COSC579: CNNs in Computer Vision

Jeremy Bolton, PhD

Assistant Teaching Professor

GEORGETOWN UNIVERSITY

Outline

I. Neural Networks: a brief review
   I. Back Propagation

II. Neural Networks in Computer Vision
   I. Convolutional Neural Networks and Structure
   II. Overfitting, overtraining, and regularization
      I. Norm regularization and sparsity
      II. Data augmentation
      III. Dropout

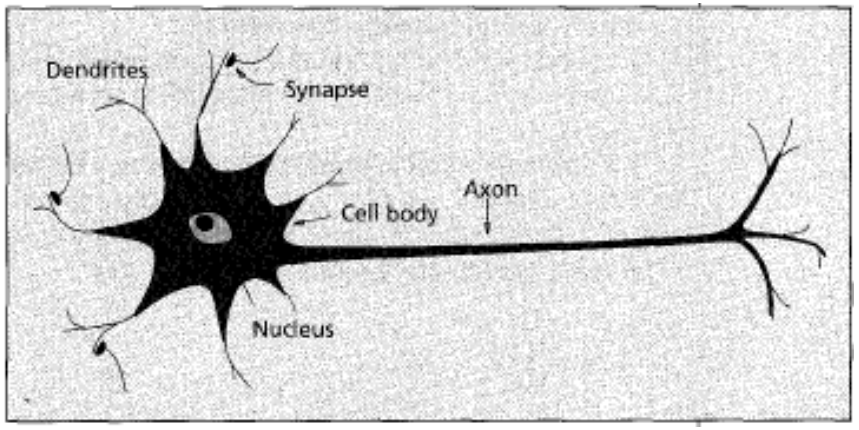
III. Thank you and Read
   I. Anil Jain (Read standard tutorial)
   II. Mittal
   III. Goodfellow

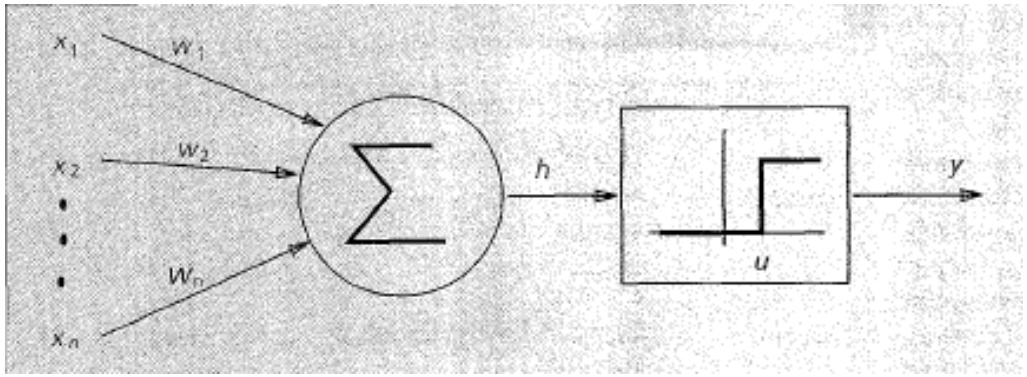
GEORGETOWN UNIVERSITY

Readings

- Nielson CHs 1 – 6
  - http://neuralnetworksanddeeplearning.com
- Goodfellow CHs 6 – 9
  - http://www.deeplearningbook.org/

Biologically Inspired Perceptron

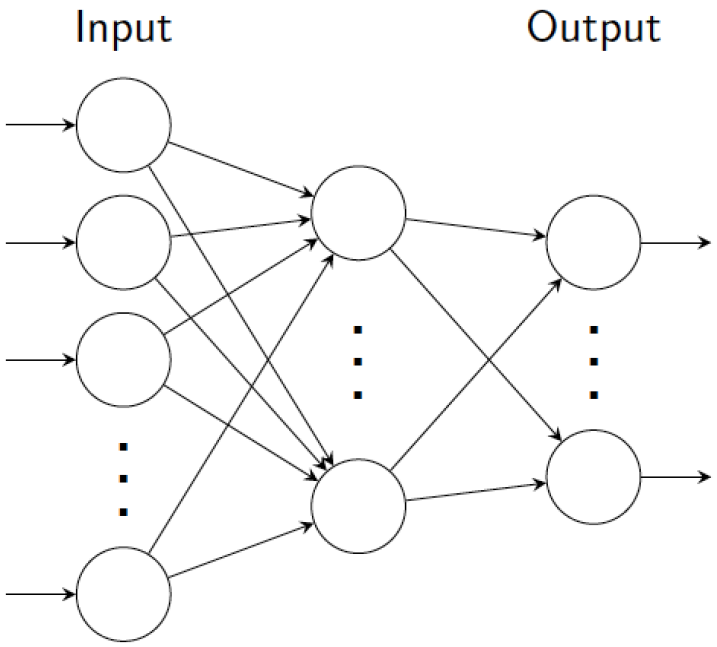




GEORGETOWN UNIVERSITY

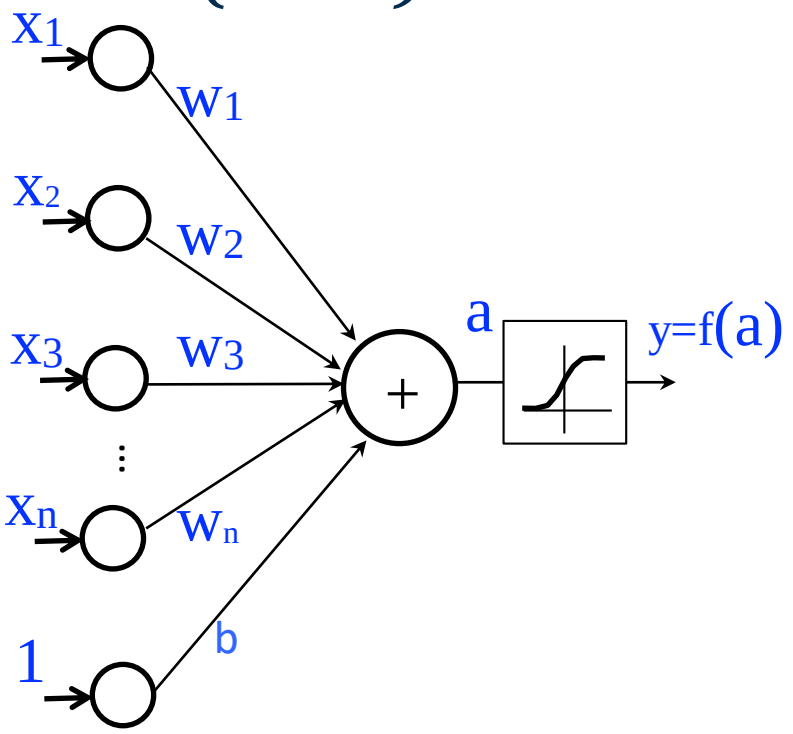
Neural networks

- Neural nets composed of layers of artificial neurons.
- Each layer computes some function of layer beneath.
- Inputs mapped in feed-forward fashion to output.
- Consider only feed-forward neural models at the moment, i.e. no cycles



An individual neuron (unit)

- Input: vector x (size n×1)
- Unit parameters: vector w (size n×1)
  bias b (scalar)
- Unit activation: $a = \sum_{i=1}^{n} x_i w_i + b$
- Output: $y = f(a) = f\left(\sum_{i=1}^{n} x_i w_i + b\right)$



f(.) is a non-linear function. E.g.:

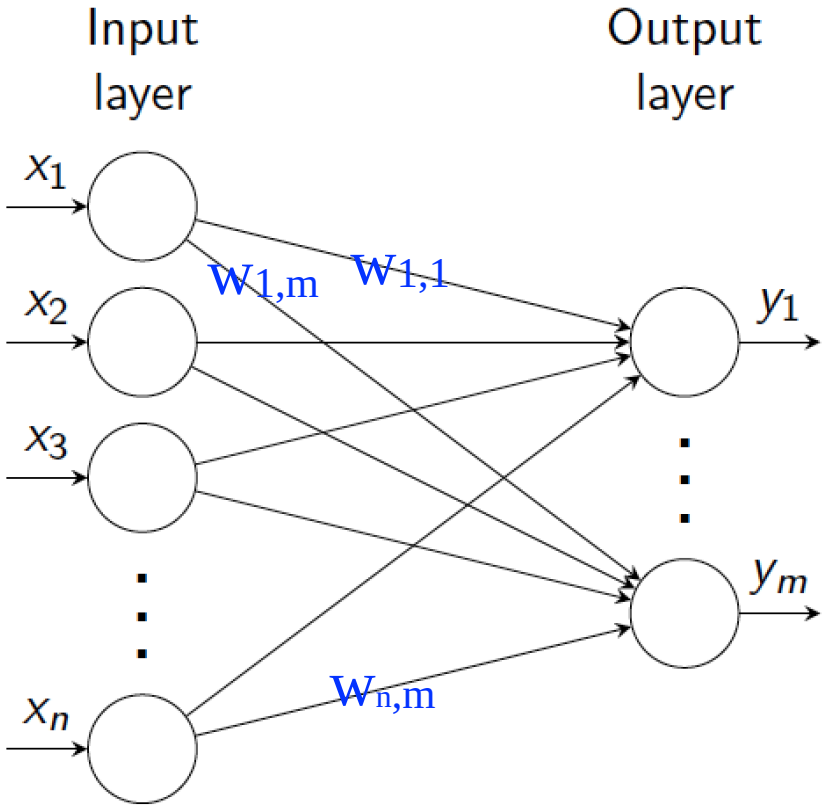
$$f(a) = \tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

Can think of bias as weight w_0, connected to constant input 1: $y = f([w_0, w]^T [1; x])$.

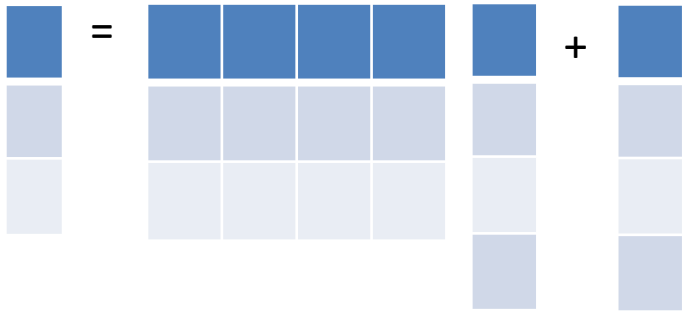
GEORGETOWN UNIVERSITY

Single layer network

- Input: column vector x (size n×1)
- Output: column vector y (size m×1)
- Layer parameters:
  weight matrix W (size n×m)
  bias vector b (m×1)
- Units activation: $a = Wx + b$



