ErSE222: Machine learning in Geoscience

March. 2nd, 2025

Tariq Alkhalifah and Omar Saad

SGD with momentum

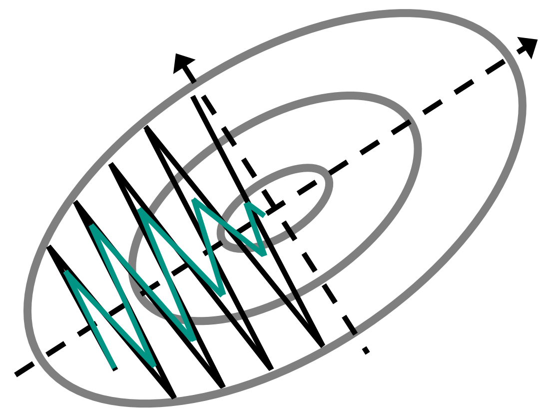
Idea of momentum (from Polyak and Nesterov in 60'): use update that is an exponentially decaying moving average of the past gradients.

Velocity

Momentum term [0,1)

$$\mathbf{v}_{i+1} = \gamma \mathbf{v}_i - \mathbf{g}_{i+1} = \gamma \mathbf{v}_i - \frac{\alpha}{N_b} \sum_{j=1}^{N_b} \nabla \mathscr{L}_j$$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \mathbf{v}_{i+1}$$



AdaGrad

Idea of AgaGrad : scale gradients by inverse of square root of sum of historical squares gradients

$$\mathbf{g}_{i+1} = \frac{1}{N_b}\sum_{j=1}^{N_b} \nabla \mathscr{L}_j$$

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{g}_{i+1} \cdot \mathbf{g}_{i+1}$$

$$\Delta\boldsymbol{\theta}_{i+1} = -\frac{\alpha}{\delta + \sqrt{\mathbf{r}_{i+1}}} \cdot \mathbf{g}_{i+1}$$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_{i+1}$$

Parameter dependent rescaling

RMSProp

Idea of RMSProp : gradient accumulation is *exponentially damped*

$$\mathbf{g}_{i+1} = \frac{1}{N_b} \sum_{j=1}^{N_b} \nabla \mathscr{L}_j$$

$$\boxed{\mathbf{r}_{i+1} = \rho \mathbf{r}_i + (1-\rho) \mathbf{g}_{i+1} \cdot \mathbf{g}_{i+1}}$$

$$\Delta \boldsymbol{\theta}_{i+1} = -\frac{\alpha}{\delta + \sqrt{\mathbf{r}_{i+1}}} \cdot \mathbf{g}_{i+1}$$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta \boldsymbol{\theta}_{i+1}$$

Slower decay of LR

Parameter dependent rescaling

Adam = Momentum + RMSProp

$$\mathbf{g}_{i+1} = \frac{1}{N_b}\sum_{j=1}^{N_b} \nabla \mathscr{L}_j$$

$$\mathbf{m}_{i+1} = \rho_1 \mathbf{m}_i + (1-\rho_1)\mathbf{g}_{i+1} \leftarrow \textit{velocity term}$$

$$\mathbf{v}_{i+1} = \rho_2 \mathbf{v}_i + (1-\rho_2)\mathbf{g}_{i+1} \cdot \mathbf{g}_{i+1} \leftarrow \textit{scaling term}$$

$$\hat{\mathbf{m}}_{i+1} = \frac{\mathbf{m}_{i+1}}{1-\rho_1^{i+1}} \leftarrow \textit{bias correction}$$

$$\hat{\mathbf{v}}_{i+1} = \frac{\mathbf{v}_{i+1}}{1-\rho_2^{i+1}} \leftarrow \textit{bias correction}$$

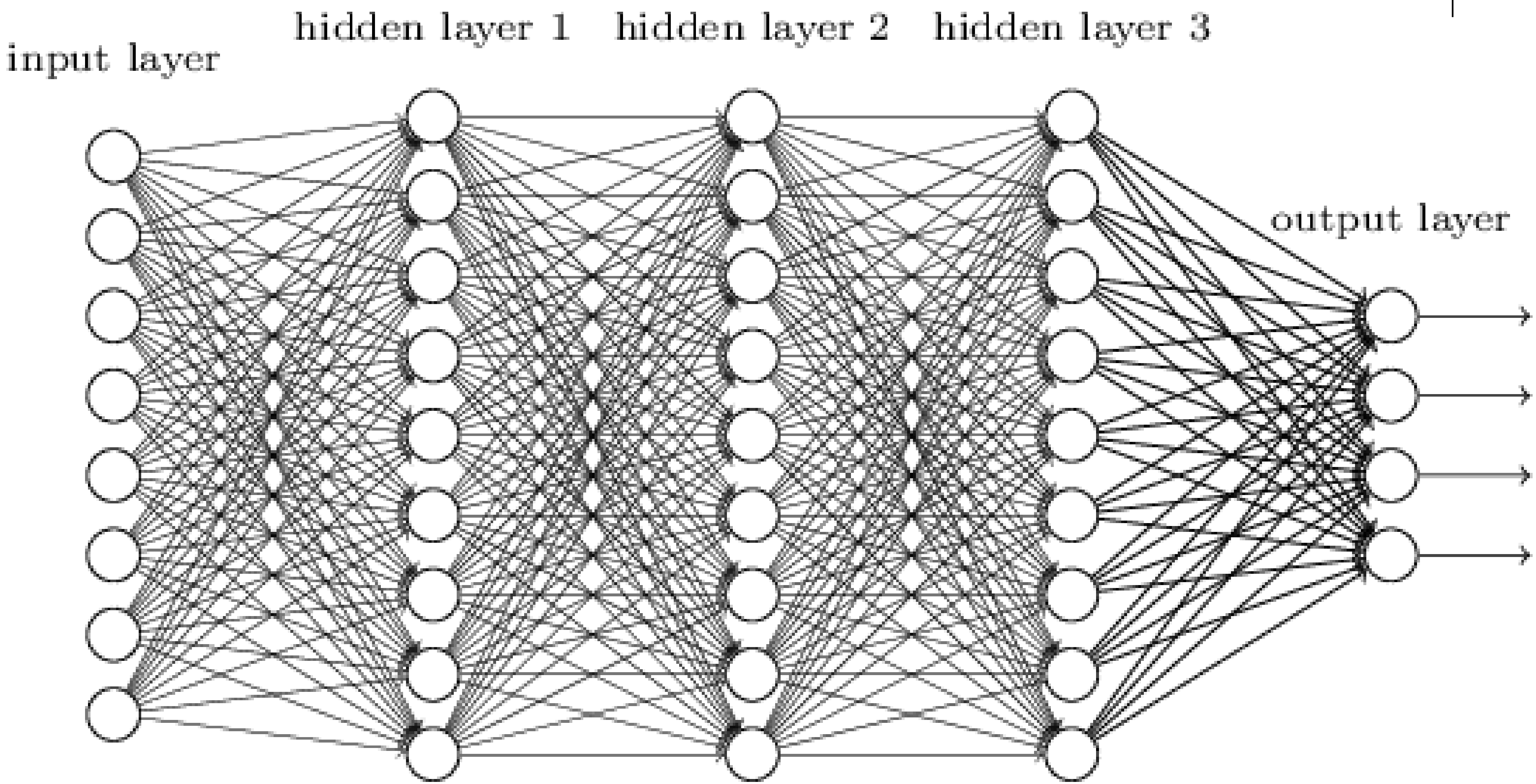
$$\Delta\boldsymbol{\theta}_{i+1} = -\frac{\alpha}{\delta + \sqrt{\hat{\mathbf{v}}_{i+1}}} \cdot \hat{\mathbf{m}}_{i+1}$$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \Delta\boldsymbol{\theta}_{i+1}$$

