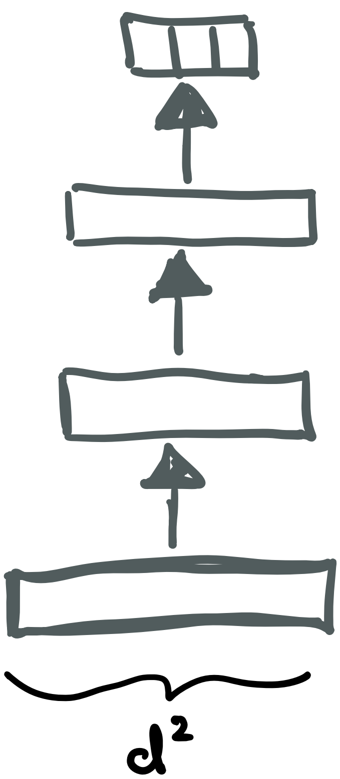
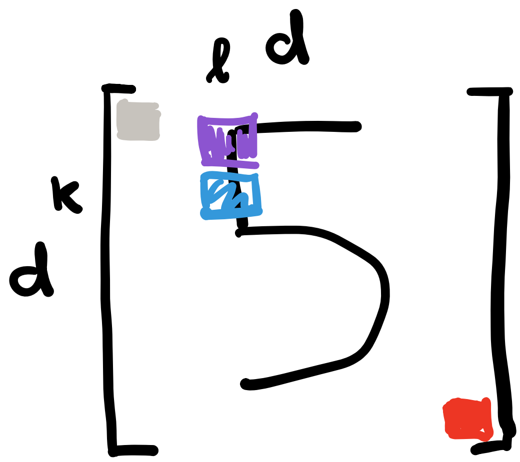
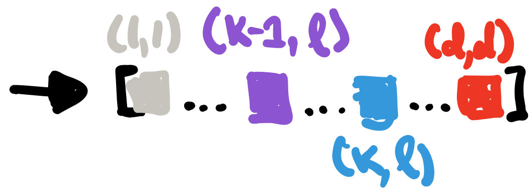
DS 4440 ConvNets

So far: Simple feed-forward Networks



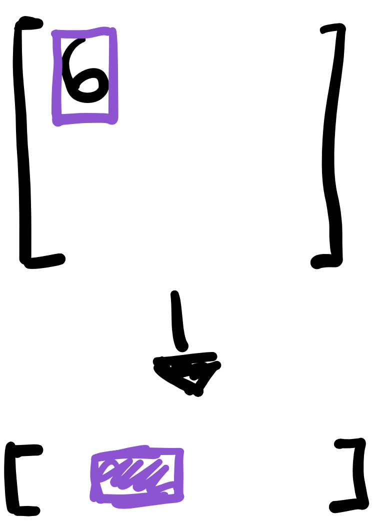




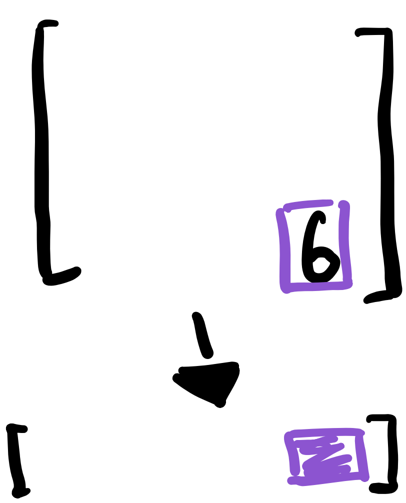
$X \in \mathbb{R}^{d^2}$

Problem: This discards structure

Consider



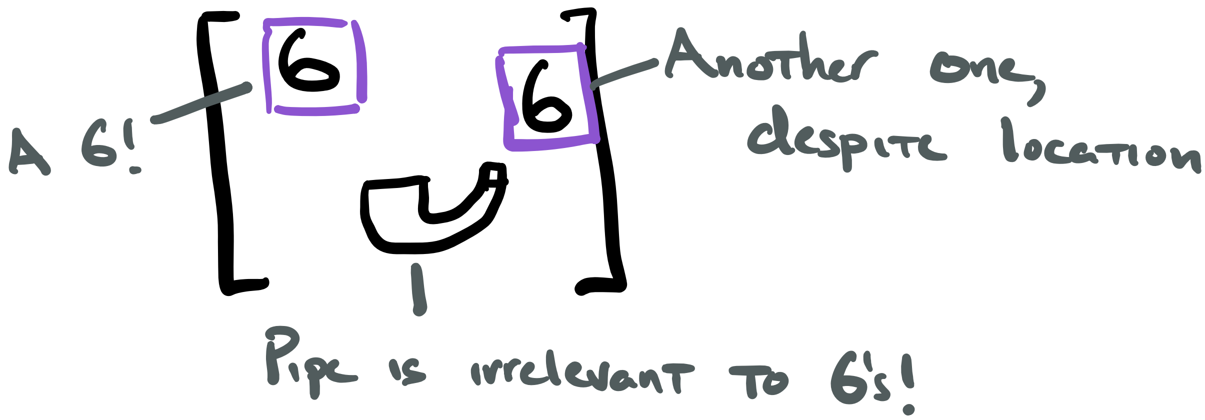
vs.



