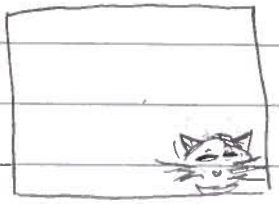
L6-1

Convolutional neural networks

An example of matching the structure of a neural network model to invariant properties of the problem.

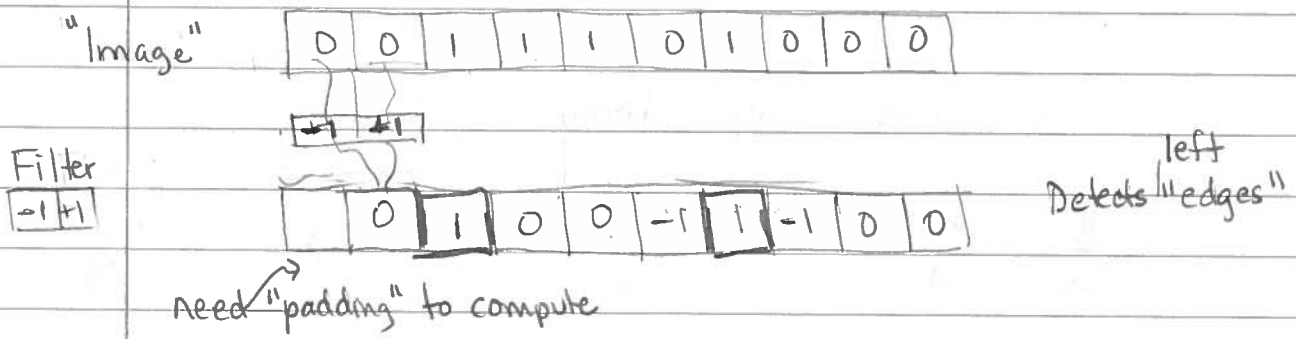


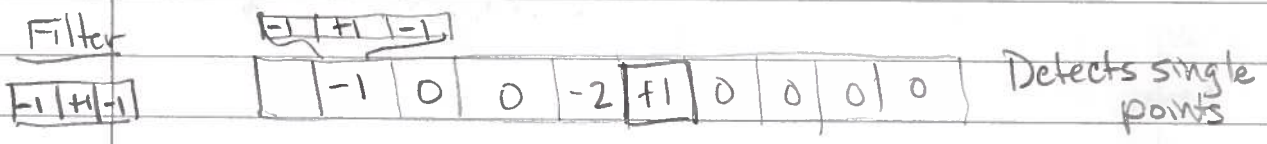
Is there a cat in this image?

- Relevant pixels are spatially compact
- Cat looks the same anywhere in the image

Filter is a function on a neighborhood of values that detects presence of a pattern in image or sequence data

Simplest case - 1-dimensional

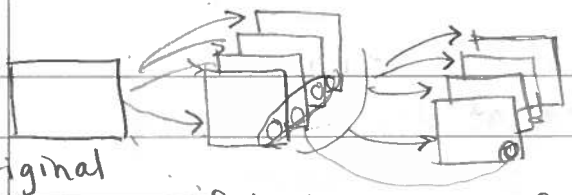




Two-d versions of these simple filters are found in visual cortex of (all?) mammal brains.
Similar patterns arise from statistical analysis of natural images ("eigen patches")

Computer vision people used to spend a lot of time hand-designing "filter banks"!

L 6-2

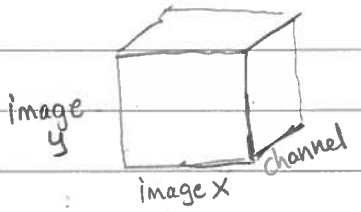


original image

filtered images "channels"

each filter in this level takes same spatial field in all images of previous layer as input

