

Dive into Deep Learning

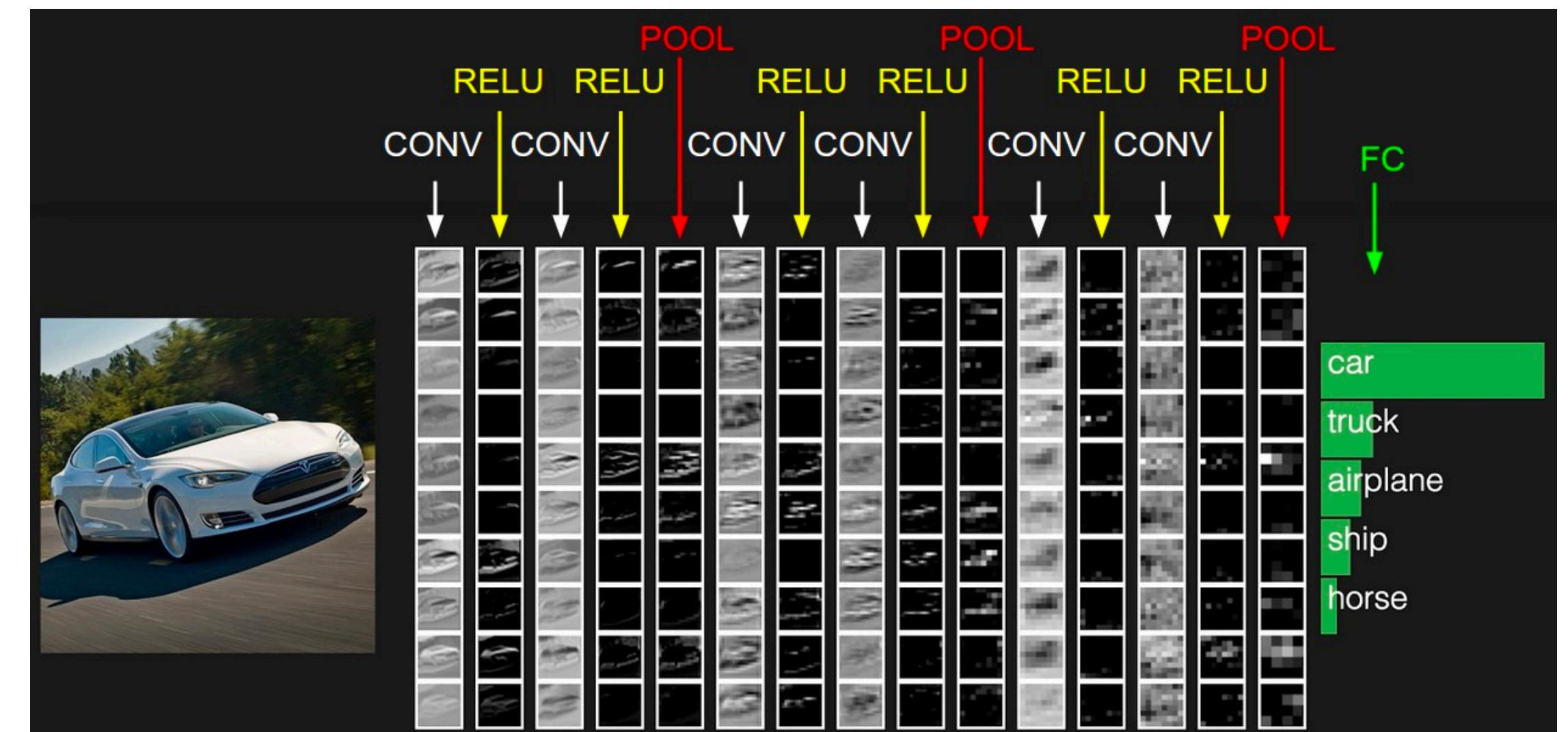
Chapter 6. Convolutional Neural Networks

Wayne L, Nov 1, 2020

CNN

CNN in a nutshell

- A typical CNN takes a raw RGB image as an input.
- It then applies a series of non-linear operations on top of each other.
- These include convolution, activation, matrix multiplication, and pooling (sub-sampling) operations.
- The output of a CNN is a highly non-linear function of the raw RGB image pixels
- Key operations
 - Convolution layer: N-d convolution
 - Fully Connected layer: matrix multiplication
 - Activation layer: sigmoid, tanh, relu
 - Pooling layer: max pool, avg. pool



Before CNN

Traditional method (hand-crafted feature)

