THE UNIVERSITY OF BRITISH COLUMBIA

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

Lecture 5: Convolutional Neural Networks (Part 2)

Course **Logistics**

— **Assignment 2** questions

— **Assignment 1**

Assignment 1 - Part A, Common Mistakes

Computational **Graph**

— Had to break functions to the elementary units

— Expectation is a single graph

Assignment 1 - Part A, Common Mistakes

Computational **Graph**

— Had to break functions to the elementary units

— Expectation is a single graph

Forward Pass

— Show work

— Had to do computations right

Assignment 1 - Part A, Common Mistakes

Computational **Graph**

— Had to break functions to the elementary units

— Expectation is a single graph

Forward Pass

— Show work

— Had to do computations right

AutoDiff **Forward Mode**

— Had to have two passes (one for each **input** variable)

— Needed to show enough work, that I was fairly certain you understood the process

Assignment 1 - Part A, Common Mistakes

Computational **Graph**

— Had to break functions to the elementary units

— Expectation is a single graph

Forward Pass

— Show work

— Had to do computations right

AutoDiff **Forward Mode**

— Had to have two passes (one for each **input** variable)

— Needed to show enough work, that I was fairly certain you understood the process

AutoDiff **Backwards Mode / BackProp**

— Had to have two passes (one for each **output** variable)

— Needed to show enough work, that I was fairly certain you understood the process

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

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

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

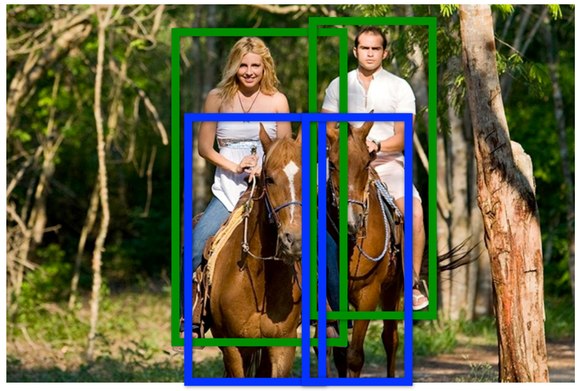
Categorization

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

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

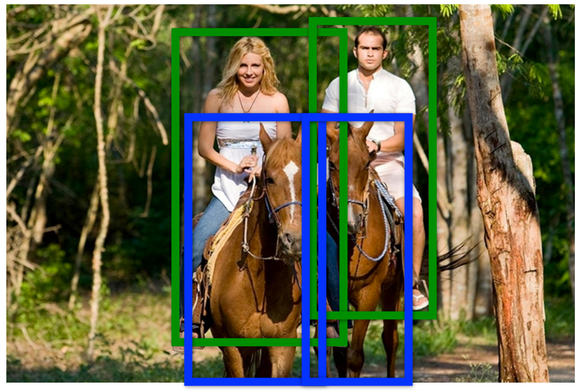
Categorization

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

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

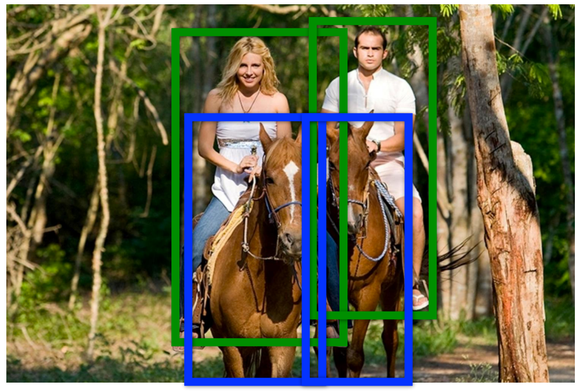
Categorization

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

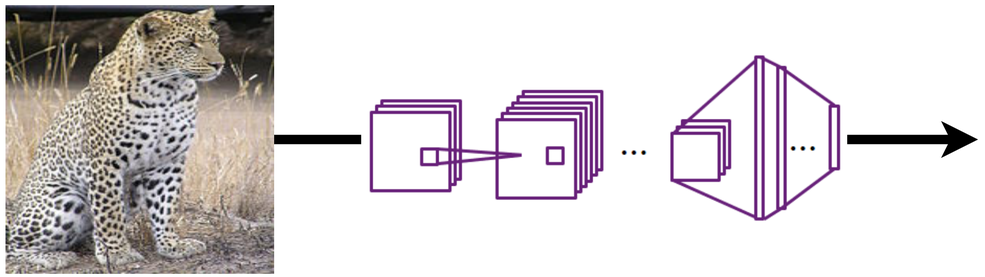
Horse1
Horse2
Person1
Person2

Object **Classification**

Category	**Prediction**
Dog	No
Cat	No
Couch	No
Flowers	No
Leopard	**Yes**
…	…

Problem: For each image predict which category it belongs to out of a fixed set

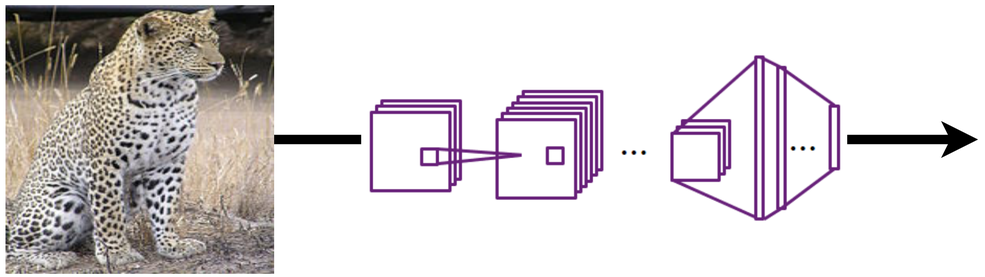
Object **Classification**

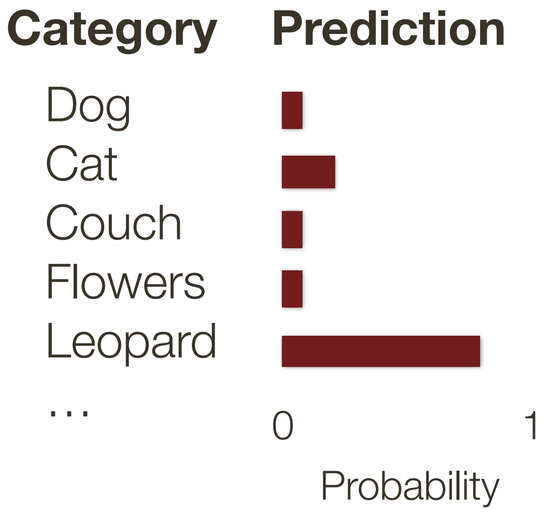


Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Leopard	**Yes**
…	…

Problem: For each image predict which category it belongs to out of a fixed set

Object **Classification**

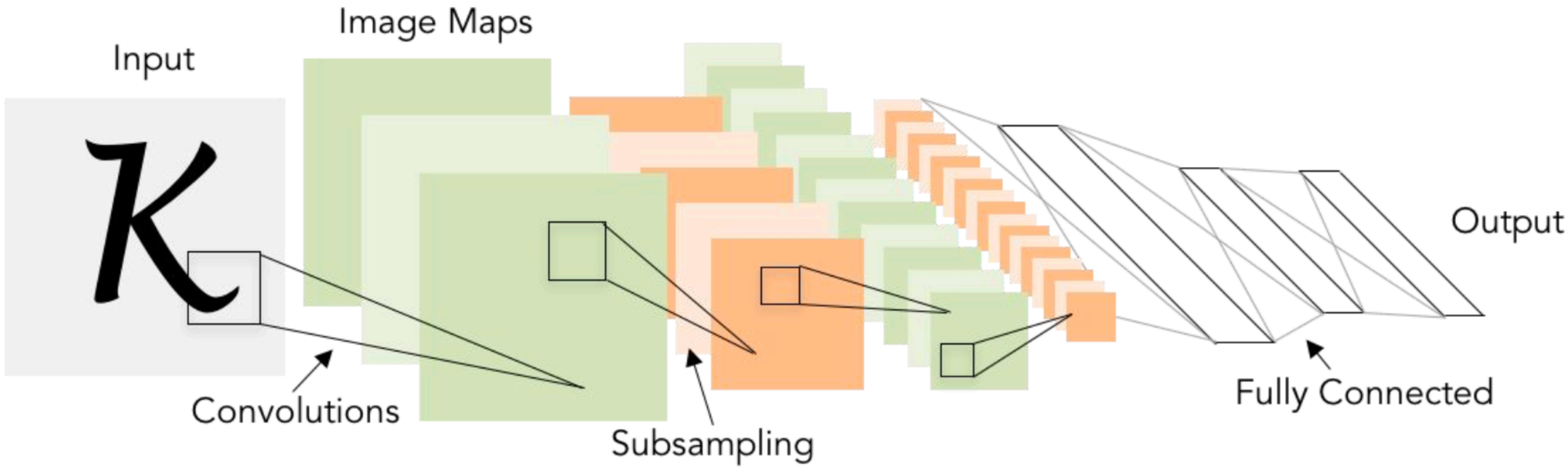




Problem: For each image predict which category it belongs to out of a fixed set

CNN Architectures: LeNet-5

[LeCun et al., 1998]



Architecture: CONV —> POOL —> CONV —> POOL —> FC —> FC

Conv filters: 5x5, Stride: 1

Pooling: 2x2, Stride: 2

ImageNet **Dataset**

Over **14 million** (high resolution) web **images**

Roughly labeled with **22K synset** categories

Labeled on Amazon Mechanical Turk (AMT)

Popular Synsets

Animal

fish
bird
mammal
invertebrate

Plant

tree
flower
vegetable

Activity

sport

Material

fabric

Instrumentation

utensil
appliance
tool
musical instrument

Scene

room
geological formation

Food

beverage

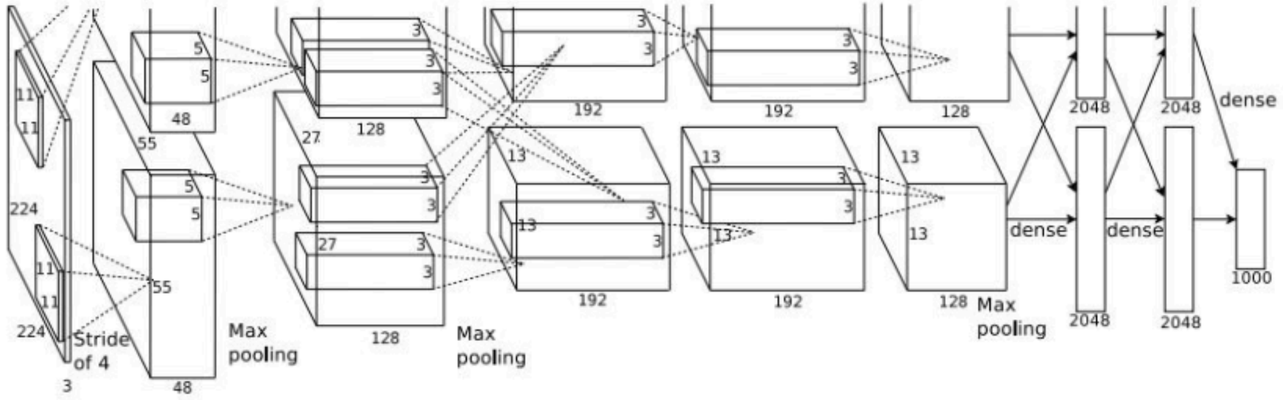
ImageNet **Competition** (ILSVRC)

Annual competition of image classification at scale

Focuses on a subset of **1K synset** categories

Scoring: need to predict true label within top K (K=5)

AlexNet



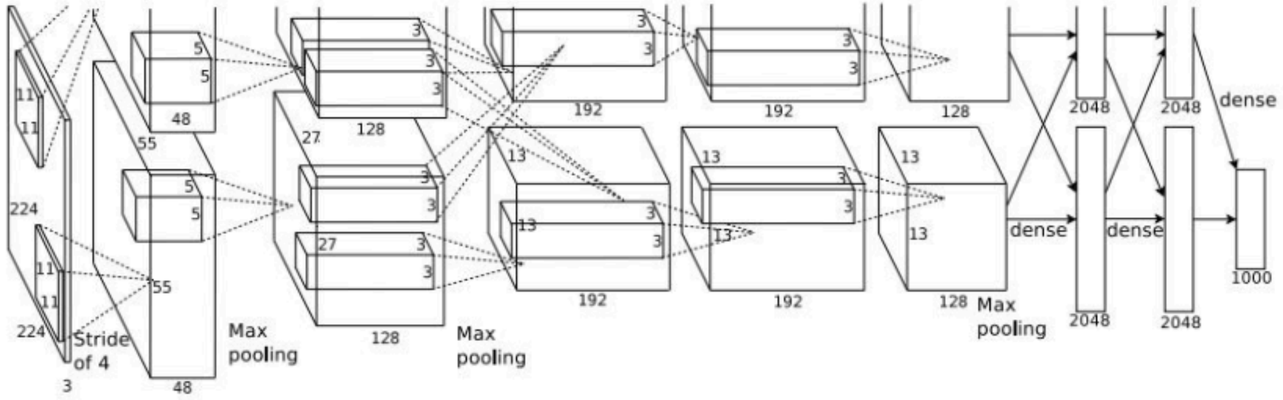
[Krizhevsky et al., 2012]

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

Input: 227 x 227 x 3 images

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

AlexNet



[Krizhevsky et al., 2012]

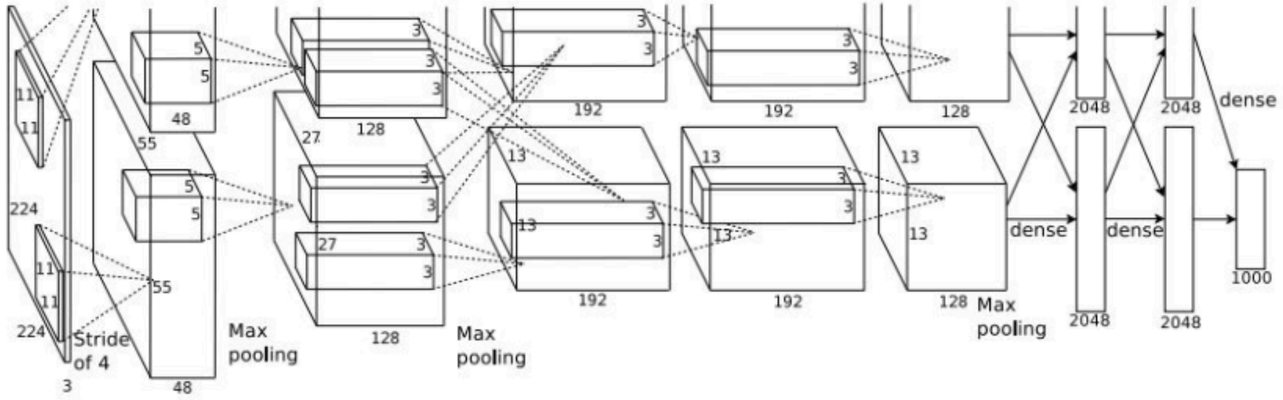
Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

Input: 227 x 227 x 3 images

CONV1: 96 11 x 11 filters applied at stride 4

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

AlexNet



[Krizhevsky et al., 2012]

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

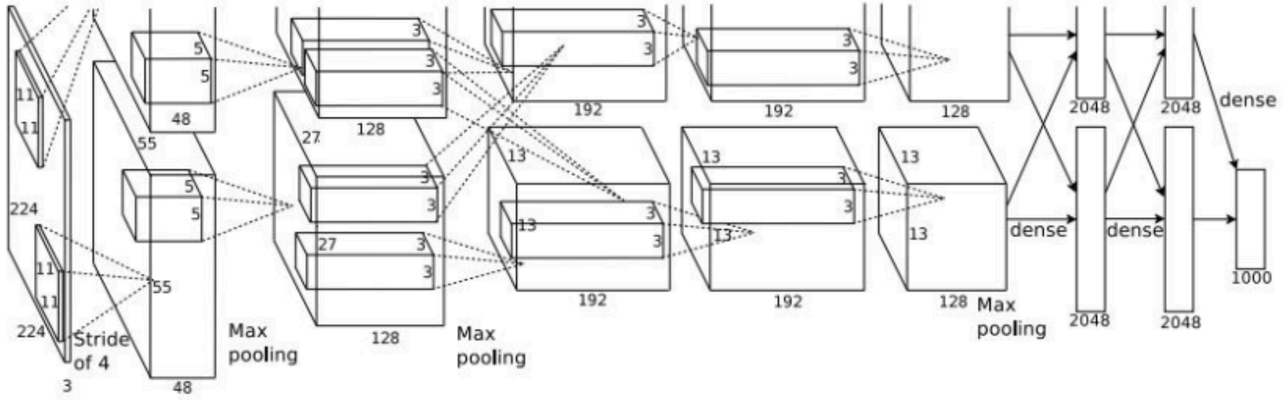
Input: 227 x 227 x 3 images

CONV1: 96 11 x 11 filters applied at stride 4

Output: 55 x 55 x 96

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

AlexNet



[Krizhevsky et al., 2012]

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

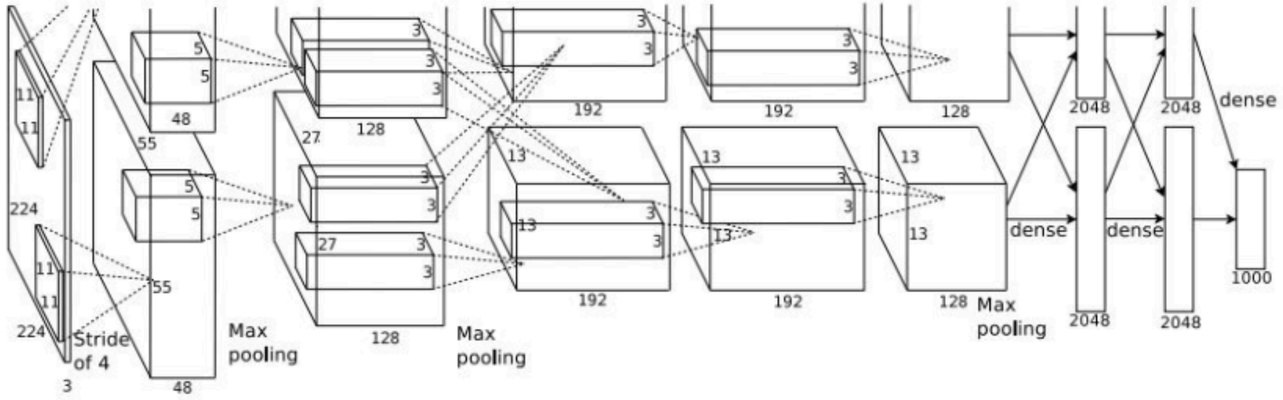
Input: 227 x 227 x 3 images

CONV1: 96 11 x 11 filters applied at stride 4

Output: 55 x 55 x 96 **Parameters:** 35K

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

AlexNet



[Krizhevsky et al., 2012]

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

Input: 227 x 227 x 3 images

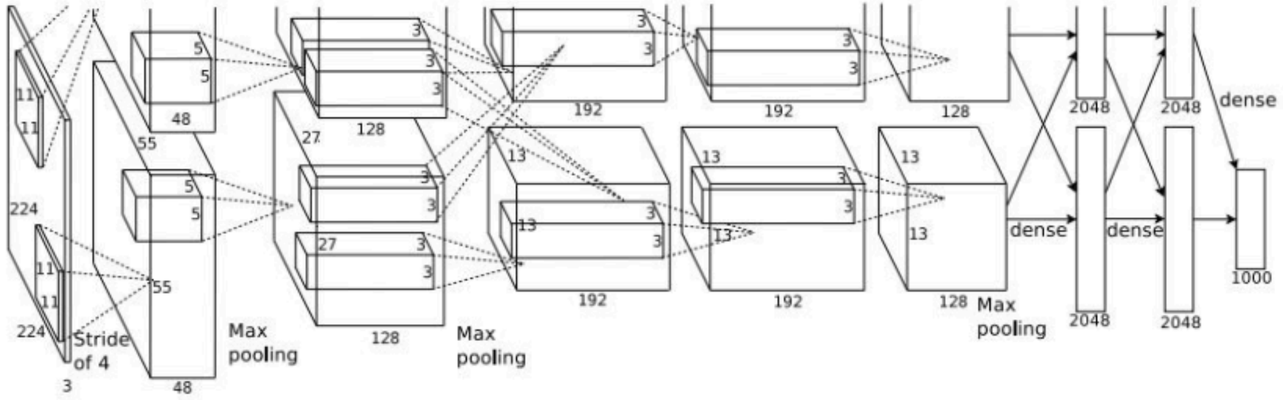
CONV1: 96 11 x 11 filters applied at stride 4

Output: 55 x 55 x 96 **Parameters:** 35K

MAX POOL1: 96 11 x 11 filters applied at stride 4

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

AlexNet



[Krizhevsky et al., 2012]

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

Input: 227 x 227 x 3 images

CONV1: 96 11 x 11 filters applied at stride 4

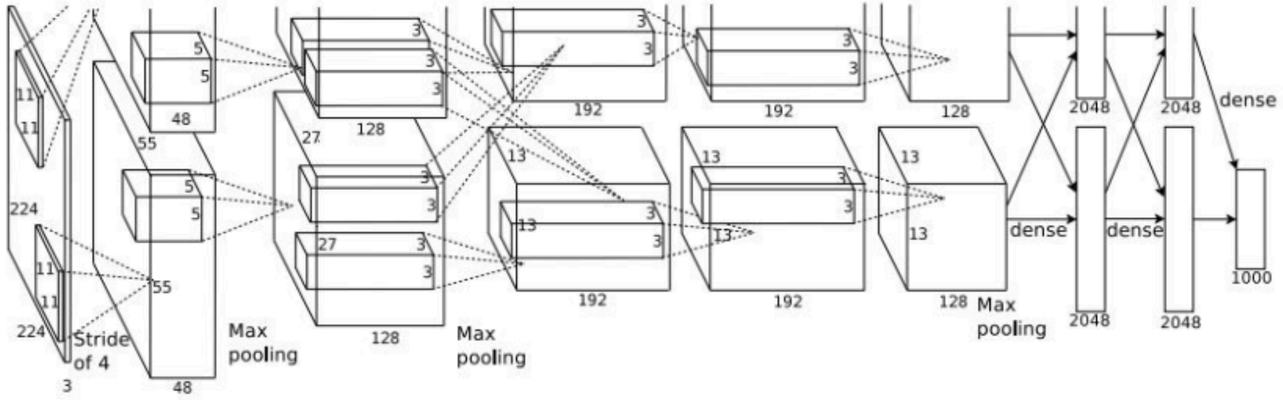
Output: 55 x 55 x 96 **Parameters:** 35K

MAX POOL1: 96 11 x 11 filters applied at stride 4

Output: 27 x 27 x 96

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

AlexNet



[Krizhevsky et al., 2012]

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

Input: 227 x 227 x 3 images

CONV1: 96 11 x 11 filters applied at stride 4

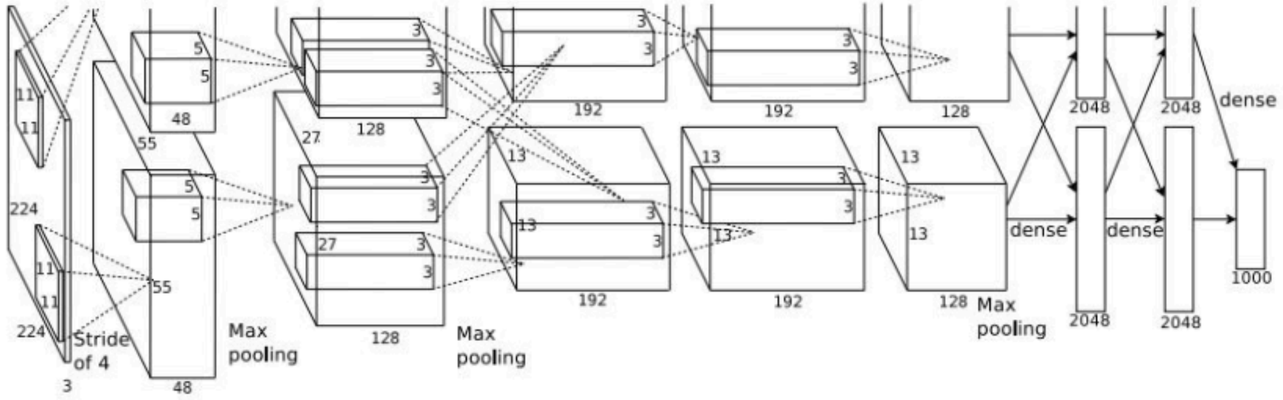
Output: 55 x 55 x 96 **Parameters:** 35K