ex. 4 inputs, 3 outputs

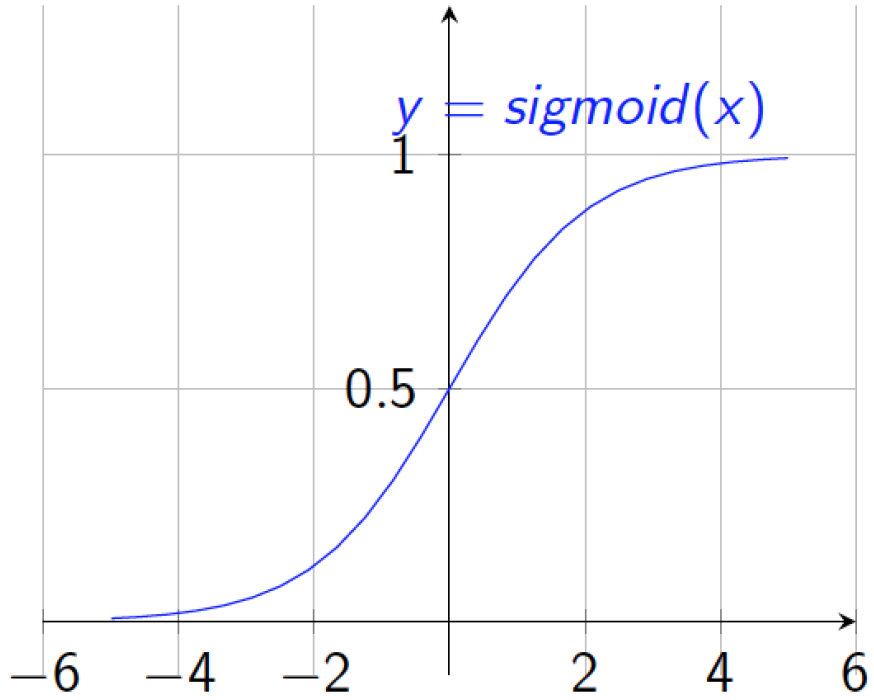


- Output: $y = f(a) = f(Wx + b)$

GEORGETOWN UNIVERSITY

Non-linearities: sigmoid

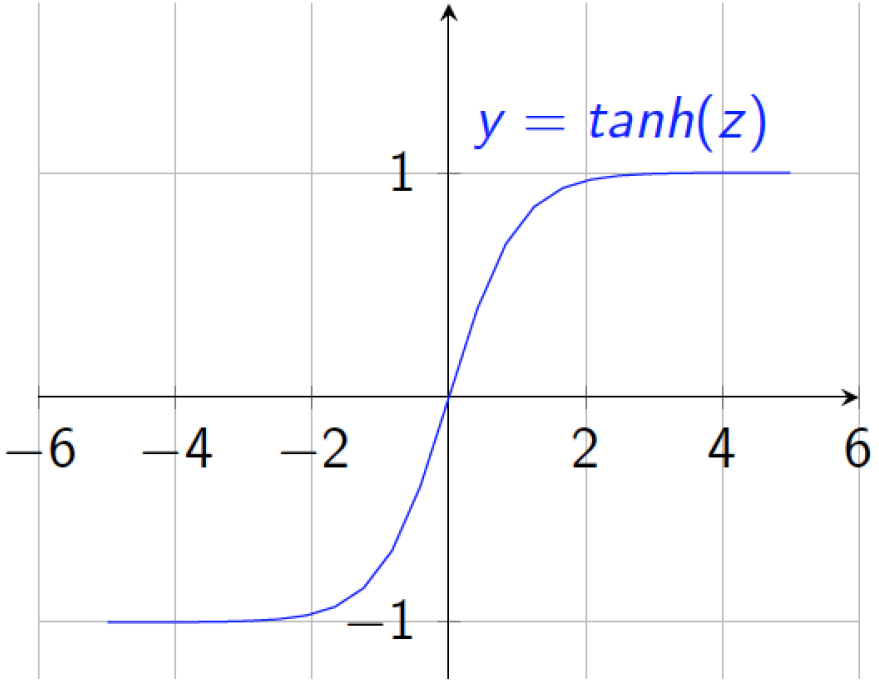
$$f(a) = sigmoid(a) = \frac{1}{1+e^{-a}}$$



- Interpretation as ring rate of neuron
- Bounded between [0,1]
- Saturation for large +ve,-ve inputs
- Gradients go to zero
- Not used in practice

Non-linearities: tanh

$$f(a) = \tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

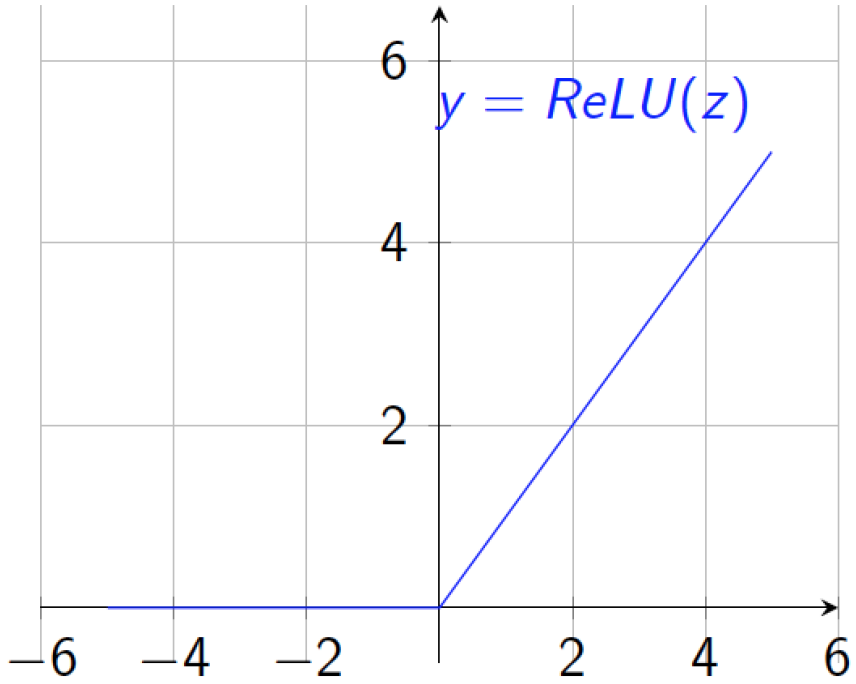


- Bounded between [-1,+1]
- Saturation for large +ve,-ve inputs
- Gradients go to zero
- Outputs centered at 0
- Preferable to sigmoid

tanh(x) = 2 sigmoid(2x) −1

Non-linearities: rectified linear (ReLU)

$$f(a) = \max(a, 0)$$



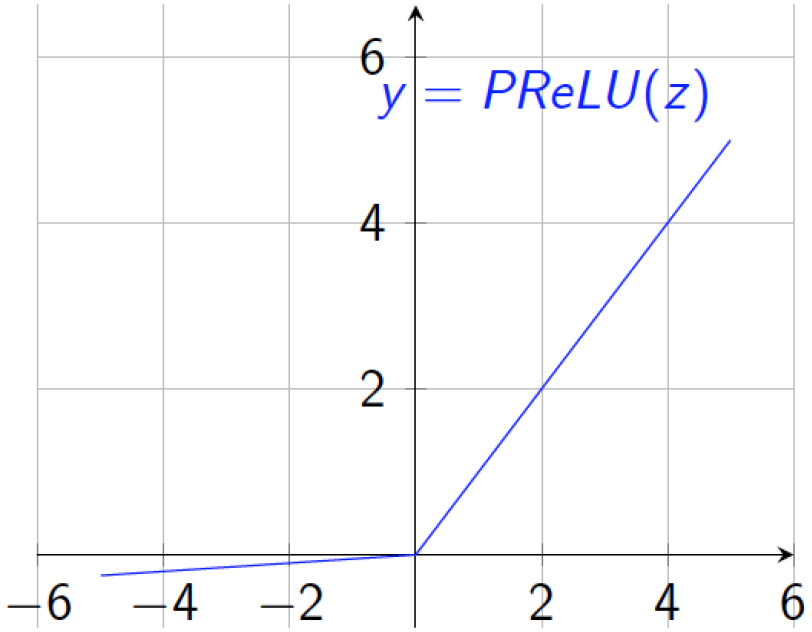
- Unbounded output (on positive side)
- Efficient to implement:

$$f'(a) = \frac{df}{da} = \begin{cases} 0 & a < 0 \\ 1 & a \geq 0 \end{cases}$$

- Also seems to help convergence (see 6x speedup vs tanh in Krizhevsky et al.)
- Drawback: if strongly in negative region, unit is dead forever (no gradient).
- Default choice: widely used in current models.

Non-linearities: Leaky ReLU

$$f(a) = \begin{cases} \max(0, a) & a > 0 \\ \alpha \min(0, a) & a < 0 \end{cases}$$



- where α is small (e.g. 0.02)
- Efficient to implement:

$$f'(a) = \frac{df}{da} = \begin{cases} -\alpha & a < 0 \\ 1 & a > 0 \end{cases}$$

- Also known as probabilistic ReLU (PReLU)
- Has non-zero gradients everywhere (unlike ReLU)
- α can also be learned (see Kaiming He et al. 2015).

Multiple layers

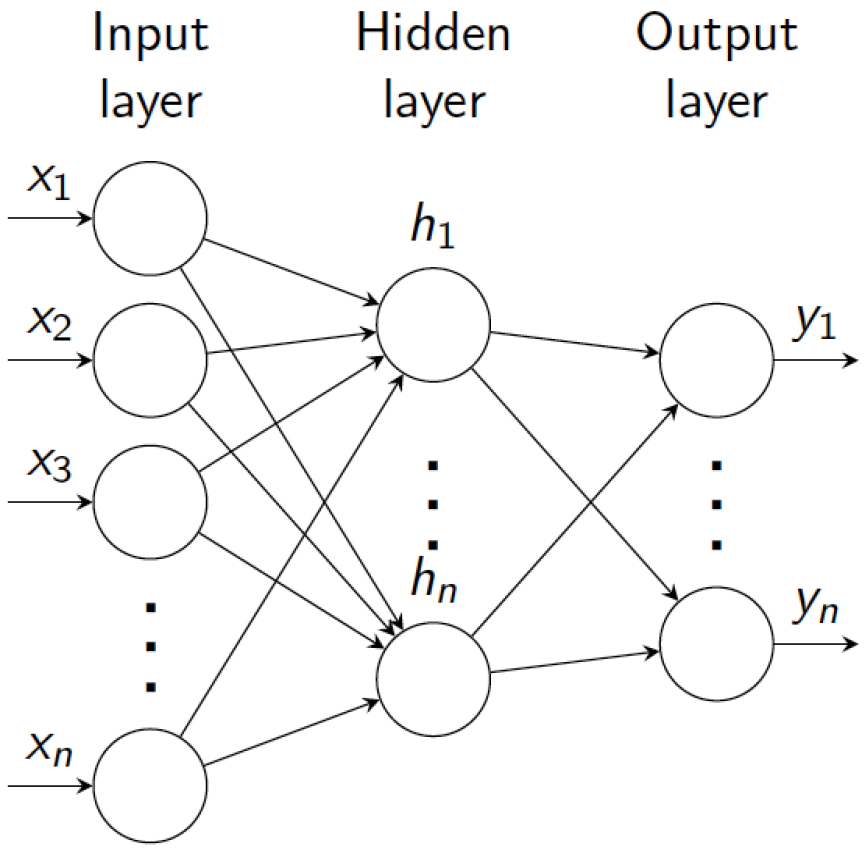
- Neural networks are composed of multiple layers of neurons.

- Feed Forward has acyclic structure. Basic model assumes full connections between layers.

- Layers between input and output are called hidden.

- Various names used:
  - Articial Neural Nets (ANN)
  - Multi-layer Perceptron (MLP)
  - Fully-connected network

- Neurons typically called units.



GEORGETOWN UNIVERSITY

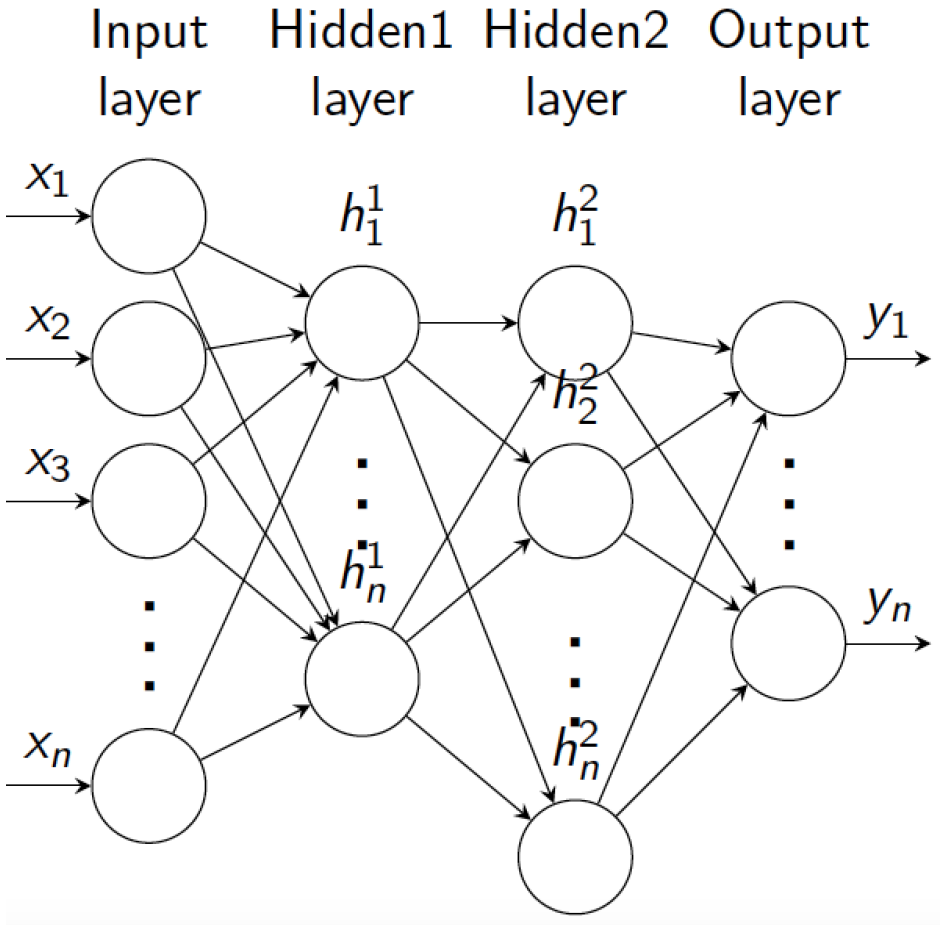
Example: 3 layer MLP

- By convention, number of layers is hidden + output (i.e. does not include input).

