Convolutional Neural Networks

Maria Florina Balcan

April 1^{2t} 2019

Convolutional neural networks

- A specialized kind of neural network for processing data that has a known grid-like topology.
  - E.g., time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels
- The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution . Convolution is a specialized kind of linear operation.
- Convolutional networks are neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Convolutional neural networks

- Strong empirical application performance

- Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers

$$h = \sigma(W^T x + b)$$

for a specific kind of weight matrix W

Convolution

Convolution: discrete version

- Given array u_t and w_t, their convolution is a function s_t

$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

- Written as

$$s = (u * w) \quad \text{or} \quad s_t = (u * w)_t$$

- When u_t or w_t is not defined, assumed to be 0

Convolution, Motivation

- Suppose we track the location of a spaceship with a laser sensor. The laser sensor provides a single output u(t), the position of the spaceship at second t.

- Suppose sensor is noisy. To obtain a less noisy estimate of the spaceship's position, we average several measurements. More recent measurements are more relevant, so we use a weighted average that gives more weight to recent measurements.

- Use a weighting function w(a), where a is the age of a measurement. If we apply such a weighted average operation at every moment, we obtain a new function s providing a smoothed estimate of the position of the spaceship:

$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

Illustration 1

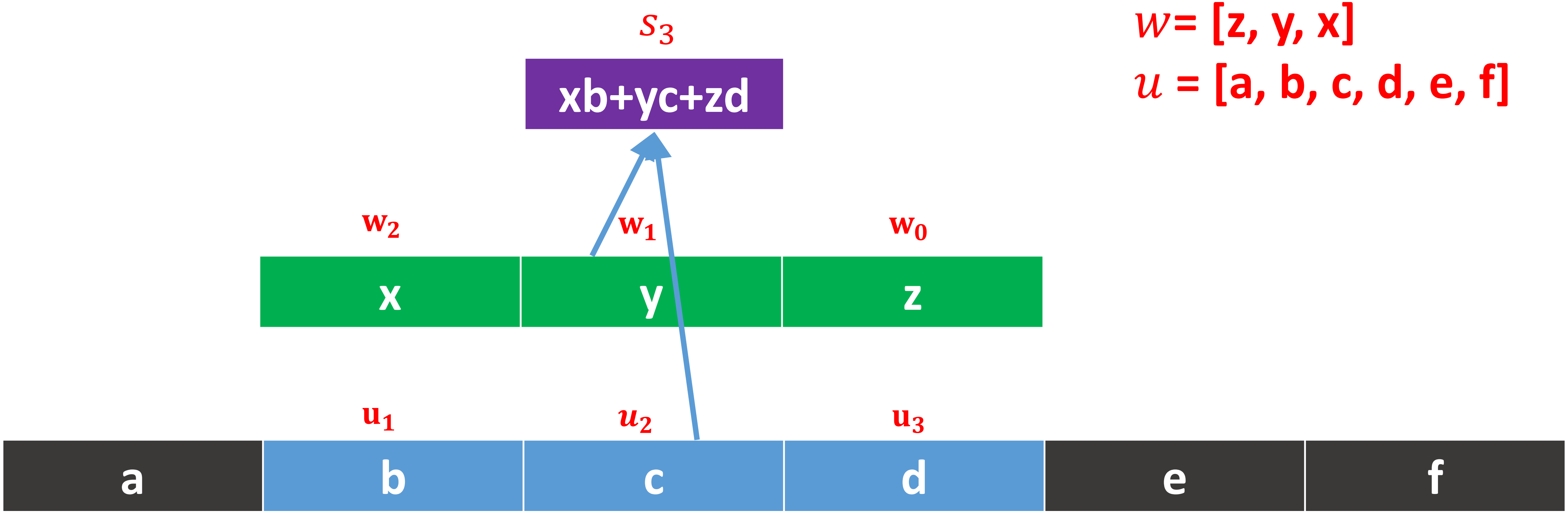


Illustration 1

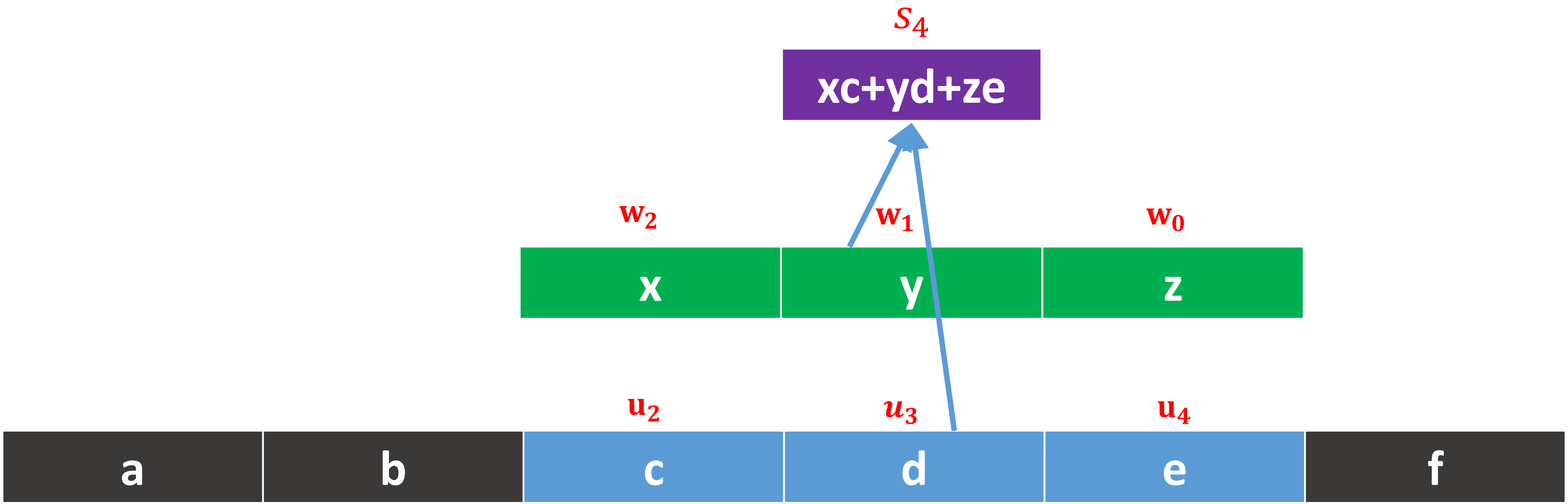


Illustration 1

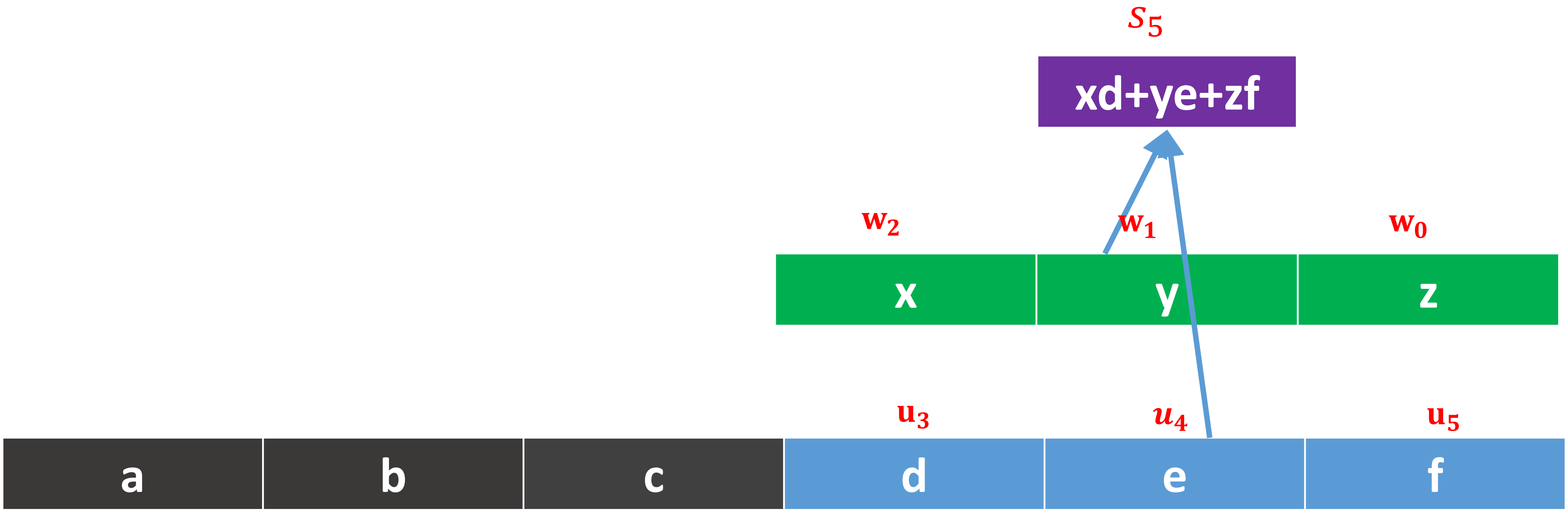


Illustration 1: boundary case

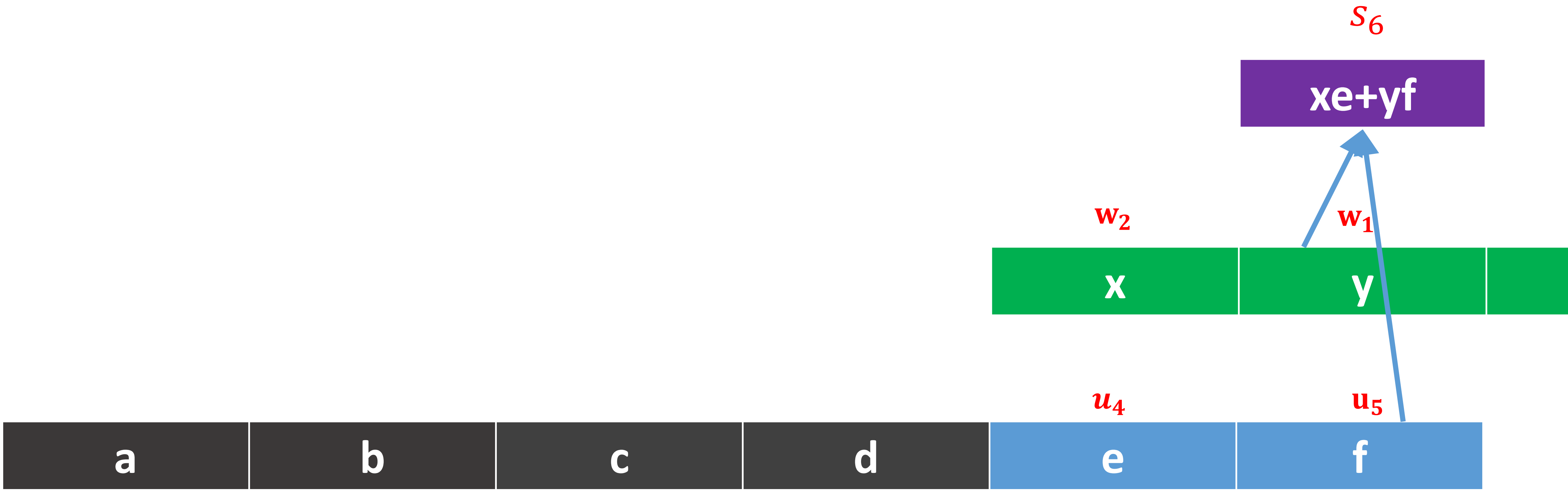


Illustration 1 as matrix multiplication

$$
\begin{bmatrix}
y & z & & & & \\
x & y & z & & & \\
 & x & y & z & & \\
 & & x & y & z & \\
 & & & x & y & z \\
 & & & & x & y
\end{bmatrix}
\quad
\begin{bmatrix}
a \\ b \\ c \\ d \\ e \\ f
\end{bmatrix}
$$

