Introduction to Computer Vision

Lecture 7 - Deep Learning IV

Prof. He Wang

Classification

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

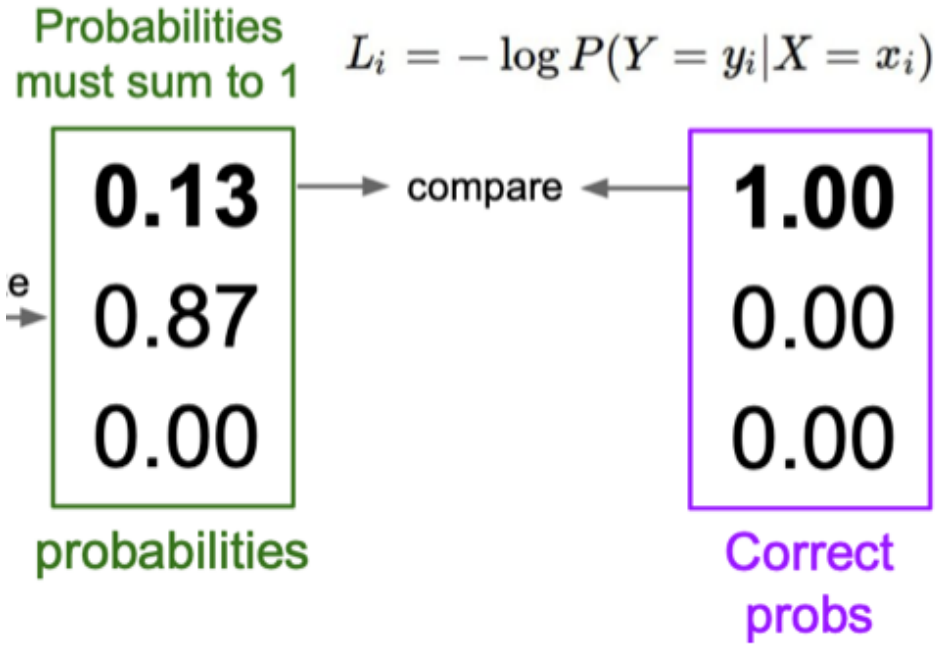
Probabilities must be >= 0

Probabilities must sum to 1

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities	normalize	probabilities	compare	Correct probs
cat	3.2		24.5		0.13		1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

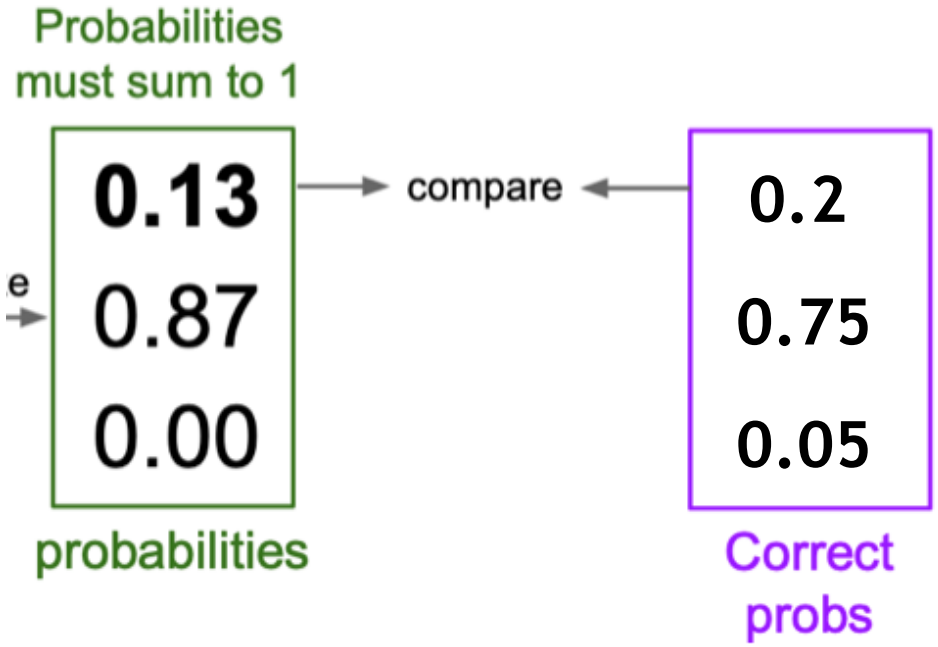
Negative Log-likelihood Loss

- 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 $D_{KL}(P||Q)$
  - a measure of how one probability distribution *Q* is different from a second, reference probability distribution *P*.
  - For discrete probability distributions:

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

Properties of KL Divergence

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

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

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

From KL Divergence to Cross-Entropy

- $$D_{KL}(P||Q) = \underbrace{-\sum_{x \in \mathcal{X}} P(x) \log Q(x)}_{H(P,Q)} - \underbrace{(-\sum_{x \in \mathcal{X}} P(x) \log P(x))}_{H(P)}$$

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

From KL Divergence to Cross-Entropy

- $$D_{KL}(P||Q) = \underbrace{-\sum_{x\in\mathcal{X}} P(x)\log Q(x)}_{H(P,Q)} - \underbrace{(-\sum_{x\in\mathcal{X}} P(x)\log P(x))}_{H(P)}$$

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

Properties of Cross Entropy Loss

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

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

Summary of SoftMax Classifier

3.2

5.1

-1.7

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:

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

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?

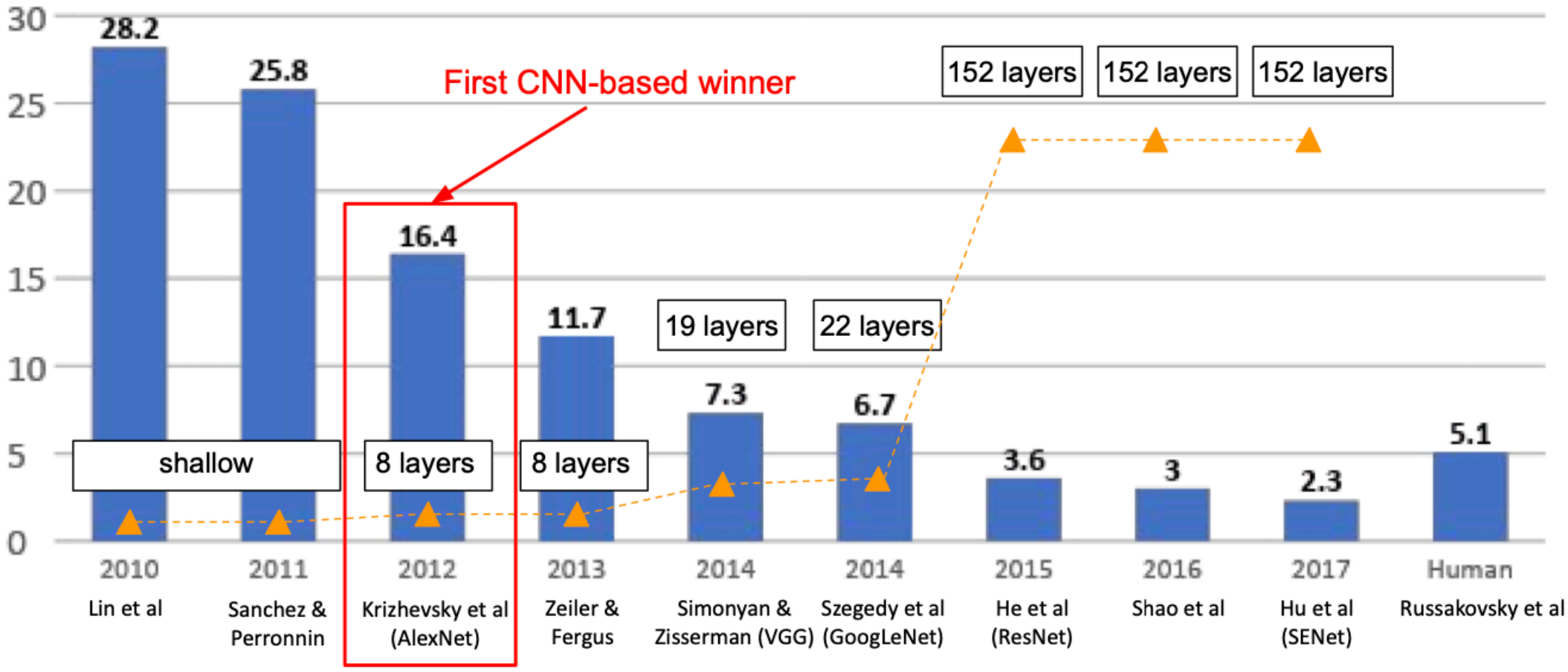
- ImageNet Challenges contains 1000 classes
  - 90 different dog breeds

- 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

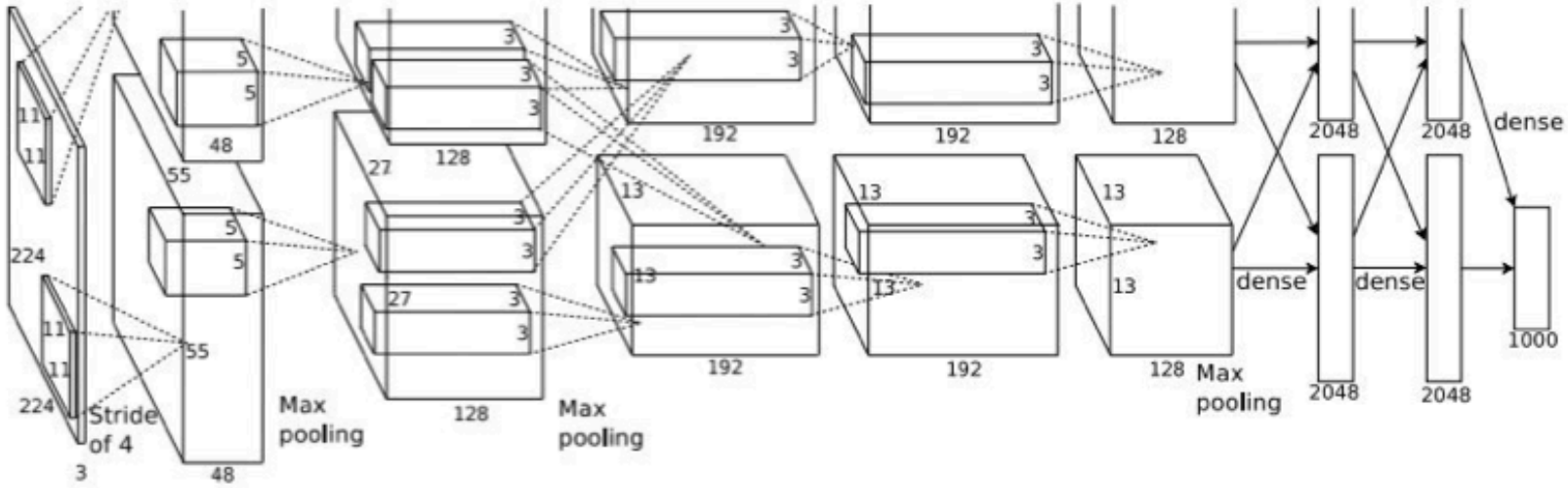
The History: ImageNet Challenge Winners



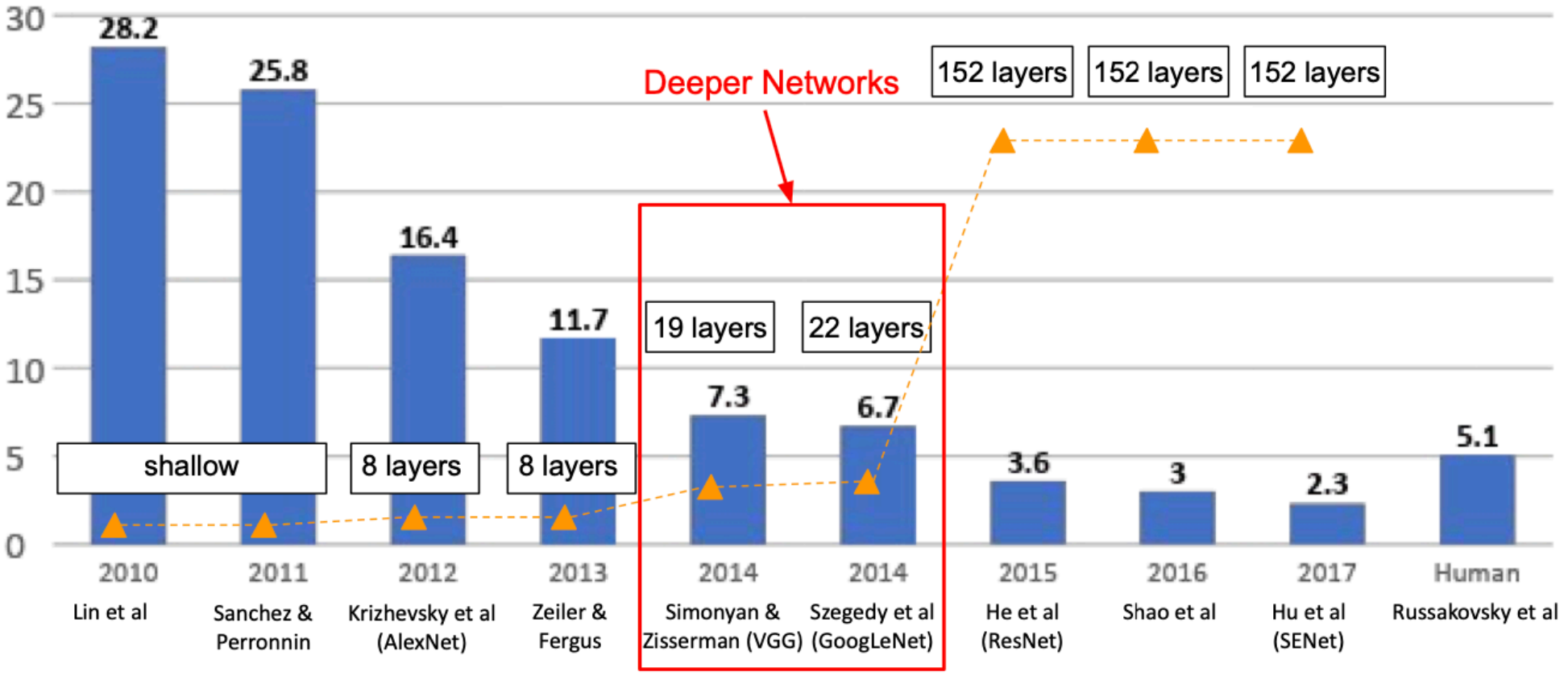
AlexNet

Architecture:

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



The History: ImageNet Challenge Winners



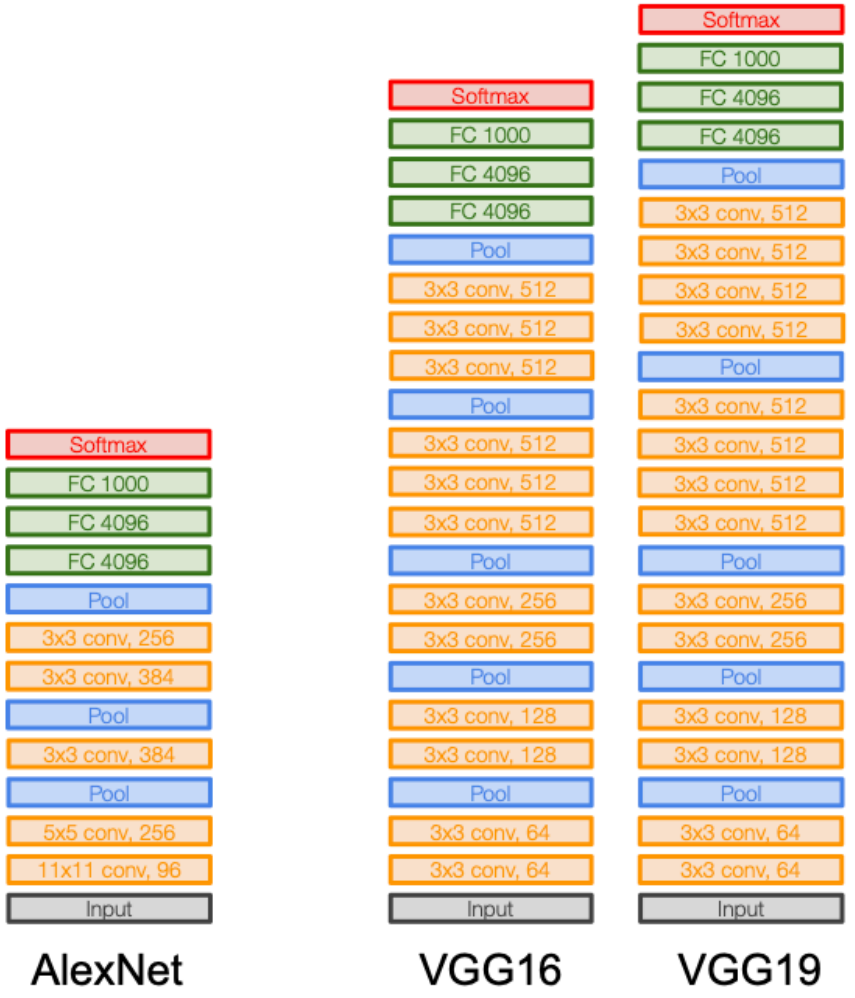
VGGNet

Small filters, Deeper networks

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

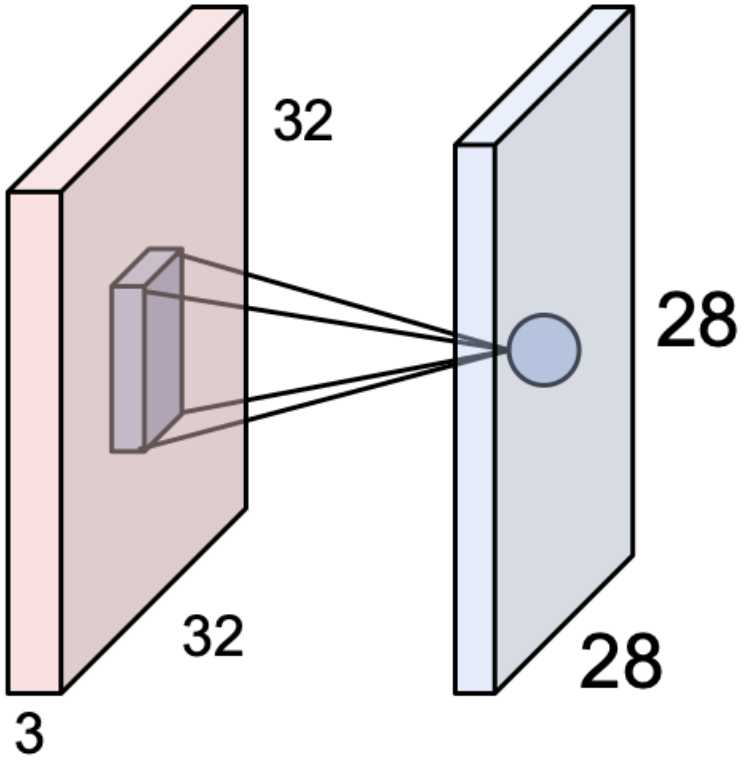
Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

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



Why use smaller filters? (3×3 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

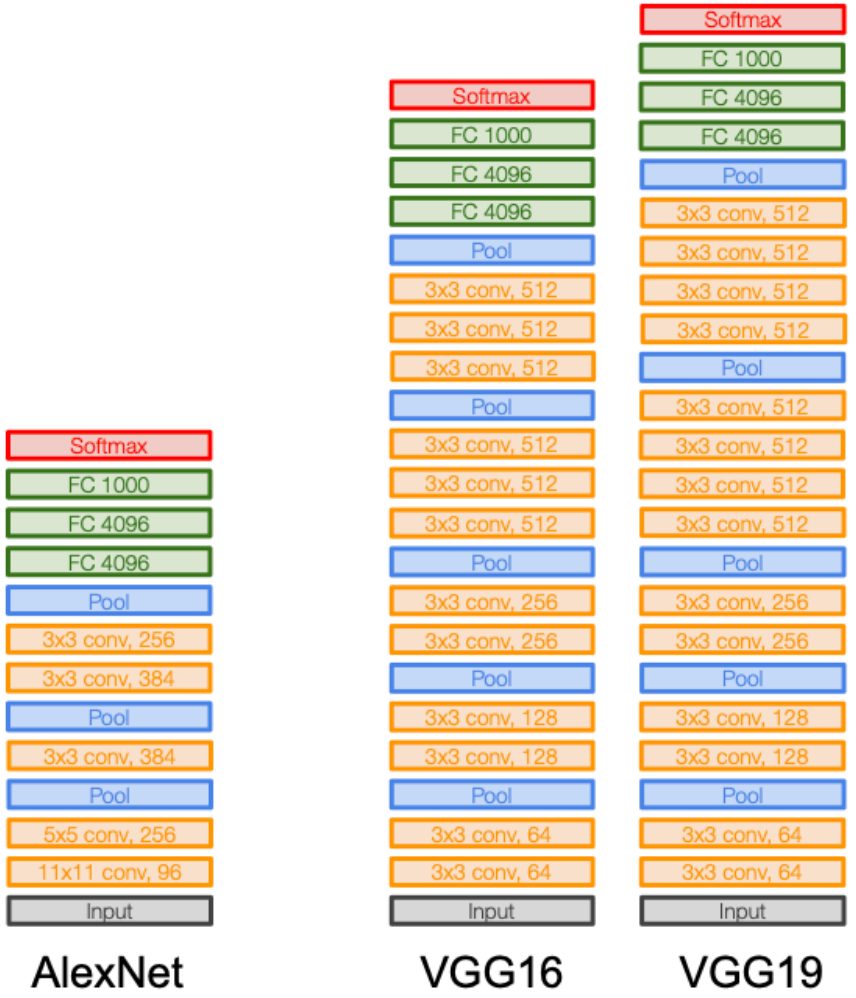
"5x5 filter" -> "5x5 receptive field for each neuron"

VGGNet

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

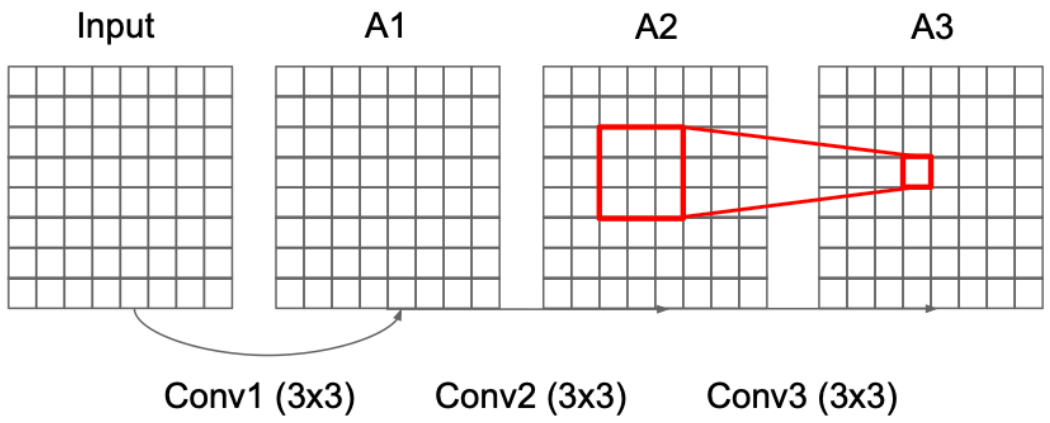
Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

