Методы машинного обучения

Шорохов С.Г.

кафедра информационных технологий

Лекция 7. Сверточные нейронные сети

Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks, CNN) представляют собой глубокие сети MLP, использующие пространственные или временные отношения между различными элементами каждого слоя для построения иерархии признаков. В отличие от обычных сетей MLP, слои которых полностью связаны, в сетях CNN слои локализованы и разрежены. В частности, сети CNN очень эффективны для обработки изображений в качестве входных данных.

Сверточная нейронная сеть — это, по сути, локализованная и разреженная сеть MLP с прямой связью, предназначенная для использования пространственной и/или временной структуры входных данных. В обычной сети MLP все нейроны слоя l связаны со всеми нейронами слоя $l+1$. Напротив, сеть CNN соединяет некоторое подмножество нейронов в слое l с одним нейроном в следующем слое $l+1$. Различные скользящие окна, содержащие смежные подмножества нейронов в слое l, соединяются с разными нейронами в слое $l+1$. Кроме того, все эти скользящие окна предусматривают совместное использование параметров, то есть один и тот же набор весов, называемый фильтром, используется для всех скользящих окон. Наконец, различные фильтры используются для автоматического извлечения признаков из слоя l для использования слоем $l+1$.

Свертка 1D (одномерная свертка)

Для вектора $\mathbf{a} \in \mathbb{R}^k$ определим оператор суммирования sum, складывающий все элементы вектора, т.е.

$$\text{sum}(\mathbf{a}) = \sum_{i=1}^{k} a_i,\ \mathbf{a} = (a_1, ..., a_k)^T$$

Свертка 1D между вектором (изображением) $\mathbf{x} \in \mathbb{R}^n$ и фильтром $\mathbf{w} \in \mathbb{R}^k$ $(n \geqslant k)$, обозначаемая символом $*$, равна

$$\mathbf{x} * \mathbf{w} = (\text{sum}(\mathbf{x}_k(1) \odot \mathbf{w}), ..., \text{sum}(\mathbf{x}_k(n-k+1) \odot \mathbf{w}))^T,$$

где символ $\odot$ обозначает поэлементное произведение и для $i = \overline{1, n-k+1}$

$$\text{sum}(\mathbf{x}_k(i) \odot \mathbf{w}) = \sum_{j=1}^{k} x_{i+j-1} w_j = x_i w_1 + x_{i+1} w_2 + ... + x_{i+k-1} w_k$$

В результате свертки векторов $\mathbf{x} \in \mathbb{R}^n$ и $\mathbf{w} \in \mathbb{R}^k$ при $n \geqslant k$ получается вектор длины $n-k+1$. При $n = k$ значением свертки является скалярное произведение векторов $\mathbf{x}$ и $\mathbf{w}$.

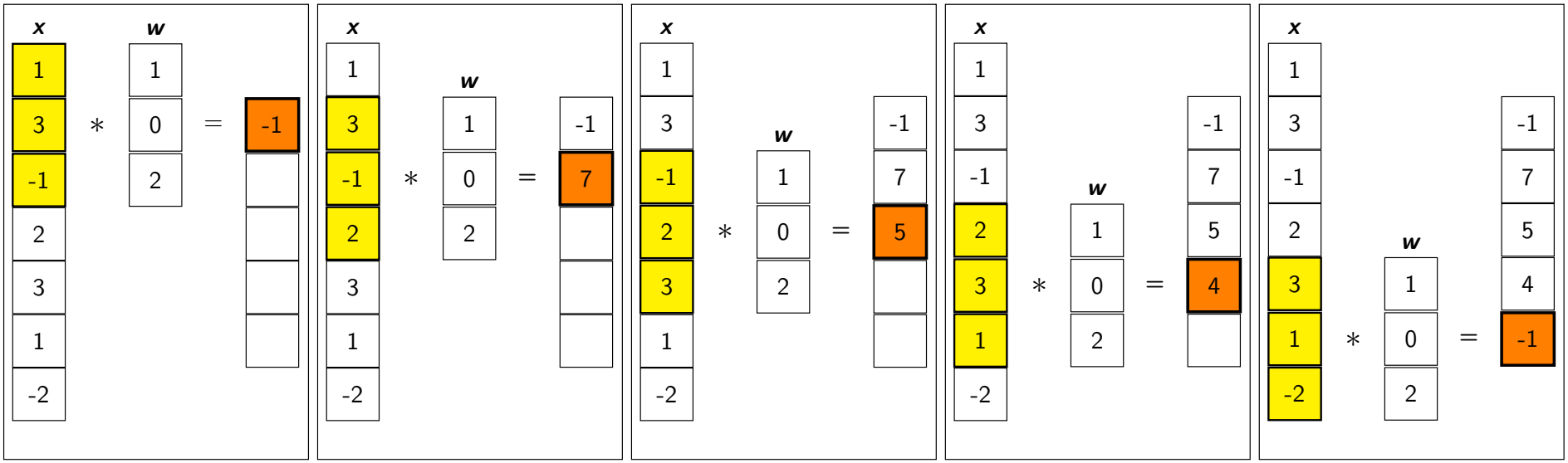
Пример свертки 1D

Пусть даны вектор $\mathbf{x}$ для $n = 7$ и фильтр $\mathbf{w} = (1, 0, 2)^T$ для $k = 3$. Первое окно для $\mathbf{x}$ равно $\mathbf{x}_3(1) = (1, 3, -1)^T$. Тогда

$$\text{sum}\left(\mathbf{x}_3(1) \odot \mathbf{w}\right) = \text{sum}\left((1, 3, -1)^T \odot (1, 0, 2)^T\right) = \text{sum}\left((1, 0, -2)^T\right) = -1$$

Свертка $\mathbf{x} * \mathbf{w}$ имеет длину $n - k + 1 = 7 - 3 + 1 = 5$ и равна

$$\mathbf{x} * \mathbf{w} = (-1, 7, 5, 4, -1)^T$$



Свертка 2D (двумерная свертка)

Для $k \times k$-матрицы $\mathbf{A} \in \mathbb{R}^{k \times k}$ определим оператор суммирования sum, складывающий все элементы матрицы, т.е.

$$\text{sum}\left(\mathbf{A}\right) = \sum_{i=1}^{k} \sum_{j=1}^{k} a_{i,j}, \; \mathbf{A} = \left(a_{i,j}\right)_{i,j=\overline{1,k}}$$

Свертка 2D между матрицей (изображением) $\mathbf{X} \in \mathbb{R}^{n \times n}$ и матрицей (фильтром) $\mathbf{W} \in \mathbb{R}^{k \times k}$ $(n \geqslant k)$ равна

$$\mathbf{X} * \mathbf{W} = \begin{pmatrix} \text{sum}\left(\mathbf{X}_k\left(1,1\right) \odot \mathbf{W}\right) & \ldots & \text{sum}\left(\mathbf{X}_k\left(1,n-k+1\right) \odot \mathbf{W}\right) \\ \ldots & \ldots & \ldots \\ \text{sum}\left(\mathbf{X}_k\left(n-k+1,1\right) \odot \mathbf{W}\right) & \ldots & \text{sum}\left(\mathbf{X}_k\left(n-k+1,n-k+1\right) \odot \mathbf{W}\right) \end{pmatrix}$$

где $\odot$ – это поэлементное произведение $\mathbf{X}_k\left(i,j\right)$ и $\mathbf{W}$ и при $i,j = 1,...,n-k+1$

$$\text{sum}\left(\mathbf{X}_k\left(i,j\right) \odot \mathbf{W}\right) = \sum_{a=1}^{k} \sum_{b=1}^{k} x_{i+a-1,j+b-1} w_{a,b} = x_{i,j} w_{1,1} + ... + x_{i+k-1,j+k-1} w_{k,k}$$

В результате свертки матриц $\mathbf{X} \in \mathbb{R}^{n \times n}$ и $\mathbf{W} \in \mathbb{R}^{k \times k}$ при $n \geqslant k$ получается $(n-k+1) \times (n-k+1)$-матрица.

Пример свертки 2D

Пусть дана матрица $\mathbf{X}$ для $n = 4$ и и фильтр $\mathbf{W}$ с размером окна $k = 2$. Свертка первого окна $\mathbf{X}$, а именно, $\mathbf{X}_2\,(1,1)$ с $\mathbf{W}$ равна

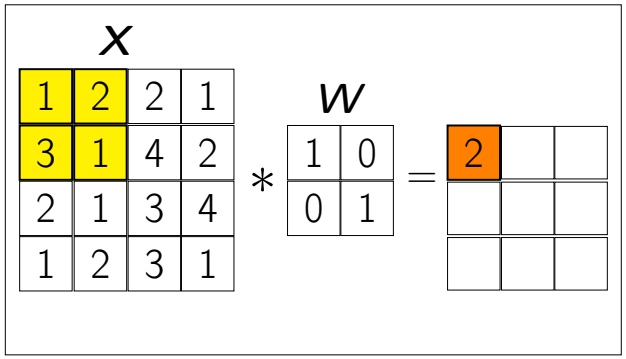
$$\text{sum}\left(\mathbf{X}_2\,(1,1) \odot \mathbf{W}\right) = \text{sum}\left(\begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) = \text{sum}\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) = 2$$

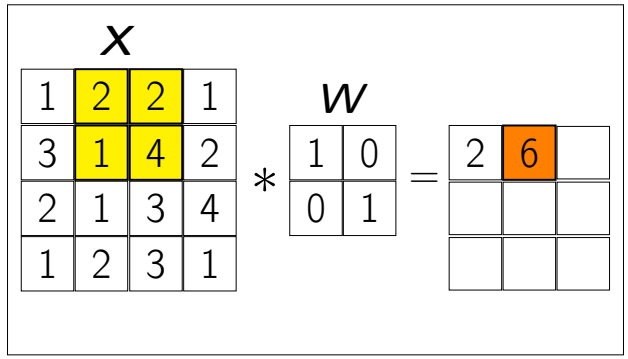
Шаги свертки для различных сдвигающихся окон размером 2×2 матрицы $\mathbf{X}$ с фильтром $\mathbf{W}$ показаны далее.

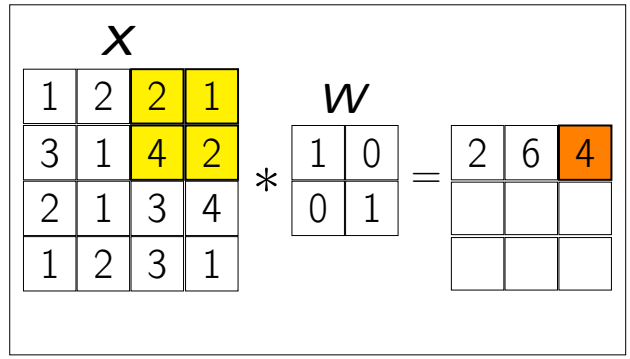
Свертка $\mathbf{X} * \mathbf{W}$ имеет размеры 3×3, так как $n - k + 1 = 4 - 2 + 1 = 3$, и равна

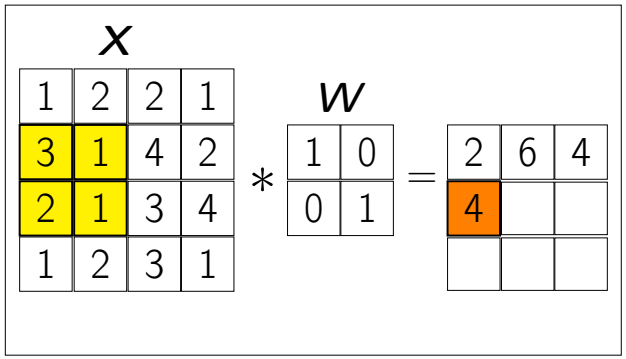
$$\mathbf{X} * \mathbf{W} = \begin{pmatrix} 2 & 6 & 4 \\ 4 & 4 & 8 \\ 4 & 4 & 4 \end{pmatrix}$$