Illustration 2: two dimensional case

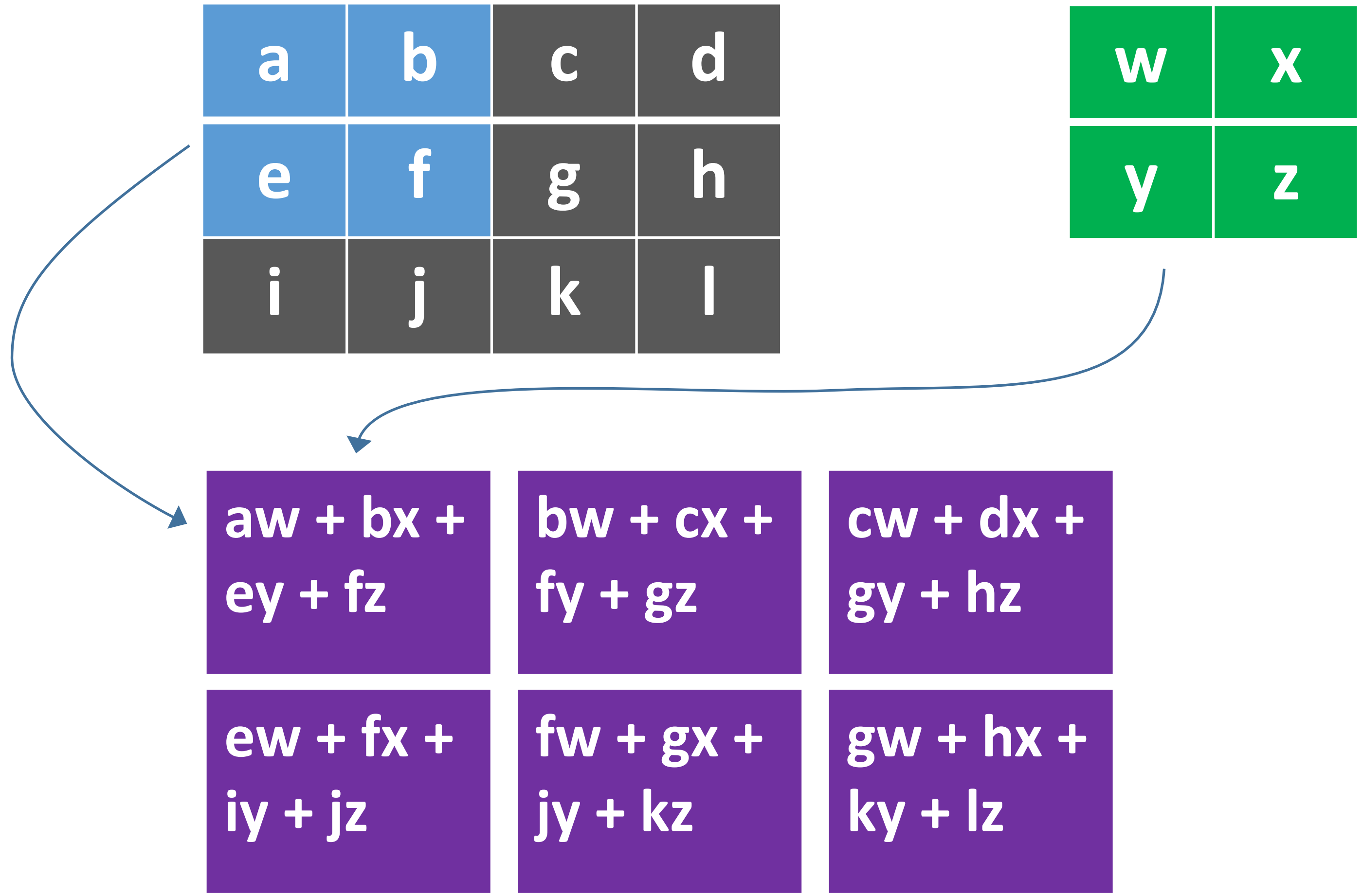


Illustration 2: two dimensional case

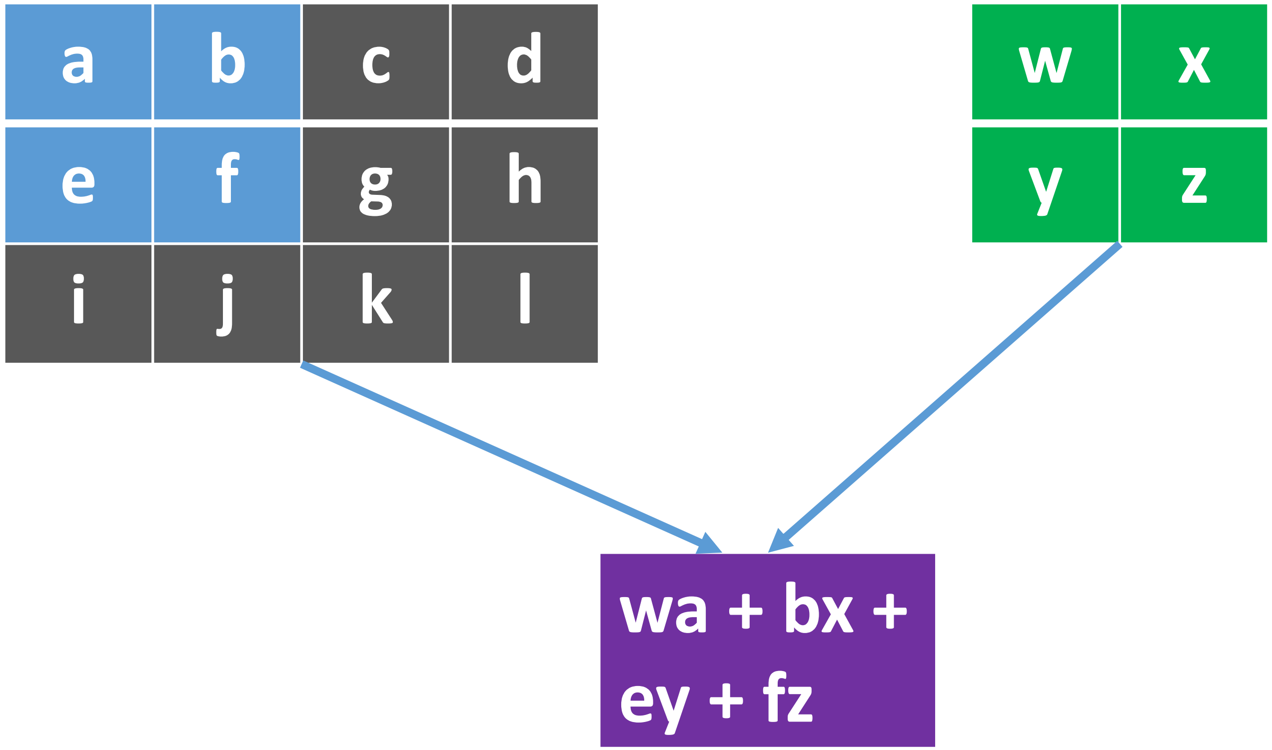


Illustration 2

