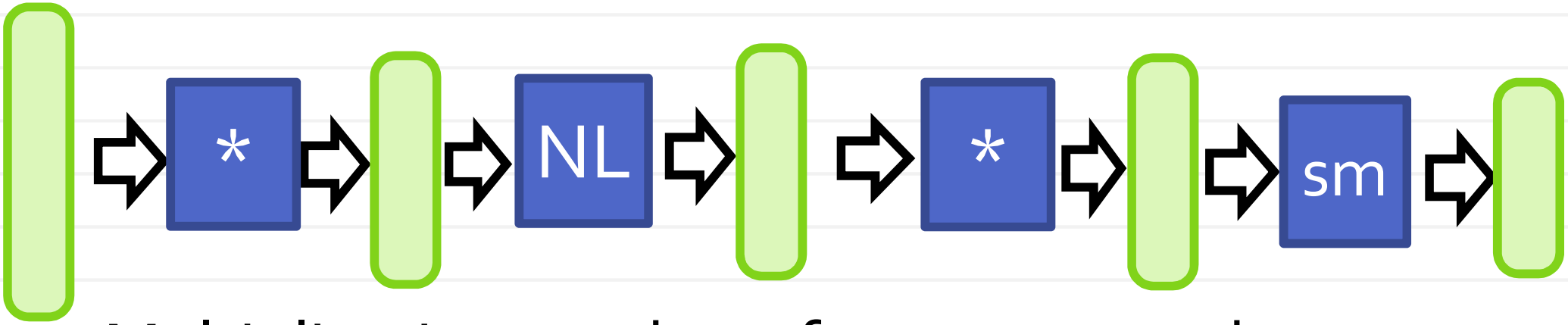
Lecture 4: Convolutional Networks

Neural network: parameter overdoze



Multiplicative number of parameters: the main problem. Solving the problem:

- Stop optimization early (always keep checking progress on validation set)
- Impose smoothness (weight decay)
- Bag multiple models

Main avenue: picking a less generic architecture with less parameters

Huber-Wiesel

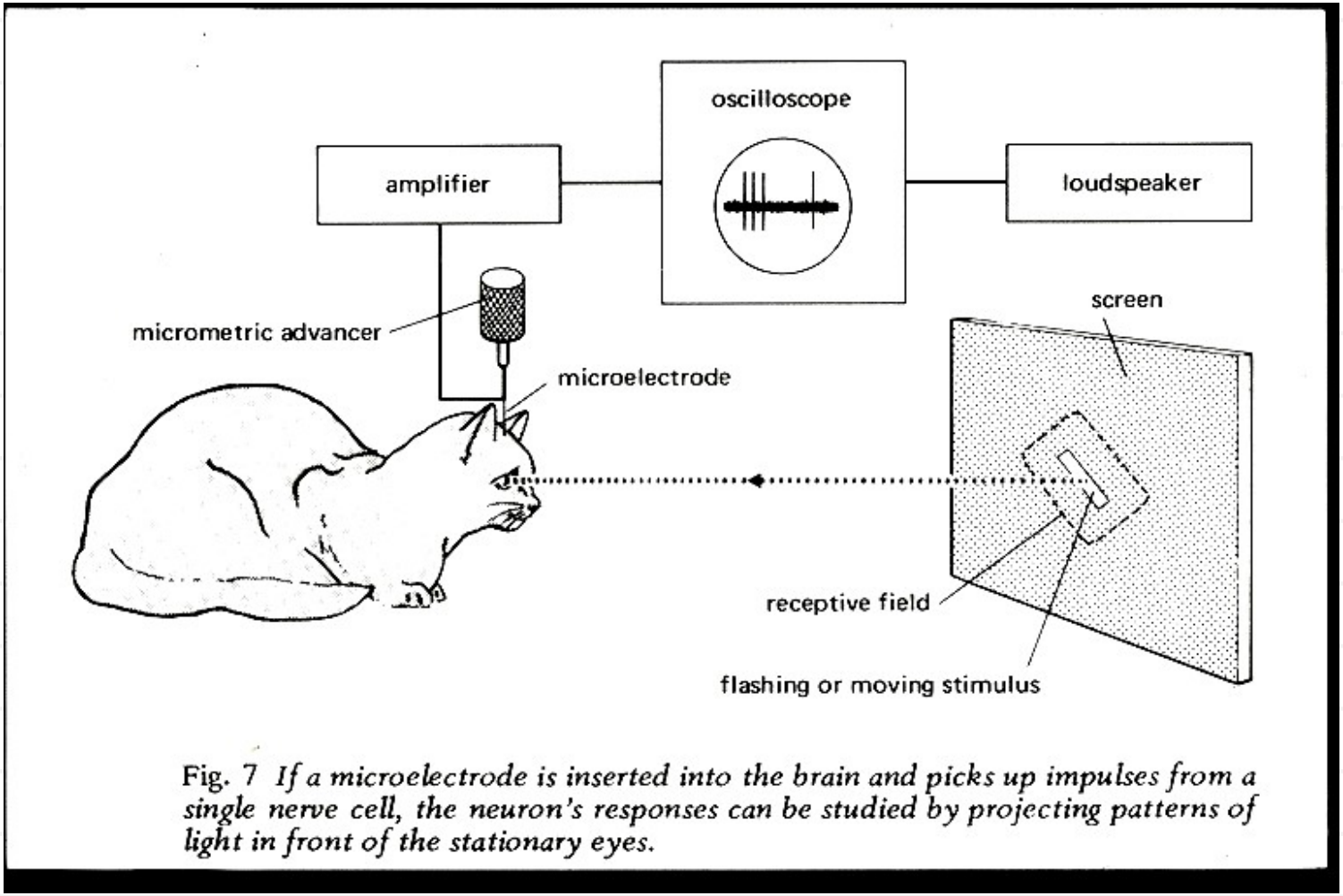
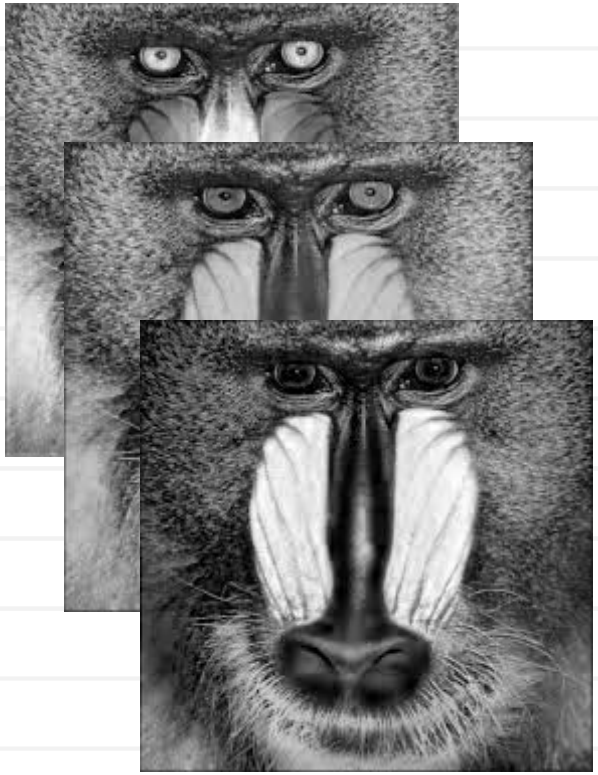
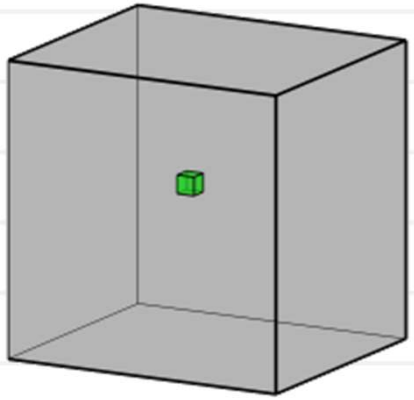


Fig. 7 *If a microelectrode is inserted into the brain and picks up impulses from a single nerve cell, the neuron's responses can be studied by projecting patterns of light in front of the stationary eyes.*

The reaction of a neuron is localized to a part of the visual field

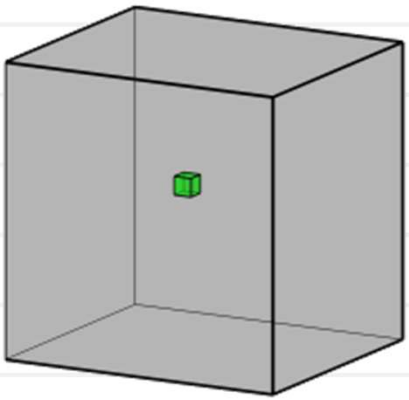
Idea 1: neuron maps

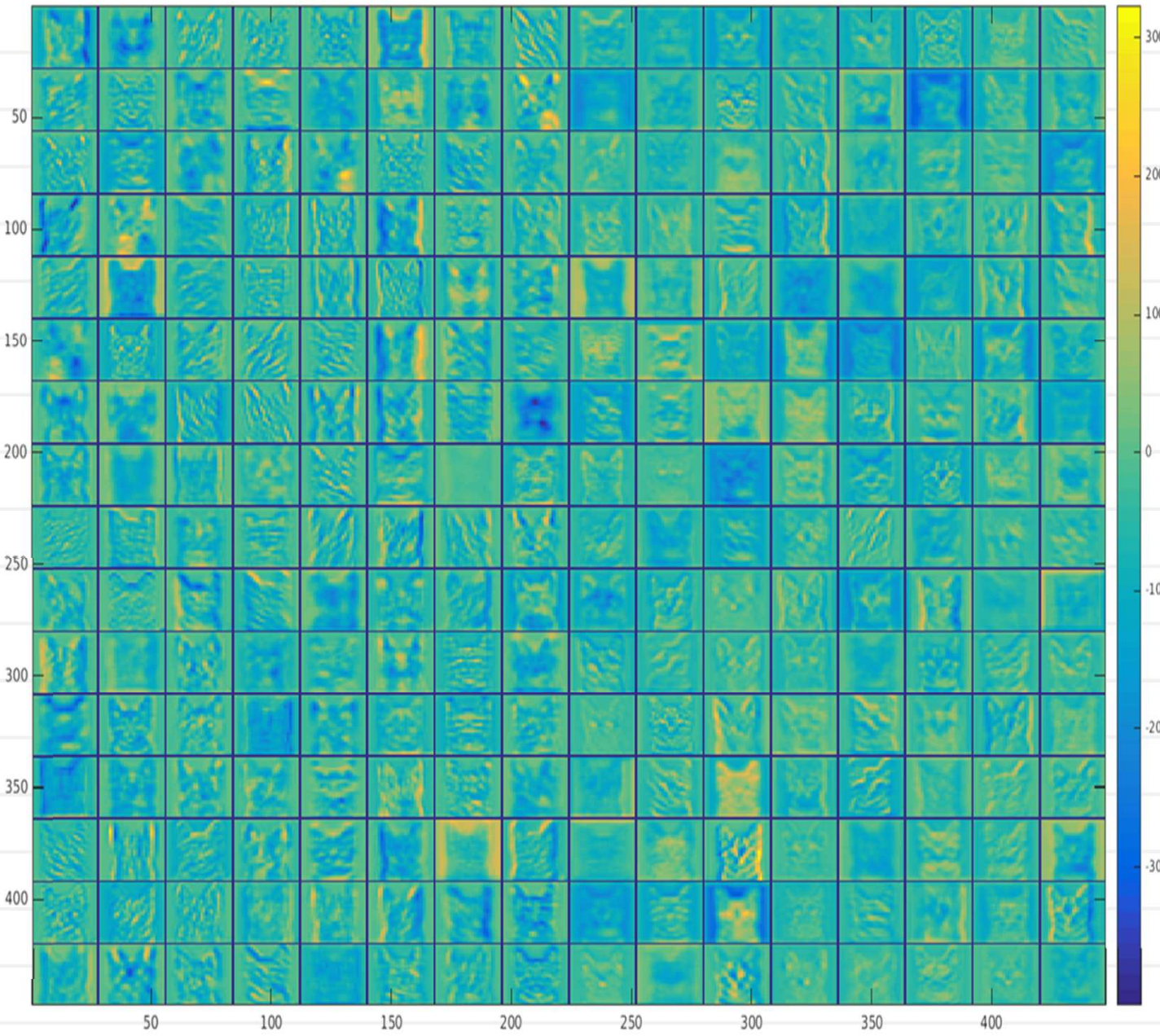




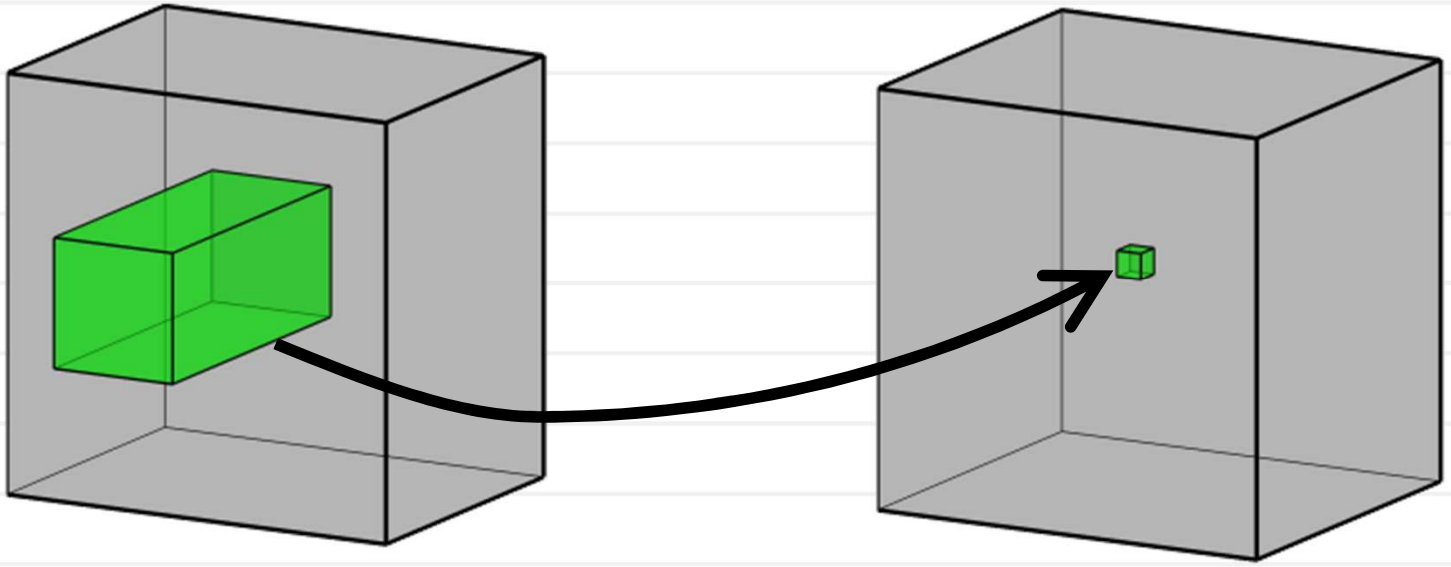
1. Organize neurons into *map stacks*
2. E.g. an image is a WxHx3 map stack

Idea 1: neuron maps





Idea 1: limited receptive field

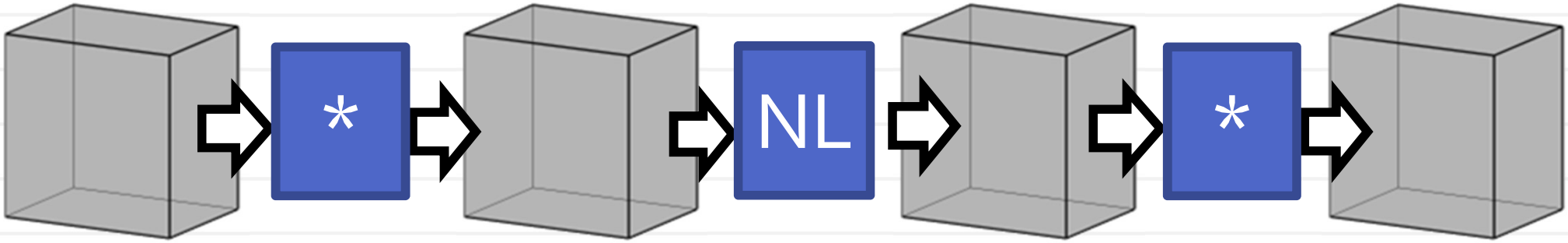


1. Organize neurons into *maps*
2. Limit the *receptive field* in the multiplicative layer:

$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s)$$

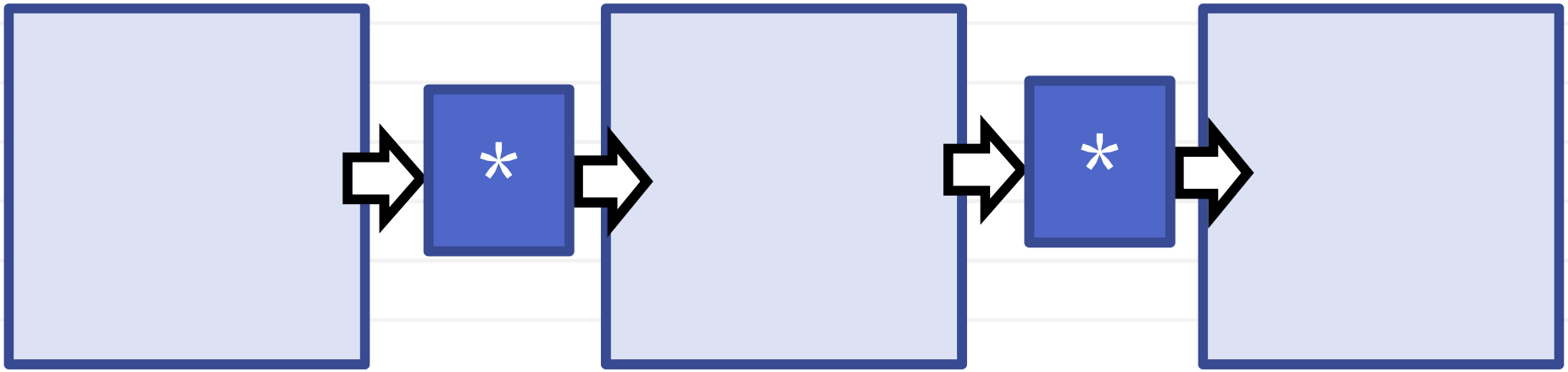
3. Huge parameter reduction $O((W/2\delta)^2)$

Stacking layers



- The layers are stacked and interleaved with non-linearities (e.g. ReLU)

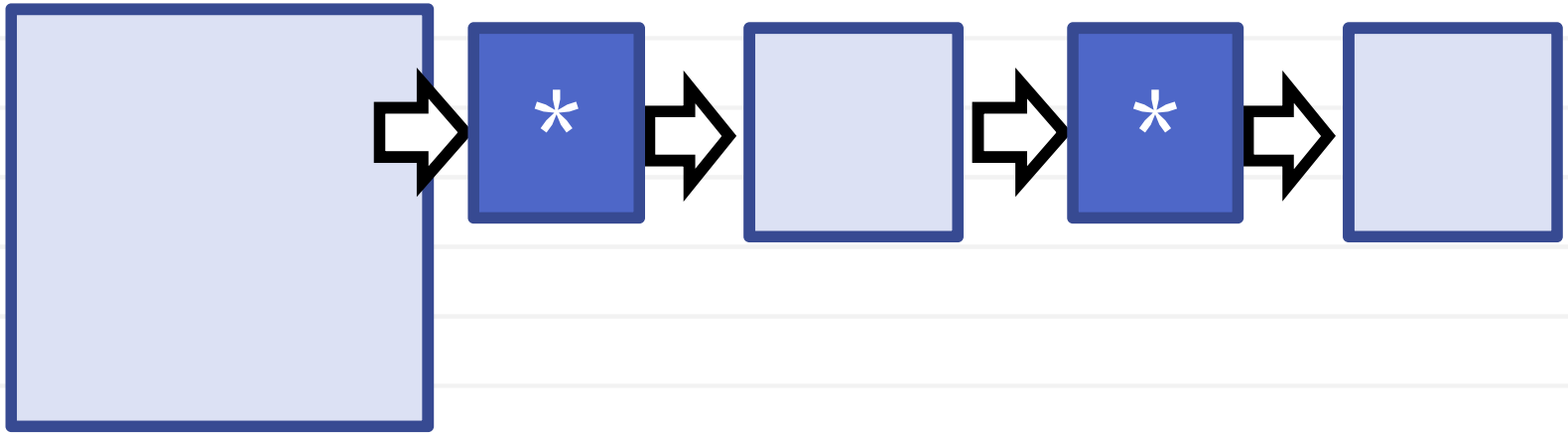
Growing receptive field



E.g. with 7x7 filters:

- Receptive field is 7x7 after 1 conv layers
- Receptive field is 13x13 after 2 conv layers

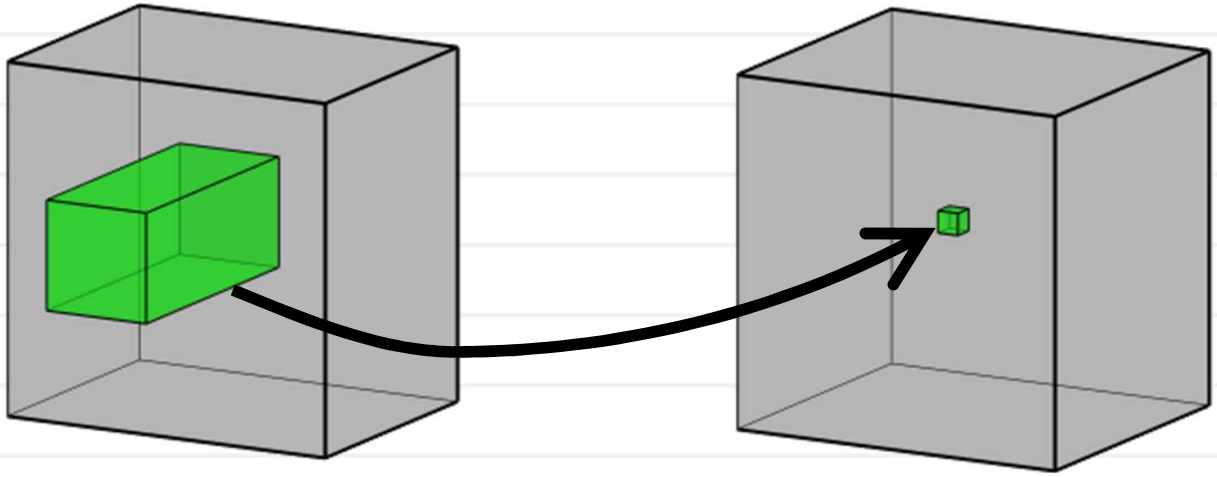
Accelerating growth: stride



- In the early layers strides are often > 1
- Strides > 1 downsample maps
- Strides > 1 increase the receptive fields

What is the receptive field after [7x7 stride=2] followed by [7x7 stride = 1] convolution?

Idea 2: tying together weights

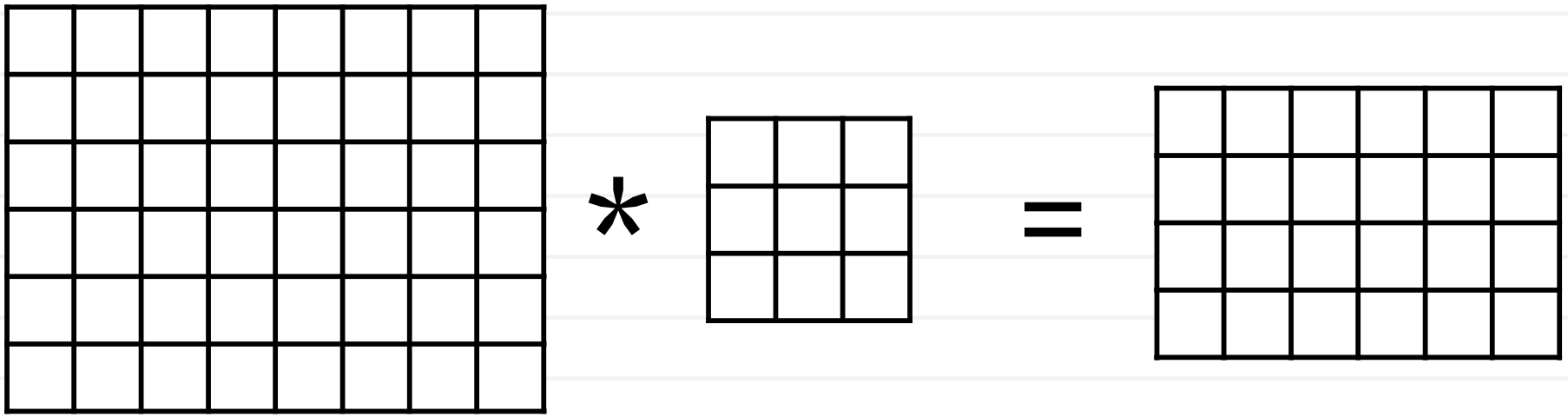


$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s)$$

Further dramatic reduction in the number of parameters: O(W^2)

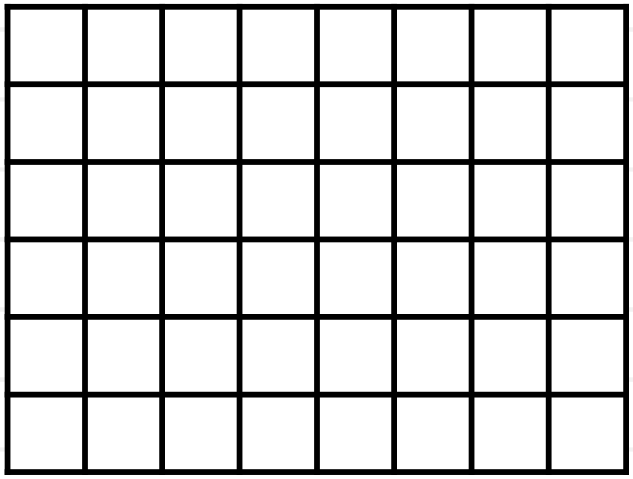
Boundary issues

"Valid" mode:



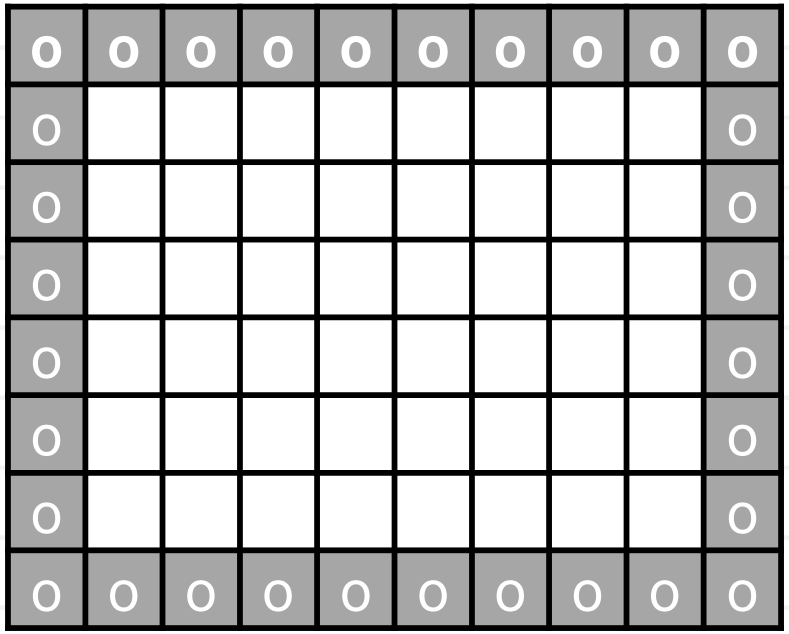
- Complicates implementation and reasoning
- Unequal contribution of elements

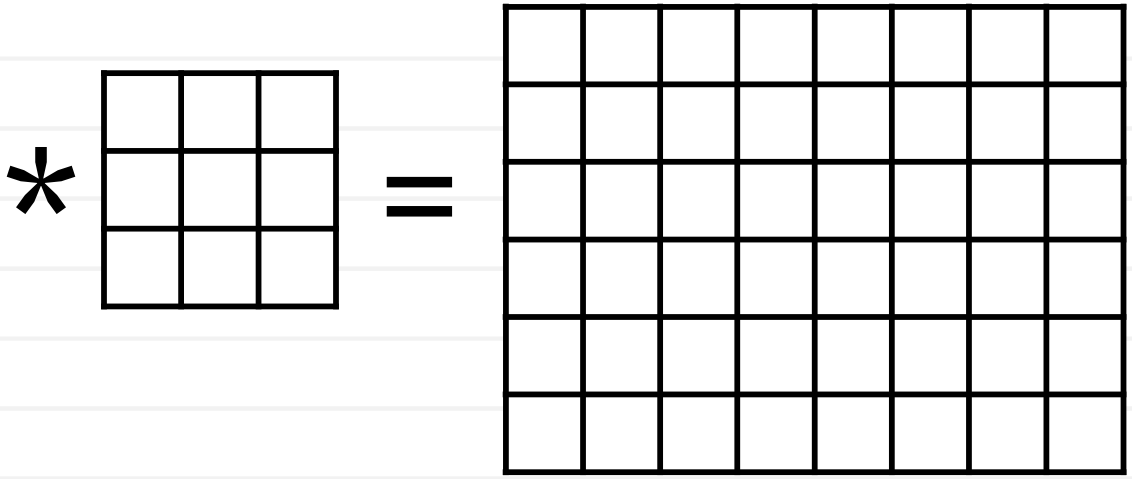
Boundary issues: padding with zeros



"Same" mode:

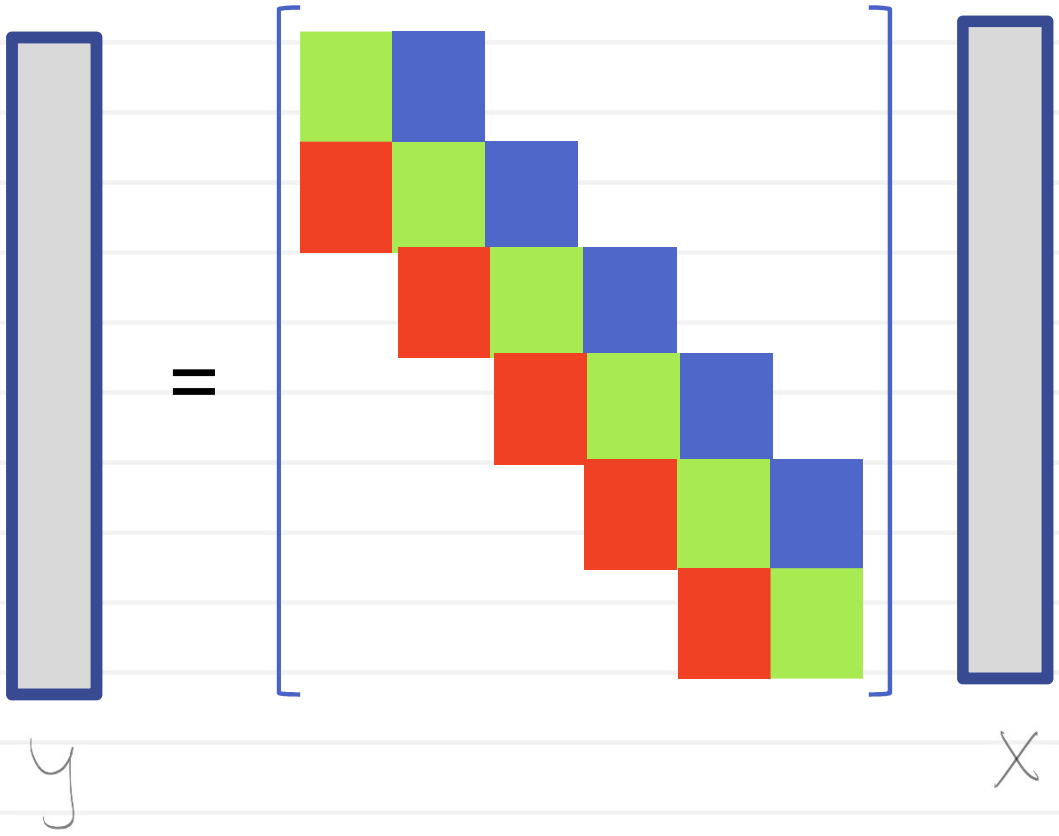
- Solves the problem
- Introduces "false" edges





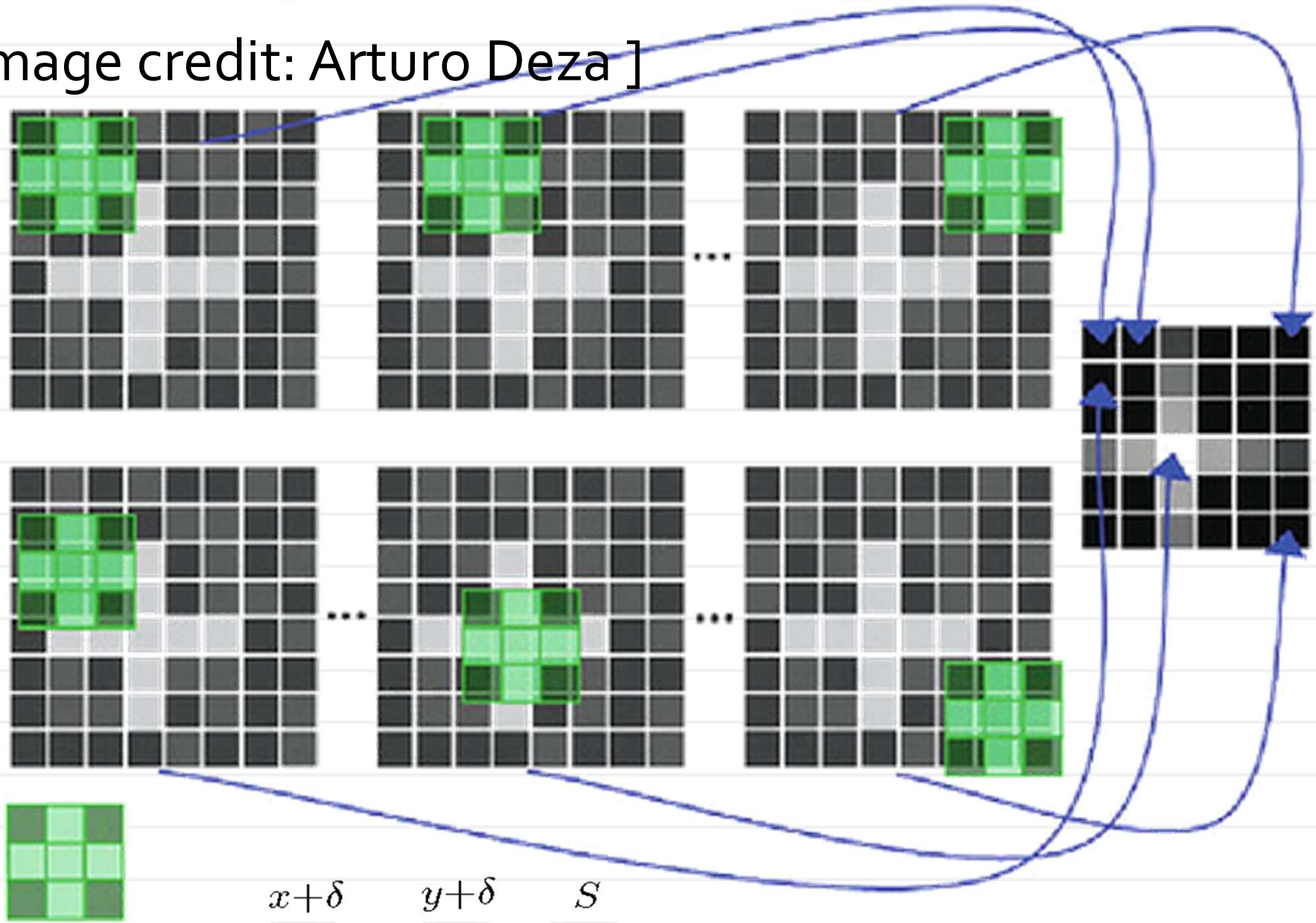
Conv. layer is still multiplicative

E.g. 1D correlation with 
is a multiplication over a banded matrix:



Interpretation: looking for patterns

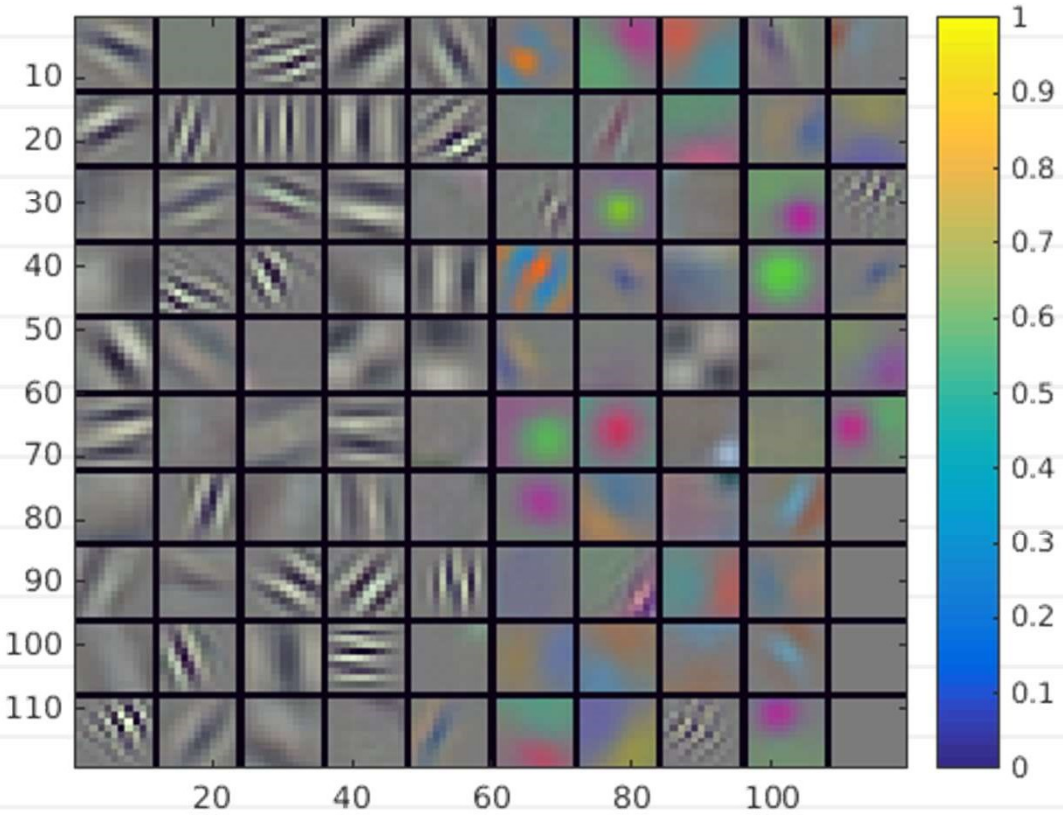
[Image credit: Arturo Deza]



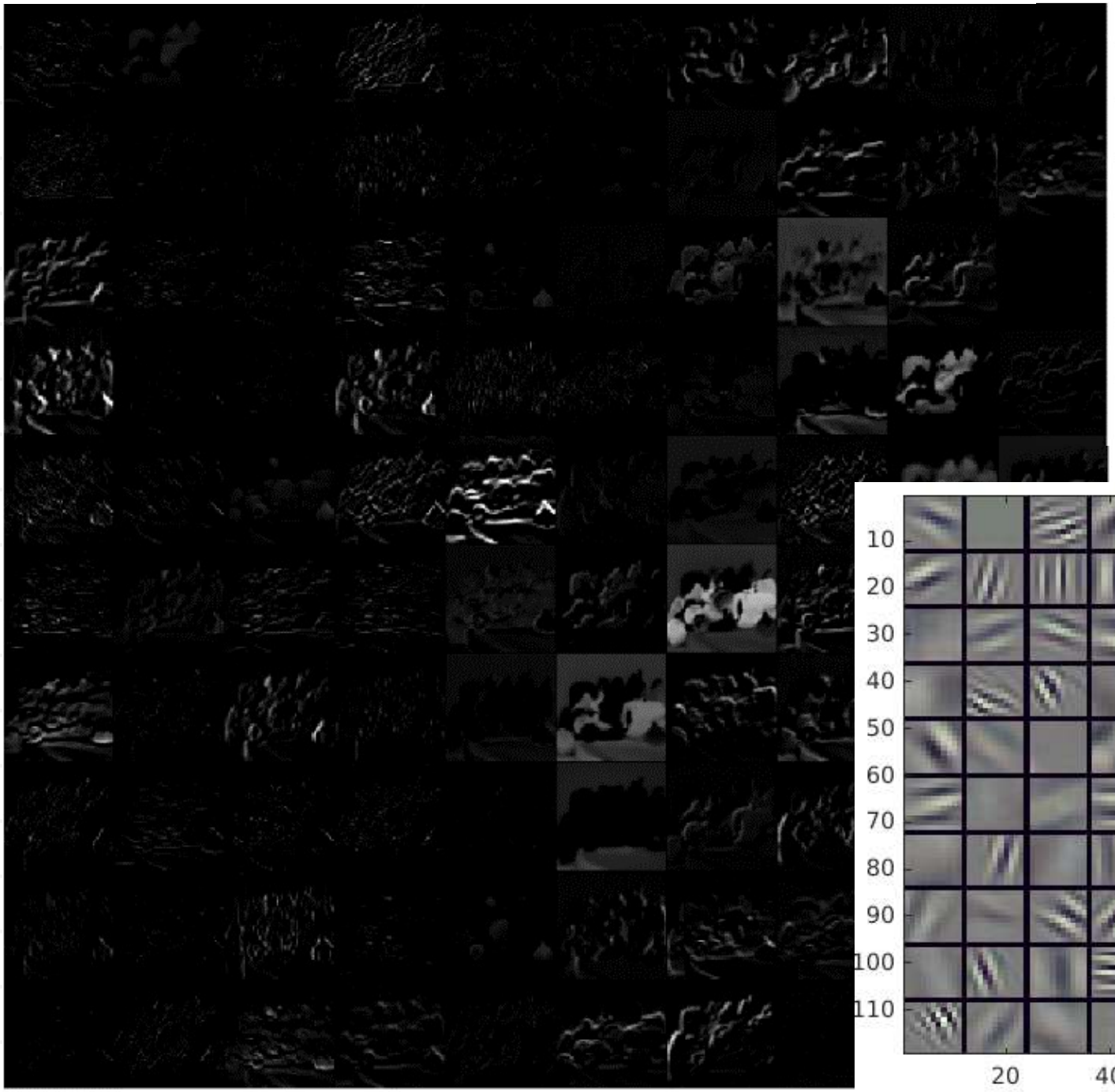
$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s)$$

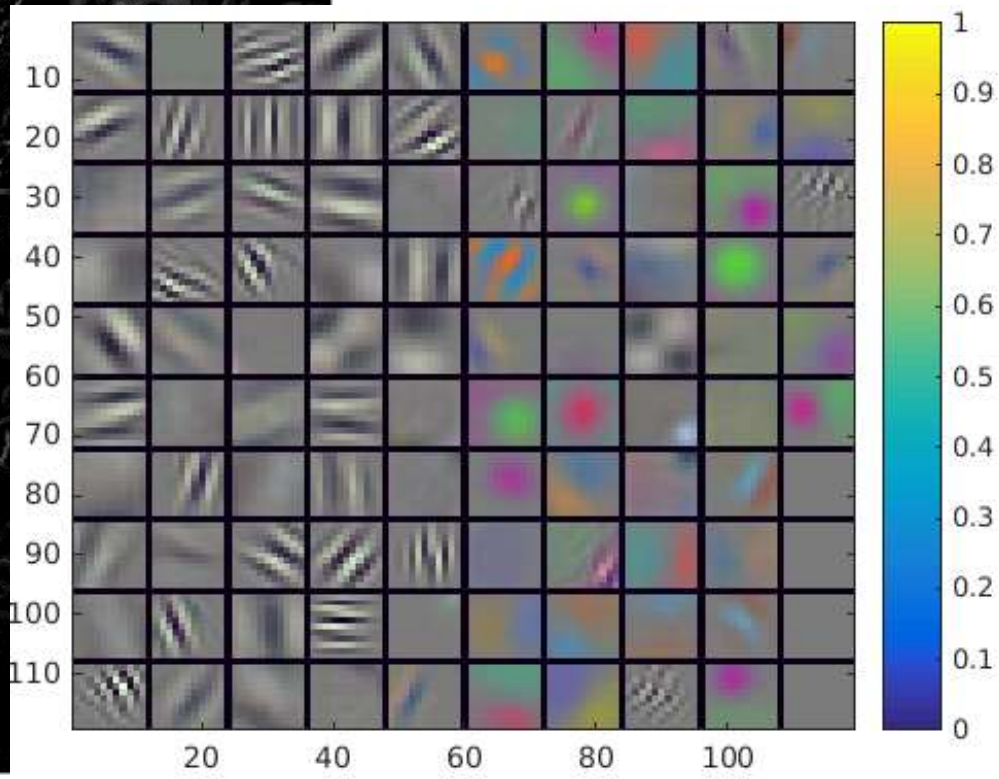
Interpretation: looking for patterns

AlexNet filters of the first layer:

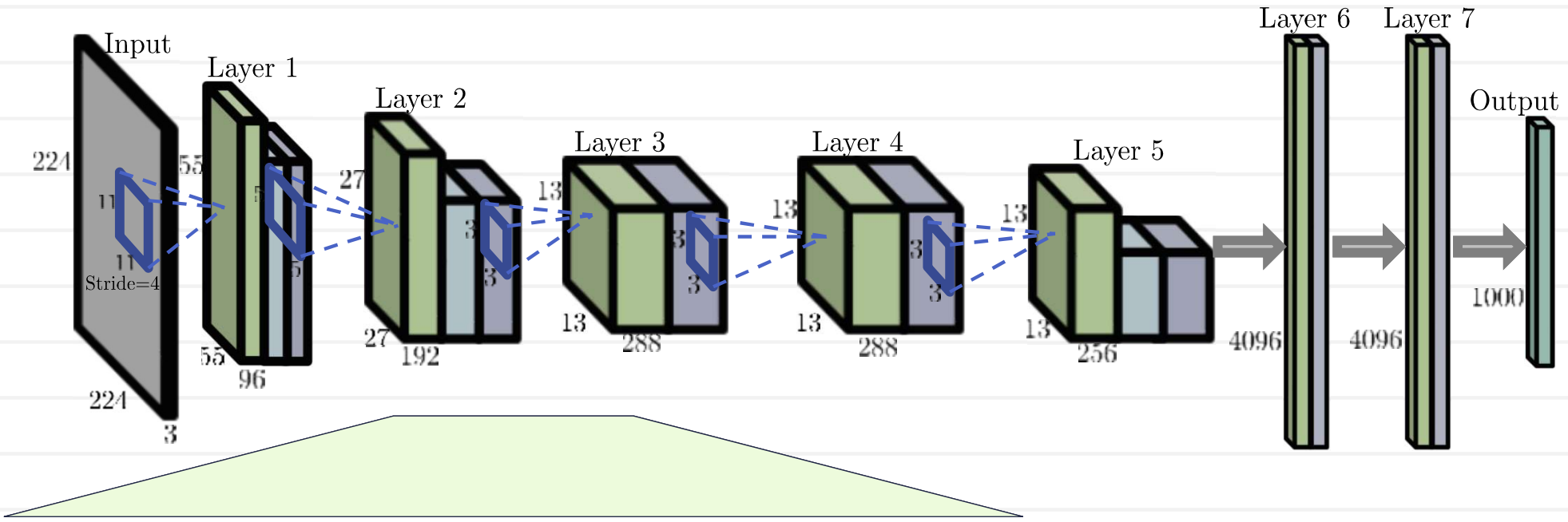


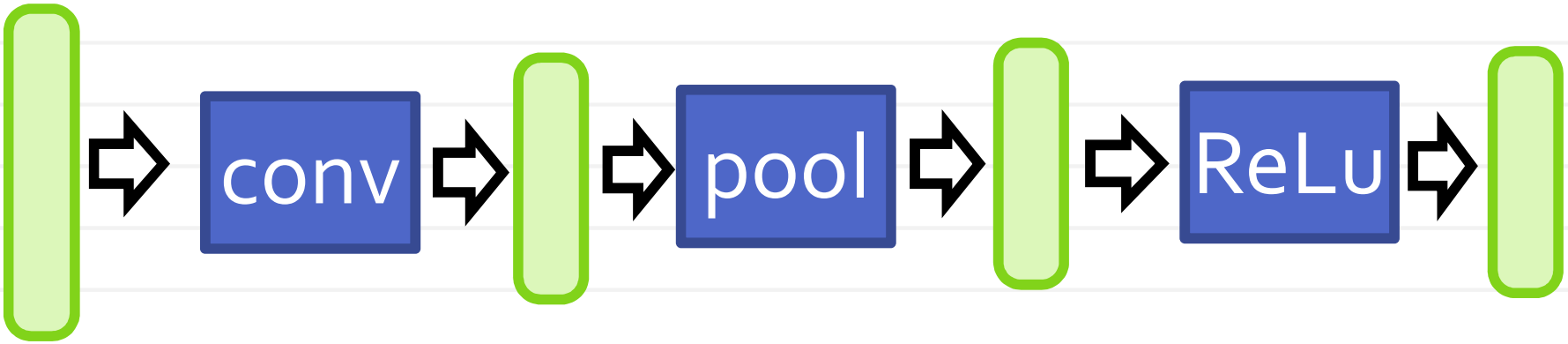
Responses in the first layer



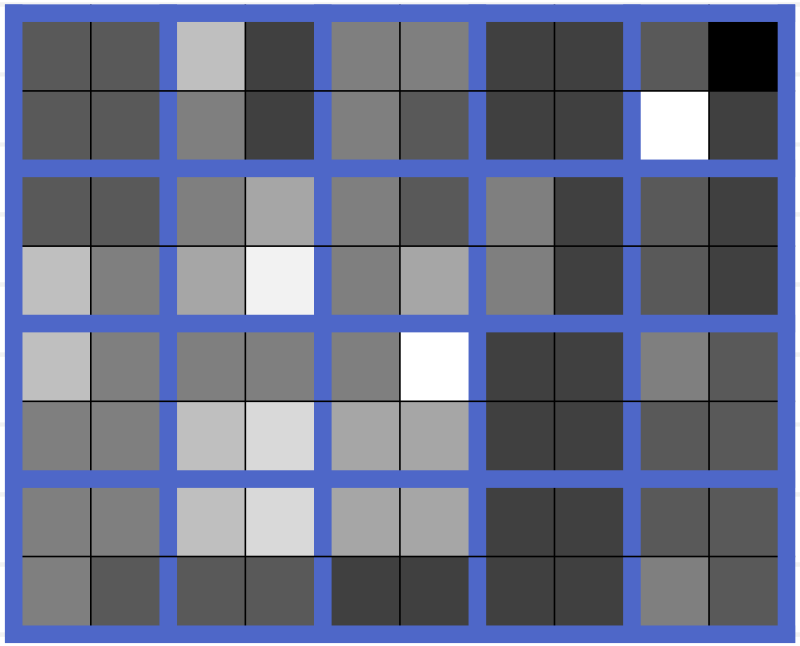


What are modern ConvNets made of

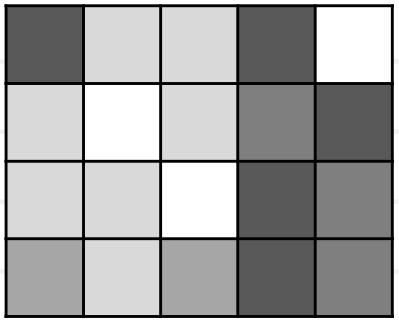




Third-component: pooling (+subsampling)



max-pooling:



Pooling is almost always with subsampling

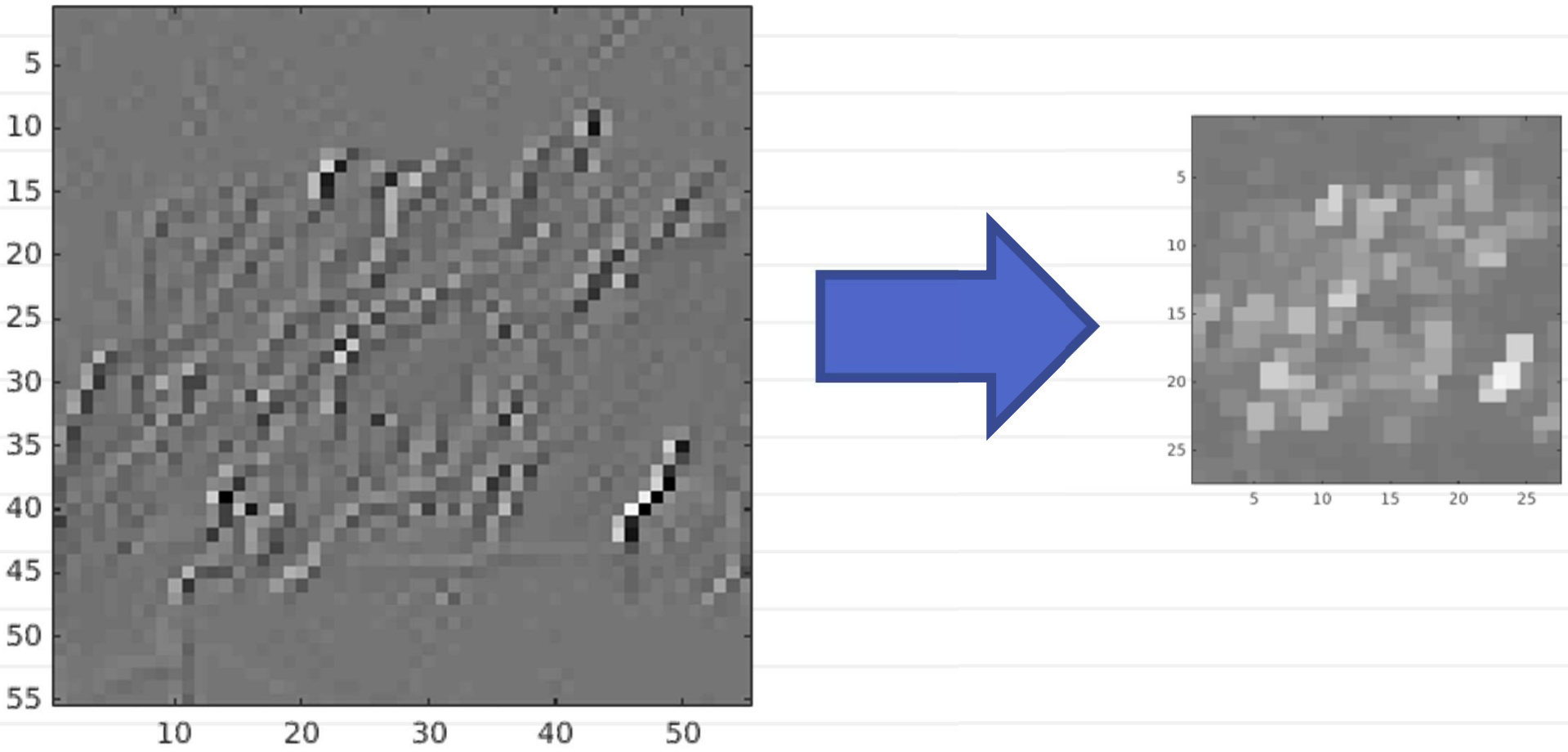
Alternatives: sum-pooling,

$$\left(\sum_{i \in \Omega} x_i^p \right)^{1/p}$$

Rapid decrease of map size

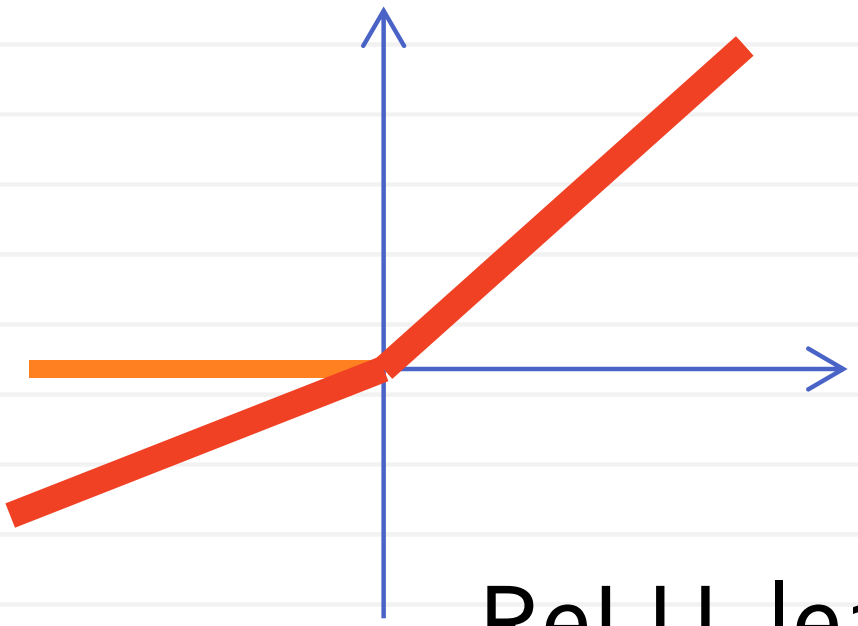
Parameter-free

Max-pooling and jitter-invariance



- Usual motivation: adding invariance to small shifts
- Several max-poolings can accumulate invariance to stronger shifts

