



# Introduction to Computer Vision

## Lecture 6 - Deep Learning IV

Prof. He Wang

# Logistics

- Assignment 1: due on 4/1 11:59PM (Saturday)
  - Assignment 2: released on 4/1 11:59PM (Saturday)
- 
- If 1 day (0 - 24 hours) past the deadline, 15% off
  - If 2 day (24 - 48 hours) past the deadline, 30% off
  - Zero credit if more than 2 days.

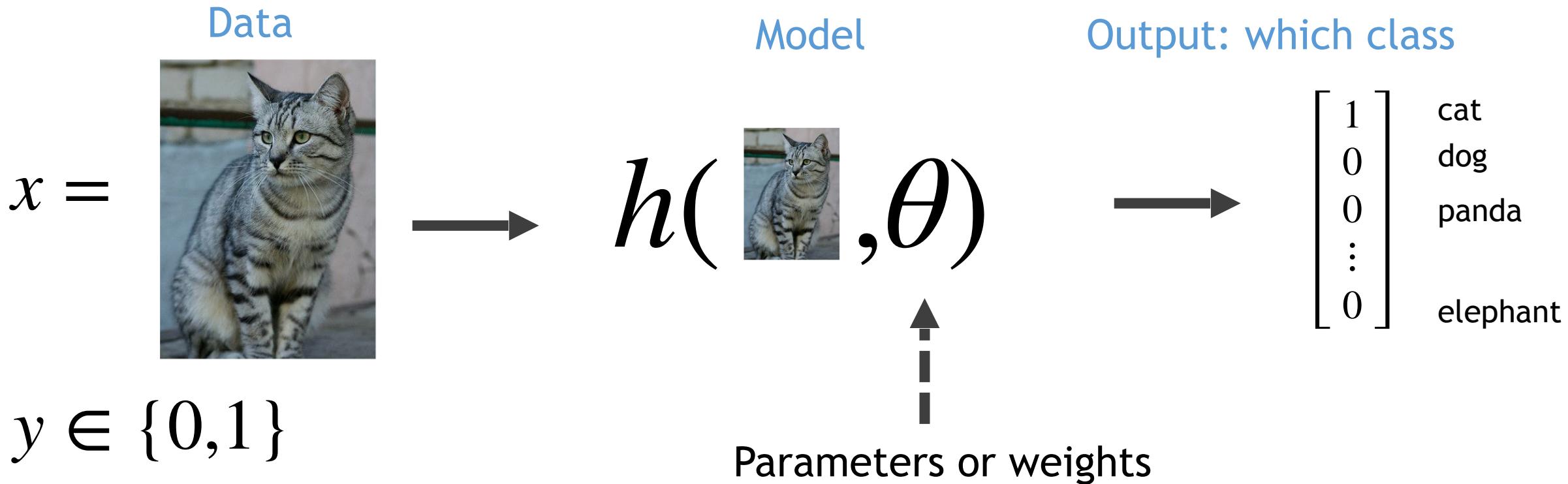
# Logistics

- Assignment 1: due on 4/1 11:59PM (Saturday)
- Assignment 2: released on 4/1 11:59PM (Saturday)
- If 1 day (0 - 24 hours) past the deadline, 15% off
- If 2 day (24 - 48 hours) past the deadline, 30% off
- Zero credit if more than 2 days.
- Midterm: 4/26, in class

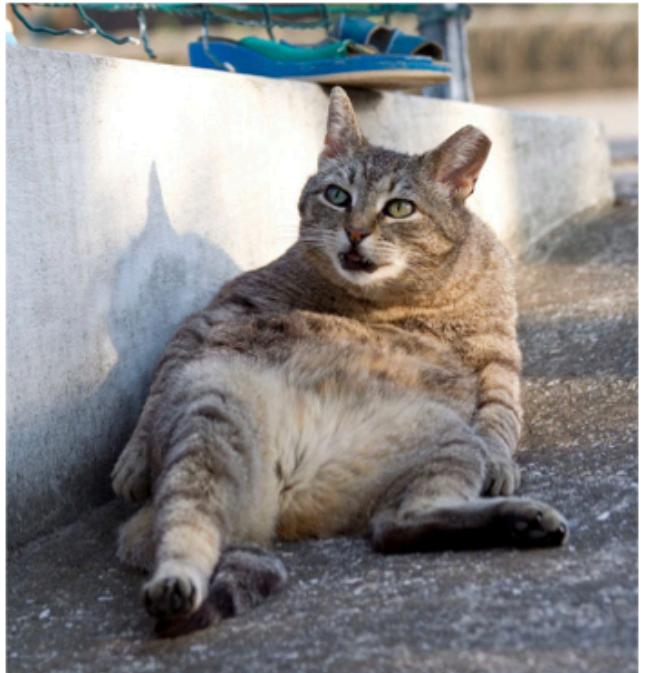
# Classification

# Image Classification

- Classic definition: image classification is to categorize an image into several known classes ( $N$ ).
- Image classification is very important for **semantic understanding**.



# Challenges: Pose and Deformation



[This image](#) by [Umberto Salvagnin](#)  
is licensed under [CC-BY 2.0](#)



[This image](#) by [Umberto Salvagnin](#)  
is licensed under [CC-BY 2.0](#)

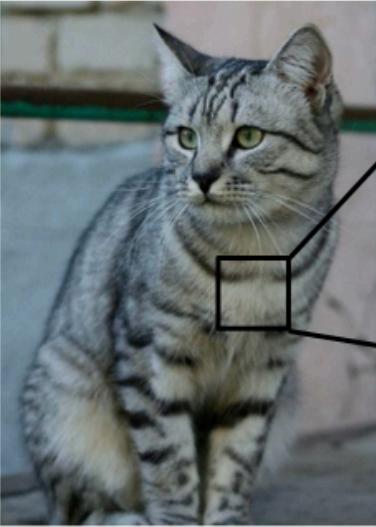
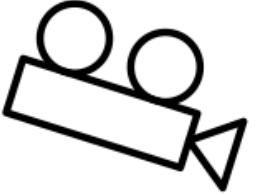


[This image](#) by [sare bear](#) is  
licensed under [CC-BY 2.0](#)

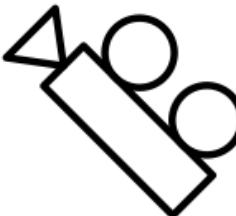
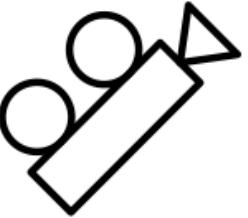


[This image](#) by [Tom Thai](#) is  
licensed under [CC-BY 2.0](#)

# Challenges: Viewpoint Variation



[105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87]
[ 91 108 106 104 101 98 105 102 100 107 109 105 103 102 106 100]
[ 76 85 98 106 120 117 107 109 105 103 102 106 100 104 105]
[ 99 81 81 93 129 117 107 109 105 103 102 106 100 99 101]
[106 81 61 64 69 91 88 85 101 107 109 98 75 84 86 94]
[114 100 85 55 55 69 64 54 64 87 112 129 98 74 84 91]
[133 137 147 103 65 81 86 65 62 52 54 74 84 102 93 85 82]
[128 137 144 140 109 95 86 78 62 65 63 63 60 73 86 101]
[125 133 148 137 119 121 117 94 65 79 88 65 54 64 72 98]
[127 125 131 147 133 127 121 131 111 96 89 75 61 64 72 94]
[115 114 109 123 150 148 131 118 113 109 108 92 74 65 72 78]
[ 89 93 98 97 108 147 131 118 113 114 113 109 106 95 77 88]
[ 63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87]
[ 62 65 82 89 78 71 88 101 124 126 119 101 107 114 131 119]
[ 63 65 75 88 89 71 62 81 120 138 135 105 81 98 110 118]
[ 87 65 71 87 106 95 69 45 76 138 126 107 92 94 105 112]
[118 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107]
[164 146 112 88 82 123 121 104 76 48 45 66 88 101 102 109]
[157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94]
[138 128 134 161 139 108 109 118 121 134 114 87 65 53 69 86]
[128 112 96 117 150 144 120 115 104 107 102 93 87 81 72 79]
[123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99]
[122 121 102 88 82 86 94 117 145 148 153 102 58 78 92 107]
[122 164 148 103 71 56 78 83 93 103 119 102 61 69 84]



All pixels change when  
the camera moves!

# Challenges: Background Variation



[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain

# Challenges: Illumination



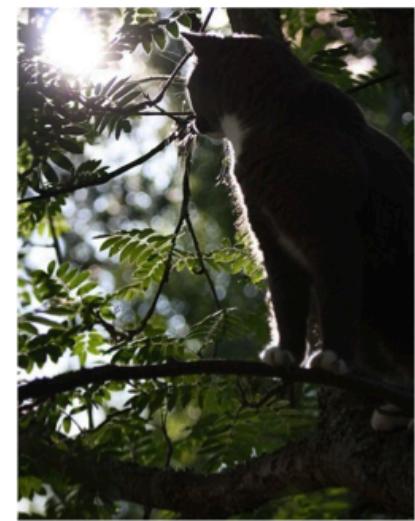
[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain

# Challenges: Occlusion



[This image](#) is [CC0 1.0](#) public domain



[This image](#) is [CC0 1.0](#) public domain



[This image](#) by [jonsson](#) is licensed  
under [CC-BY 2.0](#)

# Challenges: Intraclass Variation



[This image](#) is [CC0 1.0](#) public domain

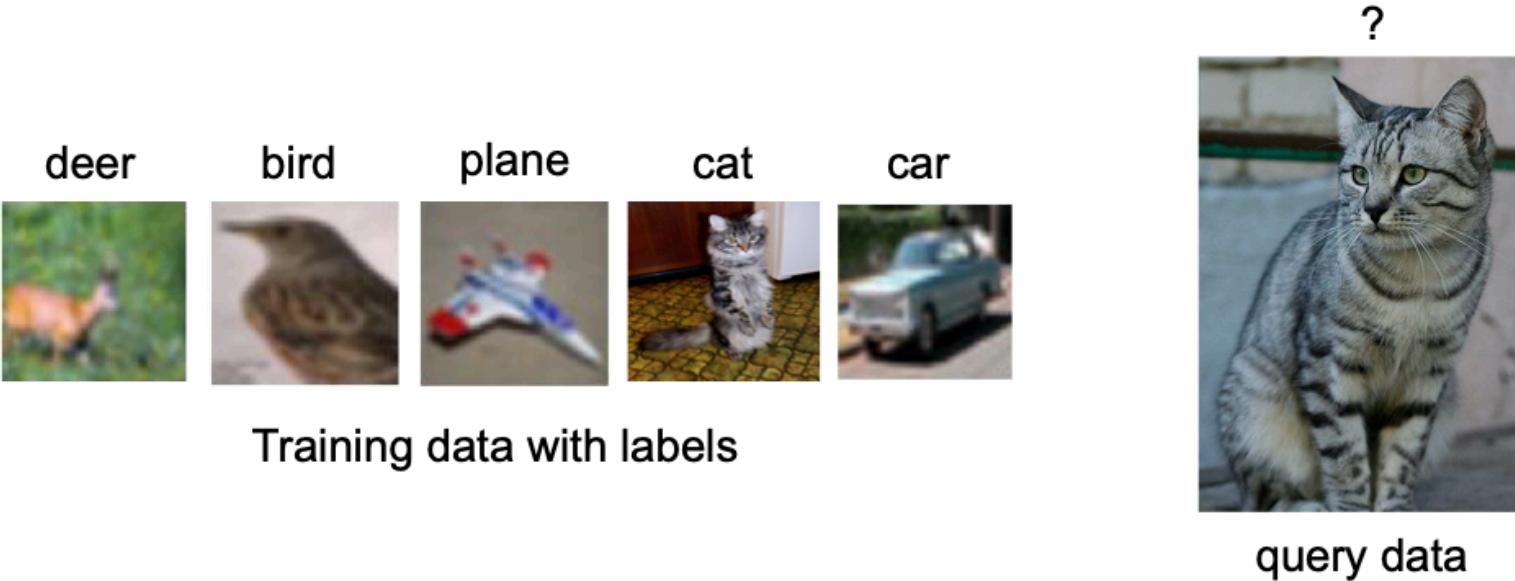
# The Requirement of A Good Image Classifier

- Invariant with all the aforementioned variations.
- That's why we want to add those data augmentations to mimic these variations.
  - Rotation → Pose/viewpoint
  - Color jittering → illumination
  - Crop and scale → viewpoint
  - ...

# Methods

- Non-parameteric models
  - Nearest Neighbor
- Parametric models
  - CNN
  - ...

# Nearest Neighbor Classifier

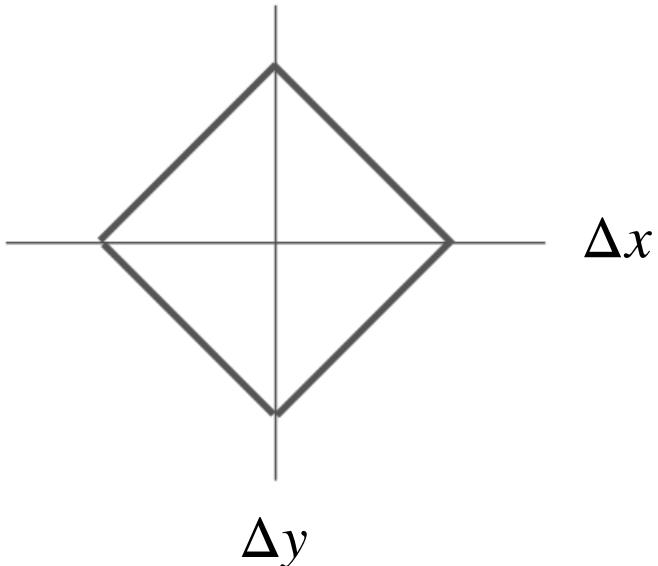


Distance Metric |  ,  |  $\rightarrow \mathbb{R}$

# Distance Metric

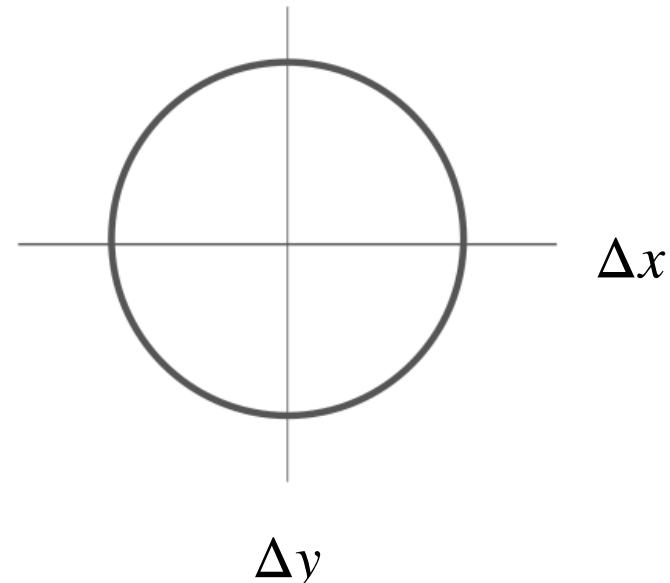
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