- So 3-layer model has 2 hidden layers.

- Parameters:

  weight matrices $W_1;W_2;W_3$ bias vectors b_1; b_2; b_3.



GEORGETOWN UNIVERSITY

Architecture selection

How to pick number of layers and units/layer?

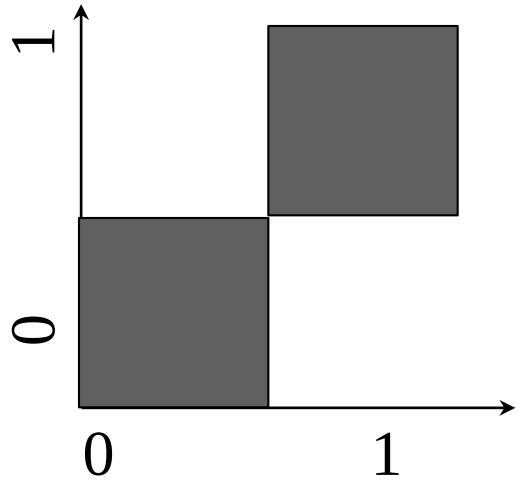
- Active area of research
- For fully connected models 2 or 3 layers seems the most that can be “effectively” trained (more later).
- Regarding number of units/layer:
  - Num parameters grows quickly eg ~ (units/layer)^2 .
  - With large units/layer, can easily overtrain.

Architecture

- 1 layer
  - Lacking in representational power
- Multiple layers
  - Universal approximator
  - Many parameters
  - May lead to
    - Overfitting
    - “erratic” behavior

XOR problem (1 layer)

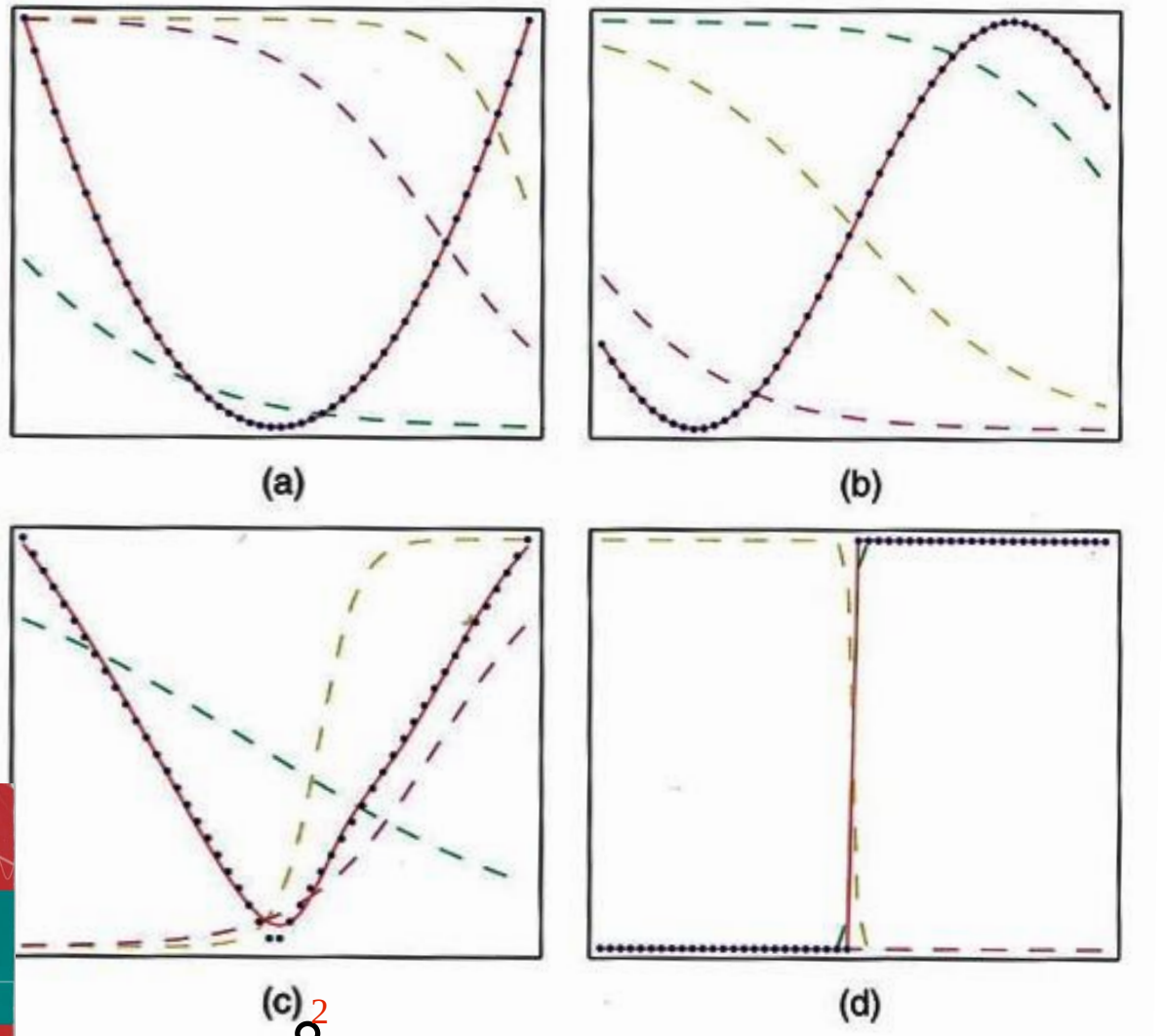
Inputs		Output
0	0	0
1	0	1
0	1	1
1	1	0

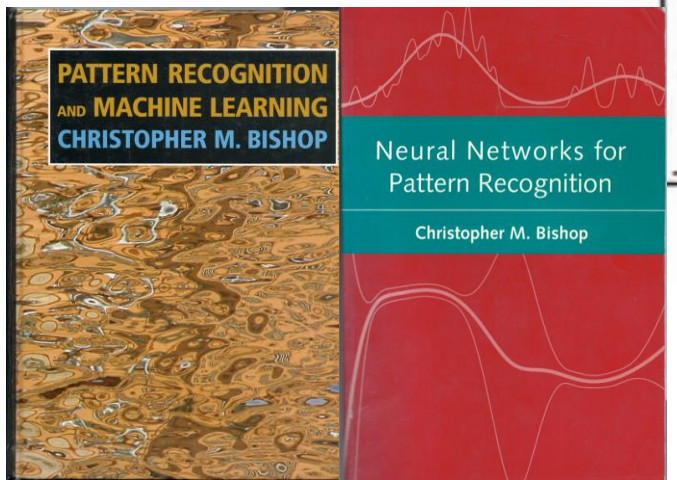


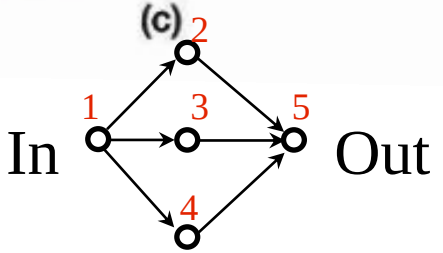
PDP authors pointed to the backpropagation algorithm as a breakthrough, allowing multi-layer neural networks to be trained. Among the functions that a multi-layer network can represent but a single-layer network cannot: the XOR function.

Representational power of two-layer network

Figure 5.3 Illustration of the capability of a multilayer perceptron to approximate four different functions comprising (a) $f(x) = x^2$, (b) $f(x) = \sin(x)$, (c), $f(x) = |x|$, and (d) $f(x) = H(x)$ where $H(x)$ is the Heaviside step function. In each case, $N = 50$ data points, shown as blue dots, have been sampled uniformly in x over the interval $(-1, 1)$ and the corresponding values of $f(x)$ evaluated. These data points are then used to train a two-layer network having 3 hidden units with 'tanh' activation functions and linear output units. The resulting network functions are shown by the red curves, and the outputs of the three hidden units are shown by the three dashed curves.







$$z_5 = \sum_{i=2}^{i=4} w_{i5} \tanh(w_{1i} z_1 + w_{0i})$$

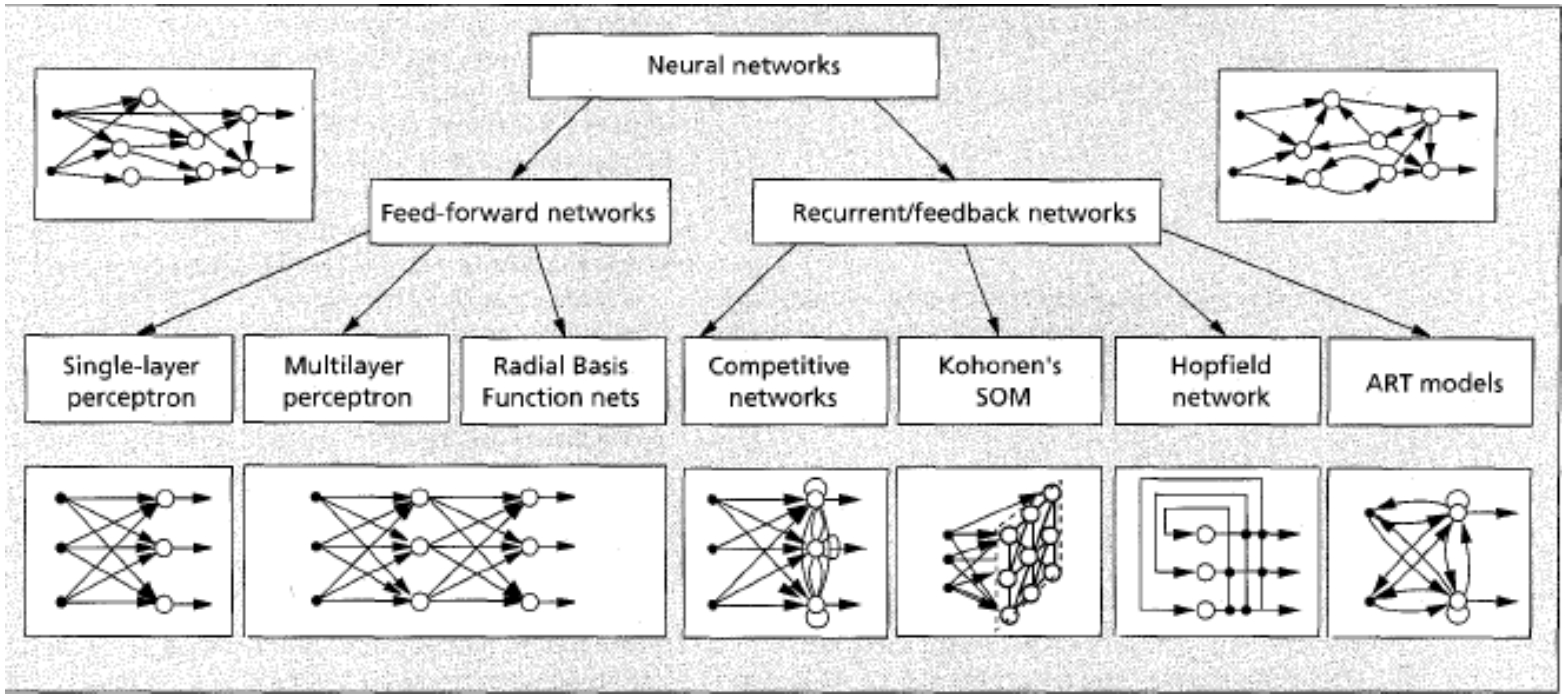
bias

GETOWN
ERSITY

Representational power

- 1 layer? Linear decision surface.
- 2+ layers? In theory, can represent any function. Assuming non-trivial non-linearity.
  - Bengio 2009, http://www.iro.umontreal.ca/~bengioy/papers/ftml.pdf
  - Bengio, Courville, Goodfellow book http://www.deeplearningbook.org/contents/mlp.html
  - Simple proof by M. Neilsen http://neuralnetworksanddeeplearning.com/chap4.html
  - D. Mackay book http://www.inference.phy.cam.ac.uk/mackay/itprnn/ps/482.491.pdf
- Design Decision: very wide two layers vs narrow deep model? In practice, more layers helps.

