AI and Deep Learning

4-1 Convolution

Zhonglei Wang

WISE, SOE, CHOW

XMU

(Version v1)

Contents

Background of convolution neural network

1. Given an image, why not use a fully connected neural network?
   - Number of parameters is overwhelmingly large
     - Increase the computation and memory burden
     - Easily leading to overfitting
   - Overlooks the spatial structure
   - Not location (translation) invariant
2. Consider a new network, convolution neural network, for image processing

Convolution neural network

1. Advantages of convolution:
   - Convolution is applied for local regions independently, so computation can be parallelized
   - Convolution shares kernels over different local regions, so it involves fewer parameters
   - Convolution takes local spatial information into consideration, so the model is more spatially invariant
2. Neural network using convolution is called convolutional neural network (CNN)

Convolution

1. Convolution is not new to us
   - In probability, convolution is related to the density function of $Z = X + Y$

$$f_Z(z) = \int f_X(x) f_Y(z - x) \mu(\mathrm{d}x),$$

     - X, Y : Two independent random variables
     - μ : Dominating measure
     - $f_\star(\cdot)$: Density function for random variable $\star$

2. For simplicity, denote the convolution of functions f and g to be

$$h(z) = (f * g)(z) = \int f(x) g(z - x) \mu(\mathrm{d}x) = \int f(z - x) g(x) \mu(\mathrm{d}x)$$

   - Functions f and g should make the definition valid

1d convolution

Input	3	6	5	4	8	9	1	7	9	10	10
Kernel									−1	0	1

Calculation

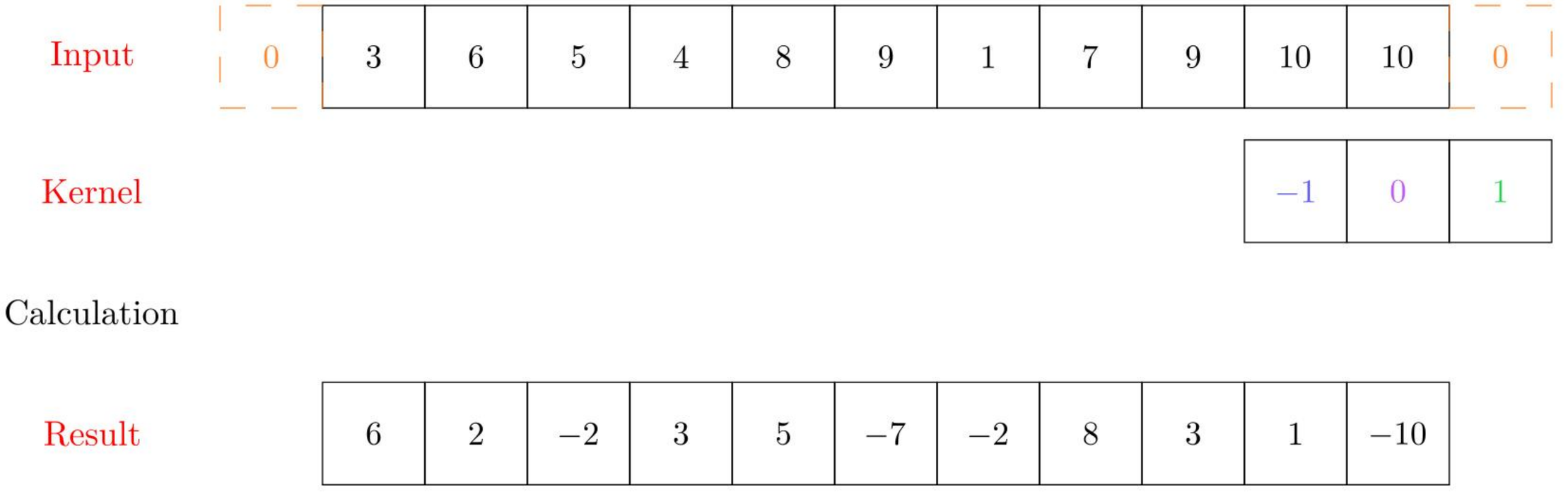
Result	2	−2	3	5	−7	−2	8	3	1

1. Usually, we flip $g(x)$ to obtain a kernel so that we can conveniently use `np.sum`
2. It is essentially a inner product
3. The values associated with x are not important

