Convolutional Neural Networks (CNNs)

Carlos Valle

Departamento de Informática
Universidad Técnica Federico Santa María

cvalle@inf.utfsm.cl

November 22, 2018

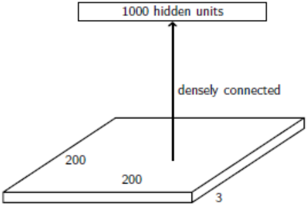
Overview

Motivation

- Inspired and highly specialized models in computer vision problems.
-

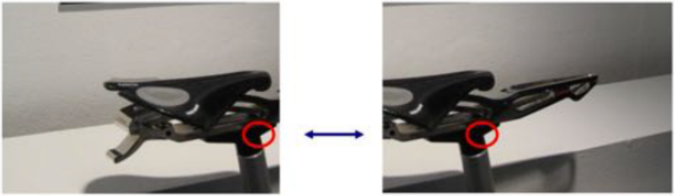
Motivation (2)

- Important problem in computer vision: high dimensionality of the patterns input that also results in networks with a very large number of parameters. Example: RGB image 200x200.
- How many parameter we would need using a feed forward neural network?



Introduction

- Another important problem in vision: highly sensitive to some other transformations, such as changes in the scale or rotation of an image.



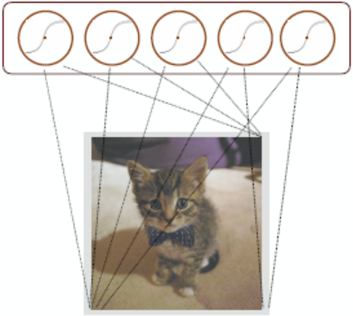
Introduction (2)

- CNNs: Implement three key ideas to deal with the dimensionality:
  1. Local / sparse connectivity.
  2. Parameter Sharing
  3. Subsampling / Pooling
- CNNs: In parallel, CNNs are designed to apply these ideas maintaining the topological structure of the input pattern.

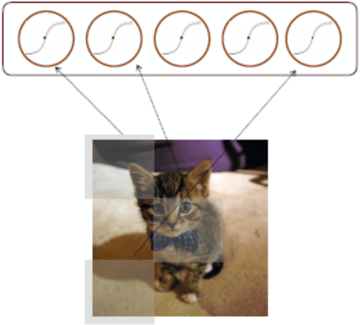
Local connectivity

- In computer vision, the neurons of a traditional MLP have an unnecessarily broad field of vision.
- It is less probable that two pixels far away compound a pattern that it is necessary to remember.
- Close pixels are highly correlated.
- We can deal with this type of "Global patterns" in superiors layers.

- Idea: Restrict the visibility of the neurons maintaining the original topological structure.



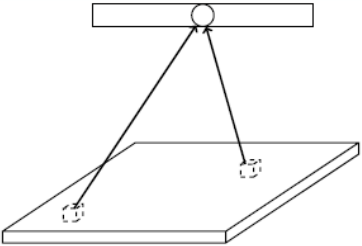
200x200 input image



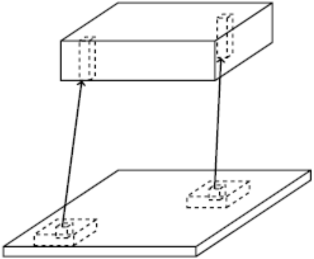
10x10 local visibility

Local connectivity (3)

- Are we reducing the number of parameters?



200x200x3 input image



10x10 local visibility

Local connectivity (4)

- Old idea: Fukushima, 1980. “The neocognitron”

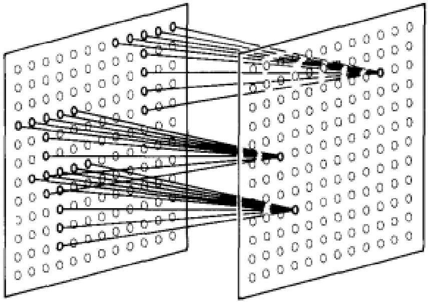
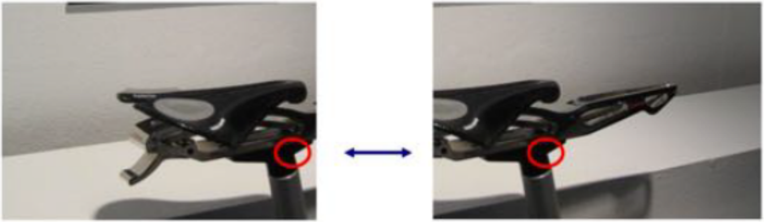