Details: nonlinearity



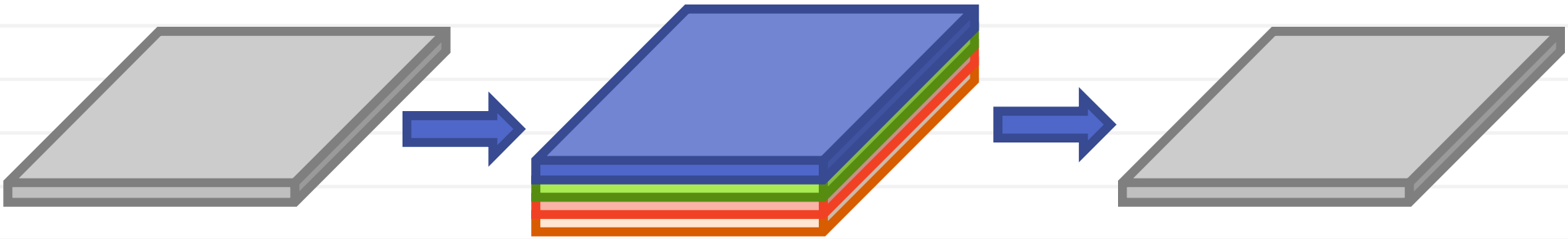
$$f(x;\alpha) = \max(x, \alpha x)$$

$$\frac{dz}{d\alpha} = [\alpha x > x]\, x^T \cdot \frac{dz}{dy}$$

ReLU, leaky ReLu

Another popular non-linearity: *maxout*

$$X \longrightarrow \max(\alpha_1 X + \beta_1, \alpha_2 X + \beta_2, \ldots, \alpha_T X + \beta_T)$$



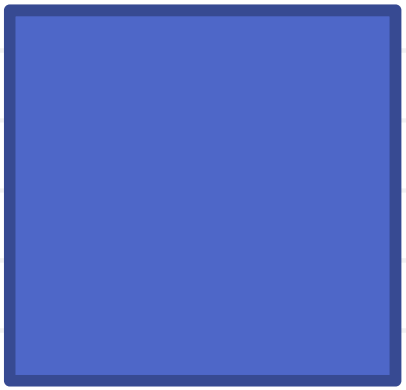
CNN applications

Pattern finding through convolution/correlation is ubiquitous:

- 2D images (and the like, e.g. speech)
- 1D signals (e.g. time series, speech)
- 3D images
- Videos
- Graphs (more generalized sense)

[Bruna et al. Spectral Networks and Locally Connected Networks on Graphs. ICLR 2014]

Reminder: Layer abstraction



Each layer is defined by:

- forward performance: $y = f(x)$
- backward performance:

$$z(x) = z(f(x;w)) \qquad y = f(x;w)$$

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy} \qquad \frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

Backprop equations: multipicative layer

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy}$$

$$\frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

$$y = Wx$$

$$\frac{dy}{dx} = W$$

$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$

$$\frac{\partial z}{\partial w_{ij}} = \left(\frac{\partial y}{\partial w_{ij}}\right)^T \frac{dz}{dy} = x_j \cdot \frac{\partial z}{\partial y_i}$$

$$\frac{dz}{dW} = \frac{dz}{dy} \cdot x^T$$

Backprop equations: convolutional layer

$$\frac{dz}{dx} = \left(\frac{dy}{dx}\right)^{\top} \frac{dz}{dy}$$

$y = Wx$ - still holds for conv layer

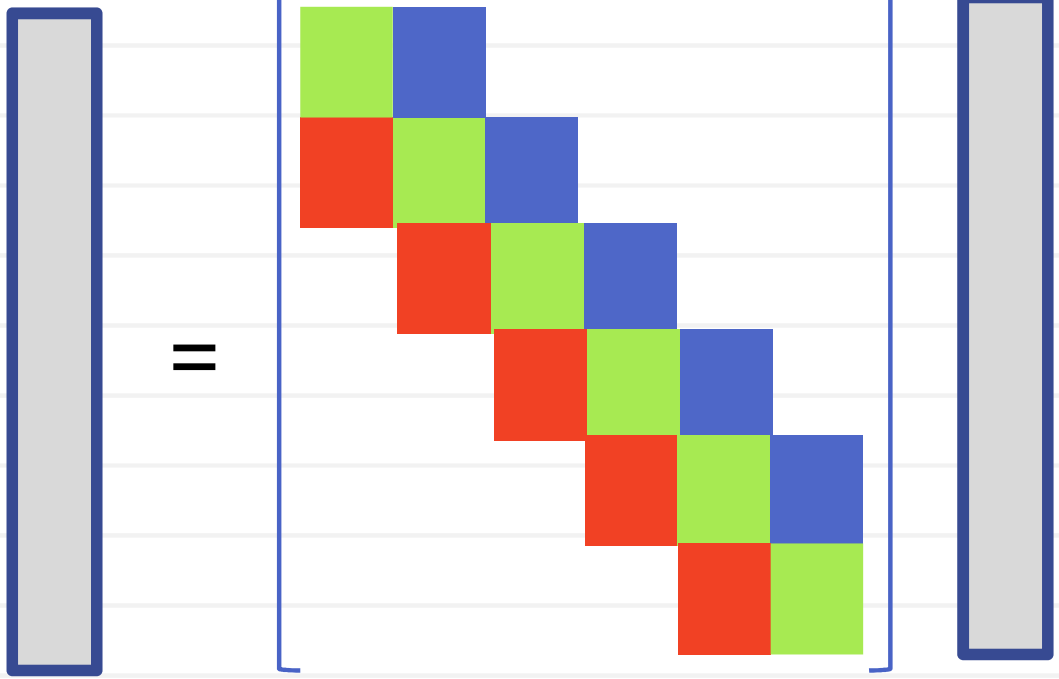
$$\frac{dz}{dx} = W^{\top} \frac{dz}{dy}$$

- also corresponds to some correlation?

Backpropagation via convolution

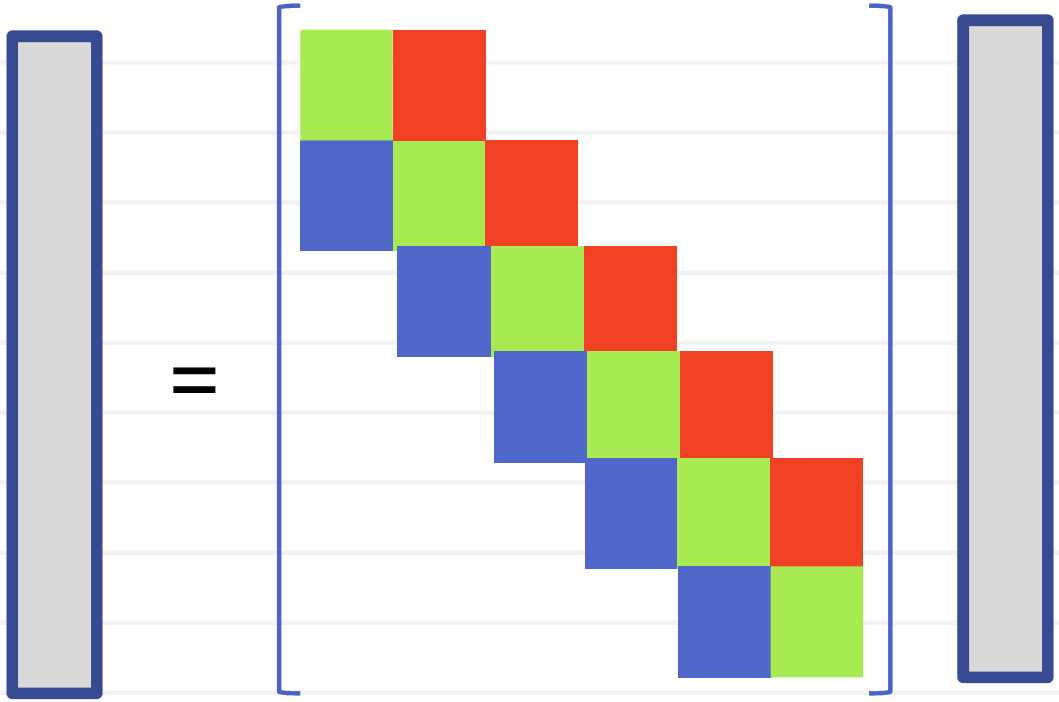
$$y = Wx$$

Corr with 



$$\frac{dz}{dx} = W^T \frac{dz}{dy}$$

Corr with 



Backprop equations: convolutional layer

$$\frac{dz}{dx} = \left(\frac{dy}{dx}\right)^{\top} \frac{dz}{dy}$$

$y = Wx$ - still holds for conv layer

$\frac{dz}{dx} = W^{\top} \frac{dz}{dy}$ - also corresponds to conv

During backpass do the same correlations but with flipped kernels

Backprop equations: convolutional layer

$$\frac{dz}{dW} = \frac{dz}{dy} \cdot X^T \qquad \frac{\partial z}{\partial w_j} = \frac{\partial z}{\partial y_i} \cdot x_j$$

In conv. layer we need to sum the terms corresponding to same relative position of y_i and x_j.

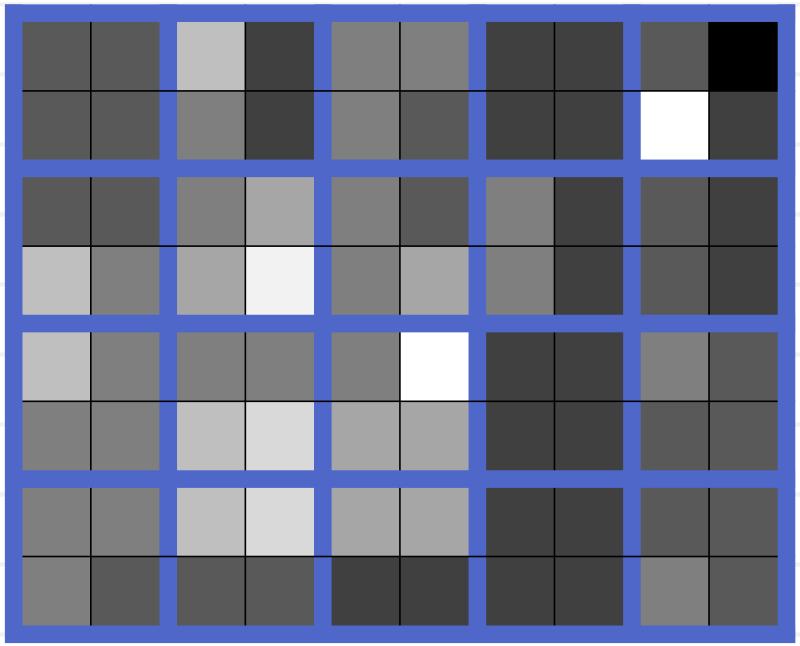
So we again need to correlate:

$$\frac{dz}{dK} = \frac{dz}{dy} \circledast X \qquad \frac{\partial z}{\partial K_{i,j,s,t}} = \sum_{m,n} \frac{\partial z}{\partial y_{m,n,t}} \cdot X_{m+i,n+j,s}$$

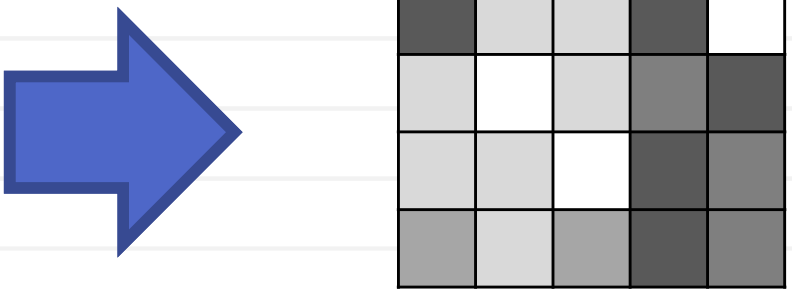
(for each pair of input-output channels)

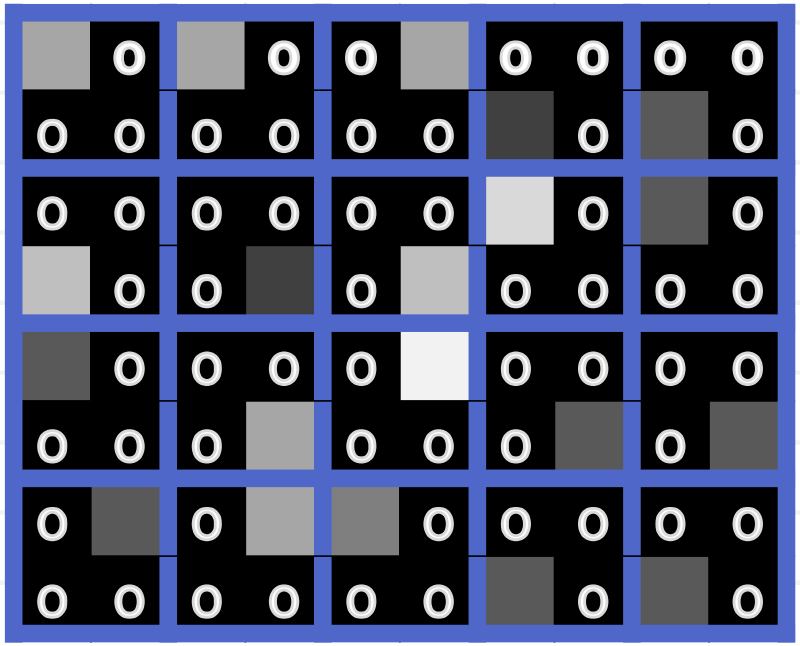
NB: strides > 1 require more "house-keeping"

Backpropagation: max-pooling



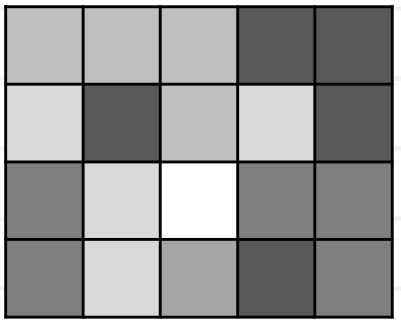
forward pass:



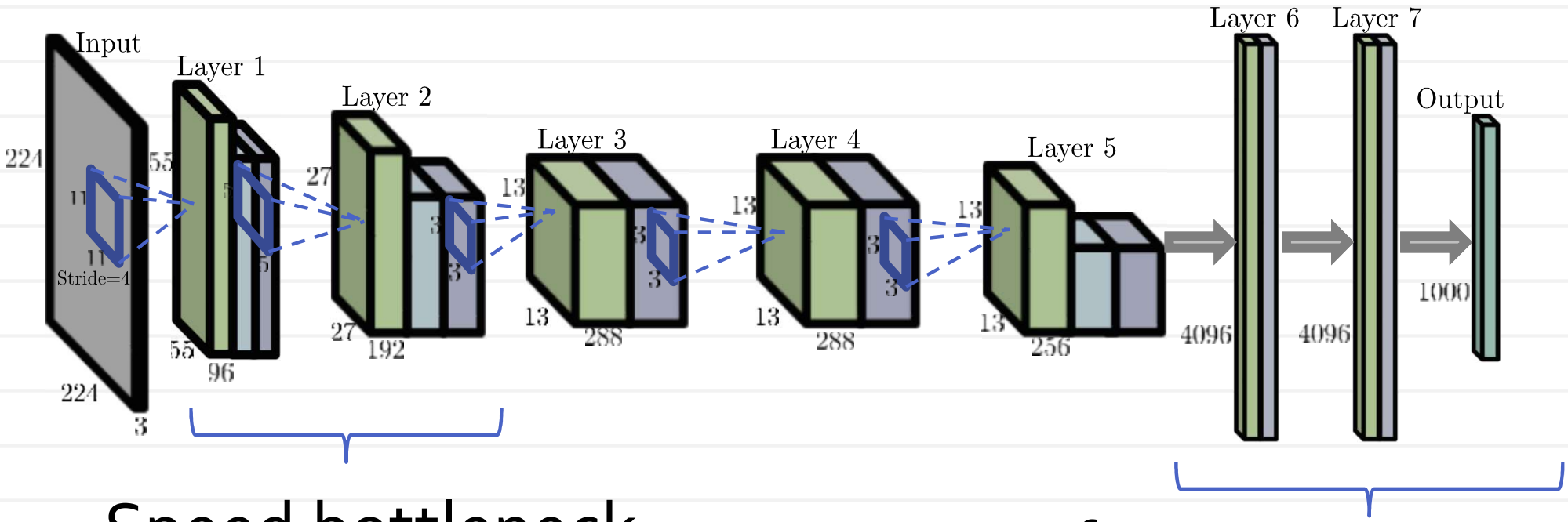


backward pass:

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy}$$



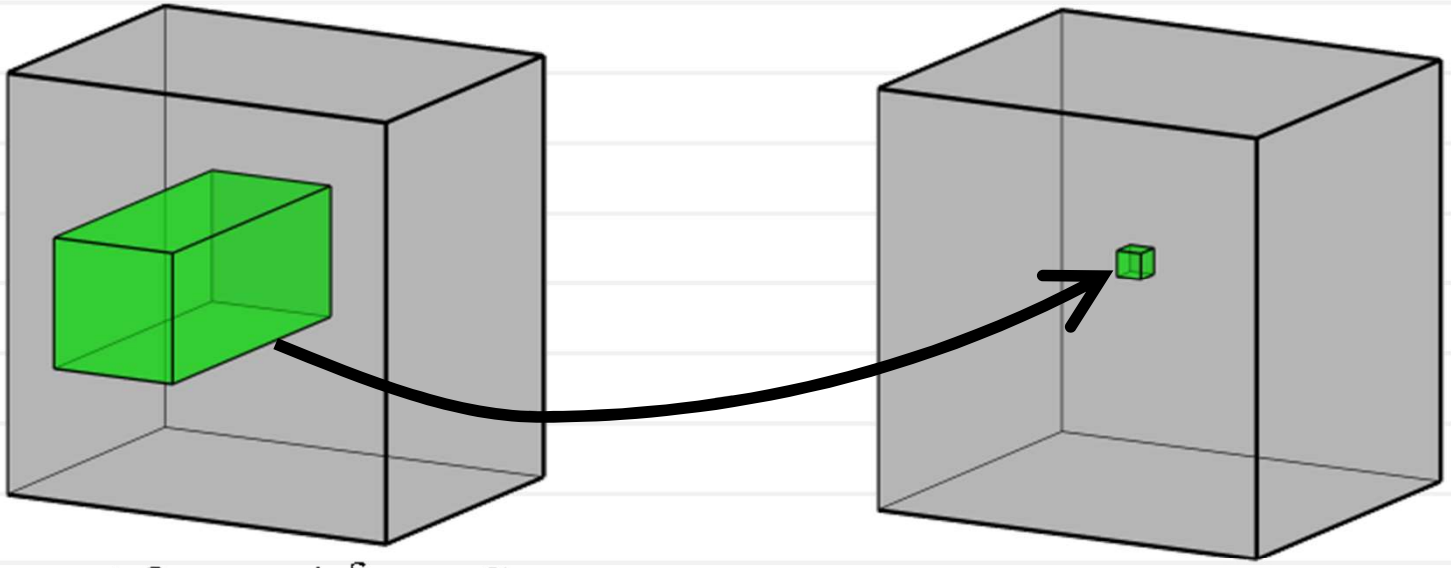
Bottlenecks



Speed bottleneck

Majority of parameters
Drop-out often applied here

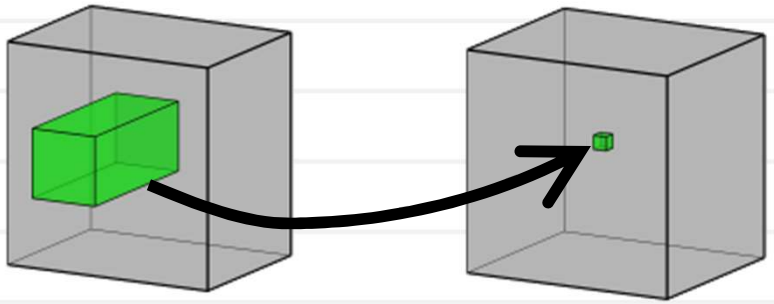
Efficient implementations: direct



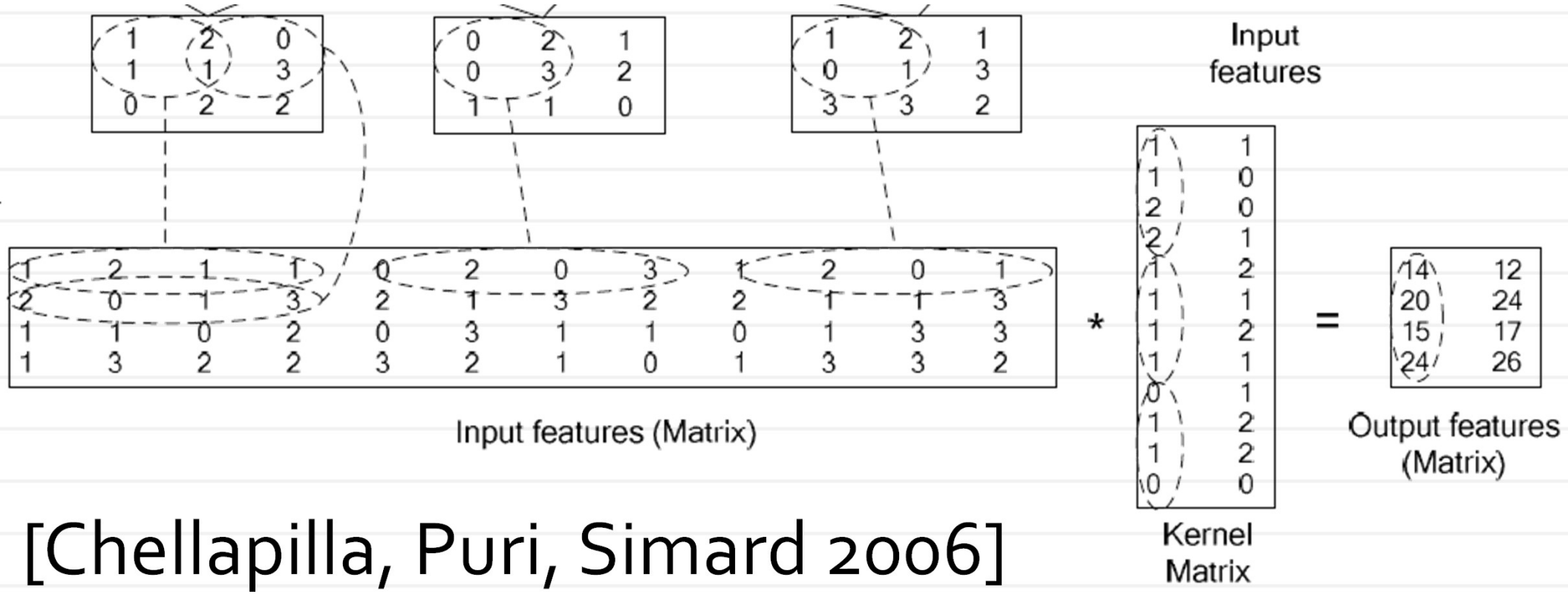
$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i-x+\delta, j-y+\delta, s, t)\, U(i,j,s)$$

- Loop ordering very important
- Data alignment very important
- NVIDIA cuDNN, Nervana kernels - efficient GPU implementations

Efficient implementations: im2col



Idea: reduce all *ST* convolutions to a single matrix multiplication



[Chellapilla, Puri, Simard 2006]

Efficient implementations: Fourier

$$A \circledast K = \mathcal{F}^{-1}\left(\mathcal{F}(A) .* \mathcal{F}(K)\right)$$

- Each map participates in many convolutions (hence FFT is reused)
- Maps are much smaller than images that most FFT codes are optimized for
- Careful implementation needed to get reasonable speed-up
- Memory hungry (why?) ☹

[Fast Training of Convolutional Networks through FFTs
Michael Mathieu; Mikael Henaff; Yann LeCun, ICLR 2014]

Benchmarking: forward times

VGG-Net (mostly 3x3 convolutions)

Library	Class	Time (ms)	forward (ms)	backward (ms)
Nervana-neon-fp16	ConvLayer	254	82	171
Nervana-neon-fp32	ConvLayer	320	103	217
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	471	140	331
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	529	162	366
TensorFlow	conv2d	540	158	382
Chainer	Convolution2D	885	251	632
fbfft (Torch)	SpatialConvolutionCuFFT	1092	355	737
cudaconvnet2*	ConvLayer	1229	408	821
CuDNN[R2] *	cudnn.SpatialConvolution	1099	342	757
Caffe	ConvolutionLayer	1068	323	745

https://github.com/soumith/convnet-benchmarks [Soumith Chintala]

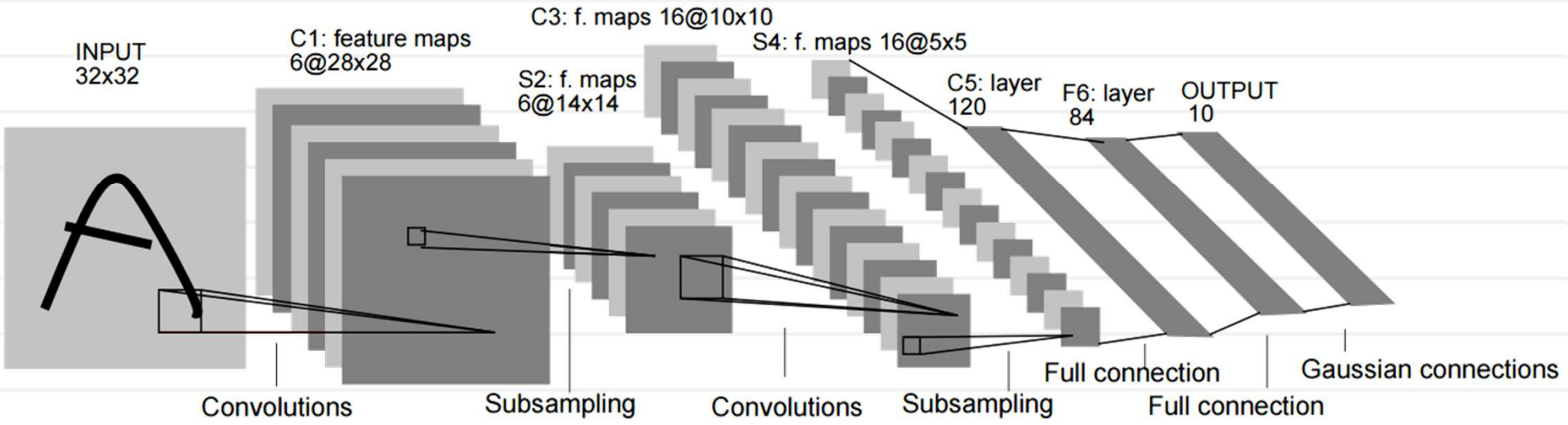
Benchmarking: forward times

AlexNet (11x11,5x5,3x3 convolutions)

Library	Class	Time (ms)	forward (ms)	backward (ms)
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	71	25	46
Nervana-neon-fp16	ConvLayer	78	25	52
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	81	27	53
TensorFlow	conv2d	81	26	55
Nervana-neon-fp32	ConvLayer	87	28	58
fbfft (Torch)	fbnn.SpatialConvolution	104	31	72
Chainer	Convolution2D	177	40	136
cudaconvnet2*	ConvLayer	177	42	135
CuDNN[R2] *	cudnn.SpatialConvolution	231	70	161
Caffe (native)	ConvolutionLayer	324	121	203

https://github.com/soumith/convnet-benchmarks [Soumith Chintala]

History: LeNet



[LeCun 89, 98]

2012: Image-net

IMAGENET

www.image-net.org

14,197,122 images,
21841 synsets indexed

(all competitions on much smaller subset 1000x1000)

Statistics of high level categories

High level category	# synset (subcategories)	Avg # images per synset	Total # images
amphibian	94	591	56K
animal	3822	732	2799K
appliance	51	1164	59K
bird	856	949	812K
covering	946	819	774K
device	2385	675	1610K
fabric	262	690	181K
fish	566	494	280K
flower	462	735	339K
food	1495	670	1001K
fruit	309	607	188K

Image-net

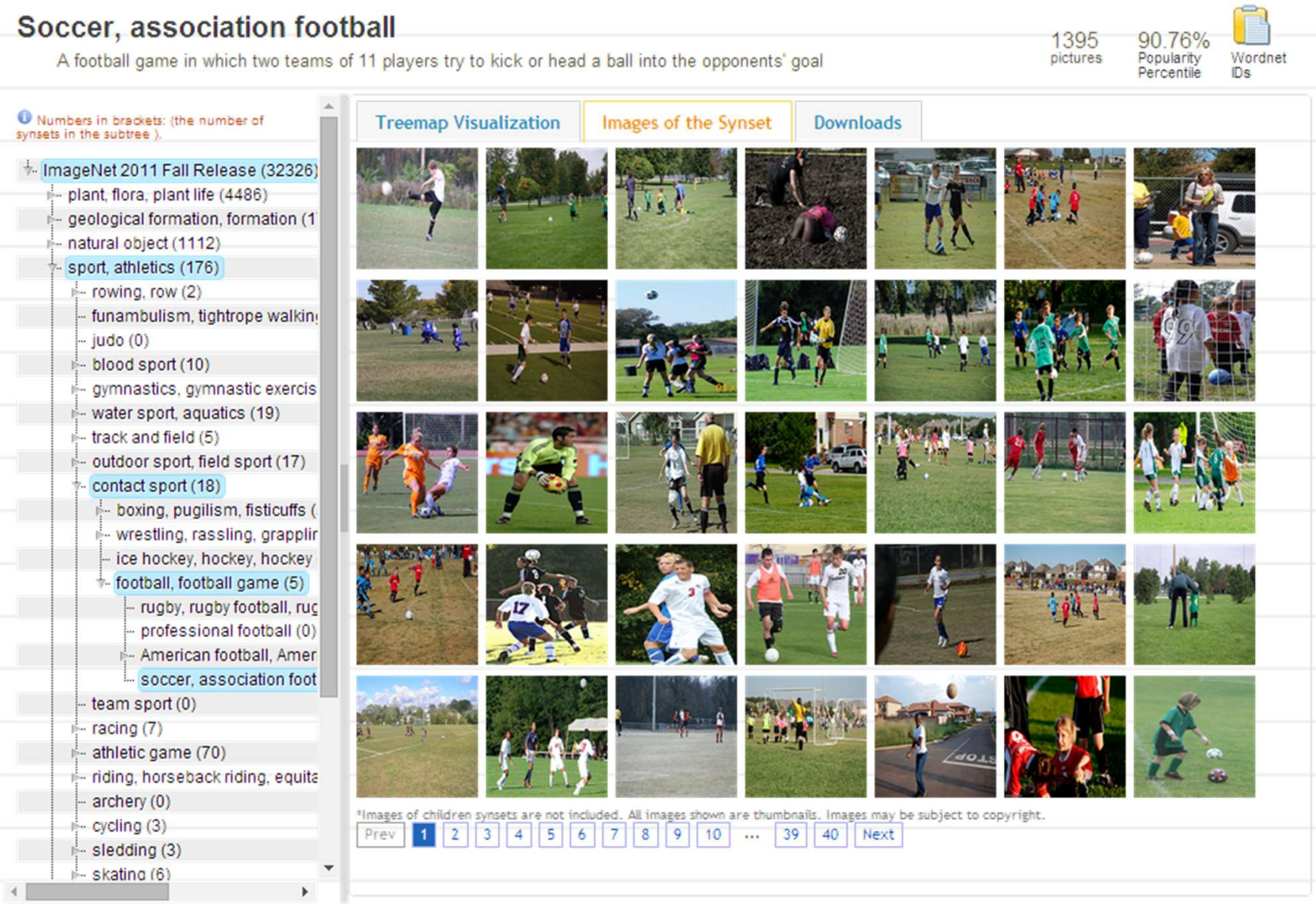
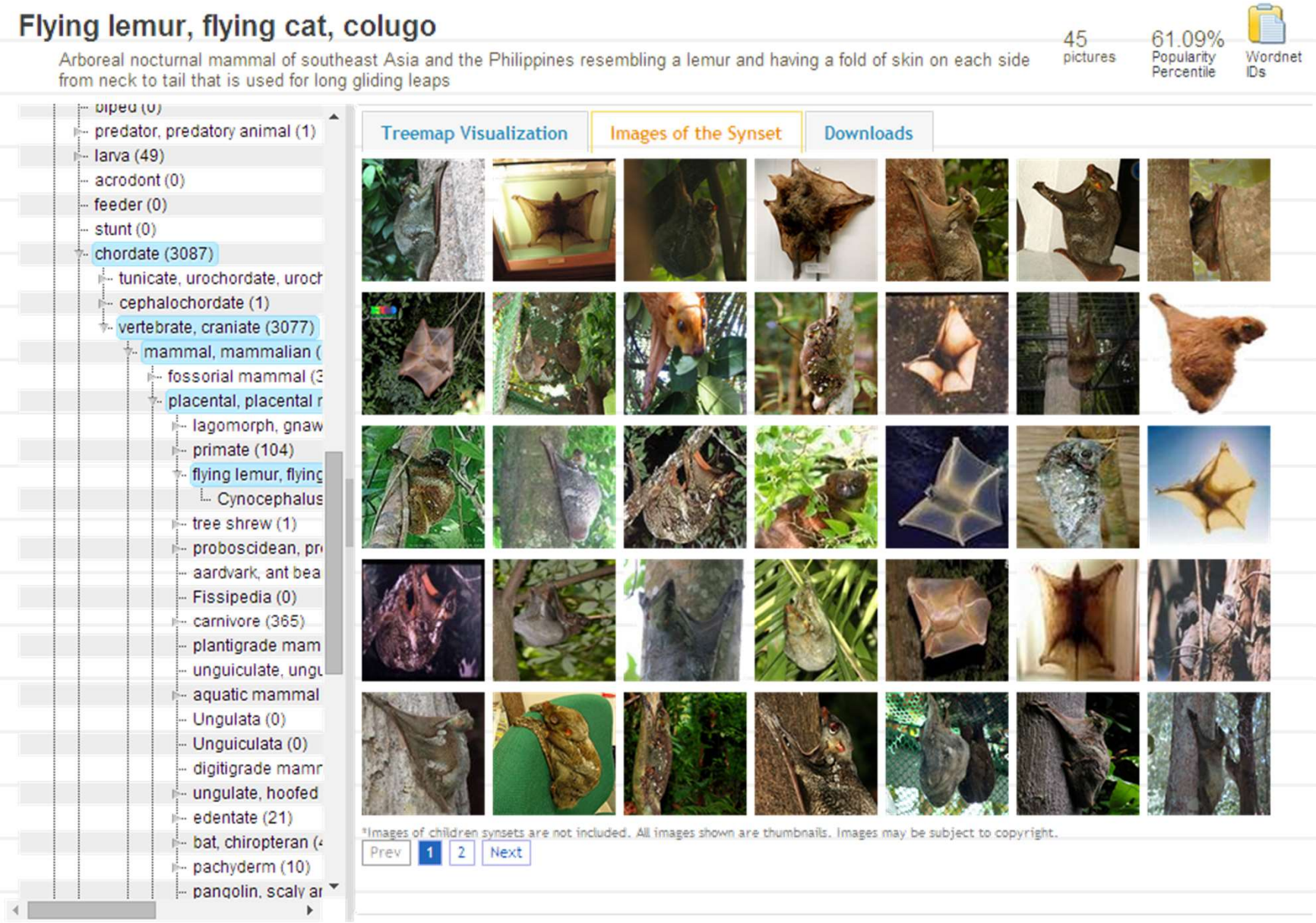
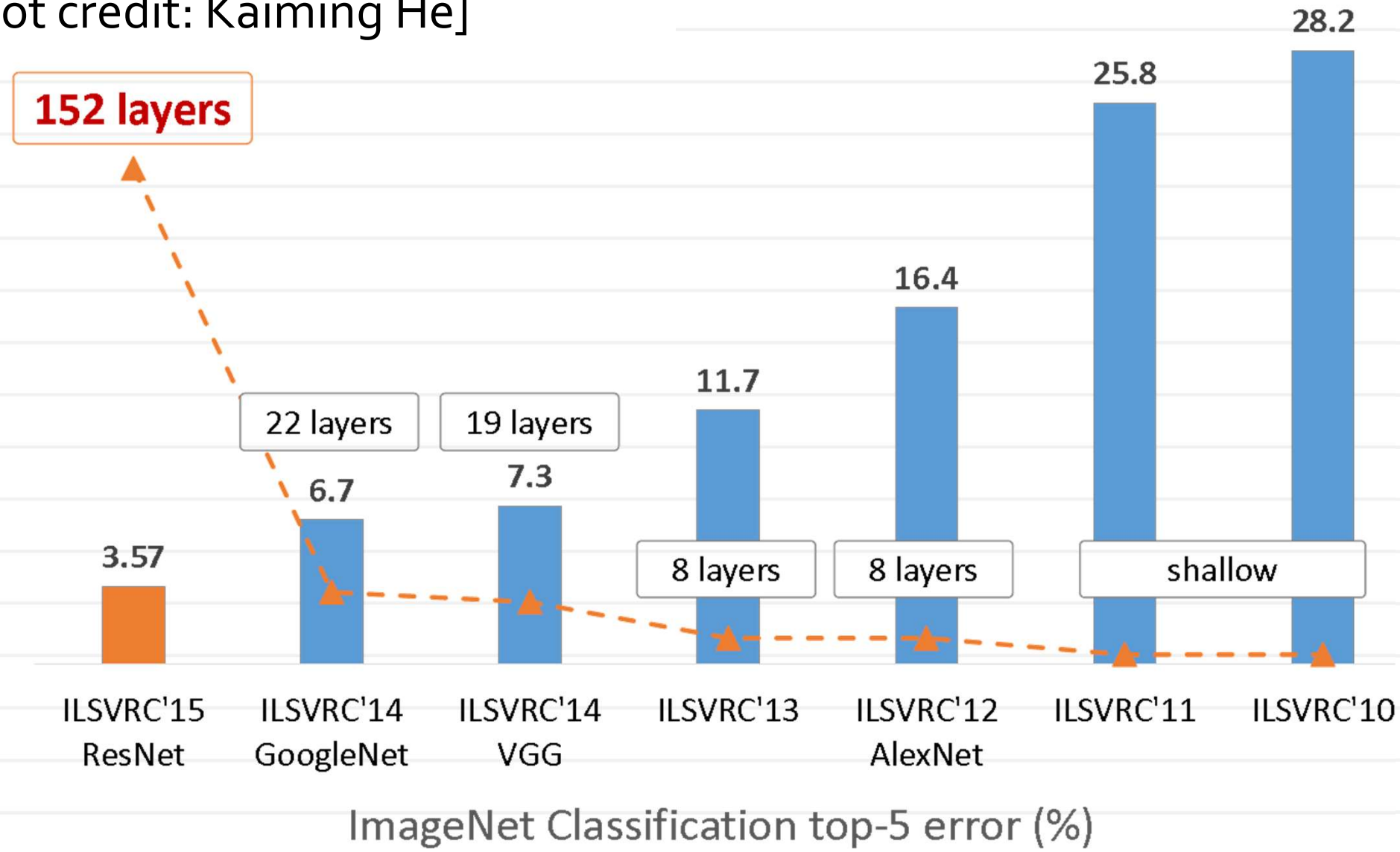

Soccer, association football
A football game in which two teams of 11 players try to kick or head a ball into the opponents' goal
1395 pictures
90.76% Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (1
natural object (1112)
sport, athletics (176)
rowing, row (2)
funambulism, tightrope walkin
judo (0)
blood sport (10)
gymnastics, gymnastic exercis
water sport, aquatics (19)
track and field (5)
outdoor sport, field sport (17)
contact sport (18)
boxing, pugilism, fisticuffs (
wrestling, rassling, grapplir
ice hockey, hockey, hockey
football, football game (5)
rugby, rugby football, rug
professional football (0)
American football, Amer
soccer, association foot
team sport (0)
racing (7)
athletic game (70)
riding, horseback riding, equita
archery (0)
cycling (3)
sledding (3)
skating (6)
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev 1 2 3 4 5 6 7 8 9 10 ... 39 40 Next

Image-net

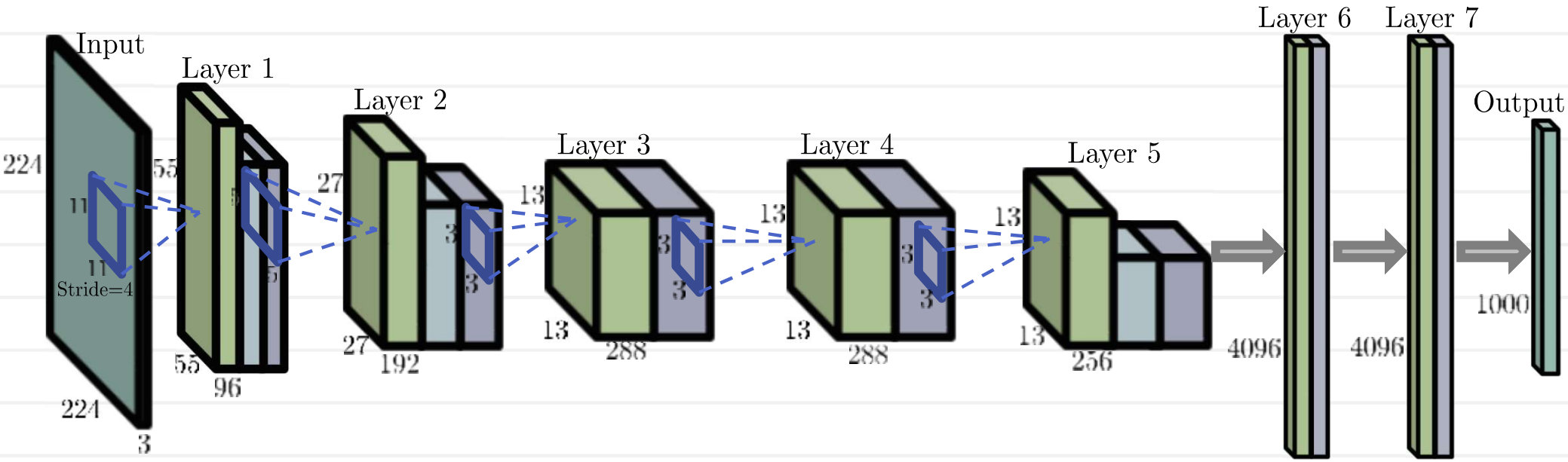

Flying lemur, flying cat, colugo
Arboreal nocturnal mammal of southeast Asia and the Philippines resembling a lemur and having a fold of skin on each side from neck to tail that is used for long gliding leaps
45 pictures
61.09% Popularity Percentile
Wordnet IDs
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, uroch
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (
fossorial mammal (3
placental, placental r
lagomorph, gnaw
primate (104)
flying lemur, flying
Cynocephalus
tree shrew (1)
proboscidean, pr
aardvark, ant bea
Fissipedia (0)
carnivore (365)
plantigrade mam
unguiculate, ungu
aquatic mammal
Ungulata (0)
Unguiculata (0)
digitigrade mamr
ungulate, hoofed
edentate (21)
bat, chiropteran (
pachyderm (10)
pangolin, scaly ar
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev
1
2
Next

Building the best network

[plot credit: Kaiming He]



AlexNet (2012)



- ..”CaffeNet” variant
- 5 conv layers (5x5,5x5,3x3,3x3,3x3)
- 60M parameters
- Learns in 3-5 days on a GPU
- Faster than subsequent architectures

[Krizhevsky et al. 2012]

VGGNet (2014)

- upto 16 conv. layers
- All filters are 3x3
- balances load between layers
- ~140M params
- Stagewise training
- The highest performance among chain-like models

[Simonyan & Zisserman, 2014]

