THE UNIVERSITY OF BRITISH COLUMBIA

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

**Lecture 7: Convolutional Neural Networks (part 4)**

# Logistics:

**Assignment 2** due **Monday**

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

Categorization

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

## Categorization

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

IMAGENET

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

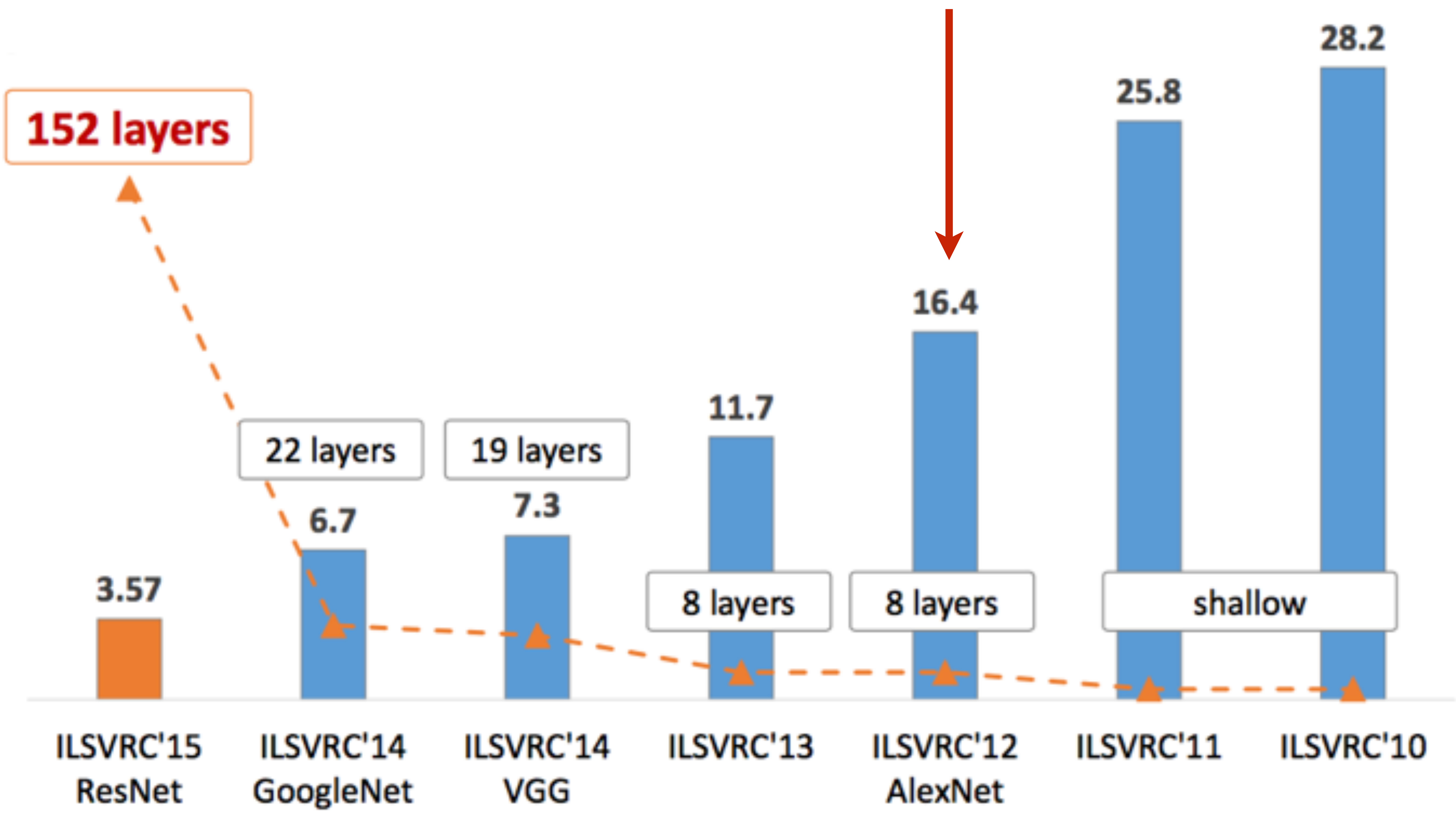
## Categorization

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

IMAGENET

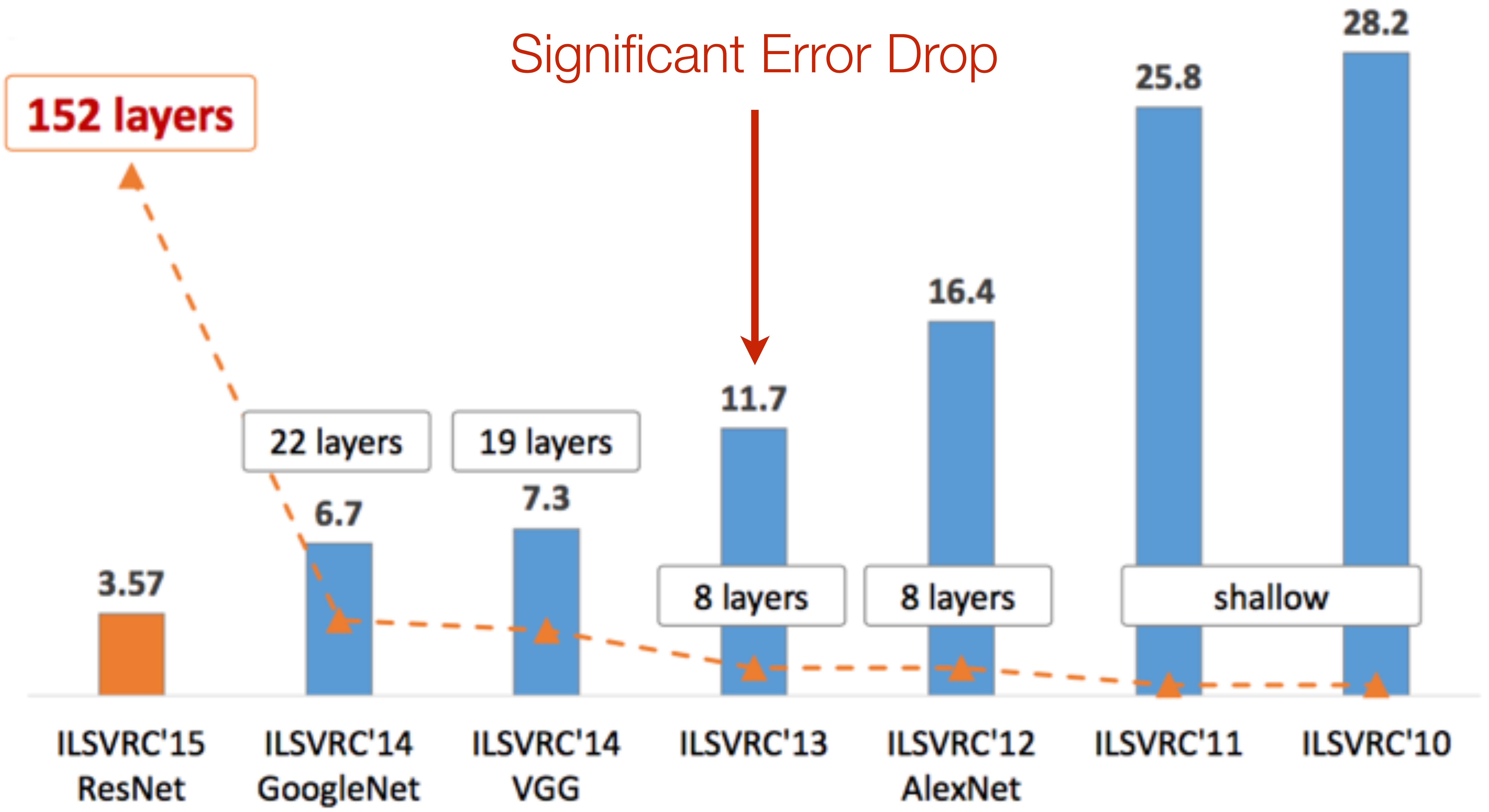
Multi-**label**: **Horse**
Church
Toothbrush
**Person**

# **ILSVRC** winner 2012



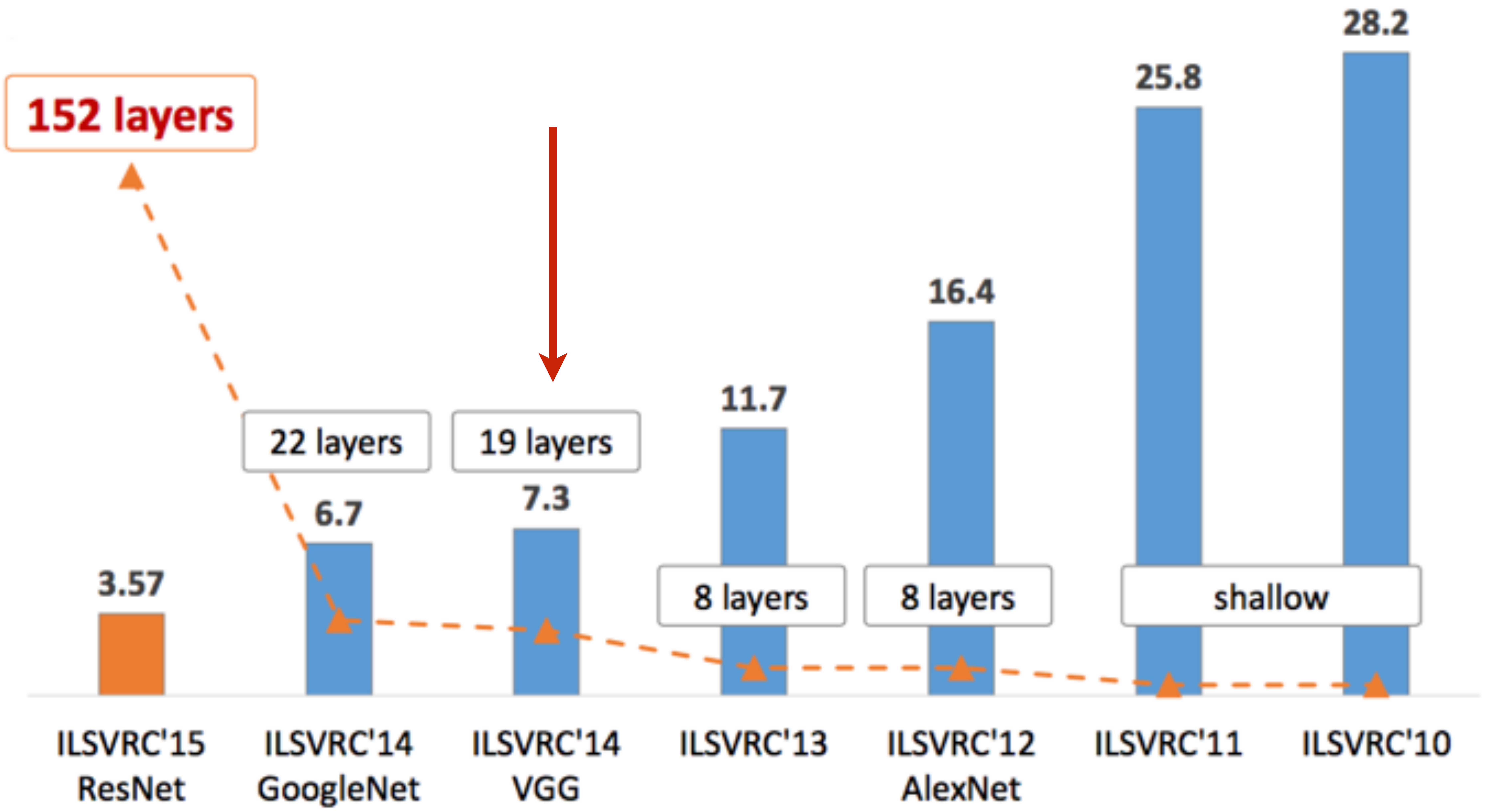

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# **ILSVRC** winner 2012




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

# **ILSVRC** winner 2012

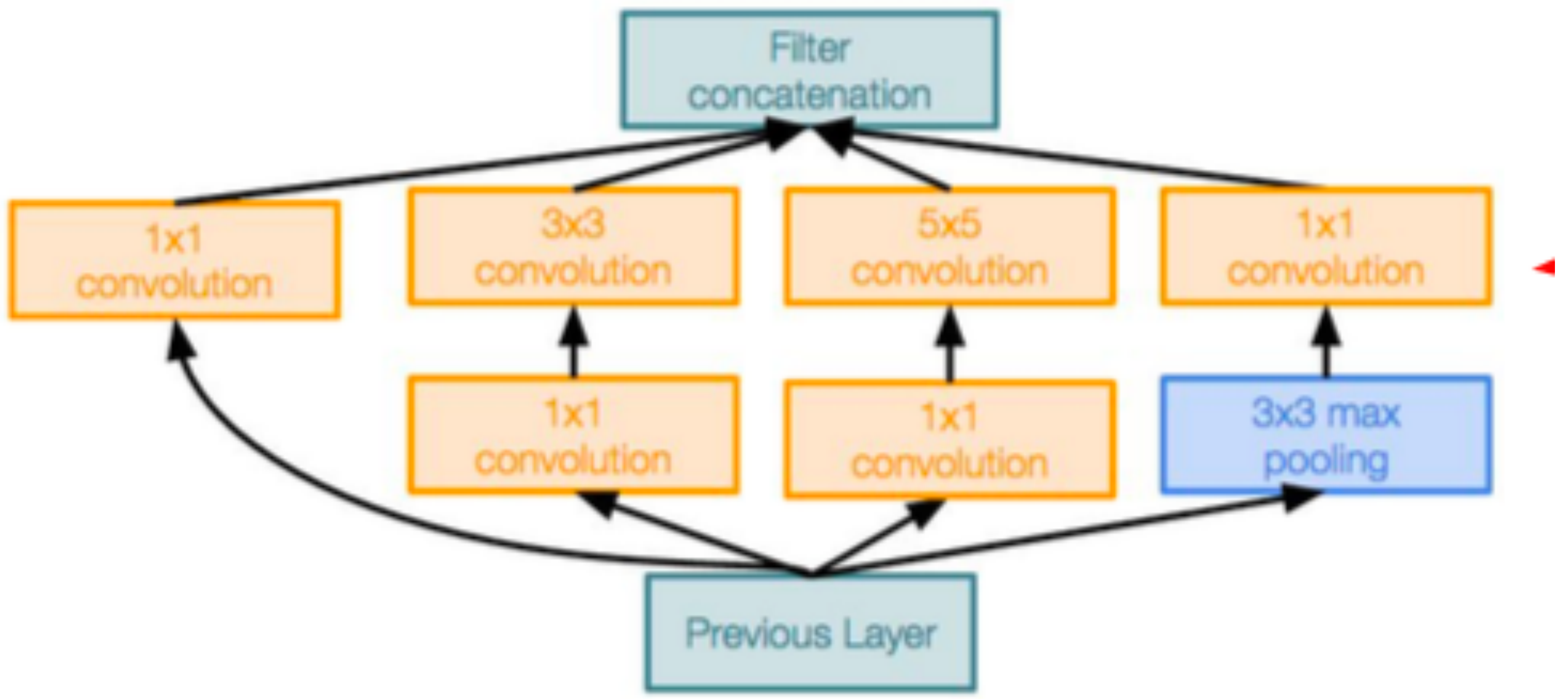



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# **Google**LeNet: Inception Module

[ Szegedy et al., 2014 ]

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




Inception module

# **Google**LeNet

[ Szegedy et al., 2014 ]

even deeper network with **computational efficiency**

— 22 layers

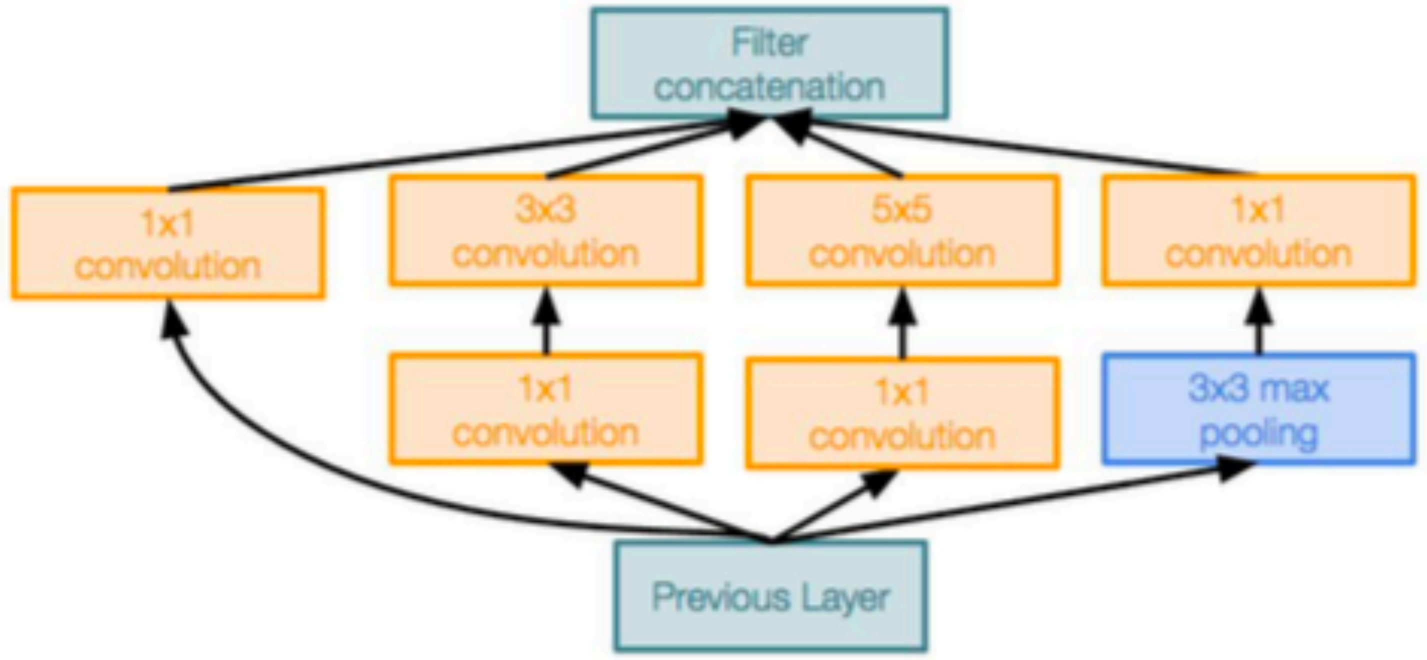
— Efficient “Inception” module

— No FC layers

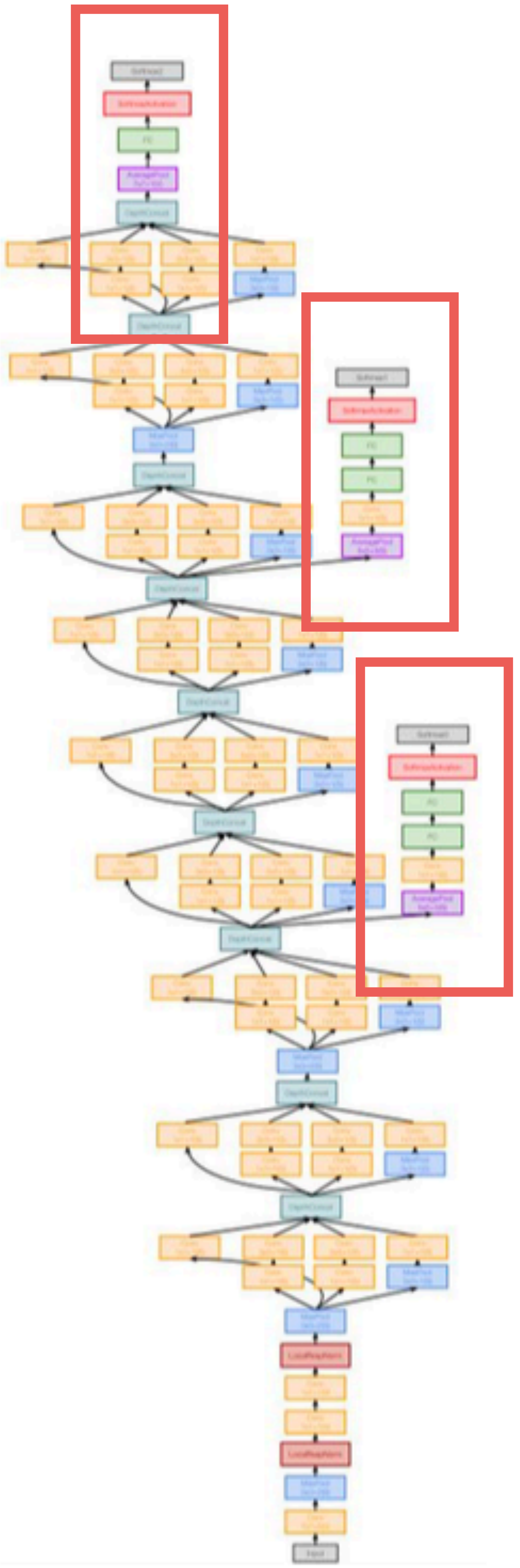
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)

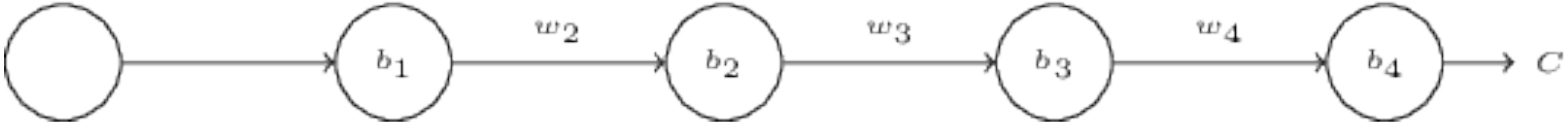



Inception module



# Optimizing **Deep** Neural Networks

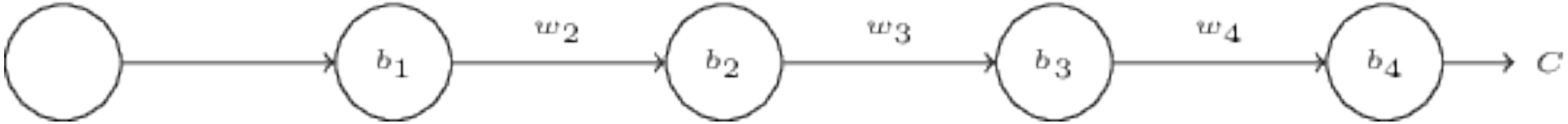
Consider multi-layer neural network with sigmoid activations and loss C




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

# Optimizing **Deep** Neural Networks

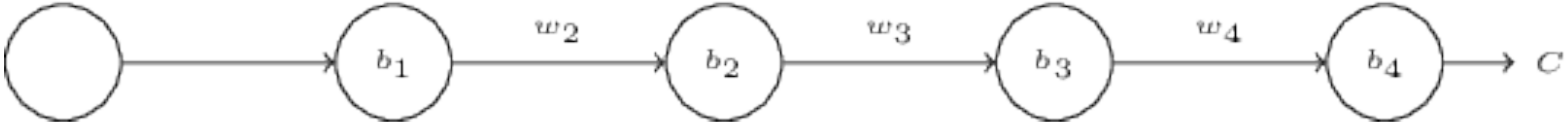
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$




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

# Optimizing **Deep** Neural Networks

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



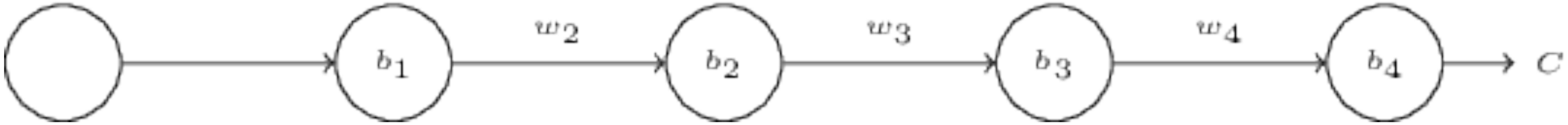

Expression for **gradient** of bias in **Layer 1**: $\frac{\partial C}{\partial b_1} = \sigma'(z_1)\, w_2\sigma'(z_2)\, w_3\sigma'(z_3)\, w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}$

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

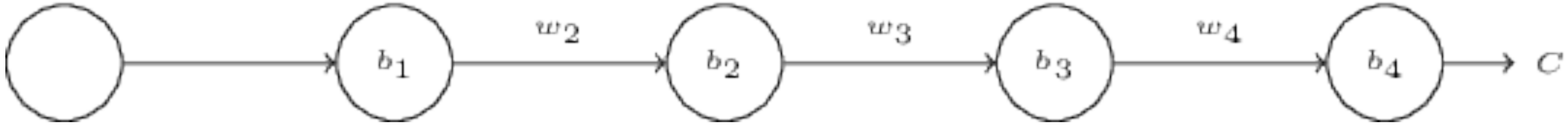
common terms

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**:

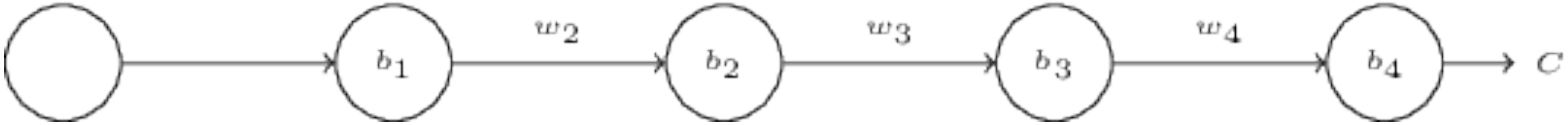
|weight| < 1 (due to initialization)

max of derivative of sigmoid = 1/4 @ 0

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

# Optimizing **Deep** Neural Networks

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




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

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

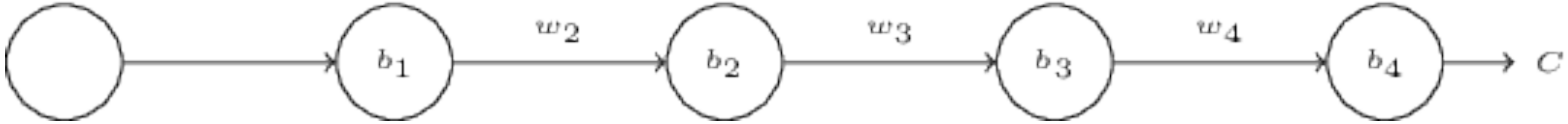
**Observations**:

- |weight| < 1 (due to initialization)
- max of derivative of sigmoid = 1/4 @ 0

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

# Optimizing **Deep** Neural Networks

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




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

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

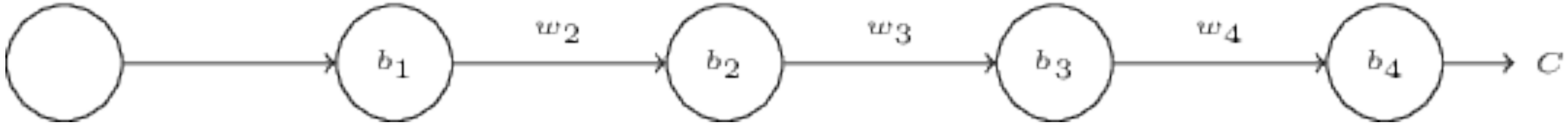
This is called **vanishing gradient** problem

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

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

# Optimizing **Deep** Neural Networks

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




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

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

**Exploding gradient** problem

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

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

# **Google**LeNet

[ Szegedy et al., 2014 ]

even deeper network with **computational efficiency**

— 22 layers

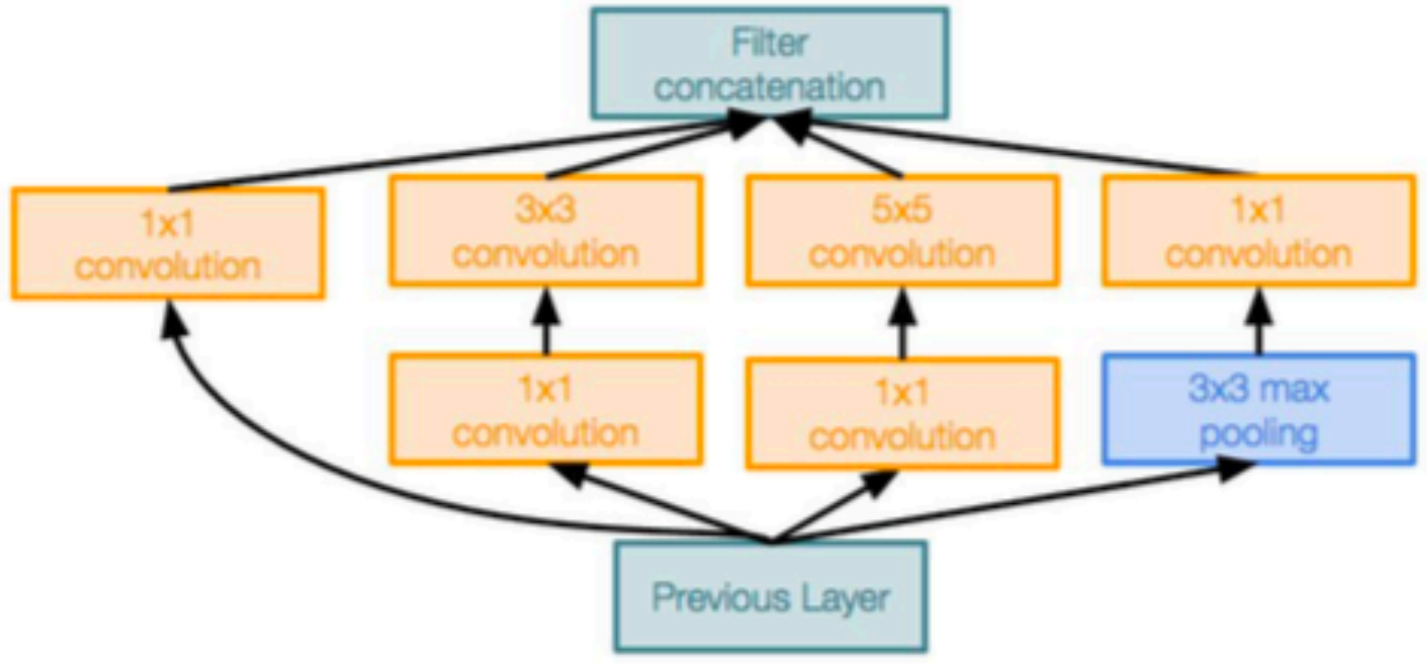
— Efficient “Inception” module

— No FC layers

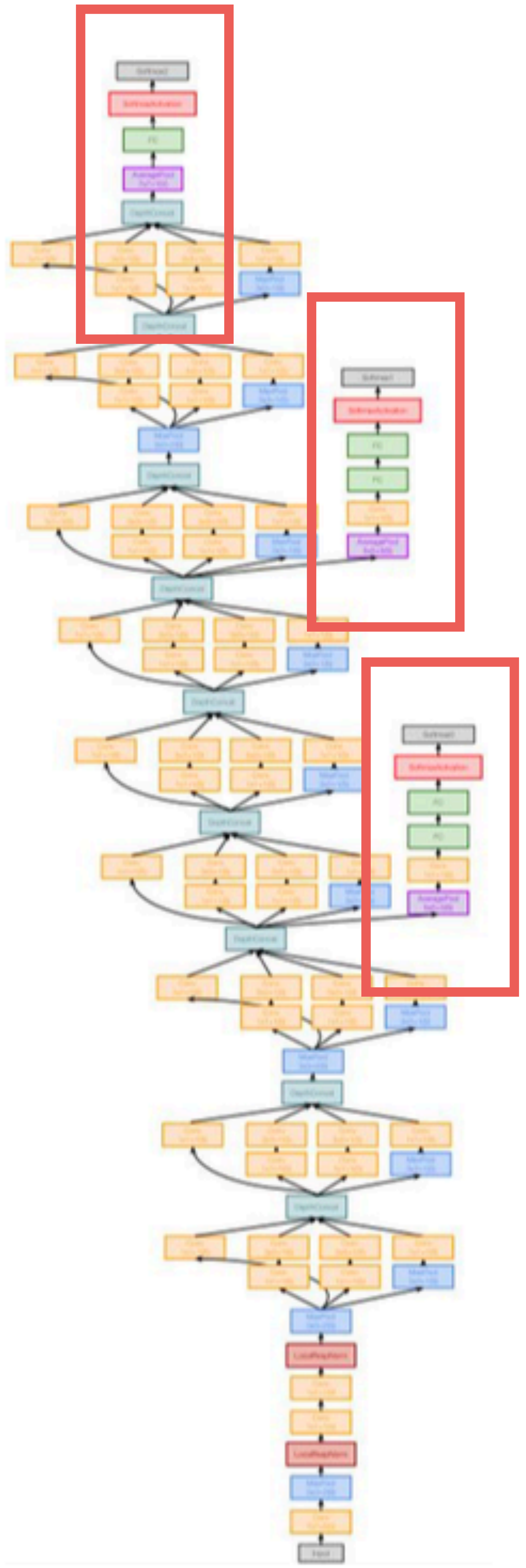
— Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)