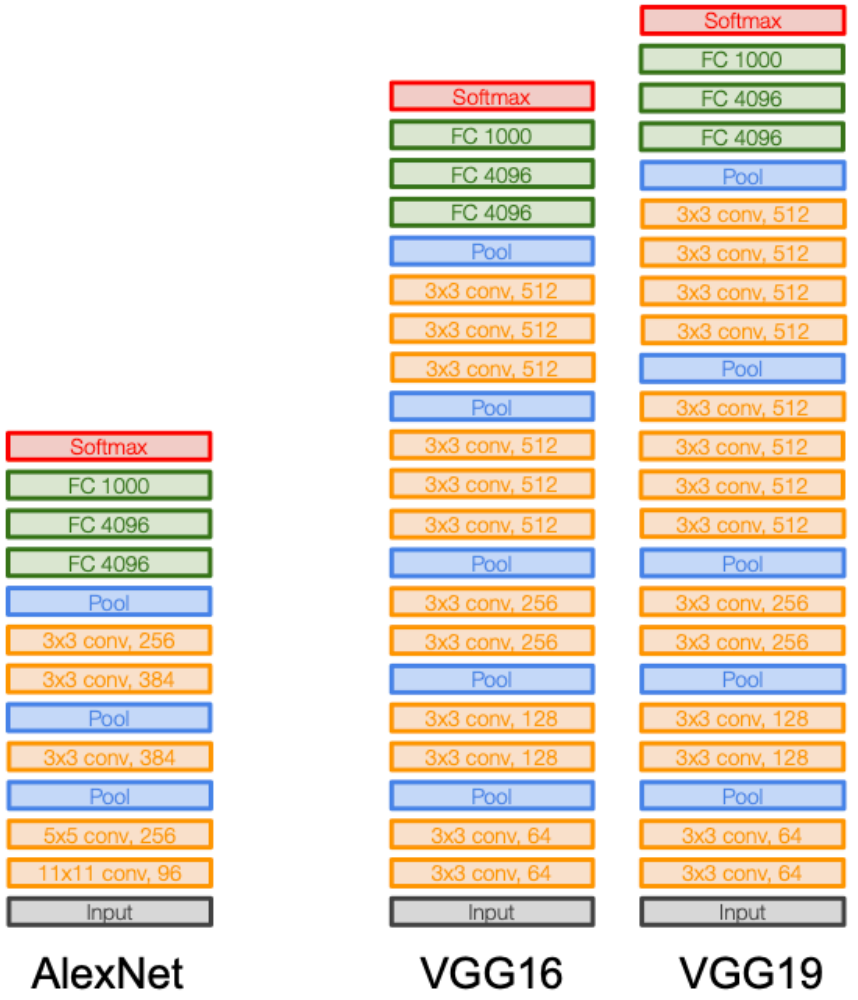
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



Receptive Field

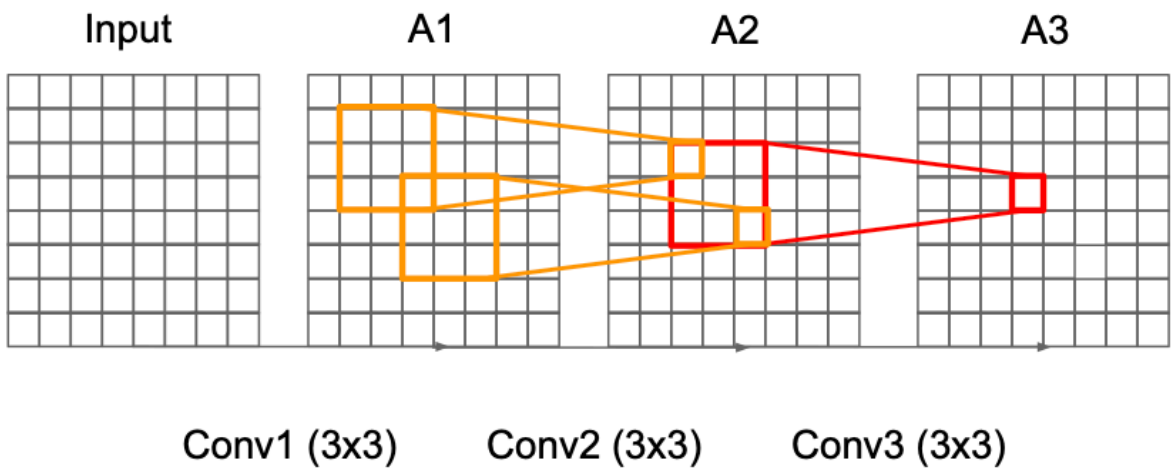
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

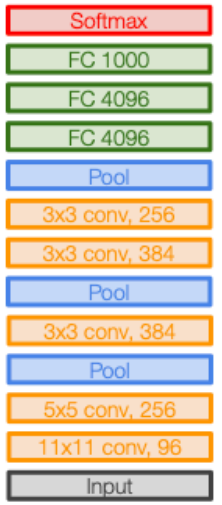




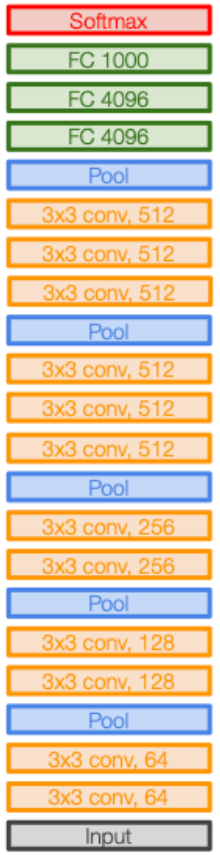
Receptive Field

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





AlexNet

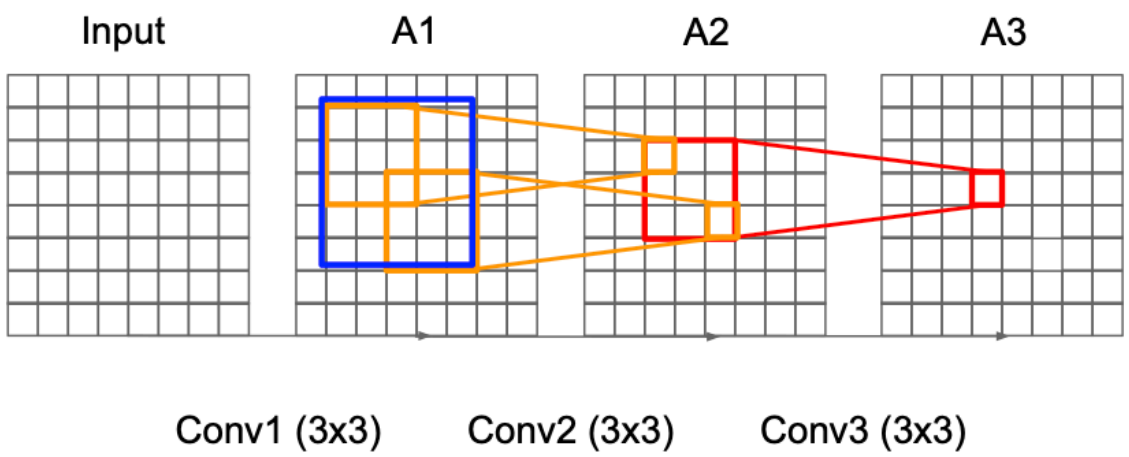


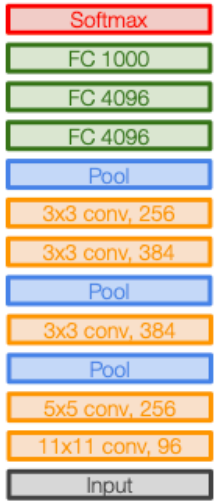
VGG16

VGG19

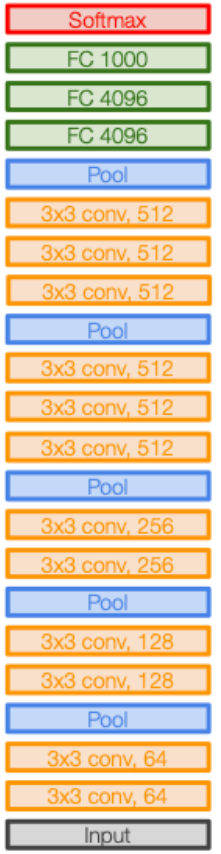
Receptive Field

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

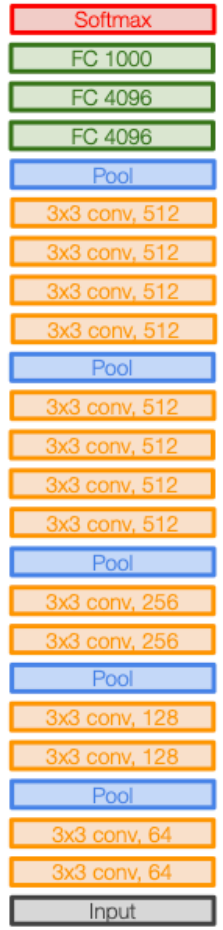




AlexNet



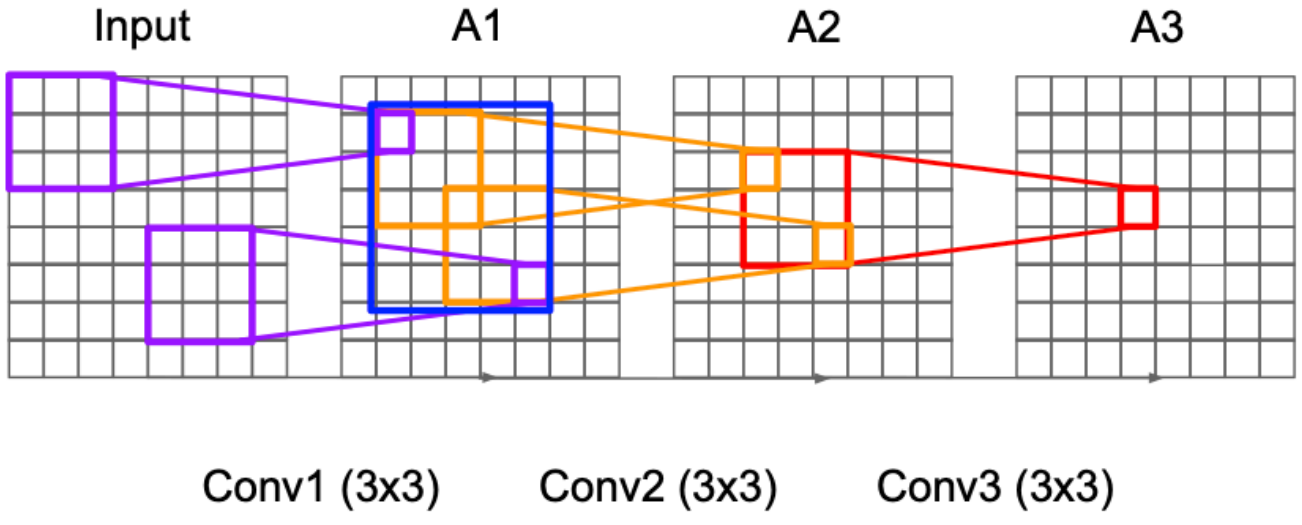
VGG16

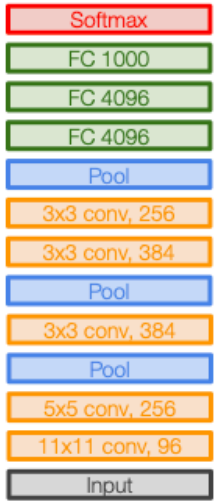


VGG19

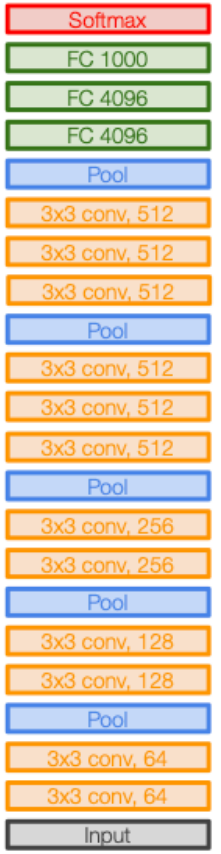
Receptive Field

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

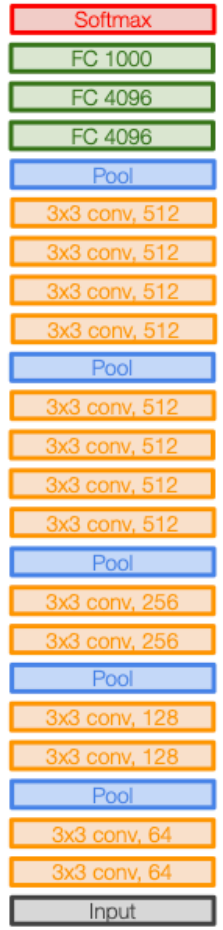




AlexNet



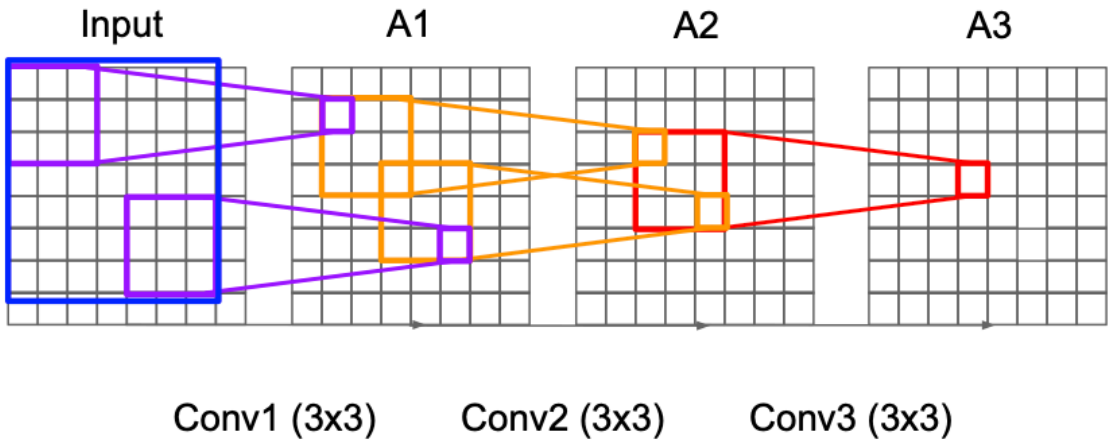
VGG16

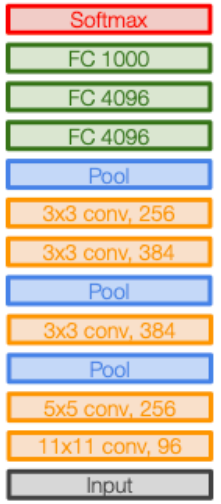


VGG19

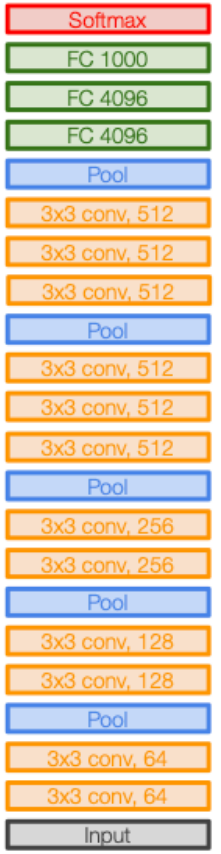
Receptive Field

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

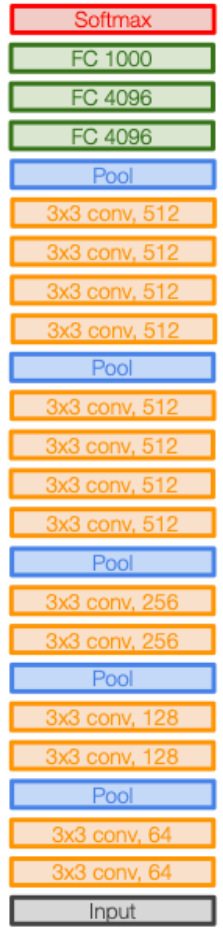




AlexNet



VGG16



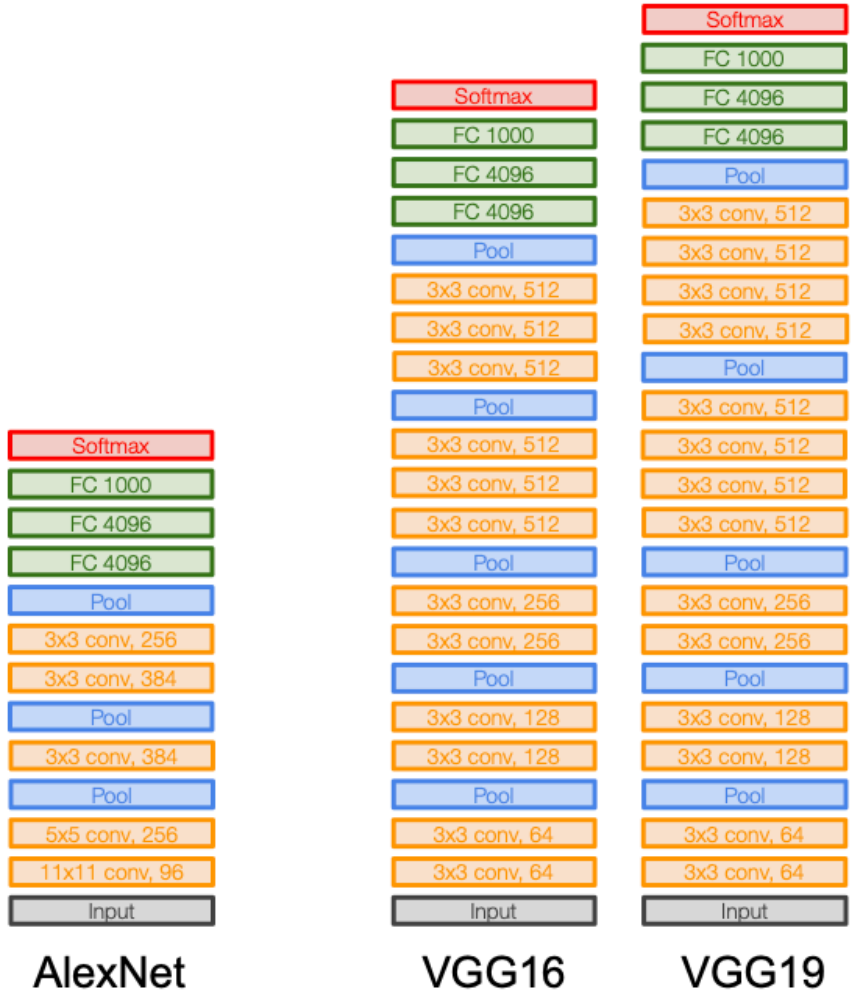
VGG19

Receptive Field

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

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

[7x7]



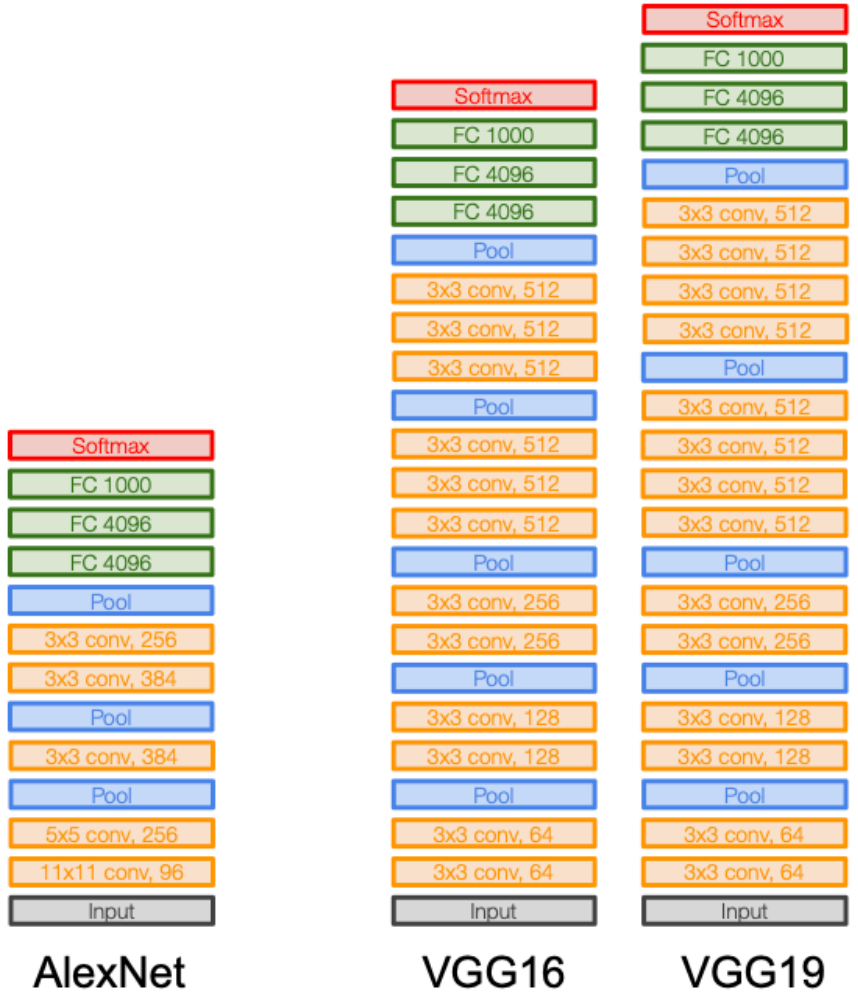
Receptive Field

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

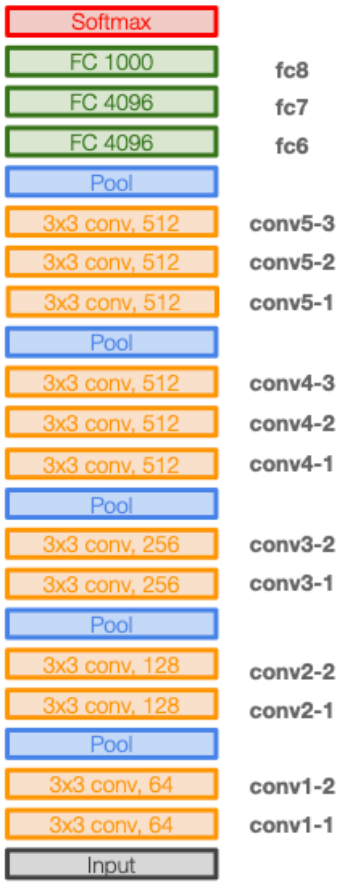


Receptive Field

(not counting biases)

