Сверточные нейронные сети. Основные элементы. Векторизация. Транспонированная свертка

Роман Малашин

2 июня 2025 г.

Содержание

1 Введение

Сверточные слои нашли широкое применение в задачах компьютерного зрения, таких как распознавание изображений, сегментация и обнаружение объектов. Их способность эффективно обрабатывать большие объемы данных с высокой степенью вариативности делает их важным компонентном современных нейронных сетей. Сети основанные на сверточных слоях называют сверточными нейронными сетями (СНС).

Можно выделить следующие особенности сверточного слоя:

- Локальное восприятие. Каждый нейрон в сверточном слое реагирует только на ограниченную область входных данных, что позволяет выделять локальные признаки. Эта область называется рецептивным полем.
- Использование общих параметров. В отличие от полносвязных слоев, где каждый вес уникален для каждой связи, в сверточных слоях один и тот же фильтр (набор весов) применяется ко всему входному изображению или карте признаков. Это значительно снижает количество параметров и увеличивает эффективность обучения.
- Эквивариантность к сдвигу. Сверточные слои обладают свойством эквивариантности к сдвигу, что означает, что карты признаков будут претерпевать то же геометрическое преобразование, что и изображение (например, если изображение сдвинется вправо, то и карта признаков сверточного слоя сдвинется вправо).

2 Свертка

Свертка представляет собой математическую операцию, при которой две функции комбинируются для получения третьей функции. В контексте СНС, свертка используется для применения фильтров (или ядер) к входным данным для выделения определенных признаков. Формально, операцию свертки можно выразить как:

$$\mathbf{Z^{(l+1)}} = \mathbf{Y^{(l)}} * \mathbf{F}, \tag{1}$$

где $\mathbf{Y^{(l)}}$ обозначает входные данные или карту признаков на слое l, $\mathbf{F}$ — ядро свертки или фильтр, а $\mathbf{Z^{(l+1)}}$ — результат применения свертки, то есть выходные данные или новая карта признаков на слое $l+1$ до применения функции активации, символ $*$ обозначает непосредственно операцию свертки. В сверточных сетях операция свертки обретает особенности.

3 Сверточный слой

Пусть размер входного изображения или карты признаков $\mathbf{Y^{(l)}}$: $H^{(l)} \times W^{(l)} \times D^{(l)}$, где:

- $H^{(l)}$: высота;
- $W^{(l)}$: ширина;
- $D^{(l)}$: глубина (количество каналов).

Размер фильтра (ядра): $F_h^{(l)} \times F_w^{(l)} \times D^{(l)}$, где:

- $F_h^{(l)}$: высота фильтра;
- $F_w^{(l)}$: ширина фильтра;

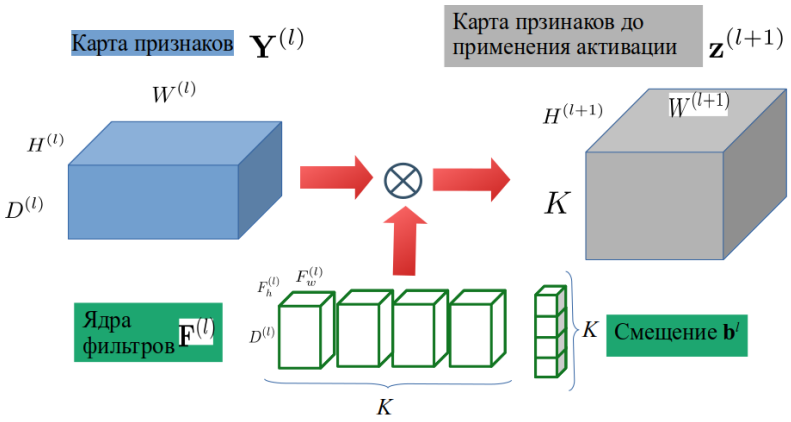

Рис. 1: Иллюстрация к процедуре свертки

Глубина $D^{(l)}$ совпадает с глубиной входных данных.

На рисунке 1 наглядно показаны размеры тензоров, используемых в сверточном слое.