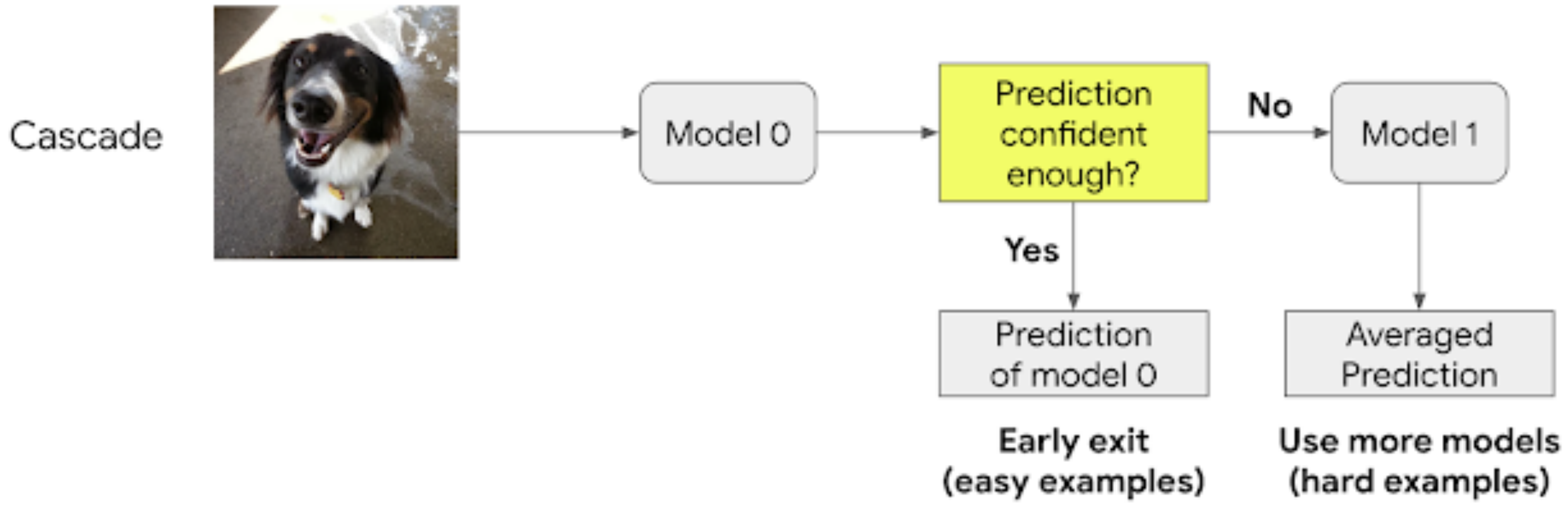

**Inception module**

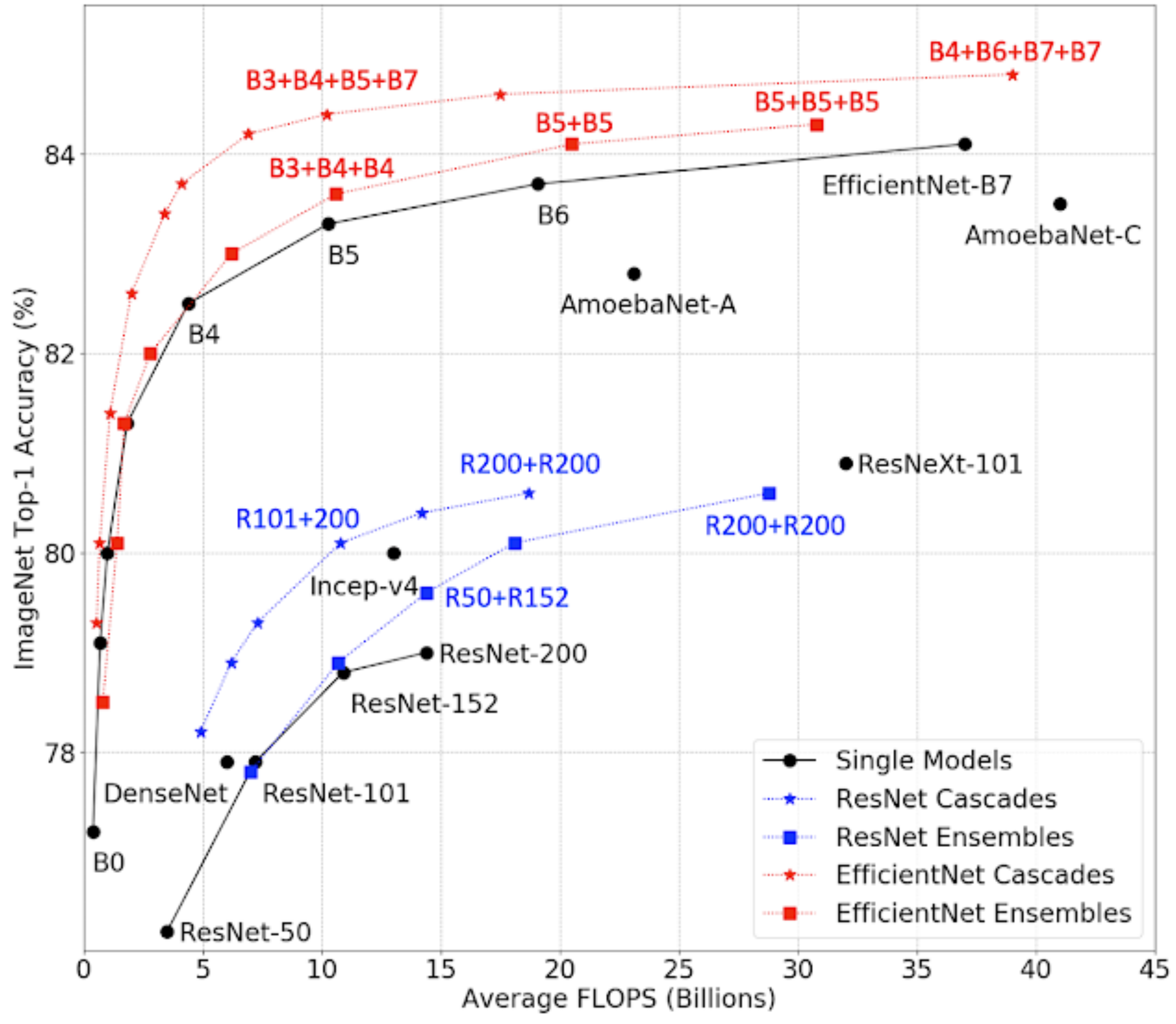


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

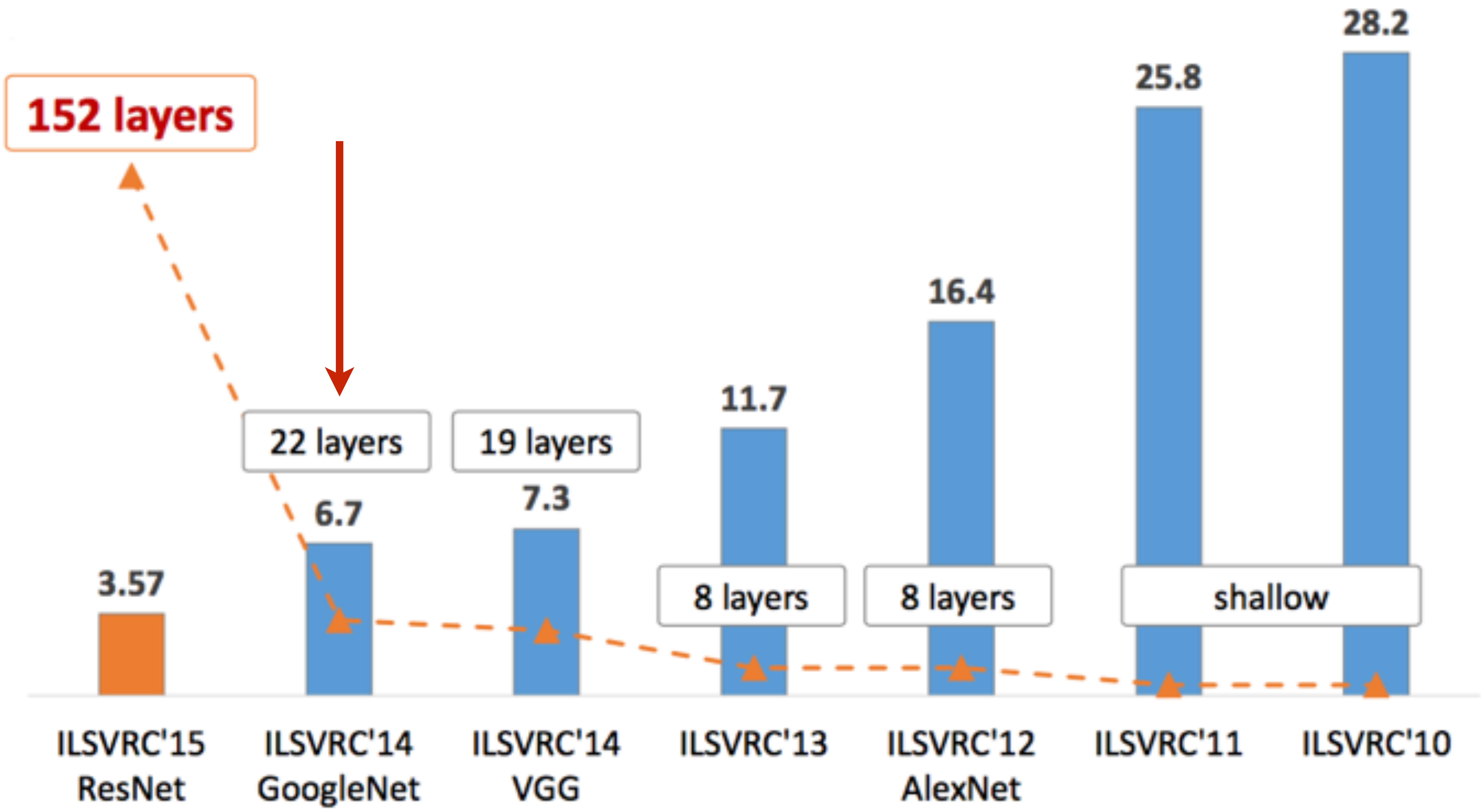
# An Aside: Neural Network **Cascades**

[ Wang et al., ICLR 2022 ]



# **ILSVRC** winner 2012



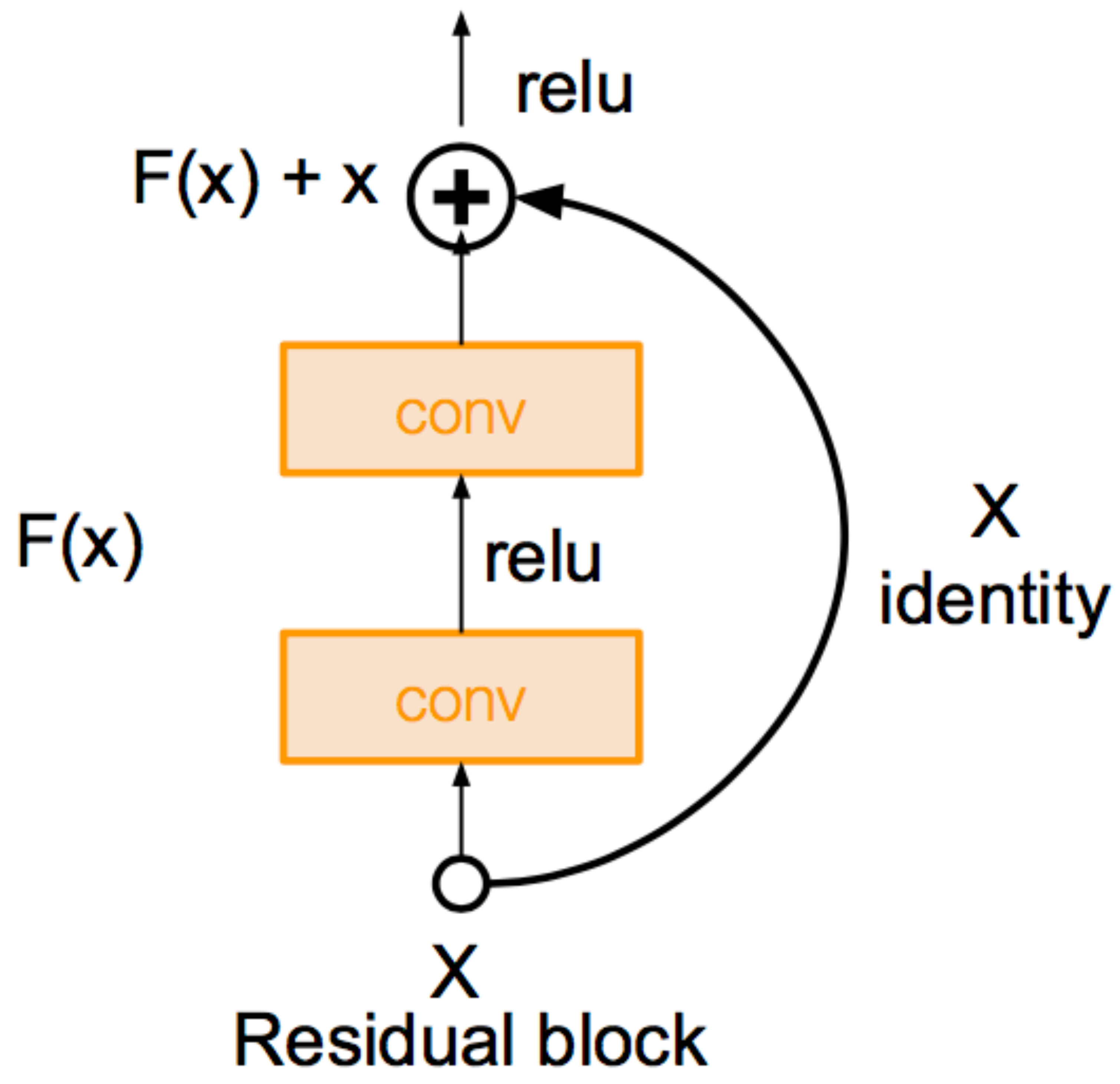

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

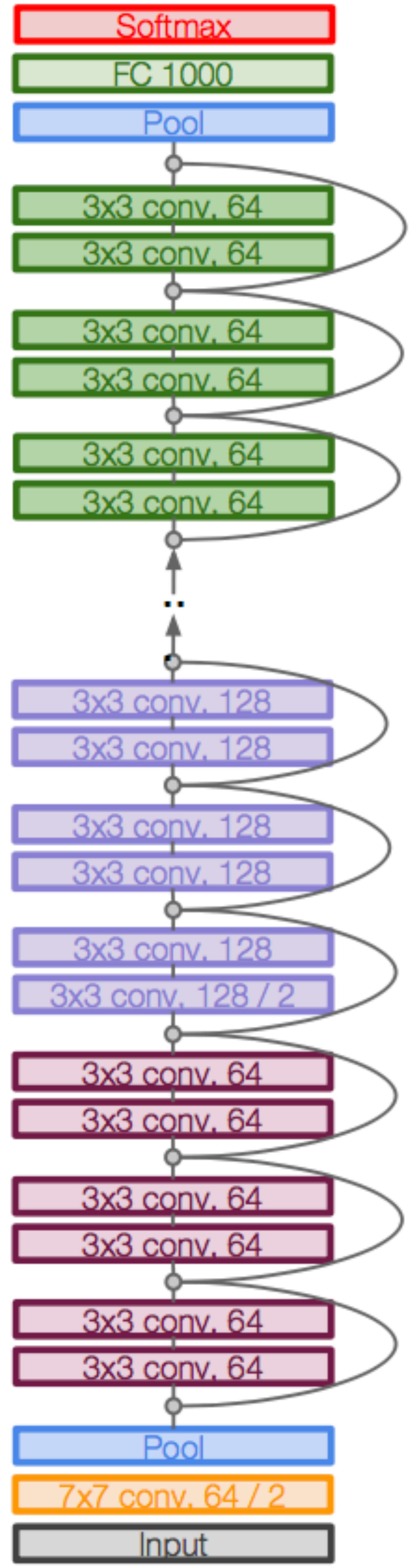
# ResNet

[ He et al., 2015 ]

even deeper — **152 layers**!

using residual connections



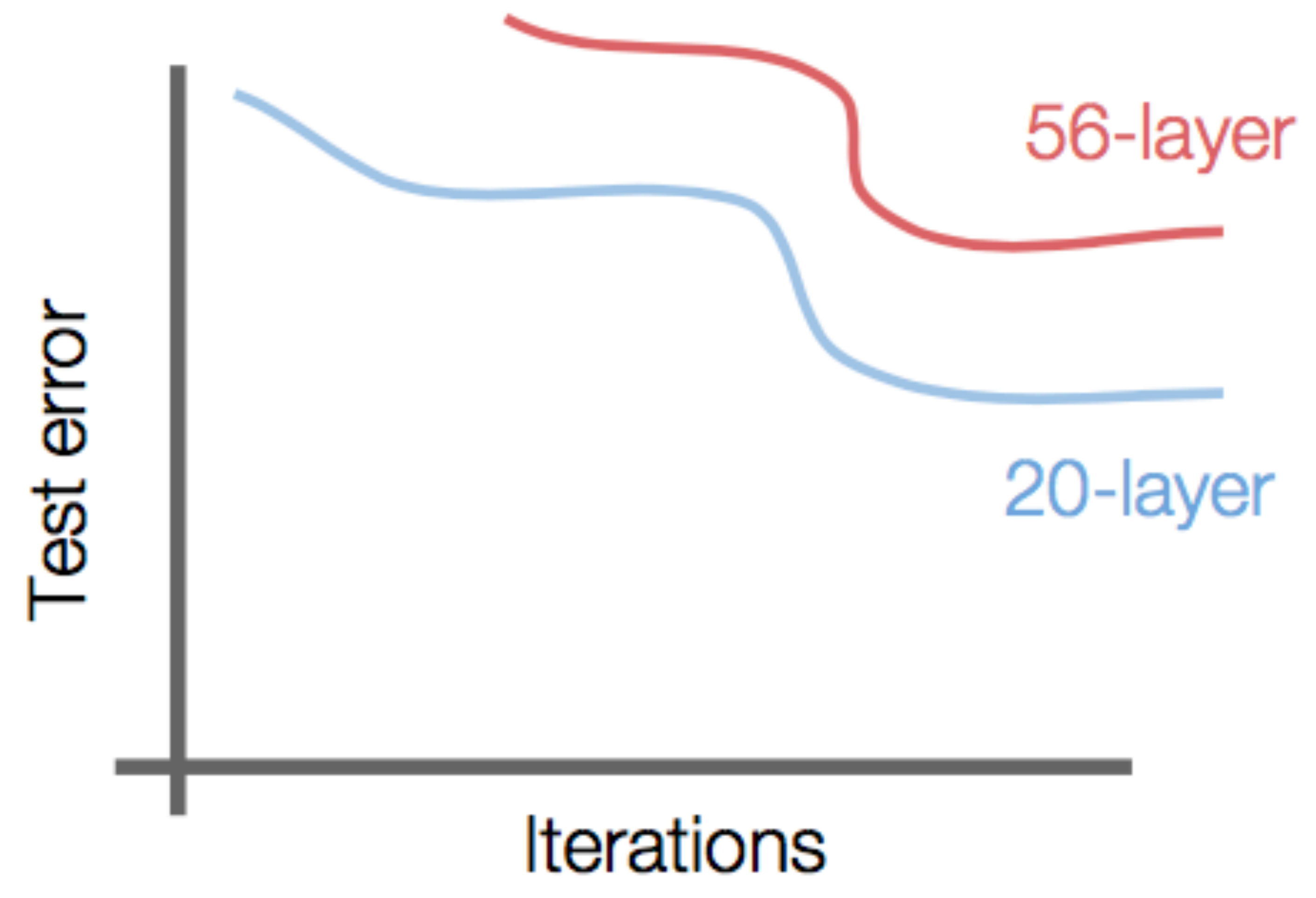




* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# **ResNet:** Motivation

[ He et al., 2015 ]

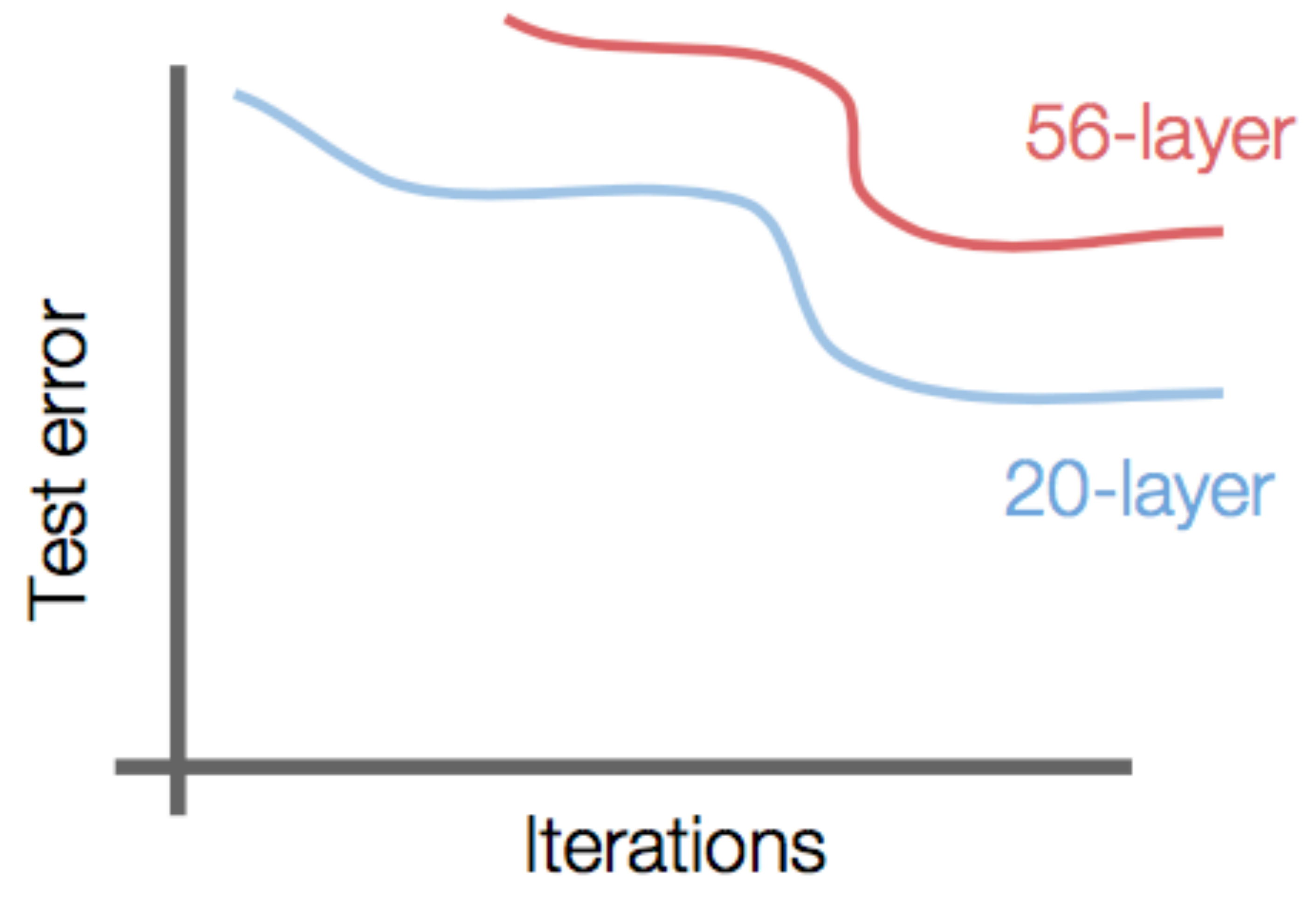
What happens when we continue to stacking deeper layers on a “plain” CNN



# **ResNet:** Motivation

[ He et al., 2015 ]

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



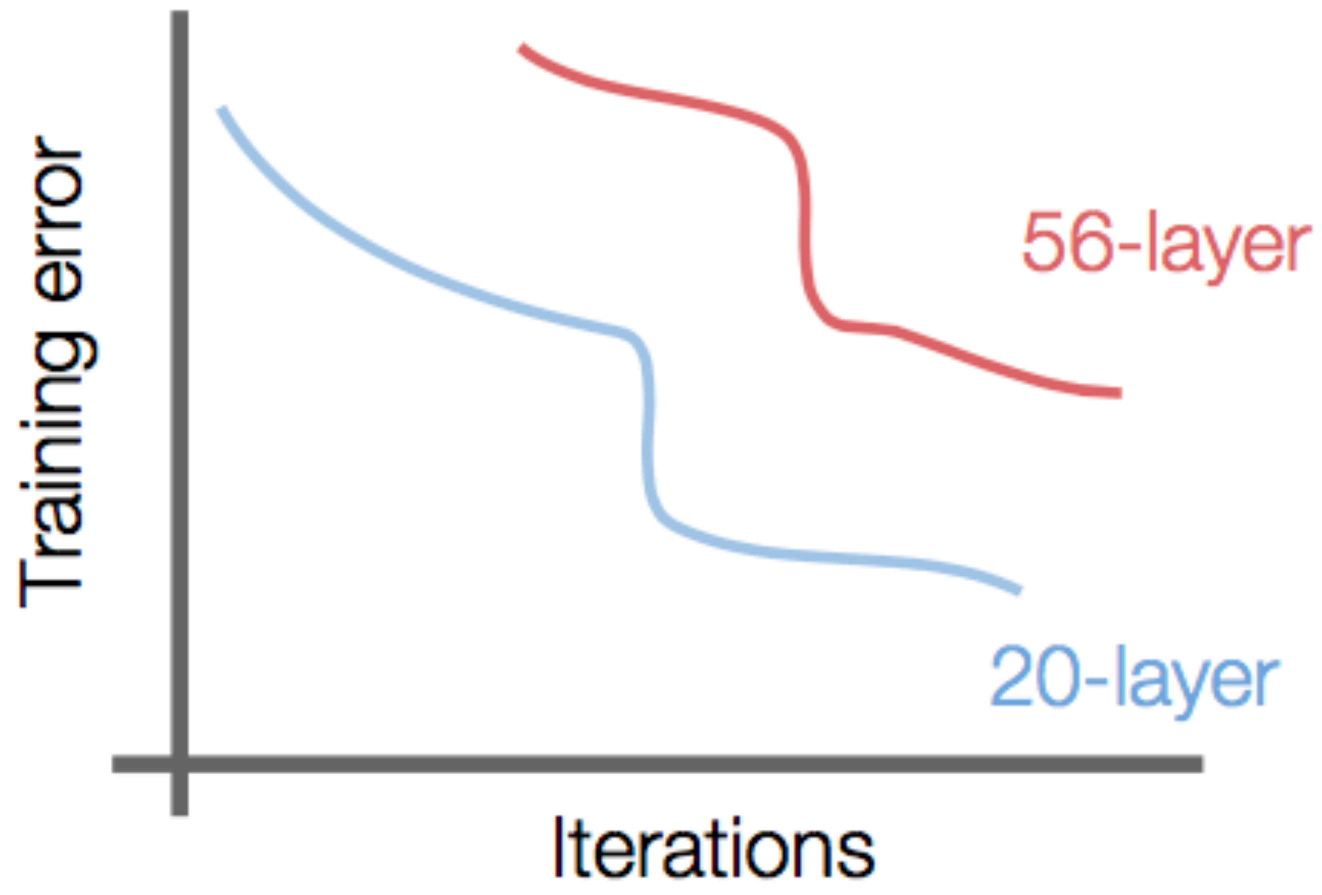

Whats the **problem**?

