Convolutional Neural Networks

Lecturer: Michael Möller – michael.moeller@uni-siegen.de
Exercises: Hartmut Bauermeister – hartmut.bauermeister@uni-siegen.de

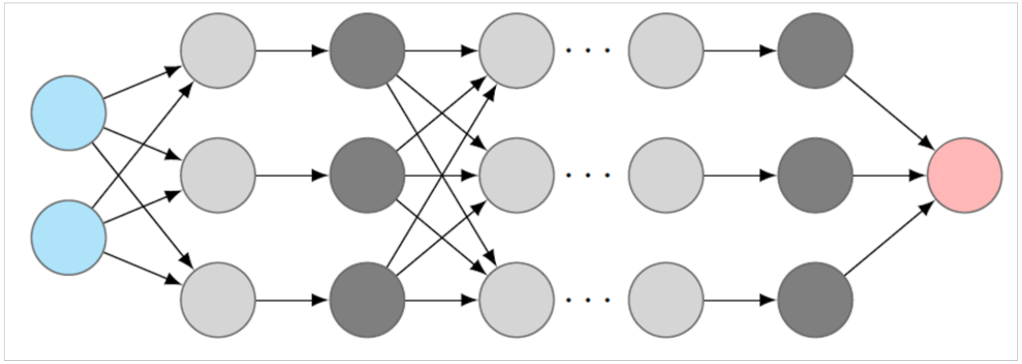


Image classification

Training data

These are zeros

These are ones

…

…

…

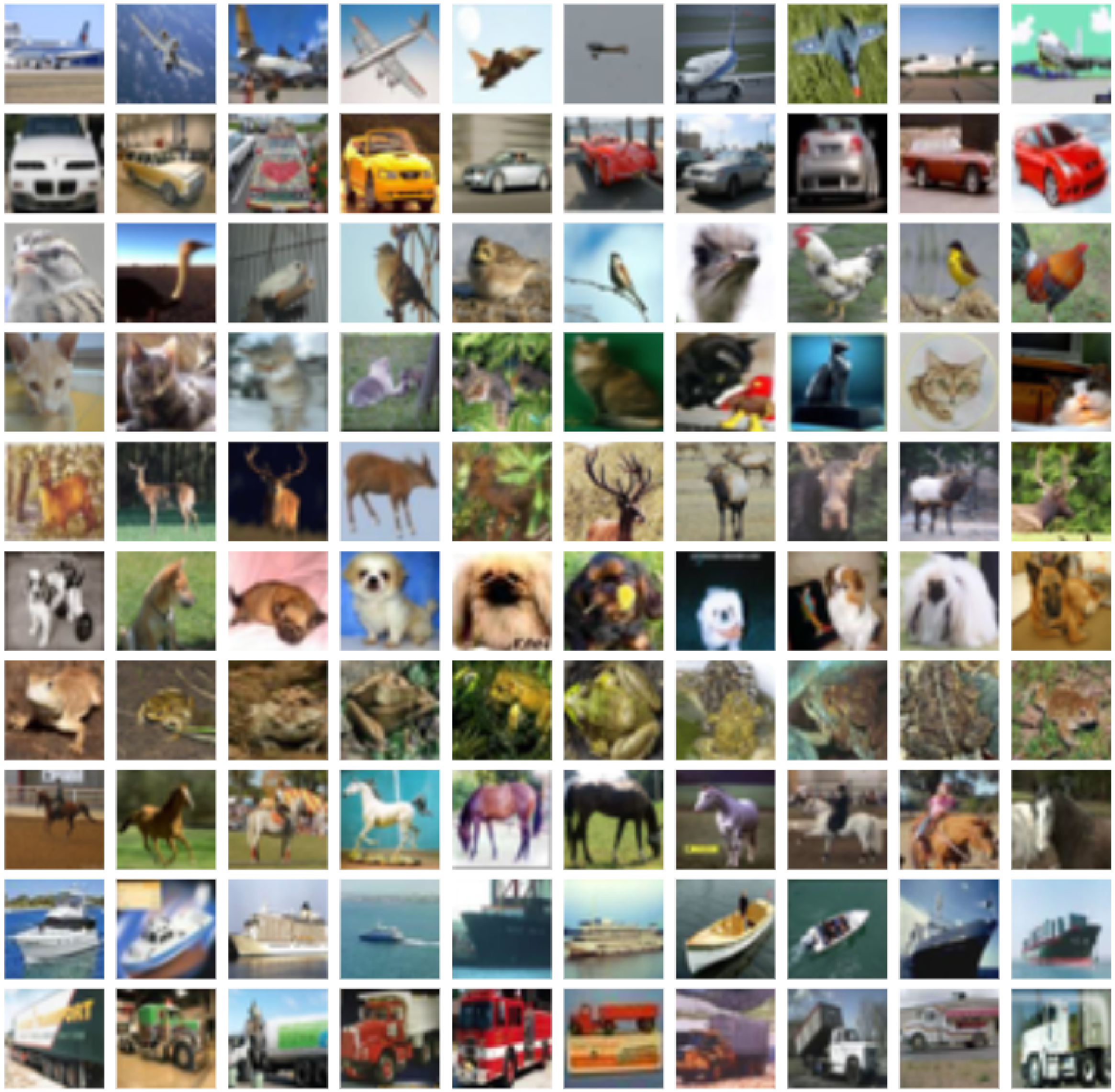
These are ninth

Question to answer

Which handwritten digit is this?

Image classification

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



Goal: Train a network on assigning images to predefined categories.

Networks with images

New challenge for us: How to handle images as an input?

Previously for fully connected networks:

$$\begin{aligned} \mathcal{N}: \mathbb{R}^{m_{in}} \times \mathbb{R}^n &\to \mathbb{R}^{m_{out}} \\ (\boxed{x,}\theta) &\mapsto \mathcal{N}(x;\theta) \end{aligned}$$

