12 Convolutional Neural Networks

The idea of convolutional linear network is to replace a F.C. layer, with a convolutional layer: Is A FNN in which at least one layer is convolutional.

CONVOLUTIONAL LAYER: A layer in which the operation that we do is not matrix moltiplication but it is convolution or a cross-correlation.

The mathematical operation of convolution is easier than matrix moltiplication, it also reduce the number of parameters (it implements parameter sharing). CONVOLUTION IS BASED on some small KERNEL and the only parameter that you want to tune are the parameters of this kernel.

INTRODUCTION

Up to now we treated inputs as general feature vectors. In some cases input have special structure:

- AUDIO
- IMAGES
- VIDEOS

SIGNALS: Numerical representation of physical quantities.

Deep learning can be directly applied on signals by using suitable operators.

Note: in many cases you can find the term tensor that is just a way of denoting a multidimensional array.

Audio $\longrightarrow$ 1D tensor / 1D vector | Image $\longrightarrow$ 2D matrix / 3D tensor

$\underbrace{\text{(width, height, color channels)}}_{\text{dimensions.}}$

Given two Gutinuos functions the convolution is:

$$(x * w)(t) = \int_{a=-\infty}^{+\infty} x(a)\, w(t-a)\, da$$

Given two discrete functions:

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a)$$

Discrete limited 2D functions:

$$\boxed{(I * K)(i,j) = \sum_{m} \sum_{n} I(m,n)\, K(i-m, j-n)}$$

I = INPUT   K = KERNEL

We are interested in these functions, they are non-zero only for some limited interval.

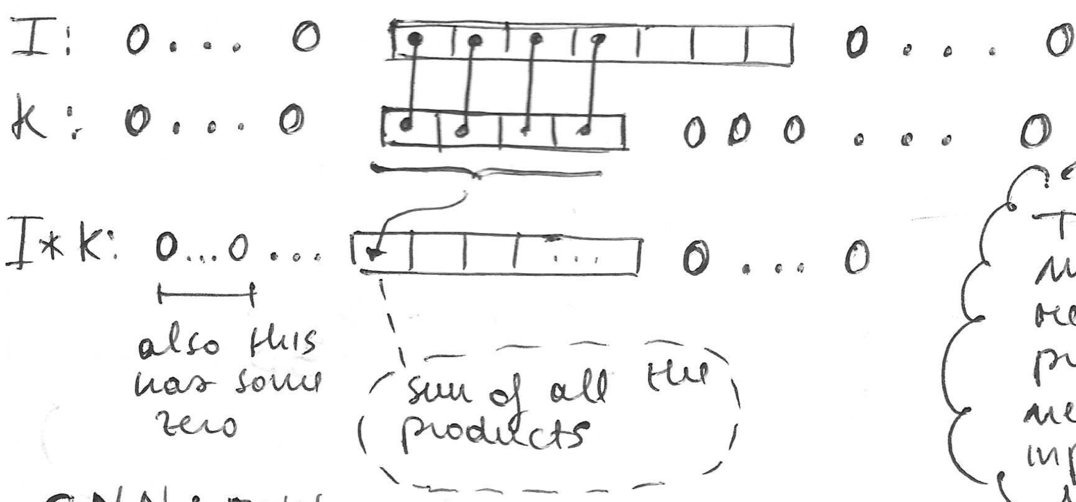
CONVOLUTION IS COMMUTATIVE!

There is another operation called CROSS-CORRELATION that is a flipping of the two function:

$$(I * K)(i,j) = \sum_{m} \sum_{n} I(i+m, j+n)\, K(m,n)$$

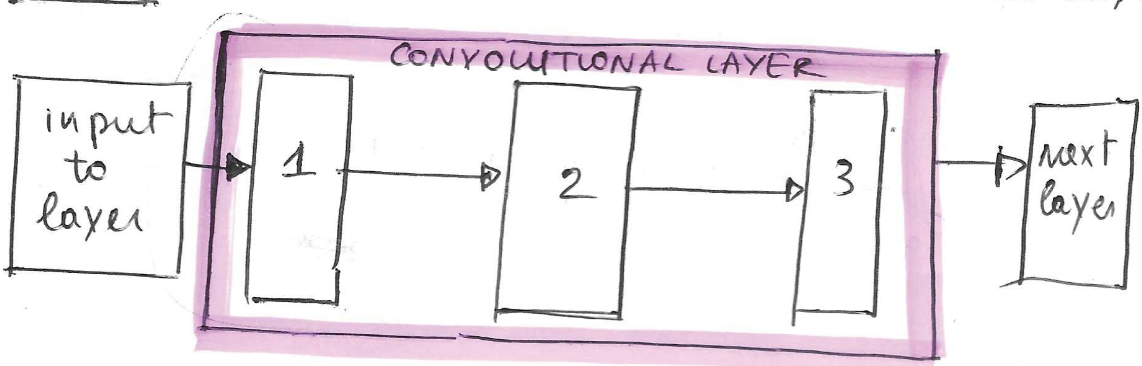
implemented in ML libraries (called convolution)

EXAMPLE of convolution:



Then you move the kernel, repeating the process with the the next section of the input function.

