BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Learning from Images

Introduction to Convolutional Neural Networks (ConvNet / CNN)

Master DataScience
Winter term 2019/20

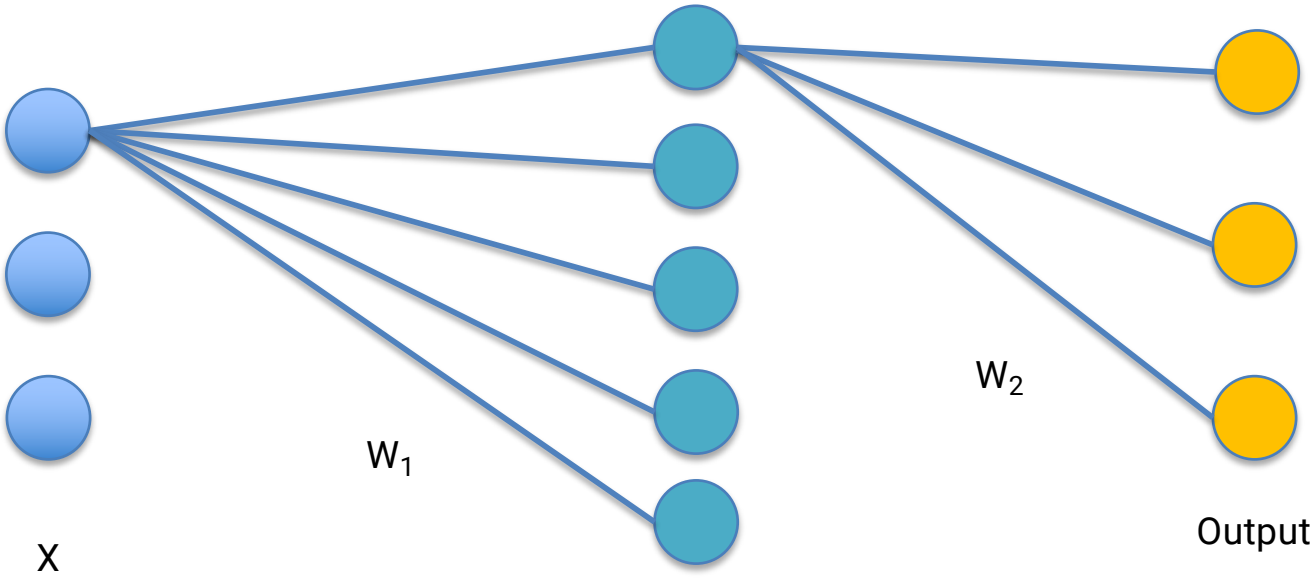
Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de

Today

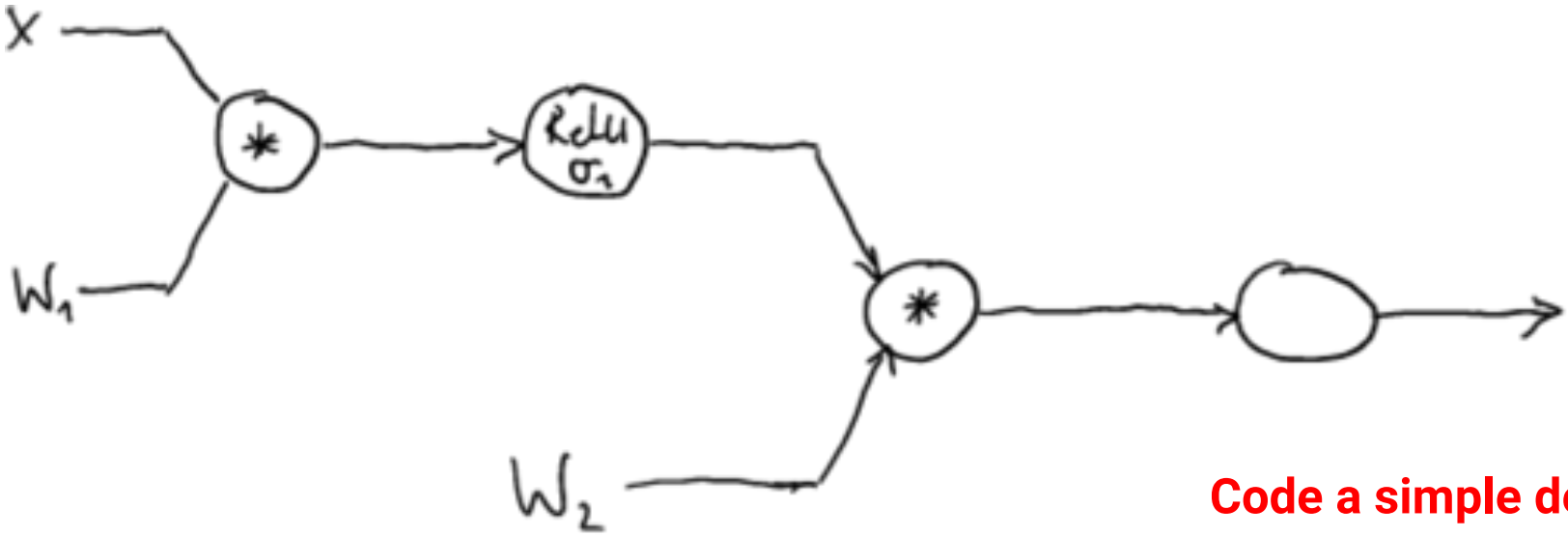
- Neural Network recap
  - Computational graph example
- CNN Intro
  - Idea
  - Network components (ConvLayer, MaxPooling, Dropout, BatchNorm etc.)
  - Architectures
- Project ideas

Neural Network Recap

Two-Layer Neural Network



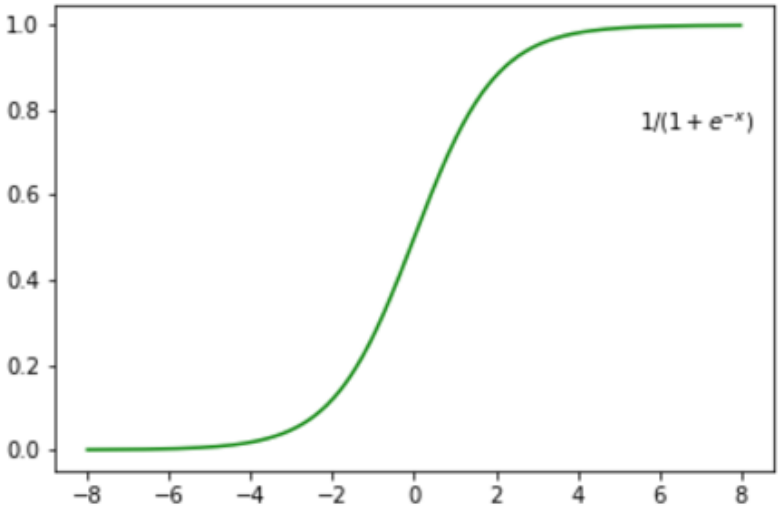
Loss:
$E = (o - y)^2$

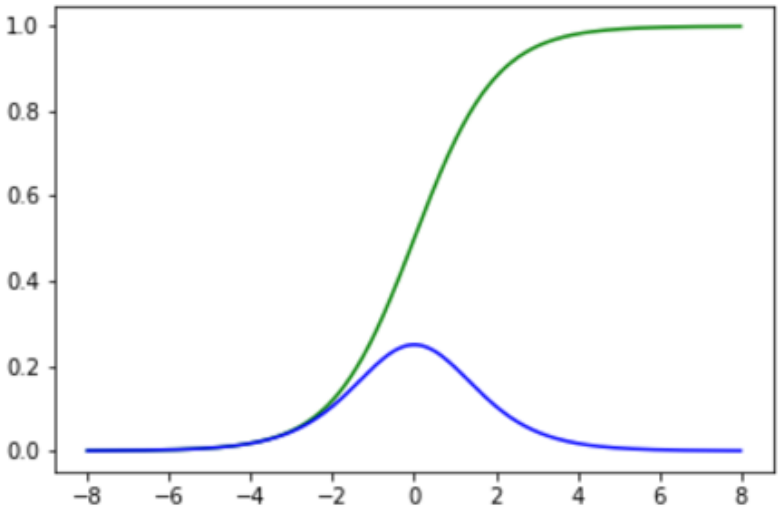


Code a simple demo

Activation functions

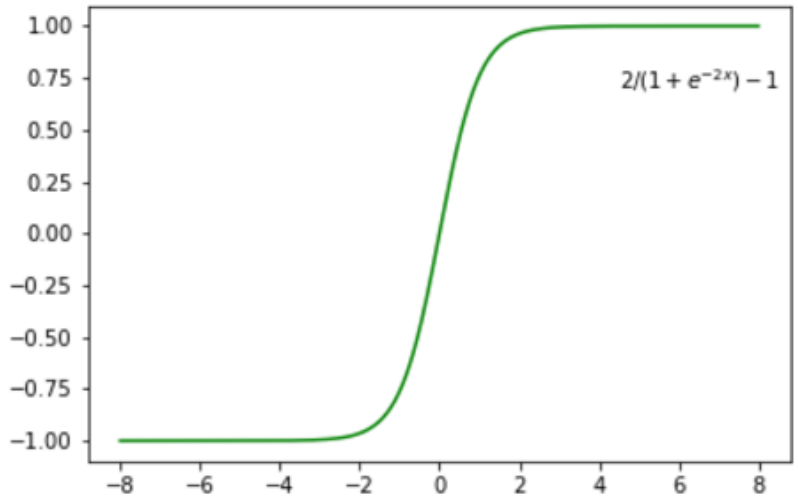
- **Sigmoid / Logistic**
  - Maps values **between [*0,1*]**
  - Used for probabilities
  - Function: ***Differentiable and monotonic***
  - **Derivative is not monotonic**
  - Generalization => ***softmax*** (multiclass problems)

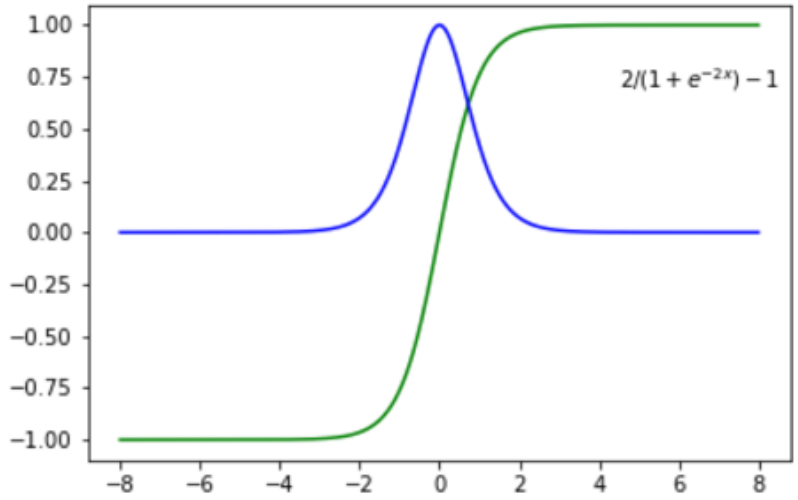




Activation functions

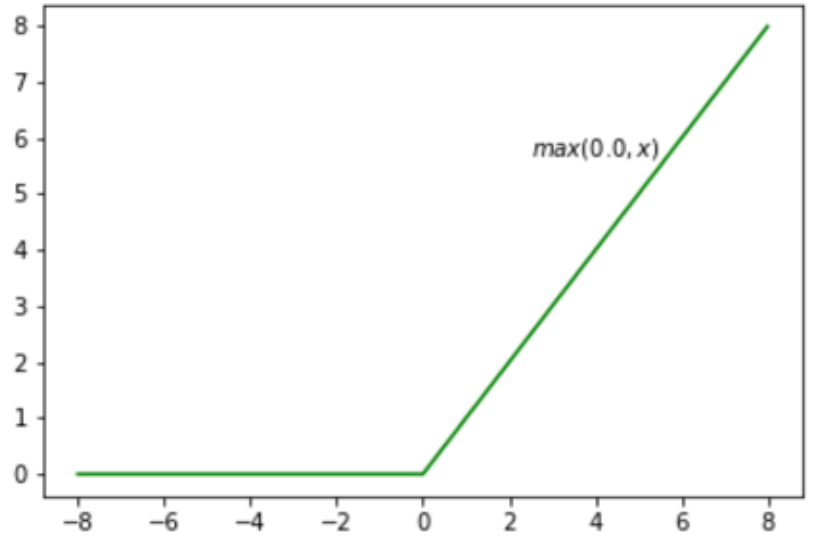
- **Tanh / hyperbolic tangent**
  - Maps between ***[-1,-1]***
  - Used for ***two-class classification***
  - Function: ***Differentiable and monotonic***
  - ***Derivative is not monotonic***