MAX POOL1: 96 11 x 11 filters applied at stride 4

Output: 27 x 27 x 96 **Parameters:** 0

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

AlexNet



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

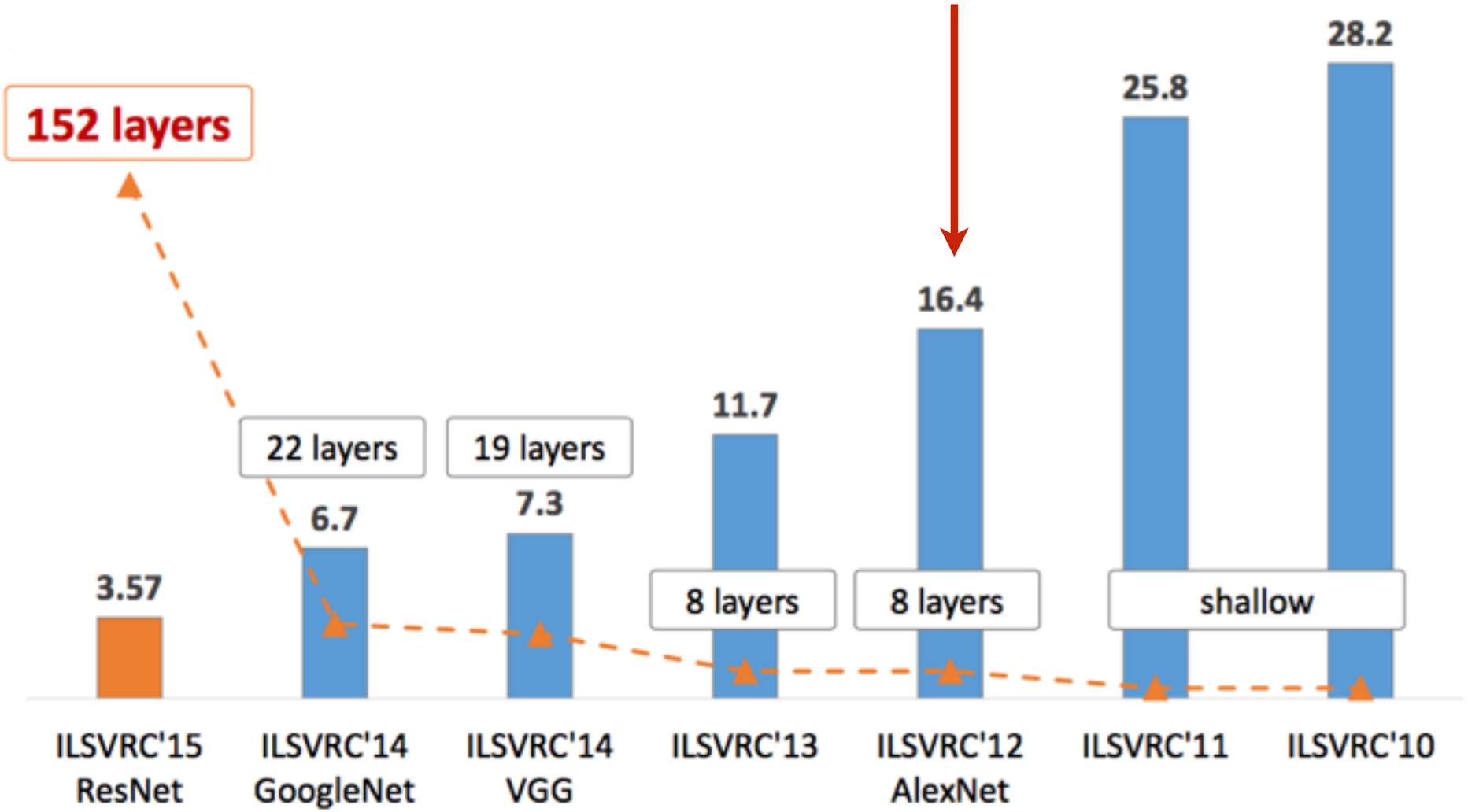
[Krizhevsky et al., 2012]

Details / Comments

— First use of ReLU
— Used contrast normalization layers
— Heavy data augmentation
— Dropout of 0.5
— Batch size of 128
— SGD Momentum (0.9)
— Learning rate (1e-2) reduced by 10 manually when validation accuracy plateaus
— L2 weight decay
— 7 CNN ensamble: 18.2% -> 15.4%

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

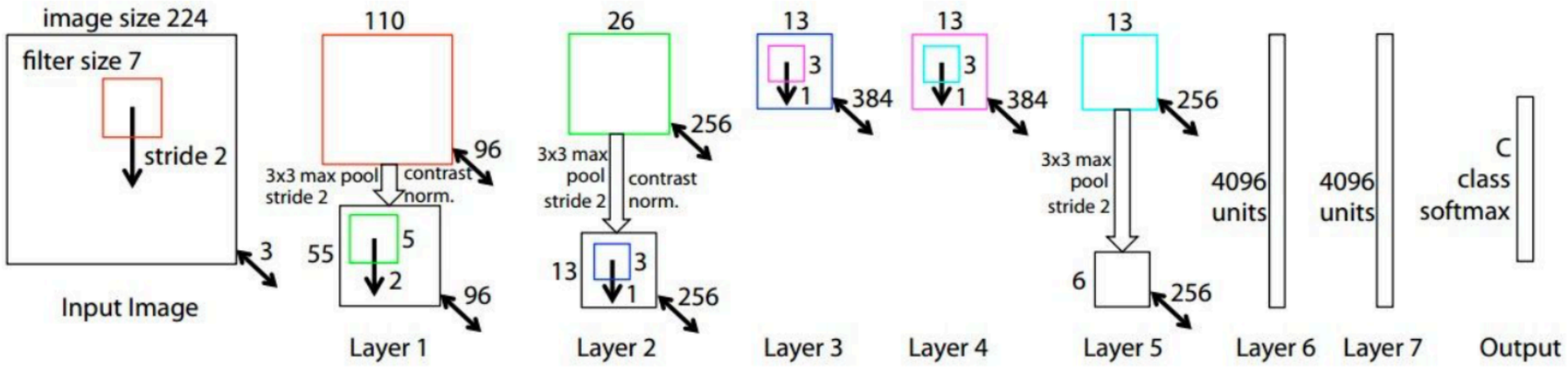
ILSVRC winner 2012



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

ZF Net

[Zeiler and Fergus, 2013]

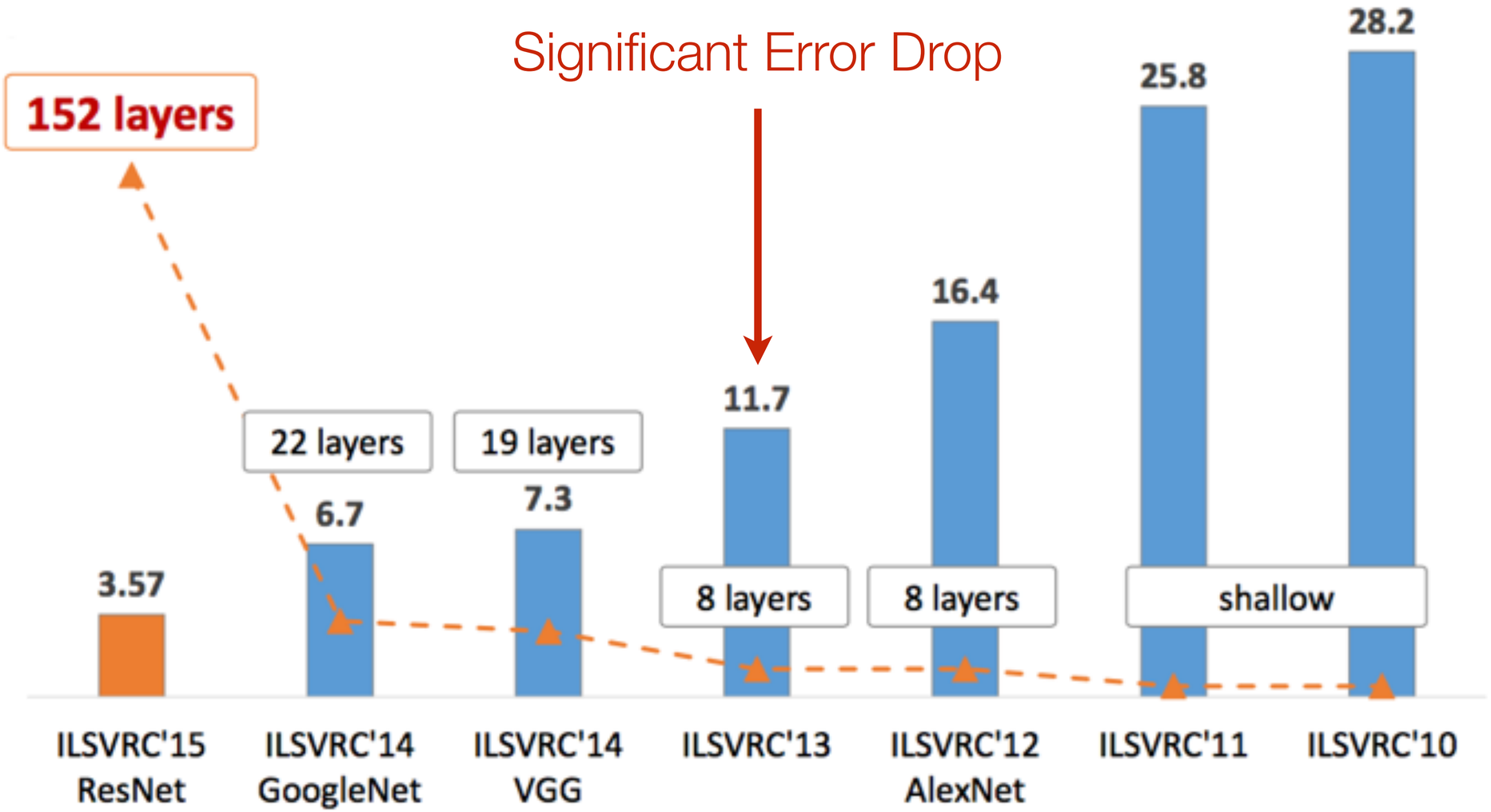


AlexNet with small modifications:

— CONV1 (11 x 11 stride 4) to (7 x 7 stride 2)
— CONV3 # of filters 384 -> 512
— CONV4 # of filters 384 -> 1024
— CONV5 # of filters 256 -> 512

* 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**

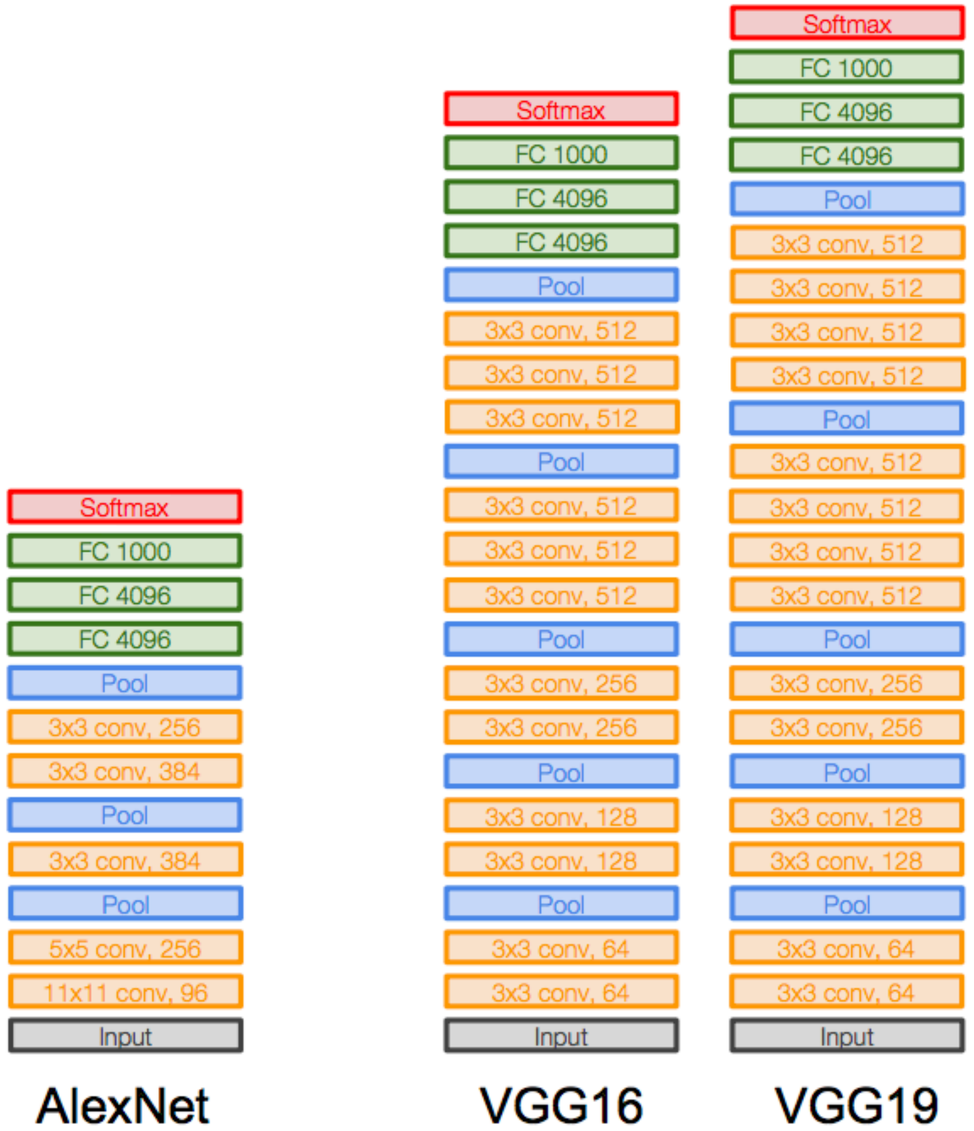
VGG Net

[Simonyan and Zisserman, 2014]

Trend:

—smaller filters (3 x 3)

—deeper network (16 or 19 vs. 8 in AlexNet)



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

VGG Net

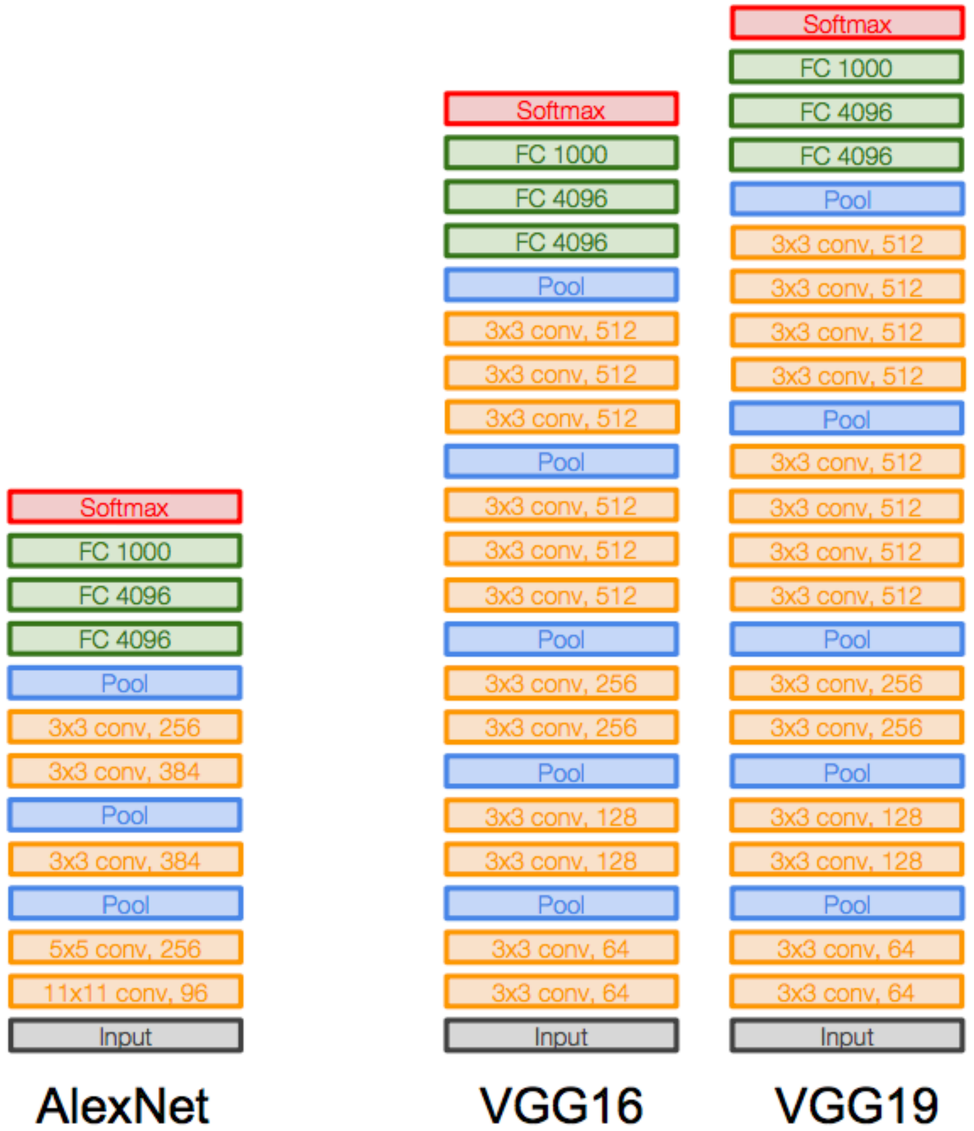
[Simonyan and Zisserman, 2014]

Trend:

—smaller filters (3 x 3)

—deeper network (16 or 19 vs. 8 in AlexNet)

Why?



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

VGG Net

[Simonyan and Zisserman, 2014]

Trend:

— smaller filters (3 x 3)

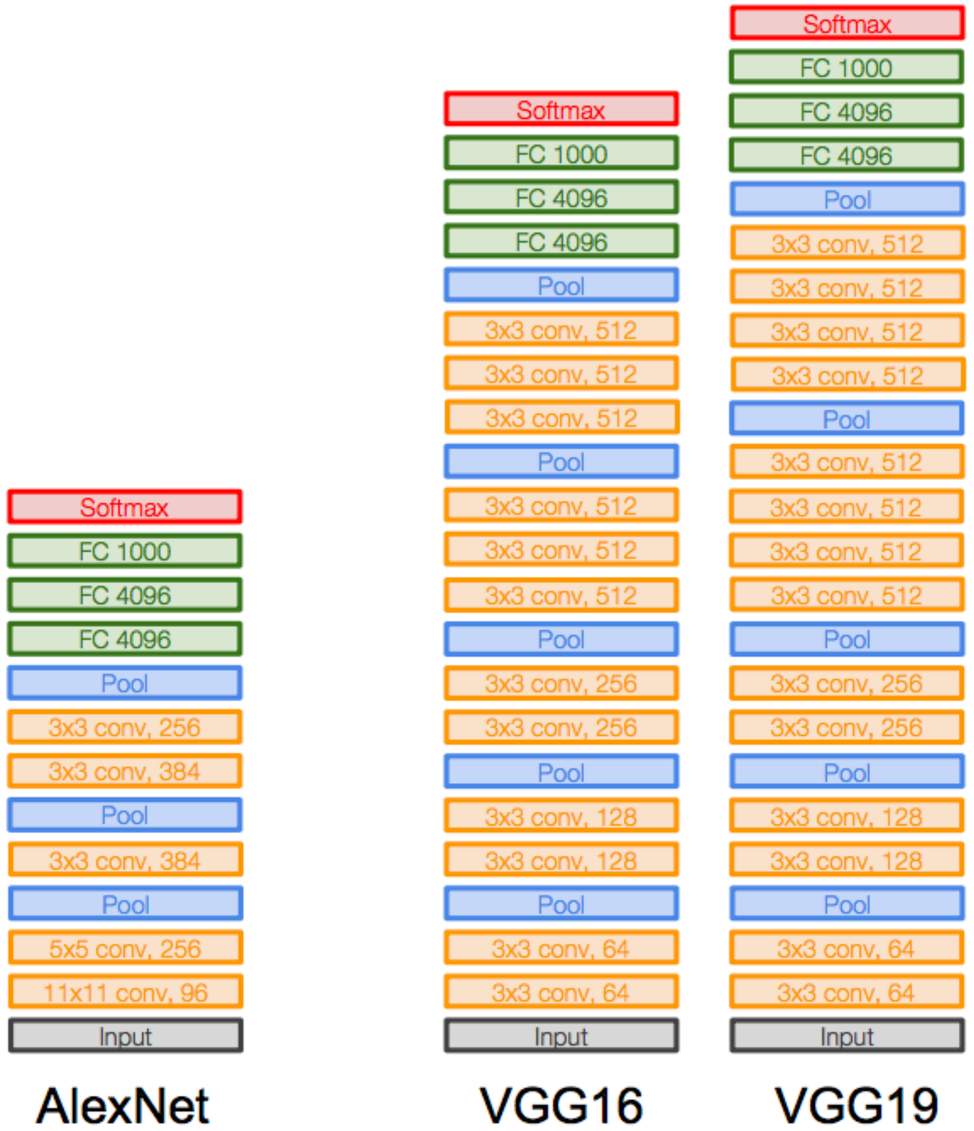
— deeper network (16 or 19 vs. 8 in AlexNet)

Why?

— **receptive field** of a 3 layer ConvNet with filter size = 3x3 is the same as 1 layer ConvNet with filter 7x7 (at stride 1)

— deeper = **more non-linearity** (so richer filters)

— **fewer parameters**



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

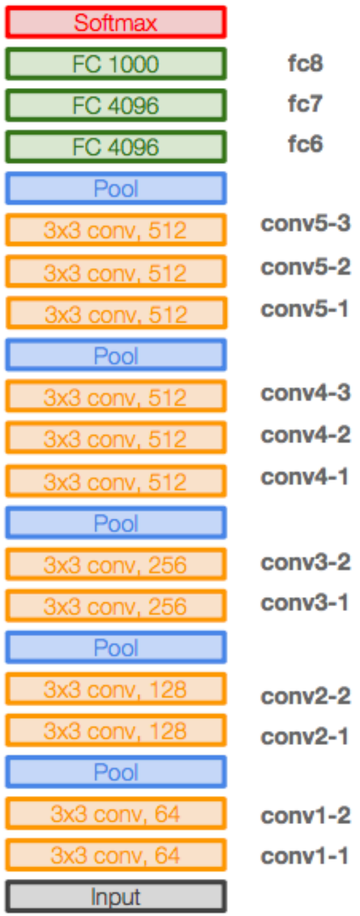
VGG Net

[Simonyan and Zisserman, 2014]

(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

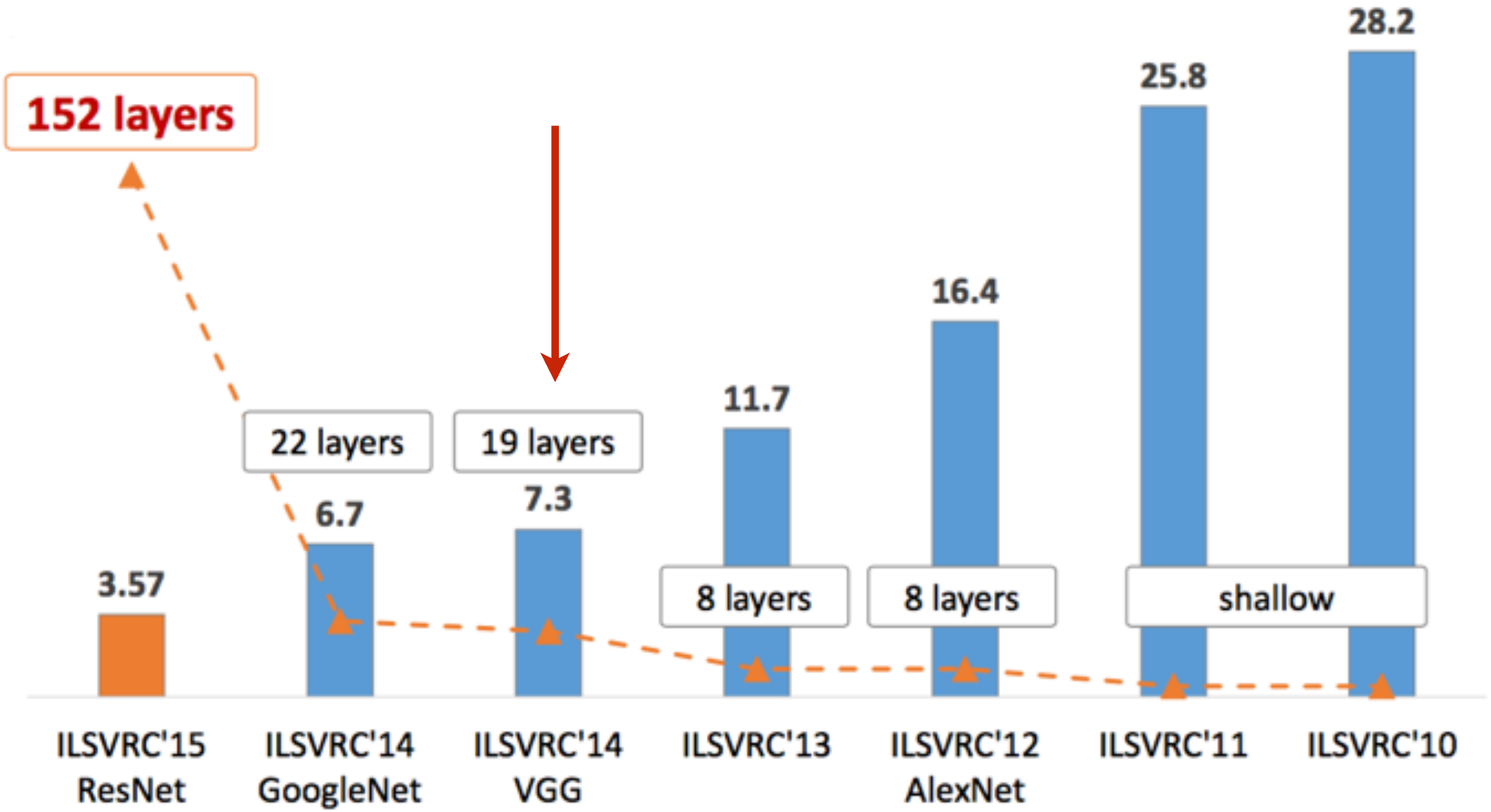
TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters



VGG16

* 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

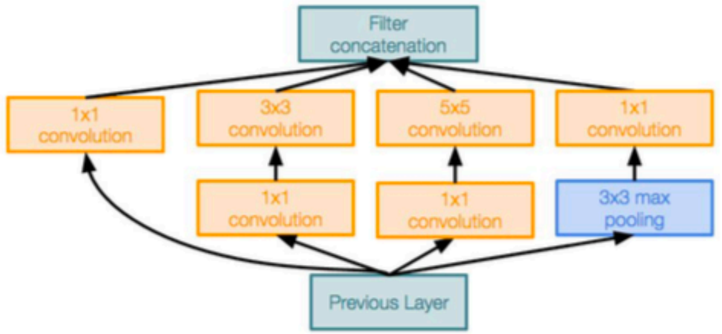
[Szegedy et al., 2014]

even deeper network with **computational efficiency**

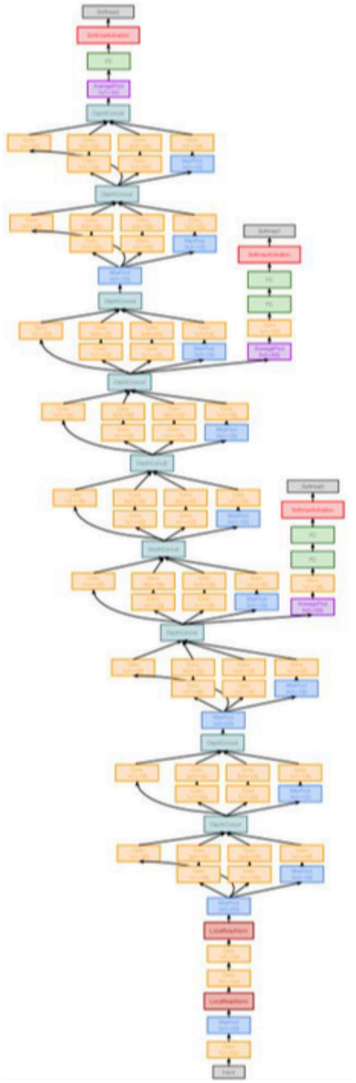
- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!