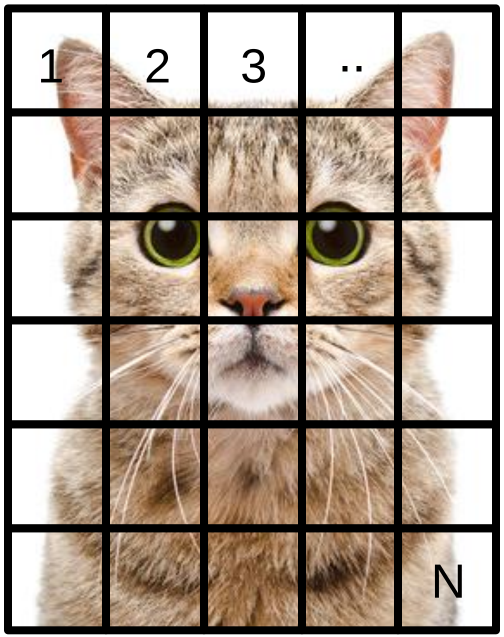
$\delta = 10^{-6}$, and two decay rates (ρ_1 and ρ_2).

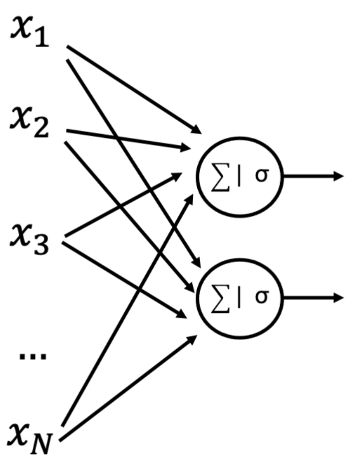
CNN

Neural Network



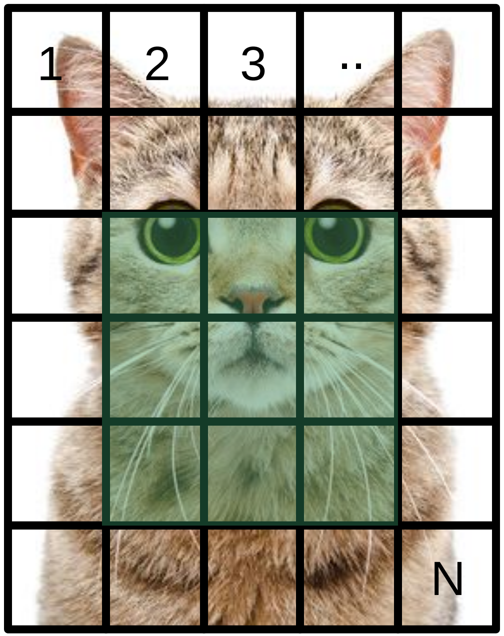
Convolutional Neural Networks (CNN)

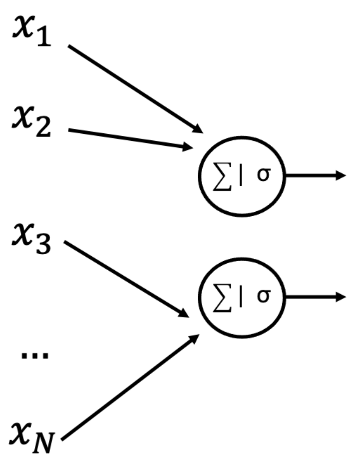




$$y_i = \sigma\left(\sum_{j=1}^{N} w_{ij} x_j\right)$$

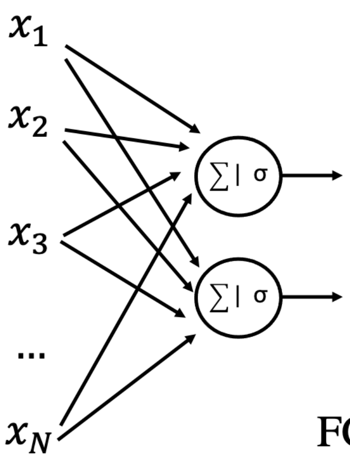
Convolutional Neural Networks (CNN)





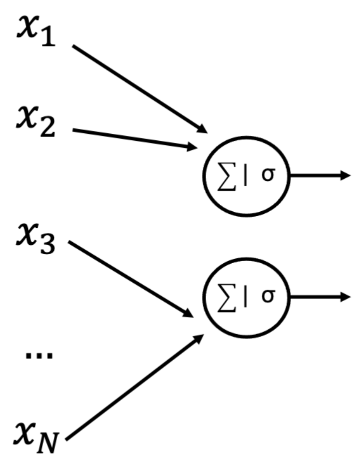
$$y_i = \sigma\left(\sum_{j=i-W}^{i+W} w_{ij} x_j\right)$$

Convolutional Neural Networks (CNN)



$$y_i = \sigma\left(\sum_{j=1}^{N} w_{ij} x_j\right)$$

FCN



$$y_i = \sigma\left(\sum_{j=i-W}^{i+W} w_{ij} x_j\right)$$

CNN

Convolution

Mathematical operation on two functions (x, h) that keeps track of the running contributions of the product of these two functions

$$y(t) = \int x(\tau)h(t-\tau)d\tau \leftrightarrow y = x * h$$

Output / Feature map

Input

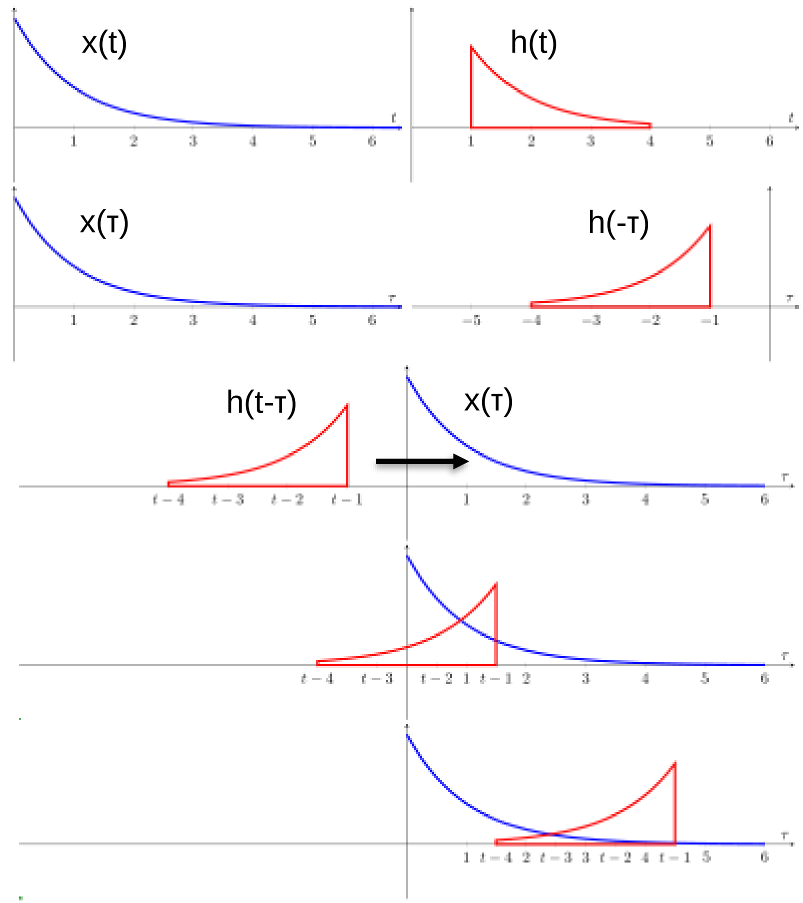
Filter / kernel

Signal Proc / DL

Widely used in signal processing, telecommunication, image processing as it describes the response of a system to an input

