ECE 5984: Introduction to Machine Learning

Topics:

- Neural Networks
  - Backprop

Readings: Murphy 16.5

Dhruv Batra

Virginia Tech

Administrativia

- HW3
  - Due: in 2 weeks
  - You will implement primal & dual SVMs
  - Kaggle competition: Higgs Boson Signal vs Background classification
  - https://inclass.kaggle.com/c/2015-Spring-vt-ece-machine-learning-hw3
  - https://www.kaggle.com/c/higgs-boson

Administrativia

- Project Mid-Sem Spotlight Presentations
  - Friday: ~~5-7pm~~, 3-5pm Whittemore 654

  - 5 slides (recommended)
  - 4 minute time (STRICT) + 1-2 min Q&A

  - Tell the class what you're working on
  - Any results yet?
  - Problems faced?
  - Upload slides on Scholar

Recap of Last Time

Not linearly separable data

- Some datasets are **not linearly separable!**
  - http://www.eee.metu.edu.tr/~alatan/Courses/Demo/AppletSVM.html

Addressing non-linearly separable data – Option 1, non-linear features

- Choose non-linear features, e.g.,
  - Typical linear features: $w_0 + \sum_i w_i x_i$
  - Example of non-linear features:
    - Degree 2 polynomials, $w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$

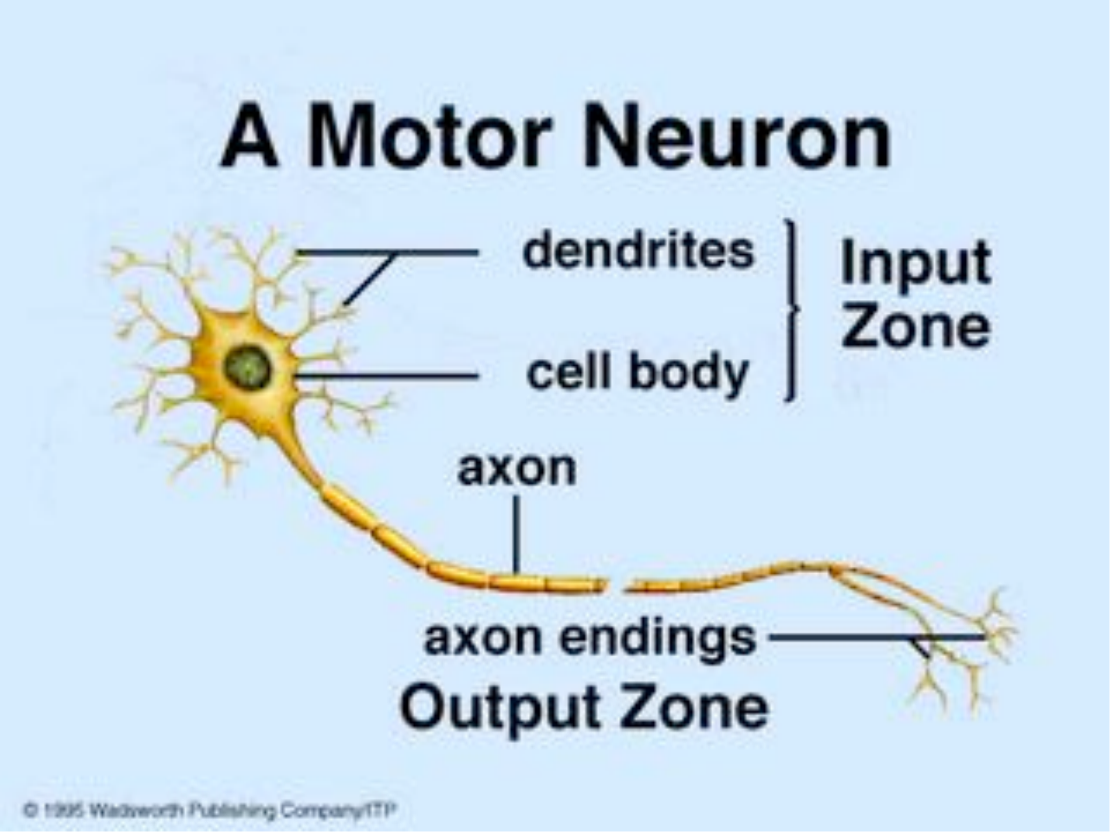
- Classifier $h_{\mathbf{w}}(\mathbf{x})$ still linear in parameters **w**
  - As easy to learn
  - Data is linearly separable in higher dimensional spaces
  - Express via kernels

Addressing non-linearly separable data – Option 2, non-linear classifier

- Choose a classifier $h_{\mathbf{w}}(\mathbf{x})$ that is non-linear in parameters $\mathbf{w}$, e.g.,
  - Decision trees, neural networks,…

