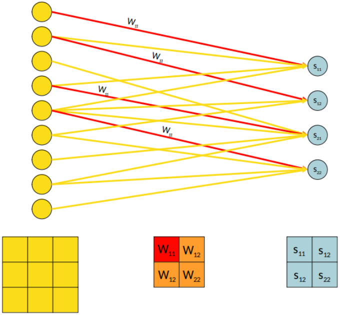
Deep Learning

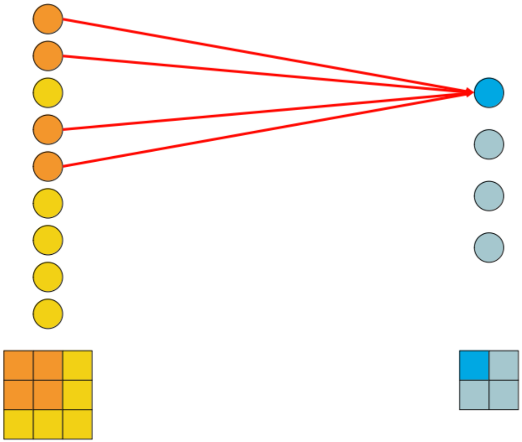
Properties of Convolution



Learning goals

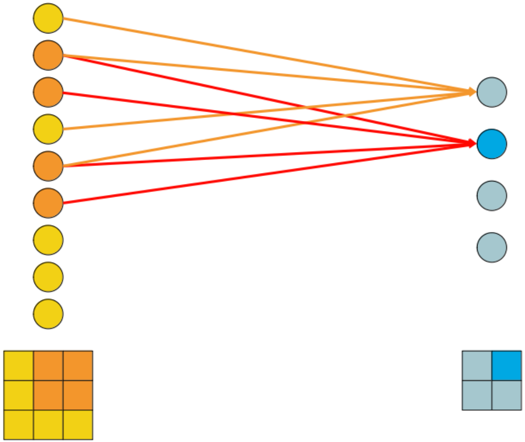
- Sparse Interactions
- Parameter Sharing
- Equivariance to Translation

SPARSE INTERACTIONS



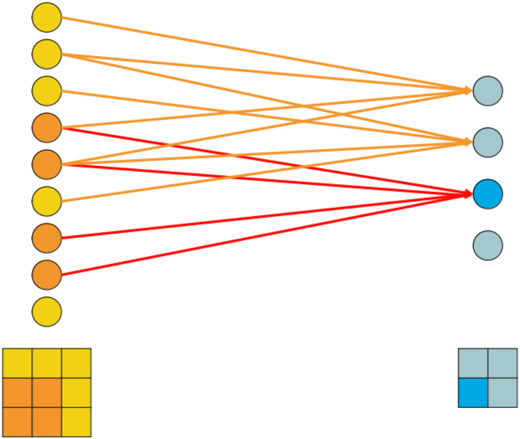
- We want to use the “neuron-wise” representation of our CNN.
- Moving the filter to the first spatial location yields us the first entry of the feature map which is composed of these four connections.

SPARSE INTERACTIONS



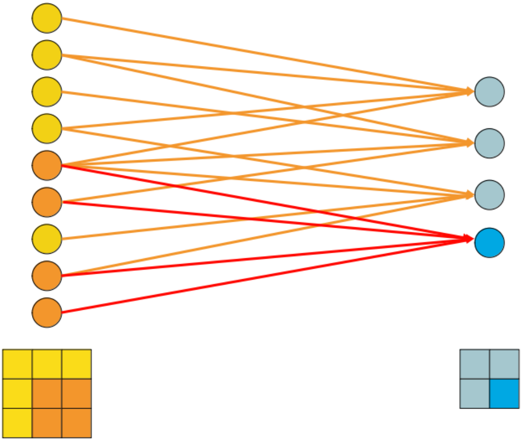
- Similarly...

SPARSE INTERACTIONS



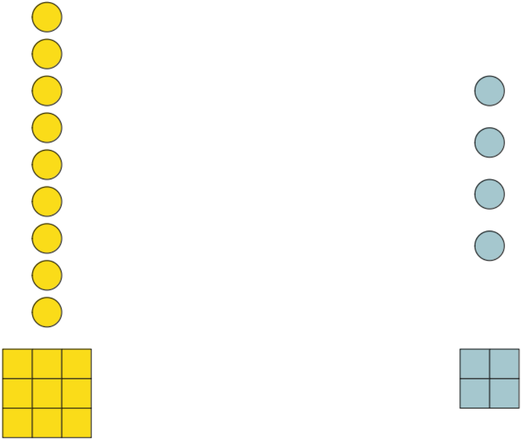
- Similarly...