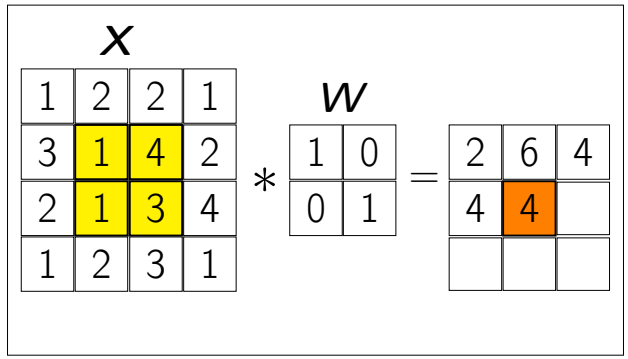
Пример свертки 2D

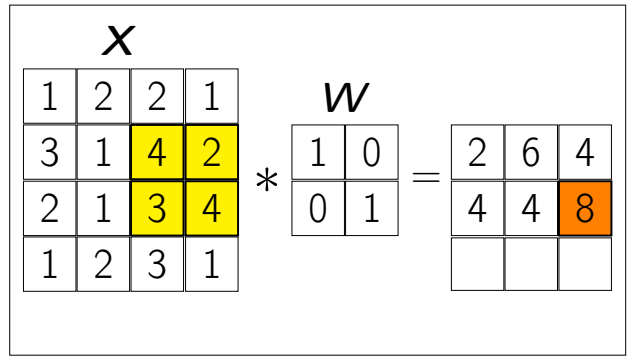


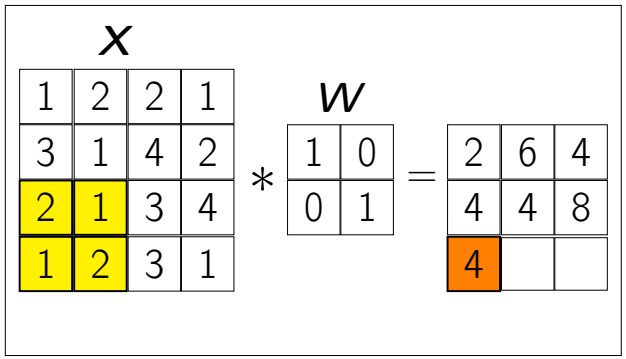


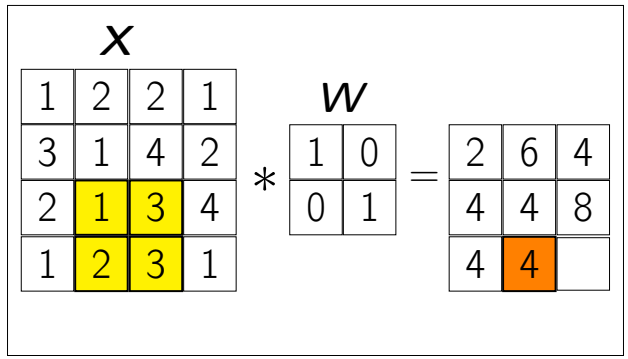


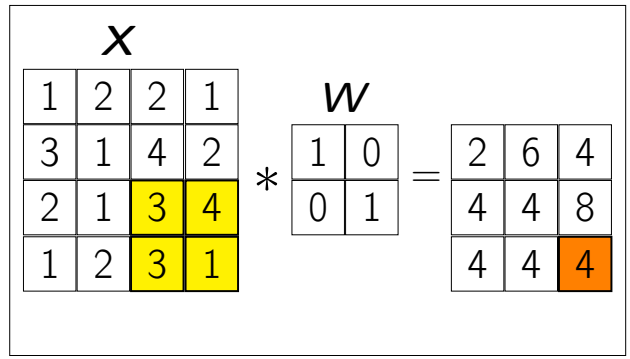












Свертка 3D (трехмерная свертка)

Пусть $\mathbf{W}$ – $k \times k \times r$-тензор весов, называемый 3D фильтром, где $k \leqslant n$ и $r \leqslant m$. Пусть $\mathbf{X}_k\,(i,j,q)$ обозначает $k \times k \times r$-подтензор $\mathbf{X}$, начиная со строки j, столбца j и канала q, где $1 \leqslant i,j \leqslant n-k+1$ и $1 \leqslant q \leqslant m-r+1$.

Для $k \times k \times r$-тензора $\mathbf{A} \in \mathbb{R}^{k \times k \times r}$ определим оператор суммирования sum, складывающий все элементы тензорв, т.е.

$$\text{sum}\,(\mathbf{A}) = \sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{q=1}^{r} a_{i,j,q},$$

где $a_{i,j,q}$ – элемент $\mathbf{A}$ со строкой i, столбцом j и каналом q. Пусть

$$\text{sum}\,(\mathbf{X}_k\,(i,j,q) \odot \mathbf{W}) = \sum_{a=1}^{k}\sum_{b=1}^{k}\sum_{c=1}^{k} x_{i+a-1,j+b-1,q+c-1} w_{a,b,c}$$

для $i,j = 1,...,n-k+1$ и $q = 1,...,m-r+1$.

В результате свертки тензоров $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$ и $\mathbf{W} \in \mathbb{R}^{k \times k \times r}$ при $n \geqslant k$ и $m \geqslant r$ получается тензор размерами $(n-k+1) \times (n-k+1) \times (m-r+1)$.

Свертка 3D

Свертка 3D тензоров $\mathbf{X}$ и $\mathbf{W}$, обозначаемая как $\mathbf{X} * \mathbf{W}$, определяется как

$$
\mathbf{X}*\mathbf{W} = \left(\begin{array}{ccc}
\operatorname{sum}\Big(\mathbf{X}_k(1,1,1)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(1,n-k+1,1)\odot\mathbf{W}\Big) \\
\operatorname{sum}\Big(\mathbf{X}_k(2,1,1)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(2,n-k+1,1)\odot\mathbf{W}\Big) \\
\cdots & \cdots & \cdots \\
\operatorname{sum}\Big(\mathbf{X}_k(n-k+1,1,1)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(n-k+1,n-k+1,1)\odot\mathbf{W}\Big)
\end{array}\right)
$$

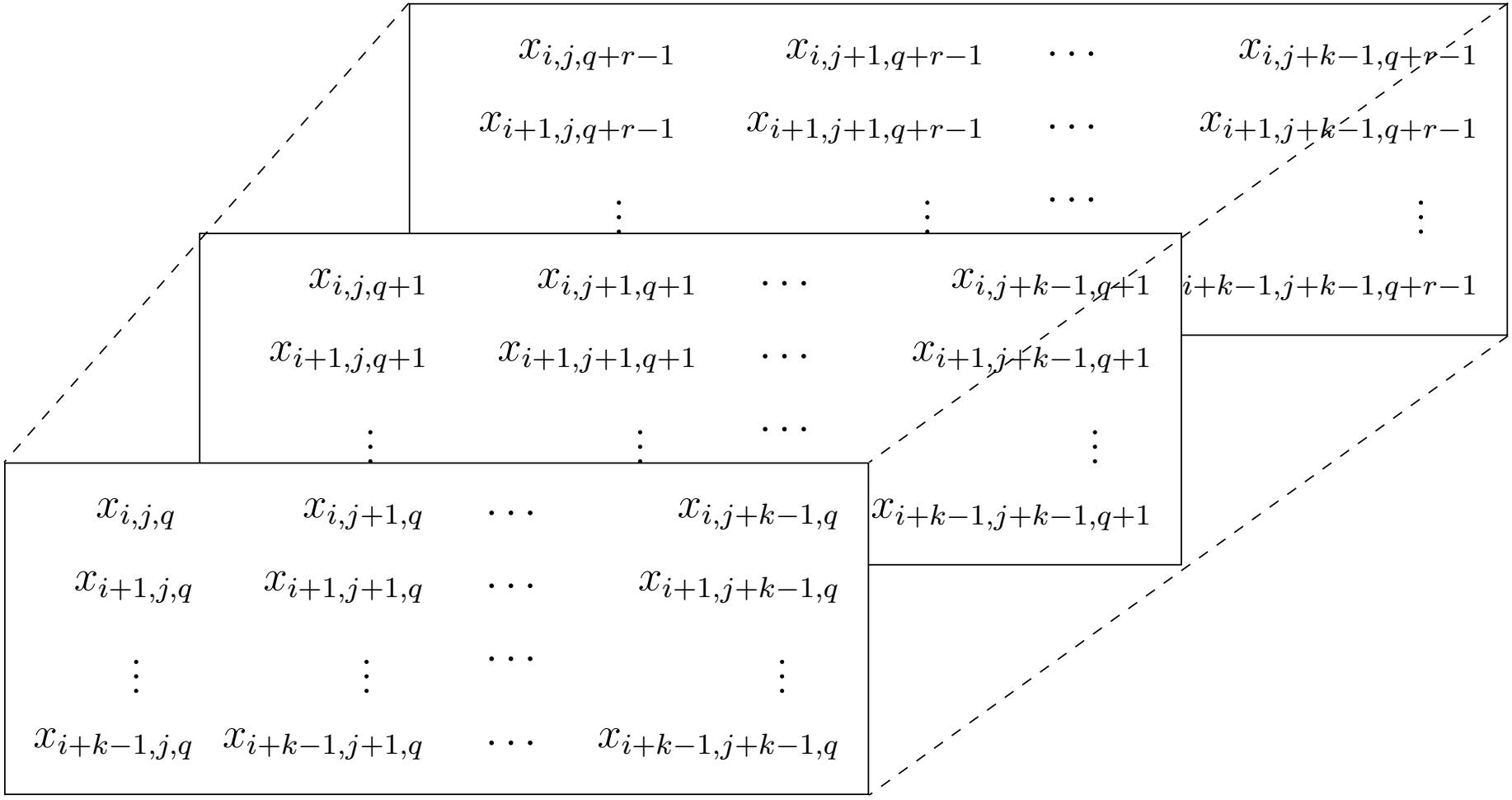
$$
\left(\begin{array}{ccc}
\operatorname{sum}\Big(\mathbf{X}_k(1,1,2)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(1,n-k+1,2)\odot\mathbf{W}\Big) \\
\operatorname{sum}\Big(\mathbf{X}_k(2,1,2)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(2,n-k+1,2)\odot\mathbf{W}\Big) \\
\cdots & \cdots & \cdots \\
\operatorname{sum}\Big(\mathbf{X}_k(n-k+1,1,2)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(n-k+1,n-k+1,2)\odot\mathbf{W}\Big)
\end{array}\right)
$$

$$
\left(\begin{array}{ccc}
\vdots & \vdots & \vdots \\
\operatorname{sum}\Big(\mathbf{X}_k(1,1,m-r+1)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(1,n-k+1,m-r+1)\odot\mathbf{W}\Big) \\
\operatorname{sum}\Big(\mathbf{X}_k(2,1,m-r+1)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(2,n-k+1,m-r+1)\odot\mathbf{W}\Big) \\
\cdots & \cdots & \cdots \\
\operatorname{sum}\Big(\mathbf{X}_k(n-k+1,1,m-r+1)\odot\mathbf{W}\Big) & \cdots & \operatorname{sum}\Big(\mathbf{X}_k(n-k+1,n-k+1,m-r+1)\odot\mathbf{W}\Big)
\end{array}\right)
$$

где $\odot$ – поэлементное произведение $\mathbf{X}_k\,(i,j,q)$ и $\mathbf{W}$.

3D подтензор $\mathbf{X}_k\,(i,j,q)$

Подтензор $\mathbf{X}_k\,(i,j,q)$ размерами $k \times k \times r$ содержит элементы тензора $\mathbf{X}$, начиная со строки i, столбца j и канала q:



Пример свертки 3D

Пусть даны $3 \times 3 \times 3$-тензор $\mathbf{X}$ $(n = 3,\, m = 3)$

$$\mathbf{X} = \left(\begin{array}{ccc|ccc|ccc} 1 & -1 & 3 & 2 & 1 & 3 & 1 & -2 & 4 \\ 2 & 1 & 4 & 3 & -1 & 1 & 2 & 1 & -2 \\ 3 & 1 & 2 & 1 & 1 & -2 & 1 & 3 & -1 \end{array}\right)$$

и $2 \times 2 \times 3$-фильтр $\mathbf{W}$ с размером окна $k = 2$ и $r = 3$:

$$\mathbf{W} = \left(\begin{array}{cc|cc|cc} 1 & 1 & 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 1 & 0 \end{array}\right)$$

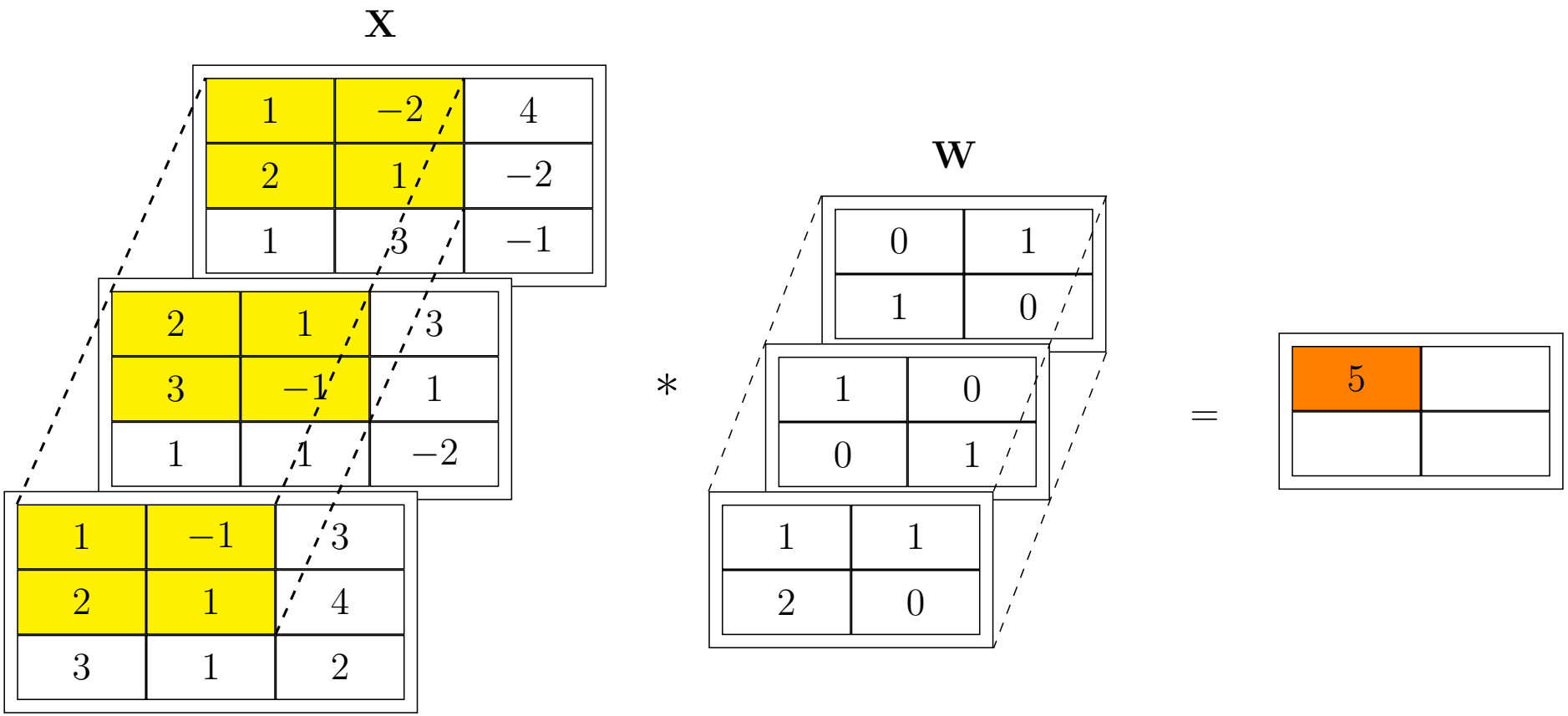
Свертка первого окна $\mathbf{X}$, а именно, $\mathbf{X}_2\,(1,1,1)$, с фильтром $\mathbf{W}$ равна

$$\operatorname{sum}\left(\left(\begin{array}{cc|cc|cc} 1 & -1 & 2 & 1 & 1 & -2 \\ 2 & 1 & 3 & -1 & 2 & 1 \end{array}\right) \odot \left(\begin{array}{cc|cc|cc} 1 & 1 & 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 1 & 0 \end{array}\right)\right) =$$

$$= \operatorname{sum}\left(\left(\begin{array}{cc|cc|cc} 1 & -1 & 2 & 0 & 0 & -2 \\ 4 & 0 & 0 & -1 & 2 & 0 \end{array}\right)\right) = 5$$