Convolution (visual)

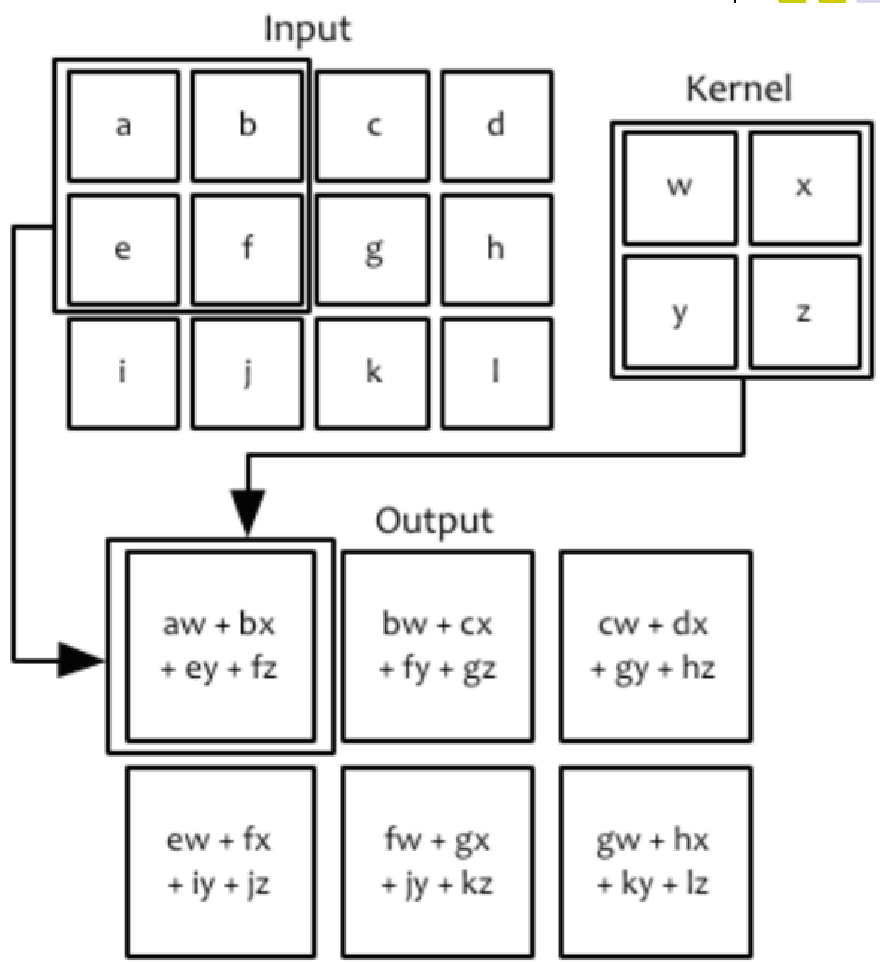
$$y(t) = \int x(\tau)h(t-\tau)d\tau \leftrightarrow y = x * h$$



Convolutional Layer

- Input: an image (2-D array):x
- Convolution kernel: W
- Feature map (2-D array of processed data): s
- Convolution operation in 2-D domains:

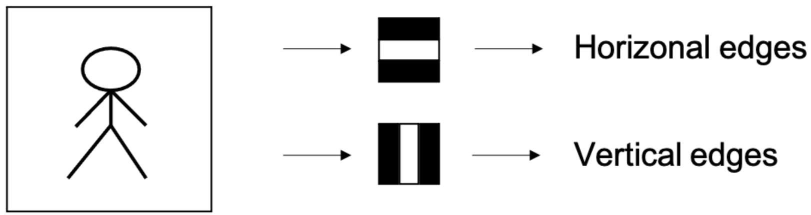
$$s[i,j] = (x * w)[i,j] = \sum_{m=-M}^{M} \sum_{n=-N}^{N} x[i+m, j+n]\, w[m,n]$$



Convolutional Layer

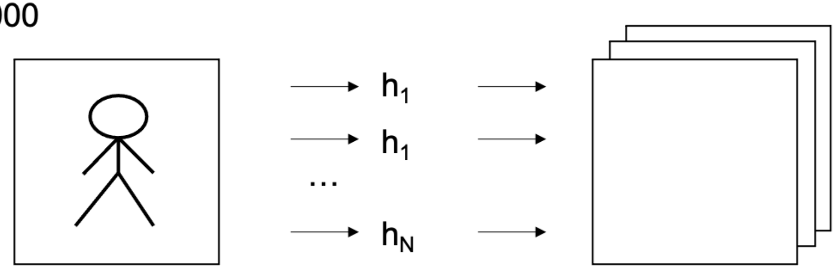
- Sparse interactions or connectivity / weights;

‘90



H filters: hand-crafted

‘2000

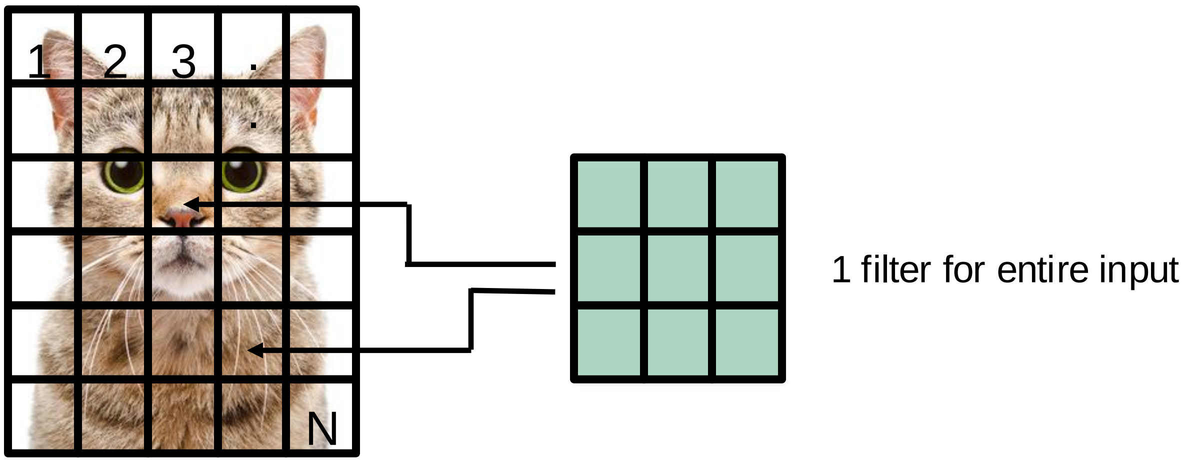


H filters: learned

Convolutional Layer

- Sparse interactions or connectivity / weights;