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

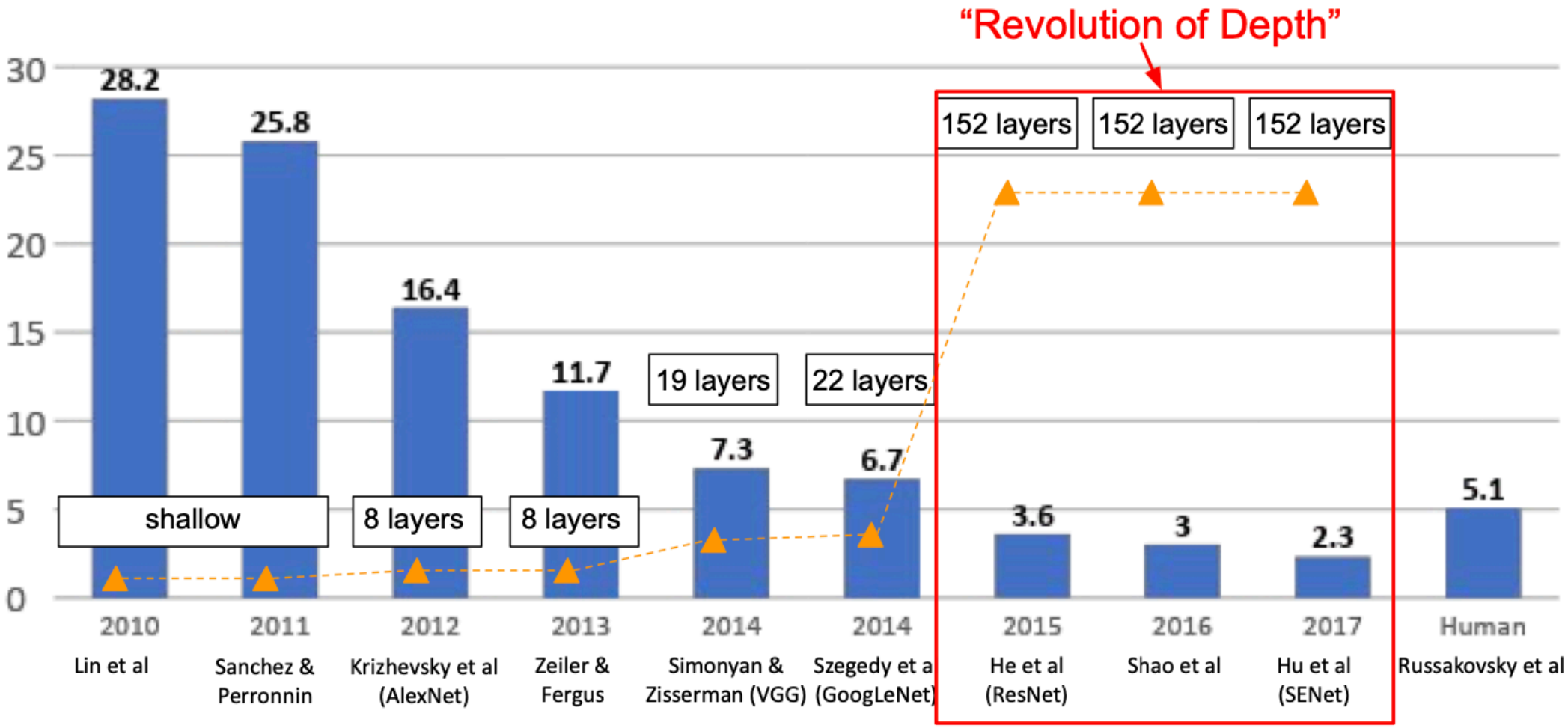
TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters



VGG16

Common names

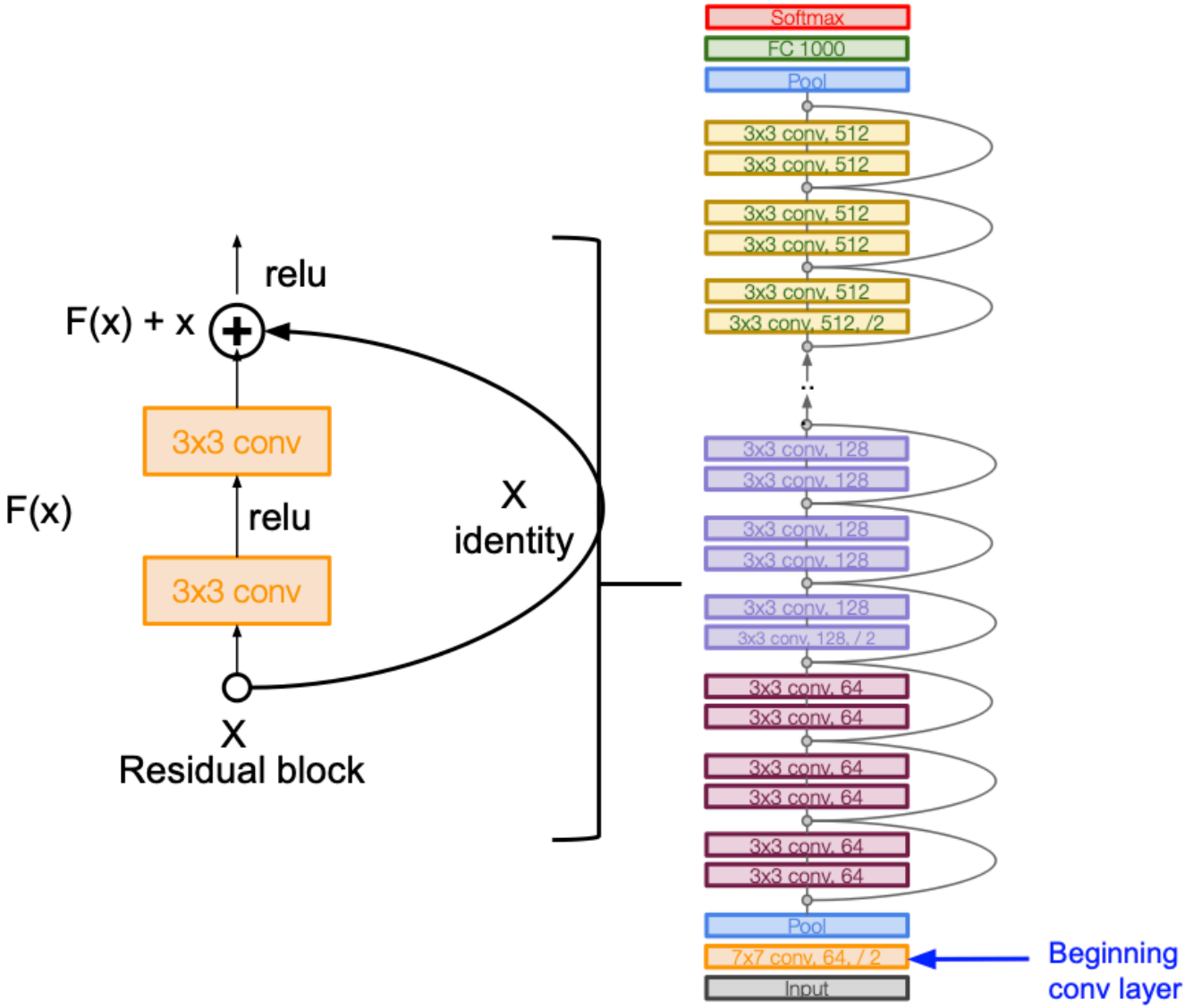
ResNet


"Revolution of Depth"
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
2011
2012
2013
2014
2014
2015
2016
2017
Human
Lin et al
Sanchez & Perronnin
Krizhevsky et al (AlexNet)
Zeiler & Fergus
Simonyan & Zisserman (VGG)
Szegedy et a (GoogLeNet)
He et al (ResNet)
Shao et al
Hu et al (SENet)
Russakovsky et al

ResNet

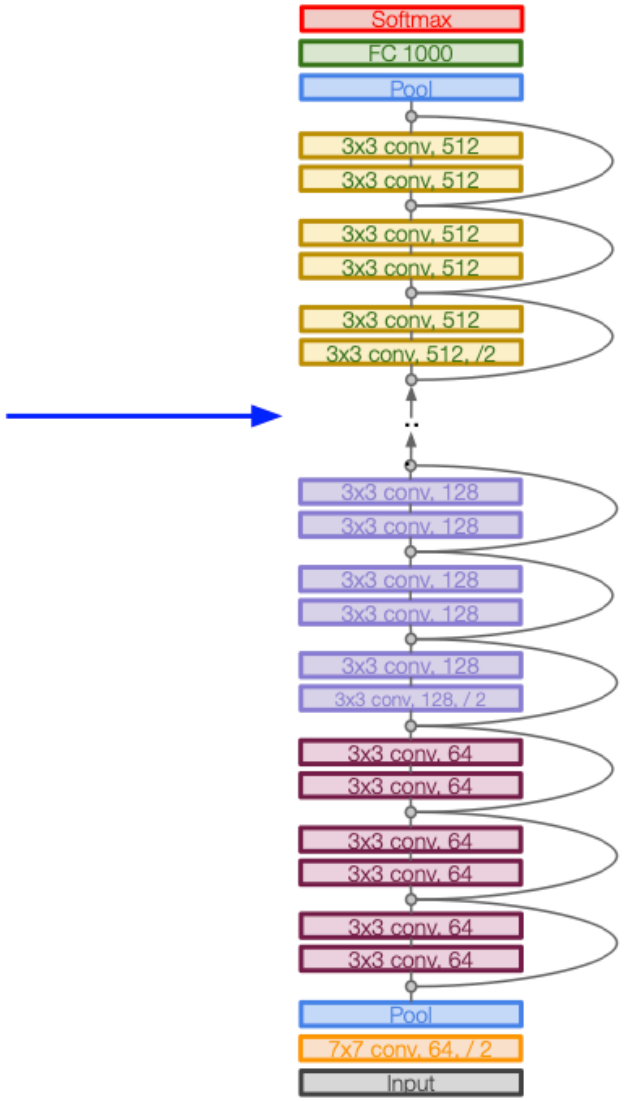
Full ResNet architecture:

- 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

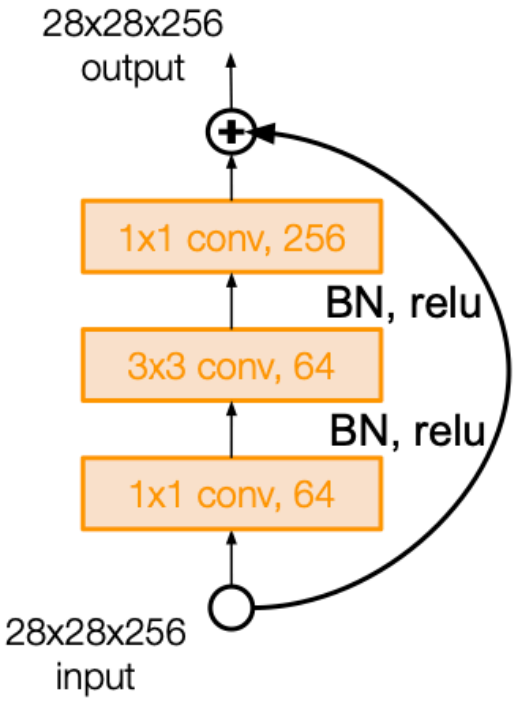
Total depths of 18, 34, 50, 101, or 152 layers for ImageNet



ResNet

[He et al., 2015]

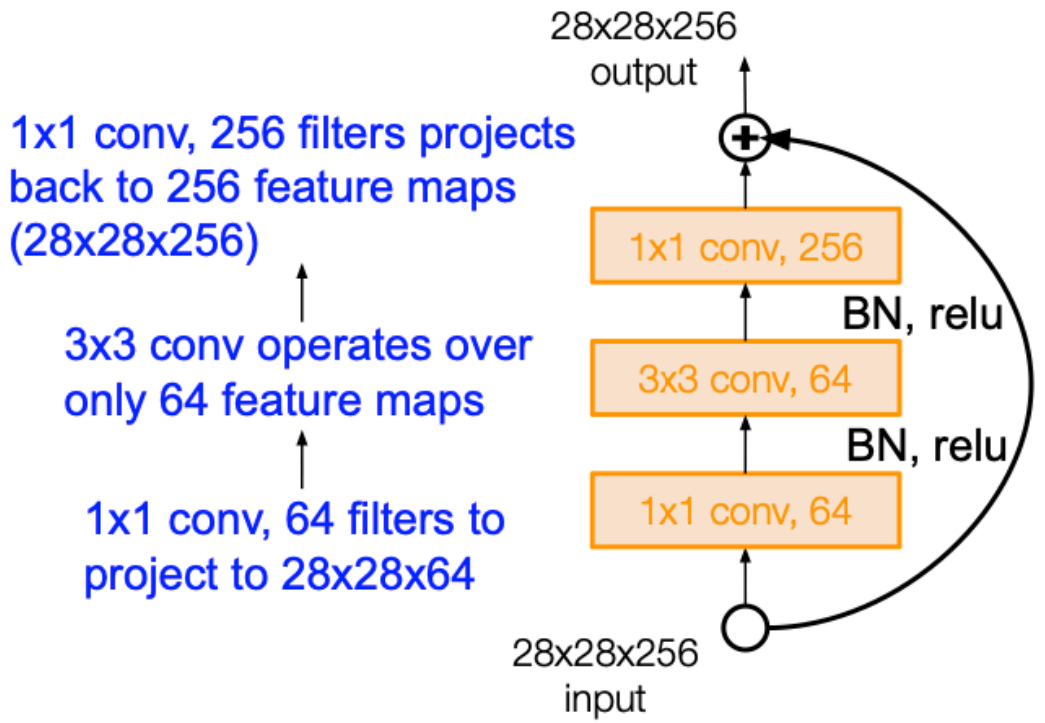
For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to GoogLeNet)



ResNet

[He et al., 2015]

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



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

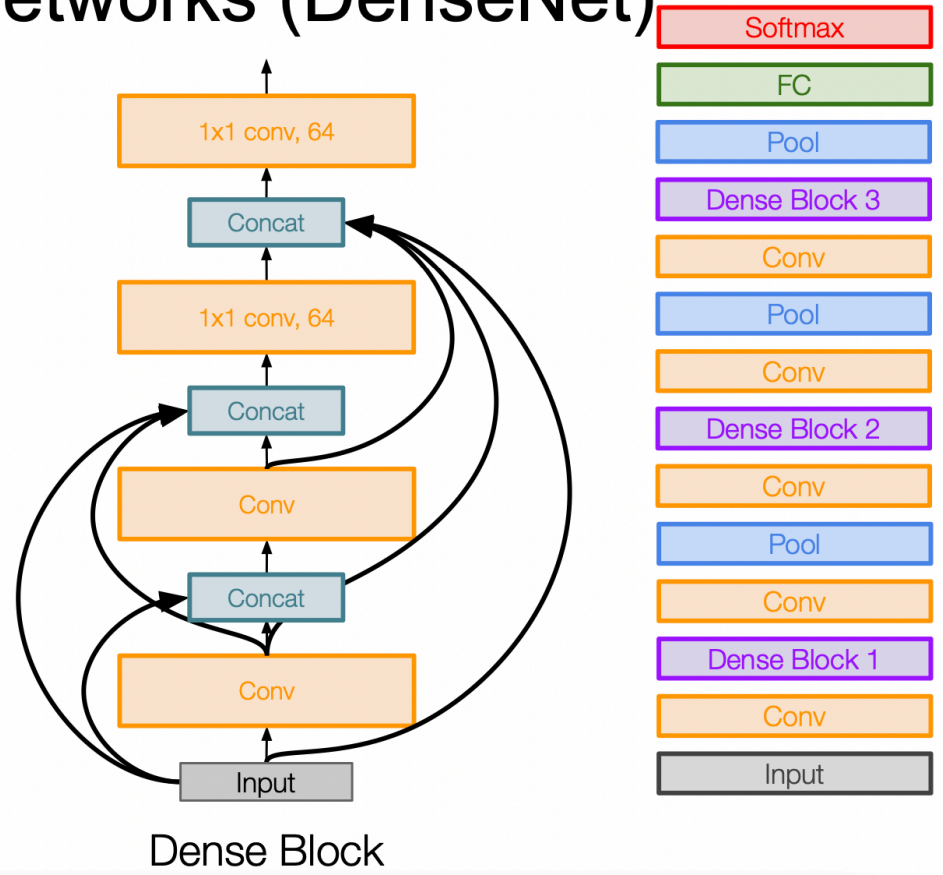
- Squeeze-and-Excitation Network (SENet)
- Wide Residual Networks
- ResNeXt
- DenseNet

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



Dense Block

Beyond ResNet

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

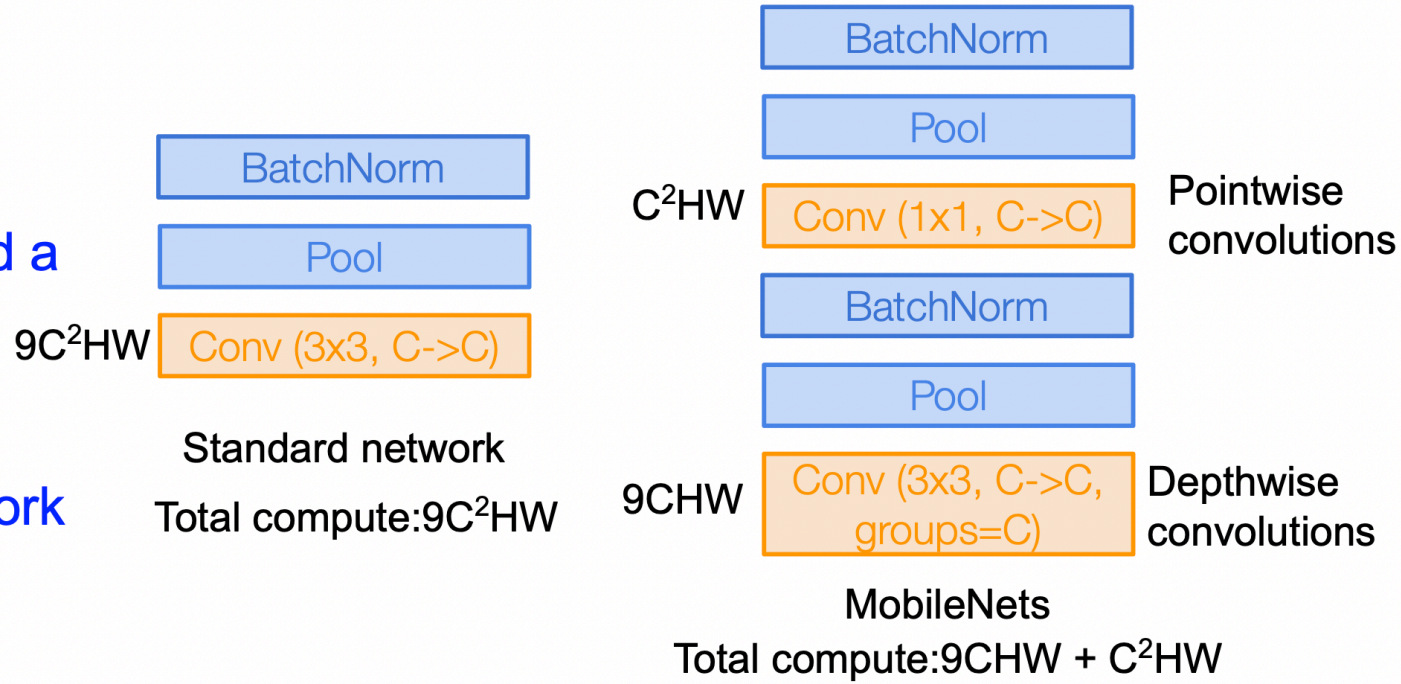
- Attention-based networks: ViT, SwinTransformer
- MLP-based networks

- 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 1x1 convolution
- Much more efficient, with little loss in accuracy
- Follow-up MobileNetV2 work in 2018 (Sandler et al.)
- ShuffleNet: Zhang et al, CVPR 2018



Beyond ResNet

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

- ViT, swinTransformer, MLP-based networks

- MobileNet —> efficiency

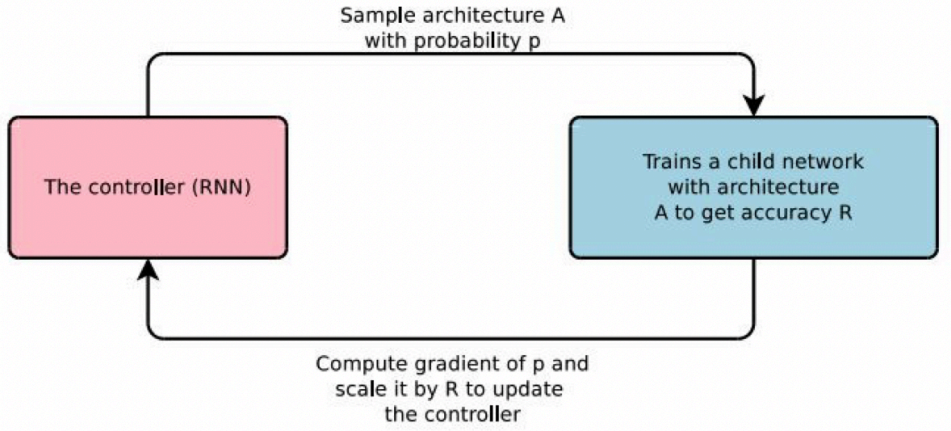
- Neural architecture search

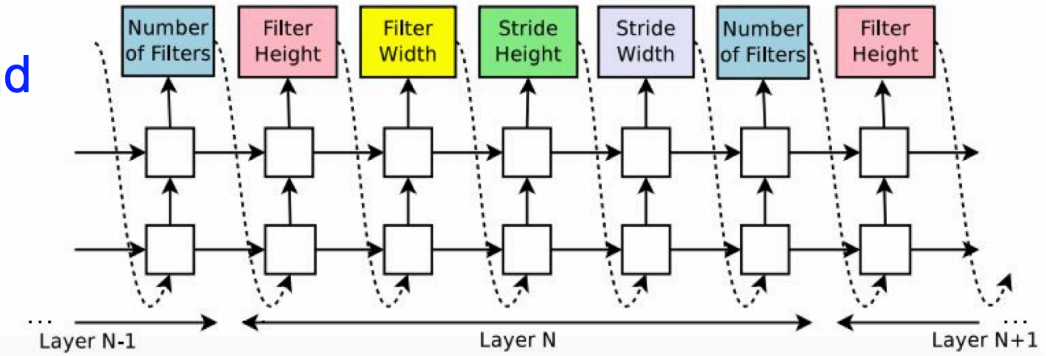
Learning to Search for Network Architecture

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).

