EEP590 Spring 2022

Deep Learning for Embedded Real Time Intelligence

Lecture 5: Neural Network Architecture for Spartial Signal Processing: Identification, Segmentation, and Detection

Prof. Richard Shi Department of Electrical and Computer Engineering
(cjshi@uw.edu)

with the material adapted from Joseph Redmon and Ali Farhadi at UW CSE

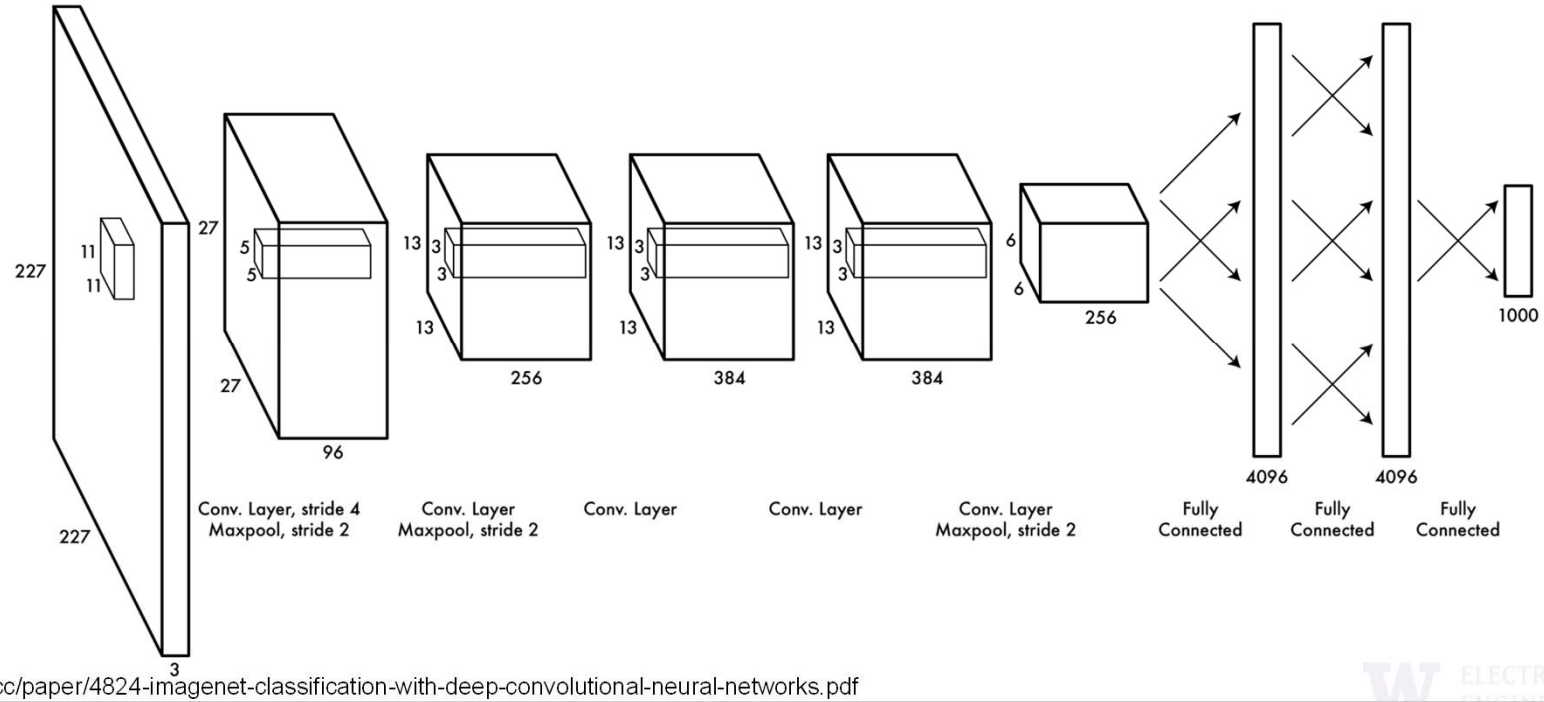
ELECTRICAL & COMPUTER ENGINEERING

AlexNet: The First Real Good Network

Entry for ImageNet challenge

Much higher accuracy than “traditional” methods (SVM on SIFT)

Why did it take so long?



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

GPUs + Data ➔ Enable CNNs Effectiveness

Alex spent a long time implementing the components of a CNN on a GPU, his software Cudaconvnet was the first “modern” neural network framework

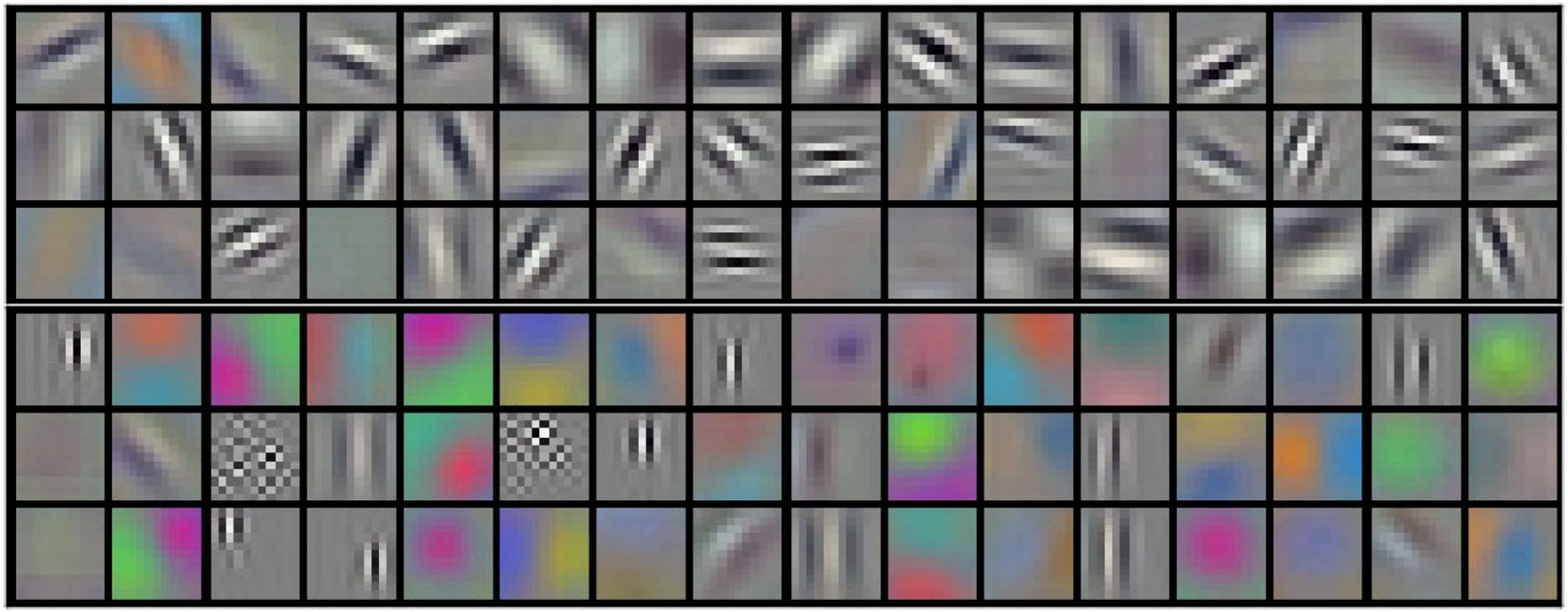
GPUs allow >100x speedup compared to CPU, training still took weeks on ImageNet

This idea, CNNs + GPUs started revolution in computer vision just 5 years ago

(check NVIDIAs stock in last 5 years)

What are These Networks Learning?

Features! Here's 1st layer of AlexNet



To Understand What Neural Networks do:

We can feed images into our network and see which ones activate certain neurons

(a) Unit sensitive to white flowers.

(c) Unit senstive to round, spiky flowers.

(d) Unit senstive to round green or yellow objects.

https://arxiv.org/pdf/1312.6199.pdf

Idea: *Make* Interesting Images

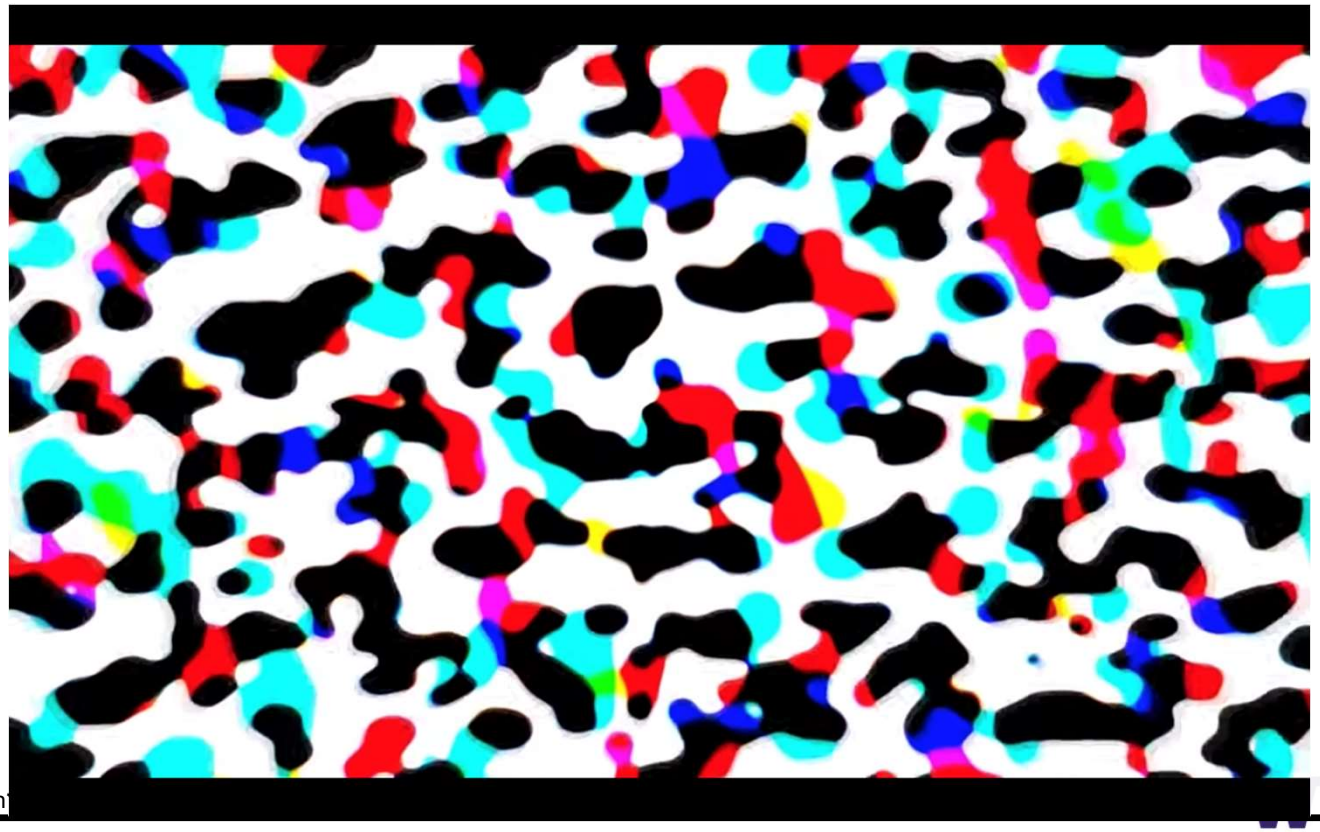
Instead of optimizing network, we can do gradient descent on the image too

Optimize the image to activate certain neurons or certain layers, then we can learn what those layers or neurons do!

You may have seen this under the name *deep dream*

Early Layers: Blobs

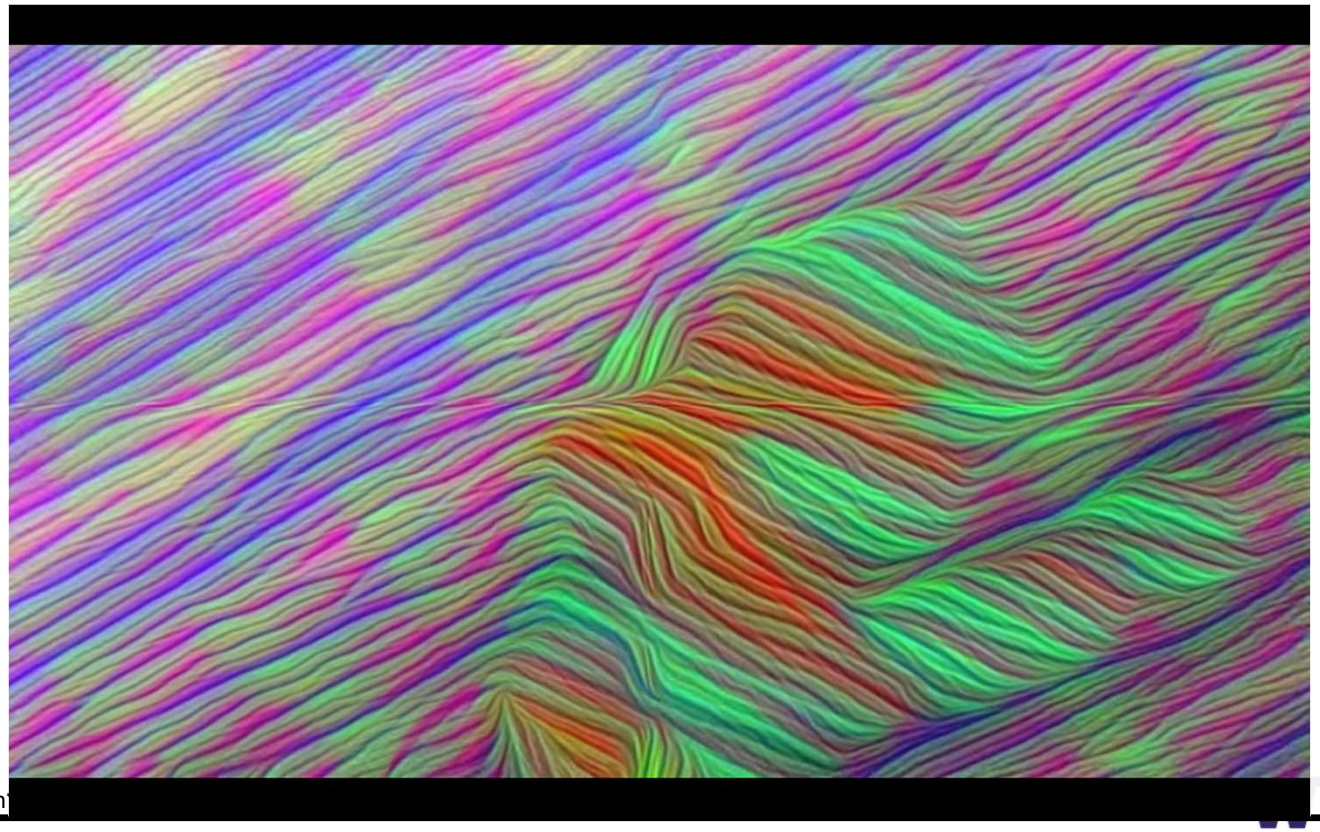
Neurons respond strongly to high contrast



https://www.youtube.com/watch

Middle Layers: Edges, Curves, Eyes

Neurons respond to simple features and shapes



https://www.youtube.com/watch

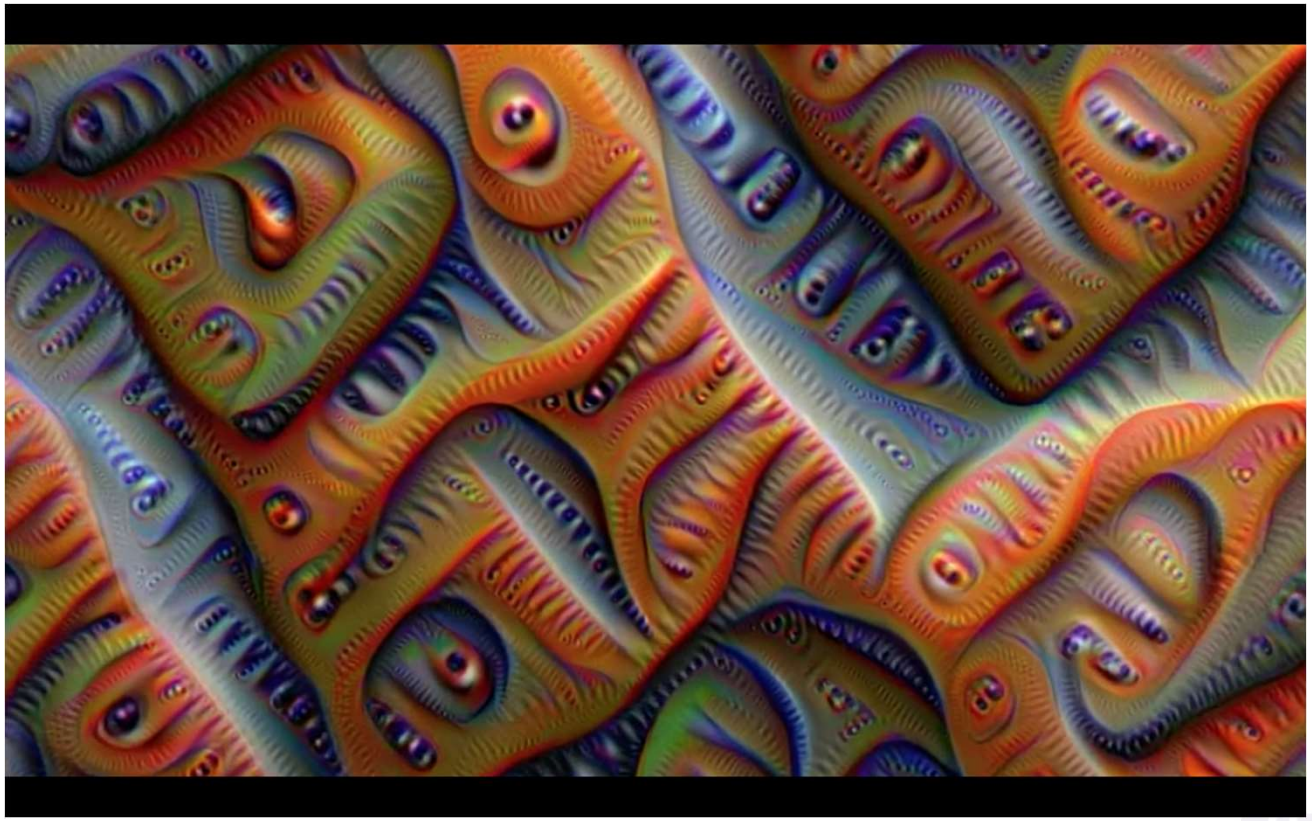
Middle Layers: Edges, Curves, Eyes

Neurons respond to simple features and shapes

https://www.youtube.com/watch?v=SCE-QeDfXtA

Middle Layers: Edges, Curves, Eyes

Neurons respond to simple features and shapes



https://www.youtube.com/watch?v=SCE-QeDfXtA

Later Layers: Eyes, Objects, Dogs

Neurons respond to arrangements of features

https://www.youtube.com/watch

Later Layers: Eyes, Objects, Dogs

Neurons respond to arrangements of features

https://www.youtube.com/watch

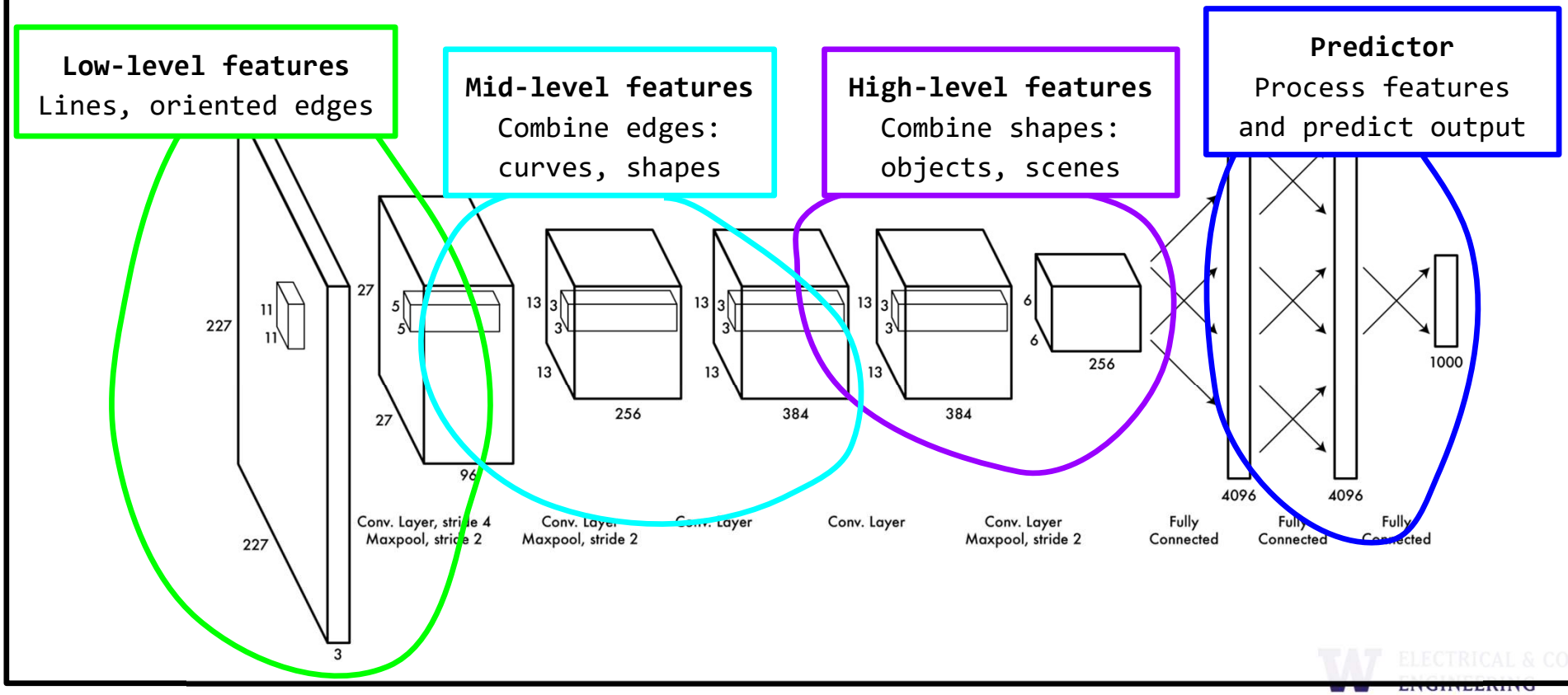
Later Layers: Eyes, Objects, Dogs

Neurons respond to arrangements of features

https://www.youtube.com/watch

Artificial Neural Networks (ANNs) Work!