Neural Networks: Networks of Perceptrons



Optimization: (Back Propagation)

- There is no clear cost function nor objective for internal nodes and corresponding parameters
- There is a clear objective at the classification layer!
- Thus we can compute the gradient at the cost layer (and via composition of functions) use the chain rule to compute gradient at internal nodes.

Computing gradients

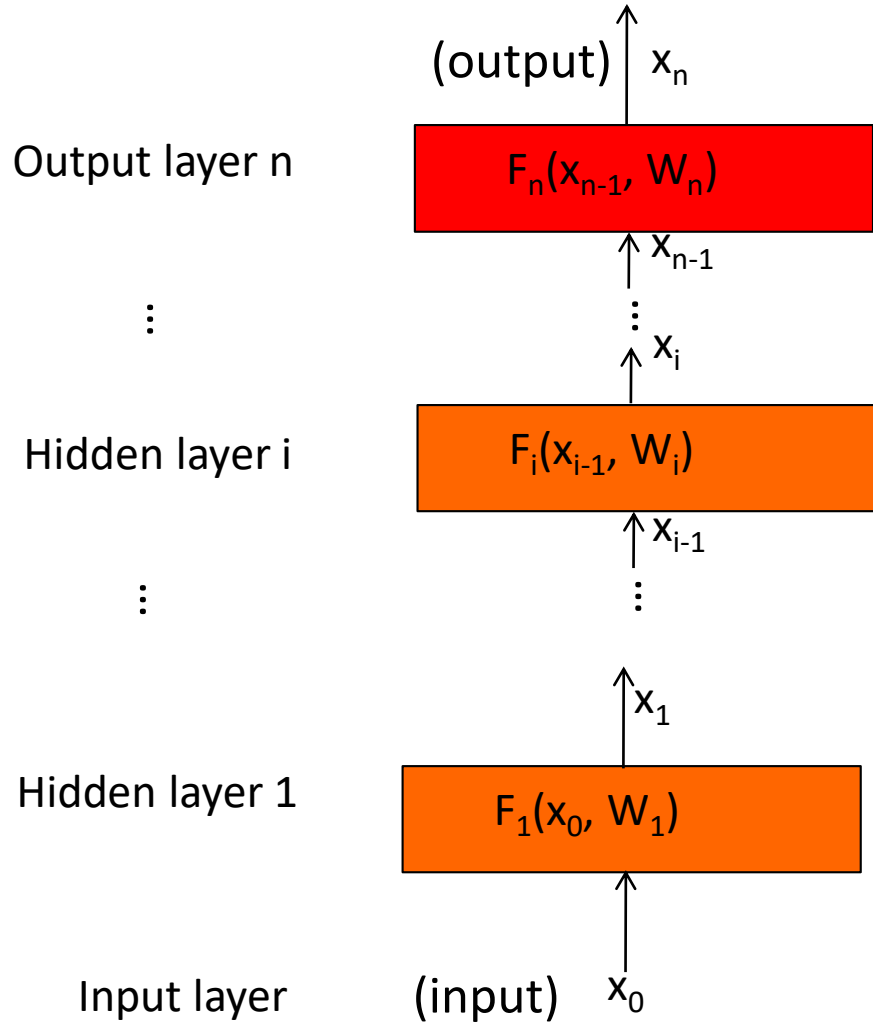
We could write the cost function to get the gradients:

$$C(x_n, y; \theta) = C(F_n(x_{n-1}, w_n), y)$$

with $\theta = [w_1, w_2, \cdots, w_n]$

If we compute the gradient with respect to the parameters of the last layer (output layer) w_n, using the chain rule:

$$\frac{\partial C}{\partial w_n} = \frac{\partial C}{\partial x_n} \cdot \frac{\partial x_n}{\partial w_n} = \frac{\partial C}{\partial x_n} \cdot \frac{\partial F_n(x_{n-1}, w_n)}{\partial w_n}$$



(how much the cost changes when we change wn, is the product between how much the cost changes when we change the output of the last layer, times how much the output changes when we change the layer parameters.)

GEORGETOWN UNIVERSITY

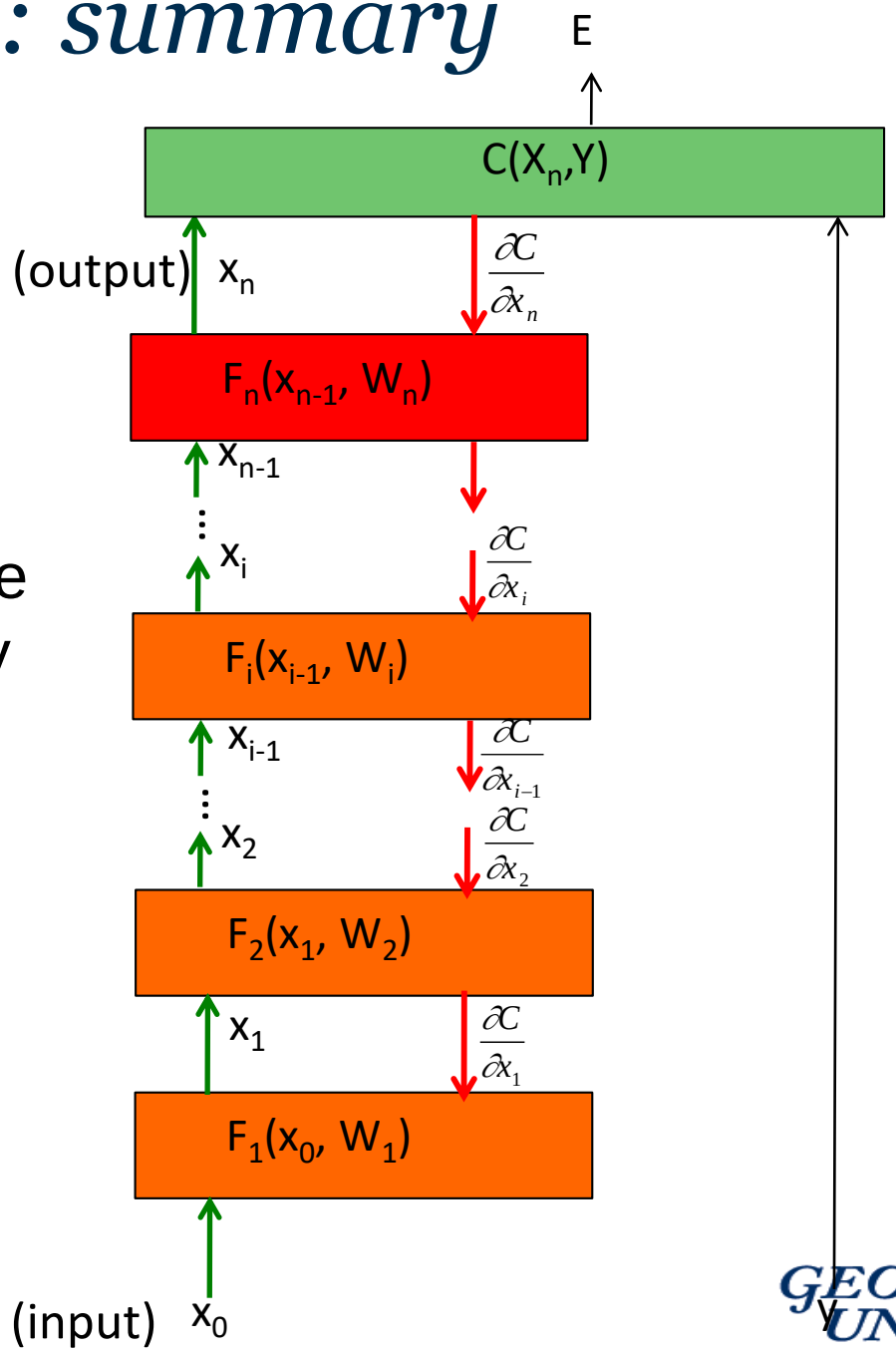
Backpropagation: summary

- Forward pass: for each training example. Compute the outputs for all layers $x_i = F_i(x_{i-1}, w_i)$

- Backwards pass: compute cost derivatives iteratively from top to bottom:

$$\frac{\partial C}{\partial x_{i-1}} = \frac{\partial C}{\partial x_i} \cdot \frac{\partial F_i(x_{i-1}, w_i)}{\partial x_{i-1}}$$

- Compute gradients and update weights.



ANNs in Computer Vision

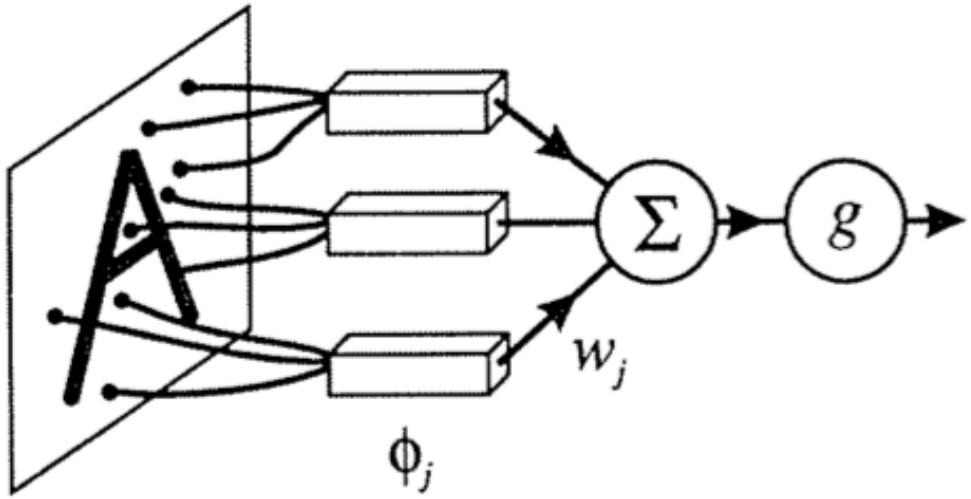
- There is a massive amount of literature
  - ANNs for OCR
  - ANNs for object detection
  - ANN for feature generation

- Convolutional Neural Networks
  - A common variant used in computer vision
  - Uses scale space / convolution pyramids

