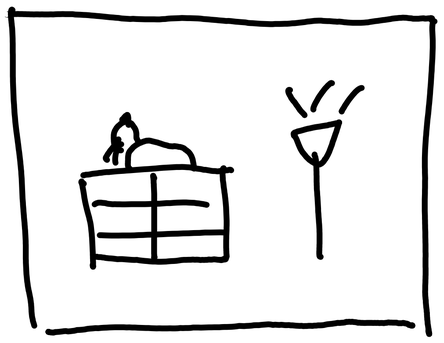
Convolutional Neural Networks



Does this image contain a cat?

1. Translation invariance
   (respond similarly everywhere)
2. Locality
   (only consider a "small" region)

$$h = u + W x$$

$$h_m = u_m + \sum_n W_{m,n} x_n$$

$$H_{i,j} = U_{i,j} + \sum_k \sum_\ell W_{i,j,k,\ell} X_{k,\ell}$$

$i \cdot j = m \,,\; k \cdot \ell = n$

$$H_{i,j} = U_{i,j} + \sum_a \sum_b V_{i,j,a,b} X_{i+a,j+b}$$

↳ can be positive or negative to go over whole image

Translation invariance:

$$H_{i,j} = U + \sum_a \sum_b V_{a,b} X_{i+a,j+b}$$

Locality:

$$H_{i,j} = U + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} V_{a,b} X_{i+a,j+b}$$

"kernel" "filter"

How much is V present at i, j?

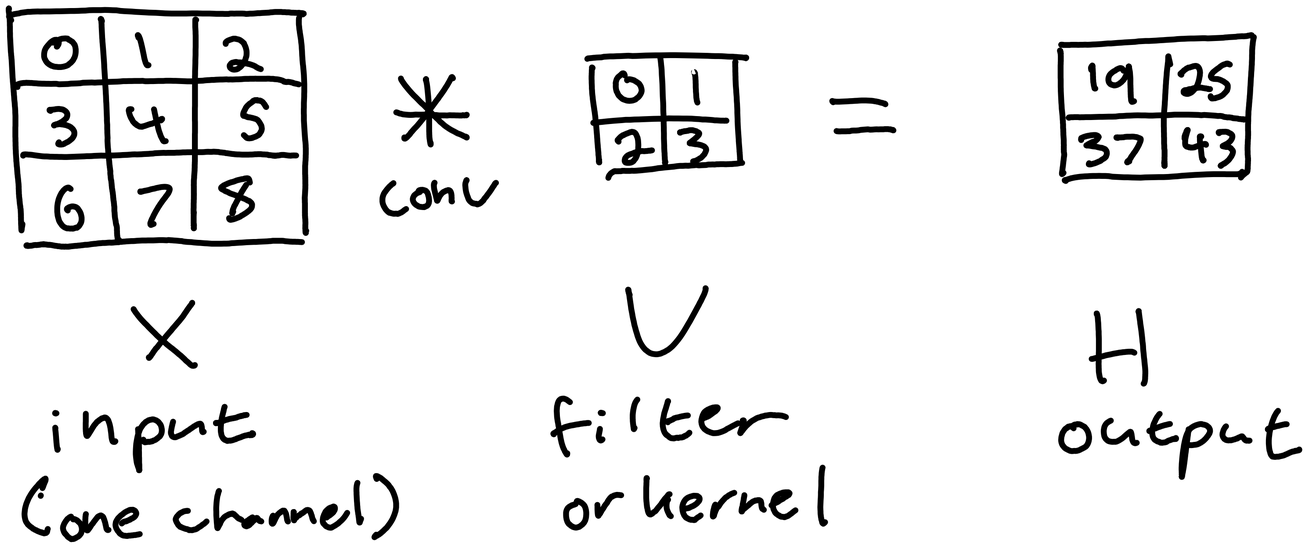
Convolution!

