Computer Vision with CNN

Behnia - Heydari

Amirkabir University of Technology

May 21, 2019

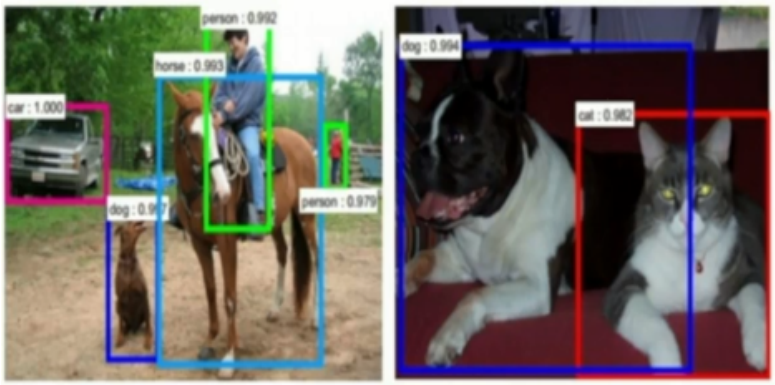
Overview

Cameras Everywhere

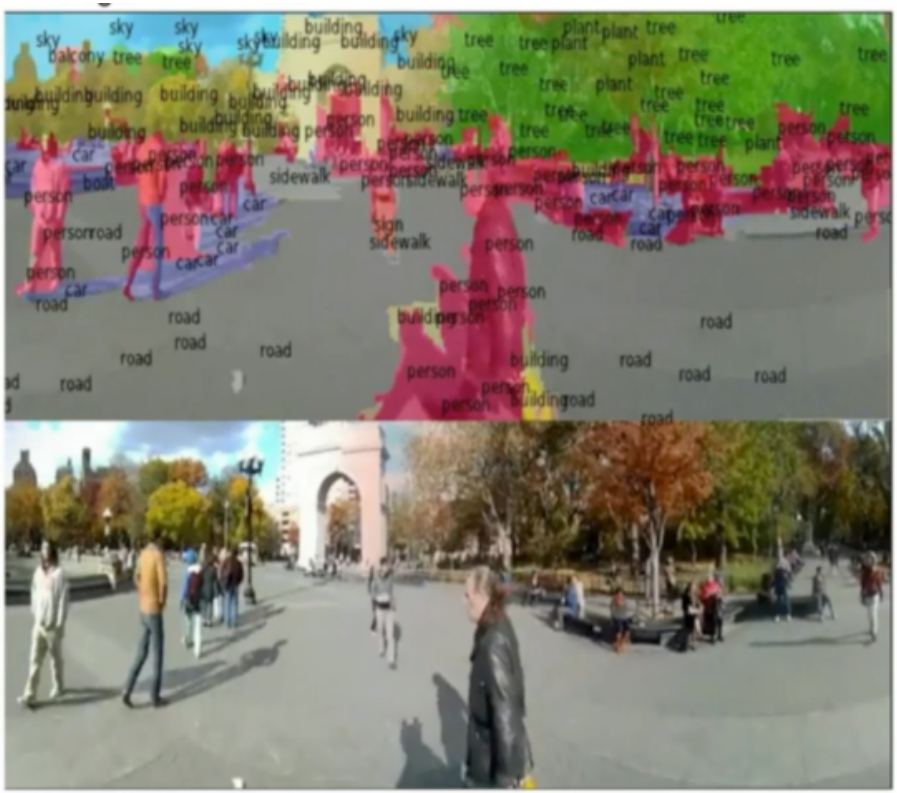
Smartphones

- Exploding number of sensors vs. humans

ConvNets Everywhere



Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with



Figures copyright Clement Farabet, 2012

ConvNets Everywhere

Photo by Lane McIntosh. Copyright CS231n 2017.

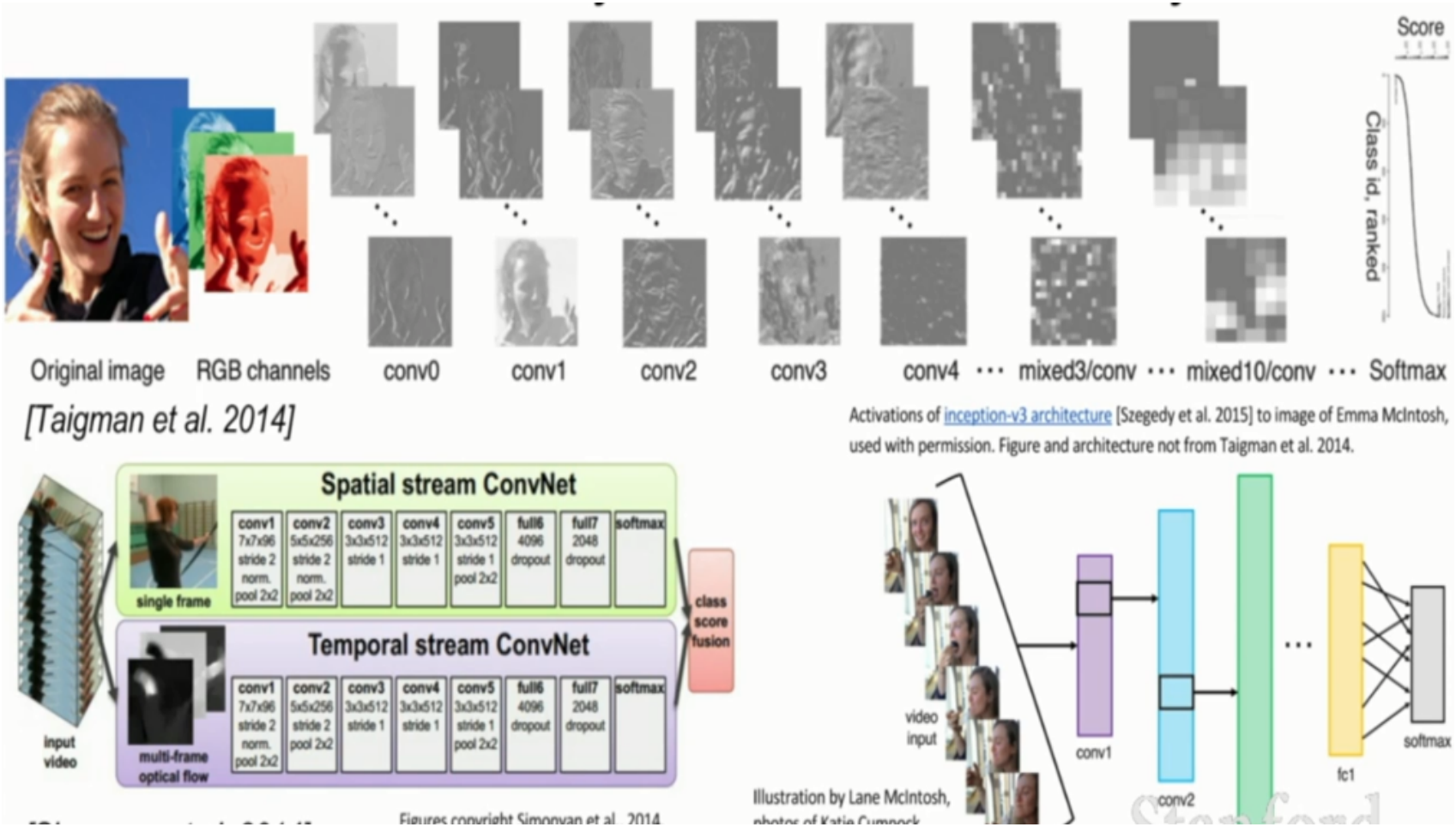
self-driving cars

NVIDIA Tesla line

(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

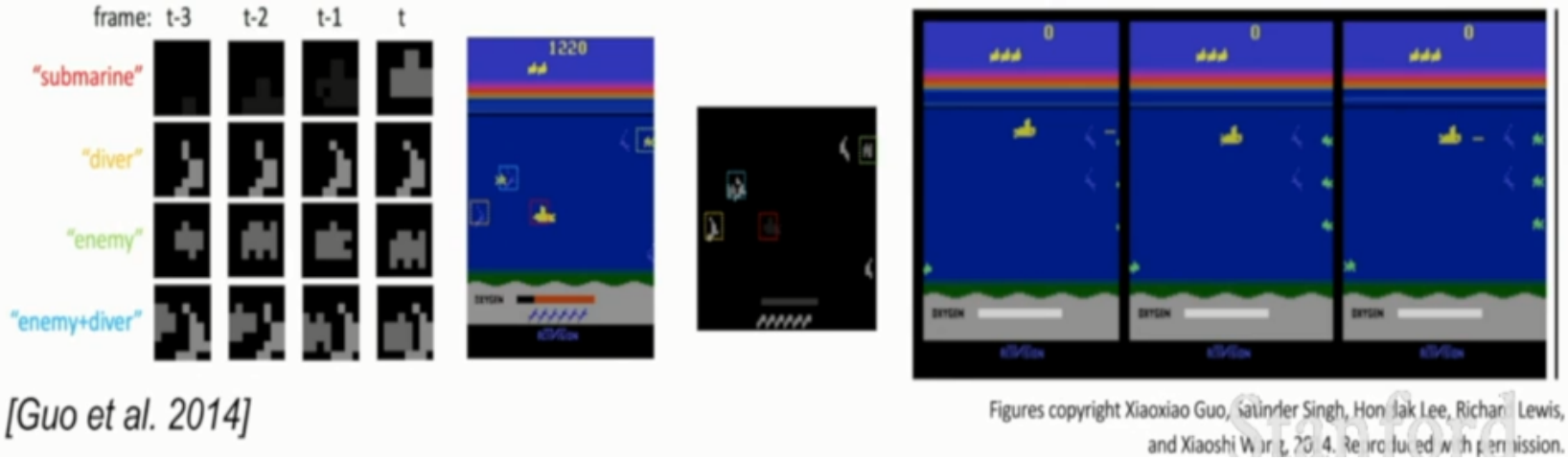
ConvNets Everywhere



ConvNets Everywhere

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

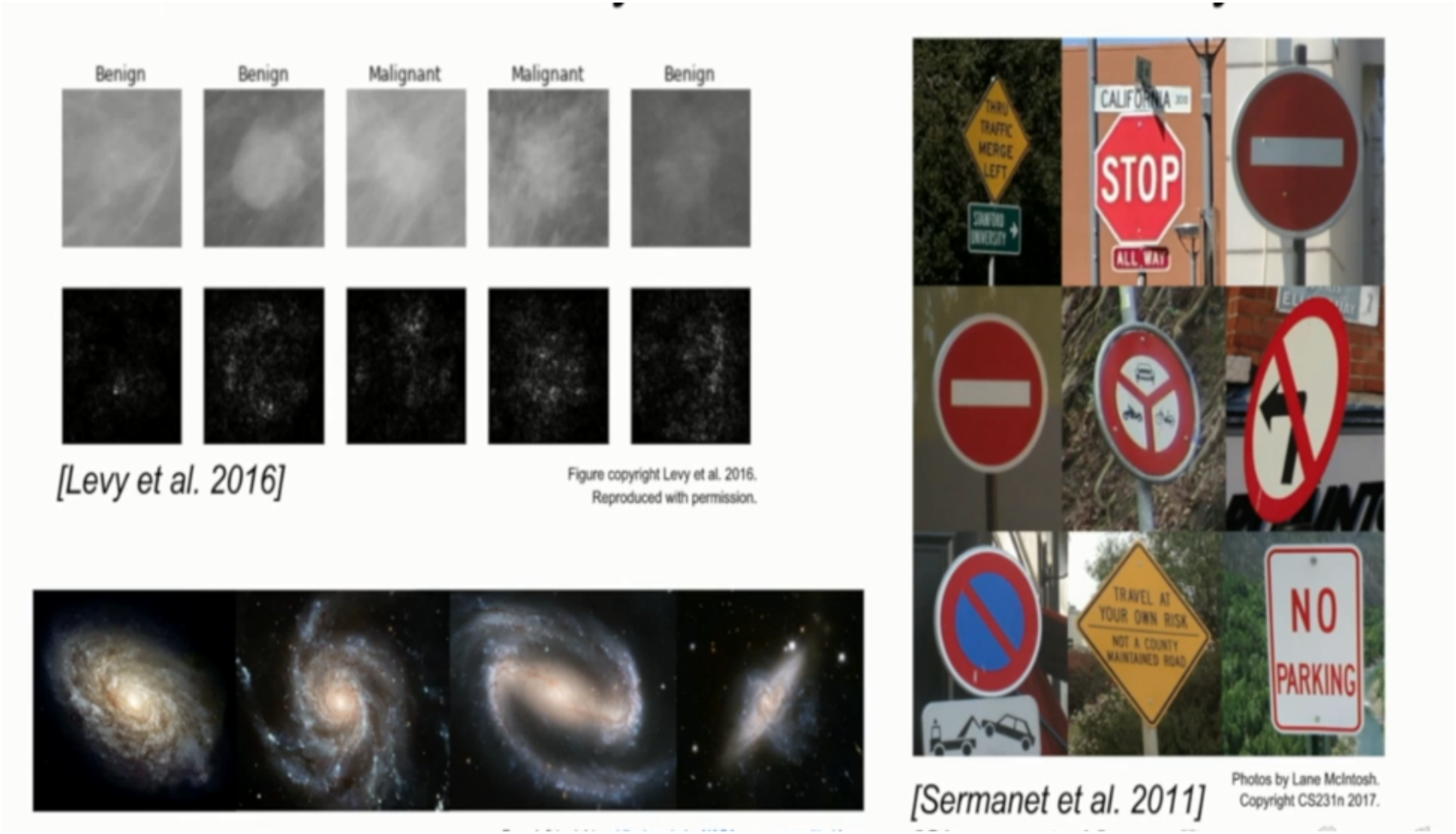
[Toshev, Szegedy 2014]



[Guo et al. 2014]

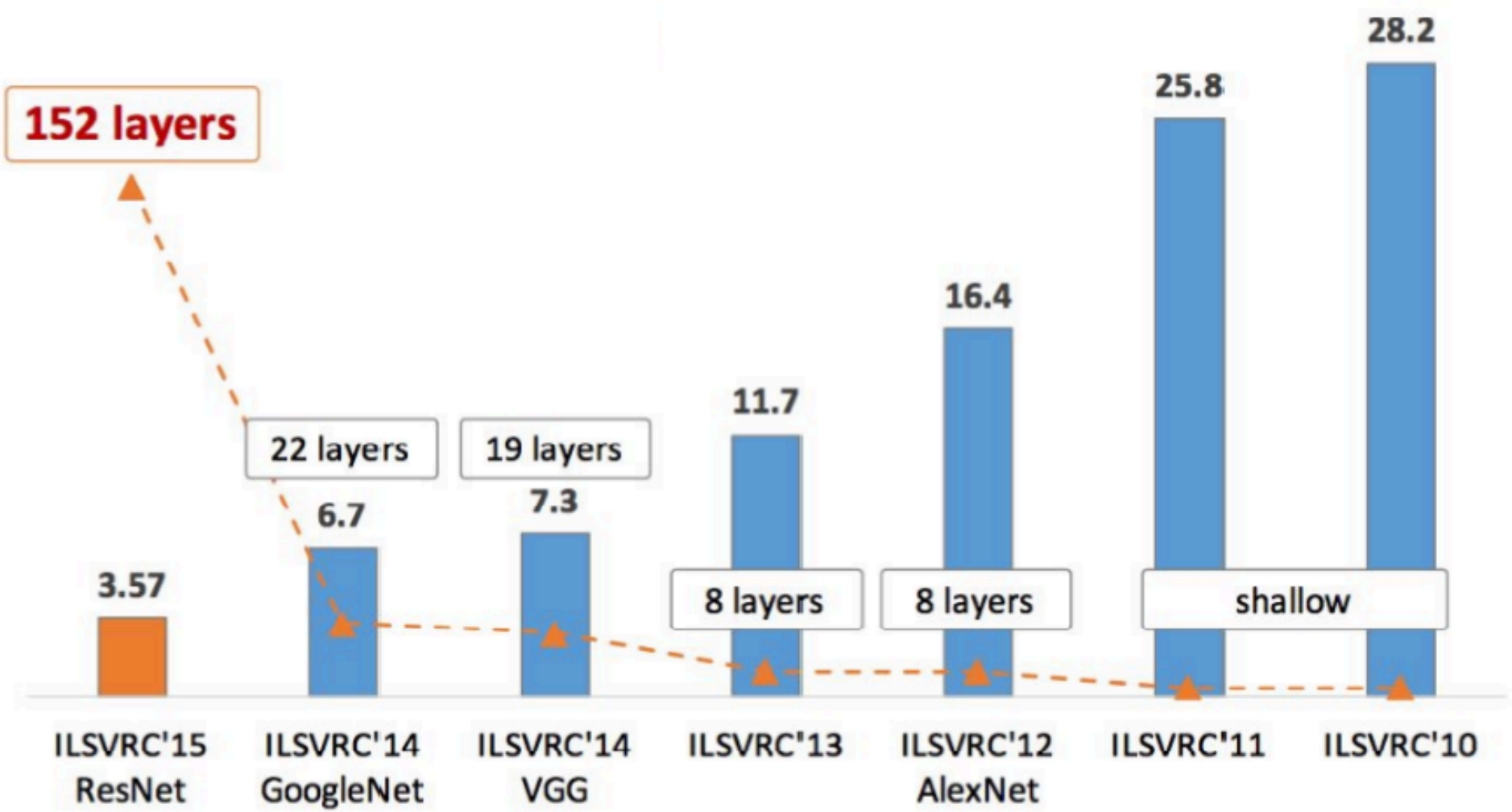
Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

