Introduction to Computer Vision

Lecture 8 - Deep Learning V

Prof. He Wang

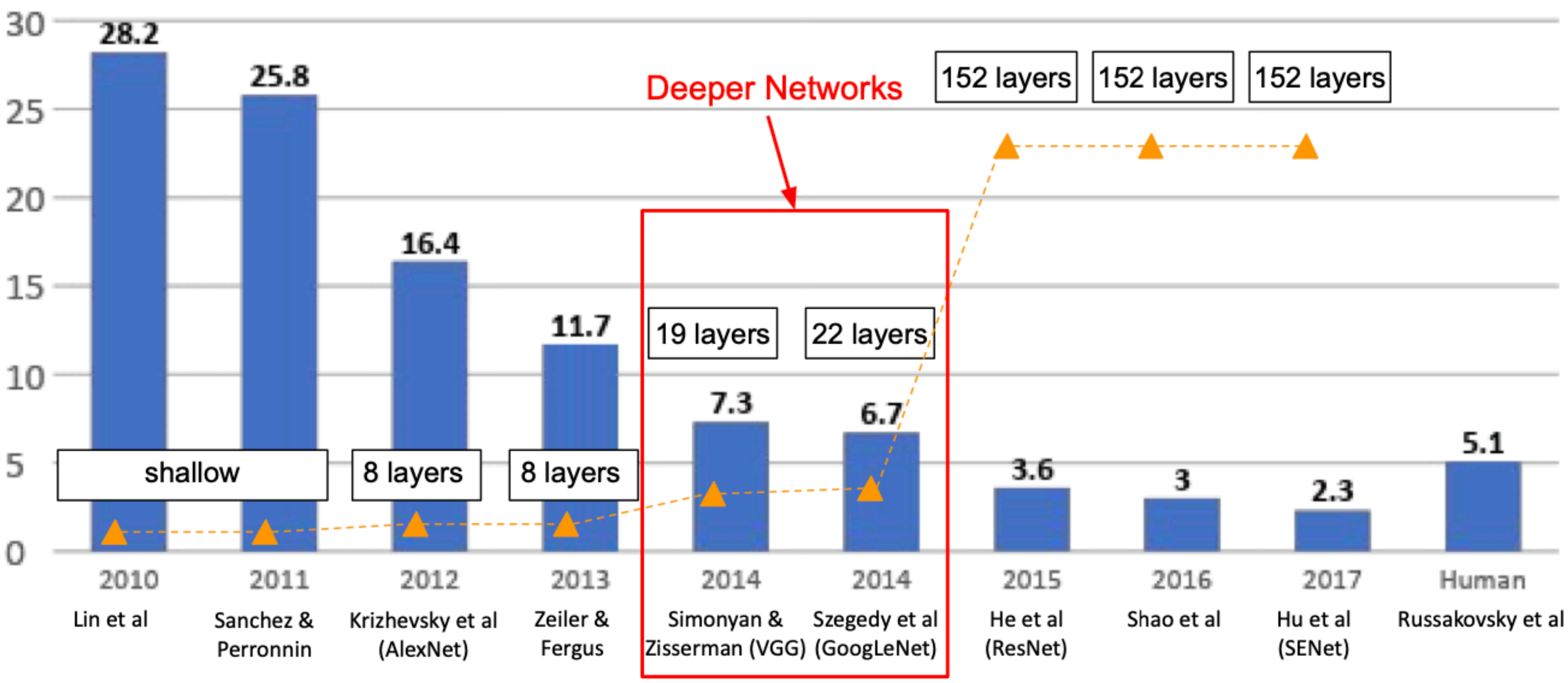
Logistics

- Assignment 2: to release on 4/11 (this Friday evening), due on 4/26 11:59PM (Saturday)

- Some functions are required to be implemented without for loop.

- 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.

The History: ImageNet Challenge Winners



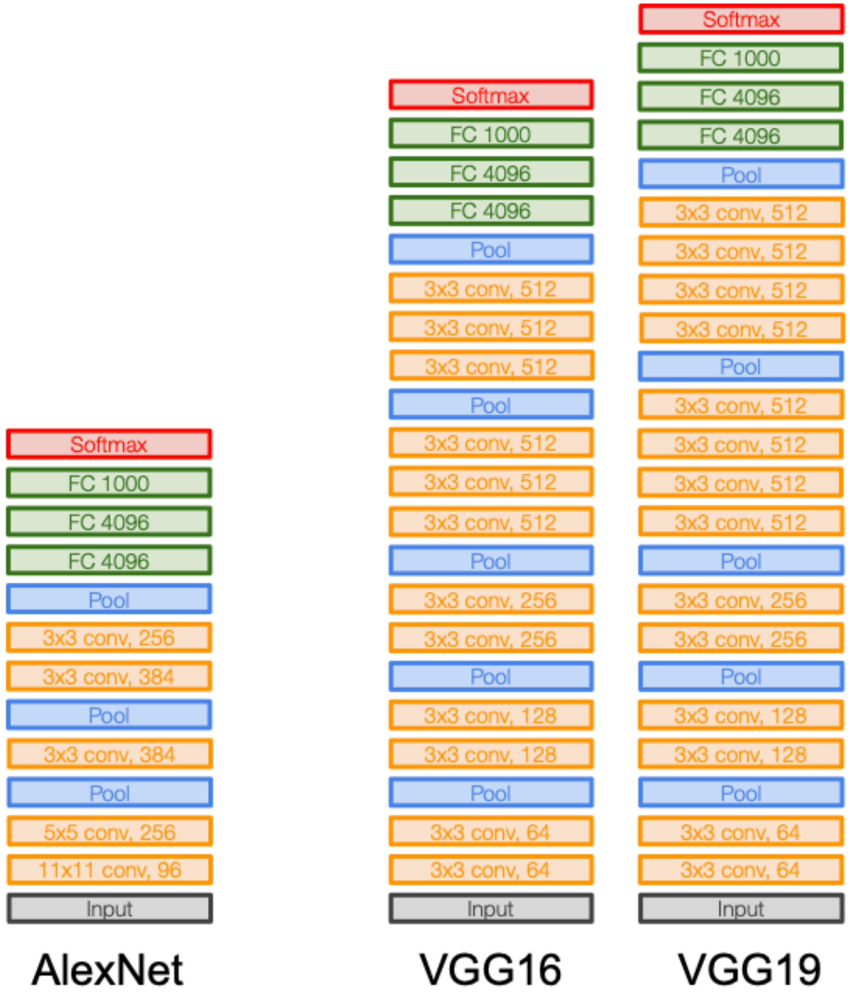
VGGNet

Small filters, Deeper networks

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

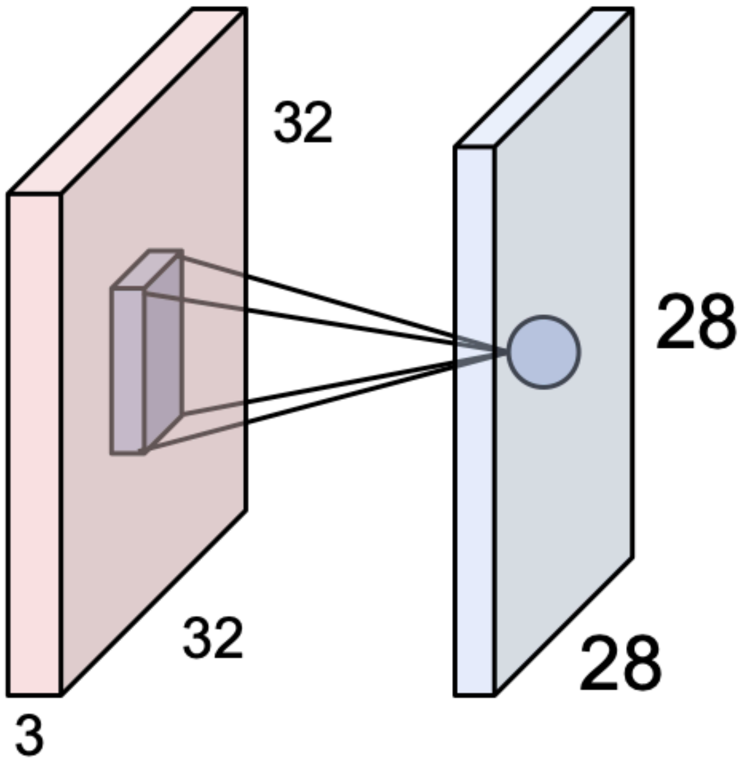
Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

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

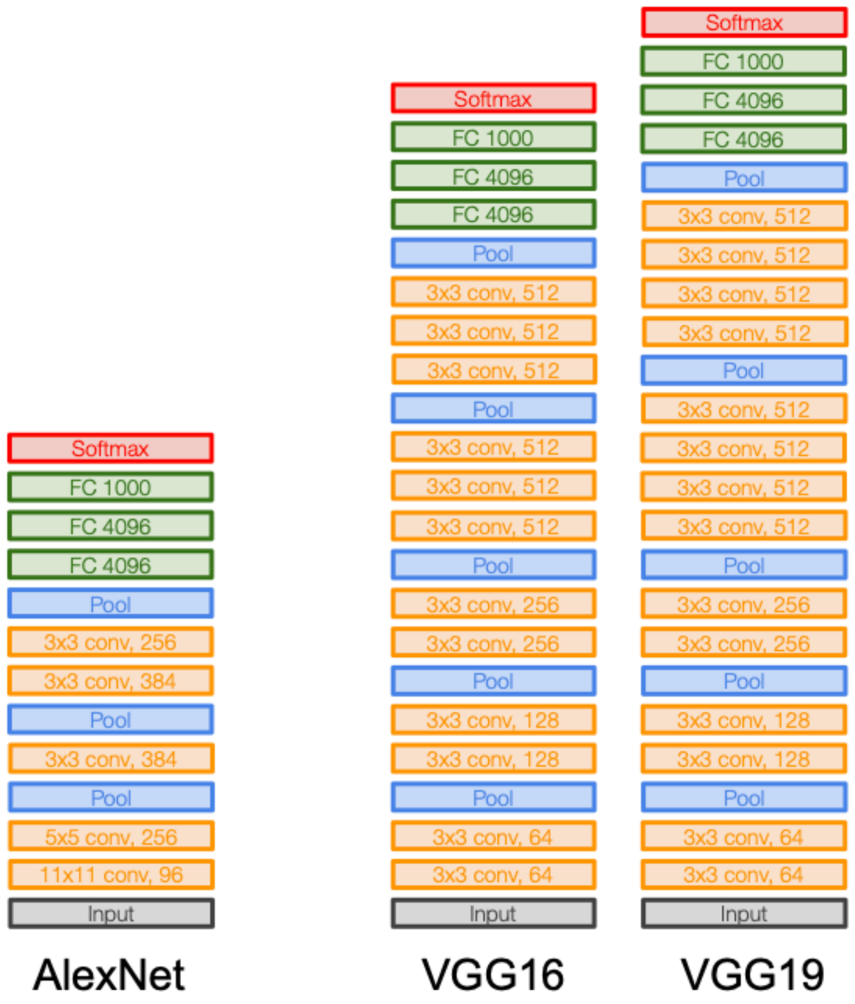
"5x5 filter" -> "5x5 receptive field for each neuron"

VGGNet

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

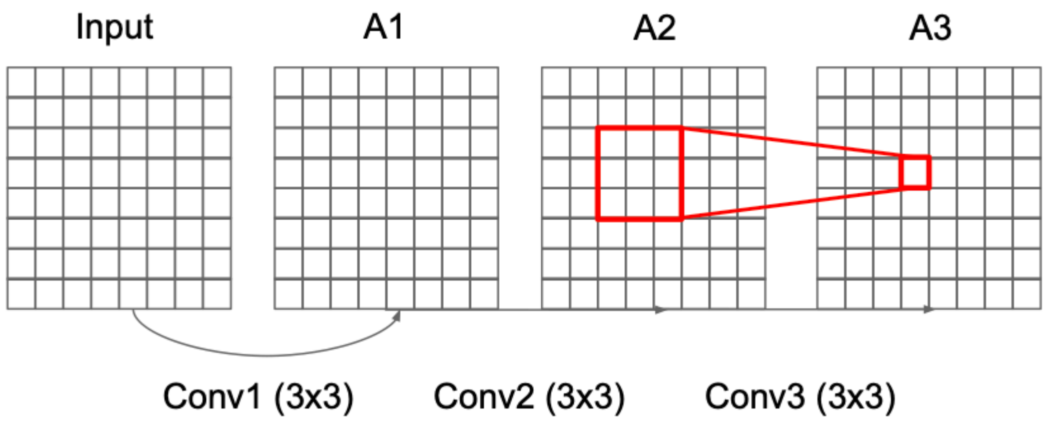
Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

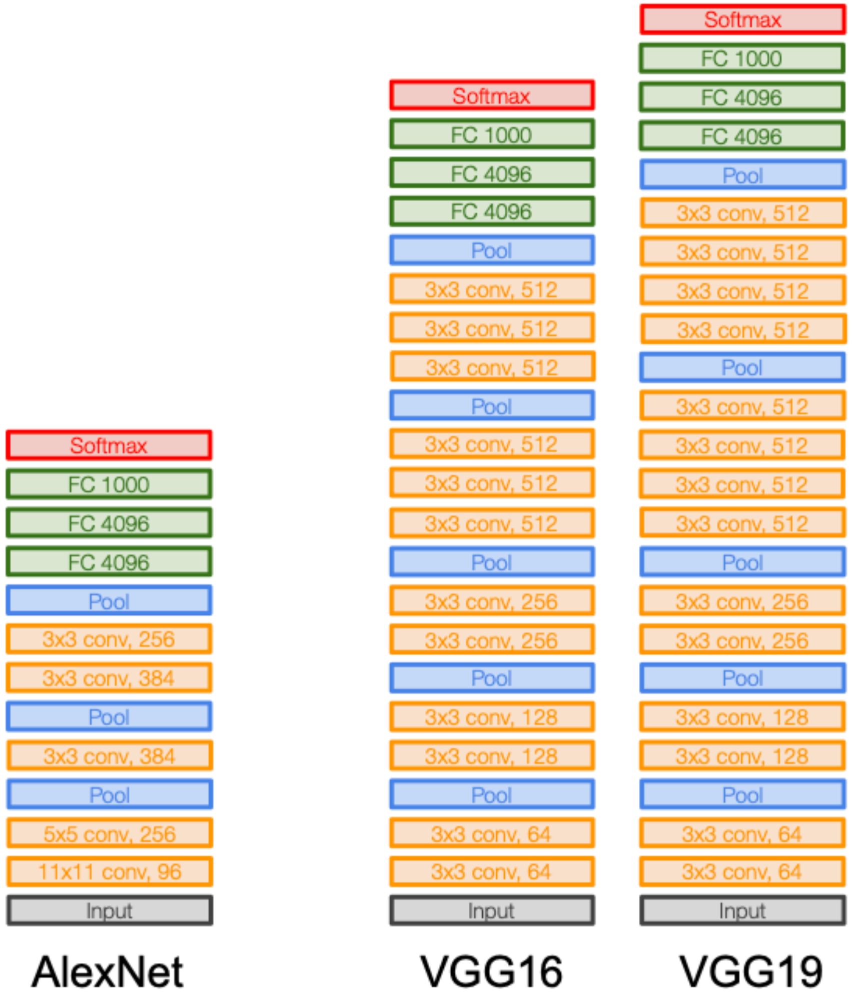
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



Receptive Field

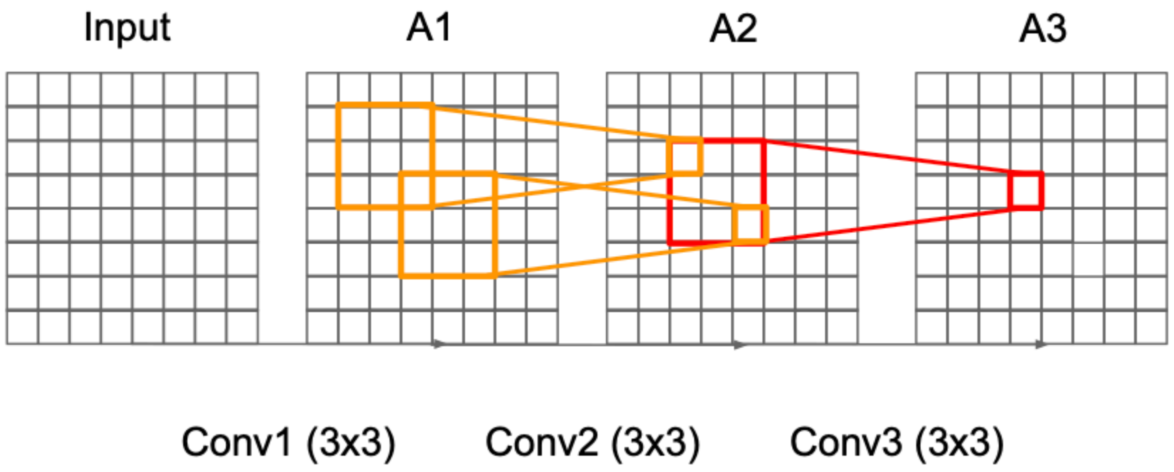
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

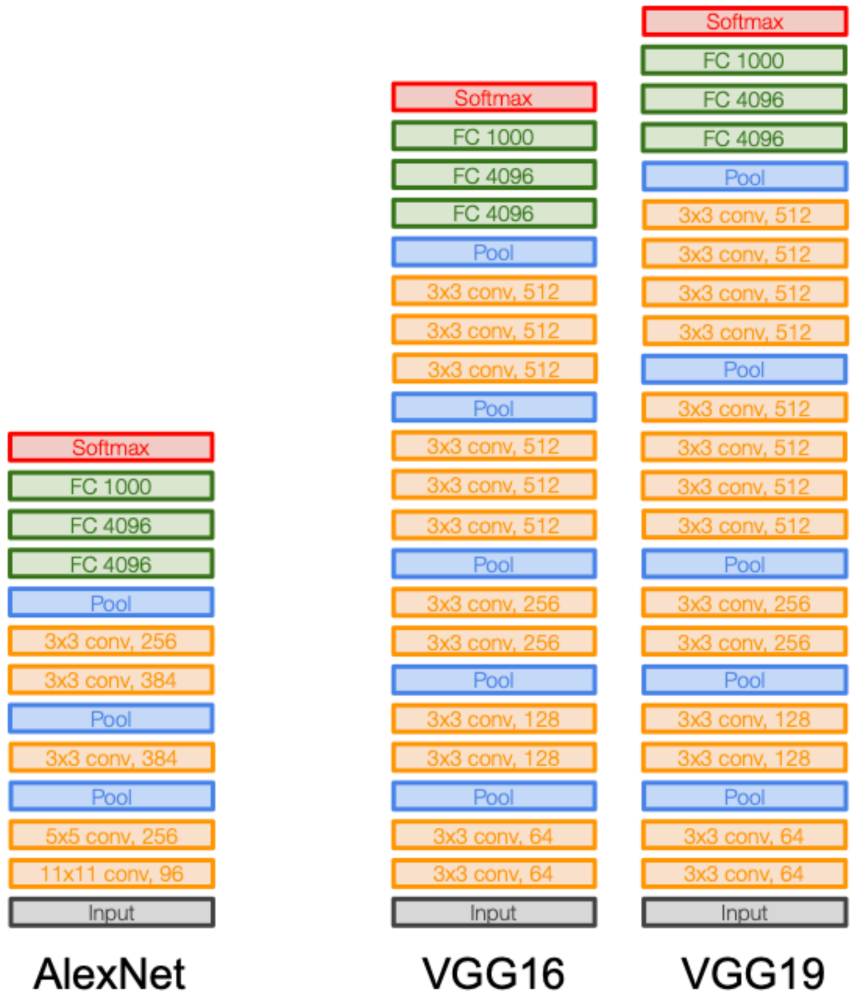




Receptive Field

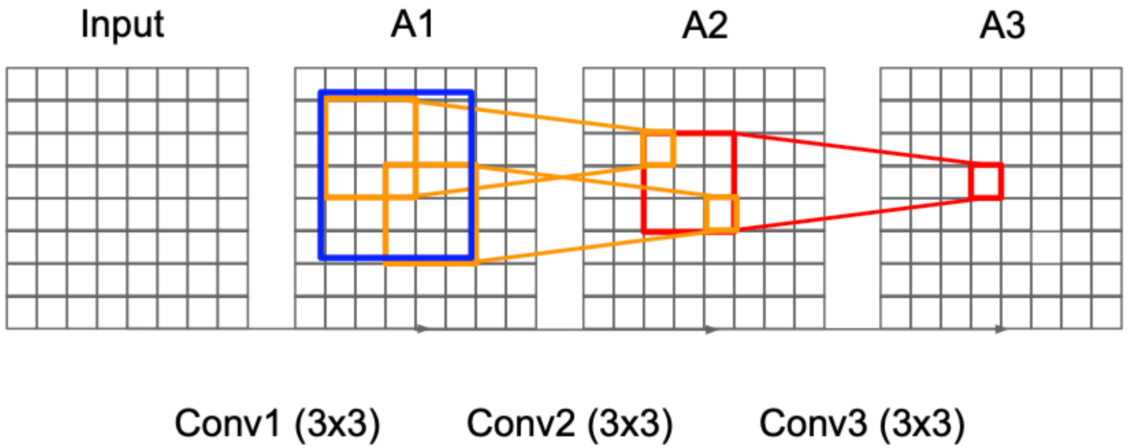
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

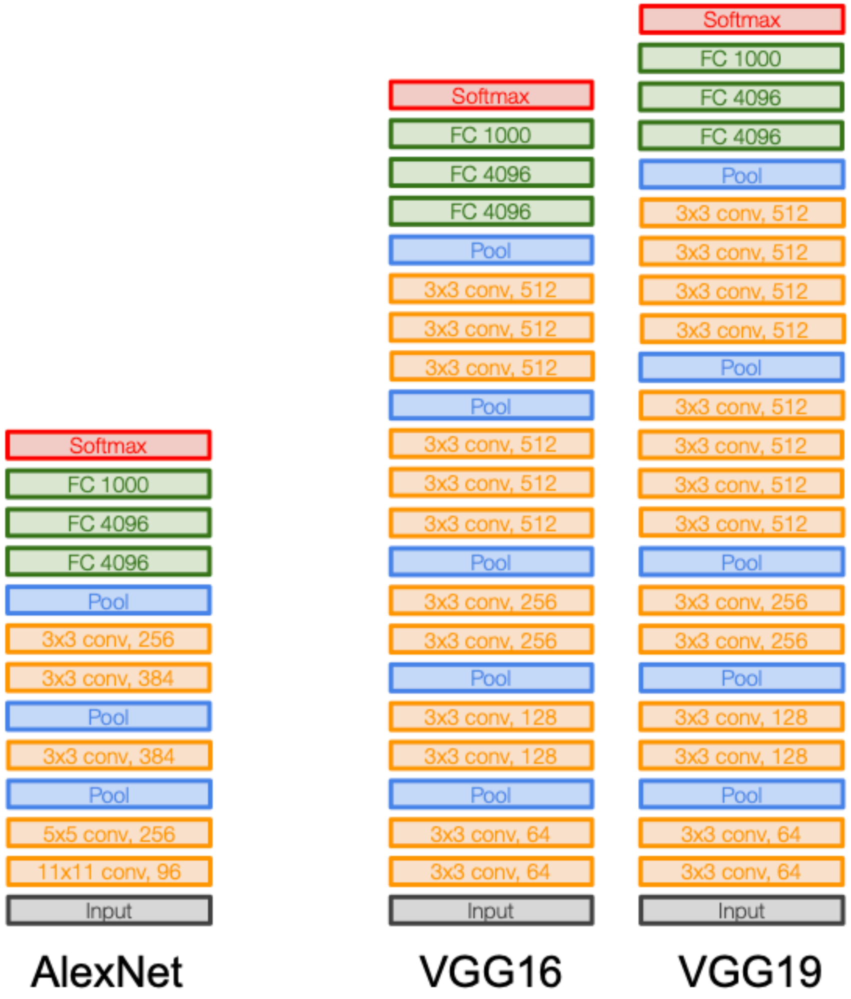




Receptive Field

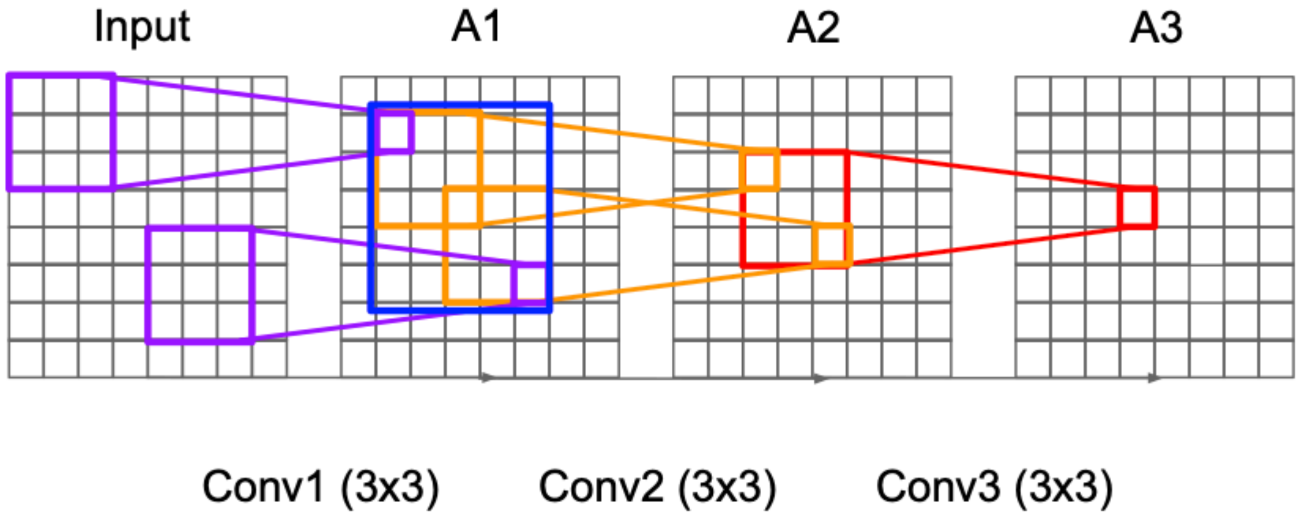
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

