CSCE 5218 & 4930
Deep Learning

Convolutional Neural Networks

Slides adapted from Adriana Kovashka

Plan for this lecture

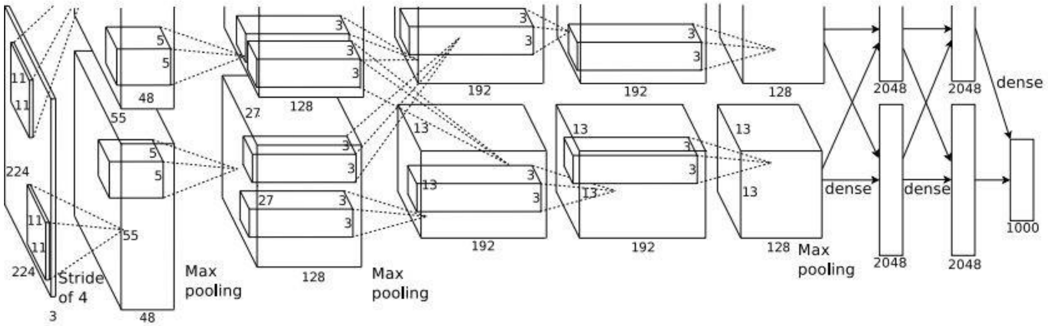
- Motivation: Scanning for patterns
- Convolutional network operations
- Common architectures
- Visualizing convolutional networks
- Applications in computer vision

Plan for this lecture

- Motivation: Scanning for patterns
- Convolutional network operations
- Common architectures
- Visualizing convolutional networks
- Applications in computer vision

Case Study: AlexNet

[Krizhevsky et al. 2012]

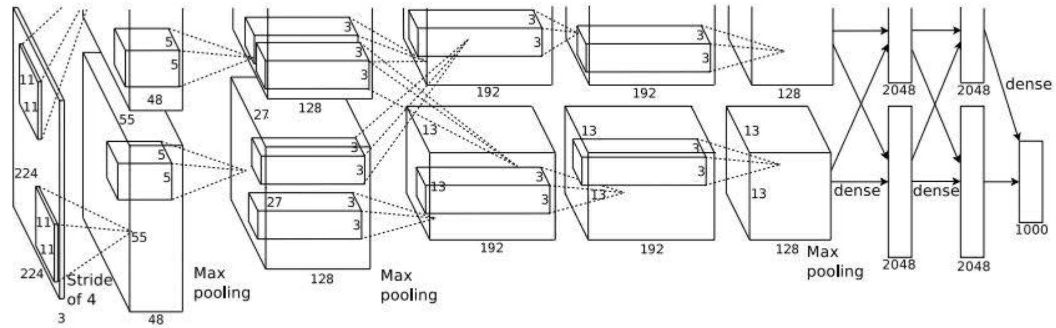


Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

Case Study: AlexNet

[Krizhevsky et al. 2012]

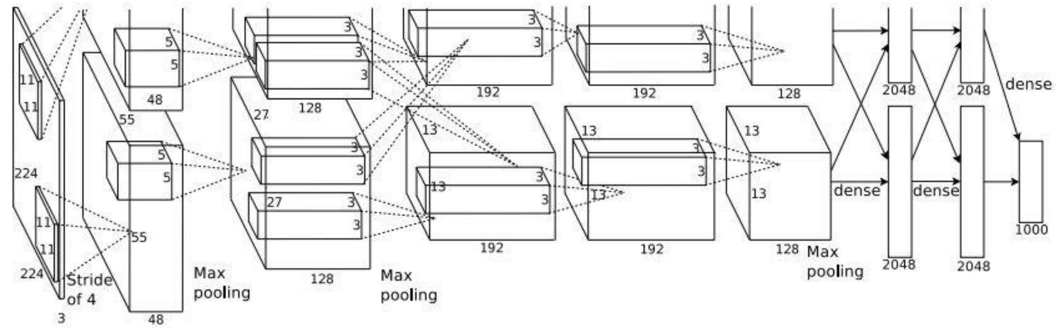


Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

Case Study: AlexNet

[Krizhevsky et al. 2012]



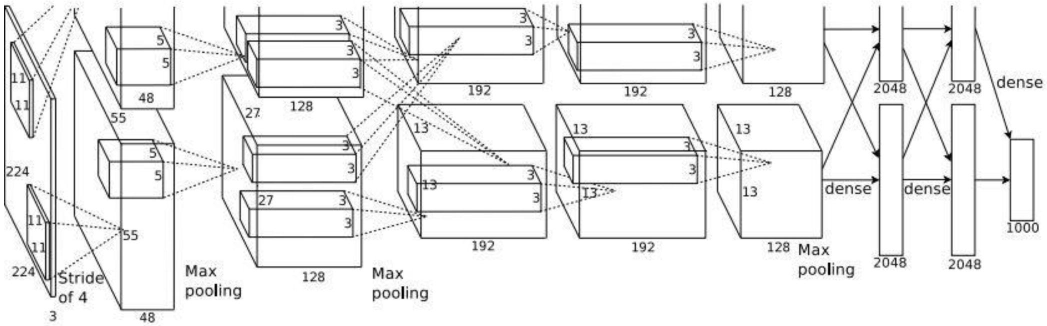
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2 [27x27x96]
NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer [13x13x384]
CONV3: 384 3x3 filters at stride 1, pad 1 [13x13x384]
CONV4: 384 3x3 filters at stride 1, pad 1 [13x13x256]
CONV5: 256 3x3 filters at stride 1, pad 1 [6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons
(class scores)

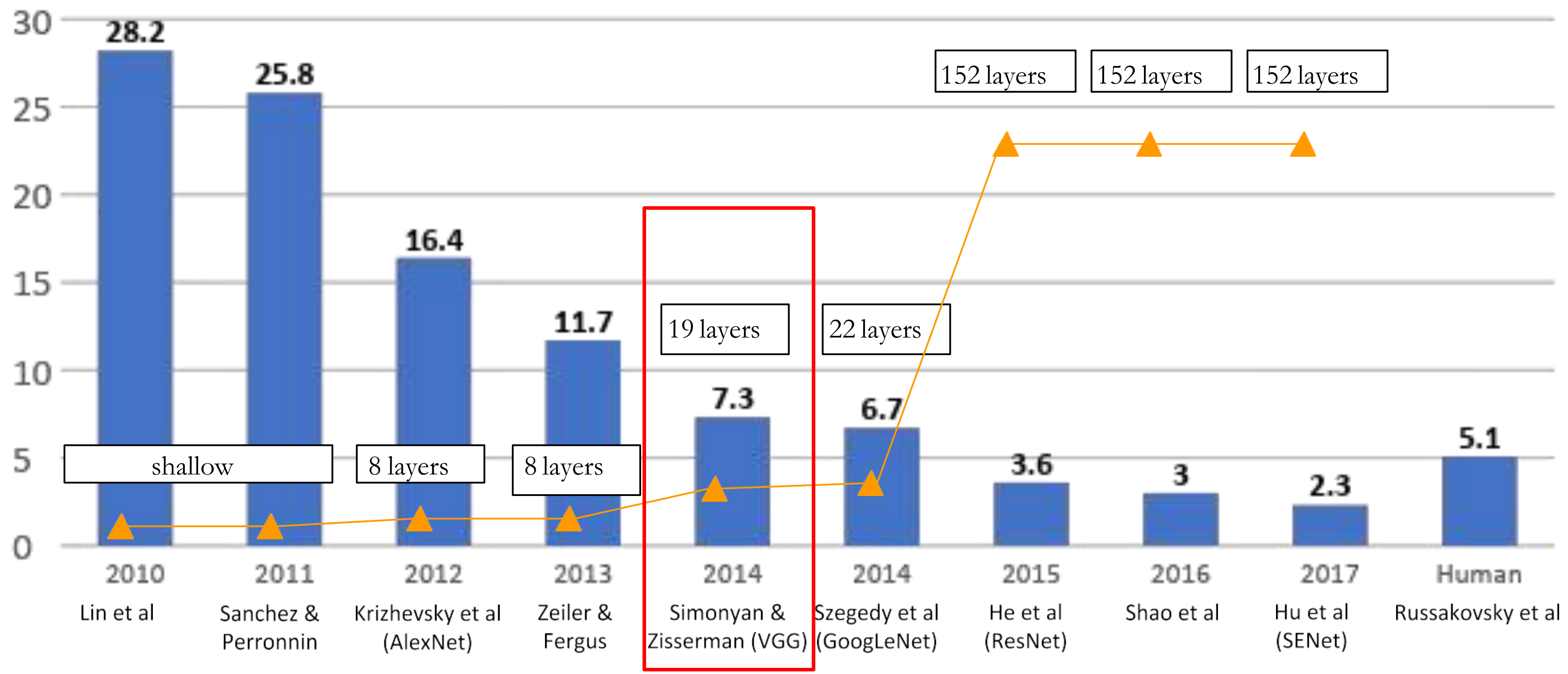
Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4

3.3 Local Response Normalization

ReLUs have the desirable property that they do not require input normalization to prevent them from saturating. If at least some training examples produce a positive input to a ReLU, learning will happen in that neuron. However, we still find that the following local normalization scheme aids generalization. Denoting by $a^i_{x,y}$ the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity, the response-normalized activity $b^i_{x,y}$ is given by the expression

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Case Study: VGGNet

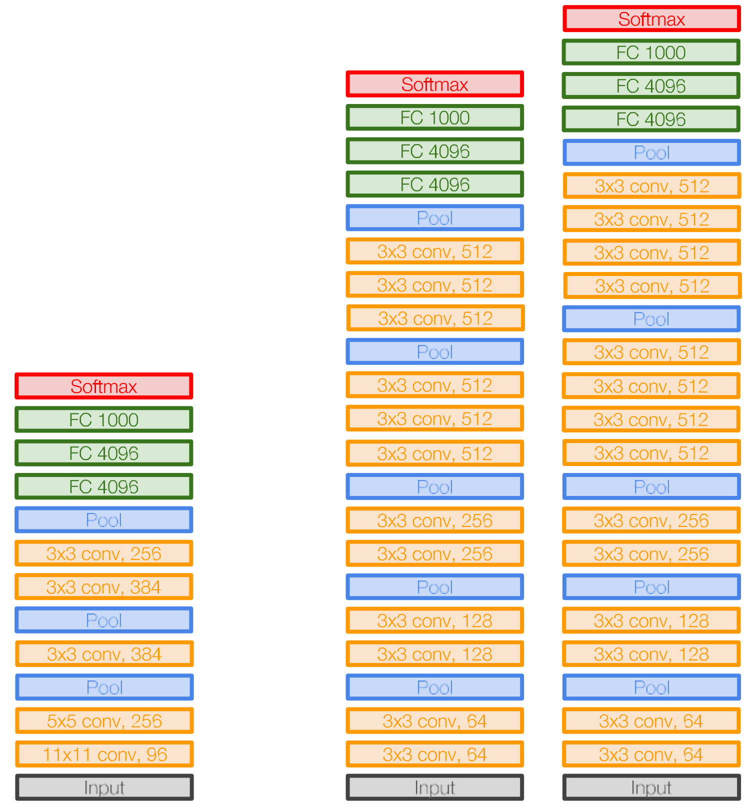
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



AlexNet VGG16 VGG19

Case Study: VGGNet

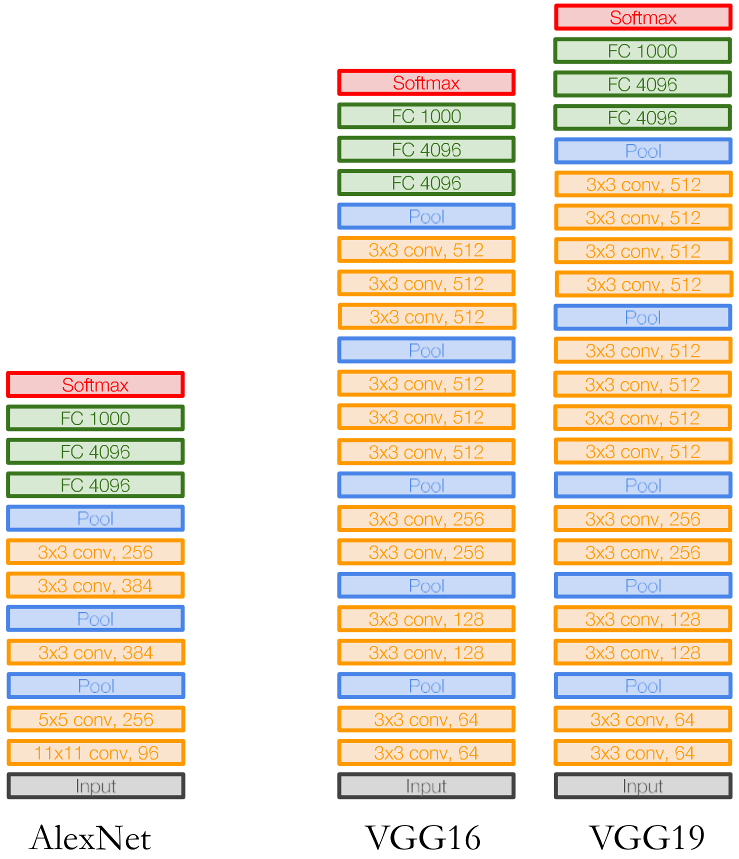
[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

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



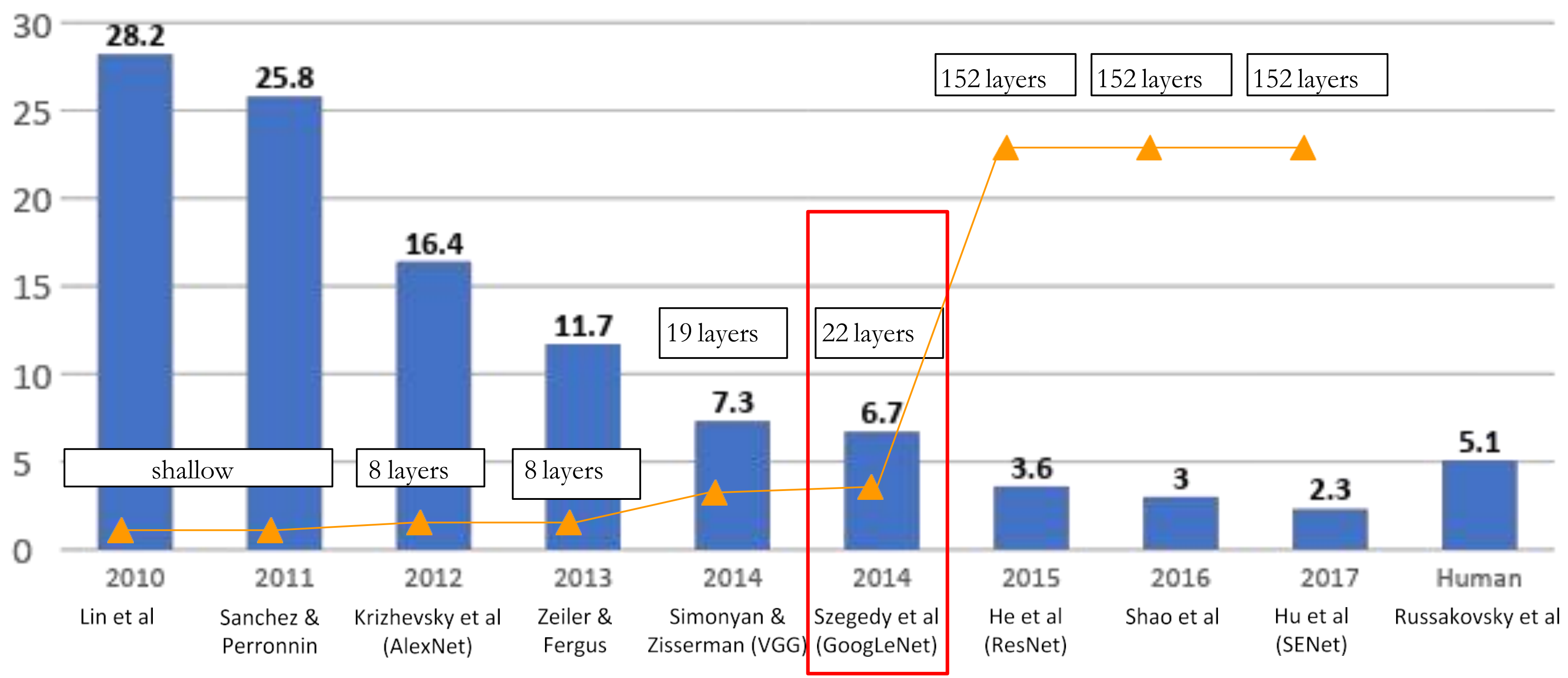
Case Study: VGGNet

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912 CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824 CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648 CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296 CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296 CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296 CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448 FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters

VGG16

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

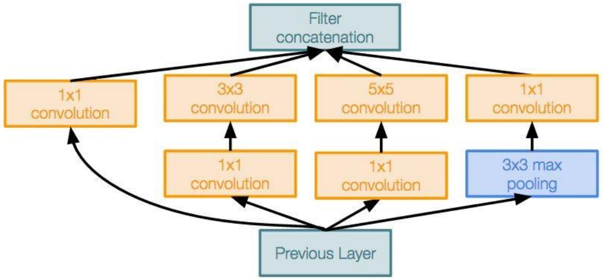


Case Study: GoogLeNet

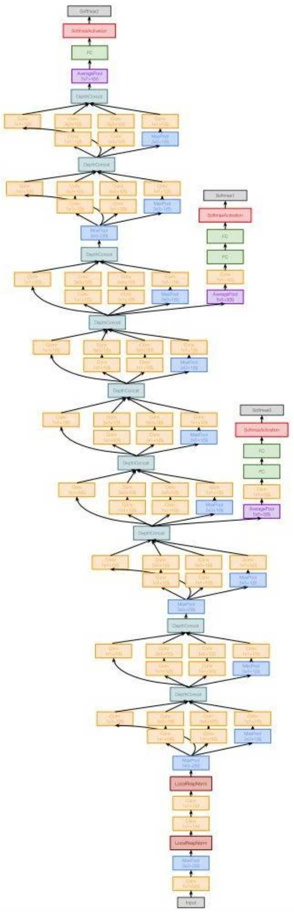
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient "Inception" module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC'14 classification winner
  (6.7% top 5 error)



