Convolutional neural networks

Deep learning course for industry

Mitko Veta

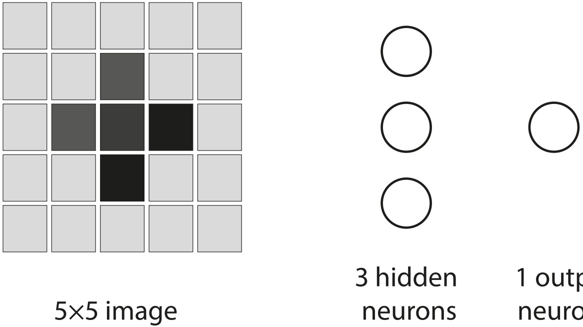
Eindhoven University of Technology
Department of Biomedical Engineering

2020

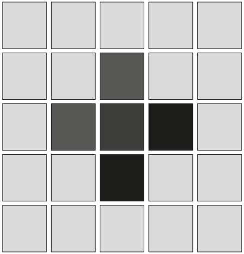
Learning goals

- Demonstrate how deep neural networks can be modified to be more suitable for image data.

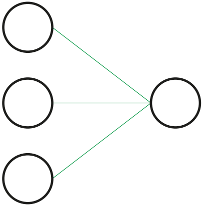
Images as inputs to a neural network



Images as inputs to a neural network



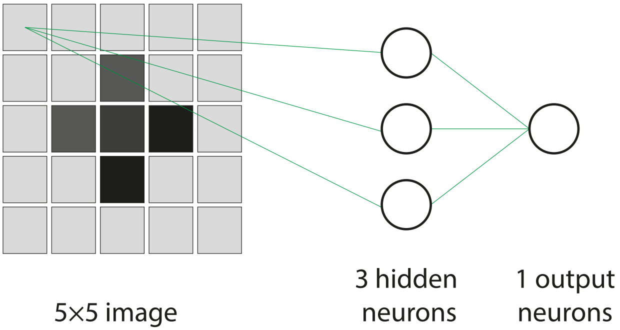
5×5 image



3 hidden neurons

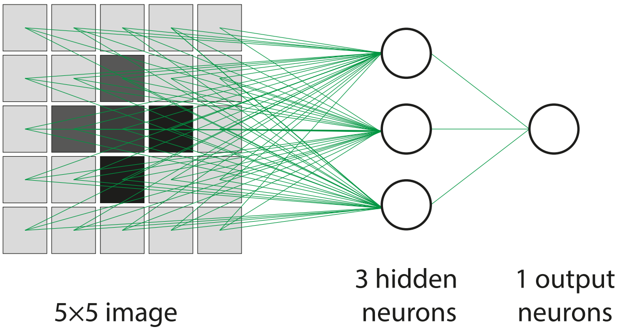
1 output neurons

Images as inputs to a neural network

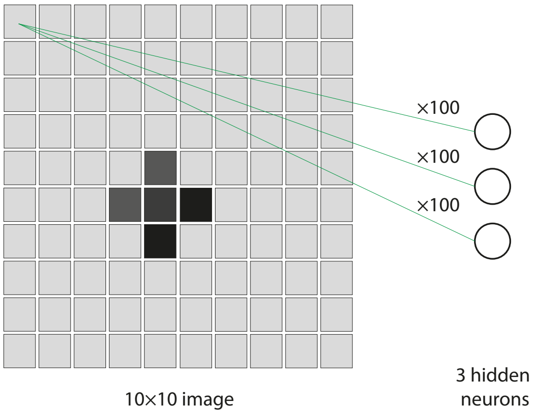


The biases $w_{i,0}$ are not shown.

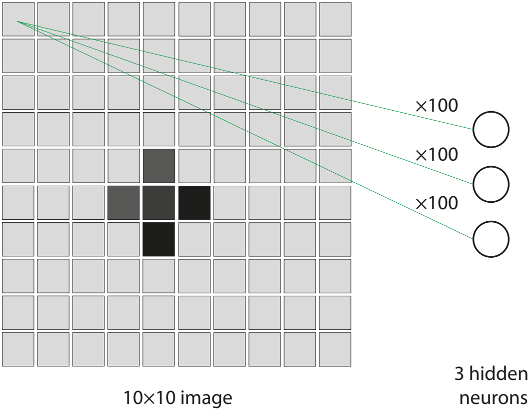
Images as inputs to a neural network



The number of parameters explodes with larger image sizes



The number of parameters explodes with larger image sizes

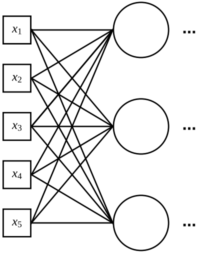


$$\# \text{ parameters} = (\text{height} \times \text{width} \times \# \text{ channels} + 1) \times \# neurons$$

The "+1" comes from the biases $w_{i,0}$.