# Distance Metric

**L1 distance:**

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

-

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

=

add  
→ 456

# Nearest Neighbor Classifier

```
def train(images, labels):  
    # Machine learning!  
    return model
```



Memorize all  
data and labels

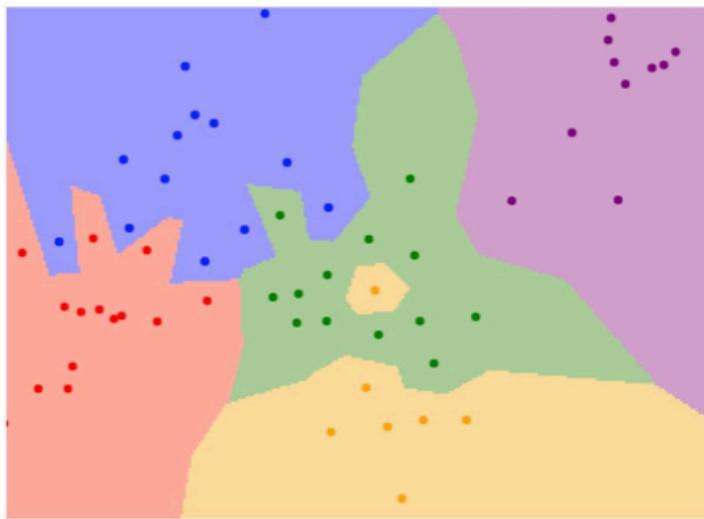
```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```



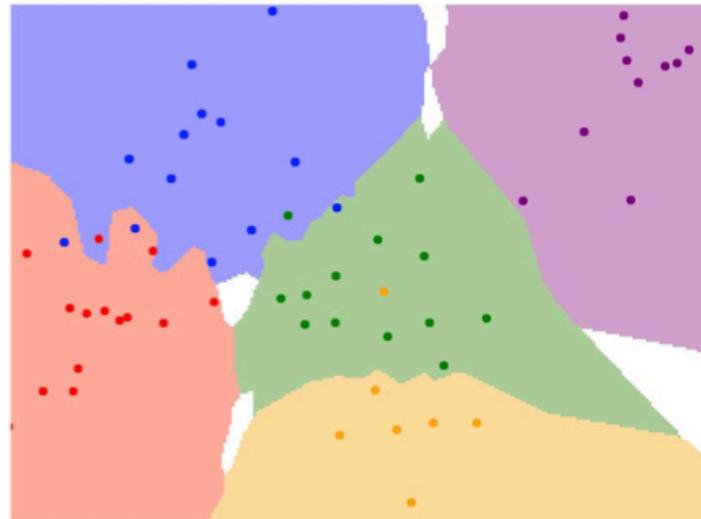
Predict the label  
of the most similar  
training image

# K-Nearest Neighbors

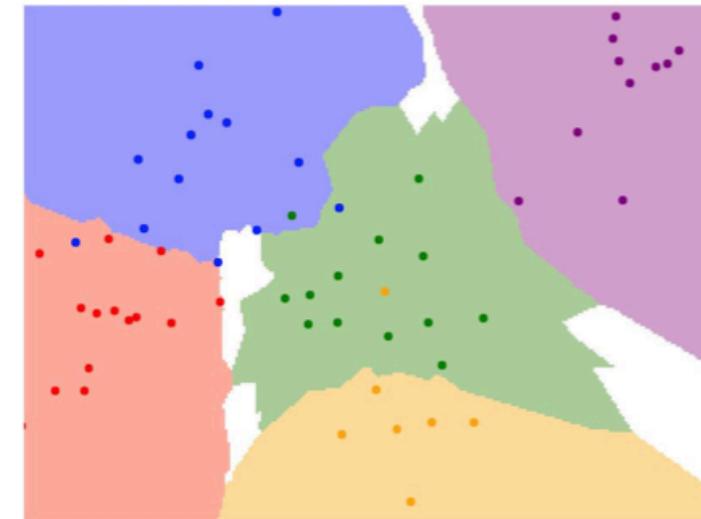
Instead of copying label from nearest neighbor,  
take **majority vote** from K closest points



$K = 1$



$K = 3$



$K = 5$

Similar strategy to RANSAC.

# Problems with Nearest Neighbor Classifier

- KNN with pixel distance never used.
  - Pixel distance as a metric is too sensitive to change in background/illumination/pose/viewpoint/occlusion/... that are not essential to the semantics.
- Very slow at test time.
- However, nearest neighbor-based techniques are still useful when the metrics is learned via a deep neural network and widely used in image retrieval, metric learning, 3D vision, and etc.

# Using CNN for Image Classification

- Things we need to take care:
  - Network architecture
  - Loss functions

# SoftMax vs. SVM

- For image classification, the most widely paradigm is **Softmax classifier + cross-entropy loss**.
- For binary classification, people also use SVM loss. However, the extended multiclass SVM loss is rarely used in current trend.

## Multiclass SVM loss:

Given an example  $(x_i, y_i)$  where  $x_i$  is the image and where  $y_i$  is the (integer) label,

and using the shorthand for the scores vector:  $s = f(x_i, W)$

the SVM loss has the form:

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} \\ &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \end{aligned}$$

# Softmax Classifier

- Also called multinomial logistic regression.



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax  
Function

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax as a Non-Linear Activation

- SoftMax, also known as Softargmax or normalized exponential function, is a generalization of the **logistic function** to multiple dimensions.
- Definition:  $\sigma(z) : z \in \mathbb{R}^K \rightarrow (0,1)^K$  when  $K > 1$

$$\sigma(z)_i = \frac{\exp(\beta z_i)}{\sum_{j=1}^K \exp(\beta z_j)}$$

- $\beta = 1$  by default.

# Softmax as a Non-Linear Activation

- SoftMax, also known as Softargmax or normalized exponential function, is a generalization of the **sigmoid function** to multiple dimensions.
- Definition:  $\sigma(z) : z \in \mathbb{R}^K \rightarrow (0,1)^K$  when  $K > 1$

$$\sigma(z)_i = \frac{\exp(\beta z_i)}{\sum_{j=1}^K \exp(\beta z_j)}$$

- $\beta = 1$  by default.
- When  $\beta \rightarrow \infty$ , Softmax  $\rightarrow$  argmax.
- When  $K = 2$ ,  $\sigma(\begin{bmatrix} z \\ 0 \end{bmatrix})_1 = \text{Sigmoid}(z)$

# SoftMax Classifier



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

cat  
car  
frog

3.2
5.1
-1.7

Unnormalized  
log-probabilities / logits

Probabilities  
must be  $\geq 0$

24.5
164.0
0.18

unnormalized  
probabilities

exp

normalize

0.13
0.87
0.00

probabilities

# SoftMax Classifier



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

cat  
car  
frog

**3.2**  
5.1  
-1.7

Probabilities  
must be  $\geq 0$

**24.5**  
164.0  
0.18

unnormalized  
probabilities

Probabilities  
must sum to 1

**0.13**  
0.87  
0.00

probabilities

**1.00**  
0.00  
0.00

Correct  
probs

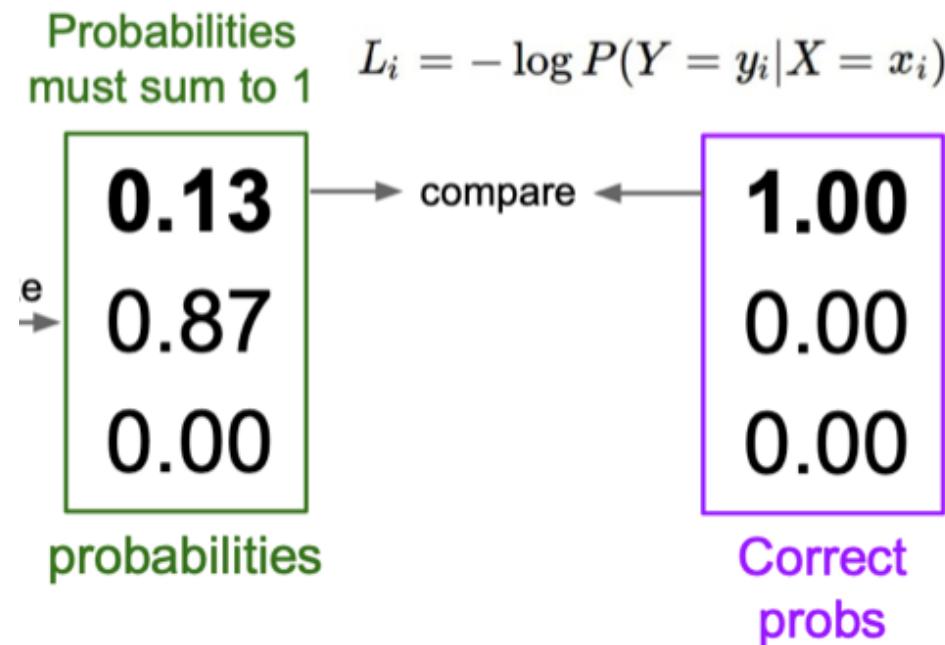
exp

normalize

compare

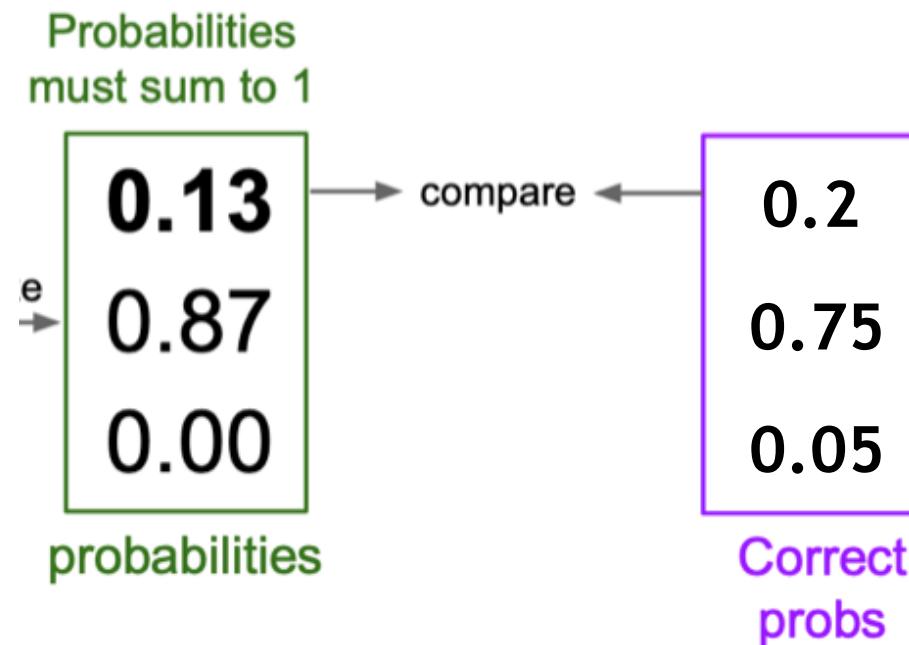
# Negative Log-likelihood Loss

- When correct probability is a one-hot vector, we can simply use negative log likelihood (NLL) loss.



# Loss between Two Discrete Distributions

- When correct probs is not one-hot, e.g. there is uncertainty in the label or the label is smoothed (label smoothing), we need to measure the difference between two distributions



Reference: [label smoothing](#)