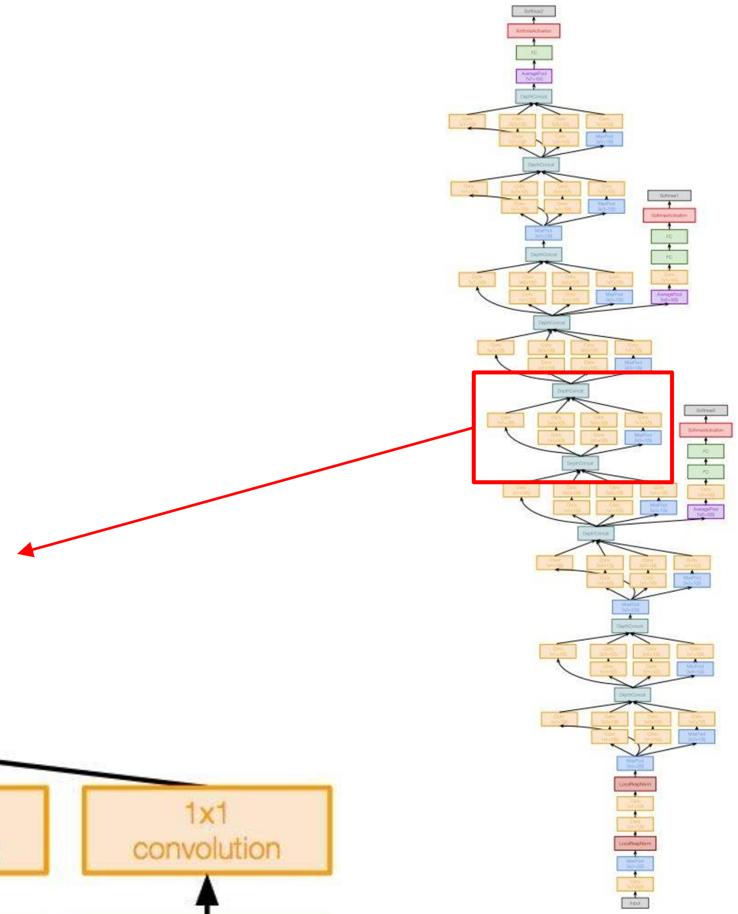
Inception module



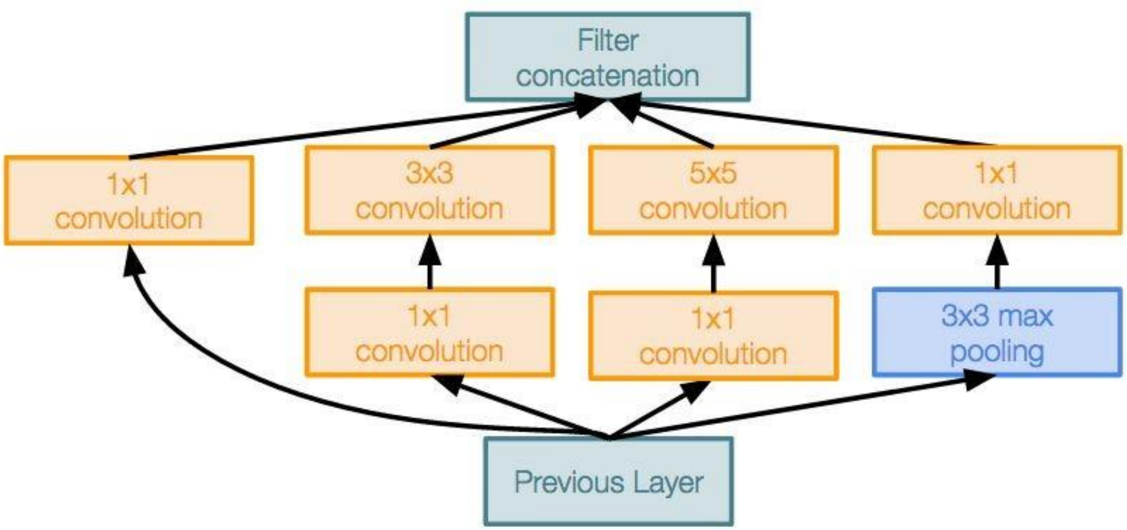
Case Study: GoogLeNet

[Szegedy et al., 2014]

“Inception module”: design a good local network topology (network within a network) and then stack these modules on top of each other

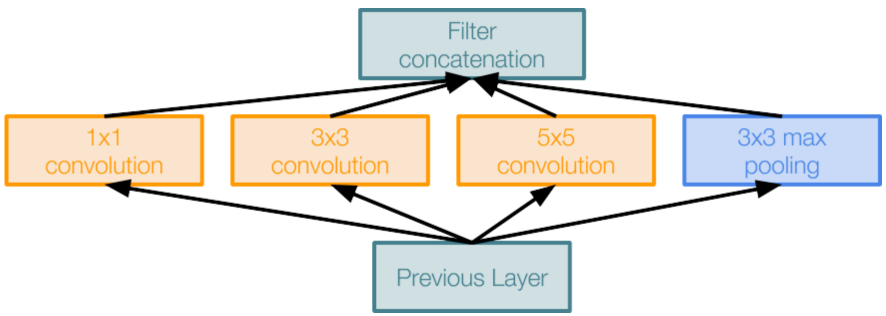


Inception module



Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

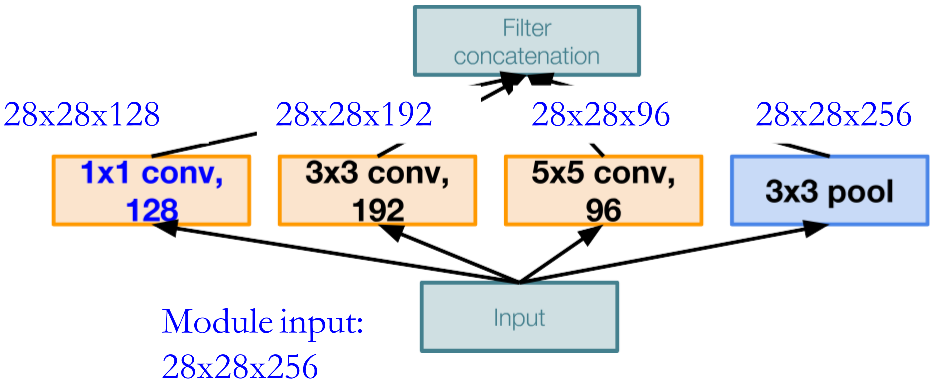
Concatenate all filter outputs together depth-wise

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?



Naive Inception module

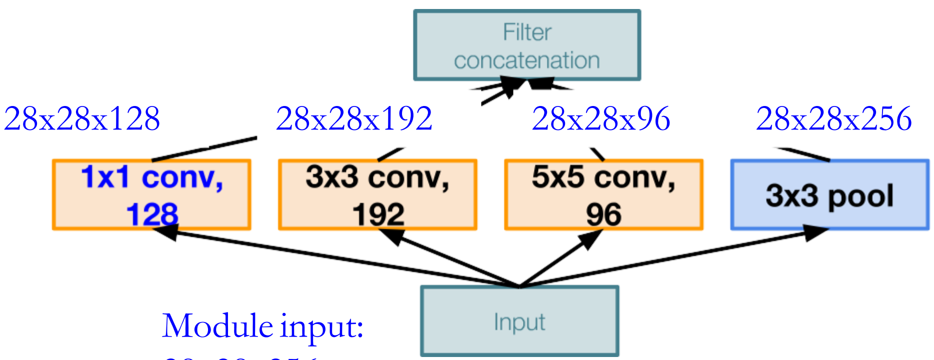
Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?

28x28x(128+192+96+256) = 28x28x672



Module input: 28x28x256

Naive Inception module

Case Study: GoogLeNet

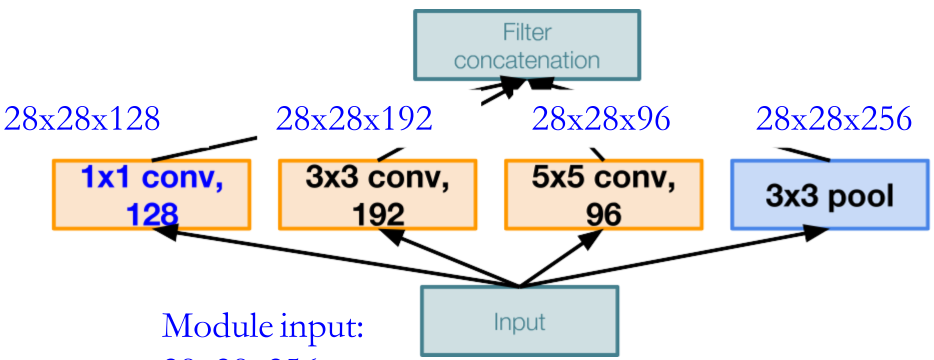
[Szegedy et al., 2014]

Q: What is the problem with this?

Example:

Q3: What is output size after filter concatenation?

28x28x(128+192+96+256) = 28x28x672



Module input: 28x28x256

Naive Inception module

Case Study: GoogLeNet

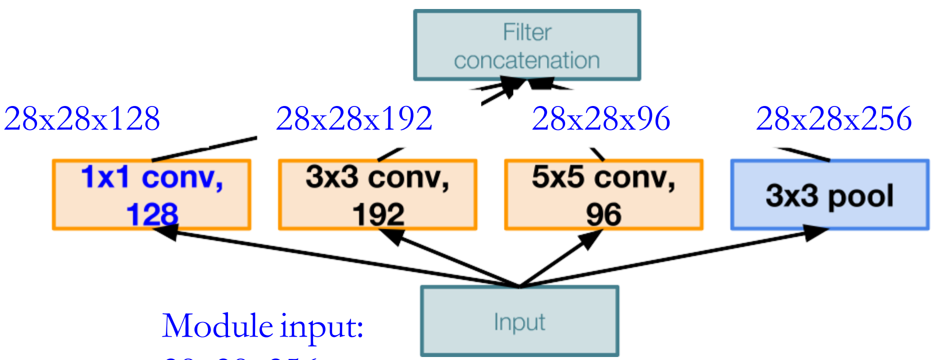
[Szegedy et al., 2014]

Q: What is the problem with this?

Example:

Q3: What is output size after filter concatenation?

28x28x(128+192+96+256) = 28x28x672



Module input: 28x28x256

Naive Inception module

Conv Ops:

[1x1 conv, 128] 28x28x128x1x1x256 [3x3 conv, 192] 28x28x192x3x3x256 [5x5 conv, 96] 28x28x96x5x5x256 **Total: 854M ops**

Very expensive to compute

Pooling layer preserves feature depth, which means total depth after concatenation can only grow at every layer!

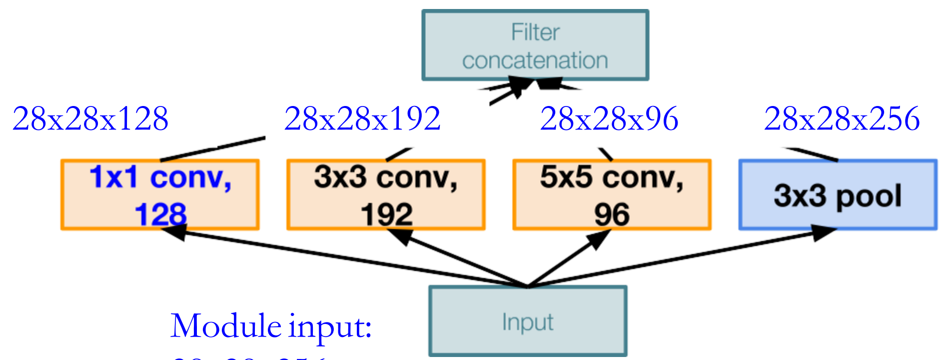
Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?

28x28x(128+192+96+256) = 28x28x672



Module input: 28x28x256

Naive Inception module

Q: What is the problem with this?

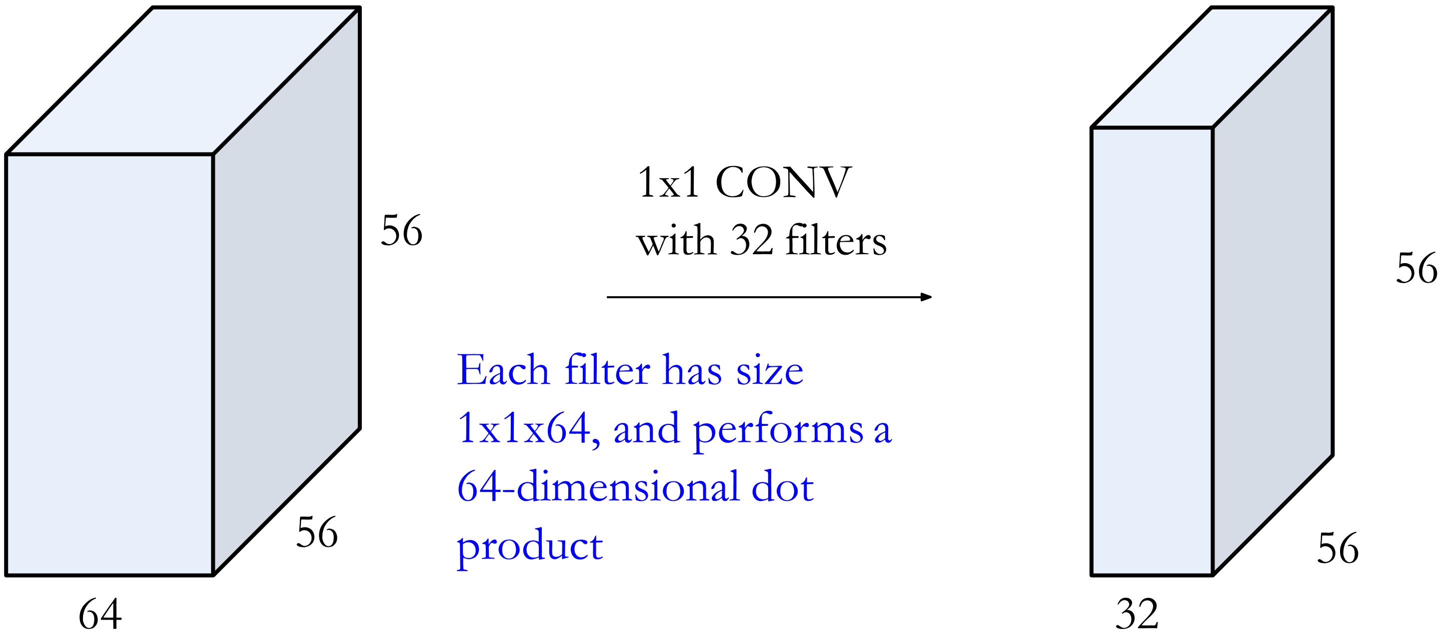
Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256 [3x3 conv, 192] 28x28x192x3x3x256 [5x5 conv, 96] 28x28x96x5x5x256 **Total: 854M ops**

Very expensive to compute

Pooling layer preserves feature depth, which means total depth after concatenation can only grow at every layer!

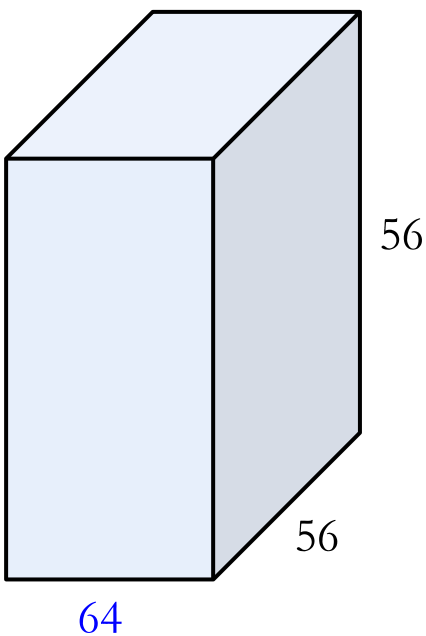
Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth

1x1 convolutions



Each filter has size 1x1x64, and performs a 64-dimensional dot product

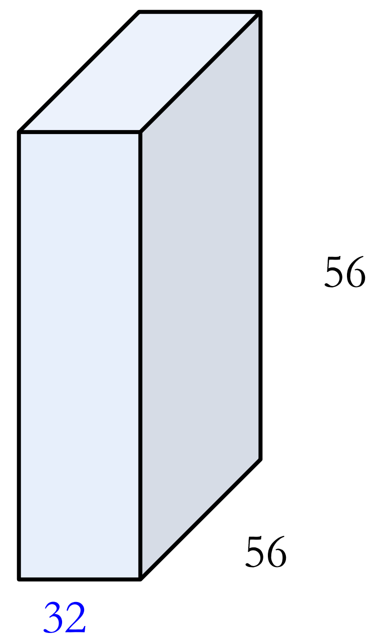
1x1 convolutions



1x1 CONV
with 32 filters

Preserves spatial dimensions, reduces depth!

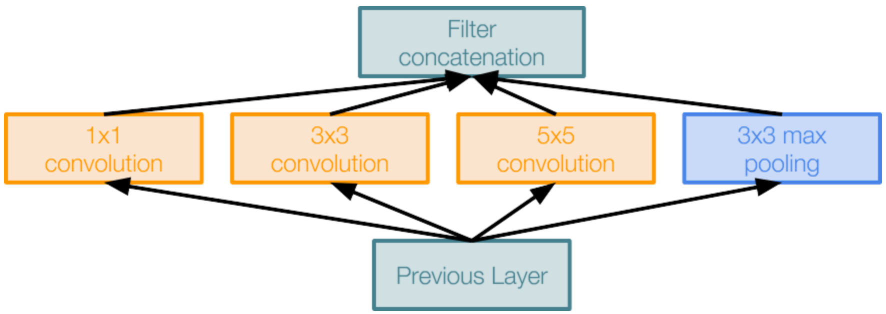
Projects depth to lower dimension (combination of feature maps)



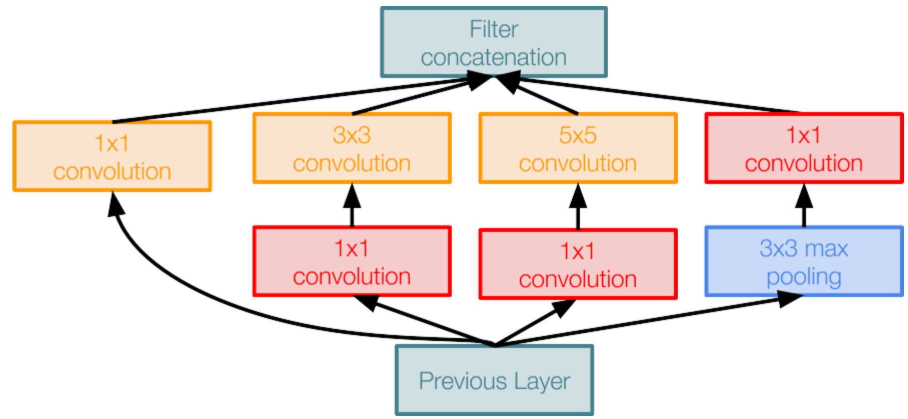
Case Study: GoogLeNet

[Szegedy et al., 2014]