CNN: FNN with one or more convolutional layer.



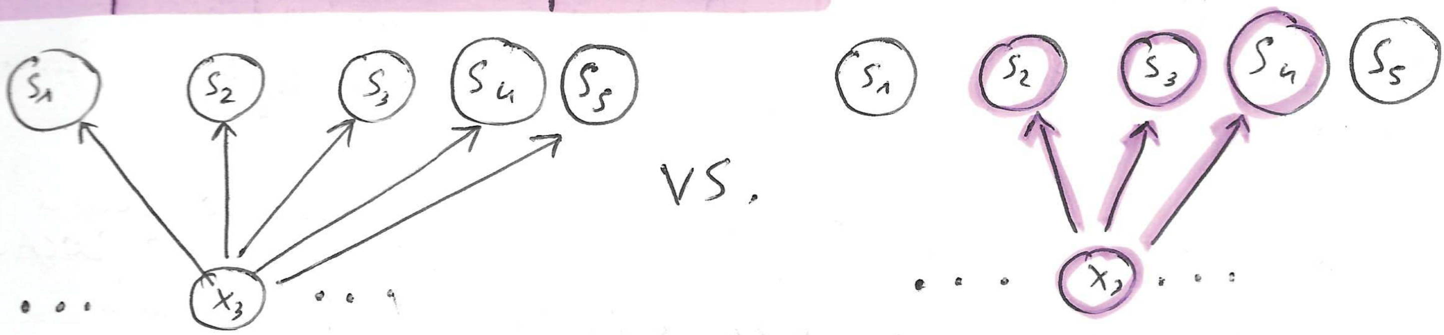
1. Convolution between input and kernel
2. non linear act. function
3. pooling

Convolution introduces two concepts to improve the general accuracy of the network and to reduce overfitting.

(1) Sparse connectivity
(2) parameter sharing.

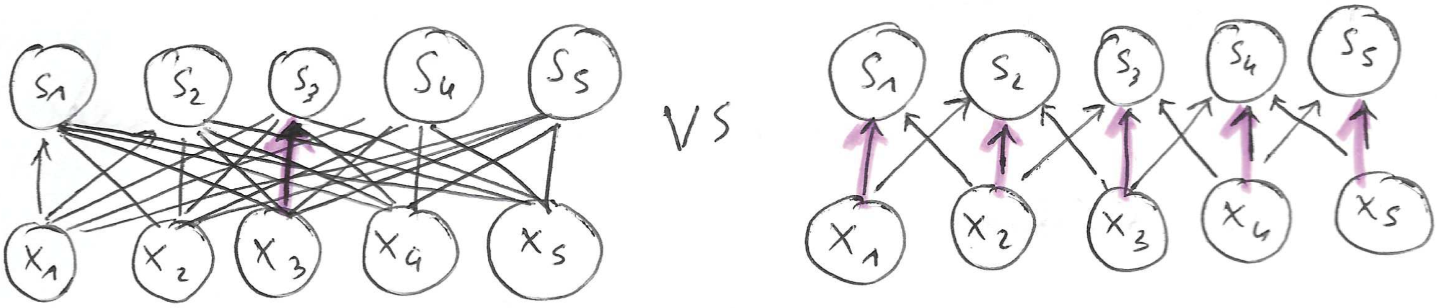
} Two important properties BETTER GENERALIZATION ERROR.

Sparse interactions/sparse connectivity: OUTPUTS DEPEND only on a few inputs. In a convolutional network one input is connected ONLY ON A SUBSET of UNITS of THE NEXT LAYER, THIS depends on the size of the kernel.



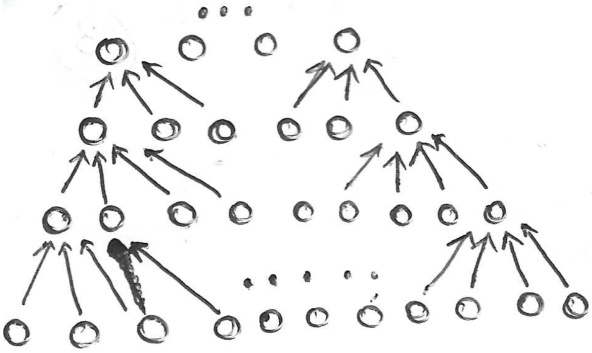
Kernel is usually much smaller than input.

