Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 6: Convolutional Neural Networks (Part 3)

Logistics:

Assignment 2 is due on **Monday**

Computer **Vision Problems** (no language for now)

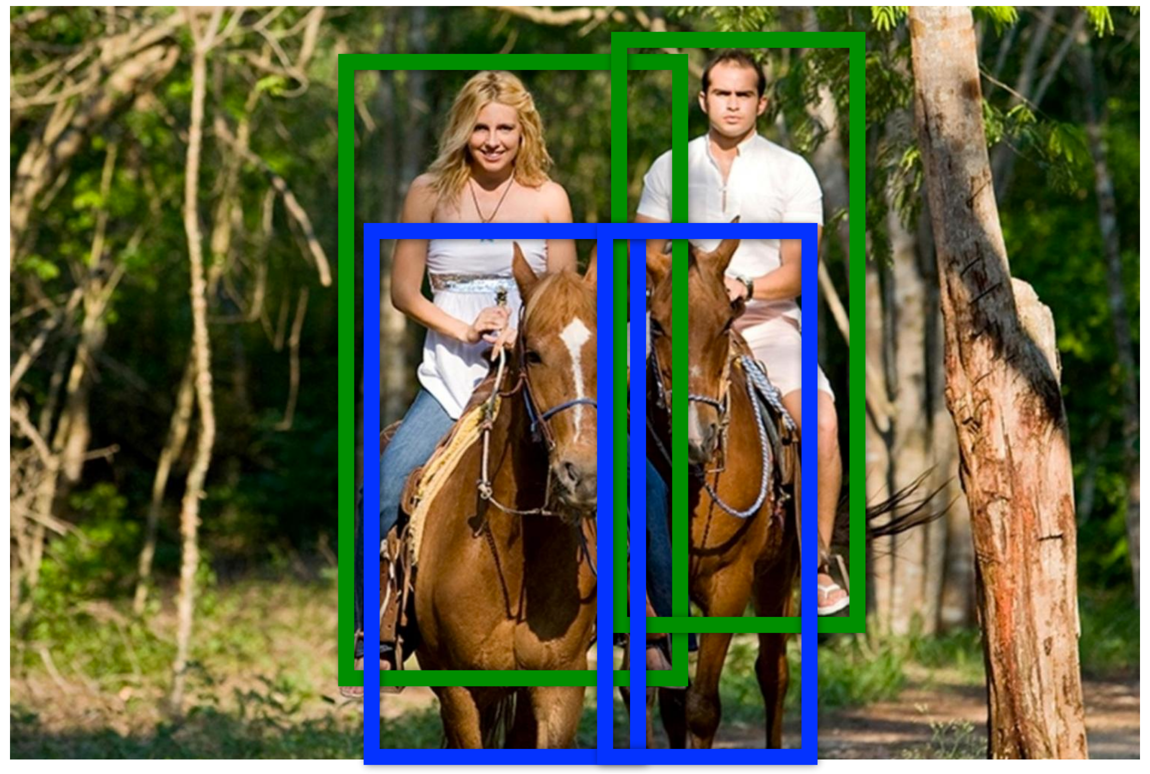
Categorization

Multi-**class**: Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

Segmentation

Horse
Person

COCO
Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2

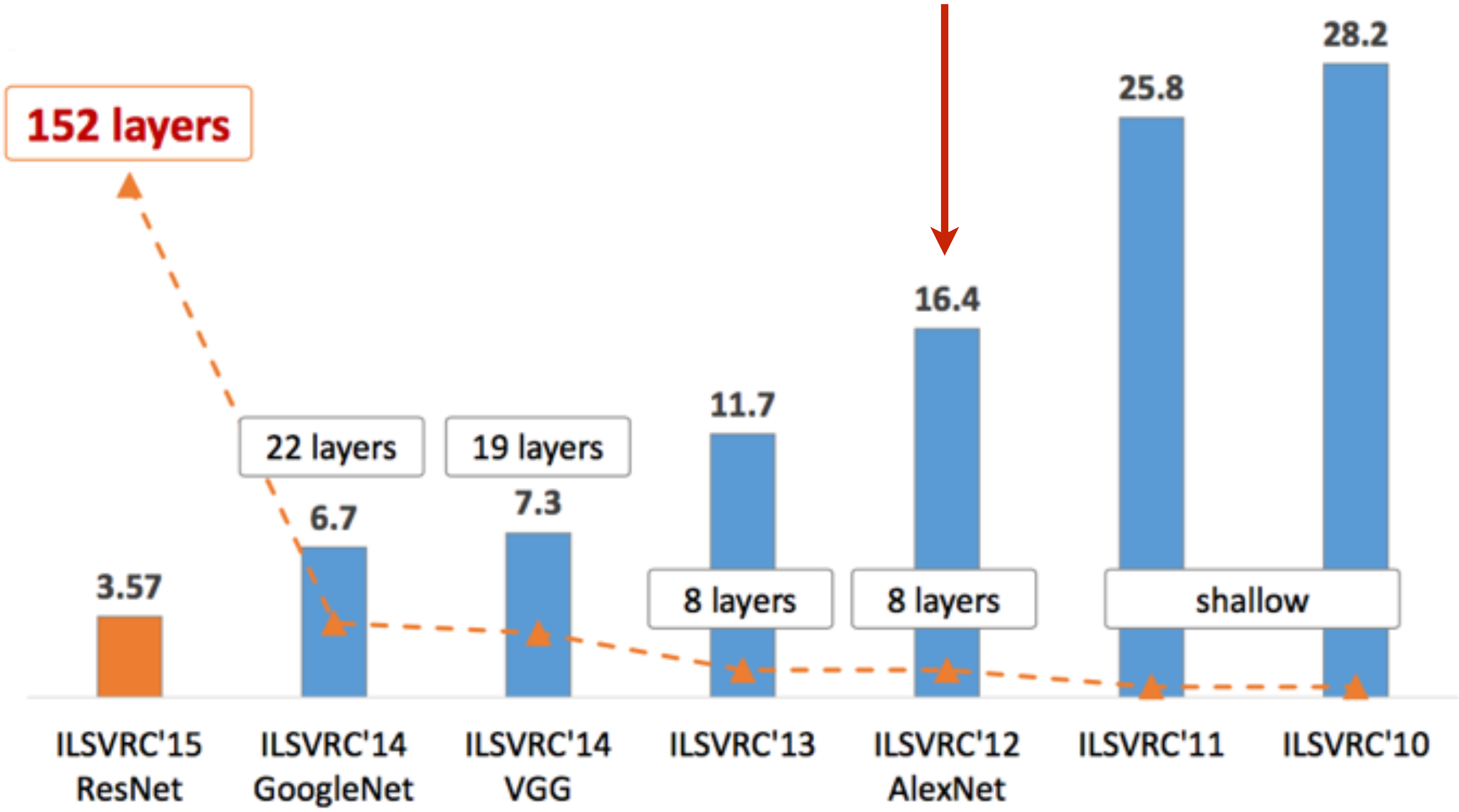
Computer **Vision Problems** (no language for now)

Categorization

Multi-**class:** Horse
Church
Toothbrush
Person

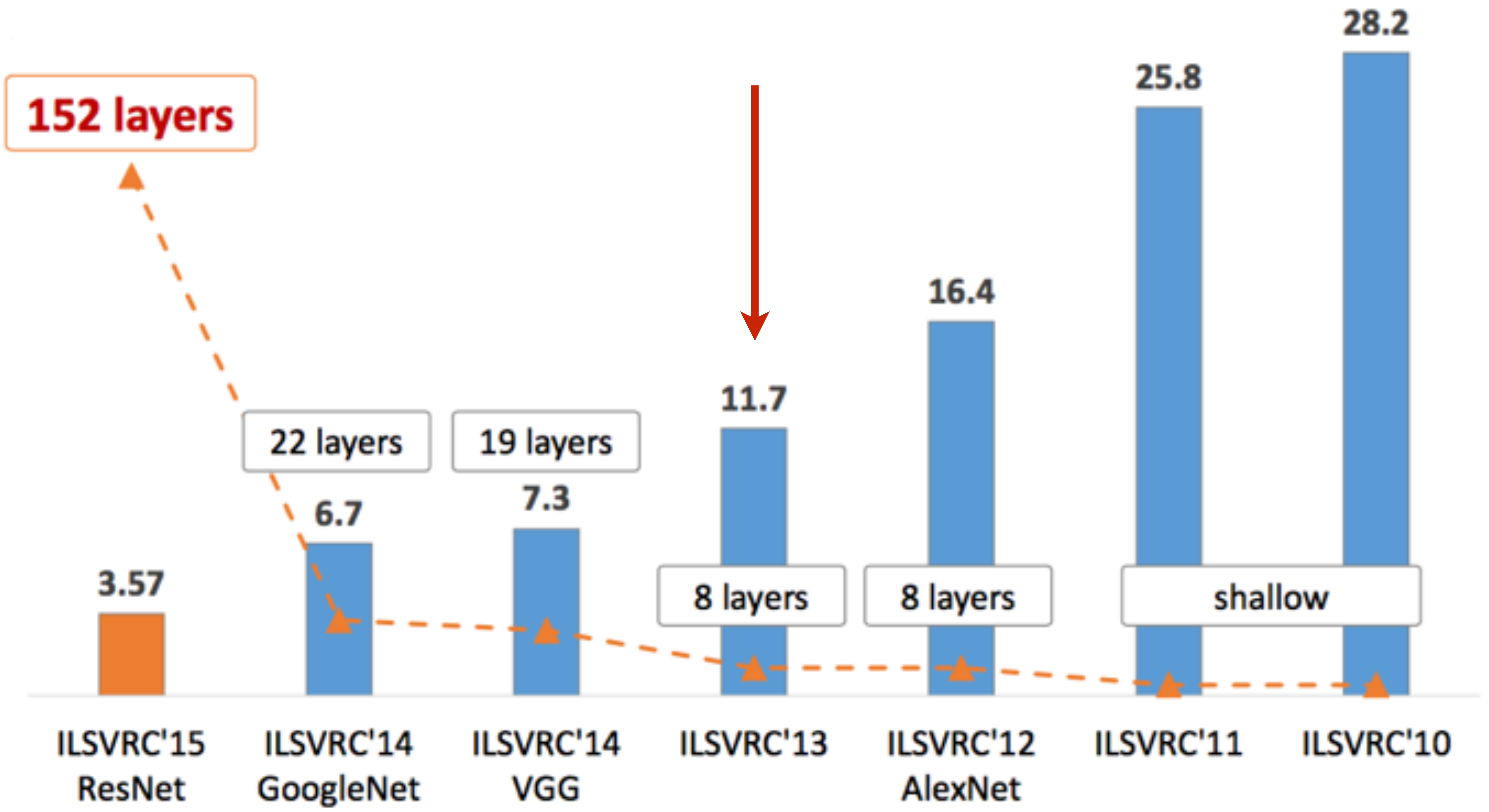
IMAGENET

ILSVRC winner 2012



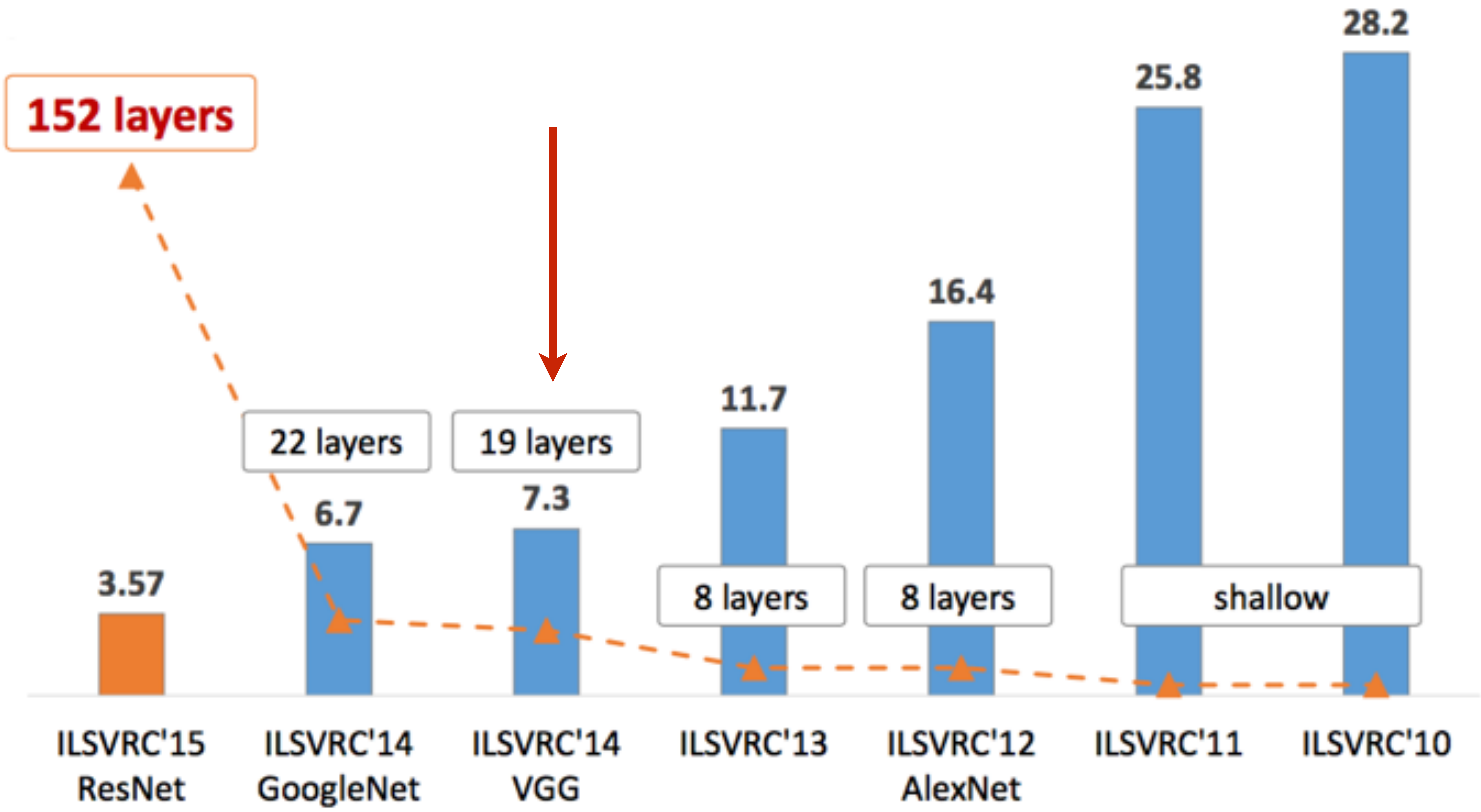
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ILSVRC winner 2012



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ILSVRC winner 2012



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

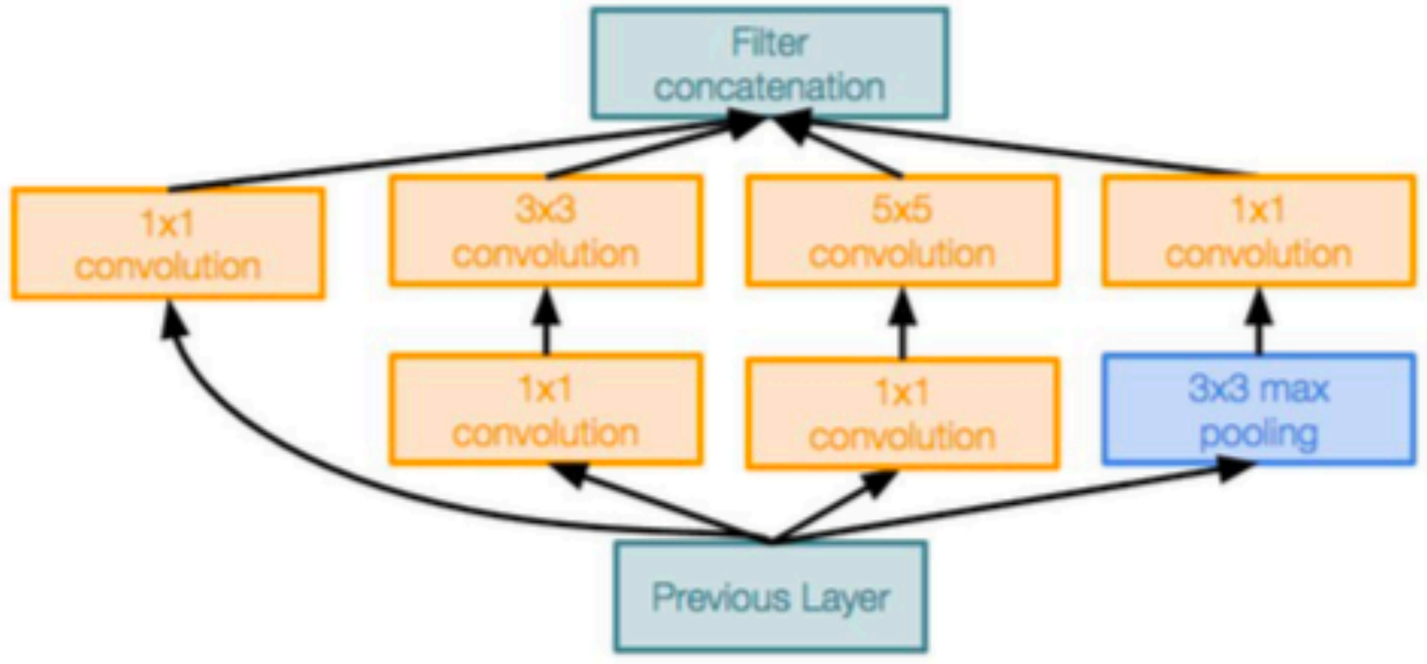
— Efficient “Inception” module

— No FC layers

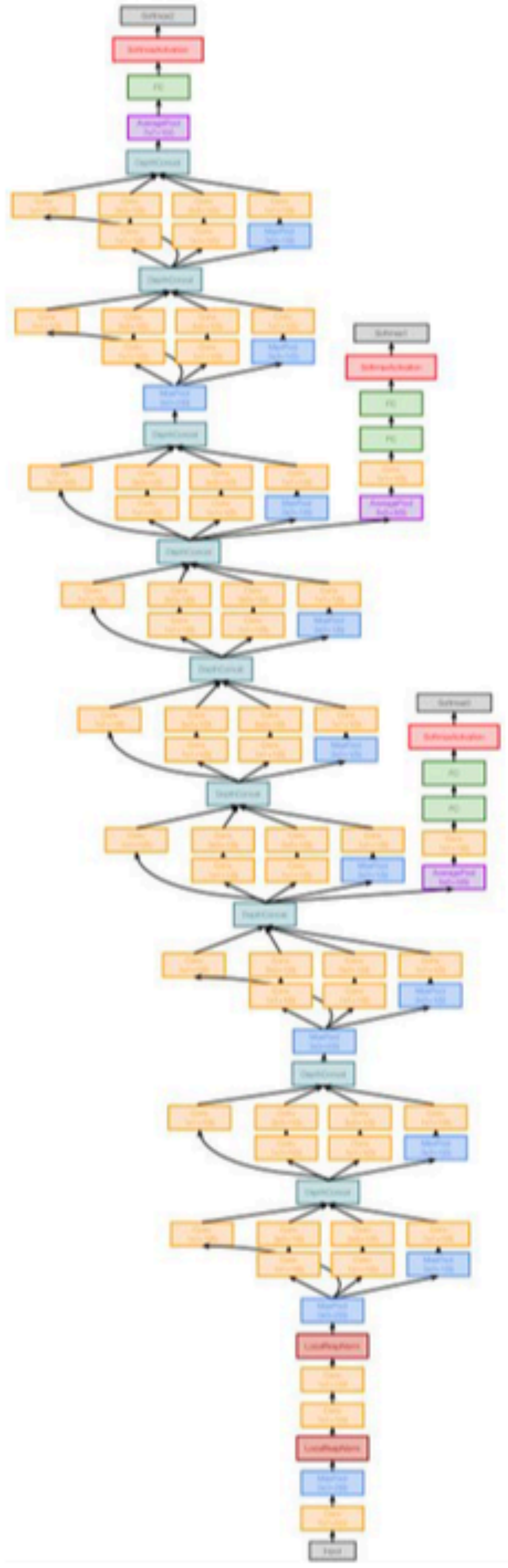
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)



Inception module

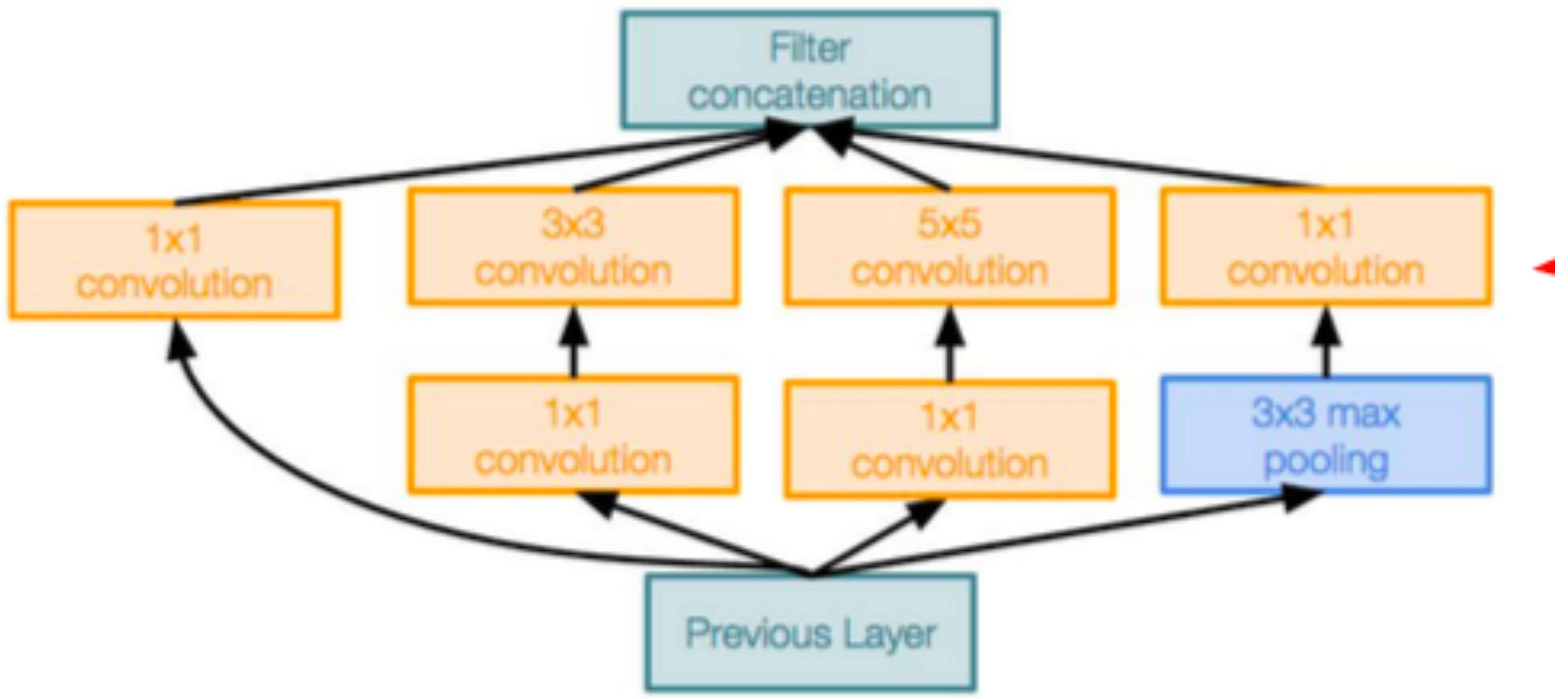


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

Idea: design good local topology ("network within network") and then stack these modules



