UNIVERSITÄT BONN

Introduction to Convolutional Neural Networks

Elena Trunz

February 24, 2025

Visual Computing Group, University of Bonn

Overview

Why Convolution?

Convolution operation in machine learning

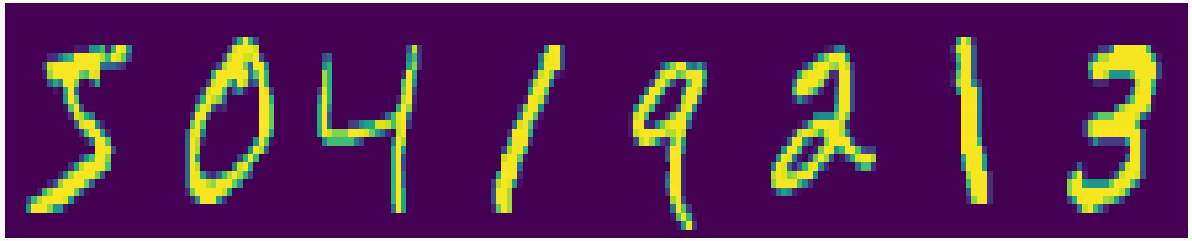
Understanding convolution

Convolutional neural networks

Deep convolutional neural networks

Why Convolution?

Invariance in the network structure



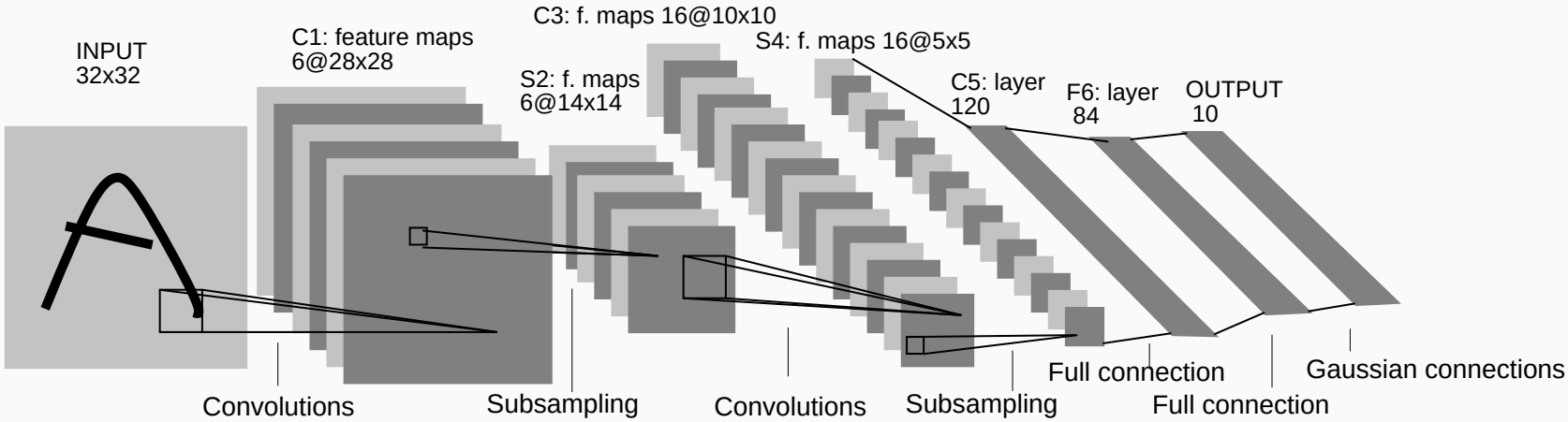
- Consider the task of recognizing handwritten digits:
  - Input image: set of pixel intensities
  - Desired output: posterior probability over the ten digits
- A system recognizing digits should be invariant to
  - Translations
  - Scaling
  - (Small) rotations
  - Some elastic deformations

Fully connected network?

- Given enough training data a fully connected network solves the task.
- However, we can do better:
  - Pixels near to each other are more correlated to each other than pixels far apart.
  - Local features which are useful in one region of the image are likely to be useful elsewhere as well, e.g. for detecting a translated object.

The invention of convolutional neural networks

In 1989 Yann LeCun proposed to build invariance properties into the structure of a neural network.



Convolution operation in machine learning

Defining discrete convolution

- For two one-dimensional signals $x \in \mathbb{R}^T$ and $k \in \mathbb{R}^T$, convolution is defined as

$$s(t) = (x * k)(t) = \sum_{m=0}^{T} x(m)k(t-m), \tag{1}$$

  where t denote the position in the output sequence.

- In 2D, we require a kernel matrix $K \in \mathbb{R}^{O,P}$ and a image matrix $I \in \mathbb{K}^{N,M}$

$$S(i,j) = (K * I)(i,j) = \sum_{m}^{M} \sum_{n}^{N} I(i-m, j-n)K(n,m) \tag{2}$$

  where (i,j) denote the position in the output image.

Defining cross-correlation

- Cross-correlation is convolution without flipping the kernel:

$$S(i,j) = (K * I)(i,j) = \sum_{m}^{M} \sum_{n}^{N} I(i+m, j+n) K(m,n) \quad (3)$$

- Many machine-learning libraries implement cross-correlation and call it convolution.
- In this course we will follow their example.

Convolution

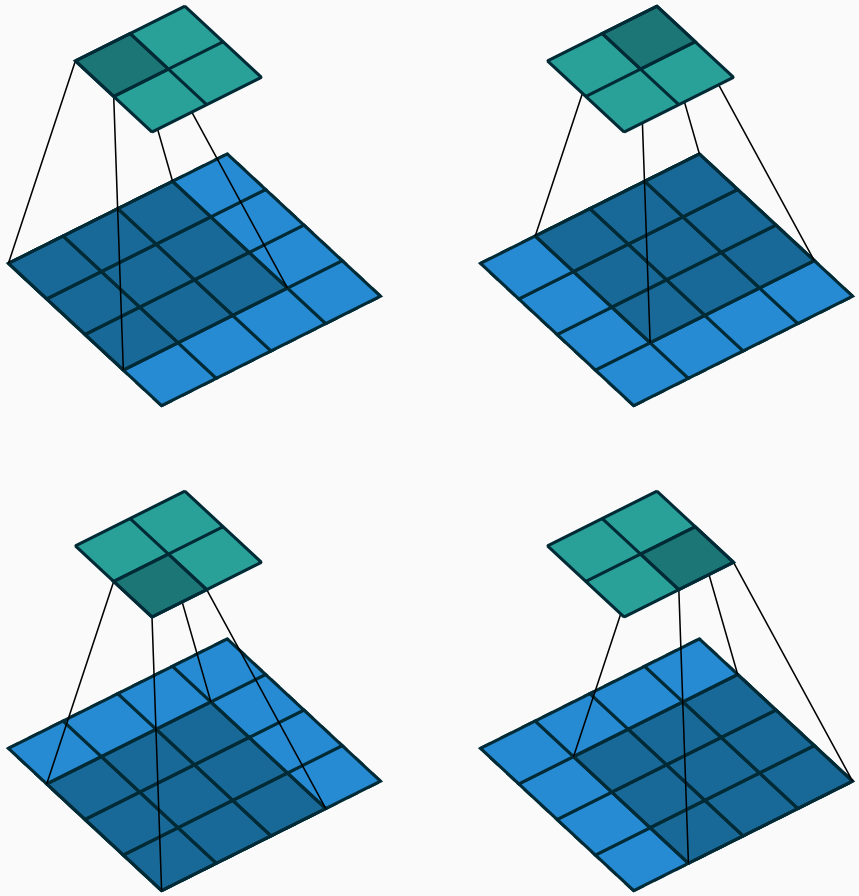


Illustration of the convolution operation [DV16].

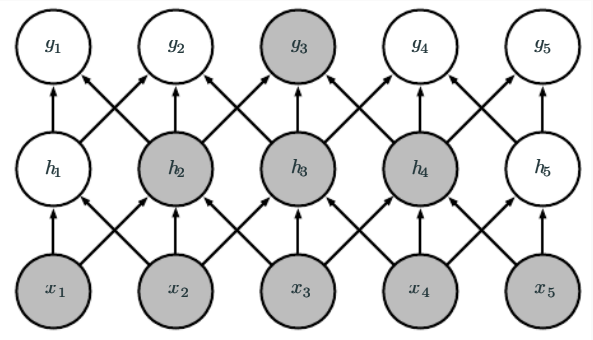
Advantages of convolutions

The convolution operation allows us to do this for inputs of varying sizes in a convenient way, and affords advantages of

1. local receptive fields (sparse connectivity limited in range vs. fully connected)
2. parameter sharing (allows for detection of same features all over the image)
3. equivariant representations and subsampling (reduces information hierarchically)

Sparse connectivity and interaction

- In CNNs, the kernel or filter is often much smaller than the input size. This allows us to process images of varying sizes. For an input image of millions of pixels we can still detect meaningful features which are local, such as edges.
- A small kernel does not mean that the area of interaction is limited. For a deep network, the units may *indirectly* interact with a larger portion of the input as information propagates up the layers:



Unit g_3 indirectly interacts with the whole input [GBC16]

Parameter sharing

- Standard feed-forward network has unit-specific weights.
- Each weight is used only once when computing the activations for subsequent layers.
- In a CNN, the weights of each kernel is applied to every position of the input.
- This *parameter sharing* reduces the amount of storage per model.

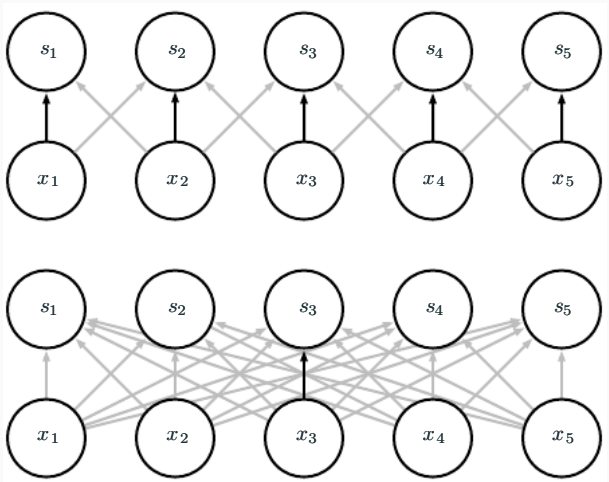


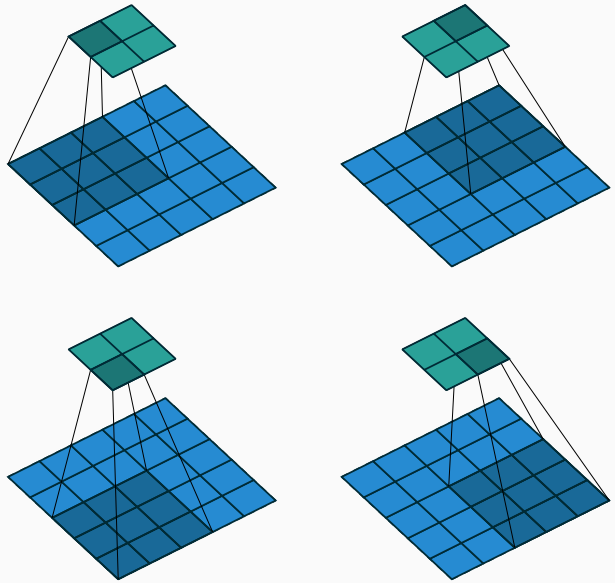
Image was taken from [GBC16]

Translation equivariance

- An *equivariant* function is one where for some given change in the input, the output changes in the same way.
- With images, convolution creates a 2-D map of where certain features appear in the input. If we move the object in the input, its representation will move the same amount in the output.

Strided convolution

- To save computational expense, we can skip over some positions of the kernel, i.e. not extract our feature response as finely.



Stride two convolutions [DV16].

$\rightarrow$ Down-sampling

Padded convolution

- We can control the size of convolution inputs and outputs through padding.
- Without padding, the spatial extent of the network would shrink, with the rate of shrinking directly proportional to the size of the filter or kernel.

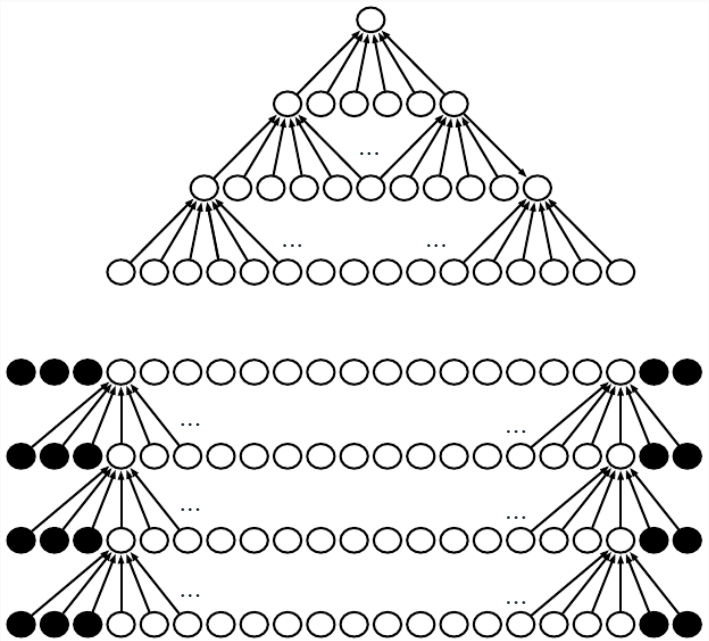


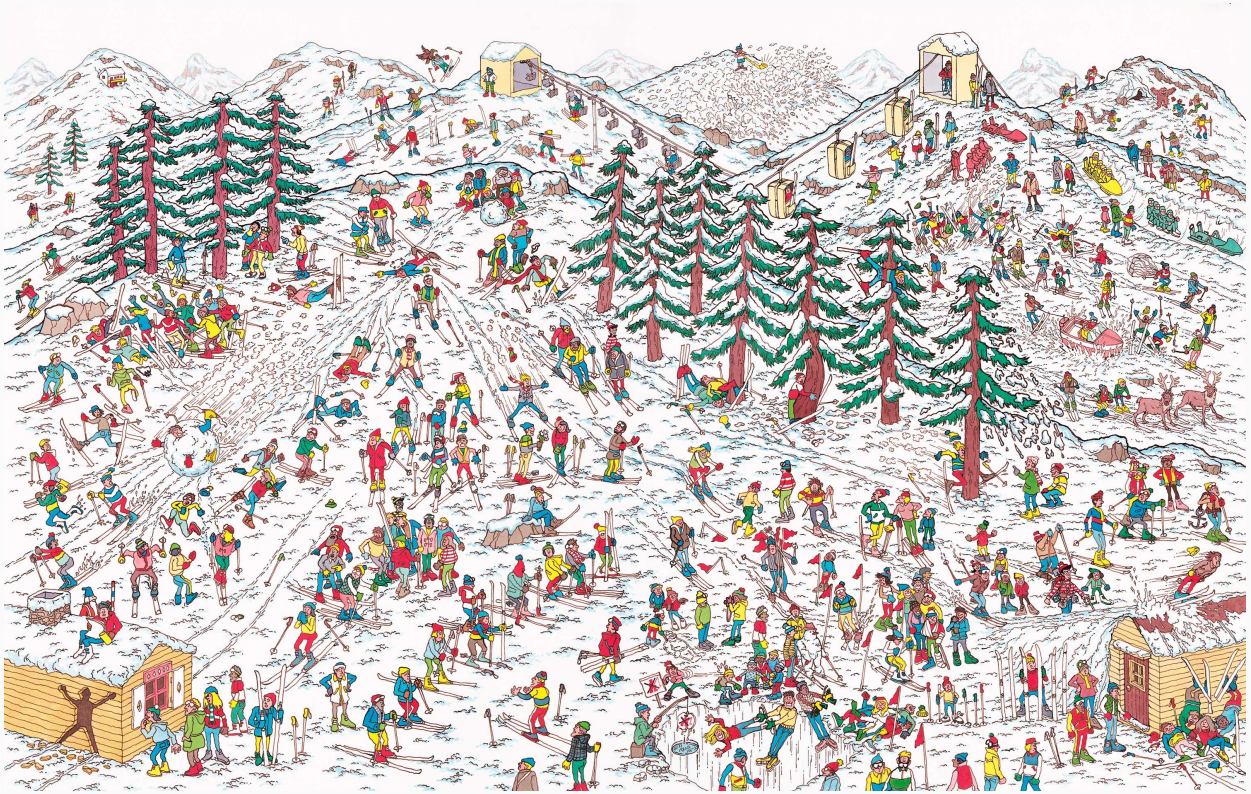
Image was taken from [GBC16]

Summary

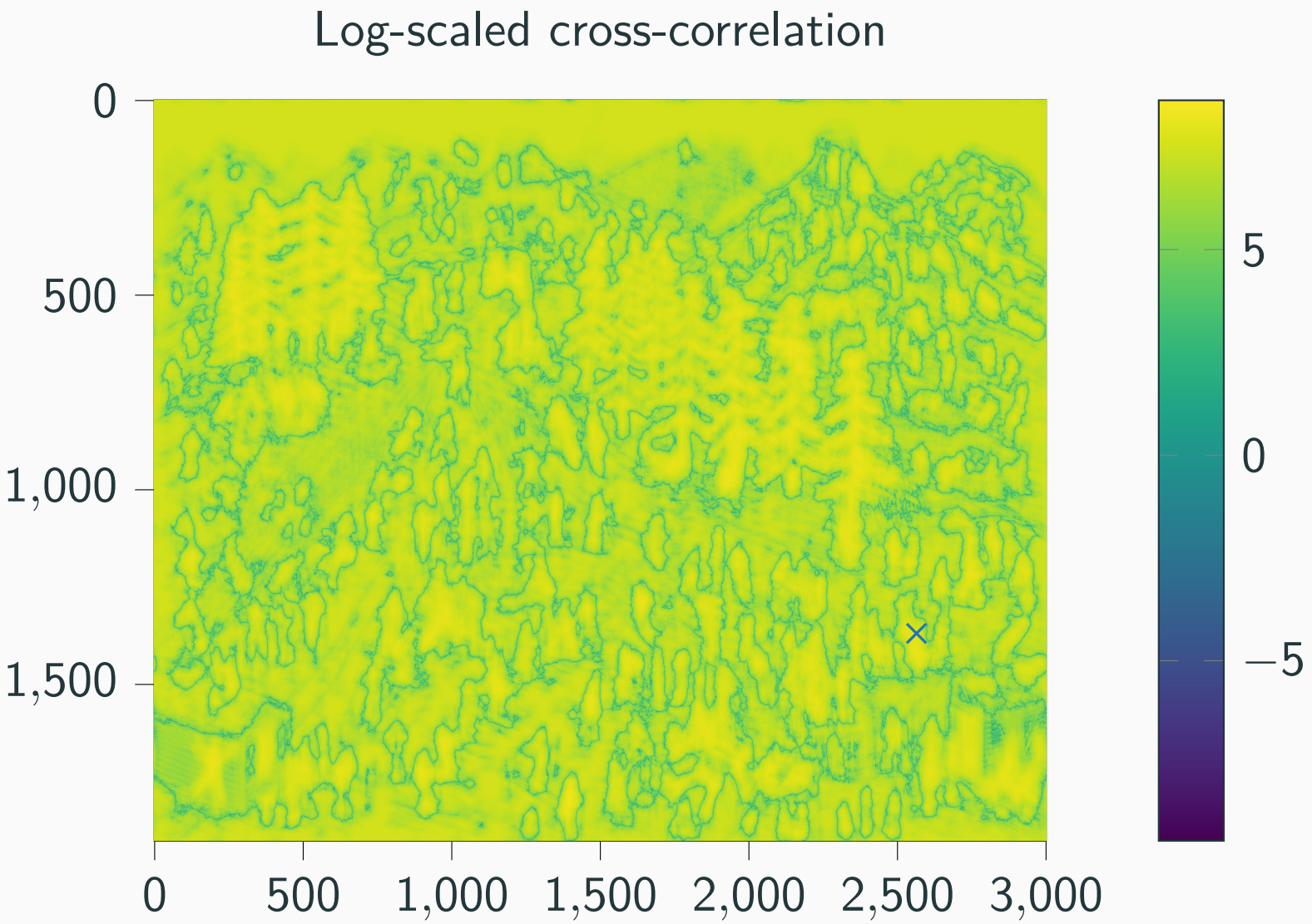
- The convolution operation slides convolution kernels over an image.
- Padding avoids losing pixels on the side.
- Strided convolutions downsample the input.
- Moving in steps of two pixels, for example, cuts the resolution in half.

Understanding convolution

Getting computers to find Waldo



Finding Waldo via cross-correlation.



Summary

- Cross-correlation is called convolution in the machine learning literature.
- Patterns can be located in signals via cross-correlation.

Convolutional neural networks

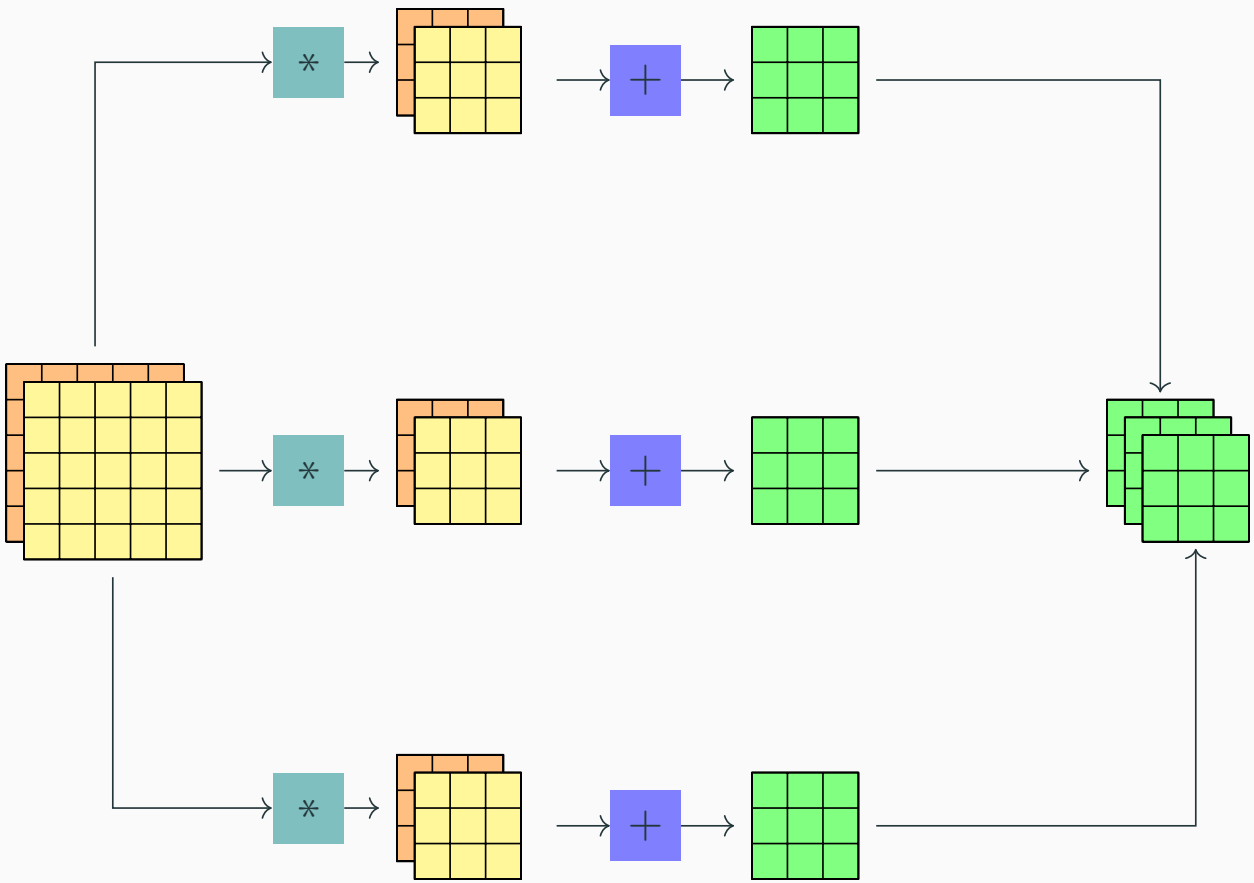
Motivating convolutional neural networks (CNN)

- Fixed filters work if we are looking for a very specific waldo.
- In other cases, we need a better solution.
- Convolutional neural networks rely on filter optimization via back-propagation.
- Filter optimization turns CNNs into very versatile tools!

Multiple filters / kernels

- In the context of CNNs, we refer to convolution not in the strict mathematical sense, but an operation that applies many convolutions in parallel.
- Convolution with a single kernel extracts only one kind of feature (at many spatial locations).
- For each layer, however, we are interested in extracting many kinds of features (at many spatial locations).

Multichannel convolution



The plot shows a convolution computation using a $3x2x3x3$ kernel on a $2x5x5$ input. The kernel pairs convolve with the input, producing $3x3$ results. $+$ adds the two channels for each of the three tensors. Finally, everything is stacked. Inspired by [DV16, page 9].

Calculating the output size

$$W_{out} = (W_{in} - K + 2P)/S + 1$$

- W_{in}: input size (width or height)
- W_{out}: output size (width or height)
- K: kernel size
- P: padding
- S: stride

Image to column and the forward pass

We already know how to train dense network layers using matrix multiplication. Training a CNN the same way requires restructuring the image to express convolution as matrix multiplication,

$$\overline{\mathbf{h}} = \mathbf{K}_f \mathbf{v}_l + \mathbf{b}, \tag{4}$$

$$\mathbf{h}_f = f(\overline{\mathbf{h}}). \tag{5}$$

$\mathbf{v}_l \in \mathbb{R}$ denotes the restructured image input. $\mathbf{K}_f \in \mathbb{R}^{k_o, k_i \cdot k_h \cdot k_w}$ the flattened restructured kernel. o, i, h, w denote the output, input, height, and width dimensions, respectively.

The backward pass

We apply the rules for dense layers to the restructured convolutional layer data,

$$\delta \mathbf{K}_f = [f'(\overline{\mathbf{h}}) \odot \triangle]_f \mathbf{v}_I^T, \qquad \delta \mathbf{b} = f'(\overline{\mathbf{h}}) \odot \triangle, \tag{6}$$

$$\delta \mathbf{x} = \left(\mathbf{K}_f^T [f'(\overline{\mathbf{h}}) \odot \triangle]_f\right)_{I^{-1}}. \tag{7}$$

With I and I^{-1} denoting the `im2col` and `col2im` operations. All major deep learning frameworks have both operations built in.

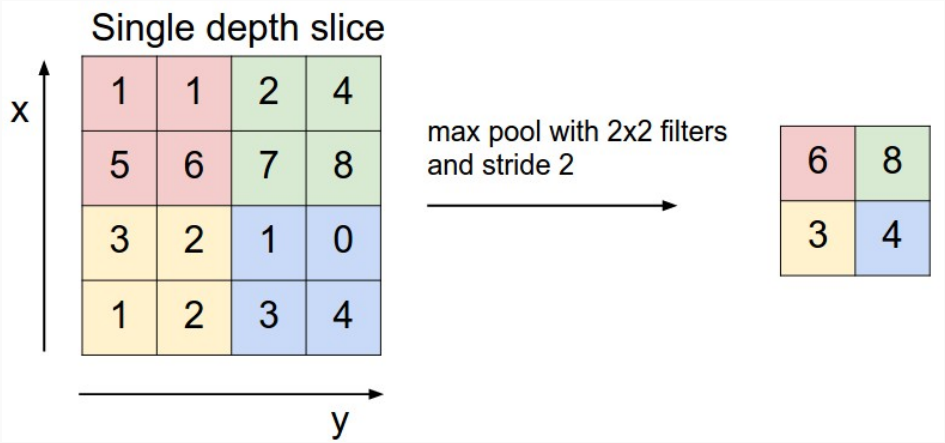
A CNN layer

A typical layer of a CNN has three stages:

1. convolution operation
2. a non-linear activation, e.g. rectified linear unit
3. a pooling function which replaces the output at a certain location with some summary statistic of nearby outputs

Max pooling

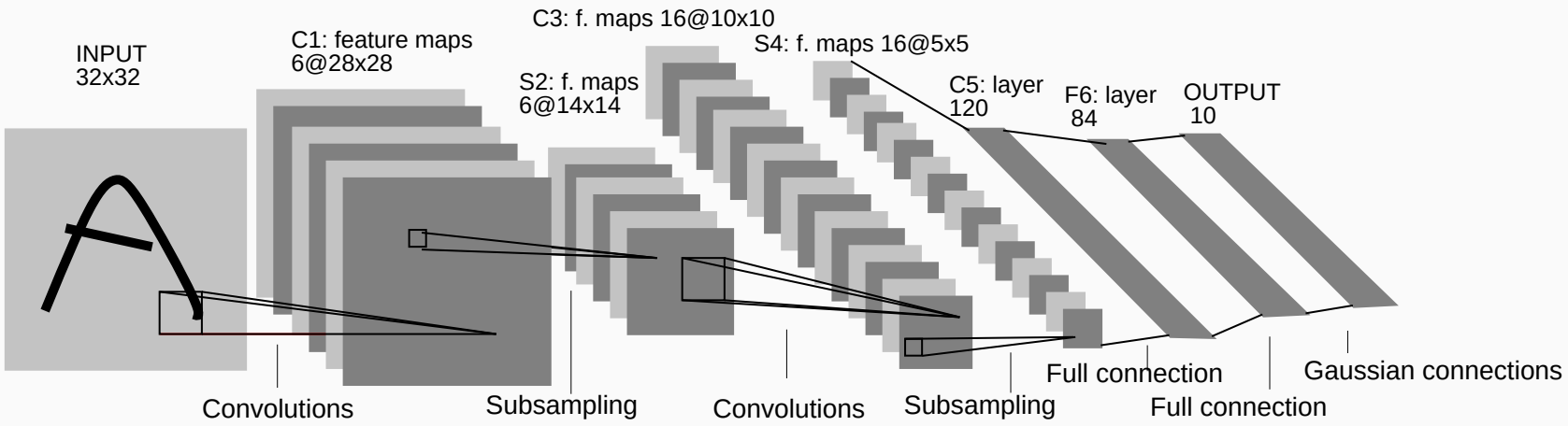
- One of the most common summary statistics used is *max-pooling*.



- Max pooling affords some extra tolerance to spatial shifts.
- For many tasks, pooling is an essential step which allows handling input images of varying sizes.
- This can be handled by varying the size of an offset between pooling regions, so that a subsequent classification layer always receives the same number of pooled statistics, regardless of input size.

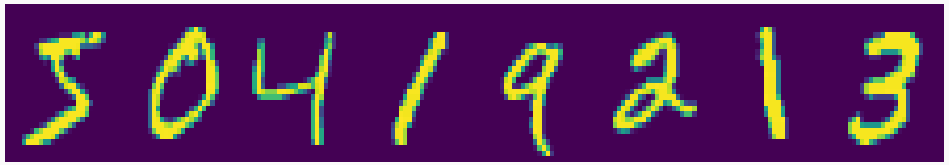
Image link

The classifier at the end

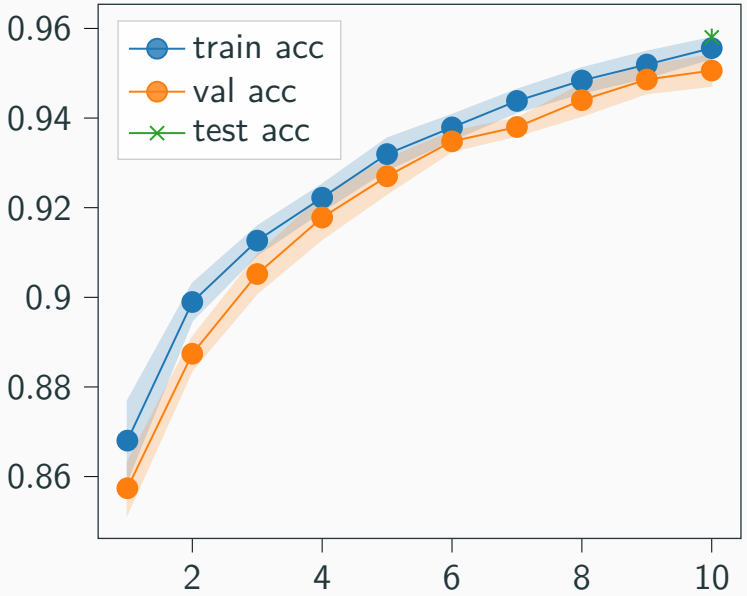


The LeNet5-architecture[LeC+89].

MNIST



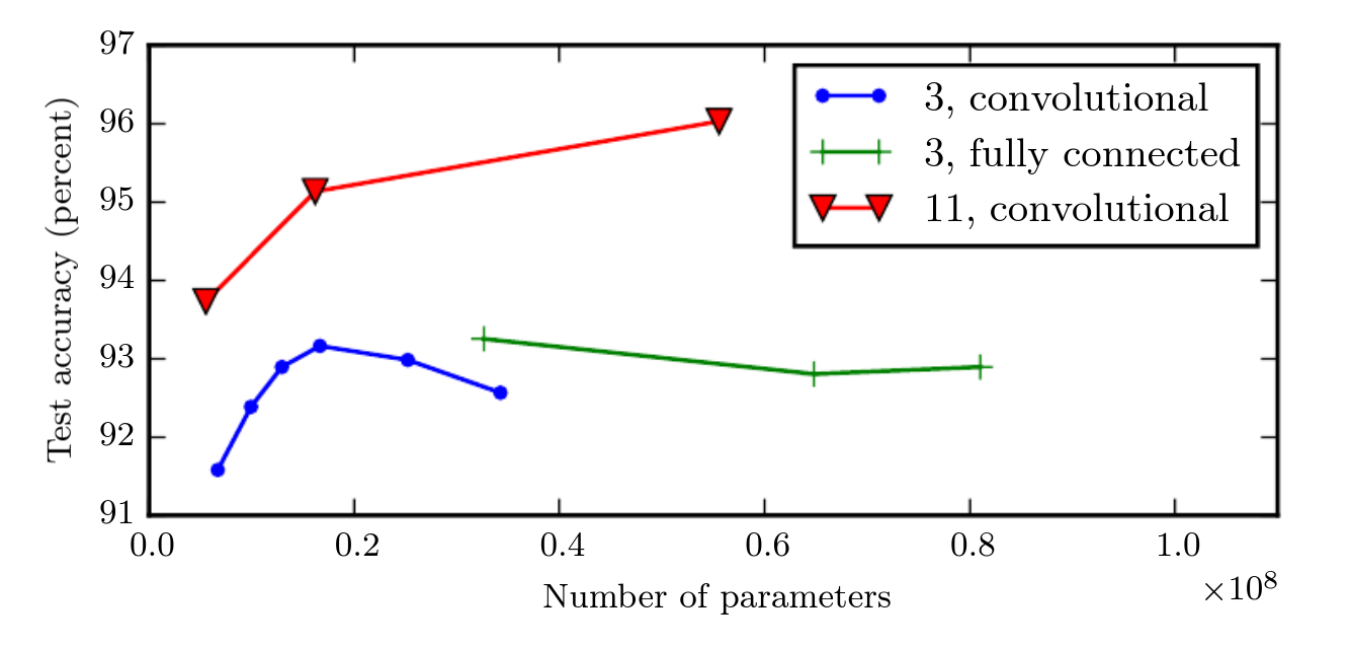
Sample digits from the MNIST-database.



Mean convergence of two-layer CNN with a dense classifier.

Deep convolutional neural networks

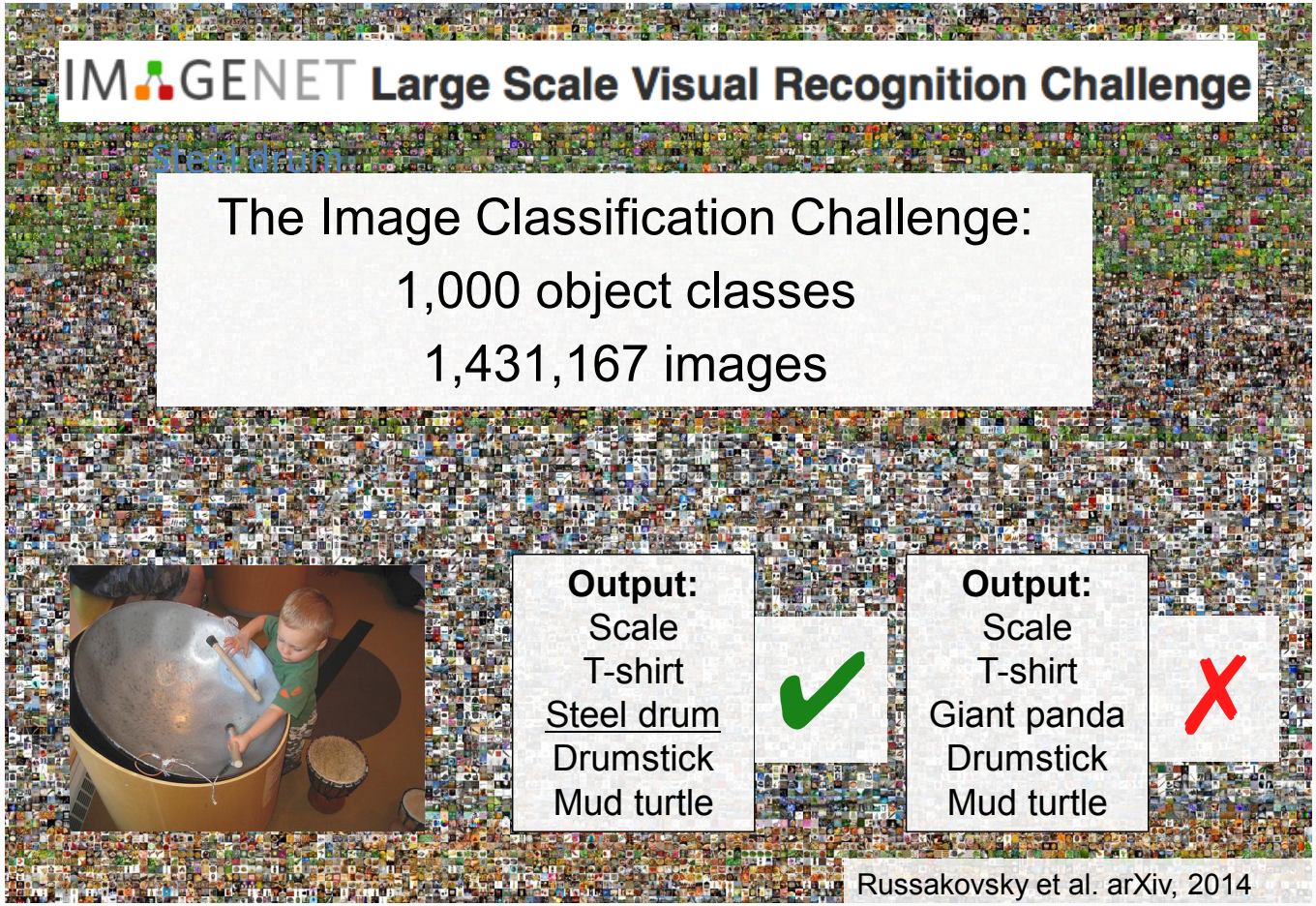
Deeper models perform better



Comparing deep networks with and without convolutional structures on the Google-Street view dataset [GBC16, page 199].

Boom of CNNs

The use of CNNs for visual recognition really took off in 2012, where the winning entry of the ImageNet object recognition challenge was a CNN.



Modern CNNs

What changed in ILSVRC classification?

Year 2010

NEC-UIUC

Dense grid descriptor: HOG, LBP

Coding: local coordinate, super-vector

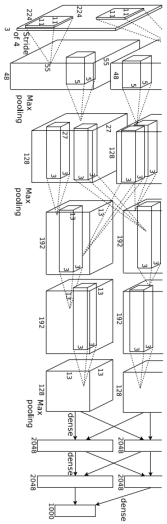
Pooling, SPM

Linear SVM

[Lin CVPR 2011]

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

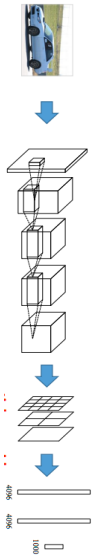
GoogLeNet

[Szegedy arxiv 2014]

VGG

[Simonyan arxiv 2014]

MSRA



[He arxiv 2014]

Slide from here.

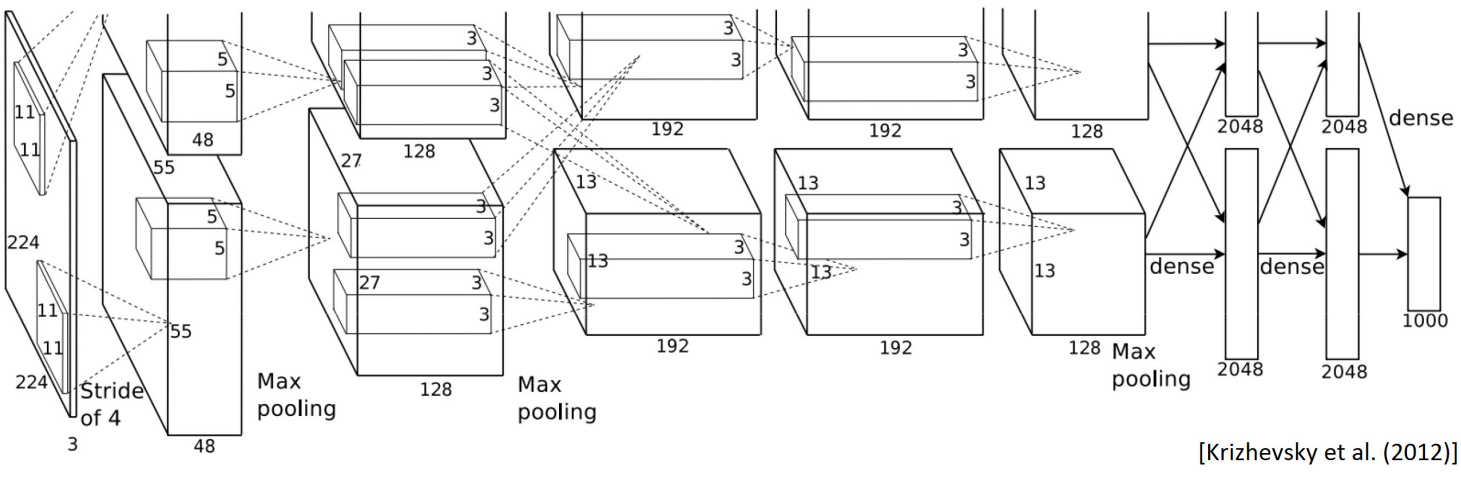
Top-5 error rate over time

Year	Team Name	Model	Error
2012	SuperVision	AlexNet	16.42
2013	Clarifai	ZF Net	11.74
2014	VGG	VGGNet	7.32
2014	GoogLeNet	GoogLeNet	6.67
2015	MSRA	ResNet	3.6

AlexNet

- The first work that popularized Convolutional Networks in Computer Vision
- The Network has a very similar architecture to LeNet, but is deeper, bigger, and featured Convolutional Layers stacked on top of each other.

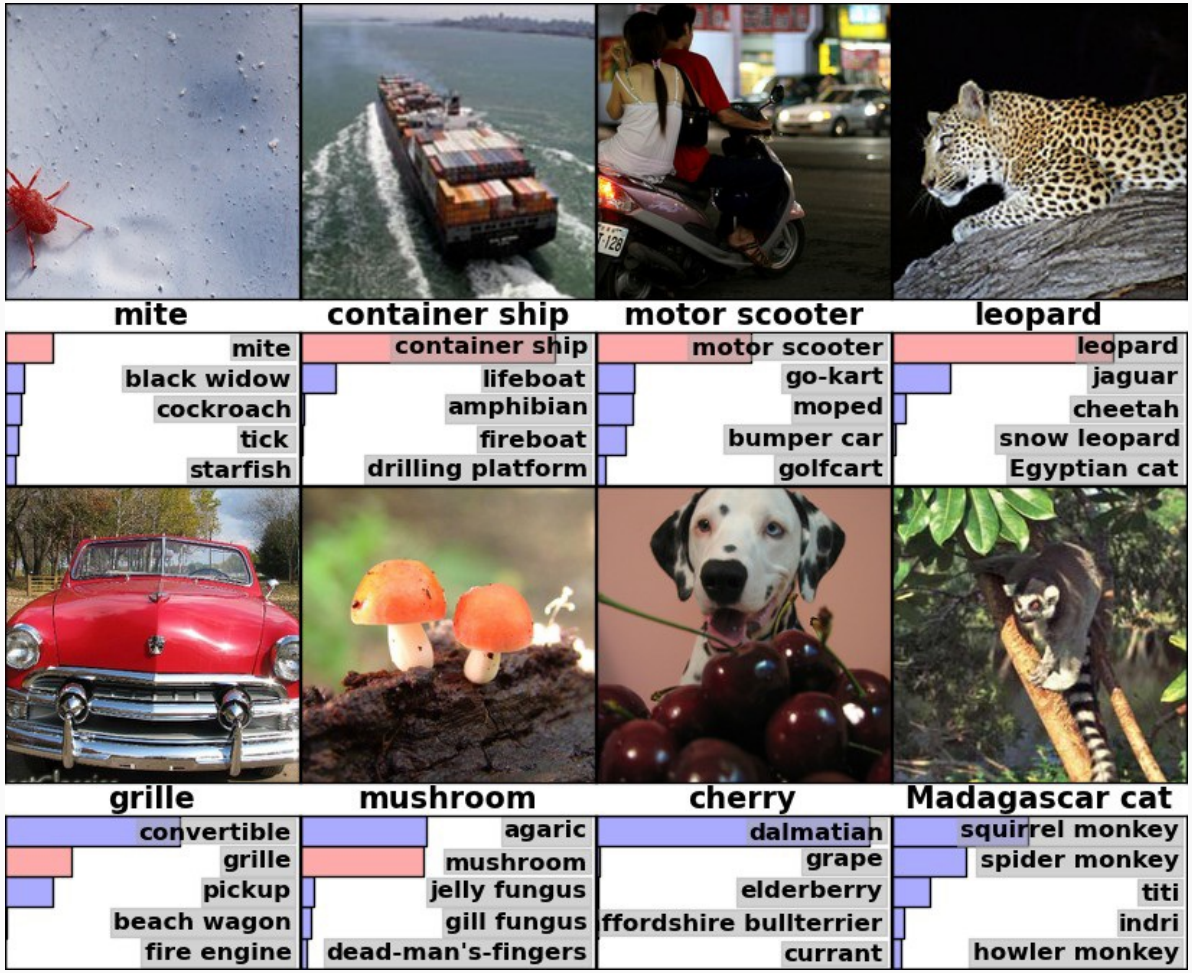
AlexNet Architecture



[Krizhevsky et al. (2012)]

- Trained with Stochastic Gradient Descent for six days on two NVIDIA GTX 580 3GB GPUs.
- Has 650.000 neurons, 60 million parameters, 630 million connections
- Down-sampled the images to a fixed resolution of 256×256

Validation classification

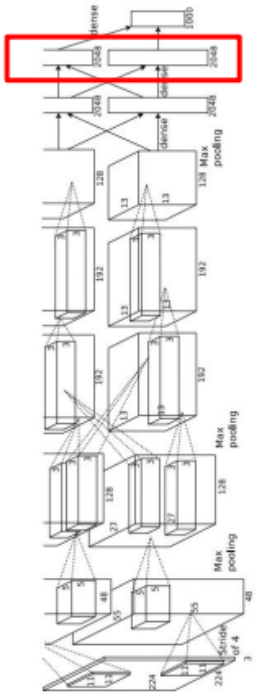


Slide from here.

Visualization: Last Layer

Last Layer: Nearest Neighbors

4096-dim vector



Test image

L2 Nearest neighbors in feature space

Recall: Nearest neighbors in pixel space

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

Slide from here.

Top-5 error rate over time

Year	Team Name	Model	Error
2012	SuperVision	AlexNet	16.42
2013	Clarifai	ZF Net	11.74
2014	VGGNet	VGG	7.32
2014	GoogLeNet	GoogLeNet	6.67
2015	MSRA	ResNet	3.6

VGGNet

- Main contribution: Depth matters!, from 11 to 19 weight layers.
- Extremely homogeneous architecture by using smaller-size convolution filters: only 3x3 convolutions and 2x2 pooling throughout the whole net.
- Expensive to evaluate and requires more memory

VGGNet Architecture

[Simonyan and Zisserman, 2014]

Key design choices :

- Convolutional layers
  - 3×3 conv. filters - the smallest size to capture the notion of left/right, up/down, center
  - Stride 1 – no loss of information, with 1 pixel padding
- Max-pooling
  - performed over a 2×2 pixel window, with stride 2
  - 5 max-pool layers (x2 reduction)
- 3 fully-connected (FC) layers

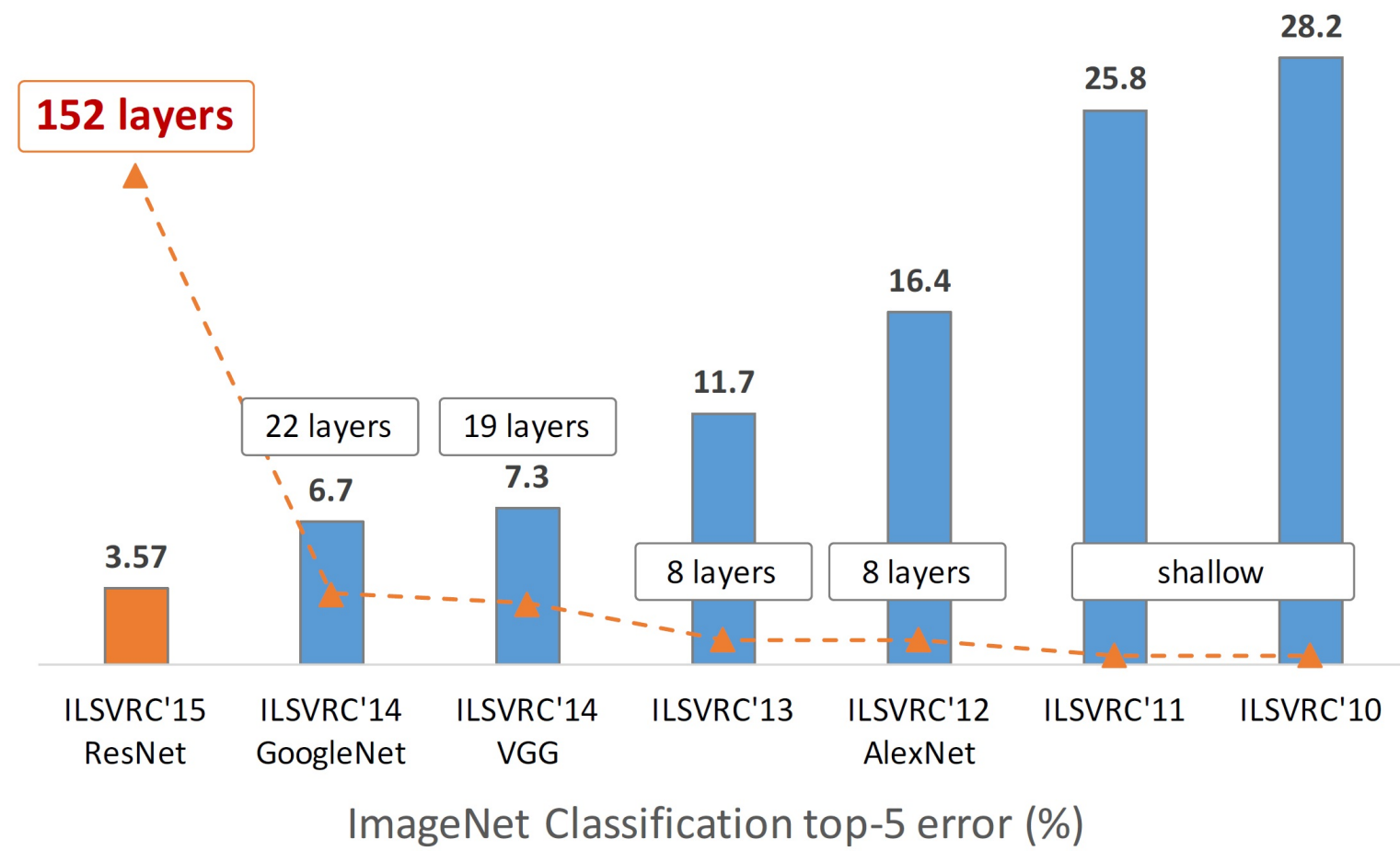
Top-5 error rate over time

Year	Team Name	Model	Error
2012	SuperVision	AlexNet	16.42
2013	Clarifai	ZF Net	11.74
2014	VGGNet	VGG	7.32
2014	GoogLeNet	GoogLeNet	6.67
2015	MSRA	ResNet	3.6

ResNet

- Revolution of Depth

Revolution of Depth



slide from [here]

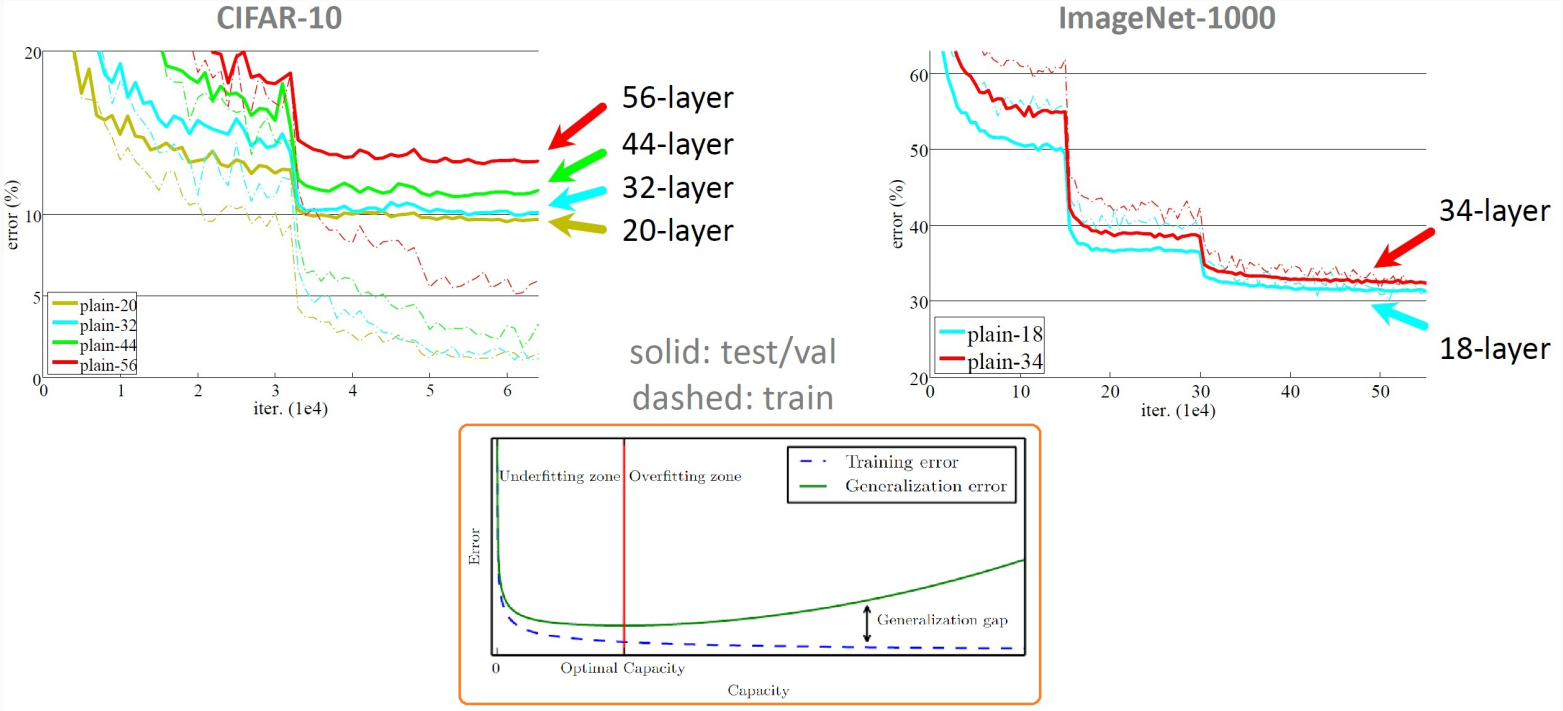
Revolution of Depth

ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

slide from [here]

Simply stacking layers?

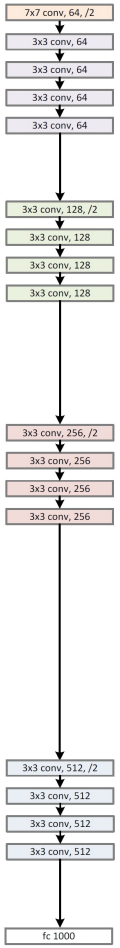


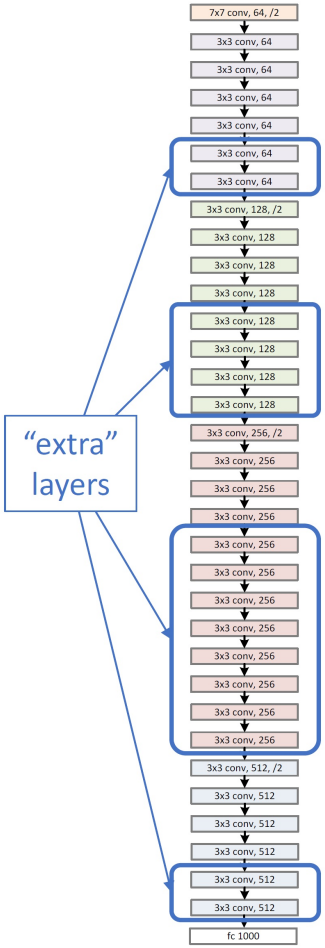
- Plain nets: stacking 3×3 conv layers.
- 'Overly deep' plain nets have higher training error
- A general phenomenon, observed in many datasets

Image from here.

ResNet Architecture

a shallower model (18 layers)





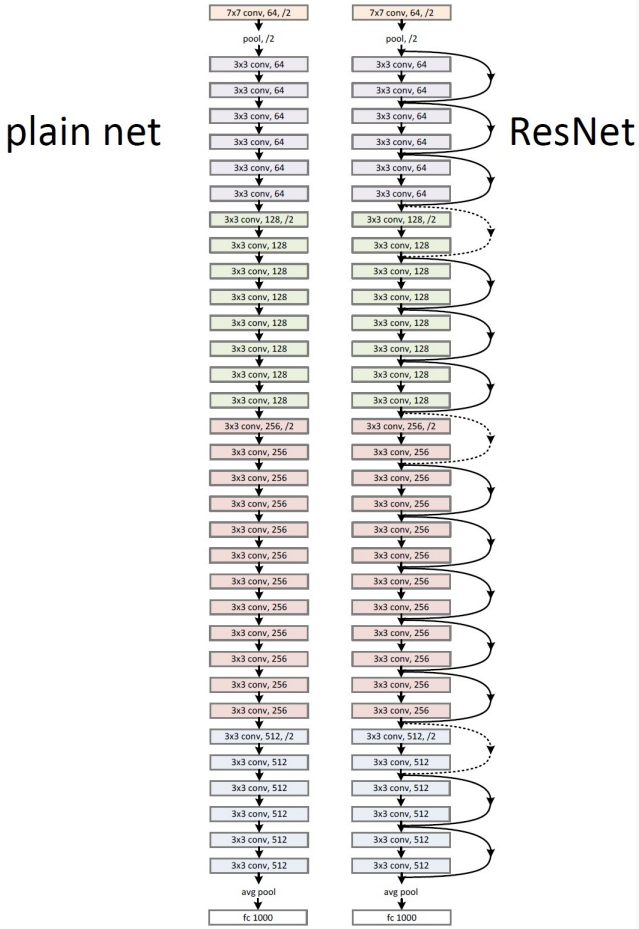
a deeper counterpart (34 layers)

- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

slide from [here]

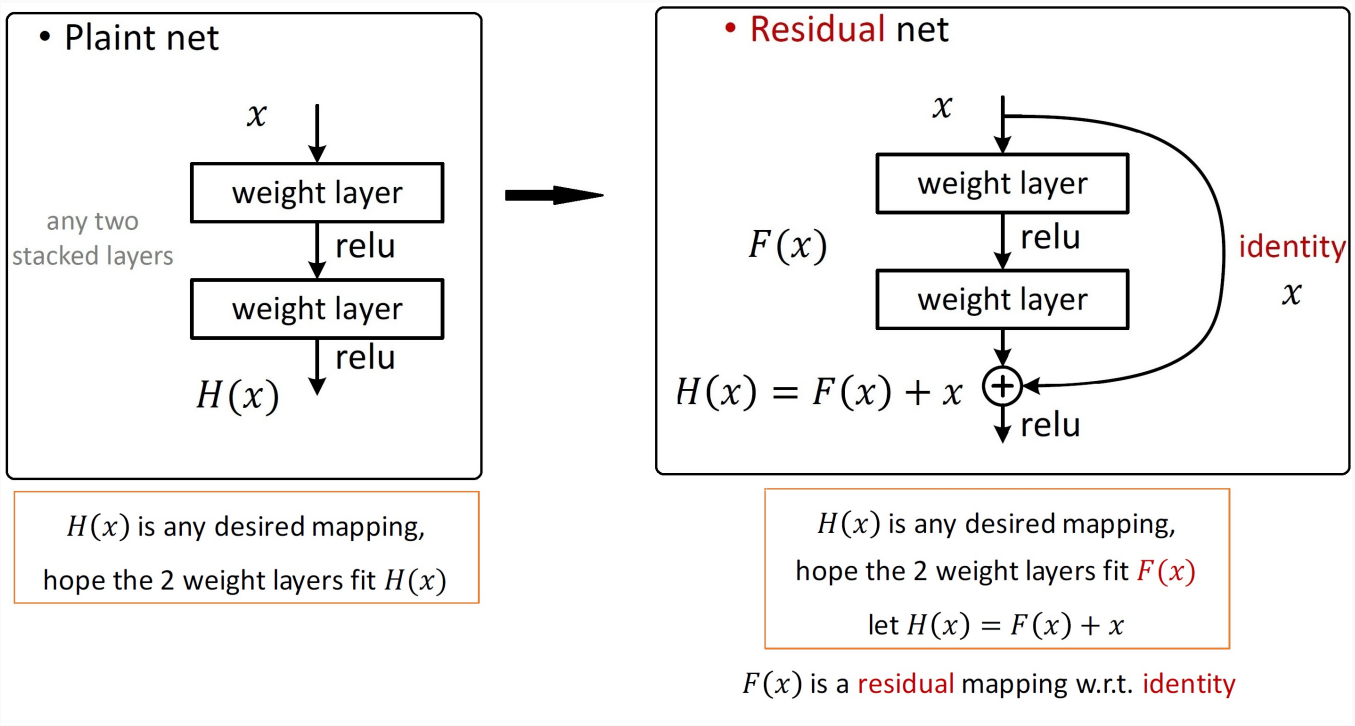
Network Design

- Keep it simple
- Our basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2
  - Simple design; just deep!
- Other remarks:
  - no max pooling (almost)
  - no hidden fc
  - no dropout



slide from [here]

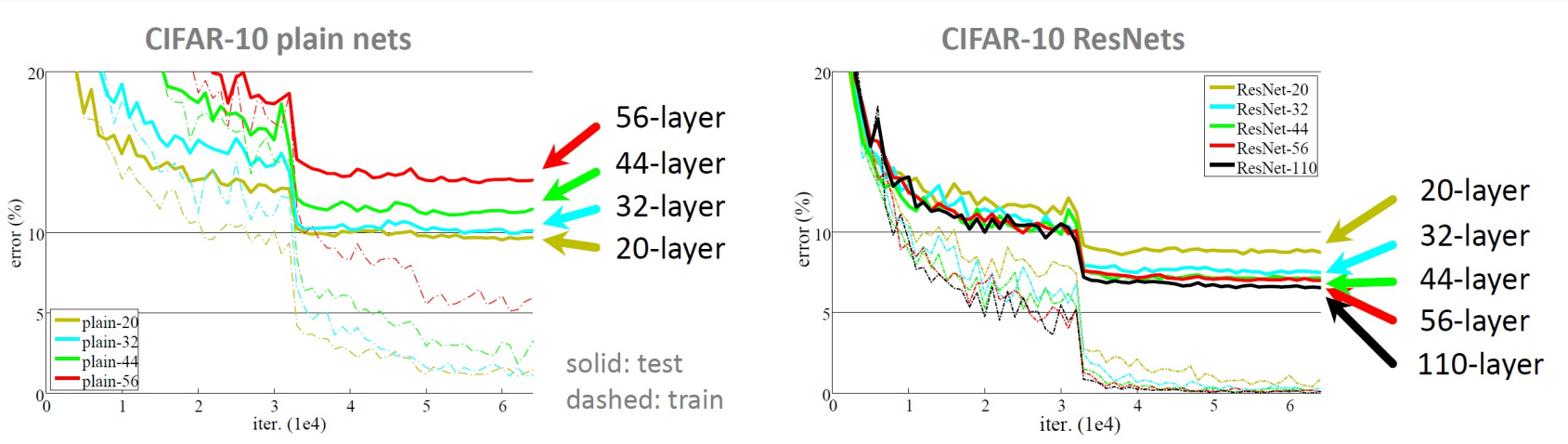
Deep Residual Learning (Residual Block)



slide from [here]

- Adding skip-connections (skipping one or more layers) that are summed with the output of the convolution layers.
- Identity skip-connections don't add extra parameter or computational complexity.

CIFAR-10 experiments



slide from [here]

- Deep ResNets can be trained without difficulties
- Deeper ResNets have lower training error, and also lower test error

Summary

- Deeper models tend to perform better.
- Skip-connections allow us to train very deep networks.

References

[DV16] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning." In: *arXiv preprint arXiv:1603.07285* (2016).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Literature ii

[LeC+89] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. “Handwritten digit recognition with a back-propagation network.” In: *Advances in neural information processing systems* 2 (1989).