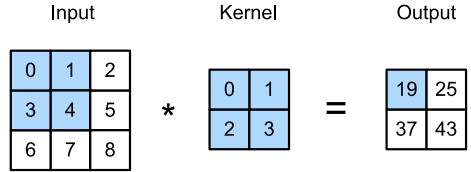
6.2 Convolutional Neural Networks

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

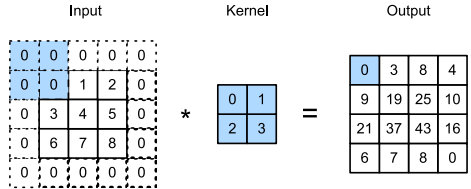
Digital Health & Machine Learning

Convolutional Layer



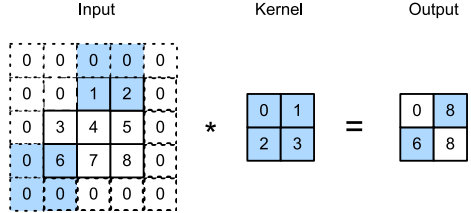
- The input is a two-dimensional array with a shape $n_h \times n_w$.
- The shape of the **kernel** (or **filter**) array is $k_h \times k_w$.
- The output has a shape of $(n_h - k_h + 1) \times (n_w - k_w + 1)$.

Padding



- The input is a two-dimensional array with a shape $n_h \times n_w$.
- The shape of the **kernel** (or **filter**) array is $k_h \times k_w$.
- We add p_h rows and p_w columns of padding
- The output has a shape of $(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$.

Stride



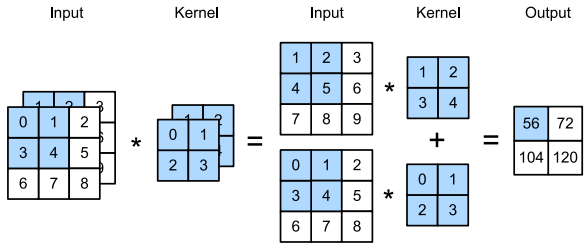
- The input is a two-dimensional array with a shape $n_h \times n_w$.
- The shape of the **kernel** (or **filter**) array is $k_h \times k_w$.
- We add p_h rows and p_w columns of padding.
- We apply a stride s_h for the height and s_w for the width.
- $$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$
- If $p_h = k_h - 1$ and $p_w = k_w - 1$,
  then $\lfloor (n_h + s_h - 1)/s_h \rfloor \times \lfloor (n_w + s_w - 1)/s_w \rfloor$.
- If the input height and width are divisible by the strides on the height and width,
  then $(n_h/s_h) \times (n_w/s_w)$.

Summary

- Padding can increase the height and width of the output.

  This is often used to give the output the same height and width as the input.
- The stride can reduce the resolution of the output, for example reducing the height and width of the output to only $1/n$ of the height and width of the input (n is an integer greater than 1).
- Padding and stride can be used to adjust the dimensionality of the data effectively.

Multiple input channels



$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56.$$

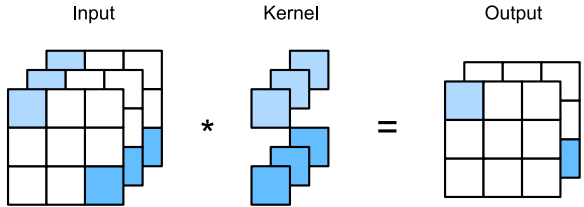
- c_i : Number of channels for the input data and number of input channels of the convolution kernel.
- If $c_i > 1$, the kernel contains an array of shape $k_h \times k_w$ *for each input channel*.
- Convolution kernel of shape: $c_i \times k_h \times k_w$.
- We perform a cross-correlation operation for each channel, adding the c_i results together (summing over the channels).

Multiple output channels

- Typically, we increase the channel dimension as we go higher up in the neural network, while decreasing the spatial resolution.
- c_i the number of input channels
- c_o the number of output channels
- kernel height k_h and width k_w
- We create a kernel array of shape $c_i \times k_h \times k_w$ for each output channel.
- the shape of the convolution kernel is $c_o \times c_i \times k_h \times k_w$.

1×1 Convolutional Layer

What happens in a 1×1 convolution, i.e. $k_h = k_w = 1$?