Inception module

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

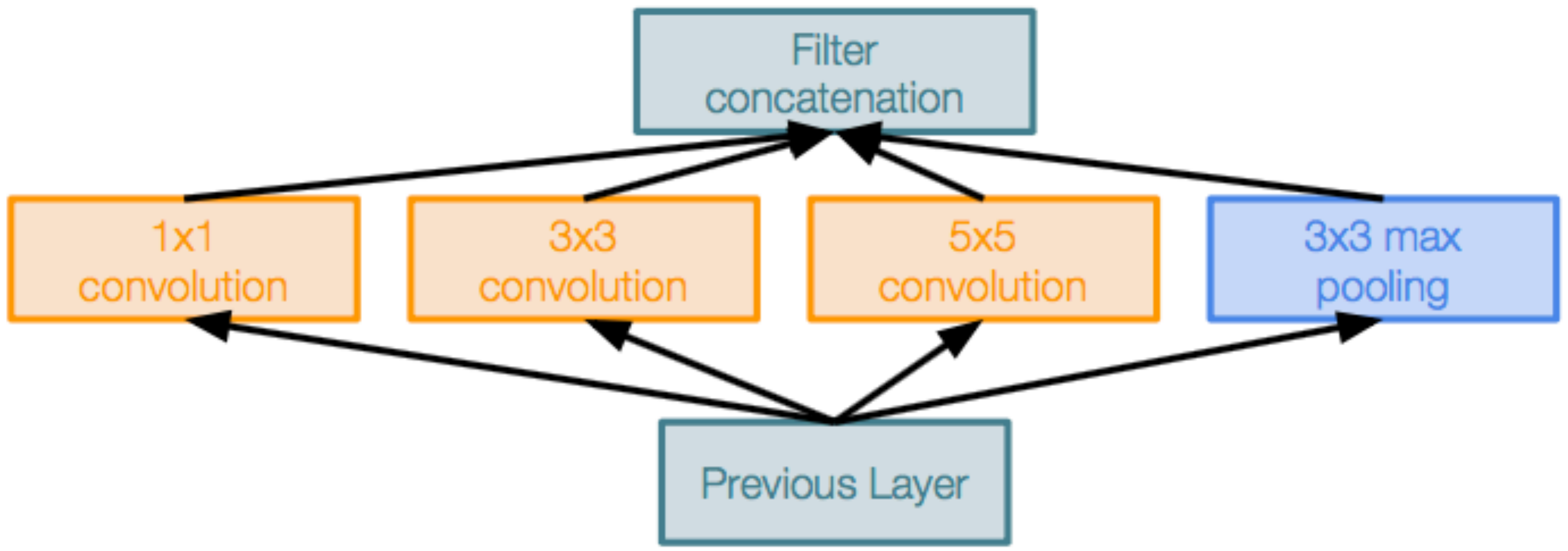
Idea: design good local topology ("network within network") and then stack these modules

Apply **parallel filter operations** on the input from previous layer

— Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)

— Pooling operation (3x3)

Concatenate all filter outputs together at output depth-wise



Naive Inception module

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

Idea: design good local topology ("network within network") and then stack these modules

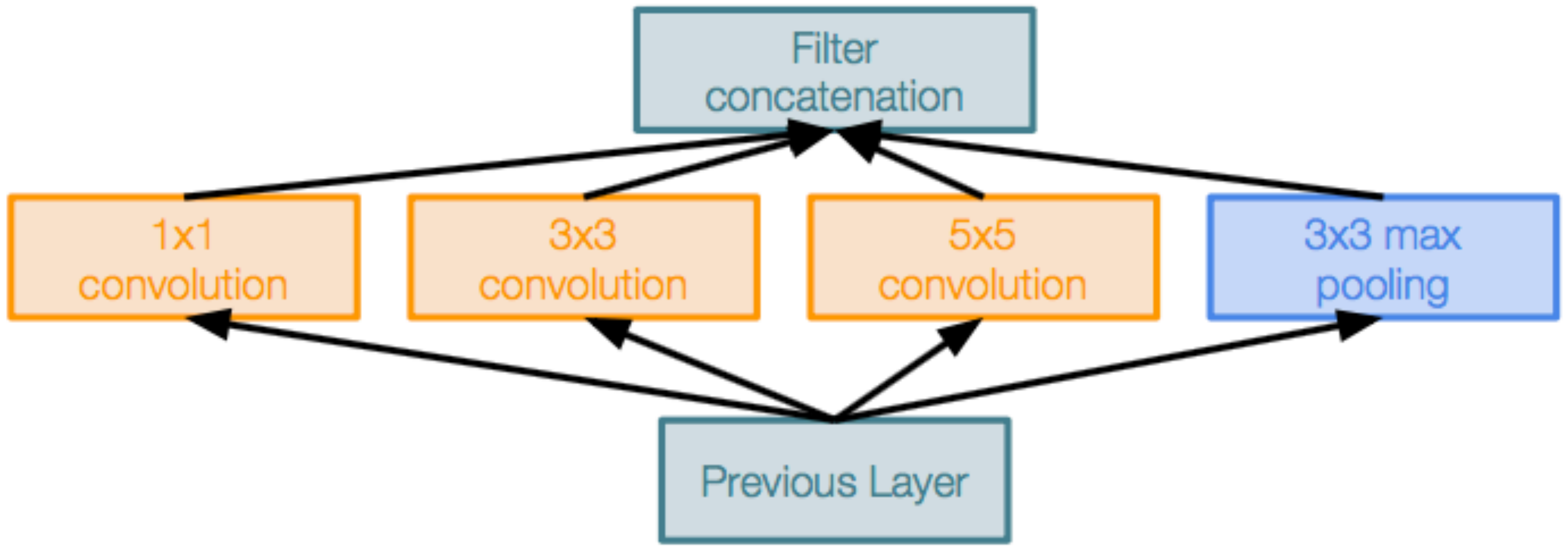
Apply **parallel filter operations** on the input from previous layer

— Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)

— Pooling operation (3x3)

Concatenate all filter outputs together at output depth-wise

What's the problem?



Naive Inception module

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

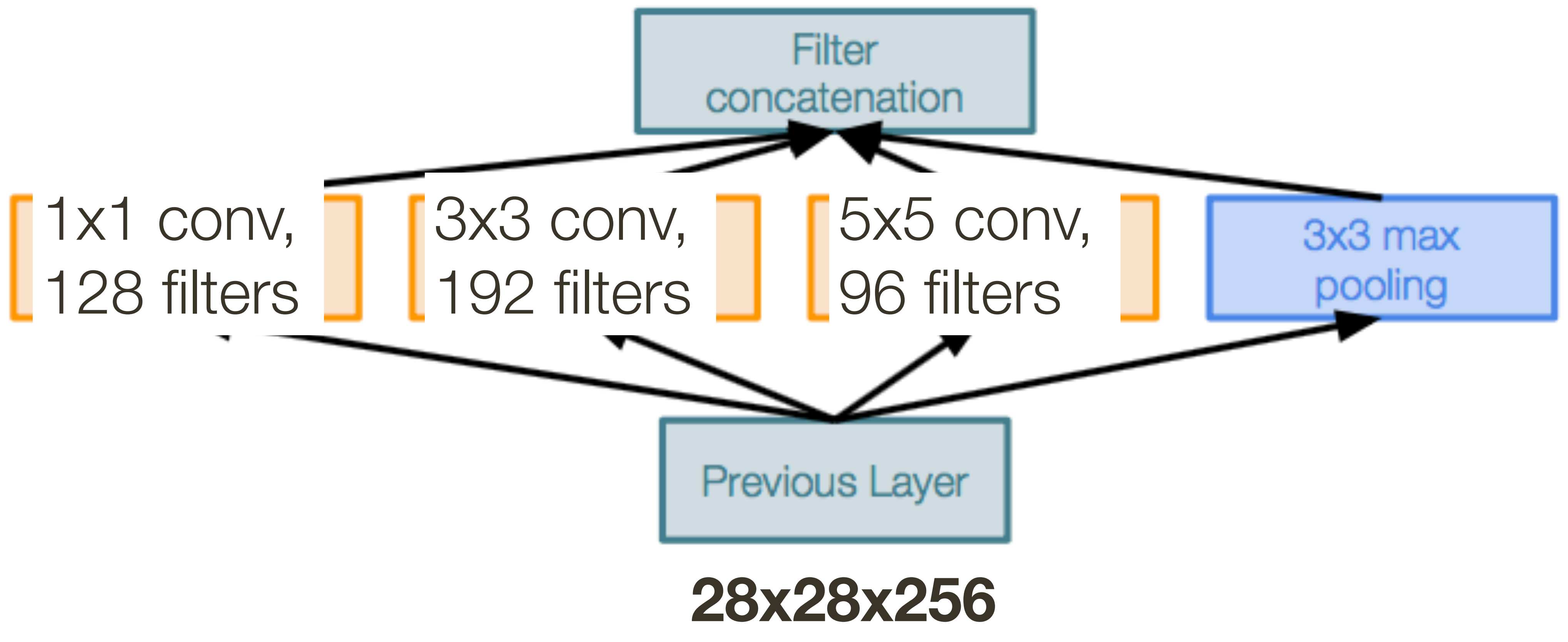
Idea: design good local topology ("network within network") and then stack these modules

Apply **parallel filter operations** on the input from previous layer

— Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)

— Pooling operation (3x3)

Concatenate all filter outputs together at output depth-wise



Naive Inception module

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

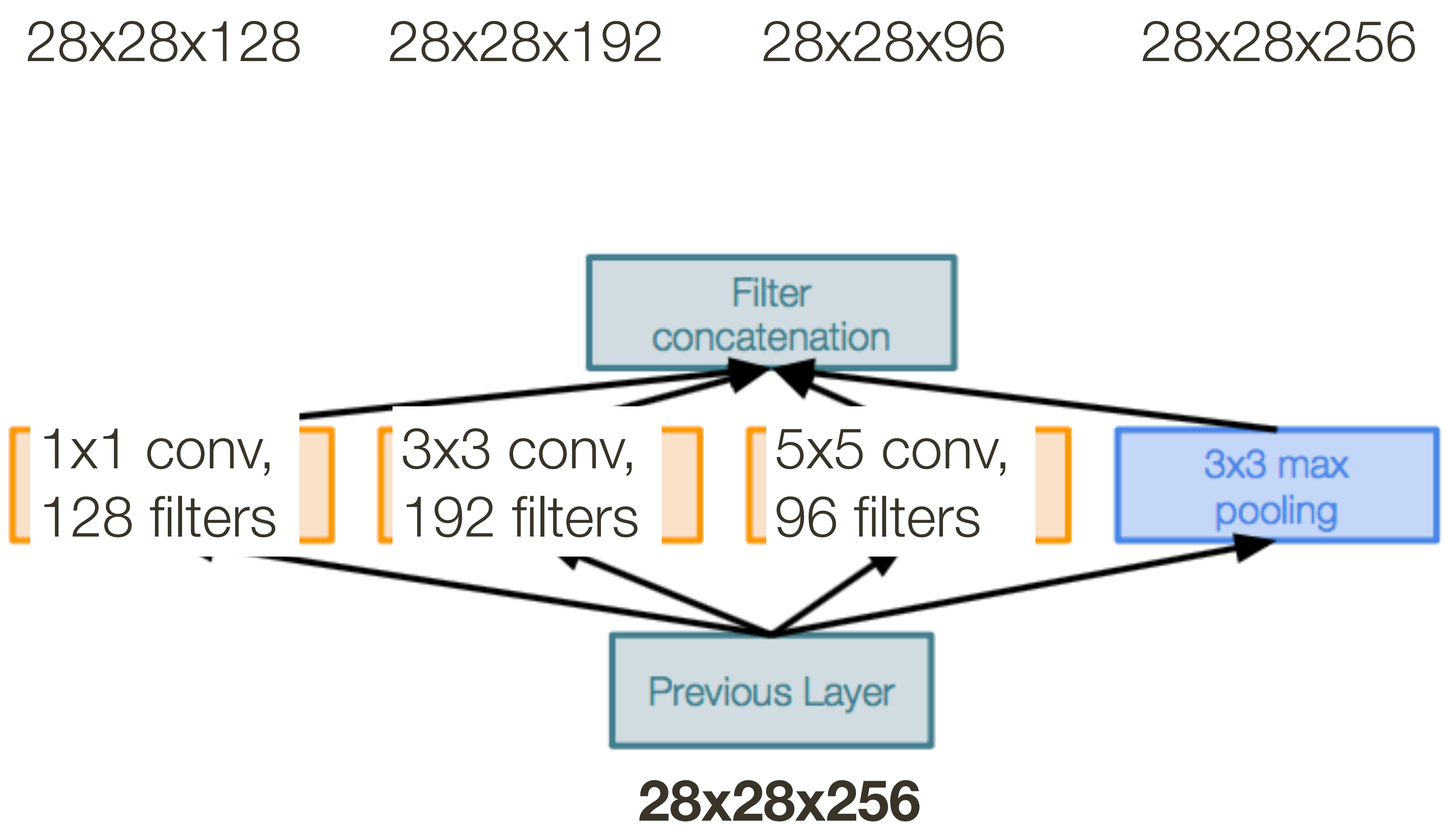
Idea: design good local topology ("network within network") and then stack these modules

Apply **parallel filter operations** on the input from previous layer

— Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)

— Pooling operation (3x3)

Concatenate all filter outputs together at output depth-wise



Naive Inception module

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

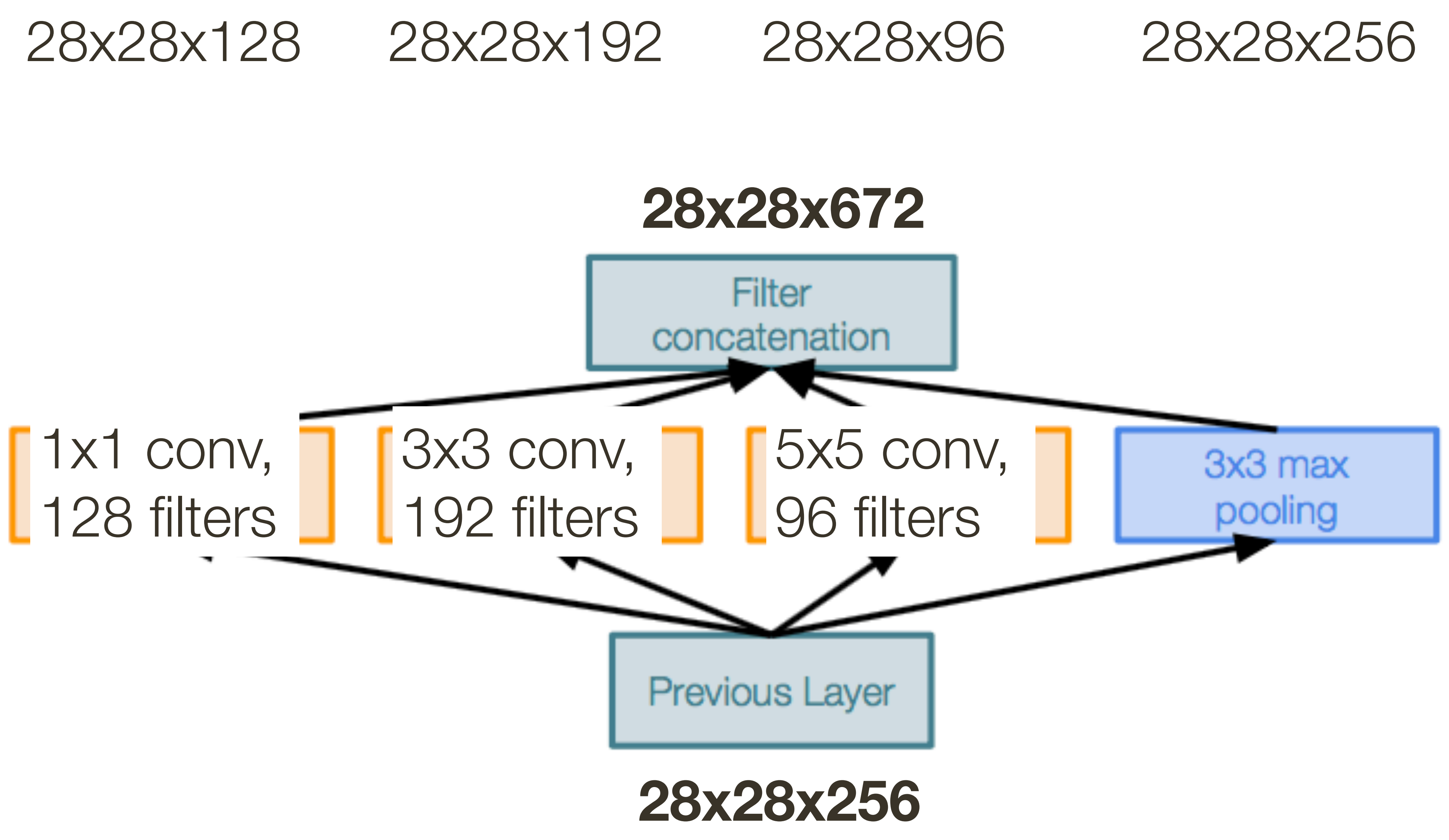
Idea: design good local topology ("network within network") and then stack these modules

Apply **parallel filter operations** on the input from previous layer

— Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)

— Pooling operation (3x3)

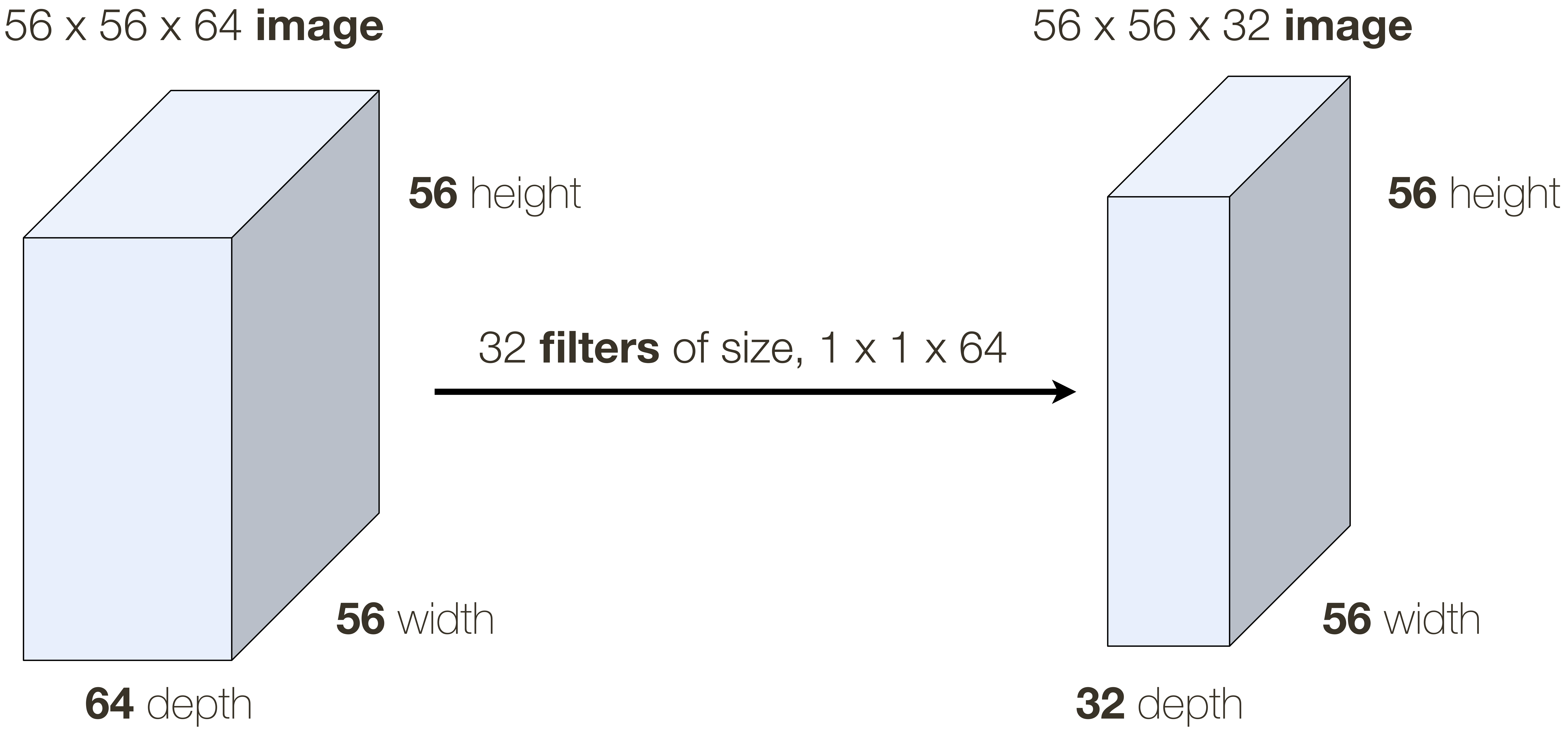
Concatenate all filter outputs together at output depth-wise



Naive Inception module

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer: **1x1 convolutions**



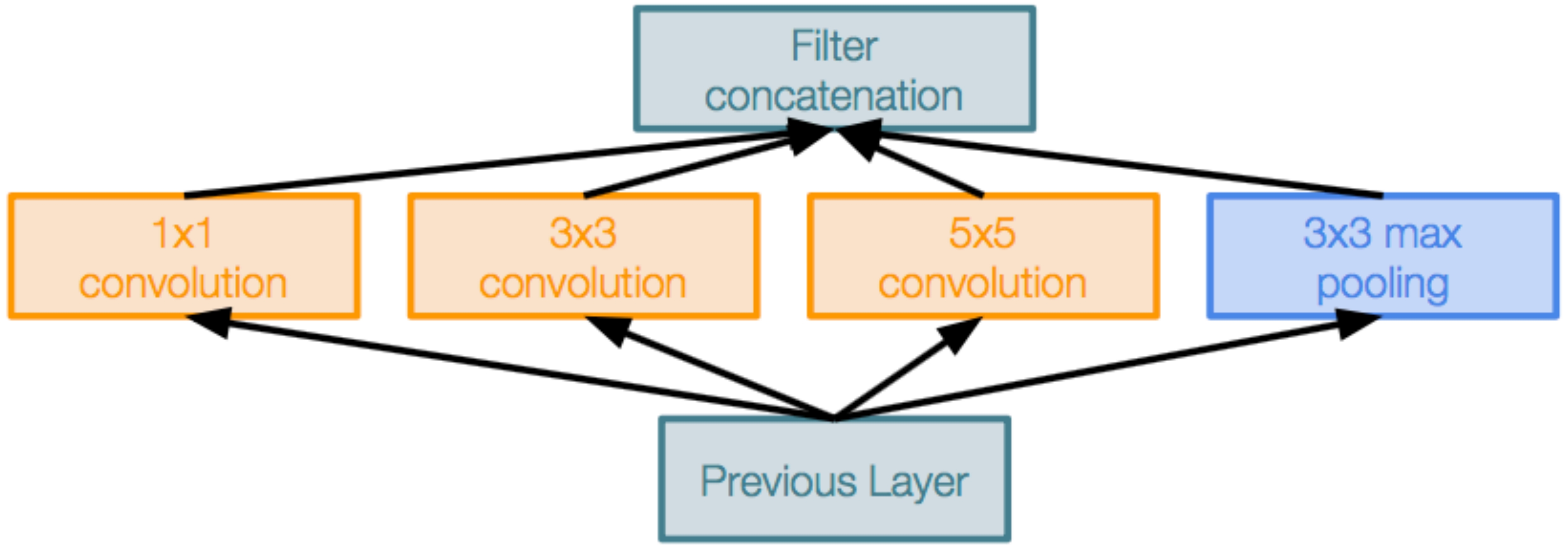
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet: Inception Module

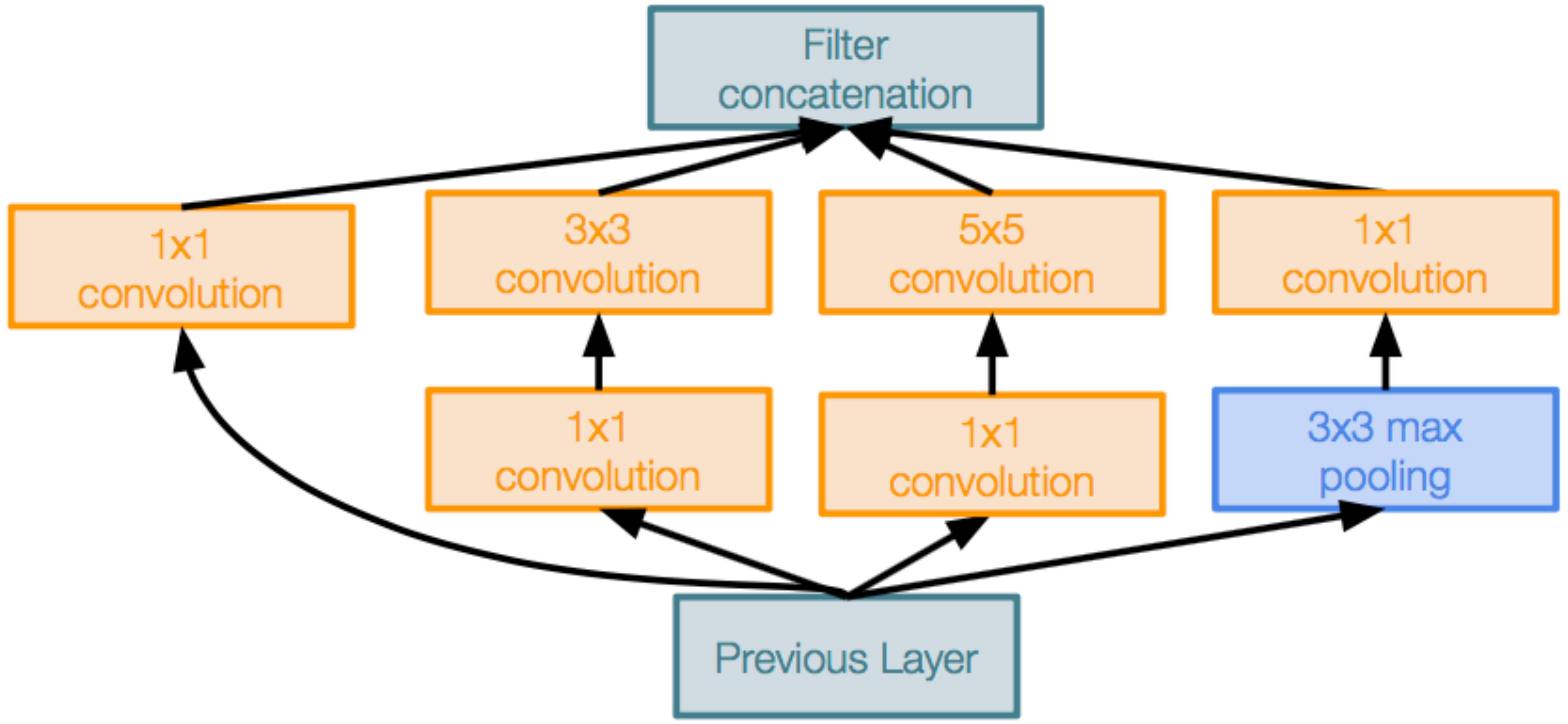
[Szegedy et al., 2014]