ConvNets Everywhere



Neat Dataset

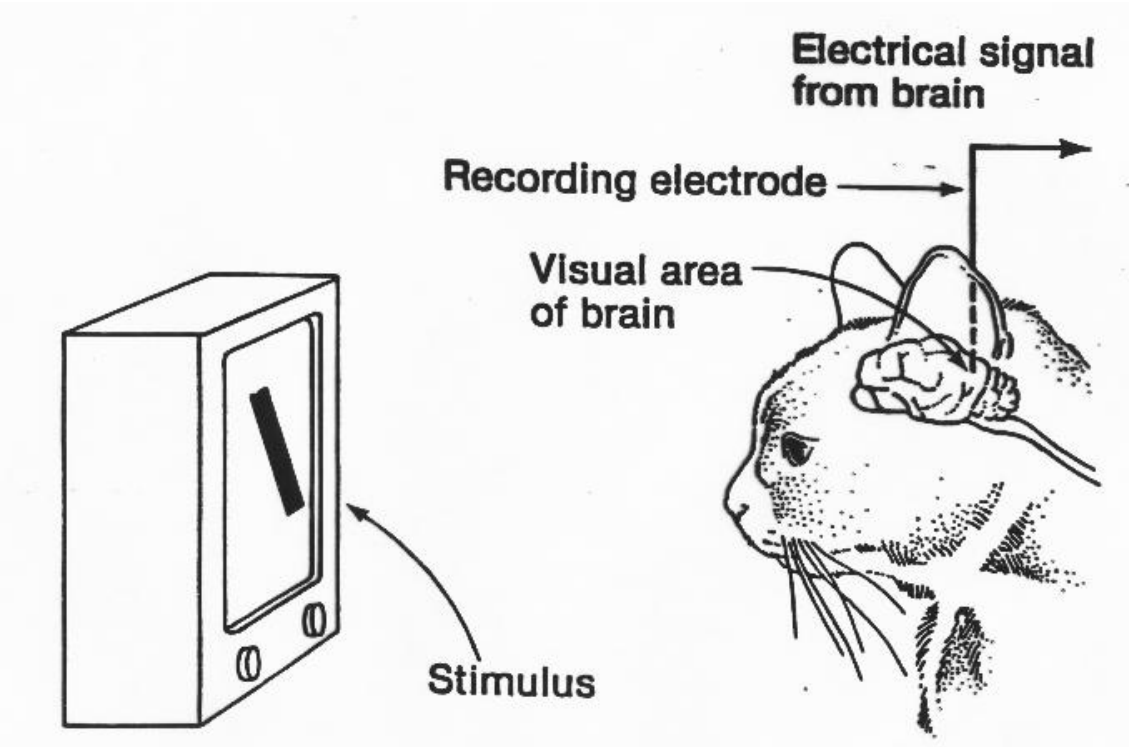
Role of CNN


152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10

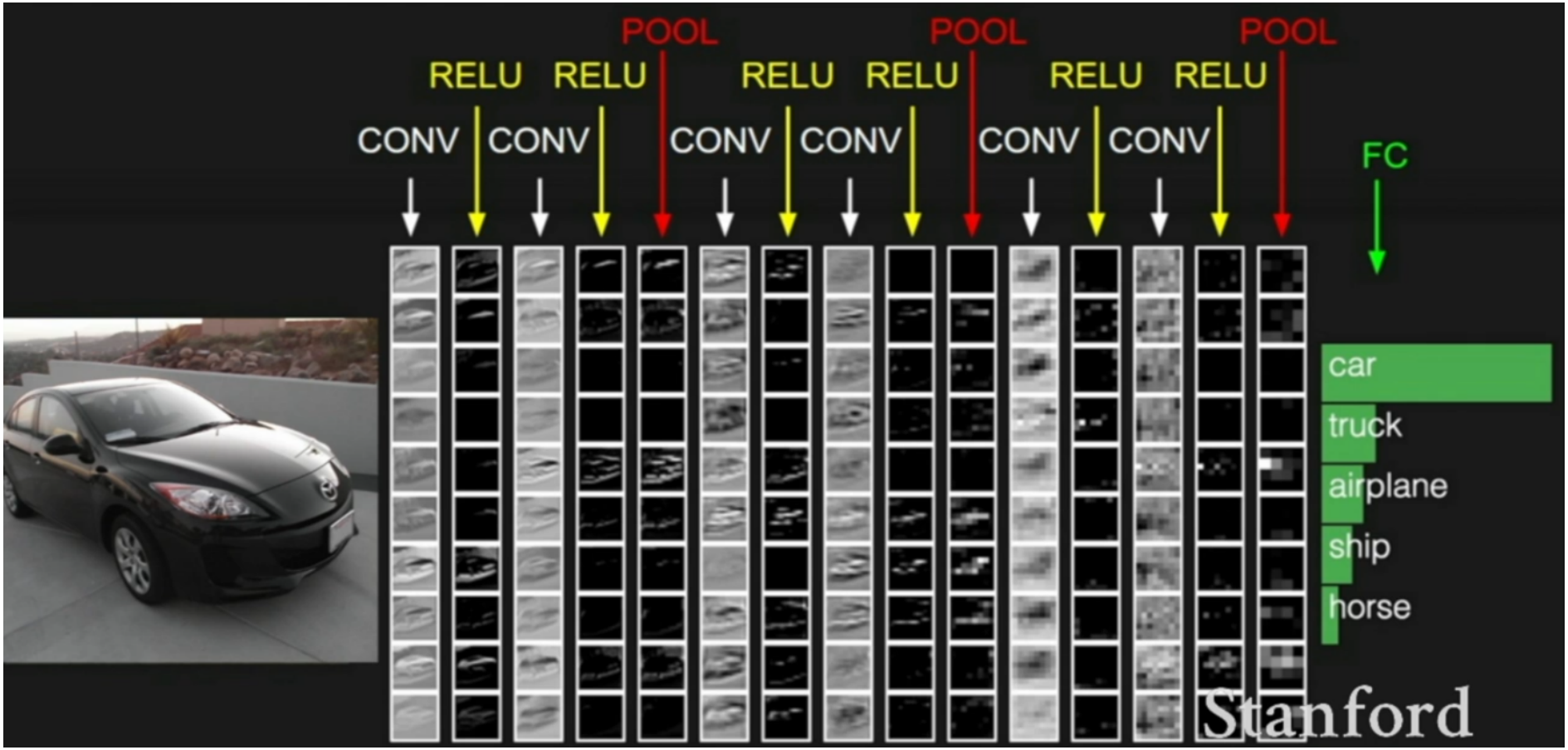
Cat's Brain

Types of cells:

- simple cells
- complex cells
- hypercomplex cells



CNN


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse
Stanford

Convolution

Figure: Convolution Layer

Complexity of Features

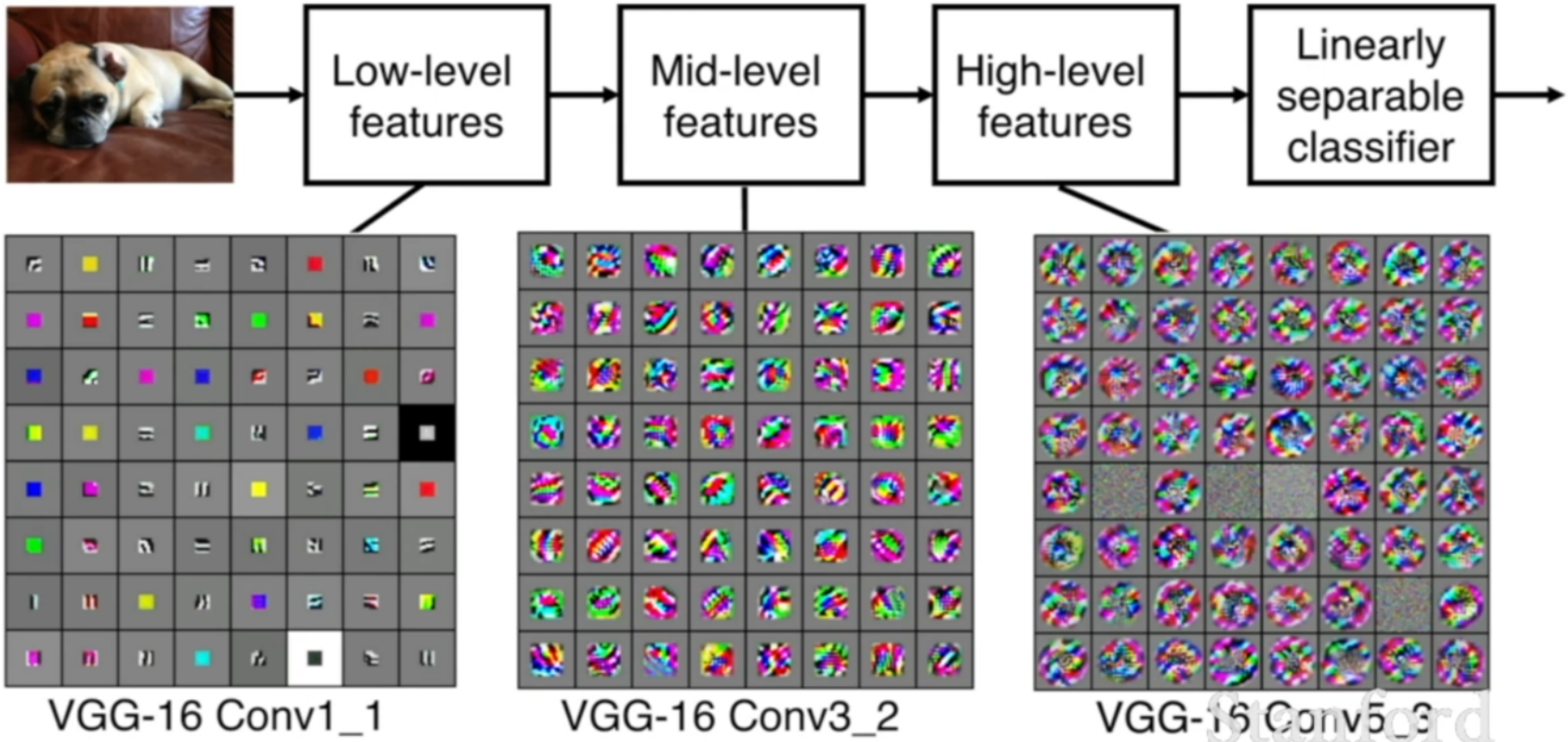


Figure: complexity of features in each layer

Complexity of Features

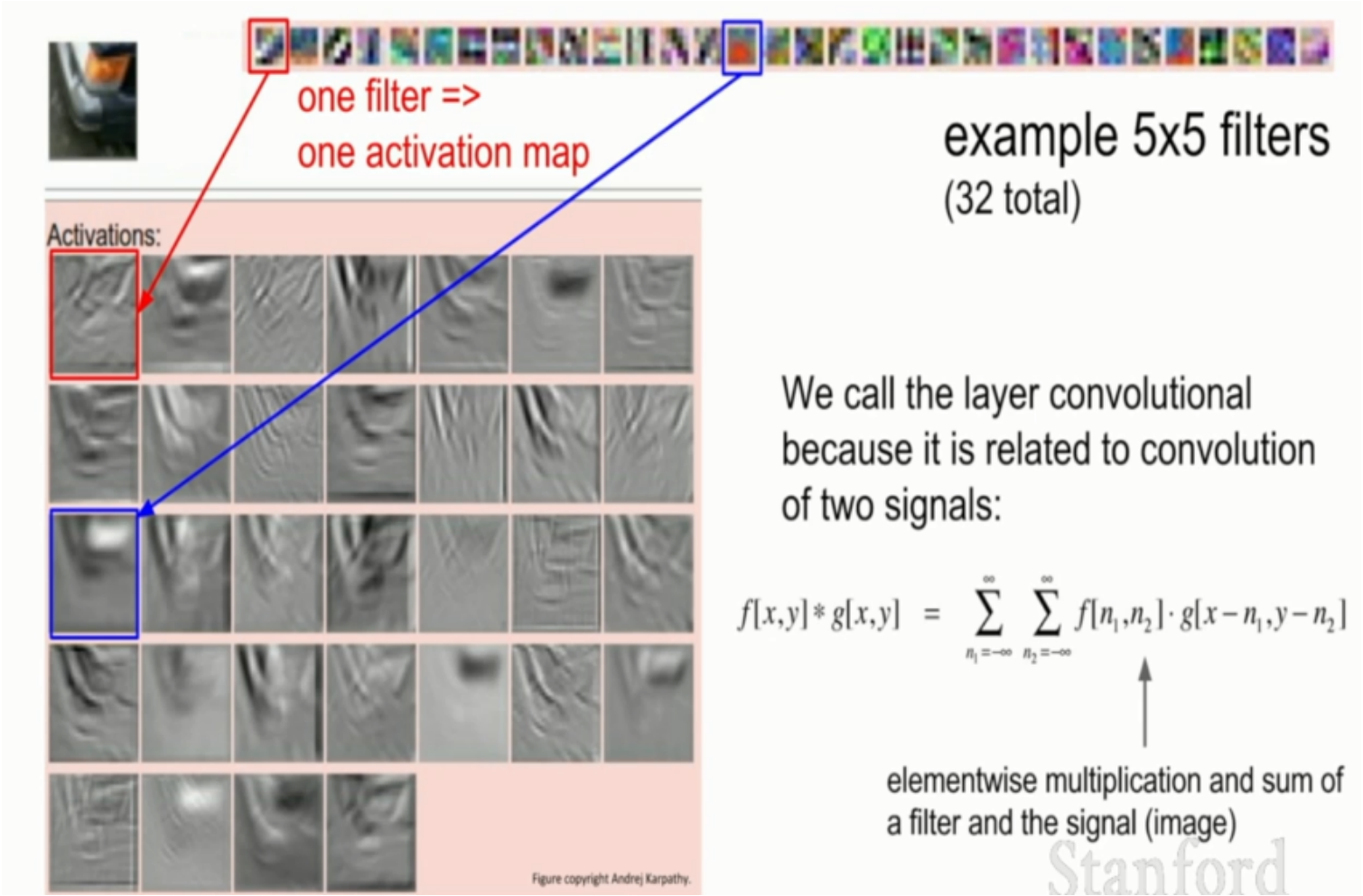


Figure: complexity of each layer output

Sliding a Filter

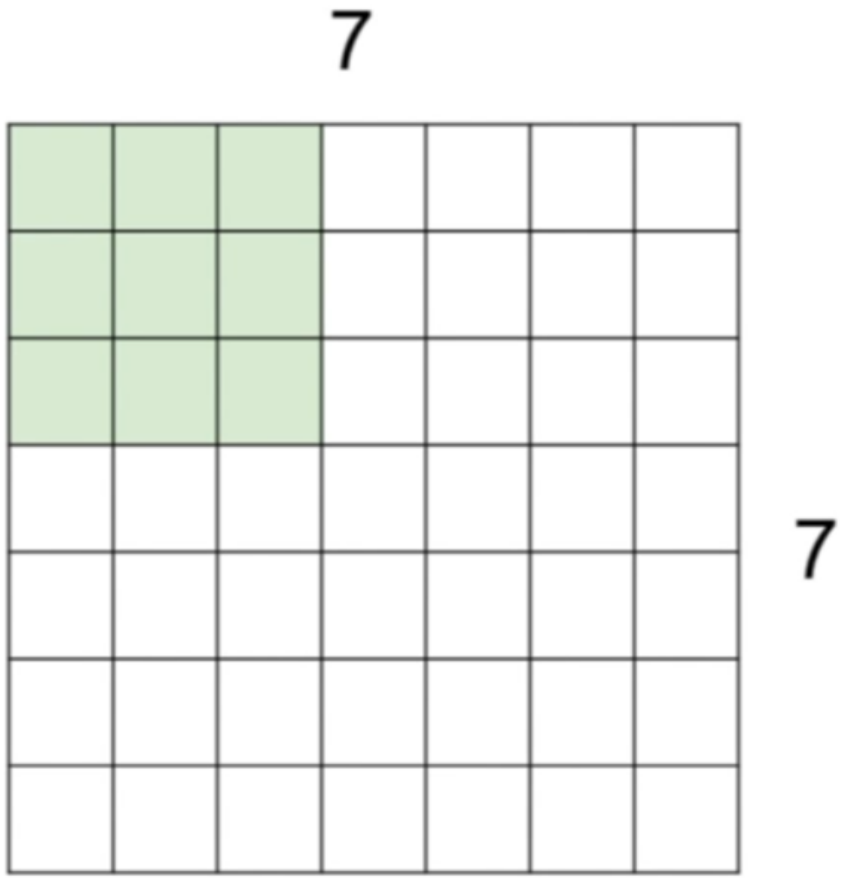


Figure: filtering example

Sliding a Filter

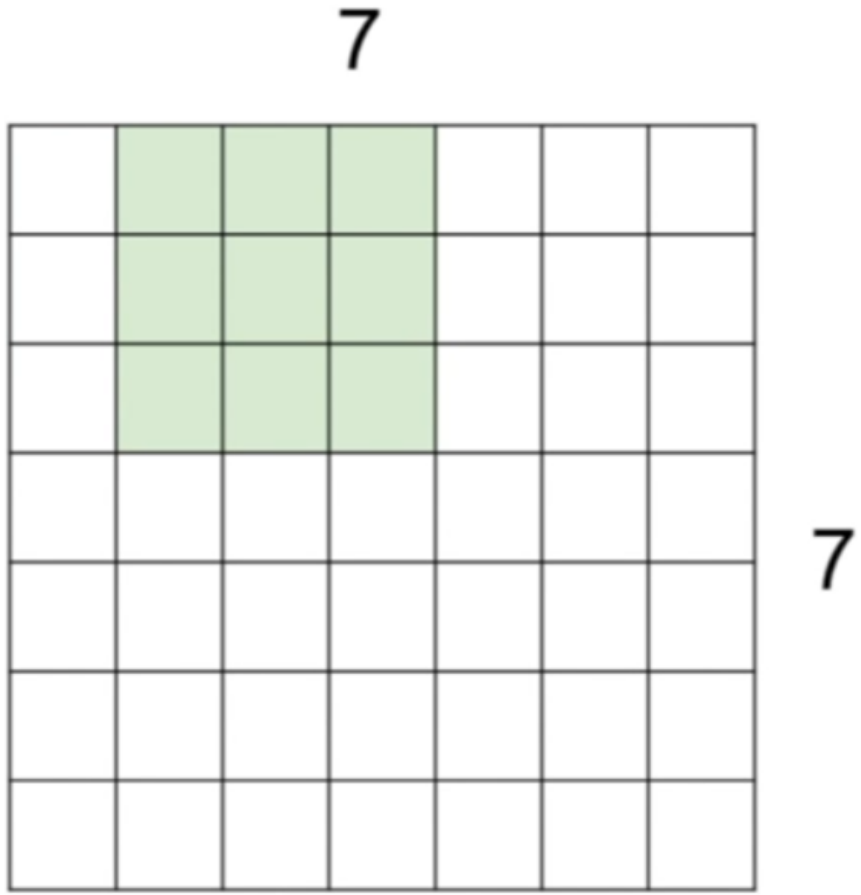


Figure: filtering example

Sliding a Filter

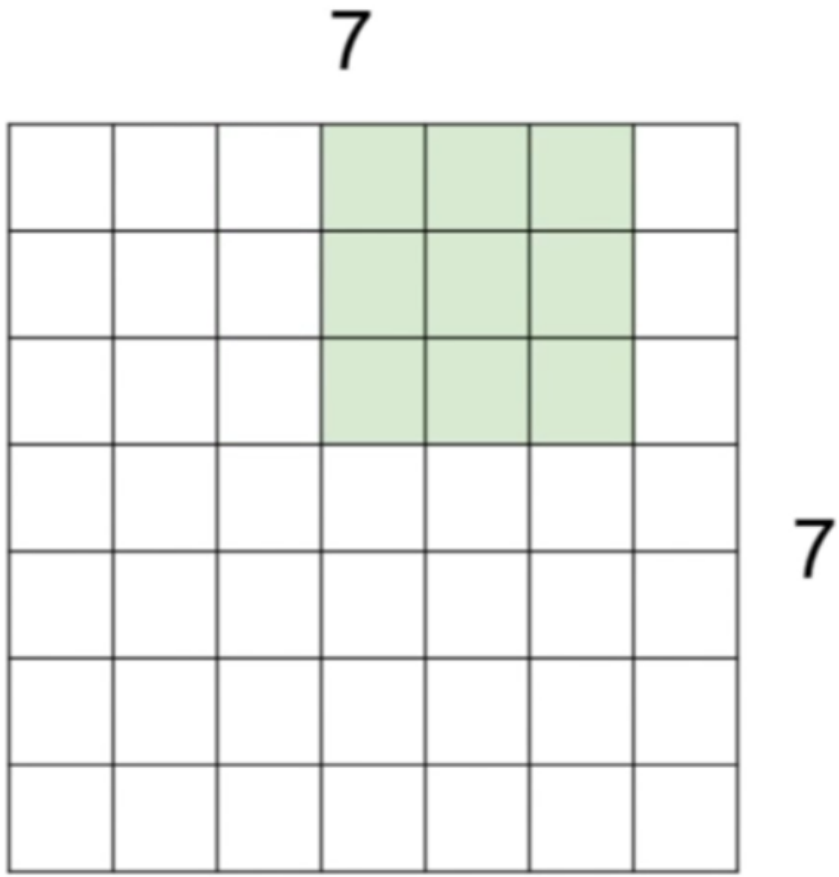


Figure: filtering example

Sliding a Filter

What if using a 3*3 filter with stride 3 ?!

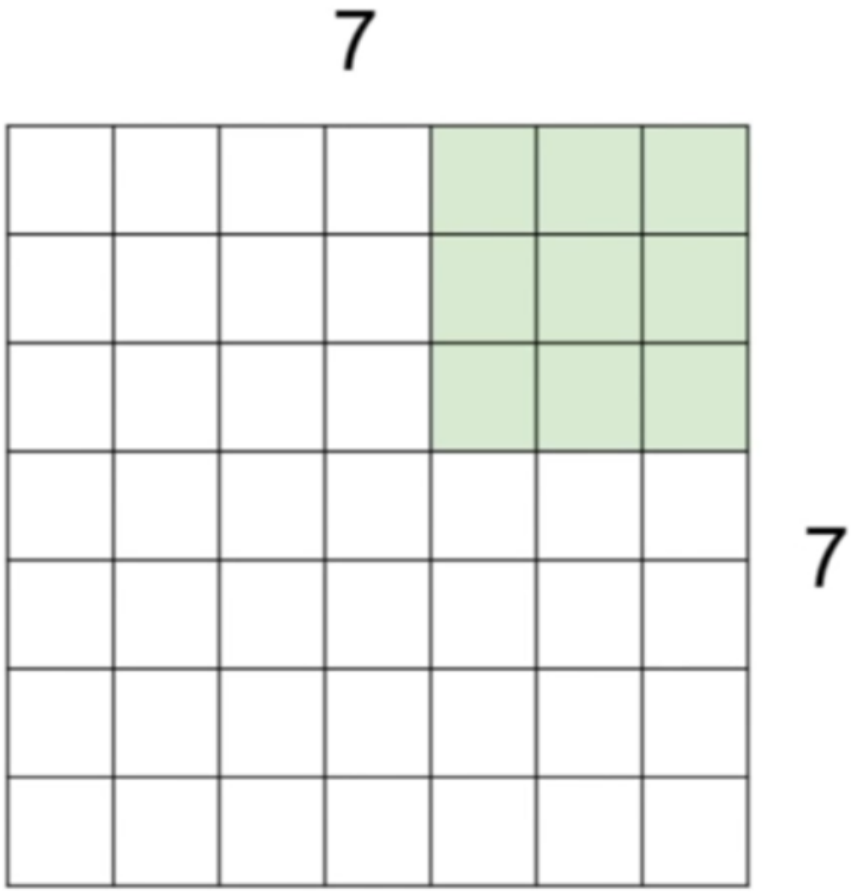


Figure: filtering example

Padding

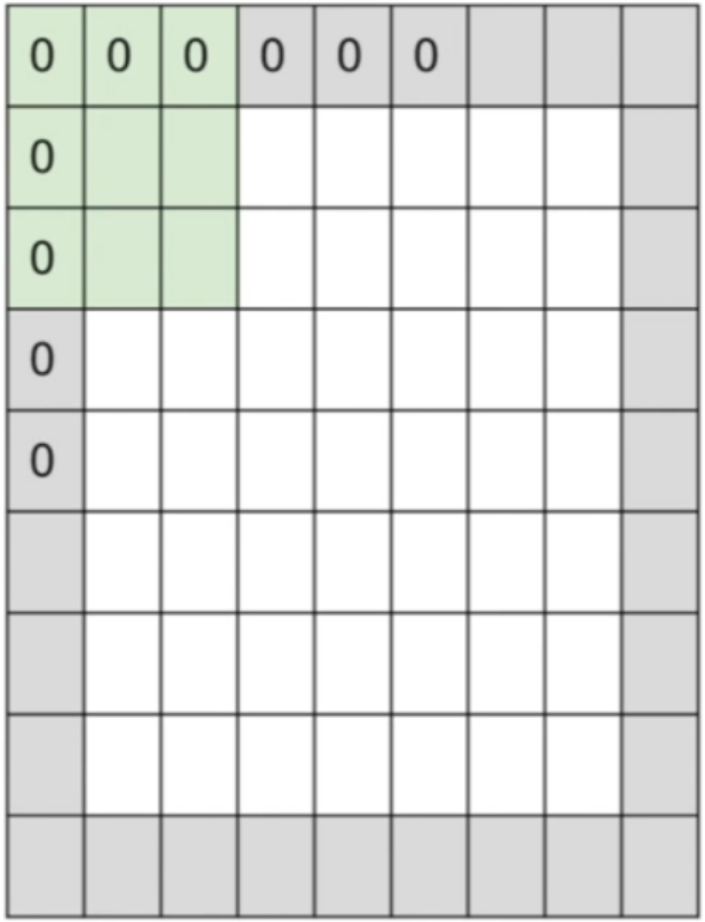


Figure: Padding: control output size

Summary

- Accept a volume of size N x N
- requires 4 hyper parameters:
  - Filter's spatial extent F
  - Filter's Stride S
  - Amount of zero padding P
  - Number of kernels ($\times depth \times$)
- Produces a volume of size $M \times M$
  - M $= \frac{(N-F+2p)}{S} + 1$
  - Number of parameters : $(F \times F \times D + b) \times k$

