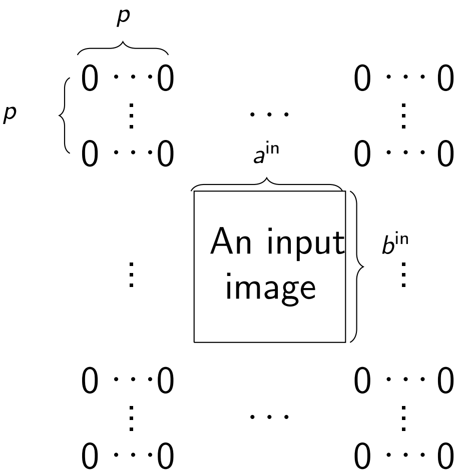
Other Operations in CNN I

- CNN involves additional operations in practice
  - padding
  - pooling
- We will explain them in detail

Zero Padding I

- To better control the size of the output image, before the convolutional operation we may enlarge the input image to have zero values around the border.
- For example, sometimes we would like the output image of the convolutional layer to have the same size as the input image
- This technique is called zero-padding in CNN training.
- An illustration:

Zero Padding II



Zero Padding III

- The size of the new image is changed from

$$a^{\text{in}} \times b^{\text{in}} \text{ to } (a^{\text{in}} + 2p) \times (b^{\text{in}} + 2p),$$

  where p is specified by users
- The operation can be treated as a layer of mapping an input $Z^{\text{in},i}$ to an output $Z^{\text{out},i}$.
- Let

$$d^{\text{out}} = d^{\text{in}}.$$

Zero Padding IV

- There exists a 0/1 matrix

$$P_{\text{pad}} \in R^{d^{\text{out}} a^{\text{out}} b^{\text{out}} \times d^{\text{in}} a^{\text{in}} b^{\text{in}}}$$

  so that the padding operation can be represented by

$$Z^{\text{out},i} \equiv \text{mat}(P_{\text{pad}} \text{vec}(Z^{\text{in},i}))_{d^{\text{out}} \times a^{\text{out}} b^{\text{out}}}. \qquad (1)$$

Pooling I

- To reduce the computational cost, a dimension reduction is often applied by a pooling step after convolutional operations.
- Usually we consider an operation that can (approximately) extract rotational or translational invariance features.
- Examples: average pooling, max pooling, and stochastic pooling,
- Let's consider max pooling as an illustration

Pooling II

- An example:

$$\text{image A} \quad \left[\begin{array}{cc|cc} 2 & 3 & 6 & 8 \\ 5 & 4 & 9 & 7 \\ \hline 1 & 2 & 6 & 0 \\ 4 & 3 & 2 & 1 \end{array}\right] \to \begin{bmatrix} 5 & 9 \\ 4 & 6 \end{bmatrix}$$

$$\text{image B} \quad \left[\begin{array}{cc|cc} 3 & 2 & 3 & 6 \\ 4 & 5 & 4 & 9 \\ \hline 2 & 1 & 2 & 6 \\ 3 & 4 & 3 & 2 \end{array}\right] \to \begin{bmatrix} 5 & 9 \\ 4 & 6 \end{bmatrix}$$

Pooling III

- B is derived by shifting A by 1 pixel in the horizontal direction.
- We split two images into four 2×2 sub-images and choose the max value from every sub-image.
- In each sub-image because only some elements are changed, the maximal value is likely the same or similar.
- This is called translational invariance
- For our example the two output images from A and B are the same.

Pooling IV

- For mathematical representation, we consider the operation as a layer of mapping an input $Z^{\text{in},i}$ to an output $Z^{\text{out},i}$.
- In practice pooling is considered as an operation at the end of the convolutional layer.
- We partition every channel of $Z^{\text{in},i}$ into non-overlapping sub-regions by $h \times h$ filters with the stride $s = h$
- Because of the disjoint sub-regions, the stride s for sliding the filters is equal to h.

Pooling V

- This partition step is a special case of how we generate sub-images in convolutional operations.
- By the same definition earlier for convolutional operations, we can generate the matrix

$$\phi(Z^{\text{in},i}) = \text{mat}(P_\phi \text{vec}(Z^{\text{in},i}))_{hh \times d^{\text{out}} a^{\text{out}} b^{\text{out}}}, \tag{2}$$

where

$$a^{\text{out}} = \lfloor \frac{a^{\text{in}}}{h} \rfloor, \; b^{\text{out}} = \lfloor \frac{b^{\text{in}}}{h} \rfloor, \; d^{\text{out}} = d^{\text{in}}. \tag{3}$$

Pooling VI

- This is the same as the earlier calculation on the output-image size of convolutional operations

$$\lfloor \frac{a^{\text{in}} - h}{h} \rfloor + 1 = \lfloor \frac{a^{\text{in}}}{h} \rfloor$$

- Note that here we consider

$$hh \times d^{\text{out}} a^{\text{out}} b^{\text{out}} \text{ rather than } hhd^{\text{out}} \times a^{\text{out}} b^{\text{out}}$$

because we can then do a max operation on each column

Pooling VII

- To select the largest element of each sub-region, there exists a 0/1 matrix

$$M^i \in R^{d^{\text{out}} a^{\text{out}} b^{\text{out}} \times h h d^{\text{out}} a^{\text{out}} b^{\text{out}}}$$

  so that each row of M^i selects a single element from $\text{vec}(\phi(Z^{\text{in},i}))$.
- Therefore,

$$Z^{\text{out},i} = \text{mat}\left(M^i \text{vec}(\phi(Z^{\text{in},i}))\right)_{d^{\text{out}} \times a^{\text{out}} b^{\text{out}}}. \tag{4}$$

Pooling VIII

- A comparison with

$$S^{\text{out},i} = W\phi(Z^{\text{in},i}) + \boldsymbol{b}\mathbb{1}^T_{a^{\text{out}}b^{\text{out}}}$$

  in convolutional operations shows that M^i is in a similar role to the weight matrix W
- While M^i is $0/1$, it is not a constant. It's positions of 1's depend on the values of $\phi(Z^{\text{in},i})$

Pooling IX

- By combining (2) and (4), we have

$$Z^{\text{out},i} = \text{mat}\left(P^{i}_{\text{pool}}\text{vec}(Z^{\text{in},i})\right)_{d^{\text{out}} \times a^{\text{out}} b^{\text{out}}}, \tag{5}$$

where

$$P^{i}_{\text{pool}} = M^{i} P_{\phi} \in R^{d^{\text{out}} a^{\text{out}} b^{\text{out}} \times d^{\text{in}} a^{\text{in}} b^{\text{in}}}. \tag{6}$$