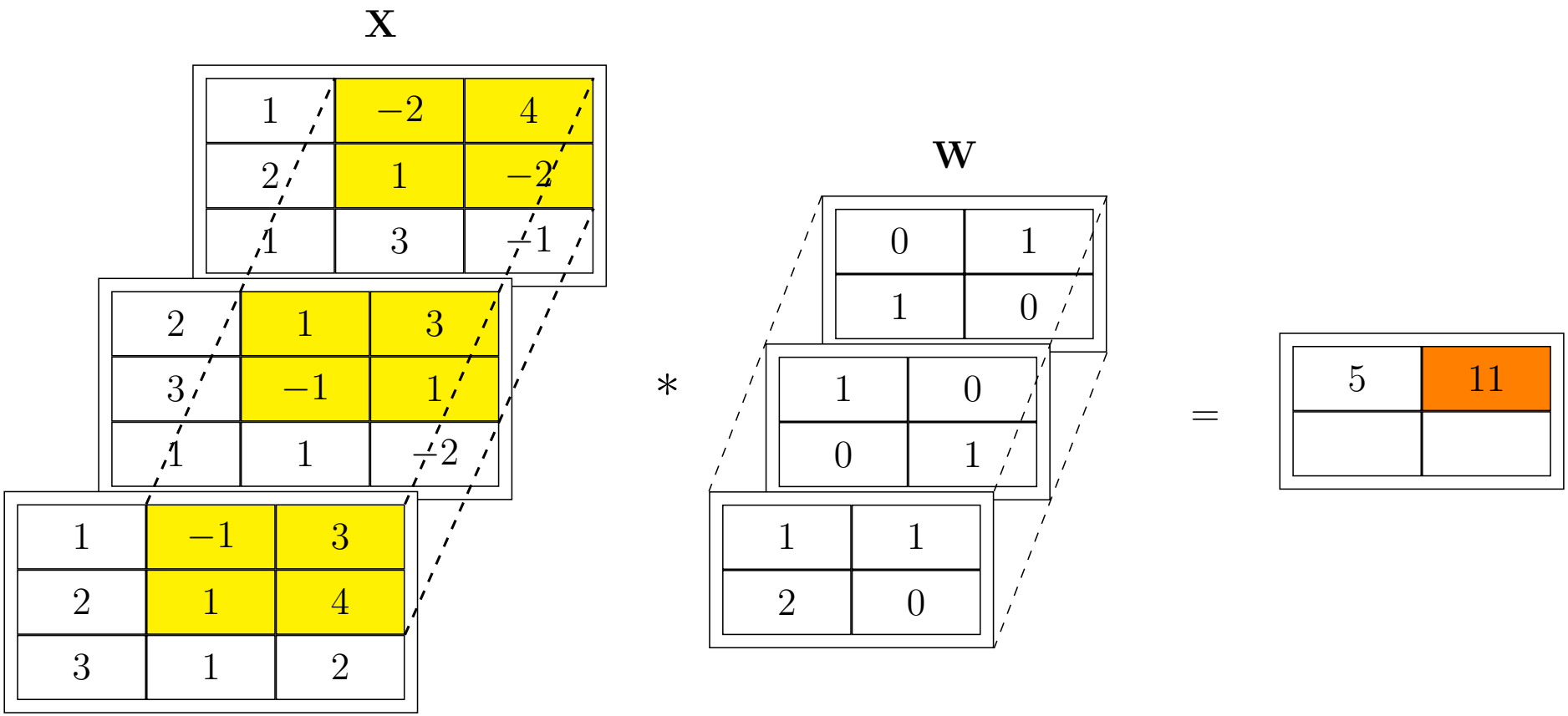
Пример свертки 3D – верхний левый элемент

Для $3 \times 3 \times 3$-тензора $\mathbf{X}$ ($n = 3$, $m = 3$) и $2 \times 2 \times 3$-фильтра $\mathbf{W}$ с размером окна $k = 2$ и $r = 3$ верхний левый элемент свертки равен:



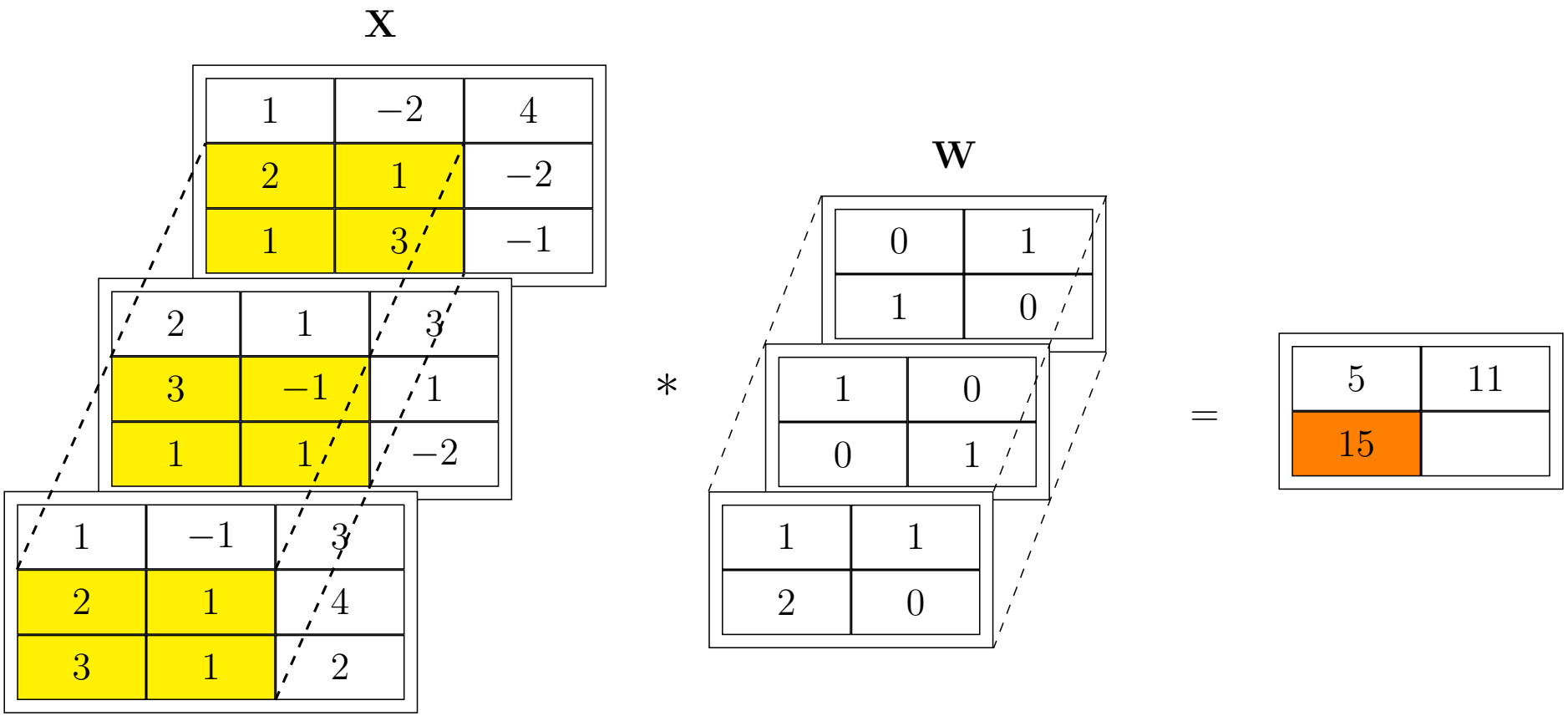
Пример свертки 3D – верхний правый элемент

Для $3 \times 3 \times 3$-тензора $\mathbf{X}$ ($n = 3$, $m = 3$) и $2 \times 2 \times 3$-фильтра $\mathbf{W}$ с размером окна $k = 2$ и $r = 3$ верхний правый элемент свертки равен:



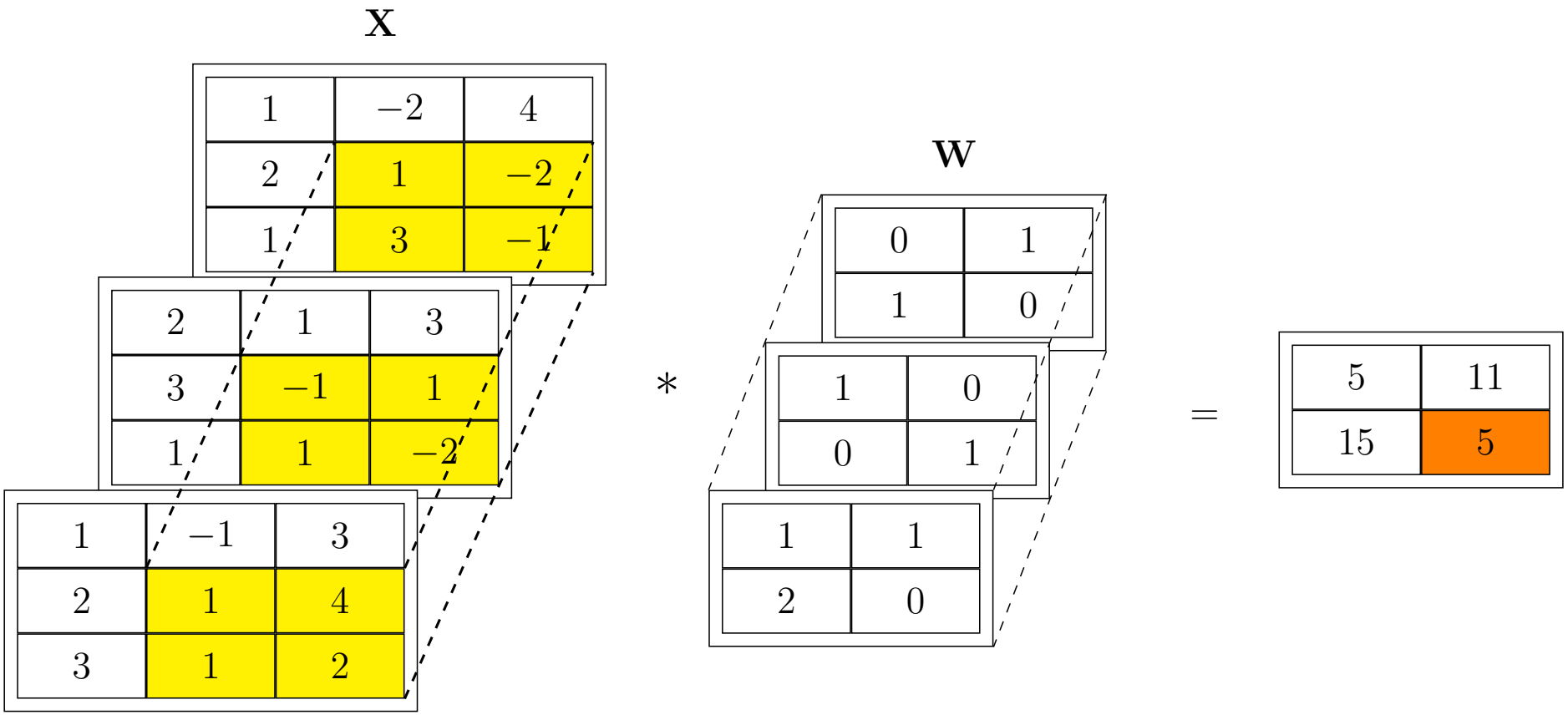
Пример свертки 3D – нижний левый элемент

Для $3 \times 3 \times 3$-тензора $\mathbf{X}$ ($n = 3$, $m = 3$) и $2 \times 2 \times 3$-фильтра $\mathbf{W}$ с размером окна $k = 2$ и $r = 3$ нижний левый элемент свертки равен:



Пример свертки 3D – нижний правый элемент

Для $3 \times 3 \times 3$-тензора $\mathbf{X}$ ($n = 3$, $m = 3$) и $2 \times 2 \times 3$-фильтра $\mathbf{W}$ с размером окна $k = 2$ и $r = 3$ нижний правый элемент свертки равен:



Свертка $\mathbf{X} * \mathbf{W}$ имеет размеры 2×2, так как $n - k + 1 = 3 - 2 + 1 = 2$ и $r = m = 3$, и равна $\mathbf{X} * \mathbf{W} = \begin{pmatrix} 5 & 11 \\ 15 & 5 \end{pmatrix}$.

Смещения и функции активации

Рассмотрим роль нейронов смещения и функций активации для тензора нейронов в слое l. Пусть $\mathbf{Z}^l$ – тензор размера $n_l \times n_l \times m_l$ из нейронов в слое l, где $z^l_{i,j,q}$ обозначает значение нейрона в строке i, столбце j и канале q для слоя l для $1 \leqslant i, j \leqslant n_l$ и $1 \leqslant q \leqslant m_l$.

