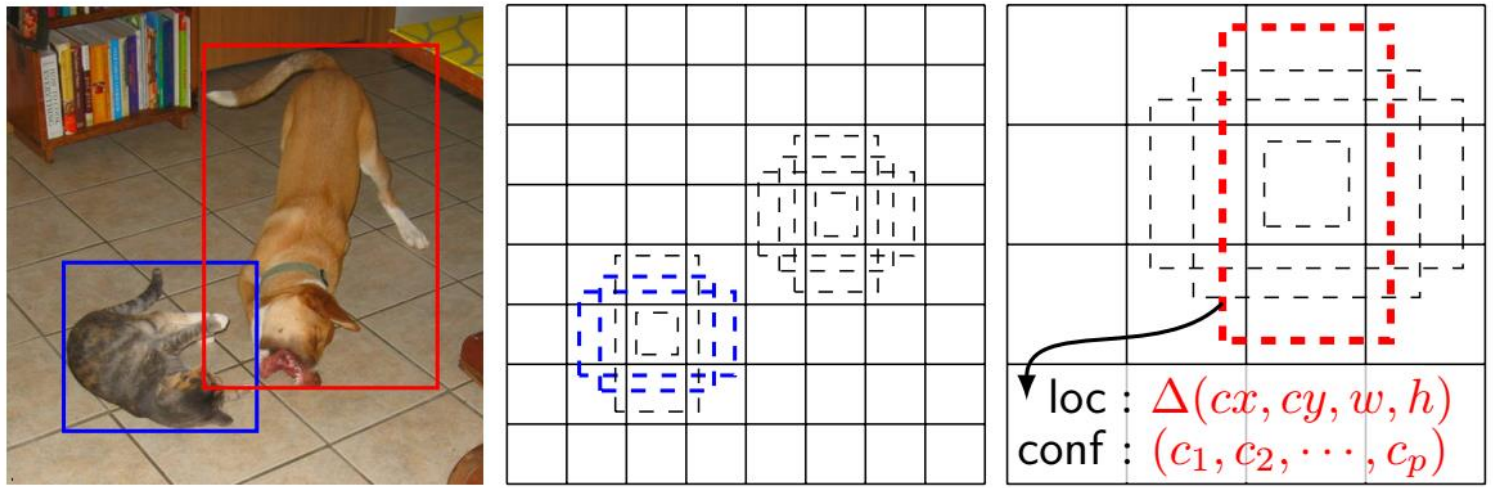
YOLO

Don’t adjust the class probabilities or coordinates




80
70
VOC2007 test mAP
Faster R-CNN, Ren 2015
73% mAP / 7 fps
6.6x faster
SSD300
74% mAP / 46 fps
Fast R-CNN, Girshick 2015
70% mAP / 0.4 fps
R-CNN, Girshick 2014
66% mAP / 0.02 fps
YOLO, Redmon 2016
66% mAP / 21 fps
All with VGGNet pretrained on ImageNet,
batch_size = 1 on Titan X
10
20
30
40
50
Speed (fps)

SSD: Single Shot MultiBox Detector



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

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, 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. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

SSD: Single Shot MultiBox Detector

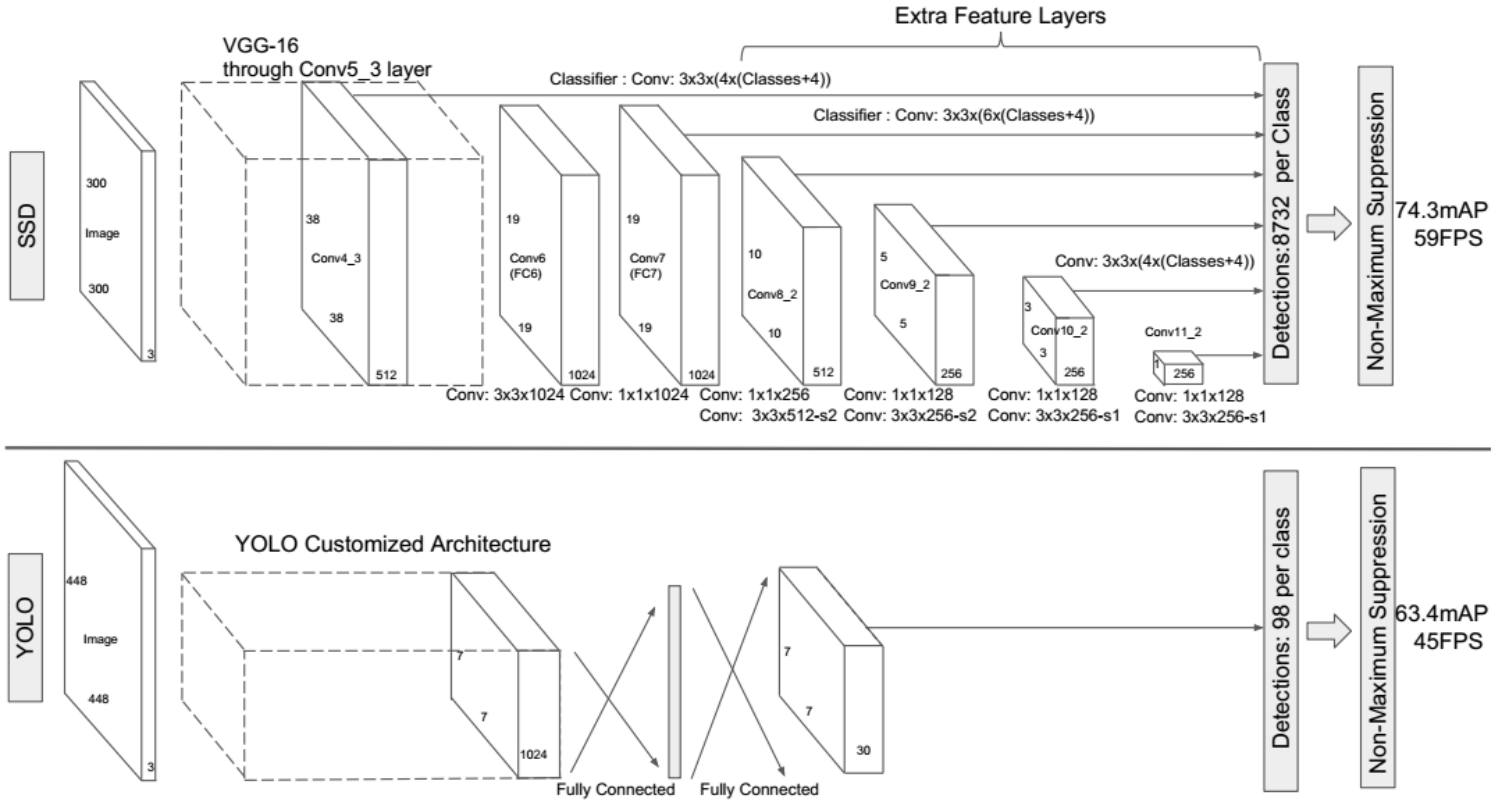
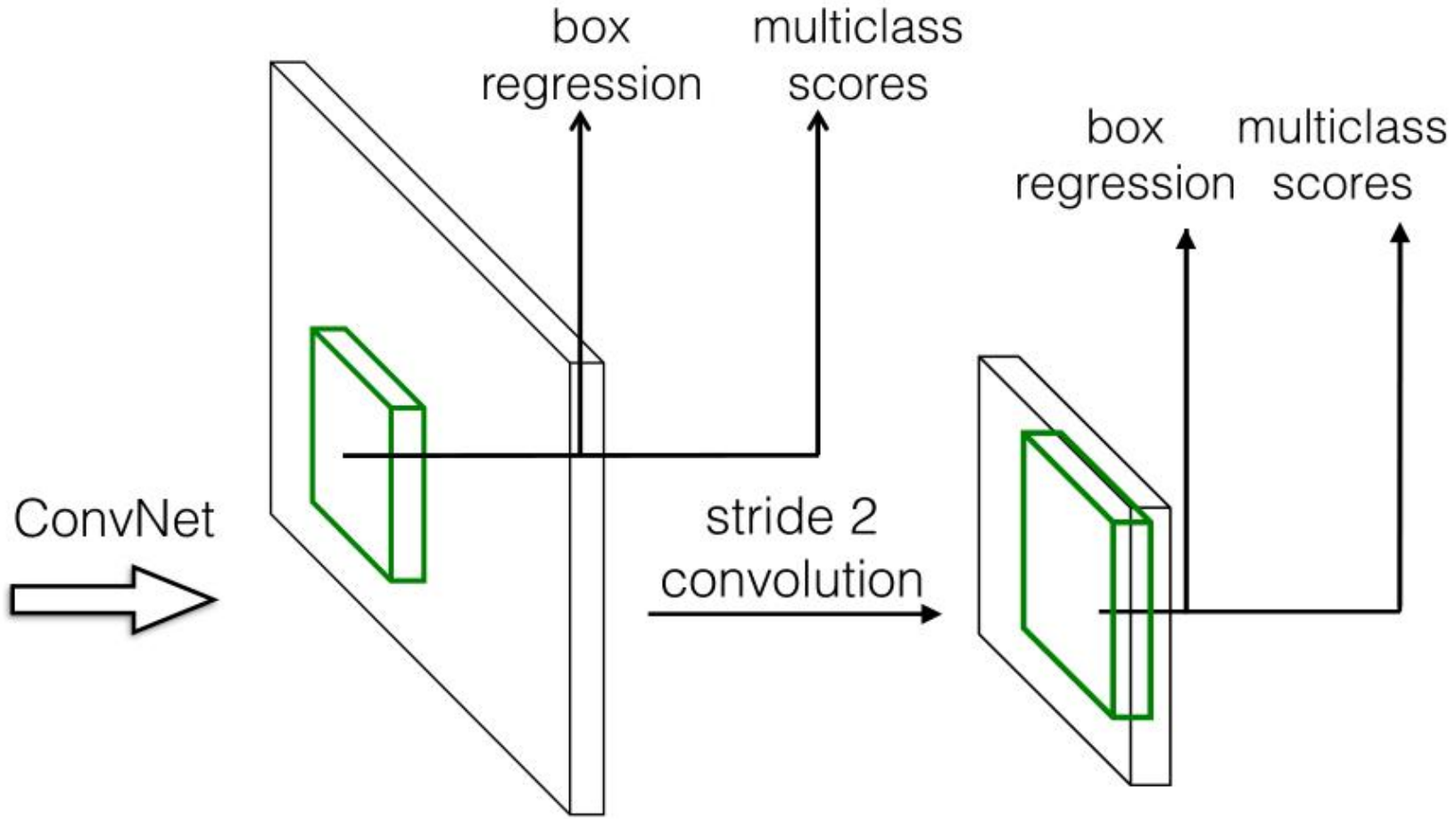


