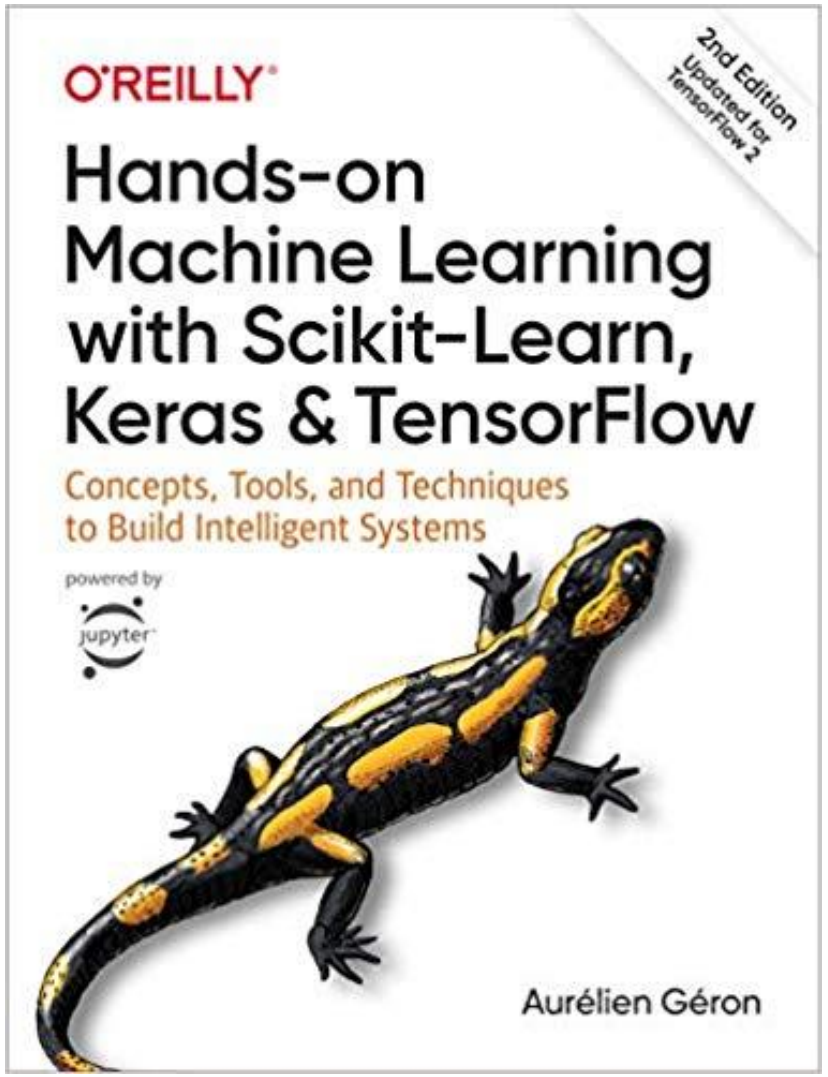
Convolutional Neural Networks 2 (Ch 14)

Wojtek Kowalczyk

wojtek@liacs.nl

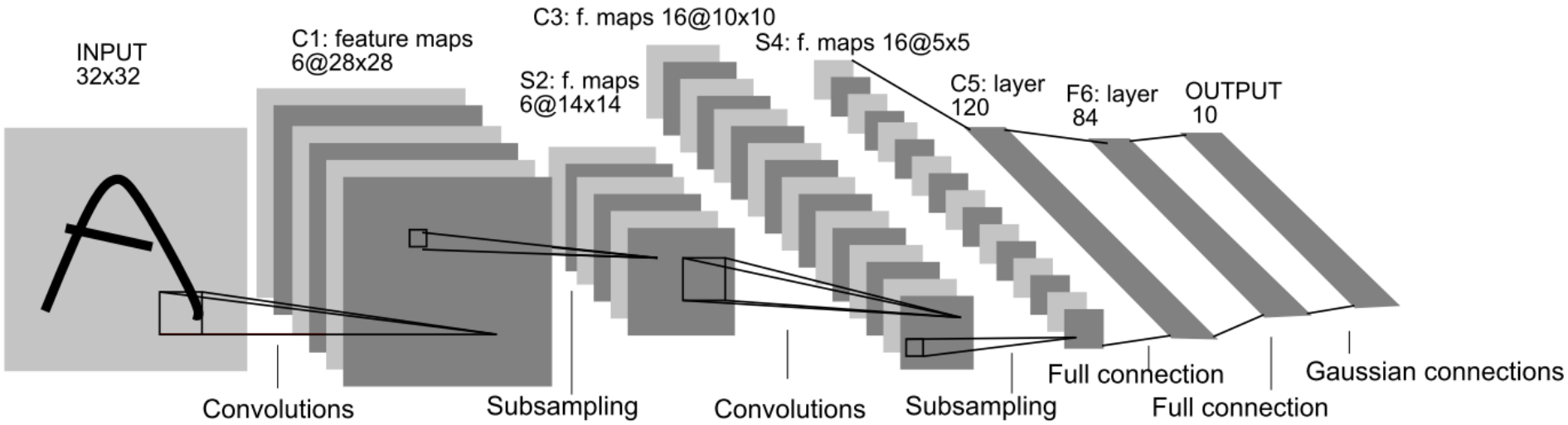
Convolutional Neural Networks 2 (Ch 14)



Agenda

- Vocabulary (rehearsal)
- Convolutional networks:
  - AlexNet
  - ResNet
  - GoogleNet

LeNet5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

Vocabulary

- Vector, matrix, tensor (a multi-dimensional array)
- Convolution
- Convolutional filter (convolutional kernel)
- Convolutional layer
- Stride
- Padding (e.g., *same*, *valid*)
- Feature map
- Pooling (max pooling, mean pooling, subsampling)

Tensors: examples

- 0-d: a single number, 3.14
- 1-d: a tuple of numbers (1,4,6,2)
- 2-d: e.g. a gray level image
- 3-d: e.g., an RGB image,
- 4-d: e.g. a stack of RGB images (movie)
- 5-d: e.g., real-time 3-D, color tomography
- …

Convolution (informal)

- An operation that takes as input a tensor and applies a “local **convolution** operation” (filter, kernel, c. matrix, c. tensor) to “all fragments” of the input
- **Convolutional layer**: a layer of neurons that perform the same operation on fragments of the input
- **Feature Map**: the result of applying convolutional layer to data
- **Stride**: “the step size when moving a filter over the input”. “Reduces resolution”, “shrinks the image”, …
- **Padding**: artificially increasing the size of the input (e.g., by zeros, mirror reflections, …) to preserve the original input size in the feature map
- Padding =‘**same**’: add zeros when needed!
- Padding = ‘**valid**’: accept the loss of some input; don’t pad your input!
- …

padding.