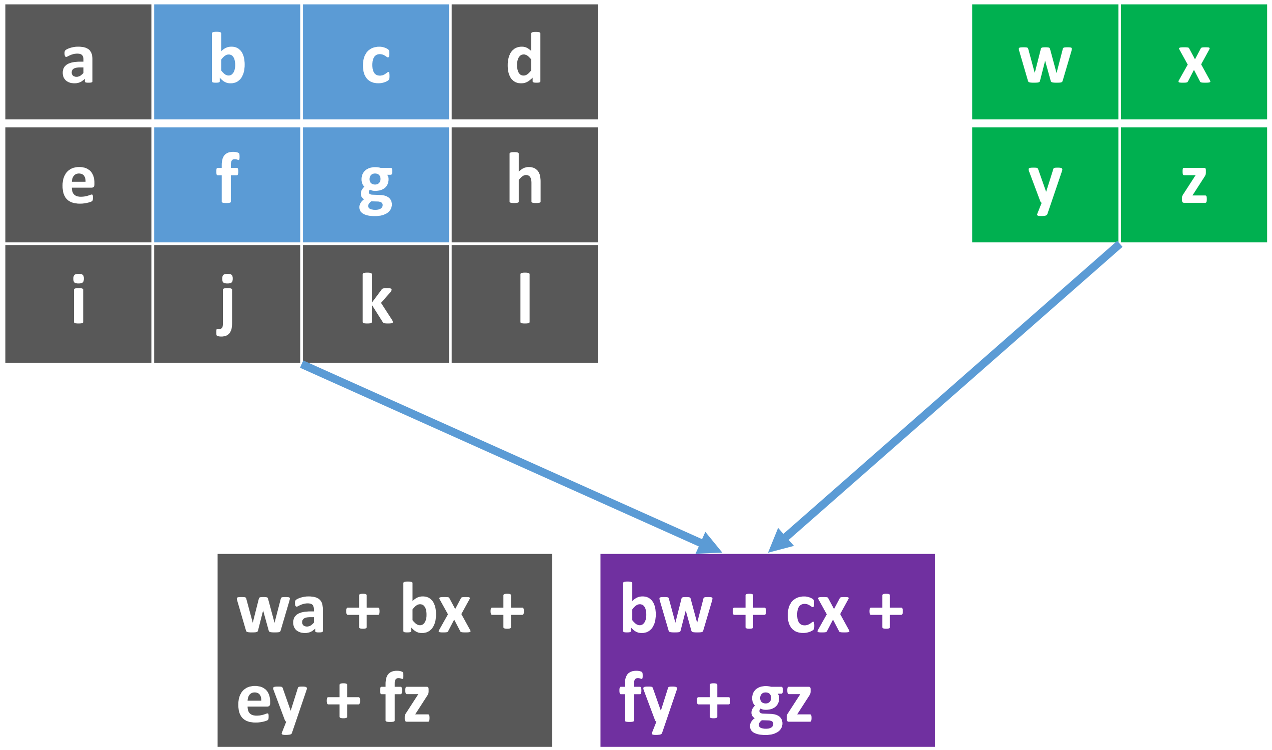


Illustration 2

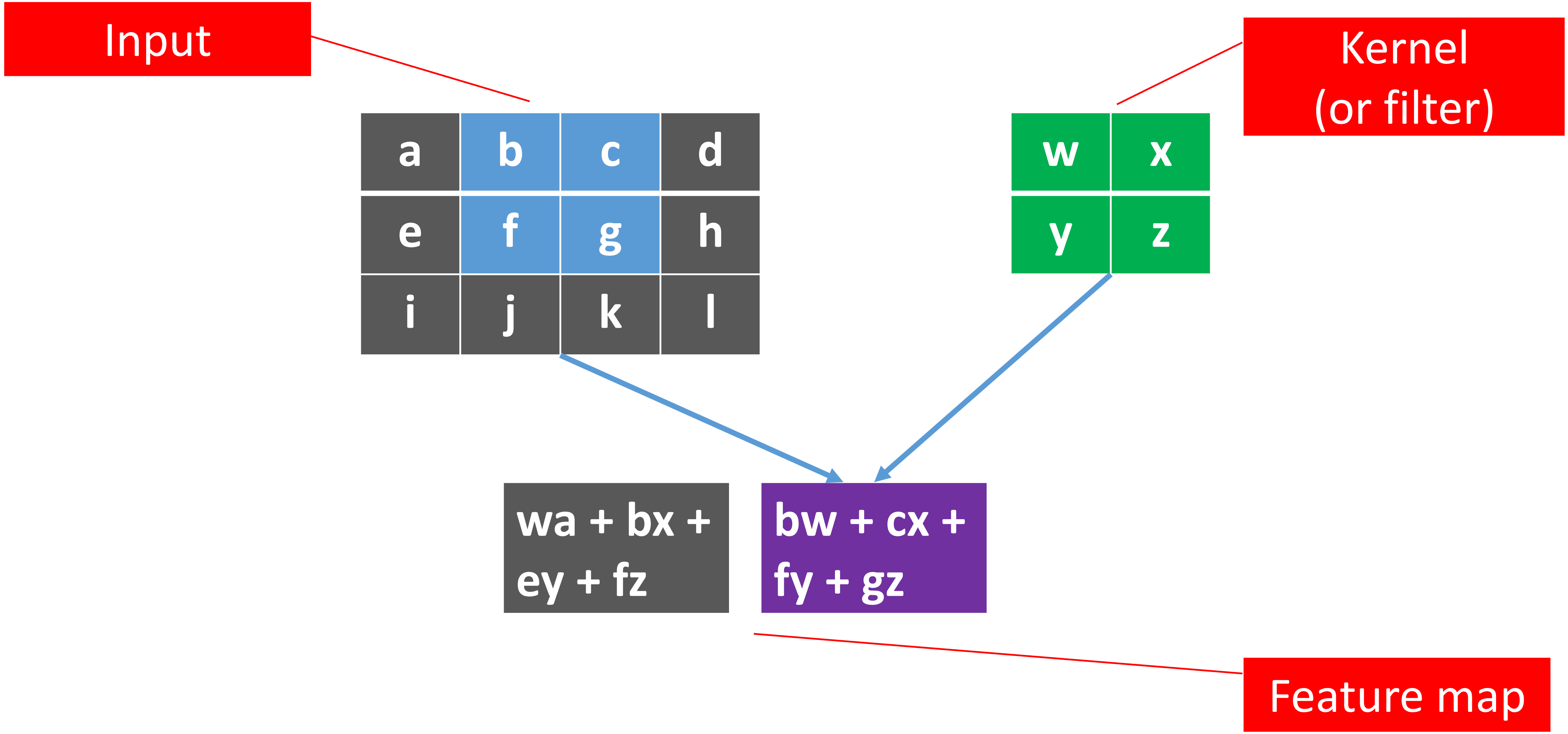
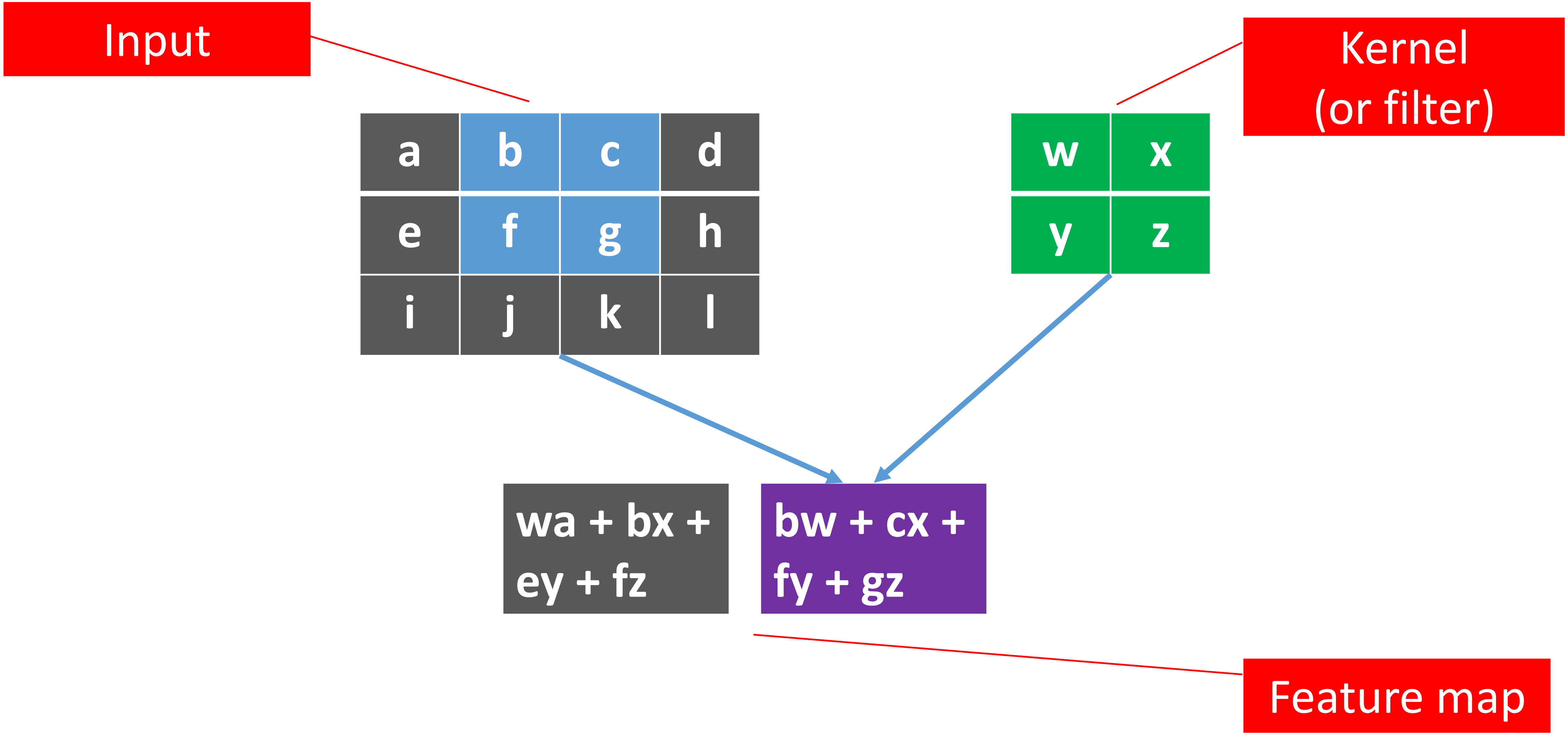
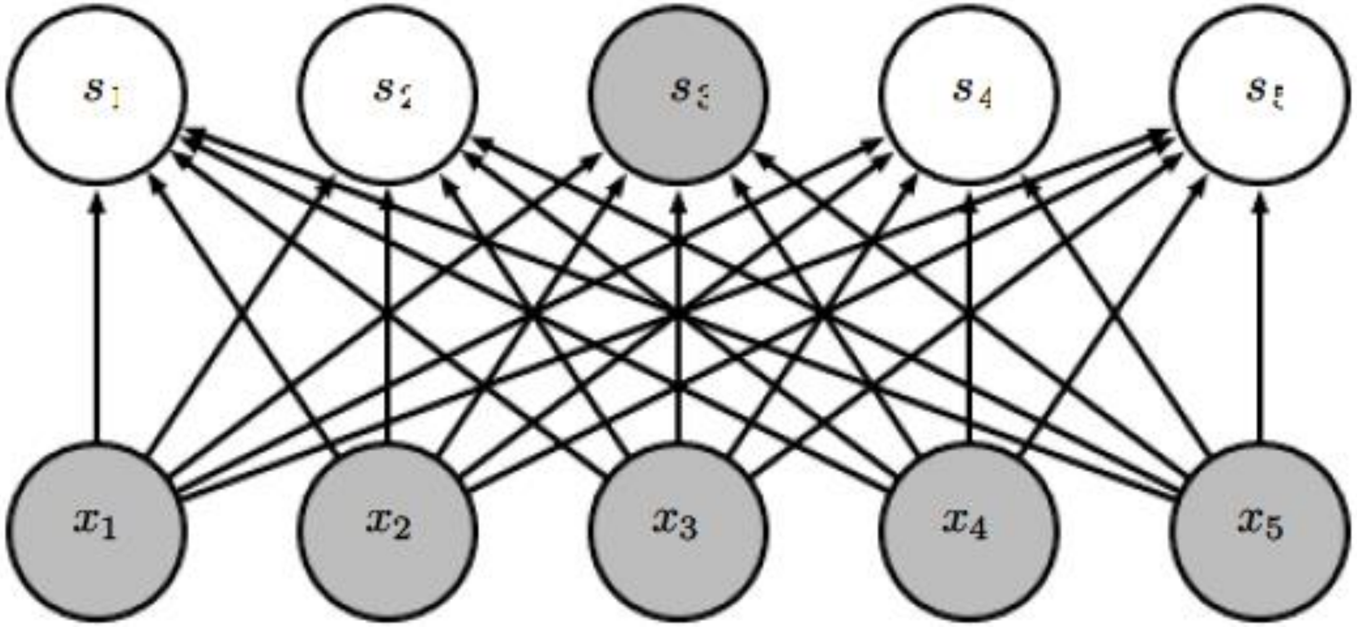