The parameters associated with these 6's are distinct!

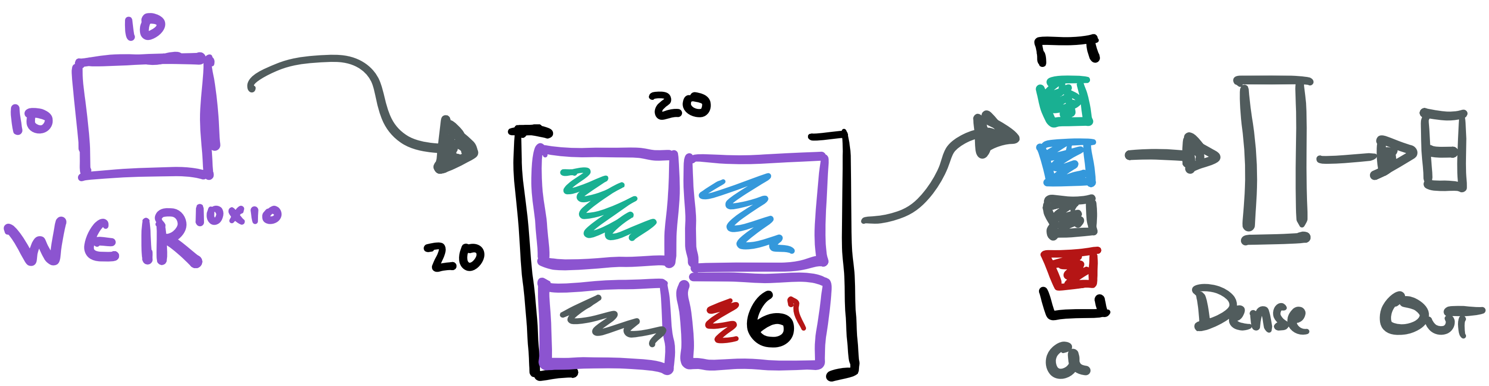
Upshot Model must learn to spot 6 in all locations.

But: a 6 is a 6! Invariance



Enter Convolutional Neural Networks (aka ConvNets aka CNNs)

Basic Idea: Slide Windows □ over inputs to yield local features



 = local features or activations

Formally assume region A starting at (A_x, A_y).

Sum over window size!

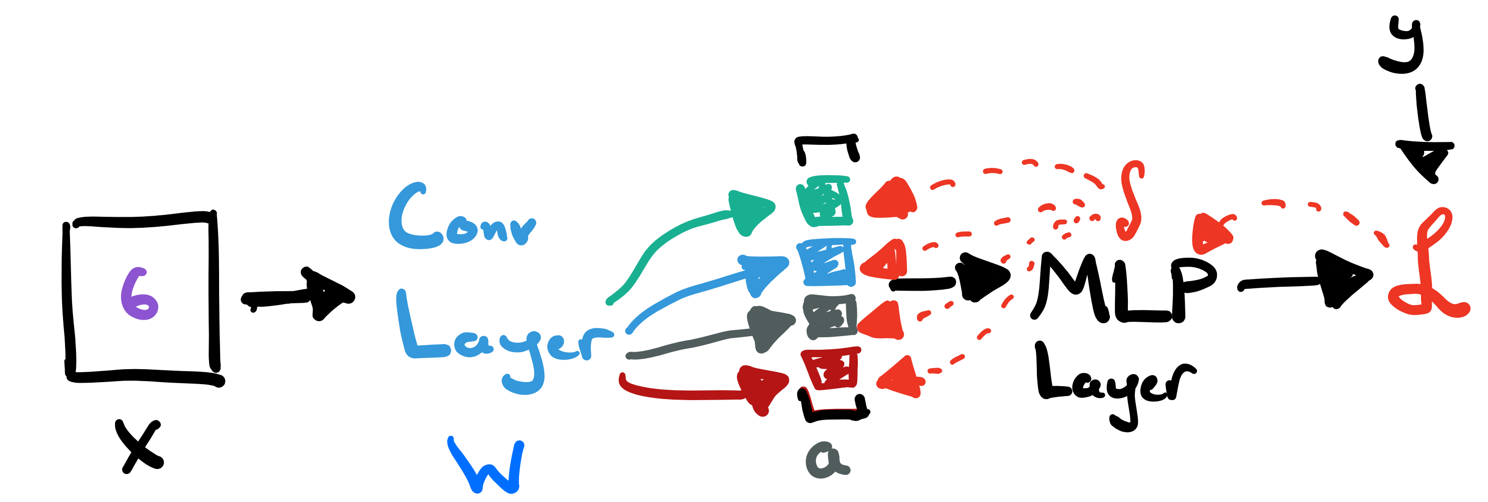
$$\square = \sigma\Big(\sum_{i,j} \underline{w}_{i+j} \cdot A_{ij}\Big)$$

$$= \sigma\Big(\sum_{i,j} \underline{w}_{i+j} X_{(A_x+j)(A_y+i)}\Big)$$

$$\square = \sigma\Big(\sum_{i,j} \underline{w}_{i+j} \cdot B_{ij}\Big)$$

The key is that weights w are shared across regions. W is called the kernel or filter.

Q What about $\nabla_w \mathcal{L}$ here?



$$\nabla_W \mathcal{L} = \sum_{a_k \in a} \left(\nabla_W a_k \cdot \frac{\partial \mathcal{L}}{\partial a_k} \right)$$

A Concrete example

* Adapted from "Deep Learning" by Goodfellow

$$\begin{array}{cc} & 4 \\ 3 & \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \end{array} \otimes \begin{array}{cc} & 2 \\ 2 & \begin{bmatrix} w & x \\ y & z \end{bmatrix} \end{array}$$

Input Kernel

$$\rightarrow \begin{array}{cc} & 3 \\ 2 & \begin{bmatrix} w \cdot a + x \cdot b + y \cdot e + z \cdot f & w \cdot b + x \cdot c + y \cdot f + z \cdot g & w \cdot c + x \cdot d + y \cdot g + z \cdot h \\ w \cdot e + x \cdot f + y \cdot i + z \cdot j & w \cdot f + x \cdot g + y \cdot j + z \cdot k & w \cdot g + x \cdot h + y \cdot k + z \cdot l \end{bmatrix} \end{array}$$

Activation

In the simplist case – as here –

Output size is

$$(\underbrace{X_h}_{\text{Input height}} - \underbrace{W_h}_{\text{Kernel height}} + 1,\ \underbrace{X_w}_{\text{Input width}} - \underbrace{W_w}_{\text{Kernel width}} + 1)$$

Padding

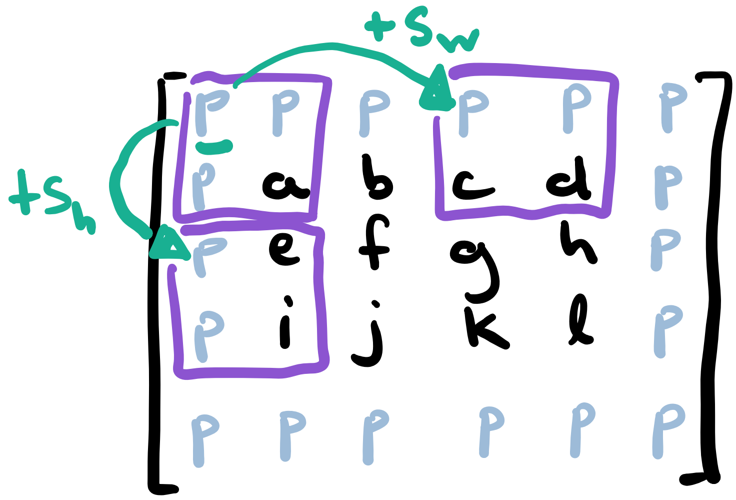
$$\begin{bmatrix} p & p & p & p & p & p \\ p & a & b & c & d & p \\ p & e & f & g & h & p \\ p & i & j & k & l & p \\ p & p & p & p & p & p \end{bmatrix}$$

Here we have padded height & width by 1.

→ Output shape

$$(X_h + P_h - W_h + 1,\ X_w + P_w - W_w + 1)$$

Stride size has to do with how we pass kernels over inputs – have assumed size of 1 above.



$S_w = 3$

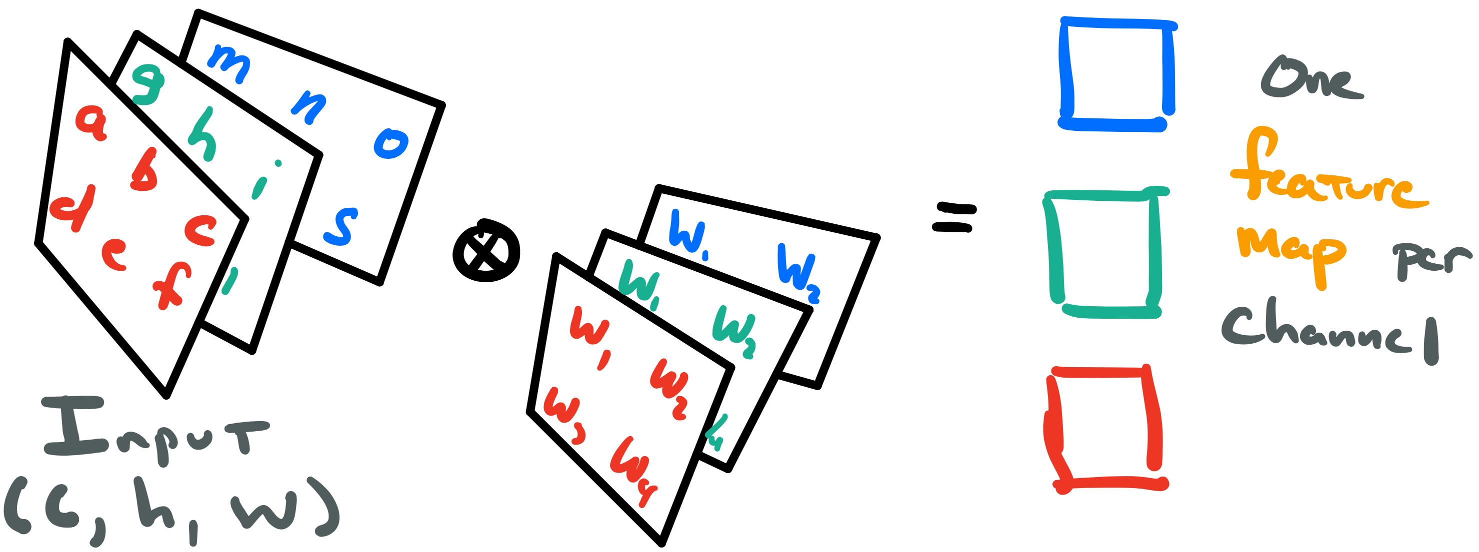
$S_h = 2$

Also affects output size

$$\left(\frac{(x_h + P_h - W_h + S_h)}{S_h}, \frac{(x_w + P_w - W_w + S_w)}{S_w} \right)$$

Higher Dimensions

Above we assumed 2d inputs but we can extend to arbitrary tensors. Useful, e.g. for RGB images.



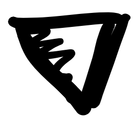
Can Sum for final map, or use another pooling strategy.

feature map $\rightarrow$ Pool $\rightarrow$ Smaller (often Scalar) output f

Max pooling extract max value

$$\underbrace{\begin{bmatrix} 10 & 5 & 3 \\ 7 & 6 & 2 \\ 14 & 8 & 90 \end{bmatrix}}_{\text{feature map}} \rightarrow \begin{bmatrix} 10 & 6 \\ 14 & 90 \end{bmatrix}$$

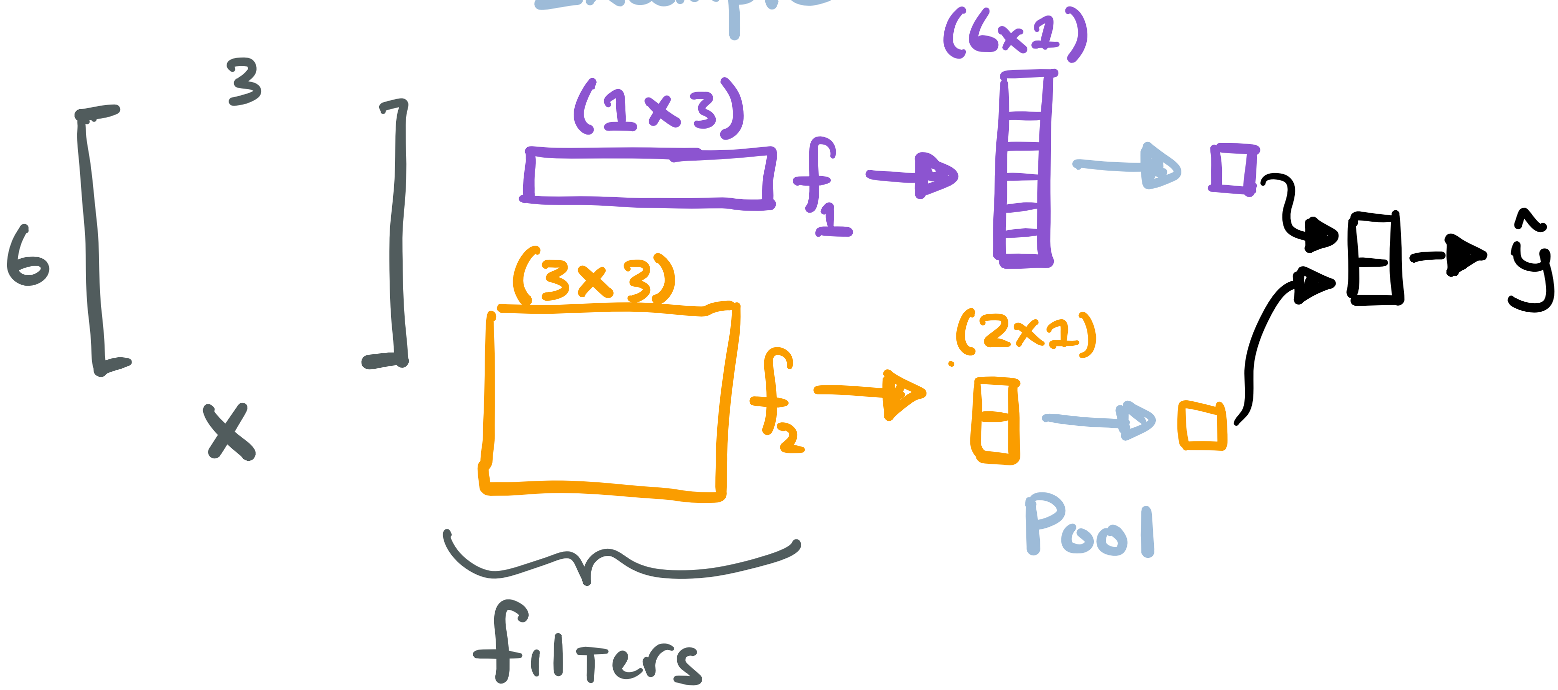
$2\left[\ \overset{2}{}\ \right]$ 2x2 Max Pool

Q What happens to  through Pooling (max) operation?

Intuition Individual filters may Specialize at recognizing something (Say, cats) $\rightarrow$ Max pooling is like asking "does this window contain a cat?" repeatedly (More on HW4!)

Often we define multiple independent filters, with same or varying size.

Example



Concretely, consider:

$$\begin{bmatrix} 5 & 3 & 1 \\ 6 & 8 & 1 \\ 2 & 3 & 4 \end{bmatrix}$$
$$x$$

- No padding
- Two (1x2) filters
- Max pooling (1x2)

$[6 \;\; 8]\, f_1 \qquad [-2 \;\; 3]\, f_2$

$$\begin{bmatrix} 6\cdot 5 + 8\cdot 3 & 6\cdot 3 + 8\cdot 1 \\ 6\cdot 6 + 8\cdot 8 & 6\cdot 8 + 8\cdot 2 \\ 6\cdot 2 + 8\cdot 3 & 6\cdot 3 + 8\cdot 4 \end{bmatrix} = \begin{bmatrix} 54 & 26 \\ 100 & 56 \\ 36 & 50 \end{bmatrix} \rightarrow \begin{bmatrix} 54 \\ 100 \\ 50 \end{bmatrix}$$

feature map 1

Pooling (1x2)

$$\begin{bmatrix} -2\cdot 5 + 3\cdot 3 & -2\cdot 3 + 3\cdot 1 \\ -2\cdot 6 + 3\cdot 8 & -2\cdot 8 + 3\cdot 1 \\ -2\cdot 2 + 3\cdot 3 & -2\cdot 3 + 3\cdot 4 \end{bmatrix} = \begin{bmatrix} -1 & -3 \\ 12 & -13 \\ 5 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ 12 \\ 6 \end{bmatrix}$$

$$\oplus \longrightarrow [54 \quad 100 \quad 50 \quad -1 \quad 12 \quad 6]$$