- 1×1 convolution loses the ability to recognize patterns in the height and width dimensions.
- The only computation occurs on the channel dimension.
- Inputs and outputs have the same height and width.
- Each element in the output is derived from a linear combination of elements *at the same position* in the input image.
- Equivalent to fully-connected layer at every single pixel from c_i input channels to c_o output channels.
- The $c_o \times c_i$ weights are tied across all pixels.

Summary

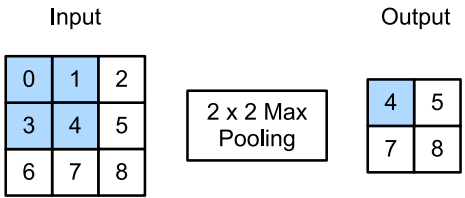
- Multiple channels can be used to extend the model parameters of the convolutional layer.
- The 1×1 convolutional layer is equivalent to the fully-connected layer, when applied on a per pixel basis.
- The 1×1 convolutional layer is typically used to adjust the number of channels between network layers and to control model complexity.

Pooling layers serve the dual purposes of

(i) mitigating the sensitivity of convolutional layers to location

(ii) of spatially downsampling the representation.

Example (Maximum pooling with a pooling window shape of 2×2)



$$\max(0, 1, 3, 4) = 4,$$
$$\max(1, 2, 4, 5) = 5,$$
$$\max(3, 4, 6, 7) = 7,$$
$$\max(4, 5, 7, 8) = 8.$$

- A pooling layer with a pooling window shape of $p \times q$ is called a $p \times q$ pooling layer.
- The pooling operation is called $p \times q$ pooling.

- Pooling operators consist of a fixed-shape window that is slid over all regions in the input according to its stride.
- It computes a single output for each location traversed by the fixed-shape window (sometimes known as the *pooling window*).
- Pooling layers contain no parameters (there is no *filter*).
- Pooling operators are deterministic, typically calculating either the **maximum** or the **average** value of the elements in the pooling window.
- When processing multi-channel input data, the pooling layer pools each input channel separately, rather than adding the inputs of each channel by channel as in a convolutional layer.
- Thus, the number of output channels for the pooling layer is the same as the number of input channels.

Summary

- Taking the input elements in the pooling window, the maximum pooling operation assigns the maximum value as the output and the average pooling operation assigns the average value as the output.
- Pooling layers alleviate the excessive sensitivity of the convolutional layer to location.
- We can specify the padding and stride for the pooling layer.
- Maximum pooling, combined with a stride larger than 1 can be used to reduce the resolution.
- The pooling layer's number of output channels is the same as the number of input channels.

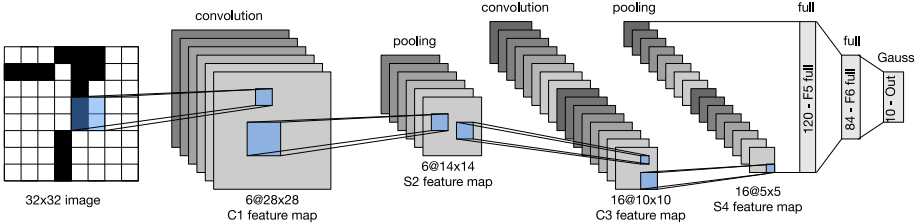
- **LeNet** was one of the first convolutional neural networks for recognizing handwritten digits in images—LeNet5 [1].
- In the 90s, Yann Lecun, at AT&T Bell Labs gave the first compelling evidence that it was possible to train convolutional neural networks by backpropagation.
- Their model was adopted to recognize digits for processing deposits in ATM machines.

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

LeNet consists of

i. a **block** of convolutional layers

ii. a block of fully-connected layers.



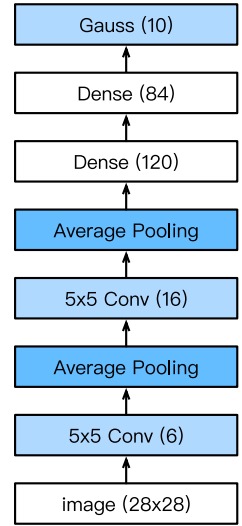


Figure: Compressed notation for LeNet5

Summary

- A convolutional neural network (in short, ConvNet) is a network using convolutional layers.
- In a ConvNet we alternate between convolutions, non-linearities and often also pooling operations.
- Ultimately the resolution is reduced prior to emitting an output via one (or more) dense layers.
- LeNet was the first successful deployment of such a network.