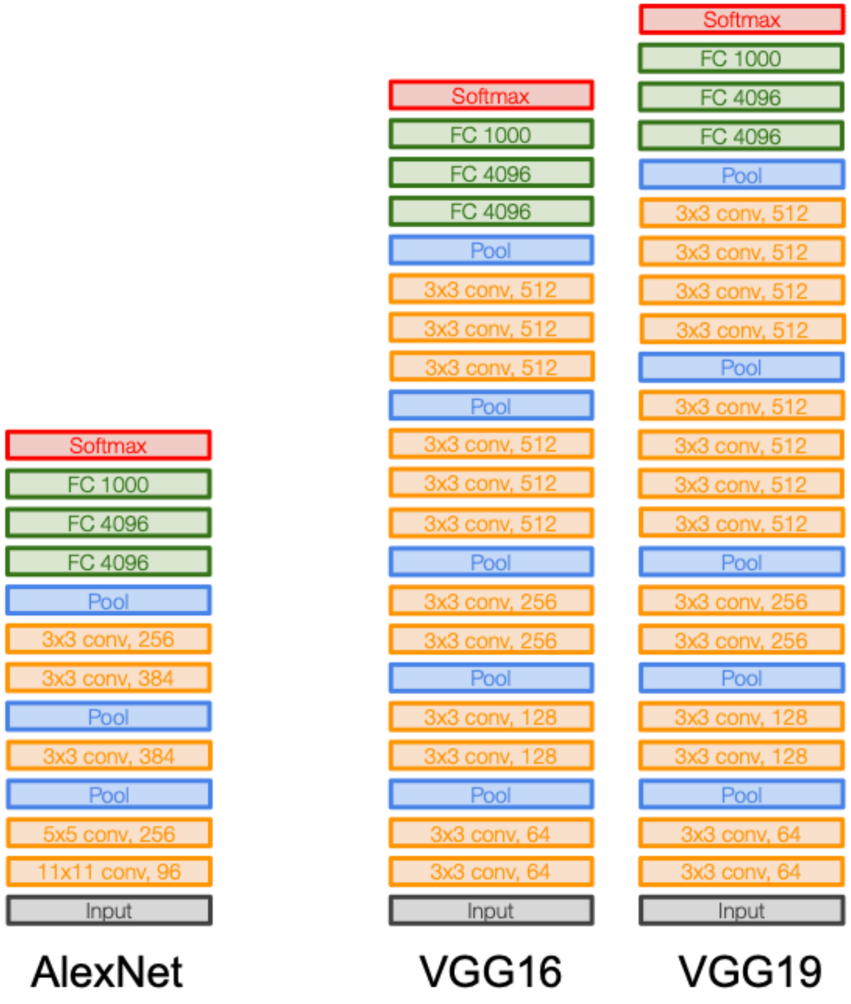




Receptive Field

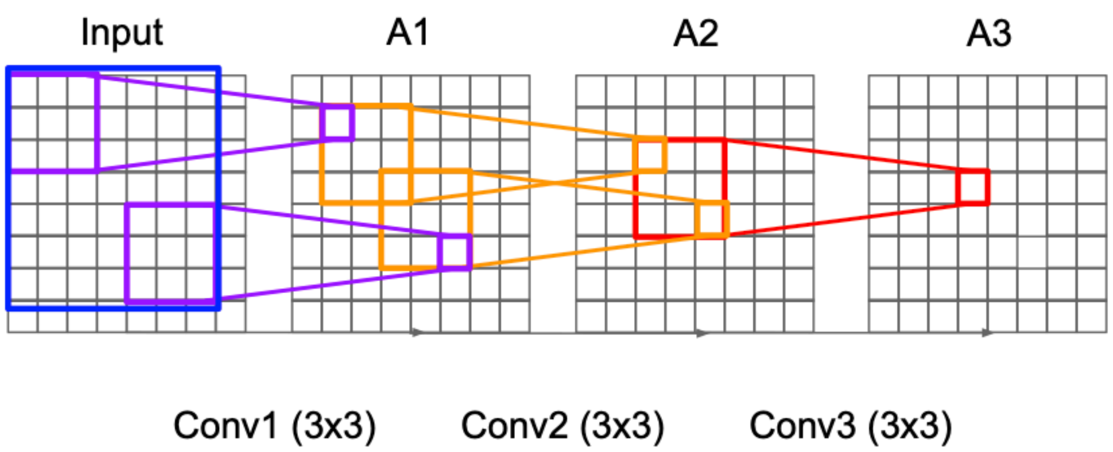
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

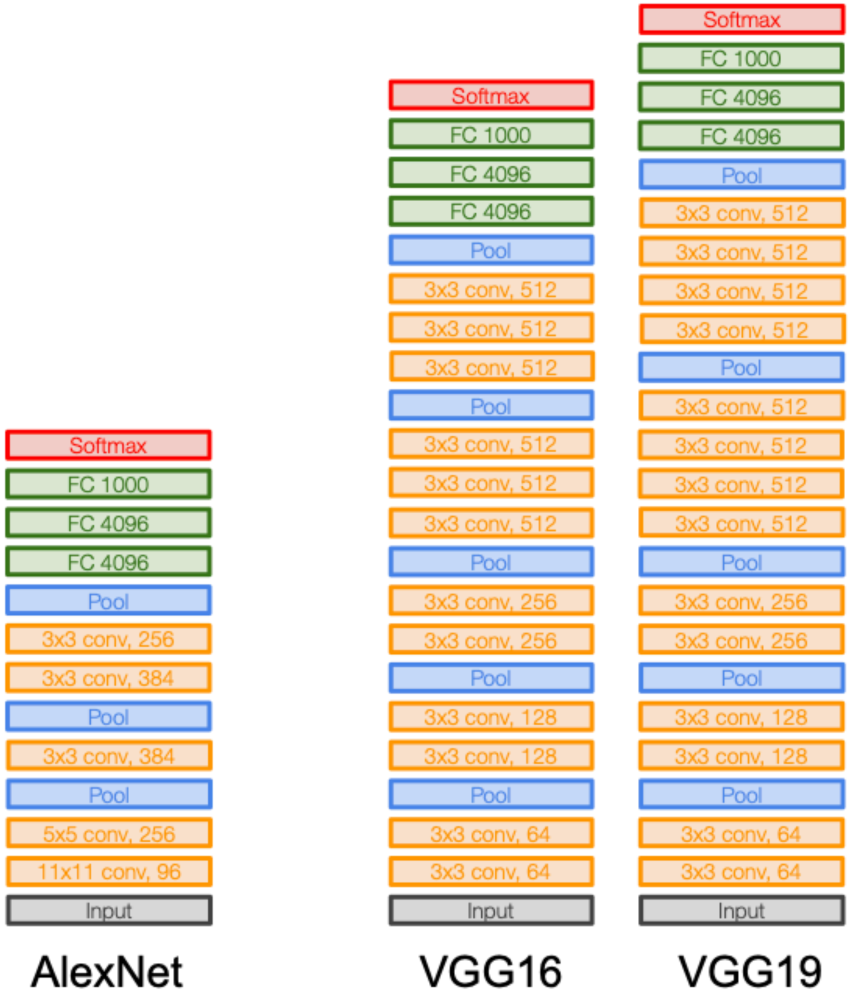




Receptive Field

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



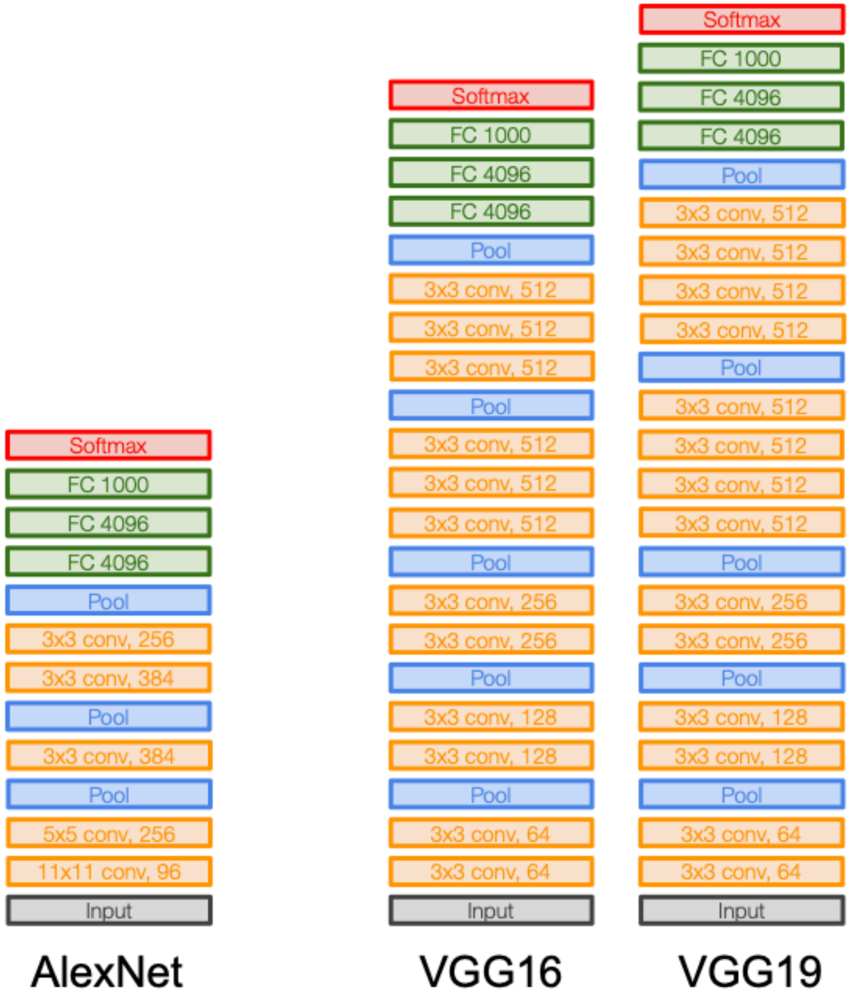


Receptive Field

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]



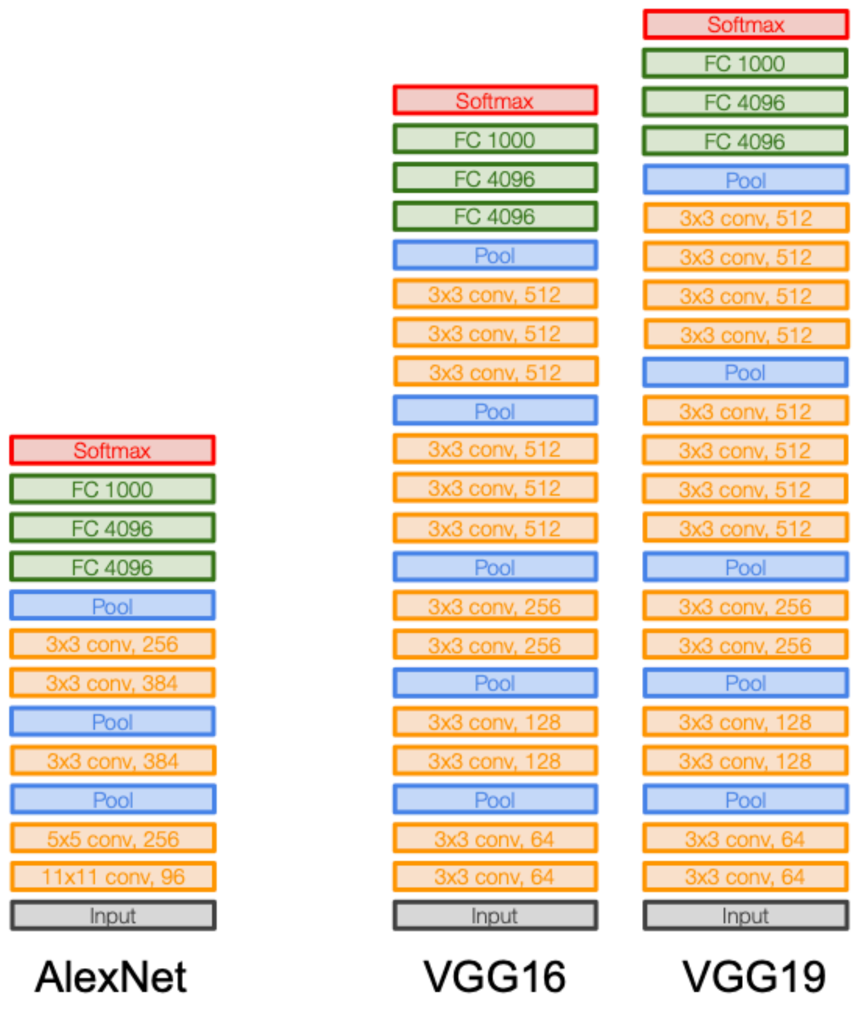
Receptive Field

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

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

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

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

POOL2: [112x112x64] memory: 112*112*64=800K params: 0

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

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

POOL2: [56x56x128] memory: 56*56*128=400K params: 0

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

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

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

POOL2: [28x28x256] memory: 28*28*256=200K params: 0

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

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

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

POOL2: [14x14x512] memory: 14*14*512=100K params: 0

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

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

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

POOL2: [7x7x512] memory: 7*7*512=25K params: 0

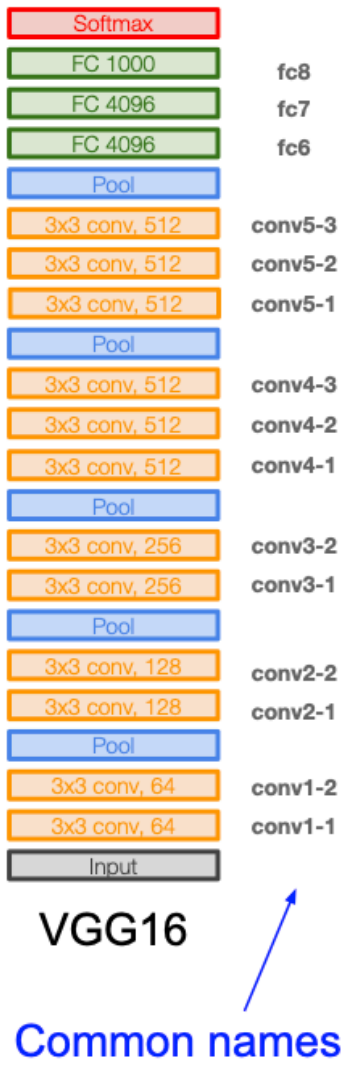
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448

FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216

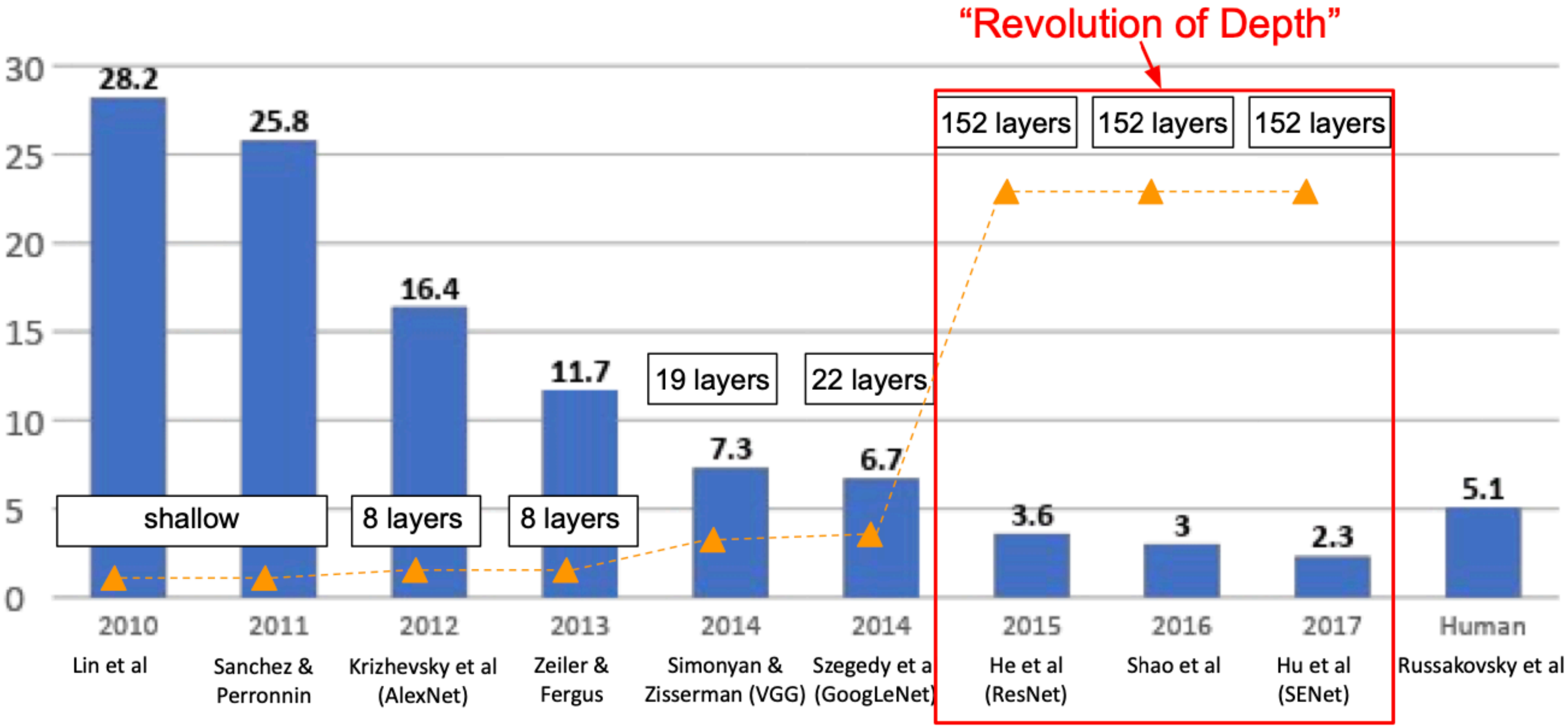
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

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

TOTAL params: 138M parameters



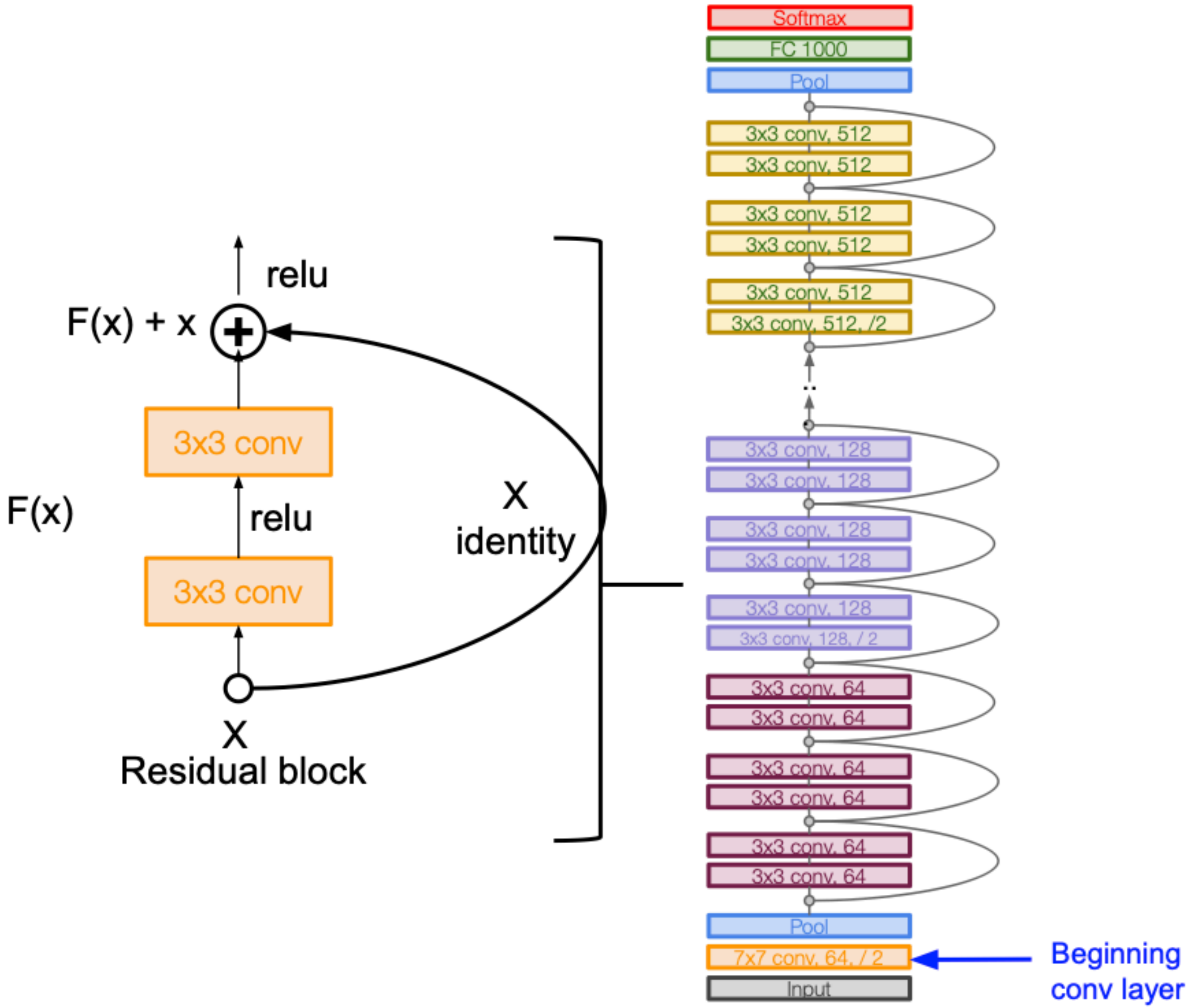
ResNet



"Revolution of Depth"
152 layers
152 layers
152 layers
19 layers
22 layers
shallow
8 layers
8 layers
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
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

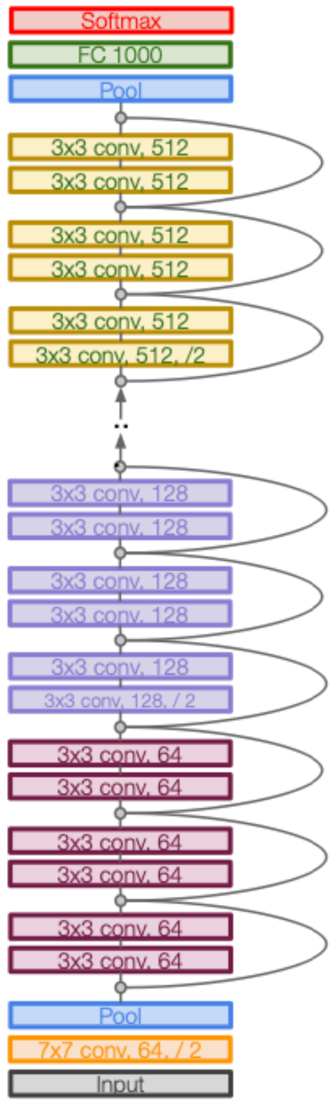
ResNet

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

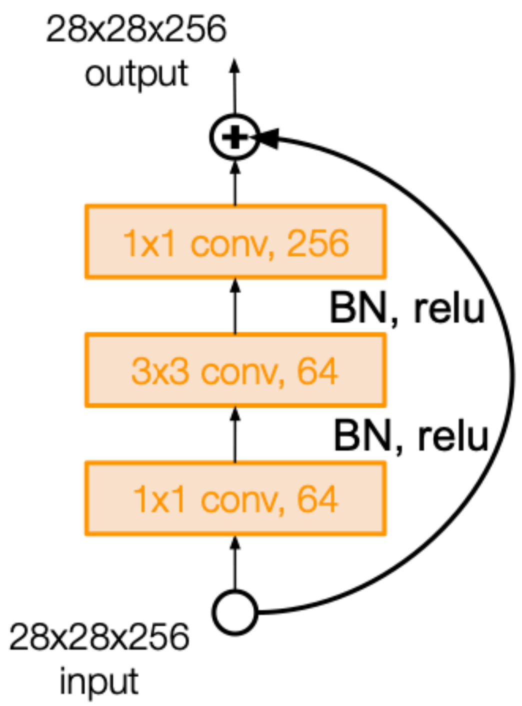
Total depths of 18, 34, 50, 101, or 152 layers for ImageNet



