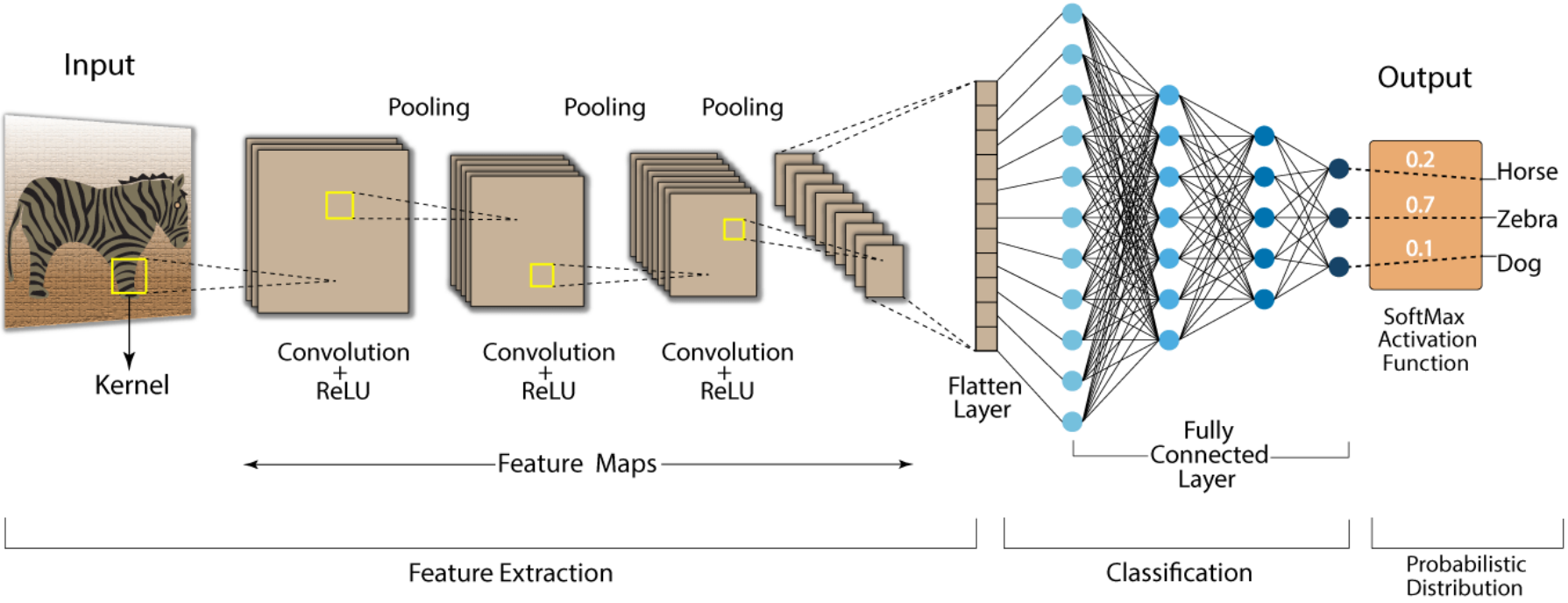
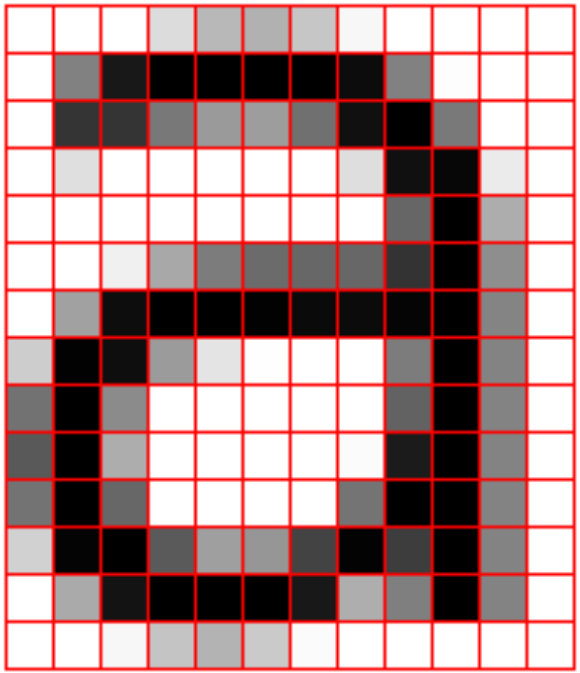
Convolution Neural Networks