Illustration 2



Advantage: sparse interaction

Fully connected layer, $m \times n$ edges



m output nodes

n input nodes

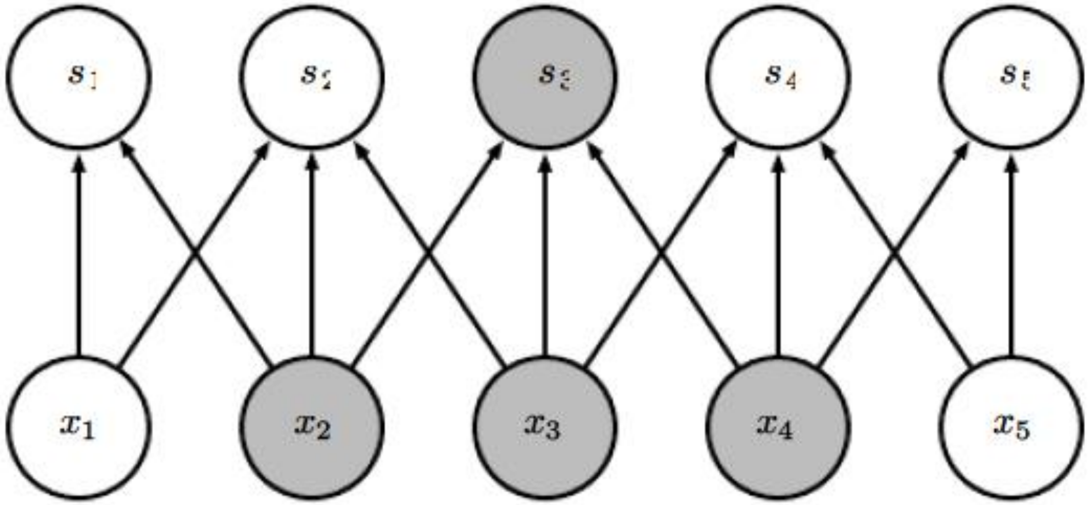
Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Advantage: sparse interaction

Convolutional layer, $\leq m \times k$ edges

Store fewer parameters:

- reduces memory requirements
- improves statistical efficiency.



m output nodes

k kernel size

n input nodes

Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Advantage: sparse interaction

Multiple convolutional layers: larger receptive field

- Receptive field of units in deeper layers larger than receptive field of units in shallow layers.
- Even though direct connections are sparse, units in the deeper layers are indirectly connected most of the input image.
- At the first layer capture more local features, but as we go deeper in the network we capture more global features.

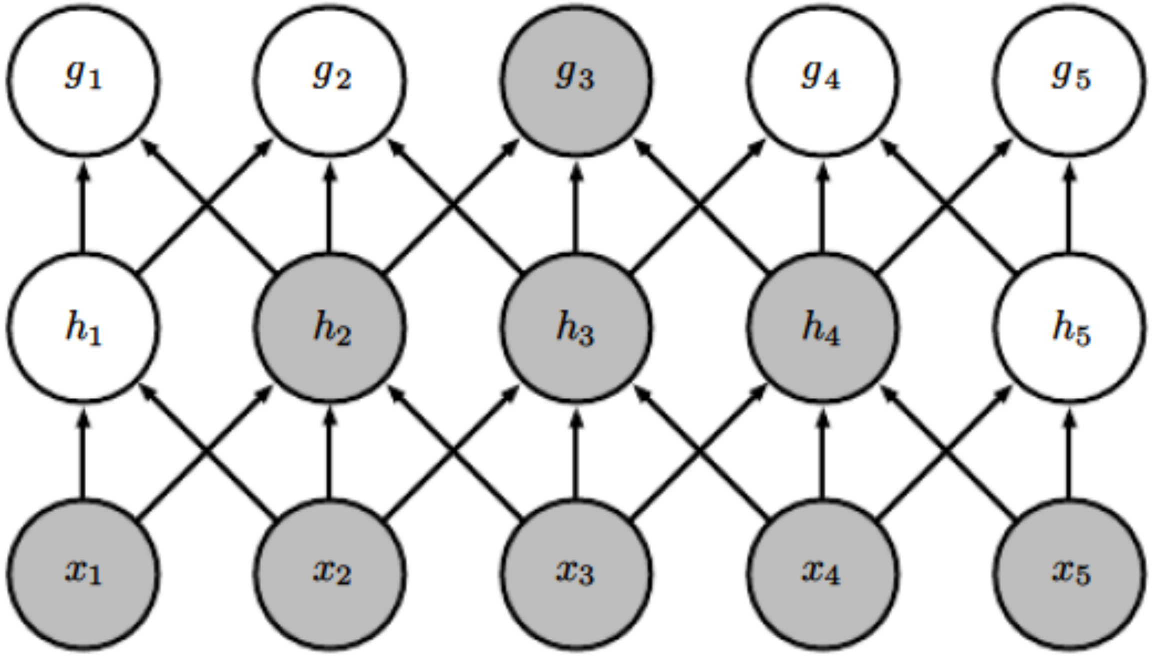
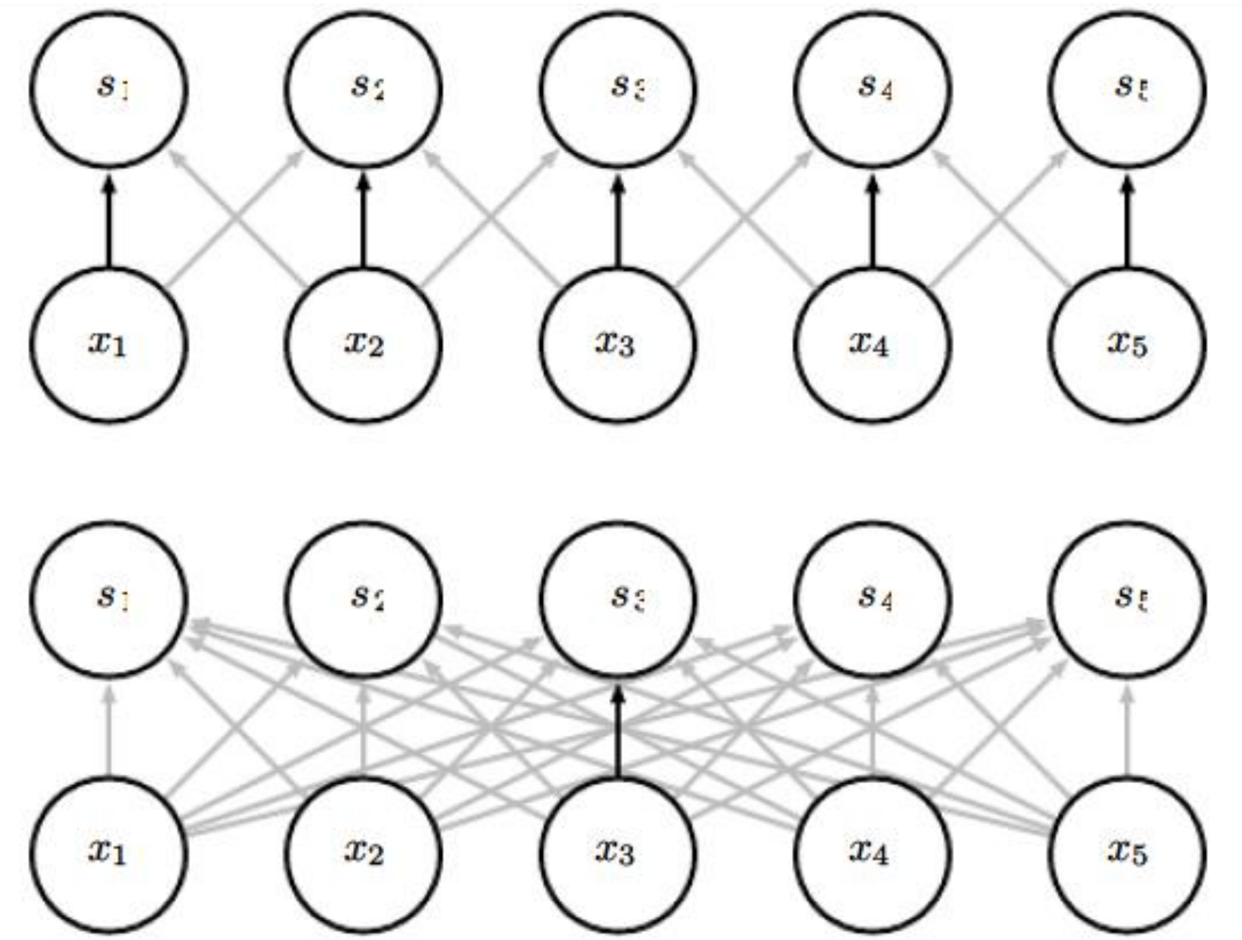


Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Advantage: parameter sharing



The same kernel are used repeatedly. E.g., the black edge is the same weight in the kernel.

