CS150: Database & Datamining

Lecture 25: Analytics & Machine Learning VII

Xuming He

Spring 2019

Outline

- What and why
  - Modeling perspective
  - Biological and computational motivation
- Deep networks
  - Building blocks
  - Structure design
  - Computation: prediction & learning
- Deep networks for images
  - CNN family
  - CNNs for vision tasks

Introduction

- Our goal is to build intelligent algorithms to make sense of data
  - Pattern recognition
  - Data analysis and prediction
- Typical problems
  - Speech recognition
    - Input: sound wave → Output: transcript
  - Language translation
    - Input: text in language A (Eng) → Output: text in language B (Chs)
  - Image classification
    - Input: images → Output: image category (cat, dog, car, house, etc.)
  - Autonomous driving
    - Input: sensory inputs → Output: actions (straight, left, right, stop, etc.)

Data-driven approach

- Building a mapping function (model)

$$y = f(x; \theta)$$

  - x: input data
  - y: expected output
  - θ: parameters to be estimated

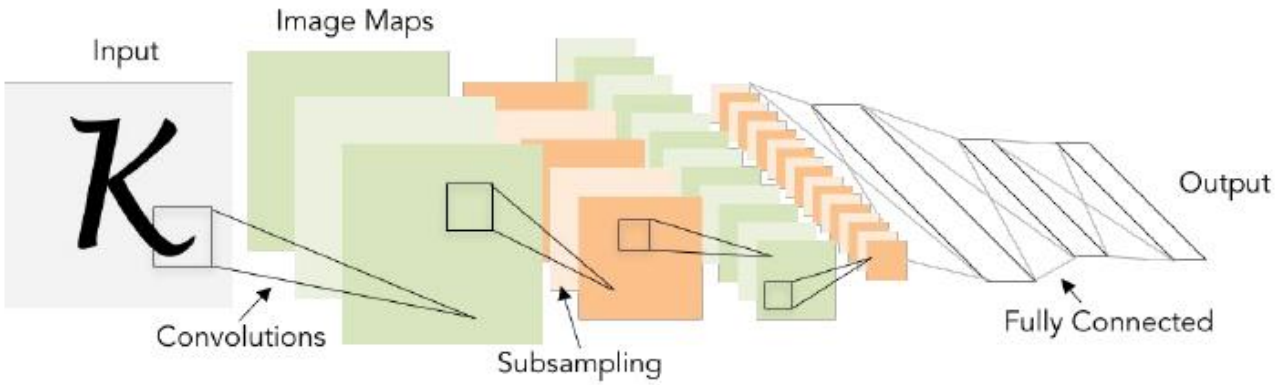
- Learning the model from data
  - Given a dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$
  - Find the 'best' parameter $\hat{\theta}$, such that

$$y_n \simeq f(x_n; \hat{\theta}) \quad \forall n$$

  - And it can be generalized to unseen input data

What is deep learning?

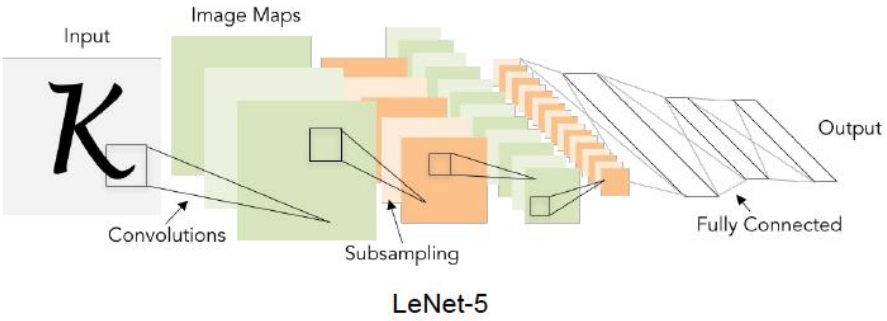
- Deep learning = Deep neural networks
  + Learning from data
- Deep neural networks
  - A family of parametric models $f(x;\theta)$
    - Consisting of many ‘simple’ computational units
    - Constructing a multi-layer representation of input



LeNet-5

What is deep learning?

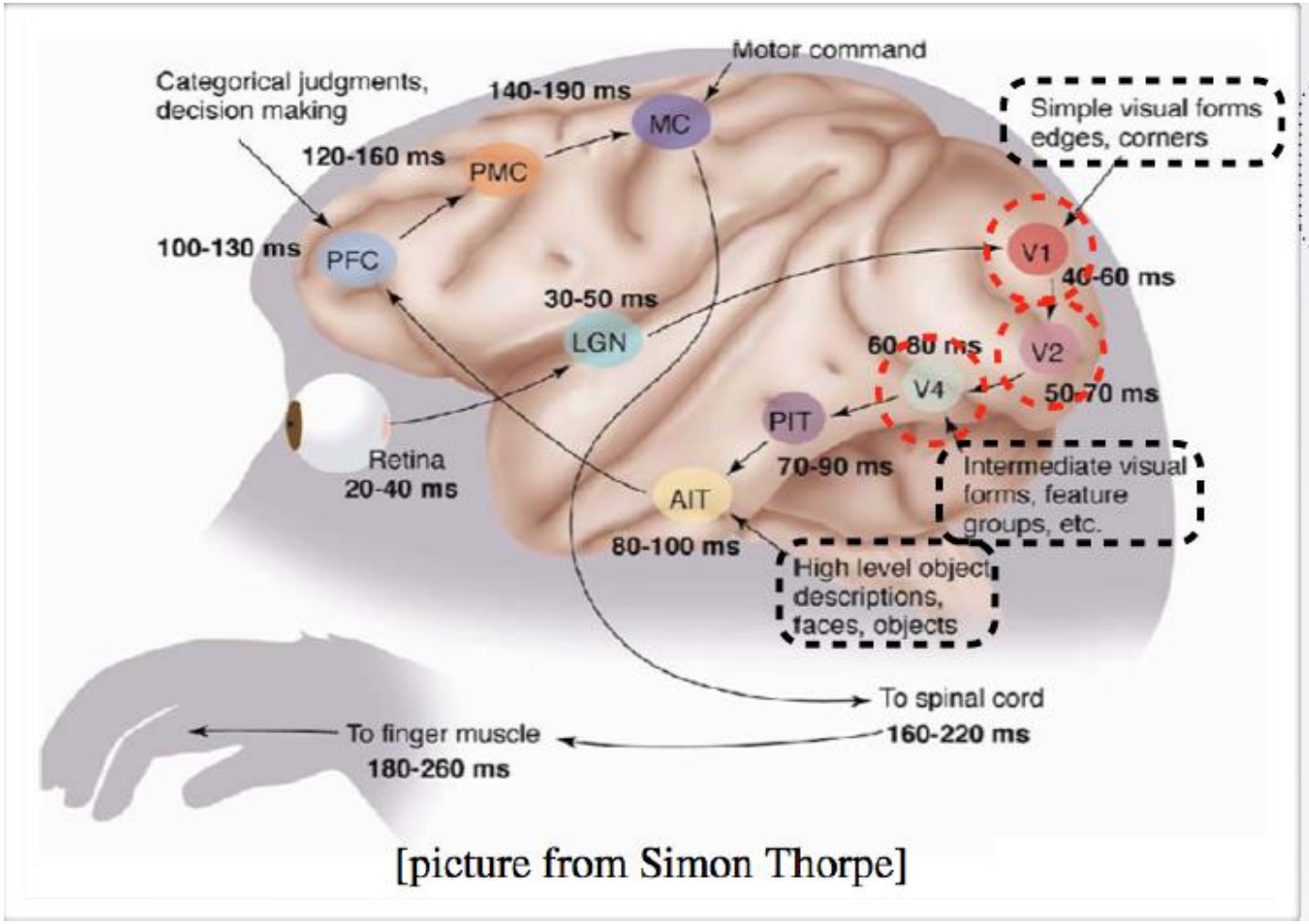
- Deep learning = Deep neural networks
  + Learning from data
- Parameter estimation from data
  - Parameters: connection weights between units
    - Formulated as an optimization problem
    - Efficient algorithms for handling large-scale models & datasets



LeNet-5

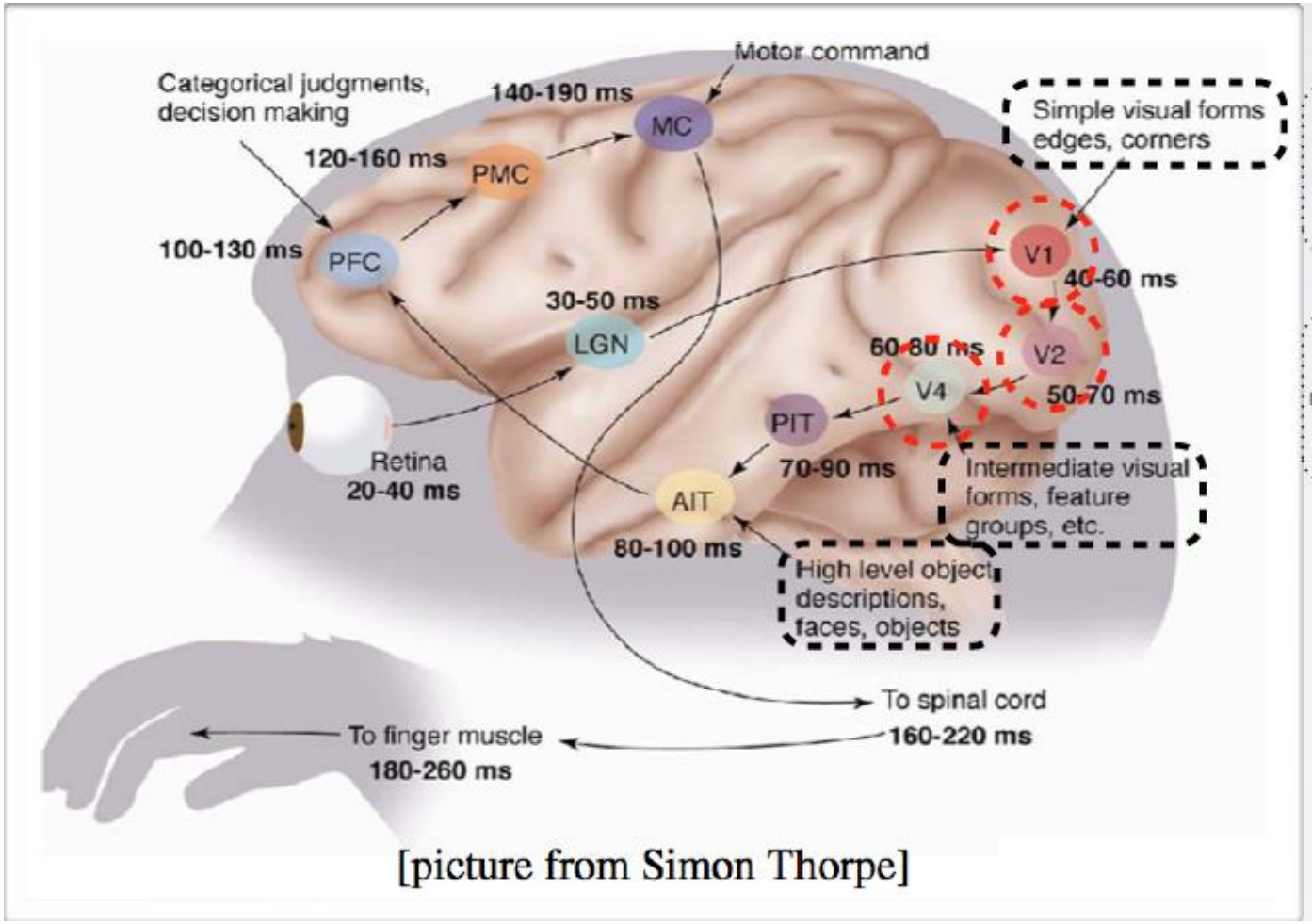
Why deep networks?

- Inspiration from visual cortex



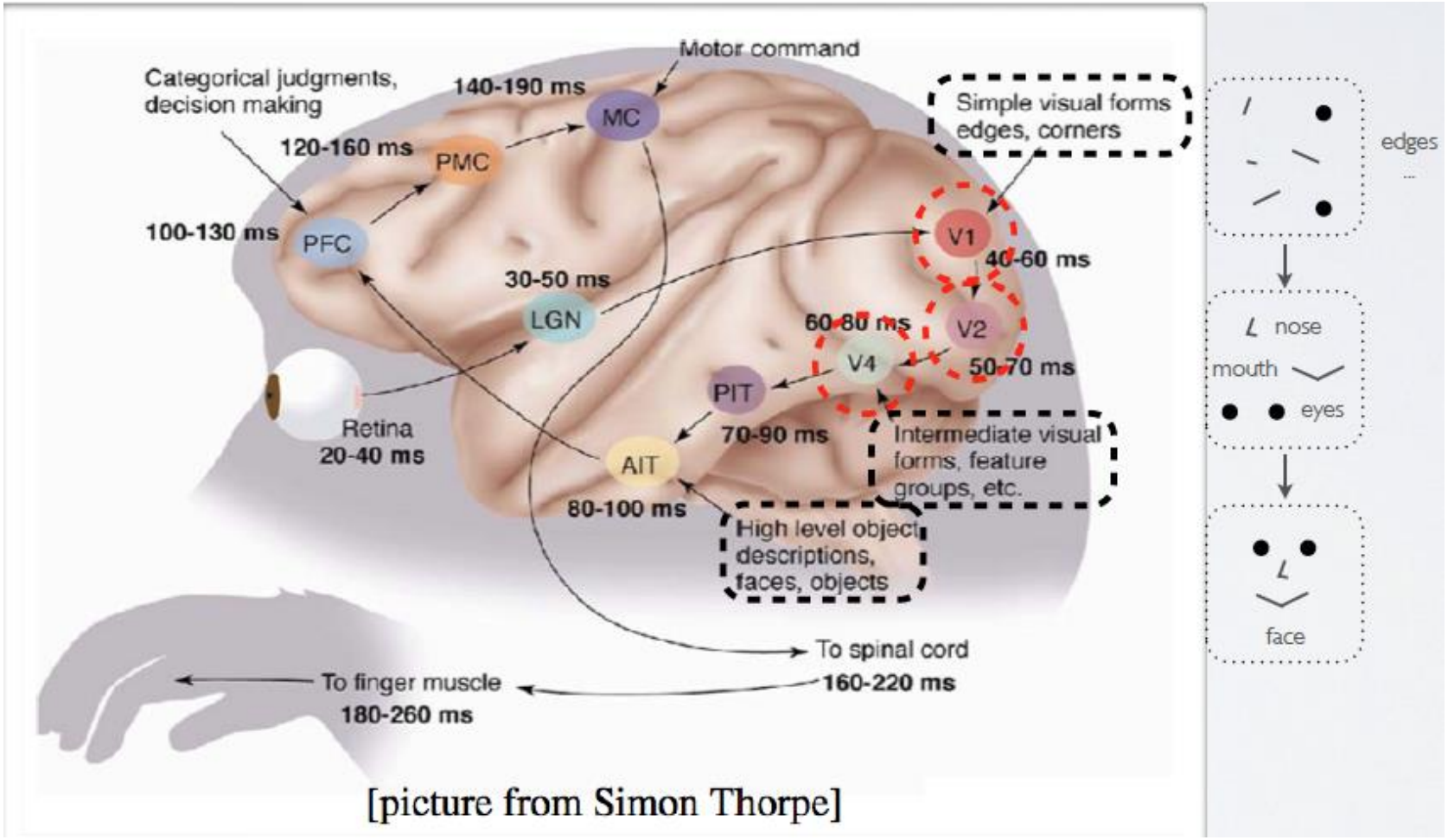
Why deep networks?

- Inspiration from visual cortex



Why deep networks?

- Inspiration from visual cortex

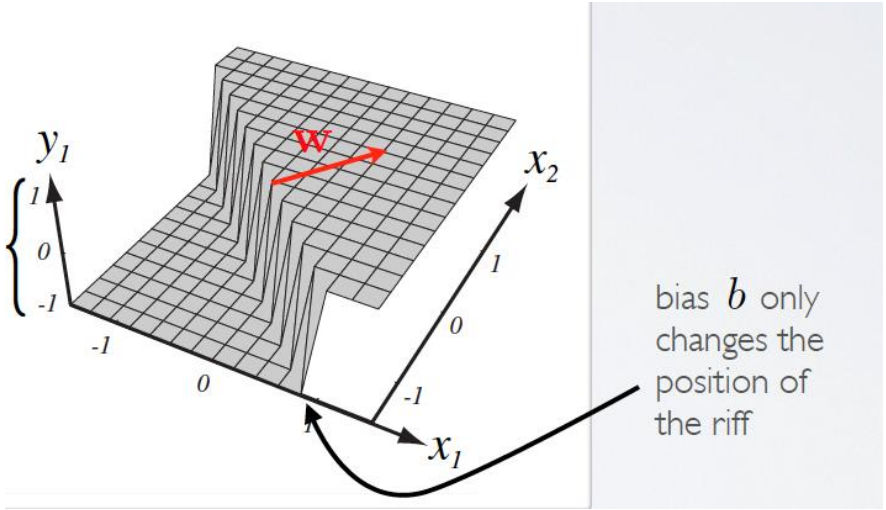


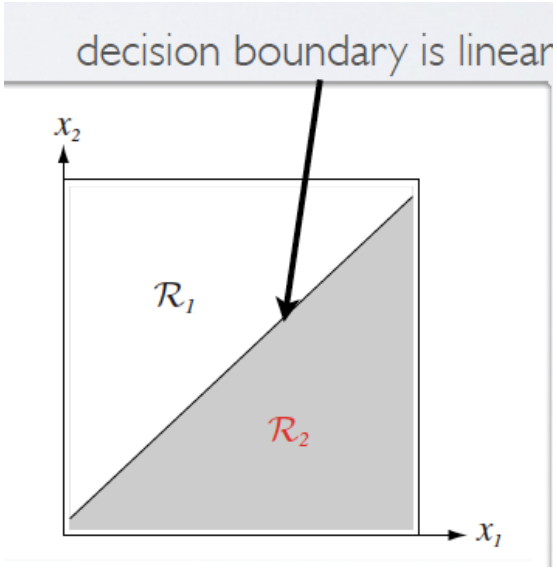
Why deep networks?

- Computational motivation
  - Example: Boolean functions
  - Perceptrons

$$y = f(\sum_i w_i x_i - b)$$

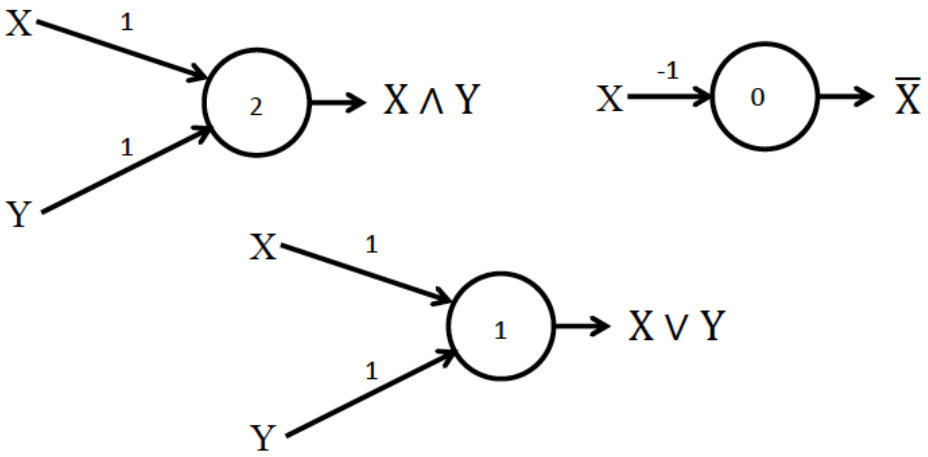
$$f(\cdot) = sign(\cdot)$$

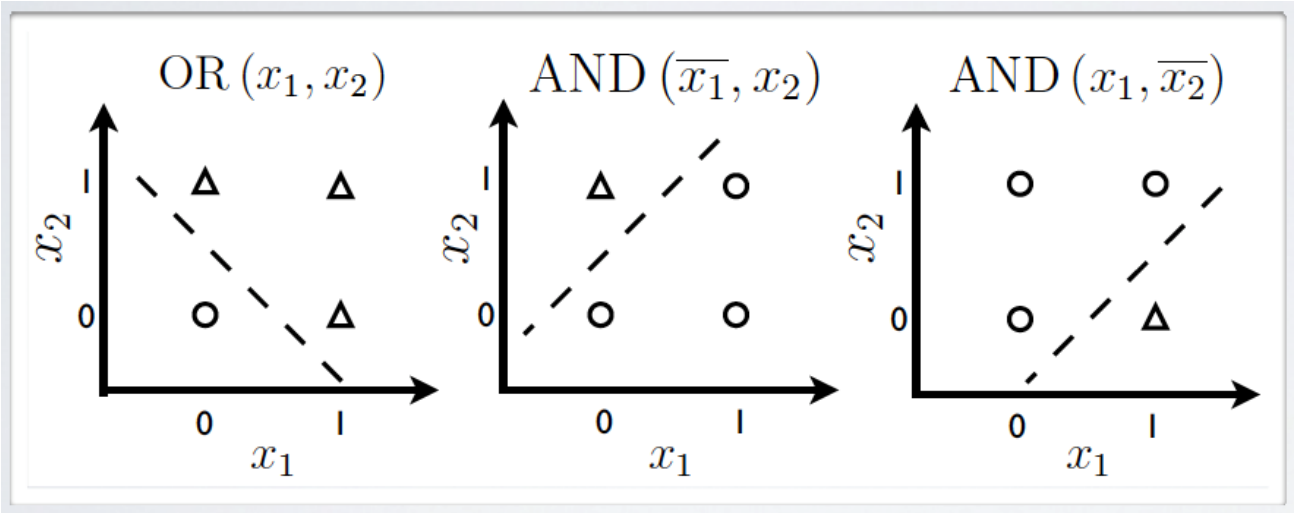




Perceptrons as a Boolean gate

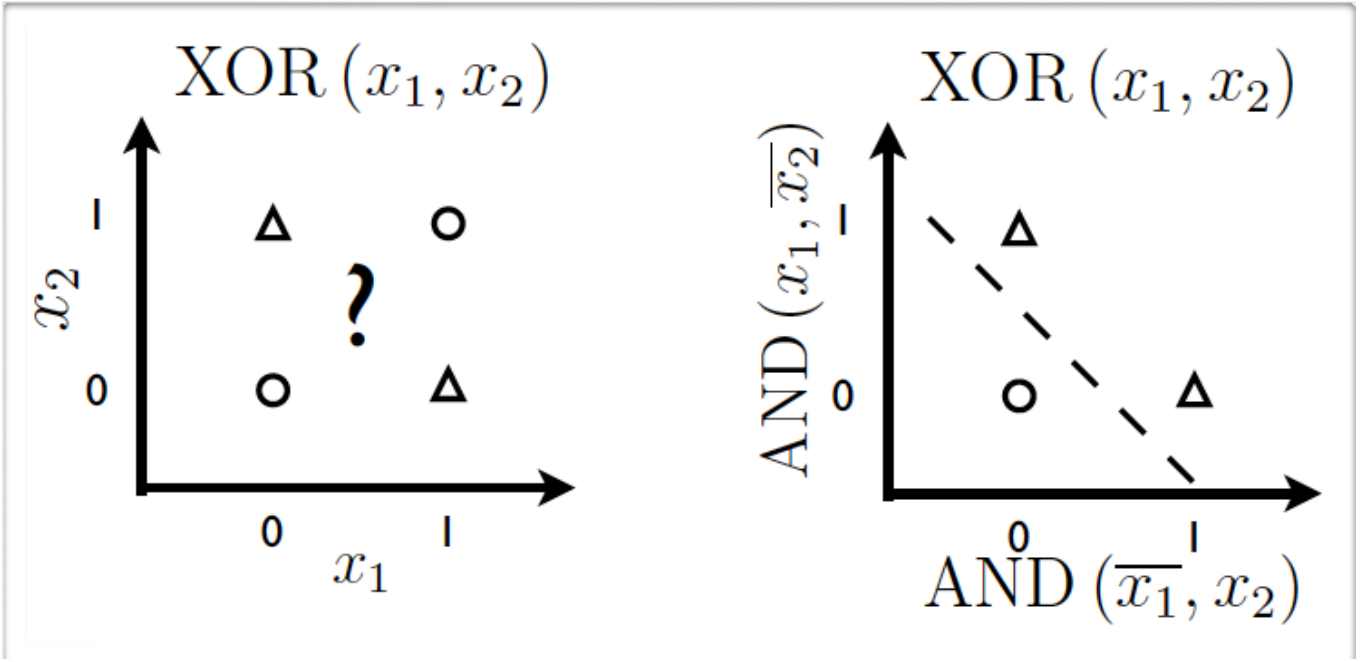
- A perceptron can model simple Boolean gates