1x1 conv "bottleneck" layers



Naive Inception module



Inception module with dimension reduction

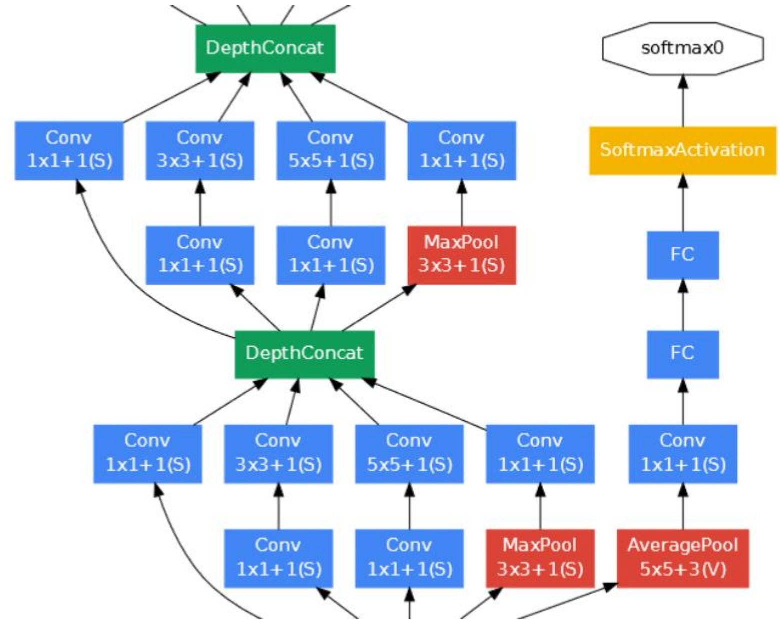
Total: 854M ops

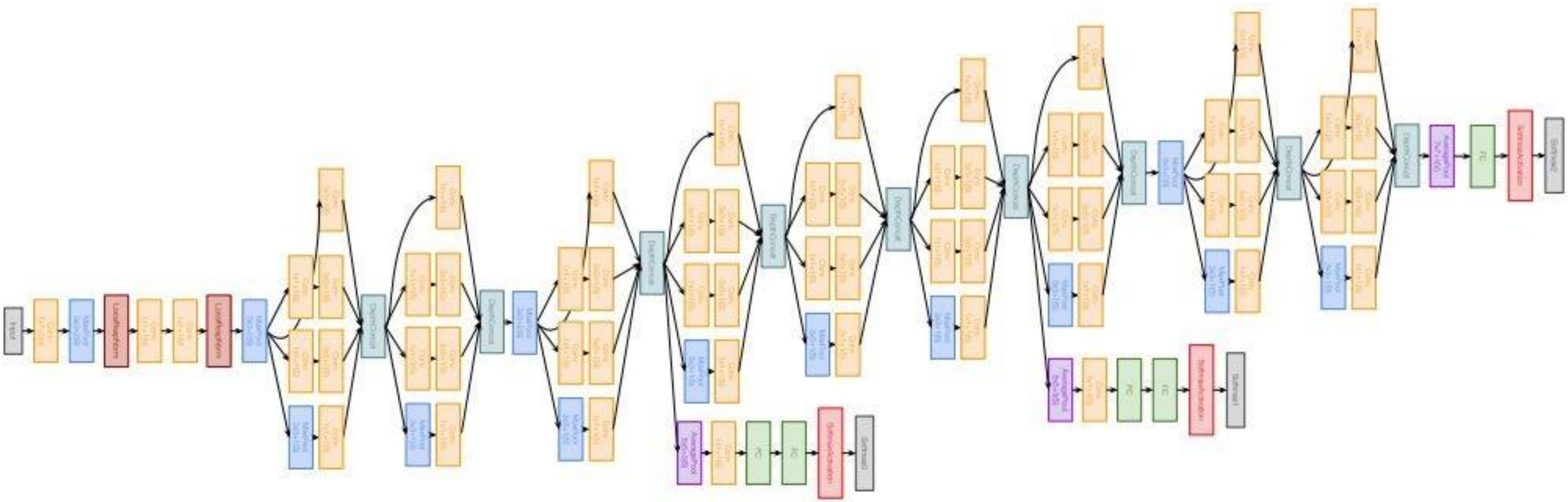
Total: 358M ops

Case Study: GoogLeNet

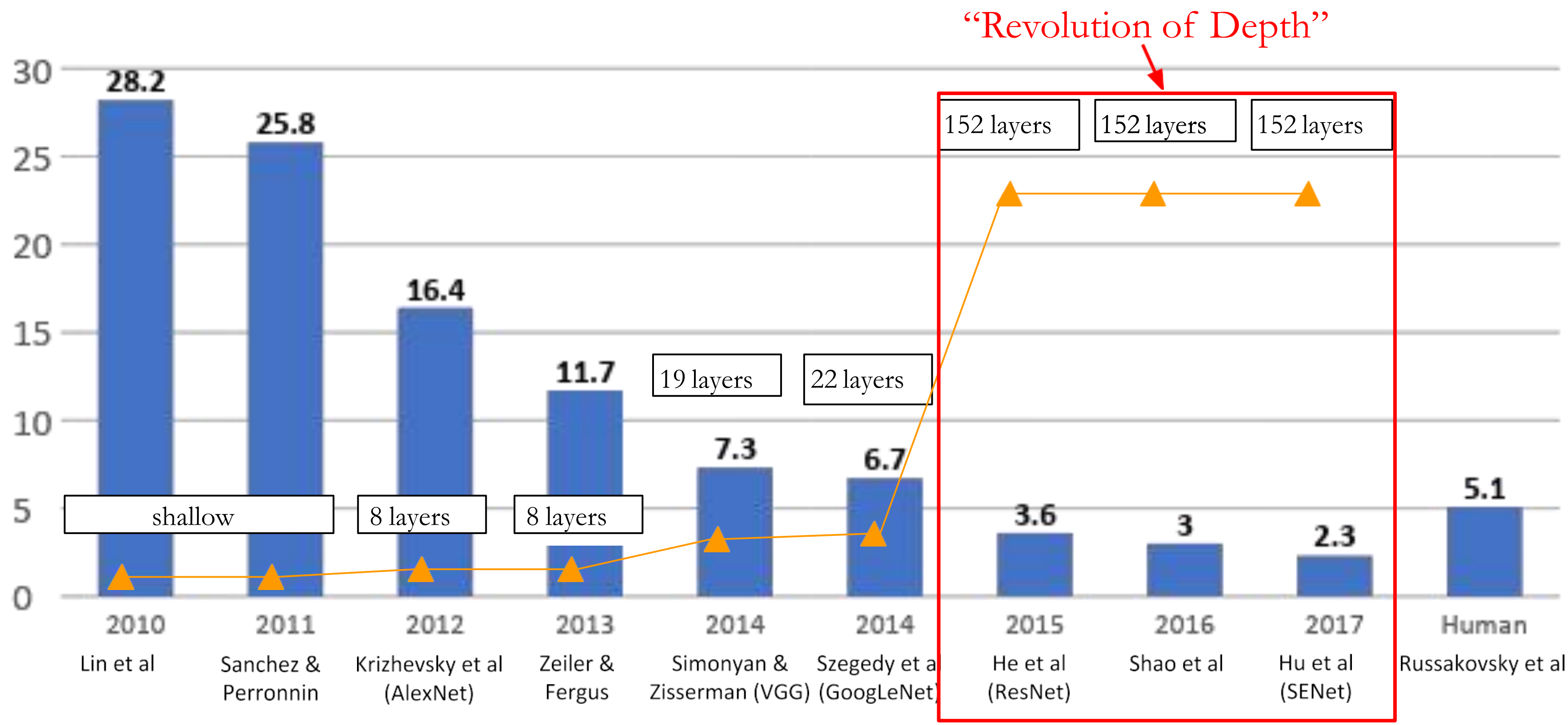
[Szegedy et al., 2014]

Full GoogLeNet architecture





ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

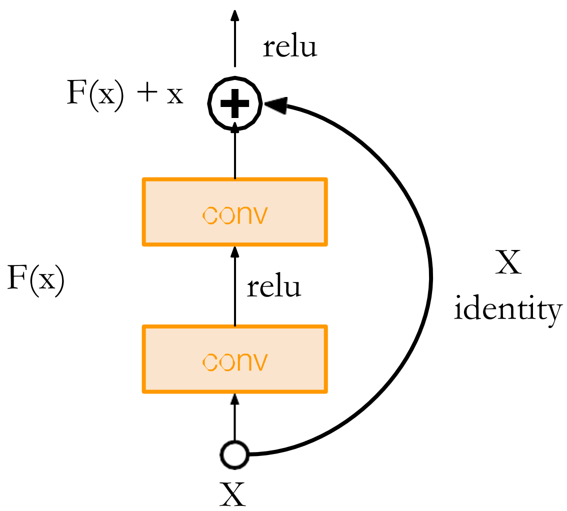


Case Study: ResNet

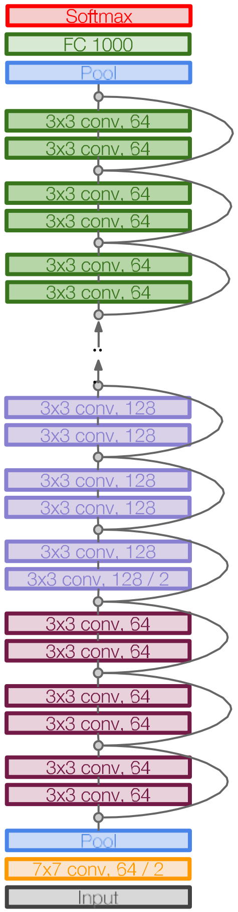
[He et al., 2016]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



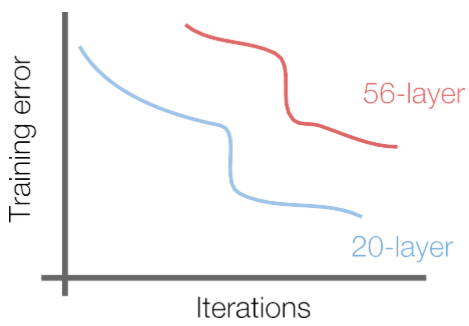
Residual block

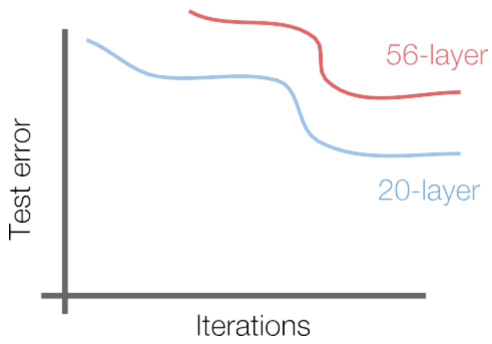


Case Study: ResNet

[He et al., 2016]

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?





Q: What's strange about these training and test curves? [Hint: look at the order of the curves]

56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it's not caused by overfitting!

Case Study: ResNet

[He et al., 2016]

Hypothesis:
The problem is an *optimization* problem, deeper models are harder to optimize

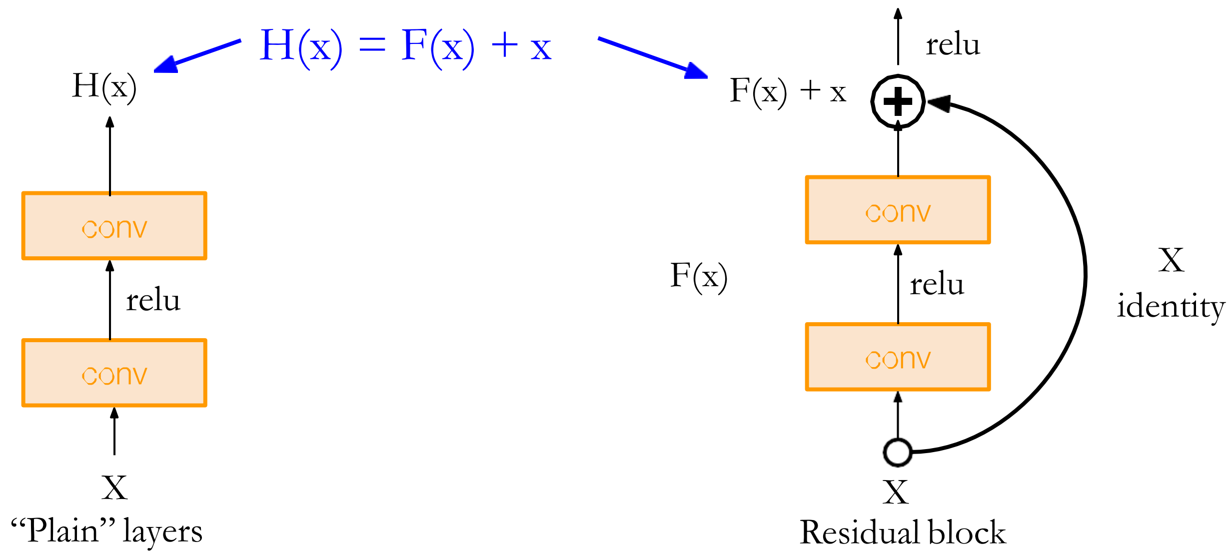
The deeper model should be able to perform at least as well as the shallower model.

A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

Case Study: ResNet

[He et al., 2016]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



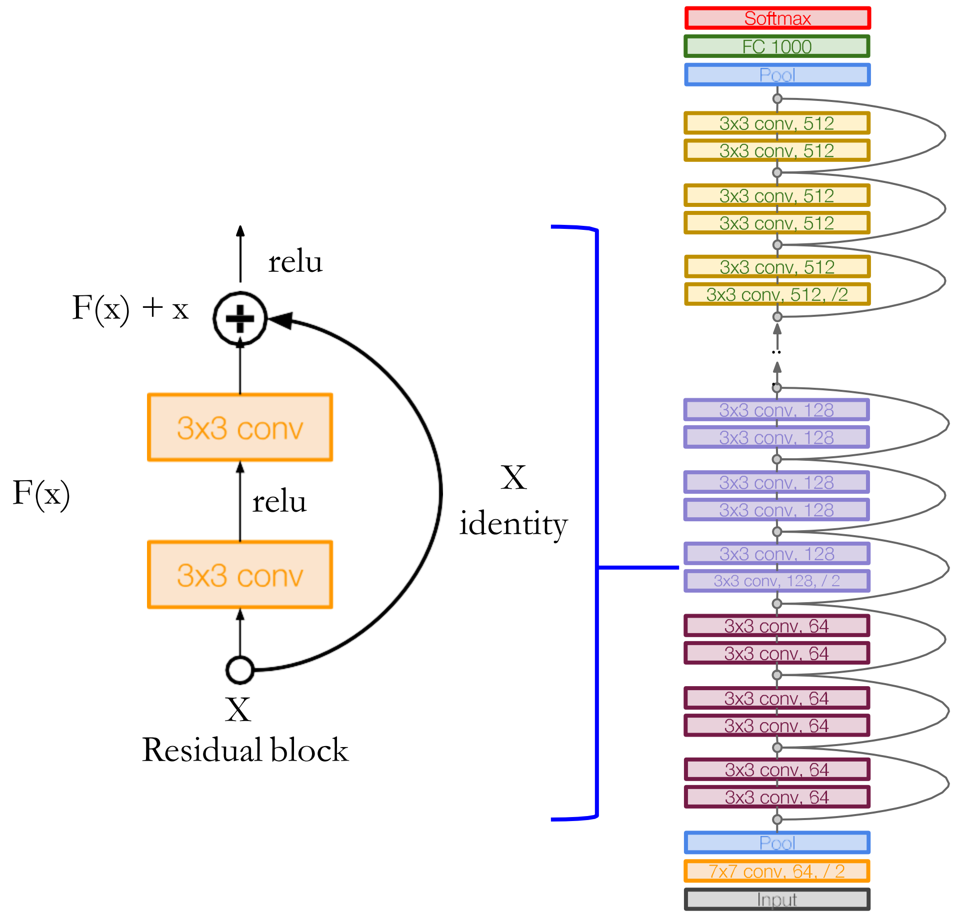
Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

Case Study: ResNet

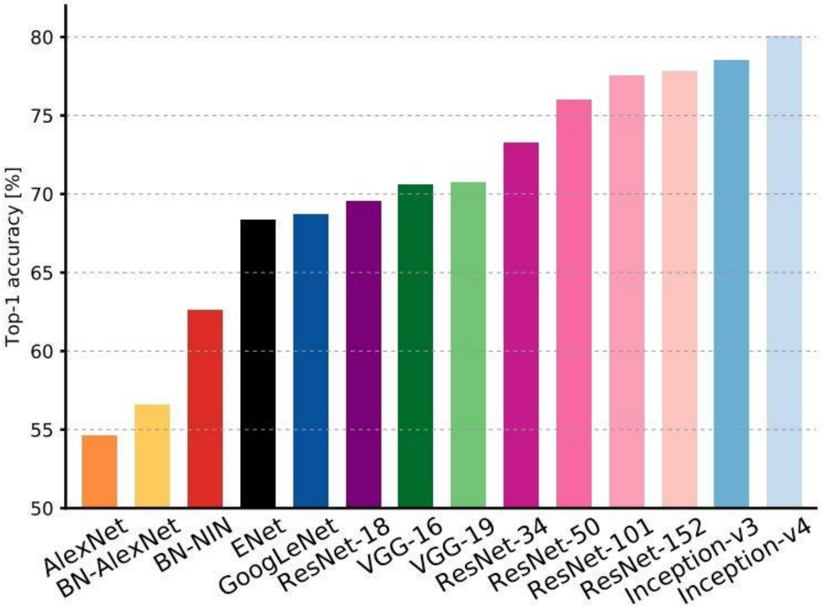
[He et al., 2016]

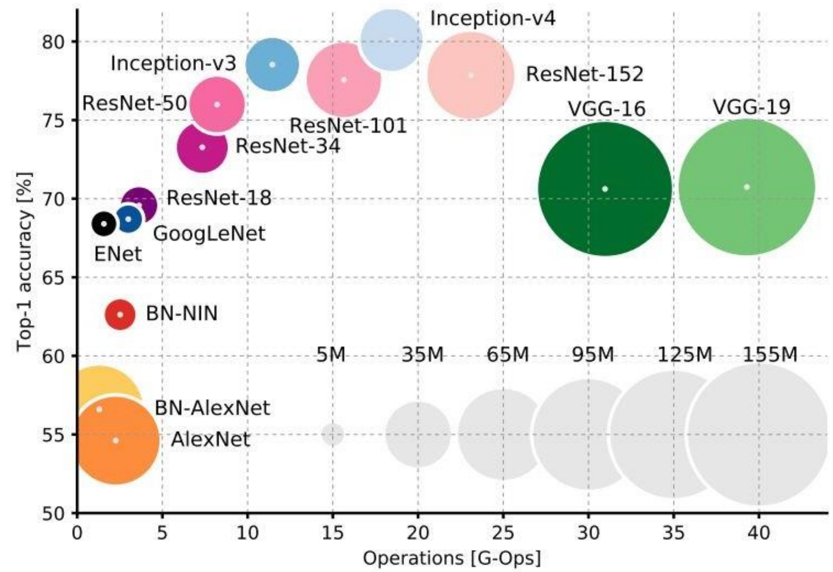
Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers



Comparing complexity...





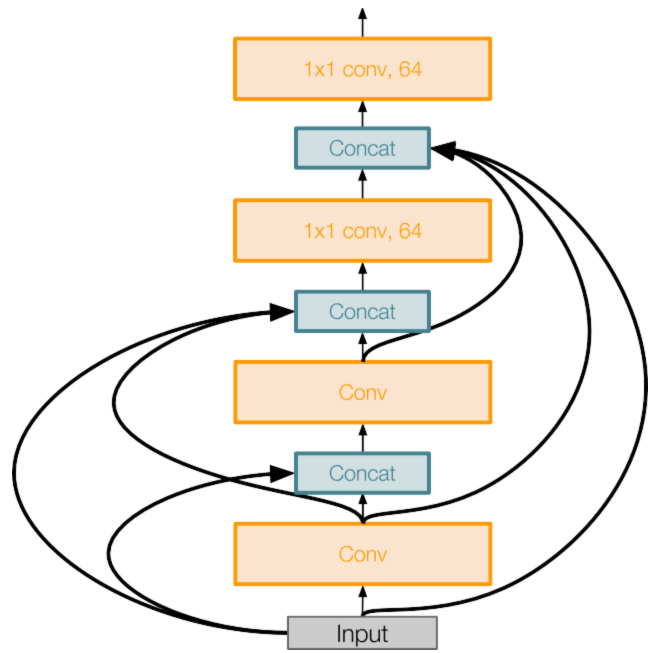
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Beyond ResNets...

Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



Dense Block

Summary: CNN Architectures

Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....