- lots options
- domain specific
- more complex
- poor performance

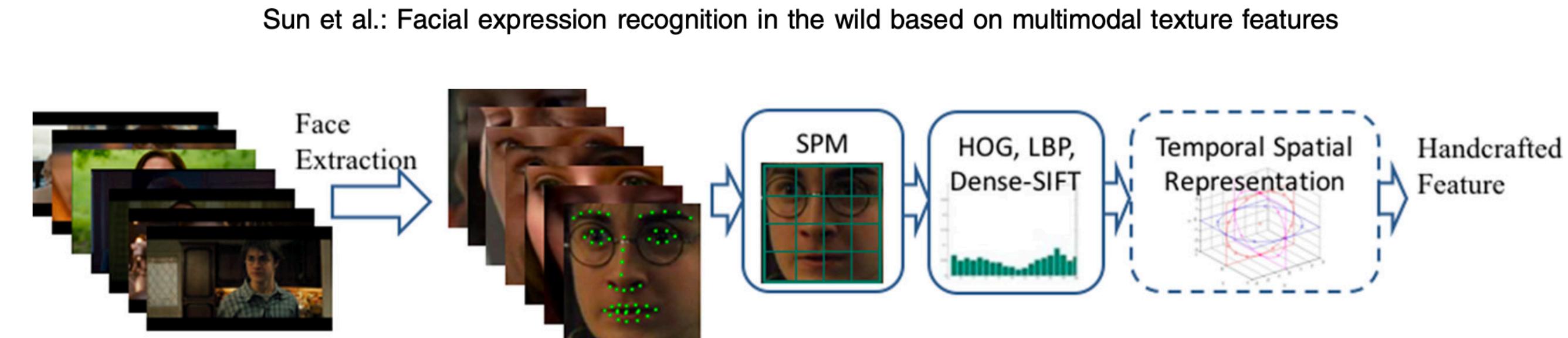
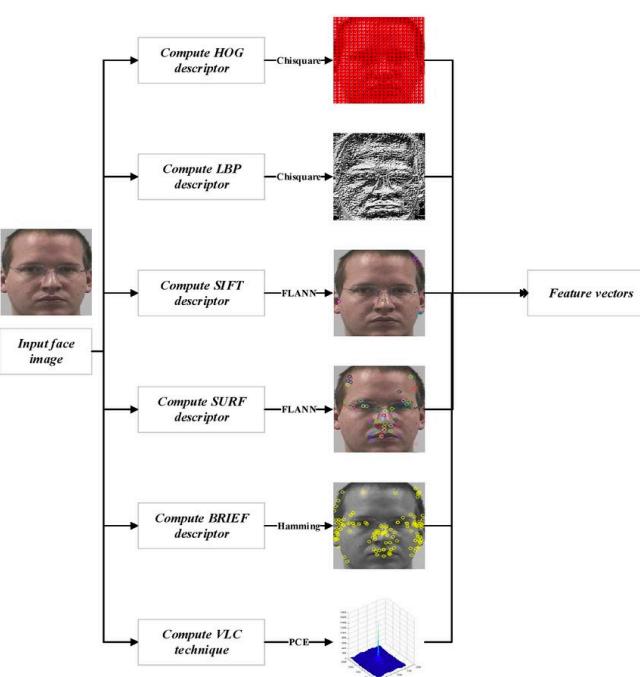
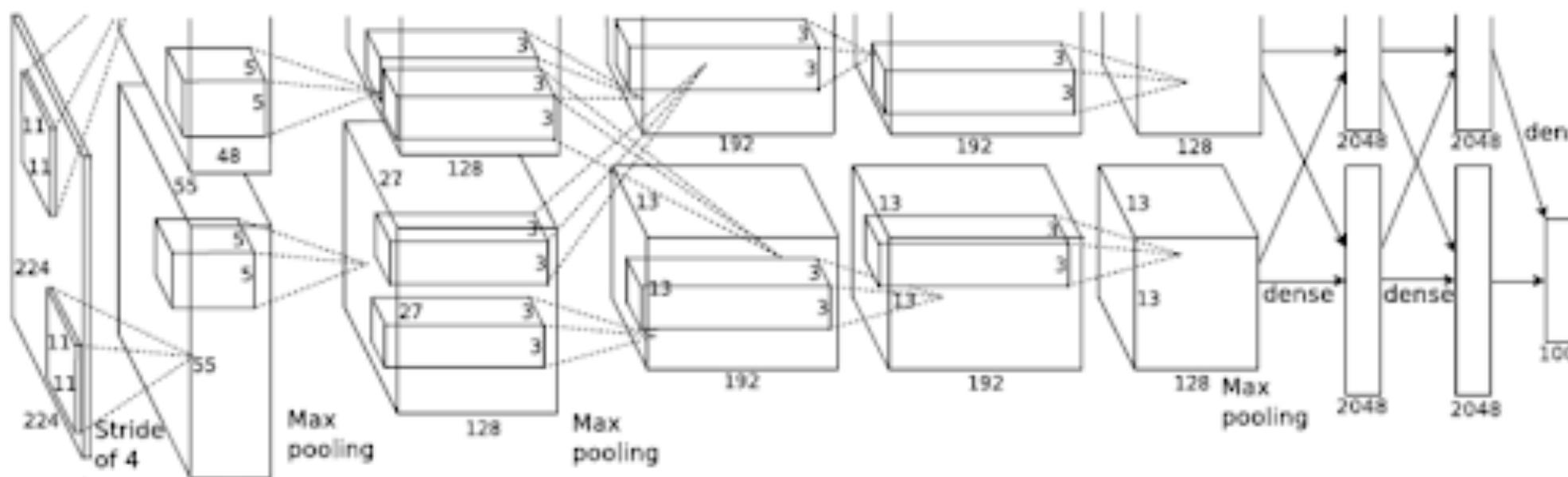
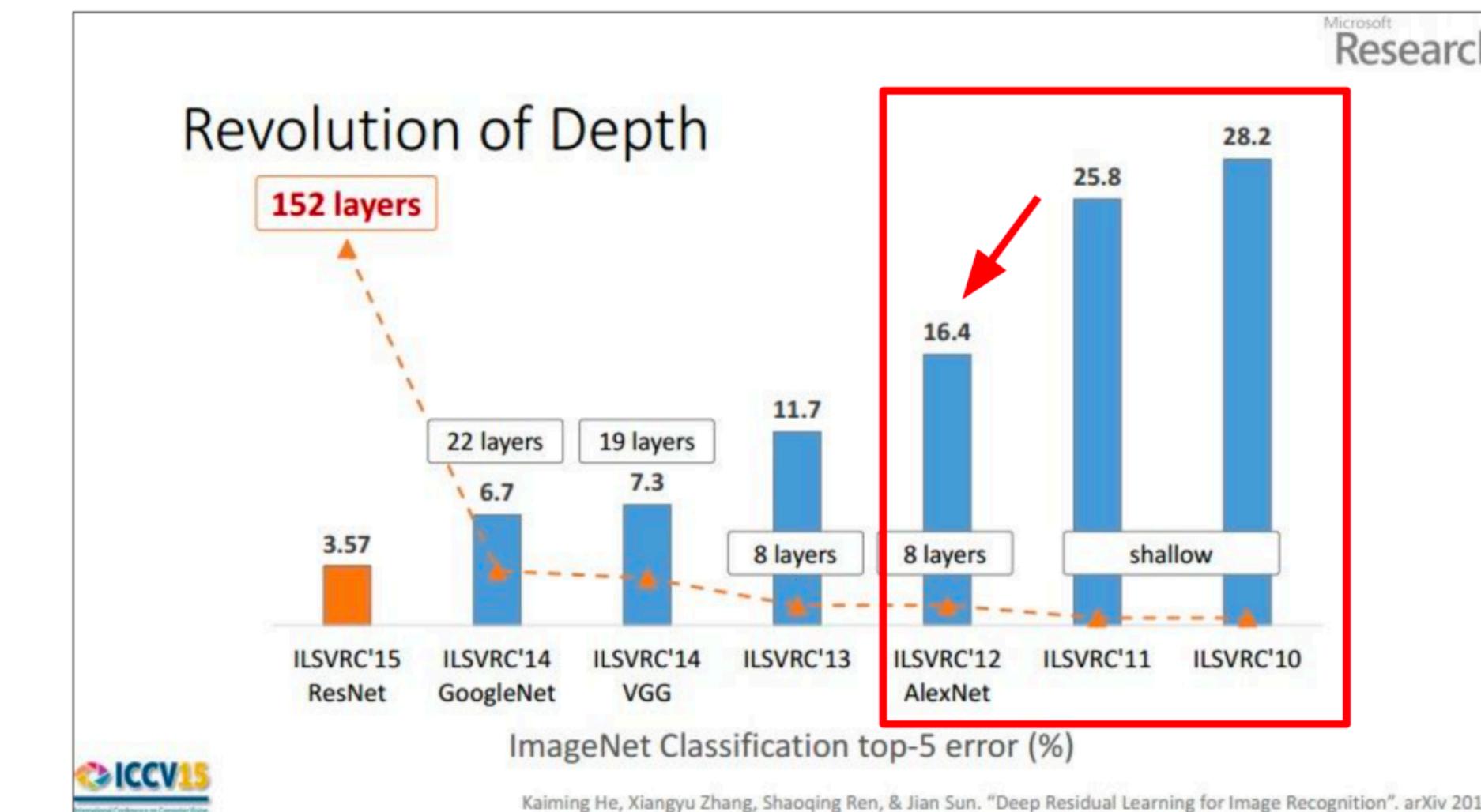


Fig. 4 Pipeline of handcrafted features extraction. The dashed box means that the temporal-spatial representation is only used for AFEW dataset.



