Neural networks

Computer vision - convolutional network

COMPUTER VISION

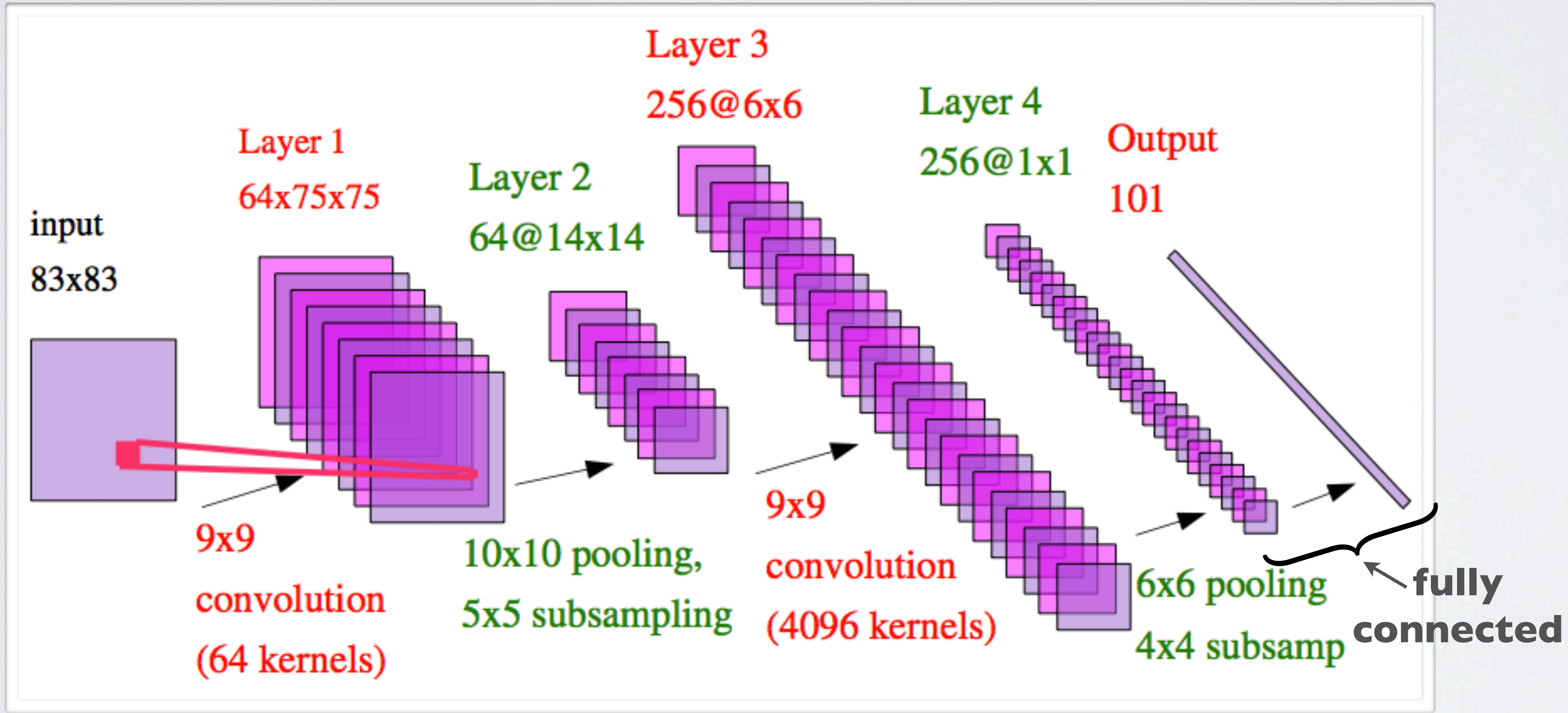
Topics: computer vision

- We can design neural networks that are specifically adapted for such problems
  - must deal with very high-dimensional inputs
    - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build in invariance to certain variations we can expect
    - translations, illumination, etc.
- Convolutional networks leverage these ideas
  - local connectivity
  - parameter sharing
  - pooling / subsampling hidden units

CONVOLUTIONAL NETWORK

Topics: convolutional network

- Convolutional neural network alternates between the convolutional and pooling layers



(from Yann Lecun)

CONVOLUTIONAL NETWORK

Topics: convolutional network

- Output layer is a regular, fully connected layer with softmax non-linearity
  - output provides an estimate of the conditional probability of each class

- The network is trained by stochastic gradient descent
  - backpropagation is used similarly as in a fully connected network
  - we have seen how to pass gradients through element-wise activation function
  - we also need to pass gradients through the convolution operation and the pooling operation

CONVOLUTIONAL NETWORK

Topics: gradient of convolution layer

- Let l be the loss function
  - for convolution operation $y_j = x_i * k_{ij}$ the gradient for x_i is

$$\nabla_{x_i} l = \sum_j (\nabla_{y_j} l) \underline{*} (W_{ij})$$

and the gradient for W_{ij} is

$$\nabla_{W_{ij}} l = (\nabla_y l) * \widetilde{x}_i$$

where $\underline{*}$ is the convolution with zero padding and $\widetilde{x}_i$ is the row/column flipped version of x_i

CONVOLUTIONAL NETWORK

Topics: gradient of pooling layer

- Let l be the loss function
  - for max pooling operation $y_{ijk} = \max_{p,q} x_{i,j+p,k+q}$ the gradient for x_{ijk} is

$$\nabla_{x_{ijk}} l = 0 \text{ except for } \nabla_{x_{i,j+p',k+q'}} l = \nabla_{y_{ijk}} l$$

where $p',\ q' = \operatorname{argmax} x_{i,j+p,k+q}$

- in words, only the "winning" units in layer x get the gradient from the pooled layer

  - for average pooling operation $y_{ijk} = \frac{1}{m^2} \sum_{p,q} x_{i,j+p,k+q}$ the gradient for x_{ijk} is

$$\nabla_x l = \frac{1}{m^2} \mathrm{upsample}(\nabla_y l)$$

where $\mathrm{upsample}$ inverts subsampling