Sapienza Training Camp 2020

Building an Image Search Engine

3 - 5 September, 2020

Recap

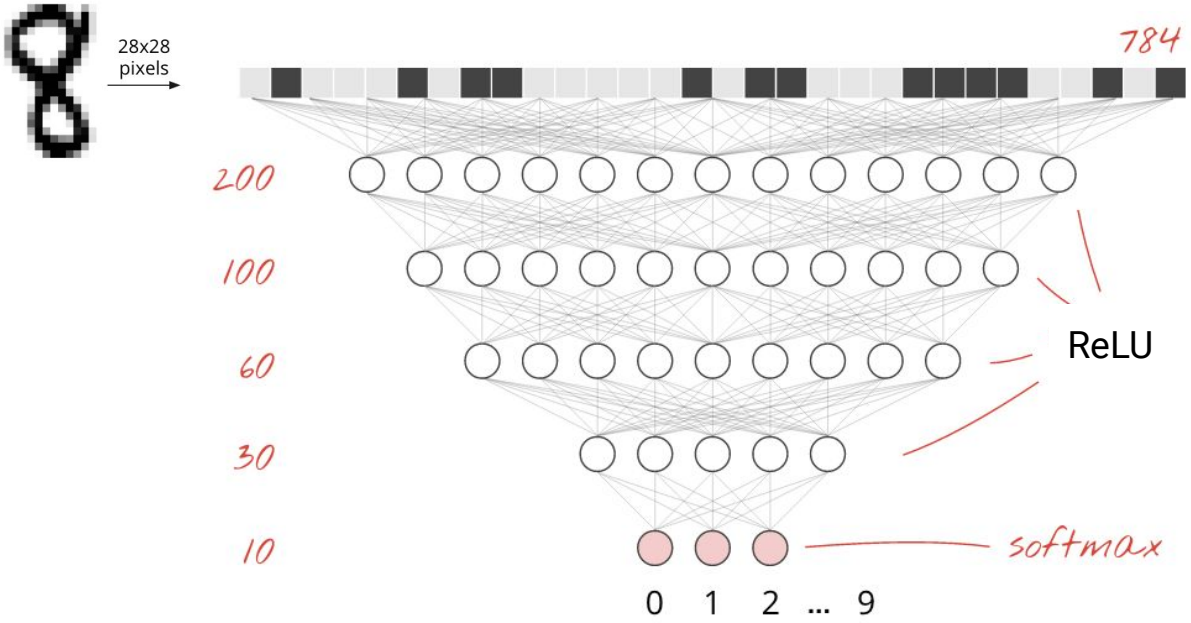
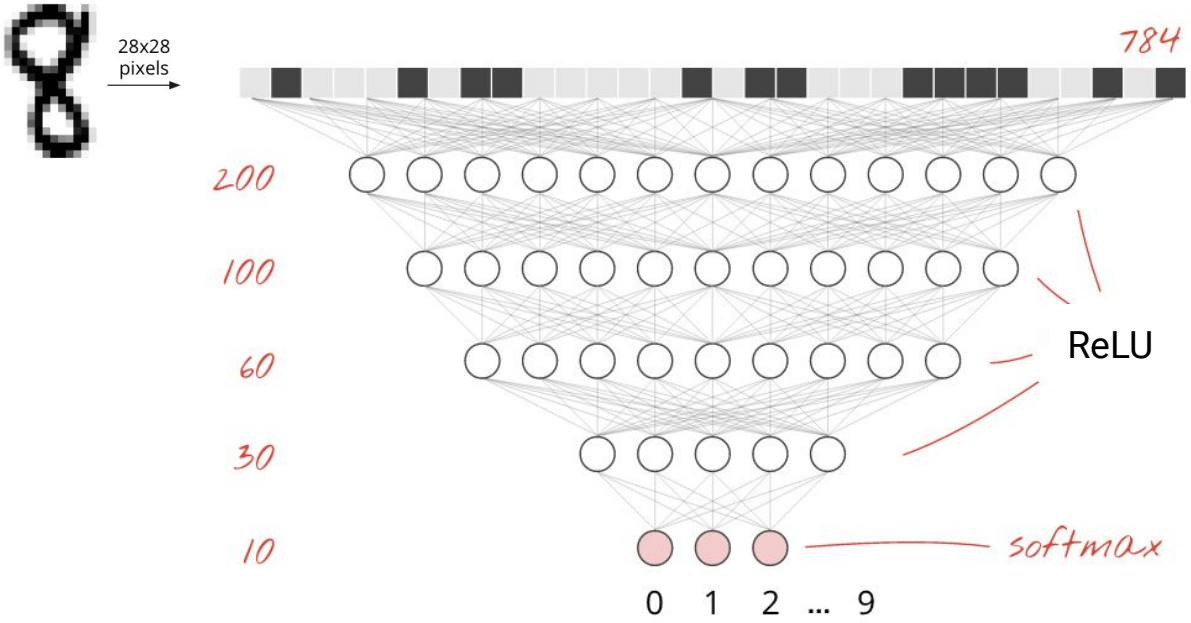