Элемент выходной карты признаков $z_{rc}^{(l+1)} \in \mathbf{Z^{(l+1)}}$ может быть рассчитан по формуле:

$$z_{rc}^{(l+1)} = \sum_{i=1}^{F_h^{(l)}} \sum_{j=1}^{F_w^{(l)}} \sum_{d=1}^{D^{(l)}} \left(\mathbf{Y^{(l)}}_{(r+i-1)(c+j-1)d} \times \mathbf{F}_{ijd} \right) + b_d, \tag{2}$$

где:

- $z_{rcd}^{(l+1)}$ — элемент выходной карты признаков на позиции (r, c) для канала d;
- $\mathbf{Y^{(l)}}_{(r+i-1)(c+j-1)d}$ — элемент входной карты признаков или изображения на позиции $(r+i-1, c+j-1)$ для канала d. Требуется вычитание 1, т.к. индексирование начинается с единицы, а крайние элементы фильтра должны быть наложены на карту признаков без смещения относительно (r, c) (если следовать стандартному в программировании индексированию с 0, то такой корректировки не требуется);
- $\mathbf{F}_{ijd}$ — элемент ядра (фильтра) свертки на позиции (i, j) для канала d;
- b_d — смещение (bias) для канала d, применяемое ко всем элементам выходной карты признаков для этого канала;
- r и c указывают на конкретную позицию в выходной карте признаков, по которой происходит расчет.

При использовании $D^{(l+1)}$ фильтров формируется трехмерный тензор - карта признаков с $D^{(l+1)}$ каналами.

После вычисления выходной карты признаков применяется активационная функция[1], например, ReLU или его варианты (Leaky ReLU, ELU).

[1]Также может применяться нормализация. Например, Batch Normalization, помогает уменьшить внутреннее ковариационное смещение и улучшить обучение, делая распределение каждого слоя более стабильным.

Заметим, что размеры выходной карты признаков H^{l+1} и W^{l+1} меньше, чем размеры H^l, W^l входной карты для фильтров с размерами $F_h^{(l)}, F_w^{(l)} > 1$, т.к. требуется, чтобы ядро фильтра полностью располагалось на исходной карте признаков, а значит:

$$H^{(l+1)} = H^l - F_h^l + 1,$$
$$W^{(l+1)} = W^l - F_w^l + 1.$$

$$valid, .$$

3.1 Алгоритм обратного распространения ошибки при общих весах

Для вычисления свертки необходимо, чтобы несмотря на различные значения градиентов в разных рецептивных полях, соответствующие веса фильтров были одинаковыми. Это достигается с использованием алгоритма 1 достаточно простым способом:

- необходимые веса инициализируется одинаково;
- градиент считается с помощью алгоритма обратного распространения ошибки для разных весов независимо.
- для обновления значений весов градиенты посчитанные в разных рецептивных полях усредняются.

Algorithm 1 Алгоритм обратного распространения ошибки с общими весами

1: Вход: индексы весов i, j, которые должны быть одинаковыми
2: Инициализировать $w_i = w_j$
3: Вычислить производные $\frac{\partial \mathcal{L}}{\partial w_i}$ и $\frac{\partial \mathcal{L}}{\partial w_j}$ с помощью алгортима обратного распространеия ошибки
4: Для обновления весов использовать $\Delta w_i = \Delta w_j = \frac{\partial \mathcal{L}}{\partial w_i} + \frac{\partial \mathcal{L}}{\partial w_j}$

3.2 Паддинг (padding)

Паддинг (padding) - расширение карты признаков, фиктивные элементы добавляются вокруг входных данных, позволяя фильтру "вписываться"в данные на краях и контролировать размер выходной карты признаков. Часто расширение карты признаков осуществляется с помощью нулей, тогда говорят о zero-padding. Существуют три основных вида паддинга:

- valid - при этом способе не используется дополнительный паддинг, и фильтр применяется только к тем областям входных данных, которые полностью перекрываются фильтром. Это обычно приводит к уменьшению размера выходной карты признаков.

- full - при этом способе добавляется достаточно паддинга, чтобы фильтр мог быть применен ко всем возможным позициям вокруг входных данных, включая края и углы. Это приводит к увеличению размера выходной карты признаков по сравнению с входными данными.
- same - этот способ подразумевает добавление такого количества паддинга, чтобы размер выходной карты признаков был равен размеру входной карты признаков. Фильтр может выходить за края входных данных, но добавленный паддинг компенсирует это, позволяя сохранить размерность данных на выходе.