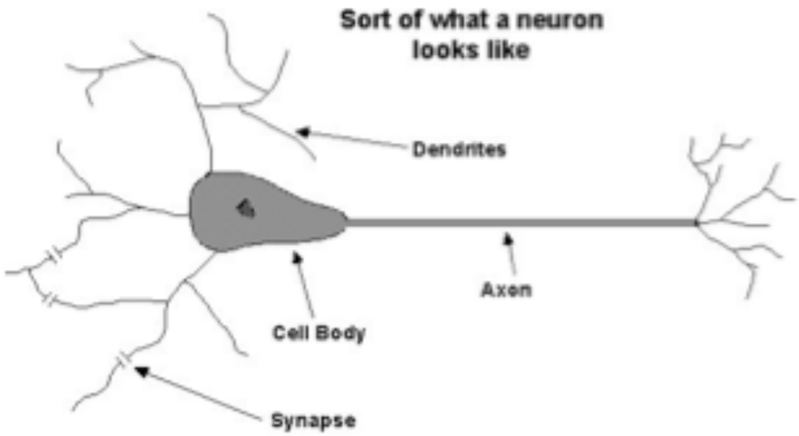
- More general than linear classifiers
- But, can often be harder to learn (non-convex optimization required)
- Often very useful (outperforms linear classifiers)
- In a way, both ideas are related

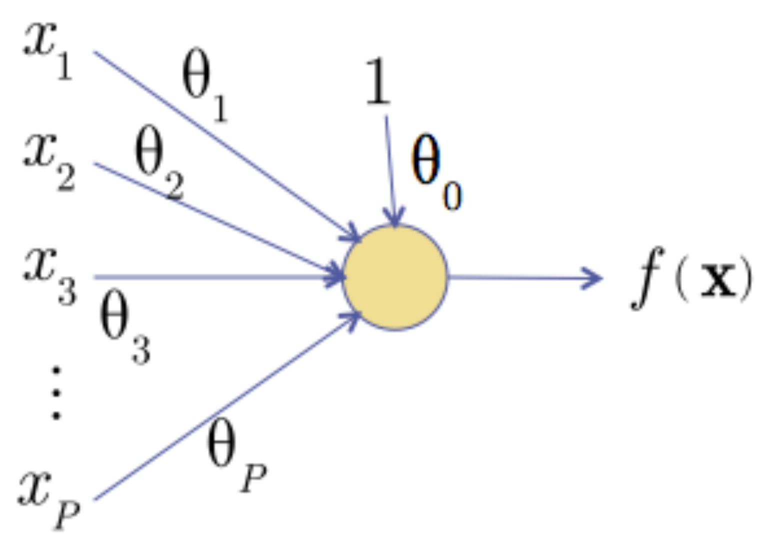
Biological Neuron



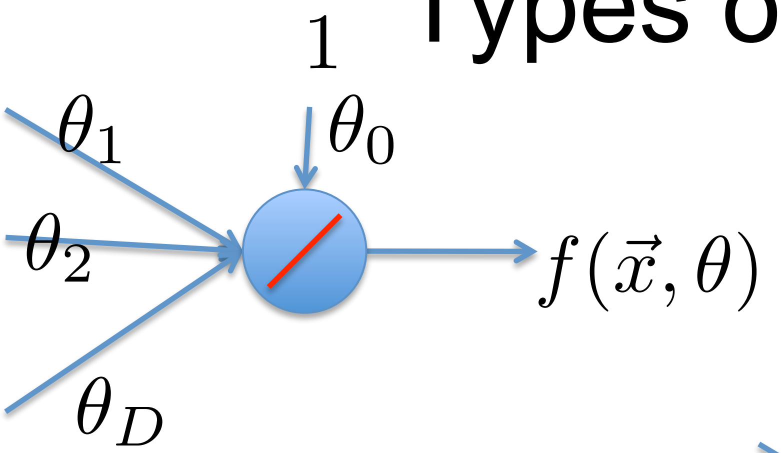
Recall: The Neuron Metaphor

- Neurons
  - accept information from multiple inputs,
  - transmit information to other neurons.
- Multiply inputs by weights along edges
- Apply some function to the set of inputs at each node

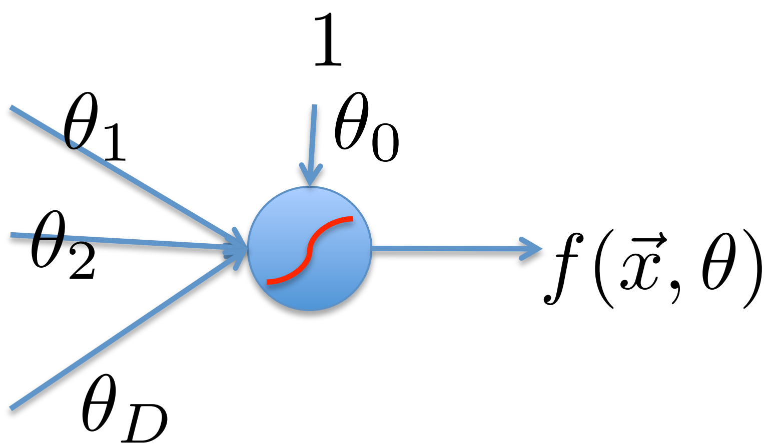




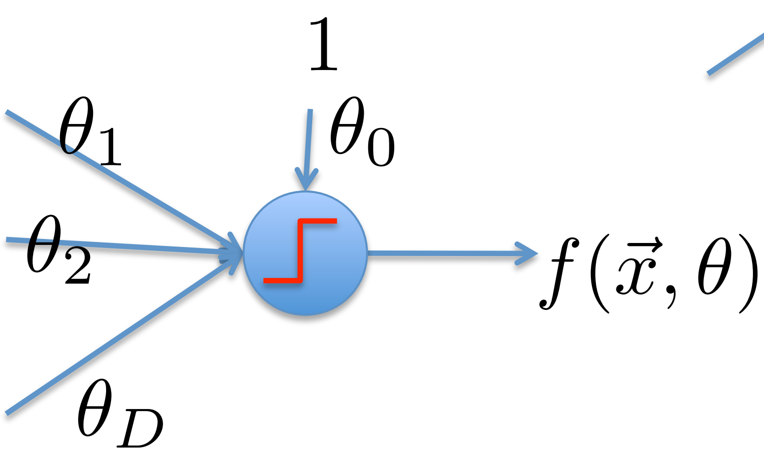
Types of Neurons



Linear Neuron



Logistic Neuron



Perceptron

Potentially more. Require a convex loss function for gradient descent training.