At least, seem to do what we want them to



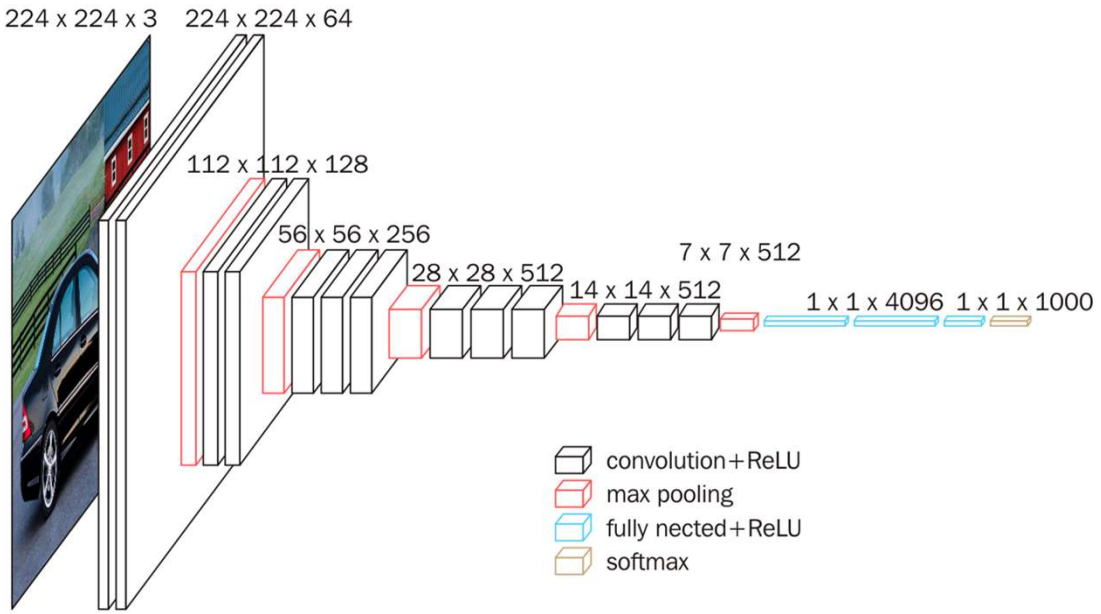
VGG: Networks Getting Deeper

From the Visual Geometry Group at Oxford

Considered “very deep” at the time, 16 - 19 layers

VGG-16 is still commonly used as a feature extractor

Although really it shouldn’t be, much better alternatives



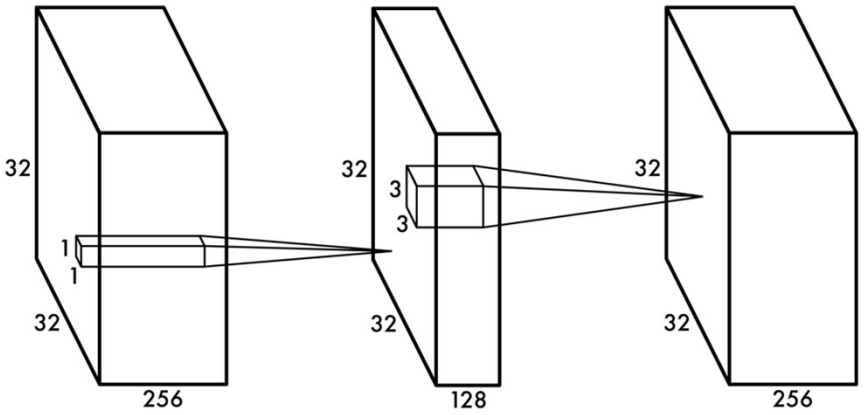
VGG Inefficient

Just stringing together a bunch of 3x3 convolutions in general, is pretty inefficient.

Network in network idea from Lin et al.:

- Use 3x3 convolutions to process spatial information
- Use 1x1 convolutions to reduce # of channels (dimensionality reduction)

E.g.



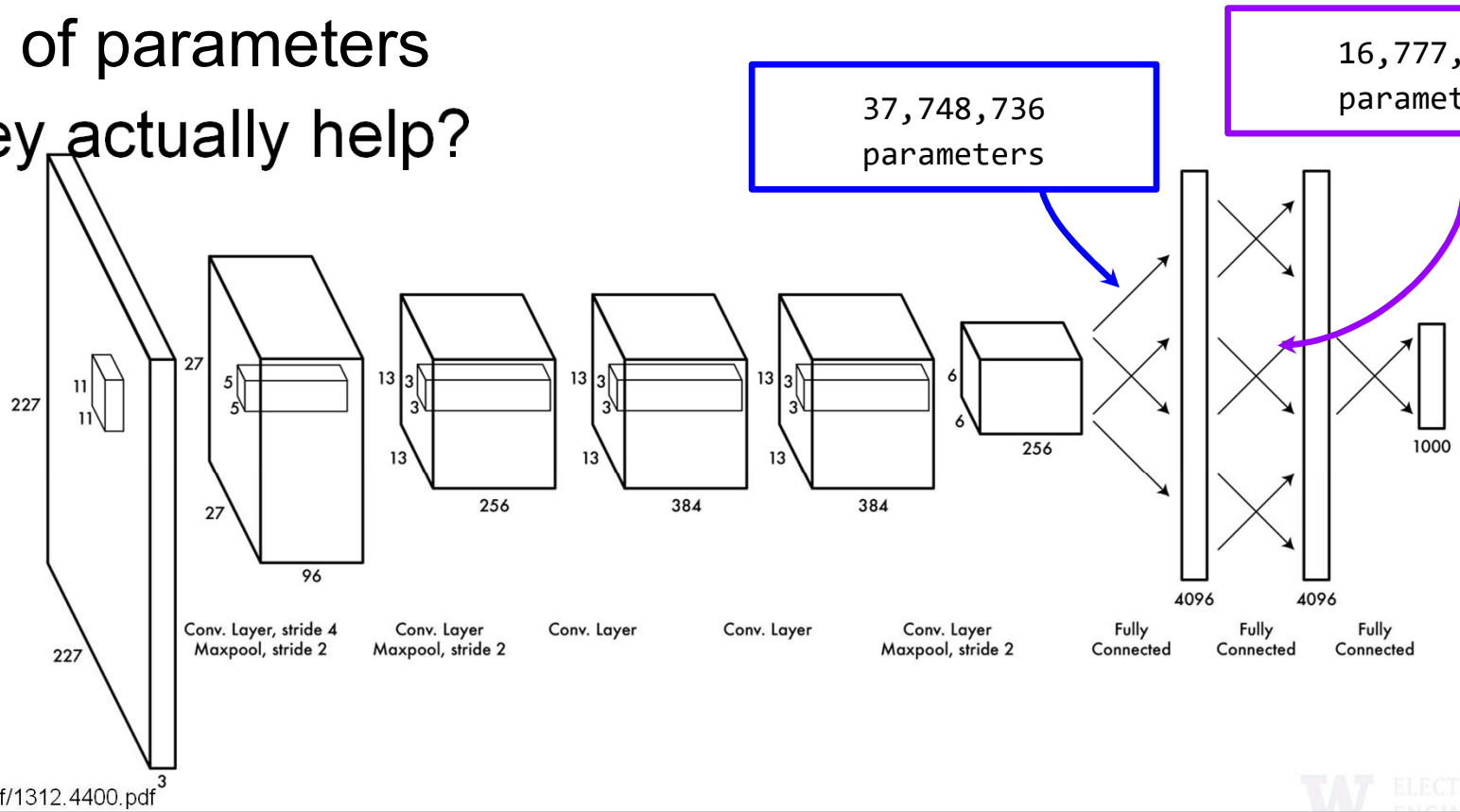
https://arxiv.org/pdf/1312.4400.pdf

VGG (and AlexNet) Inefficient

Huge fully connected layers at the end

TONS of parameters

Do they actually help?



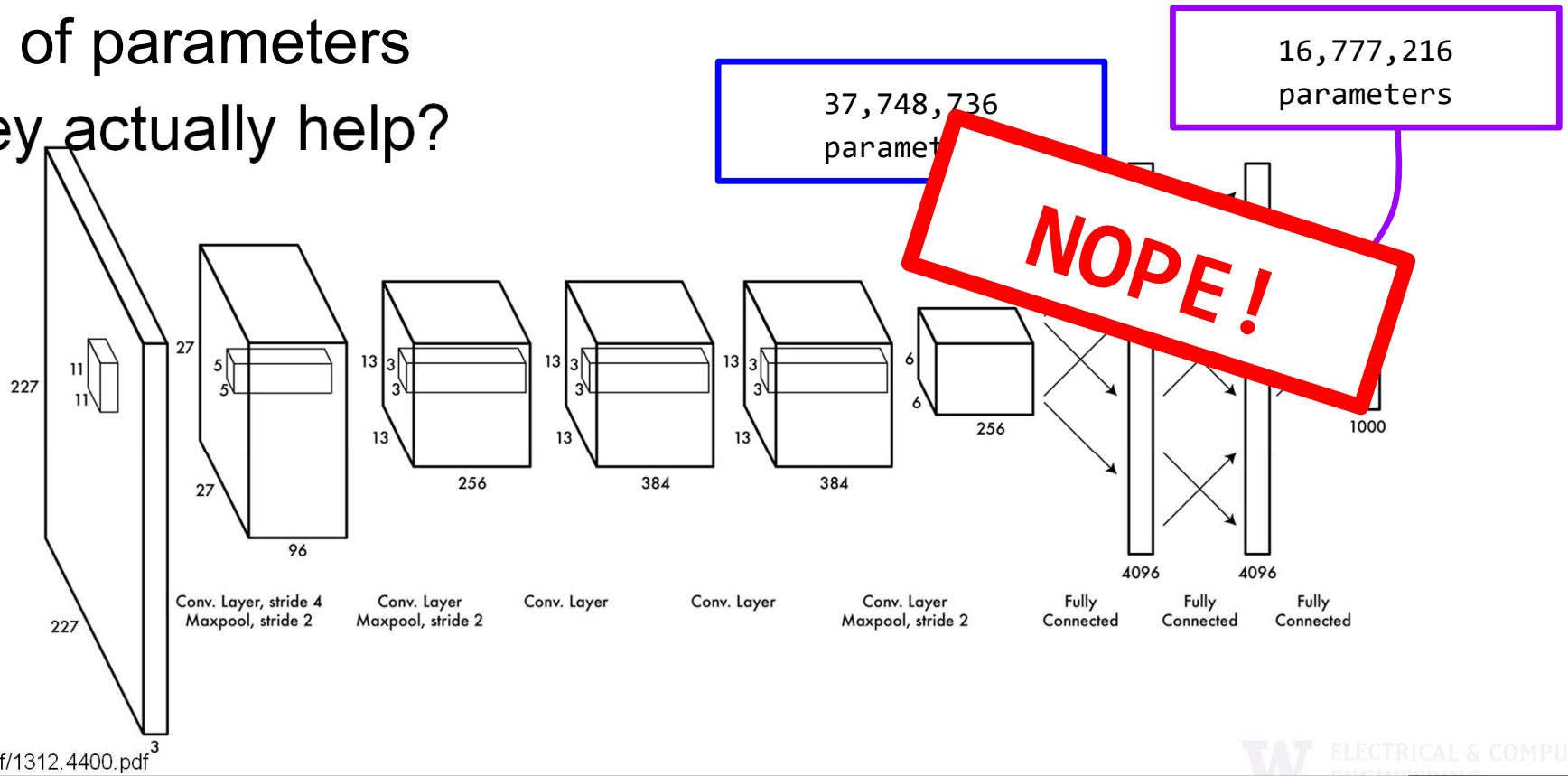
https://arxiv.org/pdf/1312.4400.pdf

VGG (and AlexNet) Inefficient

Huge fully connected layers at the end

TONS of parameters

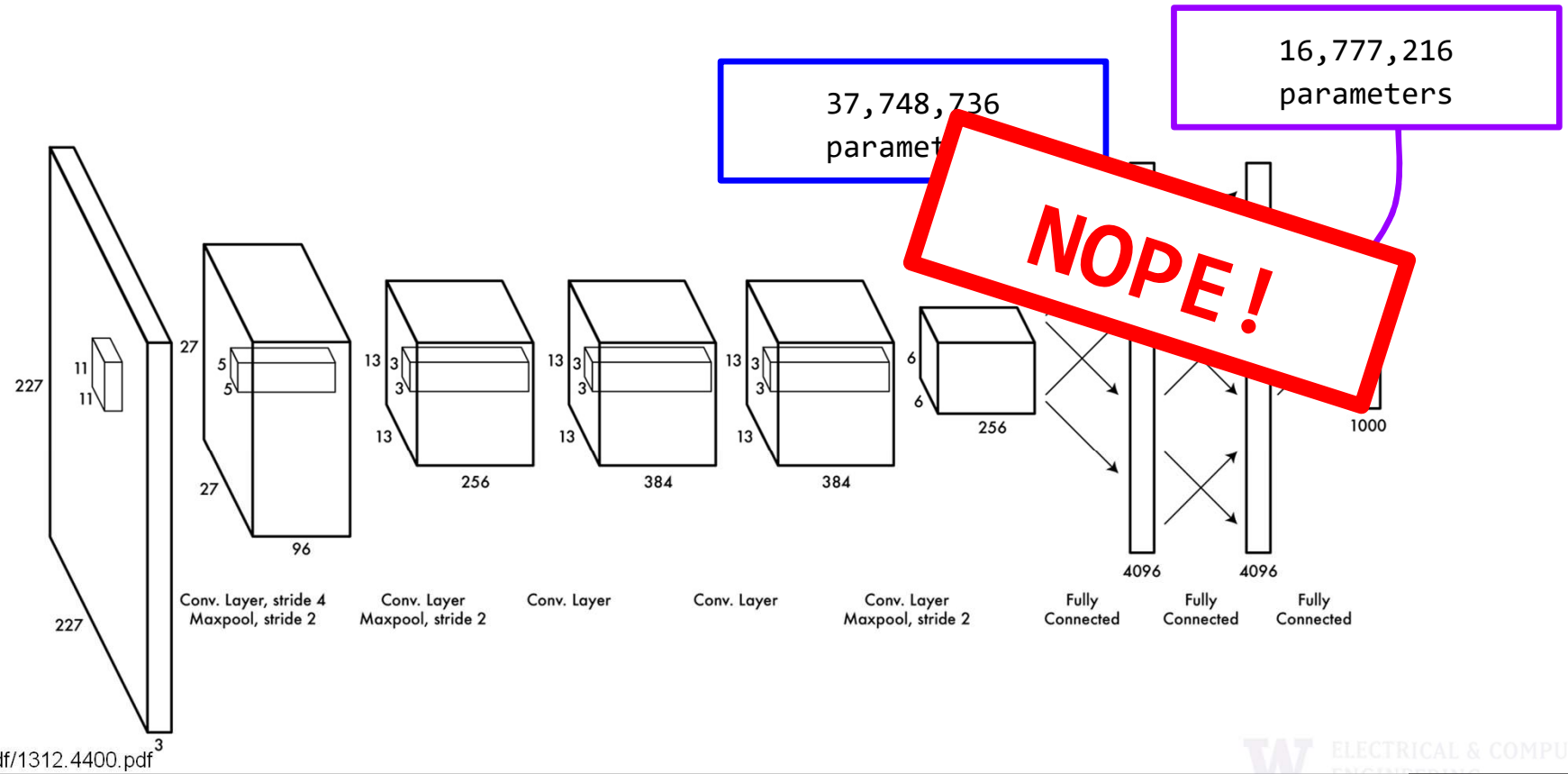
Do they actually help?



https://arxiv.org/pdf/1312.4400.pdf

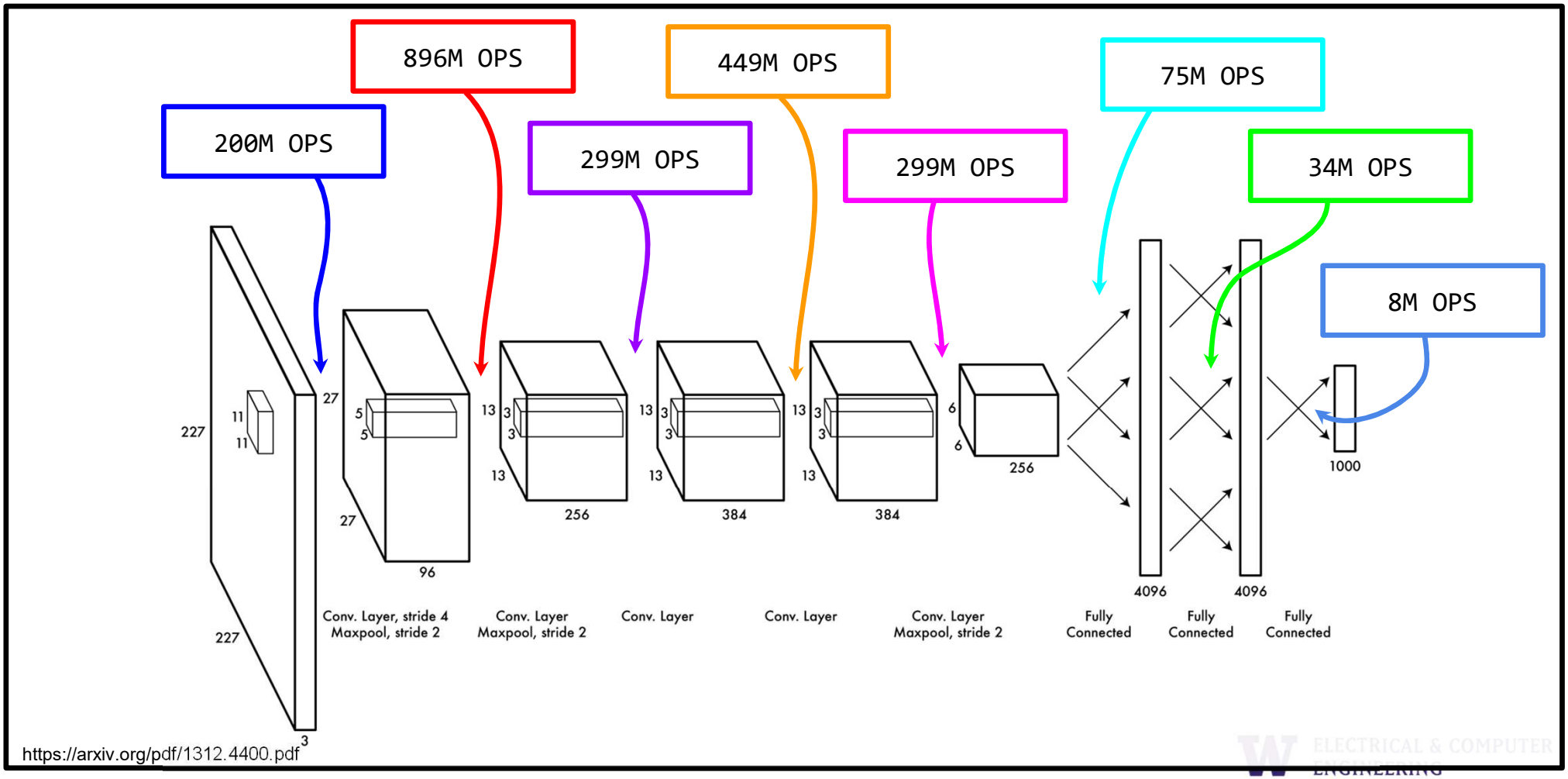
Just use All Convolutions!

How many? How big?



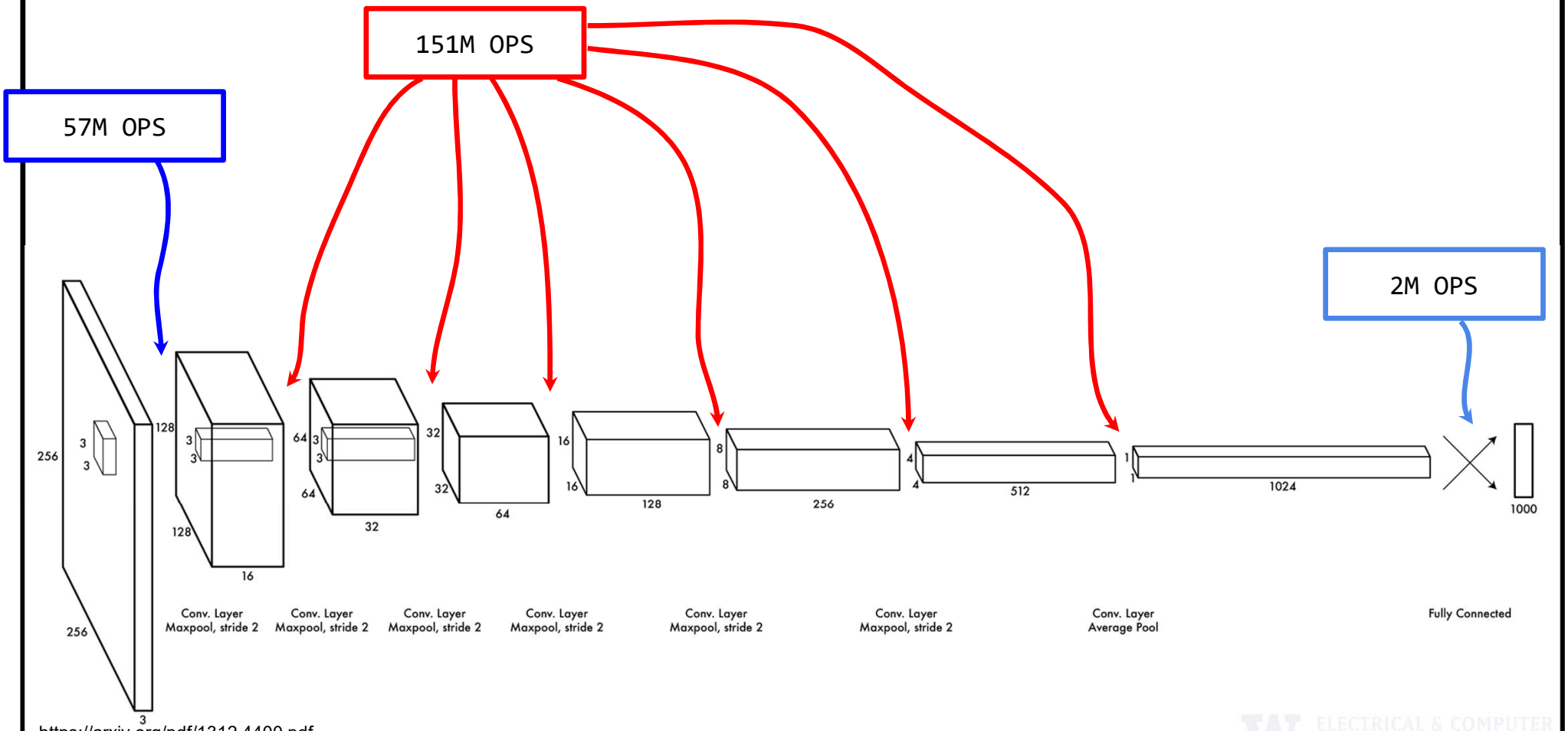
https://arxiv.org/pdf/1312.4400.pdf

AlexNet is Imbalanced



https://arxiv.org/pdf/1312.4400.pdf

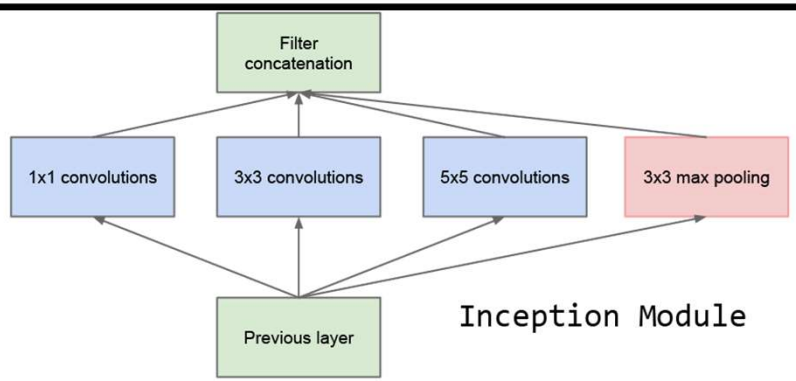
Darknet Reference Network



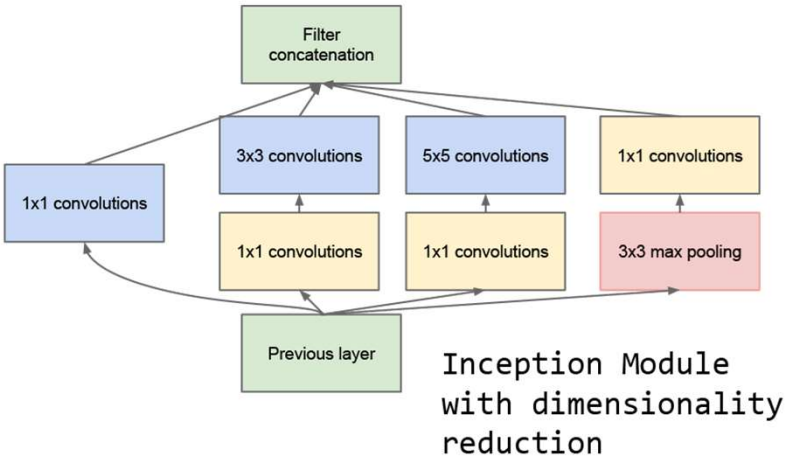
https://arxiv.org/pdf/1312.4400.pdf

GoogleNet: Networks Getting Diversified

Instead of a bunch of layer that are just 3x3 convolutions, use ->



Why are these better?