Idea: design good local topology ("network within network") and then stack these modules

1x1 "bottleneck" layers



Naive Inception module



Inception module with dimension reduction

saves approximately 60% of computations

GoogleLeNet

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

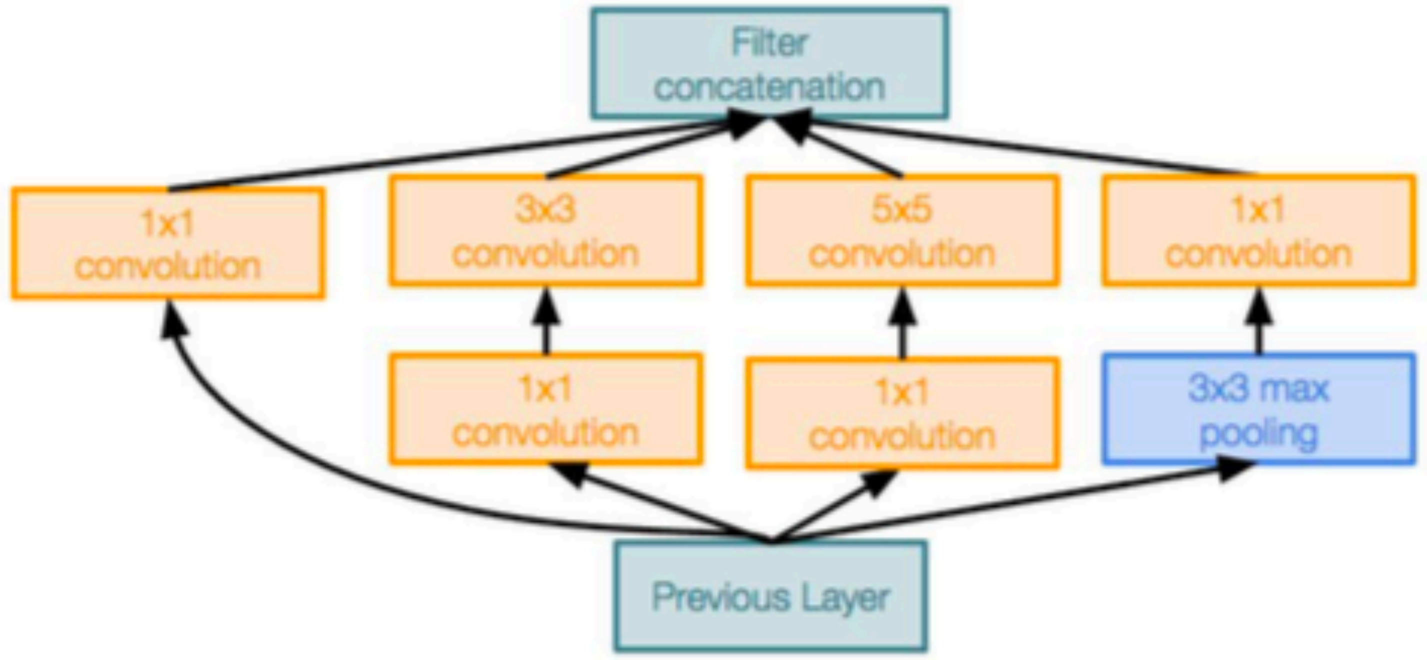
— Efficient “Inception” module

— No FC layers

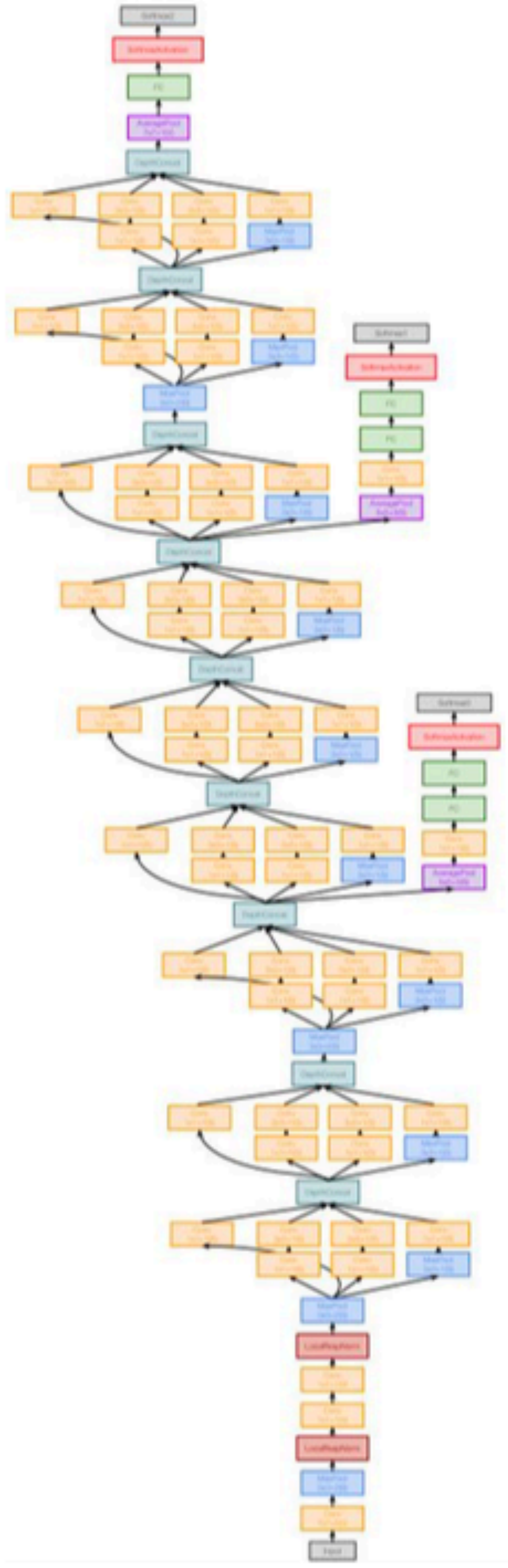
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)



Inception module



GoogleLeNet

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

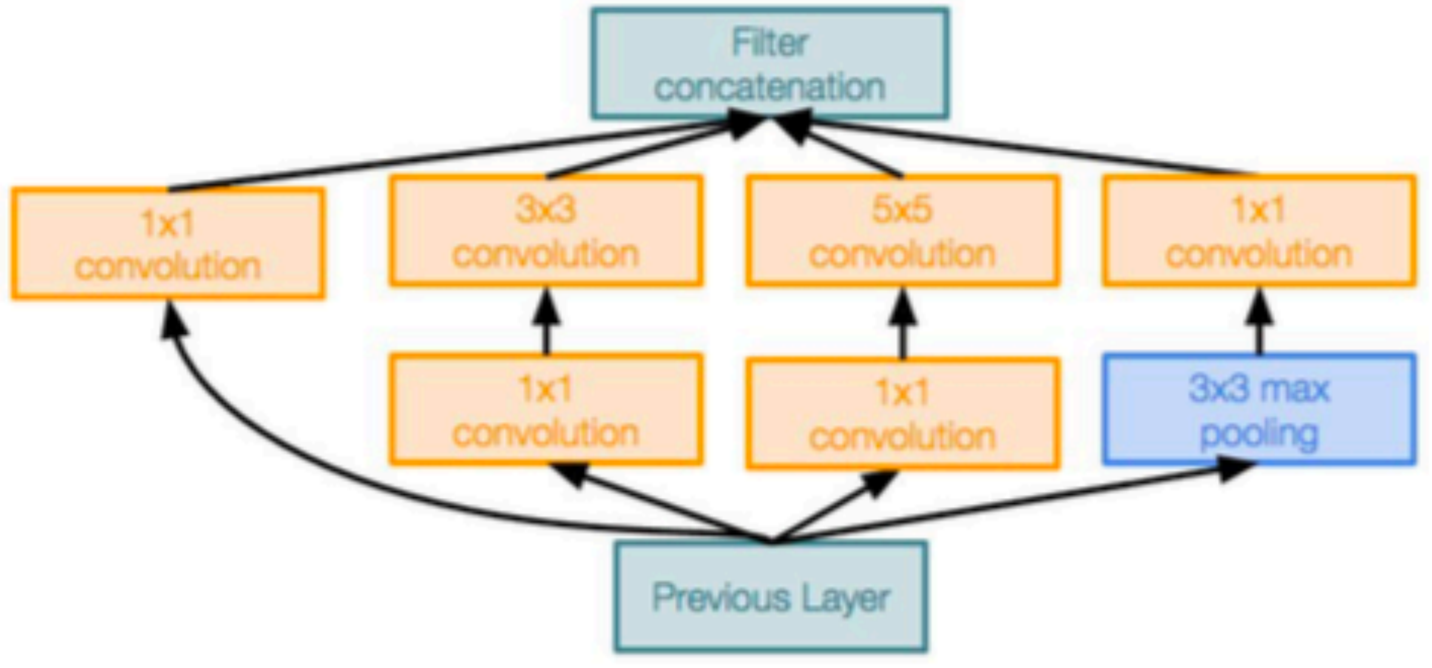
— Efficient "Inception" module

— No FC layers

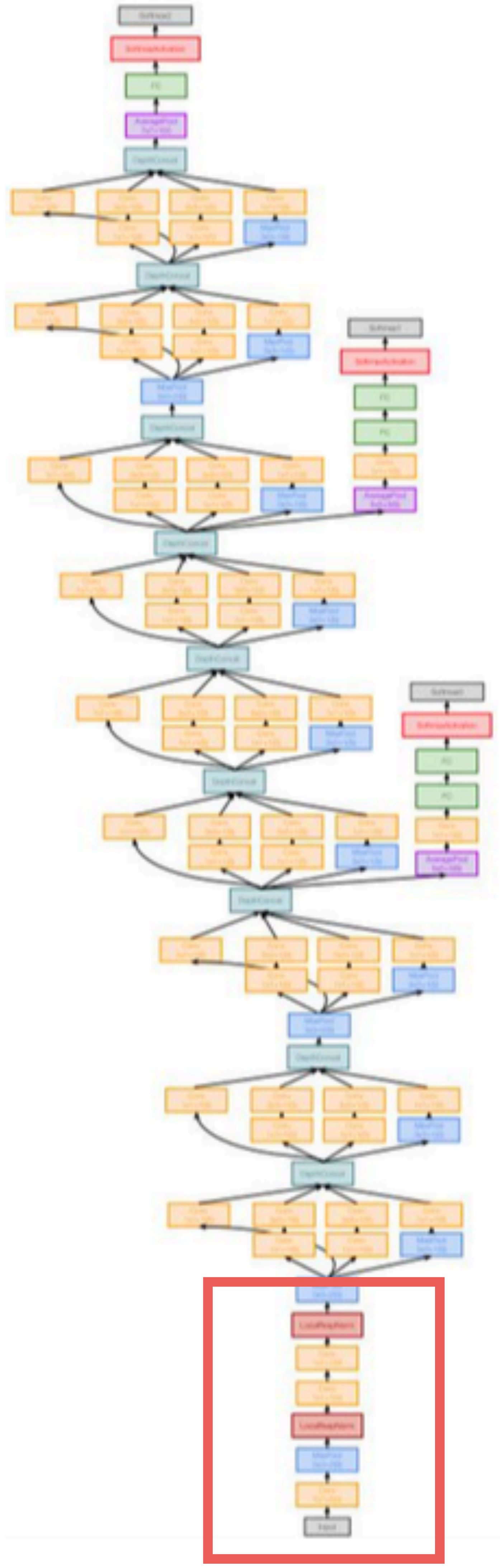
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)



Inception module



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

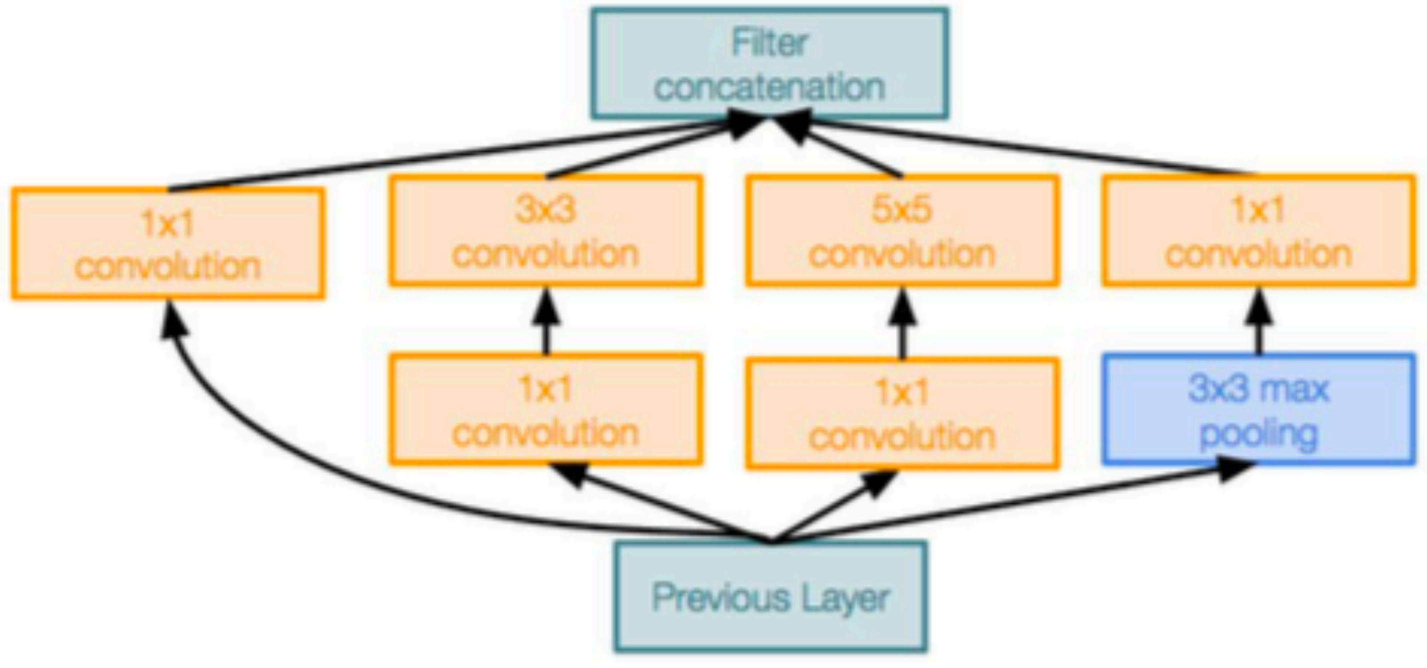
— Efficient “Inception” module

— No FC layers

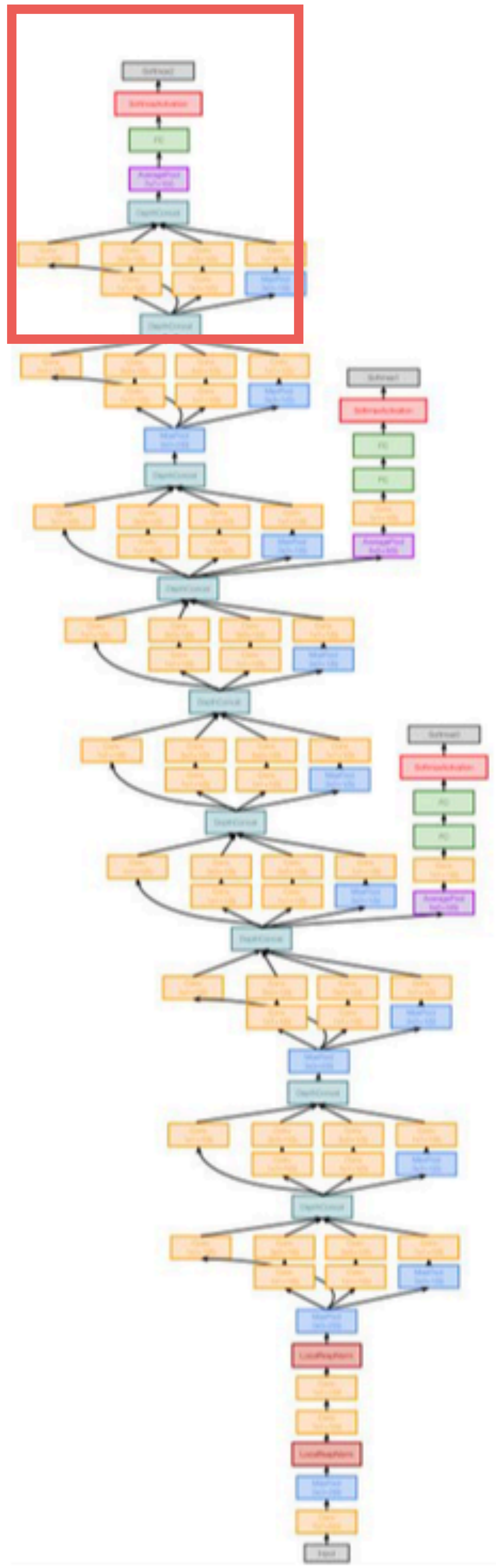
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)



Inception module



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

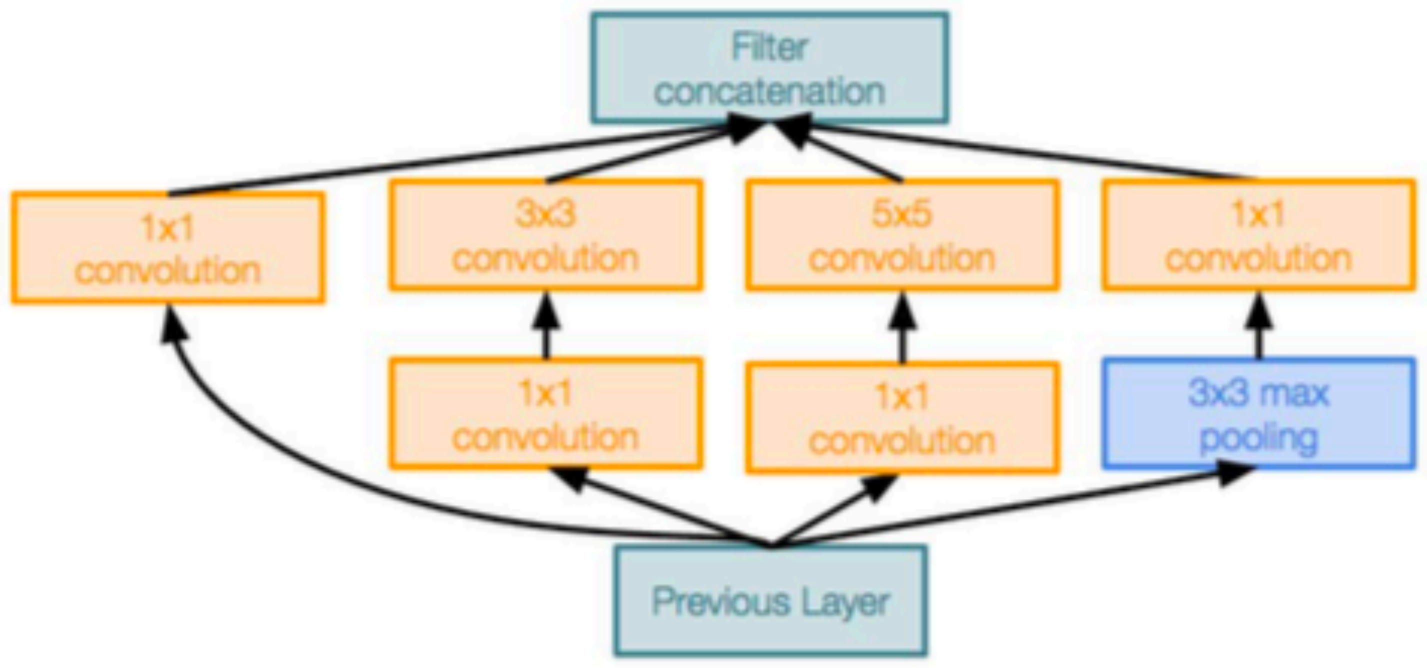
— Efficient “Inception” module

— No FC layers

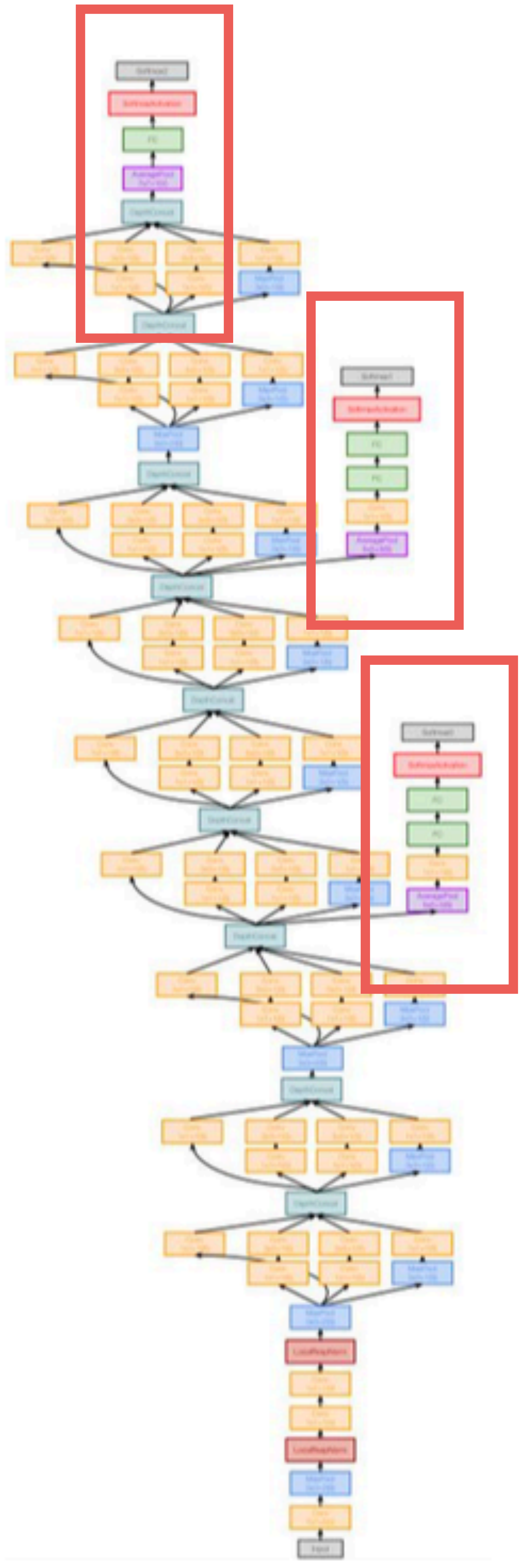
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)