Fig. 3. Illustration showing the input interconnections to the cells within a single cell-plane

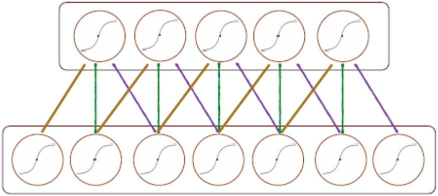
Parameter sharing

- A pattern could appear in different positions of the input image.



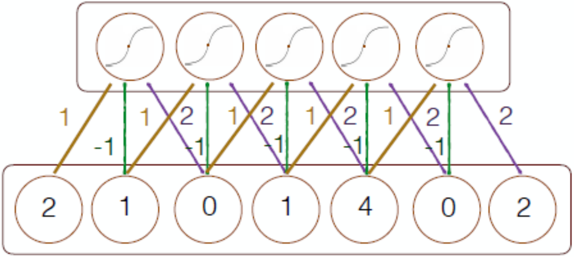
Parameter sharing (2)

- Idea 2: we tied weights between units that "look" different parts of the input pattern so that the same "detector" is applied in different positions.
- A set of units that share weights will be called a feature map.



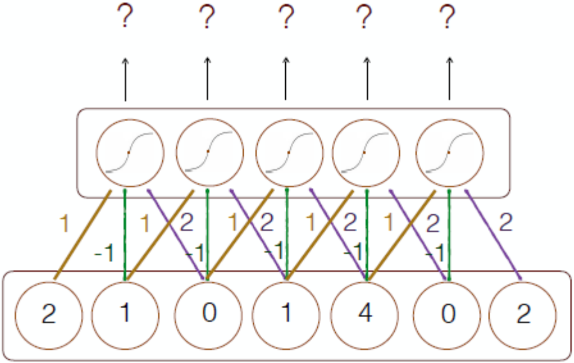
Parameter sharing (3)

- Neurons that share weights apply the same operation on the input pattern.
- This does not mean that they exhibit the same response.



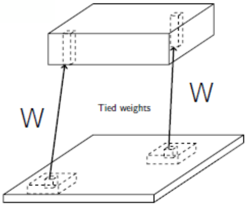
Parameter sharing (4)

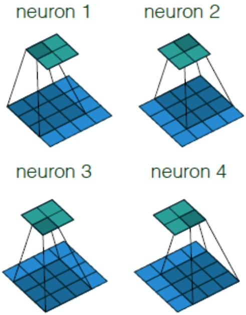
- Neurons that share weights apply the same operation on the input pattern.
- This does not mean that they exhibit the same response.



Parameter sharing (5)

- We apply this idea while maintaining the topological structure of input pattern.
- Two-dimensional case:





Parameter sharing (6)

- Example: $Input 5 \times 5$, 9 neurons, $visibility 3 \times 3$.

0	1	2
2	2	0
0	1	2

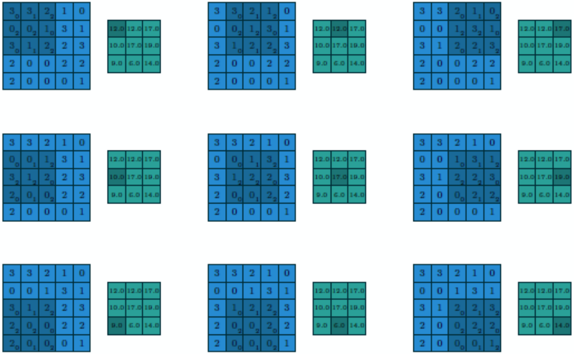
W (weights)

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Input

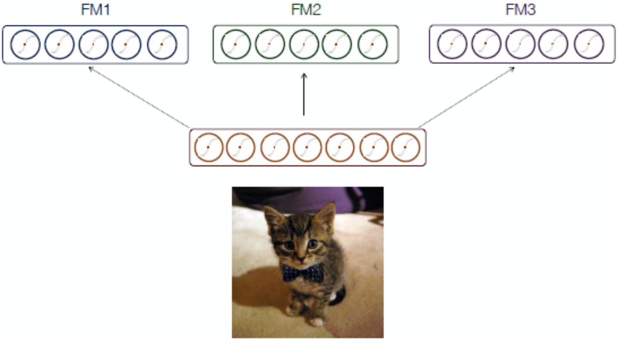
Parameter sharing (7)

- Example: $Input 5 \times 5$, 9 neurons, $visibility 3 \times 3$.



