A Small Example

\[ \hat{\mathbf{x}} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix} \]

\[ a_i = \sum_{\ell=0}^{2} g_{i+\ell} x_i \quad \text{for} \quad i = 0, \ldots, 5 \]

\[ a = Vx = \begin{bmatrix} g_0 & g_1 & g_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & g_0 & g_1 & g_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_0 & g_1 & g_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 \end{bmatrix} x \]
2D Correlation or Convolution

\[ a_{ij} = \sum_{\ell=0}^{k_1-1} \sum_{m=0}^{k_2-1} g_{\ell m} x_{i+\ell,j+m} + b_{ij} \]

- Output is \(2 \times 5\)
- Can be made to be \(4 \times 6\) using padding ("same" correlation)
Stride

- Activation $a_{ij}$ is often similar to $a_{i,j+1}$ and $a_{i+1,j}$
- Images often vary slowly over space
- Reduce the redundancy in the output by computing convolutions with a stride $s_m$ greater than one
- Only compute every $s_m$ output values in dimension $m$
- Output size shrinks from $d_1 \times d_2$ to about $d_1/s_1 \times d_2/s_2$
- Typically $s_m = s$ (same stride in all dimensions)
- Layers get smaller and smaller because of stride
Max Pooling

- Another way to reduce output resolution is *max pooling*
- This is a layer of its own, separate from convolution
- Consider $k \times k$ windows with stride $s$
- Often $s = k$ (adjacent, non-overlapping windows)
- For each window, output the maximum value
- Output is about $d_1/s \times d_2/s$
- Returns highest response in window, rather than the response in a fixed position
- Loosely analogous to using cells and histograms in HOG
The Input Layer of AlexNet

- AlexNet *circa* 2012, classifies color images into one of 1000 categories
- Trained on ImageNet, a large database with millions of labeled images

Channels = # of feature maps

\[ y = \pi(\rho(a)) \]
A more Compact Drawing

input $x$  

response maps $\rho(a)$  

convolution kernels  

feature maps $a$  

receptive field of convolution  

max pooling  

output $y = \pi(\rho(a))$  

224  

224  

55  

55  

32  

11  

11  

channels
AlexNet Numbers

- Input is $224 \times 224 \times 3$ (color image)
- First layer has 96 feature maps of size $55 \times 55$
- A fully-connected first layer would have about $224 \times 224 \times 3 \times 55 \times 55 \times 96 \approx 4.4 \times 10^{10}$ weights
- With convolutional kernels of size $11 \times 11$, there are only $96 \times 11^2 = 11,616$ weights
- That’s a big deal! Locality and reuse
- Most of the complexity is in the last few, fully-connected layers, which still have millions of parameters
- More recent neural networks have much lighter final layers, but many more layers
- There are also fully convolutional neural networks
Output

- For *regression*, the output of the network is the desired quantity.
- The last layer of a neural net used for *classification* is a *soft-max* layer:
  $$ p = \sigma(y) = \frac{e^y}{\sum_i e^{y_i}} $$
- As many entries in $y$ and $p$ as there are classes.
- One output score $p$ per class for classification.
- Classify by $\text{arg max } p$. 
The Soft-Max Function

\[ p_k(y) = \frac{e^{y_k}}{\sum_{j=1}^{K} e^{y_j}} \quad \text{or} \quad p(y) = \frac{e^y}{1^T e^y} \]

- \( y \in \mathbb{R}^K \rightarrow p \in \mathbb{R}^K \)
- \( p_k(y) > 0 \) and \( \sum_{k=1}^{K} p_k(y) = 1 \) for all \( y \)
- If \( y_i \gg y_j \) for \( j \neq i \) then \( \sum_{j=1}^{K} e^{y_i} \approx e^{y_j} \)
- Therefore, \( p_i \approx 1 \) and \( p_j \approx 0 \) for \( j \neq i \)
- “Brings out the biggest:” soft-max

\[ \lim_{\alpha \to \infty} y^T p(\alpha y) = \max(y) \]