Reduce the storage requirements of the model.

Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Advantage: equivariant representations

- Equivariant: transforming the input = transforming the output

- Example: input is an image, transformation is shifting
- Convolution(shift(input)) = shift(Convolution(input))

- Useful when care only about the existence of a pattern, rather than the location

Pooling

Terminology

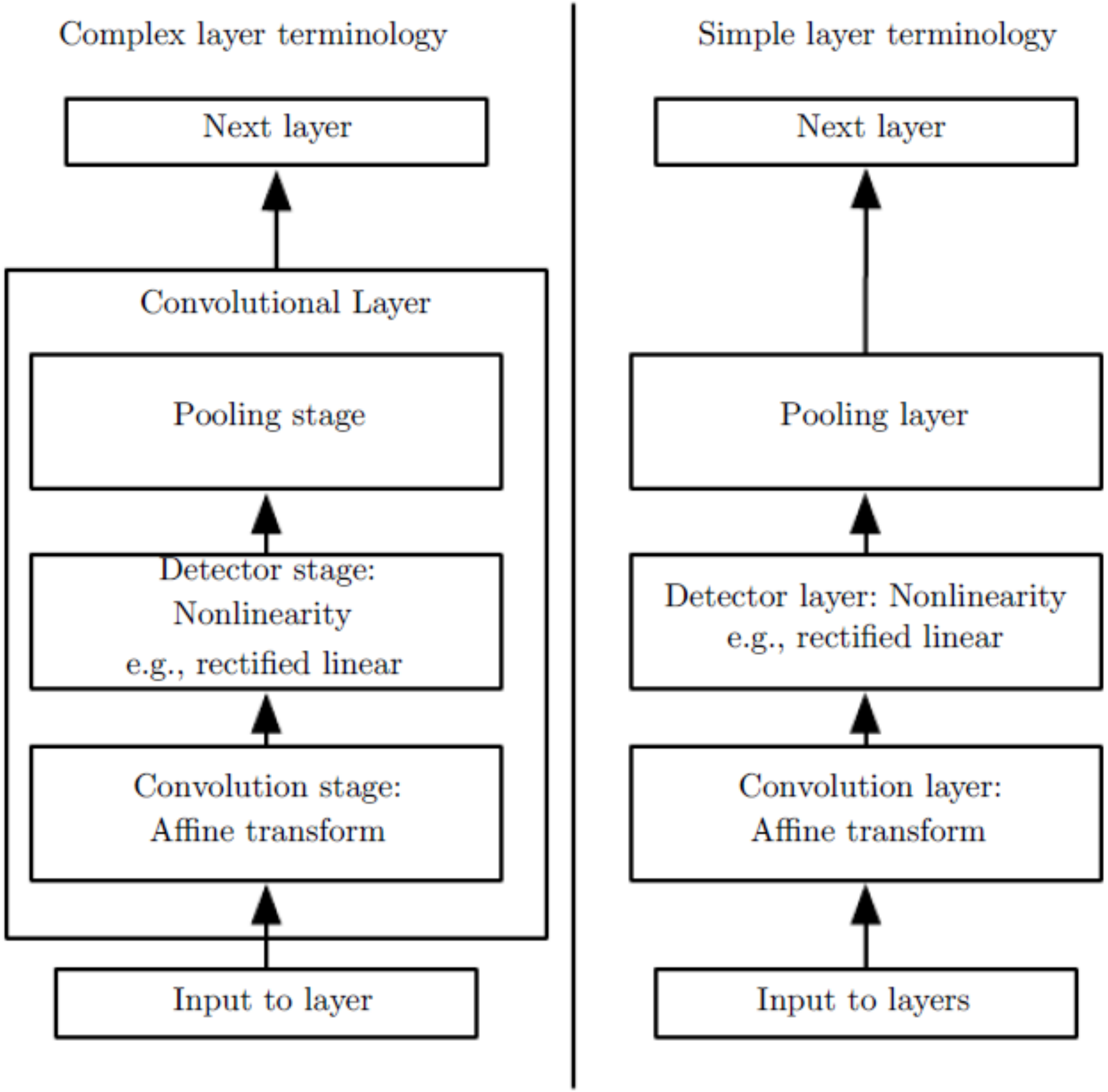
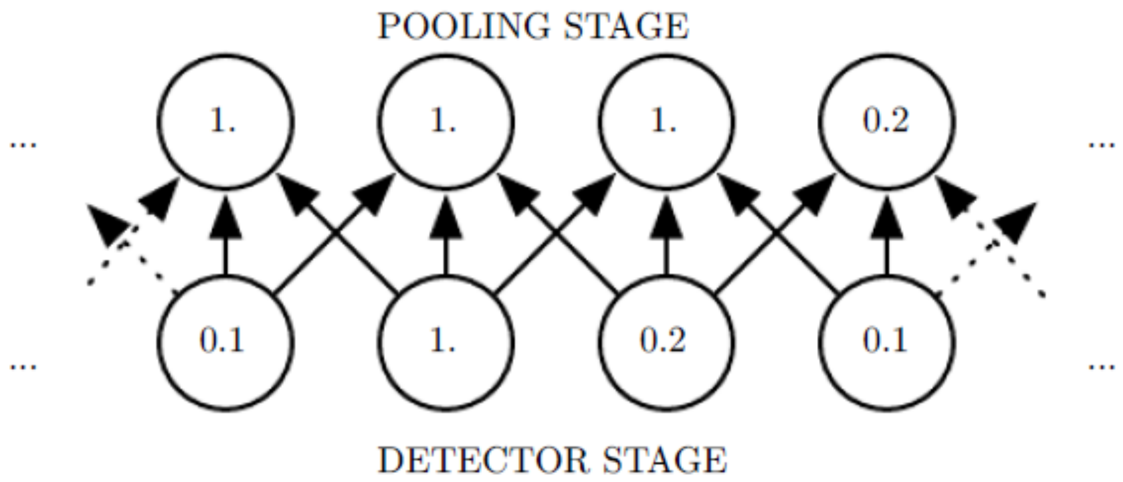


Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Pooling

- Summarizing the input (i.e., output the max of the input)



A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. . For example, the max pooling takes maximum output within a rectangular neighborhood.

Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Advantage

Induce invariance

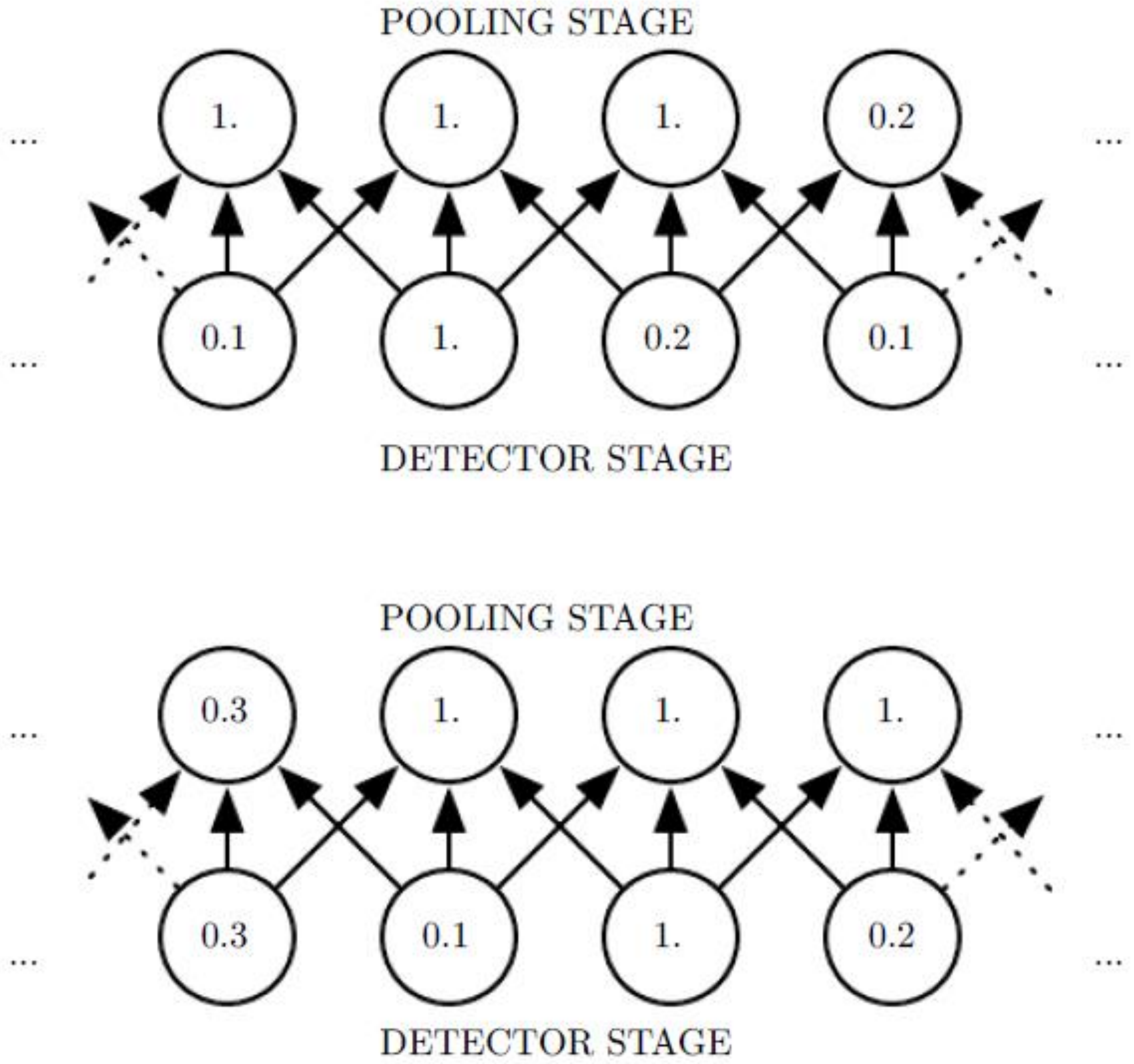


Figure from *Deep Learning,*
by Goodfellow, Bengio,
and Courville

Variants of pooling

- Max pooling $y = \max\{x_1, x_2, \dots, x_k\}$
- Average pooling $y = \mathrm{mean}\{x_1, x_2, \dots, x_k\}$