https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

GoogleNet: networks getting weird

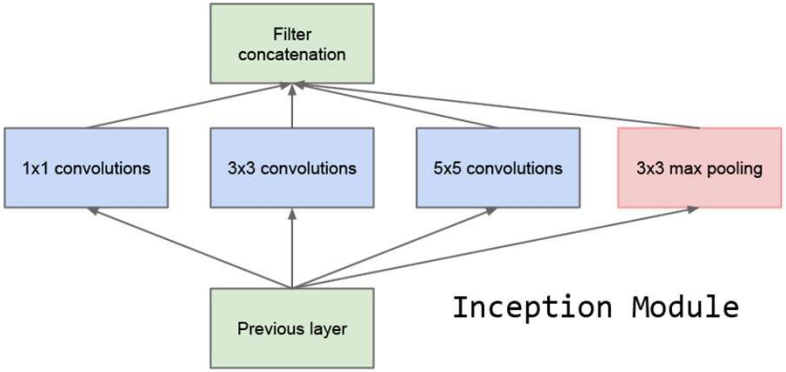
Another way to efficiency:

3x3 convolutions, use ->

Split layers

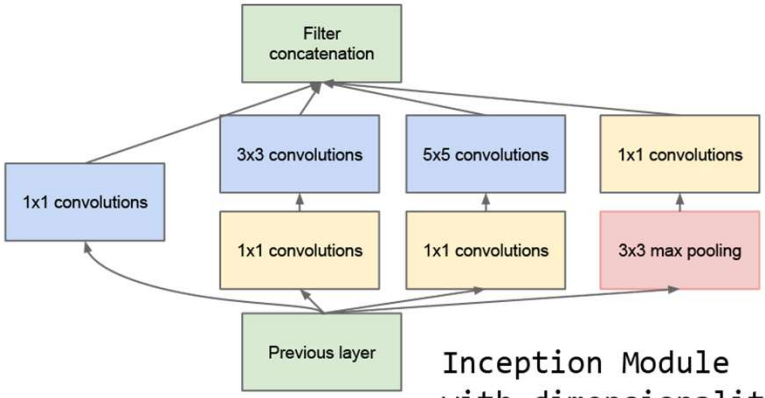
Not just one size, but many

1x1, 3x3, 5x5



Why are these better?

- Also use 1x1 convs to compress feature maps
- Can make large networks that are still efficient, better than VGG yet smaller and faster.



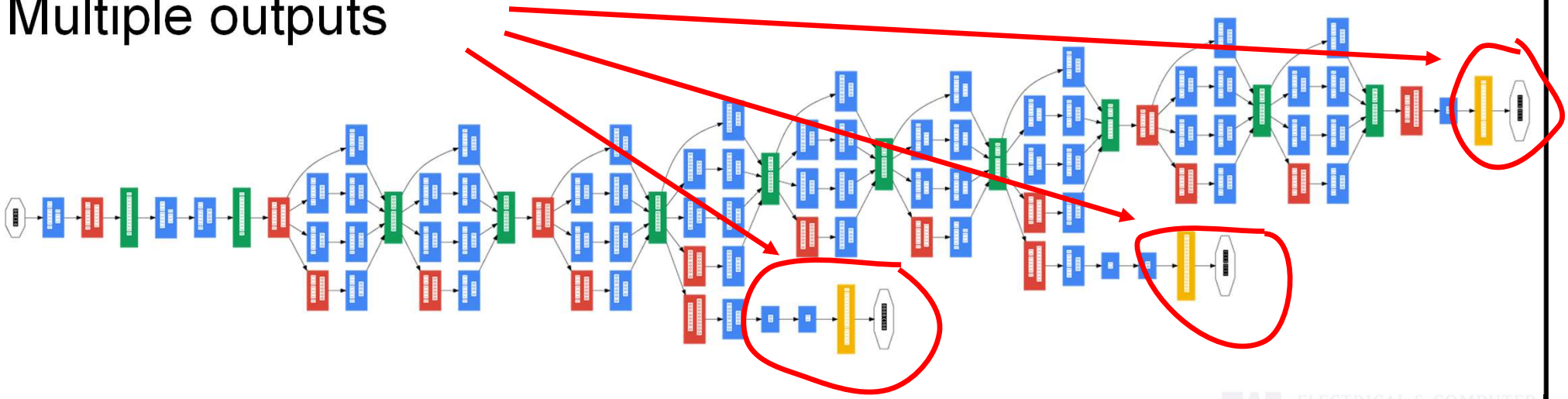
https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

GoogleNet

Like… really big

And this is sort of all we do now, try to make networks bigger without making them too expensive

Multiple outputs



GoogleNetv2 or Inception Net?

Or something, GoogleNet + batch norm

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	double #3×3 reduce	double #3×3	Pool +proj
convolution*	7×7/2	112×112×64	1						
max pool	3×3/2	56×56×64	0						
convolution	3×3/1	56×56×192	1		64	192			
max pool	3×3/2	28×28×192	0						
inception (3a)		28×28×256	3	64	64	64	64	96	avg + 32
inception (3b)		28×28×320	3	64	64	96	64	96	avg + 64
inception (3c)	stride 2	28×28×576	3	0	128	160	64	96	max + pass through
inception (4a)		14×14×576	3	224	64	96	96	128	avg + 128
inception (4b)		14×14×576	3	192	96	128	96	128	avg + 128
inception (4c)		14×14×576	3	160	128	160	128	160	avg + 128
inception (4d)		14×14×576	3	96	128	192	160	192	avg + 128
inception (4e)	stride 2	14×14×1024	3	0	128	192	192	256	max + pass through
inception (5a)		7×7×1024	3	352	192	320	160	224	avg + 128
inception (5b)		7×7×1024	3	352	192	320	192	224	max + 128
avg pool	7×7/1	1×1×1024	0						

Figure 5: Inception architecture

https://arxiv.org/pdf/1502.03167.pdf

Residual Connections

Normally, output of two layers is: f(w*f(vx))
Residual connections: f(w*f(vx) + x)

Learning how to modify x, add some transformed amount

Gives delta another path, less vanishing gradient

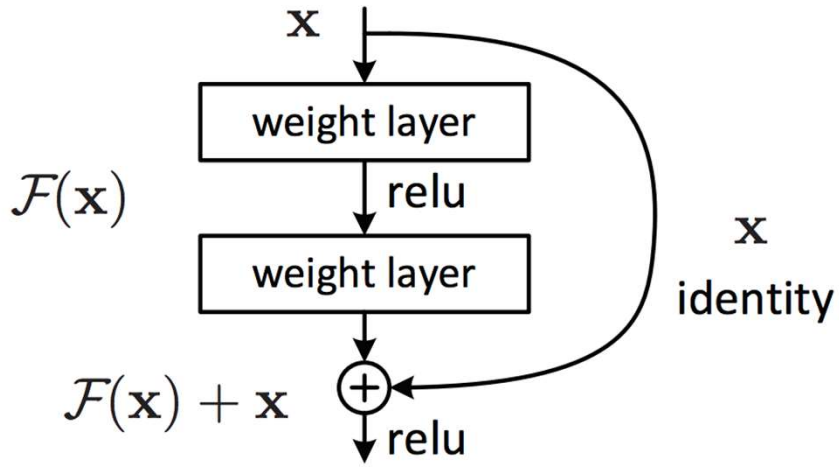
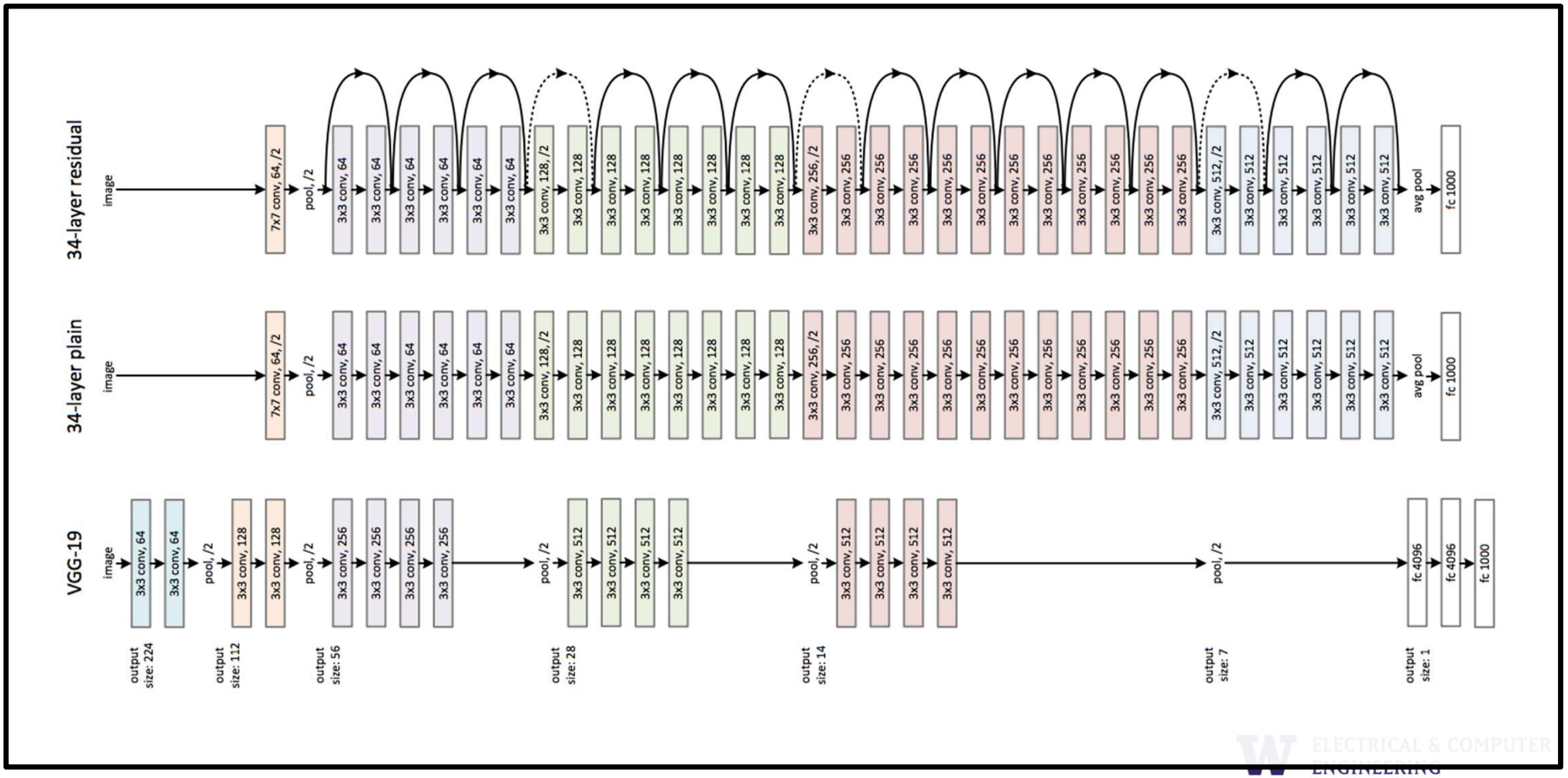


Figure 2. Residual learning: a building block.

ResNet


VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

ResNet

3x3 conv blocks or 3x3 and 1x1 conv blocks

Residual connections

VERY deep, 100+ layers

Grouped convolutions

Grouped convolutions:

- Split up input feature map intro groups
- Run convs on groups independently
- Recombine

E.g. 3x3 conv layer 32 x 32 x 256 input, 128 filters, 32 groups:

- Split input into 32 different feature maps
  - Each is 32 x 32 x 8
  - Run 4 filters, 3x3x8 on each group
  - Merge 4*32 channels back together, get 32 x 32 x 128 output

Input, output stays same dimensions, less computation

ResNeXt

Replace 3x3 blocks with larger grouped convs

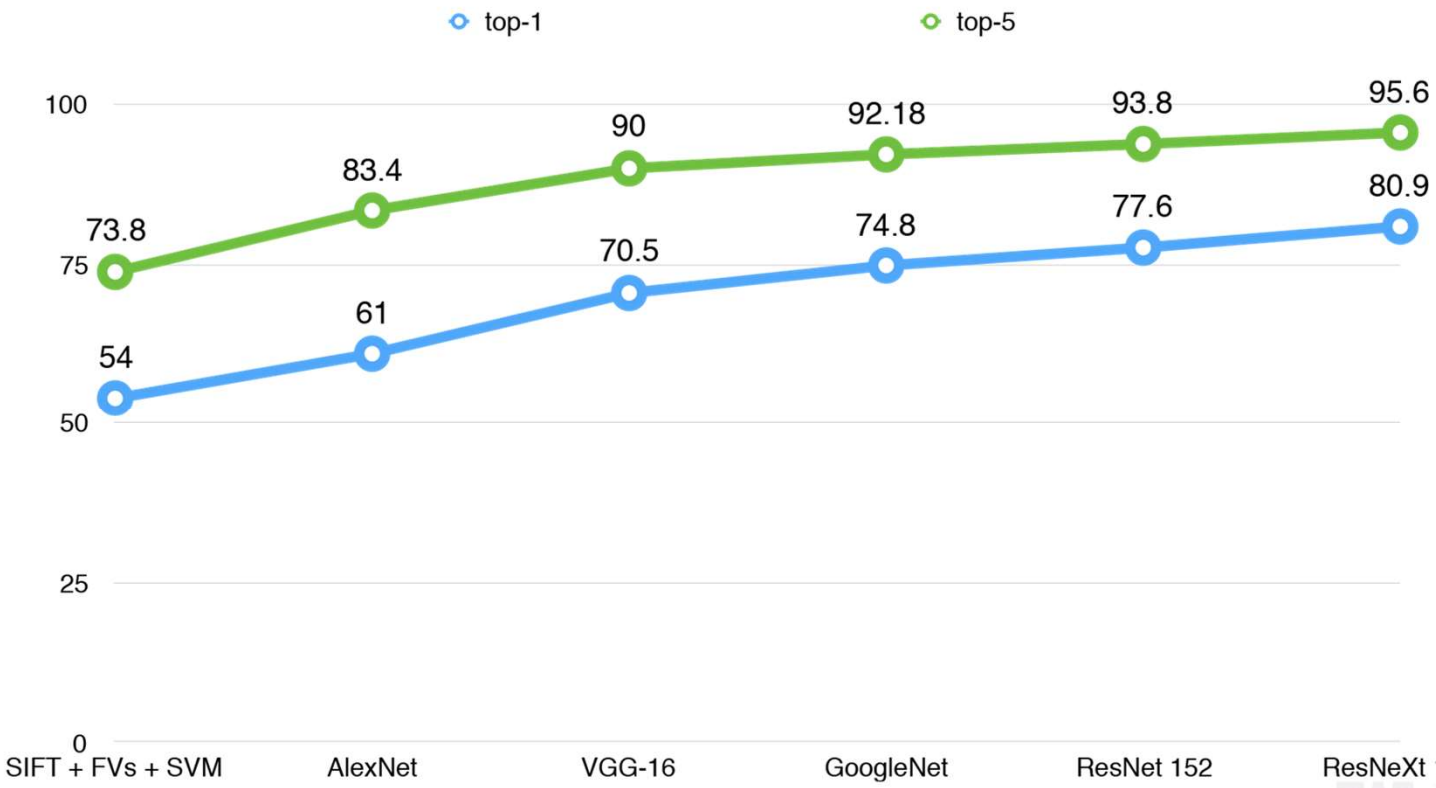
“Larger” network but same computational complexity

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

https://arxiv.org/pdf/1611.05431.pdf

What's NeXt?

Starting to saturate ImageNet, fighting over 1-2%



Training Classifier

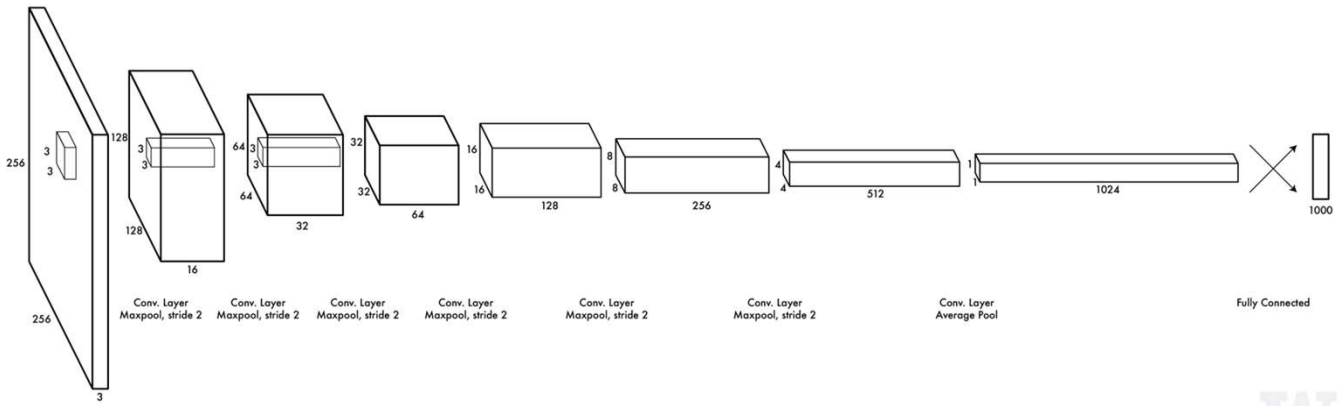
Typically you have a much smaller dataset than 1.2 million images and 1,000 classes.

What problems to do we encounter with less data, say 10,000 images and 500 classes?

First ImageNet, then the world!

For dealing with smaller datasets where we might overfit, *pretraining* is key:

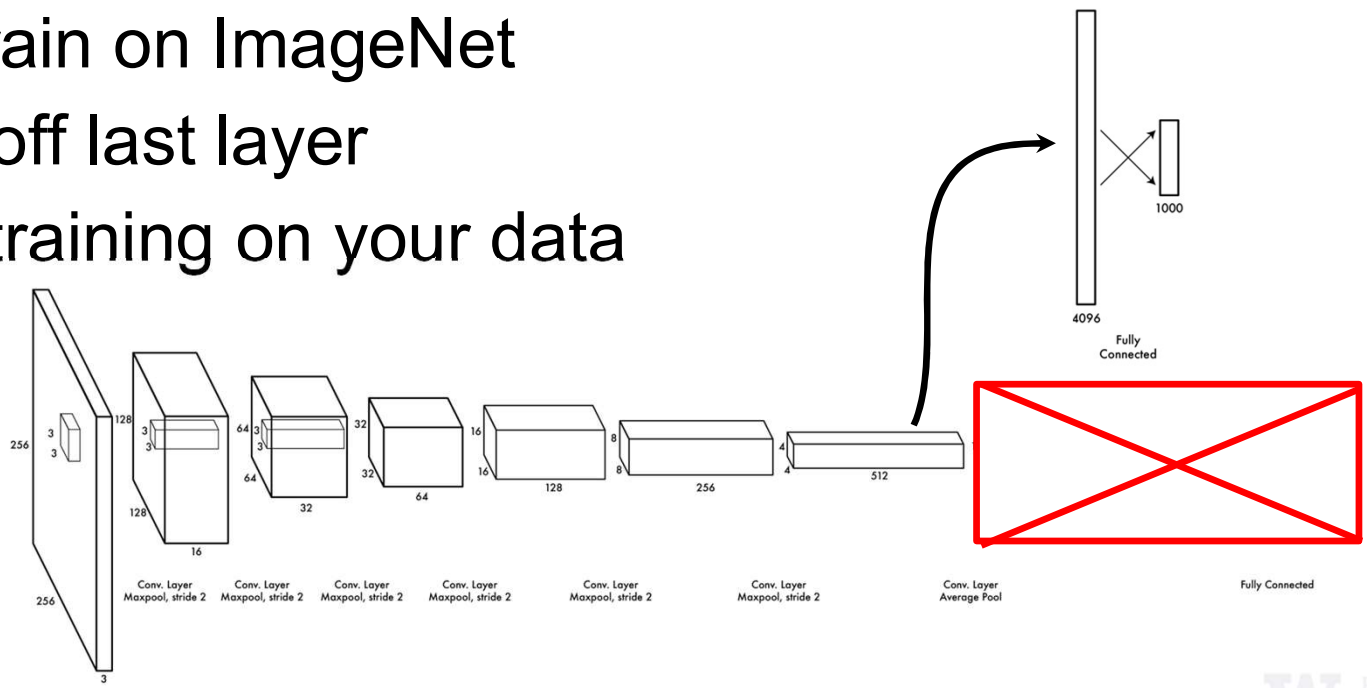
- First train on ImageNet
- Chop off last layer
- Keep training on your data



First ImageNet, then the world!