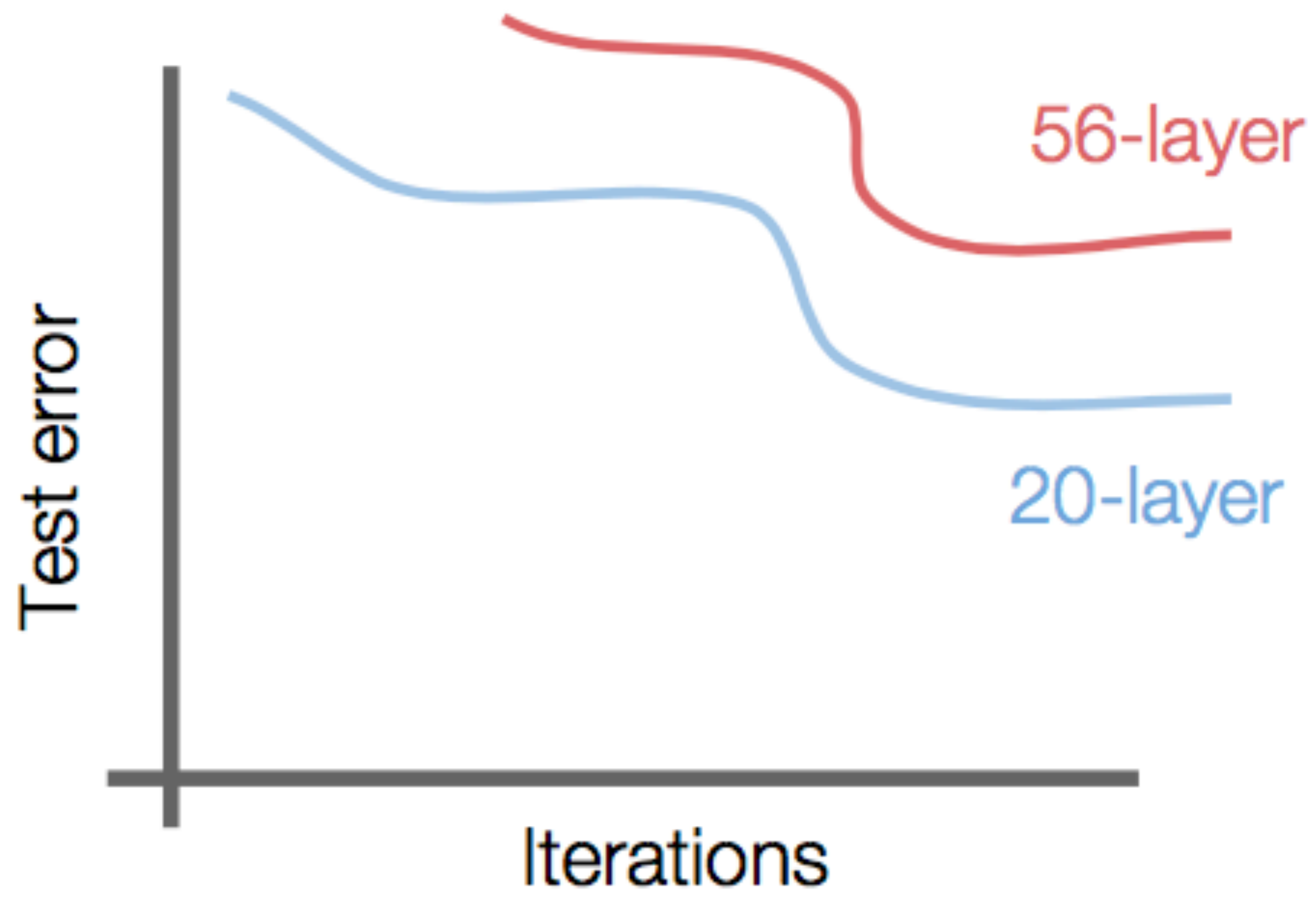
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# **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

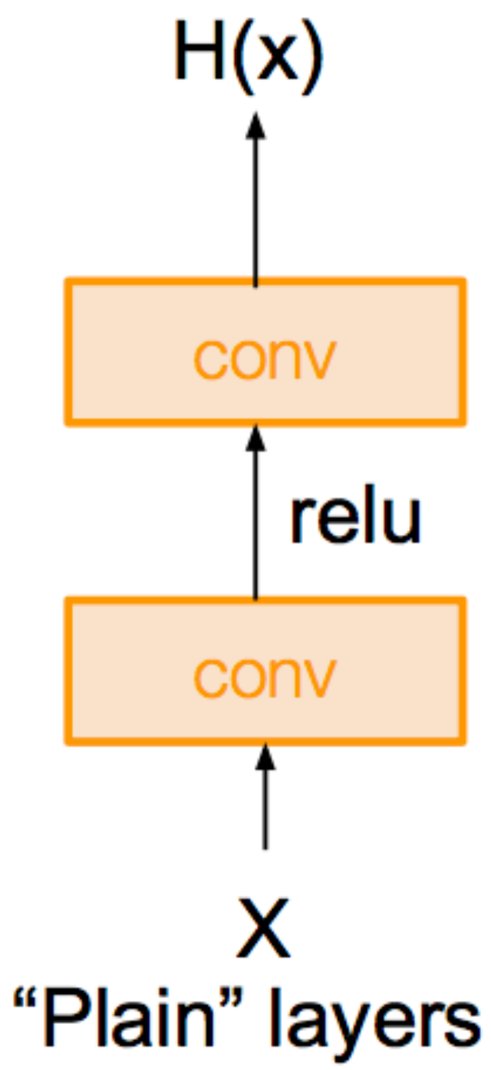
# ResNet

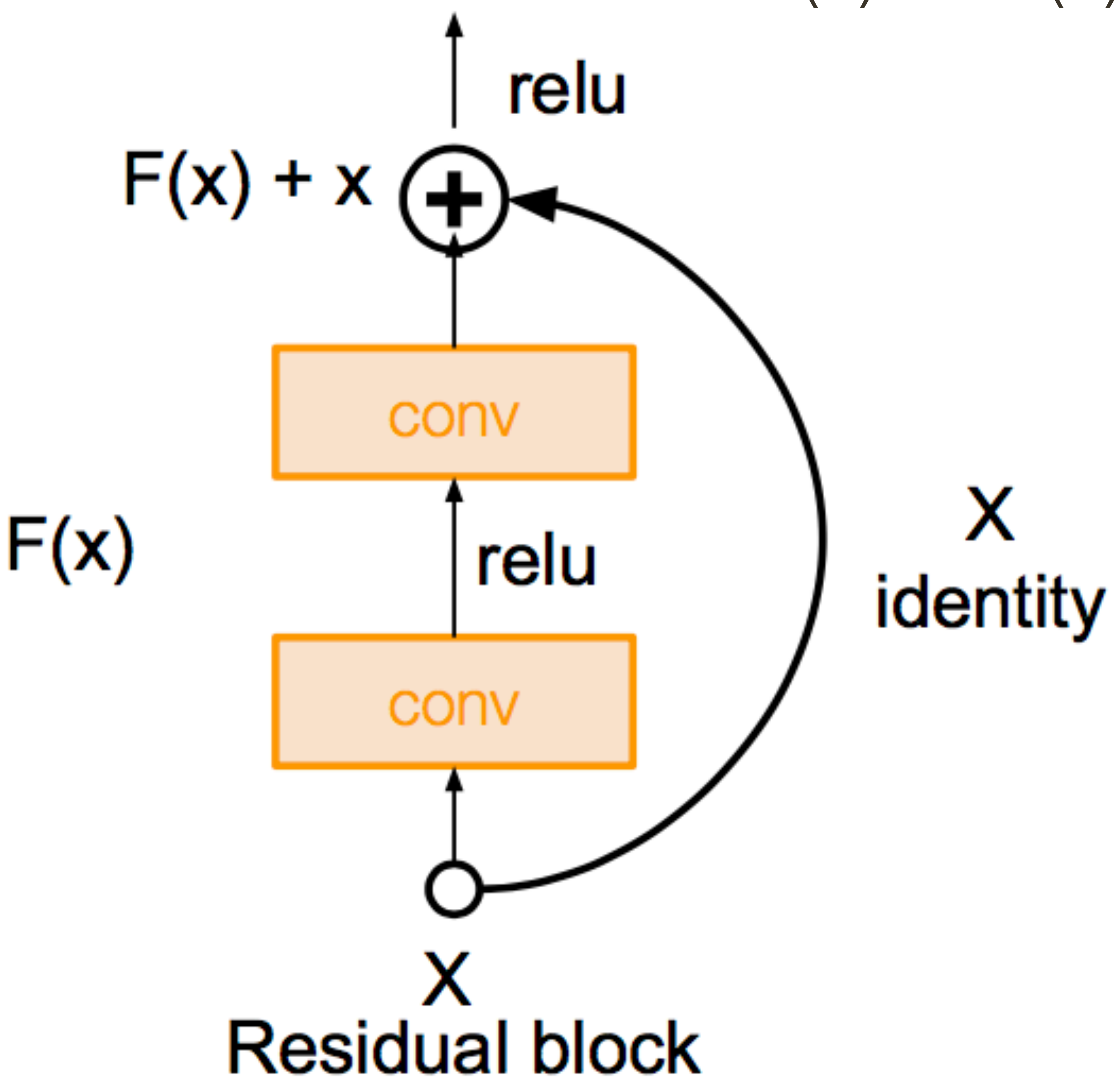
[ He et al., 2015 ]

**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



# ResNet

[ He et al., 2015 ]

Full details

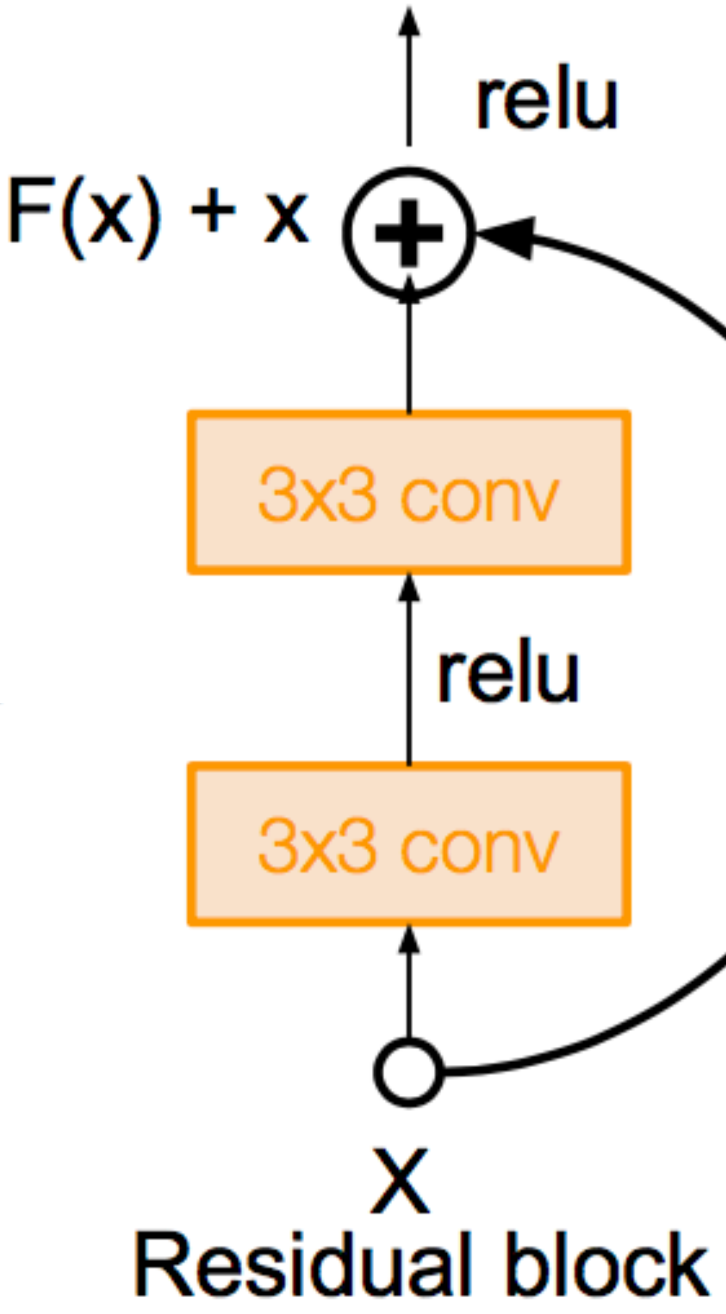
— Stacked **residual blocks**

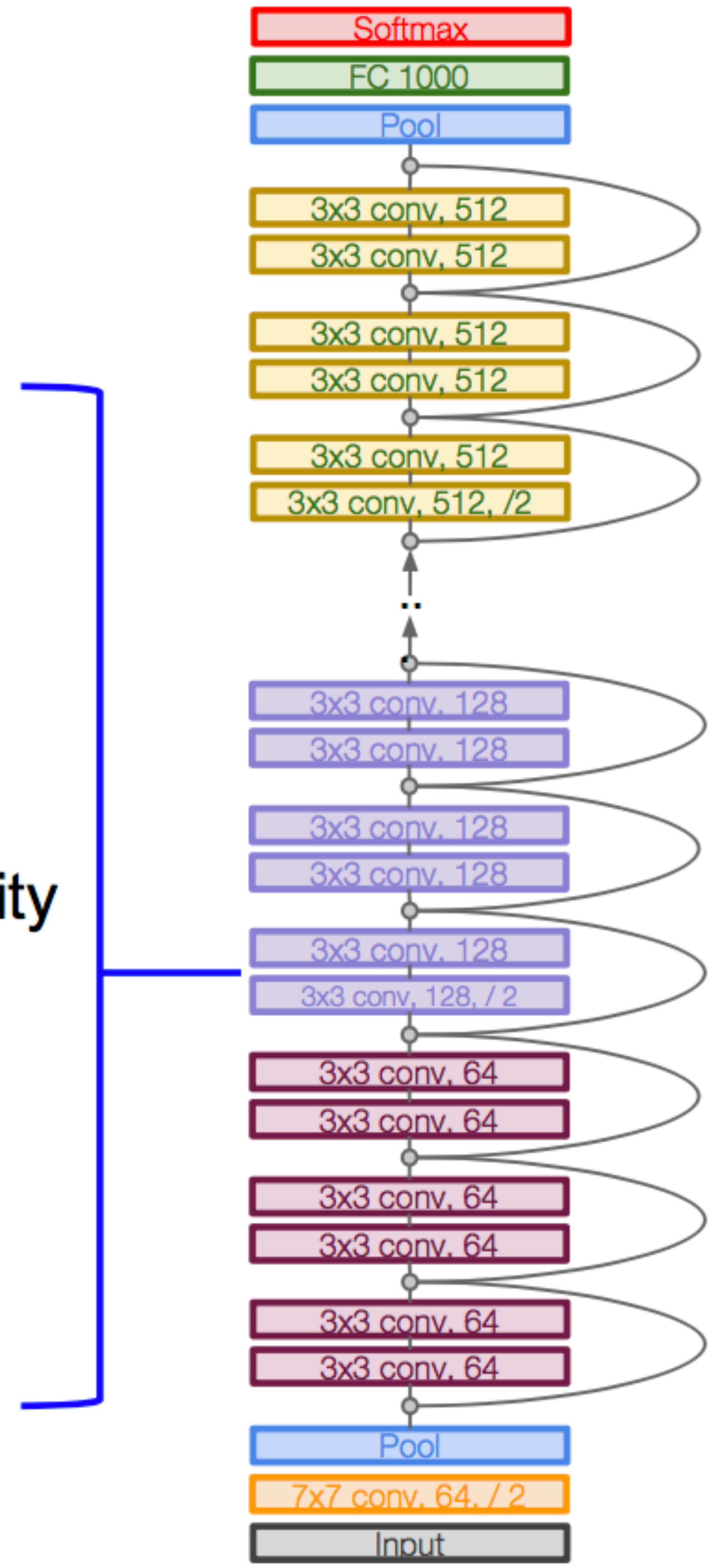
— Every residual block consists of **two 3x3 filters**

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

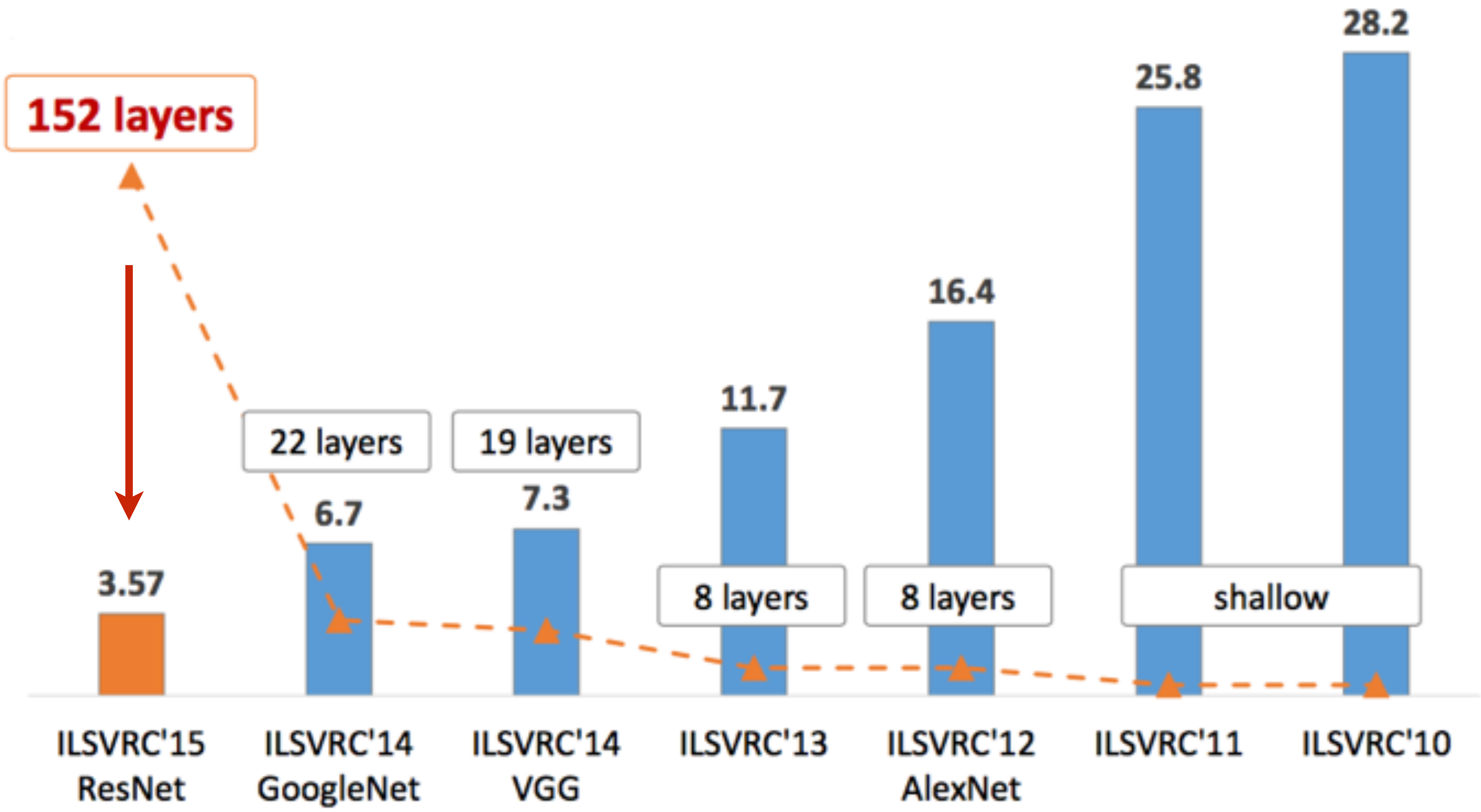
— Additional convolutional layer in the beginning

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



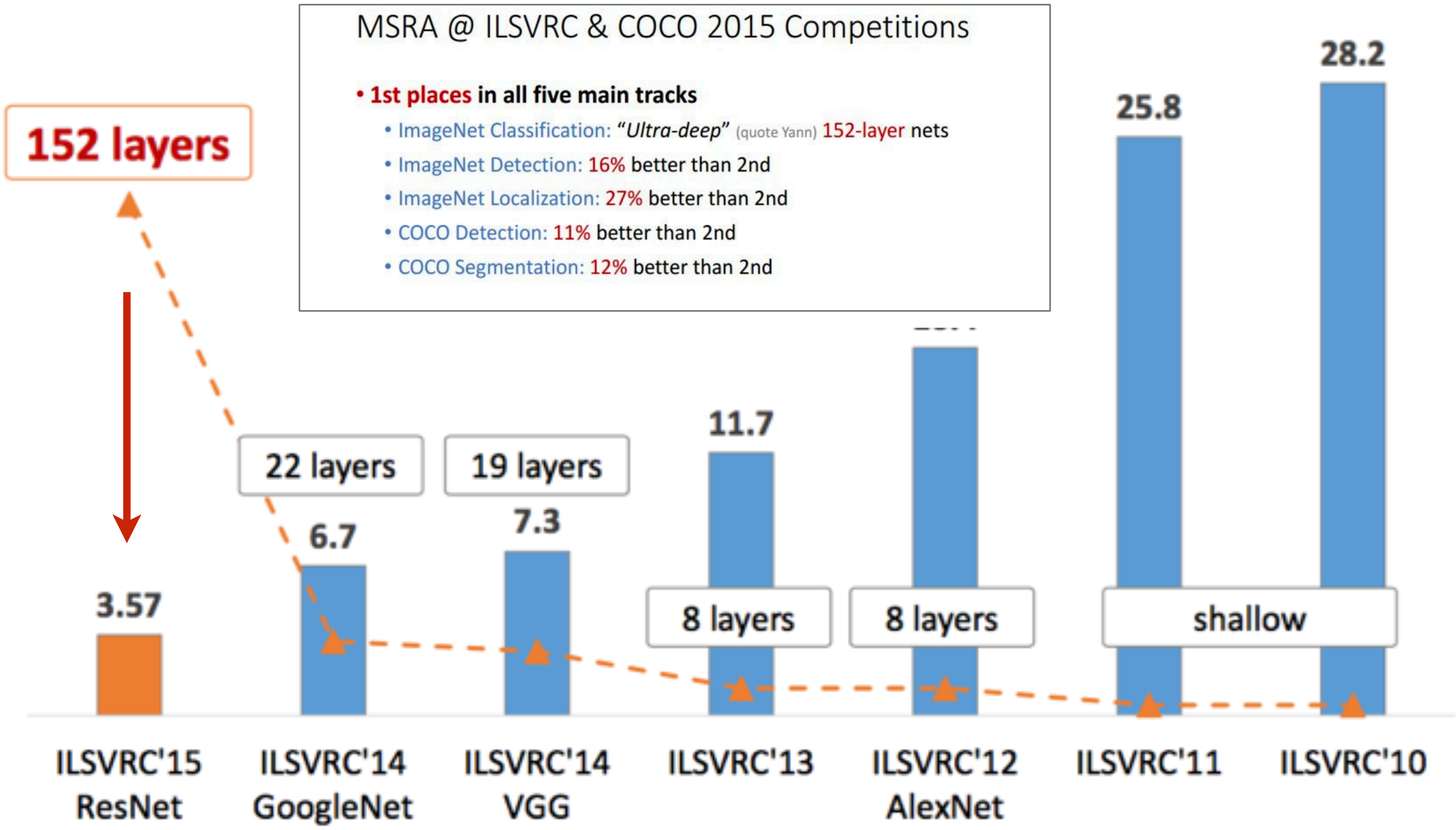



# **ILSVRC** winner 2012




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

# **ILSVRC** winner 2012



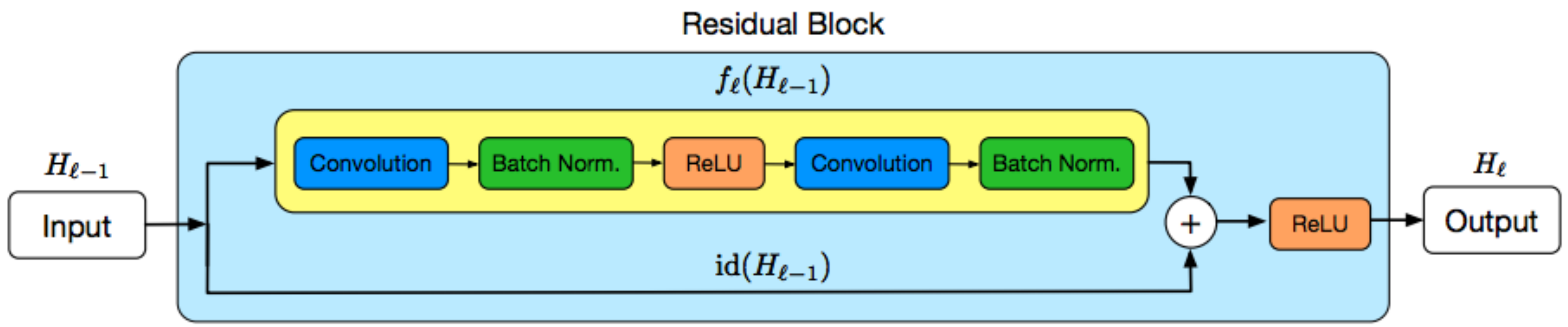
# **Regularization:** Stochastic Depth

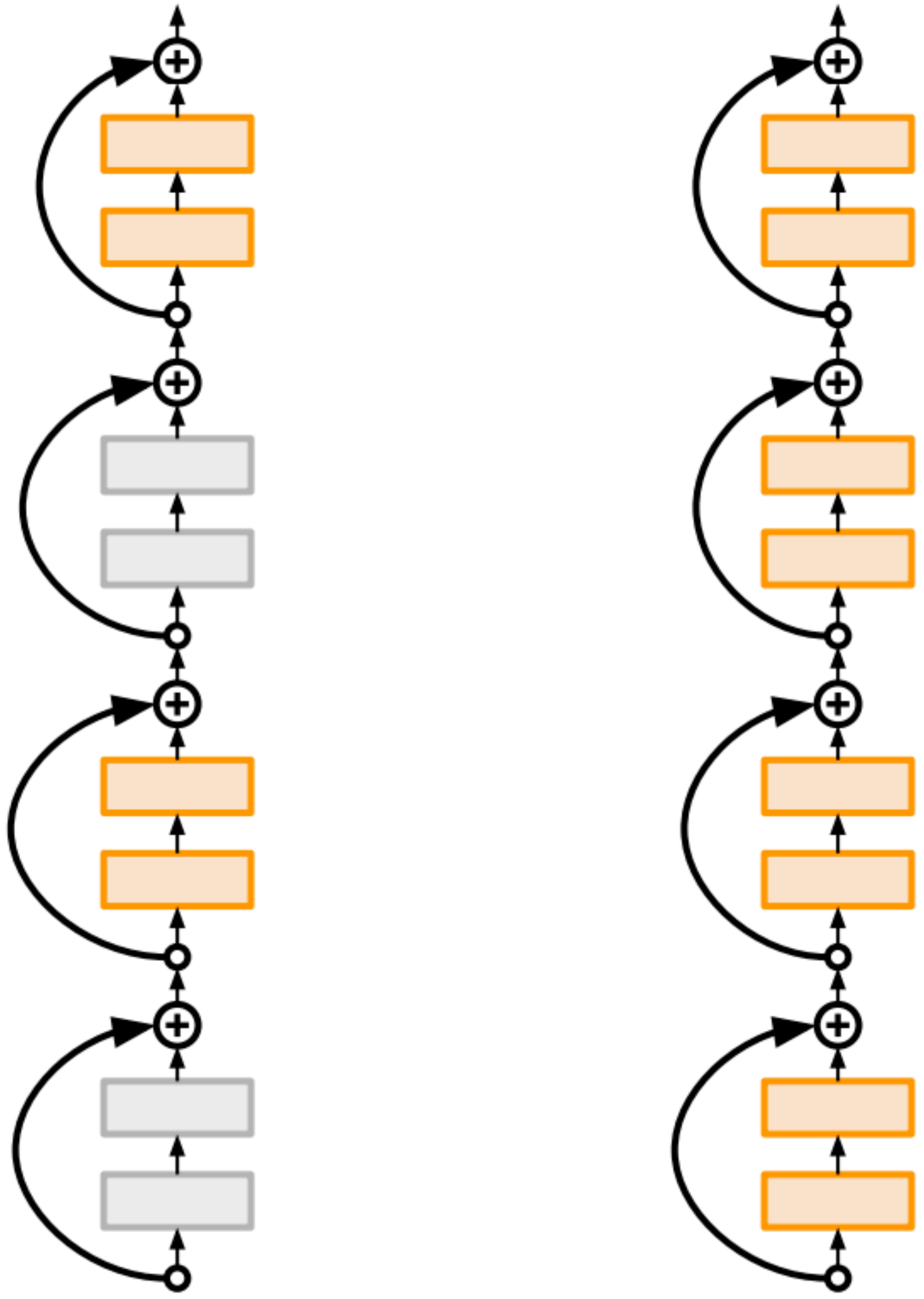
[ Huang et al., ECCV 2016 ]

Effectively "dropout" but for layers

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

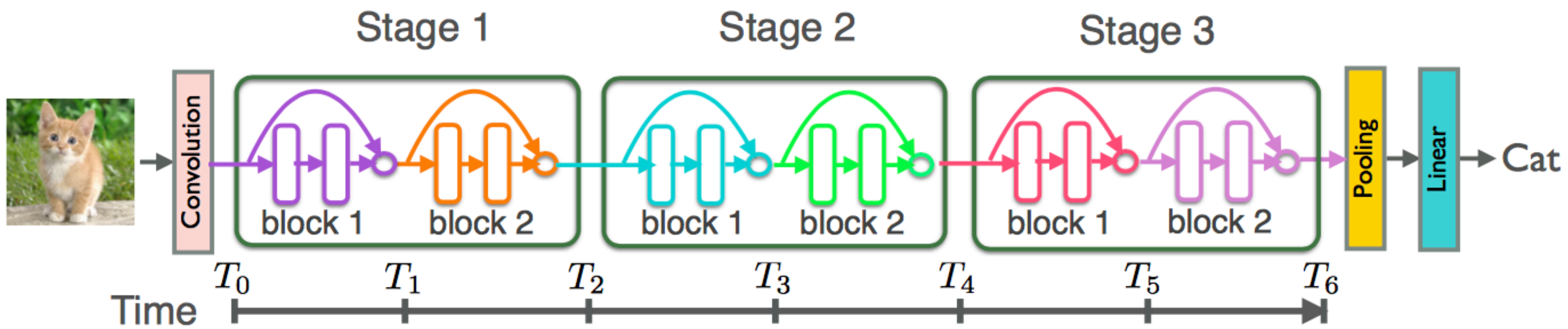
Effectively trains a collection of neural networks