Пусть $\mathbf{W}$ – 3D фильтр размера $k \times k \times m_l$. В результате свертки тензоров $\mathbf{Z}^l$ и $\mathbf{W}$ получается матрица размерами $(n_l - k + 1) \times (n_l - k + 1)$ в слое $l + 1$. Пусть $b \in \mathbb{R}$ – скалярное смещение фильтра $\mathbf{W}$ и пусть $\mathbf{Z}^l_k(i, j)$ обозначает $k \times k \times m_l$-подтензор $\mathbf{Z}^l$ в позиции (i, j). Тогда чистый выход на нейроне $z^{l+1}_{i,j}$ в слое $l + 1$ задается как

$$\text{net}^{l+1}_{i,j} = \text{sum}\left(\mathbf{Z}^l_k(i, j) \odot \mathbf{W}\right) + b$$

и значение нейрона $z^{l+1}_{i,j}$ получается применением некоторой функции активации f к чистому выходу

$$z^{l+1}_{i,j} = f\left(\text{net}^{l+1}_{i,j}\right) = f\left(\text{sum}\left(\mathbf{Z}^l_k(i, j) \odot \mathbf{W}\right) + b\right)$$

Функция активации может быть любой из применяемых в нейронных сетях, например, тождественная функция, сигмоида, гиперболический тангенс, ReLU и т.п.

Множественные 3D фильтры

В результате применения 3D фильтра $\mathbf{W}$ с соответствующим смещением b получается $(n_l - k + 1) \times (n_l - k + 1)$ матрица в слое $l + 1$.

Если необходимы m_{l+1} каналов в слое $l + 1$, то требуются m_{l+1} различных $k \times k \times m_l$ фильтров $\mathbf{W}_q$ с соответствующим смещением b_q, чтобы получить $(n_l - k + 1) \times (n_l - k + 1) \times m_{l+1}$ тензор в слое $l + 1$:

$$\mathbf{Z}^{l+1} = \left\{ z_{i,j,q}^{l+1} = f\left(\text{sum}\left(\mathbf{Z}_k^l\,(i,j) \odot \mathbf{W}_q\right) + b_q\right)\right\}_{i,j=1,\ldots,n_l-k+1,\; q=1,\ldots,m_{l+1}}$$

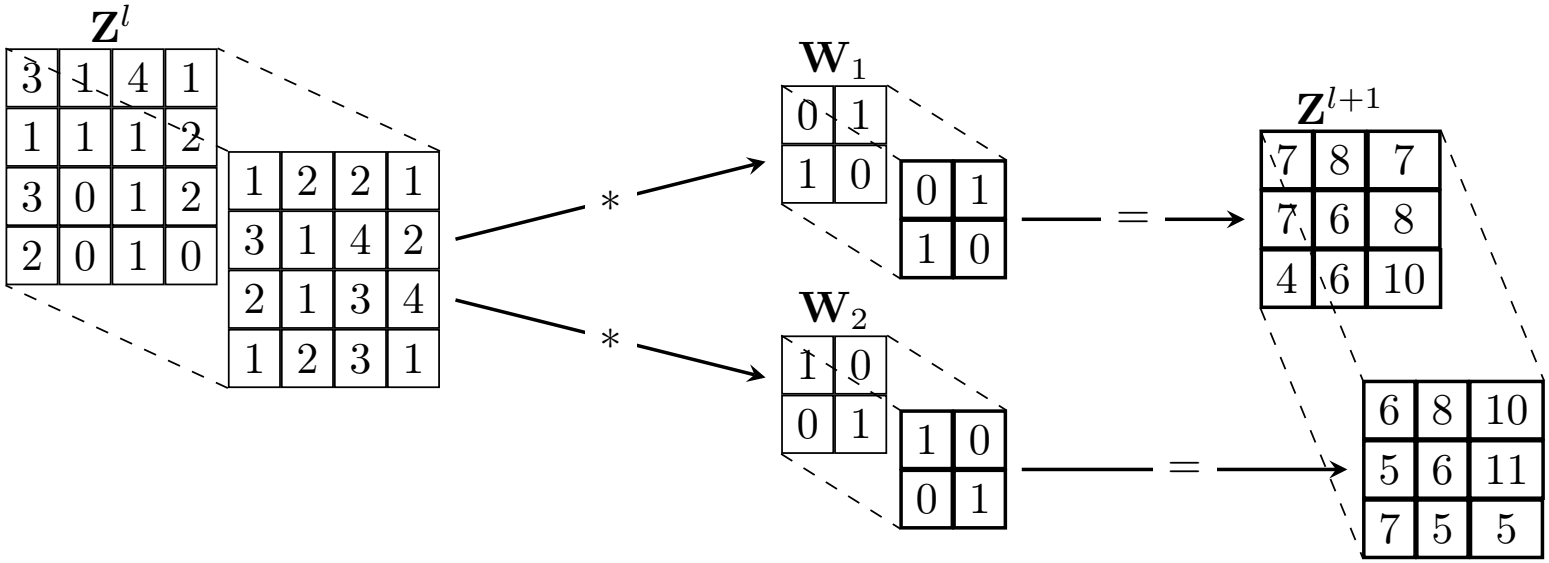
Таким образом, сверточный слой принимает в качестве входных данных $n_l \times n_l \times m_l$ тензор $\mathbf{Z}^l$ нейронов из слоя l, а затем вычисляет $n_{l+1} \times n_{l+1} \times m_{l+1}$ тензор $\mathbf{Z}^{l+1}$ нейронов для следующего слоя $l + 1$ при помощи свертки $\mathbf{Z}^l$ с набором из m_{l+1} различных 3D фильтров размером $k \times k \times m_l$ с последующим добавлением смещения и применением некоторой нелинейной функции активации f.

Каждый 3D фильтр, примененный к $\mathbf{Z}^l$, приводит к созданию нового канала в слое $l + 1$. Таким образом, m_{l+1} фильтров используются для получения m_{l+1} каналов в слое $l + 1$.

Пример множественных 3D фильтров

В примере используется $4 \times 4 \times 2$-тензор $\mathbf{Z}^l$ $(n = 4, m = 2)$ и два различных $2 \times 2 \times 2$-фильтра $\mathbf{W}_1$ и $\mathbf{W}_2$ с размером окна $k = 2$ и $r = 2$. Так как $r = m = 2$, свертка $\mathbf{Z}^l$ и $\mathbf{W}_i$ $(i = 1, 2)$ представляет собой 3×3-матрицу $(n - k + 1 = 4 - 2 + 1 = 3)$.

Фильтр $\mathbf{W}_1$ создает один канал и фильтр $\mathbf{W}_2$ создает другой канал, поэтому тензор для следующего слоя $\mathbf{Z}^{l+1}$ имеет размеры $3 \times 3 \times 2$ с двумя каналами (по одному на фильтр):



Техники padding и striding

Одна из проблем с операцией свертки заключается в том, что размер тензоров будет обязательно уменьшаться в каждом последующем слое сети CNN.

Если слой l имеет размеры $n_l \times n_l \times m_l$ и используются фильтры размера $k \times k \times m_l$, то для каждого канала тензор в слое $l+1$ будет иметь размеры $(n_l - k + 1) \times (n_l - k + 1)$. То есть количество строк и столбцов для каждого последующего тензора будет уменьшаться на $k - 1$, и это ограничит количество слоев, которые сможет иметь сеть CNN.

Техника padding (заполнение)

Простое решение в технике padding (заполнение) состоит в том, чтобы заполнить каждый тензор как по строкам, так и по столбцам в каждом канале некоторым значением по умолчанию, обычно нулем.

Для единообразия всегда добавляется одинаковое количество строк вверху и внизу, а также одинаковое количество столбцов слева и справа.

После добавления p строк/столбцов размер тензора слоя l становится равным $(n_l + 2p) \times (n_l + 2p) \times m_l$. Предположим, что каждый фильтр имеет размеры $k \times k \times m_l$ и что всего имеется m_{l+1} фильтров, тогда размер тензора слоя $l+1$ будет равен $(n_l + 2p - k + 1) \times (n_l + 2p - k + 1) \times m_l$. Желательно сохранить размер результирующего тензора, чтобы, по возможности

$$n_l + 2p - k + 1 \geqslant n_l,$$

поэтому используют значение $p = \left[\frac{k-1}{2}\right]$, где $[*]$ – целая часть числа.

Применяя технику заполнения padding, можно строить сколь угодно глубокие нейронные сети архитектуры CNN.

Заполнение (padding) для свертки 2D

На рисунках показана двумерная свертка без заполнения ($p = 0$) и с заполнением ($p = 1$). Производится 2D свертка 5×5-матрицы $\mathbf{Z}^l$ ($n = 5$) с фильтром 3×3 $\mathbf{W}$ ($k = 3$), что приводит к матрице 3×3, поскольку $n-k+1=5-3+1=3$. Поэтому размер следующего слоя $\mathbf{Z}^{l+1}$ уменьшился.

При $p = 1$ размер слоя $\mathbf{Z}^{l+1}$ сохраняется. Таким образом, при помощи заполнения можно использовать столько слоев свертки, сколько необходимо, без уменьшения размера слоев.

$\mathbf{Z}^l$

1	2	2	1	1
3	1	4	2	1
2	1	3	4	3
1	2	3	1	1
4	1	3	2	1

$*$ $\mathbf{W}$

1	0	0
0	1	1
0	1	0

$=$ $\mathbf{Z}^{l+1}$

7	11	9
9	11	12
8	8	7

$\mathbf{Z}^l$

0	0	0	0	0	0	0
0	1	2	2	1	1	0
0	3	1	4	2	1	0
0	2	1	3	4	3	0
0	1	2	3	1	1	0
0	4	1	3	2	1	0
0	0	0	0	0	0	0

$*$ $\mathbf{W}$

1	0	0
0	1	1
0	1	0

$=$ $\mathbf{Z}^{l+1}$

6	5	7	4	2
6	7	11	9	5
4	9	11	12	6
7	8	8	7	6
5	5	7	6	2

Техника striding (движение шагами)

Движение шагами (striding) часто используется для уменьшения количества скользящих окон, используемых в свертках. Вместо рассмотрения всех возможных окон индекс как по строкам, так и по столбцам увеличивается на целочисленное значение $s \geqslant 1$, называемое шагом. Свертка 3D тензора $\mathbf{Z}^l$ размерами $n_l \times n_l \times m_l$ и фильтра $\mathbf{W}$ размерами $k \times k \times m_l$ с использованием шага s дает следующее:

$$\mathbf{Z}^l * \mathbf{W} = \begin{pmatrix} \operatorname{sum}\left(\mathbf{Z}_k^l(1,1) \odot \mathbf{W}\right) & \operatorname{sum}\left(\mathbf{Z}_k^l(1,1+s) \odot \mathbf{W}\right) & \cdots & \operatorname{sum}\left(\mathbf{Z}_k^l(1,1+t\cdot s) \odot \mathbf{W}\right) \\ \operatorname{sum}\left(\mathbf{Z}_k^l(1+s,1) \odot \mathbf{W}\right) & \operatorname{sum}\left(\mathbf{Z}_k^l(1+s,1+s) \odot \mathbf{W}\right) & \cdots & \operatorname{sum}\left(\mathbf{Z}_k^l(1+s,1+t\cdot s) \odot \mathbf{W}\right) \\ \vdots & \vdots & \cdots & \vdots \\ \operatorname{sum}\left(\mathbf{Z}_k^l(1+t\cdot s,1) \odot \mathbf{W}\right) & \operatorname{sum}\left(\mathbf{Z}_k^l(1+t\cdot s,1+s) \odot \mathbf{W}\right) & \cdots & \operatorname{sum}\left(\mathbf{Z}_k^l(1+t\cdot s,1+t\cdot s) \odot \mathbf{W}\right) \end{pmatrix}$$

где $t = \left[\frac{n_l - k}{s}\right]$.

При использовании шага s результатом свертки тензора $\mathbf{Z}^l \in \mathbb{R}^{n_l \times n_l \times m_l}$ с фильтром $\mathbf{W} \in \mathbb{R}^{k \times k \times m_l}$ является $(t+1) \times (t+1)$-матрица.