Perceptrons, 1958

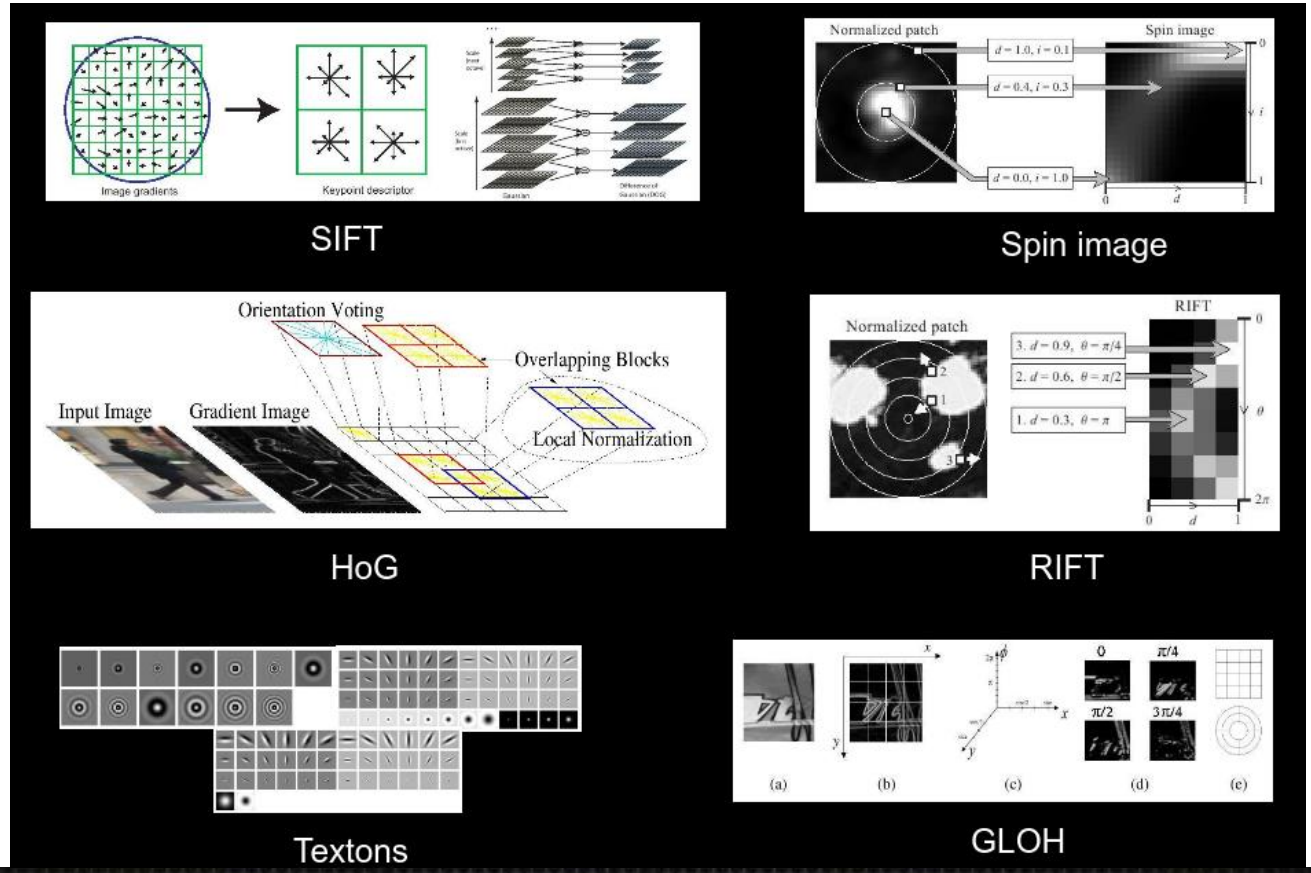
- In computer vision, intuitively, we would like to use spatial information to identify patterns

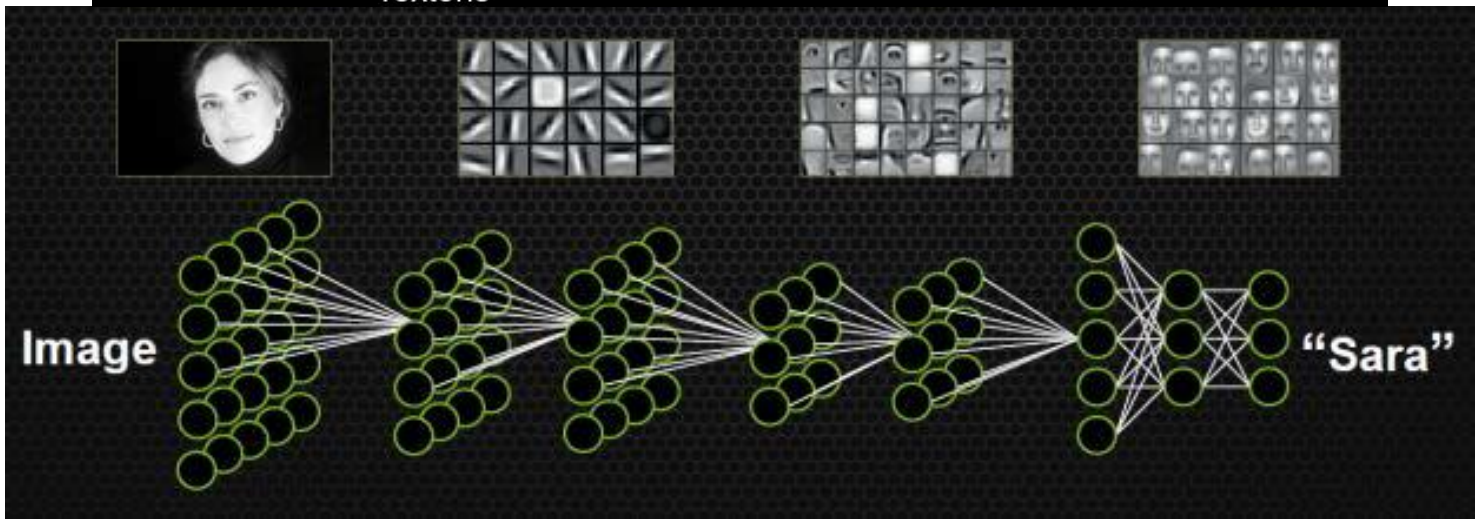
- We can use convolutions and matched filters within the ANN framework to achieve our goal



Application to Computer Vision

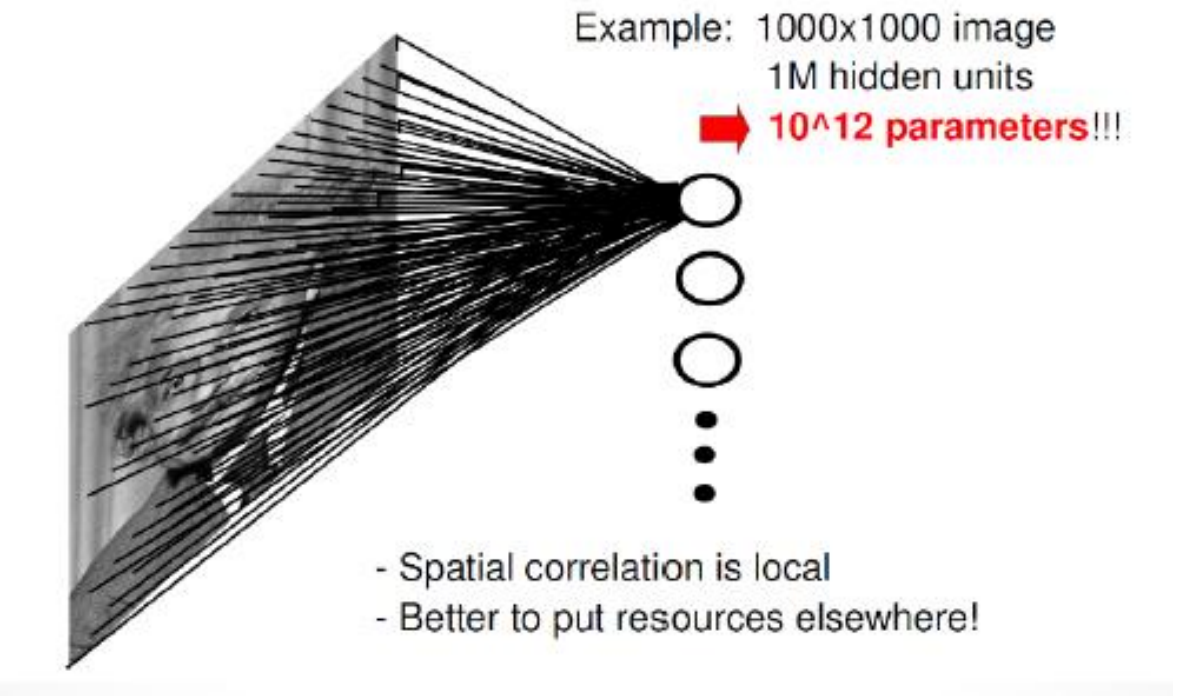
- Use standard filters to identify and learn distinct patterns in an image.
- Some models learn filters as well.



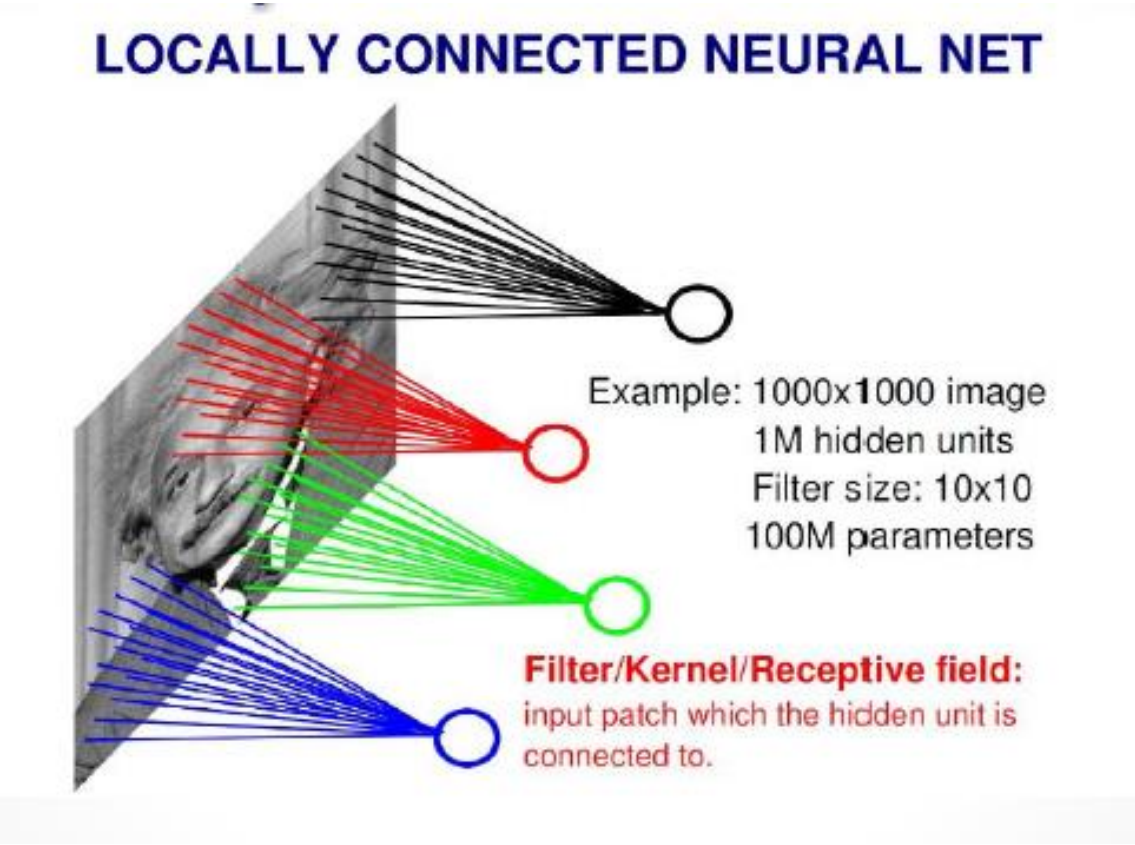


Convolution at input and hidden layers

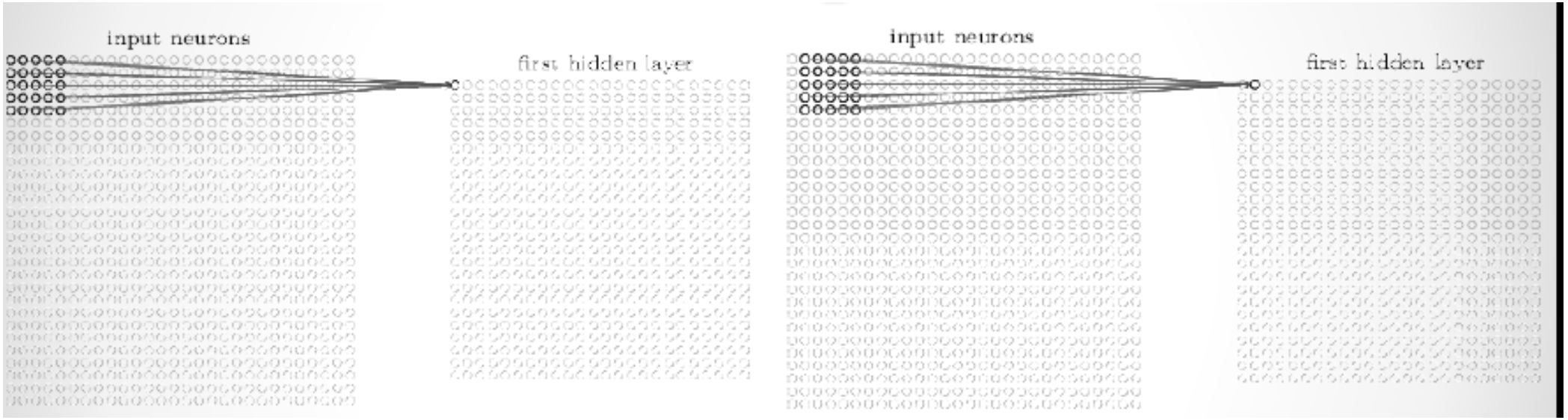
- Why?
  - Local analysis is intuitive in CV
  - Vectorizing an image will lead to many unnecessary parameters
  - Instead: Parameter Sharing and locality!