In a F.C. networks all connections are independent, WHILE IN CNN we have the property that the values of the kernel are shared among all inputs. You have K PARAMETERS INSTEAD OF $m \times m$ ($K << m$). THIS IS PARAMETER SHARING.

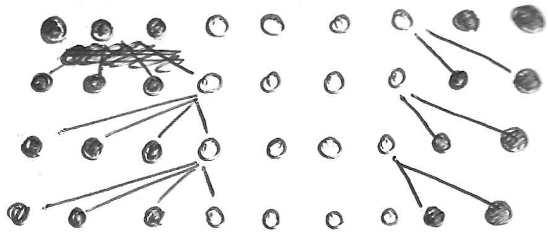


Padding → $\boxed{\text{VALID PADDING}}$: output contains only valid values (depends on kernel size)
(reduce the size of the network).

Padding → $\boxed{\text{SAME PADDING}}$: input layer is padded with zeros, output size is INDEPENDENT of kernel size.
(keep size of the network).



VS



SAME PADDING

VALID PADDING
(most used)

After the convolution, we APPLY SOME NON LINEAR TRANSFORM.
TO THE OUTPUT of THE CONVOLUTION. This operation is
the same running in F.C. layers (same activation functions)

The last stage IS POOLING that is commonly used to reduce
the size of the layer but is USED TO INTRODUCE TO LOCAL
TRANSLATION SOME **INVARIANTS** (A face of a cat can be in
whatever position).

POOLING $\xrightarrow{\text{similar}}$ TO KERNEL → BUT WE DO NOT TRAIN VALUES.

- **Max pooling**: returns the maximum value in a rectangular region.
- **Average pooling**: returns the average value in a rectangular region.

Remember! For kernel you need training!

By applying the pooling operation we can reduce the size of the layer (when applied WITH STRIDE) output.

STRIDE: after I do one operation how much I have to move when apply operator to the layer.

GENERAL IDEA

32

INPUT
We start with some images with 3 RGB channels.

apply six 5x5 kernels.

FEATURE MAPS

6 images 28x28 each generated by a different kernel

2 by 2 pooling with stride 2 → 14 x 14 six images → 64 kernels 11 x 11 64-images. → pooling ...

This is the sketch of a typical CNN

One of the first CNN was LeNet.

120kernel →

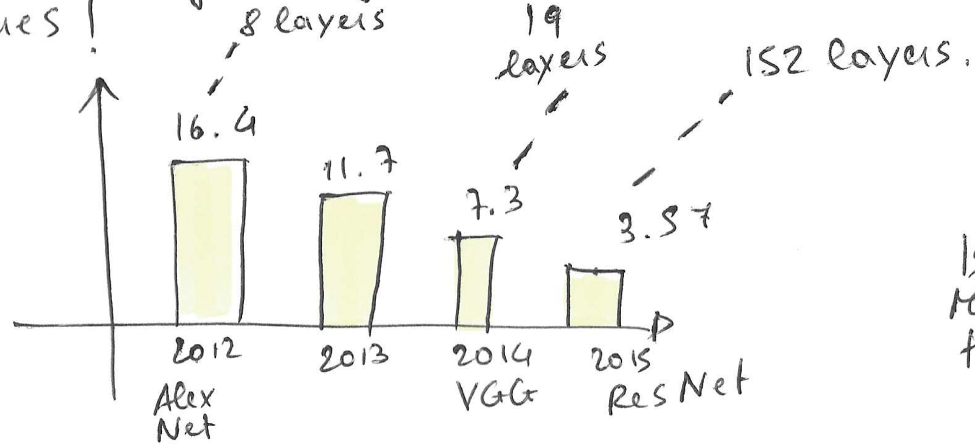
120

1x1 with depth 120

FLATTERN OPERATION
You get a linear array

→ Start some fully connected layer ... → output.

LeNet was used to recognize hand written digits, with 98% of the accuracy. AlexNet wons the competition of image classification, the dataset had 14 M images about 22 K categories! (*)



AlexNet in 2012 won after 6 days of training

IS POSSIBLE TO FIND MODELS of this network trained on imageNet dataset (*)

When you take a trained network, you can use the network to extract features, this method is CALLED DEEP FEATURE APPROACH. You may take a vector of linear numbers, such as one of the last layer and consider it as A REPRESENTATION OF THE INPUT.

You use the network as a feature extractor for images

And then use for instance SVM, and train it with these features.

Another possibility is retaining a network by considering the last layers. When you change the set of classes, the area of the network that is more affected is the one closer to the output. YOU MAY WANT TO RE-TRAIN ONLY the last layer (more efficient than training the network from scratch).