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

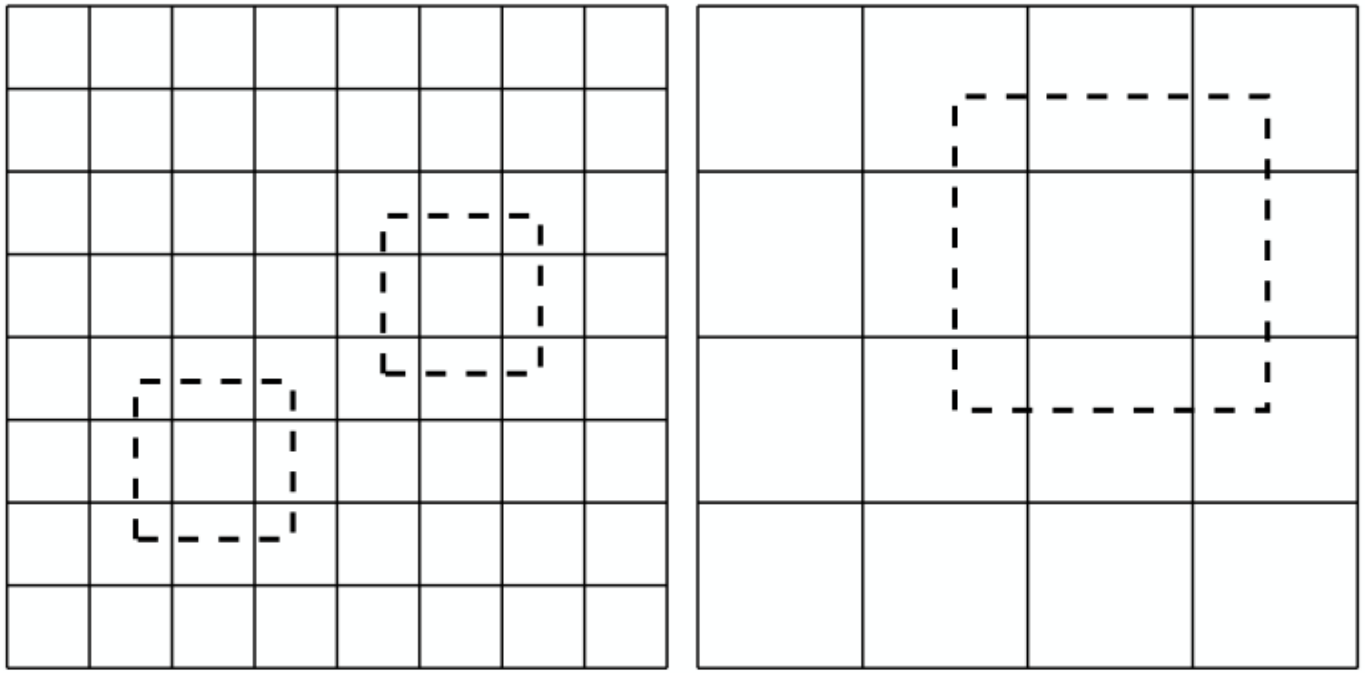
Key Ideas: Multi Scale Feature Maps



Key Ideas: Multi Scale Feature Maps

SSD

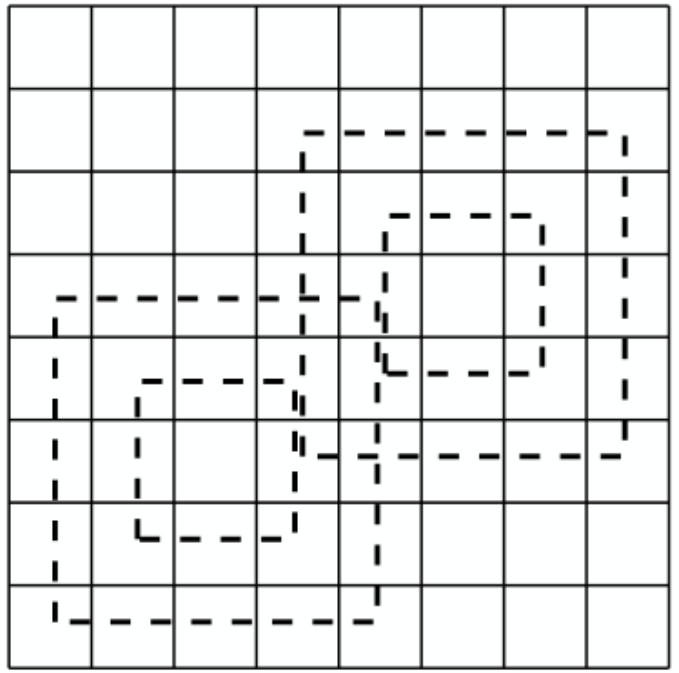
Faster R-CNN Objectness Proposal, Ren 2015



8×8 feature map

4×4 feature map

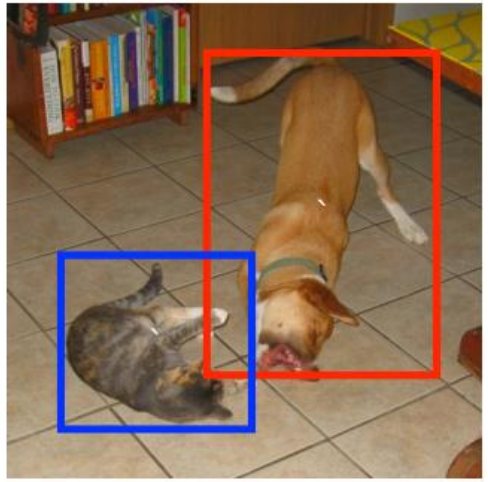
vs.