(12x less than AlexNet!)

- Better performance (@6.7 top 5 error)



Inception module

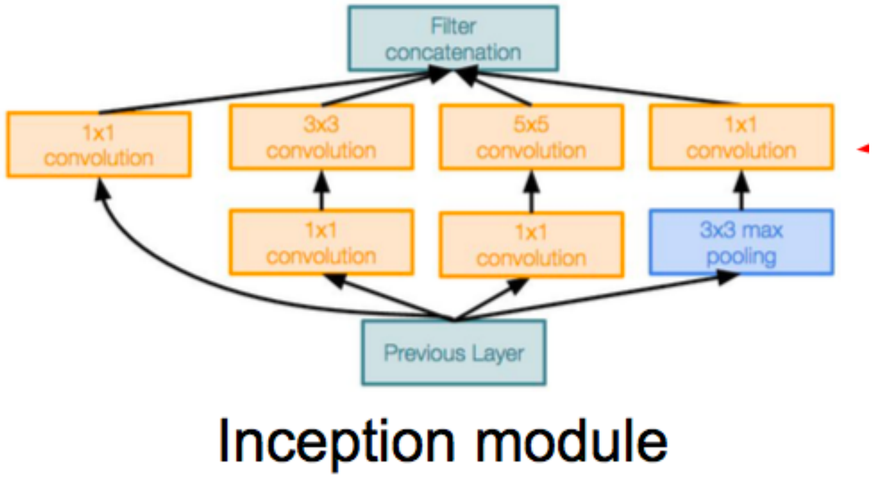


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

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



Inception module

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

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

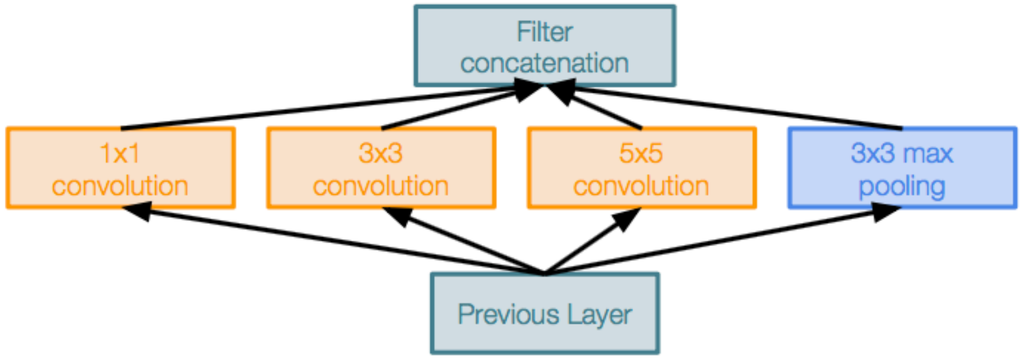
Idea: design good local topology (“network within network”) and then stack these modules

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**

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

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

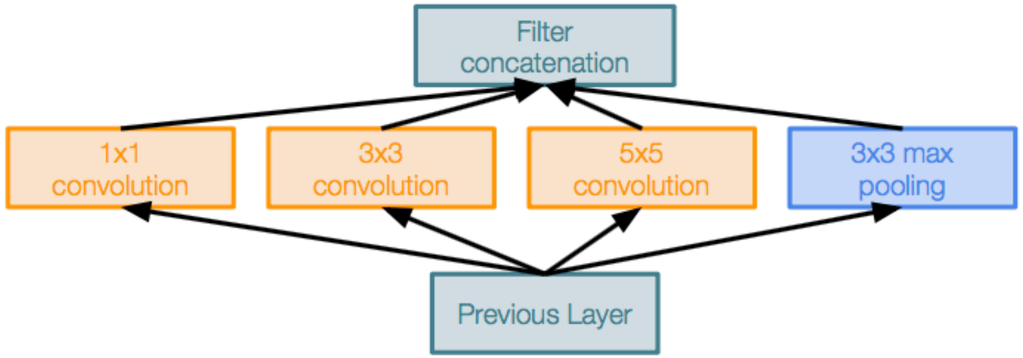
Apply **parallel filter operations** on the input from previous layer

— 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

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

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

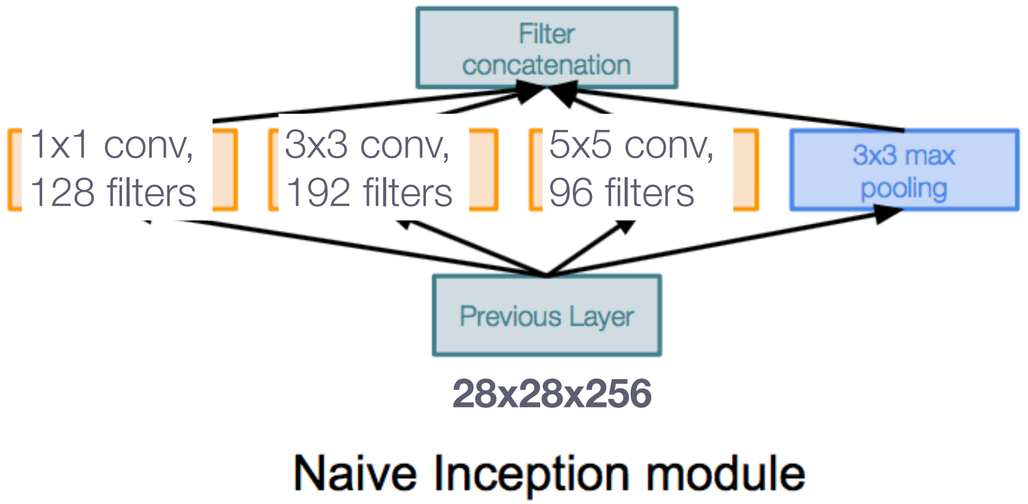
Idea: design good local topology (“network within network”) and then stack these modules

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



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

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

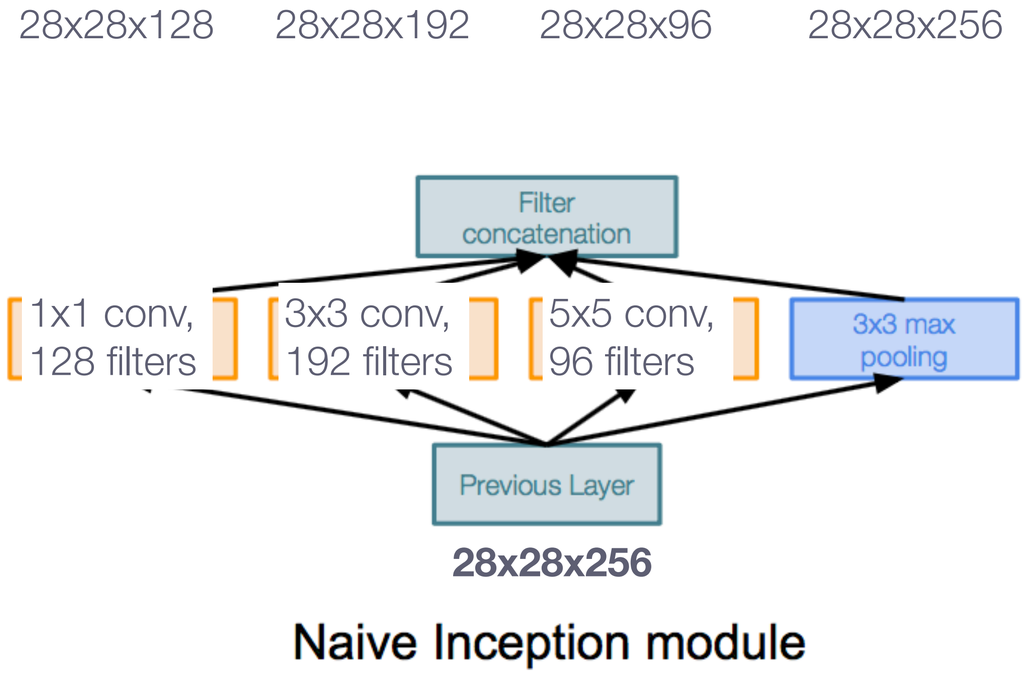
Idea: design good local topology (“network within network”) and then stack these modules

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**

GoogleLeNet: Inception Module

[Szegedy et al., 2014]

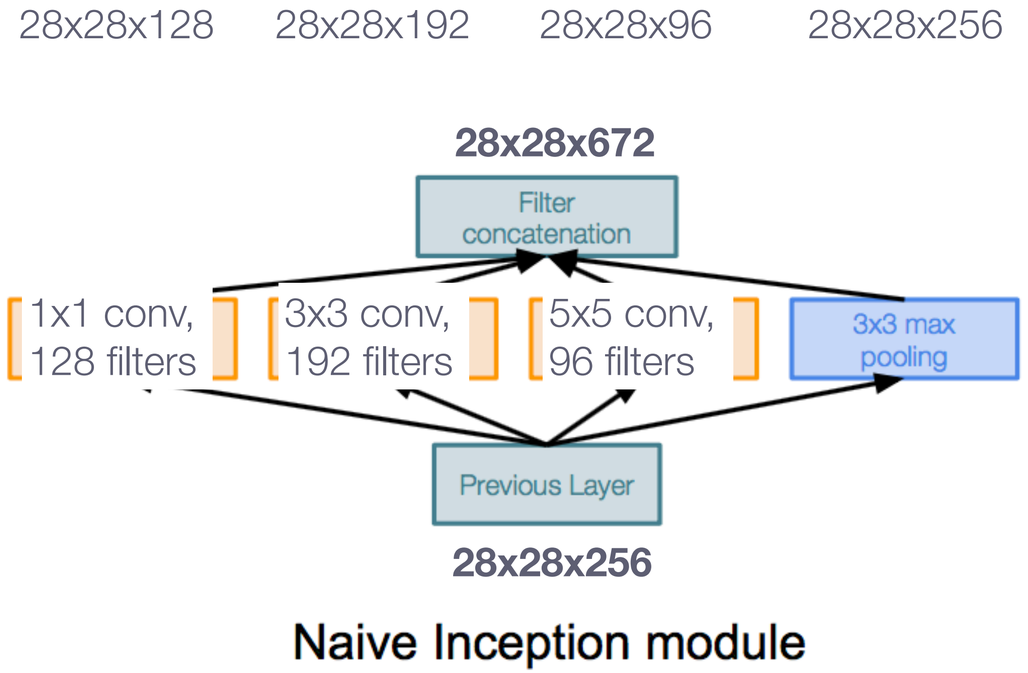
Idea: design good local topology (“network within network”) and then stack these modules

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

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

— Pooling operation (3x3)

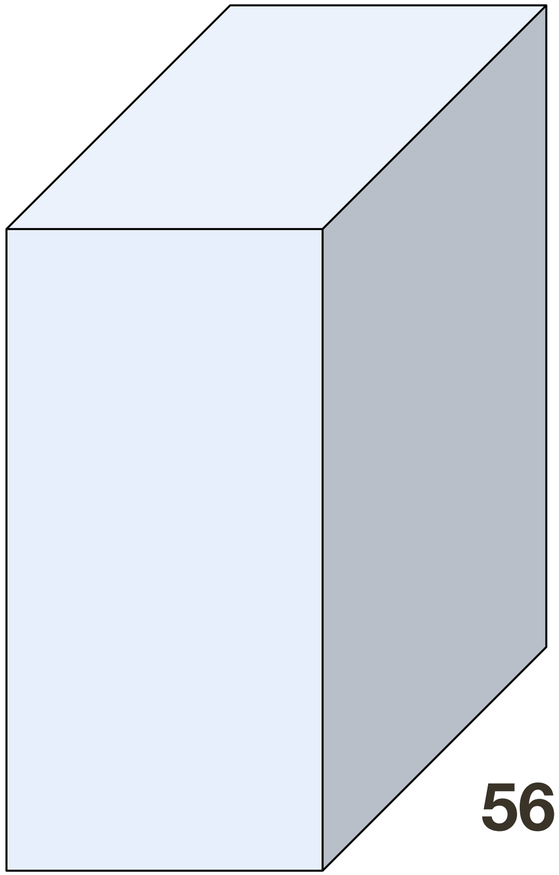
Concatenate all filter outputs together at output depth-wise



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

Convolutional Layer: **1x1 convolutions**

56 x 56 x 64 **image**



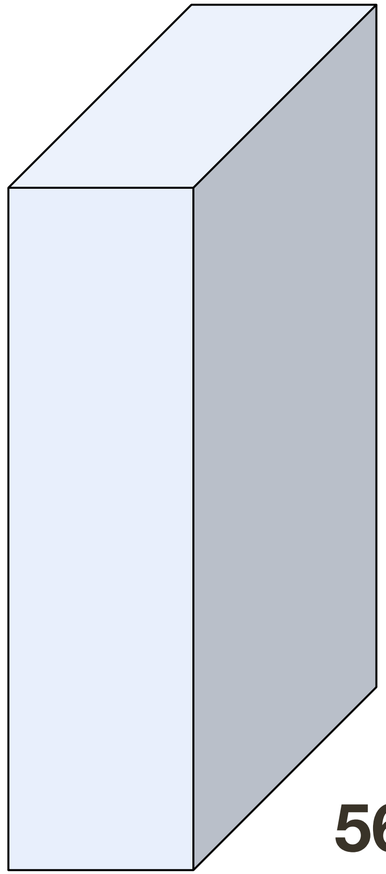
56 height

56 width

64 depth

32 **filters** of size, 1 x 1 x 64

56 x 56 x 32 **image**



56 height

56 width

32 depth

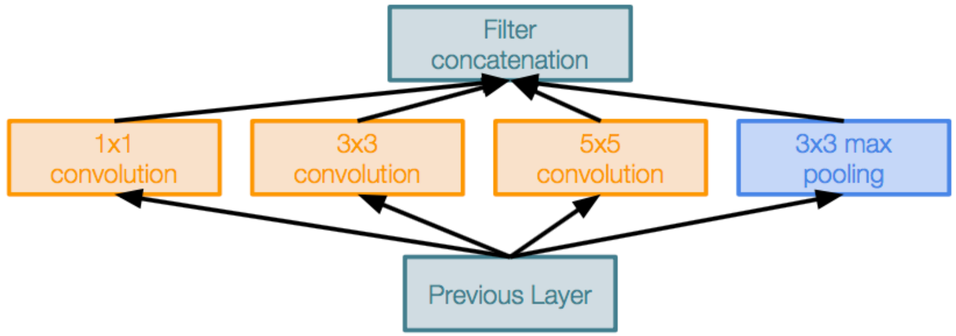
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GoogleLeNet: Inception Module

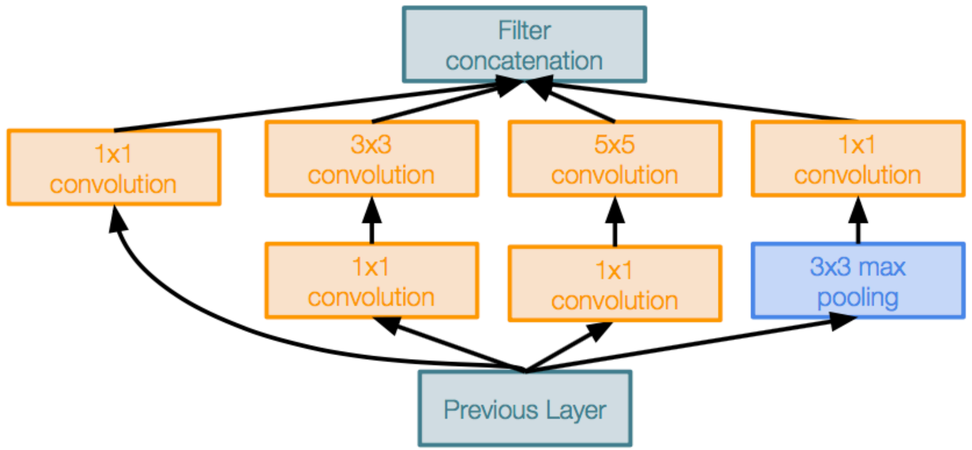
[Szegedy et al., 2014]

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

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

GoogleLeNet

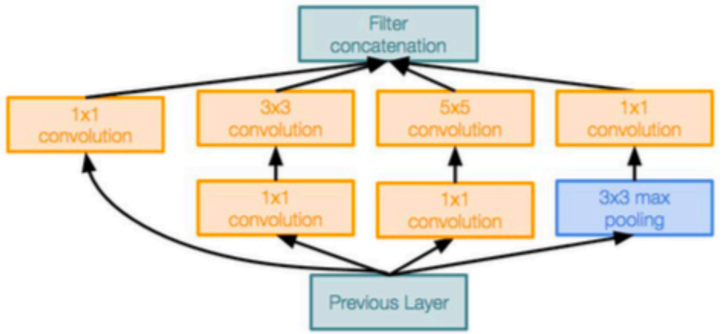
[Szegedy et al., 2014]

even deeper network with **computational efficiency**

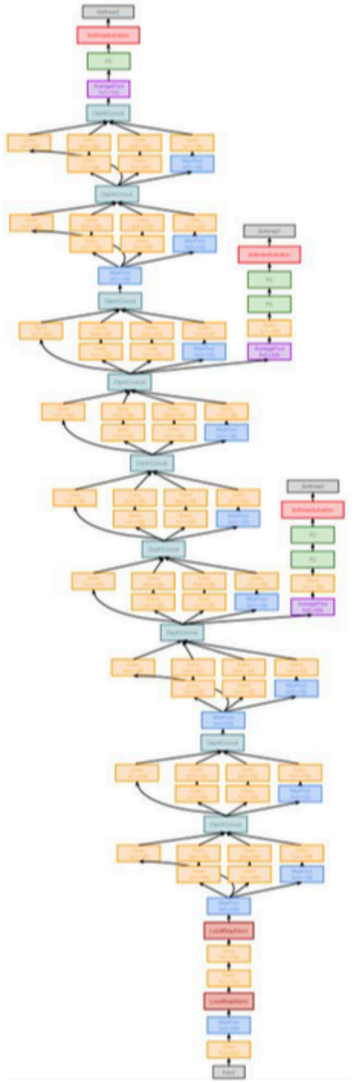
- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!

(12x less than AlexNet!)

- Better performance (@6.7 top 5 error)

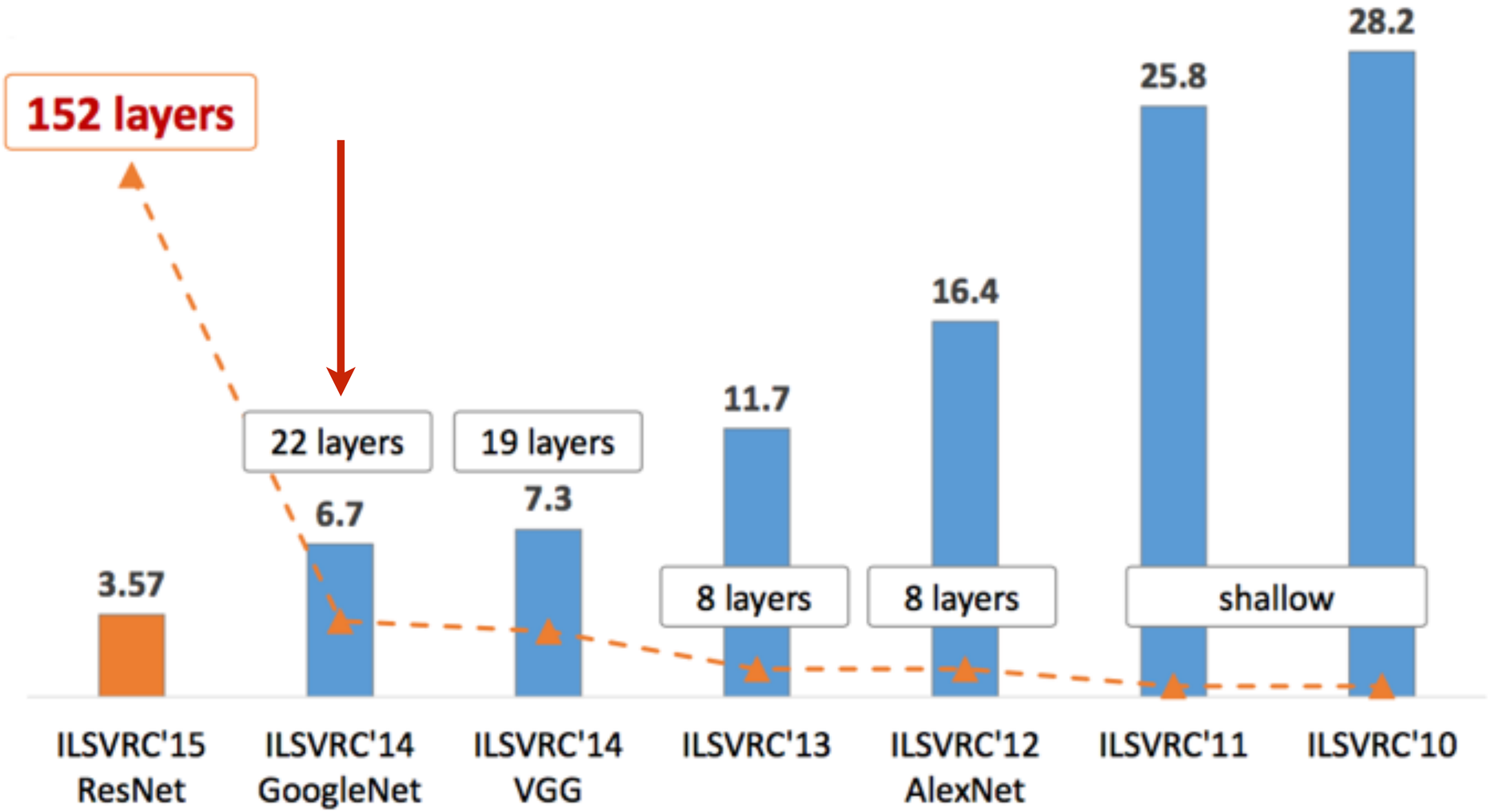


Inception module



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

ILSVRC winner 2012



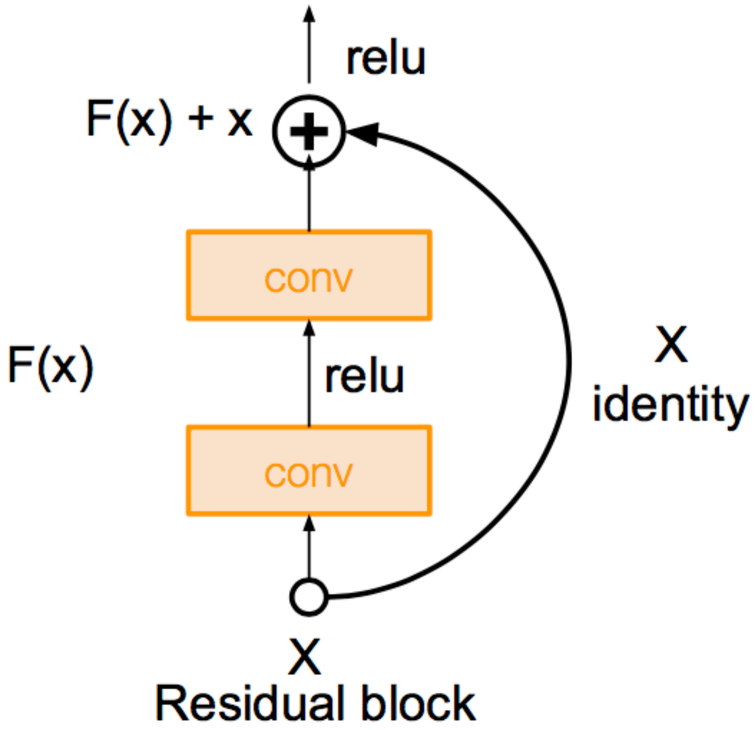
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

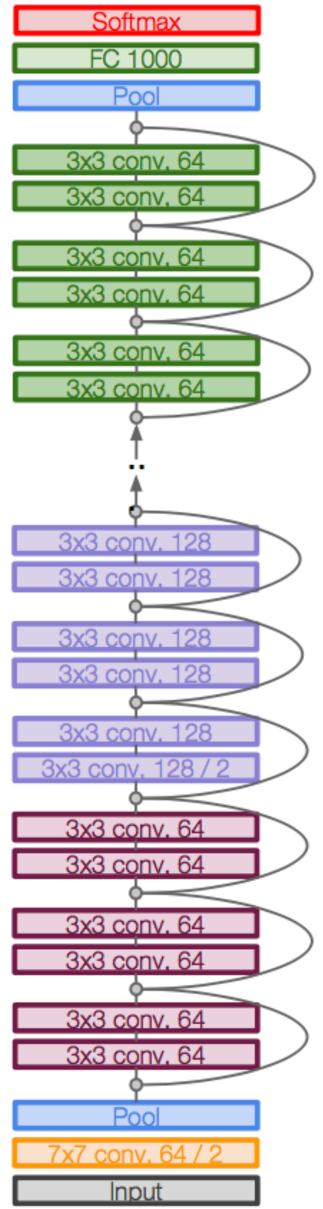
ResNet

[He et al., 2015]

even deeper — **152 layers**!

using residual connections



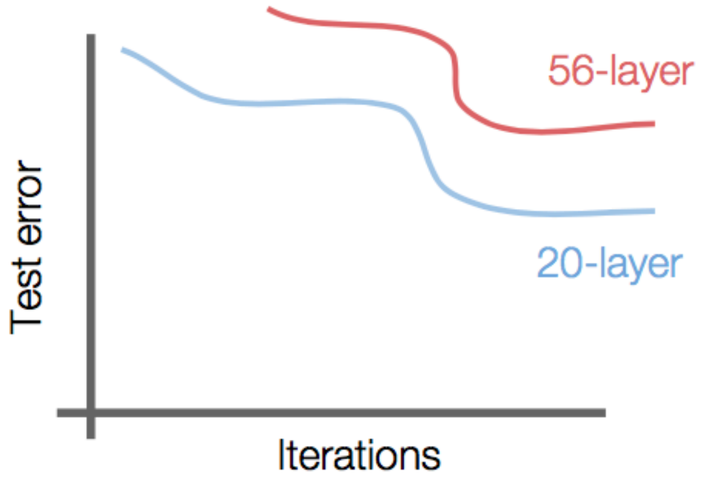


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ResNet: Motivation

[He et al., 2015]

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

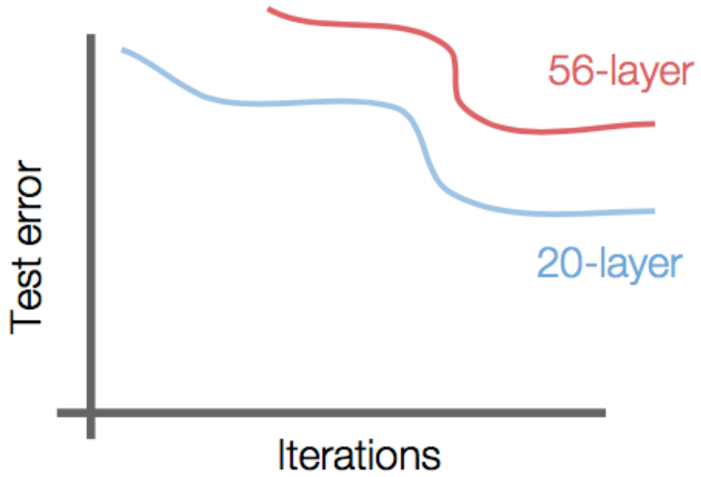


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ResNet: Motivation

[He et al., 2015]

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



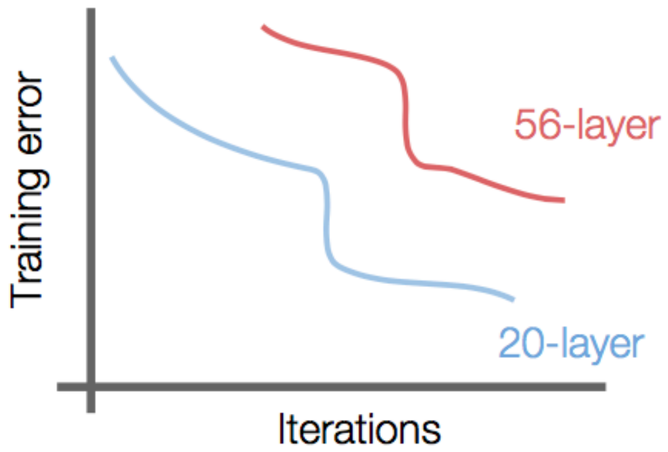
Whats the **problem**?