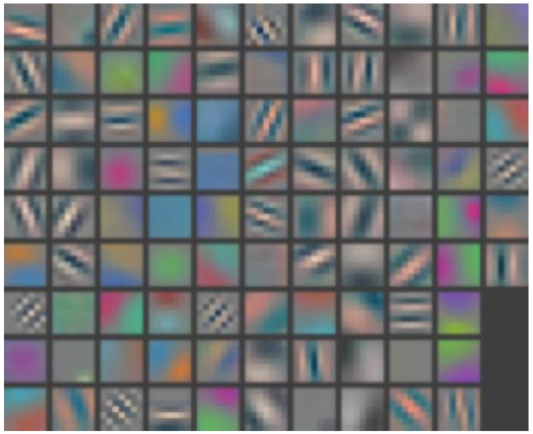
- DenseNet

Summary: CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow
- Even more recent trend towards *automatic design of network architecture* (e.g. neural architecture search)

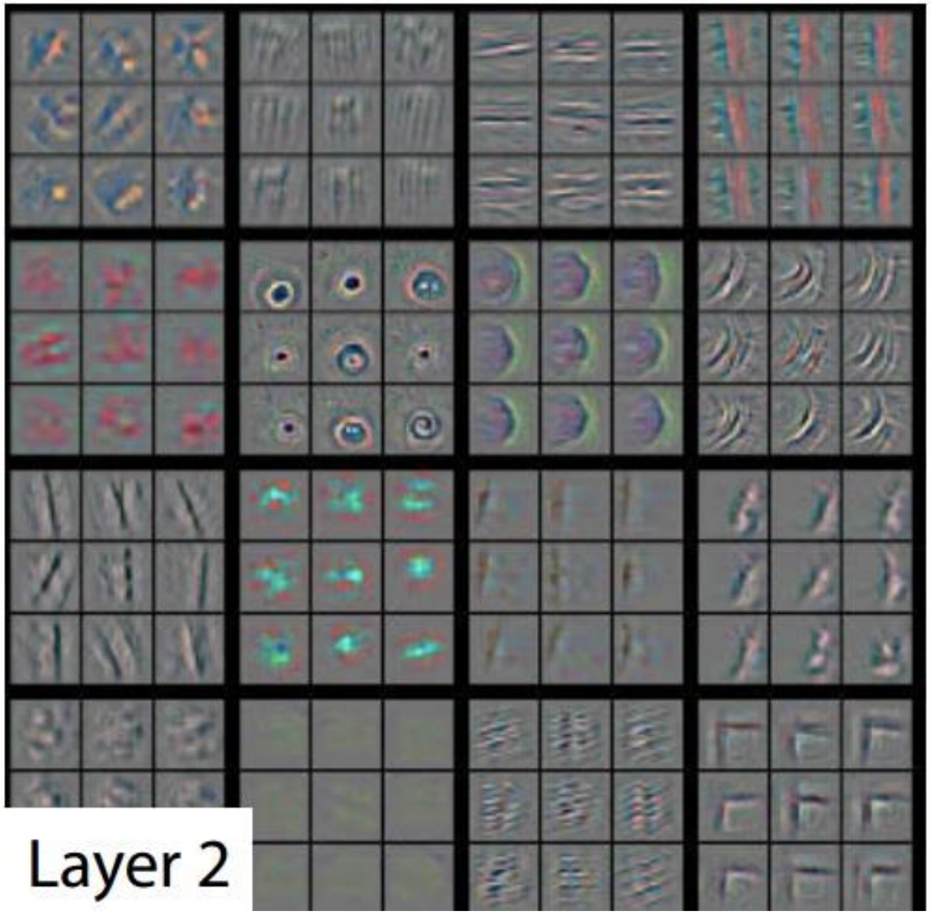
Understanding CNNs

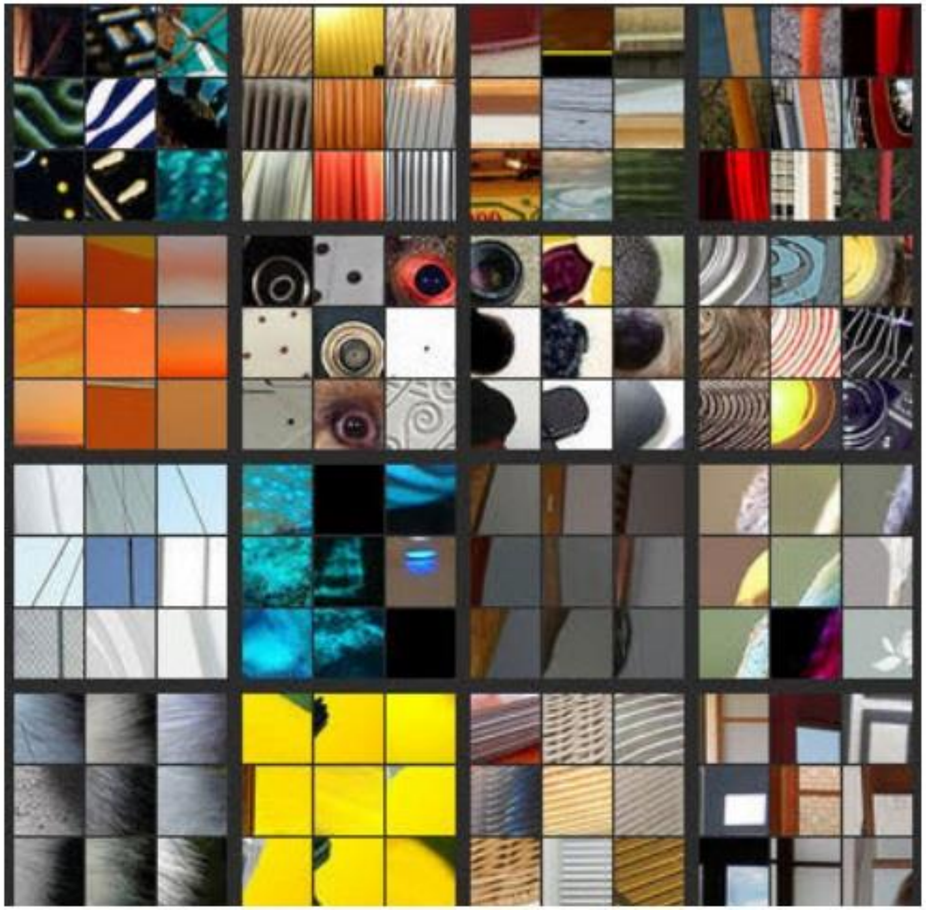
Layer 1



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 2



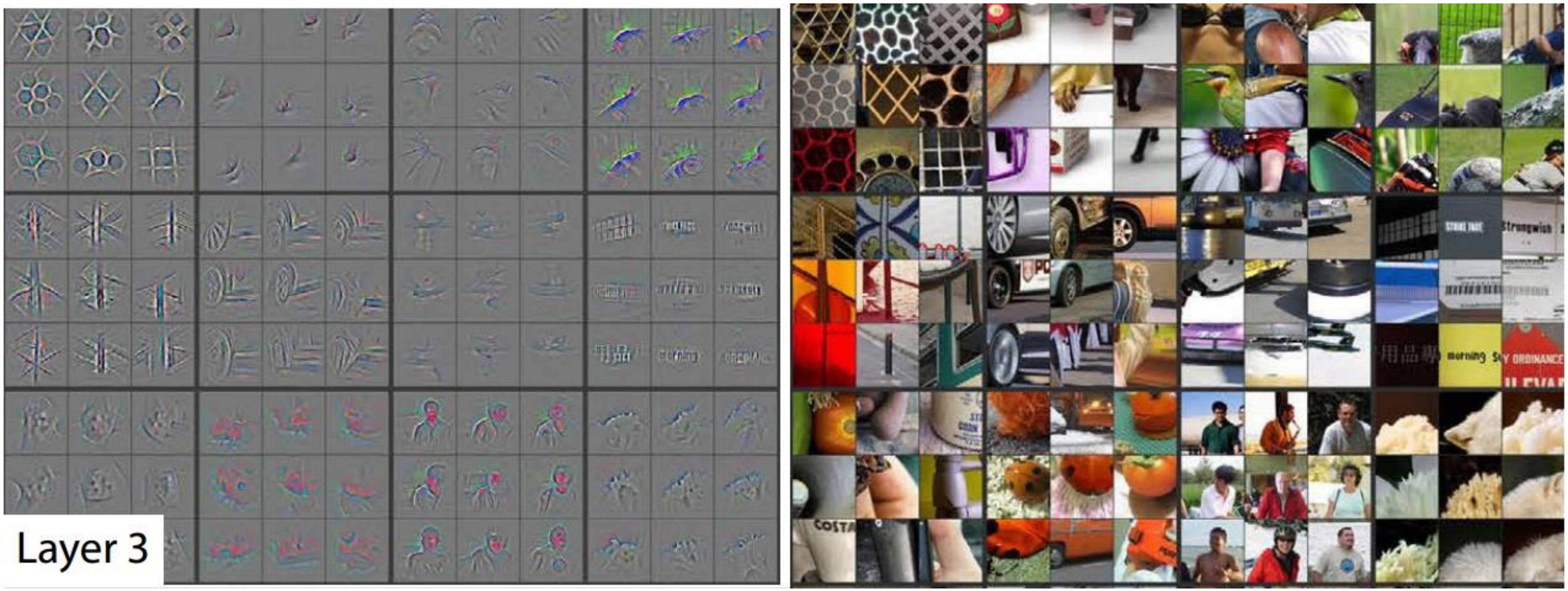


- Activations projected down to pixel level via decovolution

- Patches from validation images that give maximal activation of a given feature map

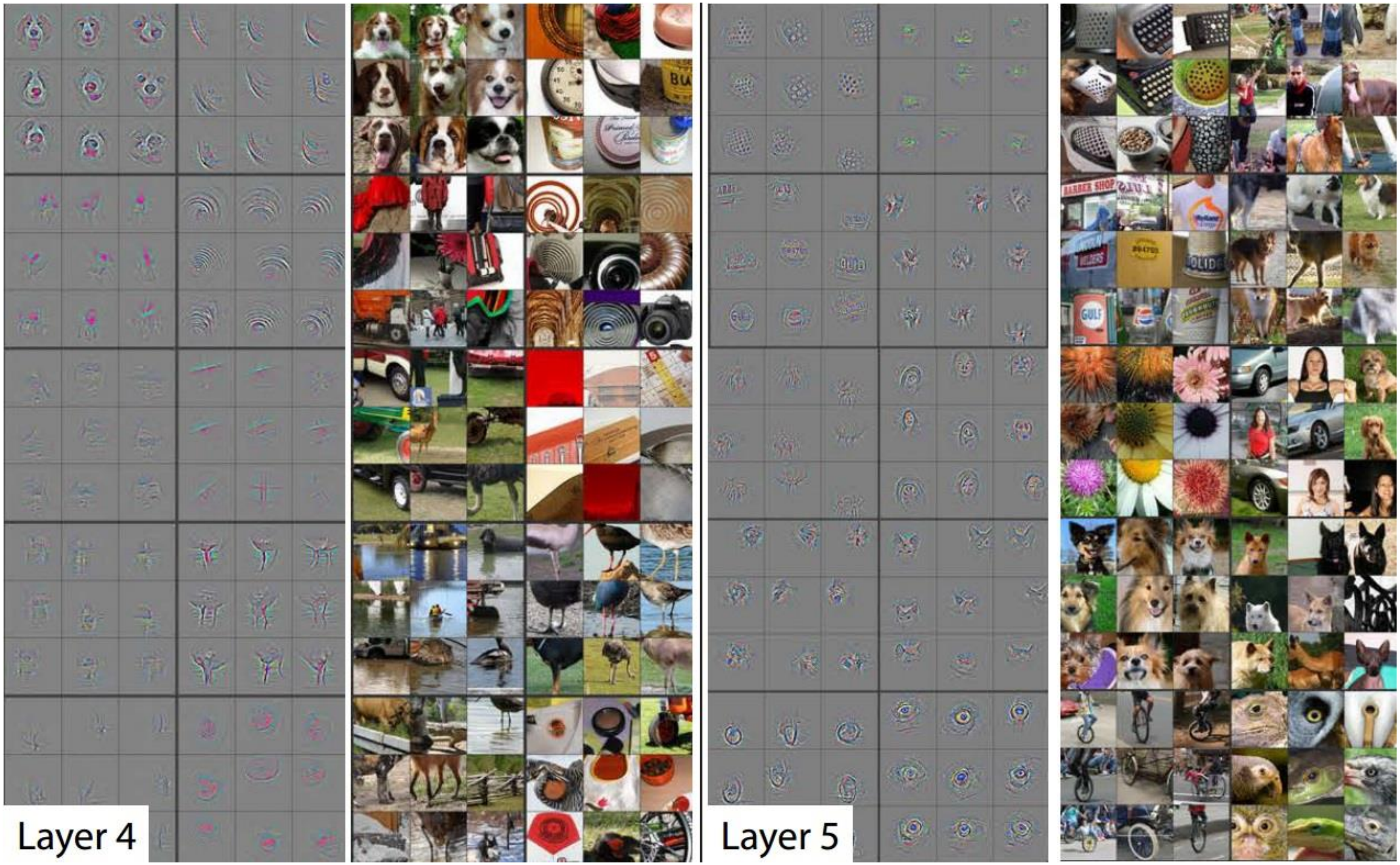
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 3



Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 4 and 5



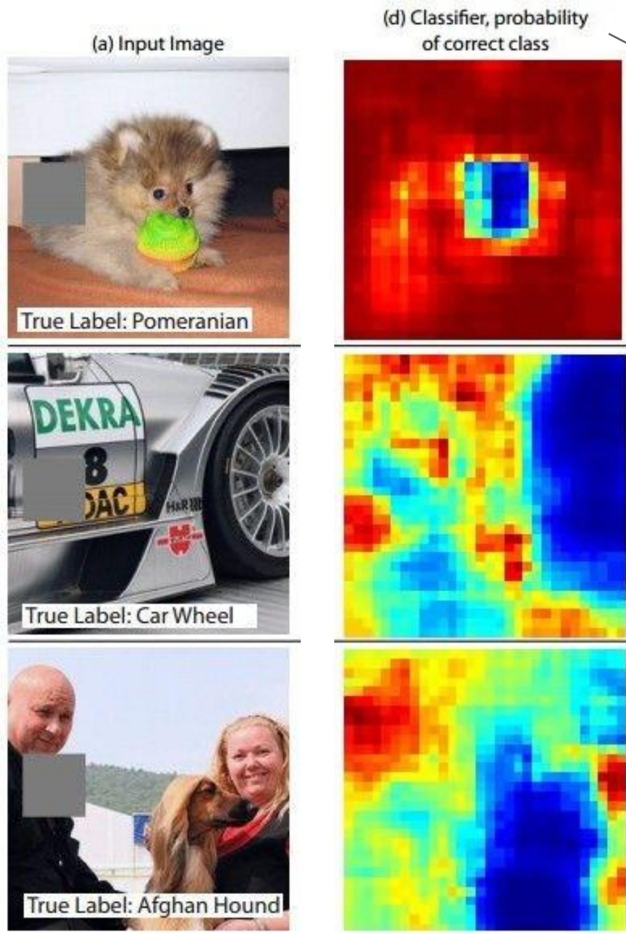
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Occlusion experiments

(as a function of the position of the square of zeros in the original image)

[Zeiler & Fergus 2014]

Occlusion experiments



(as a function of the position of the square of zeros in the original image)

[Zeiler & Fergus 2014]

CAM

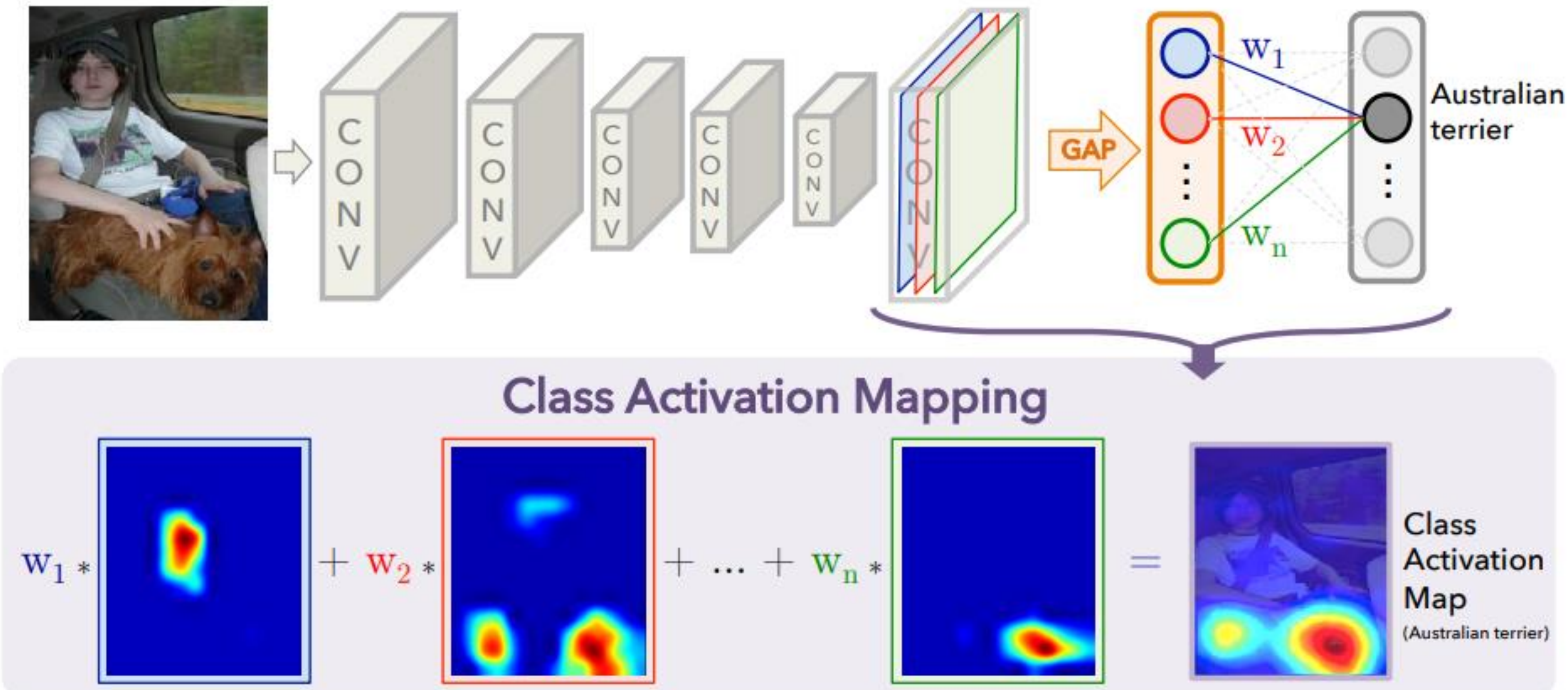
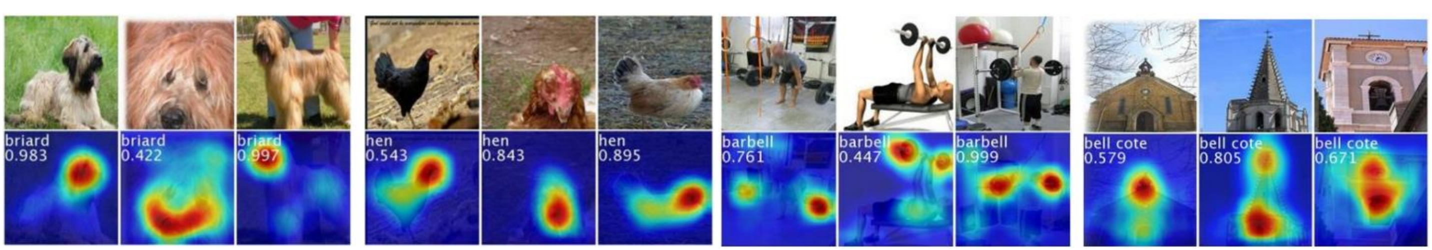