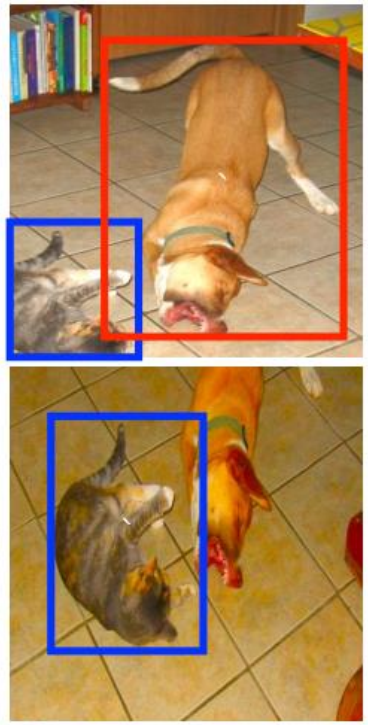


8×8 feature map

Key Ideas: Devil is in the Details!

- Data Augmentation

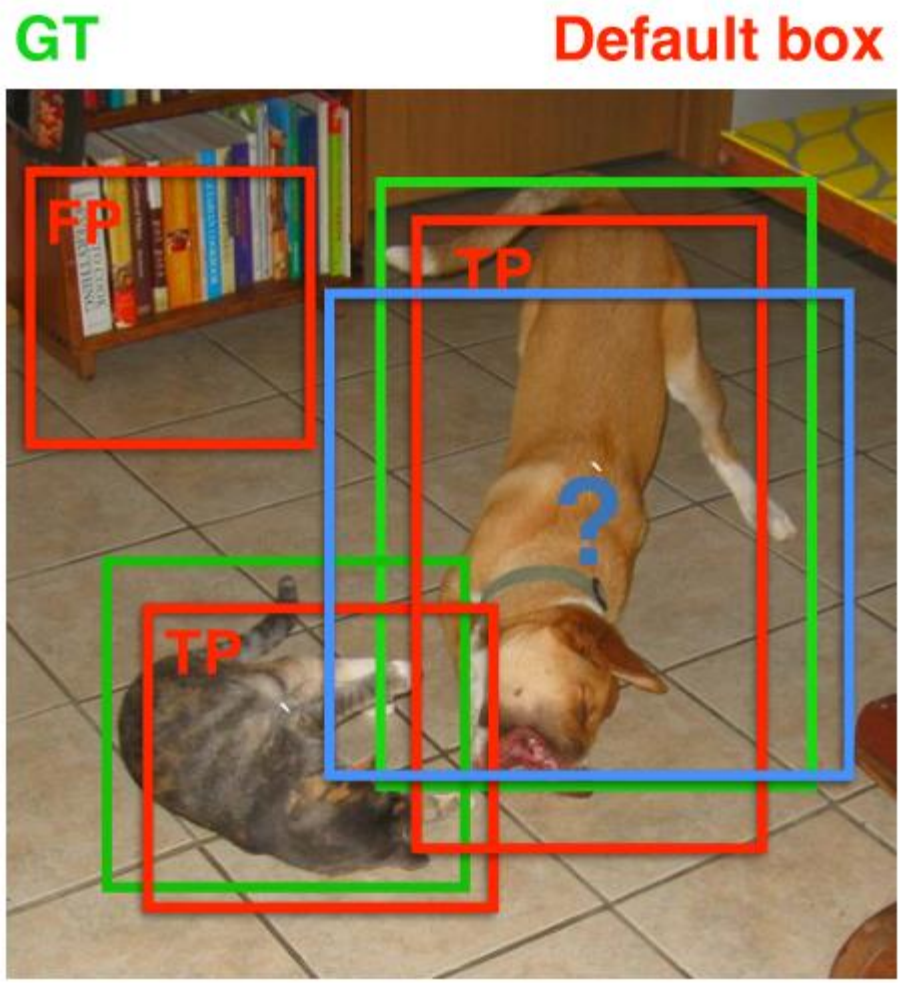




data augmentation	SSD300	
horizontal flip	✔	✔
random crop & color distortion		✔
VOC2007 `test` mAP	65.5	**74.3**

Multiple Default Boxes to Ground Truth

- Matching ground truth and default boxes
  - Match each GT box to closest default box
  - Also match each GT box to all unassigned default boxes with IoU > 0.5

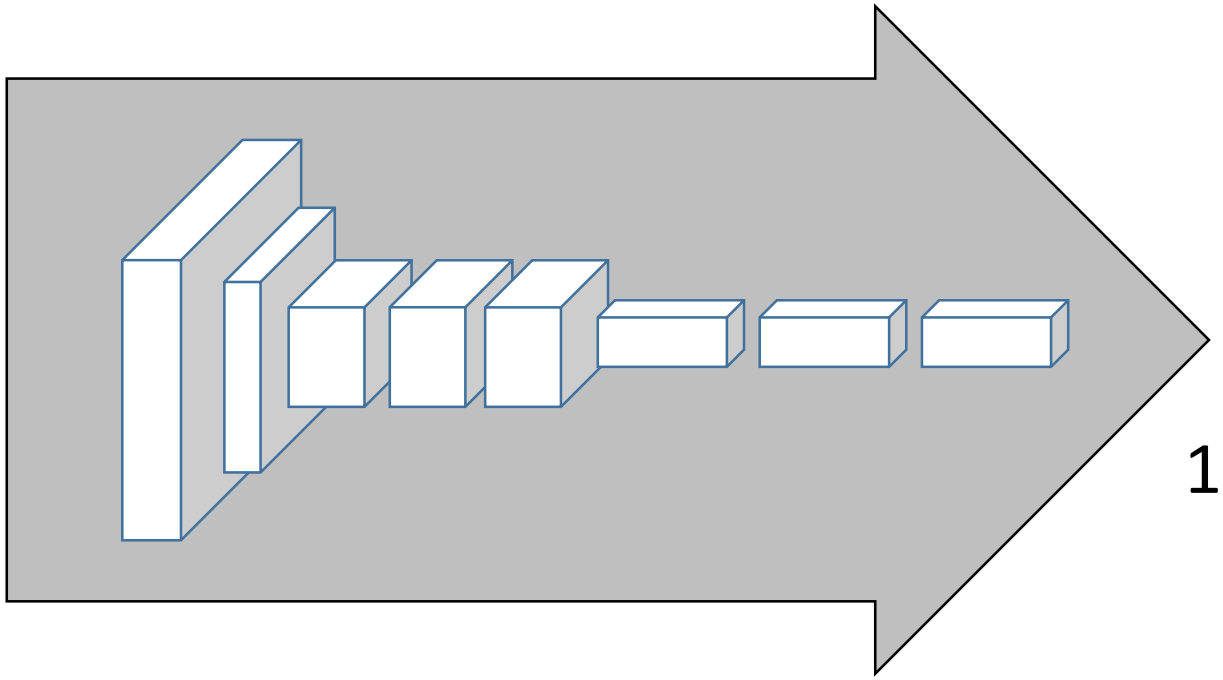


Keys to efficient CNN-based object detection

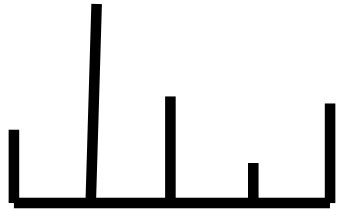
- Feature sharing
  - R-CNN => SPP-net & Fast R-CNN: sharing features among proposal regions
  - Fast R-CNN => Faster R-CNN: sharing features between proposal and detection
  - All are done by shared convolutional feature maps
- Efficient multi-scale solutions
  - Single-scale convolutional feature maps are good trade-offs
  - Multi-scale anchors are fast and flexible.
  - Multi-scale grid cells are even better.
- End to End learning helps
  - RCNN => Fast RCNN => Faster RCNN => YOLO

