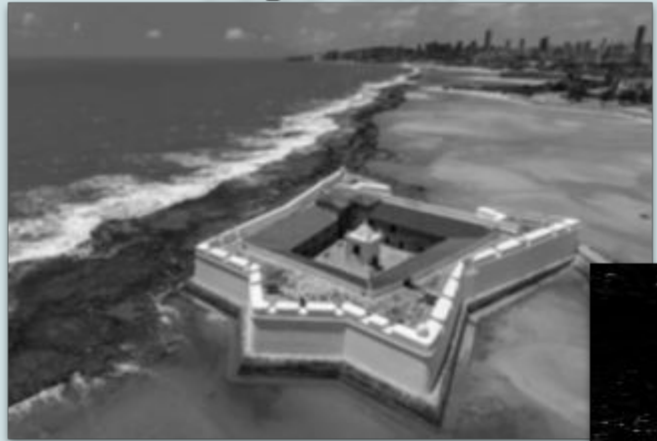
Fundamentals of Convolutional Neural Networks (CNN)

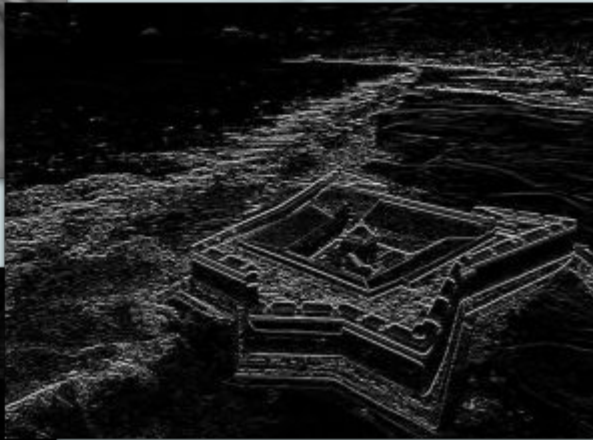
ivanovitch.silva@ufrn.br
@ivanovitchm

Original

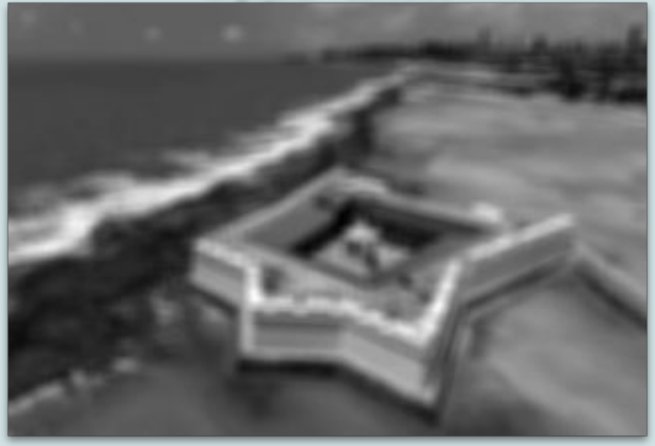
small_blur - convolve



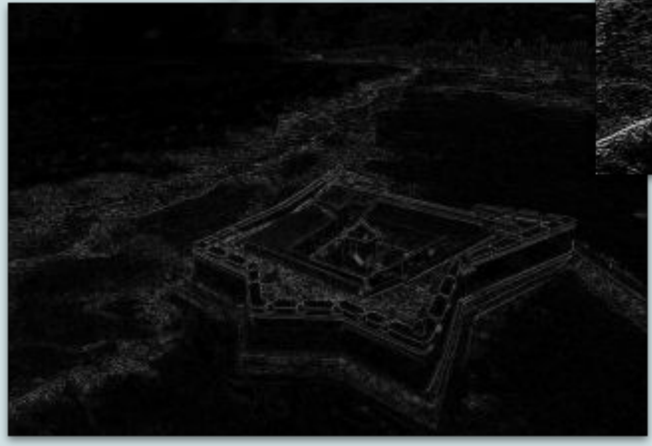
sobel_y - convolve