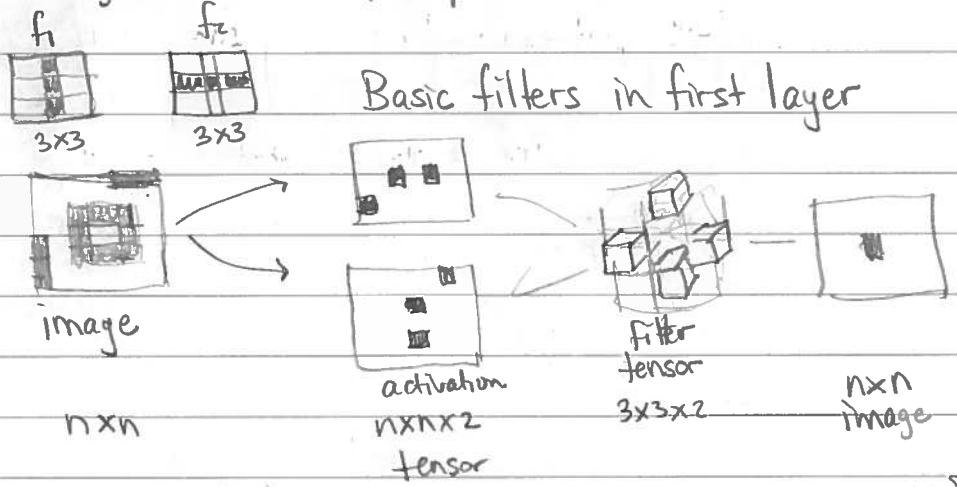
Operations on 3D chunks of data



tensor

Algebra of tensors is like matrices. But we will not go into it. But it's cool!

Finding a more complex pattern:



Defining a filter layer

- number of filters m^l
- size of filters $K^l \times K^l \times m^{l-1}$
- stride s^l
- input tensor size $n^{l-1} \times n^{l-1} \times m^{l-1}$

Produces output tensor of size $n^l \times n^l \times m^l$

where $n^l = \lfloor n^{l-1}/s^l \rfloor$

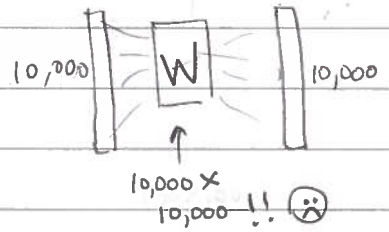
↖ floor

L 6-3

Weights are the values defining the filter

m^l different $K^l \times K^l \times m^{l-1}$ tensors of weight values

May seem complicated: but we get rich class of mappings that exploit image structure and have many fewer weights than a fully connected layer would:



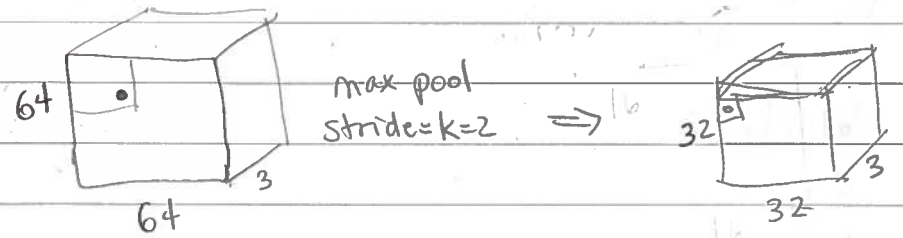
much more ~~data needed~~ ~~to train~~ fully connected network.

Max pooling

Filter pyramid has filters that recognize larger & more complex patterns by condensing images

Max pooling layer operates like a filter, but has no weights — returns the maximum value in its field. Usually

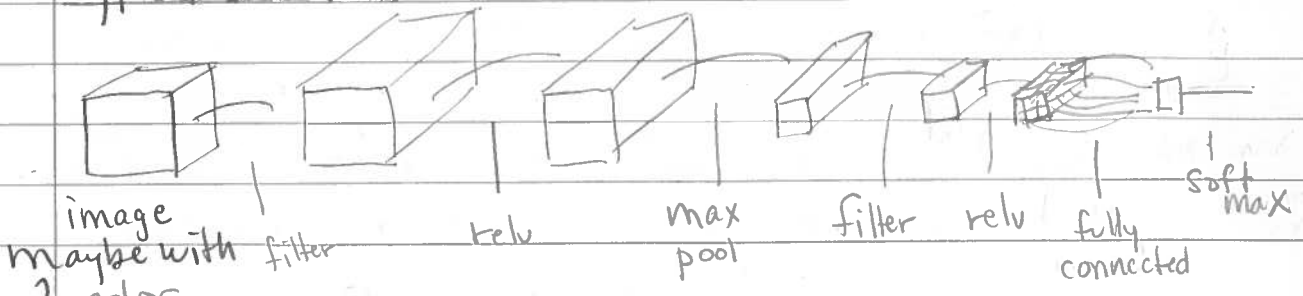
- stride > 1 $\Rightarrow$ resulting image is smaller
- $K \geq$ stride $\Rightarrow$ whole image is covered



- Lose precise location of pattern
- Filters of same K on resulting images have effective size 2K on original.

L6-4

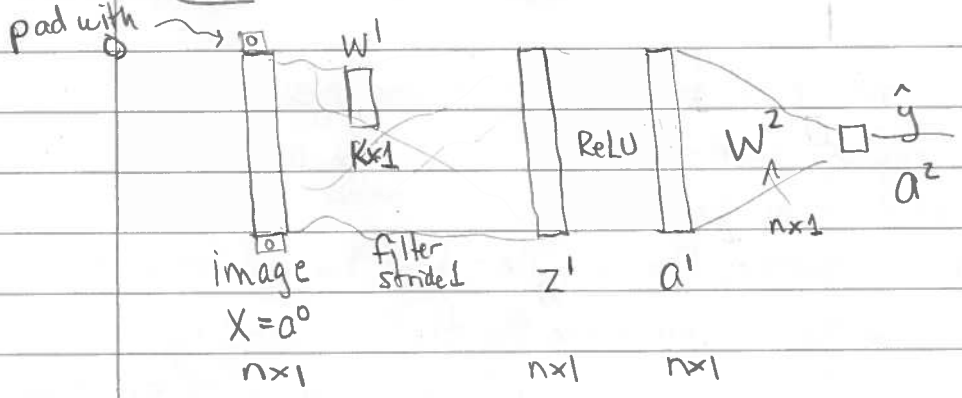
Typical architecture



Important point: this is all just a somewhat hairier neural network. output $\hat{y}$ is a differentiable function of input image and weights

$\Rightarrow$ we can do BACKPROPAGATION!

Simple backprop example



$$z^1_i = W^{1T} \cdot a^0_{[i-K/2 : i+K/2]}$$

$$a^1 = \text{ReLU}(z)$$

$$a^2 = W^{2T} a^1$$

$1 \times 1 \quad 1 \times n \quad n \times 1$

squared loss

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

Assume K is odd and, really we mean

$i - \lfloor K/2 \rfloor, \ldots, i + \lfloor K/2 \rfloor$ inclusive

so if $K = 5$, we have

$i-2, \ldots, i+2$

L 6-5

How do we update the weights in filter W^1?

$$\frac{\partial L}{\partial W^1} = \frac{\partial z^1}{\partial W^1} \cdot \frac{\partial a^1}{\partial z^1} \cdot \frac{\partial L}{\partial z^1}$$ the last z1 should be a1

$\frac{\partial L}{\partial z^1} \rightarrow 2(\hat{y} - y) W^2$

$n \times 1$

$$\begin{bmatrix} 0 & & \\ & & \\ & & 0 \end{bmatrix}$$ 0 or 1 on the diagonal where $z > 0$

$n \times n$

$\frac{\partial z^1}{\partial W^1} \rightarrow K \times n$ $\quad \frac{\partial z_i^1}{\partial w_j^1} = X_{i - \lfloor K/2 \rfloor + j}$

image chunk centered at i

$$\frac{\partial z^1}{\partial W^1} = \begin{bmatrix} & | & \\ & | & \\ & | & \end{bmatrix}_K$$

i

n