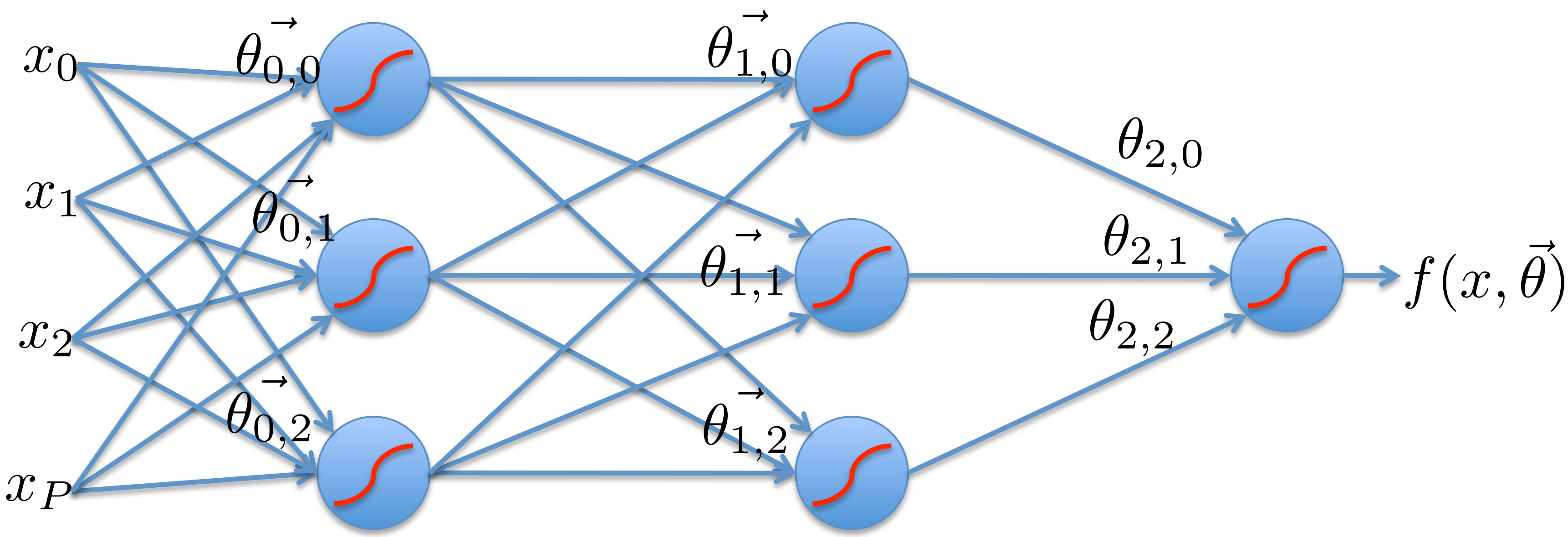
Slide Credit: HKUST

Limitation

- A single “neuron” is still a linear decision boundary
- What to do?
- Idea: Stack a bunch of them together!

Multilayer Networks

- Cascade Neurons together
- The output from one layer is the input to the next
- Each Layer has its own sets of weights



Universal Function Approximators

- Theorem
  - 3-layer network with linear outputs can uniformly approximate any continuous function to arbitrary accuracy, given enough hidden units [Funahashi ’89]