ResNet

[He et al., 2015]

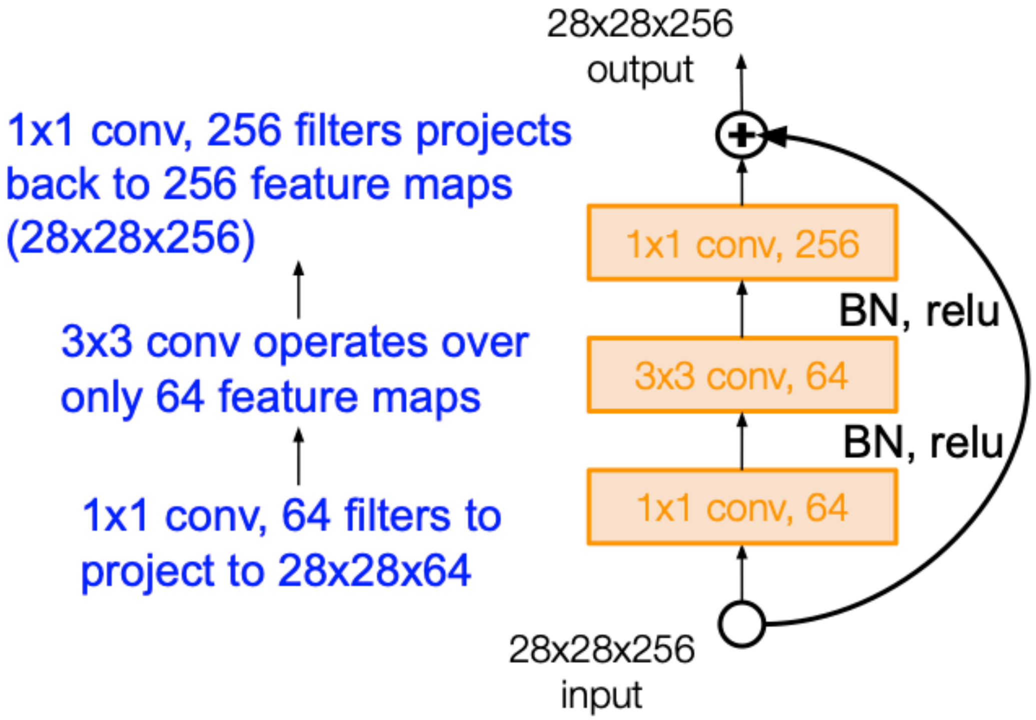
For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



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

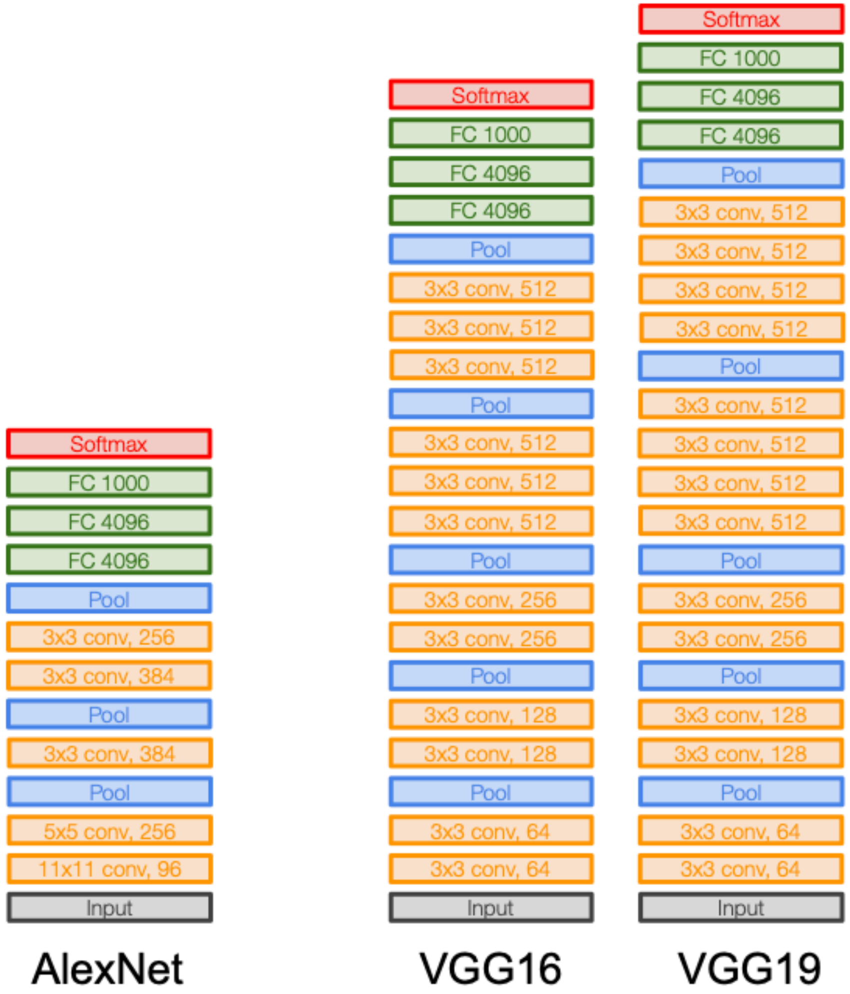
VGGNet

Small filters, Deeper networks

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

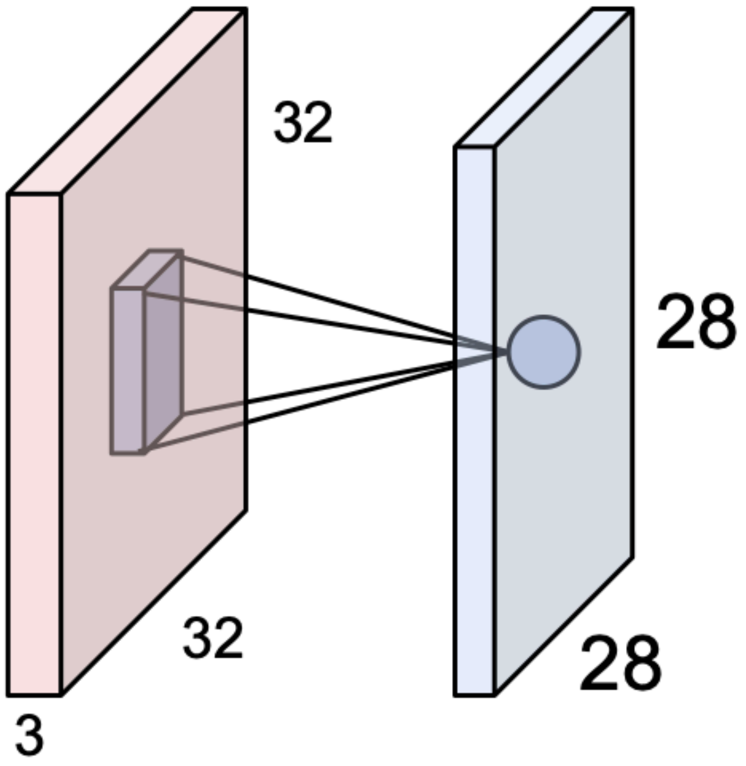
Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

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

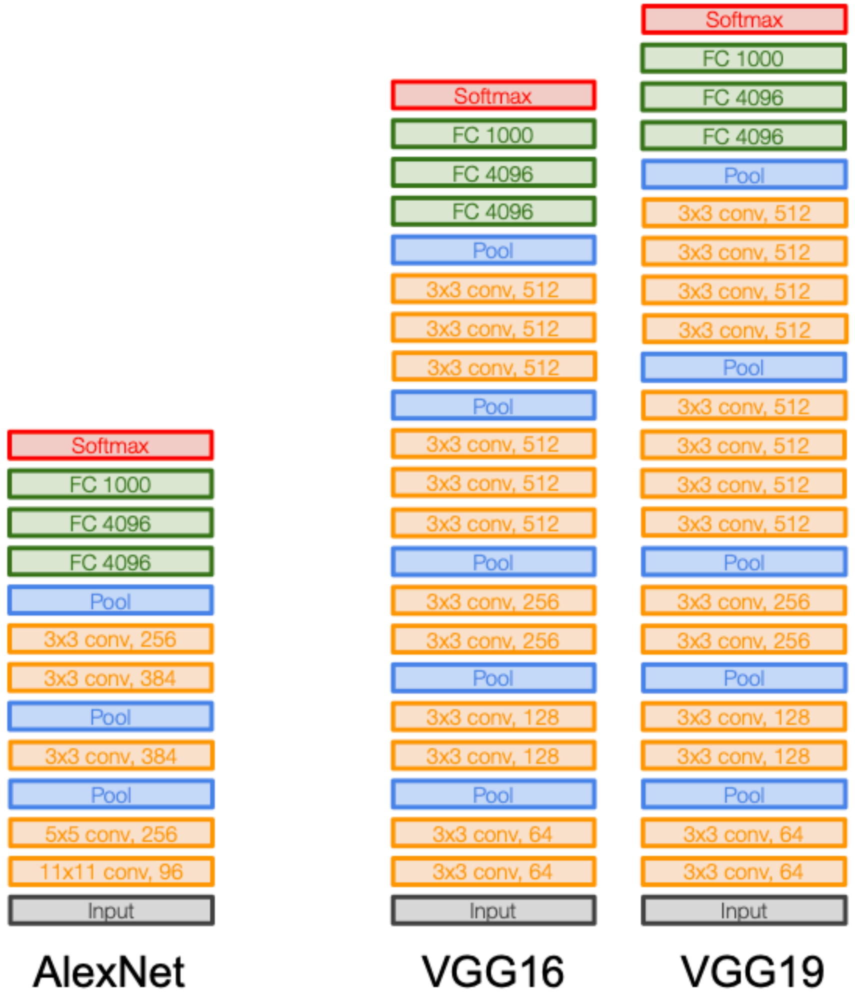
"5x5 filter" -> "5x5 receptive field for each neuron"

VGGNet

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

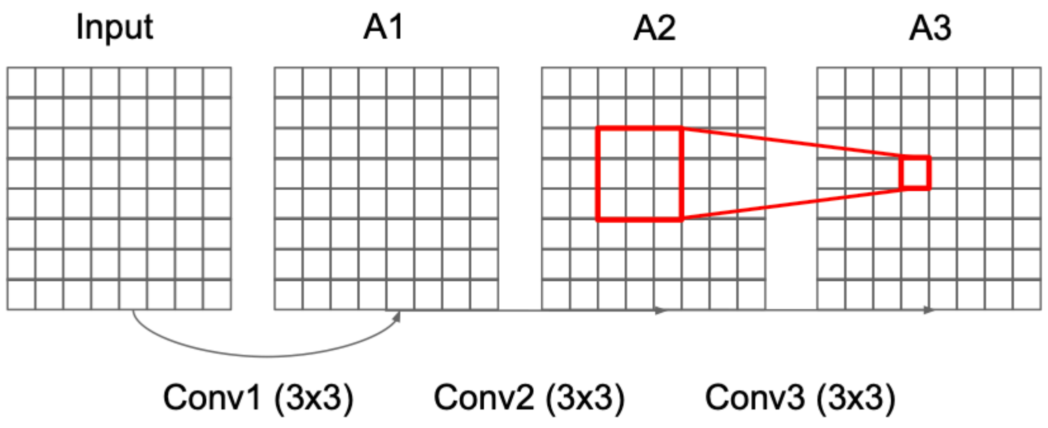
Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

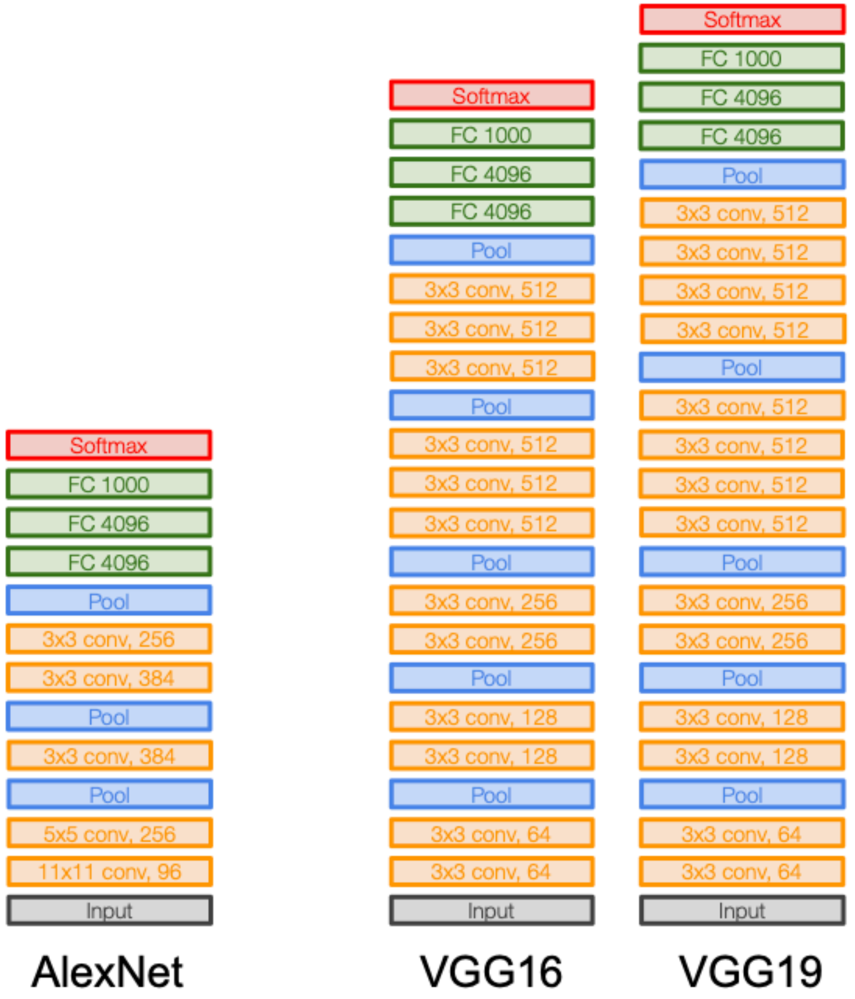
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?



Receptive Field

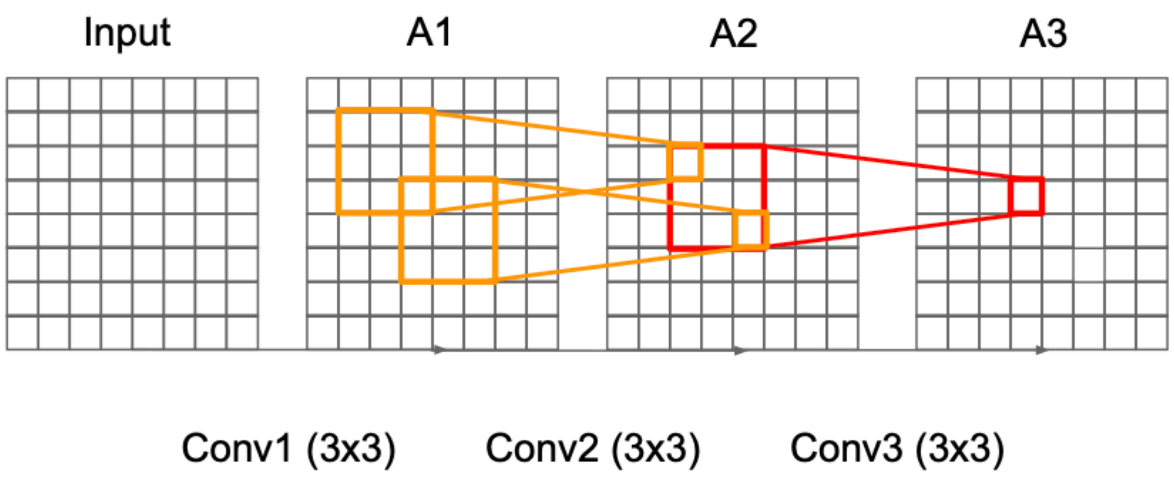
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

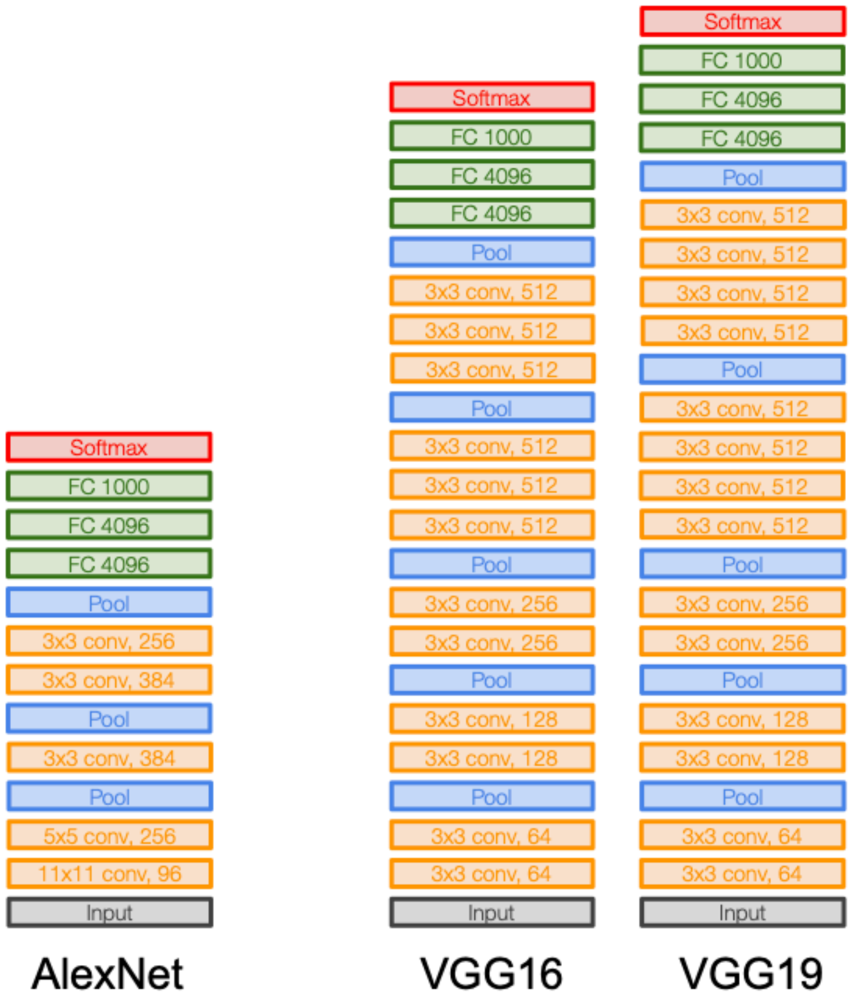




Receptive Field

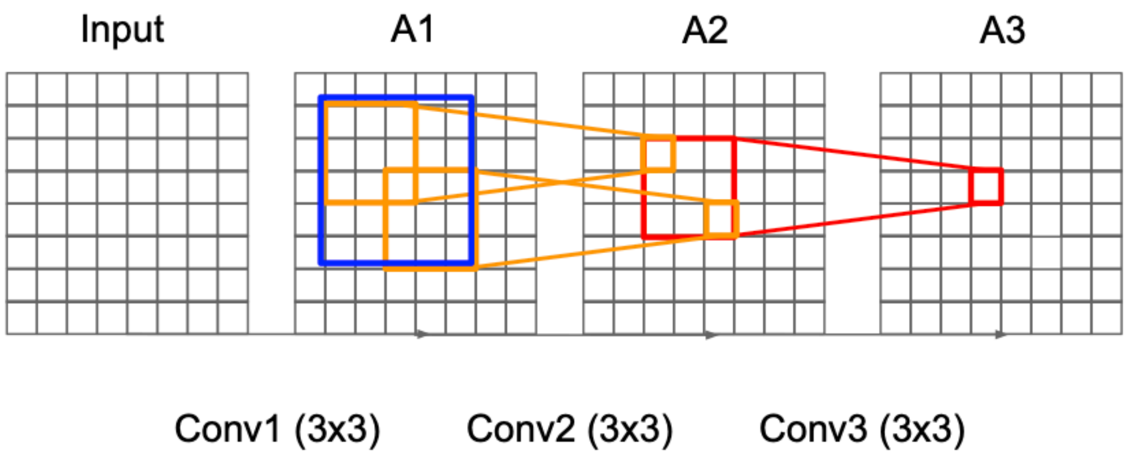
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

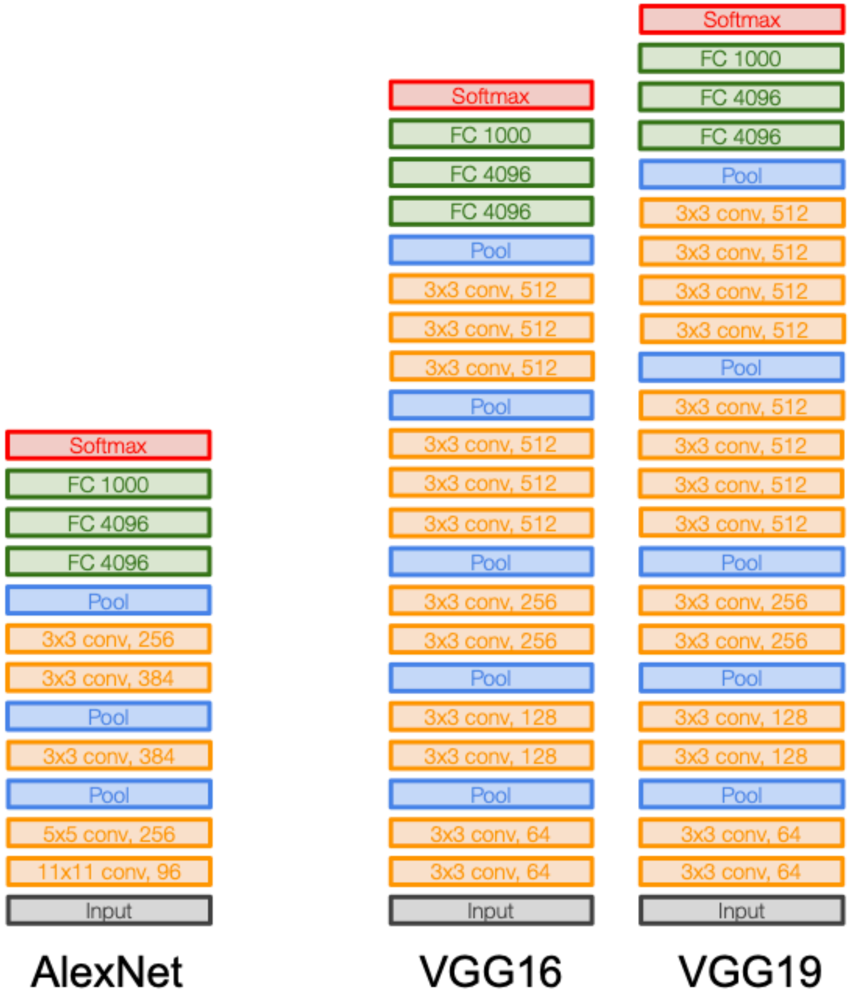




Receptive Field

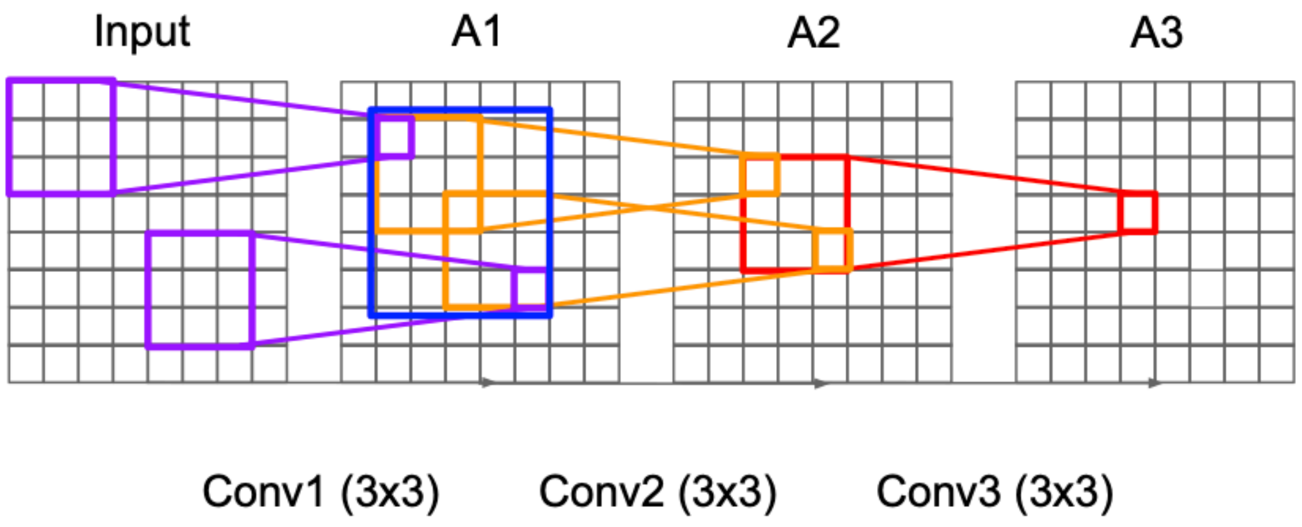
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