vector

New goal:

$$\begin{aligned} \mathcal{N}: \mathbb{R}^{n_y \times n_x \times n_c} \times \mathbb{R}^n &\to \mathbb{R}^{m_{out}} \\ (\boxed{x,}\theta) &\mapsto \mathcal{N}(x;\theta) \end{aligned}$$

image

Naïve way: Vectorize the image $x \in \mathbb{R}^{n_y \times n_x \times n_c} \to \vec{x} \in \mathbb{R}^{n_y n_x n_c \times 1}$

then design a usual fully connected network.

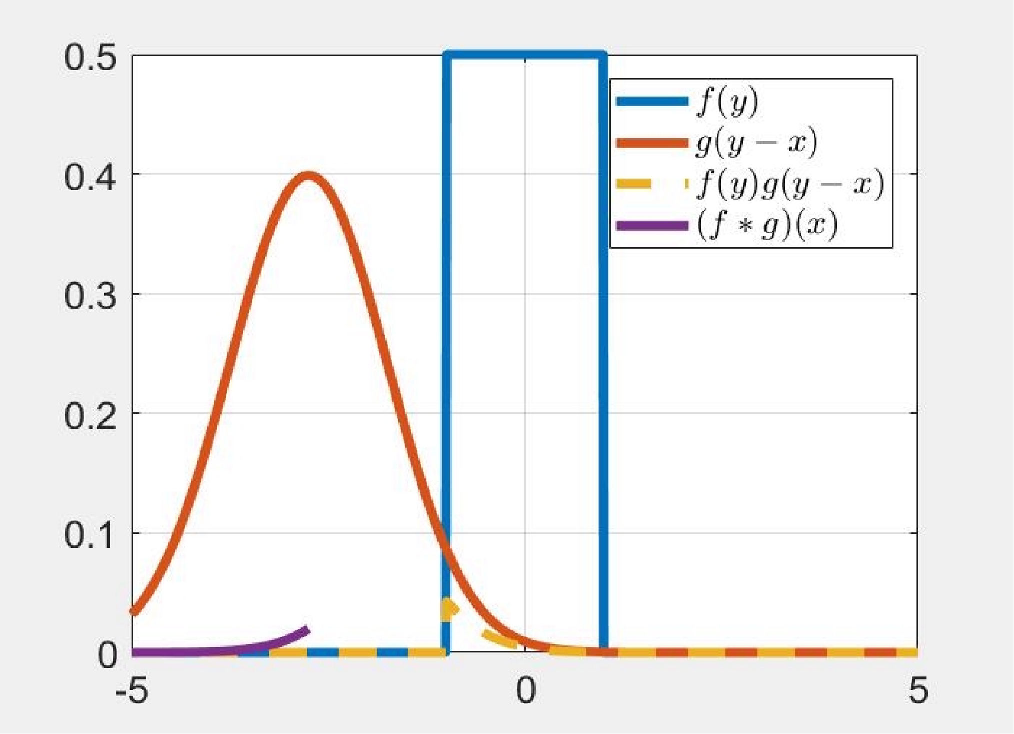
But this is almost never a good idea!

- A single fully connected layer to classify one megapixel color images into 1000 categories has three billion parameters!
- **Structure** instead of brute force: Processing images should yield largely shift-invariant results! Therefore, use **convolutions**!

Convolutions

Reminder: What is a convolution?

Continuous in 1d: $$(f * g)(x) = \int_{-\infty}^{\infty} f(y)\ g(x-y)\ dy$$



Interpretation: Take a running average of values in f using the weights specified in g.

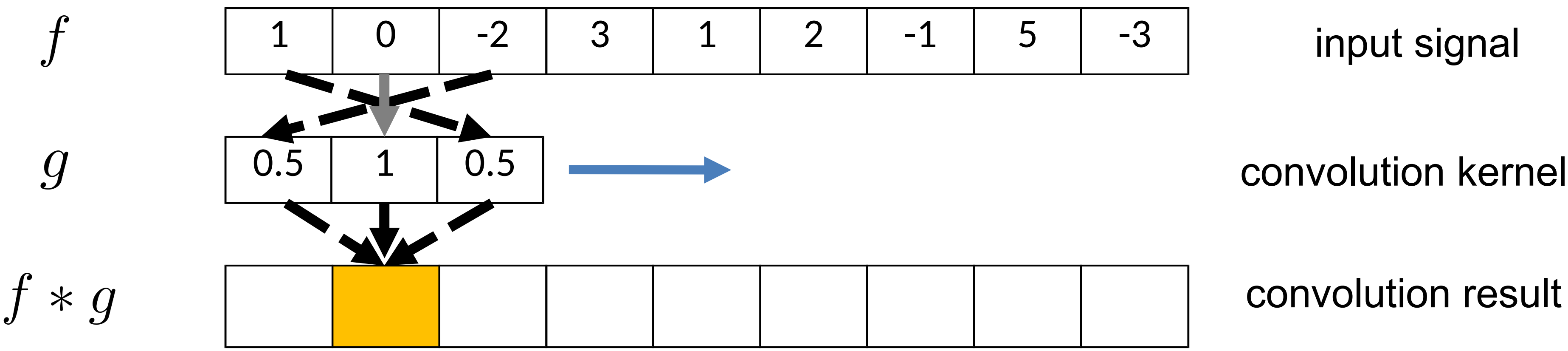
Important for us: Discrete convolution

$$(f * g)_i = \sum_{j=-\infty}^{\infty} f_j\ g_{i-j}$$

But what is “from -infinity to infinity” supposed to mean? What happens in practice for vectors?

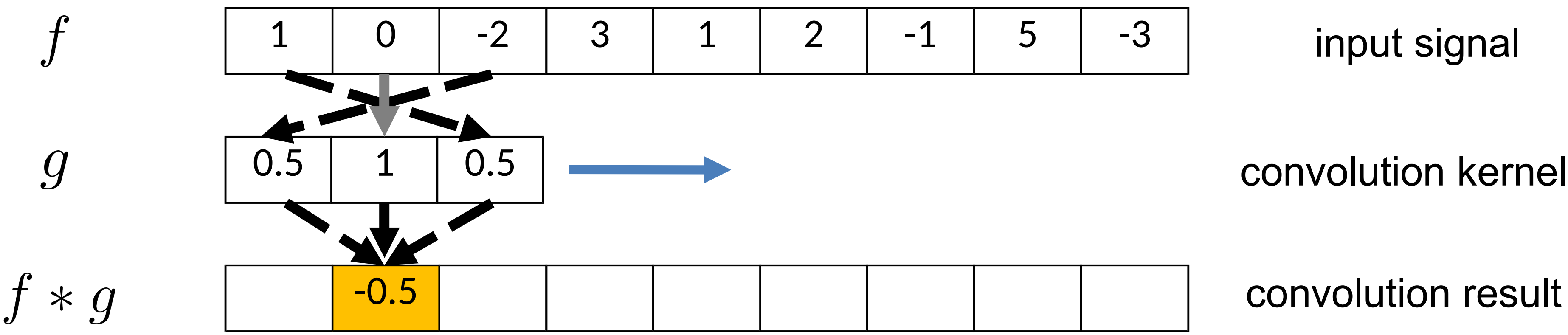
Convolutions

What does a convolution do?



Convolutions

What does a convolution do?



Convolutions

What does a convolution do?

f — input signal

1	0	-2	3	1	2	-1	5	-3

g — convolution kernel

0.5	1	0.5

$f * g$ — convolution result

	-0.5							

Convolutions

What does a convolution do?

f	1	0	-2	3	1	2	-1	5	-3	input signal
g		0.5	1	0.5						convolution kernel
$f * g$		-0.5	-0.5							convolution result

Convolutions

What does a convolution do?

f — input signal

1	0	-2	3	1	2	-1	5	-3

g — convolution kernel

0.5	1	0.5

$f * g$ — convolution result

	-0.5	-0.5	2.5					

Convolutions

What does a convolution do?

f	1	0	-2	3	1	2	-1	5	-3	input signal
g				0.5	1	0.5				convolution kernel
$f * g$		-0.5	-0.5	2.5	3.5					convolution result

Convolutions

What does a convolution do?

f	1	0	-2	3	1	2	-1	5	-3	input signal
g					0.5	1	0.5			convolution kernel
$f * g$		-0.5	-0.5	2.5	3.5	2				convolution result

Convolutions

What does a convolution do?

f	1	0	-2	3	1	2	-1	5	-3	input signal
g						0.5	1	0.5	→	convolution kernel
$f * g$		-0.5	-0.5	2.5	3.5	2	2.5			convolution result

Convolutions

What does a convolution do?

f	1	0	-2	3	1	2	-1	5	-3	input signal
g							0.5	1	0.5	convolution kernel
$f * g$		-0.5	-0.5	2.5	3.5	2	2.5	3		convolution result

Convolutions

What does a convolution do?

f — input signal

1	0	-2	3	1	2	-1	5	-3

g — convolution kernel

0.5	1	0.5

$f * g$ — convolution result

?	-0.5	-0.5	2.5	3.5	2	2.5	3	?

And what happens here?

Option 1: The result of the convolution is smaller than the input signal (*valid*)

Option 2: We assume the input signal to be periodic (*circular*)

Option 3: We assume the “missing” entries of f for computing the convolution are zero (*zero-padding*)

Option 4: Replicate the entry at the beginning/end of f (*replicate*)

Michael Möller – michael.moeller@uni-siegen.de

Convolutions

Discrete convolution formula

$$(f * g)_i = \sum_{k=-s}^{s} f_{i-k}\, g_k$$

if $n - s \geq i > s$,
where it is convenient to index a filter g of length $2s+1$ with $g_k, \quad k \in \{-s, ..., s\}$

For $i \leq s$ and $i > n - s$ the specific formula depends on the choose option how to handle the boundary.

Discussion in the lecture: The definition of ***convolutions*** would flips the kernel, and then multiply pointwise in a sliding window and sum the resulting values. If you want to avoid the flipping, you need to talk about the ***cross-correlation*** operator!

Class **torch.nn.Conv2d** → “where $\star$ is the valid 2D cross-correlation operator”

See https://en.wikipedia.org/wiki/Cross-correlation

Correlation

Discrete cross-correlation formula

$$(f * g)_i = \sum_{k=-s}^{s} f_k \, g_{i+k}$$ + boundary treatment

And in 2d?

$$(f * g)_{i,j} = \sum_{k=-s}^{s} \sum_{l=-s}^{s} f_{k,l} \; g_{i+k,j+l}, \quad \text{if} \quad n_y - s \geq i > s, \; n_x - s \geq j > s.$$

And the boundary again needs to be treated in one of the aforementioned ways.

Convolution is cross-correlation with a kernel rotated by 180 degrees

(we will often be sloppy about this and talk about convolutions meaning correlations)

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

Cross-correlation result

$f * g$

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

Cross-correlation result

$f * g$

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

$$-13 = \begin{array}{l} 0 \cdot 1 + (-1) \cdot 2 + 0 \cdot (-1) \\ (-1) \cdot 4 + 4 \cdot (-1) + (-1) \cdot 2 \\ 0 \cdot 2 + (-1) \cdot 1 + 0 \cdot 4 \end{array}$$

Cross-correlation result

$f * g$

	-13				

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

Cross-correlation result

$f * g$

	-13	3			

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

Cross-correlation result

$f * g$

	-13	3	-1		

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

Convolution result

$f * g$

	-13	3	-1	12	

Correlation

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

kernel

g

0	-1	0
-1	4	-1
0	-1	0

Cross-correlation result

$f * g$

	-13	3	-1	12	
	-6				

Correlation

And for images with multiple channels?

Although it is easy to extend convolutions to 3d

$$(f * g)_{i,j,c} = \sum_{k=-s}^{s} \sum_{l=-s}^{s} \sum_{h=-s}^{s} f_{i-k,j-l,c-h} \;\; g_{k,l,h}, \qquad \text{if} \qquad \begin{aligned} n_y - s &\geq i > s \\ n_x - s &\geq j > s \\ n_c - s &\geq c > s \end{aligned}$$

this is typically not what happens in convolutional neural networks! Instead, one convolves with a 3d filter whose extend in the third dimension coincides with the number of channels of the data to be filtered, and does not move in the third dimension.

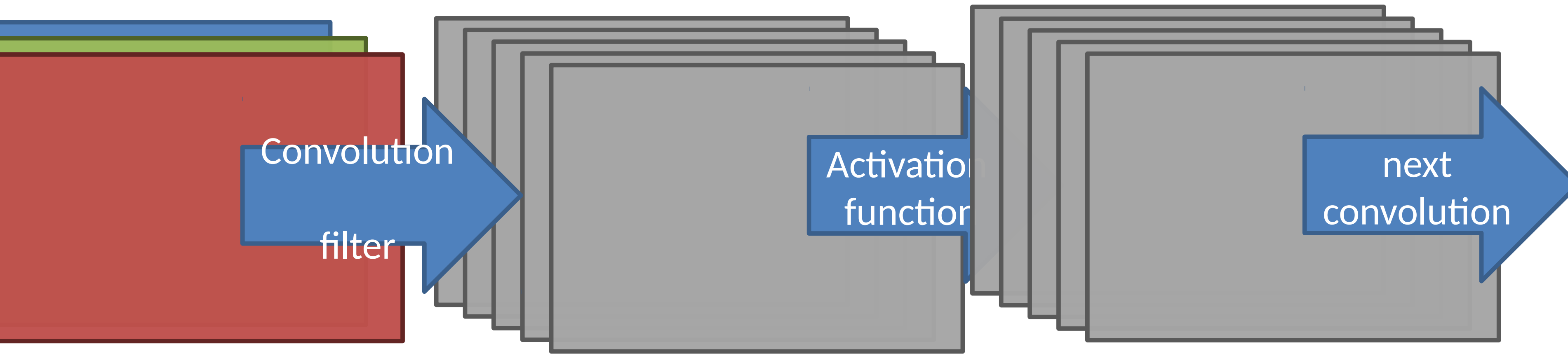
Correlation

Animation taken from https://sites.google.com/site/nttrungmtwiki/home/it/data-science---python/tensorflow/tensorflow-and-deep-learning-part-3

Michael Möller – michael.moeller@uni-siegen.de

Convolutional Neural Networks

Network architecture design for images:



RGB-image

Result of several filters
(5 in this case)

Result, e.g. after
ReLU

The number of filters determines the number of channels in the next layer!

In the lecture: Discuss number and dimension of learnable parameters in more detail!
Convolutions typically also have a bias (a single number per filter)

Convolutional Neural Networks

Just composing convolutions with ReLUs does not change the size of the image.

What if the desired output is not an image but, e.g. a vector?

input

$$\rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

desired output

Common operation: Convolution with ***stride***:

Do not compute the result of the convolution at every pixel, but at every second/third/fourth pixel only. The number of pixels you move before computing another convolution is the stride.

Convolutions with stride

Input image

f

1	2	-1	4	4	2
4	-1	2	3	6	2
2	1	4	1	3	3
1	5	2	4	8	7
3	5	2	1	9	8
6	5	7	6	6	6

Convolution result without padding and stride 3

-13	12
5	13

kernel

g

0	-1	0
-1	4	-1
0	-1	0

- Of course the computation blocks may also overlap (e.g. stride 2)
- Do not use stride to reduce the size of the image too quickly!

Pooling Layers

Another frequently used way to reduce the size of the image are ***pooling layers***

All pooling variants use a sliding window over the image (similar to a convolution), but often in a non-overlapping fashion. Each window (of which one can specify the size), gets reduced to a single number, by

- Taking the maximum value among the entries within the window (*max.-pooling*)
- Taking the average value among the entries within the window (*avg.-pooling*)
- Less frequent: Taking the ℓ^p norm of each window (*fractional max-pooling*)

Average pooling is just a strided convolution with a fixed convolution kernel (that cannot be learned).

Max-pooling has the goal to select the strongest features in a region (intuition: what is the strongest response to the “ear”/”wheel”/”eye”/”tail”-filter in each region).

Summary

We know the following layers/functions and how to utilize them in networks

Fully connected layers
Multiply with a matrix, add an offset vector.

Activation functions
Break the linearity of networks, e.g. ReLU, Sigmoid

Convolutional layers
Convolve the 3d input with a kernel of user-defined height and width. The third dimension needs to coincide with the input's third dimension. Add an offset. Possibly use stride.

Pooling Layers
Max-pooling, average pooling. "Collect and Select" activations

Dropout Layer
Not a layer during inference. Merely helps to generalize and learn redundant representations.

Batch Normalization Layers
Map your activations back to a reasonable range (zero mean, unit variance). Possibly allow the network to learn to undo this.