Plan for Today

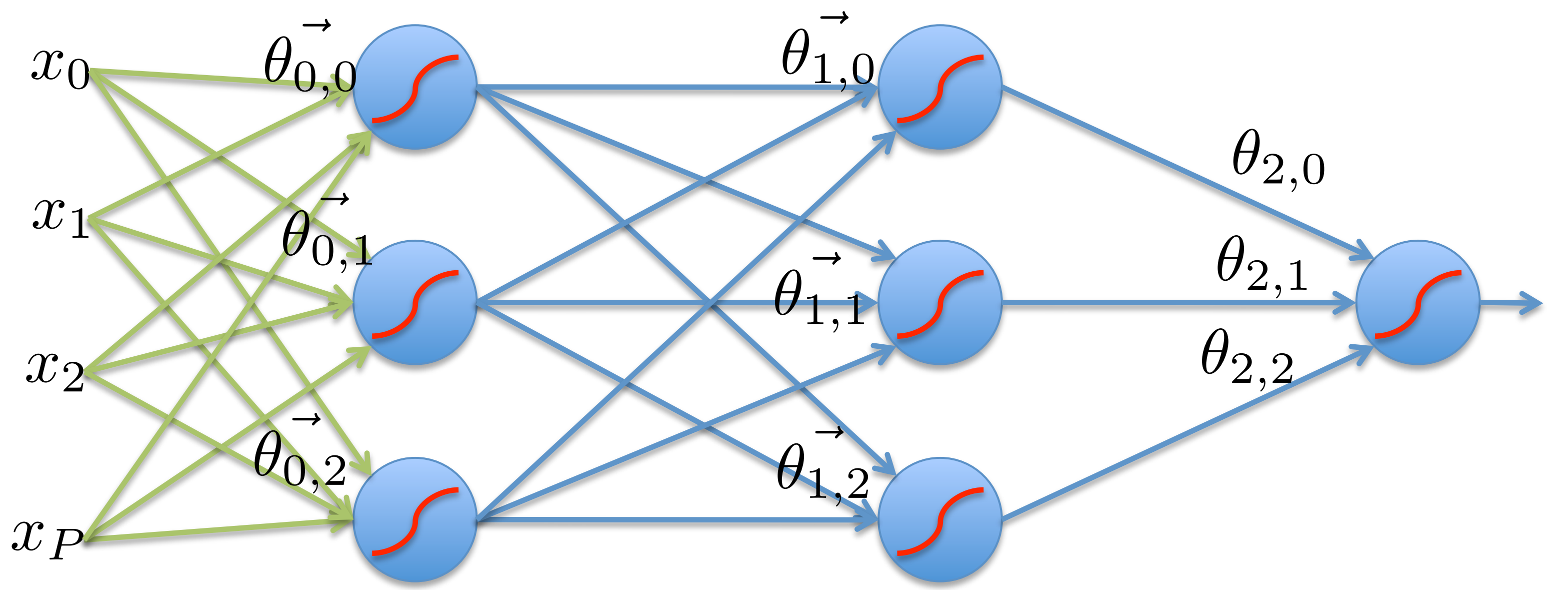
- Neural Networks
  - Parameter learning
  - Backpropagation

