Principles of Robot Autonomy I

Deep learning for computer vision

Guest Lecture by Dr. Boris Ivanovic

Stanford University

Autonomous Systems Lab

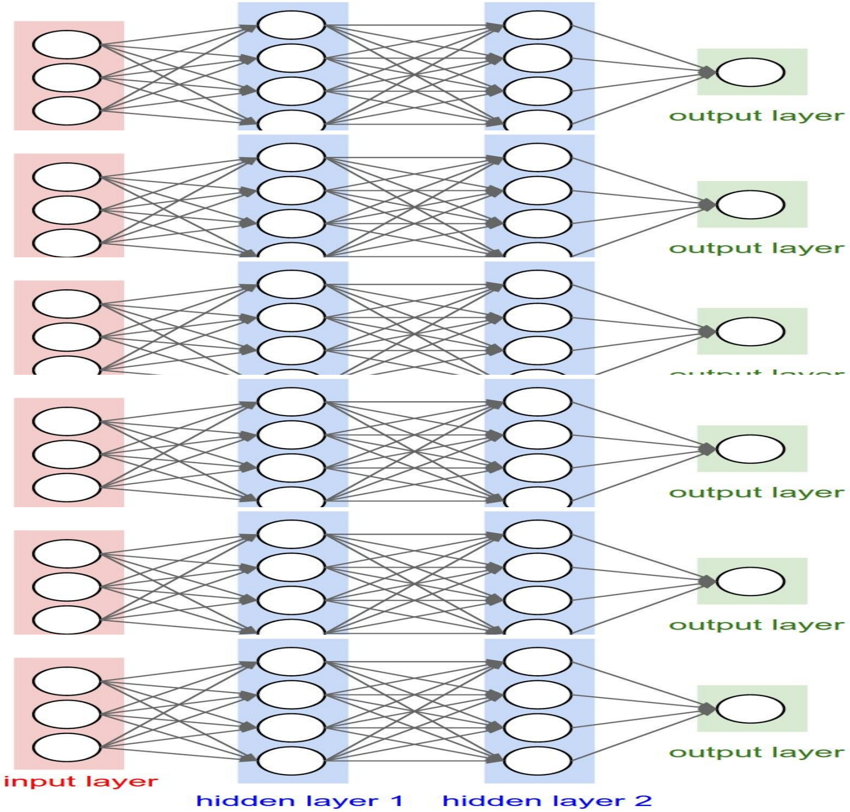
Today's itinerary

- Stats/ML review
- Neural network basics
- Convolutional neural networks
- Robotic applications

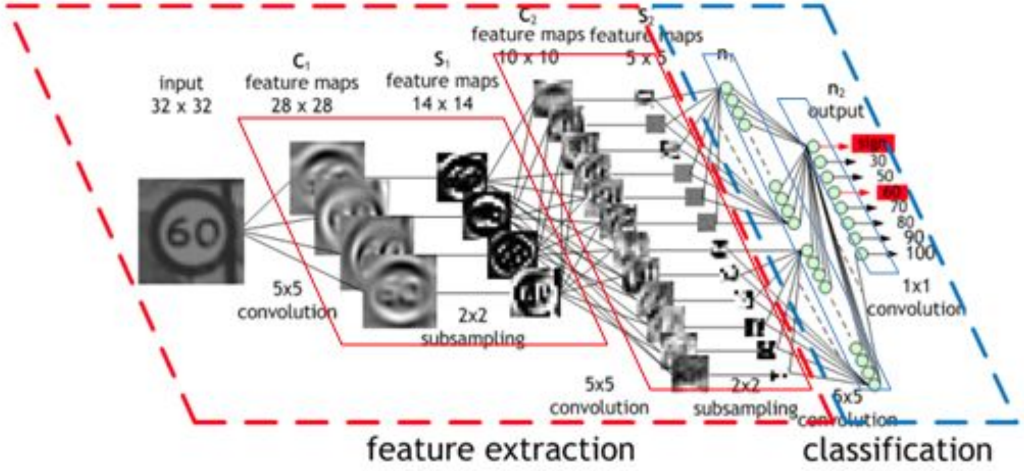
Efficient feature extraction

CIFAR-10
32x32x3

Inception-v3
299x299x3



vs.

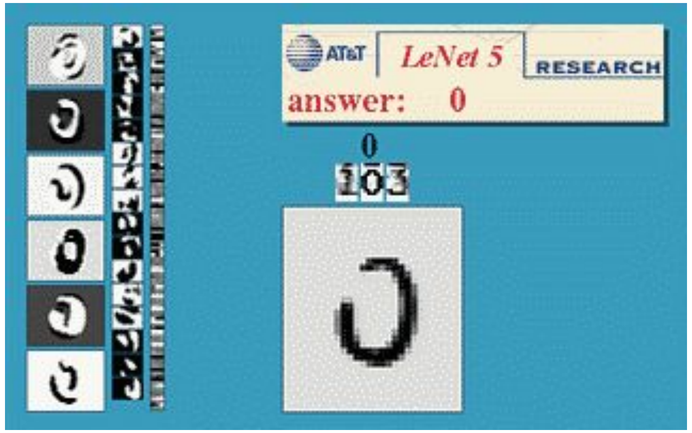


If we know the input is image data, we can assume some spatial locality
□ weight sharing

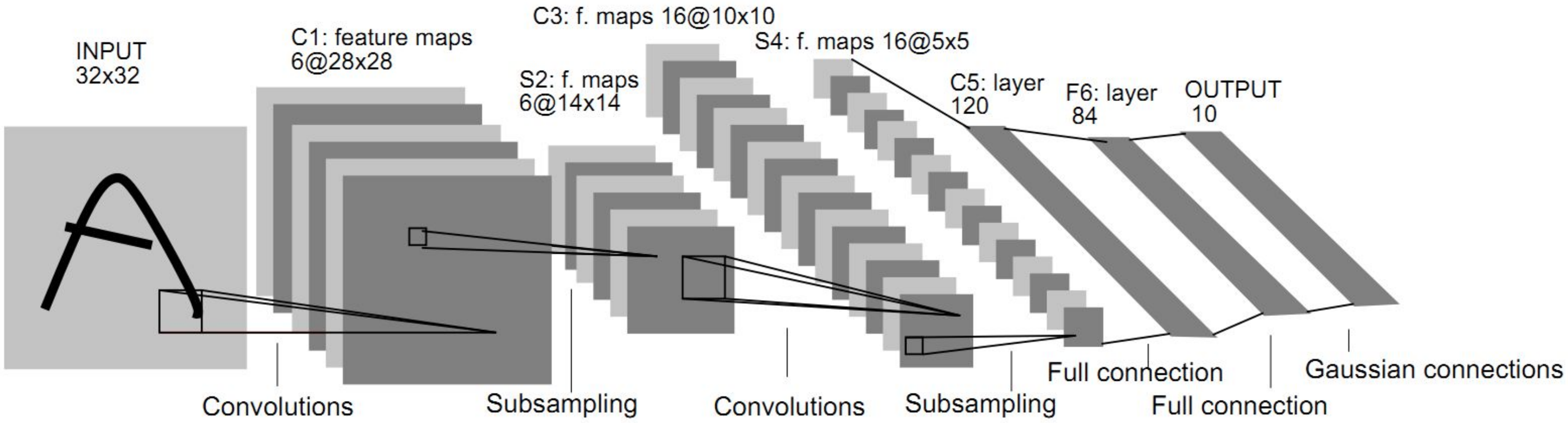
Convolutional neural networks (CNN)

Traditionally consist of 4 types of layers:

- Convolutional layers (CONV)
- Nonlinearity layers (RELU)
- Pooling layers (POOL)
- Fully-connected layers (FC)

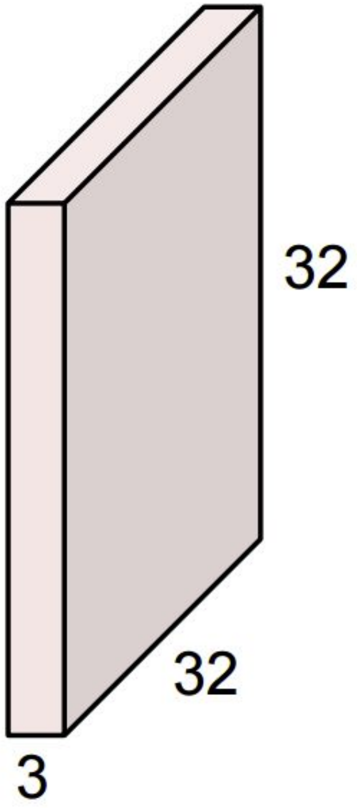


LeNet
(1998)