SPARSE INTERACTIONS



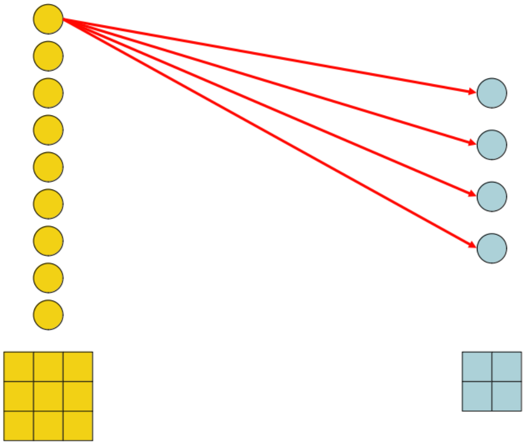
- Similarly...

SPARSE INTERACTIONS



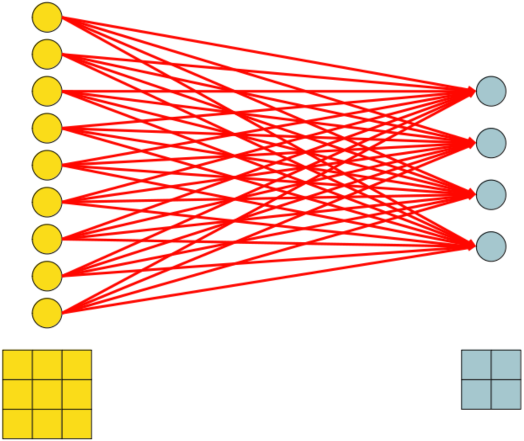
- Assume we would replicate the architecture with a dense net.

SPARSE INTERACTIONS



- Each input neuron is connected with each hidden layer neuron.

SPARSE INTERACTIONS

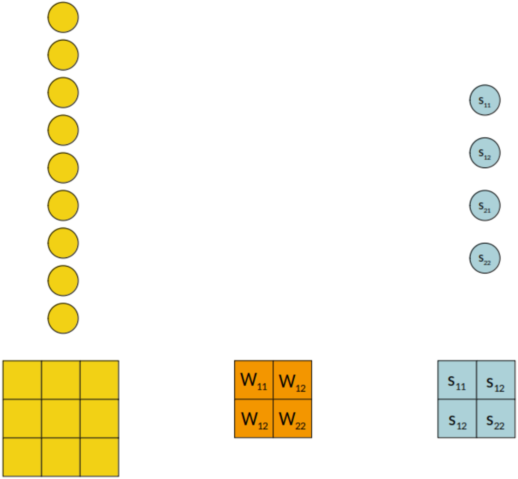


- In total, we obtain 36 connections!

SPARSE INTERACTIONS

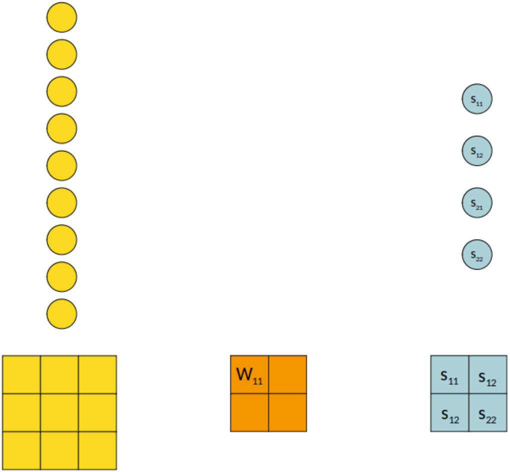
- What does that mean?
  - Our CNN has a **receptive field** of 4 neurons.
  - That means, we apply a “local search” for features.
  - A dense net on the other hand conducts a “global search”.
  - The receptive field of the dense net are 9 neurons.
- When processing images, it is more likely that features occur at specific locations in the input space.
- For example, it is more likely to find the eyes of a human in a certain area, like the face.
  - A CNN only incorporates the surrounding area of the filter into its feature extraction process.
  - The dense architecture on the other hand assumes that every single pixel entry has an influence on the eye, even pixels far away or in the background.

PARAMETER SHARING



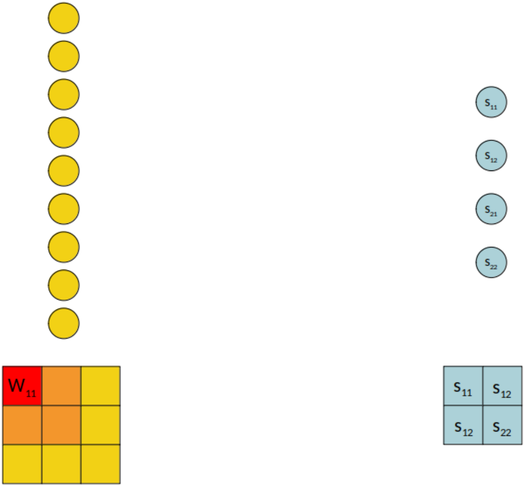
- For the next property we focus on the filter entries.

PARAMETER SHARING



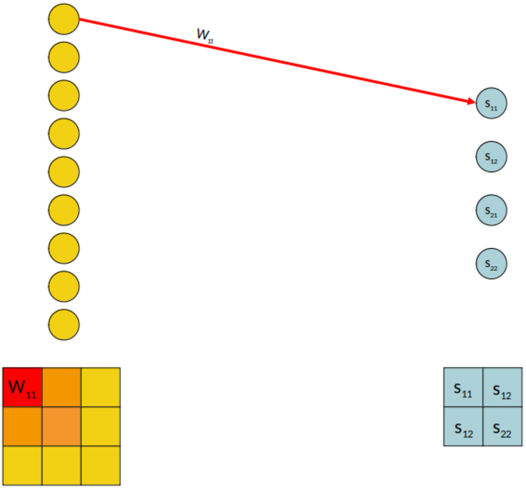
- In particular, we consider weight w_{11}

PARAMETER SHARING



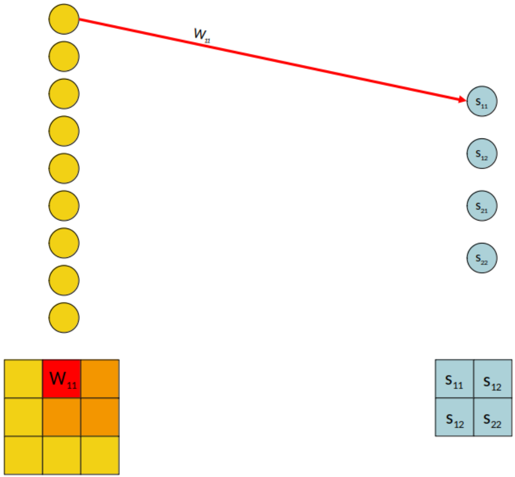
- As we move the filter to the first spatial location..

PARAMETER SHARING



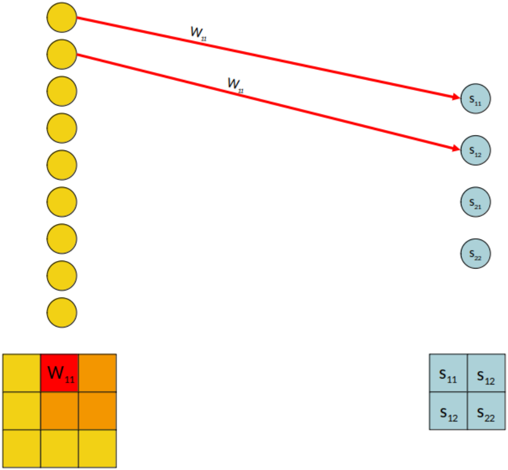
- ...we observe the following connection for weight w_{11}

PARAMETER SHARING



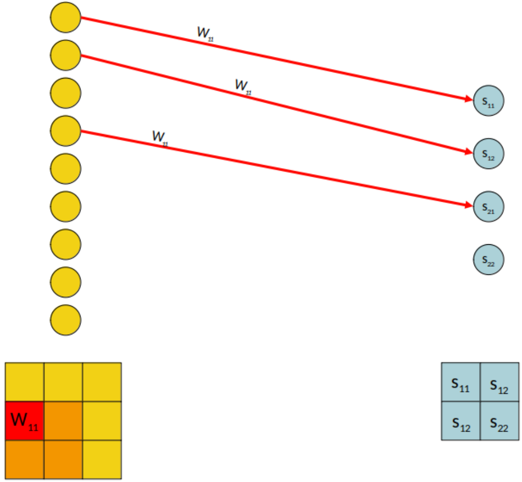
- Moving to the next location...

PARAMETER SHARING



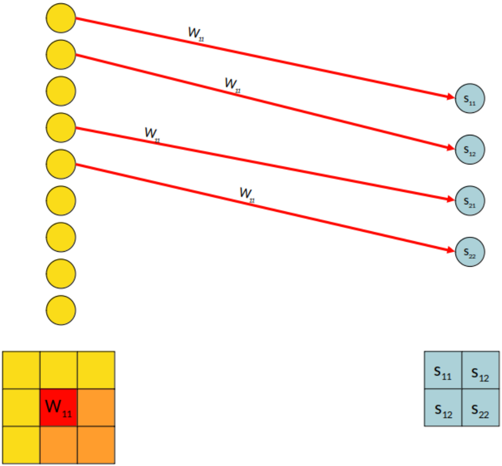
- ...highlights that we use the same weight more than once!

PARAMETER SHARING



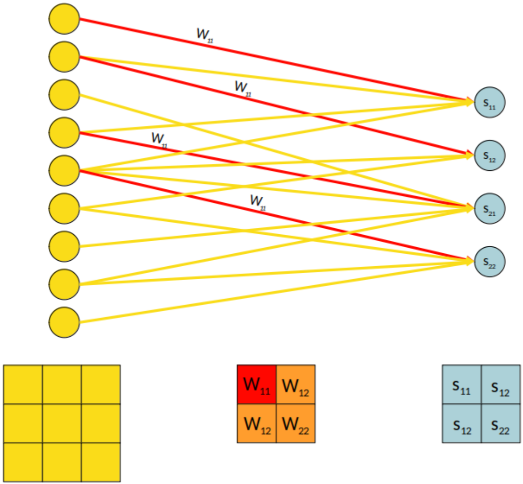
- Even three...

PARAMETER SHARING



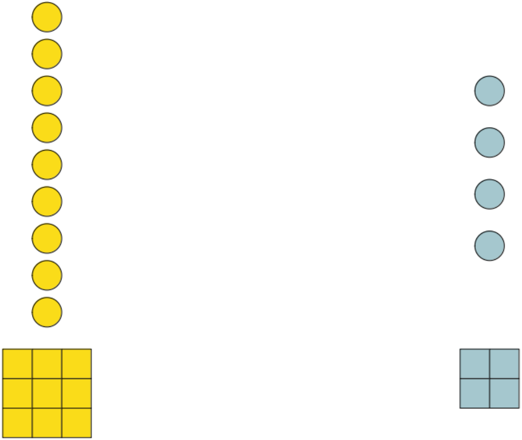
- And in total four times.

PARAMETER SHARING



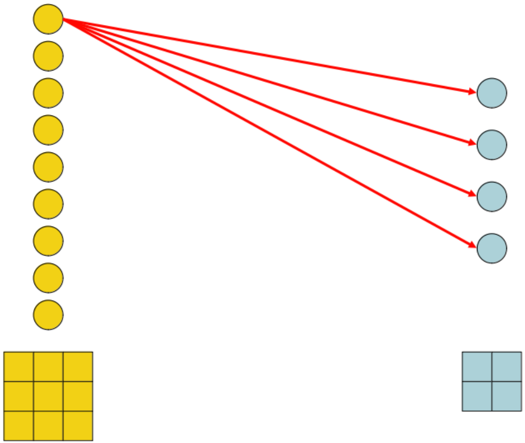
- All together, we have just used four weights.

PARAMETER SHARING



- How many weights does a corresponding dense net use?

PARAMETER SHARING



- $9 \cdot 4 = 36$! That is 9 times more weights!

SPARSE CONNECTIONS AND PARAMETER SHARING

- Why is that good?
- Less parameters drastically reduce memory requirements.
- Faster runtime:
  - For m inputs and n outputs, a fully connected layer requires $m \times n$ parameters and has $\mathcal{O}(m \times n)$ runtime.
  - A convolutional layer has limited connections $k << m$, thus only $k \times n$ parameters and $\mathcal{O}(k \times n)$ runtime.
- But it gets even better:
  - Less parameters mean less overfitting and better generalization!

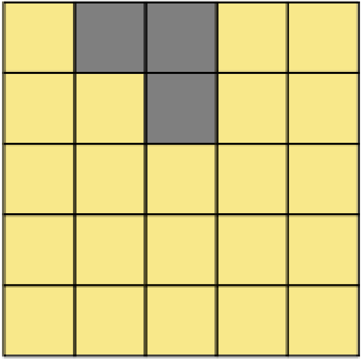
SPARSE CONNECTIONS AND PARAMETER SHARING

- Example: consider a color image with size 100×100.
- Suppose we would like to create one single feature map with a "same padding" (i.e. the hidden layer is of the same size).
  - Choosing a filter with size 5 means that we have a total of $5 \cdot 5 \cdot 3 = 75$ parameters (bias unconsidered).
  - A dense net with the same amount of "neurons" in the hidden layer results in

$$\underbrace{(100^2 \cdot 3)}_{\text{input}} \cdot \underbrace{(100^2)}_{\text{hidden layer}} = 300.000.000$$

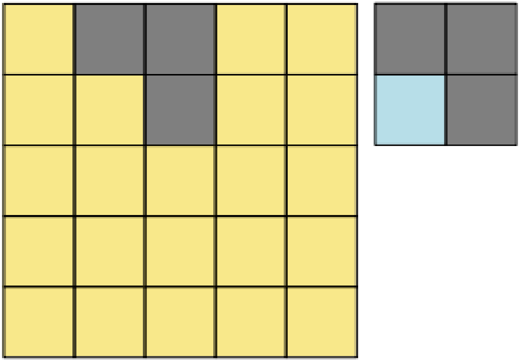
    parameters.
- Note that this was just a fictitious example. In practice we do not try to replicate CNN architectures with dense networks (actually it isn't even possible since physical limitations like the computer hardware would not allow us to).

EQUIVARIANCE TO TRANSLATION



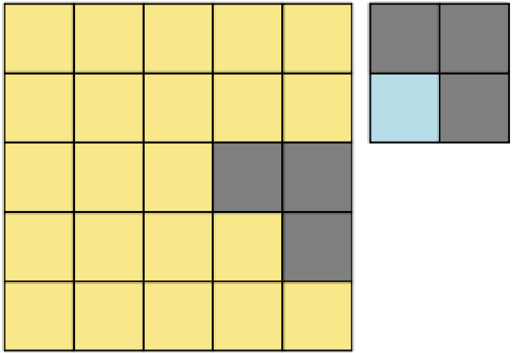
- Think of a specific feature of interest, here highlighted in grey.

EQUIVARIANCE TO TRANSLATION



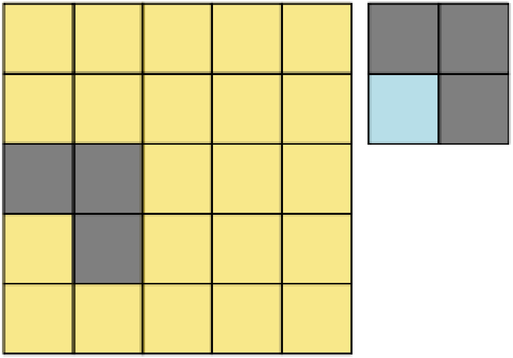
- Furthermore, assume we had a tuned filter looking for exactly that feature.

EQUIVARIANCE TO TRANSLATION



- The filter does not care at what location the feature of interest is located at.

EQUIVARIANCE TO TRANSLATION



- It is literally able to find it anywhere! That property is called **equivariance to translation**.

  Note: A function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$.

NONLINEARITY IN FEATURE MAPS

- As in dense nets, we use activation functions on all feature map entries to introduce nonlinearity in the net.
- Typically rectified linear units (ReLU) are used in CNNs:
  - They reduce the danger of saturating gradients compared to sigmoid activations.
  - They can lead to *sparse activations*, as neurons ≤ 0 are squashed to 0 which increases computational speed.
- As seen in the last chapter, many variants of ReLU (Leaky ReLU, ELU, PReLU, etc.) exist.