Пример свертки с движением шагами (striding)

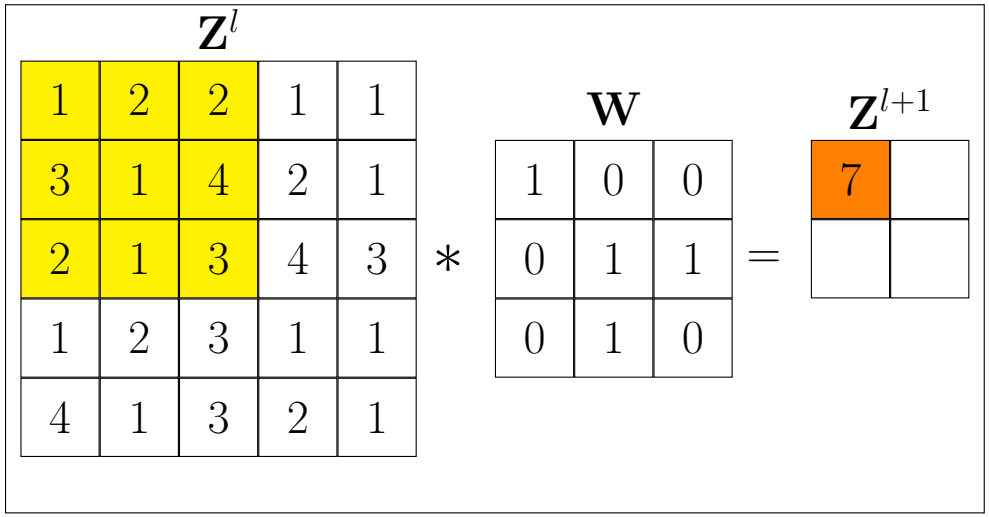
В примере показана двумерная свертка с использованием шага $s = 2$ на матрице $\mathbf{Z}^l$ размерами 5×5 ($n_l = 5$) с фильтром $\mathbf{W}$ размерами 3×3 ($k = 3$). Вместо матрицы $\mathbf{Z}^{l+1}$ размерами 3×3 для шага по умолчанию, равного единице, получаем матрицу $\mathbf{Z}^{l+1}$ размерами $(t+1) \times (t+1) = 2 \times 2$, поскольку

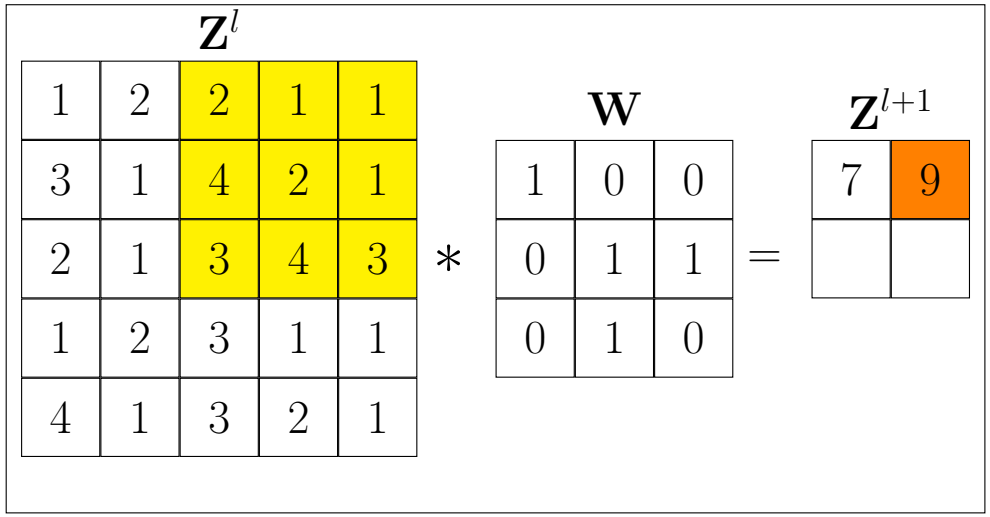
$$t = \left[\frac{n_l - k}{s}\right] = \left[\frac{5-3}{2}\right] = 1.$$

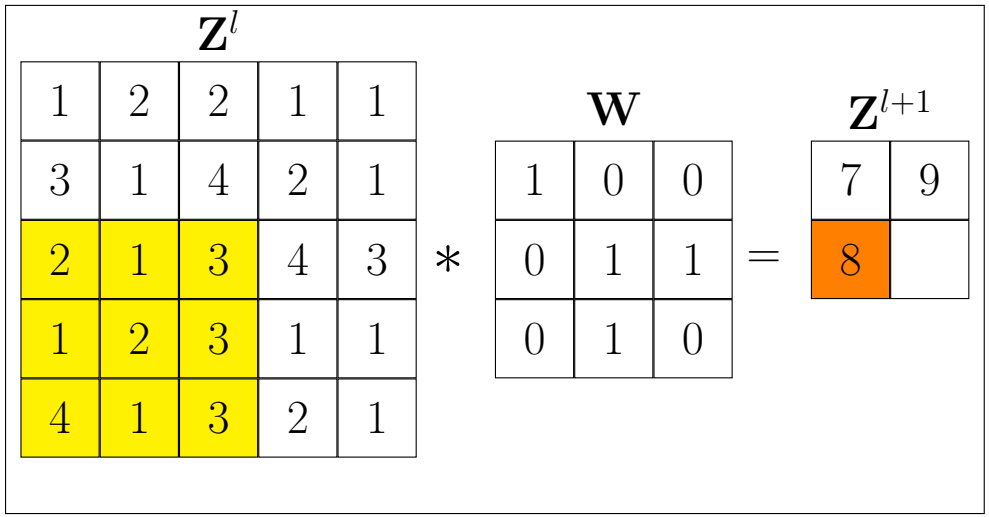
При использовании шага индекс следующего окна увеличивается на шаг s по строкам и столбцам.

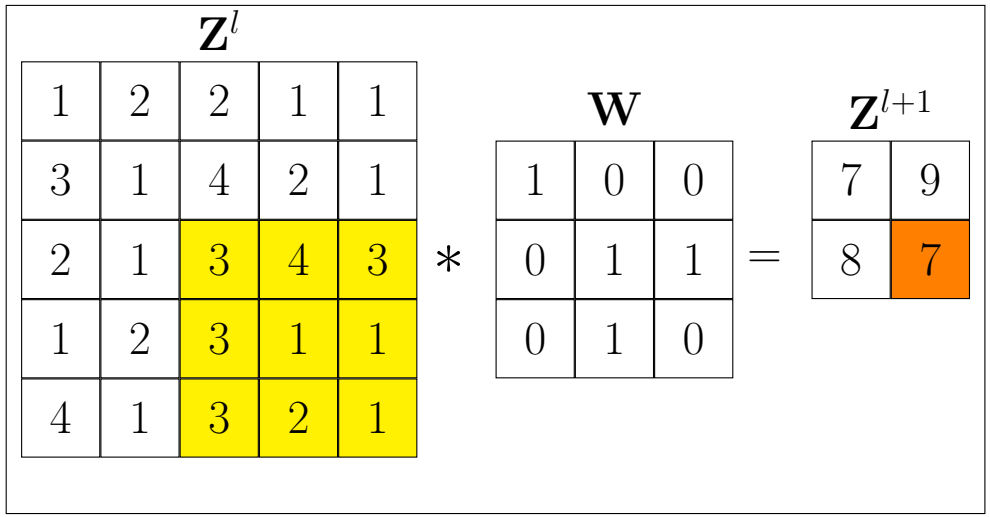
Например, первое окно – это $\mathbf{Z}_3^l(1,1)$, второе окно – $\mathbf{Z}_3^l(1,1+s) = \mathbf{Z}_3^l(1,3)$. Затем движемся от первого окна вниз на шаг $s = 2$, так что третье окно имеет вид $\mathbf{Z}_3^l(1+s,1) = \mathbf{Z}_3^l(3,1)$, а последнее окно имеет вид $\mathbf{Z}_3^l(3,1+s) = \mathbf{Z}_3^l(3,3)$.

Пример свертки с движением шагами ($s = 2$)









Пулинг (pooling)

Сети CNN также используют другие типы агрегирования (пулинга) в дополнение к суммированию, такие как среднее значение и максимальное значение.

Среднее значение (Avg-Pooling) по поэлементному произведению $\mathbf{Z}_k^l(i,j,q)$ и $\mathbf{W}$ равно:

$$\begin{aligned}\operatorname{avg}\left(\mathbf{Z}_k^l(i,j,q)\odot\mathbf{W}\right) &= \operatorname{avg}_{\substack{a=1,2,\cdots,k\\ b=1,2,\cdots,k\\ c=1,2,\cdots,r}}\left\{z^l_{i+a-1,j+b-1,q+c-1}\cdot w_{a,b,c}\right\}\\ &= \frac{1}{k^2\cdot r}\cdot\operatorname{sum}\left(\mathbf{Z}_k^l(i,j,q)\odot\mathbf{W}\right)\end{aligned}$$

Максимум (Max-Pooling) по поэлементному произведению $\mathbf{Z}_k^l(i,j,q)$ и $\mathbf{W}$ равен:

$$\max\left(\mathbf{Z}_k^l(i,j,q)\odot\mathbf{W}\right) = \max_{\substack{a=1,2,\cdots,k\\ b=1,2,\cdots,k\\ c=1,2,\cdots,r}}\left\{z^l_{i+a-1,j+b-1,q+c-1}\cdot w_{a,b,c}\right\}$$

Макс-пулинг (Max-Pooling) в сетях CNN

Обычно макс-пулинг (max-pooling) используется чаще, чем avg-pooling. Кроме того, для агрегирования очень часто шаг устанавливается равным размеру фильтра ($s = k$), так что функция агрегирования применяется к непересекающимся окнам размера $k \times k$ в каждом канале в $\mathbf{Z}^l$.

Что еще более важно, при агрегировании фильтр $\mathbf{W}$ по умолчанию принимается равным тензору размера $k \times k \times 1$, все веса которого фиксированы и равны 1, так что $\mathbf{W} = \mathbf{1}_{k \times k \times 1}$.

Другими словами, веса фильтра фиксируются на 1 и не обновляются во время обратного распространения. Кроме того, фильтр использует фиксированное нулевое смещение (то есть $b = 0$).

Наконец, обратите внимание, что пулинг неявно использует функцию тождественной активации. В этом случае свертка $\mathbf{Z}^l \in \mathbb{R}^{n_l \times n_l \times m_l}$ с фильтром $\mathbf{W} \in \mathbb{R}^{k \times k \times 1}$ с использованием шага $s = k$ создает тензор $\mathbf{Z}^{l+1}$ размером $\left[\frac{n_l}{s}\right] \times \left[\frac{n_l}{s}\right] \times m_l$.

Пример использования Max-Pooling

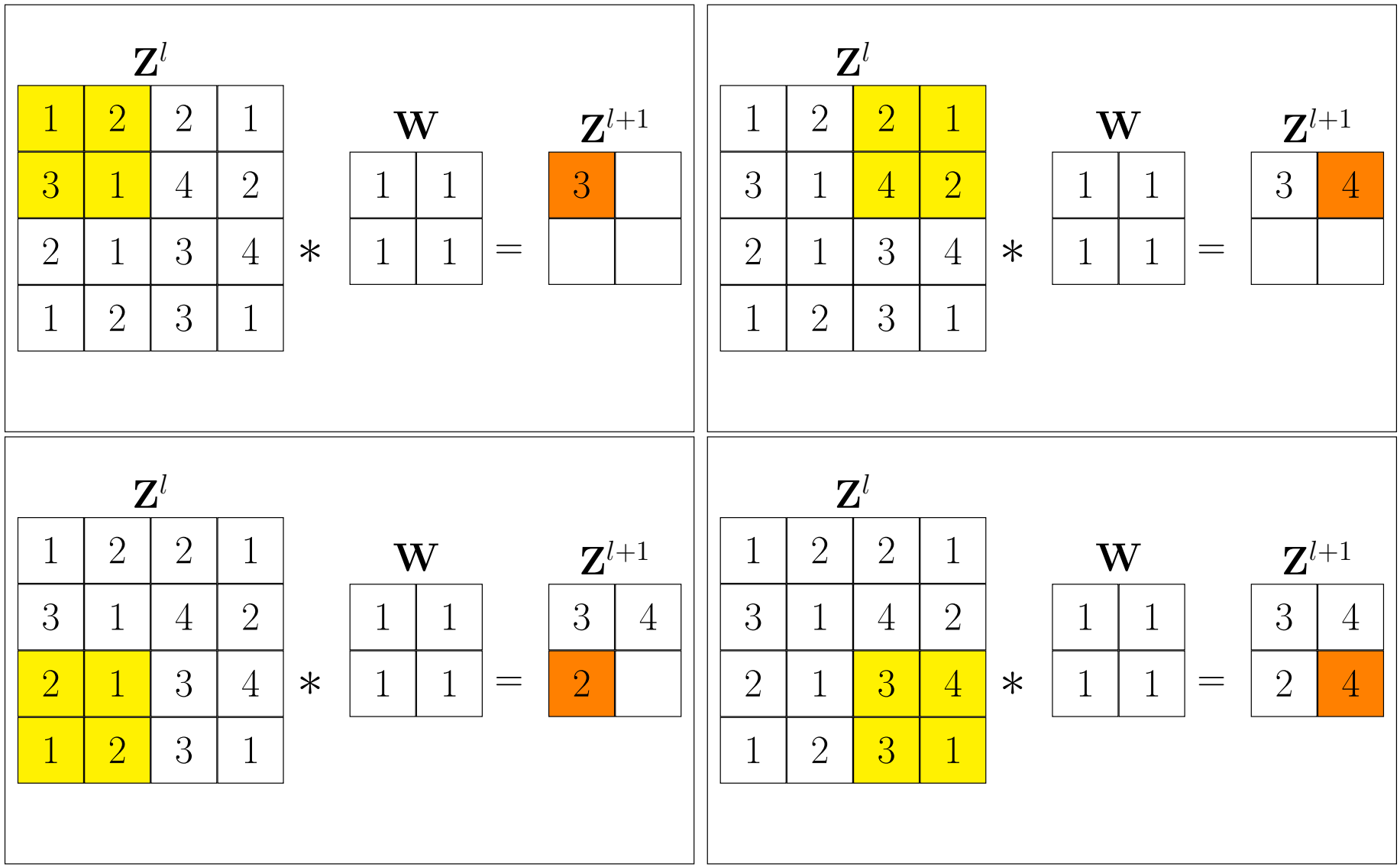
В примере показано применение Max-Pooling на матрице $\mathbf{Z}^l$ размерами 4×4 ($n_l = 4$) с использованием размера окна $k = 2$ и шага $s = 2$, равного размеру окна.

Таким образом, результирующий слой $\mathbf{Z}^{l+1}$ имеет размеры 2×2, поскольку $\left[\frac{n_l}{s}\right] = \left[\frac{4}{2}\right] = 2$. Фильтр $\mathbf{W}$ имеет фиксированные веса, равные 1.

