Activation functions

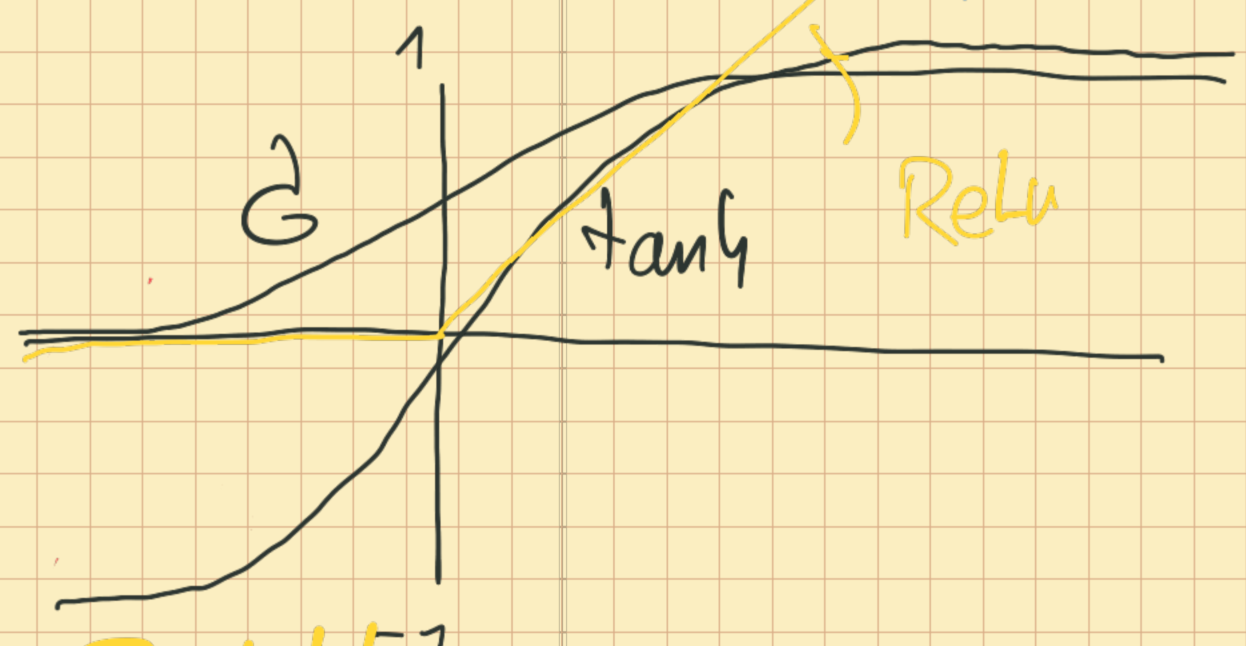
$f(x) = x$ linear : $\underline{A}(\underline{B}(\underline{C}(\underline{D}\,x) \neq \underline{G}\,x$

smooth

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

$$f(x) = \tanh$$

$$f(x) = \max(x,0) = \text{ReLU}$$



Vanishing gradients

remember backprop, where we had factors of the form: $\frac{\partial \vec{\sigma}(z)}{\partial z}$

For tanh, $\vec{\sigma}$: $\vec{\sigma}(z) \longrightarrow 1$ for $z \to \infty$

$\Rightarrow \vec{\sigma}'(z) \longrightarrow 0$ — " —

→ the error term becomes small: exponentially so in earlier layers in the net: stop of training!