Channels: Images are 2D grids of pixels with one or more channels (RGB)

In convnets, feature representations (activations) have channels.

$$H_{i,j,d} = U_{c,d} + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{c} V_{a,b,c,d} \, X_{i+a,\, j+b,\, c}$$

- $H_{i,j,d}$: output channel
- $U_{c,d}$: ???
- $\sum_{c}$: sum over input channels
- $X_{i+a,\, j+b,\, c}$: input channel



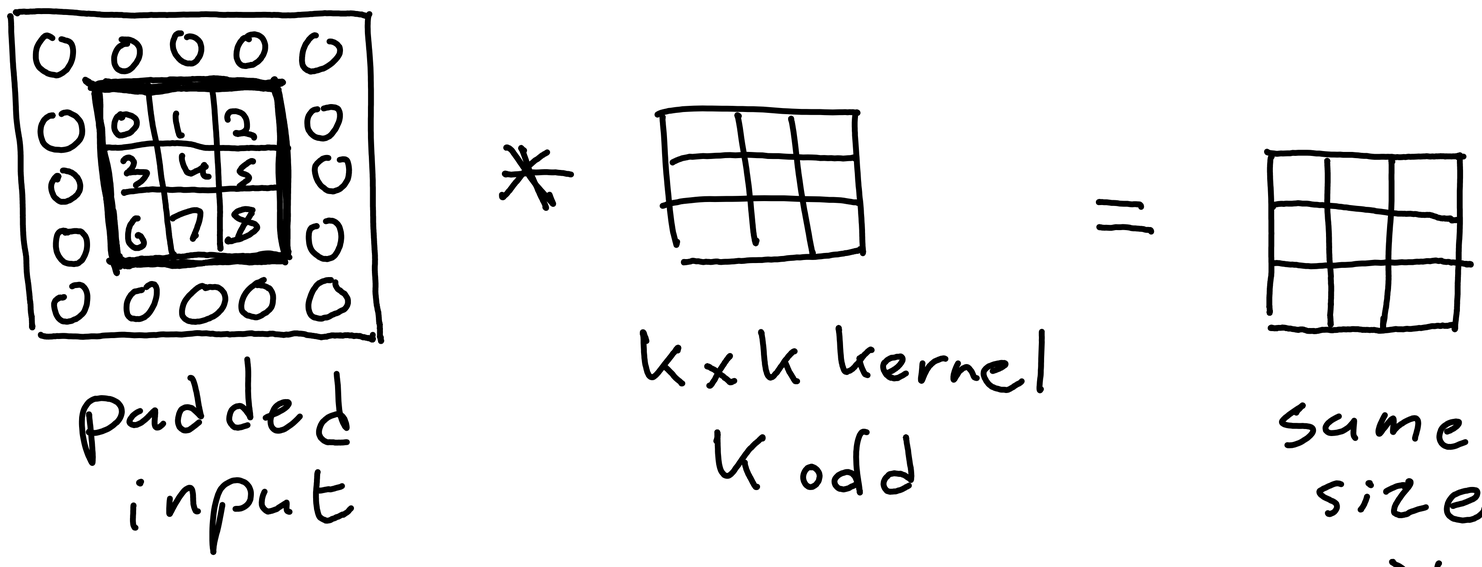
Notes:

1. Output is smaller than the input.

$$\text{output size} = (\underbrace{n_h}_{\substack{\text{input}\\\text{height}}} - \underbrace{k_h}_{\substack{\text{kernel}\\\text{height}}} + 1) \times (\underbrace{n_w}_{\substack{\text{input}\\\text{width}}} - \underbrace{k_w}_{\substack{\text{kernel}\\\text{width}}} + 1)$$

2. "Receptive field": Region in the input that a given position in the output depends on

Padding: Add zeros around the border of the input so that the output is the same size.