Capacity of single perceptron

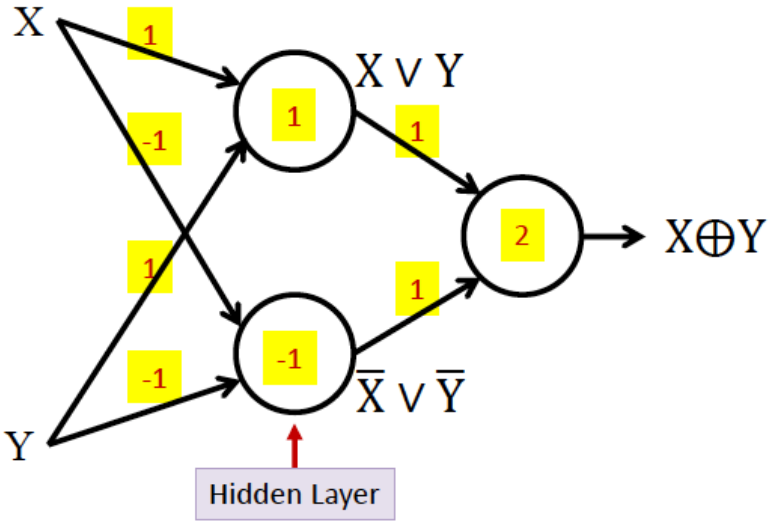
- Can’t solve non linearly separable problems



- Unless the input is transformed in a better representation

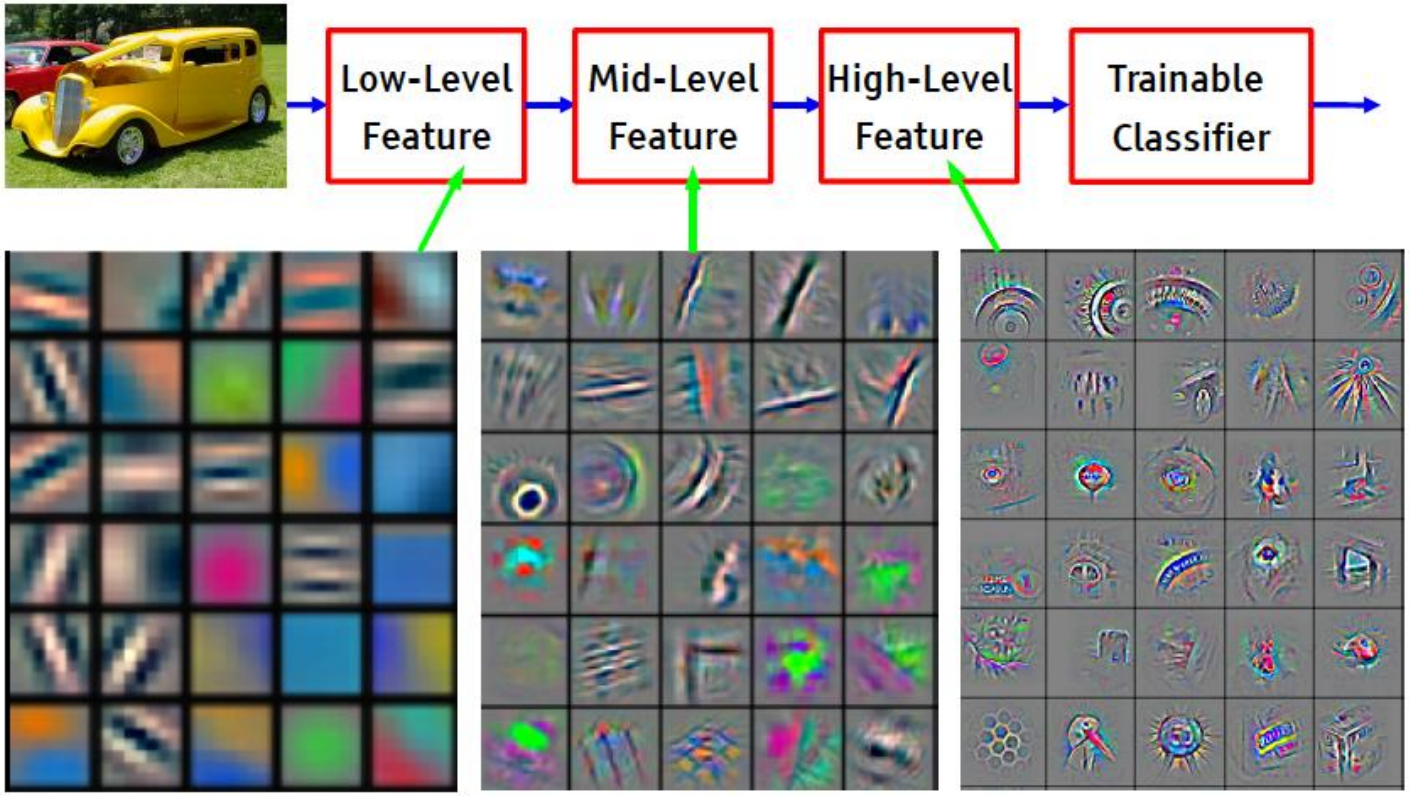
Multi-layer representation

- Can’t solve non linearly separable problems
- Unless the input is transformed in a better representation



Why deep networks?

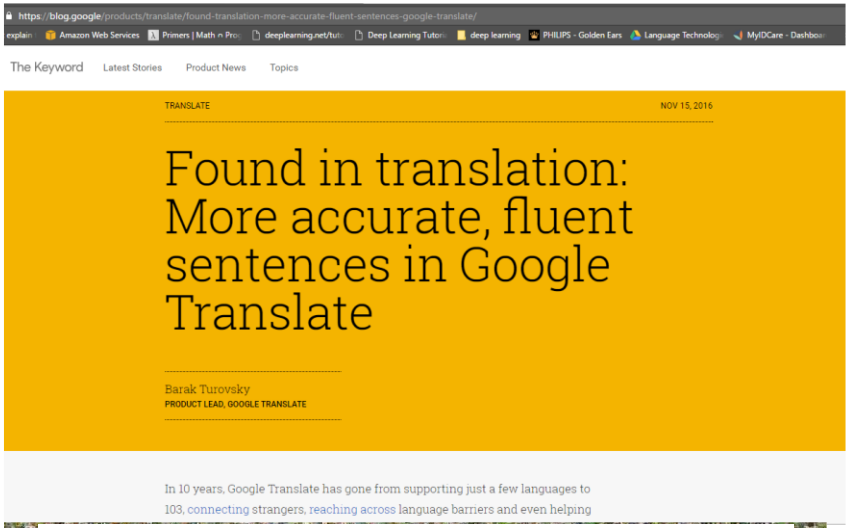
- A deep architecture can represent certain functions (exponentially) more compactly
- Learning a rich representation of input data

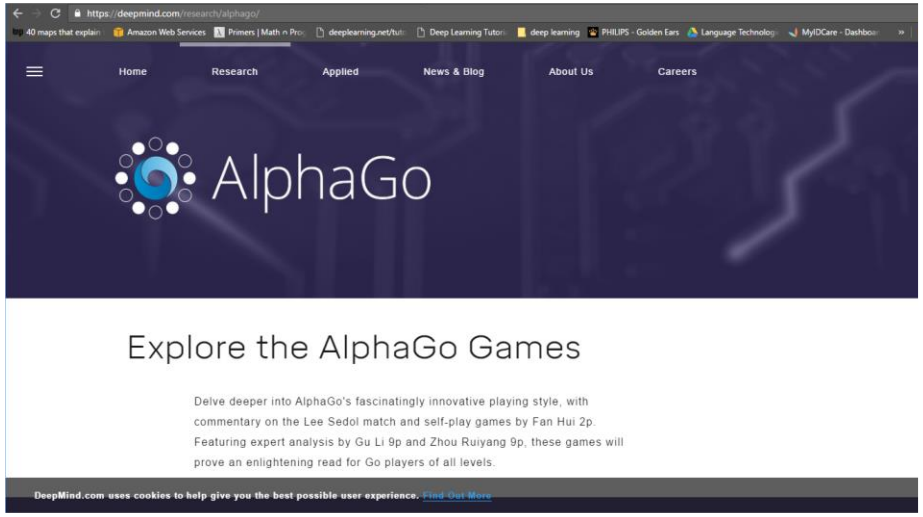


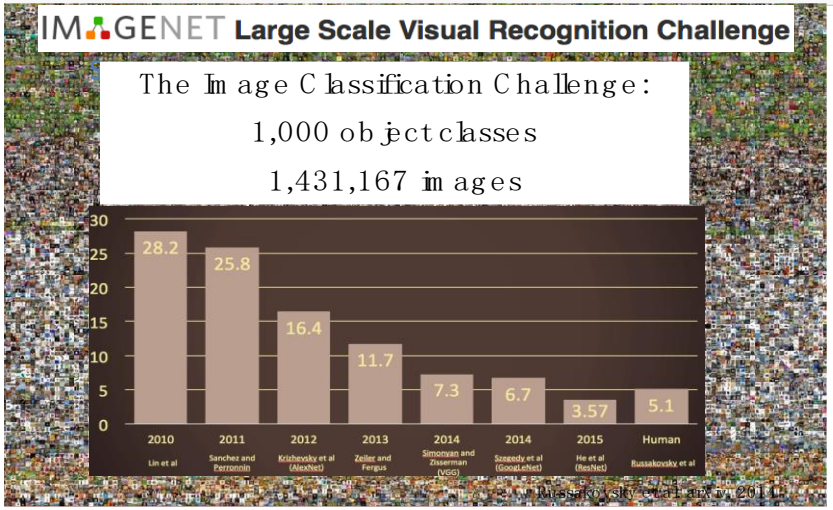
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Recent success with DL

- Some recent success with neural networks







Motivation: summary

- Inspired by biological vision
- Required by computational efficiency
- Great successes in real-world applications

Outline

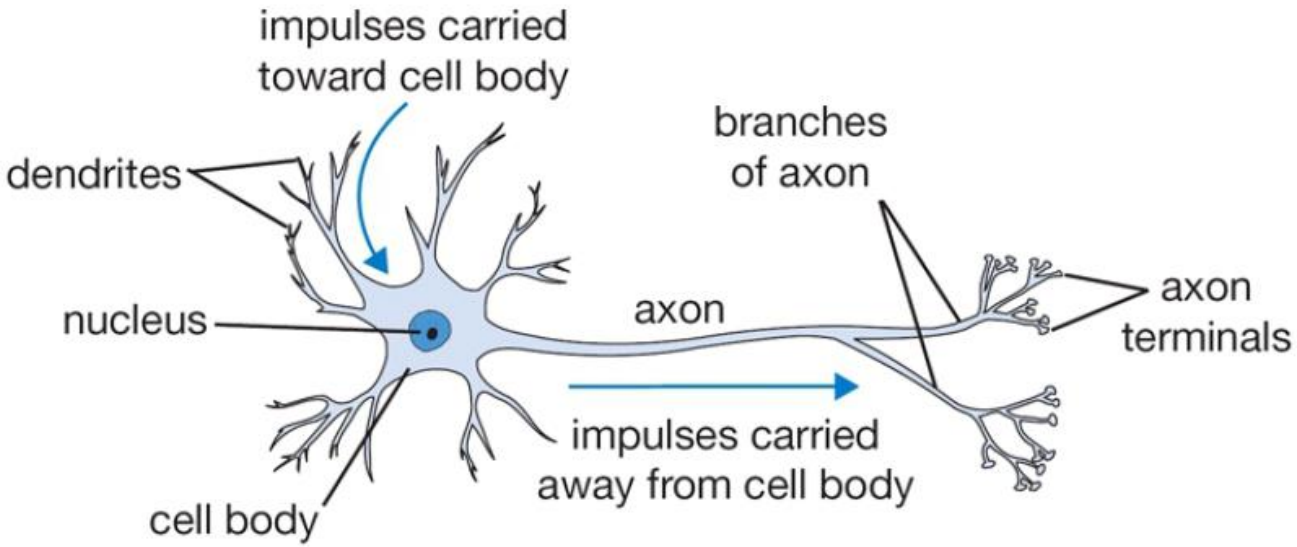
- What and why
  - Modeling perspective
  - Biological and computational motivation
- Deep networks
  - Building blocks
  - Structure design
  - Computation: prediction & learning
- Deep networks for images
  - CNN family
  - CNNs for vision tasks

Overview of deep networks

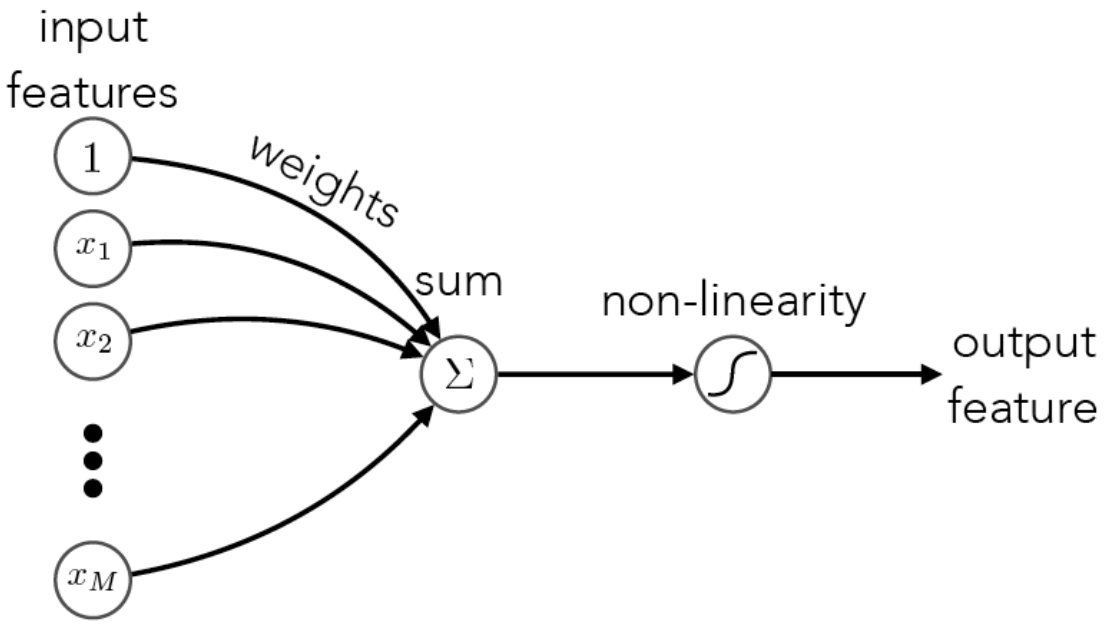
- Building blocks
  - I. Artificial neuron
  - II. Fully connected layer
- Network design
  - Multi-layer network
  - RNN
- Computation flow
  - Inference/Prediction
  - Learning

Simple neuron model

- Biological inspiration
  - Our brain has $\sim 10^{11}$ neurons, each of which communicates (is connected) to $\sim 10^4$ other neurons



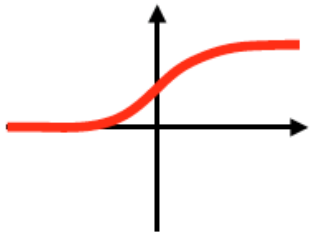
Building block I: artificial neuron



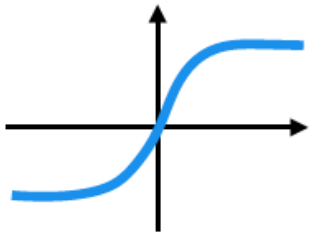
Activation functions

"old school"

logistic sigmoid



hyperbolic tangent (tanh)



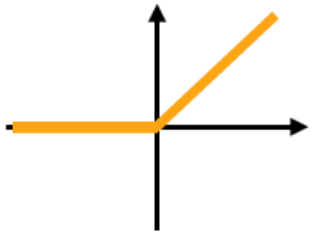
saturating

derivative goes to

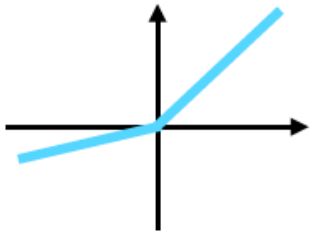
zero at +∞ and -∞

"new school"

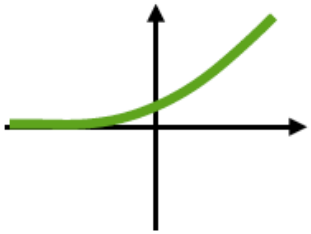
rectified linear unit (ReLU)



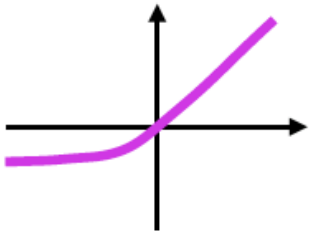
leaky ReLU



softplus



exponential linear unit (ELU)



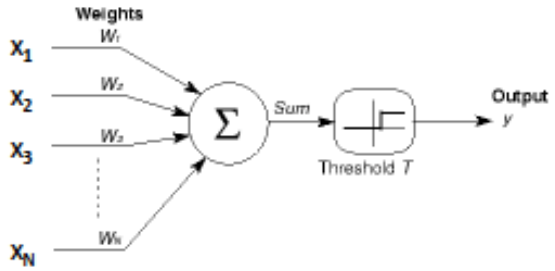
non-saturating

non-zero derivative

at +∞ and/or -∞

Interpretation

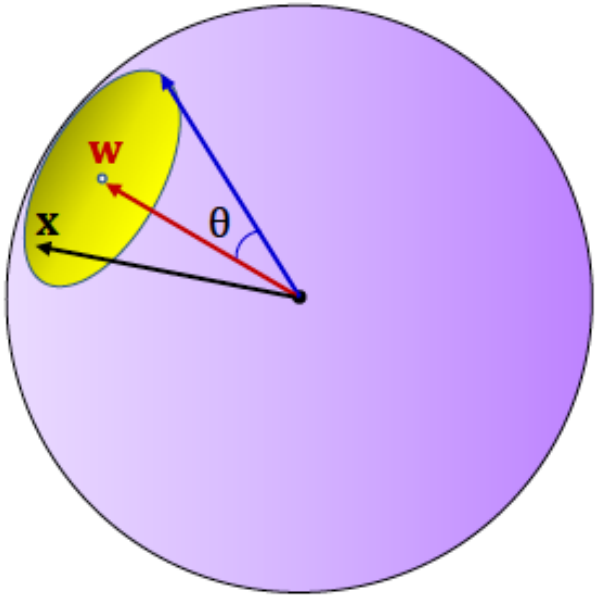
- An artificial neuron example



$$\mathbf{x}^T\mathbf{w} > T$$

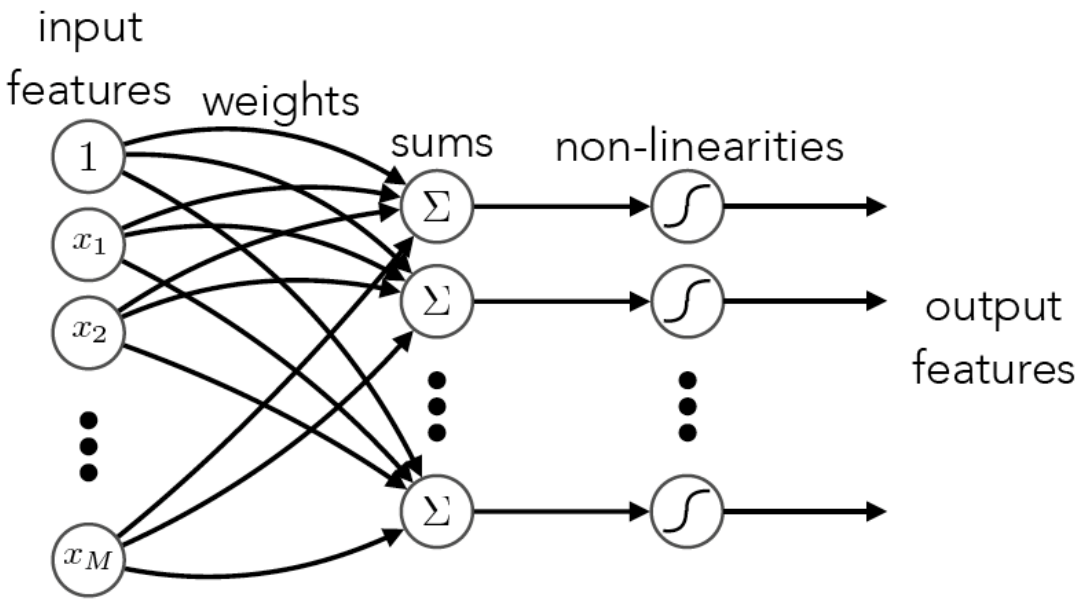
$$\Rightarrow \cos\theta > \frac{T}{|\mathbf{x}||\mathbf{w}|}$$

$$\Rightarrow \theta < \cos^{-1}\left(\frac{T}{|\mathbf{x}||\mathbf{w}|}\right)$$

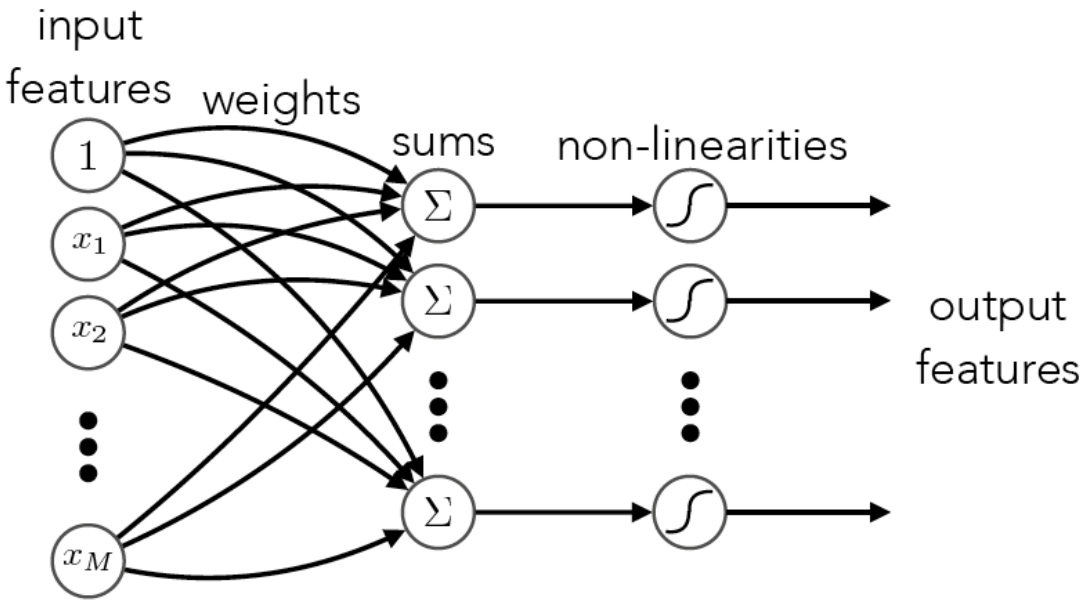


- A neuron fires if its input is close to the weight vector
- neuron functions as a pattern matching unit

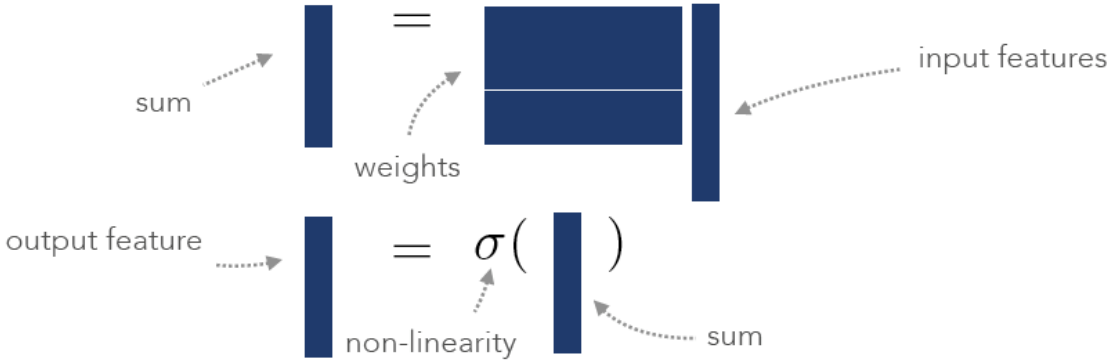
Building block II: fully-connected layer