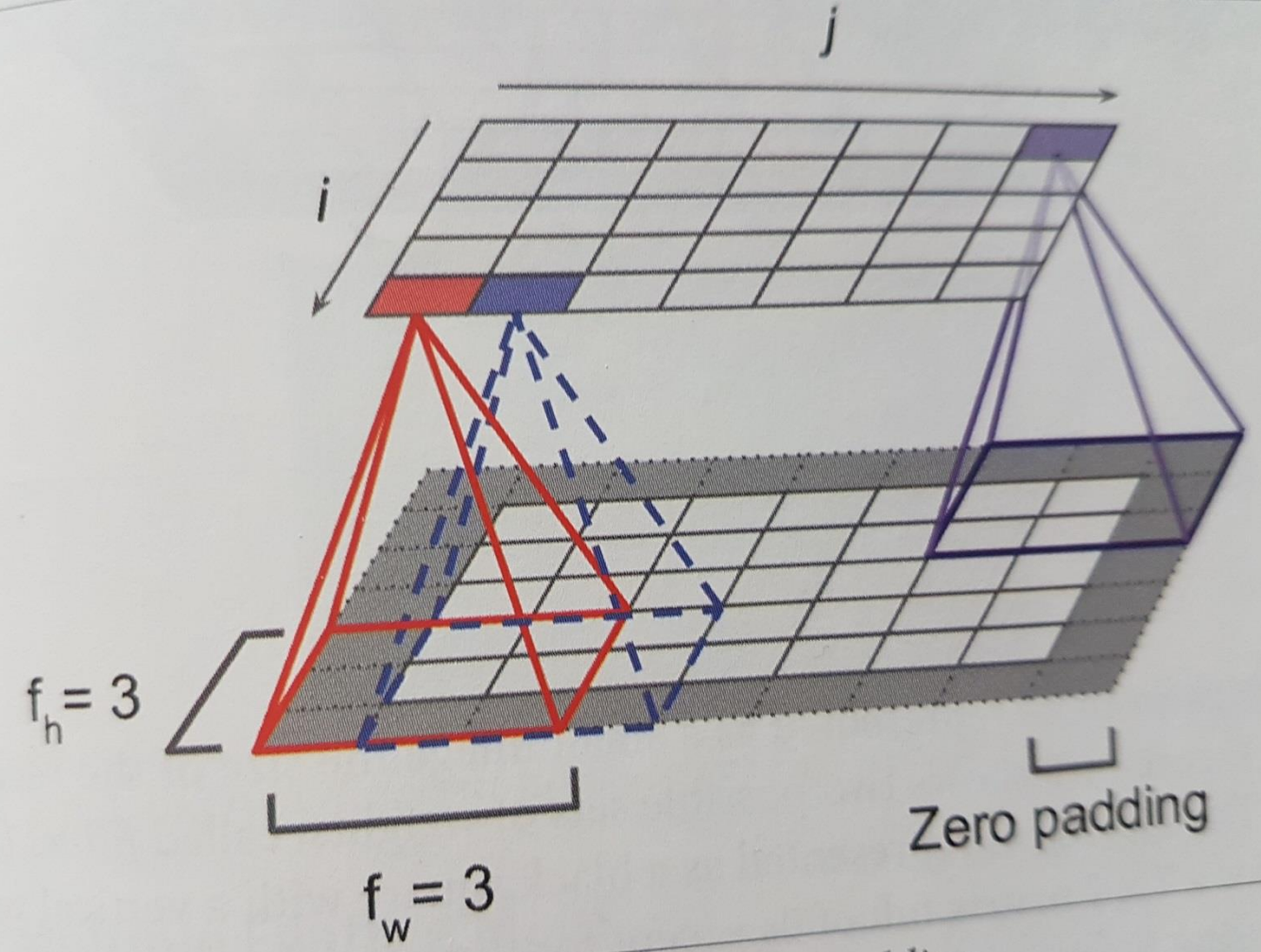


Figure 14-3. *Connections between layers and zero padding*

much smaller layer by sp

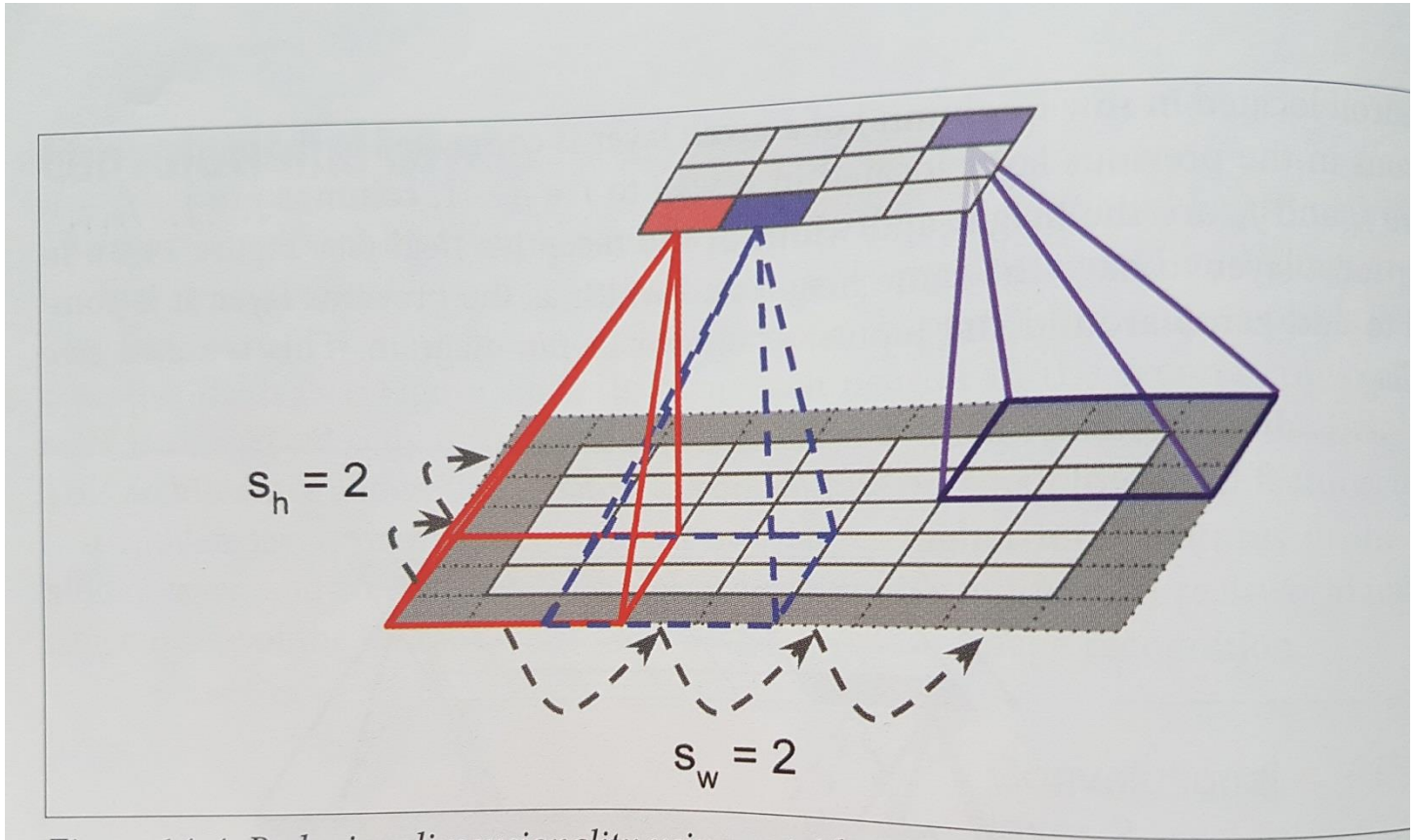


Figure 14-4. Reducing dimensionality using a stride of 2

Filters

A neuron's weights can be represented as a small image the