Semantic Segmentation

Regular CNNs



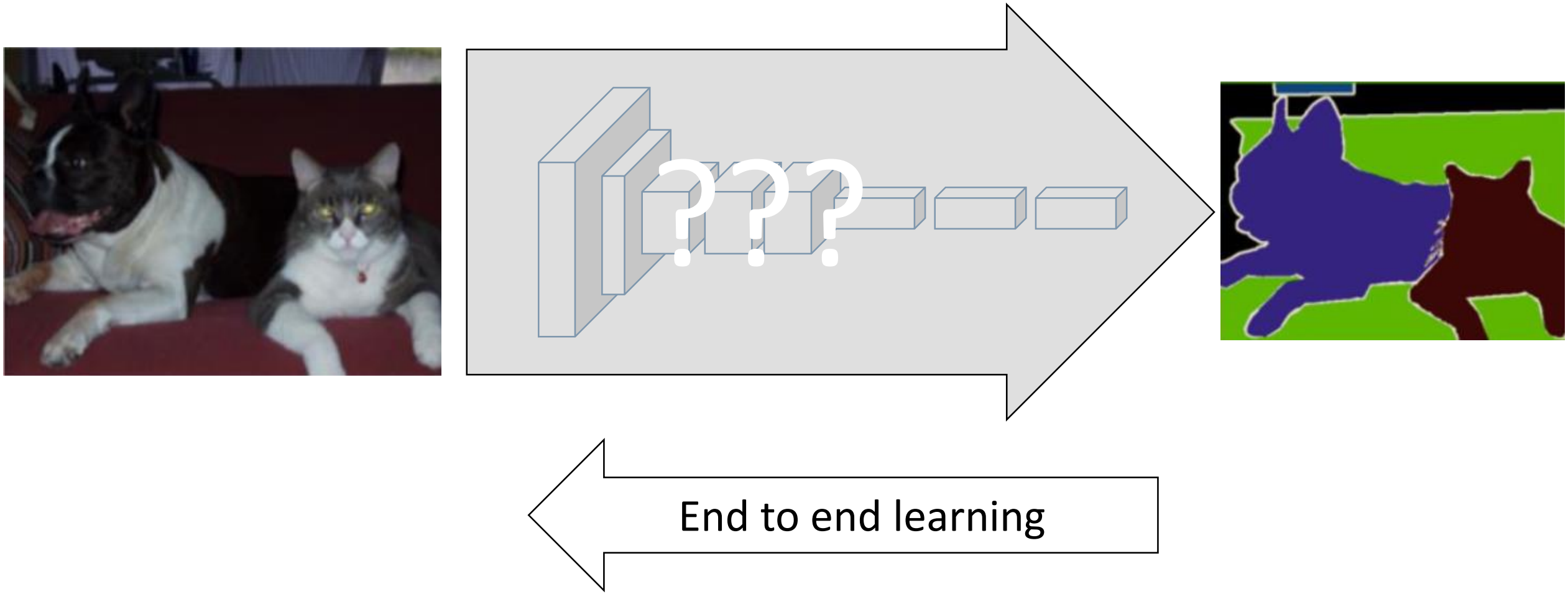
“Tabby Cat”



1000 dimensional vector

End to end learning

How to do semantic segmentation?



Fully Convolutional Networks for Semantic Segmentation

- Using Convolutional Net for Segmentation (Long, Shelhamer, Darrell, 2015)
  - Determine what objects are where
  - “what” depends on global information
  - “where” depends on local information

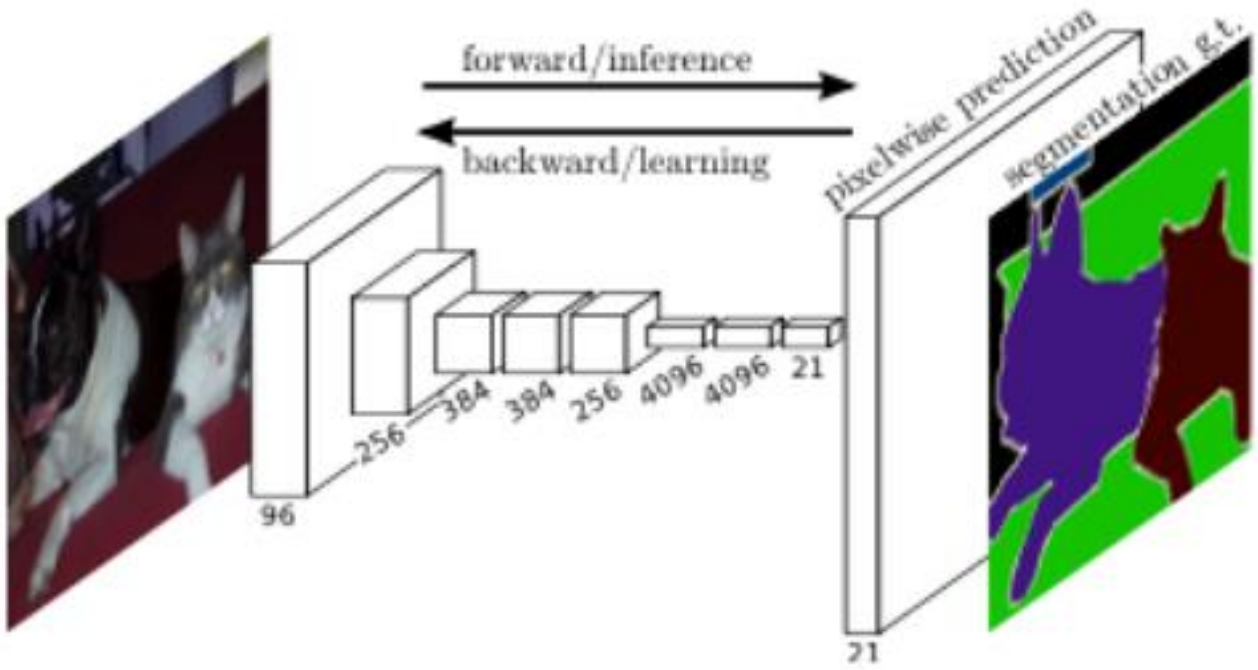


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Fully Convolutional Networks for Semantic Segmentation

- Higher layers of typical convnets are nonspatial
- If every layer remains spatial, then output can specify where as well as what (albeit coarsely)