Building block II: fully-connected layer



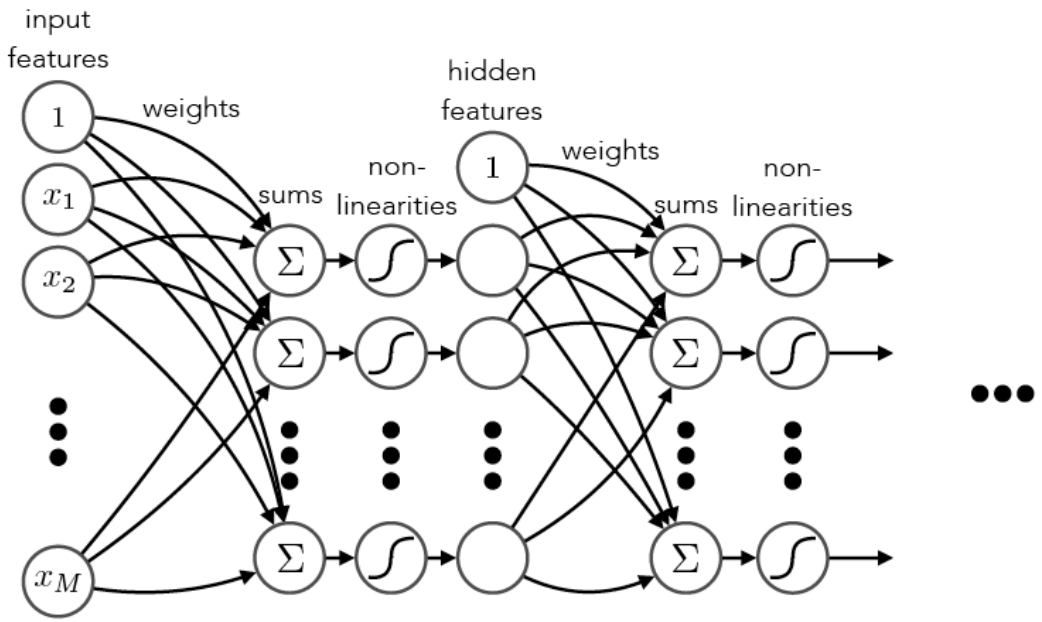
layer: *parallelized weighted sum and non-linearity*



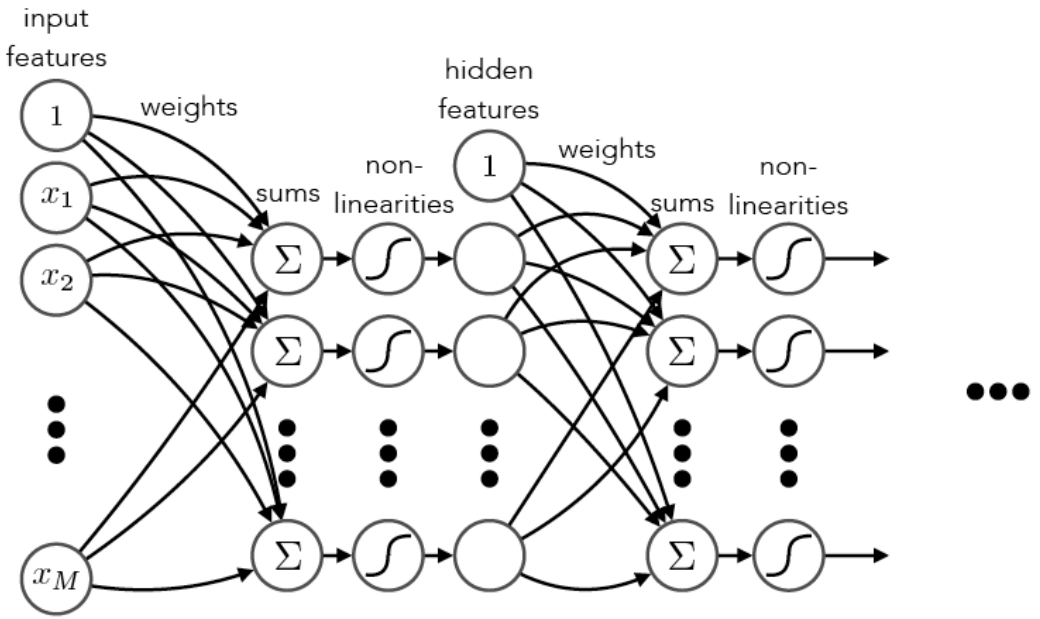
Overview of deep networks

- Building blocks
  - I. Artificial neuron
  - II. Fully connected layer
- Network design
  - Multi-layer network
  - RNN
- Computation flow
  - Inference/Prediction
  - Learning

Multilayer networks



Multilayer networks



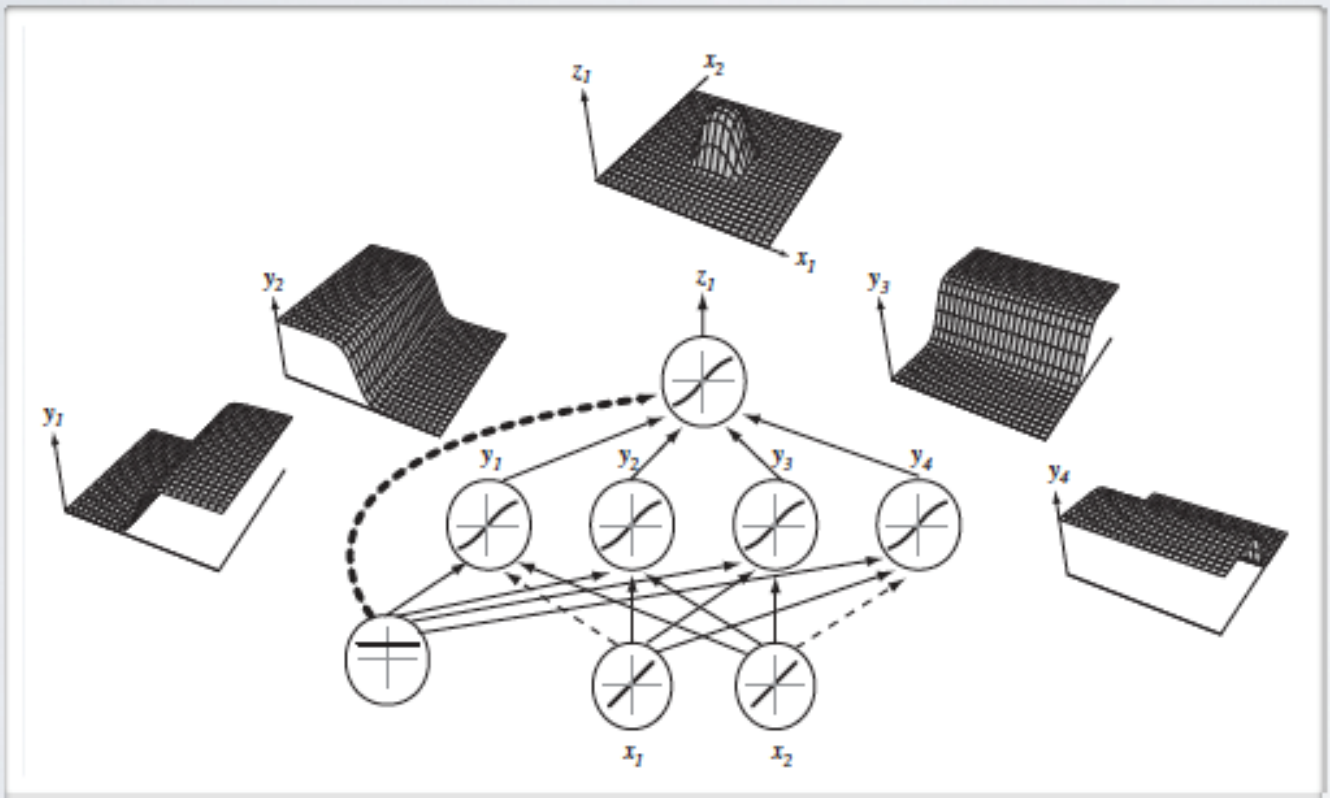
network: *sequence of parallelized weighted sums and non-linearities*

$$= \sigma(\ \cdots \ \sigma(\quad \sigma(\quad) \ ) \cdots)$$

output 2nd weights 1st weights input

Multilayer networks

- A method for building ***expressive*** non-linear functions
  - Multilayer networks are universal function approximators



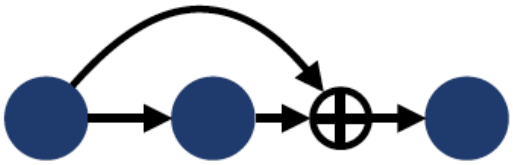
Other DNN variants

sequential connectivity: *information must flow through the entire sequence to reach the output*

information may not be able to propagate easily

→ *make shorter paths to output*

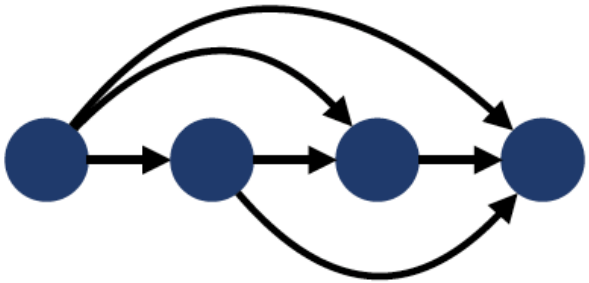
residual & highway connections



Deep residual learning for image recognition, He et al., 2016

Highway networks, Srivastava et al., 2015

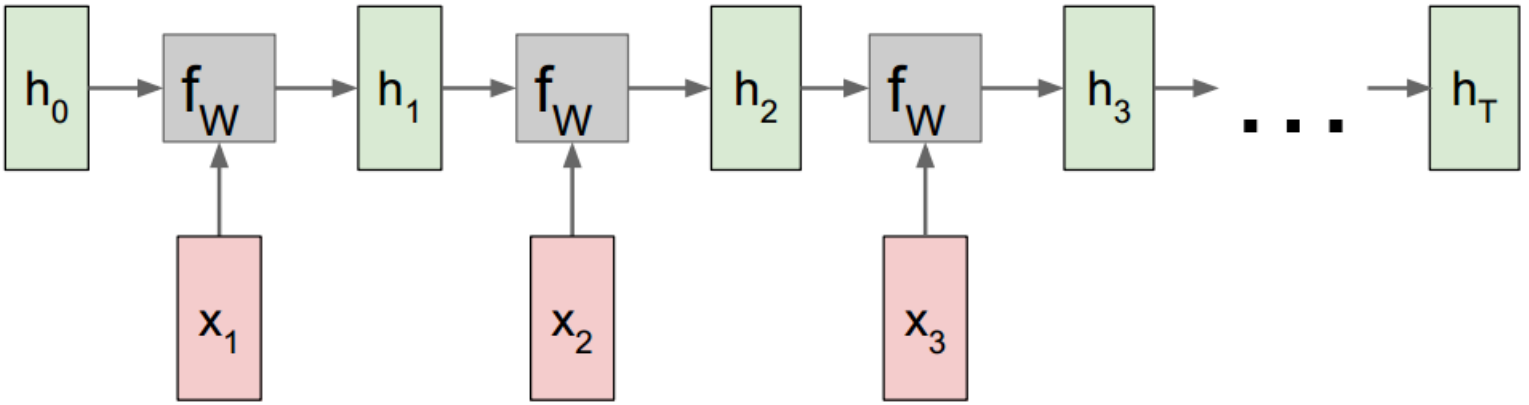
dense (concatenated) connections



Densely connected convolutional networks, Huang et al., 2017

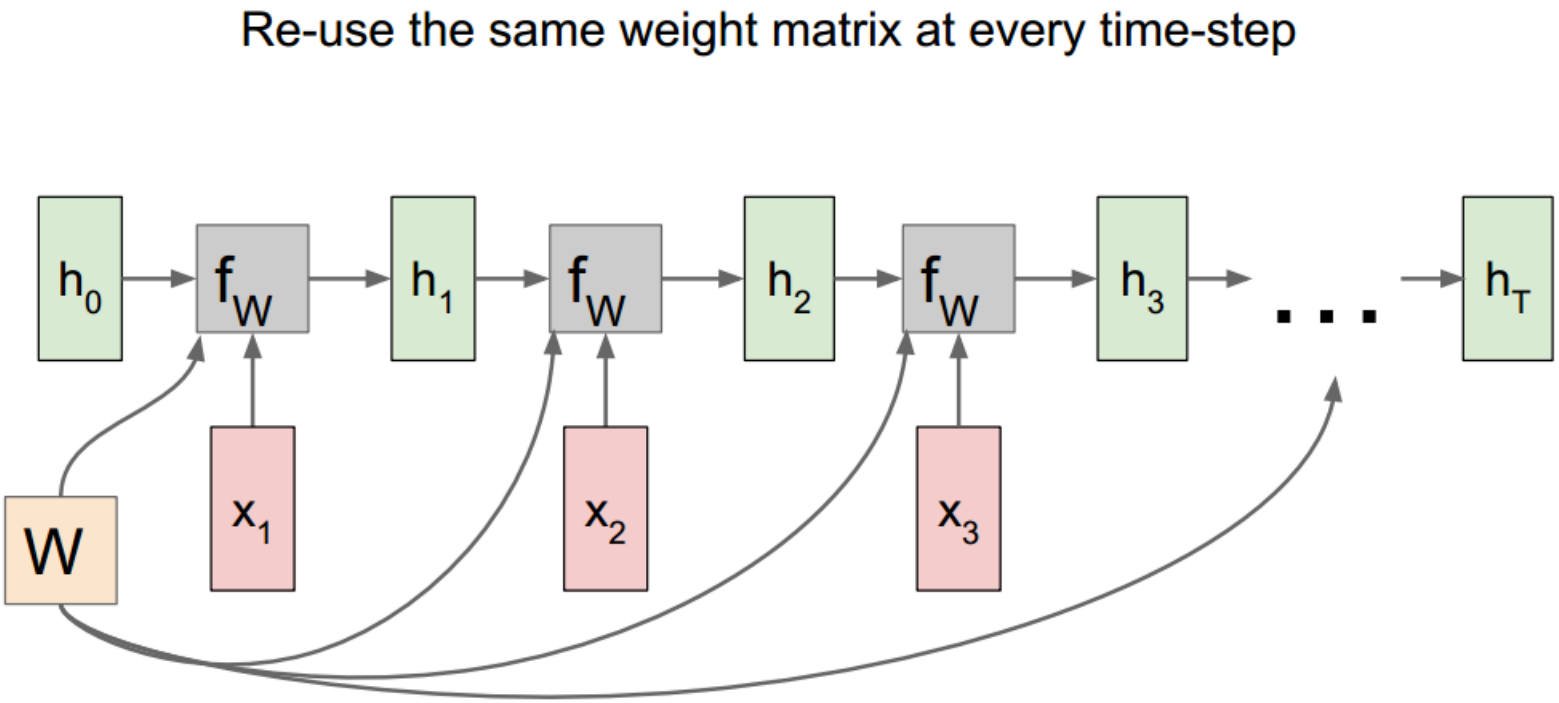
Recurrent neural networks

- Sequence predictions



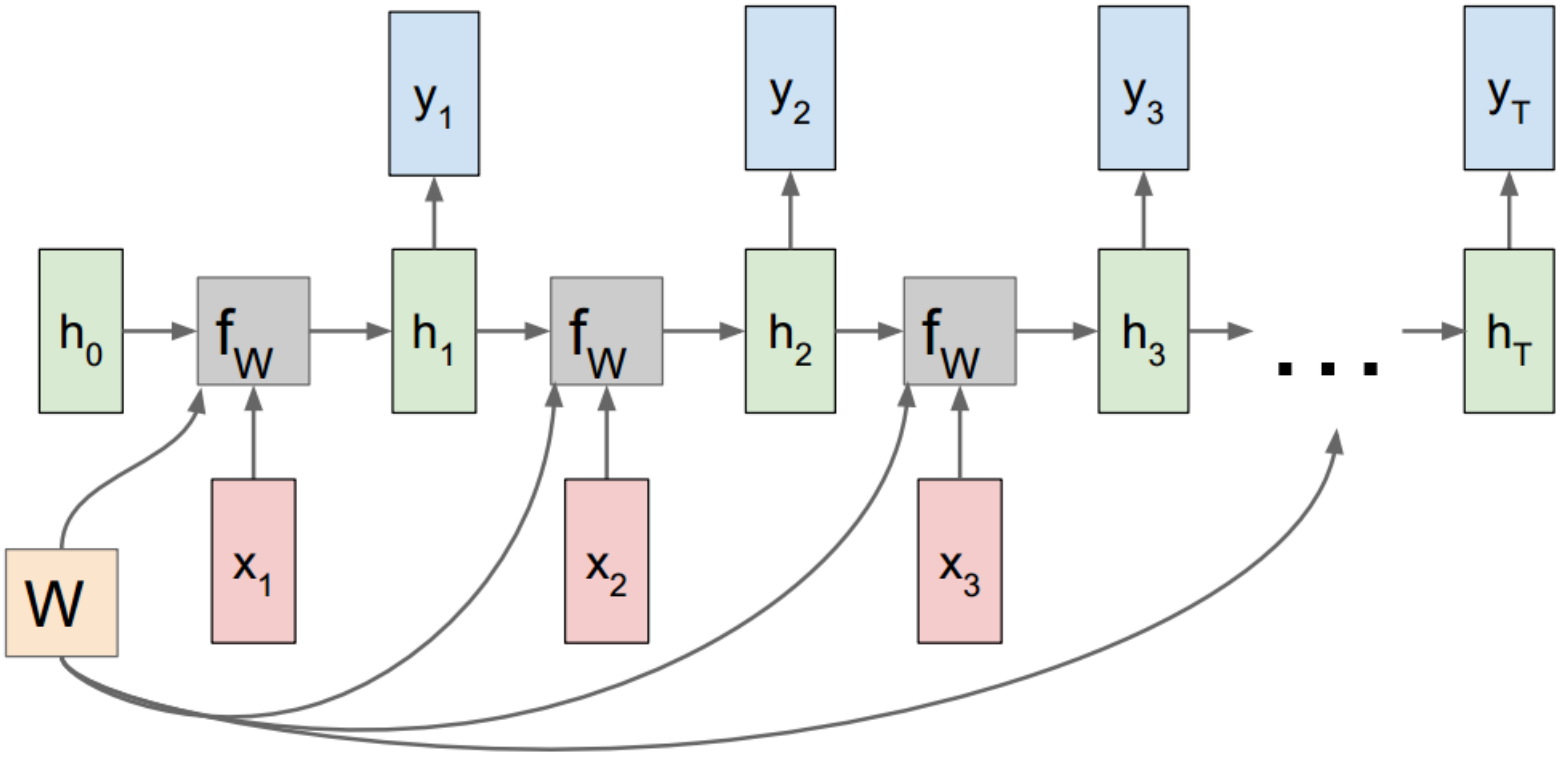
Recurrent neural networks

- Sequence predictions



Recurrent neural networks

- Sequence predictions



Network structure: summary

- Multilayer networks
- RNNs
- Many different network architectures
  - Depend on specific applications
  - Combine different network modules or types
  - Still lack of principled methodology

Overview of deep networks

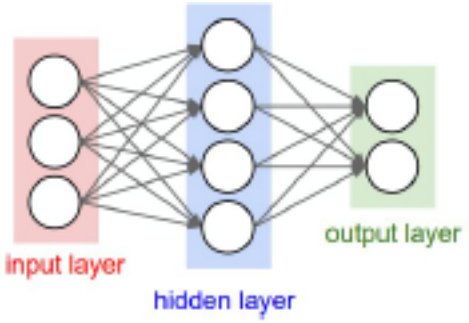
- Building blocks
  - I. Artificial neuron
  - II. Fully connected layer
- Network design
  - Multi-layer network
  - RNN
- Computation flow
  - Inference/Prediction
  - Learning

Computation in neural network

- We only need to know two algorithms
  - Forward pass: performs inference/prediction
  - Backward pass: performs learning

Forward pass

- What does the network compute?
  - Follow the directed network graph to compute the outputs



- Output of the network can be written as:

$$h_j(\mathbf{x}) = f\left(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}\right)$$

$$o_k(\mathbf{x}) = g\left(w_{k0} + \sum_{j=1}^{J} h_j(\mathbf{x}) w_{kj}\right)$$

(j indexing hidden units, k indexing the output units, D number of inputs)

Backward pass

- Training neural networks
  - Find weights:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \operatorname{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)})$$

    where $\mathbf{o} = f(\mathbf{x}; \mathbf{w})$ is the output of a neural network
  - Define a loss function, eg:
    - Squared loss: $\sum_k \frac{1}{2}(o_k^{(n)} - t_k^{(n)})^2$
    - Cross-entropy loss: $-\sum_k t_k^{(n)} \log o_k^{(n)}$
  - Gradient descent:

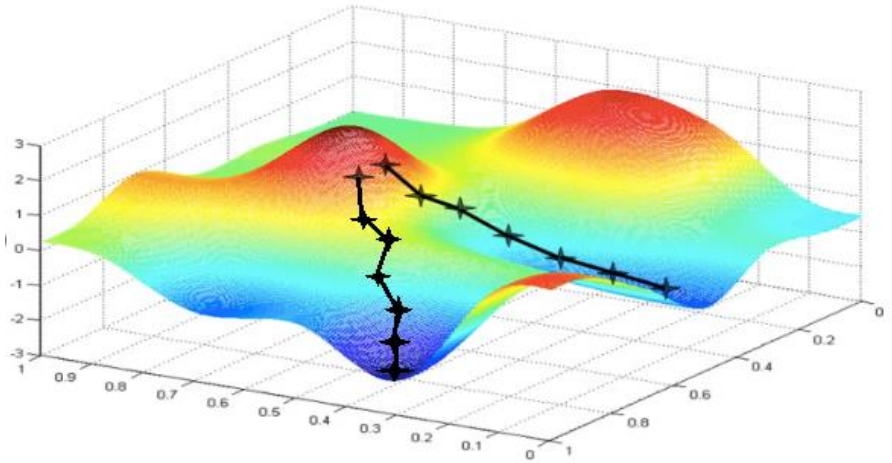
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \frac{\partial E}{\partial \mathbf{w}^t}$$

    where η is the learning rate (and E is error/loss)

Learning as optimization

- Update weights by gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$



- **Back-propagation:** gradients are computed in the direction from output to input layers and combined using chain rule
- **Stochastic gradient descent:** compute the weight update w.r.t. one training example (or a small batch of examples) at a time, cycle through training examples in random order in multiple epochs

Backward pass

- Backpropagation
  - An efficient method for computing gradients in NNs
  - A neural network as a function of composed operations

$$f_L(\mathbf{w}_L, f_{L-1}(\mathbf{w}_{L-1}, \dots f_1(\mathbf{w}_1, \mathbf{x}) \dots))$$

and the loss $\mathcal{L}$ is a function of the network output

→ use chain rule to calculate gradients

chain rule example

$$y = w_2 e^{w_1 x}$$

input x output y parameters w_1, w_2

evaluate parameter derivatives: $\frac{\partial y}{\partial w_1}, \frac{\partial y}{\partial w_2}$

define

$$v \equiv e^{w_1 x} \longrightarrow y = w_2 v$$

$$u \equiv w_1 x \longrightarrow v = e^u$$

then $\frac{\partial y}{\partial w_2} = v = e^{w_1 x}$

chain rule

$$\frac{\partial y}{\partial w_1} = \boxed{\frac{\partial y}{\partial v}\frac{\partial v}{\partial u}\frac{\partial u}{\partial w_1}} = w_2 \cdot e^{w_1 x} \cdot x$$

Gradient descent iteration

- Forward pass

1st layer

$$\mathbf{s}^{(1)} = \mathbf{W}^{(1)\intercal}\mathbf{x}^{(0)}$$
$$\mathbf{x}^{(1)} = \sigma(\mathbf{s}^{(1)})$$

2nd layer

$$\mathbf{s}^{(2)} = \mathbf{W}^{(2)\intercal}\mathbf{x}^{(1)}$$
$$\mathbf{x}^{(2)} = \sigma(\mathbf{s}^{(2)})$$

