Convolutional Neural Networks (CNN)

EE 599 Deep Learning

Spring 2019

Brandon Franzke

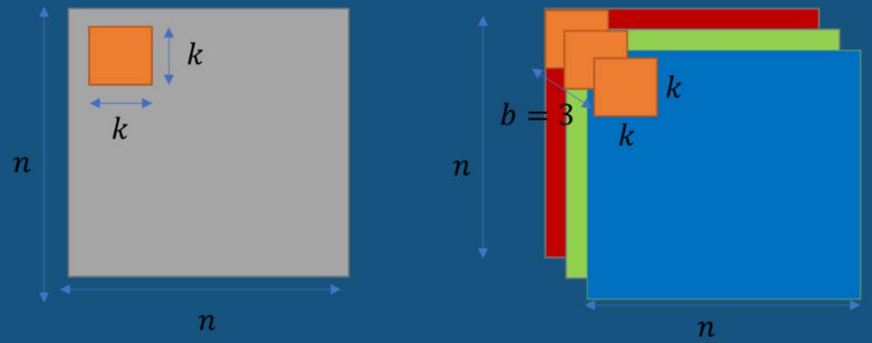
Convolutional are so **DEEP.**

How deep are they?

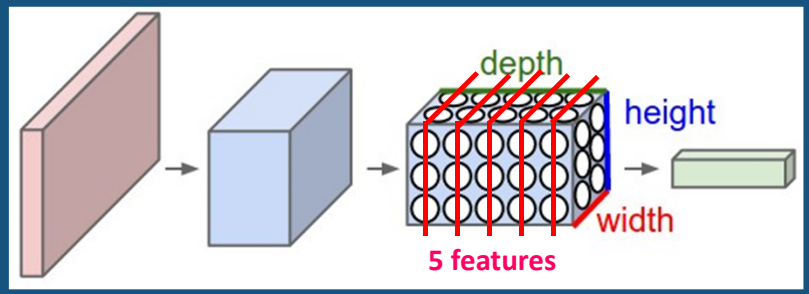
- Convolutional neural networks use “localized” matrix multiplication
- Train to find the *features* (convolution kernels)

This week we cover

- What can you do with convnets?
- Standard architecture
- Thirty years of CNN
- Parameter count vs compute time
- Filters, convolutions, and pooling
- Blocks
- A little Keras



LOCAL matrix multiplication



Train FEATURES

Is this a cat?

(estimate $P[cat]$)

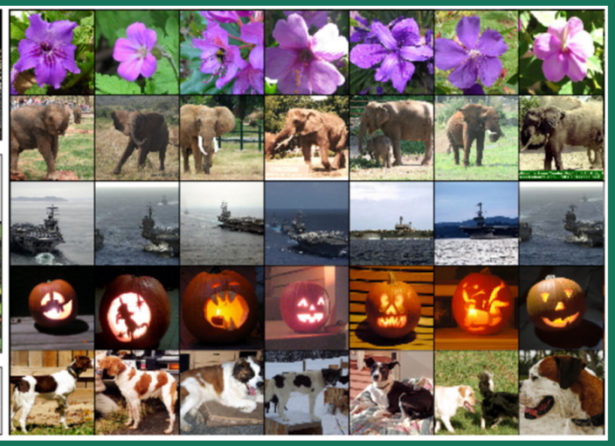
Is this a cat?

(estimate $P[cat]$)

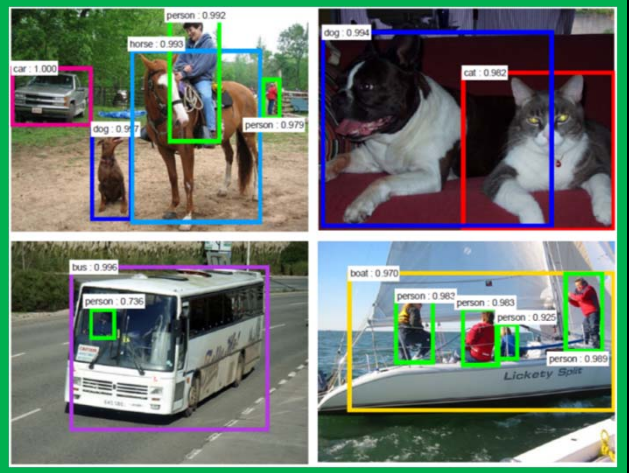
Convnets are Everywhere!