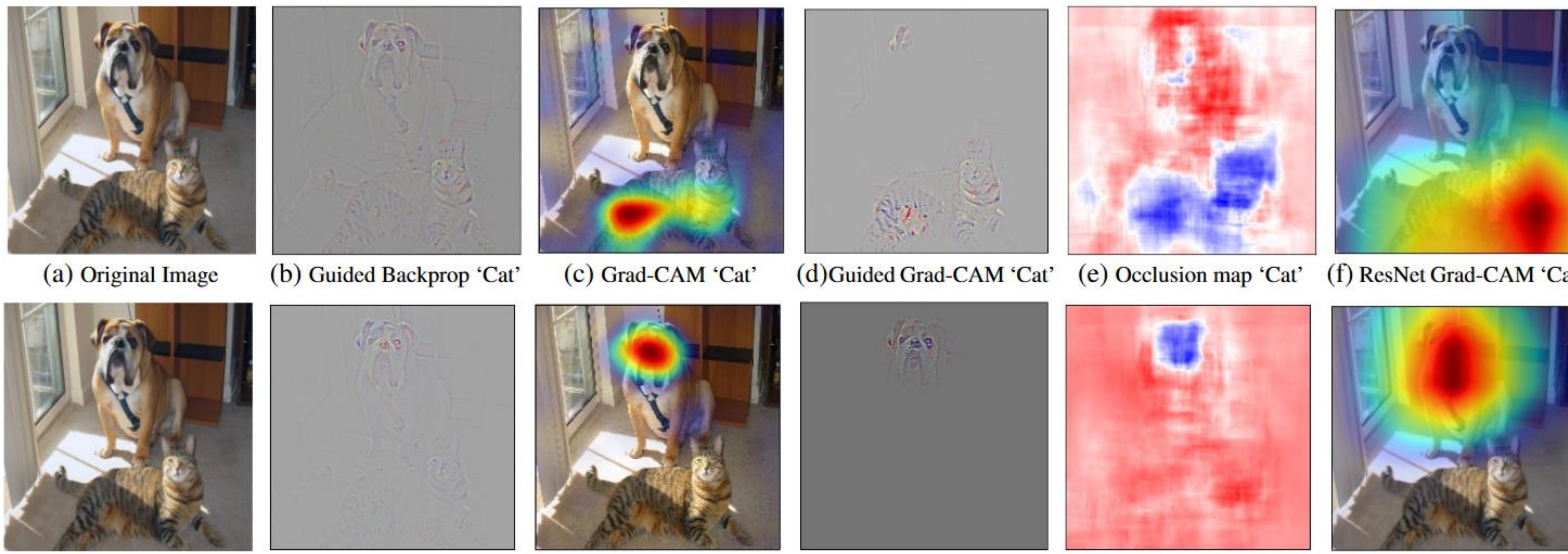
Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.



Zhou et al. CVPR 2016

GradCAM

Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization 3



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d)Guided Grad-CAM 'Cat' (e) Occlusion map 'Cat' (f) ResNet Grad-CAM 'Cat'

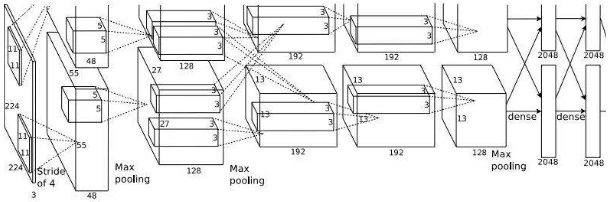
(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j)Guided Grad-CAM 'Dog' (k) Occlusion map 'Dog' (l)ResNet Grad-CAM 'Dog'

Fig. 1: (a) Original image with a cat and a dog. (b-f) Support for the cat category according to various visualizations for VGG-16 and ResNet. (b) Guided Backpropagation [53]: highlights all contributing features. (c, f) Grad-CAM (Ours): localizes class-discriminative regions, (d) Combining (b) and (c) gives Guided Grad-CAM, which gives high-resolution class-discriminative visualizations. Interestingly, the localizations achieved by our Grad-CAM technique, (c) are very similar to results from occlusion sensitivity (e), while being orders of magnitude cheaper to compute. (f, l) are Grad-CAM visualizations for ResNet-18 layer. Note that in (c, f, i, l), red regions corresponds to high score for class, while in (e, k), blue corresponds to evidence for the class. Figure best viewed in color.

Selvaraju et al. ICCV 2017, IJCV 2019

Applications in computer vision

Image Classification



Vector:
4096

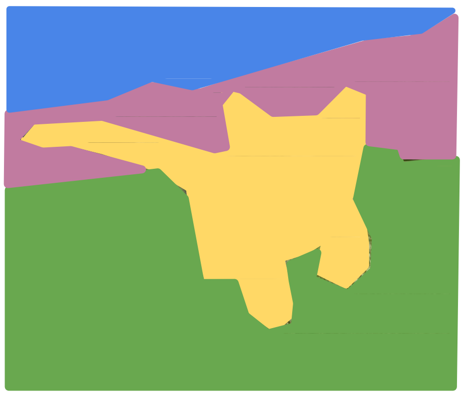
Fully-Connected:
4096 to 1000

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

Slide by: Justin Johnson

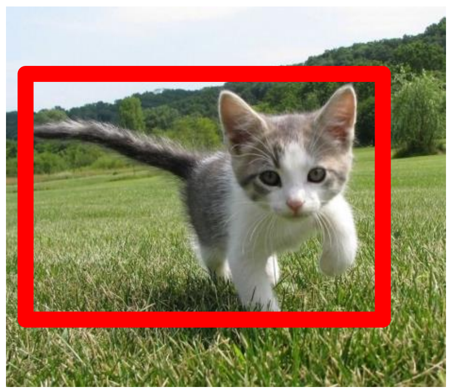
Other Computer Vision Tasks

Semantic Segmentation



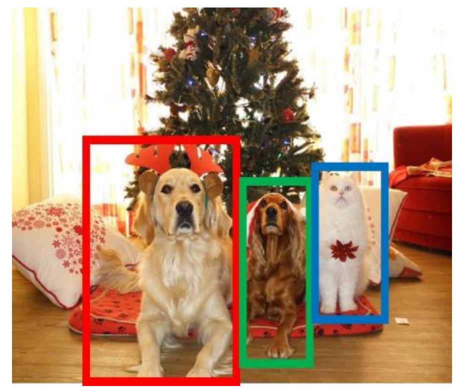
GRASS, CAT, TREE, SKY

Classification + Localization



CAT

Object Detection



DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

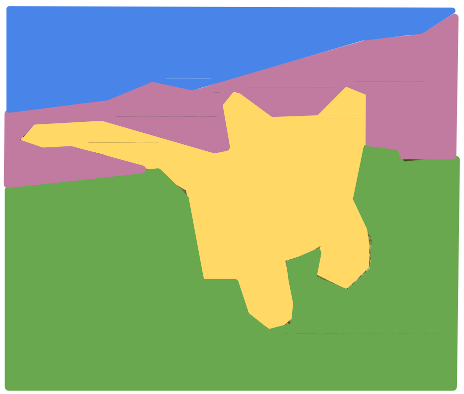
No objects, just pixels

Single Object

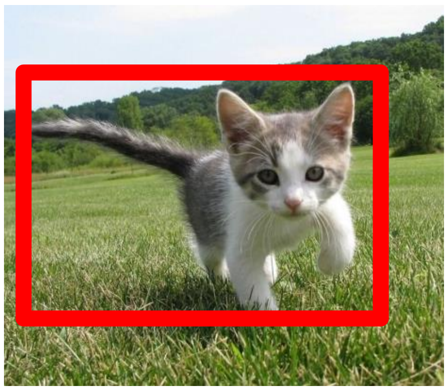
Multiple Object

Slide by: Justin Johnson

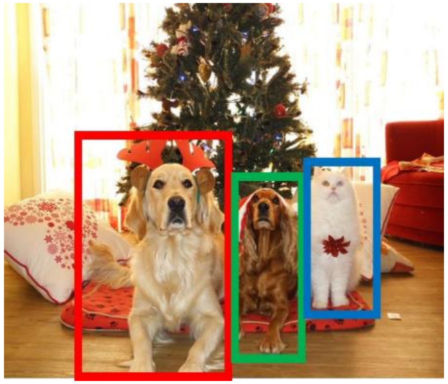
Classification + Localization



GRASS, CAT, TREE, SKY



CAT



DOG, DOG, CAT

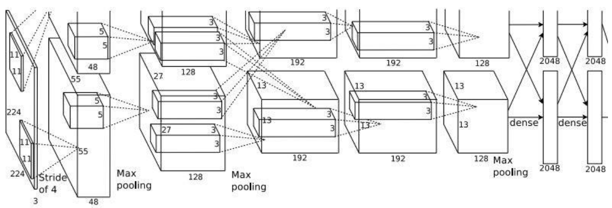
DOG, DOG, CAT

No objects, just pixels

Single Object

Multiple Object

Classification + Localization



Fully Connected:
4096 to 1000

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

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

Slide by: Justin Johnson

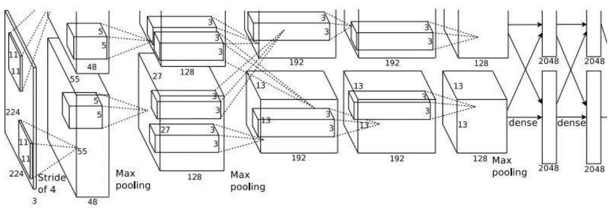
Classification + Localization

Correct label:
Cat

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

Softmax Loss

Fully Connected:
4096 to 1000



Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

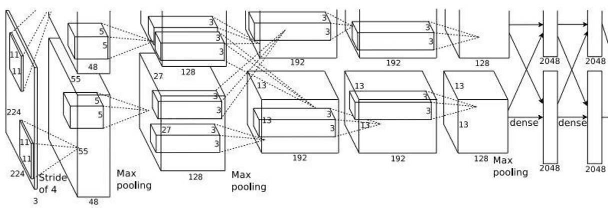
L2 Loss

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

Treat localization as a regression problem!

Slide by: Justin Johnson

Classification + Localization



Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

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

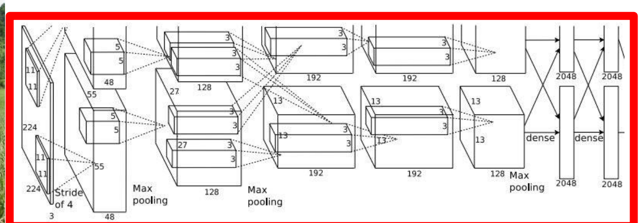
Multitask Loss

\+ **Loss**

Treat localization as a regression problem!

Slide by: Justin Johnson

Classification + Localization



Often pretrained on ImageNet
(Transfer learning)

Vector:
4096

Fully Connected:
4096 to 1000

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

Correct label:
Cat

Softmax Loss

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

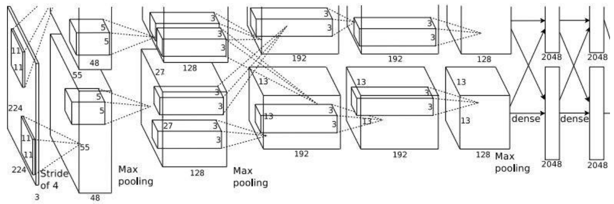
Correct box:
(x', y', w', h')

+ **Loss**

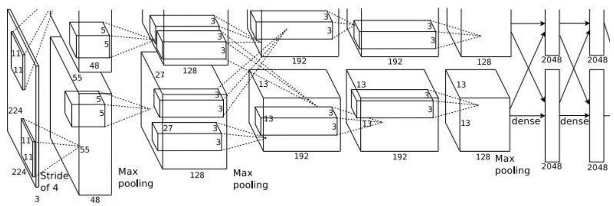
Treat localization as a regression problem!

Slide by: Justin Johnson

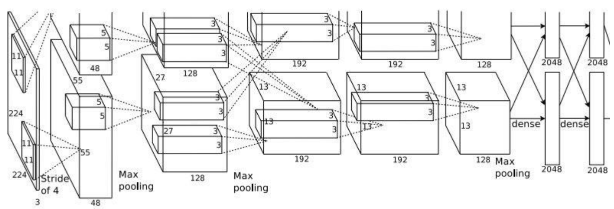
Object Detection as Regression?



CAT: (x, y, w, h)



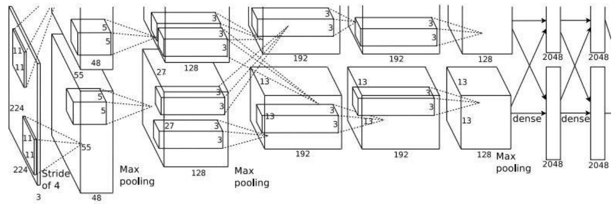
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

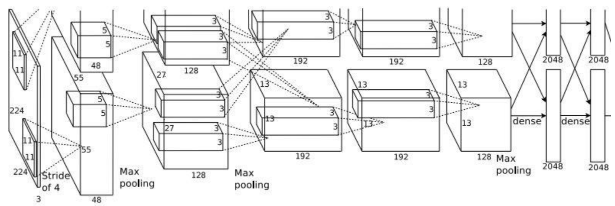
Slide by: Justin Johnson

Object Detection as Regression?



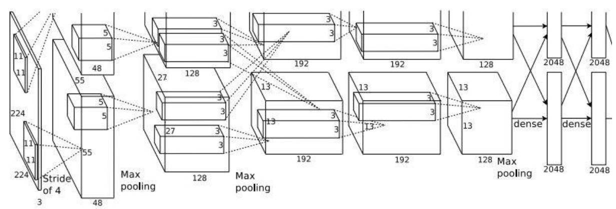
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

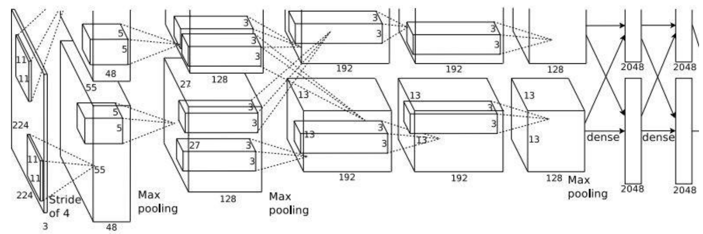
Many numbers!

Each image needs a different number of outputs!

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

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

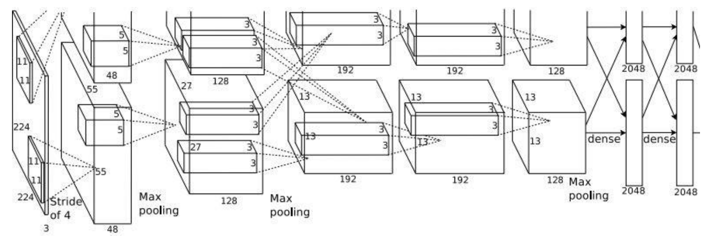


Dog? NO
Cat? NO
Background? YES

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

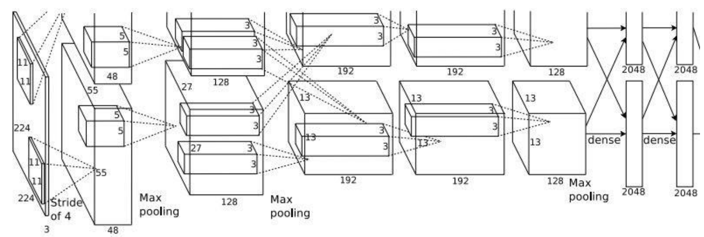
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

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

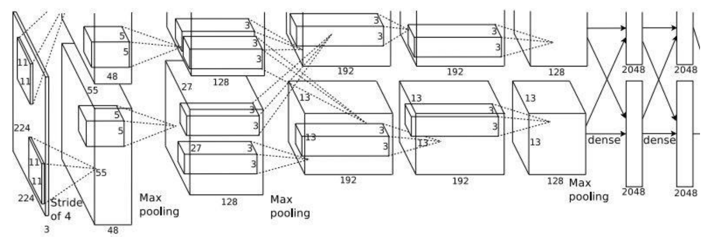


Dog? YES
Cat? NO
Background? NO

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

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

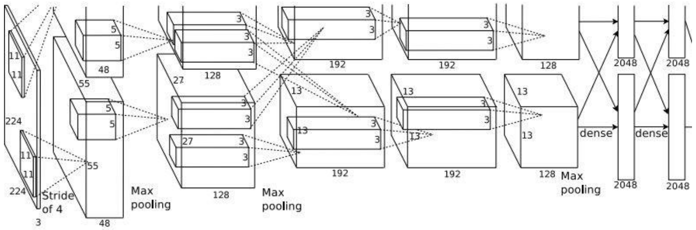


Dog? NO
Cat? YES
Background? NO

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

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



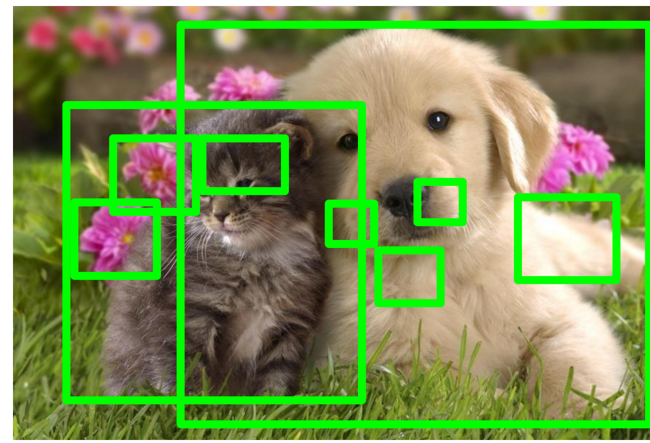
Dog? NO
Cat? YES
Background? NO

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

Slide by: Justin Johnson

Region Proposals

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



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

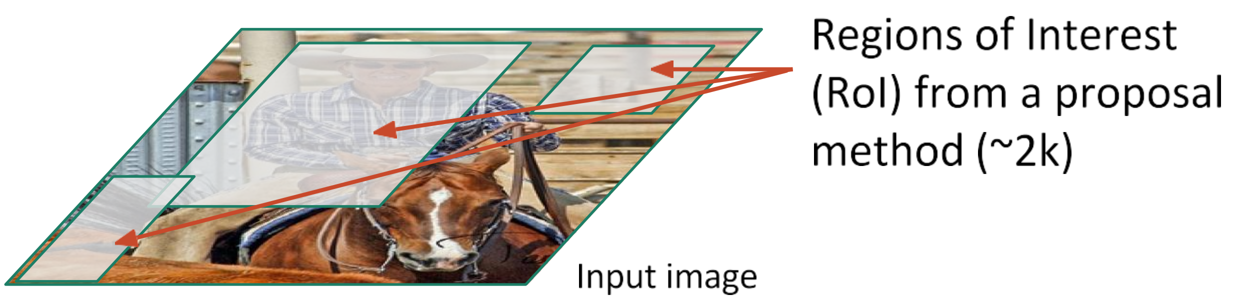
Slide by: Justin Johnson

R-CNN

Input image

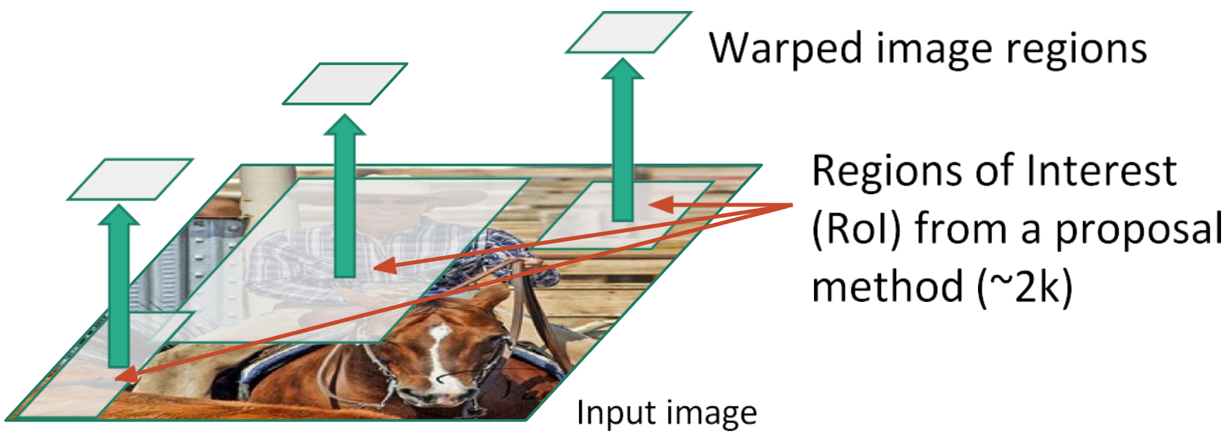
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