# **ResNet**: A little theory

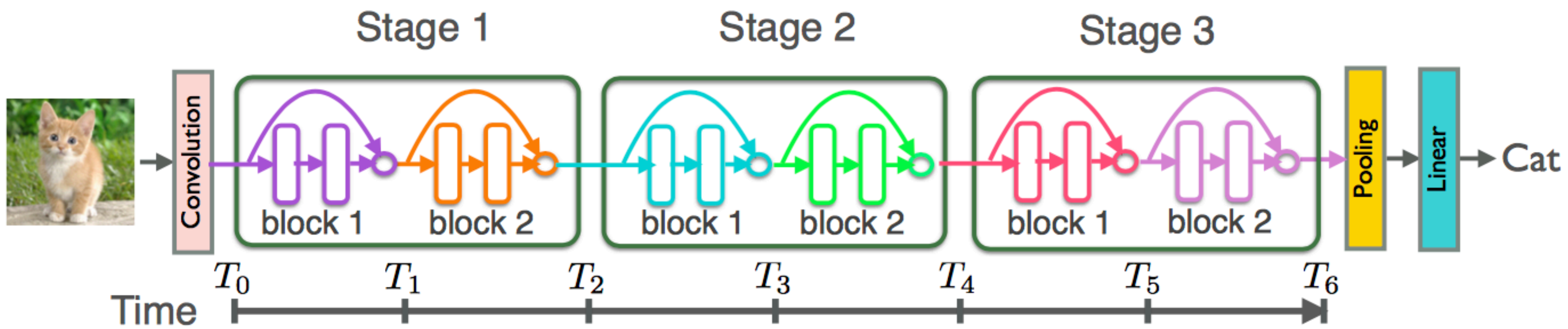
One can view a sequence of outputs from residual layers as a **Dynamical System**



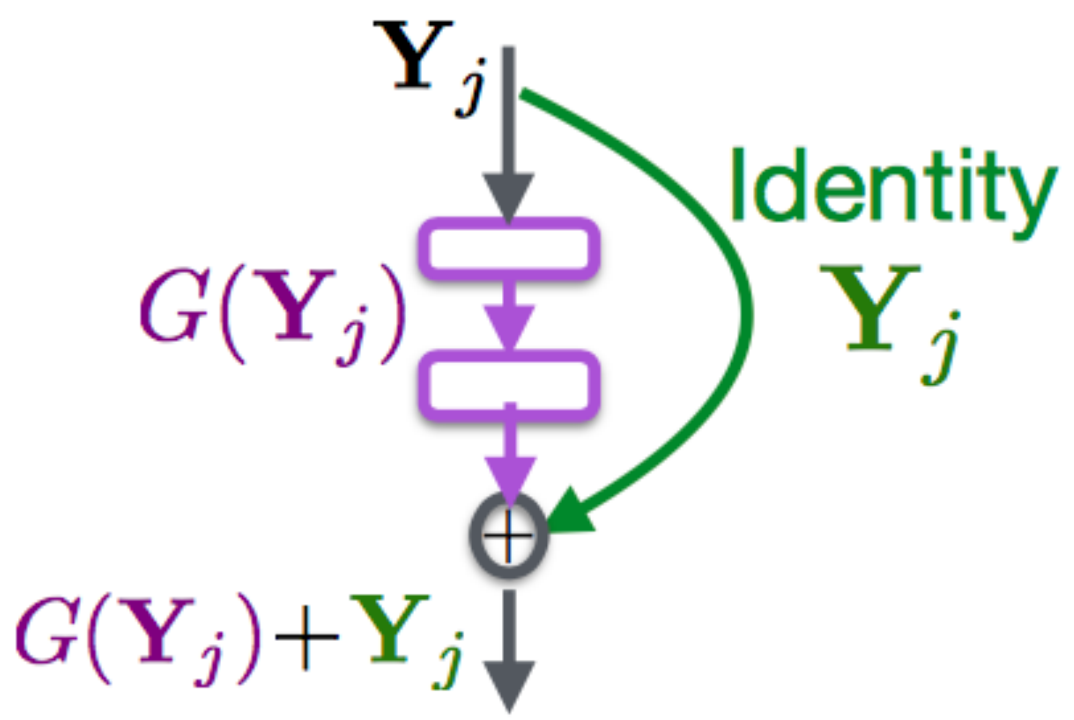

[ Cheng et al., ICLR 2018 ]

# **ResNet**: A little theory

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



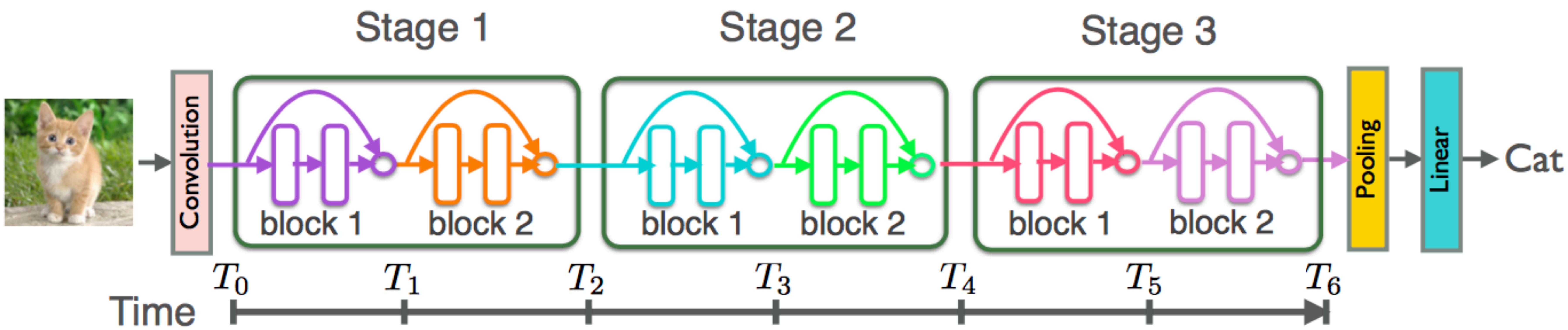

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + G(\mathbf{Y}_j, \boldsymbol{\theta}_j)$$




[ Cheng et al., ICLR 2018 ]

# **ResNet**: A little theory

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

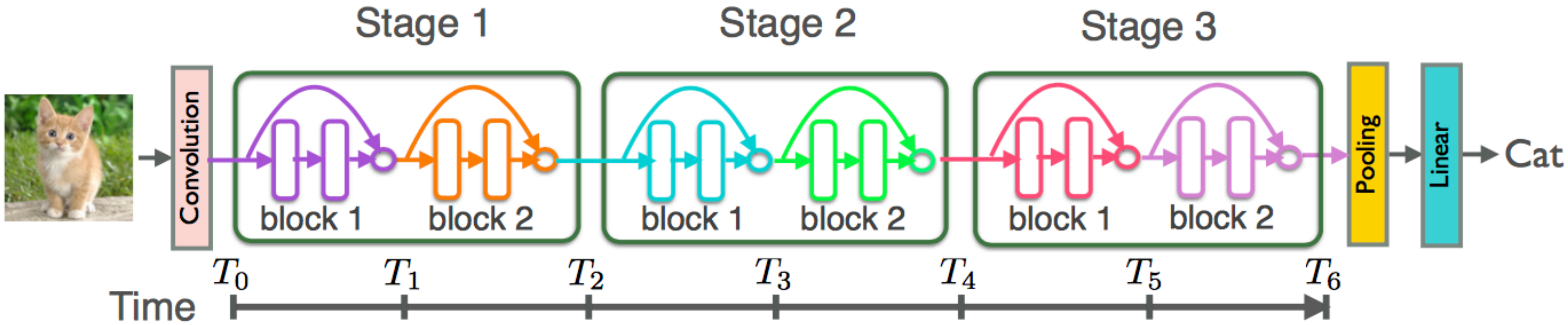



What happens if you take more layers and take smaller steps?

[ 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**

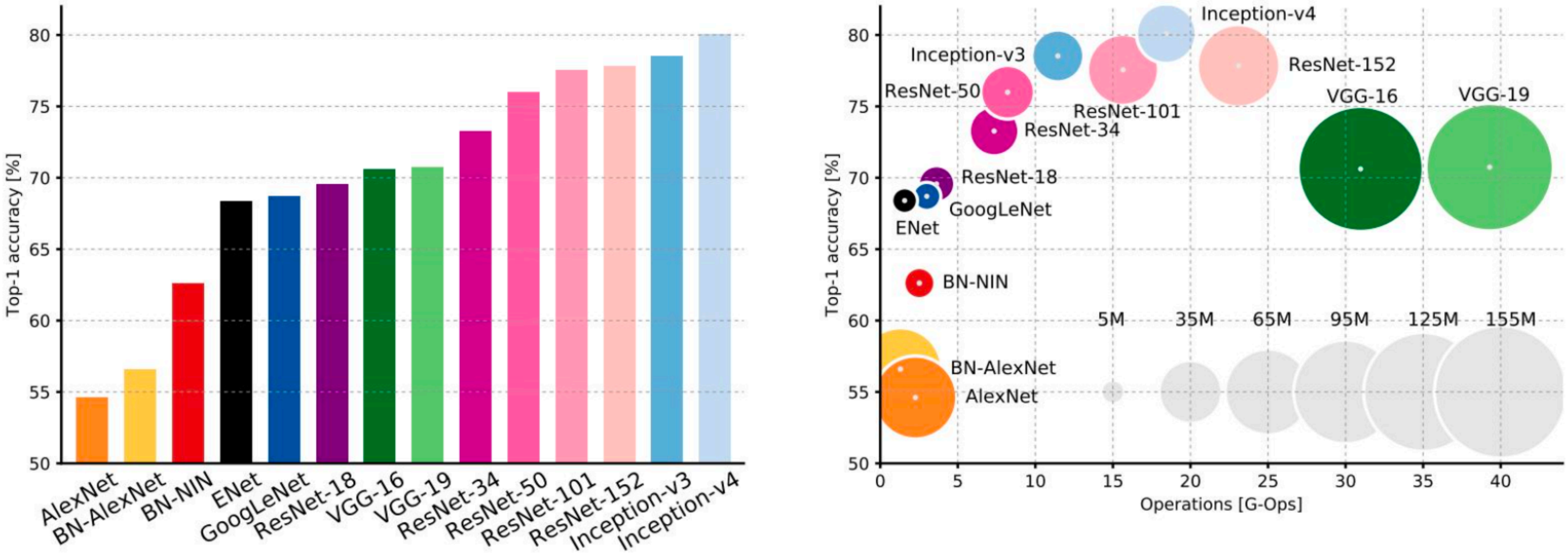



What happens if you take more layers and take smaller steps?

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.

# Neural **Architecture Search**

# Neural **Architecture Search**

Consider finding an optimal neural cell (with a total number of layers = N and number of possible operations = K). The space of all possible architectures is:

$$K^N 2^{N\times(N-1)/2}$$

**Note:** K may include operator types (conv, pool), kernel sizes (1x1, 3x3, 5x5), strides (1, 3, 5), etc.

# Neural **Architecture Search**

Consider finding an optimal neural cell (with a total number of layers = N and number of possible operations = K). The space of all possible architectures is:

$$K^N 2^{N \times (N-1)/2}$$

**Random** Search:

1. Randomly pick operators and connectivity of the graph
2. Evaluate how good is this cell architecture (this is REALLY expensive)
3. Keep best performing architecture

# Neural **Architecture Search**

Consider finding an optimal neural cell (with a total number of layers = N and number of possible operations = K). The space of all possible architectures is:

$$K^N 2^{N \times (N-1)/2}$$

**Smarter** Search:

1. Sample operators and connectivity of the graph (initially at random)
2. Evaluate how good is this cell architecture (this is REALLY expensive)
3. Keep all sampled and evaluated architectures <Architecture, Score>

# Neural **Architecture Search**

Consider finding an optimal neural cell (with a total number of layers = N and number of possible operations = K). The space of all possible architectures is:

$$K^N 2^{N \times (N-1)/2}$$

**Smarter** Search:

Policy

1. Sample operators and connectivity of the graph (initially at random)
2. Evaluate how good is this cell architecture (this is REALLY expensive)
3. Keep all sampled and evaluated architectures <Architecture, Score>

Reward

Replay Buffer

discrete optimization problem => **Reinforcement Learning**

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

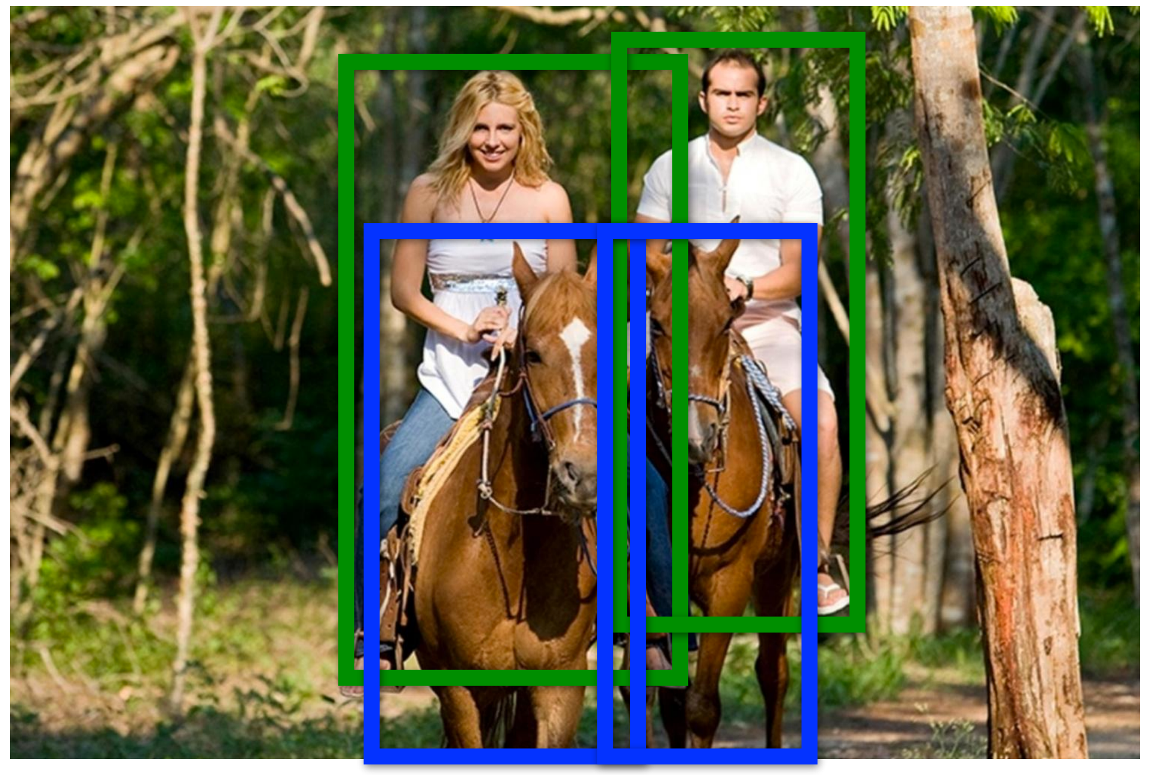
## Categorization

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)

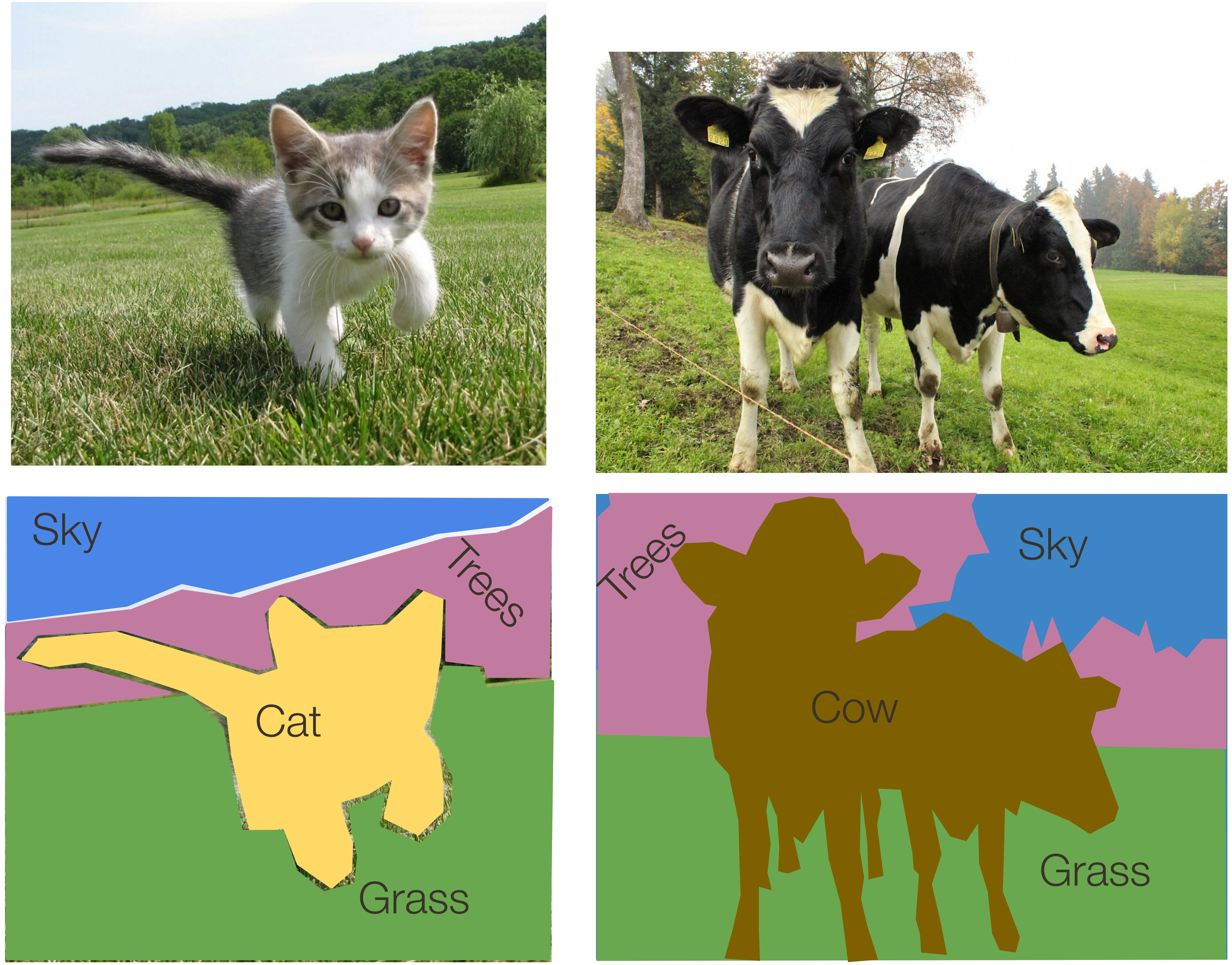
Segmentation

Horse
Person

COCO
Common Objects in Context

# Semantic **Segmentation**

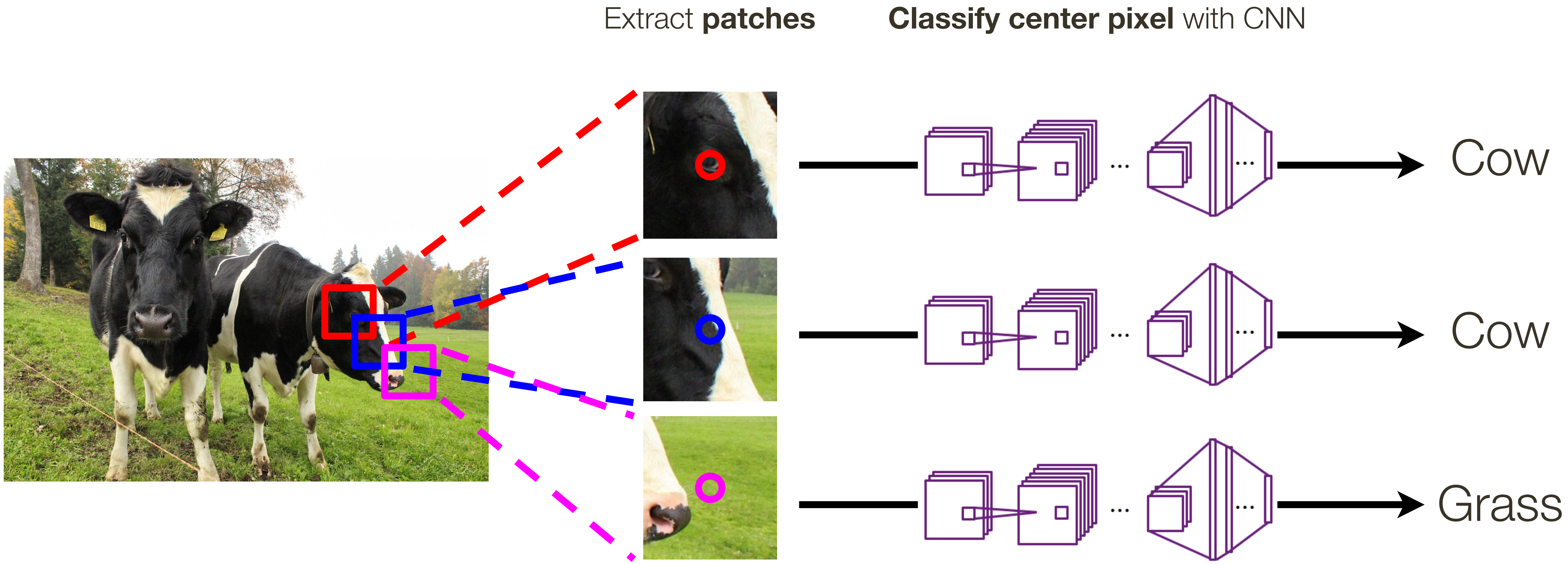
Label **every pixel** with a category label (without differentiating instances)




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

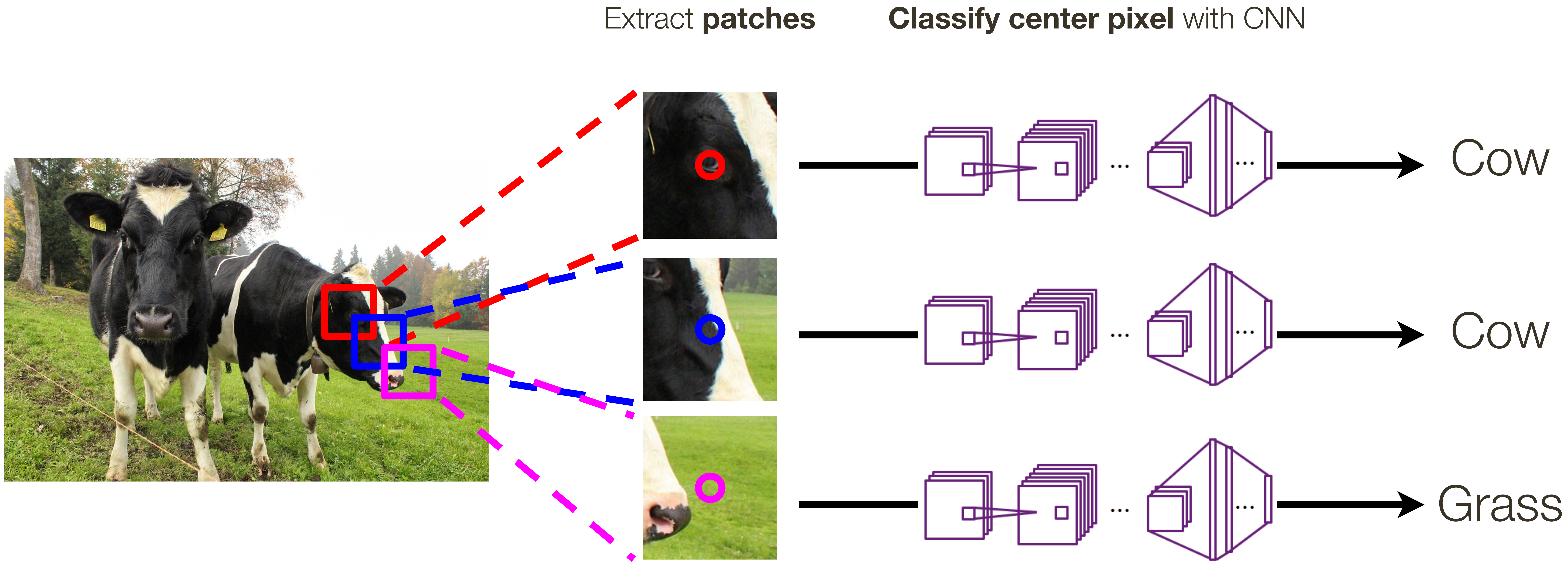
# Semantic **Segmentation:** Sliding Window

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



# Semantic **Segmentation:** Sliding Window

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

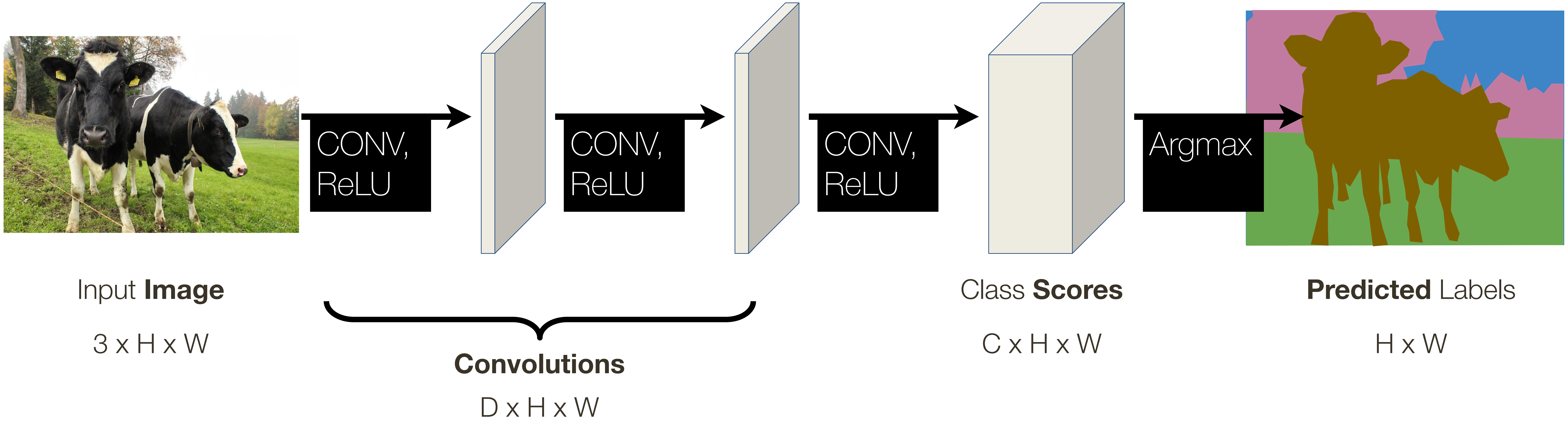



**Problem:** VERY inefficient, no reuse of computations for overlapping patches

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

# Semantic **Segmentation:** Fully Convolutional CNNs

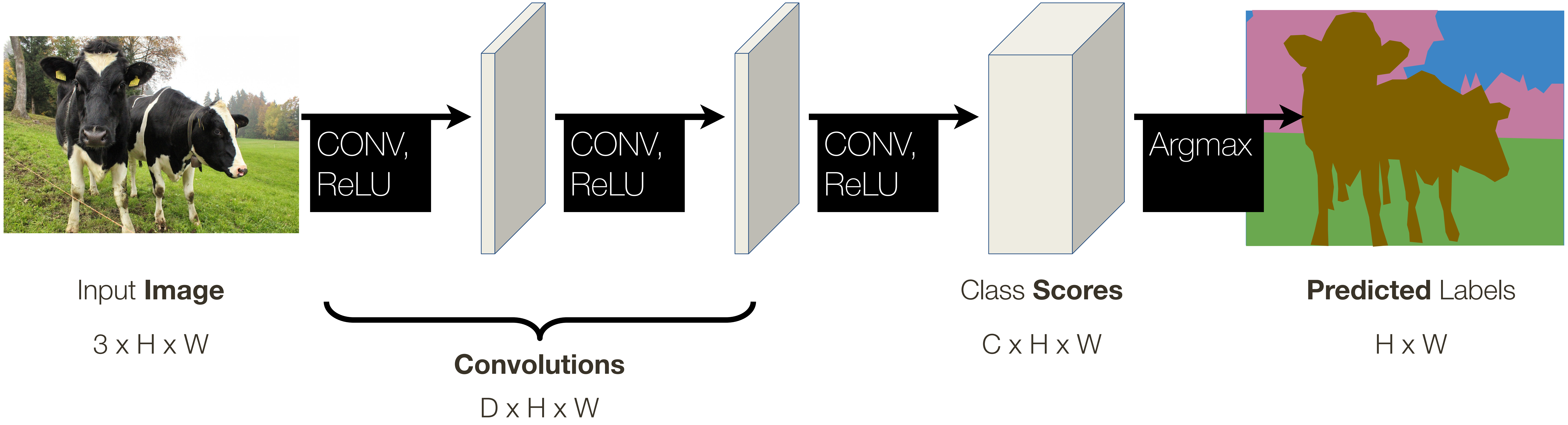
Design a network as a number of convolutional layers to make predictions for all pixels at once!