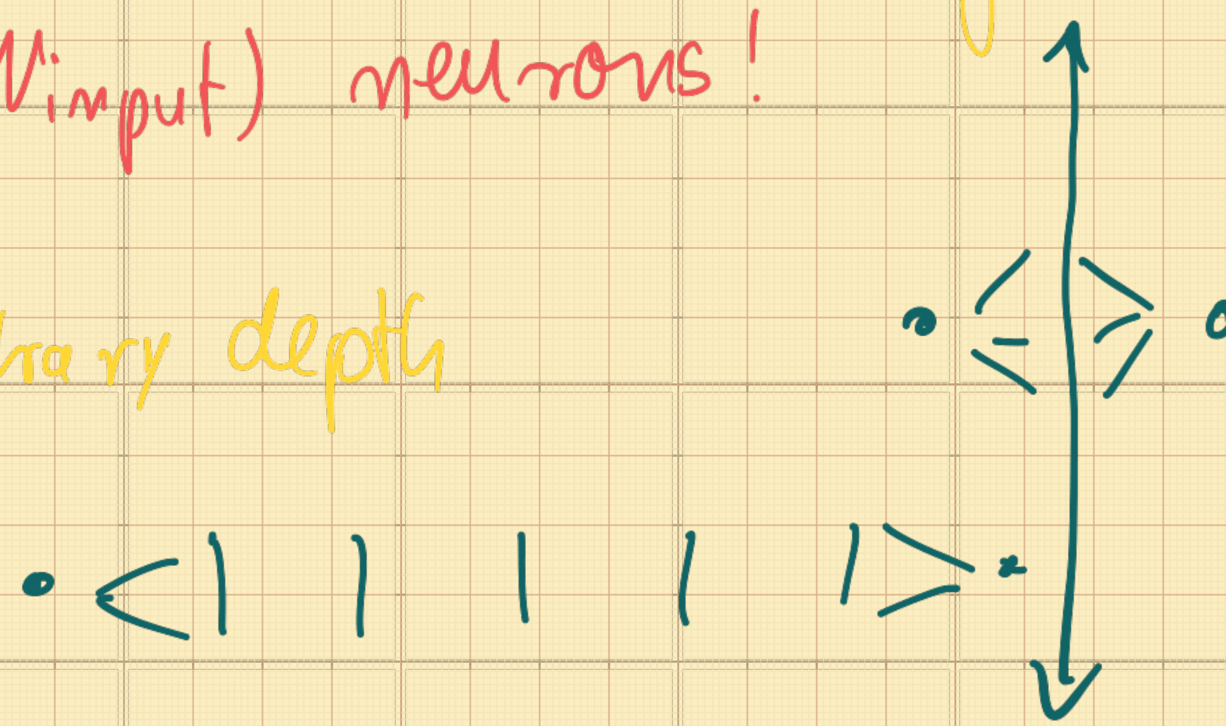
→ use other activations

→ use Resnets with skip connectors!

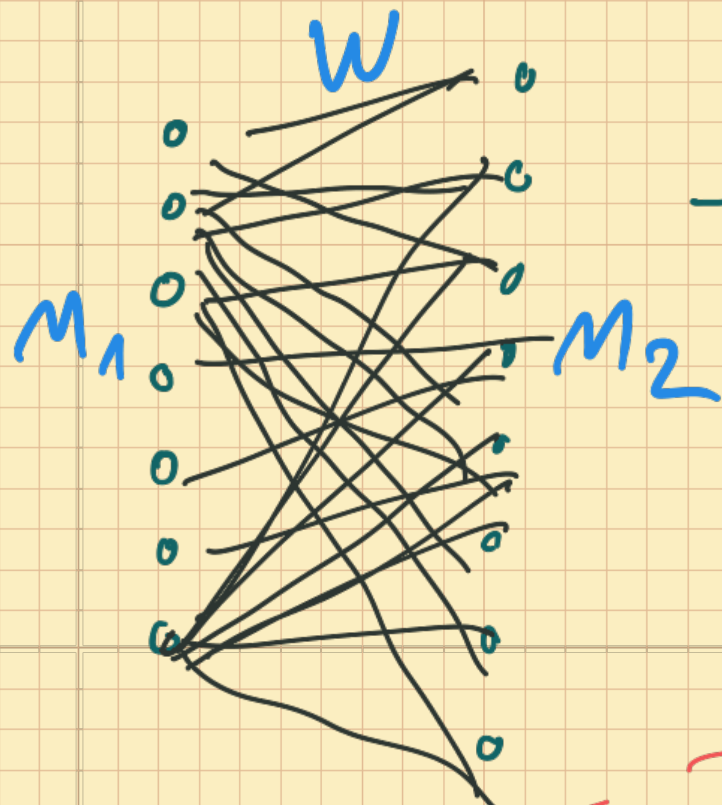
Universal approximators:

A ANN with a single hidden layer can exactly represent any continuous function of multiple inputs:

- Cybenko, Hornik 1989: smooth activation function
  → might need $\exp(N_{input})$ neurons!
- Leshno 1993: ReLU
- Lu et al 2017: arbitrary depth