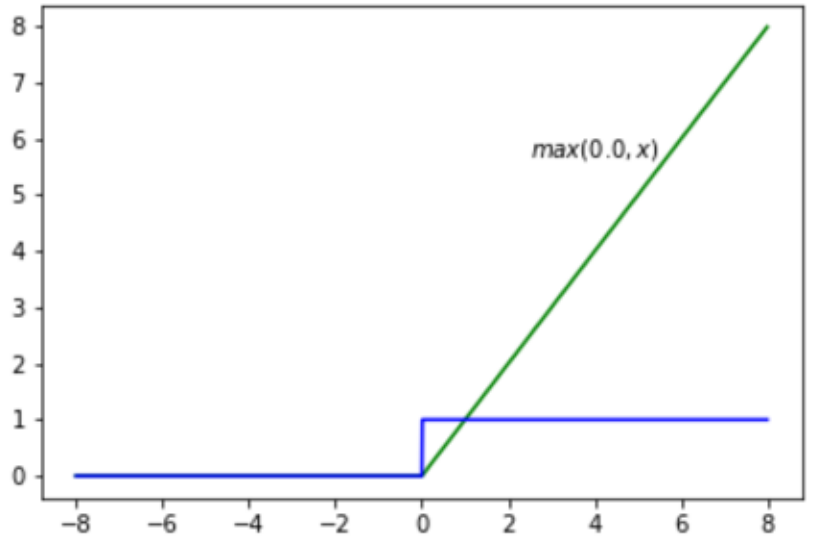




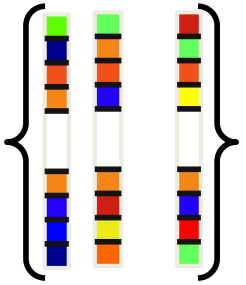
Activation functions

- **ReLU**
  - Maps between ***[0, inf]***
  - Function and derivative are ***differentiable and monotonic***
  - ***Kills gradients < 0***
  - Decreases the ability to fit model
  - Works very well in practice





Feature description



ConvNets

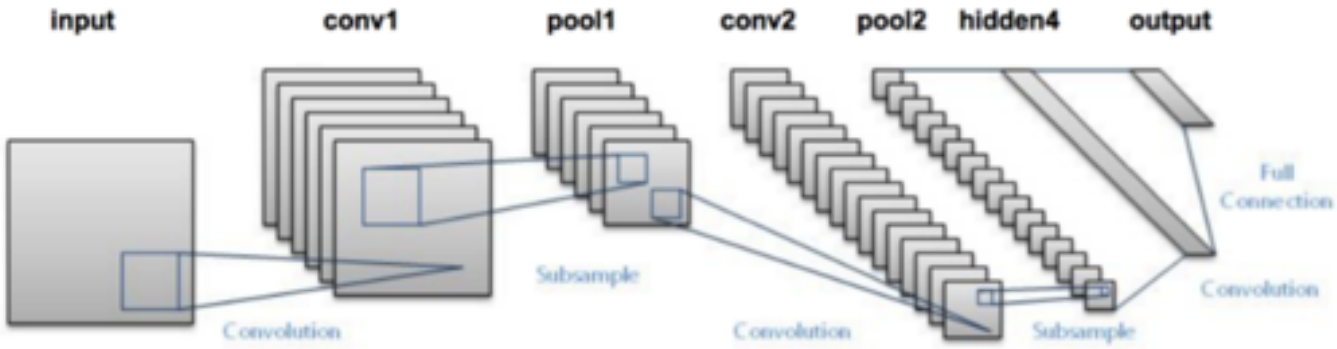
What if?

We could train a ***neural network*** to create image features that we can easily match.

Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (2012)

History

- ConvNets were pioneered by LeCun et al. 1998



- LeNet-5 architecture with 7 layers of convolutional and fully connected units
- Used with MNIST dataset

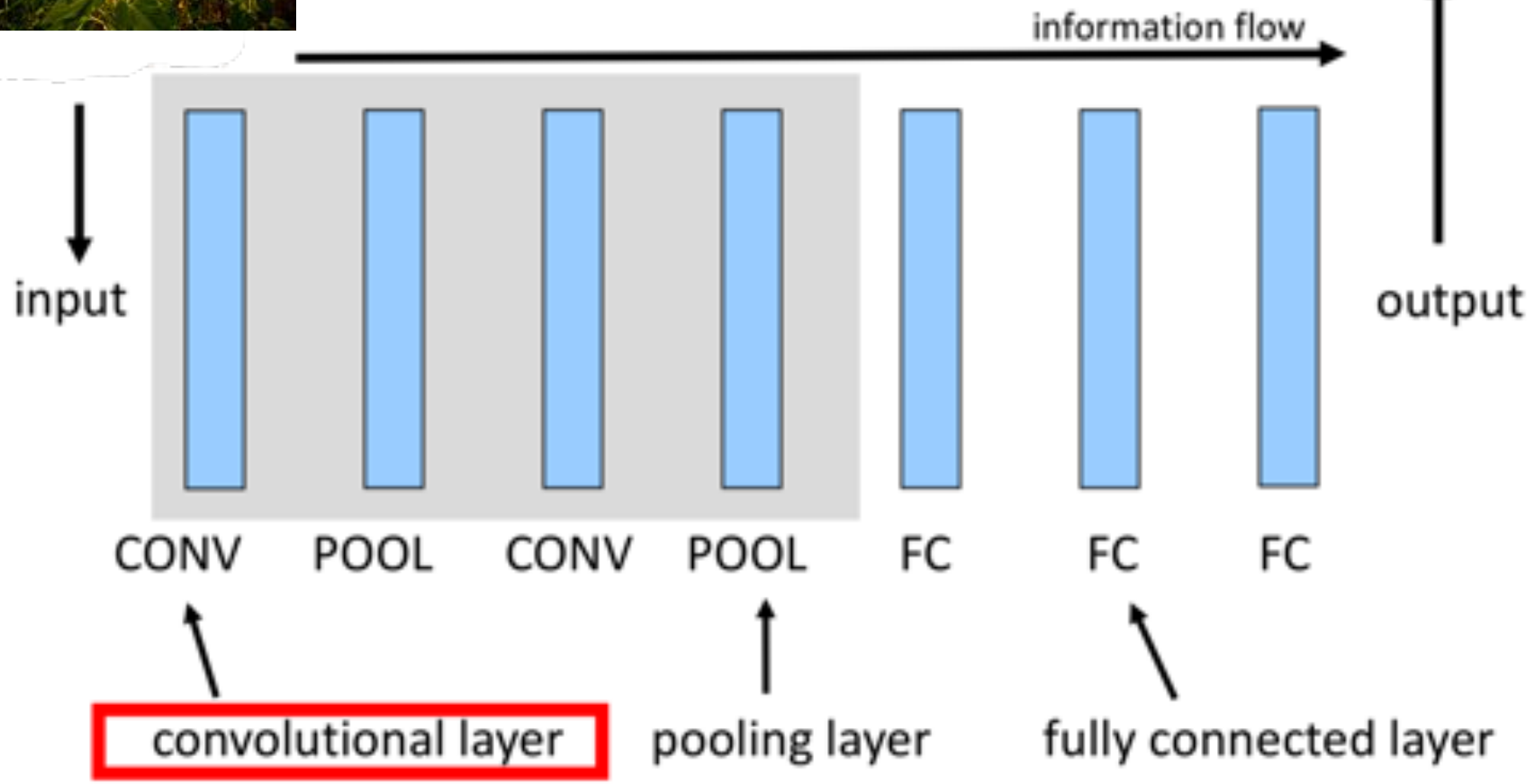
Why use convolutional layers?

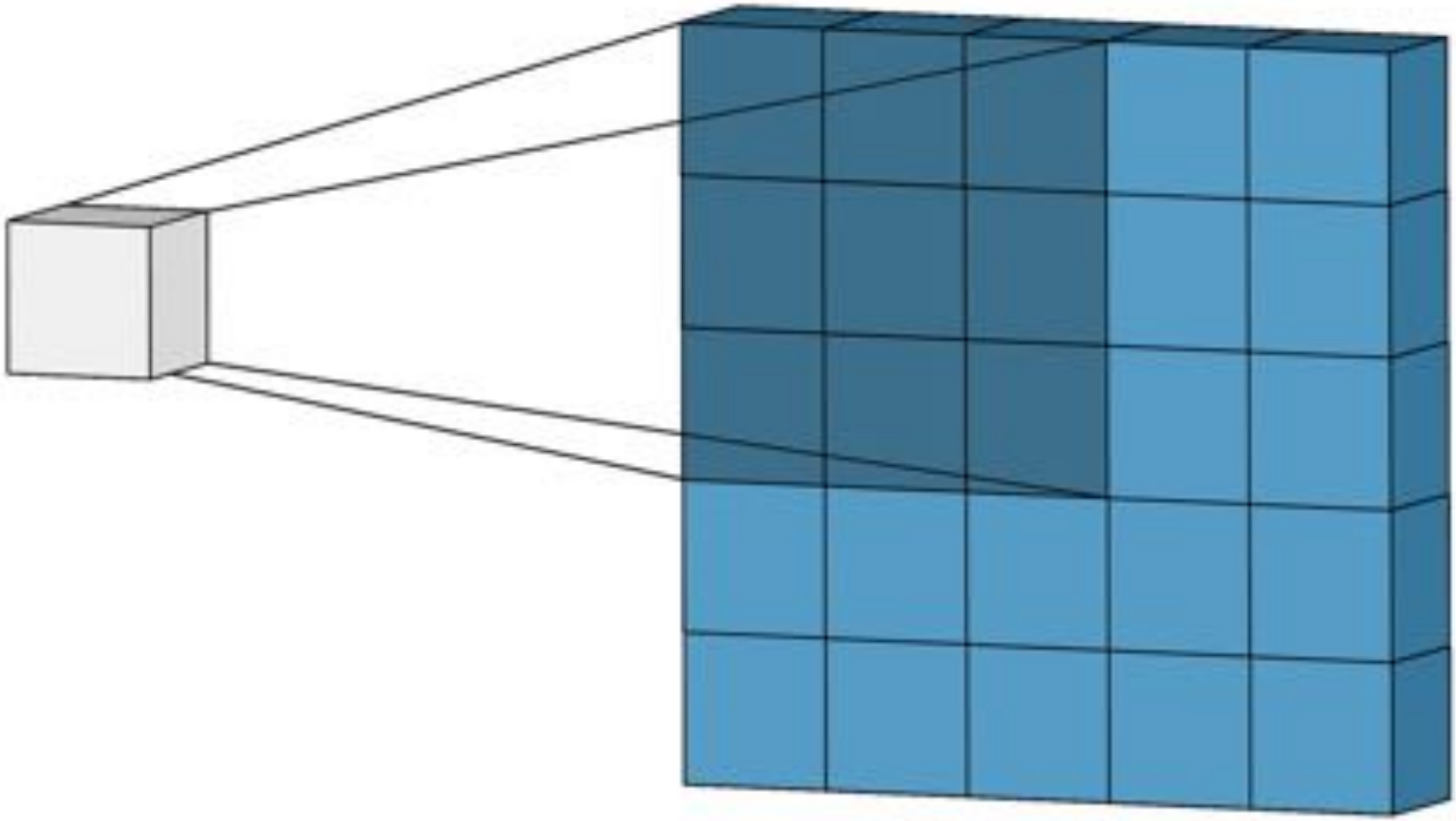
Example Architecture

Number of parameters FC with 5x5 image and 3x3 output features?