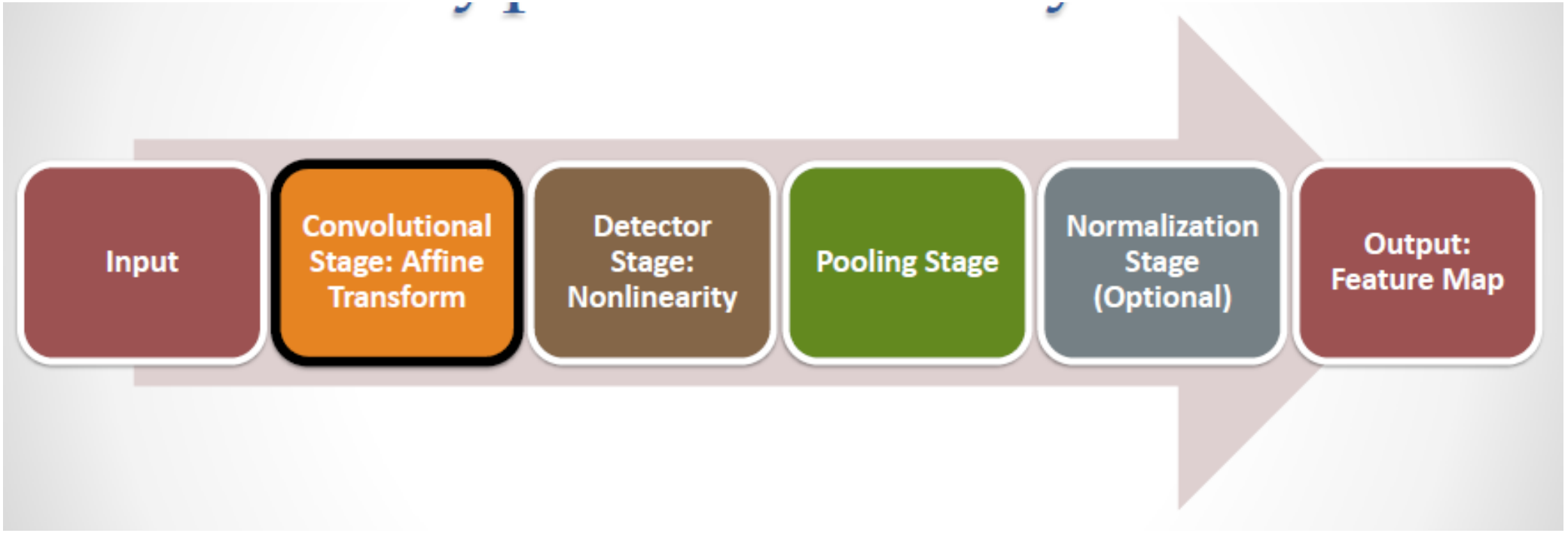
Basis of CNN


LOCALLY CONNECTED NEURAL NET
Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters
Filter/Kernel/Receptive field:
input patch which the hidden unit is
connected to.

Use convolution!

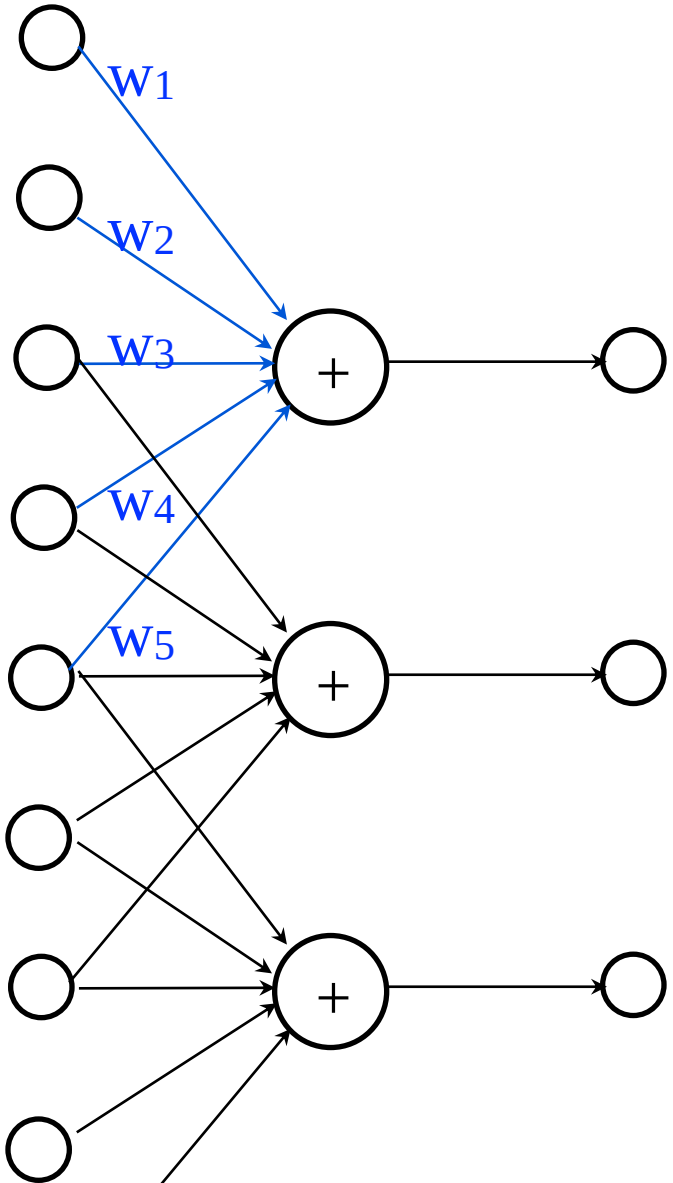


CNN: Basic Structure and Idea

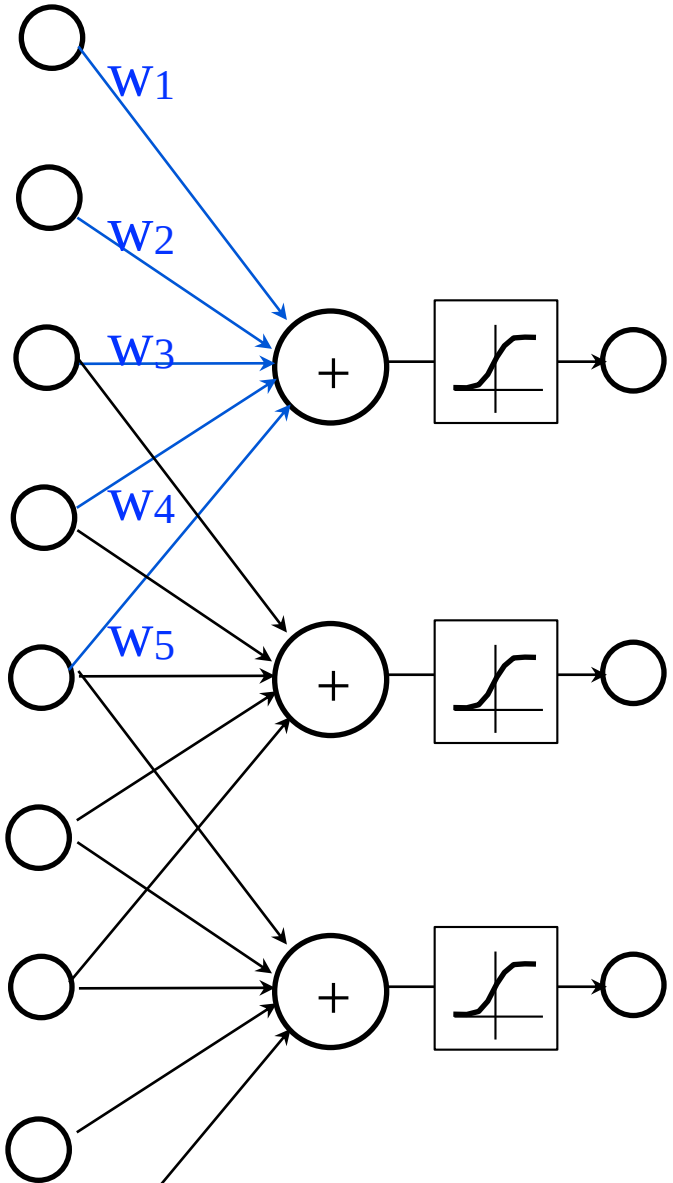


- "Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers."

Linear filtering pyramid architecture



Convolutional neural network architecture

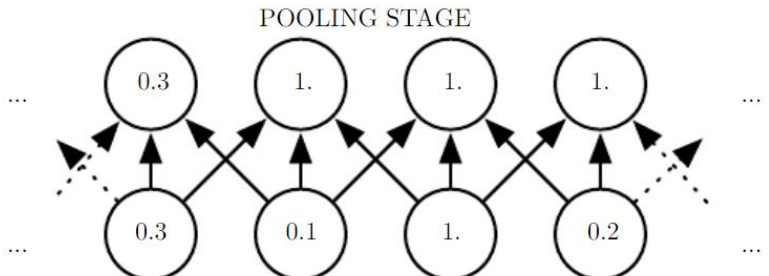


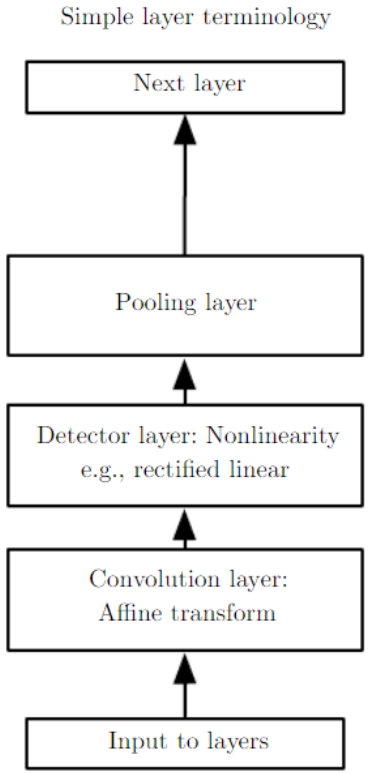
What kernels to use?

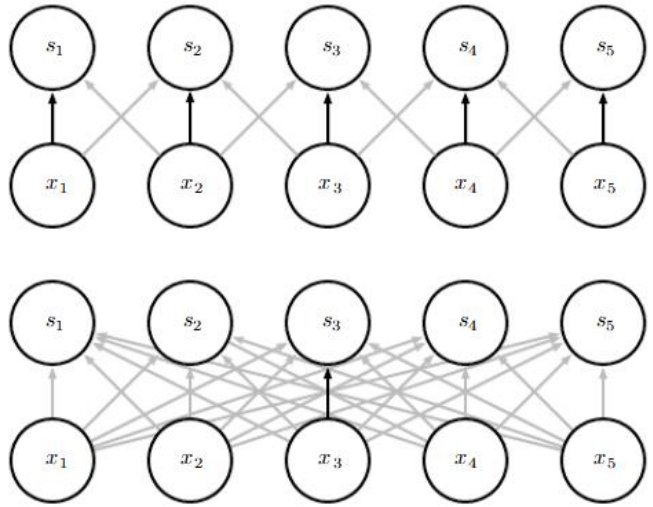
- Some General Approaches
  - Randomly select kernels and hope for the best!
  - Hand-craft kernels based on *apriori* knowledge of data and experience
  - Learn the “optimal” kernels
    - This approach is used more and more today
    - Increases the number of parameters dramatically
    - Increases training time

Convolutional Neural Network Details

- Kernels are often parameters
  - parameter sharing
- Parameter sharing leads to reduced parameters
  - (Believe it -- It could be worse)
- Pooling
  - Summarize, aggregate, reduce result from previous layer
  - EG. Max-Pooling Zhou and Chellappa
  - Pooling helps to make the representation invariant to small translations







GEORGETOWN UNIVERSITY

Gabor Features often used (and learned!)