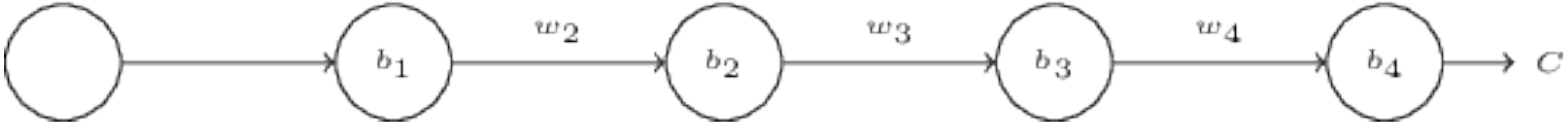


Inception module



Optimizing **Deep** Neural Networks

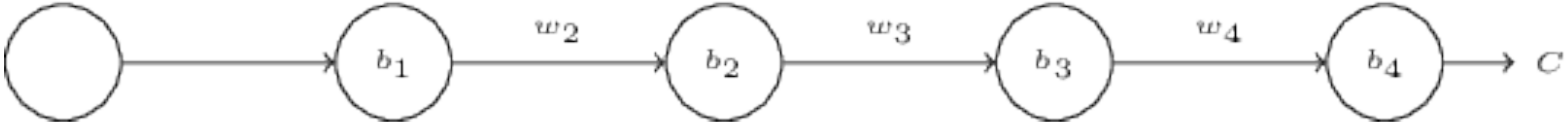
Consider multi-layer neural network with sigmoid activations and loss C



Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

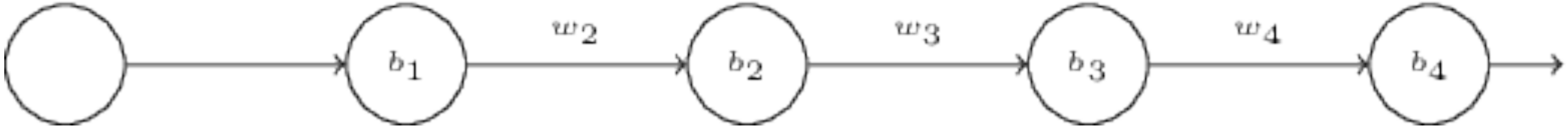
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



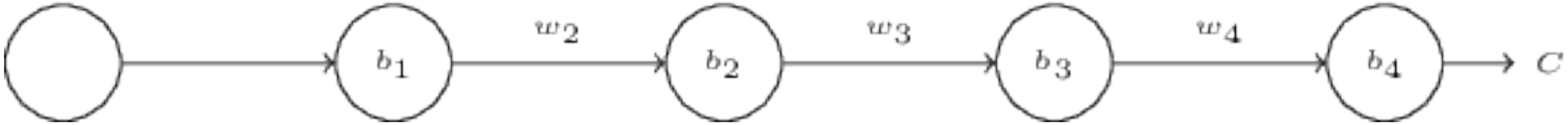
Expression for **gradient** of bias in **Layer 1**: $\frac{\partial C}{\partial b_1} = \sigma'(z_1)\, w_2 \sigma'(z_2)\, w_3 \sigma'(z_3)\, w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4}$

Expression for **gradient** of bias in **Layer 3**: $\frac{\partial C}{\partial b_3} = \sigma'(z_3)\, w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4}$

Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



Expression for **gradient** of bias in **Layer 1**: $\frac{\partial C}{\partial b_1} = \sigma'(z_1)\, w_2\sigma'(z_2)\, w_3\sigma'(z_3)\, \underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$

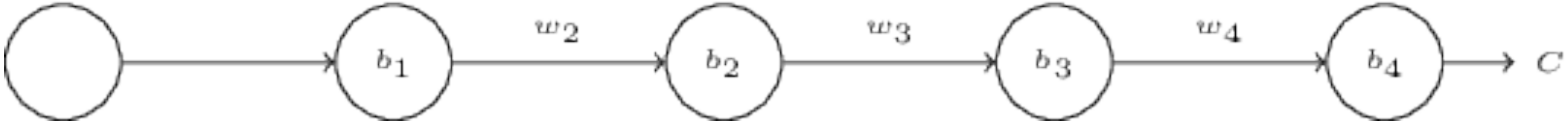
common terms

Expression for **gradient** of bias in **Layer 3**: $\frac{\partial C}{\partial b_3} = \sigma'(z_3)\, \overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$

Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\, w_2\sigma'(z_2)\, w_3\sigma'(z_3)\, \underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\, \overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

Observations:

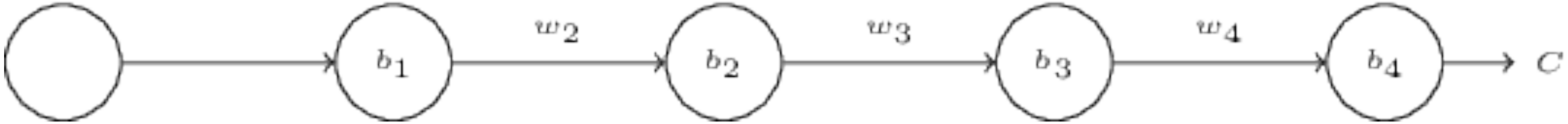
|weight| < 1 (due to initialization)

max of derivative of sigmoid = 1/4 @ 0

Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\,\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\,\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\,\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\,\overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

Observations:

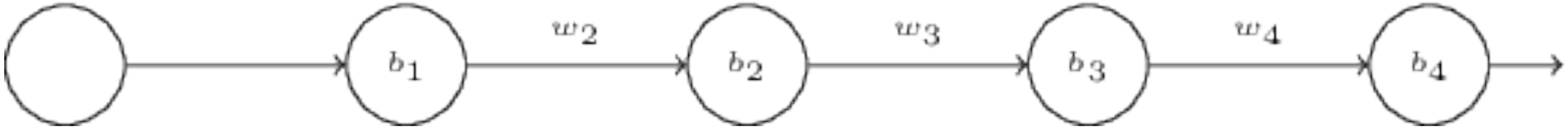
|weight| < 1 (due to initialization)

max of derivative of sigmoid = 1/4 @ 0

Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\,\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\,\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\,\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\,\overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

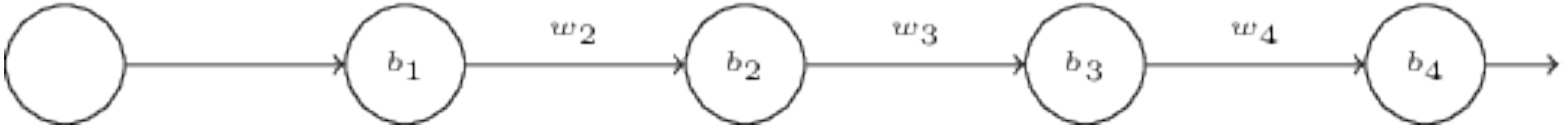
This is called vanishing gradient problem

— makes deep networks hard to train
— later layers learn faster than earlier ones

Source: http://neuralnetworksanddeeplearning.com/chap5.html

Optimizing **Deep** Neural Networks

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{>1}\overbrace{w_3\sigma'(z_3)}^{>1}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

Exploding gradient problem

— makes weights large (e.g., 100)
— make bias such that pre-activation = 0

Source: http://neuralnetworksanddeeplearning.com/chap5.html

GoogleLeNet

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

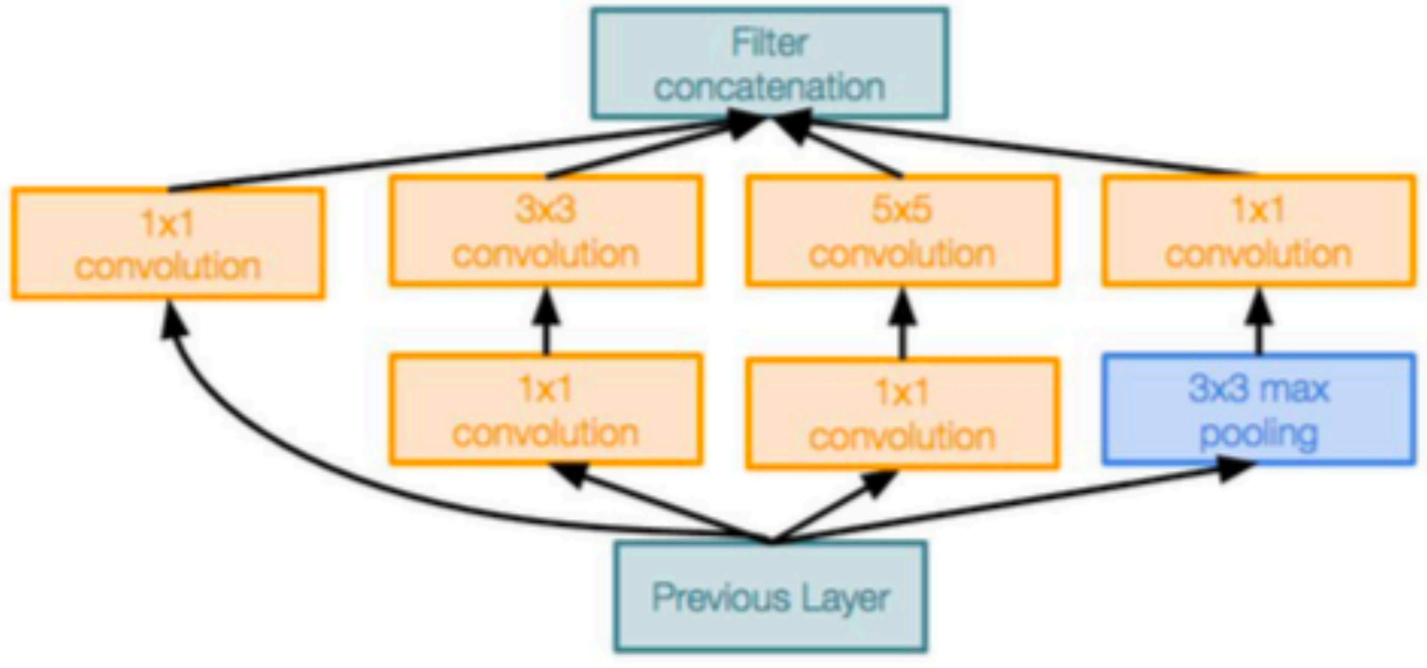
— Efficient “Inception” module

— No FC layers

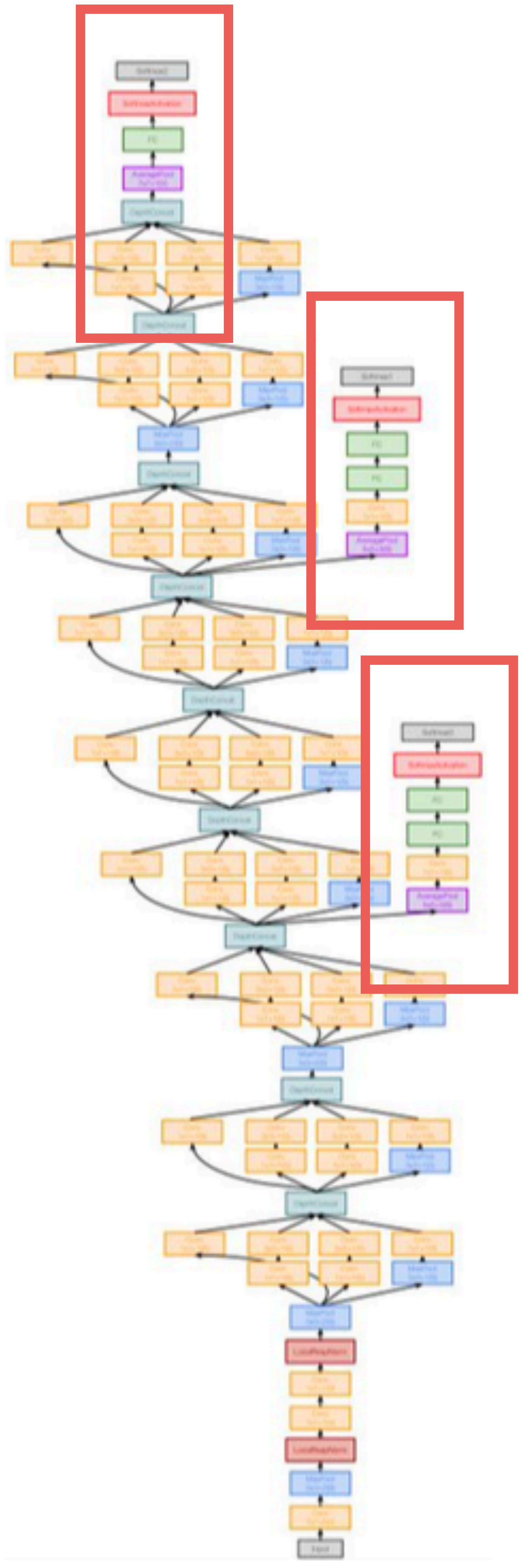
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)

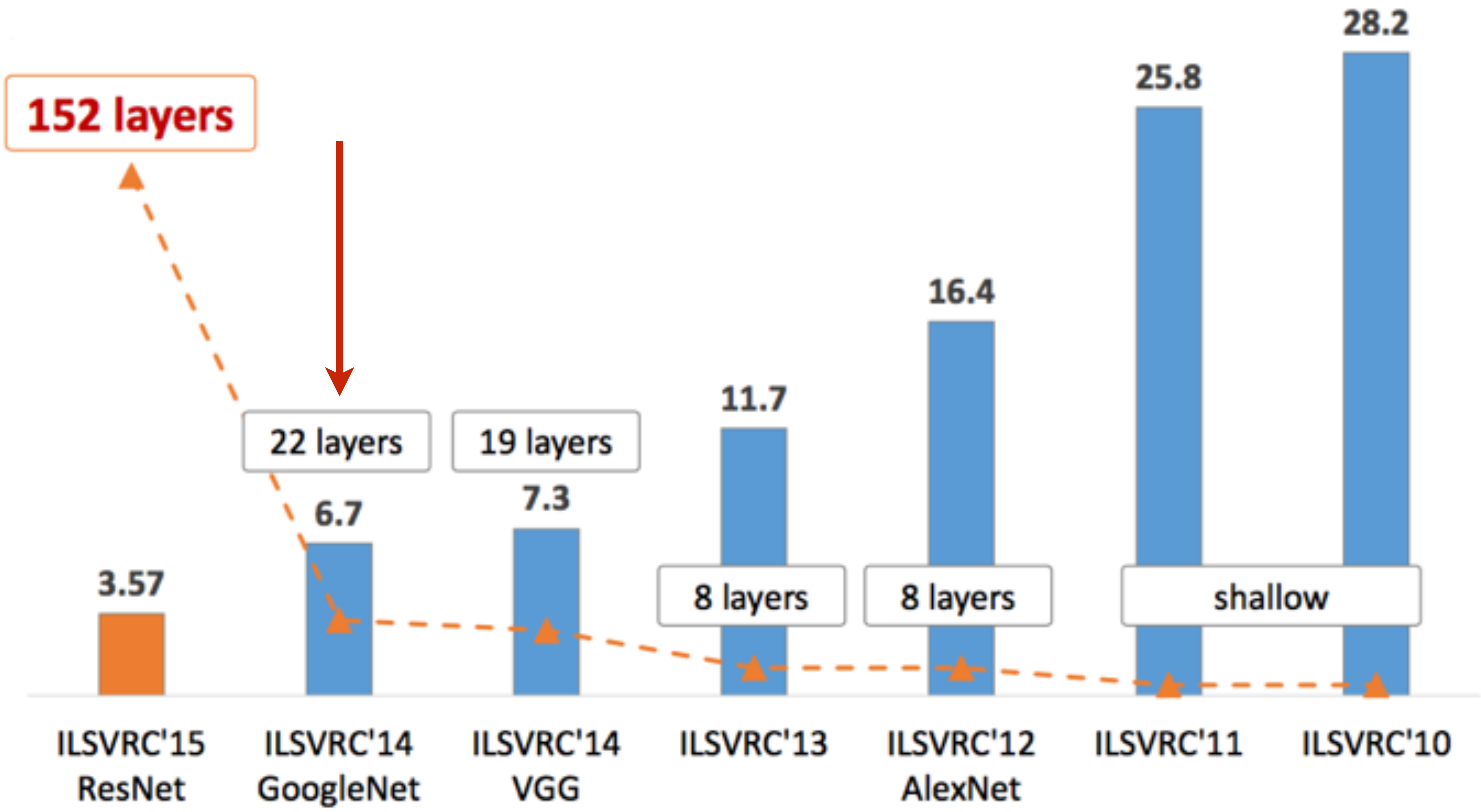


Inception module



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ILSVRC winner 2012



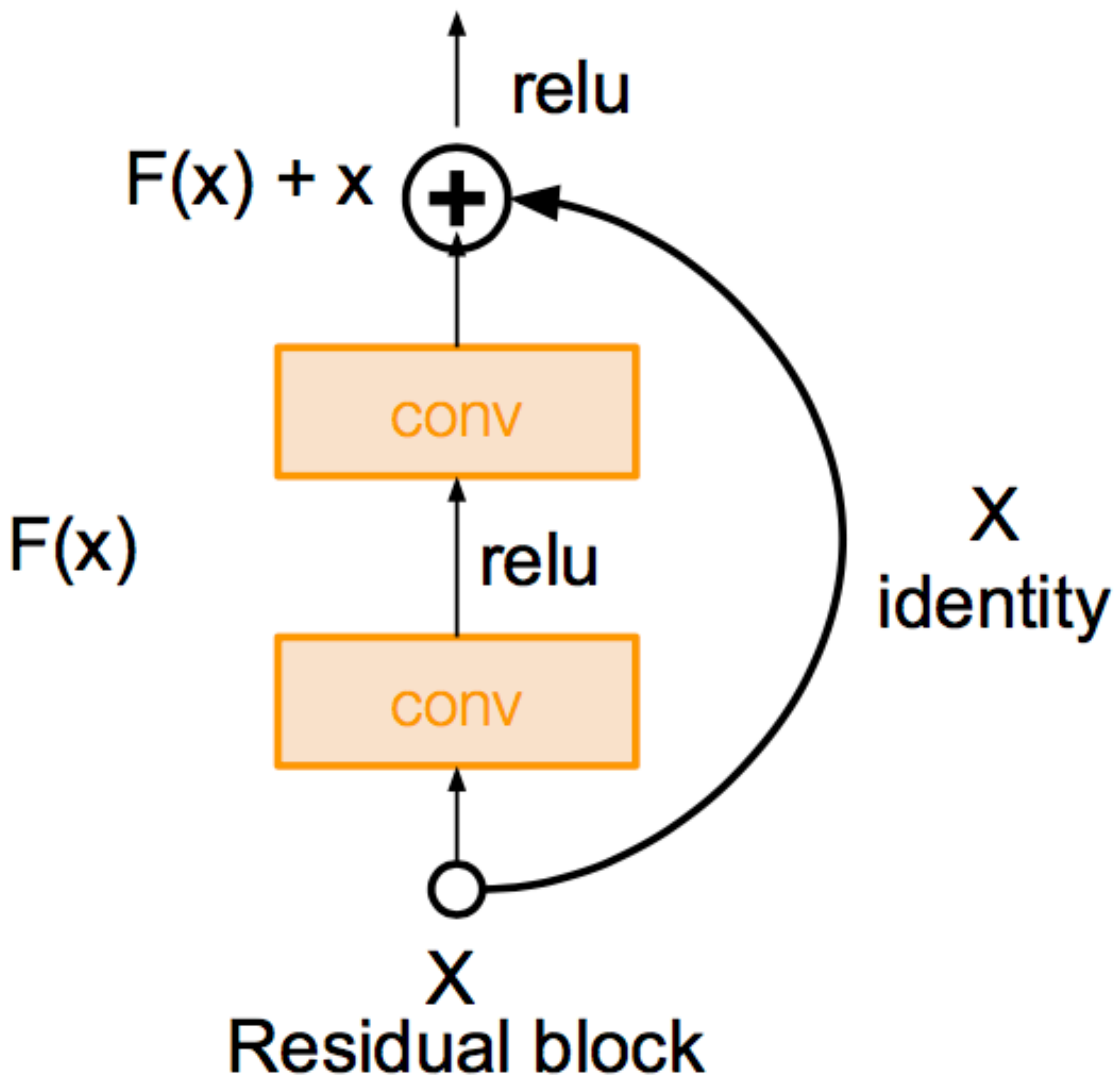
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

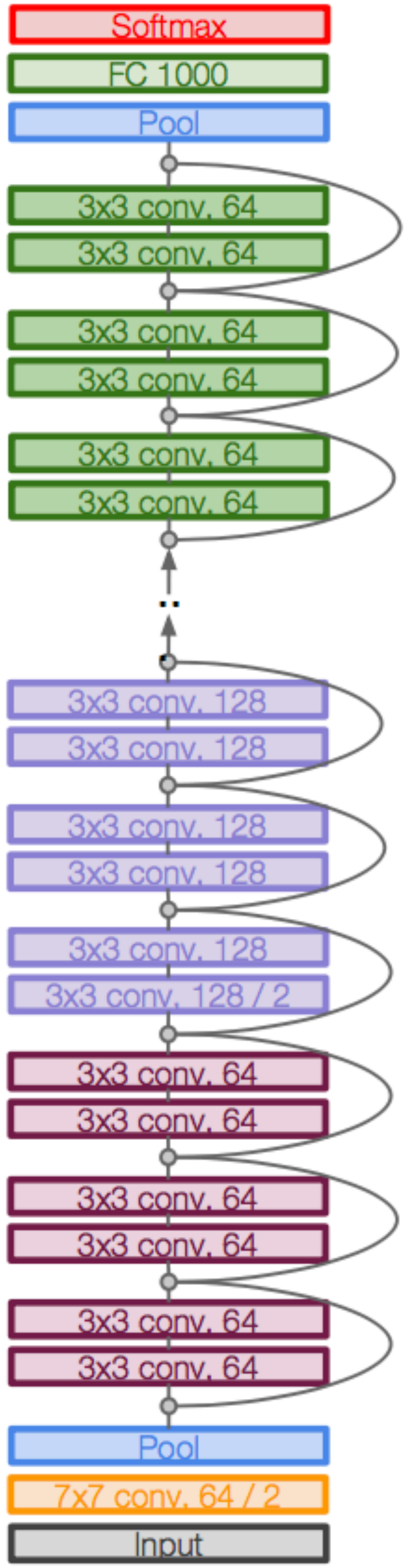
ResNet

[He et al., 2015]

even deeper — **152 layers**!

using residual connections

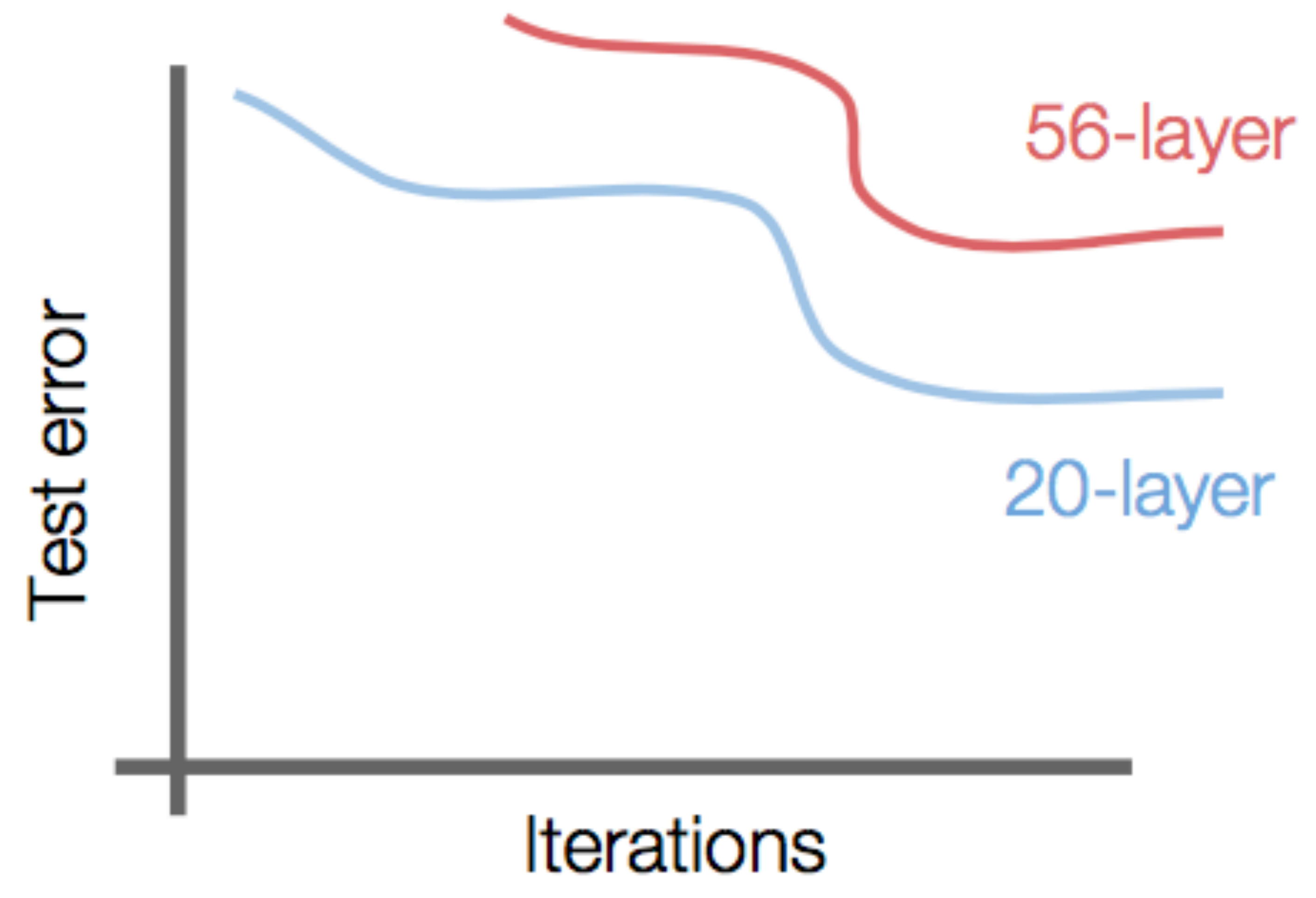




ResNet: Motivation

[He et al., 2015]