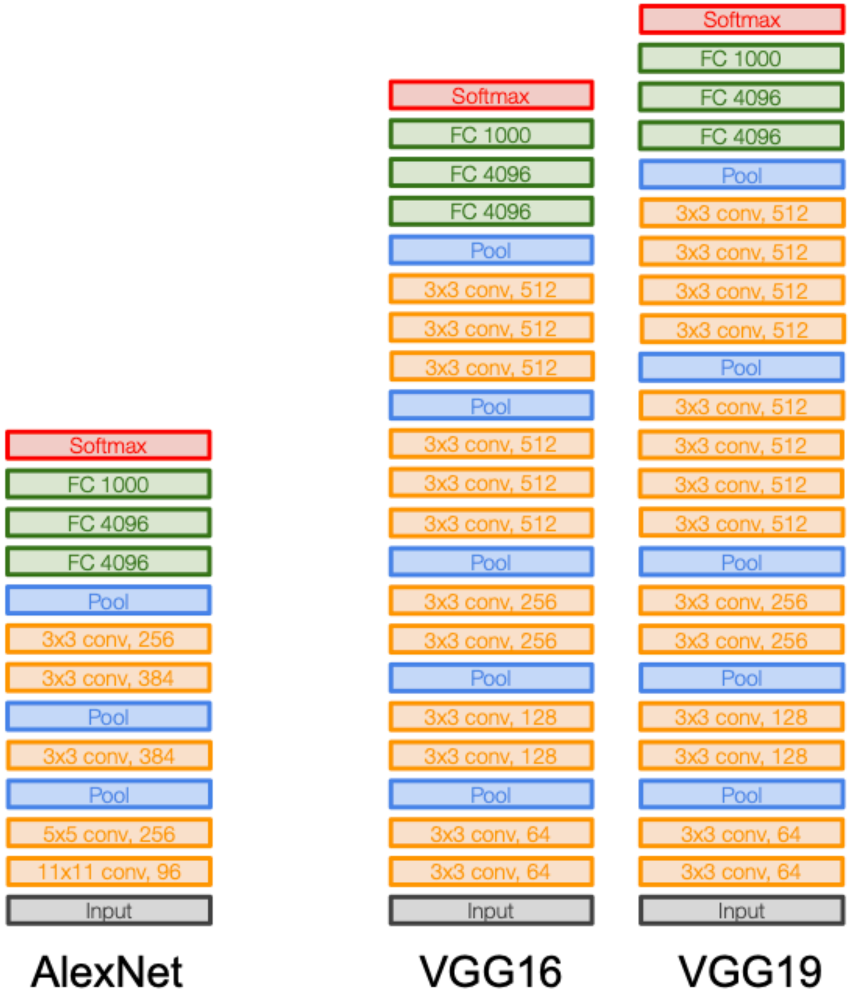




Receptive Field

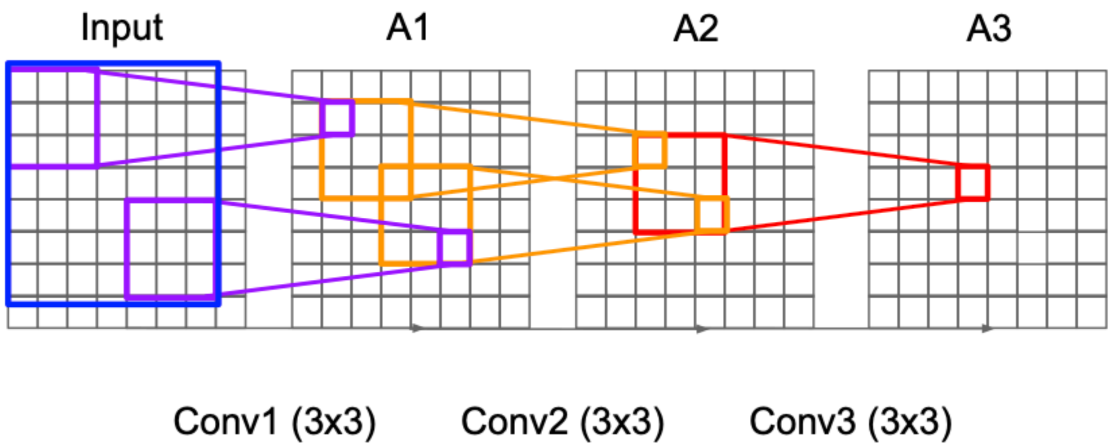
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

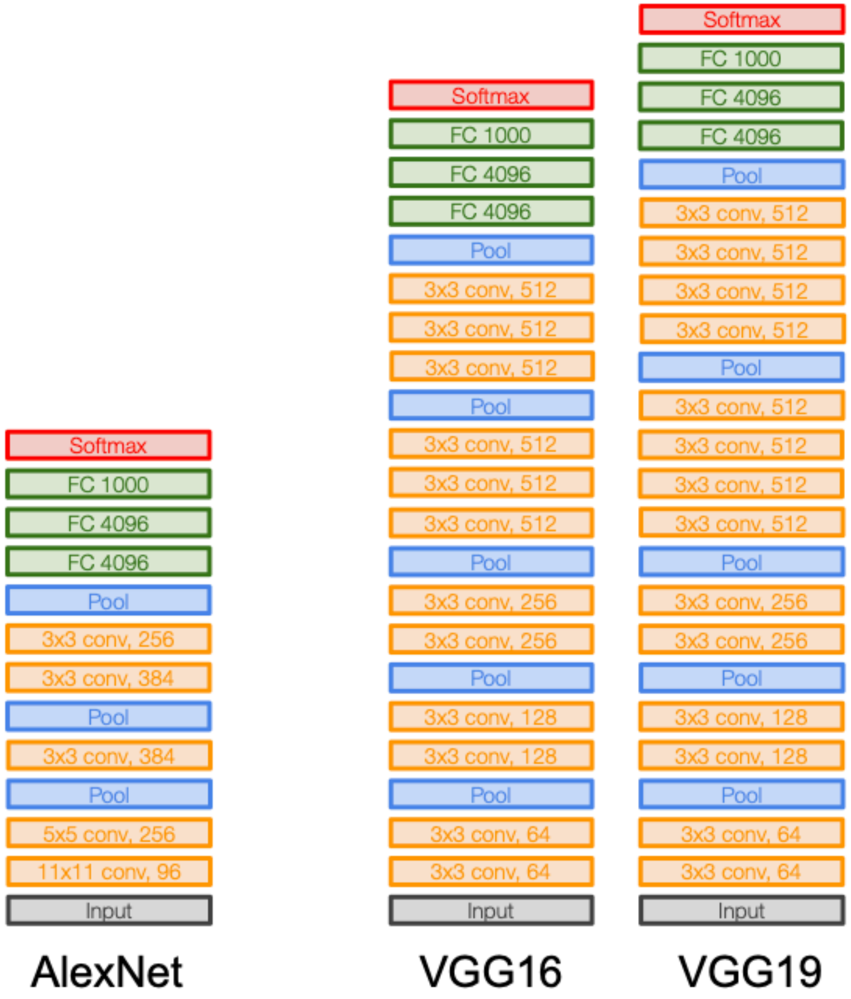




Receptive Field

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



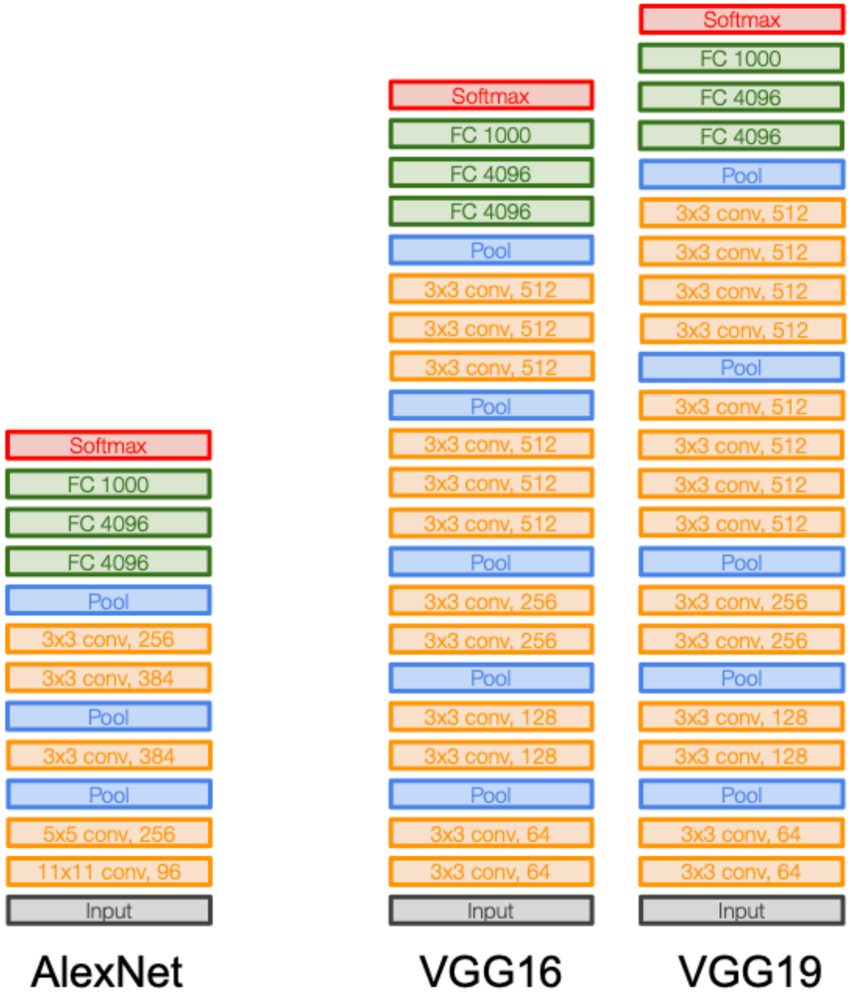


Receptive Field

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]



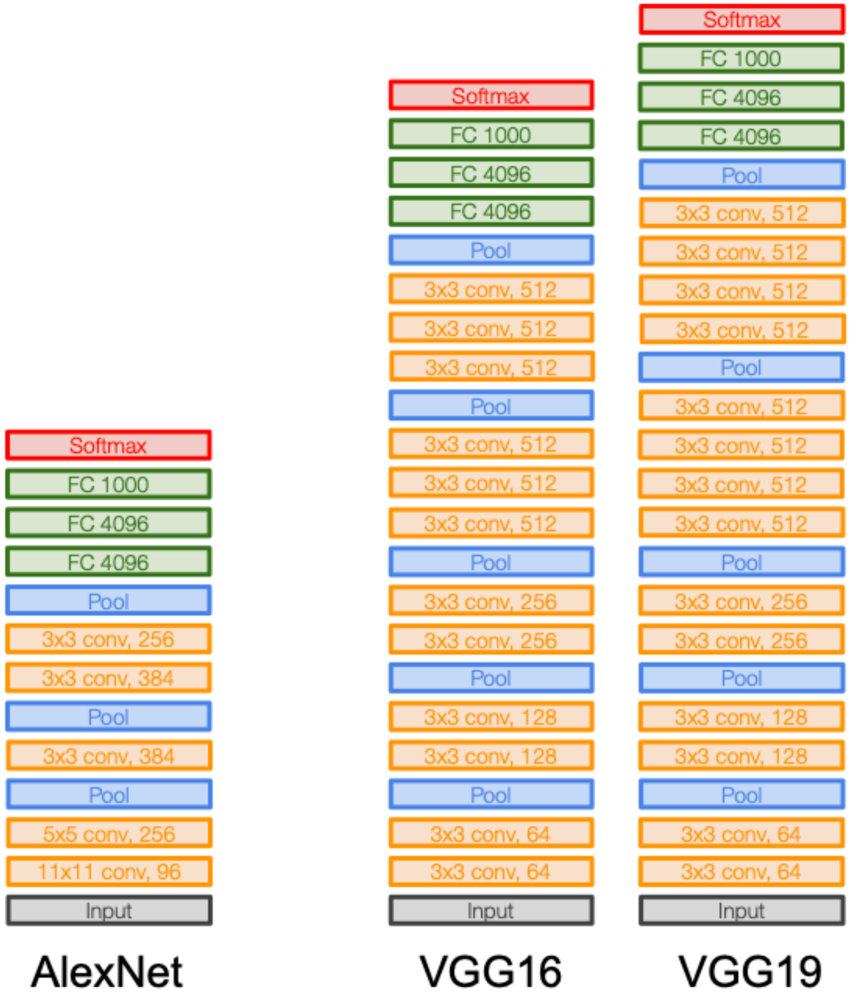
Receptive Field

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

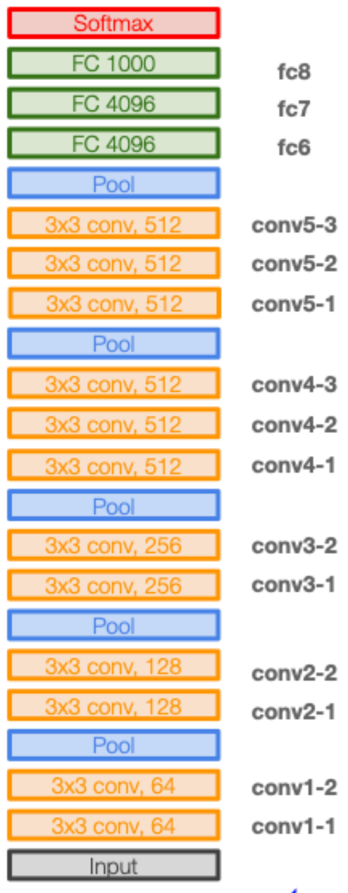
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448

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)

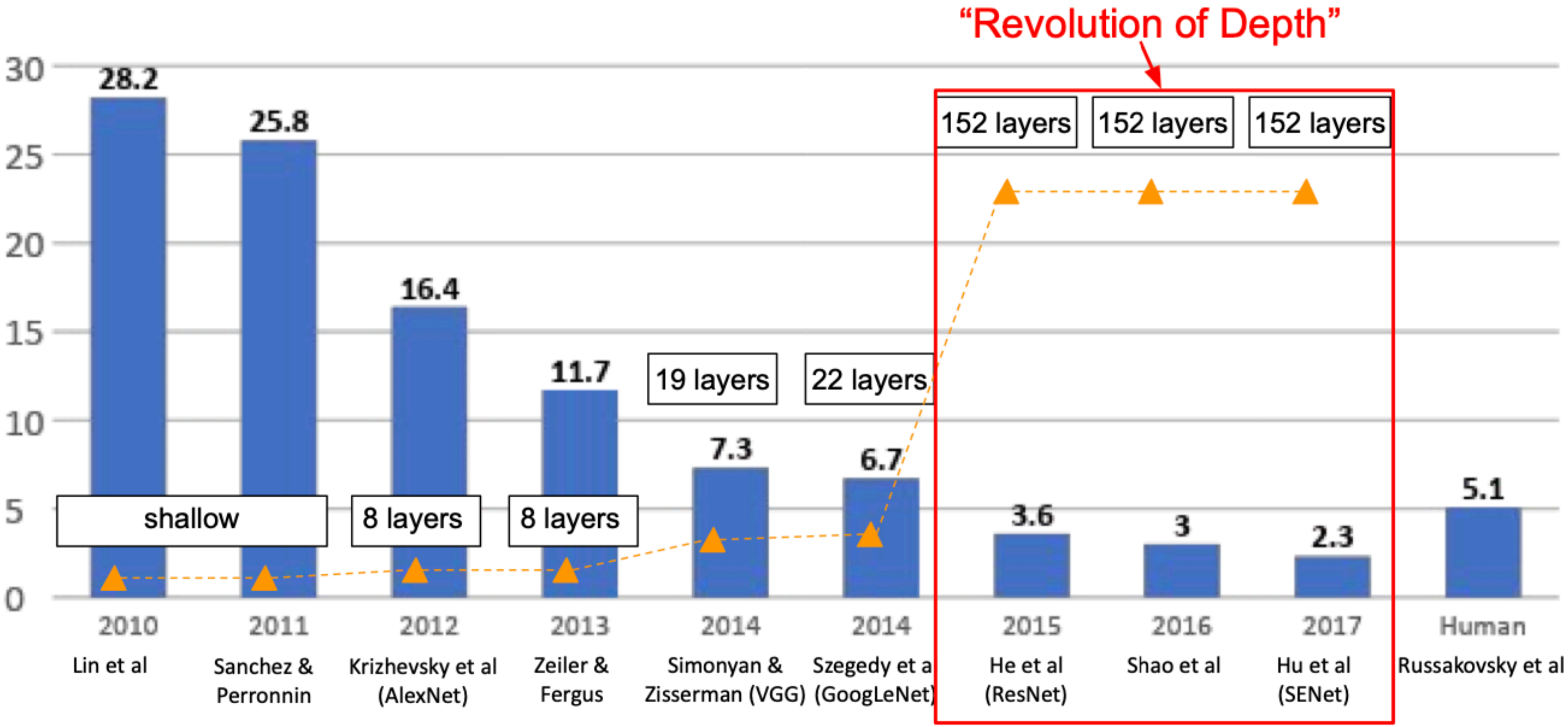
TOTAL params: 138M parameters



VGG16

Common names

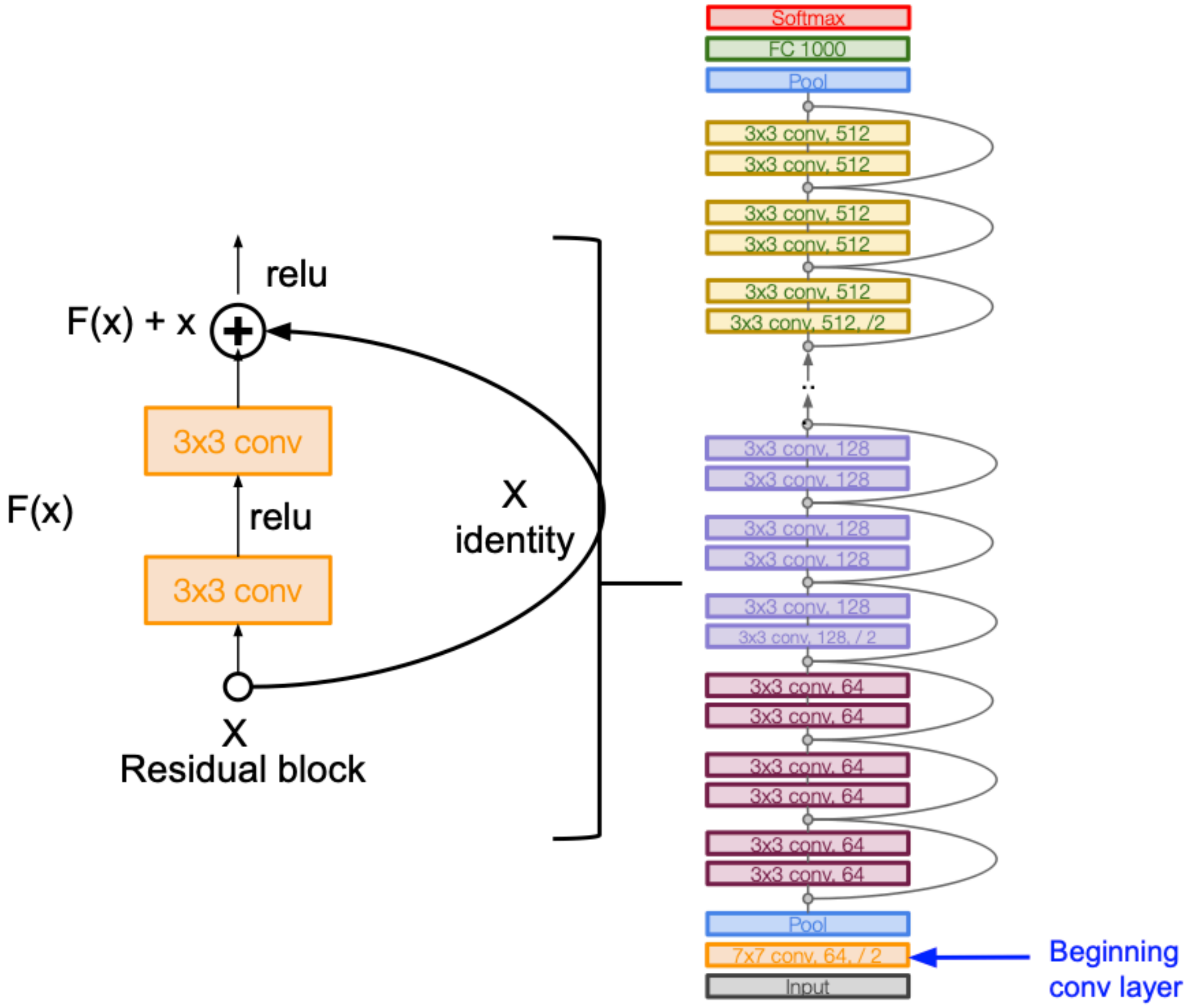
ResNet



"Revolution of Depth"
152 layers
152 layers
152 layers
19 layers
22 layers
shallow
8 layers
8 layers
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
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

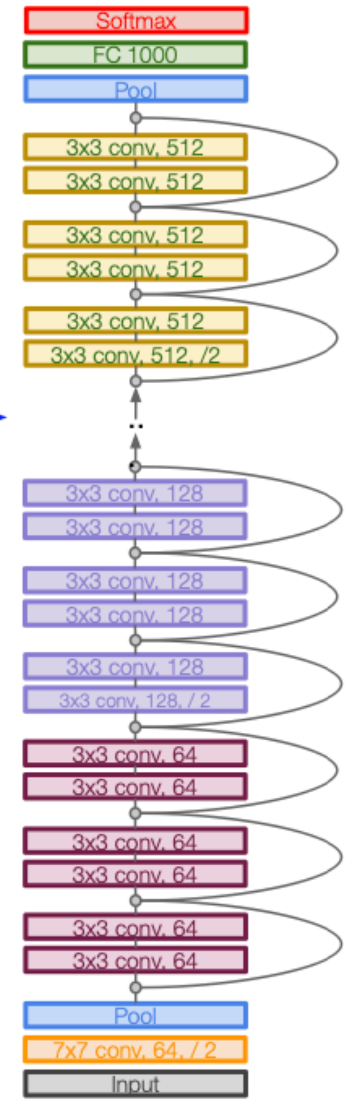
ResNet

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

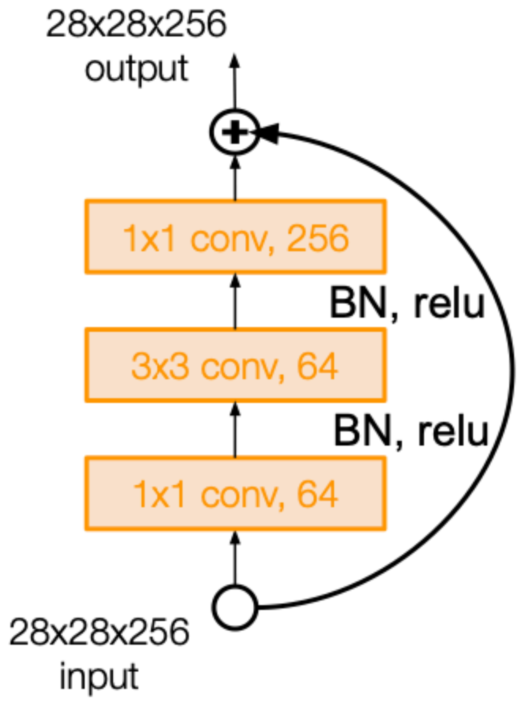
Total depths of 18, 34, 50, 101, or 152 layers for ImageNet



ResNet

[He et al., 2015]