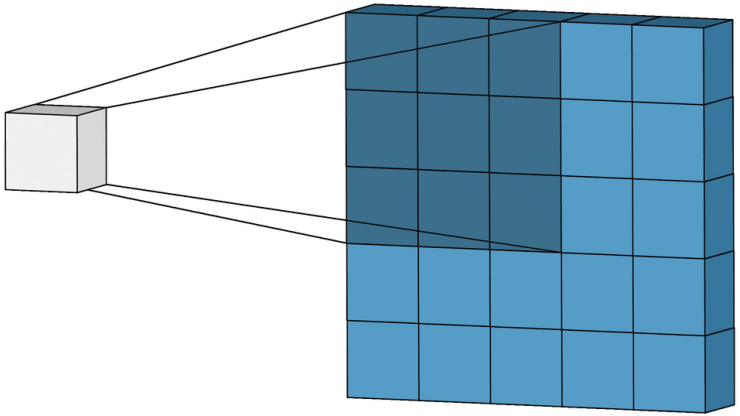
Example input data

a



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

Convolution layer



Input $I_{i,j}$

Kernel $K_{i,j}$

Activation map $$S_{i,j} = \sum_{n,m} I_{n,m} K_{i-n,j-m}$$

Example

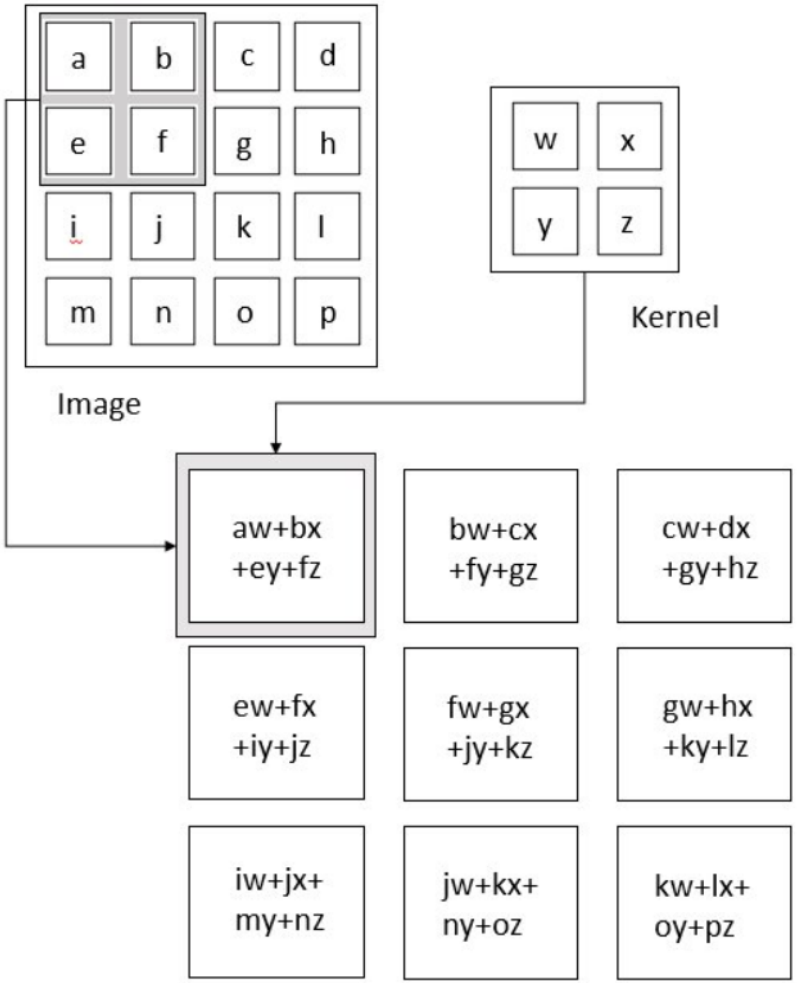


Image: w x w

Kernel: f x f

Stride: q Padding: g

Activation: a x a

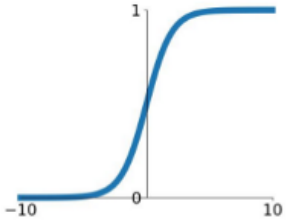
$$a = (w - f + 2g)/q + 1$$

Nonlinearity

Activation Functions

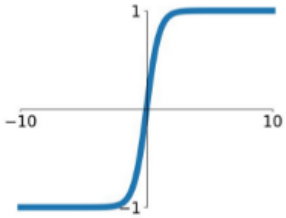
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