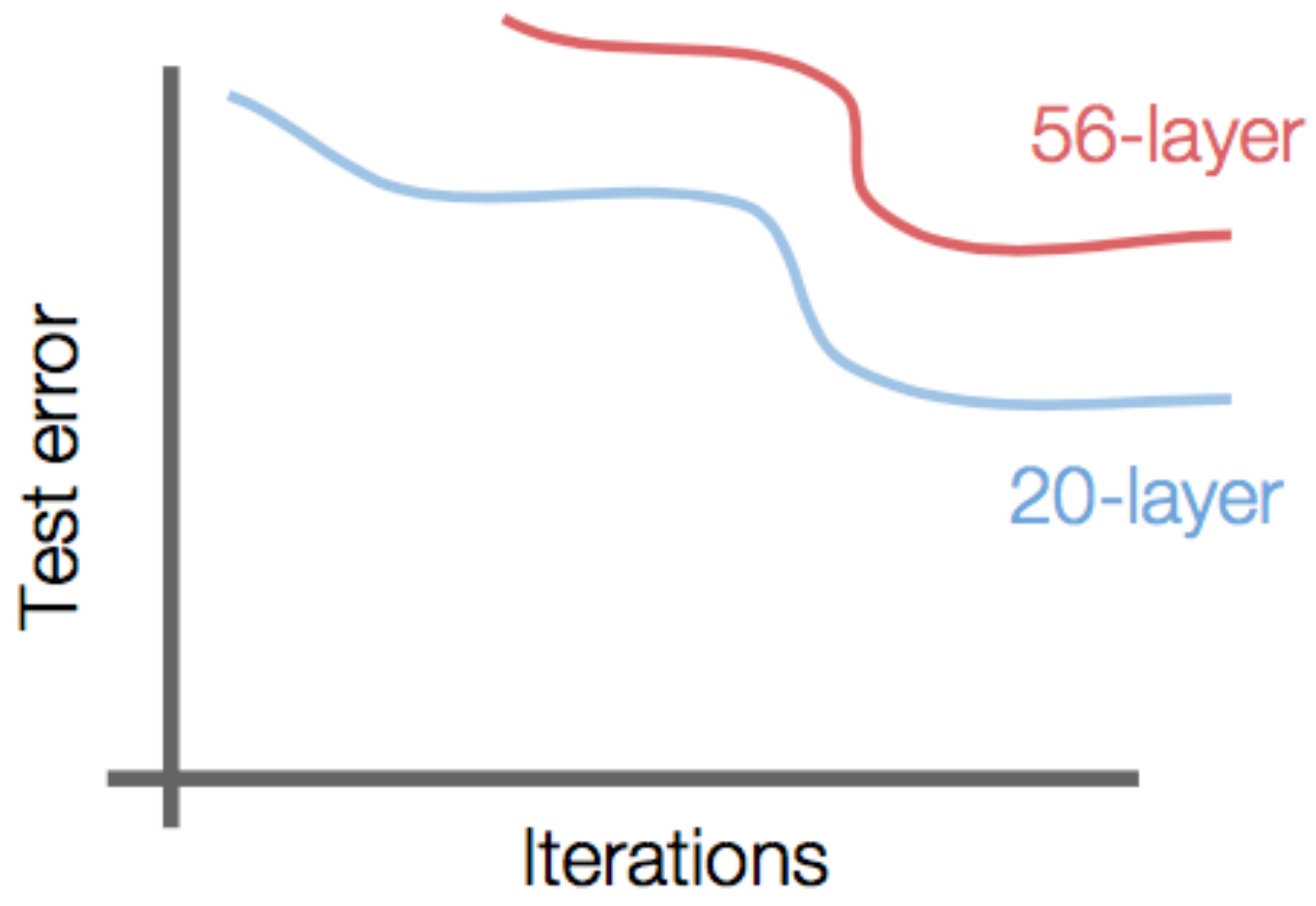
What happens when we continue to stacking deeper layers on a “plain” CNN



ResNet: Motivation

[He et al., 2015]

What happens when we continue to stacking deeper layers on a “plain” CNN



Whats the **problem**?

ResNet: Motivation

[He et al., 2015]

What happens when we continue to stacking deeper layers on a “plain” CNN

Whats the **problem**?

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ResNet: Motivation

[He et al., 2015]

Hypothesis: deeper models are harder to optimize (optimization problem)

ResNet: Motivation

[He et al., 2015]

Hypothesis: deeper models are harder to optimize (optimization problem)

Observation: the deeper model should (conceptually) perform just as well (e.g., take shallower model and use identity for all remaining layers)

ResNet: Motivation

[He et al., 2015]

Hypothesis: deeper models are harder to optimize (optimization problem)

Observation: the deeper model should (conceptually) perform just as well (e.g., take shallower model and use identity for all remaining layers)

How do we implement this idea in practice

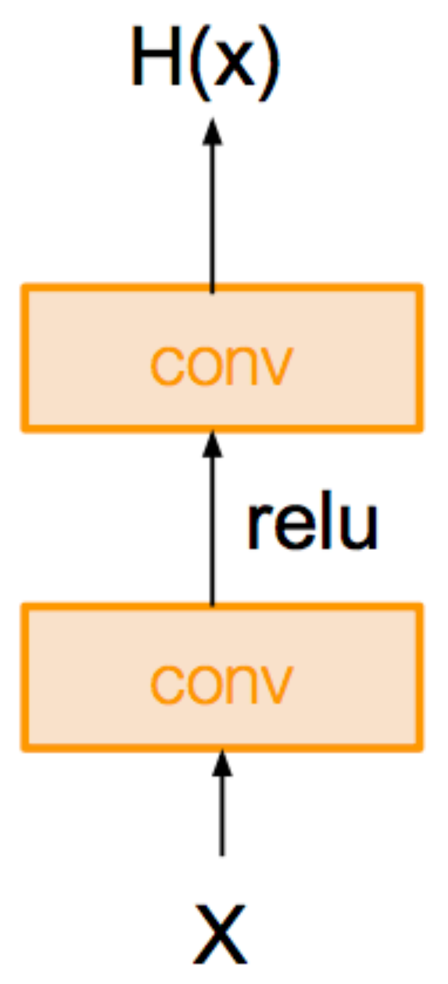
ResNet

[He et al., 2015]

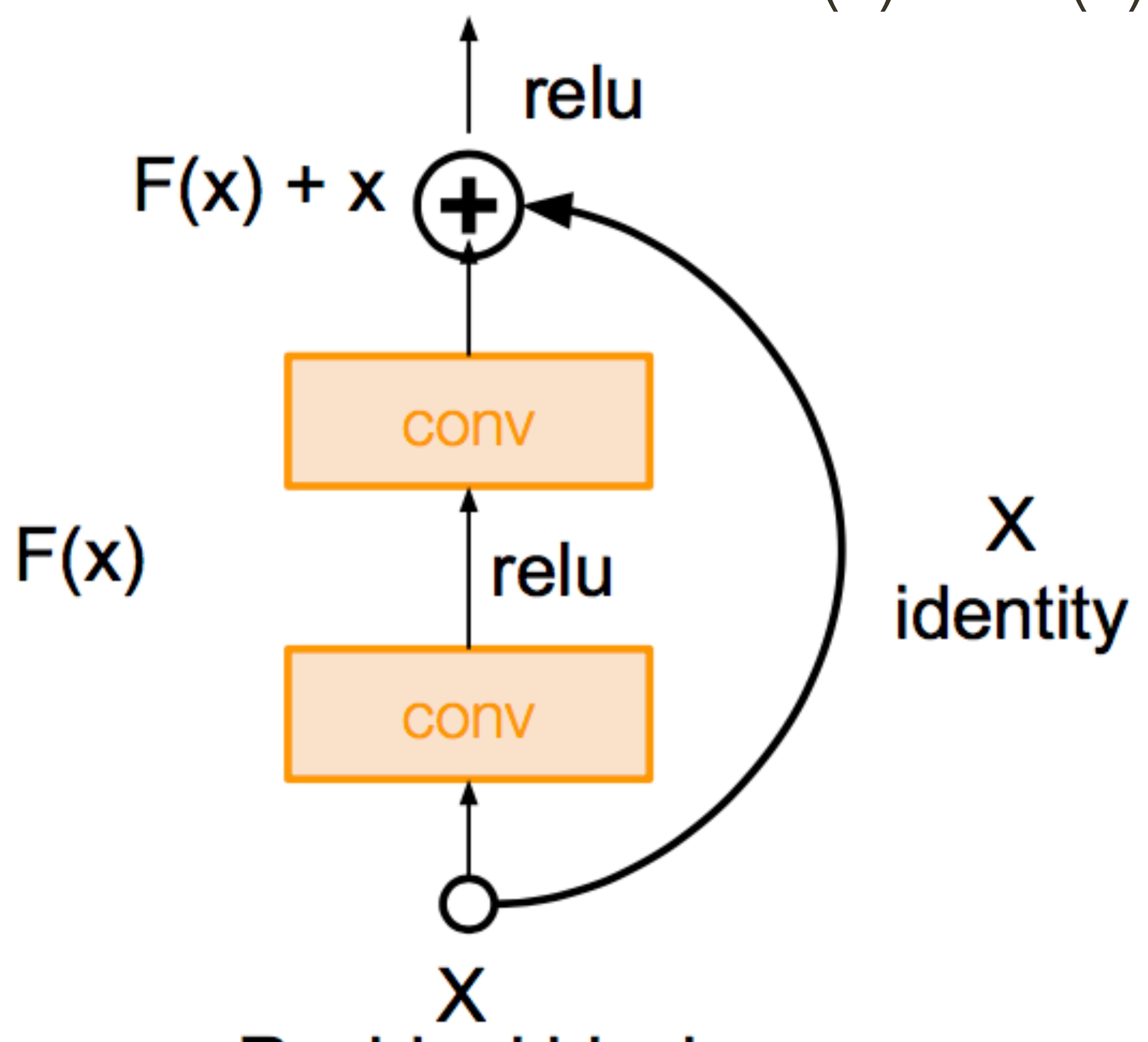
Solution: use network to fit residual mapping instead of directly trying to fit a desired underlying mapping

$$H(x) = F(x) + X$$

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



“Plain” layers



Residual block

ResNet

[He et al., 2015]

Full details

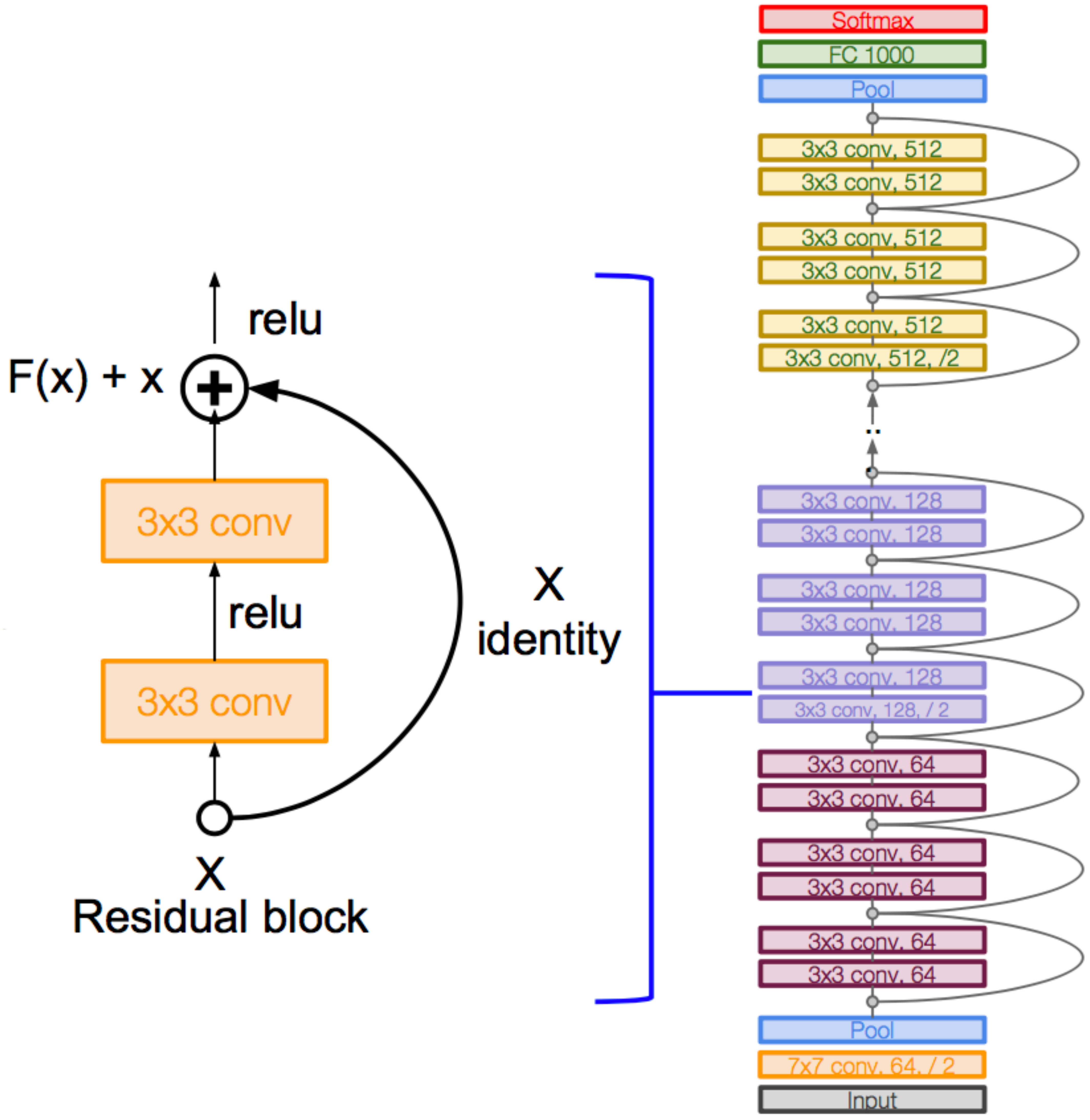
— Stacked **residual blocks**

— Every residual block consists of **two 3x3 filters**

— Periodically double # of filters and downsample spatially using stride of 2

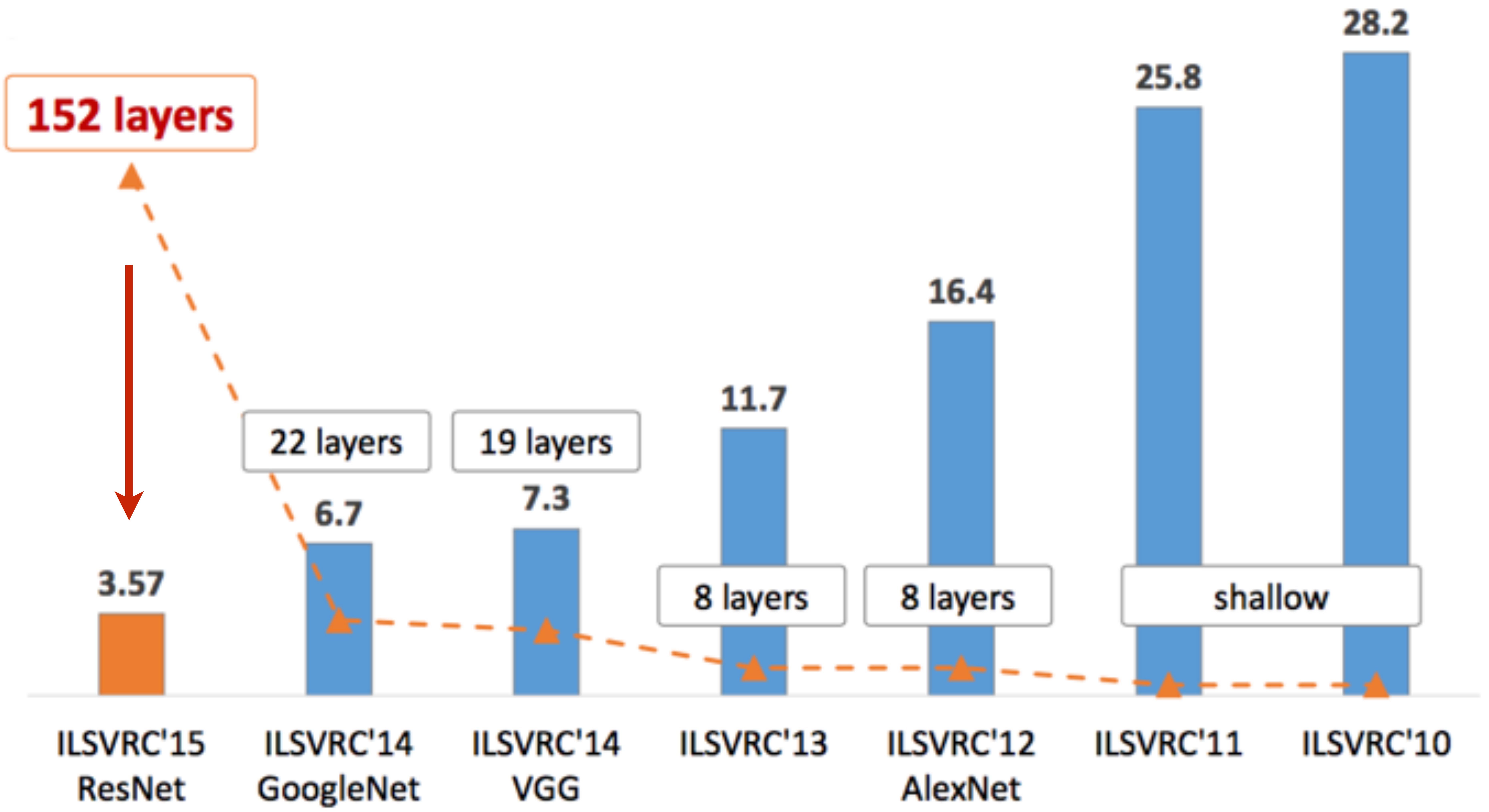
— Additional convolutional layer in the beginning

— **No FC layers** at the end (only FC to output 1000 classes)



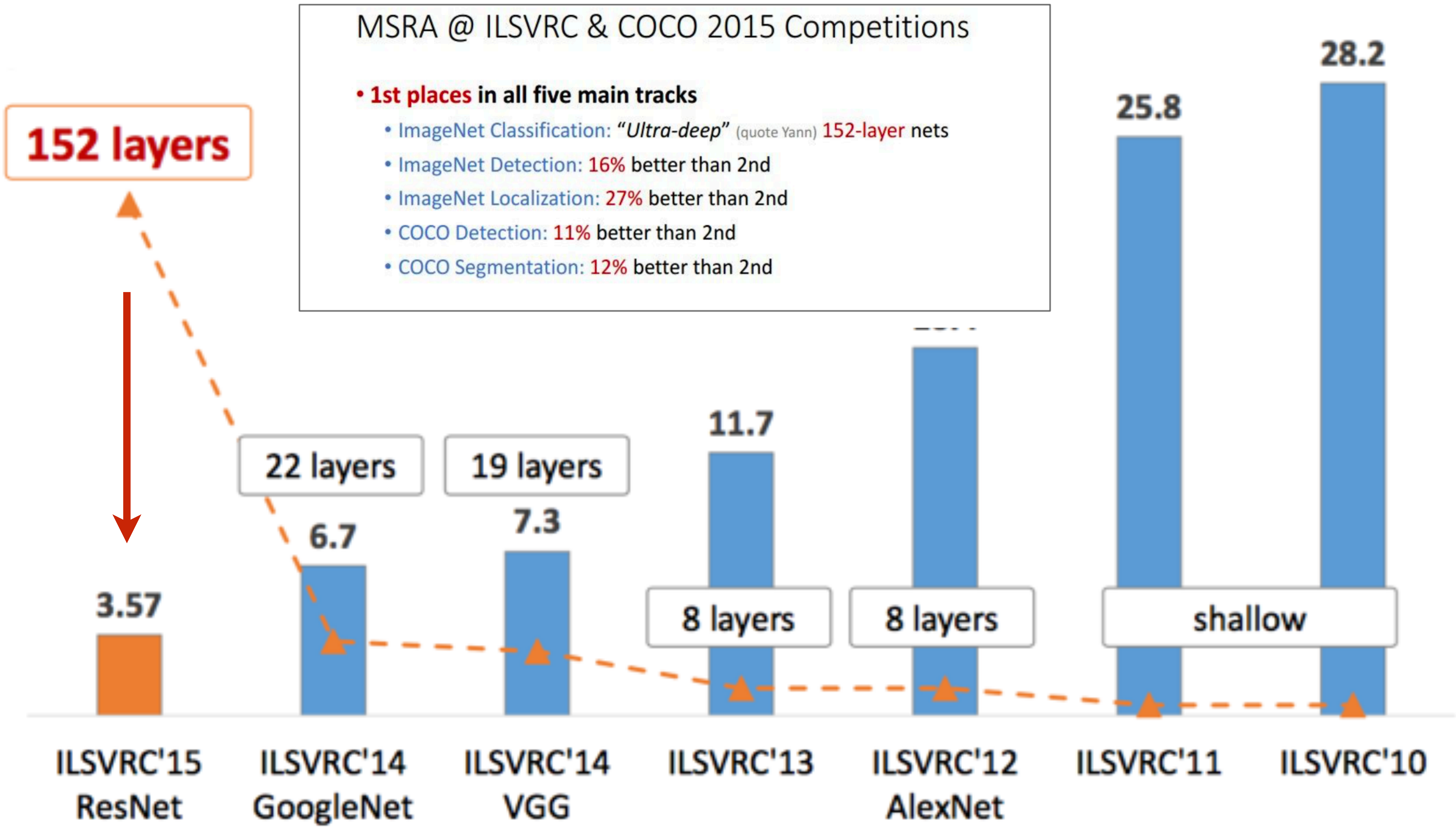
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ILSVRC winner 2012



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ILSVRC winner 2012



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

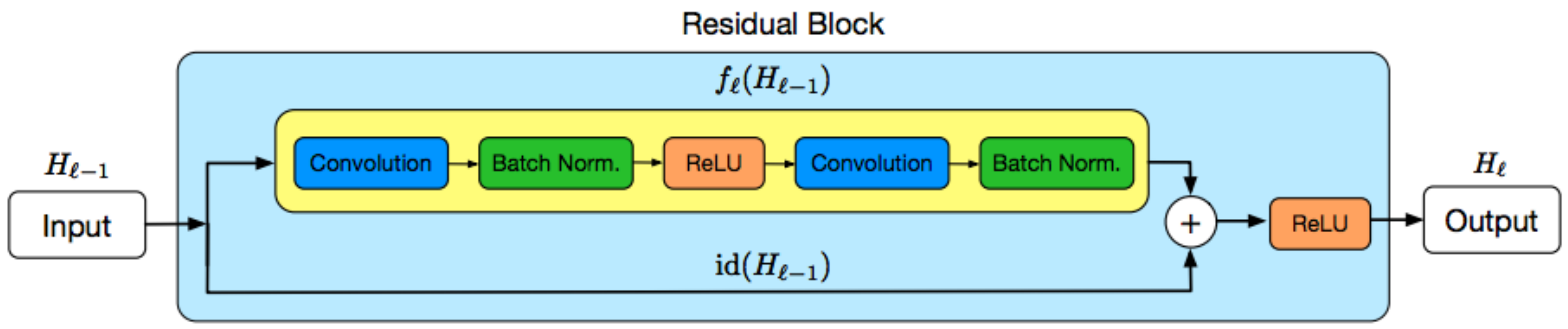
Regularization: Stochastic Depth

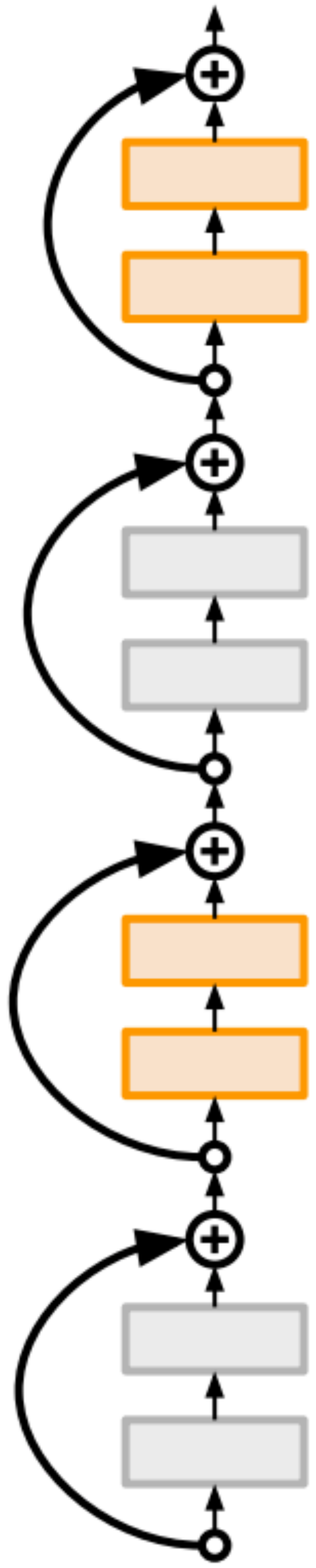
[Huang et al., ECCV 2016]