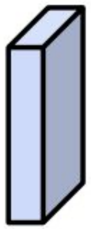


Convolution layer

32x32x3 image



5x5x3 filter

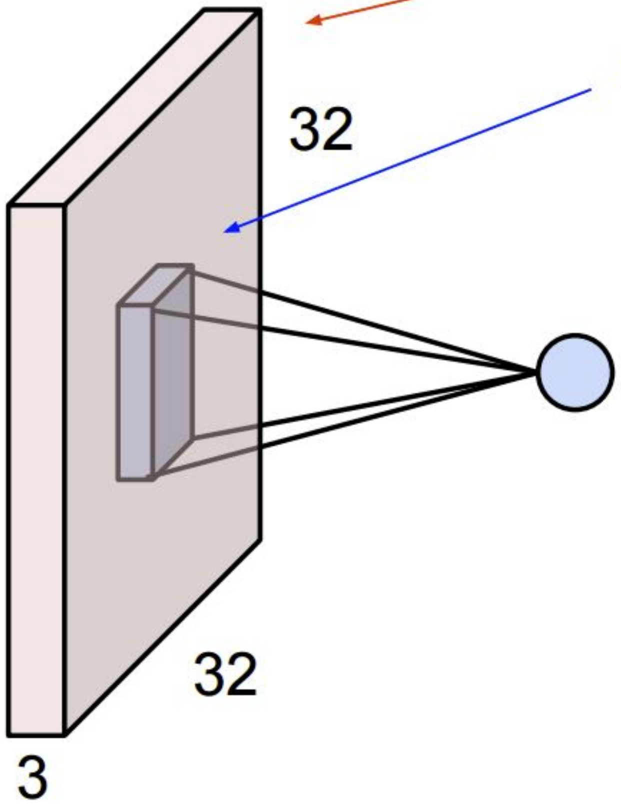


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution layer

32x32x3 image

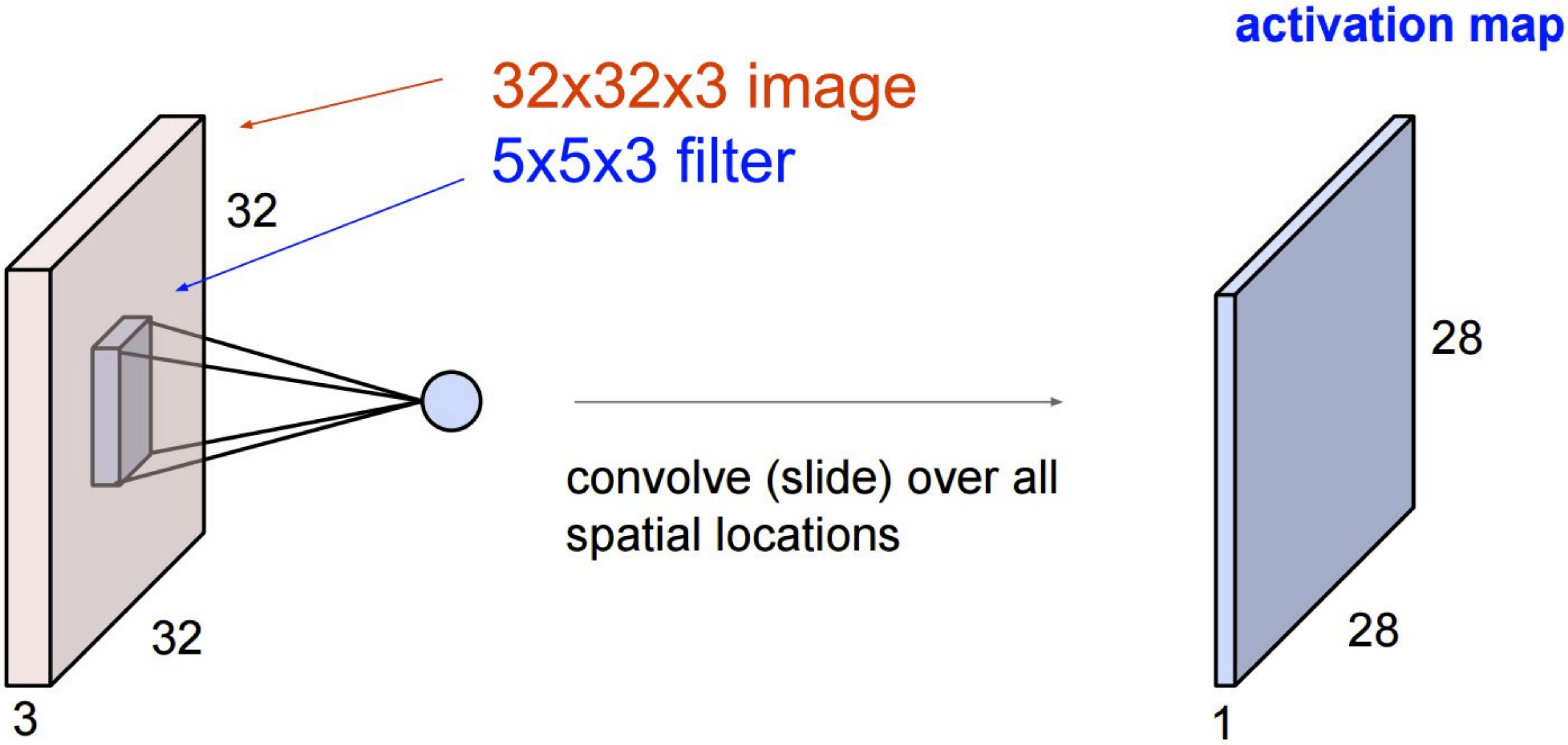
5x5x3 filter w



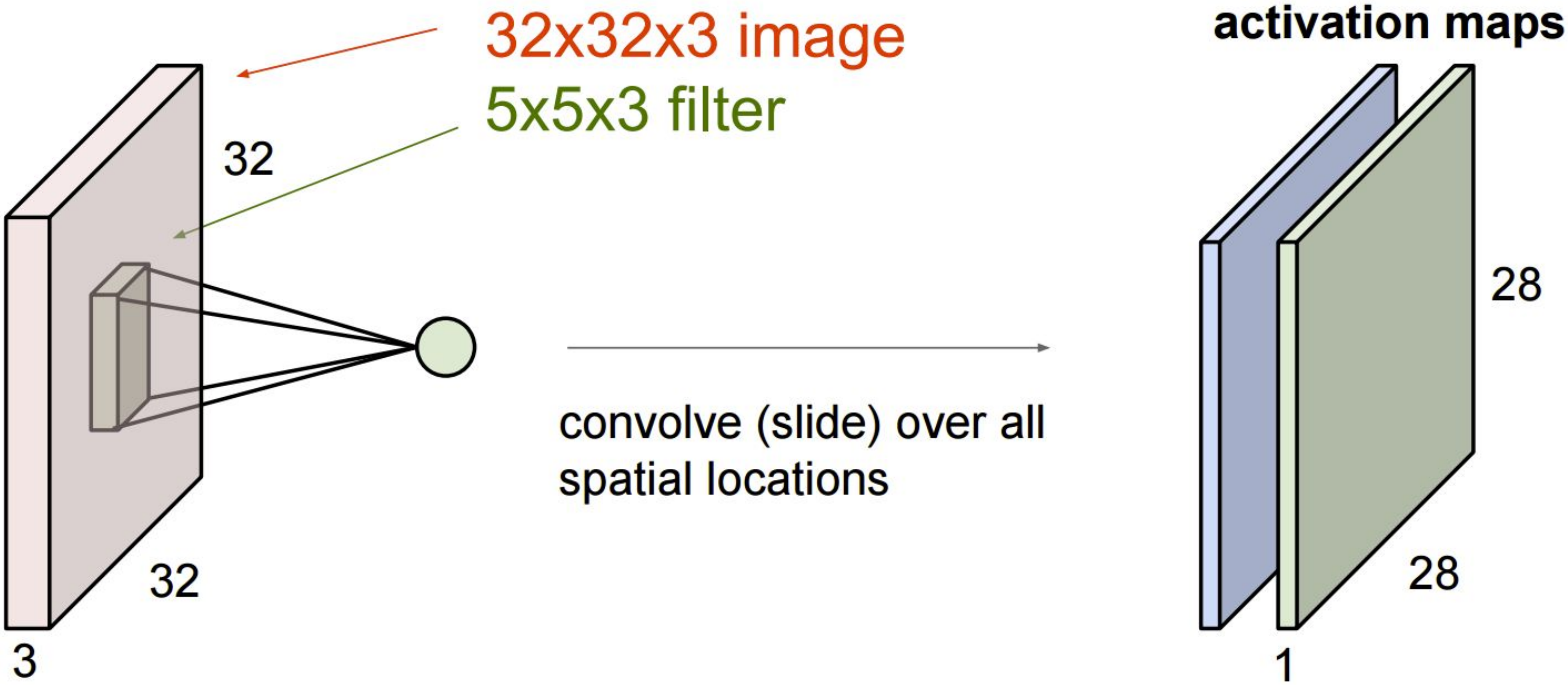
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution layer

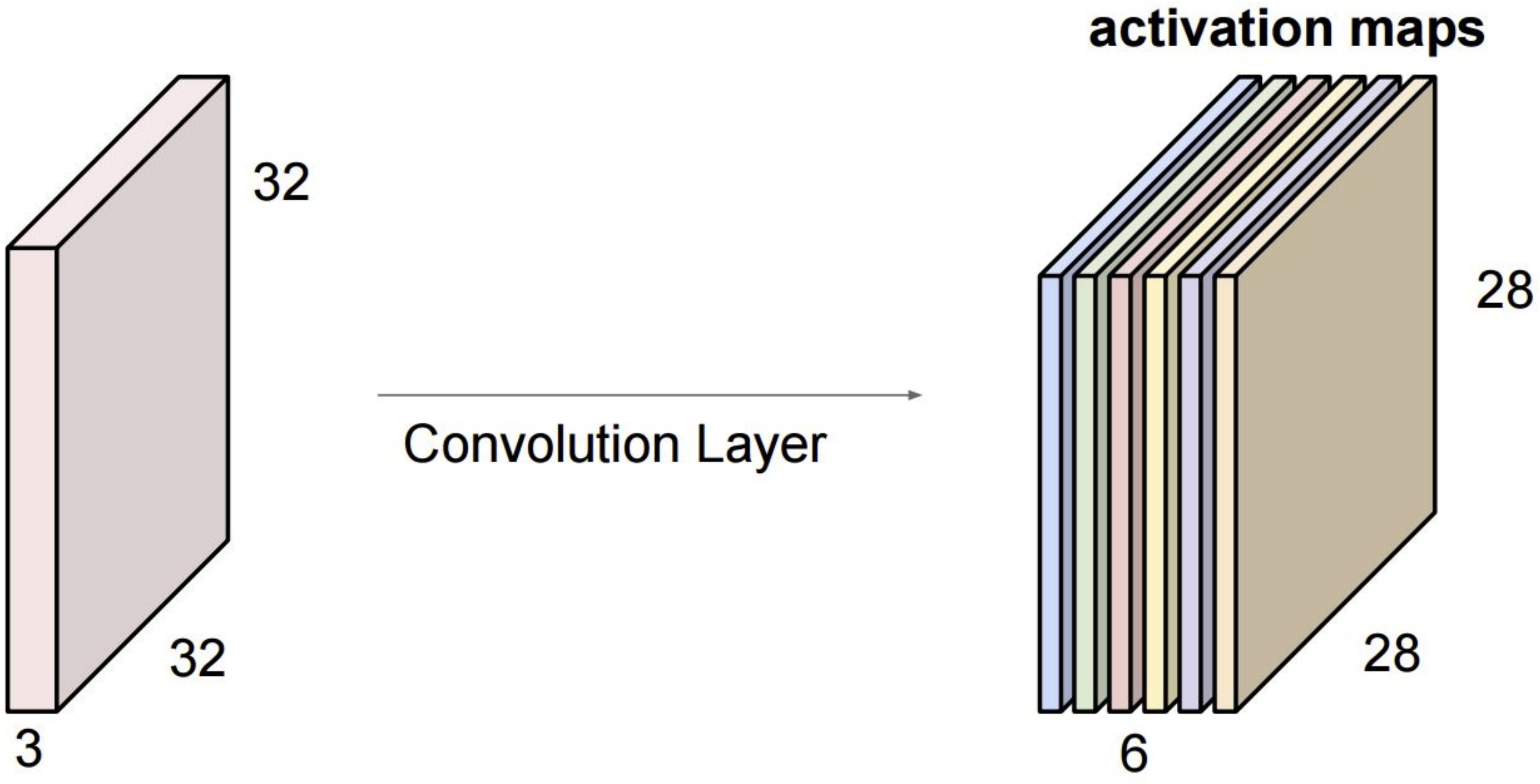


Convolution layer



Convolution layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

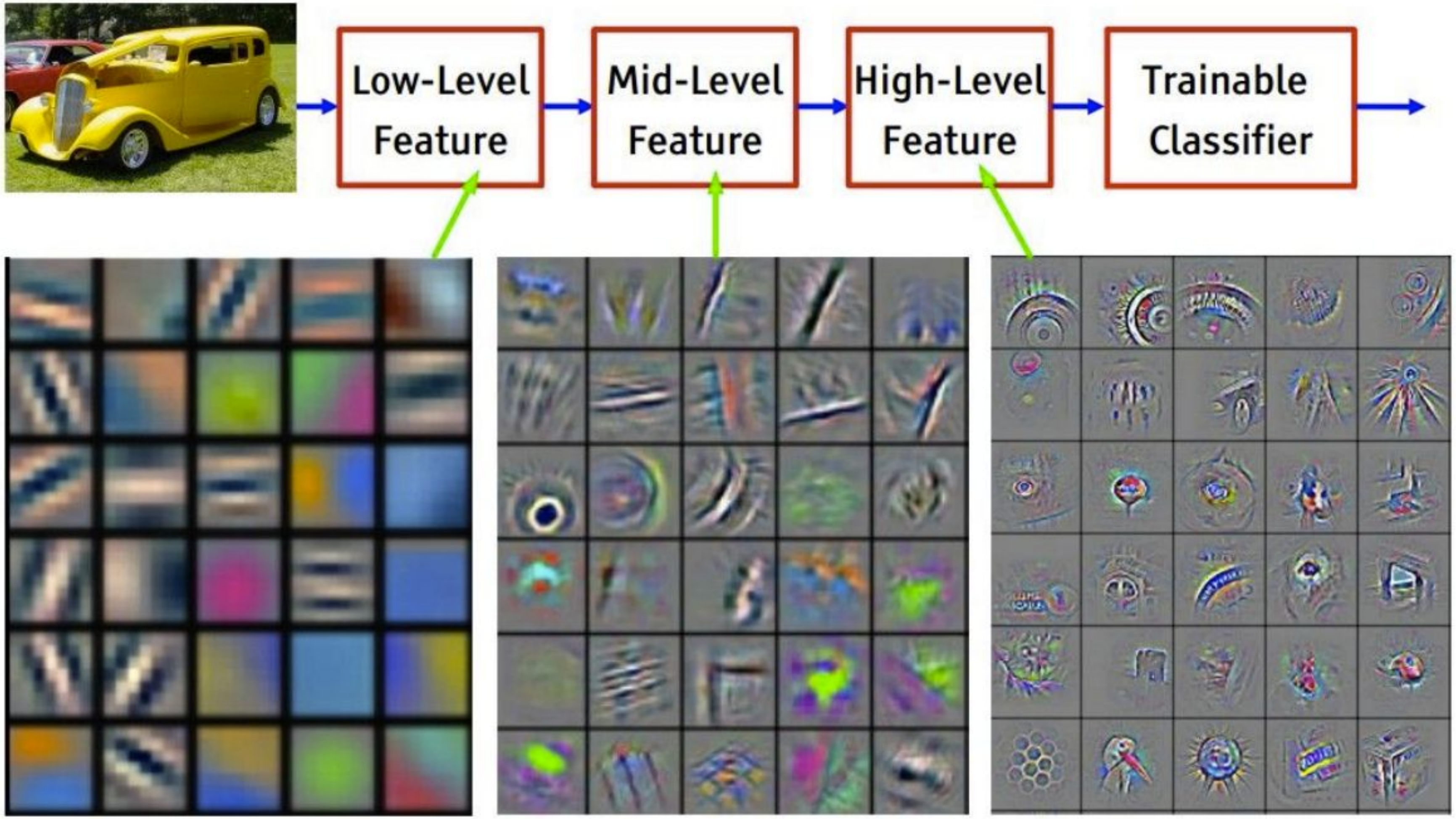


We stack these up to get a "new image" of size 28x28x6!