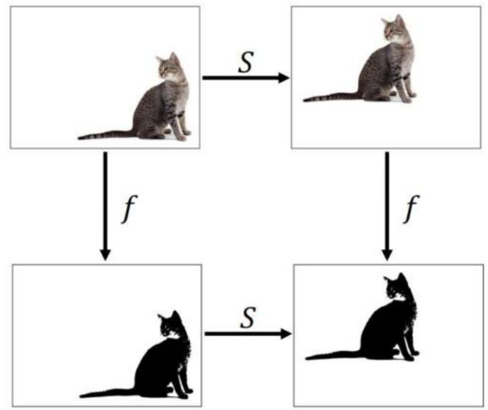
- Parameter sharing



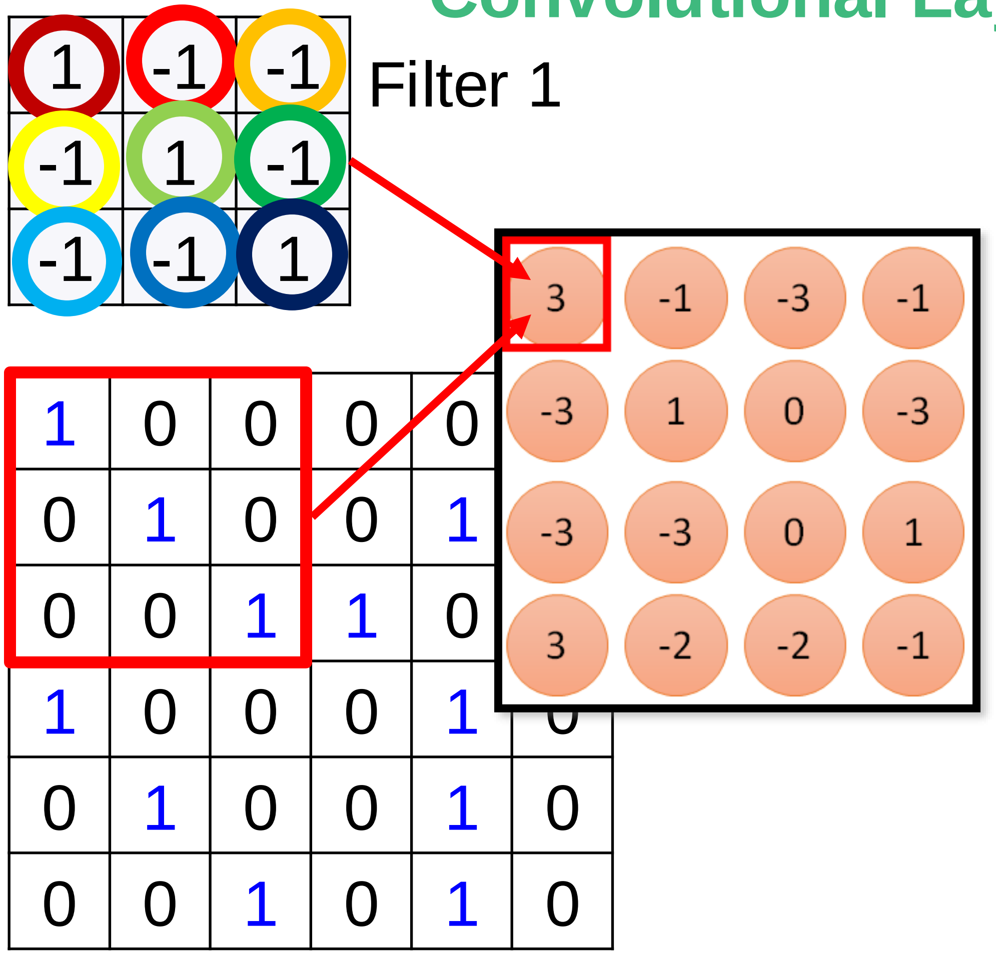
Convolutional Layer

- Sparse interactions or connectivity / weights;
- Parameter sharing
- Equivariance to translation

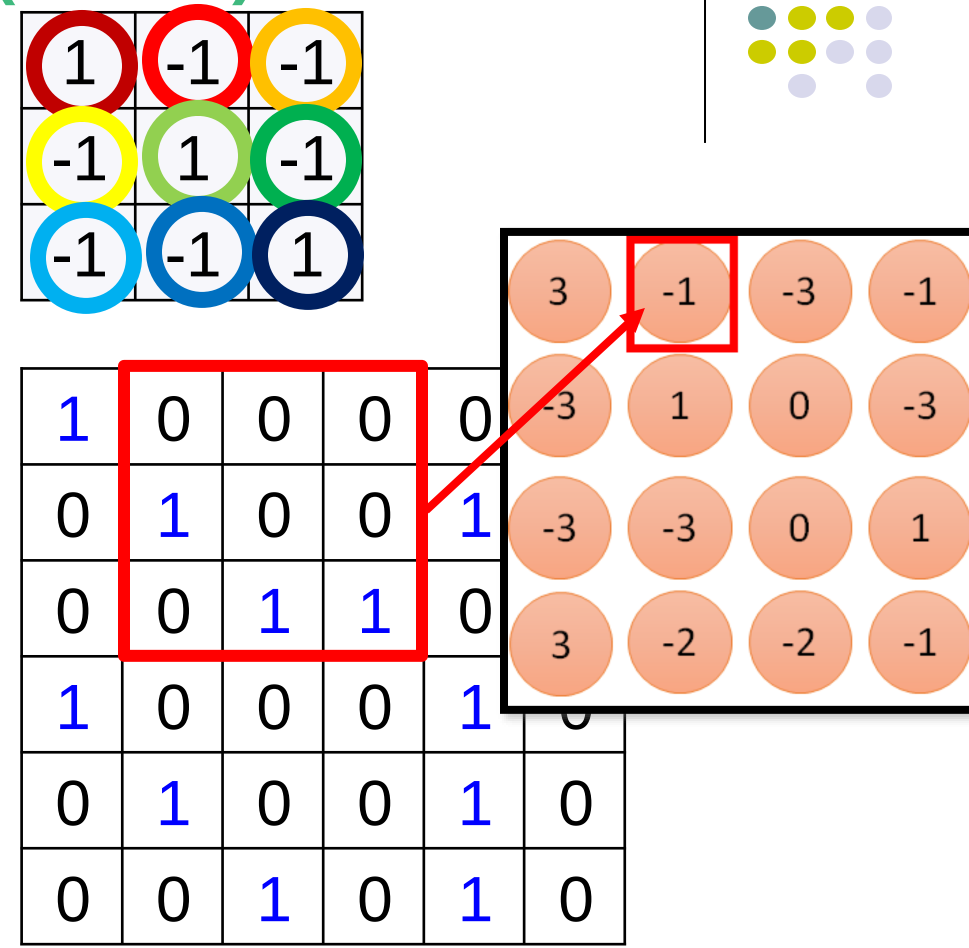
shift the input by k samples, the output will also be shifted by the same number of samples;



Convolutional Layer (Kernel)