Classification

Recall

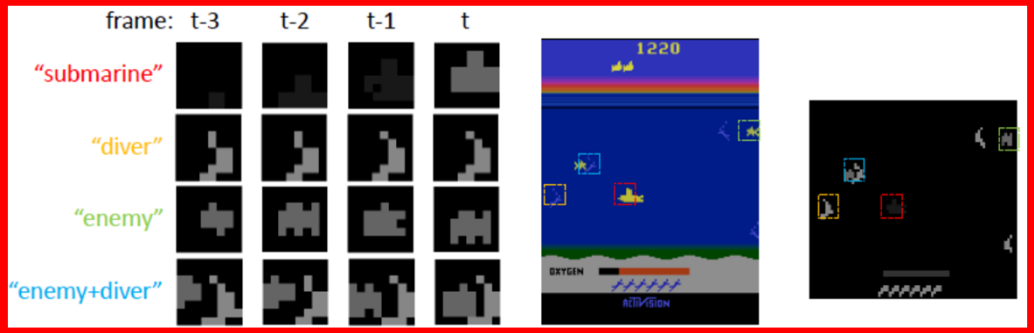


Detection



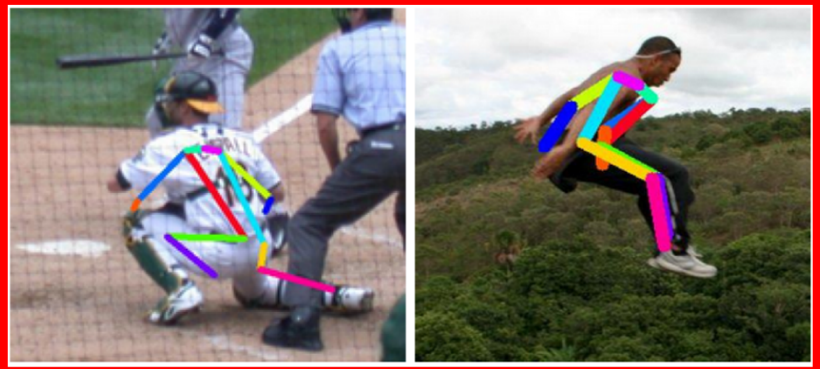
Segmentation

Policy selection



Convnets are one of the most influential technological advancements in the past 7 years.