Convolution Layer Visualization

http://cs231n.github.io/convolutional-networks/

Feature hierarchy



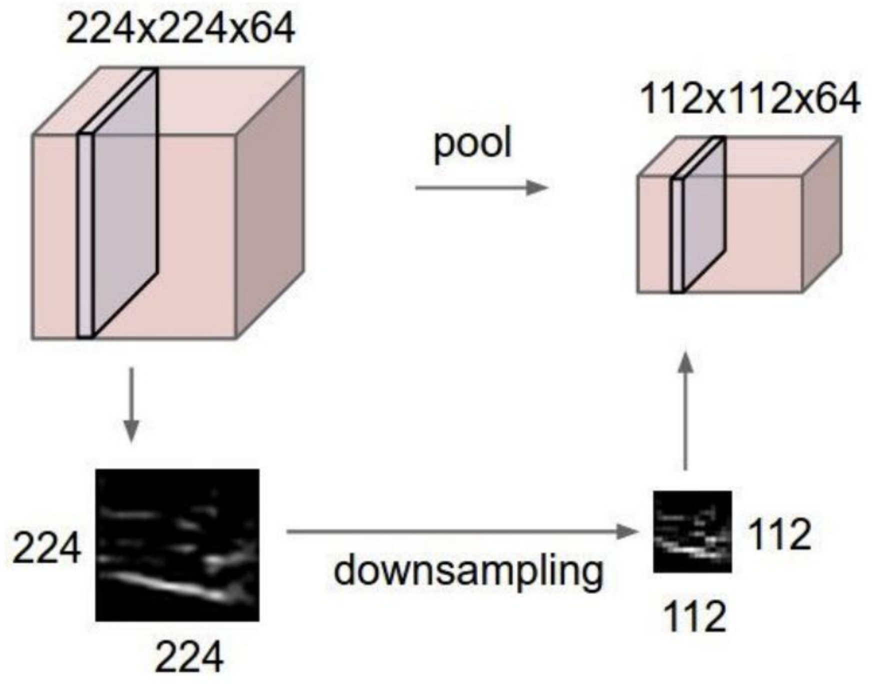
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

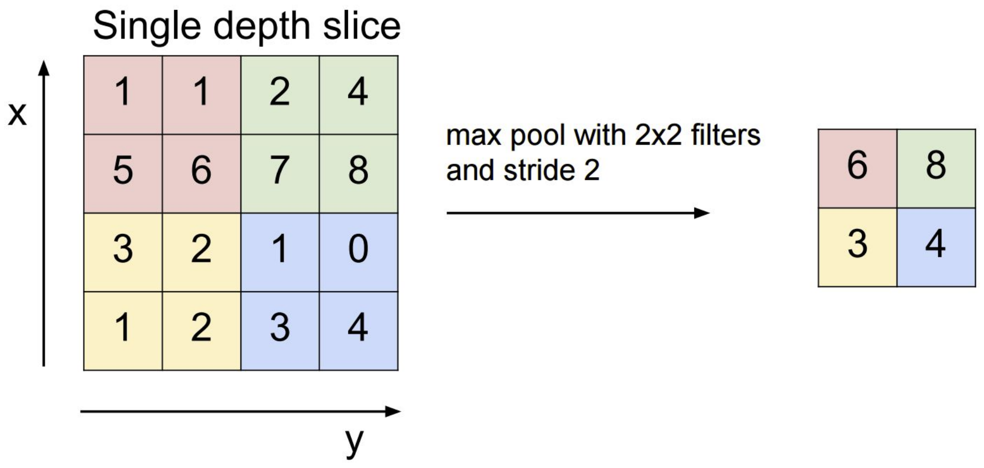
Pooling layer

As we move higher up the feature "food chain" we can save ourselves some computational effort by lowering the resolution

Types of pooling:

- MAX pooling
- MEAN pooling

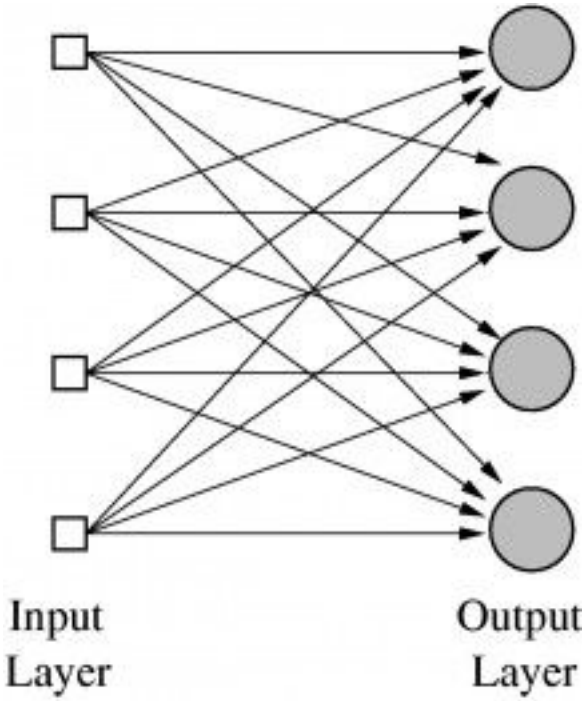




Fully connected layer

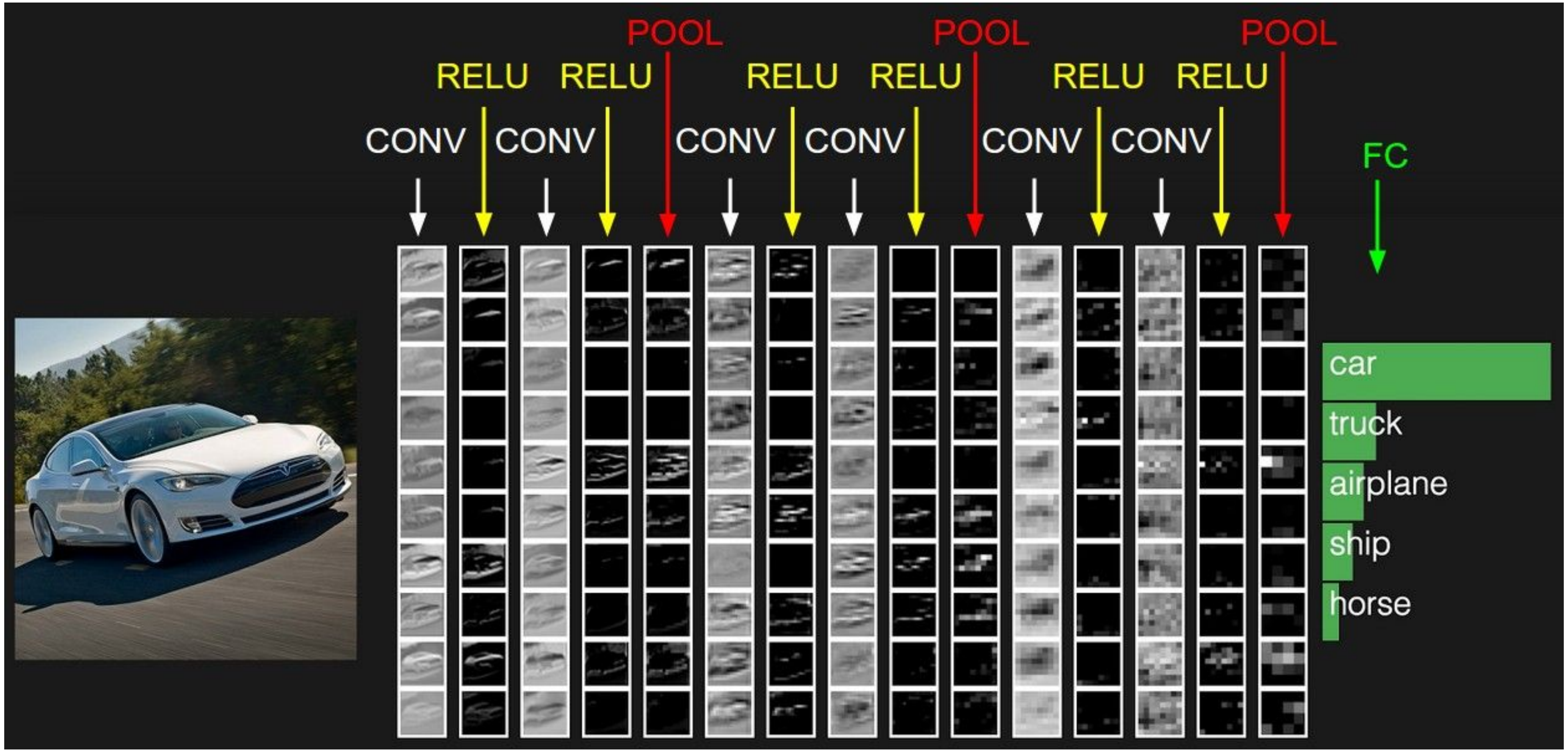
We’ve seen this one before!