Data

Model

Output: which class

$x =$ 

$h($  $,\theta)$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

cat

dog

panda

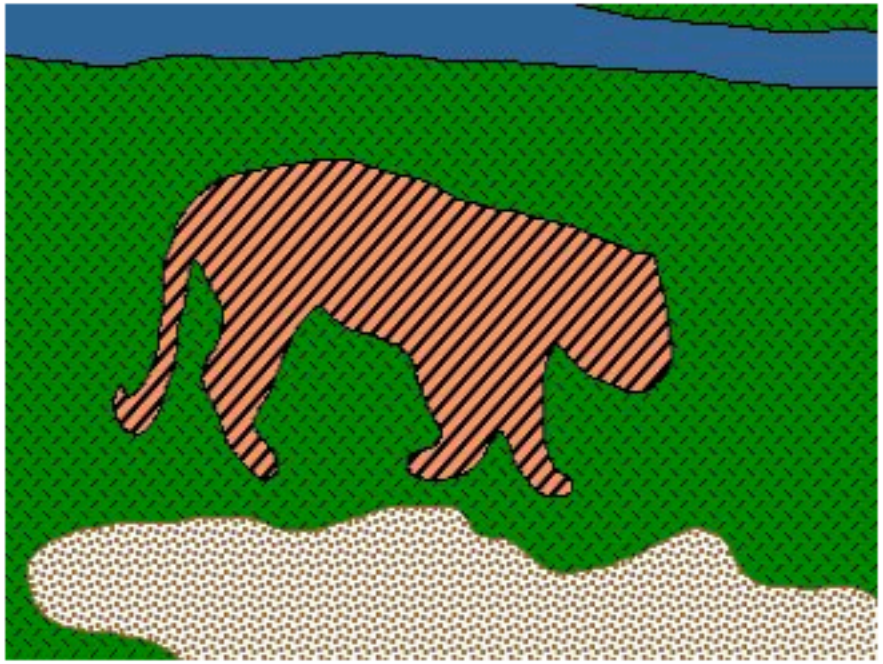
elephant

$y \in \{0,1\}$

Parameters or weights

Image Segmentation

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

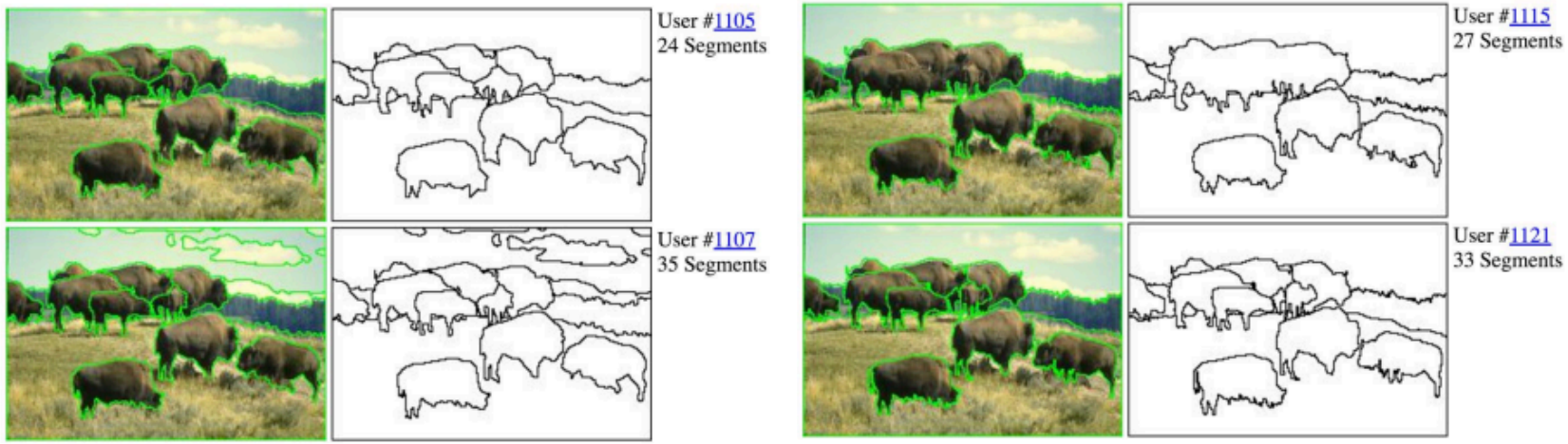


Slides credit: Steve Seitz, Kristen Grauman

Image Segmentation

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

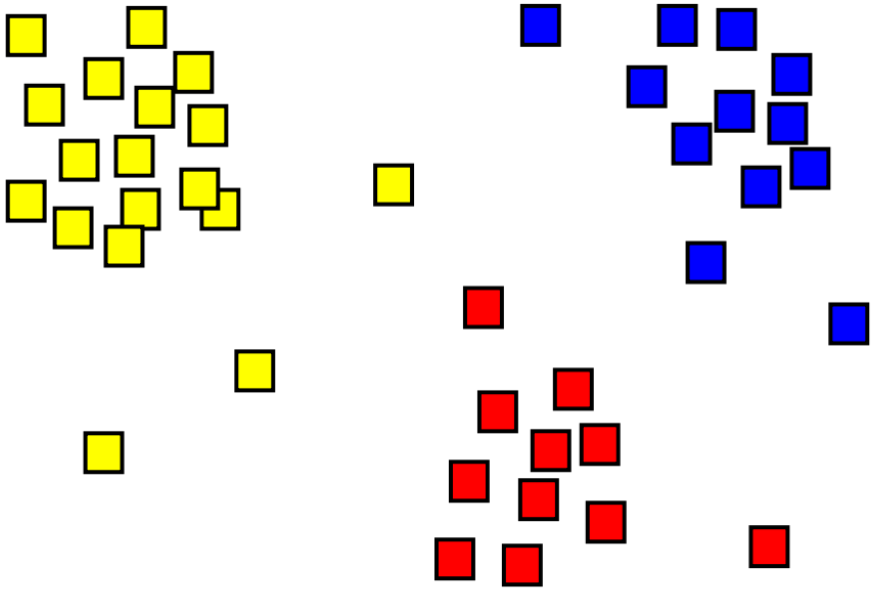
“Divide each image into pieces, where each piece represents a distinguished thing in the image. It is important that all of the pieces have approximately equal importance.“



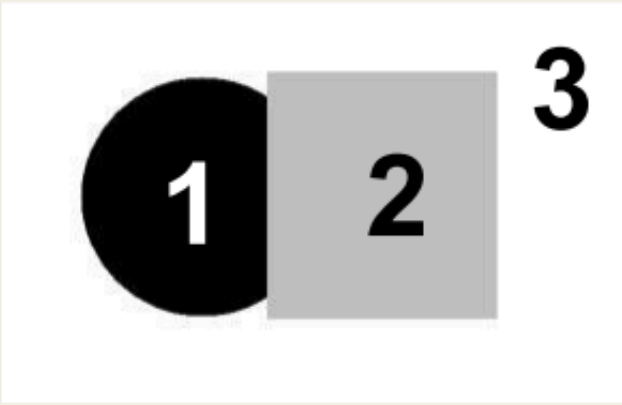
Berkeley Segmentation Dataset [Martin et al., ICCV 2021]

Classic Techniques for Image Segmentation

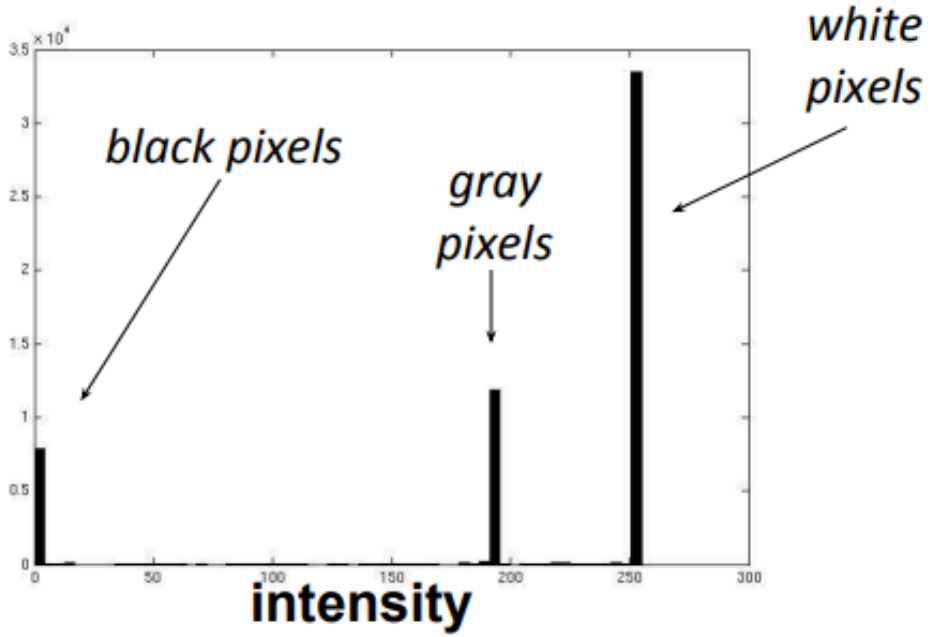
- Philosophy: grouping and clustering
  - Clustering: group together similar data points and represents them with a single token



Clustering: Toy Example



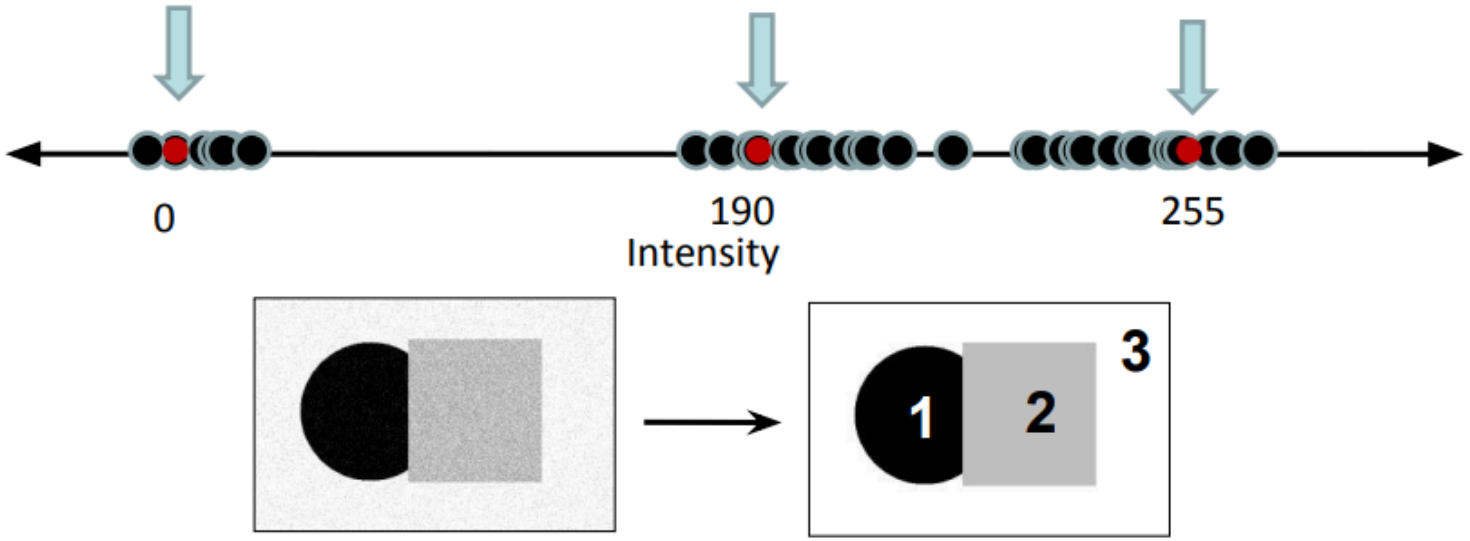
input image



- 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?

Slides credit: Kristen Grauman

Clustering-based Segmentation



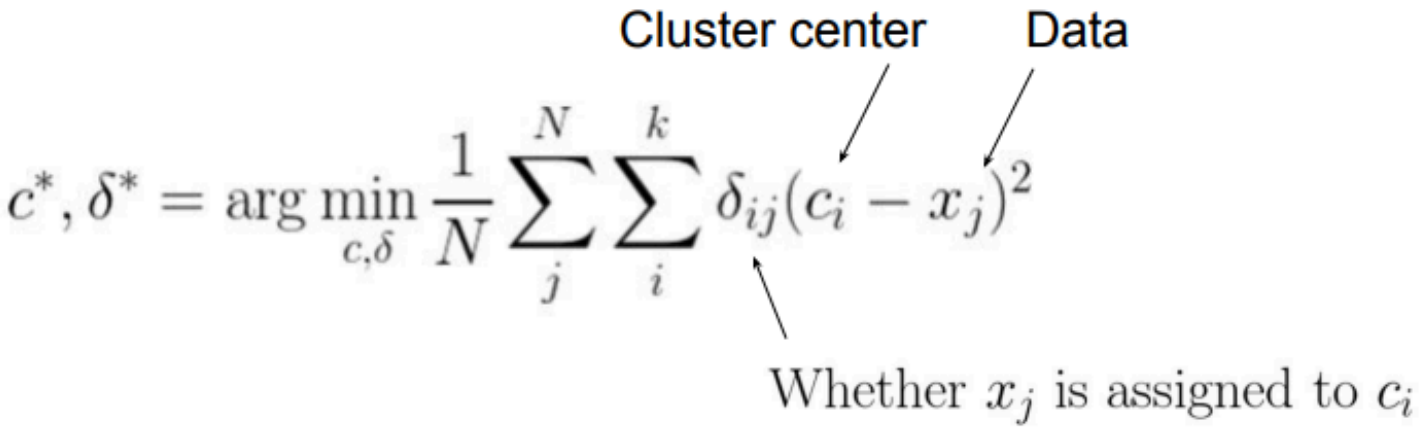
- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize Sum of Square Distance (SSD) between all points and their nearest cluster center c_i:

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

Slides credit: Kristen Grauman

Objective Function

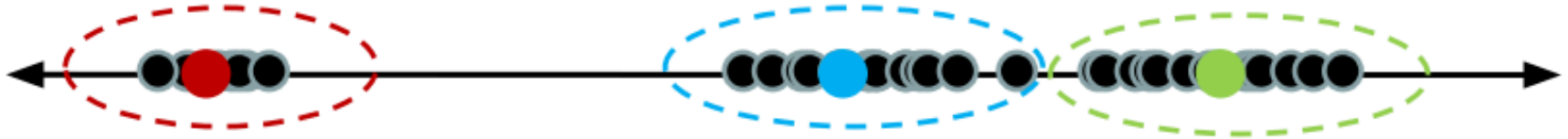
- Goal: minimize the distortion in data given clusters
  - Preserve information



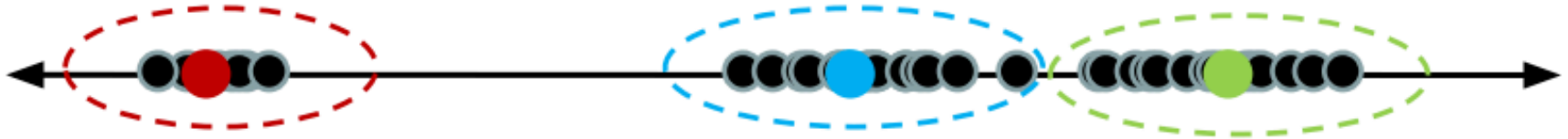
Slides credit: Derek Hoiem

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.



Slides credit: Kristen Grauman

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

[Lloyd, 1957]

K-Means Clustering

- **Input:** N examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ ($\mathbf{x}_n \in \mathbb{R}^D$); the number of partitions K
- **Initialize:** K cluster centers $\mu_1, \ldots, \mu_K$. 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

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

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

  - Recompute the new cluster centers μ_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

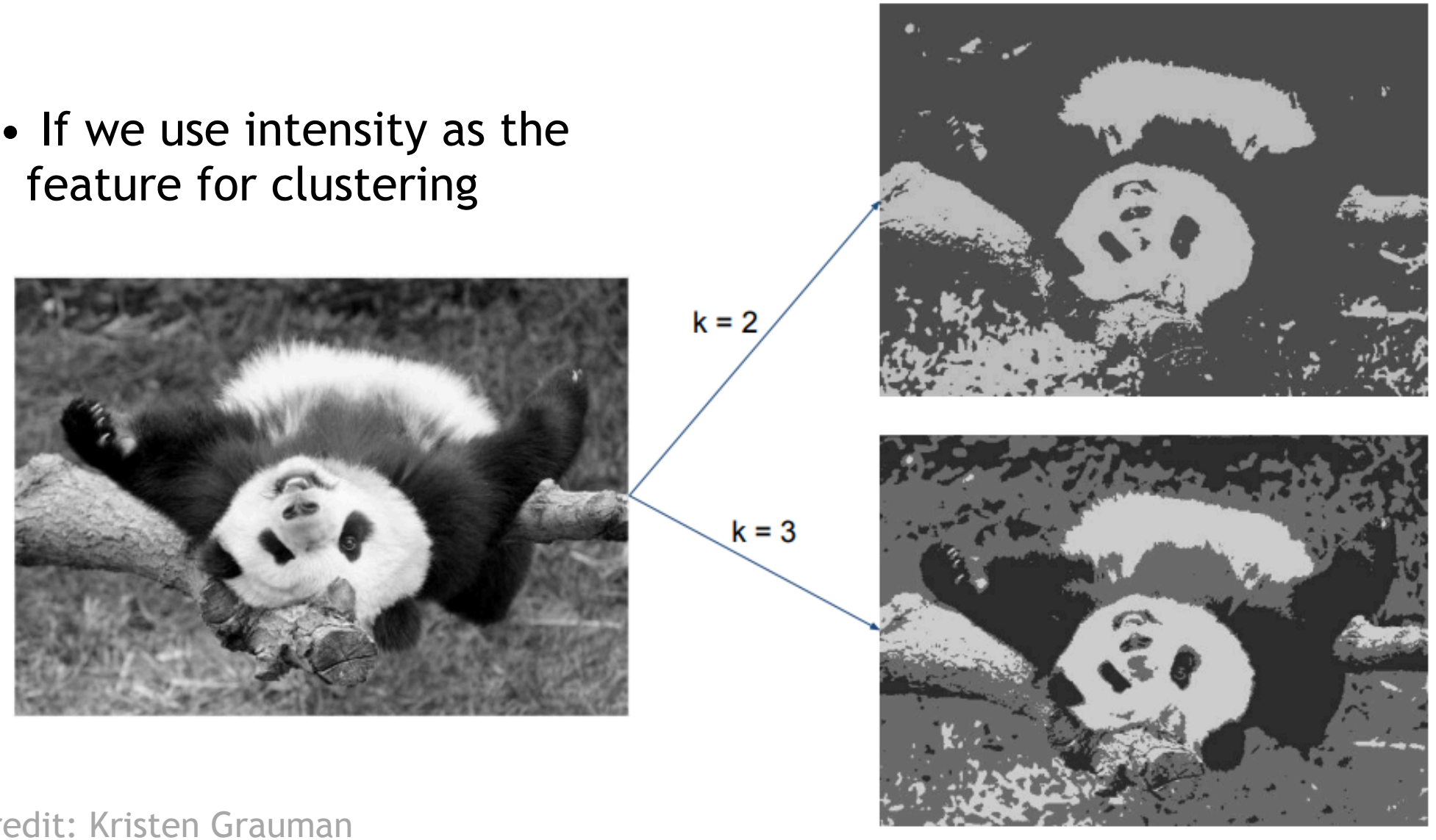
Slides credit: P. Rai

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



Slides credit: Kristen Grauman

K-Means Clustering Results

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

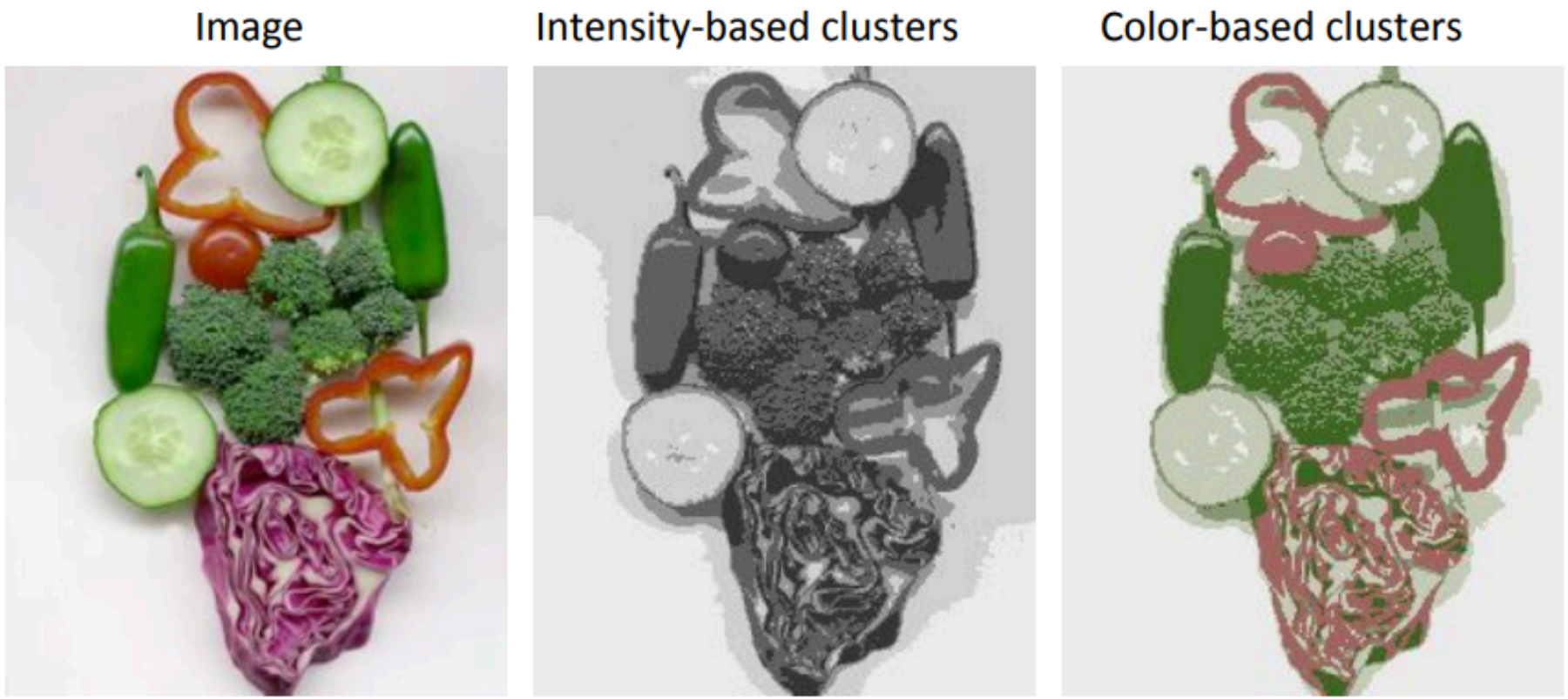
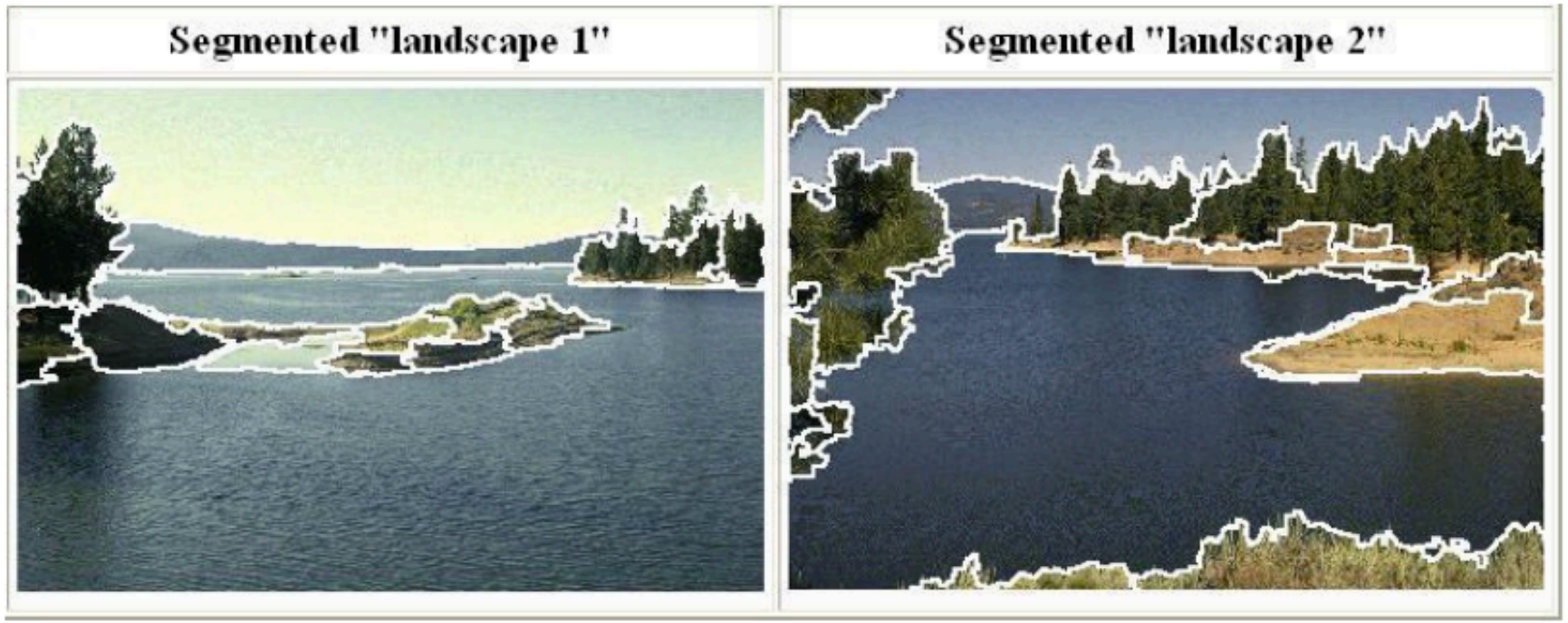


