Intro

Example :

Vertical Edge detection.

$$\begin{bmatrix}3&0&1&2&7\\1&5&8&9&3\\2&7&2&5&1\\&&\vdots&&\end{bmatrix} * \begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix} = \begin{bmatrix}-5&4&0\\&\cdots&\\&&\end{bmatrix}$$

filter / kernel

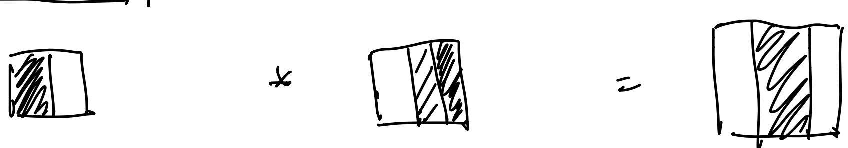
convolve operator.

python: conv-forward.
keras: Con2D
tensorflow: tf.nn.conv2d

5×5 3×3 3×3

Basically move the filter right one column at a time then do next row (sum of element-wise multiplication with filter)

$$\begin{bmatrix}10&10&0&0\\10&10&0&0\\10&10&0&0\\10&10&0&0\\10&10&0&0\\10&10&0&0\end{bmatrix} * \begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix} = \begin{bmatrix}0&30&30&0\\0&30&30&0\\0&30&30&0\\0&30&30&0\end{bmatrix}$$



detected edge

Horizontal detector

$$\begin{bmatrix}1&1&1\\0&0&0\\-1&-1&-1\end{bmatrix}$$

eg. $$\begin{bmatrix}10&10&0&0\\10&10&0&0\\0&0&10&10\\0&0&10&10\end{bmatrix} * \begin{bmatrix}1&1&1\\0&0&0\\-1&-1&-1\end{bmatrix} = \begin{bmatrix}0&0&0&0\\30&10&-10&-30\\30&10&-10&-30\\0&0&0&0\end{bmatrix}$$

Sobel filter

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \leftarrow \text{emphasize center pixel}$$

Schors filter

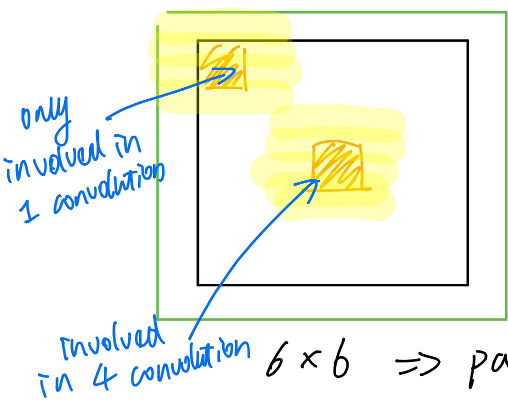
$$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$$

example of handpicked edge detector.

now researchers uses backprop to learn these filters

Padding

Solves 2 problem
- shrinking output → image get smaller and smaller
- through away info from edge → see blue ink



3×3

$* \text{ filter} = 4 \times 4$.

$6 \times 6 \Rightarrow$ pad 1 pixel at all edges $\Rightarrow 8 \times 8 \; * \text{filter} = 6 \times 6$

Valid convolution. (no padding)

$$n \times n \; * \; f \times f \longrightarrow (n-f+1) \times (n-f+1)$$

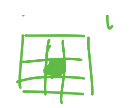
Same convolution (pad so that output size = input size)

$$(n+2p-f+1) \times (n+2p-f+1)$$

solve for $n+2p-f+1 = n$

padding size $\Rightarrow$ $p = \frac{f-1}{2}$

f is usually odd.
- sysmetric padding
- have central position



Stride Convolution

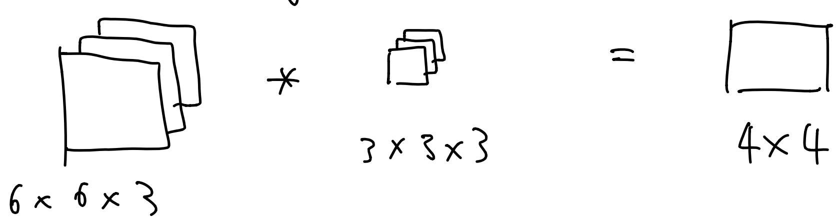
Basically number of pixels stepped each convolution.

$n \times n * f \times f \Rightarrow$ padding : p. stride : s

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

3-D convolution.

eg. RGB image.



for example, if you want to detect red vertical edge filter can be $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ then all 0 for blue and green channel.

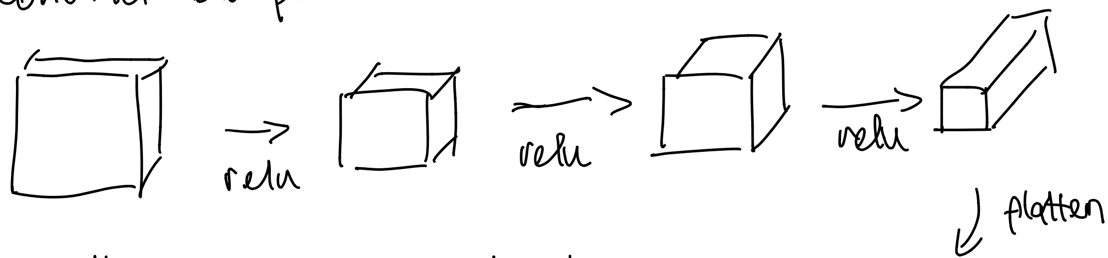
Multiple filter

stack output of apply each filter at 3D.

eg. output of horizontal, vertical edge detector will have dimension $4 \times 4 \times 2$ depth

\# of feature detecting

ConvNet Example



usually image gets smaller but features gets more.
lastly we flatten 3D output into a vector then → softmax to classify

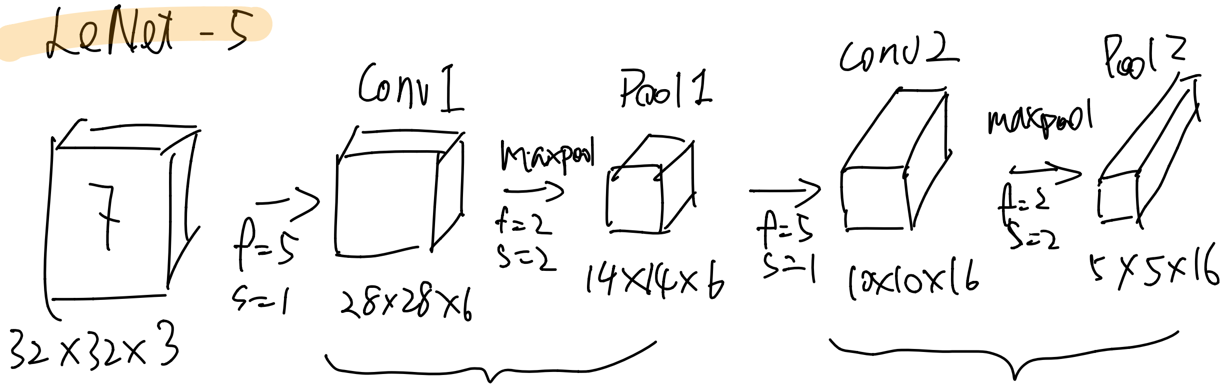
Pooling

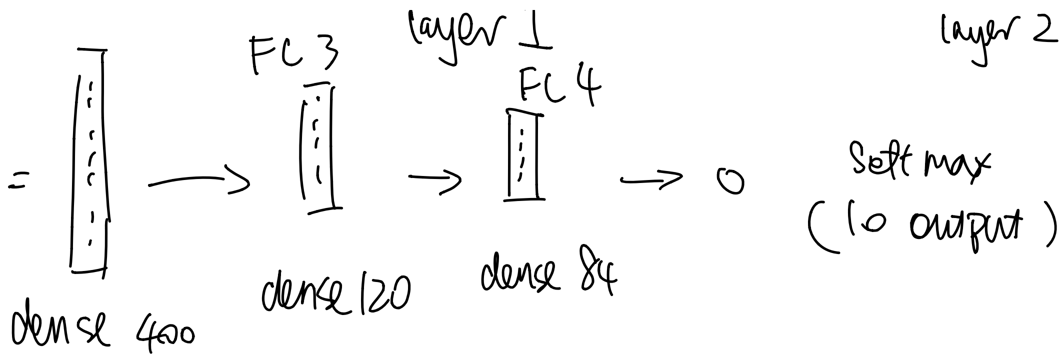
eg. max pooling with f = 2, s = 2.

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 9 & 2 \\ 6 & 3 \end{bmatrix}$$

Intuition: no param to learn. detect if some feature is detected in some region.

LeNet-5





Summary: conv → conv → fc → fc → softmax

where fc - fully-connected, conv - convolution with maxpooling

Observation:

height and # width go down, # channel go up (3rd Dimension)

Note:

- Activation size: 3072 → 6272 → 1568 → 1600 → 400 → 120 → 84 → 10
- # of parameter: 0, 208, 0, 416, 0, $\underbrace{48001, 10081, 841}_{\text{FC layers}}$

* Activation size decrease gradually.
* # of parameter concentrated at fully connected layers.

Why Convolution?

- parameter sharing : feature detector (eg. eye detector) useful in one part is also useful in another part of image ✓
- sparsity of connections : In each layer, the output depend on a small # of input. (since filter is small)

⇒ Thus, less parameter to train, less prone to overfitting.