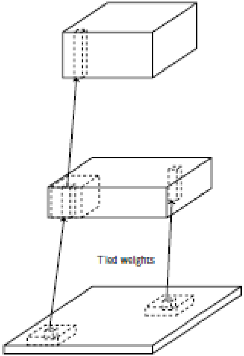
Extending this idea

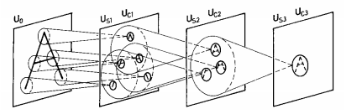
- To detect different patterns we will have in general different features maps (FMs).
- At one of these processing levels with K FMs that share weights and look at different regions of the input will be called a convolutional layer.



Extending this idea (2)

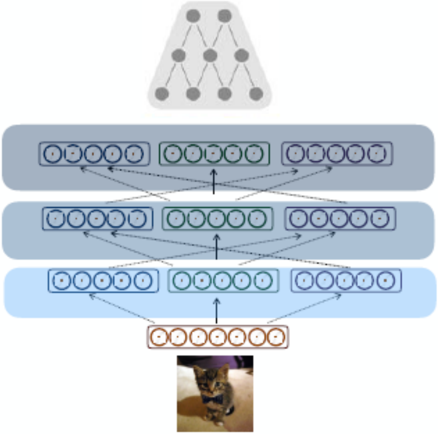
- We can use deep learning to get a model capable of "discovering" patterns with degrees growing of generality.





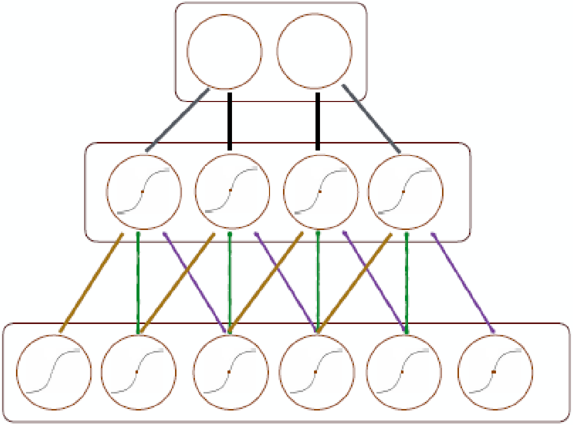
Extending this idea (3)

- After a certain number of levels, we will have sufficiently powerful features to feed an MLP.
- Problem: increasing the number of FMs we could not have reduced the number of parameters enough.



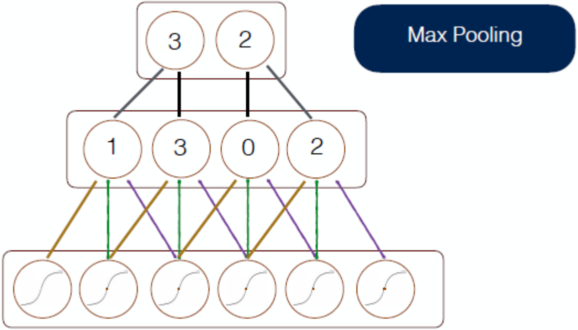
Pooling

- Idea 3: Alternating feature extraction with a strong dimensionality reduction operation (usually not adaptive).



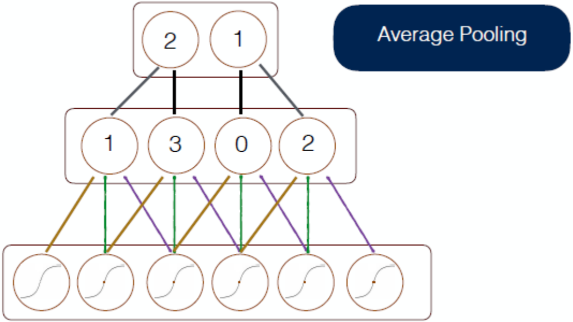
Pooling (2)

- For example, we could take either the average or maximum of a set of k adjacent neurons.