- Others like max-out

Motivation from neuroscience

- David Hubel and Torsten Wiesel studied early visual system in human brain (V1 or primary visual cortex), and won Nobel prize for this

- V1 properties
  - 2D spatial arrangement
  - Simple cells: inspire convolution layers
  - Complex cells: inspire pooling layers

Variants of convolution and pooling

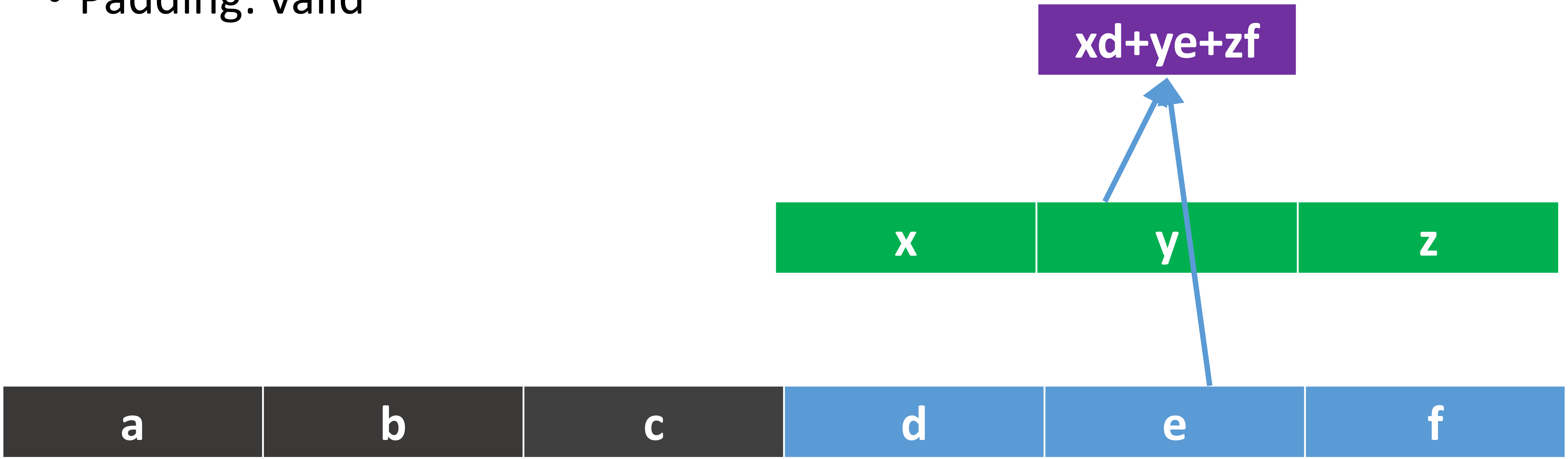
Variants of convolutional layers

- Multiple dimensional convolution

- Input and kernel can be 3D
  - E.g., images have (width, height, RBG channels)
- Multiple kernels lead to multiple feature maps (also called channels)

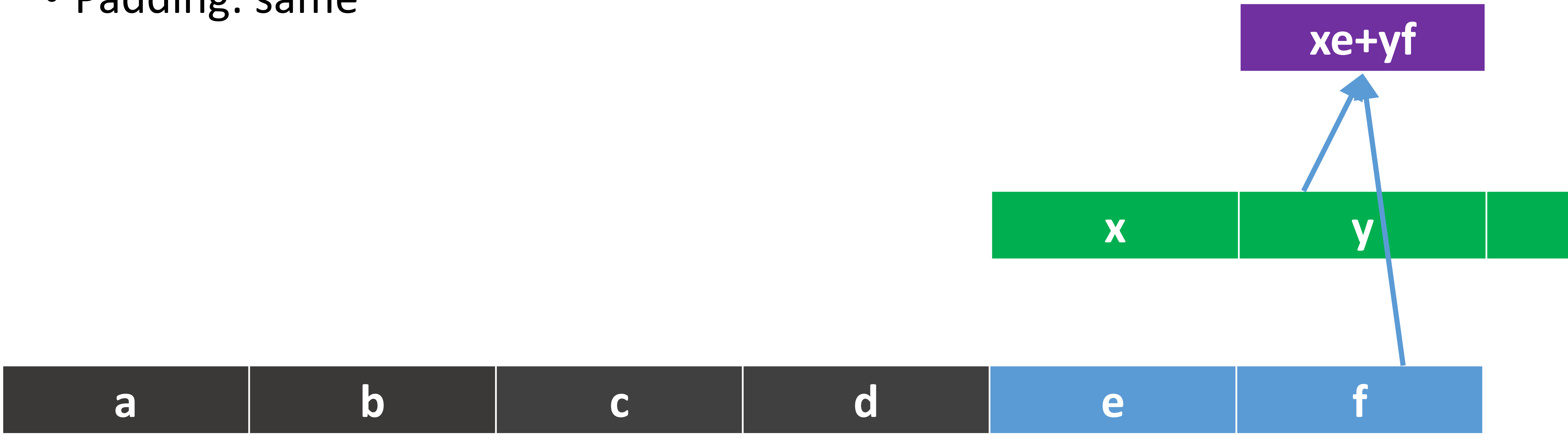
Variants of convolutional layers

- Padding: valid



Variants of convolutional layers

- Padding: same



Variants of convolutional layers

- Stride

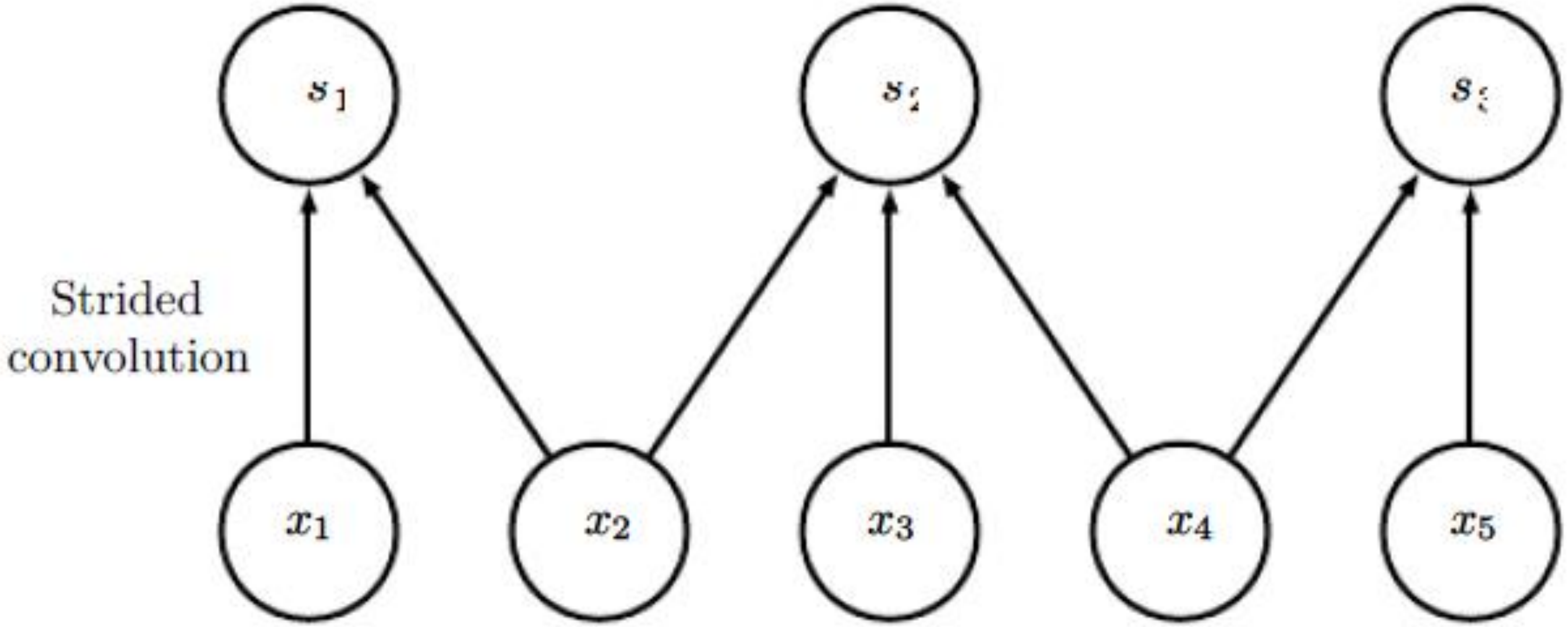


Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Variants of pooling

- Stride and padding

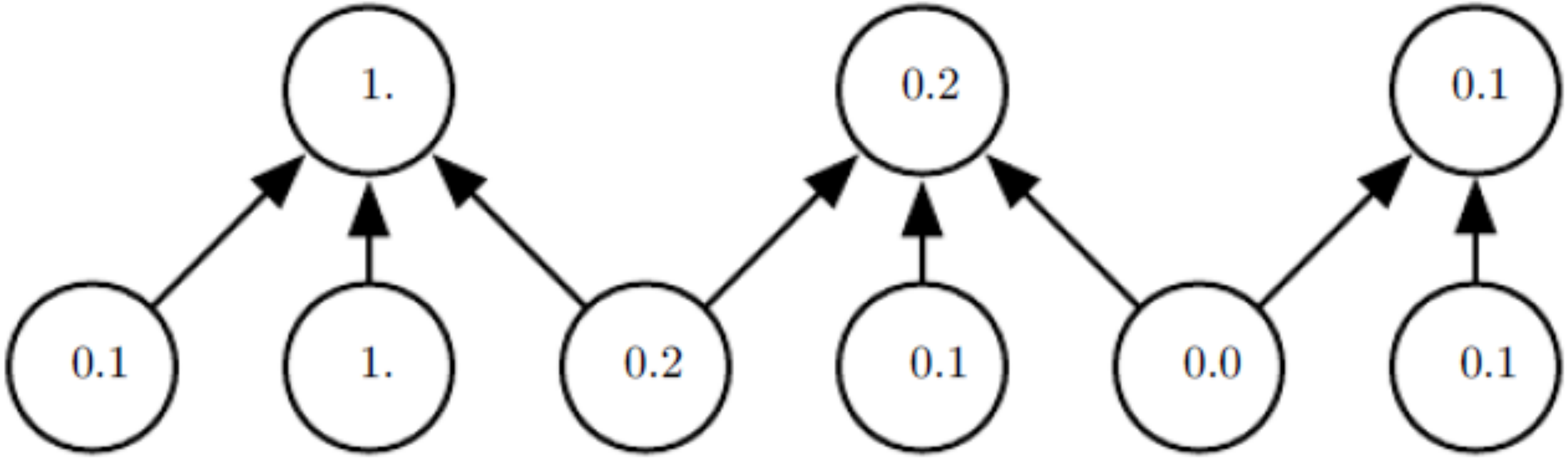


Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Case study: LeNet-5

LeNet-5

- Proposed in *"Gradient-based learning applied to document recognition", by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