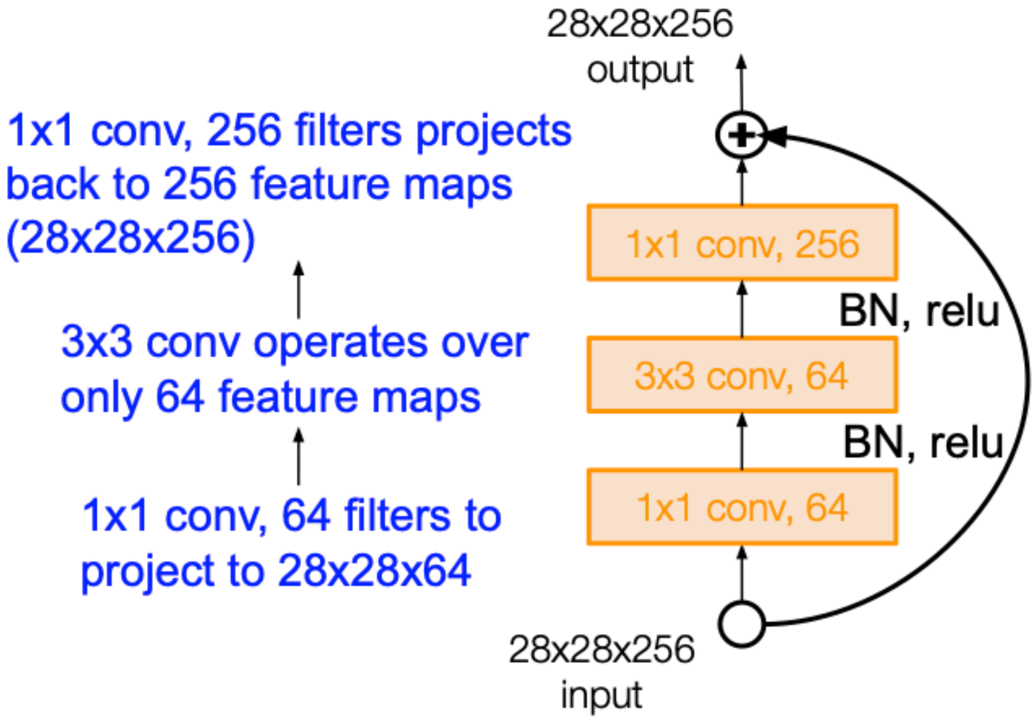
For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to GoogLeNet)



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

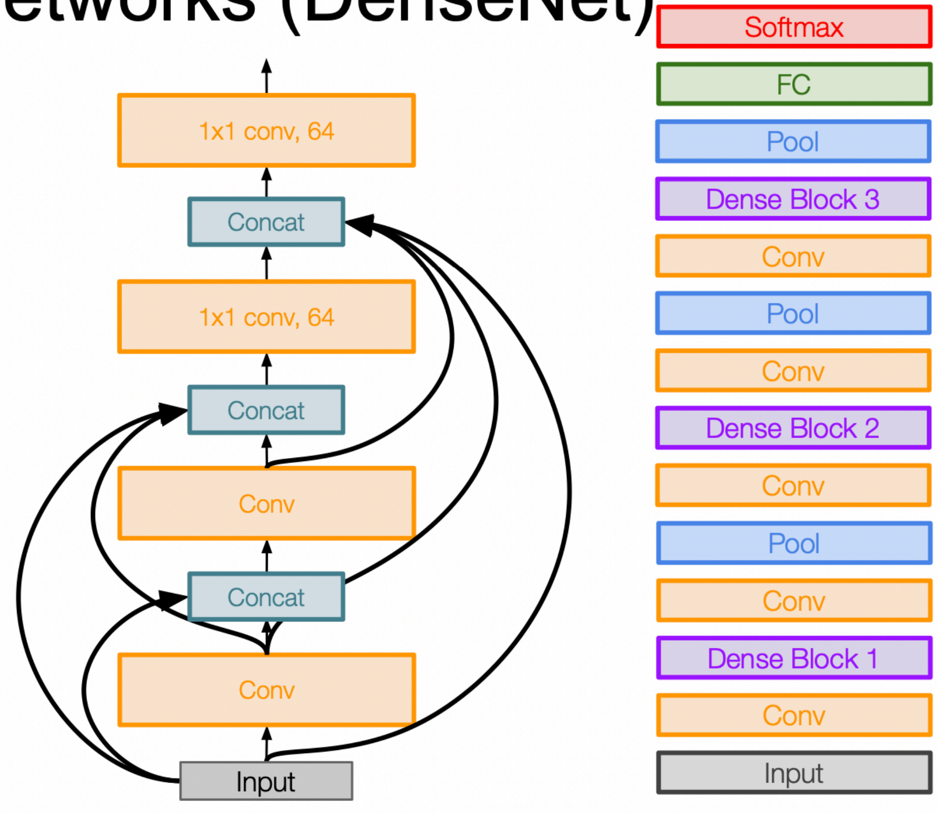
- Squeeze-and-Excitation Network (SENet)
- Wide Residual Networks
- ResNeXt
- DenseNet

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

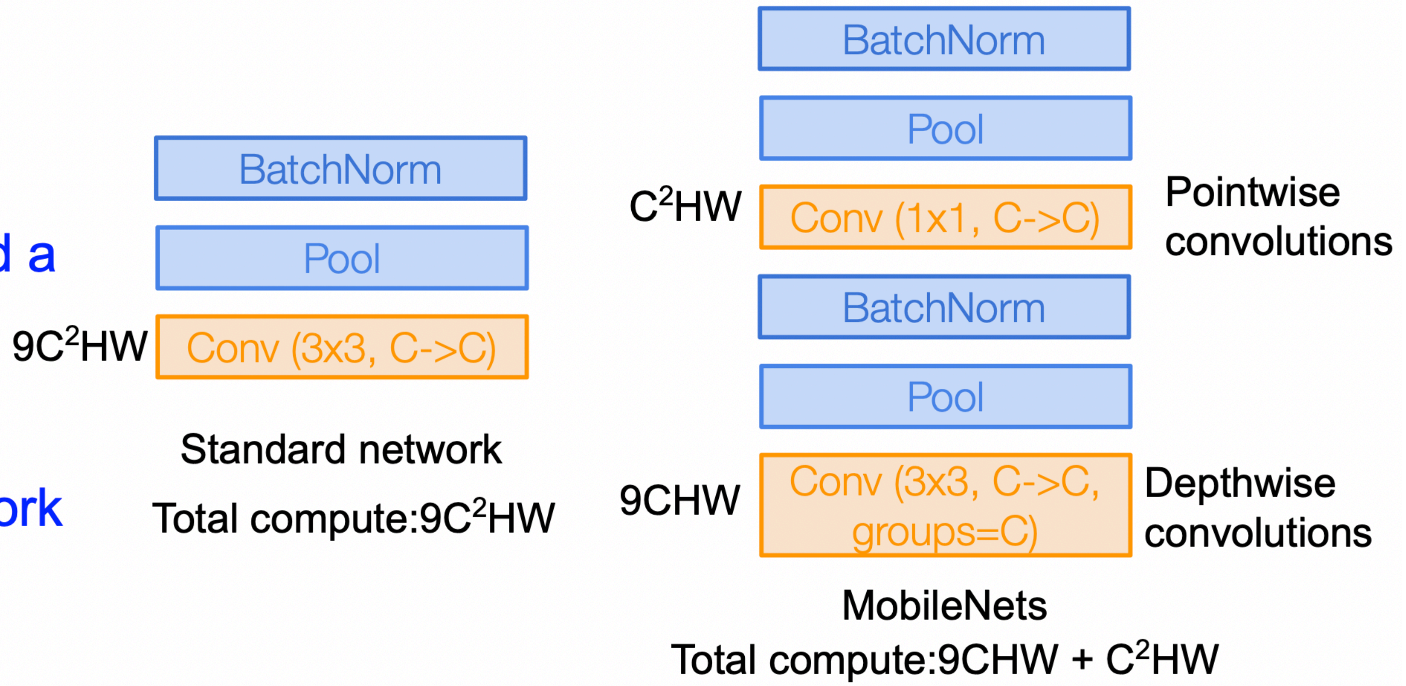
- Attention-based networks: ViT, SwinTransformer
- MLP-based networks

- 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

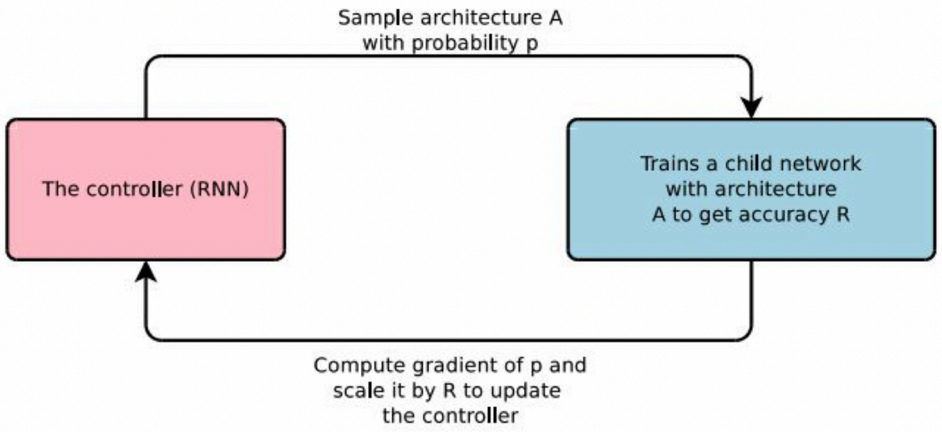
- Neural architecture search

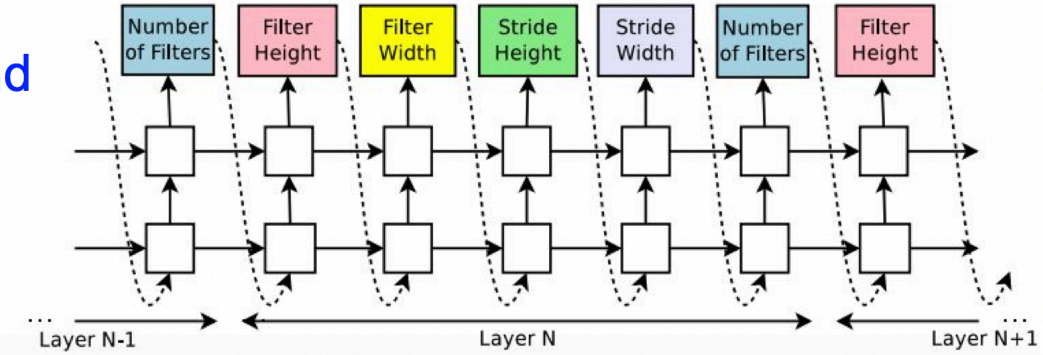
Learning to Search for Network Architecture

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

$x =$

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

Model

$h($  $,\theta)$

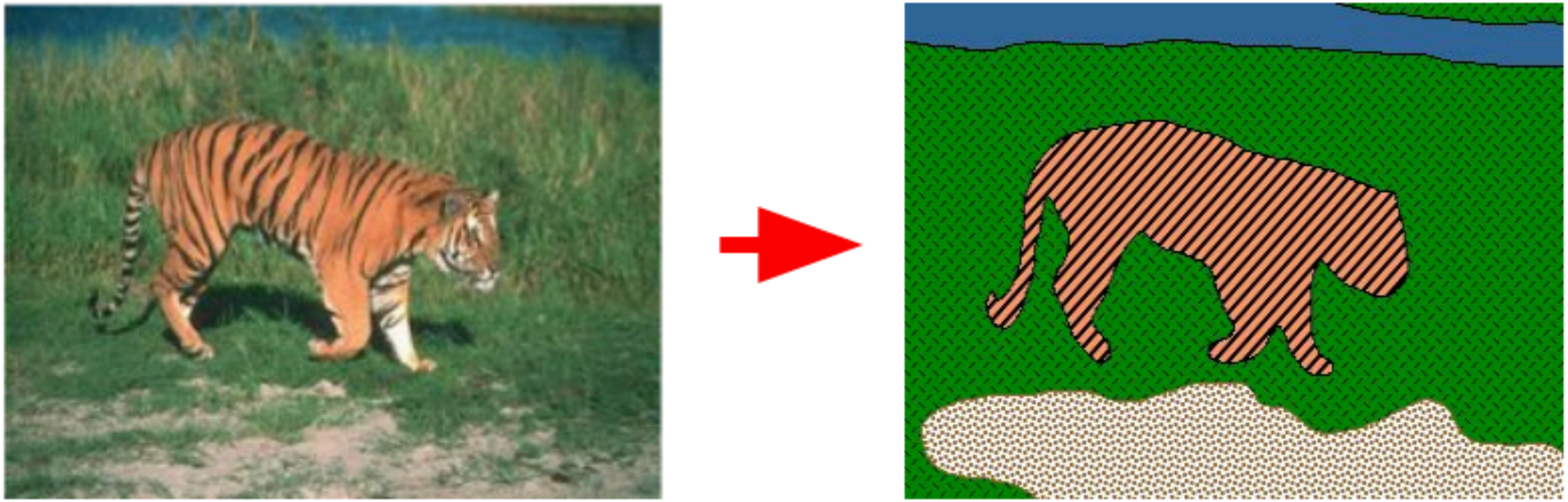
Parameters or weights

Output: which class

$$\begin{array}{cl} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \begin{array}{l} \text{cat} \\ \text{dog} \\ \text{panda} \\ \\ \text{elephant} \end{array} \end{array}$$

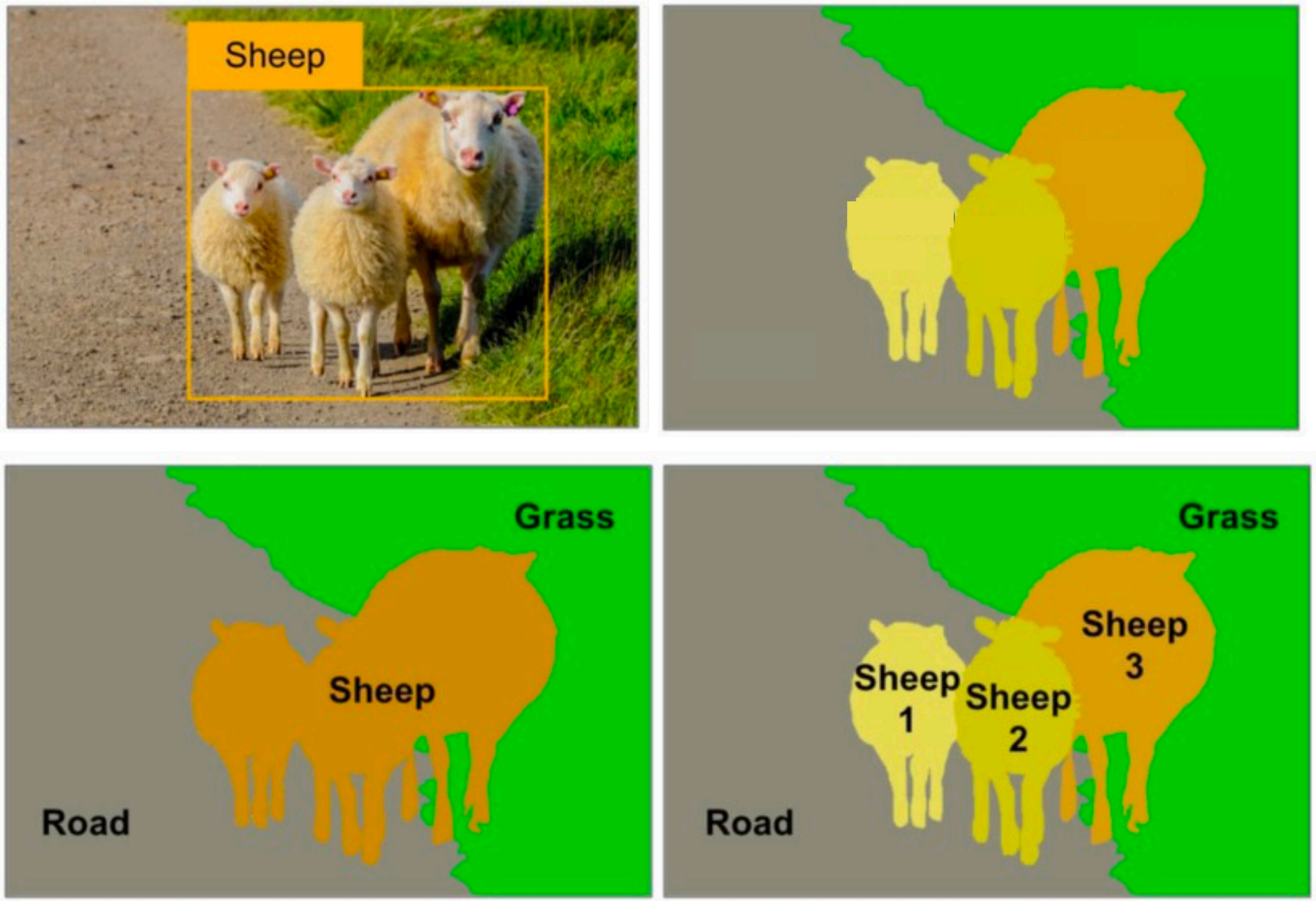
Image Segmentation

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



We Care About Semantics

Classification + localization | Instance Segmentation

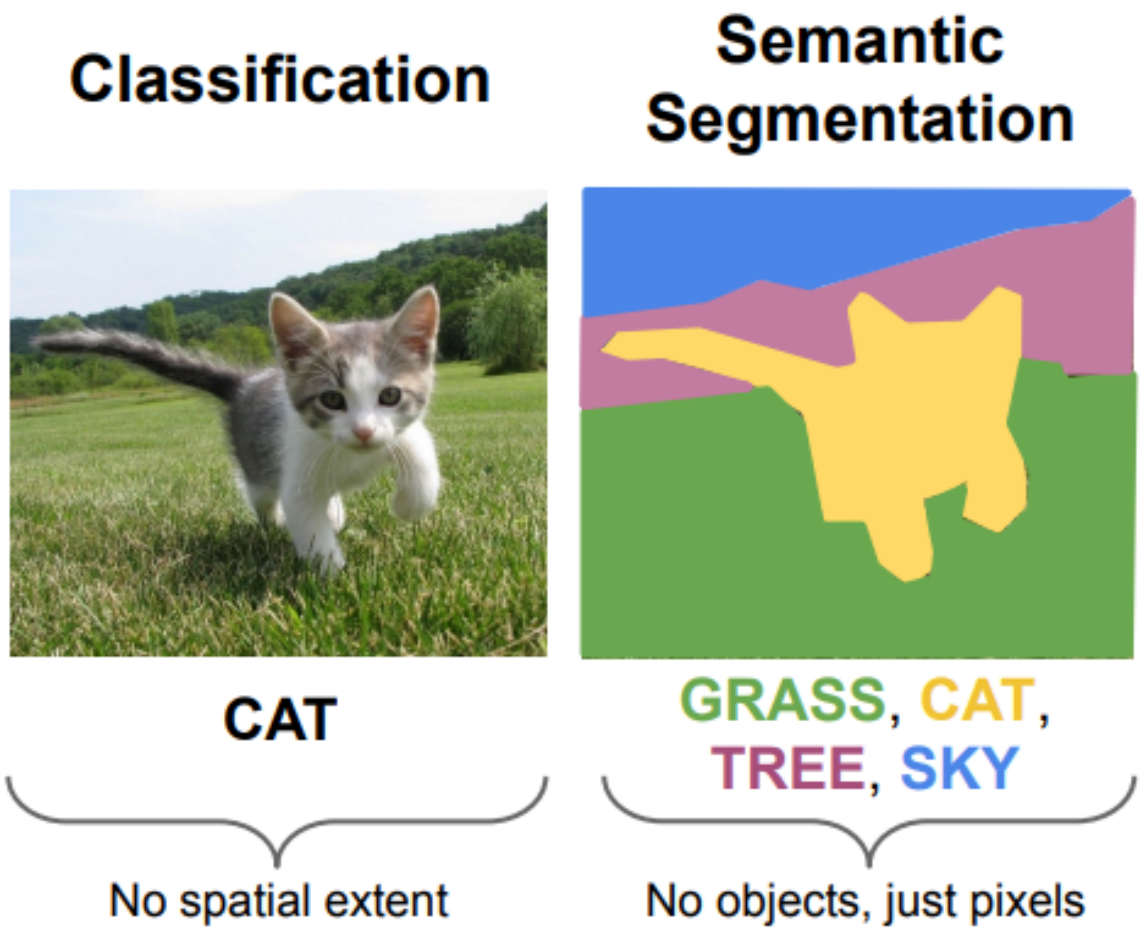


Semantic Segmentation | Semantic Instance Segmentation

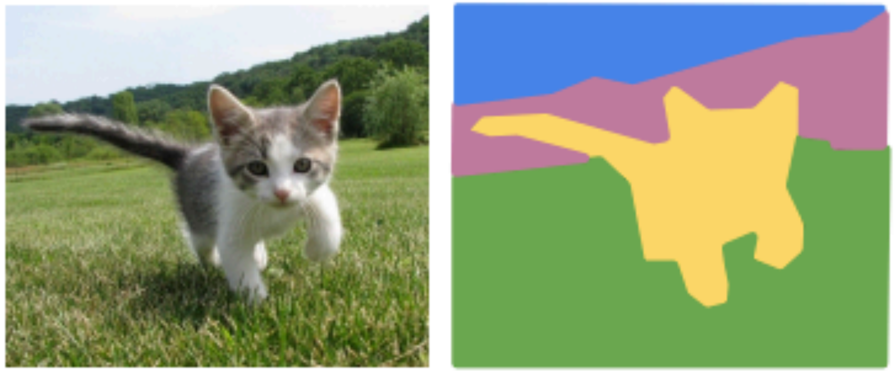
Semantic Segmentation

- 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.

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

Semantic Segmentation using Sliding Window

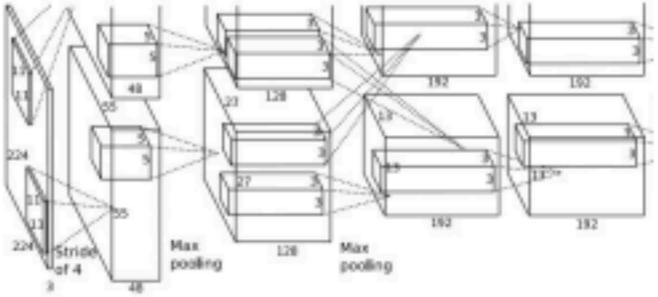
Full image

Impossible to classify without context

Q: how do we include context?

Semantic Segmentation using CNN

Full image

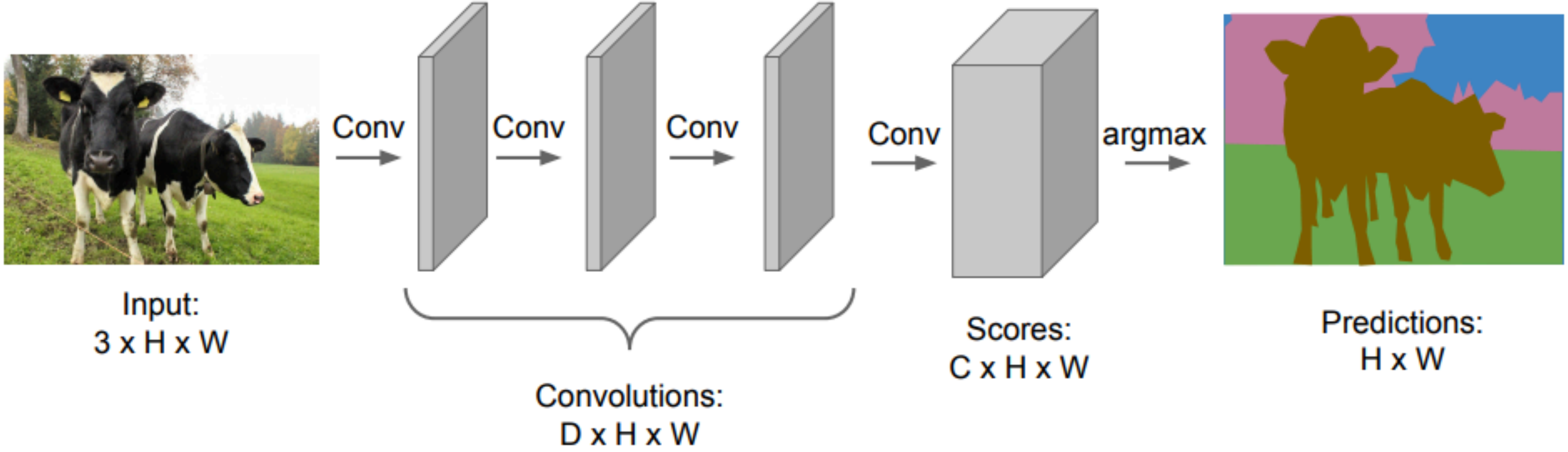


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

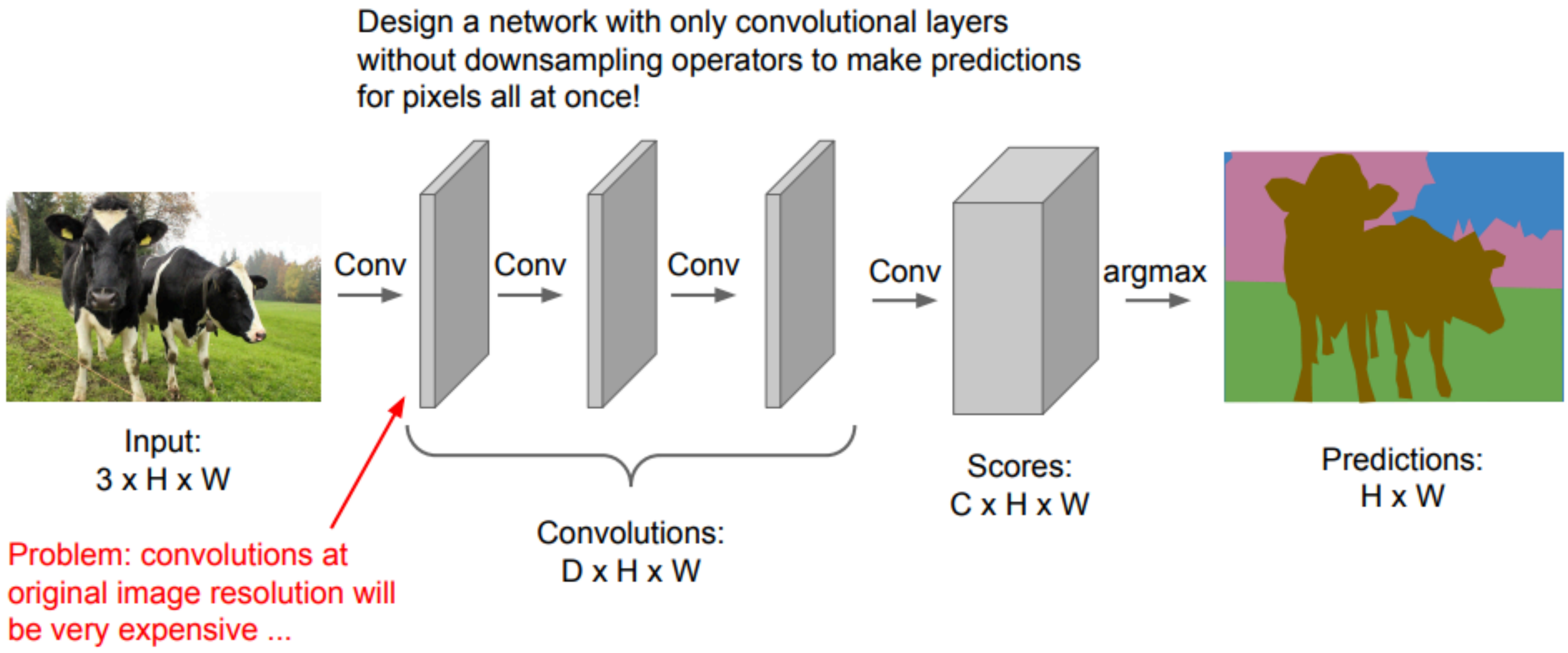
Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

Semantic Segmentation using Fully Convolution

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

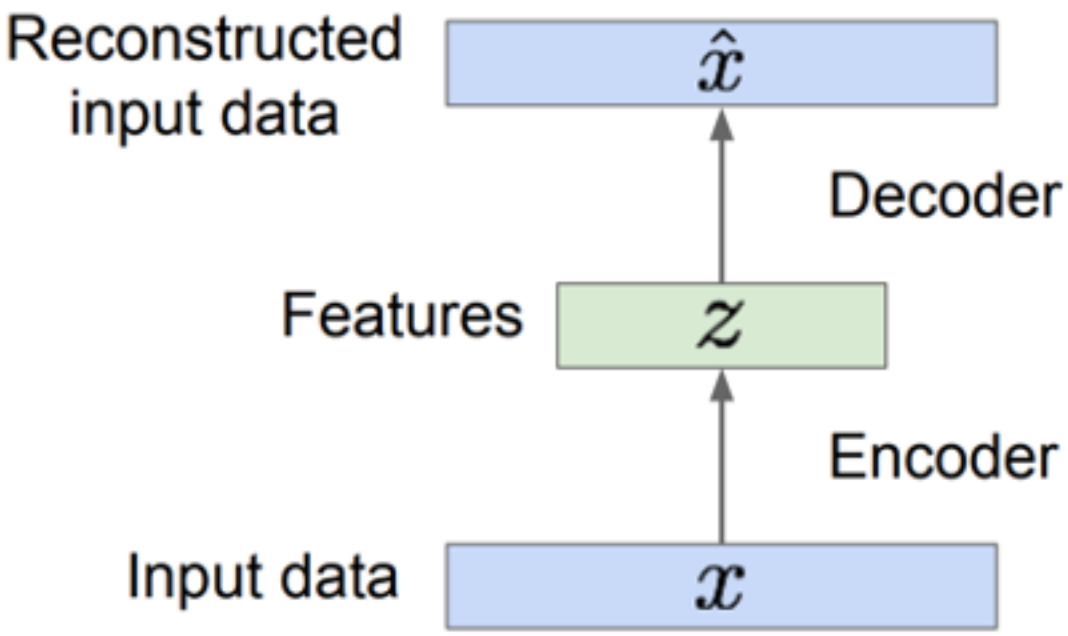


Semantic Segmentation using Fully Convolution



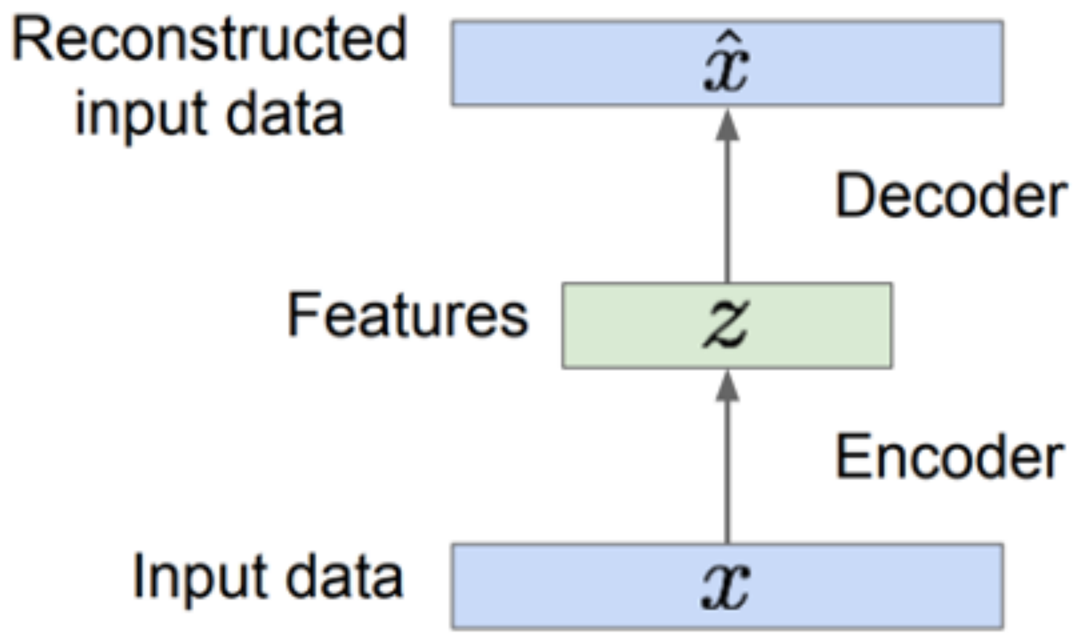
We need to reduce resolutions.

Auto-Encoder



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

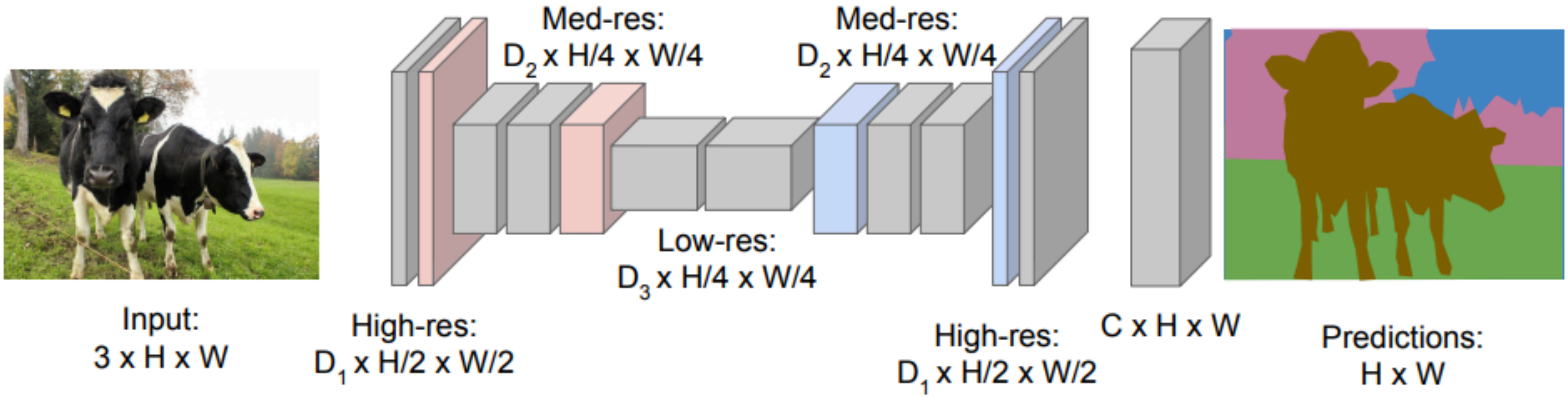
Auto-Encoder



- 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!



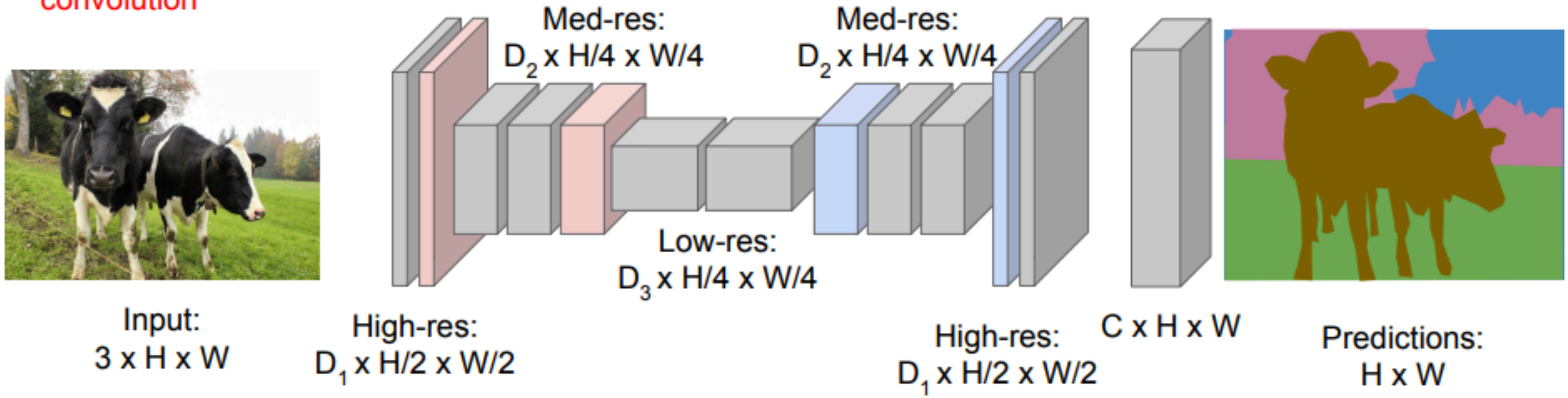
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

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:
???



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

Input: 2 x 2

→

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

Output: 4 x 4

"Bed of Nails"