Свертка первого окна $\mathbf{Z}^l$, а именно $\mathbf{Z}_2^l(1,1)$ с фильтром $\mathbf{W}$ задается как

$$\max\left(\mathbf{Z}_2^l(1,1) \odot \mathbf{W}\right) = \max\left(\begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\right) = \max\left(\begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}\right) = 3$$

Пример Max-Pooling с движением шагами ($s = 2$)



Глубокие сети CNN

В типичной архитектуре сети CNN чередуются слои свертки (с суммированием в качестве функции агрегации, а также обучаемыми весами фильтра и коэффициентом смещения) и слои пулинга (например, макс-пулинга с фиксированным фильтром из единиц).

Идея состоит в том, что в то время как слой свертки обучает фильтры для извлечения информативных признаков, слой пулинга применяет функцию агрегирования, такую как max (или avg), для извлечения наиболее важного значения нейрона (или среднего значения нейрона) в каждом скользящем окне для каждого из каналов.

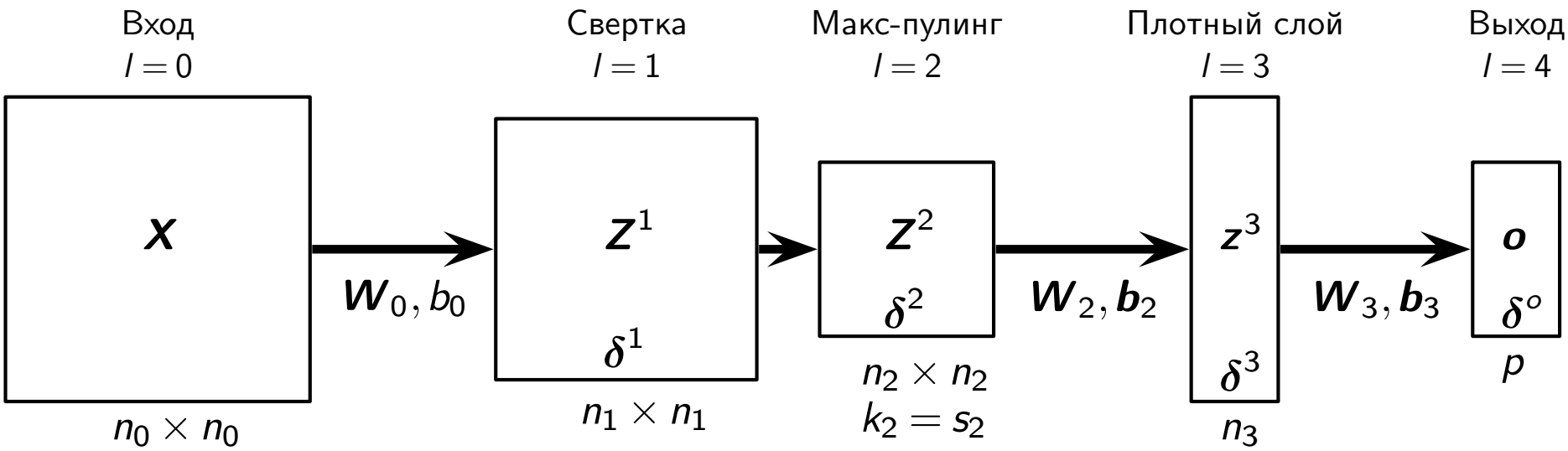
Начиная с входного слоя, глубокая сеть CNN состоит из нескольких, обычно чередующихся, слоев свертки и пулинга, за которыми следуют один или несколько полносвязных слоев, а затем последний выходной слой.

Для каждого слоя свертки и пулинга нужно выбрать размер окна k, а также значение шага s и использовать заполнение (padding) p или нет. Также нужно выбрать нелинейные функции активации для сверточных слоев, а также количество рассматриваемых слоев.

Обучение сети CNN

Чтобы увидеть, как обучать сеть CNN, рассмотрим сеть с одним слоем свертки и слоем макс-пулинга, за которым следует полносвязный слой.

Для простоты будем считать, что на входных данных $\mathbf{X}$ имеется только один канал, и далее используем только один фильтр.



Сеть CNN – фаза прямого прохода

Обозначим через $\mathbf{D} = \{\mathbf{X}_i, \mathbf{y}_i\}_{i=1}^{n}$ обучающие данные, включающие n тензоров $\mathbf{X}_i \in \mathbb{R}^{n_0 \times n_0 \times m_0}$ ($m_0 = 1$ для простоты) и через $\mathbf{y}_i \in \mathbb{R}^p$ соответствующий вектор отклика.

Для обучающей пары $(\mathbf{X}, \mathbf{y}) \in \mathbf{D}$ на этапе прямого прохода прогнозируемый результат $\mathbf{o}$ задается с помощью следующих уравнений:

$$\mathbf{Z}^1 = f^1\left((\mathbf{X} * \mathbf{W}_0) + b_0\right),$$

$$\mathbf{Z}^2 = \mathbf{Z}^1 *_{s_2,\,\max} \mathbf{1}_{k_2 \times k_2},$$

$$\mathbf{Z}^3 = f^3\left(\mathbf{W}_2^T \mathbf{Z}^2 + \mathbf{b}_2\right),$$

$$\mathbf{o} = f^o\left(\mathbf{W}_3^T \mathbf{Z}^3 + \mathbf{b}_3\right),$$

где $*_{s_2,\,\max}$ обозначает макс-пулинг с шагом s_2.

Пусть $\boldsymbol{\delta}^1$, $\boldsymbol{\delta}^2$ и $\boldsymbol{\delta}^3$ обозначают чистые векторы градиента в слоях $l = 1, 2, 3$ соответственно и пусть $\boldsymbol{\delta}^o$ обозначает чистый вектор градиента в выходном слое.

Выходной чистый вектор градиента может быть получен обычным способом путем вычисления частных производных функции потерь $(\partial \mathcal{E}_{\mathbf{X}})$ и функции активации $(\partial \mathbf{f}^o)$:

$$\boldsymbol{\delta}^o = \partial \mathbf{f}^o \odot \partial \mathcal{E}_{\mathbf{X}}$$

при условии, что выходные нейроны независимы.

Так как слой $l = 3$ полностью соединен с выходным слоем и, аналогично, слой макс-пулинга $l = 2$ полностью соединен с $\mathbf{Z}^3$, чистые векторы градиента на этих слоях вычисляются так же, как в обычной сети MLP:

$$\boldsymbol{\delta}^3 = \partial \mathbf{f}^3 \odot (\mathbf{W}_o \cdot \boldsymbol{\delta}^o),$$

$$\boldsymbol{\delta}^2 = \partial \mathbf{f}^2 \odot \left(\mathbf{W}_2 \cdot \boldsymbol{\delta}^3\right) = \mathbf{W}_2 \cdot \boldsymbol{\delta}^3$$

Последнее выражение вытекает из равенства $\partial \mathbf{f}^2 = 1$, так как макс-пулинг неявно использует тождественную функцию активации. Рассмотрим чистый градиент δ_{ij}^1 в нейроне z_{ij}^1 в слое $l = 1$ при $i, j = 1, ..., n_1$:

$$\delta_{ij}^1 = \begin{cases} \delta_{ij}^2 \cdot \partial f_{ij}^1 & \text{при } i = i^*,\ j = j^* \\ 0 & \text{в противном случае} \end{cases}$$

Другими словами, чистый градиент на нейроне z_{ij}^1 в слое свертки равен нулю, если этот нейрон не имеет максимального значения в своем окне.

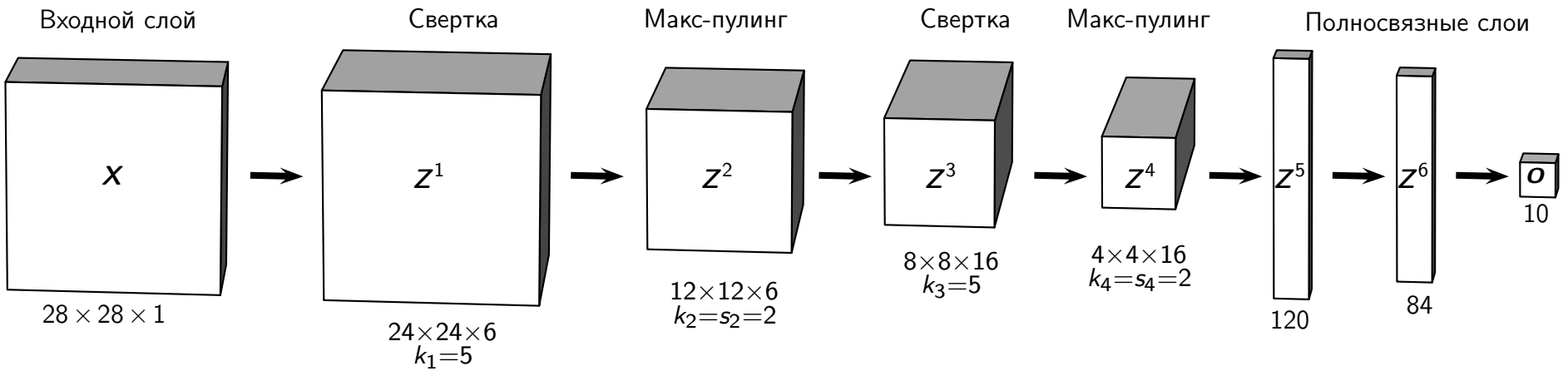
В противном случае, если значение является максимальным на этом нейроне, то чистый градиент распространяется обратно от слоя макс-пулинга к этому нейрону, а затем умножается на частную производную функции активации.

Матрица чистых градиентов $\boldsymbol{\delta}^1$ содержит чистые градиенты δ_{ij}^1 для всех $i, j = 1, ..., n_1$.

Пример сети CNN

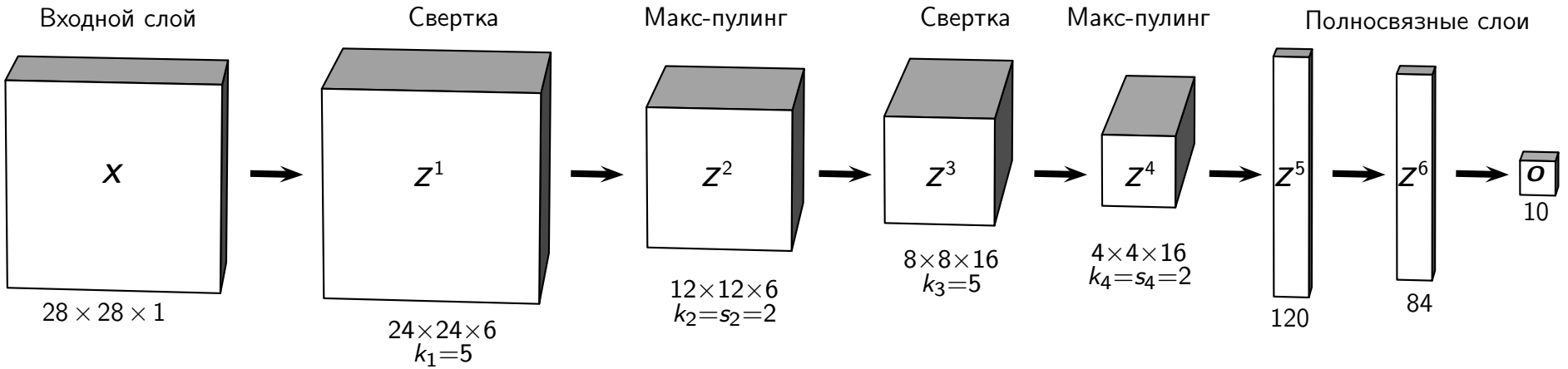
Пример показывает применение сети CNN для распознавания рукописных цифр. Эта CNN обучена и протестирована на наборе данных MNIST, который содержит 60 000 обучающих изображений и 10 000 тестовых изображений.

Каждое входное изображение представляет собой матрицу размерами 28×28 со значениями пикселей от 0 до 255, которые делятся на 255, так что каждый пиксель находится в интервале $[0, 1]$. Соответствующий (истинный) выход $\mathbf{y}_i$ представляет собой двоичный вектор с однократным кодированием, который обозначает цифру от 0 до 9; цифра 0 кодируется как $\mathbf{e}_1 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$, цифра 1 как $\mathbf{e}_2 = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0)^T$ и так далее.



Пример сети CNN

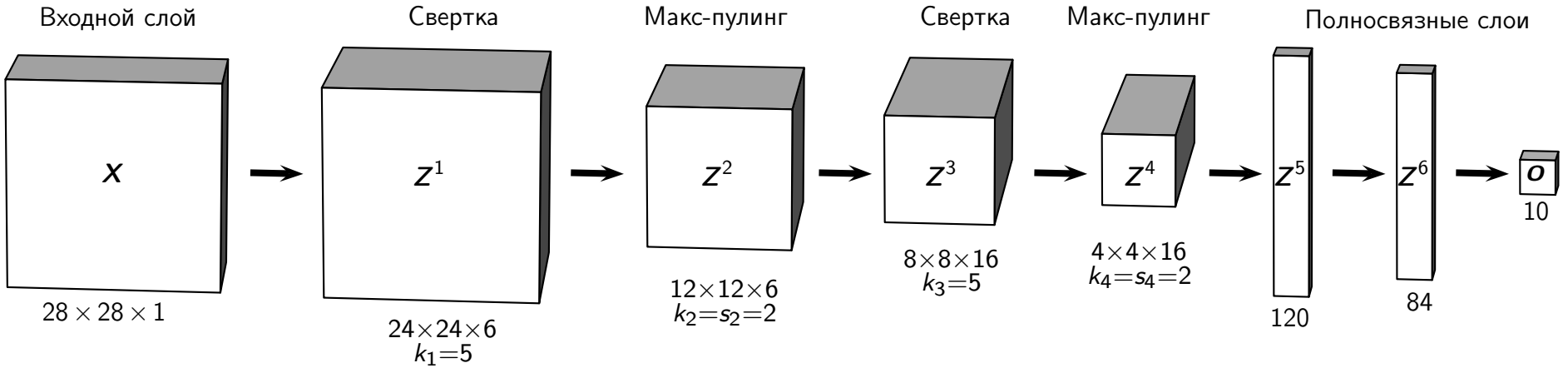
В представленной сети CNN все слои свертки используют шаг, равный единице, и не используют заполнение (padding), тогда как все слои макс-пулинг используют шаг, равный размеру окна. Поскольку каждый вход представляет собой изображение цифры размером 28×28 пикселей с одним каналом (оттенки серого), то $n_0 = 28$ и $m_0 = 1$, и, следовательно, вход $\mathbf{X} = \mathbf{Z}^0$ представляет собой тензор $n_0 \times n_0 \times m_0 = 28 \times 28 \times 1$. Первый слой свертки использует $m_1 = 6$ фильтров с $k_1 = 5$ и шагом $s_1 = 1$ без заполнения (padding).