CNN

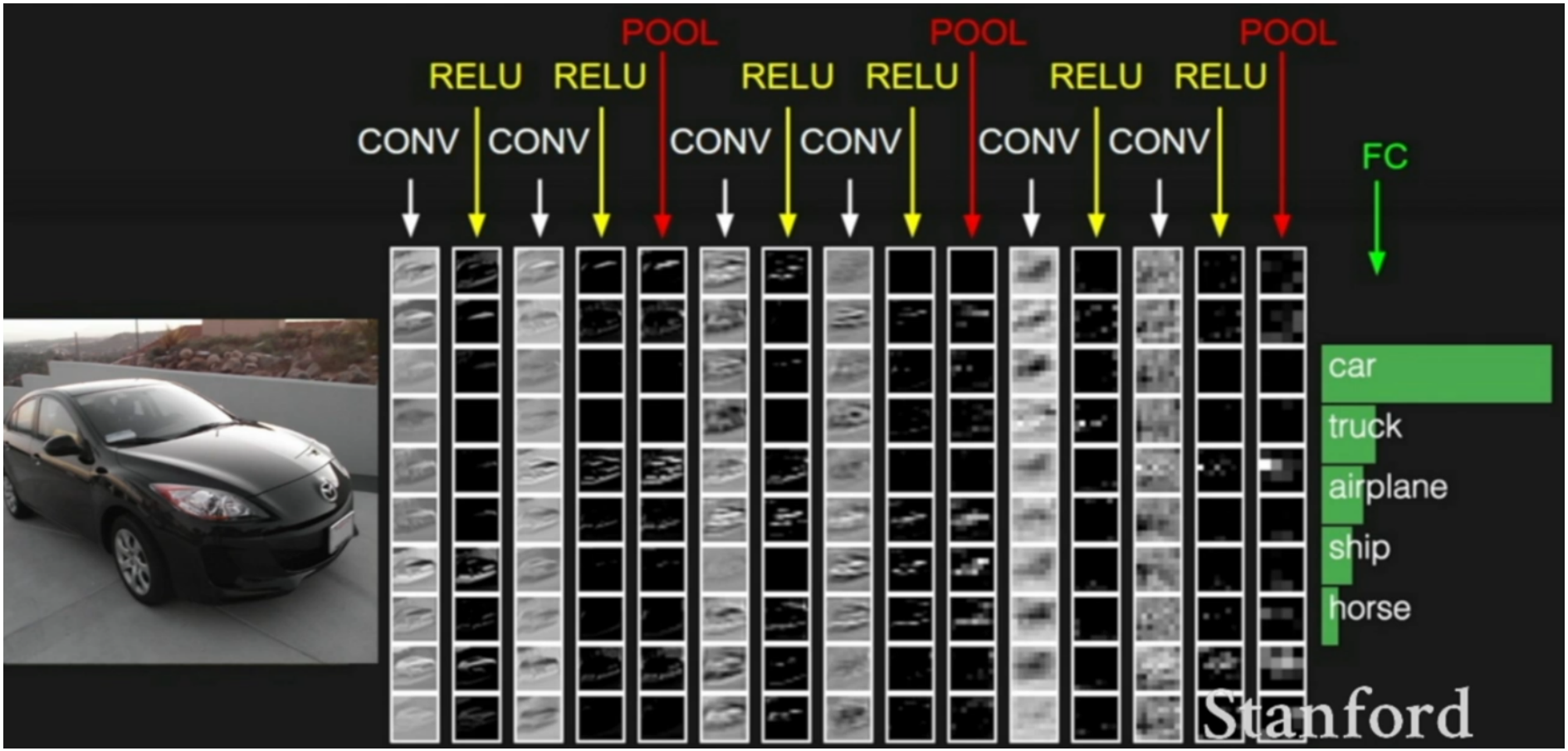


Figure: Fully Connected Layer

Pooling

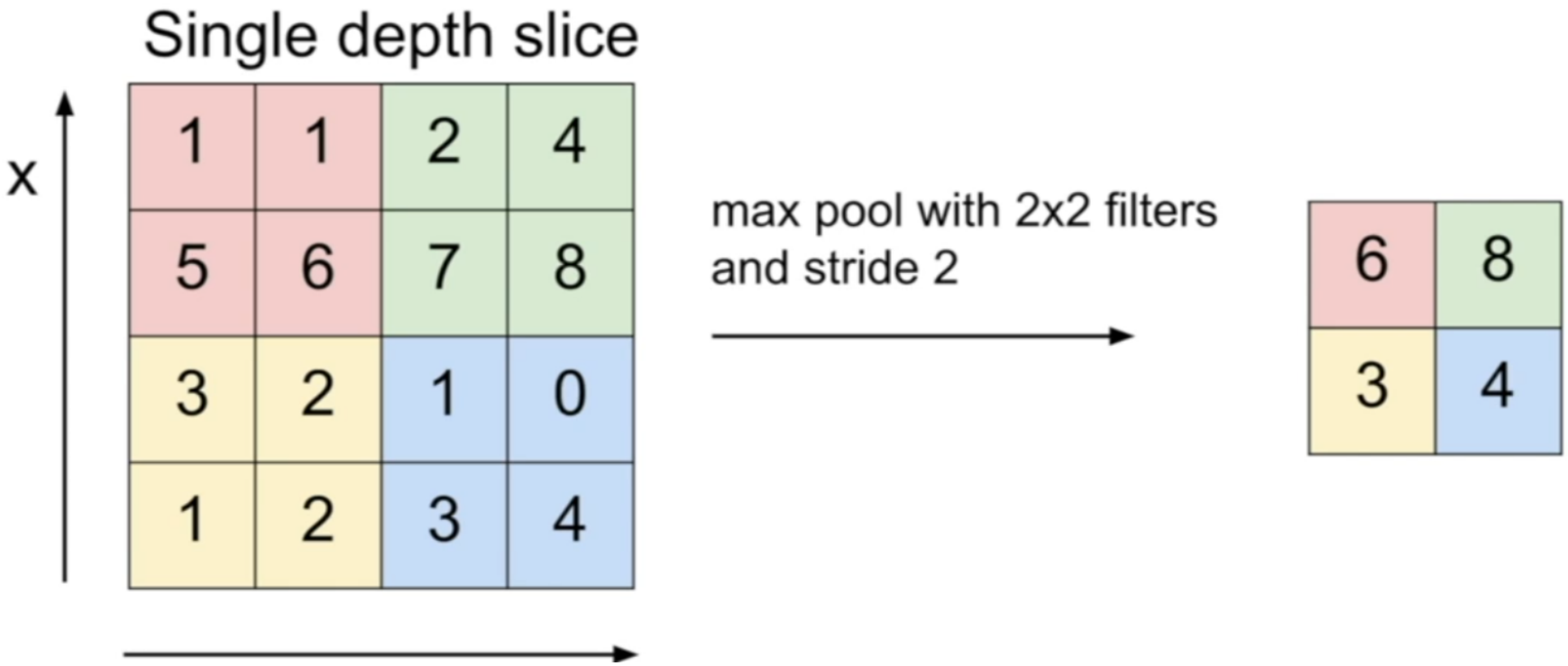


Figure: Max-pooling

CNN

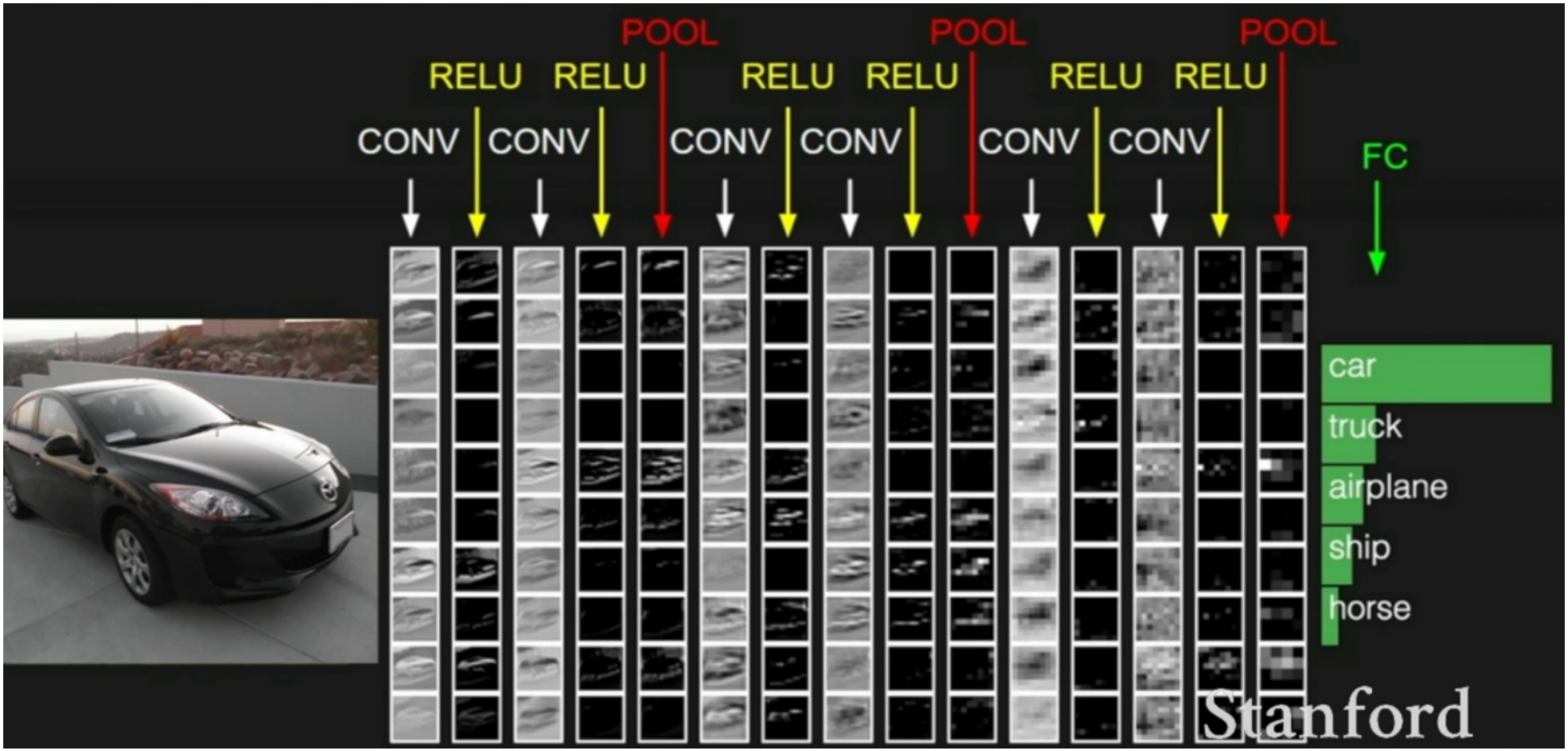


Figure: Fully Connected Layer

Comparing

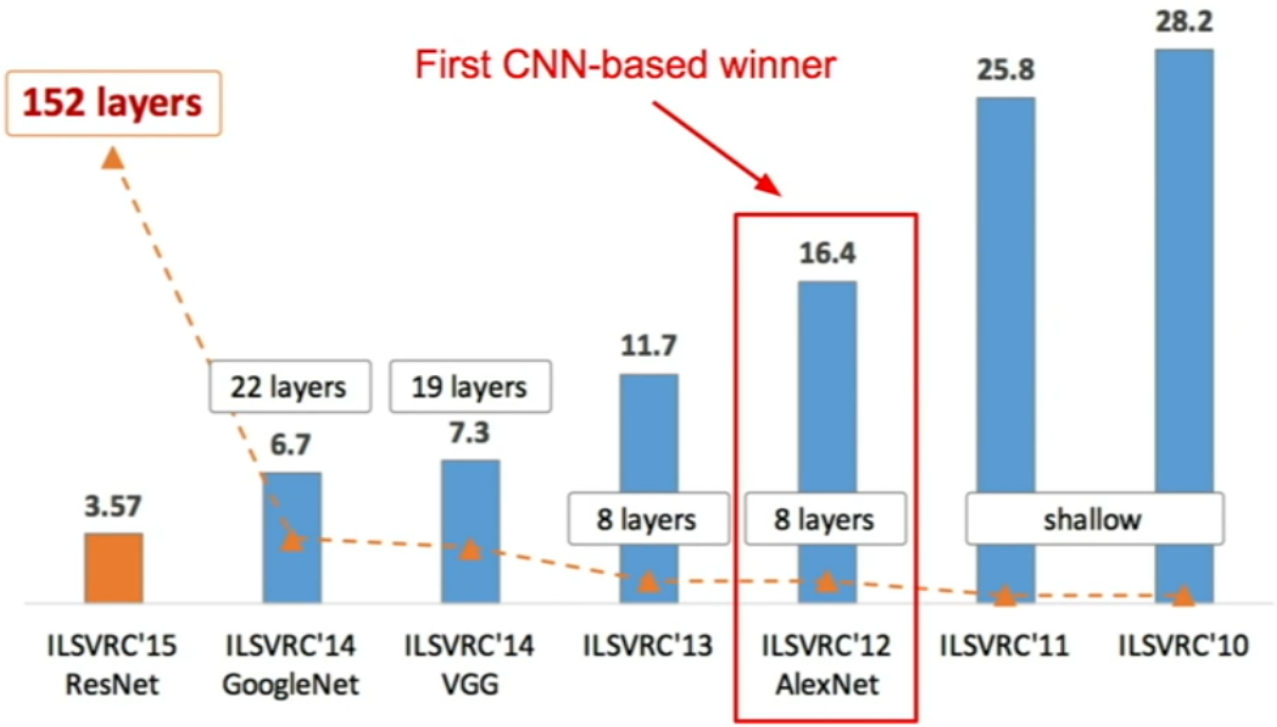


Figure: Comparing Structures

AlexNet

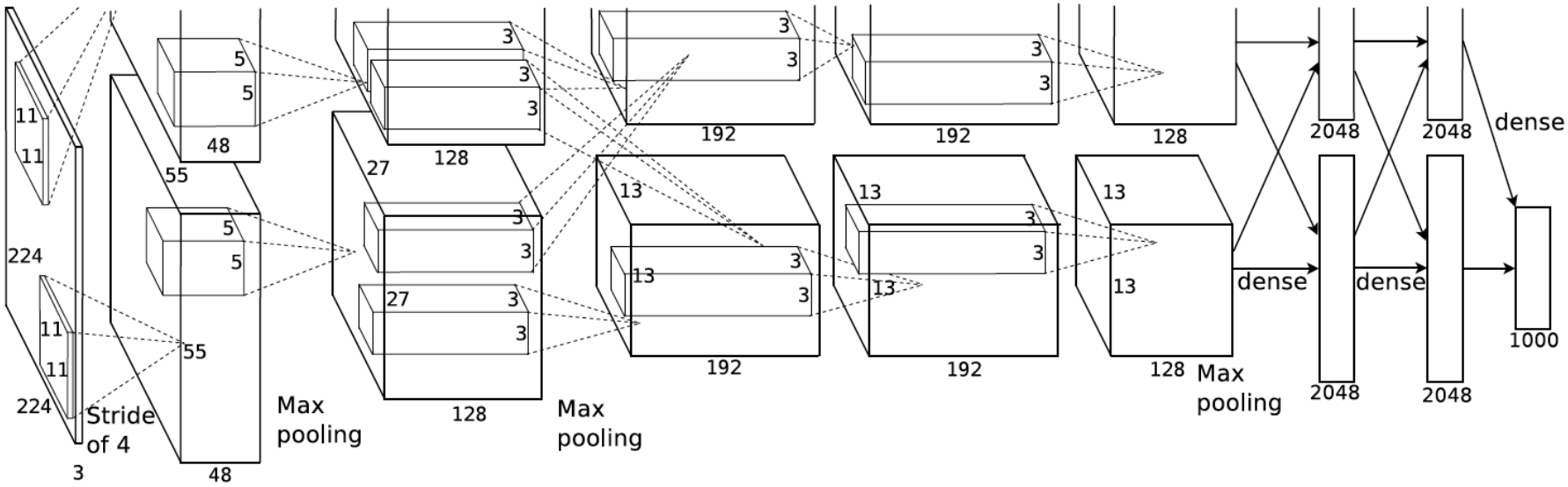


Figure: AlexNet architecture

AlexNet

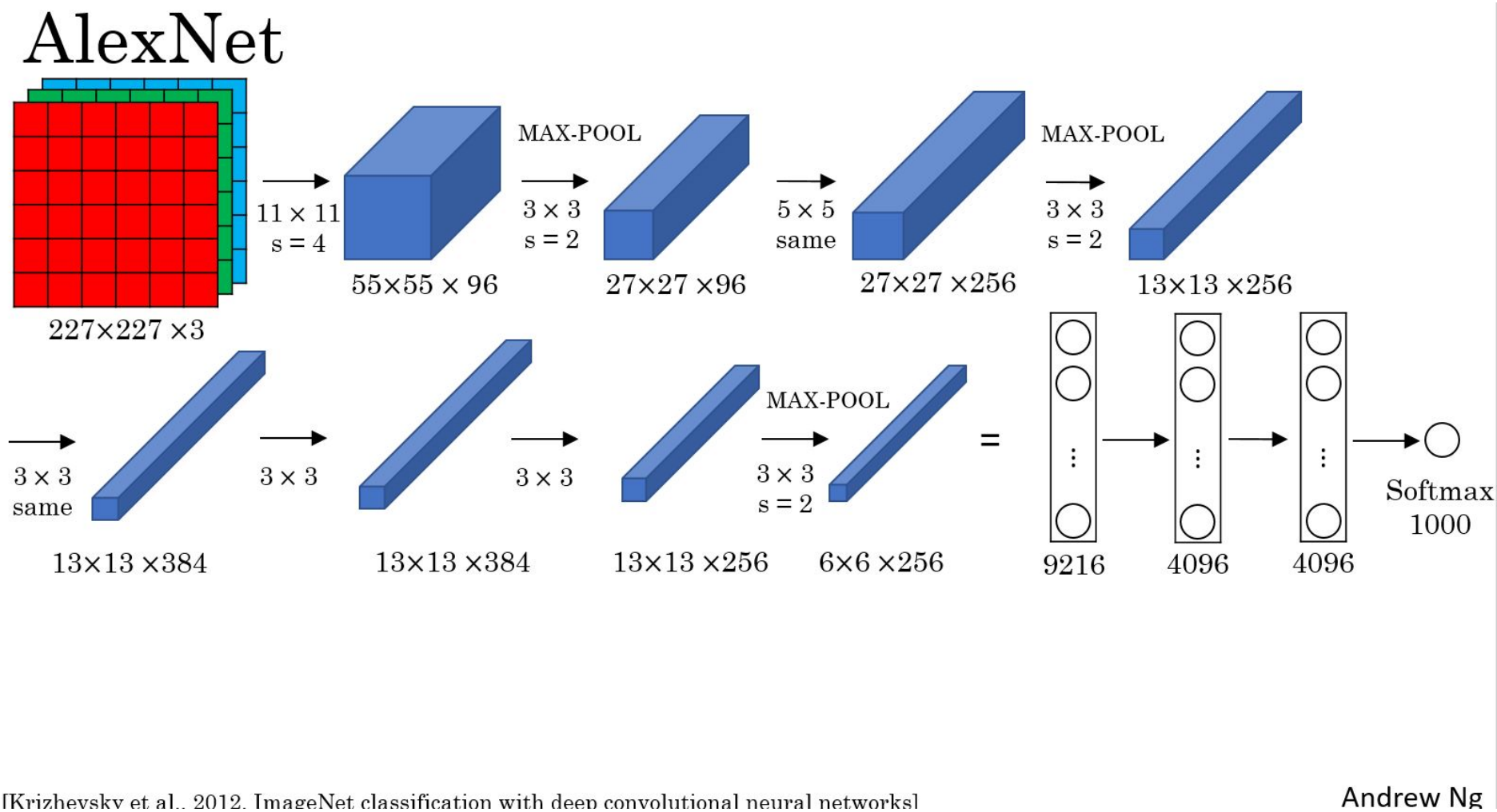


Figure: AlexNet architecture

Comparing

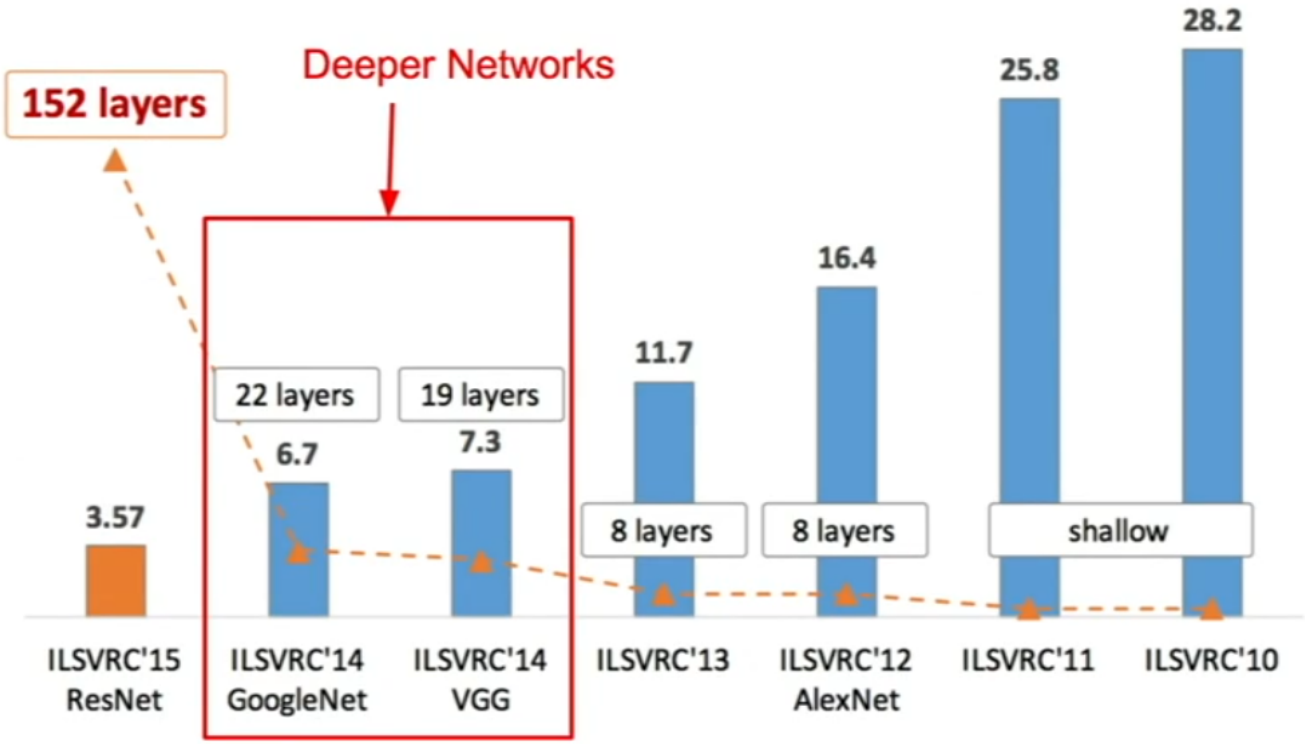
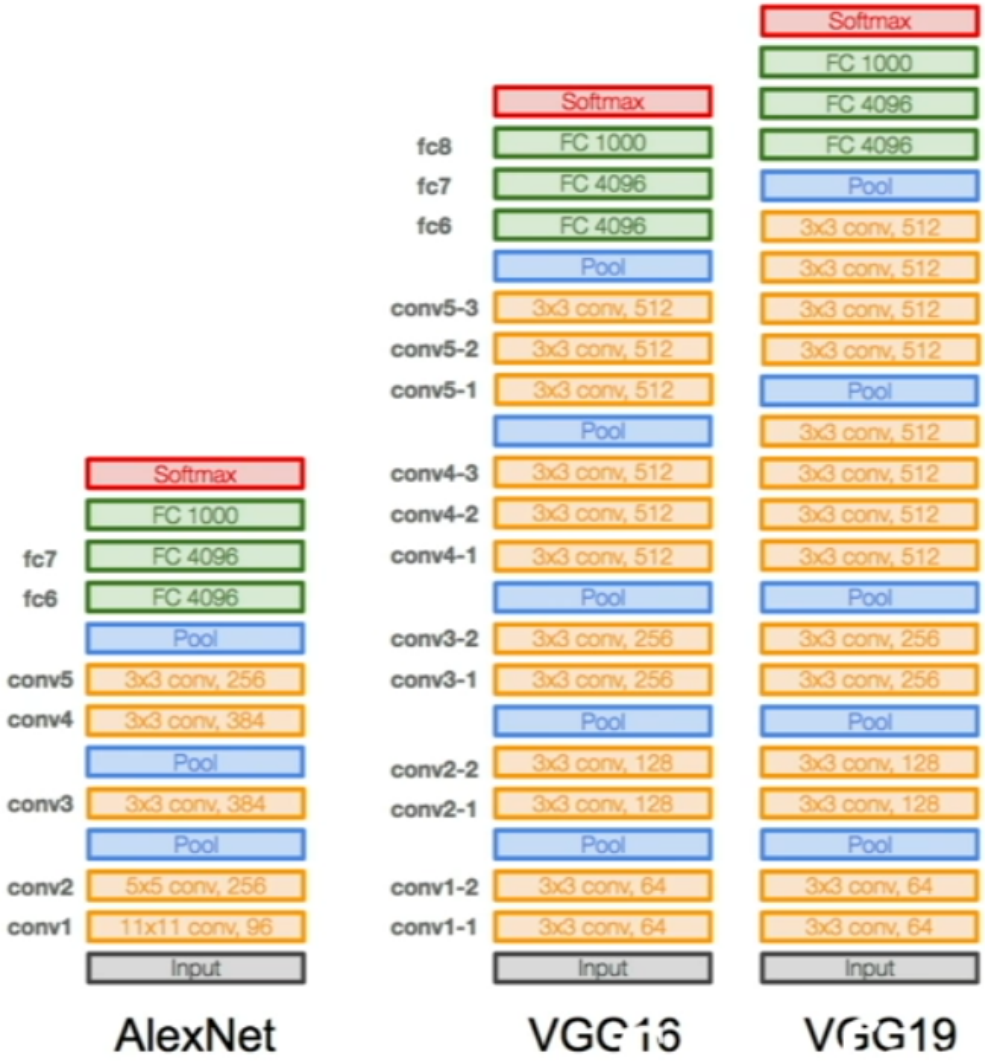


