# Lecture 20:
# Advanced recognition with DL

COS 429: Computer Vision

# Summary of training

1. **One time setup**
   *gradient checking*: do
   *activation functions*: use ReLU
   *data preprocessing*: subtract mean of the image
   *weight initialization*: use Xavier init
   *regularization*: use L2+dropout+data augmentation

2. **Training dynamics**
   *starting the learning process*: lots of sanity-checks
   *hyperparameter selection*: random sample in log space
   *parameter optimization*: use Adam
   *transfer learning*: use freely

3. **Evaluation**
   *model ensembles*: simple 2% boost

# CNN architectures

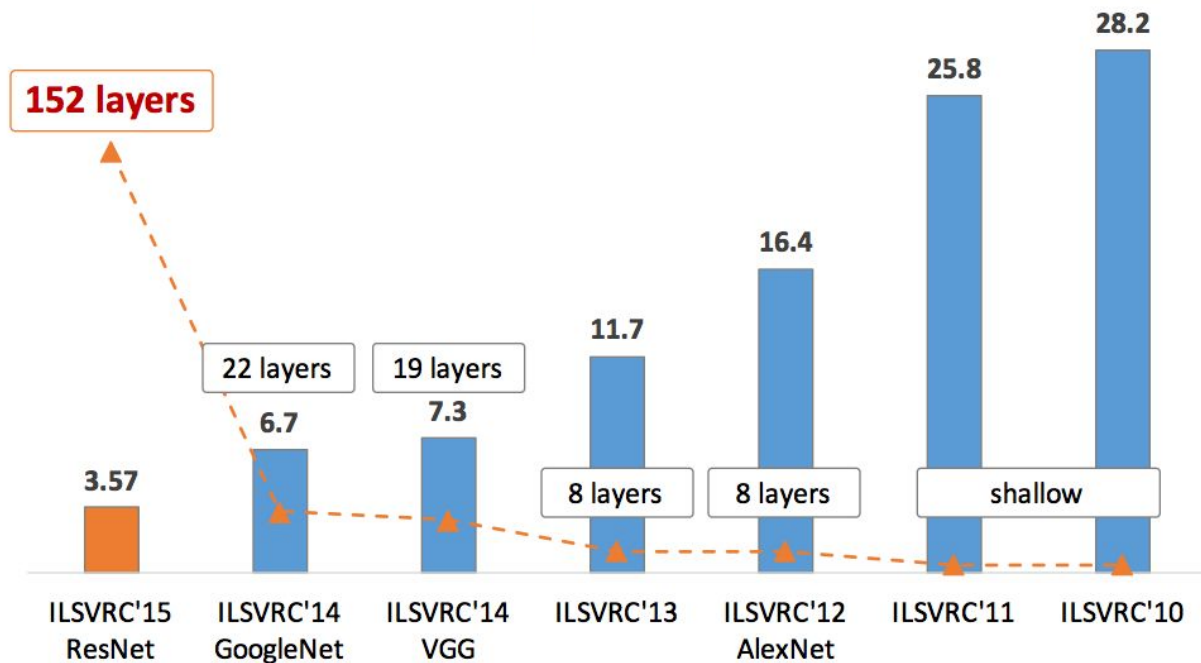# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



First CNN-based winner

152 layers

28.2

25.8

16.4

11.7

7.3

6.7

3.57

22 layers

19 layers

8 layers

8 layers

shallow

ILSVRC'15 ResNet

ILSVRC'14 GoogleNet

ILSVRC'14 VGG

ILSVRC'13

ILSVRC'12 AlexNet

ILSVRC'11

ILSVRC'10

Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
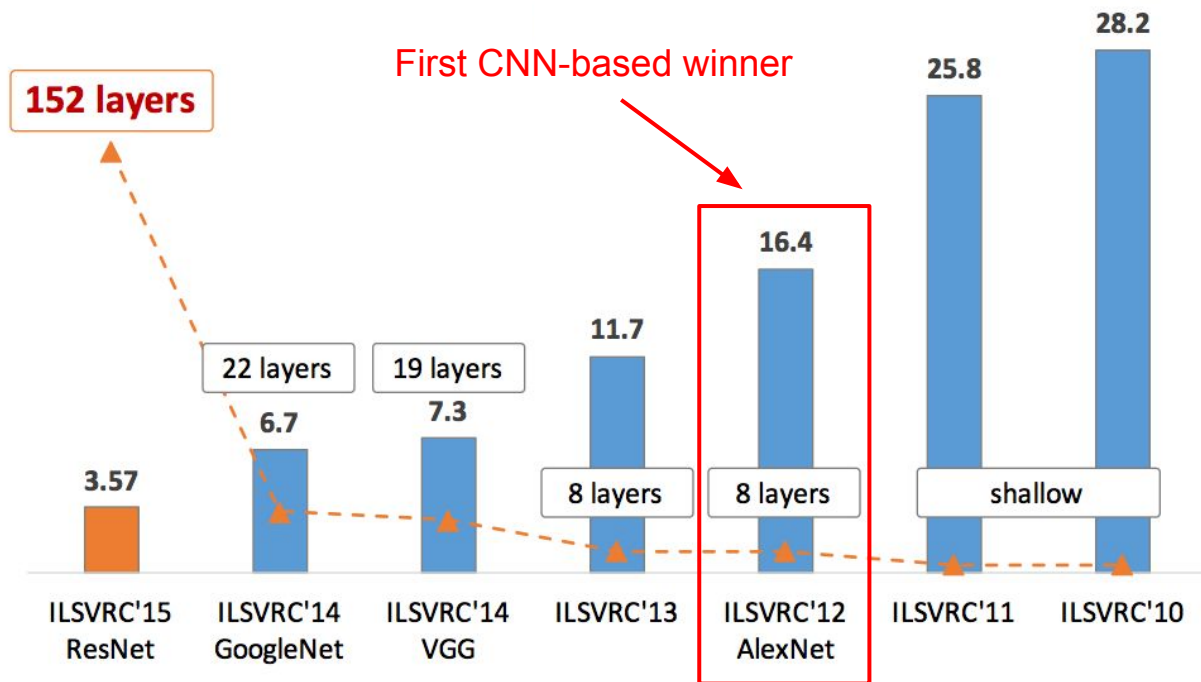
# Case Study: AlexNet

*[Krizhevsky et al. 2012]*



Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

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)

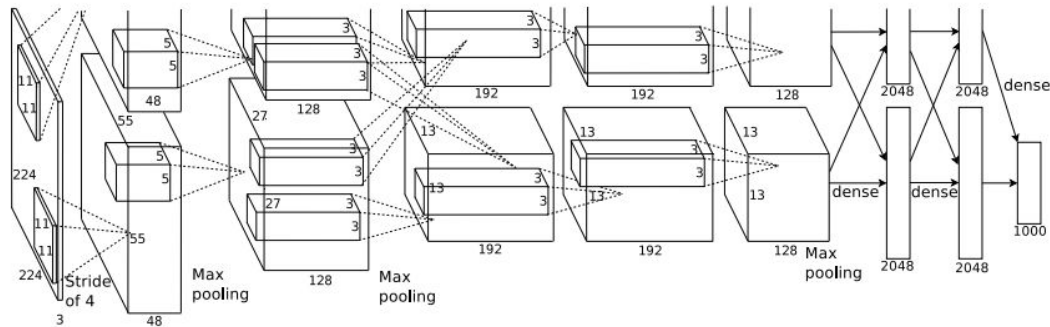# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
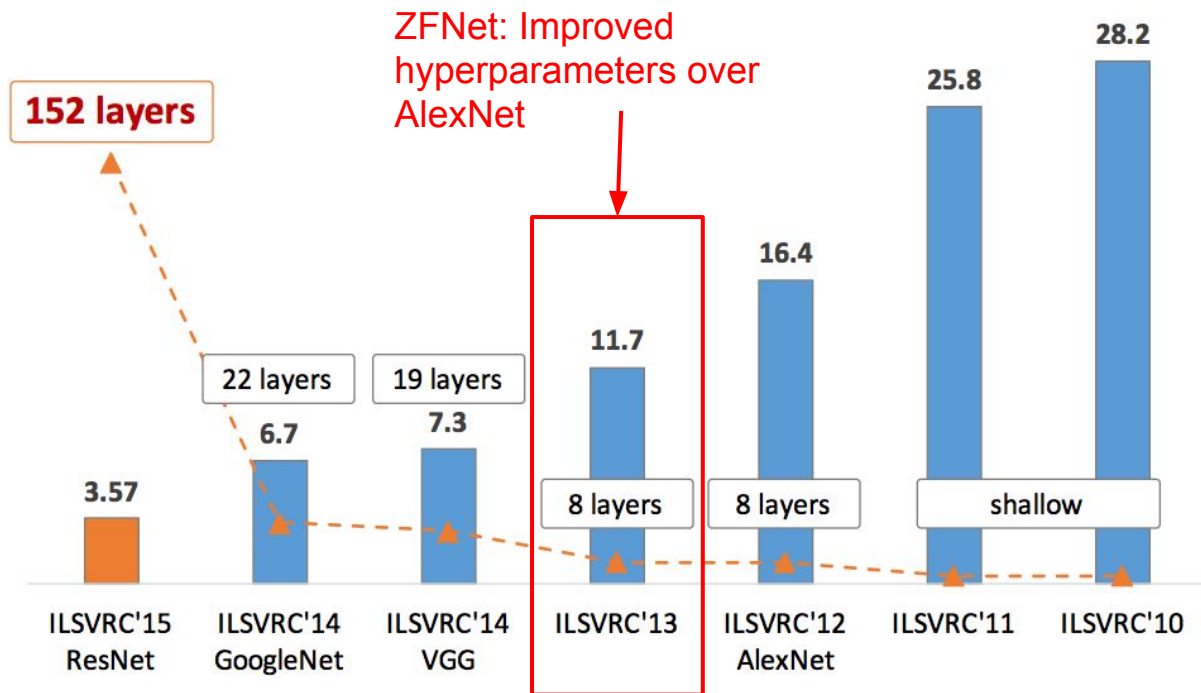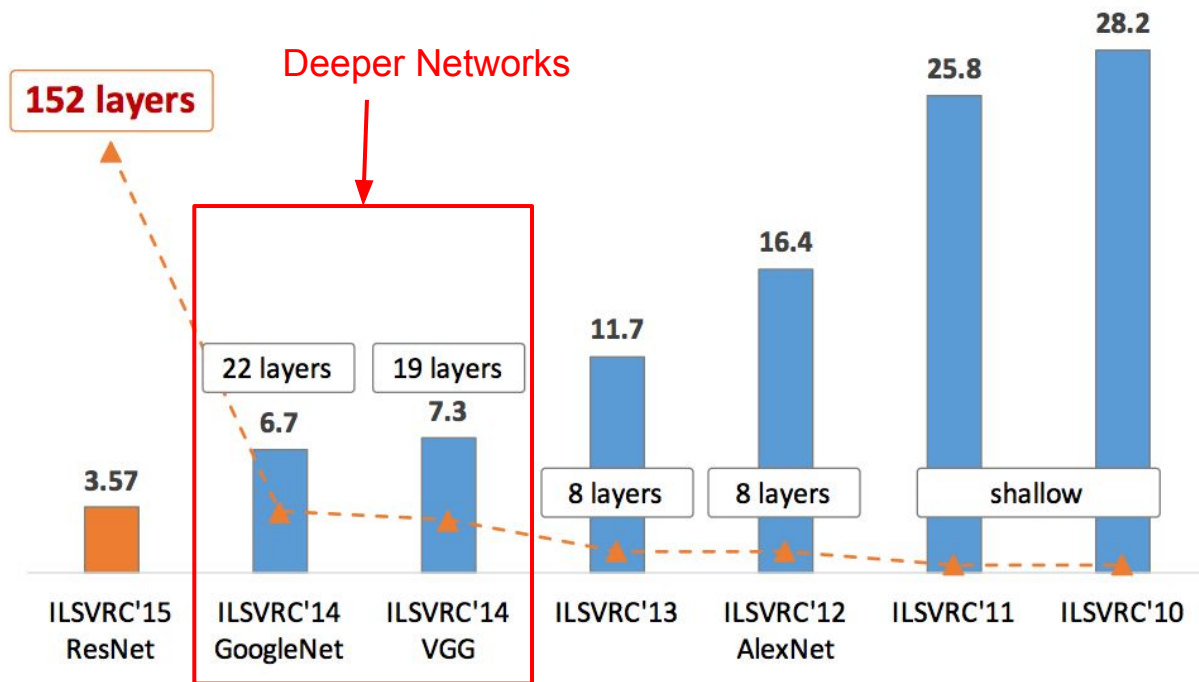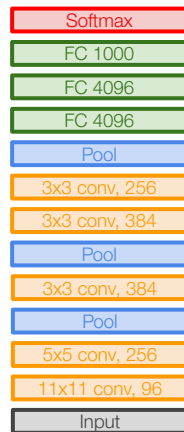
# Case Study: VGGNet

*[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



**AlexNet**

| |
|---|
| Softmax |
| FC 1000 |
| FC 4096 |
| FC 4096 |
| Pool |
| 3x3 conv, 256 |
| 3x3 conv, 384 |
| Pool |
| 3x3 conv, 384 |
| Pool |
| 5x5 conv, 256 |
| 11x11 conv, 96 |
| Input |

**VGG16**

| |
|---|
| Softmax |
| FC 1000 |
| FC 4096 |
| FC 4096 |
| Pool |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| Pool |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| Pool |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| Pool |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| Pool |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| Input |

**VGG19**

| |
|---|
| Softmax |
| FC 1000 |
| FC 4096 |
| FC 4096 |
| Pool |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| Pool |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| Pool |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| Pool |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| Pool |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| Input |

# Case Study: VGGNet

*[Simonyan and Zisserman, 2014]*

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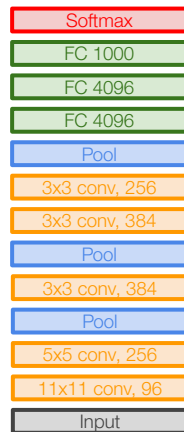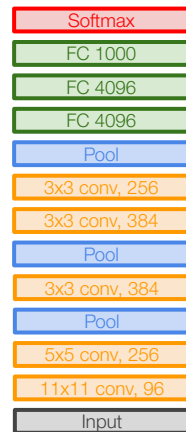


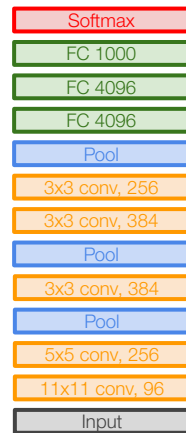AlexNet          VGG16          VGG19

# 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



AlexNet        VGG16        VGG19

# Case Study: VGGNet

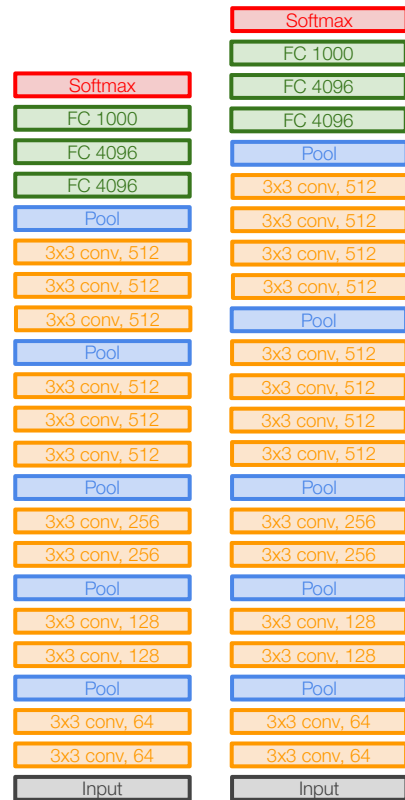*[Simonyan and Zisserman, 2014]*

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: $3 * (3^2 C^2)$ vs. $7^2 C^2$ for C channels per layer



AlexNet

VGG16

VGG19

INPUT: [224x224x3]        memory: 224*224*3=150K   params: 0        (not counting biases)
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 (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Softmax
FC 1000        fc8
FC 4096        fc7
FC 4096        fc6
Pool
3x3 conv, 512        conv5-3
3x3 conv, 512        conv5-2
3x3 conv, 512        conv5-1
Pool
3x3 conv, 512        conv4-3
3x3 conv, 512        conv4-2
3x3 conv, 512        conv4-1
Pool
3x3 conv, 256        conv3-2
3x3 conv, 256        conv3-1
Pool
3x3 conv, 128        conv2-2
3x3 conv, 128        conv2-1
Pool
3x3 conv, 64        conv1-2
3x3 conv, 64        conv1-1
Input

VGG16

Common names

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
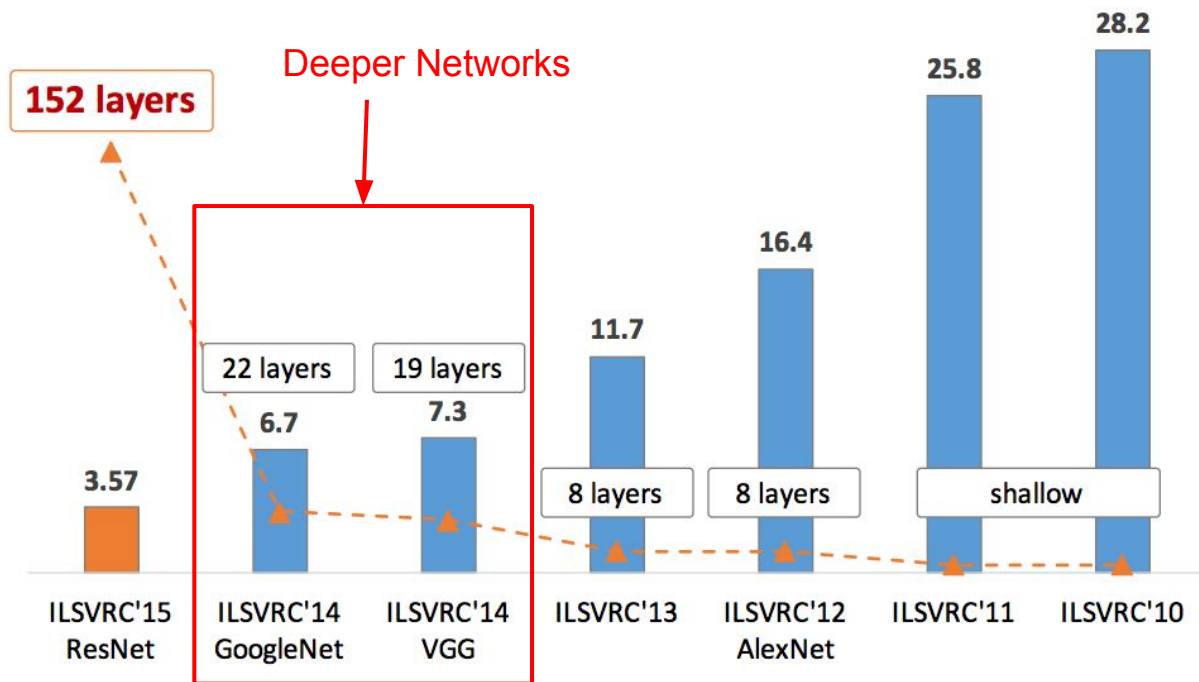


Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

"Revolution of Depth"



Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
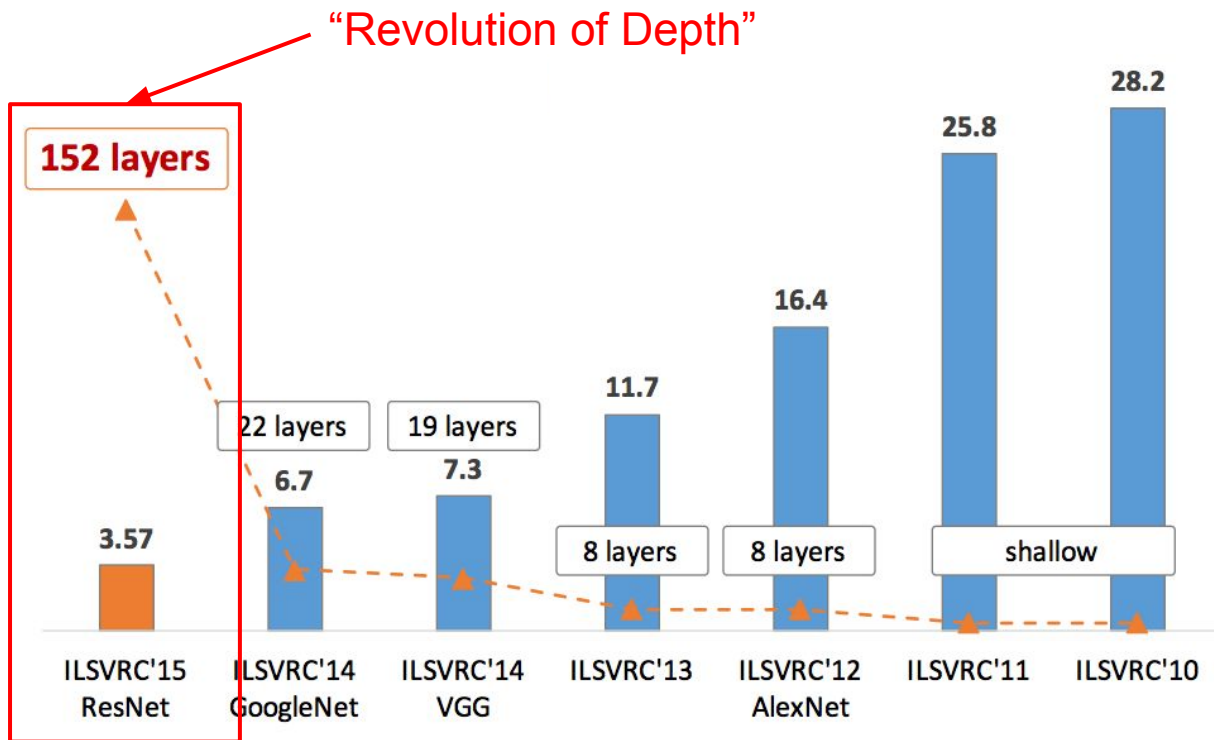
# Case Study: ResNet