Figure credit: Forsyth & Ponce

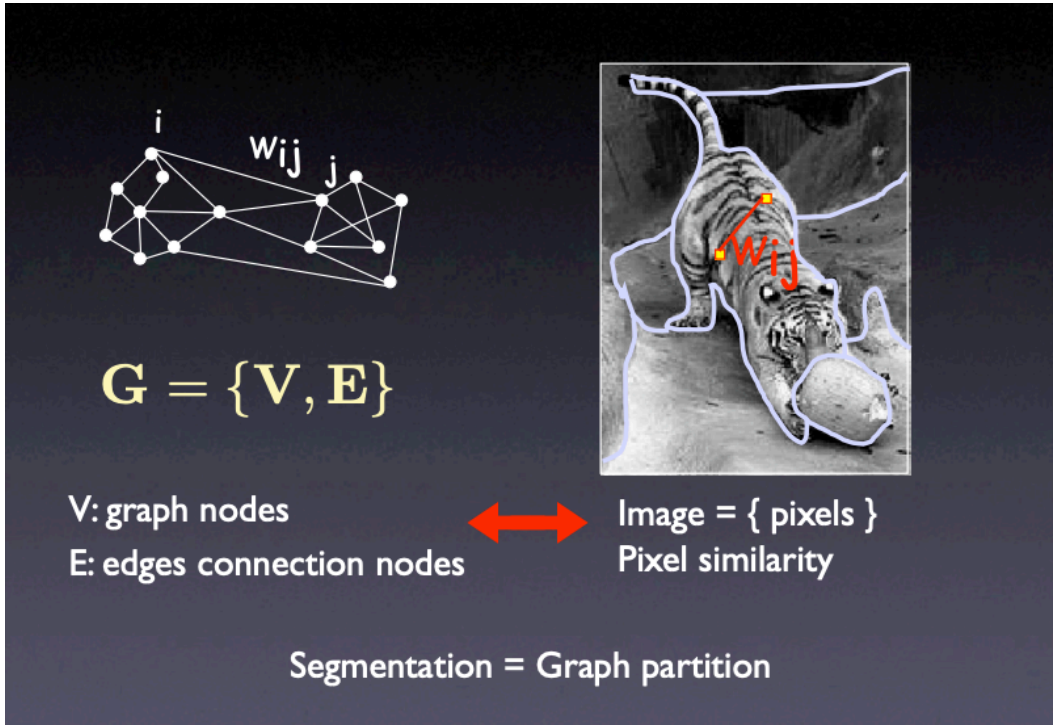
Mean-Shift Segmentation

- An advanced and versatile technique for clustering-based segmentation



D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

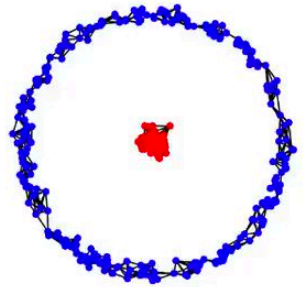
Graph-based Image Segmentation



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.



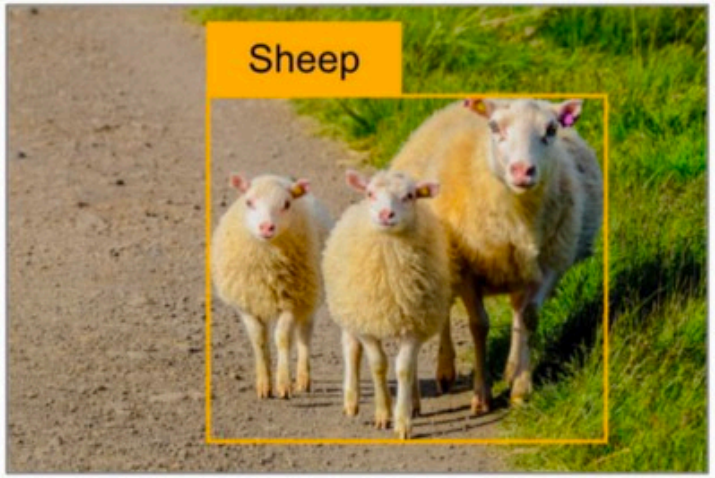
https://www.cis.upenn.edu/~jshi/GraphTutorial/Tutorial-ImageSegmentationGraph-cut1-Shi.pdf

Summary of Grouping-based Segmentation

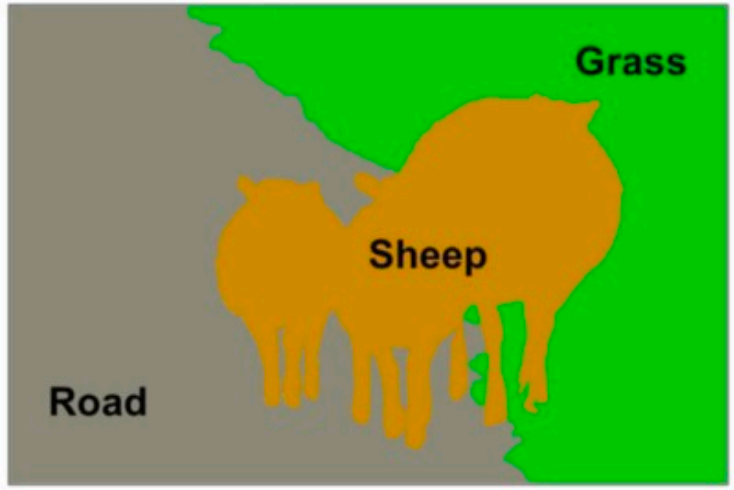
- A mid-level vision task
  - Grouping-based or clustering based: a bottom-up approach
  - Doesn’t care about semantics
- Unsupervised
- To improve, we can use more expressive features when performing clustering.

We Care About Semantics

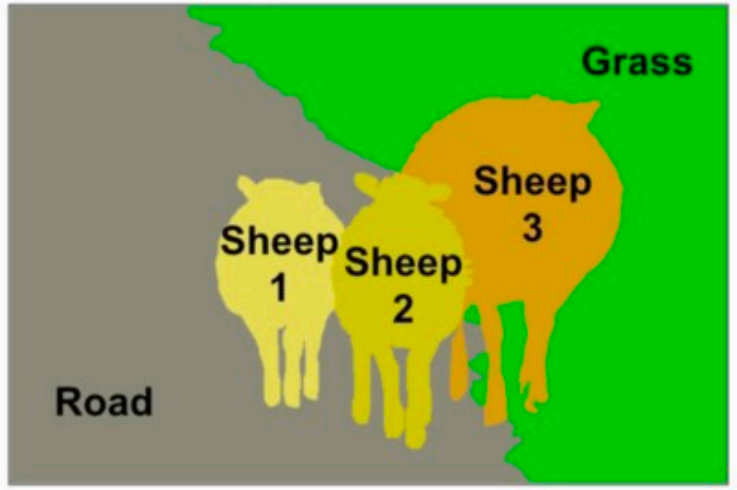
Classification + localization



Instance Segmentation



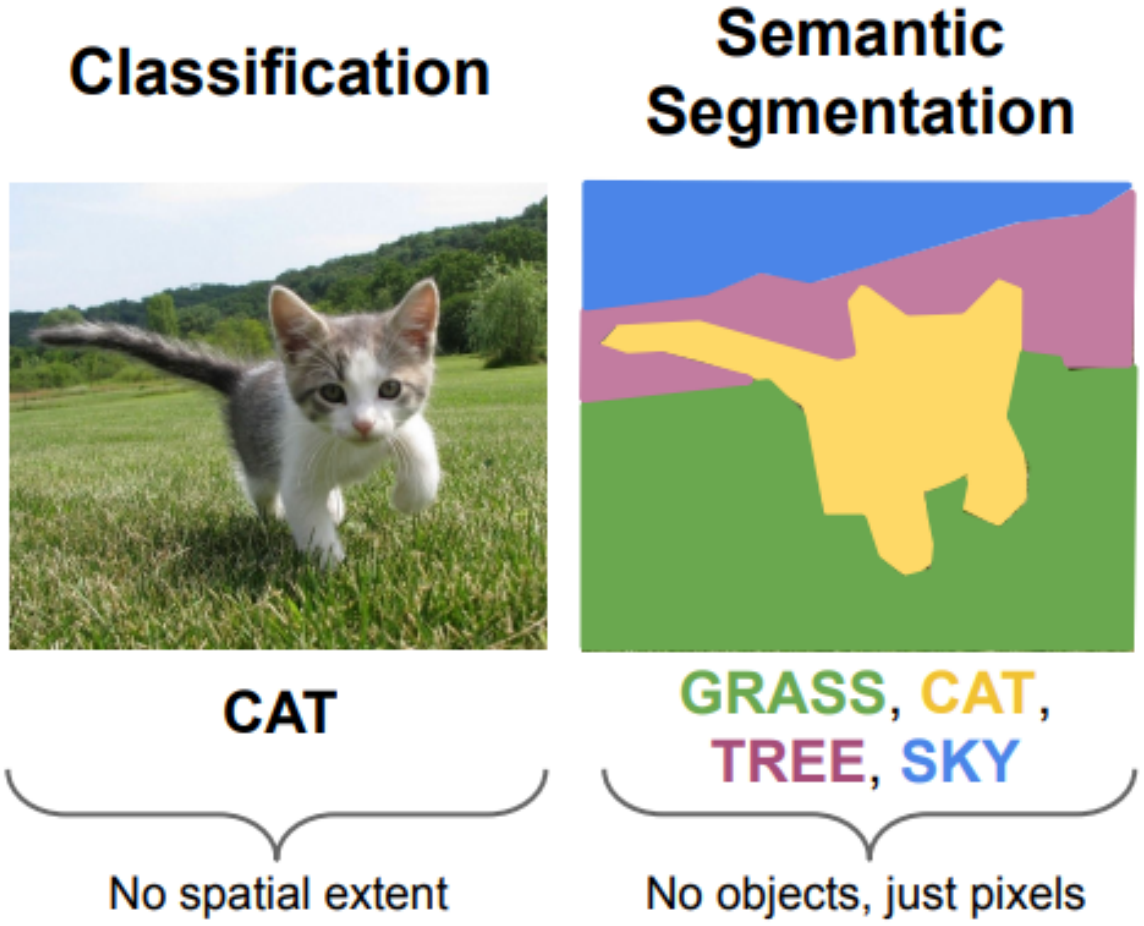
Semantic Segmentation



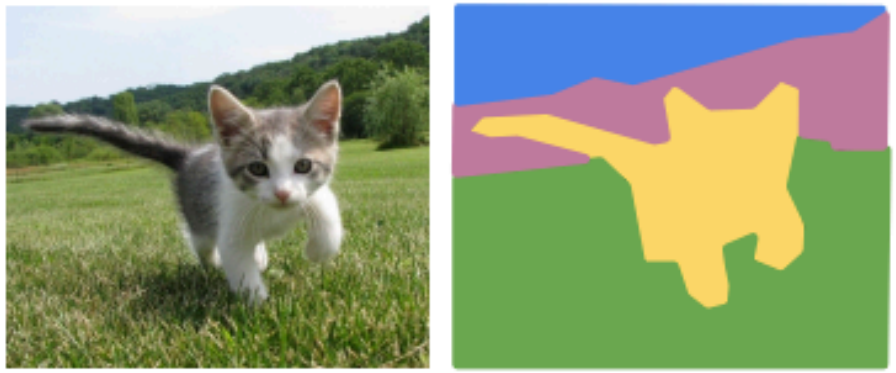
Semantic Instance Segmentation

Semantic Segmentation

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

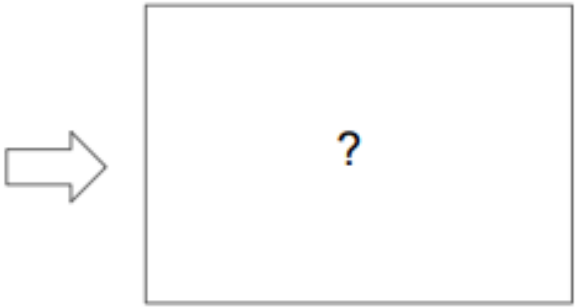


Semantic Segmentation



GRASS, CAT,
TREE, SKY, ...

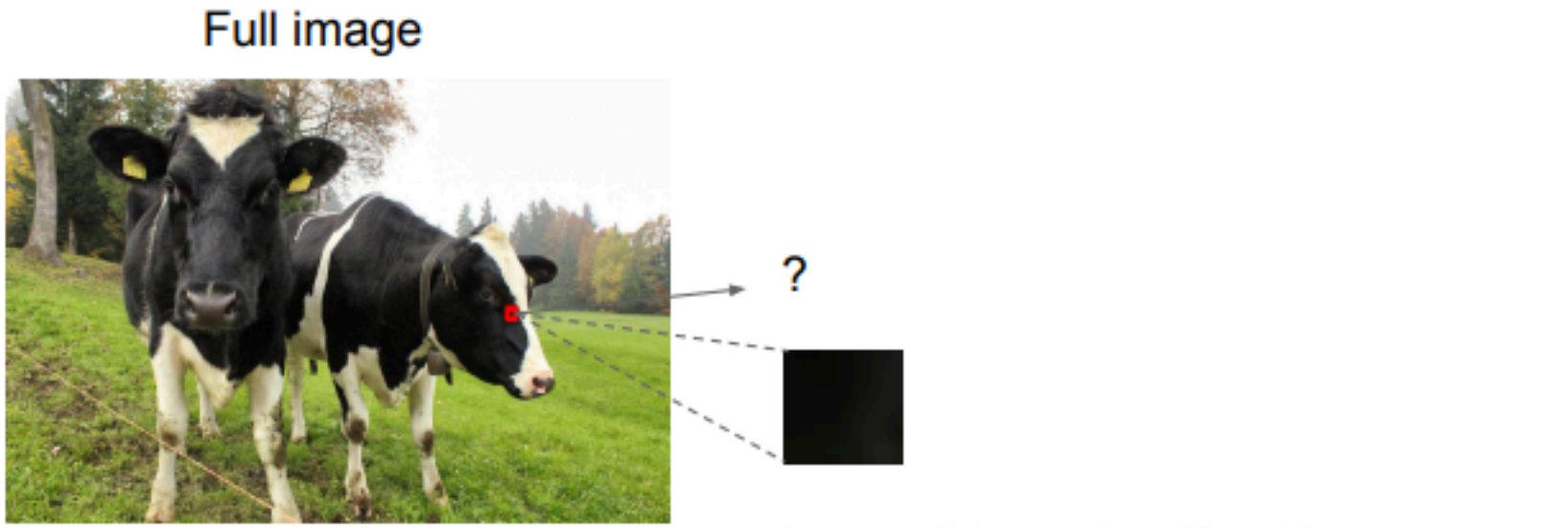
Paired training data: for each training image,
each pixel is labeled with a semantic category.



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

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

Semantic Segmentation using Sliding Window

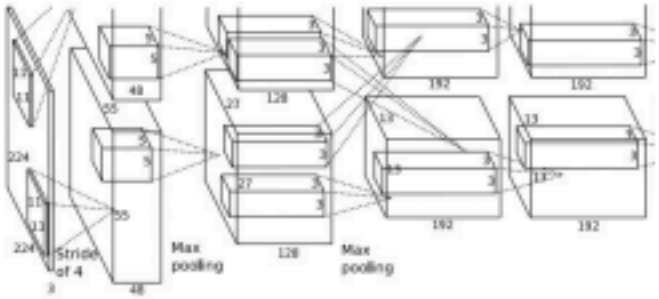


Impossible to classify without context

Q: how do we include context?

Semantic Segmentation using CNN

Full image

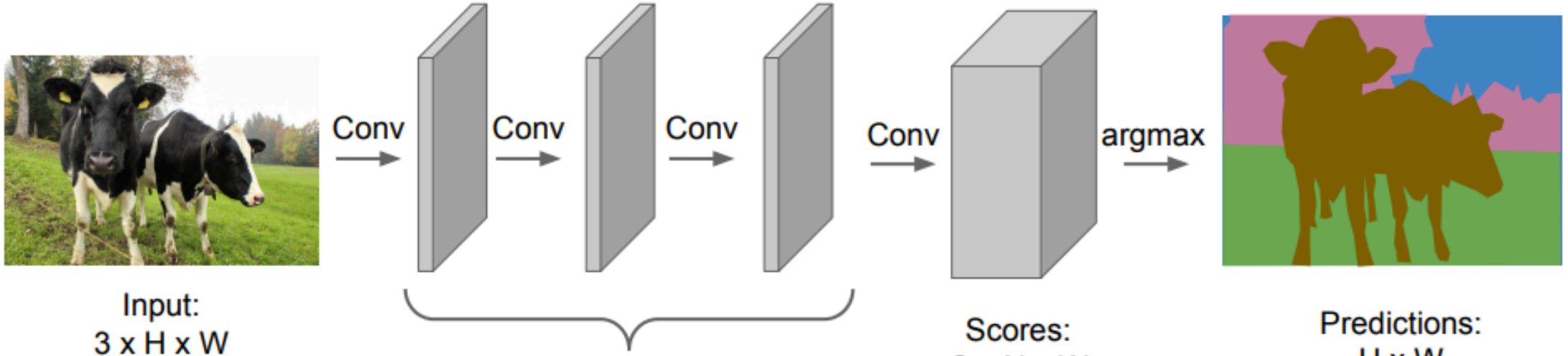


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

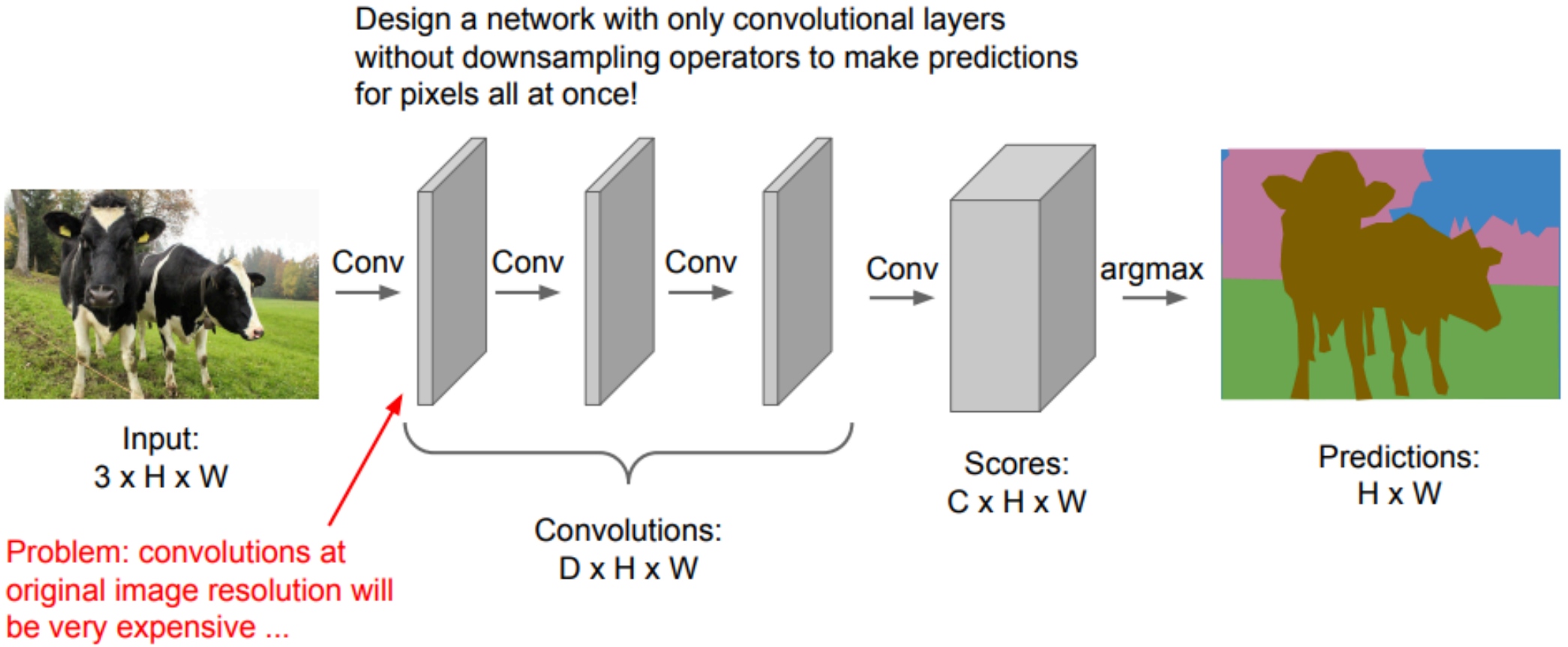
Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

Semantic Segmentation using Fully Convolution

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

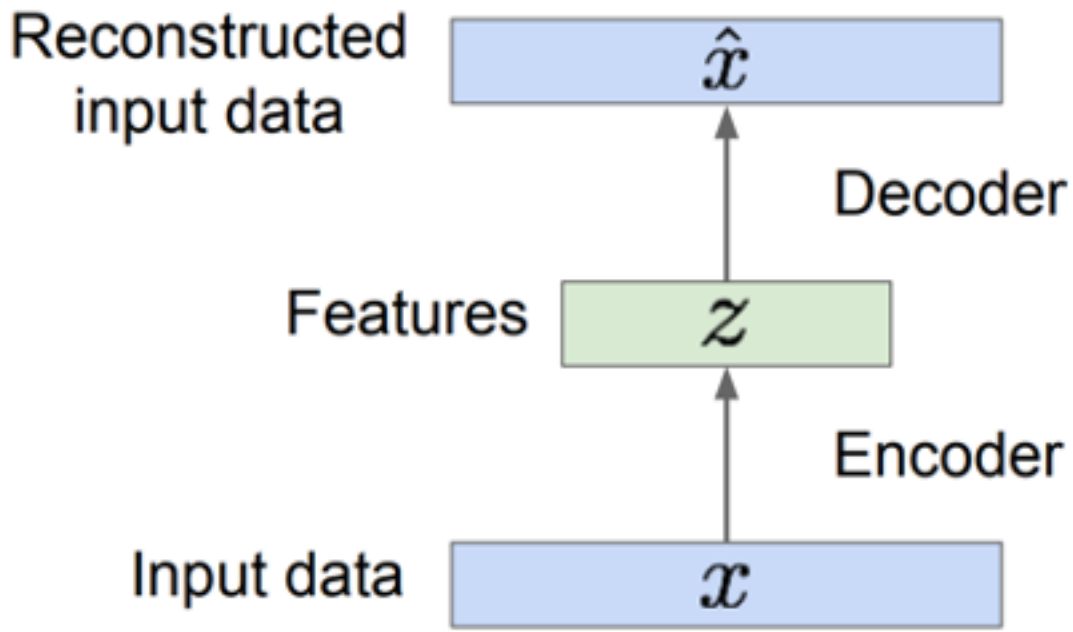


Semantic Segmentation using Fully Convolution



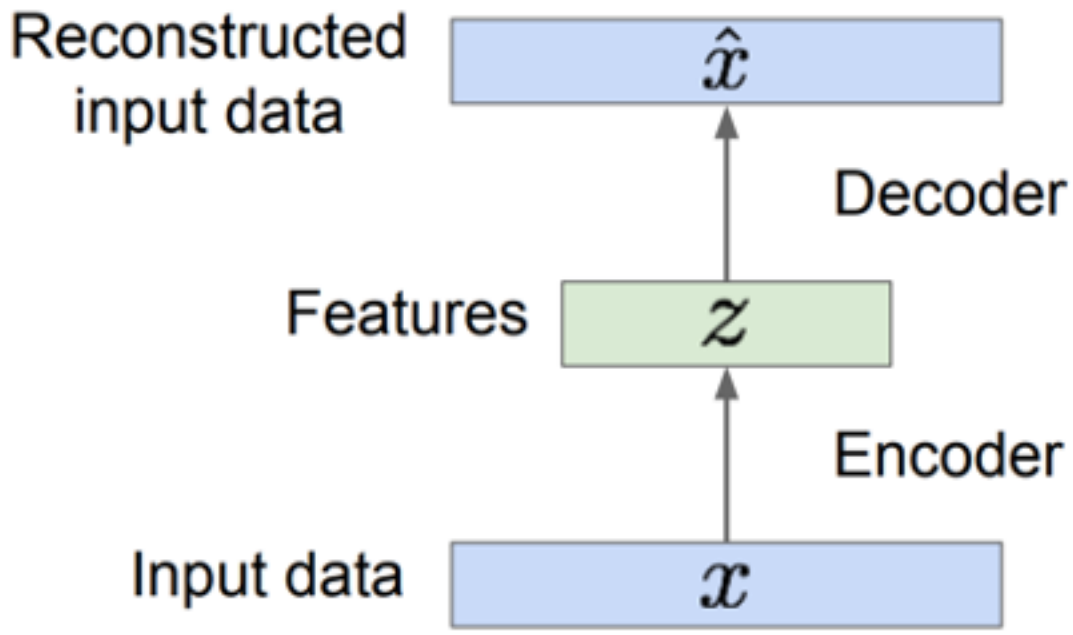
We need to reduce resolutions.

