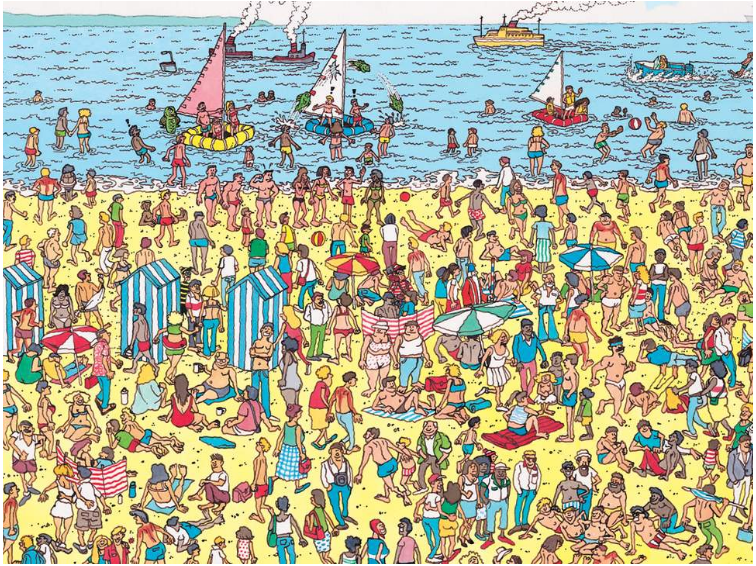
6.1 Convolutional Layer

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning



For object detection, vision systems should,

1. respond similarly to the same object regardless of *where* it appears in the image (**Translation Invariance**)
2. focus on *local regions*, without regard for what else is happening in the image at greater distances. (**Locality**)

Fully connected layer for images $h \times w$ images

- pixel images as inputs (represented as matrices)
  $x[i,j]$ pixel at location i j
- $h[i,j]$ hidden pixel at location i j

$$h[i,j] = \sum_{k,l} W[i,j,k,l] \cdot x[k,l]$$
$$= \sum_{a,b} V[i,j,a,b] \cdot x[i+a, j+b]$$

Where we set $V[i,j,a,b] = W[i,j,i+a,j+b]$, and re-index the subscripts (k,l) such that $k = i+a$ and $l = j+b$.

Dropping dependence on i and j in V yields a **convolutional** layer.

$$h[i,j] = \sum_a \sum_b V[a,b] \cdot x[i+a, j+b] \quad \textit{translation invariance}$$

$$h[i,j] = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} V[a,b] \cdot x[i+a, j+b] \quad \textit{locality}$$

In mathematics, the **convolution** between two functions f, g

- $f, g : \mathbb{R}^d \to R$

$$[f \circledast g](x) = \int_{\mathbb{R}^d} f(z)g(x-z)dz$$

  That is, we measure the overlap between f and g when both functions are shifted by x and 'flipped'.

- In the discrete case, the integral turns into a sum.

$$[f \circledast g](i) = \sum_a f(a)g(i-a)$$

- For two-dimensional discrete functions

$$[f \circledast g](i,j) = \sum_{a,b} f(a,b)g(i-a, j-b)$$

The convolutional layer actually performs a **cross-correlation** (i.e. no flipping):

$$h[i,j] = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} V[a,b] \cdot x[i+a, j+b]$$

$$h[i,j] = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} V[a,b] \cdot x[i+a, j+b]$$

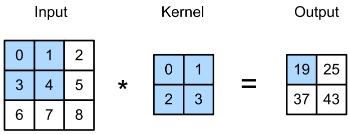
- The convolutional layer weighs intensities in windows of fixed size according to the **filter** V.
- Wherever correlation with the filter is high, we will also find a peak in $\mathbf{h}$.

So far we blissfully ignored that images consist of 3 **channels**: *red, green and blue*.

- Images are a 3rd order tensors
  - e.g., $1024 \times 1024 \times 3$.
  - index $\mathbf{x}$ as $x[i, j, k]$.
- The hidden layers also are 3rd order tensors
  - index $\mathbf{h}$ as $h[i, j, k]$.
  - The third coordinate is called **channels** or **feature maps**.
- The convolutional mask also is a 4th order tensor
  - indexed by $V[a, b, c, d]$.

$$h[i, j, k] = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{c} V[a, b, c, k] \cdot x[i + a, j + b, c]$$

Example



$$
\begin{aligned}
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 &= 19, \\
1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 &= 25, \\
3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 &= 37, \\
4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 &= 43.
\end{aligned}
$$

- The input is a two-dimensional array with a shape $H \times W$.
- The shape of the **kernel** (or **filter**) array is $h \times w$.
- Note that the output has a size of $(H - h + 1) \times (W - w + 1)$.

Summary

- **Translation invariance** in images implies that all patches of an image will be treated in the same manner.
- **Locality** means that only a small neighborhood of pixels will be used for computation.
- **Channels** on input and output allows for meaningful feature analysis.
- The core computation of a two-dimensional convolutional layer is a two-dimensional **cross-correlation** operation