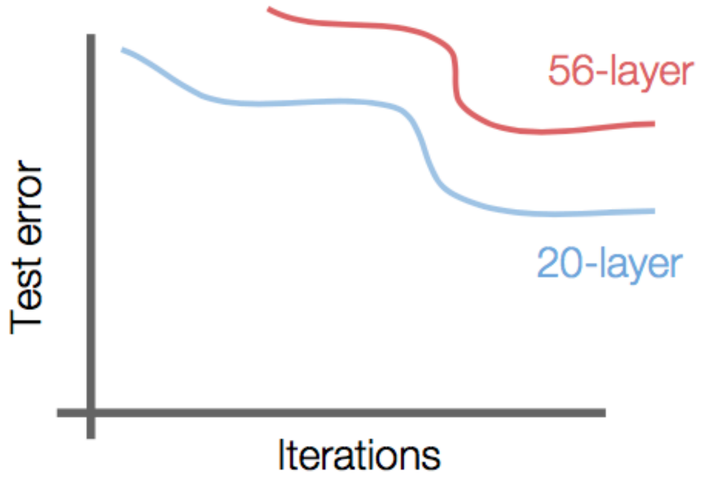
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

ResNet: Motivation

[He et al., 2015]

What happens when we continue to stack ing 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

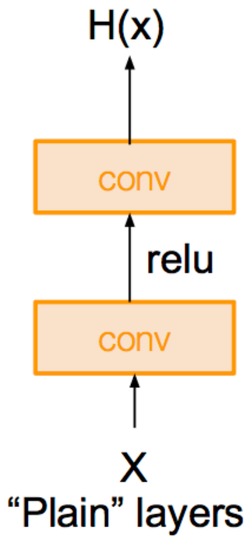
ResNet

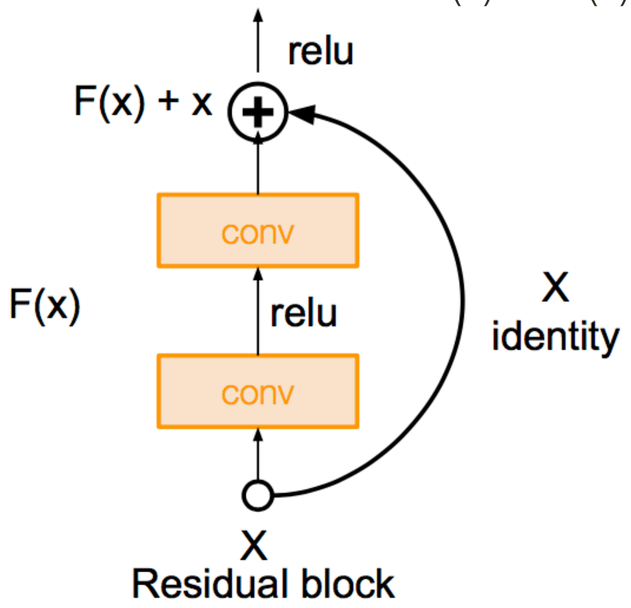
[He et al., 2015]

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





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

ResNet

[He et al., 2015]

Full details

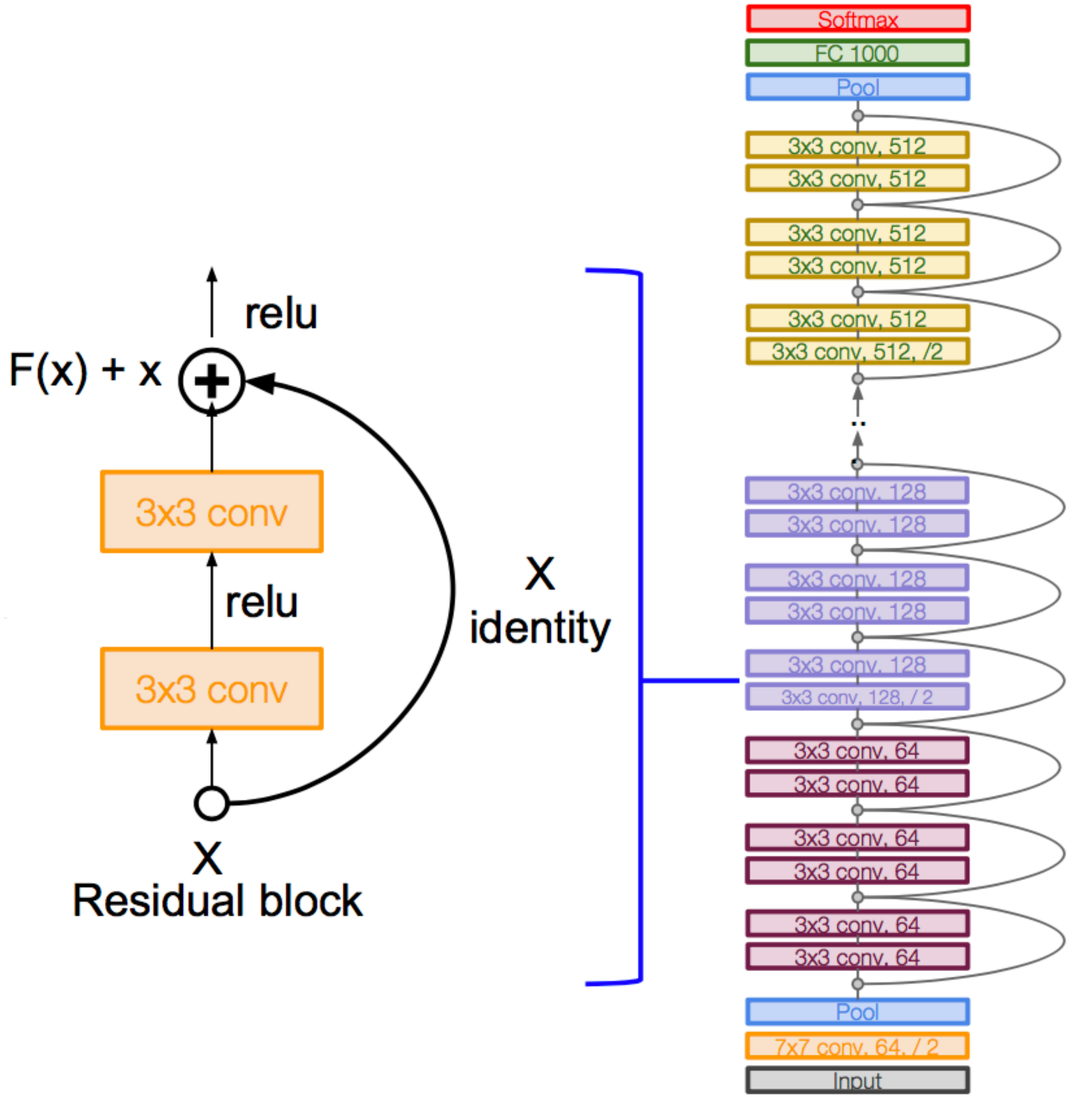
— Stacked **residual blocks**

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

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

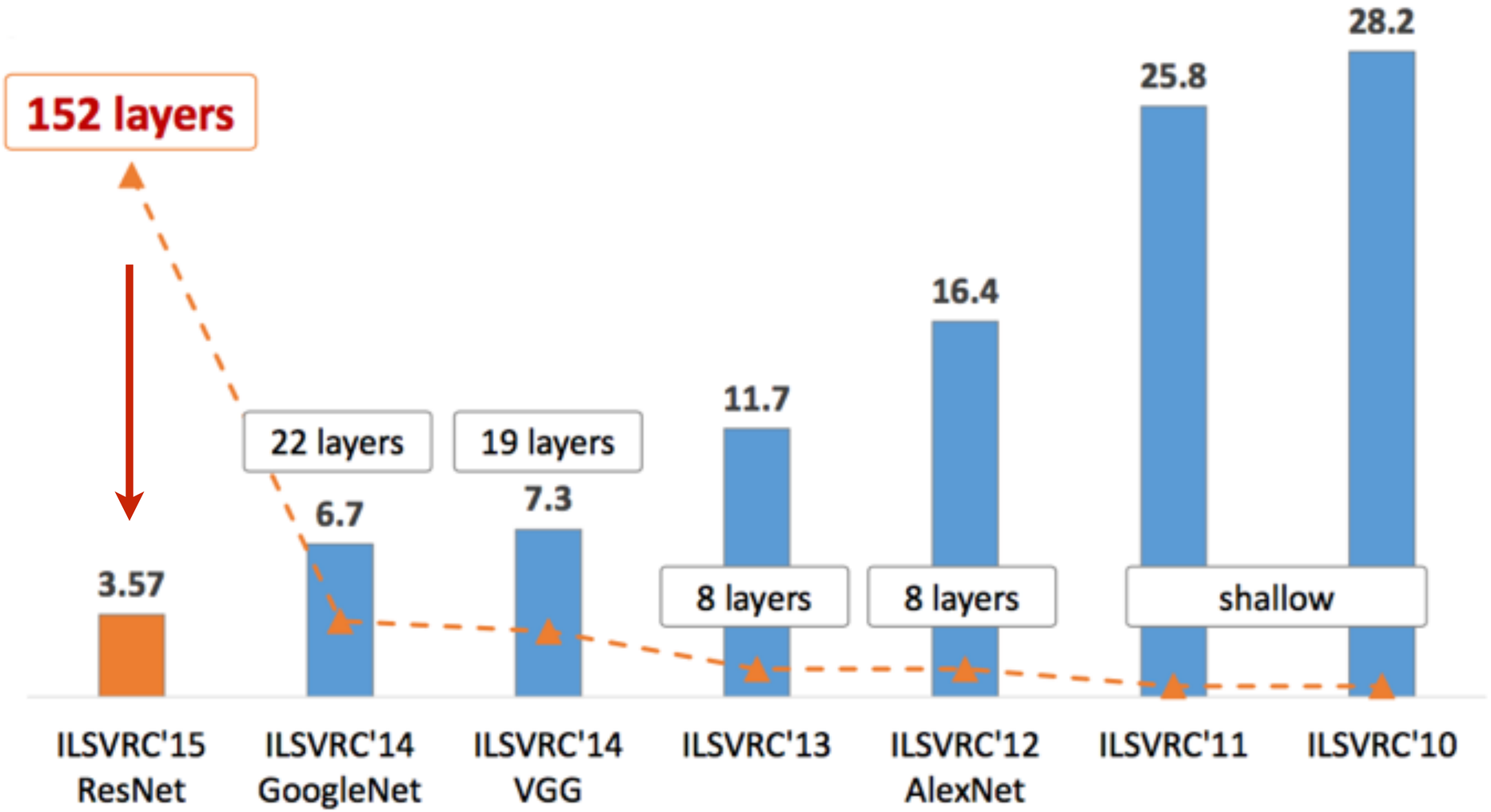
— Additional convolutional layer in the beginning

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



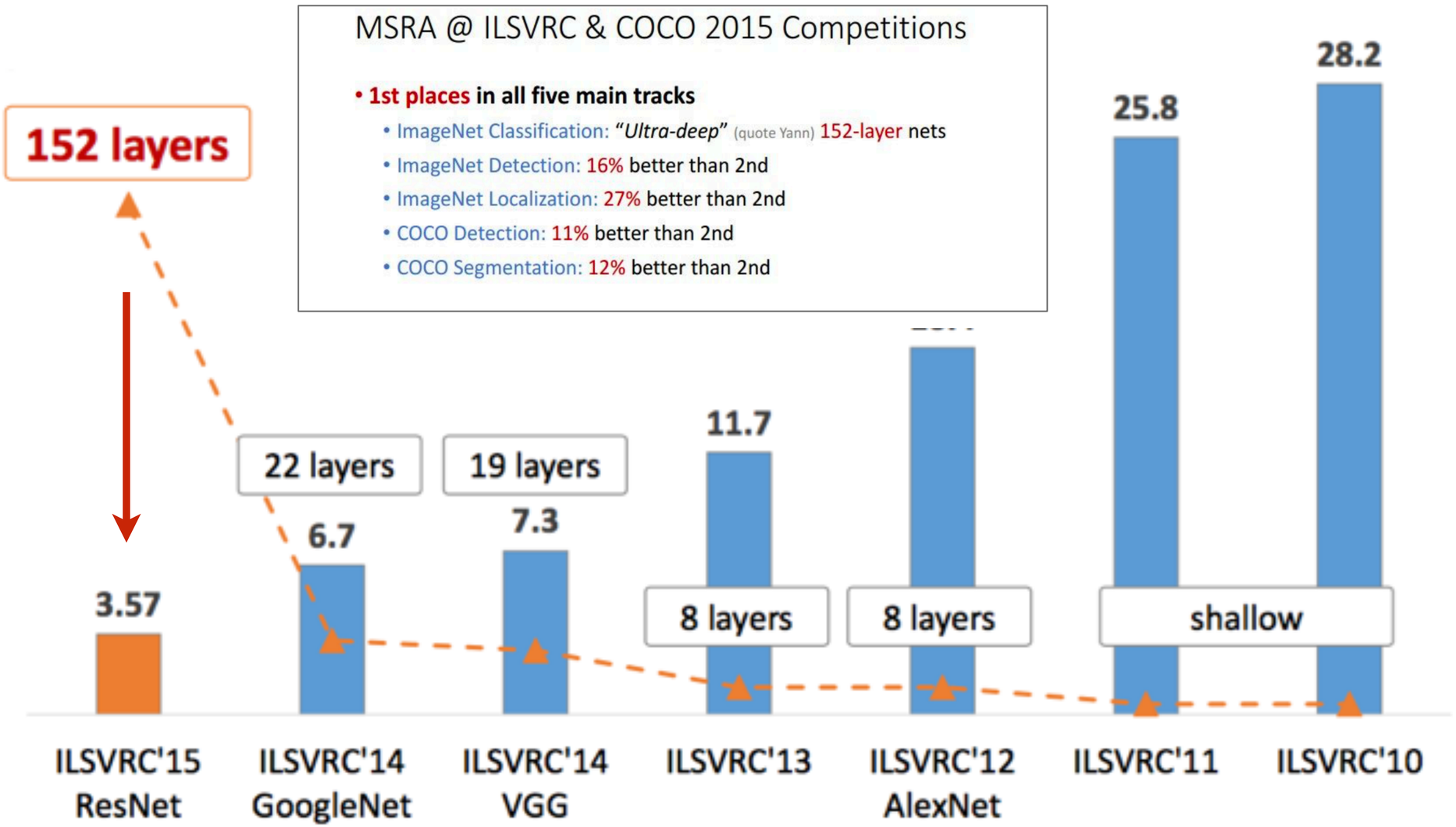
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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**

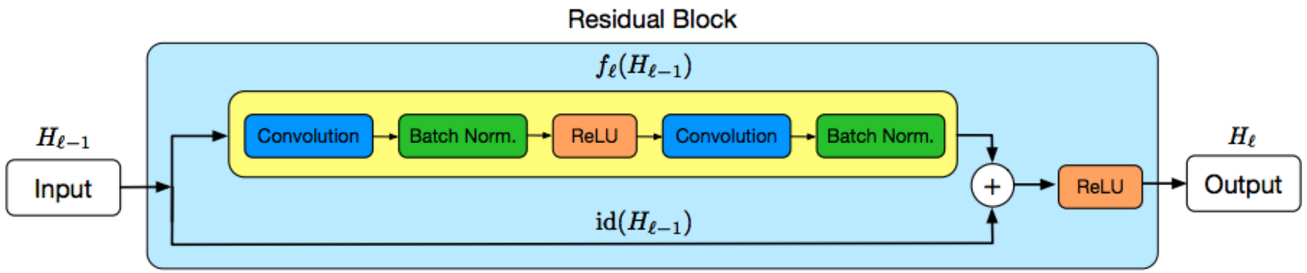
Regularization: Stochastic Depth

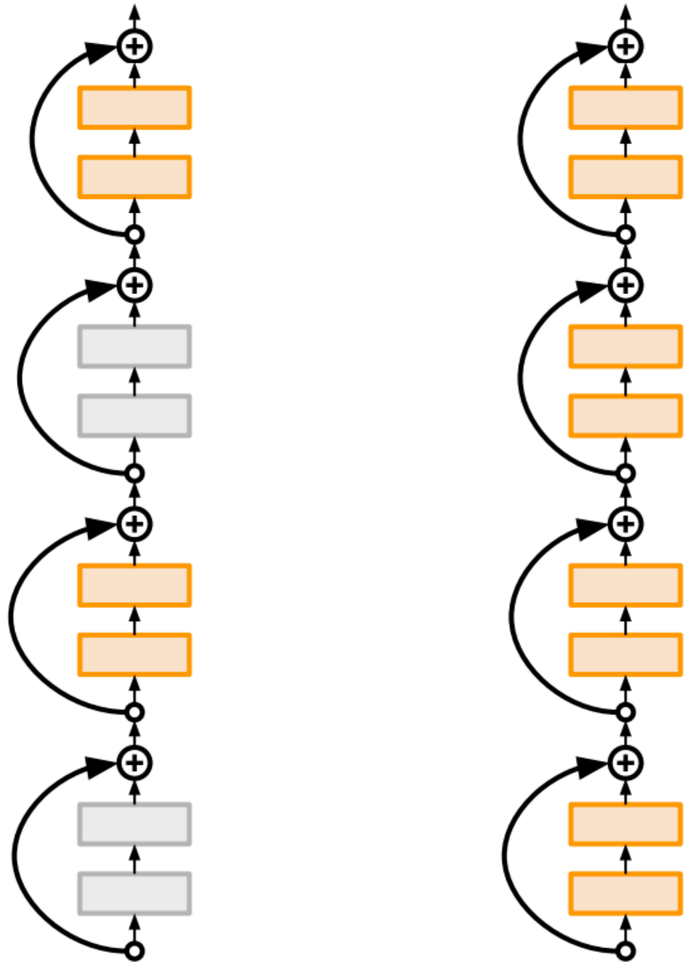
[Huang et al., ECCV 2016]

Effectively “dropout” but for layers

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

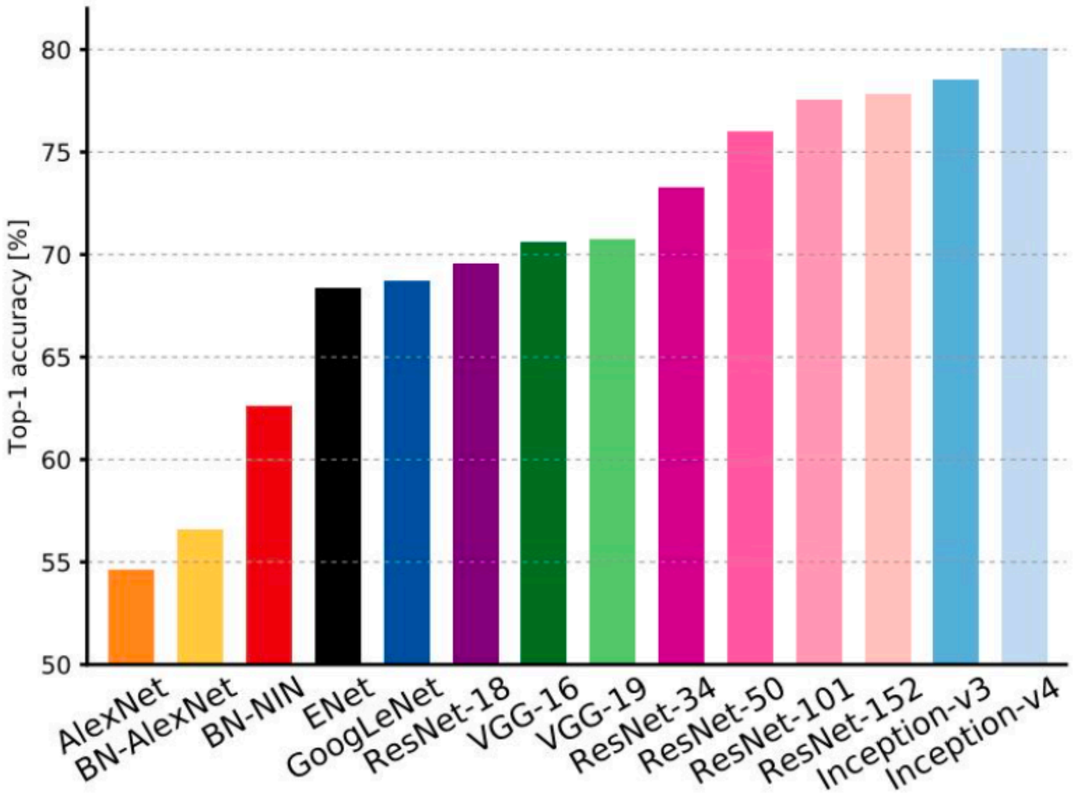
Effectively trains a collection of neural networks

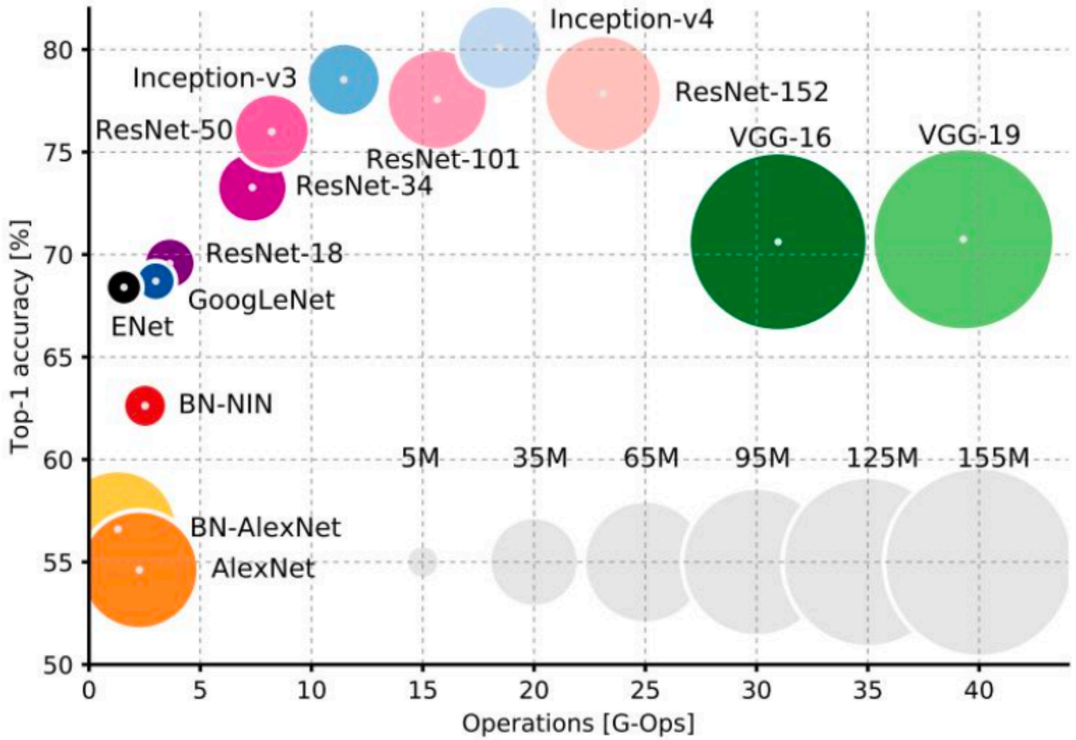




* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Comparing **Complexity**





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

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

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

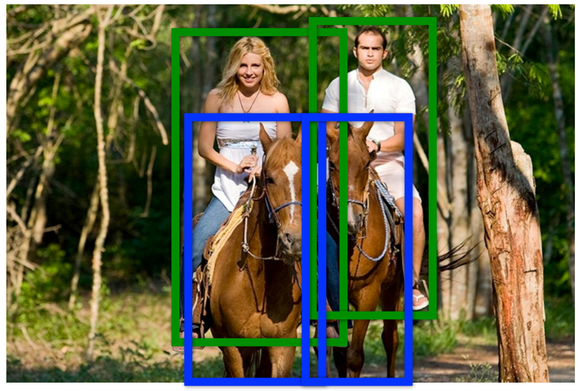
Categorization

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)