Figure: Comparing Structures

VGG

Deeper Networks, Smaller Filters



Comparing

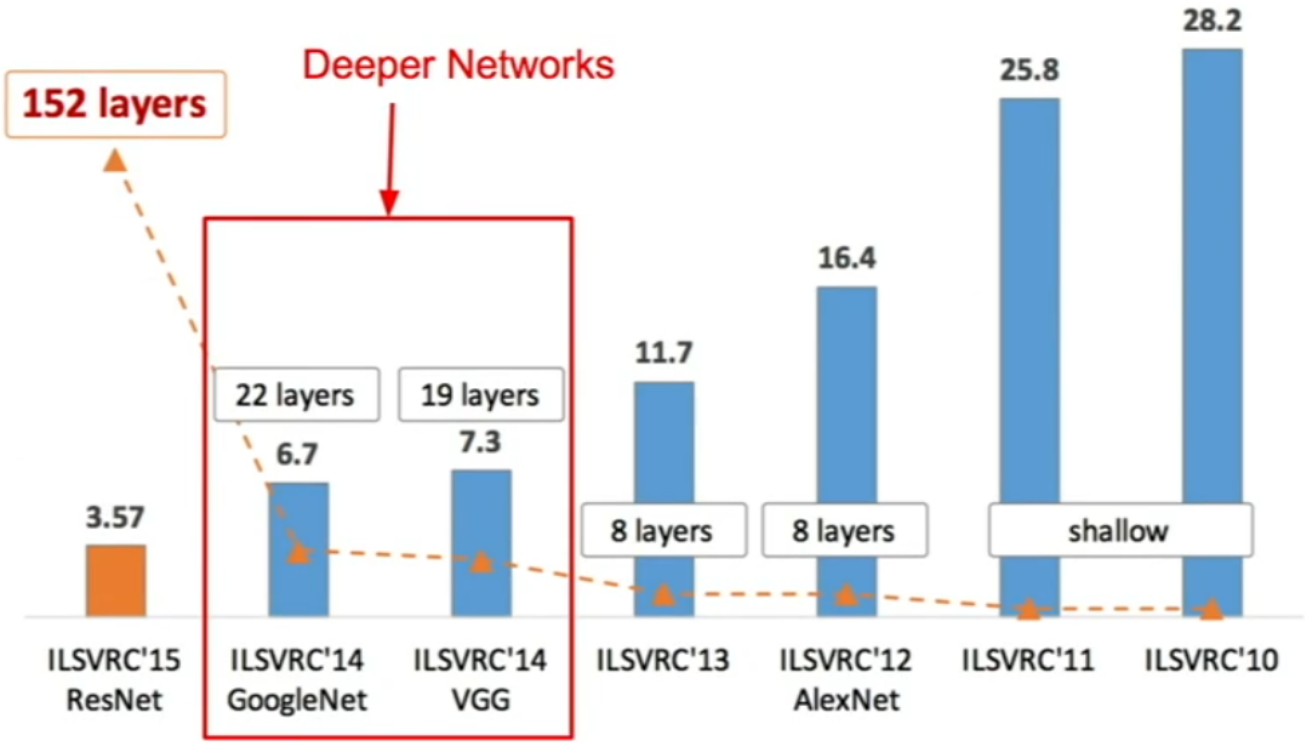
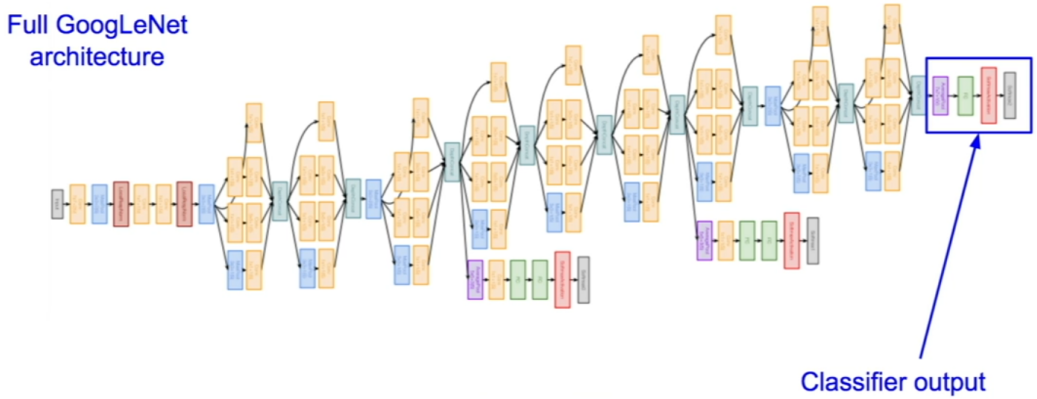
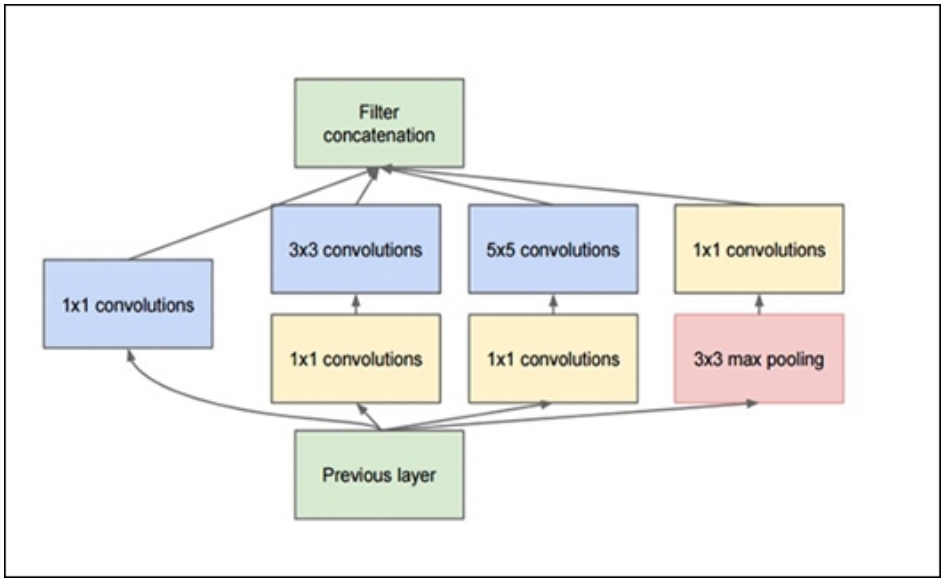


Figure: Comparing Structures

GoogLeNet

Make deeper networks computationally inexpensive





Comparing

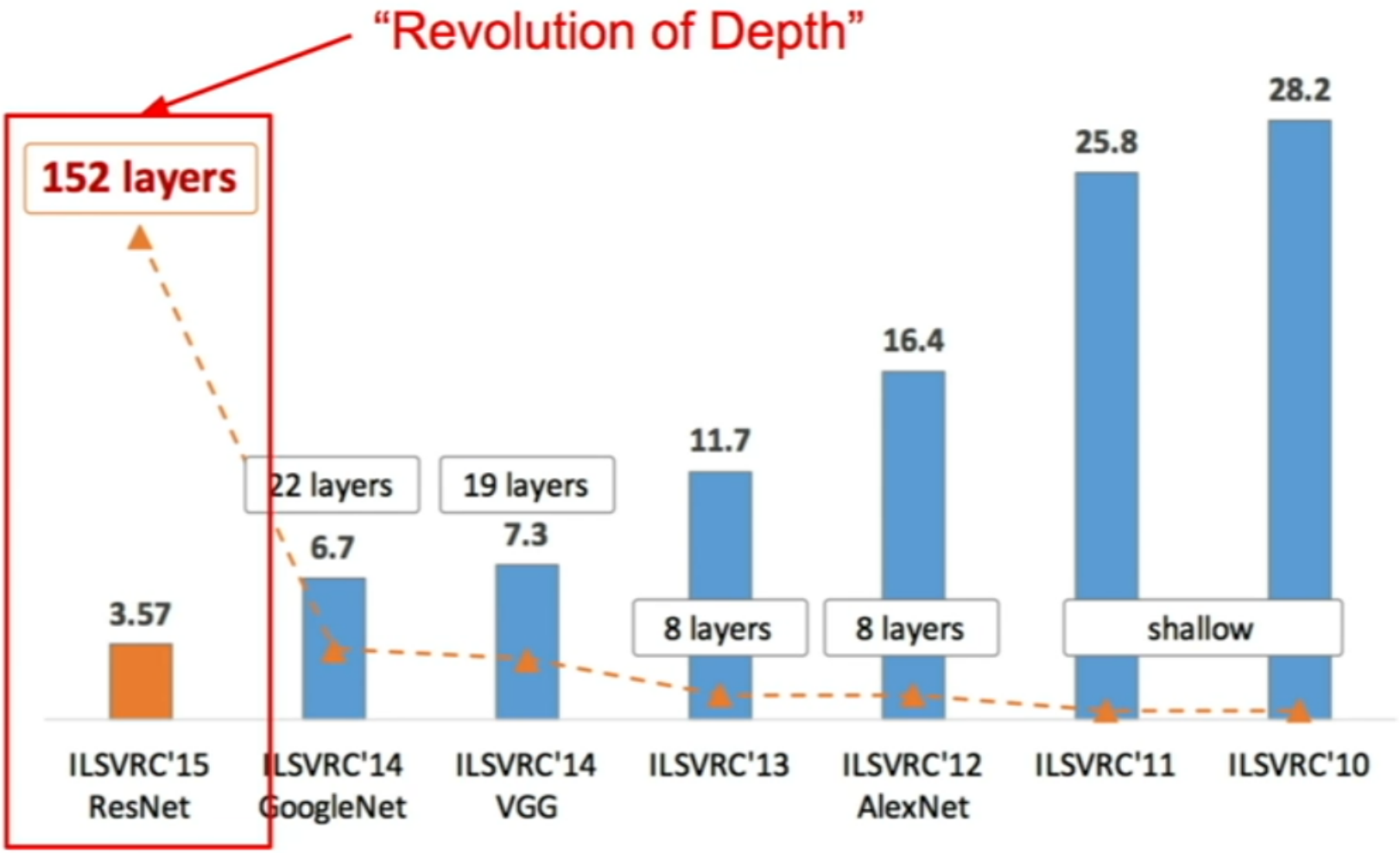
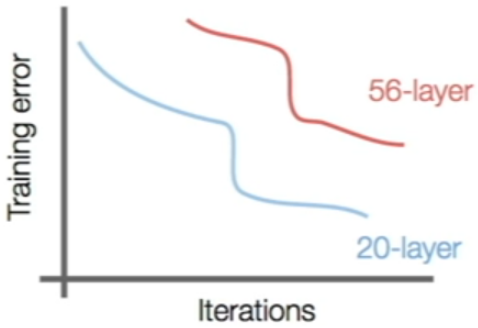
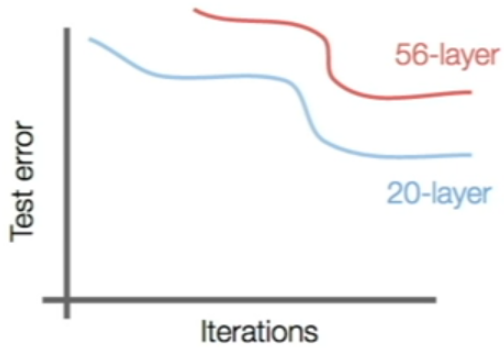


Figure: Comparing Structures

ResNet

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?