$\bullet\bullet\bullet$

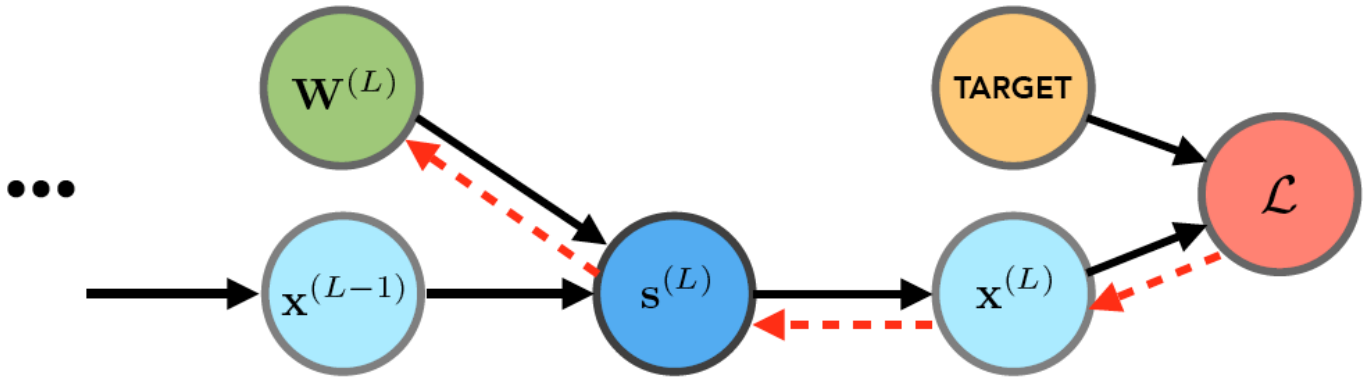
Loss

$$\mathcal{L}$$

- Backward pass

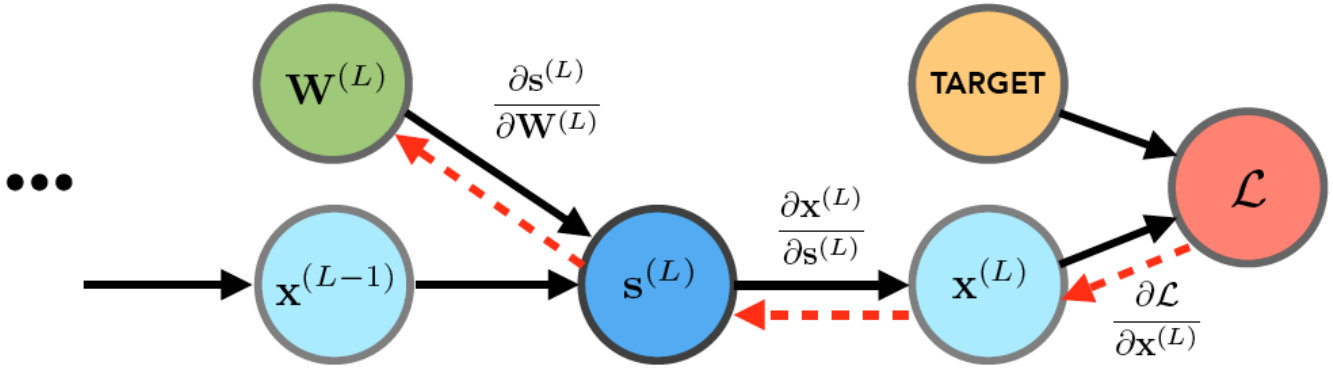
calculate $\nabla_{W^{(1)}}\mathcal{L}, \nabla_{W^{(2)}}\mathcal{L}, \ldots$ let's start with the final layer: $\nabla_{W^{(L)}}\mathcal{L}$

to determine the chain rule ordering, we'll draw the dependency graph



Gradient descent iteration

- Backward pass



$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}} \frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{W}^{(L)}}$$

depends on the form of the loss

derivative of the non-linearity

$$\frac{\partial}{\partial \mathbf{W}^{(L)}}(\mathbf{W}^{(L)\intercal}\mathbf{x}^{(L-1)}) = \mathbf{x}^{(L-1)\intercal}$$

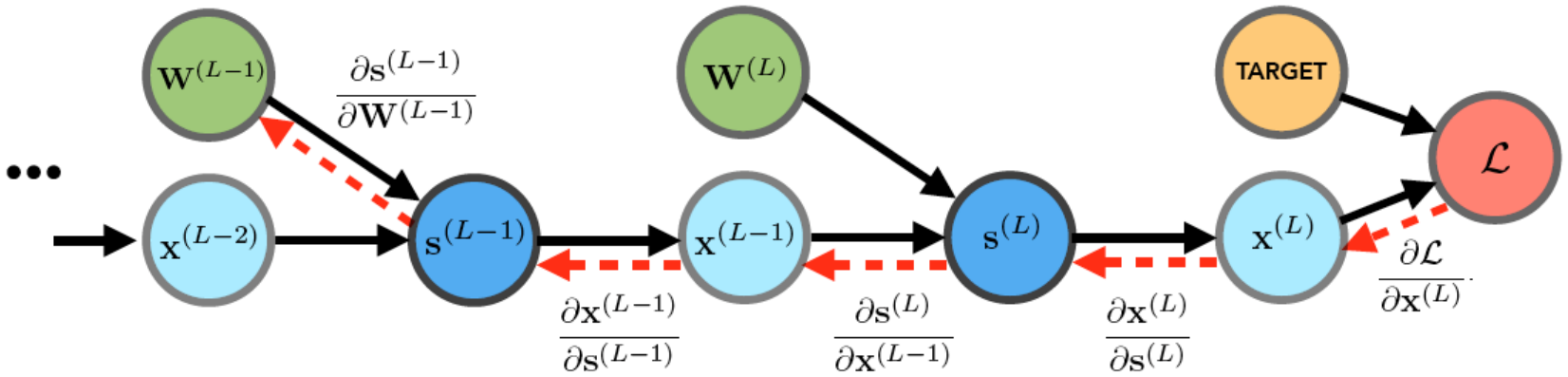
note $\nabla_{\mathbf{W}^{(L)}}\mathcal{L} \equiv \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}}$ is notational convention

Gradient descent iteration

- Backward pass

now let's go back one more layer…

again we'll draw the dependency graph:



$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}} \frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{x}^{(L-1)}} \frac{\partial \mathbf{x}^{(L-1)}}{\partial \mathbf{s}^{(L-1)}} \frac{\partial \mathbf{s}^{(L-1)}}{\partial \mathbf{W}^{(L-1)}}$$

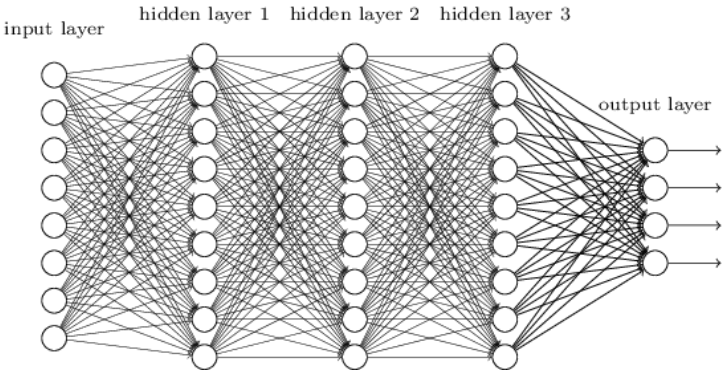
Network computation: summary

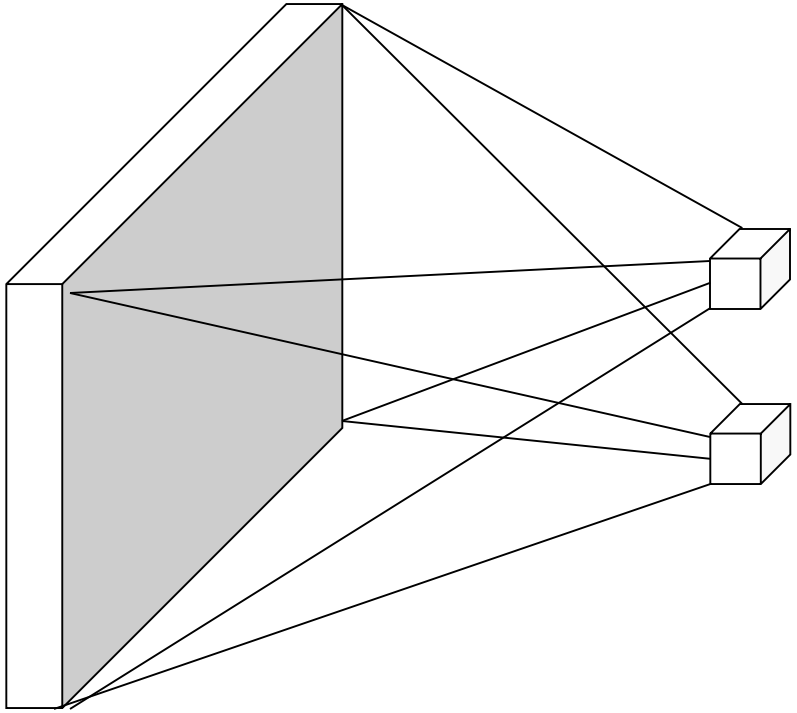
- Inference/Prediction: forward pass
- Learning from data: backward pass
- Core method: gradient descent

Outline

- What and why
  - Modeling perspective
  - Biological and computational motivation
- Deep networks
  - Building blocks
  - Structure design
  - Computation: prediction & learning
- Deep networks for images
  - CNN family
  - CNNs for vision tasks