LeNet-5

- Proposed in *"Gradient-based learning applied to document recognition", by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

- Apply convolution on 2D images (MNIST) and use backpropagation

LeNet-5

- Proposed in *“Gradient-based learning applied to document recognition”*, *by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

- Apply convolution on 2D images (MNIST) and use backpropagation

- Structure: 2 convolutional layers (with pooling) + 3 fully connected layers
  - Input size: 32x32x1
  - Convolution kernel size: 5x5
  - Pooling: 2x2

LeNet-5

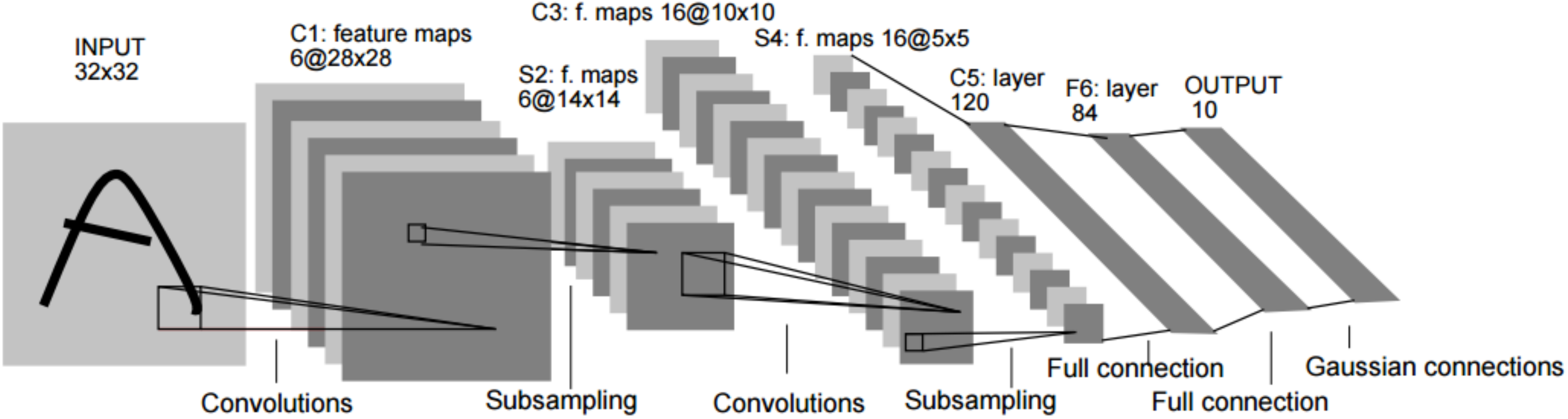


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

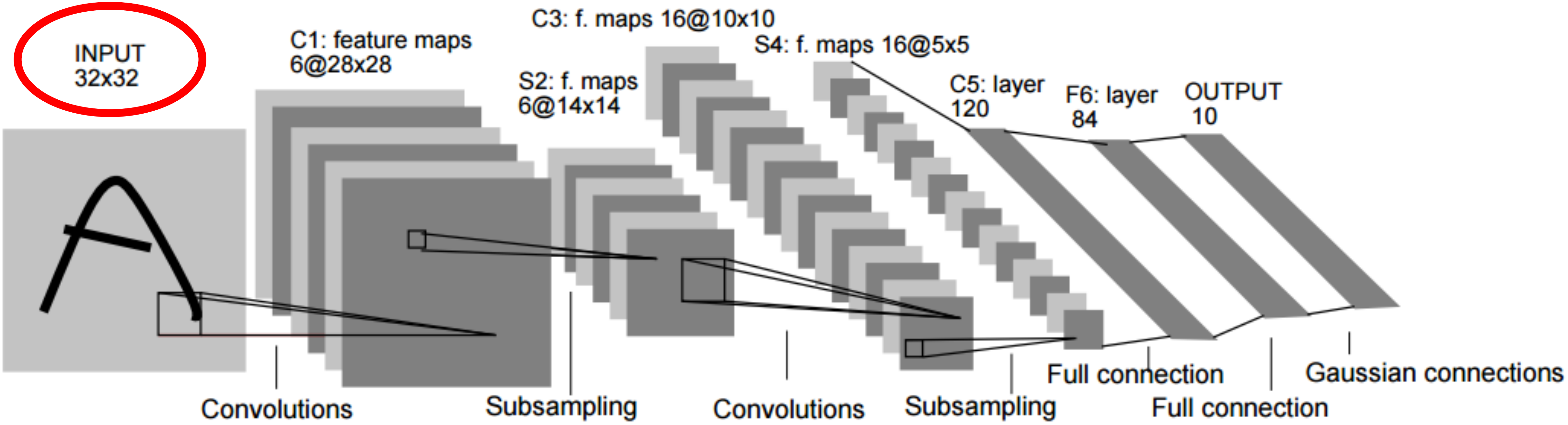