For dealing with smaller datasets where we might overfit, *pretraining* is key:

- First train on ImageNet
- Chop off last layer
- Keep training on your data



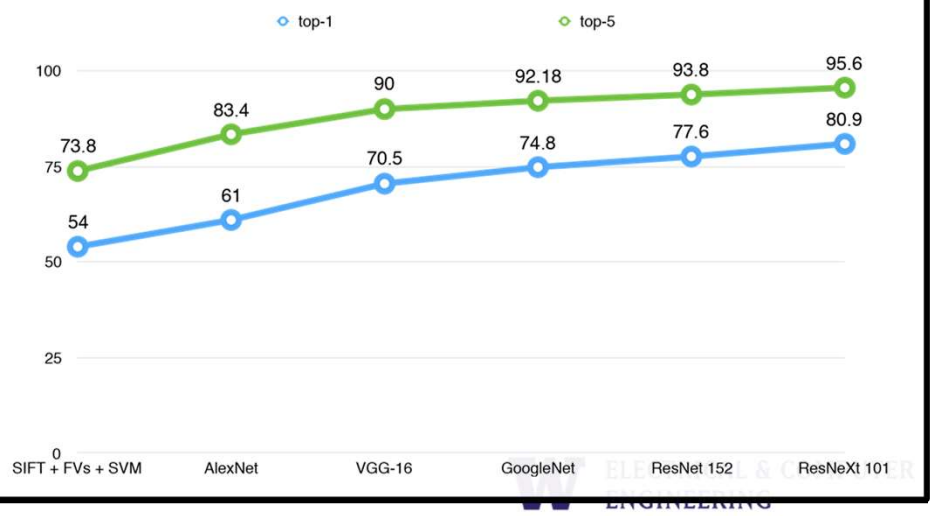
What else is NeXt?

Starting to saturate ImageNet, fighting over 1-2%

But now vision really *works*, other tasks

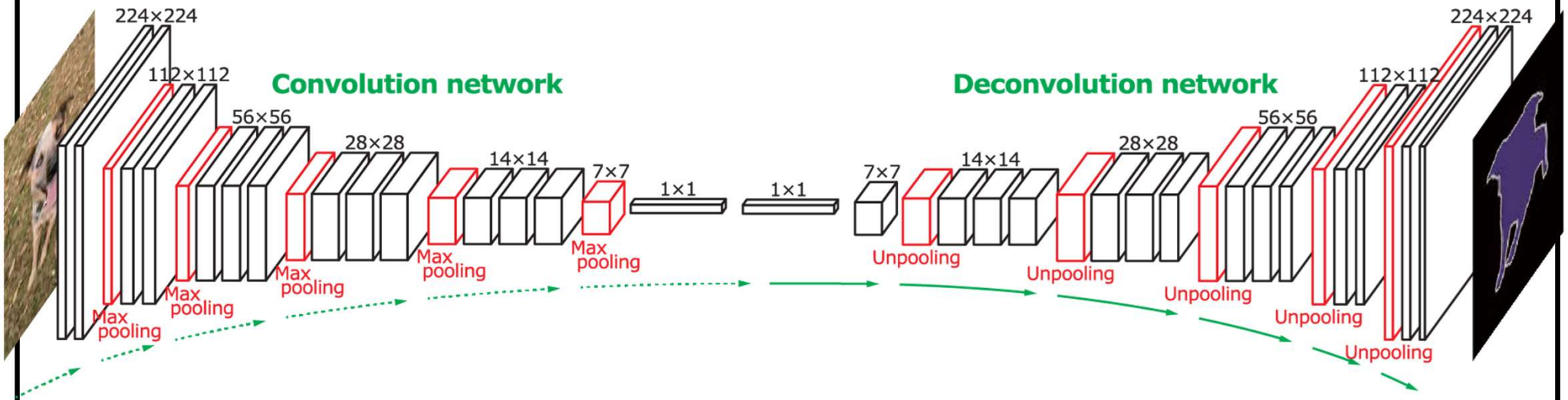
Segmentation
Object detection
Captioning
...

Just different loss functions!



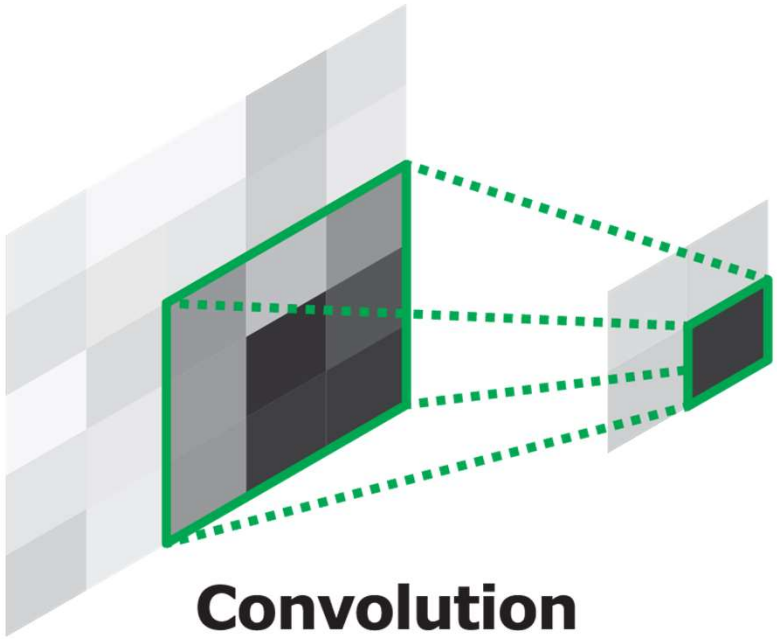
Semantic Segmentation

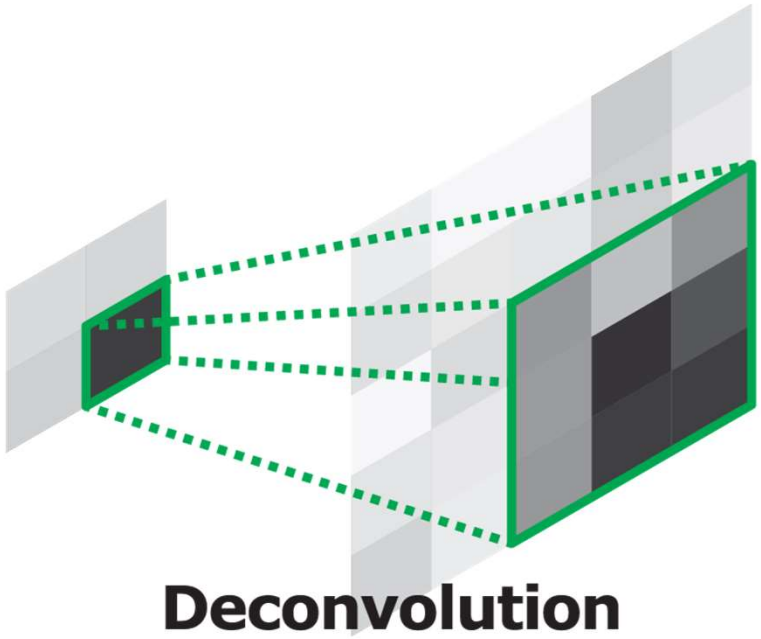
Semantic Segmentation



https://arxiv.org/pdf/1505.04366.pdf

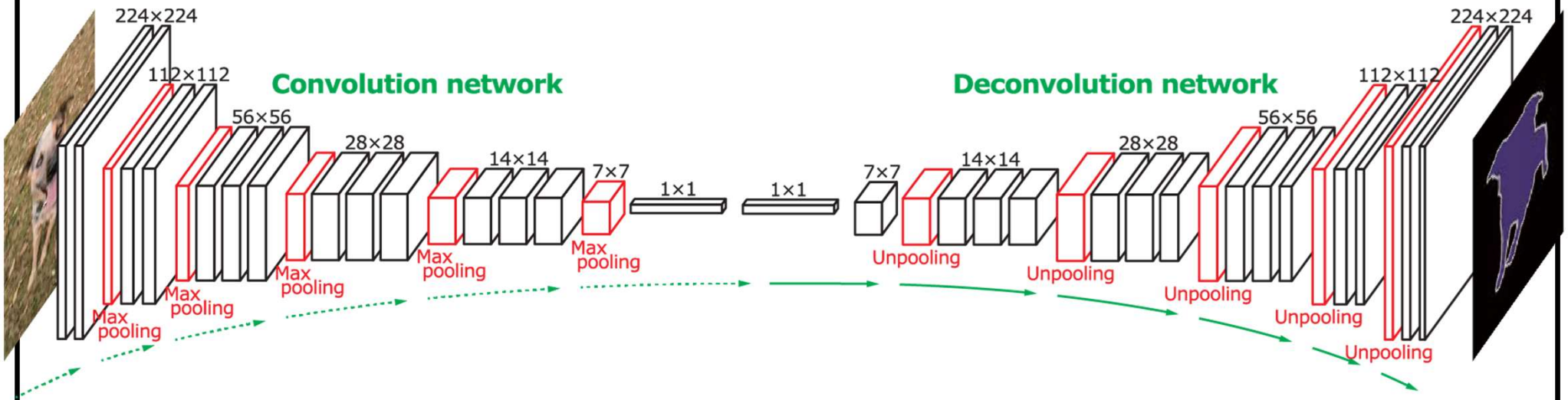
Semantic Segmentation





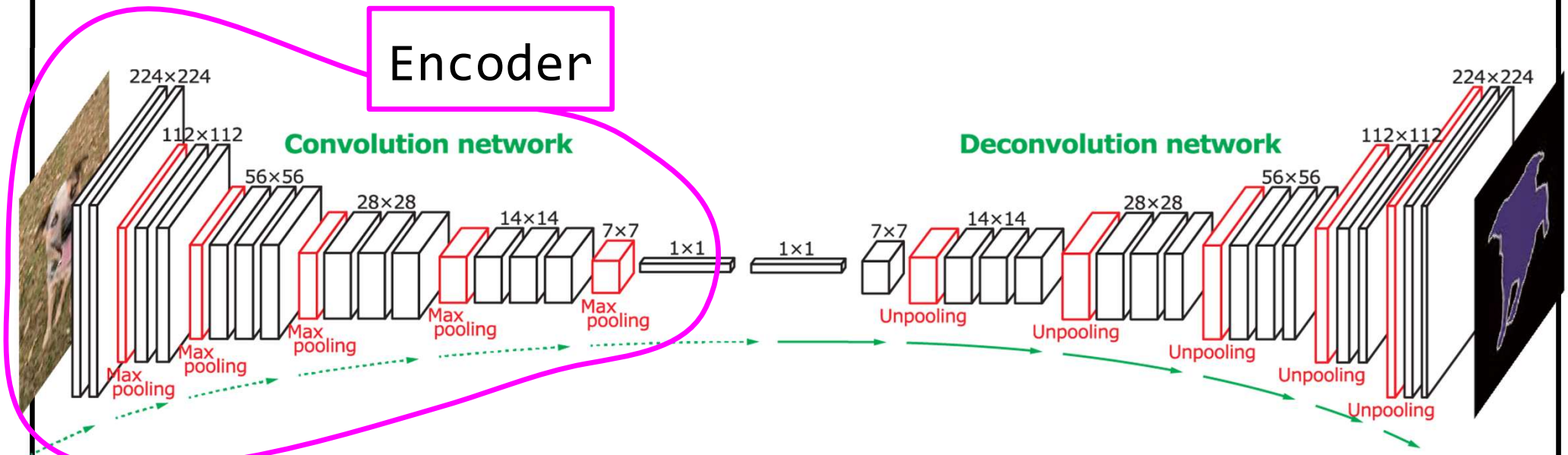
https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation



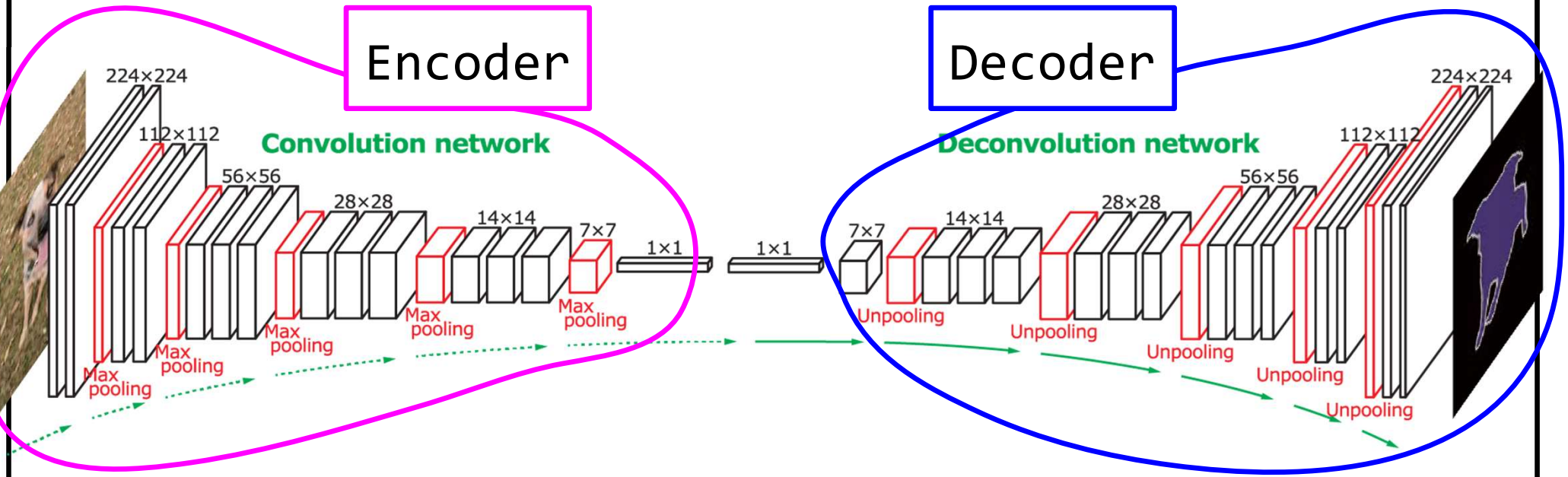
https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation



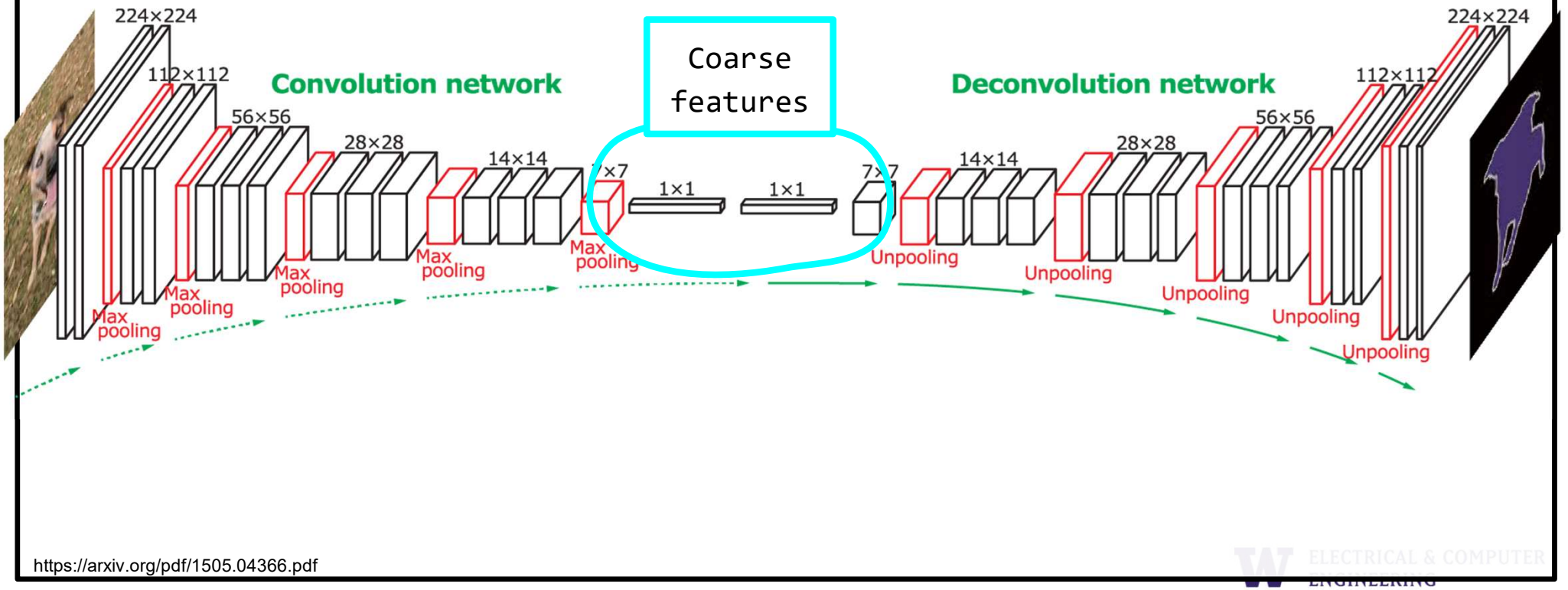
https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation



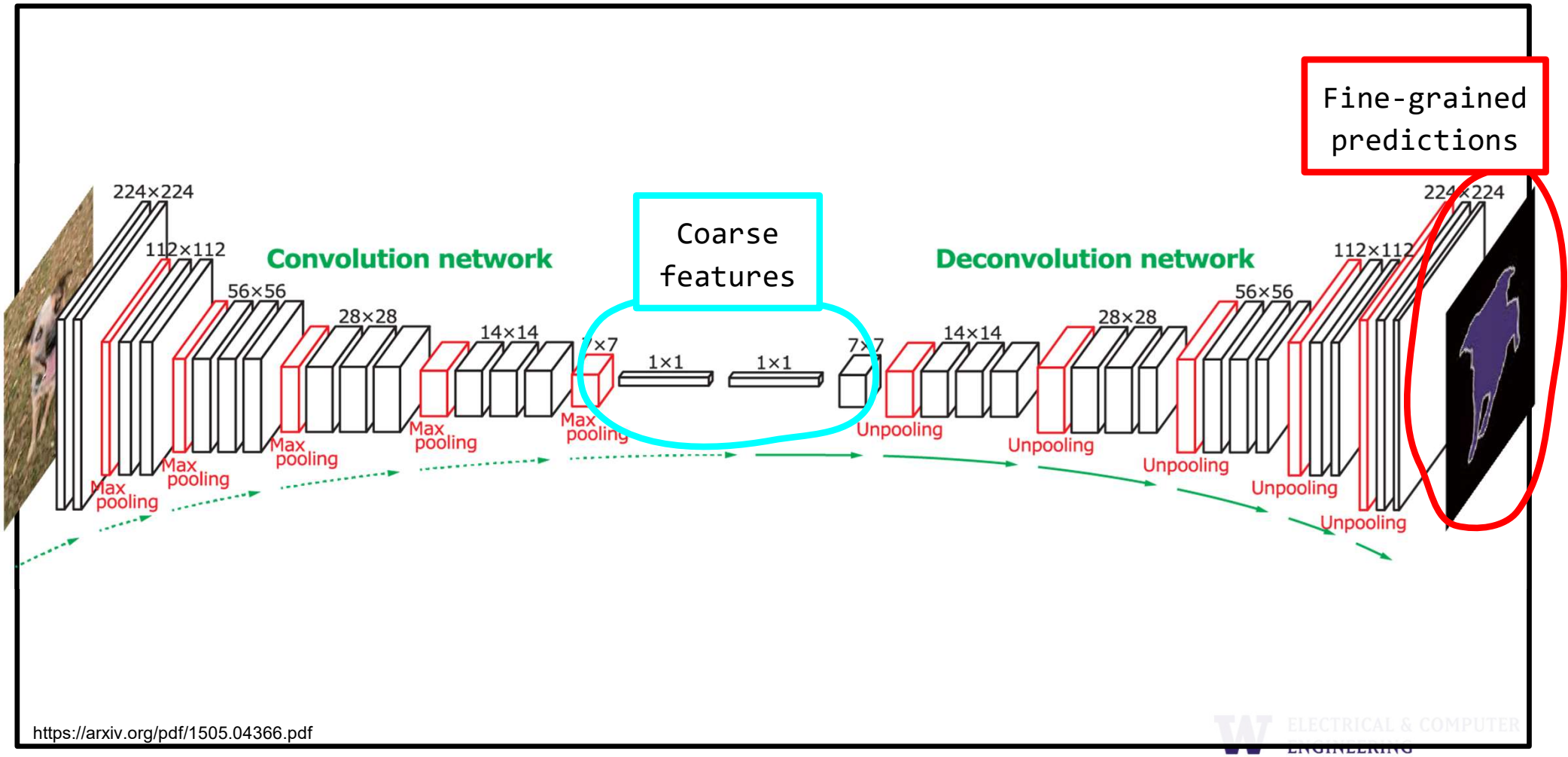
https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation



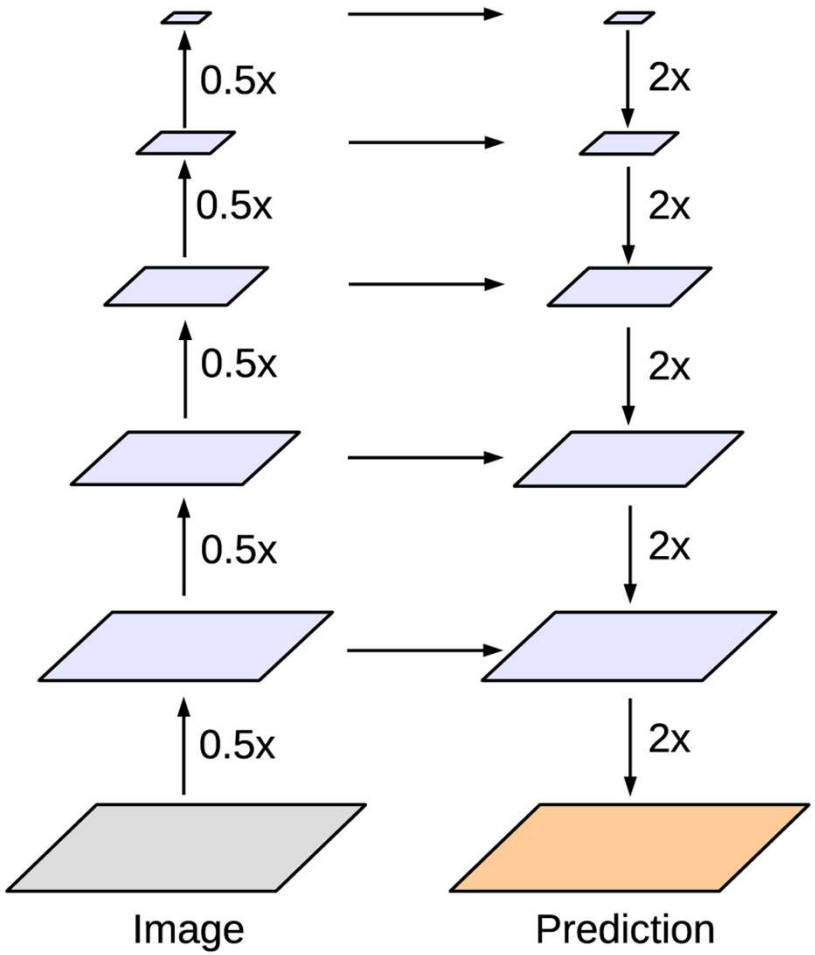
https://arxiv.org/pdf/1505.04366.pdf

Semantic Segmentation



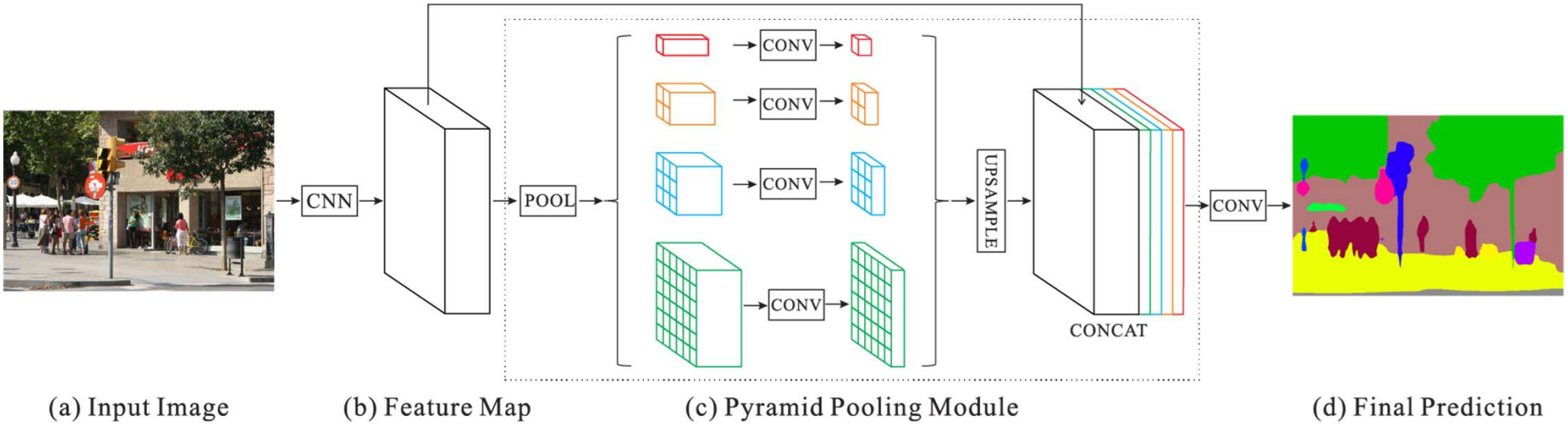
https://arxiv.org/pdf/1505.04366.pdf

U-net/Segnet



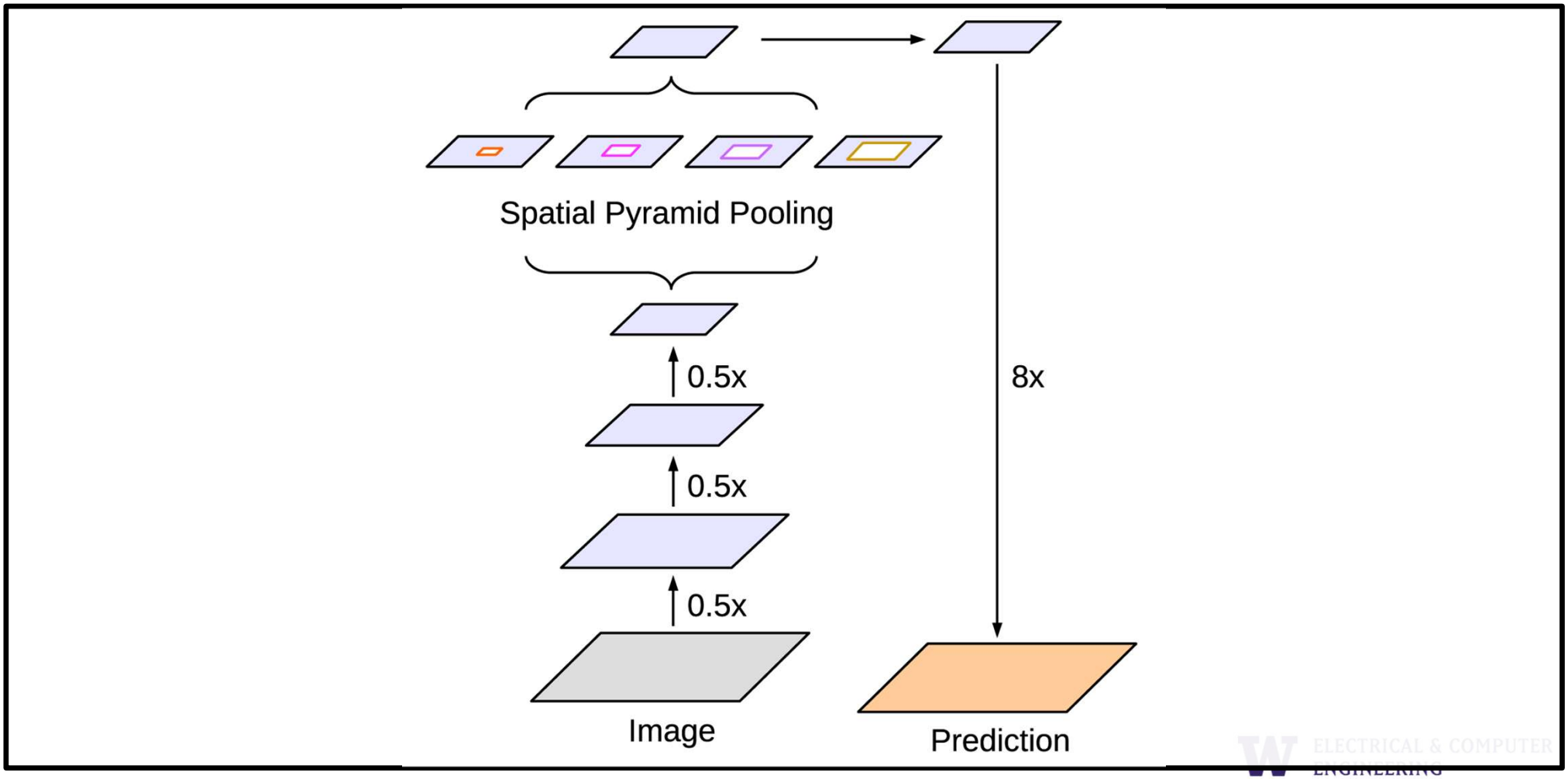
https://arxiv.org/pdf/1511.00561.pdf, https://arxiv.org/pdf/1505.04597.pdf

Spatial Pyramid Pooling



https://arxiv.org/pdf/1612.01105.pdf

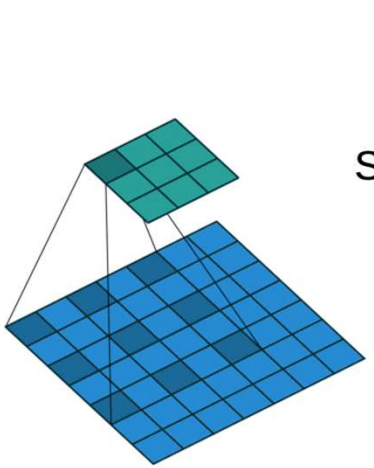
Spatial Pyramid Pooling



DeepLabv3+

Atrous convolutions

Spaced inputs



https://arxiv.org/pdf/1802.02611.pdf

PASCAL VOC

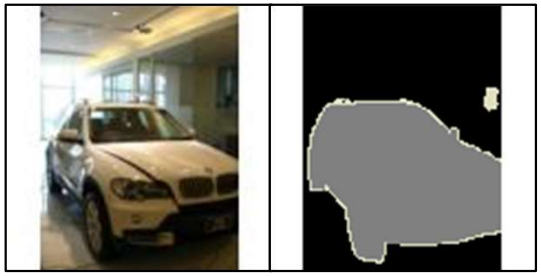
Segmentation dataset:

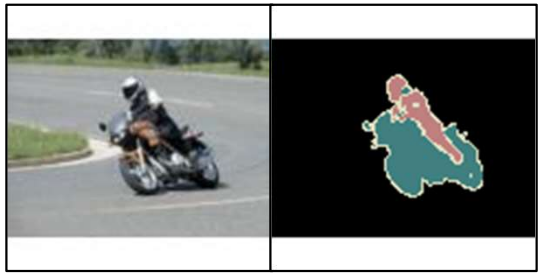
20 classes

9,993 images with pixel-level labels

Classes:

aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse motorbike, person, pottedplant, sheep, sofa, train, tvmonitor





Cityscapes

Segmentation dataset:

30 classes, 50 cities, different seasons, daytime, good weather

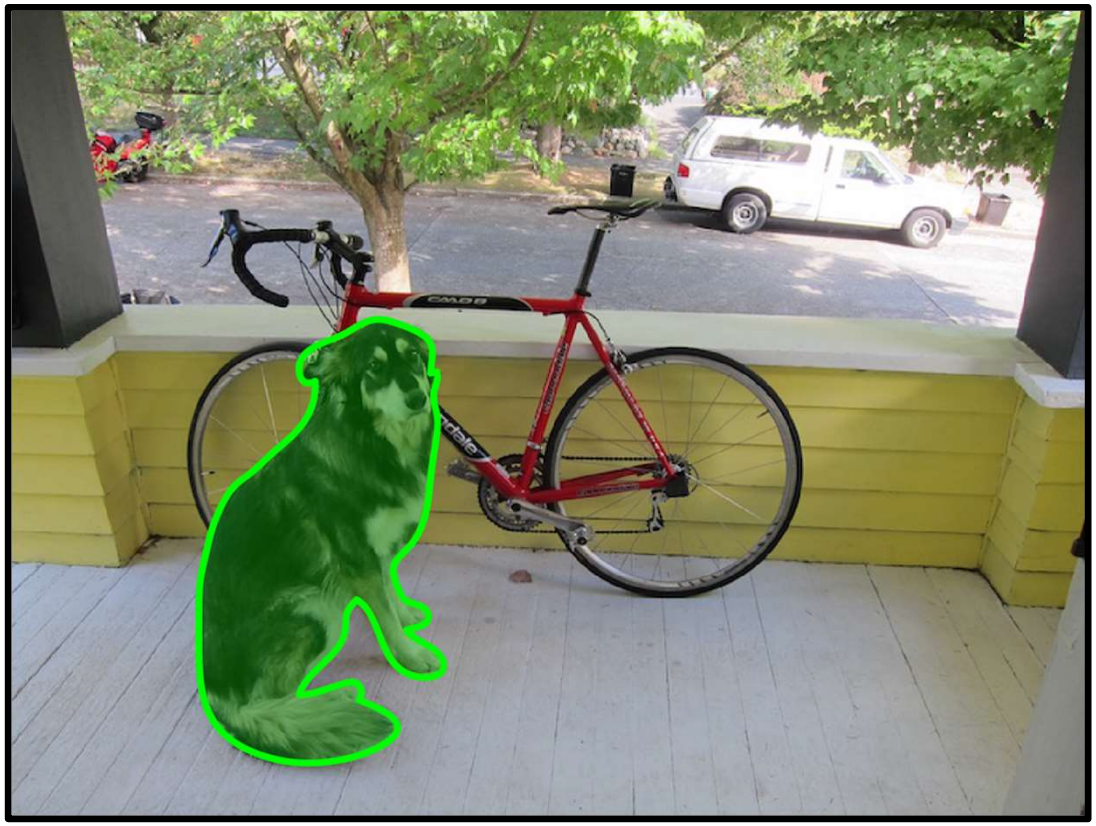
5,000 images with fine labels, 20,000 with dense labels

Classes:

road, sidewalk, parking, rail track, building, wall, fence, guard rail, bridge, tunnel, pole, polegroup, traffic light, traffic sign, vegetation, terrain, sky, person, rider, car, truck, bus, caravan, trailer, train, motorcycle, bicycle, license plate

Evaluating Segmentation: IoU

Intersection over Union = TP / (TP + FP + FN)

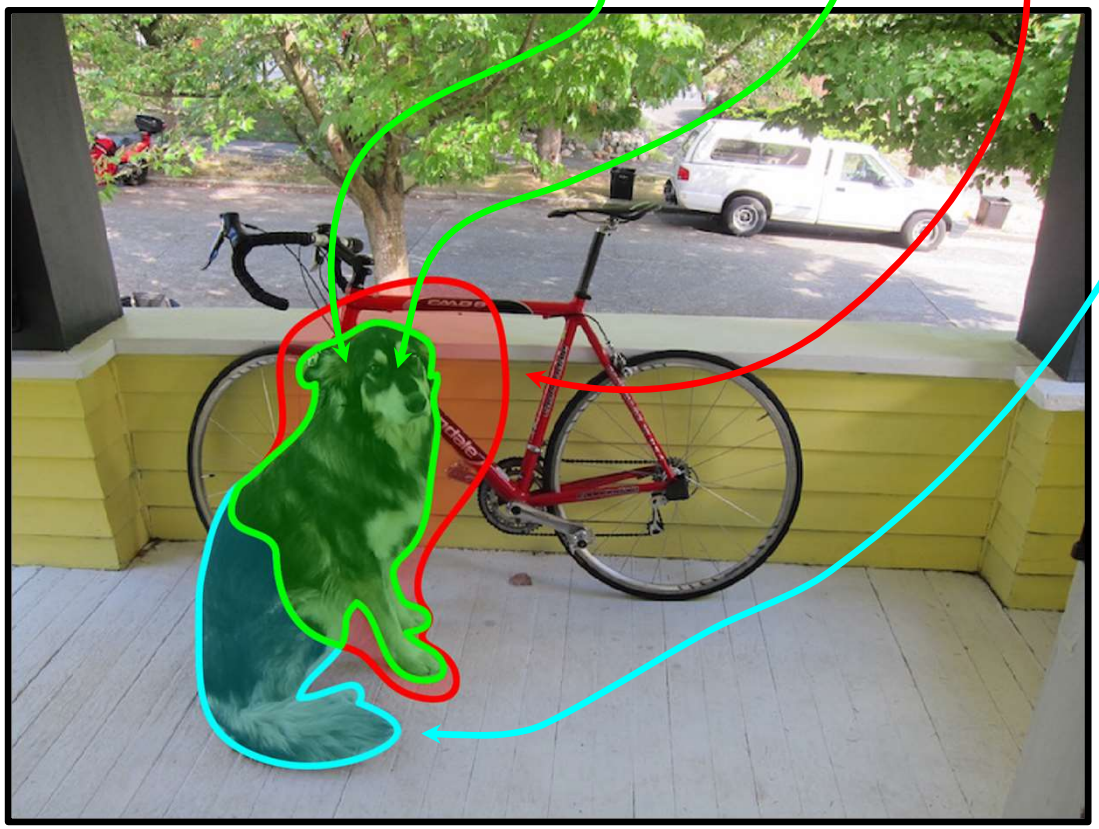


Evaluating Segmentation: IoU

Intersection over Union = TP / (TP + FP + FN)

Evaluating Segmentation: IoU

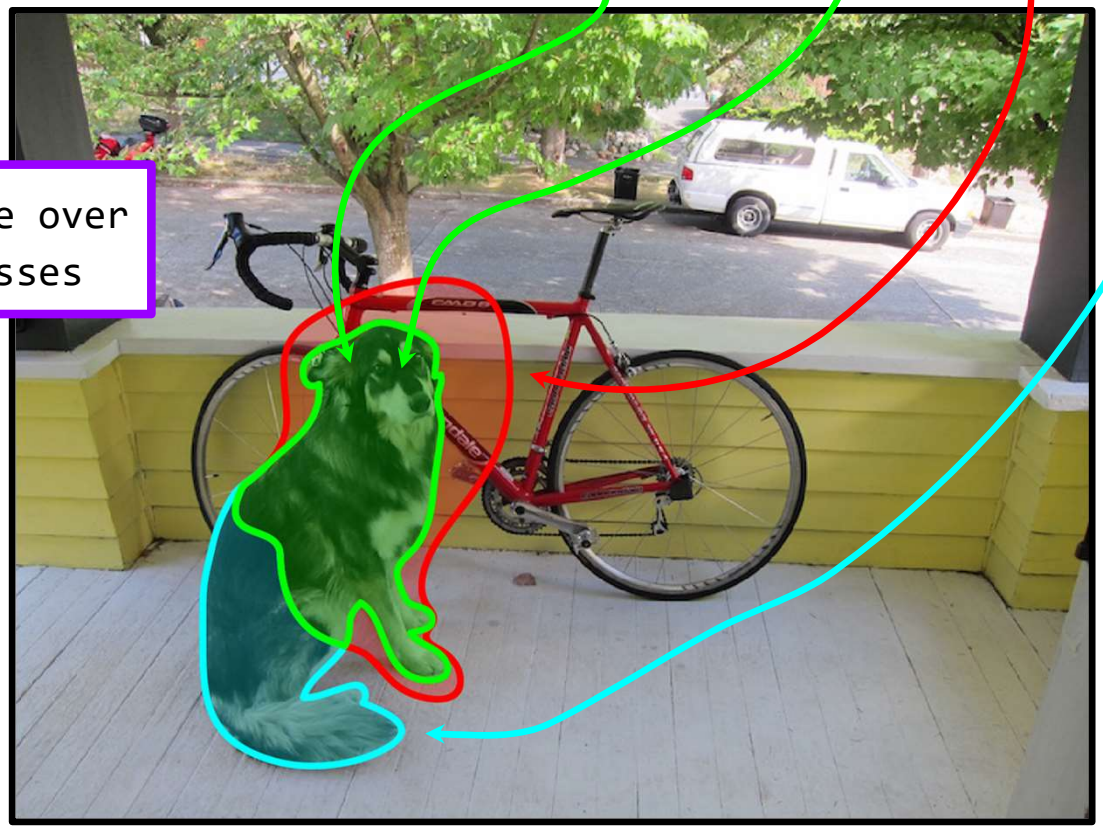
Intersection over Union = TP / (TP + FP + FN)



Evaluating Segmentation: IoU

Intersection over Union = TP / (TP + FP + FN)

Typically average over all images/classes

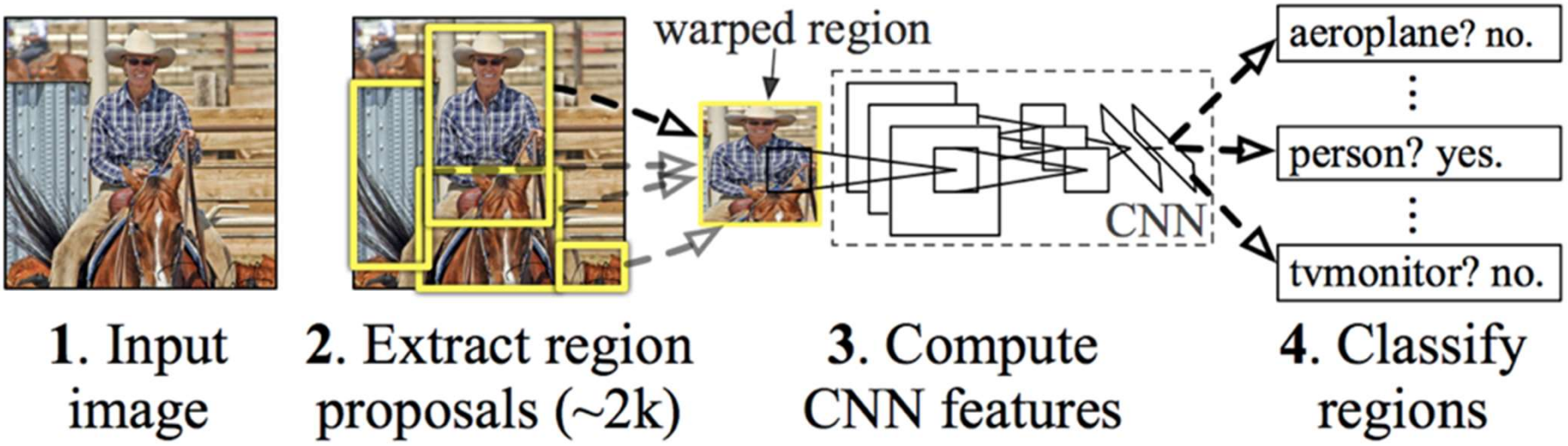


PASCAL VOC

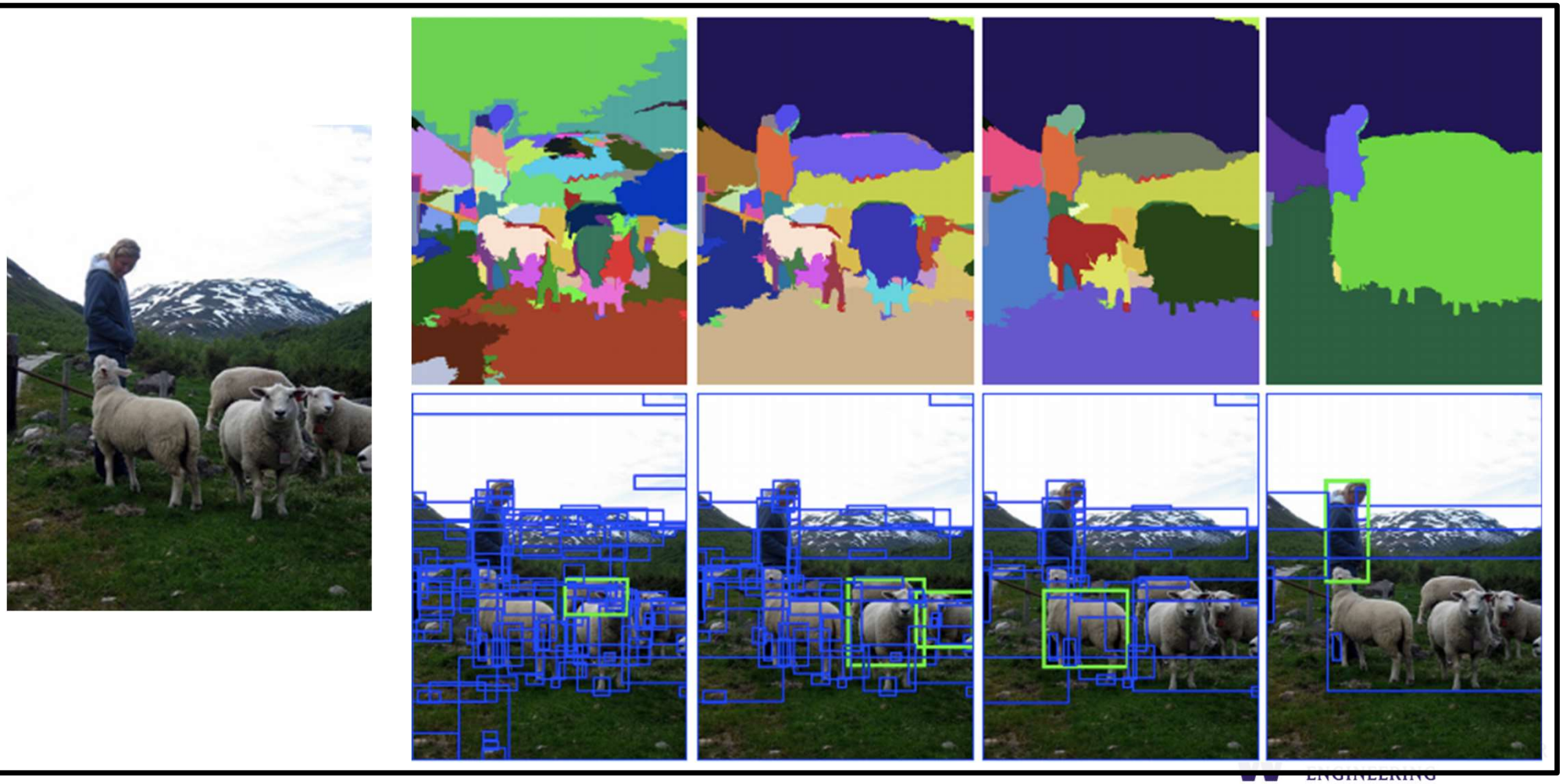
One of the first large detection datasets:

- 20 classes
- 11,530 training images
- 27,450 annotated objects

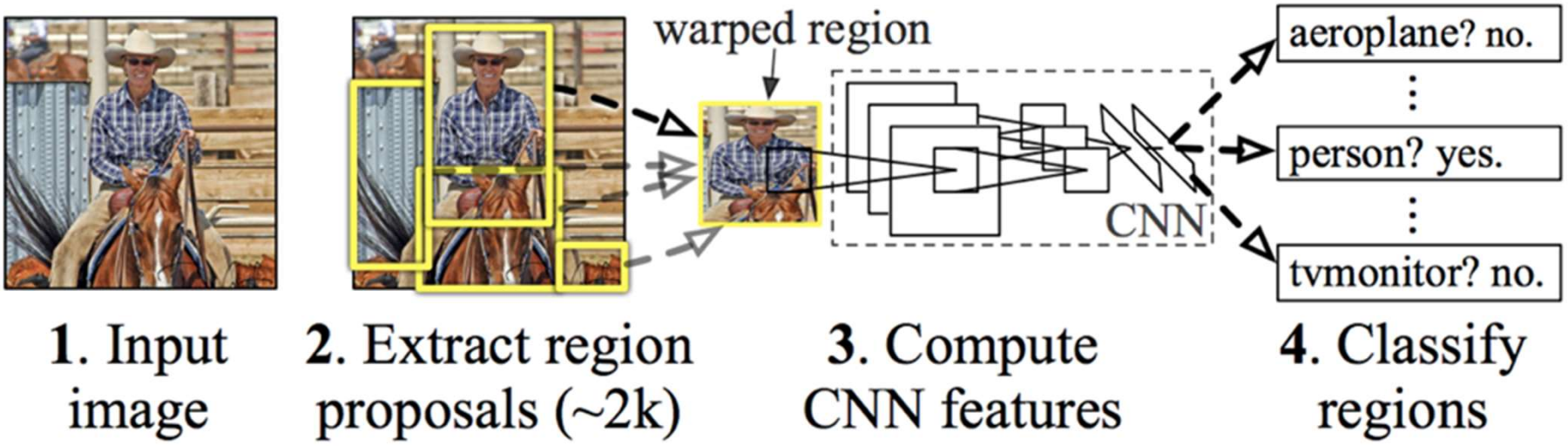
R-CNN: Regions with CNN features



Selective search: fewer proposals



R-CNN: Regions with CNN features



Lots of post processing, 20 sec/im

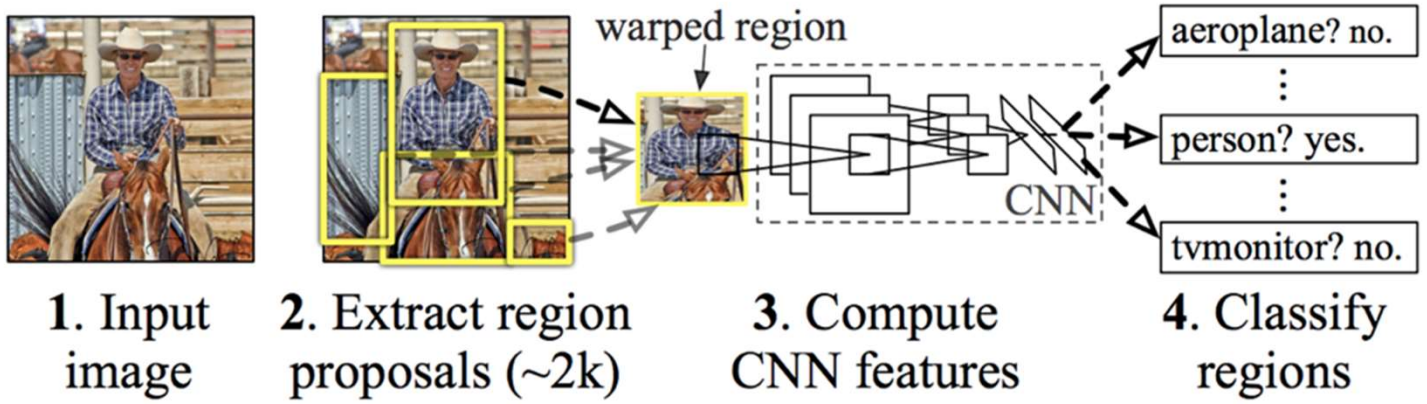
Pascal VOC:

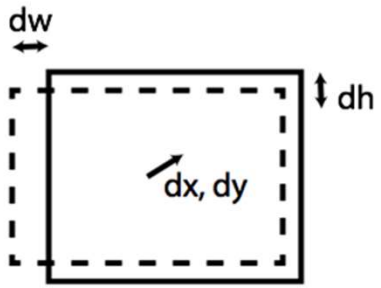
AlexNet

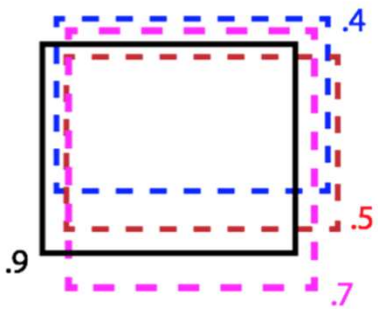
53.3% mAP

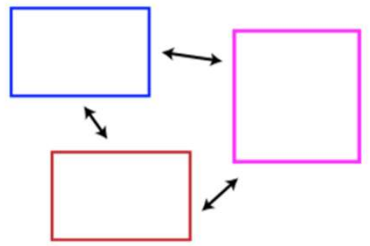
VGG-16

62.4% mAP









Scoring object detection

Multiple classes, multiple objects per images

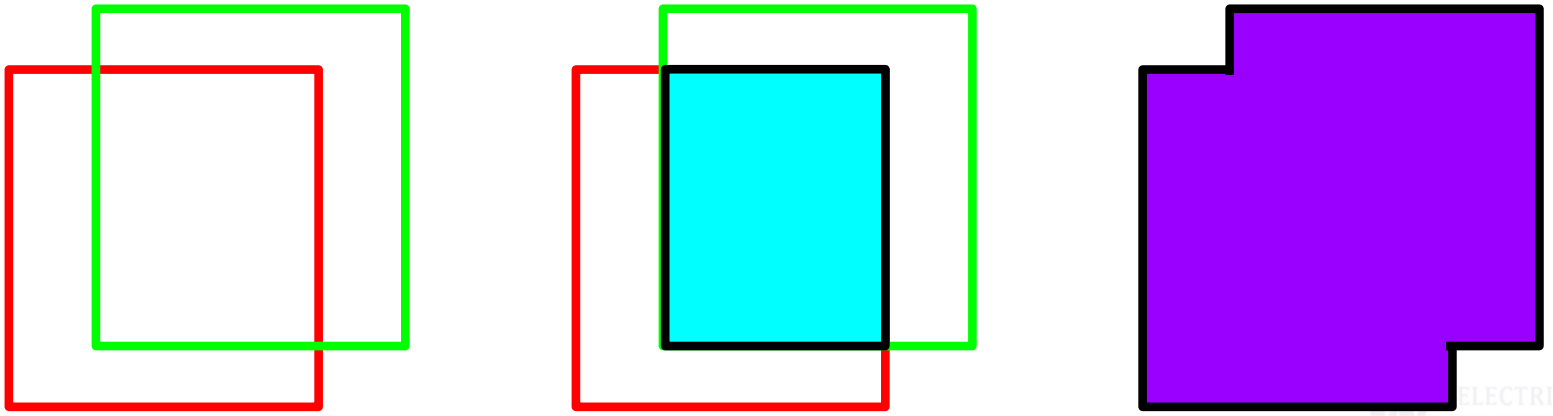
Can’t just use accuracy

“Correct” bounding box:

Intersection / Union > 0.5

Intersection: Ground truth ∩ prediction

Union: Ground truth ∪ prediction



Scoring object detection

"Correct" bounding box:

Intersection / Union > 0.5

Recall:

Correct bounding boxes / total ground-truth boxes

Precision:

Correct bounding boxes / total predicted boxes

Only the most confident predictions: High precision, low recall

All the predictions: Low precision, high recall

Scoring object detection

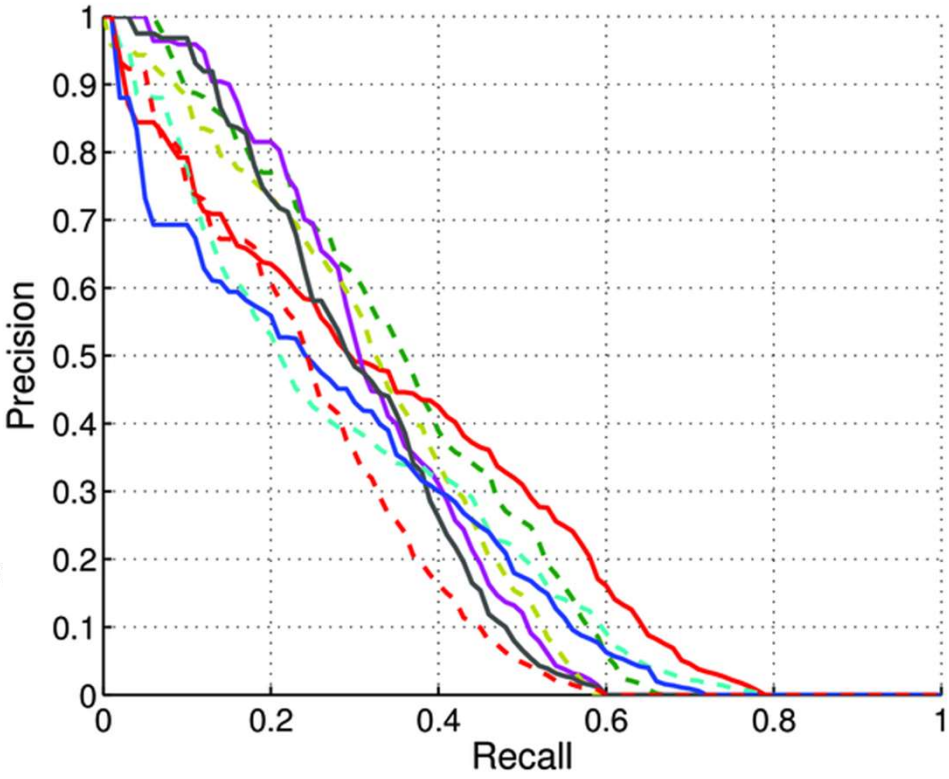
Precision-Recall curve: vary threshold, plot precision and recall

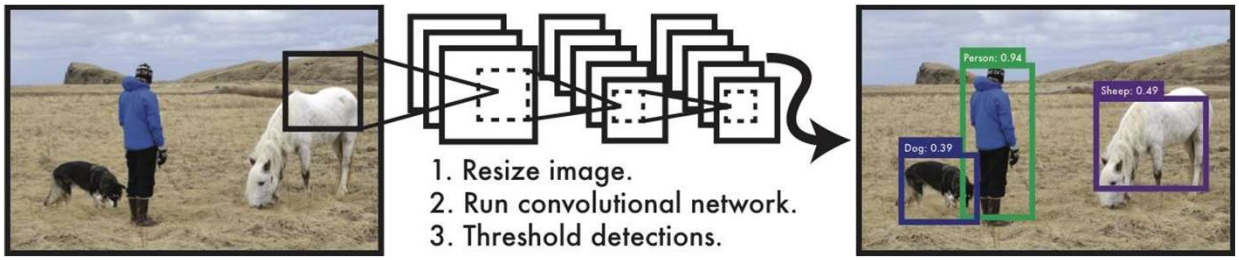
Average precision:

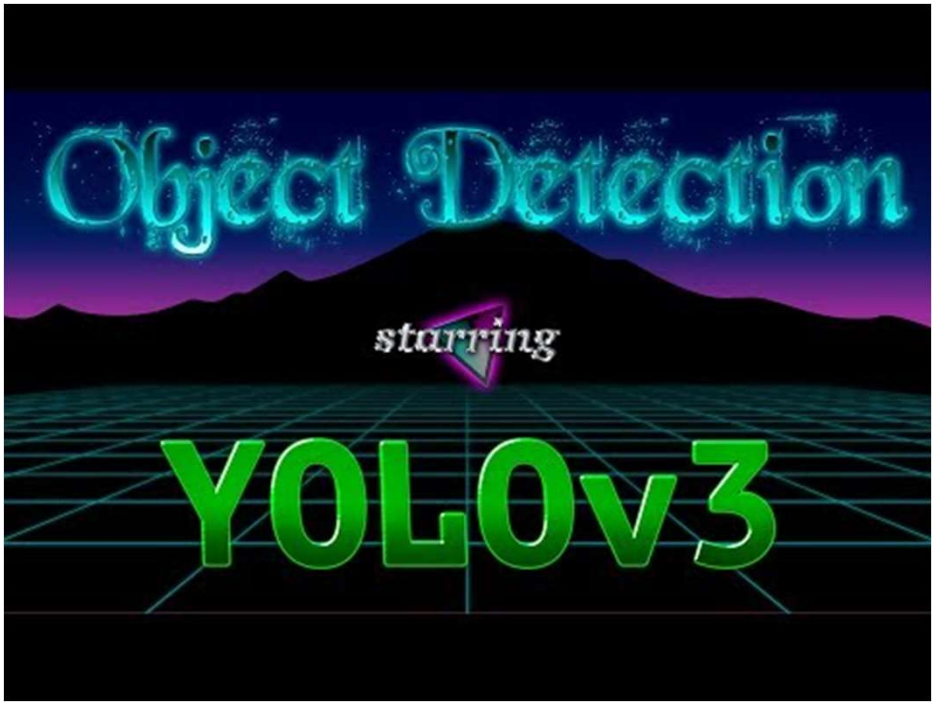
- Area under PR curve
- Only for a single class

Take mean of AP across classes:

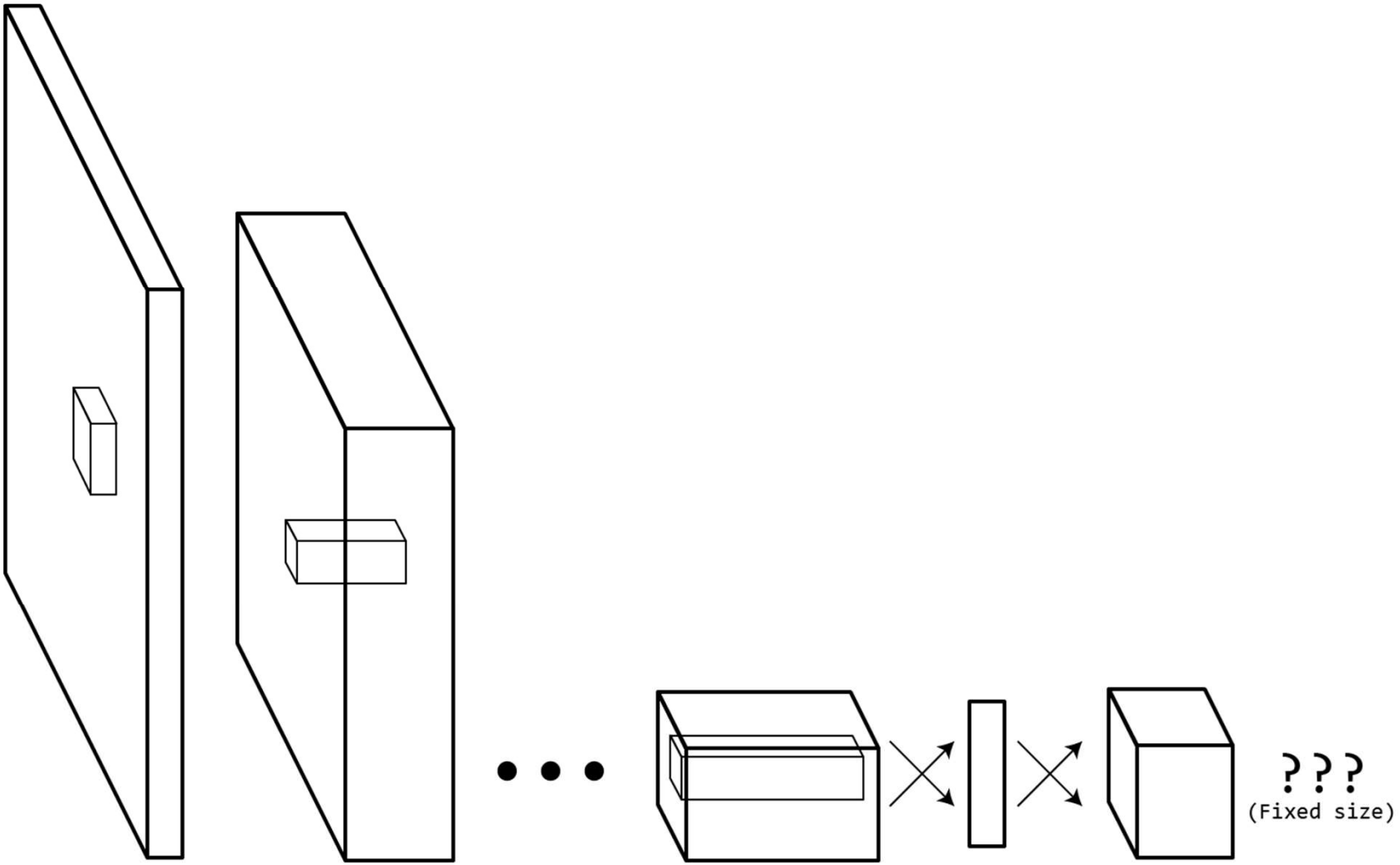
- Mean AP (mAP)
- Standard detection metric
- Sometimes at particular IOU
  - I.e. mAP@.5 or mAP@.7




1. Resize image.
2. Run convolutional network.
3. Threshold detections.
Person: 0.94
Sheep: 0.49
Dog: 0.39


Object Detection
starring
YOLOv3

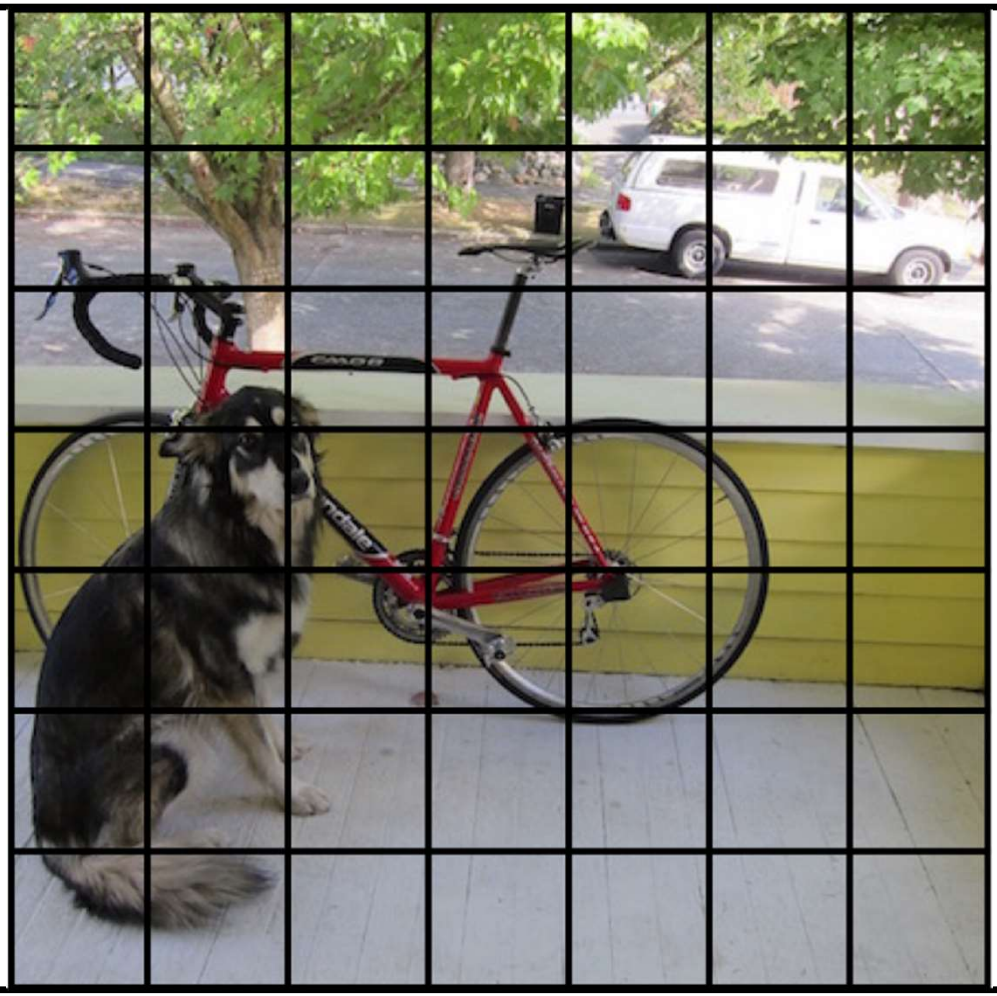
	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
YOLO	63.4	45 FPS	22 ms/img