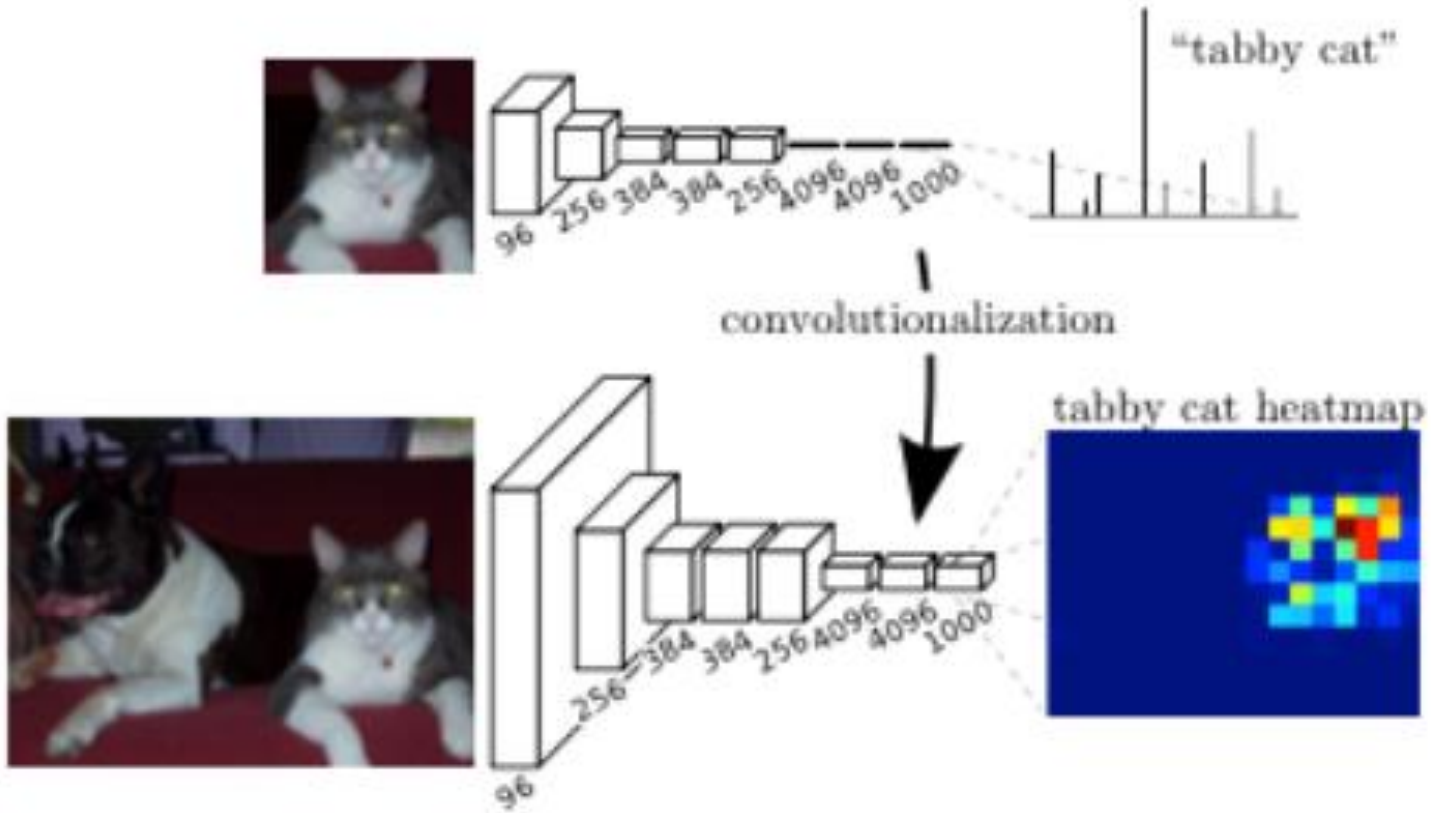
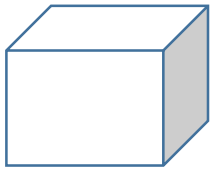
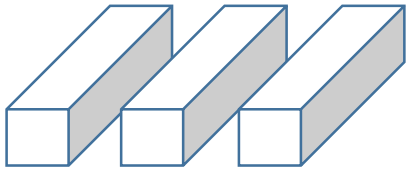
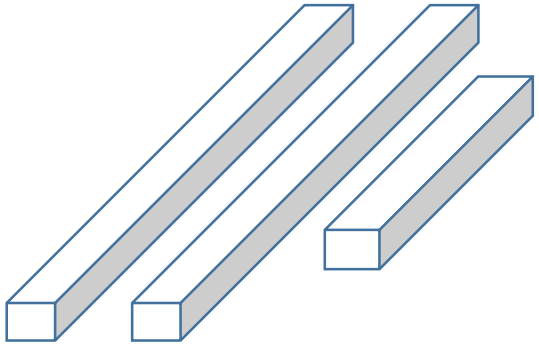


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Making Fully Convolutional + Upsampling Network