Пример сети CNN

Каждый фильтр представляет собой тензор весов размерами $5 \times 5 \times 1$, и для шести фильтров результирующий тензор $\mathbf{Z}^1$ слоя $l = 1$ имеет размеры $24 \times 24 \times 6$, при этом $n_1 = n_0 - k_1 + 1 = 28 - 5 + 1 = 24$ и $m_1 = 6$. Второй скрытый слой представляет собой слой макс-пулинга ($k_2 = 2$ и $s_2 = 2$).

Поскольку макс-пулинг по умолчанию использует фиксированный фильтр $\mathbf{W} = \mathbf{1}_{k_2 \times k_2 \times 1}$, результирующий тензор $\mathbf{Z}^2$ имеет размеры $12 \times 12 \times 6$, при этом $n_2 = \left[\frac{n_1}{k_2}\right] = \left[\frac{24}{2}\right] = 12$ и $m_2 = 6$. Третий слой представляет собой сверточный слой с $m_3 = 16$ каналами с размером окна $k_3 = 5$ (и шагом $s_3 = 1$), что дает тензор $\mathbf{Z}^3$ размерами $8 \times 8 \times 16$, в котором $n_3 = n_2 - k_3 + 1 = 12 - 5 + 1 = 8$.

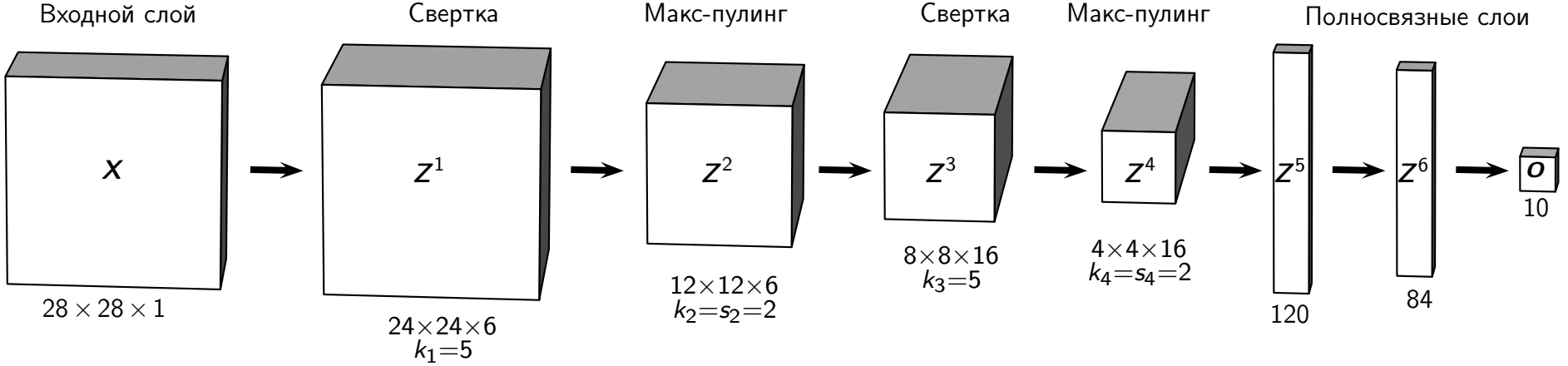


Пример сети CNN

Затем следует еще один слой макс-пулинга, который использует $k_4 = 2$ и $s_4 = 2$, что дает тензор $\mathbf{Z}^4$ с размерами $4 \times 4 \times 16$, где $n4 = \left[\frac{n_3}{k_4}\right] = \left[\frac{8}{2}\right] = 4$, а $m_4 = 16$.

Следующие три слоя полностью связаны, как в обычной сети MLP. Все 256 нейронов ($256 = 4 \times 4 \times 16$) в слое $l = 4$ связаны с нейронами тензора $\mathbf{Z}^5$ слоя $l = 5$, который представляет собой просто вектор, и его можно рассматривать как вырожденный тензор размерами $120 \times 1 \times 1$.

Слой $l = 5$ также полностью связан со слоем $l = 6$ с 84 нейронами. Выход $\mathbf{o}$ имеет 10 нейронов и использует функцию активации softmax. Нейроны в тензорах $\mathbf{Z}^1$, $\mathbf{Z}^3$, $\mathbf{Z}^5$ и $\mathbf{Z}^6$ используют активацию ReLU.



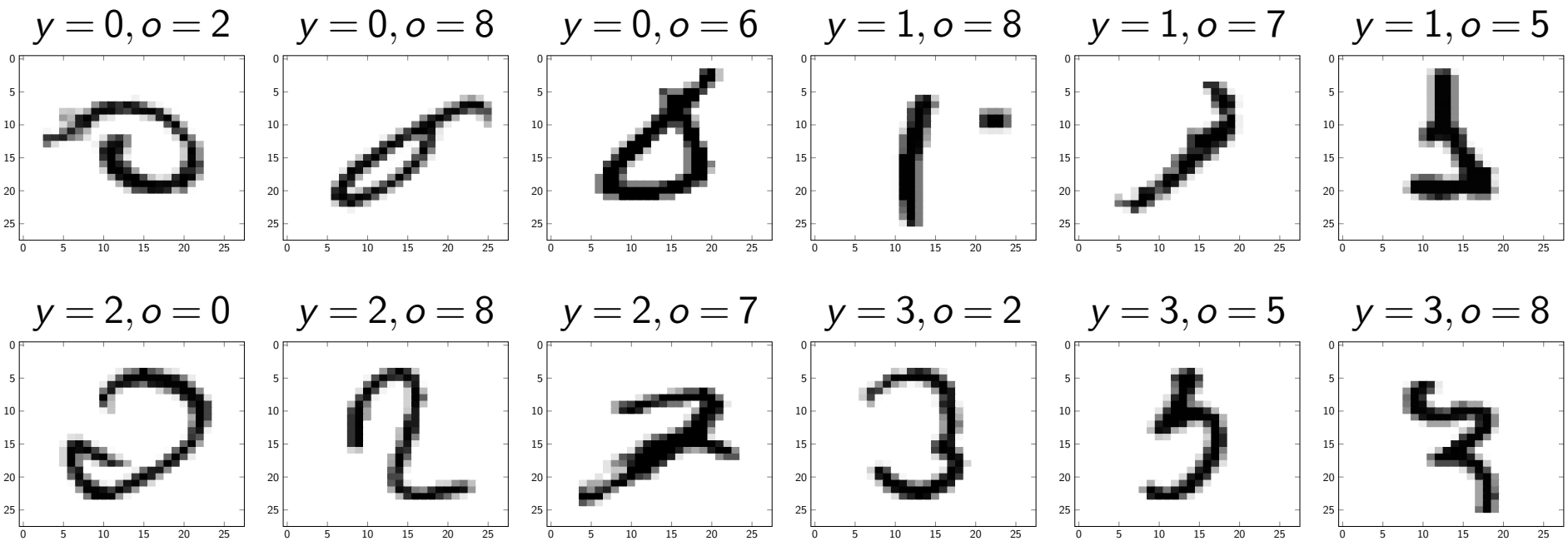
Пример сети CNN

Модель сети CNN обучается на $n = 60000$ изображениях из набора данных MNIST. Обучение проводится в течение 15 эпох с размером шага обучения $\eta = 0.2$ и функцией кросс-энтропийной ошибки (поскольку в данных встречается 10 классов).

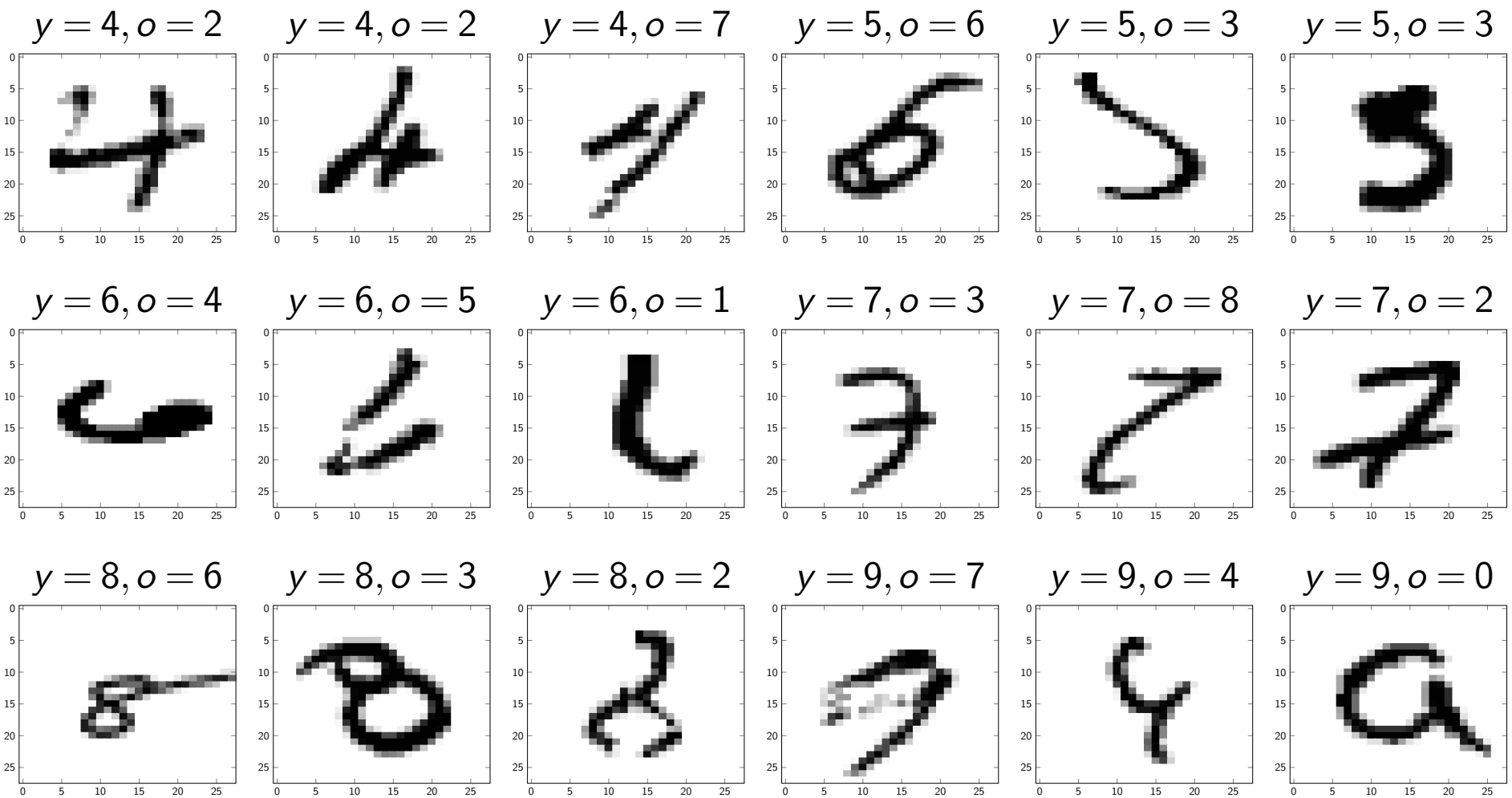
Обучение проводилось с использованием мини-пакетов с размером пакета 1000. После обучения модель CNN оценивается на тестовом наборе данных из 10 000 изображений. Модель CNN допускает 147 ошибок в тестовом наборе, что приводит к частоте ошибок 1.47%.

MNIST: некорректные предсказания

На рисунке показаны примеры изображений, которые были неправильно классифицированы сетью CNN. Видно, что некоторые из неправильно классифицированных изображений зашумлены, неполны или ошибочны, и их трудно правильно классифицировать даже человеку.