Image “summary vector” with all of the redundant pixel info boiled out



Linear classifier (softmax)

Putting it all together – CNN



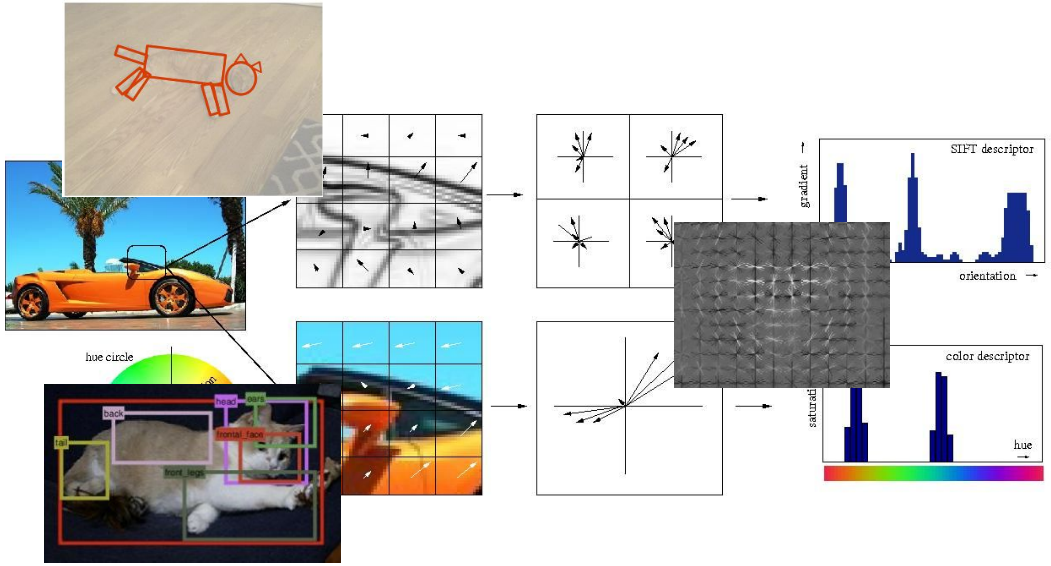
http://cs231n.stanford.edu/

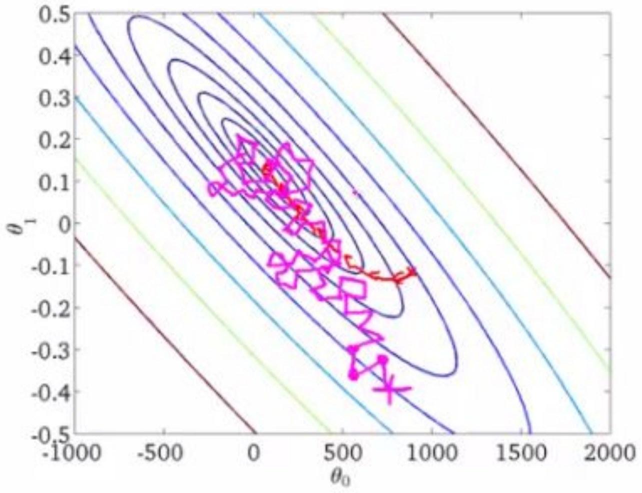
Live Demo - Inner Workings of a CNN

https://adamharley.com/nn_vis/cnn/3d.html

There's also a 2D version:
https://adamharley.com/nn_vis/cnn/2d.html

Classification showdown





vs.

$$\nabla(f \circ g)(x) = ((Dg)(x))^T (\nabla f)(g(x))$$

vector describing various image statistics

[32x32x3]

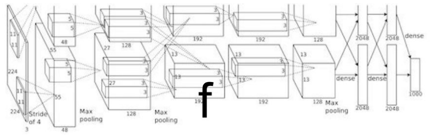
Feature Extraction

f

training

10 numbers, indicating class scores

[32x32x3]



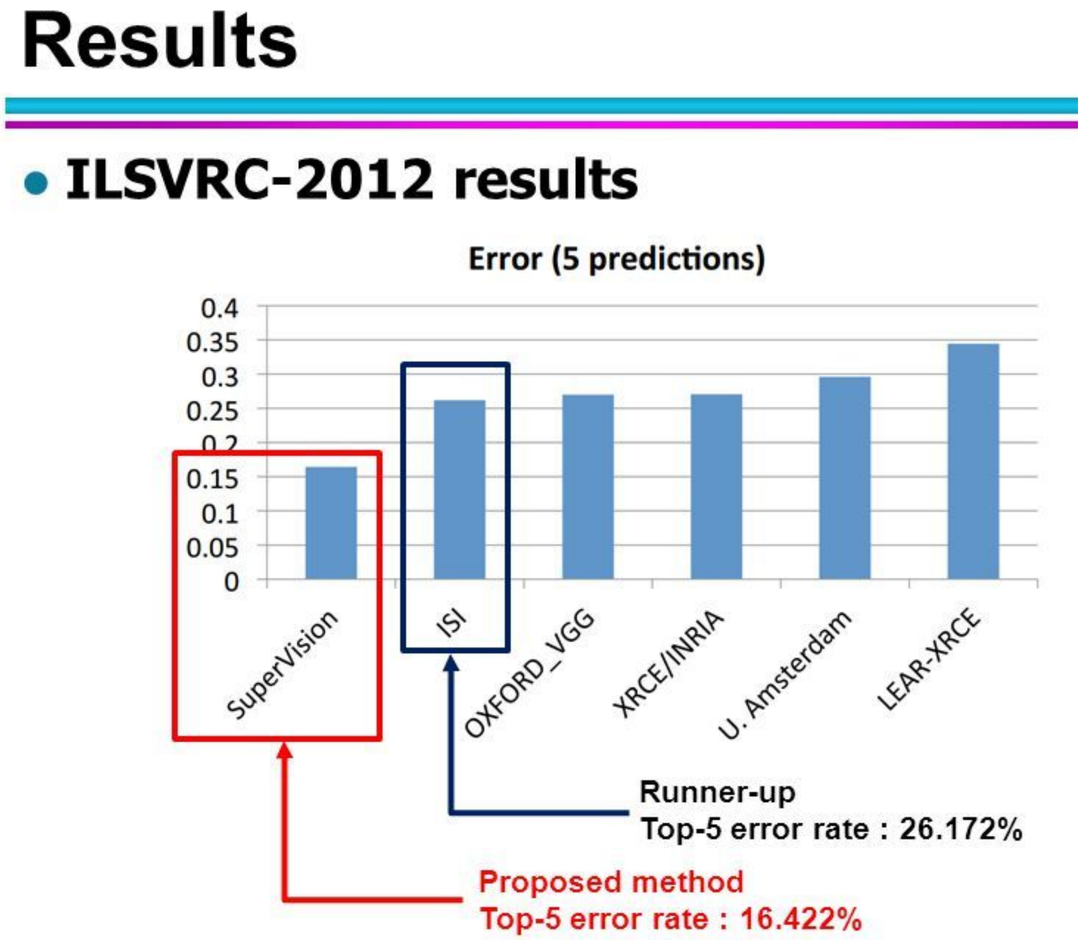
f

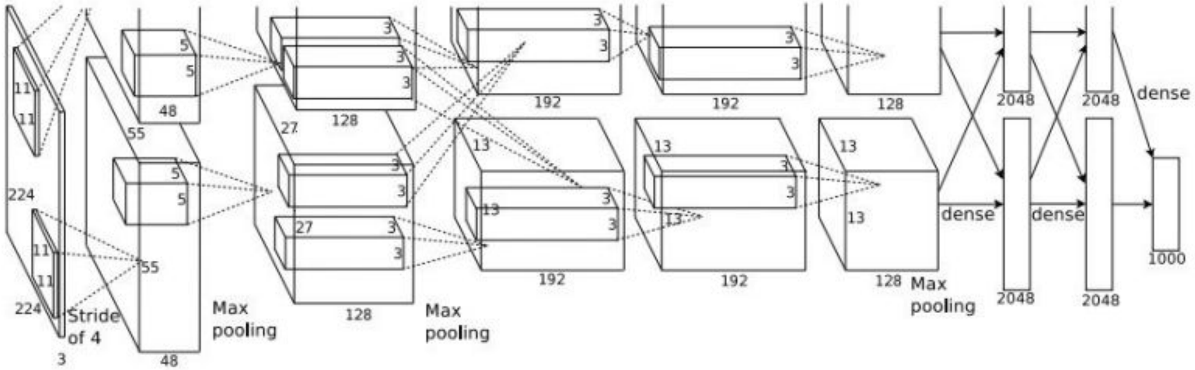
training

10 numbers, indicating class scores

Who wins?

End-to-end learning wins!

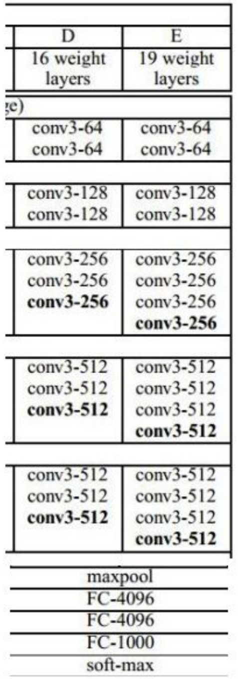




AlexNet (2012)

Disclaimer: hand-crafted features may still be the right choice for your niche application

Modern architectures (deeper and deeper)

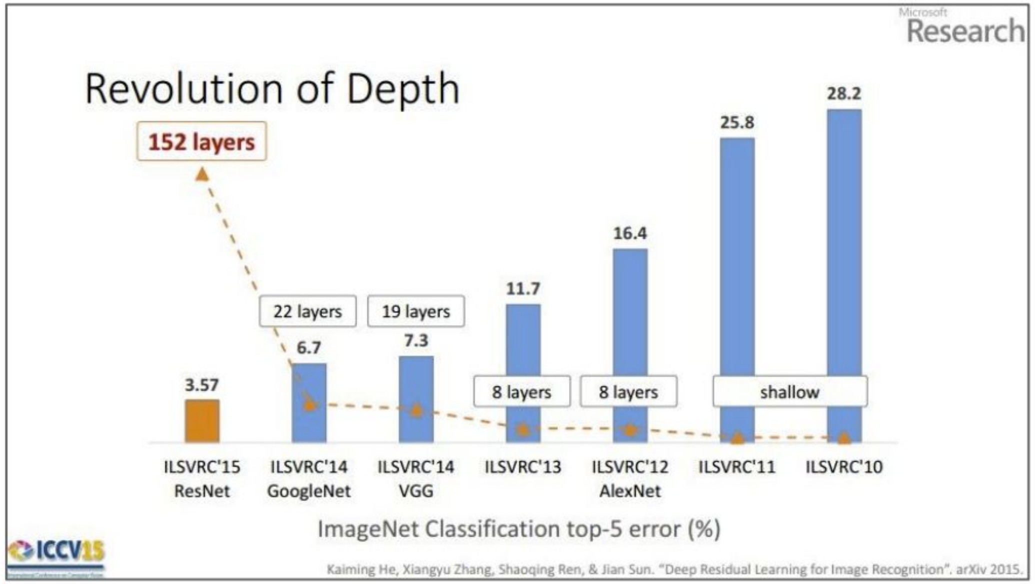


D	E
16 weight layers	19 weight layers
ge)	
conv3-64 conv3-64	conv3-64 conv3-64
conv3-128 conv3-128	conv3-128 conv3-128
conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

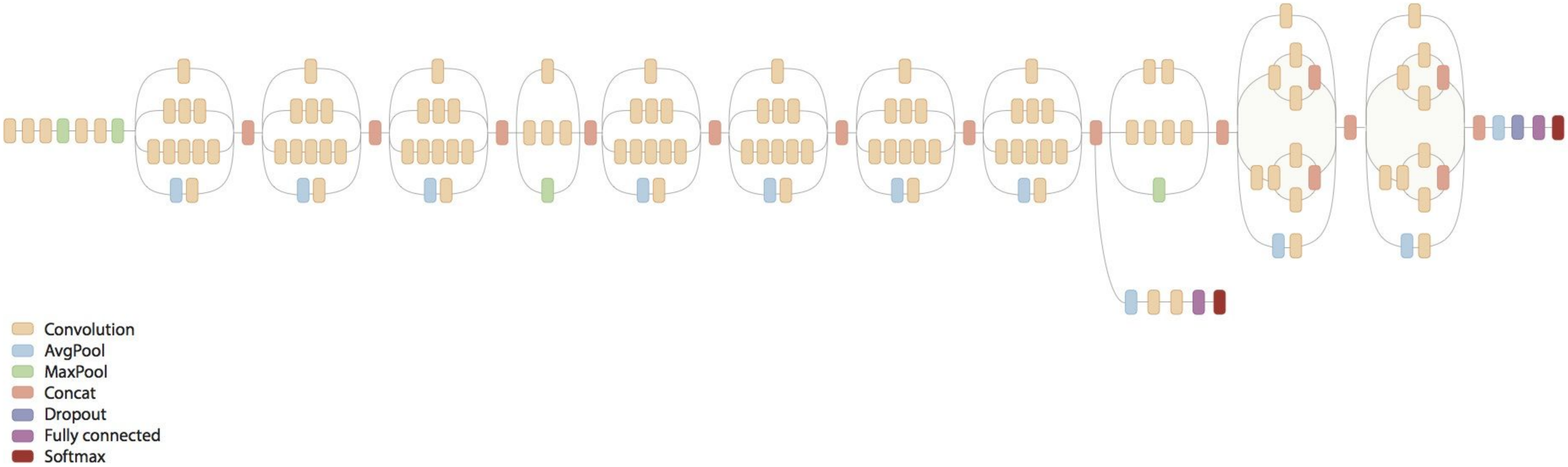
VGG
(2014)

