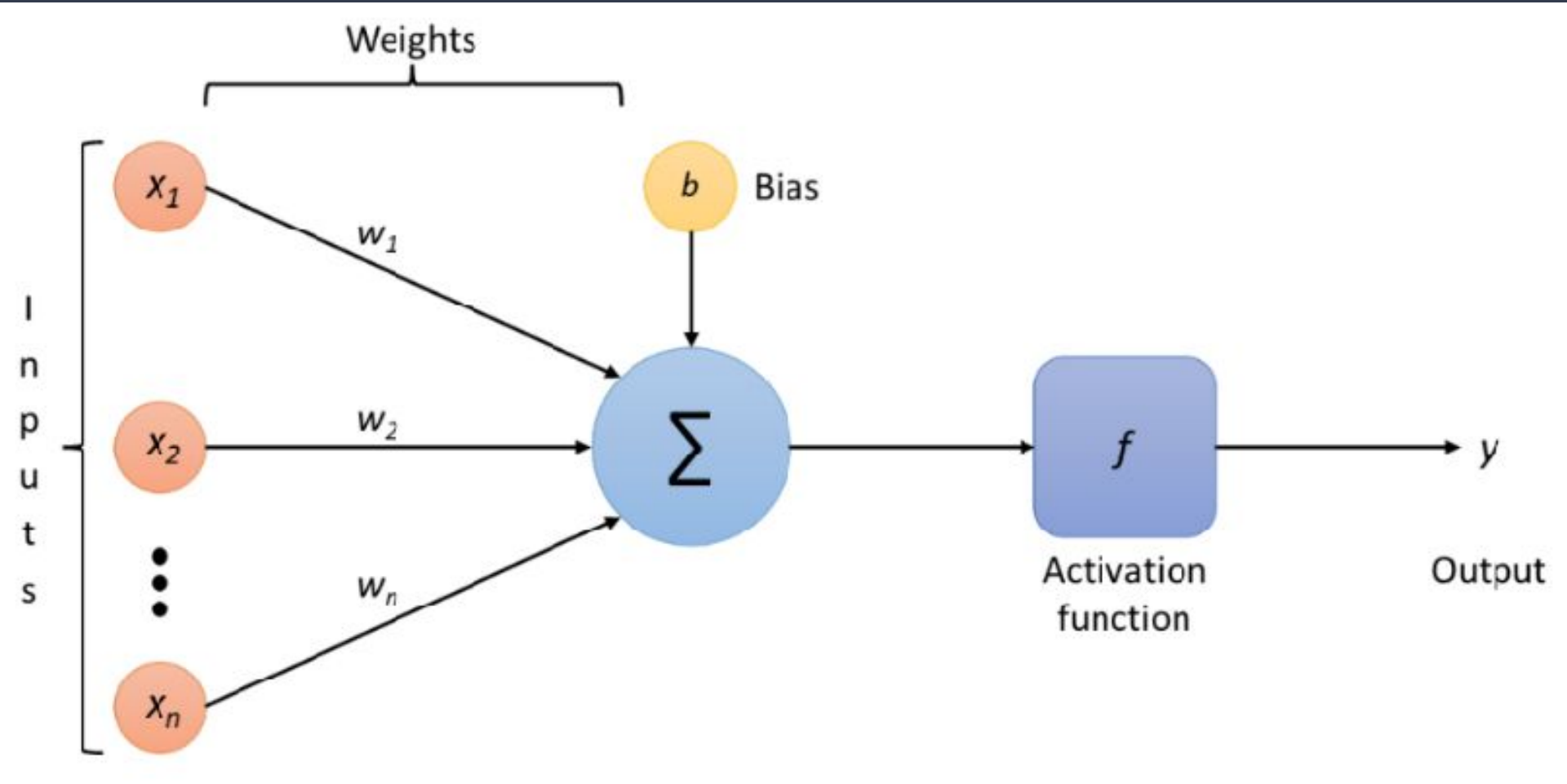
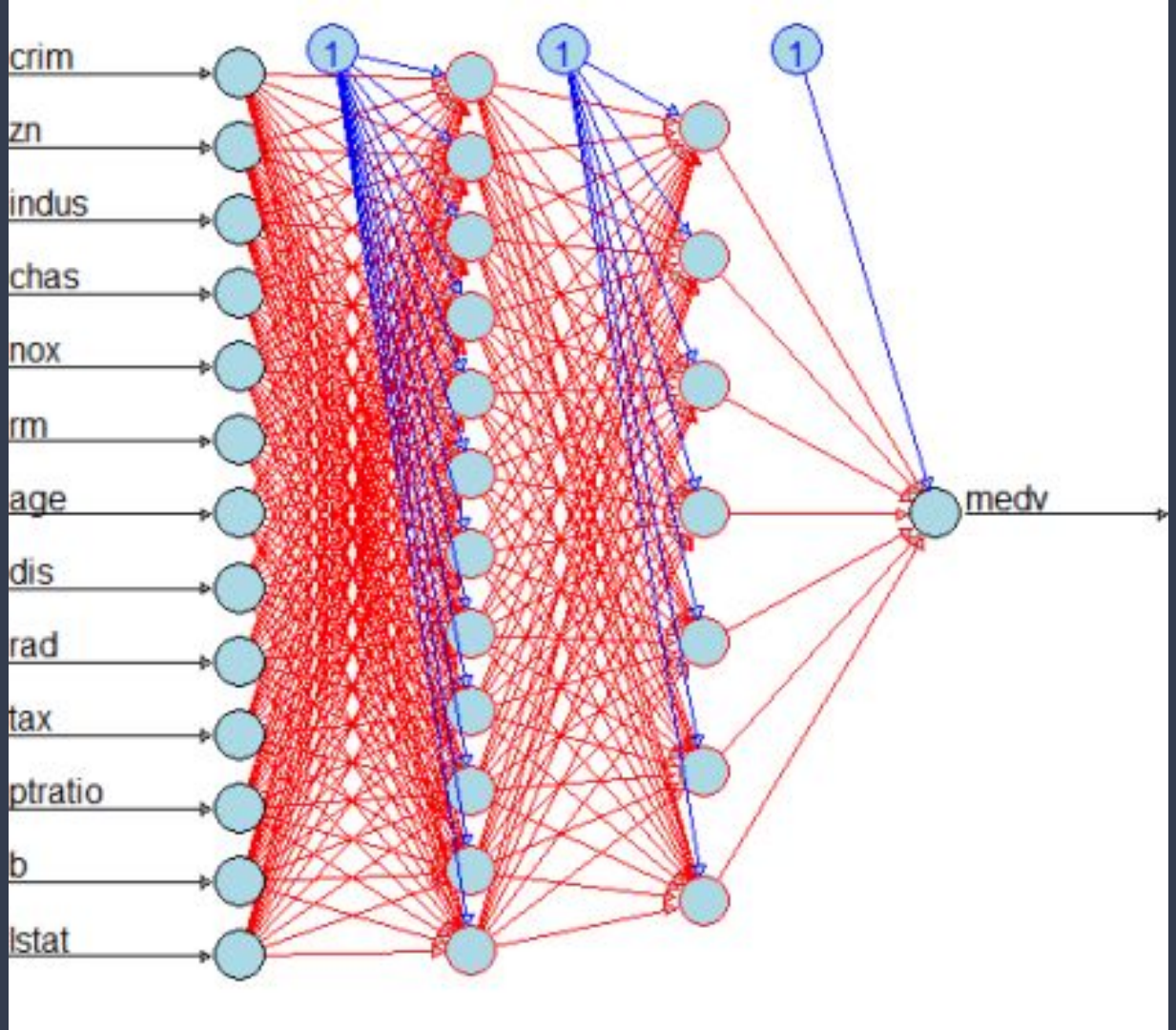
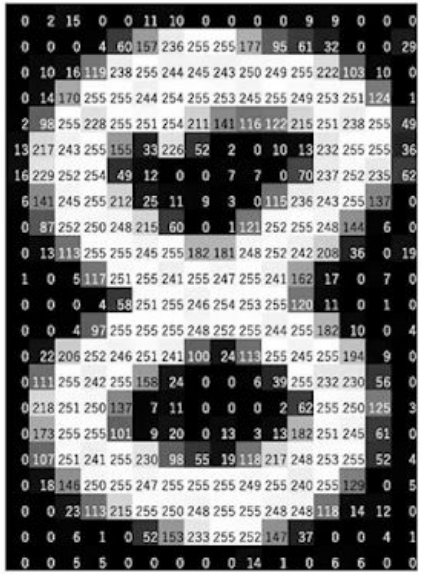
Index Class

- Image data
- Convolution Operation (Edge detection)
- Filters - patches - padding - borders
- Pooling


Weights
x_1
x_2
x_n
w_1
w_2
w_n
b
Bias
Inputs
$\sum$
f
Activation function
y
Output


crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
b
lstat
1
1
1
medv

Introduction to Image Data



```
  0   2  15   0   0  11  10   0   0   0   0   9   9   0   0   0
  0   0   0   4  60 157 236 255 255 177  95  61  32   0   0  29
  0  10  16 119 238 255 244 245 243 250 249 255 222 103  10   0
  0  14 170 255 255 244 254 255 253 245 255 249 253 251 124   1
  2  98 255 228 255 251 254 211 141 116 122 215 251 238 255  49
 13 217 243 255 155  33 226  52   2   0  10  13 232 255 255  36
 16 229 252 254  49  12   0   0   7   7   0  70 237 252 235  62
  6 141 245 255 212  25  11   9   3   0 115 236 243 255 137   0
  0  87 252 250 248 215  60   0   1 121 252 255 248 144   6   0
  0  13 113 255 255 245 255 182 181 248 252 242 208  36   0  19
  1   0   5 117 251 255 241 255 247 255 241 162  17   0   7   0
  0   0   0   4  58 251 255 246 254 253 255 120  11   0   1   0
  0   0   4  97 255 255 255 248 252 255 244 255 182  10   0   4
  0  22 206 252 246 251 241 100  24 113 255 245 255 194   9   0
  0 111 255 242 255 158  24   0   0   6  39 255 232 230  56   0
  0 218 251 250 137   7  11   0   0   0   2  62 255 250 125   3
  0 173 255 255 101   9  20   0  13   3  13 182 251 245  61   0
  0 107 251 241 255 230  98  55  19 118 217 248 253 255  52   4
  0  18 146 250 255 247 255 255 255 249 255 240 255 129   0   5
  0   0  23 113 215 255 250 248 255 255 248 248 118  14  12   0
  0   0   6   1   0  52 153 233 255 252 147  37   0   0   4   1
  0   0   5   5   0   0   0   0   0  14   1   0   6   6   0   0
```

Analytics Vidhya
Learn everything about analytics

JPG 260 X 194

260 X 194 X 3

8,11,0, 55,13,25,19

15,241,2,155,13,35,65

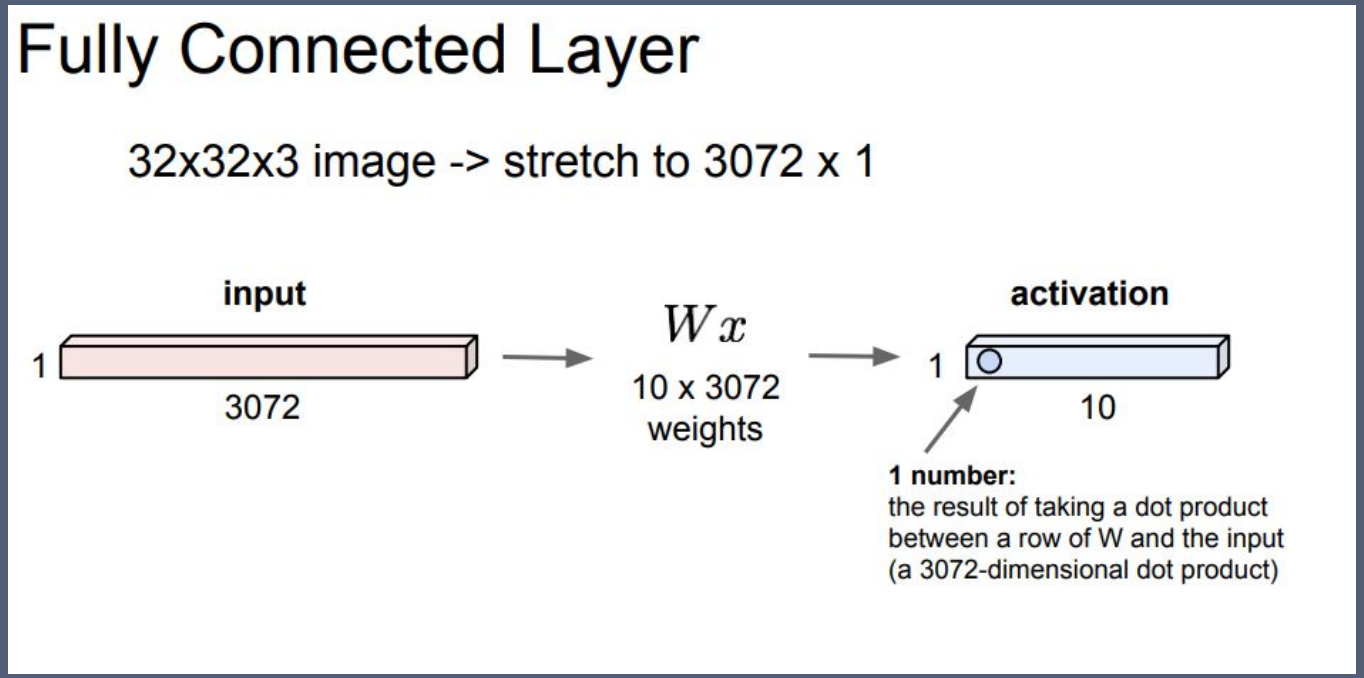
14,211,0,255,23,45,11

05,255,1,255,10,17,23

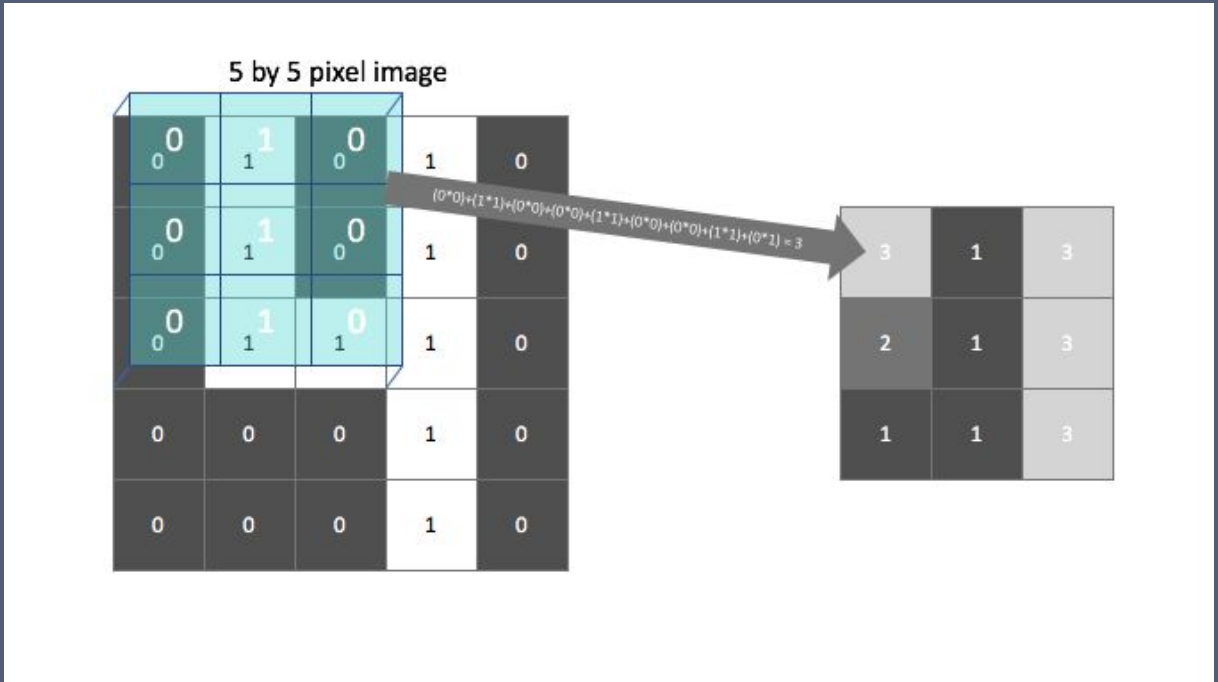
77,167,9,112,56,16,90

45,245,0,145,22,55,48

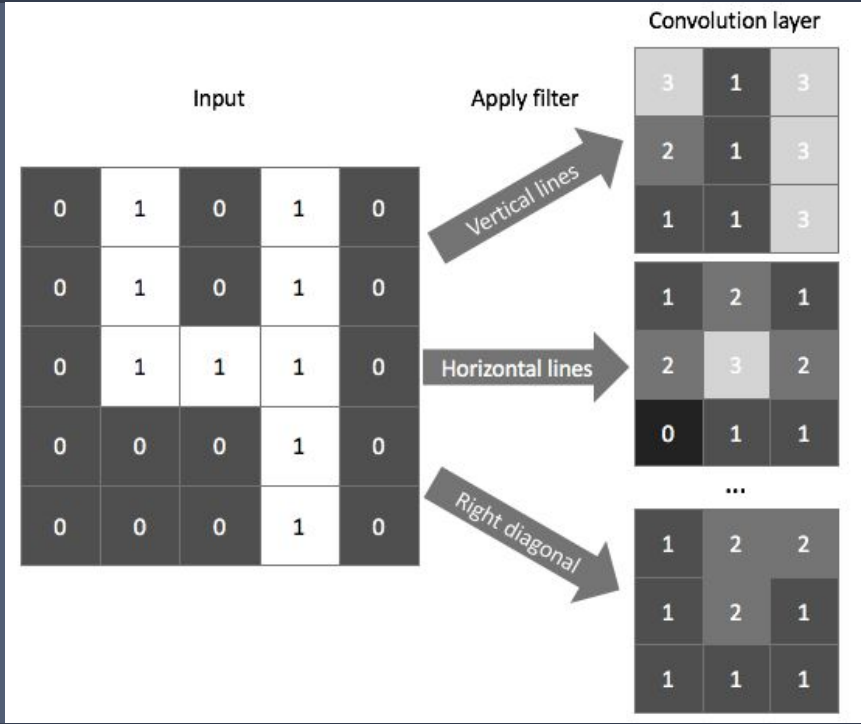
Fully connected layer



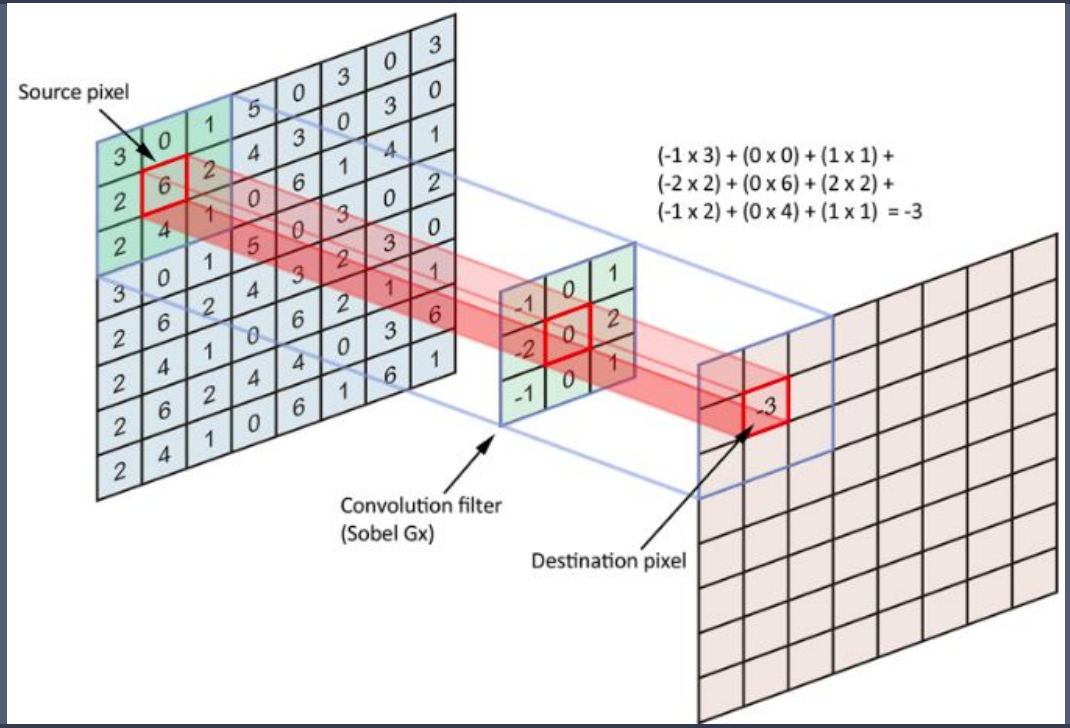
Conv Operation - what learns?



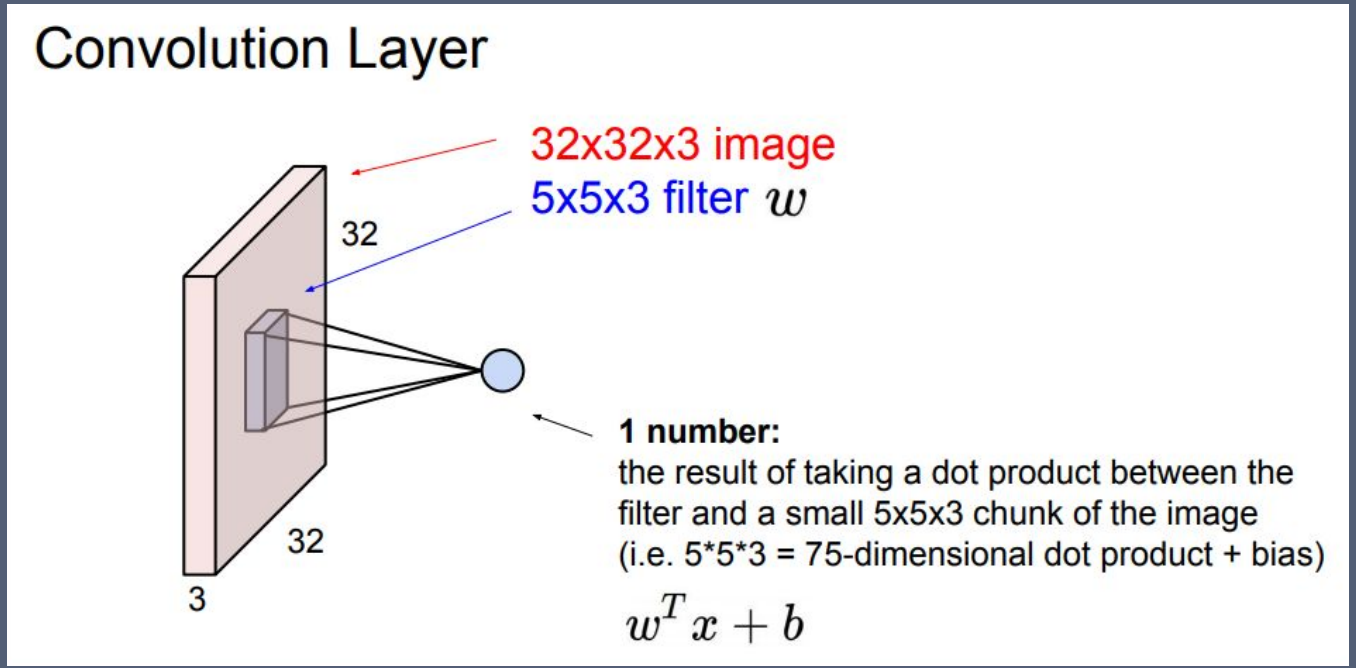
Operation


Input
Apply filter
Convolution layer
0 1 0 1 0
0 1 0 1 0
0 1 1 1 0
0 0 0 1 0
0 0 0 1 0
Vertical lines
Horizontal lines
Right diagonal
3 1 3
2 1 3
1 1 3
1 2 1
2 3 2
0 1 1
...
1 2 2
1 2 1
1 1 1

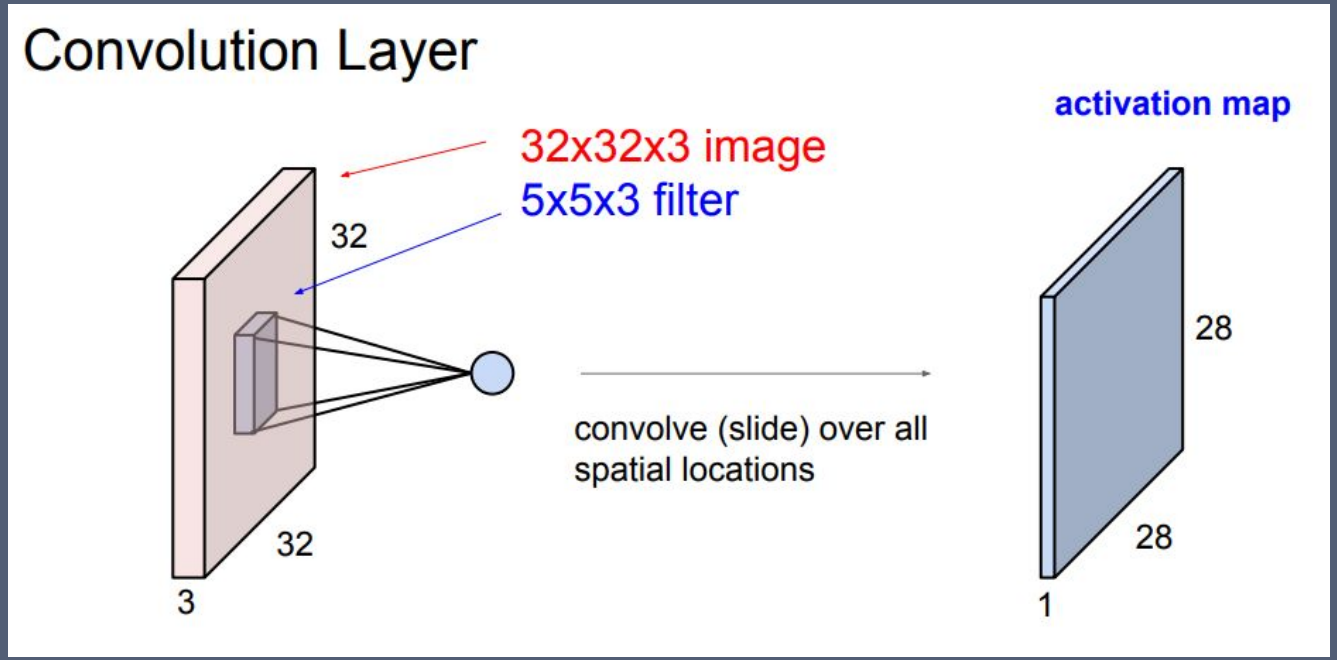
Convolutional Operation



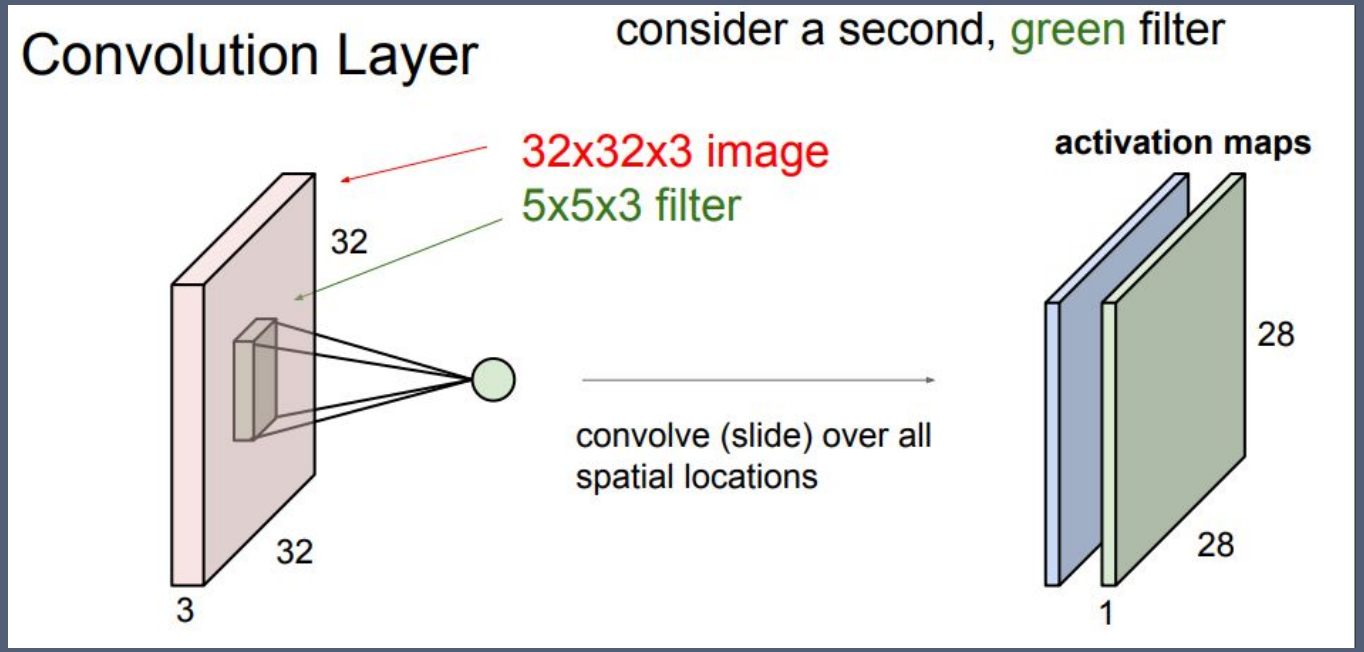
Convolutional Operation



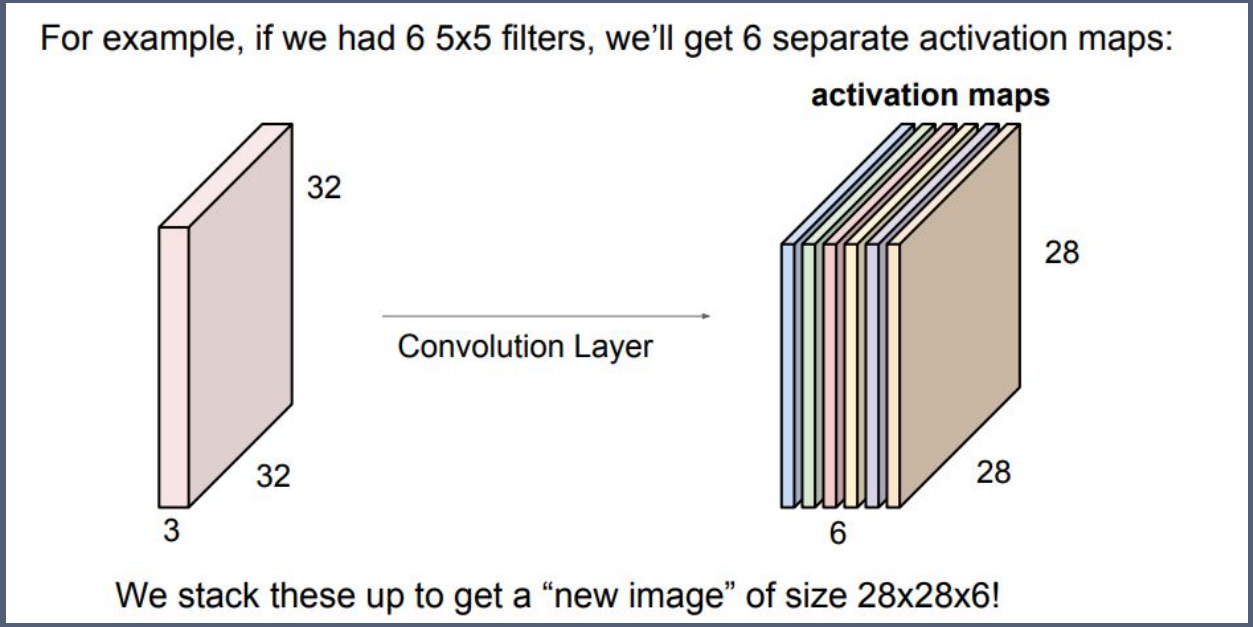
Convolutional Operation



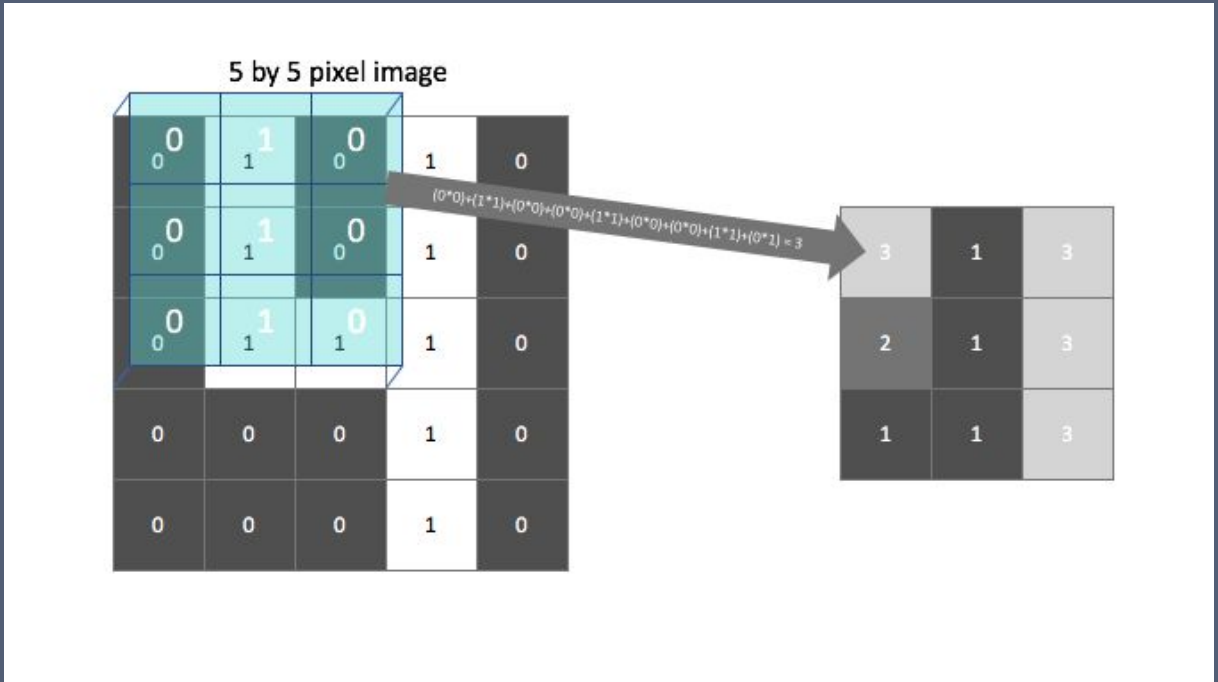
Convolutional Operation



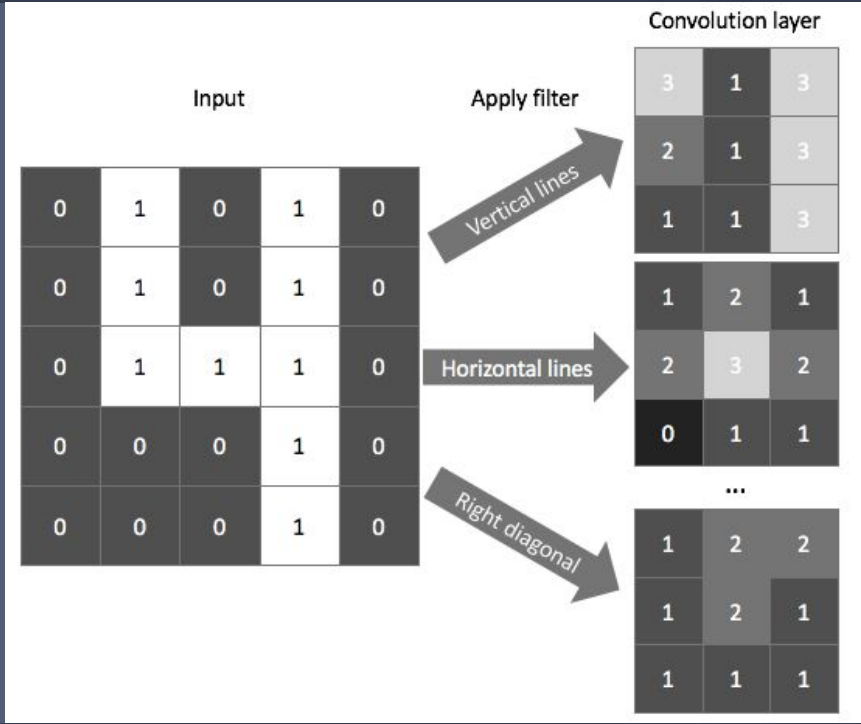
Convolutional Operation



Conv Operation - what learns?

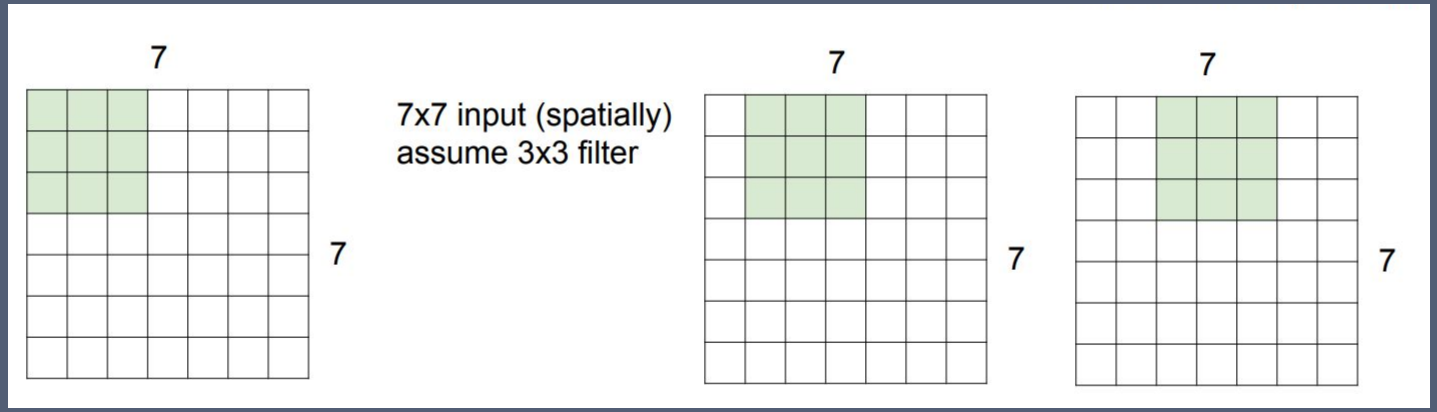


Conv Operation - what learns?



Convolutional Operation - Stride

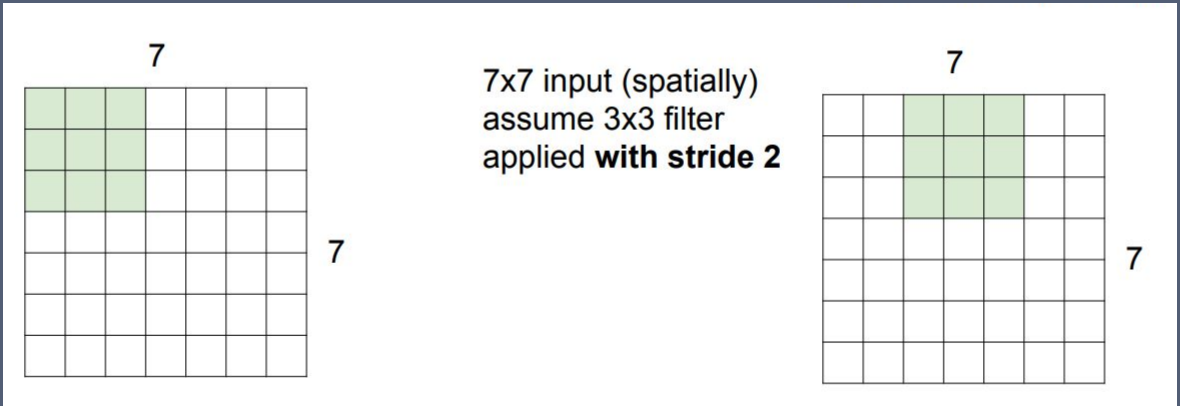
- Stride = 1 (Basic)
- 3x3 filter



- Output 5x5

Convolutional Operation - Stride

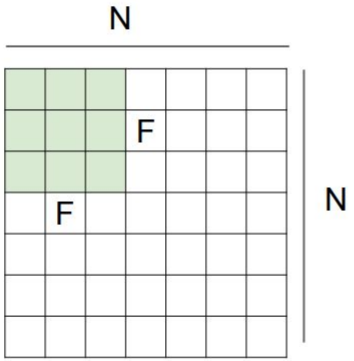
- Stride = 2
- 3x3 filter



- Output 3x3

Convolutional Operation - Stride

- This is not working for stride = 3

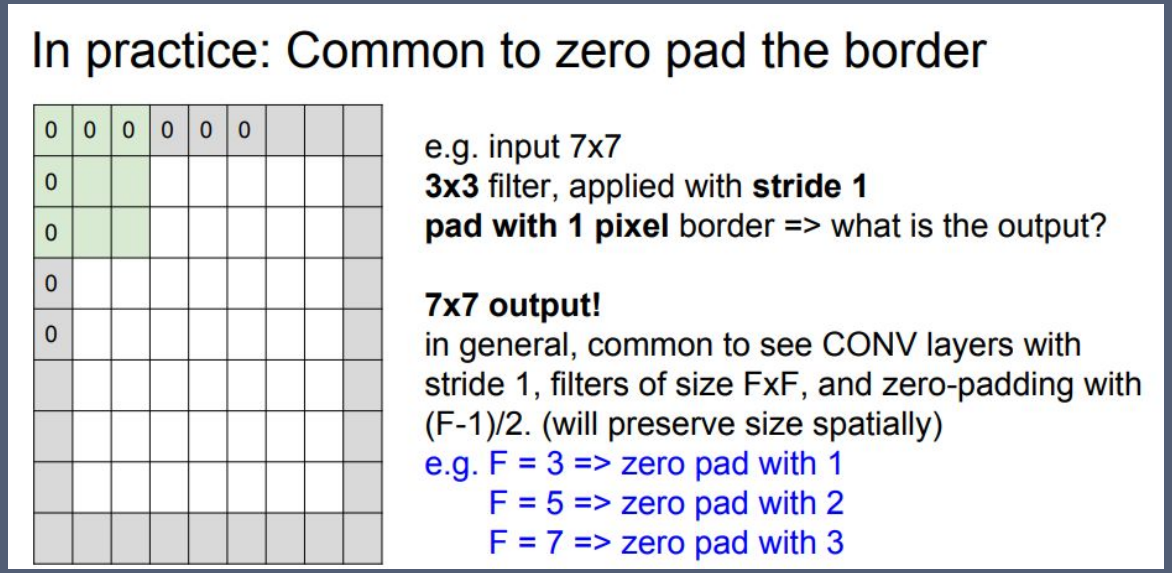


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Border - Padding

We can add a border to transform 7x7 to 9x9

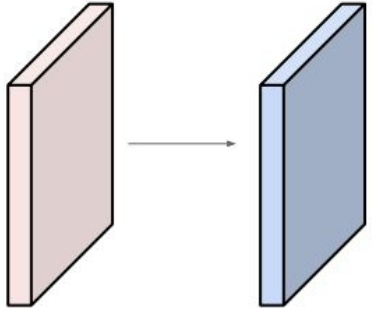


ConvNet

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

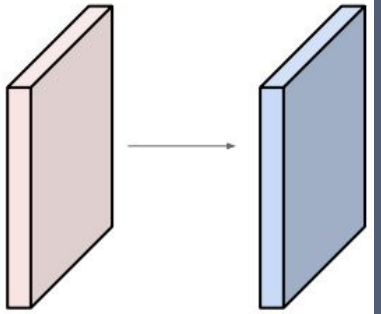


ConvNet

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

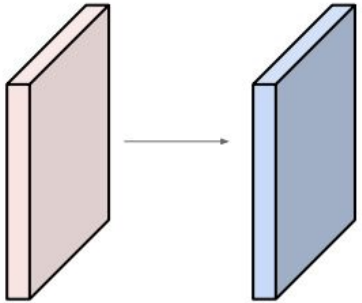


ConvNet

Examples time:

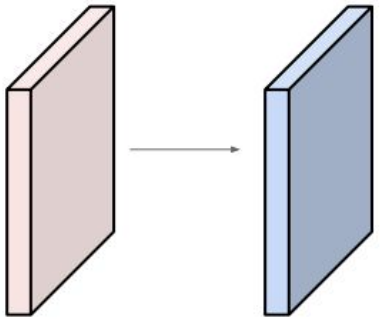
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



ConvNet

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

ConvNet

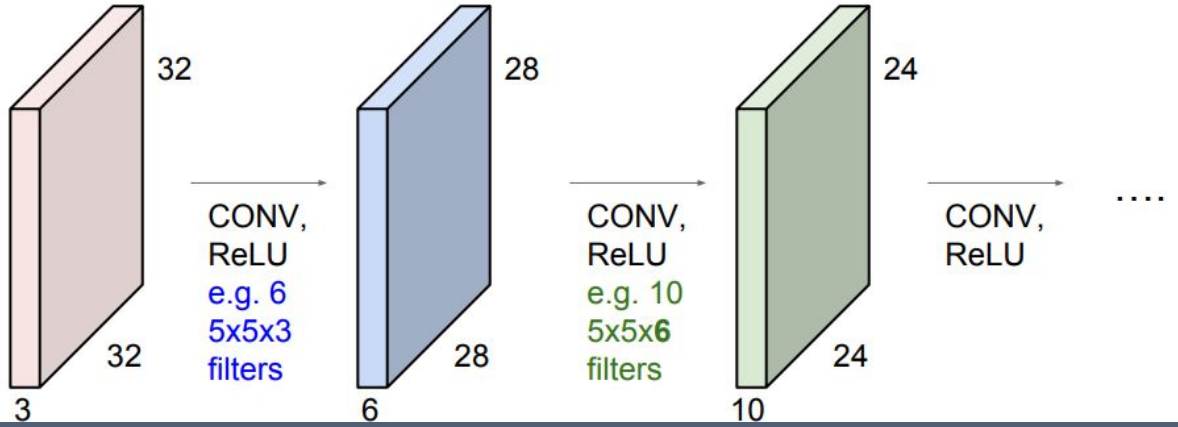
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

ConvNet

Remember back to…

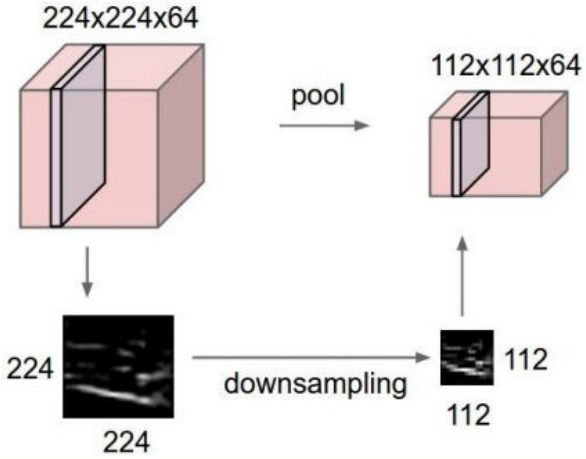
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



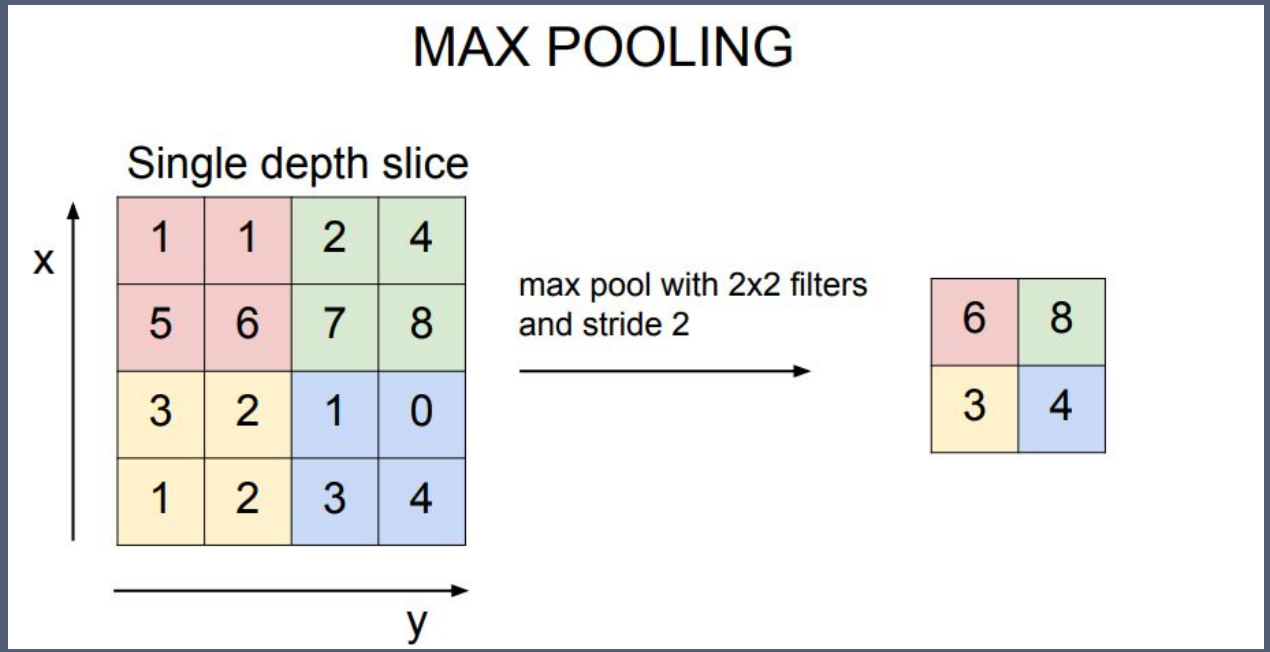
Pooling

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

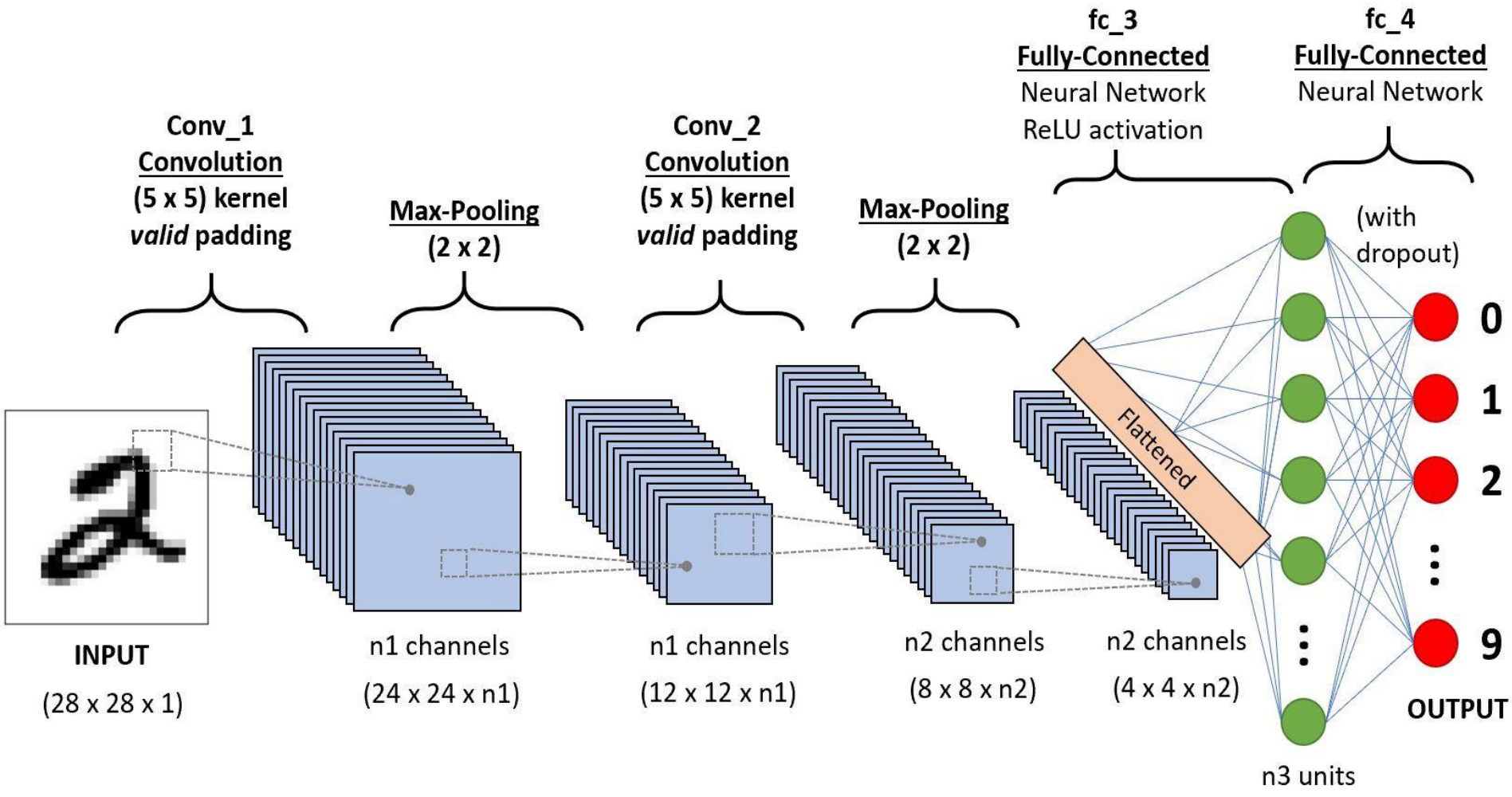


Max Pooling


MAX POOLING
Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Max Pooling

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers


fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
(with
dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

An entire convolutional network