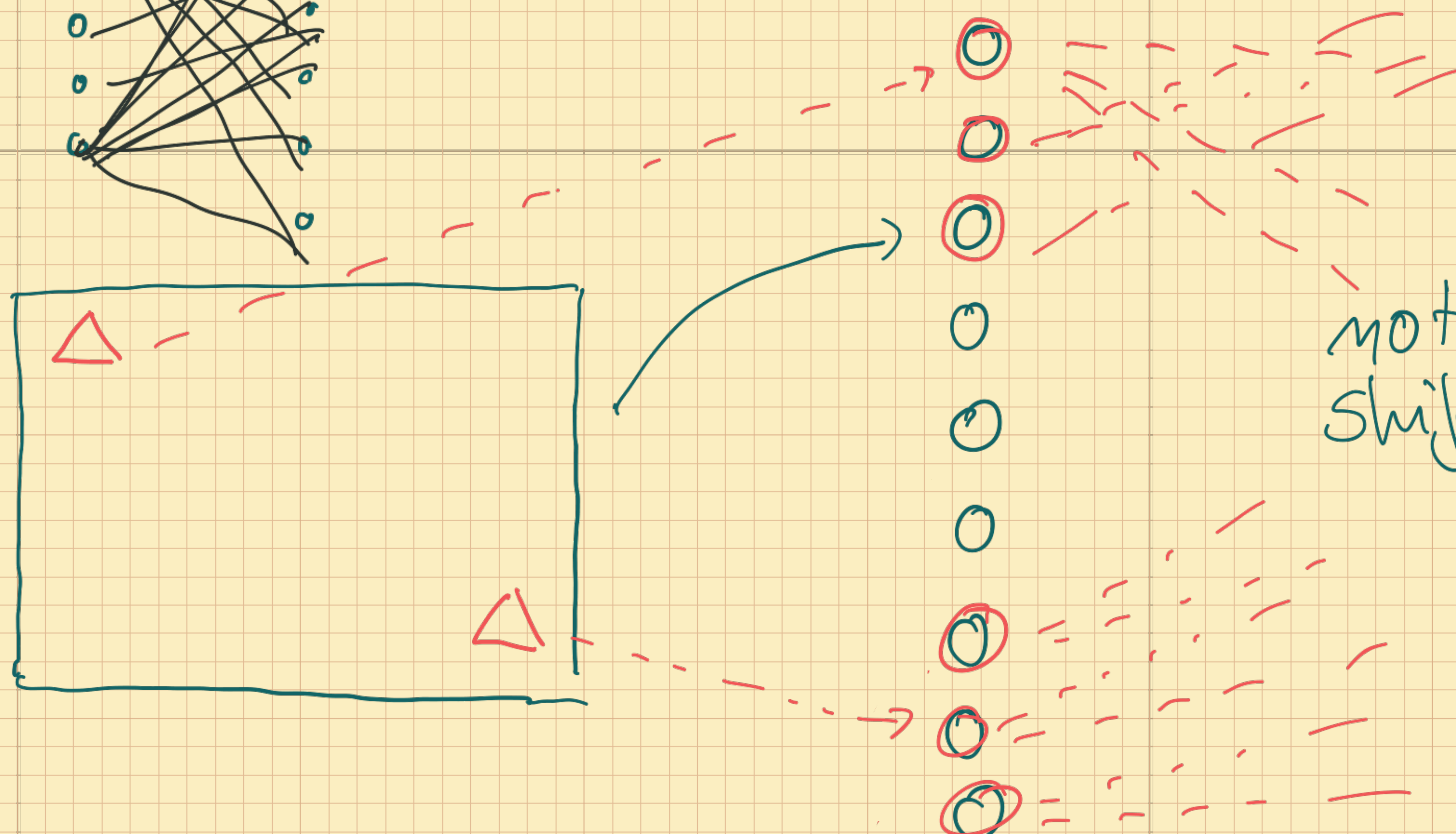


The MLP - problem(s):



→ N^2 behaviour of ~~neurons~~ parameters w

→ hard to train



not shift-invariant!

A better architecture?