Neural networks for images

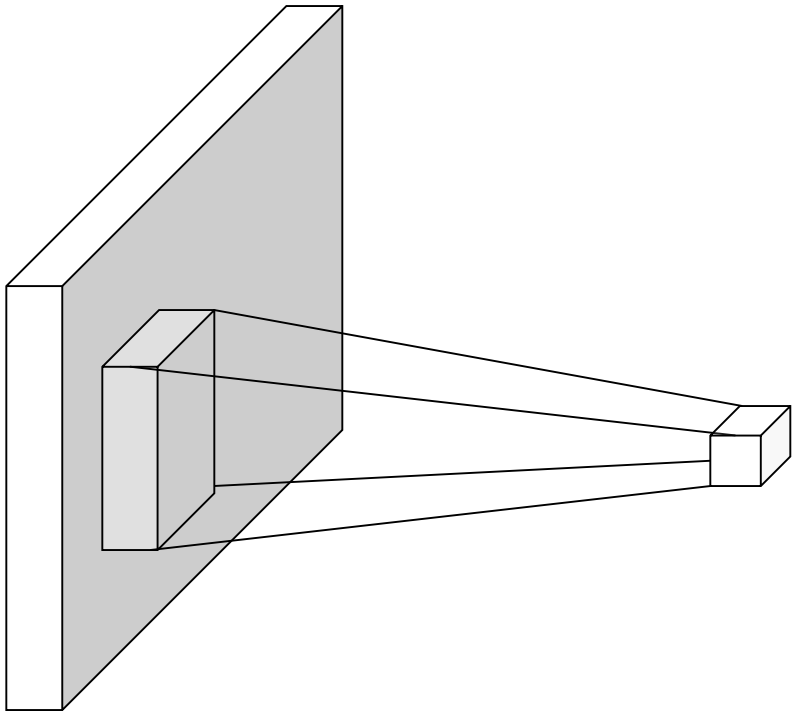




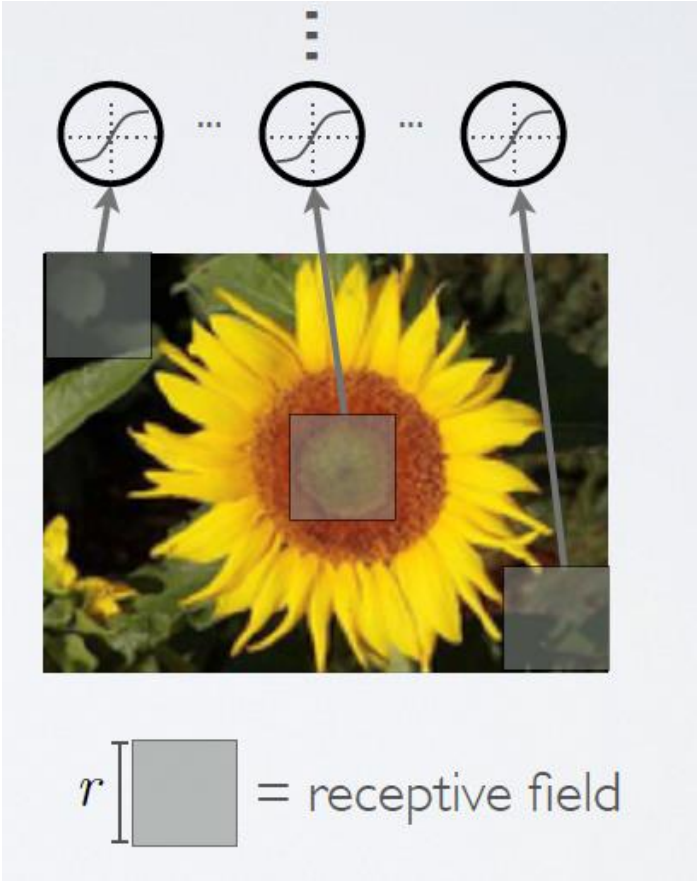
image

Fully connected layer

Neural networks for images

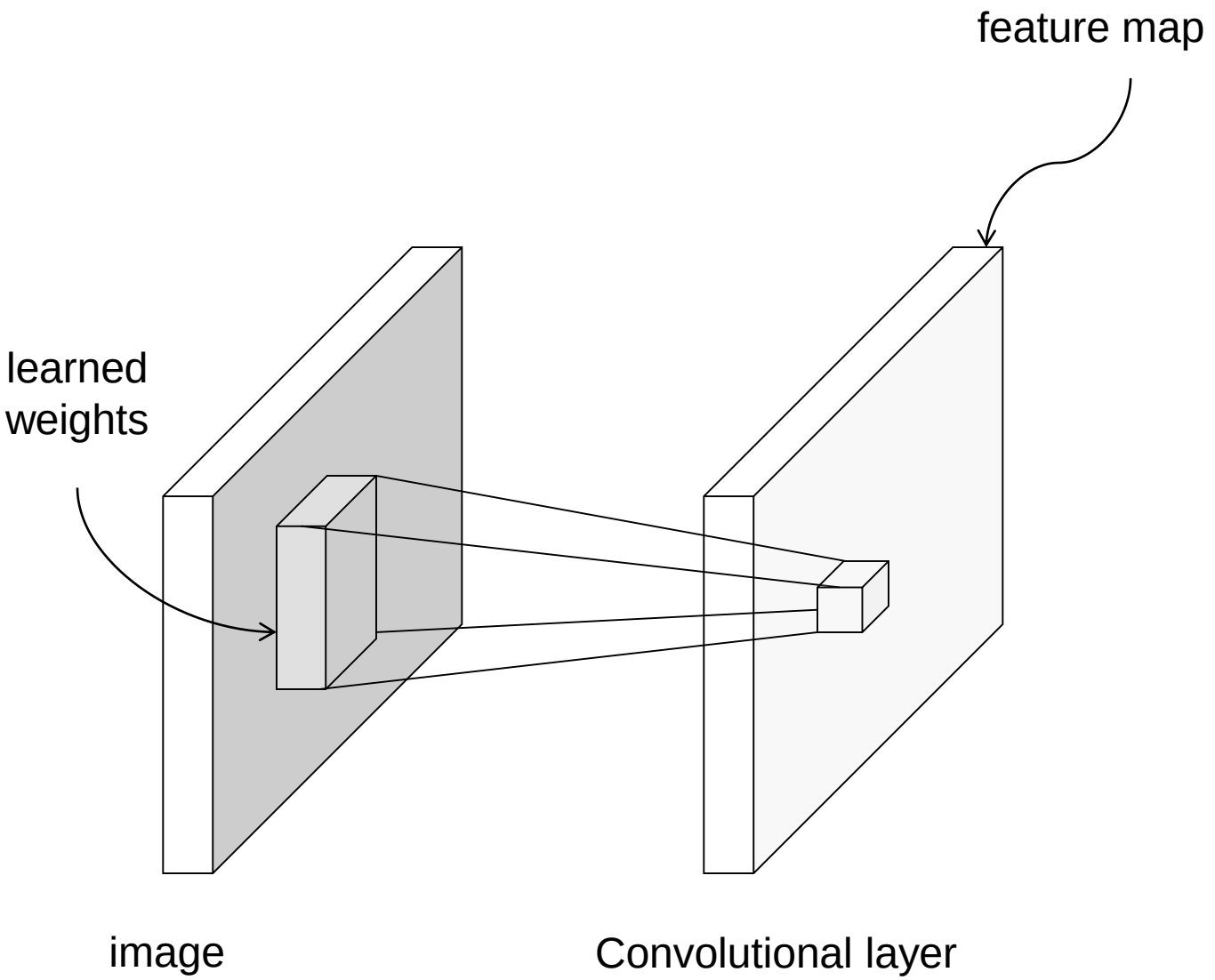


image

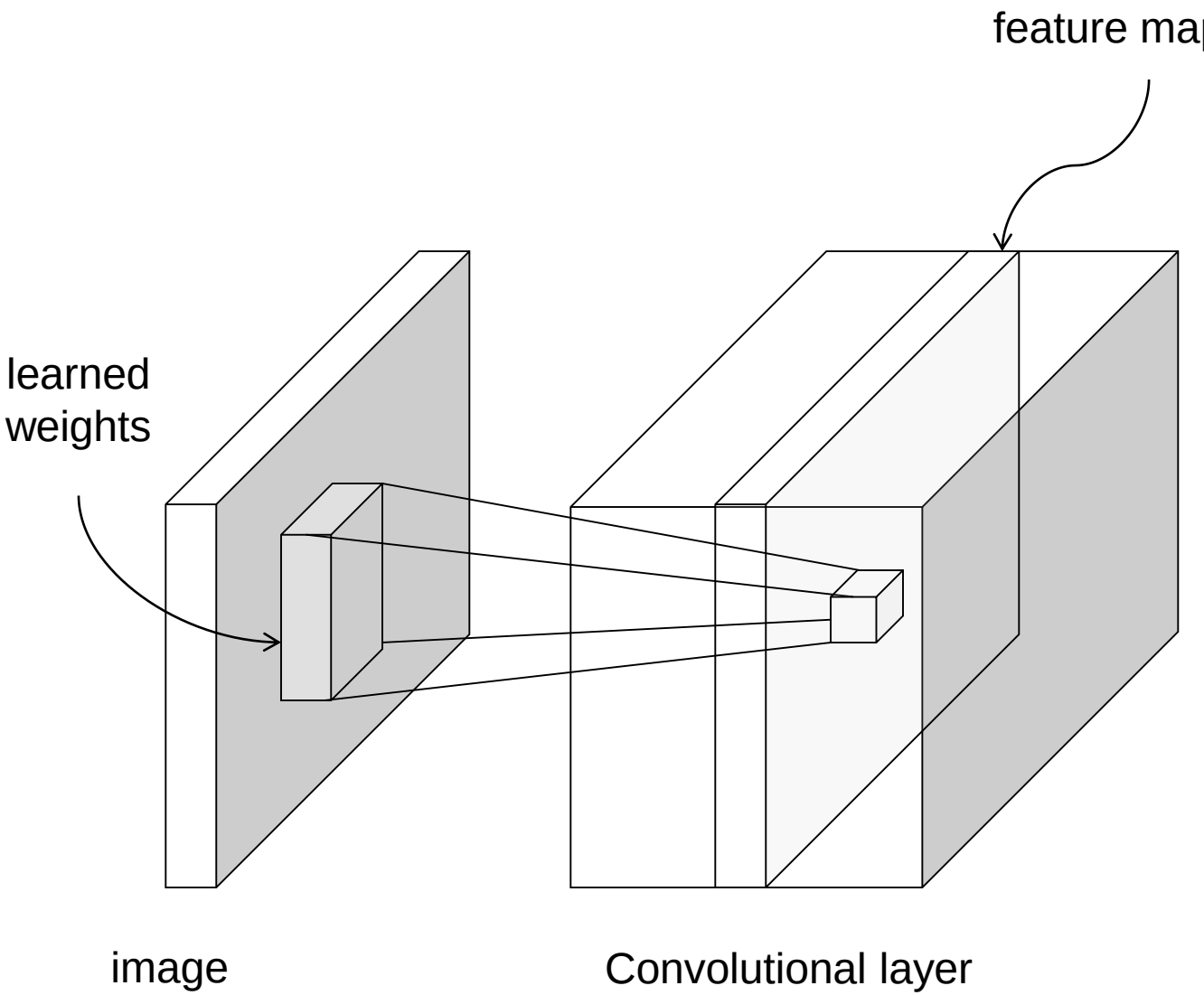


Convolutional layer

Neural networks for images

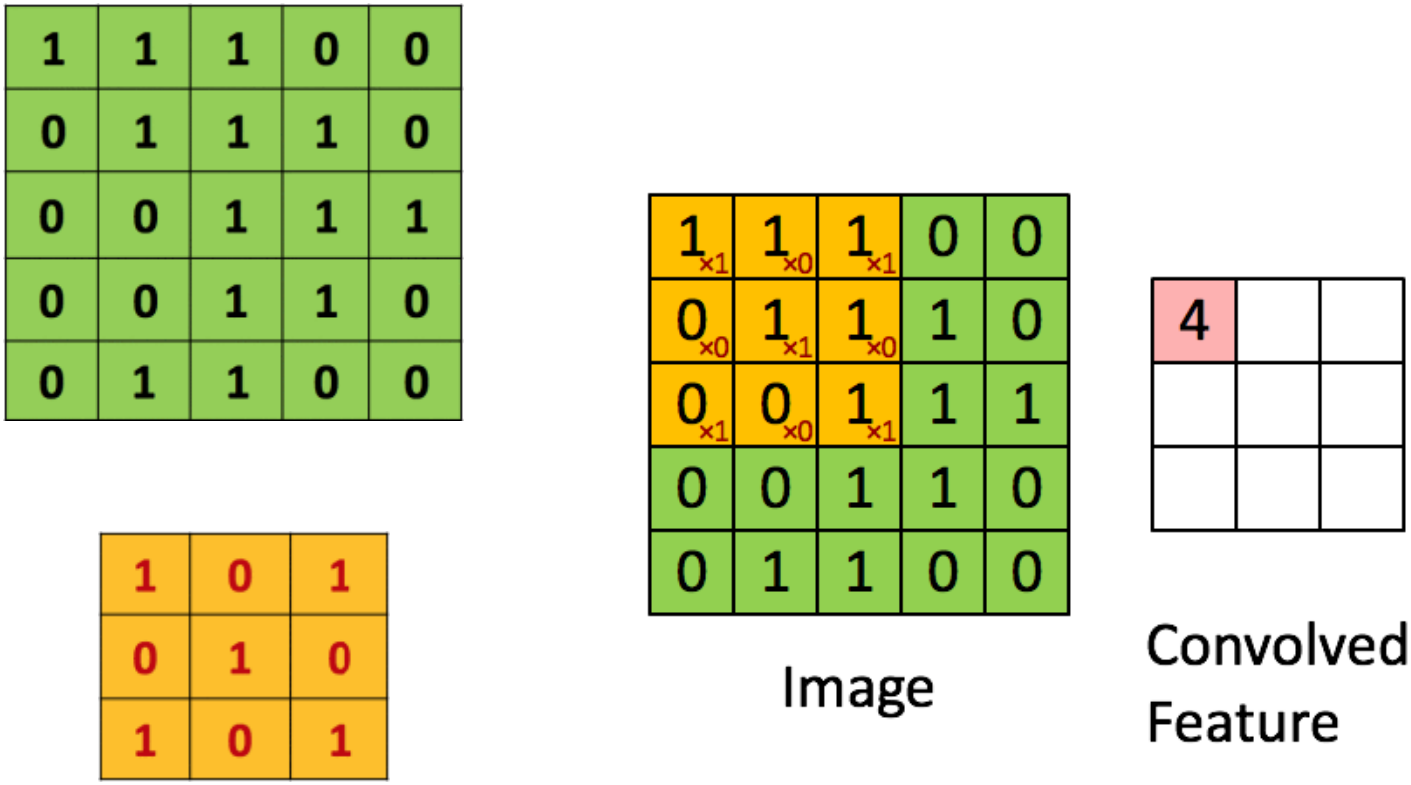


Neural networks for images

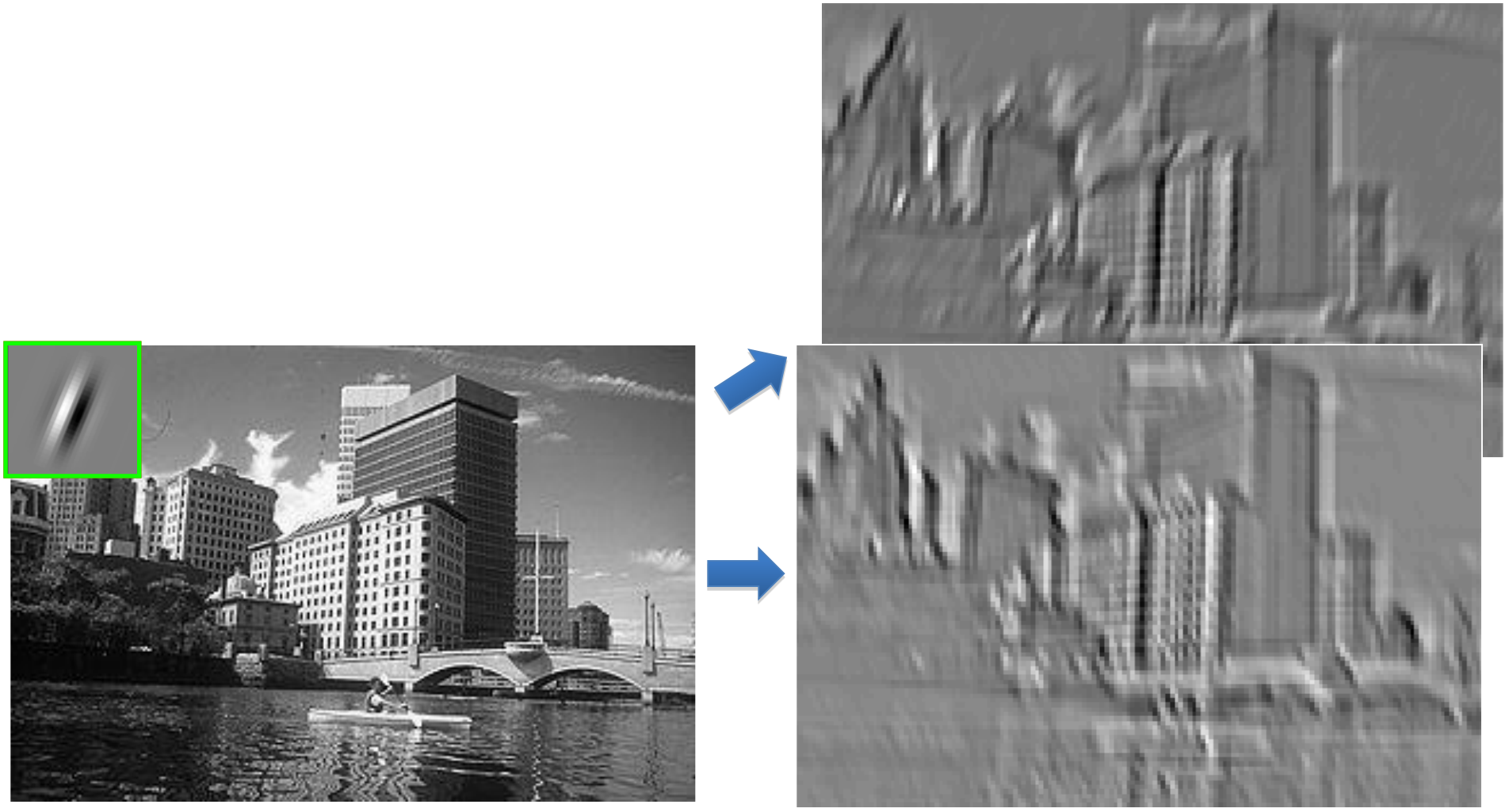


Convolution Layers

- A set of neurons with shared weights, different location



Convolution as feature extraction

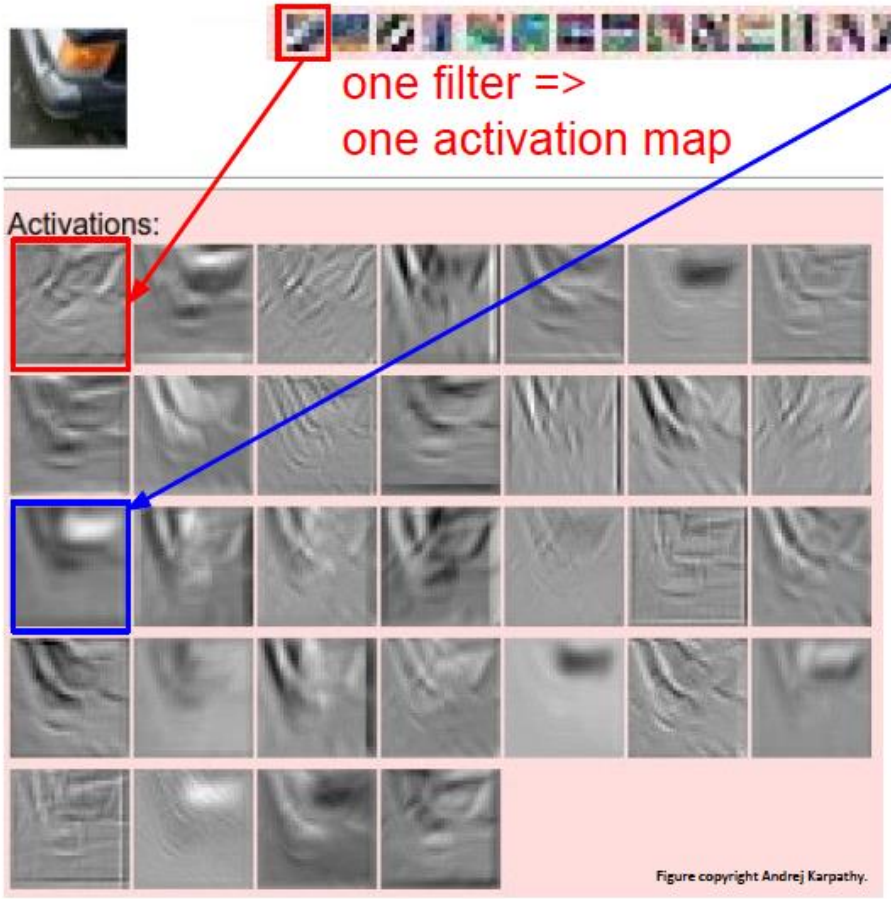


Input

Feature Map

Convolution Layers

- Interpretation



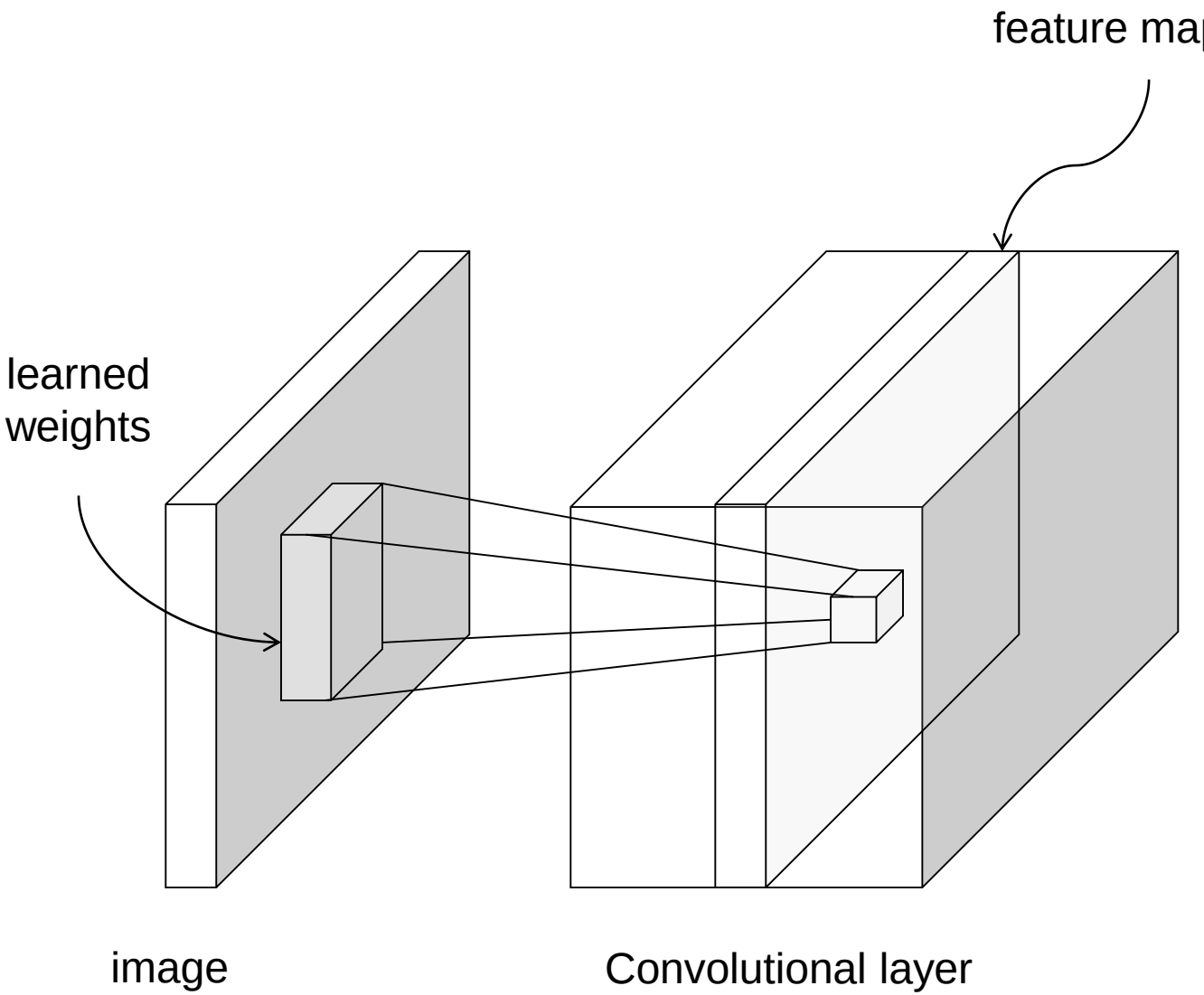
example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

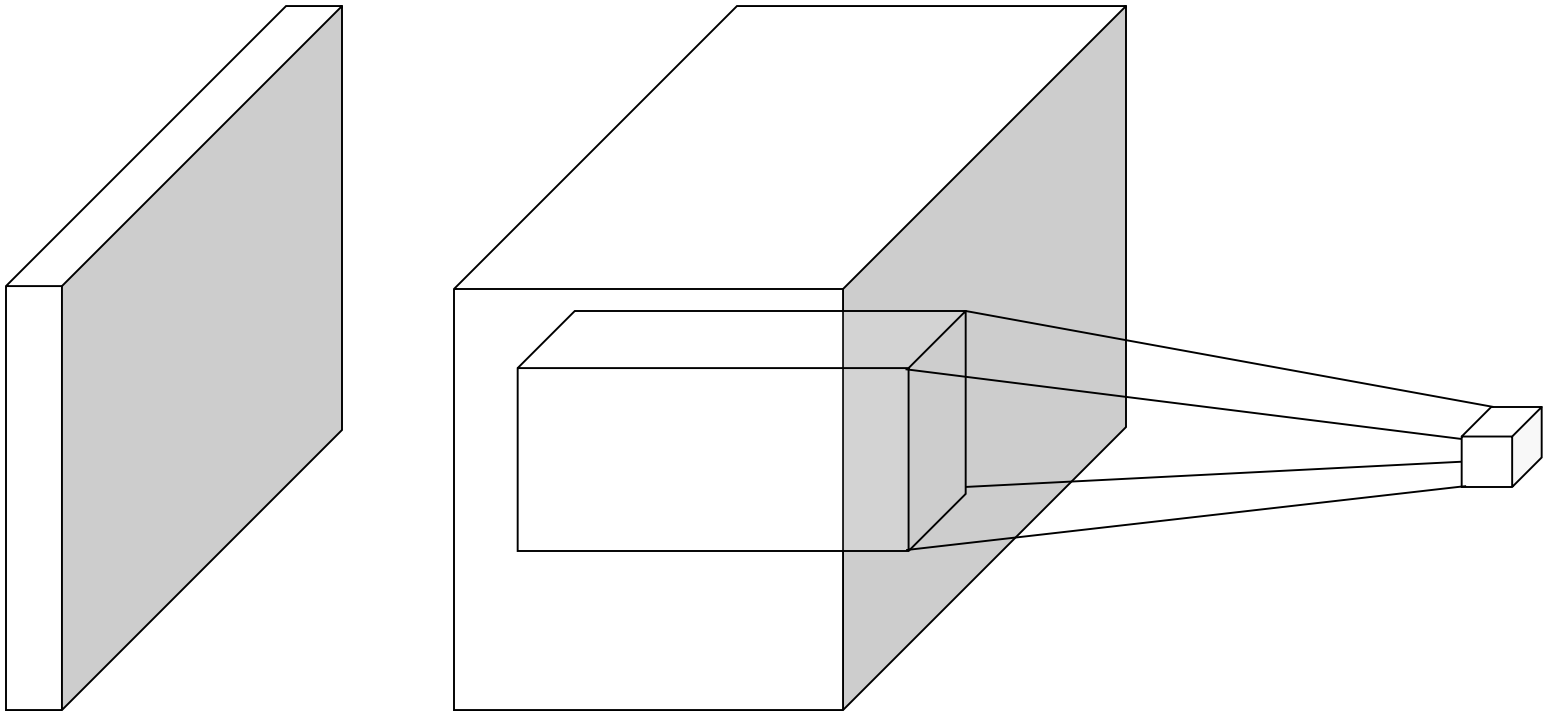
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

Neural networks for images



Neural networks for images

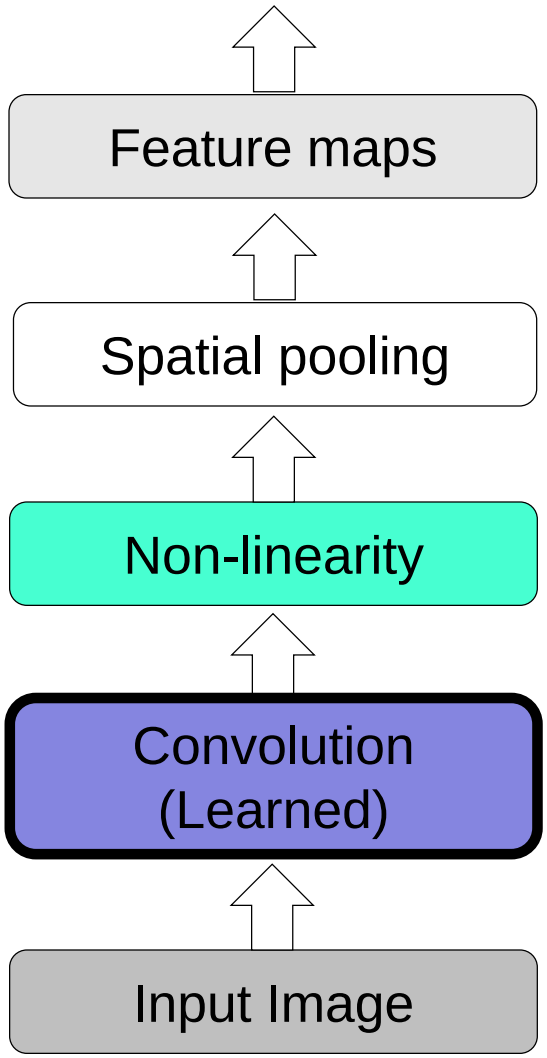


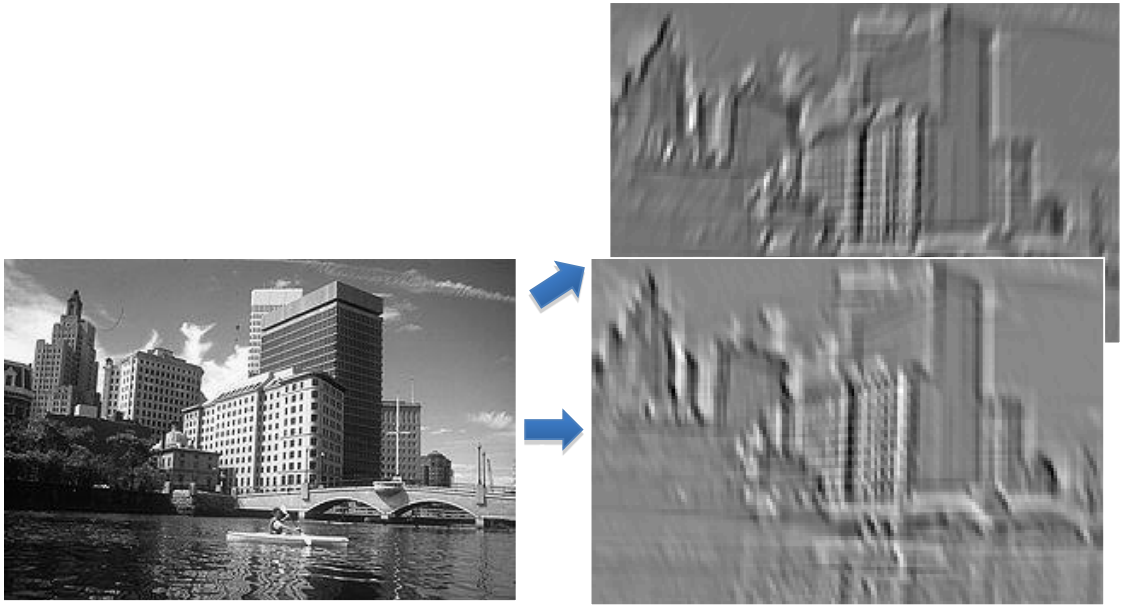
image

Convolutional layer
+ ReLU

next
layer

Key operations in a CNN



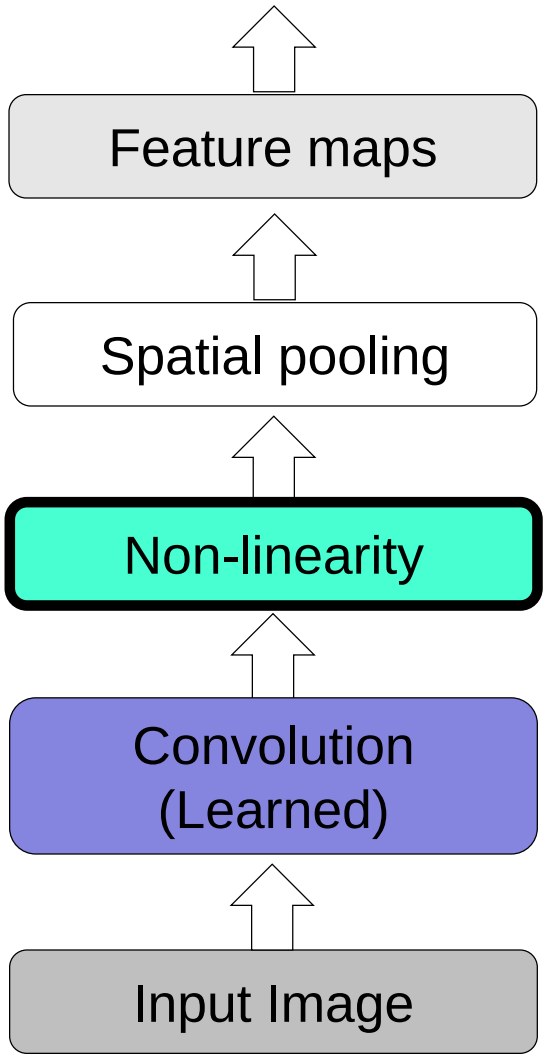


Input

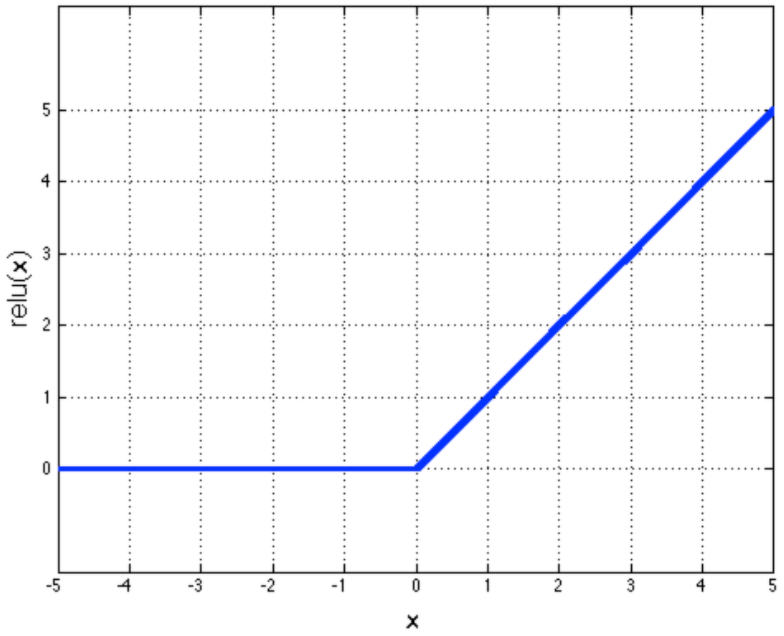
Feature Map

Source: R. Fergus, Y. LeCun

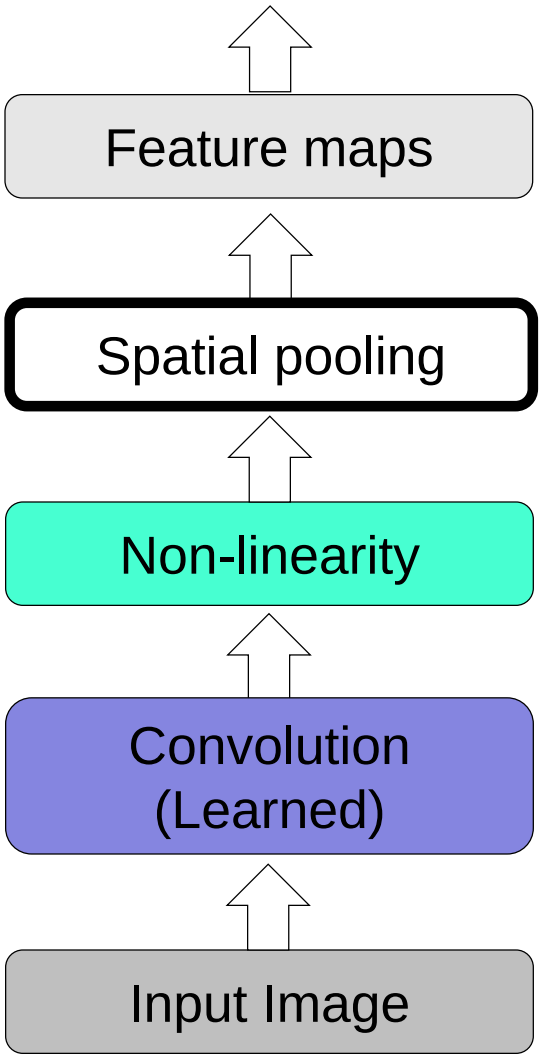
Key operations in a CNN

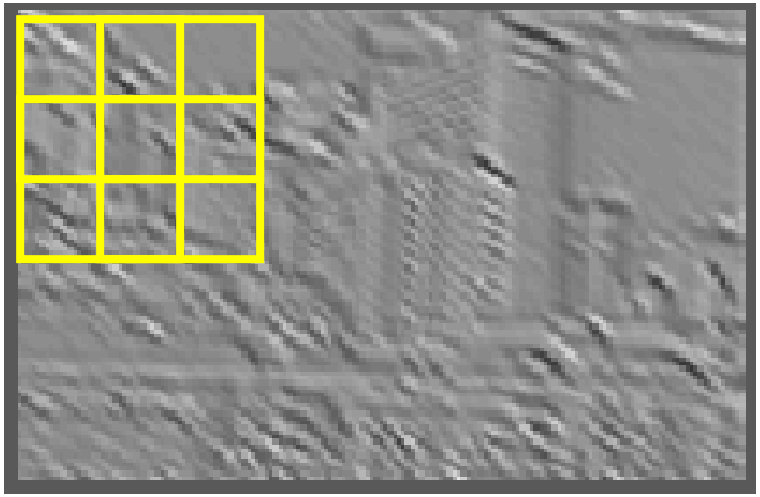


Rectified Linear Unit (ReLU)