3.3 Шаг (Stride)

Шаг (Stride) определяет, на сколько пикселей фильтр перемещается после каждой операции. Увеличение шага приводит к уменьшению размера выходной карты признаков. Можно представить себе использование шага stride=2, как результат отбрасывания всех четных элементов результирующего тензора $\mathbf{Z}^{(l+1)}$ по измерению соответствующему ширине и высоте исходных данных. Таким образом, при использовании stride=2 итоговый тензор будет иметь размер $H^{(l+1)}/2 \times W^{(l+1)}/2 \times D^{(l+1)}$.

3.4 Подвыборка (Pooling)

Слой подвыборки, следующий за сверточным слоем и слоем активации, помогает уменьшить размерность выходных данных, сохраняя при этом важные признаки. Максимальный (maxpooling) и средний пулинг (average pooling) — два основных типа подвыборки, используемых для этой цели.

4 Векторизация свертки

В сверточных нейронных сетях (СНС), векторизация вычислений играет ключевую роль, обеспечивая эффективность процесса обучения с использованием современных центральных и графических процессоров.

4.1 Векторизация с помощью операции Im2Col

На самом деле оказывается, что можно свести выполнение свертки к перемножению матриц весов и матриц активации. После выполнения такого преобразования можно использовать векторную форму для выполнения прямого и обратного прохода так, как это представлено в материале, посвященному обратному распространению ошибки.

Рассмотрим подробнее, как происходит преобразования изображения и весов в вид пригодный для выполнения матричного перемножения.

Пусть размер входного изображения или карты признаков: $H^{(l)} \times W^{(l)} \times D^{(l)}$, где:

- $H^{(l)}$: высота;
- $W^{(l)}$: ширина;
- $D^{(l)}$: глубина (количество каналов).

Пусть размер фильтра (ядра): $F_h^{(l)} \times F_w^{(l)} \times D^{(l)}$, где:

- $F_h^{(l)}$: высота фильтра;
- $F_w^{(l)}$: ширина фильтра;

Глубина фильтра $D^{(l)}$ совпадает с глубиной входных данных. stride$^{(l)}$ - размер шага, с которым фильтр перемещается по входным данным. Отступ pad$^{(l)}$ определяет насколько расширяется карта признаков, что требуется для контроля размера выходной карты признаков.

Для преобразования изображения используется так называемая операция Im2Col. Шаги операции Im2Col:

1. Извлечение фрагментов изображения/карты признаков. Для каждой позиции фильтра на входных данных (учитывая шаг и отступ), извлекается фрагмент размером $F_h^{(l)} \times F_w^{(l)} \times D^{(l)}$. Эти 3D фрагменты преобразуются в 1D столбцы.
2. Стекинг столбцов: Горизонтально складываются эти столбцы, формируя 2D матрицу. Количество столбцов равно количеству позиций фильтра на входных данных, а количество строк — $F_h^{(l)} \cdot F_w^{(l)} \cdot D^{(l)}$, размер развернутого фильтра.

В таблице 1 рассмотрен пример с одноканальной картой признаков $\mathbf{Y}^{(l)}$ размером 4×4 со значениями от 1 до 16.

Таблица 1: Операция Im2Col. Пример. stride = 1, pad=0

<table>
<tr><th>Шаг</th><th>Визуальное представление</th></tr>
<tr><td>Входная карта признаков $\boldsymbol{Y}^{(l)}$</td><td>
<table>
<tr><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>15</td><td>16</td></tr>
</table>
</td></tr>
<tr><td>Фильтр $\boldsymbol{F}$</td><td>
<table>
<tr><td>F_{11}</td><td>F_{12}</td></tr>
<tr><td>F_{21}</td><td>F_{22}</td></tr>
</table>
</td></tr>
<tr><td>Результат $\hat{\boldsymbol{Y}}^{(l)}$</td><td>
<table>
<tr><td>1</td><td>2</td><td>3</td><td>5</td><td>6</td><td>7</td><td>9</td><td>10</td><td>11</td></tr>
<tr><td>2</td><td>3</td><td>4</td><td>6</td><td>7</td><td>8</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>5</td><td>6</td><td>7</td><td>9</td><td>10</td><td>11</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>6</td><td>7</td><td>8</td><td>10</td><td>11</td><td>12</td><td>14</td><td>15</td><td>16</td></tr>
</table>
</td></tr>
</table>