Auto-Encoder



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

Auto-Encoder

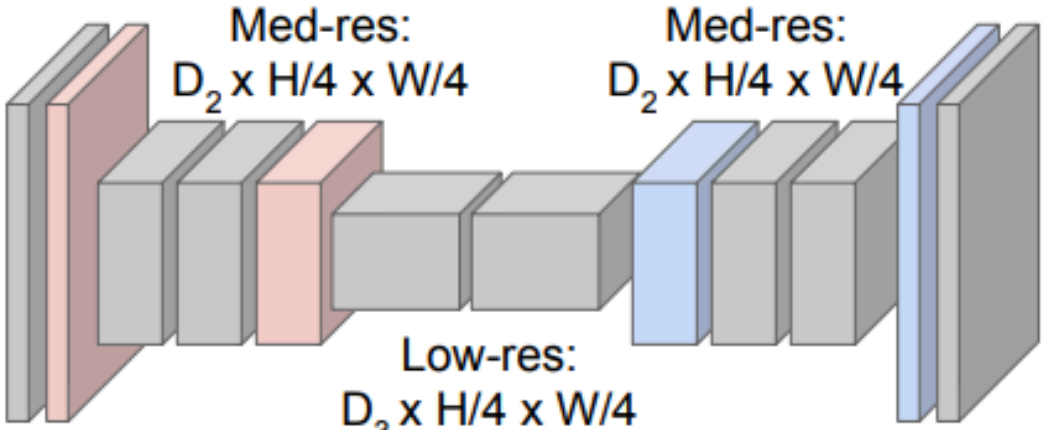


- Understanding AE
  - Information bottleneck: the dimension of z space is much smaller than that of x
  - Get rid of redundant information via dimension reduction
  - The first step to all advanced segmentation networks

Semantic Segmentation using Fully Convolution

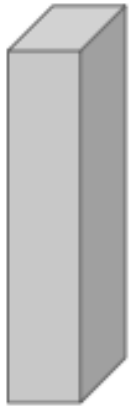
Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2



C x H x W

Predictions:
H x W

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

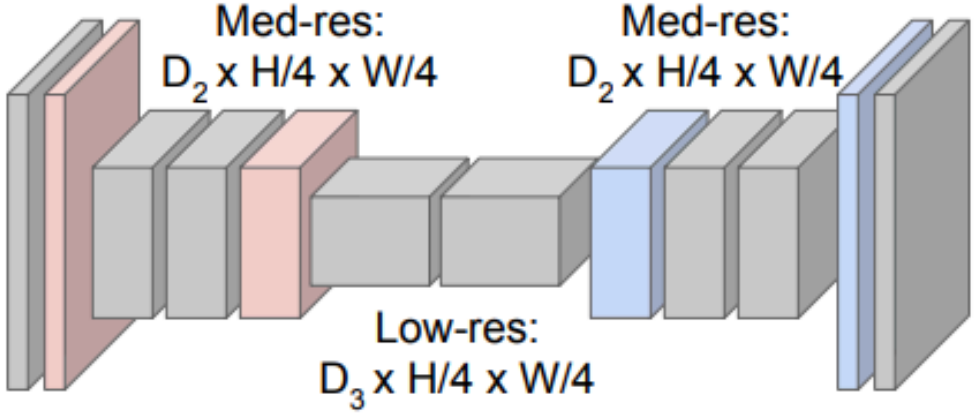
Semantic Segmentation using Fully Convolution

Downsampling:
Pooling, strided convolution

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

Upsampling:
???

Input:
3 x H x W



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

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

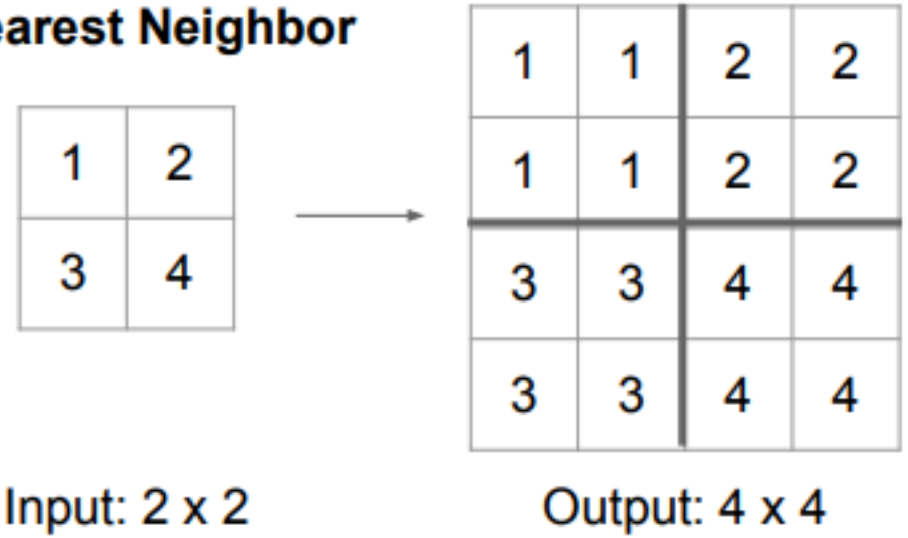
C x H x W

Predictions:
H x W

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

In-Network Upsampling: Unpooling

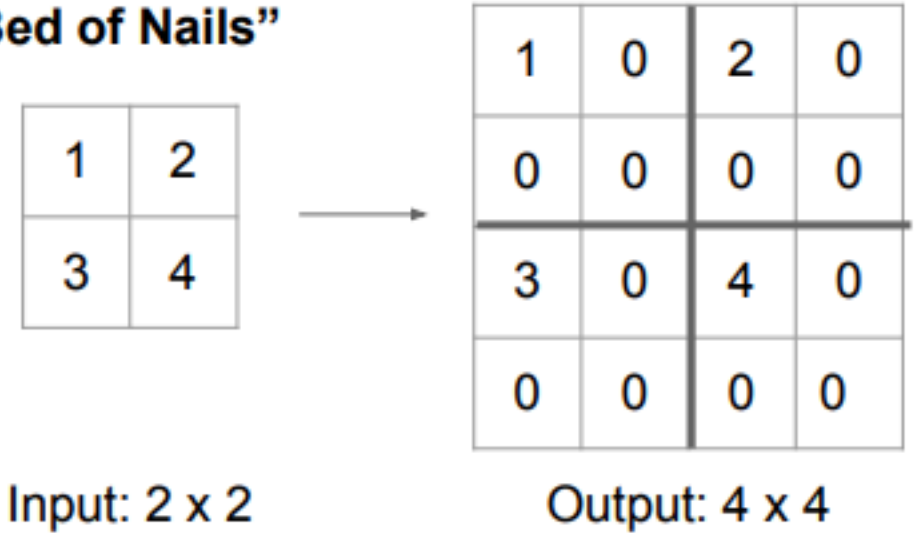
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

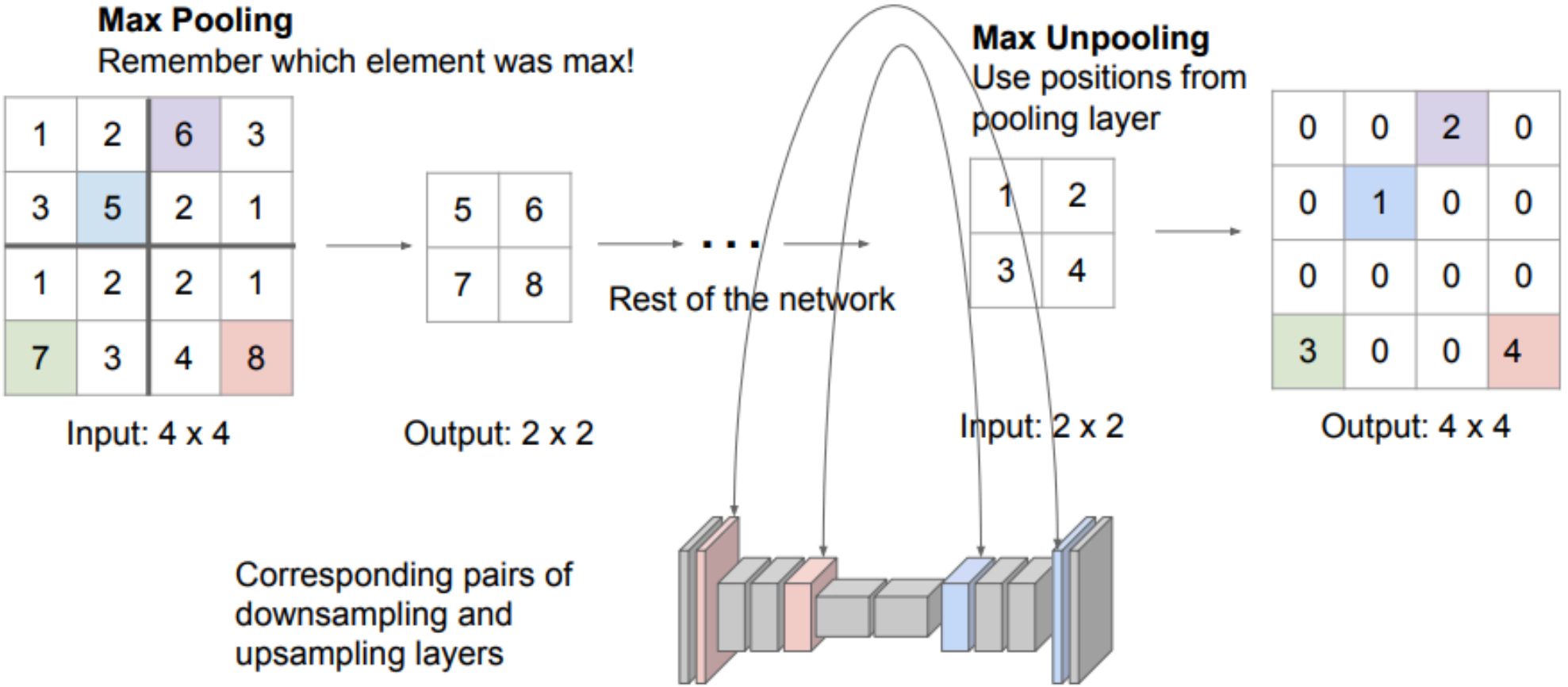
"Bed of Nails"



Input: 2 x 2

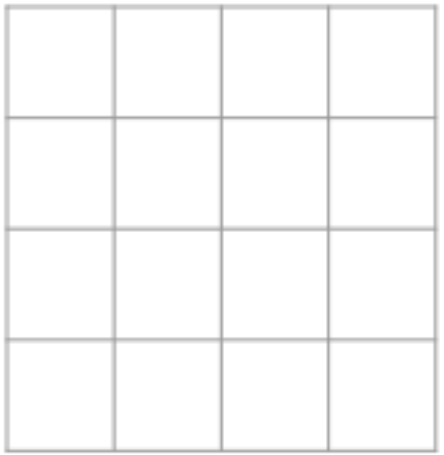
Output: 4 x 4

In-Network Upsampling: Max Unpooling

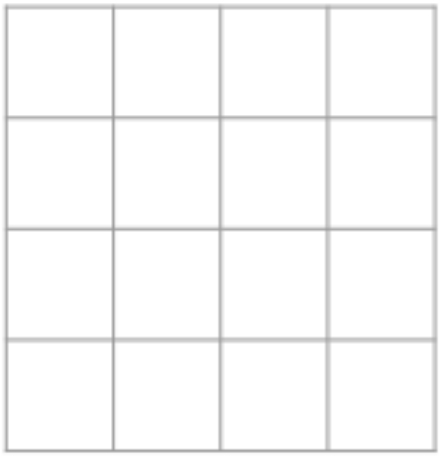


Learnable Upsampling

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



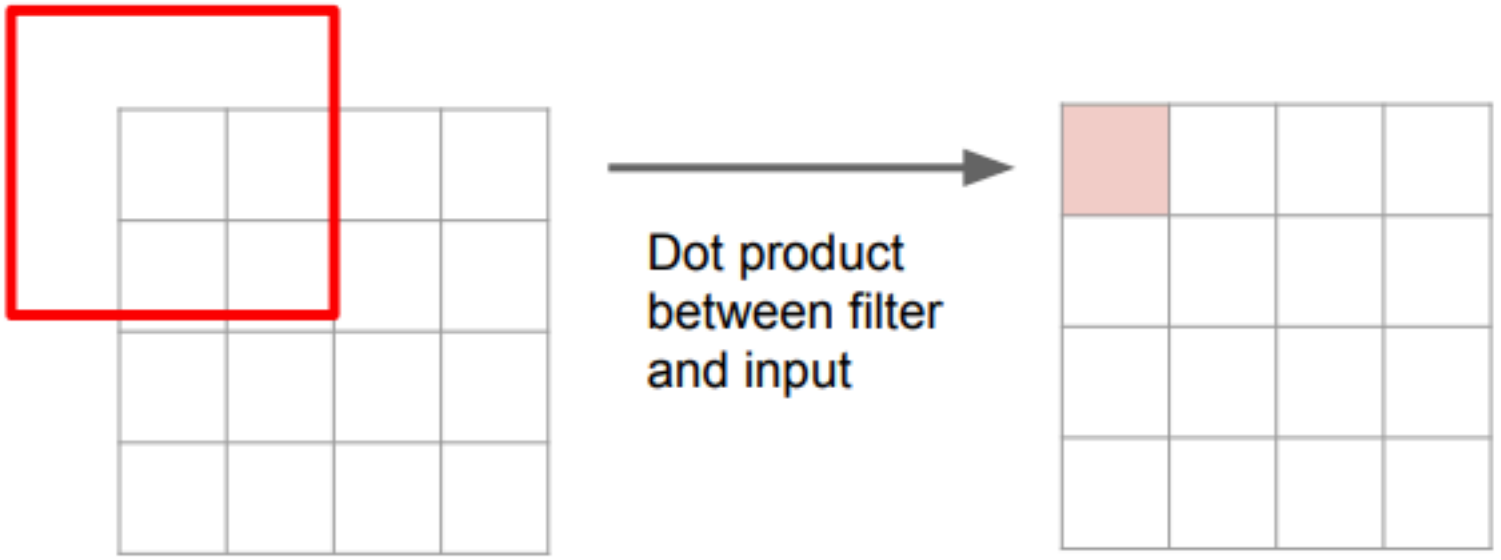
Input: 4 x 4



Output: 4 x 4

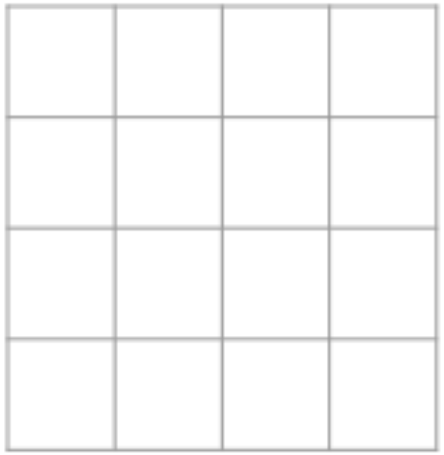
Learnable Upsampling

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

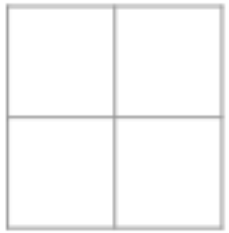


Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



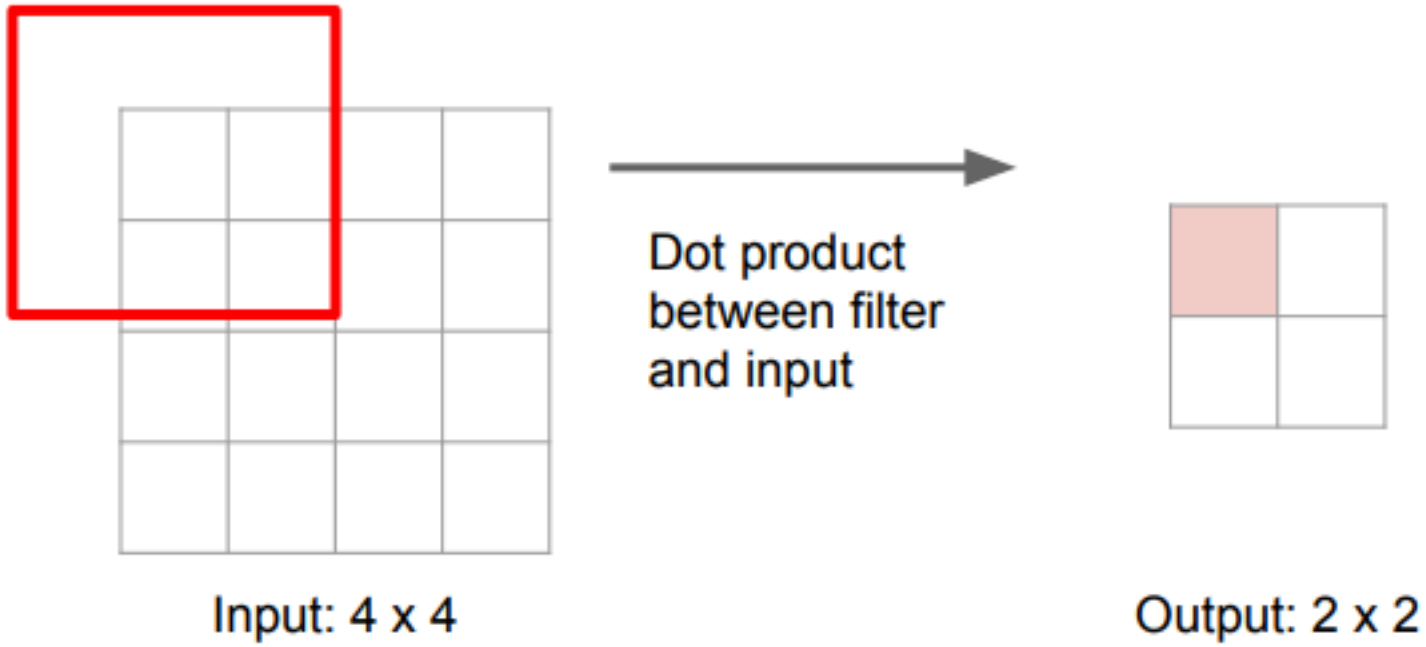
Input: 4 x 4



Output: 2 x 2

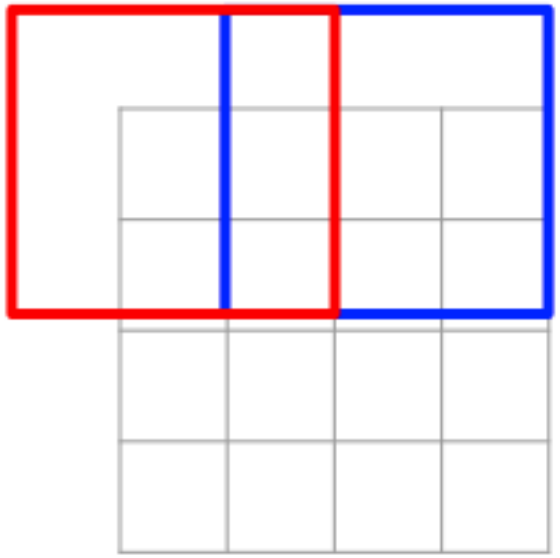
Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input