within valid positions inside the input (it does not go out of bounds), hence the name *valid*.

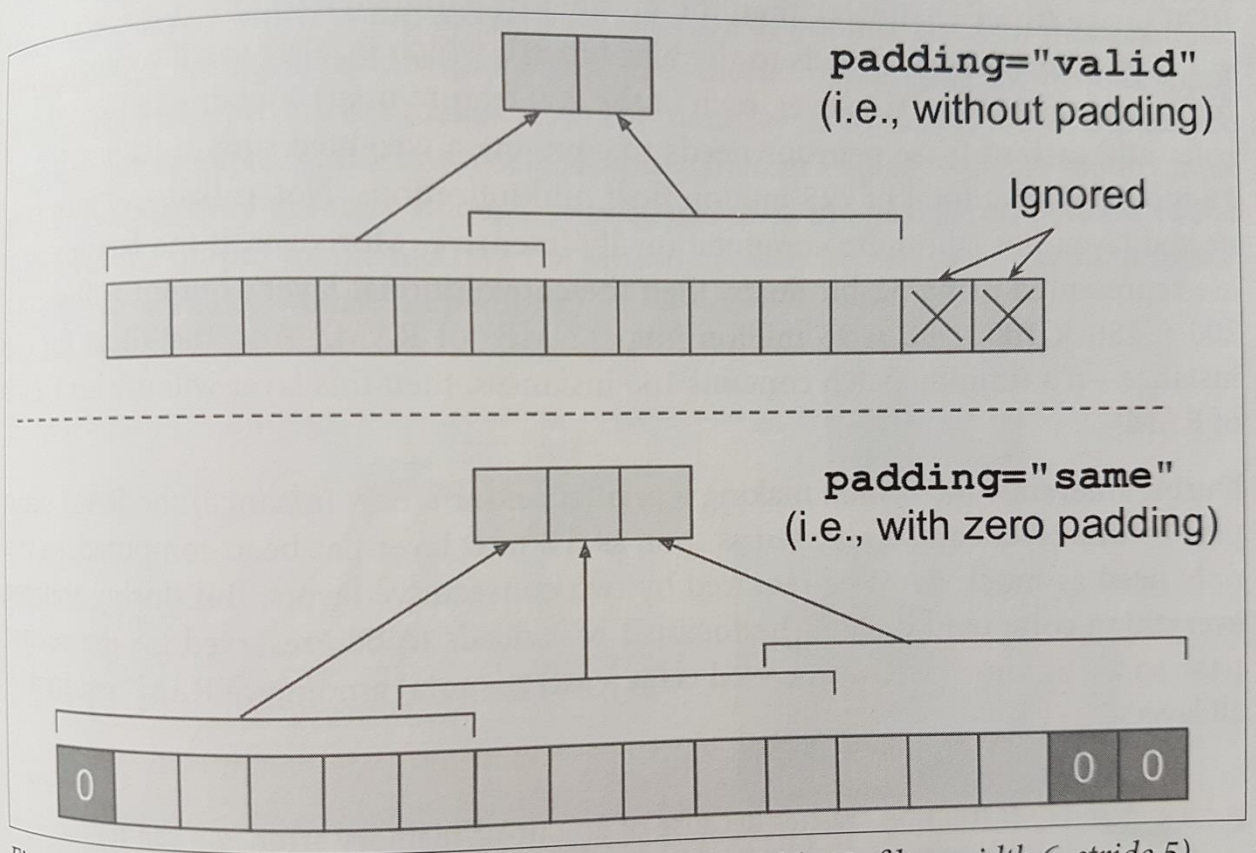


Figure 14-7. Padding="same" or "valid" (with input width 13, filter width 6, stride 5)

In this example we manually defined the filters, but in a real CNN you would nor-

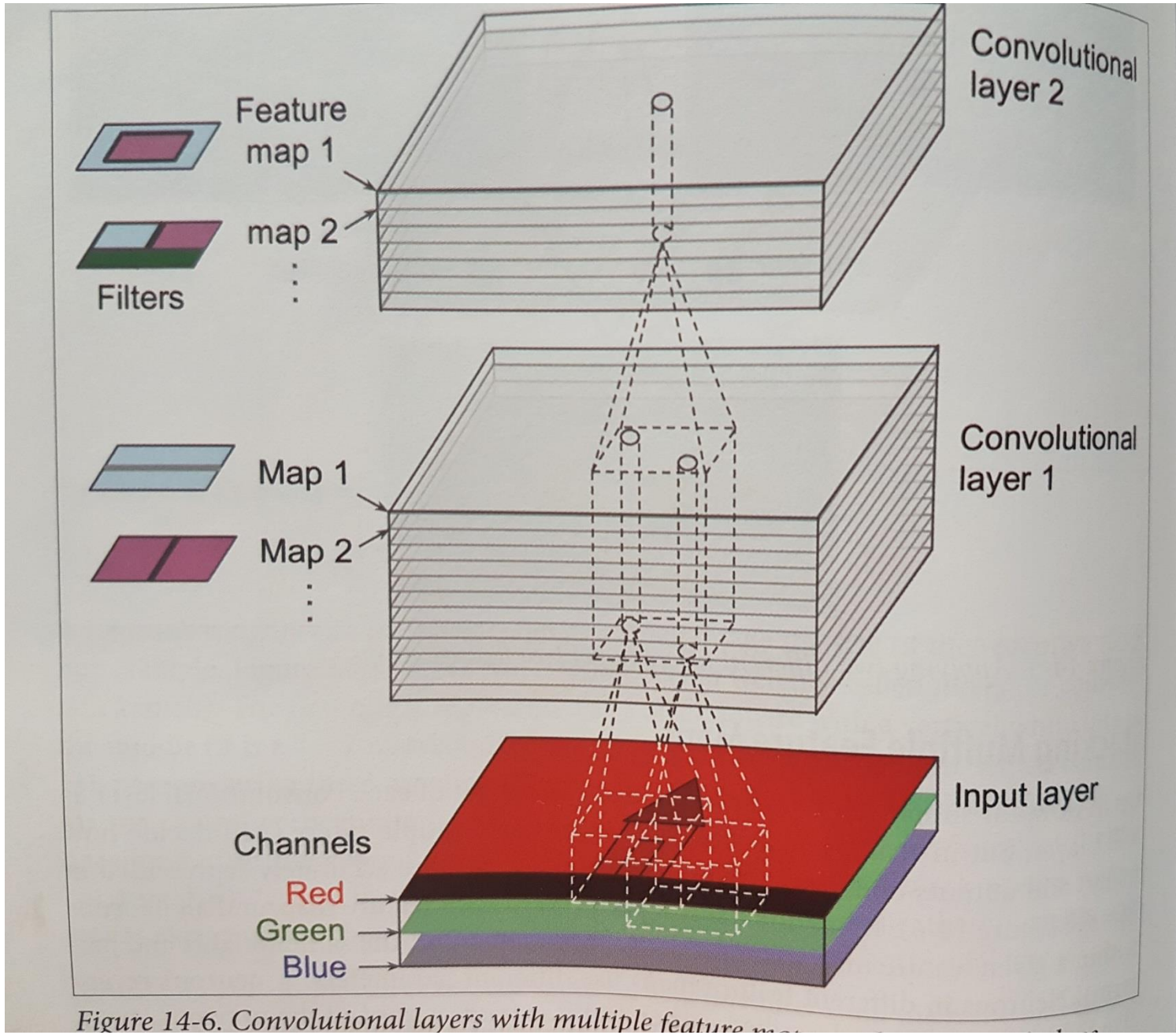


Figure 14-6. Convolutional layers with multiple feature

Pooling

- Used to reduce the size of data by “subsampling”; “reducing image resolution”
  - Max pooling
  - Mean poling
  - “weighted mean” (LeNet5)
  - “in depth”-pooling (over several layers)
  - Local Response Normalization (with 4 hyperparameters) Motivation: when we consider a single “pixel” in a stack of layers (a vector or numbers) we want to “amplify strong” and “weaken weak neighboring” values. The formula 14.2 does the trick! (Is it pooling? NO!)

From LeNet5 to ImageNet (2010/2012)

ImageNet

- 15M images
- 22K categories
- Images collected from Web
- RGB Images
- Variable-resolution
- Human labelers (Amazon's Mechanical Turk crowd-sourcing)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)

- 1K categories
- 1.2M training images (~1000 per category)
- 50,000 validation images
- 150,000 testing images

ImageNet (study slides 28-40)