Рассмотрим фильтр $\mathbf{F}$ размером 2×2:

$$\mathbf{F} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}. \tag{4}$$

Для выполнения преобразования Im2Col фильтр $\mathbf{F}$ перемещается по карте признаков с использованием шага stride (в примере в таблице 1 stride = 1, padding = 0). В каждой позиции фрагмент 2×2, который покрывает фильтр, преобразовывается в столбец новой матрицы $\hat{\mathbf{Y}}^{(l)}$.

После преобразования карты признаков фильтр может быть сам вытянут в вектор:

$$\mathbf{f} = \begin{bmatrix} F_{11} \\ F_{12} \\ F_{21} \\ F_{22} \end{bmatrix} \tag{5}$$

и перемножен с матрицей полученной в результате преобразования Im2Col:

$$\hat{\mathbf{z}}^{(l+1)} = \hat{\mathbf{Y}}^{(l)T}\mathbf{f} + \mathbf{b}_{\mathbf{F}}, \tag{6}$$

где символом T обозначается транспонирование матрицы, $\mathbf{b}_{\mathbf{F}} = [b, b, \cdots]^T$ - это вектор размером $1 \times C$, полученный путем копирования скалярного смещения b фильтра $\mathbf{F}$ N раз, где C - количество столбцов в матрице $\hat{\mathbf{Y}}^{(l)}$. В результате получается вектор $\hat{\boldsymbol{y}}^{(l+1)}$ размером $1 \times C$, который может быть преобразован обратно в карту признаков $\mathbf{Z}^{(l+1)}$, с помощью процедуры Col2Im, обратной Im2Col. В результате получается новая карта признаков, соответствующая выходу сверточного слоя (до применения активационной функции). Таким образом, вычисление свертки сведено к выражению перемножению матрицы и вектора.

4.2 Пакетная векторизация

Как и в случае полносвязных слоев, векторизация сразу на mini-batch пакете данных позволяет дополнительно оптимизировать вычисления. Кроме того, можно проводить вычисления сразу с несколькими фильтрами[2]. Рассмотрим случай, когда имеется M образцов в мини-пакете и K различных фильтров в слое. В этом случае, входные данные и фильтры трансформируются в соответствии с принципами операции Im2Col для каждого образца и каждого фильтра соответственно, а затем производится расчет свертки для всего мини-пакета одновременно.

Для этого формируется матрица активаций $\mathcal{Y}^{(l)}$ для всего мини-пакета. Если рассматривать каждую матрицу, полученную с помощью Im2Col для отдельного образца, как блок, то матрица активаций для мини-пакета формируется путем горизонтальной[3] конкатенации этих блоков. Таким образом, каждый блок представляет собой данные одного образца из мини-пакета, преобразованные в формат, пригодный для матричного умножения:

$$\mathcal{Y}^{(l)} = \begin{bmatrix} \hat{\mathbf{Y}}_1^{(l)} & \cdots & \hat{\mathbf{Y}}_M^{(l)} \end{bmatrix}. \tag{7}$$

Размер матрицы активаций по одному измерению соответствует количеству элементов в фильтре а по другому соответствует количеству возможных положений фильтра на карте признаков, умноженному на количество примеров в пакете: $F_h^{(l)} F_w^{(l)} D^{(l)} \times MC^{(l+1)}$.

[2]Что даже более важно, чем объединение пакетов данных в большую матрицу, т.к. распараллеливание умножения матрицы на множества матриц может быть эффективно выполнено с помощью современных пакетов линейной алгебры.