→ a trainable set of basis functions
  aka filters

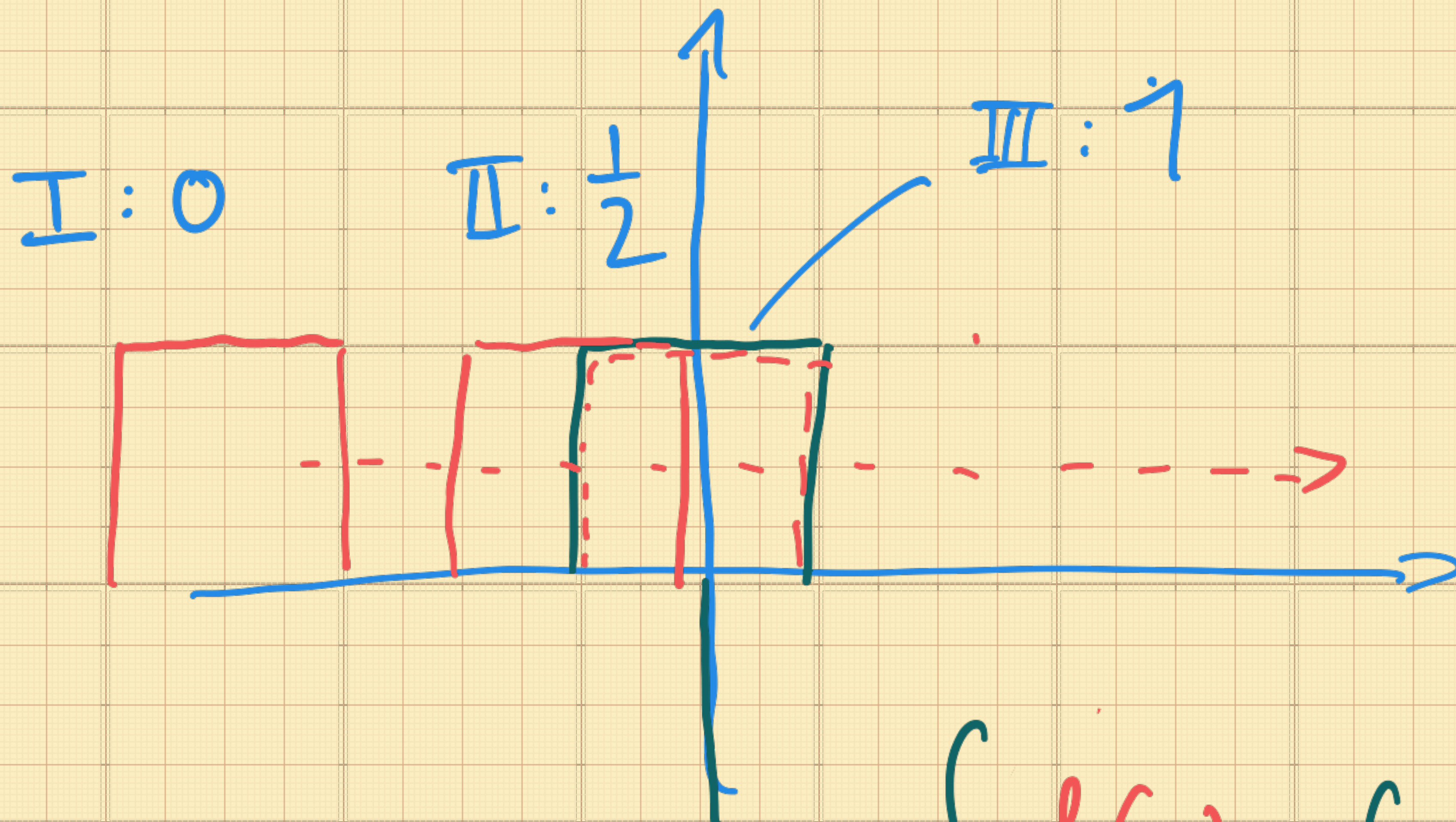
→ Convolutional NNs

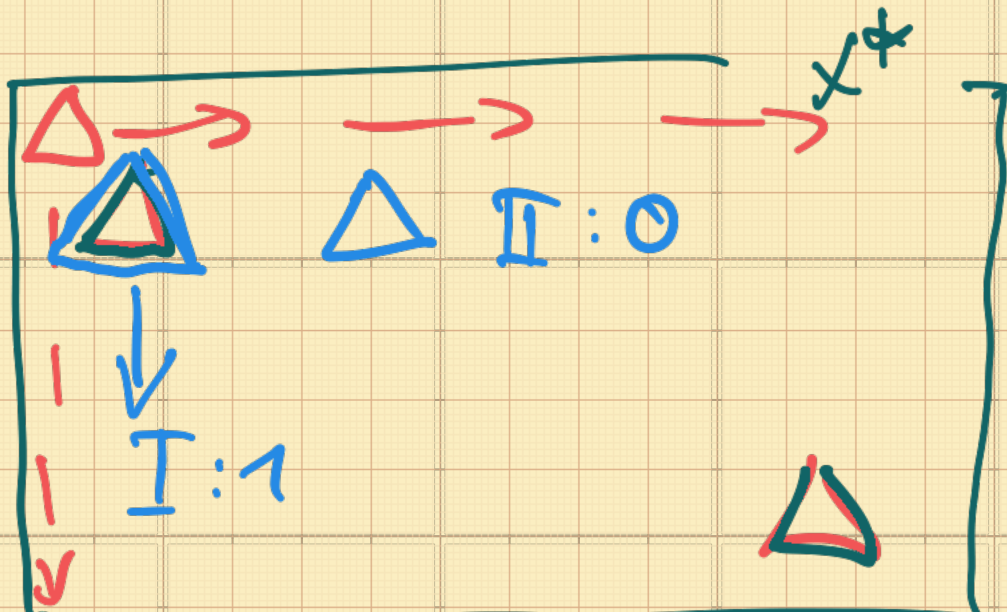
→ Convolution: a sliding projection:

continuous: $\int f(x)\, g(x^* - x)\, dx = (f * g)(x^*)$

$\downarrow$ approx.

discrete: $\sum_k f(k)\, g(k^* - k) = (f * g)(k^*)$





$$\int f(x)\, g(x^* - x)\, dx$$

trainable! → w !