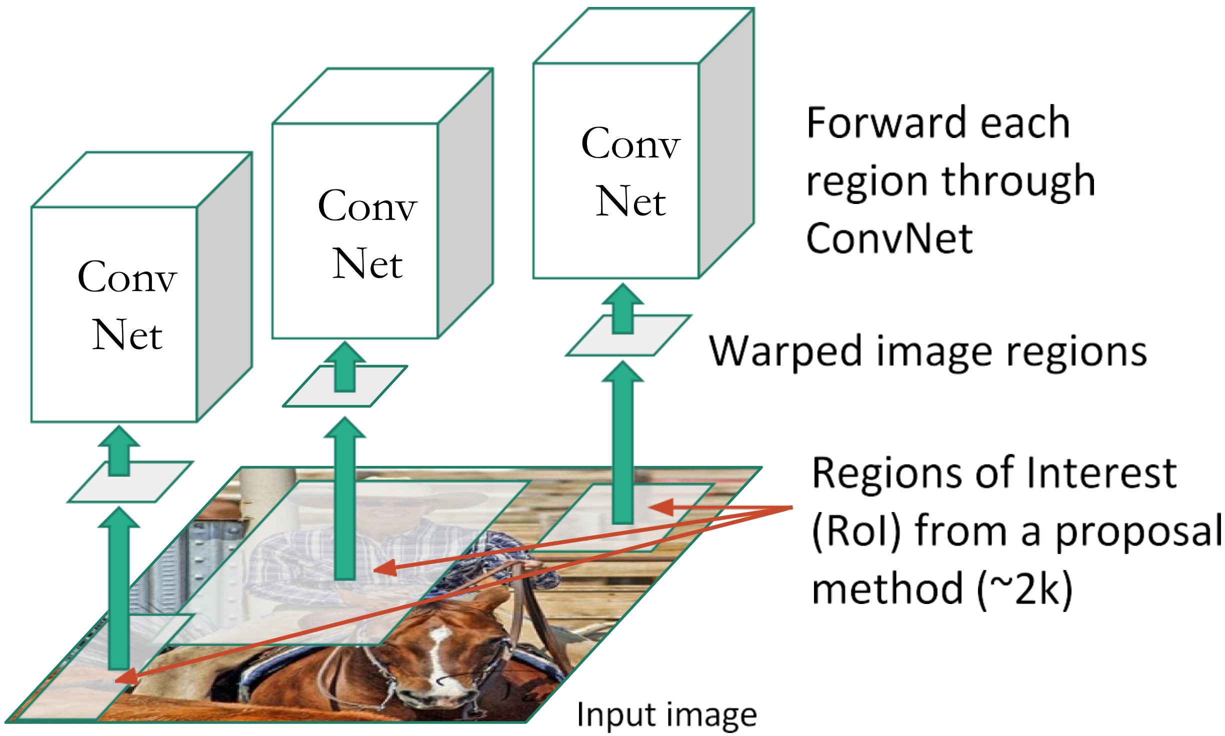
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



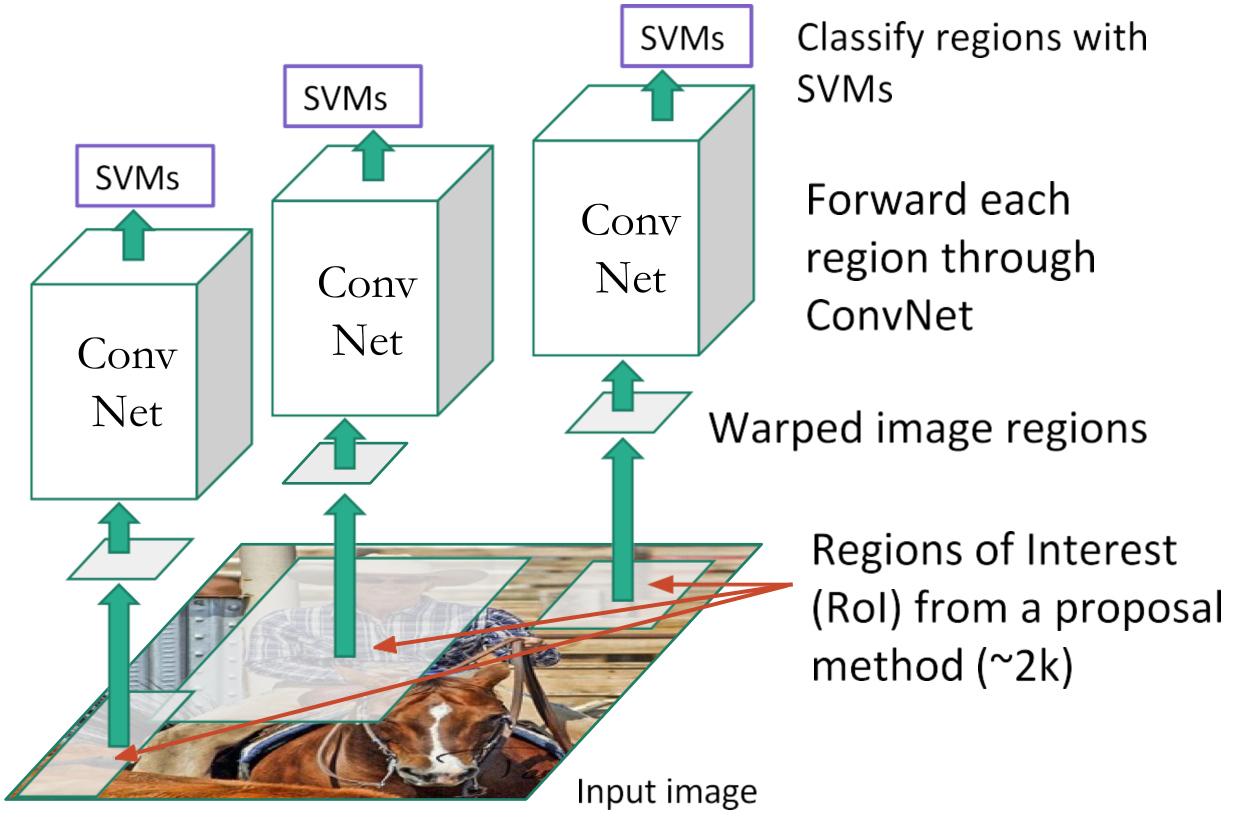
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



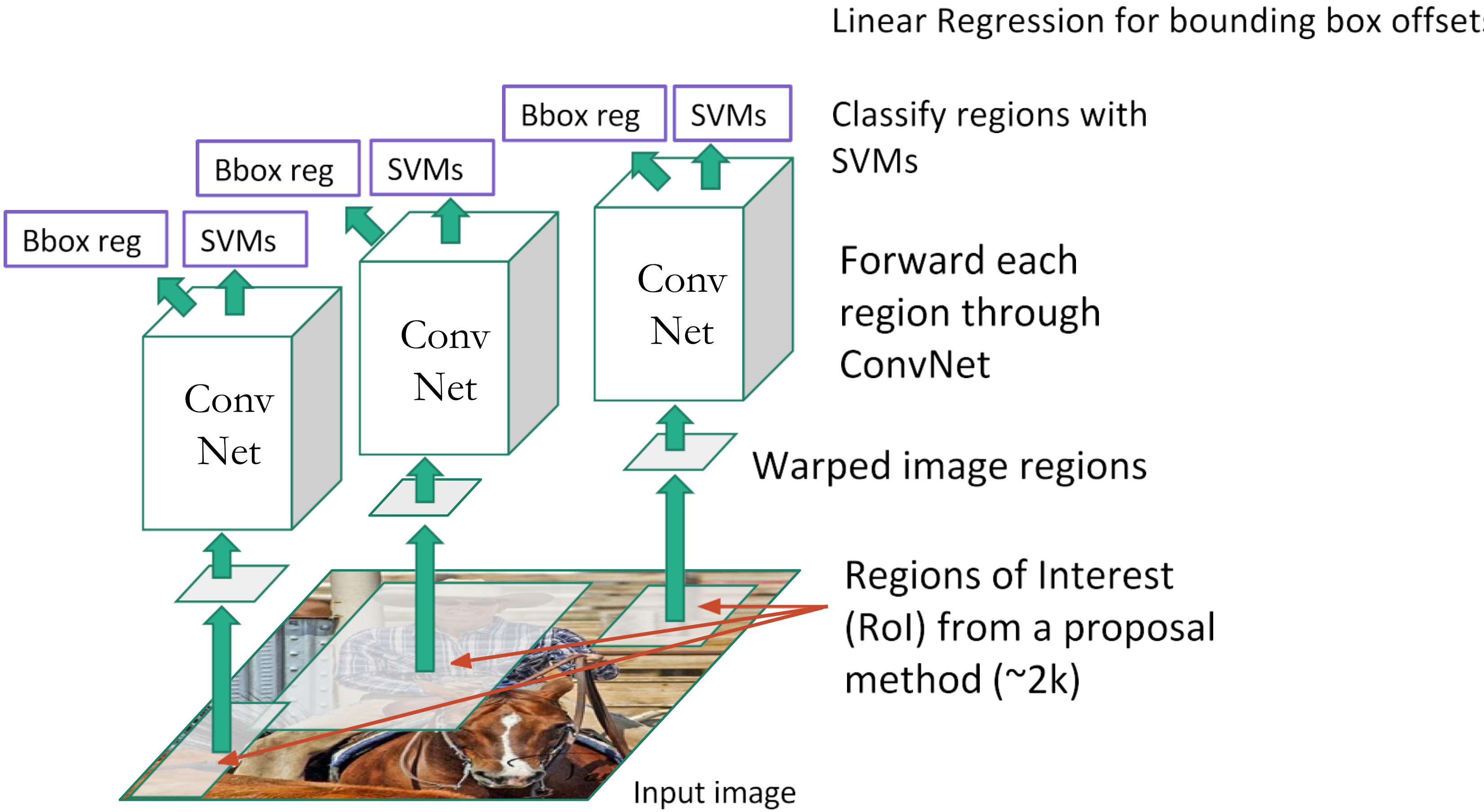
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



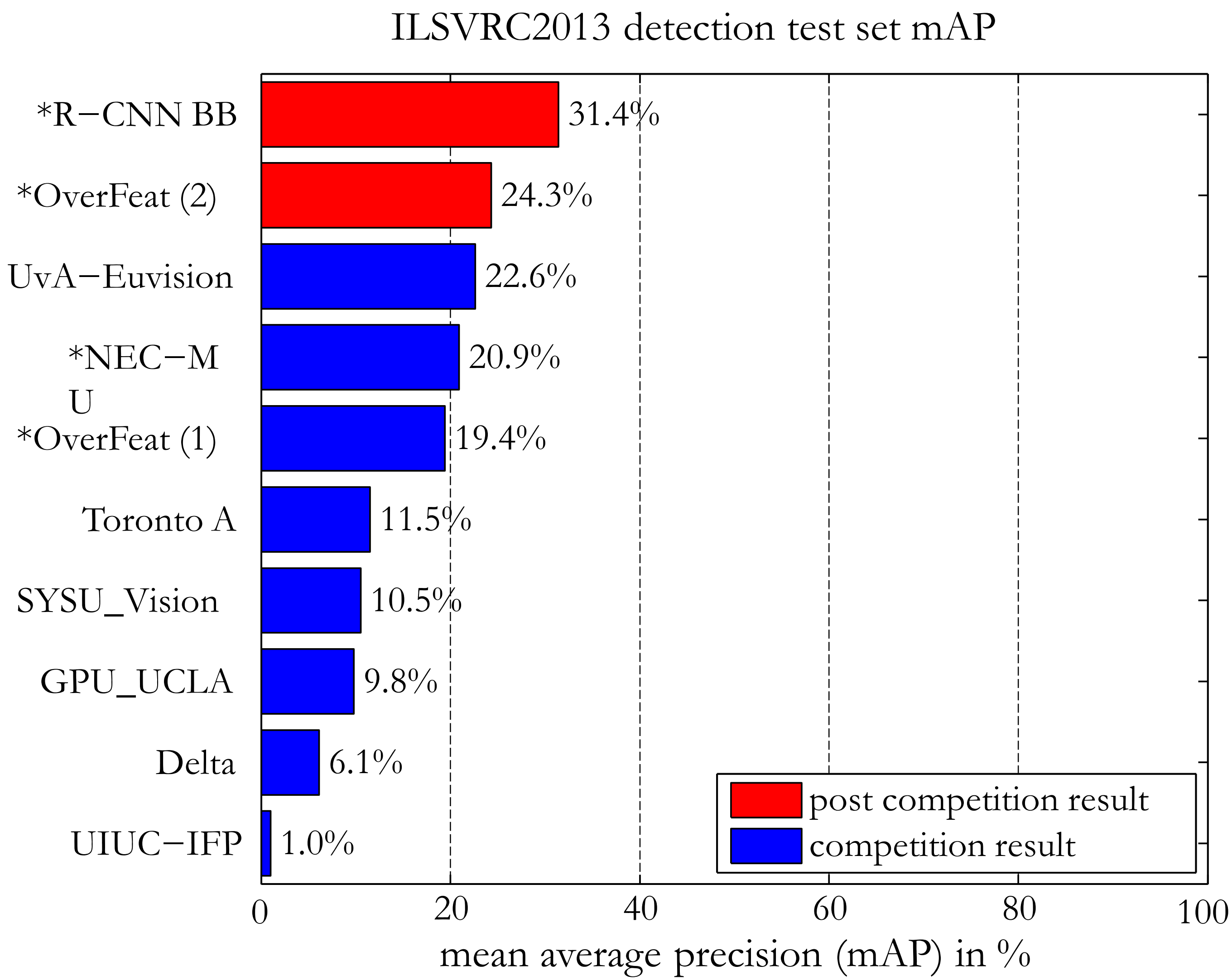
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



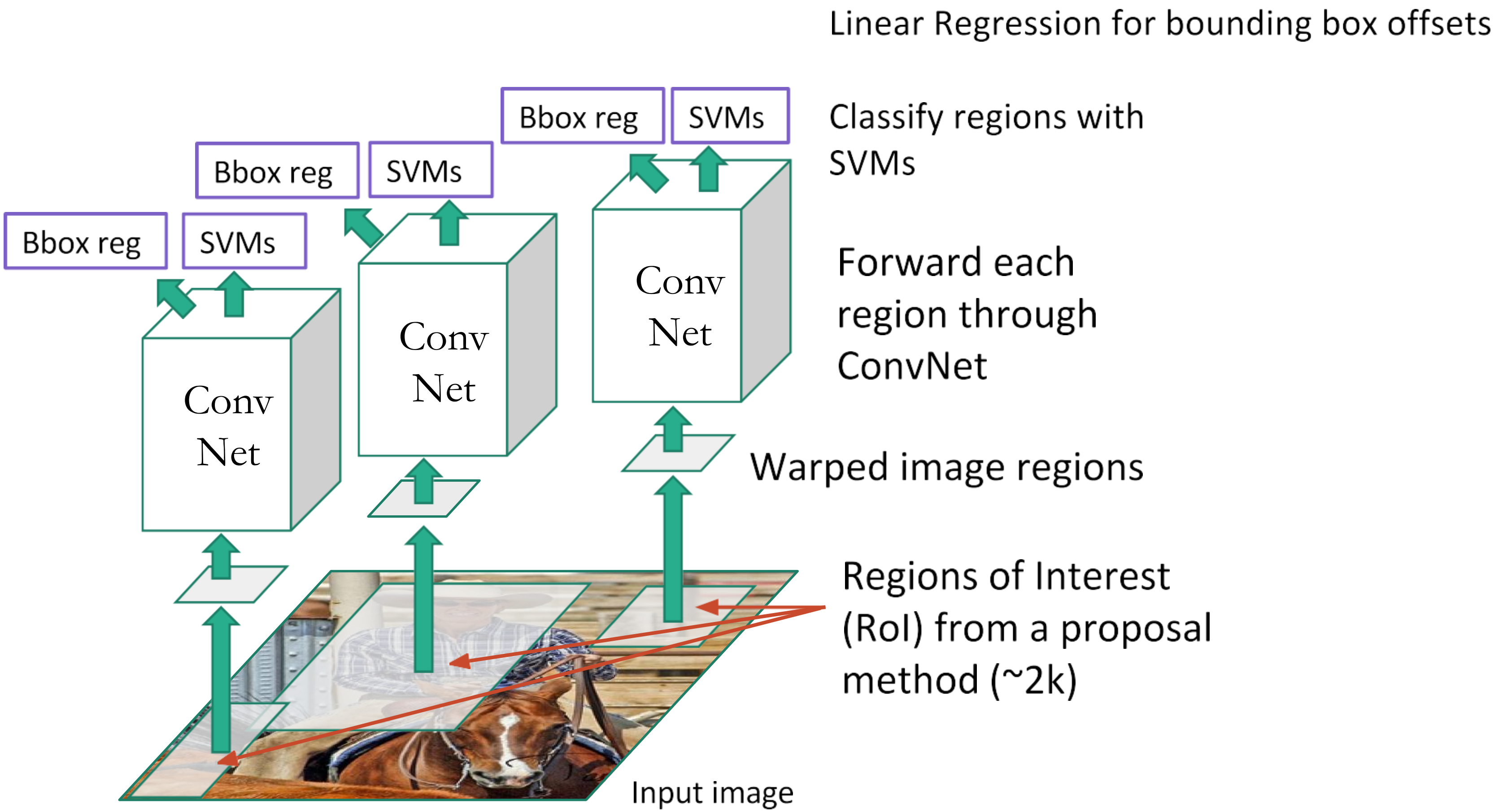
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN on ImageNet detection



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN

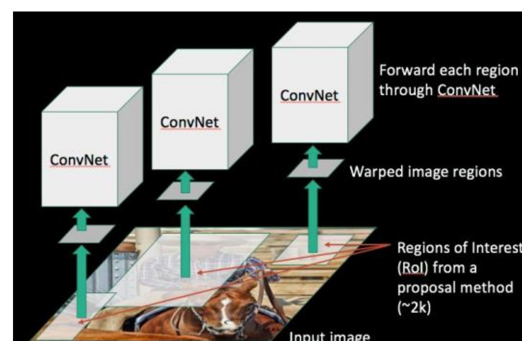


Post hoc component

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

What's wrong with slow R-CNN?

- 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]



~2000 ConvNet forward passes per image

Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN

- Fast test time
- One network, trained in one stage
- Higher mean average precision

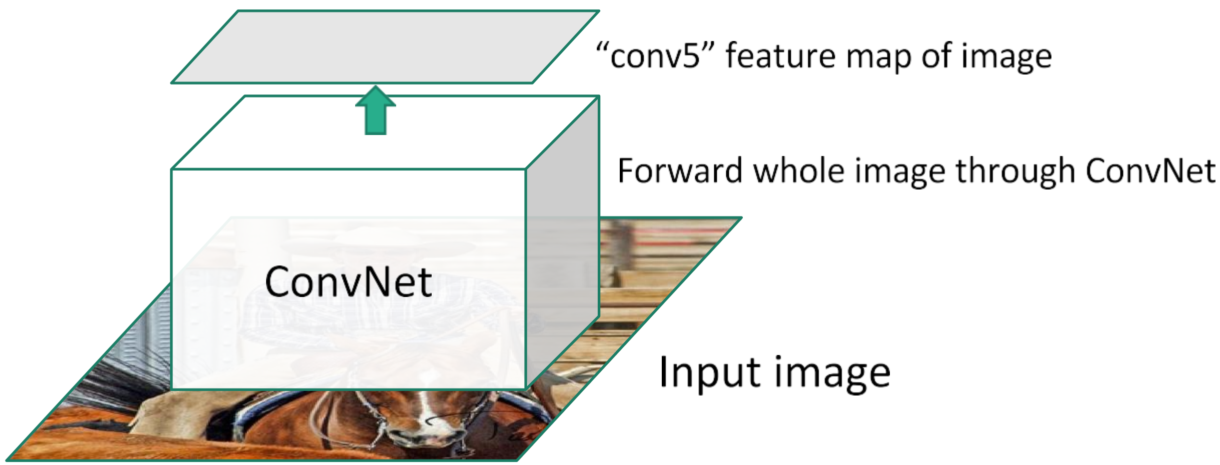
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN

Input image

Girshick, "Fast R-CNN", ICCV 2015

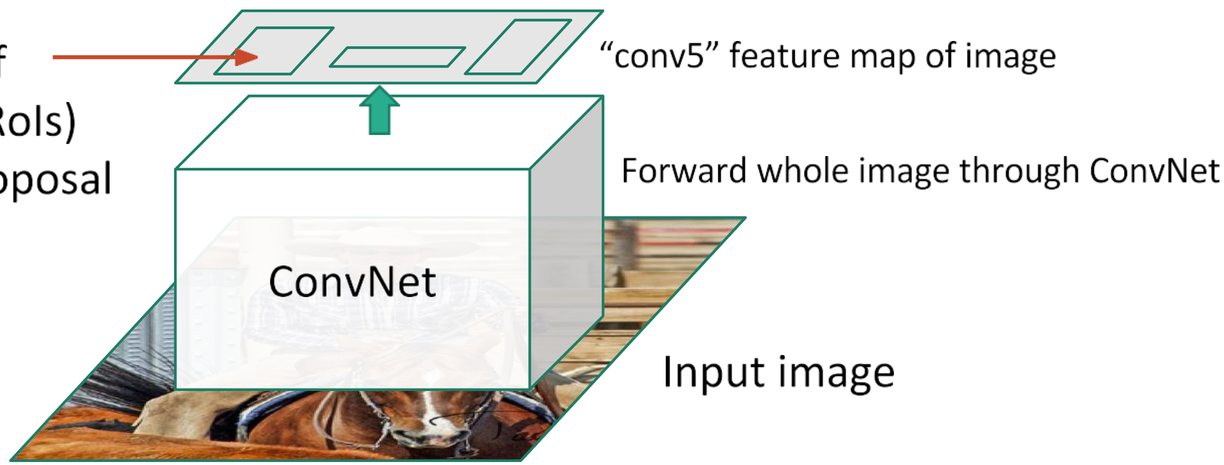
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015

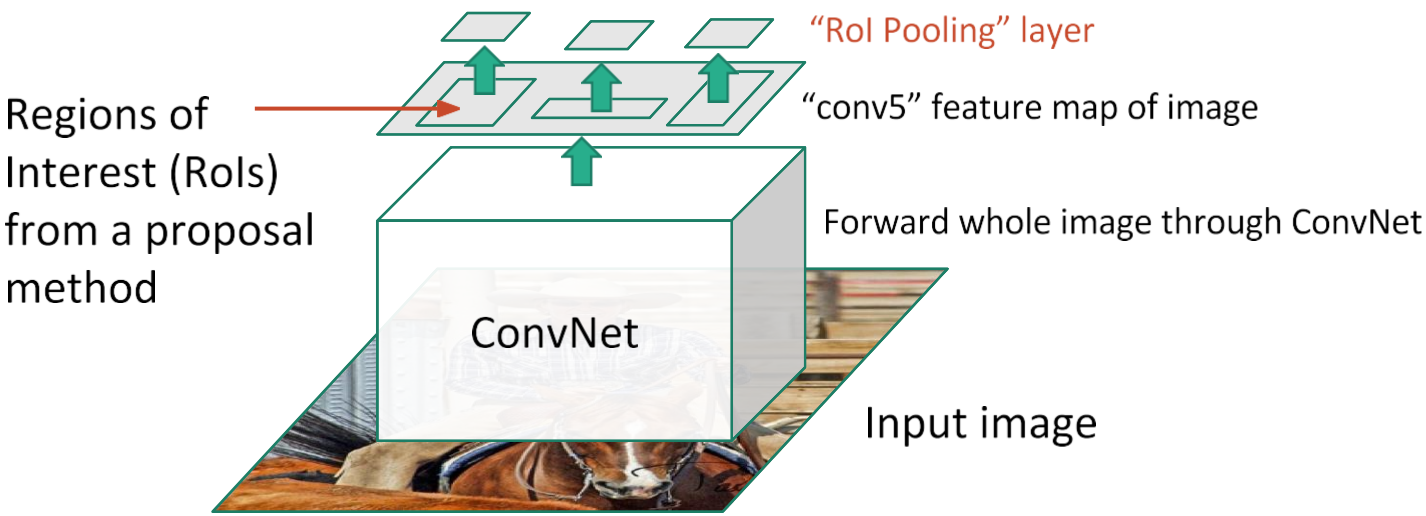
Fast R-CNN

Regions of Interest (RoIs) from a proposal method



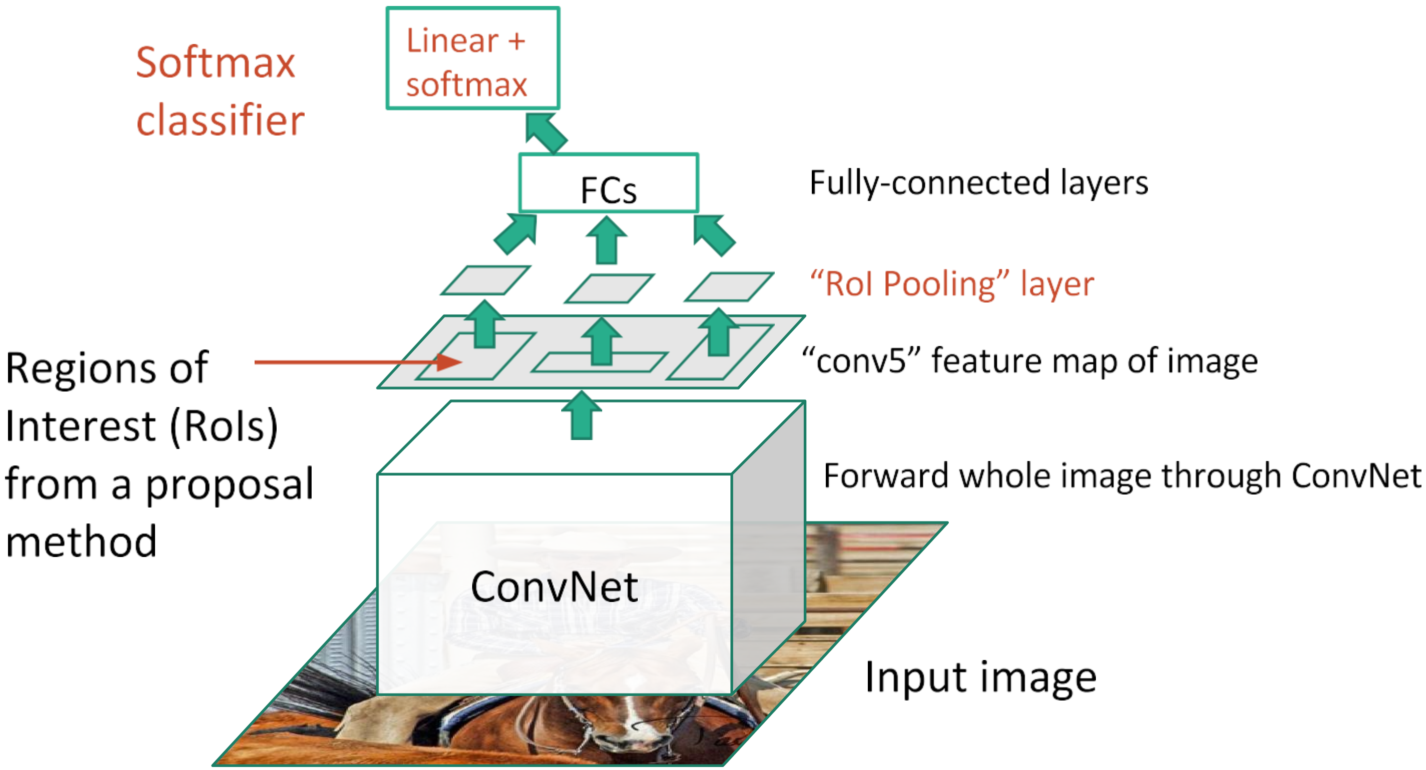
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN



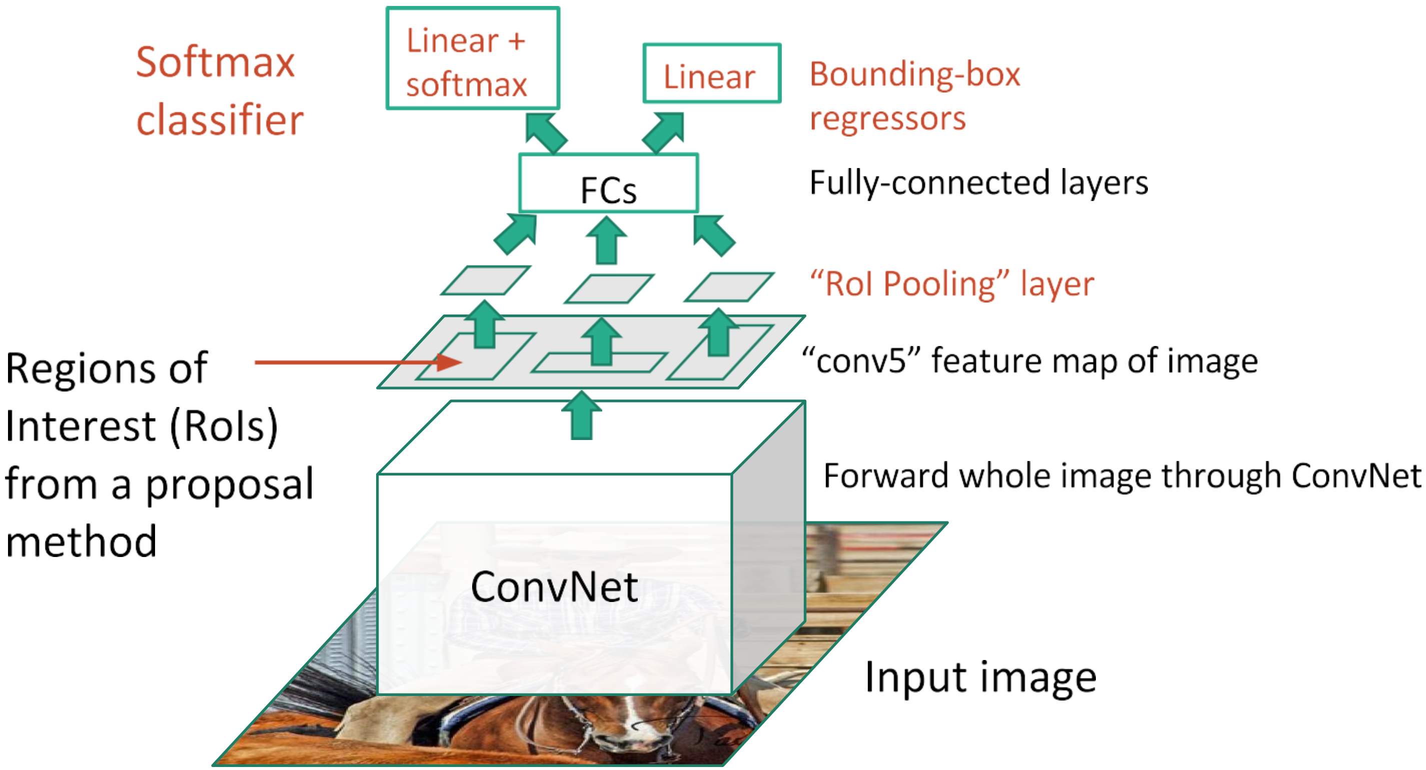
Girshick, “Fast R-CNN”, ICCV 2015

Fast R-CNN



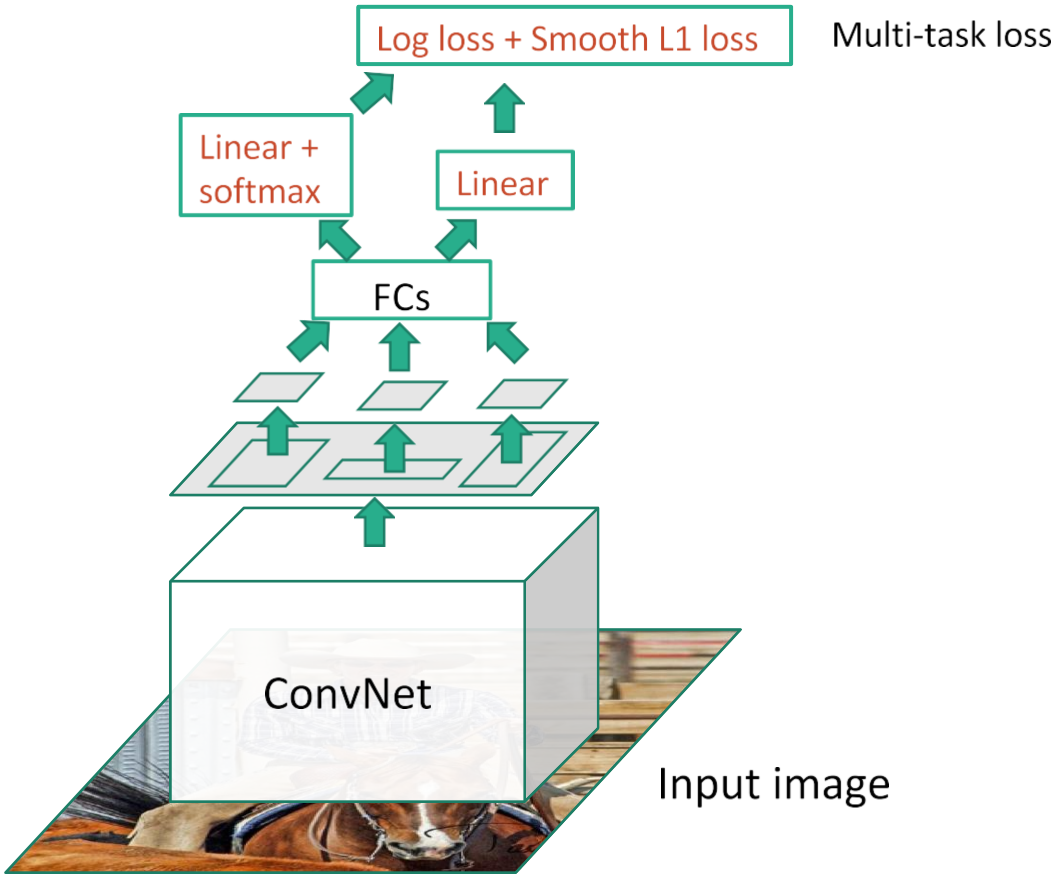
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



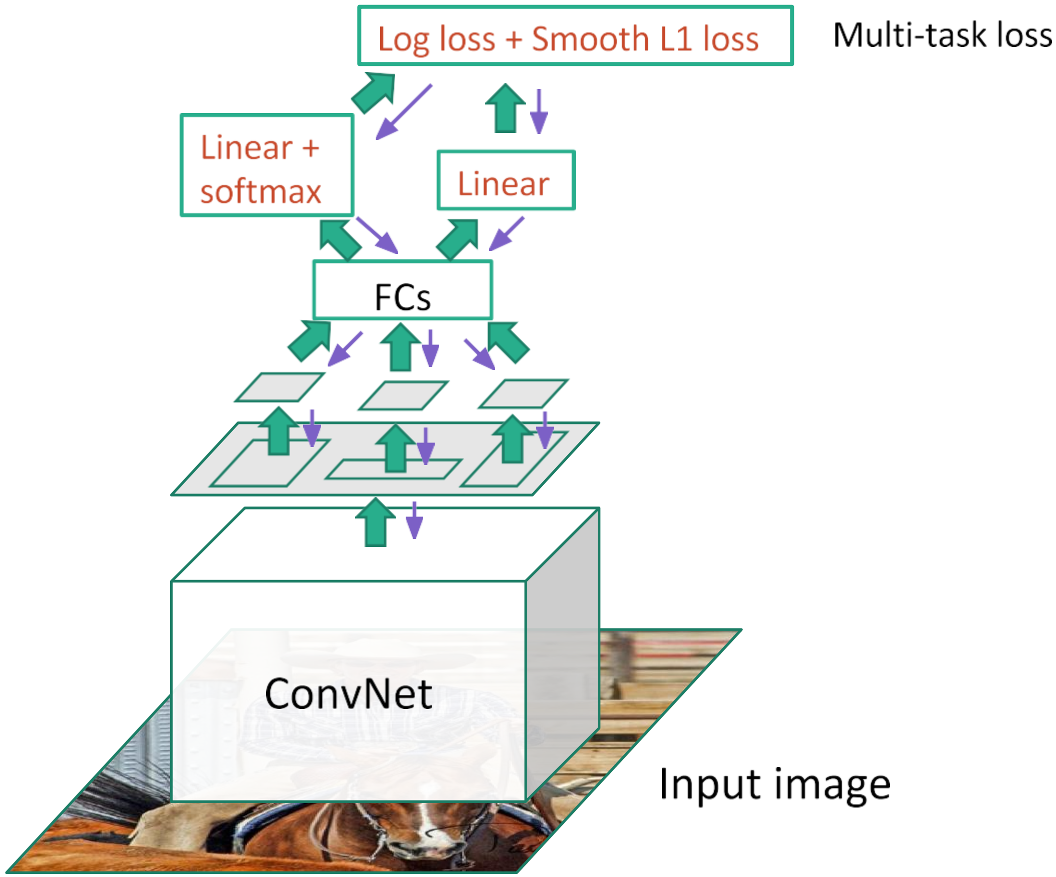
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN (Training)



Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN (Training)



Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN vs R-CNN

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
Speedup	8.8x	1x
Test time / image	0.32s	47.0s
Test speedup	146x	1x
mAP	66.9%	66.0%

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

Girshick, "Fast R-CNN", ICCV 2015

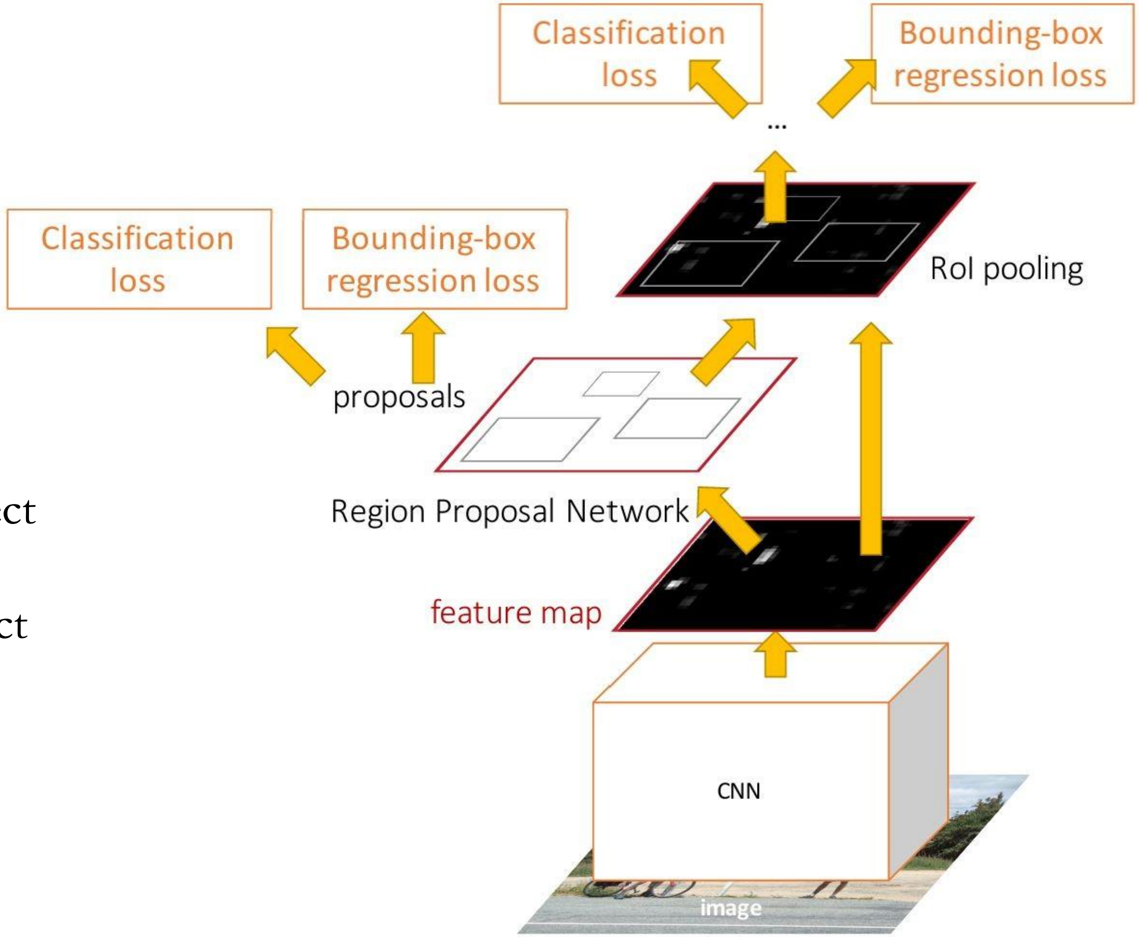
Faster R-CNN

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

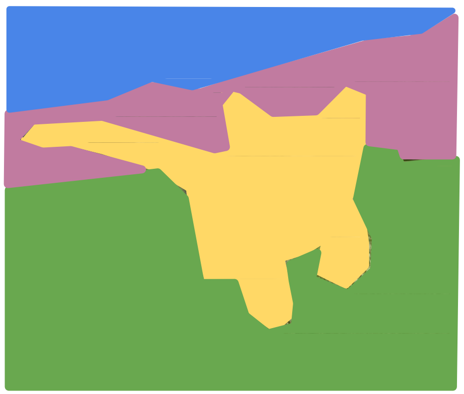
Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



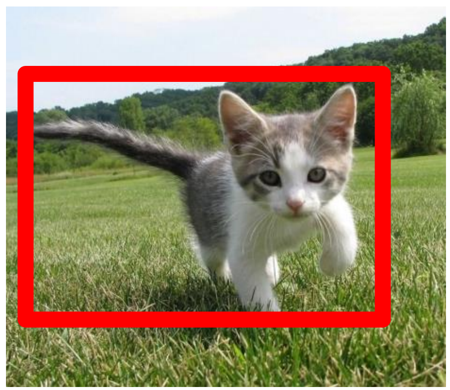
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Semantic Segmentation



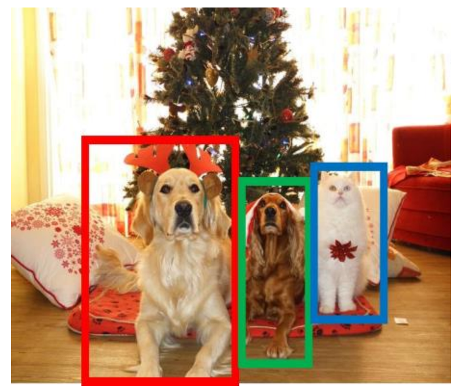
GRASS, CAT, TREE, SKY

No objects, just pixels



CAT

Single Object



DOG, DOG, CAT

DOG, DOG, CAT

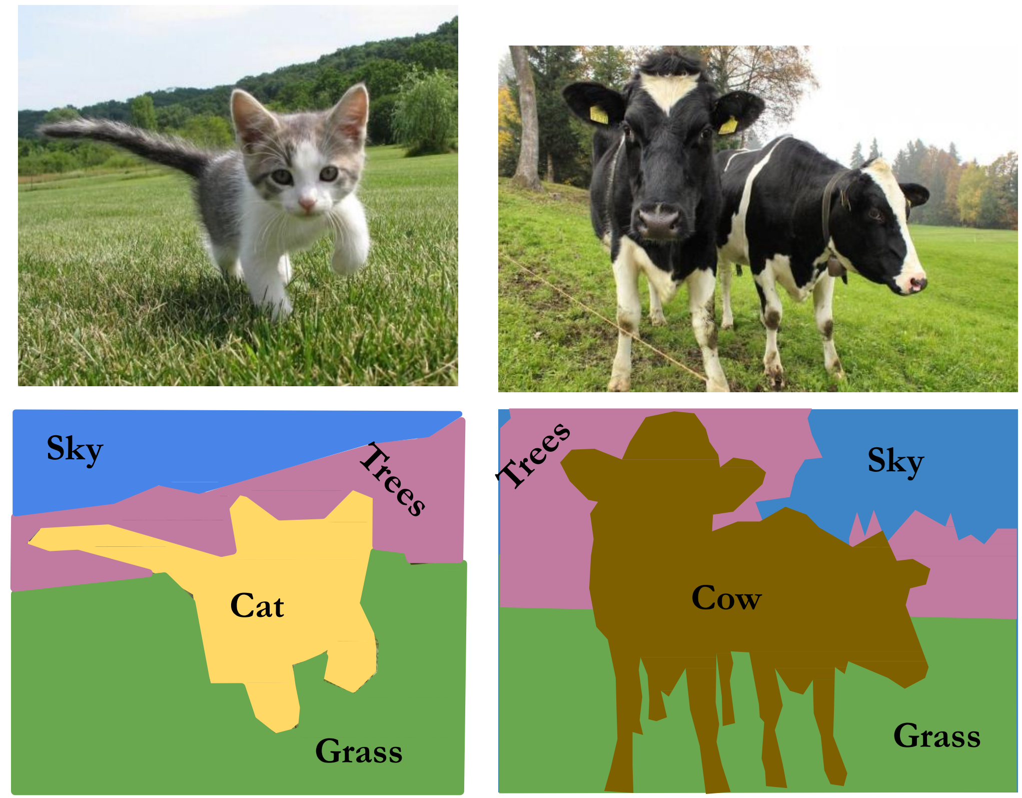
Multiple Object

Slide by: Justin Johnson

Semantic Segmentation

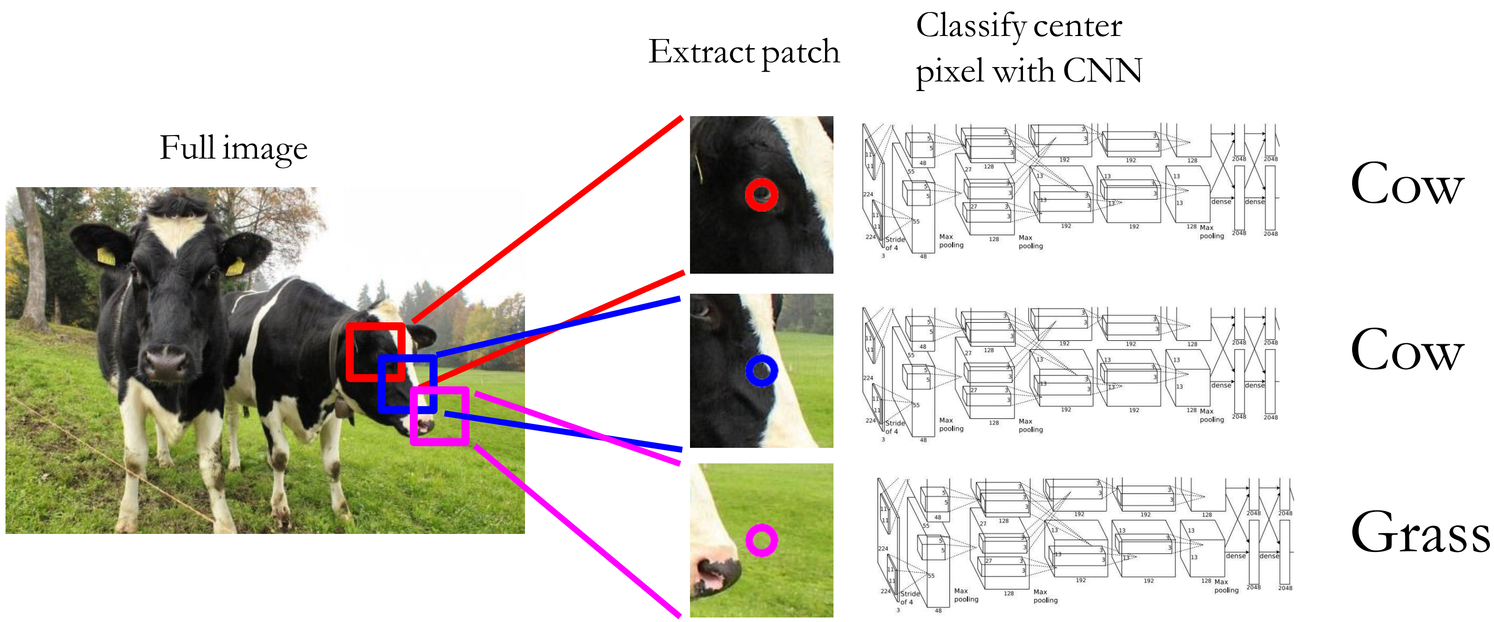
Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels



Slide by: Justin Johnson

Semantic Segmentation Idea: Sliding Window



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

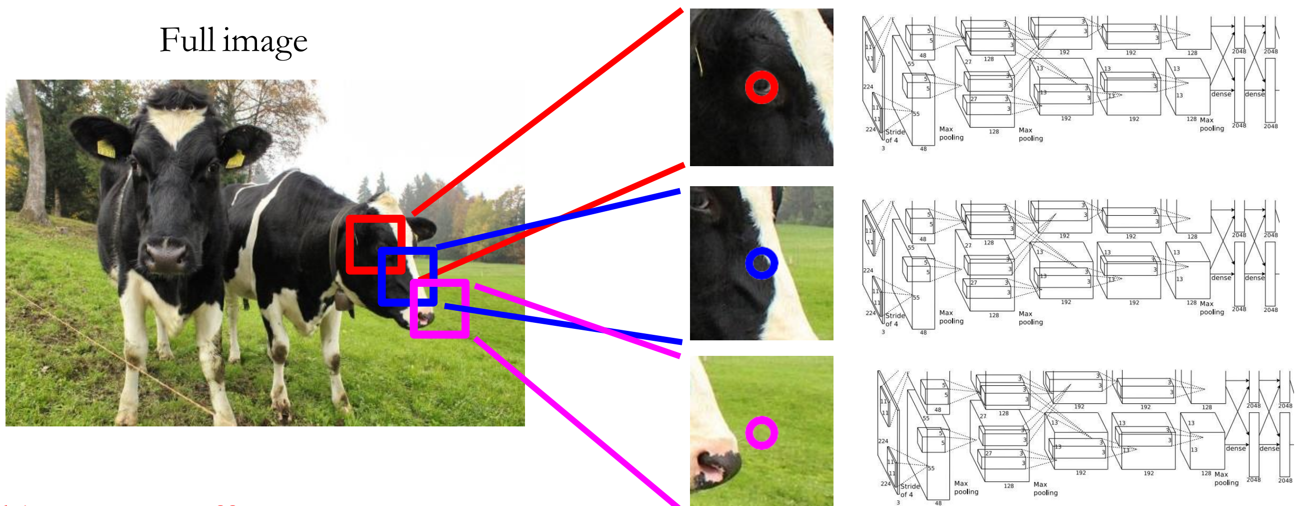
Slide by: Justin Johnson

Semantic Segmentation Idea: Sliding Window

Extract patch

Classify center pixel with CNN

Full image



Cow

Cow

Grass

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

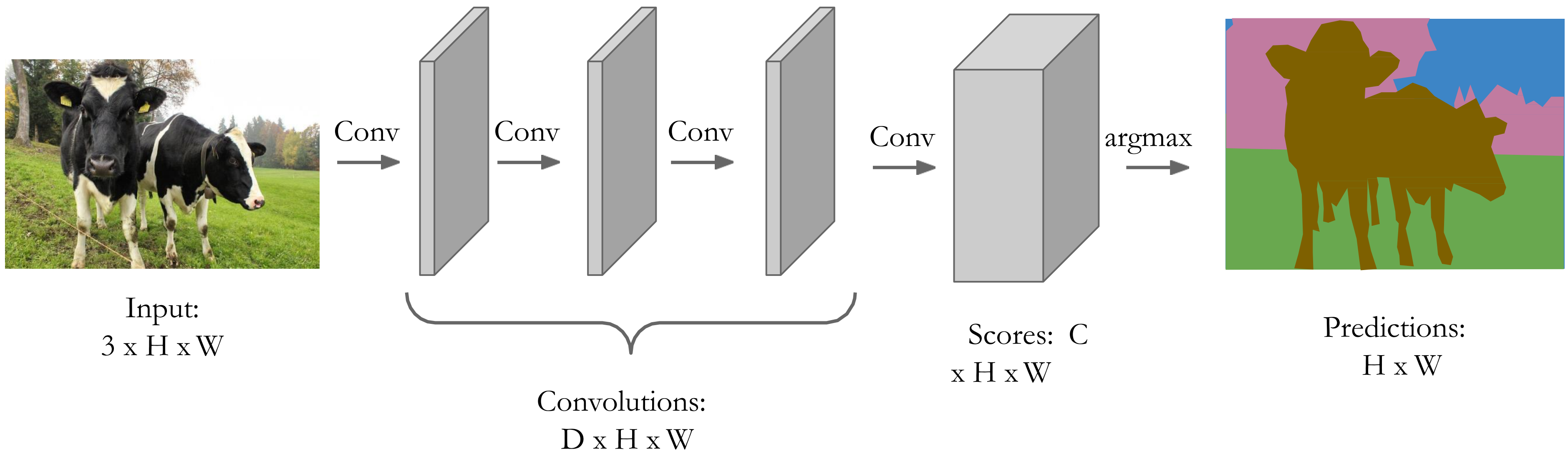
Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Slide by: Justin Johnson

Semantic Segmentation Idea: Fully Convolutional

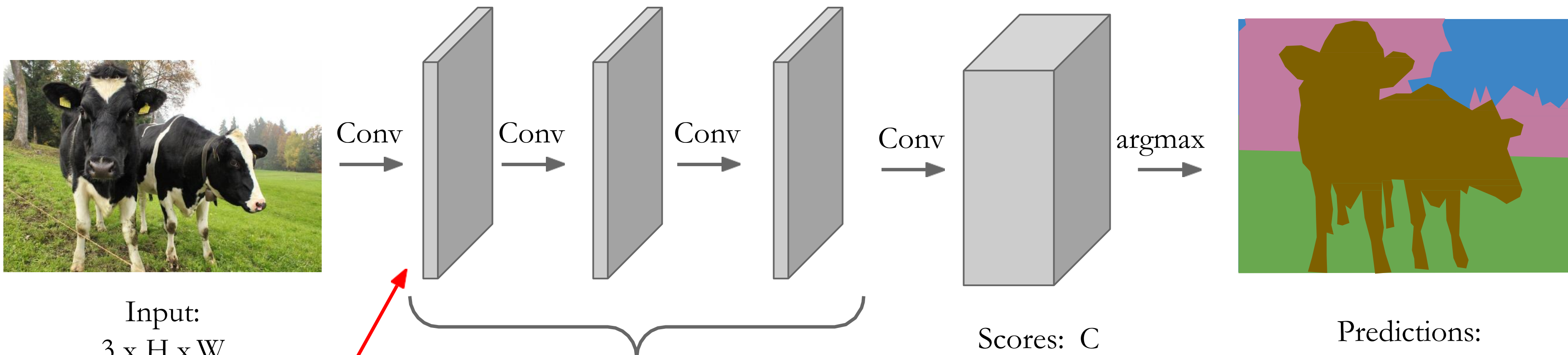
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Slide by: Justin Johnson

Semantic Segmentation Idea: Fully Convolutional

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



Input:
3 x H x W

Convolutions:
D x H x W

Scores: C x H x W

Predictions:
H x W

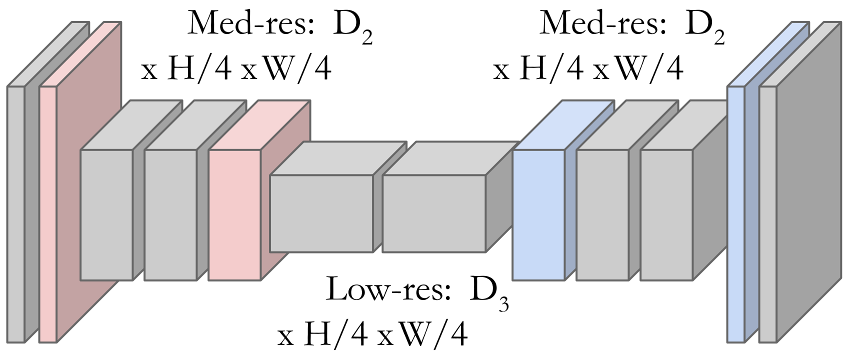
Problem: convolutions at original image resolution will be very expensive ...

Slide by: Justin Johnson

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input: 3 x H x W



Predictions: H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Slide by: Justin Johnson