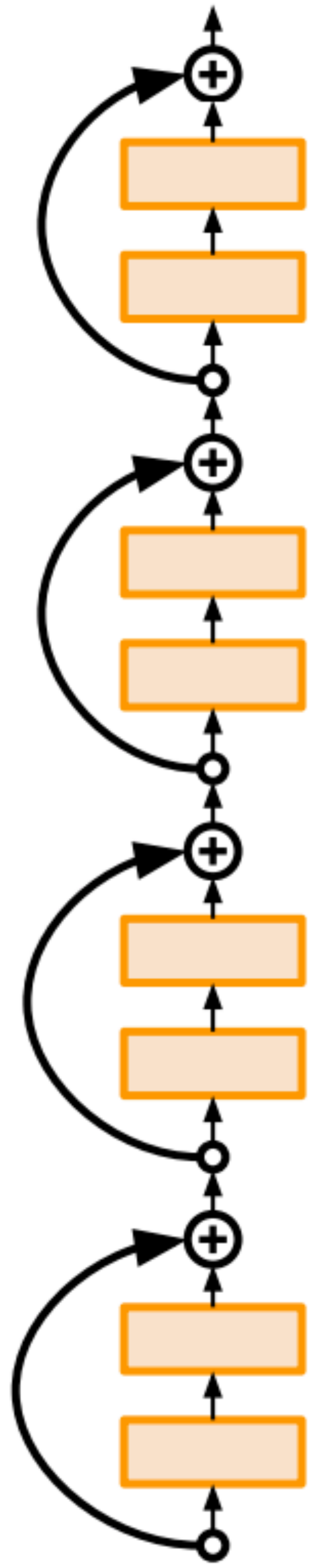
Effectively "dropout" but for layers

Stochastically with some probability **turn off some layer** (for each batch)

Effectively trains a collection of neural networks



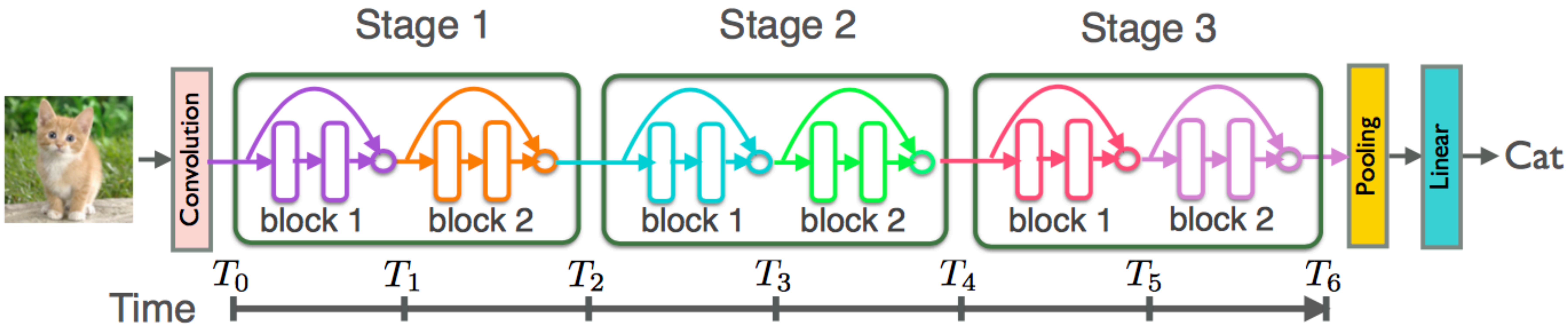




* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ResNet: A little theory

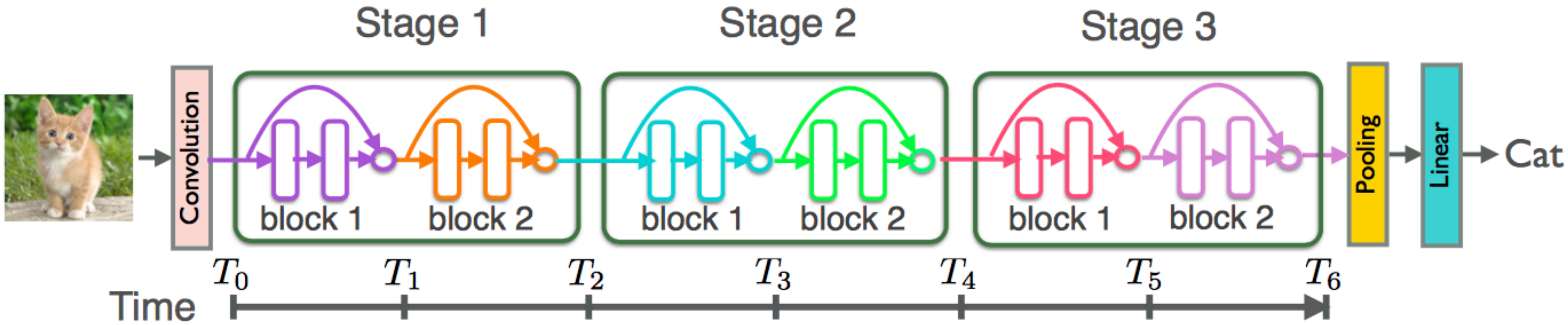
One can view a sequence of outputs from residual layers as a **Dynamical System**



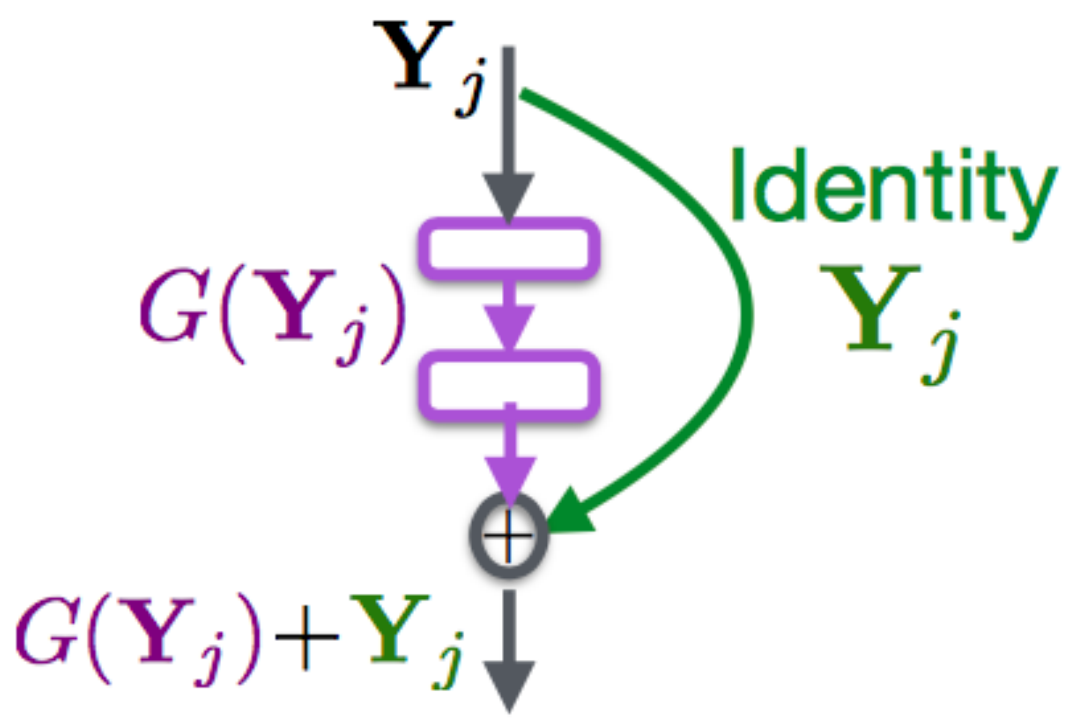
[Cheng et al., ICLR 2018]

ResNet: A little theory

One can view a sequence of outputs from residual layers as a **Dynamical System**



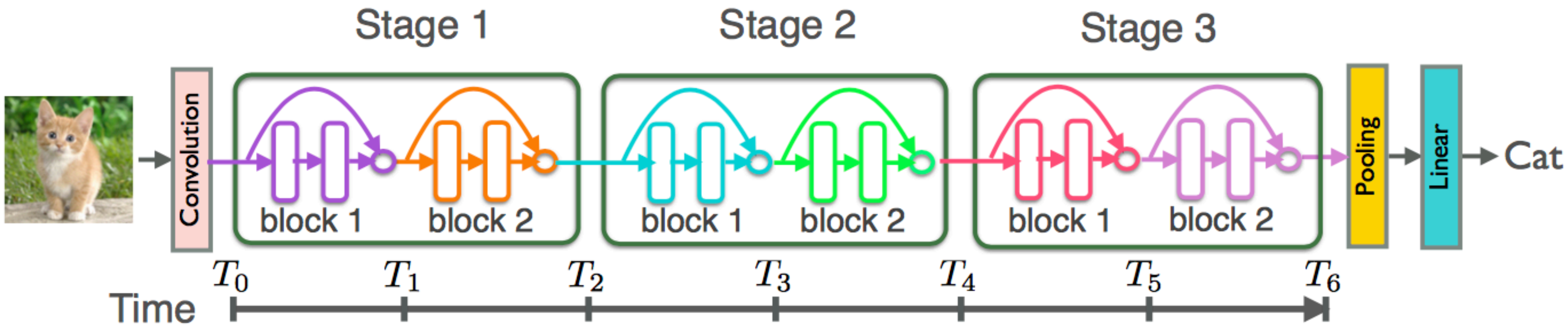
$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + G(\mathbf{Y}_j, \boldsymbol{\theta}_j)$$



[Cheng et al., ICLR 2018]

ResNet: A little theory

One can view a sequence of outputs from residual layers as a **Dynamical System**

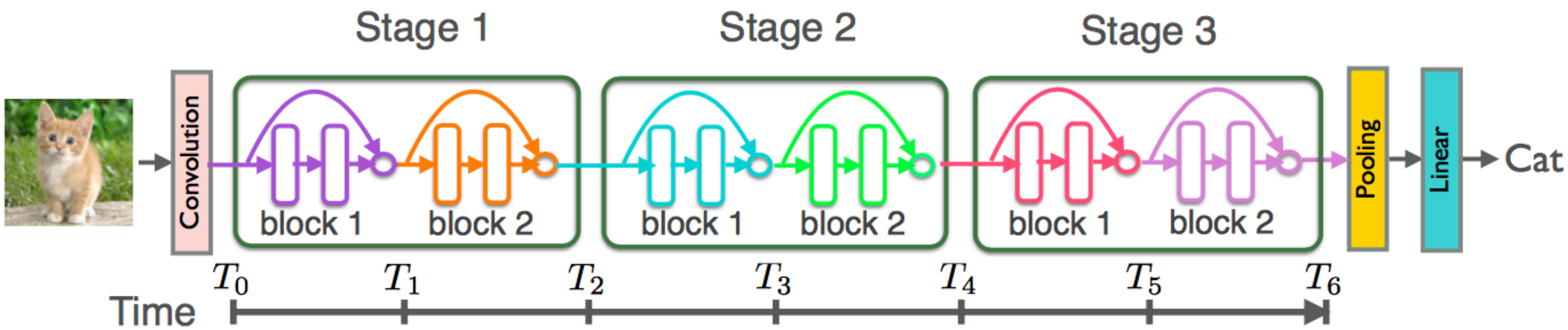


What happens if you take more layers and take smaller steps?

[Chen et al., NIPS 2018 **best paper**]

ResNet: A little theory

One can view a sequence of outputs from residual layers as a **Dynamical System**

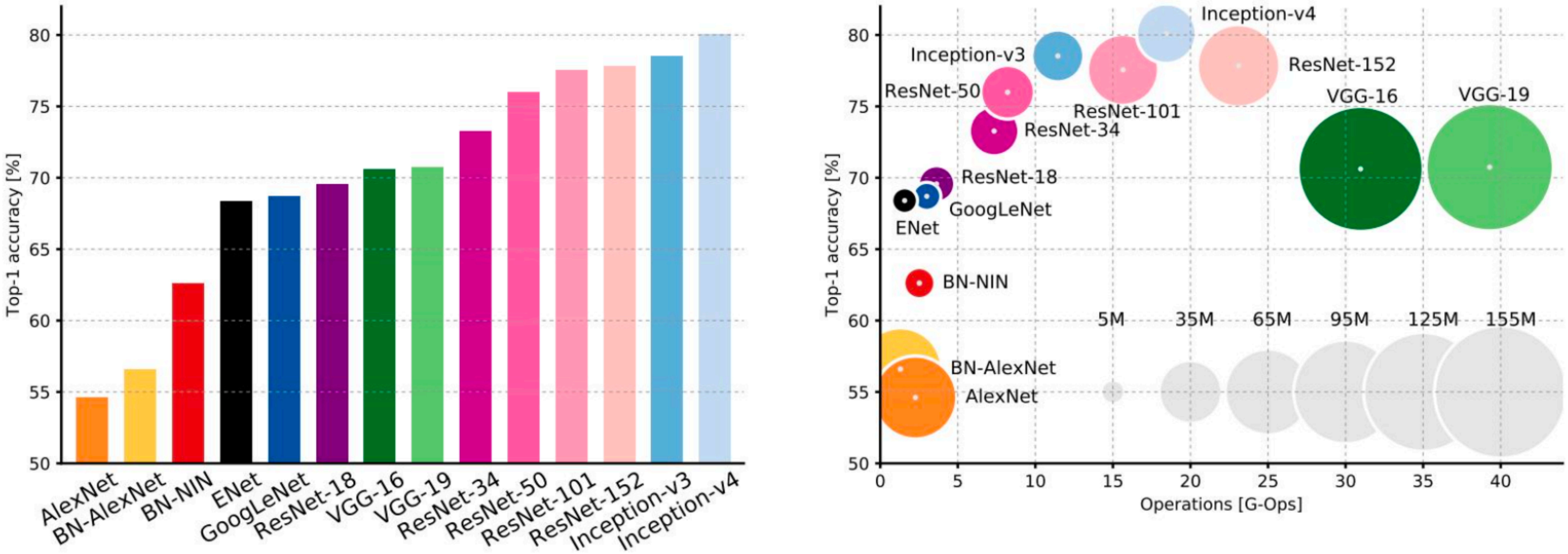


What happens if you take more layers and take smaller steps?

You can actually treat a neural network as an **ODE**: $$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

[Chen et al., NIPS 2018 **best paper**]

Comparing **Complexity**



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

Computer **Vision Problems** (no language for now)

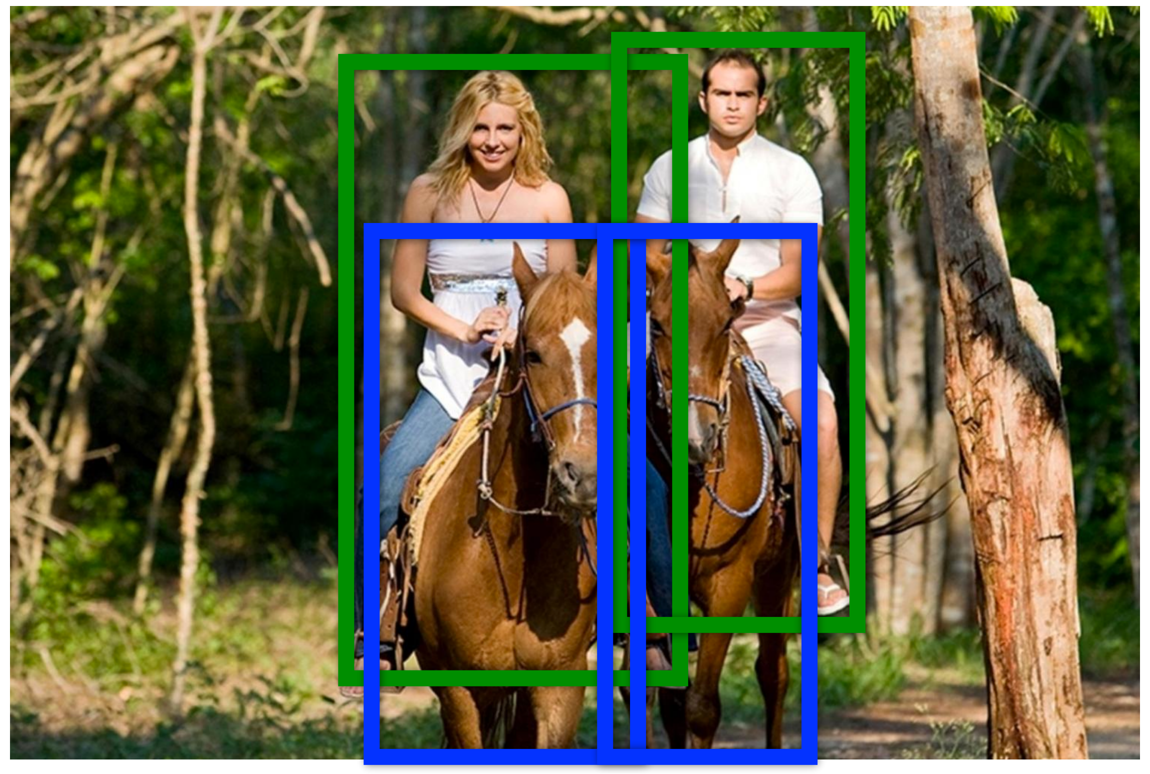
Categorization

Multi-**class**: Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

Segmentation

Horse
Person

COCO
Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2

Computer **Vision Problems** (no language for now)

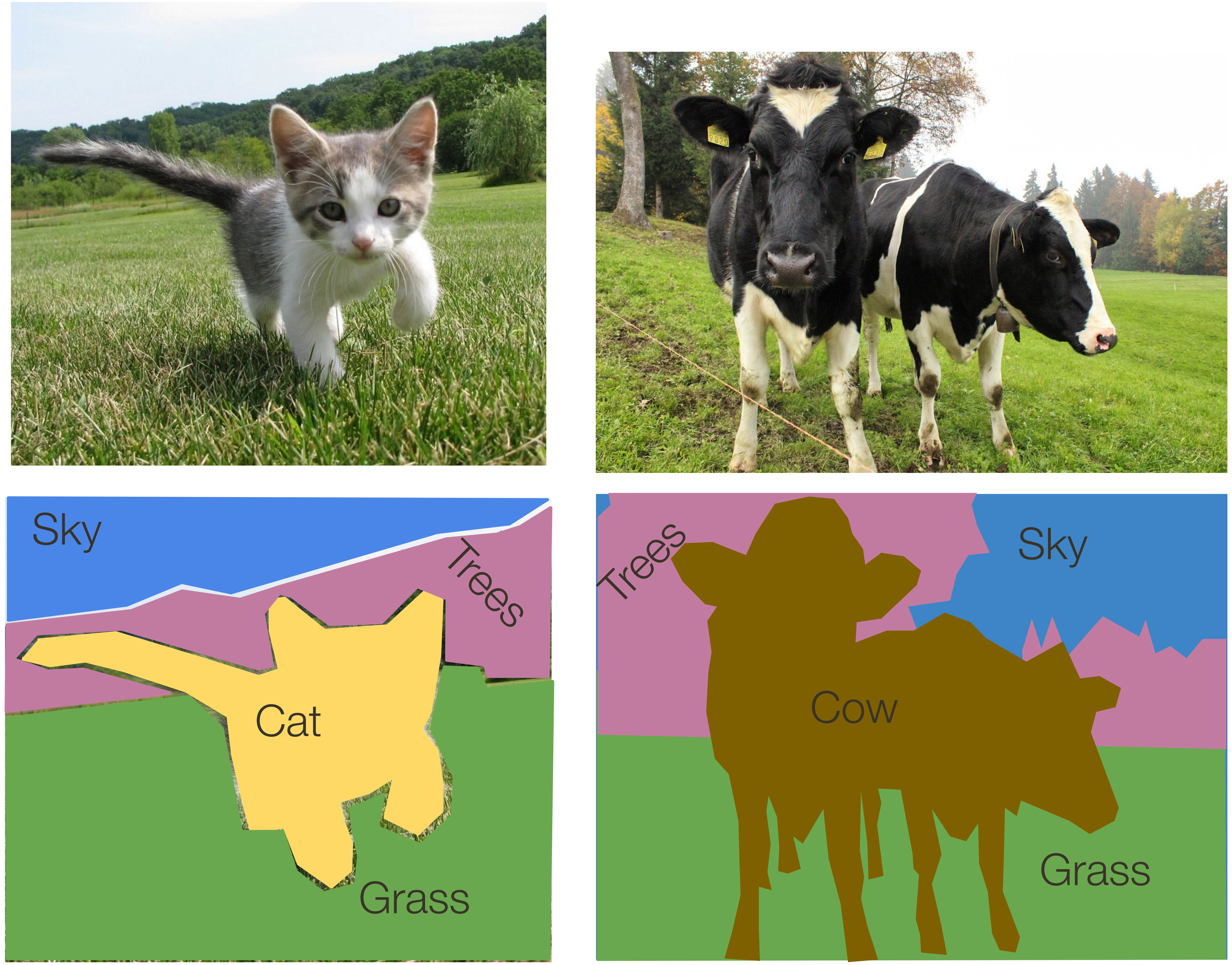
Segmentation

Horse
Person

COCO
Common Objects in Context

Semantic **Segmentation**

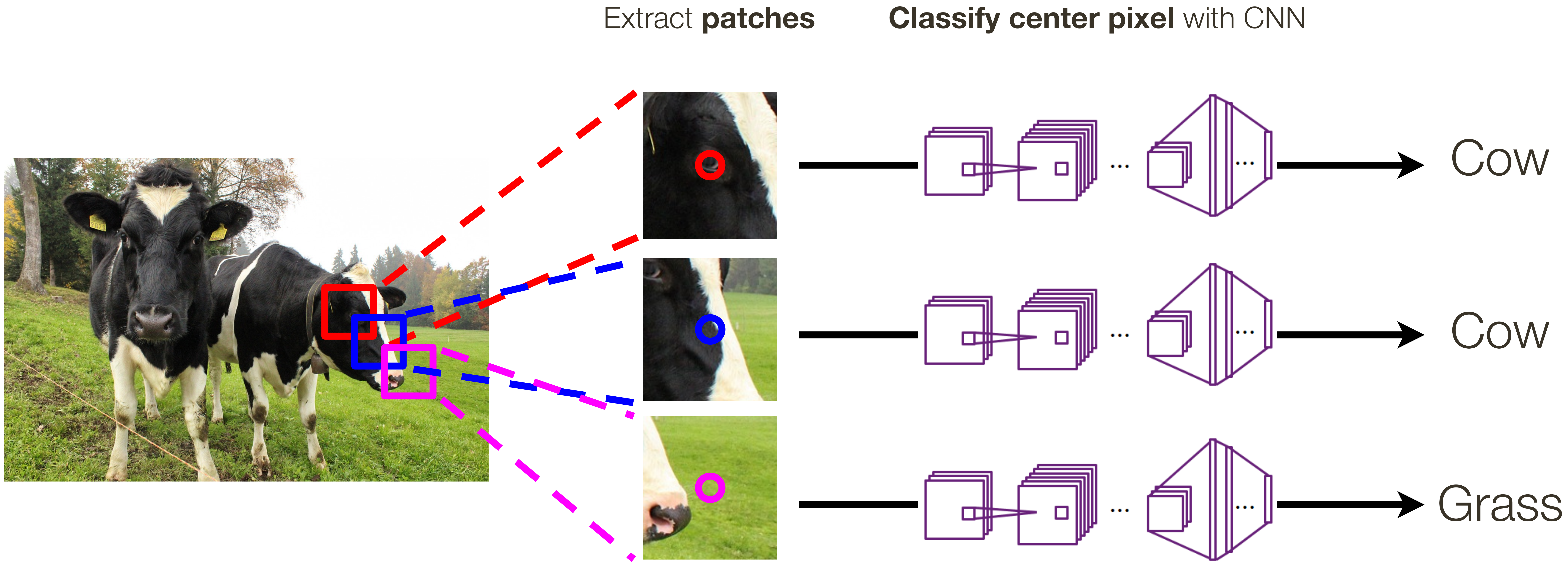
Label **every pixel** with a category label (without differentiating instances)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Sliding Window