225

Sunflower





https://arxiv.org/abs/1603.07285

Discrete convolution is sparse and reuses parameters by applying the same weights to multiple locations.

https://arxiv.org/abs/1603.07285

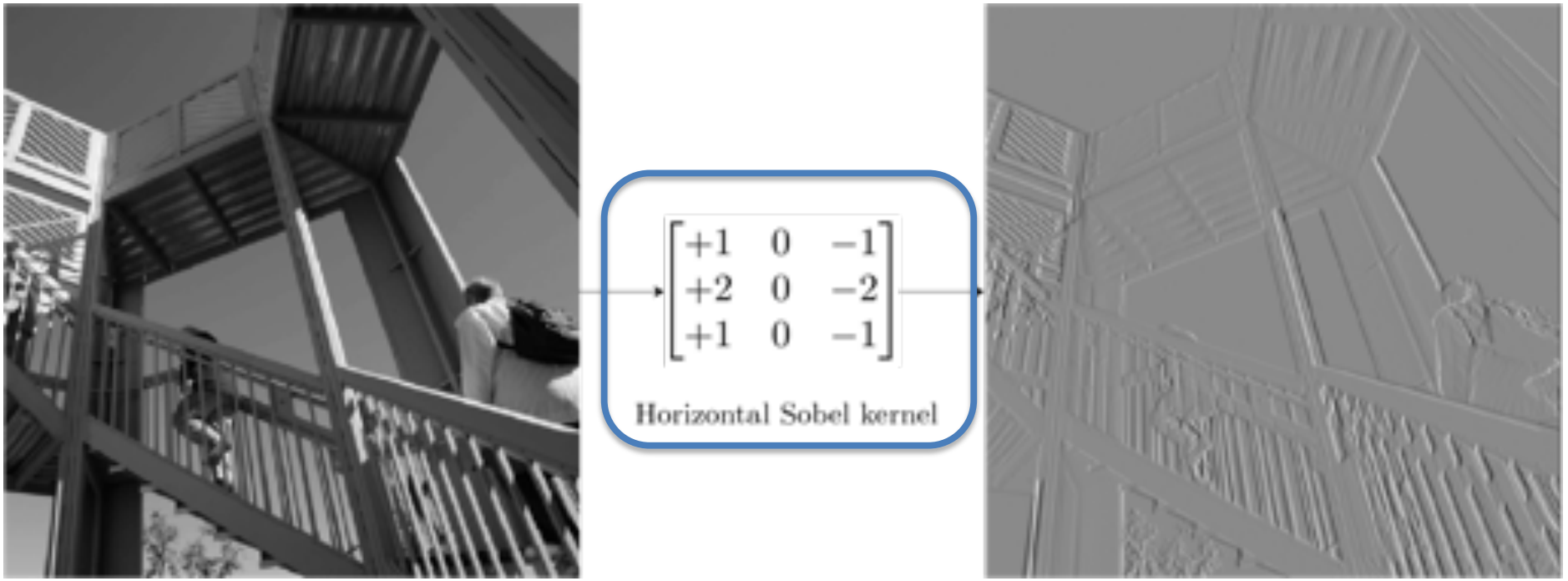
Number of parameter convolutions?

9

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

Andrew Ng

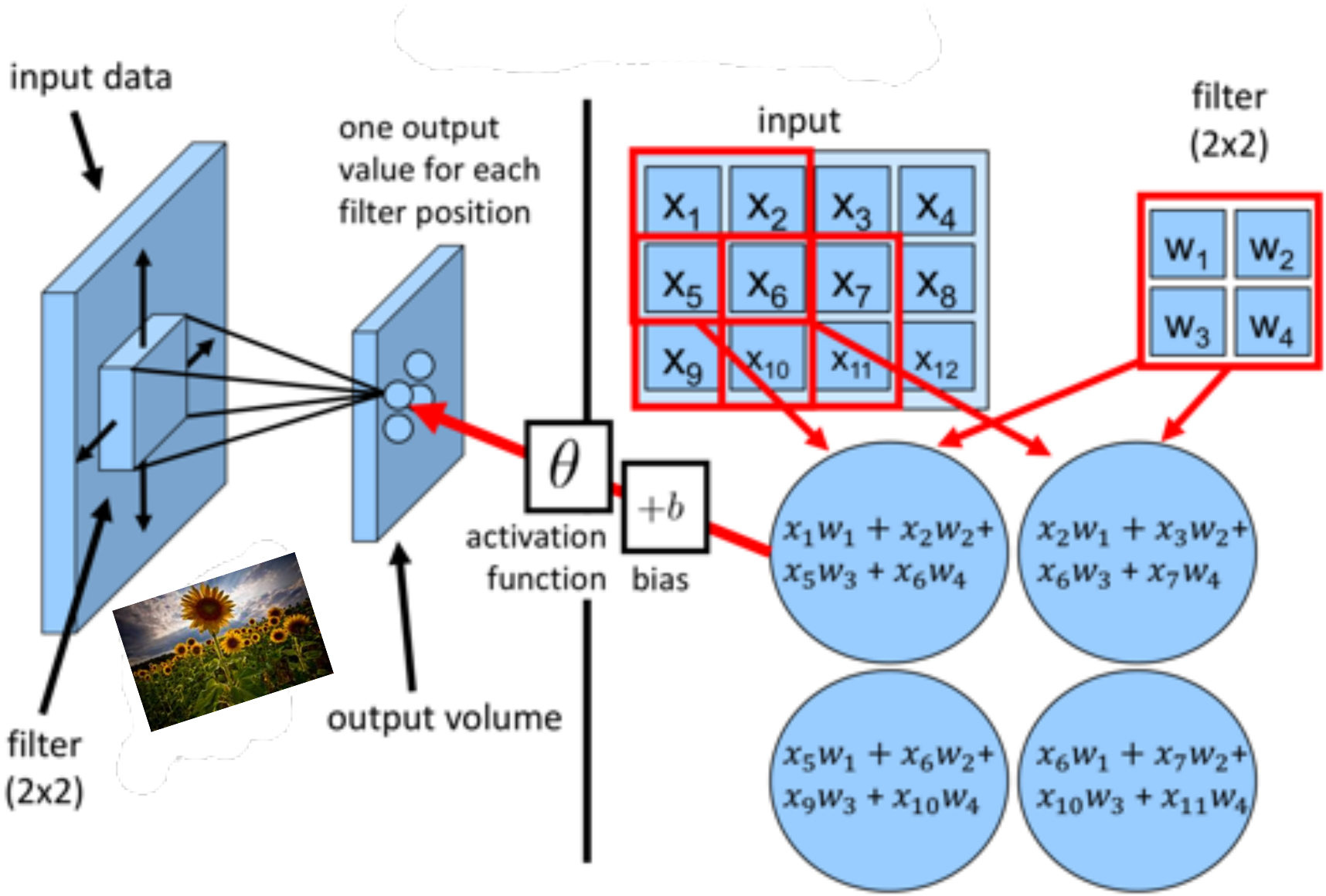
2D Convolution - Intuition



In ConvNets we learn the values of the kernel matrix!

Convolutional Layer - Intuition

Lets code this!



2D Convolution – Intuition / Comparison to Neural Networks

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} & w_{1,7} & w_{1,8} & w_{1,9} & w_{1,10} & w_{1,11} & w_{1,12} & w_{1,13} & w_{1,14} & w_{1,15} & w_{1,16} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} & w_{2,7} & w_{2,8} & w_{2,9} & w_{2,10} & w_{2,11} & w_{2,12} & w_{2,13} & w_{2,14} & w_{2,15} & w_{2,16} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} & w_{3,7} & w_{3,8} & w_{3,9} & w_{3,10} & w_{3,11} & w_{3,12} & w_{3,13} & w_{3,14} & w_{3,15} & w_{3,16} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & w_{4,5} & w_{4,6} & w_{4,7} & w_{4,8} & w_{4,9} & w_{4,10} & w_{4,11} & w_{4,12} & w_{4,13} & w_{4,14} & w_{4,15} & w_{4,16} \end{bmatrix}$$

$$\begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

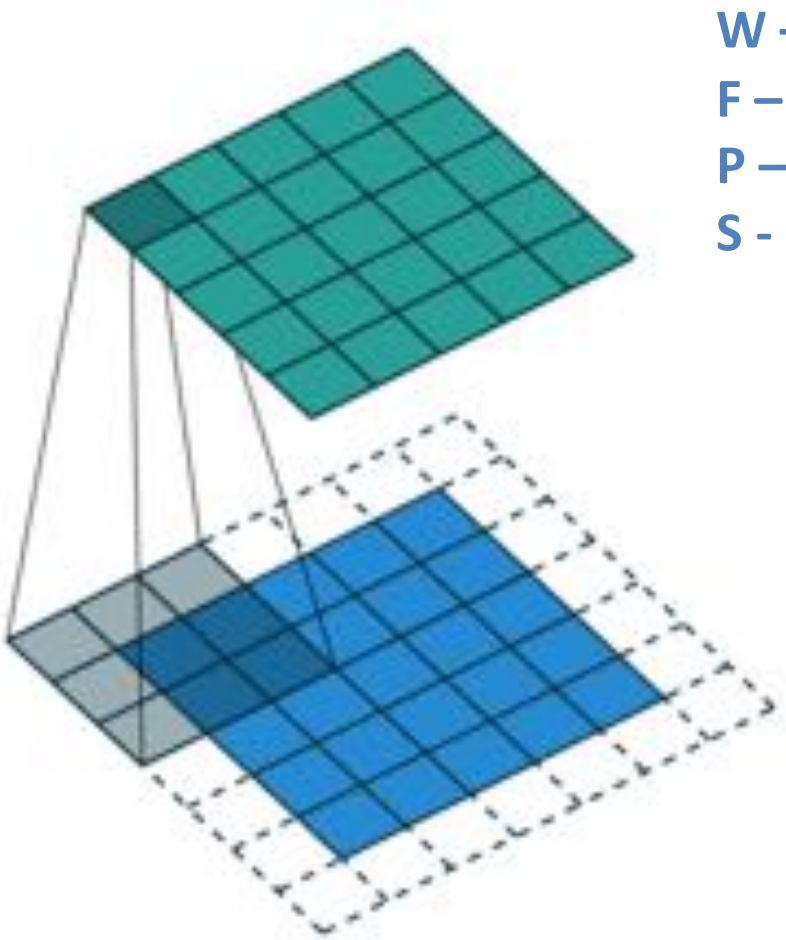
https://arxiv.org/abs/1603.07285

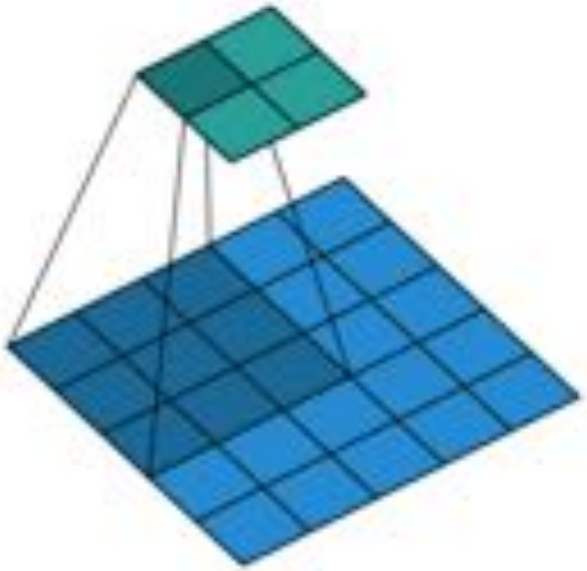
Padding

Output = (W−F+2P)/S+1

Stride

W – Window
F – Filter
P – Padding
S - Stride

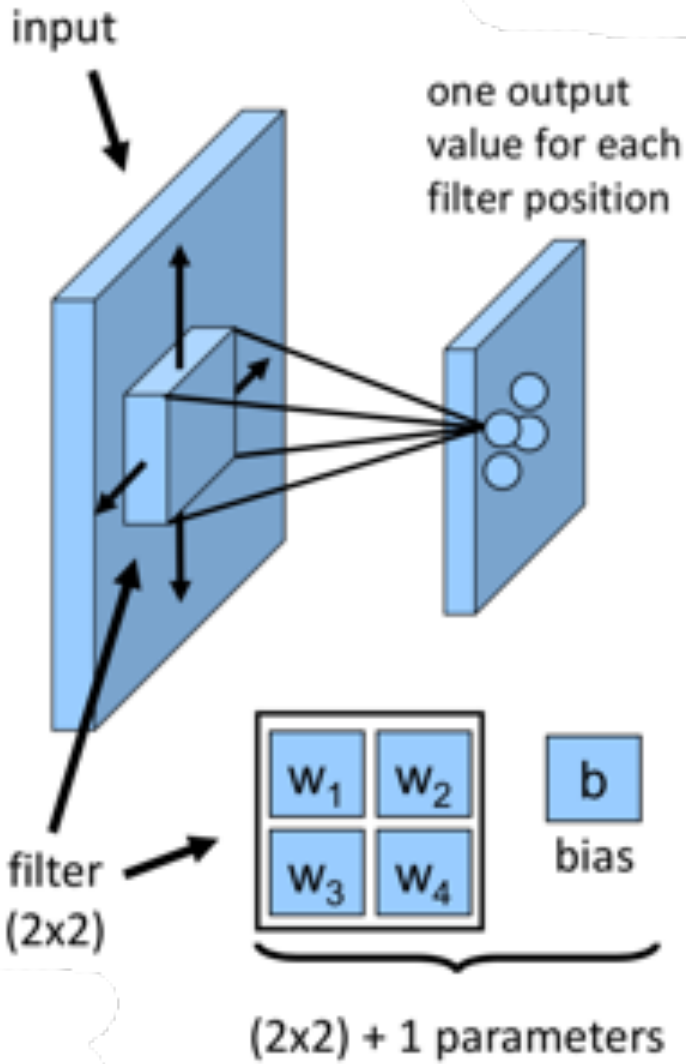




https://ezyang.github.io/convolution-visualizer/index.html

https://arxiv.org/abs/1603.07285

Convolutional Layer - Summary



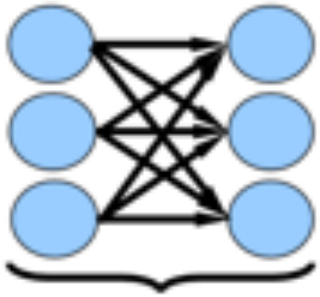
In a convolutional layer, all neurons

→ use the same weights (=filter) (*parameter sharing*)