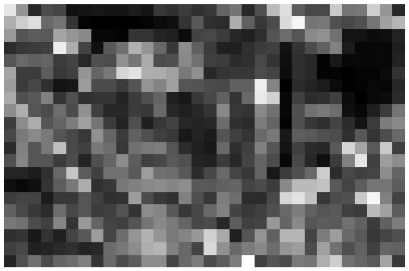


Key operations in a CNN



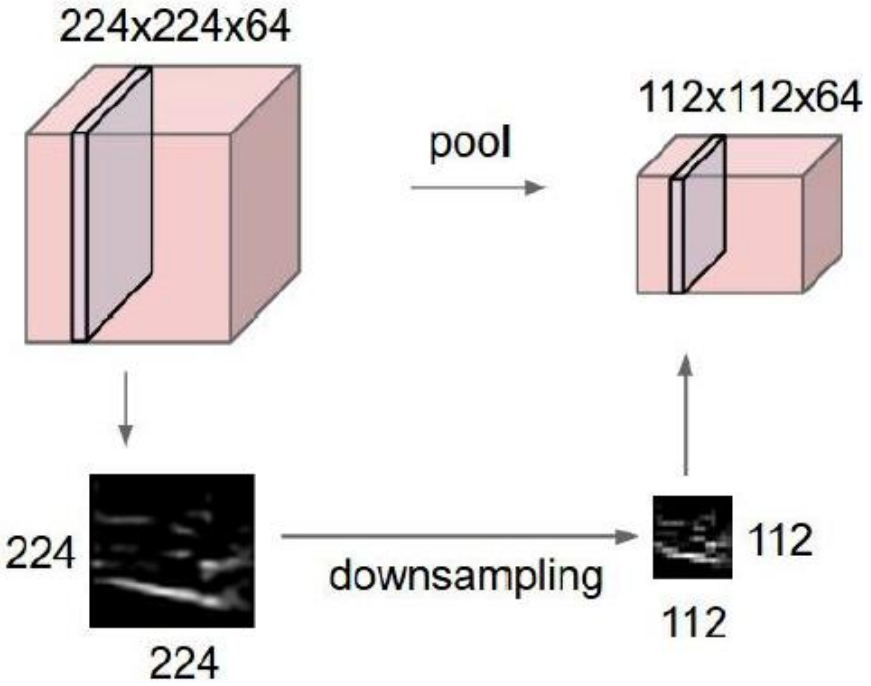


Max



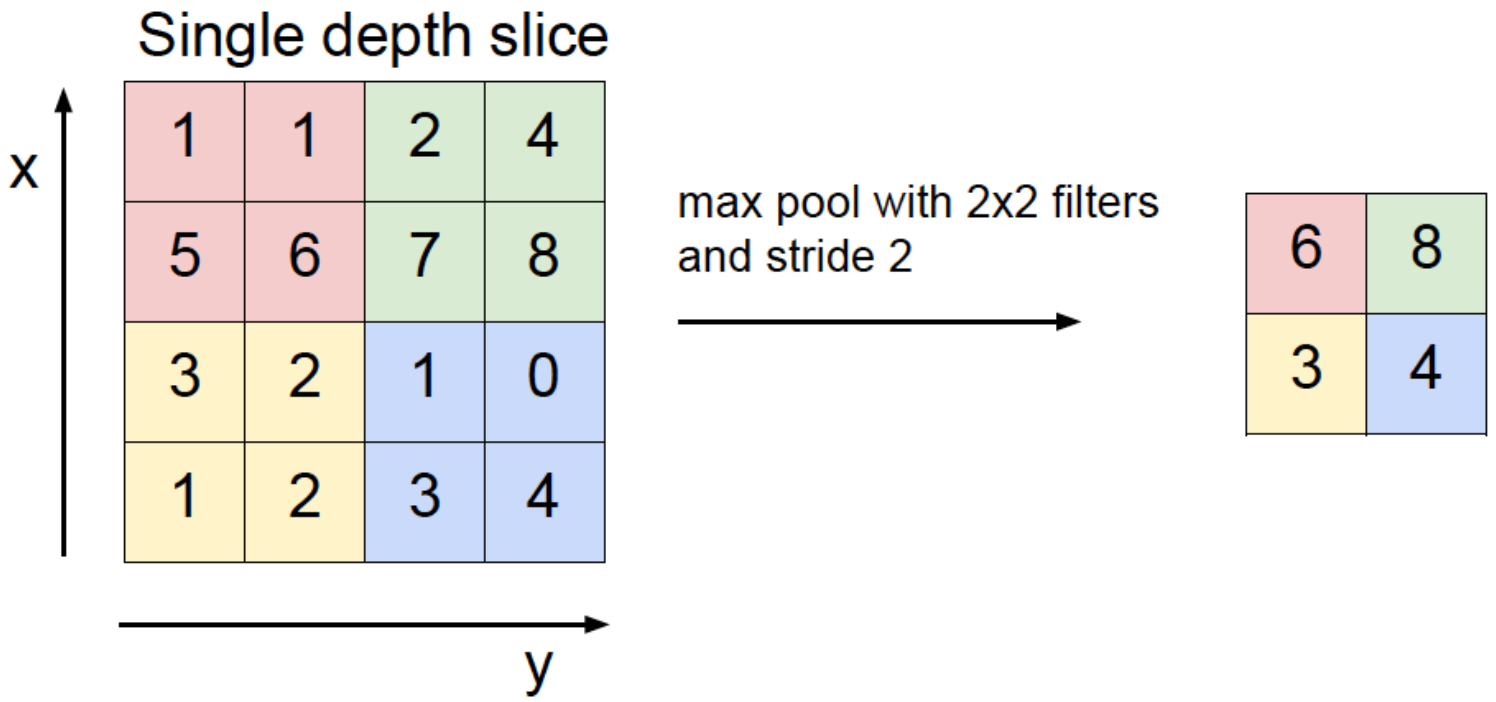
Pooling Layers

- Reducing spatial resolution
  - Make the representations smaller and more manageable
  - Generate larger receptive field
  - Operate over each feature channel independently

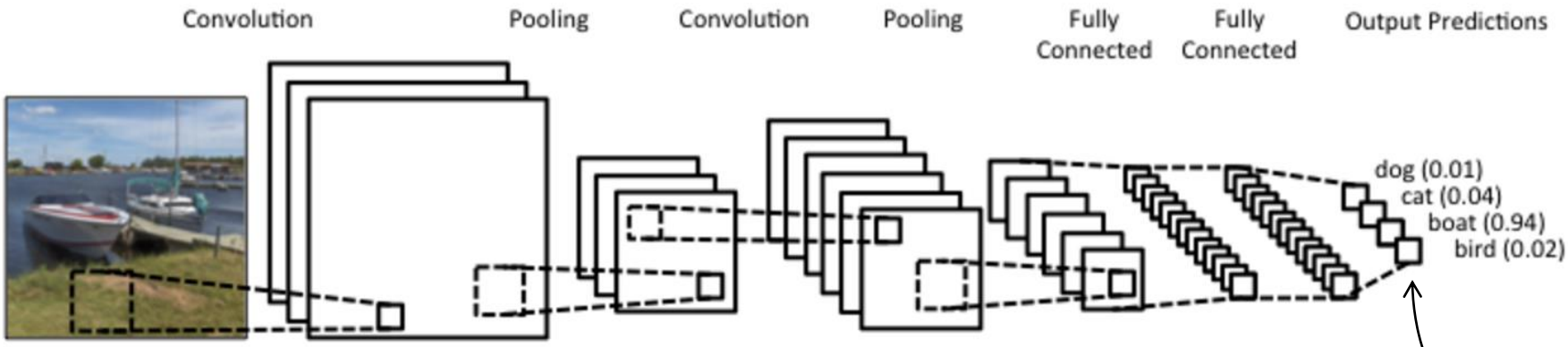


Pooling Layers

- Example: Max pooling



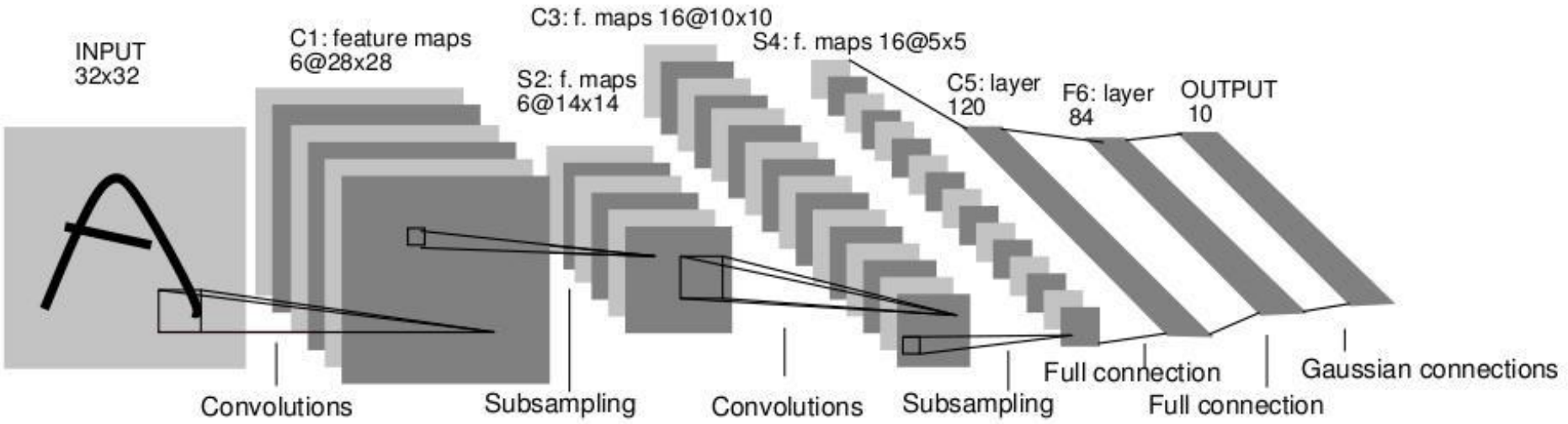
Key operations in a CNN



Softmax layer:

$$P(c \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_c \times \mathbf{x})}{\sum_{k=1}^{C} \exp(\mathbf{w}_k \times \mathbf{x})}$$

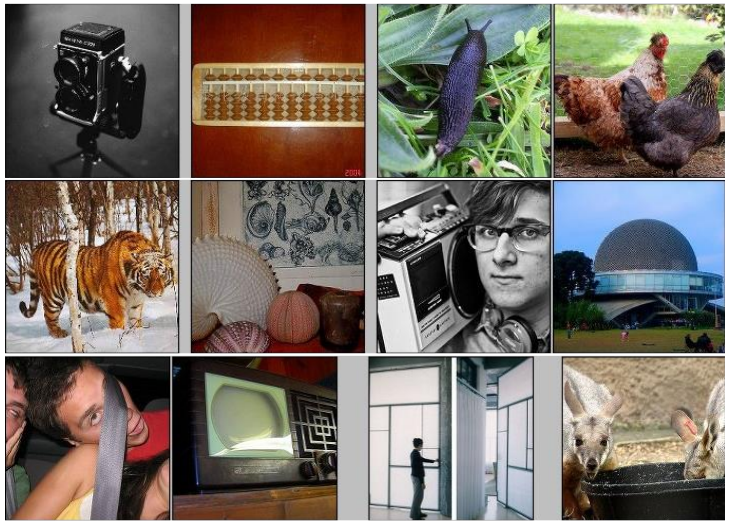
History of CNNs: LeNet-5



- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

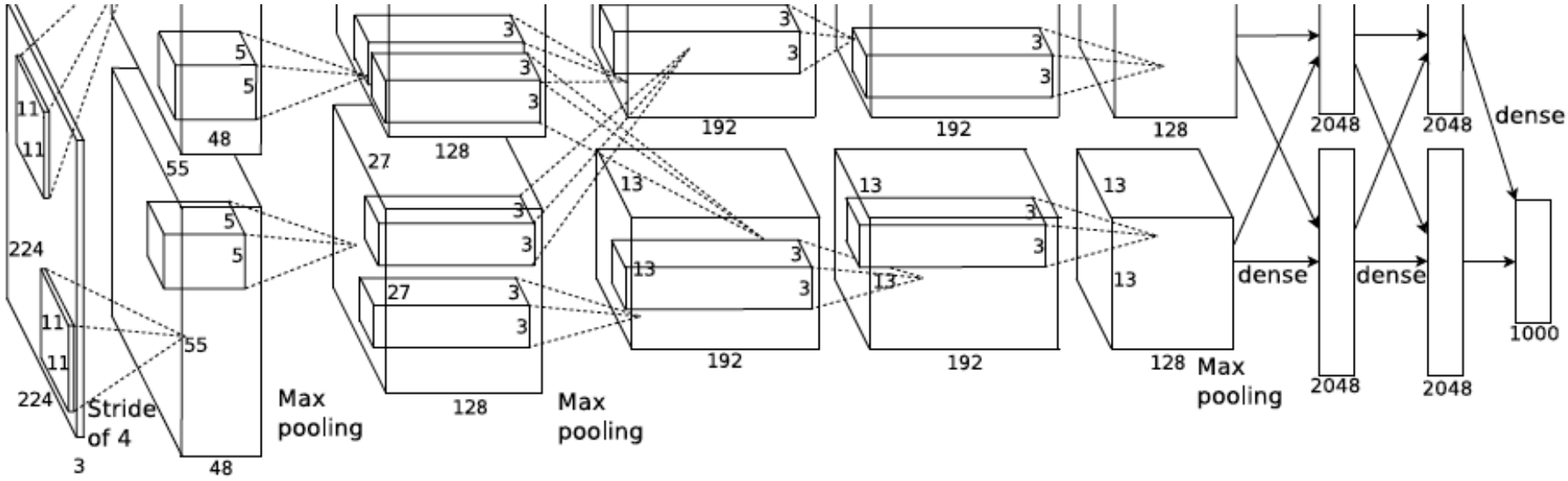
ImageNet dataset



- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

AlexNet: ILSVRC 2012 winner



- Similar framework to LeNet but:
  - Max pooling, ReLU nonlinearity
  - More data and bigger model (7 hidden layers, 650K units, 60M params)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Clarifai: ILSVRC 2013 winner

- Refinement of AlexNet

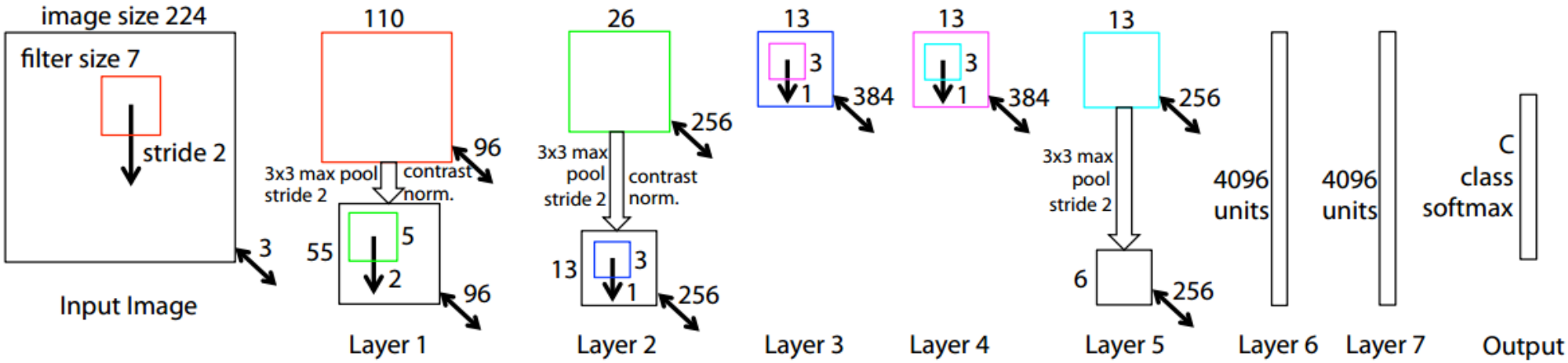
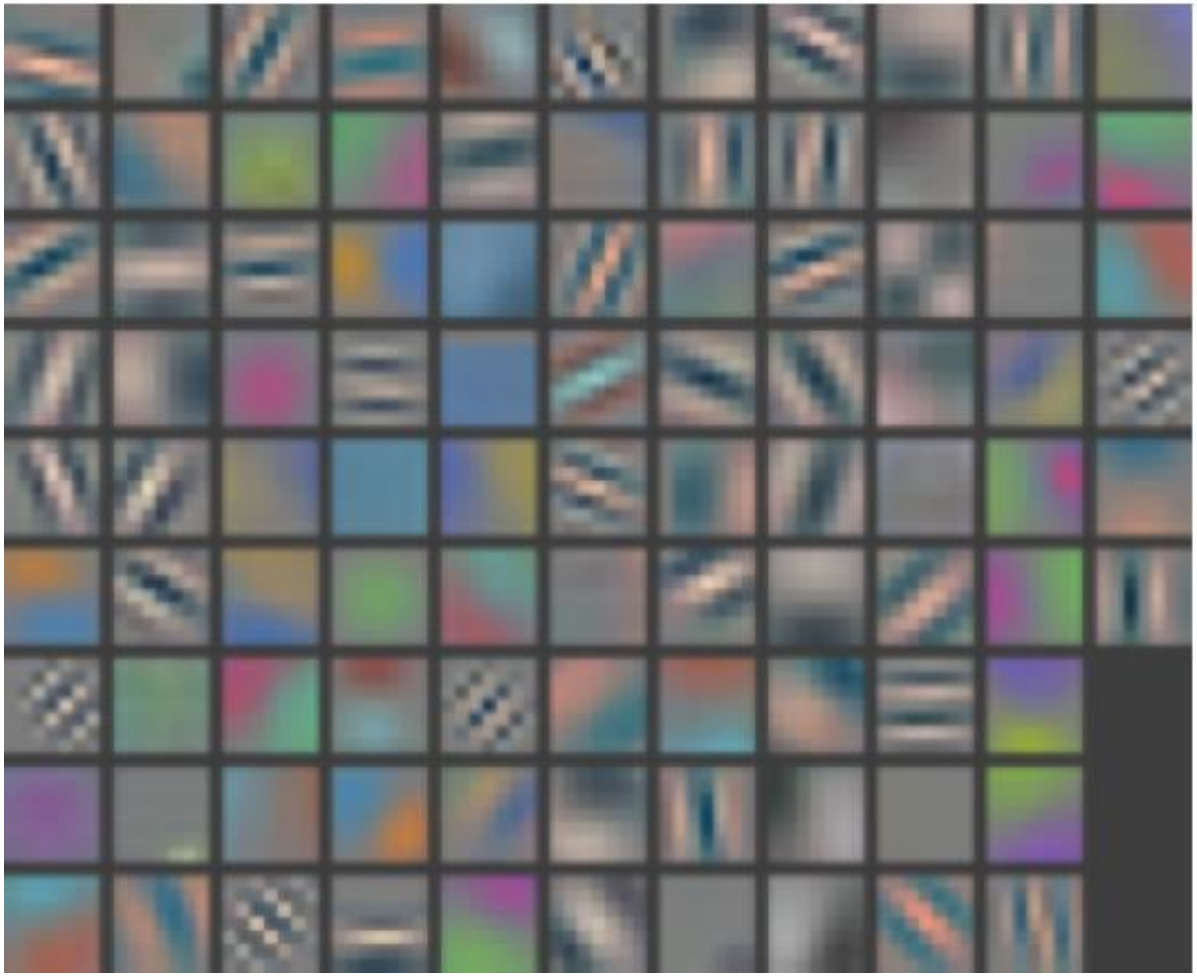


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

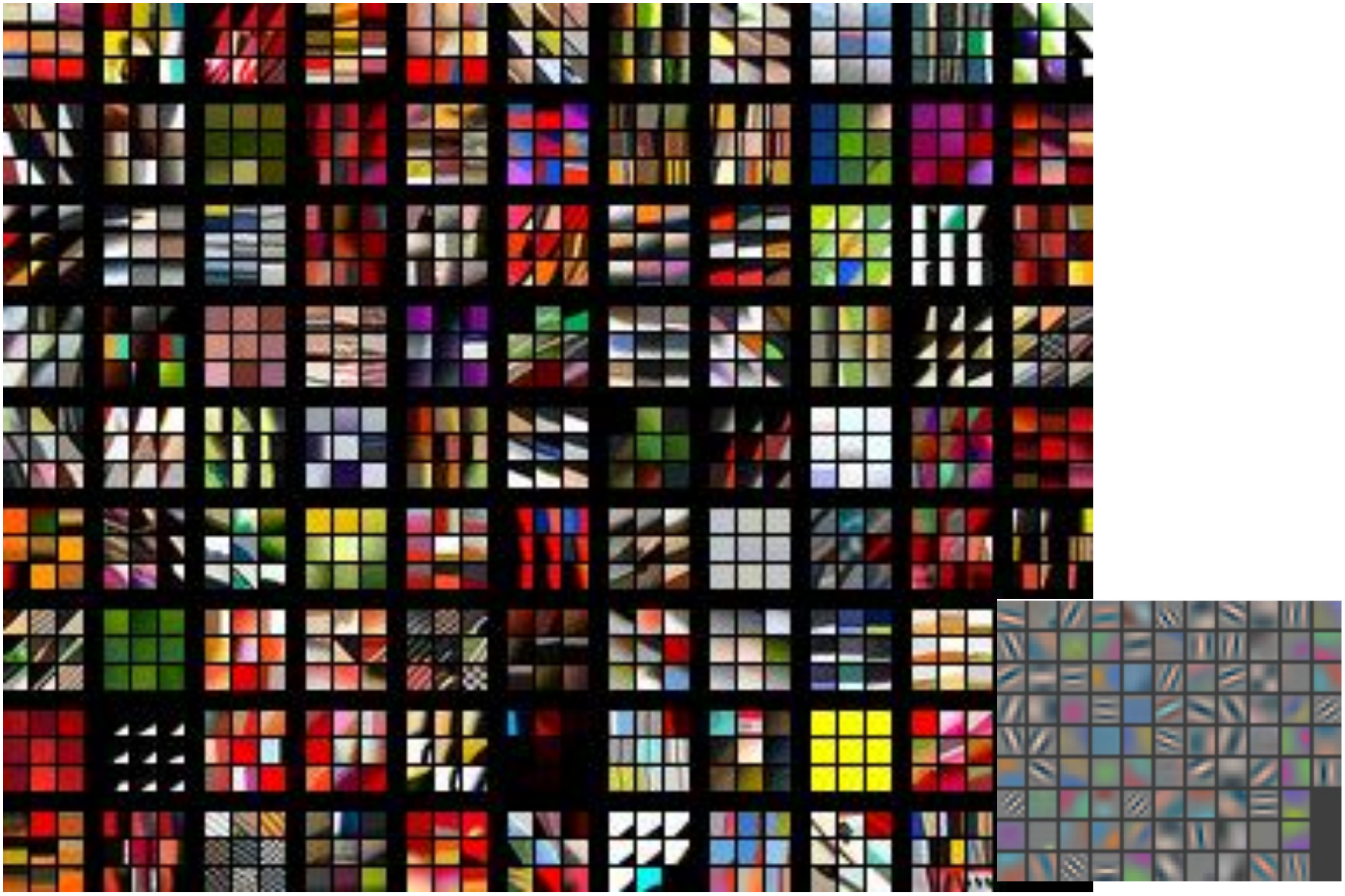
M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014 (Best Paper Award winner)

Layer 1 Filters

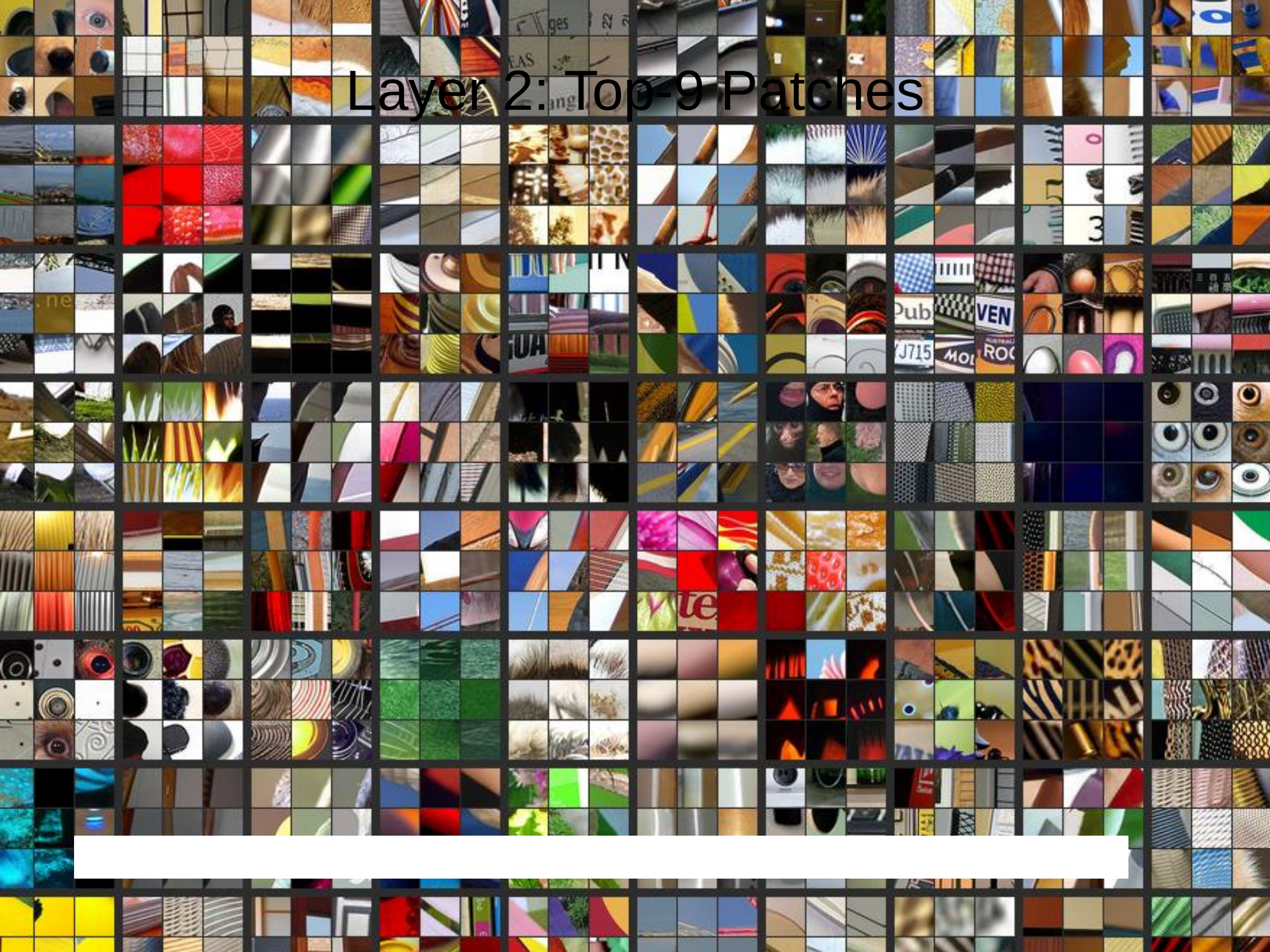


M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014 (Best Paper Award winner)