*[He et al., 2015]*

**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!
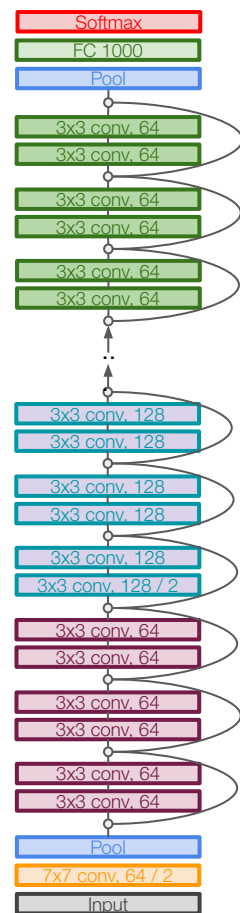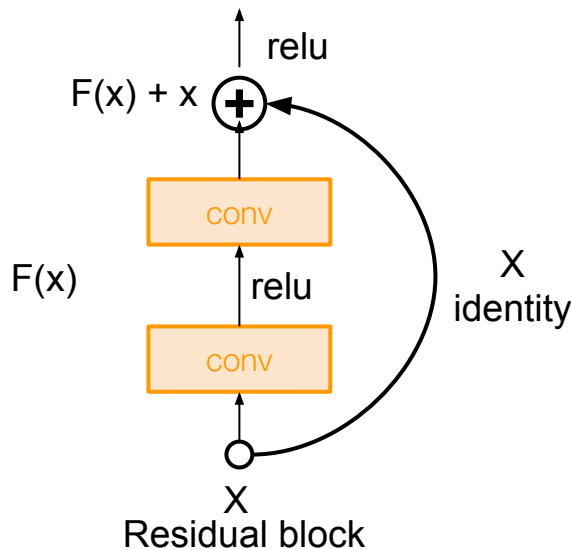


Residual block

# Case Study: ResNet

*[He et al., 2015]*

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

# Case Study: ResNet

*[He et al., 2015]*

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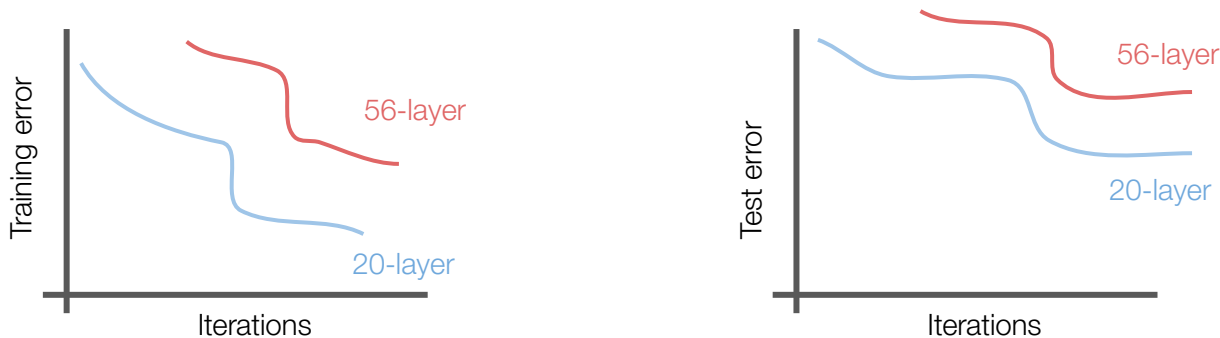


Q: What's strange about these training and test curves?

# Case Study: ResNet

*[He et al., 2015]*

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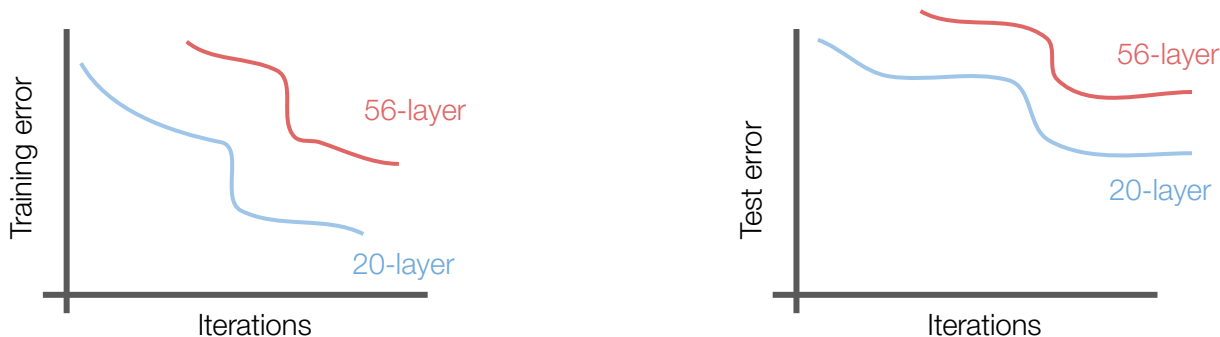


56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it's not caused by overfitting!

# Case Study: ResNet

*[He et al., 2015]*

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

# Case Study: ResNet

*[He et al., 2015]*

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., 2015]*

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



"Plain" layers

Residual block

# Case Study: ResNet

*[He et al., 2015]*

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

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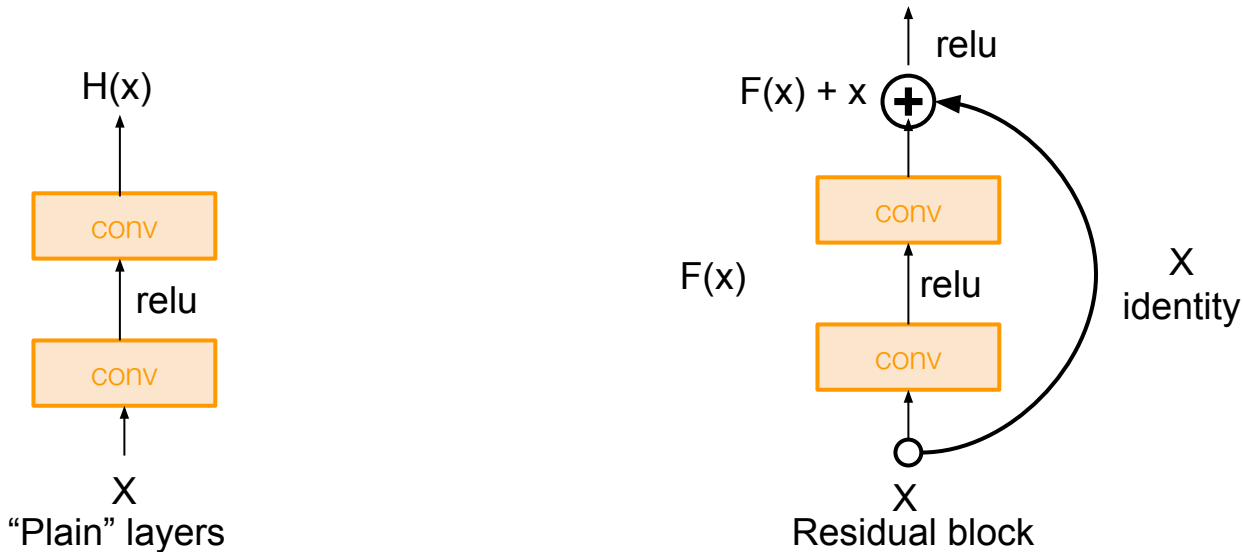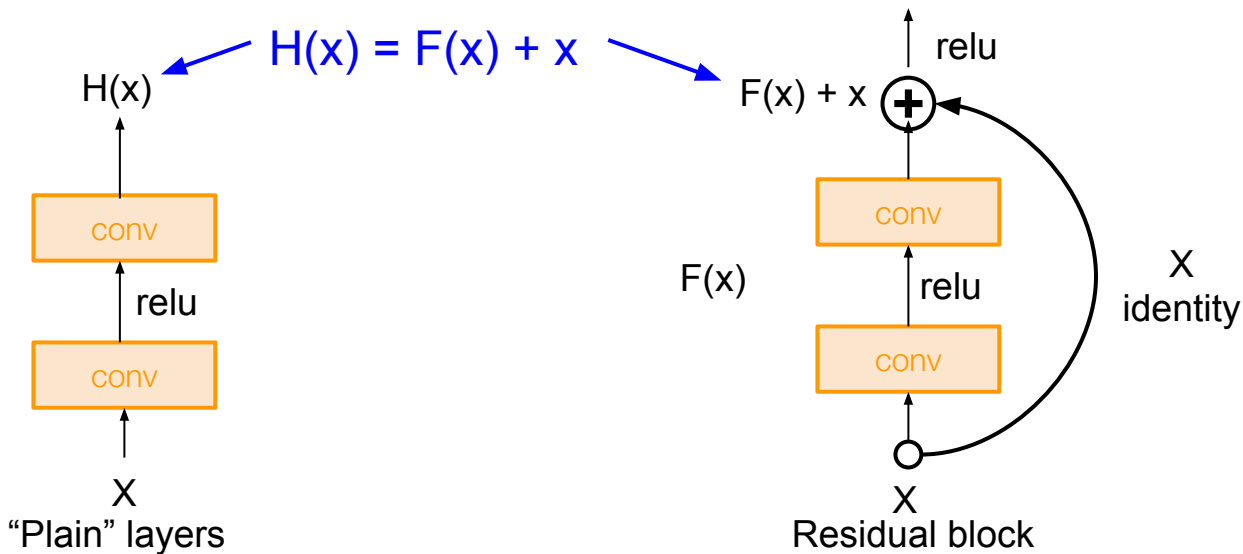
Use layers to fit residual $F(x) = H(x) - x$ instead of $H(x)$ directly



"Plain" layers

Residual block

# Case Study: ResNet

*[He et al., 2015]*

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers



Residual block

# Case Study: ResNet

*[He et al., 2015]*

Full ResNet architecture:
-   Stack residual blocks
-   Every residual block has two 3x3 conv layers
-   Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
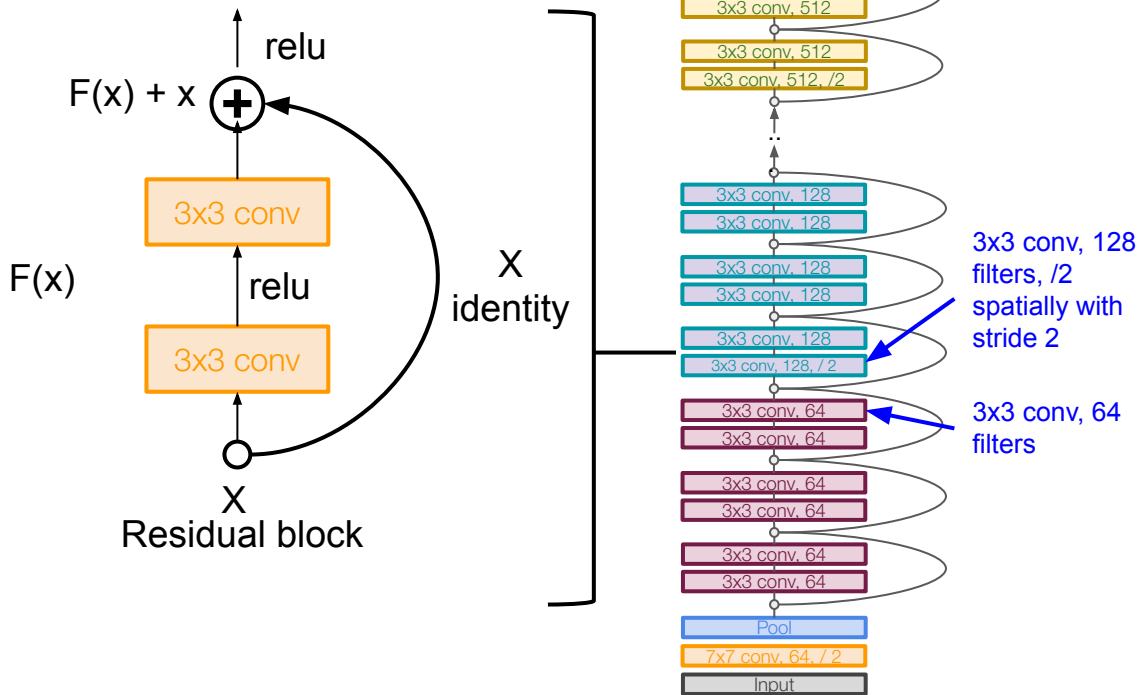


Residual block

# Case Study: ResNet

*[He et al., 2015]*

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning

# Case Study: ResNet

*[He et al., 2015]*

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)



F(x) + x

relu

3x3 conv

F(x)  relu

3x3 conv

X

X identity

X
Residual block

No FC layers besides FC 1000 to output classes

Global average pooling layer after last conv layer

Softmax
FC 1000
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, /2
...
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
Input

# Case Study: ResNet

*[He et al., 2015]*

Total depths of 34, 50, 101, or 152 layers for ImageNet



| Softmax |
| FC 1000 |
| Pool |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512, /2 |
| ⋮ |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128, / 2 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| Pool |
| 7x7 conv, 64, / 2 |
| Input |

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Figure copyright Kaiming He, 2016. Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Comparing complexity...



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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Now: Segmentation, Localization, Detection

# So far: Image Classification



This image is CC0 public domain
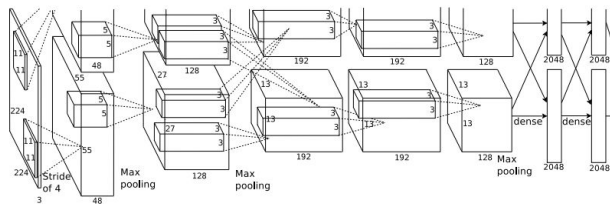
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

**Vector:**
4096

**Fully-Connected**:
4096 to 1000

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

# Other Computer Vision Tasks



**Semantic Segmentation**
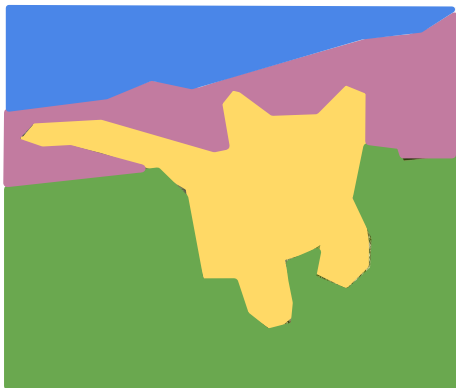
**GRASS**, **CAT**, **TREE**, **SKY**

No objects, just pixels

**Classification + Localization**
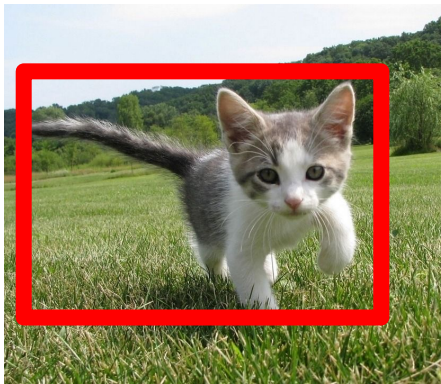
**CAT**

Single Object

**Object Detection**
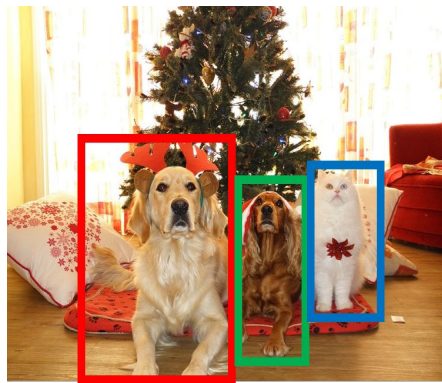
**DOG**, **DOG**, **CAT**

**Instance Segmentation**

**DOG**, **DOG**, **CAT**
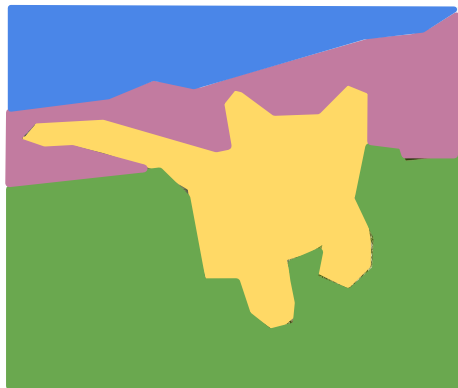
Multiple Object

This image is CC0 public domain

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Semantic Segmentation

GRASS, CAT, TREE, SKY
No objects, just pixels

CAT
Single Object

DOG, DOG, CAT
DOG, DOG, CAT
Multiple Object

This image is CC0 public domain

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
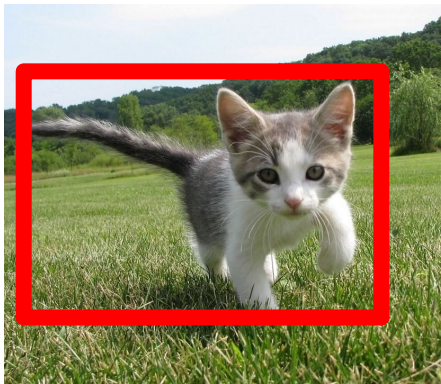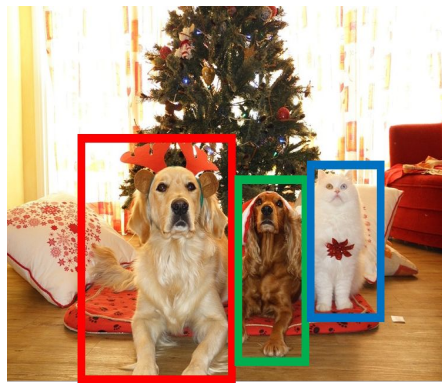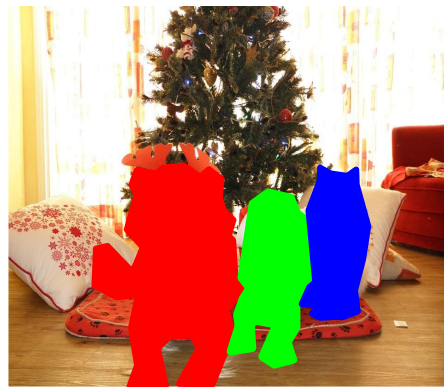
# Semantic Segmentation



This image is CC0 public domain

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels
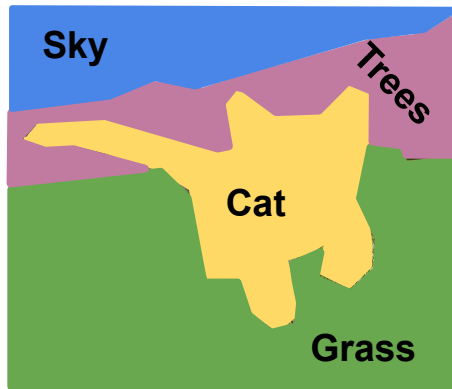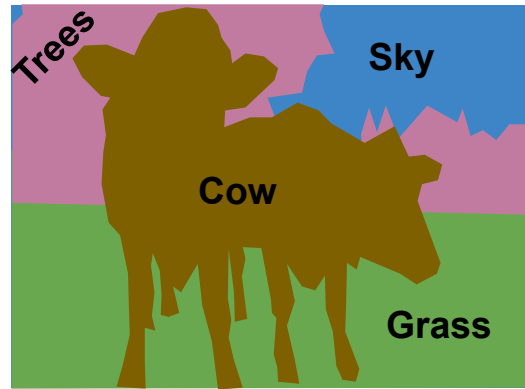
# Semantic Segmentation Idea: Sliding Window