Image from "TensorFlow, Keras and deep learning, without a PhD"

Recap



- ✘ Too many parameters
- ✘ No translation invariance
- ✘ Fixed input size

Image from "TensorFlow, Keras and deep learning, without a PhD"

Convolutional Neural Networks (CNNs)

Advantages of CNNs (or convnets):

- 2D translation invariance

Convolutional Neural Networks (CNNs)

Advantages of CNNs (or convnets):

- 2D translation invariance
- More efficient use of the model parameters

Convolutional Neural Networks (CNNs)

Advantages of CNNs (or convnets):

- 2D translation invariance
- More efficient use of the model parameters
- Some robustness to image deformations

Convolutional Neural Networks (CNNs)

Advantages of CNNs (or convnets):

- 2D translation invariance
- More efficient use of the model parameters
- Some robustness to image deformations
- Applicable to input of any size

Convolutional Neural Networks (CNNs)

What is convolution? How do CNNs work?

2D Convolution

input image: $\mathbf{x}[m, n]$

Image source: http://www.robots.ox.ac.uk/~vgg/research/affine/

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

-1	0	1
-1	0	1
-1	0	1

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

-1	0	1
-1	0	1
-1	0	1

Convolution: the value of a pixel in the output is given by a linear combination of the pixel values in its local neighborhood:

$$a[m, n] = (x \otimes w)[m, n] = \sum_{k,l} x[m - k, n - l]w[k, l]$$

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

-1	0	1
-1	0	1
-1	0	1

=

$\mathbf{a}[m, n]$

?

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

-1	0	1
-1	0	1
-1	0	1

=

$\mathbf{a}[m, n]$

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

-1	-1	-1
0	0	0
1	1	1

=

$\mathbf{a}[m, n]$

?

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

-1	-1	-1
0	0	0
1	1	1

$=$

$\mathbf{a}[m, n]$

2D Convolution

input image: $\mathbf{x}[m, n]$

$\otimes$

filter: $\mathbf{w}[k, l]$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

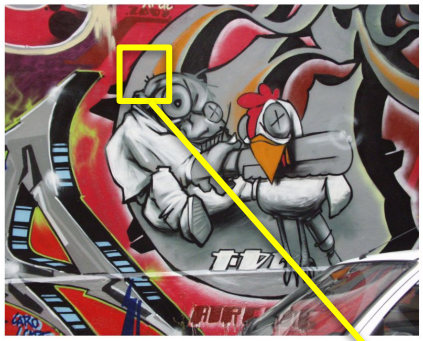
$=$

$\mathbf{a}[m, n]$

?