Layer 1: Top-9 Patches



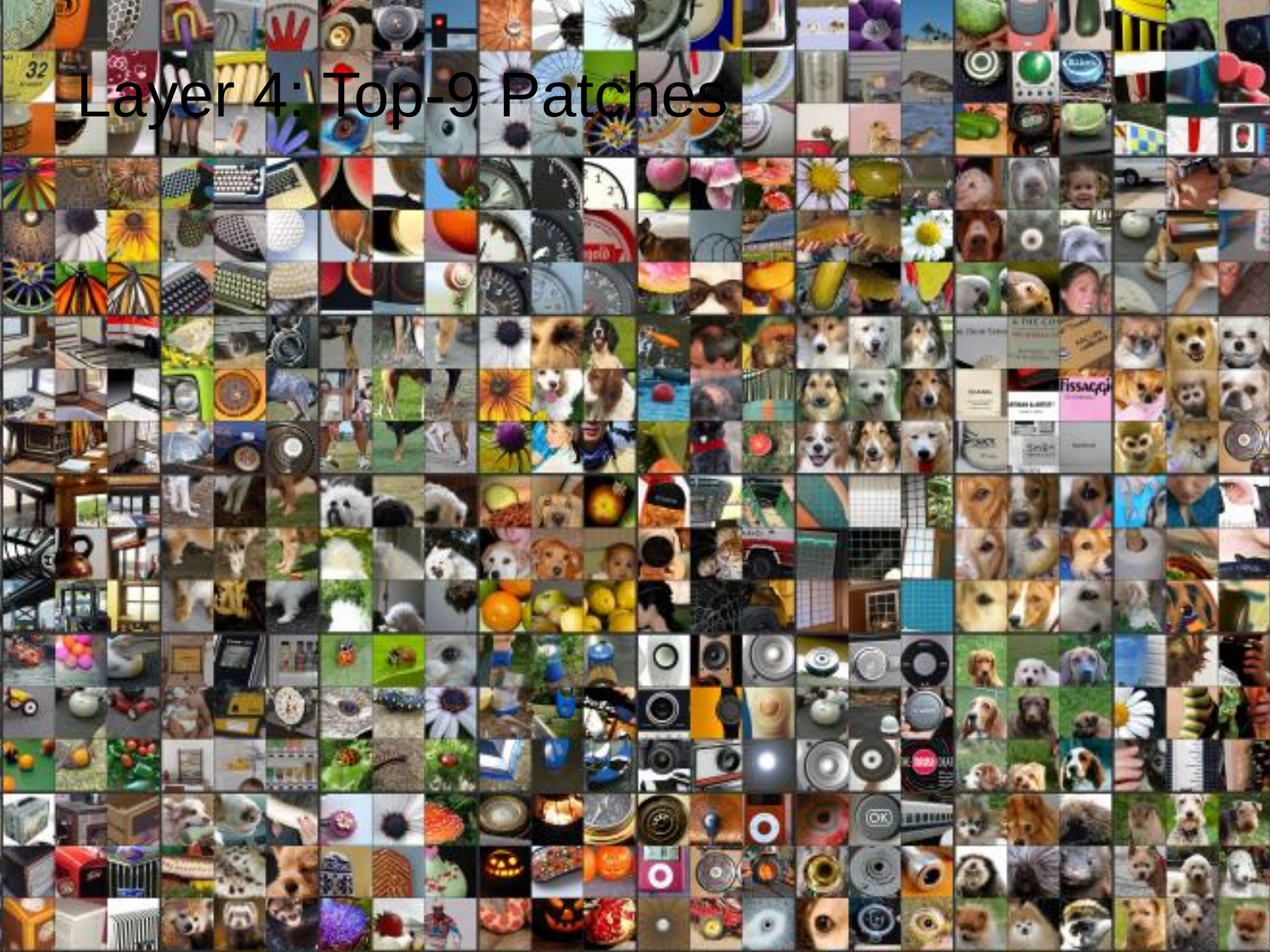
Layer 2: Top-9 Patches



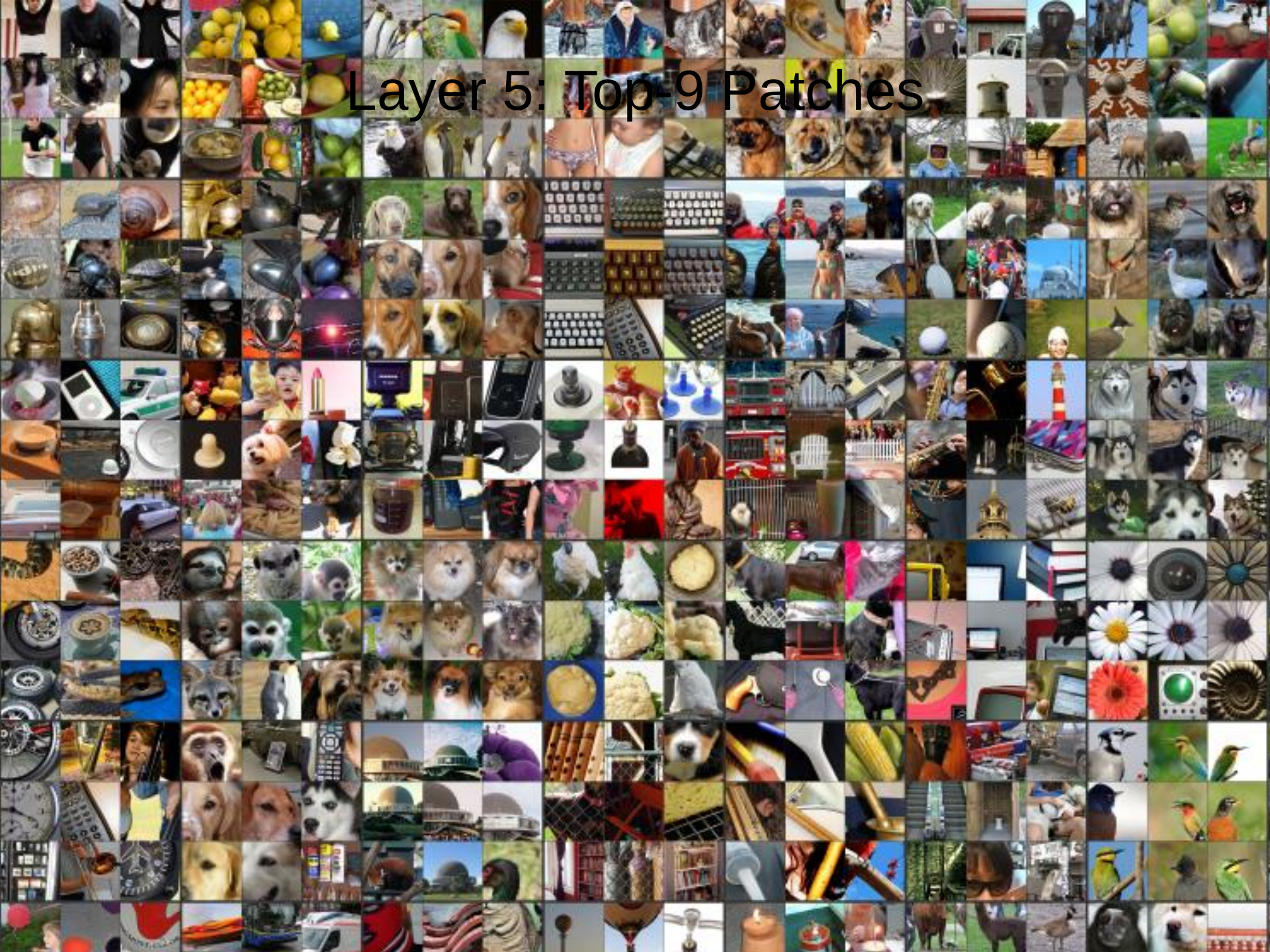
Layer 2: Top-9 Patches


Layer 3: Top-9 Patches

Layer 3: Top-9 Patches


Layer 4: Top-9 Patches

Layer 4: Top-9 Patches


Layer 5: Top-9 Patches

Layer 5: Top-9 Patches

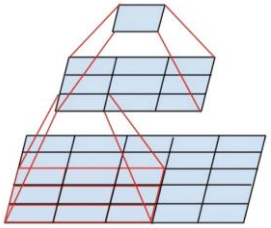
VGGNet: ILSVRC 2014 2nd place

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

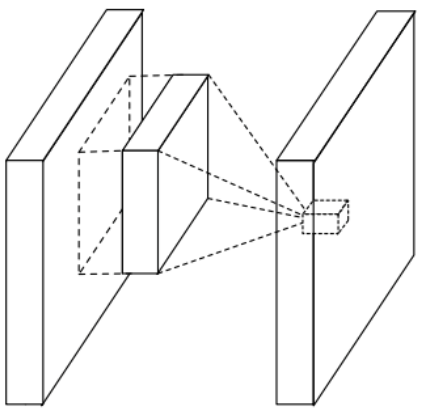
- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)



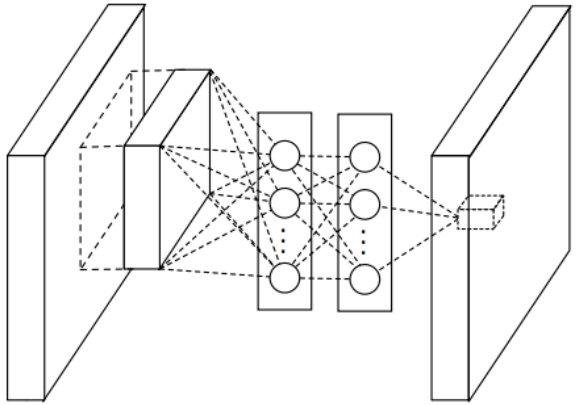
- One 7x7 conv layer with C feature maps needs $49C^2$ weights, three 3x3 conv layers need only $27C^2$ weights
- Experimented with 1x1 convolutions

K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

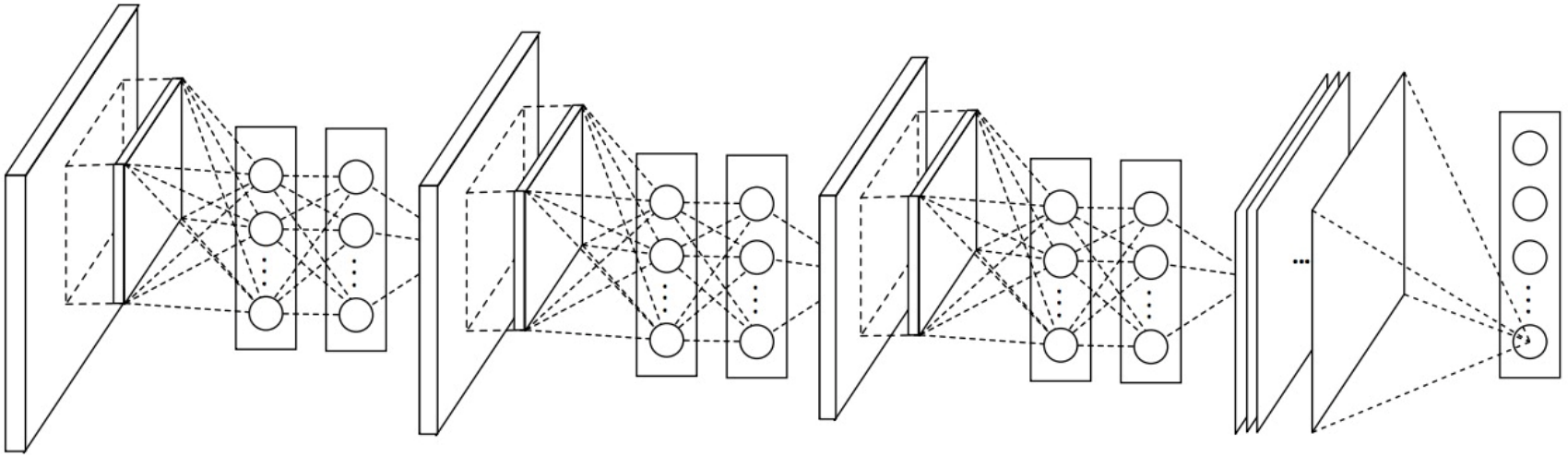
Network in network



(a) Linear convolution layer

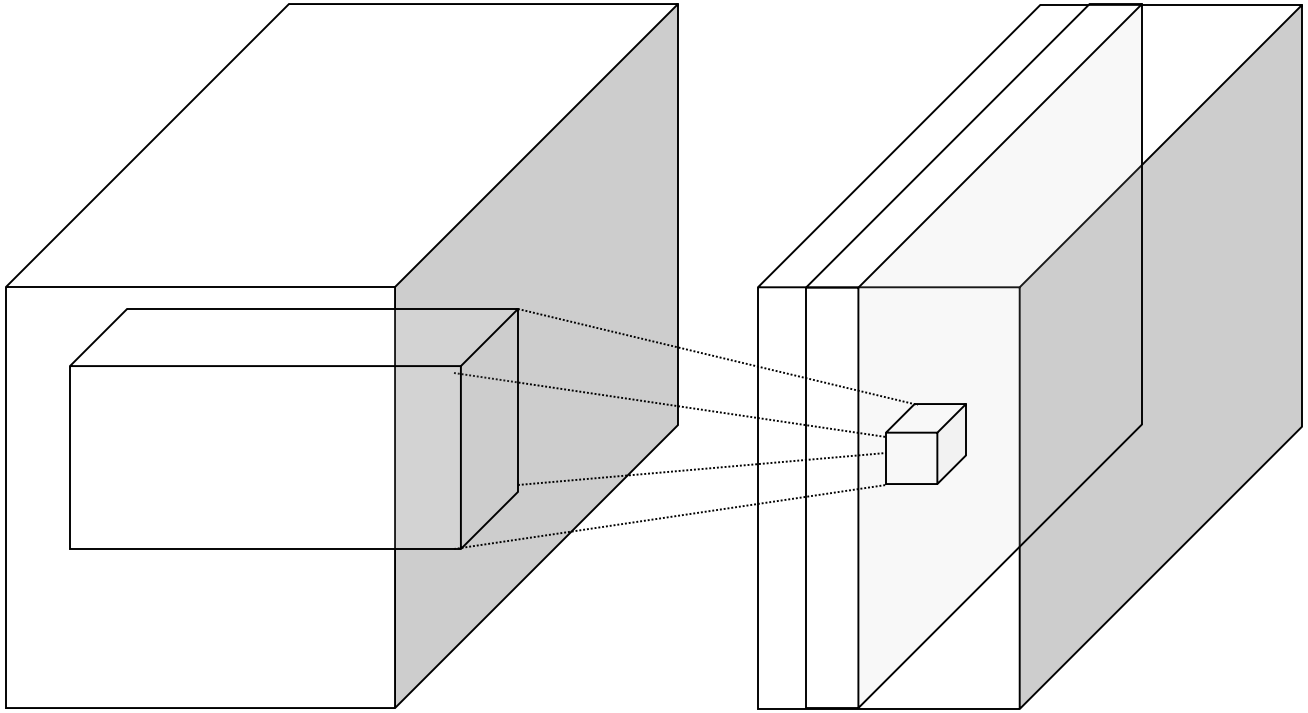


(b) Mlpconv layer



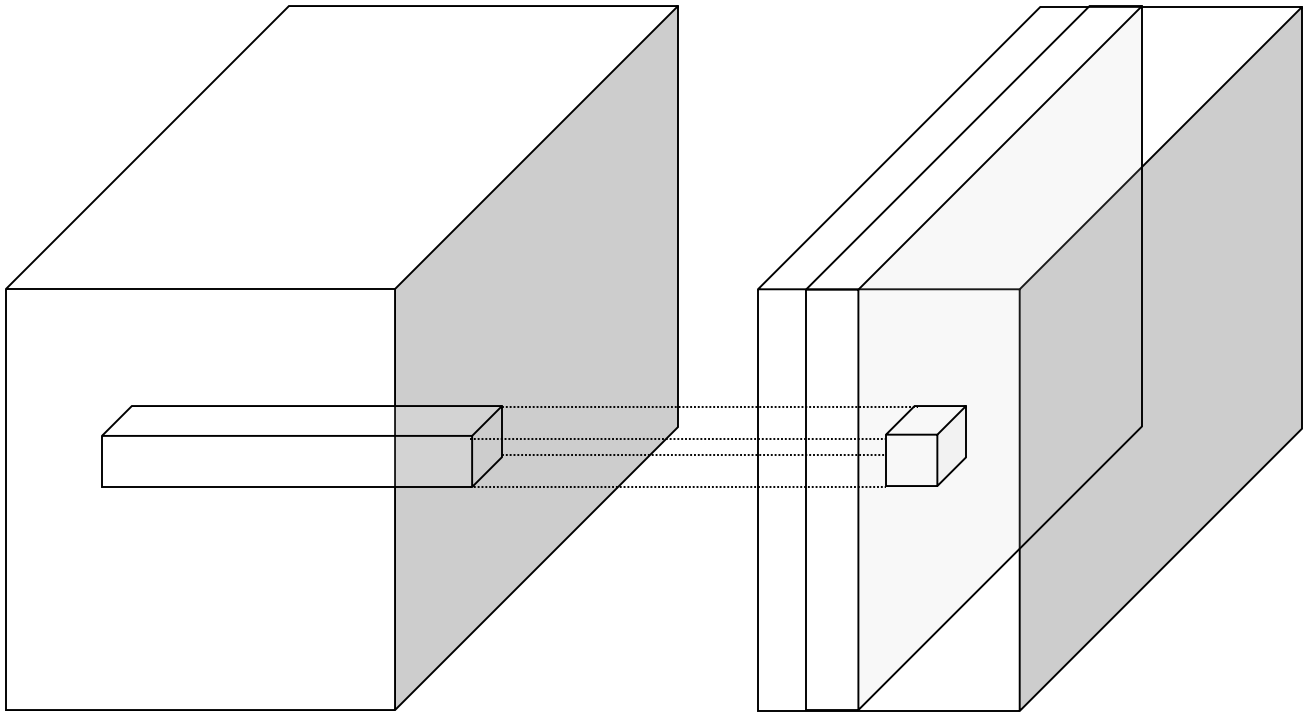
M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014

1x1 convolutions



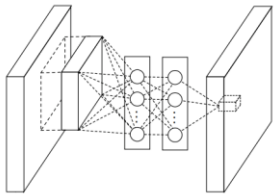
conv layer

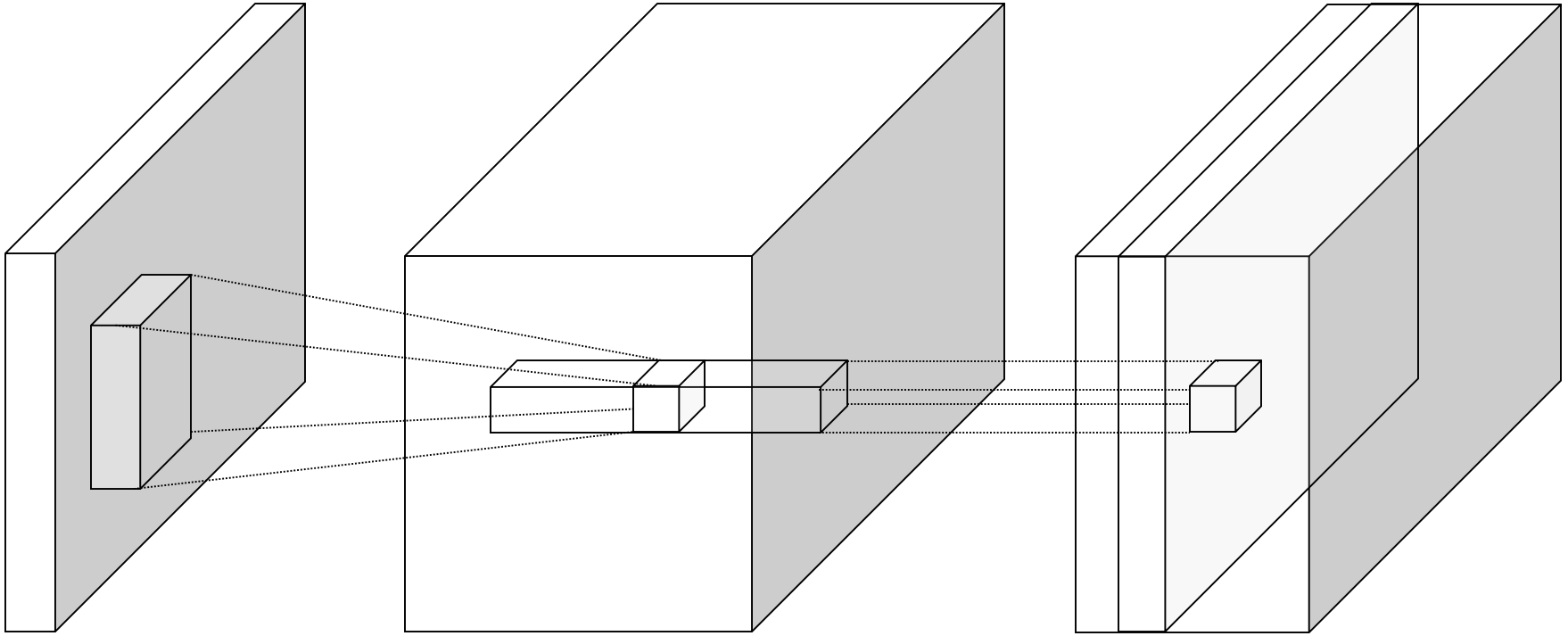
1x1 convolutions



1x1 conv layer

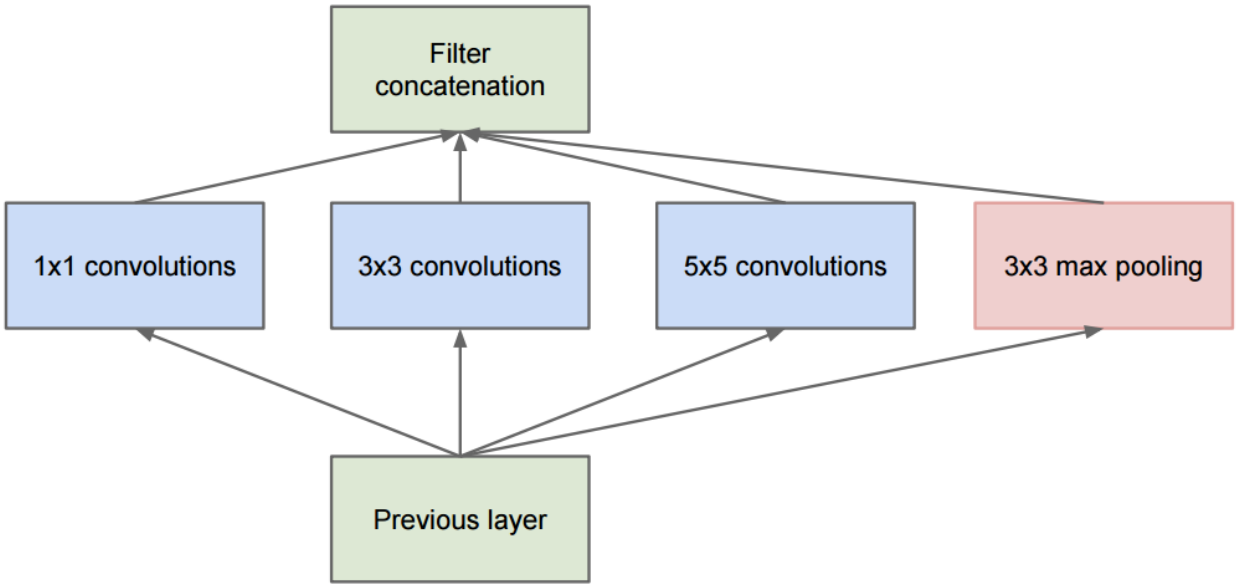
1x1 convolutions





GoogLeNet

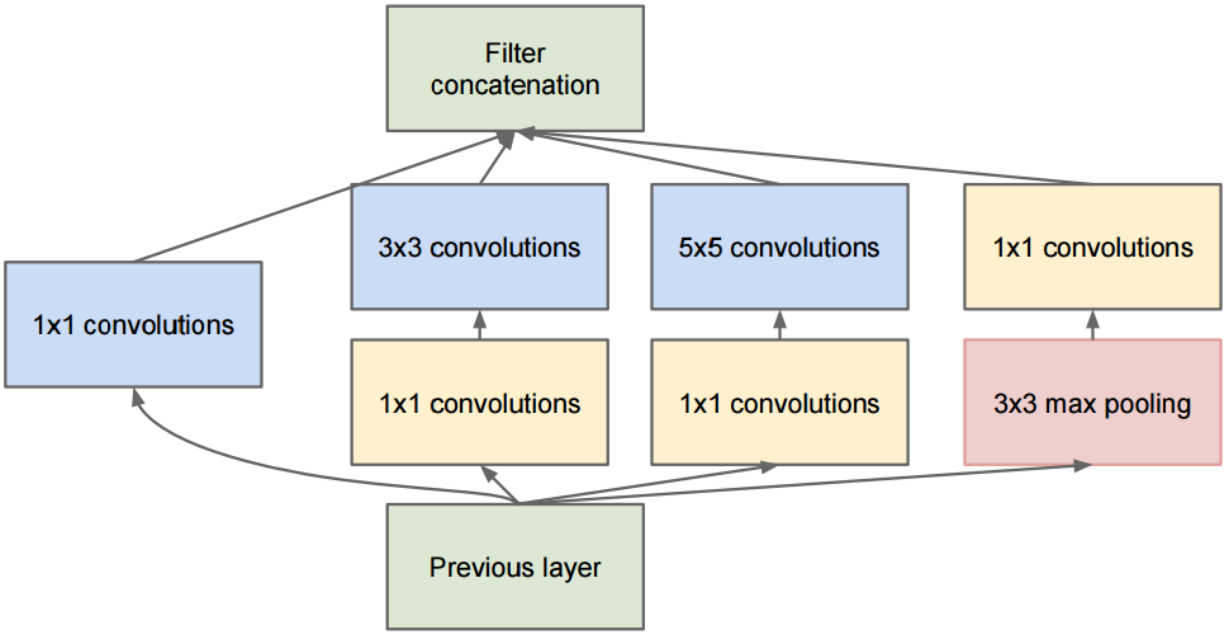
- The Inception Module
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