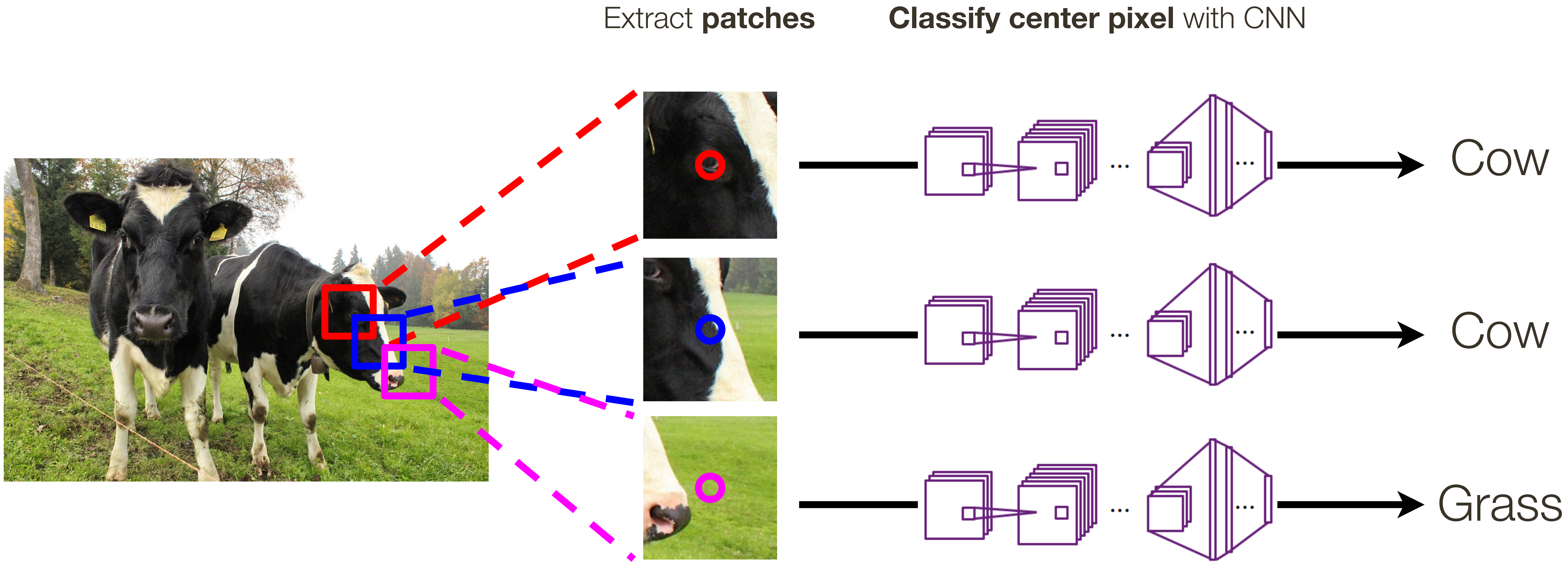
[Farabet et al, TPAMI 2013]
[Pinheiro et al, ICML 2014]



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Sliding Window

[Farabet et al, TPAMI 2013]
[Pinheiro et al, ICML 2014]

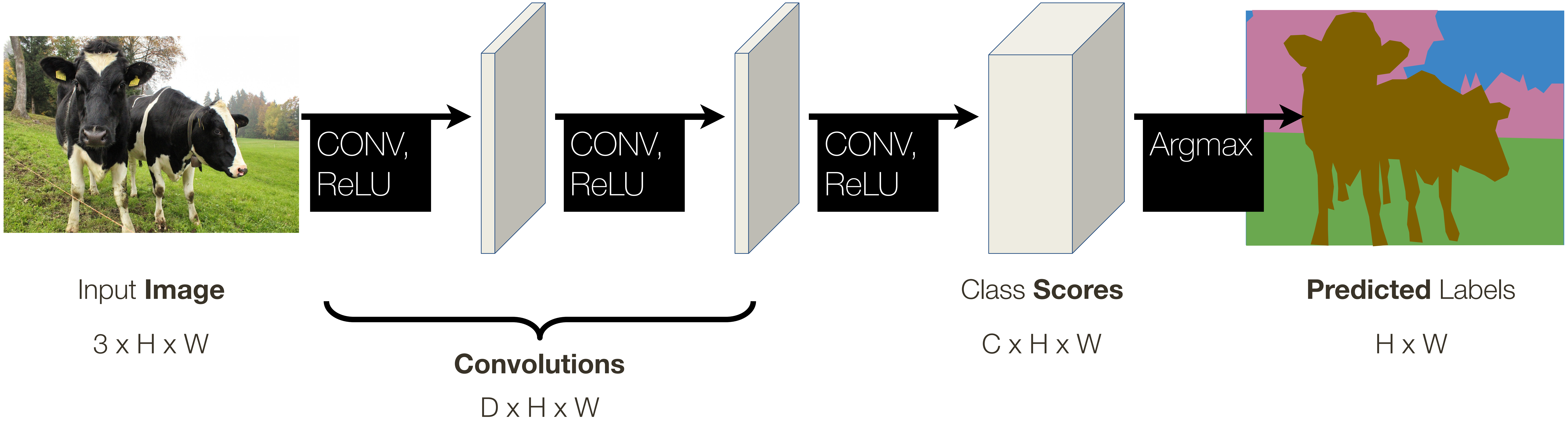


Problem: VERY inefficient, no reuse of computations for overlapping patches

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

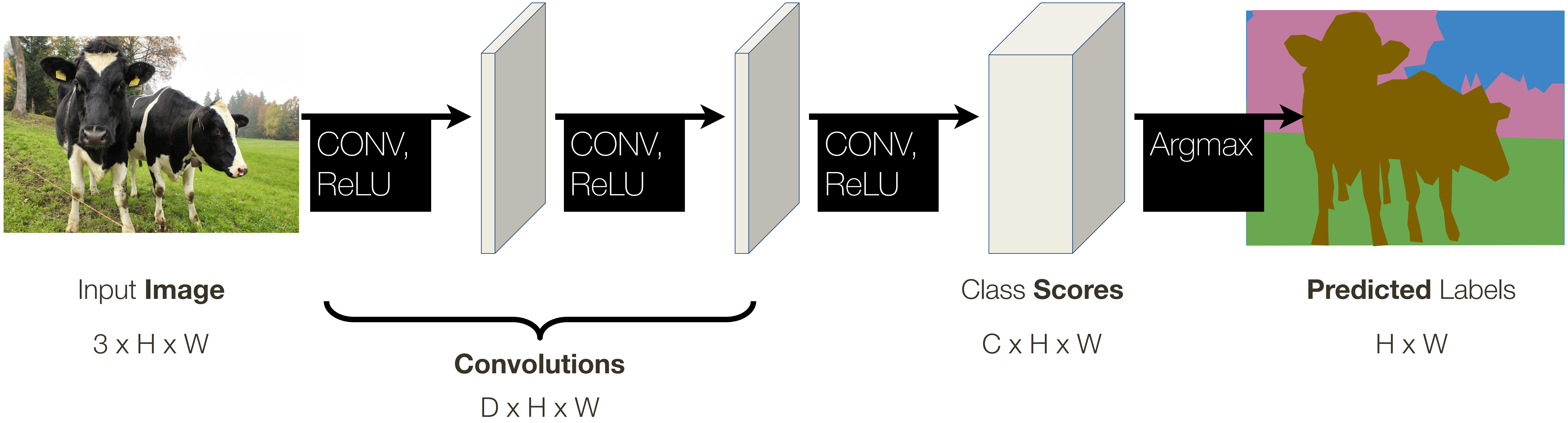
Design a network as a number of convolutional layers to make predictions for all pixels at once!



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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



Problem: Convolutions at the original image scale will be very expensive

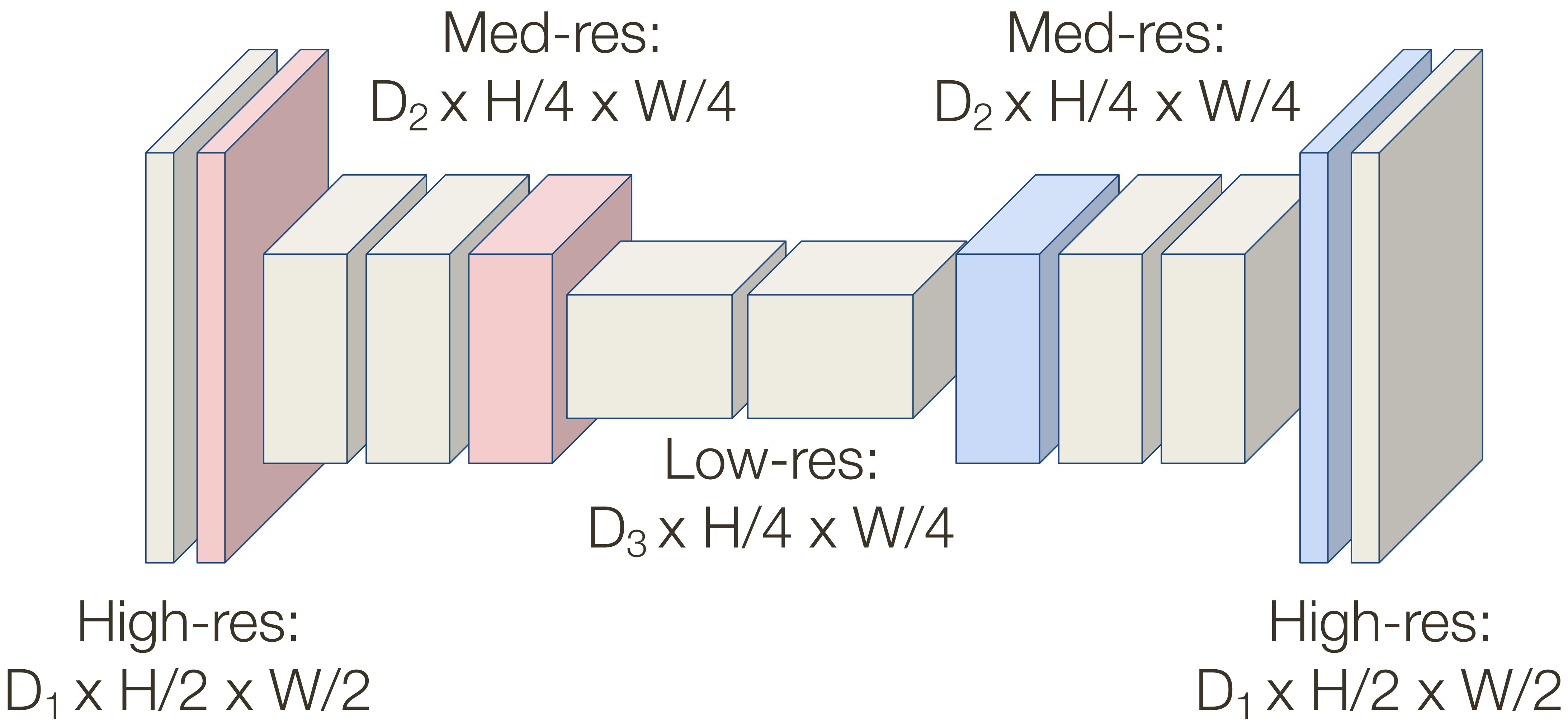
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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

Input **Image**

3 x H x W



Predicted Labels

H x W

[Long et al, CVPR 2015]
[Noh et al, ICCV 2015]

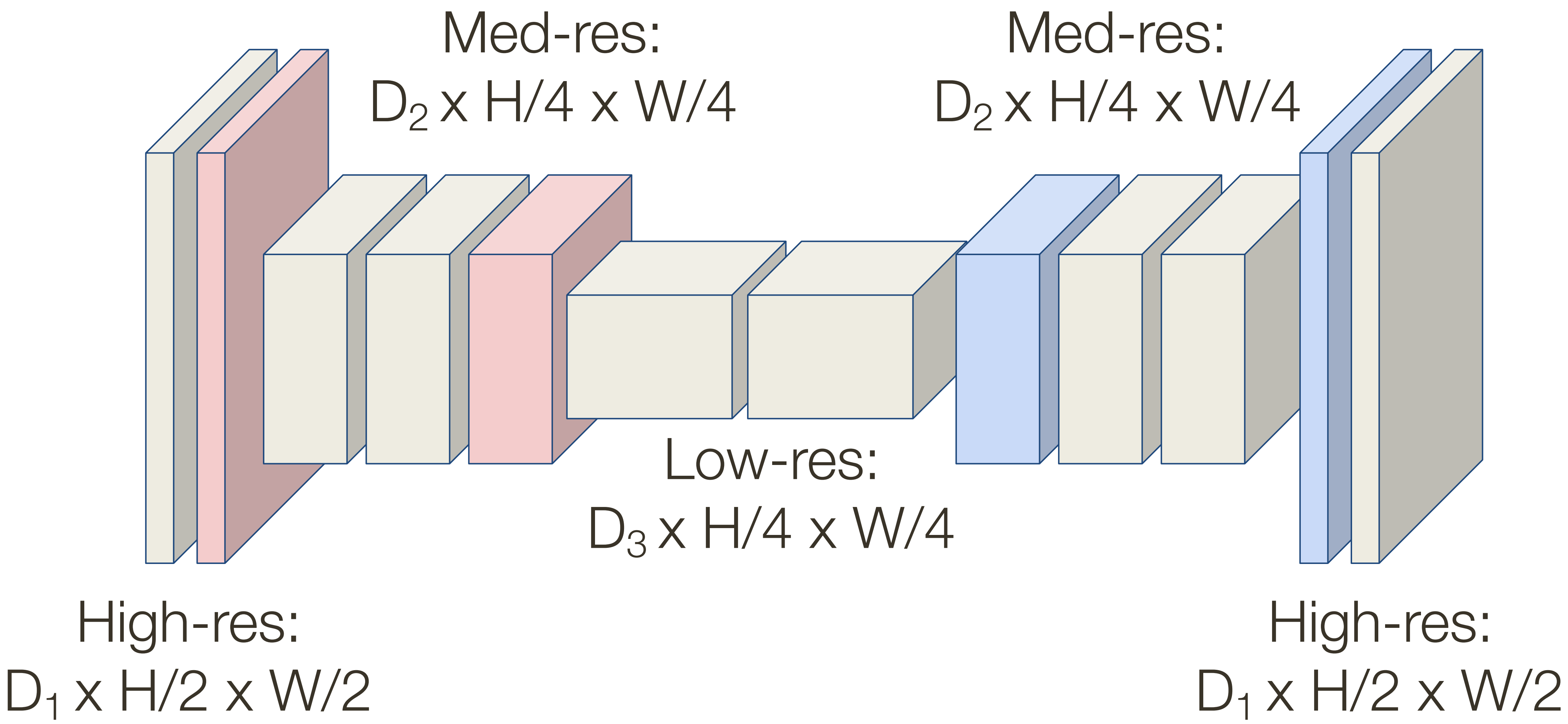
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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

Input **Image**

3 x H x W



Predicted Labels

H x W

Downsampling = Pooling

Upsampling = ???

[Long et al, CVPR 2015]
[Noh et al, ICCV 2015]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling** (a.k.a "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

In-network **Up Sampling** (a.k.a “Unpooling”)

Nearest Neighbor

$$\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \rightarrow \begin{array}{cc|cc} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ \hline 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{array}$$

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

$$\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \rightarrow \begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{array}$$

Input: 2 x 2

Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Max Unpooling

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

→

5	6
7	8

Output: 2 x 2

→ … →

Rest of the network

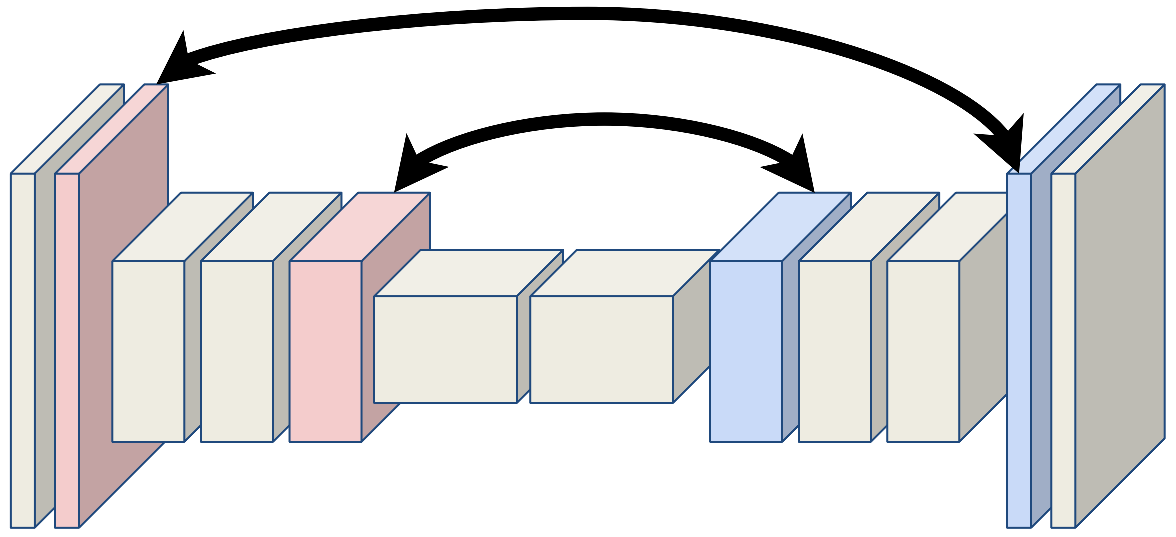
Max Unpooling
Use positions from pooling layer

1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

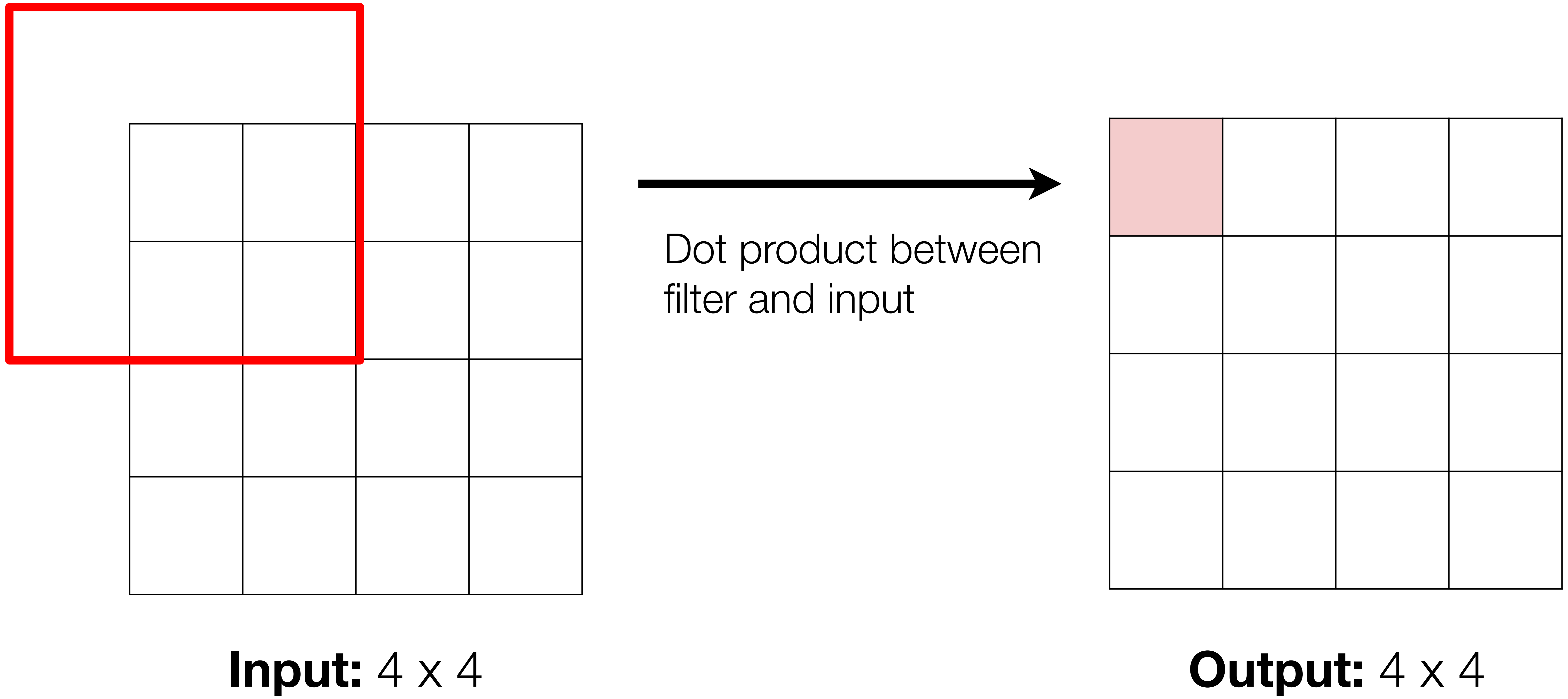


Corresponding pairs of downsampling and upsampling layers

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

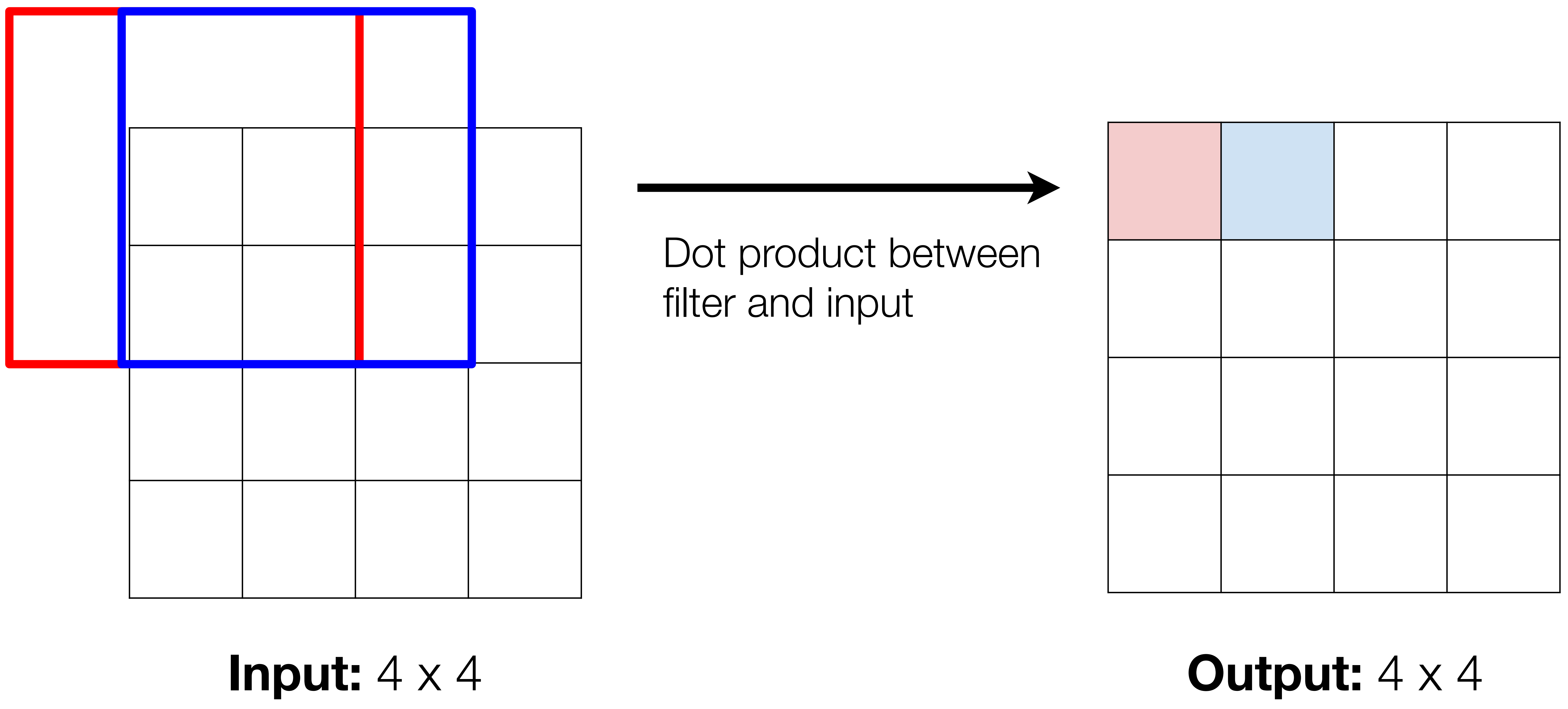
In-network **Up Sampling:** Transpose Convolution

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



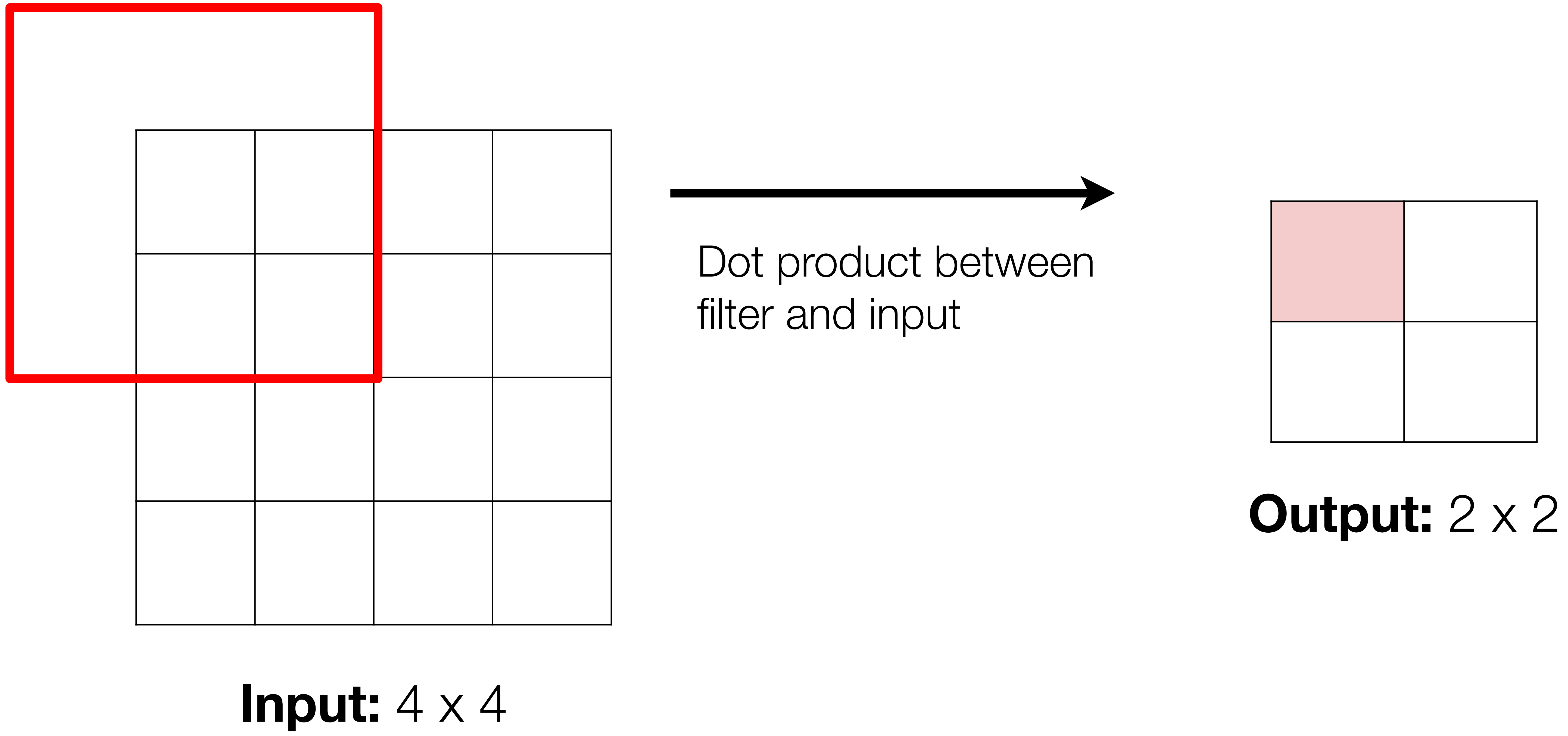
In-network **Up Sampling:** Transpose Convolution

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



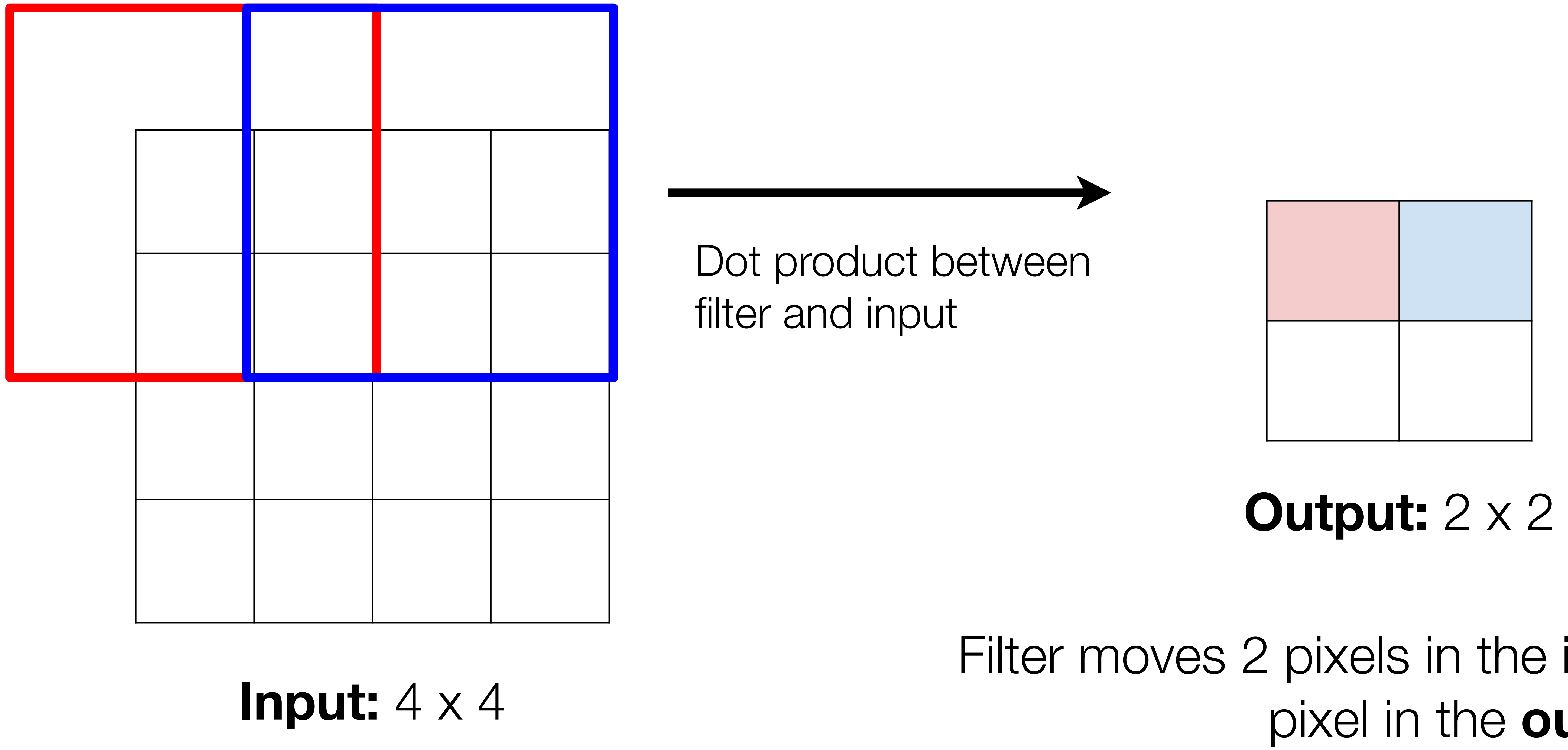
In-network **Up Sampling:** Transpose Convolution

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



In-network **Up Sampling:** Transpose Convolution

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 in movement in input vs output

In-network **Up Sampling:** Transpose Convolution

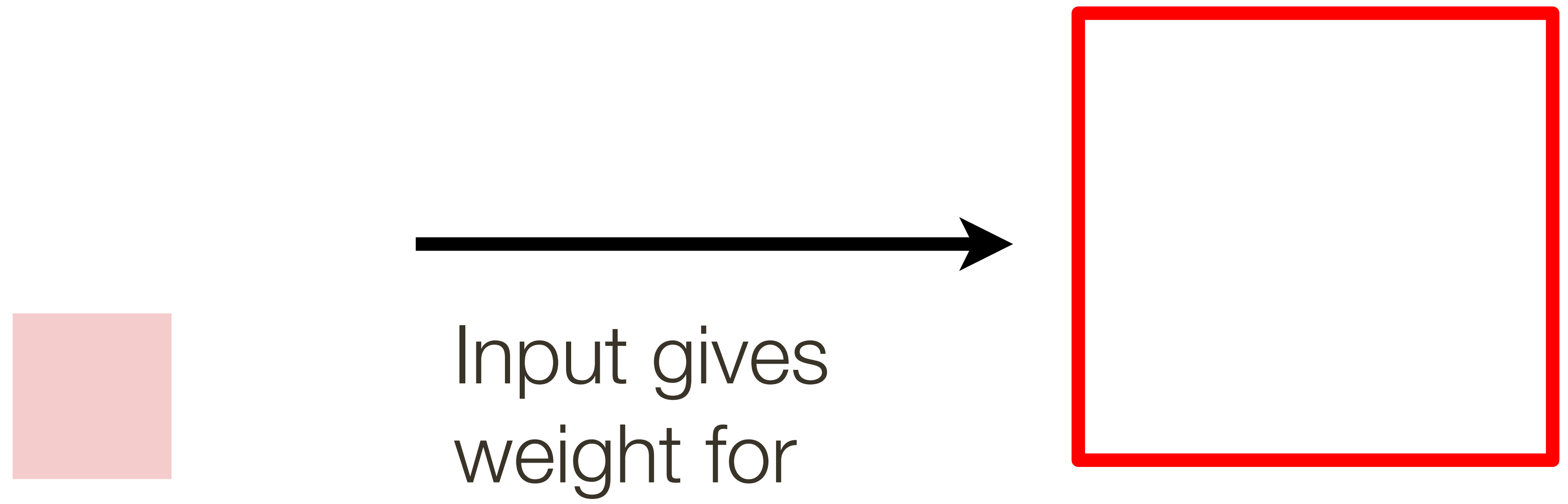
3 x 3 **transpose** convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

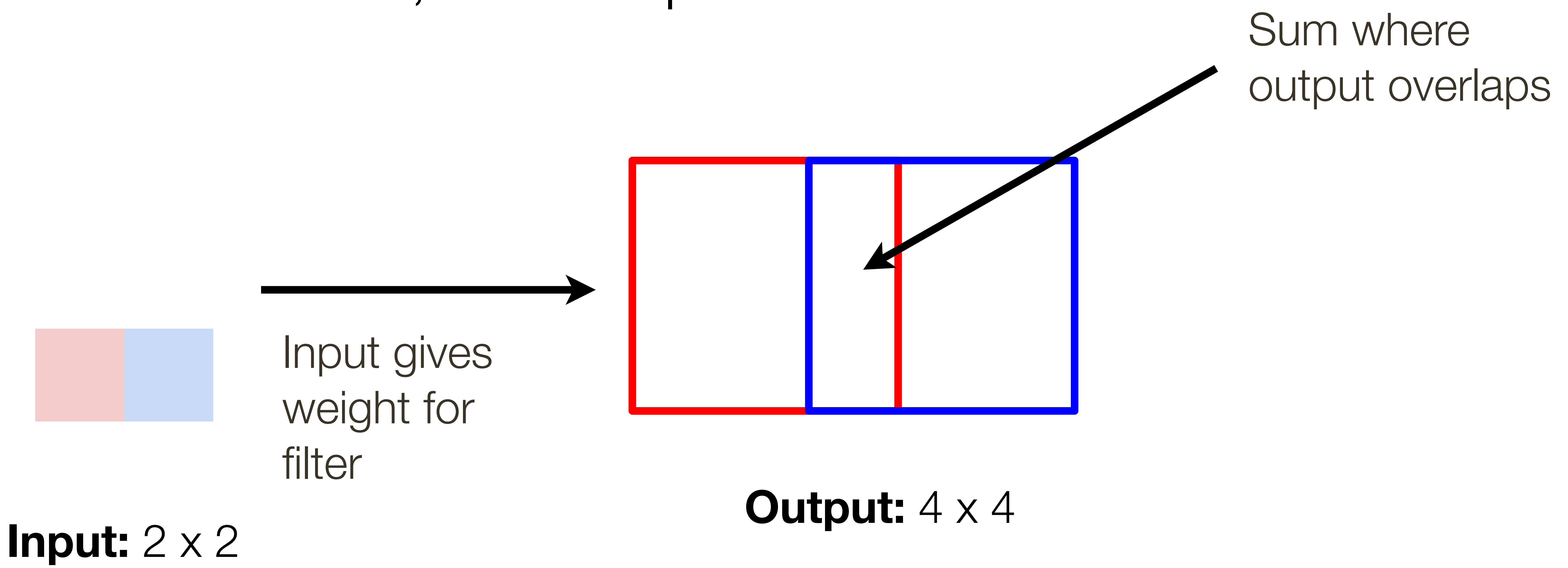
In-network **Up Sampling:** Transpose Convolution

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



In-network **Up Sampling:** Transpose Convolution

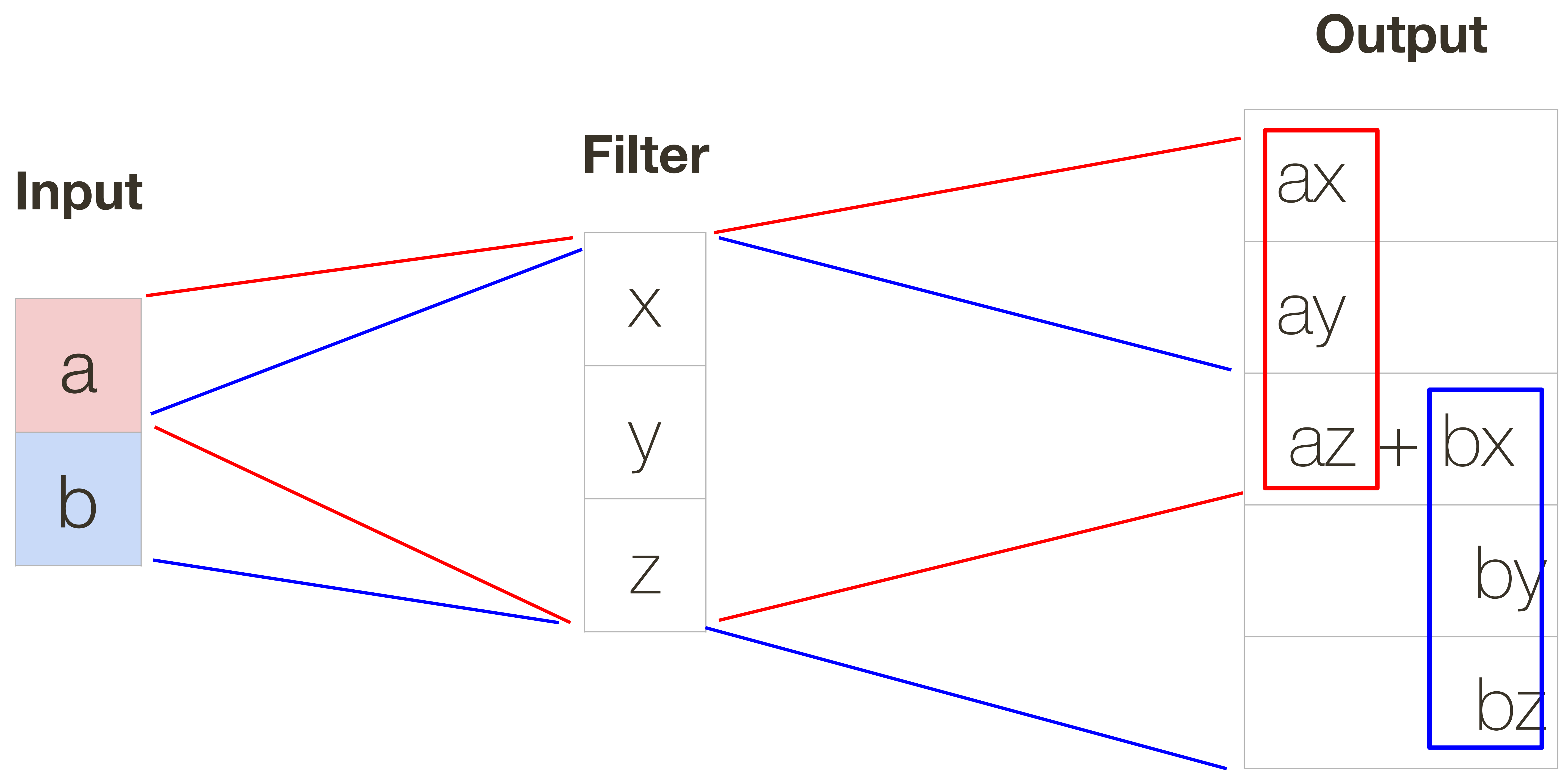
3 x 3 **transpose** convolution, stride 2 pad 1



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

Stride gives ratio in movement in output vs input

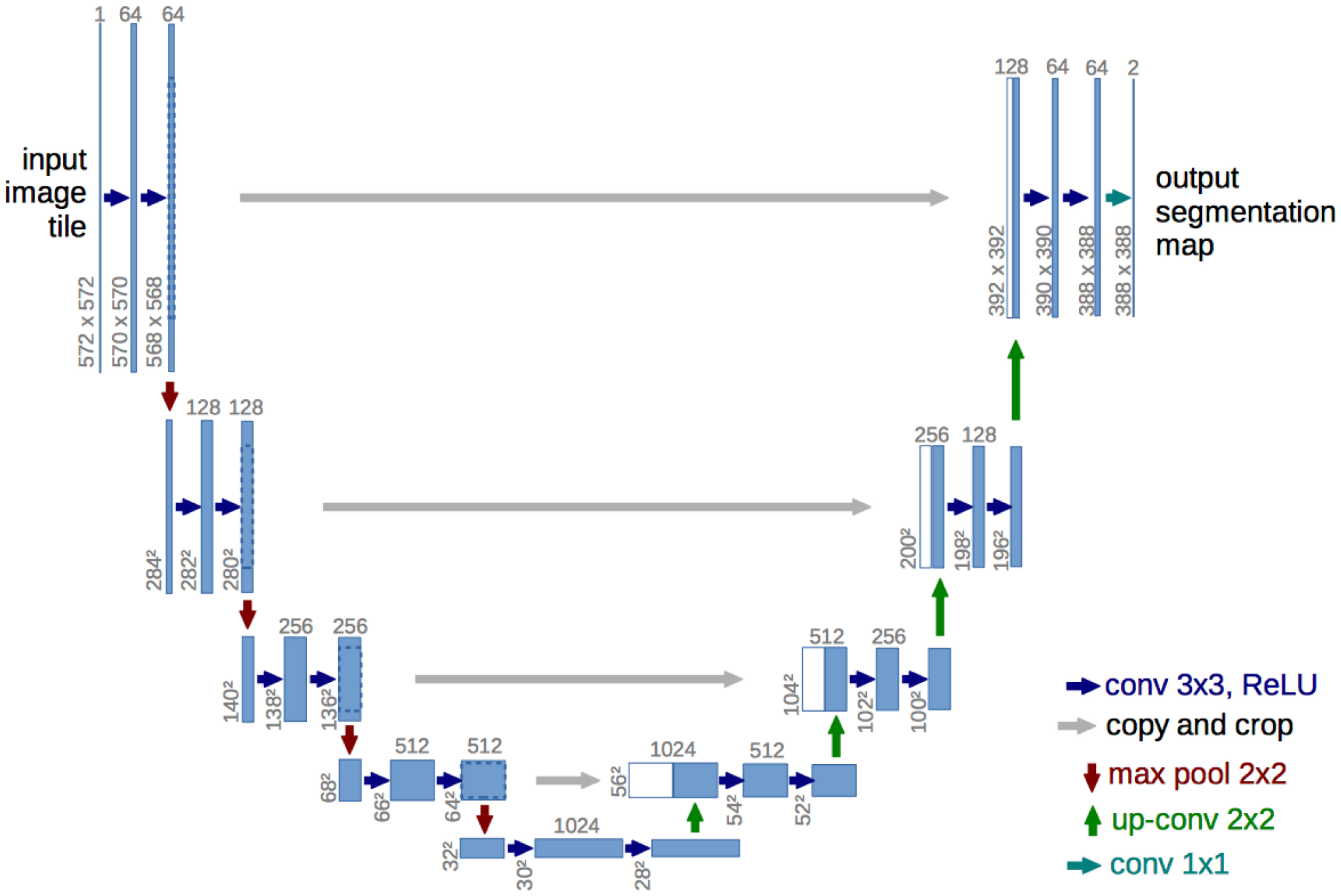
Transpose Convolution: 1-D Example



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

U-Net Architecture

ResNet-like Fully convolutional CNN



[Ronneberger et al, CVPR 2015]

Computer **Vision Problems** (no language for now)

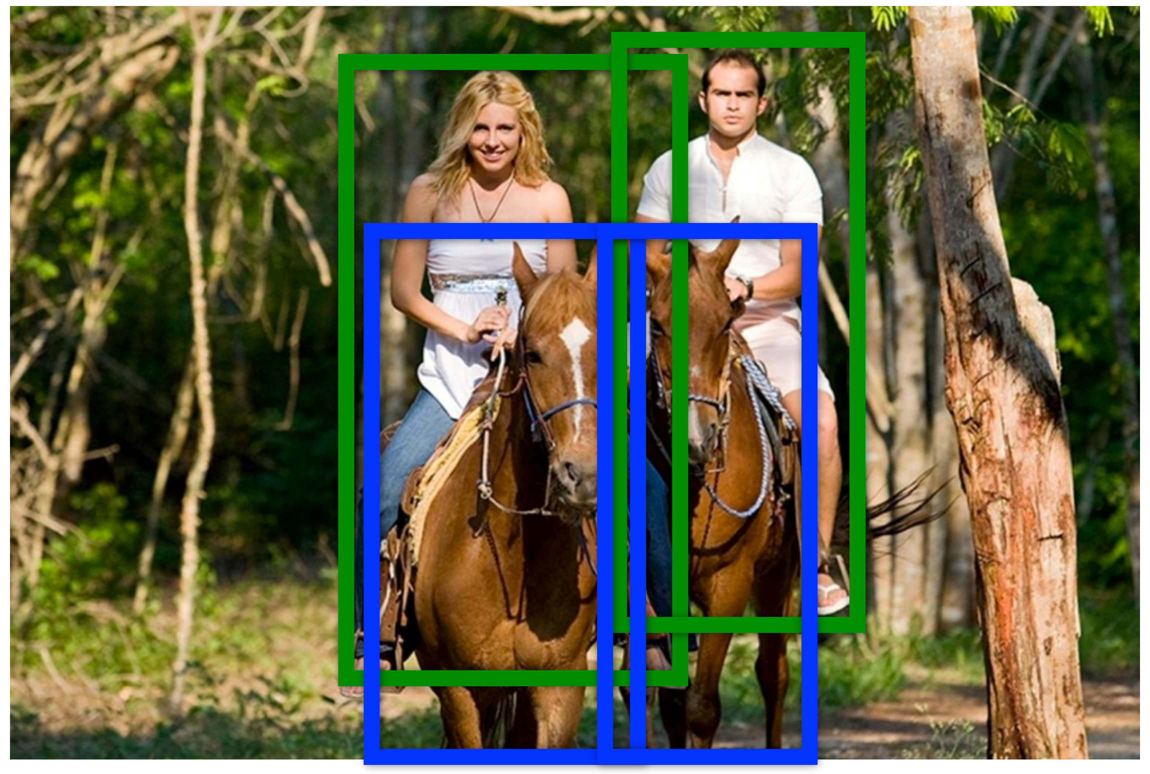
Categorization

Multi-**class:** Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**: **Horse**
Church
Toothbrush
Person

Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO Common Objects in Context

Segmentation

Horse
Person

COCO Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2