HXW

$\frac{H}{4} X \frac{W}{4}$

$\frac{H}{8} X \frac{W}{8}$

$\frac{H}{16} X \frac{W}{16}$

$\frac{H}{32} X \frac{W}{32}$

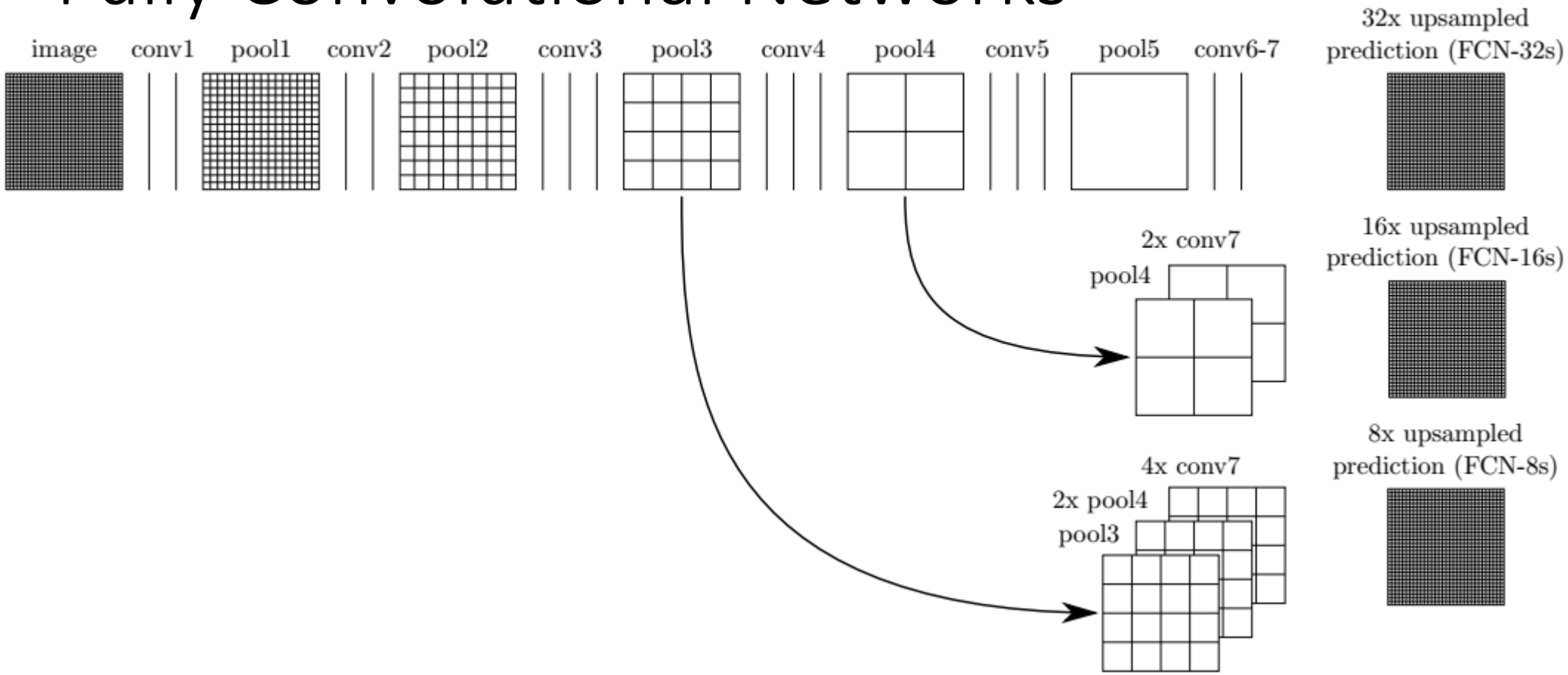
HXW

Pixelwise Output + Loss

Conv, Pool,

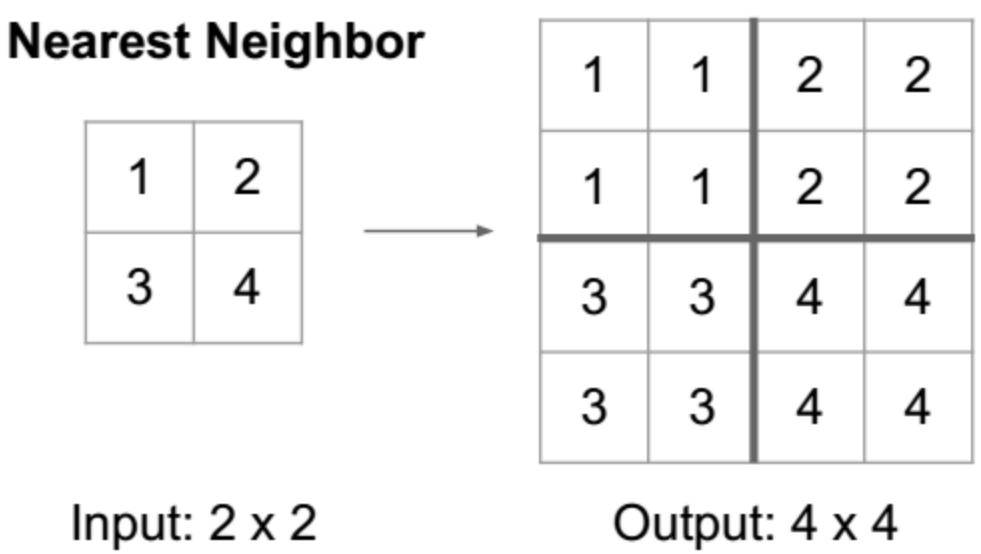
Upsampling

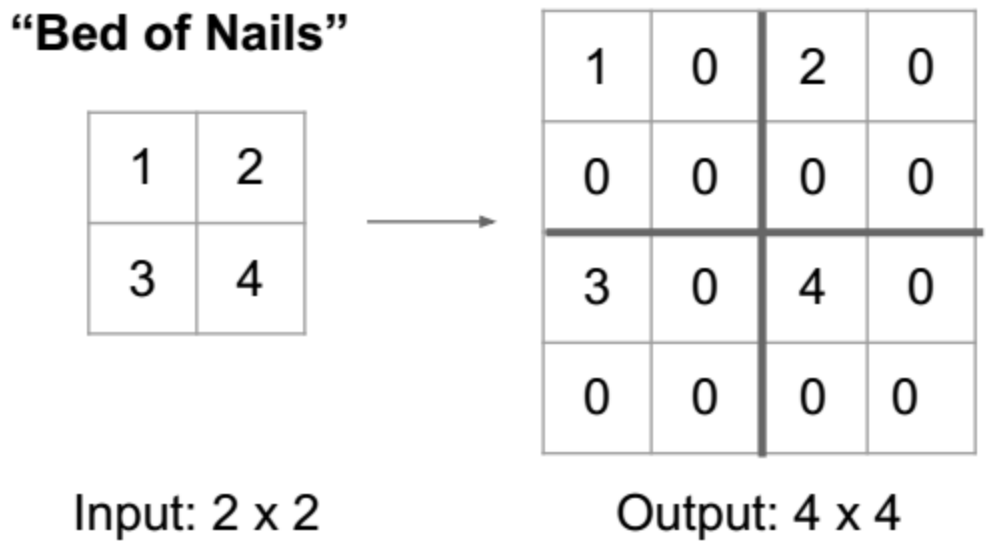
Fully Convolutional Networks



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 singlestream 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.

Upsampling/Unpooling





Upsampling: Max Unpooling

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

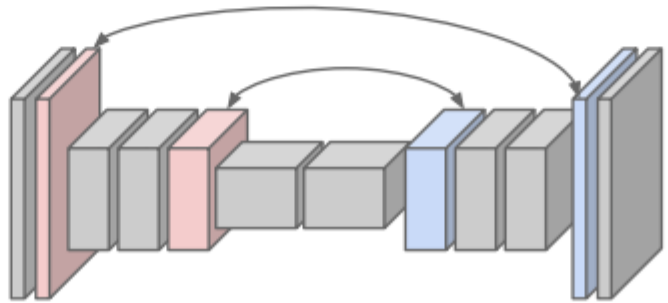
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

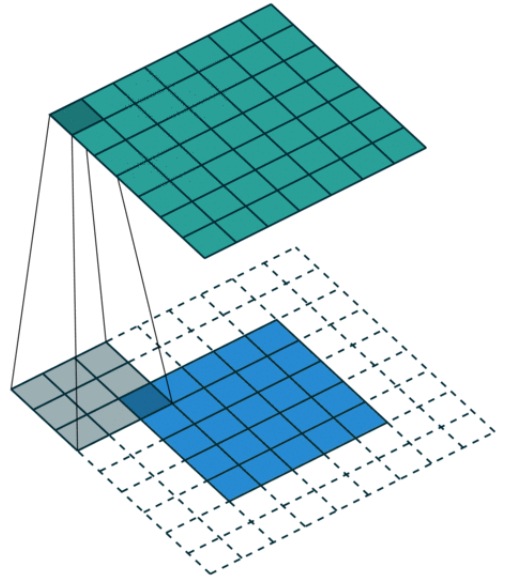
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



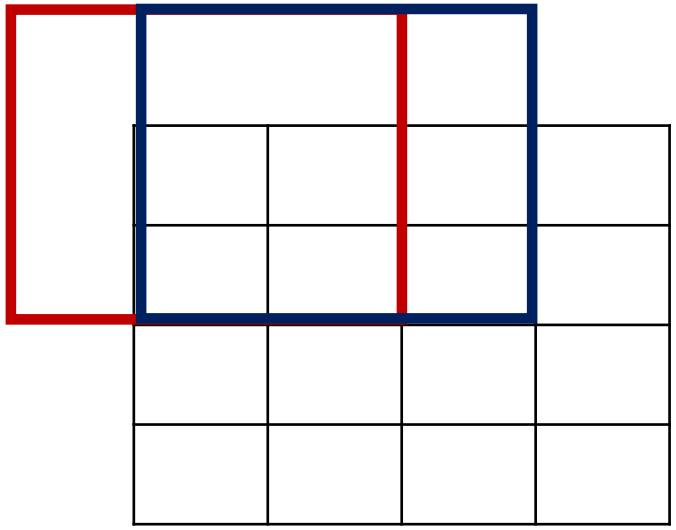
Key Idea: Learnable Upsampling

- Simple bilinear interpolation computes each output y_{ij} from the nearest four inputs by a linear map that depends only on the relative positions of the input and output cells.
- In normal convolution you take each pixel of your input image (or feature map) and calculate the dot product with your let's say 3x3 kernel. The number you get goes into the output pixel, and then you shift to the next pixel