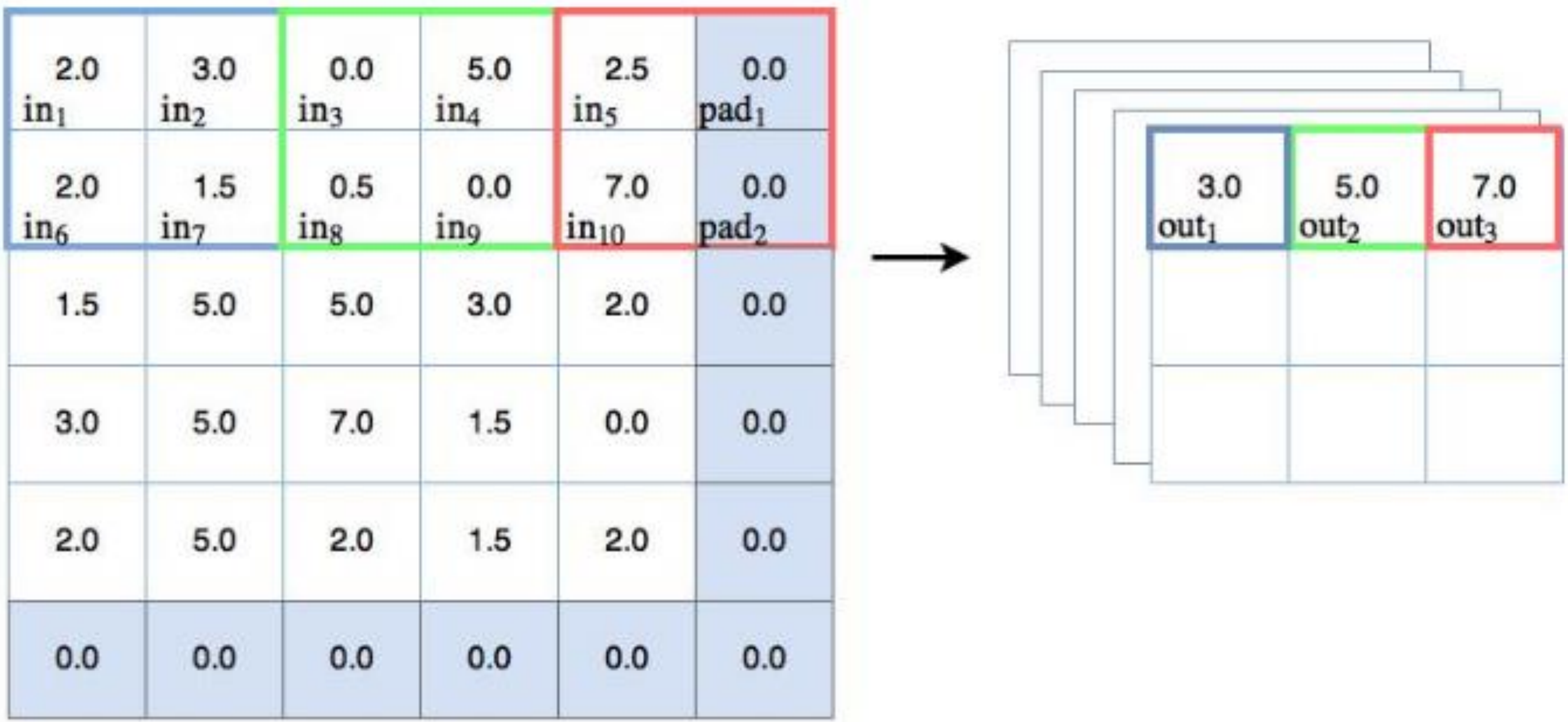


6 x 6 image



6 x 6 image

Convolutional Layer (Padding)



Convolution and Correlation outputs

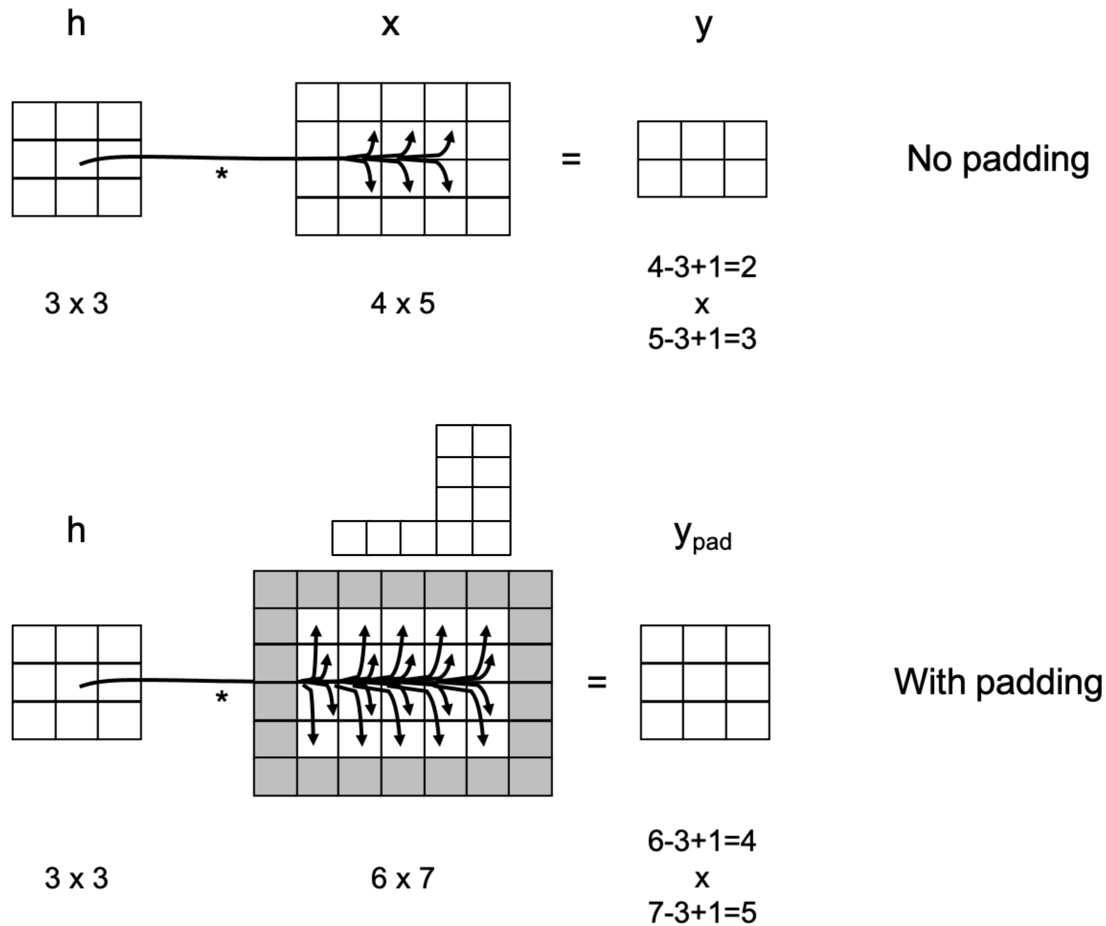
The size of the output of these two operations can be:

$$N_y = N_x - N_h + 1$$

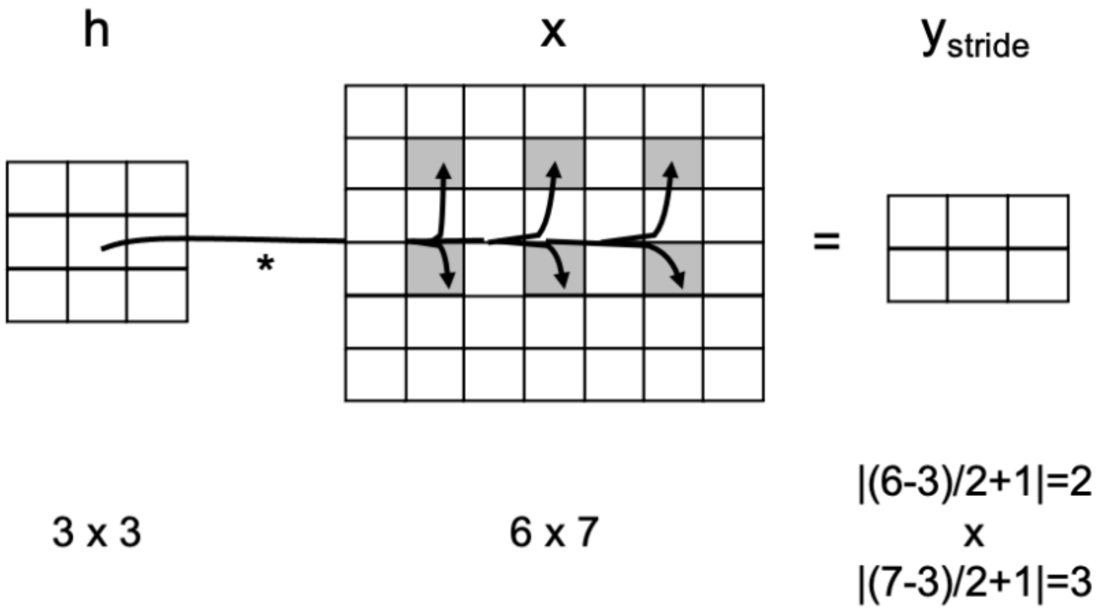
when we are interested in the valid signal
(i.e., the entire filter contributes to the output calculation)