# “Distance” between Two Distributions

- One widely used measure is Kullback-Leibler divergence  $D_{KL}(P \parallel Q)$ 
  - a measure of how one **probability distribution**  $Q$  is different from a second, reference probability distribution  $P$ .
  - For discrete probability distributions:

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right).$$

# Properties of KL Divergence

- Kullback-Leibler divergence  $D_{KL}(P \parallel Q)$

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right).$$

- $D_{KL}(P \parallel Q) \geq 0$  and  $D_{KL}(P \parallel Q) = 0$  only if  $P = Q$
- It is not a metric, since  $D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$  and does not satisfy the triangle inequality.

# From KL Divergence to Cross-Entropy

$$\cdot D_{KL}(P || Q) = - \sum_{x \in \mathcal{X}} P(x) \log Q(x) - \left( - \sum_{x \in \mathcal{X}} P(x) \log P(x) \right)$$



$$H(P, Q)$$



$$H(P)$$

- Entropy  $H(P)$
- Cross entropy  $H(P, Q)$

# From KL Divergence to Cross-Entropy

$$\bullet D_{KL}(P || Q) = - \sum_{x \in \mathcal{X}} P(x) \log Q(x) - \left( - \sum_{x \in \mathcal{X}} P(x) \log P(x) \right)$$

$$H(P, Q) \qquad \qquad H(P)$$

- Entropy  $H(P)$ , cross entropy  $H(P, Q)$
- When  $P$  is the ground truth prob.,  $Q$  is the predicted prob.,  $H(P)$  is a constant.
- **Cross entropy loss:**  $\mathcal{L}_{CE} = H(P, Q) = - \sum_{x \in \mathcal{X}} P(x) \log Q(x)$

# Properties of Cross Entropy Loss

$$\mathcal{L}_{CE} = H(P, Q) = - \sum_{x \in \mathcal{X}} P(x) \log Q(x)$$

- With random initialization,  $\mathcal{L}_{CE} \approx 1/(\# \text{ of classes})$
- No upper bound.
- Minimum = 0.

# Summary of Softmax Classifier



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax  
Function

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

3.2

5.1

-1.7

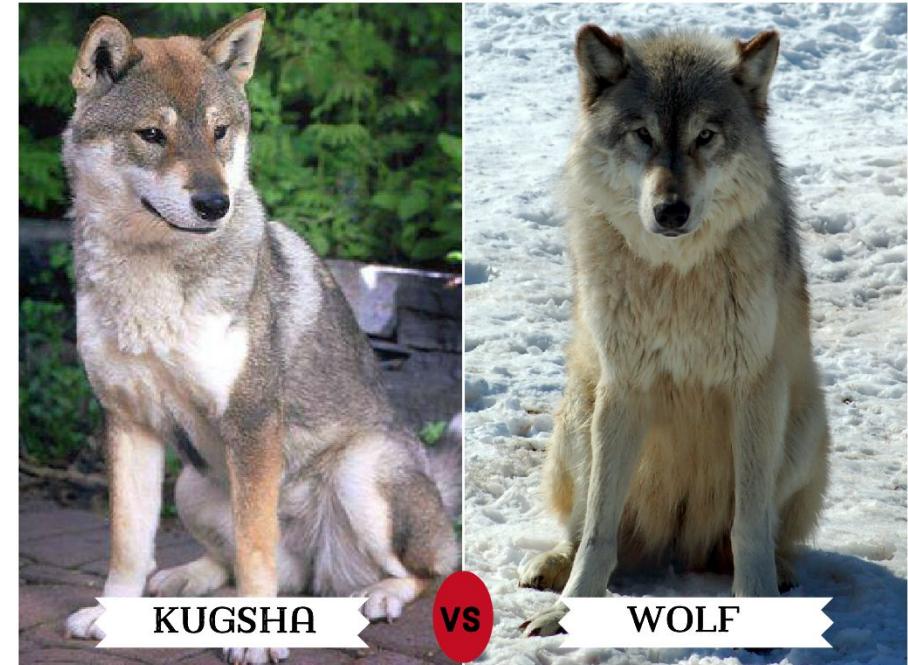
# CNNs for Image Classification

# How to Analyze a CNN Architecture

- Expressivity/Capacity
- Fitness for task
- Optimization properties
- Cost

# What Do We Need for a CNN on ImageNet?

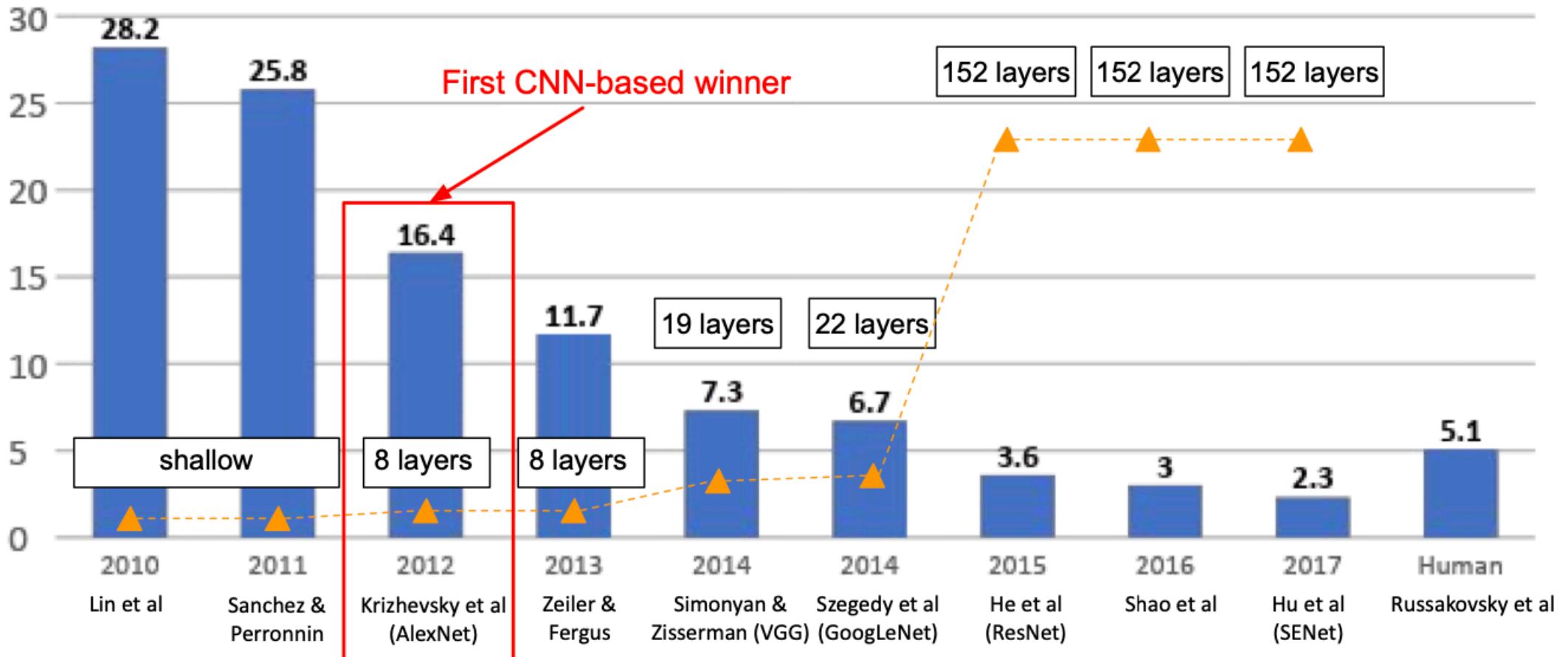
- ImageNet Challenges contains 1000 classes
  - 90 different dog breeds
- The network need to look at
  - details
  - context



# How to Analyze a CNN Architecture

- Expressivity/Capacity
  - The deeper and the wider a network is, theoretically the larger its capacity is.
- Fitness for task
- Optimization properties
- Cost

# The History: ImageNet Challenge Winners



# AlexNet

## Architecture:

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

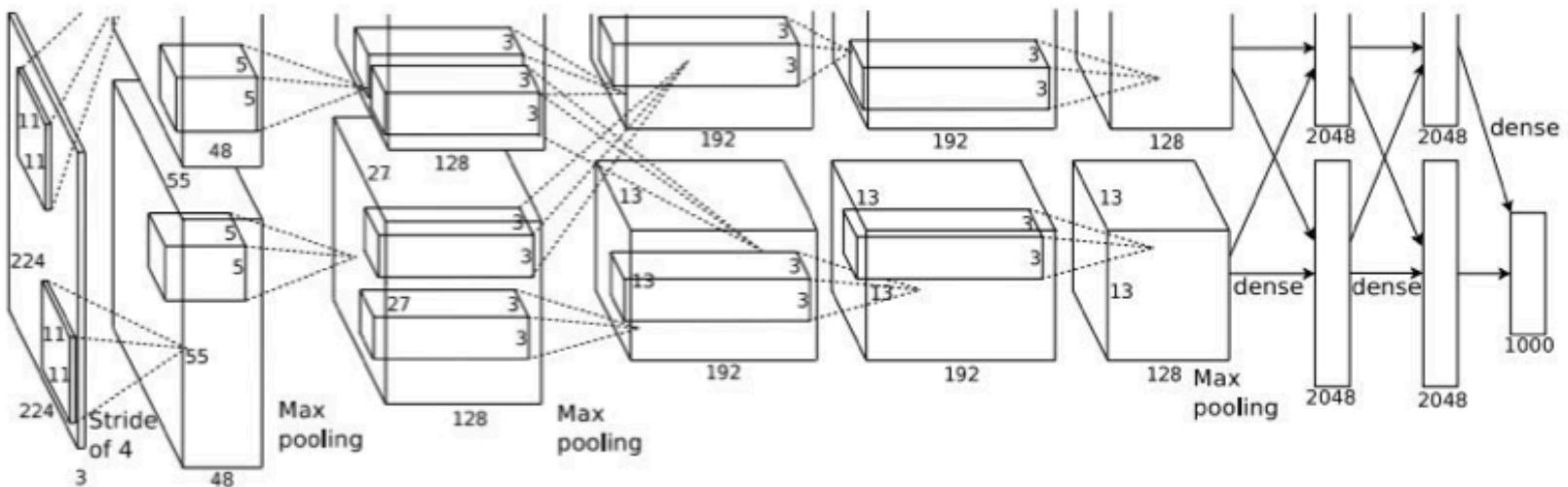
CONV5

Max POOL3

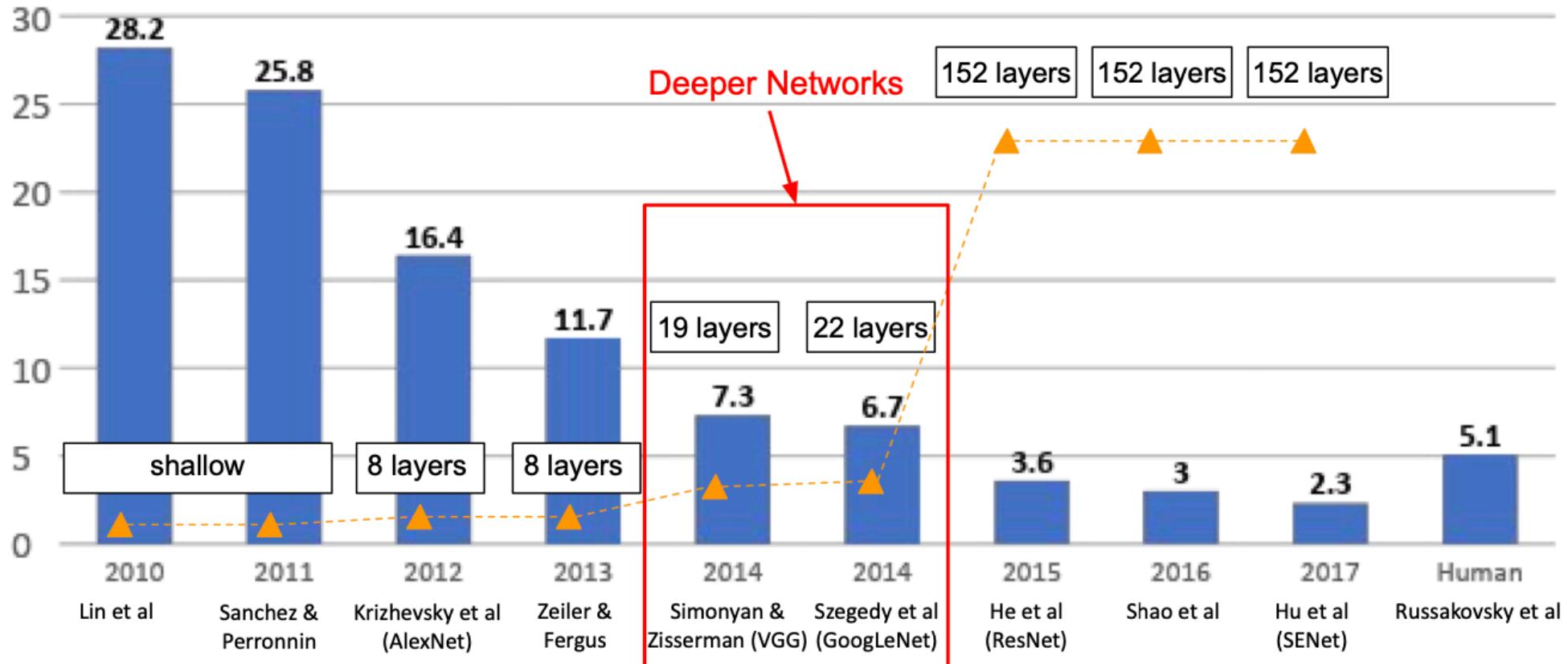
FC6

FC7

FC8



# The History: ImageNet Challenge Winners



# VGGNet

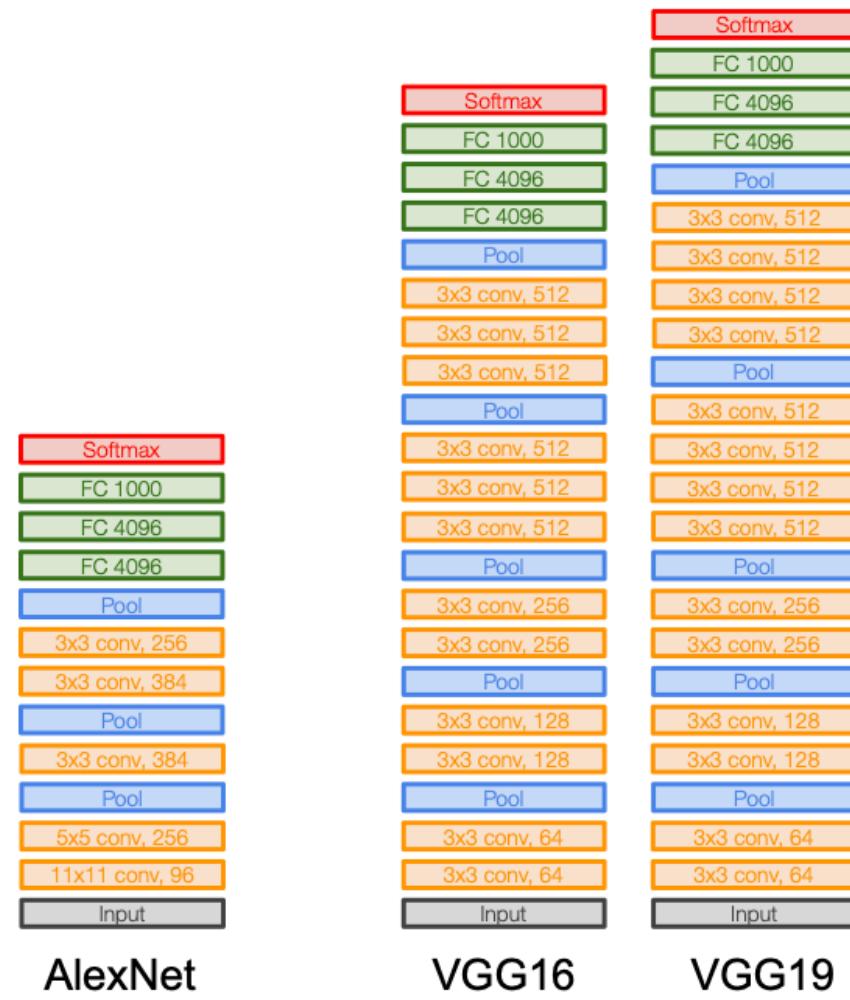
## Small filters, Deeper networks

## 8 layers (AlexNet)

-> 16 - 19 layers (VGG16Net)

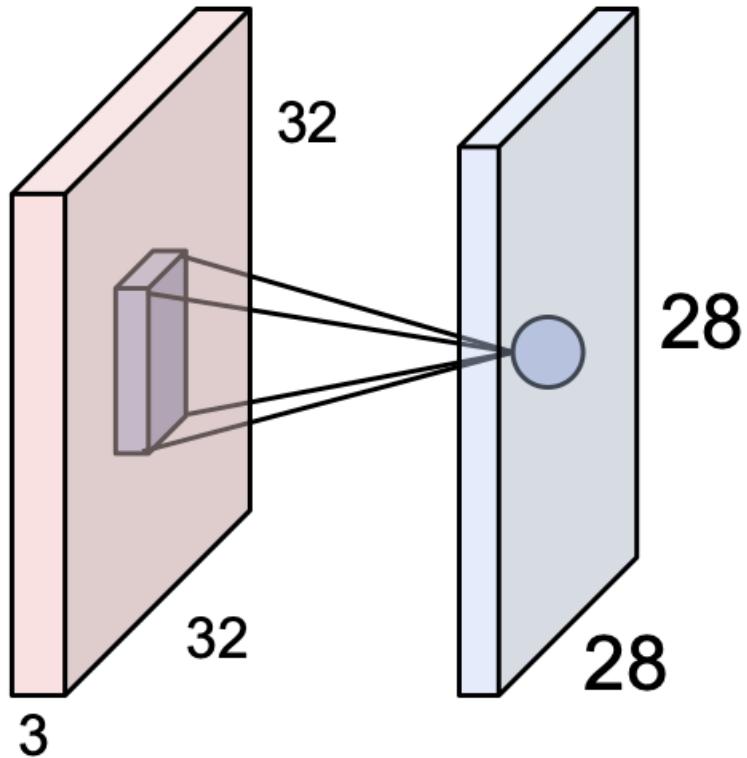
Only 3x3 CONV stride 1, pad 1  
and 2x2 MAX POOL stride 2

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



# Why use smaller filters? ( $3 \times 3$ conv)

# Receptive Field



An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

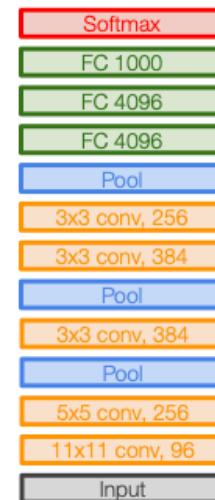
“5x5 filter”  $\rightarrow$  “5x5 receptive field for each neuron”

# VGGNet

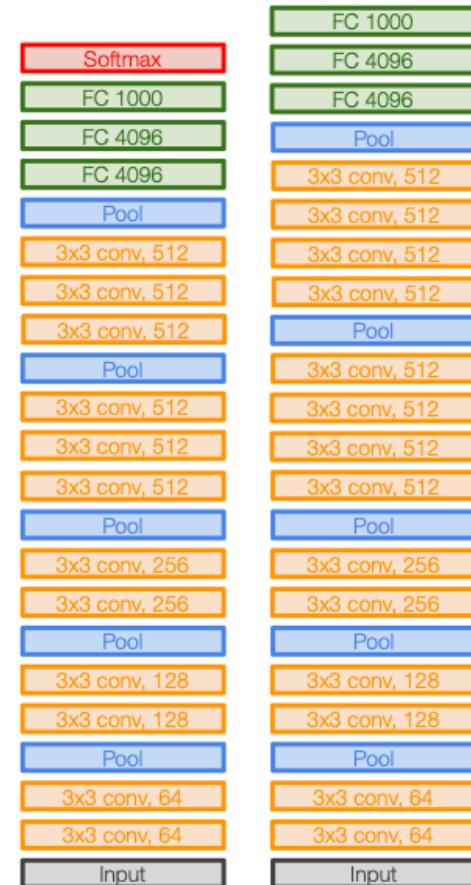
**Q: Why use smaller filters? (3x3 conv)**

Stack of three  $3 \times 3$  conv (stride 1) layers  
has same **effective receptive field** as  
one  $7 \times 7$  conv layer

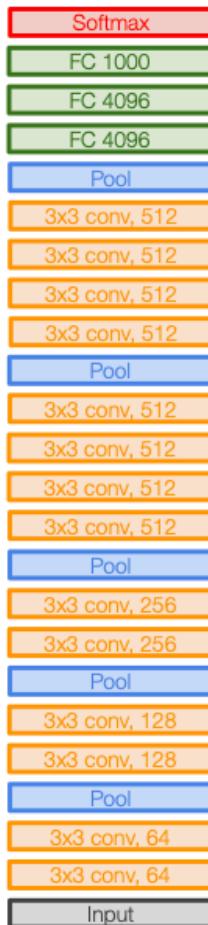
**Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?**



## AlexNet



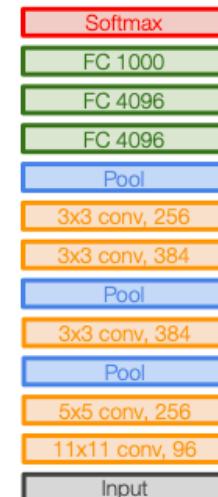
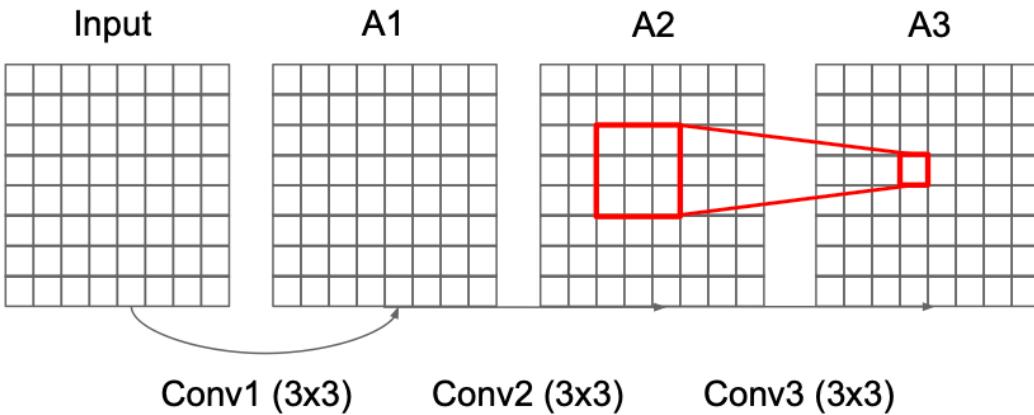
VGG16



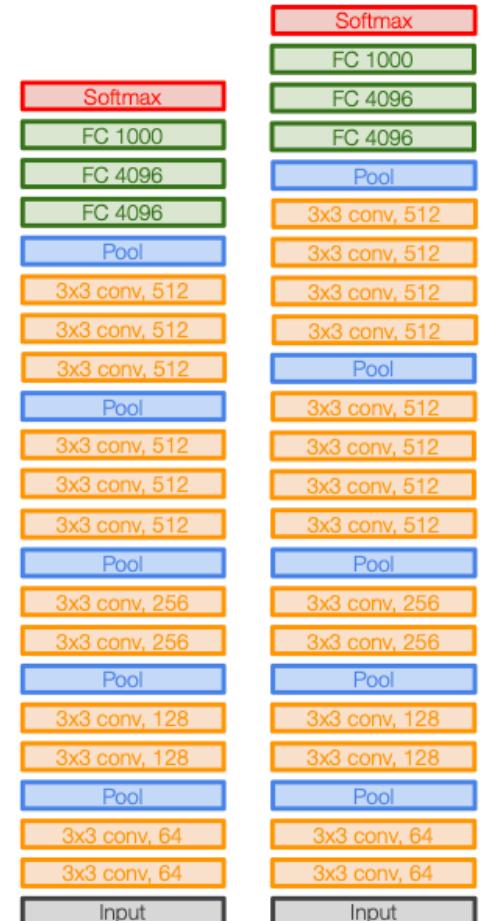
VGG19

# Receptive Field

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



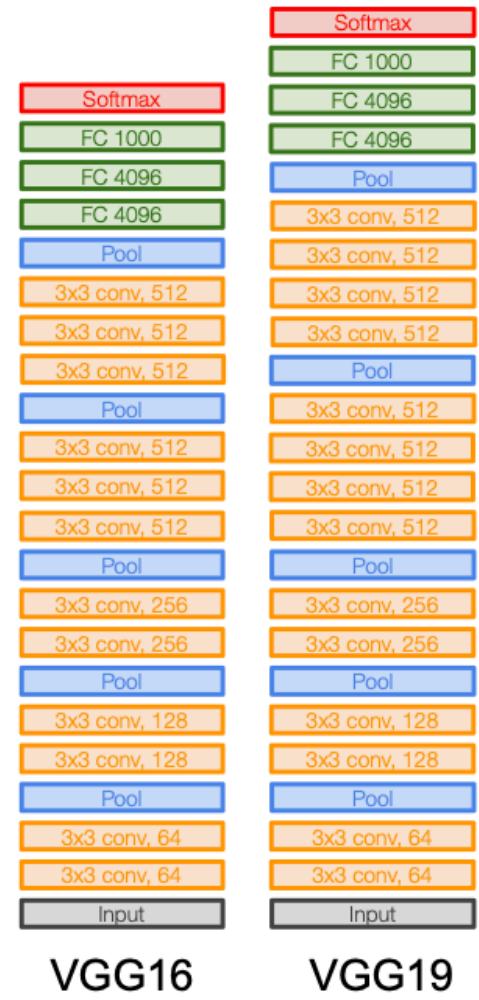
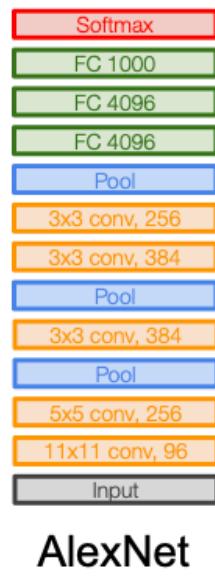
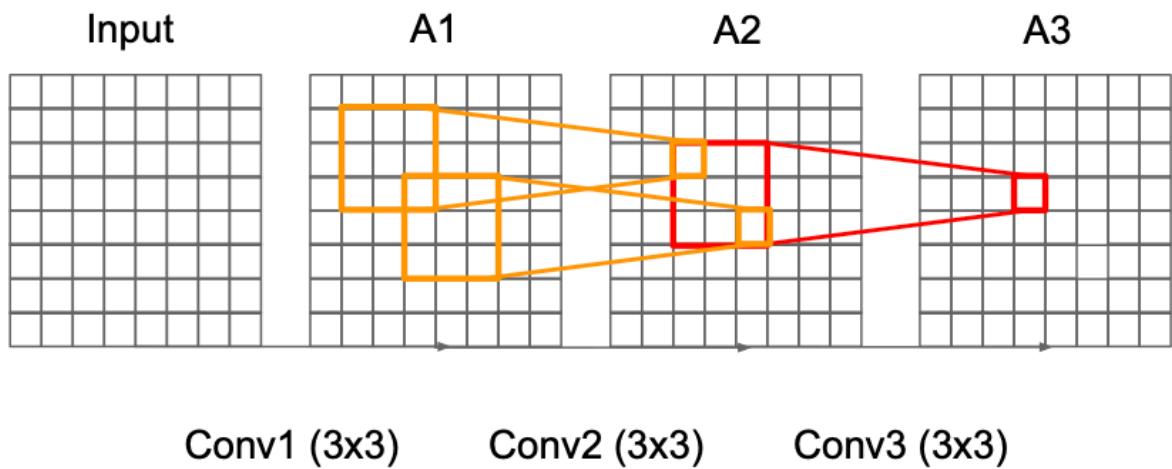
AlexNet



VGG16      VGG19

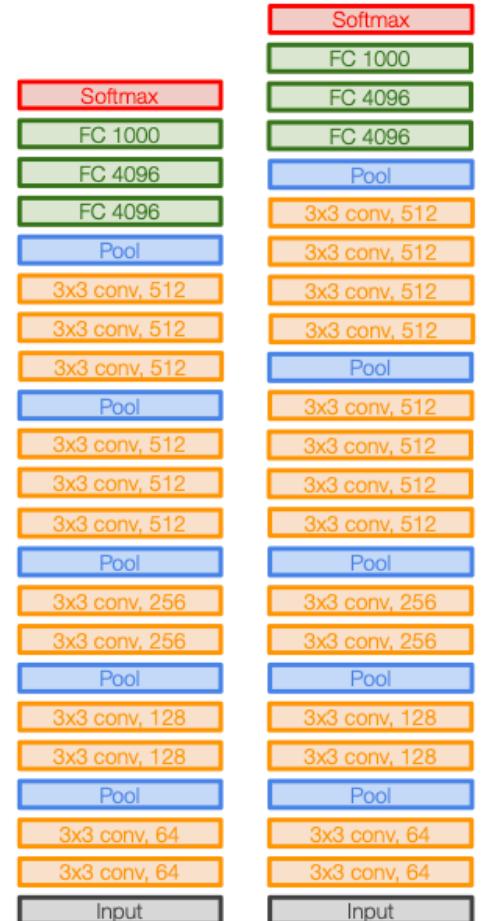
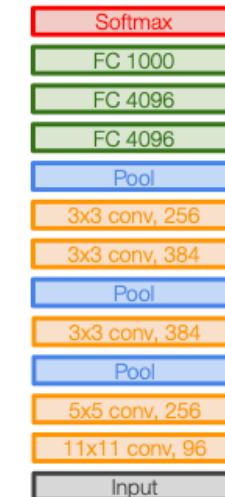
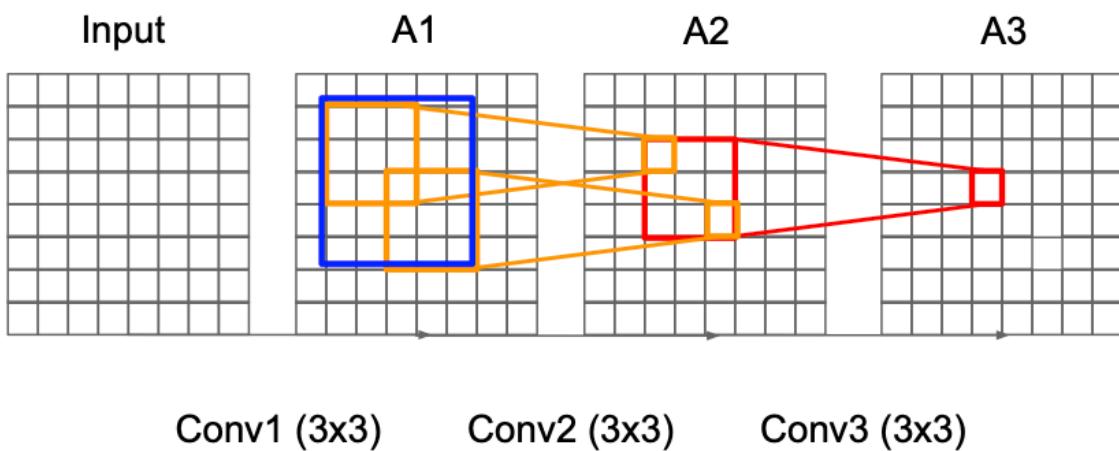
# Receptive Field

**Q: What is the effective receptive field of three  $3 \times 3$  conv (stride 1) layers?**



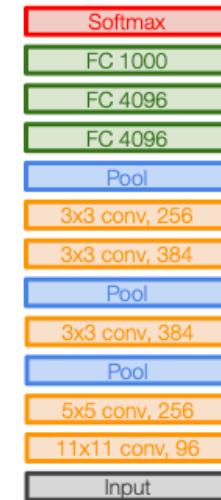