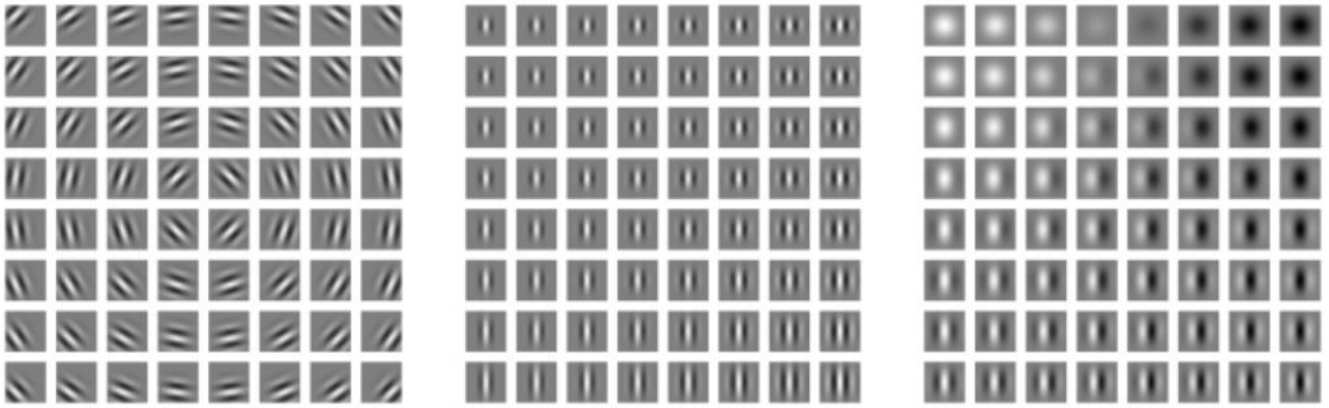


Figure 9.18: Gabor functions with a variety of parameter settings. White indicates

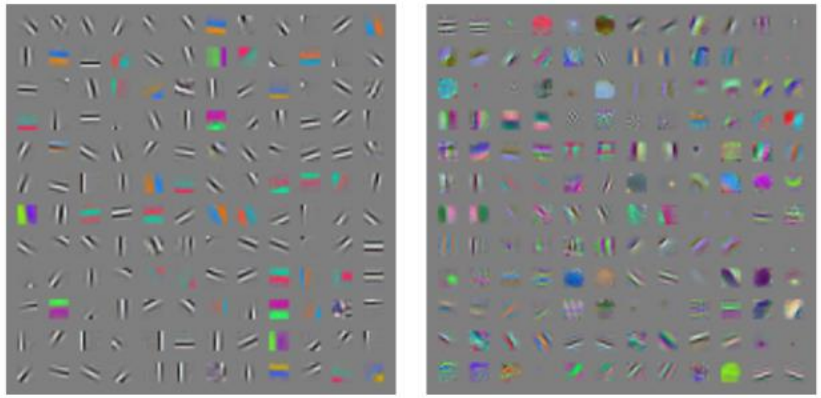


Figure 9.19: Many machine learning algorithms learn features that detect edges or specific colors of edges when applied to natural images. These feature detectors are reminiscent of the Gabor functions known to be present in primary visual cortex. *(Left)*Weights learned by an unsupervised learning algorithm (spike and slab sparse coding) applied to small

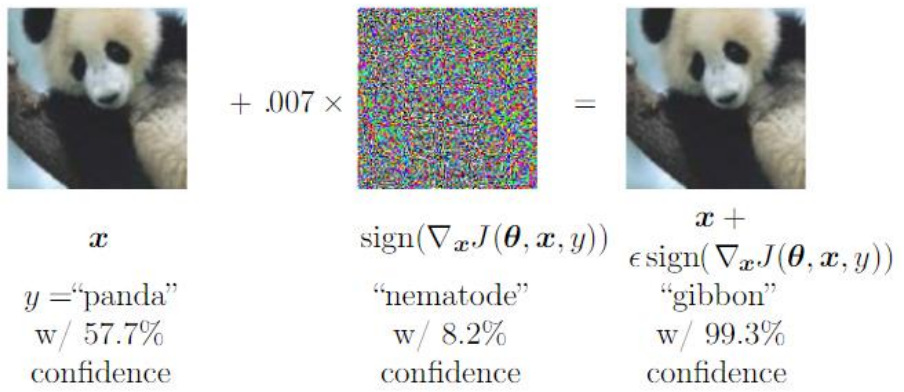
GEORGETOWN UNIVERSITY

Regularization

- Regularization in Computer Vision
  - Add constraints to solve an ill-posed problem
  - Impose a penalty for over-fitting

- Overfitting: Output should change “continuously” as input changes “continuously”
  - Smoothness

- We have seen examples of regularization already
  - Smoothness terms in energy minimization
    - Disparity Map (Stereo)
    - Label Maps (Image Segmentation) Min-Cut (Graphs)
    - Active Contours (Variational Methods)



Common Regularization Strategies

- Deep ANNs have many parameters; subject to overfitting
  - Simple Example k-means with many parameters

- Regularization
  - Pooling
    - Max pooling
  - Data Augmentation
    - Add Noise
  - Parameter Sharing
  - Sparsity Promotion (wrt parameters)
    - L1 and L2 norms; adds term to update equation.
    - Bayesian Solution: Sparsity promoting priors
  - Dropout

Norm-based Regularization

- Increase generalization ability by reducing chance of overfitting.
- Add a penalty for non-zero parameter
  - Penalty often related to norm: L1 or L2
  - Increased cost results in driving parameter values to zero thus reduces number of parameters
- Scheme often used for feature selection
- Similarly, some Bayesian approaches utilize sparsity promoting priors which have similar results.

Data Augmentation

- The best way to train a classifier to generalize better is to have more data!
- Data is often limited in practice
- Create “fake” data
  - Alter existing training samples: rotation, translation, warping… etc
  - Add noise

Parameter Sharing

- Restrict the number of parameters explicitly using intuition
  - Most patterns in CV are conceptually, translation invariant.
  - Use similar filter during matched filter computations.

- Force sharing of parameters at multiple stages of computation for multiple purposes
  - Used often in CNNs (kernels are shared parameters)

Historic Note: NIPS 2000

- NIPS, Neural Information Processing Systems, is the premier conference on machine learning. Evolved from an interdisciplinary conference to a machine learning conference.
- For the NIPS 2000 conference:
  - title words predictive of paper acceptance: "Belief Propagation" and "Gaussian".
  - title words predictive of paper rejection: "Neural" and "Network".

LeCun conv nets, 1998

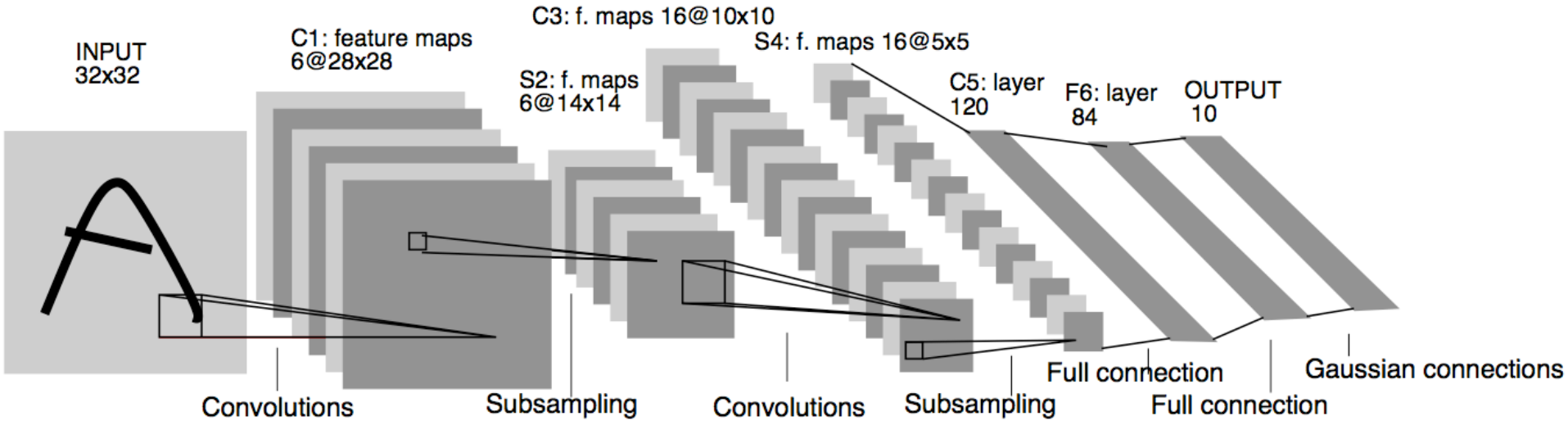
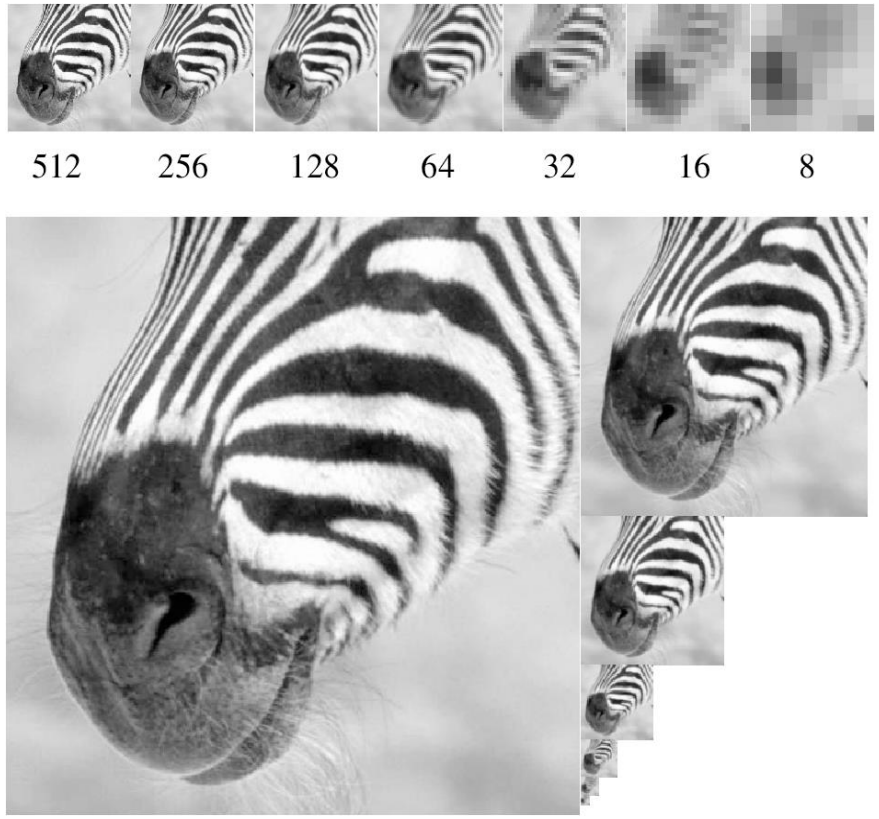


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Demos:
http://yann.lecun.com/exdb/lenet/index.html

Scale Space

- Observe.
  - Many CNN structures inherently perform scale space analysis
    - One layer: Linear filter applied to input image
    - Next layer: subsampling
    - Next layer: Linear filter …