MNIST: некорректные предсказания



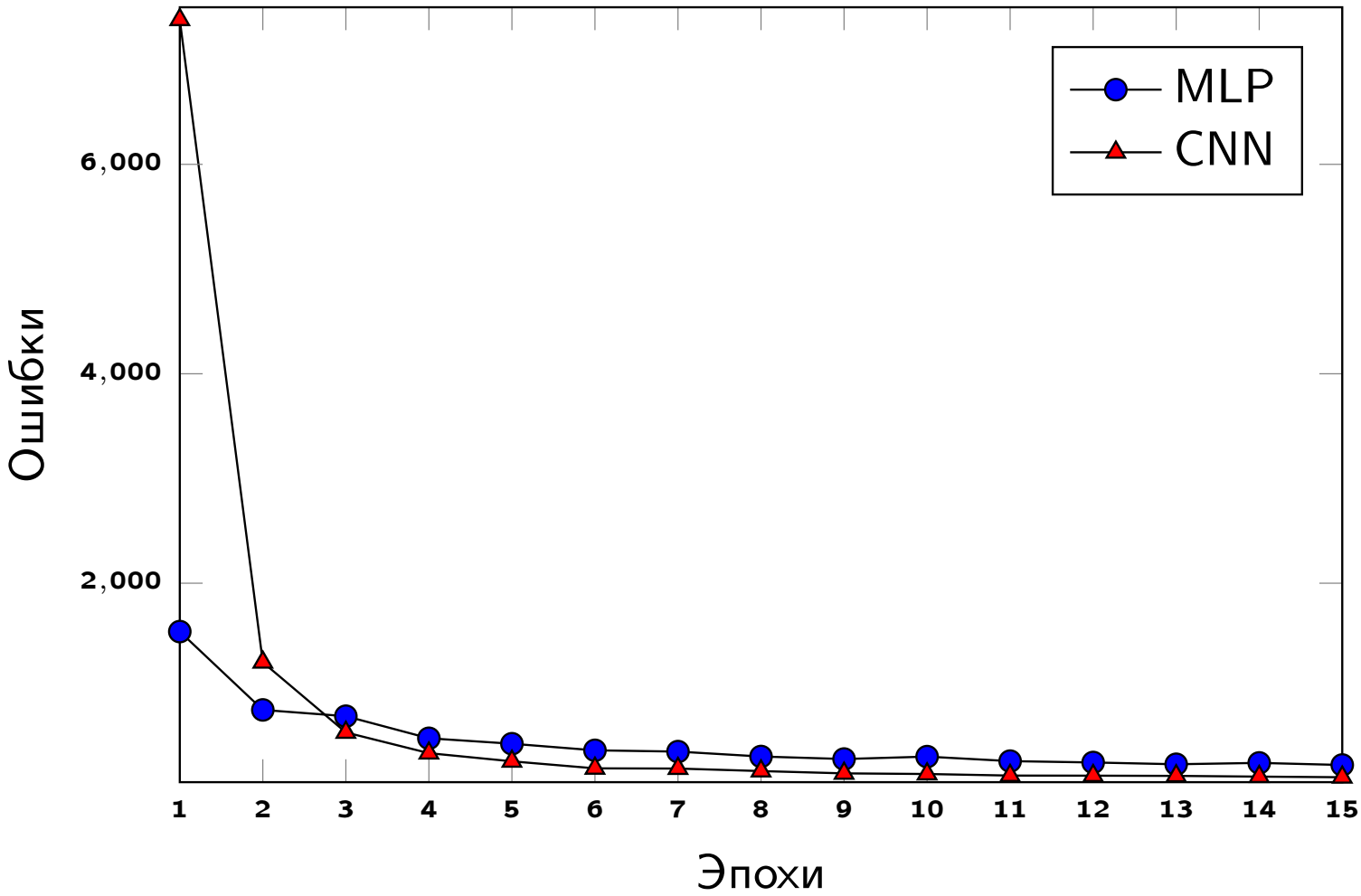
Пример сети CNN

Для сравнения также обучается глубокая сеть MLP с двумя полносвязными скрытыми слоями с теми же размерами, что и два полносвязных слоя перед выходным слоем в CNN, показанном на рисунке.

Следовательно, сеть MLP состоит из слоев $\mathbf{X}$, $\mathbf{Z}^5$, $\mathbf{Z}^6$ и $\mathbf{o}$, при этом входные изображения размерами 28×28 рассматриваются как вектор размера $d = 784$. Первый скрытый слой имеет размеры $n_1 = 120$, второй скрытый слой имеет размеры $n_2 = 84$, а выходной слой имеет размеры $p = 10$. Функция активации ReLU используется для всех слоев, кроме выходного, который использует функцию softmax. Мы обучаем модель MLP для 15 эпох на обучающем наборе данных с $n = 60000$ изображений, используя размер шага $\eta = 0.5$. В тестовом наборе данных сеть MLP допустила 264 ошибки, что соответствует частоте ошибок 2.64%.

MNIST: CNN или MLP

На графике показано количество ошибок в тестовом наборе после каждой эпохи обучения как для модели сети CNN, так и для модели сети MLP. Модель сети CNN обеспечивает значительно лучшую точность, чем MLP.



Регуляризация при обучении нейронных сетей

Регуляризация нейронной сети – это подход, при котором параметры (веса) нейронной сети ограничиваются, чтобы уменьшить переобучение, за счет уменьшения дисперсии при небольшом увеличении смещения.

В общем, для любой модели обучения M, если $L\left(\mathbf{y}, \hat{\mathbf{y}}\right))$ – некоторая функция потерь для заданных входных данных $\mathbf{x}$, а Θ обозначает все параметры модели, где $\hat{\mathbf{y}} = M\left(\mathbf{x} \mid \Theta\right)$, цель обучения состоит в том, чтобы найти параметры, минимизирующие потери по всем точкам (экземплярам) обучающего набора данных:

$$\min_{\Theta} J(\Theta) = \min_{\Theta} \sum_{i=1}^{n} L\left(\mathbf{y}_i, \hat{\mathbf{y}}_i\right) = \min_{\Theta} \sum_{i=1}^{n} L(\mathbf{y}_i, M(\mathbf{x}_i|\Theta))$$

При регуляризации к потерям на точках обучающего набора добавляется штраф за величину параметров (весов) Θ:

$$\min_{\Theta} J'(\Theta) = \min_{\Theta} \sum_{i=1}^{n} L(\mathbf{y}_i, M(\mathbf{x}_i|\Theta)) + \alpha R\left(\Theta\right),$$

где $\alpha \geqslant 0$ – регуляризационная константа.

Регуляризация L_2 для одного скрытого слоя

Рассмотрим сначала случай многослойного персептрона с одним скрытым слоем, а затем обобщим его. Параметры модели равны

$$\Theta = \{\mathbf{W}_h, \mathbf{b}_h, \mathbf{W}_o, \mathbf{b}_o\}$$

Целевая функция при регуляризации L_2 задается как

$$\min_{\Theta} J(\Theta) = \min_{\Theta} \left(\mathcal{E}_{\mathbf{x}} + \frac{\alpha}{2} R_{L_2}(\mathbf{W}_h, \mathbf{W}_o) \right) = \min_{\Theta} \left(\mathcal{E}_{\mathbf{x}} + \frac{\alpha}{2} \left(\|\mathbf{W}_h\|^2 + \|\mathbf{W}_o\|^2 \right) \right)$$

Регуляризованная целевая функция пытается минимизировать индивидуальные веса для пар нейронов. Это добавляет некоторое смещение, но снижает дисперсию, поскольку малые веса более устойчивы к изменениям входных данных с точки зрения прогнозируемых выходных значений. Правило обновления градиента задается как

$$\mathbf{W}_o \leftarrow \mathbf{W}_o - \eta \nabla_{\mathbf{W}_o} = \mathbf{W}_o - \eta \left(\mathbf{z}\, \boldsymbol{\delta}_o^T + \alpha\, \mathbf{W}_o \right) = (1 - \eta\, \alpha)\, \mathbf{W}_o - \eta\, \mathbf{z}\, \boldsymbol{\delta}_o^T$$

Регуляризация L_2 также называется уменьшением весов (weight decay), поскольку обновленная матрица весов использует уменьшенные веса из предыдущего шага с использованием коэффициента затухания $1 - \eta\, \alpha$.

Регуляризация L_2 в глубоких сетях MLP

Если дана функция потерь (ошибок) $\mathcal{E}_{\mathbf{x}}$, то целевая функция с регуляризацией L_2 равна

$$\min_{\Theta} J(\Theta) = \min_{\Theta} \left(\mathcal{E}_{\mathbf{x}} + \frac{\alpha}{2} R_{L_2}(\mathbf{W}_o, \mathbf{W}_1, ..., \mathbf{W}_h) \right) = \min_{\Theta} \left(\mathcal{E}_{\mathbf{x}} + \frac{\alpha}{2} \sum_{l=0}^{h} \|\mathbf{W}_l\|^2 \right),$$

где параметры модели равны $\Theta = \{\mathbf{W}_o, \mathbf{b}_o, \mathbf{W}_1, \mathbf{b}_1, ..., \mathbf{W}_h, \mathbf{b}_h\}$.

На основе вывода для одного скрытого слоя регуляризованный градиент равен:

$$\nabla_{\mathbf{W}_l} = \mathbf{z}^l \left(\boldsymbol{\delta}^{l+1} \right)^T + \alpha \, \mathbf{W}_l$$

и правило обновления для весовых матриц имеет вид

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \nabla_{\mathbf{W}_l} = (1 - \eta \, \alpha) \, \mathbf{W}_l - \eta \, \mathbf{z}^l \left(\boldsymbol{\delta}^{l+1} \right)^T$$

для $l = 0, 1, ..., h$, где $\boldsymbol{\delta}^l$ – чистый вектор градиента для скрытого слоя l.

Регуляризация дропаут (dropout)

Идея регуляризации dropout (отсев) состоит в том, чтобы случайным образом обнулить определенную часть значений нейронов в слое во время обучения. Цель состоит в том, чтобы сделать нейронную сеть более надежной и в то же время избежать переобучения. Отбрасывая случайным образом нейроны для каждой точки обучения, нейронная сеть вынуждена не полагаться на какой-либо конкретный набор связей между нейронами (ребер).

С точки зрения конкретного нейрона, поскольку он не может полагаться на наличие всех входящих связей, нейрон не концентрирует веса на определенных входных ребрах, а скорее распределяет веса среди входящих ребер.

Чистый эффект дропаута аналогичен регуляризации L_2, поскольку распределение весов приводит к уменьшению весов на входящих ребрах. Таким образом, полученная модель с дропаутом более устойчива к небольшим возмущениям входных данных, что может уменьшить переобучение за счет небольшого увеличения смещения.

Однако, в то время как регуляризация L_2 напрямую изменяет целевую функцию, регуляризация дропаут является формой структурной регуляризации, которая не меняет целевую функцию, а вместо этого изменяет топологию сети с точки зрения того, какие соединения между нейронами в настоящее время активны или неактивны.

Дропаут для сети MLP с одним скрытым слоем

На этапе обучения для каждой точки входных данных $\mathbf{x}$ создается случайный вектор маски, чтобы отбросить часть скрытых нейронов. Пусть $r \in [0, 1]$ – вероятность сохранения нейрона, тогда вероятность отсева (дропаута) равна $1 - r$.

Пусть скрытый слой состоит из m нейронов, тогда создадим m-мерный многомерный случайный вектор с распределением Бернулли $\mathbf{u} \in \{0, 1\}^m$ (вектор маски), каждый из элементов которого равен 0 с вероятностью дропаута $1 - r$ и 1 с вероятностью r. Пусть $\mathbf{u} = (u_1, u_2, ..., u_m))^T$, где

$$u_i = \begin{cases} 0 & \text{с вероятностью } 1 - r \\ 1 & \text{с вероятностью } r \end{cases}$$

Тогда фаза прямого прохода задается формулами:

$$\mathbf{z} = f^h\left(\mathbf{b}_h + \mathbf{W}_h^T \mathbf{x}\right),$$

$$\tilde{\mathbf{z}} = \mathbf{u} \odot \mathbf{z},$$
$$\mathbf{o} = f^o\left(\mathbf{b}_o + \mathbf{W}_o^T \tilde{\mathbf{z}}\right),$$

где знак $\odot$ означает поэлементное умножение.

Инвертированный дропаут

В приведенном выше базовом подходе дропаута есть одна сложность, а именно, ожидаемый выход нейронов скрытого слоя во время обучения и тестирования отличается, поскольку дропаут не применяется на этапе тестирования (например, не нужно, чтобы прогнозы менялись случайным образом на заданном тестовом входе). При вероятности сохранения скрытого нейрона, равной r, его ожидаемое выходное значение во время обучения равно

$$\mathbb{E}\left[z_i\right] = r\, z_i + (1 - r)\, 0 = r\, z_i.$$

Поскольку во время тестирования дропаут не применяется, выходные данные скрытых нейронов во время тестирования будут выше, чем во время обучения. Один из подходов состоит в том, чтобы масштабировать значения скрытых нейронов по r во время тестирования. С другой стороны, существует более простой подход, называемый инвертированным дропаутом, который не требует изменений во время тестирования. Идея состоит в том, чтобы изменить масштаб скрытых нейронов после шага дропаута на этапе обучения следующим образом:

$$\mathbf{z} = f^h\left(\mathbf{b}_h + \mathbf{W}_h^T\mathbf{x}\right),\ \tilde{\mathbf{z}} = \frac{1}{r}\mathbf{u} \odot \mathbf{z},\ \mathbf{o} = f^o\left(\mathbf{b}_o + \mathbf{W}_o^T\tilde{\mathbf{z}}\right)$$

Дропаут в глубоких сетях MLP

Аналогичным образом выполняется регуляризация дропаутом для глубоких сетей MLP. Пусть $r_l \in [0, 1]$ для $l = 1, 2, ..., h$ обозначает вероятность сохранения скрытого нейрона для слоя l, так что $1 - r_l$ – вероятность дропаута. Можно также использовать единую вероятность r для всех слоев, установив $r_l = r$. Определим вектор маски $\mathbf{u}^l \in \{0, 1\}^{n_l}$ для скрытого слоя l следующим образом:

$$u_i^l = \begin{cases} 0 & \text{с вероятностью } 1 - r_l \\ 1 & \text{с вероятностью } r_l \end{cases}$$

Тогда шаг прямого прохода между слоями l и $l + 1$ задается так:

$$\mathbf{z}^l = f^h \left(\mathbf{b}_l + \mathbf{W}_l^T \tilde{\mathbf{z}} \right),$$

$$\tilde{\mathbf{z}} = \frac{1}{r} \mathbf{u}^l \odot \mathbf{z}^l,$$

используя инвертированный дропаут.