* 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

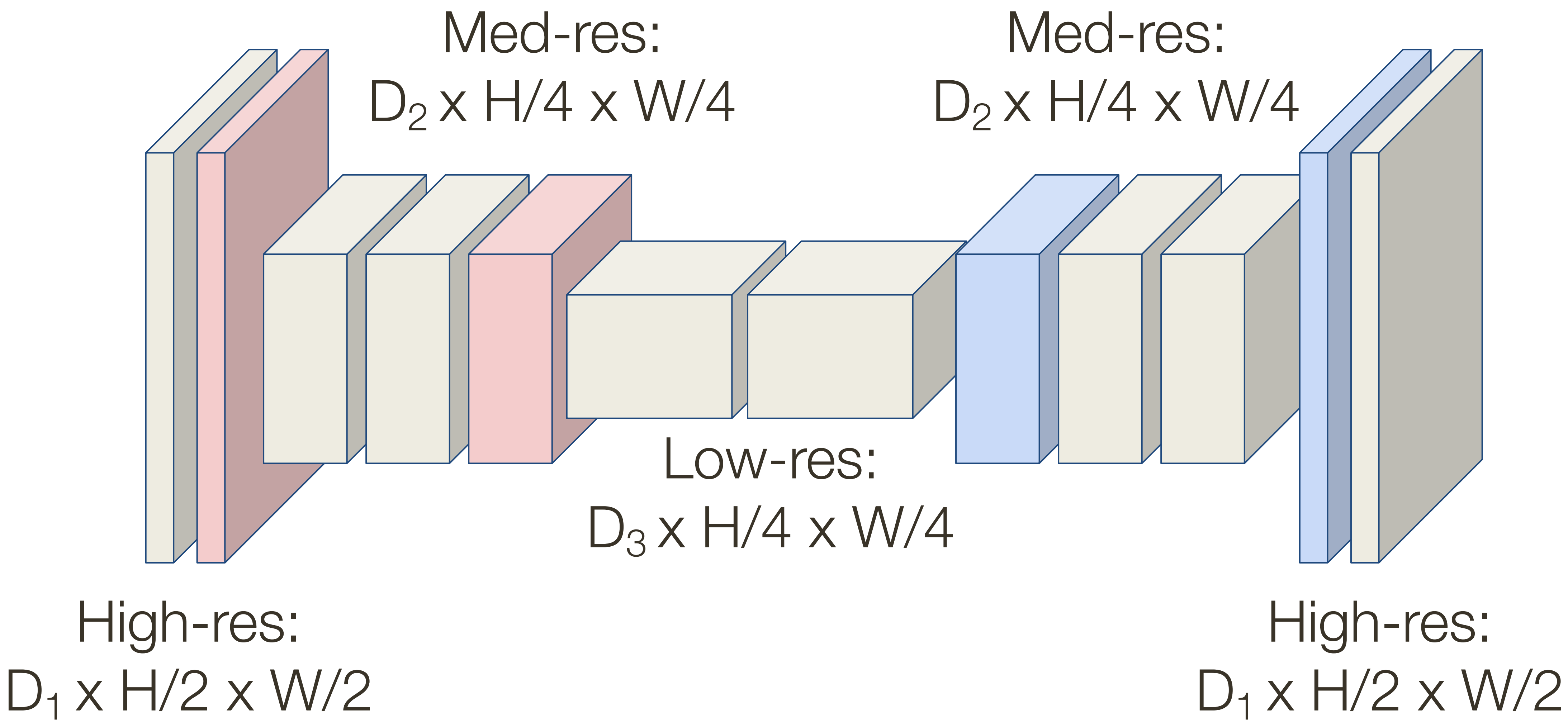
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# 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 ]

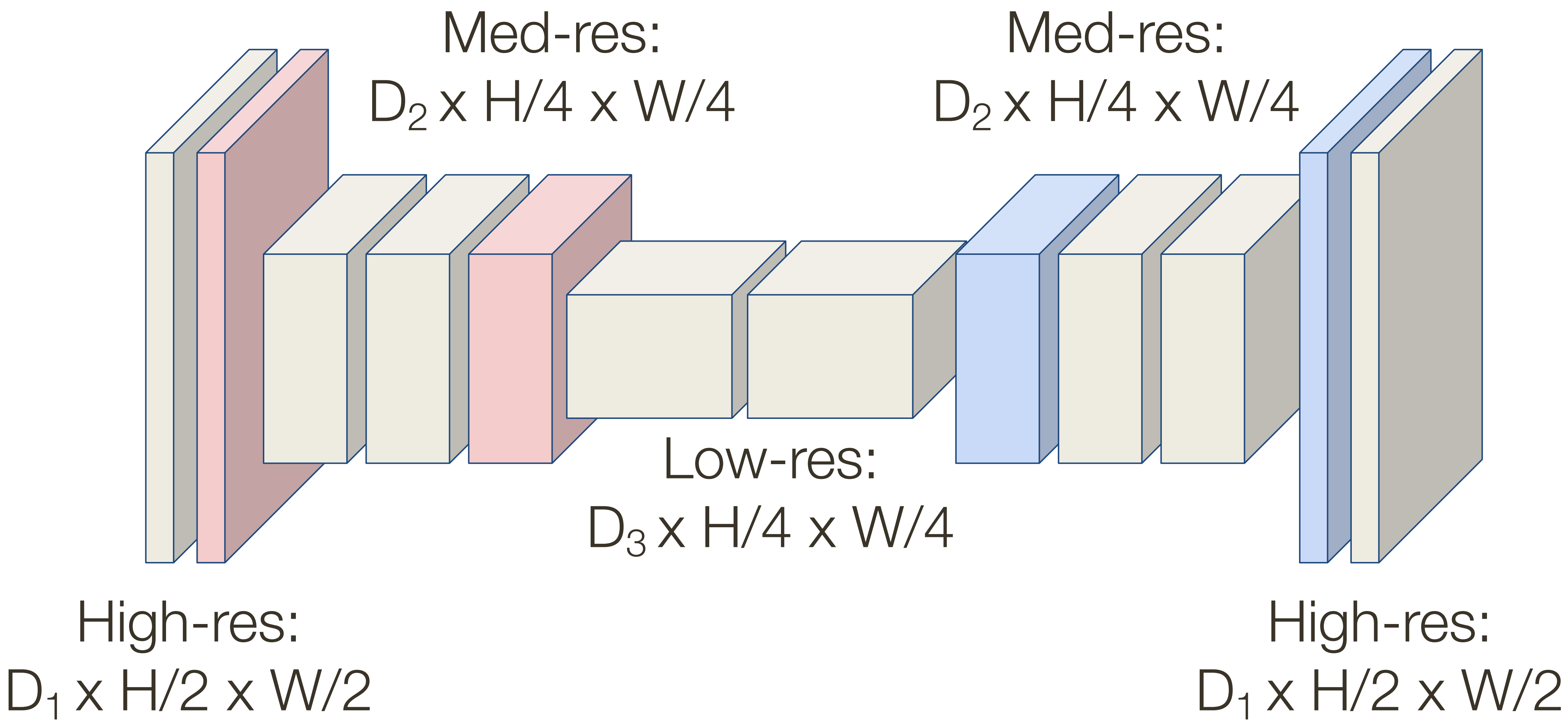
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# 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 ]

# In-network **Up Sampling** (a.k.a "Unpooling")

Nearest Neighbor

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

→

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

**Input:** 2 x 2

**Output:** 4 x 4

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

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

## Nearest Neighbor

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \longrightarrow \left[\begin{array}{cc|cc} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ \hline 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{array}\right]$$

**Input:** 2 x 2 **Output:** 4 x 4

## “Bed of Nails”

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \longrightarrow \left[\begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

**Input:** 2 x 2 **Output:** 4 x 4

# 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

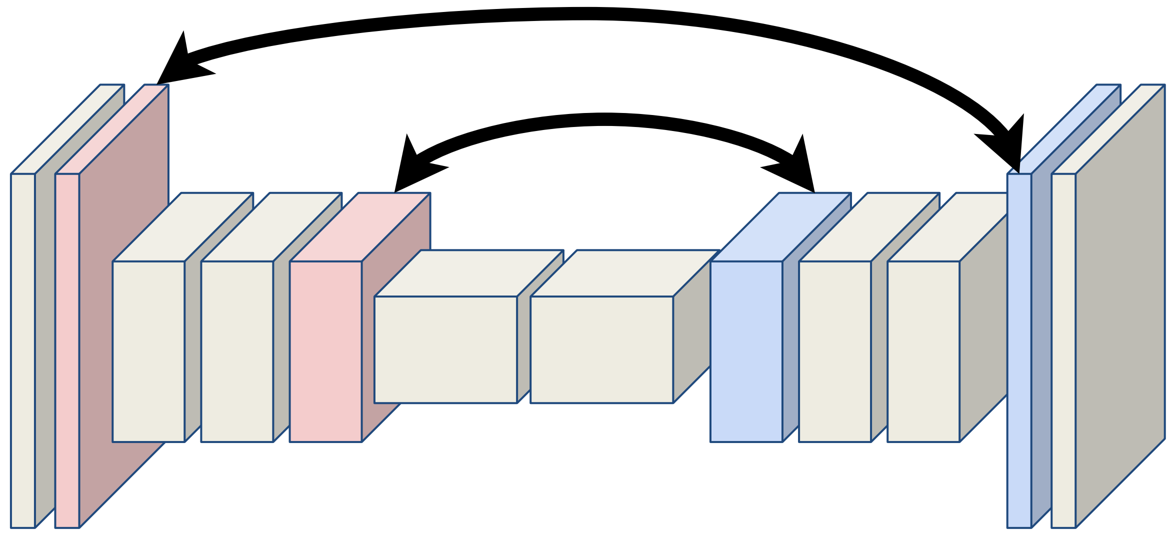
| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

**Input:** 2 x 2

→

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

**Output:** 4 x 4

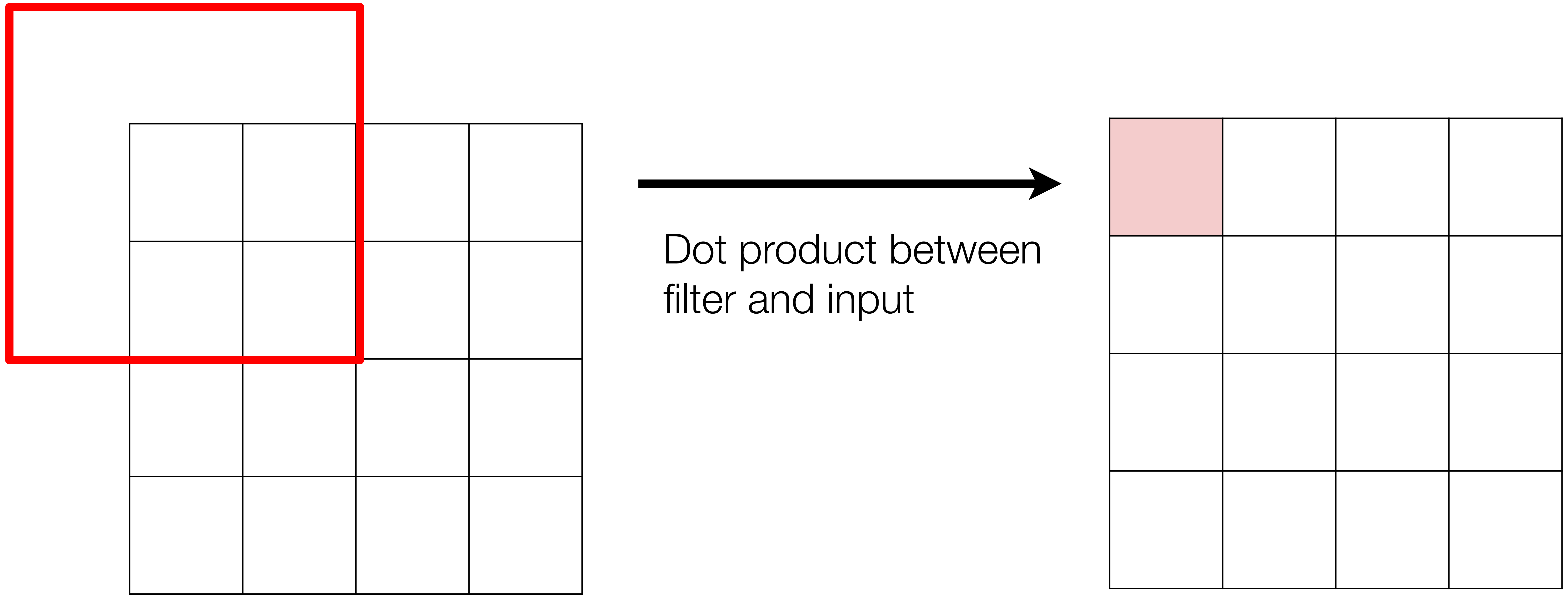


Corresponding pairs of downsampling and upsampling layers

* 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

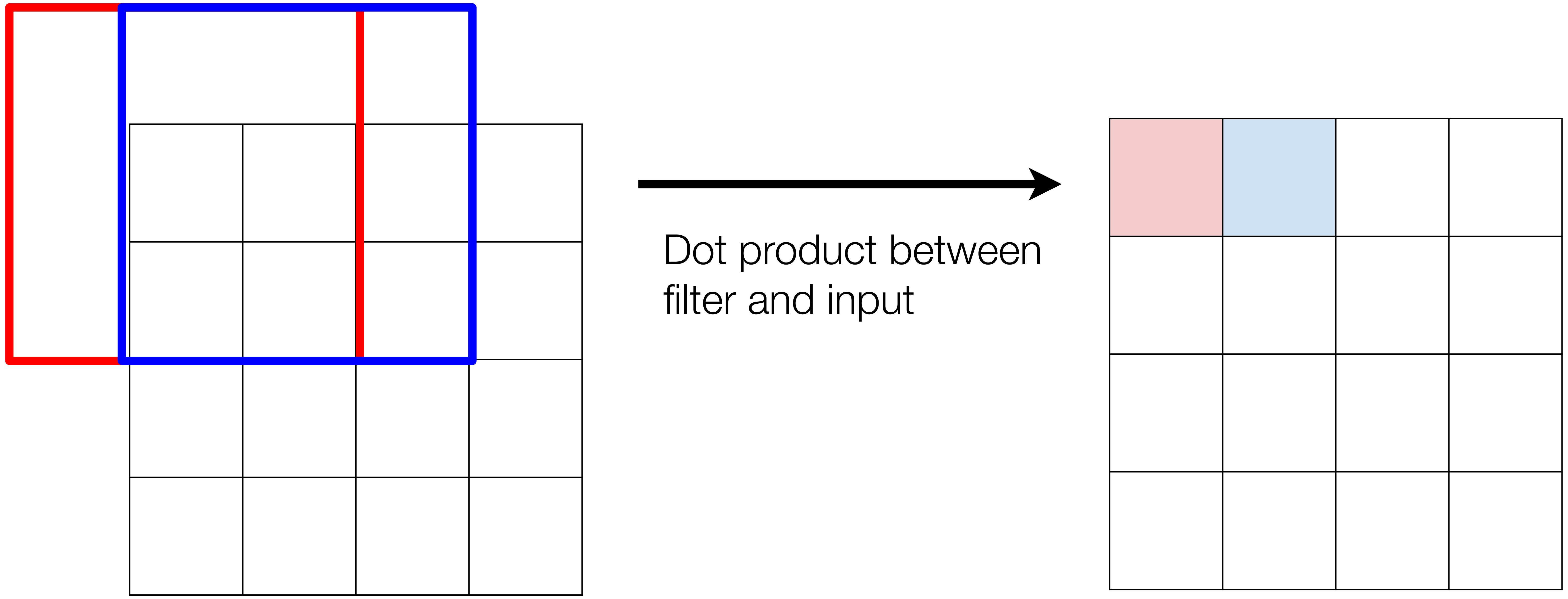



**Input:** 4 x 4

**Output:** 4 x 4

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

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



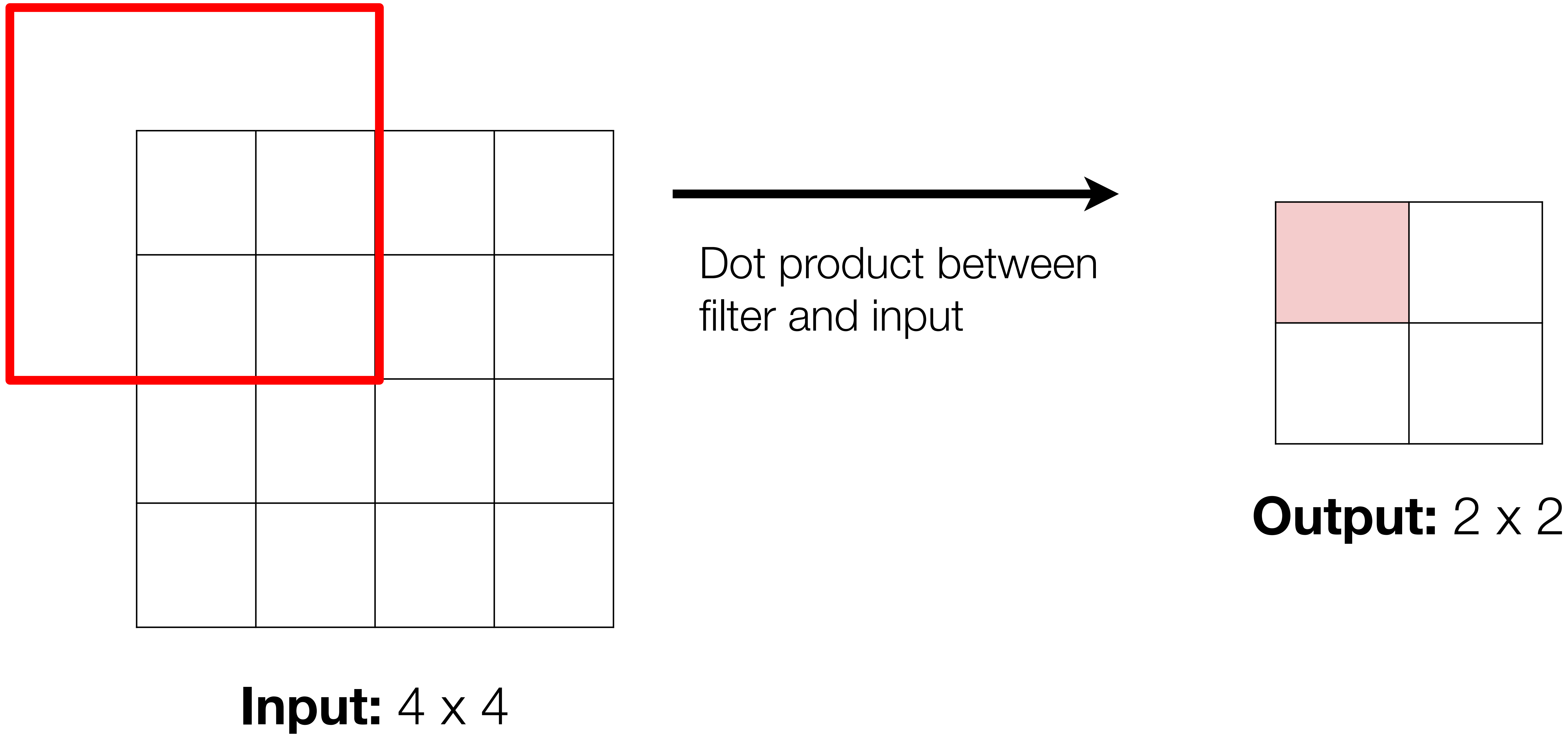

**Input:** 4 x 4

**Output:** 4 x 4

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

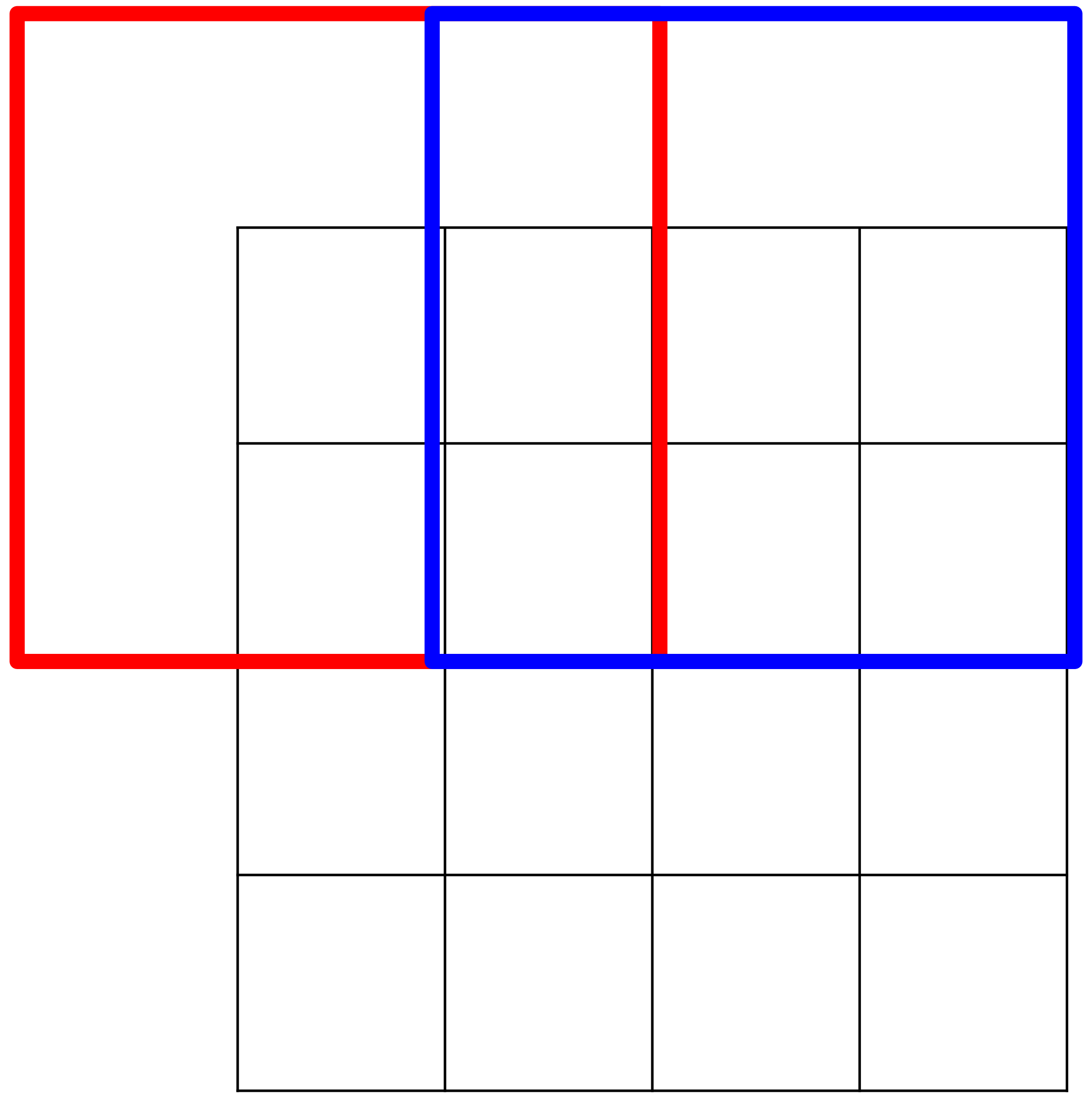
# In-network **Up Sampling:** Transpose Convolution

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



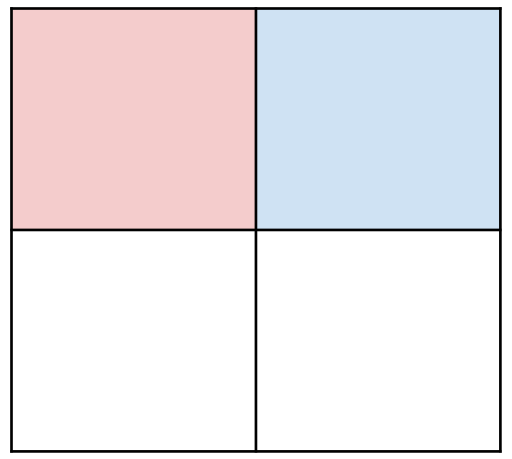
# In-network **Up Sampling:** Transpose Convolution

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



**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 in movement in input vs output

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

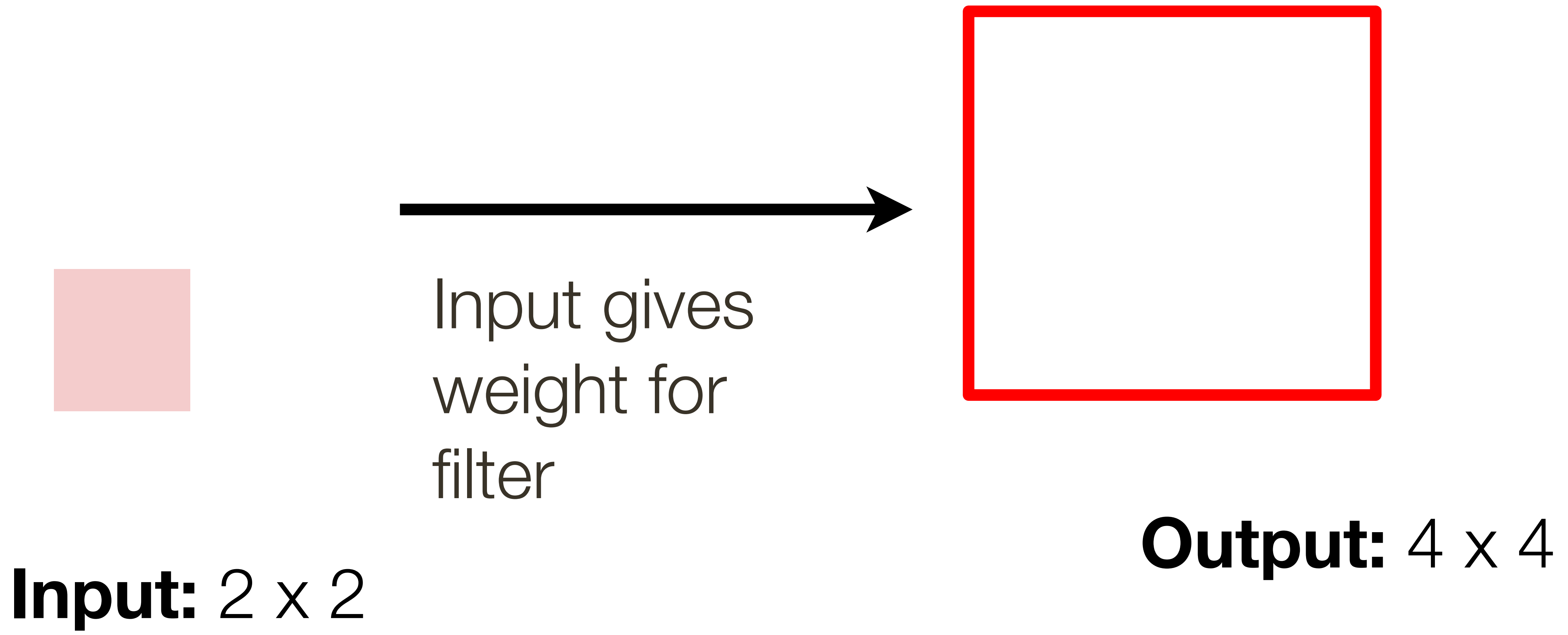
3 x 3 **transpose** convolution, stride 2 pad 1

**Input:** 2 x 2

**Output:** 4 x 4

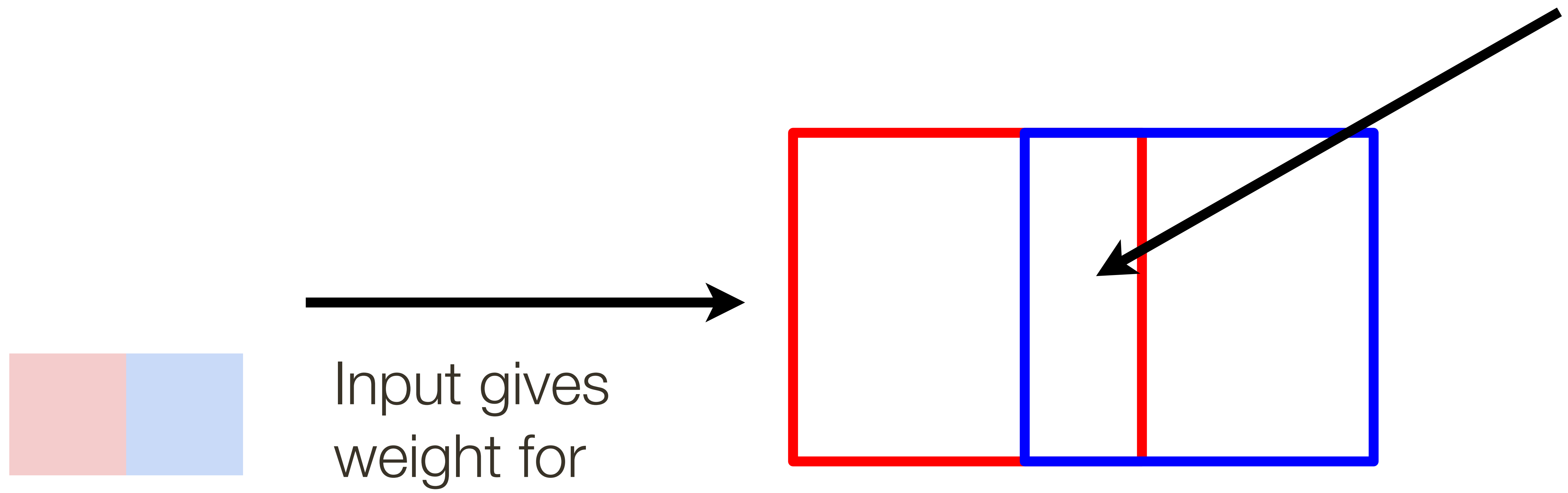
# In-network **Up Sampling:** Transpose Convolution

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



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

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



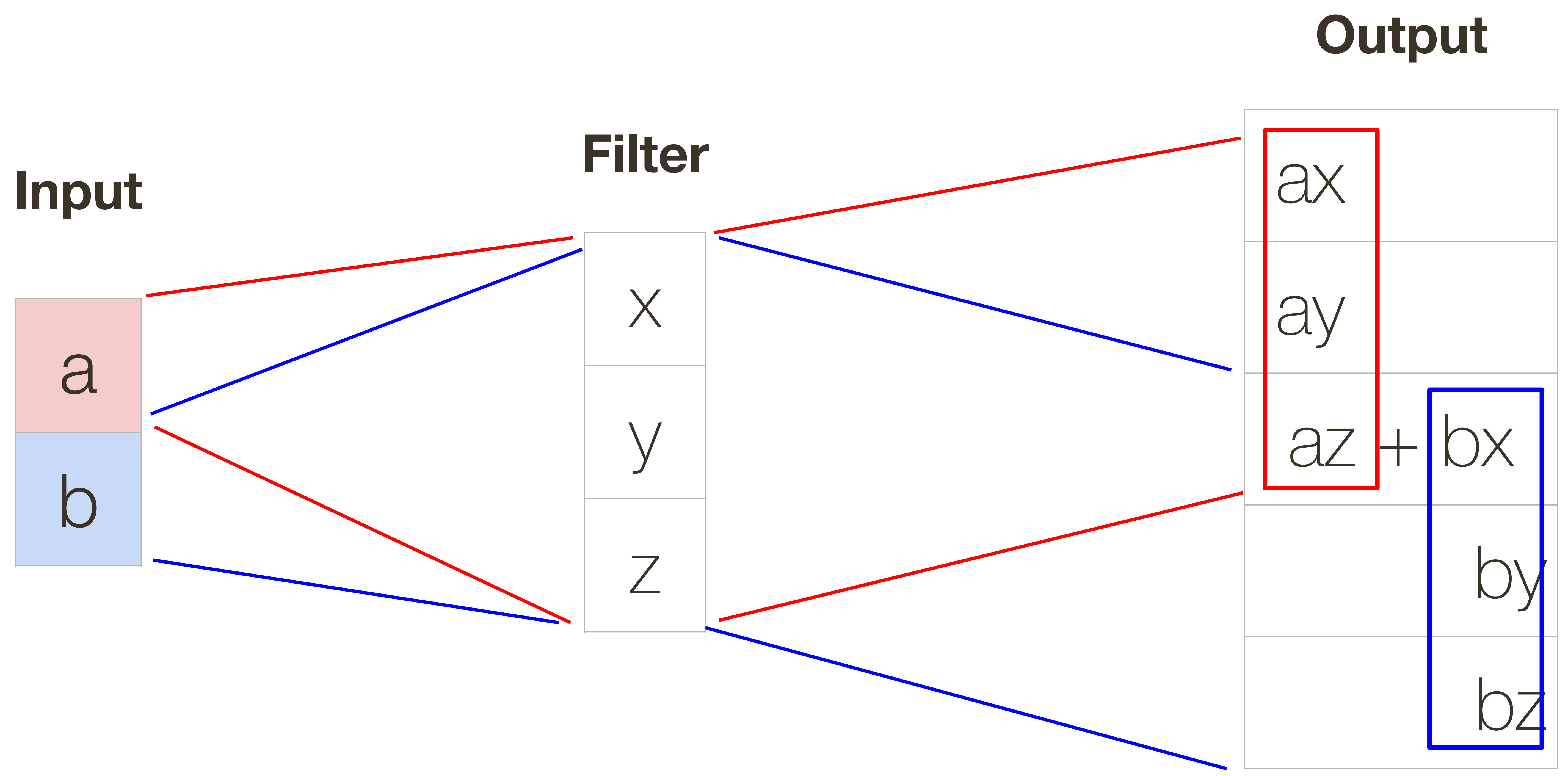

**Input:** 2 x 2

**Output:** 4 x 4

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

Stride gives ratio in movement in output vs input

# **Transpose Convolution**: 1-D Example

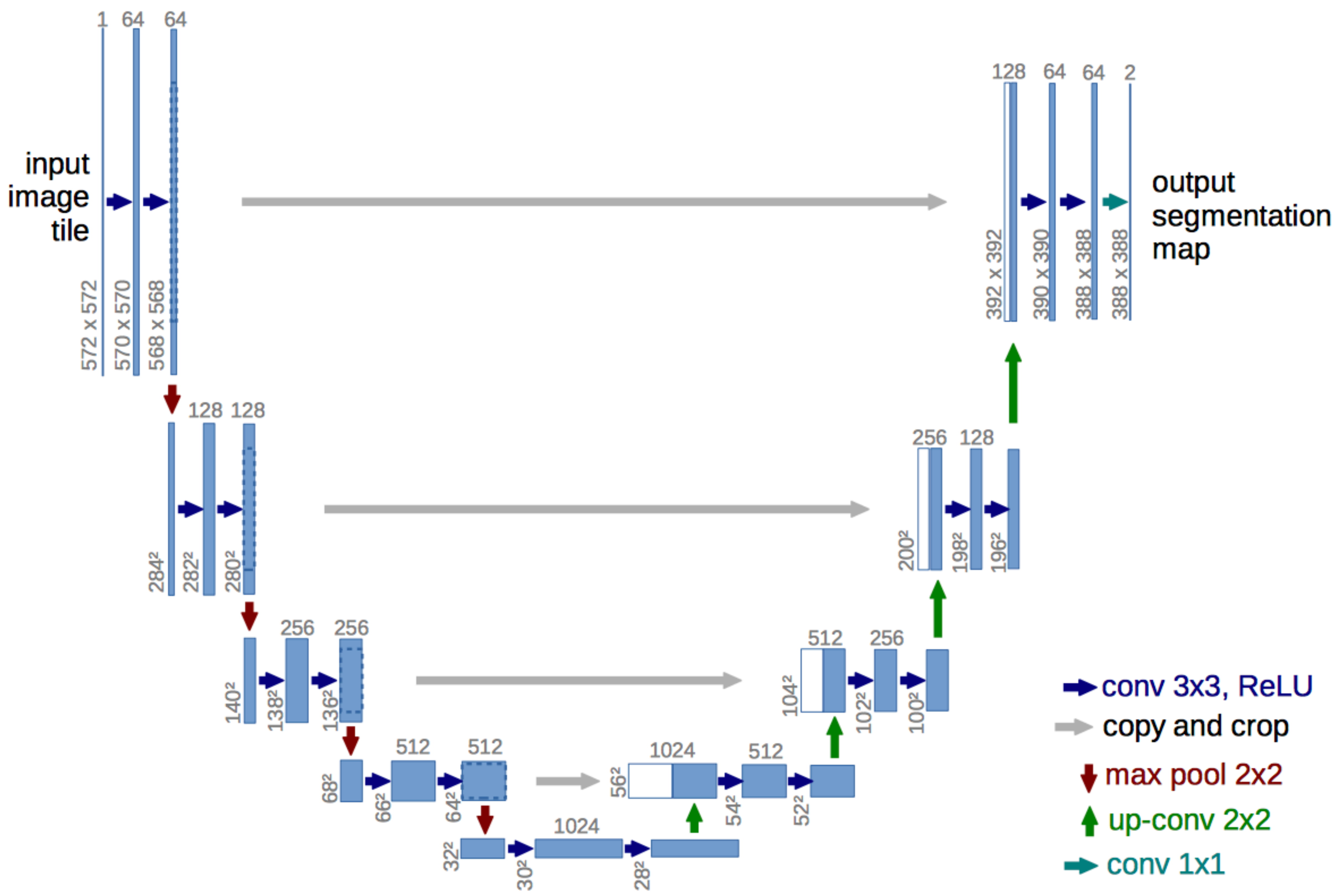



Output contains copies of the filter weighted multiplied by the input, summing at overlaps in the output

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

# **U-Net** Architecture

ResNet-like Fully convolutional CNN




[ Ronneberger et al, CVPR 2015 ]

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

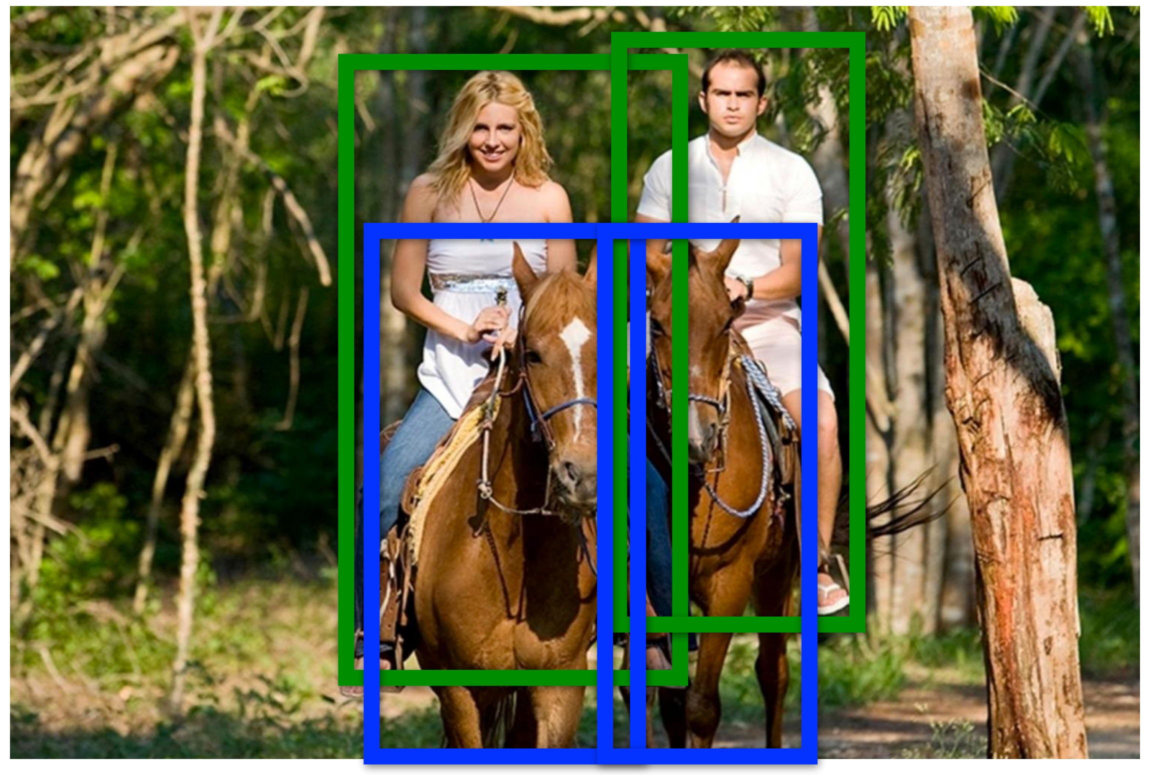
## Categorization

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

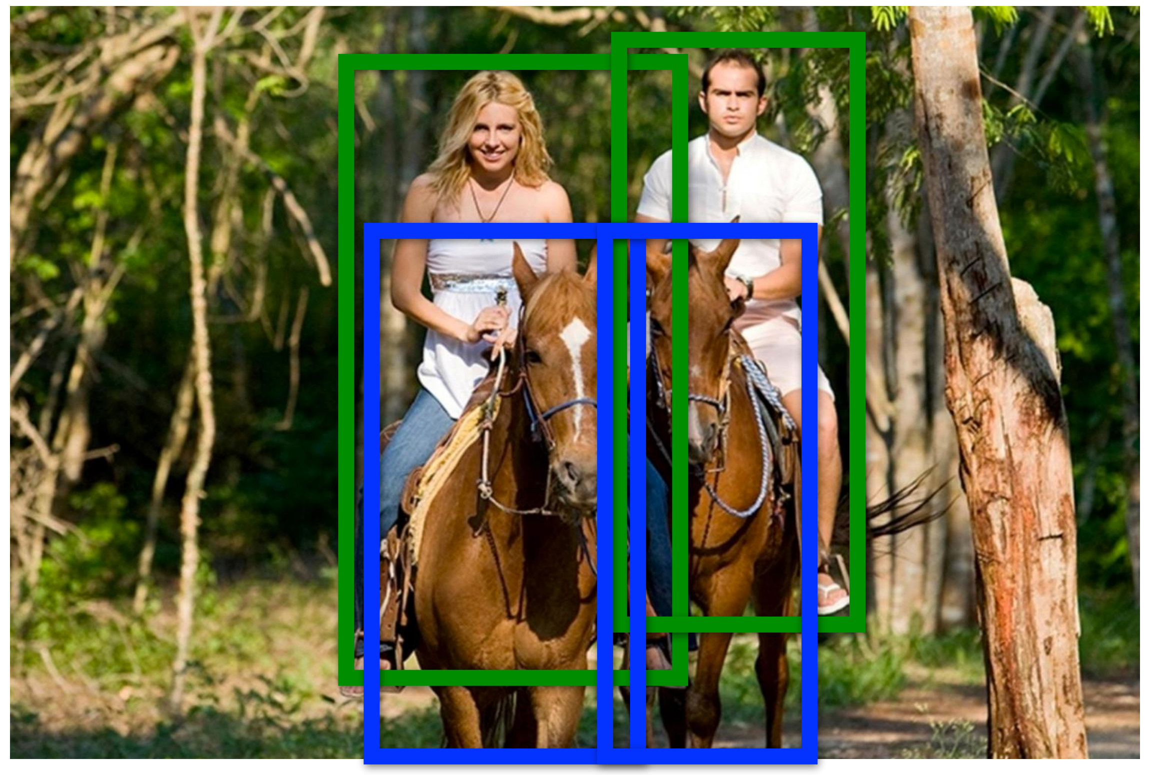
COCO Common Objects in Context

## Instance Segmentation

Horse1
Horse2
Person1
Person2

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

## Detection

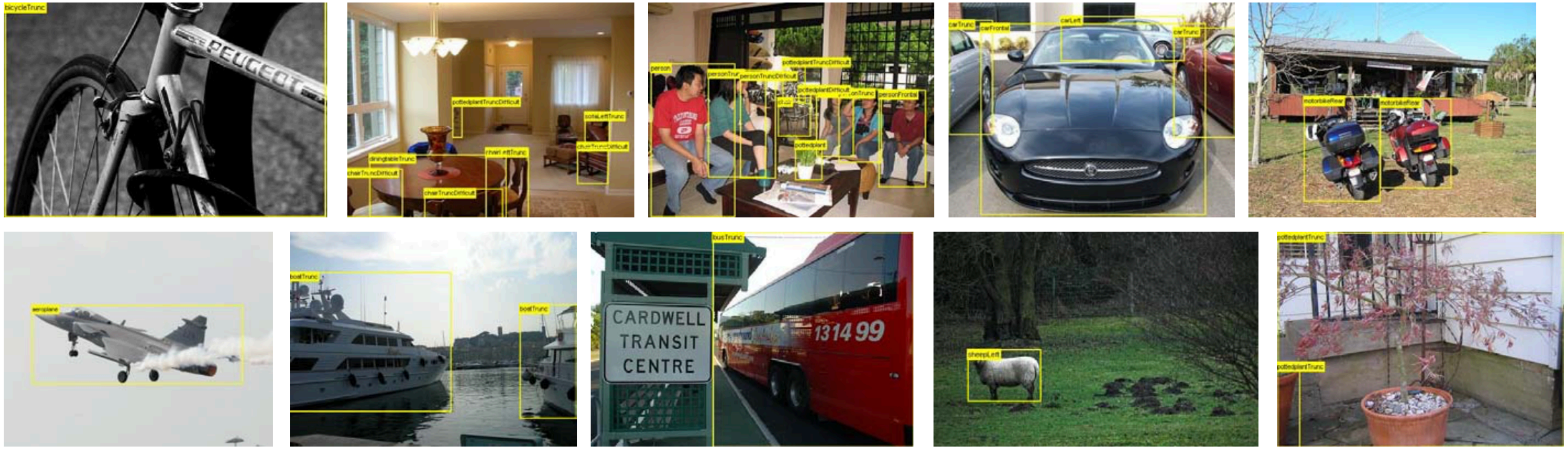


Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

# **Datasets:** Pascal VOC

20 classes: aeroplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, TV




Real images downloaded from flickr, not filtered for "quality"

* slide from Andrew Zisserman

# **Datasets:** Pascal VOC

20 classes: aeroplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, TV

| | Training | Testing |
|---|---|---|
| **Images** | 10,103 | 9,637 |
| **Objects** | 23,374 | 22,992 |

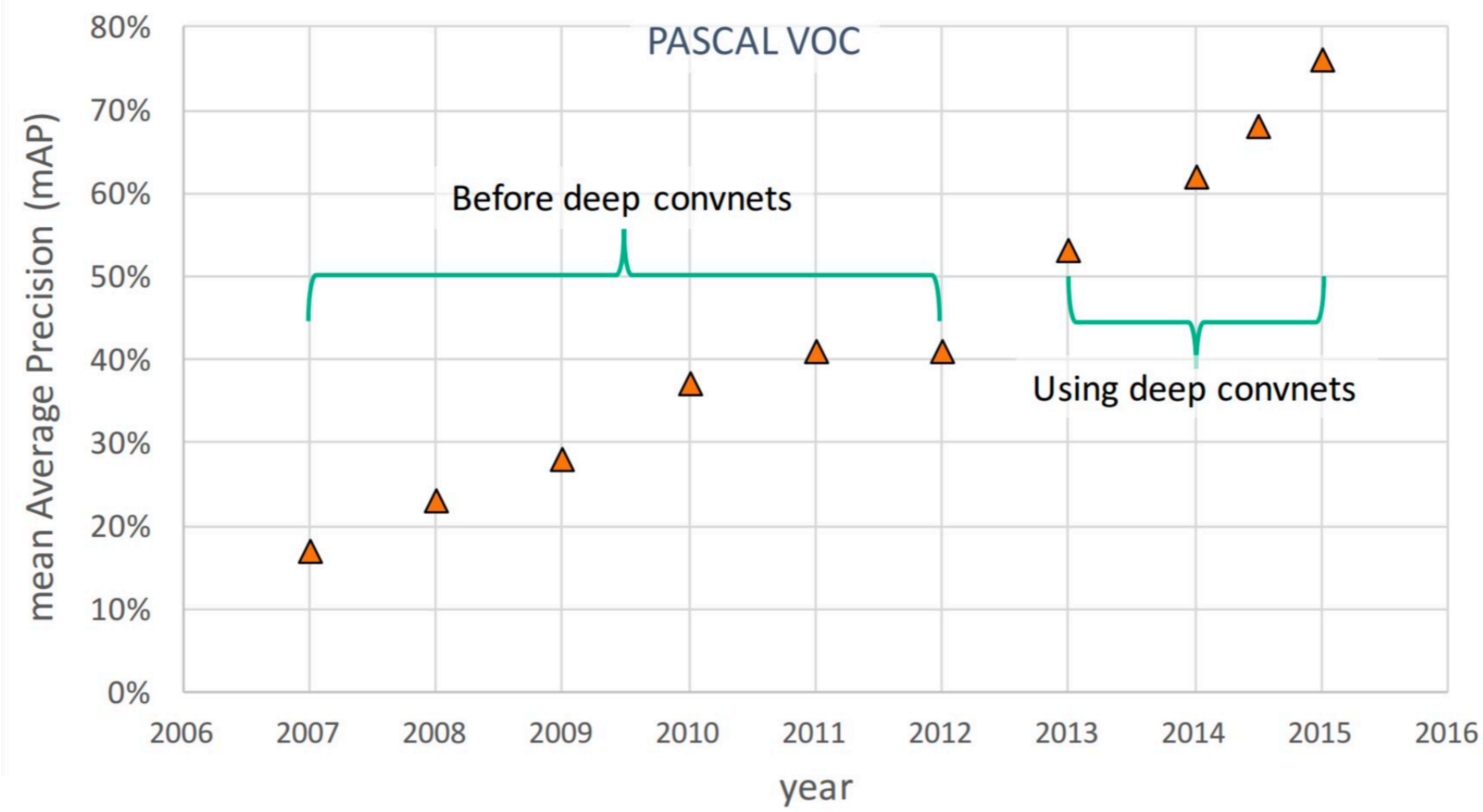
Real images downloaded from flickr, not filtered for “quality”

* slide from Andrew Zisserman

# Datasets: COCO

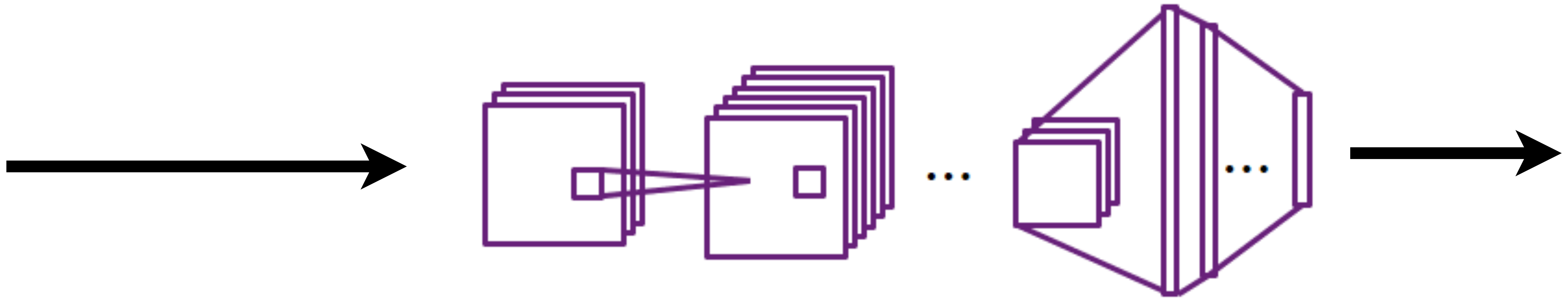
- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

# Object **Detection**



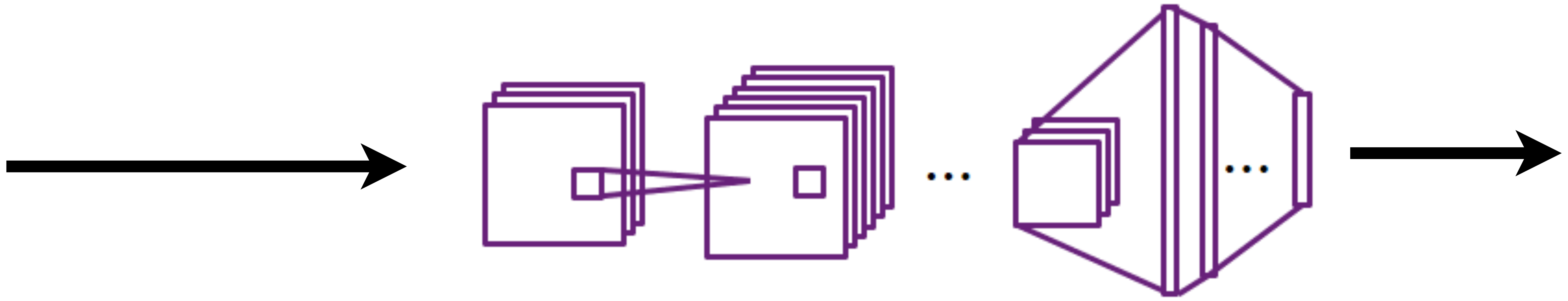

* plot from Ross Girshick, 2015

# Object **Detection** as Regression Problem

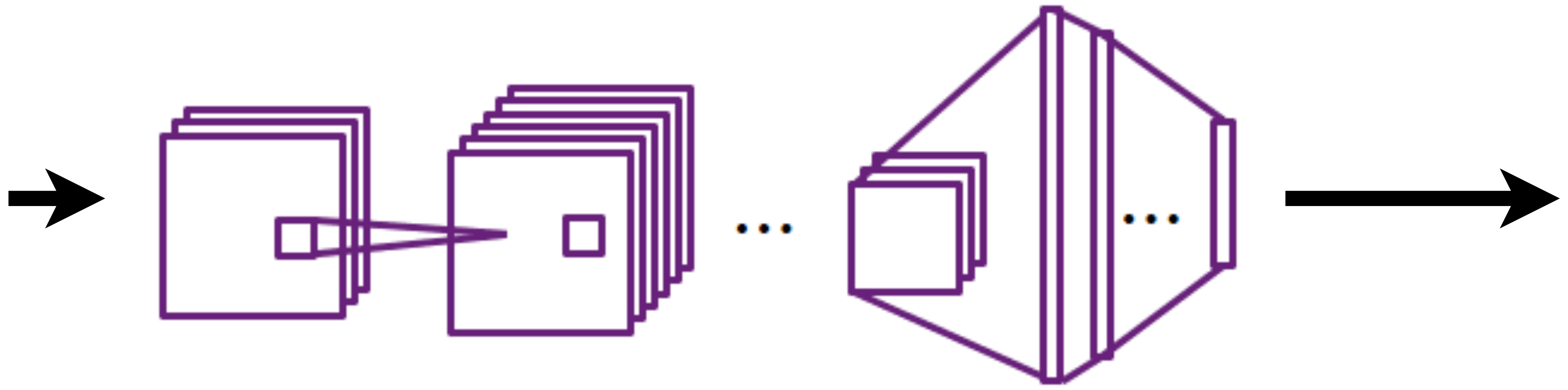



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# Object **Detection** as Regression Problem



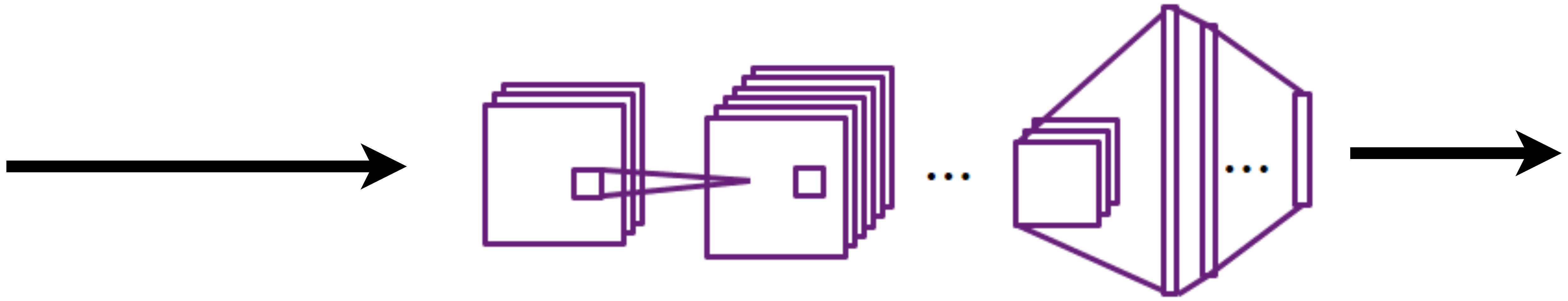
CAT (x, y, w ,h)



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

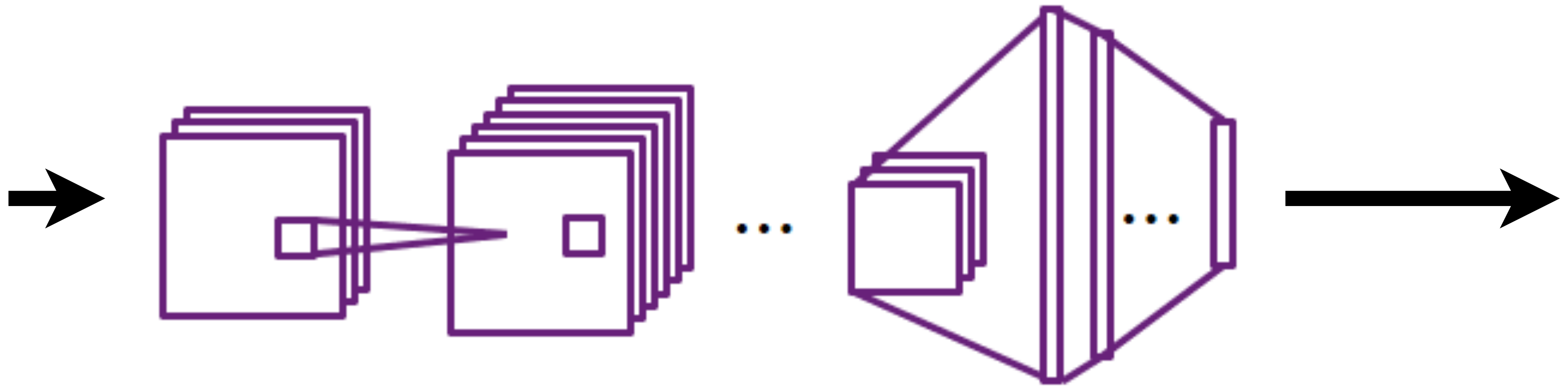
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

# Object **Detection** as Regression Problem



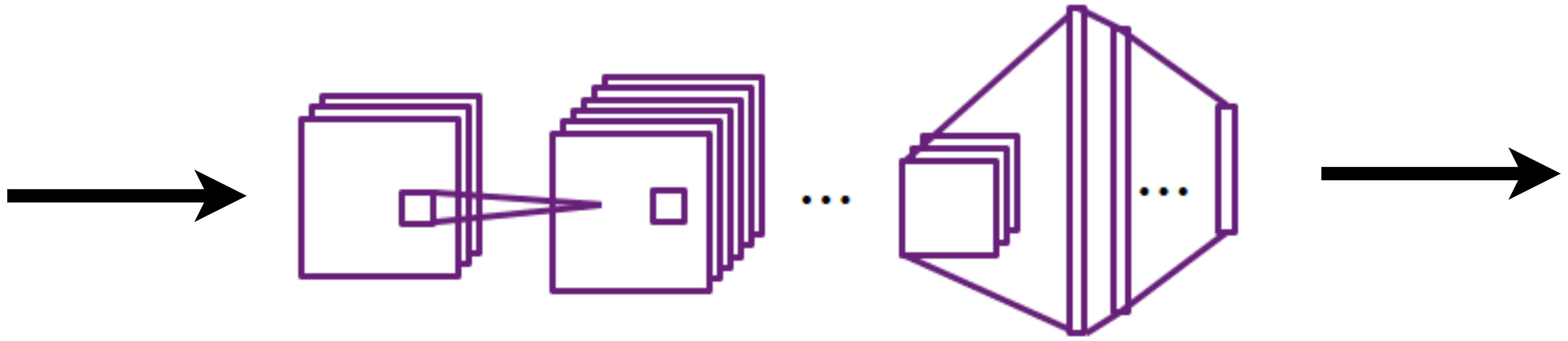
CAT (x, y, w ,h)

**Problem:** each image needs a different number of outputs



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

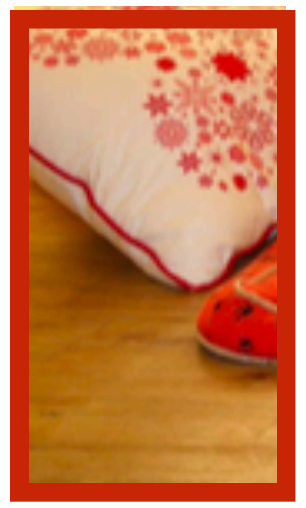
# Object **Detection** as Classification Problem