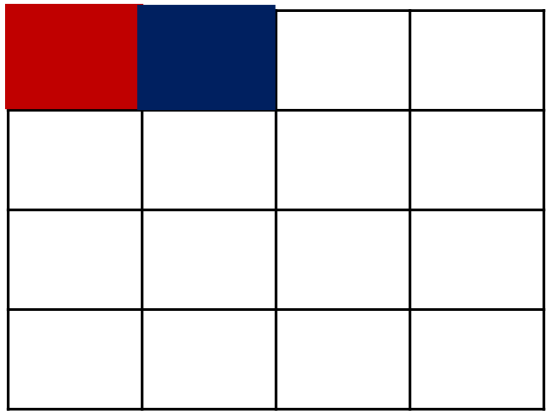


Deconvolution or Transposed Convolution

Typical 3X3 convolution: Stride 1 Pad 1



Dot product between filter and input

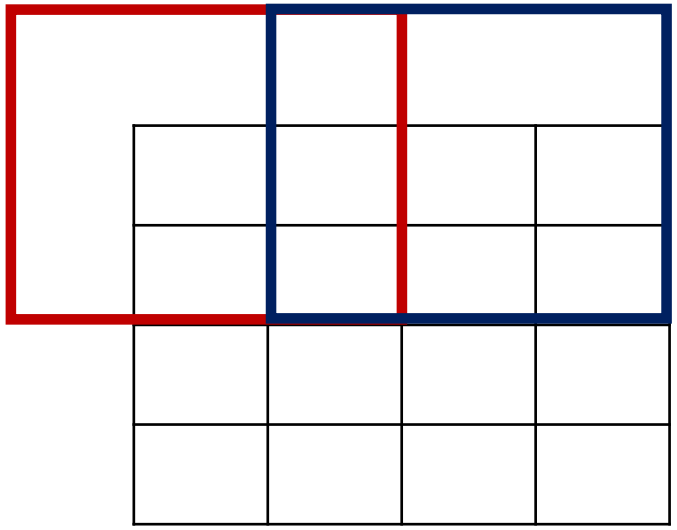


Input: 4X4

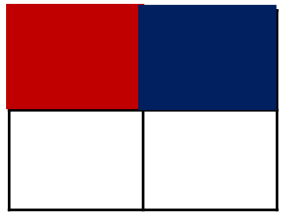
Output: 4X4

Deconvolution or Transposed Convolution

Typical 3X3 convolution: **Stride 2** Pad 1



Dot product between filter and input

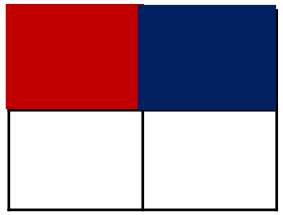


Input: 4X4

Output: 2X2

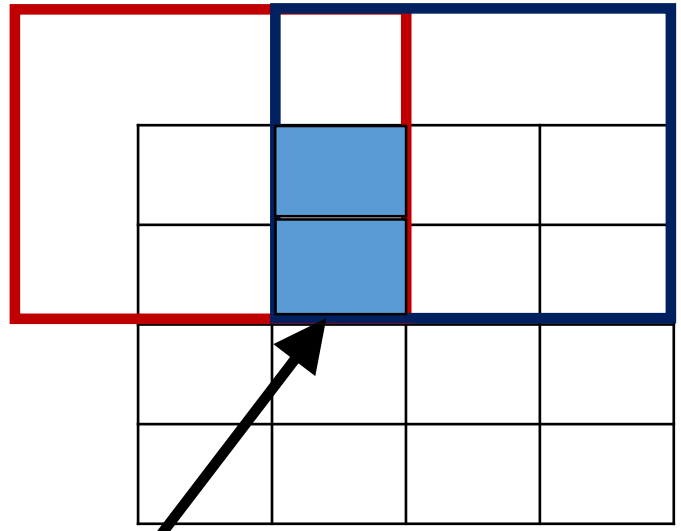
Deconvolution or Transposed Convolution

3X3 **deconvolution**: Stride 2 Pad 1



Input: 2X2

Input gives
weight for filter

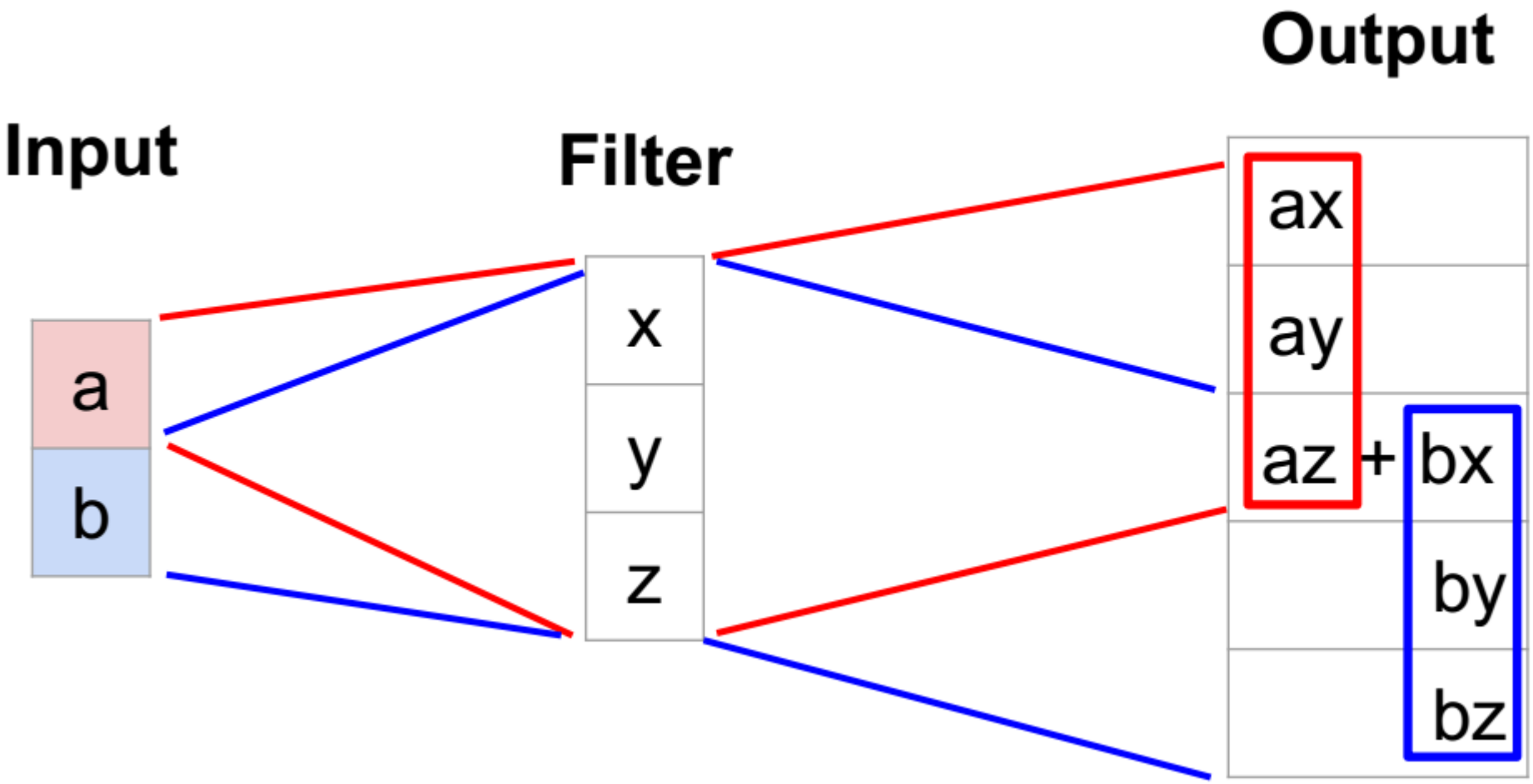


Output: 4X4

Sum where
output overlaps

Same as backward pass for normal convolution!

Deconvolution: 1D Example

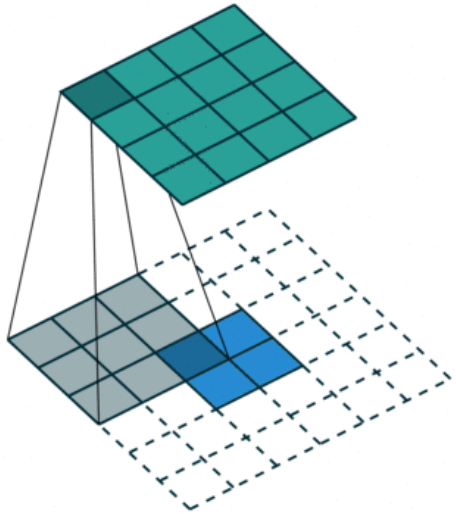


Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

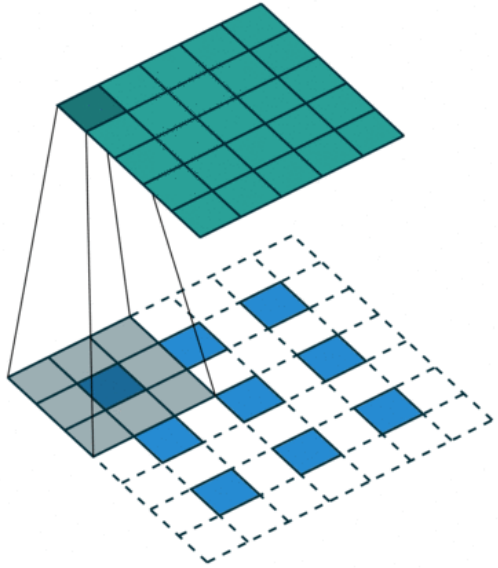
Need to crop one pixel from output to make output exactly 2x input

Transposed Convolution: Fractional Stride

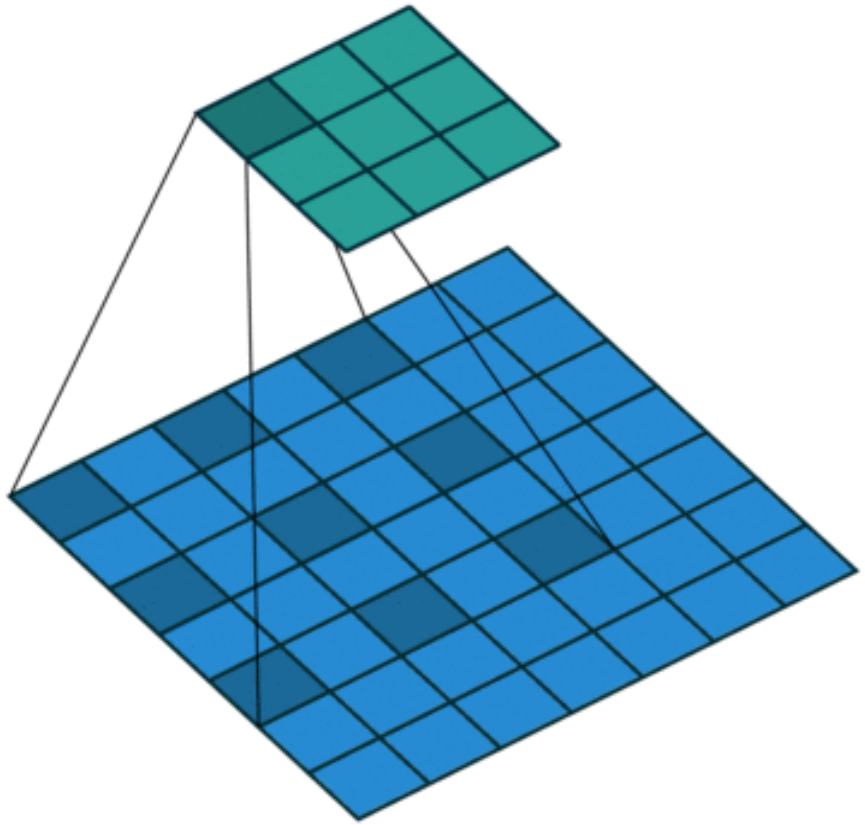
- Normal Stride 1: You take each pixel of your input image, you multiply each pixel of your say 3x3 kernel with the input pixel to get a 3x3 weighted kernel, then you insert this in your output image. Where the outputs overlap you sum them.