Forward Propagation

- On board

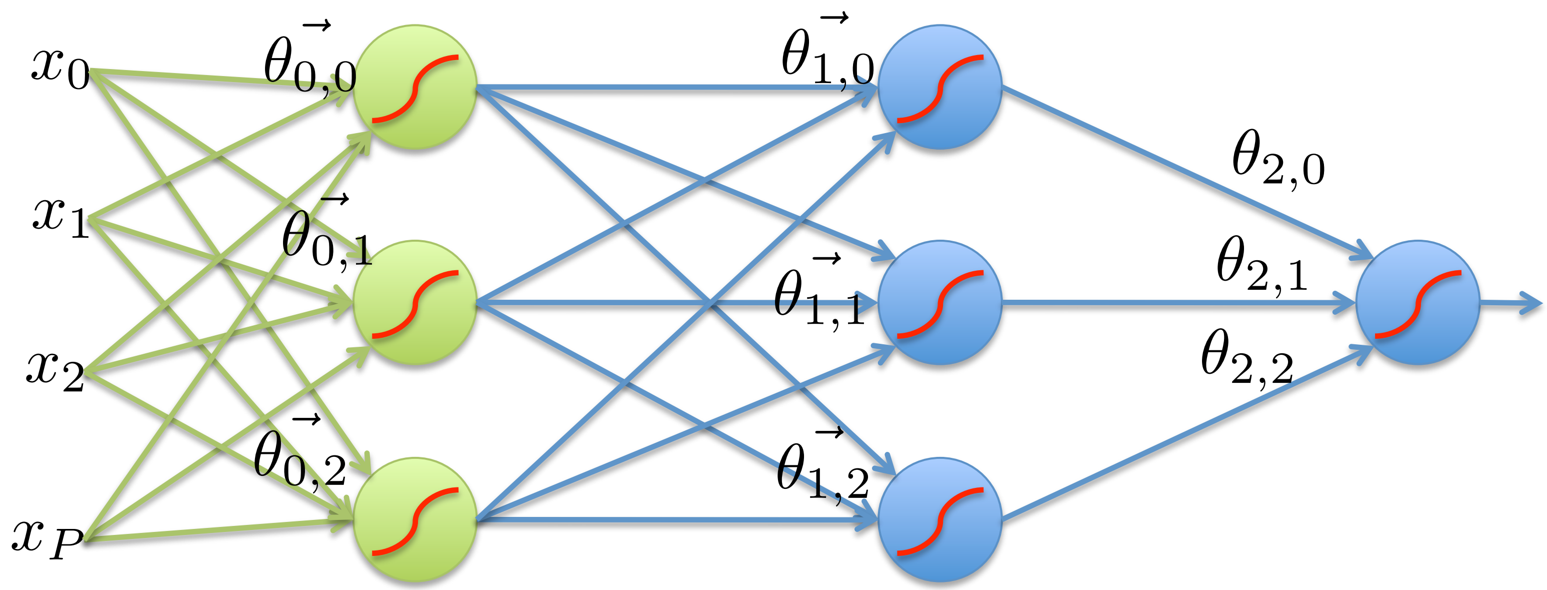
Feed-Forward Networks

- Predictions are fed forward through the network to classify



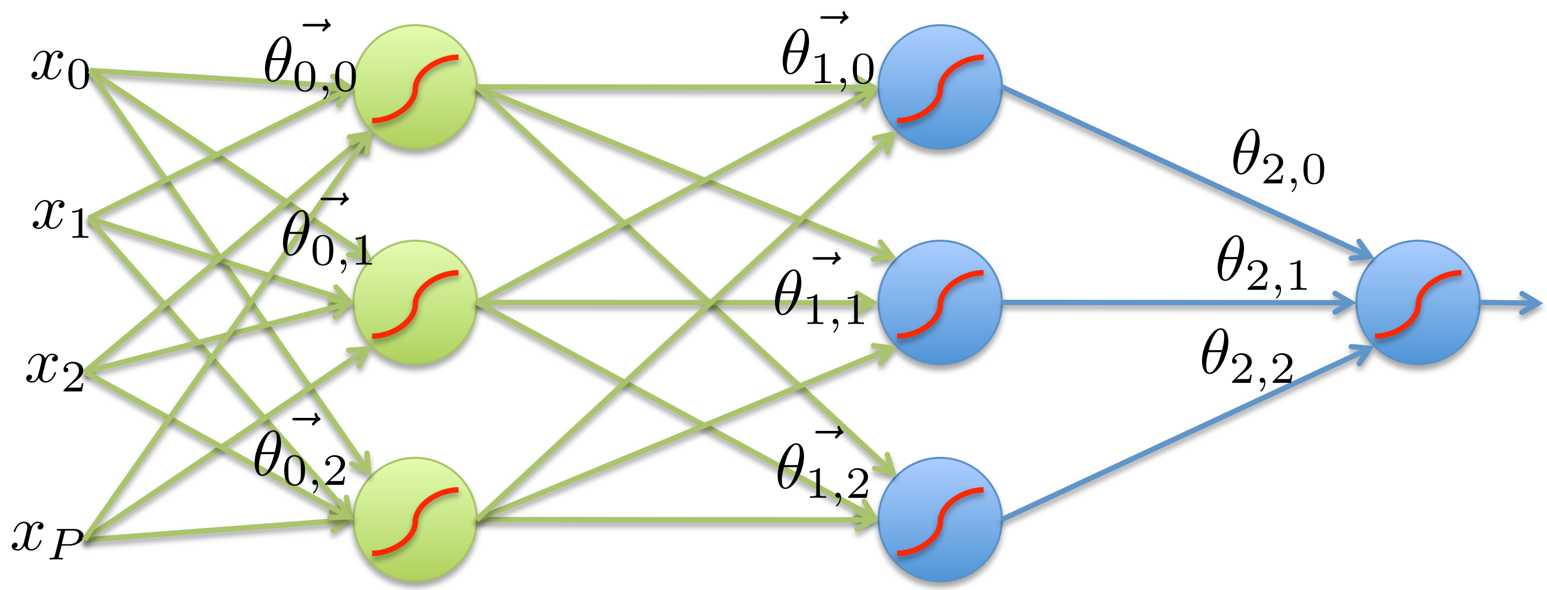
Feed-Forward Networks

- Predictions are fed forward through the network to classify



Feed-Forward Networks

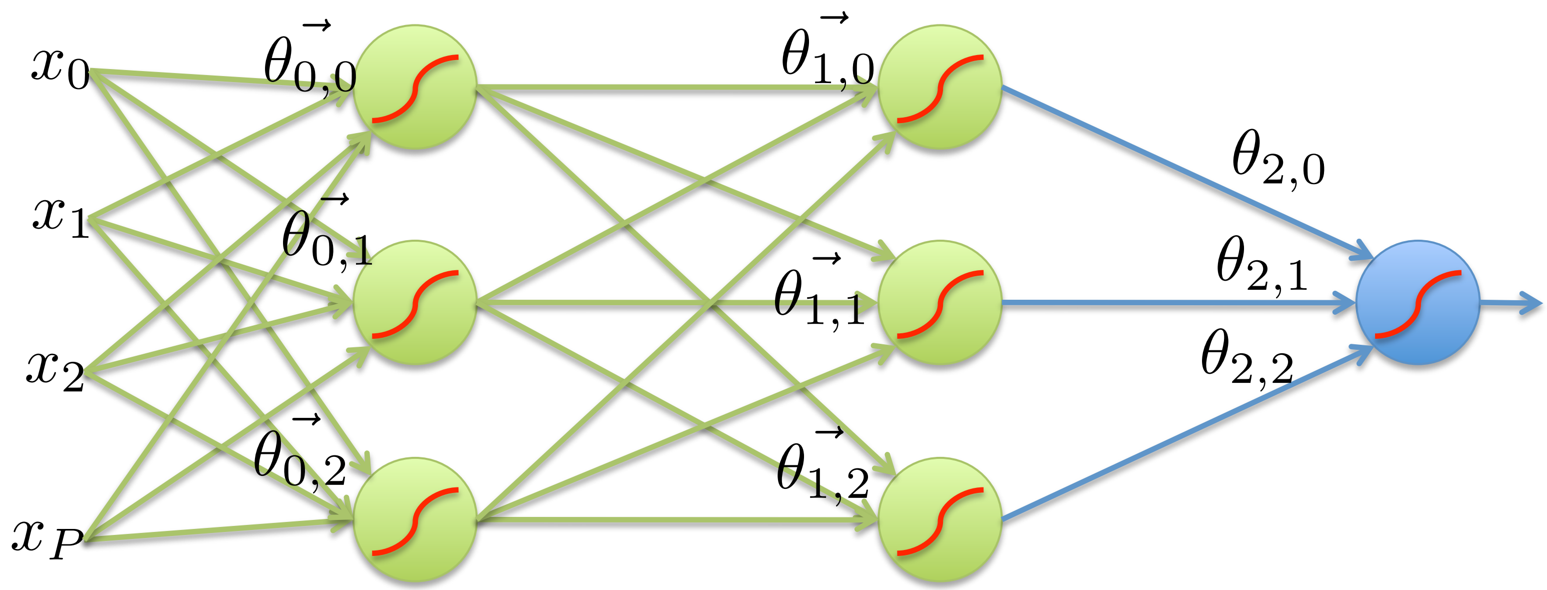
- Predictions are fed forward through the network to classify