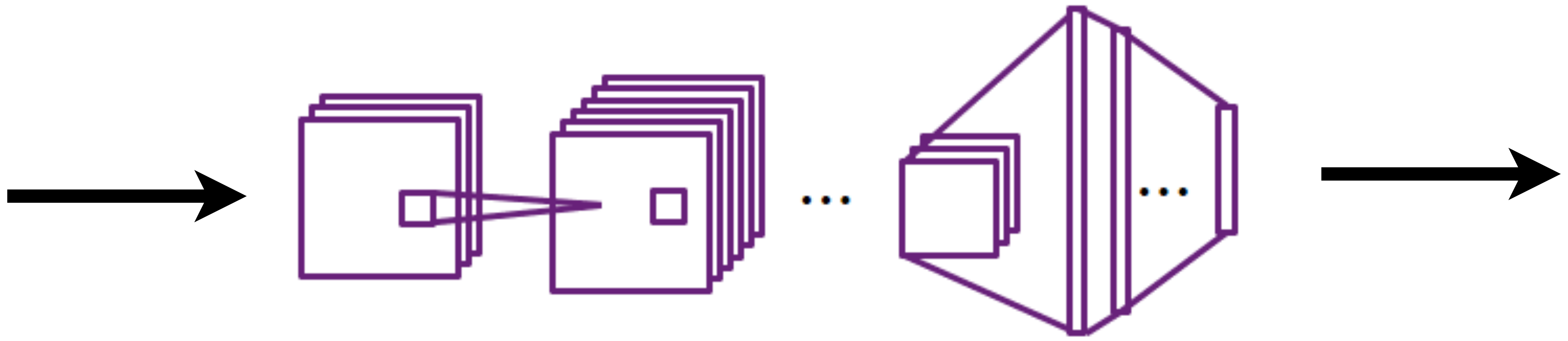


| **Category** | **Prediction** |
|---|---|
| Dog | No |
| Cat | No |
| Couch | No |
| Flowers | No |
| Background | **Yes** |
| … | … |

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

# Object **Detection** as Classification Problem

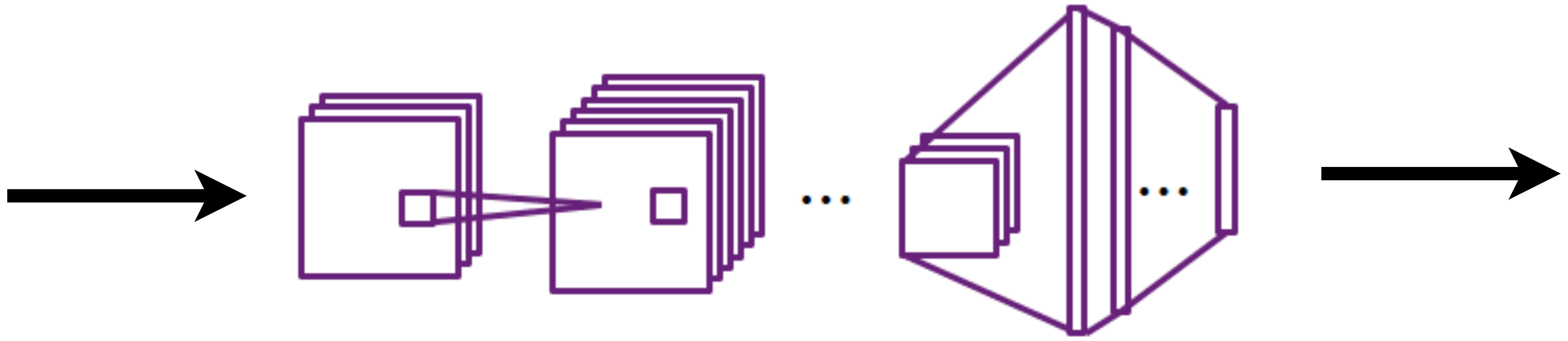




| **Category** | **Prediction** |
|---|---|
| Dog | No |
| Cat | No |
| Couch | No |
| Flowers | No |
| Background | **Yes** |
| … | … |

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

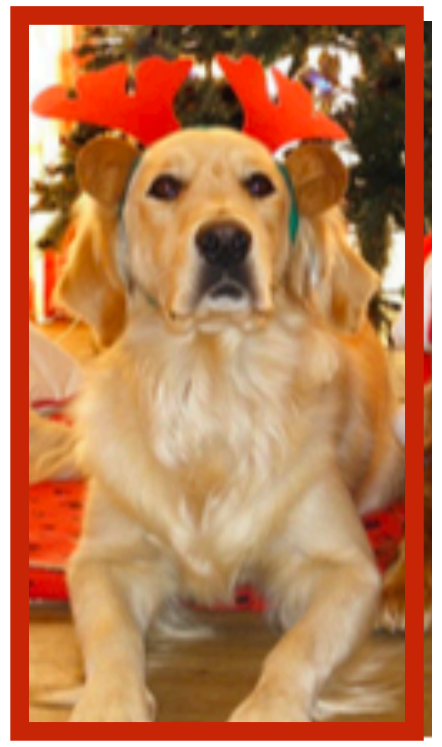
# Object **Detection** as Classification Problem

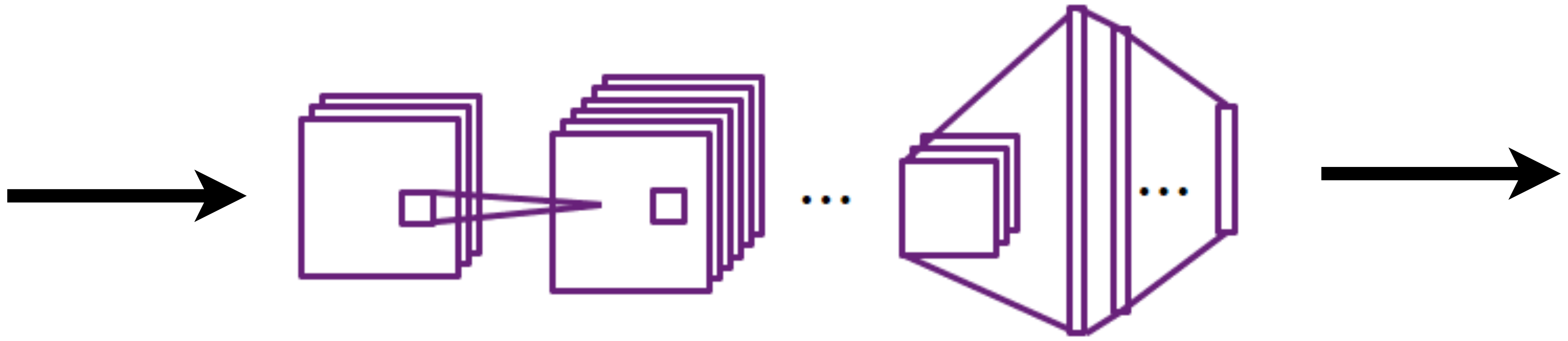


| **Category** | **Prediction** |
|---|---|
| Dog | **Yes** |
| Cat | No |
| Couch | No |
| Flowers | No |
| Background | No |
| … | … |

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

# Object **Detection** as Classification Problem





| **Category** | **Prediction** |
|---|---|
| Dog | **Yes** |
| Cat | No |
| Couch | No |
| Flowers | No |
| Background | No |
| … | … |

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

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

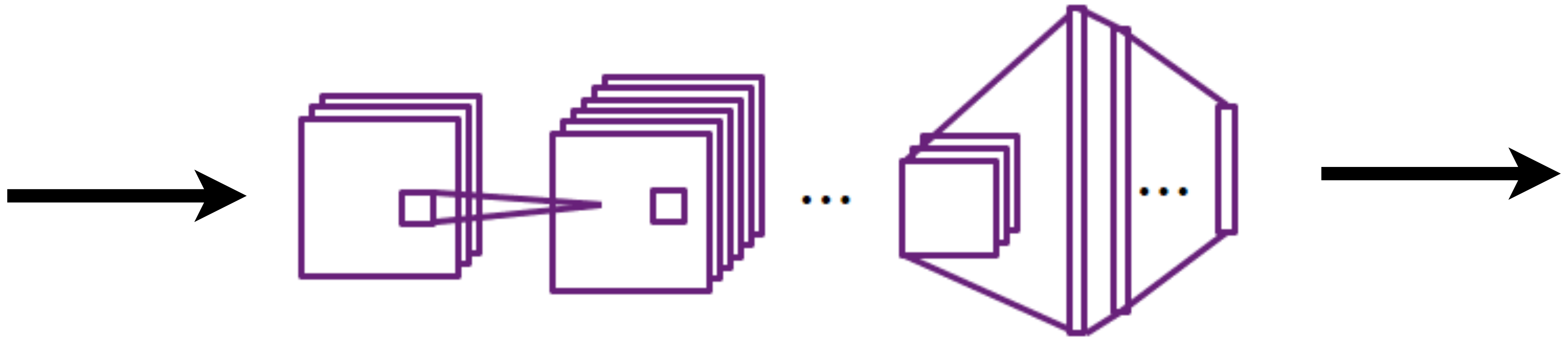
# Object **Detection** as Classification Problem


| **Category** | **Prediction** |
|---|---|
| Dog | No |
| Cat | **Yes** |
| Couch | No |
| Flowers | No |
| Background | No |
| … | … |

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

# Object **Detection** as Classification Problem

**Problem:** Need to apply CNN to **many** patches in each image



| **Category** | **Prediction** |
|---|---|
| Dog | No |
| Cat | **Yes** |
| Couch | No |
| Flowers | No |
| Background | No |
| … | … |

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

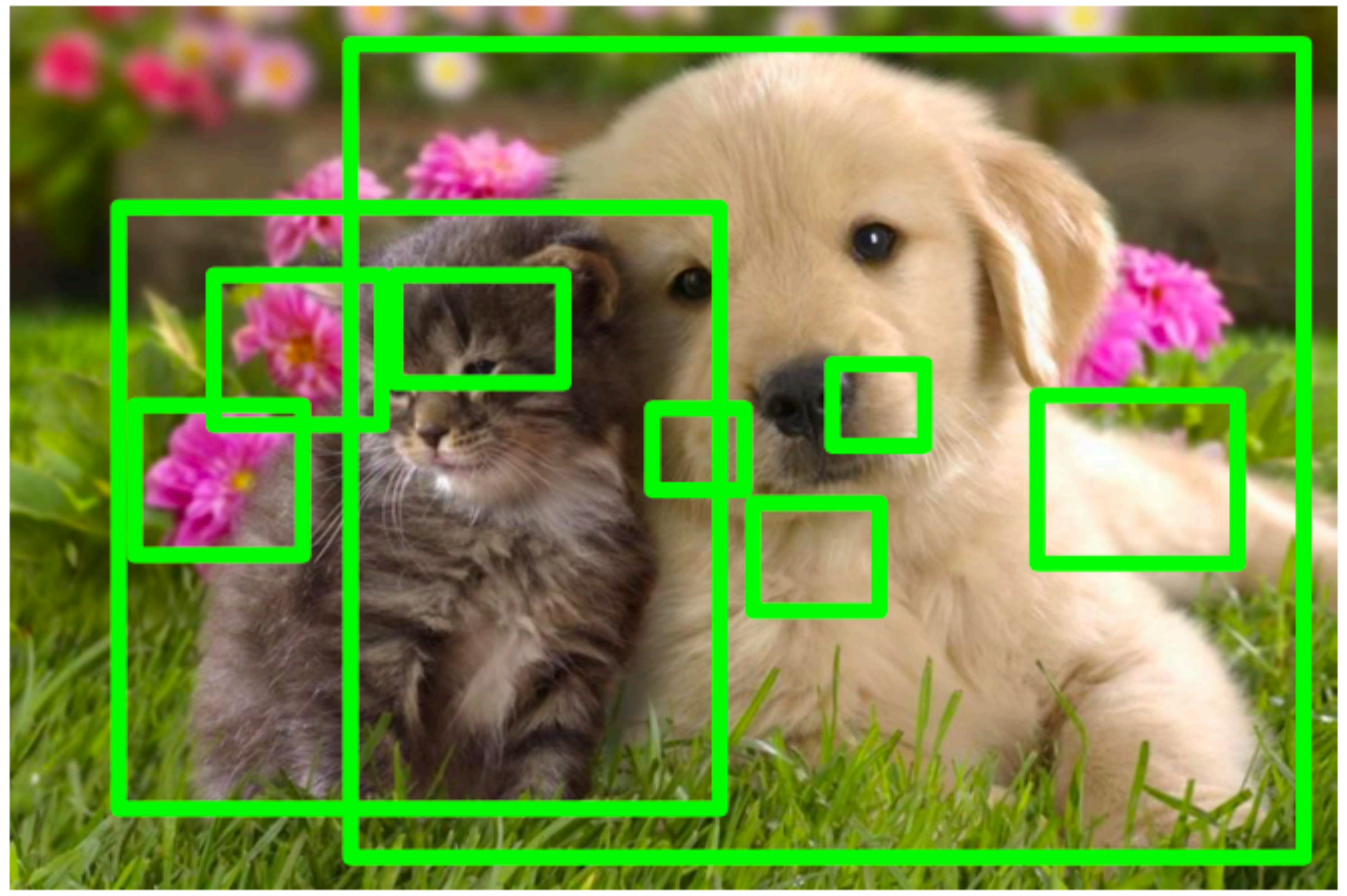
# **Region** Proposals (older idea in vision)

[ Alexe et al, TPAMI 2012 ]
[ Uijkings et al, IJCV 2013 ]
[ Cheng et al, CVPR 2014 ]
[ Zitnick and Dollar, ECCV 2014 ]

Find image **regions that are likely contain objects** (any object at all)

- typically works by looking at histogram distributions, region aspect ratio, closed contours, coherent color

Relatively **fast to run** (Selective Search gives 1000 region proposals in a few seconds on a CPU)



**Goal:** Get "true" object regions to be in as few top K proposals as possible

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

# R-CNN

[ Girshick et al, CVPR 2014 ]

Input **Image**

* image from Ross Girshick

# R-CNN

[ Girshick et al, CVPR 2014 ]

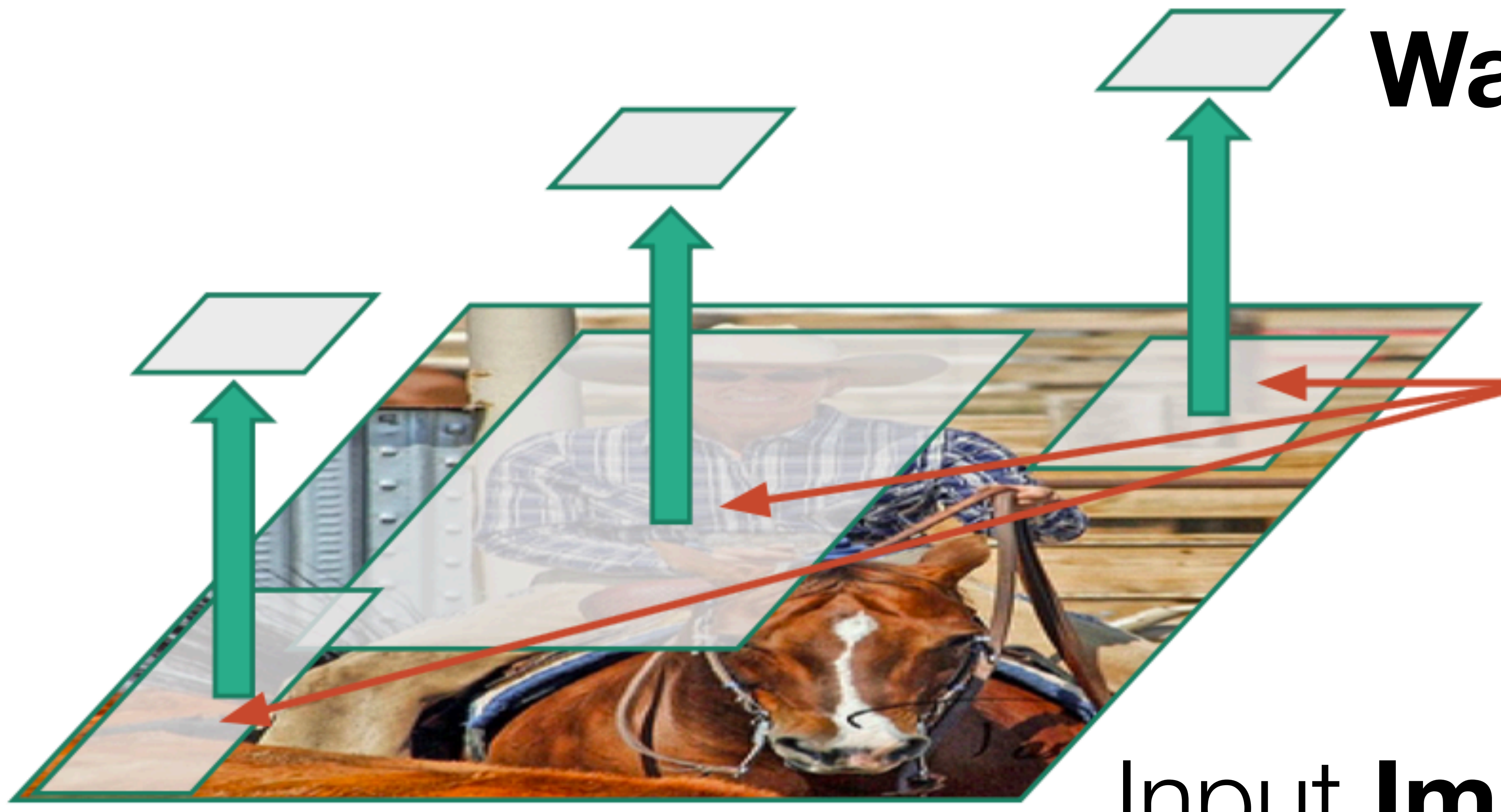
**Regions of Interest** from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

# R-CNN

[ Girshick et al, CVPR 2014 ]



**Warped** image regions

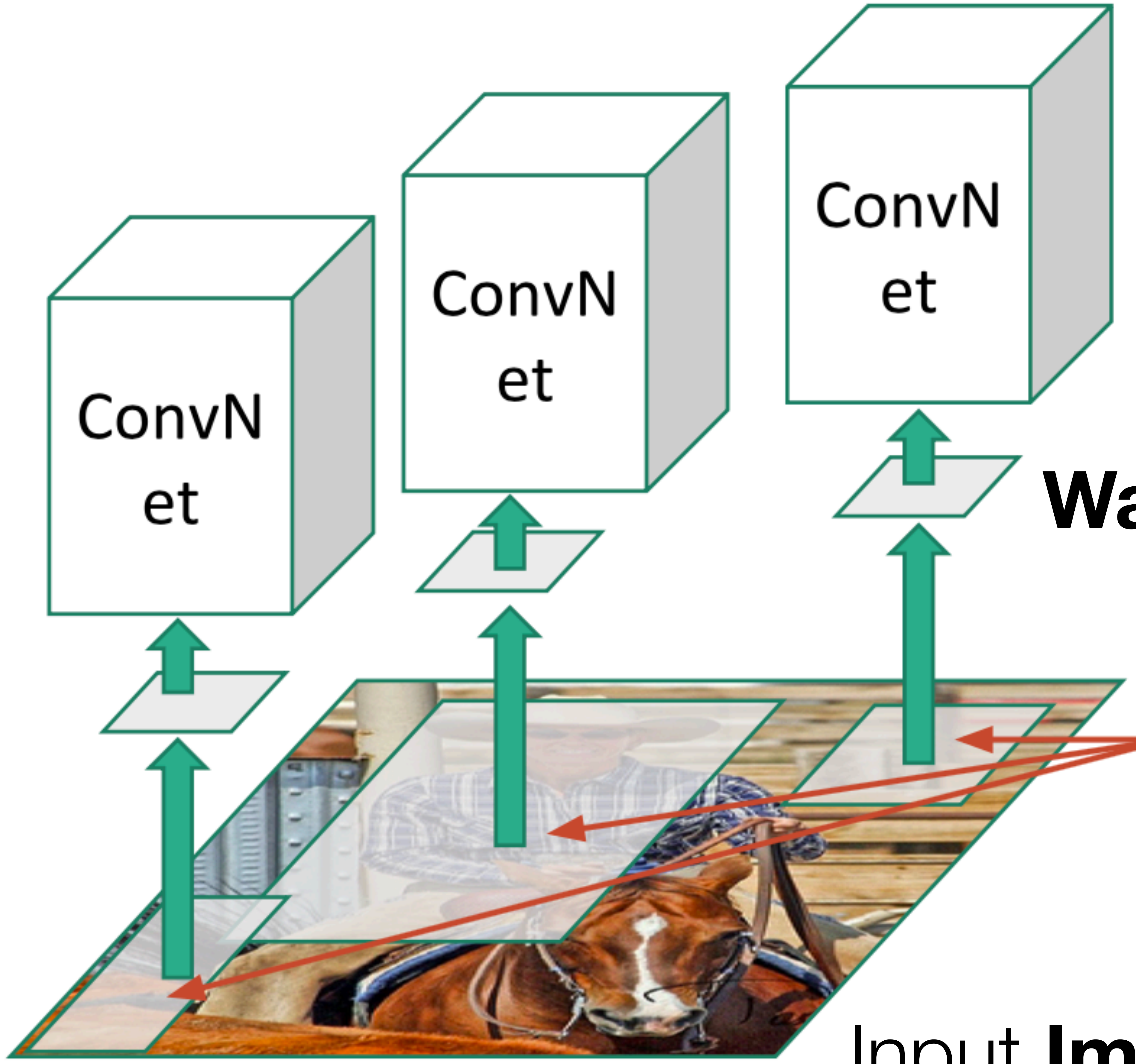
**Regions of Interest** from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

# R-CNN

[ Girshick et al, CVPR 2014 ]




Forward each region through a **CNN**

**Warped** image regions

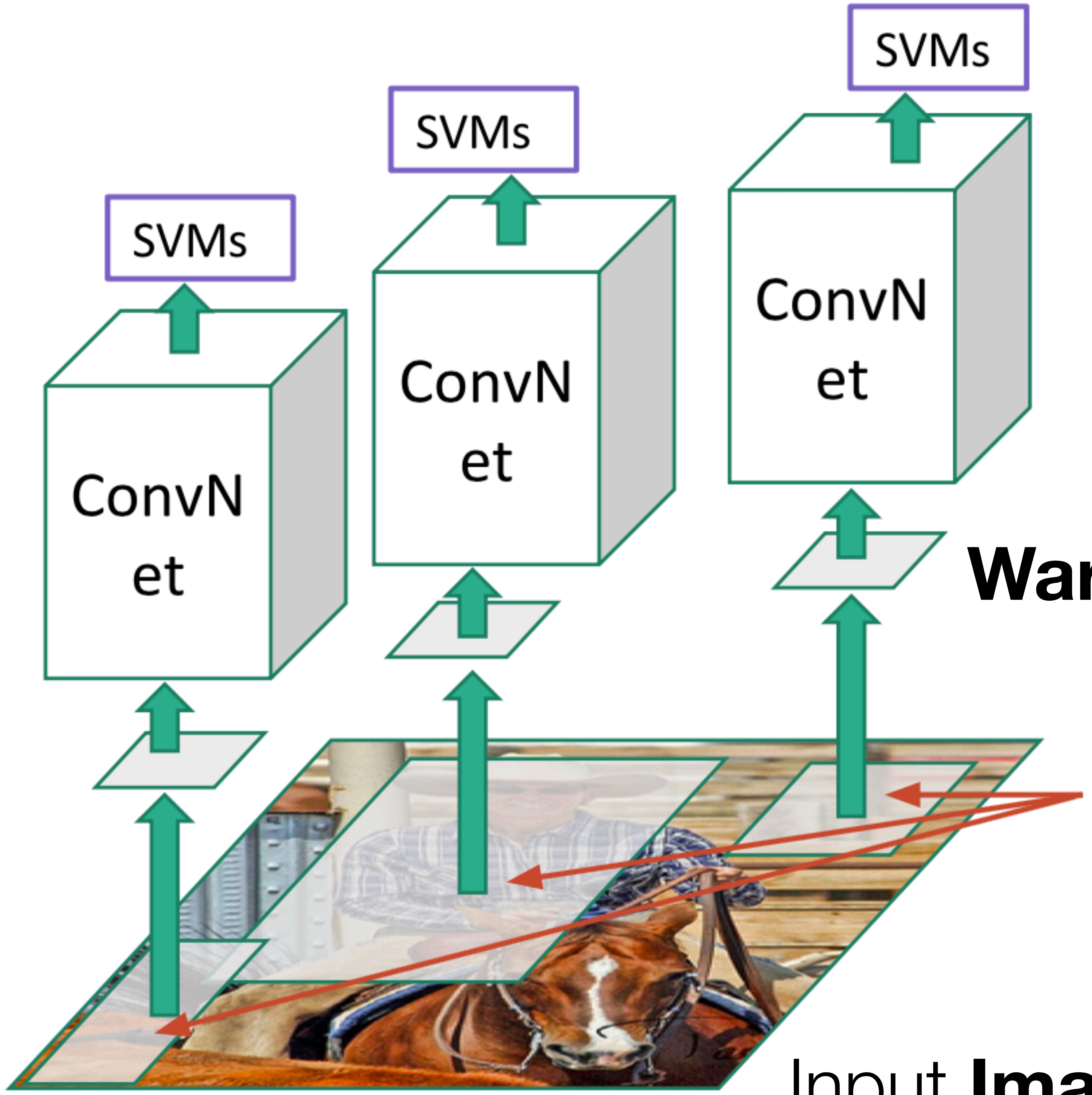
**Regions of Interest** from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

# R-CNN

[ Girshick et al, CVPR 2014 ]




**Classify** regions with SVM

Forward each region through a **CNN**

**Warped** image regions

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

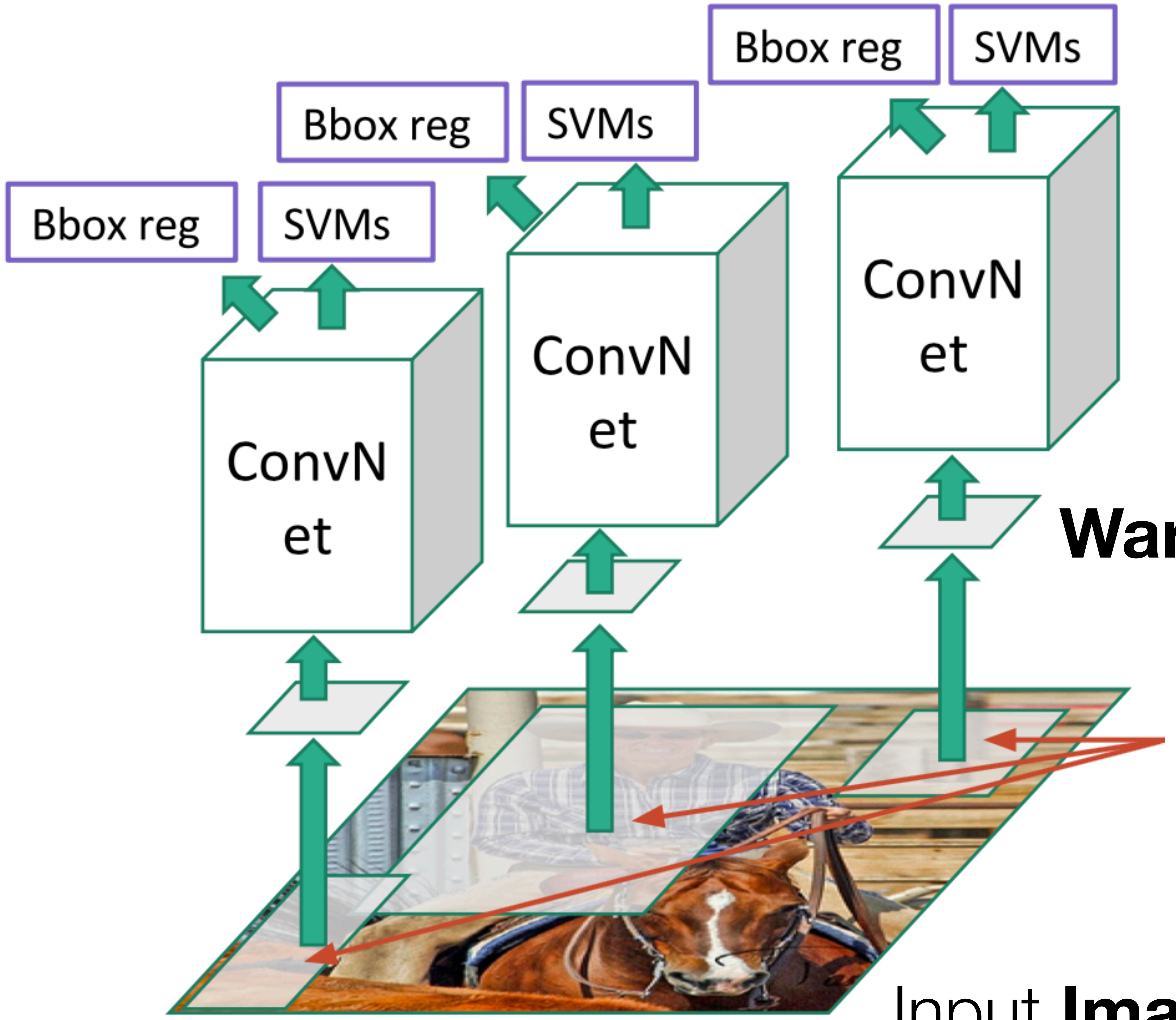
Input **Image**

* image from Ross Girshick

# R-CNN

[ Girshick et al, CVPR 2014 ]