- You can also use a stride larger than one to increase the upsampling effect (or some would call this a convolution with a fractional stride 1/f)



Dilated Convolution



Deconvolution or Transposed Convolution

- “Deconvolution” is a bad name, already defined as “inverse of convolutions

- Better names:
  - Convolution Transpose
  - Backward Strided Convolution
  - ½ Strided Convolution
  - UpConvolution

Deconvolution or Transposed Convolution

- It is more proper to say “convolutional transpose operation” rather than “deconvolutional” operation. Hence, we will be using “convolutional transpose” from now
  - Im et al., Generating images with recurrent adversarial networks. arXiv 2016.

- A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)
  - Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

Key Idea: Skip Connection

- The 32 pixel stride at the final prediction layer limits the scale of detail in the upsampled output

- This is resolved by adding skips that combine the final prediction layer with lower layers with finer strides. This turns a line topology into a DAG, with edges that skip ahead from lower layers to higher ones

- Combining fine layers and coarse layers lets the model make local predictions that respect global structure

Results

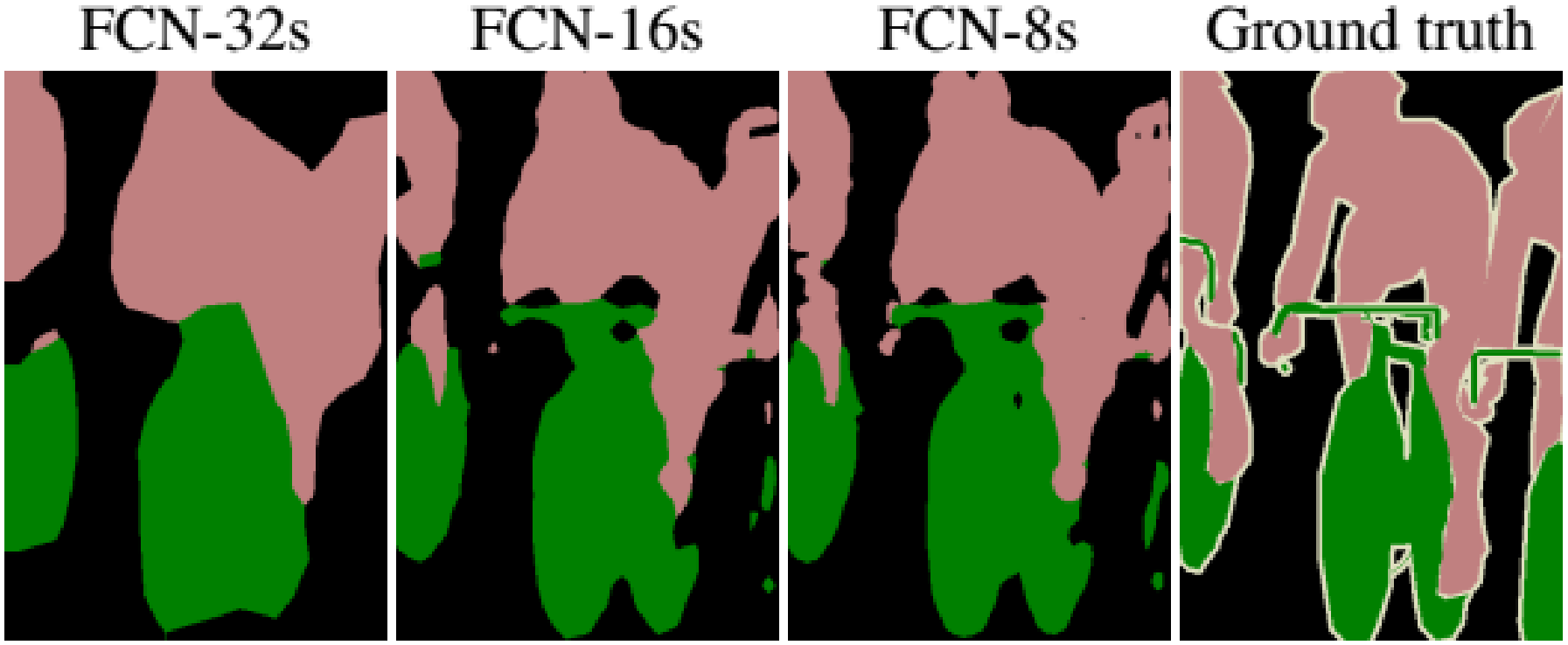
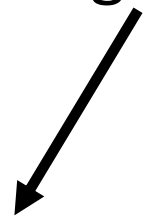


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

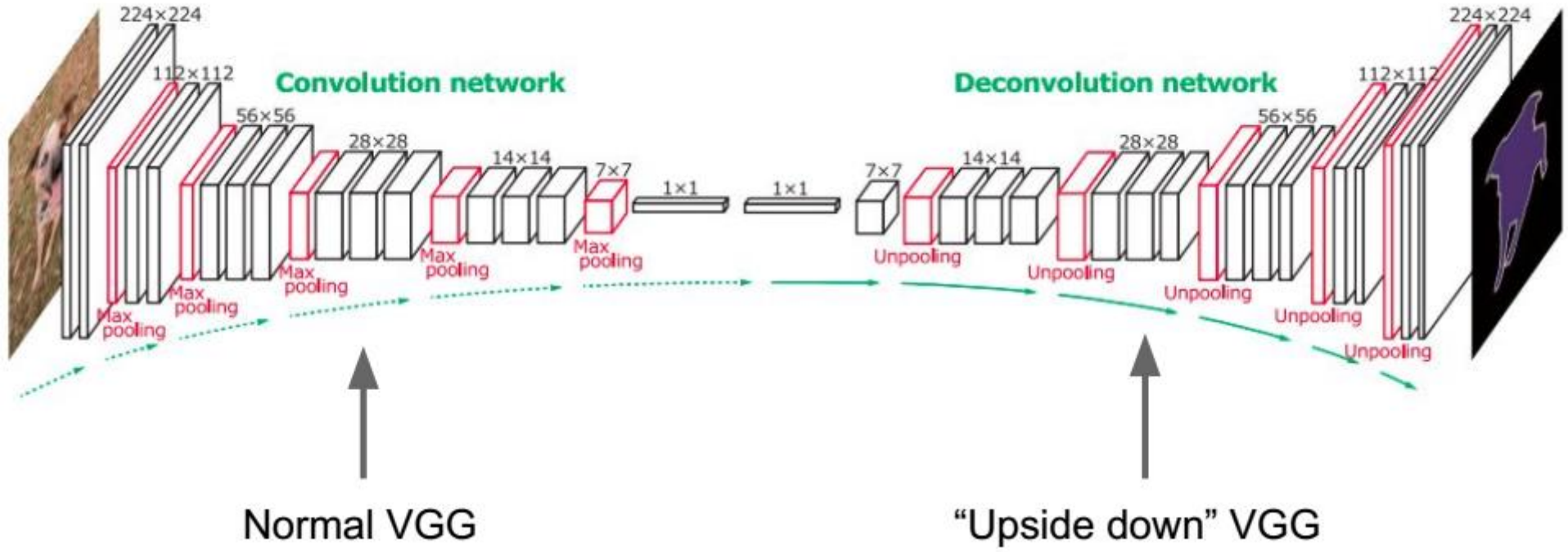
Results on Pascal VOC 2011

Frequency weighted IU



	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**

Upsampling: Other Works



Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

6 days of training on Titan X…