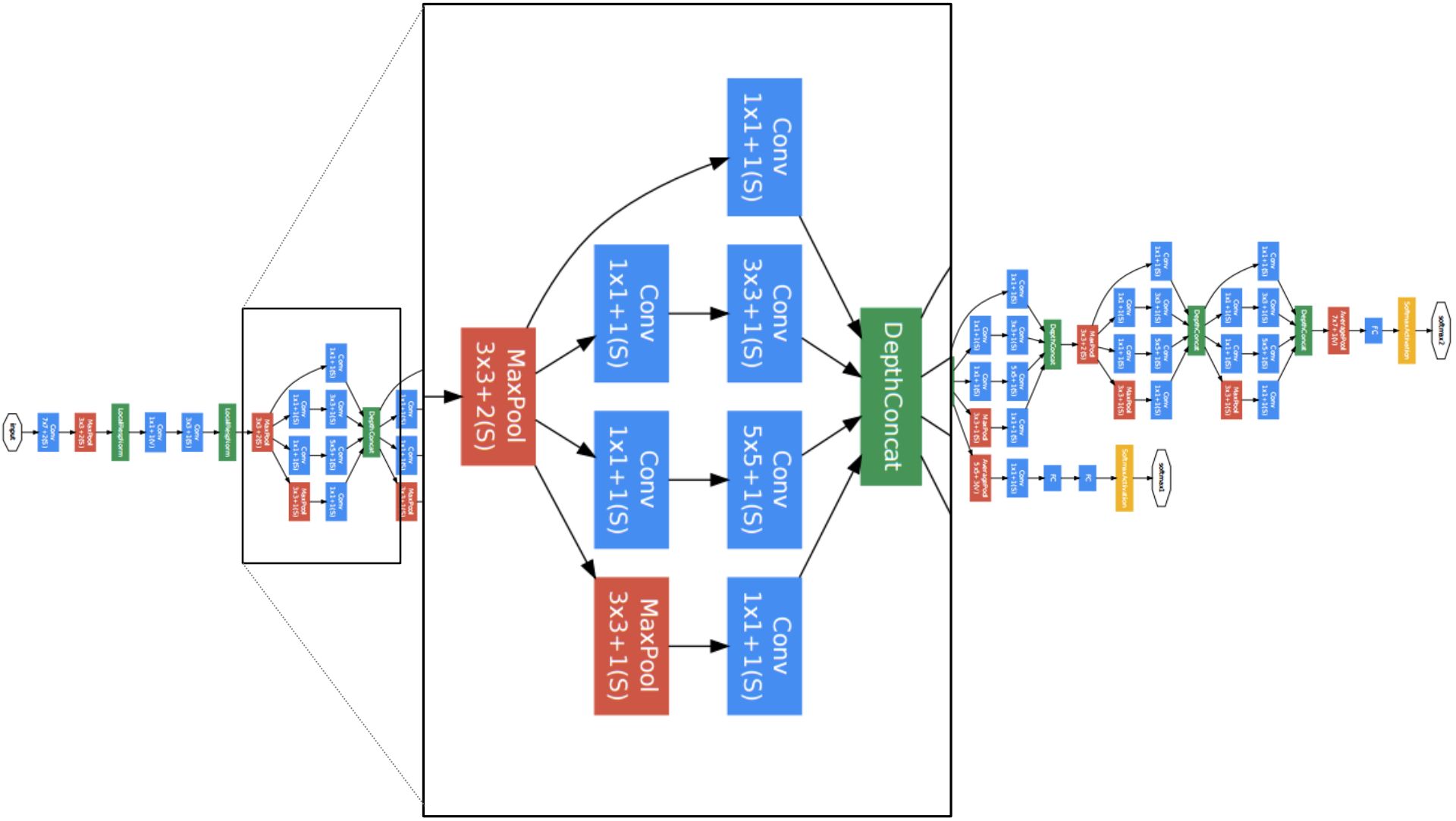
GoogLeNet

- The Inception Module
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps
  - Use 1x1 convolutions for dimensionality reduction before expensive convolutions



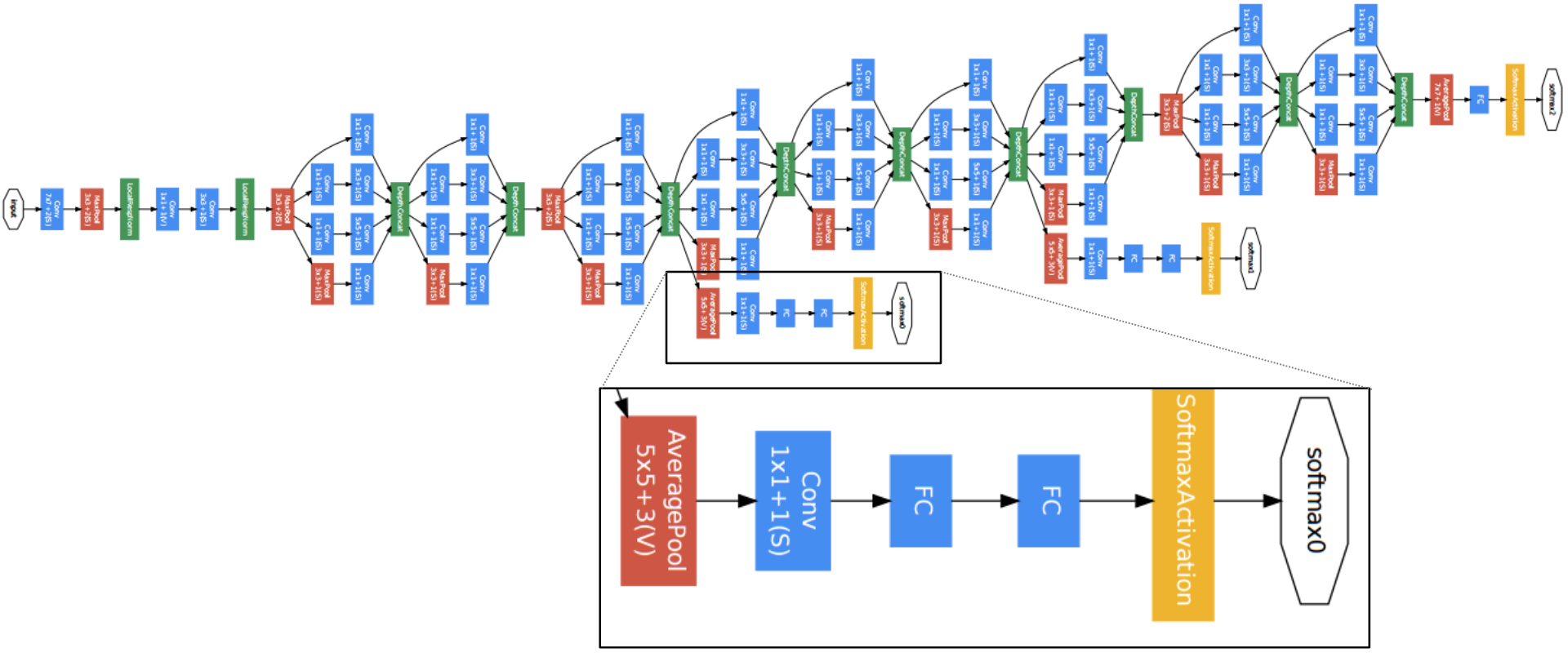
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

ResNet: ILSVRC 2015 winner

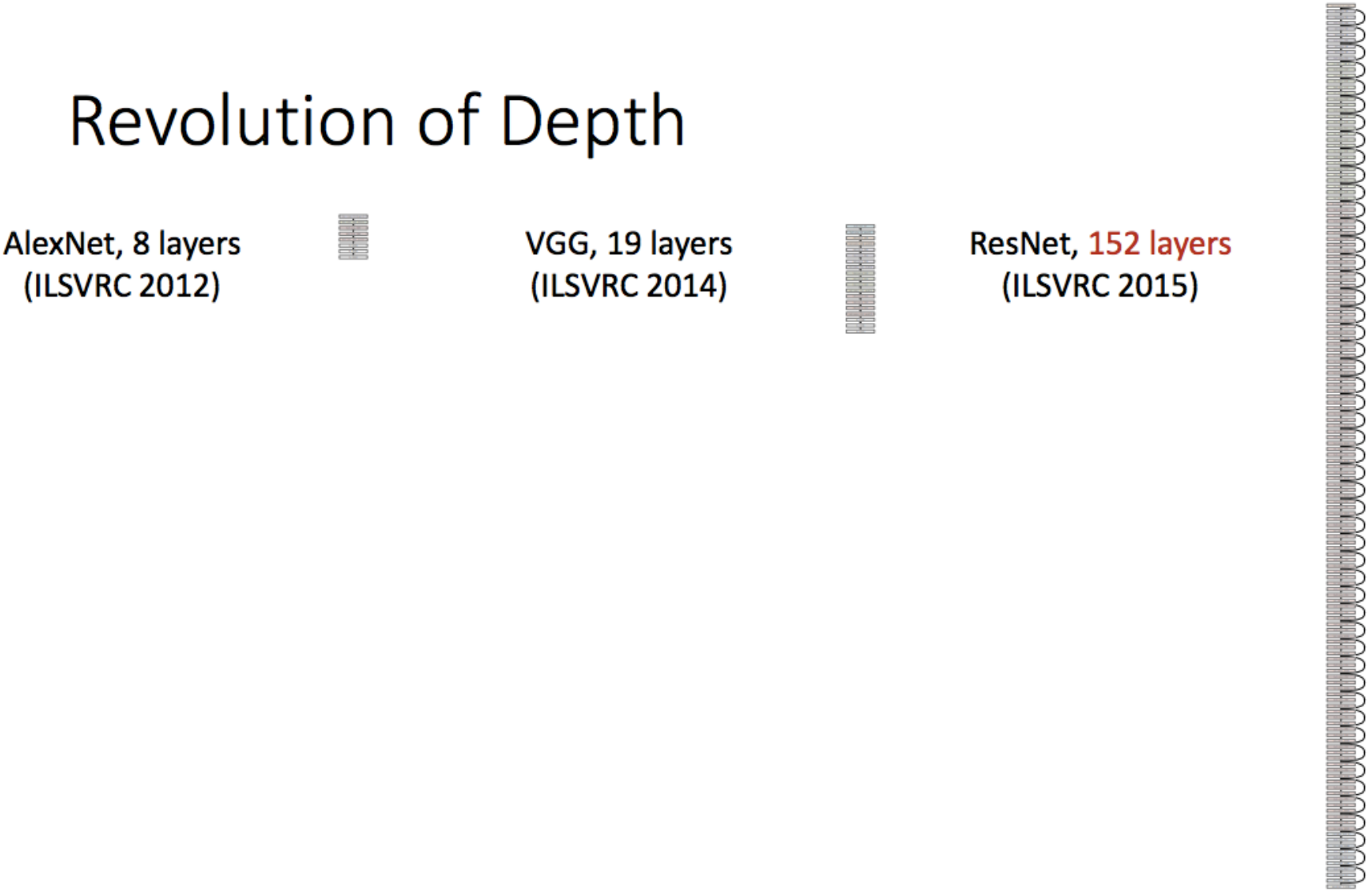
Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016

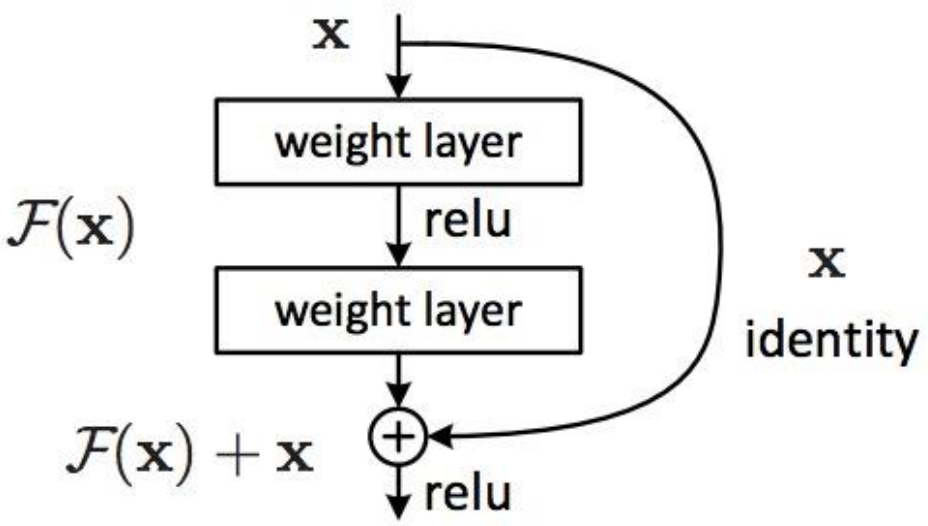
ResNet: ILSVRC 2015 winner



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016

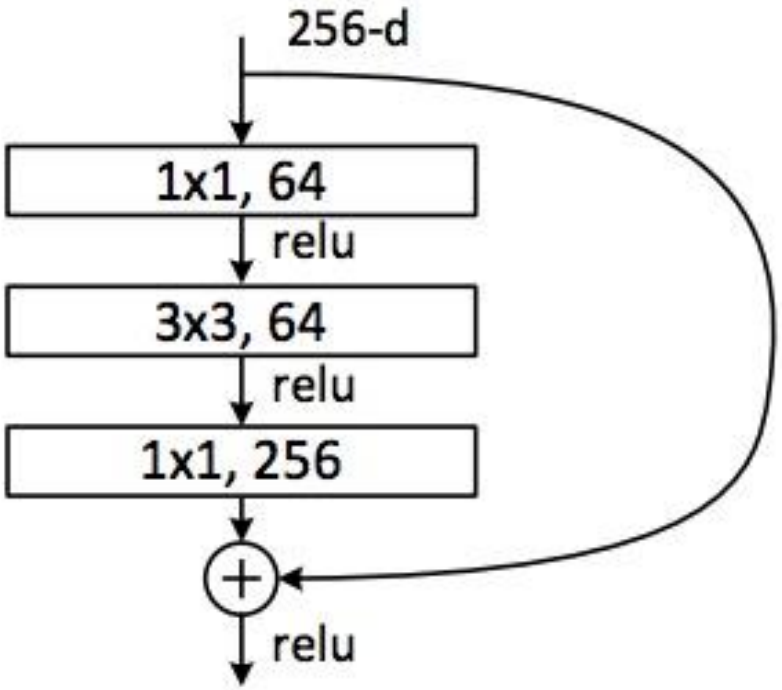
ResNet

- The residual module
  - Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
  - Make it easy for network layers to represent the identity mapping
  - For some reason, need to skip at least two layers



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet



- Directly performing 3x3 convolutions with 256 feature maps at input and output:
  256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

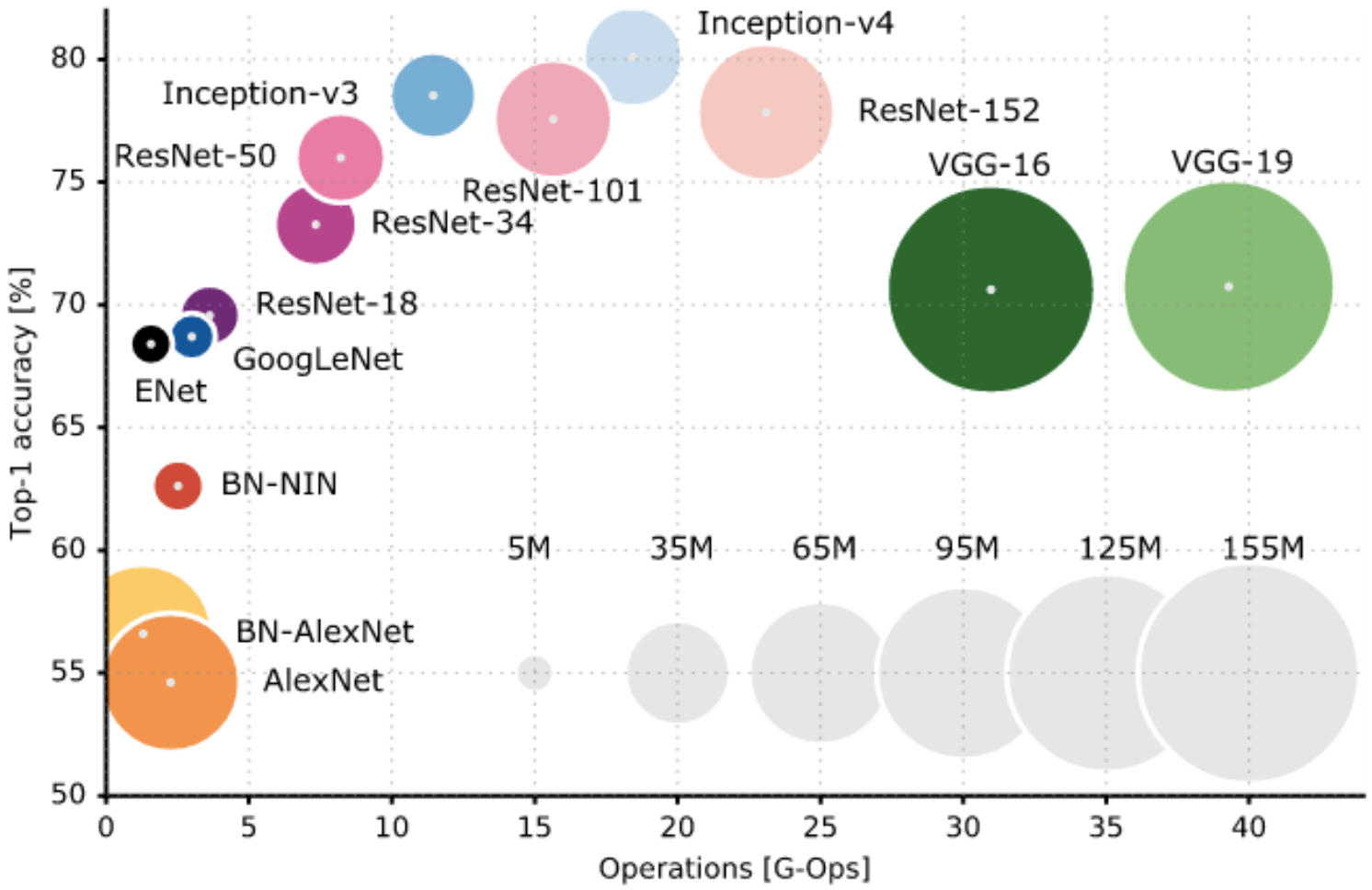
Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

Accuracy vs. efficiency



https://culurciello.github.io/tech/2016/06/04/nets.html

Design principles

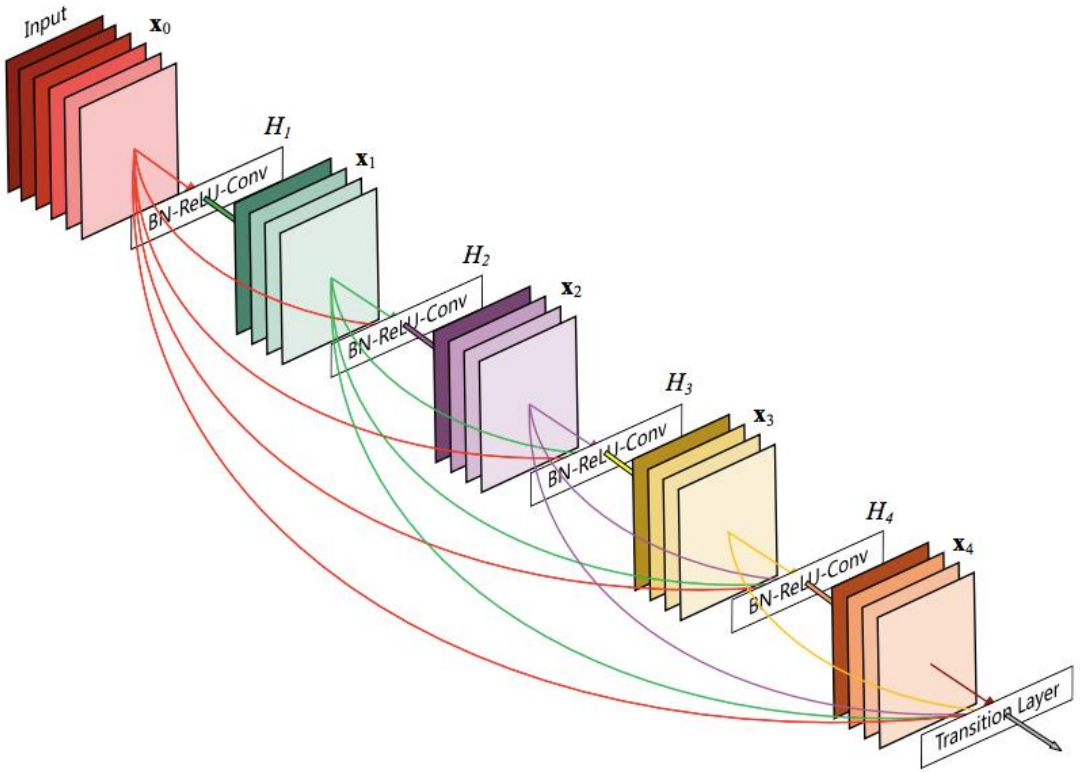
- Reduce filter sizes (except possibly at the lowest layer), factorize filters aggressively
- Use 1x1 convolutions to reduce and expand the number of feature maps judiciously
- Use skip connections and/or create multiple paths through the network

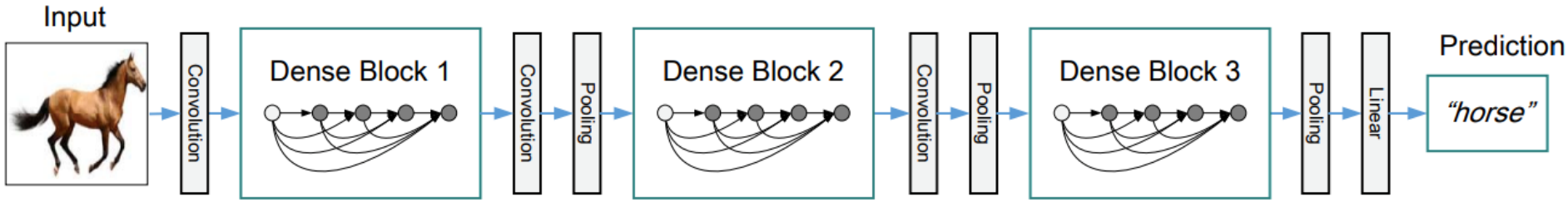
What's missing from the picture?

- Training tricks and details: initialization, regularization, normalization
- Training data augmentation
- Averaging classifier outputs over multiple crops/flips
- Ensembles of networks

- What happened in ILSVRC 2016 and beyond?
  - No more ImageNet classification

DenseNets





G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017 (Best Paper Award)

Summary

- Basic concepts in deep learning
- Deep network structure and learning procedure

- Reference
  - Many online courses: Stanford CS231n, CS230, etc.
  - Deep learning book: http://www.deeplearningbook.org/
  - More PyTorch tutorials: https://pytorch.org/tutorials/

Acknowledgement: Slide materials taken from Zemel et al's CSC411, Bhiksha Raj@CMU, Yisong Yue's Caltech CS155, Hugo Larochelle's and Feifei Li et al's cs231n