**Linear Regression** for bounding box offsets




**Classify** regions with SVM

Forward each region through a **CNN**

**Warped** image regions

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

Input **Image**

* image from Ross Girshick

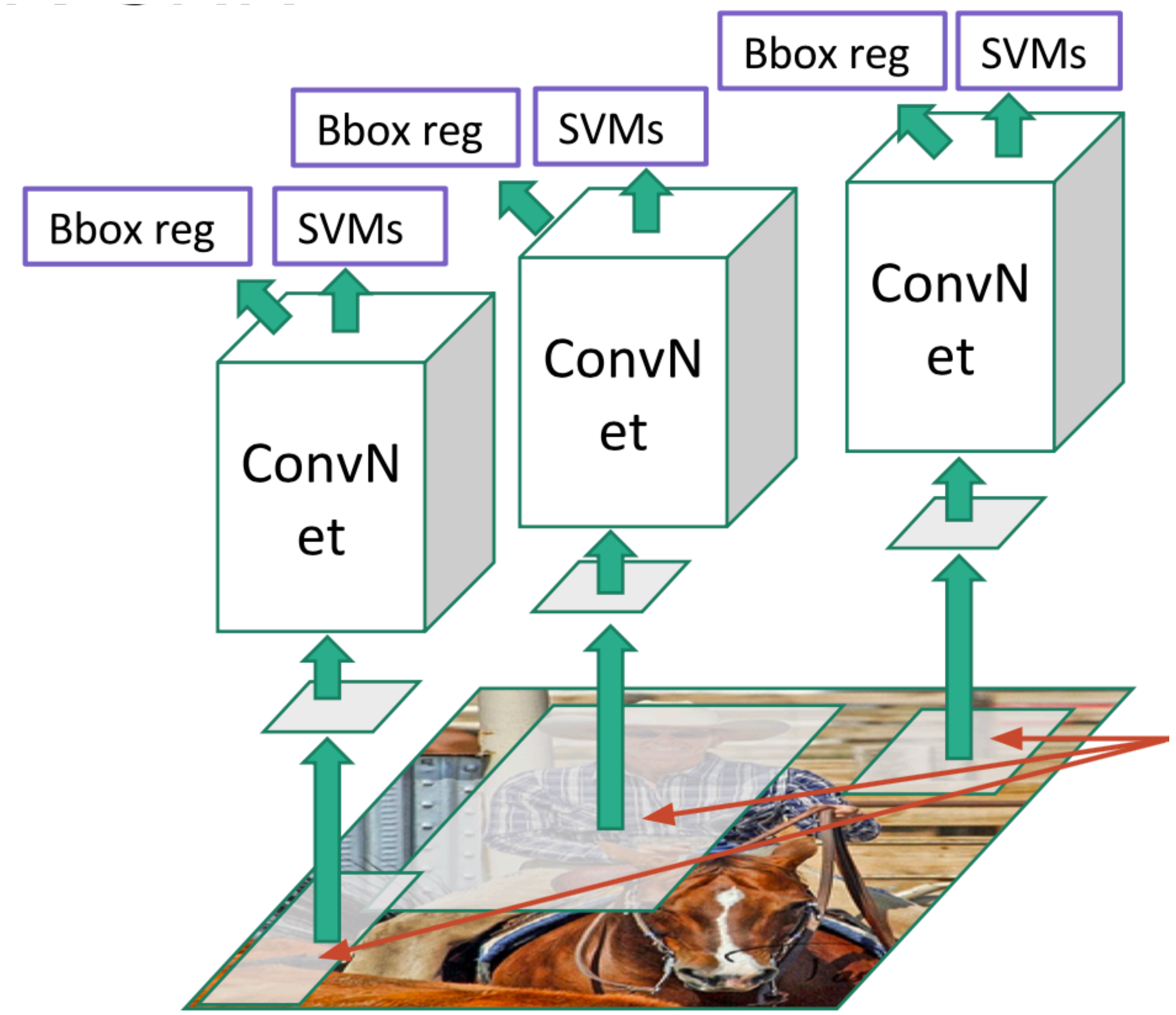
# **R-CNN:** Training

[ Girshick et al, CVPR 2014 ]

## **Fine-tuning ImageNet** CNN on object proposal patches

— > 50% Intersection-over-Union overlap with GT considered “object” others “background”

— batches of 128 (**32 positives, 96 negatives**)




* image from Ross Girshick

# **R-CNN:** Issues

[ Girshick et al, CVPR 2014 ]

## **Ad-hoc** training objectives

— Fine-tune network with softmax objective (**log** loss)

— Train post-hoc linear SVM (**hinge** loss)

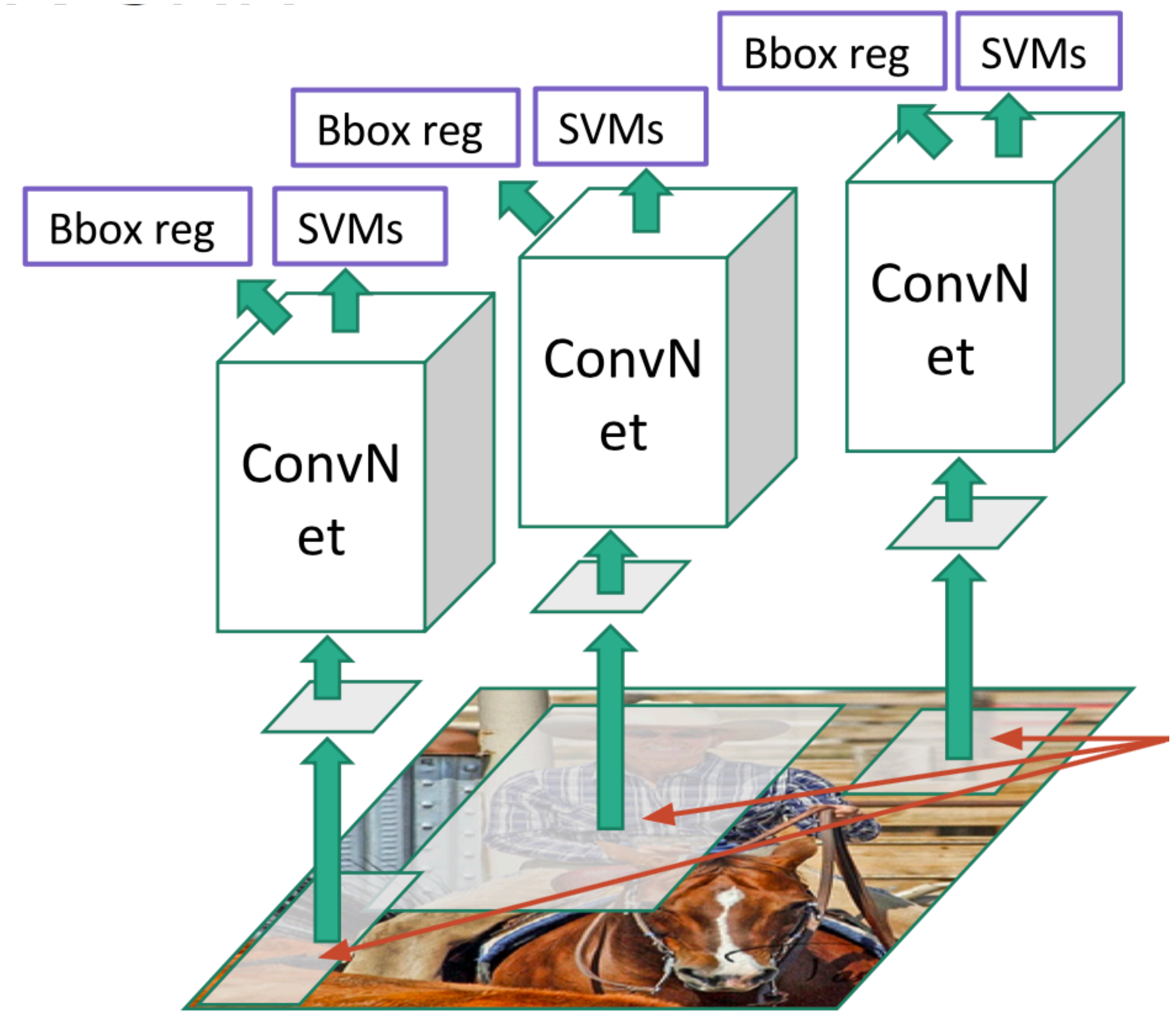
— Train post-hoc bounding-box regression (**least squares**)

## **Training** is slow

— 84 hours and takes a lot of disk space

## Inference / **Detection is slow**

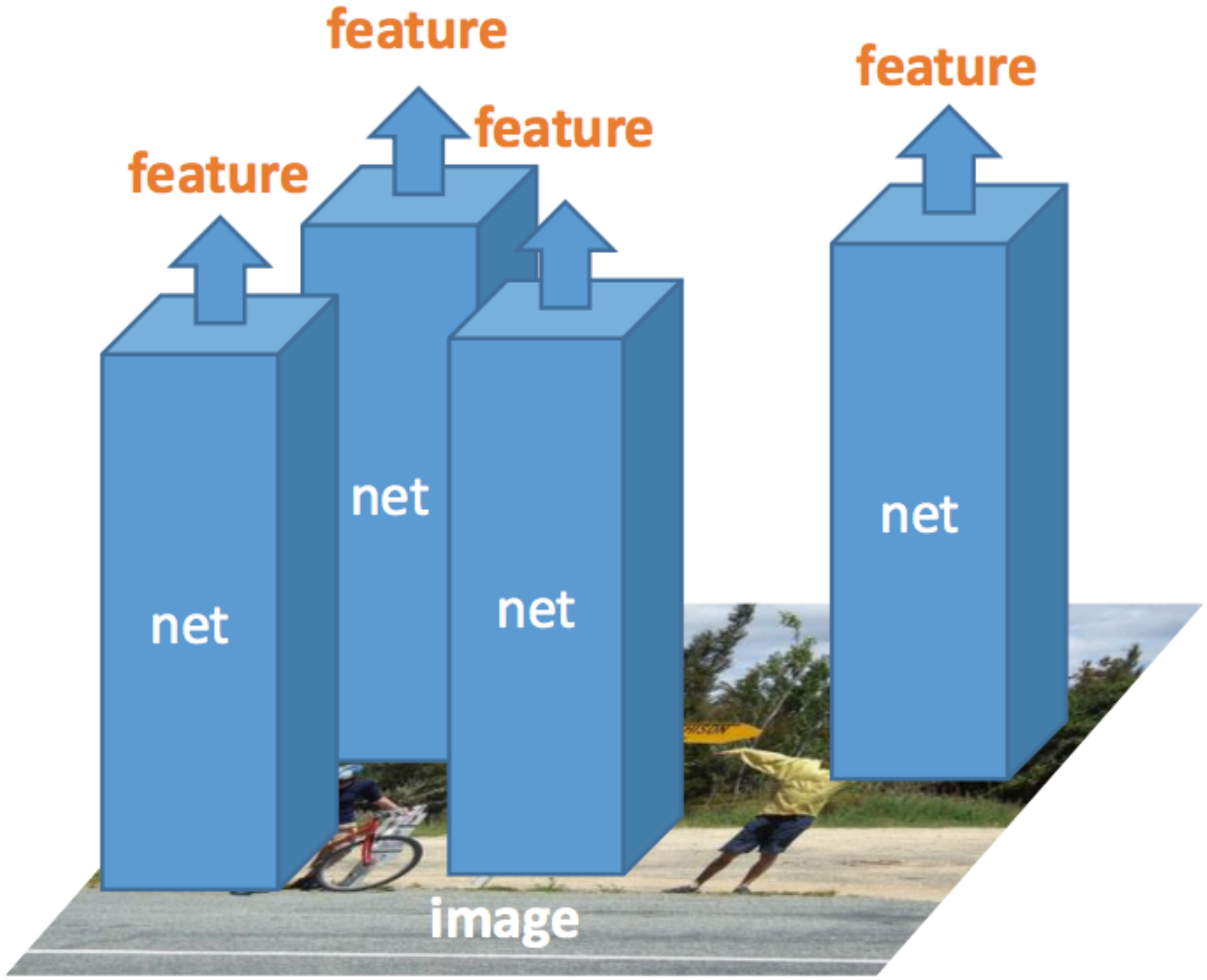
— 47 sec / image with VGG16 [ Simonyan et al, ICLR 2015 ]



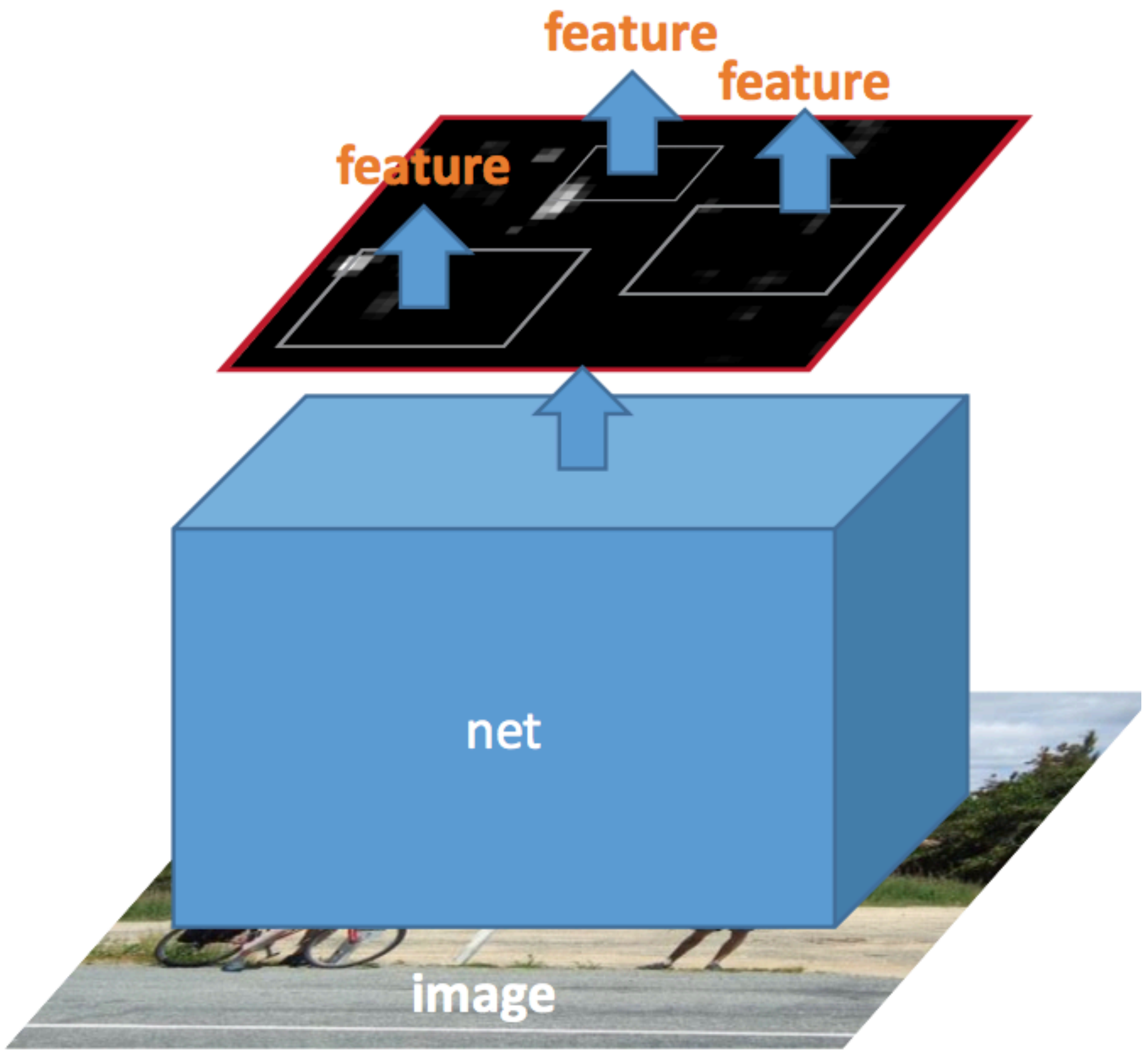

* image from Ross Girshick

# R-CNN vs. SPP

[ He et al, ECCV 2014 ]




**R-CNN**
2000 nets on image regions




**SPP-net**
**1 net on full image**

# Fast **R-CNN**

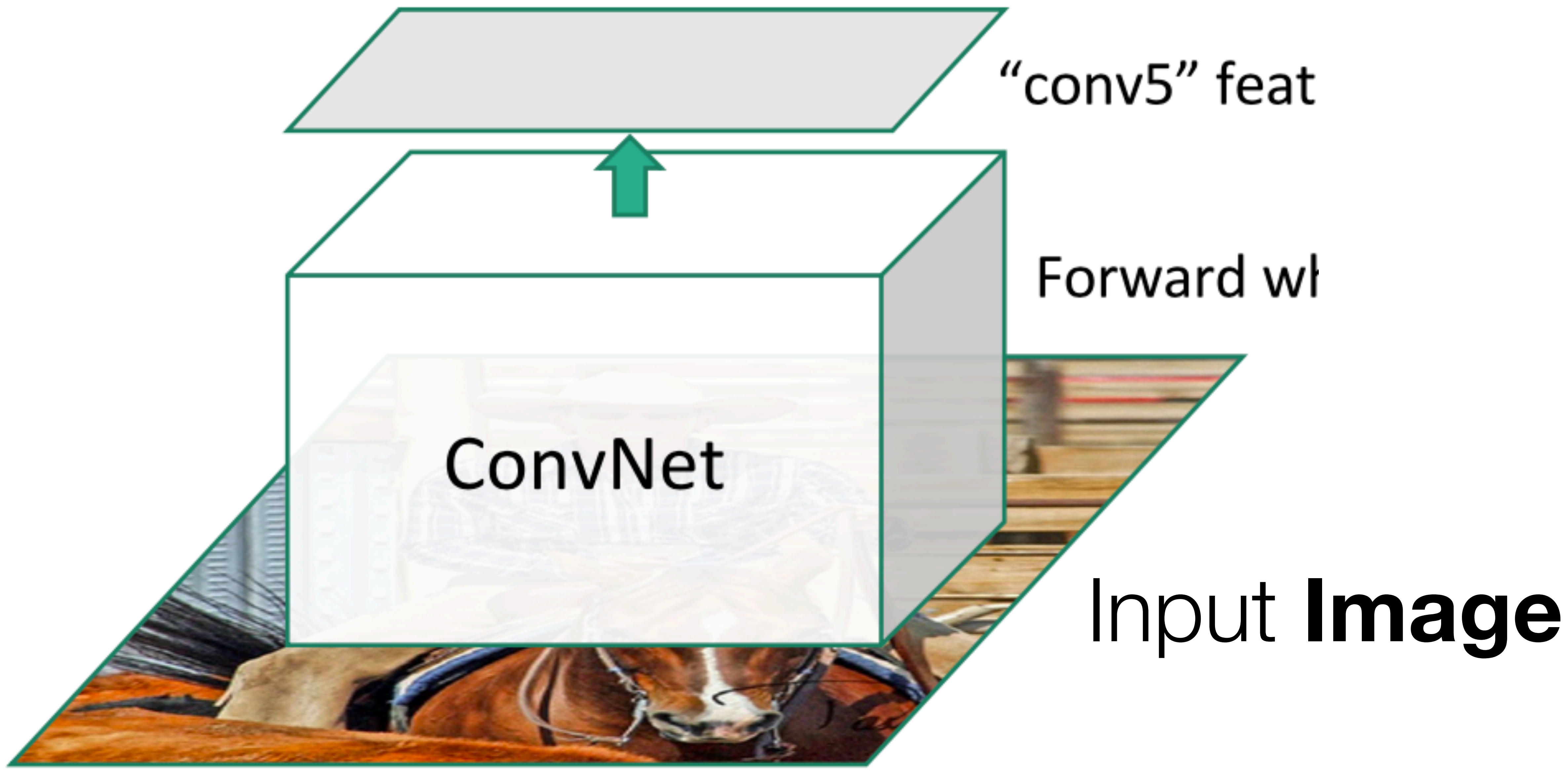
[ Girshick et al, ICCV 2015 ]

Input **Image**

* image from Ross Girshick

# Fast **R-CNN**

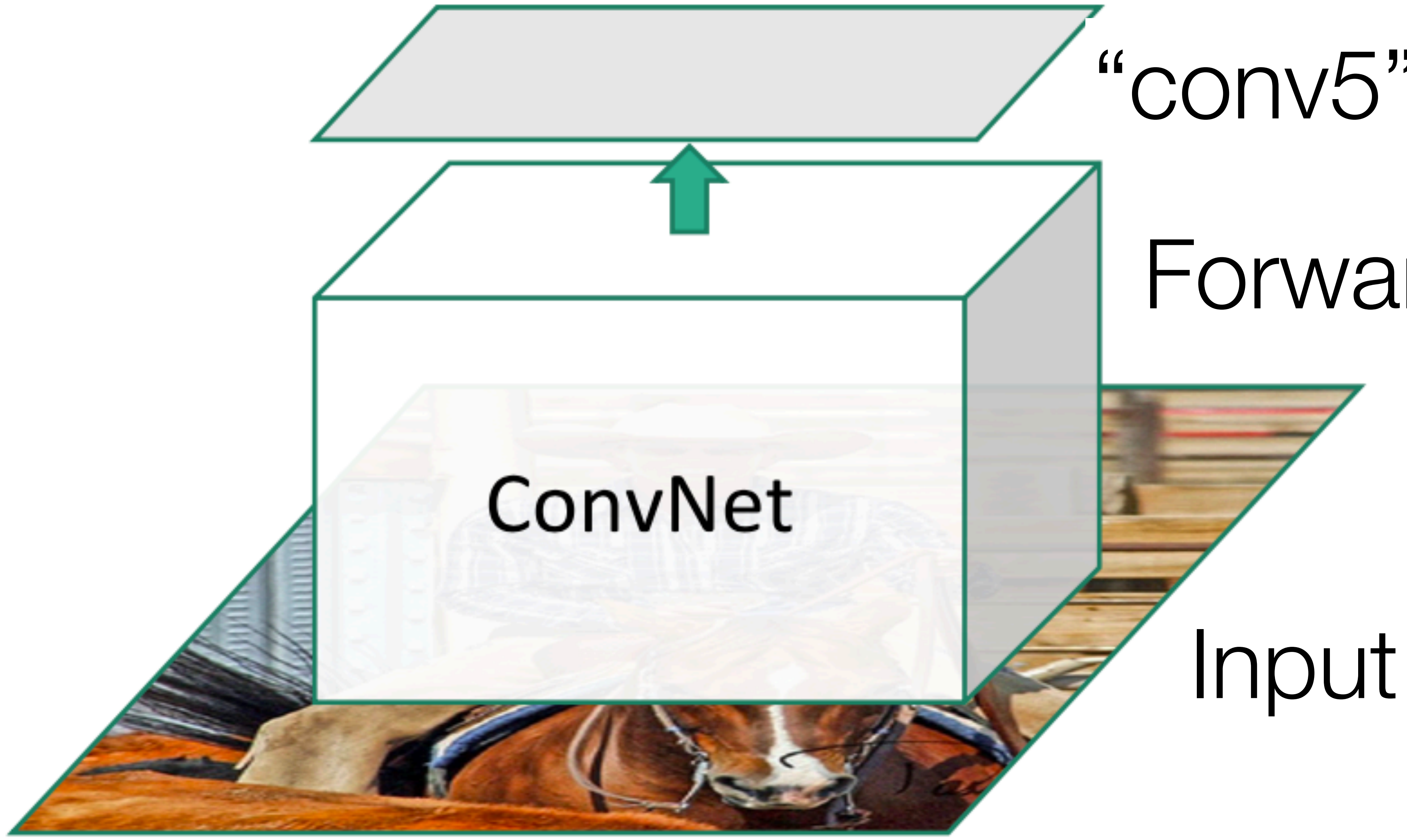
[ Girshick et al, ICCV 2015 ]




* image from Ross Girshick

# Fast **R-CNN**

[ Girshick et al, ICCV 2015 ]




“conv5” feature map

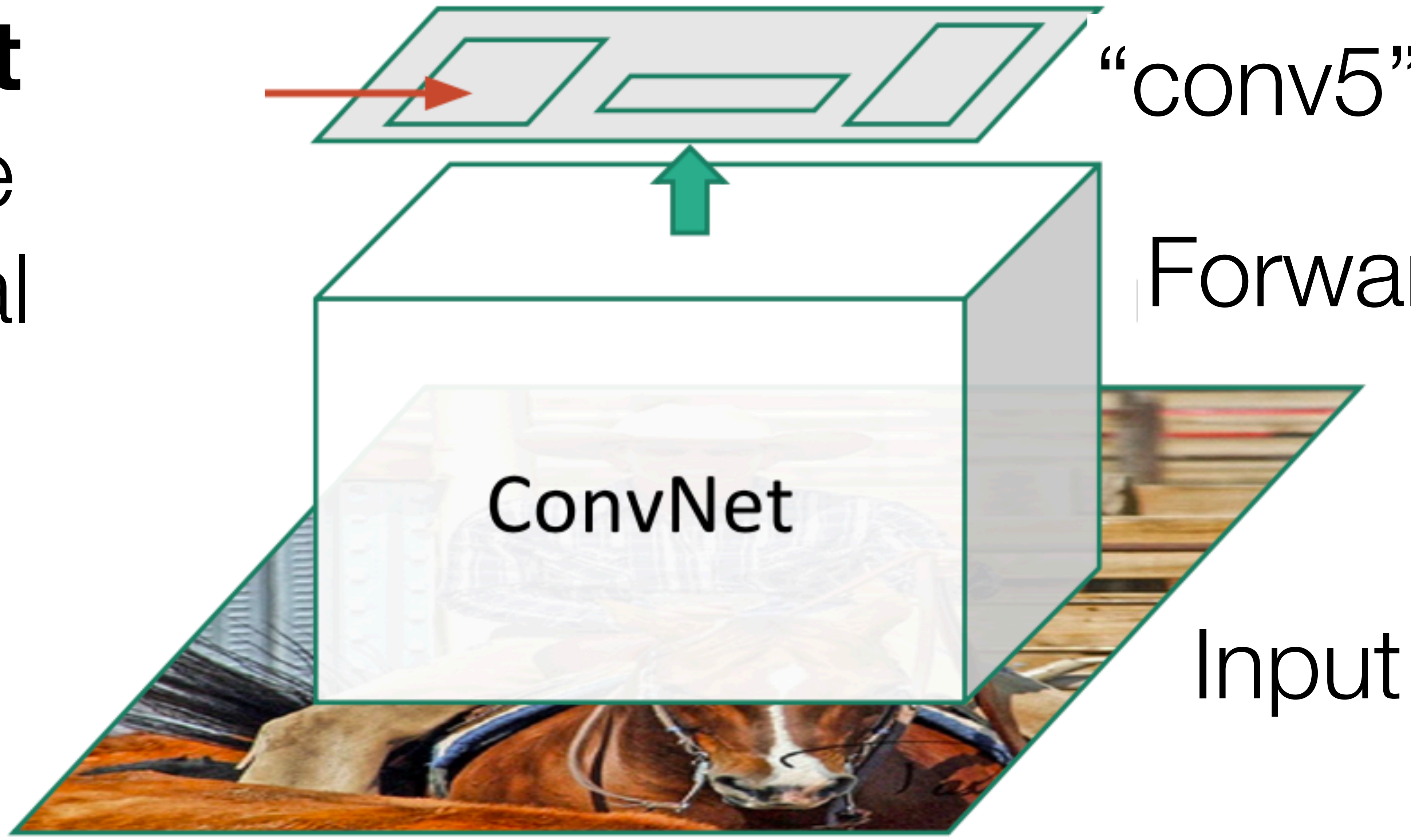
Forward prop the **whole image** through CNN

Input **Image**

* image from Ross Girshick

# Fast **R-CNN**

[ Girshick et al, ICCV 2015 ]

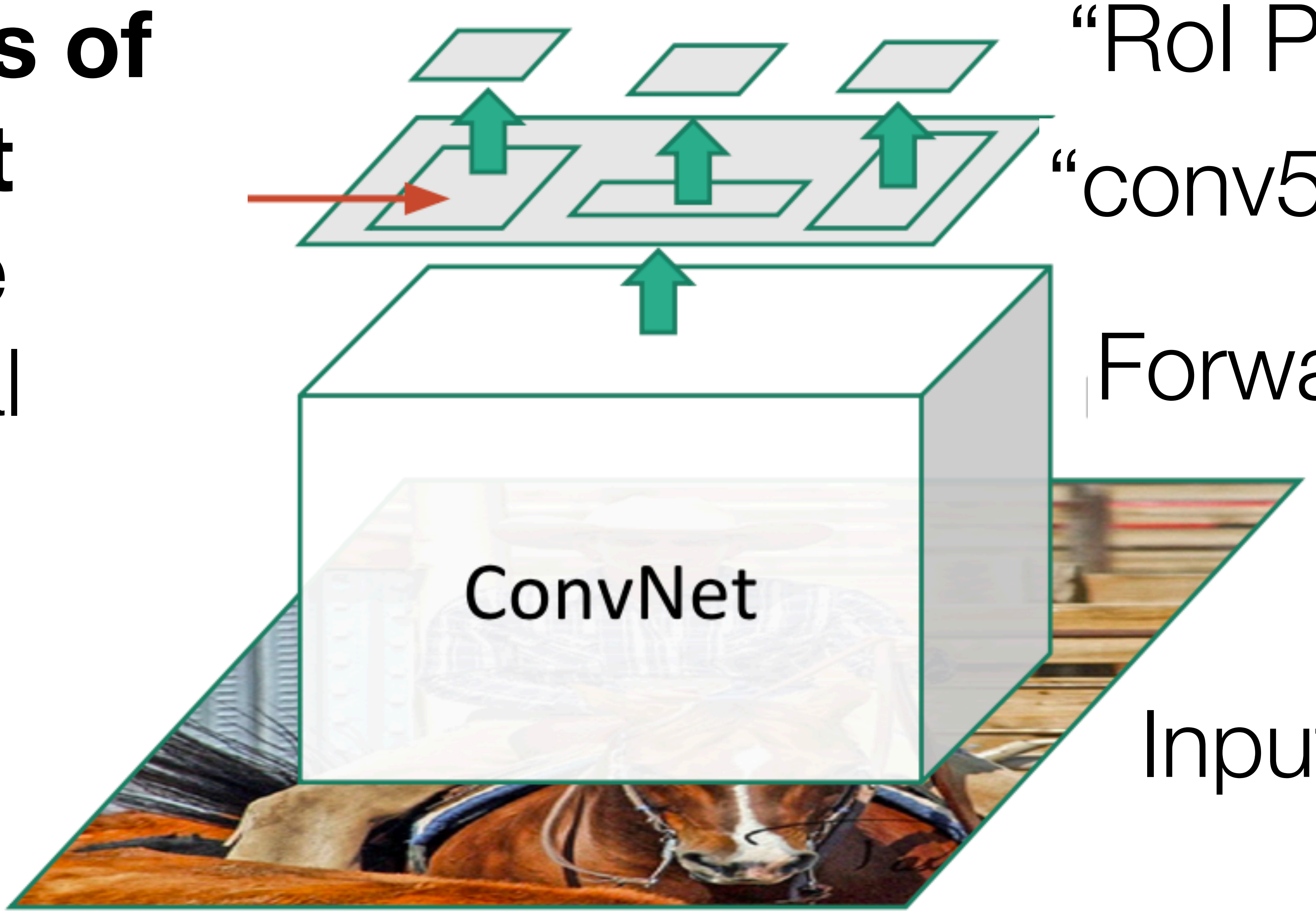



* image from Ross Girshick

# Fast **R-CNN**

[ Girshick et al, ICCV 2015 ]

**Regions of Interest** from the proposal method




“RoI Pooling” layer

“conv5” feature map

Forward prop the **whole image** through CNN

Input **Image**

Girshick, “Fast R-C
Figure copyright R

* image from Ross Girshick