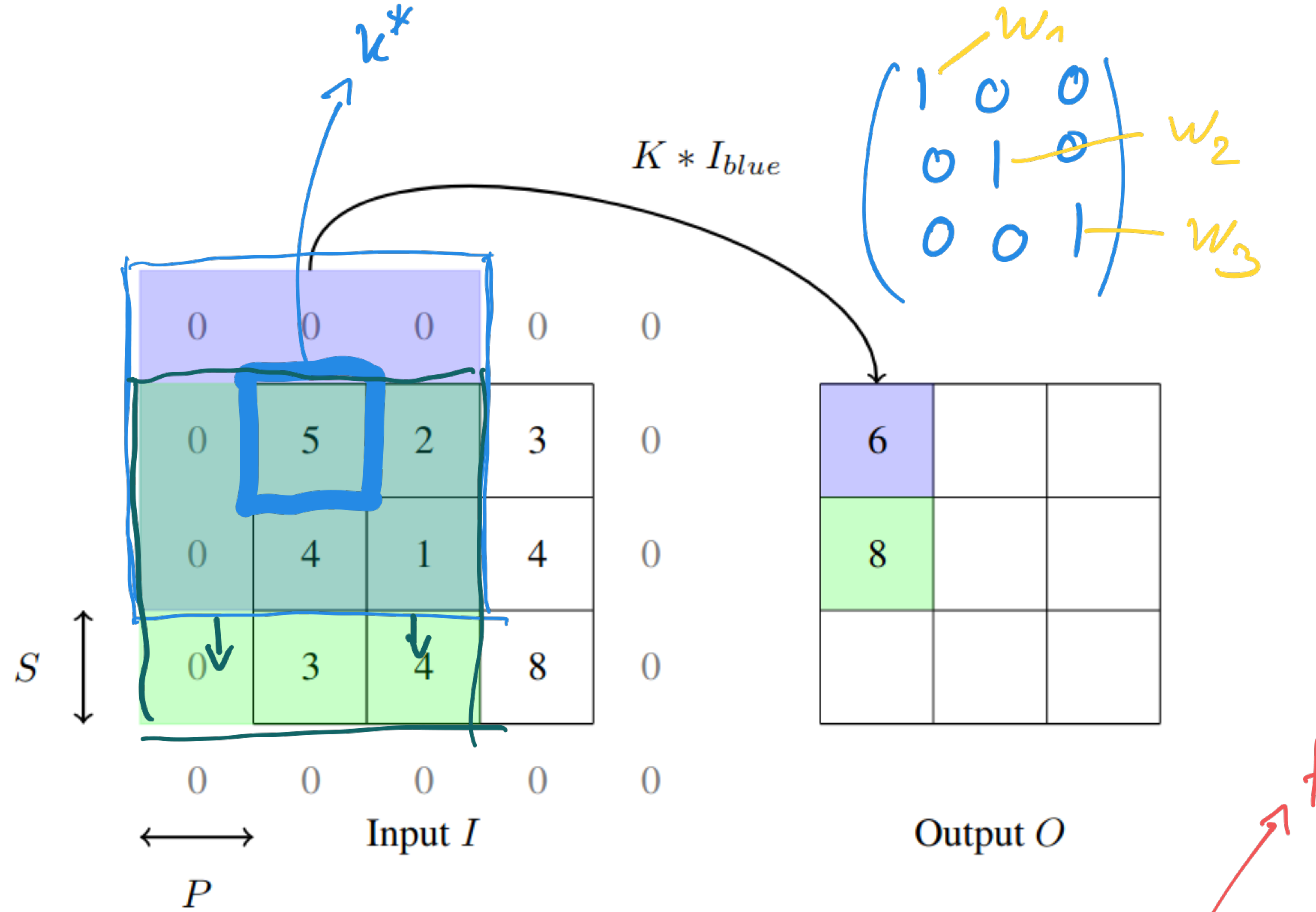
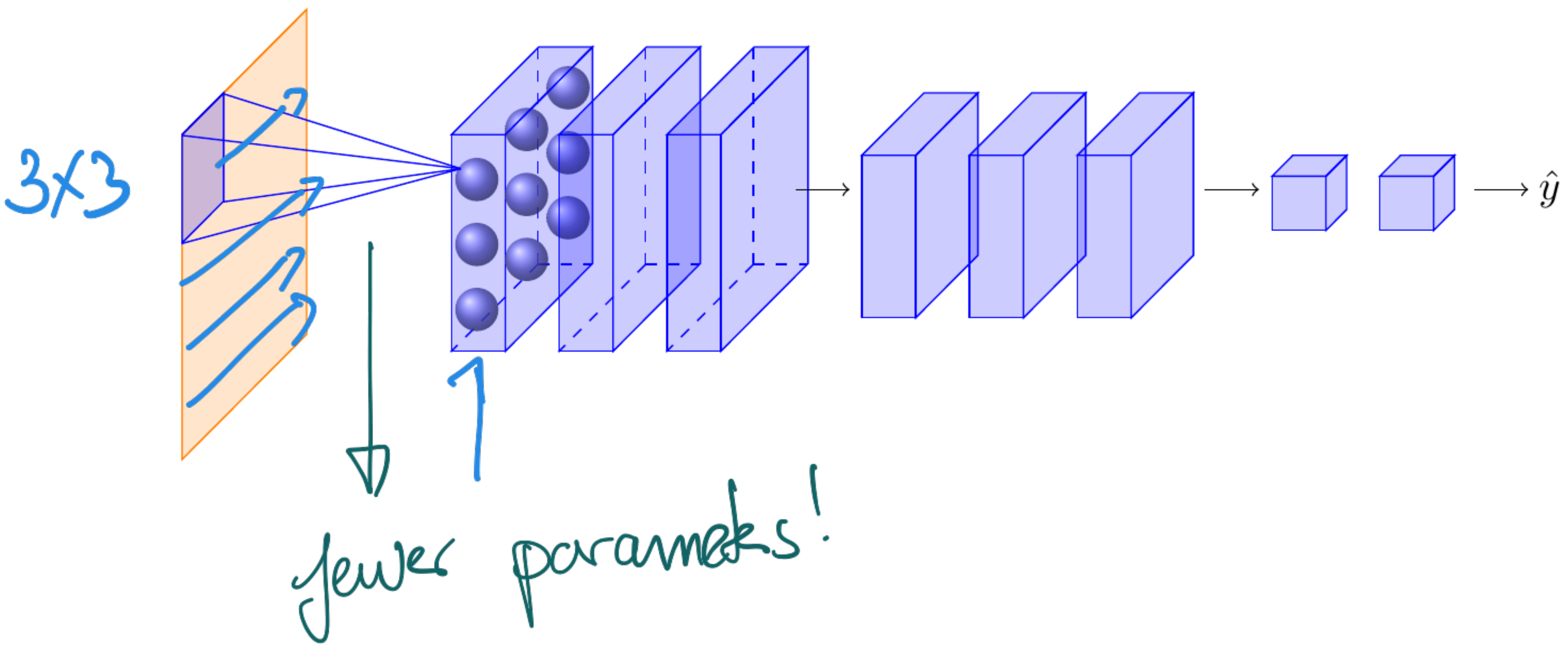