Padding

$$N_y = N_x - N_h + 1$$



Strides



With striding

$$N_y = \lfloor (N_x - N_h)/stride + 1 \rfloor.$$

Padding and Strides

$$N_y = \left\lfloor \frac{N_x + 2pad - N_h}{stride} + 1 \right\rfloor$$

Convolutional Layer (Pooling)

- Subsampling pixels will not change the object

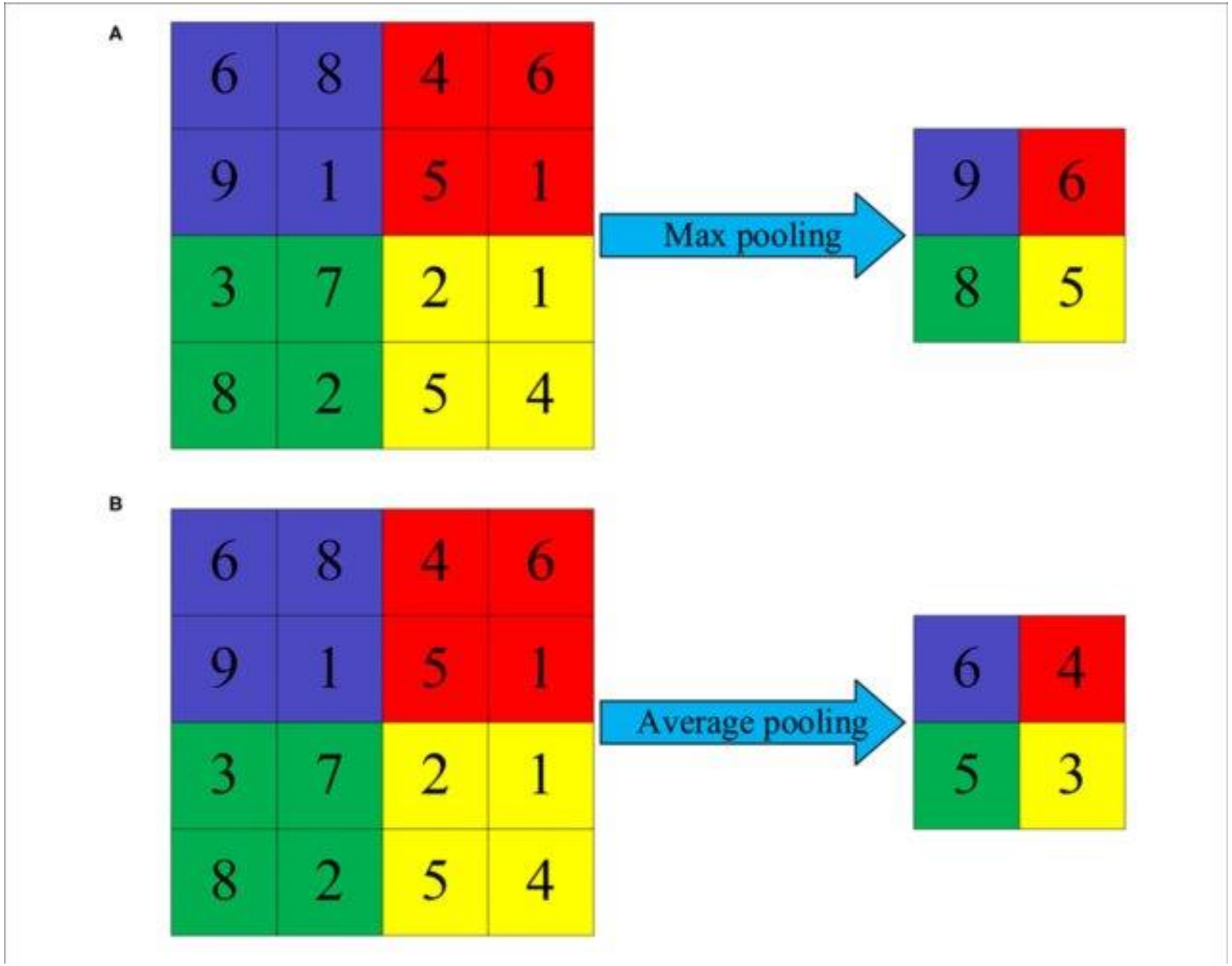
bird

Subsampling

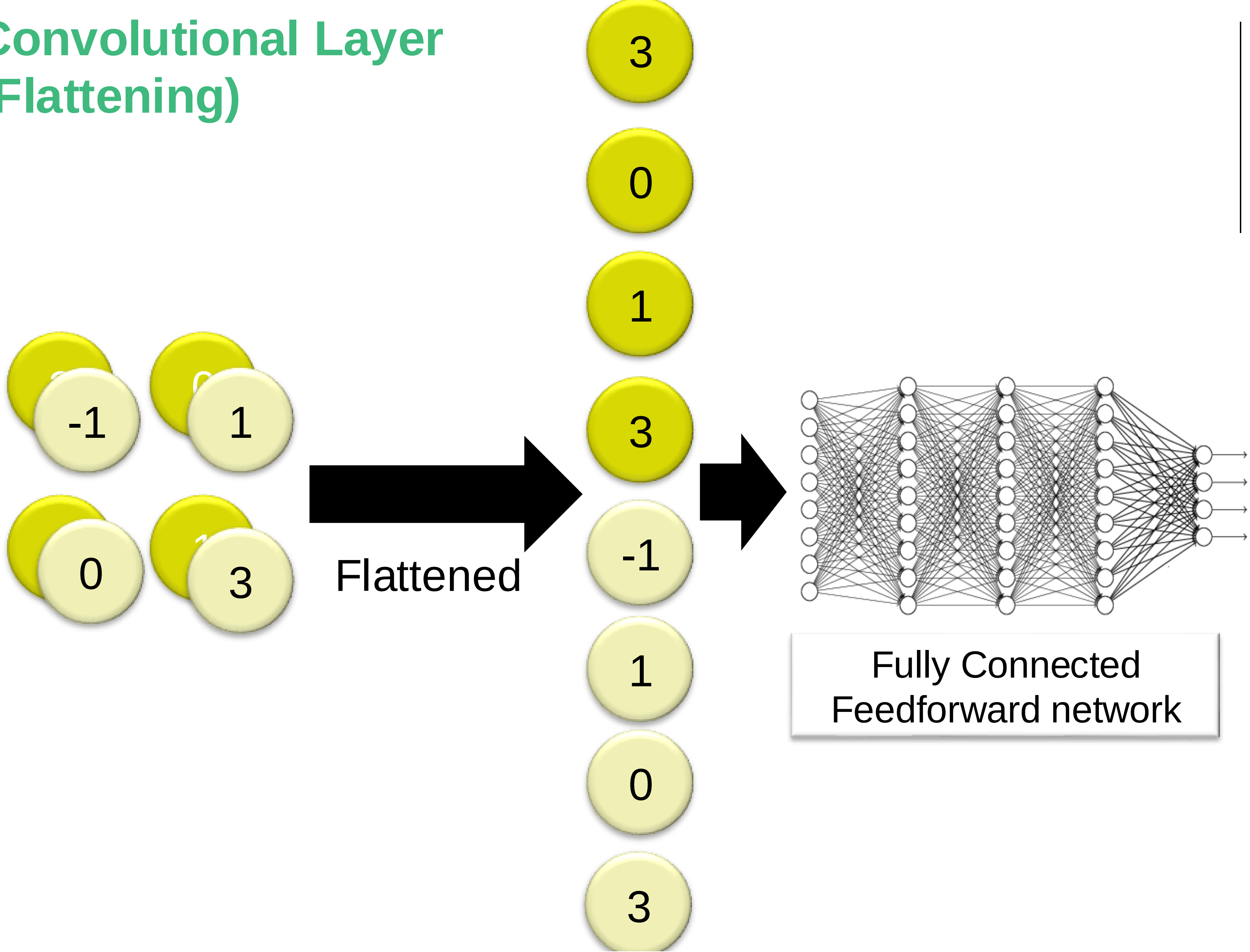
bird

We can subsample the pixels to make image smaller, fewer parameters to characterize the image