Padding

1. Intuition
   - The size of the input sequence decreases after convolution
   - Some (important) information on the boundary is overlooked
2. Padding is used to solve the above problems
   - Add zeros about the boundary manually
   - Information on the boundary can be sufficiently used
   - Control the length of the output sequence

Padding



1. Output sequence has the same length as the input sequence

Padding

1. Two types of convolutions:
   - Valid convolution: No padding
   - Same convolution: Apply padding to keep the dimension of the input and output unchanged
2. We can pad more than one 0 around the boarders.
   - Different number of 0's can be used for different boarders
3. We can move more than one step to do convolution to
   - Decrease the output size
   - Balance accuracy and computation cost

Stride

Input	3	6	5	4	8	9	1	7	9	10	10

Kernel	−1	0	1

Calculation

Result	2	3	−7	8	1

1. In the above example, stride is 2, and it can be other positive integers
2. Padding focuses on boundary information, and stride on sampling density
3. Padding and stride can be used at the same time

Dilated/Atrous convolution

Input	3	6	5	4	8	9	1	7	9	10	10

Dilated Kernel	−1	0	1	0	1

Calculation

Result	10	7	4	12	2	8	18

1. In the above example, dilation rate is 2
   - Dilation rate n means that we insert $(n-1)$ between elements of the kernel
2. Dilation increases the receptive field without (essentially) increasing kernel size

Trade-off

1. More padding preserves better boundary information at the cost of higher computational load and memory usage
2. A larger stride reduces computational cost, but produces sparser outputs and potentially loses more details
3. A larger dilation enlarges the receptive field, but the local information between adjacent positions becomes less coherent

2d convolution

Original sequence

(Padding = 1)

0	0	0	0	0	0	0
0	3	6	5	4	8	0
0	9	1	7	9	6	0
0	8	0	5	0	9	0
0	6	2	0	5	2	0
0	6	3	7	0	9	0
0	0	0	0	0	0	0

Kernel

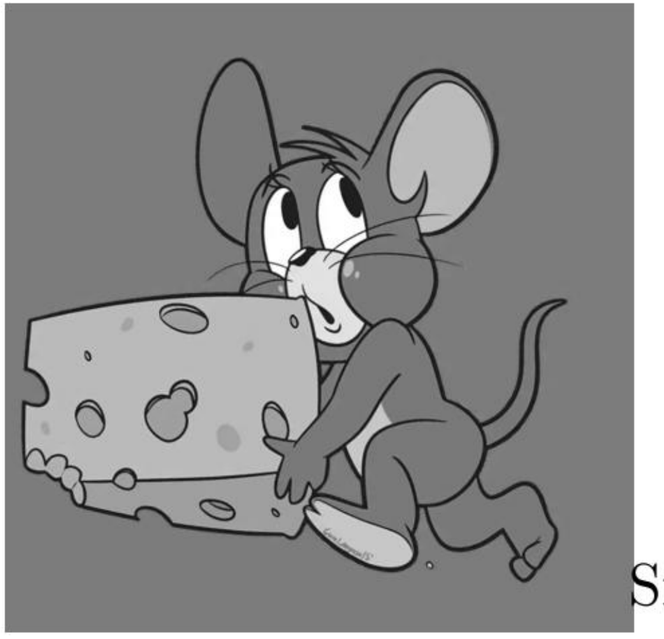
-1	1
-1	1

Result

(Stride=2)

3	-1	4
17	11	6
12	2	6

Average kernel



$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Smooth output

Attenuate local noise and sharp information

Gaussian kernel

$$\begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix}$$

Smooth output

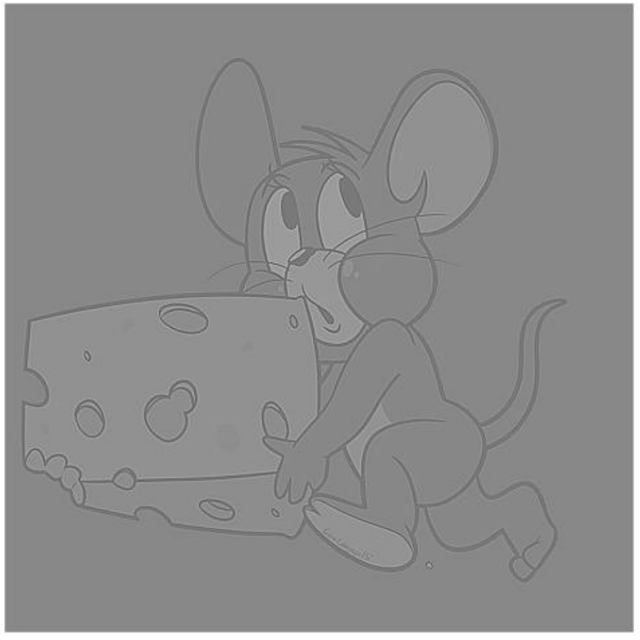
More weight to the center

Less harsh blurring

Sharpen kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Highlighting positions with

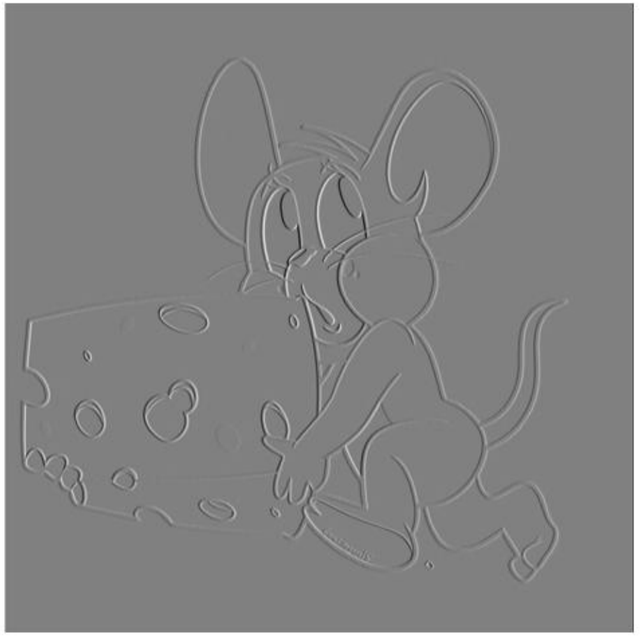


Sobel kernel for vertical boundary

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Sensitive to left-right variations

Detect vertical boundaries

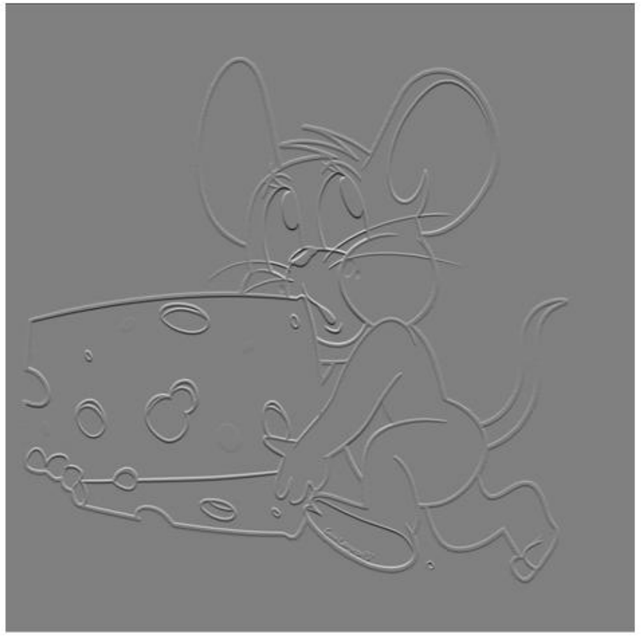


Sobel kernel for horizontal boundary

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

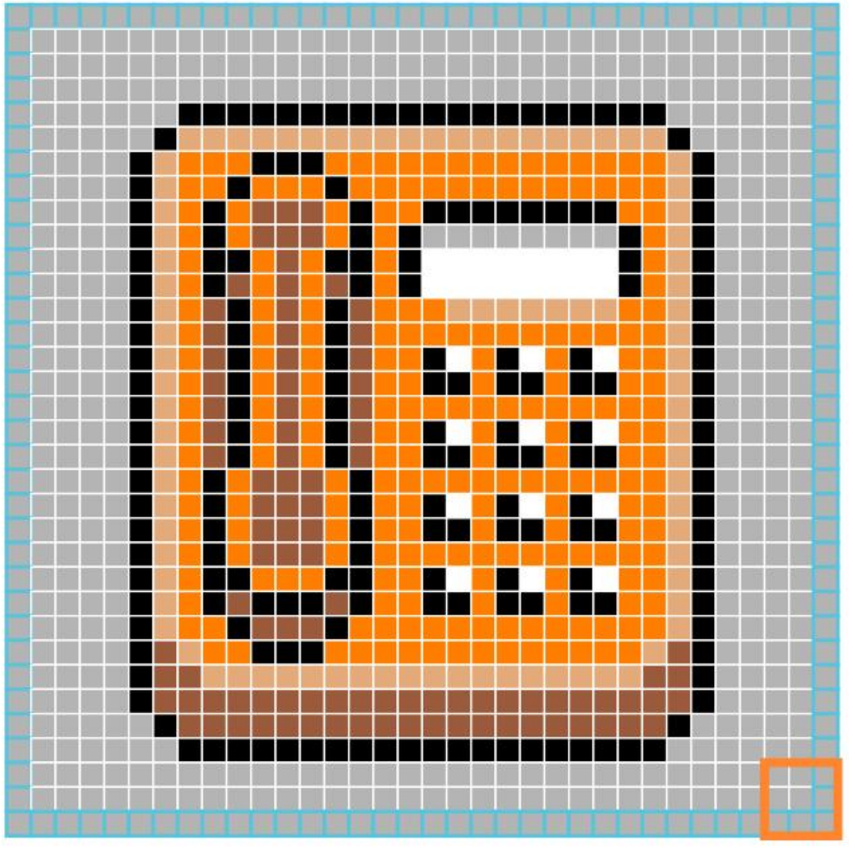
Sensitive to up-down variations

Detect horizontal boundaries



Another example

Original image



Kernel

−1	−1	−1
−1	9	−1
−1	−1	−1

Sharpen

−1	0	1
−2	0	2
−1	0	1

Sobel (vertical)

−1	−2	−1
0	0	0
1	2	1

Sobel (horizontal)

Multi-channel convolution

1. In general, the third dimension of kernels is the same as the input
2. Output size calculation
   - Input size: $d_C \times d_H \times d_W$
   - Kernel size: $d_C \times f_1 \times f_2$
   - Padding: $p_1 \times p_2 \quad (p_1 = p_2 = p$by default$)$
   - Stride: $s_1 \times s_2 \quad (s_1 = s_2 = s$by default$)$
   - Dilation rate: $d_1 \times d_2 \quad (d_1 = d_2 = d$by default$)$
   - Output size: $d'_H \times d'_W$

$$d'_H = \left\lfloor \frac{d_H + 2p_1 - d_1(f_1 - 1) - 1}{s_1} + 1 \right\rfloor, \quad d'_W = \left\lfloor \frac{d_W + 2p_2 - d_2(f_2 - 1) - 1}{s_2} + 1 \right\rfloor$$

Remarks

1. For multi-channel convolution, one kernel corresponds to a matrix
   - We may use multiple kernels with the same dimension to get a tensor
   - Different kernels are used to extract different information
2. By training a CNN, our goal is to determine the elements in those kernels
   - In order to efficiently extract information from images for certain tasks

Summary

1. Dramatically decrease number of parameters: Parameters of a certain kernel are shared within the same channel
2. High computation efficiency: GPU parallelization can be used for convolution
3. Spatial structure is considered: Convolution is applied to local areas
4. Local translation invariant: The same kernel is repeatedly used for different regions