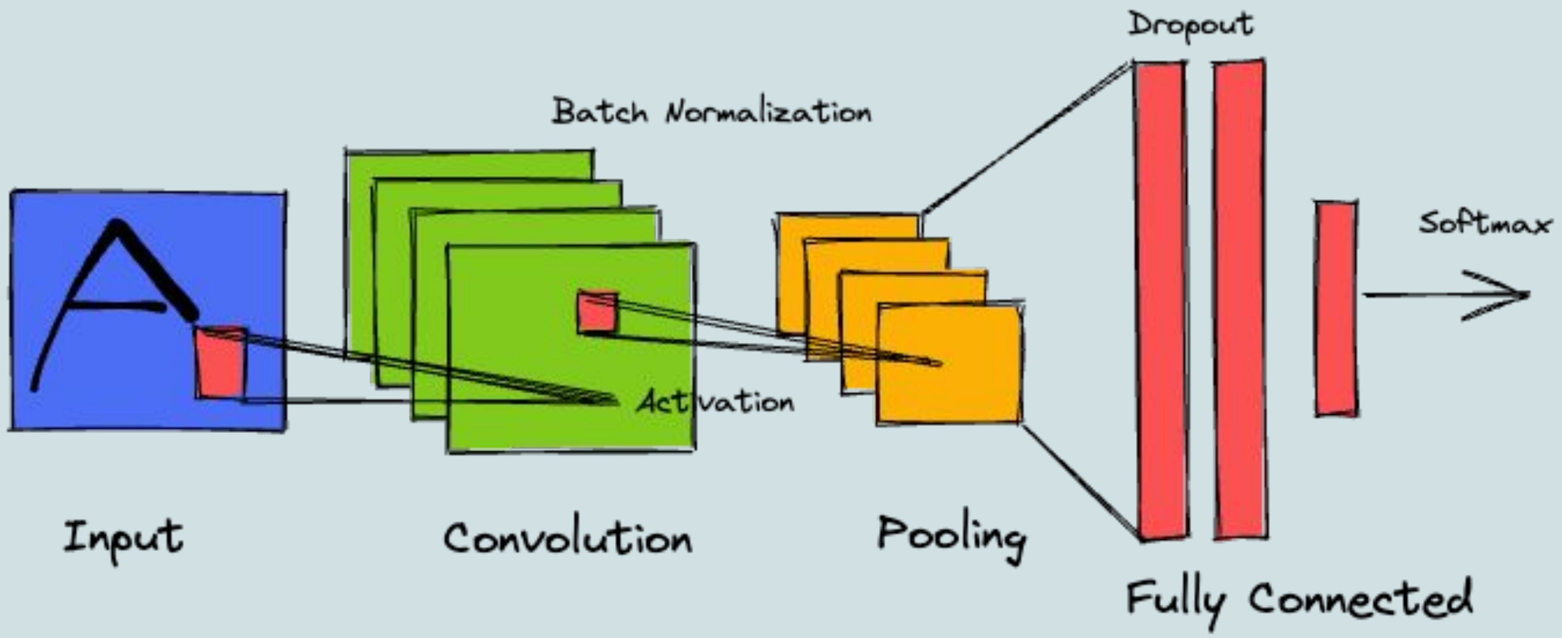
large_blur - convolve



laplacian - convolve



CNN Building Blocks

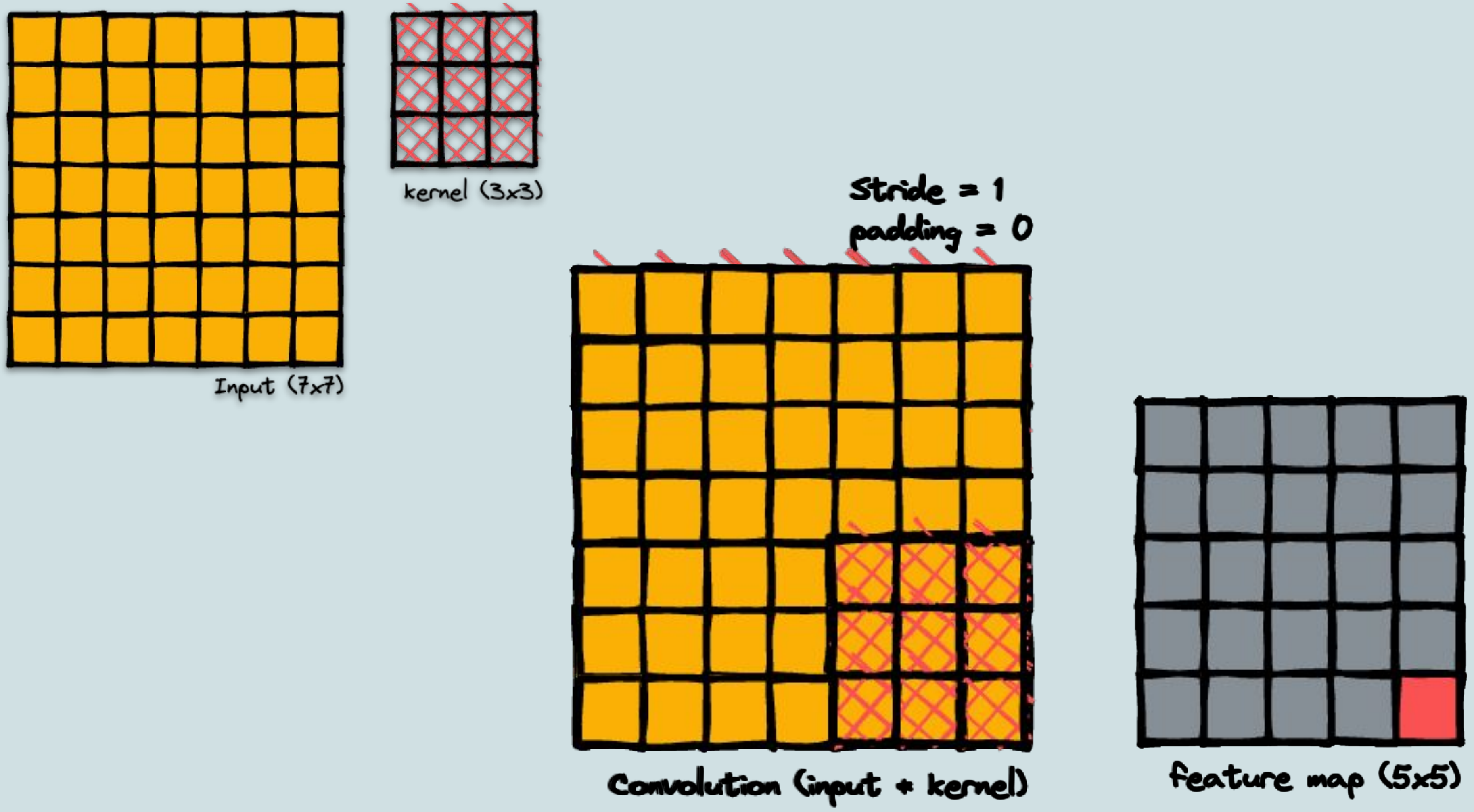


Convolution Explained in Code

```
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
```

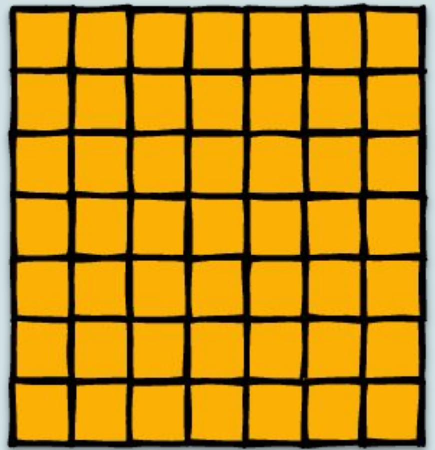

kernel (3x3)
Input (7x7)
Stride = 1
padding = 0
Convolution (input * kernel)
feature map (5x5)

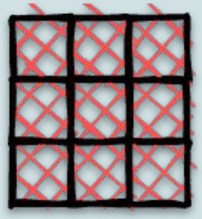
Summary of Convolutions

n x n input
k x k kernel
padding p
stride s

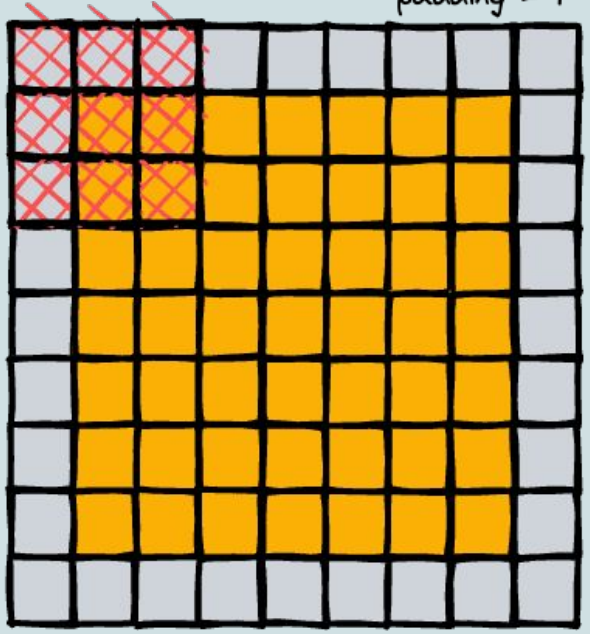
Padding

Valid (0)
Same(1)

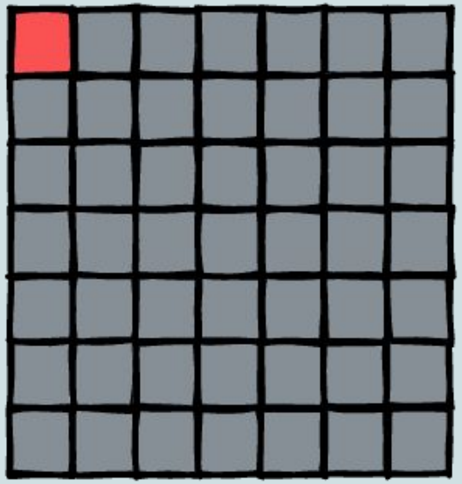

Input (7x7)


kernel (3x3)

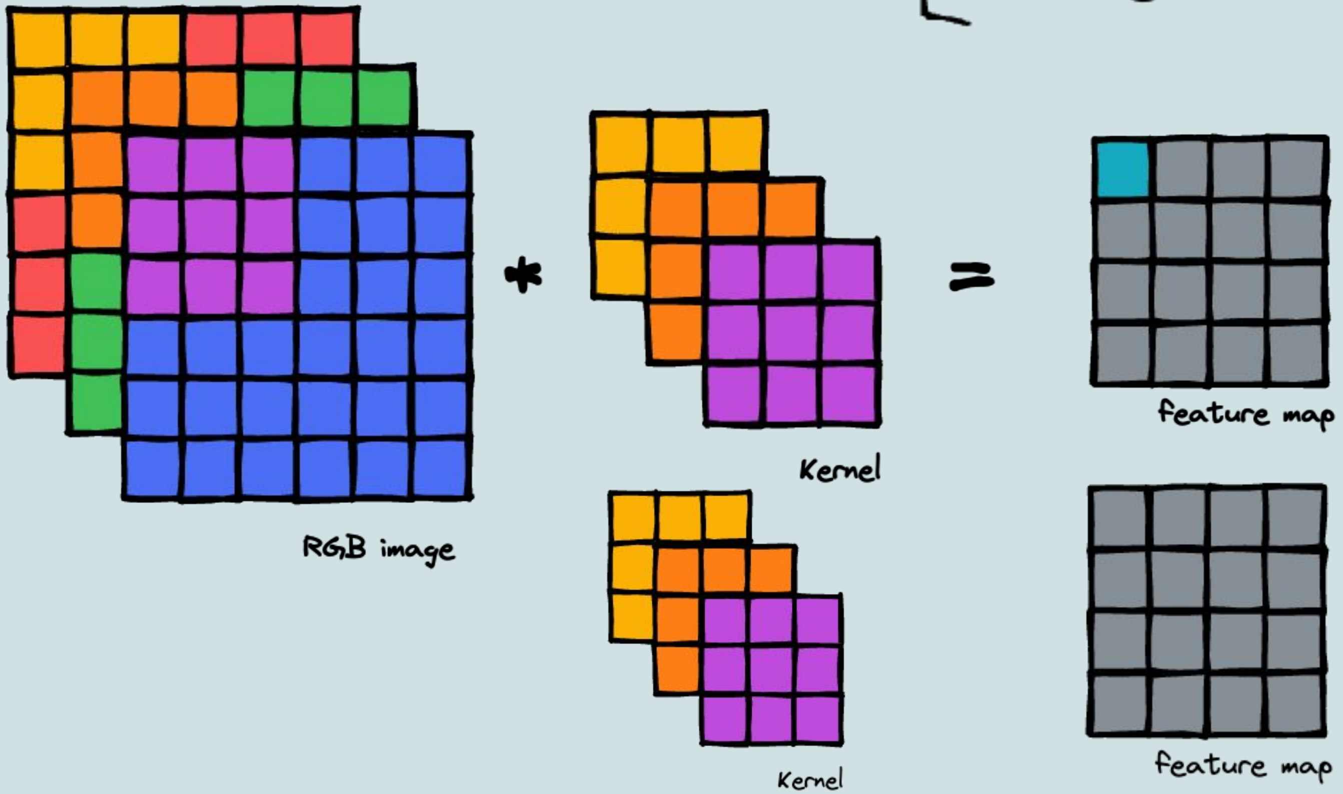
$$\left\lfloor \frac{n + 2p - k}{s} + 1 \right\rfloor$$



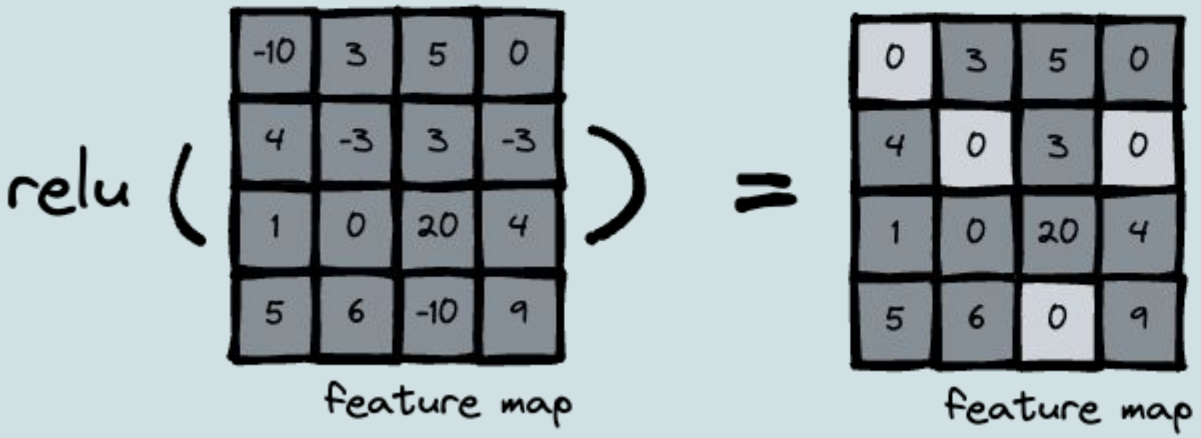
Convolution (input * kernel)


feature map (7x7)

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



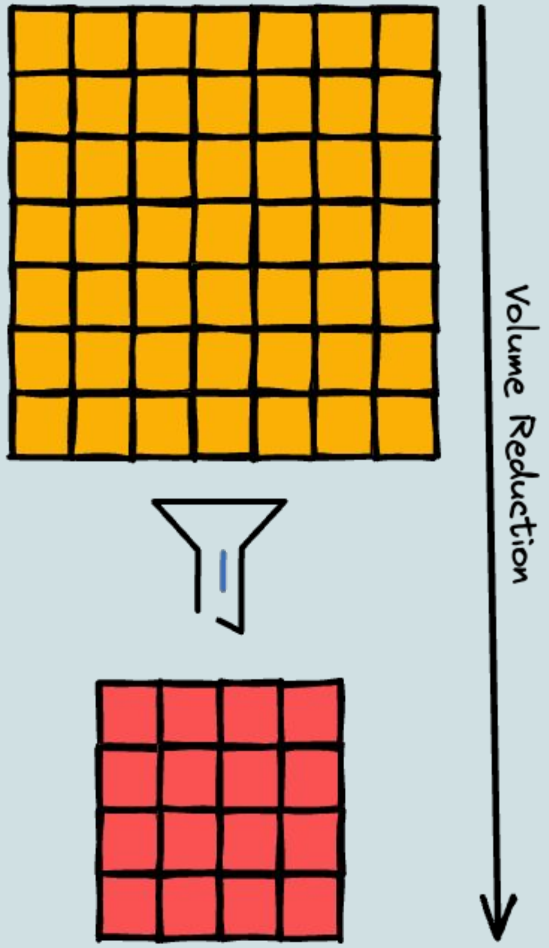
Activation Function



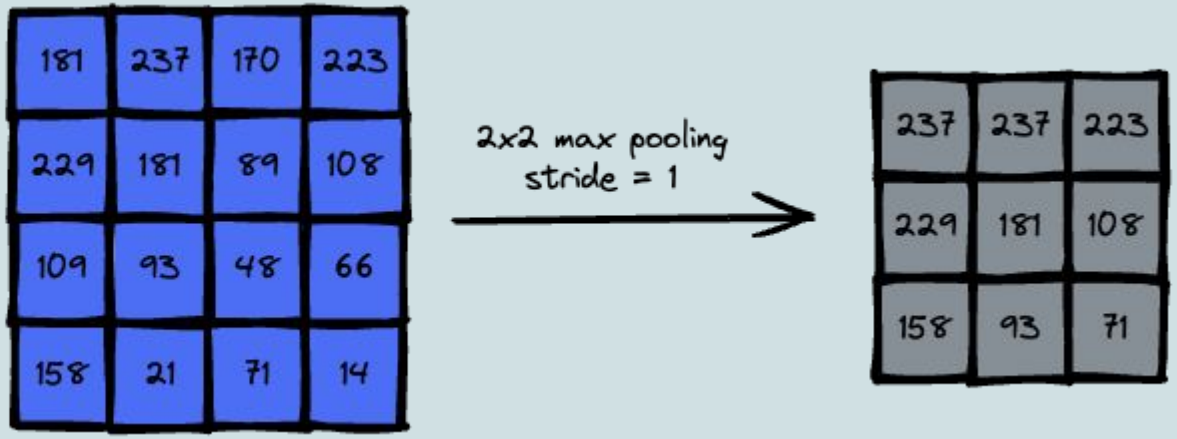
Pooling Layers

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

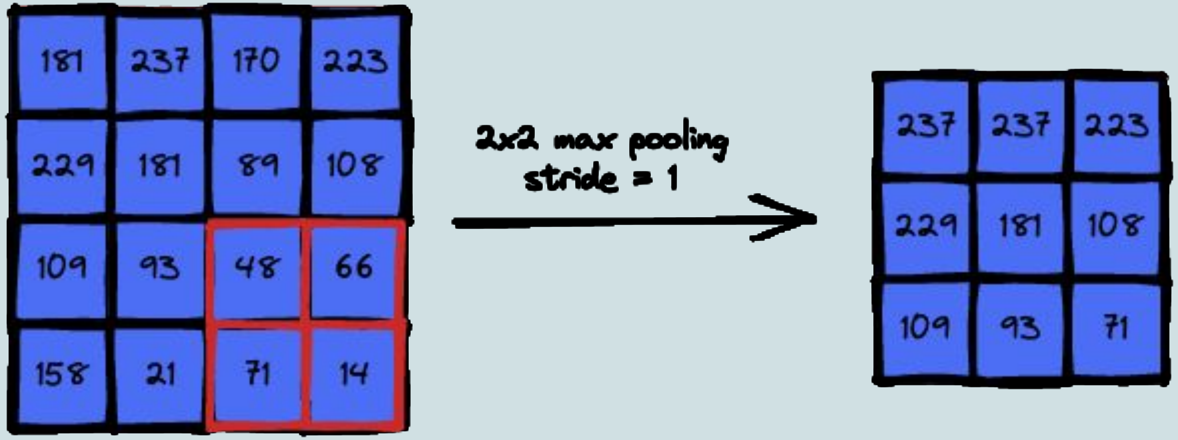


181	237	170	223
229	181	89	108
109	93	48	66
158	21	71	14

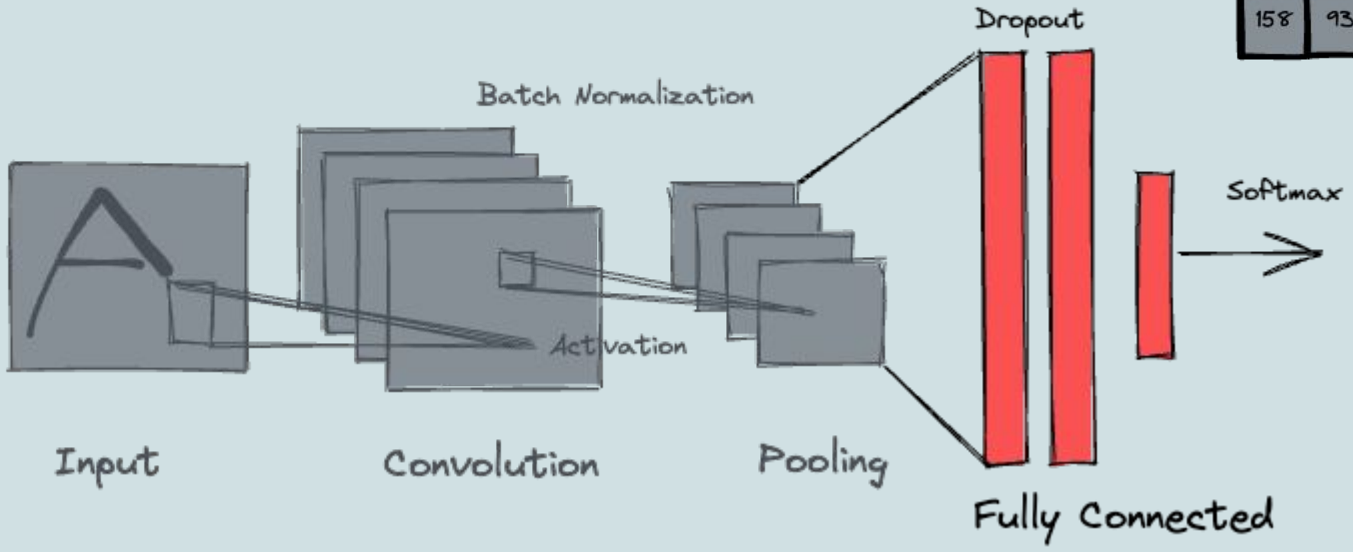
237	237	223
229	181	108
158	93	71

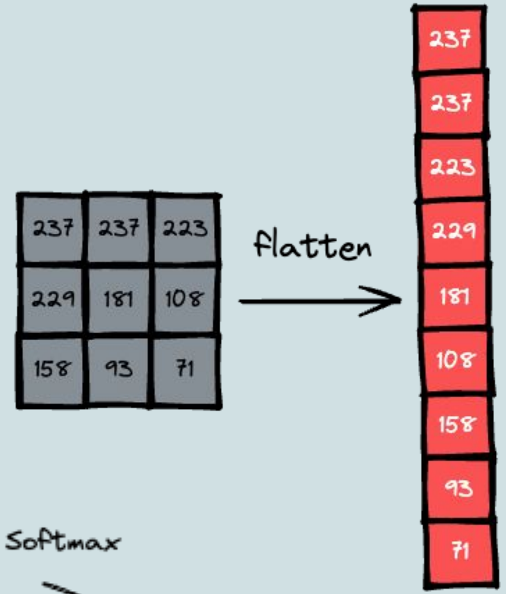
$$\left\lfloor \frac{n + 2p - k}{s} + 1 \right\rfloor$$

Pooling Layers



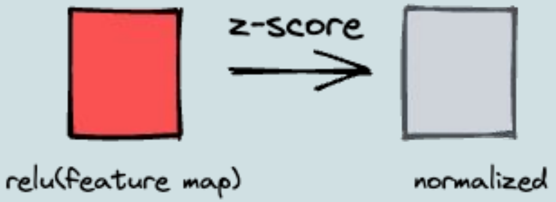
Fully Connected Layer

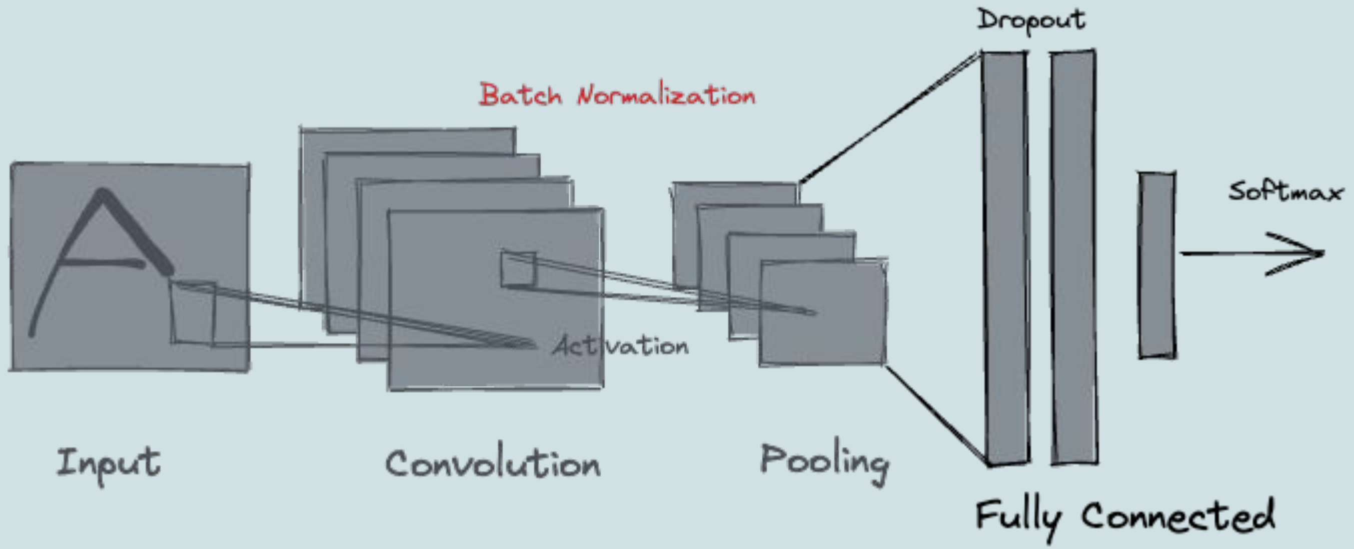




CONV => RELU => POOL => ... => FC

Batch Normalization

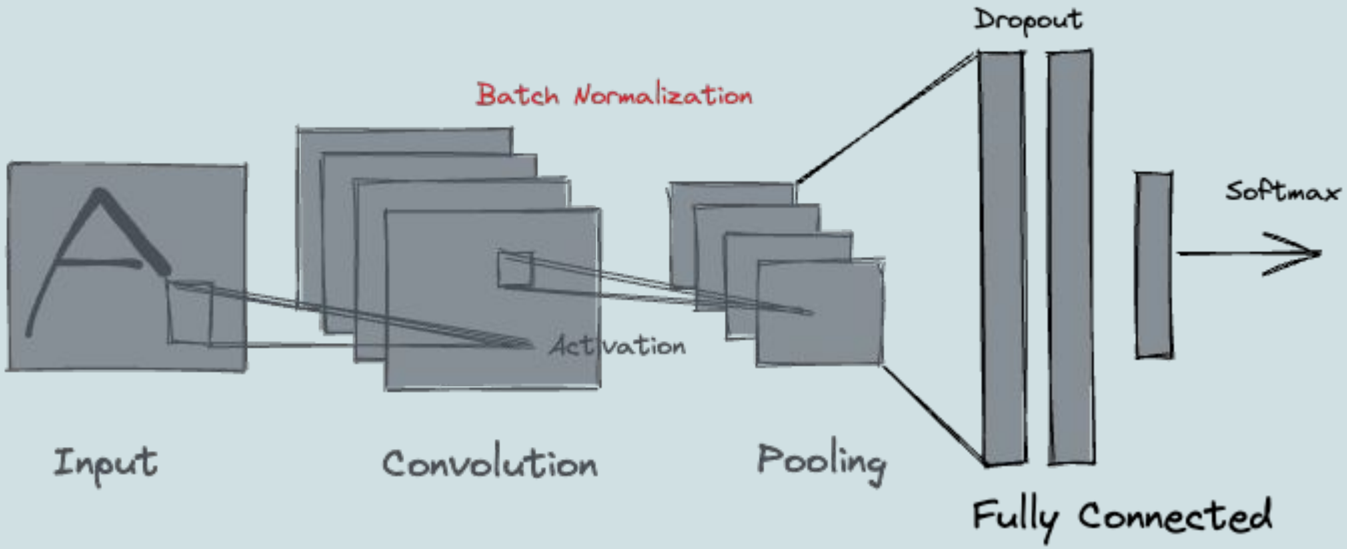




Accelerate the training process
Lightweight regularization

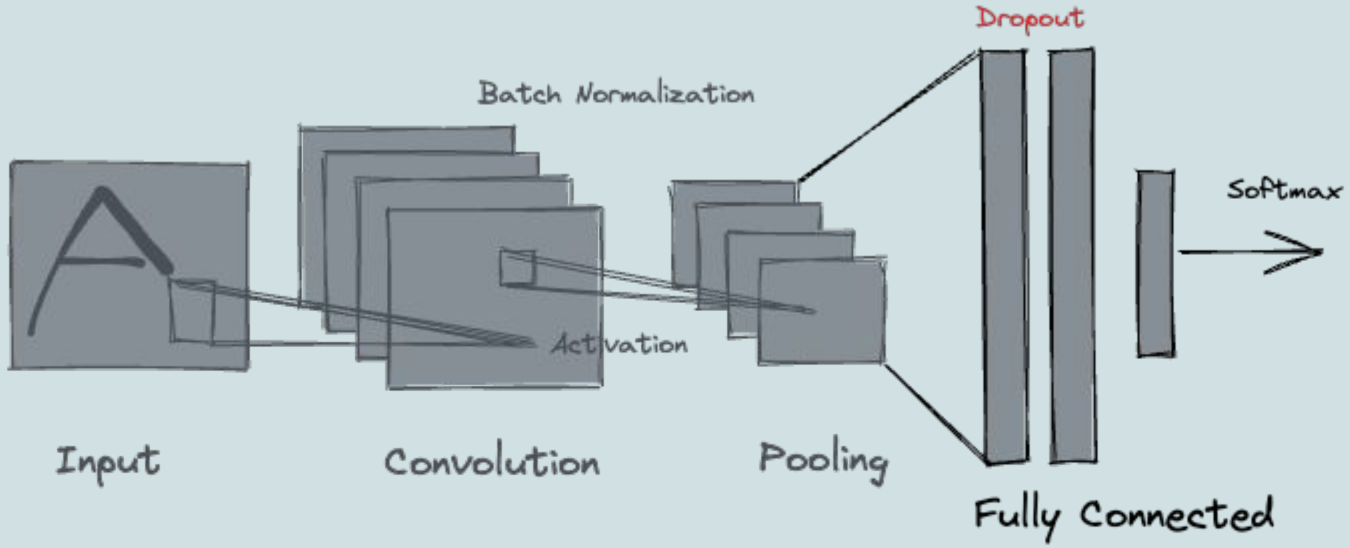
```
z = np.random.rand(3,3)
mean, std = np.mean(z), np.std(z)
norm = (z - mean)/(std + 1e-7)
```

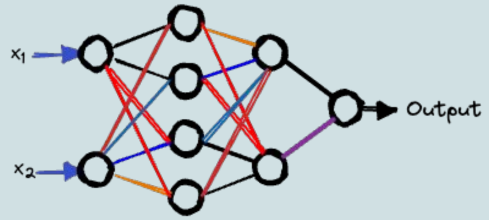
Batch Normalization

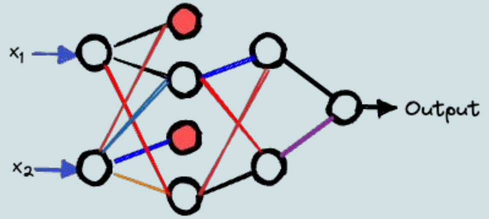


CONV => RELU => **BN** => POOL => ... => FC
CONV => **BN** => RELU => POOL => ... => FC

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