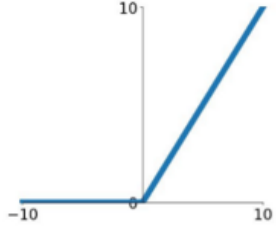
tanh

$$\tanh(x)$$



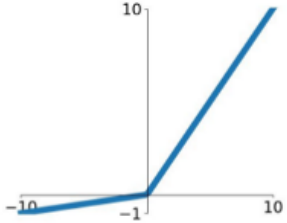
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

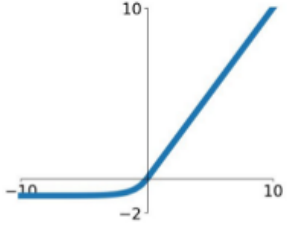


Maxout

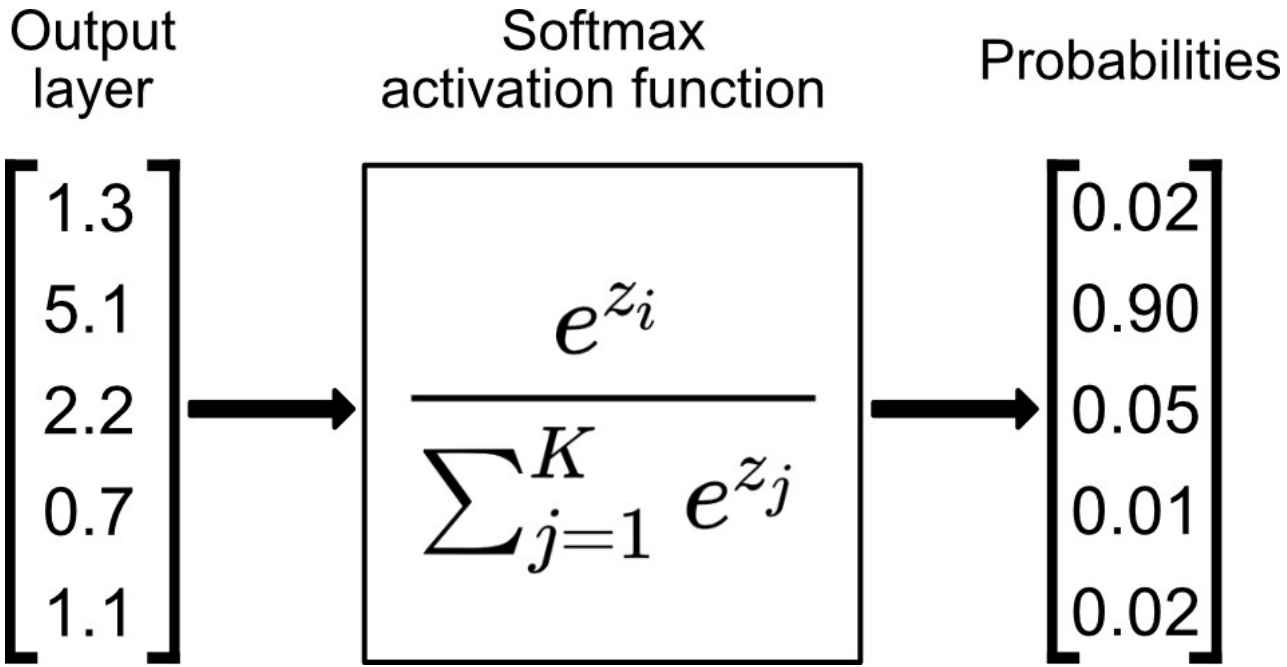
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Softmax