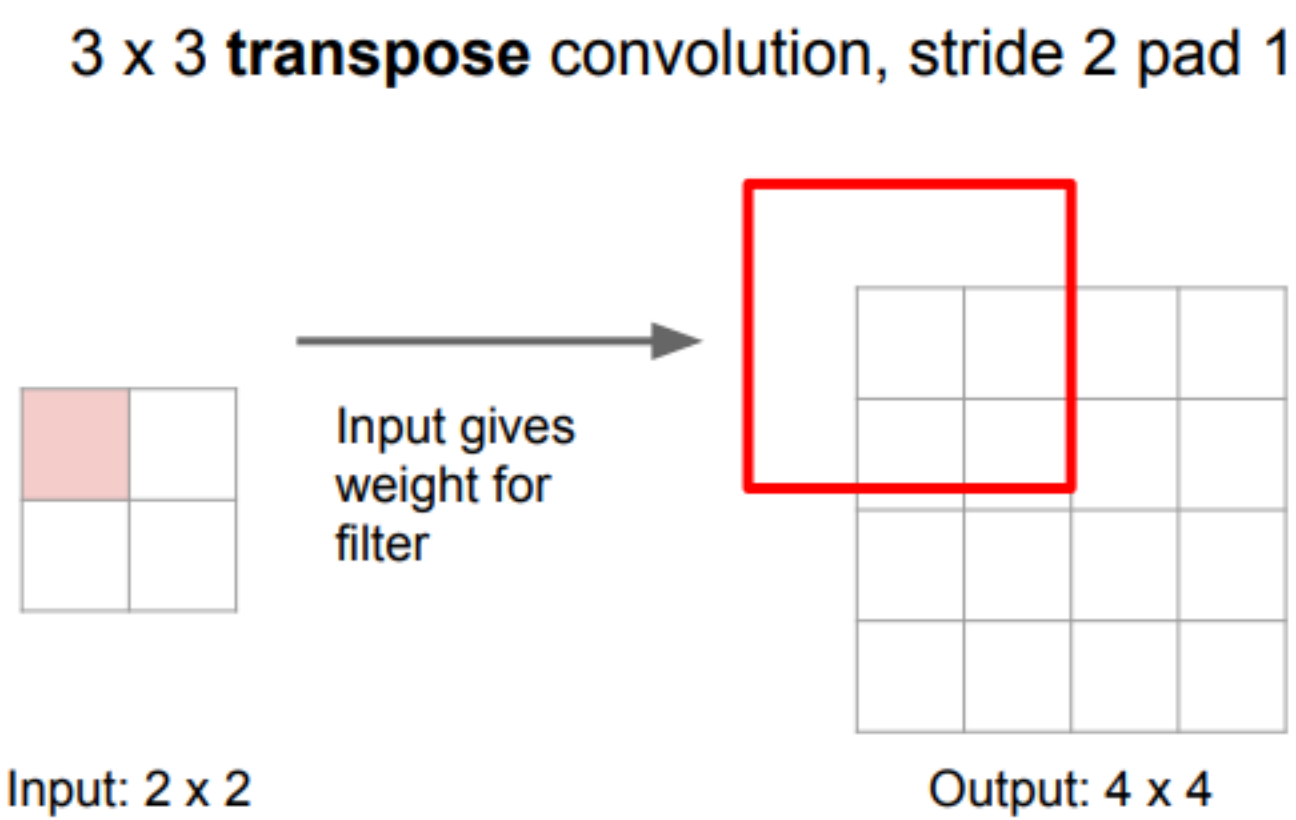
Output: 2 x 2

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

Stride gives ratio between movement in input and output

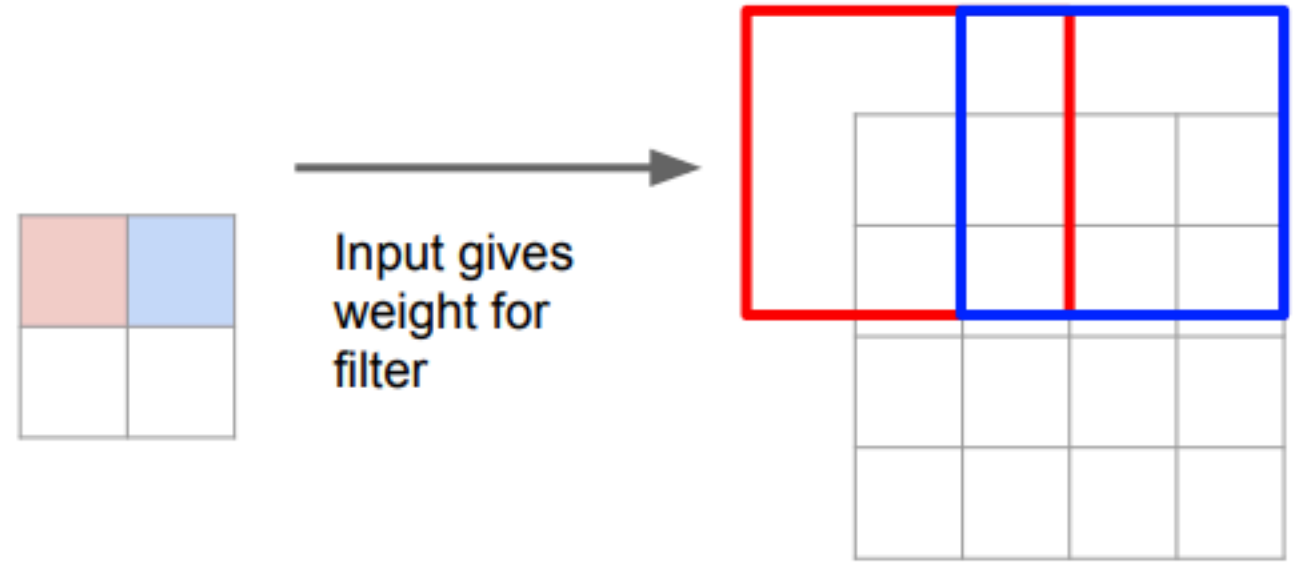
We can interpret strided convolution as “learnable downsampling”.

Learnable Upsampling: Transposed Convolution



Learnable Upsampling: Transposed Convolution

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



Input gives weight for filter

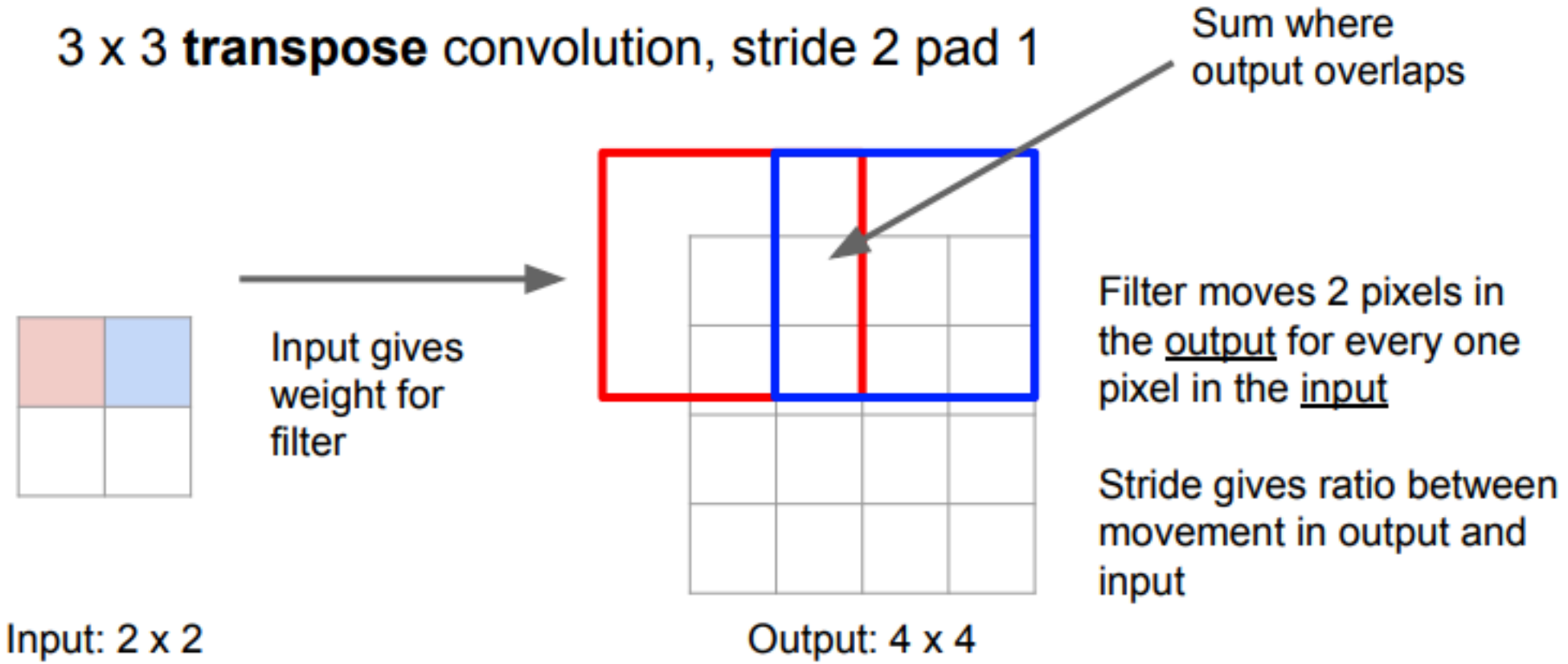
Input: 2 x 2

Output: 4 x 4

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

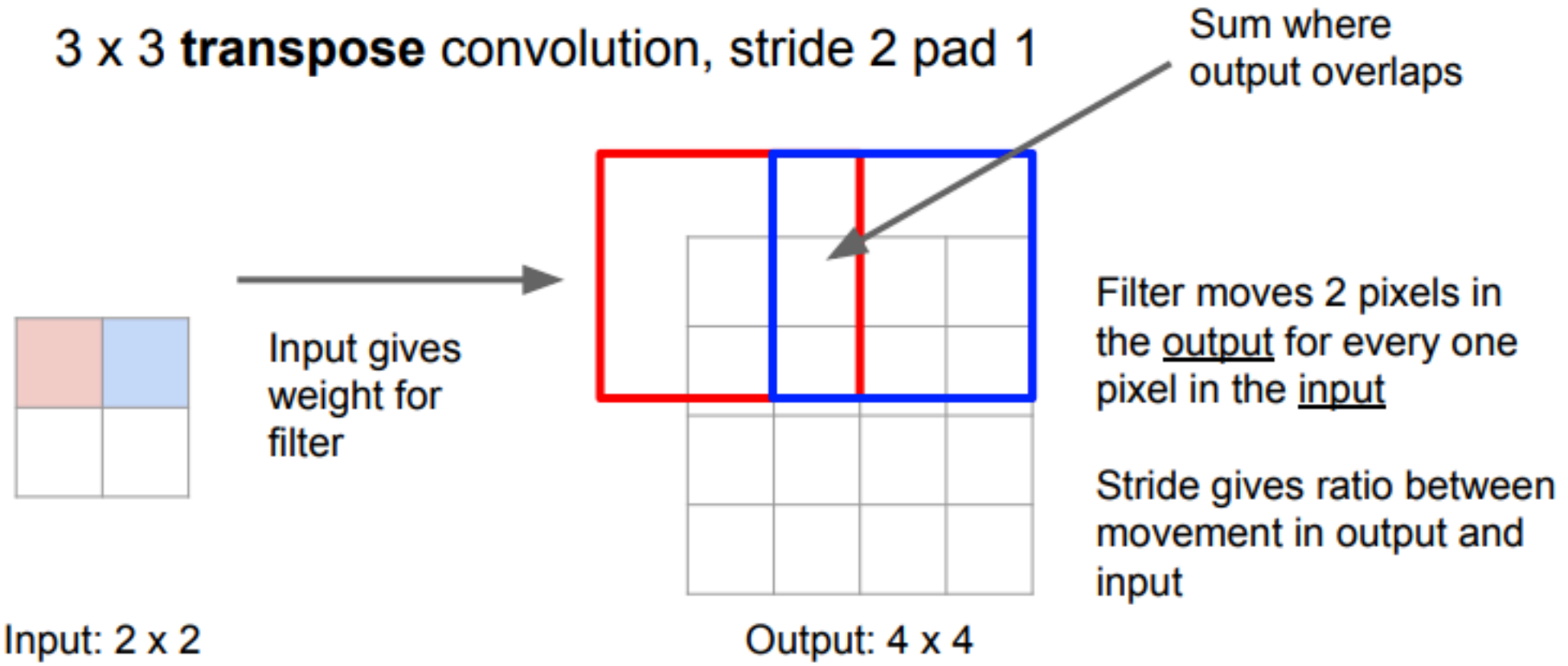
Stride gives ratio between movement in output and input

Learnable Upsampling: Transposed Convolution

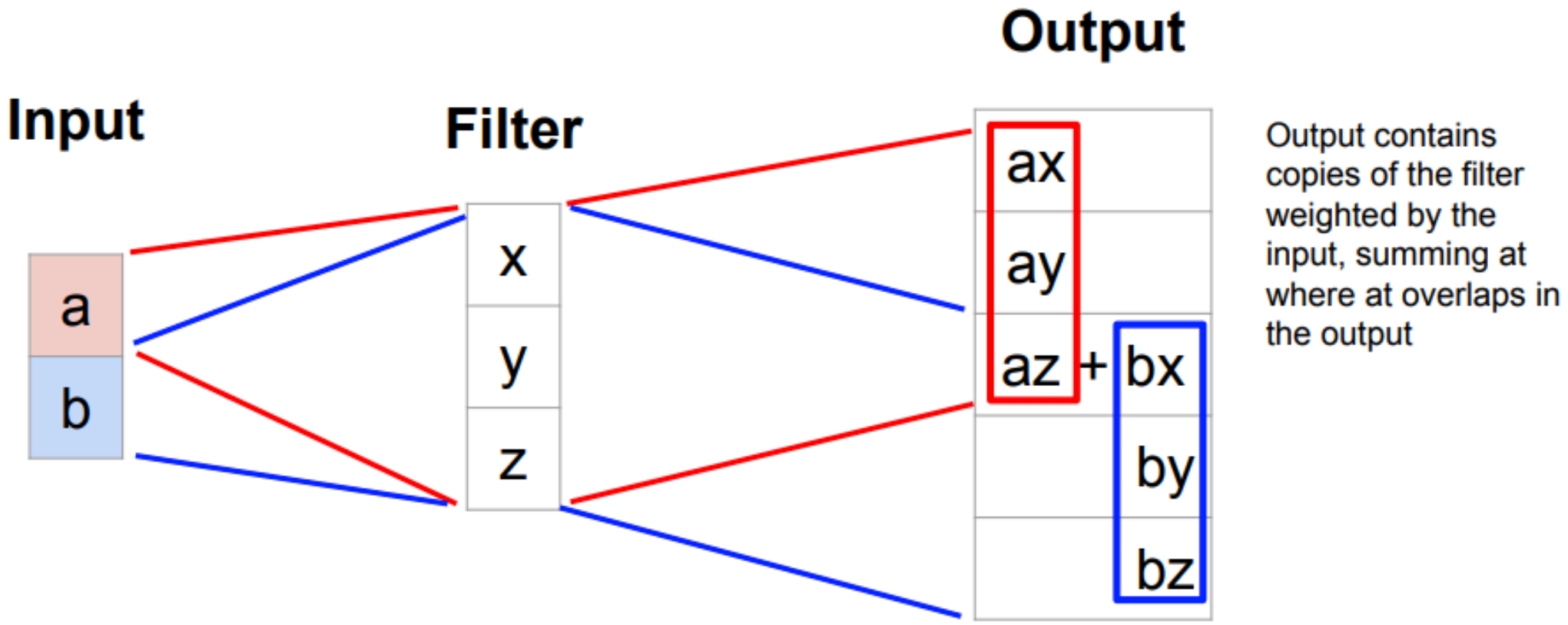


Learnable Upsampling: Transposed Convolution

Q: Why is it called transpose convolution?



Learnable Upsampling: Transposed Convolution



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

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

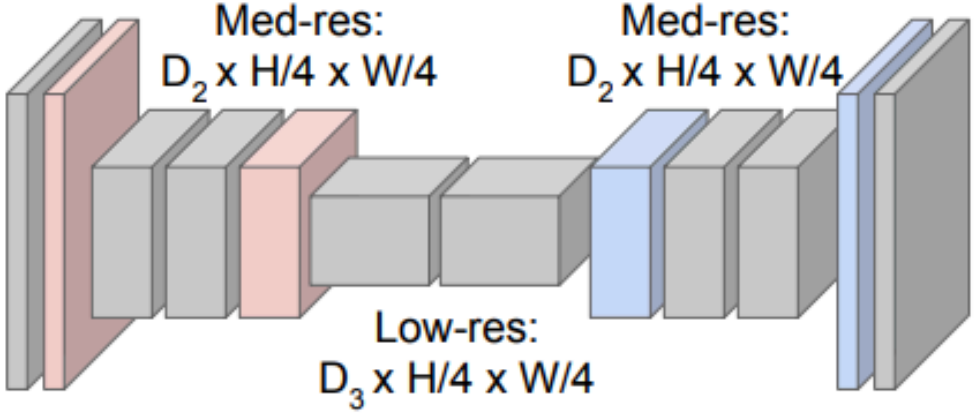
Semantic Segmentation: Fully Convolutional

Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



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

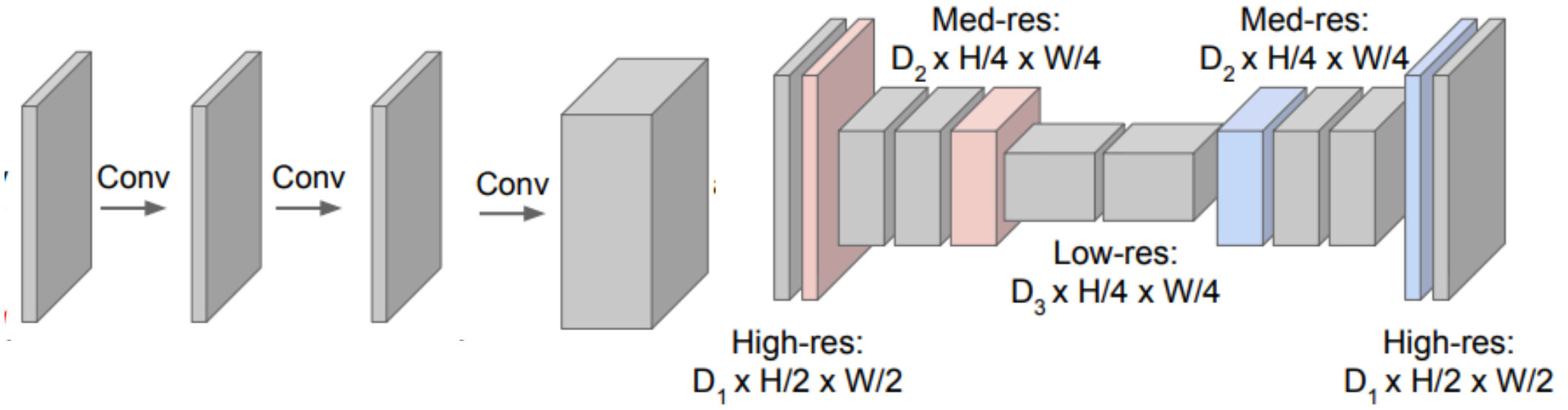
High-res:
D_1 x H/2 x W/2

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

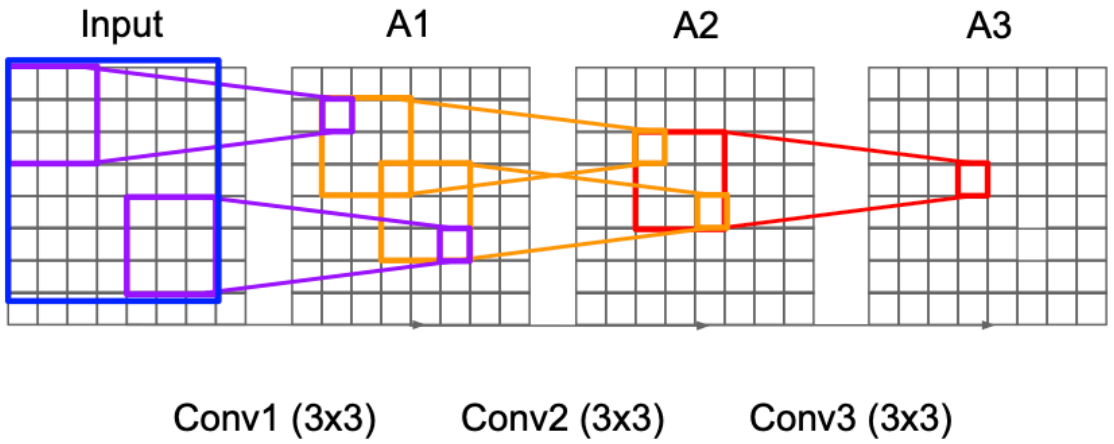
Advantage of Bottleneck

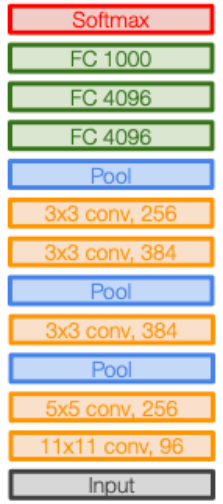
- Lower memory cost



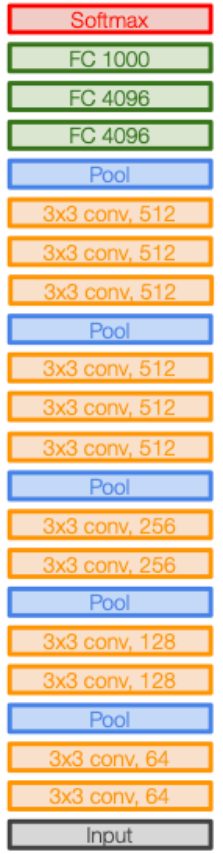
Recap of Receptive Field

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

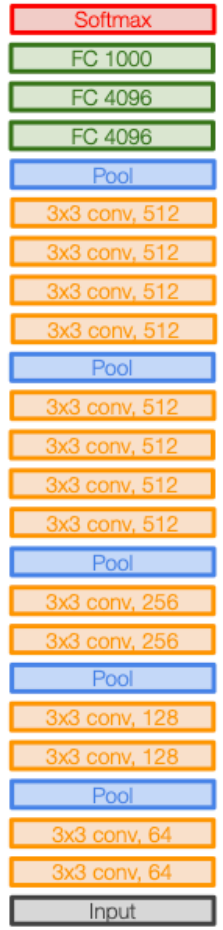




AlexNet



VGG16

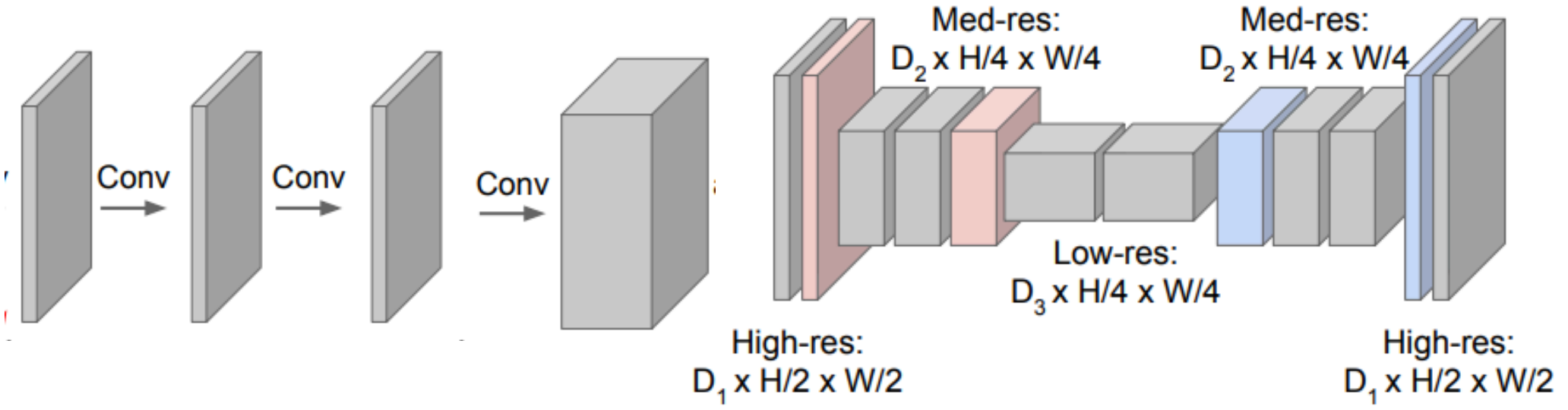


VGG19

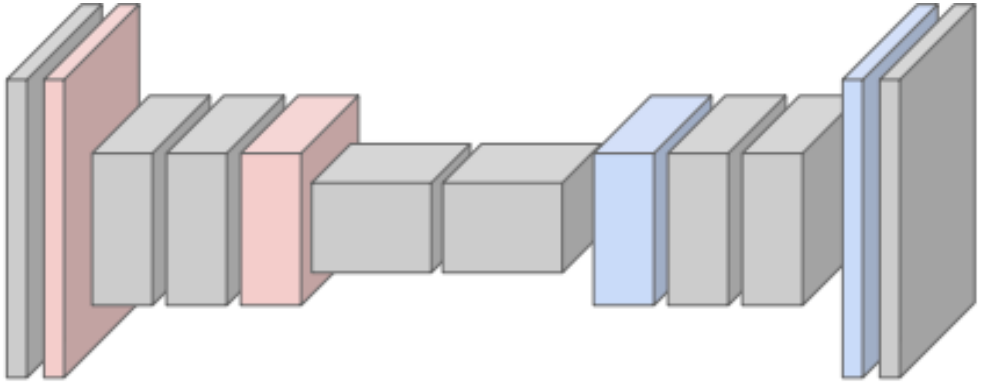
Advantage of Bottleneck

- 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

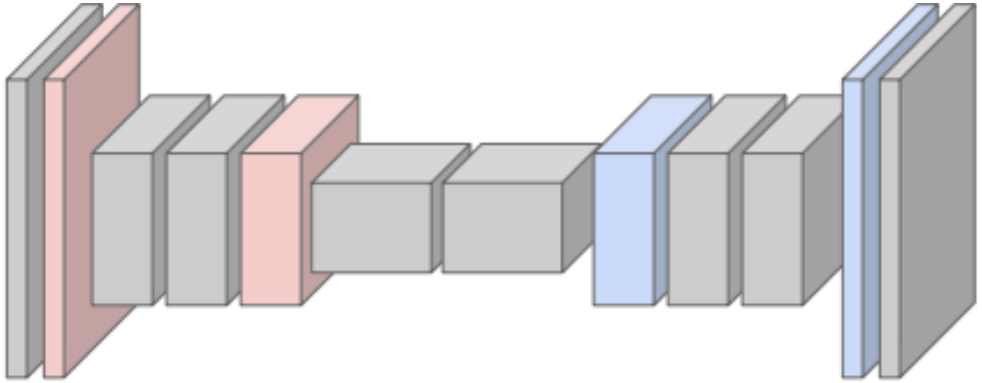


Improving FCN



What needs to be stored in the bottleneck?