padded input

$*$

k x k kernel
k odd

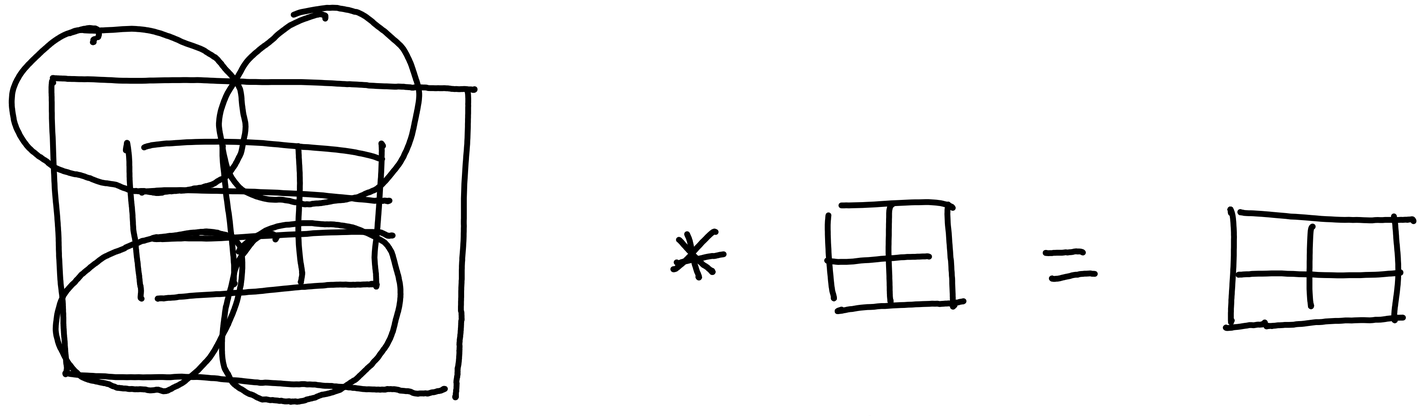
$=$

same-size

total padding on axis: $k-1$ ← kernel width or height

on each edge, we need $\frac{k-1}{2}$ zeros.

Stride:



* =

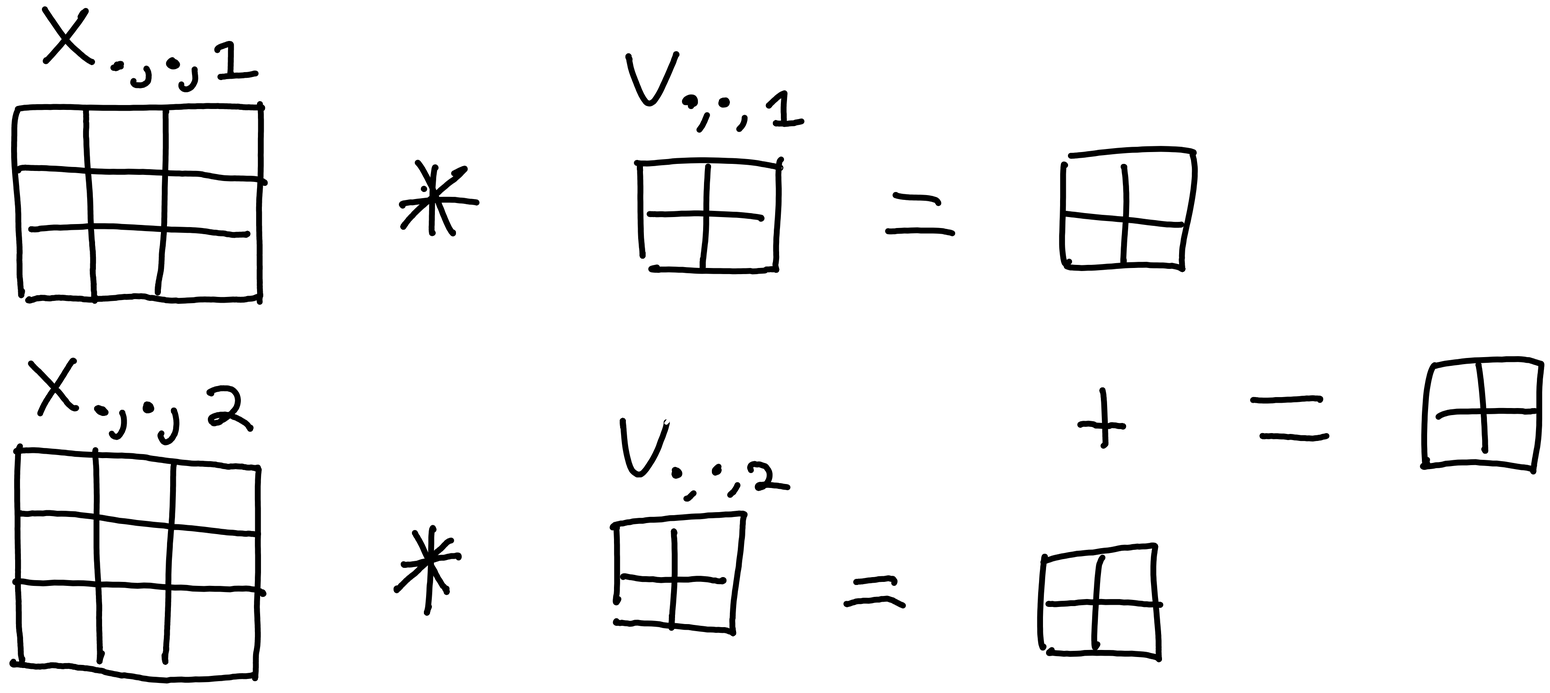
↗ amount we move horizontally

Stride of 2_{x3}

↳ amount we move vertically

- output gets smaller
  "down sampling"
- cheaper!