GoogLeNet
(2014)

ResNet
(2015)



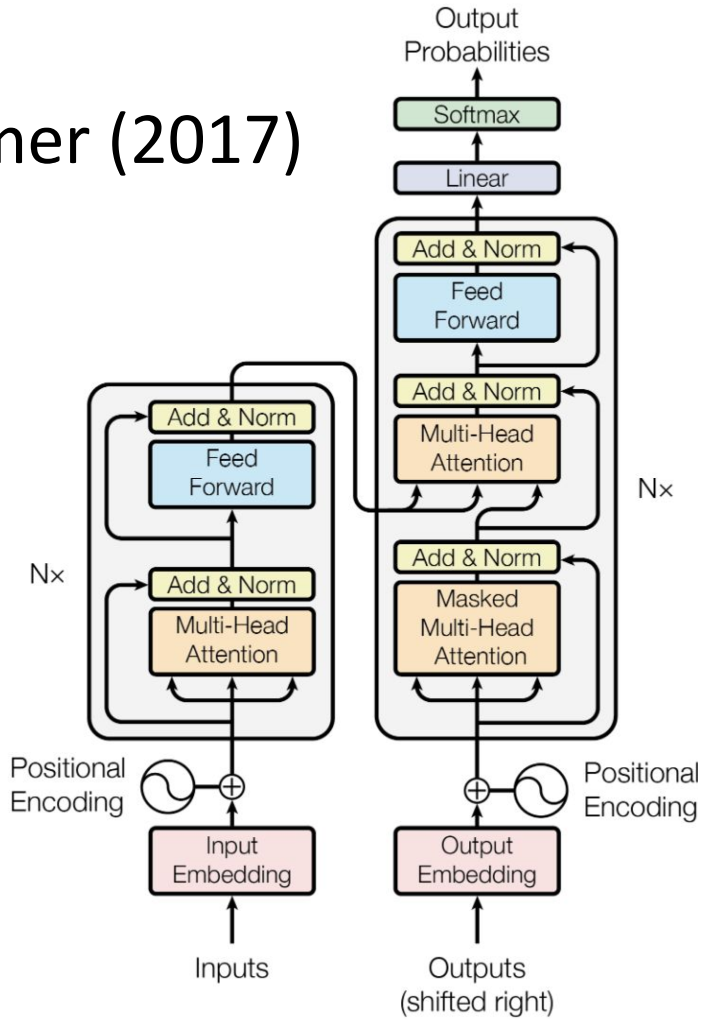
Modern architectures (deeper and deeper)



Inception-v3 (2016)

Even more modern architectures

Transformer (2017)



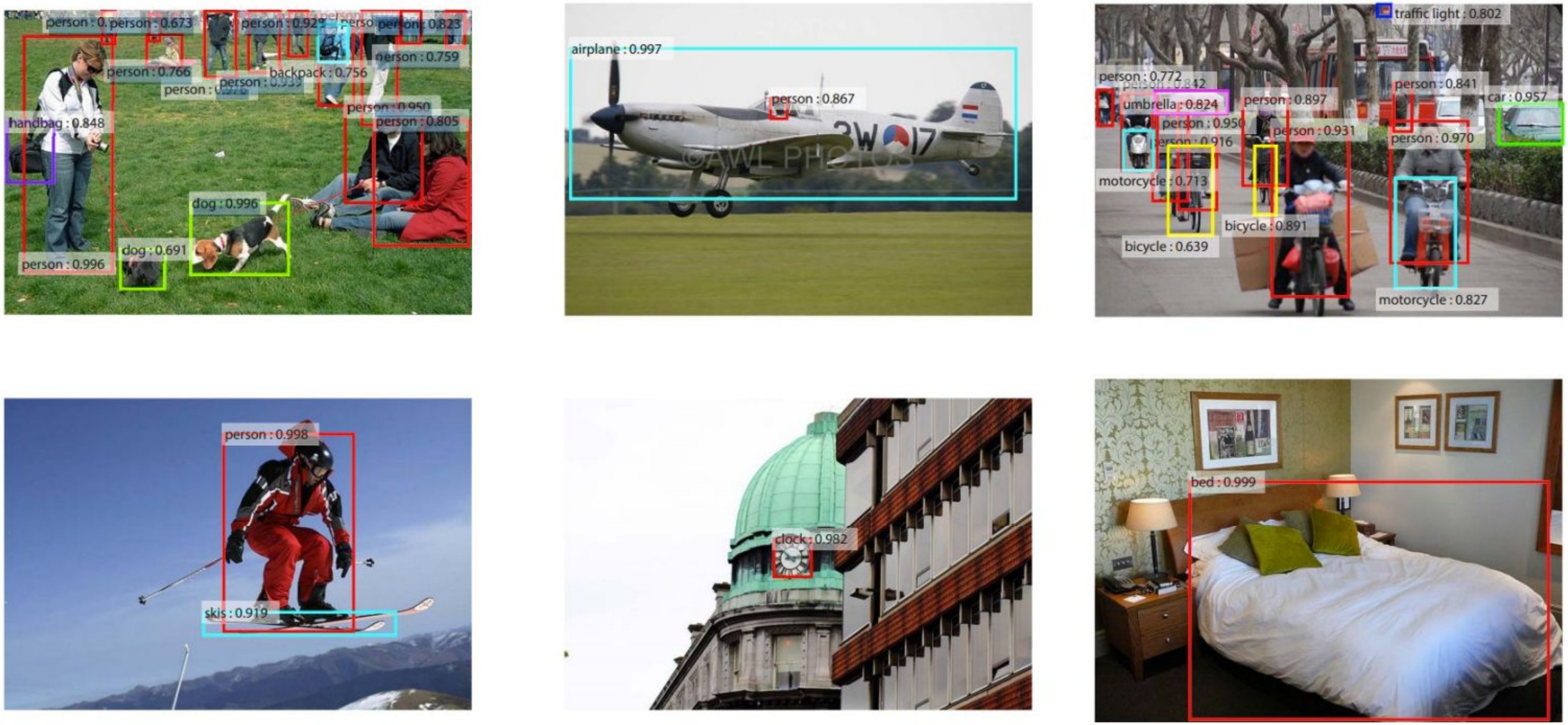
Vision Transformer (2020)

Transformers | Davide Coccomini | 2021

Today's itinerary

- Stats/ML review
- Neural network basics
- Convolutional neural networks
- Robotic applications

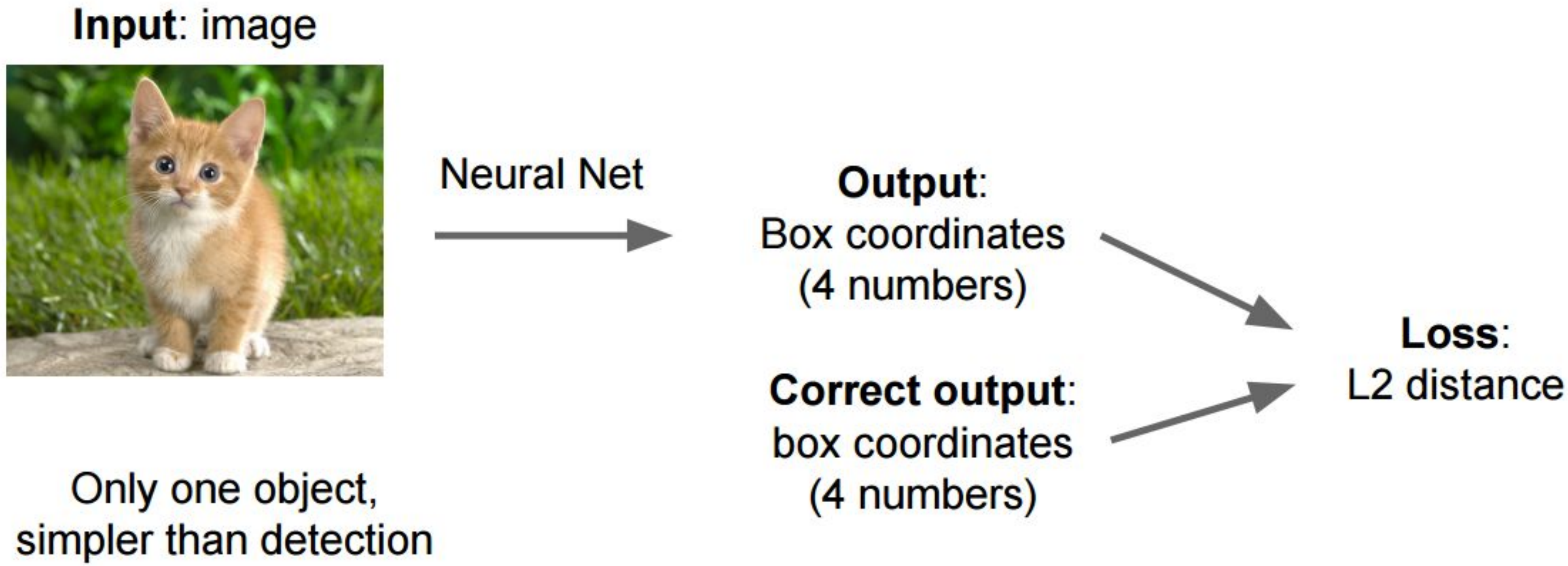
Object localization and detection



Results from Faster R-CNN, Ren et al 2015

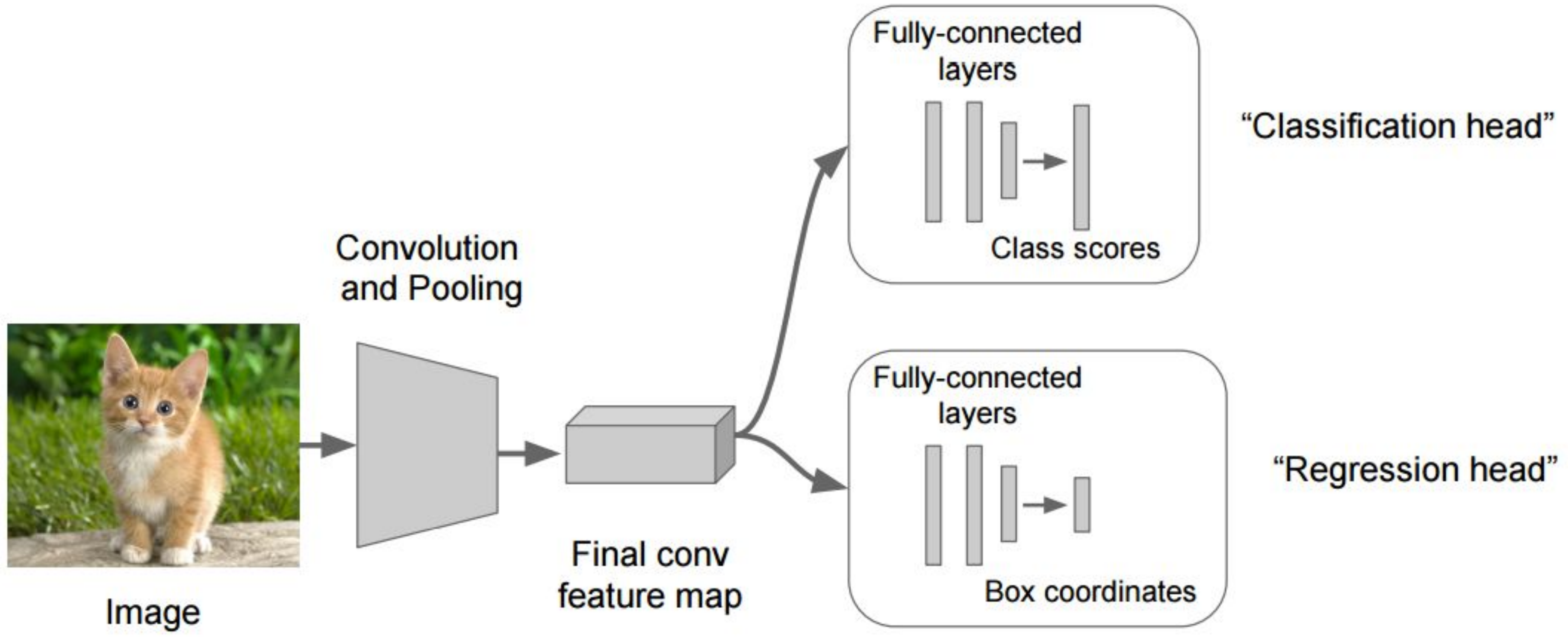
Object localization

Instead of outputting only a class (with associated loss function), also regress on 4 numbers defining the edges of a bounding box



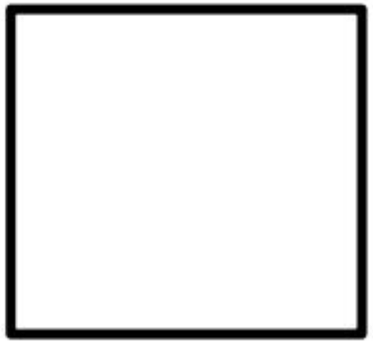
Localization and detection

Instead of outputting only a class (with associated loss function), also regress on 4 numbers defining the edges of a bounding box

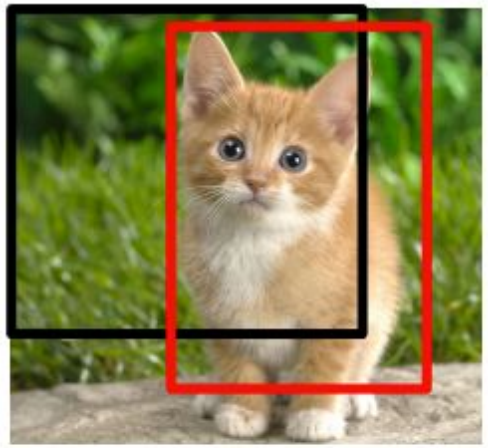


Object detection

Sliding window: using a classifier as the basis for a detector



Network input:
3 x 221 x 221

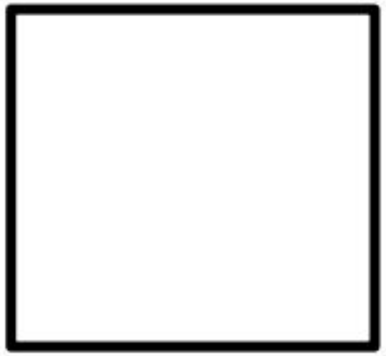


Larger image:
3 x 257 x 257

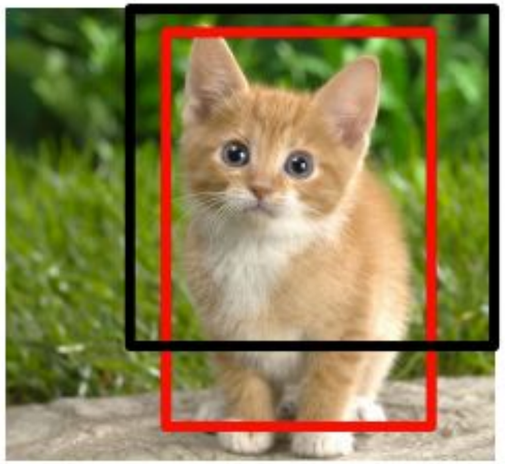
Classification scores:
P(cat)

Object detection

Sliding window: using a classifier as the basis for a detector



Network input:
3 x 221 x 221

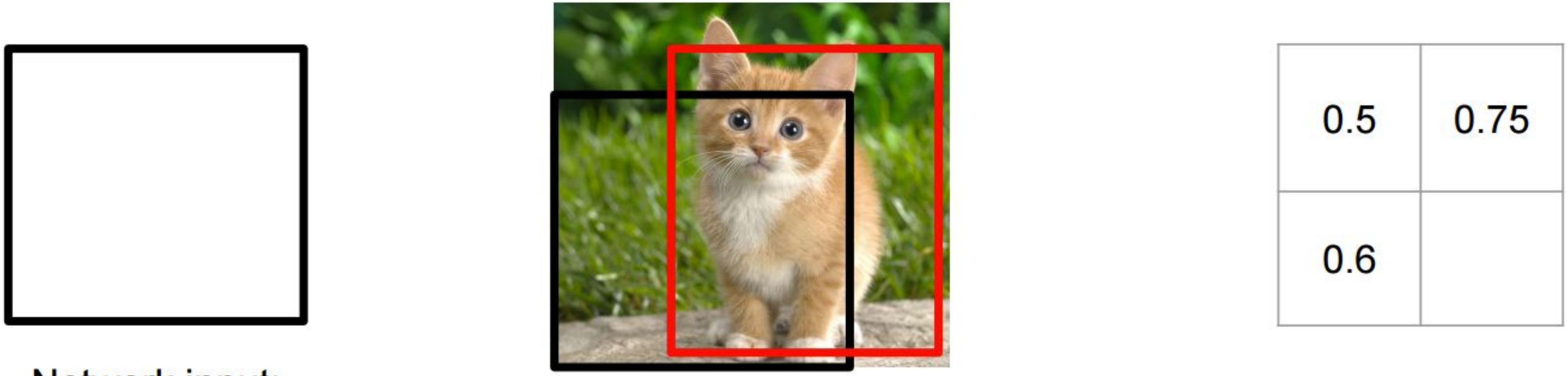


Larger image:
3 x 257 x 257

Classification scores:
P(cat)

Object detection

Sliding window: using a classifier as the basis for a detector



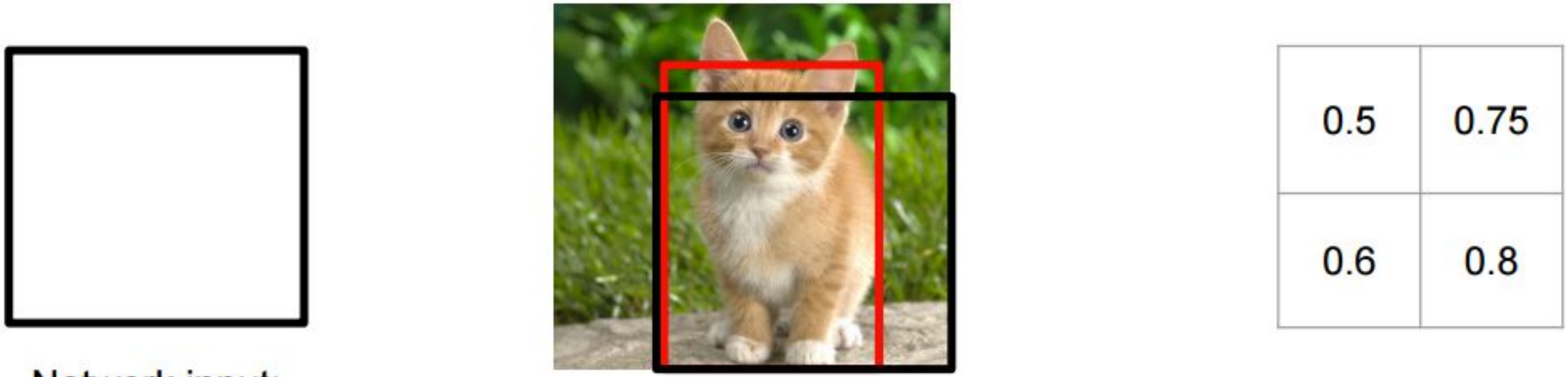
Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

Classification scores:
P(cat)

Object detection

Sliding window: using a classifier as the basis for a detector



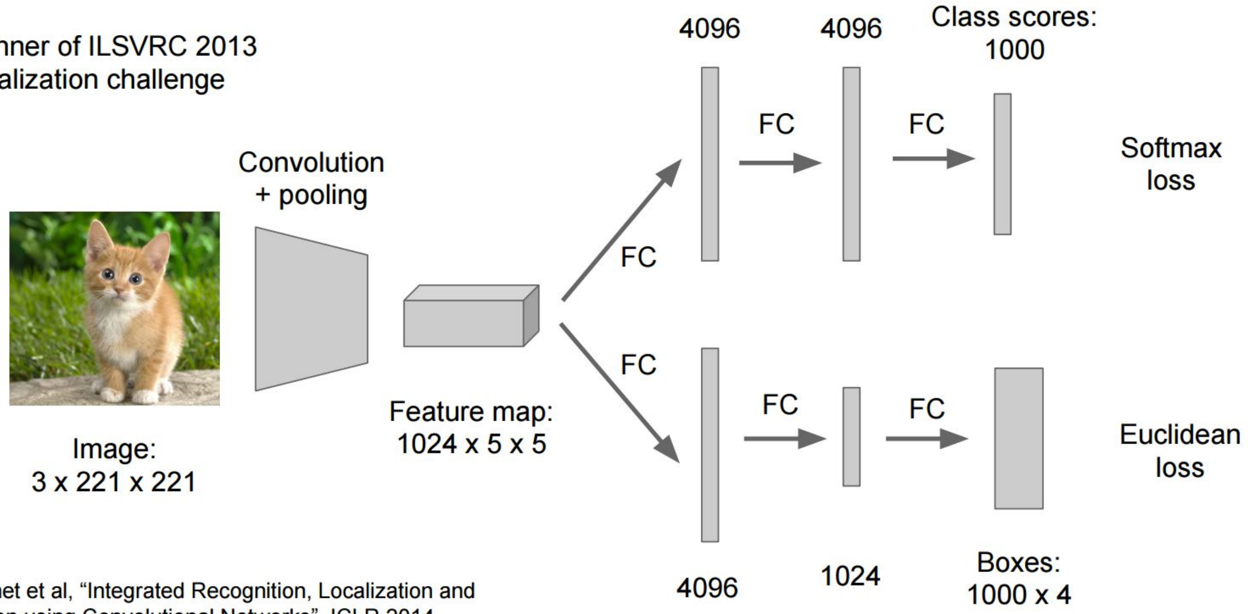
Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

Classification scores:
P(cat)

Object detection – sliding window

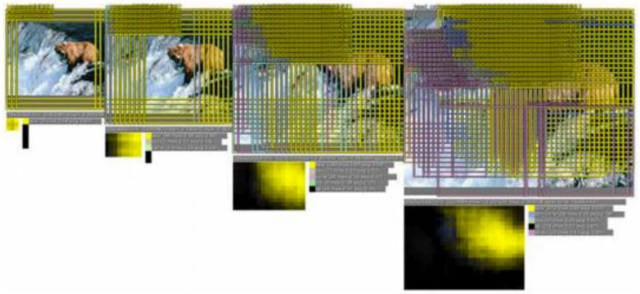
Winner of ILSVRC 2013 localization challenge



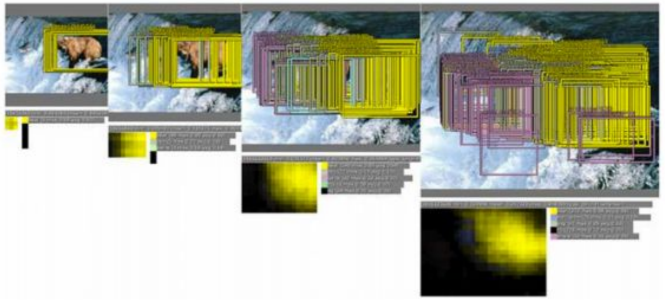
Overfeat (Sermanet et al. 2014)