ResNet

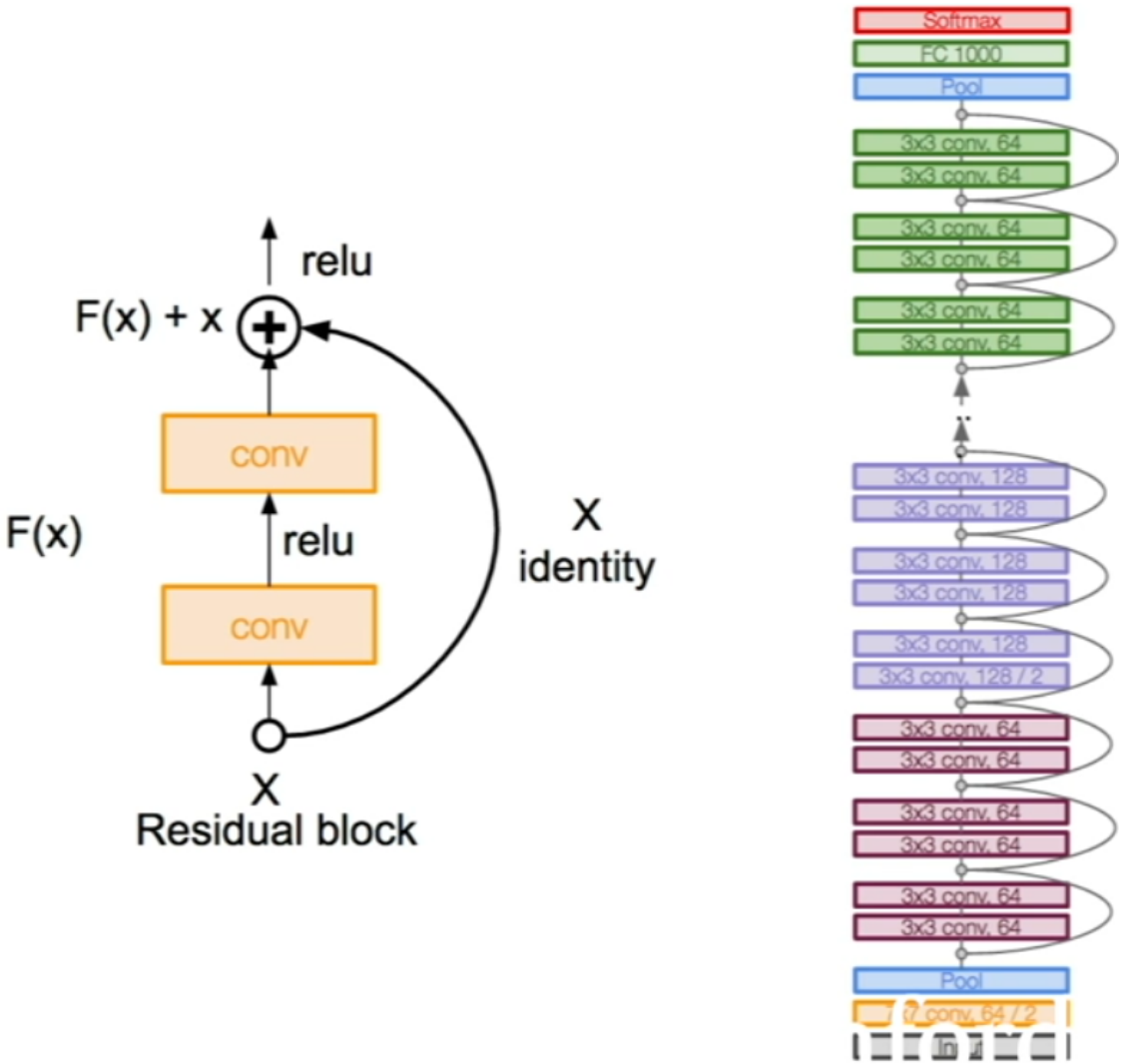


Figure: ResNet Architecture

Comparing

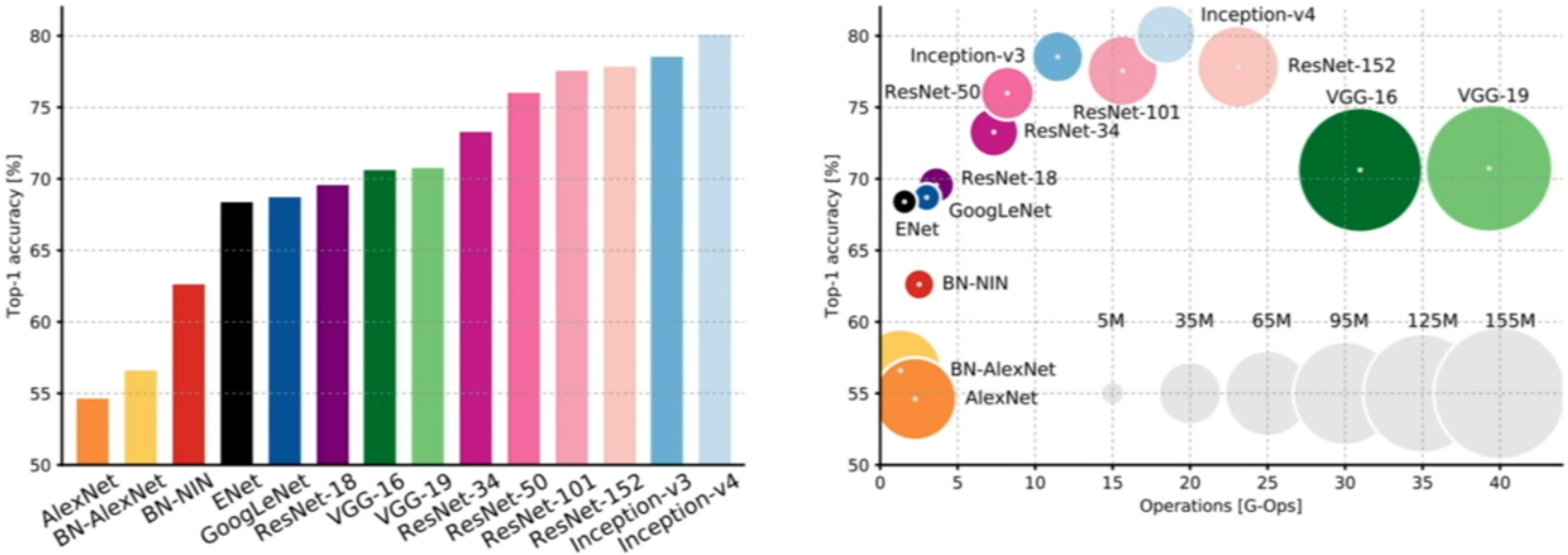


Figure: Comparison

Example

ConvNetJS CIFAR-10 demo

Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.

Training Stats

pause

Forward time per example: 12ms
Backprop time per example: 28ms
Classification loss: 1.38449
L2 Weight decay loss: 0.04078
Training accuracy: 0.51
Validation accuracy: 0.43
Examples seen: 54991

Learning rate: 0.01 change

Momentum: 0.9 change

Batch size: 4 change

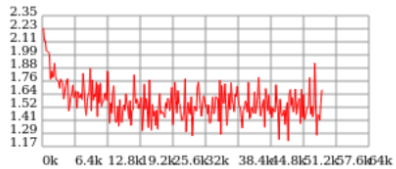
Weight decay: 0.0001 change

save network snapshot as JSON

init network from JSON snapshot

load a pretrained network (achieves ~80% accuracy)

Loss:

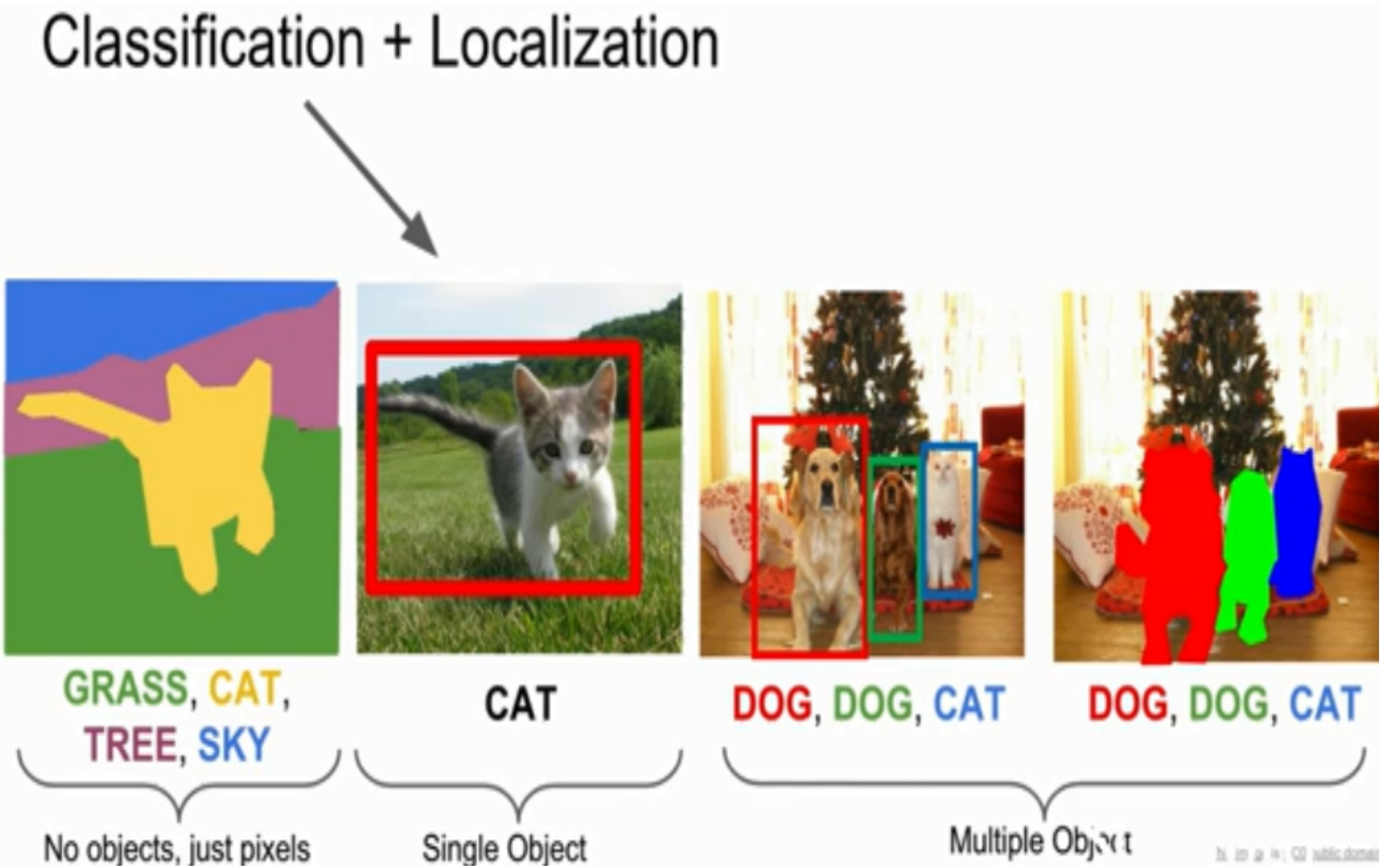


clear graph

Instantiate a Network and Trainer

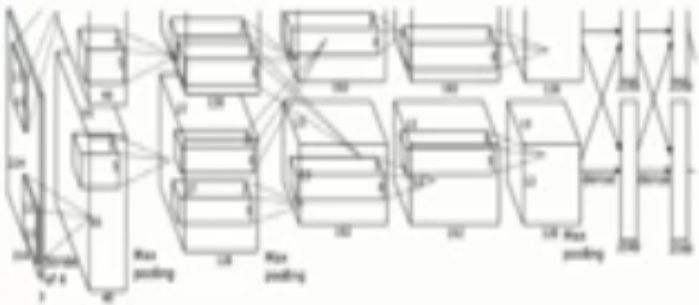
```
layer_defs = [];
layer_defs.push({type:'input', out_sx:32, out_sy:32, out_depth:3});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:20, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:20, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);
```