Pose estimation



Captioning

a train is traveling down the tracks at a train station

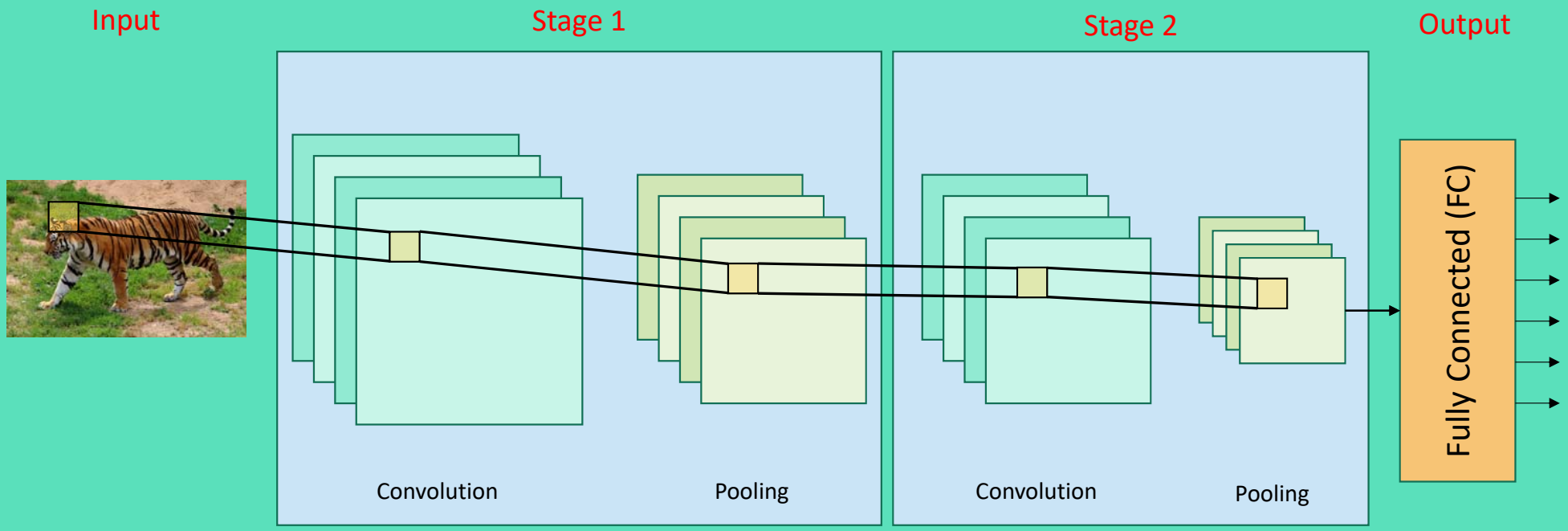
a cake with a slice cut out of it

a bench sitting on a patch of grass next to a sidewalk

Deep Fakes

Style transfer

Convnets



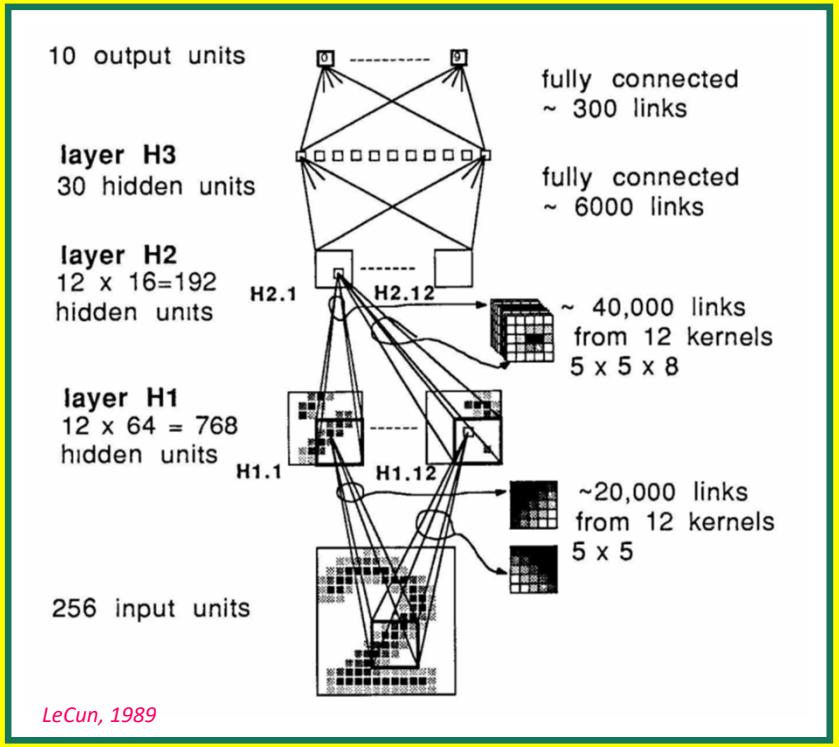
Convolutional units = feature detectors

Pooling units = down sample and aggregate

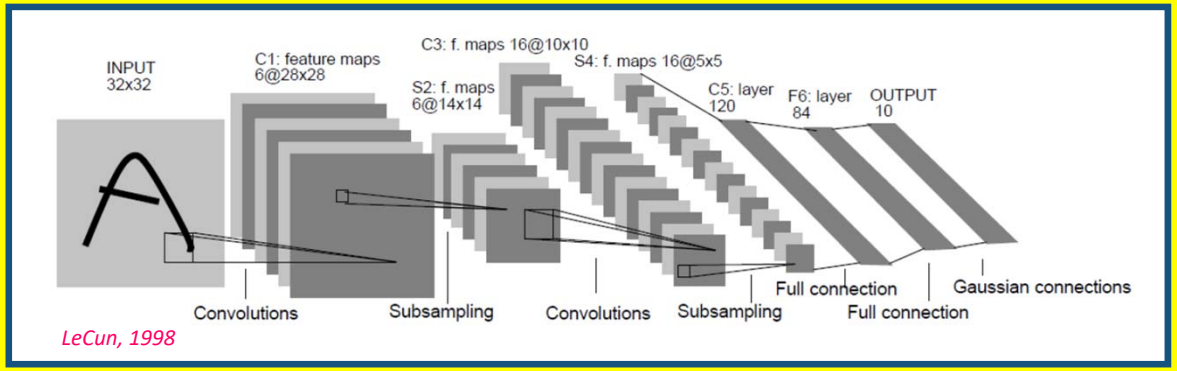
More stages gives more *powerful* networks