Slide Credit: HKUST

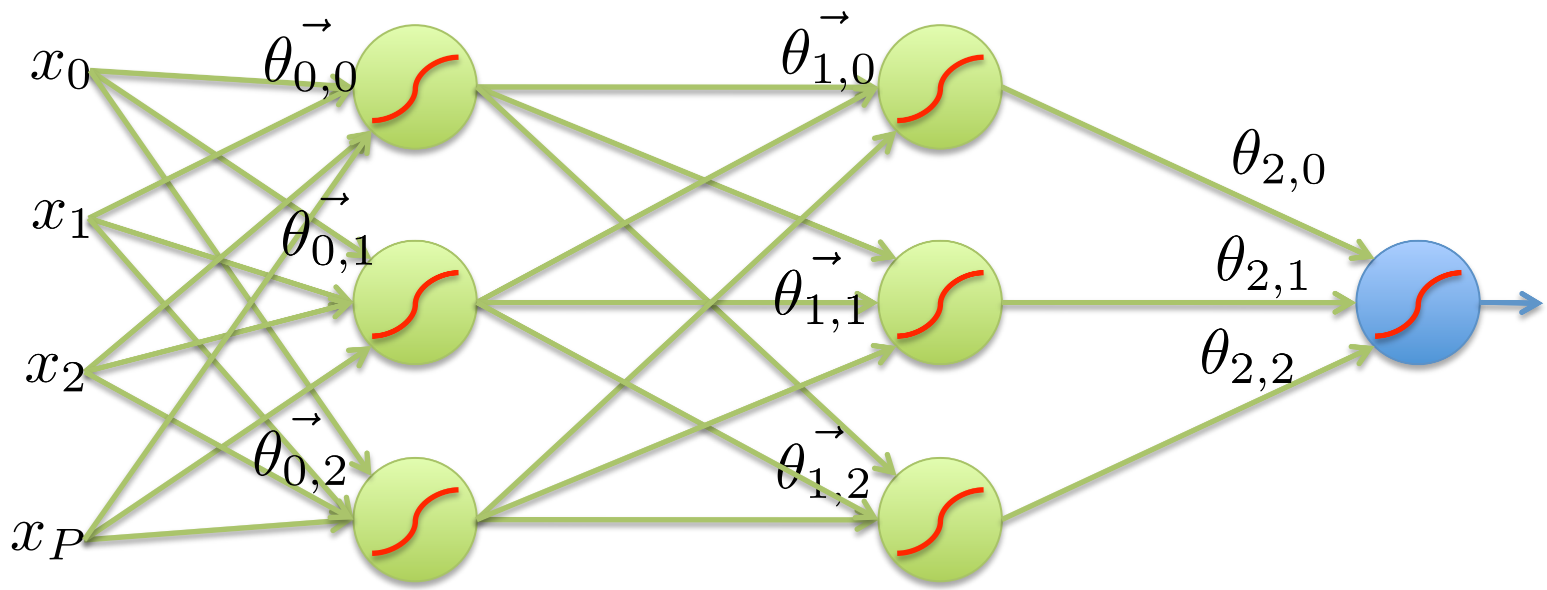
Feed-Forward Networks

- Predictions are fed forward through the network to classify



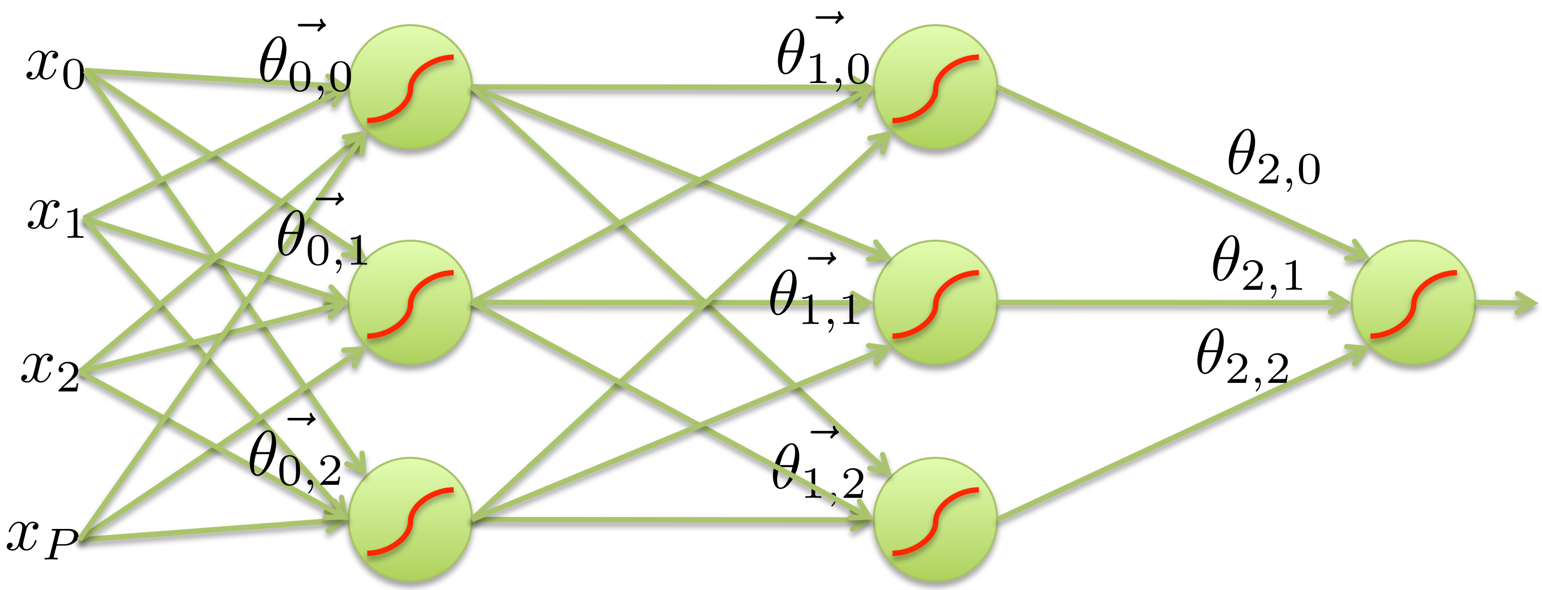
Feed-Forward Networks

- Predictions are fed forward through the network to classify



Feed-Forward Networks

- Predictions are fed forward through the network to classify



Gradient Computation

- First let’s try:
  – Single Neuron for Linear Regression
  – Single Neuron for Logistic Regresion

Logistic regression

- Learning rule – MLE:

$$\begin{aligned}\frac{\partial \ell(W)}{\partial w_i} &= \sum_j x_i^j [y^j - P(Y^j = 1 \mid x^j, W)] \\ &= \sum_j x_i^j [y^j - g(w_0 + \sum_i w_i x_i^j)]\end{aligned}$$

$$\boxed{w_i \leftarrow w_i + \eta \sum_j x_i^j \delta^j}$$

$$\delta^j = y^j - g(w_0 + \sum_i w_i x_i^j)$$

Gradient Computation

- First let's try:
  - Single Neuron for Linear Regression
  - Single Neuron for Logistic Regresion

- Now let's try the general case

- Backpropagation!
  - Really efficient

Neural Nets

- Best performers on OCR
  - http://yann.lecun.com/exdb/lenet/index.html

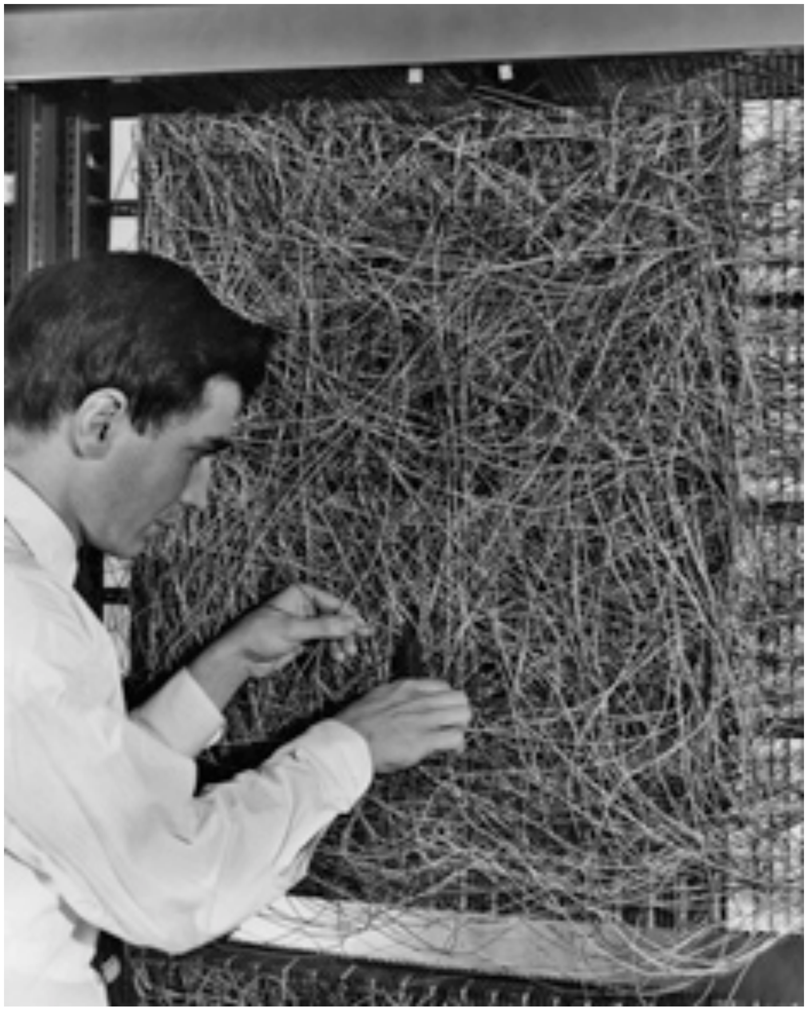
- NetTalk
  - Text to Speech system from 1987
  - http://youtu.be/tXMaFhO6dIY?t=45m15s