→ receive only **local** information from previous layer

Shifted input creates shifted output

Compare with fully connected layer:



Each Neuron

→ receives all information from previous layer

→ use their own weights (no parameter sharing)

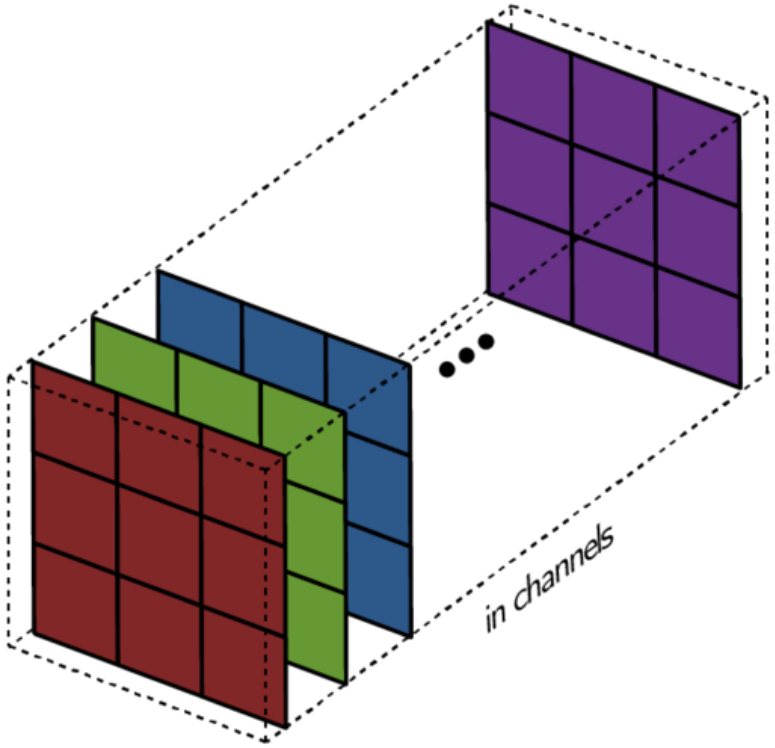
(3x3) + 3 parameters

Multi-channel version

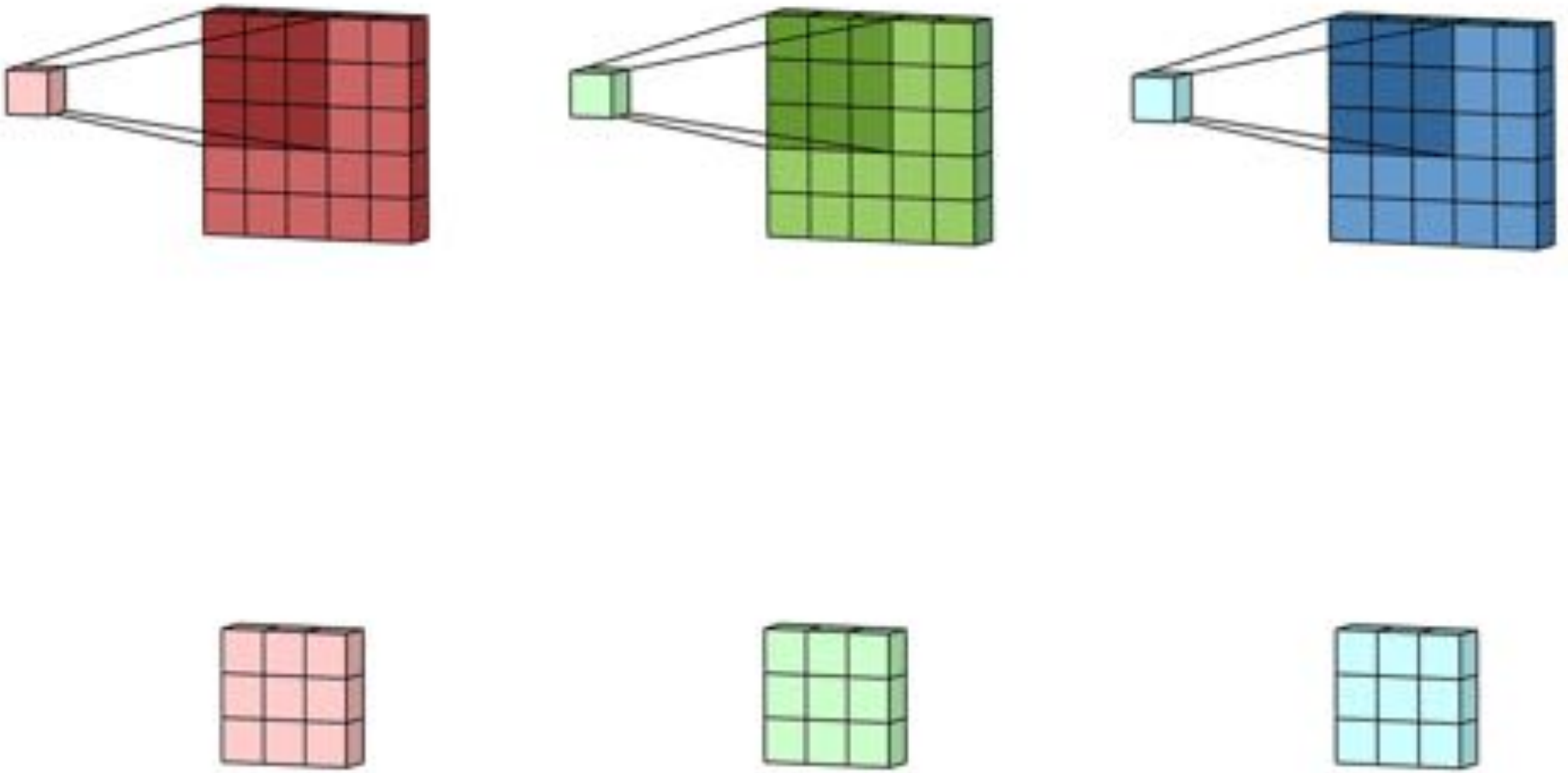
Red

Green

Blue



https://arxiv.org/abs/1603.07285



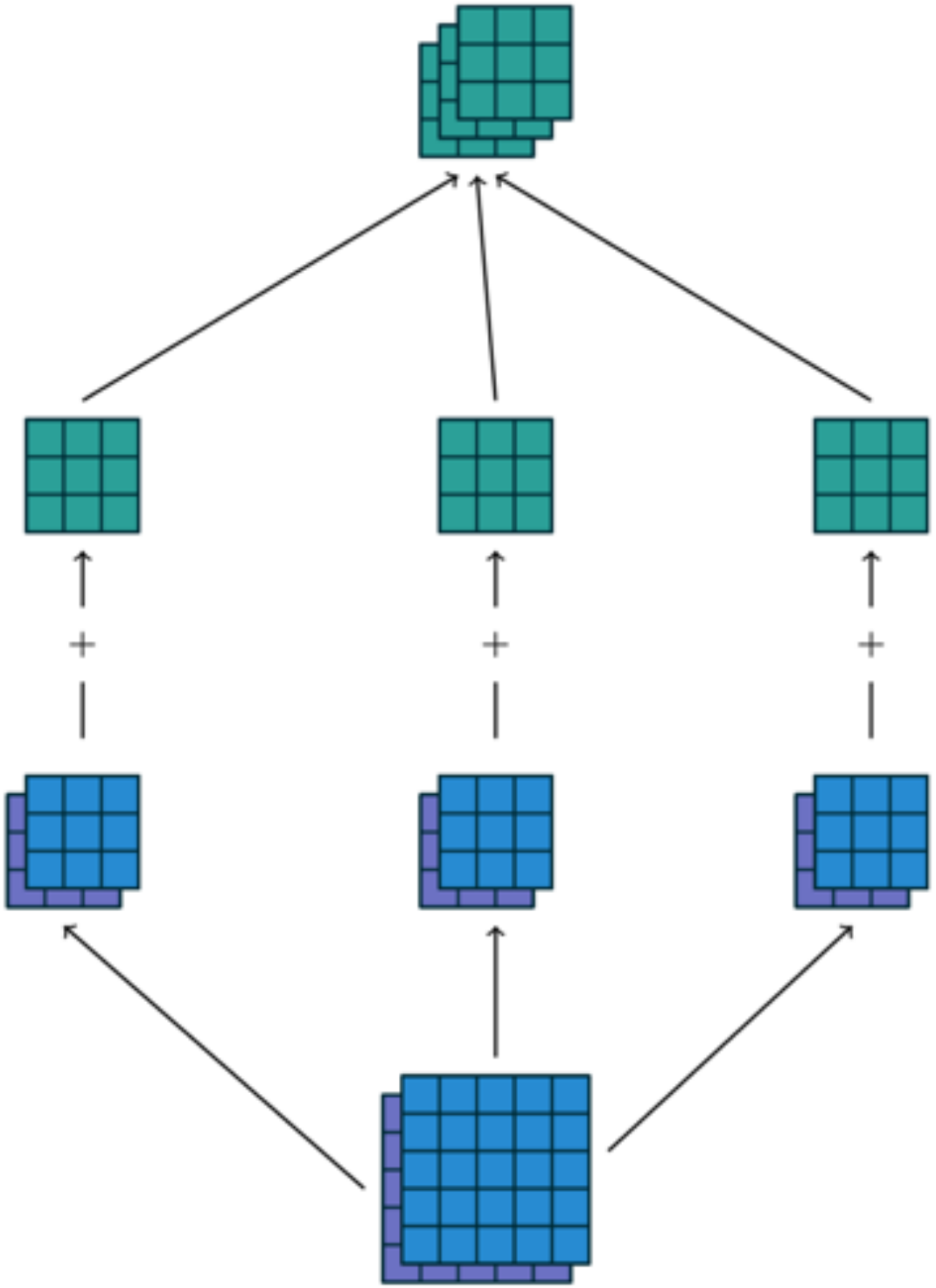
https://arxiv.org/abs/1603.07285




https://arxiv.org/abs/1603.07285

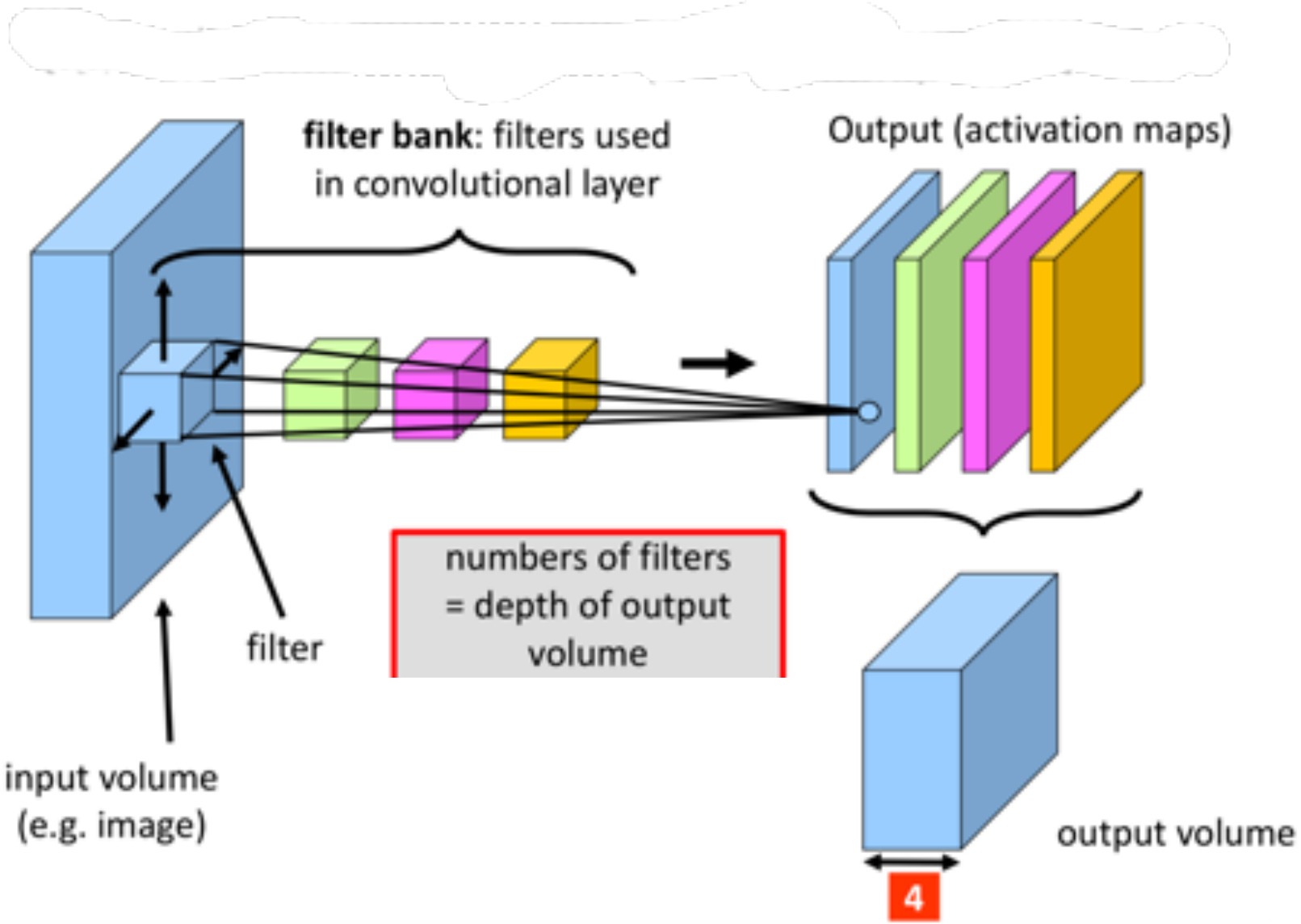
Outputs are summed

2x3 kernel produce one output for each input map



Input maps (2 Channels)

Convolutional Layer – Depth of Output Volume



Pooling Layer

- replaces regions of input with summary statistics (e.g. maximum value)
- shrinks spatial dimensions of the input

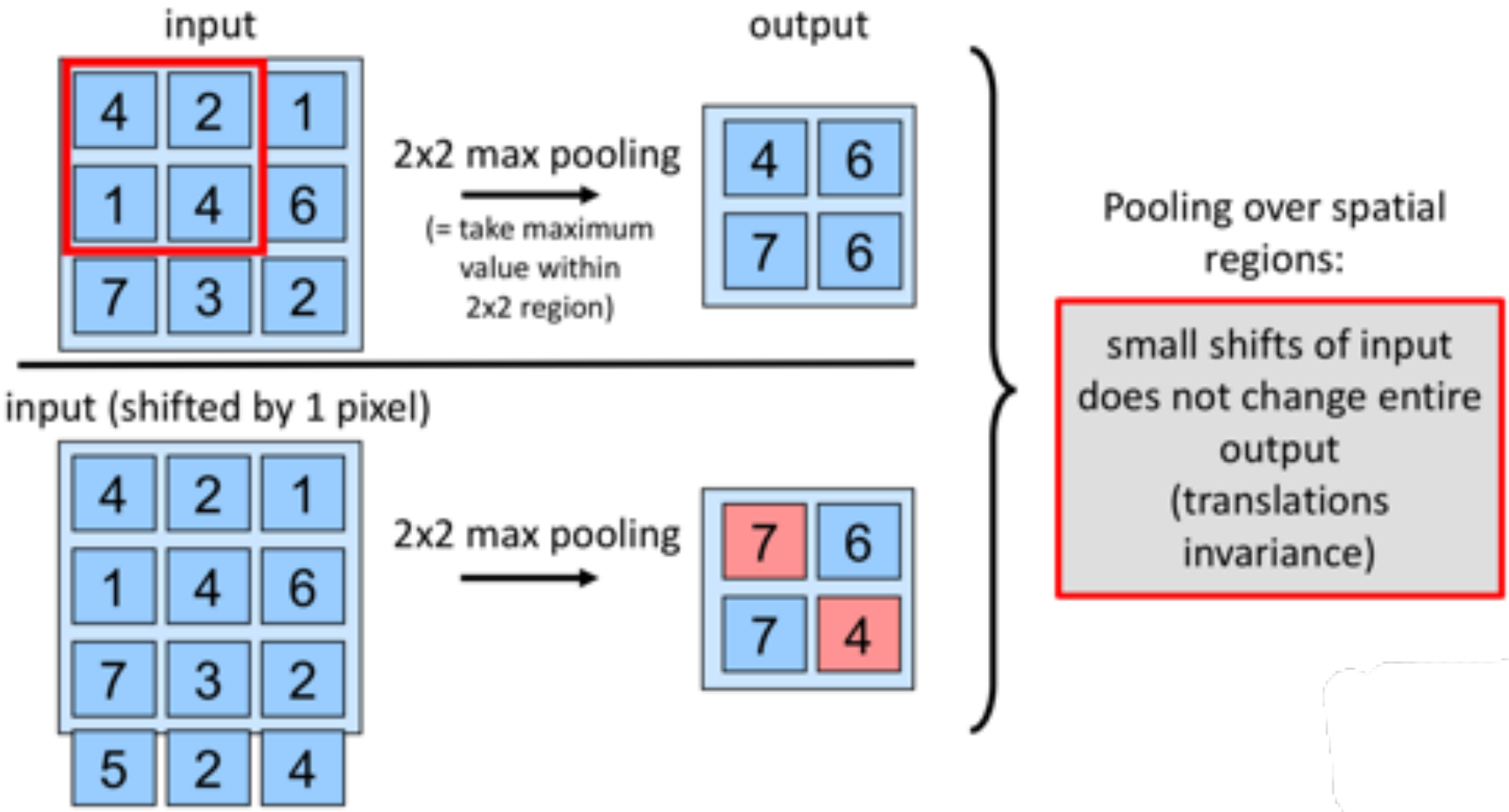
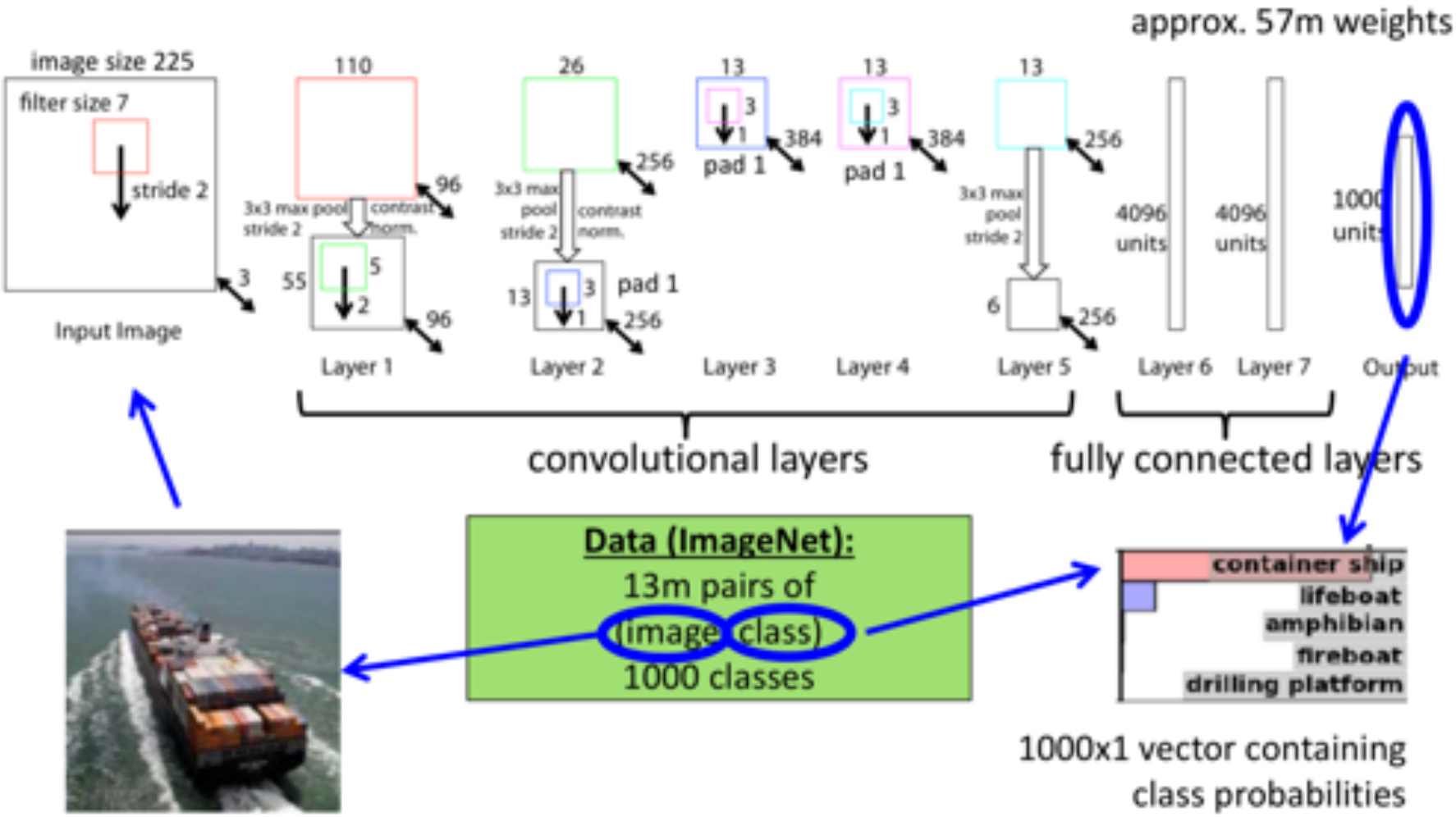
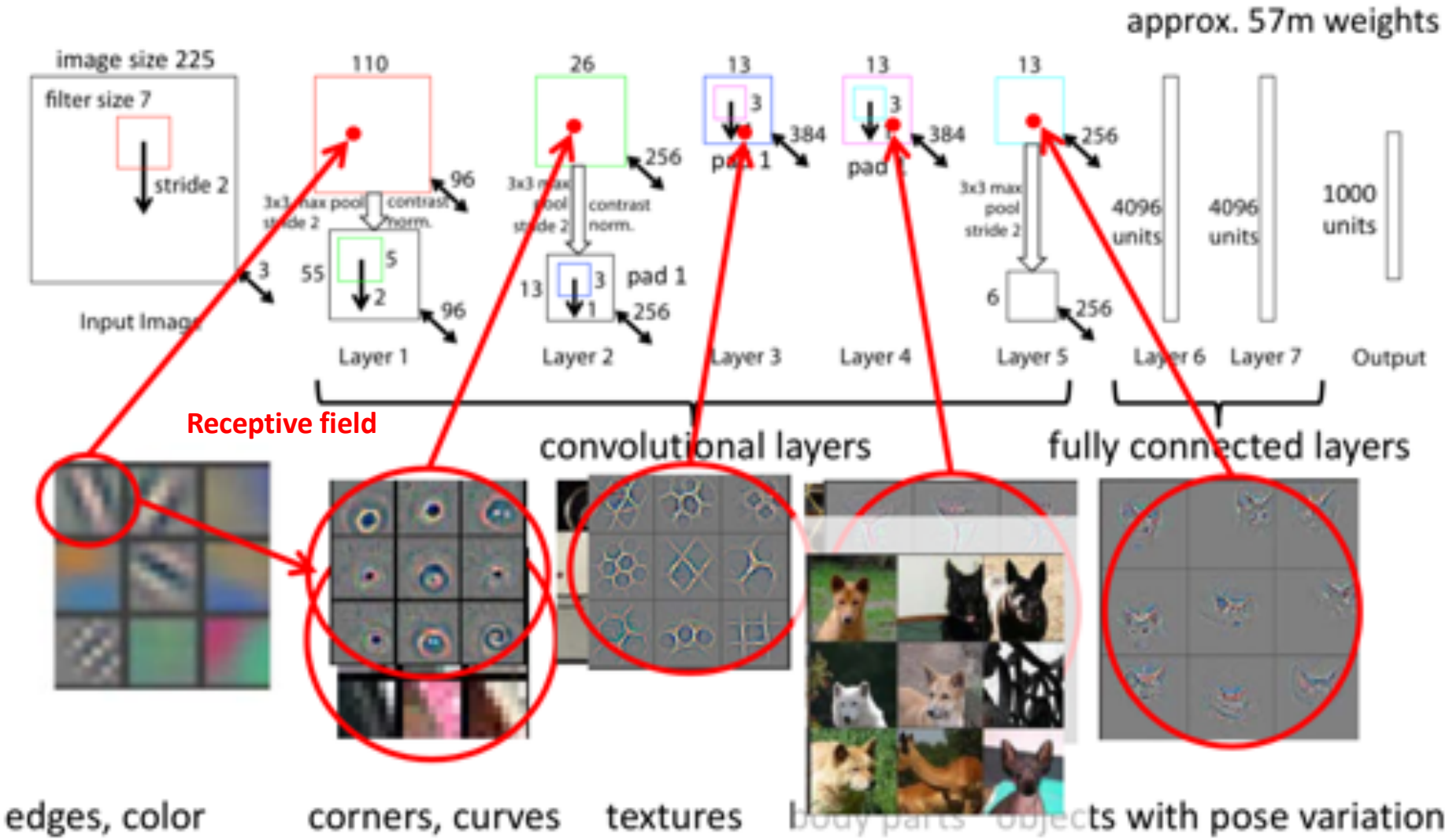


Image Classification with Deep Learning

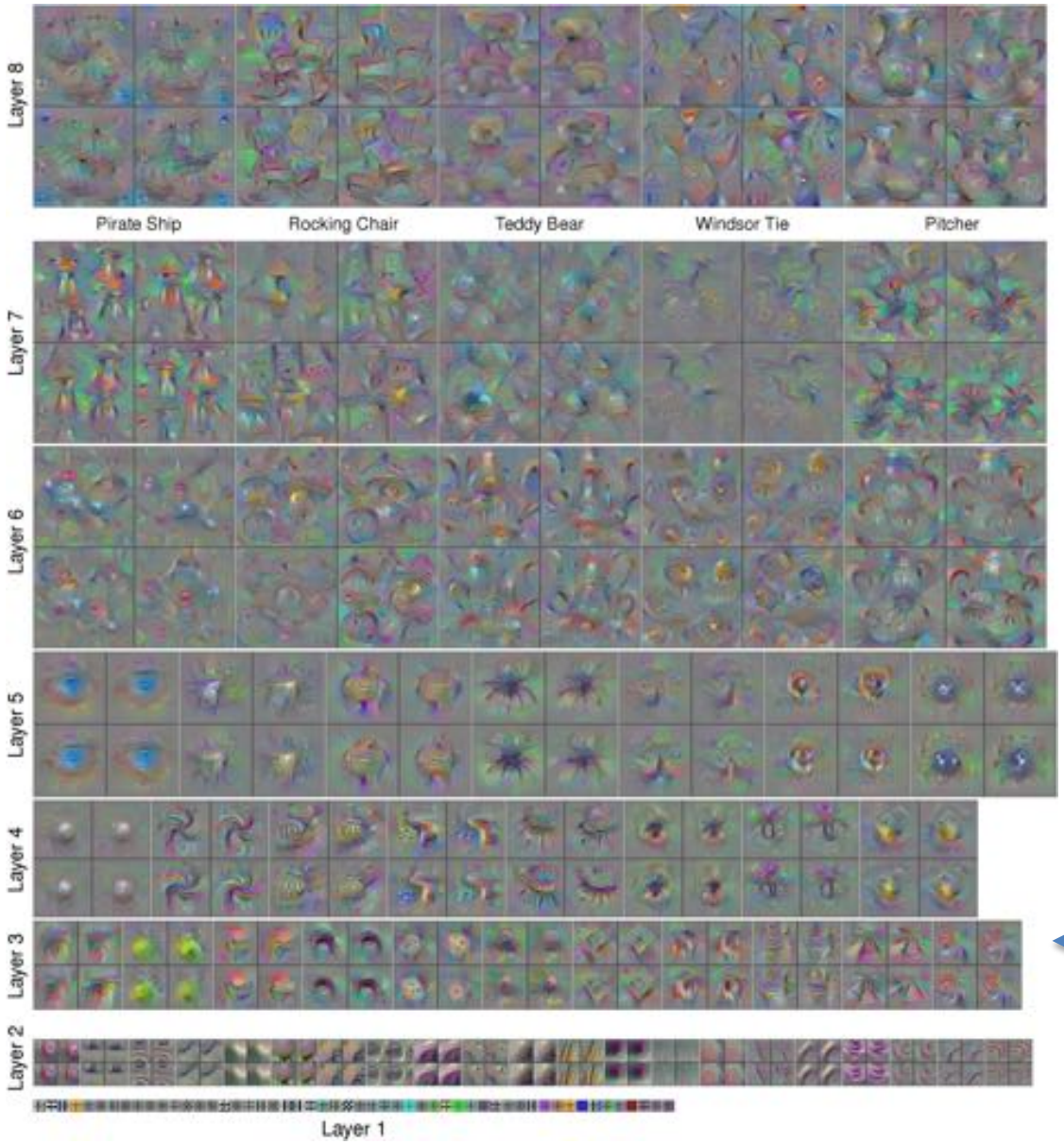


Feature Visualization

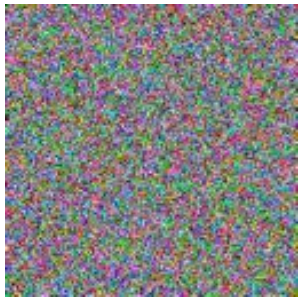


http://yosinski.com/deepvis#toolbox

Feature Visualization



Start



$$x \leftarrow x + \alpha \cdot \partial a_i(x)/\partial x$$

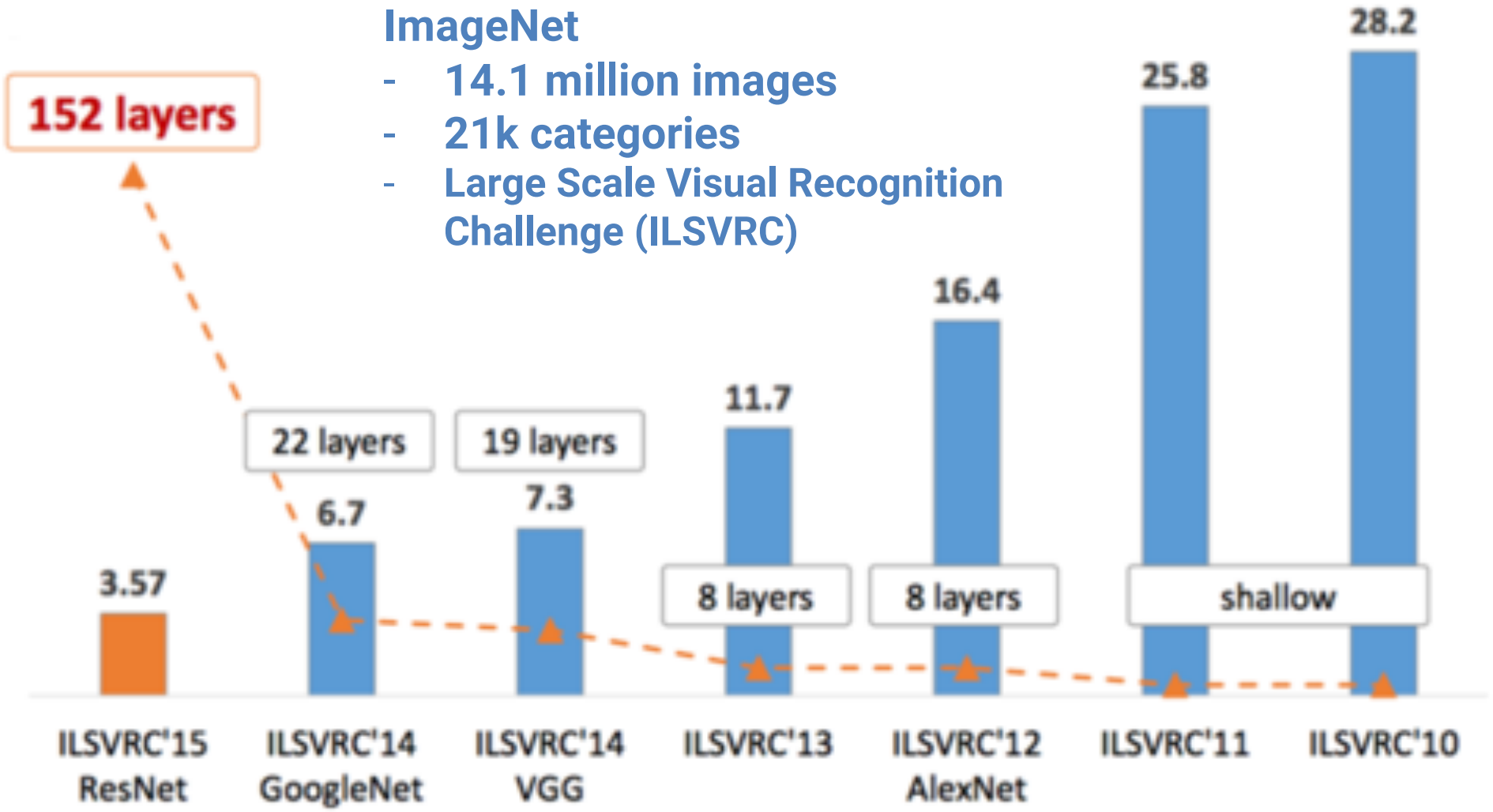
Repeat until image with high activation of the neuron

Image classification
ConvNet Architectures

ConvNet Architectures

Standardized Image Database ImageNet

- 14.1 million images
- 21k categories
- Large Scale Visual Recognition Challenge (ILSVRC)

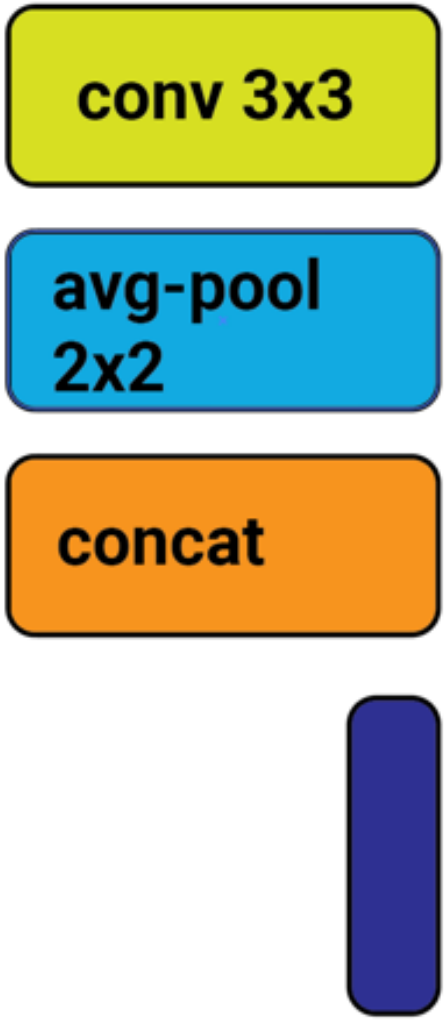


https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

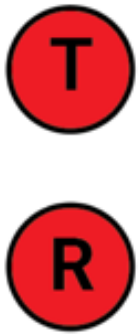
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 MB	0.713	0.901	138,357,544	23
InceptionV3	92 MB	0.779	0.937	23,851,784	159
ResNet50	98 MB	0.749	0.921	25,636,712	-
Xception	88 MB	0.790	0.945	22,910,480	126
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
ResNeXt50	96 MB	0.777	0.938	25,097,128	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Layers

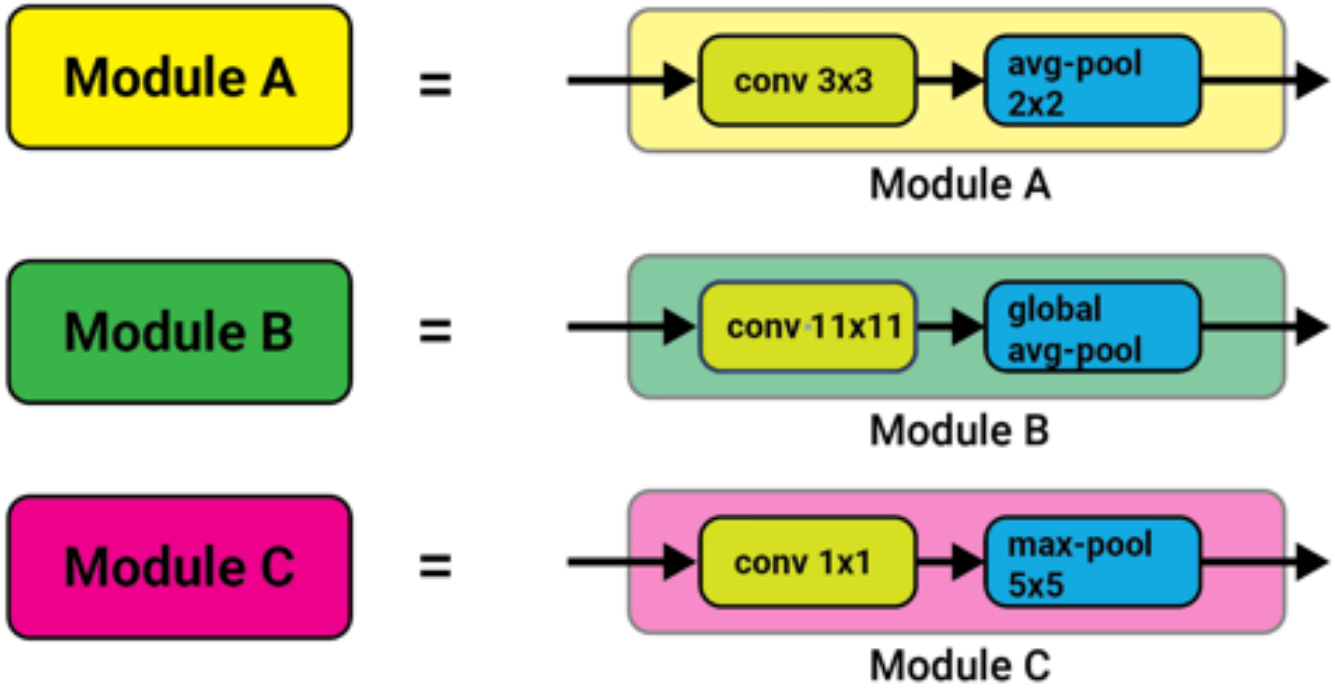


Activation Functions

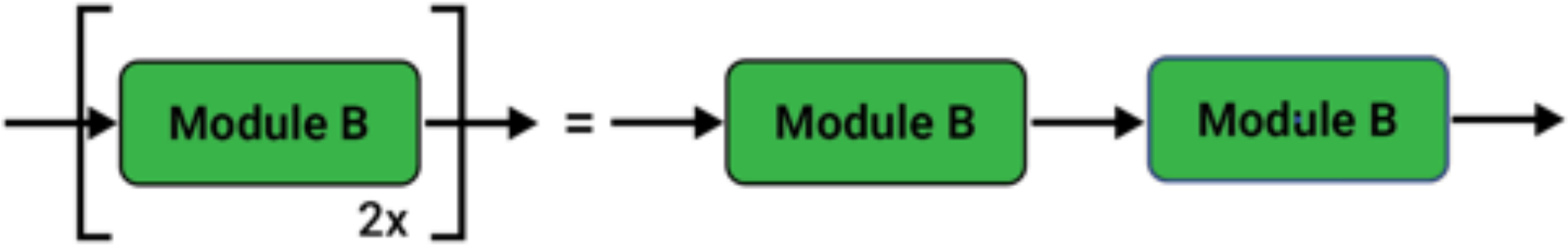


Other Functions

Modules / Blocks



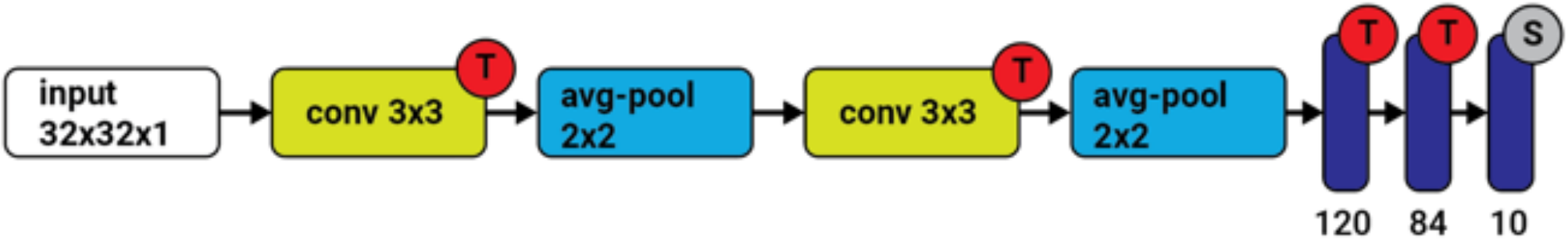
Repeated Layers



LeNet-5 (1998)

- **Paper:**
  LeCun et al. :Gradient-Based Learning Applied to Document Recognition

- **Novelty:**
  - Architecture has become the standard ‘template’
  - Stacking convolutions and pooling layers, and ending the network with one or more fully-connected layers.



AlexNet (2012)

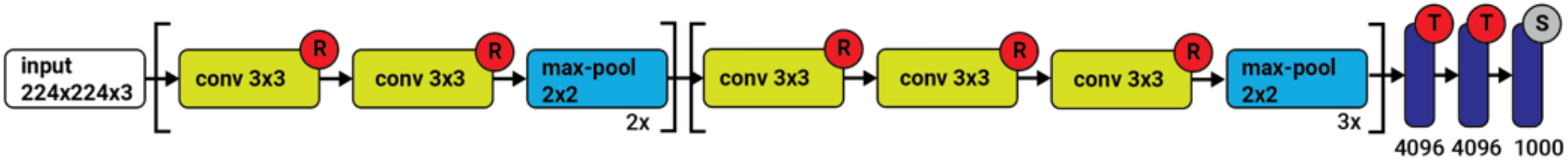
- **Paper:**
  Krizhevsky et al.: ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS 2012

- **Novelty:**
  - First to implement Rectified Linear Units (ReLUs) as activation functions
  - **AlexNet** significantly outperformed all the prior classifiers
  - won challenge by reducing the top-5 error from 26% to 15.3%

VGG-16 (2014)

- **Paper:**
  Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition

- **Novelty:**
  - Contribution from this paper is the design of *deeper* networks (roughly twice as deep as AlexNet)
  - 13 convolutional and 3 fully-connected layers, carrying with them the ReLU tradition from AlexNet (**138M parameters)**

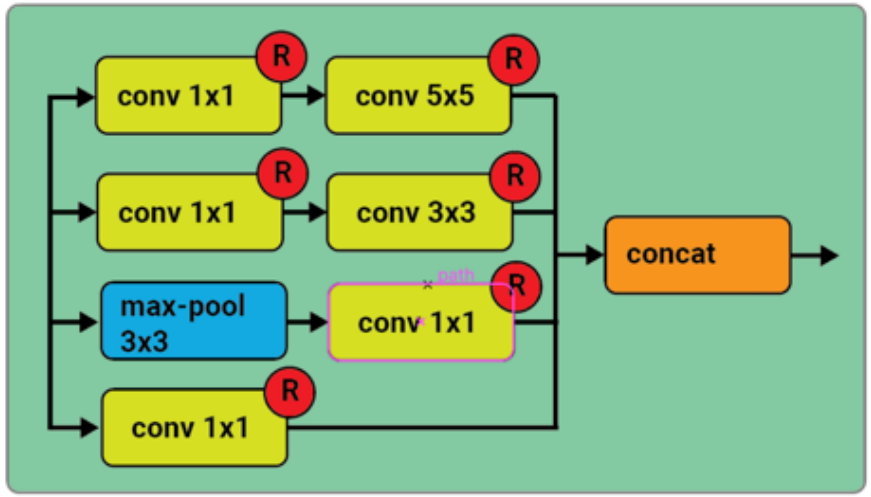


Inception v1 (2014)

- **Paper:**
  Szegedy et. al: Going Deeper with Convolutions. CVPR 2015

- **Novelty / Idea:**
  - Building networks using dense modules/blocks instead of stacking convolutional layers
  - Parallel streams of convolutions followed by concatenation (cluster 1x1, 3x3, 5x5 features)
  - 1x1 convolutions for dimensionalty reduction

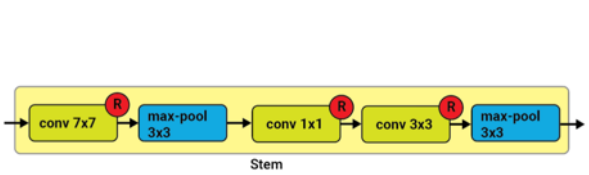
Stem

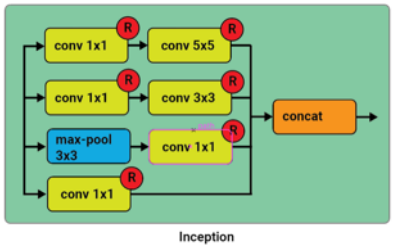


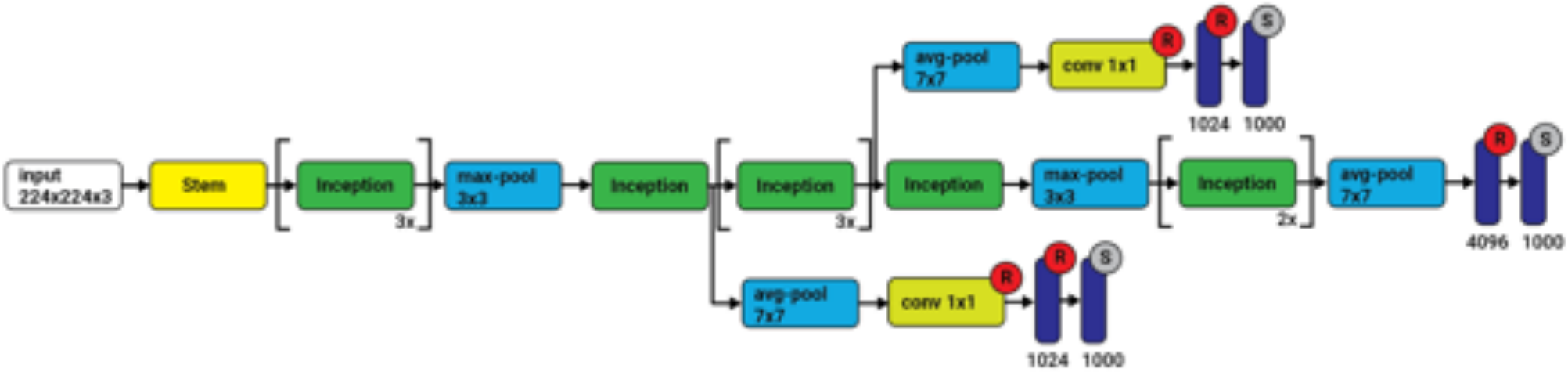
Inception

Inception v1 (2014)

- **Novelty / Idea:**
  - two auxiliary classifiers to encourage discrimination in the lower stages of the classifier -> this helps increasing the gradient signal for the backprop step and additional regularisation
  - auxiliary networks are discarded at inference time

