Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Maximilian Bernhard, Niklas Strauß

Munich, 11.11.2022

Deep Learning and Artificial Intelligence

WS 2024/25

Exercise 4: Convolutional Neural Networks

Exercise 4-1 Convolutions

Given the following 5x5 input image with one channel:

5	5	2	5	5
5	5	2	5	5
7	7	5	7	7
5	5	2	5	5
5	5	2	5	5

Let's assume we have the following 3x3 filters:

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

0	1	0
1	-4	1
0	1	0

(a) Apply the given filters (by *cross-correlation*) to the above dataset, i.e.:

$$Y_{i,j} = (K \star X)_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K_{m,n} X_{i+m,j+n}.$$

where Y is the output and $M = N = 3$ are the kernel sizes. Use 'valid' padding and a stride of one. You can also write a small program (e.g. using the method scipy.signal.convolve2d()) for that purpose.

Detailed Solution

Solution: Channel 1:

11	0	-11
10	0	-10
11	0	-11

Channel 2:

-9	-10	-9
0	0	0
9	10	9

Channel 3:

-1	9	-1
-6	-2	-6
-1	9	-1

The output values are calculated as: $Y_{i,j} = (K \star X)_{i,j} \sum_{m=0}^{2} \sum_{n=0}^{2} K_{m,n} X_{i+m,j+n}$.

Examples for Channel 1:
$Y_{0,0} = 1 \cdot 5 + 0 \cdot 5 + (-1) \cdot 2 + 2 \cdot 5 + 0 \cdot 5 + (-2) \cdot 2 + 1 \cdot 7 + 0 \cdot 7 + (-1) \cdot 5 = 11.$

$Y_{1,1} = 1 \cdot 5 + 0 \cdot 2 + (-1) \cdot 5 + 2 \cdot 7 + 0 \cdot 5 + (-2) \cdot 7 + 1 \cdot 5 + 0 \cdot 2 + (-1) \cdot 5 = 0.$

Note that using *convolution* instead of *cross-correlation* would lead to inversed signs in the output (flip the kernel).

(b) Look at the structure of the filters. What do they do?

Detailed Solution

Filter 1 and 2 are Sobel operators. They detect vertical (1) and horizontal (2) edges by computing an approximation of the gradient of the image intensity. Note that they can be decomposed as follows:

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

where the first term is a smoothing term (the middle row of the convolved input section (3 x 3 part) counts twice as much as the outer ones) and the second term corresponds to the approximate derivative (between the left and the right column of the input).

The two Sobel operators can be combined to get the magnitude and angle of the gradients. Refer to https://de.wikipedia.org/wiki/Sobel_operator for more information.

Filter 3 is a Laplace operator. It detects rapid changes in intensity (e.g., edges).

Exercise 4-2 Backpropagtion through Convolutional Layers

Let the output of a convolutional layer with weights $W \in \mathbb{R}^{k \times k}$ and an input image $X \in \mathbb{R}^{d \times d}$ be given by the cross-correlation $Y = W \star X$ (stride = 1, valid padding).

(a) Derive the quantity $\frac{\partial Y_{i,j}}{\partial W_{u,v}}$!

Detailed Solution

$$Y_{i,j} = (W \star X)_{i,j} = \sum_{m,n} W_{m,n} X_{i+m,j+n}$$

$$\frac{\partial Y_{i,j}}{\partial W_{u,v}} = X_{i+u,j+v}$$

where m and n run from 0 to k. Note that $Y \in \mathbb{R}^{(d-k+1)\times(d-k+1)}$.

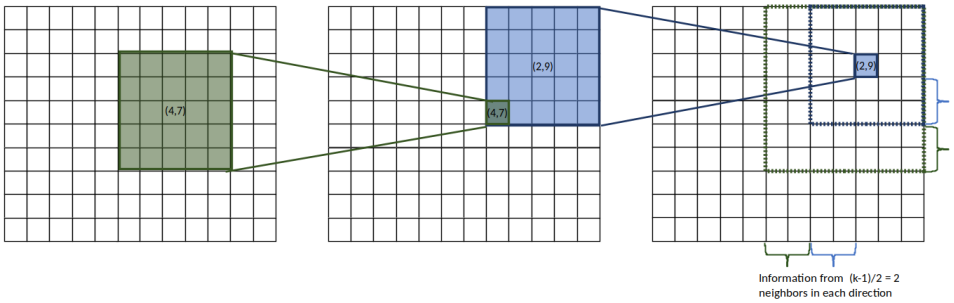
(b) Assume we have the following input image (640x428 pixels):

This image is an input to a convolutional neural network that has 10 convolutional layers, each with one channel and a 5x5 filter (padding: same, stride: 1x1).

Compute whether the top right neuron of the last layer in the CNN contains any information about the eye of the cat (assume the eye is at pixel 255x210)?

Detailed Solution

In every layer, a neuron gets information about its $\frac{k-1}{2}$ neighbors, where k is the kernel size (see figure below).



Thus, if we have $k = 5$ and ten layers, we get information from $10 \cdot \frac{4}{2} = 20$ pixels to the left/right and top/bottom.

$640 - 20 = 620 > 255$ and $428 - 20 = 408 > 210 \Rightarrow$ the neuron has no chance of containing any information about the cat's eyes.

Exercise 4-3 Equivariance of Convolutional Layers

Let X be an input image and K be a filter. For all $(x, y) \in \mathbb{Z}^2$ we define $T_{x,y}$ to be the translation operator that moves every point $X_{i,j}$ in the image by x in the x-direction and by y in the y-direction, i.e.:

$$T_{x,y}X_{i,j} = X_{i-x,j-y}.$$

Show that the *convolution* operator $*$ is translation-equivariant, i.e. commutes with translations:

$$T_{x,y}X * K = T_{x,y}(X * K).$$

Detailed Solution

Solution: Let $x, y, i, j \in \mathbb{Z}$. Remember that:

$$(X * K)_{i,j} = \sum_{m,n} X_{m,n}K_{i-m,j-n} = \sum_{m,n} X_{i-m,j-n}K_{m,n} = (K * X)_{i,j}.$$

We have:

$$\begin{aligned}
(T_{x,y}X * K)_{i,j} &= \sum_{m,n} T_{x,y}X_{m,n}\, K_{i-m,j-n} \\
&= \sum_{m,n} X_{m-x,n-y}\, K_{i-m,j-n} \\
&= \sum_{m',n'} X_{m',n'}\, K_{i-(m'+x),\, j-(n'+y)} \qquad (m' = m - x, n' = n - y) \\
&= \sum_{m',n'} X_{m',n'}\, K_{i-x-m',\, j-y-n'} \\
&= (X * K)_{i-x,\, j-y} \qquad \text{by definition} \\
&= T_{xy}(X * K)_{i,j} \qquad \square
\end{aligned}$$

Alternatively, one could see this in the following way:

$$
\begin{aligned}
(K * T_{x,y}X)_{i,j} &= \sum_{m,n} K_{m,n}\, T_{x,y}X_{i-m,j-n} \\
&= \sum_{m,n} K_{m,n}\, X_{i-m-x,j-n-y} \\
&= \sum_{m,n} K_{m,n}\, X_{i-x-m,j-y-n} \\
&= (K * X)_{i-x,\, j-y} \qquad \text{by definition} \\
&= T_{xy}(K * X)_{i,j} \qquad \square
\end{aligned}
$$

Exercise 4-4 Convolutional Neural Network in PyTorch

In Uni2Work / Moodle you find a Jupyter notebook asking you to implement CNNs in PyTorch.