Neural networks

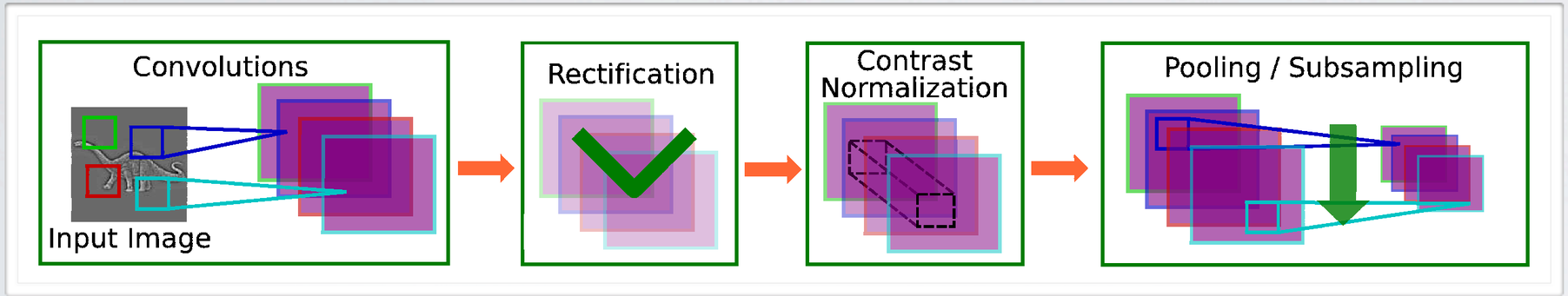
Computer vision - example

CONVOLUTIONAL NETWORK

Topics: convolutional network

Jarret et al. 2009

- These operations are inserted after the convolutions and before the pooling



- Images should also be preprocessed by
  - converting to grayscale (if appropriate)
  - resizing images to 150 x 150 pixels (use zero padding for non-square images)
  - removing (intra image) mean and dividing by standard deviation of the image
  - applying local contrast normalization

CONVOLUTIONAL NETWORK

Topics: initialization of parameters

Jarret et al. 2009

- Initialization of parameters:
  - can do as in regular neural network and initialize them randomly
  - can also use unsupervised pretraining approach
    - to use unsupervised neural networks we've seen so far, we have to convert pretraining as a patch-wise learning problem
      - ✓ extract patches of the same as the receptive fields of the hidden units, at random positions
      - ✓ train an unsupervised neural network (RBM, autoencoder, sparse coding) on those patches
      - ✓ use weights connecting an input patch to each hidden unit to initialize each feature map parameters
      - ✓ map images through all feature maps and repeat previous steps, for as many layers as desired
- We will compare:
  - using random initialization (R) or unsupervised pretraining (U)
  - using fine-tuning of whole network ($^+$) or only training output layer (no $^+$)

CONVOLUTIONAL NETWORK

Topics: convolutional network

Jarret et al. 2009

- Results on Caltech: F_{CSG} = convolution layer
  R = rectification layer
  N = local contrast normalization layer
  P_M = max pooling layer, P_A = average pooling layer

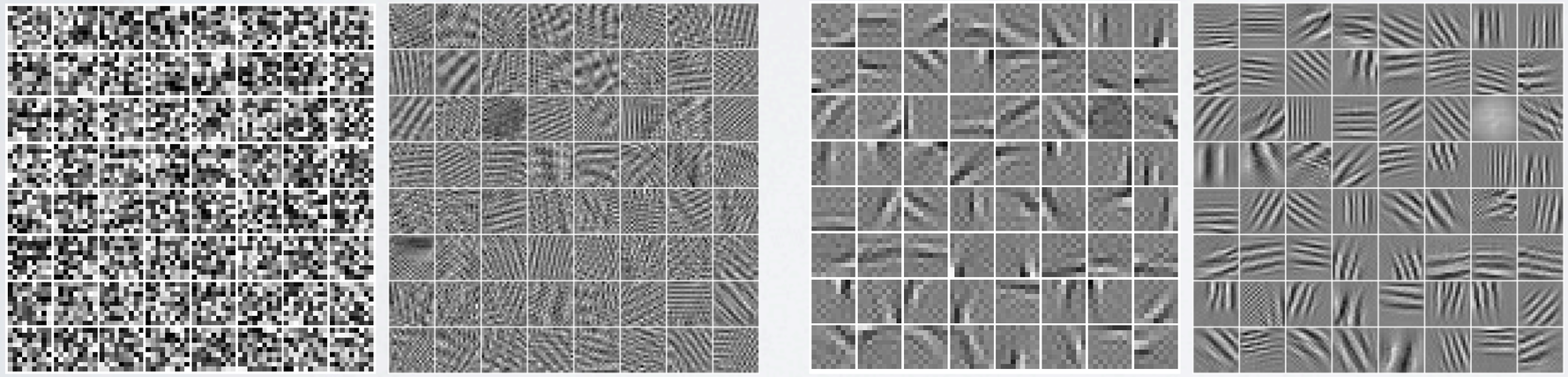
Single Stage System: $[64.\mathbf{F}^{9\times9}_{\mathbf{CSG}} - \mathbf{R/N/P}^{5\times5}]$ **- log_reg**					
	$\mathbf{R_{abs} - N - P_A}$	$\mathbf{R_{abs} - P_A}$	$\mathbf{N - P_M}$	$\mathbf{N - P_A}$	$\mathbf{P_A}$
$\mathbf{U^+}$	54.2%	50.0%	44.3%	18.5%	14.5%
$\mathbf{R^+}$	54.8%	47.0%	38.0%	16.3%	14.3%
$\mathbf{U}$	52.2%	43.3%($\pm$1.6)	44.0%	17.2%	13.4%
$\mathbf{R}$	53.3%	31.7%	32.1%	15.3%	12.1%($\pm$2.2)
$\mathbf{G}$	52.3%				
Two Stage System: $[64.\mathbf{F}^{9\times9}_{\mathbf{CSG}} - \mathbf{R/N/P}^{5\times5}] - [256.\mathbf{F}^{9\times9}_{\mathbf{CSG}} - \mathbf{R/N/P}^{4\times4}]$ **- log_reg**					
	$\mathbf{R_{abs} - N - P_A}$	$\mathbf{R_{abs} - P_A}$	$\mathbf{N - P_M}$	$\mathbf{N - P_A}$	$\mathbf{P_A}$
$\mathbf{U^+U^+}$	65.5%	60.5%	61.0%	34.0%	32.0%
$\mathbf{R^+R^+}$	64.7%	59.5%	60.0%	31.0%	29.7%
$\mathbf{UU}$	63.7%	46.7%	56.0%	23.1%	9.1%
$\mathbf{RR}$	62.9%	33.7%($\pm$1.5)	37.6%($\pm$1.9)	19.6%	8.8%
$\mathbf{GT}$	55.8%				

CONVOLUTIONAL NETWORK

Topics: random filters

Jarret et al. 2009

- Results on Caltech:
  - random filters are surprisingly good
  - turns out that random filters give units that are still sensitive to a particular frequency
    - can analyze this by finding input which maximizes the activation of a given hidden unit (with gradient ascent applied in input space)



random filters | optimal input | learned filters | optimal input

CONVOLUTIONAL NETWORK

Topics: importance of architecture

Jarret et al. 2009

- Results on Caltech:
  - choice of right architecture can be more important than learning algorithm
    - the use of rectification and local contrast normalization layers is important
    - this is particularly true if little training data
- Results on NORB:
  - architecture makes less of a difference with lots of training data per class
  - random filters are also not as good