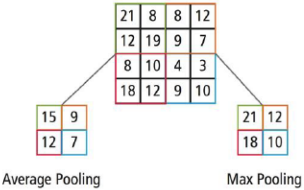
Pooling (3)

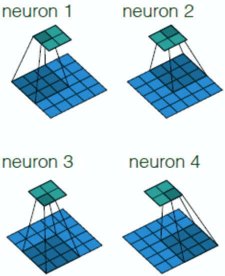
- For example, we could take either the average or maximum of a set of k adjacent neurons.



Pooling (4)

- This idea should be applied in concordance to the structure topological entry pattern.





Pooling (5)

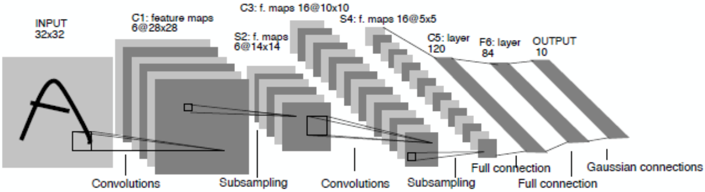
- Another desirable effect: Neurons of higher layers can process detect patterns that involve a larger area of the image.

Pooling (6)

- In all cases, pooling helps to make the representation become approximately invariant to small translations of the input.
- Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.
- Invariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is.

CNN Architecture

- A CNN (basic) is essentially an FNN made up of two parts: a initial feature extractor, which is composed of a certain number of convolutional layers and pooling, plus a final predictor implemented typically through a classic MLP.



Convolutions

- The idea of convolution formally represents the transformation that suffers an input instance when going through one of convolutional layers.
- Convolution is one of the fundamental concepts in digital signal processing because it allows to describe the response of a system given a signal.

Convolutions

- Definition (Continuous): Given two real-time signals $x(t)$ and $k(t)$

$$s(t) = (x * k)(t) = \int x(a)k(t-a)da.$$

- Definition (Discrete): Given two discrete signals $x(i)$ and $k(i)$

$$s(i) = (x * k)(i) = \sum_{p=-\infty}^{p=\infty} x(p)k(i-p).$$

Convolutions

- Why are they so important in digital signal processing?
- Suppose we know the answer $k(t - t_0)$ of the system to an impulse (instantaneous shock) at time t_0.
- Then, it is possible to know the response of the system to any signal $x(t)$ by the convolution between $x(t)$ and $k(t)$.

Convolutions

- If $x(t)$ were an impulse at time t_0, we obtain:

$$s(t) = (x * k)(t) = \int \delta(a - t_0)k(t - a)da = k(t - t_0)$$

- Impulse response: $k(t)$ can be interpreted as the answer of the system t units of time after a "shock".
- If $x(t)$ is an arbitrary (integrable) signal, we can decompose it as sum of impulses:

$$x(t) = \sum_{i=-\infty}^{i=\infty} \alpha_i \delta(t - t_i)$$

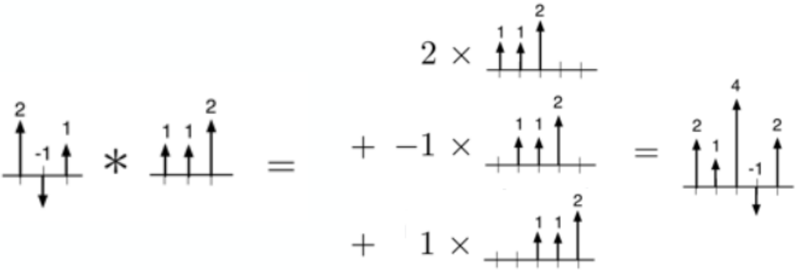
Convolutions

- We have:

$$
\begin{aligned}
s(t) = (x * k)(t) &= \int \sum_{i=-\infty}^{i=\infty} \alpha_i \delta(a - t_0) k(t - a) da \\
&= \sum_{i=-\infty}^{i=\infty} k(t - t_0)
\end{aligned}
$$

- Then, it is possible to know the response of the system to any signal $x(t)$ if we know $k(t)$, via convolutions.

