PHY654

Machine learning (ML) in particle physics

Swagata Mukherjee • IIT Kanpur

9th September 2024

Useful links

https://playground.tensorflow.org/

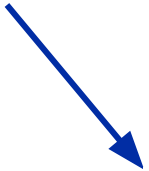
Uproot https://github.com/scikit-hep/uproot3

A convolution operation converts all the pixels in its receptive field into a single value.

If you apply a convolution to an image, generally you will be decreasing the image size and bringing all the information in the receptive field together into a single pixel.

Filter for horizontal edge detector

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



w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Instead of hand-picking these numbers, they can be treated as parameters that can be learned.

Padding

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

2 problems

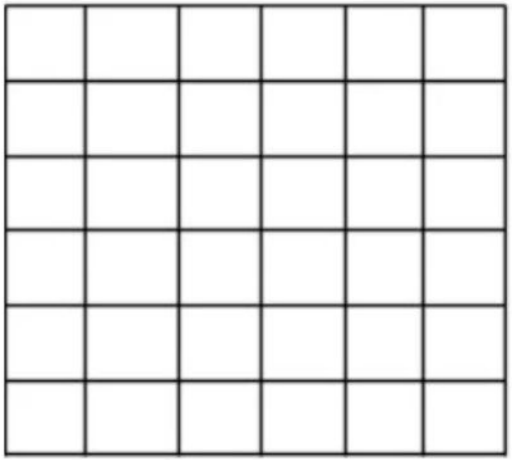
- Convolution shrink the image. So convolution can be done only a few times.
- Pixels at corners are used less.
  - May be we are not utilizing corner information enough.

Solution? → Before using convolution, pad the image.

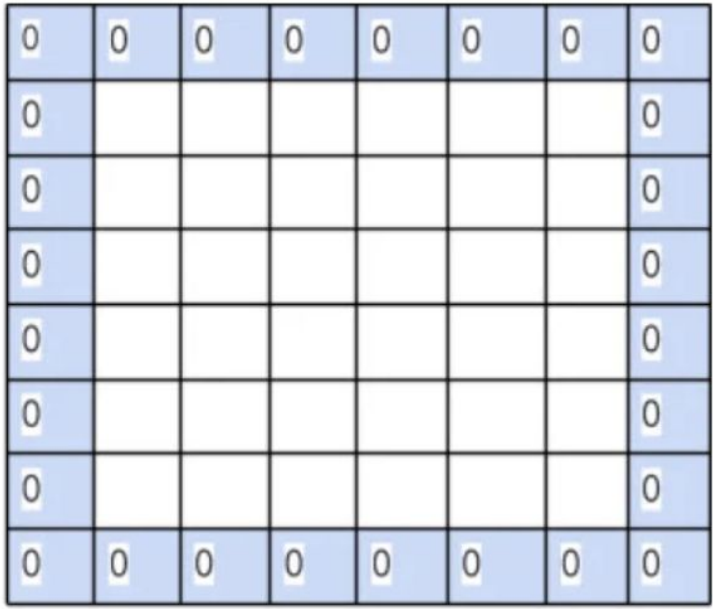
p=padding amount, If p=1 then 6x6 image becomes 8x8.

After convolution with 3x3 filter, the output is 6x6

You can pad with values=0



6x6 image



6x6 image with 1 layer of zero padding

Valid and same convolution

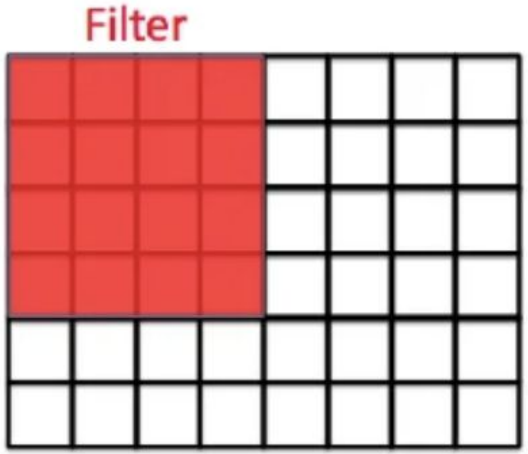
Valid → no padding

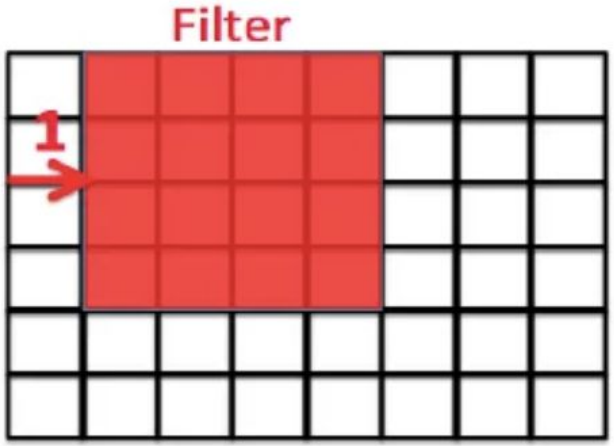
Same → Output size same as input size

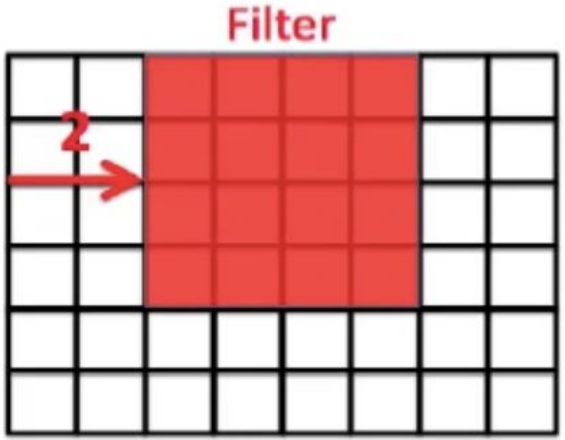
Strided convolution

What we discussed is stride = 1

If we move the filter by 2 units, then stride = 2







$n \times n$ image $\quad f \times f$ filter

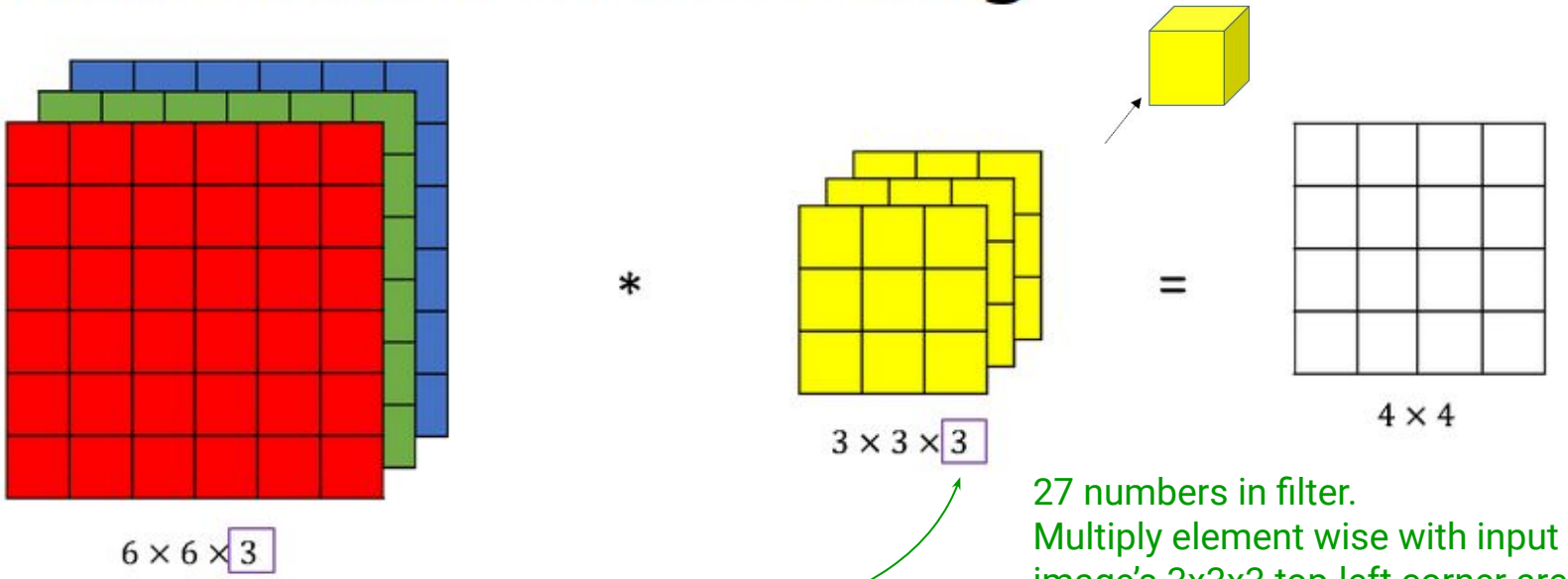
padding p $\quad$ stride s

Output

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

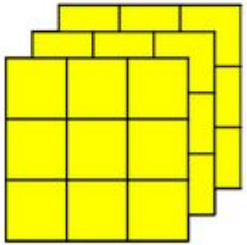
RGB image = 3D image

Convolutions on RGB image



These 2 numbers should be equal.
This is the number of channels (or depth).

27 numbers in filter.
Multiply element wise with input image's 3x3x3 top-left corner area and add up these 27 values. That's the value of top-left cell in output.



$3 \times 3 \times 3$

27 numbers

Red color vertical edge detector

R

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

G

0	0	0
0	0	0
0	0	0

B

0	0	0
0	0	0
0	0	0

→ $3 \times 3 \times 3$

Vertical edge detector for all 3 channels

R

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

G

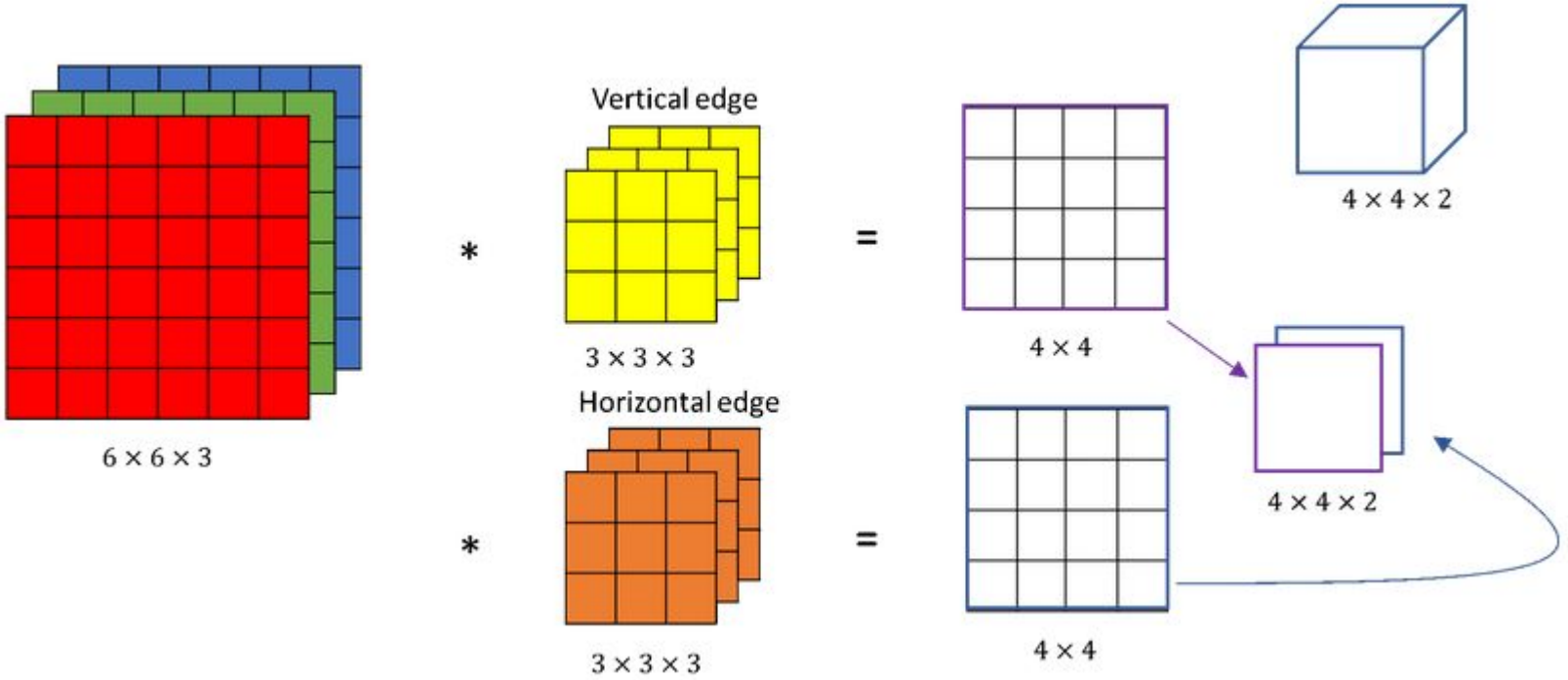
1	0	-1
1	0	-1
1	0	-1

B

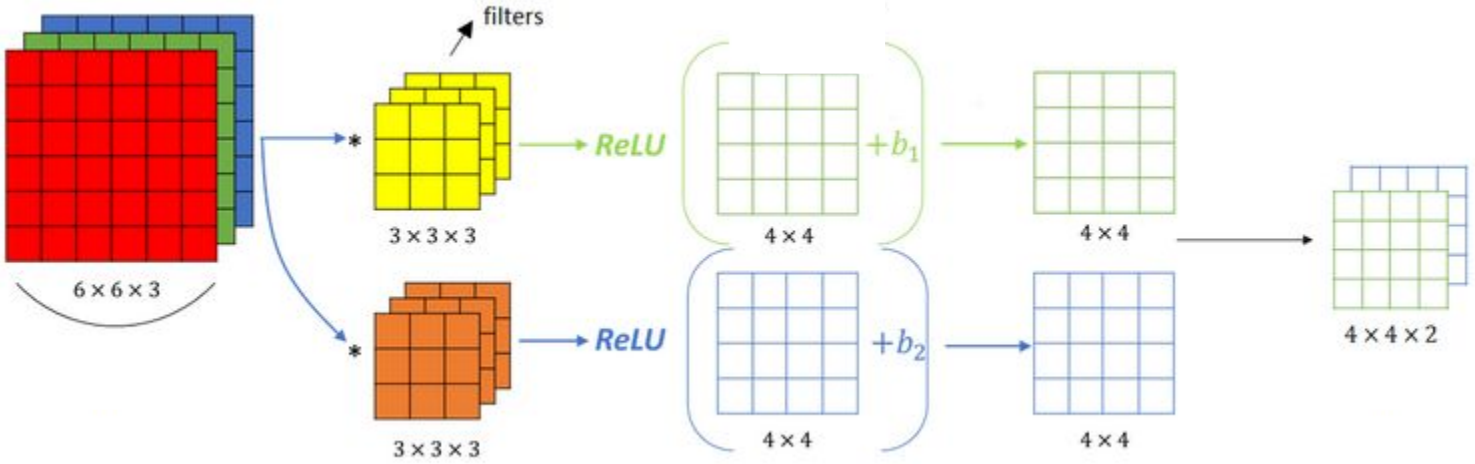
1	0	-1
1	0	-1
1	0	-1

→ $3 \times 3 \times 3$

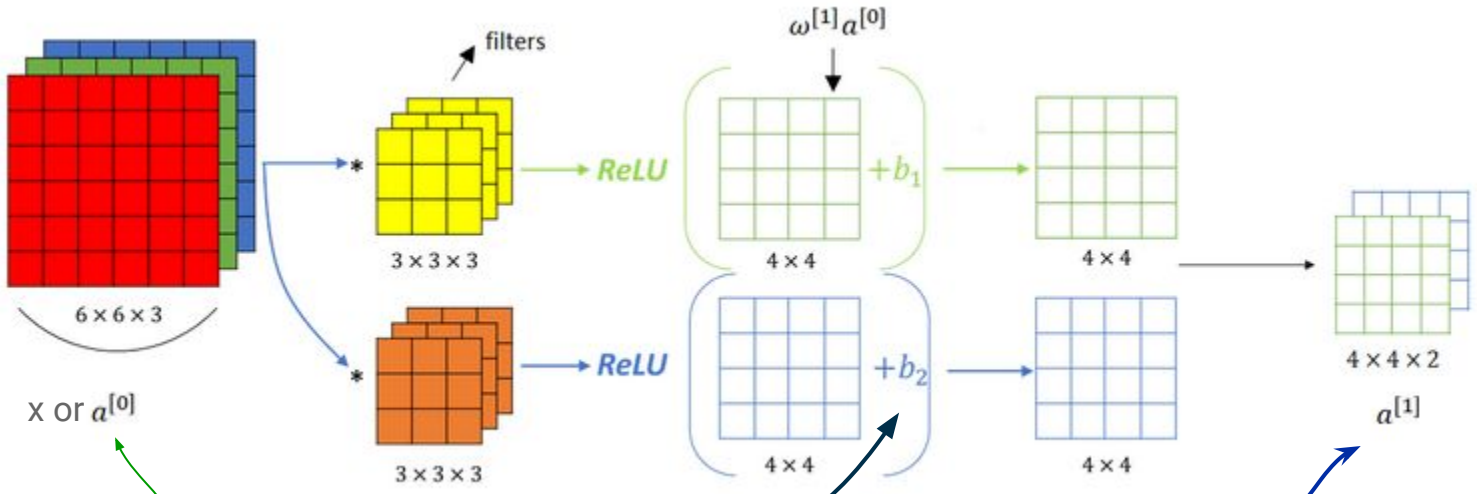
Multiple filters



One layer of CNN



One layer of CNN



Filters play a role of $w^{[1]}$

$$Z^{[1]} = \omega^{[1]}a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{(z^{[1]})}$$

No matter how big the image is, we have fixed number of parameters and much smaller number of parameters. Thus CNN will be less prone to overfitting.

Notation for layer l

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_{H/W}^{[l]} = \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} - 1$$

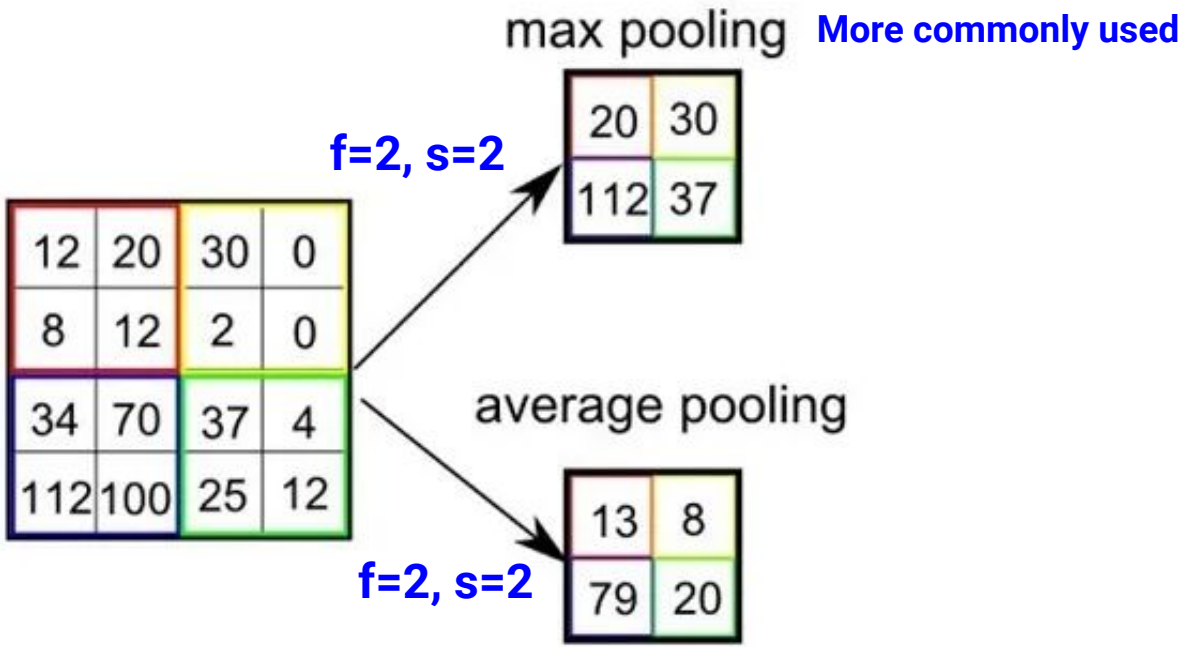
Each filter is $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Types of layers in a realistic CNN

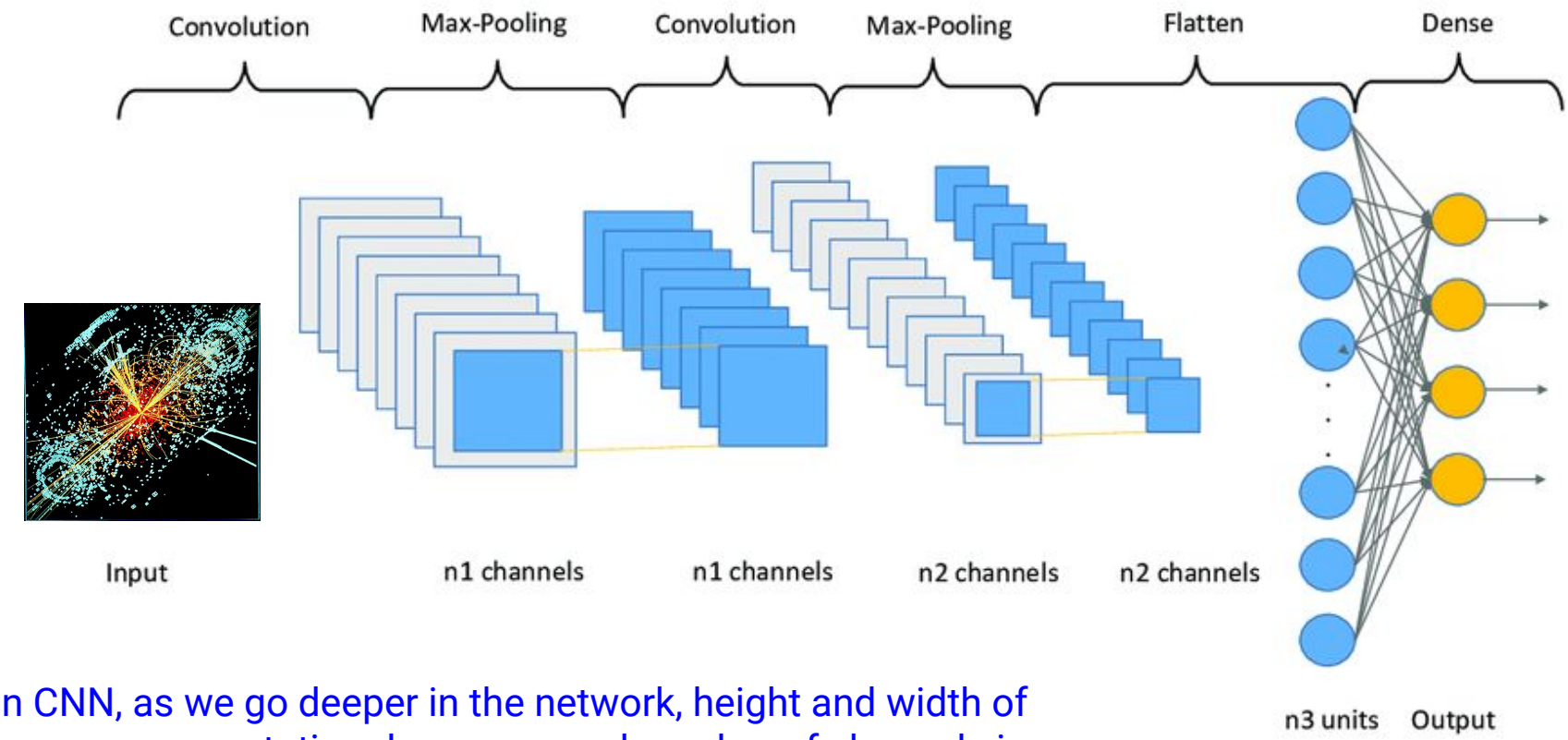
- Convolution (conv)
- Pooling (pool)
- Fully connected (FC)

Pooling

Two hyperparameters f and s, but **no parameters to learn**.



A typical CNN example



In CNN, as we go deeper in the network, height and width of image-representation decreases and number of channels increases.