Figure 3: Illustration of a convolutional operation. The filter kernel K is of size 3×3, the colors indicate the corresponding filter input and output neurons. For demonstration purposes, the kernel is chosen the identity matrix, and for completeness stride (S) and padding (P) are chosen as $S = P = 1$.

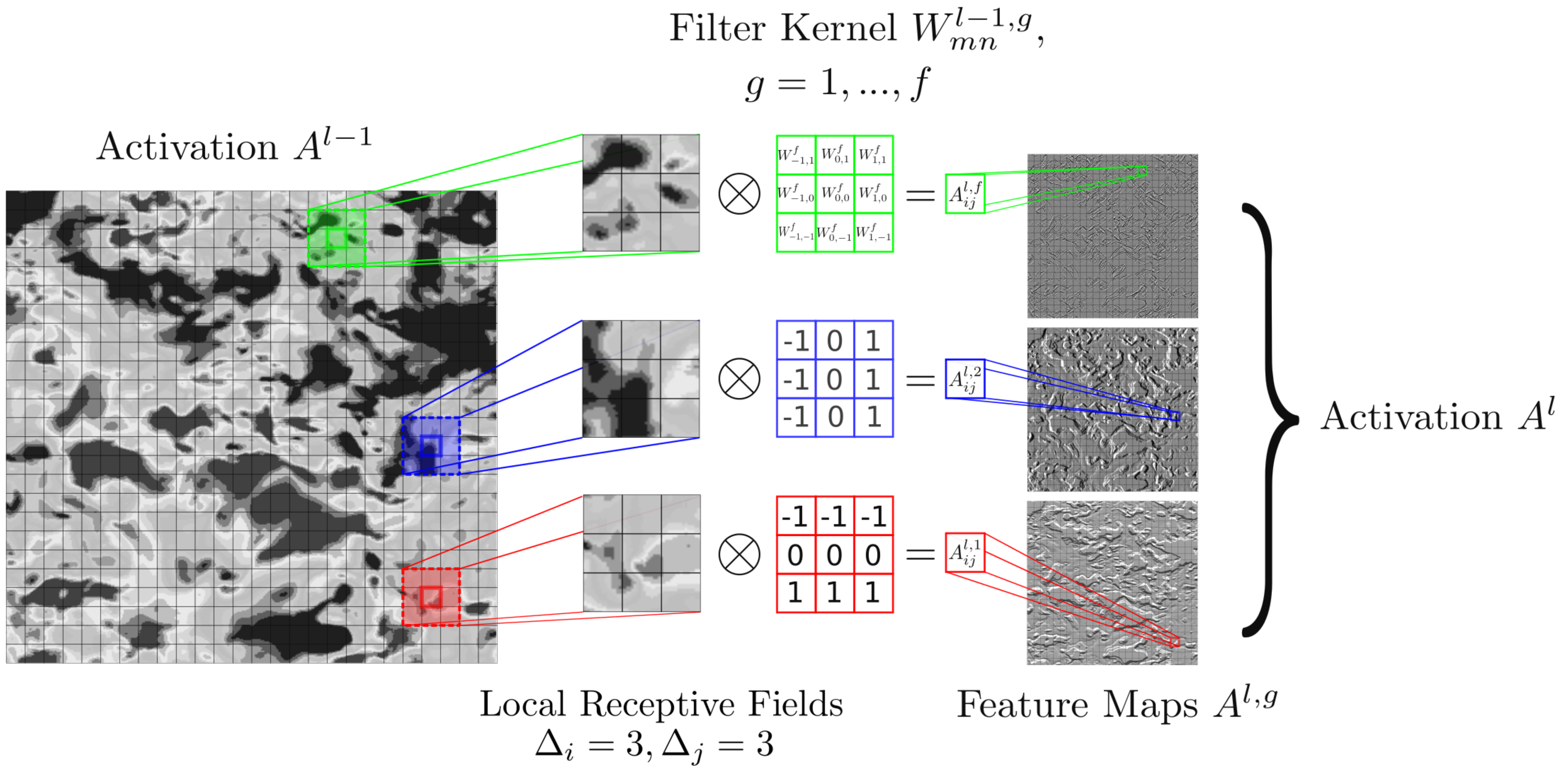
Advanced Architectures

- Convolutional Neural Networks
  - Local connectivity, multidimensional trainable filter kernels, discrete convolution, shift invariance, hierarchical representation
  - Current state of the art for multi-D data and segmentation



