Deep Learning

Convolutional Operation

Filter: 2x2x1 Output: 2x2x1

Input: 3x3x1

$$
\begin{aligned}
s_{11} &= a \cdot w_{11} + b \cdot w_{12} + d \cdot w_{21} + e \cdot w_{22} \\
s_{12} &= b \cdot w_{11} + c \cdot w_{12} + e \cdot w_{21} + f \cdot w_{22} \\
s_{21} &= d \cdot w_{11} + e \cdot w_{12} + g \cdot w_{21} + h \cdot w_{22} \\
s_{22} &= e \cdot w_{11} + f \cdot w_{12} + h \cdot w_{21} + i \cdot w_{22}
\end{aligned}
$$

Learning goals

- What are filters?
- Convolutional Operation
- 2D Convolution

FILTERS TO EXTRACT FEATURES

- Filters are widely applied in Computer Vision (CV) since the 70's.
- One prominent example: **Sobel-Filter**.
- It detects edges in images.

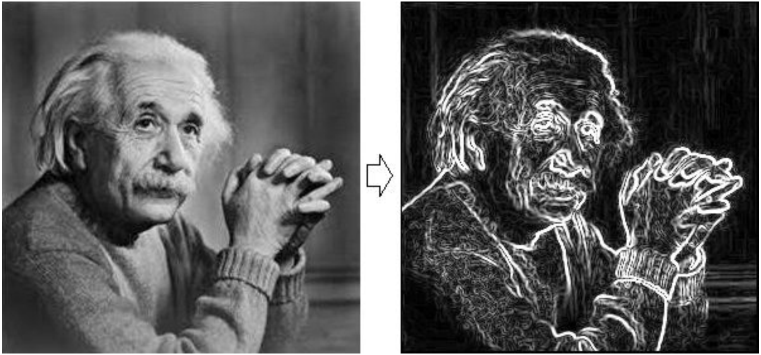


Figure: Sobel-filtered image.

FILTERS TO EXTRACT FEATURES

- Edges occur where the intensity over neighboring pixels changes fast.
- Thus, approximate the gradient of the intensity of each pixel.
- Sobel showed that the gradient image $\mathbf{G}_x$ of original image $\mathbf{A}$ in x-dimension can be approximated by:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} = \mathbf{S}_x * \mathbf{A}$$

  where $*$ indicates a mathematical operation known as a **convolution**, not a traditional matrix multiplication.
- The filter matrix $\mathbf{S}_x$ consists of the product of an **averaging** and a **differentiation** kernel:

$$\underbrace{\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T}_{\textit{averaging}} \underbrace{\begin{bmatrix} -1 & 0 & +1 \end{bmatrix}}_{\textit{differentiation}}$$

FILTERS TO EXTRACT FEATURES

- Similarly, the gradient image $\mathbf{G}_y$ in y-dimension can be approximated by:

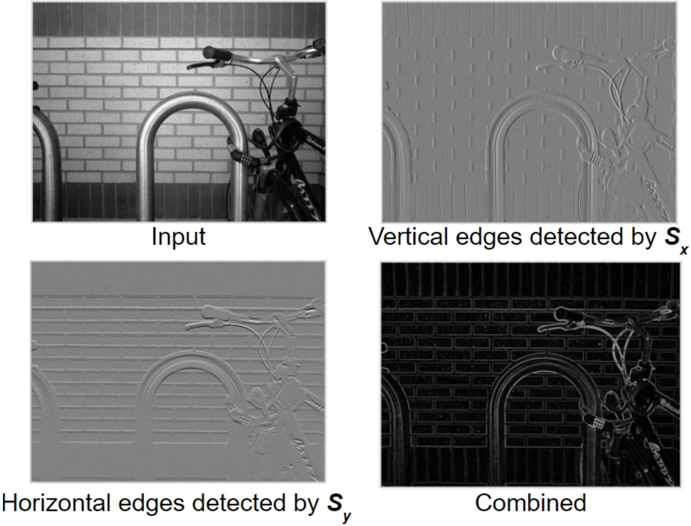
$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} = \mathbf{S}_y * \mathbf{A}$$

- The combination of both gradient images yields a dimension-independent gradient information $\mathbf{G}$:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

- These matrix operations were used to create the filtered picture of Albert Einstein.

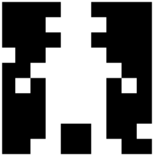
HORIZONTAL VS VERTICAL EDGES



Source: Wikipedia

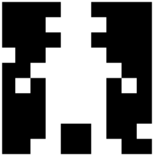
Figure: Sobel filtered images. Outputs are normalized in each case.

FILTERS TO EXTRACT FEATURES



- Let's do this on a dummy image.
- How to represent a digital image?

FILTERS TO EXTRACT FEATURES



0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

- Basically as an array of integers.

FILTERS TO EXTRACT FEATURES

Sobel-Operator

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \boxed{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

- $\mathbf{S}_x$ enables us to to detect vertical edges!

FILTERS TO EXTRACT FEATURES

Sobel-Operator

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \boxed{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

FILTERS TO EXTRACT FEATURES

Sobel-Operator

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \boxed{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

$$\begin{aligned}(\mathbf{G}_x)_{(i,j)} = (\mathbf{I} \star \mathbf{S}_x)_{(i,j)} = & -1 \cdot 0 \ + \ 0 \cdot 255 \ + \ \mathbf{1 \cdot 255} \\ & -2 \cdot 0 \ + \ 0 \cdot 0 \ + \ \mathbf{2 \cdot 255} \\ & -1 \cdot 0 \ + \ 0 \cdot 255 \ + \ \mathbf{1 \cdot 255}\end{aligned}$$

FILTERS TO EXTRACT FEATURES

0	510	1020	510	-510	-1020	-510	0
-255	510	$\boxed{1020}$	510	-510	-1020	-510	0
-255	765	765	255	-255	-765	-765	-255
255	765	510	0	0	-510	-765	-510
255	510	765	0	0	-765	-510	-255
0	765	1020	0	0	-1020	-765	0
0	1020	765	-255	255	-765	-1020	255
255	1020	0	-765	765	0	-1020	255

- Applying the Sobel-Operator to every location in the input yields the **feature map**.

FILTERS TO EXTRACT FEATURES

128	191	255	191	64	0	64	128
96	191	255	191	64	0	64	128
96	223	223	159	96	32	32	96
159	223	191	128	128	64	32	64
159	191	223	128	128	32	64	96
128	223	255	128	128	0	32	128
128	255	223	96	159	32	0	159
159	255	128	32	223	128	0	159

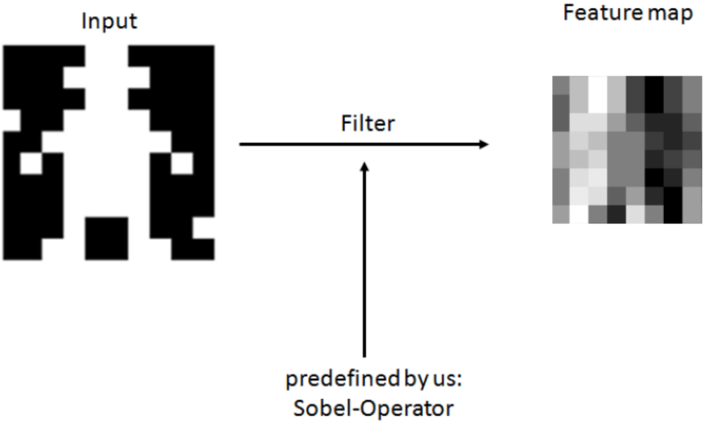
- Normalized feature map reveals vertical edges.
- Note the dimensional reduction compared to the dummy image.

WHY DO WE NEED TO KNOW ALL OF THAT?

- What we just did was extracting **pre-defined** features from our input (i.e. edges).
- A convolutional neural network does almost exactly the same: “extracting features from the input”.

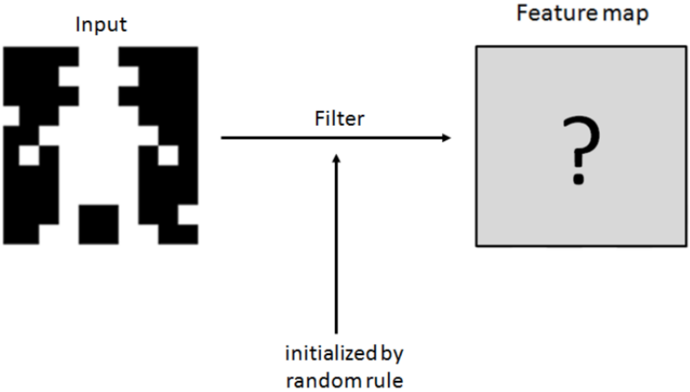
  $\Rightarrow$ The main difference is that we usually do not tell the CNN what to look for (pre-define them), **the CNN decides itself**.
- In a nutshell:
  - We initialize a lot of random filters (like the Sobel but just random entries) and apply them to our input.
  - Then, a classifier which (e.g. a feed forward neural net) uses them as input data.
  - Filter entries will be adjusted by common gradient descent methods.

WHY DO WE NEED TO KNOW ALL OF THAT?



$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

WHY DO WE NEED TO KNOW ALL OF THAT?



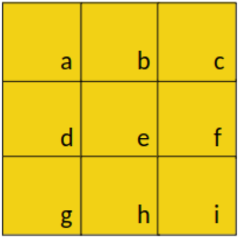
$$S_x = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

WORKING WITH IMAGES

- In order to understand the functionality of CNNs, we have to familiarize ourselves with some properties of images.
- Grey scale images:
  - Matrix with dimensions **h**eight $\times$ **w**idth $\times$ 1.
  - Pixel entries differ from 0 (black) to 255 (white).
- Color images:
  - Tensor with dimensions **h**eight $\times$ **w**idth $\times$ 3.
  - The depth 3 denotes the RGB values (red - green - blue).
- Filters:
  - A filter's depth is **always** equal to the input's depth!
  - In practice, filters are usually square.
  - Thus we only need one integer to define its size.
  - For example, a filter of size 2 applied on a color image actually has the dimensions $2 \times 2 \times 3$.

THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.

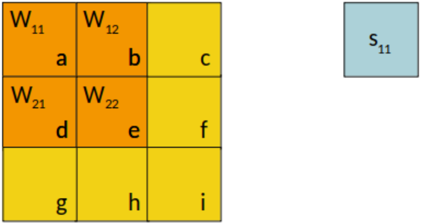


Input: 3x3x1

Filter: 2x2x1

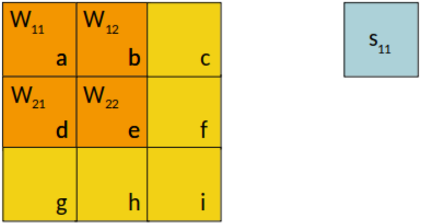
THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.



THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.

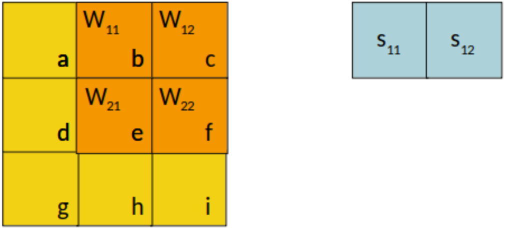


To obtain s_{11} we simply compute the dot product:

$$s_{11} = a \cdot w_{11} + b \cdot w_{12} + d \cdot w_{21} + e \cdot w_{22}$$

THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.

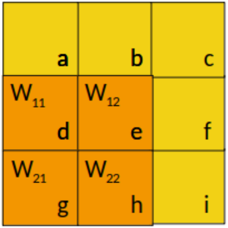


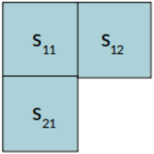
Same for s_{12}:

$$s_{12} = b \cdot w_{11} + c \cdot w_{12} + e \cdot w_{21} + f \cdot w_{22}$$

THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.



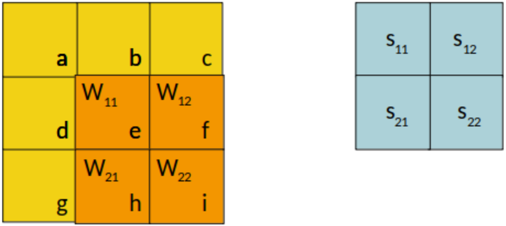


As well as for s_{21}:

$s_{21} = d \cdot w_{11} + e \cdot w_{12} + g \cdot w_{21} + h \cdot w_{22}$

THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.

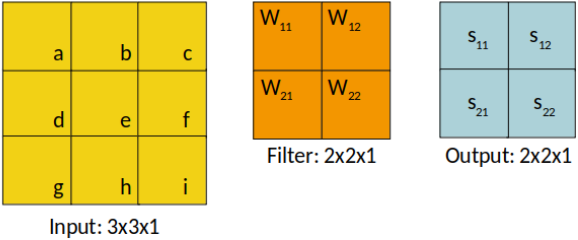


And finally for s_{22}:

$$s_{22} = e \cdot w_{11} + f \cdot w_{12} + h \cdot w_{21} + i \cdot w_{22}$$

THE 2D CONVOLUTION

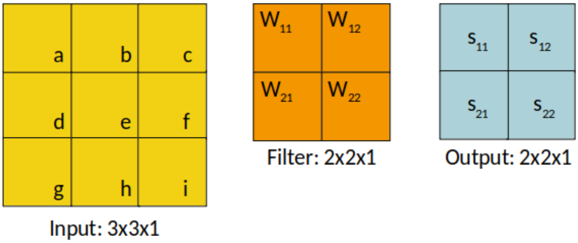
- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.



$$s_{11} = a \cdot w_{11} + b \cdot w_{12} + d \cdot w_{21} + e \cdot w_{22}$$
$$s_{12} = b \cdot w_{11} + c \cdot w_{12} + e \cdot w_{21} + f \cdot w_{22}$$
$$s_{21} = d \cdot w_{11} + e \cdot w_{12} + g \cdot w_{21} + h \cdot w_{22}$$
$$s_{22} = e \cdot w_{11} + f \cdot w_{12} + h \cdot w_{21} + i \cdot w_{22}$$

THE 2D CONVOLUTION

- Suppose we have an input with entries $a, b, \ldots, i$ (think of pixel values).
- The filter we would like to apply has weights w_{11}, w_{12}, w_{21} and w_{22}.



More generally, let I be the matrix representing the input and W be the filter/kernel. Then the entries of the output matrix are defined by $s_{ij} = \sum_{m,n} I_{i+m-1,j+n-1} w_{mn}$ where m, n denote the image size and kernel size respectively.