Sermanet et al, “Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

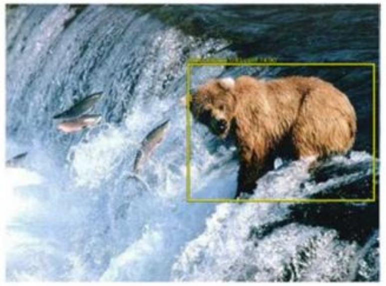
Window positions + score maps



Box regression outputs



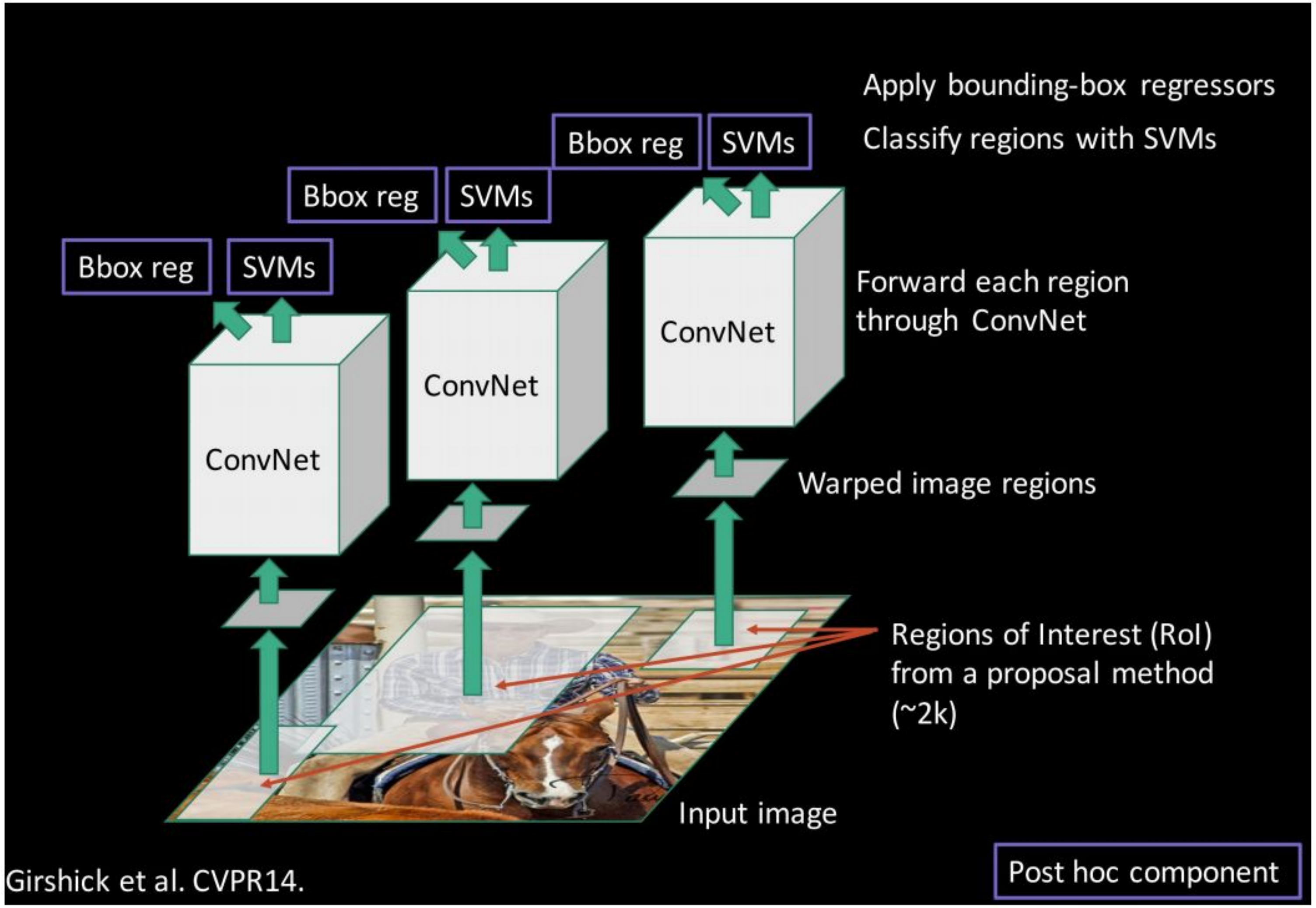
Final Predictions



Object detection – more efficient approaches

"Proposal" method to identify "blobby" regions of interest
(could be another NN)

Two-headed classifer/bounding box regressor



Object detection – more efficient approaches

YOLO: You Only Look Once
Detection as Regression

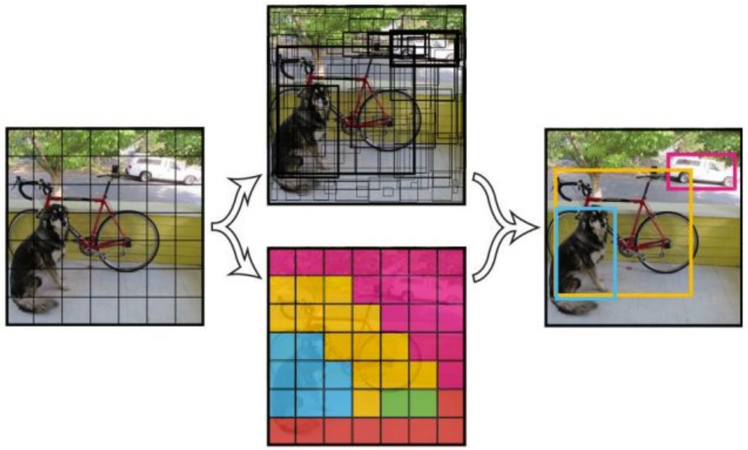
Divide image into S x S grid

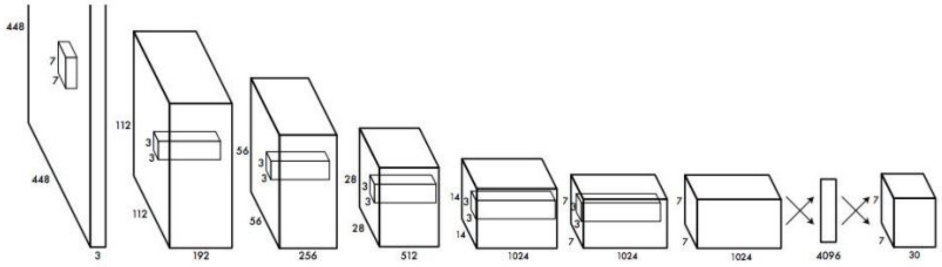
Within each grid cell predict:
- B Boxes: 4 coordinates + confidence
- Class scores: C numbers

Regression from image to
7 x 7 x (5 * B + C) tensor

Direct prediction using a CNN

Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", arXiv 2015





Robotics – need for speed!

Model Checkpoint	Million MACs	Million Parameters	Top-1 Accuracy	Top-5 Accuracy
MobileNet_v1_1.0_224	569	4.24	70.7	89.5
MobileNet_v1_1.0_192	418	4.24	69.3	88.9
MobileNet_v1_1.0_160	291	4.24	67.2	87.5
MobileNet_v1_1.0_128	186	4.24	64.1	85.3
MobileNet_v1_0.75_224	317	2.59	68.4	88.2
MobileNet_v1_0.75_192	233	2.59	67.4	87.3
MobileNet_v1_0.75_160	162	2.59	65.2	86.1
MobileNet_v1_0.75_128	104	2.59	61.8	83.6
MobileNet_v1_0.50_224	150	1.34	64.0	85.4
MobileNet_v1_0.50_192	110	1.34	62.1	84.0
MobileNet_v1_0.50_160	77	1.34	59.9	82.5
MobileNet_v1_0.50_128	49	1.34	56.2	79.6
MobileNet_v1_0.25_224	41	0.47	50.6	75.0
MobileNet_v1_0.25_192	34	0.47	49.0	73.6
MobileNet_v1_0.25_160	21	0.47	46.0	70.7
MobileNet_v1_0.25_128	14	0.47	41.3	66.2

MobileNets (2017)

An Alaskan Malamute (left) and a Siberian Husky (right). Images from Wikipedia

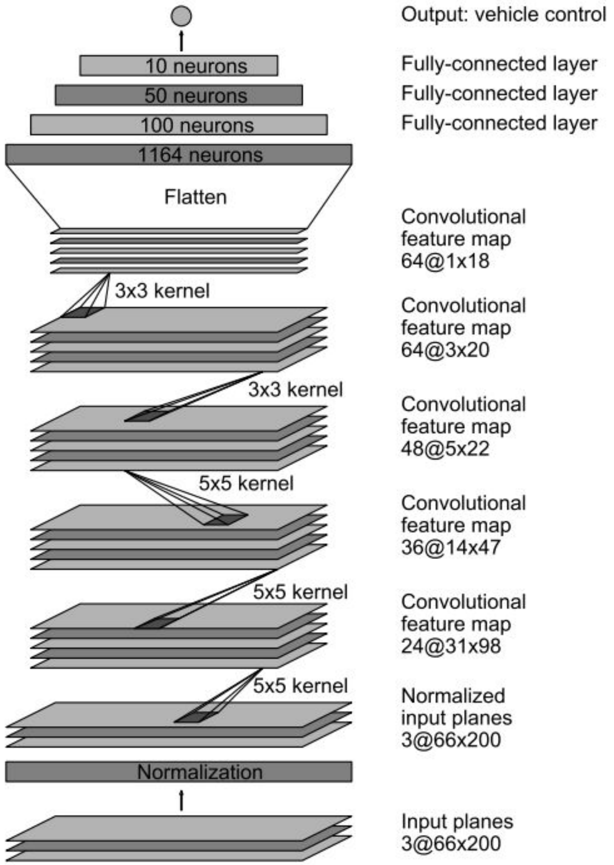
Inception-ResNet-v2

Model	Train	Test	mAP	FLOPS	FPS
Old YOLO	VOC 2007+2012	2007	63.4	40.19 Bn	45
SSD300	VOC 2007+2012	2007	74.3	-	46
SSD500	VOC 2007+2012	2007	76.8	-	19
YOLOv2	VOC 2007+2012	2007	76.8	34.90 Bn	67
YOLOv2 544x544	VOC 2007+2012	2007	78.6	59.68 Bn	40
Tiny YOLO	VOC 2007+2012	2007	57.1	6.97 Bn	207

Tiny YOLO (2017)