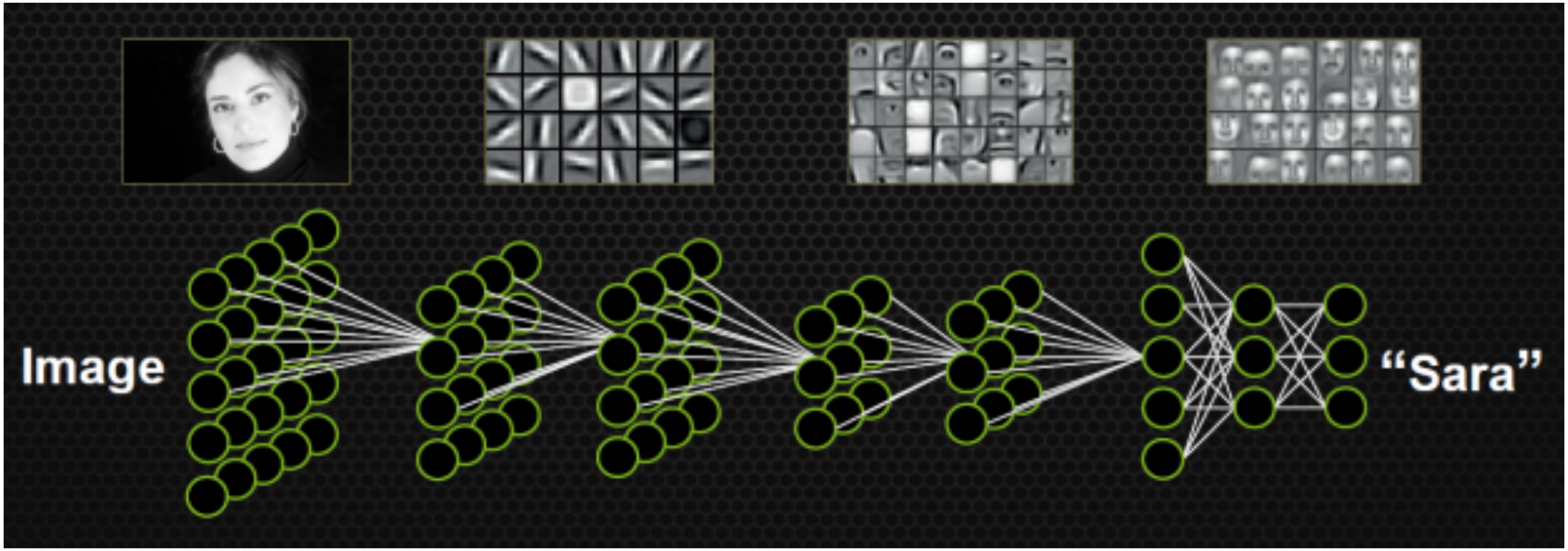
Advanced Architectures

- Convolutional Neural Networks



What does a CNN learn?