Towards convolutional neural networks

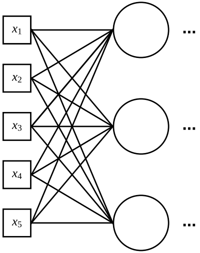
Example (1-D image for simplicity): 5×1 input image, 3 hidden neurons.



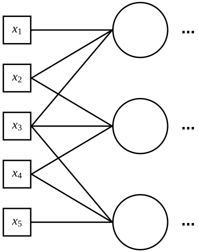
full connectivity: 15 parameters

Towards convolutional neural networks

Example (1-D image for simplicity): 5×1 input image, 3 hidden neurons.



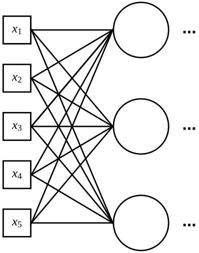
full connectivity: 15 parameters



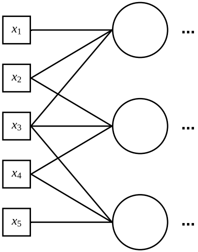
sparse connectivity: 9 parameters

Towards convolutional neural networks

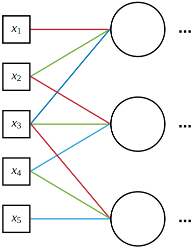
Example (1-D image for simplicity): 5×1 input image, 3 hidden neurons.



full connectivity: 15 parameters



sparse connectivity: 9 parameters

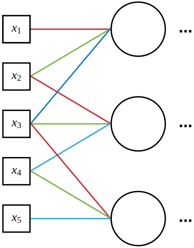


shared weights: 3 parameters

Note: the poor biases are again, ignored, but there are three of them in each case

Towards convolutional neural networks

Let the outputs of the three neurons be $\sigma(a_1), \sigma(a_2), \sigma(a_3)$. Then:



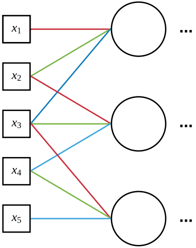
$$a_1 = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$a_2 = x_2 w_1 + x_3 w_2 + x_4 w_3$$

$$a_3 = x_3 w_1 + x_4 w_2 + x_5 w_3$$

Towards convolutional neural networks

Let the outputs of the three neurons be $\sigma(a_1), \sigma(a_2), \sigma(a_3)$. Then:



$$a_1 = x_1 w_1 + x_2 w_2 + x_3 w_3$$

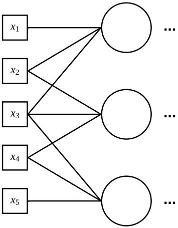
$$a_2 = x_2 w_1 + x_3 w_2 + x_4 w_3$$

$$a_3 = x_3 w_1 + x_4 w_2 + x_5 w_3$$

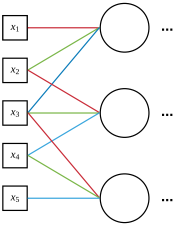
$$[a_1, a_2, a_3] = [x_1, x_2, x_3, x_4, x_5] * [w_3, w_2, w_1]$$

, where $*$ is the convolution operator, thus a **convolutional layer**.