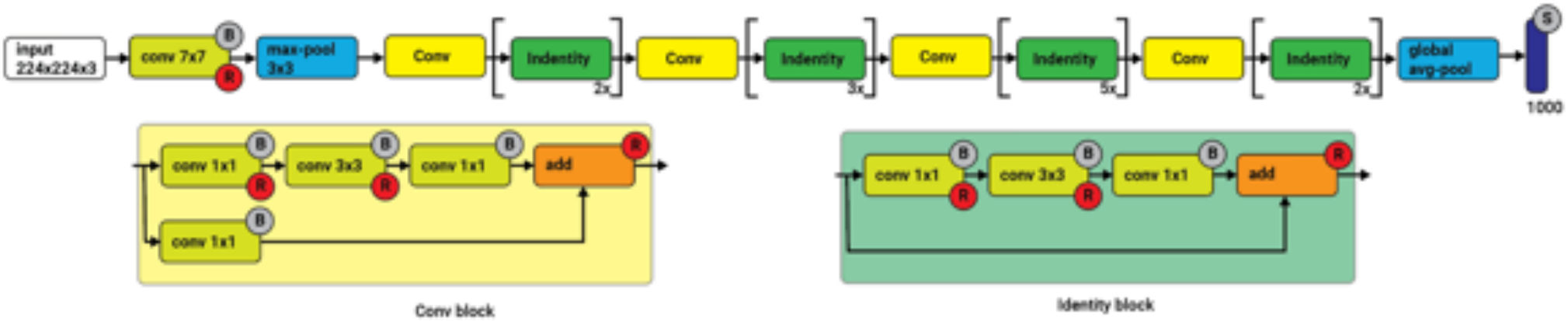






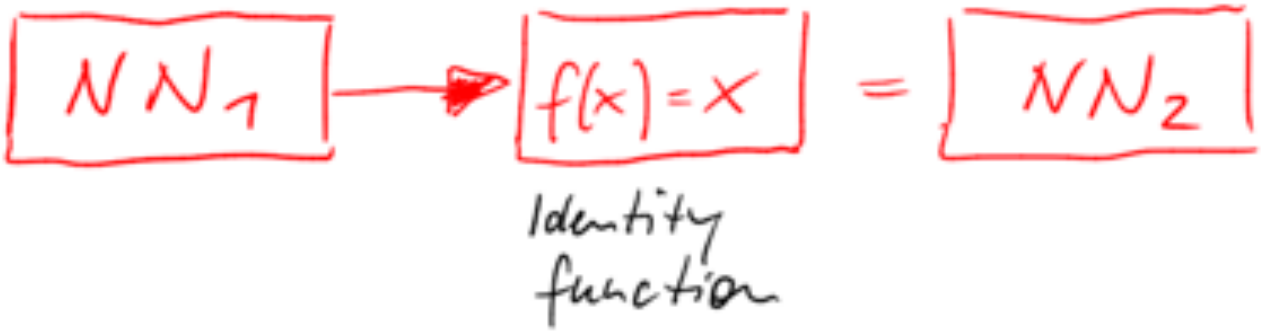
ResNet 50 (2015)

- **Paper:**
  Kaiming He et al.: Deep Residual Learning for Image Recognition, CVPR 2016
- **Novelty / Idea:**
  - Skip connections
  - Designing deeper CNNs (152 Layers) without without losing generalization power
  - Batch normalization



Residual Neural Network (ResNet)

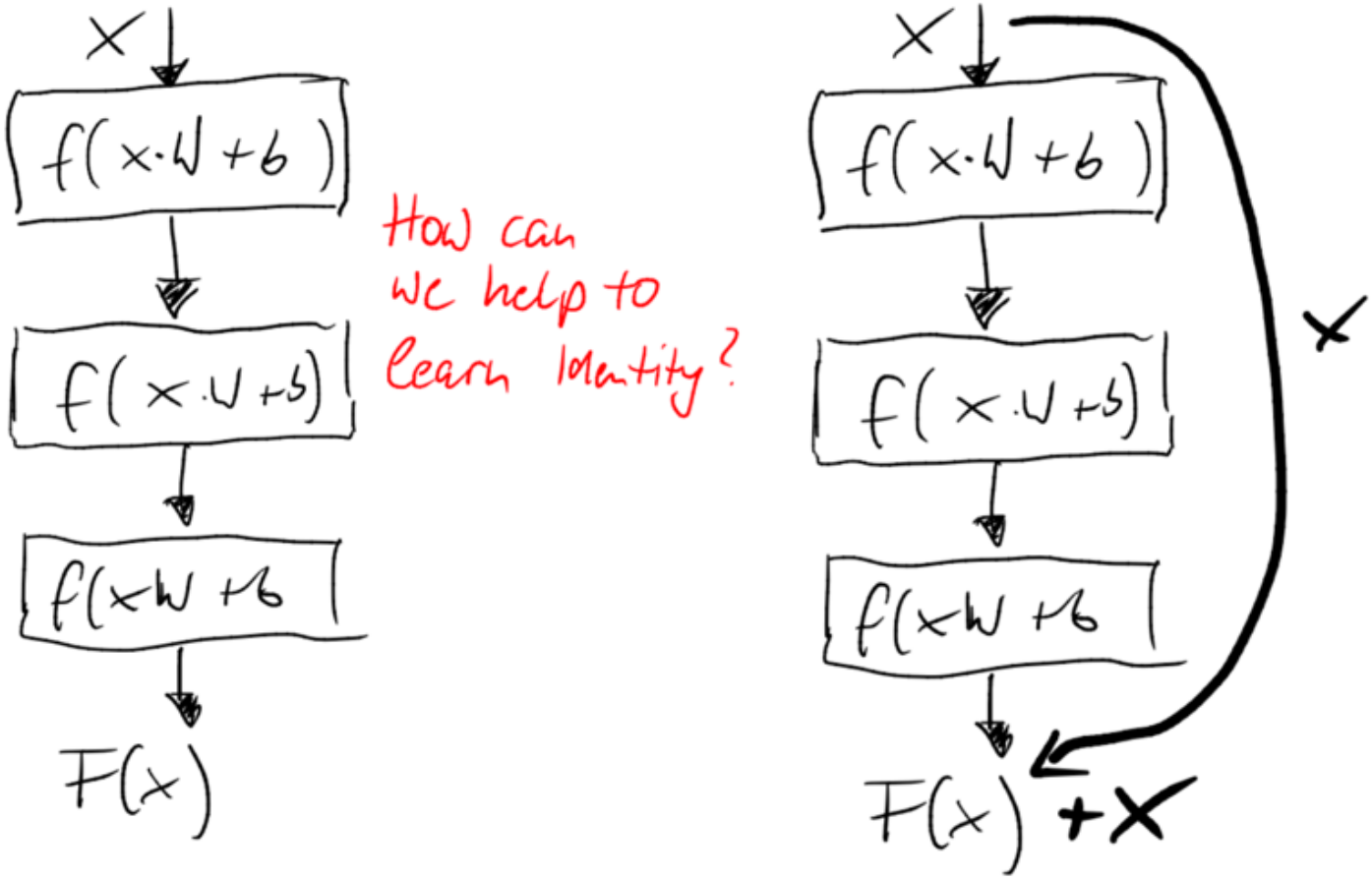
Neural Networks are great function approximators

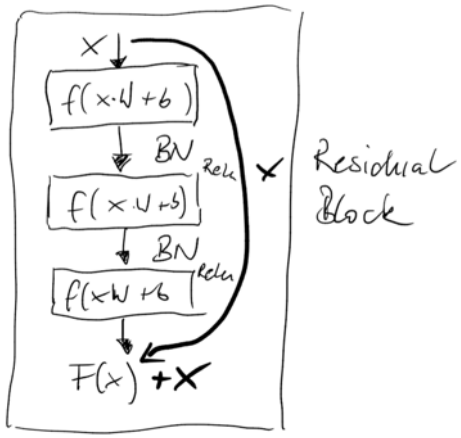


NN_2 should work at least as good as NN_1

→ Reality: It gets worse!

Residual Neural Network (ResNet)



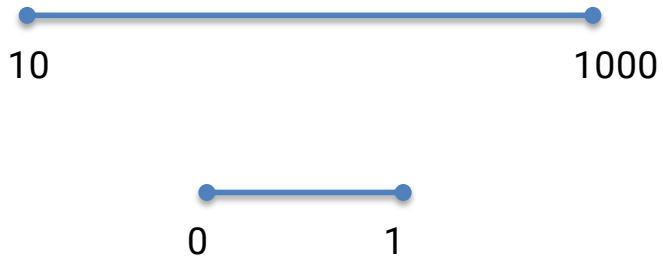


Desired mapping: $H(x) = F(x) + X$

Easier to learn residuals $H(x) - X$

Batch Normalization

- We want to normalize / standardize data as proprocessing step
  - Prepare data to get it ready for training
  - All data on same scale
  - Mean = 0, std = 1
- Why?
  - Instability in NN, because diffs in values can lead to imbalanced gradients (exploding gradient), decrease training speed



$$z = \frac{x - m}{s}$$

Batch Normalization

- NN learns by adjusting gradients via SGD
- Instability when a value during training becomes very large

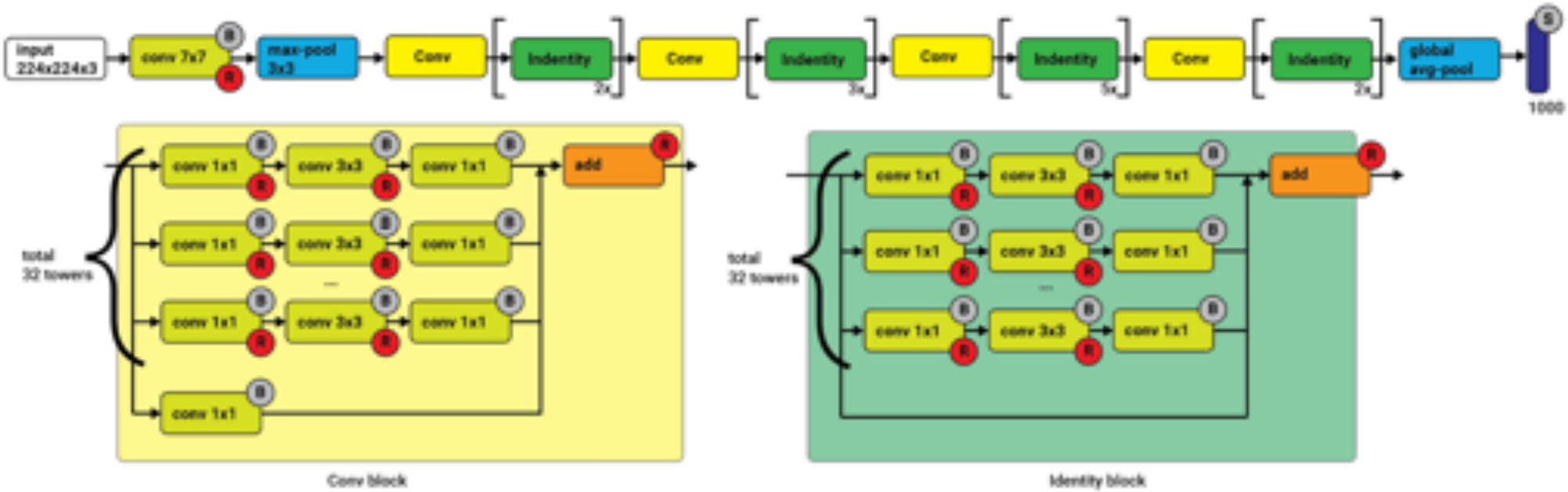
- Batch normalization is applied to a layer
  - Normalizes the output of activation function
  - Multiply normalized output by arbitrary parameter
  - Add arbitrary parameter to resulting product

- Results in new mean and std for the data
  - Parameters become also trainable params

- This stabilizes and speedup training
- Done on each batch

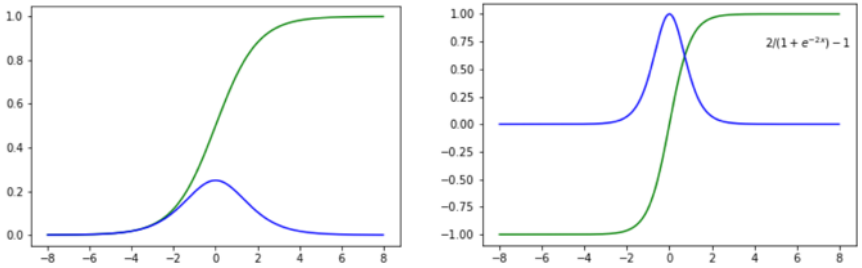
ResNeXt-50 (2015)

- **Paper:**
  Saining Xie et al.: Aggregated Residual Transformations for Deep Neural Networks, CVPR 2017
- **Novelty / Idea:**
  - Scaling up the number of parallel towers (“cardinality”) within a module similar to Inception networks



Weight Initialization

Zero / Random initialization



- In general:
  - **Biases are initialized with zero**
  - **Weights are initialized randomly**

What happens if Weights are zero?

1. Weights are initialized with very high values
   - *np.dot(W,X)+b* becomes higher
   - activation function e.g sigmoid() maps its value near to 1 where slope of gradient changes slowly **-> learning takes a lot of time, gradient vanishes**
2. Weights are initialized with very low values
   - *np.dot(W,X)+b* get mapped to zero
   - activation function e.g sigmoid() maps its value near to 1 where slope of gradient changes slowly **-> learning takes a lot of time, gradient vanishes**

Xavier / He initialization

- He et al. (2015) proposed activation aware initialization of weights (for ReLu and leaky ReLU)

$$W^l = randn(n^l, n^{l-1}) \cdot \sqrt{2/(n^{l-1})}$$

- Xavier initialization is standard heuristic method (tanh)

$$W^l = randn(n^l, n^{l-1}) \cdot \sqrt{1/(n^{l-1})}$$

- Other commonly used initialization:

$$W^l = randn(n^l, n^{l-1}) \cdot \sqrt{2/(n^{l-1} + n^l)}$$