Just pay the computational cost

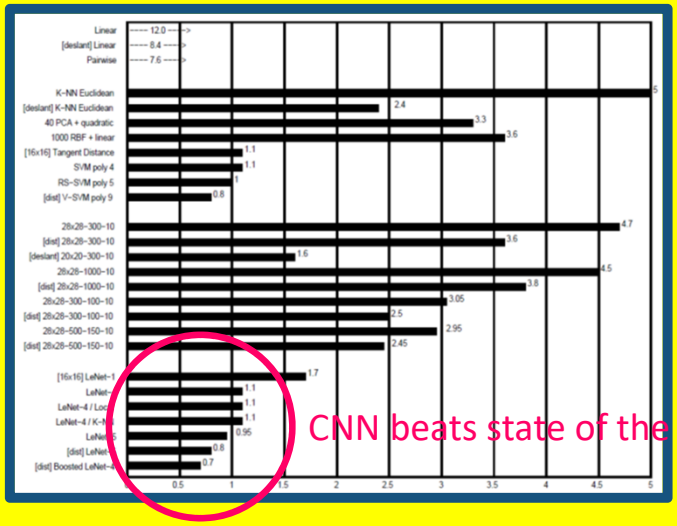
Thirty years of convolutional networks



LeCun, 1989



LeCun, 1998

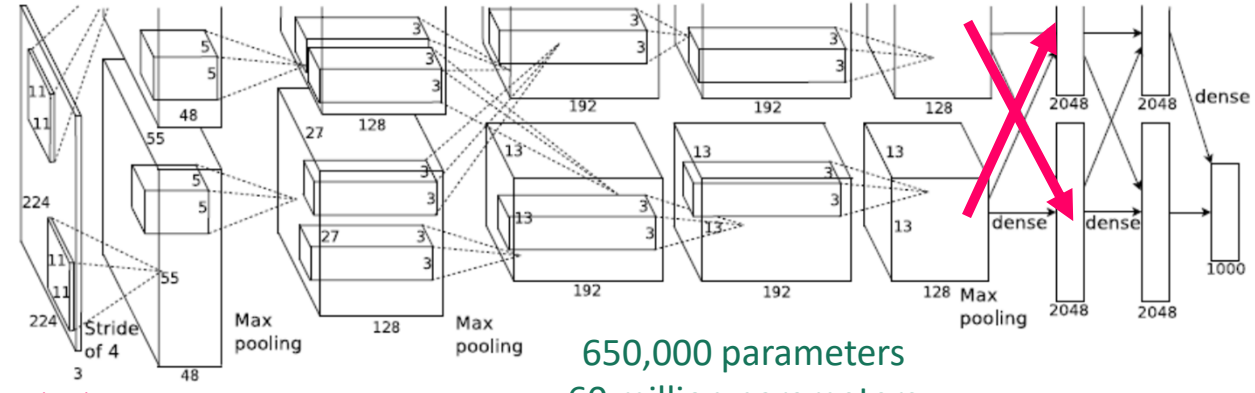


CNN beats state of the art

LeNet-5

And then a 15 year break. In 2012 AlexNet

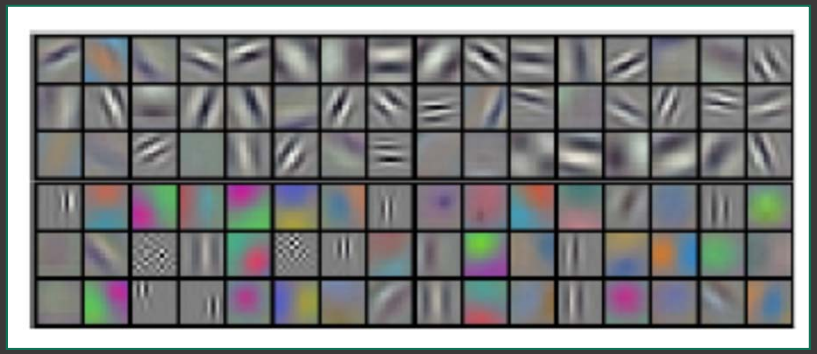
Imagenet has 1000 classes



Krizhevsky, 2012

Rank	Name	Accuracy
1.	AlexNet	16.4%
2.	SIFT + FVs	26.2%

CNN 10% better than state of the art!