1	2
3	4

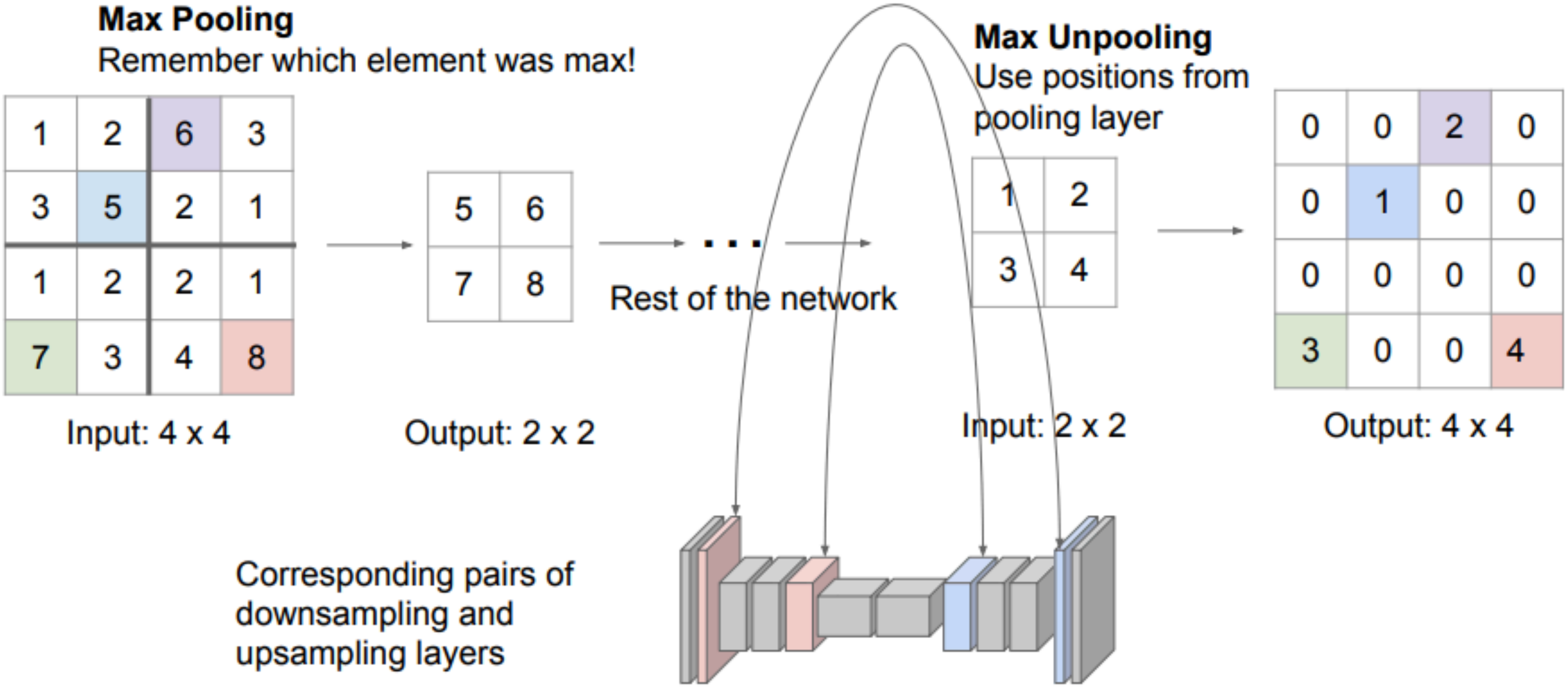
Input: 2 x 2

→

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

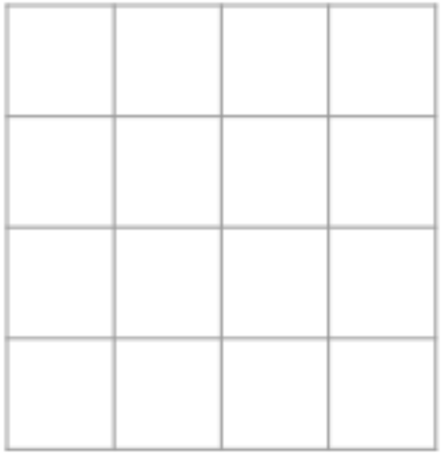
Output: 4 x 4

In-Network Upsampling: Max Unpooling

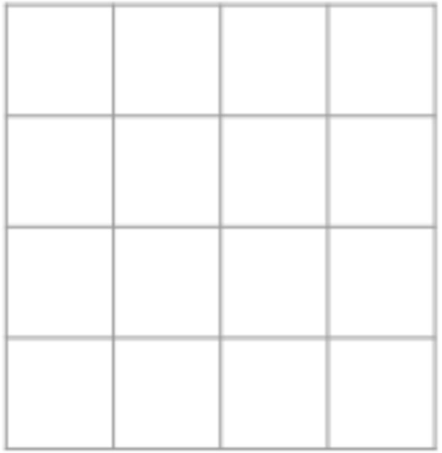


Learnable Upsampling

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



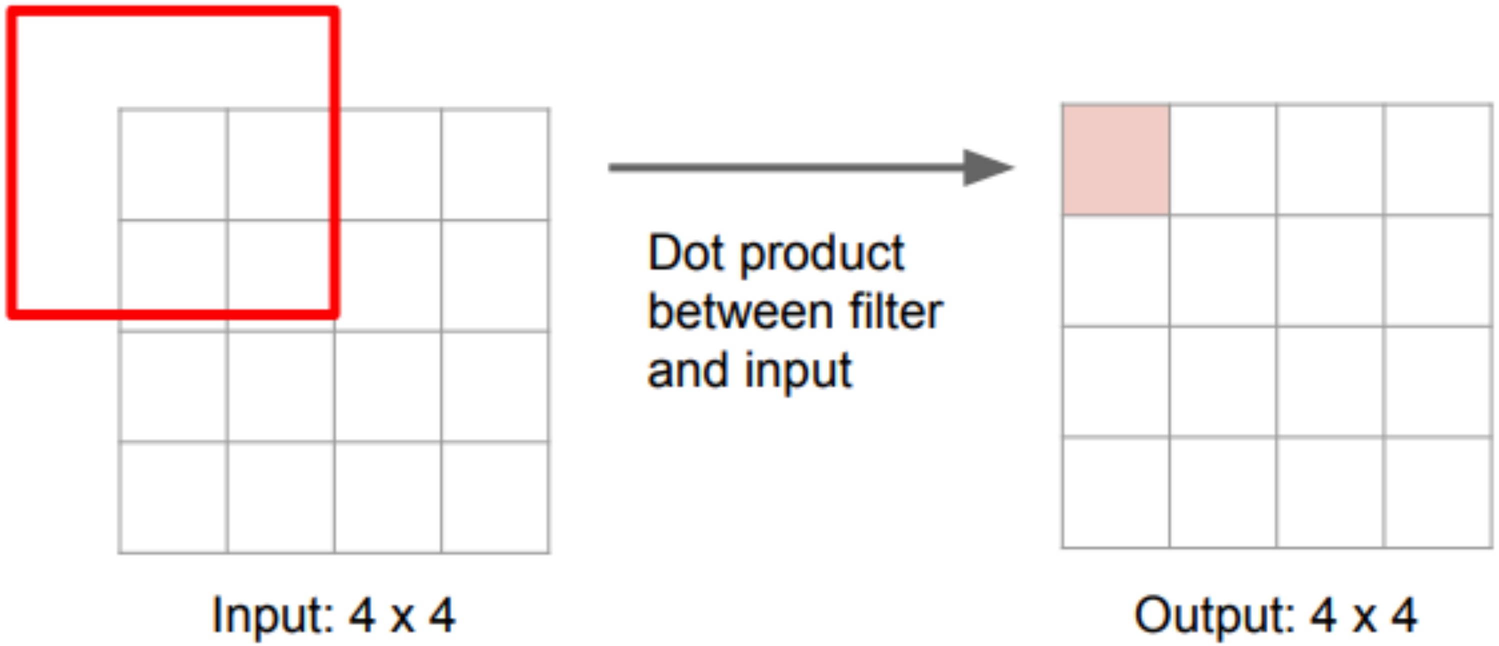
Input: 4 x 4



Output: 4 x 4

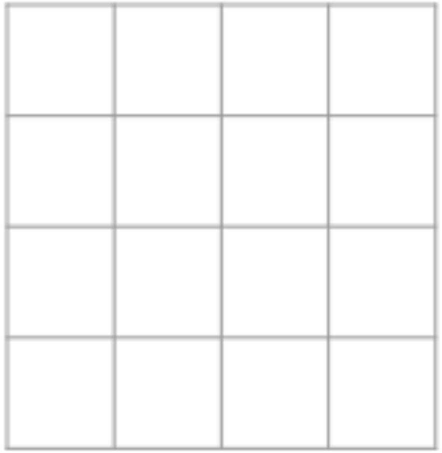
Learnable Upsampling

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

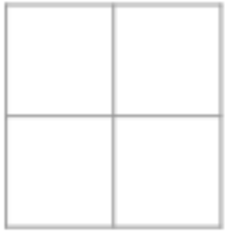


Learnable Upsampling

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



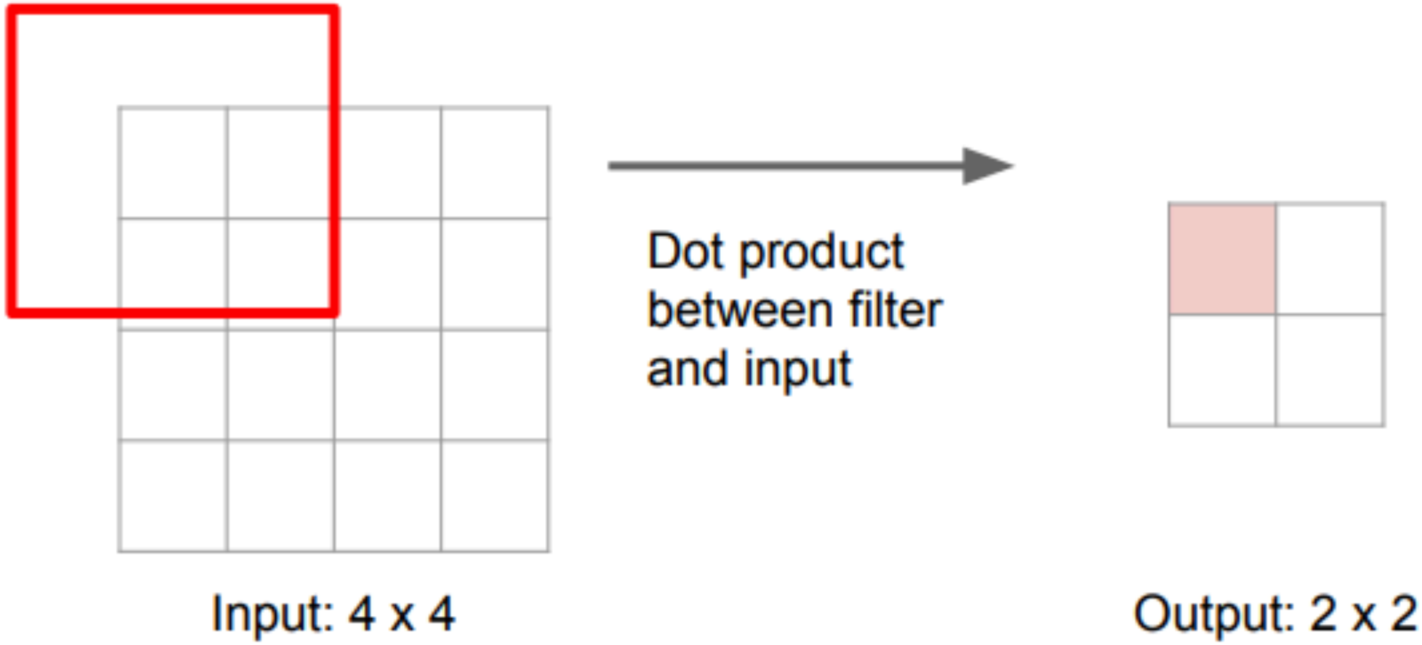
Input: 4 x 4



Output: 2 x 2

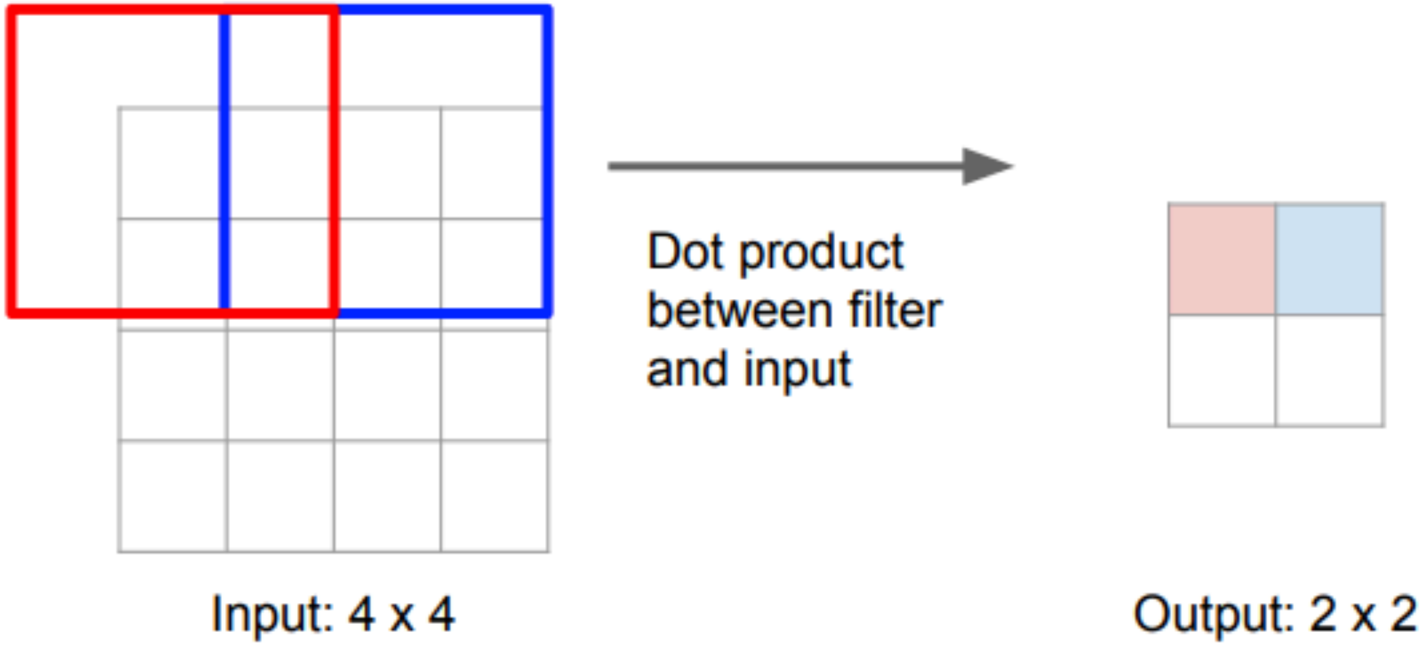
Learnable Upsampling

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



Learnable Upsampling

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

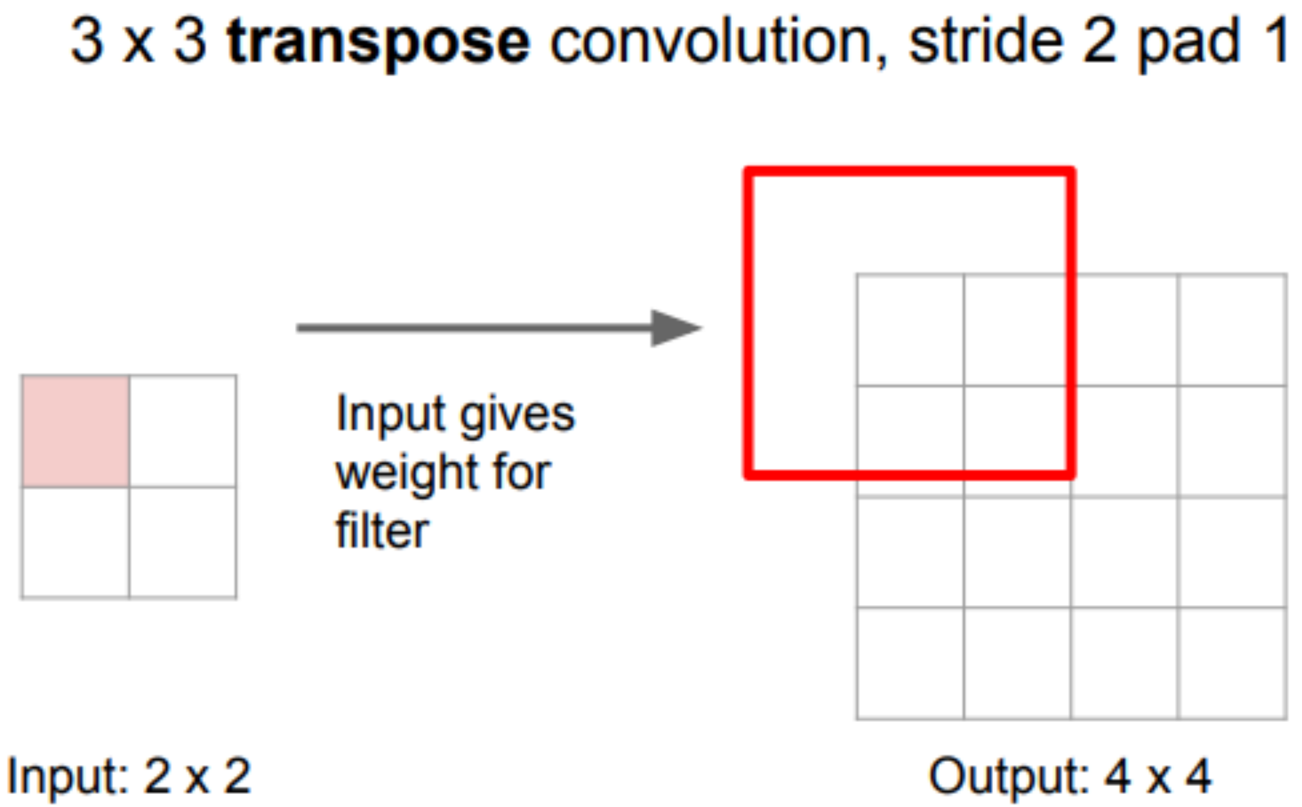


Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

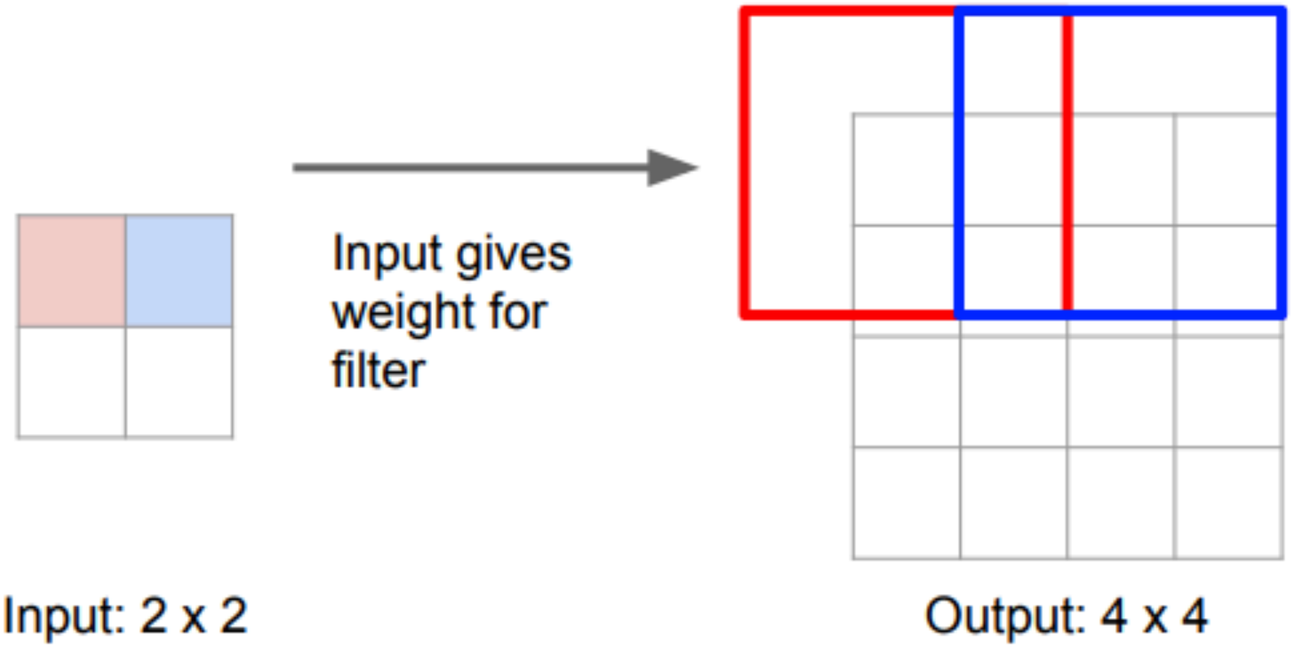
We can interpret strided convolution as "learnable downsampling".

Learnable Upsampling: Transposed Convolution



Learnable Upsampling: Transposed Convolution

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

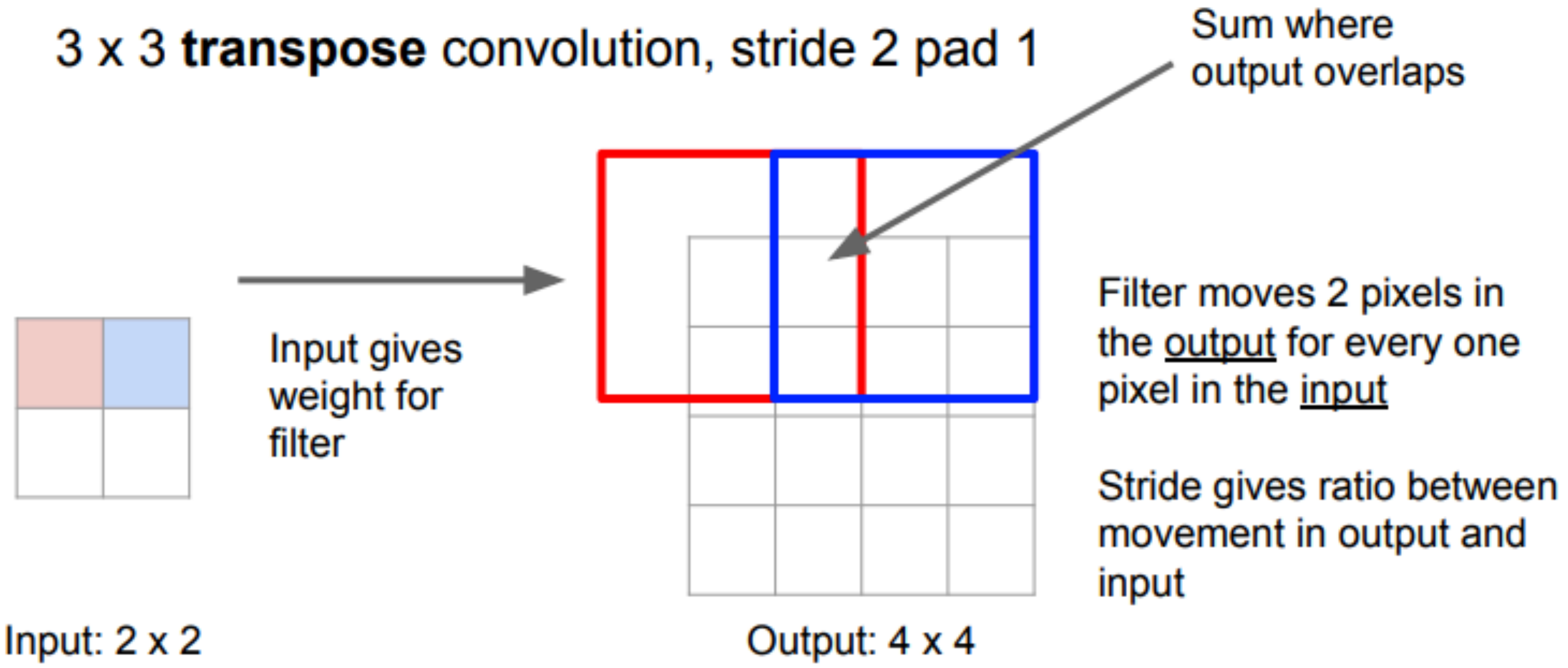


Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Learnable Upsampling: Transposed Convolution

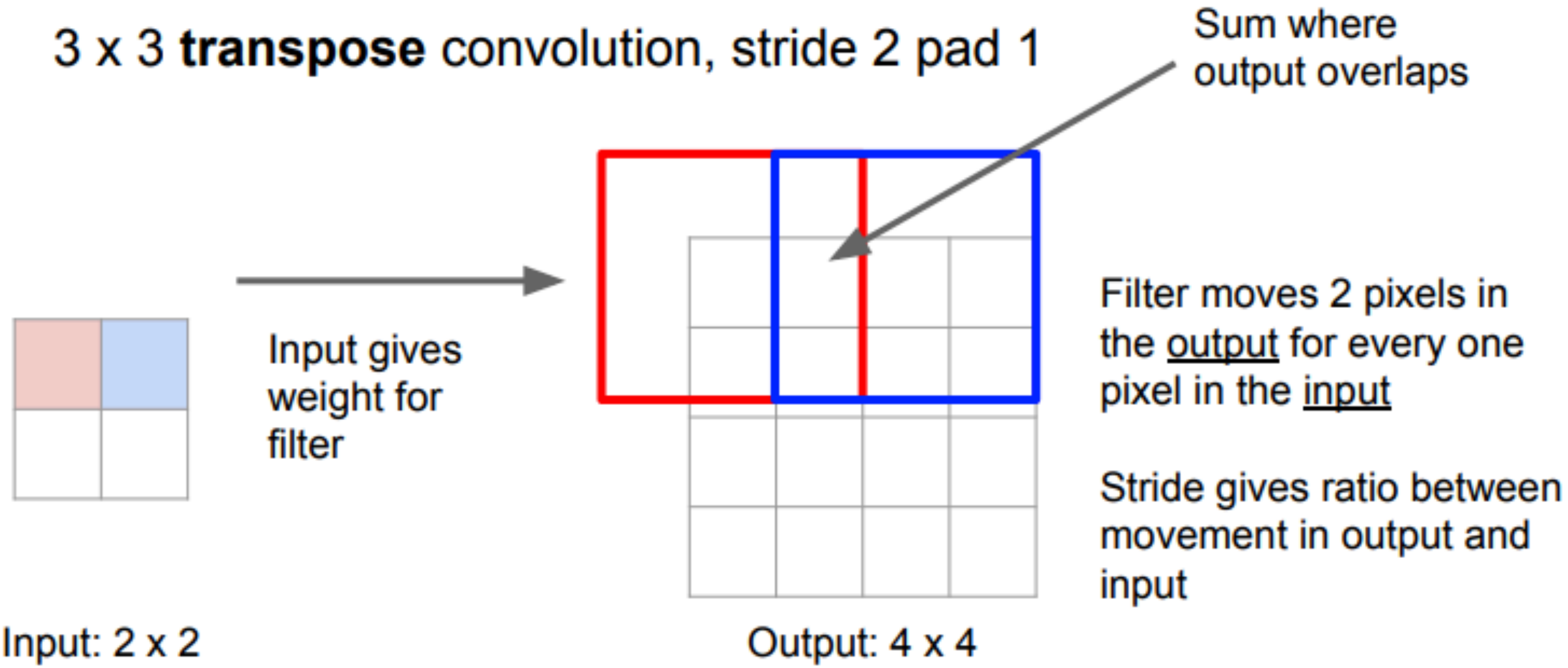
3 x 3 **transpose** convolution, stride 2 pad 1



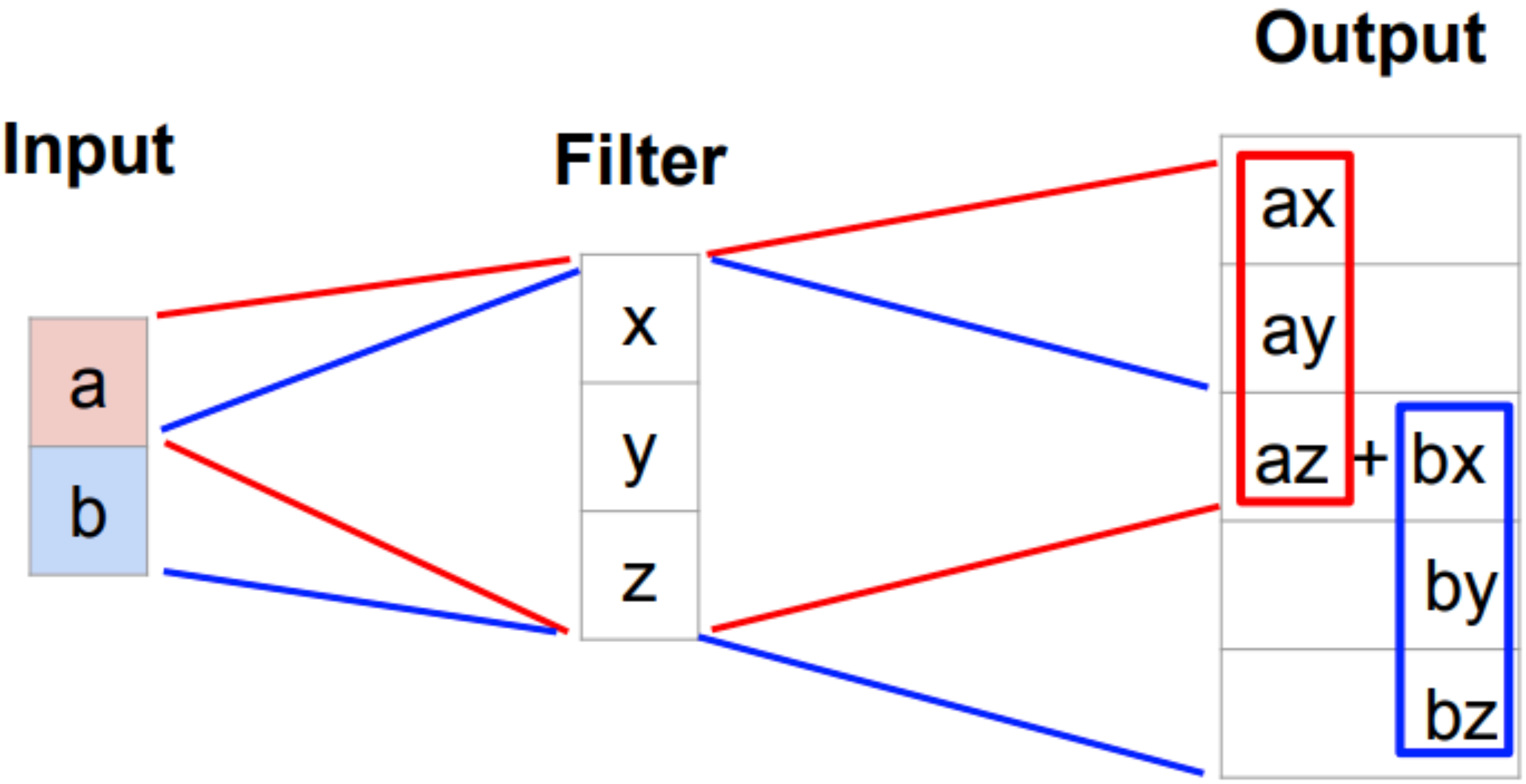
Learnable Upsampling: Transposed Convolution

Q: Why is it called transpose convolution?

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



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, <u>stride=2</u>, 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, <u>stride=2</u>, 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, <u>stride=2</u>, padding=0

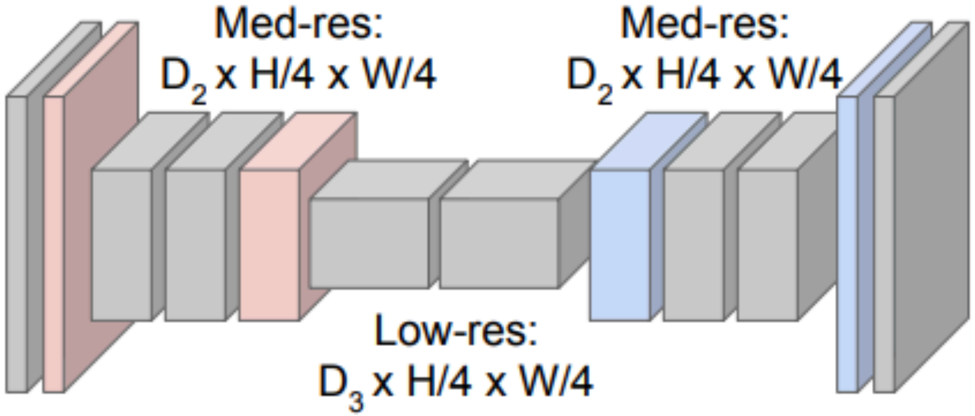
Semantic Segmentation: Fully Convolutional