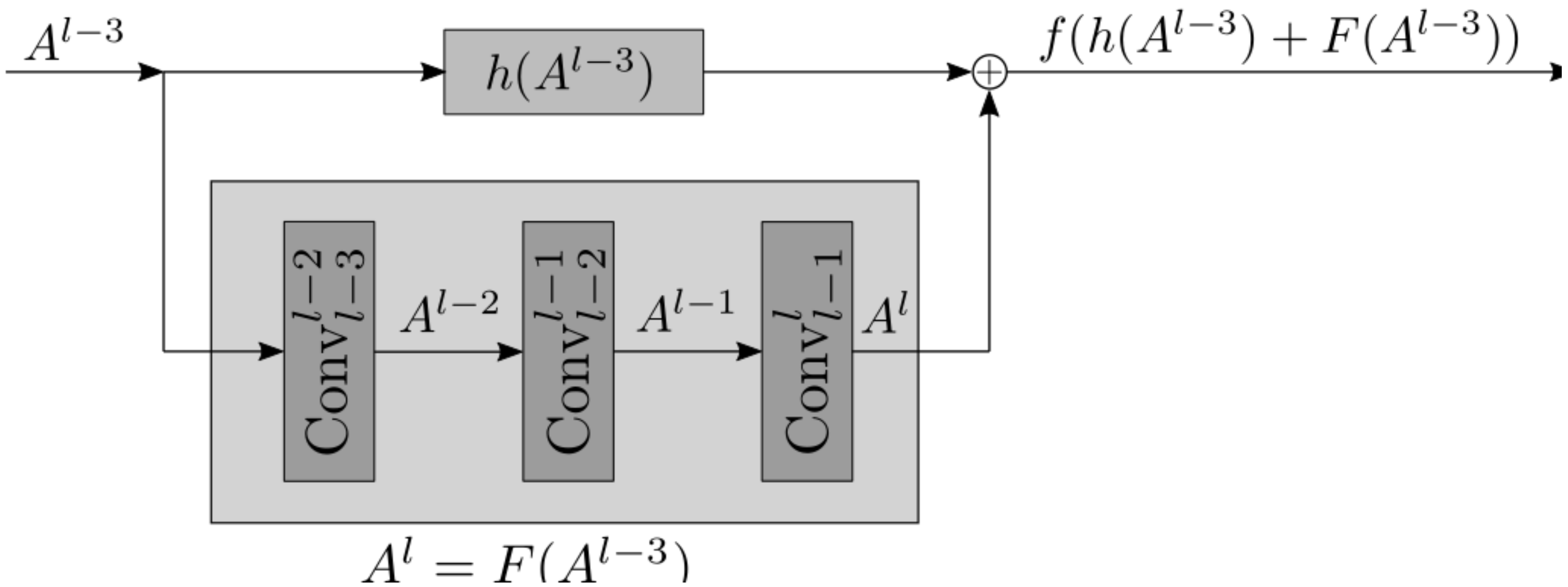
- Representation in hierarchical basis



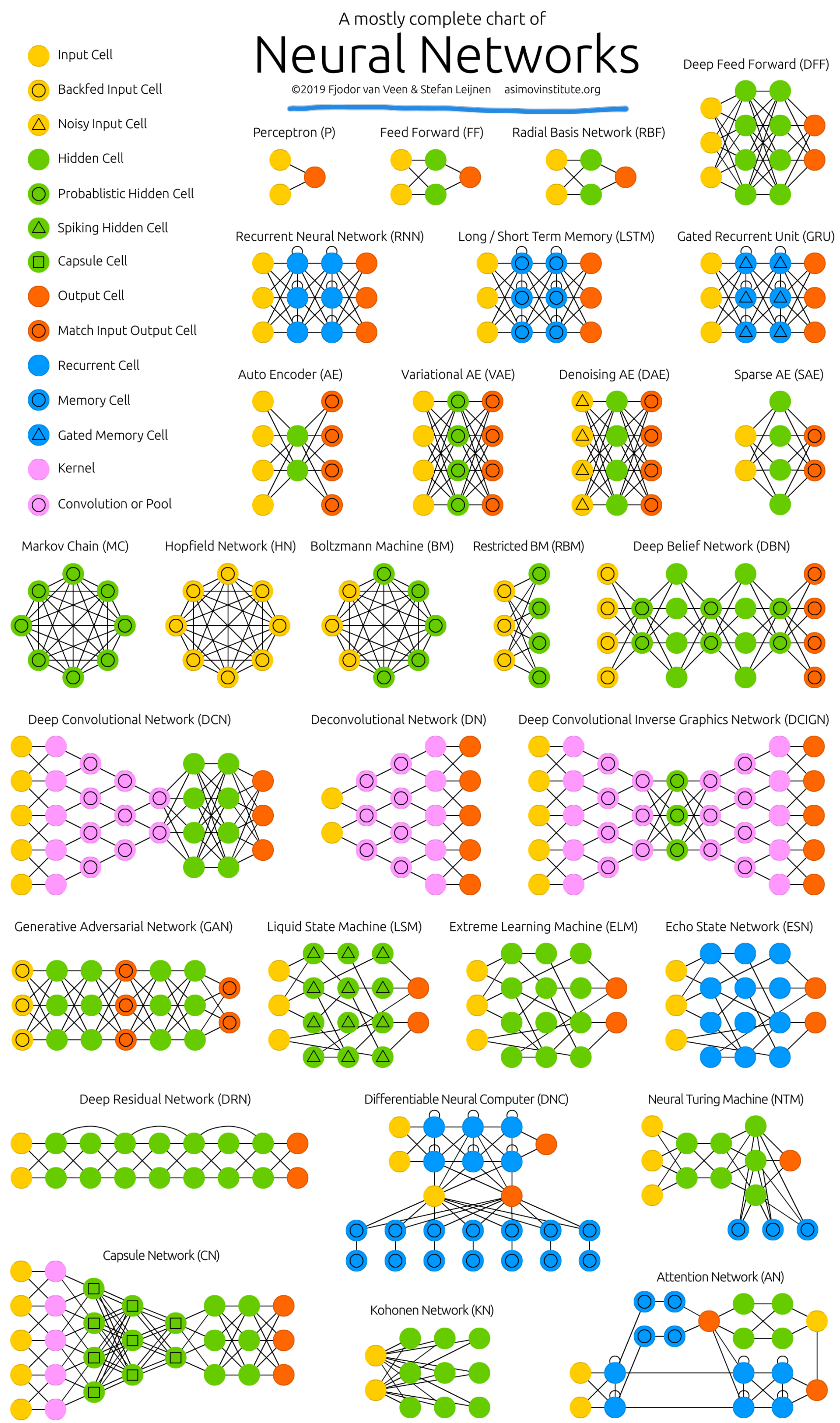
from: H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." In ICML 2009.

Residual Neural Networks

- He et al. recognized that the prediction performance of CNNs may deteriorate with depths (not an overfitting problem)
- Introduction of skip connectors or shortcuts, most often identity mappings
- A sought mapping, e.g. $G(A^{l-3})$ is split into a linear and non-linear (residual) part
- Fast passage of the linear part through the network: hundreds of CNN layers possible
- More robust identity mapping



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)