96 first layer convolutional kernels (11x11x3)

The idea: compromise to reduce trainable parameters

input size: ≈ 200,000 (256x256x3)

Fully connected network

Say, 1 hidden layer with 1000 neurons

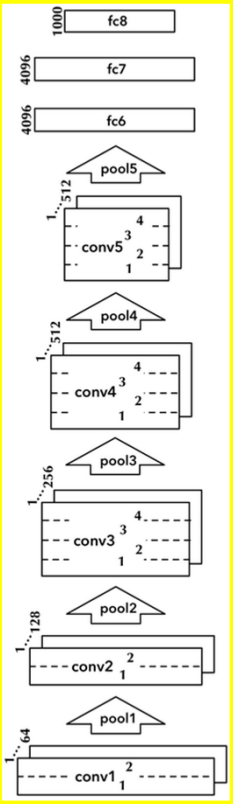
> 200 million parameters!!

First generation CNN (VGG16)

```
>>> from keras.applications import VGG16
>>> model = VGG16()
>>> model.summary()
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Trainable params: 14,714,688

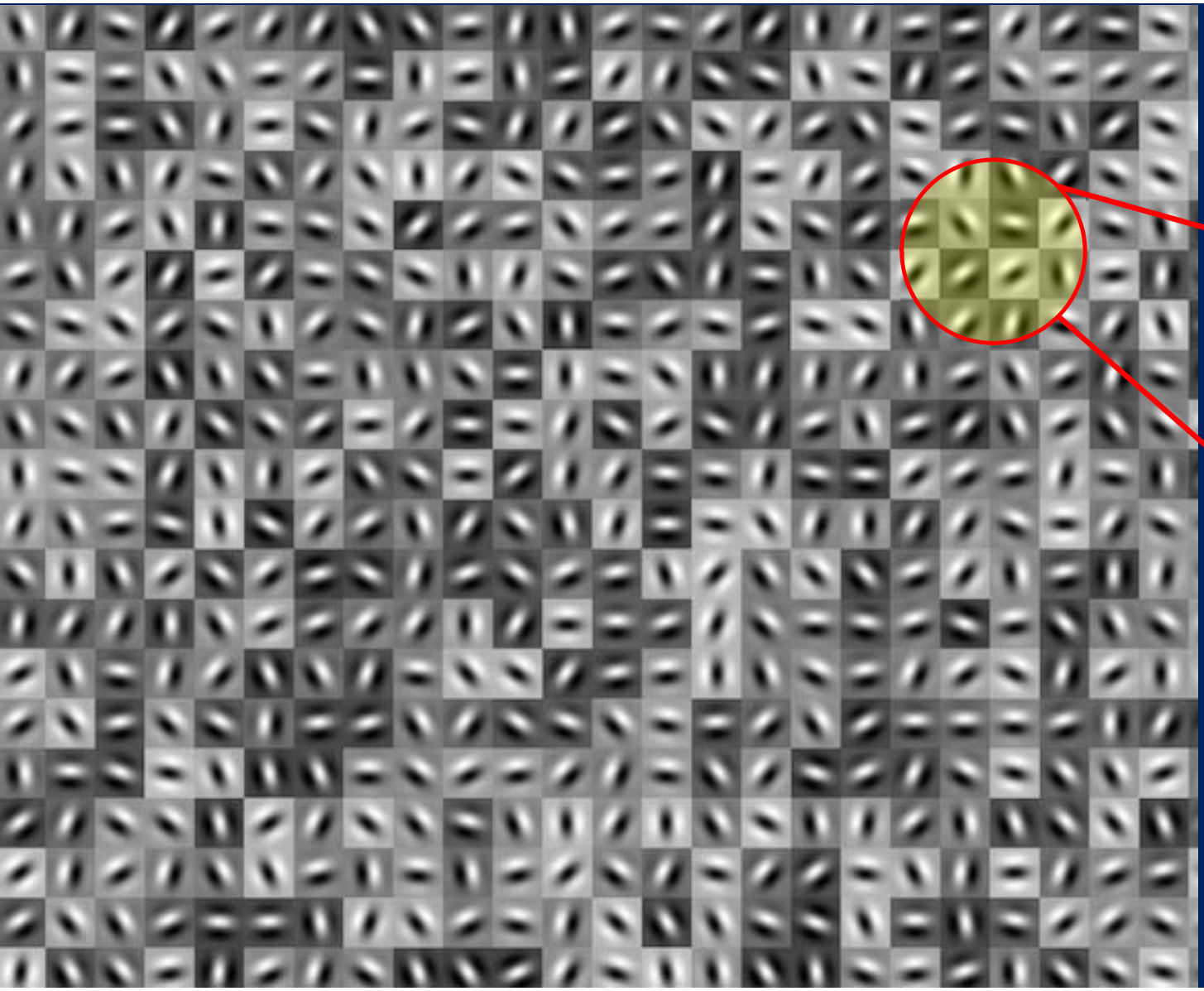


Reduction from:

1. Sparse interactions
2. Parameter sharing
3. Equivariant representations