Computer Vision Tasks



Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

Example

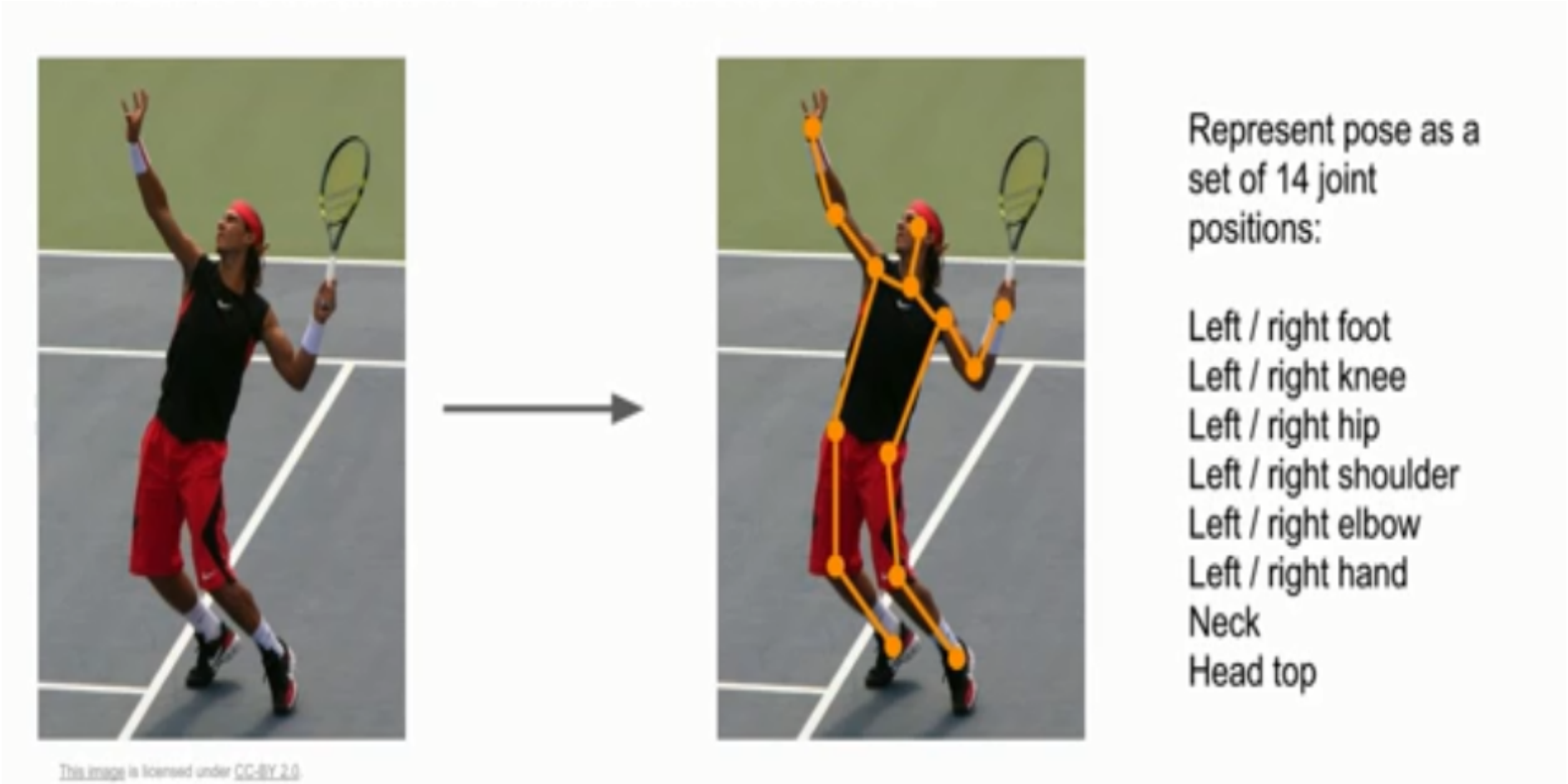
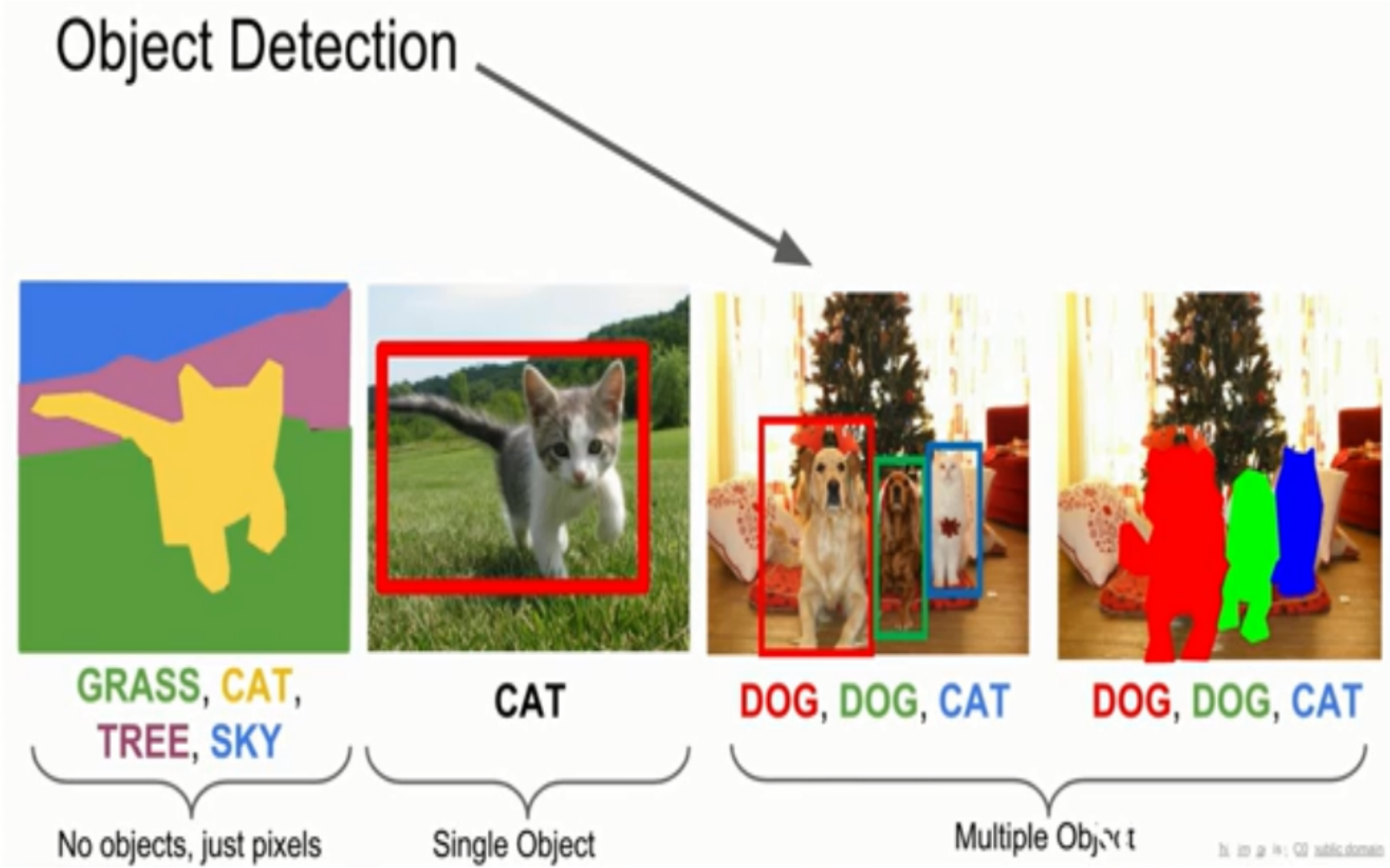


Figure: Body pose estimation

Fancier Task



Object Detection

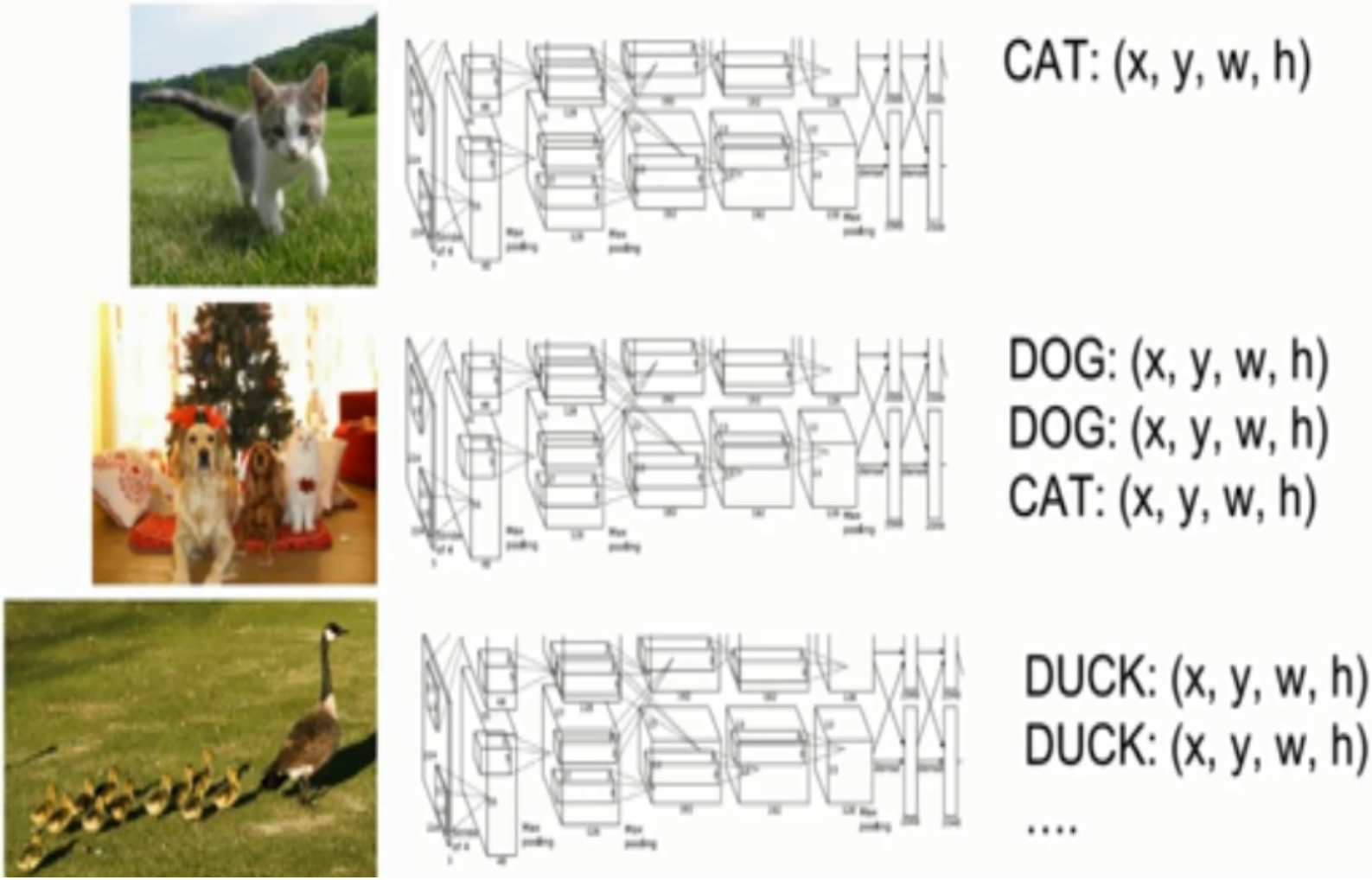
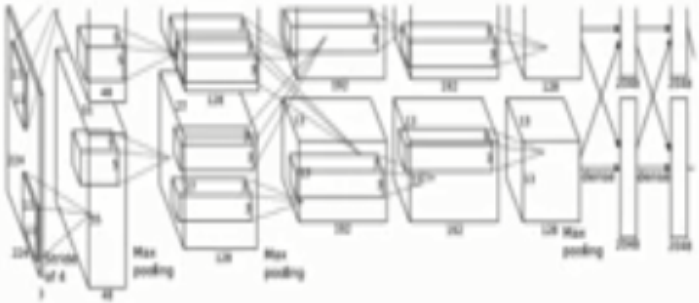