[3]Или вертикальной, в зависимости от реализации

Аналогично, для всех фильтров создается матрица весов $\mathcal{F}^{(l)}$, где каждый столбец представляет собой векторизованную форму одного фильтра:

$$\mathcal{F}^{(l)} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_K \\ | & | & \cdots & | \end{bmatrix}, \tag{8}$$

где $\mathbf{f}_i$ обозначает векторизованную форму i -го фильтра, преобразованного в столбец. Размер этой матрицы $F_h^{(l)} F_w^{(l)} D^{(l)} \times K$. Затем производится матричное умножение матрицы активаций и транспонированной матрицы весов:

$$\hat{\mathcal{Z}}^{(l+1)} = \mathcal{Y}^{(l)T} \mathcal{F}^{(l)} + \mathbf{B}^{(l)}, \tag{9}$$

где T обозначает транспонирование, $\mathbf{B}^{(l)}$ — матрица смещений, состоящая из копий вектора смещений для каждого фильтра, повторенная в соответствии с количеством образцов в мини-пакете.

Размер матрицы $\hat{\mathcal{Y}}^{(l+1)}$ равен $K \times C^{(l+1)}$, а после применения процедуры Col2Im к каждому столбцу можно получить K двумерных карт признаков в привычном для интерпретации виде с восстановленной пространственной структурой данных.

5 Транспонированная свертка

Транспонированная свертка (transposed convoludion, upconvolution, deconvolution) - это обратная операция свертки. Она позволяет увеличить размерность входных данных. Транспонированная свертка обычно применяется, например, в задачах сегментации изображений (Unet, Segnet), где нужно перейти от сжатого пространства признаков к изображению исходного размера, или в генеративно-состязательных сетях для генерации изображений высокого разрешения.

5.1 Свертка с использованием матрицы Тёплица (1D)

Для простоты рассмотрим одномерный случай использования свертки. Пусть имеется входной вектор $\mathbf{y}^{(l)}$ размером $N \times 1$, а также ядро свертки $\mathbf{f}$ размером $K \times 1$. Тогда результат свертки будет вектор $\mathbf{z^{(l+1)}}$ размером $(N - K + 1) \times 1$, если padding=valid (расчет только в местах, где фильтр полность попадает на карту признаков), или размером $N \times 1$ при padding=same.

Мы будем пренебрегать смещением (bias=0) в формулах ниже для простоты записи.

Одномерная свертка определяется операцией $\mathbf{z^{(l+1)}} = \mathbf{y^{(l)}} * \mathbf{f}$, где символ $*$ обозначает операцию свертки. Для получения результата свертки, ядро $\mathbf{f}$ последовательно применяется к каждому сегменту вектора $\mathbf{y}^{(l)}$, сдвигаясь на один элемент за раз. На каждом шаге вычисляется скалярное произведение между ядром и текущим сегментом вектора.

Выше был рассмотрен эффективный способ представления свертки в виде перемножения матриц, однако можно представить свертку и по-другому: ядро представляется в виде матрицы, где в области диагонали располагаются одинаковые элементы (матрица Тёплица). Например, если исходный сигнал $\mathbf{y}^{(l)} = [y_1^{(l)}, y_2^{(l)}, y_3^{(l)}, y_4^{(l)}, y_5^{(l)}]^T$ и ядро свертки $\mathbf{f} = [f_1, f_2, f_3]^T$, то процесс

свертки можно представить в виде умножения исходного вектора на матрицу Тёплица, сформированную из элементов ядра $\mathbf{f}$ (исходя из того, что фильтр не выходит за границу входа, zero padding=0):

$$\mathrm{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}, \quad \mathrm{T} = \begin{bmatrix} f_1 & f_2 & f_3 & 0 & 0 \\ 0 & f_1 & f_2 & f_3 & 0 \\ 0 & 0 & f_1 & f_2 & f_3 \end{bmatrix} \quad \mathrm{y}^{(l)} = \begin{bmatrix} y_1^{(l)} \\ y_2^{(l)} \\ y_3^{(l)} \\ y_4^{(l)} \\ y_5^{(l)} \end{bmatrix} \tag{10}$$

Тогда получение результата свертки можно представить как умножение матриц:

$$\mathbf{z}^{(l+1)} = \mathbf{T}\mathbf{y}^{(l)} = \begin{bmatrix} f_1 y_1^{(l)} + f_2 y_2^{(l)} + f_3 y_3^{(l)} \\ f_1 y_2^{(l)} + f_2 y_3^{(l)} + f_3 y_4^{(l)} \\ f_1 y_3^{(l)} + f_2 y_4^{(l)} + f_3 y_5^{(l)} \end{bmatrix} \tag{11}$$

В этом случае размерность выходного вектора сократилась и составляет 3.