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

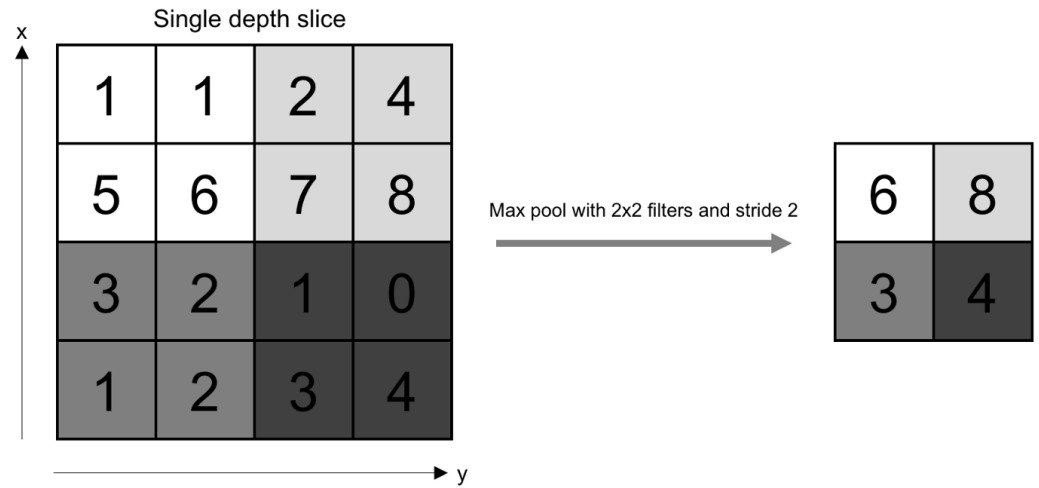
Pooling layer

Activation: a x a

Pooling kernel: p x p

Stride: q

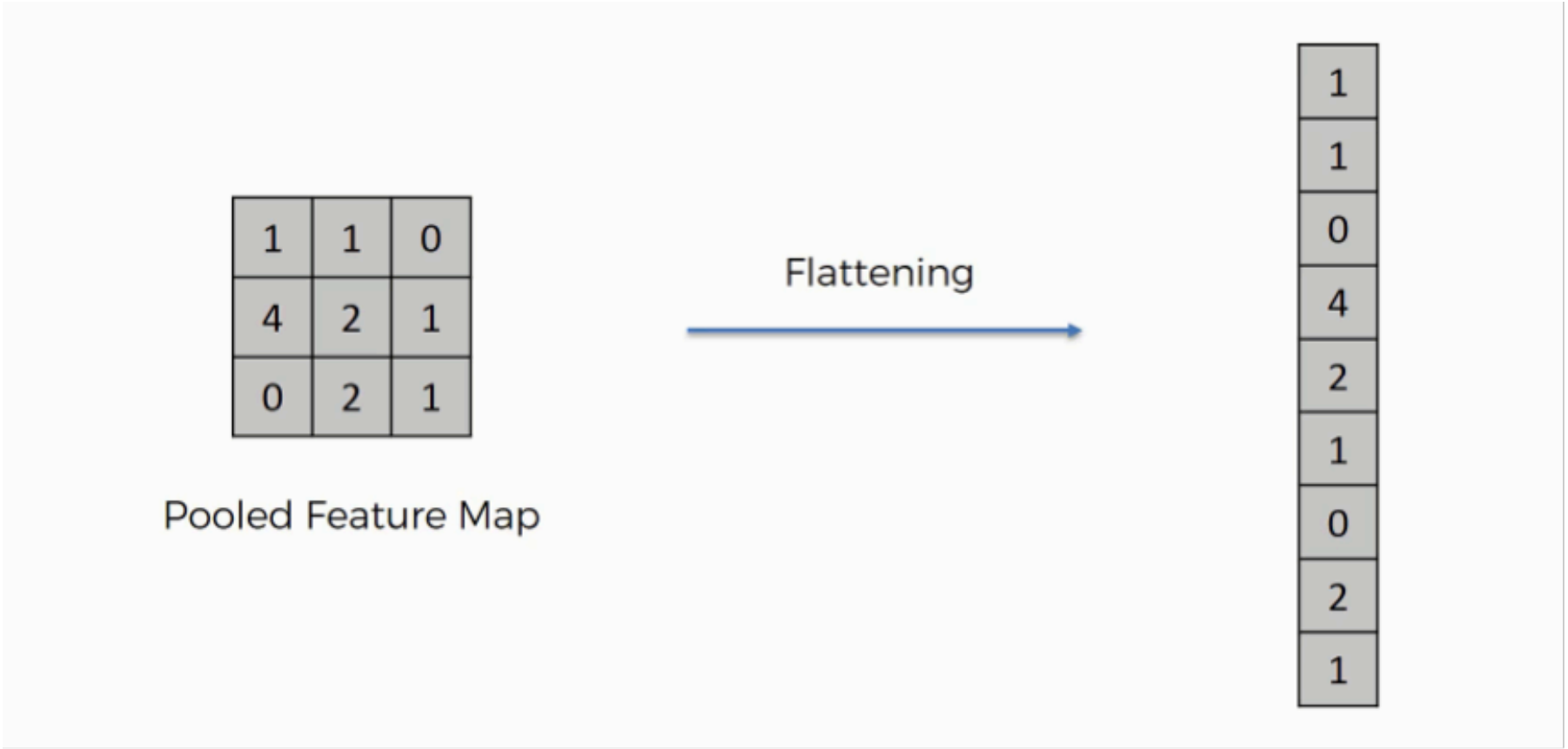
Output: b x b



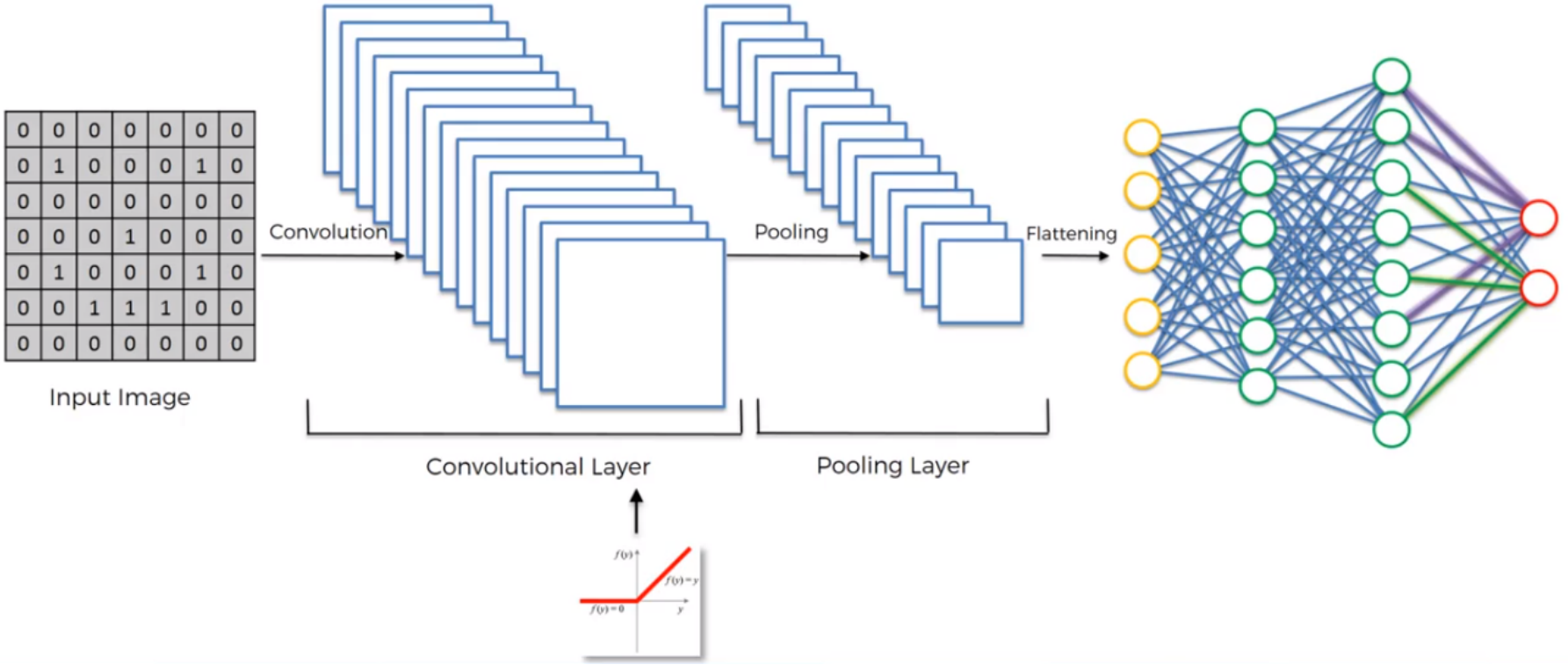
Example: maxpool

$$b = (a - p)/q + 1$$

Flattening layer



In summary


0 0 0 0 0 0 0
0 1 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 1 0
0 0 1 1 1 0 0
0 0 0 0 0 0 0
Input Image
Convolution
Pooling
Flattening
Convolutional Layer
Pooling Layer
f(y)
f(y)=y
f(y)=0
y

Example ENSO prediction (2019)

LETTER

https://doi.org/10.1038/s41586-019-1559-7

Deep learning for multi-year ENSO forecasts

Yoo-Geun Ham[1]*, Jeong-Hwan Kim[1] & Jing-Jia Luo[2,3]

Example (2019)

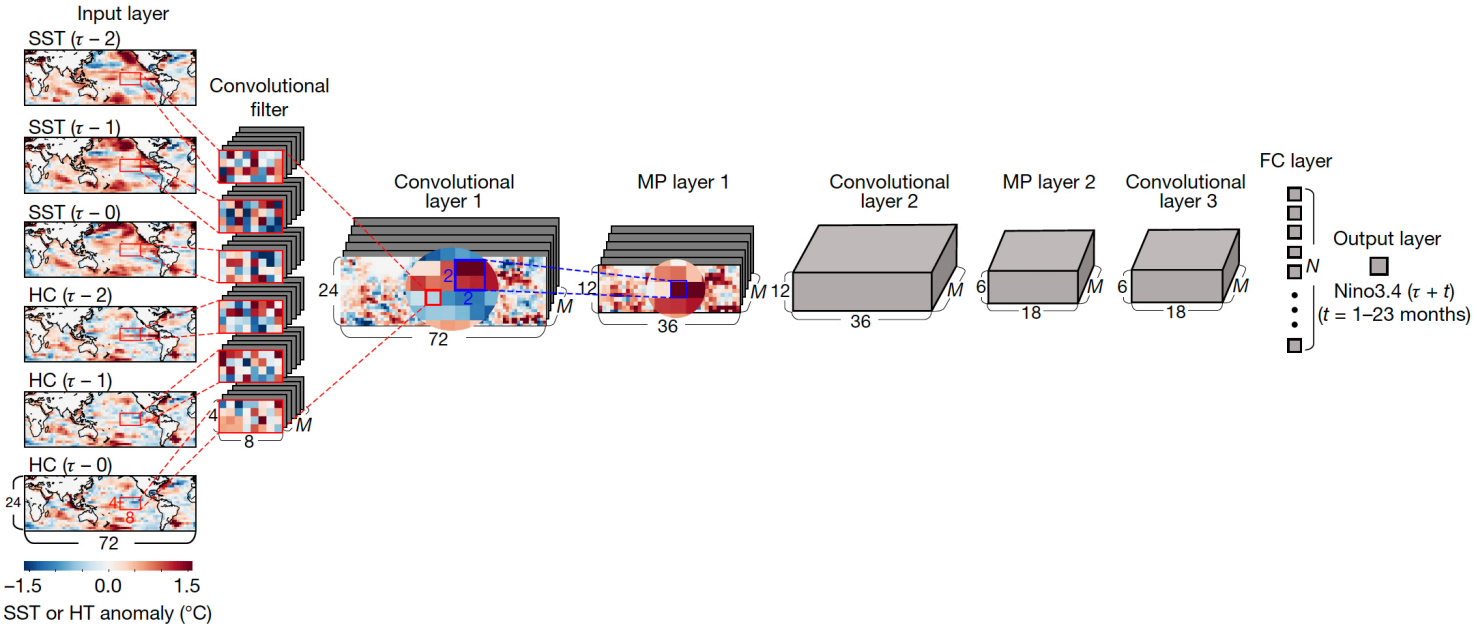
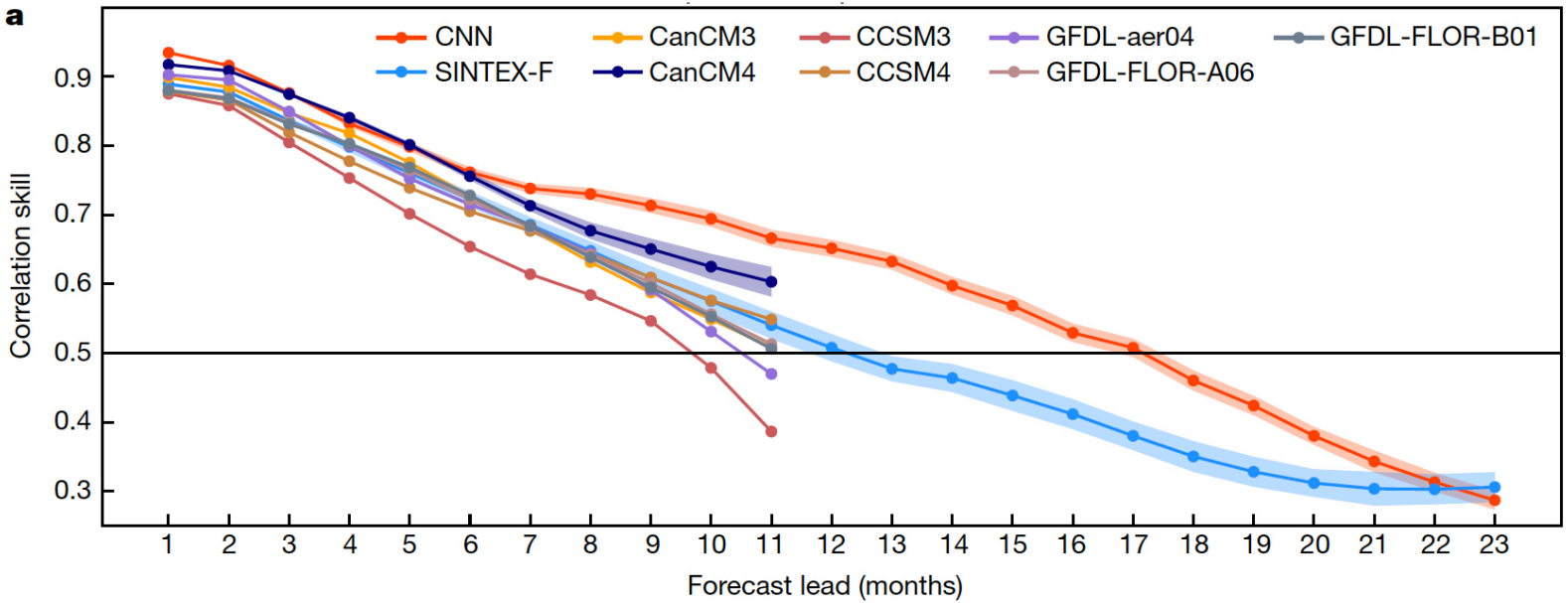
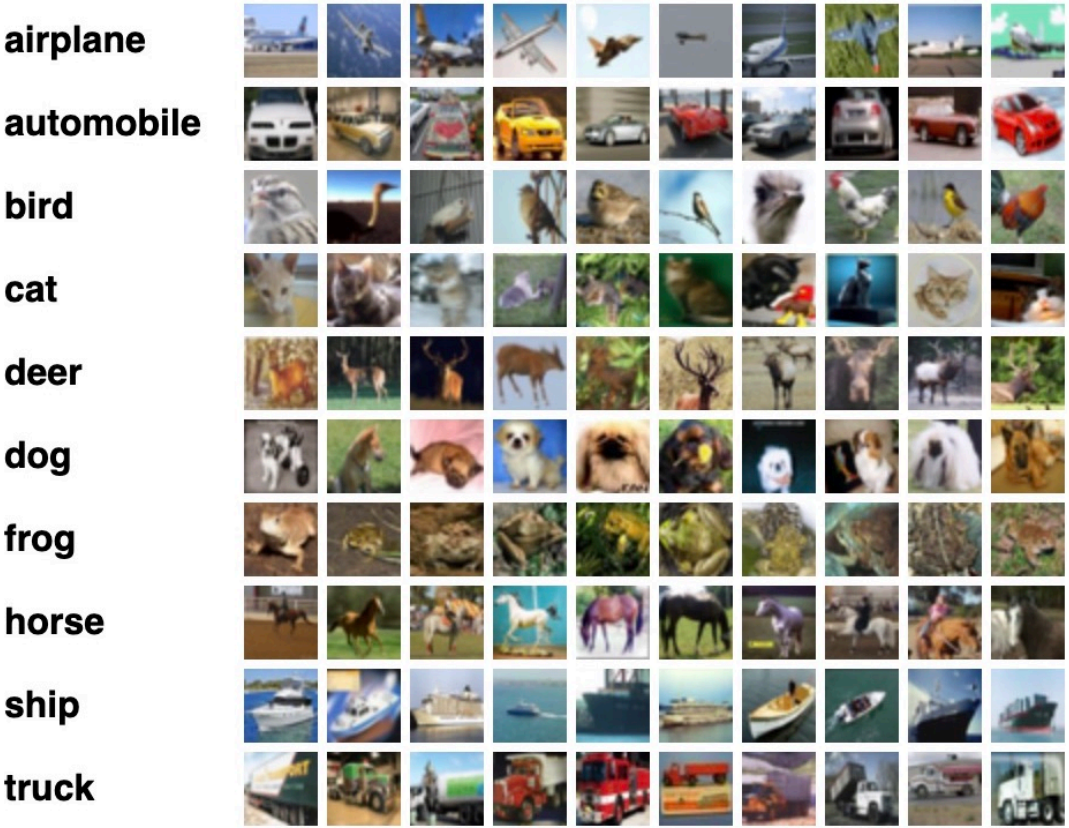