Convolutions



Convolutions properties

- Conmutativity:
$$(x * k) = (x * k)$$
- Associativity:
$$x * (k_1 * k_2) = (x * k_1) * k_2$$
- Distributivity:
$$(x * k_1) + (x * k_2) = x * (k_1 + k_2)$$
- Linearity:
$$(a \cdot x + b \cdot y) * k = a(x * k) + b(y * k)$$
- Indeed, if $x(i)$ has finite support, there exists a matrix K such as
$$x * k = Kx$$

Convolutions properties

- Example:

$$(2,-1,1) * (1,1,2) = \underbrace{\begin{pmatrix} 1 & & \\ 1 & 1 & \\ 2 & 1 & 1 \\ & 2 & 1 \\ & & 2 \end{pmatrix}}_{\text{Circulant matrix}} \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$

Convolutions properties

- A convolution is an efficient representation way.

$$x * k = Kx$$

- If k is unknown: 3 free parameters.
- If K is unknown: 15 free parameters.
- In general, considering that x, k can be of greater dimension, learning K as in traditional neural networks would require a number of parameters exponentially larger than those we need working with k.

Correlations

- Definition (Continuous): Given two real-time signals $x(t)$ and $k(t)$

$$s(t) = (x \star k)(t) = \int x(t+a)k(a)da.$$

- Definition (Discrete): Given two discrete signals $x(i)$ and $k(i)$

$$s(i) = (x \star k)(i) = \sum_{p=-\infty}^{p=\infty} x(i+p)k(p).$$

Convolutions and correlations

- Link between correlation and convolution:

$$x(t) * k(t) = x(t) \star k(-t)$$

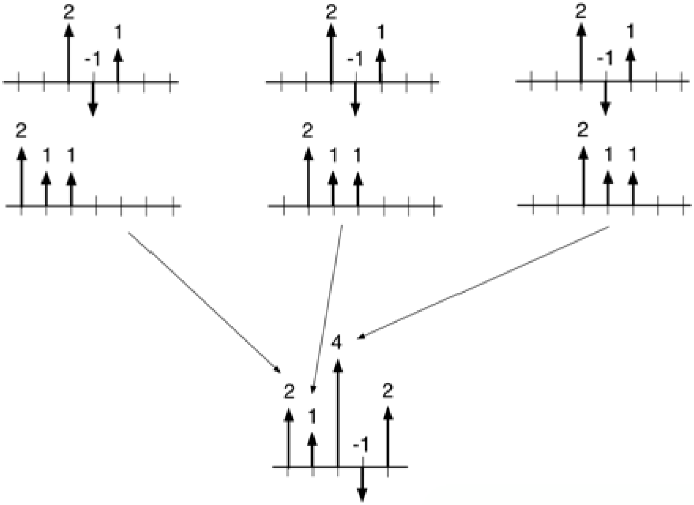
- Unlike convolution, the correlation is neither commutative nor neither associative
- In particular:

$$(x \star k1)\star \neq x \star (k1 \star k2)$$

$$(x * k1)* \neq x * (k1 * k2)$$

- Very important in practice

Convolutions and correlations



Convolutions (2D)

- Definition (Continuous): Given two real-time signals $x(t)$ and $k(t)$

$$s(t) = (x * k)(t, w) = \int \int x(a)k(t - a, w - b)dadb.$$

- Definition (Discrete): Given two discrete signals $x(i)$ and $k(i)$

$$s(i) = (x * k)(i, j) = \sum_{p,q=-\infty}^{\infty} x(p, q)k(i - p, j - q).$$

- We have that:

$$x(s, t) * k(s, t) = x(s, t) * k(t, -s)$$

Convolutions (2D) (2)

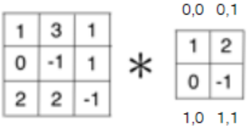
$$
\begin{array}{|c|c|c|}\hline 1 & 3 & 1 \\ \hline 0 & -1 & 1 \\ \hline 2 & 2 & -1 \\ \hline \end{array}
\;*\;
\begin{array}{cc} 0,0 & 0,1 \\ \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & -1 \\ \hline \end{array} & \\ 1,0 & 1,1 \end{array}
$$

$$
1 \times \begin{array}{|c|c|c|c|}\hline 1 & 3 & 1 & \\ \hline 0 & -1 & 1 & \\ \hline 2 & 2 & -1 & \\ \hline & & & \\ \hline \end{array}
$$