В случае использования zero-padding=same размерность выходного вектора сохраняется:

$$\mathbf{T}_{padded} = \begin{bmatrix} f_1 & f_2 & f_3 & 0 & 0 & 0 & 0 \\ 0 & f_1 & f_2 & f_3 & 0 & 0 & 0 \\ 0 & 0 & f_1 & f_2 & f_3 & 0 & 0 \\ 0 & 0 & 0 & f_1 & f_2 & f_3 & 0 \\ 0 & 0 & 0 & 0 & f_1 & f_2 & f_3 \end{bmatrix}, \quad \mathbf{y}_{padded}^{(l)} = \begin{bmatrix} 0 \\ y_1^{(l)} \\ y_2^{(l)} \\ y_3^{(l)} \\ y_4^{(l)} \\ y_5^{(l)} \\ 0 \end{bmatrix} \tag{12}$$

$$\mathbf{z}^{(l+1)} = \mathbf{T}_{padded}\mathbf{y}_{padded}^{(l)} = \begin{bmatrix} f_2 y_1^{(l)} + f_3 y_2^{(l)} \\ f_1 y_1^{(l)} + f_2 y_2^{(l)} + f_3 y_3^{(l)} \\ f_1 y_2^{(l)} + f_2 y_3^{(l)} + f_3 y_4^{(l)} \\ f_1 y_3^{(l)} + f_2 y_4^{(l)} + f_3 y_5^{(l)} \\ f_1 y_4^{(l)} + f_2 y_5^{(l)} \end{bmatrix} \tag{13}$$

Теперь рассмотрим случай, когда stride (шаг свертки) равен 2. Это означает, что мы перемещаем фильтра сразу на элемента входного вектора после каждого применения фильтра, что и приводит к уменьшению размерности выходного вектора вдвое по сравнению с исходным вектором.

В этом случае матрица Тёплица и результат выглядят следующим образом:

$$\mathbf{T}_{strided} = \begin{bmatrix} f_1 & f_2 & f_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & f_1 & f_2 & f_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_1 & f_2 & f_3 \end{bmatrix}, \quad \mathbf{y}_{strided}^{(l)} = \begin{bmatrix} 0 \\ y_1^{(l)} \\ y_2^{(l)} \\ y_3^{(l)} \\ y_4^{(l)} \\ y_5^{(l)} \\ 0 \end{bmatrix} \tag{14}$$

$$\mathbf{z}^{(l+1)} = \mathbf{T}_{strided}\mathbf{y}^{(l)}_{strided} = \begin{bmatrix} f_2 y_1^{(l)} + f_3 y_2^{(l)} \\ f_1 y_2^{(l)} + f_2 y_3^{(l)} + f_3 y_4^{(l)} \\ f_1 y_4^{(l)} + f_2 y_5^{(l)} \end{bmatrix}. \tag{15}$$

5.2 Транспонированная свертка (1D)

Транспонированная свертка работает в обратном направлении. Пред положим, мы хотим получить вектор $\mathbf{y}^{(l)}$ из вектора $\mathbf{z}^{(l+1)}$, имея ту же матрицу свертки $\mathbf{f}$. В этом случае используется транспонированная матрица Тёплица $\mathbf{T}^T$, соответствующая ядру $\mathbf{f}$[4]. Это позволяет "распаковать"данные обратно к исходной размерности или даже увеличить ее (если stride > 1). Транспонированную свертку иногда обозначают с помощью символа $*^T$:

$$\mathbf{y}^{(l)} = \mathbf{f} *^T \mathbf{z}^{(l+1)} = \mathbf{T}^T * \mathbf{z}^{(l+1)} \tag{16}$$