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

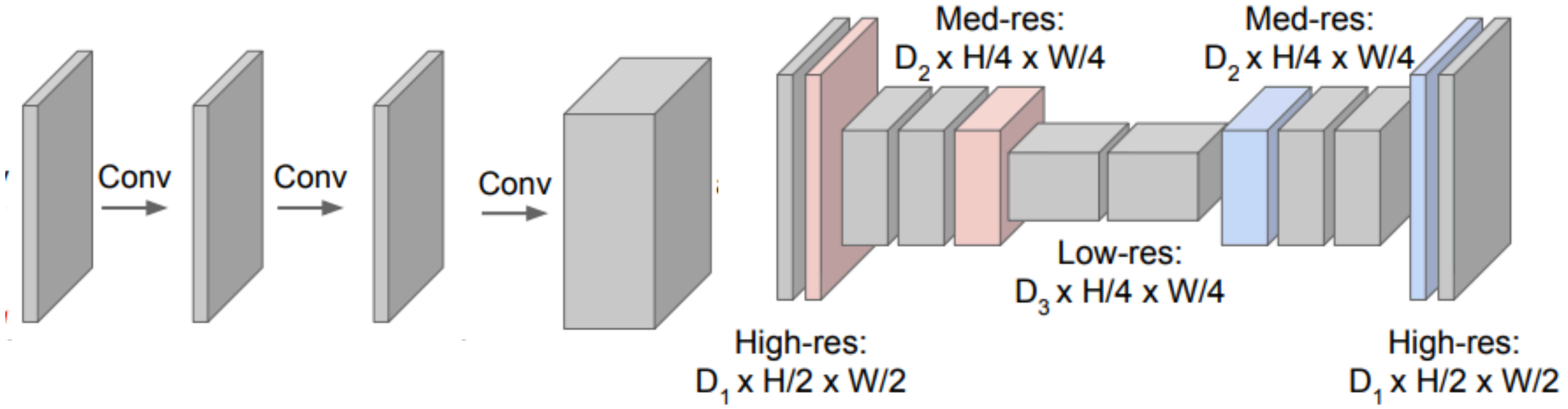
High-res:
D_1 x H/2 x W/2

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

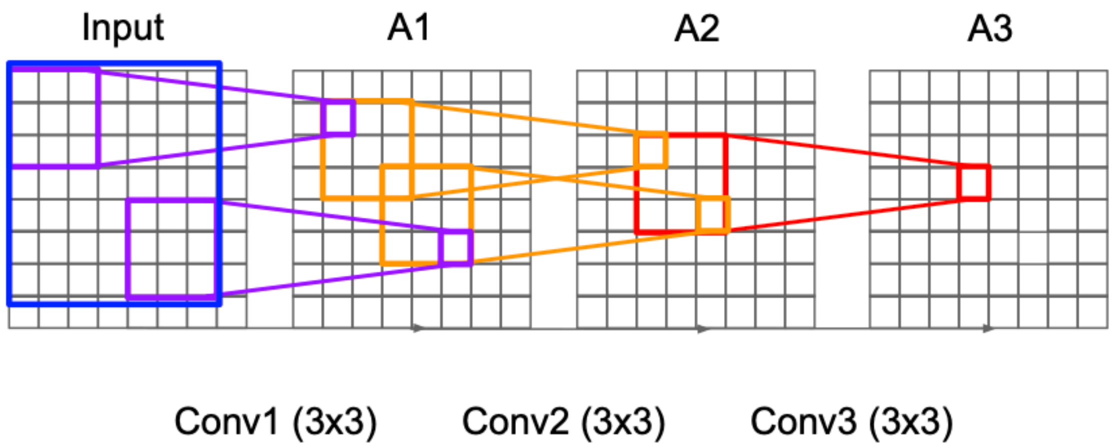
Advantage of Bottleneck

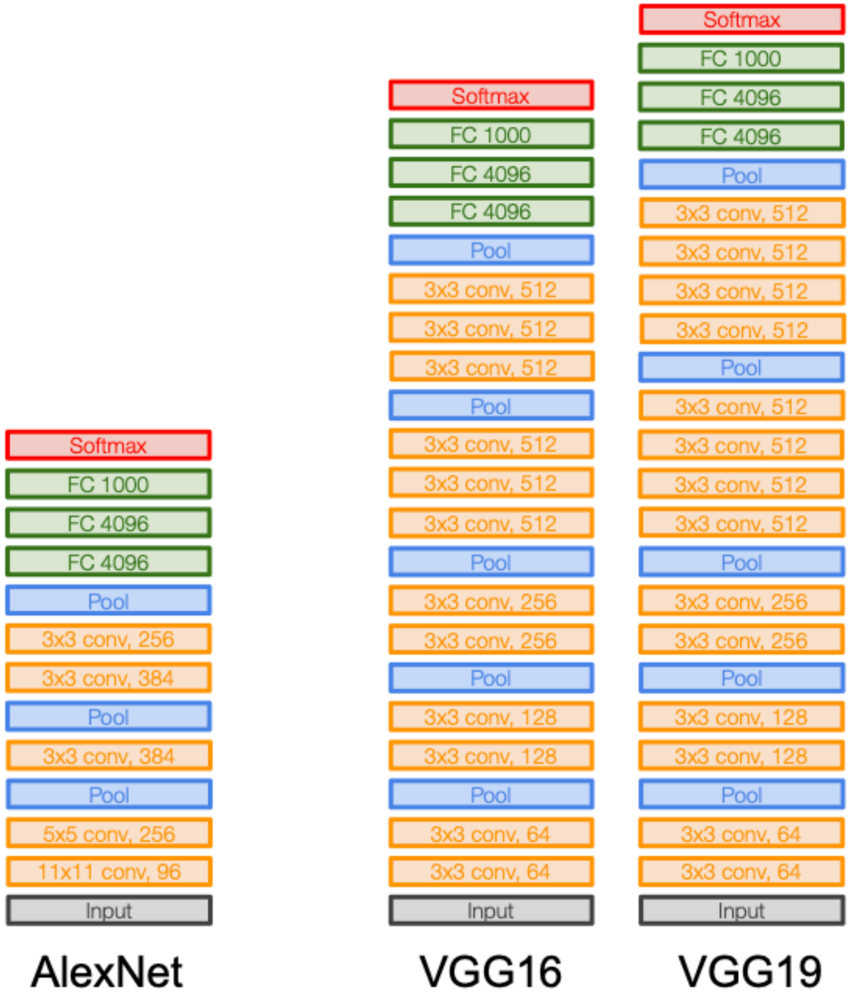
- Lower memory cost



Recap of Receptive Field

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

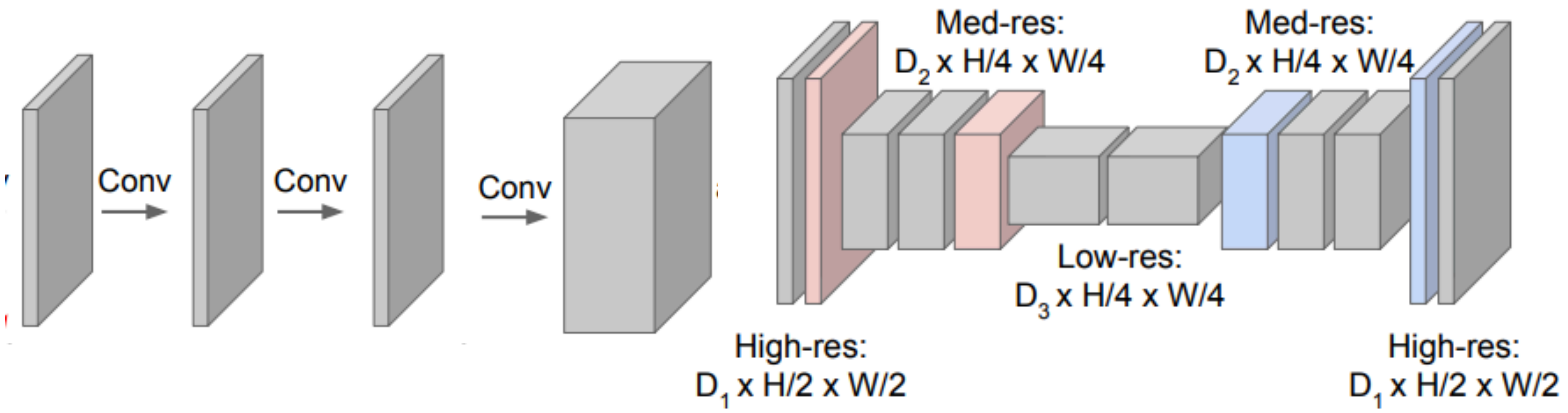




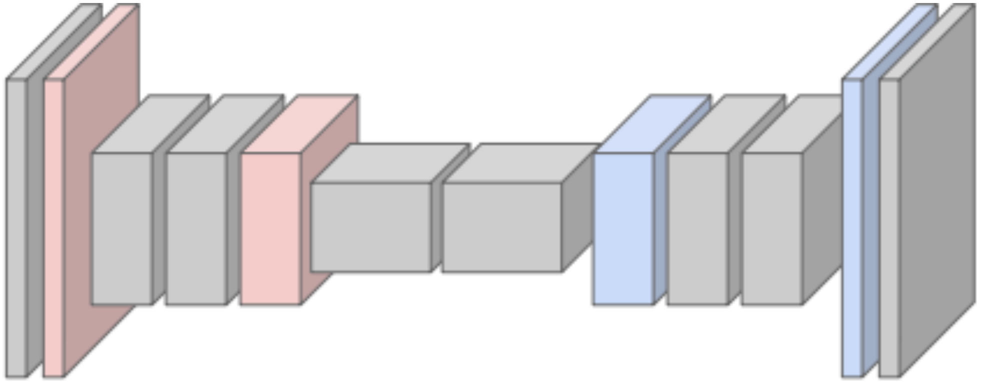
Advantage of Bottleneck

- 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

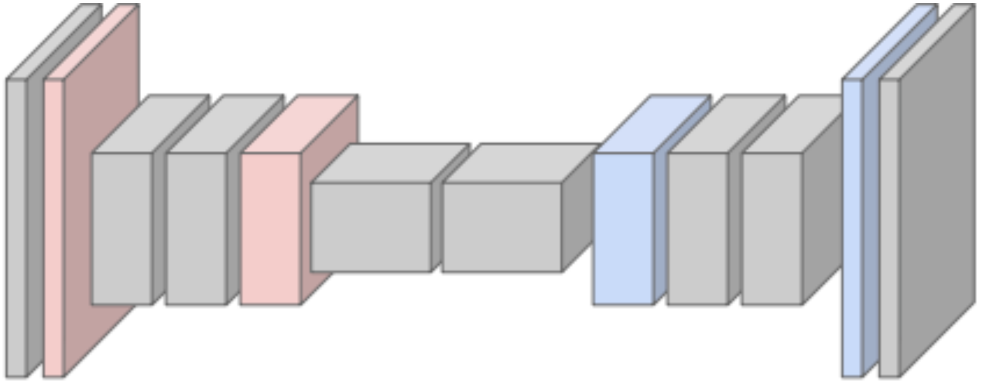


Improving FCN



What needs to be stored in the bottleneck?

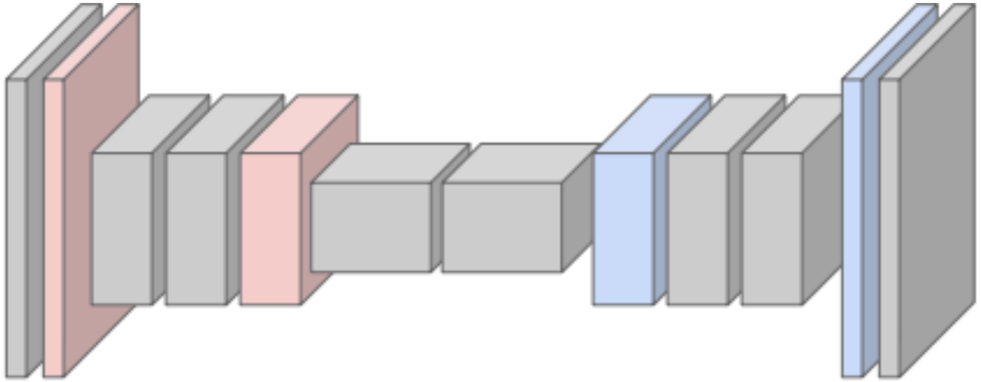
Improving FCN



What needs to be stored in the bottleneck?

- Global context

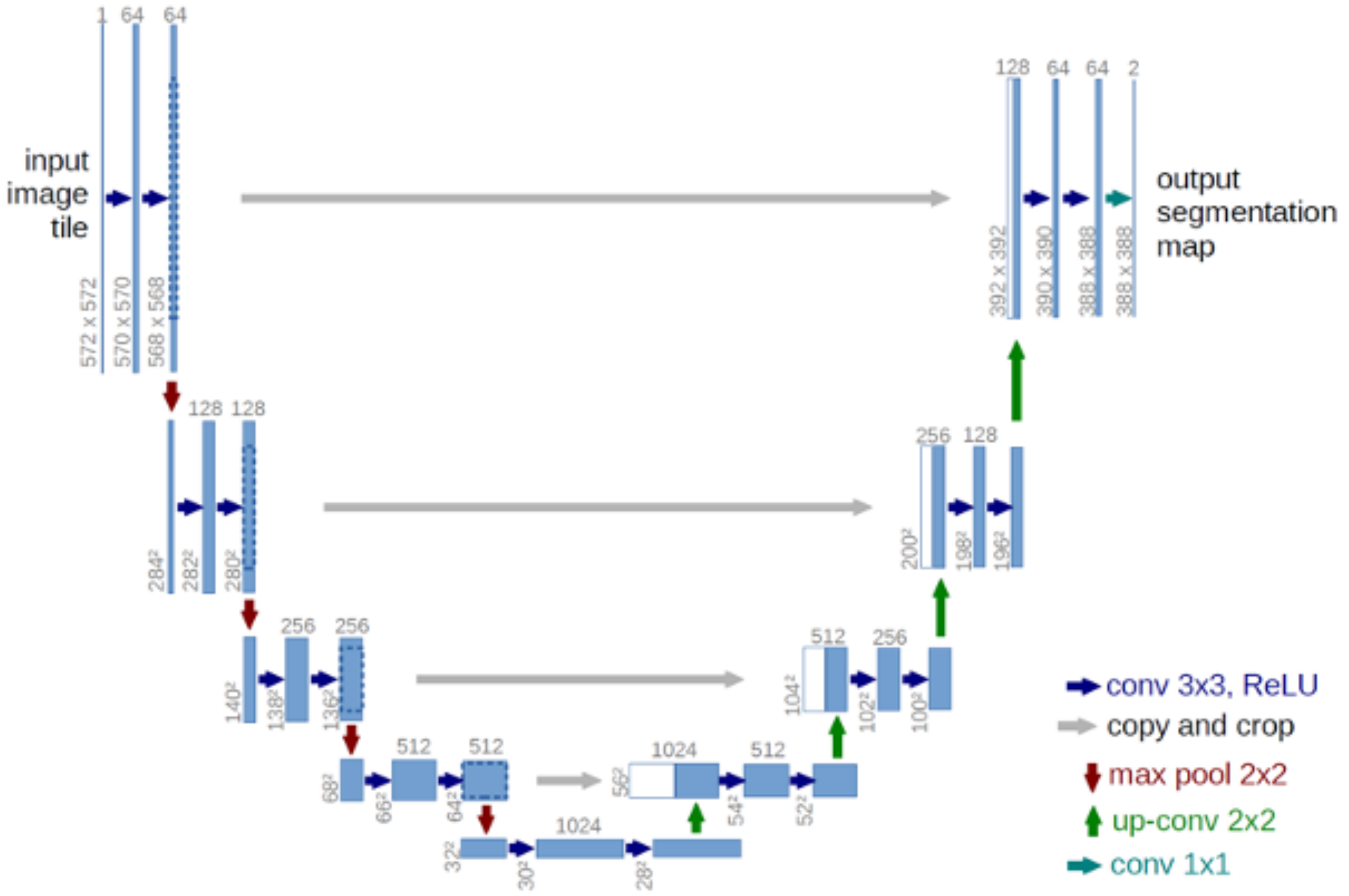
Improving FCN



What needs to be stored in the bottleneck?

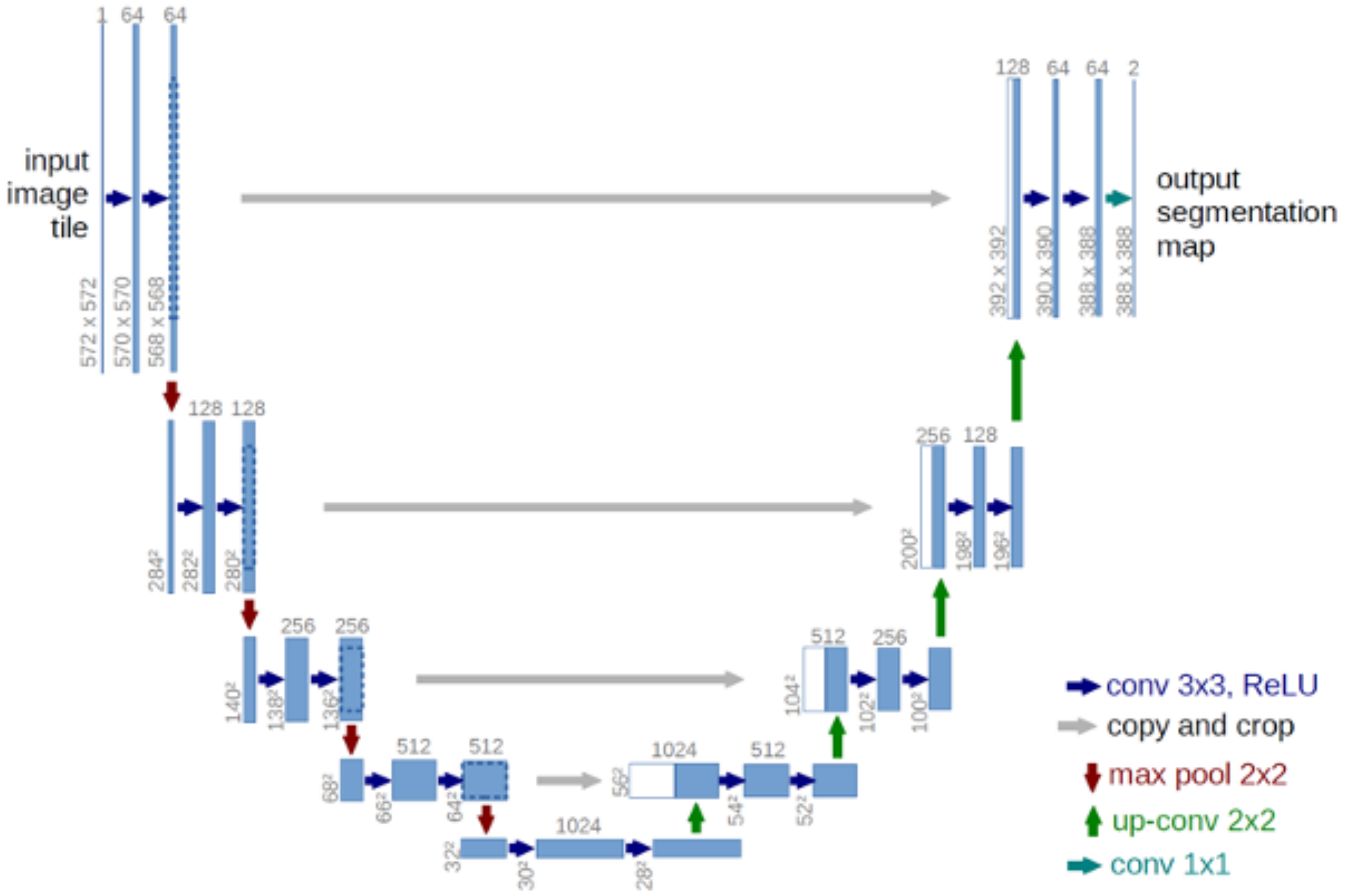
- Global context
- Per-pixel spatial information, especially around the boundary

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

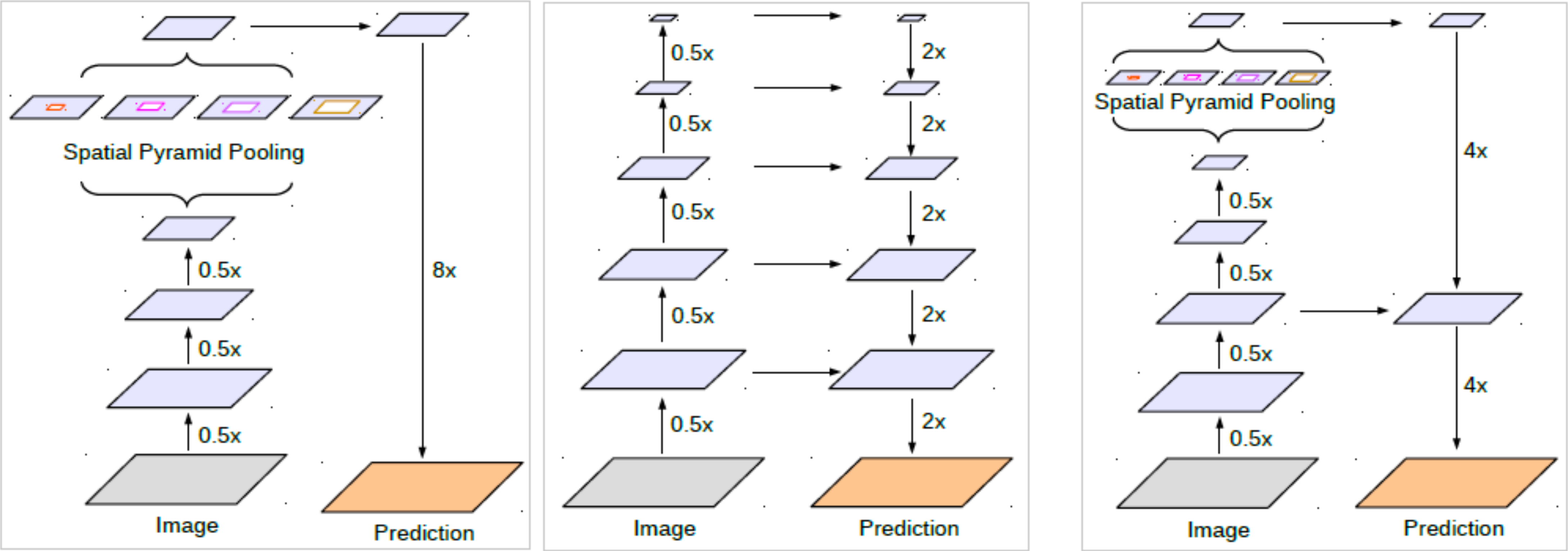
Summary of Semantic Segmentation

- A top-down approach

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

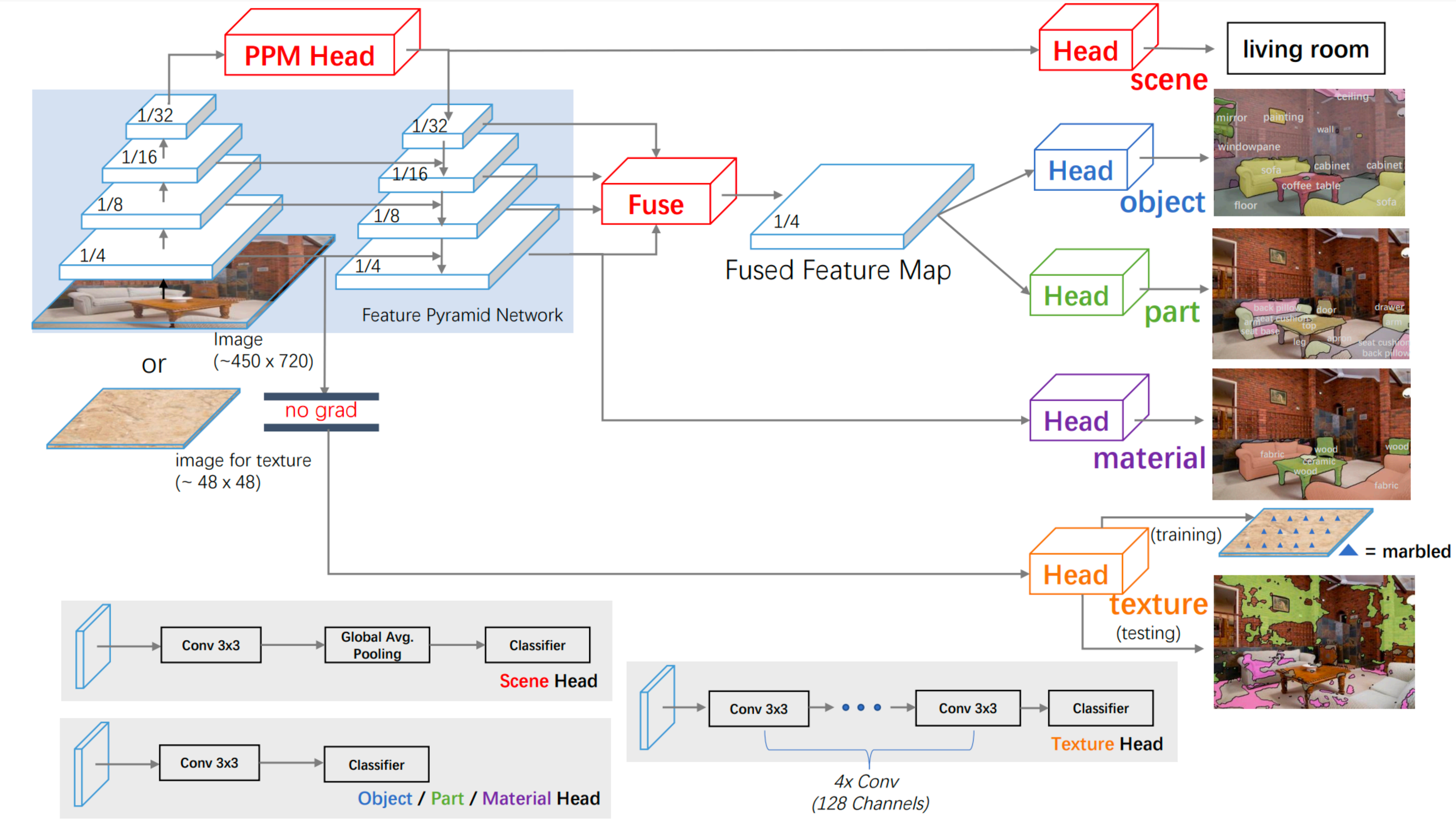
- Skip link:
  - Assist final segmentation
  - Avoid memorization

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.

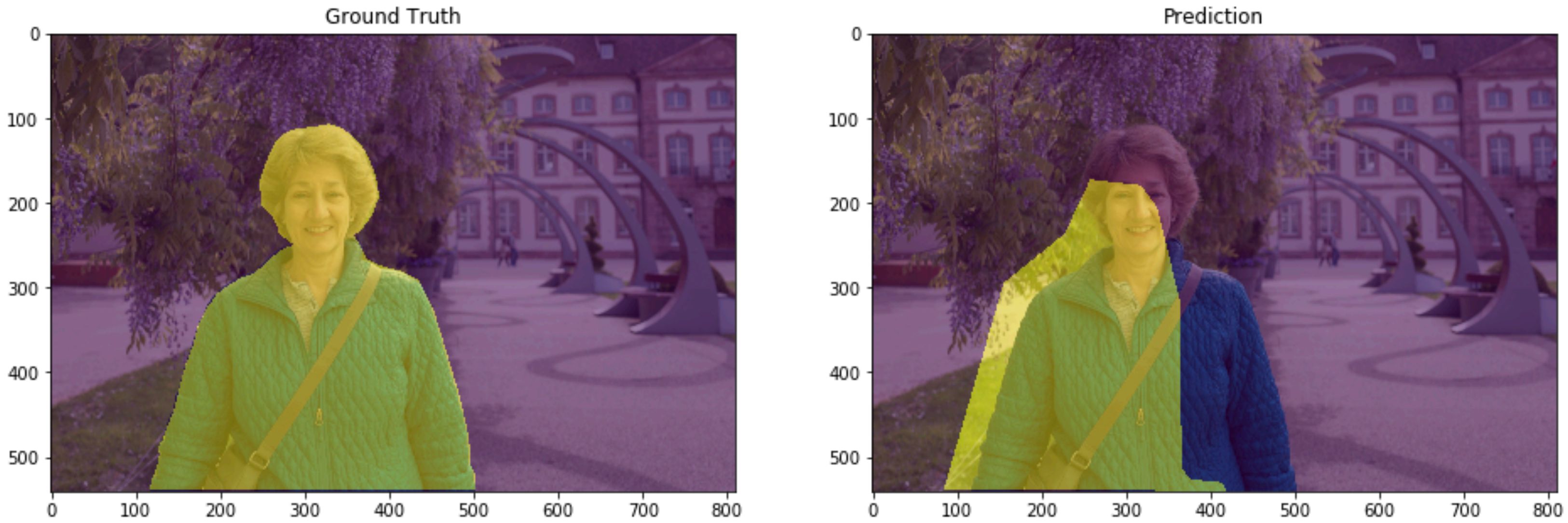
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- 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 9,
3D Vision I