End-to-end: from pixels to motor commands

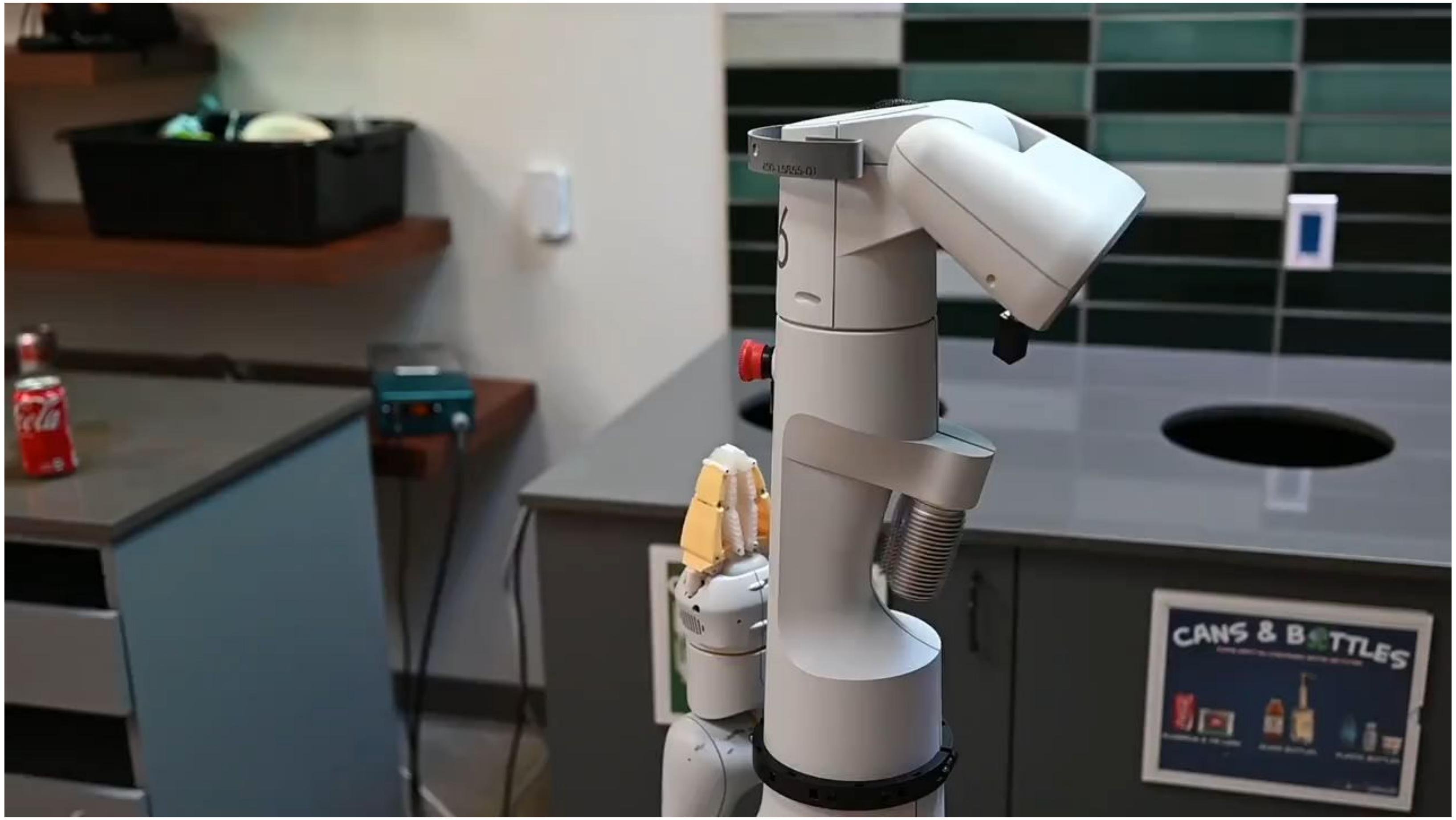
DAVE-2 (NVIDIA 2016)



Somewhat less scary:
https://www.youtube.com/watch?v=HJ58dbd5g8g

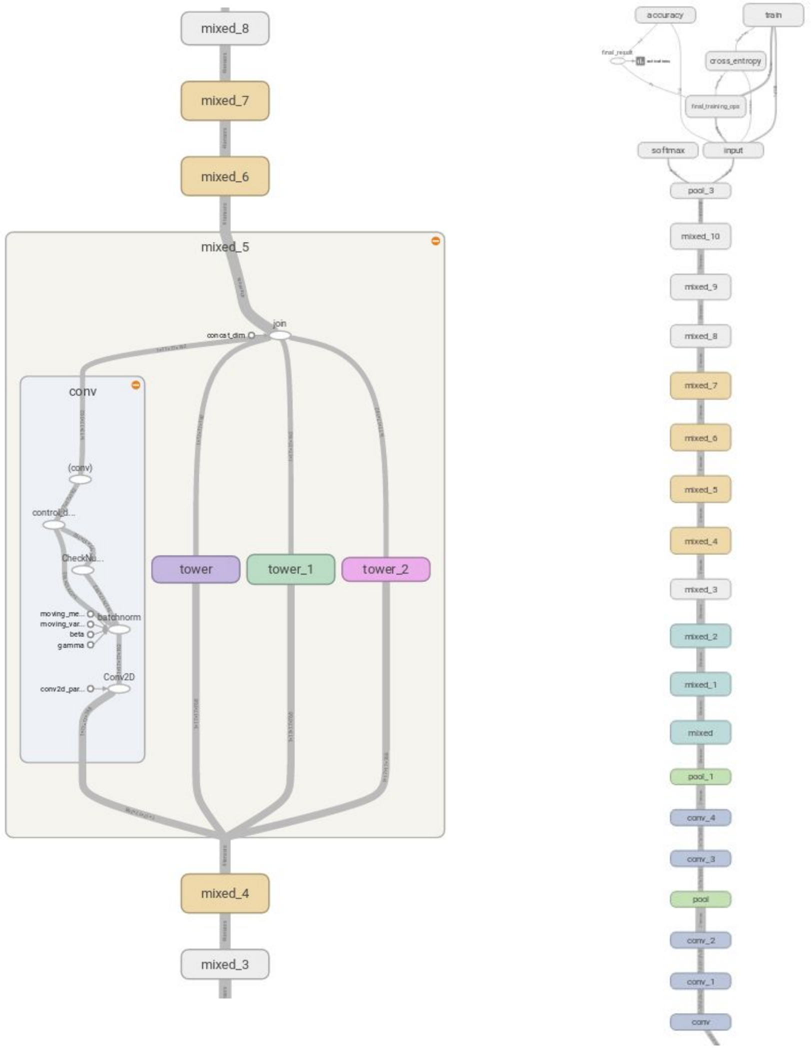
End-to-end: from sensors+language to action

SayCan (Google 2022)



Tools of the trade

- Software packages for automatic differentiation/gradient computation
  - Caffe (old)
  - Torch (old)
  - Theano (old)
  - TensorFlow (Google, Heavyweight #1)
  - PyTorch (Facebook, Heavyweight #2)
  - MXNet/Chainer/… (Others, better at some things for specific applications)
- Specify an abstract computation graph (inputs and outputs of NN equations); software does the rest!



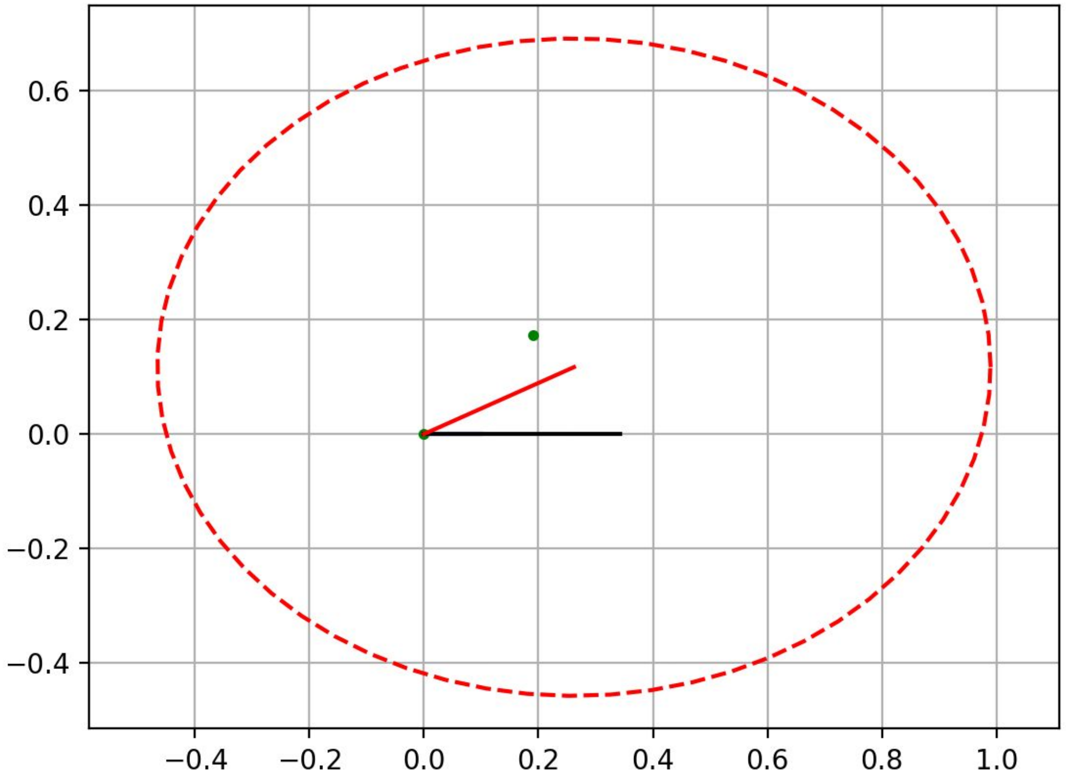
TensorFlow: a *lot* of chain rule in this picture

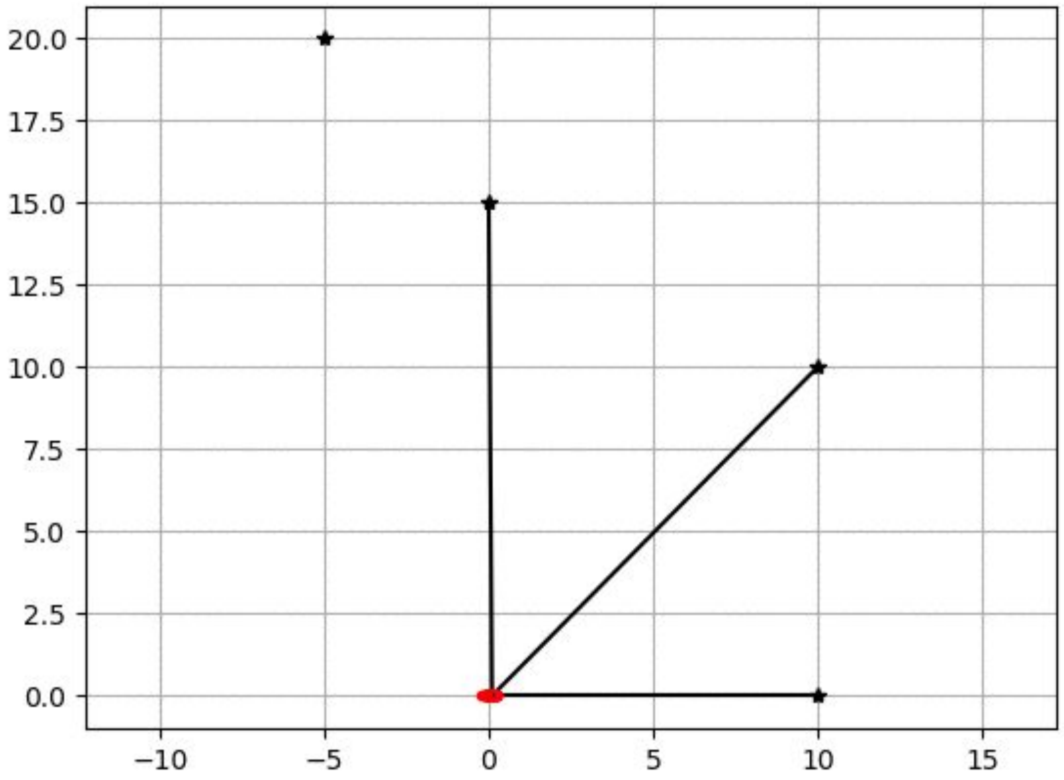
Lots of stuff left out

- Generative vs. discriminative models
- Train/validation/test sets
- Learning rate and other hyperparameter tuning
- Recurrent neural networks for sequential data (e.g., videos)
- Reinforcement learning and ML outside of purely visual recognition-focused tasks

Consider STATS216, CS229, CS231n, CS224n, CS331b to learn more!

Next time


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−10
−5
0
5
10
15