Fundamentals of Convolutional Neural Networks (CNN) Part I

ivanovitch.silva@ufrn.br
@ivanovitchm

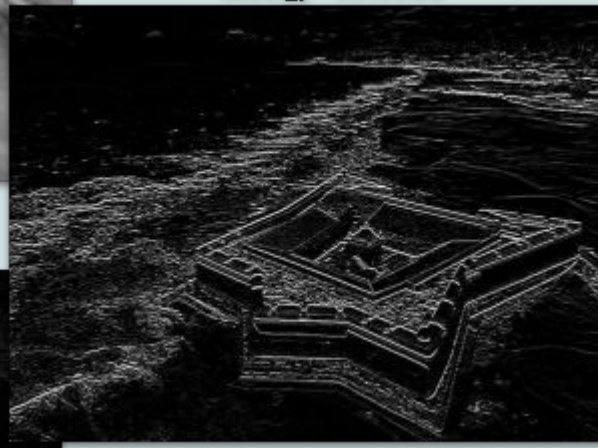
UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DCA

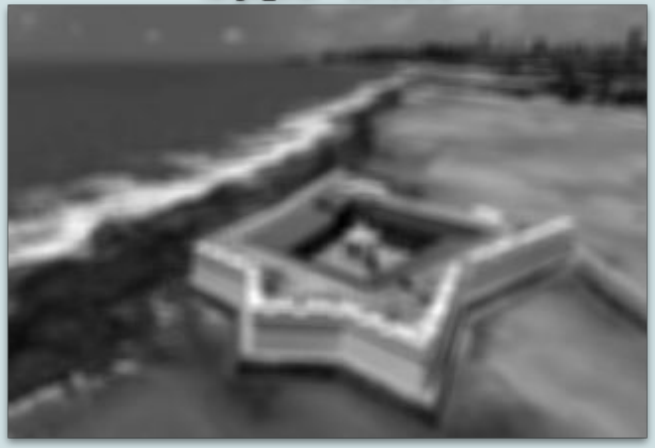
Original

small_blur - convolve

sobel_y - convolve

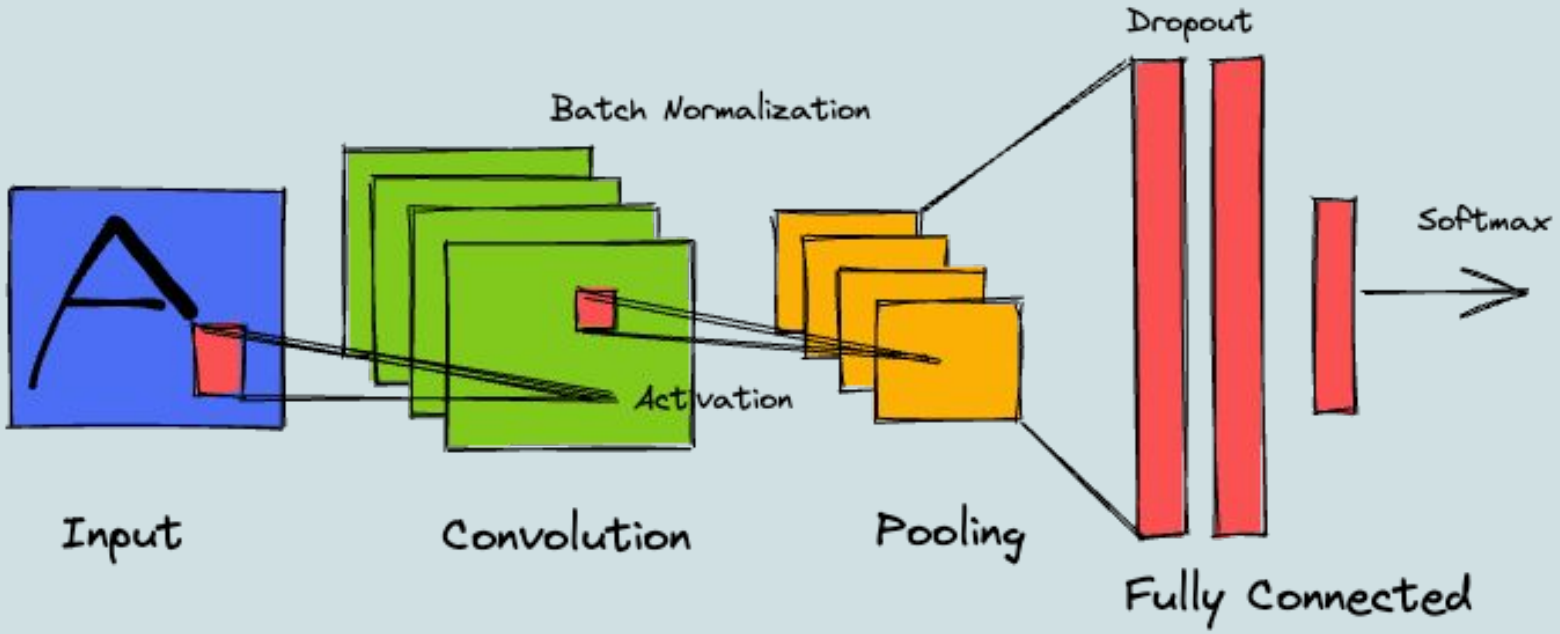


large_blur - convolve



laplacian - convolve

CNN Building Blocks

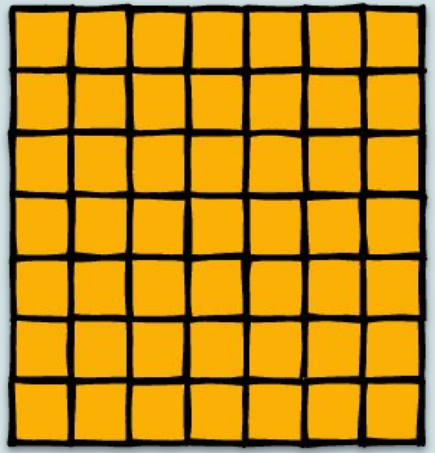


Convolution Explained in Code

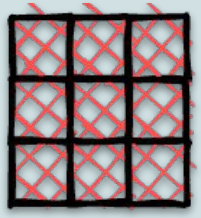
```
import numpy as np
image = np.array((
    [1,2,3],
    [4,5,6],
    [7,8,9]
))
kernel = np.array((
    [-1,0,1],
    [-2,0,2],
    [-1,0,1]
))

conv = image*kernel
conv
array([[-1,  0,  3],
       [-8,  0, 12],
       [-7,  0,  9]])
```

```
np.sum(conv)
8
```

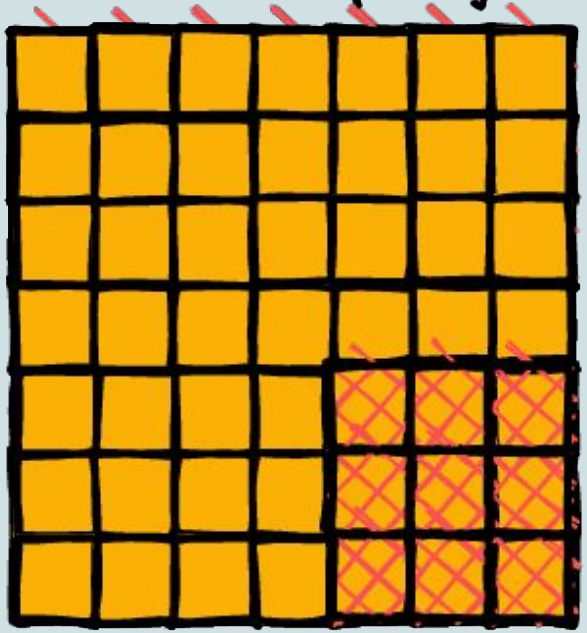


Input (7x7)

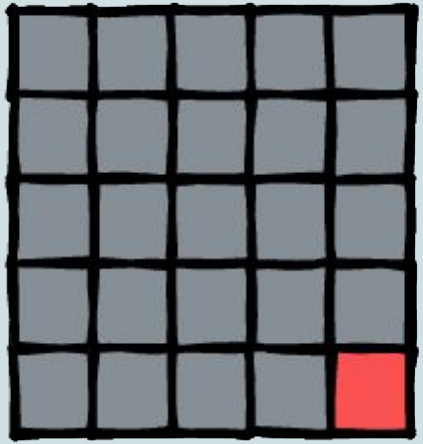


kernel (3x3)

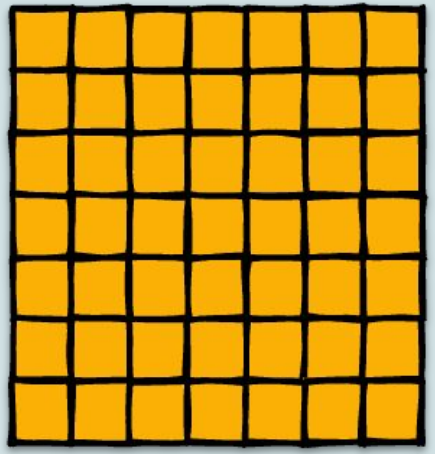
Stride = 1
padding = 0



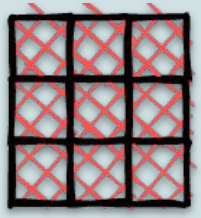
Convolution (input * kernel)



feature map (5x5)



Input (7x7)



kernel (3x3)

Padding

Valid (0)

Same(1)

Summary of Convolutions

n x n input

k x k kernel

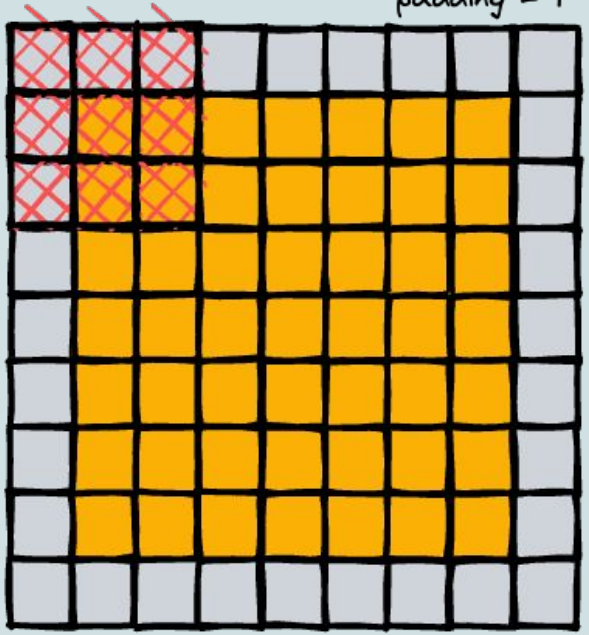
padding p

stride s

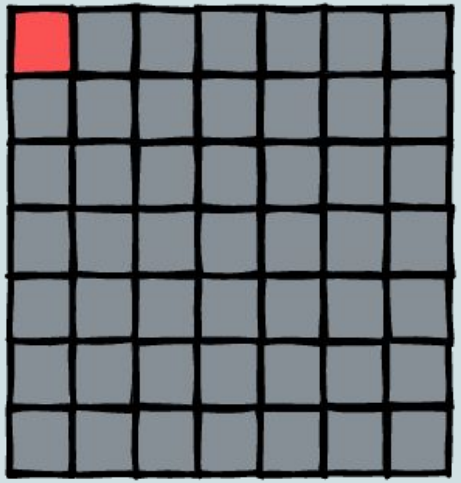
$$\left\lfloor \frac{n + 2p - k}{s} + 1 \right\rfloor$$

Stride = 1

padding = 1

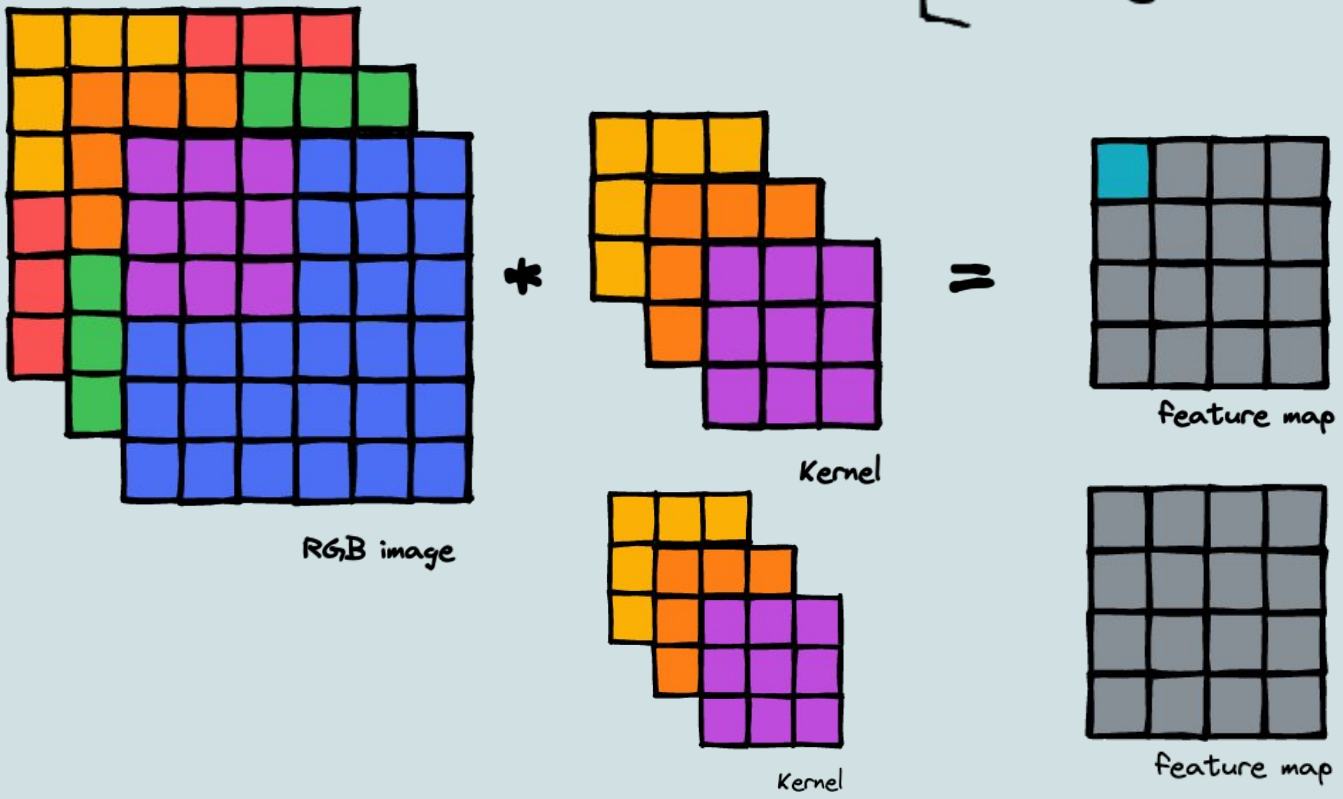


Convolution (input * kernel)

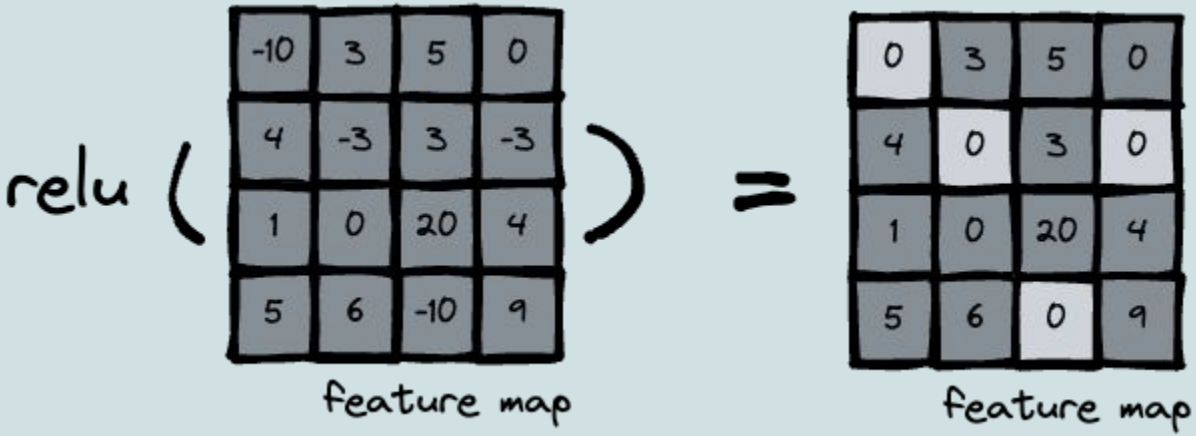


feature map (7x7)

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



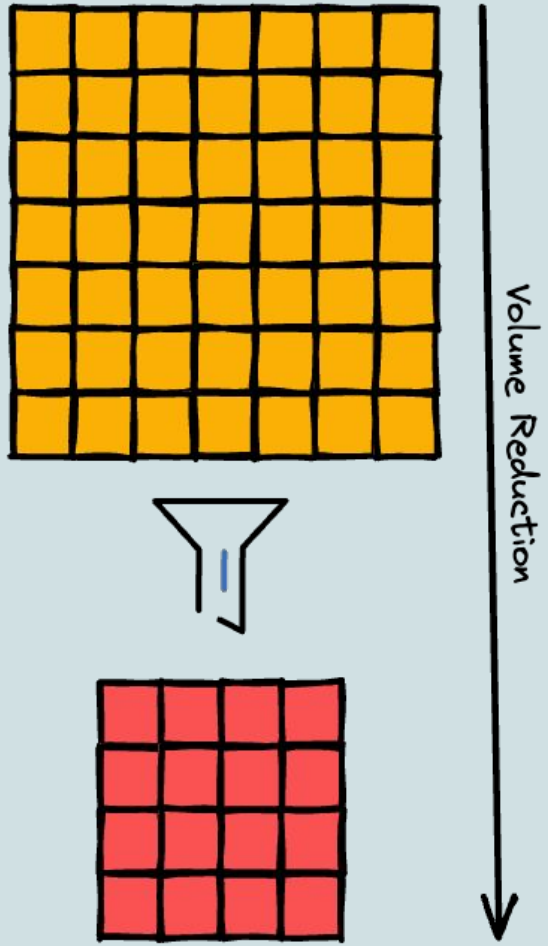
Activation Function



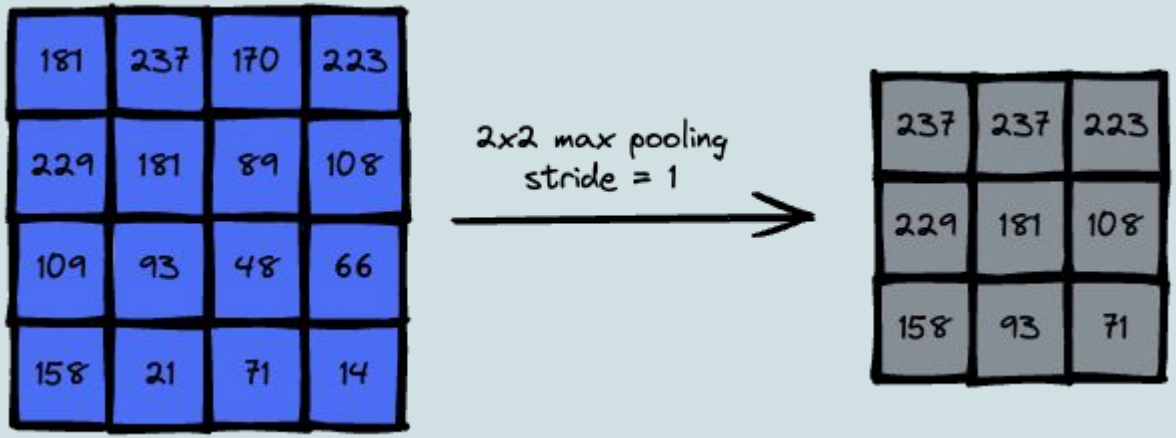
Pooling Layers

Two methods to reduce size of volume in CNN

1. CONV layers, typically with stride S > 1
2. Pooling layers
    a. Kernel (only a function and without weights, S > 2)

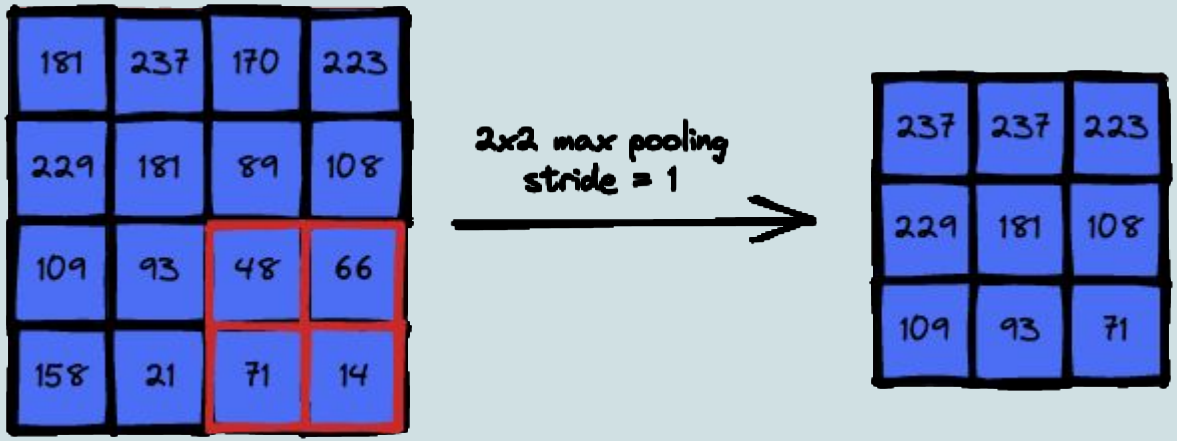


Pooling Layers

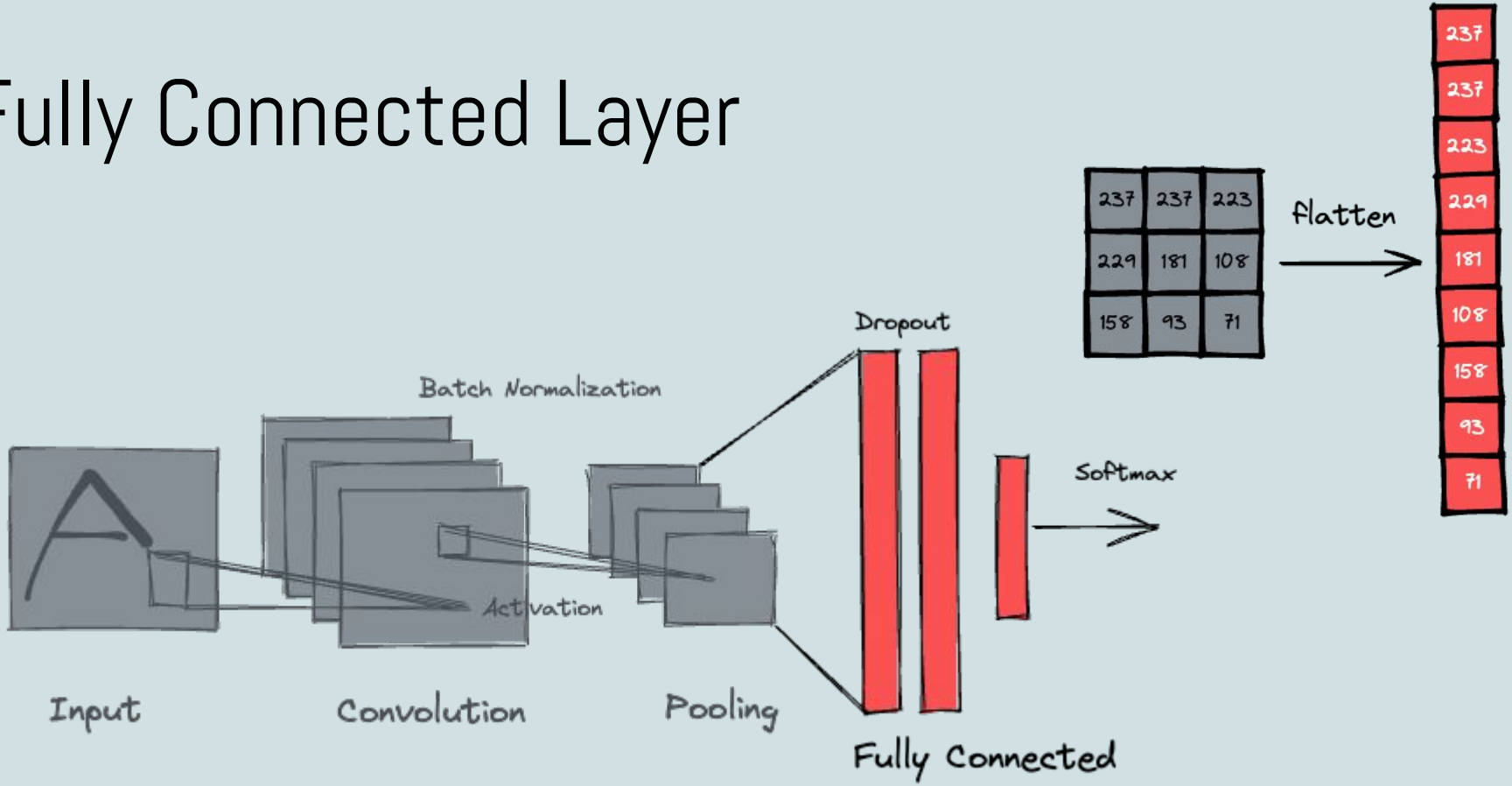


$$\left\lfloor \frac{n + 2p - k}{s} + 1 \right\rfloor$$

Pooling Layers

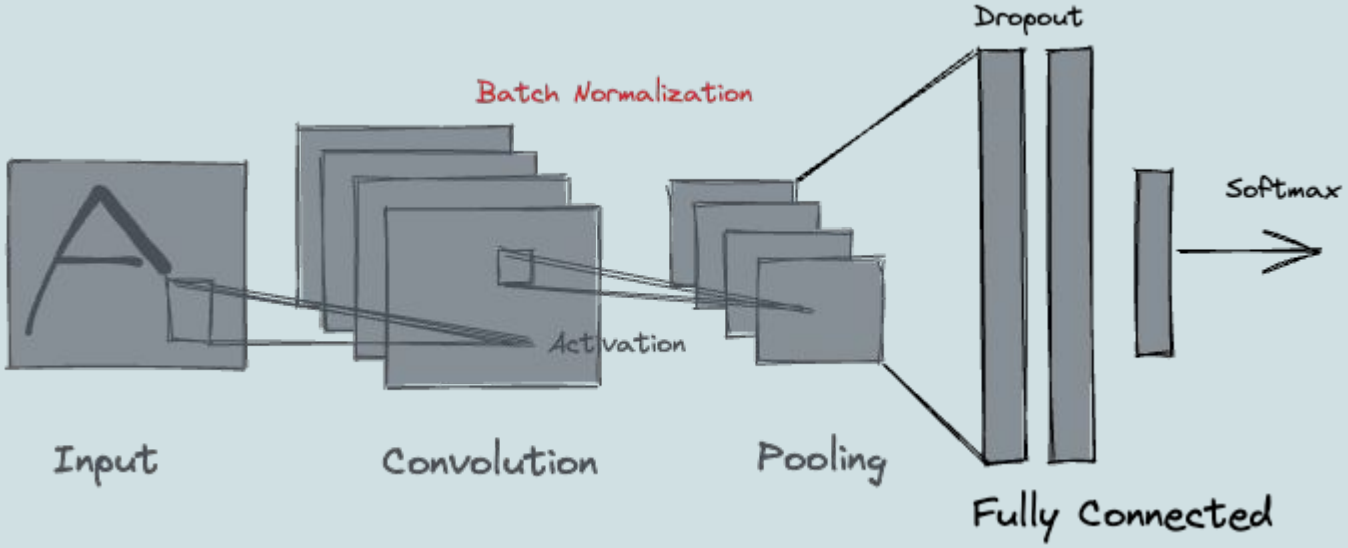


Fully Connected Layer



CONV => RELU => POOL => ... => FC

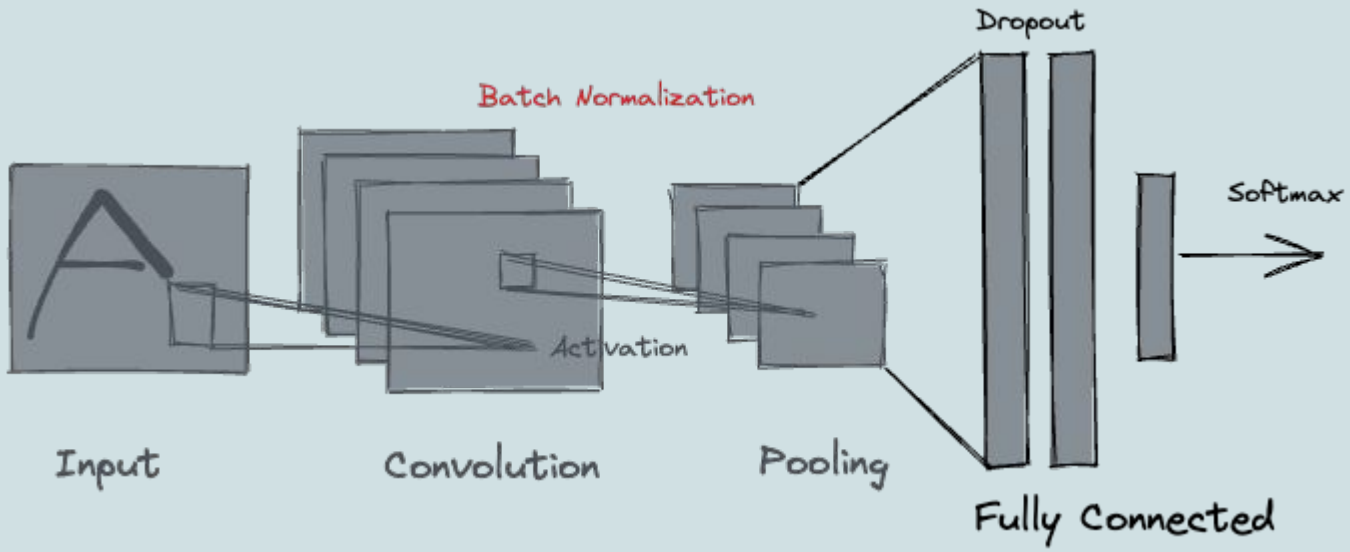
Batch Normalization



Accelerate the training process
Lightweight regularization

```
z = np.random.rand(3,3)
mean, std = np.mean(z), np.std(z)
norm = (z - mean)/(std + 1e-7)
```

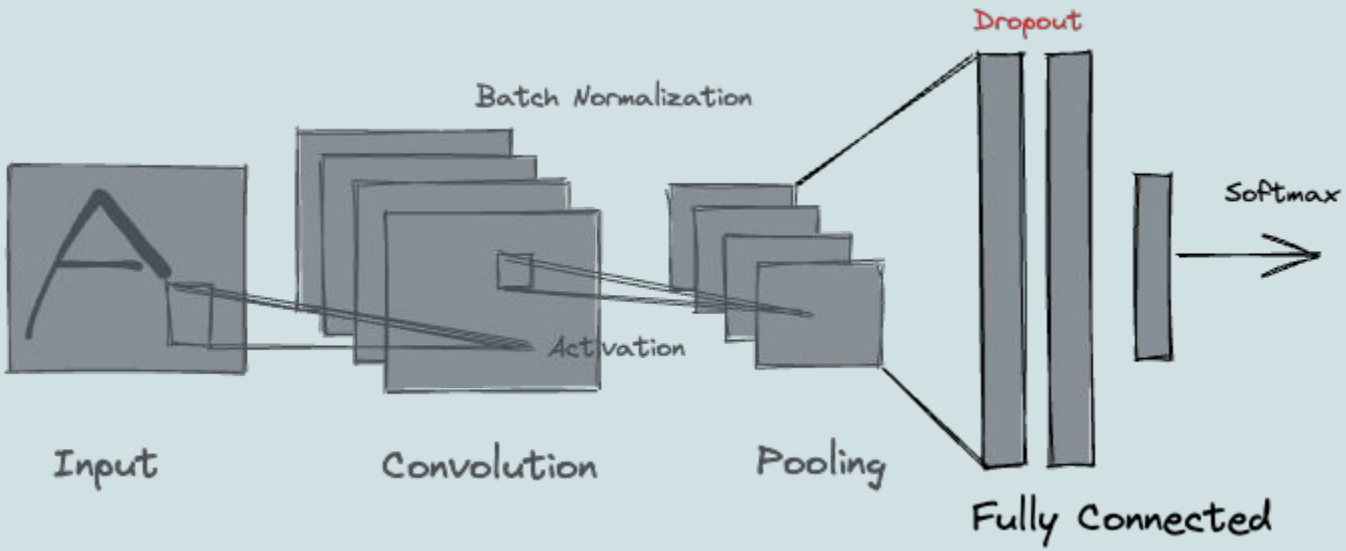
Batch Normalization

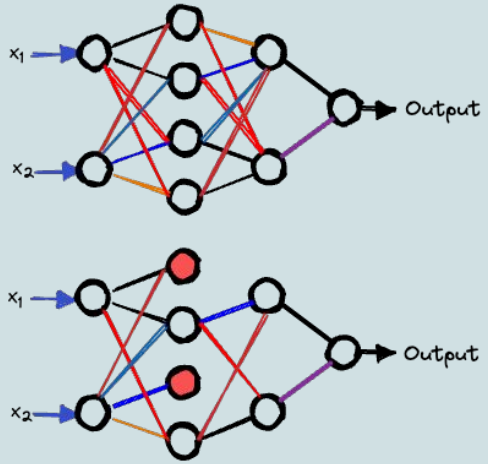


CONV => RELU => **BN** => POOL => ... => FC

CONV => **BN** => RELU => POOL => ... => FC

Dropout





It is a form of regularization
Reduces overfitting
Increases test/validation accuracy (sometimes at expense of training accuracy)
Randomly disconnects node from current layers to next layer with probability, p