Fig. 1 | Architecture of the CNN model used for the ENSO forecasts. The CNN model consists of one input layer (the predictor), three convolutional layers, two max-pooling (MP) layers, one fully connected (FC) layer, and one output layer (the predictand). The variables of the input layer correspond to the SST (in units of °C) and the oceanic heat content (HC, in units of °C) anomaly maps from time $\tau - 2$ months to time τ (in months), between 0°–360° E and 55° S–60° N. The three-month-averaged Nino3.4 index from time $\tau + 1$ month to $\tau + 23$ months is used as a variable for the output layer. The red boxes and lines highlight the convolutional filter and convolutional process, respectively; and the blue box and lines indicate the max-pooling window and the max-pooling process, respectively. *M* denotes the number of feature maps, while *N* denotes the number of neurons in the FC layer, which are set to be either 30 or 50 in this study. The global map is generated in Matplotlib[31]. The *x* and *y* dimensions of the map in each layer are denoted (6, 12, 18, 24, 36, 72) next to the map.

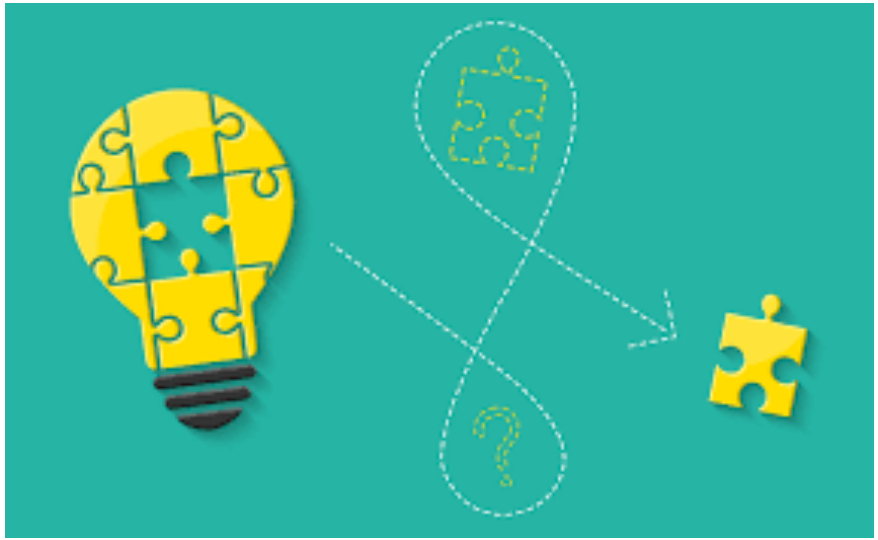
Example (2019)



CIFAR10 dataset



Exercise B6



LINK

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/cnn.ipynb#scrollTo=0LvwaKhtUdOo