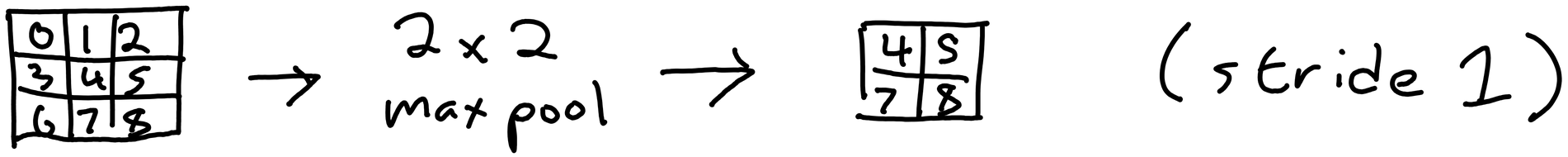
Channels



- Separate kernel for each input/output channel combination
- Sum the result over input channels

Pooling: Aggregate spatial information

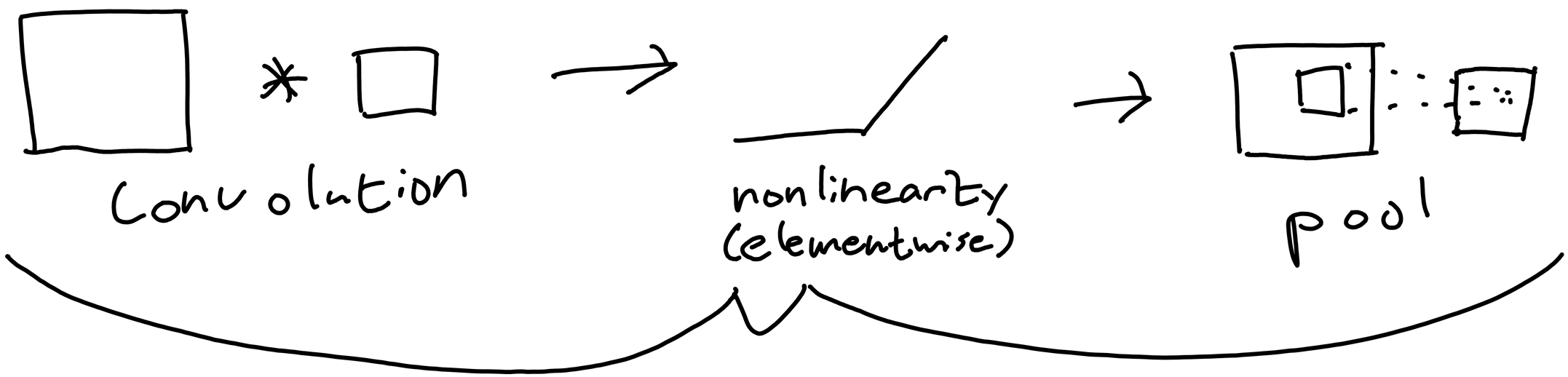
max pooling:



When performing pooling, almost always stride by pooling size

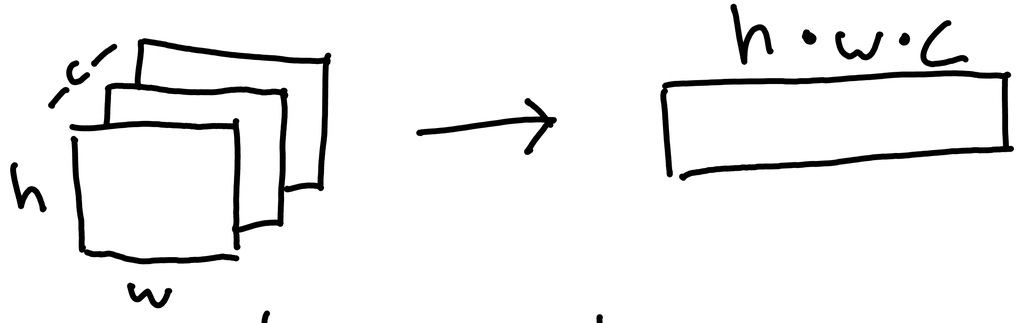
average pooling is also not uncommon.

Simple ConvNet structure:



Eventually we need to perform classification.

1. Flatten (7 x 7 x 100 → 4900)



reshape(·,-1)

2. Apply dense layers (like an MLP)

3. $BN(x) = \gamma \odot \frac{x - \mu_B}{\sigma_B + \varepsilon} + \beta$ (learned params) $\quad \gamma, \beta \in \mathbb{R}^{dim(x)}$

4. At inference time, use EMA of μ_B and σ_B computed over traing.

Batch norm for images:

$$x \in \mathbb{R}^{b \times h \times w \times c}$$

$$\mu = \text{mean}(x, \text{axis}=(0,1,2)) \in \mathbb{R}^c$$

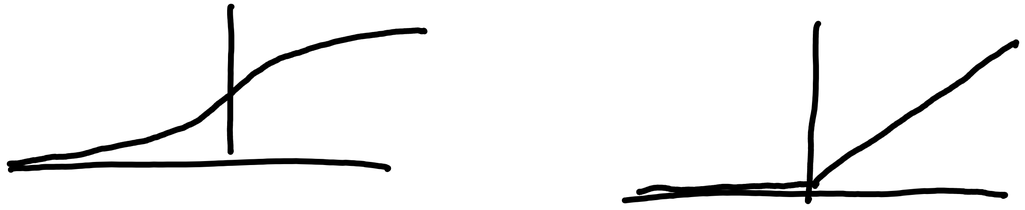
$$\sigma = \text{std}(x, \text{axis}=(0,1,2)) \in \mathbb{R}^c$$

$$BN(x) = \gamma \odot \frac{x - \mu}{\sigma + \varepsilon} + \beta$$

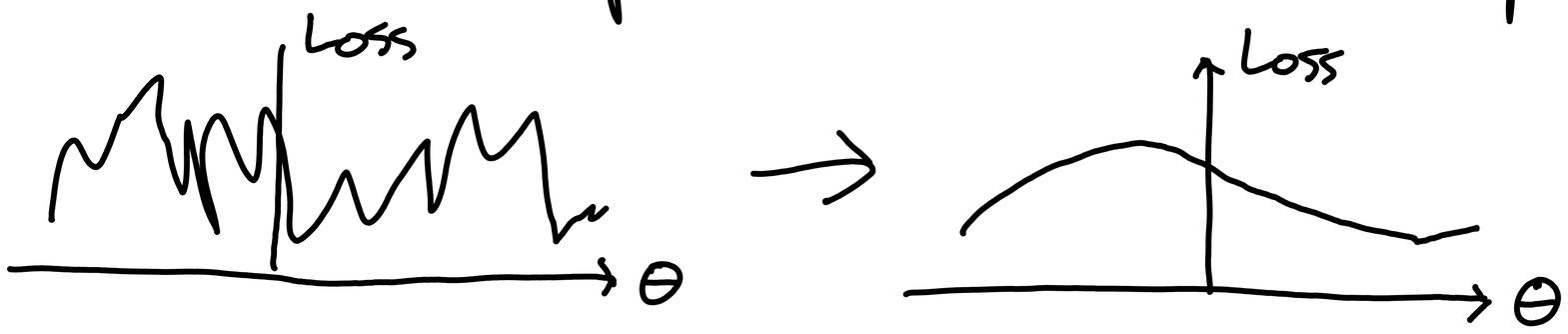
Why batch norm?

1. "Reduces internal covariate shift"
   → But in practice, this doesn't happen

2. Moves the activations toward the non-saturated region of the nonlinearity



3. Smooths the optimization landscape



4. Regularization due noisy estimate of mean and variance

Residual Connections

Say we have a model and we want to add a layer, but we only want to enlarge the set of functions it can implement. Alternatively, we want to propagate gradients as if the layer is not there.

Say x is the output of orig. model

$f(\cdot)$ is the new layer

"Residual connection"

$f(x) + x$