Alexnet'12

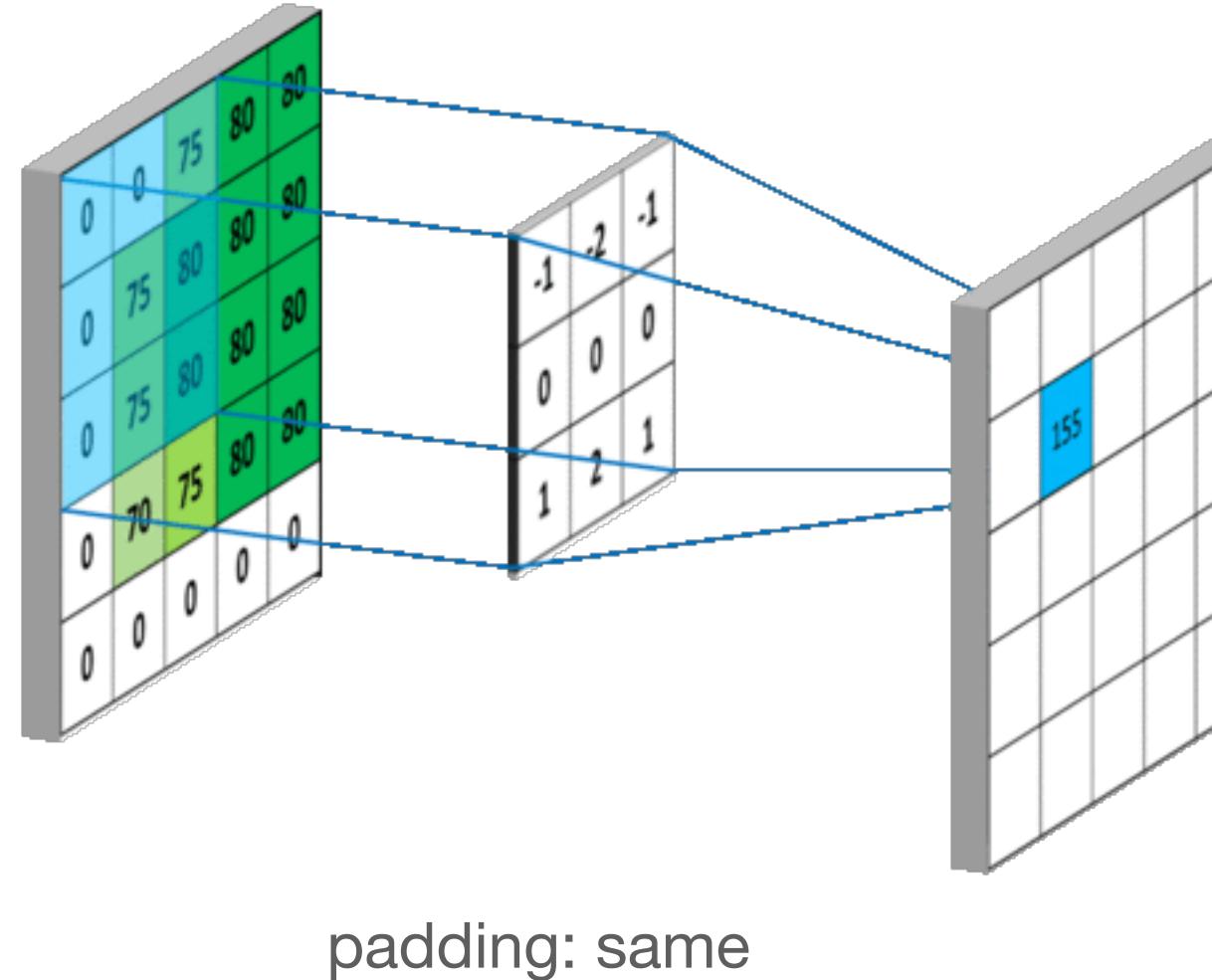


(slide from Kaiming He's recent presentation)

CNN

Multiple {Input & output} Channels

- Multiple channels (e.g., color image: RGB)
 - Shape: (3 x Height x Width)
 - Kernel(Filter, Convolution map)



Input Volume (+pad 1) (7x7x3)	Filter W0 (3x3x3)	Filter W1 (3x3x3)	Output Volume (3x3x2)
$x[:, :, 0]$	$w0[:, :, 0]$	$w1[:, :, 0]$	$o[:, :, 0]$
0 0 0 0 0 0 0 0 0 0 1 0 2 0 0 1 0 2 0 1 0 0 1 0 2 2 0 0 0 2 0 0 2 0 0 0 2 1 2 2 0 0 0 0 0 0 0 0 0	-1 0 1 0 0 1 1 -1 1 -1 0 1 1 -1 1 0 1 0 1 1 1 1 1 0 0 -1 0 0 0 0	0 1 -1 0 -1 0 0 -1 1 -1 0 0 1 -1 0 1 -1 0 -1 1 -1 0 -1 -1 1 0 0	2 3 3 3 7 3 8 10 -3 -8 -8 -3 -3 1 0 -3 -8 -5
$x[:, :, 1]$	$w0[:, :, 1]$	$w1[:, :, 1]$	$o[:, :, 1]$
0 0 0 0 0 0 0 0 2 1 2 1 1 0 0 2 1 2 0 1 0 0 0 2 1 0 1 0 0 1 2 2 2 2 0 0 0 1 2 0 1 0 0 0 0 0 0 0 0	1 -1 1 1 1 1 0 -1 0 1 1 1 1 1 0 0 -1 0 1 1 1 1 1 0 0 -1 -1	-1 0 0 1 -1 0 1 -1 0 -1 1 -1 0 -1 -1 1 0 0	-8 -8 -3 -3 1 0 -3 -8 -5
$x[:, :, 2]$	$w0[:, :, 2]$	$w1[:, :, 2]$	$o[:, :, 2]$
0 0 0 0 0 0 0 0 2 1 1 2 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 2 1 0 0 0 2 2 1 1 1 0 0 0 0 0 0 0 0	1 1 1 1 1 0 0 -1 0 1 1 1 1 1 0 0 -1 0 1 1 1 1 1 0 0 -1 -1	1 0 0 0 -1 -1 1 0 0	0
Bias b0 (1x1x1)	b0[:, :, 0]	toggle movement	
	1		

-75%

2x2 receptive fields

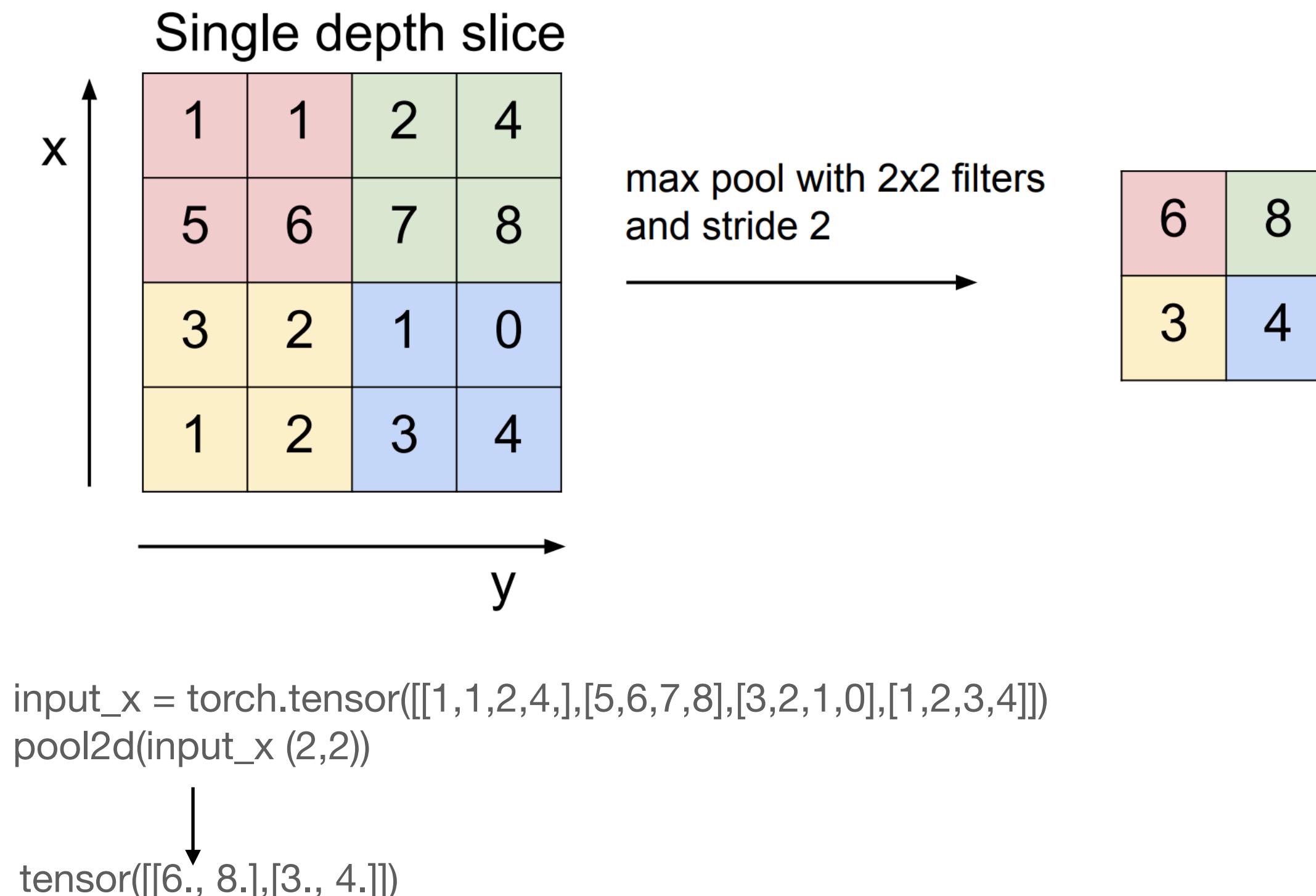
CNN

Pooling

- No parameters (there is no kernel)
- deterministic
- robustness

```
from d2l import torch as d2l
import torch
from torch import nn

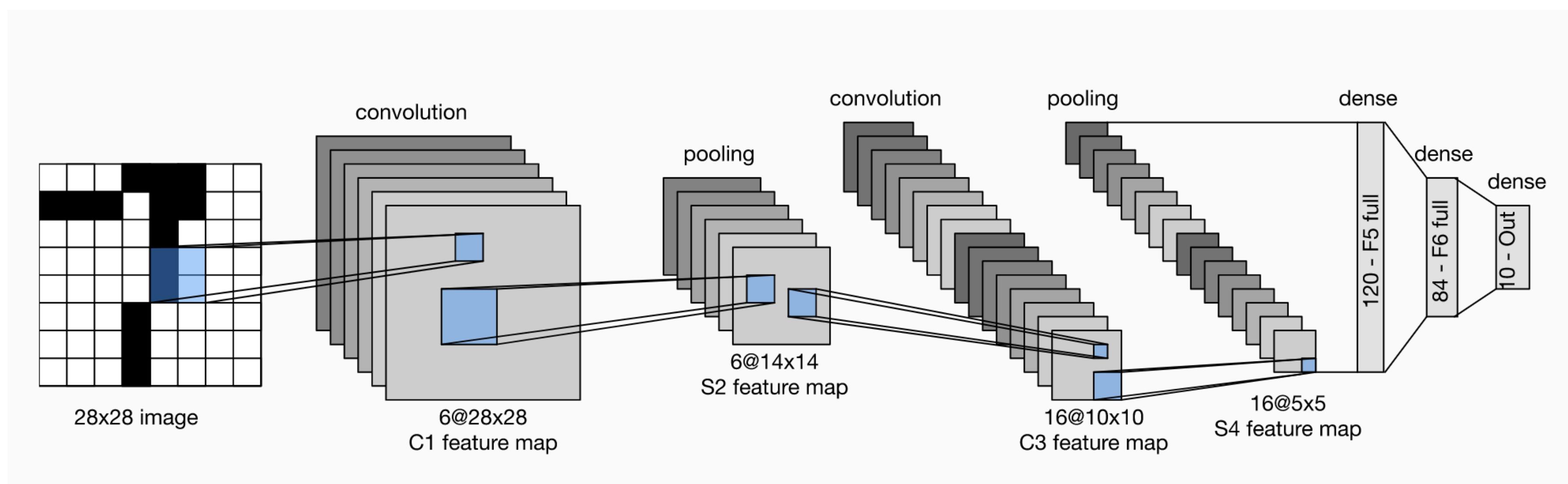
def pool2d(X, pool_size, mode='max'):
    p_h, p_w = pool_size
    Y = torch.zeros((X.shape[0] - p_h + 1, X.shape[1] - p_w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            if mode == 'max':
                Y[i, j] = X[i: i + p_h, j: j + p_w].max()
            elif mode == 'avg':
                Y[i, j] = X[i: i + p_h, j: j + p_w].mean()
    return Y
```



CNN

LeNet

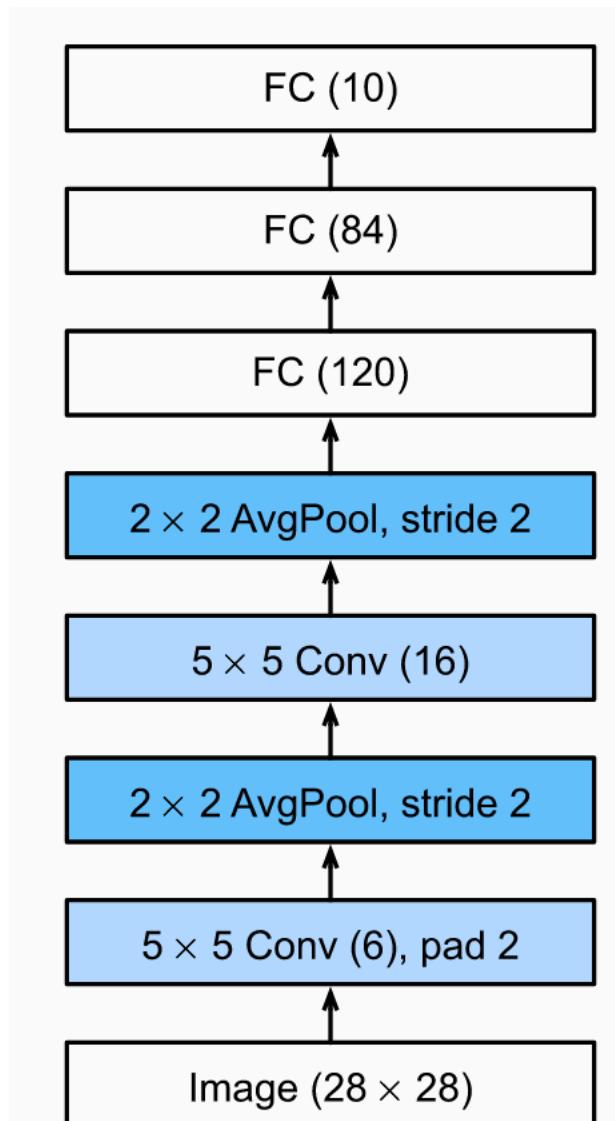
- The first successful applications of convolutional networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.



5x5 kernel

average pooling

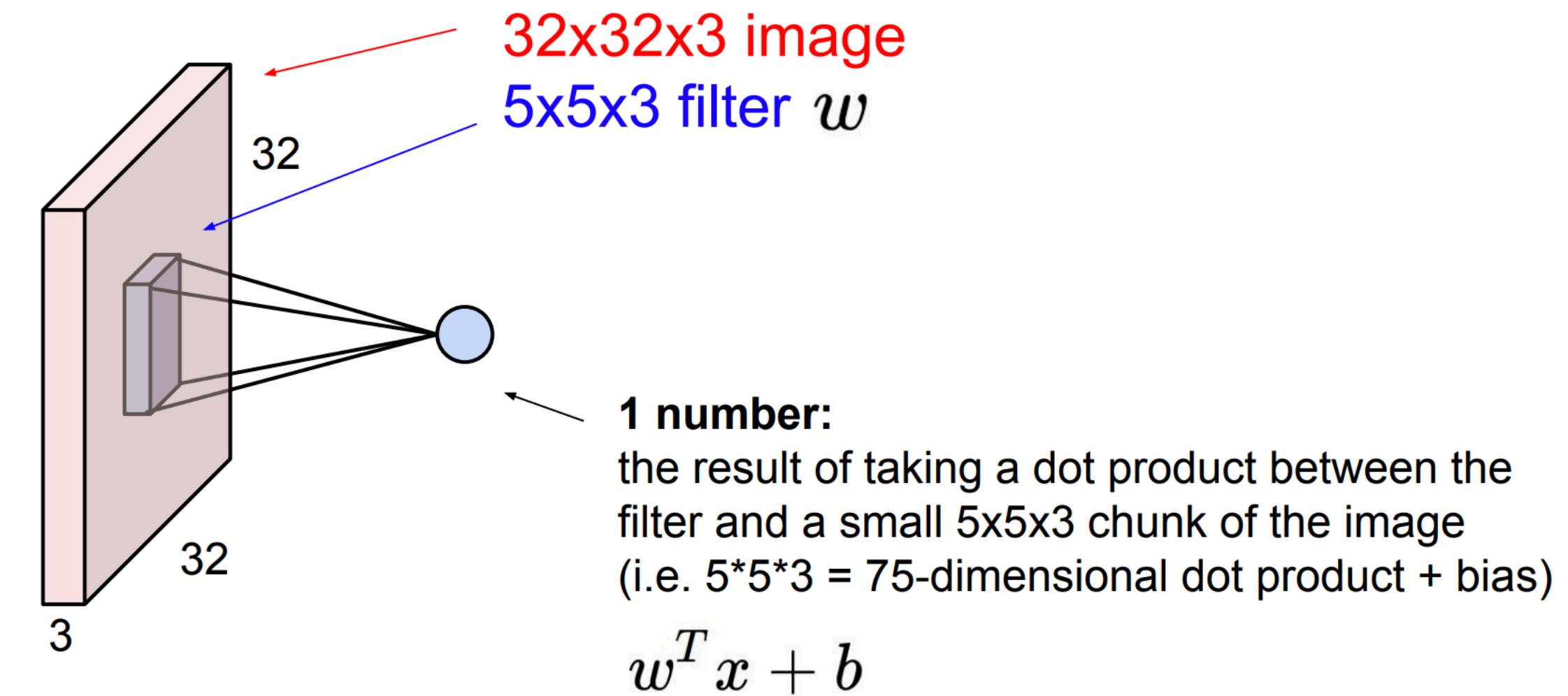
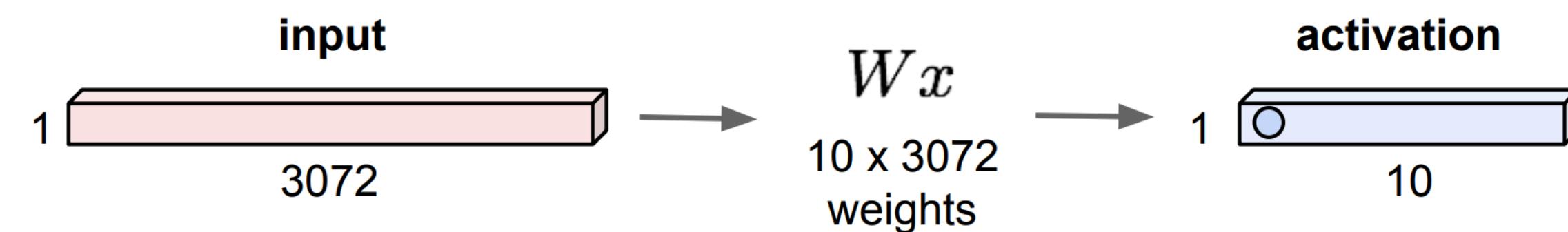
sigmoid



FC vc Conv

Why Conv is better?

- Fully connected layer
 - loss spatial information
 - massive weights
- Convolution layer
 - preserve spatial structure
 - extremely small weights



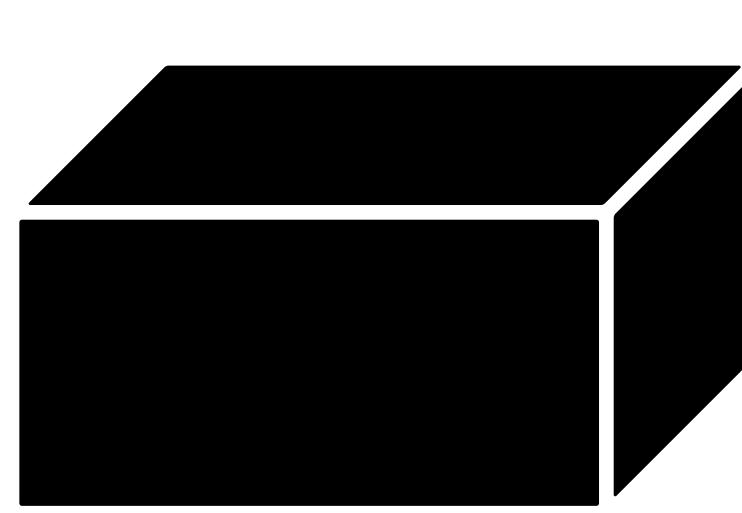
CNN

1 x 1 Convolution layer

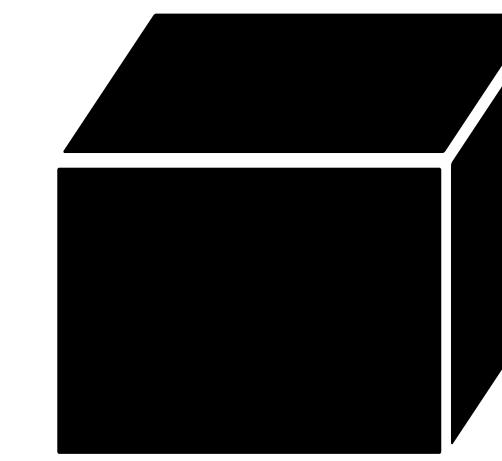
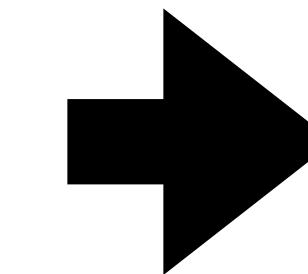
Managing channel

Efficient

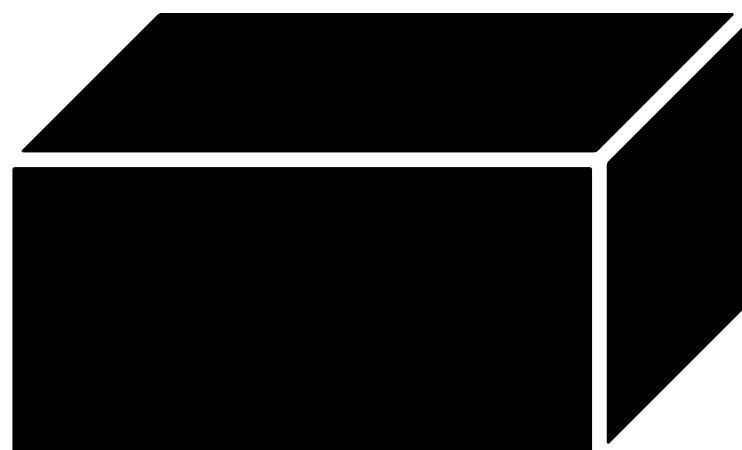
Non-linearity



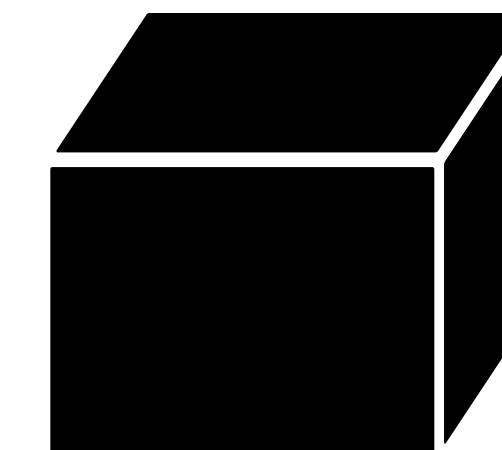
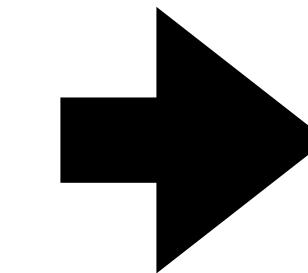
Conv 5x5 (64)



(128,28,28)



Conv 1x1 (32)

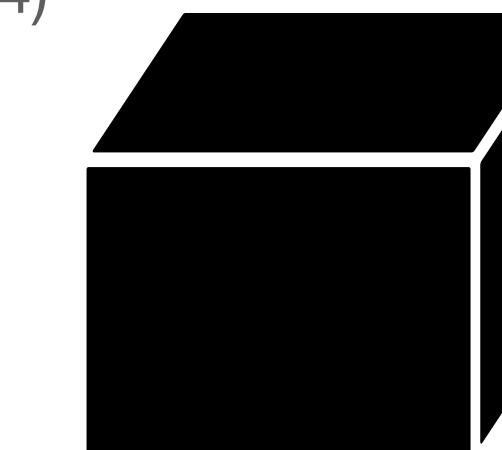
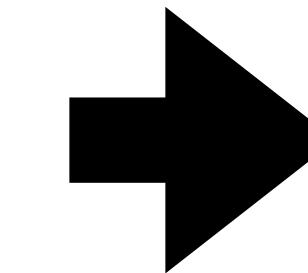


(128,28,28)

of params: $28*28*32*1*1*128 = 4.8M$

of params: $28*28*64*5*5*128 = 160M$

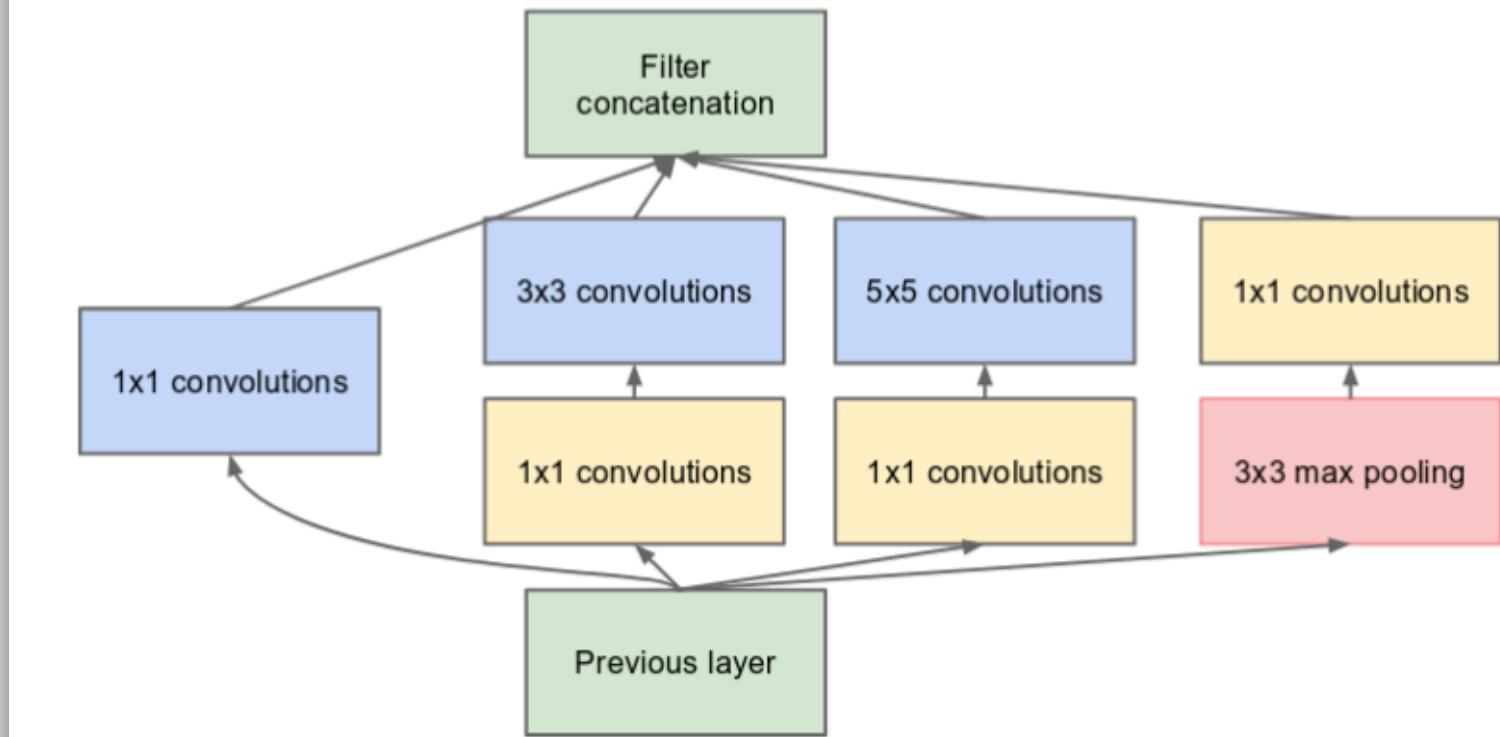
Conv 5x5 (64)



(64,28,28)

of params: $28*28*64*5*5*32 = 40M$

GoogLeNet (Inception module)

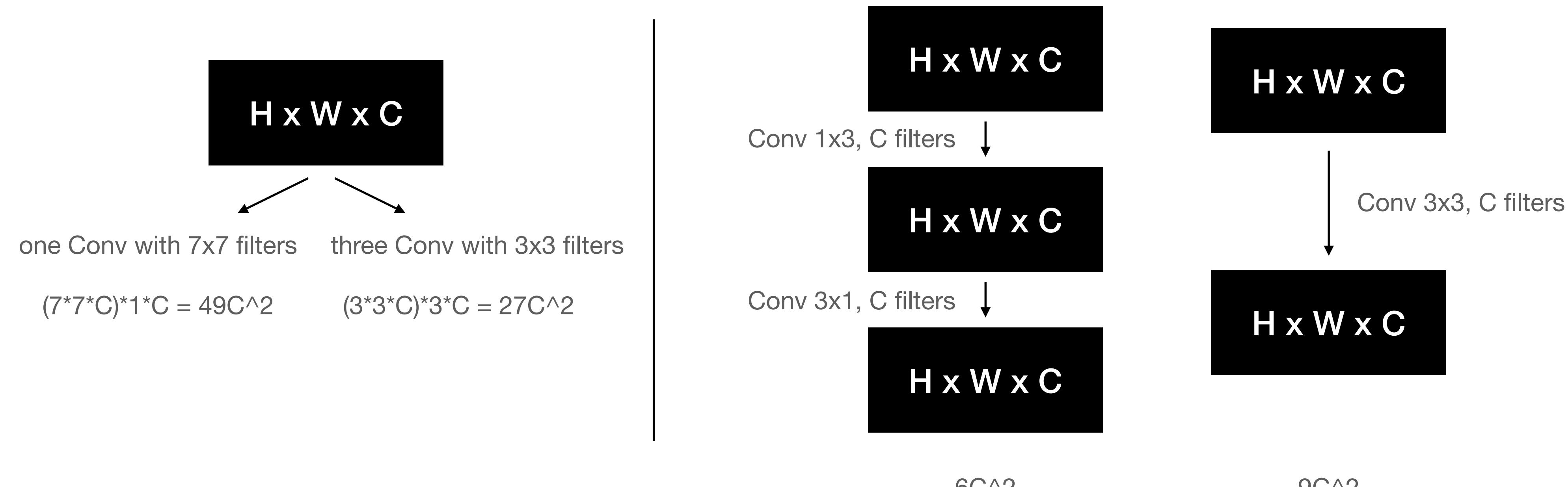


(b) Inception module with dimension reductions

total: 44.8M

CNN

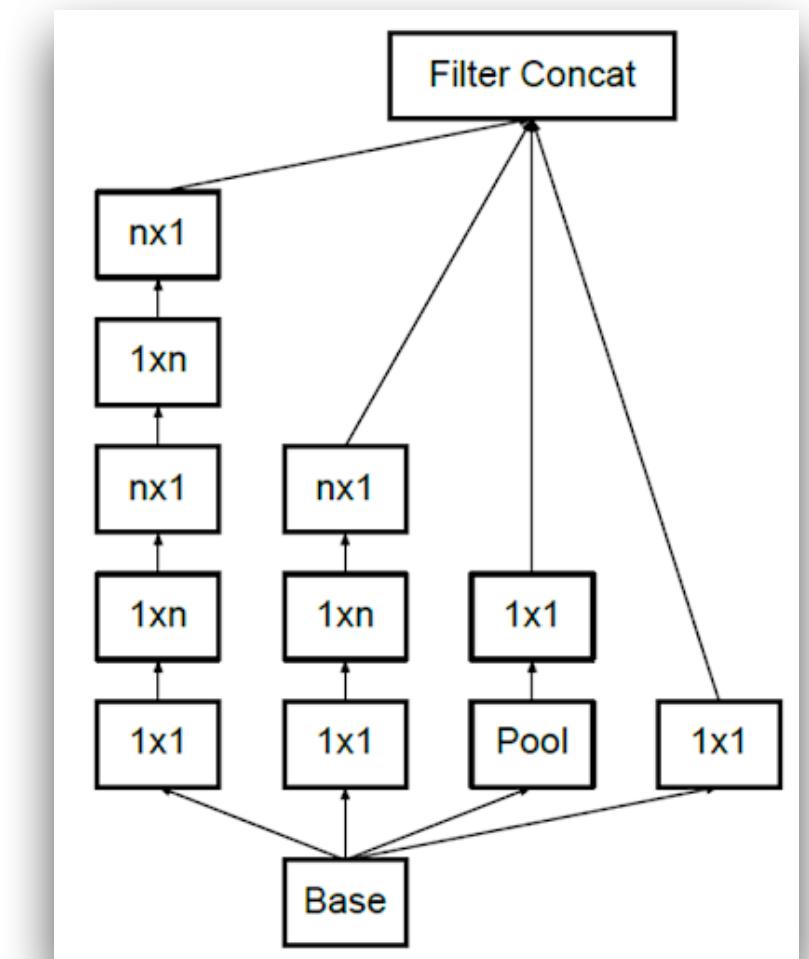
The power of small filters



$$\text{num_weights} = (\text{filter_width} * \text{filter_height} * \text{num_channel}) * \text{num_filters}$$

More non-linearity
fewer params
less compute

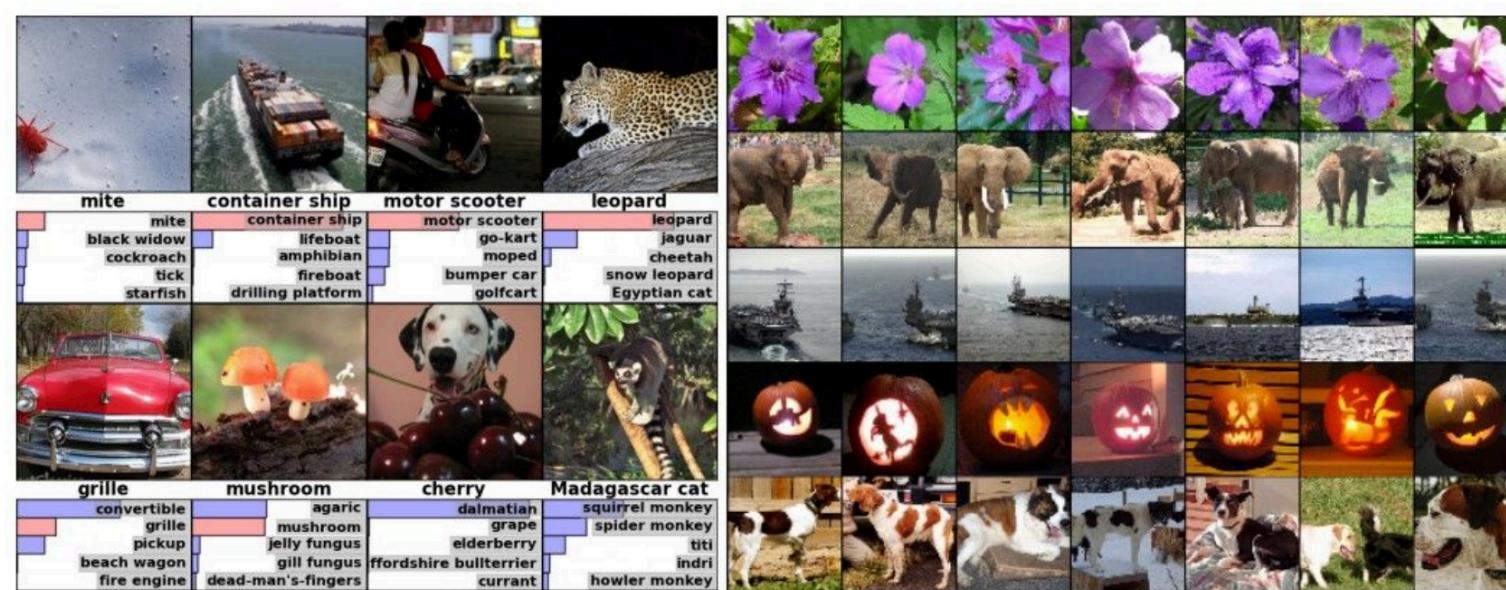
Why not try 1×1 ?



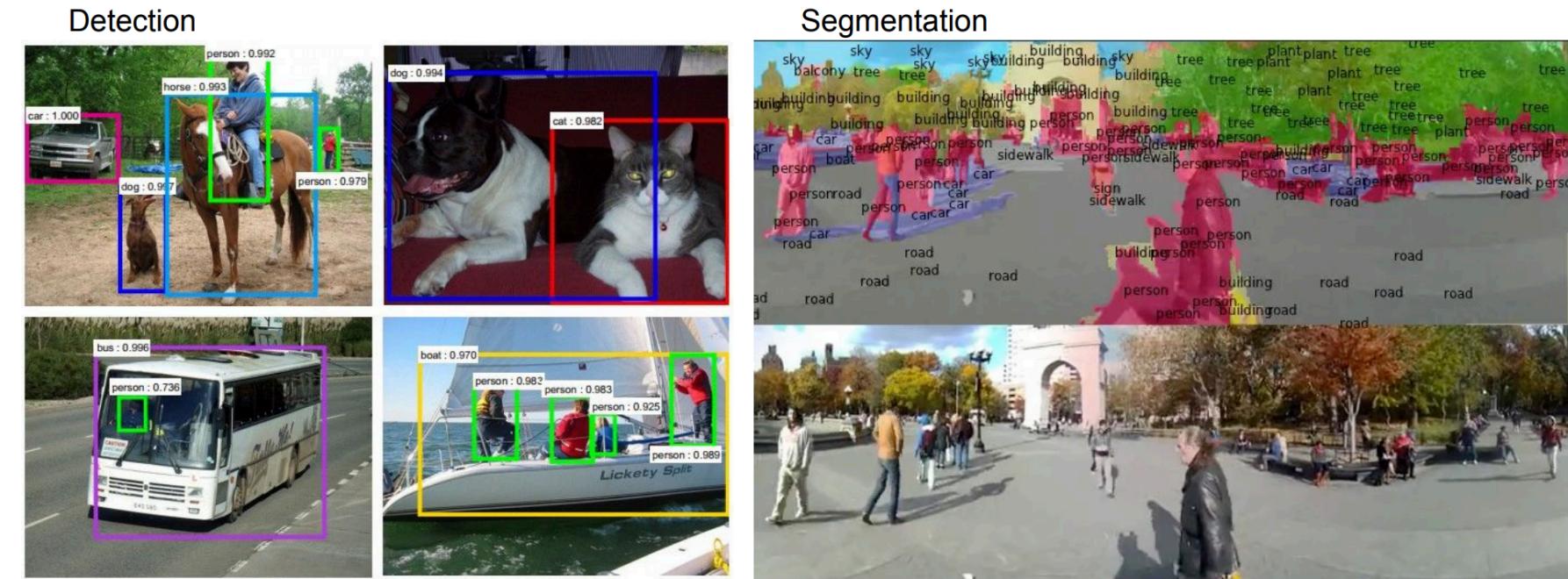
CNN

ConvNets are everywhere

Classification



Retrieval



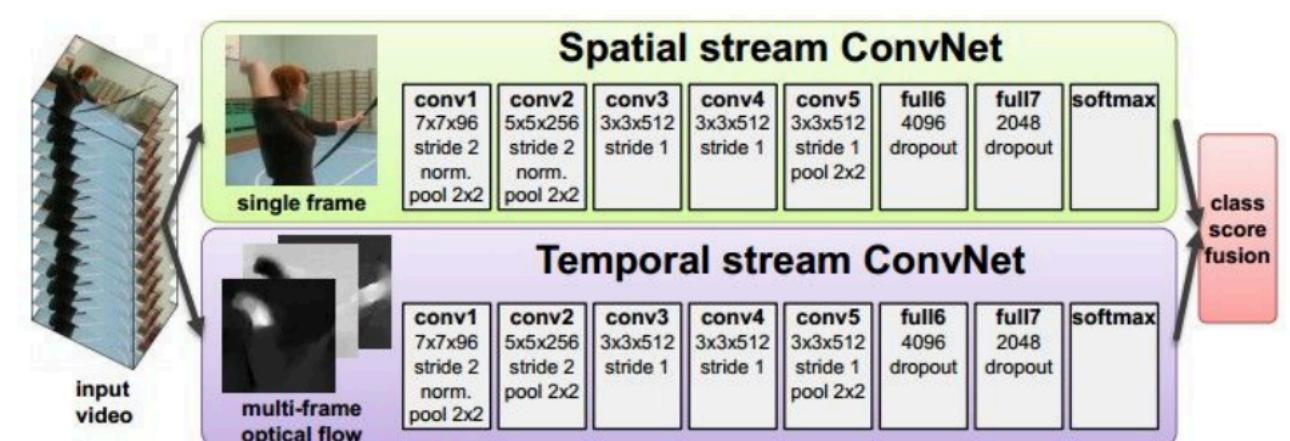
A photograph taken from the driver's seat of a car, looking down a two-lane road. The road is flanked by rows of tall palm trees. In the center lane, a dark-colored car is positioned ahead, with a yellow rectangular box drawn around it and the word "car" written above it. To the right side of the road, two figures are standing on a sidewalk; they are also enclosed in a yellow rectangular box, with the word "pedestrians" written above them. The interior of the car is visible at the bottom of the frame, showing the dashboard and a rearview mirror. The sky is clear and blue.



*A white teddy bear sitting in
the grass*



A man riding a wave on top of a surfboard



CNN

summary

- ConvNets stack CONV, POOL, FC layers
- Historically architectures looked like
[(CONV-RELU)*N - POOL?] * M - (FC-RELU)*K, SOFTMAX
where N is usually up to ~5, M is large, $0 \leq k \leq 2$
- but recent advances such as ResNet/GoogLeNet have challenged this paradigm
- ConvNetJS demo: training on CIFAR-10
<https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>