LeCun Results

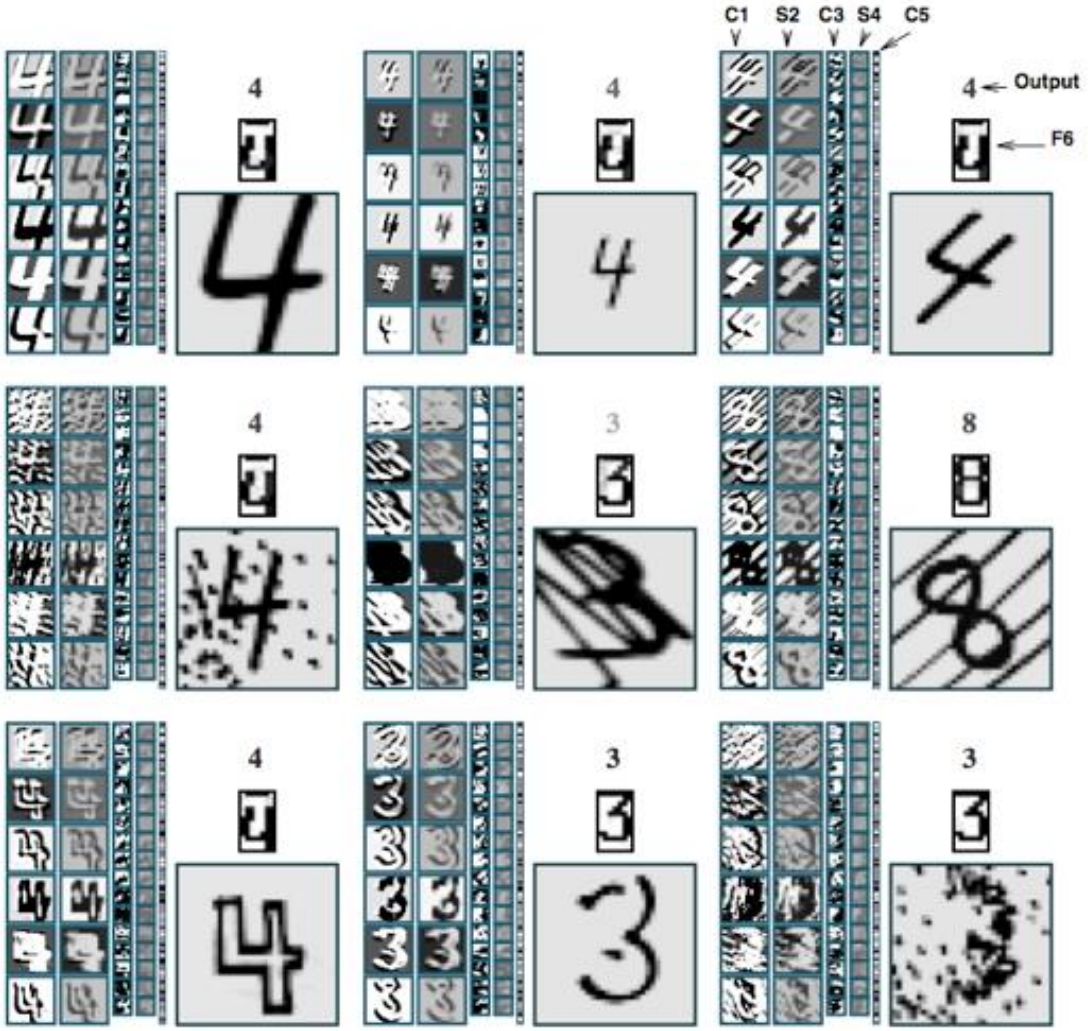


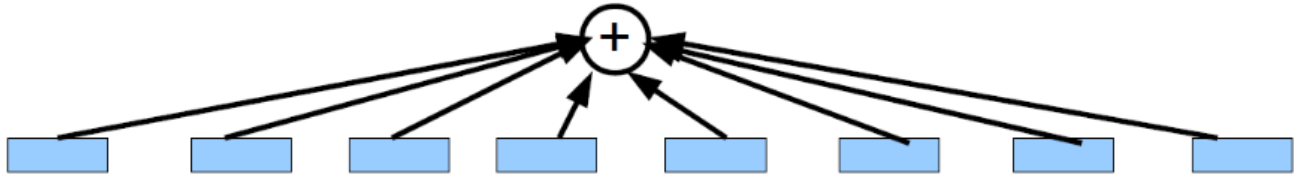
Fig. 13. Examples of unusual, distorted, and noisy characters correctly recognized by LeNet-5. The grey-level of the output label represents the penalty (lighter for higher penalties).

Structural Designs

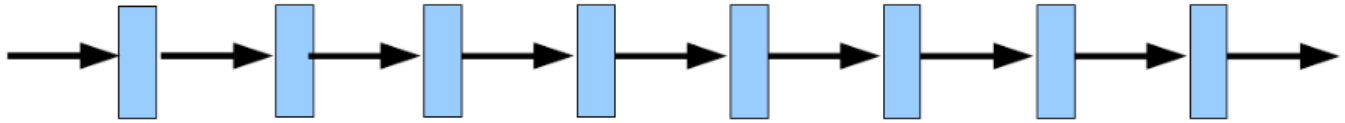
- Shallow may require more templates
- Deep more powerful classifier but subject to overtraining

Given a dictionary of simple non-linear functions: $g_1, \ldots, g_n$

- Proposal 1: linear combination $f(x) \approx \sum_j g_j$



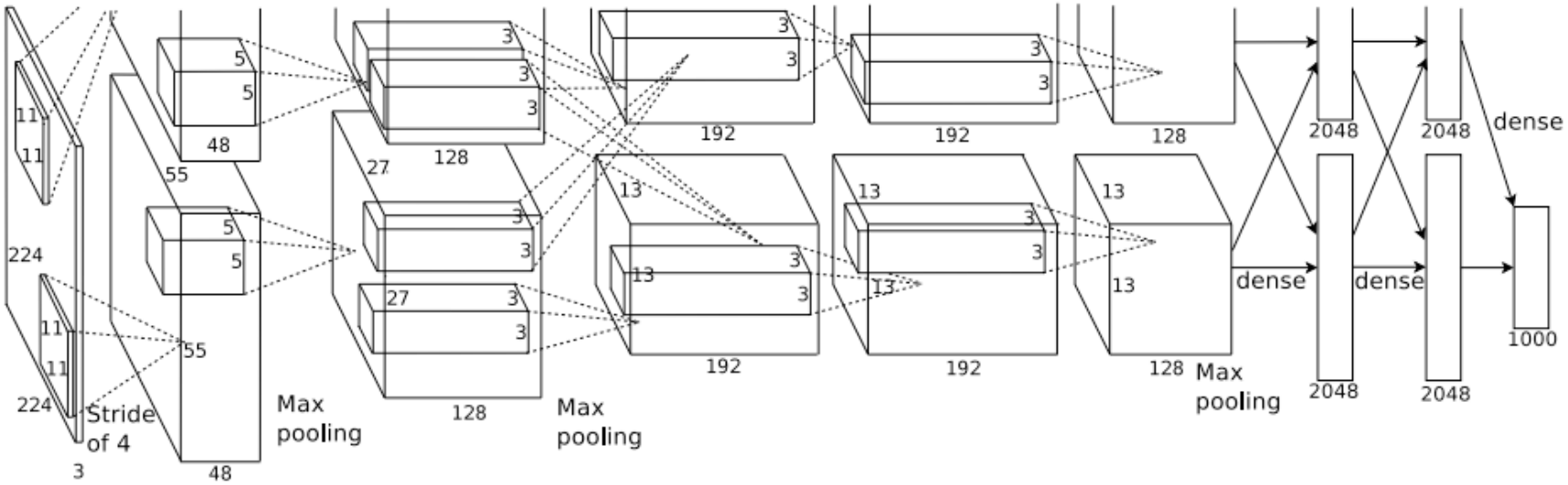
- Proposal 2: composition $f(x) \approx g_1(g_2(\ldots g_n(x)\ldots))$



Krizhevsky et al (NIPS 2012)

- Structure
  - 5 convolutional layers
  - 3 fully connected layers
  - Activation: ReLU
  - (See paper)

- Reduce Overtraining by
  - Data Augmentation
    1. Translate inputs and horizontally mirror input
    2. Randomly perturb RGB values
  - Dropout
    - Randomly zero-out neurons during feedforward and backprop.
      - (Noted significantly less overtraining with Drop Out.)

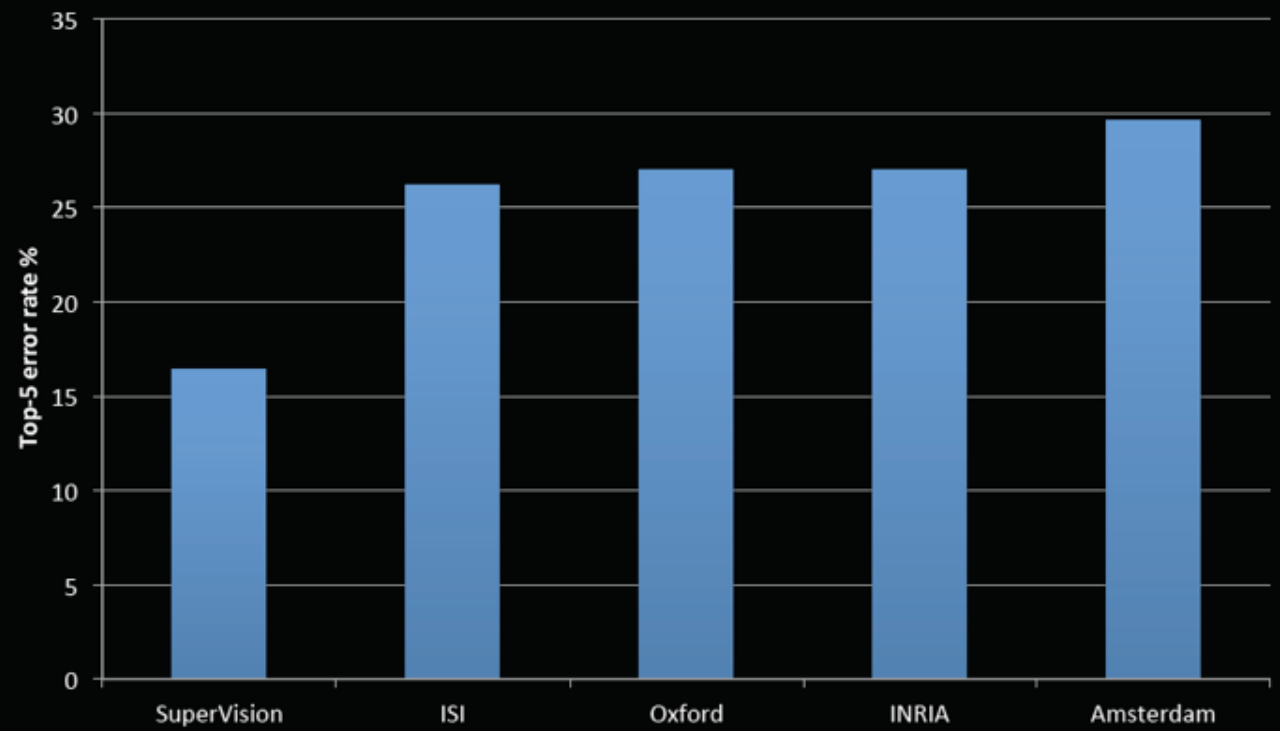


GEORGETOWN UNIVERSITY

Slide from Rob Fergus, NYU

ImageNet Classification 2012

- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



Krizhevsky, Sutskever, and Hinton, NIPS 2012

GEORGETOWN UNIVERSITY

Test Nearby images, according to NN features

Krizhevsky, Sutskever, and Hinton, NIPS 2012

In Summary

- ANNs have a rich history in machine learning

- CNNs are an intuitive variant for Computer Vision tasks
  - Utilize convolutions of kernels
  - Learning / Classification is robust given comprehensive kernels
  - Understanding learned parameters: more “accessible” than generic ANNs
  - Regularization and structure are key factors in deep networks

Projects

- Experimental Design with CNNs
  - What features are used?
  - What structure is used?
    - Shallow (with more units per layer) or deep (with less units per layer)
  - What regularization scheme is used?

COSC579: Appendix

Jeremy Bolton, PhD
Assistant Teaching Professor

GEORGETOWN UNIVERSITY