Для случая stride=1, padding=full (все возможные положения перекрытия фильтра с картой признаков):

$$\mathbf{y}^{(l)} = \begin{bmatrix} f_1 & 0 & 0 & 0 & 0 \\ f_2 & f_1 & 0 & 0 & 0 \\ f_3 & f_2 & f_1 & 0 & 0 \\ 0 & f_3 & f_2 & f_1 & 0 \\ 0 & 0 & f_3 & f_2 & f_1 \\ 0 & 0 & 0 & f_3 & f_2 \\ 0 & 0 & 0 & 0 & f_3 \end{bmatrix} \begin{bmatrix} z_1^{(l+1)} \\ z_2^{(l+1)} \\ z_3^{(l+1)} \\ z_4^{(l+1)} \\ z_5^{(l+1)} \end{bmatrix} = \begin{bmatrix} f_1 z_1^{(l+1)} \\ f_2 z_1^{(l+1)} + f_1 z_2^{(l+1)} \\ f_3 z_1^{(l+1)} + f_2 z_2^{(l+1)} + f_1 z_3^{(l+1)} \\ f_3 z_2^{(l+1)} + f_2 z_3^{(l+1)} + f_1 z_4^{(l+1)} \\ f_3 z_3^{(l+1)} + f_2 z_4^{(l+1)} + f_1 z_5^{(l+1)} \\ f_3 z_4^{(l+1)} + f_2 z_5^{(l+1)} \\ f_3 z_5^{(l+1)} \end{bmatrix}, \tag{17}$$

а для stride=2:

$$\mathbf{y}^{(l)} = \begin{bmatrix} f_1 & 0 & 0 \\ f_2 & 0 & 0 \\ f_3 & f_1 & 0 \\ 0 & f_2 & 0 \\ 0 & f_3 & f_1 \\ 0 & 0 & f_2 \\ 0 & 0 & f_3 \end{bmatrix} \begin{bmatrix} z_1^{(l)} \\ z_2^{(l)} \\ z_3^{(l)} \end{bmatrix} = \begin{bmatrix} f_1 z_1^{(l)} \\ f_2 z_1^{(l)} \\ f_3 z_1^{(l)} + f_1 z_3^{(l)} \\ f_2 z_2^{(l)} \\ f_3 z_2^{(l)} + f_1 z_3^{(l)} \\ f_2 z_3^{(l)} \\ f_3 z_3^{(l)} \end{bmatrix} \tag{18}$$

В формуле 18 мы видим, что третий и пятый элементы итогового вектора вычисляются с использованием элементов сразу двух фильтров, в то время, как остальные с использованием лишь одного. Такое наложение на практике часто может приводить к артефактам вида шахматная доска[5]. Надо помнить, что в реальной сети мы не восстанавливаем исходные данные а получаем новую карту признаков, поэтому правильная запись будет такая:

$$\mathbf{z}^{(l+1)} = \mathbf{F}^T *^T \mathbf{y}^{(l)} \tag{19}$$

[4]Отсюда и название "транспонированная"

[5]Хороший пример реальных артефактов, а также визуализацию обратной свертки в двумерном случае можно посмотреть на сайте distill.pub

5.3 Свертка с использованием матрицы Тёплица (2D)

Рассмотрим двумерную карту признаков $\mathbf{Y}^{(\mathbf{l}}$ размером $3 \times 3 \times 1$ и двумерный фильтр $\mathbf{F}$ размером $3 \times 3 \times 1$ ($D^{(l)} = 1$):

$$\mathbf{Y}^{(\mathbf{l}} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}.$$

Предположим, что мы используем padding=same, сохраняющую размер карты признаков, тогда результат свертки должен быть

$$\mathrm{Z}^{(l+1)} = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{bmatrix}.$$

Как и до этого будем для простоты записи считать, что смещение не используется[6]. Теперь рассмотрим свертку с использованием матрицы Тёплица для двумерной свертки:

$$\begin{bmatrix} z_{11} \\ z_{12} \\ z_{13} \\ z_{12} \\ z_{22} \\ z_{23} \\ z_{31} \\ z_{32} \\ z_{33} \end{bmatrix} = \begin{bmatrix} f_{22} & f_{23} & 0 & f_{32} & f_{33} & 0 & 0 & 0 & 0 \\ f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} & 0 & 0 & 0 \\ 0 & f_{21} & f_{22} & 0 & f_{31} & f_{32} & 0 & 0 & 0 \\ f_{12} & f_{13} & 0 & f_{22} & f_{23} & 0 & f_{32} & f_{33} & 0 \\ f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \\ 0 & f_{11} & f_{12} & 0 & f_{21} & f_{22} & 0 & f_{31} & f_{32} \\ 0 & 0 & 0 & f_{12} & f_{13} & 0 & f_{22} & f_{23} & 0 \\ 0 & 0 & 0 & f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} \\ 0 & 0 & 0 & 0 & f_{11} & f_{12} & 0 & f_{21} & f_{22} \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}.$$