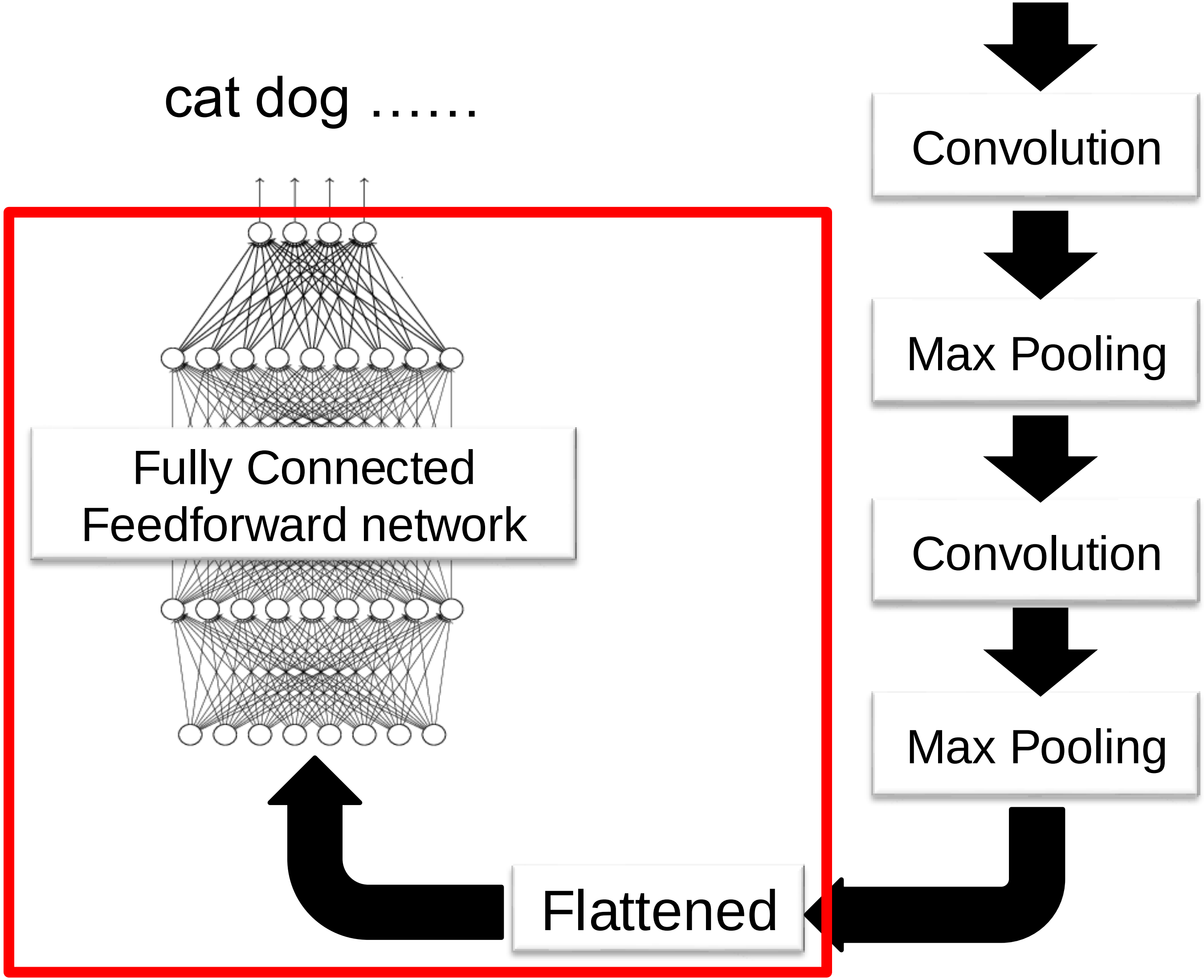
Convolutional Layer (Pooling)



Convolutional Layer (Flattening)

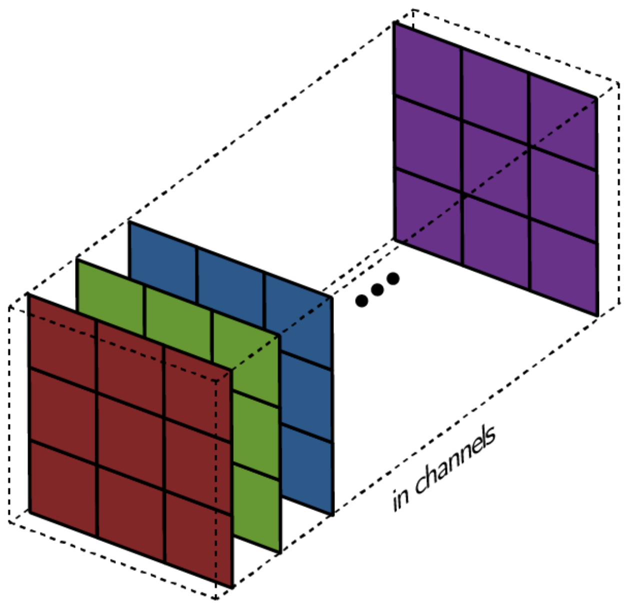


Convolutional Layer (All in one)