Motivation (or rather a justification)

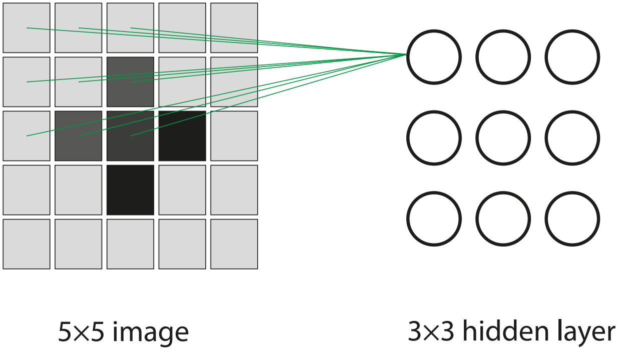


sparse connectivity
motivation: the features appear locally

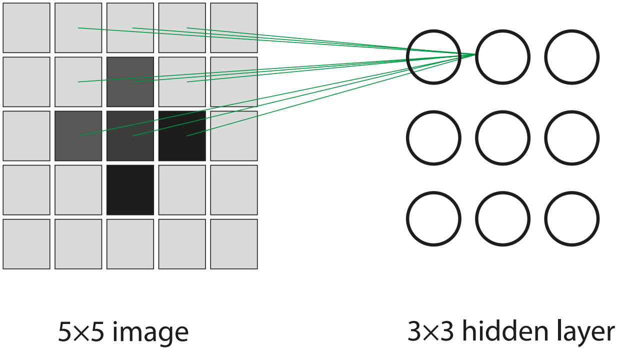


shared weights
motivation: the features repeat throughout the image

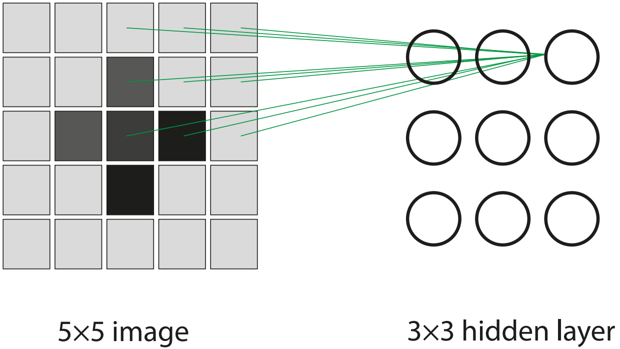
Towards convolutional neural networks in 2D



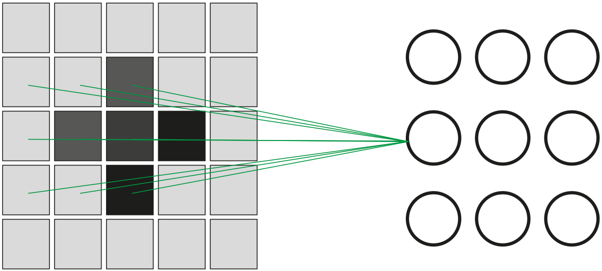
Towards convolutional neural networks in 2D



Towards convolutional neural networks in 2D



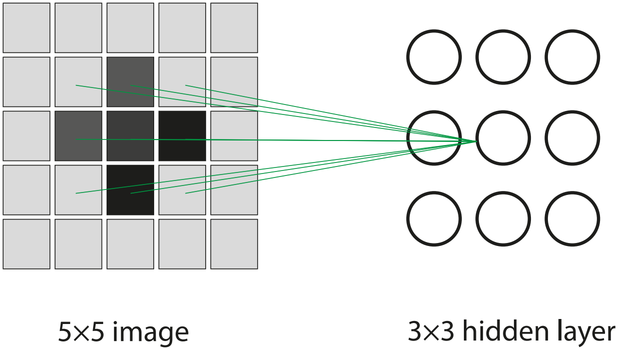
Towards convolutional neural networks in 2D



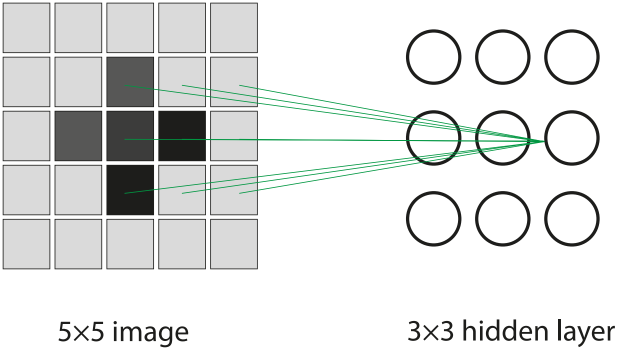
5×5 image

3×3 hidden layer

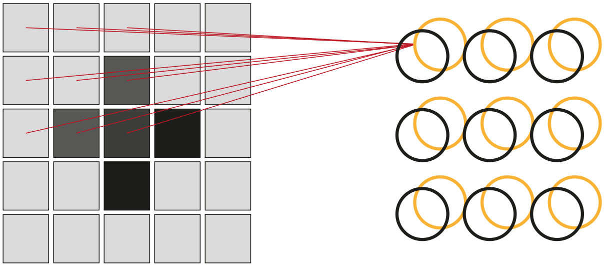
Towards convolutional neural networks in 2D



Towards convolutional neural networks in 2D



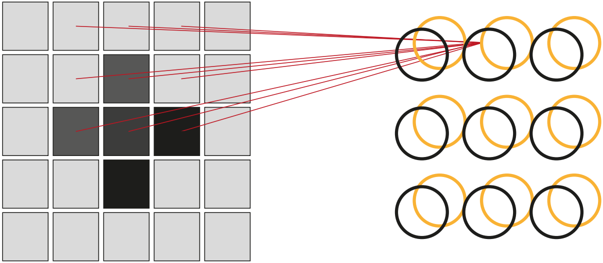
Adding a second feature map



5×5 image

3×3×2 hidden layer
(2 feature maps)

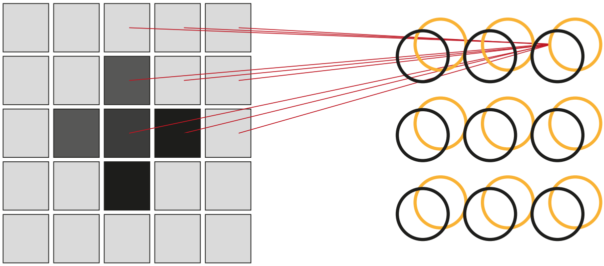
Adding a second feature map



5×5 image

3×3×2 hidden layer
(2 feature maps)

Adding a second feature map



5×5 image

3×3×2 hidden layer
(2 feature maps)

Convolution with padding

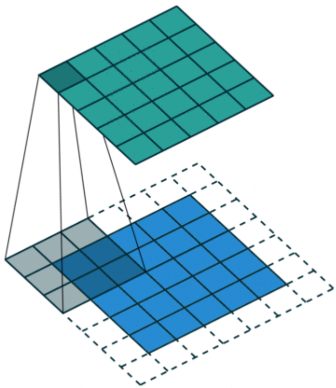
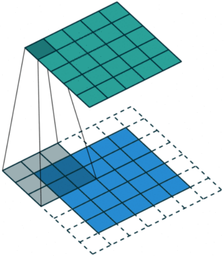


Figure source: https://github.com/vdumoulin/conv_arithmetic

Computing the output size

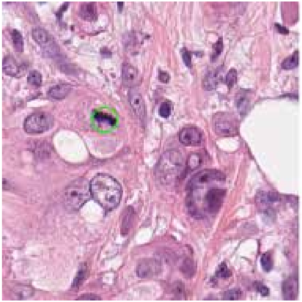


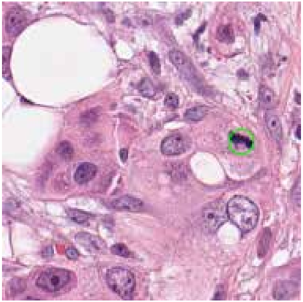
$$\text{output size} = \frac{\text{input size} - \text{kernel size} + 2 \times \text{padding}}{\text{stride}} + 1$$

In this example: input size $= 5$, kernel size $= 3$, padding $= 1$, stride $= 1$.
The output size is $(5 - 3 + 2 \times 1)/1 + 1 = 5$.

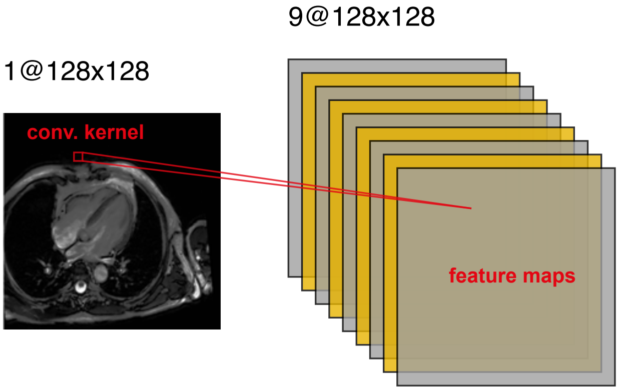
Motivation (or rather justification) for CNNs

The features of interest can appear at different locations in the image.

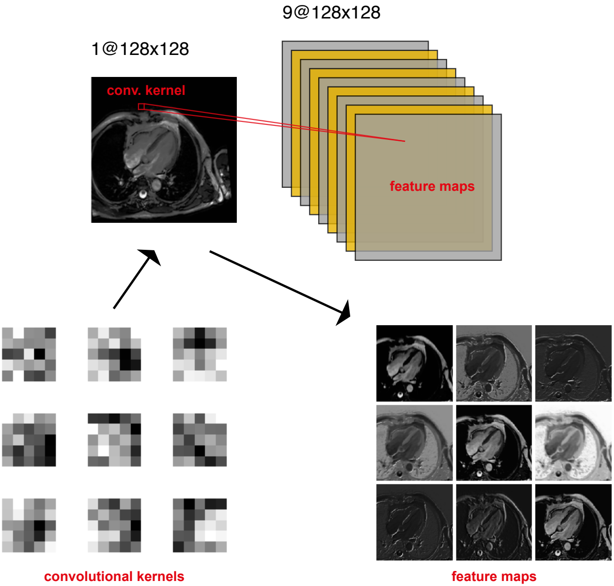




Kernels and feature maps



Kernels and feature maps



Motivation (or rather a consequence) for deep CNNs

The network learns low-level features in the first layers, and builds up towards more complex features in the deeper layers: intensity $\rightarrow$ edges and colour blobs $\rightarrow$ junctions $\rightarrow$ shapes $\rightarrow$ etc.

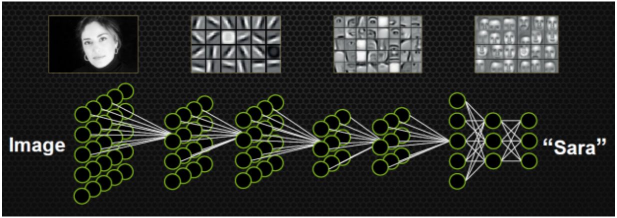


Figure source: nvidia.com

Equivariance and invariance to translation

The convolutional layers are **equivariant with translation**: as the input is translated, the output is translated in a predictable manner.

Equivalence and invariance to translation

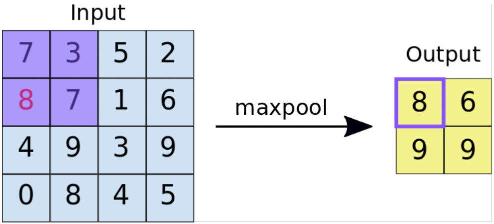
The convolutional layers are **equivariant with translation**: as the input is translated, the output is translated in a predictable manner.

A desired property of neural networks for classification is **invariance**: as the input is translated, the output remains the same.

Partial translational invariance of CNNs is achieved with the max-pooling operator.

Note: there are other types of invariance e.g. rotational.

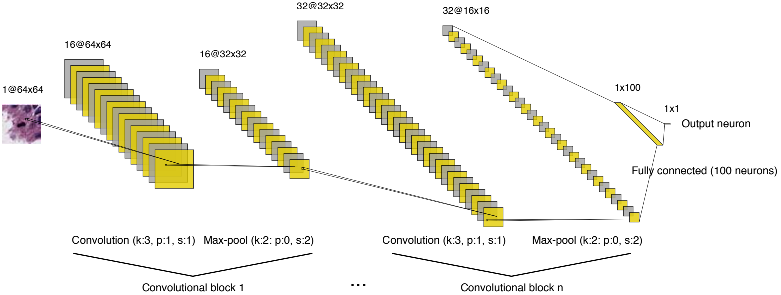
Max-pooling



A max-pool with a 2×2 kernel stride and size 2 (most common form) will reduce the image size by 2 in each dimension (a useful side-effect).

A "typical" CNN architecture for 2D image classification

Note that the convolution is a linear operation so non-linearities (such as ReLU) are still needed.



Summary

- Compared to fully connected neural networks, convolutional neural networks have sparse connectivity and weight sharing, which makes them suitable for image data.