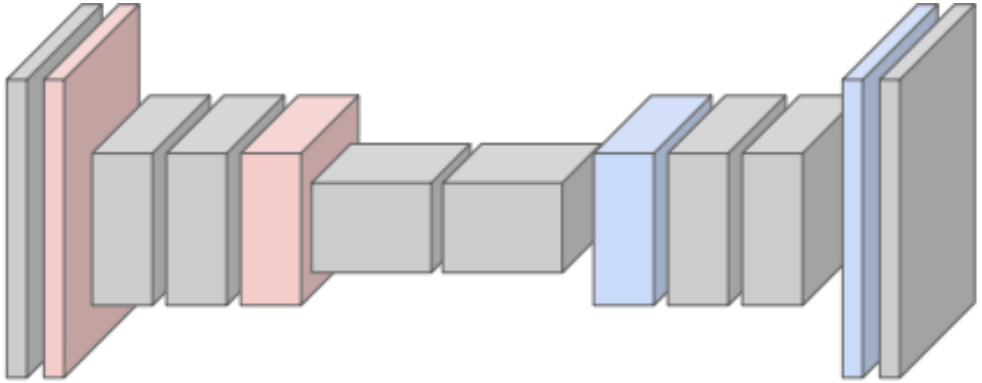
Improving FCN



What needs to be stored in the bottleneck?

- Global context

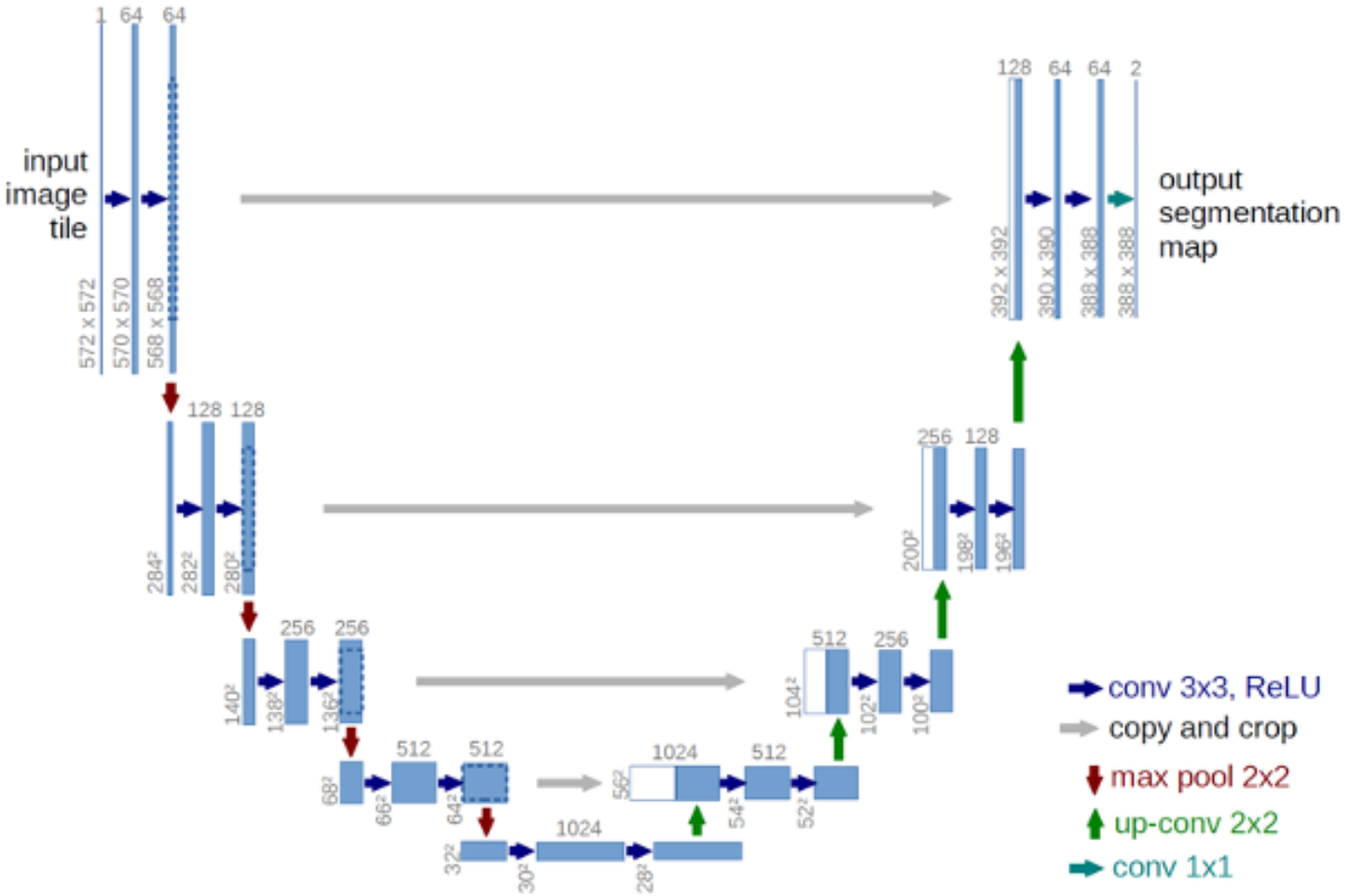
Improving FCN



What needs to be stored in the bottleneck?

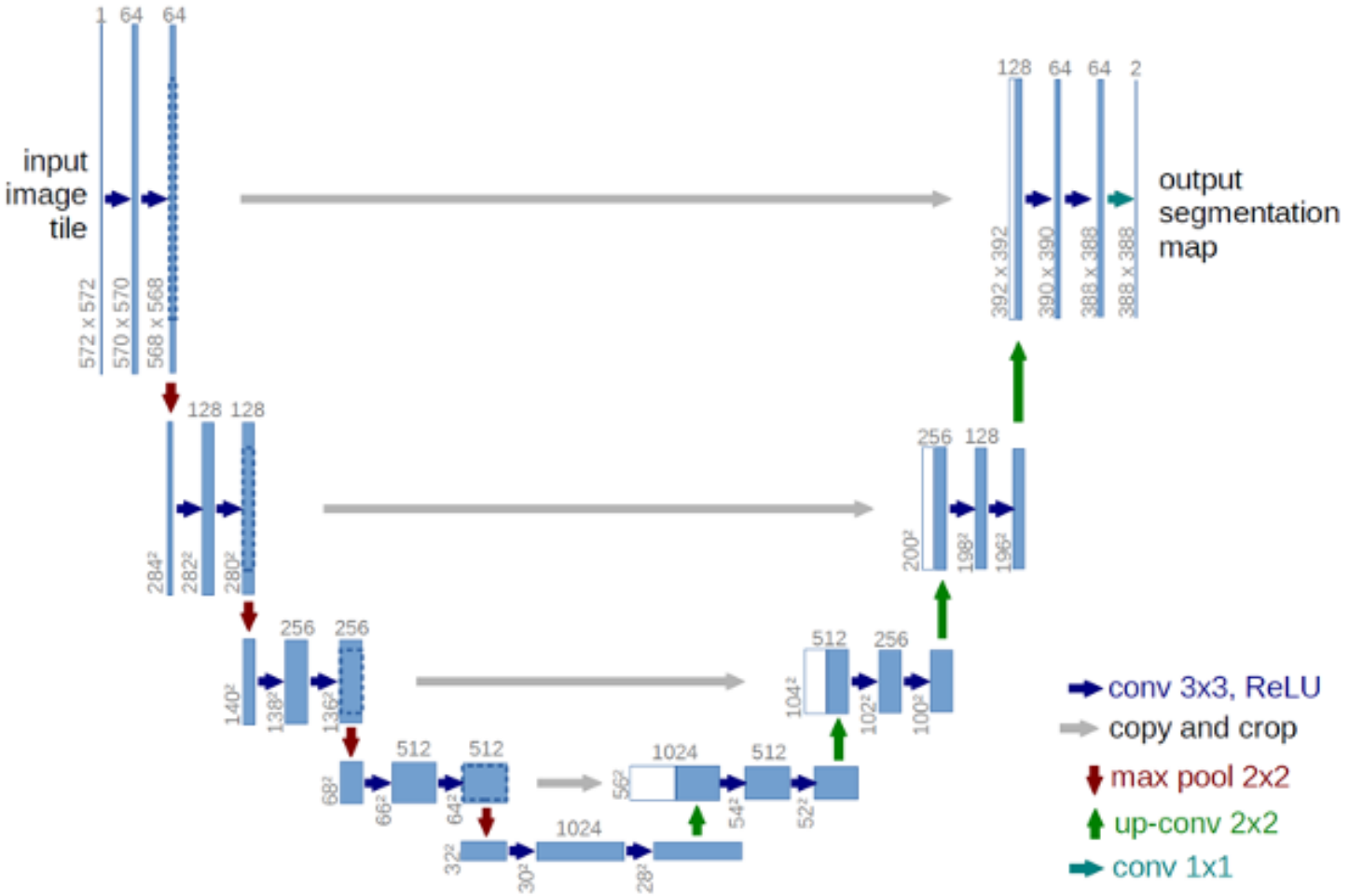
- Global context
- Per-pixel spatial information, especially around the boundary

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

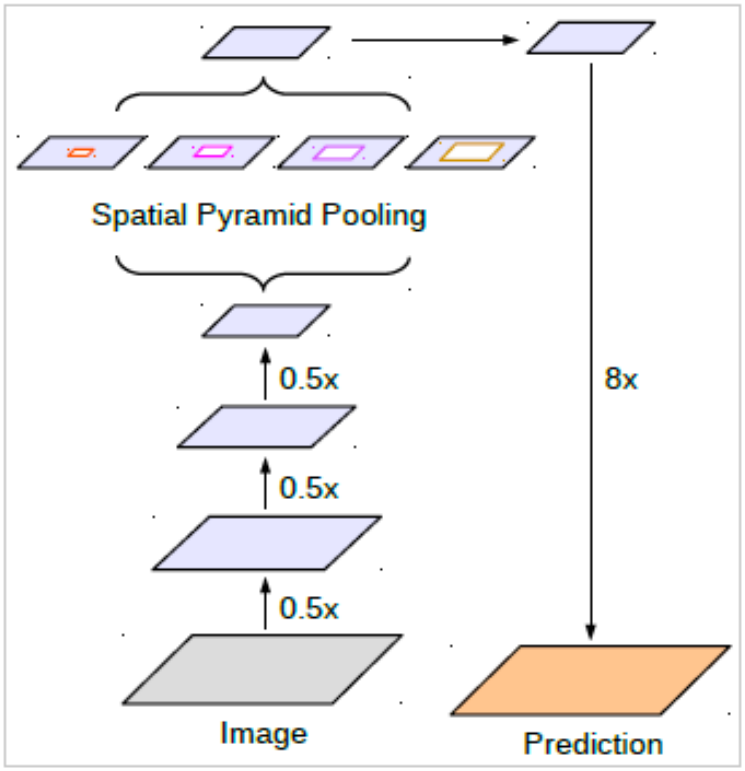
Summary of Semantic Segmentation

- A top-down approach

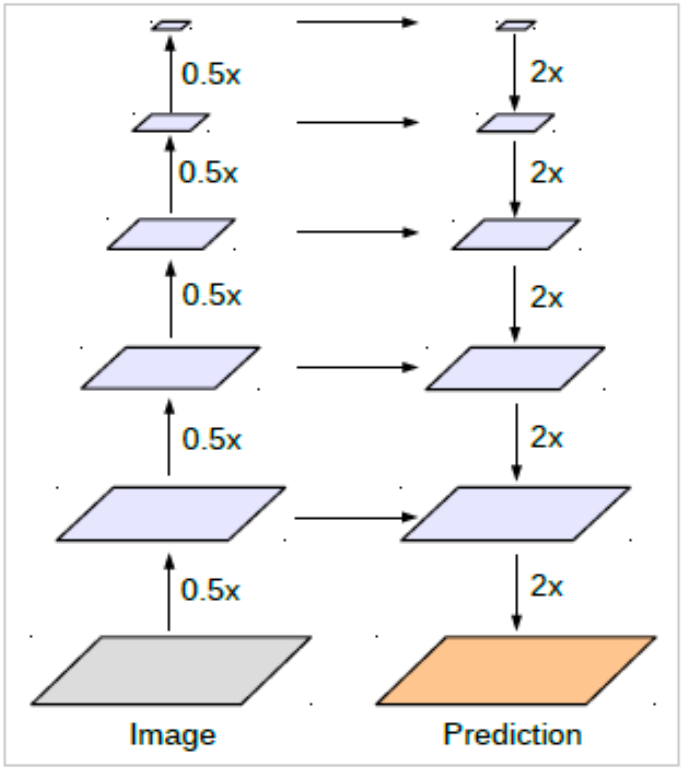
- Bottleneck structure:
  - Large receptive field and provides global context
  - Get rid of redundant information
  - Lower the computation cost

- Skip link:
  - Assist final segmentation
  - Avoid memorization

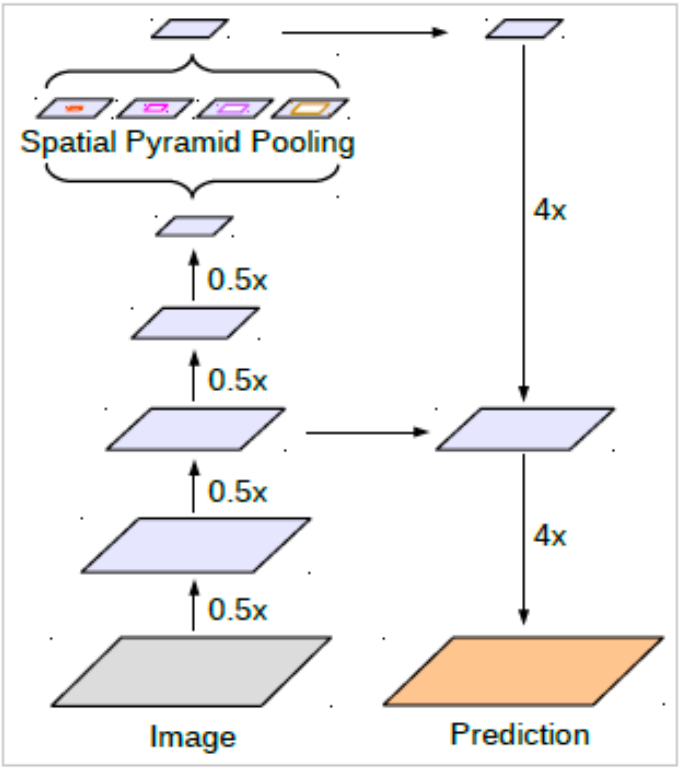
DeepLab V3



(a) Spatial Pyramid Pooling

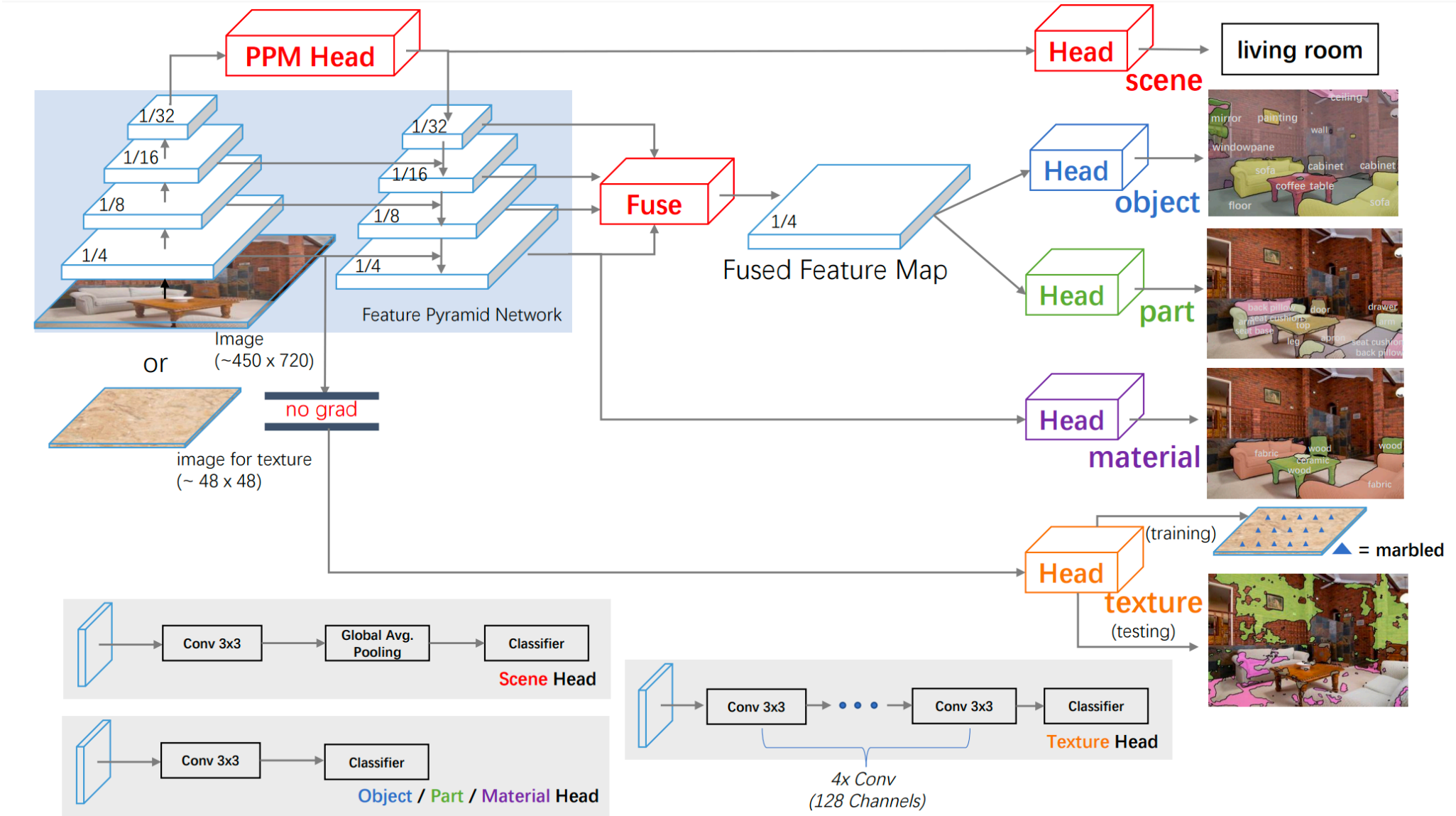


(b) Encoder-Decoder



(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.

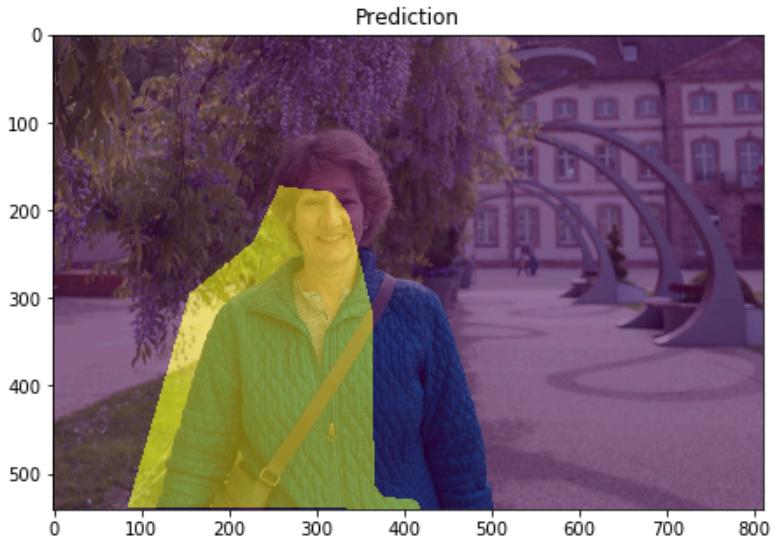
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- 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

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



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:

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

Rahman, Md Atiqur, and Yang Wang. "Optimizing intersection-over-union in deep neural networks for image segmentation." International symposium on visual computing. Springer, Cham, 2016

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.

Next week: Lecture 8,
3D Vision I