Можно убедиться, что представленное выражение обеспечивает вычисление свертки. Так например,

$$z_{22} = f_{11}y_{11} + f_{12}y_{12} + f_{13}y_{13} + f_{21}y_{21} + f_{22}y_{22} + f_{23}y_{23} + f_{31}y_{31} + f_{32}y_{32} + f_{33}y_{33}. \tag{23}$$

Теперь рассмотрим случай с использованием шага равного двум (stride=2). Для этого надо удалить строки для каждого второго столбца и каждой второй строки результирующей карты признаков:

[6] Добавление смещения является тривиальным

$$\begin{bmatrix} z_{11} \\ z_{13} \\ z_{31} \\ z_{33} \end{bmatrix} = \begin{bmatrix} f_{22} & f_{23} & 0 & f_{32} & f_{33} & 0 & 0 & 0 & 0 \\ 0 & f_{21} & f_{22} & 0 & f_{31} & f_{32} & 0 & 0 & 0 \\ 0 & 0 & 0 & f_{12} & f_{13} & 0 & f_{22} & f_{23} & 0 \\ 0 & 0 & 0 & 0 & f_{11} & f_{12} & 0 & f_{21} & f_{22} \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}.$$

5.4 Транспонированная свертка (2D)

Как и прежде, для получения траспонированной свертки достаточно просто, транспонировать матрицу Тёплица для выражения двумерной свертки 24:

$$\begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{12} \\ y_{22} \\ y_{32} \\ y_{13} \\ y_{23} \\ y_{33} \end{bmatrix} = \begin{bmatrix} f_{22} & 0 & 0 & 0 \\ f_{23} & f_{21} & 0 & 0 \\ 0 & f_{22} & 0 & 0 \\ f_{32} & 0 & f_{12} & 0 \\ f_{33} & f_{31} & f_{13} & f_{11} \\ 0 & f_{32} & 0 & f_{12} \\ 0 & 0 & f_{22} & 0 \\ 0 & 0 & f_{23} & f_{21} \\ 0 & 0 & 0 & f_{22} \end{bmatrix} \begin{bmatrix} z_{11} \\ z_{13} \\ z_{31} \\ z_{33} \end{bmatrix}. \tag{25}$$

Заметим, что для выполнения транспонированной свертки приходится использовать более разреженные матрицы большего размера, что делает ее менее вычислительно эффективной, чем саму свертку, при этом однако на практике требования к ресурсам приемлемы, поэтому транспонированная свертка используется достаточно часто. Также еще раз заметим, что при рассмотрении вопросов, связанных с транспонированной сверткой в записи пренебрегалось смещением (bias=0). Расширение записей для его использования является тривиальным.

6 Заключение

Было праведно описание сверточных слоев, а также другие основные элементы сверточной сети: паддинг и подвыборка. Был рассмотрен вопрос векторизации прямого прохода для сверточных нейронных сетей. Выполнение свертки может быть представлено в виде произведения матриц, а значит и прямой, и обратный проход могут быть выполнены аналогично тому, как это делается для полносвязных слоев. Заметим, что рекуррентные нейронные сети не имеют столь же удачных способов векторизации вычислений для обработки временных последовательностей, как сверточные сети, поскольку скрытое состояние рекуррентной сети в момент времени t зависит от скрытого состояния, полученного в предыдущий момент времени $t-1$. Это ведет к необходимости выполнения операций последовательно; такая

неэффективность является одной из причин, почему в настоящее время для обработки временных последовательностей используется другая архитектура - трансформер. Мы также рассмотрели транспонированную свертку: сперва для простоты был рассмотрен случай одномерной свертки с использованием матрицы Тёплица. Транспонирование этой матрицы позволяет получать операцию транспонированной свертки. Для полноты был рассмотрен и случай двумерной транспонированной свертки с использованием матрицы Тёплица.