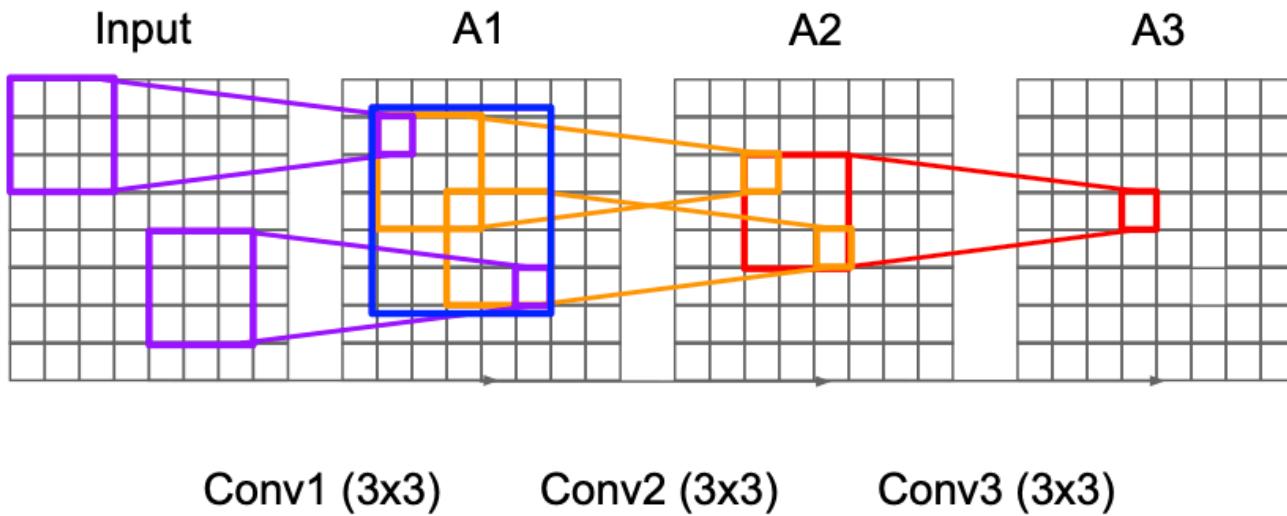
# Receptive Field

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

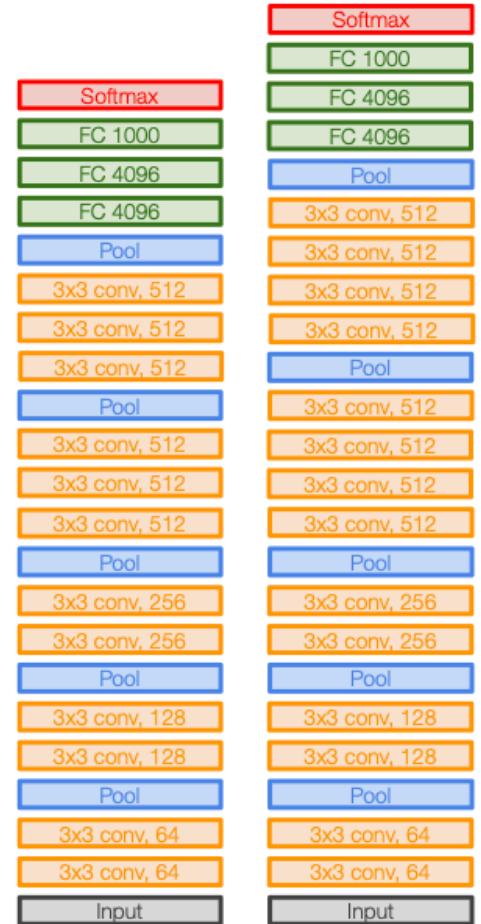


# Receptive Field

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



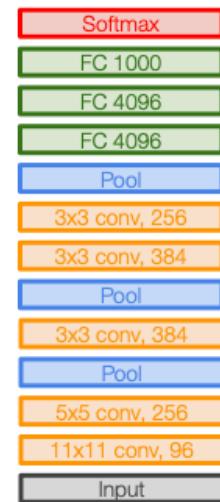
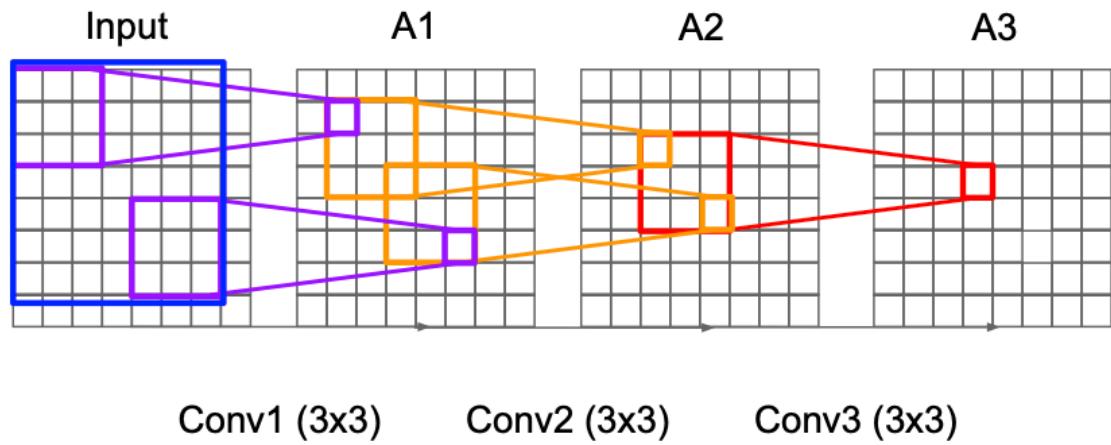
AlexNet



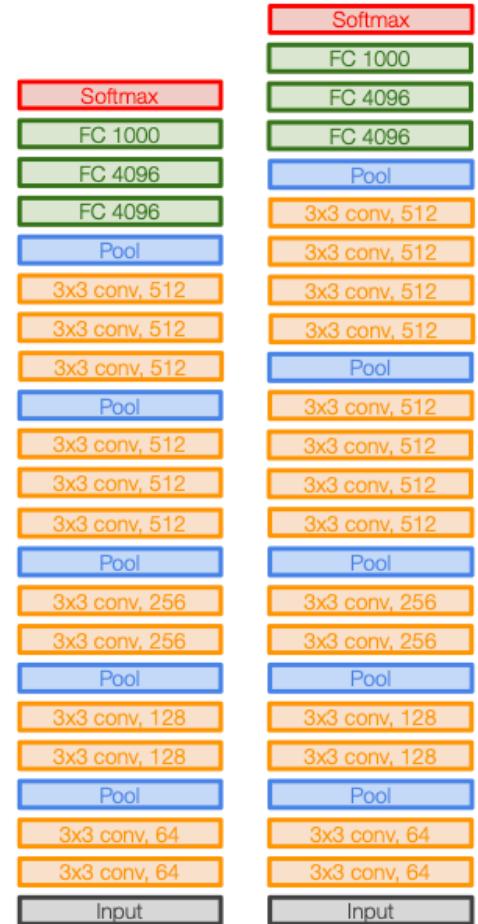
VGG16      VGG19

# Receptive Field

**Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?**



## AlexNet



VGG16

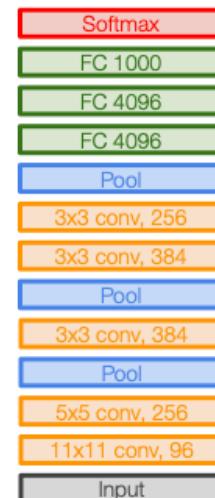
VGG19

# Receptive Field

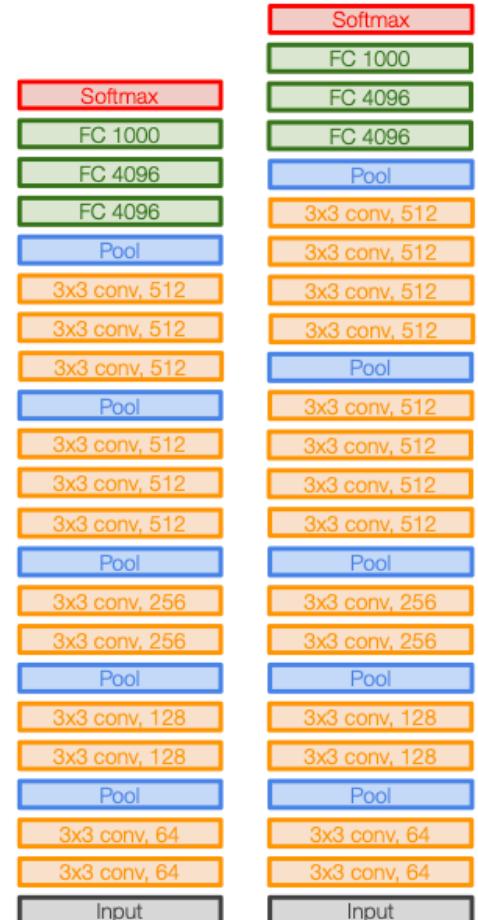
**Q: Why use smaller filters? (3x3 conv)**

Stack of three  $3 \times 3$  conv (stride 1) layers has same **effective receptive field** as one  $7 \times 7$  conv layer

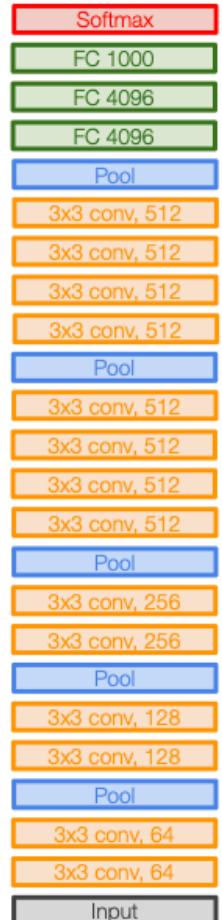
[7x7]



## AlexNet



VGG16



VGG19

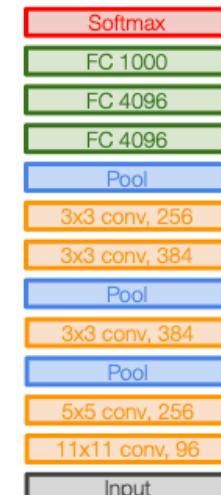
# Receptive Field

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

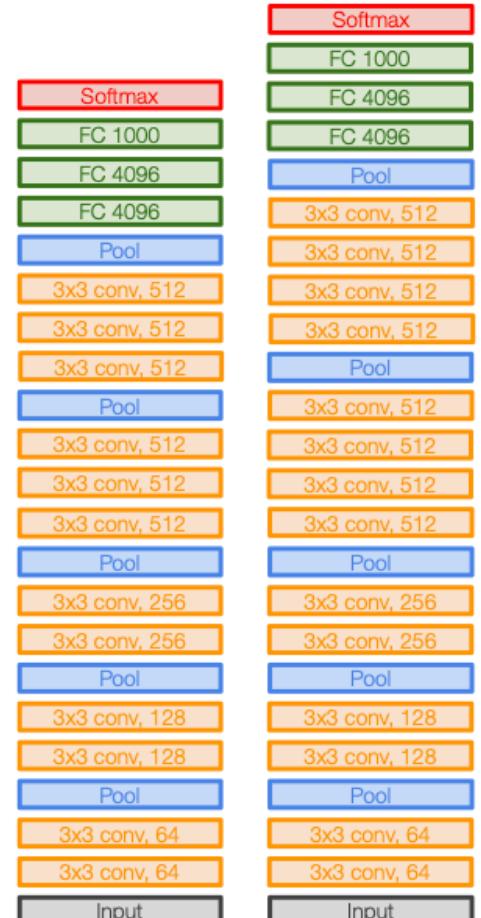
Stack of three 3x3 conv (stride 1) layers  
has same **effective receptive field** as  
one 7x7 conv layer

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

# Receptive Field

INPUT: [224x224x3] memory:  $224 \times 224 \times 3 = 150K$  params: 0 (not counting biases)

CONV3-64: [224x224x64] memory:  $224 \times 224 \times 64 = 3.2M$  params:  $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory:  $224 \times 224 \times 64 = 3.2M$  params:  $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory:  $112 \times 112 \times 64 = 800K$  params: 0

CONV3-128: [112x112x128] memory:  $112 \times 112 \times 128 = 1.6M$  params:  $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory:  $112 \times 112 \times 128 = 1.6M$  params:  $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory:  $56 \times 56 \times 128 = 400K$  params: 0

CONV3-256: [56x56x256] memory:  $56 \times 56 \times 256 = 800K$  params:  $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory:  $56 \times 56 \times 256 = 800K$  params:  $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory:  $56 \times 56 \times 256 = 800K$  params:  $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory:  $28 \times 28 \times 256 = 200K$  params: 0

CONV3-512: [28x28x512] memory:  $28 \times 28 \times 512 = 400K$  params:  $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory:  $28 \times 28 \times 512 = 400K$  params:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory:  $28 \times 28 \times 512 = 400K$  params:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory:  $14 \times 14 \times 512 = 100K$  params: 0

CONV3-512: [14x14x512] memory:  $14 \times 14 \times 512 = 100K$  params:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory:  $14 \times 14 \times 512 = 100K$  params:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory:  $14 \times 14 \times 512 = 100K$  params:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory:  $7 \times 7 \times 512 = 25K$  params: 0

FC: [1x1x4096] memory: 4096 params:  $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params:  $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params:  $4096 \times 1000 = 4,096,000$

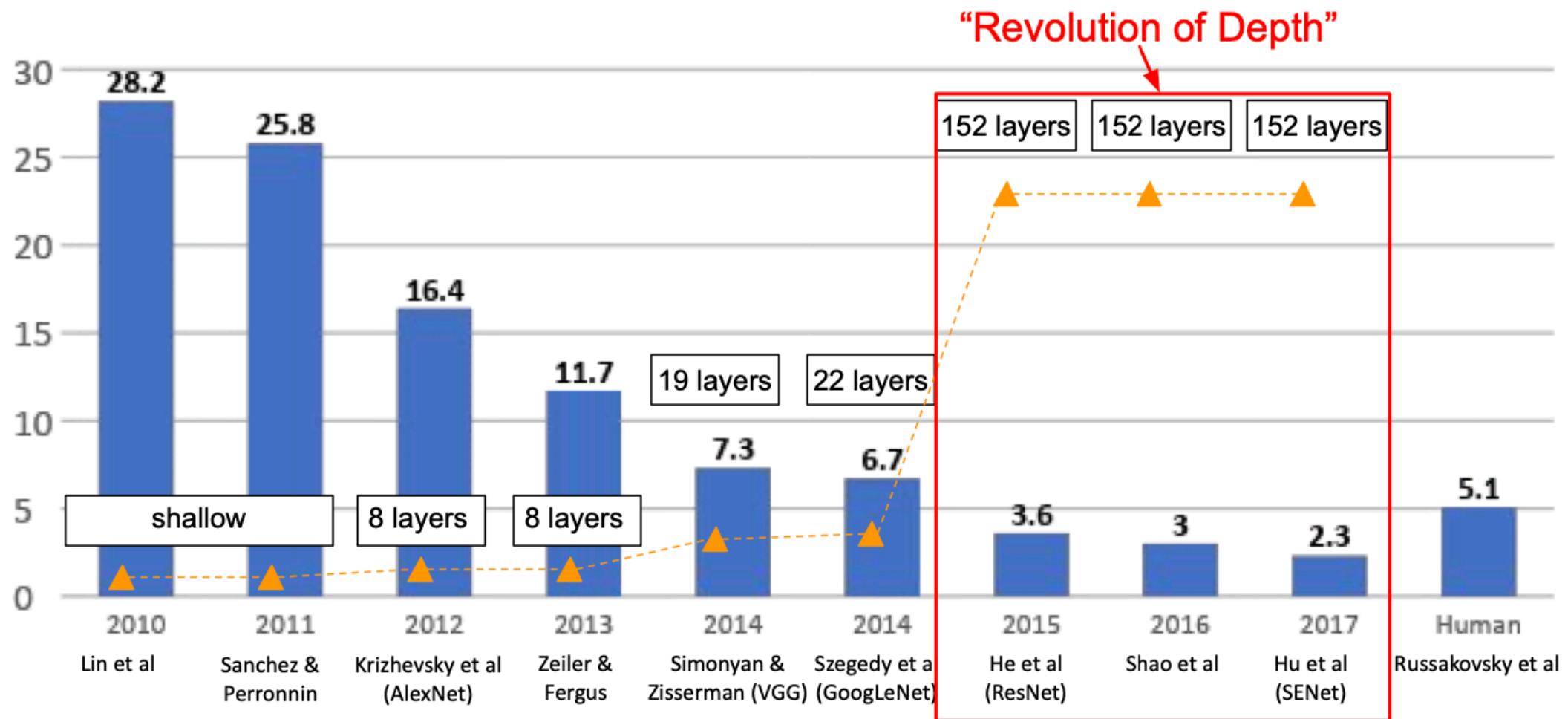
**TOTAL memory:**  $24M * 4 \text{ bytes} \approx 96\text{MB} / \text{image}$  (only forward!  $\sim 2$  for bwd)

**TOTAL params:** 138M parameters



VGG16  
Common names

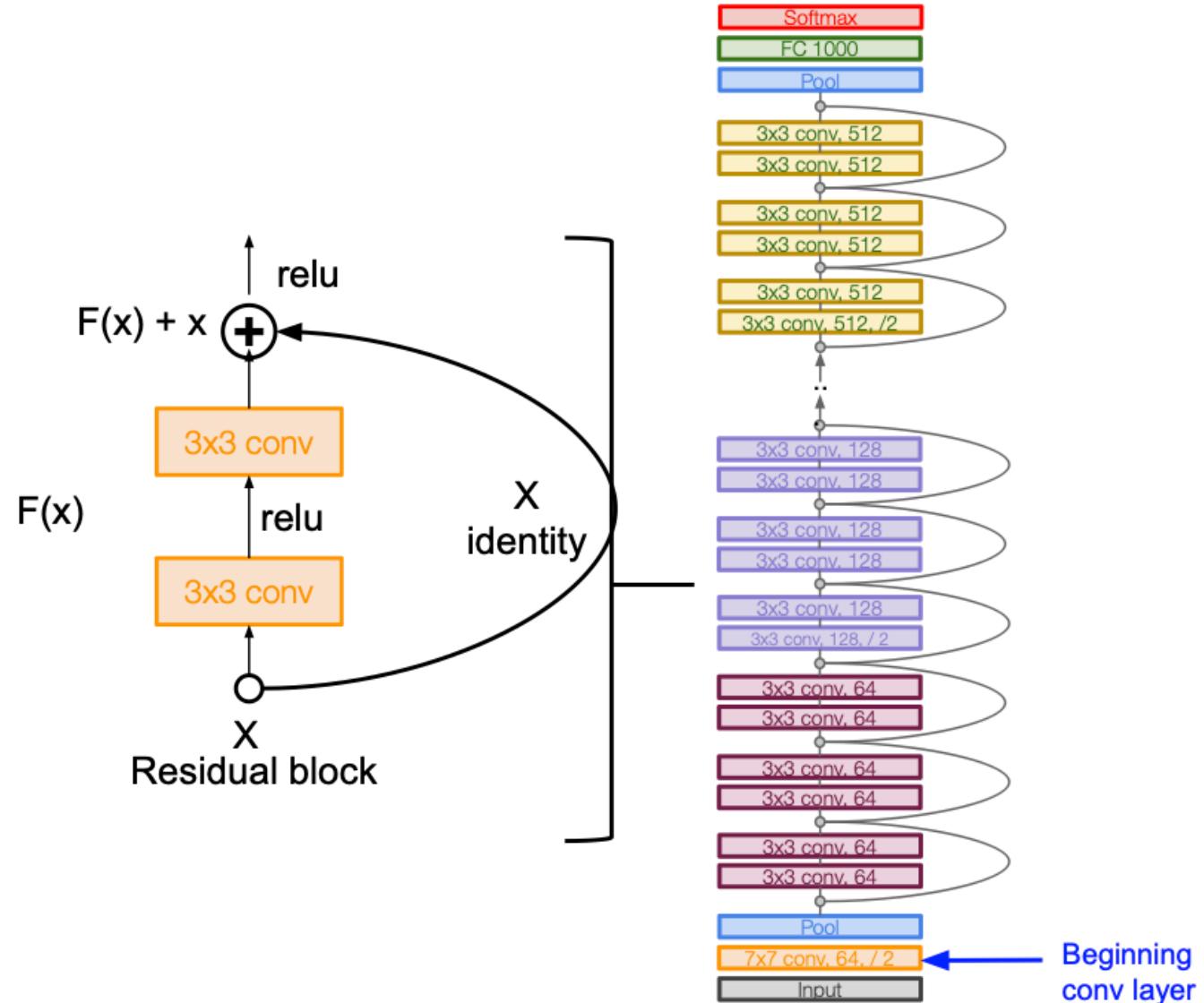
# ResNet



# ResNet

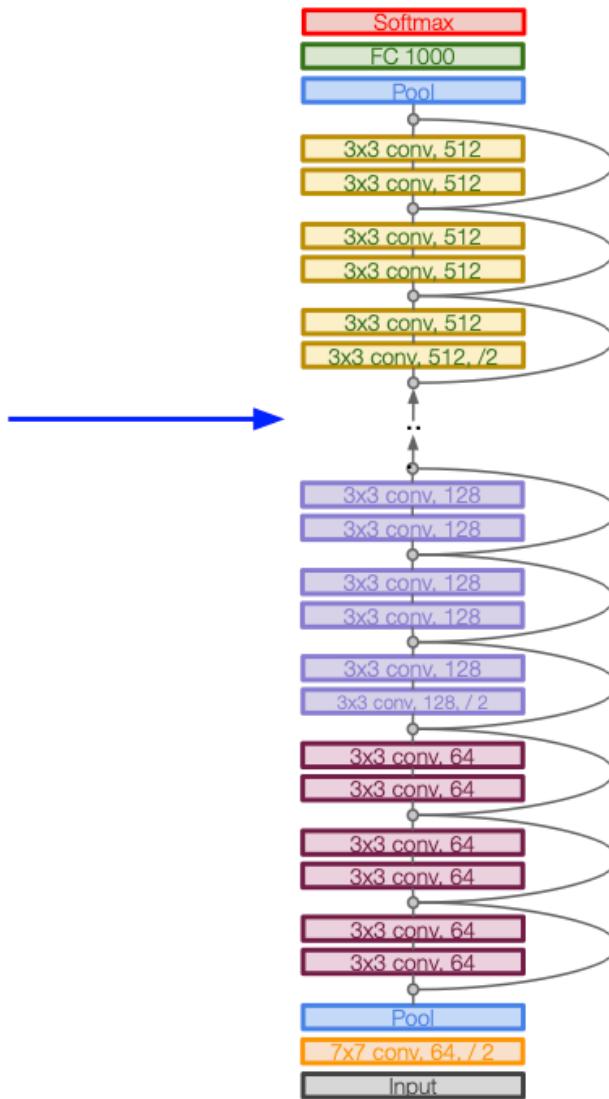
## Full ResNet architecture:

- 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 (stem)



# ResNet

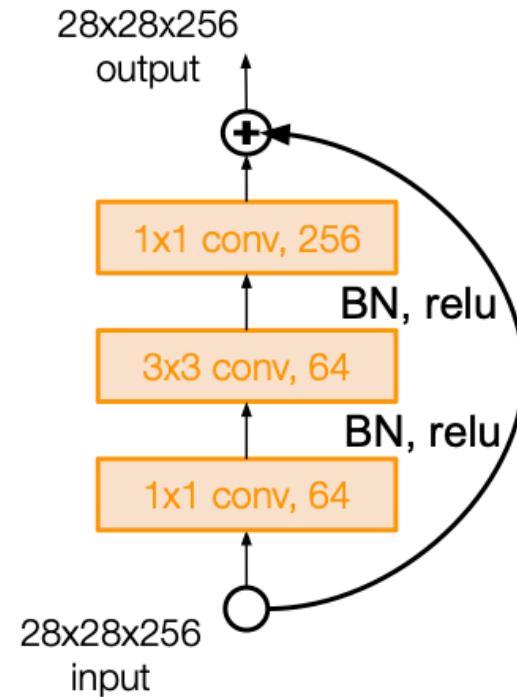
Total depths of 18, 34, 50,  
101, or 152 layers for  
ImageNet



# ResNet

[He et al., 2015]

For deeper networks  
(ResNet-50+), use “bottleneck”  
layer to improve efficiency  
(similar to GoogLeNet)



# ResNet

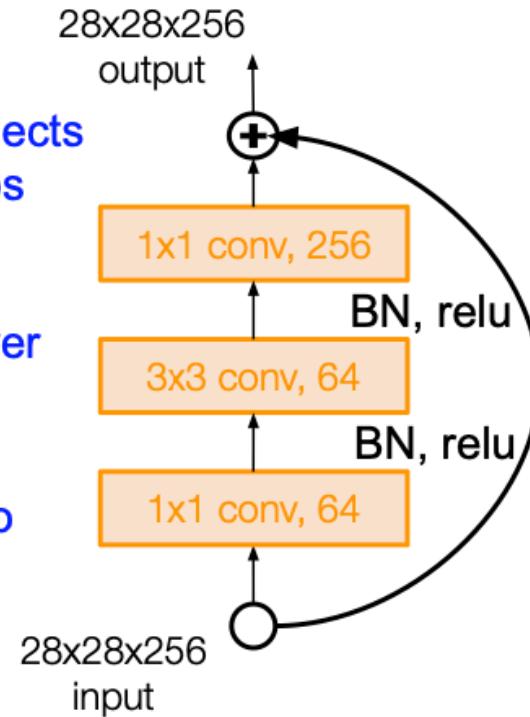
[He et al., 2015]

For deeper networks  
(ResNet-50+), use “bottleneck”  
layer to improve efficiency  
(similar to GoogLeNet)

1x1 conv, 256 filters projects  