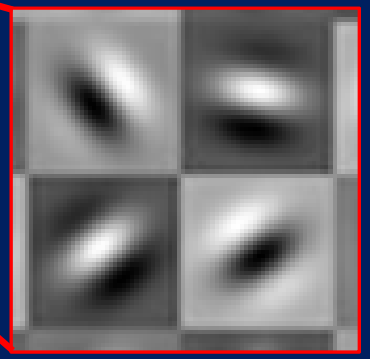
Added benefit:

Variable input size



Lots of filters

(think: *Gabor*)



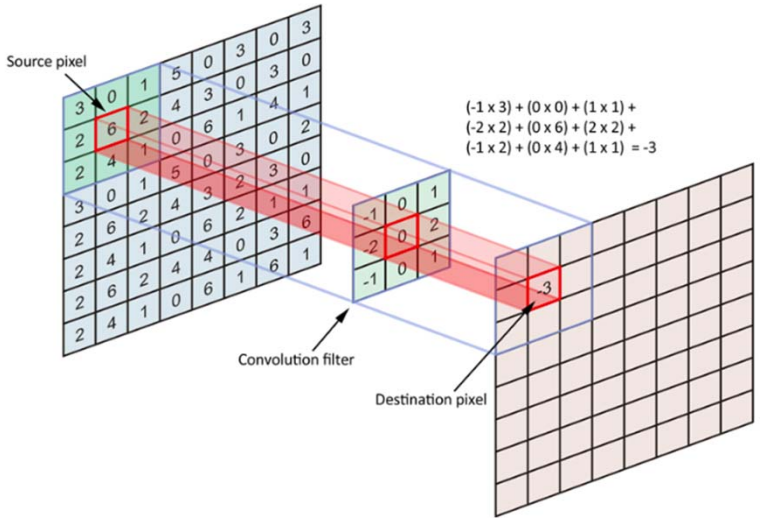
line edge

Multiply filter with input

dark = 0 and light = 1

Convolutional layers capture local structure

Discrete “convolution” is a local dot product.



$$S(i,j) = (I * K)(i,j)$$

$$= \sum_m \sum_n I(i+m, j+n)\, K(m,n)$$

K has small dimension $\Longrightarrow \therefore$ fast compute

This is what you learn!

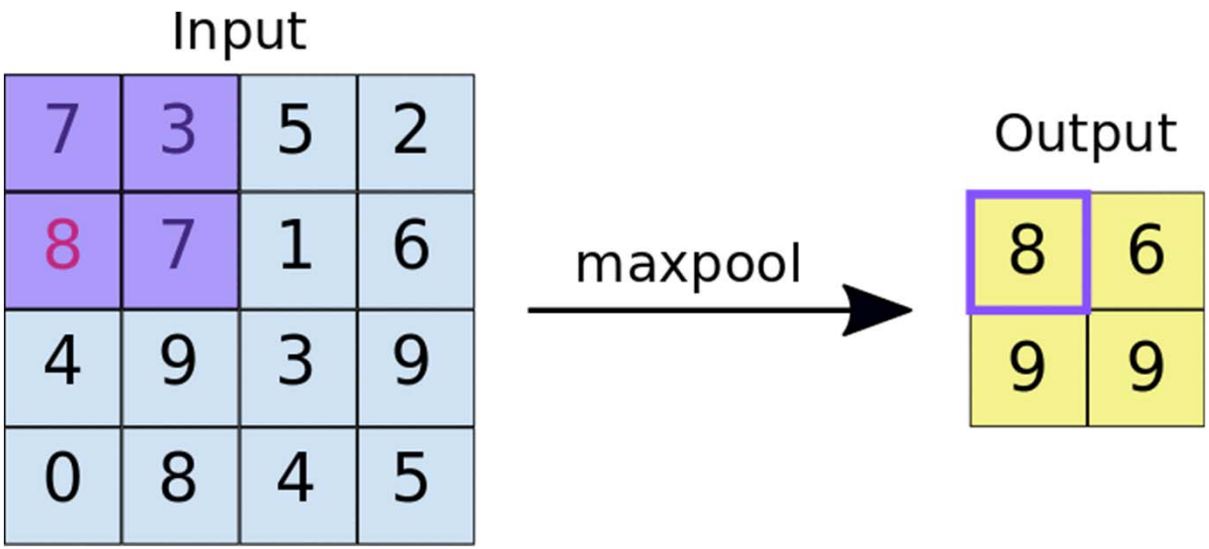
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

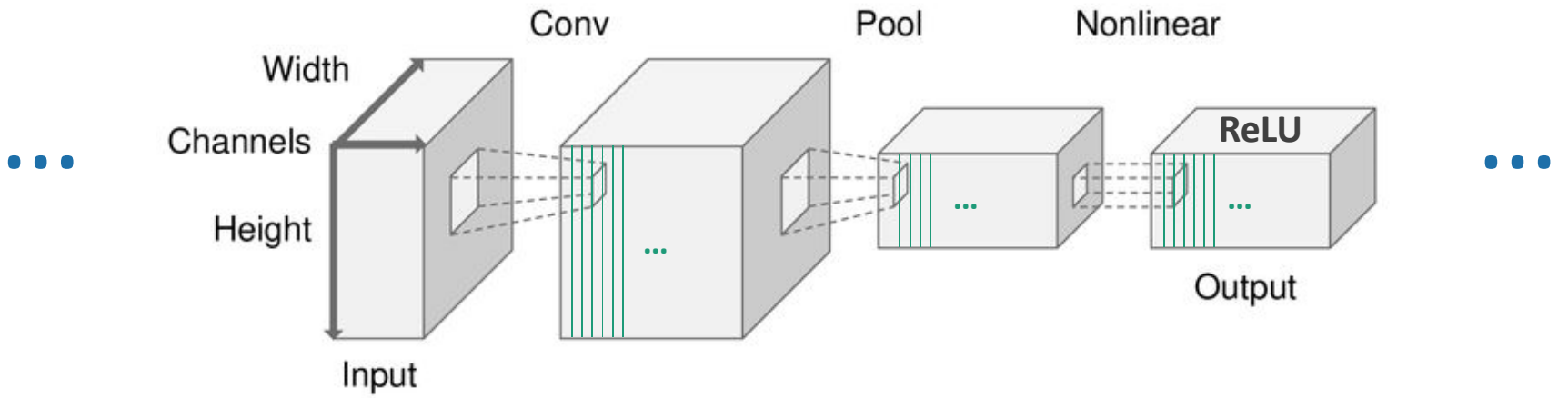
Pooling layers downsample



Aggregate filter matching over local

Downsampling does more than just reduce size
It concentrates the image for deeper feature detection

CNNs use block templates



Block N-1

Block N

Block N+1

New blocks drive state of the art results

Next week we cover

- Convolution layer
  - Dimension, stride, channels and depth, padding
- Pooling layers
- Backpropagation
- Architectures
  - VGGNet, AlexNet, GoogLeNet (inception), ResNet (residual)
  - Single Shot and YOLO (R-CNN)
  - MobileNet (“depthwise” convolution)
- Datasets
  - ImageNet, CIFAR, MS-COCO, Pose, StreetView
- CNN in other fields (non-CV)