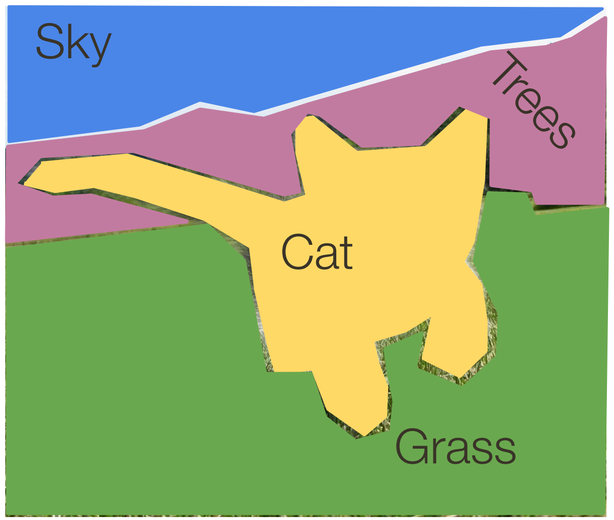
Segmentation

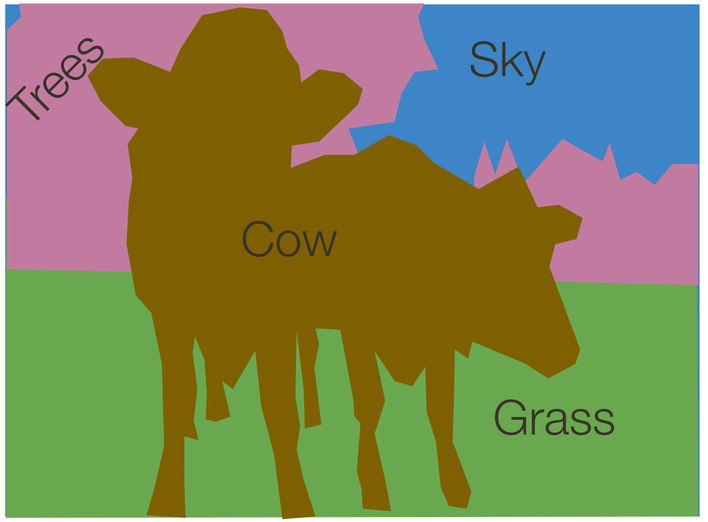
Horse
Person

COCO
Common Objects in Context

Semantic **Segmentation**

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



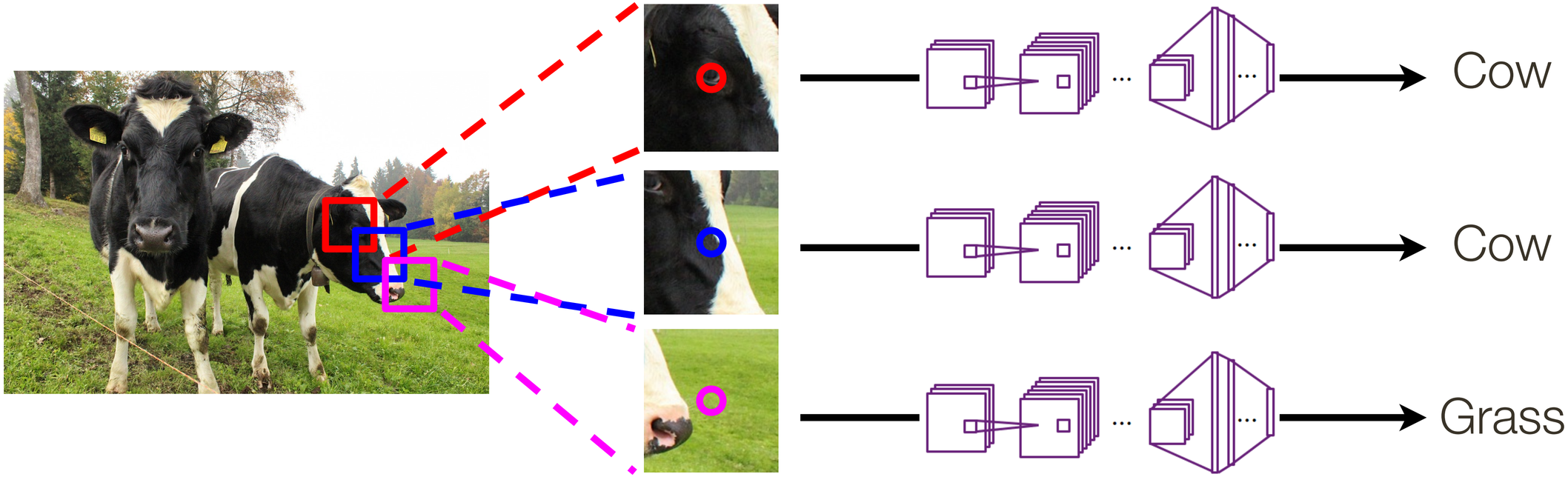


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Sliding Window

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

Extract **patches** **Classify center pixel** with CNN

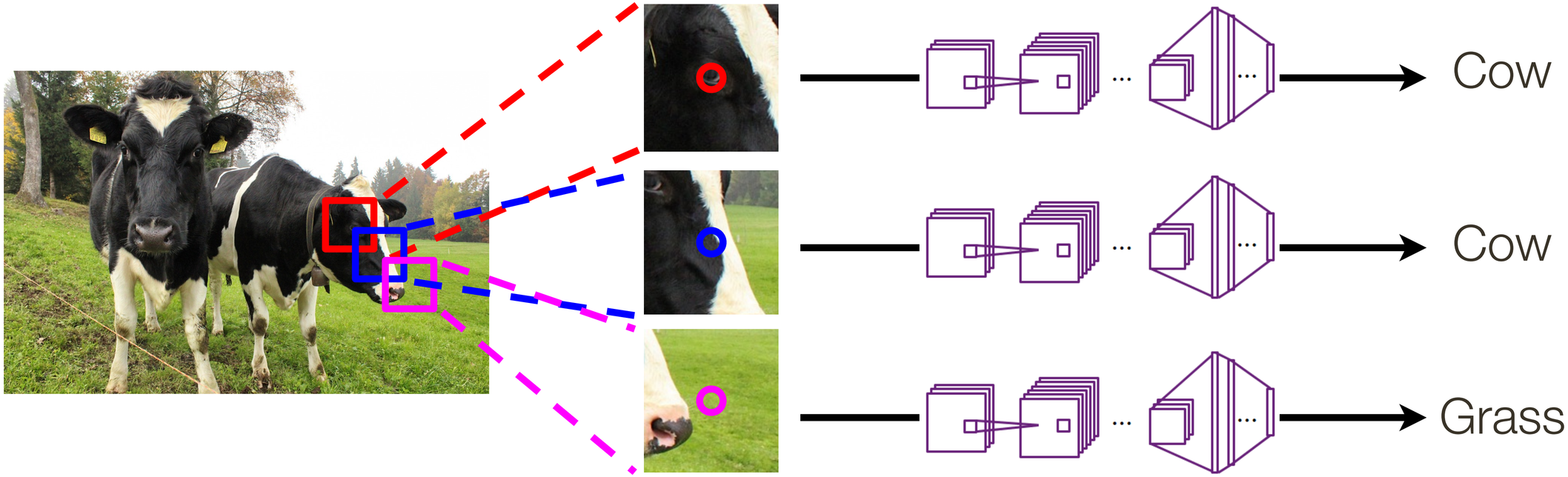


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Sliding Window

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

Extract **patches** **Classify center pixel** with CNN

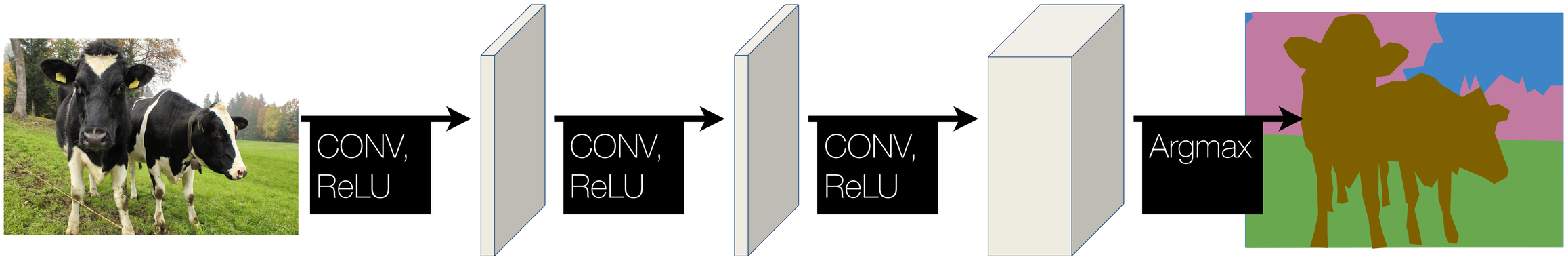


Problem: VERY inefficient, no reuse of computations for overlapping patches

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

Semantic **Segmentation:** Fully Convolutional CNNs

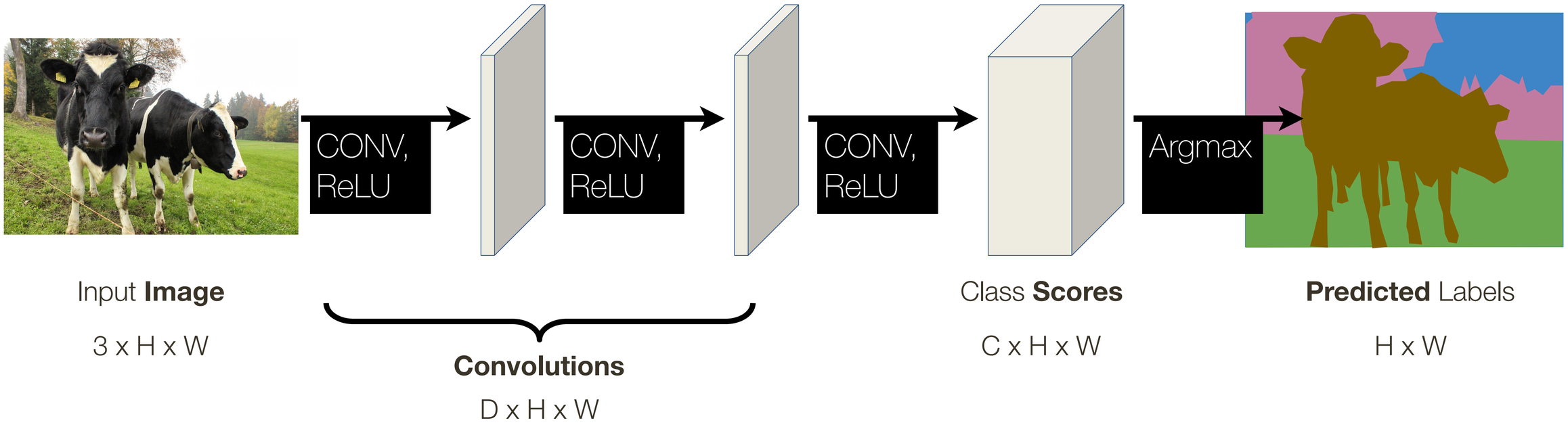
Design a network as a number of convolutional layers to make predictions for all pixels at once!



* 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

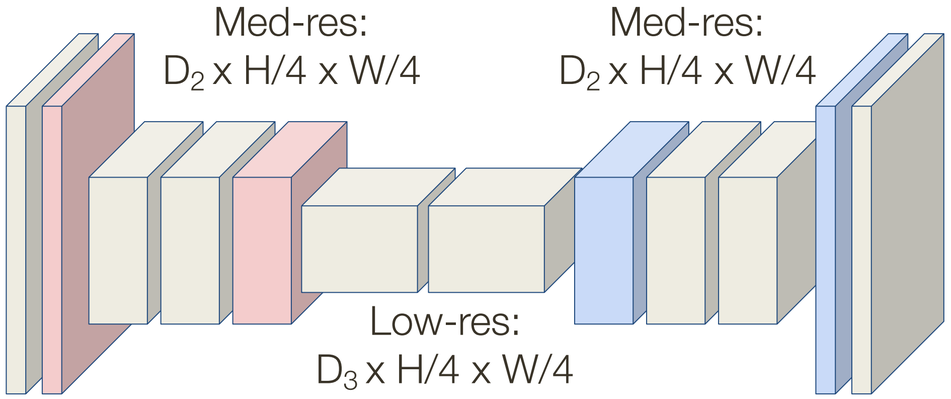
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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]

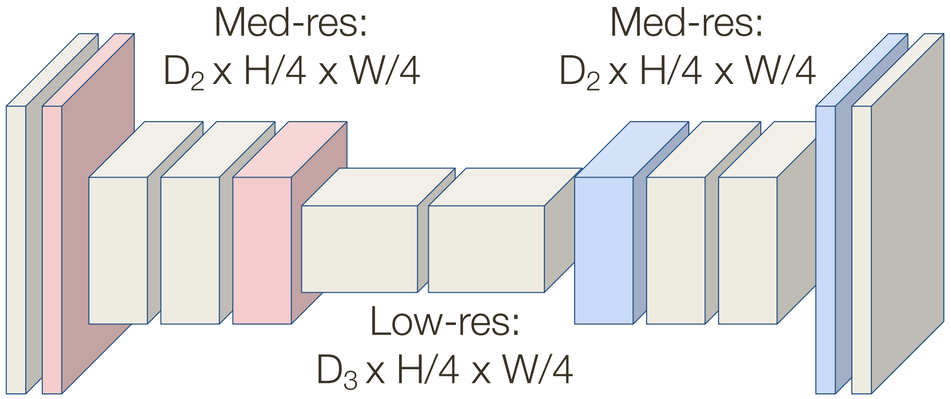
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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

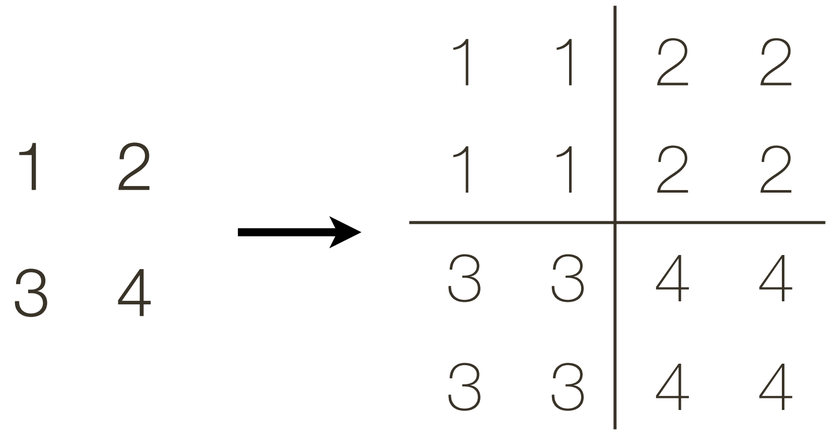
Upsampling = ???

[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



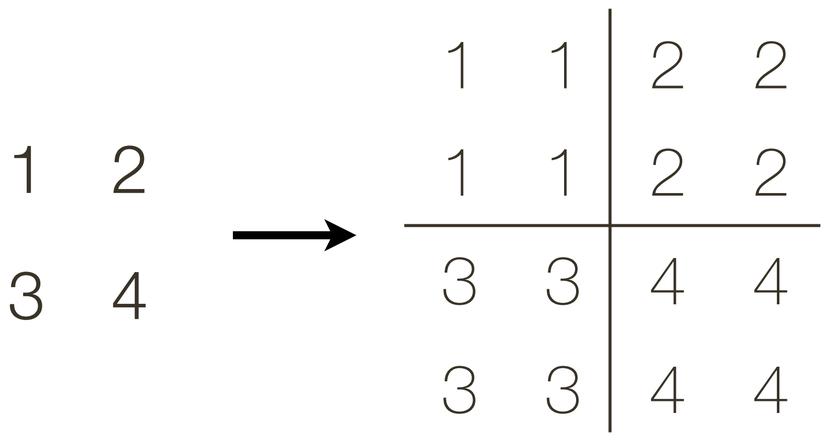
Input: 2 x 2

Output: 4 x 4

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

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

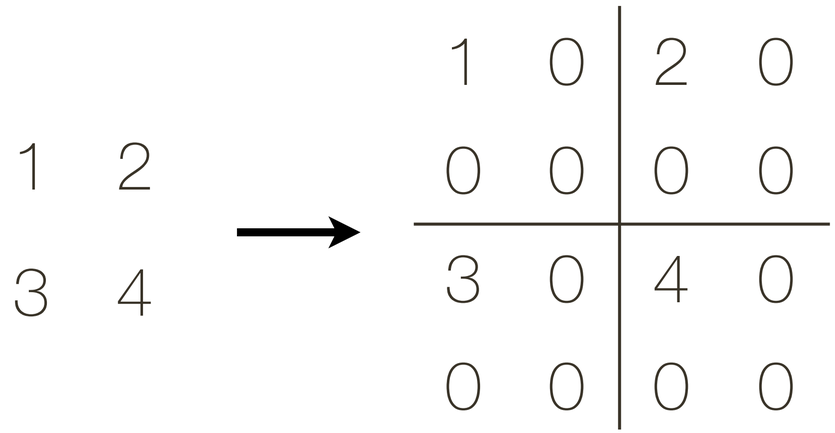
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



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

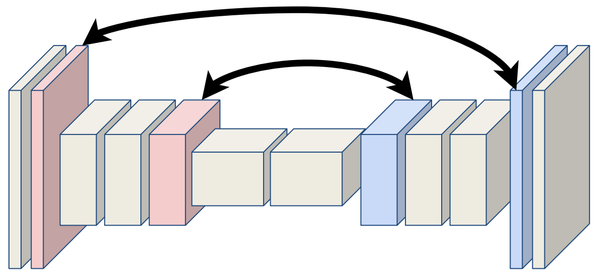
Max Unpooling
Use positions from pooling layer

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

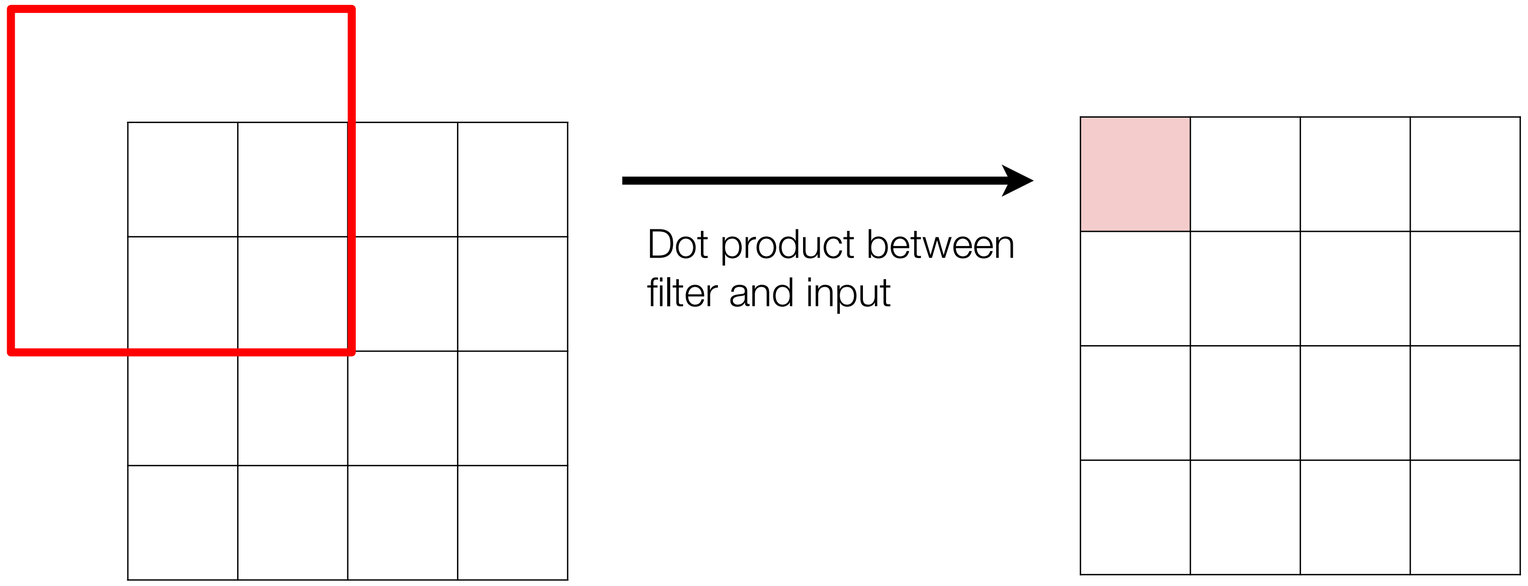


Corresponding pairs of downsampling and upsampling layers

* 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



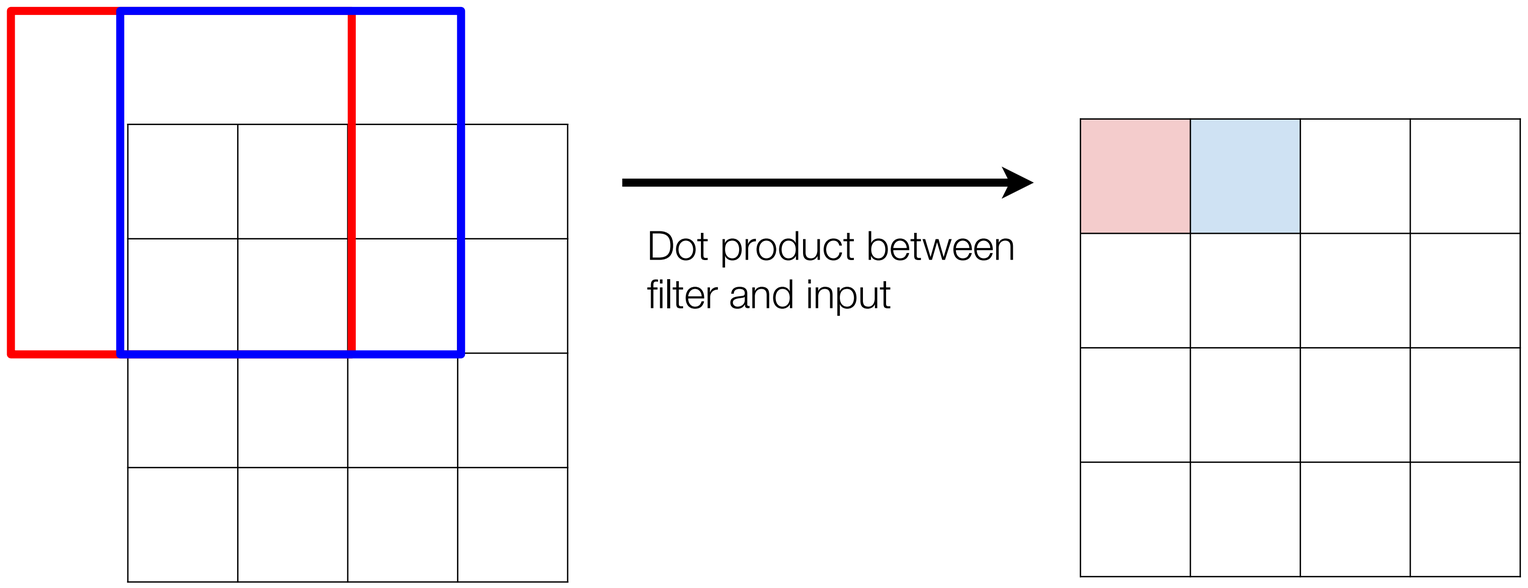
Input: 4 x 4

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 1 pad 1



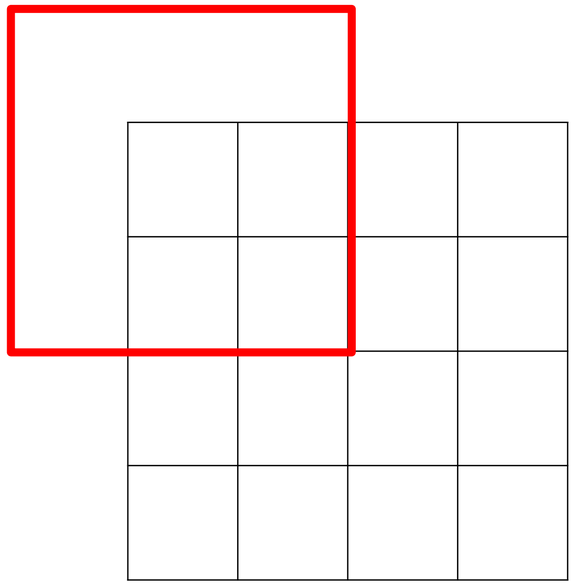
Input: 4 x 4

Output: 4 x 4

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

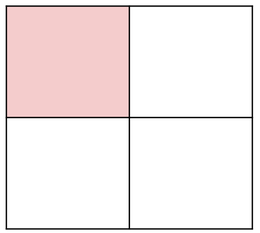
In-network **Up Sampling:** Transpose Convolution

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



Input: 4 x 4

Dot product between
filter and input

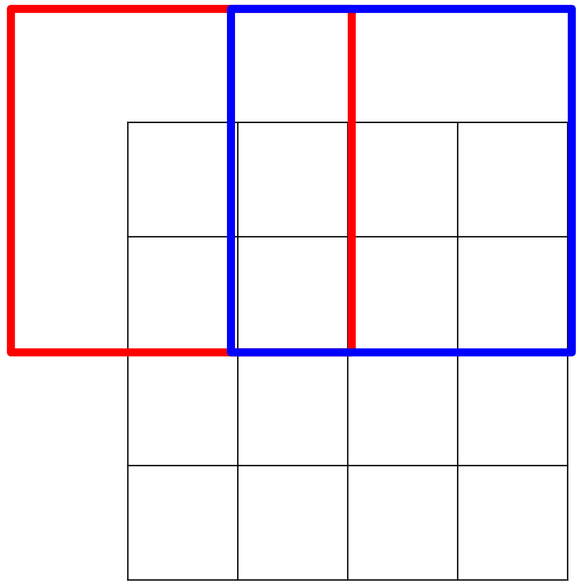


Output: 2 x 2

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

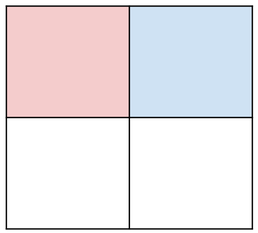
In-network **Up Sampling:** Transpose Convolution