2D Convolution

input image: $\mathbf{x}[m, n]$



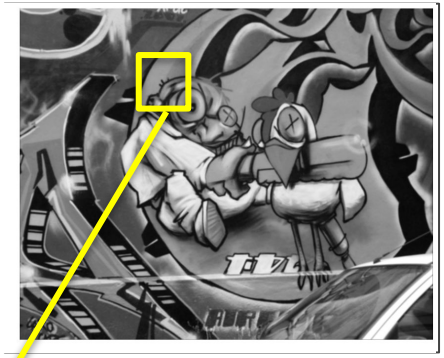
$\otimes$

filter: $\mathbf{w}[k, l]$

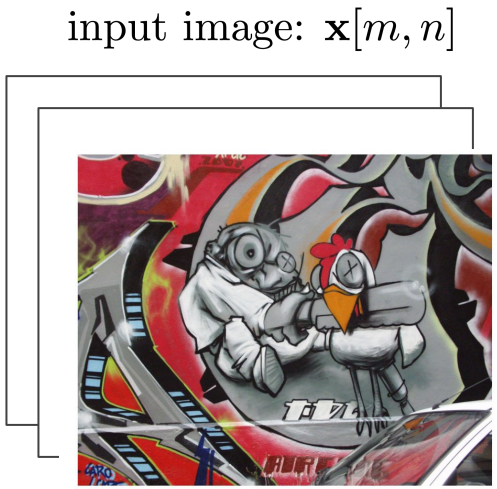
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$=$

$\mathbf{a}[m, n]$



Discussion: multiple input channels

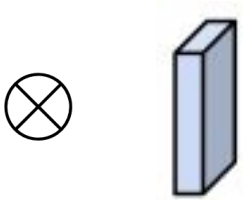


Input might have multiple “channels” (e.g. RGB image with 3 channels)

Discussion: multiple input channels

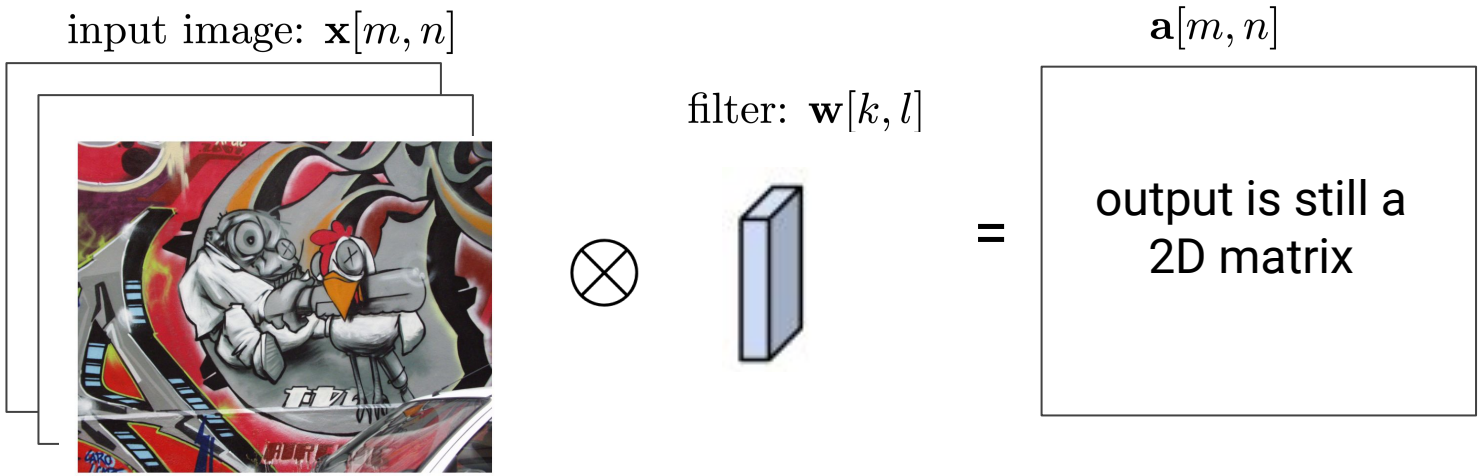
input image: $\mathbf{x}[m, n]$

filter: $\mathbf{w}[k, l]$



Input might have multiple “channels” (e.g. RGB image with 3 channels)
Filter kernel will “extend” along the third dimension to match the number of input channels

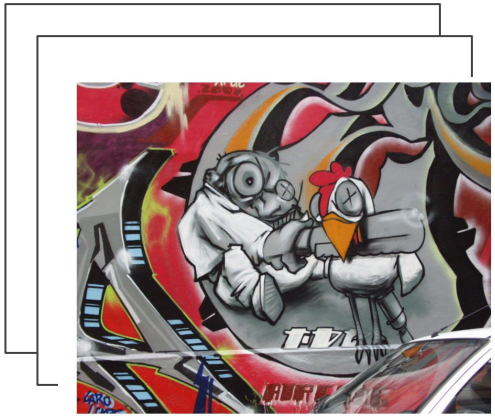
Discussion: multiple input channels



Input might have multiple “channels” (e.g. RGB image with 3 channels)
Filter kernel will “extend” along the third dimension to match the number of input channels

Discussion: learned filters

input image: $\mathbf{x}[m, n]$



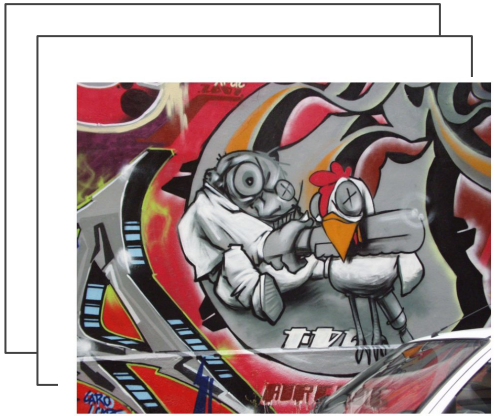
$\otimes$

filter: $\mathbf{w}[k, l]$

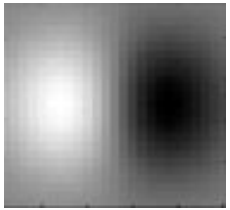
-1	0	1
-1	0	1
-1	0	1

Discussion: learned filters

Input image: I[k, l]

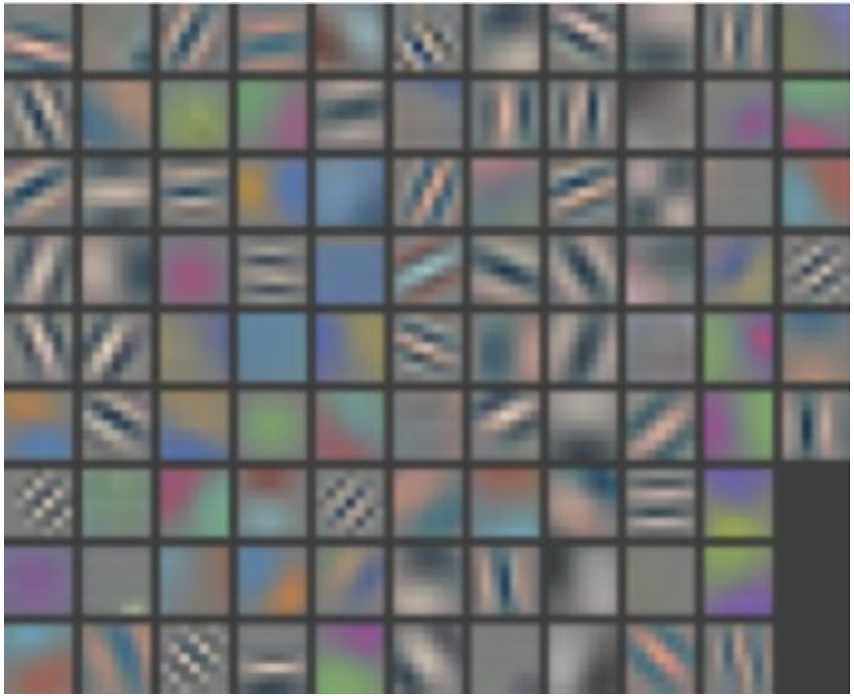


filter: g[k, l]



Learn the filters suitable for a particular image understanding task

Discussion: learned filters



Layer 1 filters

source: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Discussion: multiple layers

input image: $\mathbf{x}[m, n]$

$\mathbf{x} \otimes \mathbf{w}_1$

$\mathbf{x} \otimes \mathbf{w}_2$

$\mathbf{x} \otimes \mathbf{w}_3$

Discussion: multiple layers

input image: $\mathbf{x}[m, n]$

stack output of each filter along the "depth" dimension

Discussion: multiple layers

input image: $\mathbf{x}[m, n]$

stack output of each filter along the "depth" dimension

apply convolution to the stacked filter outputs

$\otimes \quad \mathbf{w}_1^{(2)}$

Convolution as neural network layer

- Two views:
  - instantiate neural network at every location, share the weights
  - matrix multiplication view

Convolution as neural network layer

- Two views:
  - instantiate neural network at every location, share the weights
  - matrix multiplication view

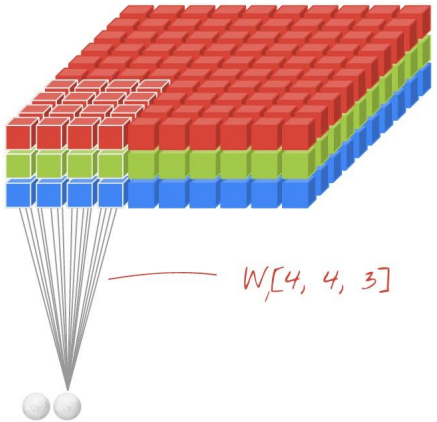


Image from "TensorFlow, Keras and deep learning, without a PhD"

Convolution as neural network layer

- Two views:
  - instantiate neural network at every location, share the weights
  - matrix multiplication view

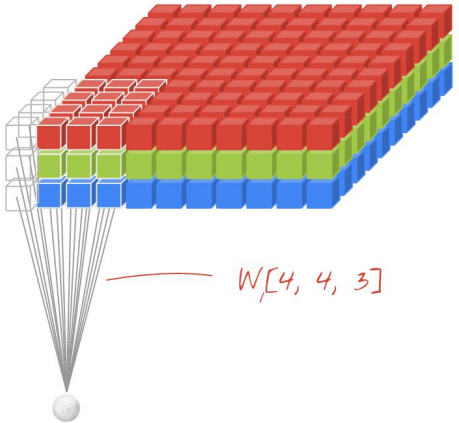


Image from "TensorFlow, Keras and deep learning, without a PhD"

Convolution as neural network layer

- Two views:
  - instantiate neural network at every location, share the weights
  - matrix multiplication view

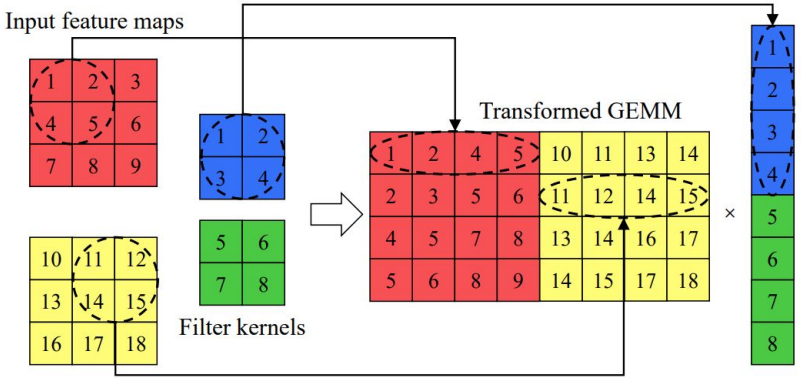


Fig. 1. The “im2col”+GEMM (explicit GEMM) method.

Image from: Zhang et al., “Evaluating Low-Memory GEMMs for Convolutional Neural Network Inference on FPGAs”

Take a quiz!