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

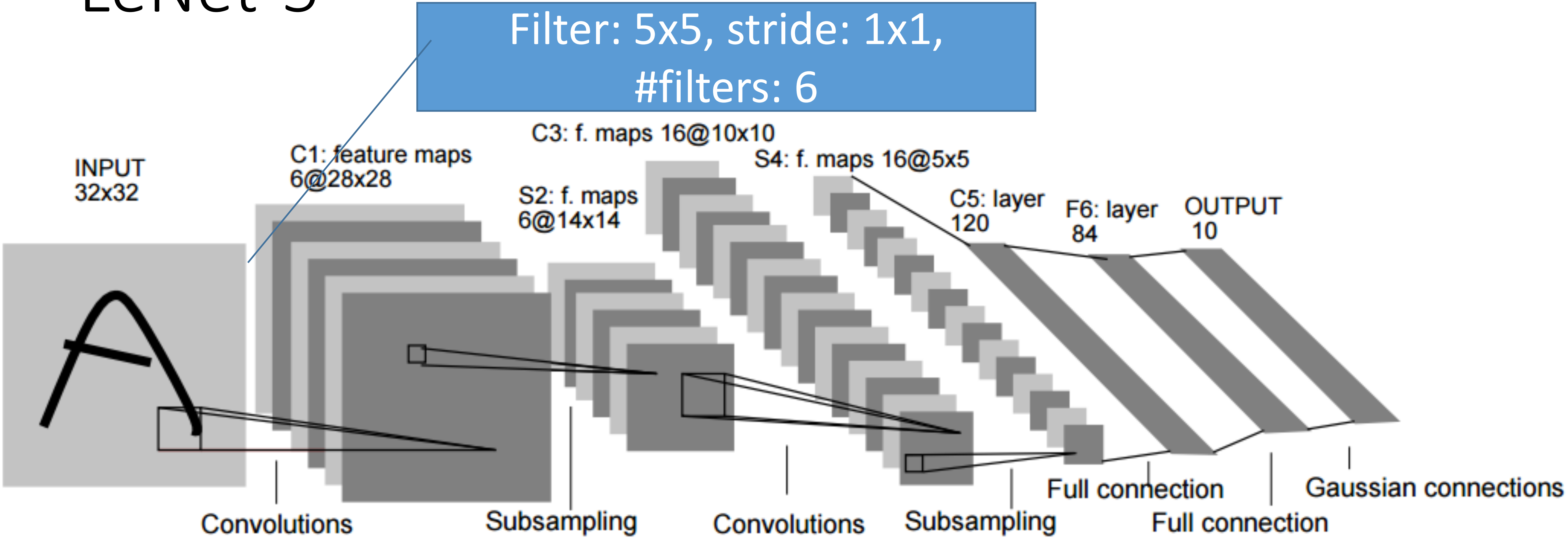


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

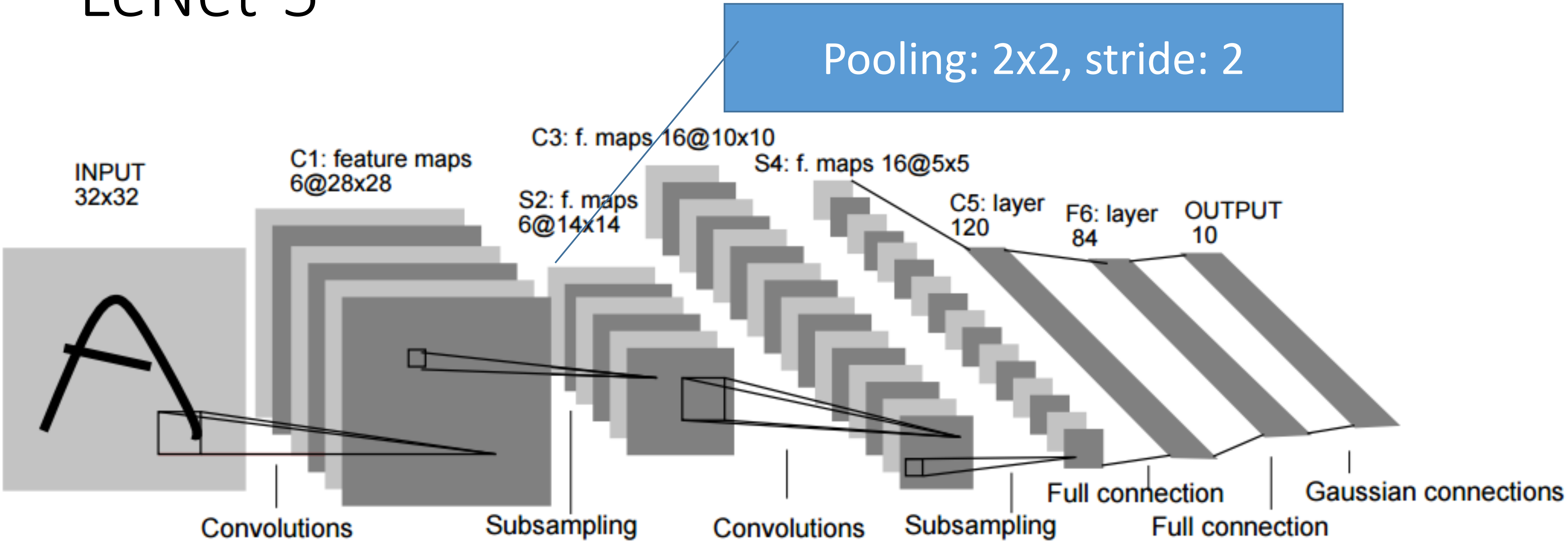


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

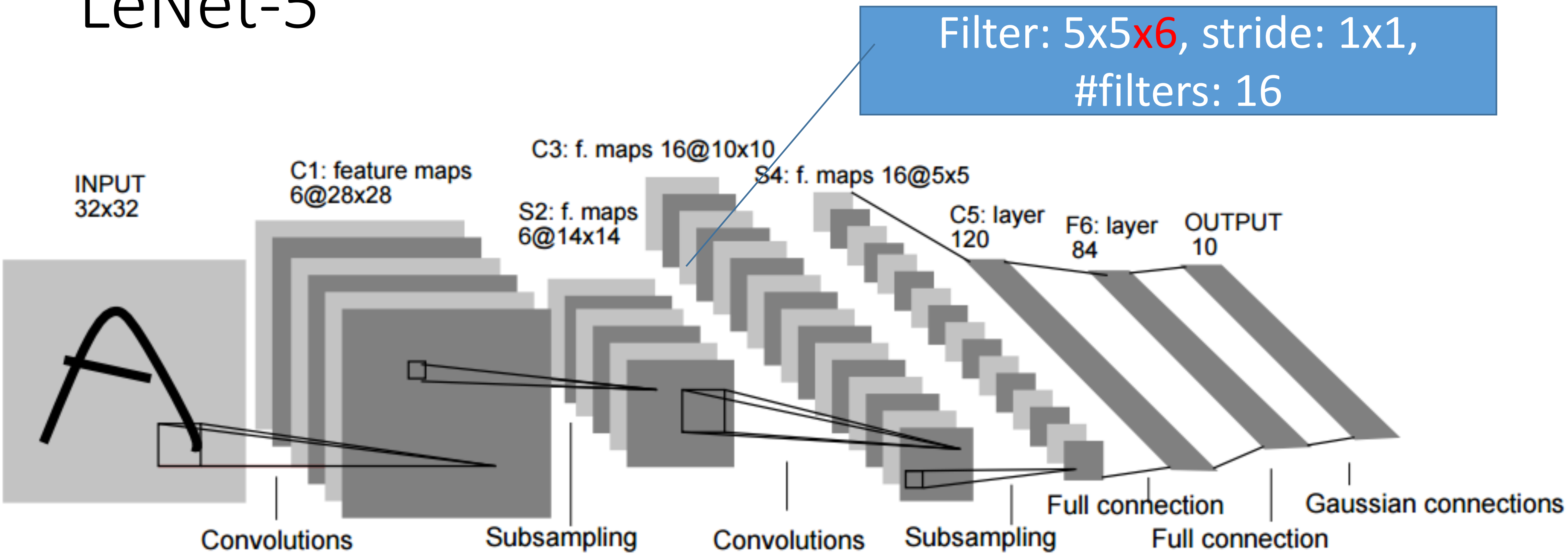


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

Pooling: 2x2, stride: 2

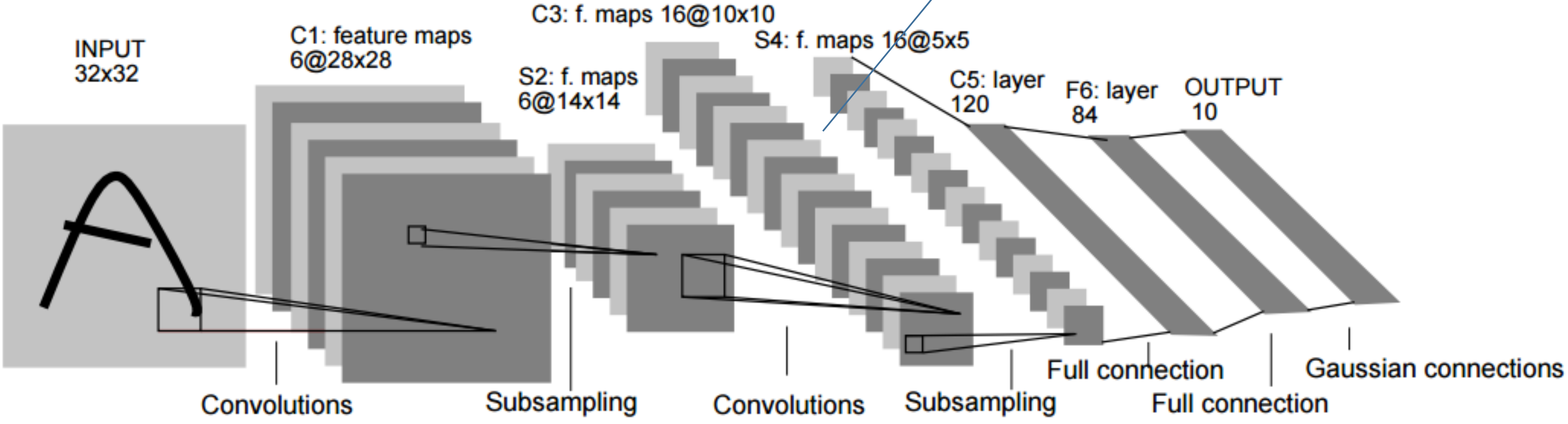


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

Weight matrix: 400x120

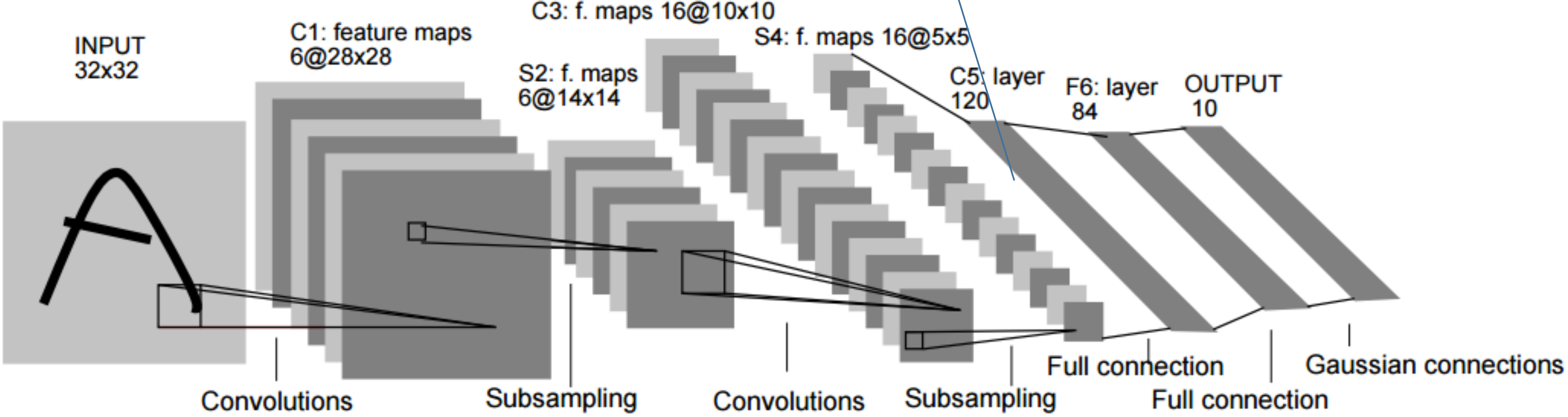


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*

LeNet-5

Weight matrix: 84x10

Weight matrix: 120x84

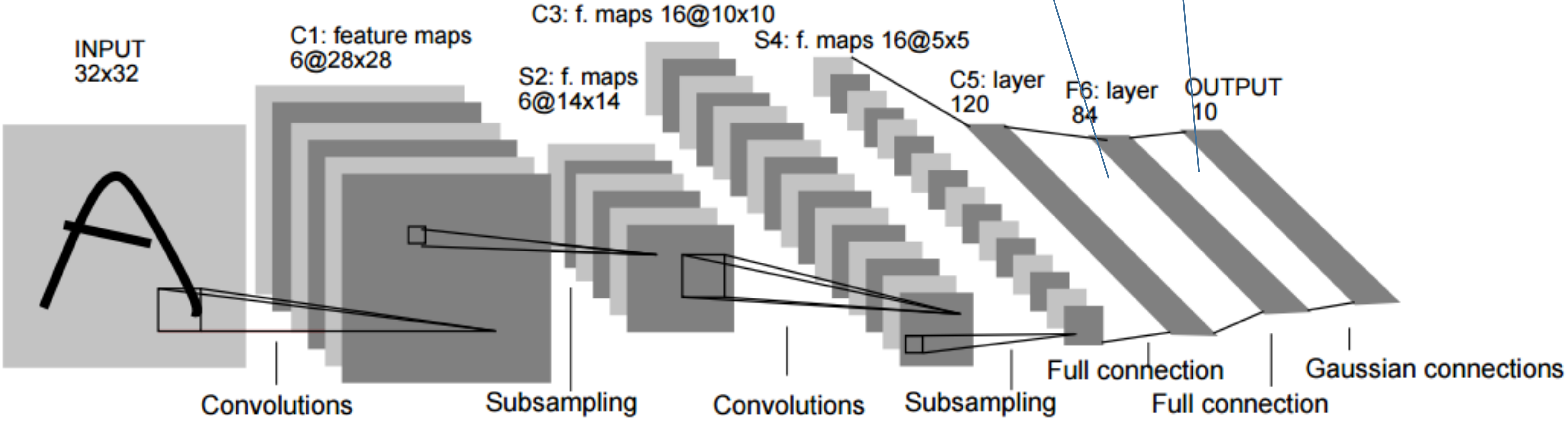


Figure from *Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner*