+ Computer Vision:

One of the area that has benefited from CNN's

Different Types of problems:

1. Image classification
2. Objet Détection
3. Neural style Transfer (combing pattern and image)

The resolution of the image will play a role on the input features

+ Edge Detection Example

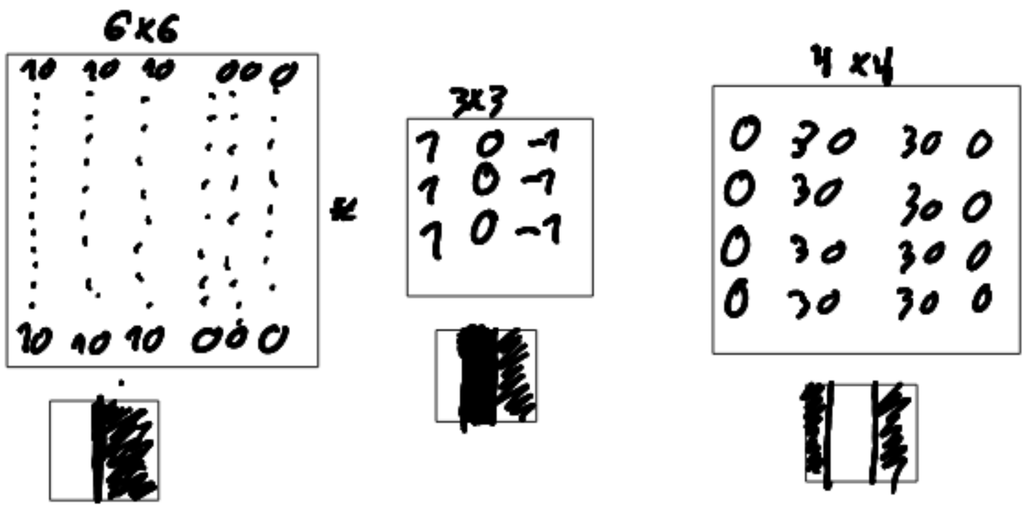
In a computer vision problem edges will be detected first

Filter (kernel) are used to convolve it with the network.

Lets say we have an image.

matrix (6 x 6) $*$ matrix (3x3) (filter) $\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$ = matrix 4 x 4

$*$ ↳ Sign for convolution



It detected an edge vertically

+ More Edge Detection

Edges can be dark or light, and the intensities will be flipped.

Types of filters examples:

1. Vertical

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

2. Horizontal.

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

3. Sobel Filter

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

3. Scharr filter

3	0	-3
10	0	-10
3	0	-3

+ Padding:

Before ⇒ 6×6 $(n \times n)$ * 3×3 $(f \times f)$ = 4×4

$n - f + 1 \; * \; n - f + 1$

$6 - 3 + 1 = 4$

Some issues ⇒ Shrinking output (not good for large networks)

⇒ Loss of information on the edges of the image.

Padding is adding an extra layer of 0's around your matrix.

$6 \times 6 \Rightarrow 8 \times 8$

So our result will stay 6×6 and preserve the original output.

The border could be padded with even more pixels.

- Valid and Same convolutions.

→ Valid: $n \times n$ * $f \times f$ → $n - f + 1 \times n - f + 1$

6×6 * 3×3 → 4×4

→ Same: Pad so that output size is the same as input size.

- $n + 2p - f + 1 \times n + 2p - f + 1 \Rightarrow$

$\Rightarrow p = \frac{f-1}{2}$ where p is padding.

f is usually odd

+ Strided Convolution:

Skipping over one position extra.
It will result in a much smaller output.

$n \times n \quad * \quad f \times f \quad \approx \frac{n+2p-f}{s} \times \frac{n+2p-f}{s}+1$

With padding p and Stride s

We usually round down if its not an integer

+ Convolution over Volume (3d)

Now we can detect images in RGB

$(6 \times 6 \times 3) \quad * \quad (3 \times 3 \times 3) \quad = \quad (4 \times 4)$

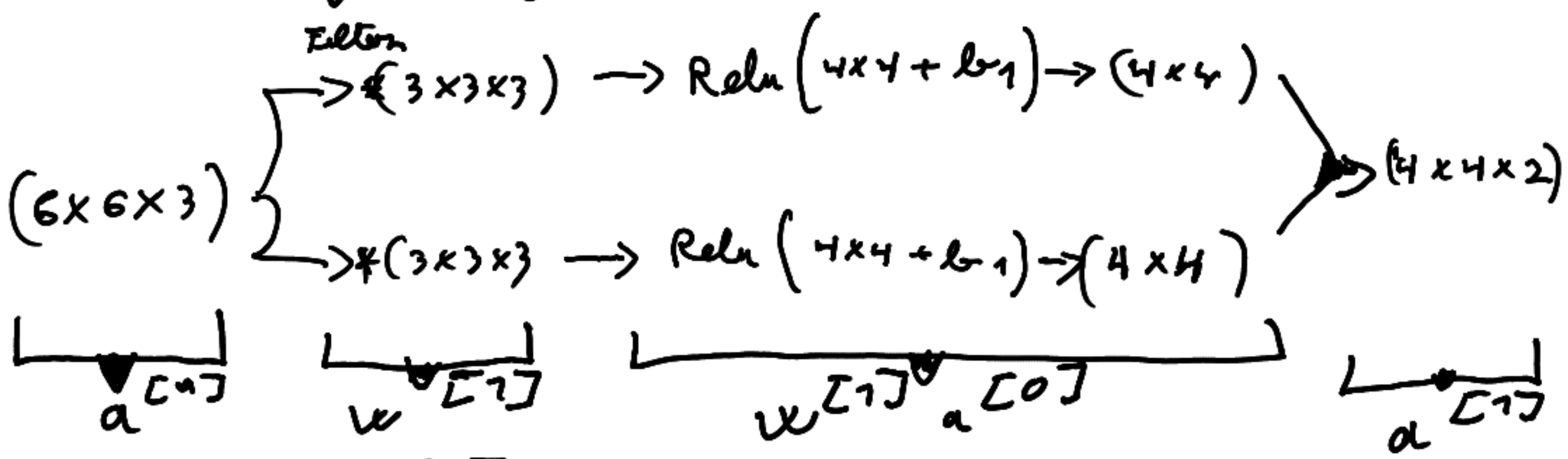
(height x width, # channels) Number of channels needs to be the same

Operations are the same but with depth.

$(n \times n \times n_c) * (f \times f \times n_c) = n-f+1 \times n-f+1 \times n_c'$

Where n_c = number of channels
n_c' = number of filters

+ One layer of a convolutional network



$$z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$

- Summary of notation:

- If layer l is a convolutional layer:

$f^{[l]}$ = filter size
$p^{[l]}$ = padding
$s^{[l]}$ = stride

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

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

$$n^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

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

Each filter: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$

Activation: $a^{[l]} \to n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

$A^{[l]} \to m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$ → Number of filters in layer l

Bias: $n_C^{[l]} \to (1, 1, 1, n_C^{[l]})$

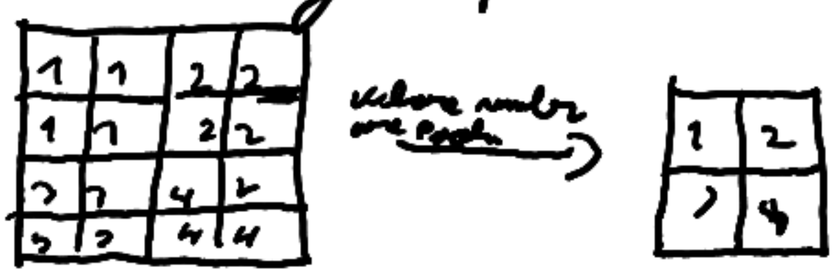
+ CNN example:

- Types of layers in a convolutional network.
  + Convolution (conv)
  + Pooling (Pool)
  + Fully connected (FC)

+ Pooling Layers.

Used to reduce the size of the representation and to speed up the computation.

Breaking up the input into regions



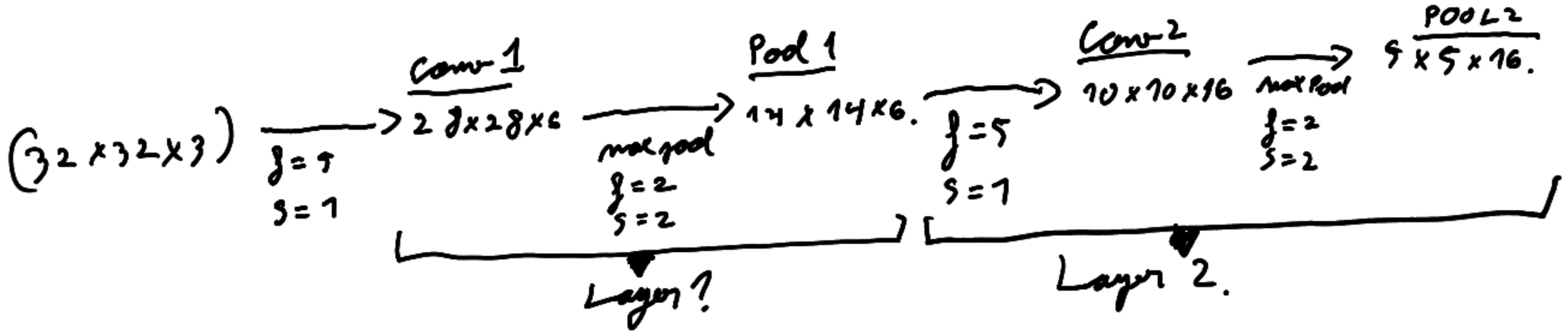
(Max Pooling) where the max is taken from each section

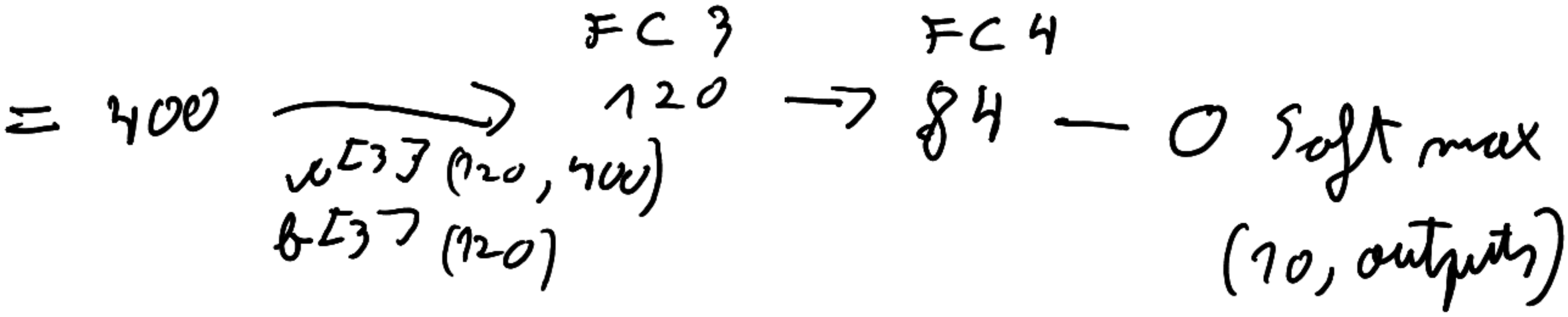
In a $(5 \times 5 \times nc) \Rightarrow \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

(Average Pooling) average of the networks are taken.

↳ Usually used to collapse the network.

+ CNN example:





+ Why Convolutions

- Parameter sharing: A feature detector (such as a vertical edge detector), that's useful in one part of the image is probably useful in another part of the image.
- Sparsity of connections: In each layer, each output value depends on a small number of inputs.