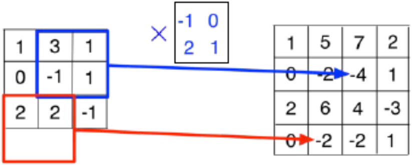
$$
+\;2 \times \begin{array}{|c|c|c|c|}\hline & 1 & 3 & 1 \\ \hline & 0 & -1 & 1 \\ \hline & 2 & 2 & -1 \\ \hline & & & \\ \hline \end{array}
\;=\;
\begin{array}{|c|c|c|c|}\hline 1 & 5 & 7 & 2 \\ \hline 0 & -2 & -4 & 1 \\ \hline 2 & 6 & 4 & -3 \\ \hline 0 & -2 & -2 & 1 \\ \hline \end{array}
$$

$$
+\,-1 \times \begin{array}{|c|c|c|c|}\hline & & & \\ \hline & 1 & 3 & 1 \\ \hline & 0 & -1 & 1 \\ \hline & 2 & 2 & -1 \\ \hline \end{array}
$$

Convolutions (2D) (3)



using a flipped rotated kernel



Padding

- One essential feature of any convolutional network implementation is the ability to implicitly zero-pad the input in order to make it wider.
- Without this feature, the width of the representation shrinks by one pixel less than the kernel width at each layer.
- Valid convolution: all pixels in the output are a function of the same number of pixels in the input, so the behavior of an output pixel is somewhat more regular.
- It may drastically limits the number of convolutional layers that can be included in the network.

Padding

- Full convolution: Adding enough zeroes to the input in order to compute every non-empty convolution when at least one element from the kernel touches any element from the image.
- In this case, the output pixels near the border are a function of fewer pixels than the output pixels near the center.
- This can make it difficult to learn a single kernel that performs well at all positions in the convolutional feature map.

Why are so important in digital image processing?

$*$

0	-1	0
-1	4	-1
0	-1	0

Laplacian Filter

Why are so important in digital image processing?

$*$

0	1	0
1	4	1
0	1	0

Gaussian Filter

Why are so important in digital image processing?

-1	0	+1
-2	0	+2
-1	0	+1

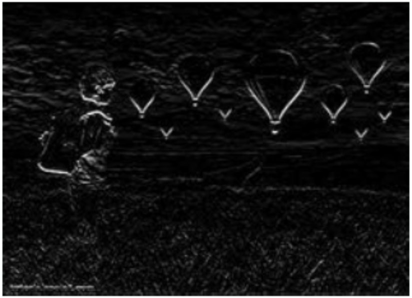
Gx

Sobel Vertical Filter

Why are so important in digital image processing?

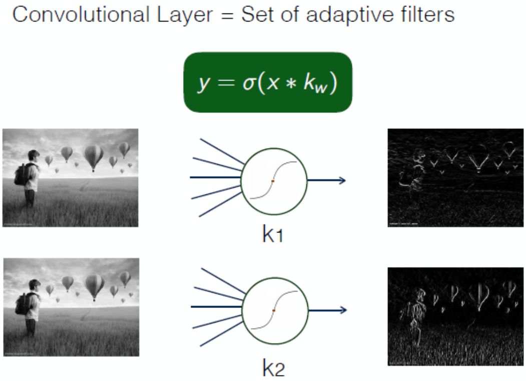
+1	+2	+1
0	0	0
-1	-2	-1

Gy



Sobel Horizontal Filter

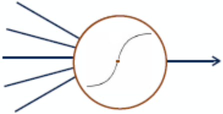
CNNs



- Usually σ is a ReLu activation function.
- We can use correlation: $\sigma(x \star \tilde{k})$,
- where $\tilde{k} = \text{rotate}(k)$.

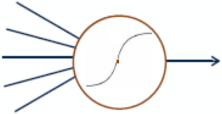
CNNs

Traditional ANN



$\sigma(w^T x)$

FMs of a CNN



$y = \sigma(x * k_w)$

CNNs

- In general, a CNN is applied on one-dimensional or two-dimensional patterns or three-dimensional.
- In the latter case (e.g. RGB images), we can think the input as a collection of two-dimensional signal of $I \times J$ form indexed in K channels.
- In this case, the transformation that allows the nth FM of a layer is designed as

$$y_n = \sigma\left(\sum_{k=1}^{K} x_k * k_{W^{k,n}}\right)$$

- Thus, a different filter is applied to each input channel, and the signal aggregate is passed through non-linearity

Any questions?