Extract patch    Classify center pixel with CNN

Full image

Cow

Cow

Grass

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

# Semantic Segmentation Idea: Sliding Window

Full image

Extract patch

Classify center pixel with CNN



Cow

Cow

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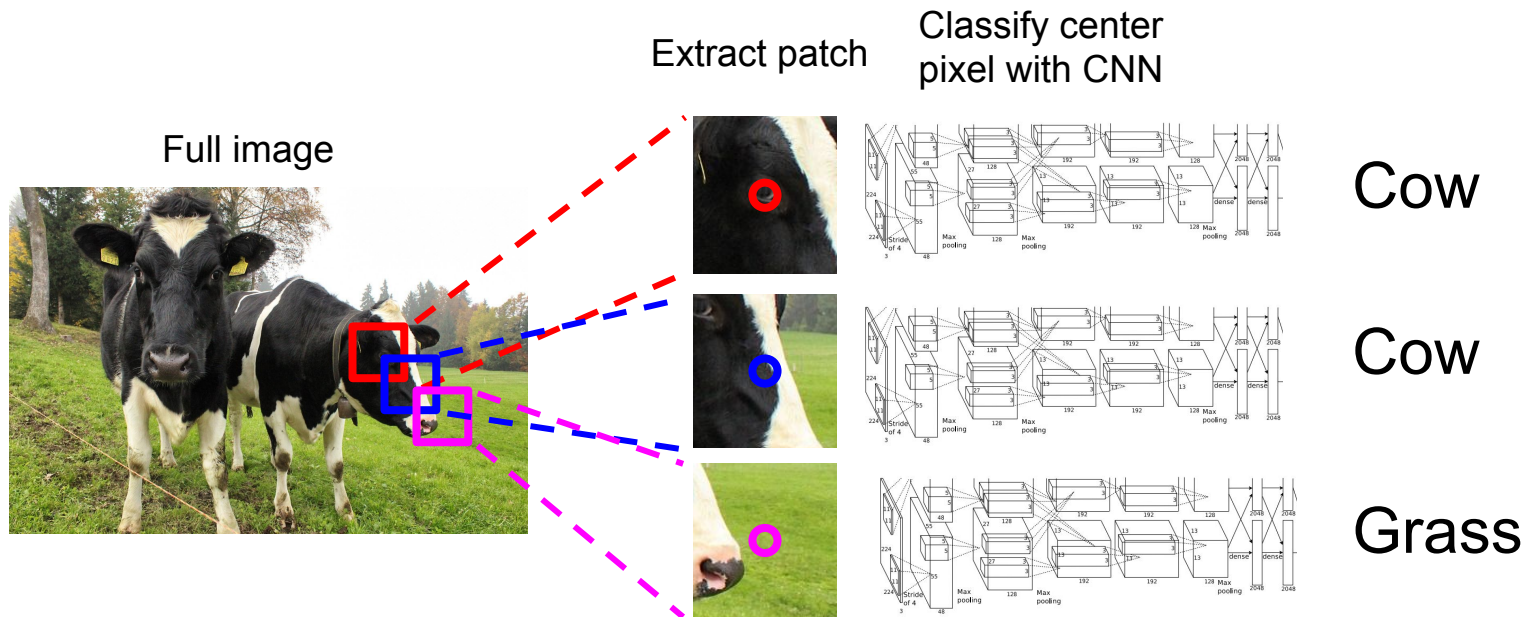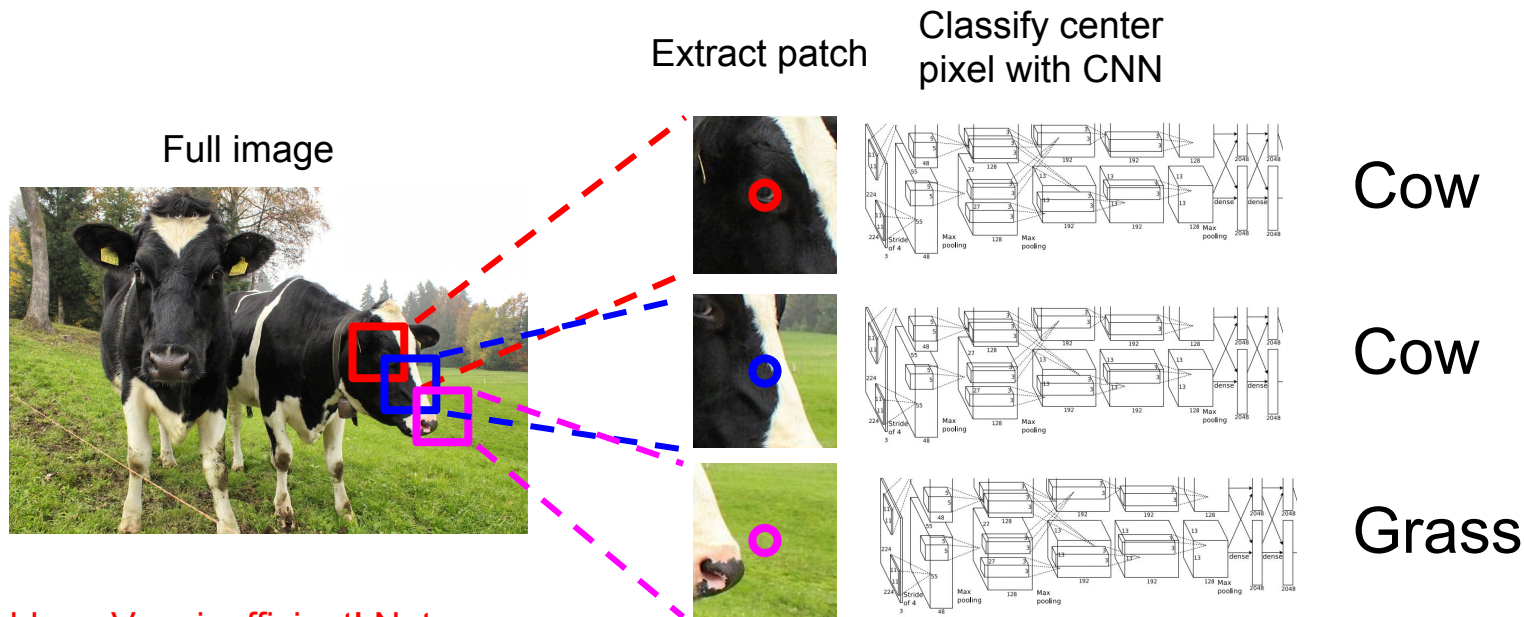
Problem: Very inefficient! Not reusing shared features between overlapping patches

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

# 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

Conv    Conv    Conv    Conv    argmax

Convolutions:
D x H x W

Scores:
C x H x W

Predictions:
H x W

# 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

Conv    Conv    Conv    Conv    argmax

Convolutions:
D x H x W

Scores:
C x H x W
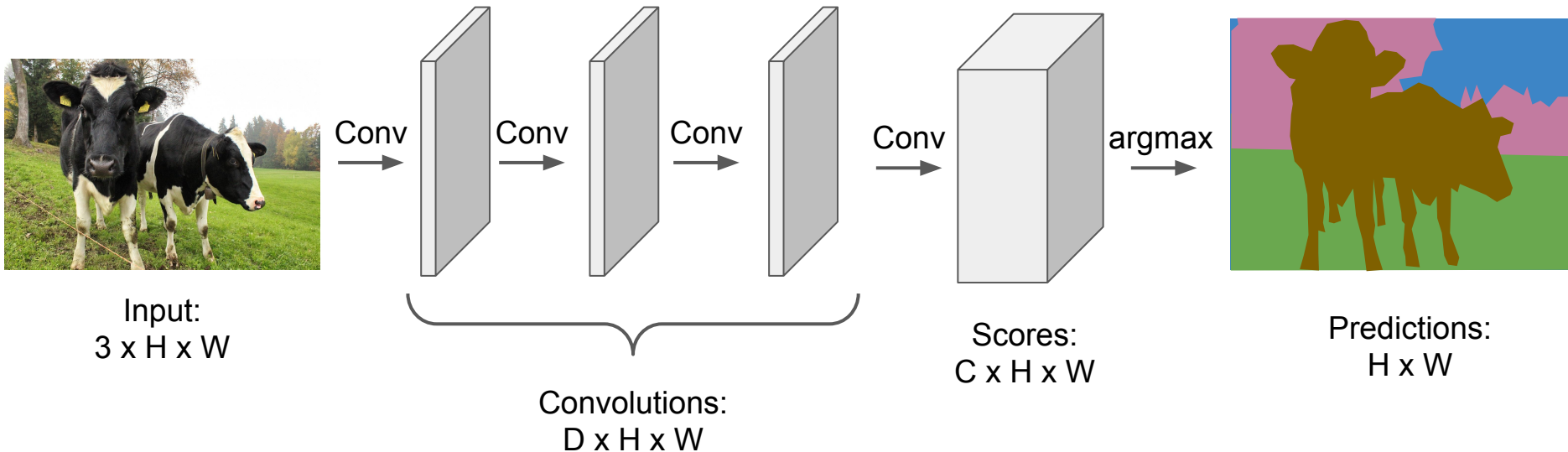
Predictions:
H x W

Problem: convolutions at original image resolution will be very expensive ...
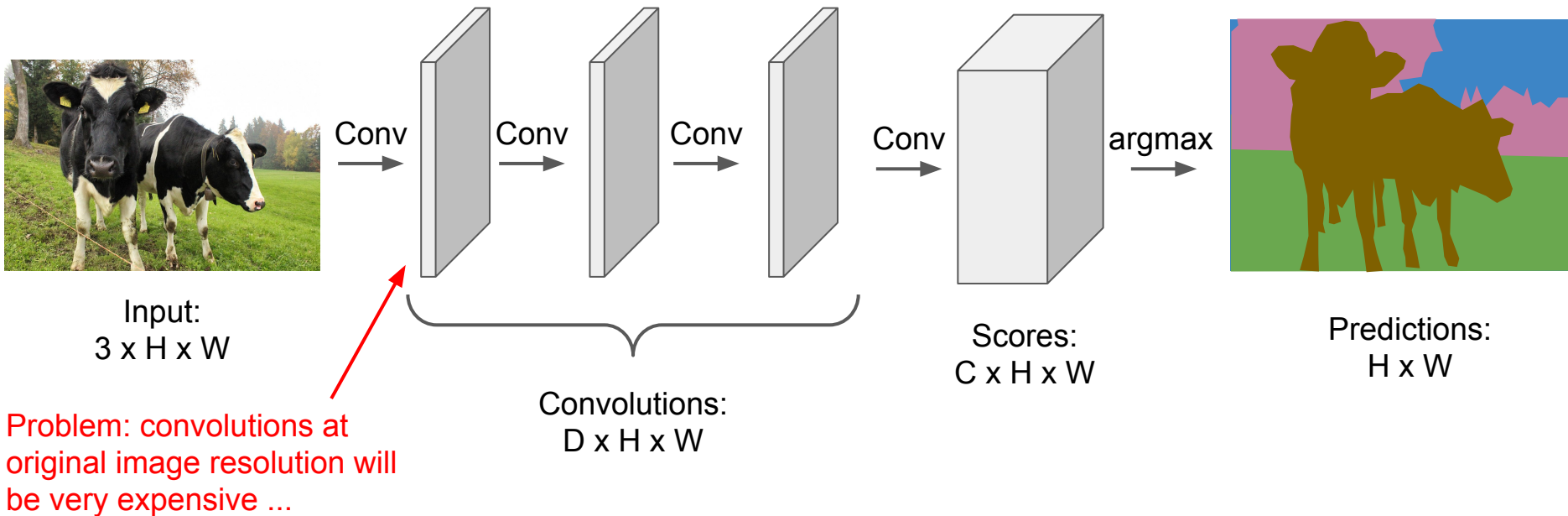
# Semantic Segmentation Idea: Fully Convolutional

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



Input:
3 x H x W

High-res:
$D_1$ x H/2 x W/2

Med-res:
$D_2$ x H/4 x W/4

Low-res:
$D_3$ x H/4 x W/4

Med-res:
$D_2$ x H/4 x W/4

High-res:
$D_1$ x H/2 x W/2

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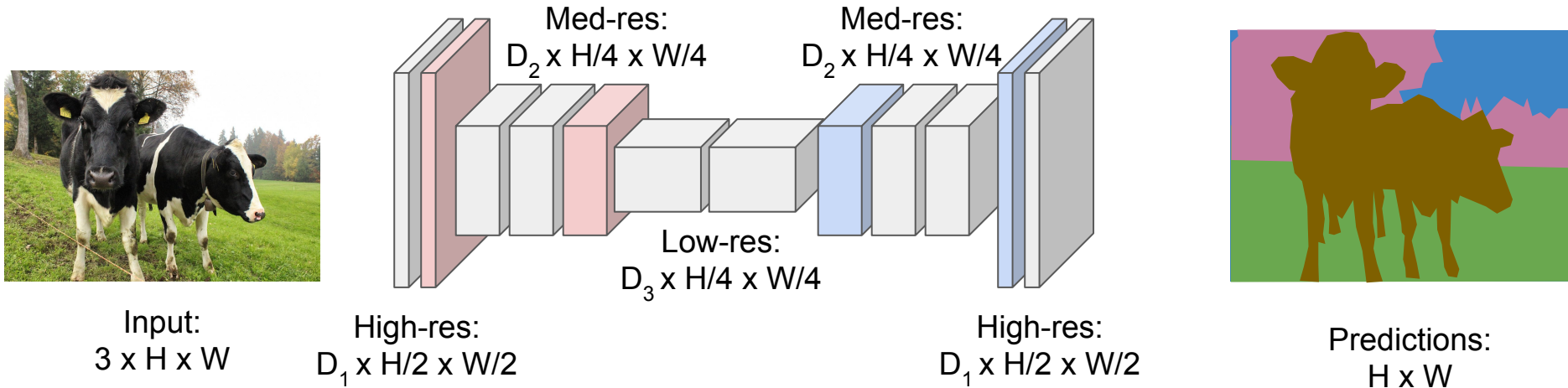
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

# Semantic Segmentation Idea: Fully Convolutional

**Downsampling**:
Pooling, strided convolution

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

**Upsampling**:
???



Input:
$3 \times H \times W$

High-res:
$D_1 \times H/2 \times W/2$

Med-res:
$D_2 \times H/4 \times W/4$

Low-res:
$D_3 \times H/4 \times W/4$

Med-res:
$D_2 \times H/4 \times W/4$

High-res:
$D_1 \times H/2 \times W/2$
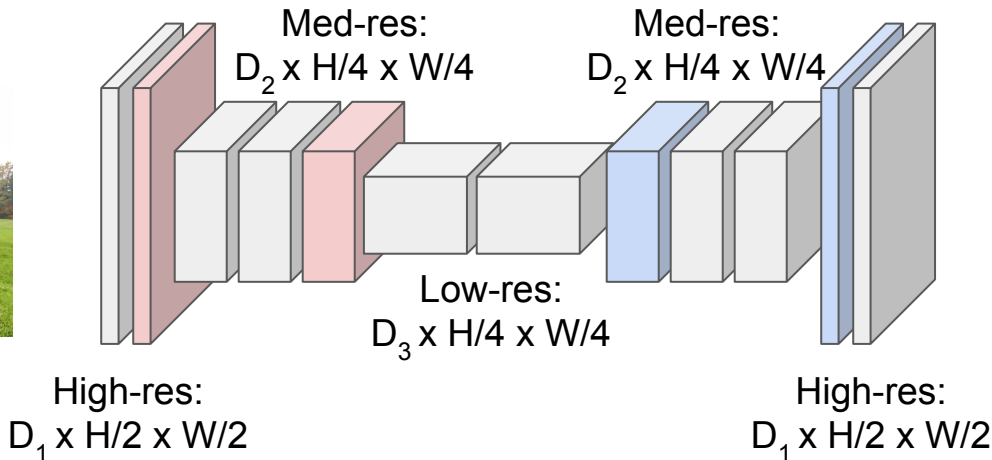
Predictions:
$H \times W$

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

# In-Network upsampling: "Unpooling"

**Nearest Neighbor**

| 1 | 2 |
|---|---|
| 3 | 4 |

→

| 1 | 1 | 2 | 2 |
|---|---|---|---|
| 1 | 1 | 2 | 2 |
| 3 | 3 | 4 | 4 |
| 3 | 3 | 4 | 4 |

Input: 2 x 2          Output: 4 x 4

**"Bed of Nails"**

| 1 | 2 |
|---|---|
| 3 | 4 |

→

| 1 | 0 | 2 | 0 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 3 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 |

Input: 2 x 2          Output: 4 x 4

# In-Network 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 |

→

| 5 | 6 |
|---|---|
| 7 | 8 |

→ . . . → Rest of the network

Input: 4 x 4          Output: 2 x 2

**Max Unpooling**
Use positions from pooling layer

| 1 | 2 |
|---|---|
| 3 | 4 |

→

| 0 | 0 | 2 | 0 |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 4 |

Input: 2 x 2          Output: 4 x 4
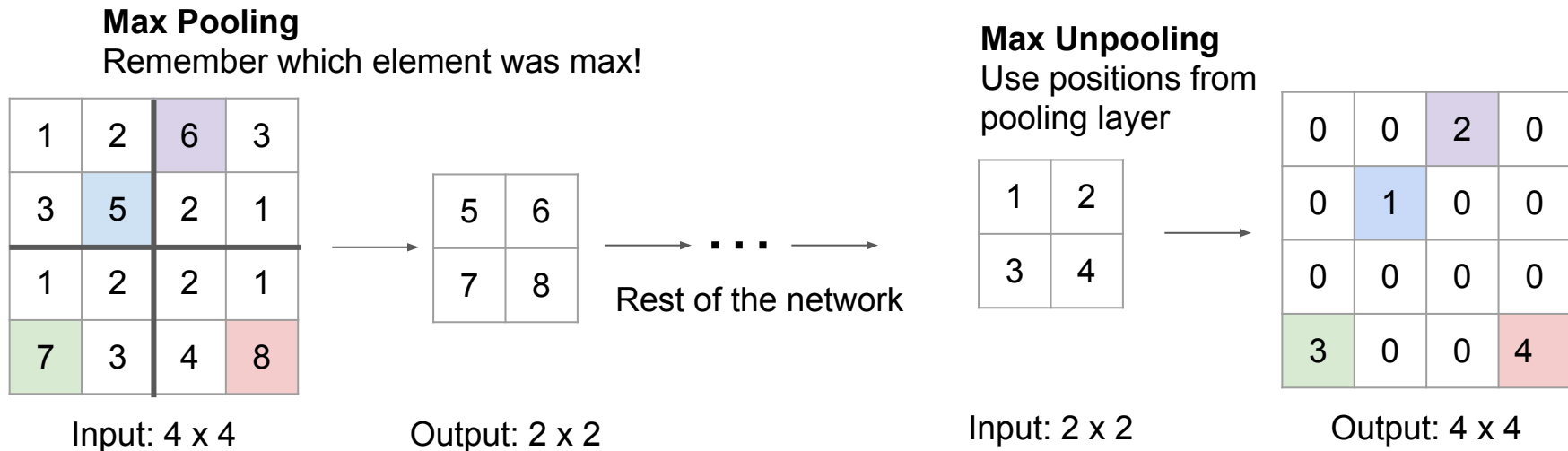
Corresponding pairs of downsampling and upsampling layers
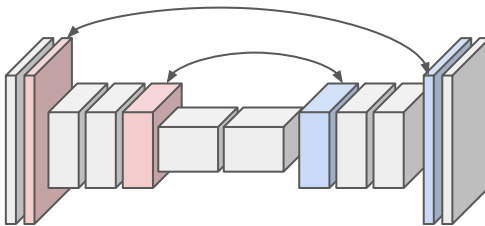
# Learnable Upsampling: Transpose Convolution

**Recall:** Typical 3 x 3 convolution, stride 1 pad 1
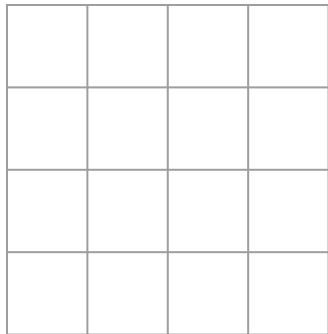


Input: 4 x 4
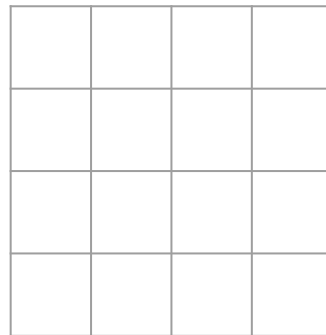


Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1

Dot product between filter and input

Input: 4 x 4

Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1



Dot product
between filter
and input
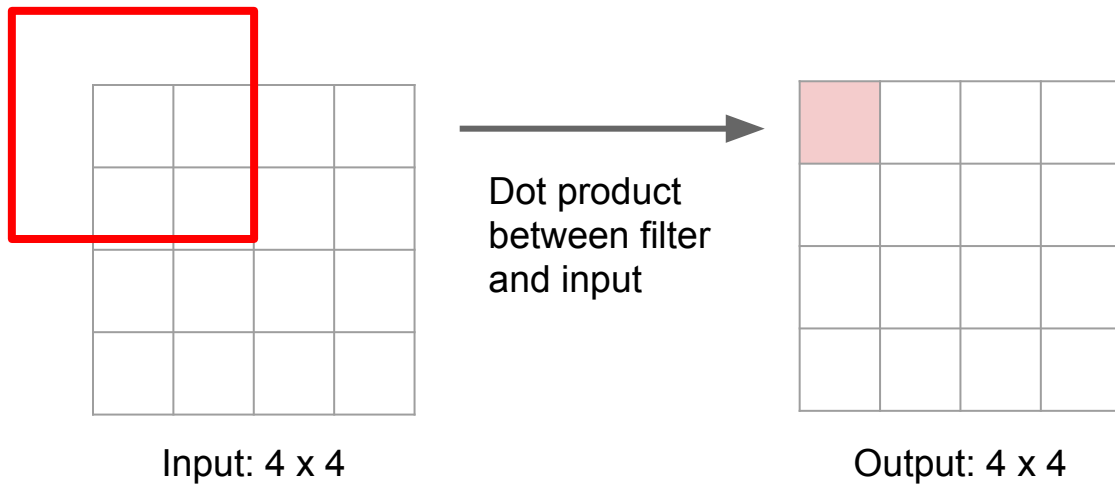
Input: 4 x 4
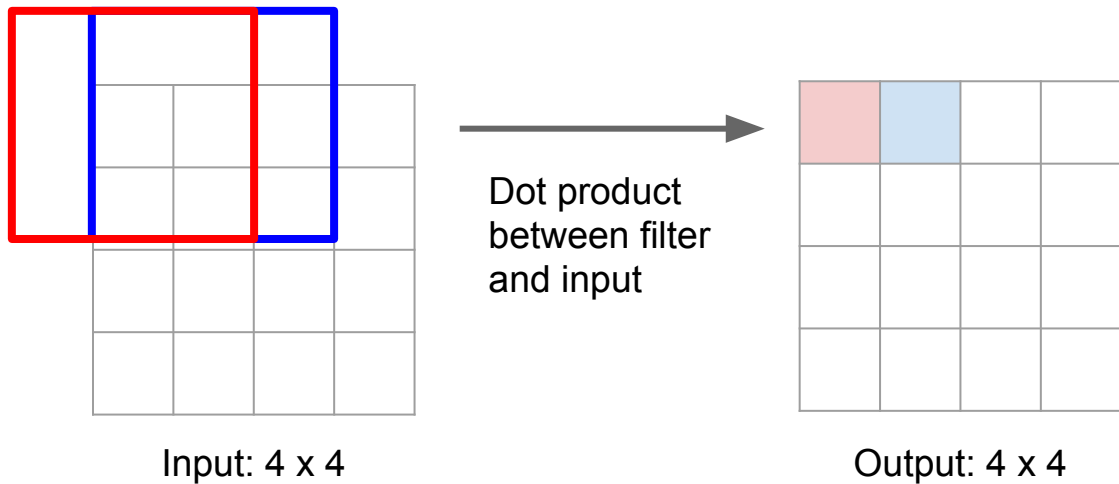
Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, <u>stride 2</u> pad 1

Input: 4 x 4

Output: 2 x 2
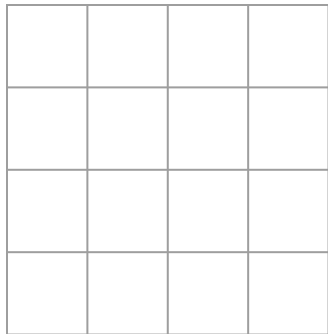
# Learnable Upsampling: Transpose Convolution
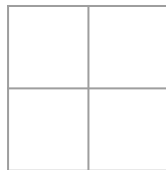
**Recall:** Normal 3 x 3 convolution, <u>stride 2</u> pad 1

Dot product
between filter
and input
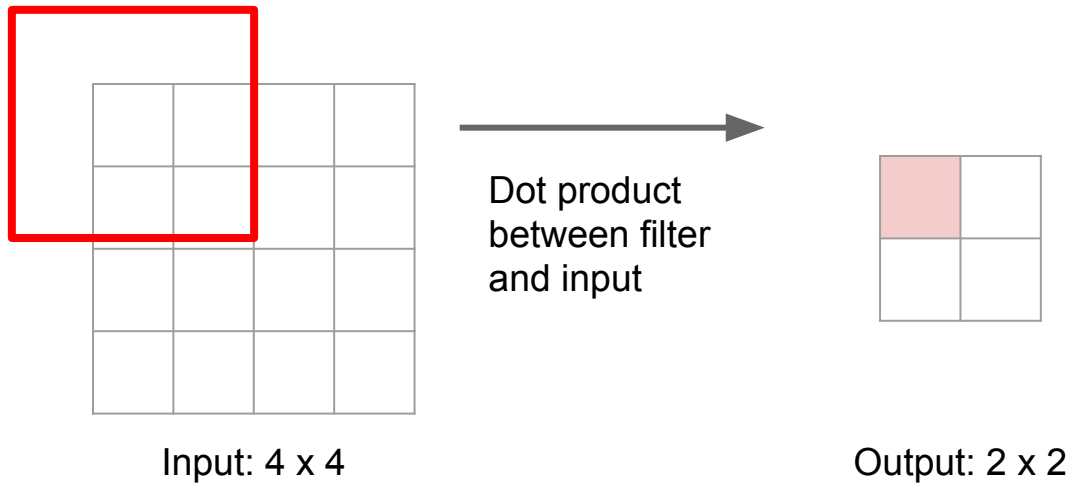
Input: 4 x 4

Output: 2 x 2

# Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, <u>stride 2</u> pad 1
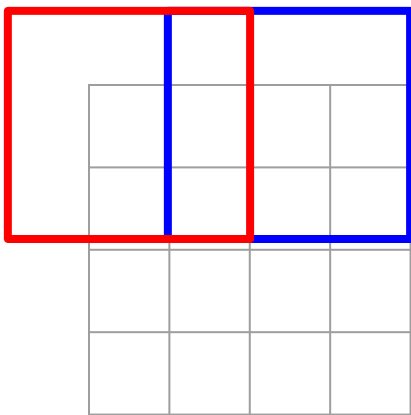


Dot product between filter and input
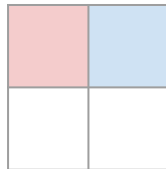
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Input: 4 x 4

Output: 2 x 2

# Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

Input: 2 x 2
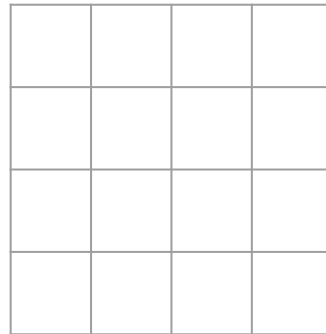
Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Input gives weight for filter

Input: 2 x 2

Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

Sum where output overlaps

Input gives weight for filter

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

Input: 2 x 2

Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Sum where output overlaps

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

Input gives weight for filter

Input: 2 x 2

Output: 4 x 4
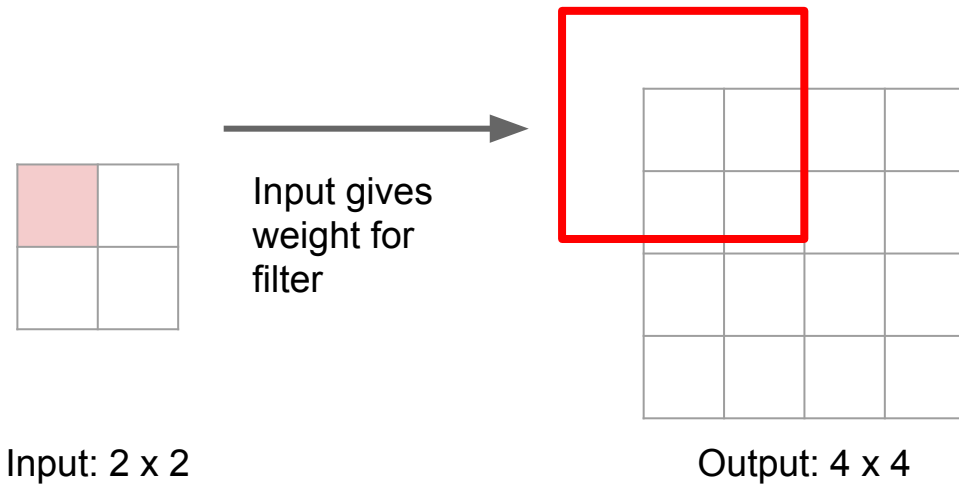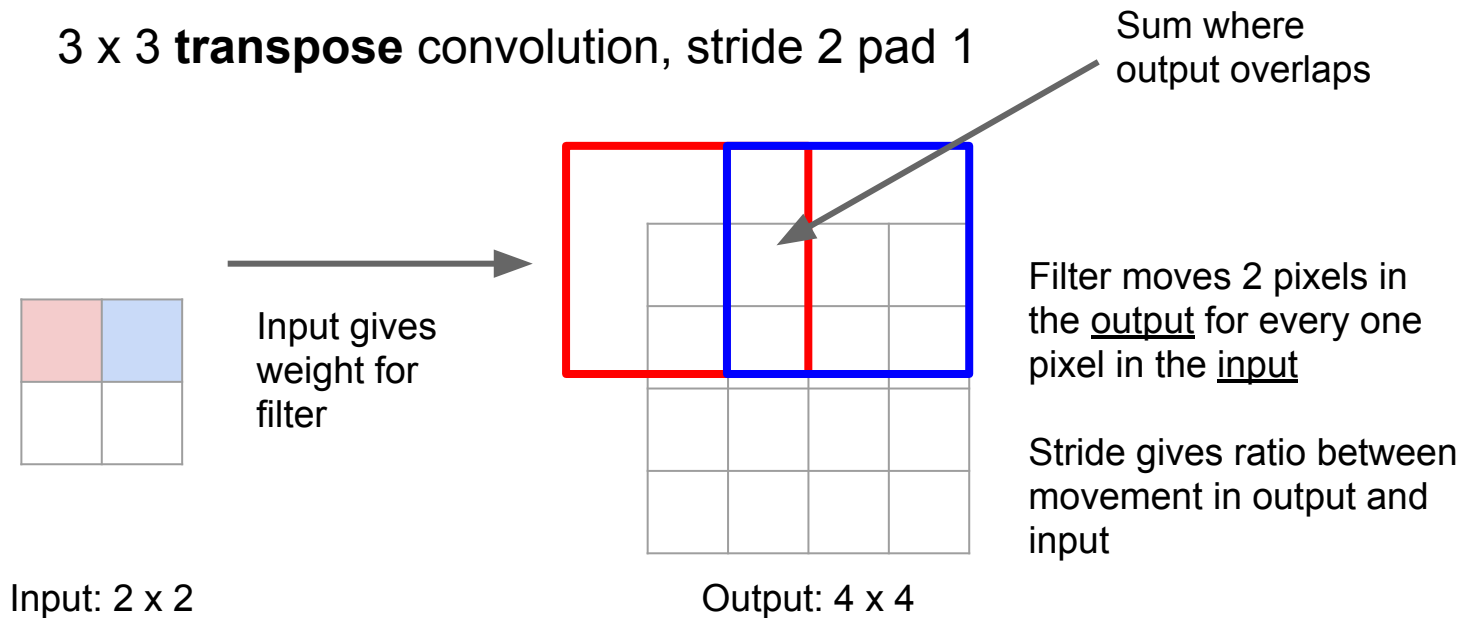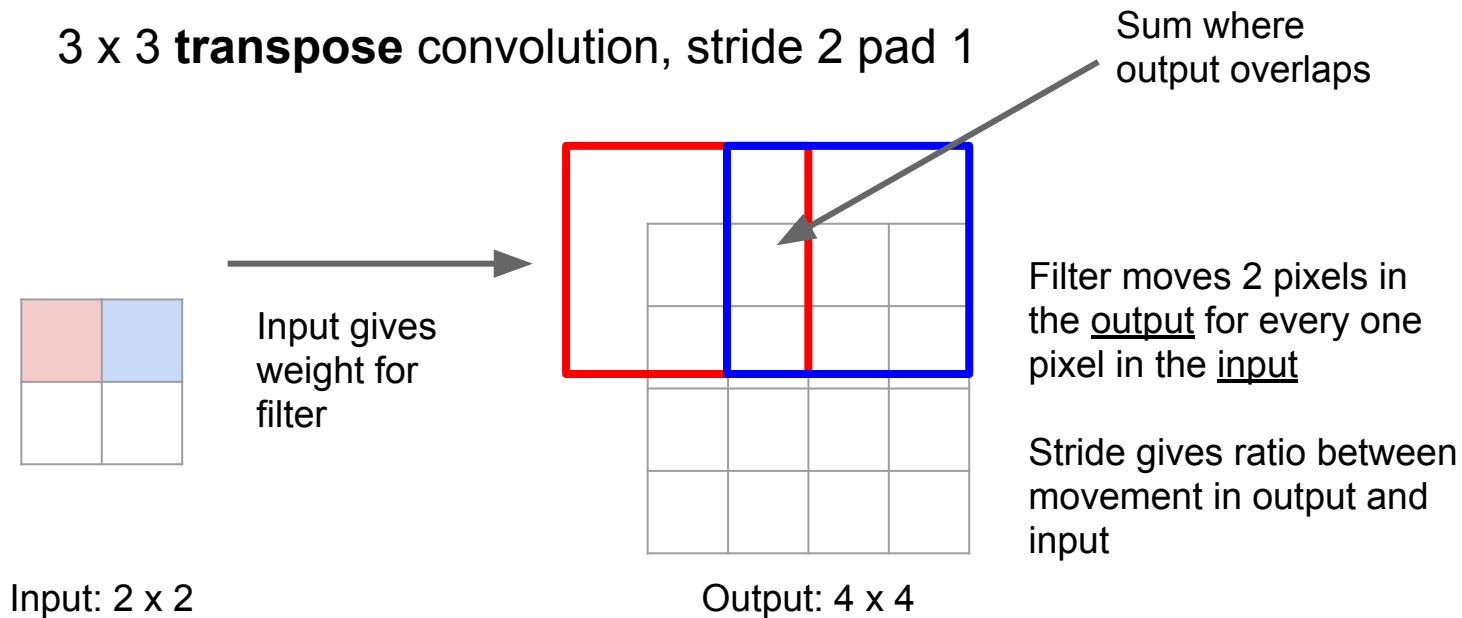
# Learnable Upsampling: Transpose Convolution

**Other names:**
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1

Sum where output overlaps

Input gives weight for filter

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

Input: 2 x 2

Output: 4 x 4
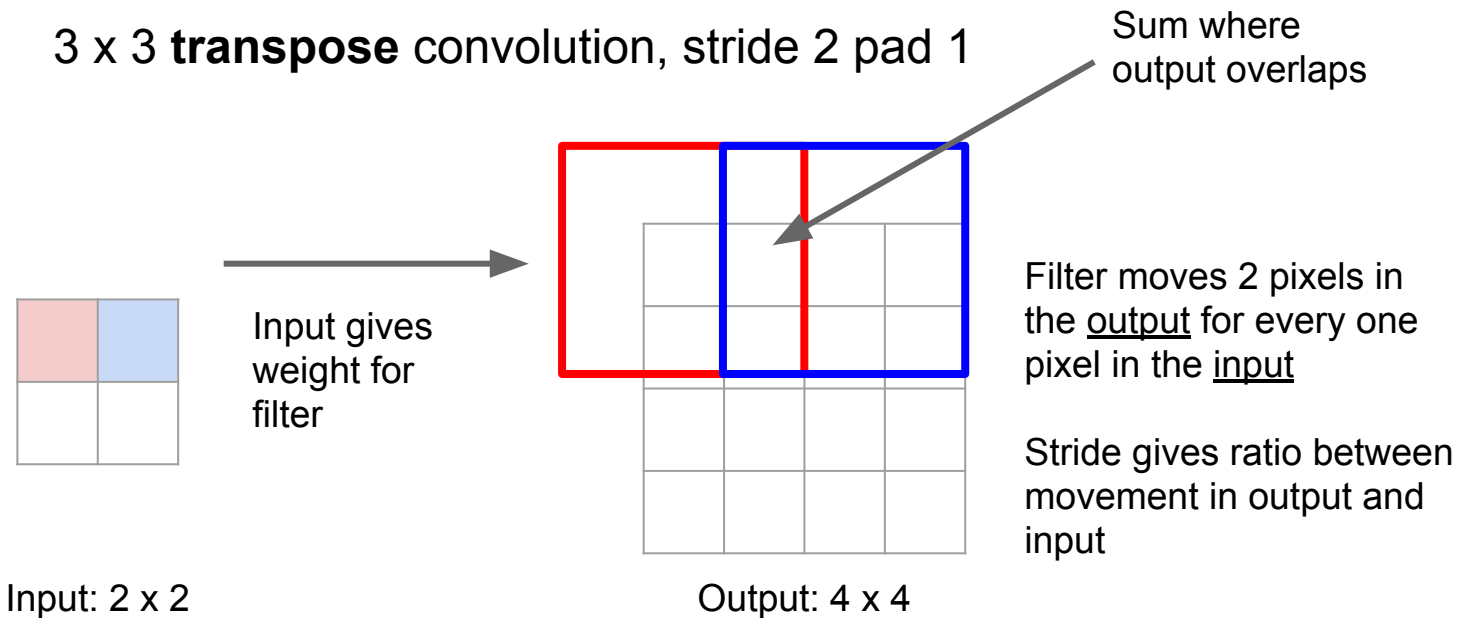
# Transpose Convolution: 1D Example

**Input**
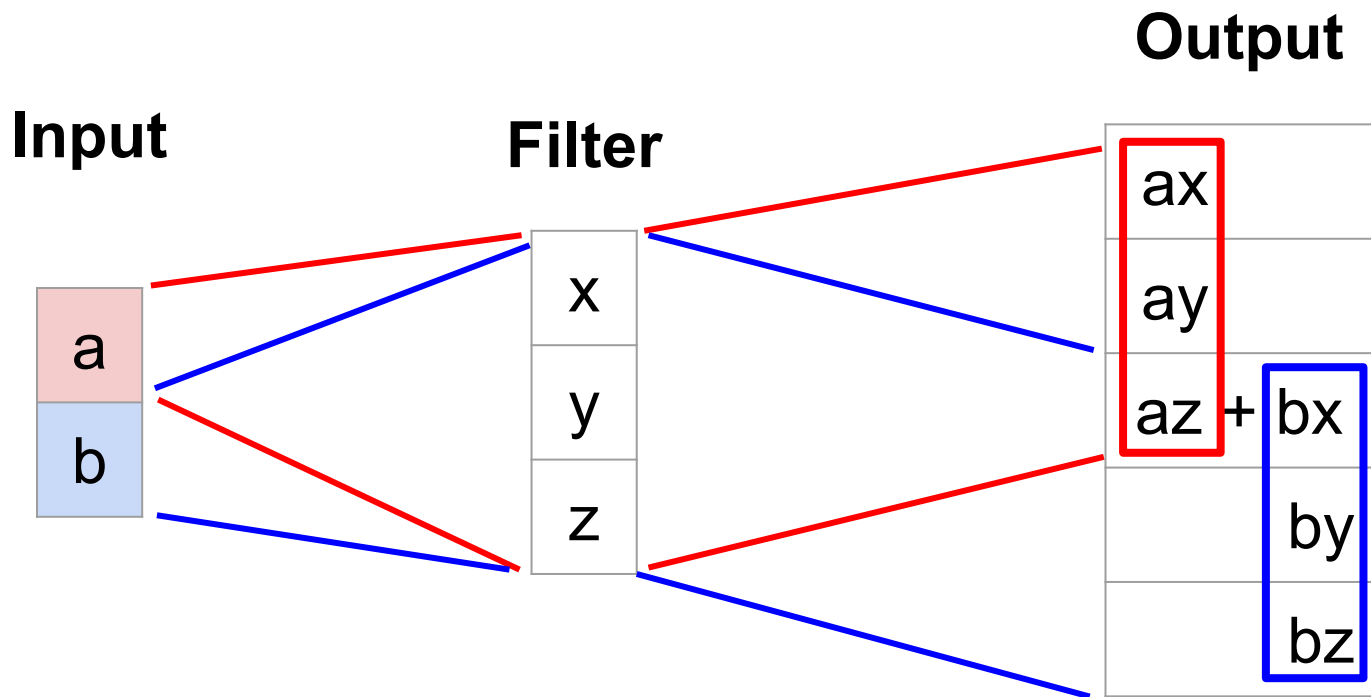
**Filter**

**Output**



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

# Semantic Segmentation Idea: Fully Convolutional

**Downsampling**:
Pooling, strided convolution



Input:
3 x H x W

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



Med-res:
$D_2$ x H/4 x W/4

Med-res:
$D_2$ x H/4 x W/4

Low-res:
$D_3$ x H/4 x W/4
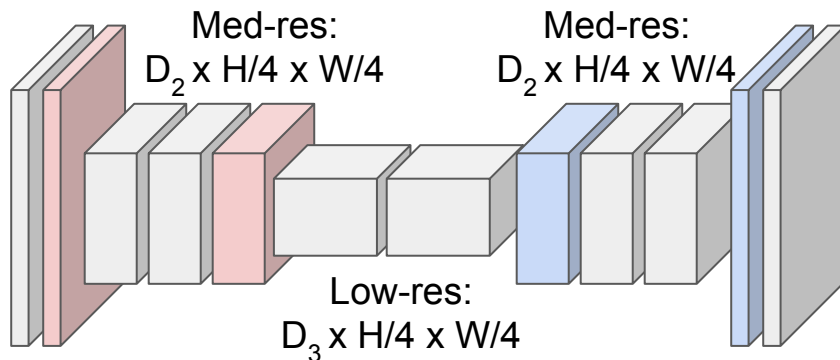
High-res:
$D_1$ x H/2 x W/2

High-res:
$D_1$ x H/2 x W/2

**Upsampling**:
Unpooling or strided transpose convolution



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

# Classification + Localization



**GRASS, CAT, TREE, SKY**

No objects, just pixels

**CAT**

Single Object

**DOG, DOG, CAT**

**DOG, DOG, CAT**

Multiple Object

This image is CC0 public domain

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Classification + Localization



This image is CC0 public domain
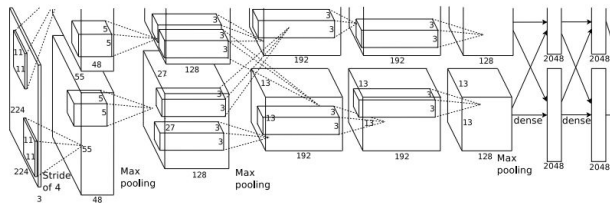
**Fully Connected**: 4096 to 1000

**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!

# Classification + Localization



Correct label:
Cat

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

**Fully Connected**:
4096 to 1000

**Softmax Loss**

**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!

# Classification + Localization



Fully
Connected:
4096 to 1000

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

Correct label:
Cat

Softmax
Loss

Multitask Loss

**+** → **Loss**

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!

This image is CC0 public domain

# Classification + Localization



Often pretrained on ImageNet
(Transfer learning)

Vector: 4096

**Fully Connected**: 4096 to 1000

**Fully Connected**: 4096 to 4

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

**Box Coordinates**
(x, y, w, h)

**Correct label:** Cat

**Softmax Loss**

**L2 Loss**

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

**+** → **Loss**

Treat localization as a regression problem!

This image is CC0 public domain

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Aside: Human Pose Estimation

Johnson and Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation", BMVC 2010

Represent pose as a set of 14 joint positions:

Left / right foot
Left / right knee
Left / right hip
Left / right shoulder
Left / right elbow
Left / right hand
Neck
Head top

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Aside: Human Pose Estimation



**Vector:** 4096

**Left foot**: (x, y)

**Right foot**: (x, y)

…

**Head top**: (x, y)

Toshev and Szegedy, "DeepPose: Human Pose
Estimation via Deep Neural Networks", CVPR 2014

# Aside: Human Pose Estimation



**Correct left foot**: (x', y')

**Left foot**: (x, y) → L2 loss

**Right foot**: (x, y) → L2 loss

…

**Head top**: (x, y) → L2 loss

**Correct head top**: (x', y')

**Vector:** 4096

...

**+** → **Loss**
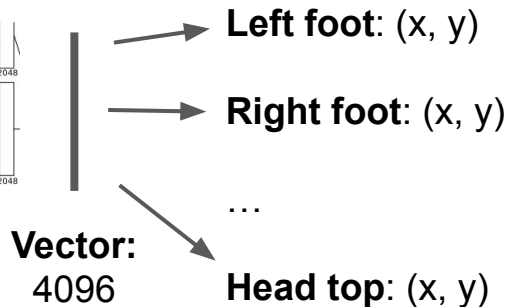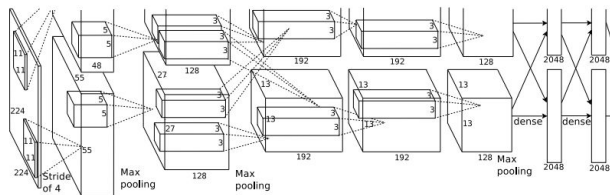
Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
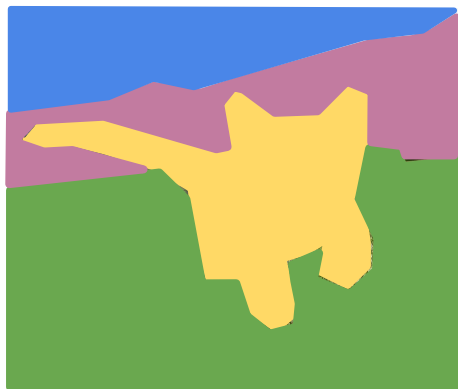
# Object Detection



**GRASS, CAT, TREE, SKY**
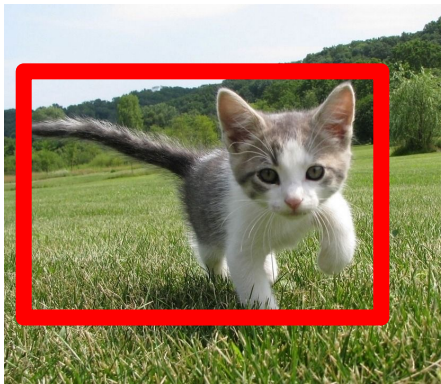
No objects, just pixels

**CAT**

Single Object

**DOG, DOG, CAT**

**DOG, DOG, CAT**

Multiple Object
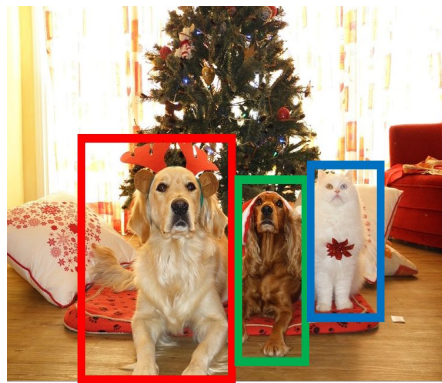
This image is CC0 public domain

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Object Detection: Impact of Deep Learning
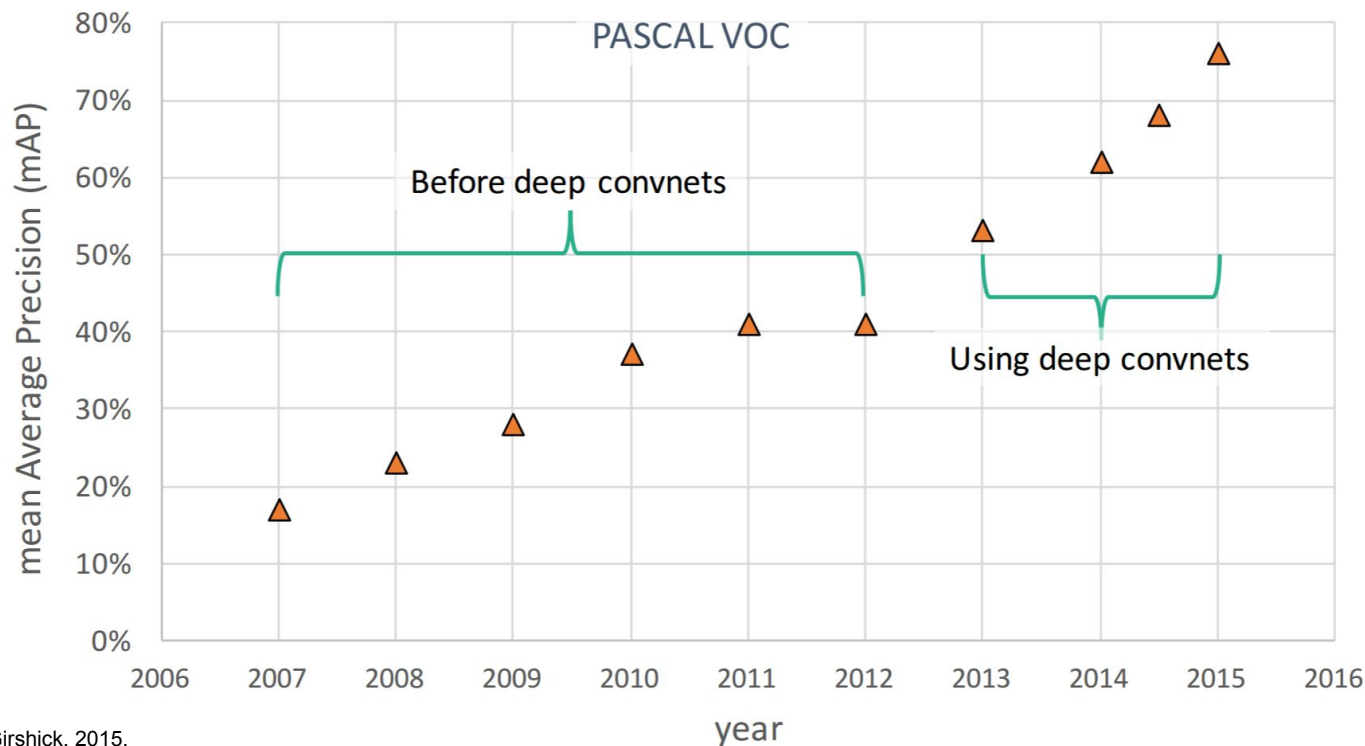


Figure copyright Ross Girshick, 2015.
Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
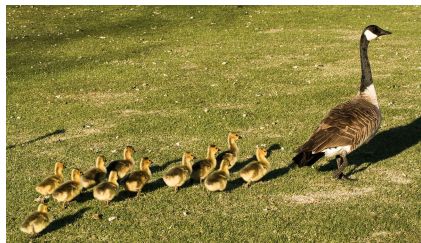
# Object Detection as Regression?
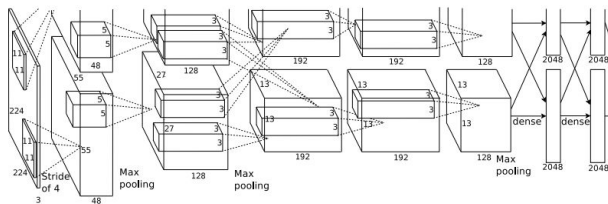


CAT: (x, y, w, h)
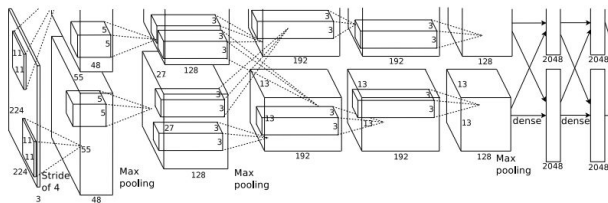
DOG: (x, y, w, h)
DOG: (x, y, w, h)
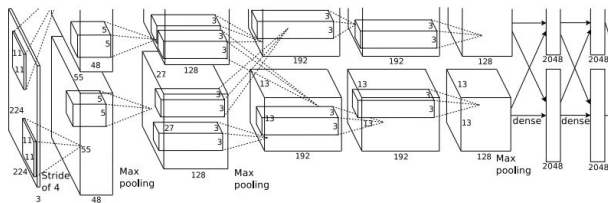CAT: (x, y, w, h)
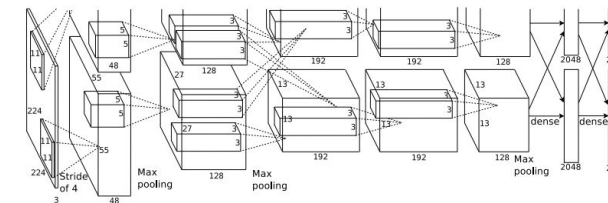
DUCK: (x, y, w, h)
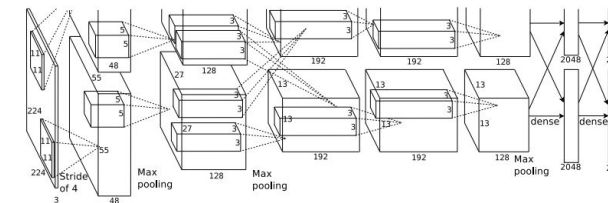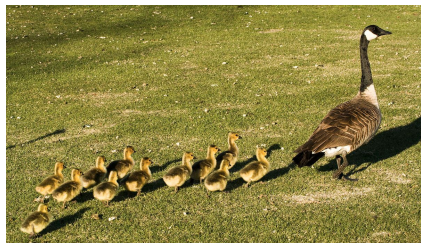DUCK: (x, y, w, h)
….

# Object Detection as Regression?

Each image needs a different number of outputs!
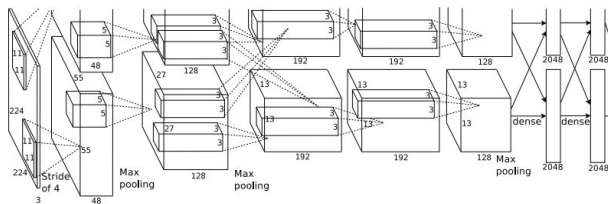


CAT: (x, y, w, h)

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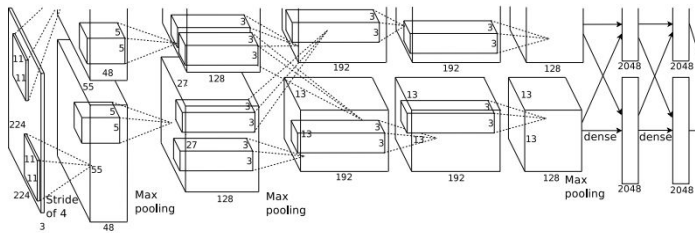
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers

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

Many numbers!

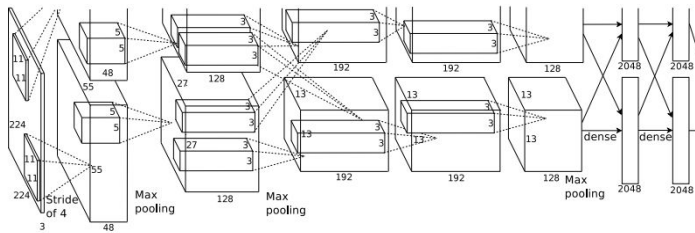# Object Detection as Classification: Sliding Window

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



Dog? NO
Cat? NO
Background? YES

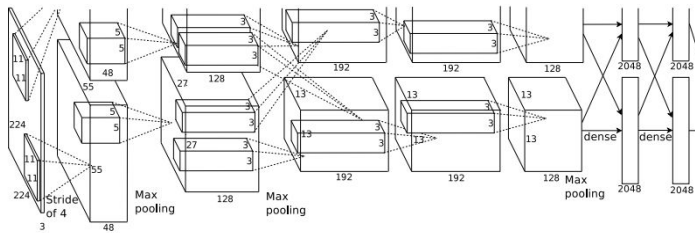# Object Detection as Classification: Sliding Window

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



Dog? YES
Cat? NO
Background? NO

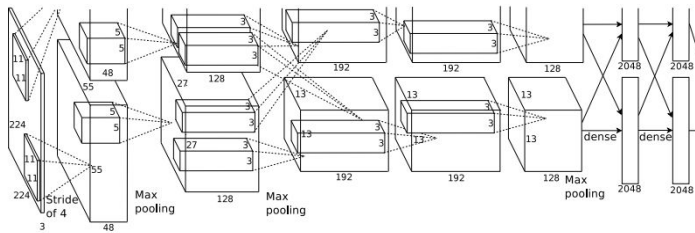# Object Detection as Classification: Sliding Window

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



Dog? YES
Cat? NO
Background? NO

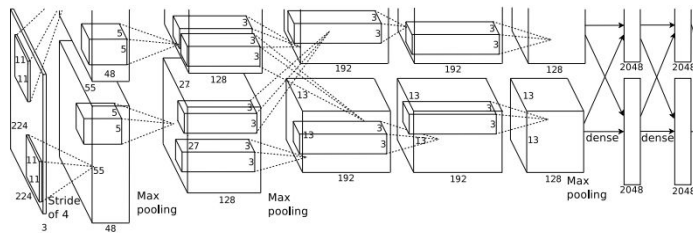# Object Detection as Classification: Sliding Window

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



Dog? NO
Cat? YES
Background? NO

# 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!

# 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
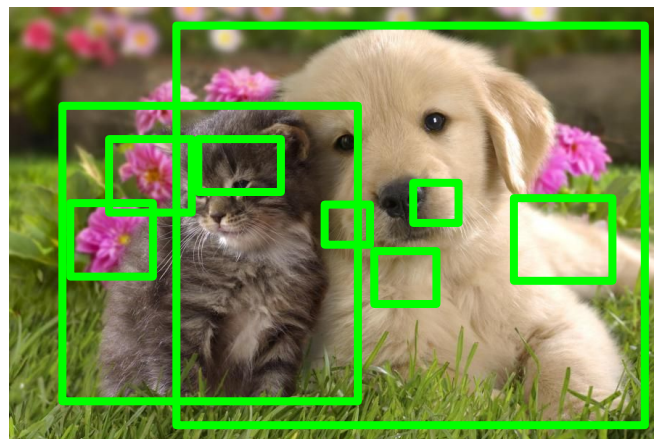


Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges",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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# R-CNN



Input image

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

# R-CNN



Regions of Interest (RoI) from a proposal method (~2k)

Input image

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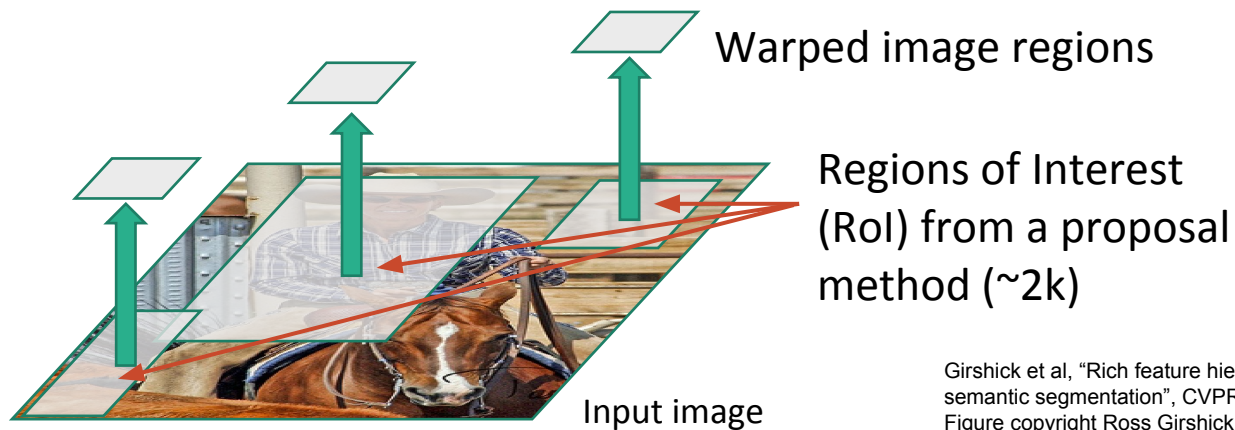Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

# R-CNN



Warped image regions

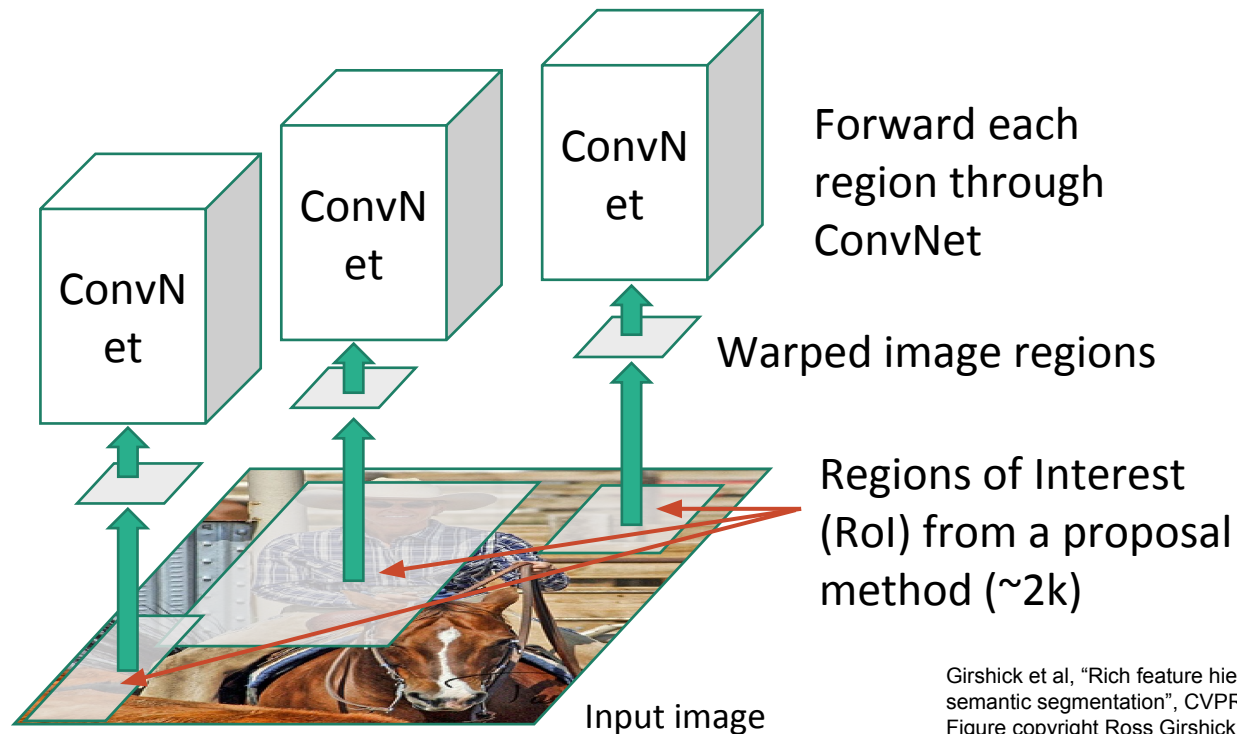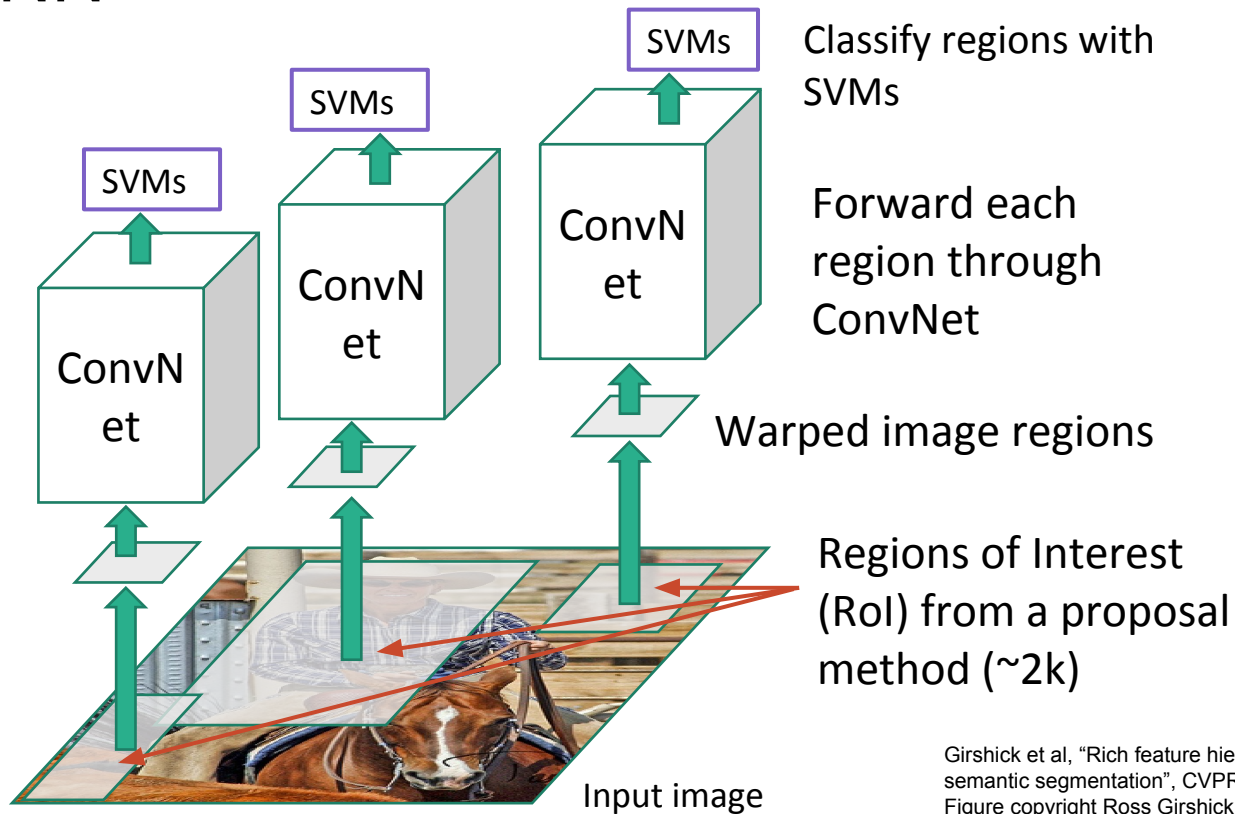Regions of Interest (RoI) from a proposal method (~2k)

Input image

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

# R-CNN



Forward each region through ConvNet

Warped image regions

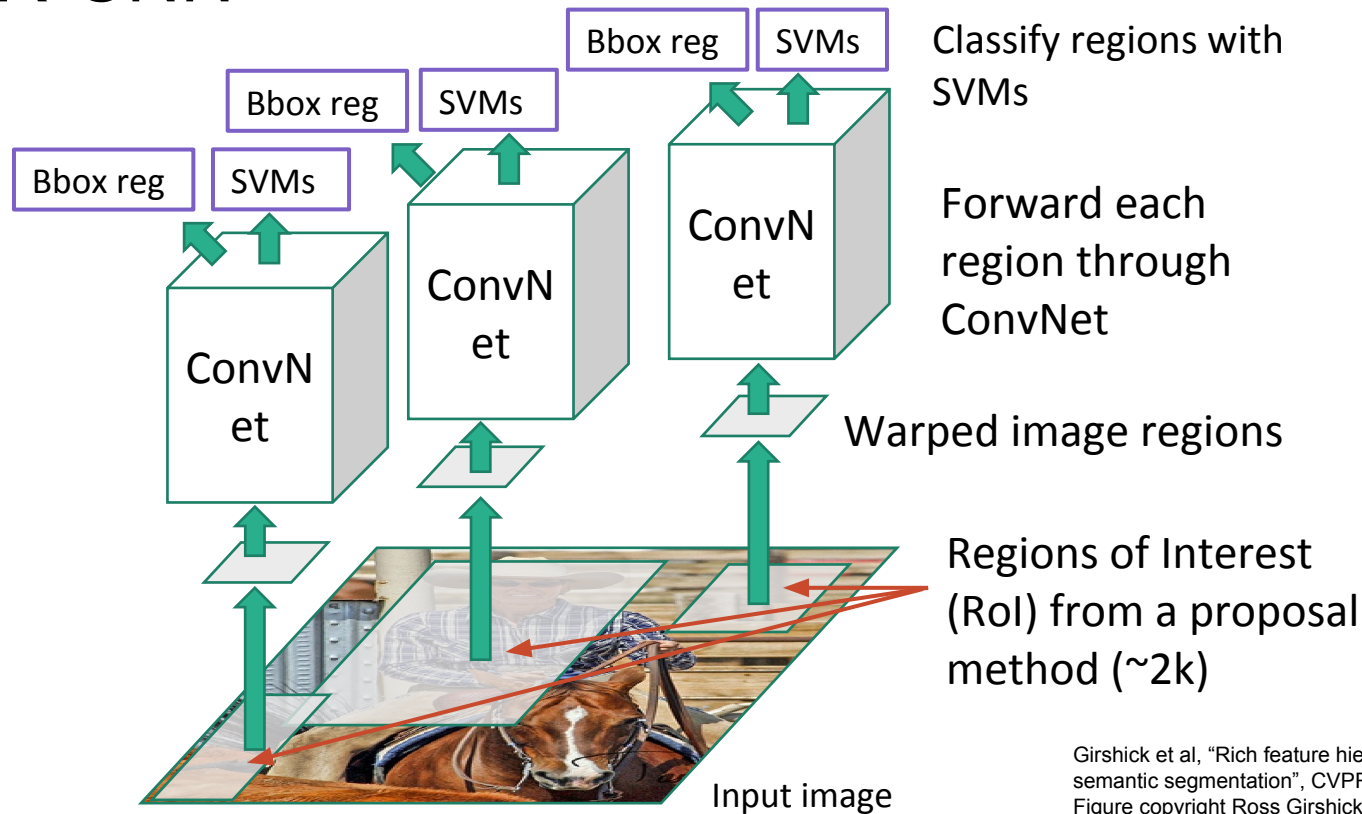Regions of Interest (RoI) from a proposal method (~2k)

Input image

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

# R-CNN



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Input image

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

# R-CNN

Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Input image

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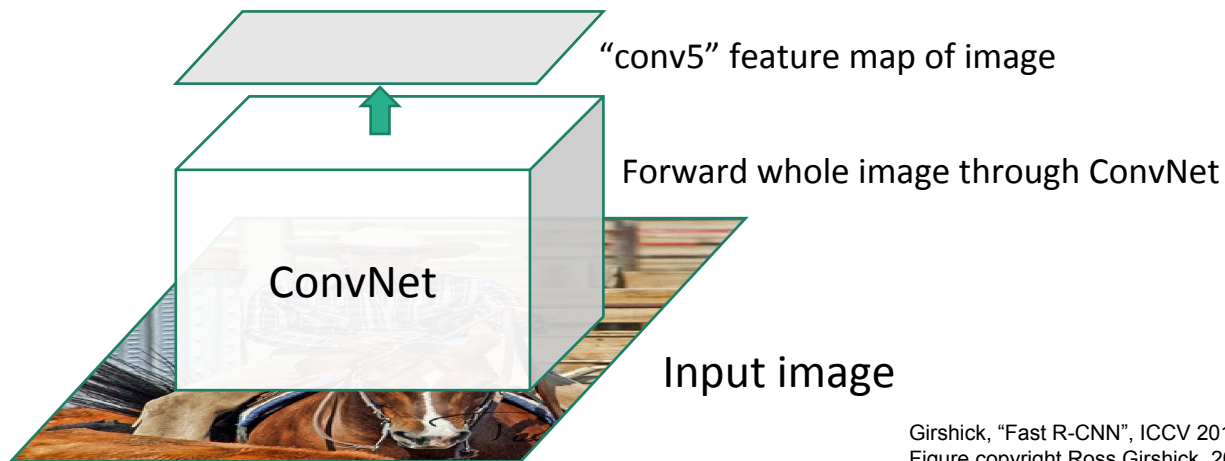Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

# Fast R-CNN



Input image

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Fast R-CNN



"conv5" feature map of image

Forward whole image through ConvNet

ConvNet

Input image

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

# Fast R-CNN


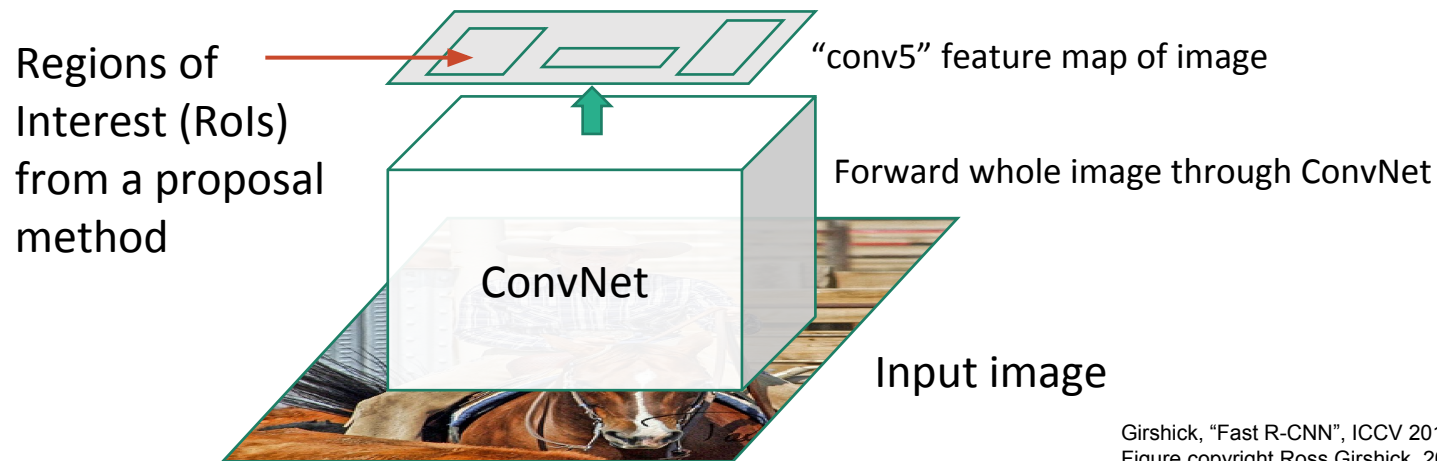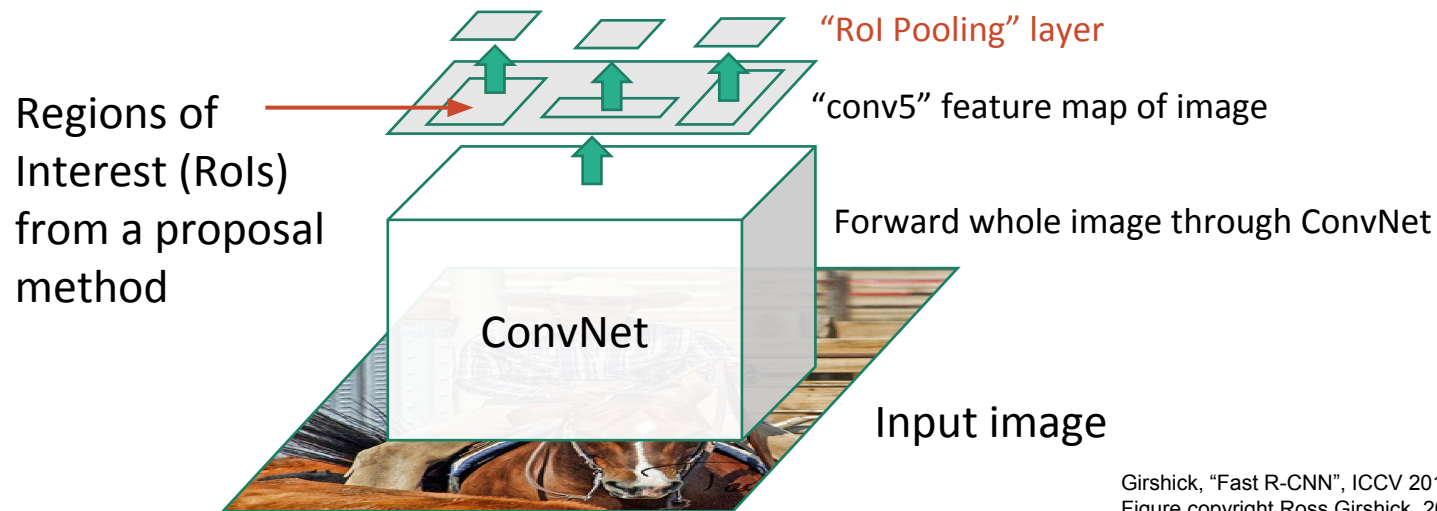
Regions of Interest (RoIs) from a proposal method

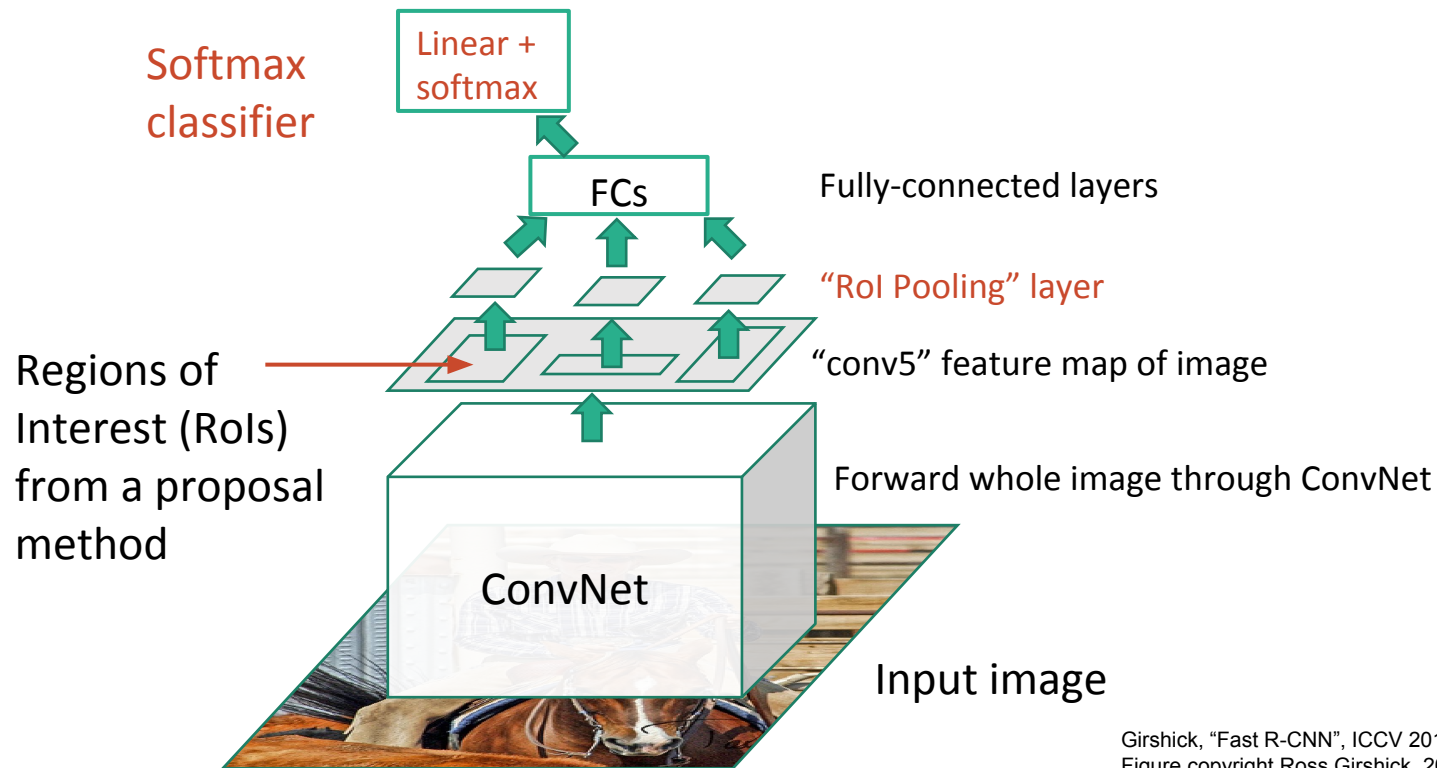"conv5" feature map of image

Forward whole image through ConvNet

ConvNet

Input image

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

# Fast R-CNN



"RoI Pooling" layer
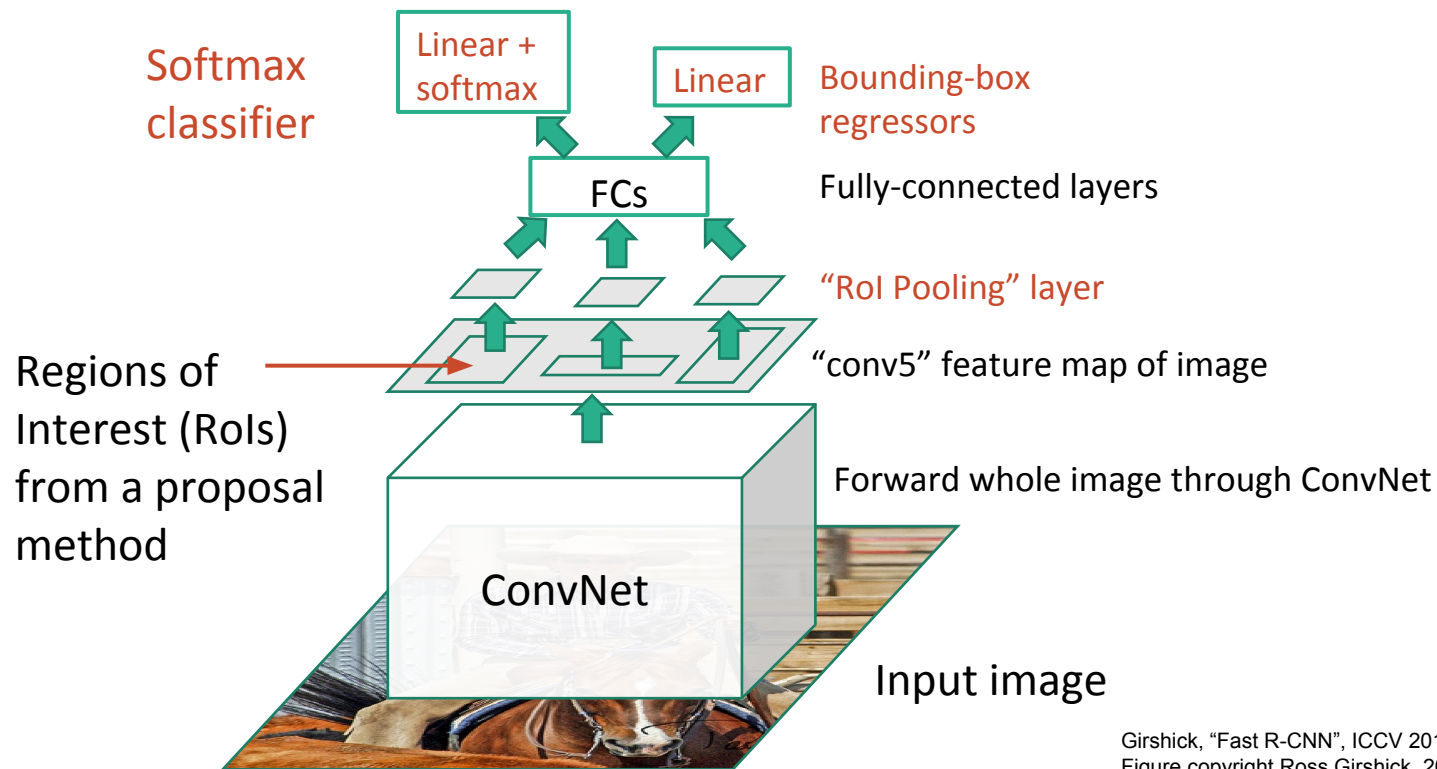
Regions of Interest (RoIs) from a proposal method

"conv5" feature map of image

Forward whole image through ConvNet

ConvNet

Input image

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
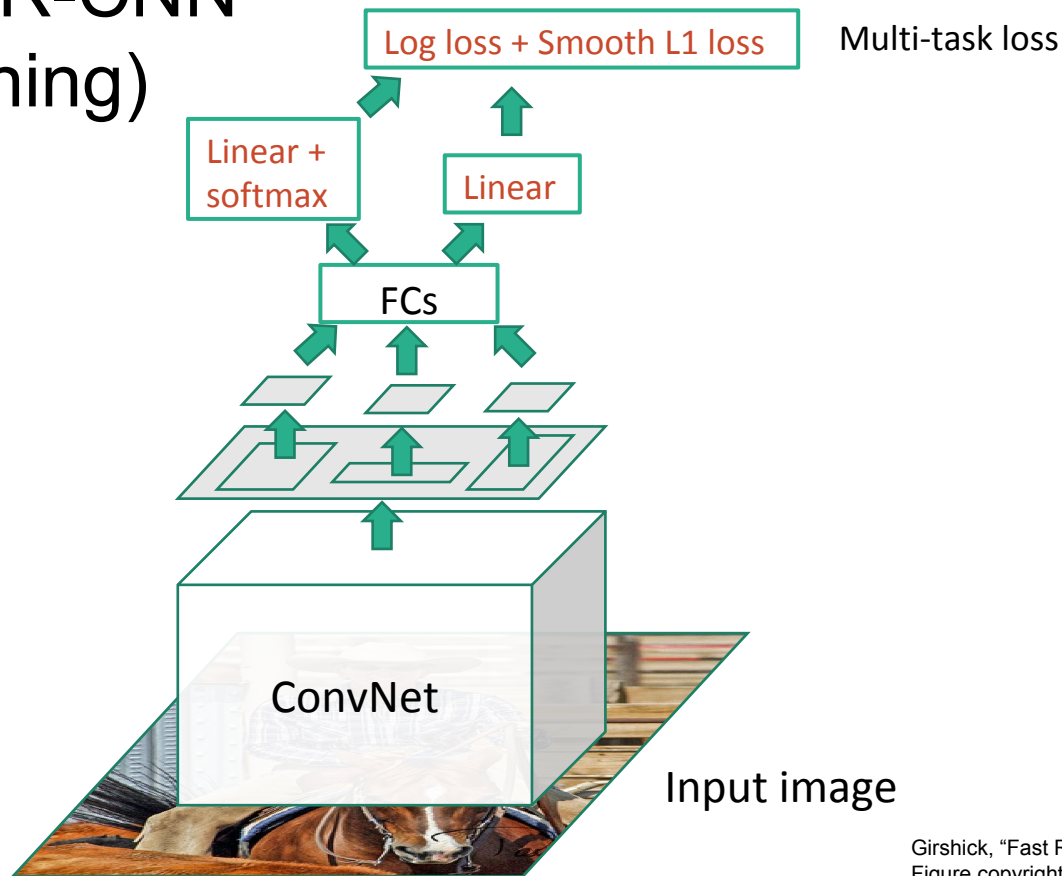
# Fast R-CNN



Softmax classifier

Linear + softmax

FCs — Fully-connected layers

"RoI Pooling" layer

Regions of Interest (RoIs) from a proposal method

"conv5" feature map of image

Forward whole image through ConvNet

ConvNet

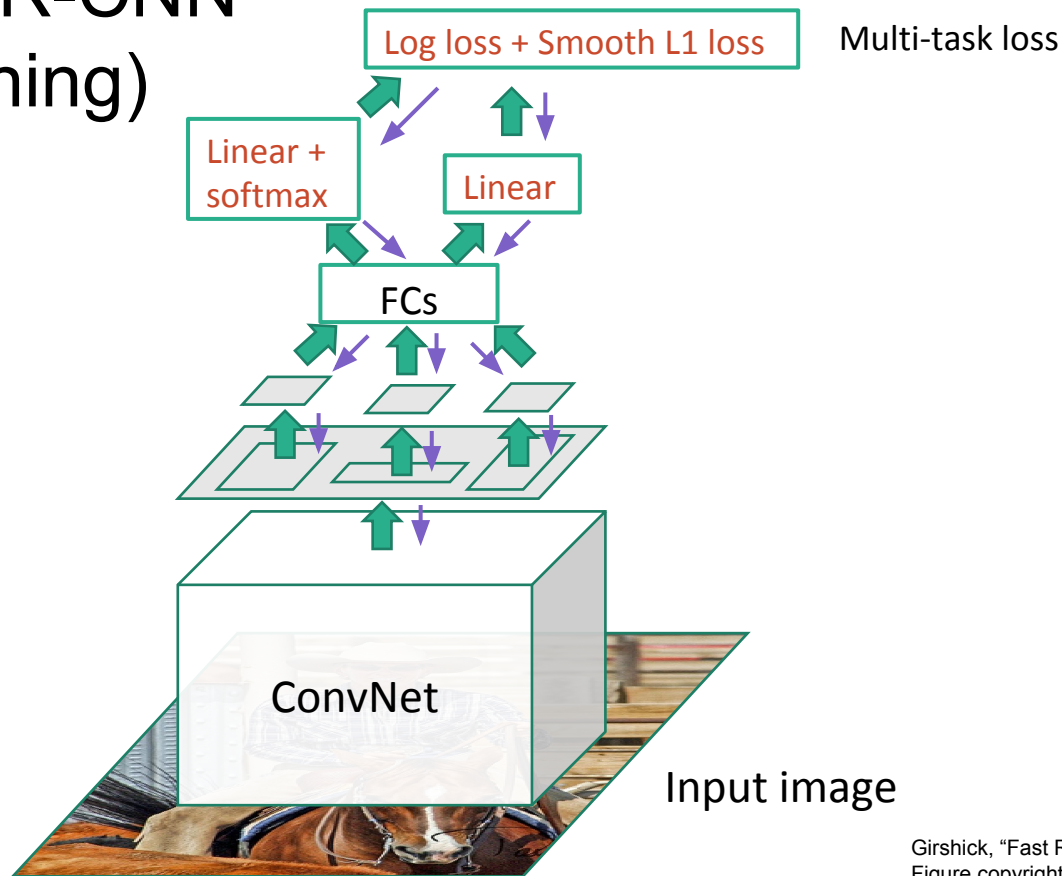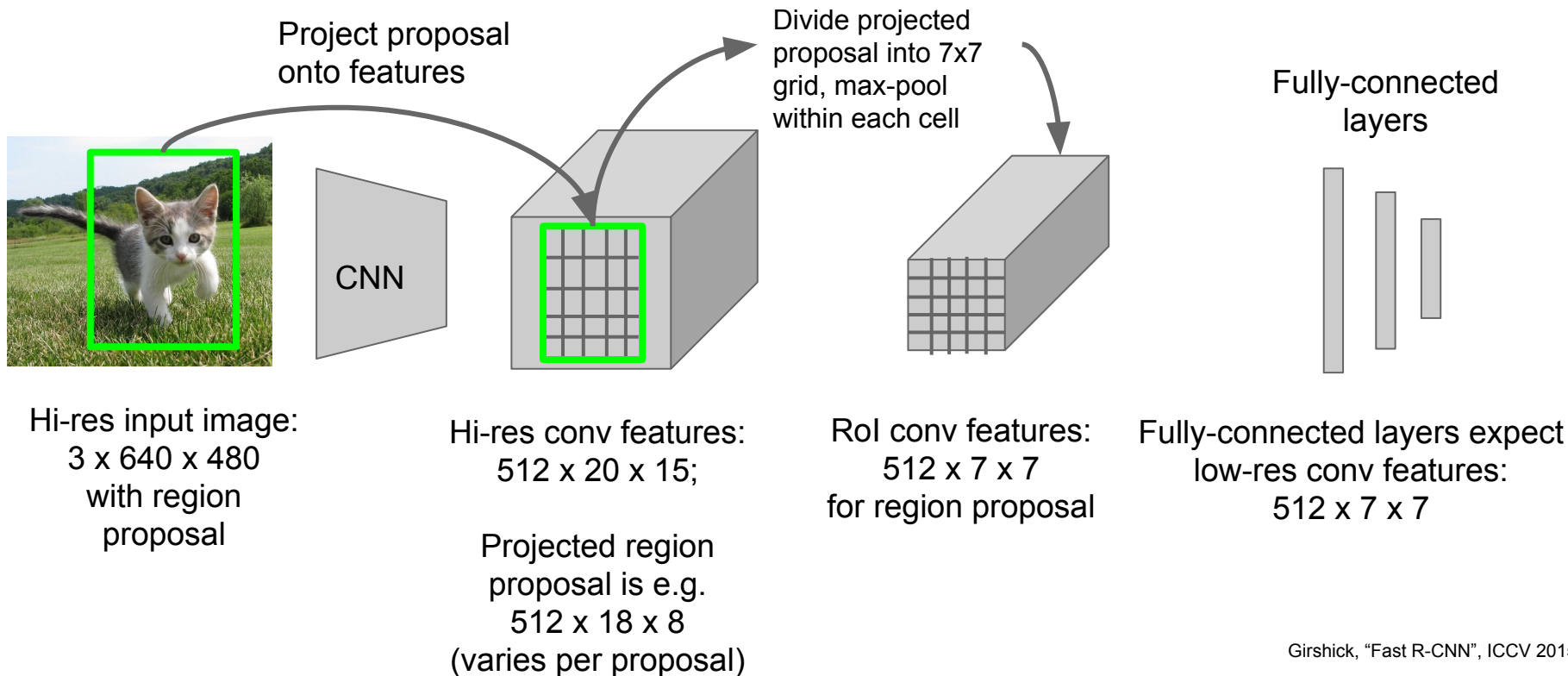Input image

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Fast R-CNN



Softmax classifier

Linear + softmax

Linear

Bounding-box regressors

FCs

Fully-connected layers

"RoI Pooling" layer

Regions of Interest (RoIs) from a proposal method

"conv5" feature map of image

ConvNet

Forward whole image through ConvNet

Input image

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

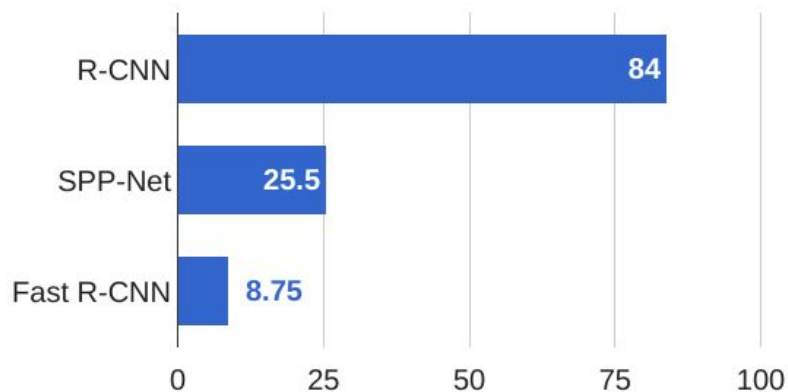Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Fast R-CNN
## (Training)



Log loss + Smooth L1 loss

Multi-task loss

Linear + softmax

Linear

FCs

ConvNet

Input image

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

# Fast R-CNN (Training)



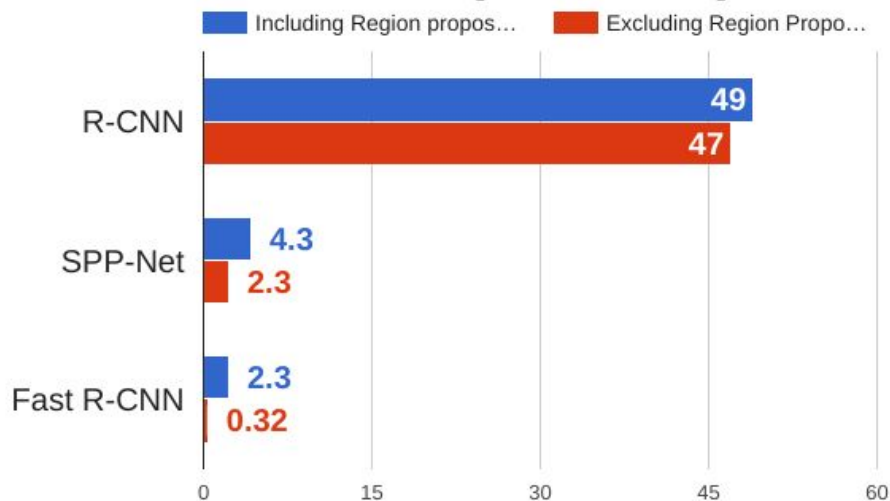Log loss + Smooth L1 loss

Multi-task loss

Linear + softmax

Linear

FCs

ConvNet

Input image

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Faster R-CNN: RoI Pooling

Project proposal onto features

Divide projected proposal into 7x7 grid, max-pool within each cell

Fully-connected layers

CNN

Hi-res input image:
3 x 640 x 480
with region
proposal

Hi-res conv features:
512 x 20 x 15;

Projected region
proposal is e.g.
512 x 18 x 8
(varies per proposal)

RoI conv features:
512 x 7 x 7
for region proposal

Fully-connected layers expect
low-res conv features:
512 x 7 x 7

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

# R-CNN vs SPP vs Fast R-CNN



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

# R-CNN vs SPP vs Fast R-CNN



**Training time (Hours)**
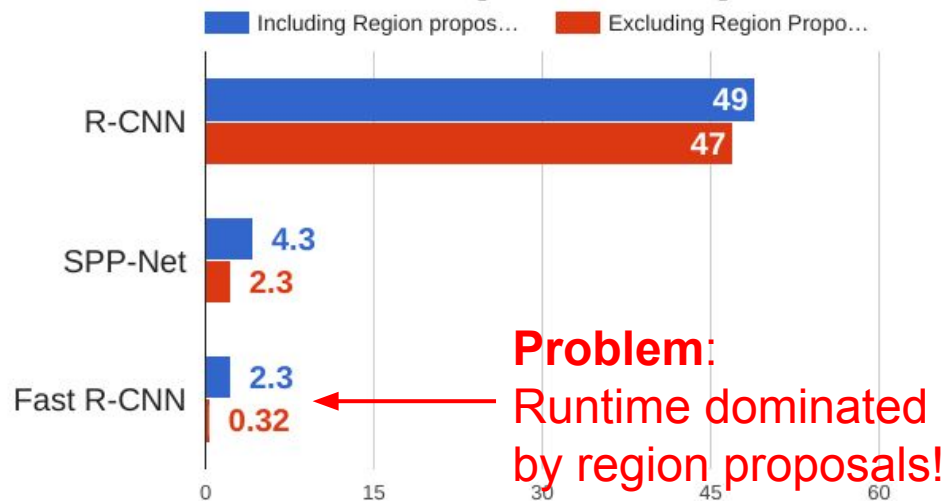
R-CNN: 84
SPP-Net: 25.5
Fast R-CNN: 8.75

**Test time (seconds)**

Including Region propos...  Excluding Region Propo...

R-CNN: 49 / 47
SPP-Net: 4.3 / 2.3
Fast R-CNN: 2.3 / 0.32

**Problem**:
Runtime dominated by region proposals!

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

# 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
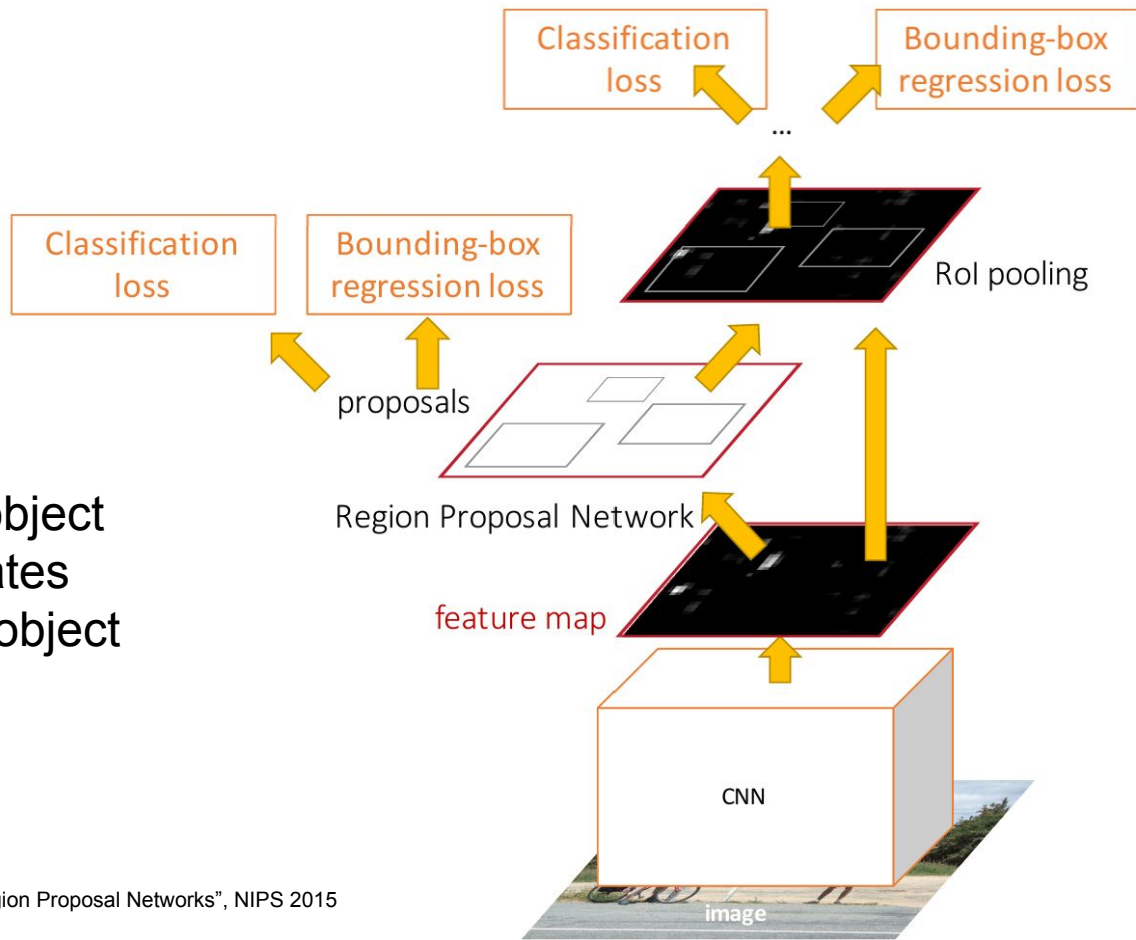


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
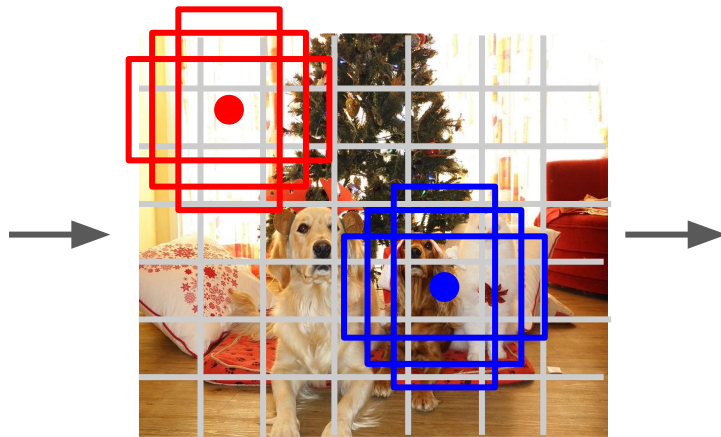Figure copyright 2015, Ross Girshick; reproduced with permission

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

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



**R-CNN Test-Time Speed**

| | Speed |
|---|---|
| R-CNN | 49 |
| SPP-Net | 4.3 |
| Fast R-CNN | 2.3 |
| Faster R-CNN | 0.2 |
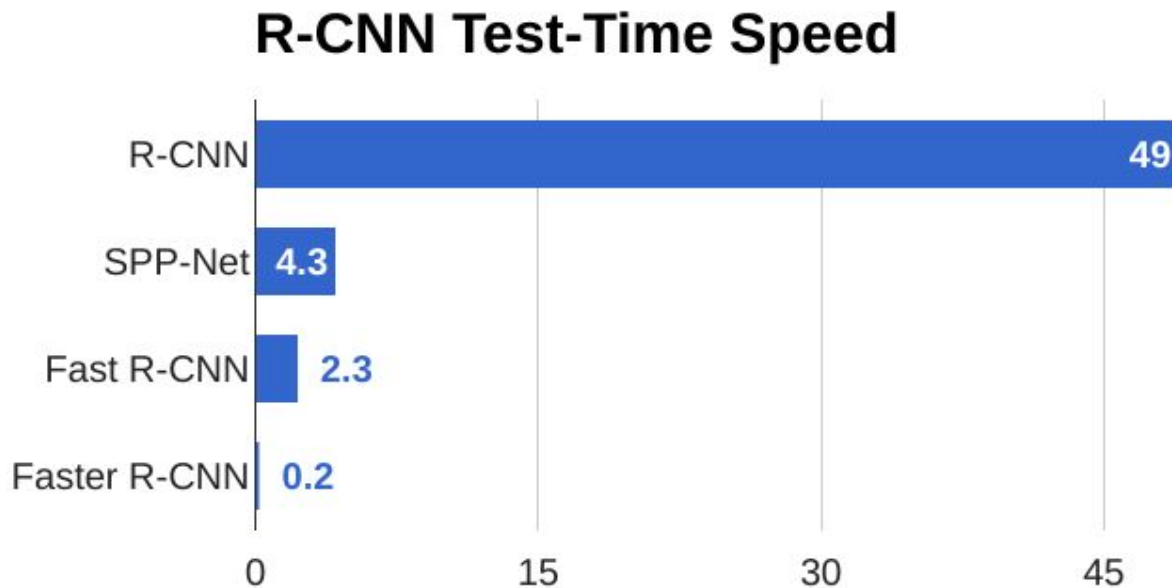
# Detection without Proposals: YOLO / SSD



Input image
3 x H x W

Divide image into grid
7 x 7

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

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

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

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Detection without Proposals: YOLO / SSD

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



Input image
3 x H x W
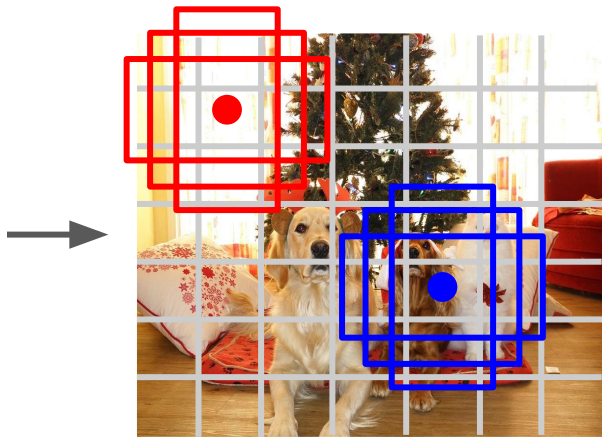
Divide image into grid
7 x 7

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

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

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

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

# Object Detection: Lots of variables ...

**Base Network**
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

**Object Detection architecture**
Faster R-CNN
R-FCN
SSD

**Image Size**
**# Region Proposals**
**…**

**Takeaways**
Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

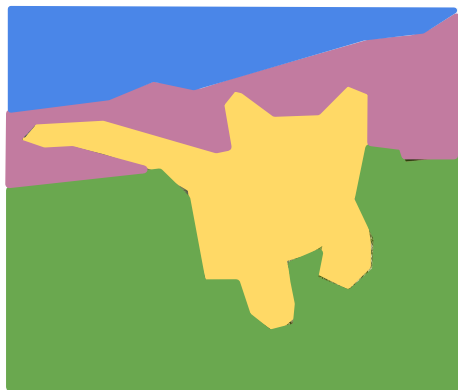# Aside: Object Detection + Captioning = Dense Captioning

# Aside: Object Detection + Captioning = Dense Captioning
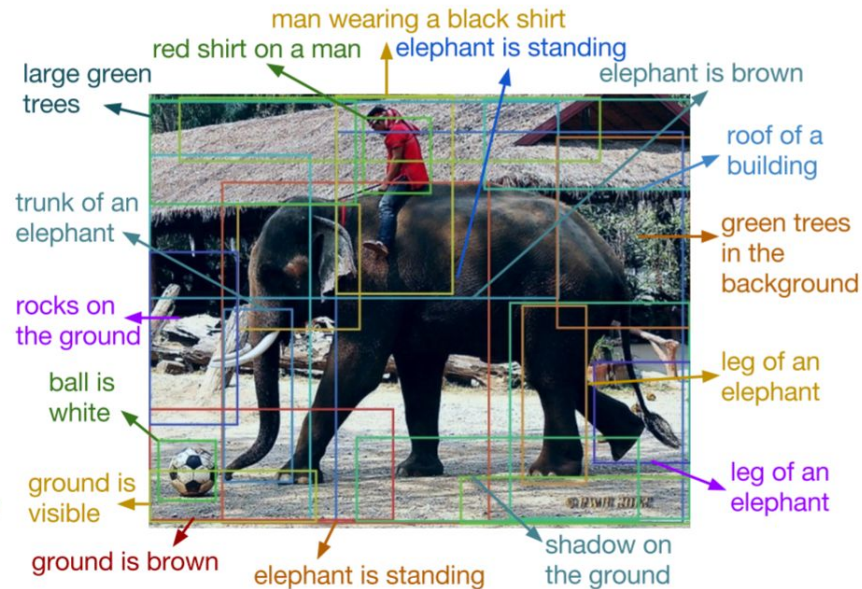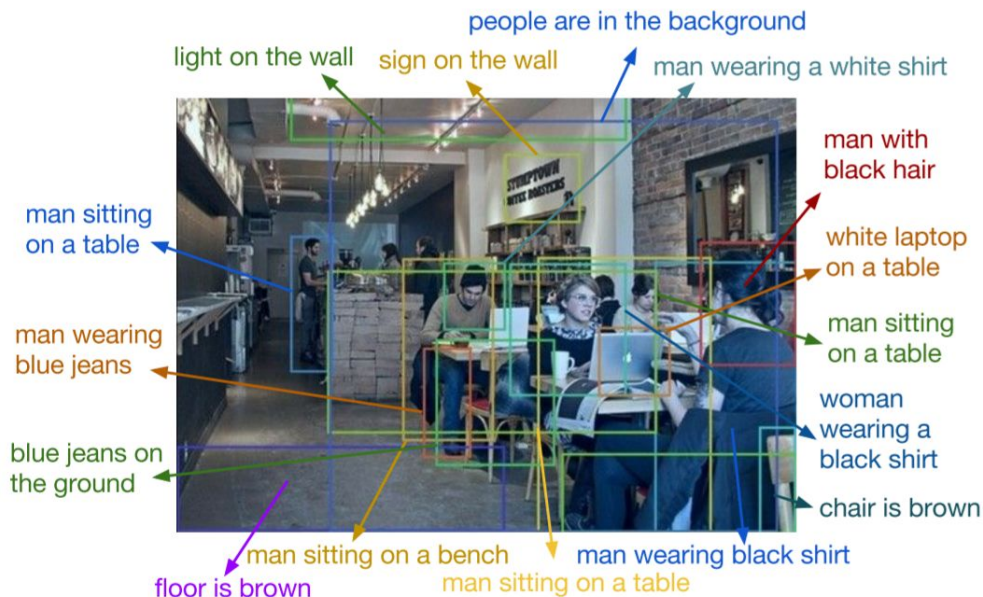


Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
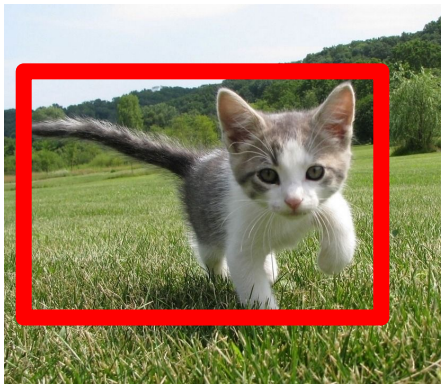Figure copyright IEEE, 2016. Reproduced for educational purposes.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
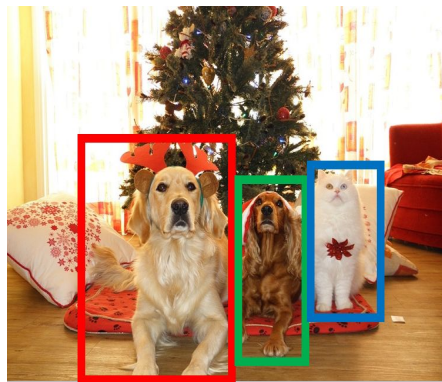
# Instance Segmentation



**GRASS**, **CAT**,
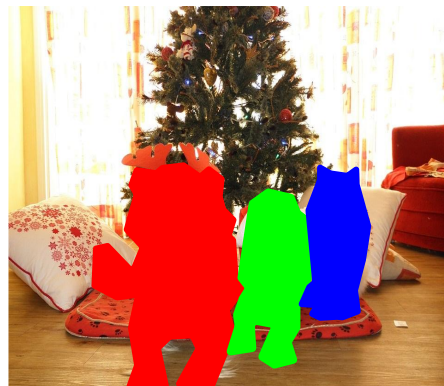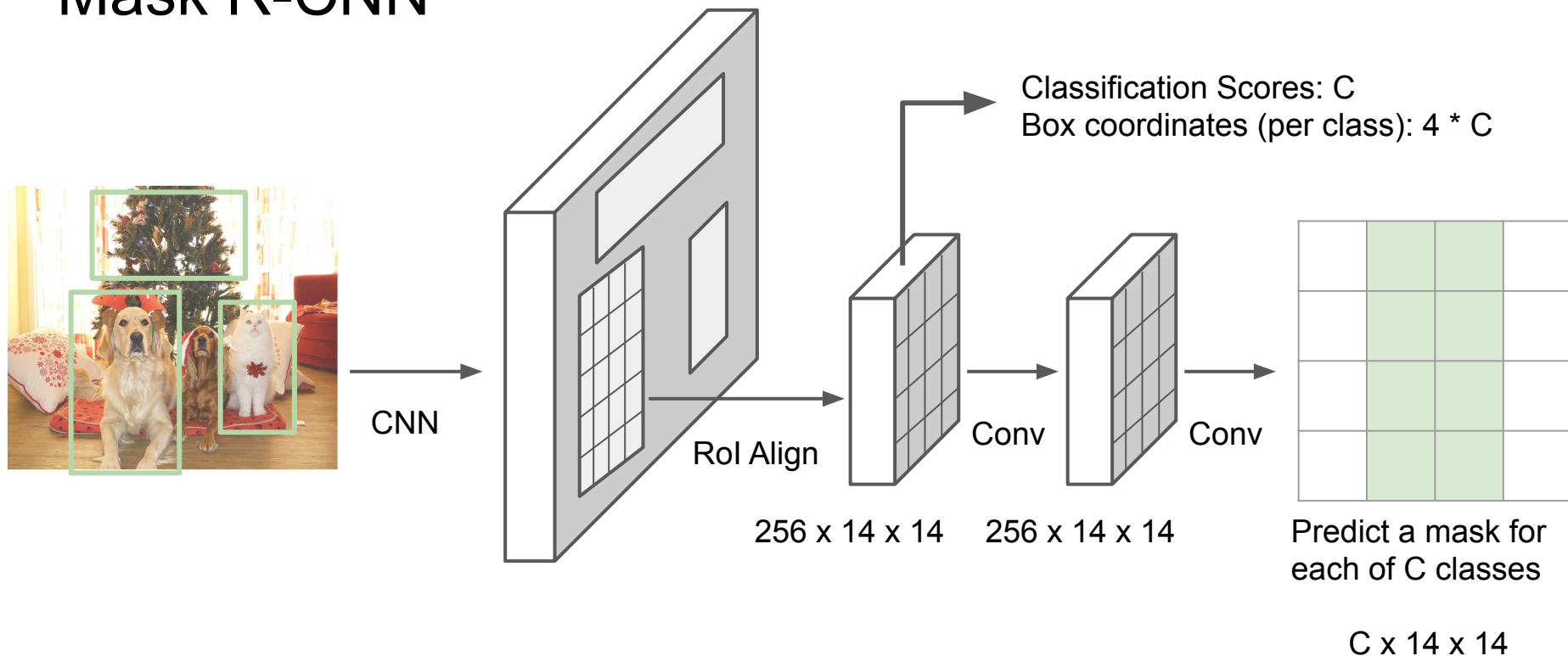**TREE**, **SKY**

No objects, just pixels

**CAT**

Single Object

**DOG**, **DOG**, **CAT**

**DOG**, **DOG**, **CAT**

Multiple Object

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Mask R-CNN



Classification Scores: C
Box coordinates (per class): 4 * C

CNN

RoI Align

256 x 14 x 14

Conv

256 x 14 x 14

Conv

Predict a mask for each of C classes

C x 14 x 14

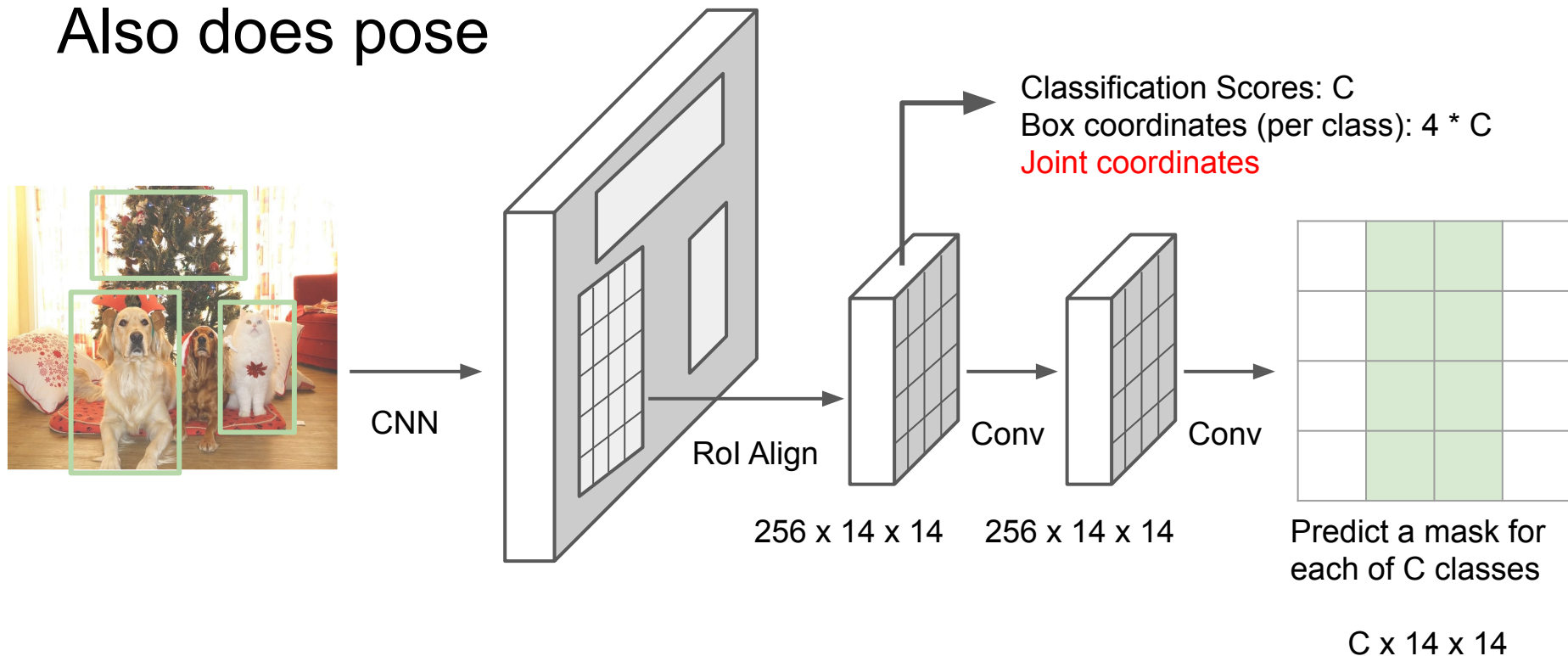He et al, "Mask R-CNN", arXiv 2017

# Mask R-CNN: Very Good Results!
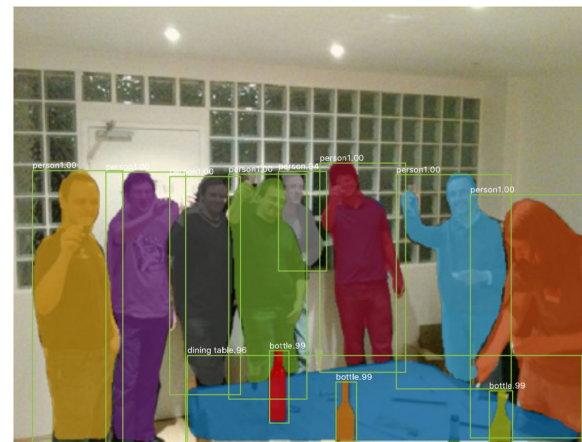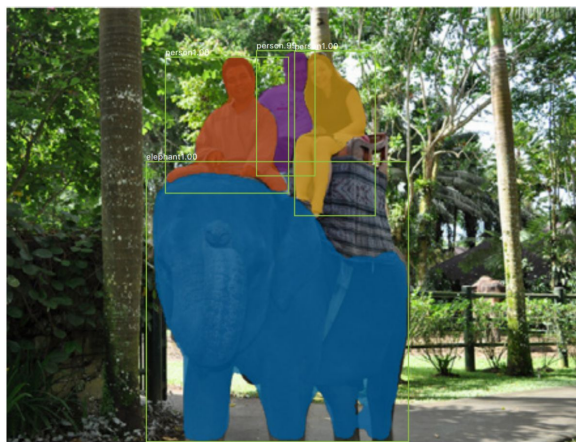


He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung
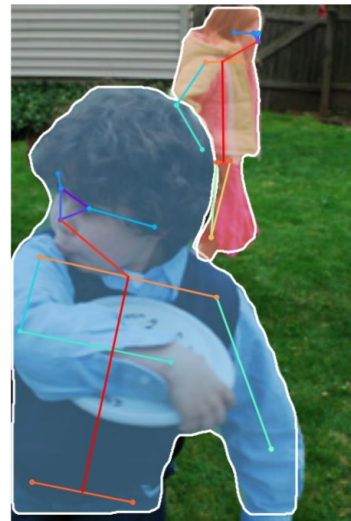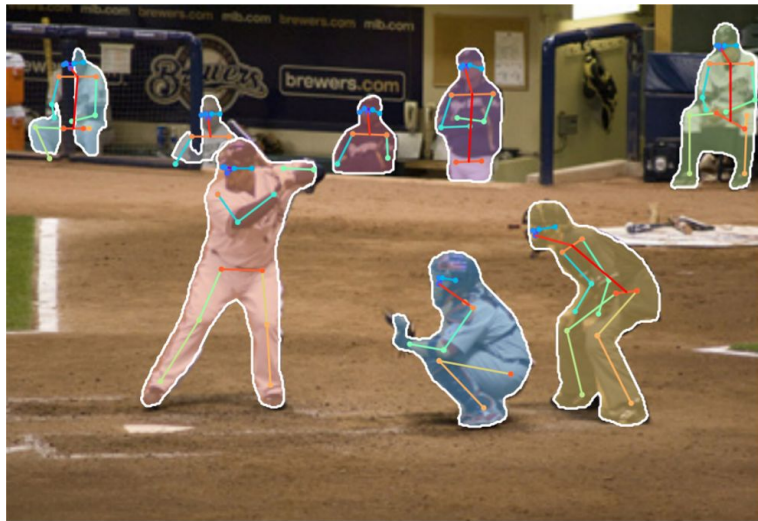
# Mask R-CNN
# Also does pose



Classification Scores: C
Box coordinates (per class): 4 * C
Joint coordinates

CNN

RoI Align

256 x 14 x 14

Conv

256 x 14 x 14

Conv

Predict a mask for each of C classes

C x 14 x 14

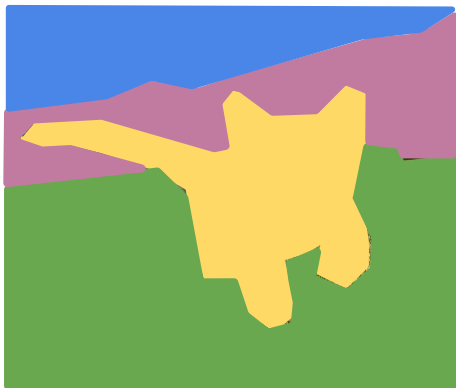He et al, "Mask R-CNN", arXiv 2017

# Mask R-CNN
# Also does pose



He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

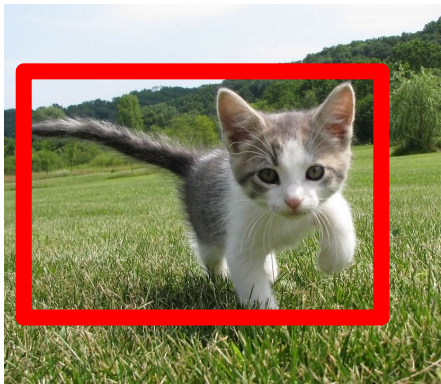Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Recap:

| Semantic Segmentation | Classification + Localization | Object Detection | Instance Segmentation |
|---|---|---|---|



**GRASS**, **CAT**, **TREE**, **SKY**

**CAT**

**DOG**, **DOG**, **CAT**

**DOG**, **DOG**, **CAT**

No objects, just pixels

Single Object

Multiple Object

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung