14.332.435/16.332.530
Introduction to Deep Learning

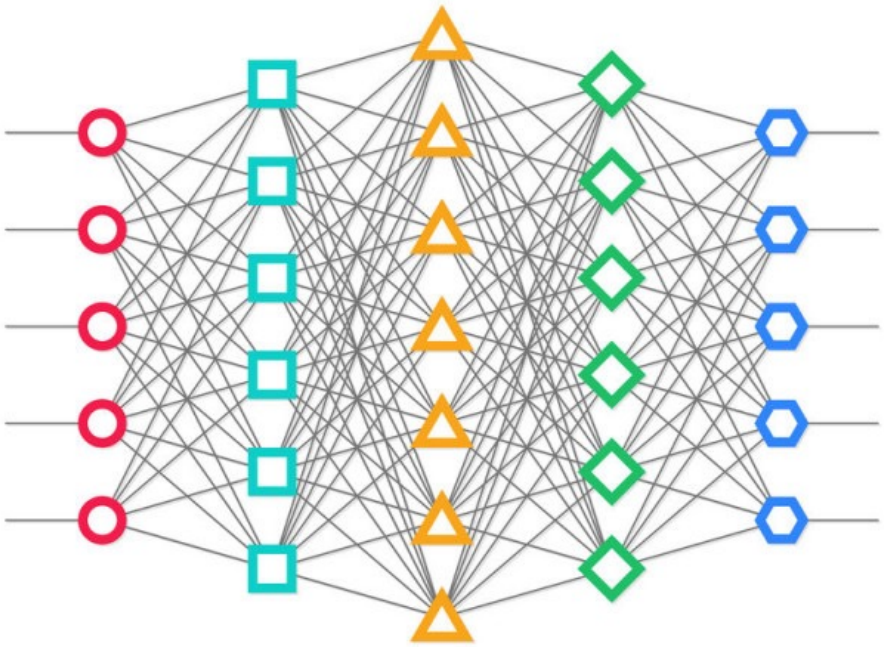
Lecture 7
Convolutional Neural Network

Yuqian Zhang

Department of Electrical and Computer Engineering

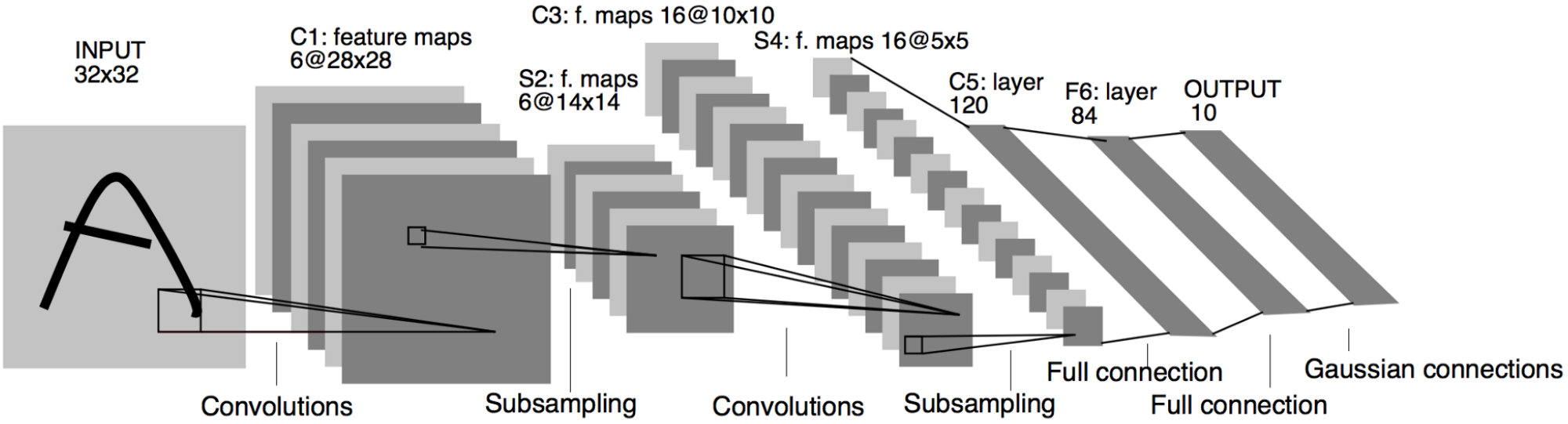
Recall Lecture 5

- Fully-connected Neural Network



Today's Agenda

- Convolutional Neural Network



History

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

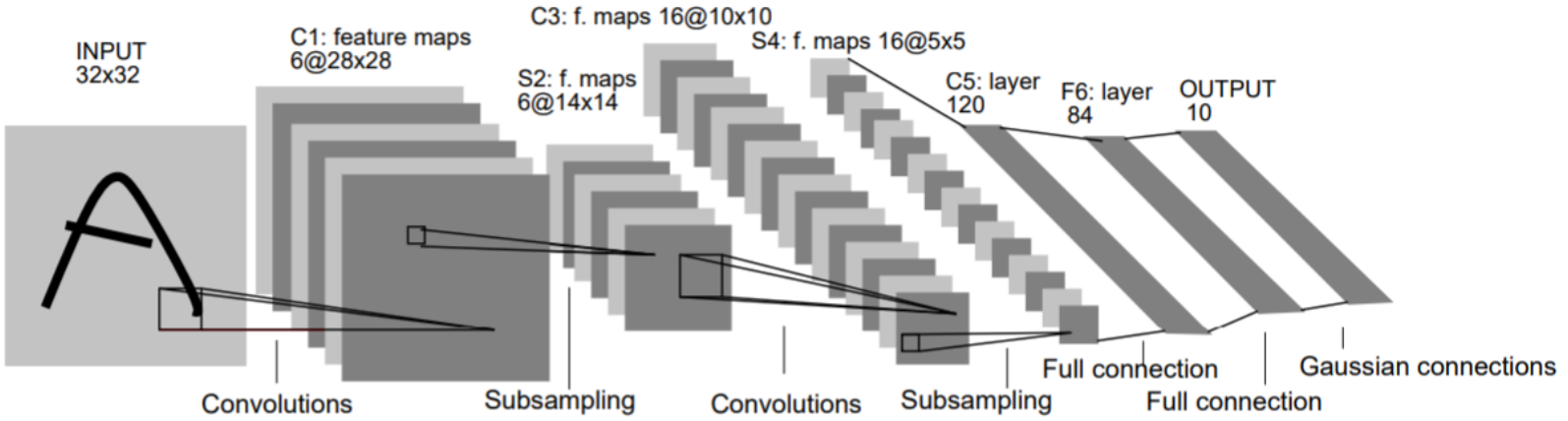


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

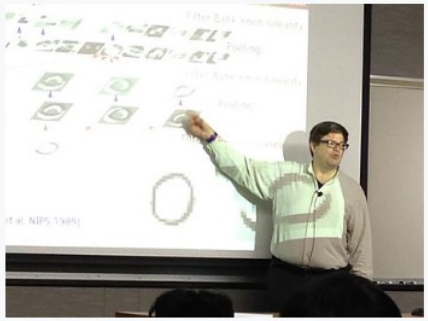
Gradient-based learning applied to document recognition - IEEE Xplore

https://ieeexplore.ieee.org/document/726791

by Y Lecun - 1998 - Cited by 16398 - Related articles

Yann LeCun (/ləˈkʌn/;[1] born 1960) is a French computer scientist working primarily in the fields of machine learning, computer vision, mobile robotics, and computational neuroscience. He is the Chief Artificial Intelligence Scientist at Facebook AI Research, and he is well known for his work on optical character recognition and computer vision using convolutional neural networks (CNN), and is a founding father of convolutional nets.[2][3] He is also one of the main creators of the DjVu image compression technology (together with Léon Bottou and Patrick Haffner). He co-developed the Lush programming language with Léon Bottou.

Yann LeCun



Born	July 8, 1960 (age 58)
Alma mater	Pierre and Marie Curie University
Known for	Deep learning
	Scientific career
Institutions	Bell Labs New York University Facebook Artificial Intelligence Research
Thesis	*Modèles connexionnistes de l'apprentissage (connectionist learning models)* (1987)

https://www.youtube.com/watch?v=FwFduRA_L6Q

uTube Search

Convolutional Network Demo from 1993

63,207 views 483 2 SHARE SAVE ...

Yann LeCun
Published on Jun 2, 2014

SUBSCRIBE 864

This is a demo of "LeNet 1", the first convolutional network that could recognize handwritten digits with good speed and accuracy.

It was developed between 1988 and 1993 in the Adaptive System Research Department, headed by Larry Jackel, at Bell Labs in Holmdel, NJ.

This "real time" demo shows ran on a DSP card sitting in a 486 PC with a video camera and frame grabber card. The DSP card had an AT&T DSP32C chip, which was the first 32-bit floating-point DSP and could reach an amazing 12.5 million multiply-accumulate operations per second.

Shortly after this demo was put together, we started working with a development group and a product group at NCR (then a subsidiary of AT&T). NCR soon deployed ATM machines that could read the numerical amounts on checks, initially in Europe and then in the US. The ConvNet was running on the DSP32C card sitting in a PC inside the ATM. Later, NCR deployed a similar system in large check reading machines that banks use in their back offices. At some point in the late 90's these machines were processing 10 to 20% of all the checks in the US.

The network shown in this demo is described in our NIPS 1989 paper "Handwritten digit recognition with a back-propagation network".

The check reading system is described in our 1998 Proc. IEEE paper "Gradient-Based Learning Applied to Document Recognition" and in various shorter papers before that.

History

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

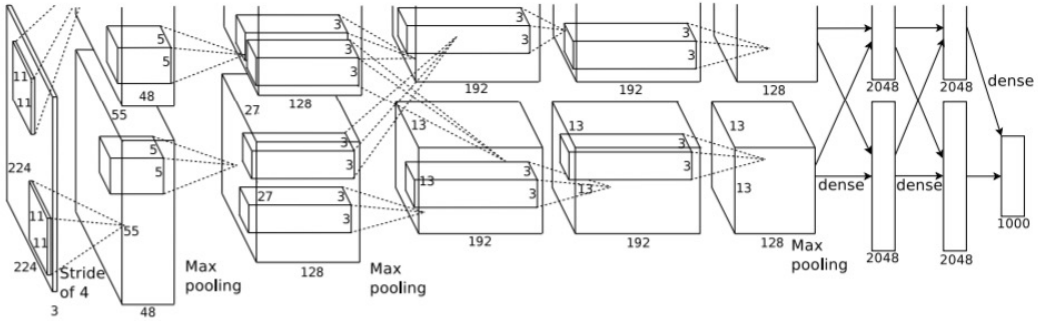


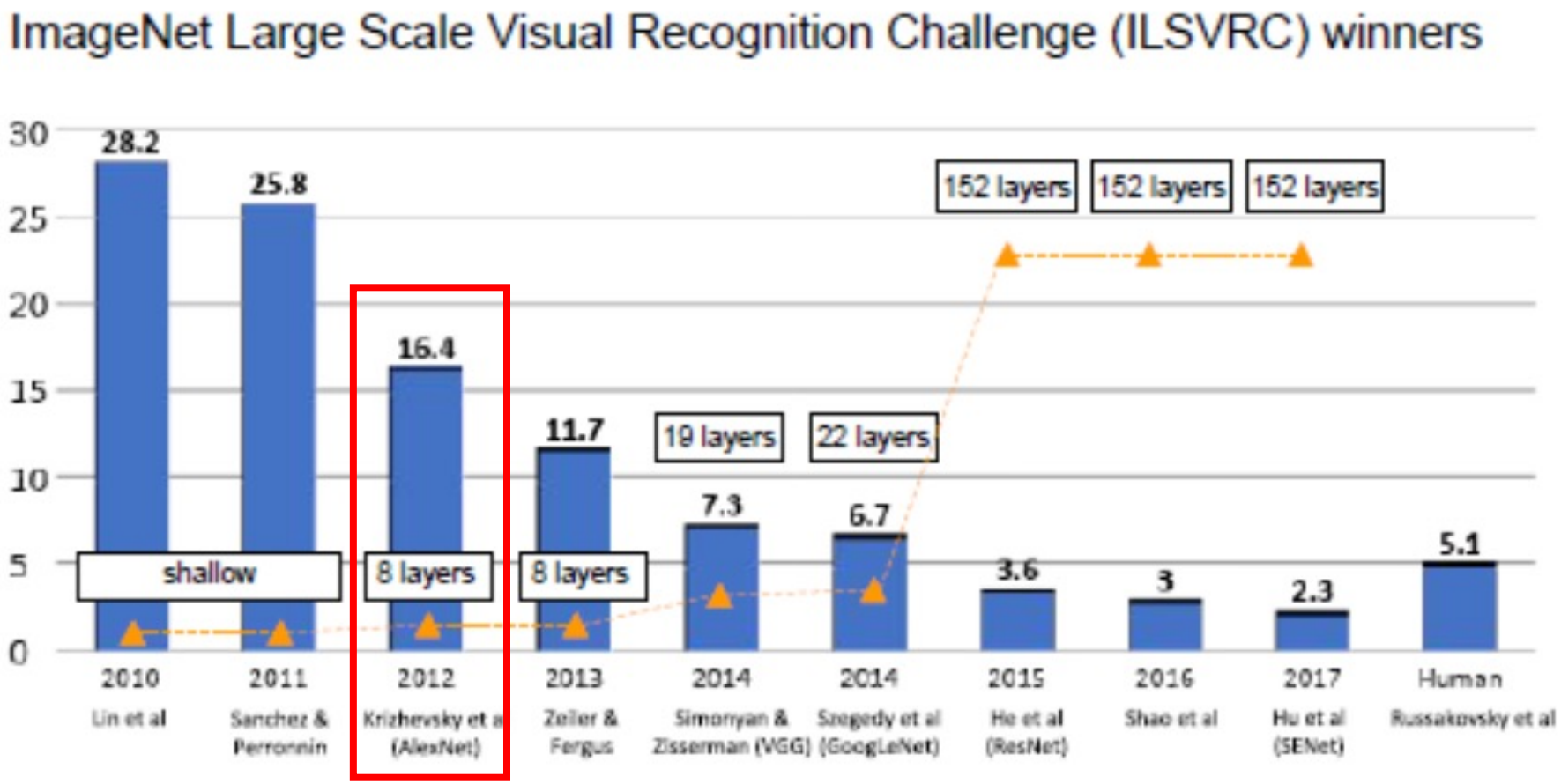
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

[PDF]

ImageNet Classification with Deep Convolutional ... - NIPS Proceedings

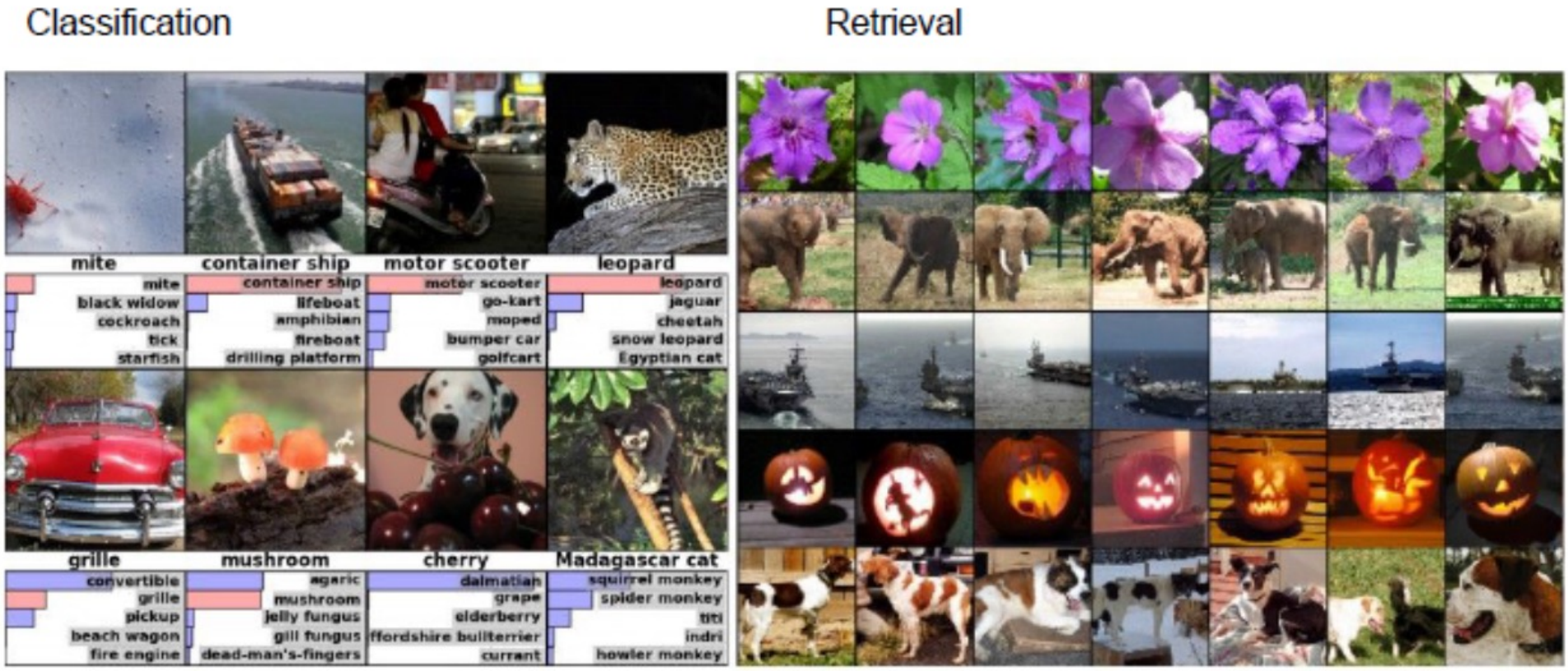
https://papers.nips.cc/.../4824-imagenet-classification-with-deep-convolutional-neural-...

by A Krizhevsky - 2012 - Cited by 34672 - Related articles



Russakovsky et al. IJCV 2015

Current Applications



Current Applications

Fast-forward to today: ConvNets are everywhere

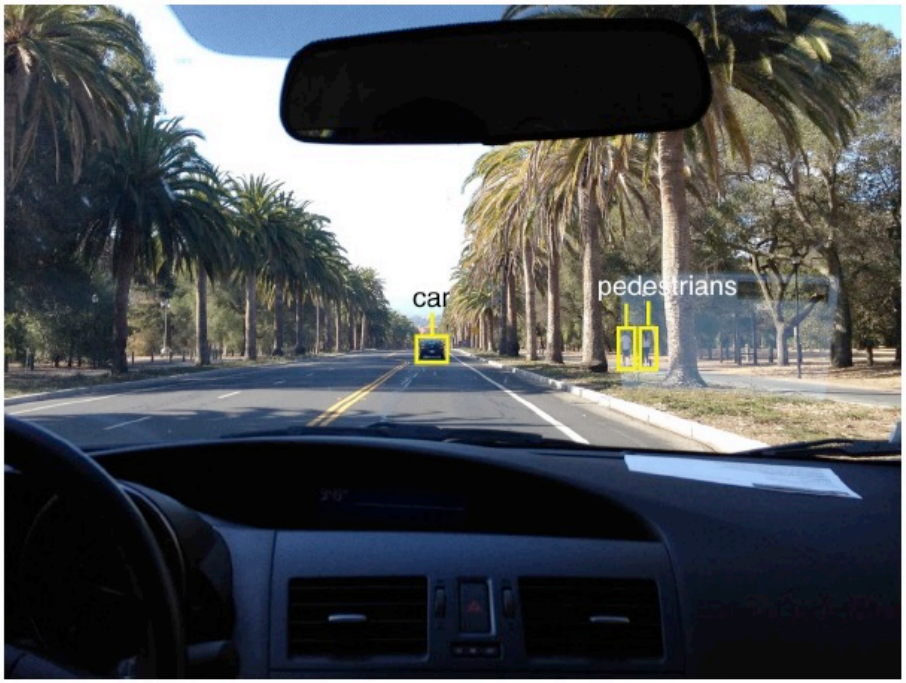


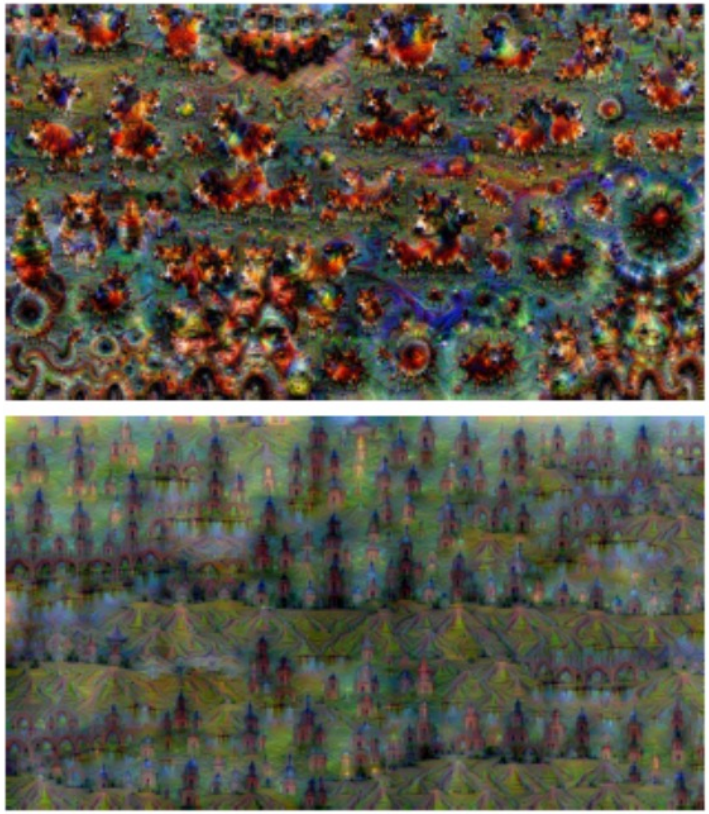
Photo by Lane McIntosh. Copyright CS231n 2017.

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.

Current Applications



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Current Applications

No errors

A white teddy bear sitting in the grass

Minor errors

A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

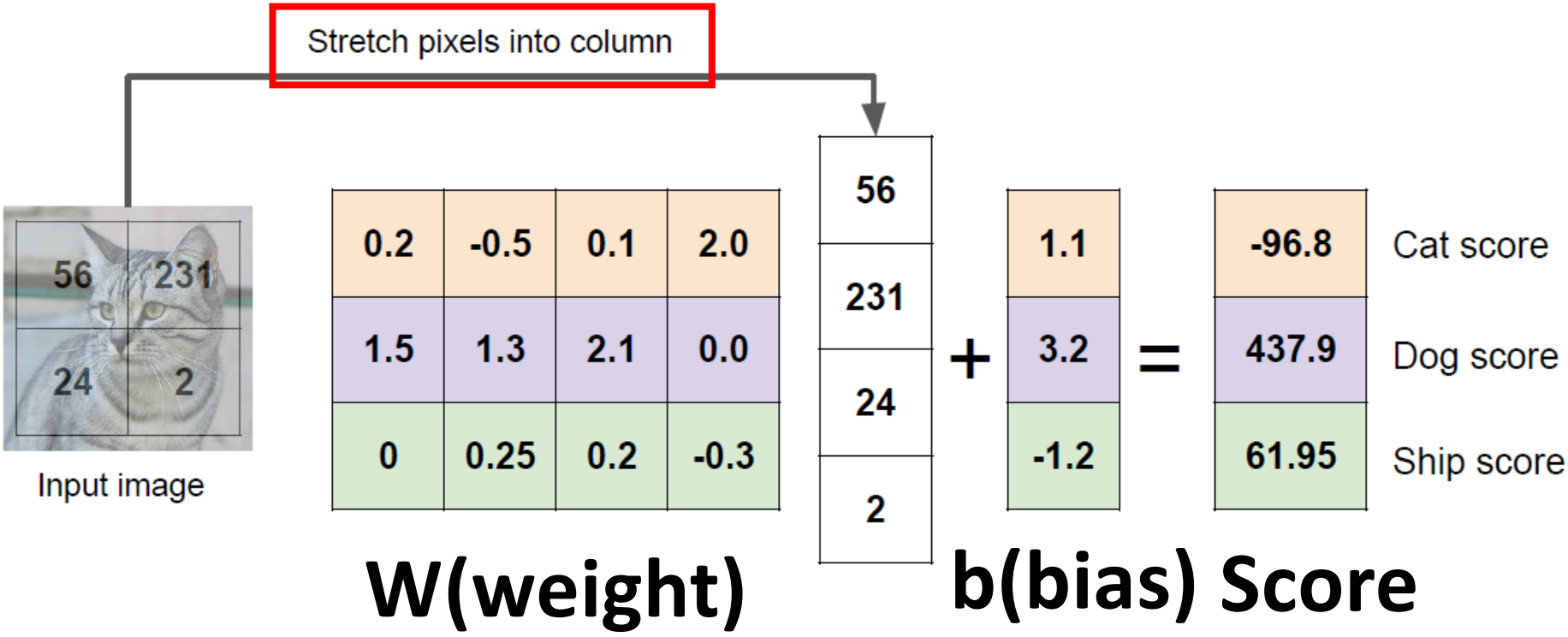
Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

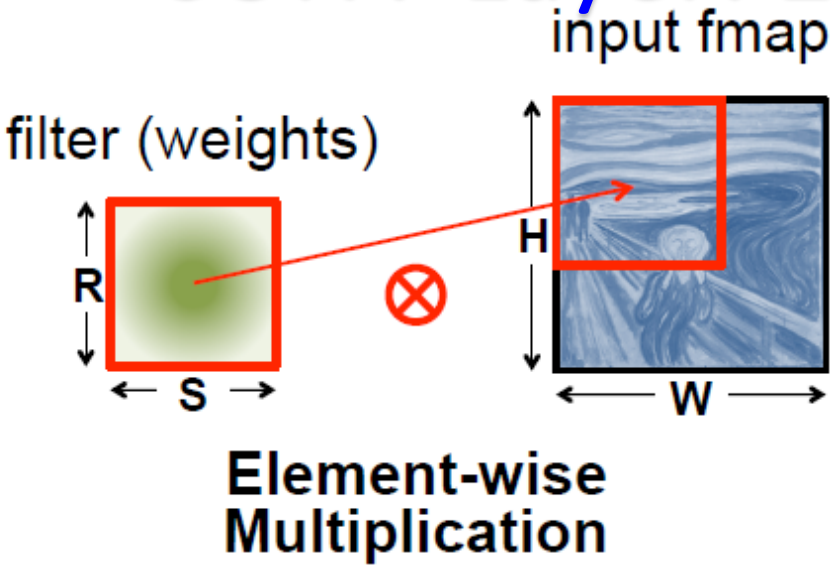
All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-pose-meditation-486008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

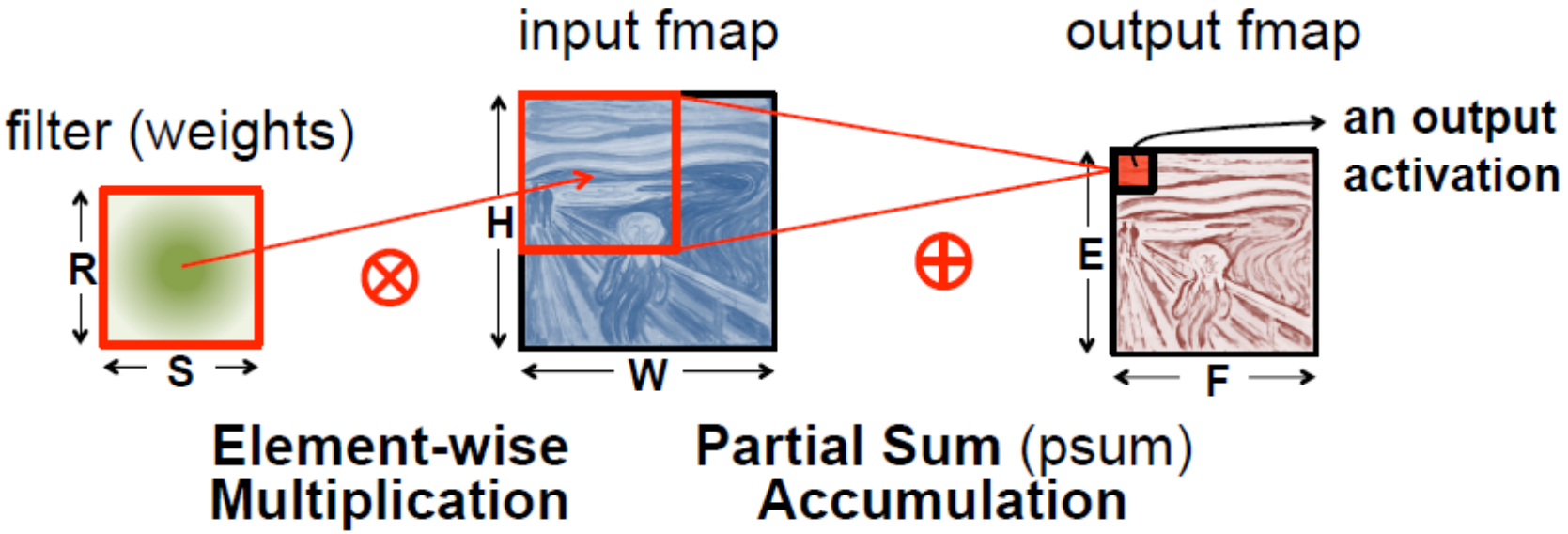
Captions generated by Justin Johnson using NeuralTalk2

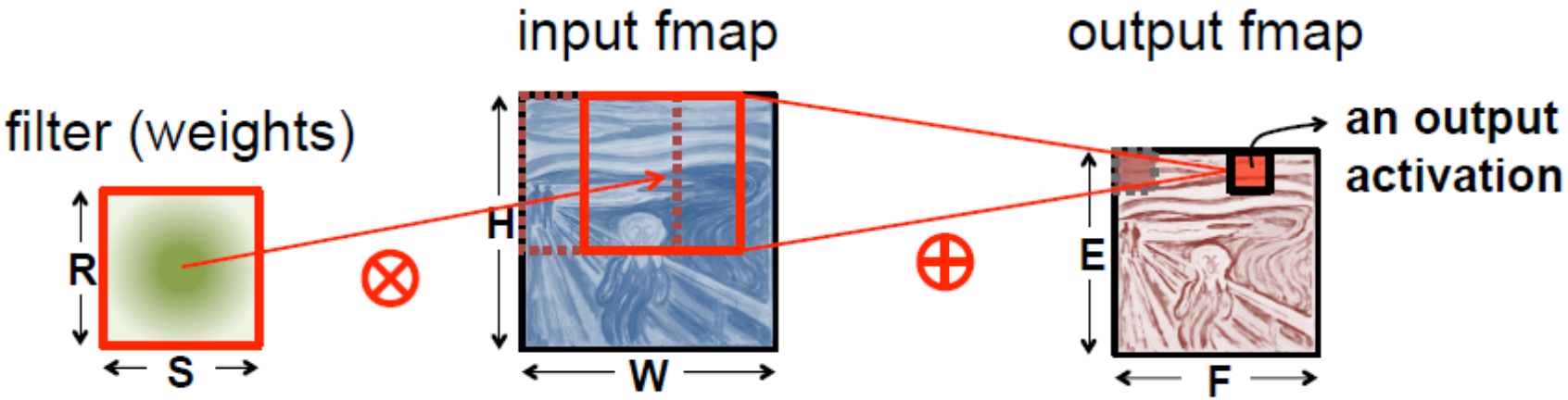
Recall in Lecture 2 for FC Layer



CONV Layer: 2D Operation

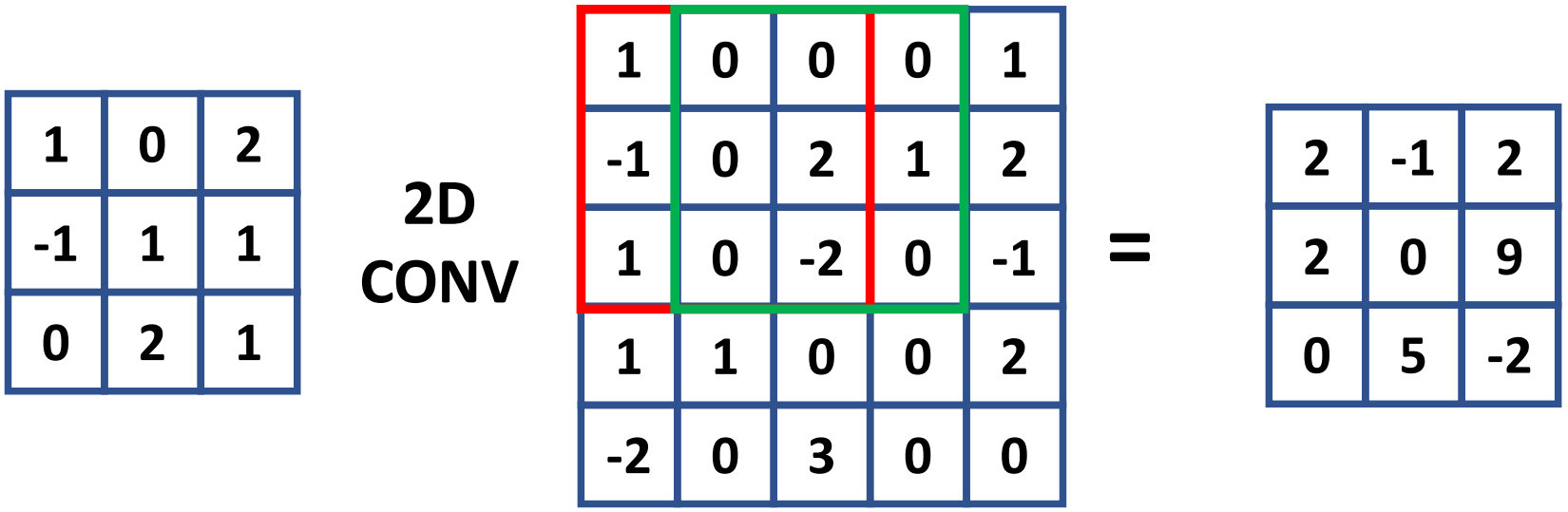



input fmap
output fmap
filter (weights)
an output activation
R
S
H
W
E
F
Element-wise Multiplication
Partial Sum (psum) Accumulation

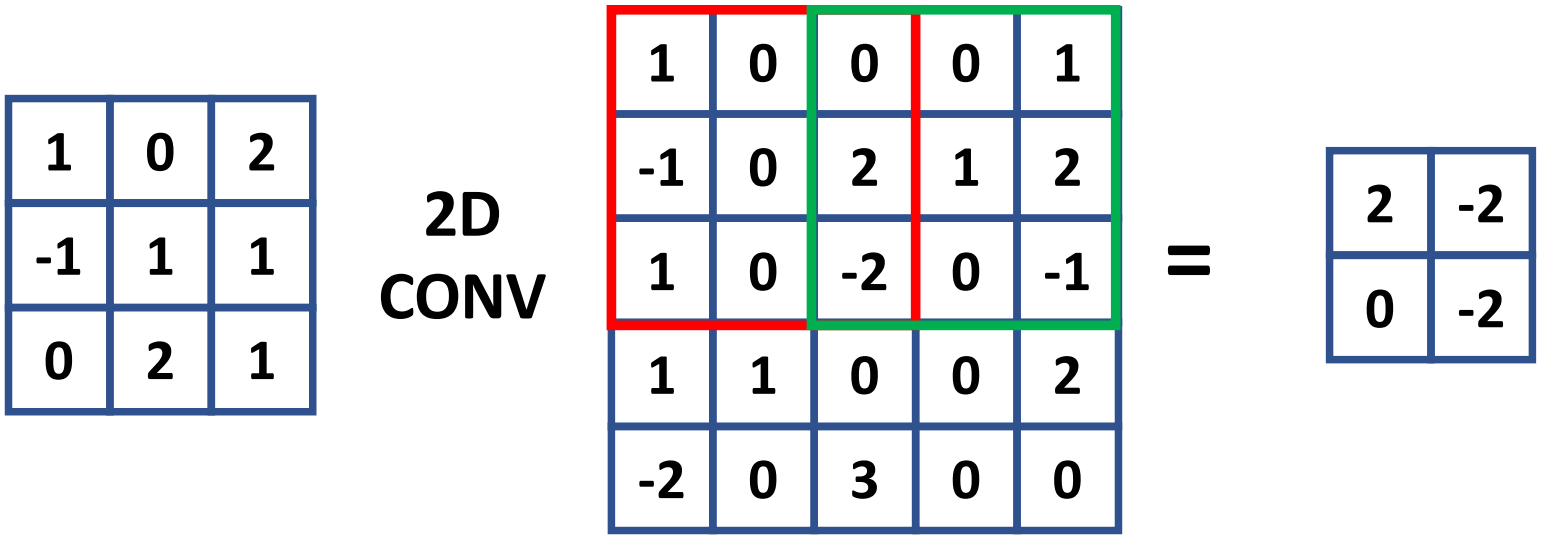


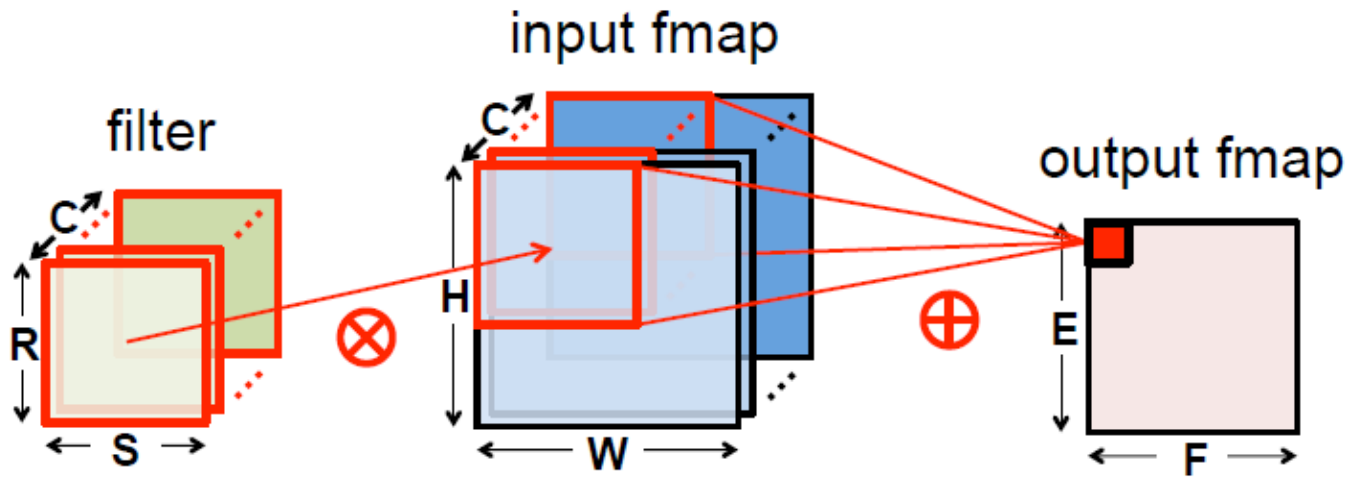
Sliding Window Processing

Practice Example (stride=1)



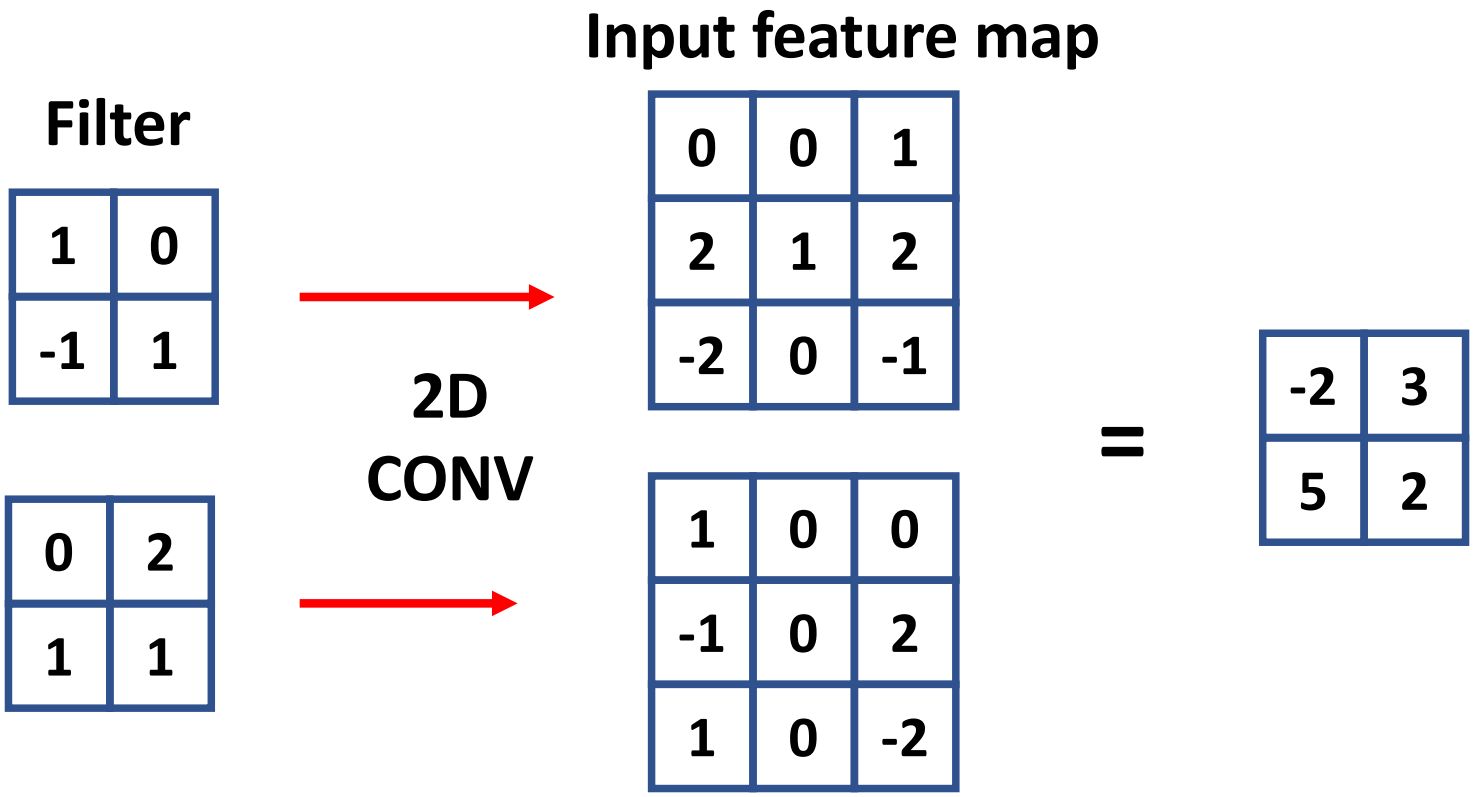
Practice Example (stride=2)

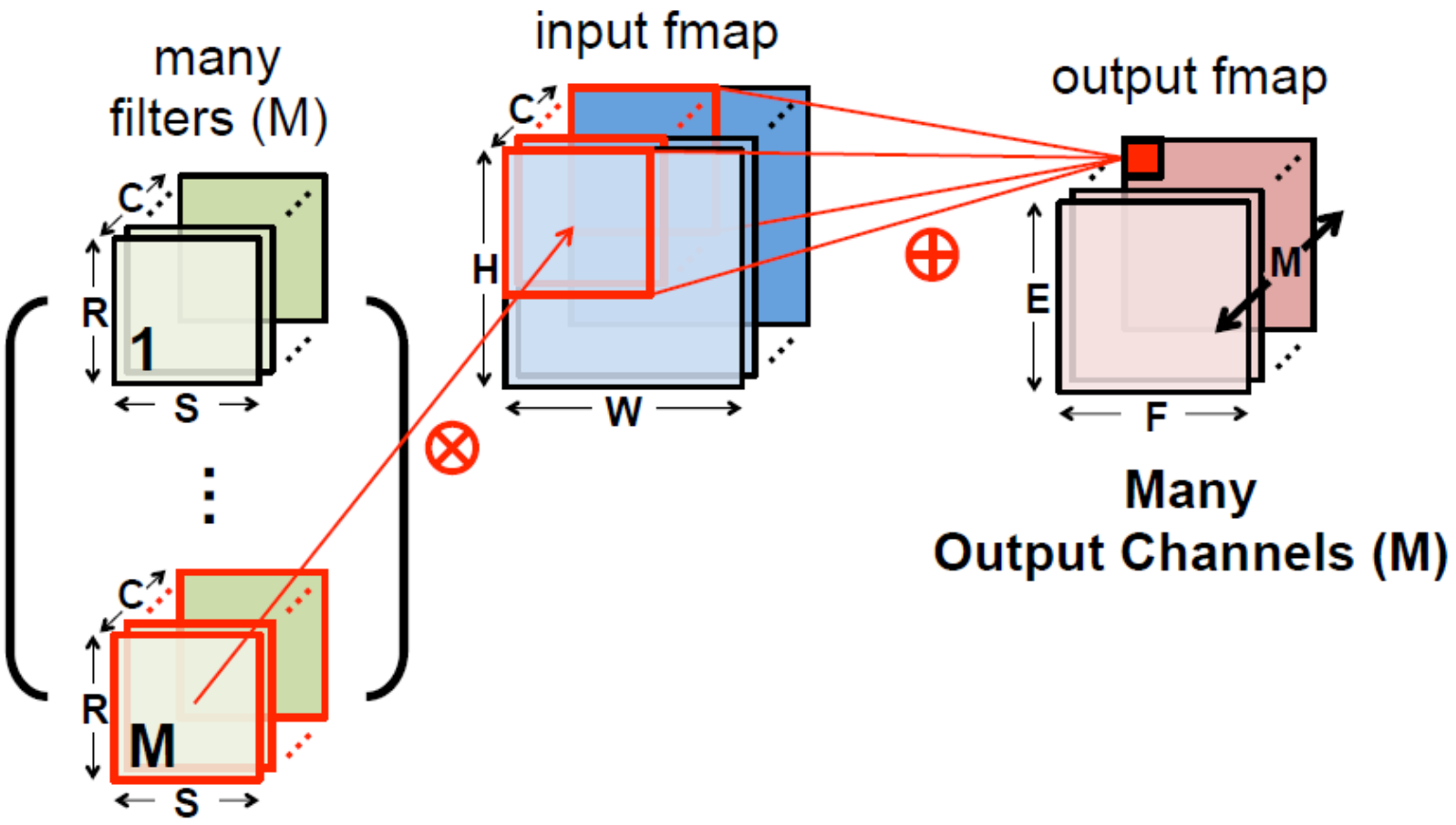


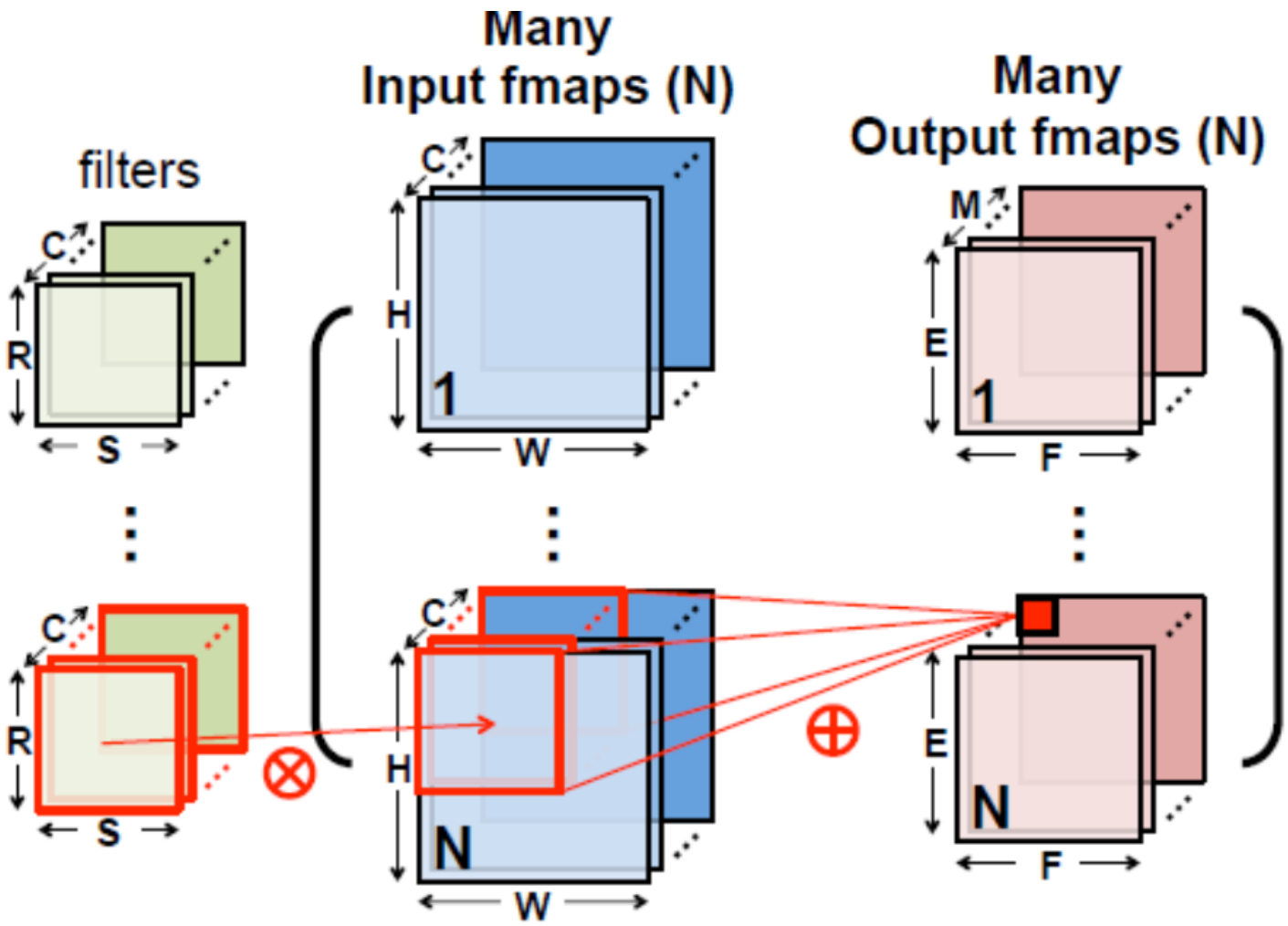


Many Input Channels (C)

Practice Example (Multi-input Channel)

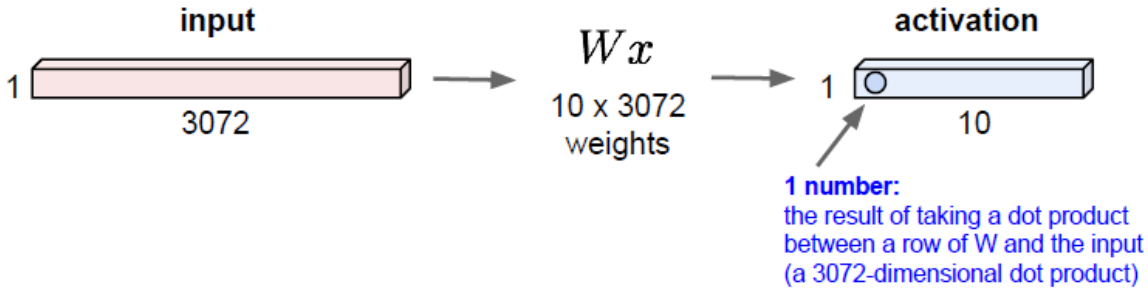


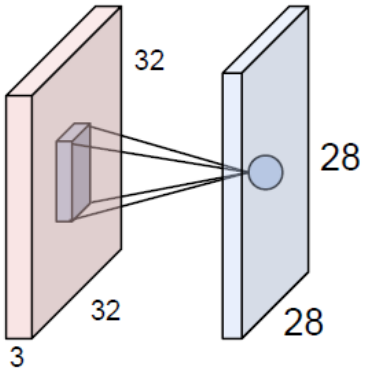

many
filters (M)
C
R
1
S
M
input fmap
H
W
output fmap
E
F
Many
Output Channels (M)


Many
Input fmaps (N)
Many
Output fmaps (N)
filters
C
R
S
H
W
1
M
E
F
N

FC vs CONV

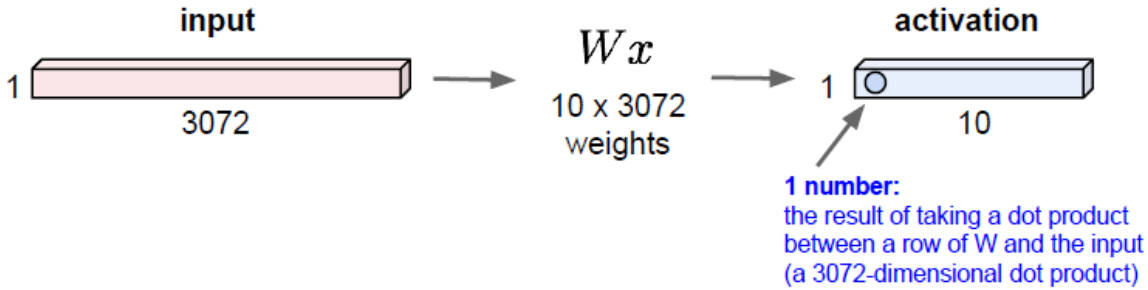
- # of weights and multiplications?



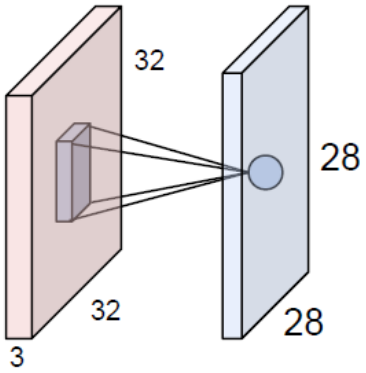


FC vs CONV

- # of weights and multiplications?

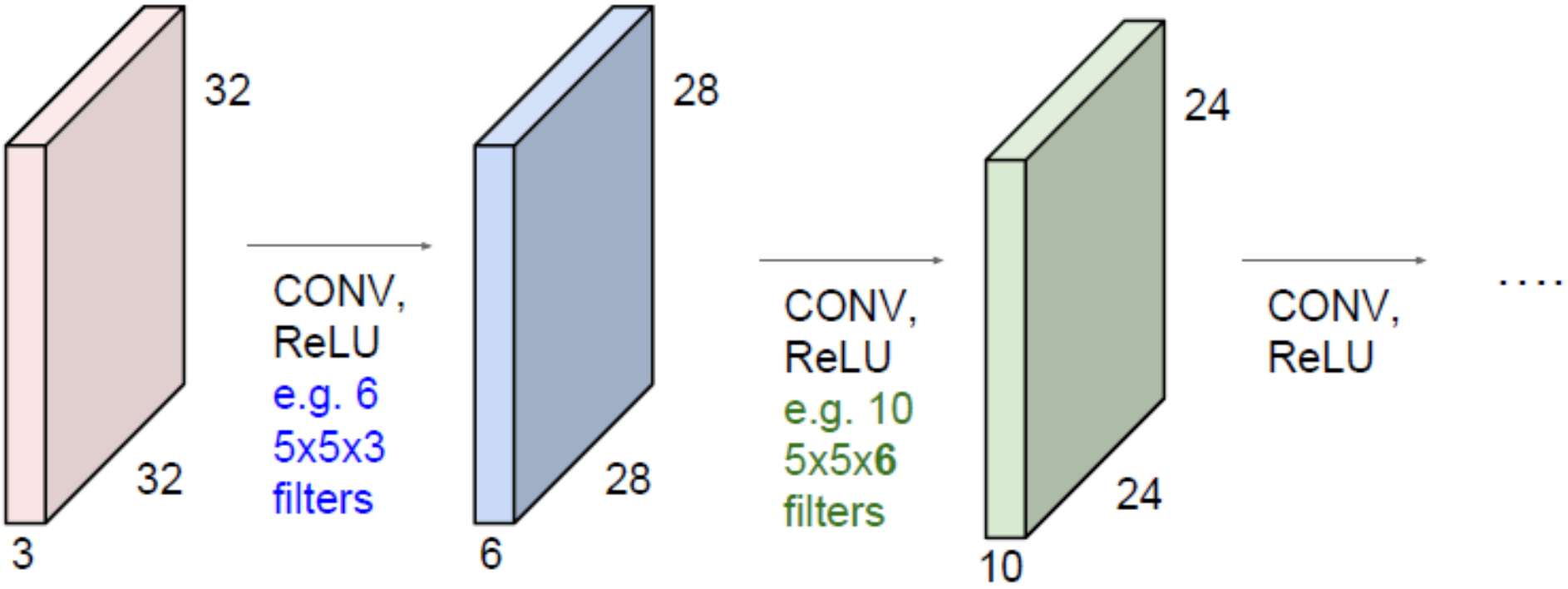


large # of weights
small # of computation



small # of weights
large # of computation

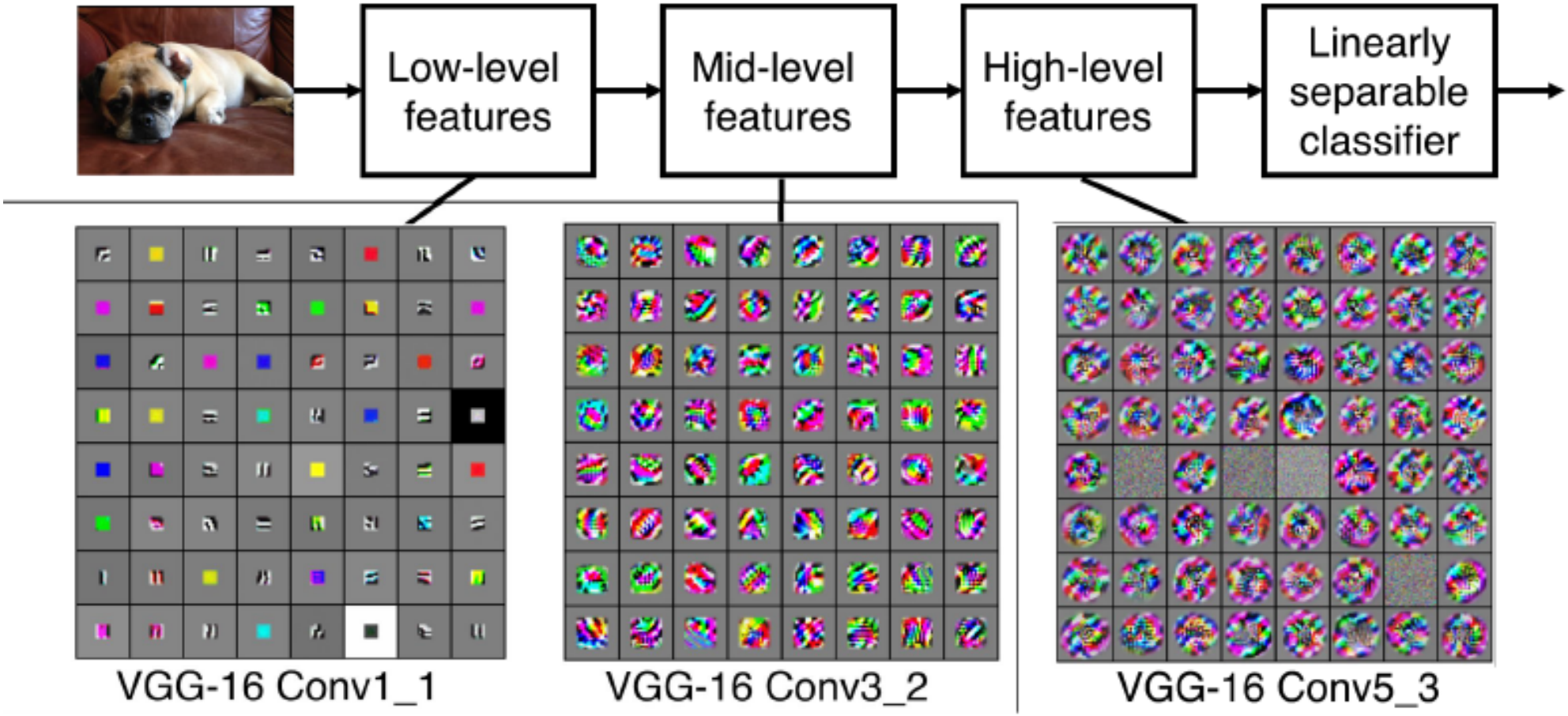
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Preview

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



Why CNN Successes in Comp. Vision

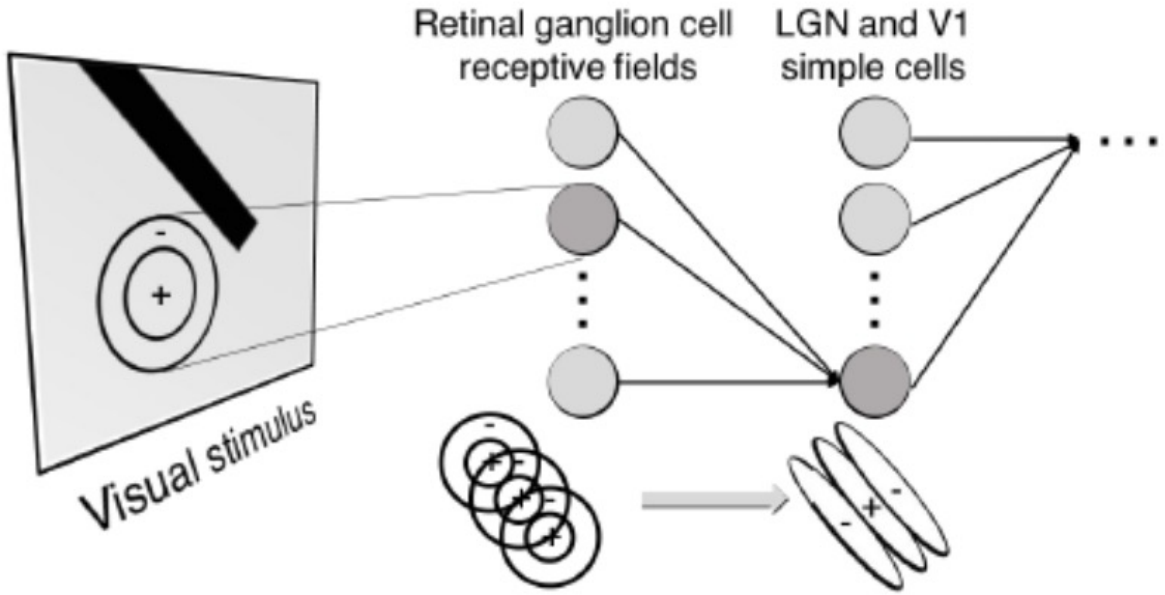
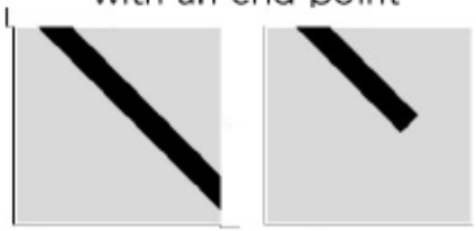


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

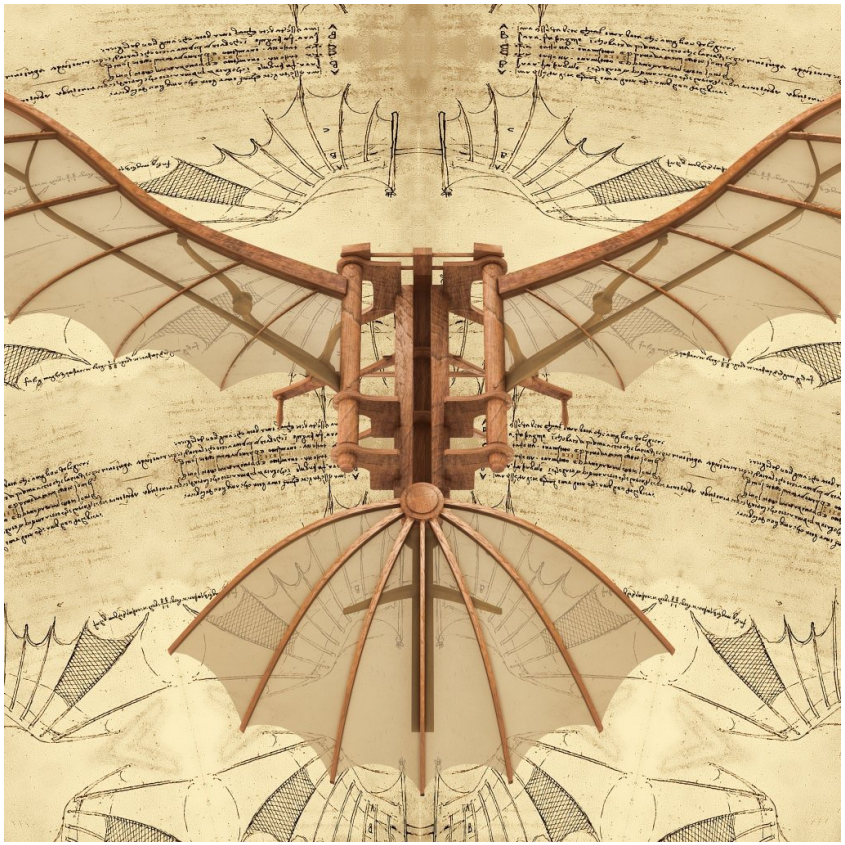
Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



Should We Mimic Brain's Behavior?



https://designformsofart.com

Debate: Deep NN vs Spiking NN

Yann LeCun
@ylecun

Follow

In my keynote at ISSCC, I said that I was skeptical about neuromorphic hardware efforts that rely on spiking neurons.
I have publically expressed skepticism about this... zdnet.com/google-amp/art ...

12:56 PM - 23 Feb 2019

26 Retweets **80** Likes

Intel's neuro guru slams deep learning: 'it's not actually learning'

Intel's director of its neuromorphic computing initiative, Mike Davies, chided Facebook's Yann LeCun at an industry conference for failing to appreciate the virtues of the Intel technology. He derided the deep learning approach of LeCun and others as failing to truly add up to "learning."

By Tiernan Ray | February 23, 2019 -- 18:22 GMT (10:22 PST) | Topic: Artificial Intelligence

"Backpropogation doesn't correlate to the brain," insists Mike Davies, head of Intel's neuromorphic computing unit, dismissing one of the key tools of the species of A.I. In vogue today, deep learning. "For that reason, "it's really an optimizations procedure, it's not actually learning."

https://www.zdnet.com

Speculations

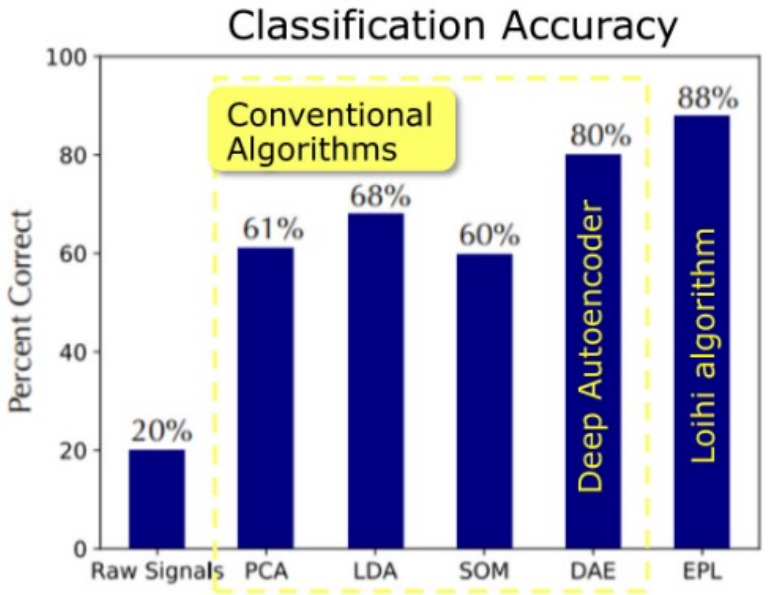
- ***Spiking Neural Nets, and neuromorphic architectures?***
  - *I'm skeptical…..*
  - *No spike-based NN comes close to state of the art on practical tasks*
  - *Why build chips for algorithms that don't work?*

- ***Exotic technologies?***
  - *Resistor/Memristor matrices, and other analog implementations?*
    - *Conversion to and from digital kills us.*
    - *No possibility of hardware multiplexing*
  - *Spintronics?*
  - *Optical implementations?*

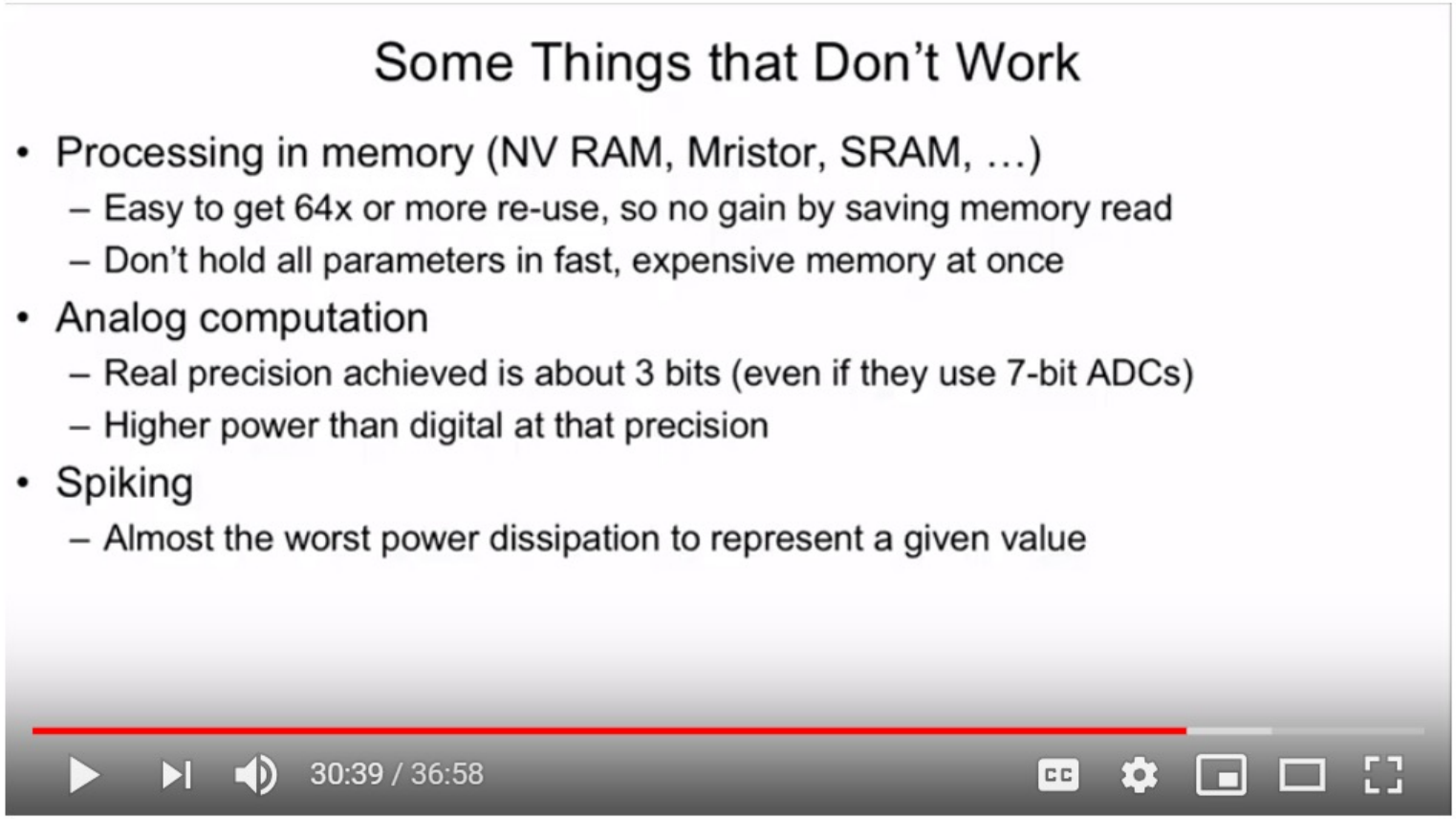
Lecun ISSCC 2019

Outperforms Conventional Algorithms

- Provides average of **8% accuracy improvement** vs deep autoencoder
- **40x more data efficient** learning vs backpropagation
- Supports **online learning** (robust to catastrophic forgetting)



Mike Davies ISSCC 2019



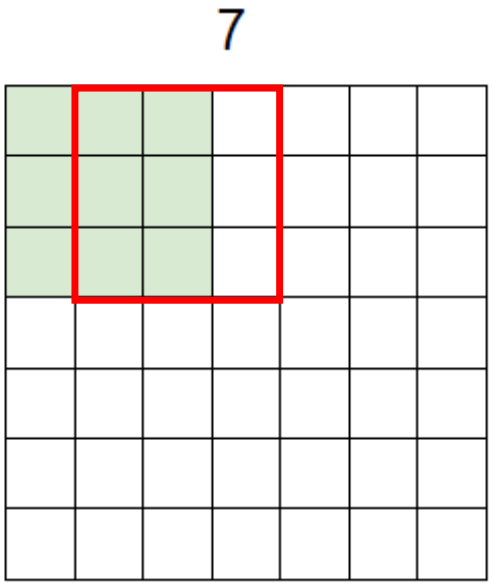
Hardware for Deep Learning, SysML 2018 Invited Talk | Bill Dally, NVIDIA, Stanford

906 views

24 0 SHARE SAVE

	Spiking Neural Networks (SNNs)	Deep Neural Networks (DNNs)
Computation	Event-driven (spike-based)	Continuous (matrix-based)
Energy Efficiency	High (sparse activation, low power)	Low (high power consumption, always active)
Biological Plausibility	High (mimics brain function)	Low (uses artificial activation functions)
Information Encoding	Uses spike timing and frequency	Uses continuous activations (ReLU, Sigmoid, etc.)
Training	Difficult (non-differentiable spikes)	Easy (uses backpropagation)
Hardware Compatibility	Neuromorphic chips (Intel Loihi, IBM TrueNorth)	Standard GPUs, TPUs
Computational Speed	Slower on traditional hardware	Optimized for GPUs (fast training)
Memory Requirements	Lower (spikes are sparse)	Higher (large weight matrices)
Best Use Cases	Low-power, real-time edge AI, robotics, neuromorphic computing	General-purpose AI, image/audio processing, NLP, large-scale AI

Closer Look at Spatial Dimensions

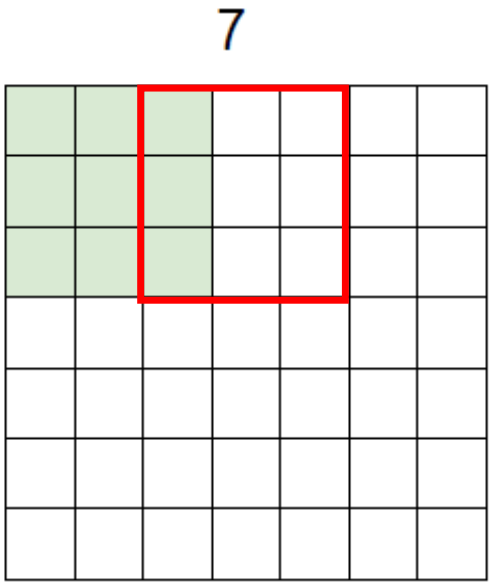


7x7 input (spatially)

assume 3x3 filter with **stride = 1**

=> 5x5 output

Closer Look at Spatial Dimensions

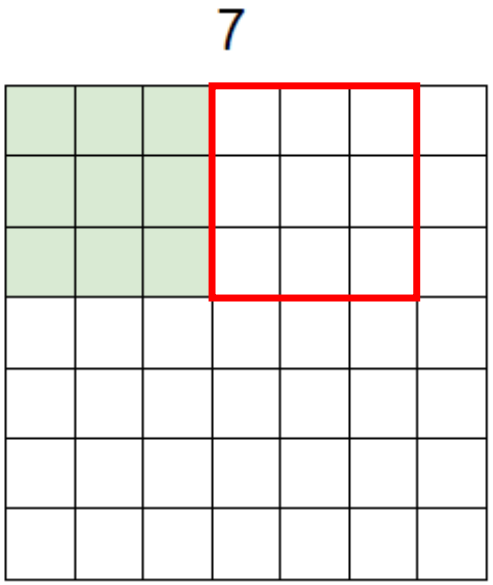


7x7 input (spatially)

assume 3x3 filter with **stride = 2**

=> 3x3 output

Closer Look at Spatial Dimensions

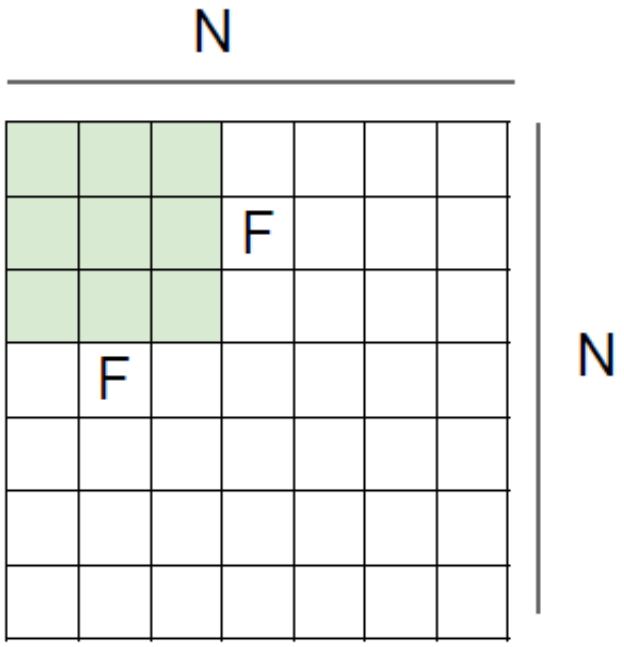


7x7 input (spatially)

assume 3x3 filter with **stride = 3**

=> Cannot fit

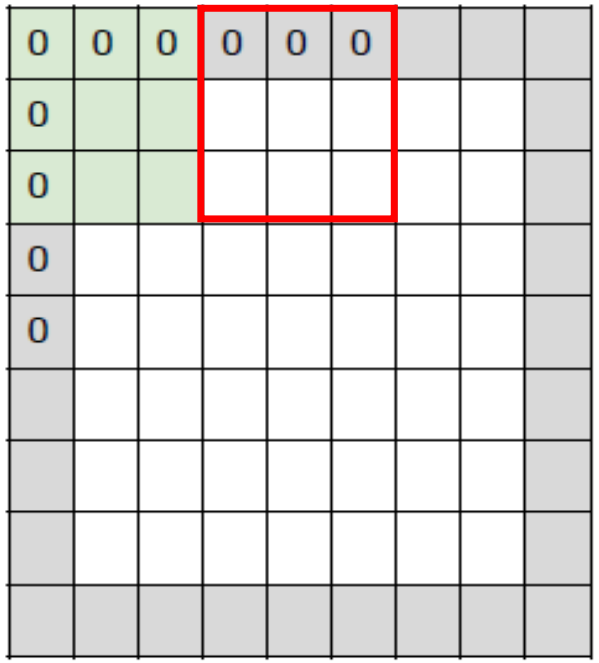
Closer Look at Spatial Dimensions



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33

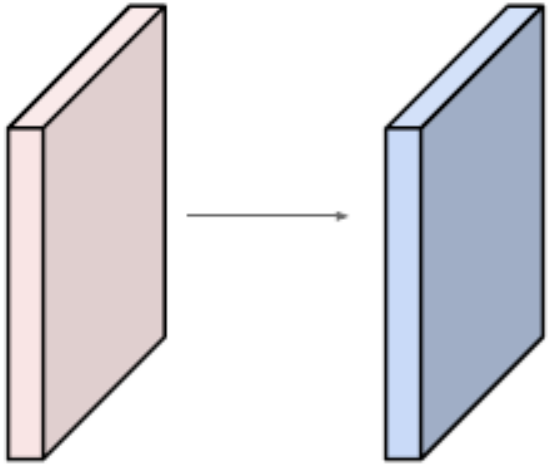
Solution: Zero Padding



Benefits:

- Fit for any size filters
- Preserve spatial dimension when needed
  -- Shrinking too fast is not good

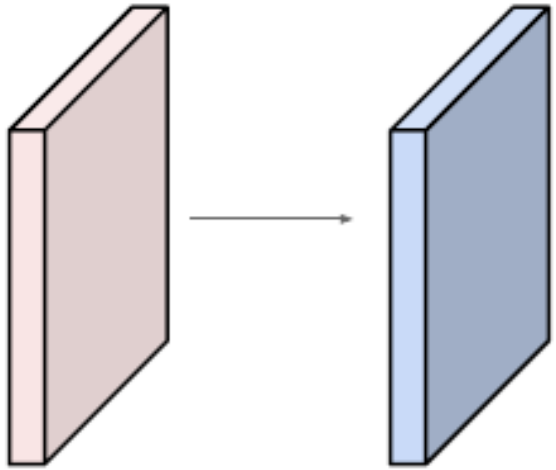
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

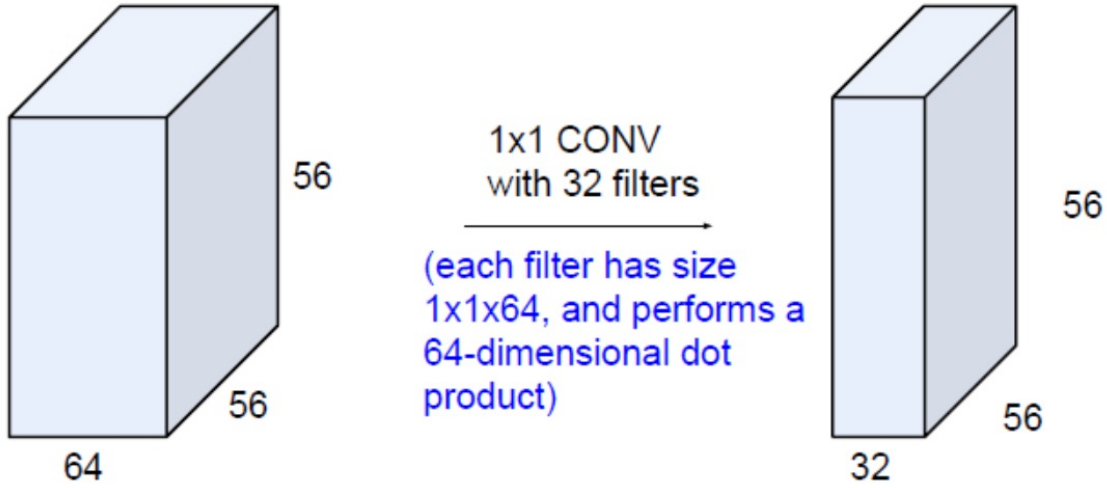
Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

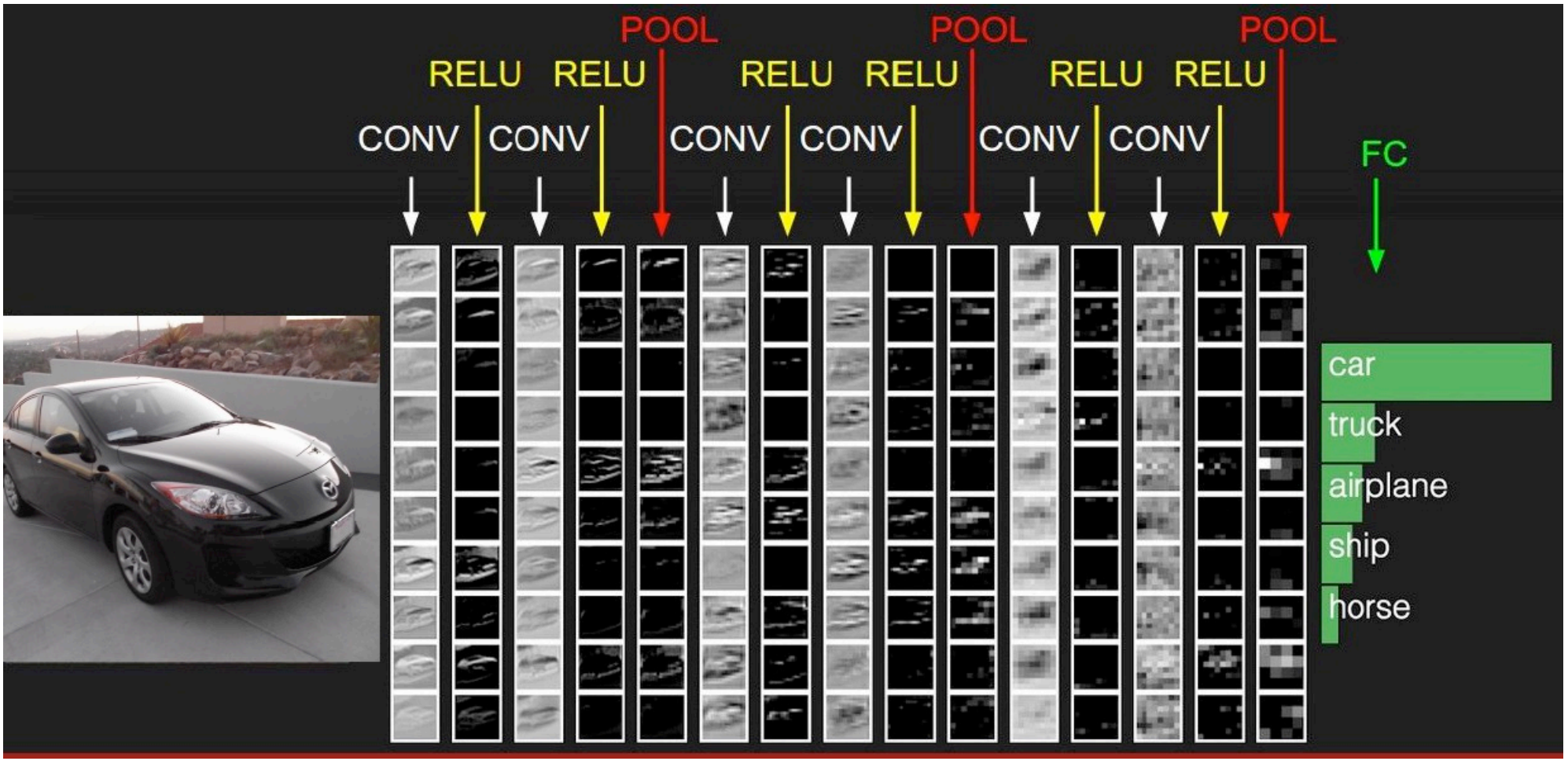
1X1 CONV



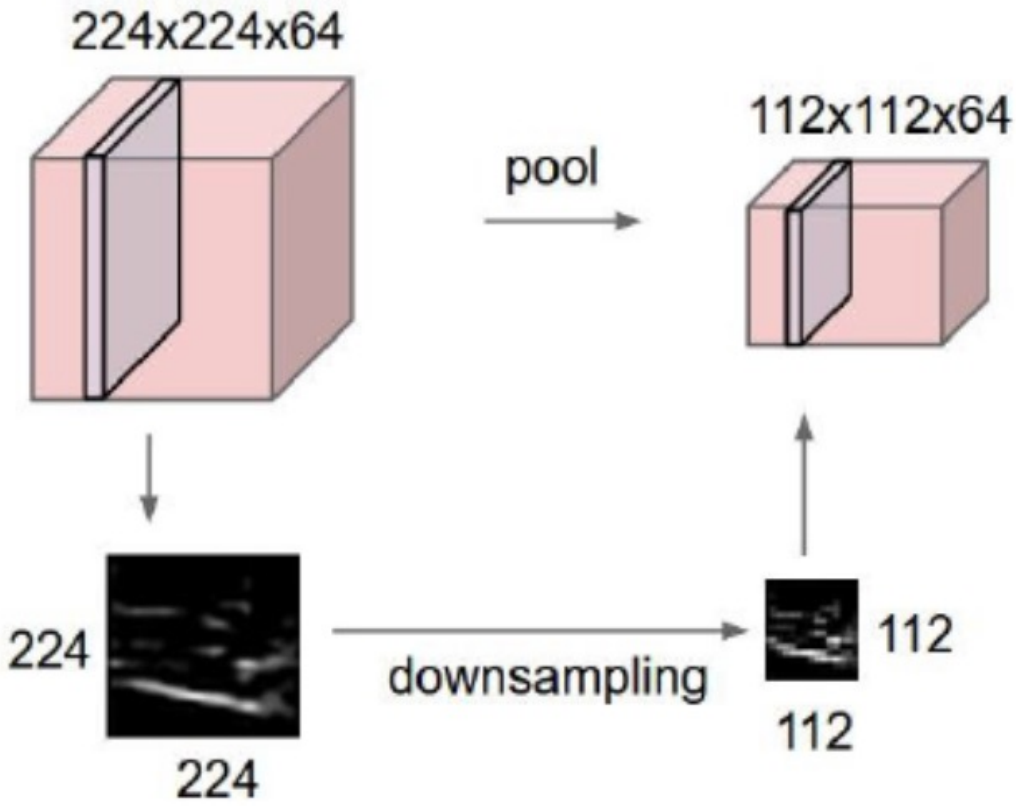
Can be used for implementing FC layer

```
# Use conv2d instead of fully_connected layers.
with slim.arg_scope([slim.conv2d],
                    weights_initializer=trunc_normal(0.005),
                    biases_initializer=tf.constant_initializer(0.1)):
  net = slim.conv2d(net, 4096, [5, 5], padding='VALID',
                    scope='fc6')
  net = slim.dropout(net, dropout_keep_prob, is_training=is_training,
                     scope='dropout6')
  net = slim.conv2d(net, 4096, [1, 1], scope='fc7')
```

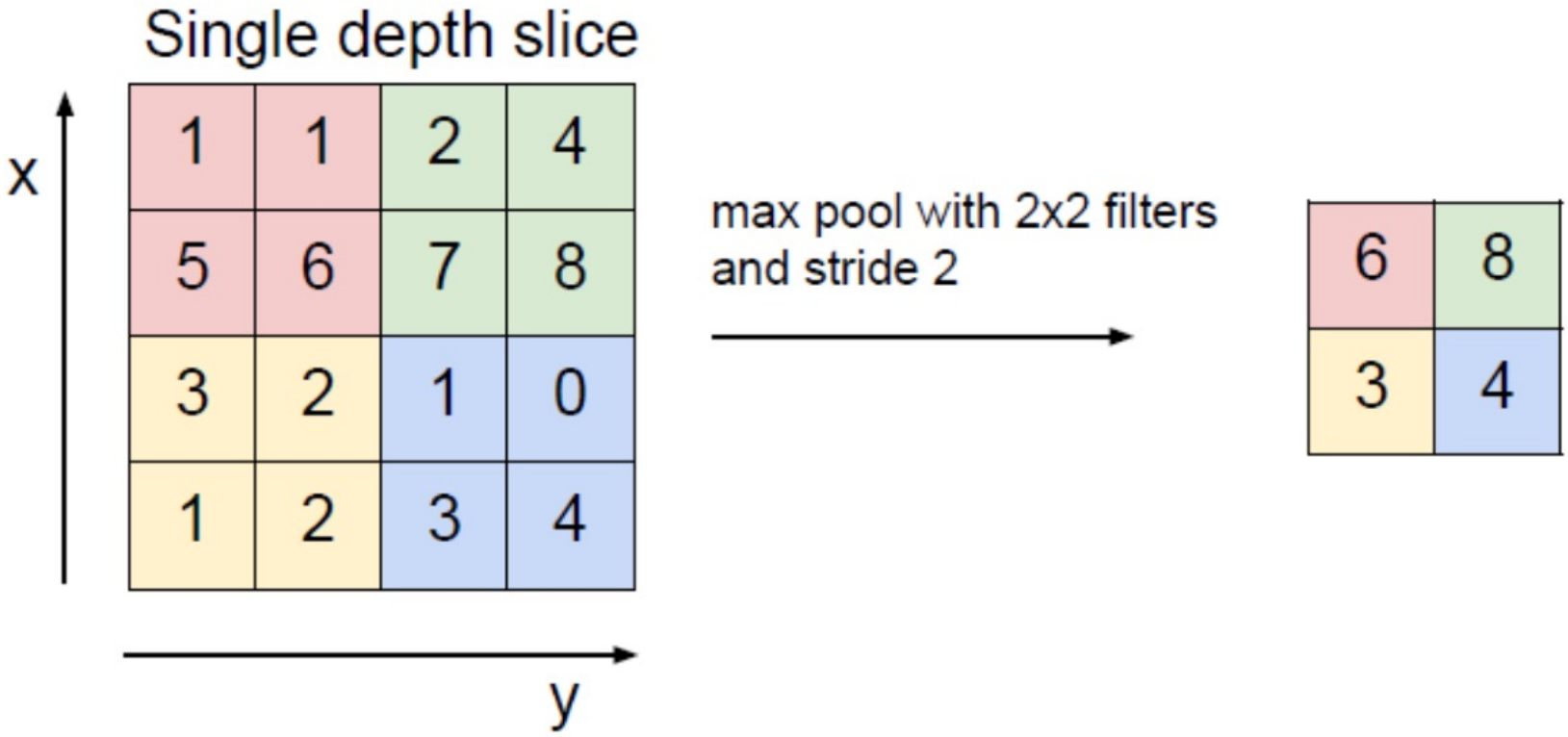
TensorFlow official model for Alexnet


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

Pooling Layer



Max Pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Summary

ConvNets stack CONV,POOL,FC layers

- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures (AlexNet, VGG) look like

  [(CONV-RELU)*N-POOL]*M-(FC-RELU)*K,SOFTMAX

  where N is usually up to ~5, M is large, 0 <= K <= 2.
- However, ResNet/GoogLeNet are different (See lecture10)

Acknowledgement

Many materials of the slides of this course are adopted and re-produced from several deep learning courses and tutorials.

- Prof. Fei-fei Li, Stanford, CS231n: Convolutional Neural Networks for Visual Recognition (online available)
- Prof. Andrew Ng, Stanford, CS230: Deep learning (online available)
- Prof. Yanzhi Wang, Northeastern, EECE7390: Advance in deep learning
- Prof. Jianting Zhang, CUNY, CSc G0815 High-Performance Machine Learning: Systems and Applications
- Prof. Vivienne Sze, MIT, “Tutorial on Hardware Architectures for Deep Neural Networks”
- Pytorch official tutorial https://pytorch.org/tutorials/
- https://github.com/jcjohnson/pytorch-examples