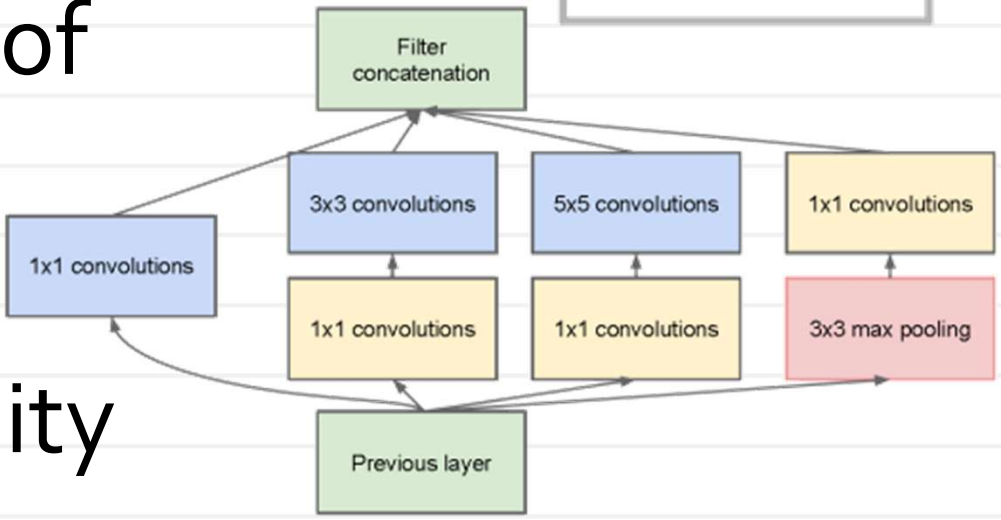
B	C	D	E
weight ayers	16 weight layers	16 weight layers	19 weight layers
(224 × 224 RGB image)			
nv3-64 **nv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool			
nv3-128 **nv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool			
nv3-256 nv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool			
nv3-512 nv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool			
nv3-512 nv3-512	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			

GoogleNet (2014)

Convolution
Pooling
Softmax
Other

"Large but fast":

- Keep large number of maps
- apply convolutions only to dimensionality reduced stacks



[Szegedy et al. 2014]

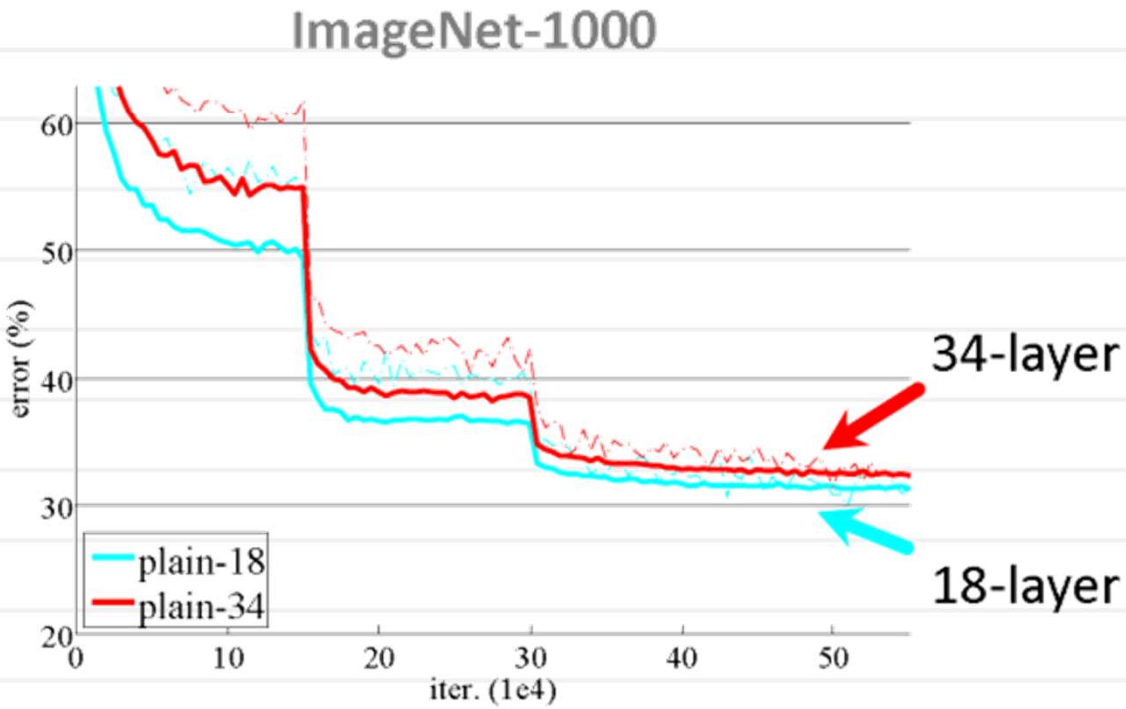
ResNet (2015)

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

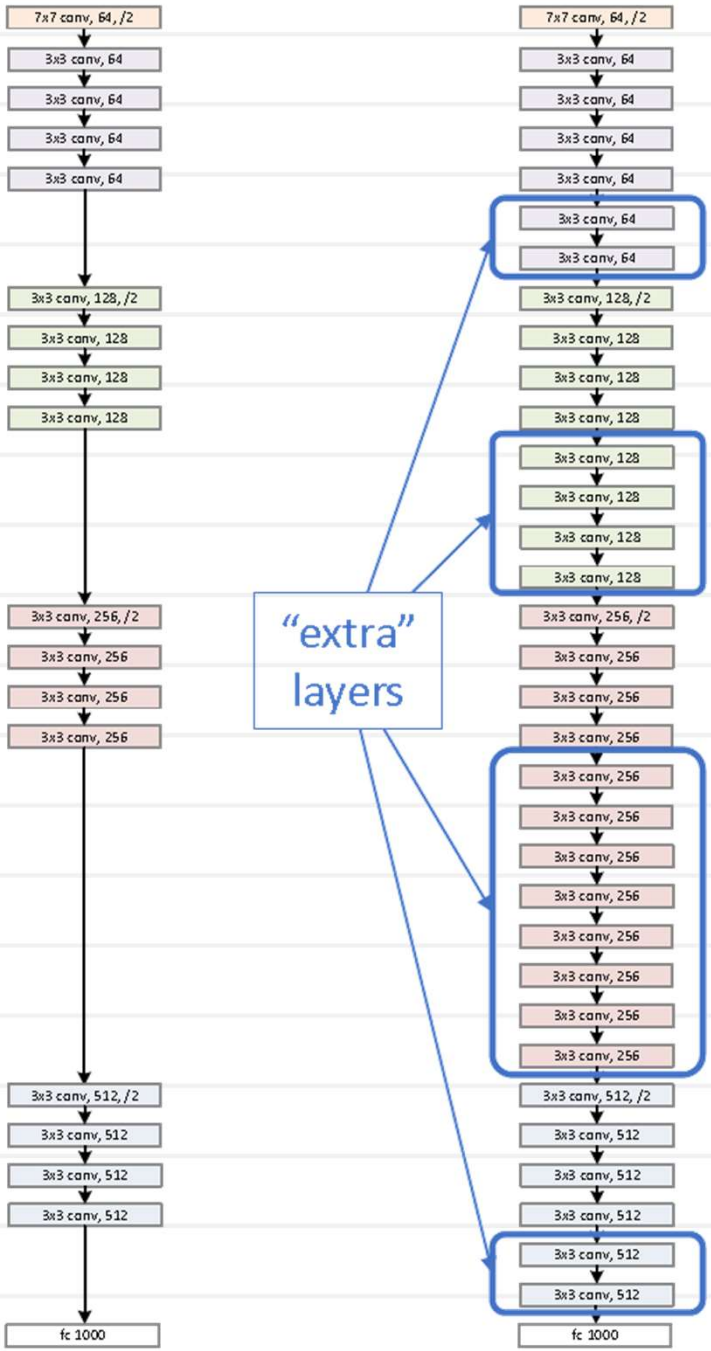
ResNet, 152 layers
(ILSVRC 2015)

Simply deepening does not work:



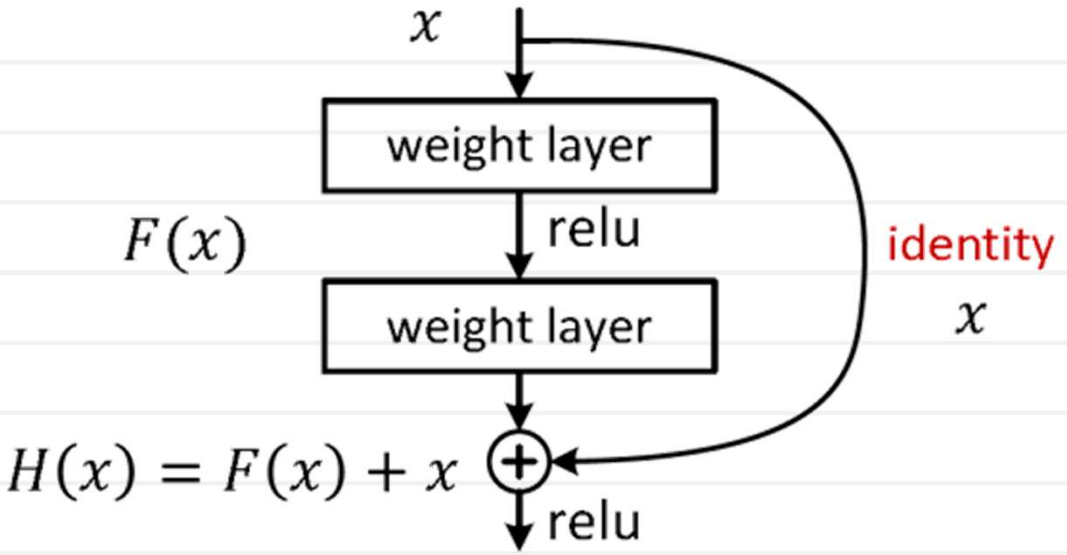
[He et al. 2015]

ResNet (2015)



Q: How to ensure that the training error does not go up?

A: shortcuts

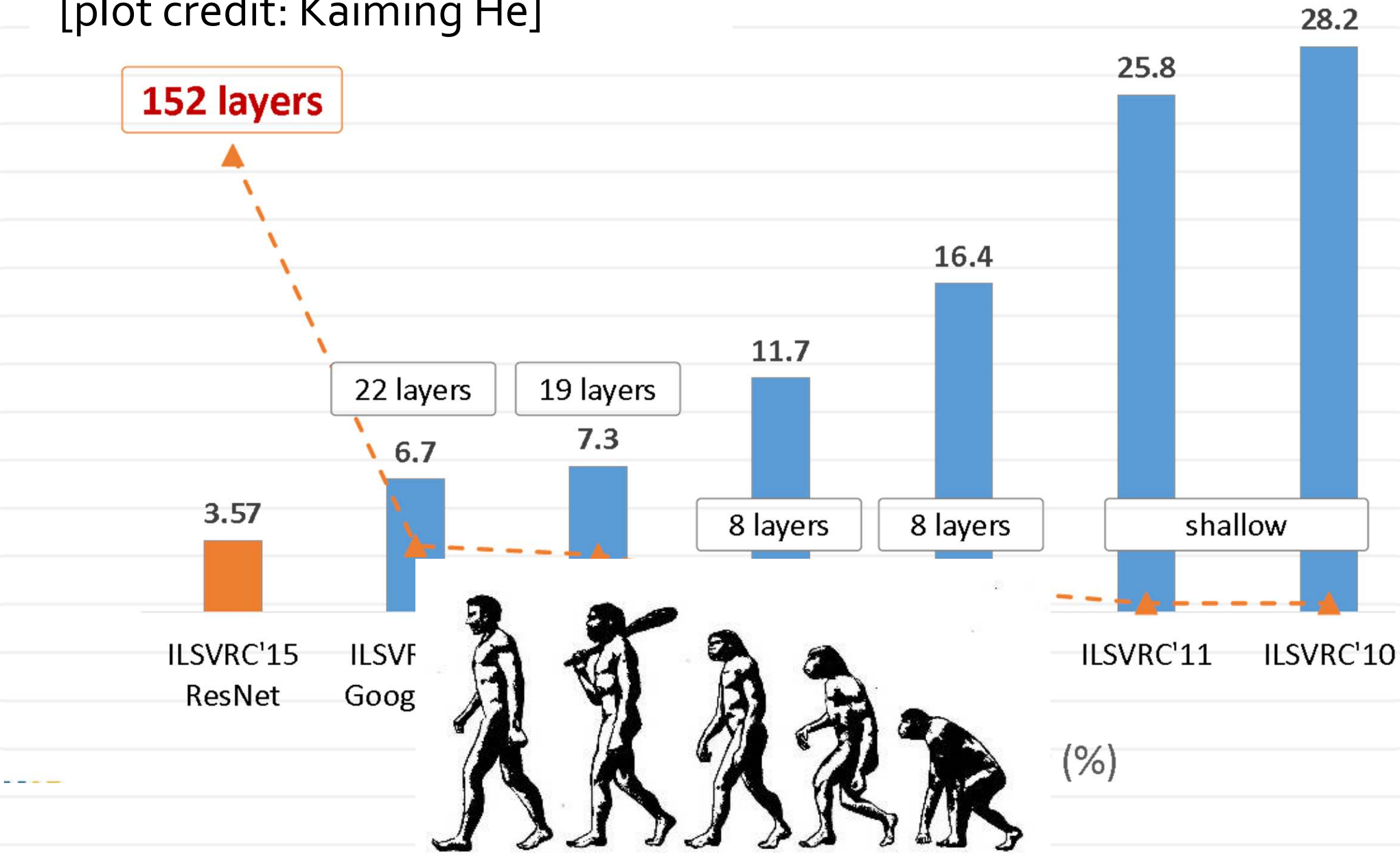


Learn F(x) instead of H(x)

[He et al. 2015]

Building the best network

[plot credit: Kaiming He]



Recap

- Convolutional networks are the most popular and influential model in modern deep learning
- Convolutional layer is a special type of a multiplicative one with greatly reduced number of parameters
- Different ways to compute convolutions exist

Bibliography

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard,Lawrence D. Jackel: Handwritten Digit Recognition with a Back-Propagation Network. NIPS 1989: 396-404

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012: 1106-1114

Joan Bruna, Wojciech Zaremba, Arthur Szlam, Yann LeCun: Spectral Networks and Locally Connected Networks on Graphs. ICLR 2014

Kumar Chellapilla and Sidd Puri and Patrice Simard, High Performance Convolutional Neural Networks for Document Processing

Bibliography

Karen Simonyan, Andrea Vedaldi, Andrew Zisserman:
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.CoRR abs/1312.6034 (2013)

Karen Simonyan, Andrew Zisserman:
Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR abs/1409.1556 (2014)

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan,Vincent Vanhoucke, Andrew Rabinovich:
Going deeper with convolutions. CVPR 2015: 1-9

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun:
Deep Residual Learning for Image Recognition. CoRR abs/1512.03385 (2015)