back to 256 feature maps  
(28x28x256)

3x3 conv operates over  
only 64 feature maps

1x1 conv, 64 filters to  
project to 28x28x64



# ResNet

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

# ResNet

## Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

### MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places** in all five main tracks
  - ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer** nets
  - ImageNet Detection: **16%** better than 2nd
  - ImageNet Localization: **27%** better than 2nd
  - COCO Detection: **11%** better than 2nd
  - COCO Segmentation: **12%** better than 2nd

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than “human performance”! (Russakovsky 2014)

# Beyond ResNet

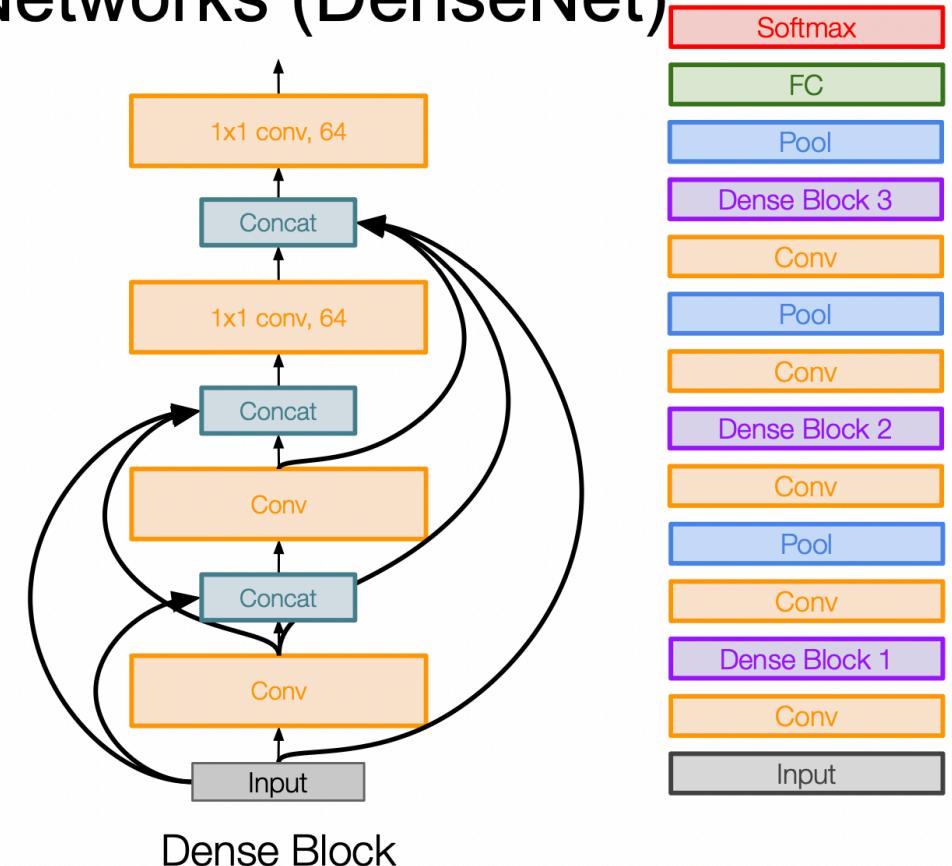
- Squeeze-and-Excitation Network (SENet)
- Wide Residual Networks
- ResNeXt
- DenseNet

# DenseNet

## Densely Connected Convolutional Networks (DenseNet)

[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
- Showed that shallow 50-layer network can outperform deeper 152 layer ResNet



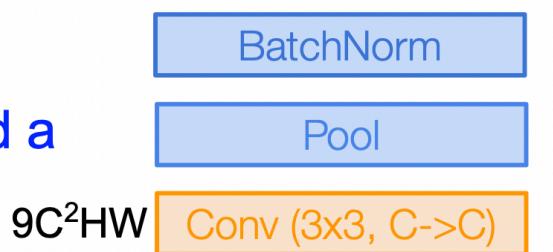
# Beyond ResNet

- Squeeze-and-Excitation Network (SENet)
- Wide Residual Networks
- ResNeXt
- DenseNet
- Attention-based networks: ViT, SwinTransformer
- MLP-based networks
- MobileNet → efficiency

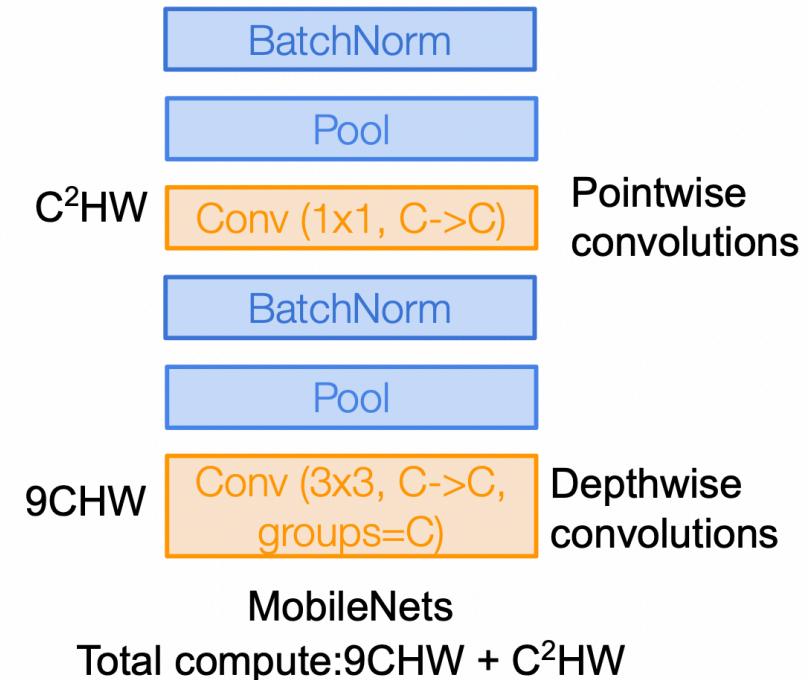
# Efficient Networks

## MobileNets: Efficient Convolutional Neural Networks for Mobile Applications *[Howard et al. 2017]*

- Depthwise separable convolutions replace standard convolutions by factorizing them into a depthwise convolution and a  $1 \times 1$  convolution
- Much more efficient, with little loss in accuracy
- Follow-up MobileNetV2 work in 2018 (Sandler et al.)
- ShuffleNet: Zhang et al., CVPR 2018



Standard network  
Total compute:  $9C^2HW$



# Beyond ResNet

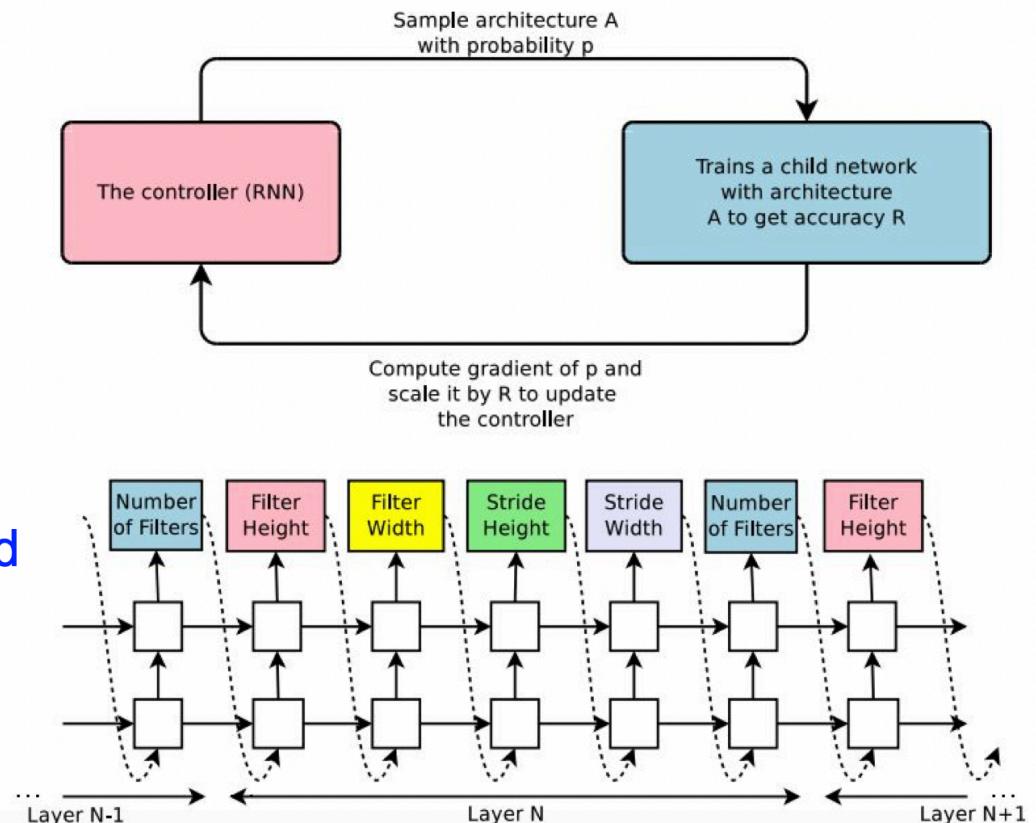
- Squeeze-and-Excitation Network (SENet)
- Wide Residual Networks
- ResNeXt
- DenseNet
- ViT, swinTransformer, MLP-based networks
- MobileNet → efficiency
- Neural architecture search

# Learning to Search for Network Architecture

## Neural Architecture Search with Reinforcement Learning (NAS)

[Zoph et al. 2016]

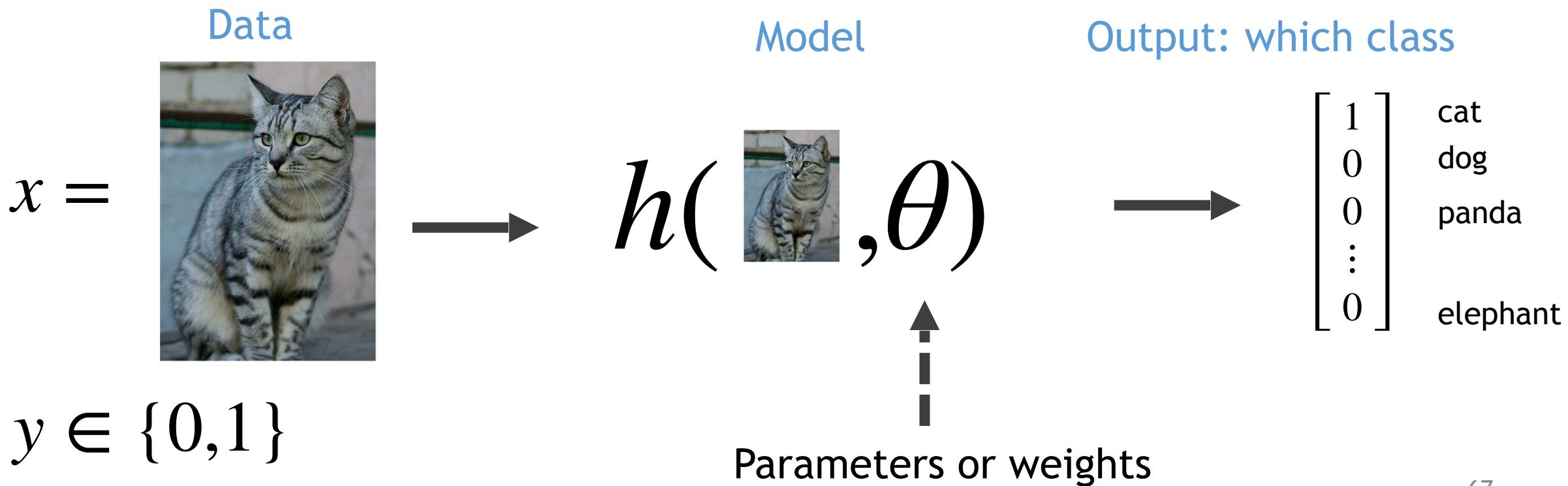
- “Controller” network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
  - 1) Sample an architecture from search space
  - 2) Train the architecture to get a “reward”  $R$  corresponding to accuracy
  - 3) Compute gradient of sample probability, and scale by  $R$  to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)



# Segmentation

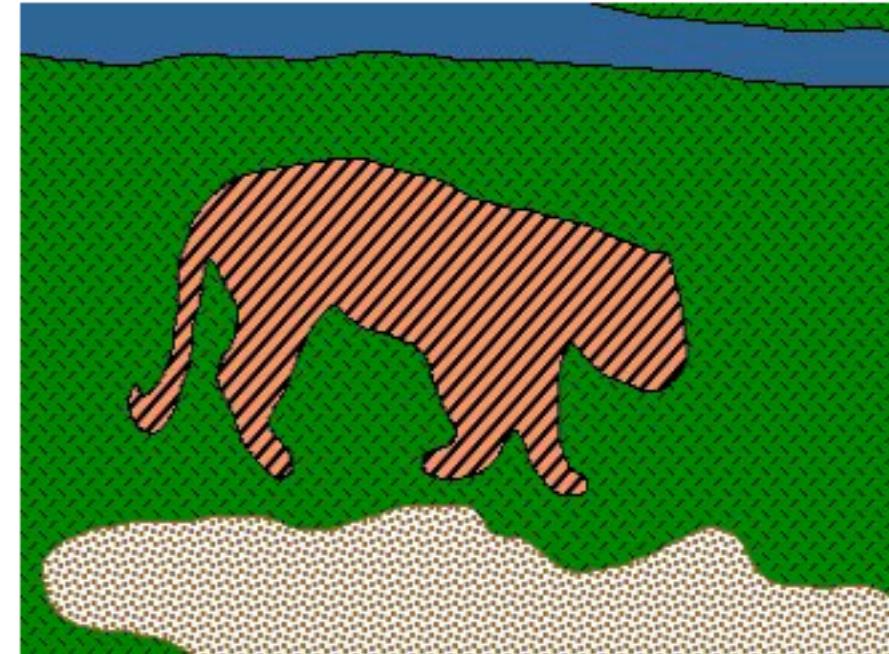
# Image Classification

- Classic definition: image classification is to categorize an image into several known classes ( $N$ ).



# Image Segmentation

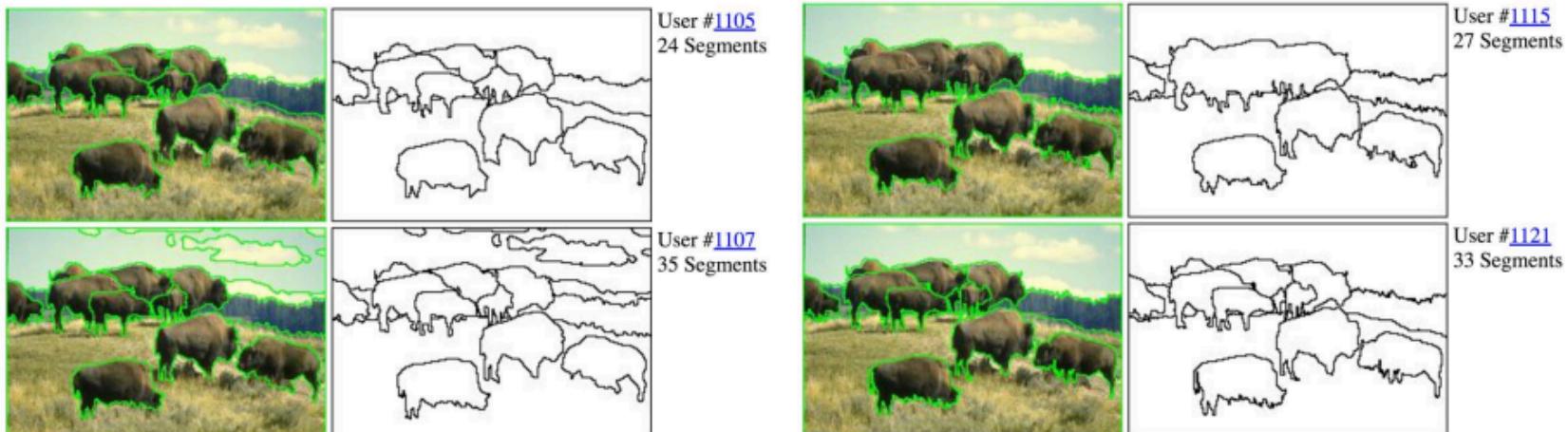
- Goal: identify groups of pixels that go together
  - Care about spatial extent
  - But not a global label



# Image Segmentation

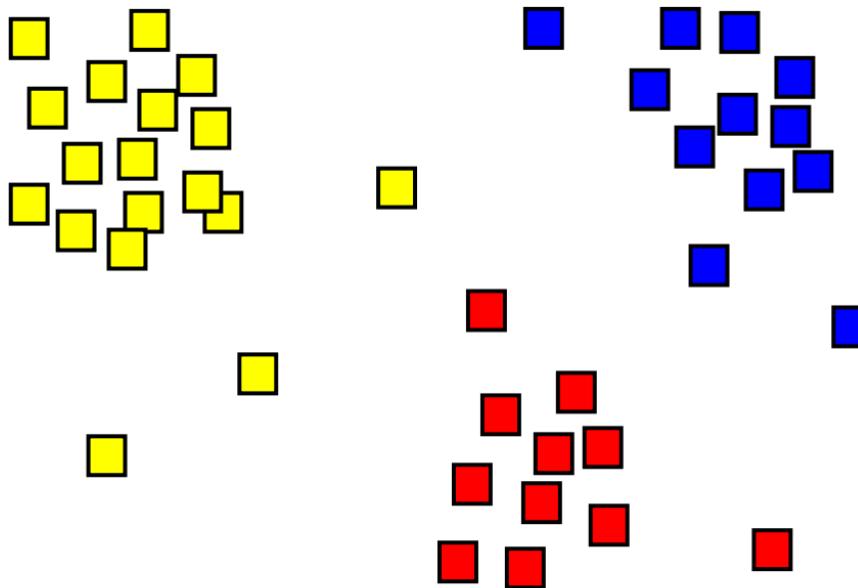
- Objective
  - separate image into coherent “objects”
  - however, doesn’t care about semantics

*“Divide each image into pieces, where each piece represents a distinguished thing in the image. It is important that all of the pieces have approximately equal importance.”*

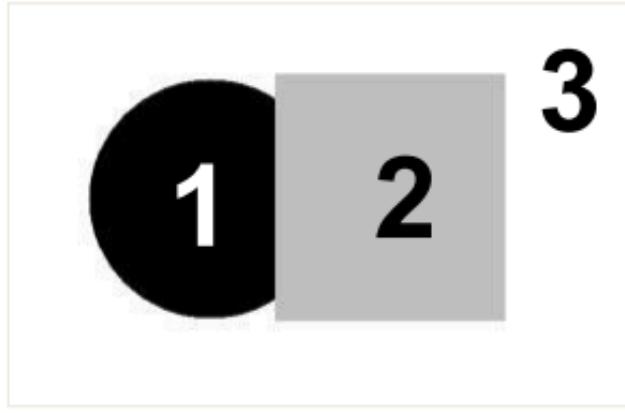


# Classic Techniques for Image Segmentation

- Philosophy: grouping and clustering
  - Clustering: group together similar data points and represents them with a single token

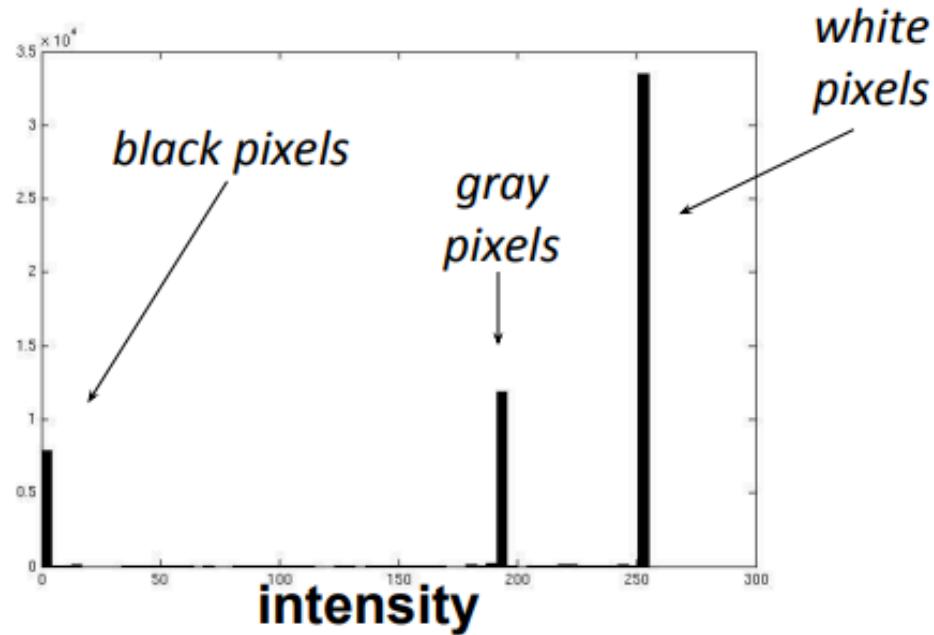


# Clustering: Toy Example

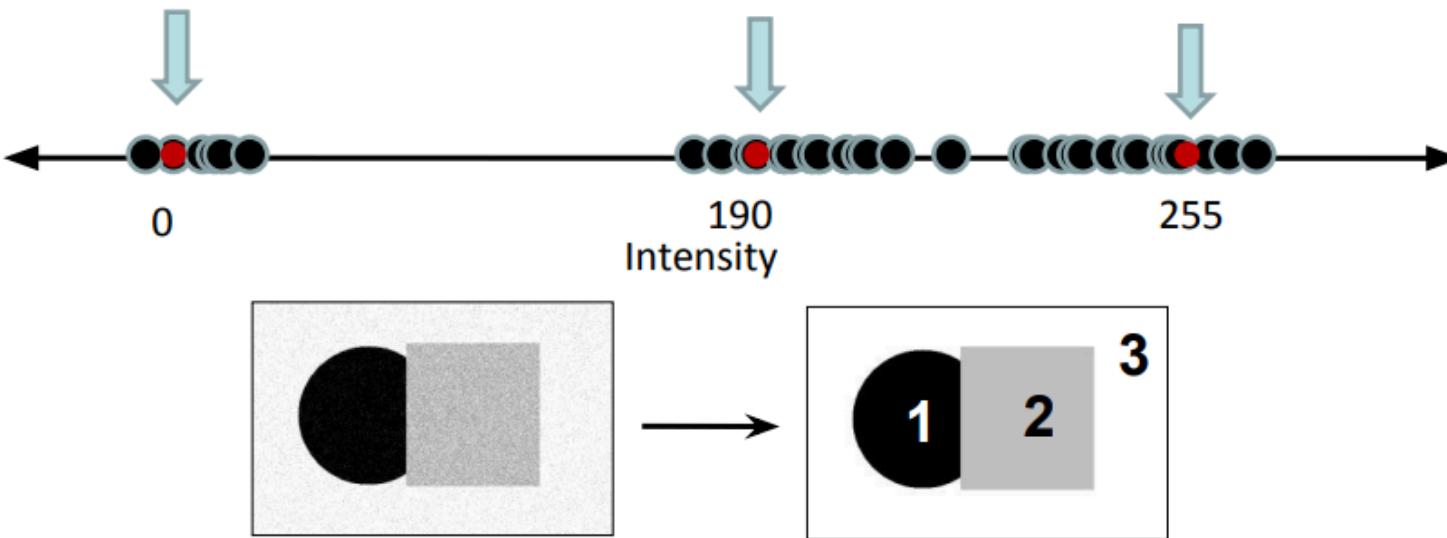


**input image**

- These intensities define the
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., segment the image based on the intensity feature.
- What if the image isn't quite so simple?



# Clustering-based Segmentation



- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize Sum of Square Distance (SSD) between all points and their nearest cluster center  $c_i$ :

$$SSD = \sum_i^k \sum_{x \in c_i} (x - c_i)^2$$

# Objective Function

- Goal: minimize the distortion in data given clusters
  - Preserve information