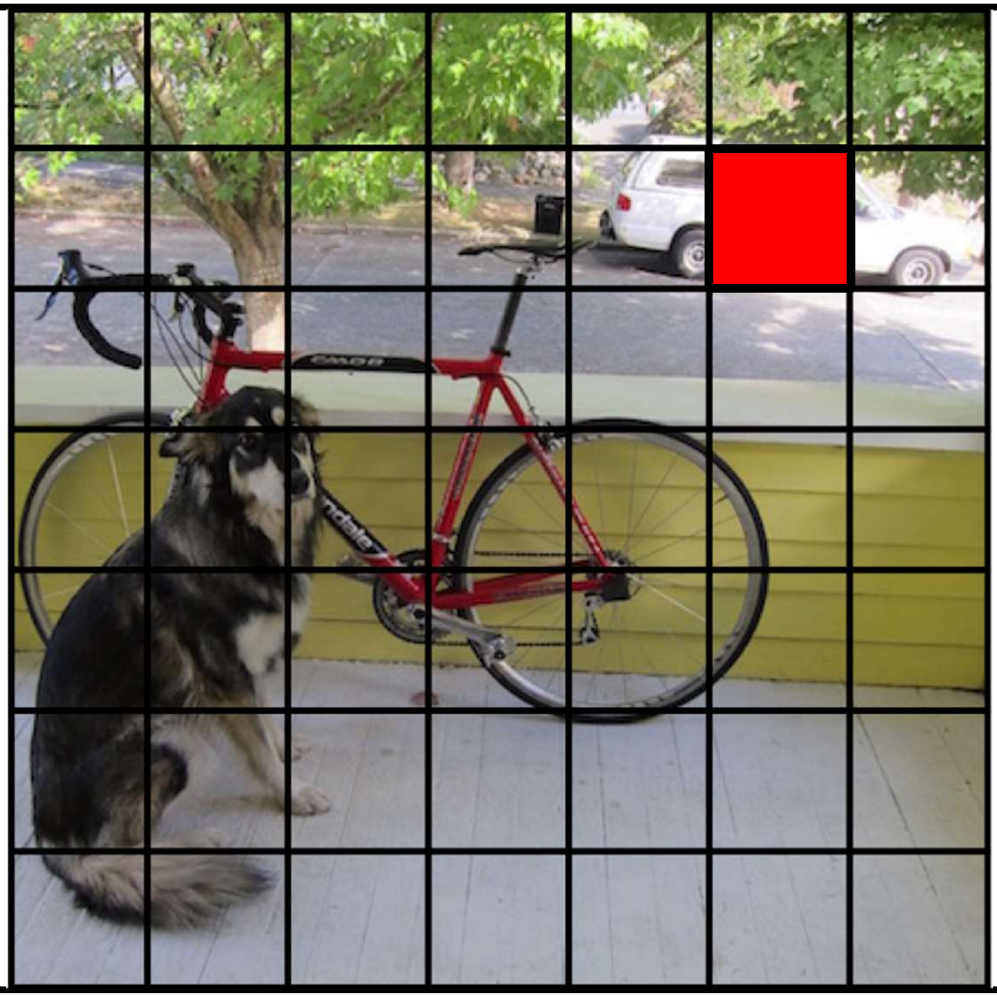
ELECTRICAL & COMPUTER ENGINEERING

Say you have an image...

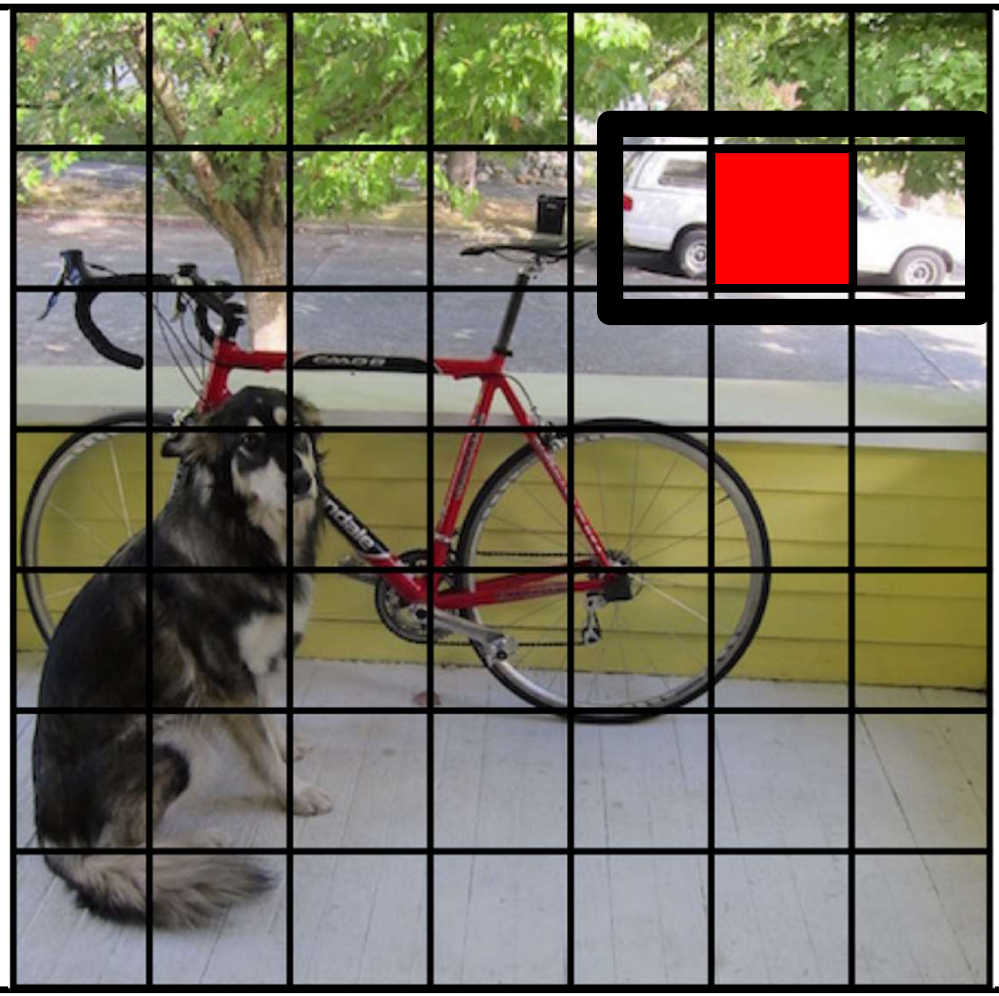
Split it into a grid



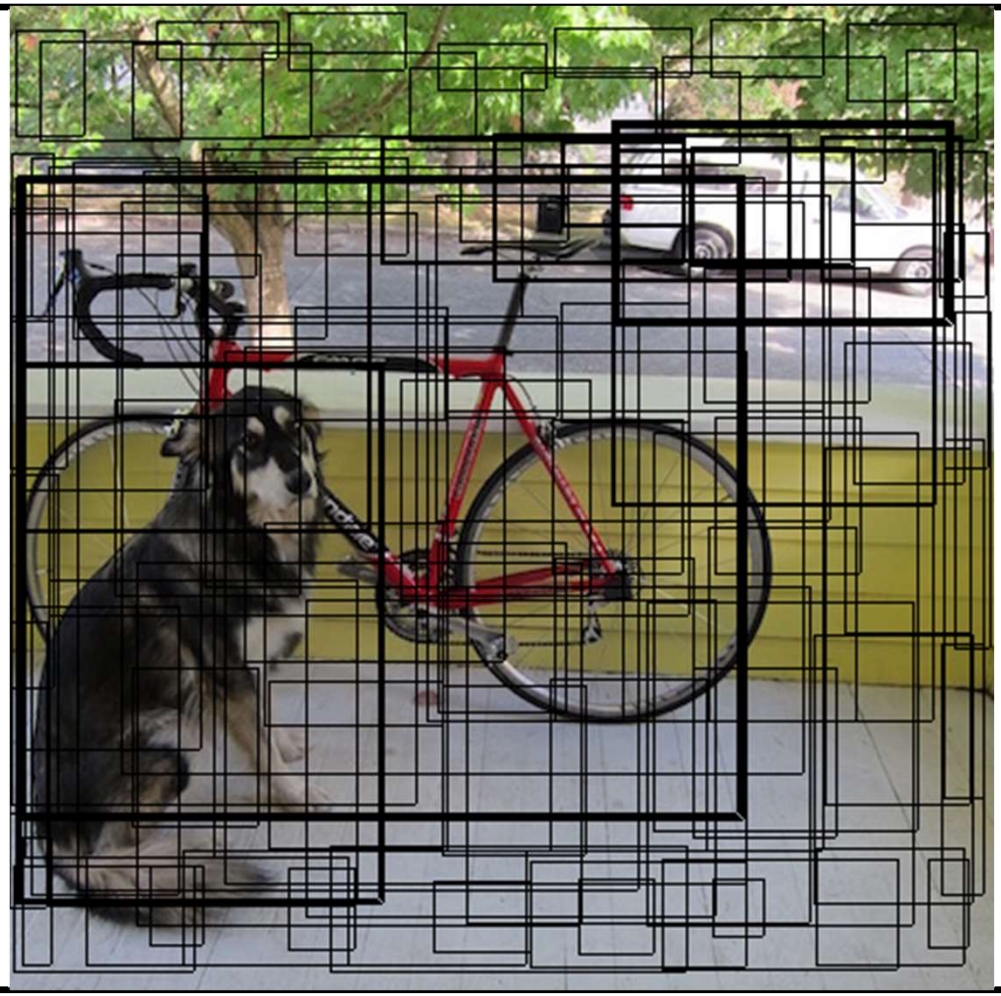
For each cell predicts P(obj)



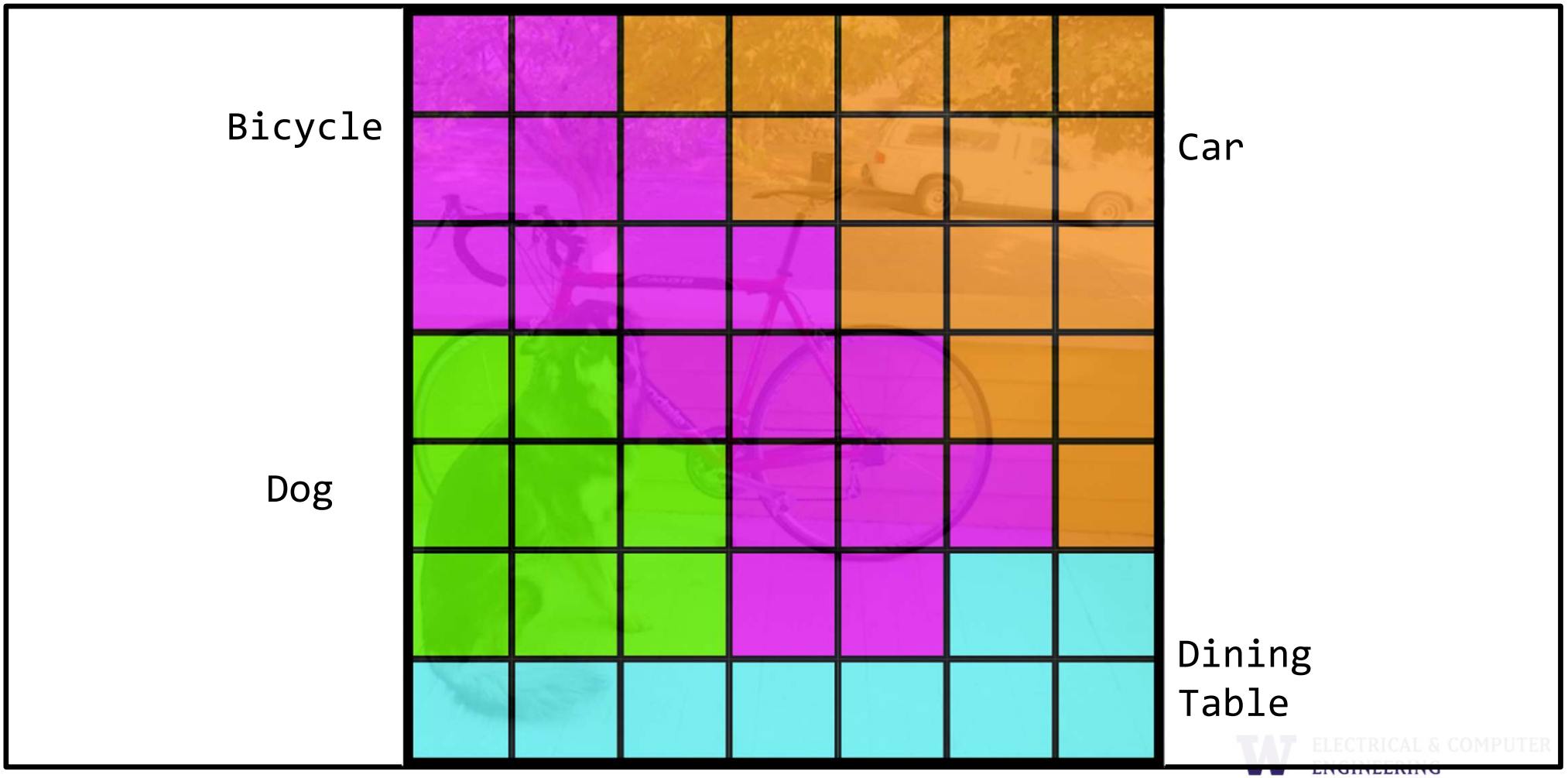
Also predict a bounding box



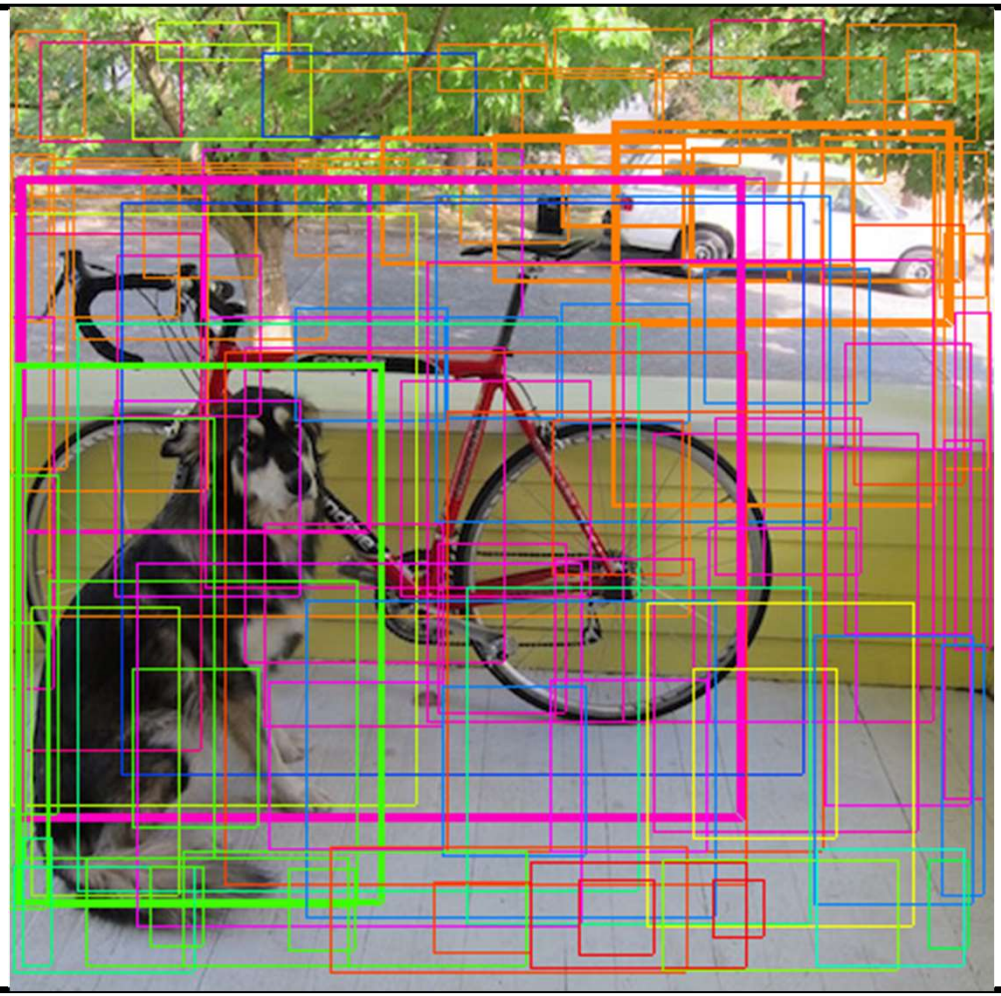
Also predict a bounding box



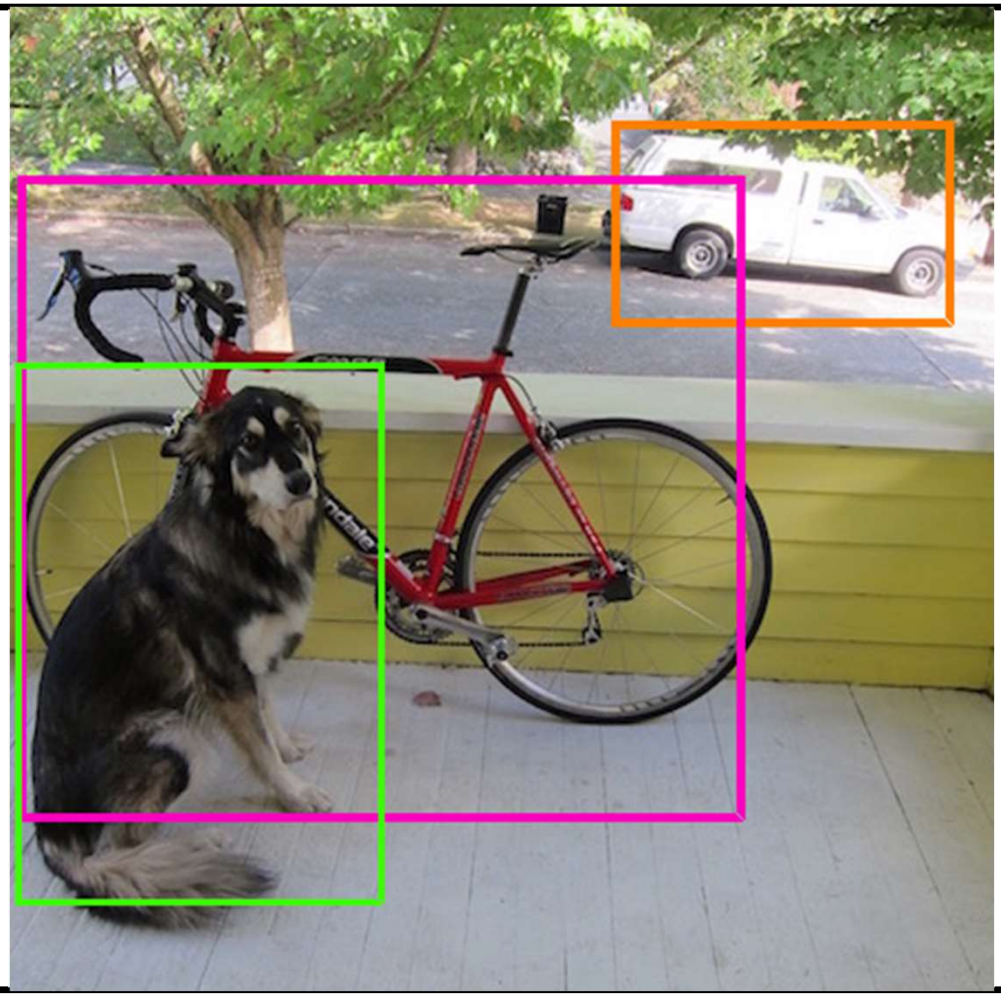
Also class probabilities



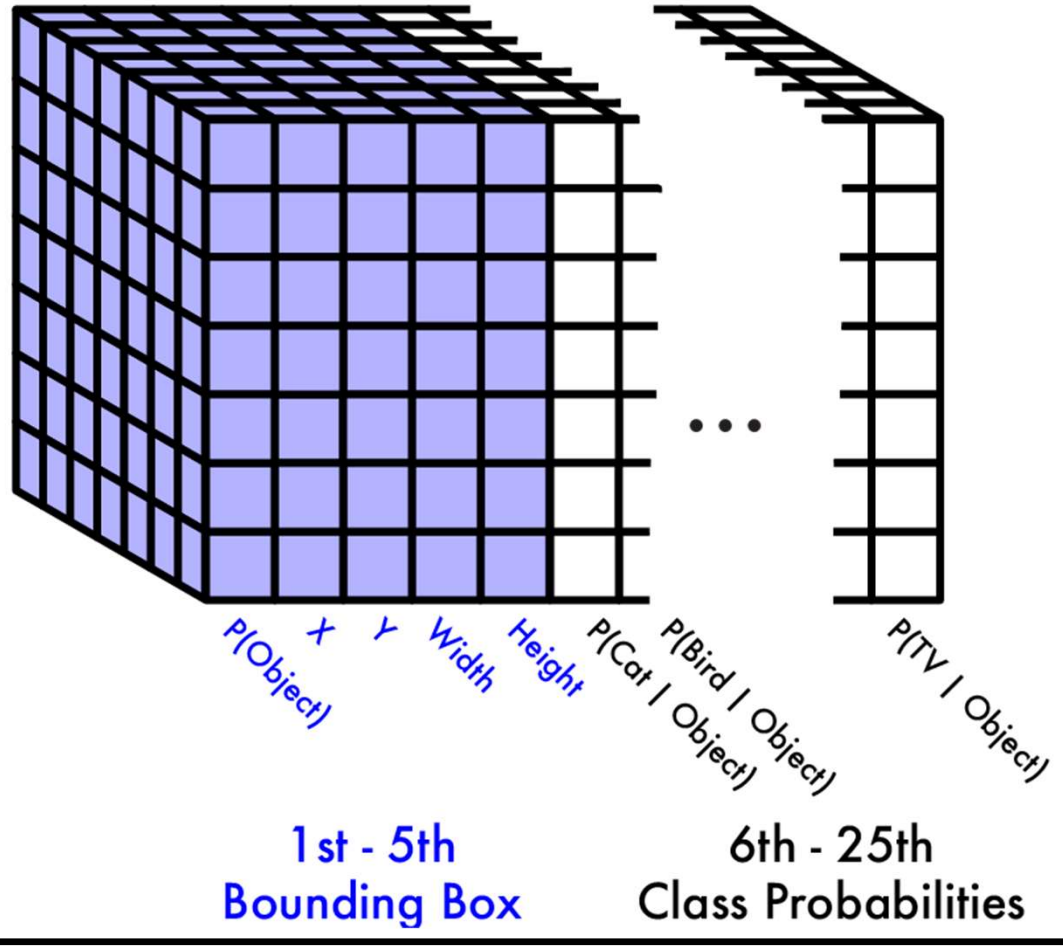
Also class probabilities



Threshold and non-max suppression

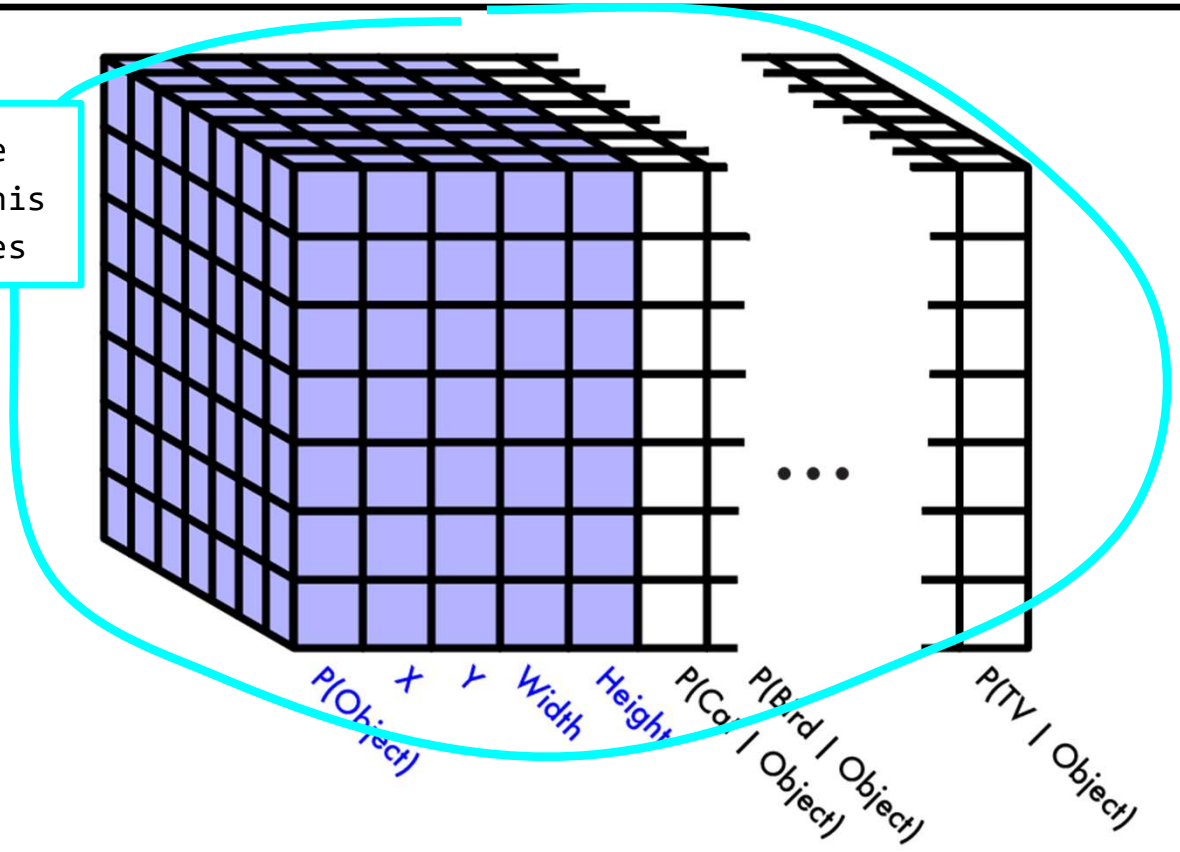


Tensor encoding detection



Tensor encoding detection

Can predict multiple boxes, just stack this tensor multiple times



Multiple bounding boxes per cell

Going to predict 5 boxes per cell

Want spatial diversity between boxes

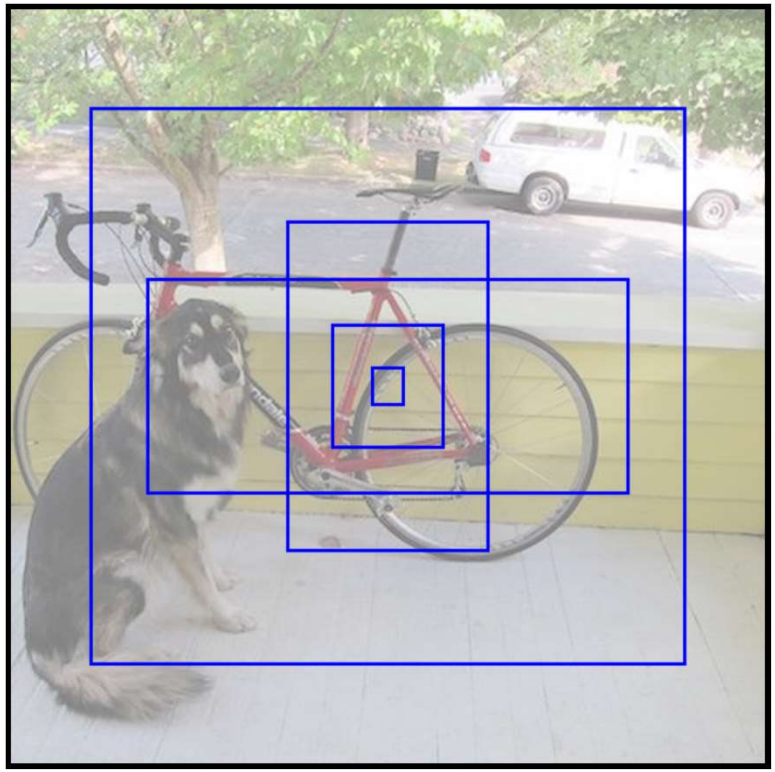
- Some small, medium, large

Don't predict boxes directly

Predict offsets from priors

Priors should be sensible

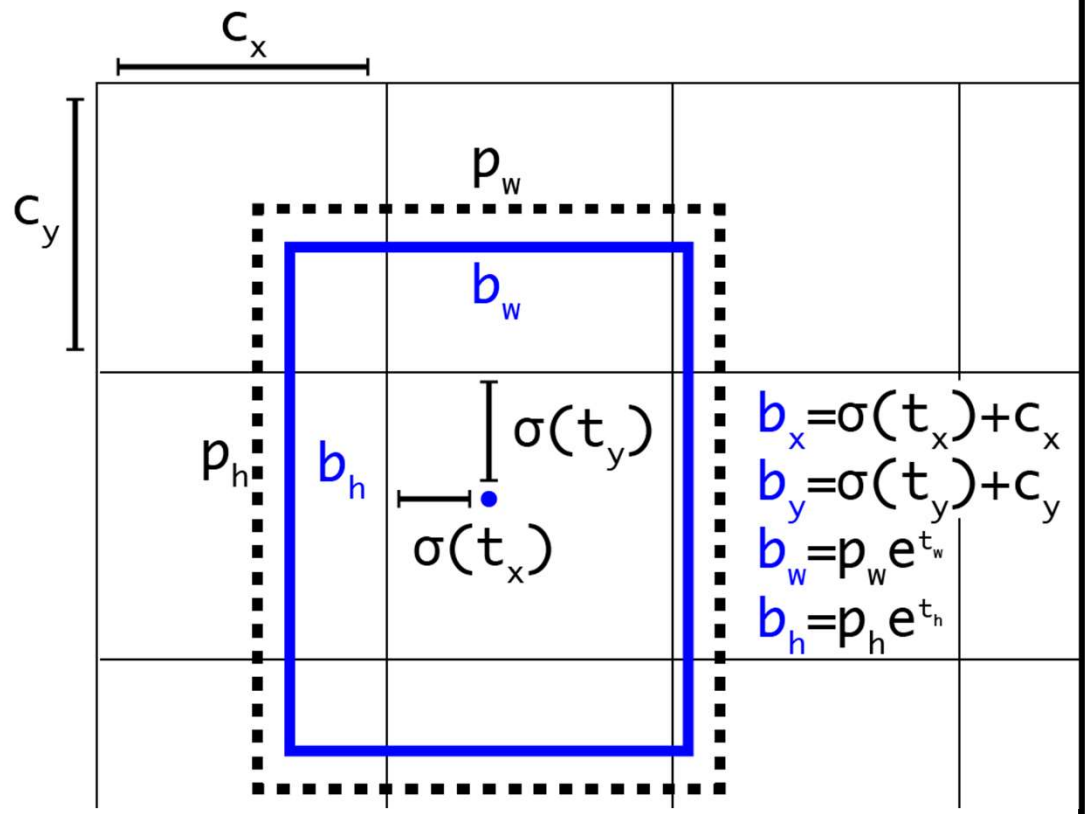
- YOLO uses k-means clustering of training data bounding box (w,h)



Bounding box encoding (YOLOv2 and v3)

Given cell offset c_x and c_y
Prior box p_w and p_h

Our predicted bounding box is given by:



YOLO loss function

L(YOLO) = α_1L(confidence) + α_2L(localization) + α_3L(classification)

Can tune all the alphas

L(confidence): binary cross-entropy

L(localization): RMSE

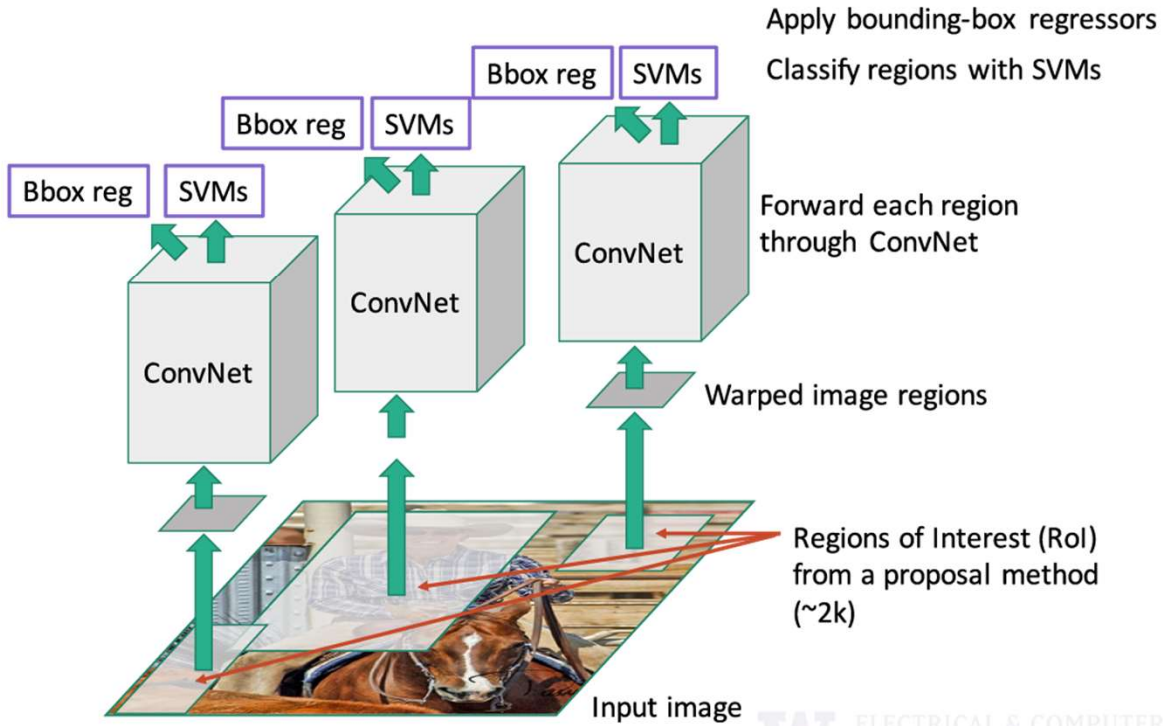
L(classification): multi-class cross-entropy or 1 v all binary cross-entropy

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
YOLO	63.4	45 FPS	22 ms/img
YOLOv2	78.6	40 FPS	25 ms/img
YOLOv3	83.1	30 FPS	33 ms/img

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
YOLO	63.4	45 FPS	22 ms/img
YOLOv2	78.6	40 FPS	25 ms/img
YOLOv3	83.1	30 FPS	33 ms/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
Resnet FRCNN	76.4	??	??
ResNet + COCO data	83.8	??	??
R-FCN	83.6	6 FPS	170 ms/img

R-CNN is slow

Run convnet independently over every ROI



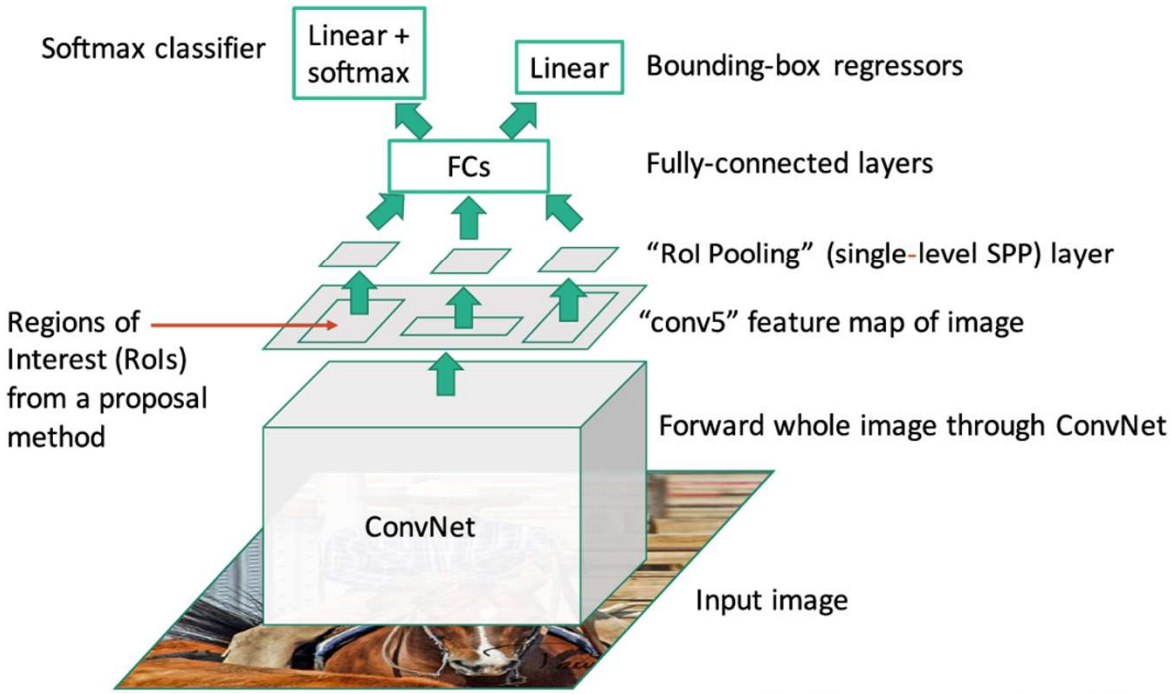
https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0

Fast R-CNN

Run convnet once, extract features using ROI pooling

ROI Pool:

Convert variable sized
ROI to fixed size output



https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0

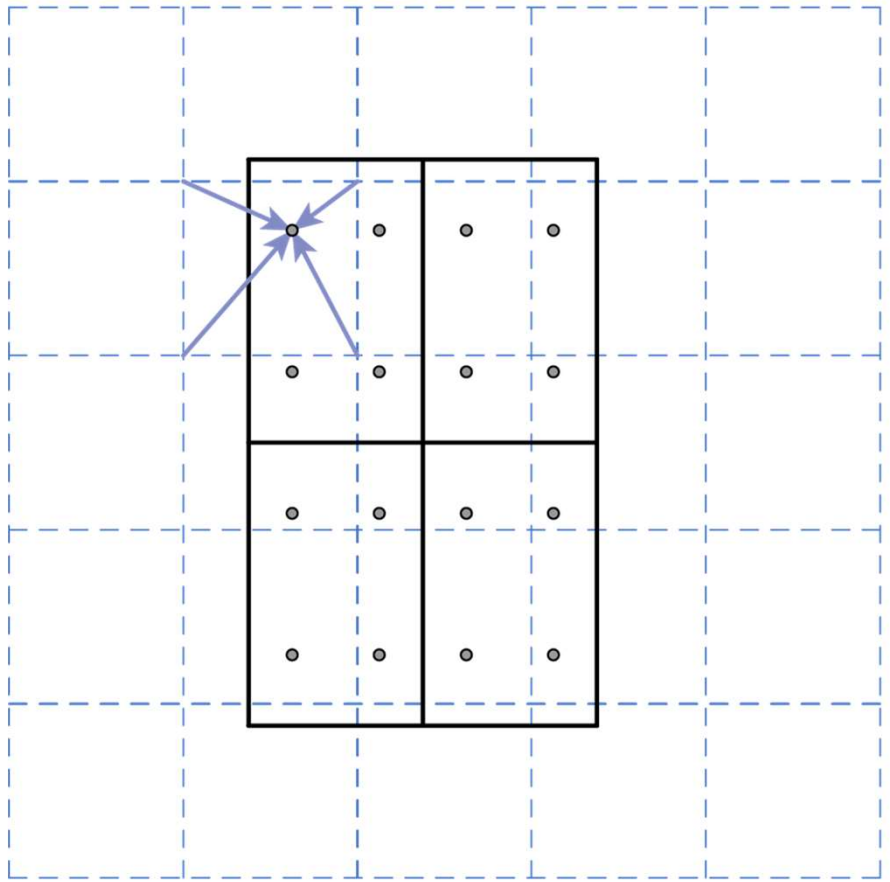
ROI Align

Better than ROI Pool so we'll talk about it instead

Split ROI into fixed size (say 2x2)

Sample image at multiple points for each cell in ROI (bilinear interp.)

Pool over these samples (max, avg…)



https://arxiv.org/pdf/1703.06870.pdf

Fast R-CNN

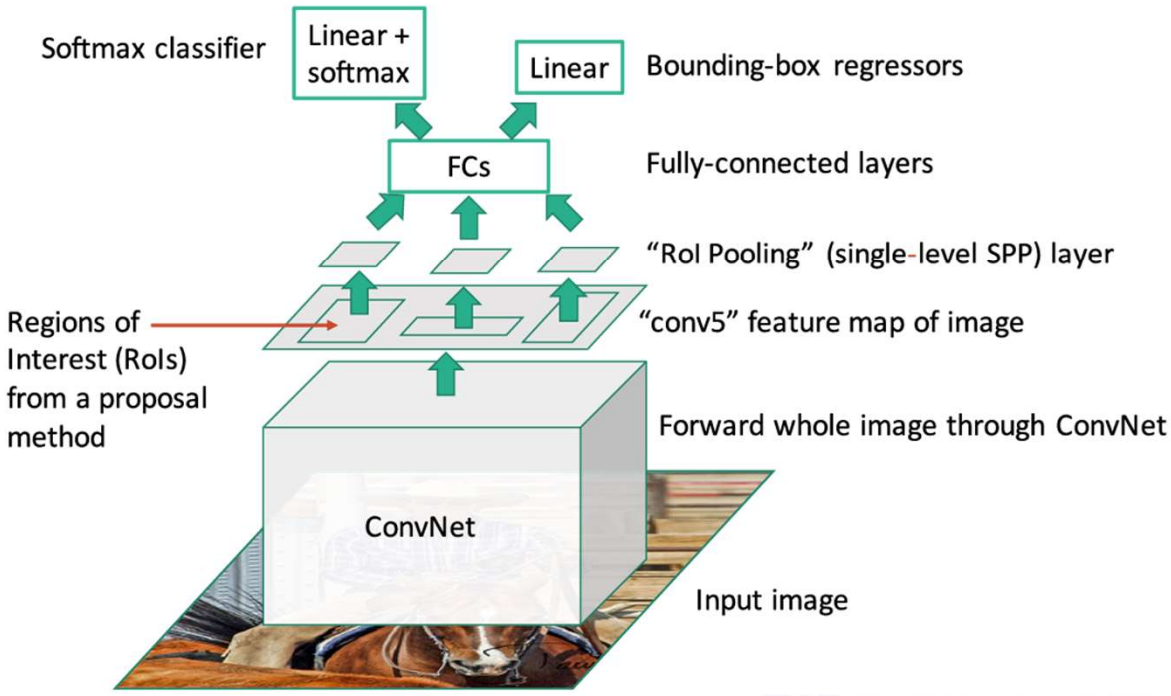
Run convnet once, extract features using ROI pooling

ROI Pool:

Convert variable sized
ROI to fixed size output

Much faster, no independent network evals (except last linear layer)

Still slow region proposer, selective search takes ~2 sec



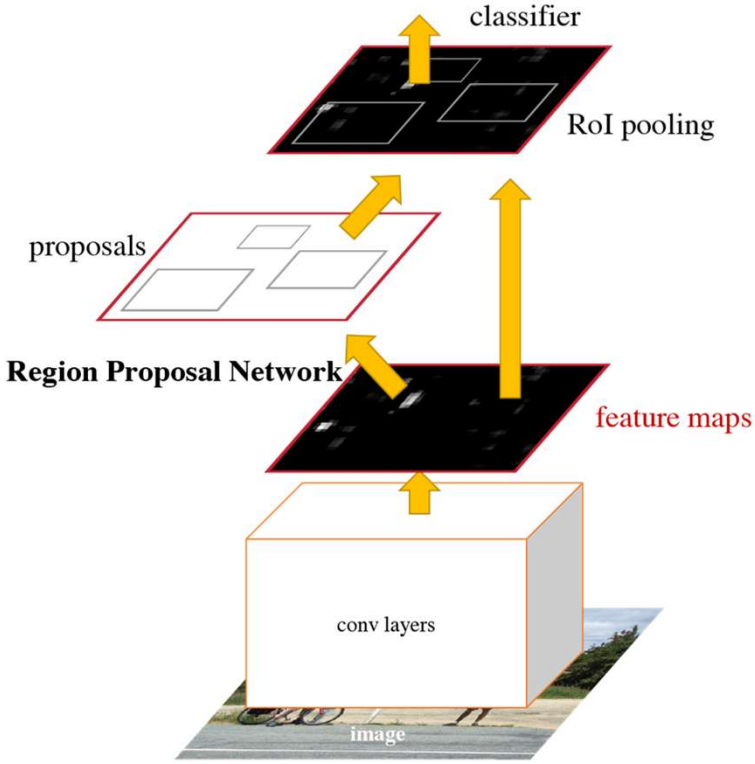
https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0

Faster R-CNN

Use Convnet to propose regions *and* generate features

ROI Pool to fix size of ROI features

Additional layers to classify and predict bbox for ROIs



https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0

SIAMESE Network for Multiple-Object Tracking