Figure: New approach is needed

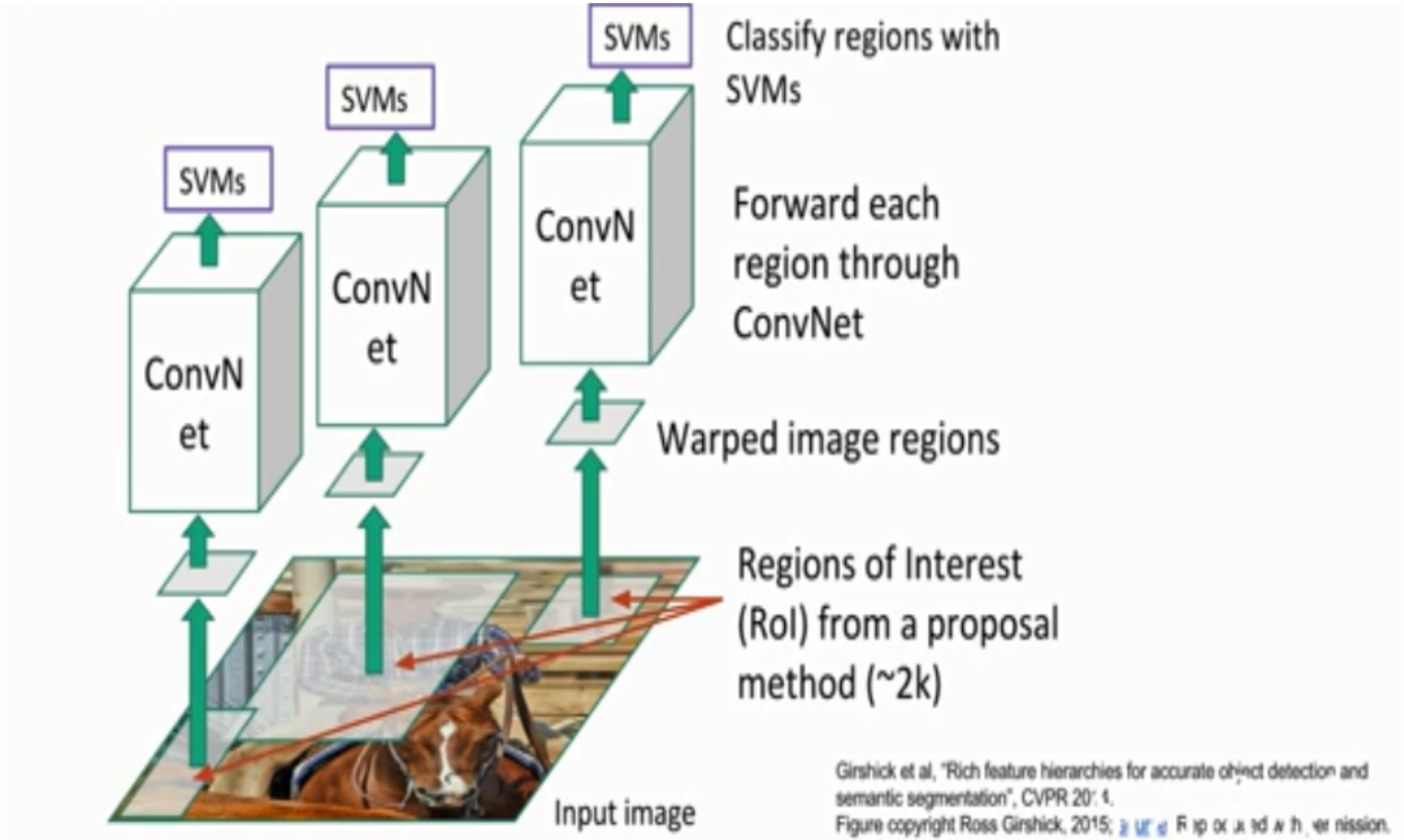
Object Detection

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



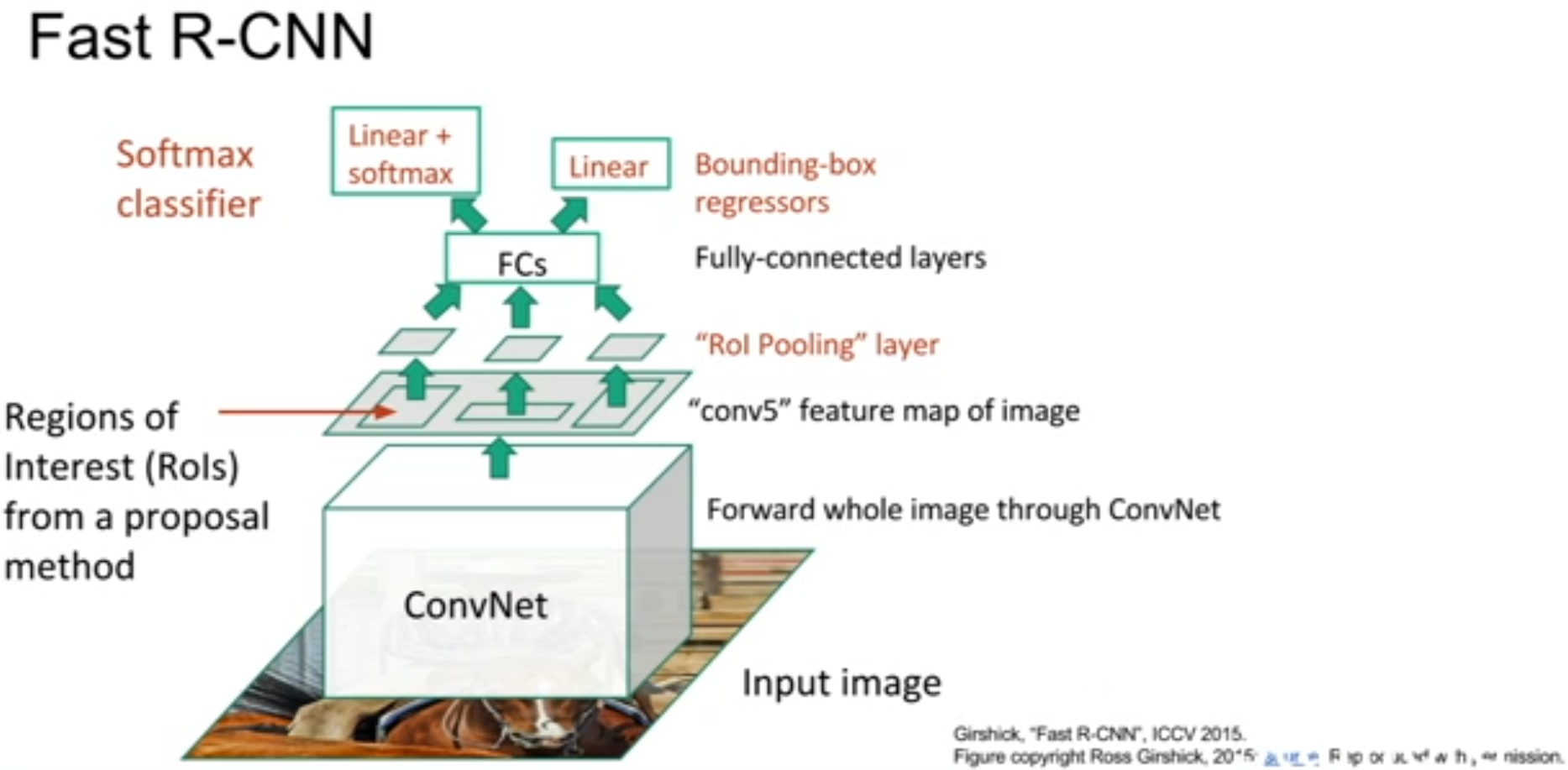
Dog? NO
Cat? NO
Background? YES

RCNN

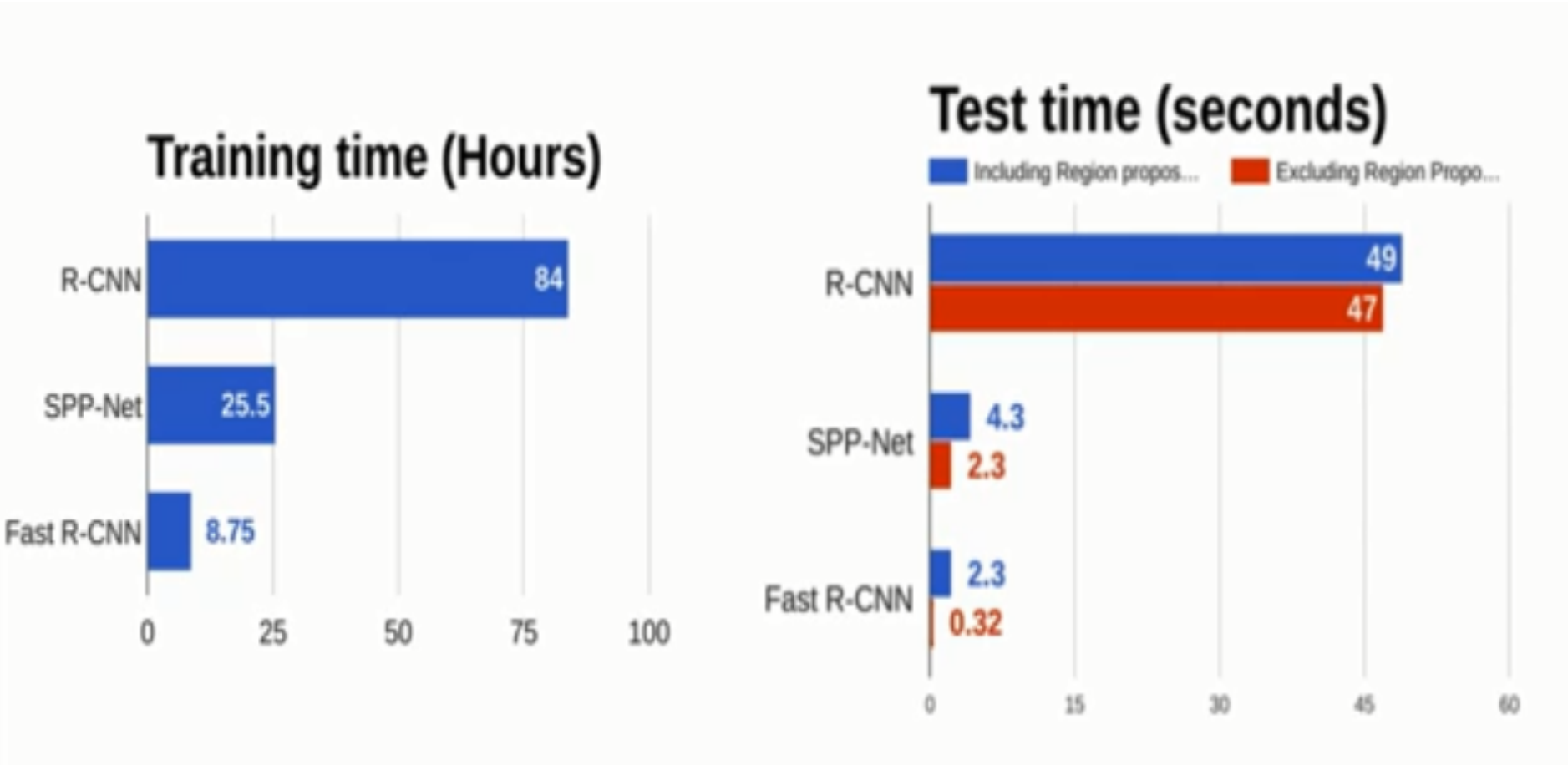


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 20[illegible].
Figure copyright Ross Girshick, 2015; [illegible] Reproduced with permission.

Fast RCNN


Fast R-CNN
Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image
Girshick, "Fast R-CNN", ICCV 2015.

Comparing


Training time (Hours)
R-CNN
84
SPP-Net
25.5
Fast R-CNN
8.75
0
25
50
75
100
Test time (seconds)
Including Region propos...
Excluding Region Propo...
R-CNN
49
47
SPP-Net
4.3
2.3
Fast R-CNN
2.3
0.32
0
15
30
45
60

Even Faster

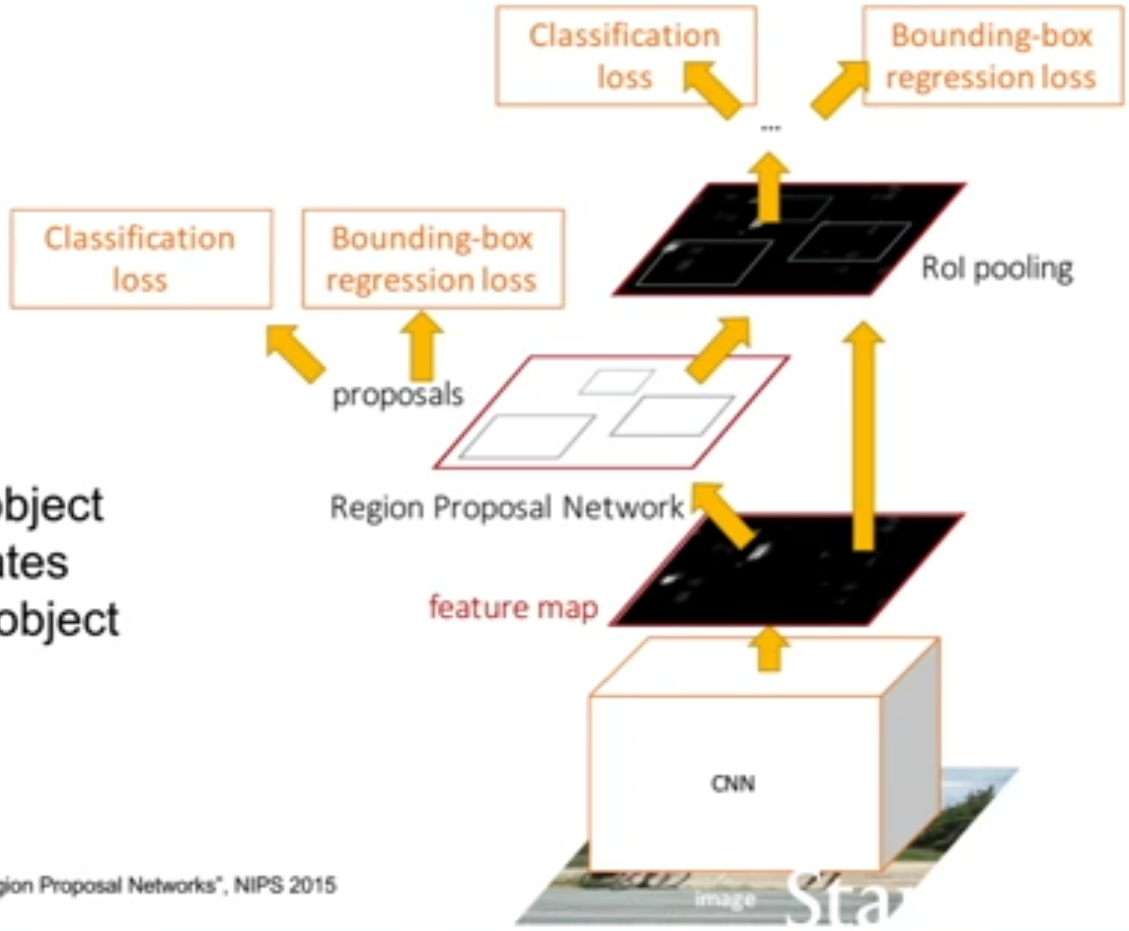
Faster R-CNN:

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

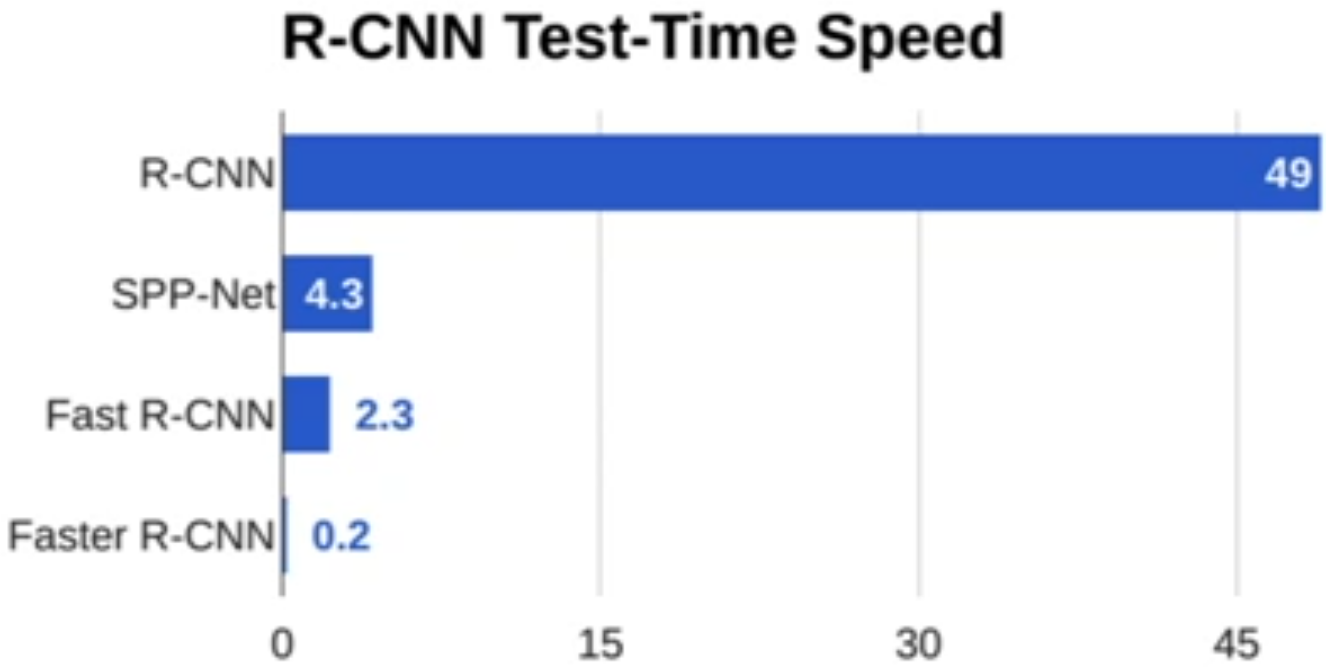


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

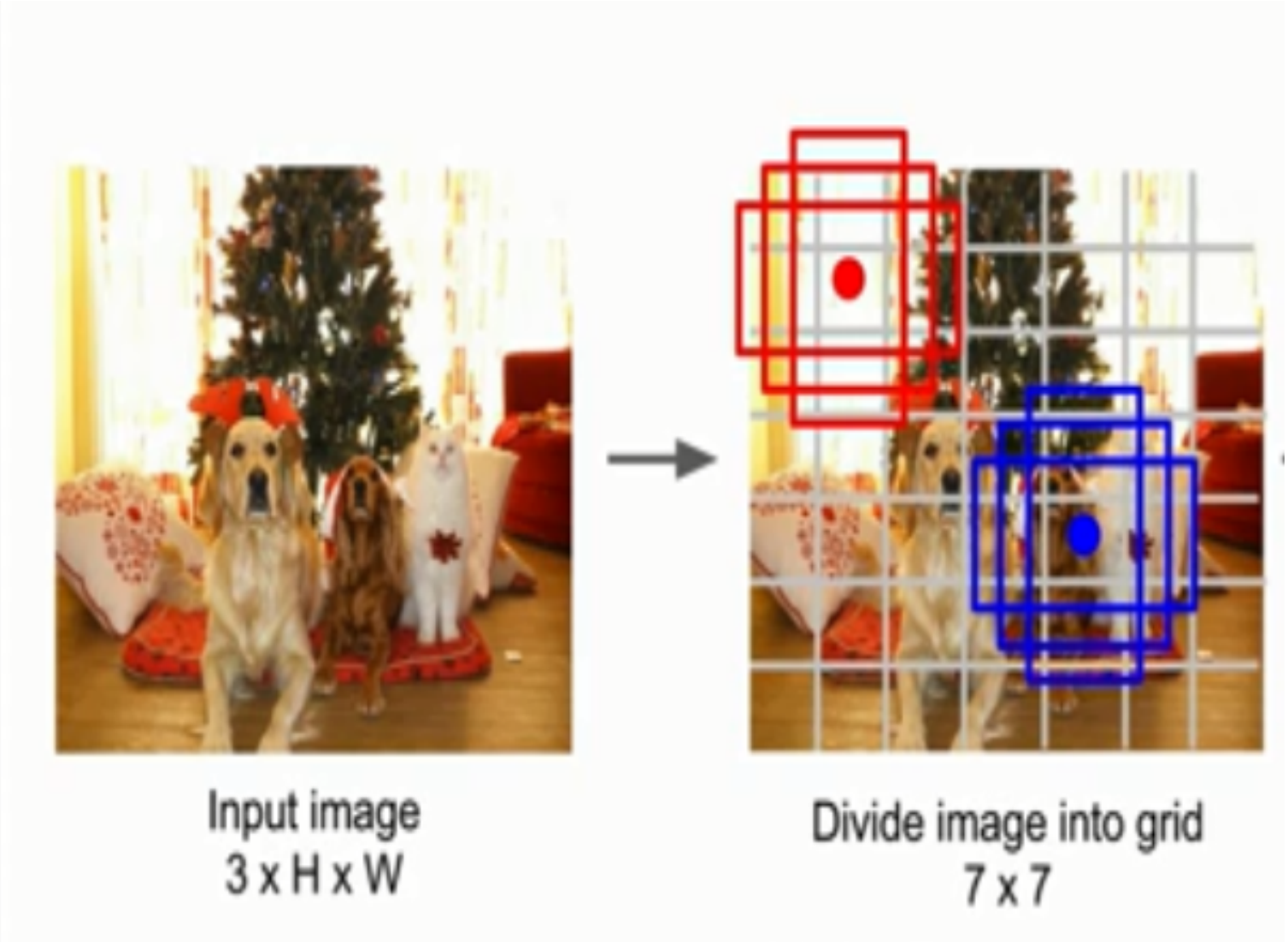
Even Faster

Faster R-CNN:

Make CNN do proposals!



Yolo



References

References:

- Krizhevsky, Alex & Sutskever, Ilya & E. Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.
- Very Deep Convolutional Networks for Large-Scale Image Recognition Karen Simonyan, Andrew Zisserman (Submitted on 4 Sep 2014 (v1), last revised 10 Apr 2015 (this version, v6))
- Going Deeper with Convolutions Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich (Submitted on 17 Sep 2014)
- Stanford CS231n 2017

Any Question?