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



Input: 4 x 4

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

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

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

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

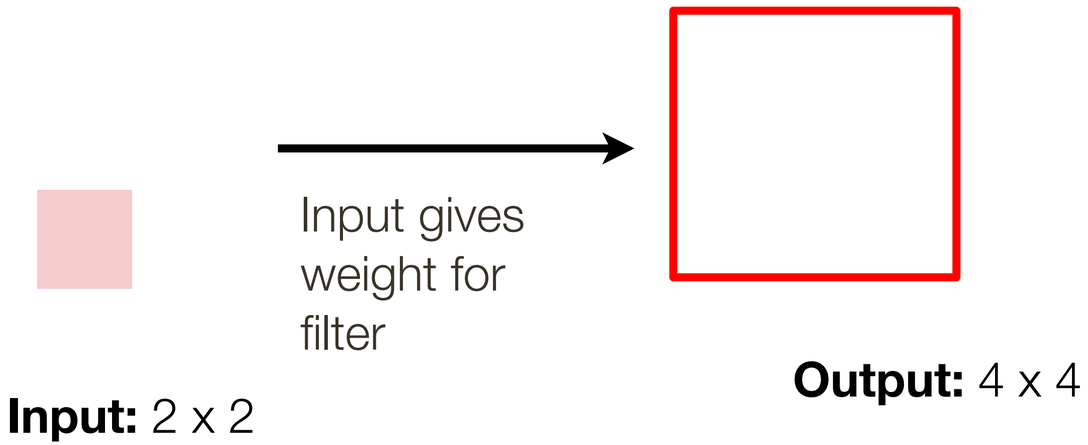
Input: 2 x 2

Output: 4 x 4

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

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

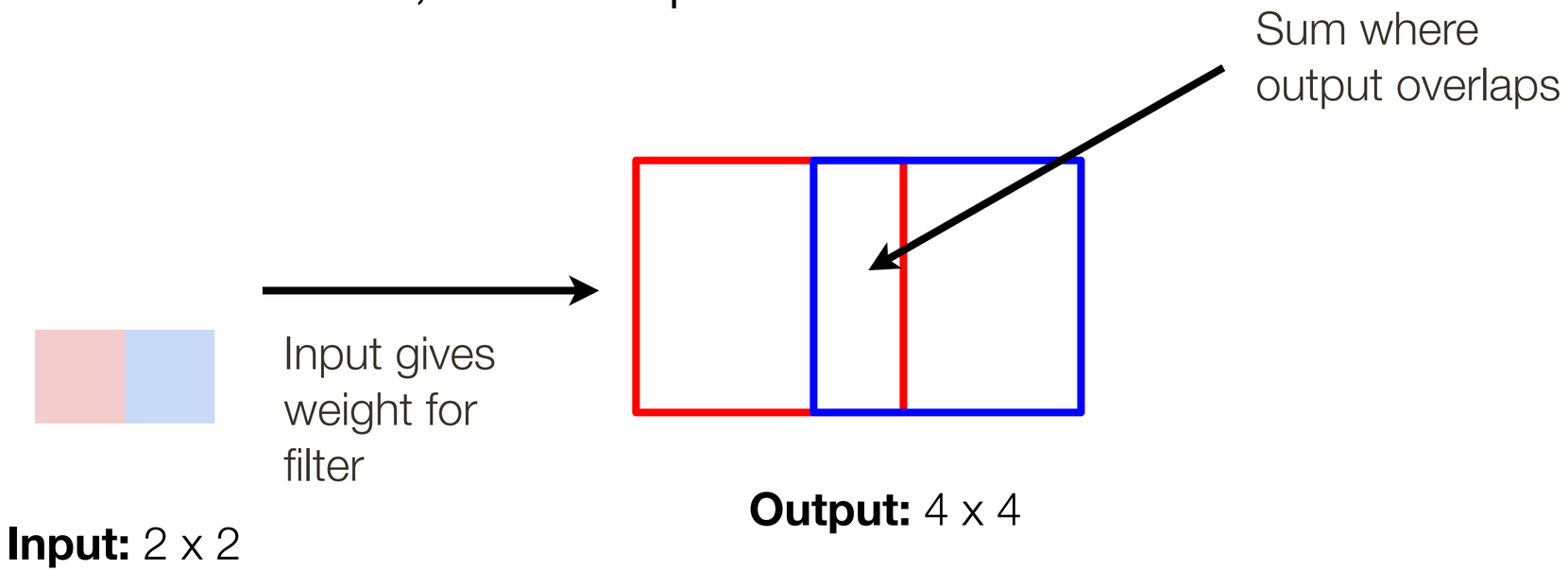
3 x 3 **transpose** convolution, stride 2 pad 1



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

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

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

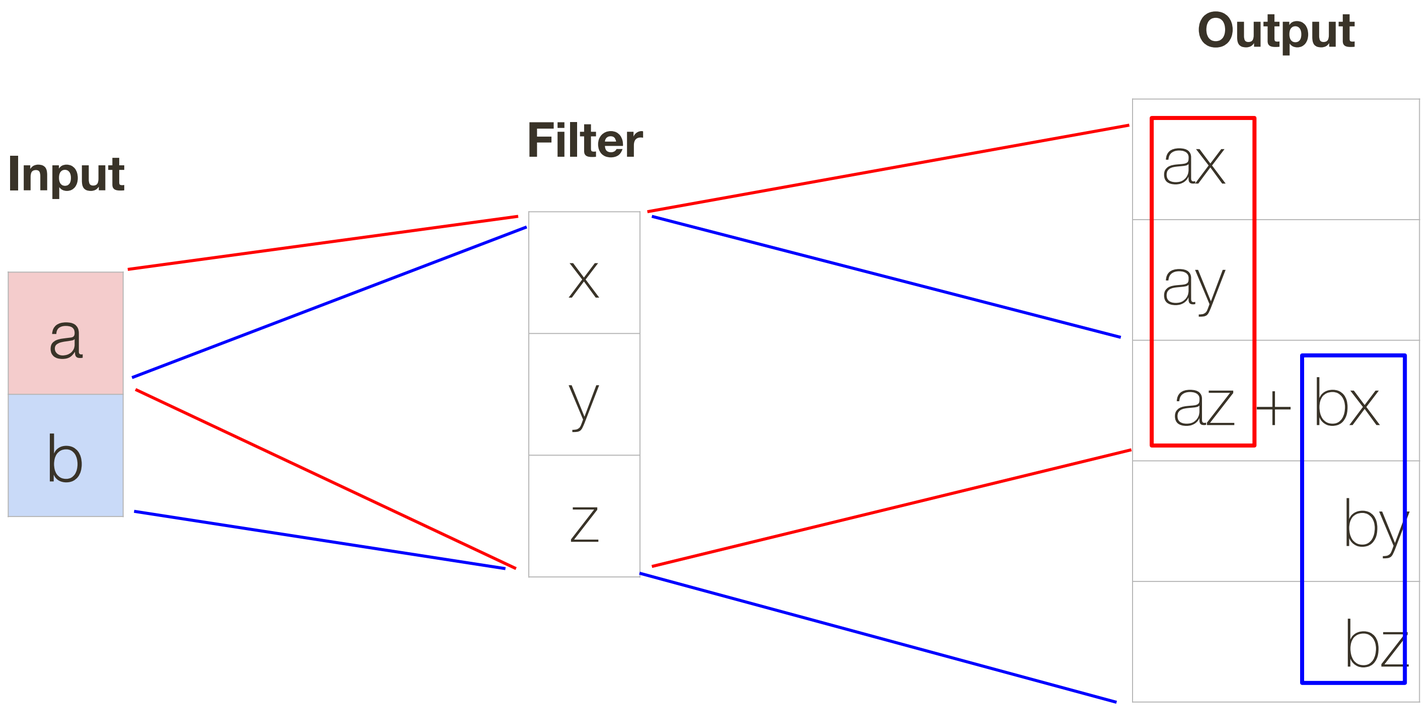


Filter moves 2 pixels in the **output** for every one pixel in the **input**

Stride gives ratio in movement in output vs input

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

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**

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

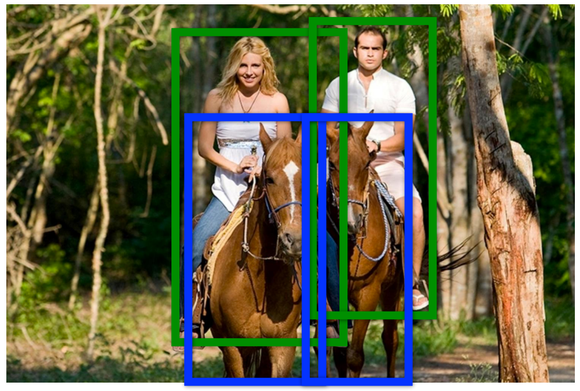
Categorization

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

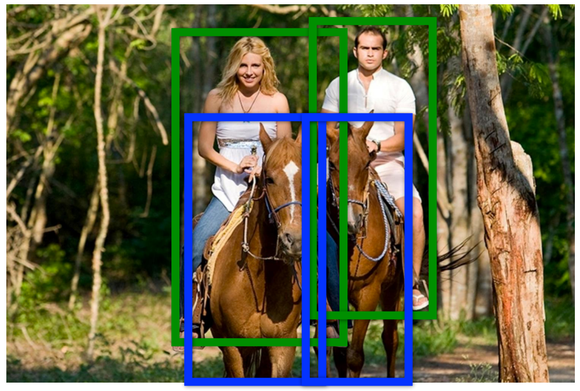
COCO
Common Objects in Context

Instance Segmentation

Horse1
Horse2
Person1
Person2

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

Detection

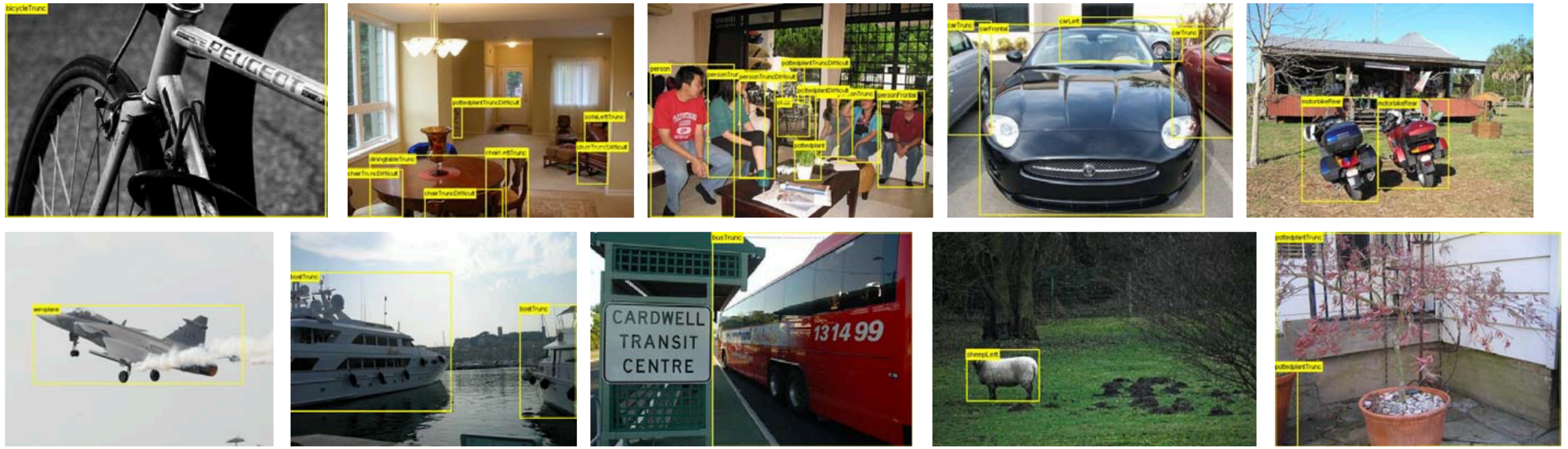


Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

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

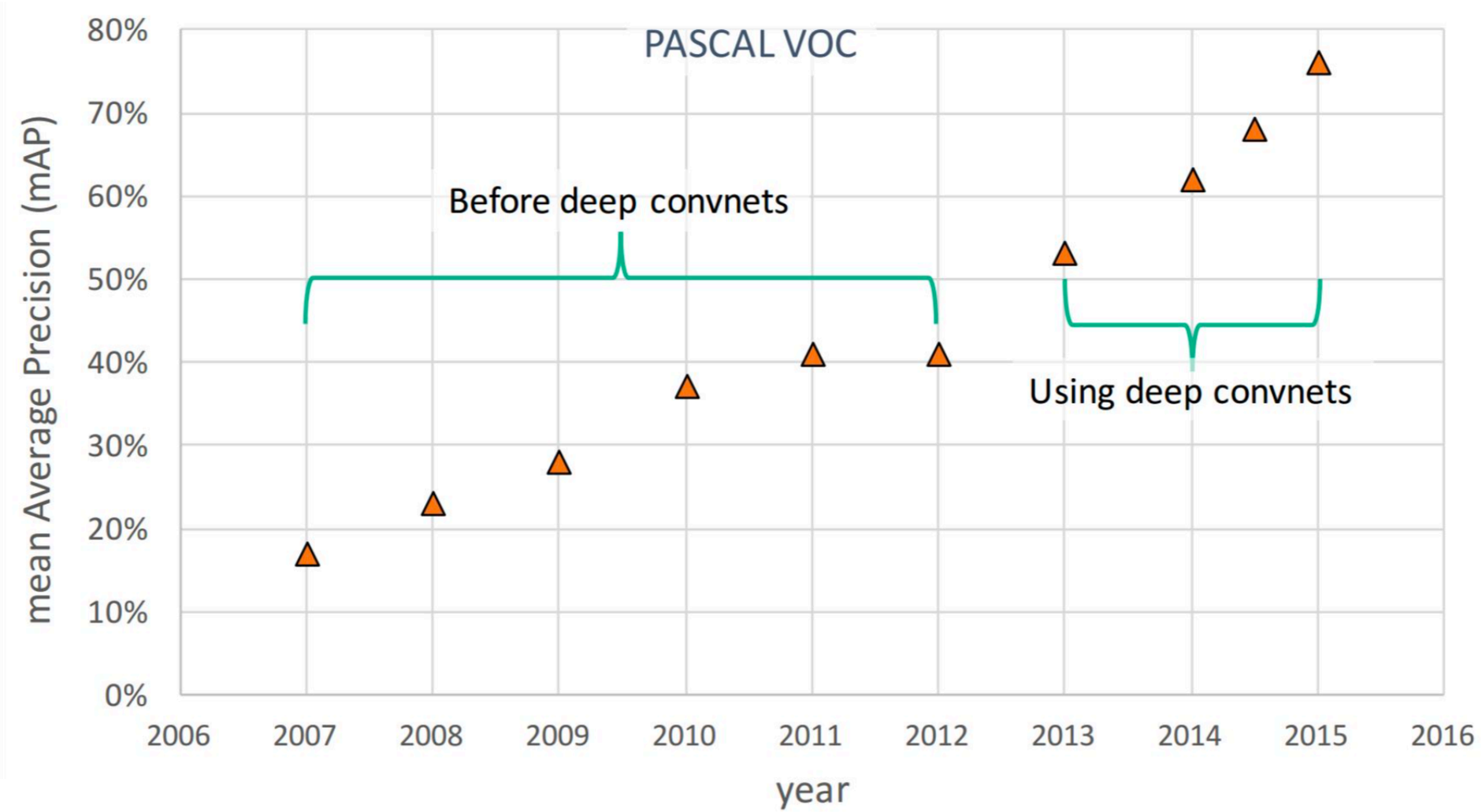
Real images downloaded from flickr, not filtered for “quality”

* slide from Andrew Zisserman

Datasets: COCO

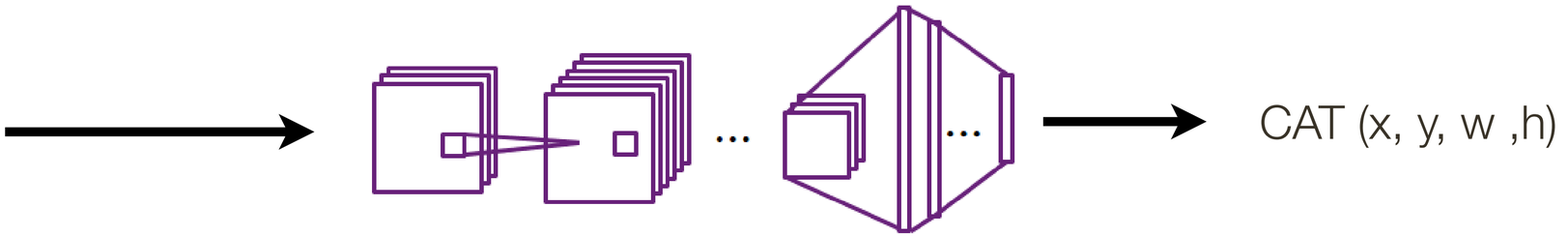
- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

Object **Detection**



* plot from Ross Girshick, 2015

Object **Detection** as Regression Problem



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

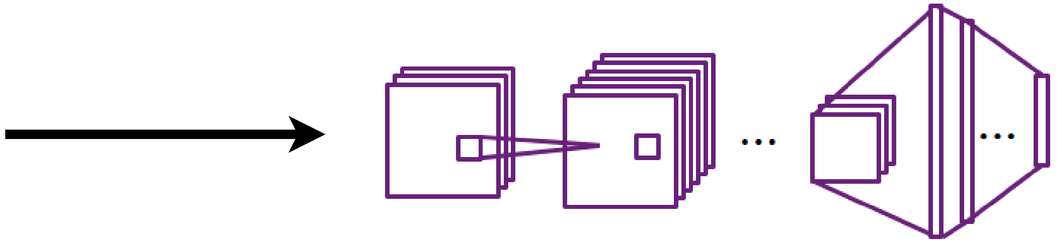
Object **Detection** as Regression Problem

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



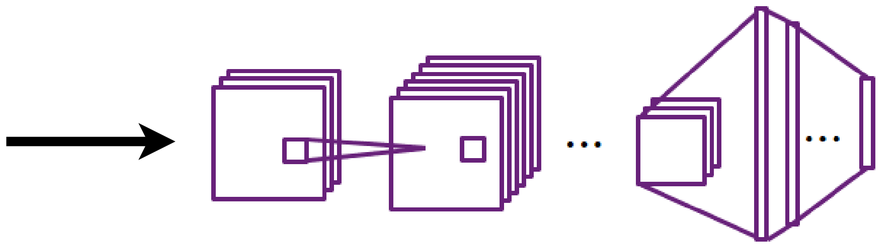
CAT (x, y, w ,h)

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)
…

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

Object **Detection** as Classification Problem

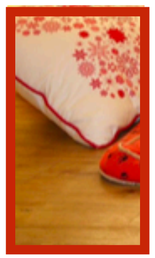


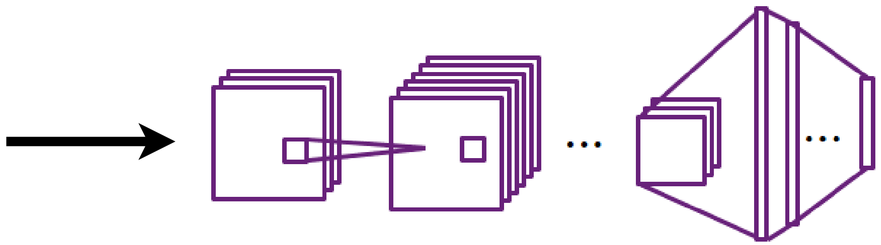
Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Background	**Yes**
…	…

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

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

Object **Detection** as Classification Problem



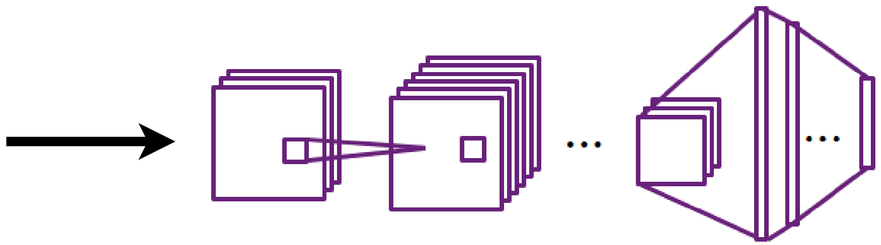


Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Background	**Yes**
…	…

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

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

Object **Detection** as Classification Problem

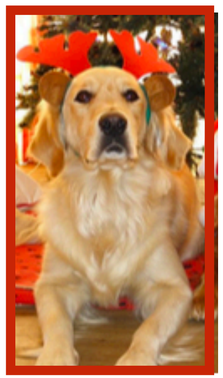


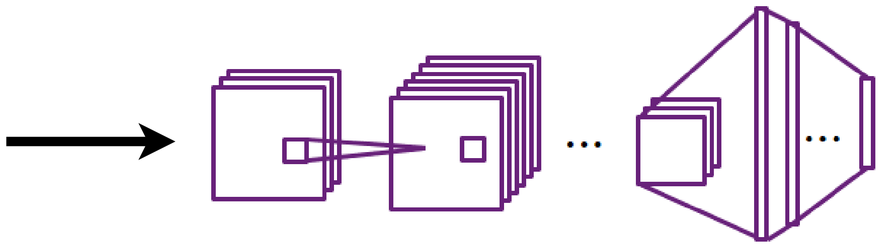
Category	Prediction
Dog	**Yes**
Cat	No
Couch	No
Flowers	No
Background	No
…	…

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



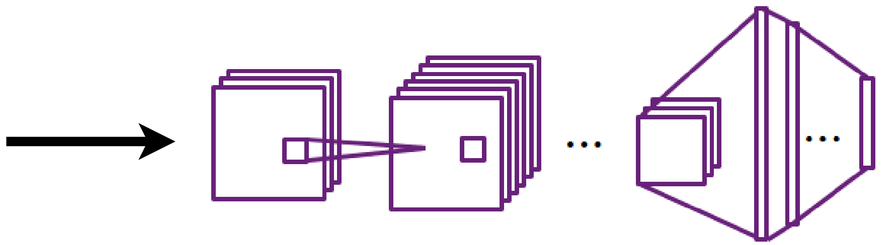


Category	Prediction
Dog	**Yes**
Cat	No
Couch	No
Flowers	No
Background	No
…	…

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



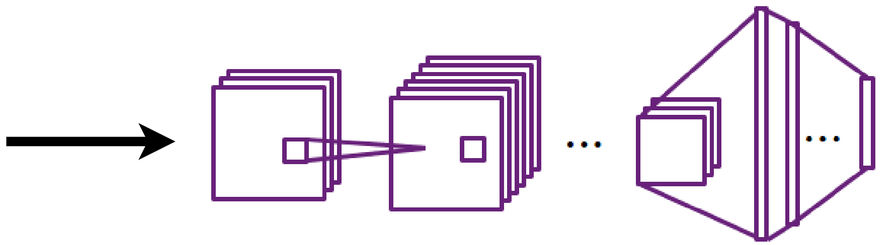
Category	Prediction
Dog	No
Cat	**Yes**
Couch	No
Flowers	No
Background	No
…	…

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

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

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

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

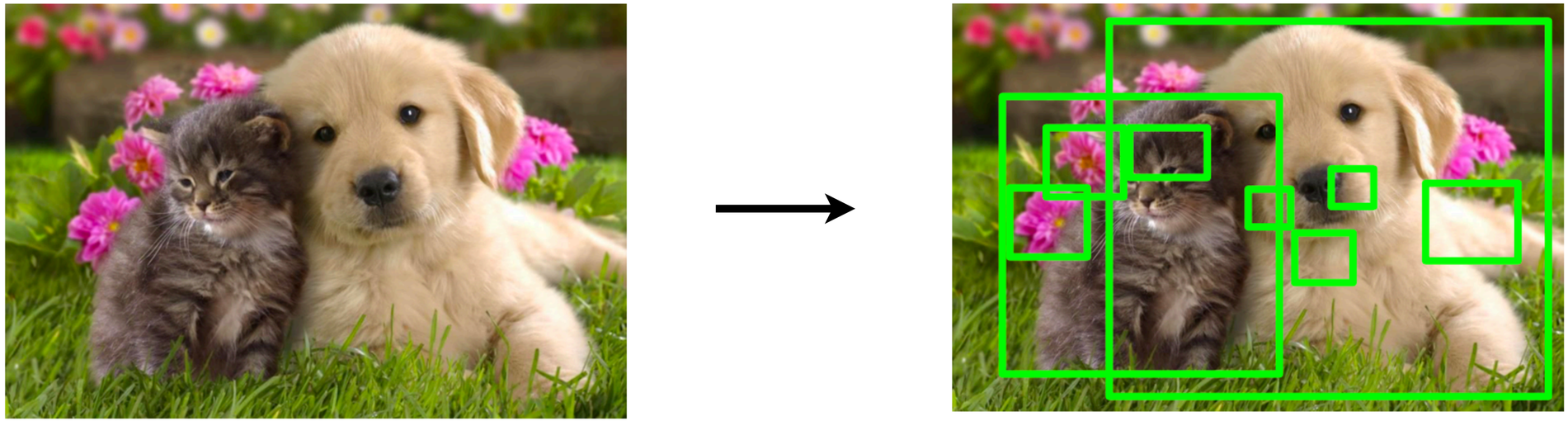
Region Proposals (older idea in vision)

[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]

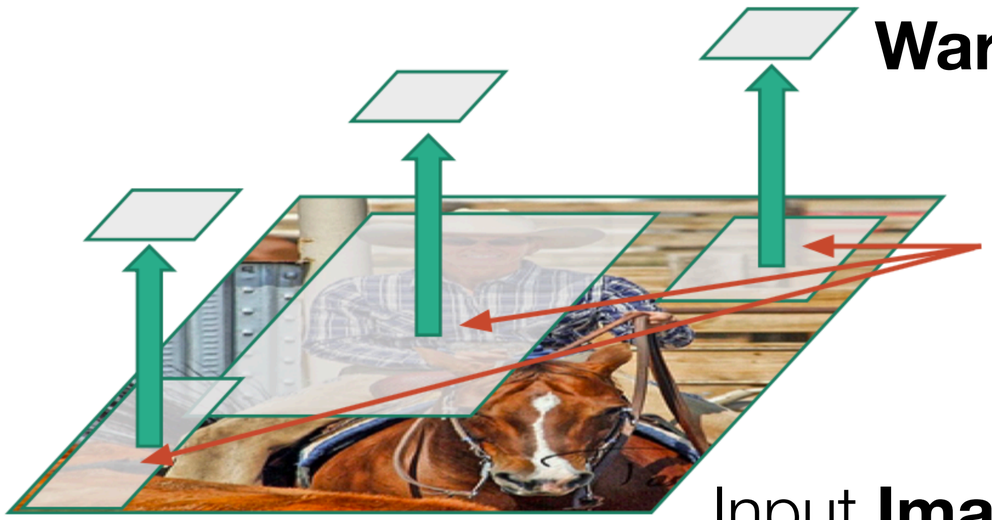
Regions of Interest from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]



Warped image regions

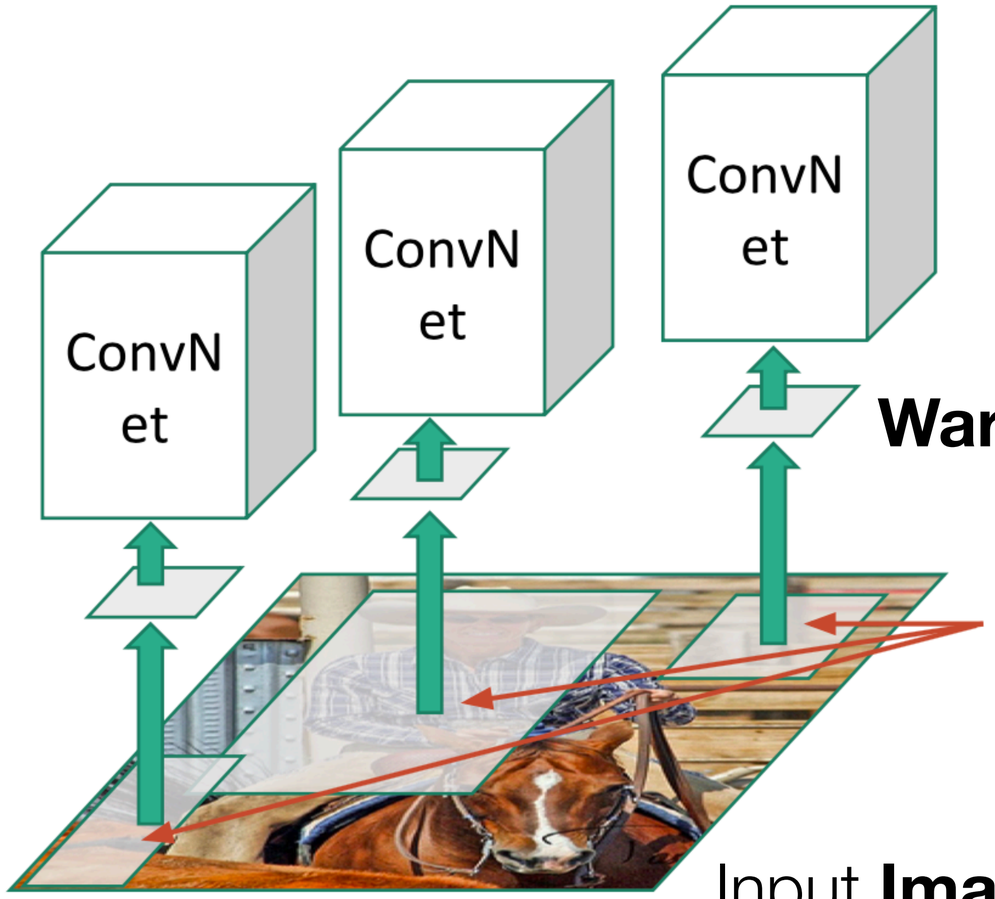
Regions of Interest from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]



Forward each region through a **CNN**

Warped image regions

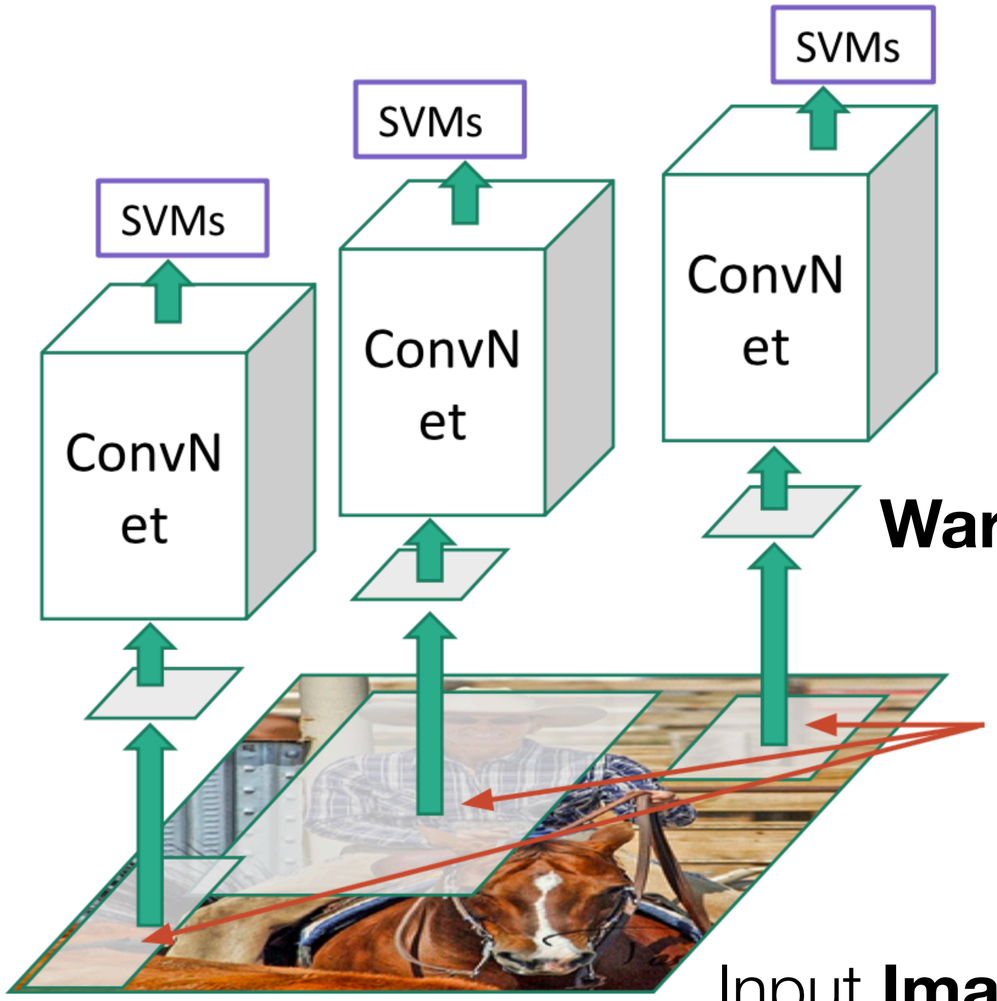
Regions of Interest from a proposal method (~2k)

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)

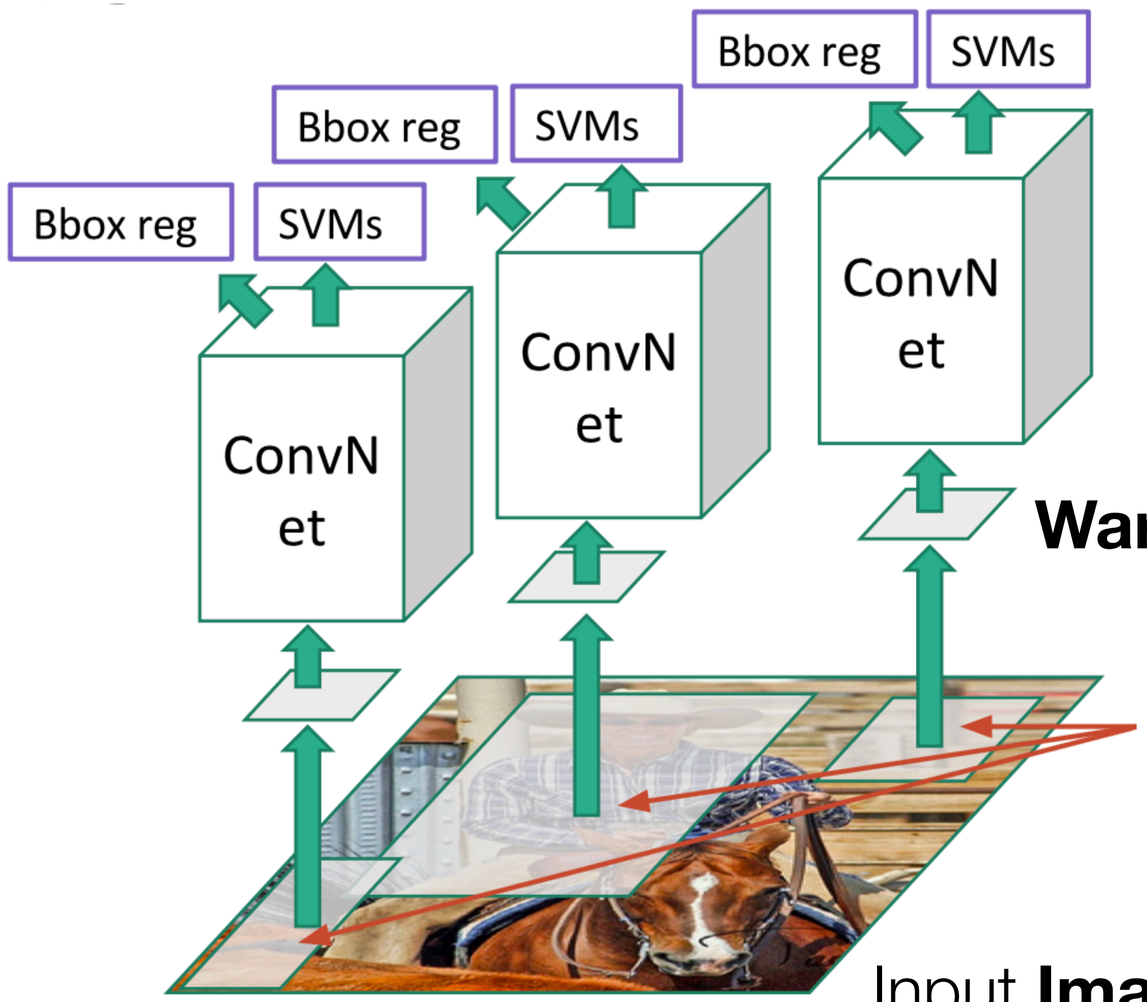
Input **Image**

* image from Ross Girshick

R-CNN

Linear Regression for bounding box offsets

[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

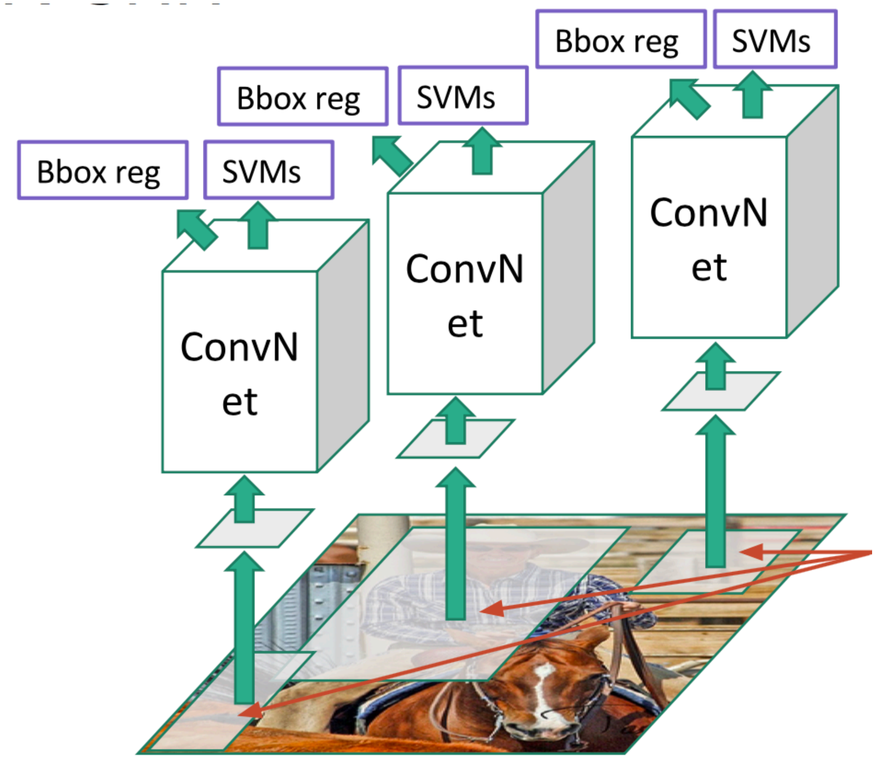
R-CNN: Training

[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)

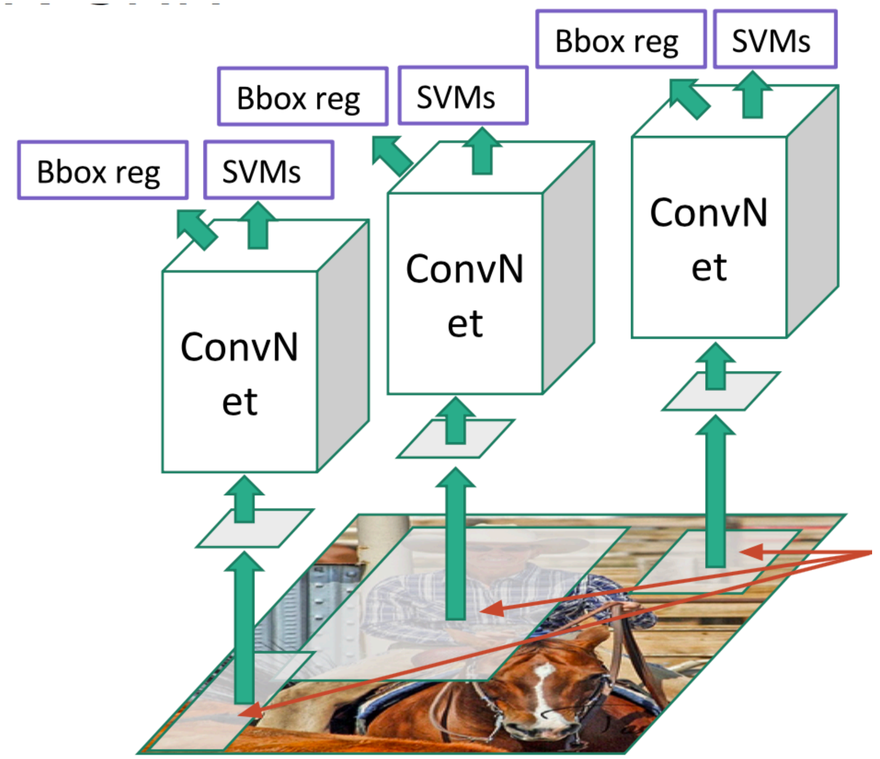
— Train post-hoc bounding-box regression (**least squares**)

Training is slow

— 84 hours and takes a lot of disk space

Inference / **Detection is slow**

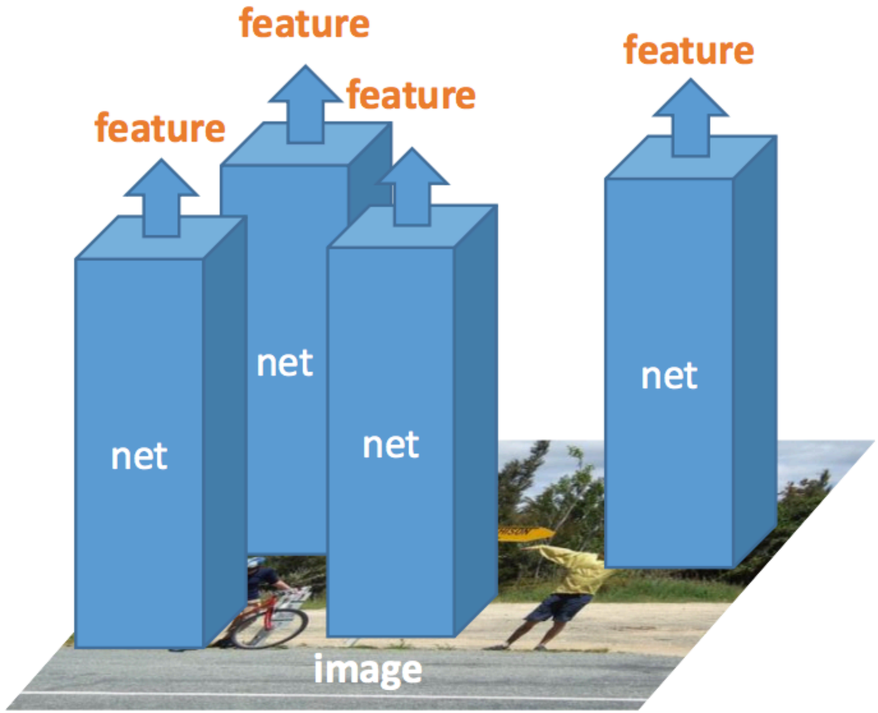
— 47 sec / image with VGG16 [Simonyan et al, ICLR 2015]



* image from Ross Girshick

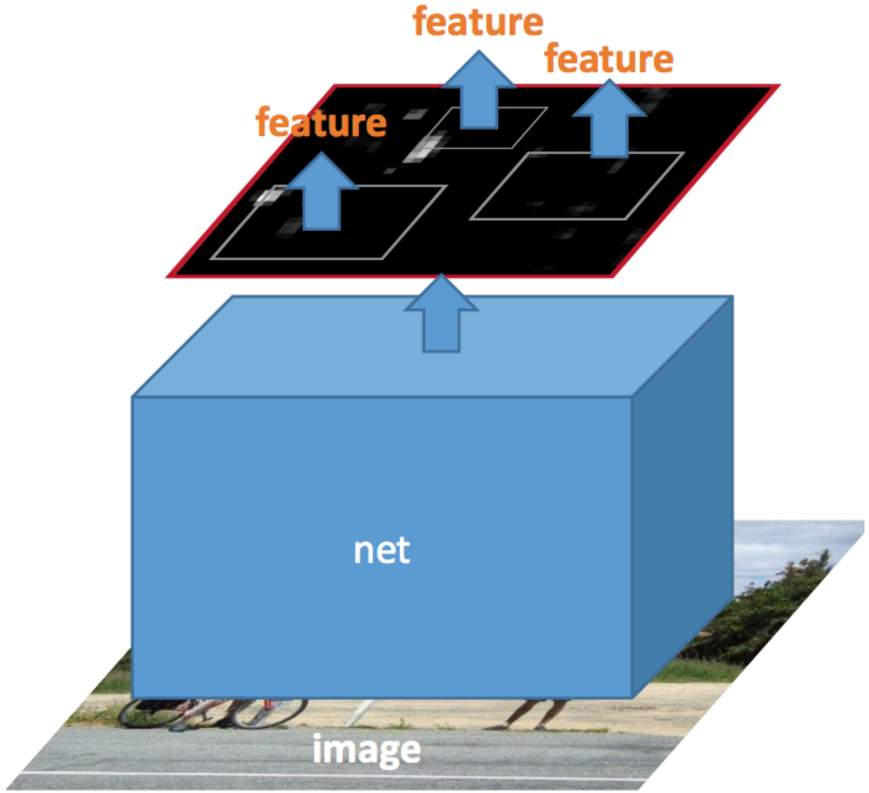
R-CNN vs. SPP

[He et al, ECCV 2014]



R-CNN

2000 nets on image regions



SPP-net

1 net on full image

Fast **R-CNN**

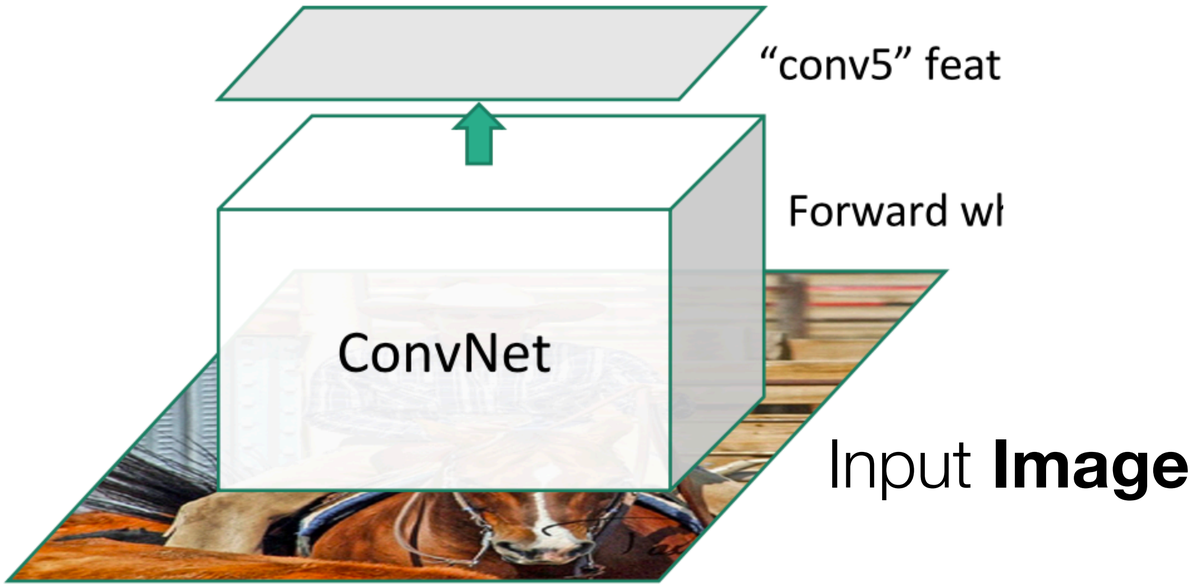
[Girshick et al, ICCV 2015]

Input **Image**

* image from Ross Girshick

Fast **R-CNN**

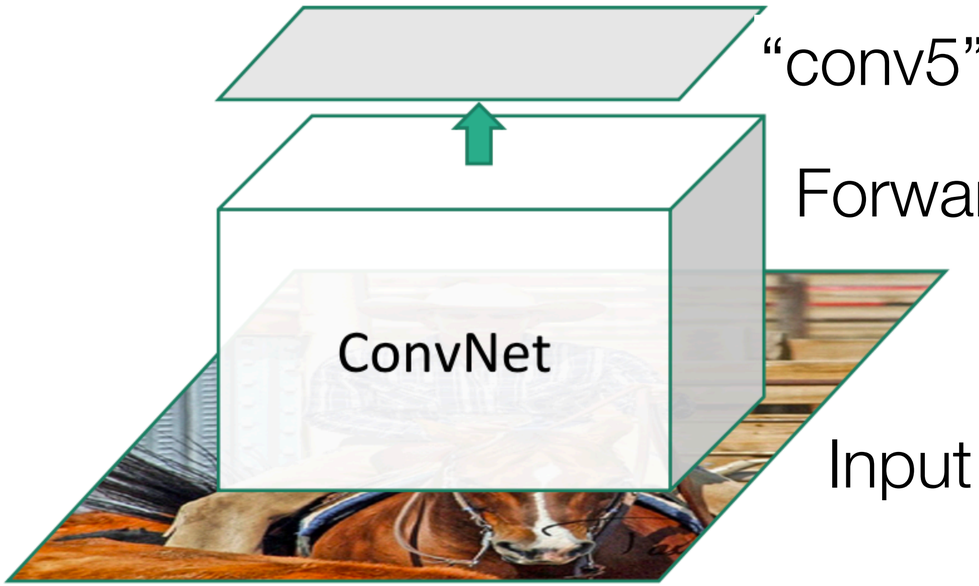
[Girshick et al, ICCV 2015]



* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]



"conv5" feature map

Forward prop the **whole image** through CNN

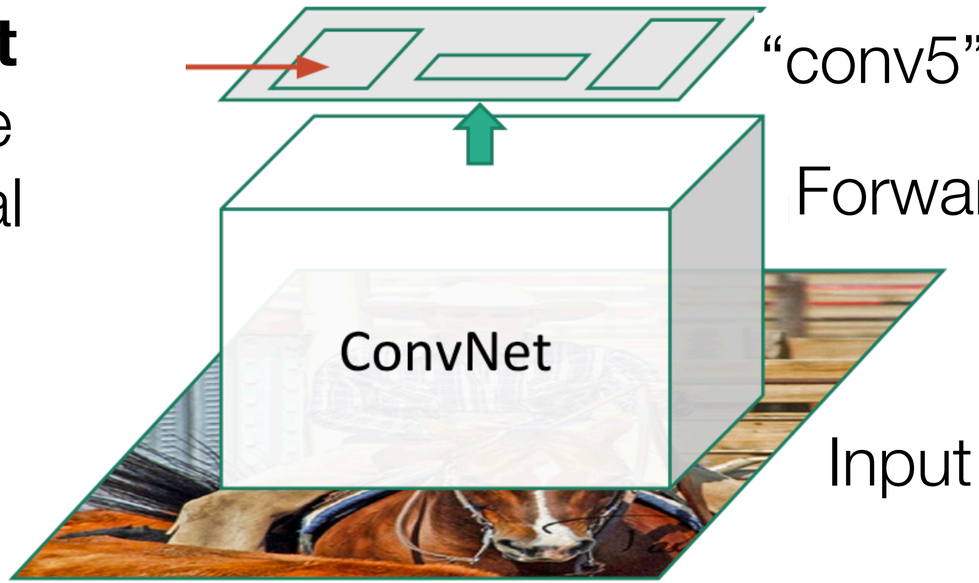
Input **Image**

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

Regions of Interest from the proposal method



"conv5" feature map

Forward prop the **whole image** through CNN

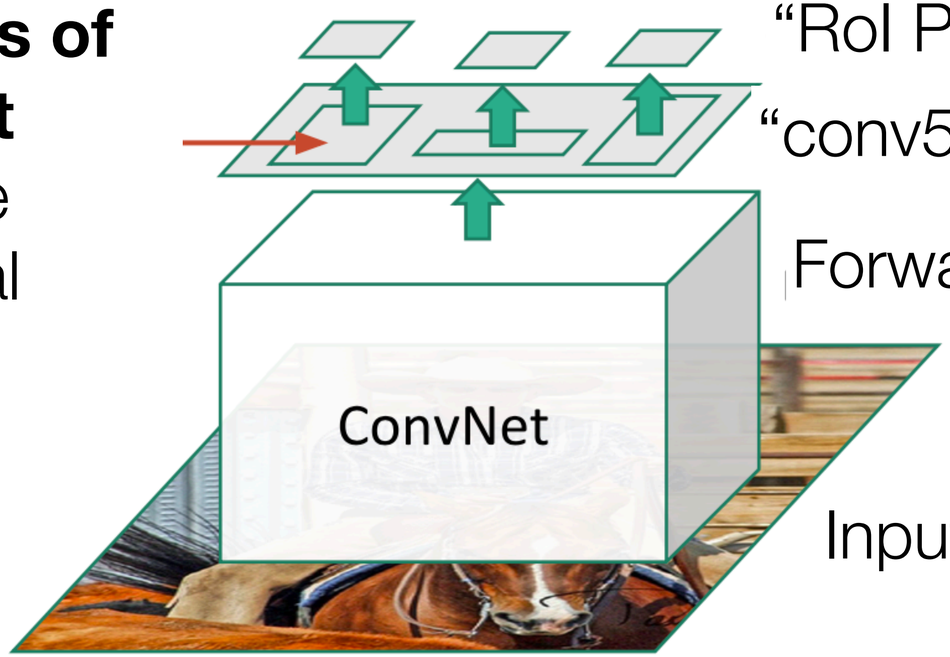
Input **Image**

* 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

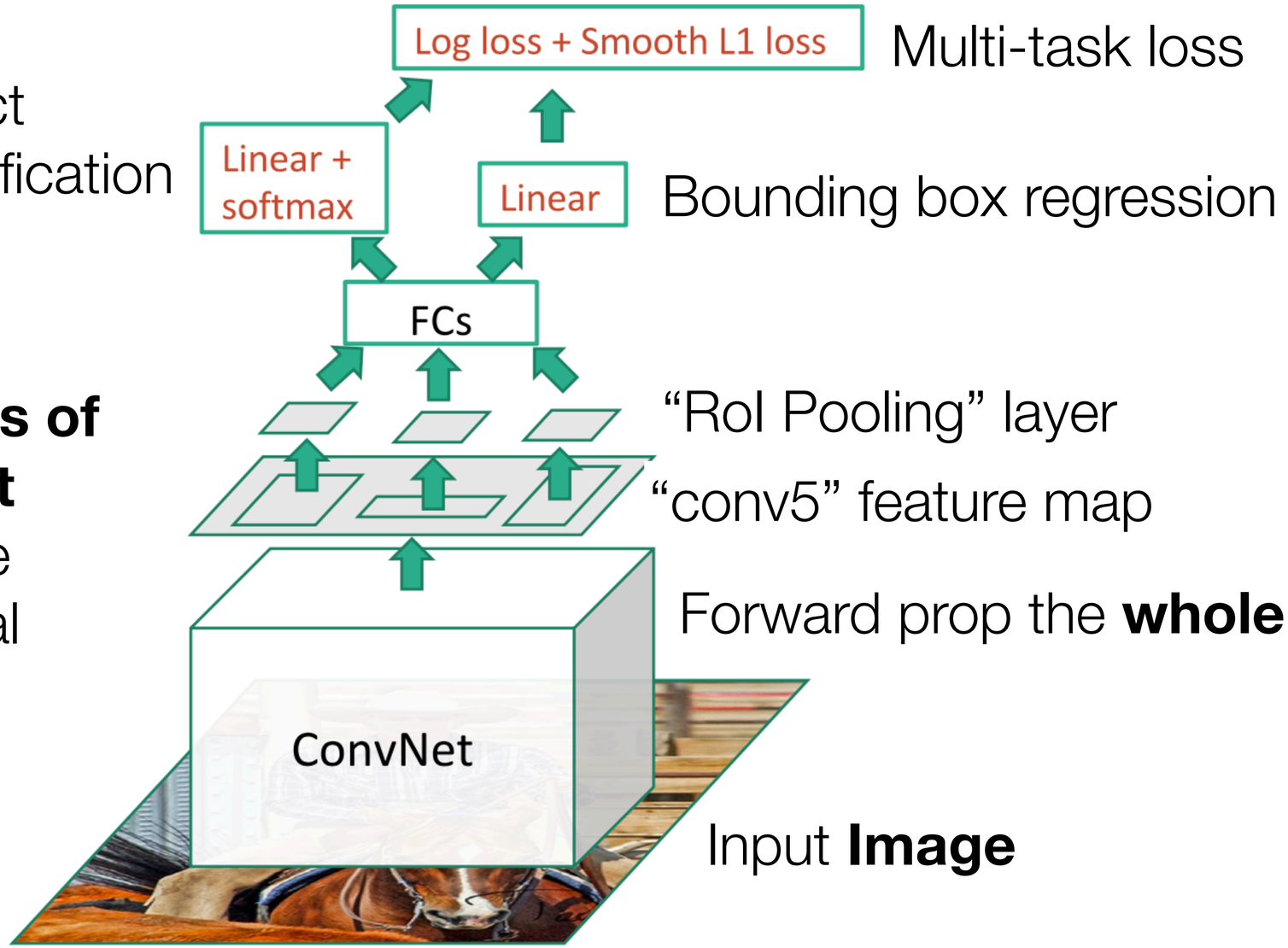
Input **Image**

Girshick, "Fast R-C
Figure copyright R

* image from Ross Girshick

Fast **R-CNN**

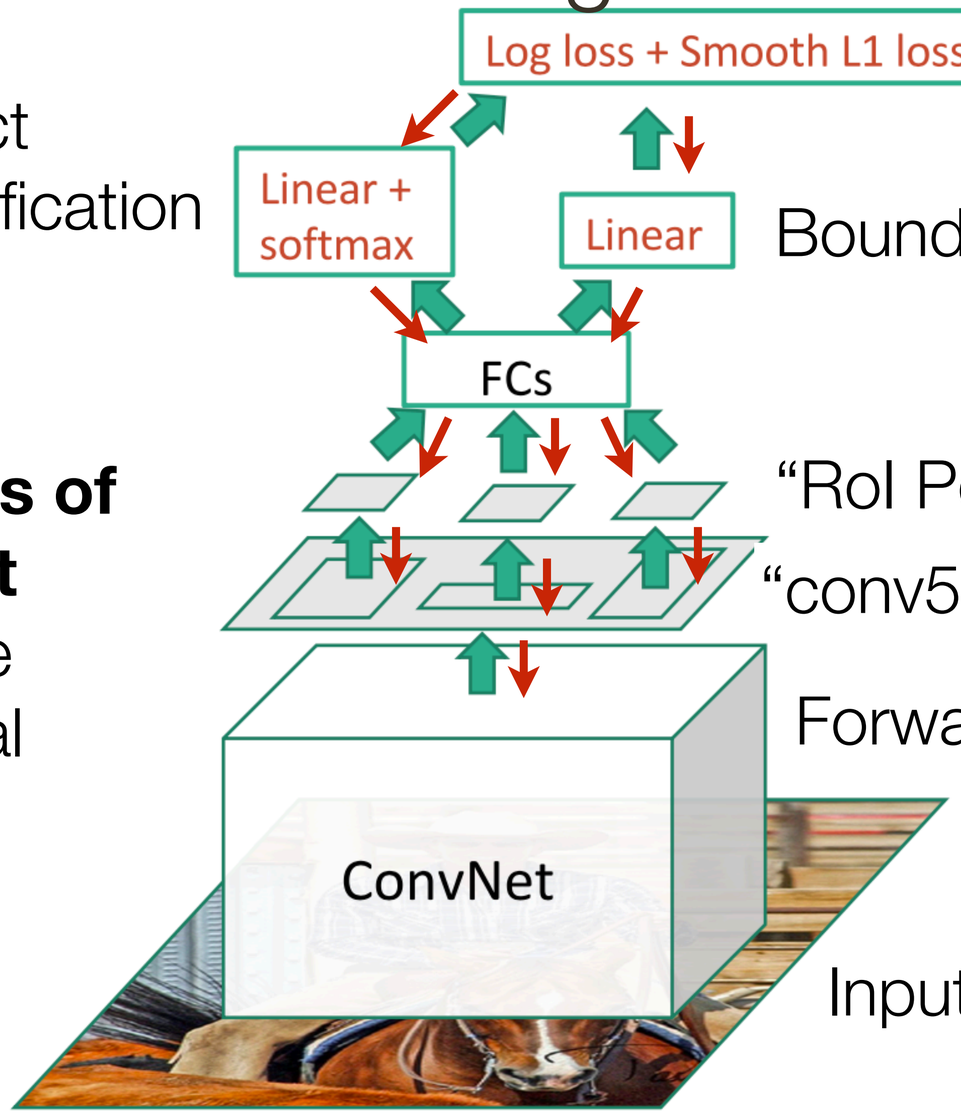
[Girshick et al, ICCV 2015]



Regions of Interest from the proposal method

* image from Ross Girshick

Fast **R-CNN:** Training



Multi-task loss

Object classification

Bounding box regression

[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**

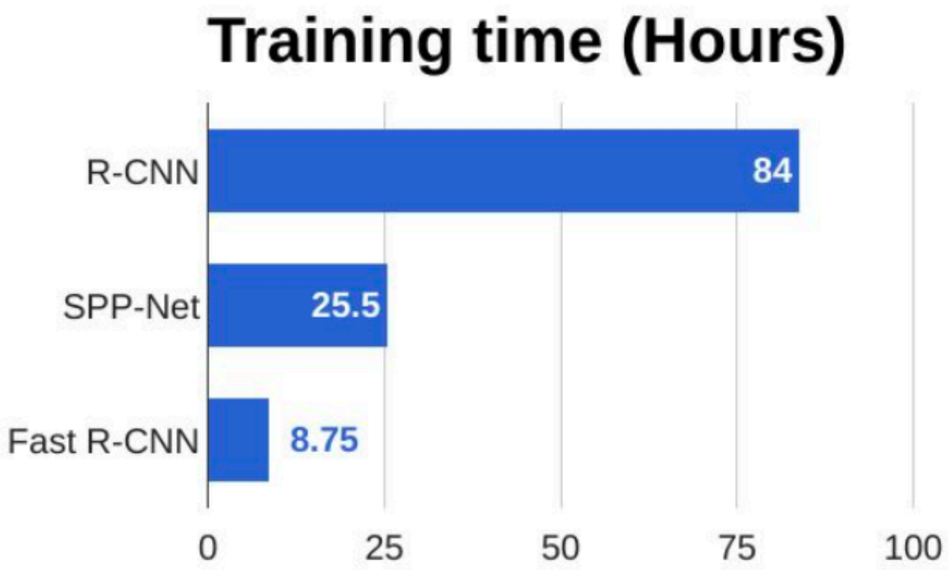
* image from Ross Girshick

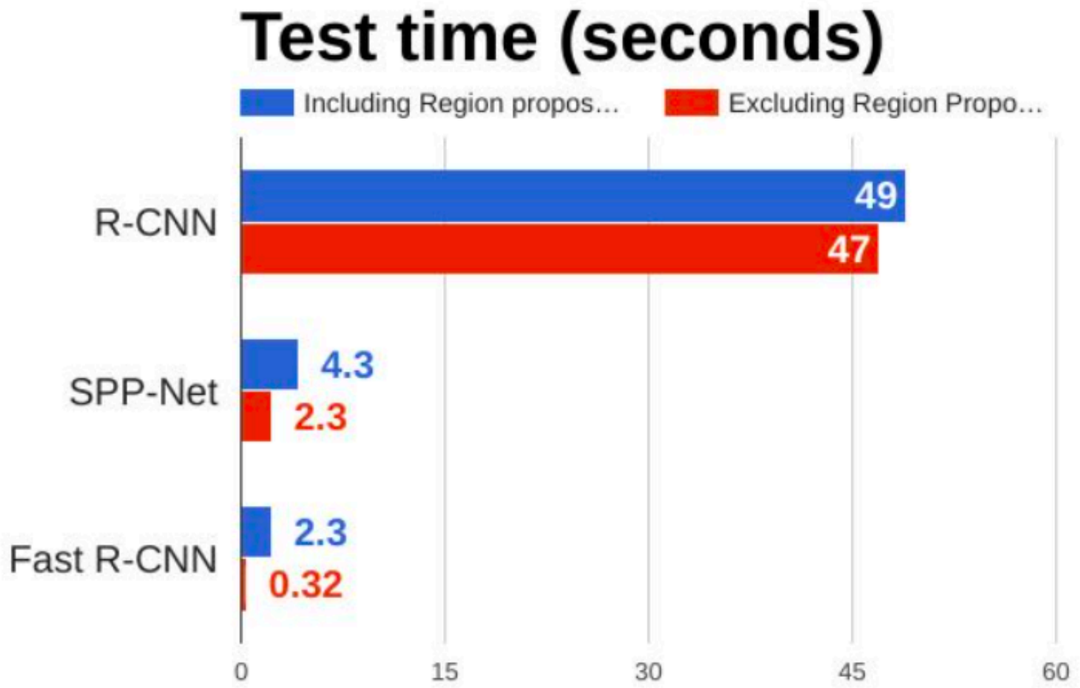
R-CNN vs. SPP vs. Fast R-CNN

[Girshick et al, CVPR 2014]

[Girshick et al, ICCV 2015]

[He et al, ECCV 2014]

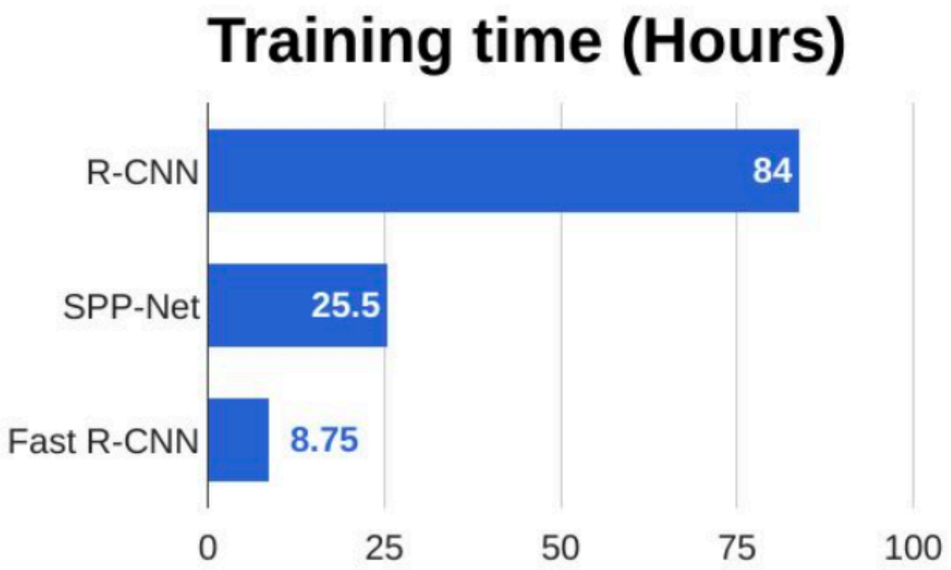


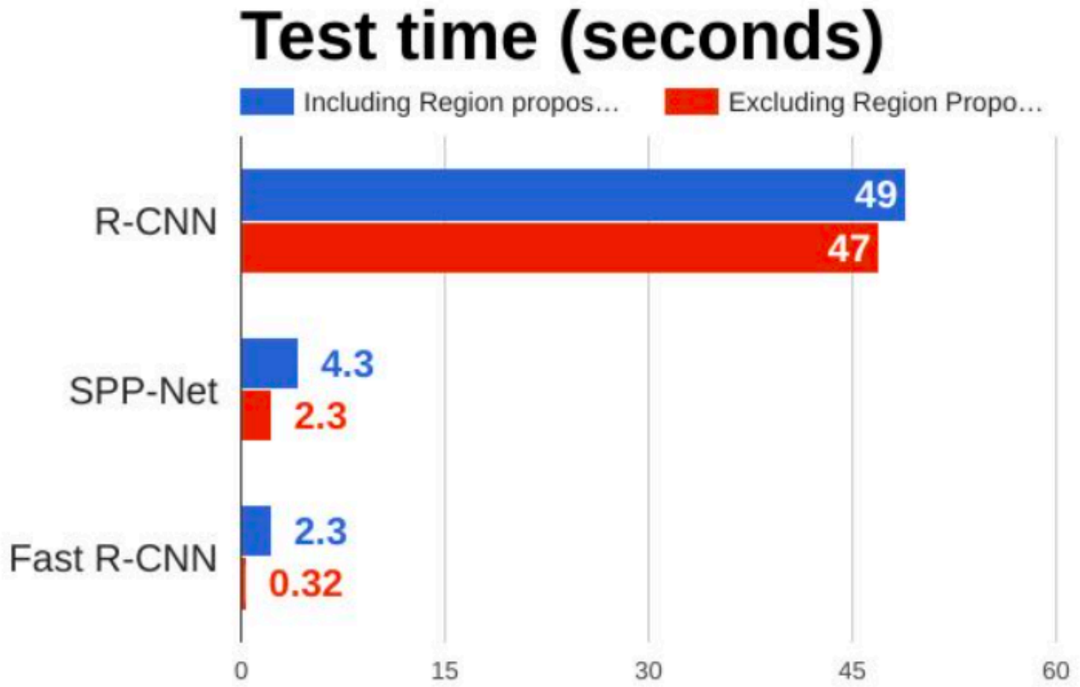


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

R-CNN vs. SPP vs. Fast R-CNN

[Girshick et al, CVPR 2014]
[Girshick et al, ICCV 2015]
[He et al, ECCV 2014]





Observation: Performance dominated by the region proposals at this point!

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

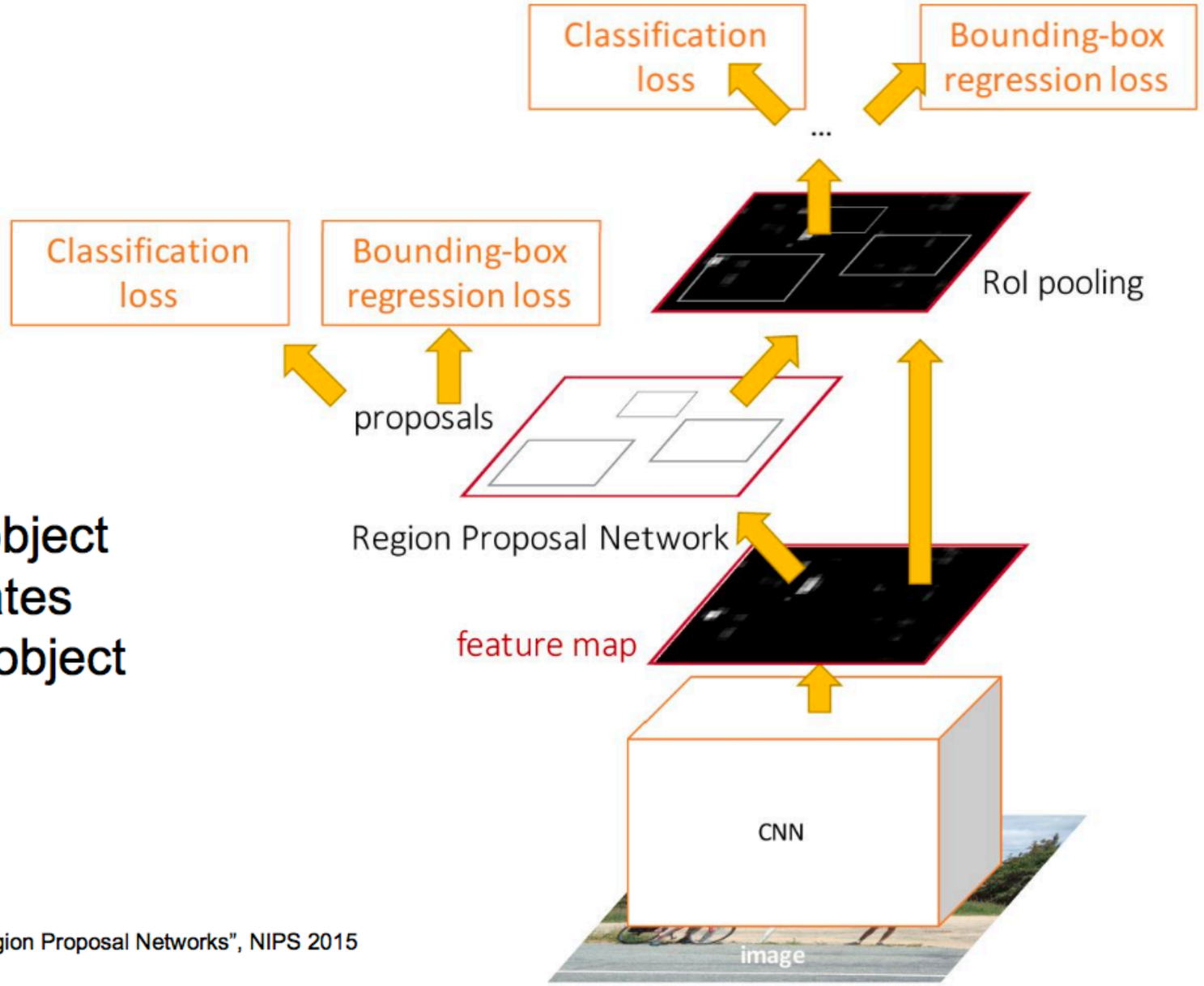
Faster **R-CNN**

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

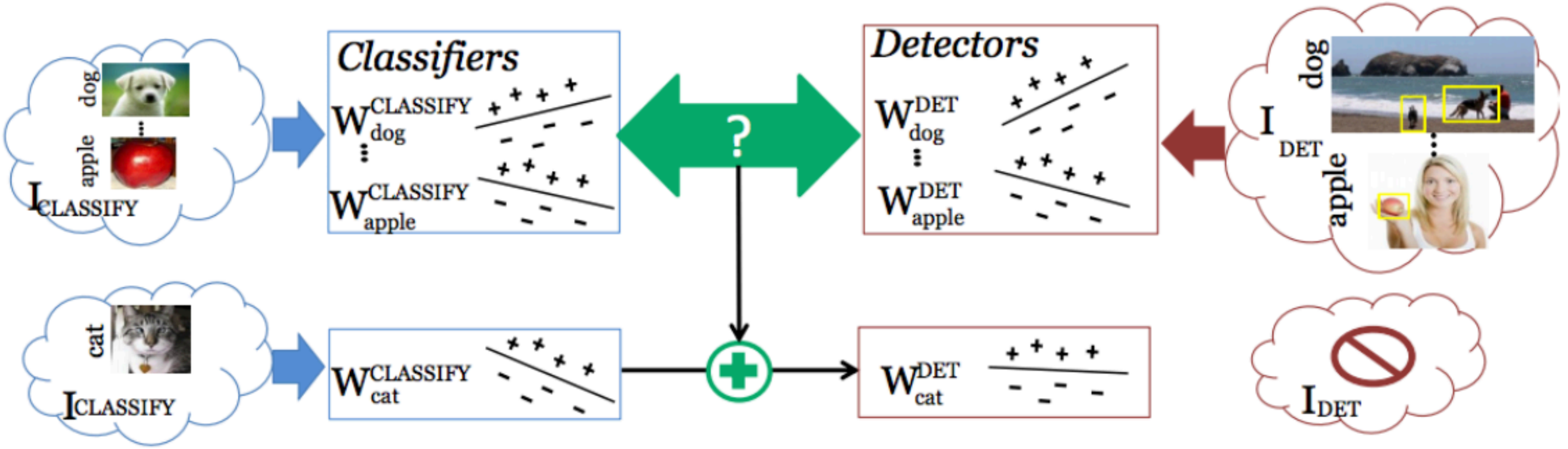
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

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

LSDA: Large Scale Detection through Adaptation



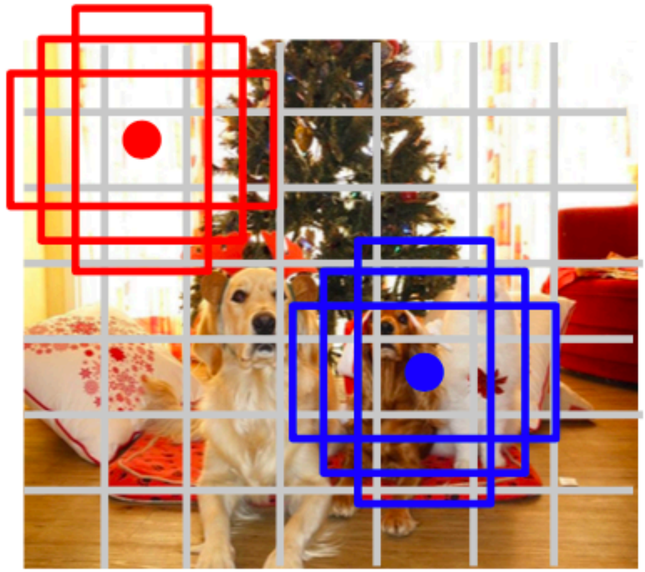
$$\mathbf{W}_{cat}^{DETECT} = \mathbf{W}_{cat}^{CLASIFY} + \delta\mathbf{W}_{cat}$$

[Hoffman et al, NIPS 2014]

YOLO: You Only Look Once

[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

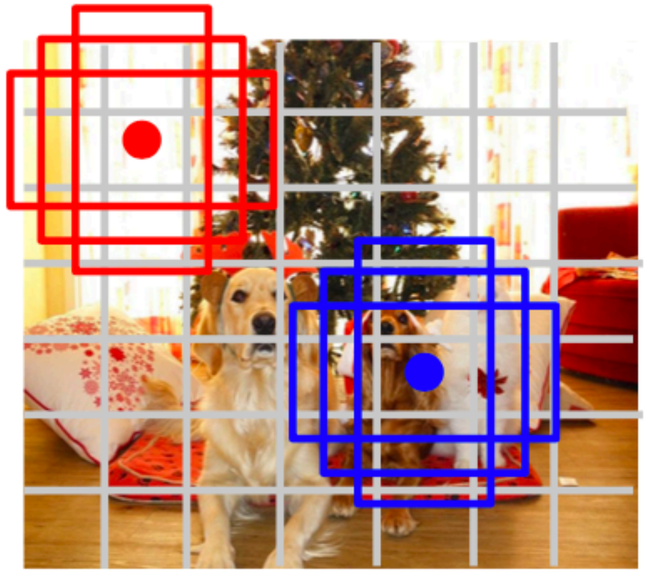
Output:
7 x 7 x (5 * B + C)

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

YOLO: You Only Look Once

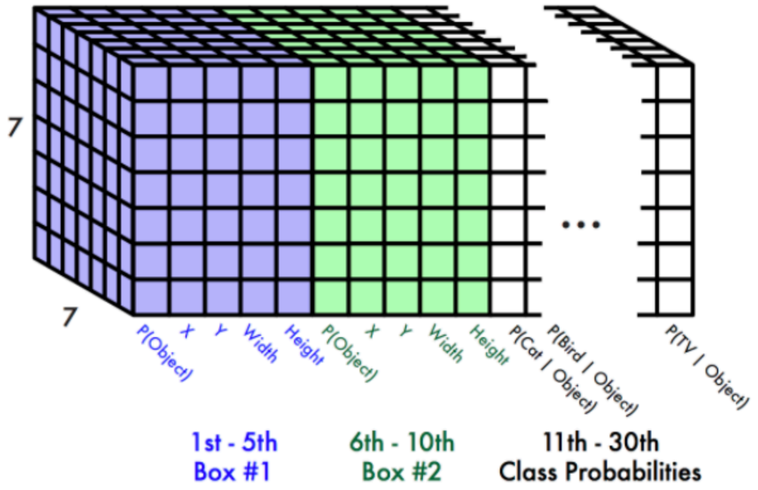
[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes**
centered at each grid cell
Here B = 3




YOLO v2
http://pjreddie.com/yolo


YOLO v2
http://pjreddie.com/yolo