Channels

In deep learning, we work with multiple feature maps at the same time

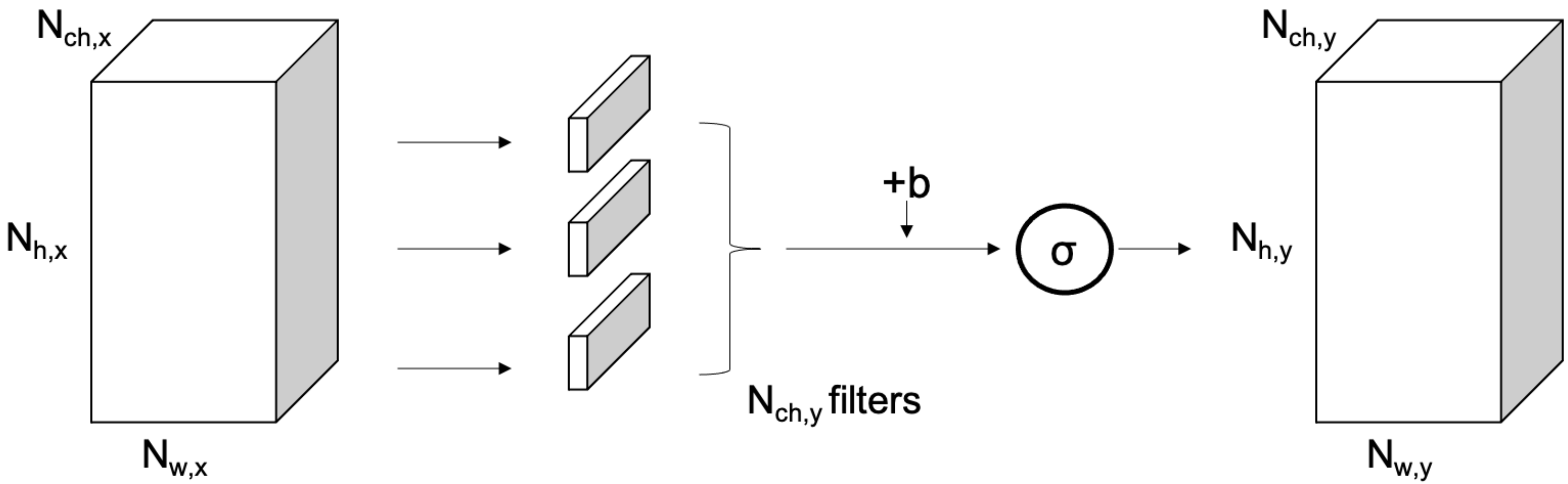


Input: can be RGB, spectral bands, horizontal/vertical motion, etc.

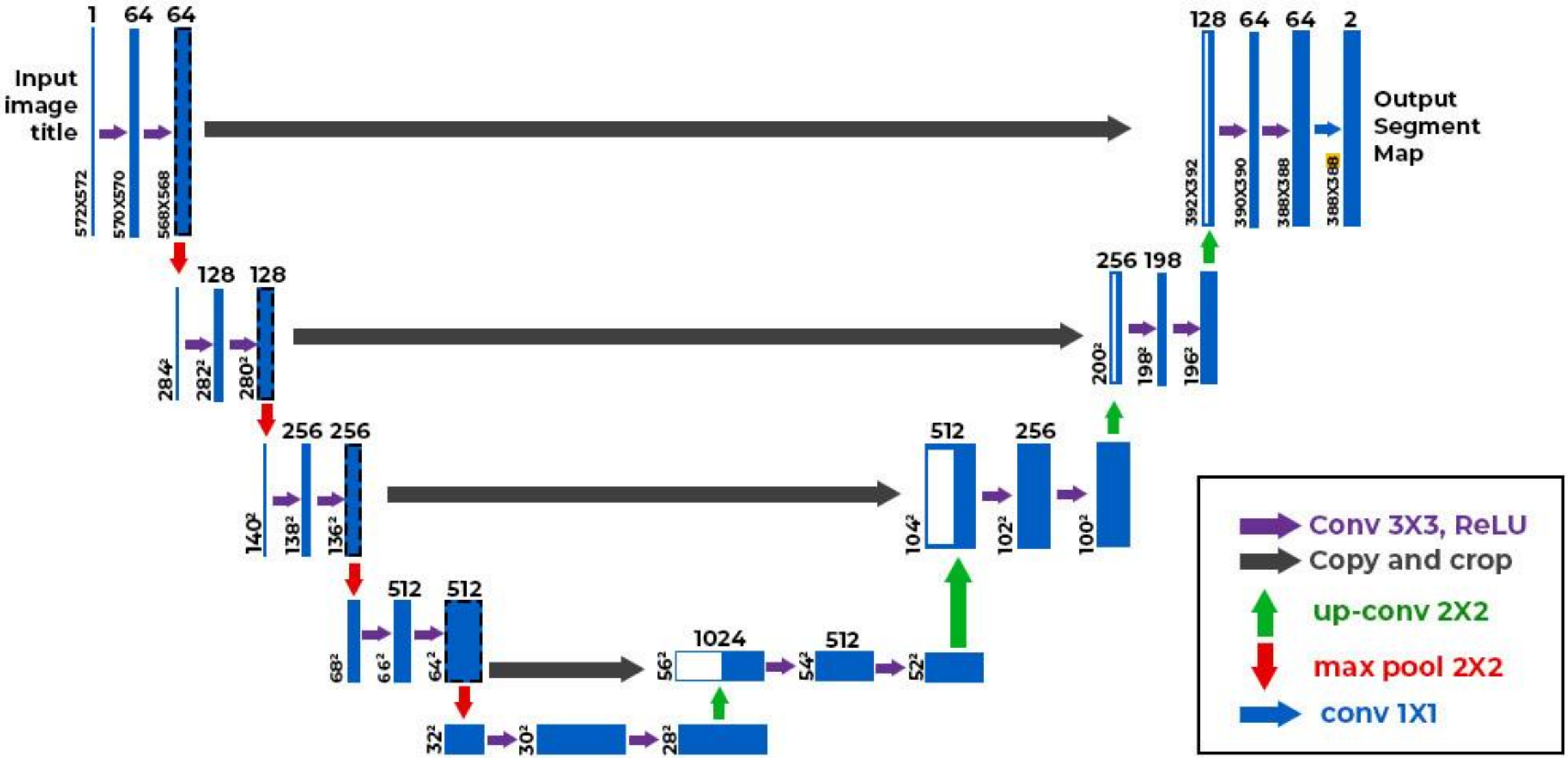
Hidden layers: produced by applying multiple filters in previous conv layet

Convolutional layer

As usual, we can apply multiple filters to produce multiple outputs (like in MLP we used to apply multiple weight vectors = weight matrix)

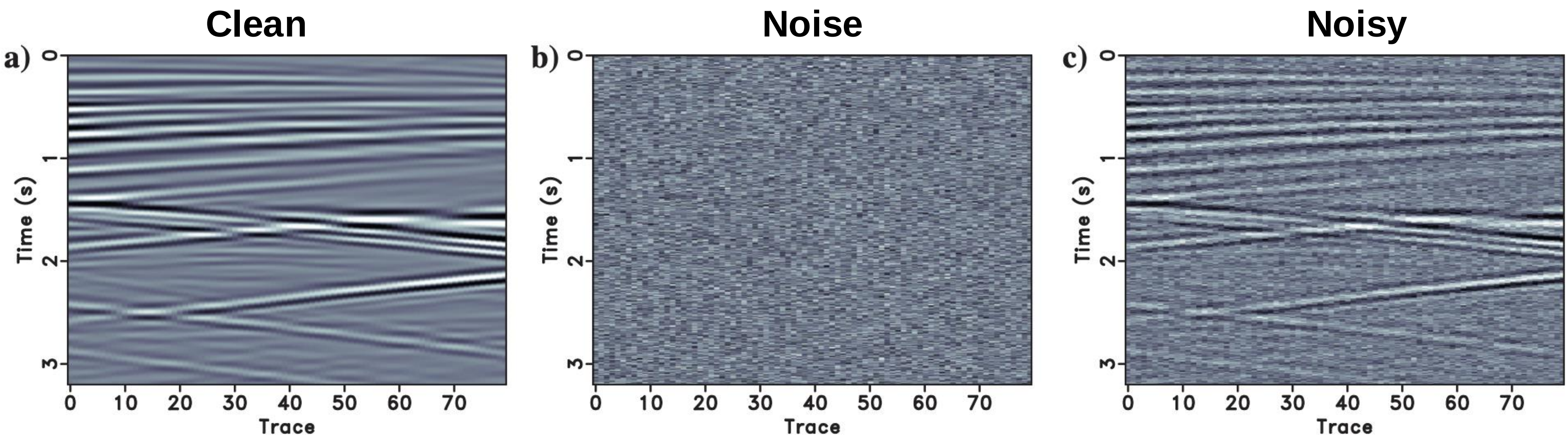


Convolutional Layer (U-NET)



Application

Seismic Data Denoising



Thank You