$$c^*, \delta^* = \arg \min_{c, \delta} \frac{1}{N} \sum_j^N \sum_i^k \delta_{ij} (c_i - x_j)^2$$

Cluster center      Data  
                          ↓  
                          ↓  
                          Whether  $x_j$  is assigned to  $c_i$

# Clustering

- With this objective, it is a “chicken and egg” problem:
  - If we knew the *cluster centers*, we could allocate points to groups by assigning each to its closest center.



- If we knew the *group memberships*, we could get the centers by computing the mean per group.



# K-Means Clustering

- Initialization:
  - choose  $k$  cluster centers
- Repeat:
  - **assignment step:**
    - For every point find its closest center
  - **update step:**
    - Update every center as the mean of its points
- Until:
  - The maximum number of iterations is reached, or
  - No changes during the assignment step, or
  - The average distortion per point drops very little

# K-Means Clustering

- **Input:**  $N$  examples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  ( $\mathbf{x}_n \in \mathbb{R}^D$ ); the number of partitions  $K$
- **Initialize:**  $K$  cluster centers  $\mu_1, \dots, \mu_K$ . Several initialization options:
  - Randomly initialized anywhere in  $\mathbb{R}^D$
  - Choose any  $K$  examples as the cluster centers
- **Iterate:**
  - Assign each of example  $\mathbf{x}_n$  to its closest cluster center

$$\mathcal{C}_k = \{n : k = \arg \min_k \|\mathbf{x}_n - \mu_k\|^2\}$$

( $\mathcal{C}_k$  is the set of examples closest to  $\mu_k$ )

- Recompute the new cluster centers  $\mu_k$  (mean/centroid of the set  $\mathcal{C}_k$ )

$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

- Repeat while not converged

# K-Means

- A heuristics or coordinate descent way to find the local minima of the objective function
- Sensitive to initialization
  - Bad initialization may lead to poor convergence speed or bad overall clustering
  - We can try multiple times or find K “spread-out” points
- Need to choose k
- Unsupervised

# K-Means Clustering Results

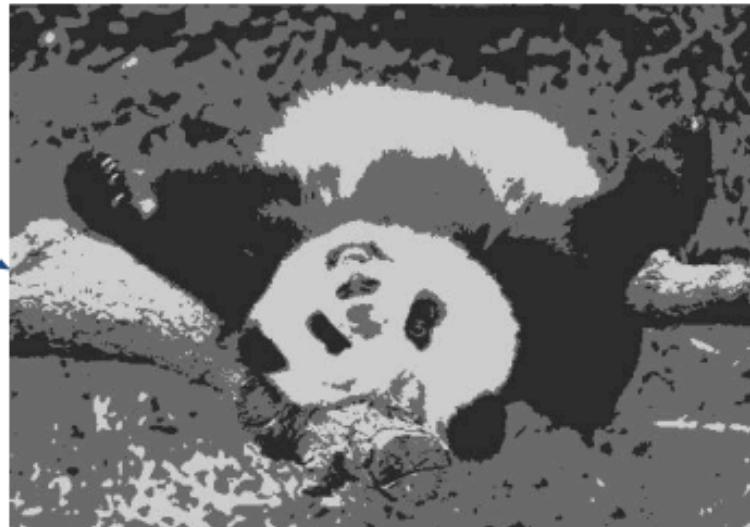
- If we use intensity as the feature for clustering



$k = 2$

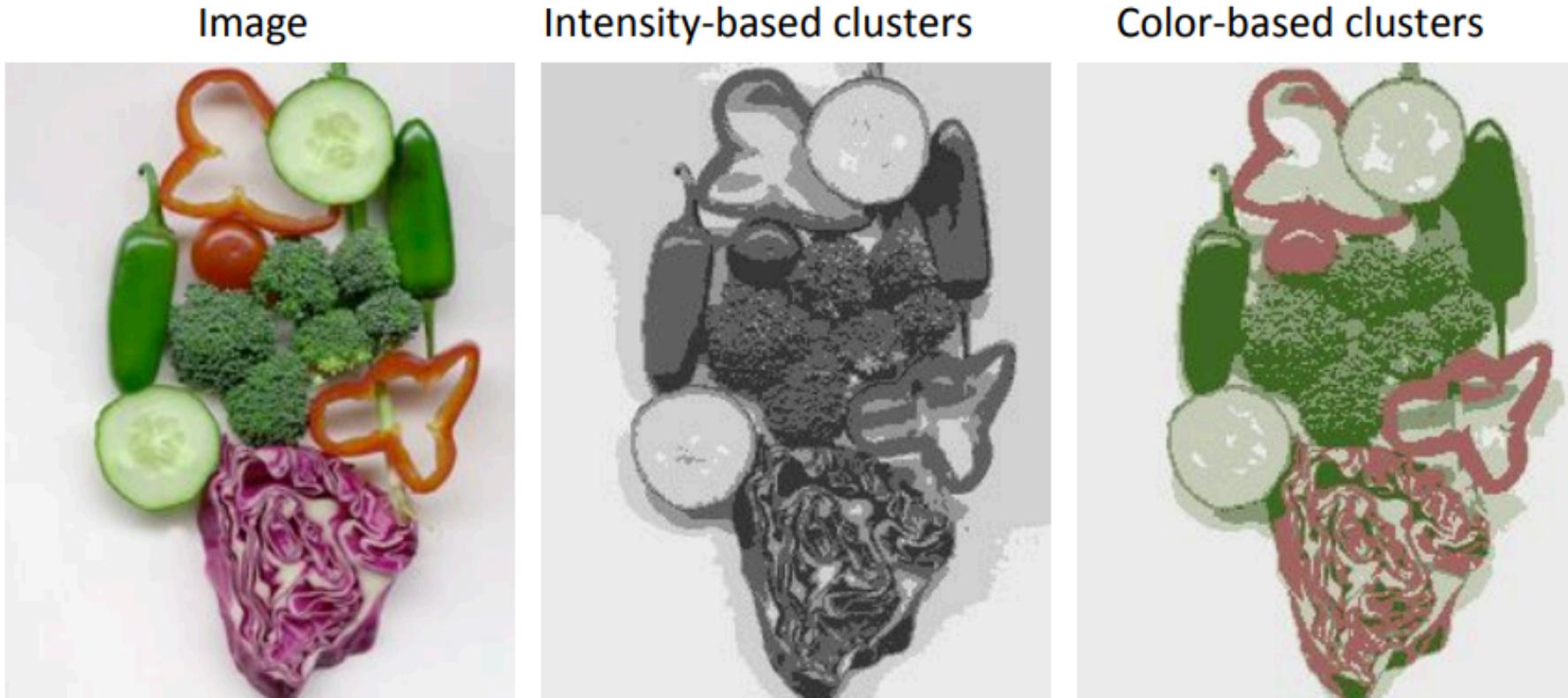


$k = 3$



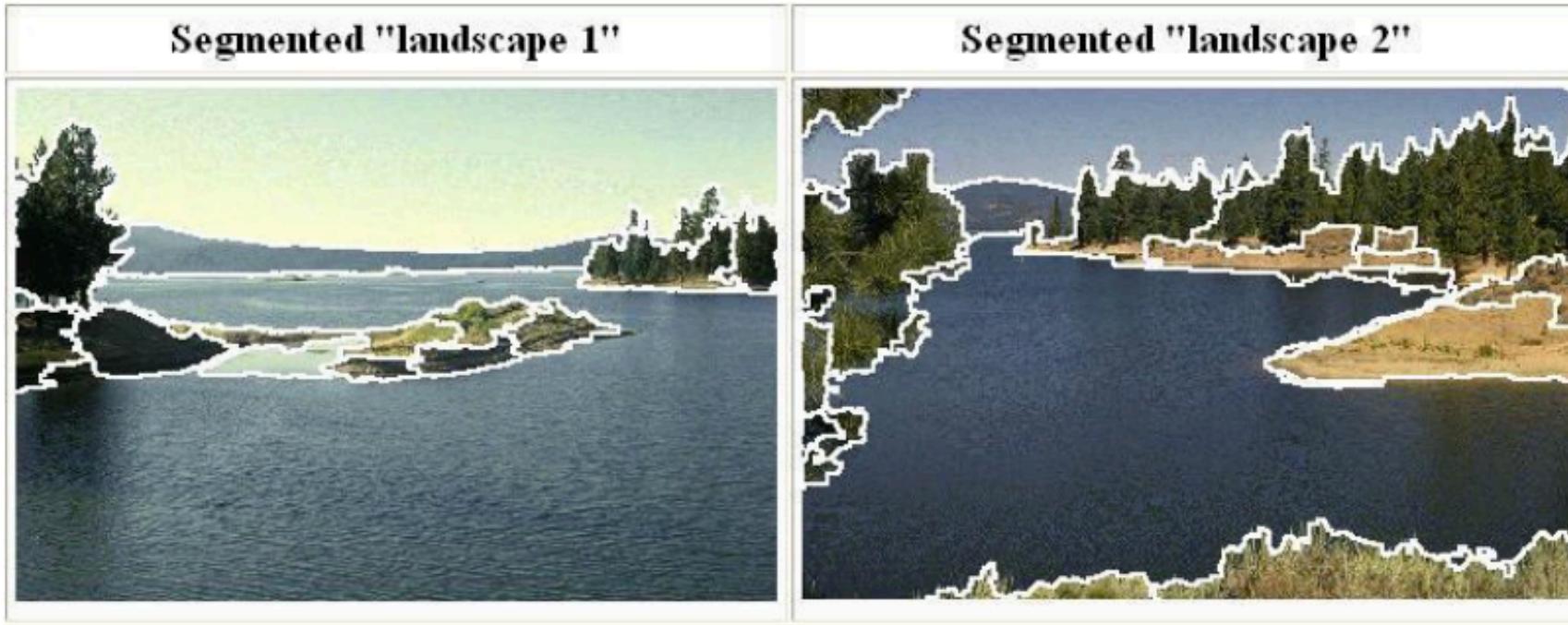
# K-Means Clustering Results

- We can also use color as feature
- However, both of them are essentially vector quantization

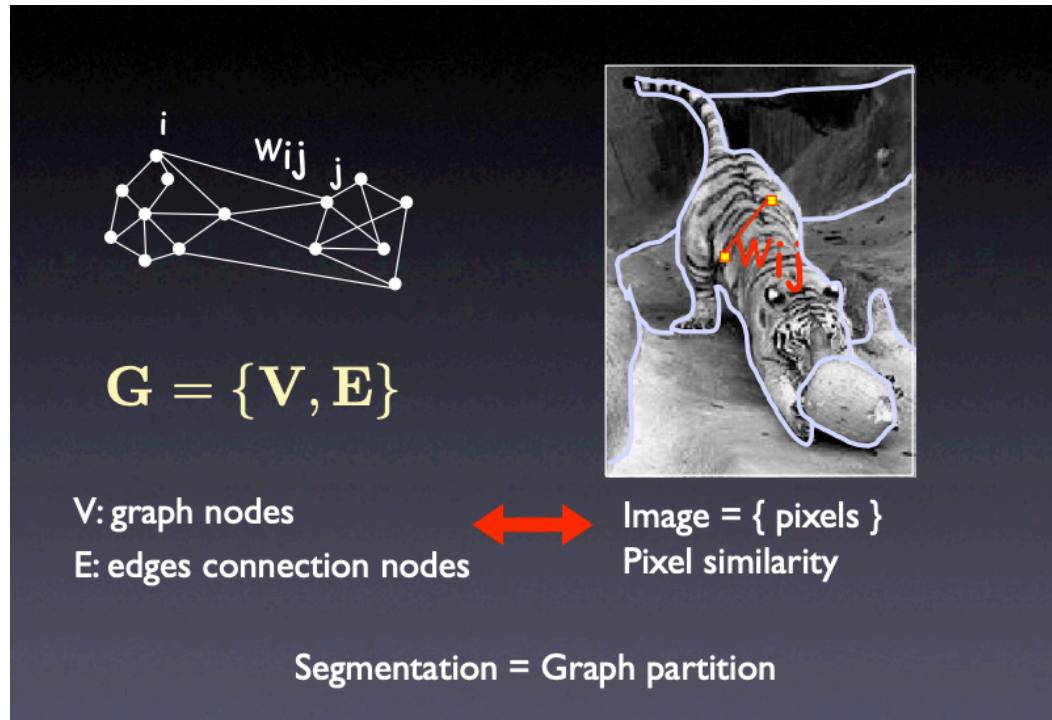


# Mean-Shift Segmentation

- An advanced and versatile technique for clustering-based segmentation



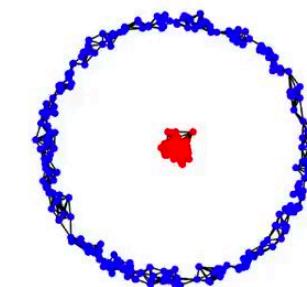
# Graph-based Image Segmentation



## Normalized Cuts and Image Segmentation

Jianbo Shi and Jitendra Malik, *Member, IEEE*

**Abstract**—We propose a novel approach for solving the perceptual grouping problem in vision. Rather than focusing on local features and their consistencies in the image data, our approach aims at extracting the global impression of an image. We treat image segmentation as a graph partitioning problem and propose a novel global criterion, the *normalized cut*, for segmenting the graph. The *normalized cut* criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups. We show that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize this criterion. We have applied this approach to segmenting static images, as well as motion sequences, and found the results to be very encouraging.

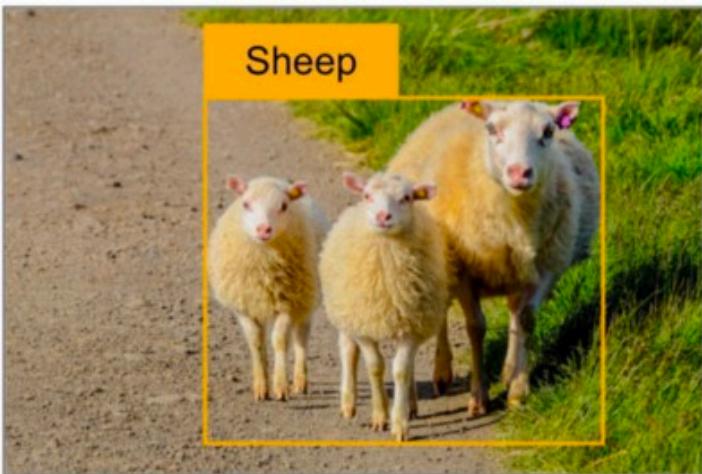


# Summary of Grouping-based Segmentation

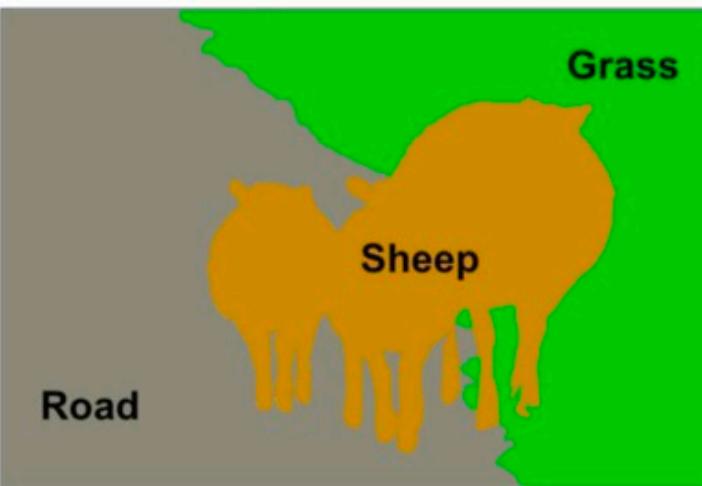
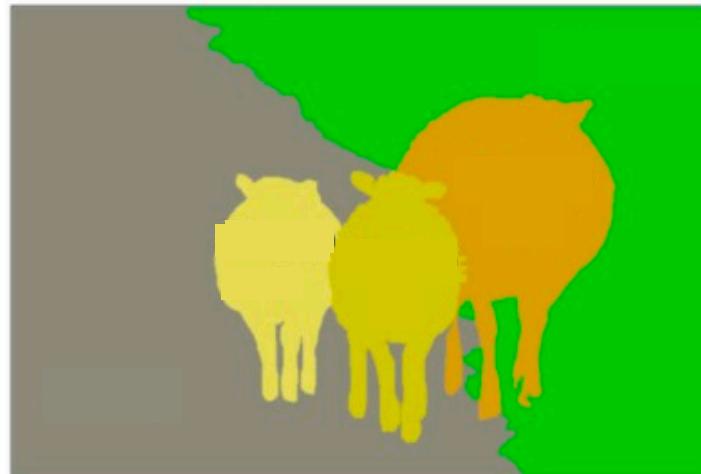
- A mid-level vision task
  - Grouping-based or clustering based: a bottom-up approach
  - Doesn't care about semantics
- Unsupervised
- To improve, we can use more expressive features when performing clustering.

# We Care About Semantics

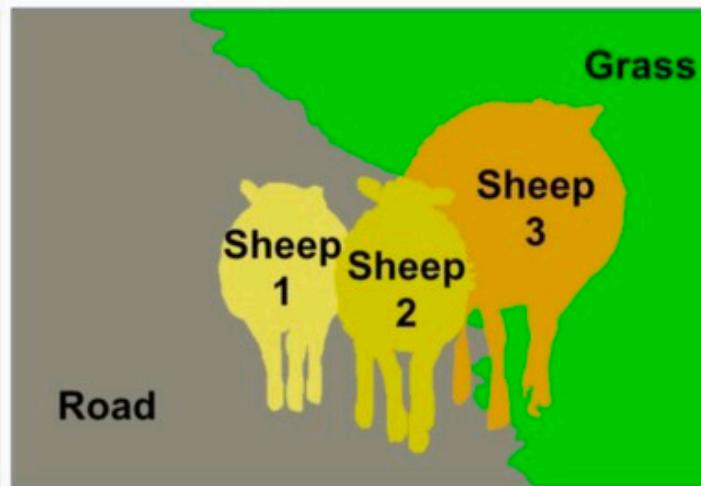
Classification + localization



Instance Segmentation



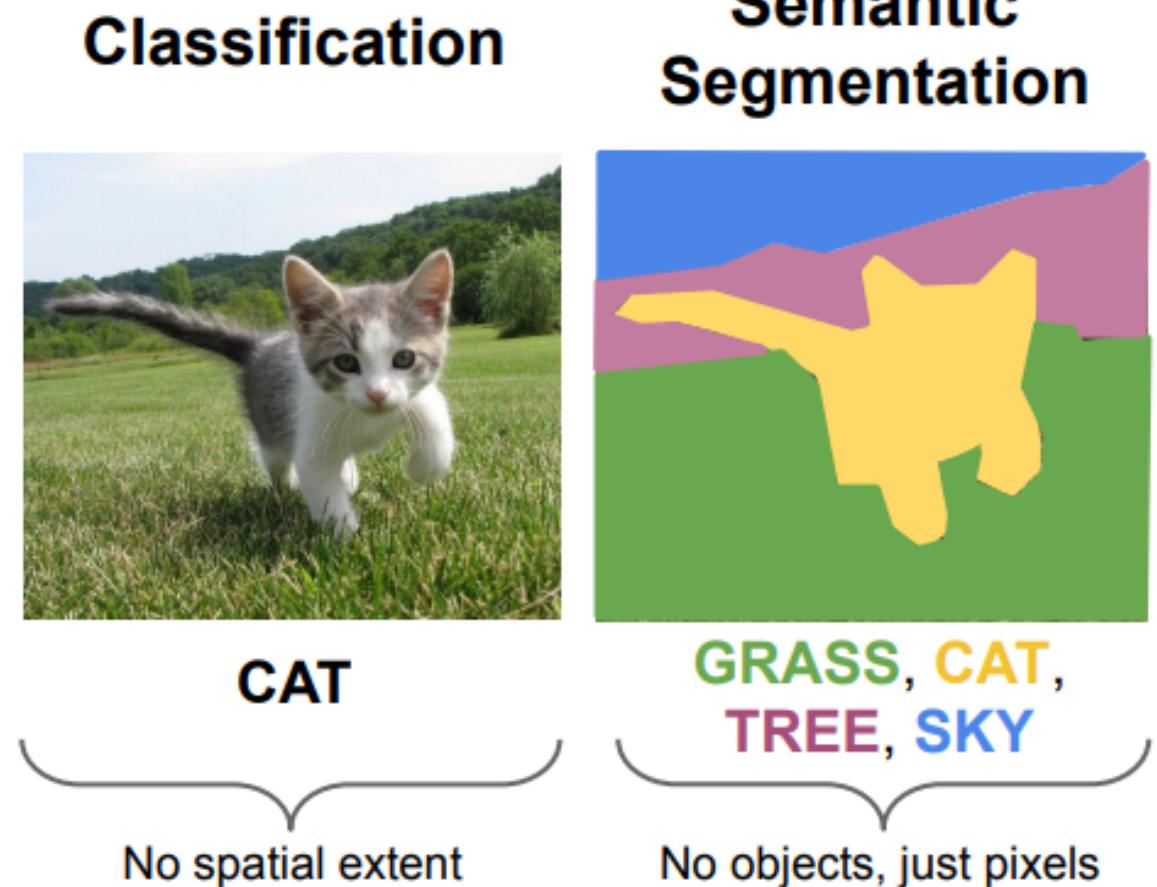
Semantic Segmentation



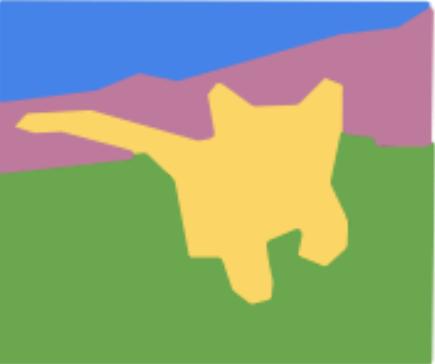
Semantic Instance Segmentation

# Semantic Segmentation

- Semantic segmentation is a dense labeling problem. Or, per-pixel classification problem.
- Sharing similar assumptions to classification: classes are pre-defined.

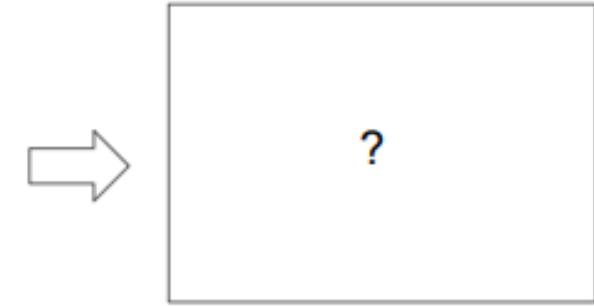


# Semantic Segmentation



GRASS, CAT,  
TREE, SKY, ...

Paired training data: for each training image,  
each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

$$\mathcal{L}_{CE} = \text{mean}(H(P, Q)) = -\text{mean}\left(\sum_{x \in \mathcal{X}} P(x) \log Q(x)\right)$$

# Semantic Segmentation using Sliding Window

Full image



?

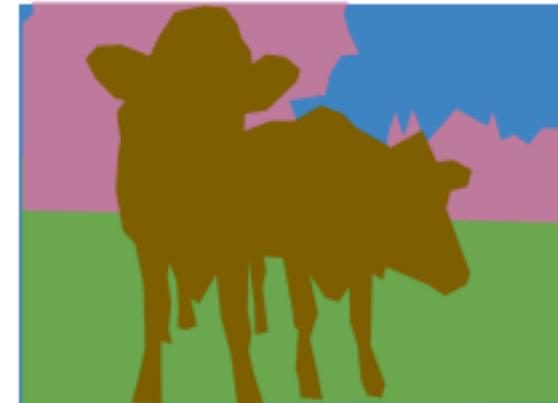
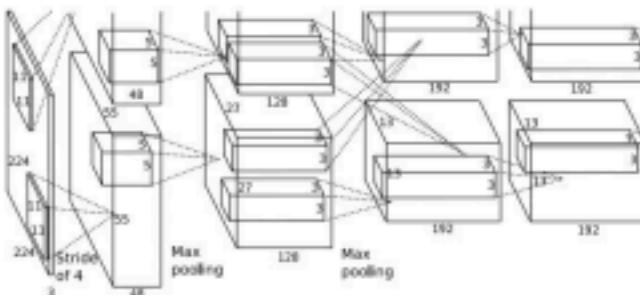


Impossible to classify without context

Q: how do we include context?

# Semantic Segmentation using CNN

Full image

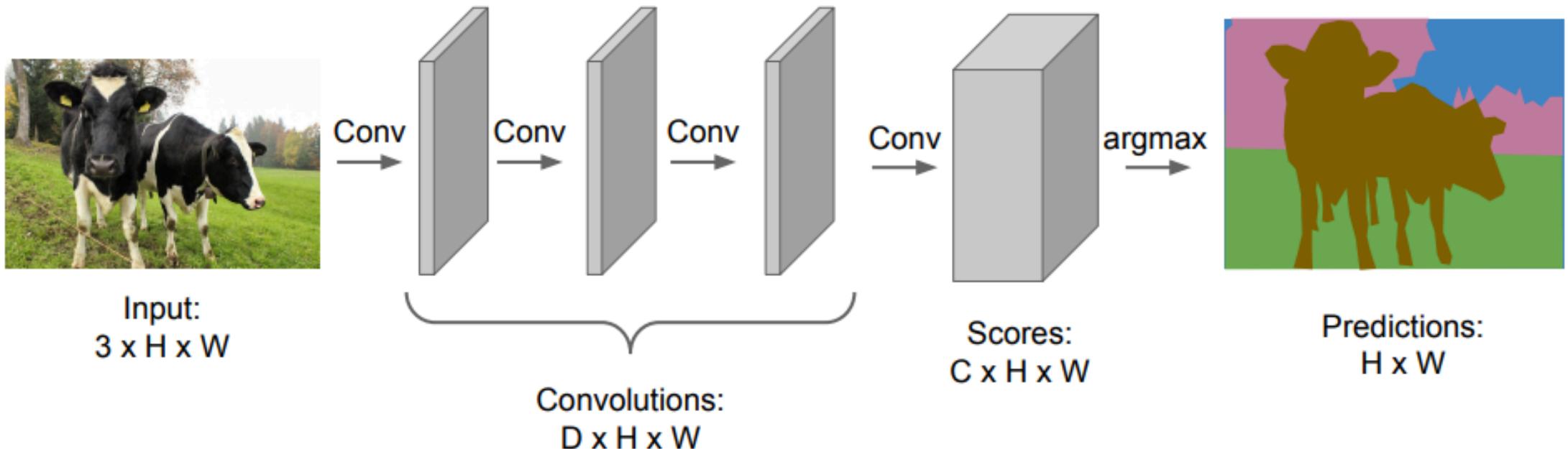


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

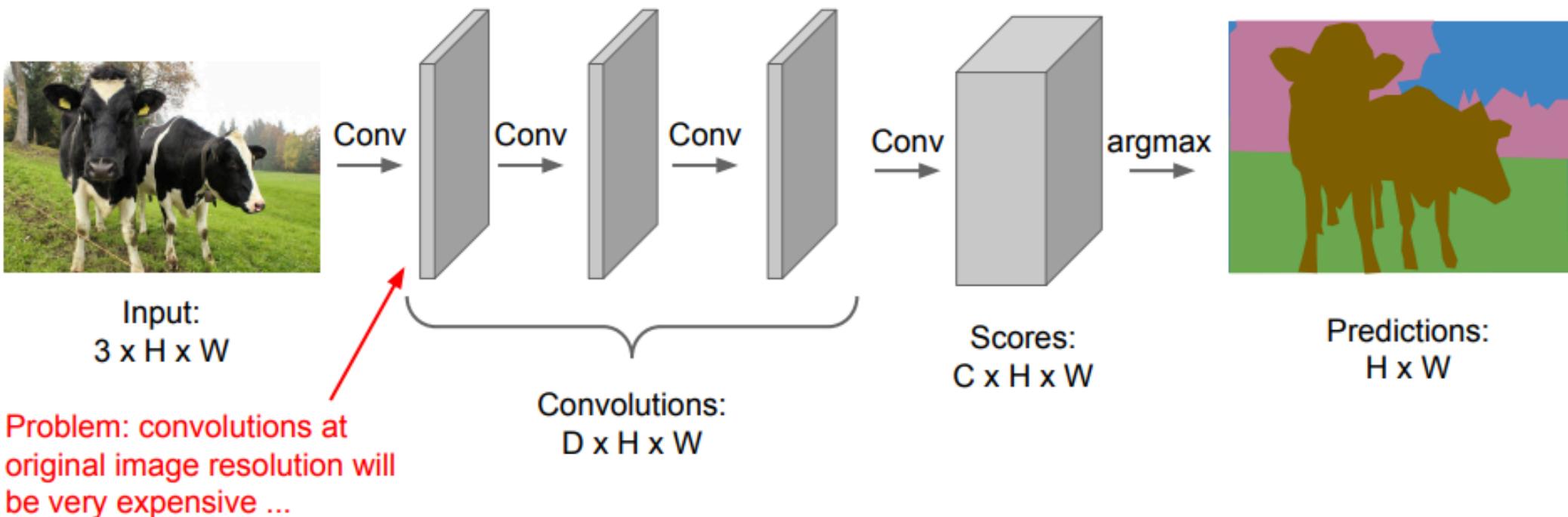
# Semantic Segmentation using Fully Convolution

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



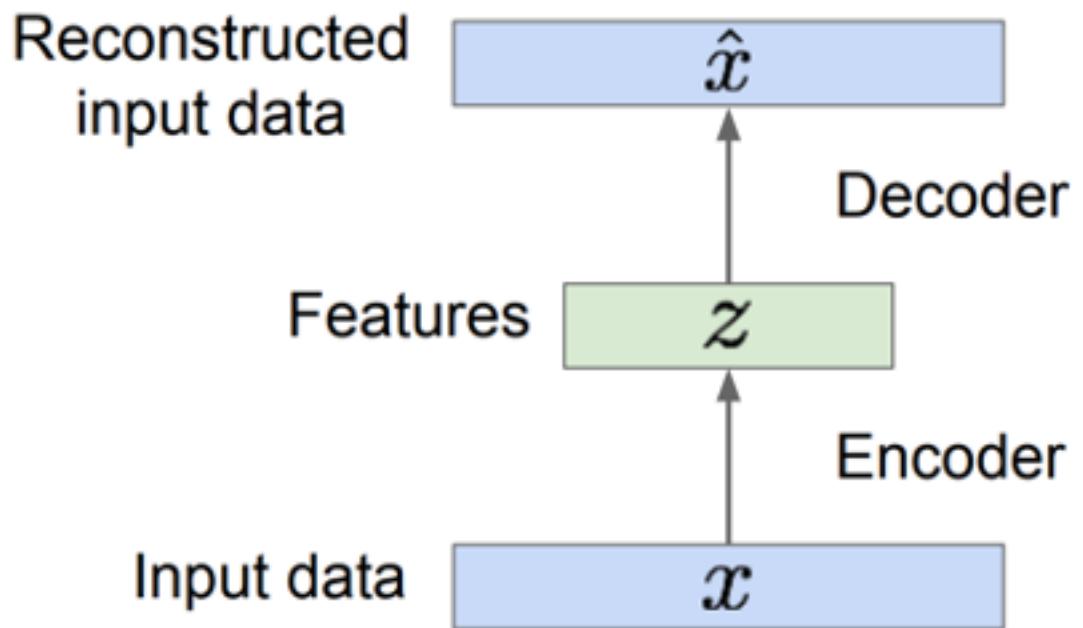
# Semantic Segmentation using Fully Convolution

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



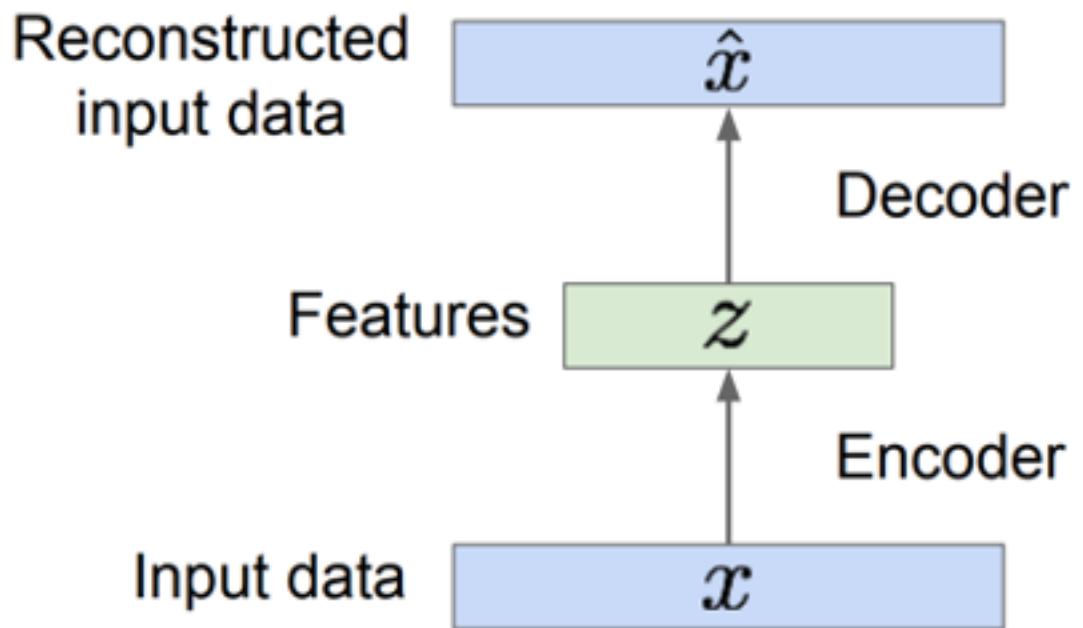
We need to reduce resolutions.

# Auto-Encoder



- AE encodes itself into a latent  $z$
- AE then decodes the latent  $z$  back to itself

# Auto-Encoder



- Understanding AE
  - Information bottleneck: the dimension of  $z$  space is much smaller than that of  $x$
  - Get rid of redundant information via dimension reduction
  - The first step to all advanced segmentation networks

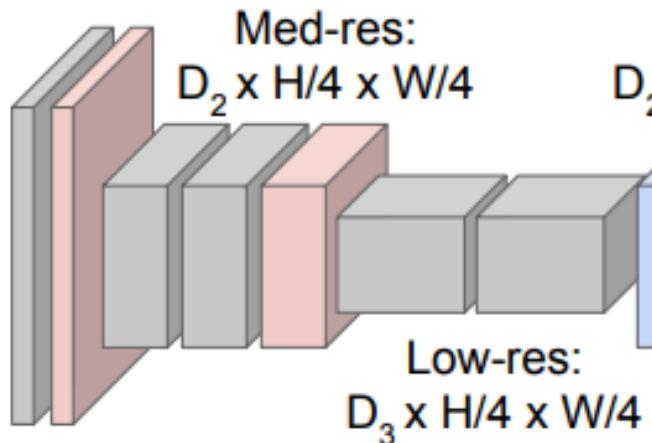
# Semantic Segmentation using Fully Convolution

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

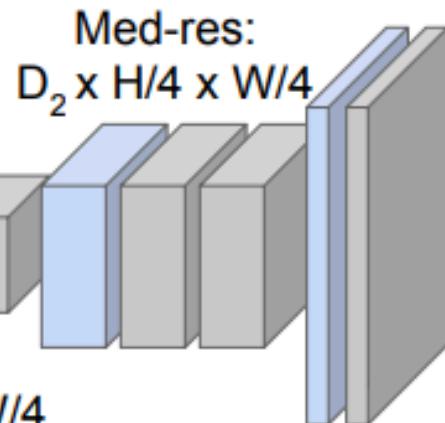


Input:  
 $3 \times H \times W$

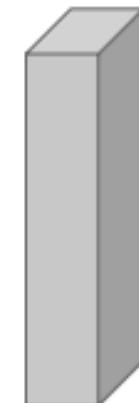
High-res:  
 $D_1 \times H/2 \times W/2$



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



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



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

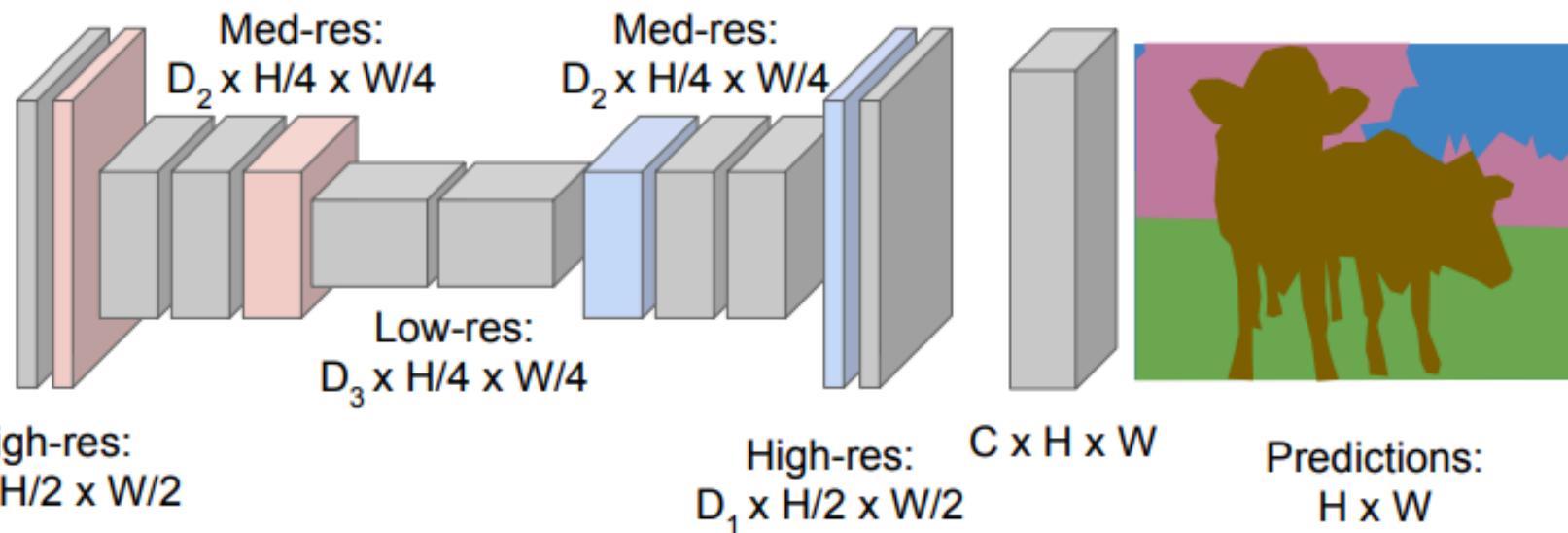
# Semantic Segmentation using Fully Convolution

**Downsampling:**  
Pooling, strided  
convolution



Input:  
 $3 \times H \times W$

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



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

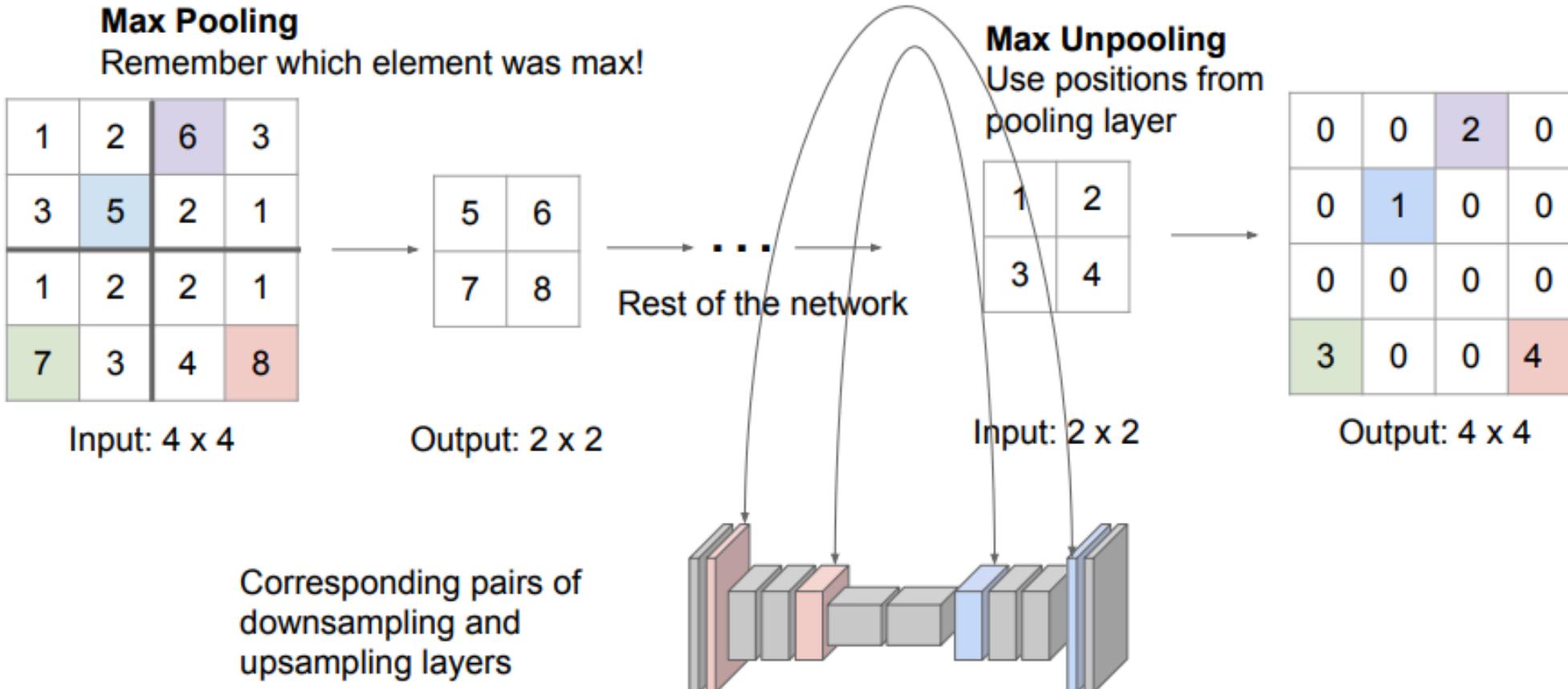


1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2

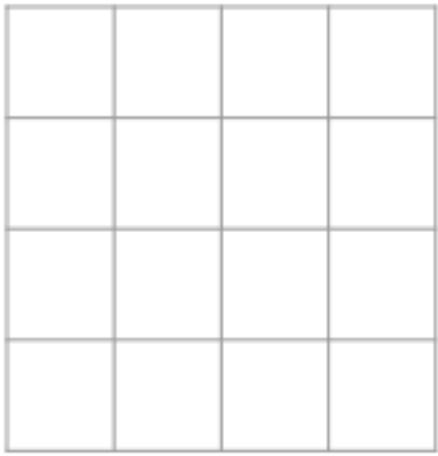
Output: 4 x 4

# In-Network Upsampling: Max Unpooling

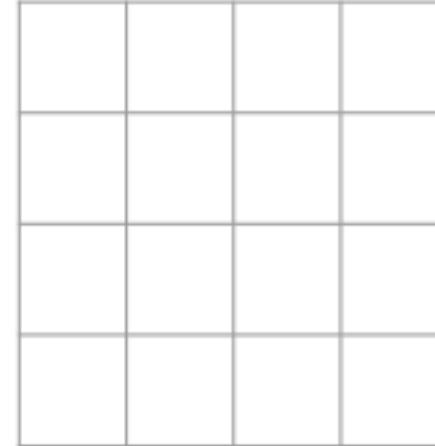


# Learnable Upsampling

**Recall:** Normal  $3 \times 3$  convolution, stride 1 pad 1



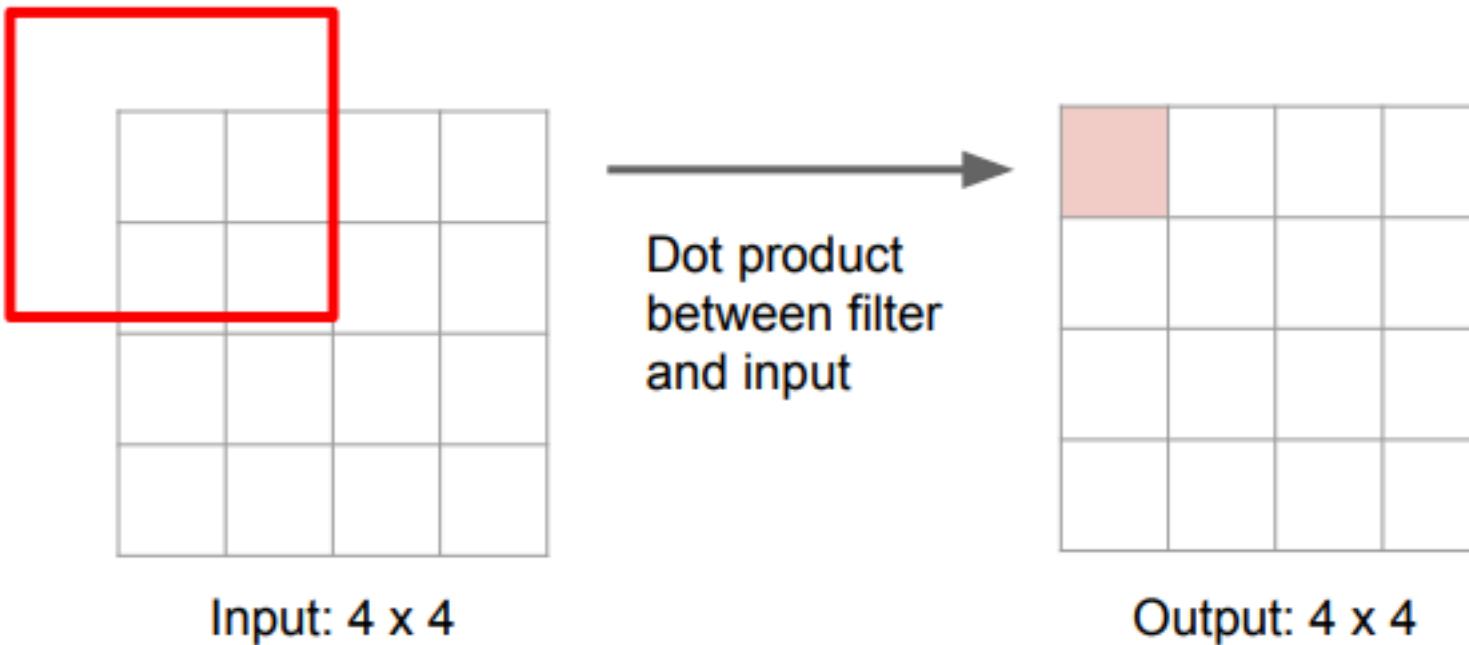
Input:  $4 \times 4$



Output:  $4 \times 4$

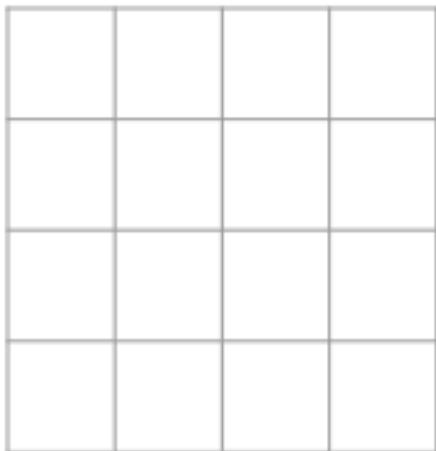
# Learnable Upsampling

**Recall:** Normal  $3 \times 3$  convolution, stride 1 pad 1

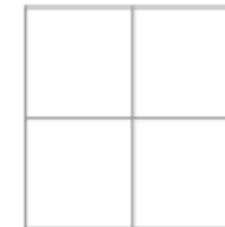


# Learnable Upsampling

**Recall:** Normal  $3 \times 3$  convolution, stride 2 pad 1



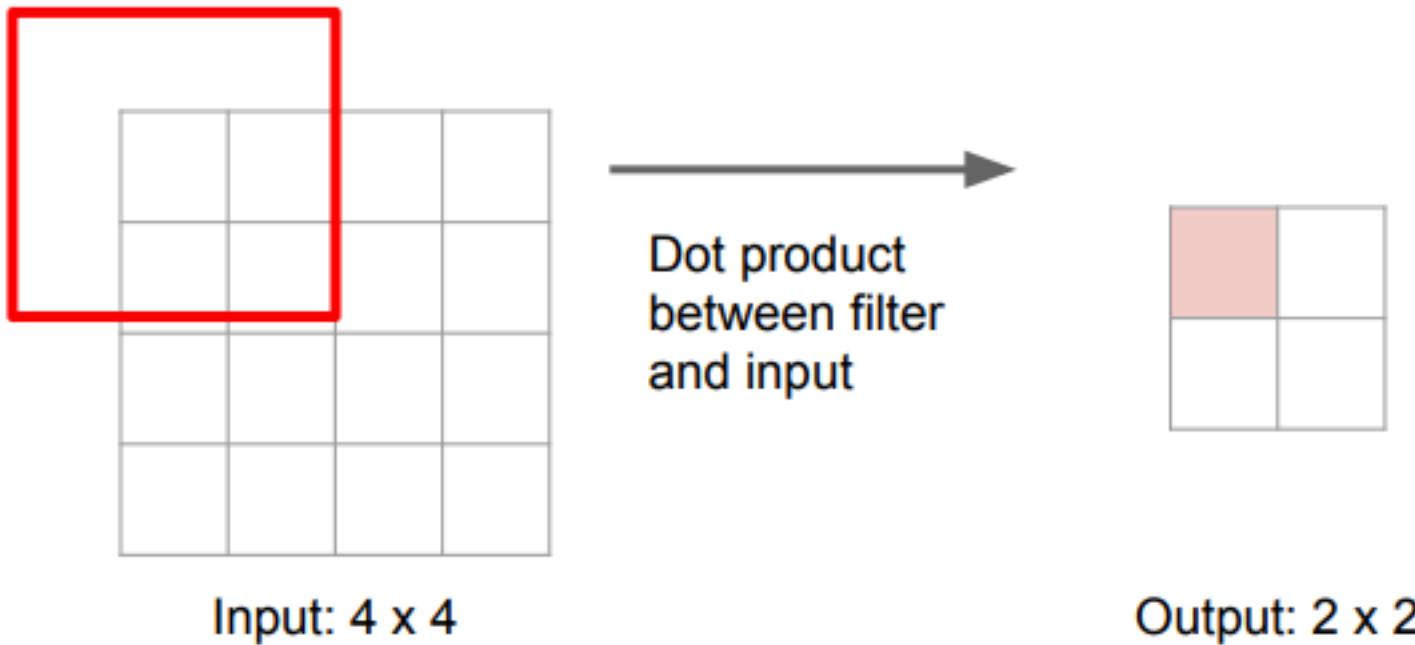
Input:  $4 \times 4$



Output:  $2 \times 2$

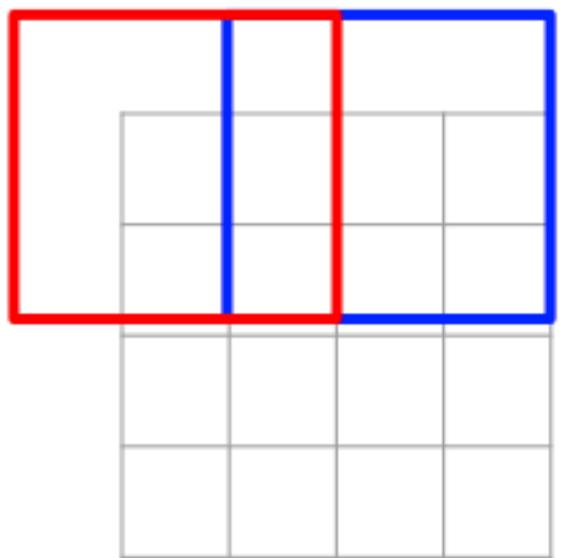
# Learnable Upsampling

**Recall:** Normal  $3 \times 3$  convolution, stride 2 pad 1



# Learnable Upsampling

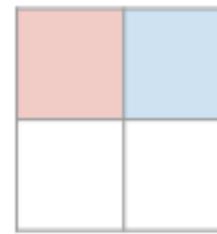
**Recall:** Normal  $3 \times 3$  convolution, stride 2 pad 1



Input:  $4 \times 4$



Dot product  
between filter  
and input



Output:  $2 \times 2$

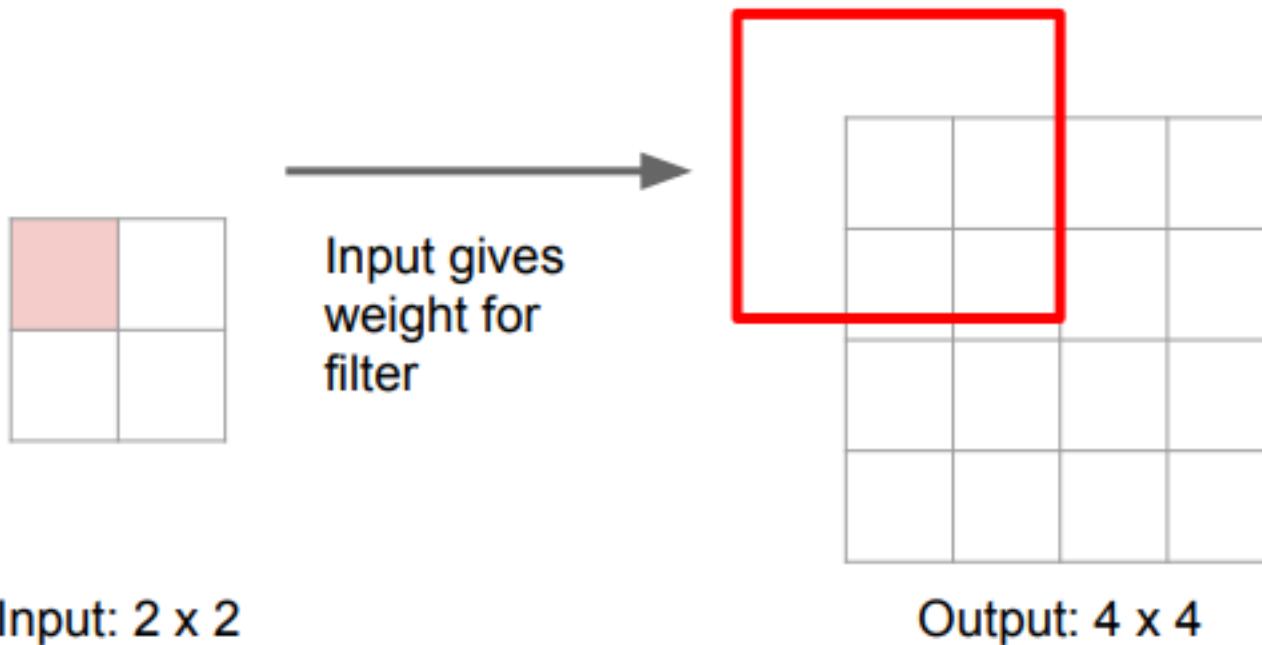
Filter moves 2 pixels in  
the input for every one  
pixel in the output

Stride gives ratio between  
movement in input and  
output

We can interpret strided  
convolution as “learnable  
downsampling”.

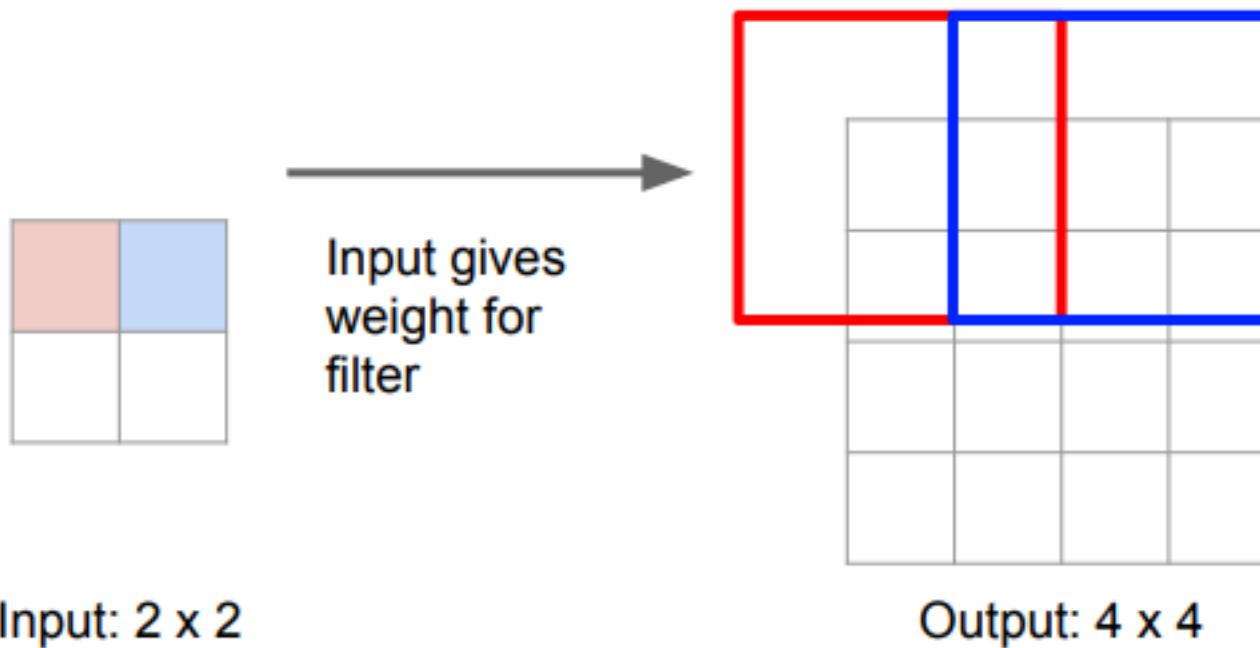
# Learnable Upsampling: Transposed Convolution

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



# Learnable Upsampling: Transposed Convolution

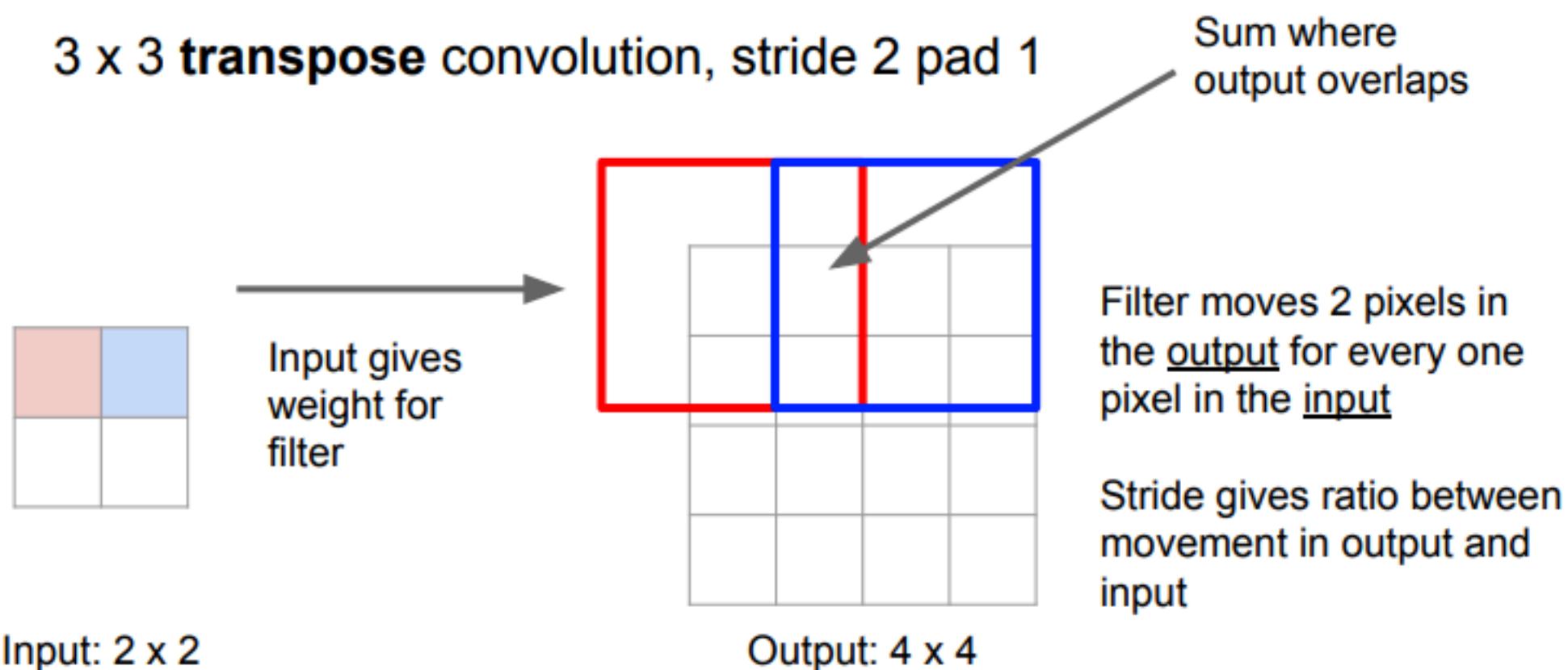
3 x 3 **transpose** convolution, stride 2 pad 1



Filter moves 2 pixels in  
the output for every one  
pixel in the input

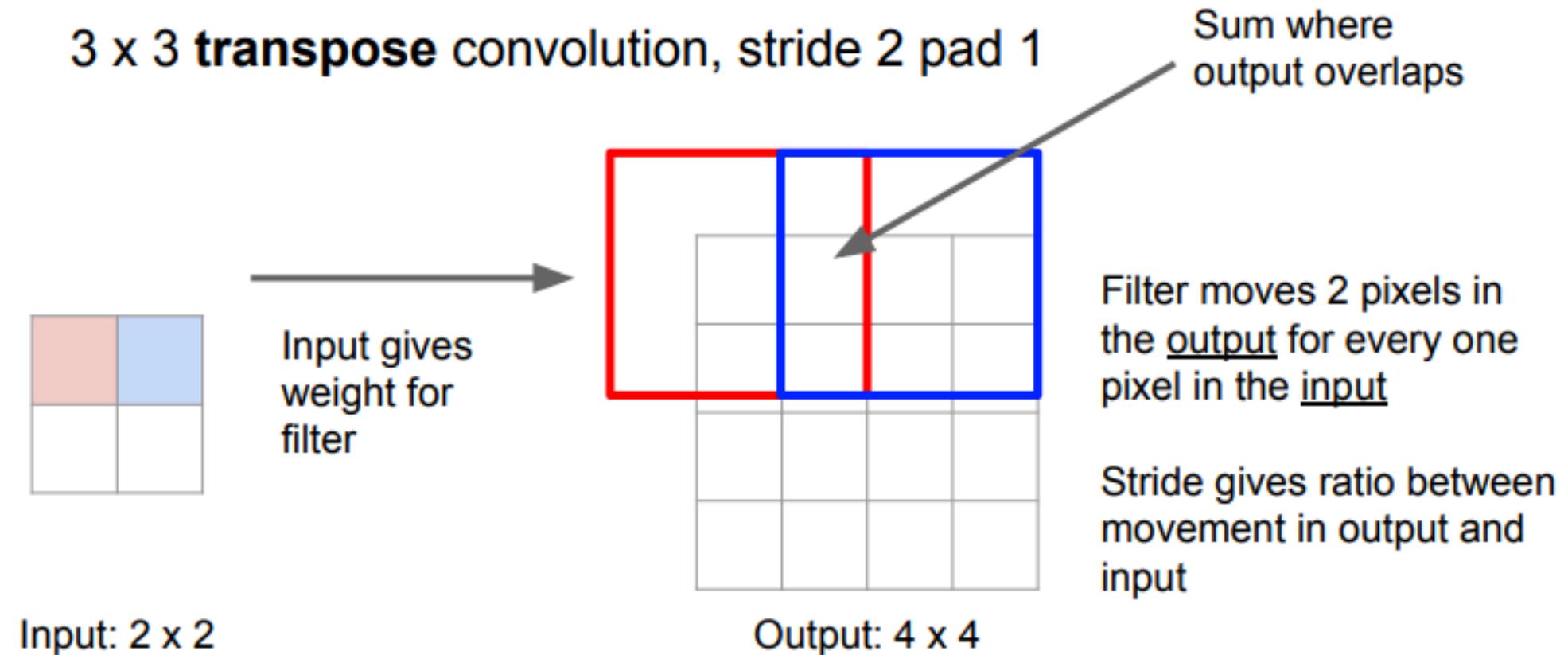
Stride gives ratio between  
movement in output and  
input

# Learnable Upsampling: Transposed Convolution

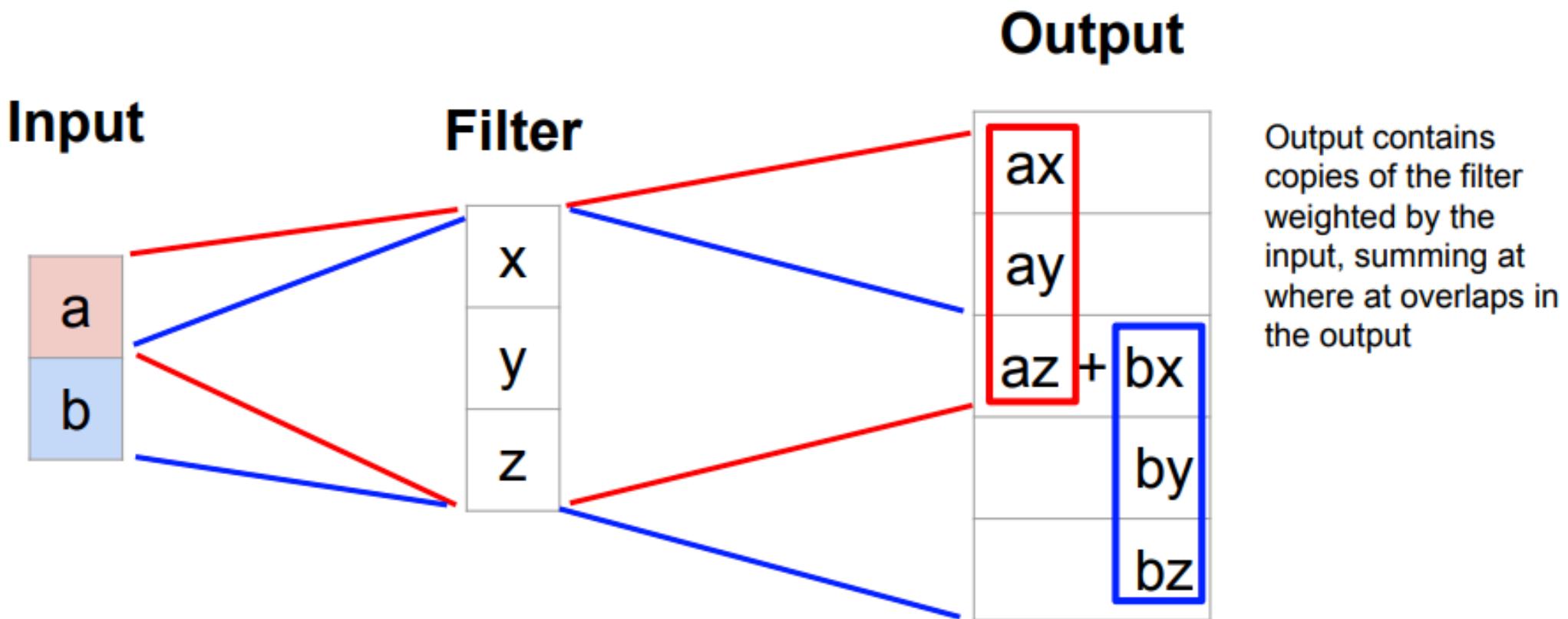


# Learnable Upsampling: Transposed Convolution

Q: Why is it called transpose convolution?



# Learnable Upsampling: Transposed Convolution



# Convolution as Matrix Multiplication

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

# Convolution as Matrix Multiplication

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Example: 1D transpose conv, kernel size=3, stride=2, padding=0

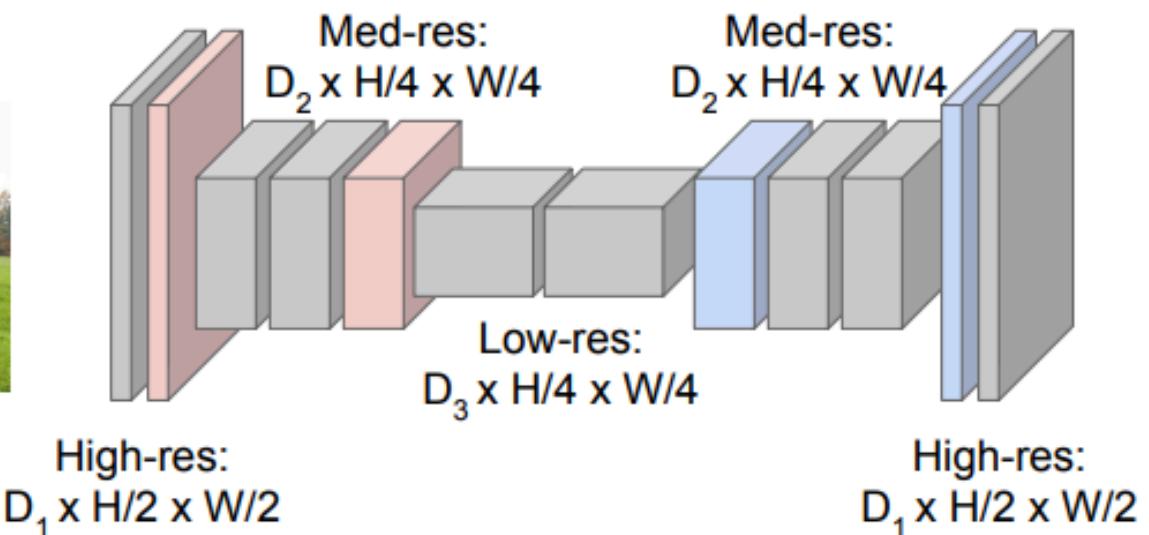
# Semantic Segmentation: Fully Convolutional

**Downsampling:**  
Pooling, strided convolution



Input:  
 $3 \times H \times W$

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



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

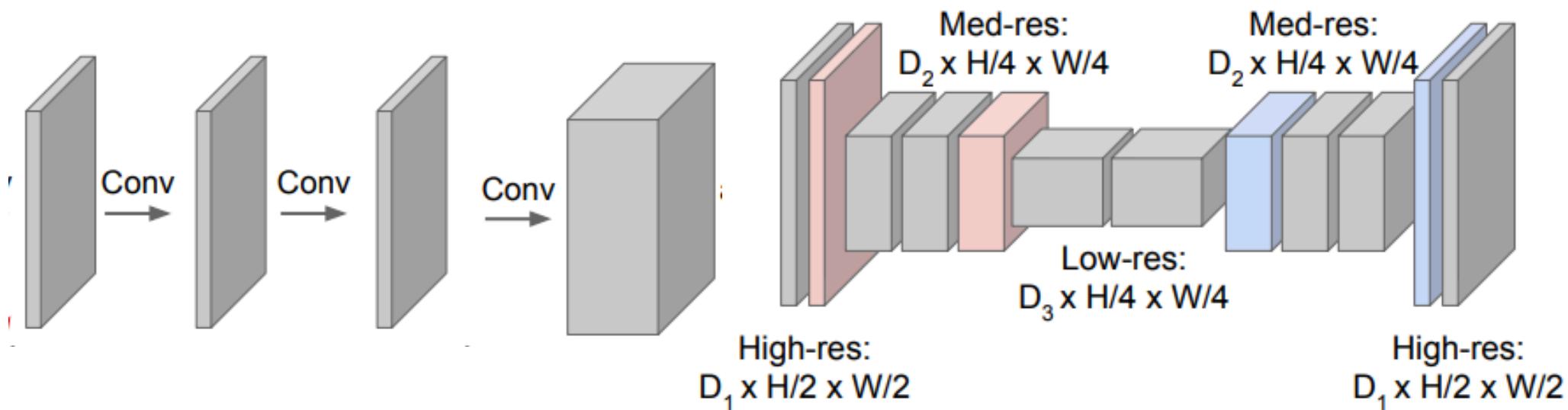
**Upsampling:**  
Unpooling or strided transpose convolution



Predictions:  
 $H \times W$

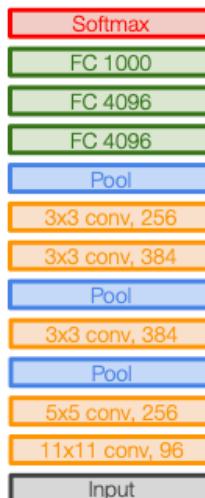
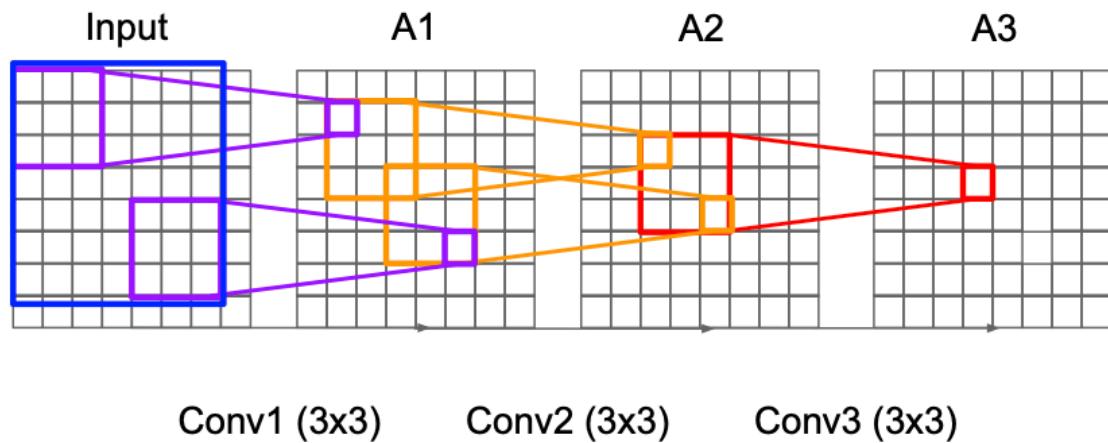
# Advantage of Bottleneck

- Lower memory cost

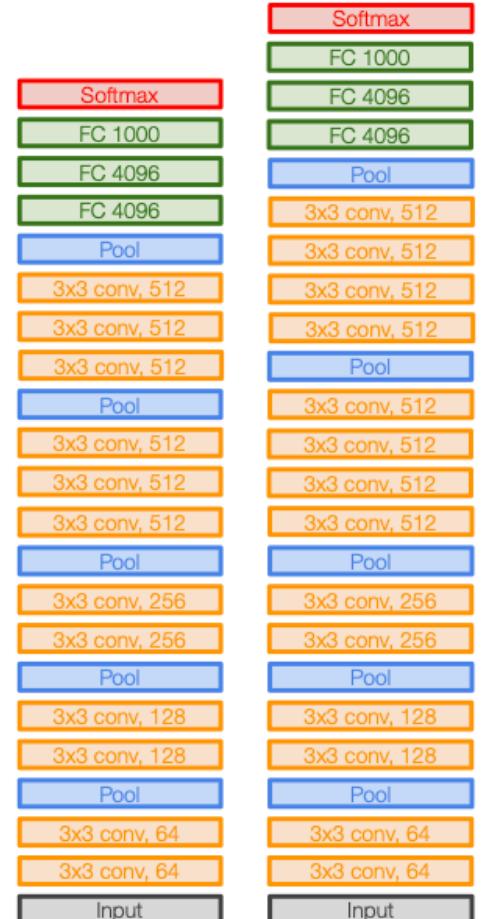


# Recap of Receptive Field

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



AlexNet

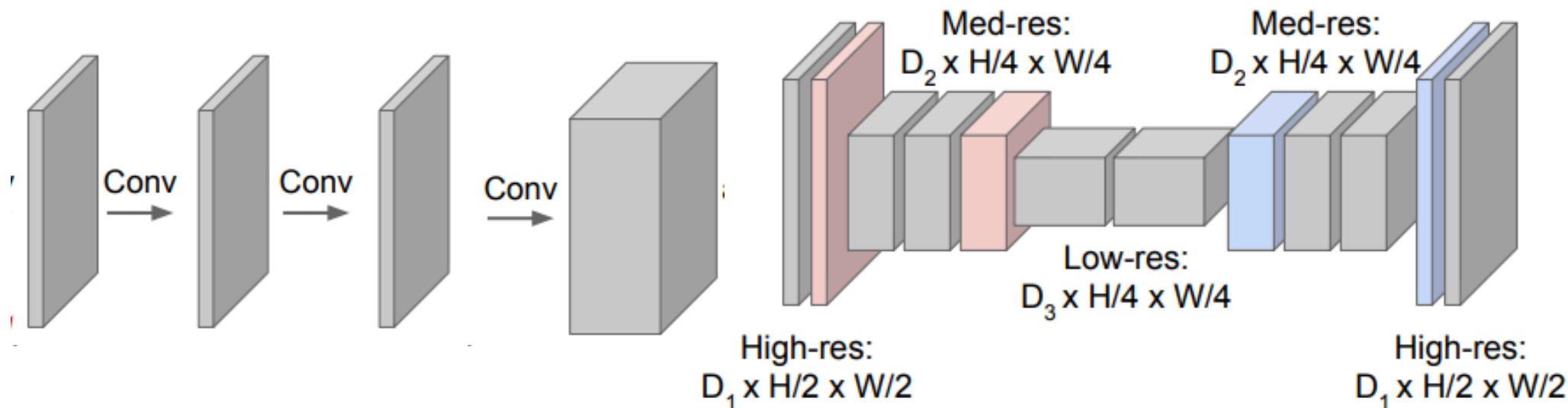


VGG16

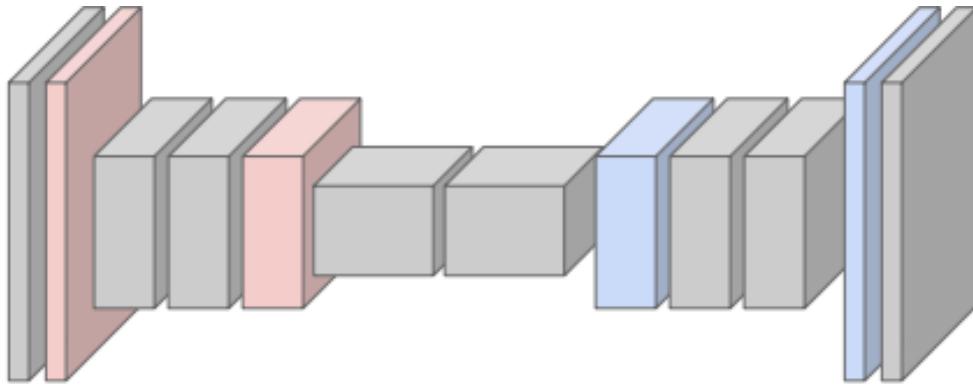
VGG19

# Advantage of Bottleneck

- Lower memory cost
- Larger receptive field and thus better global context
  - Convolution on a smaller feature map correspond to conv with a big kernel size at the original resolution

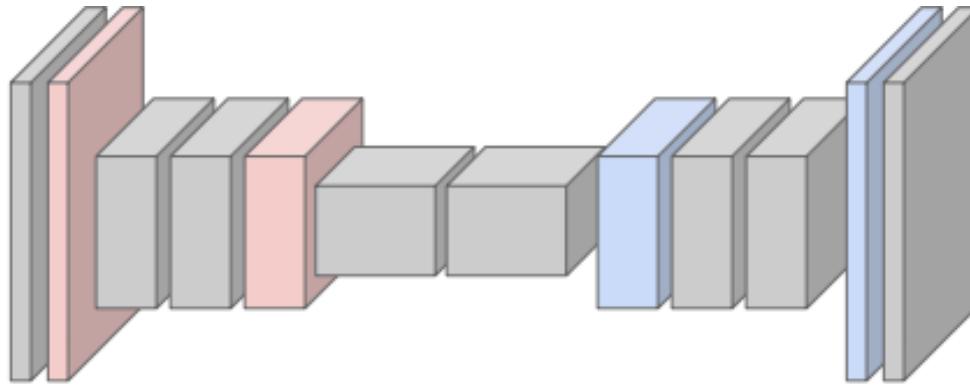


# Improving FCN



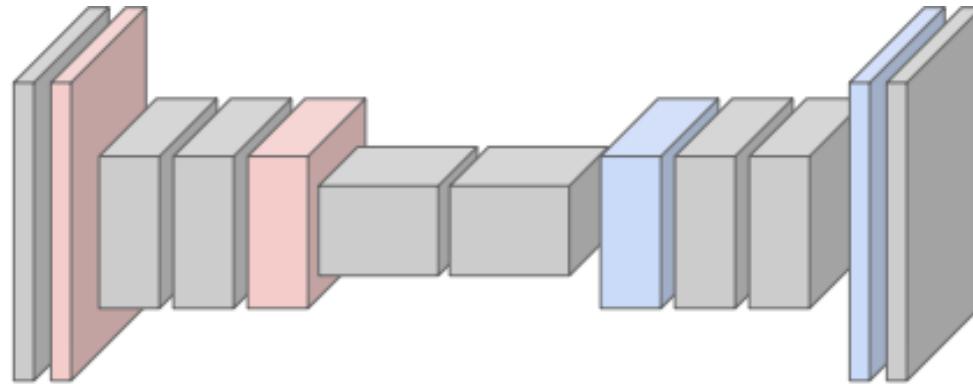
What needs to be stored in the bottleneck?

# Improving FCN



What needs to be stored in the bottleneck?  
• Global context

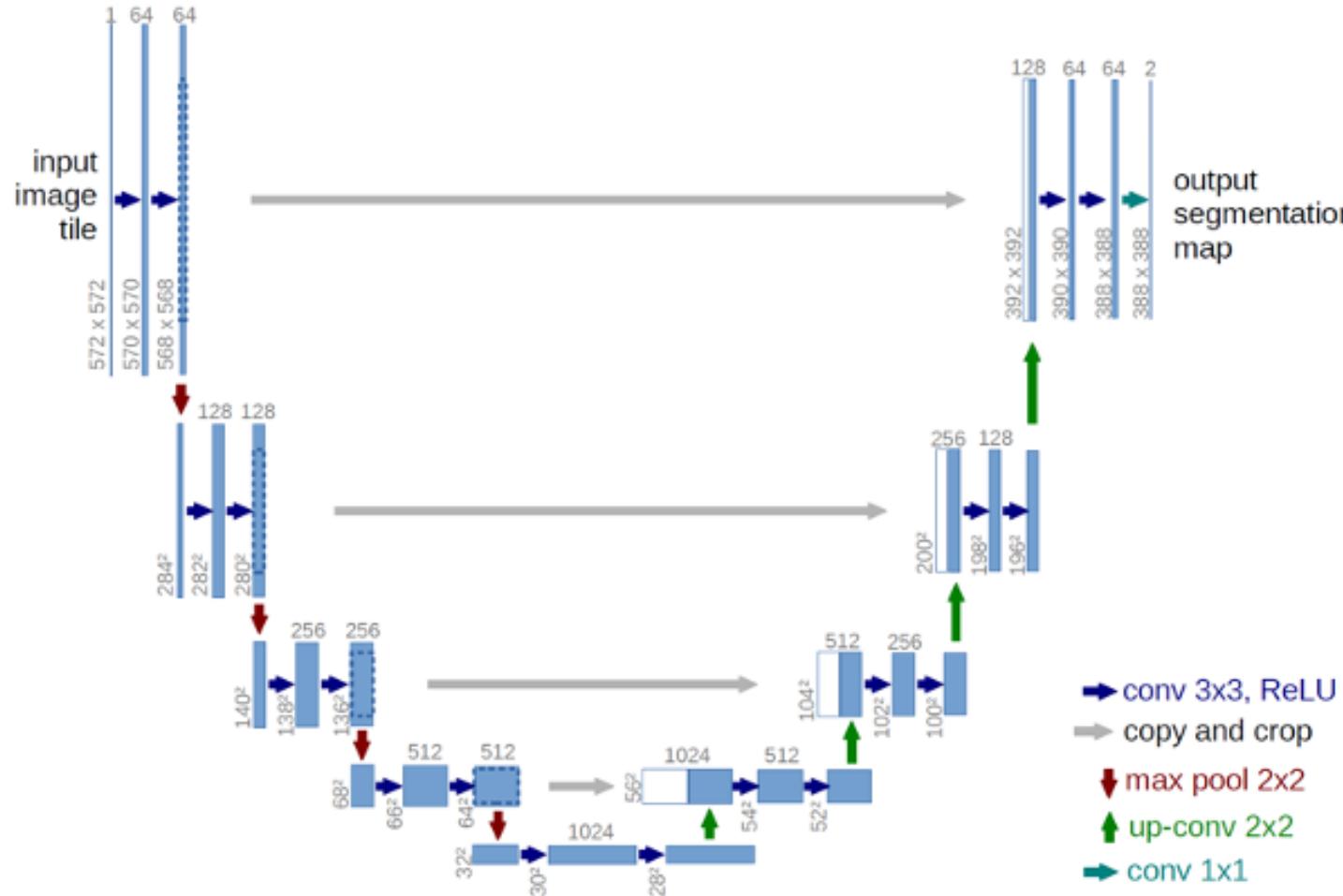
# Improving FCN



What needs to be stored in the bottleneck?

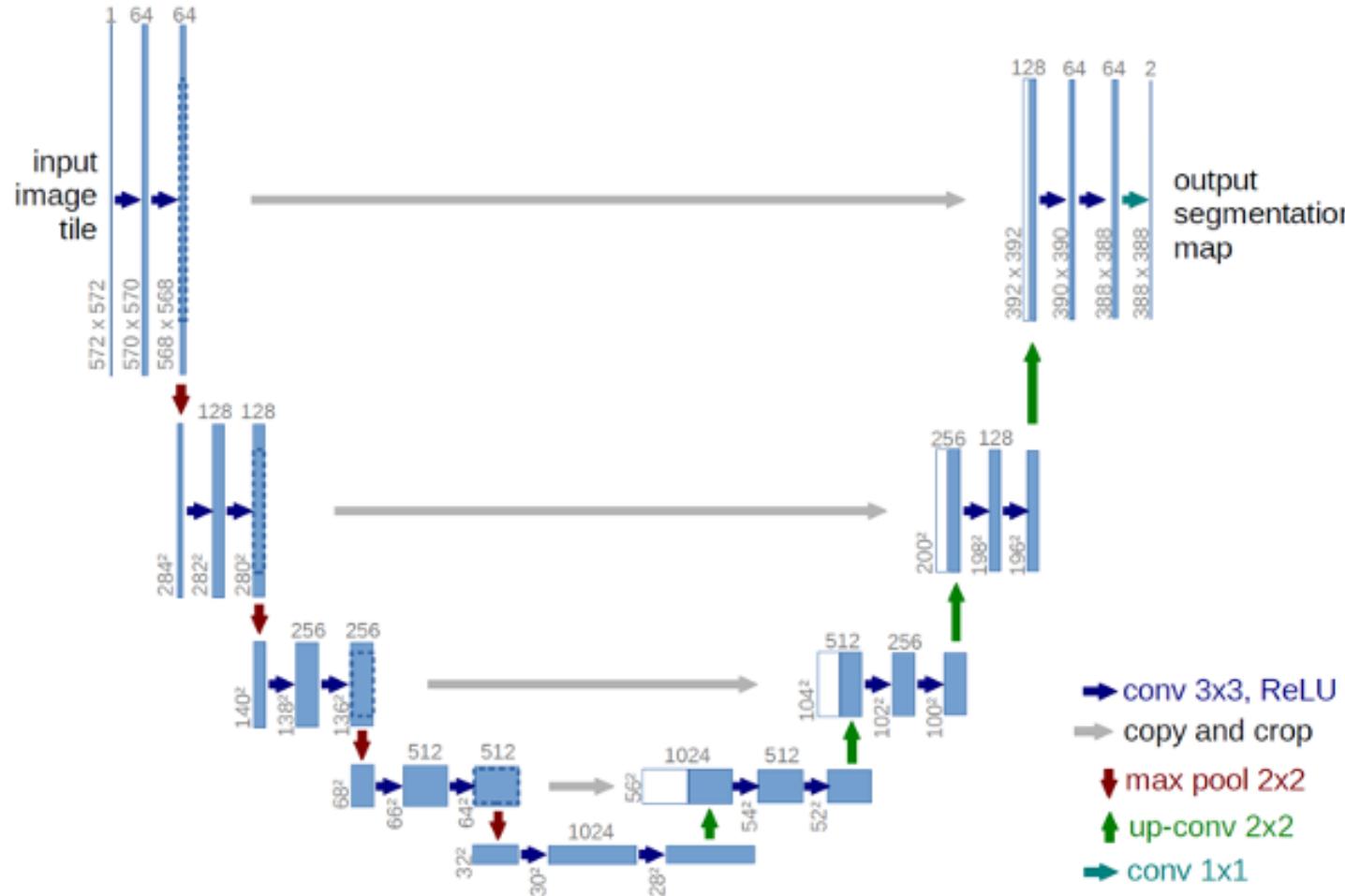
- Global context
- Per-pixel spatial information, especially around the boundary

# UNet Structure



- Skip link between the feature maps from the encoder and the decoder with the same resolution.
- Now what needs to store in the bottleneck?

# UNet Structure

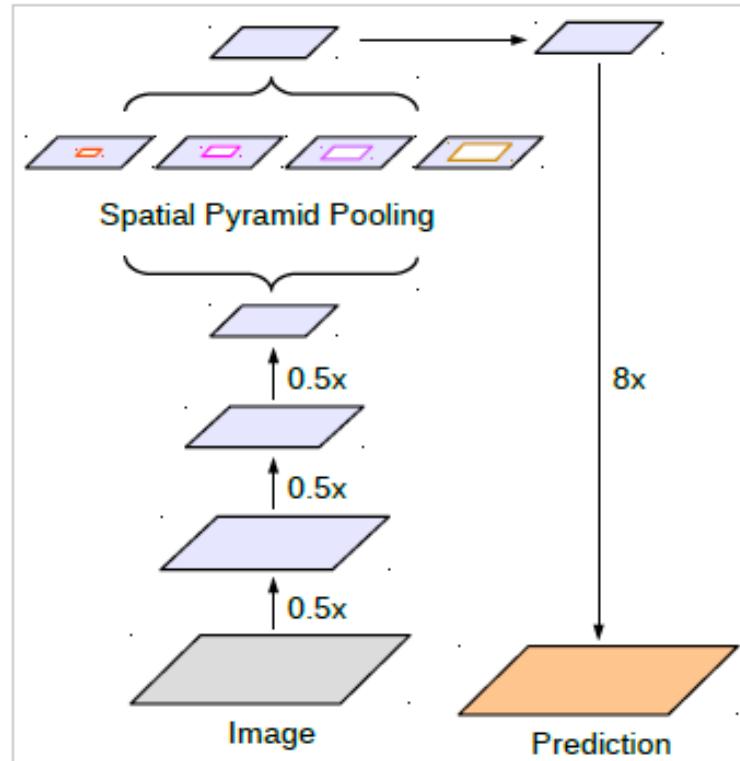


- The skip link makes shortcut from the inputs to the outputs
- Bottleneck: no need to memorize the whole image but only provides global context

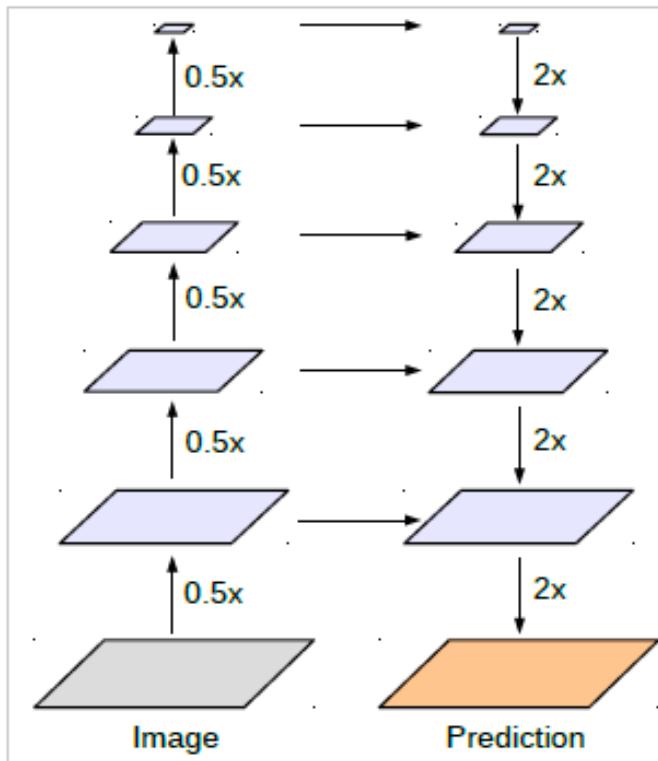
# Summary of Semantic Segmentation

- A top-down approach
- Bottleneck structure:
  - Large receptive field and provides global context
  - Get rid of redundant information
  - Lower the computation cost
- Skip link:
  - Assist final segmentation
  - Avoid memorization

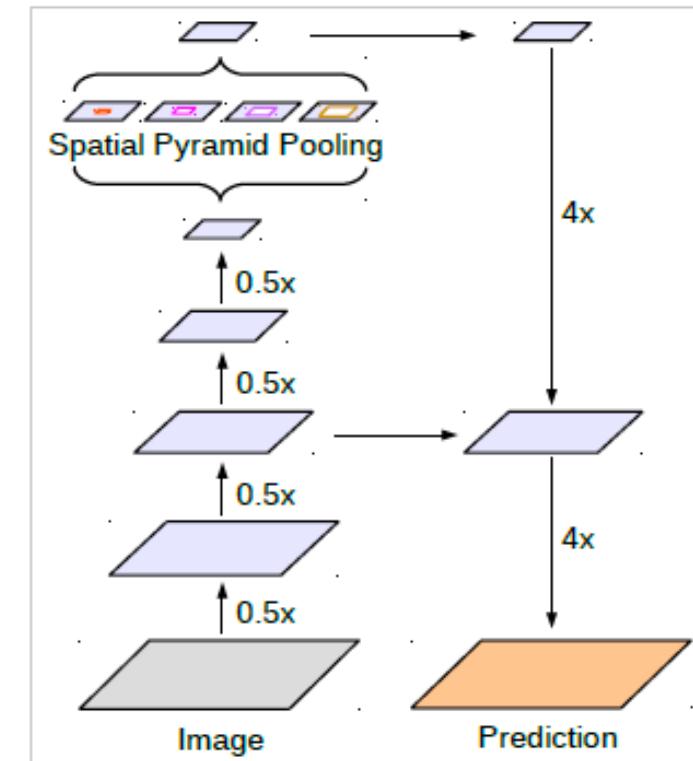
# DeepLab V3



(a) Spatial Pyramid Pooling

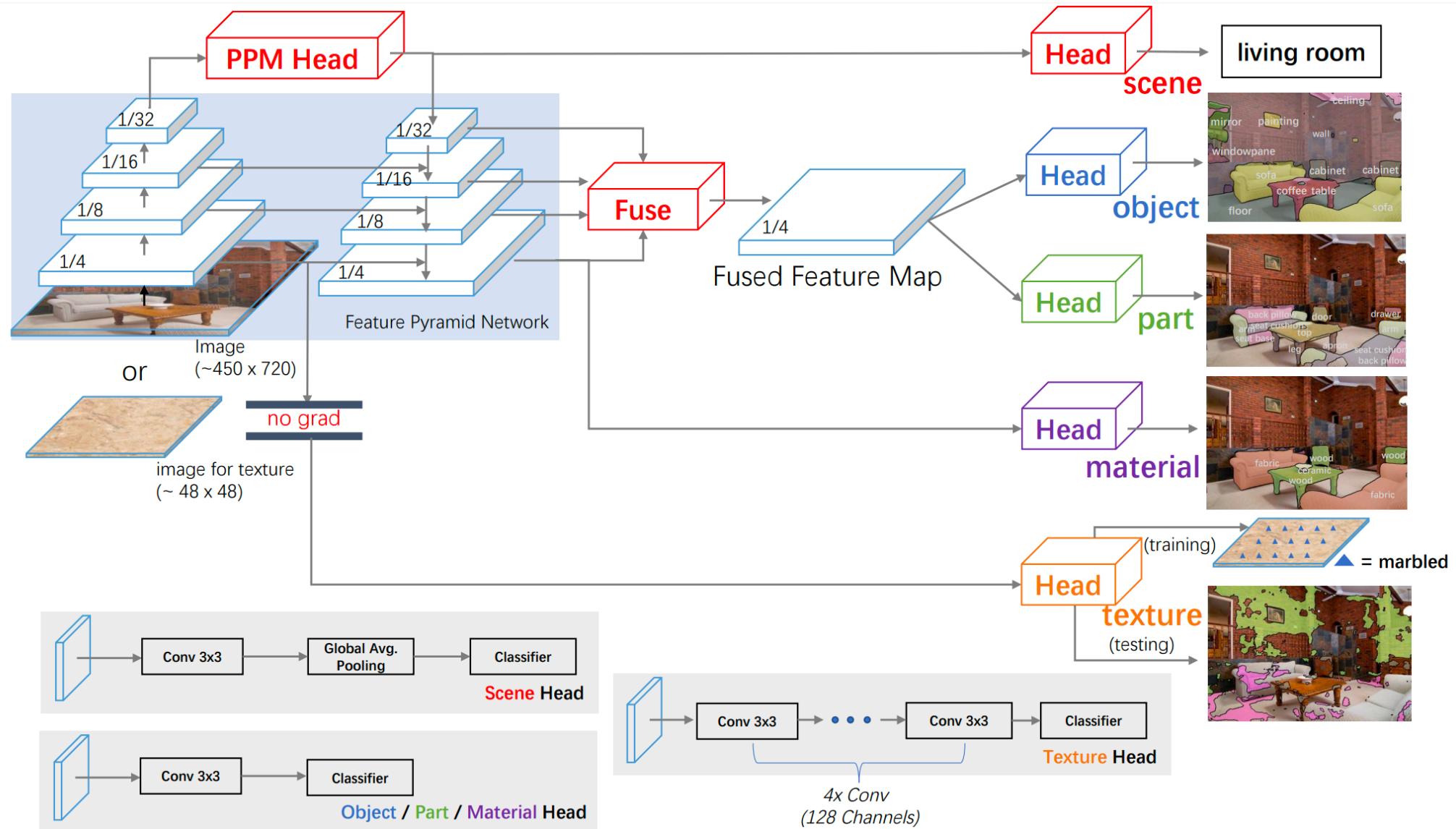


(b) Encoder-Decoder



(c) Encoder-Decoder with Atrous Conv

# General Dense Prediction: UperNet



# Evaluation Metrics: Pixel Accuracy

- Pixel accuracy: simply report the percent of pixels in the image which were correctly classified.

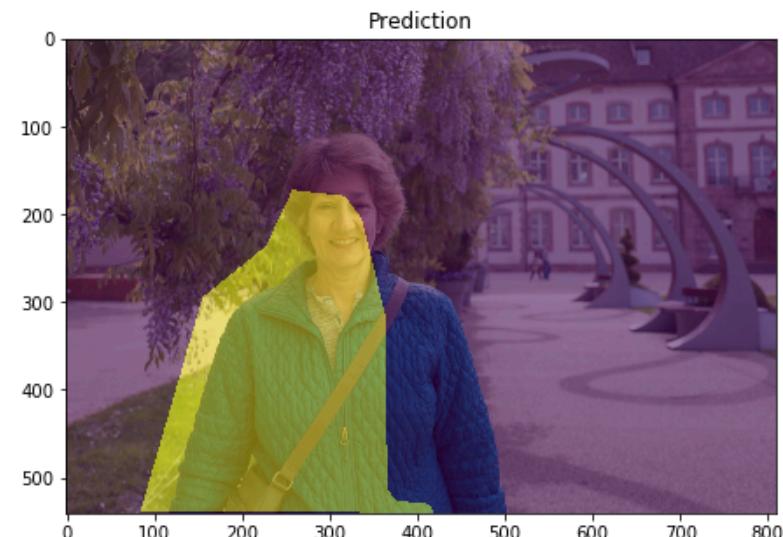
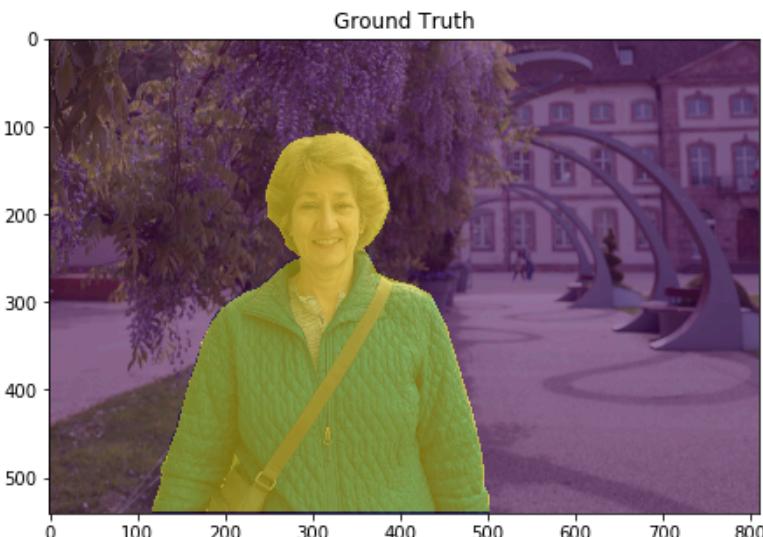
$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- However, may be misleading when the class representation is small within the image, as the measure will be biased in mainly reporting how well you identify negative case (ie. where the class is not present).

# Evaluation Metrics: Intersection over Union

- Intersection over Union

$$IoU = \frac{\text{target} \cap \text{prediction}}{\text{target} \cup \text{prediction}}$$



# Alternative Loss: Soft IoU Loss

$$IoU = \frac{I(X)}{U(X)} .$$

where,  $I(X)$  and  $U(X)$  can be approximated as follows:

$$I(X) = \sum_{v \in V} X_v * Y_v .$$

$$U(X) = \sum_{v \in V} (X_v + Y_v - X_v * Y_v) .$$

Therefore, the IoU loss  $L_{IoU}$  can be defined as follows:

$$L_{IoU} = 1 - IoU = 1 - \frac{I(X)}{U(X)} .$$

# Evaluation Metrics: mIoU

- For each class, we can compute the metrics above by finding the intersection between the ground truth and predicted one-hot encoded masks for each class.
- Metrics can be examined class-by-class, or by taking the average over all the classes, to get a mean IoU.



# Introduction to Computer Vision

Next Week: Lecture 7,  
3D Vision I