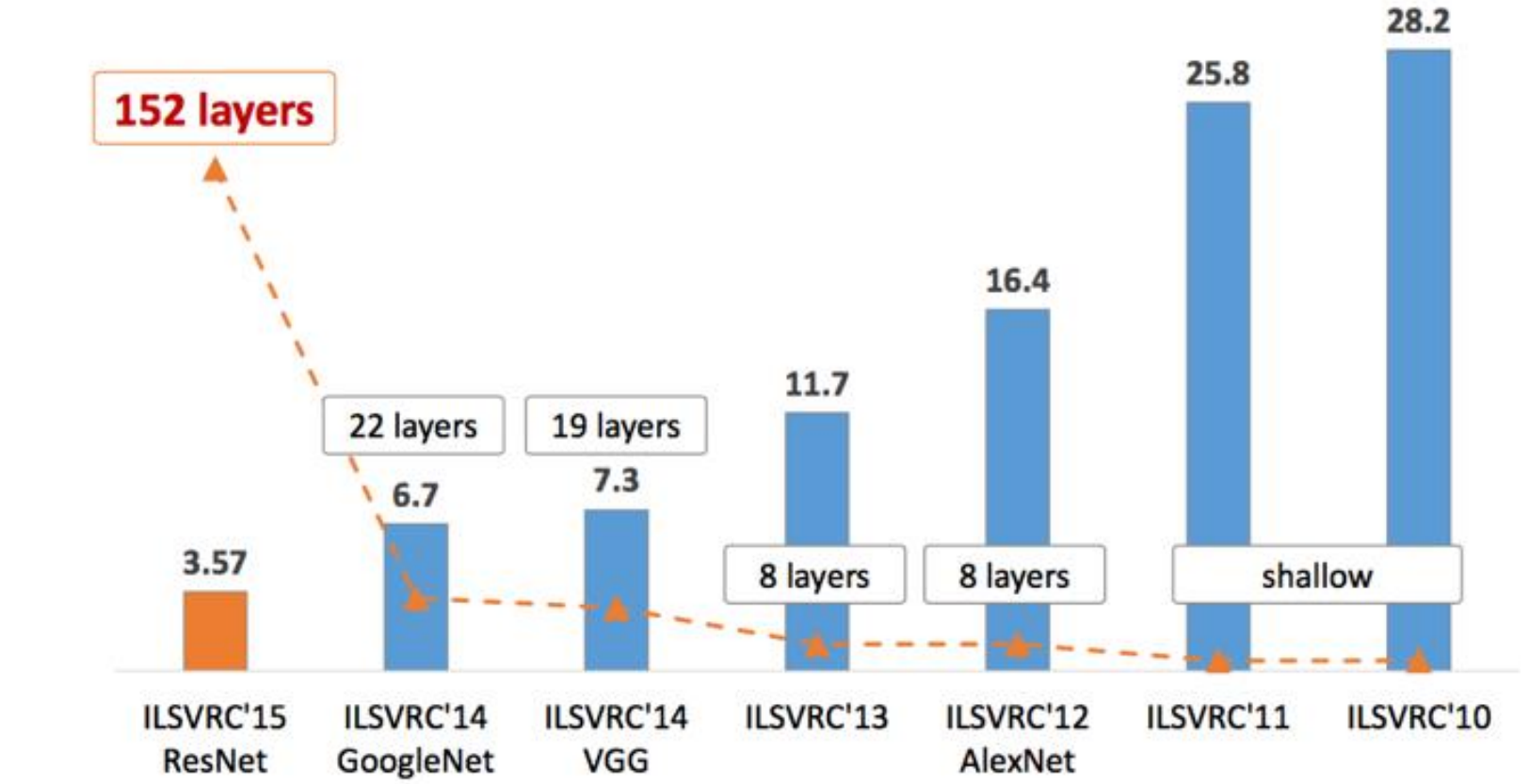
- ILSVRC-2010 test set

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

- ILSVRC-2012 test set

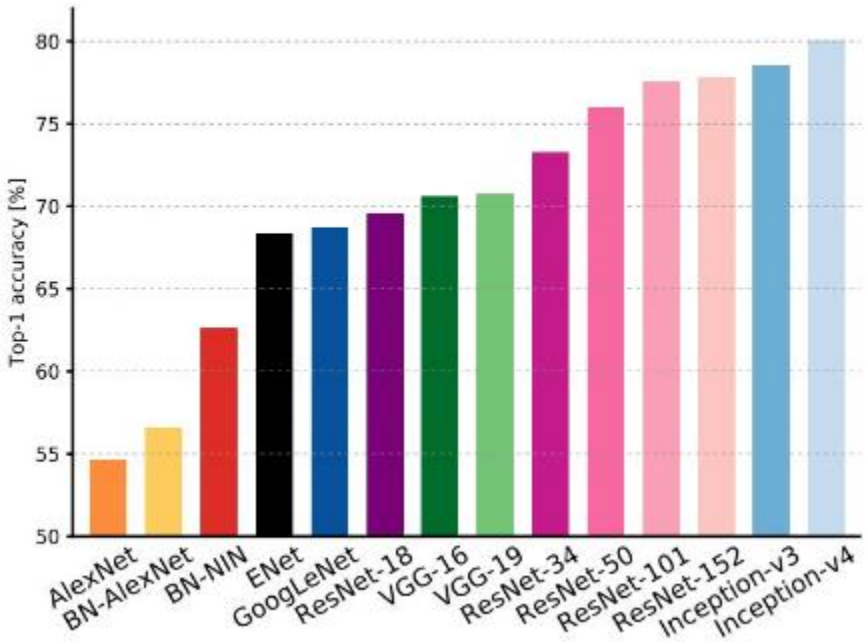
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

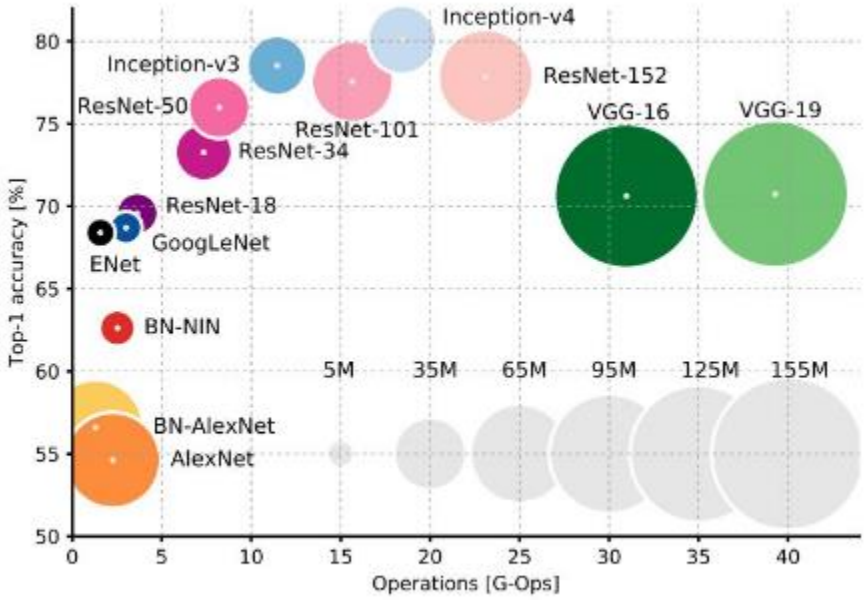
CNNs: progress



https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

CNNs: progress





An Analysis of Deep Neural Network Models for Practical Applications, 2017.

https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

CNNs: progress

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Key Ideas (see slides Tugce, Kyunghee)

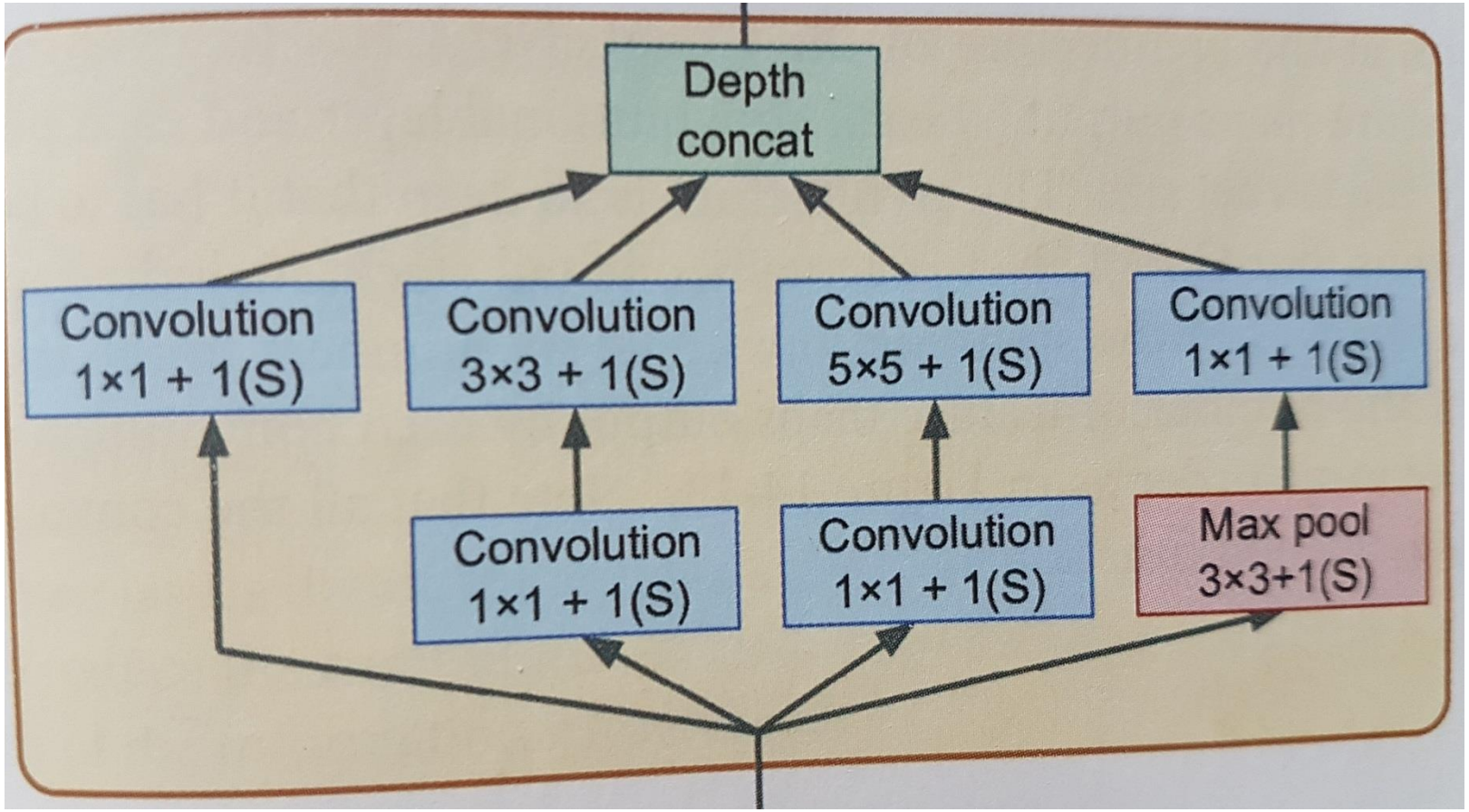
- The “classical” CNN architecture:
  Input ->(CNN->Pool)->FullyConn->FullyConn-> SoftMax*
- ReLU instead of sigmoid activations
- Local Response/Contrast Normalization
- Overlapping Pooling
  - Data Augmentation:
  - 224x224 patches (from 256x256)+horizontal reflections (~1000x)
  - Variations of RGB intensities
- Dropout

ResNet (see the “Tutorial” slides)

- “Classical CNN” + Shortcut connections
- Free flow of gradients during backpropagation
  => very deep networks (up to 1000’s layers)
- Well understood
- Top accuracy
- Common architecture for “deep CNNs”
  (vision, games, Reinforcement Learning, …)

GoogleNet

- 2014, winner ImageNet challenge, 7% better than the rest
- Main trick: **the Inception Module**



GoogleNet

- 2014, winner ImageNet challenge, 7% better than the rest
- Main trick: **the Inception Module**

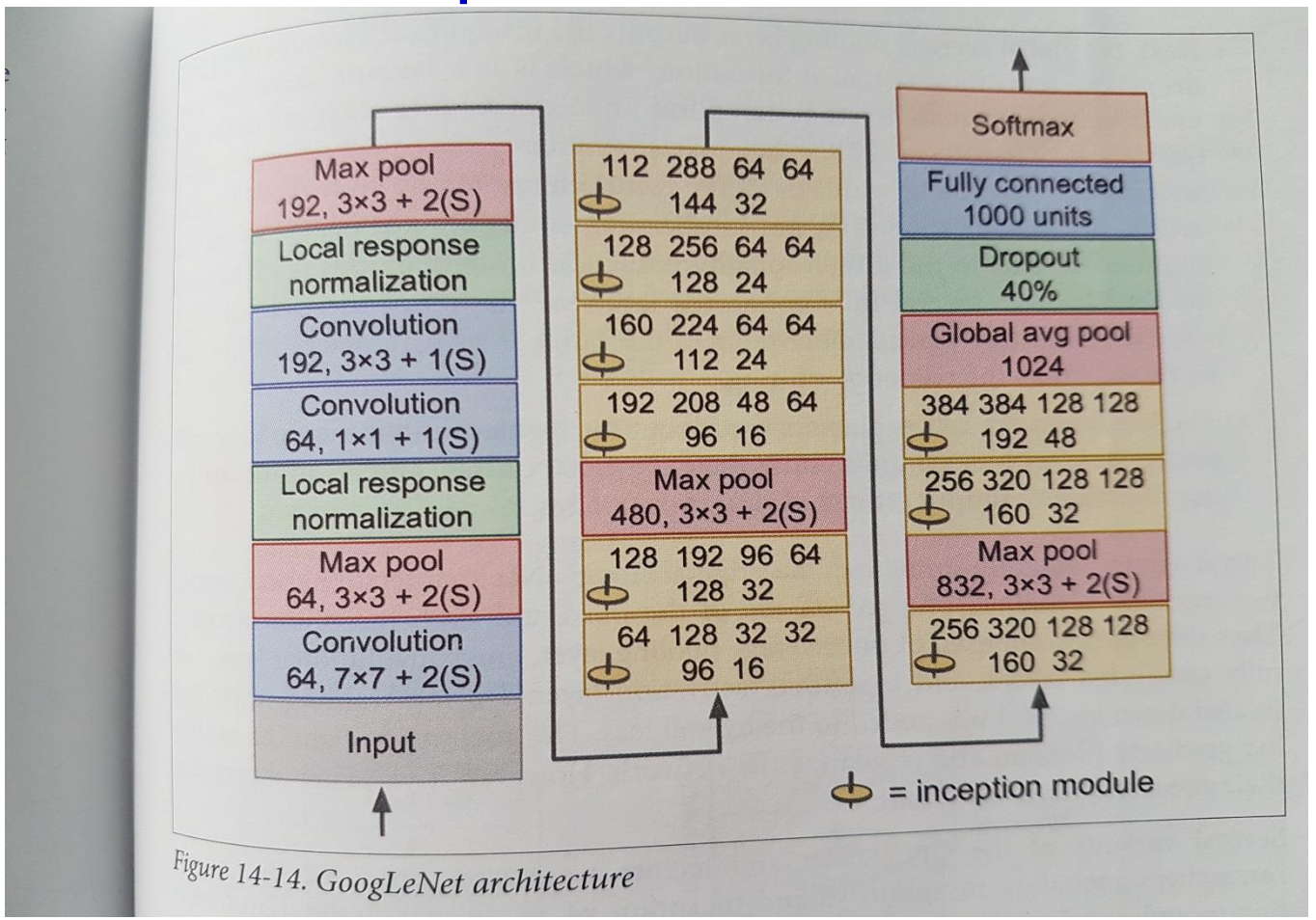
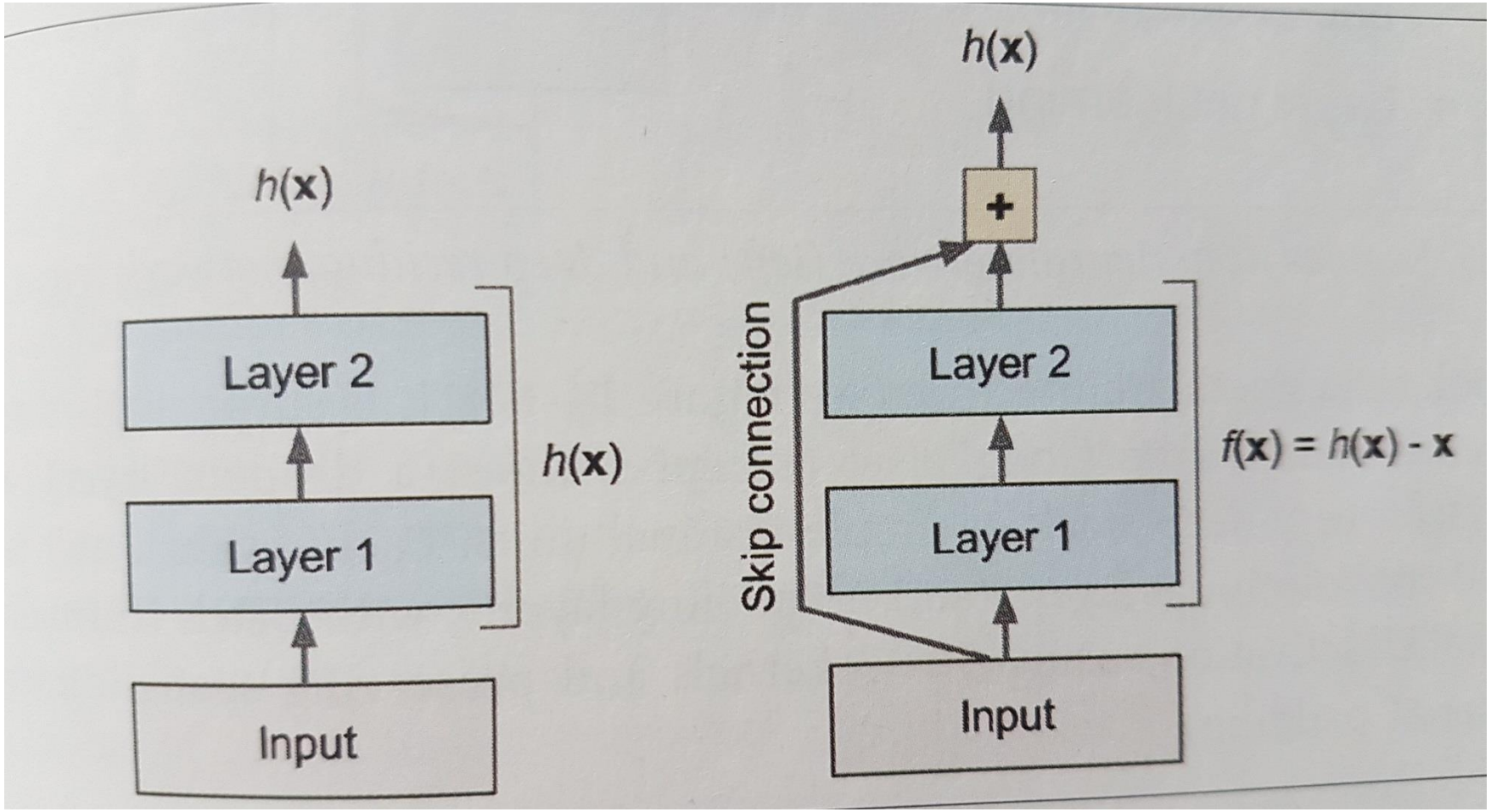


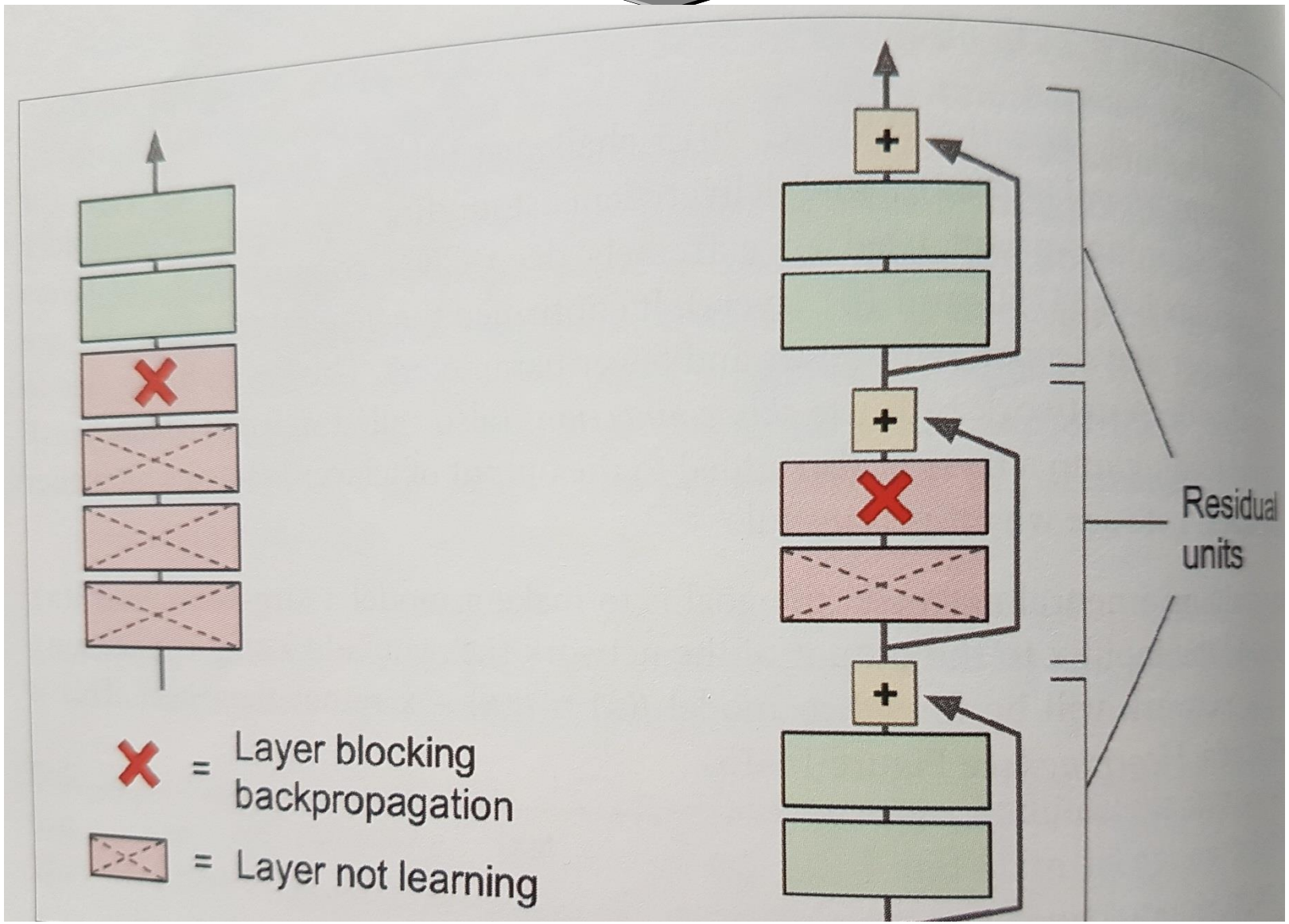
Figure 14-14. GoogLeNet architecture

ResNet

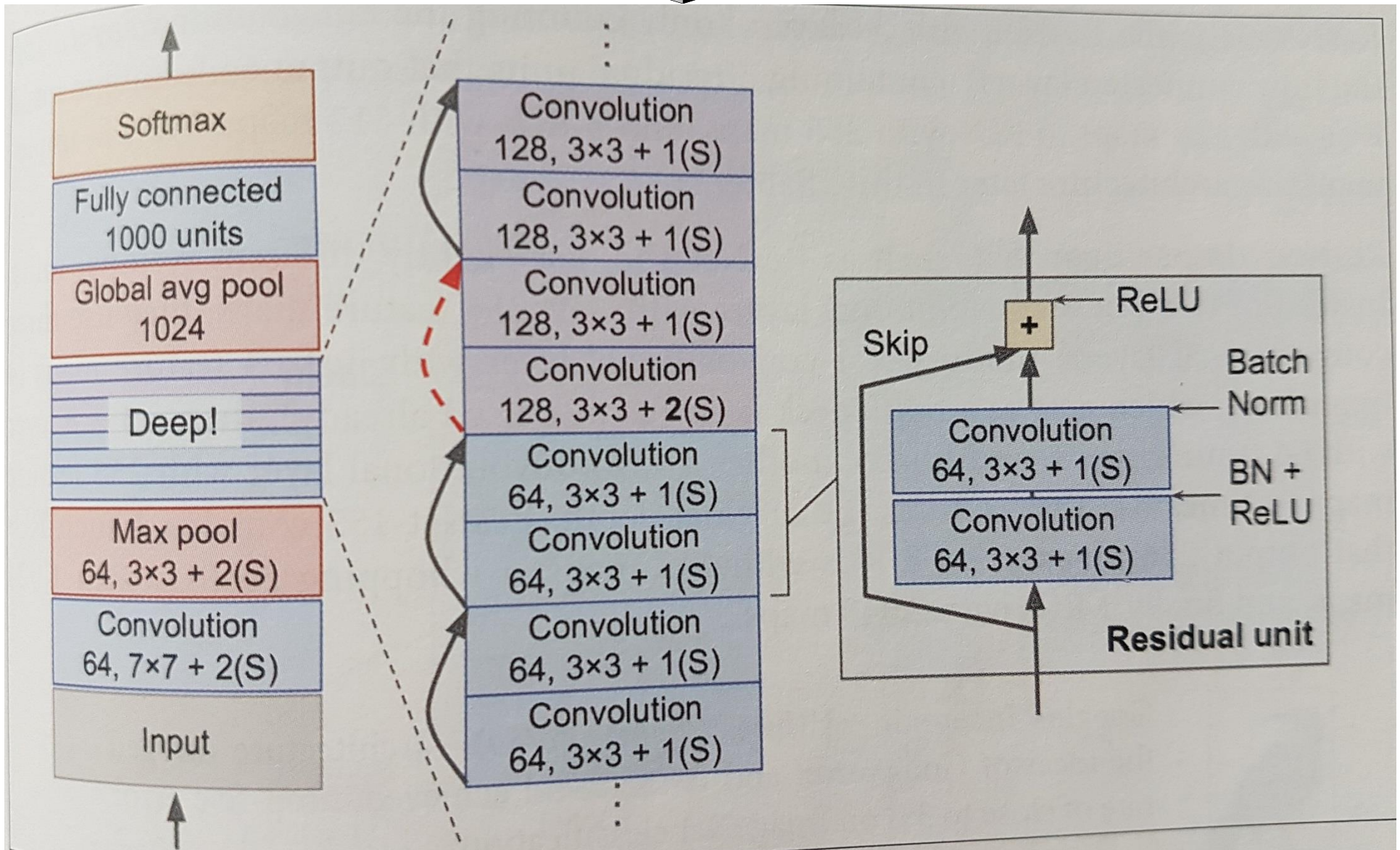
- 2015, winner ImageNet challenge, top5-error < 3.6%
- Main trick: **the shortcut connections!**



ResNet


= Layer blocking backpropagation
= Layer not learning
Residual units

ResNet


Softmax
Fully connected
1000 units
Global avg pool
1024
Deep!
Max pool
64, 3×3 + 2(S)
Convolution
64, 7×7 + 2(S)
Input
Convolution
128, 3×3 + 1(S)
Convolution
128, 3×3 + 1(S)
Convolution
128, 3×3 + 1(S)
Convolution
128, 3×3 + 2(S)
Convolution
64, 3×3 + 1(S)
Convolution
64, 3×3 + 1(S)
Convolution
64, 3×3 + 1(S)
Convolution
64, 3×3 + 1(S)
+
ReLU
Skip
Batch
Norm
Convolution
64, 3×3 + 1(S)
BN +
ReLU
Convolution
64, 3×3 + 1(S)
Residual unit