- Rick Rashid speaks Mandarin
  - http://youtu.be/Nu-nlQqFCKg?t=7m30s

Neural Networks

- Demo
  - http://neuron.eng.wayne.edu/bpFunctionApprox/bpFunctionApprox.html

Historical Perspective



Convergence of backprop

- Perceptron leads to convex optimization
  - Gradient descent reaches **global minima**

- Multilayer neural nets **not convex**
  - Gradient descent gets stuck in local minima
  - Hard to set learning rate
  - Selecting number of hidden units and layers = fuzzy process
  - NNs had fallen out of fashion in 90s, early 2000s
  - Back with a new name and significantly improved performance!!!!
    - Deep networks
      - Dropout and trained on much larger corpus

Overfitting

- Many many many parameters

- Avoiding overfitting?
  - More training data
  - Regularization
  - Early stopping

A quick note

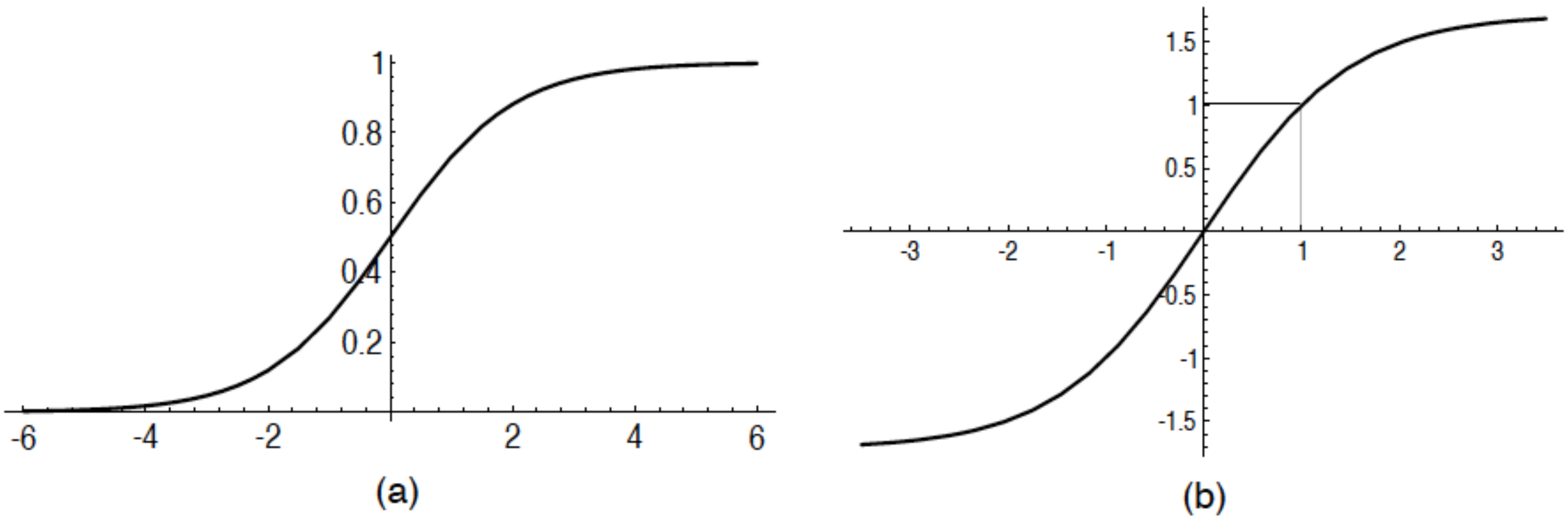
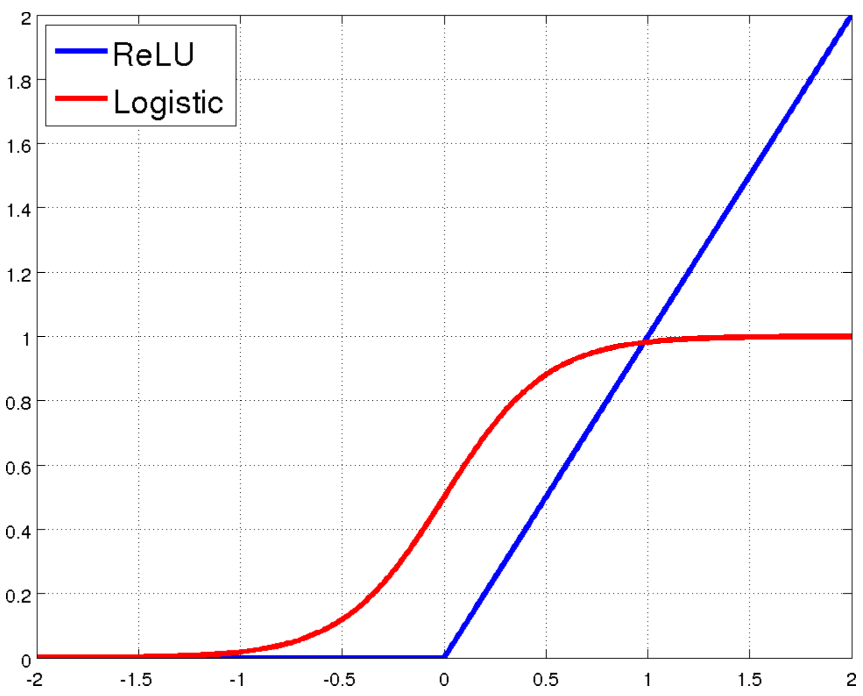
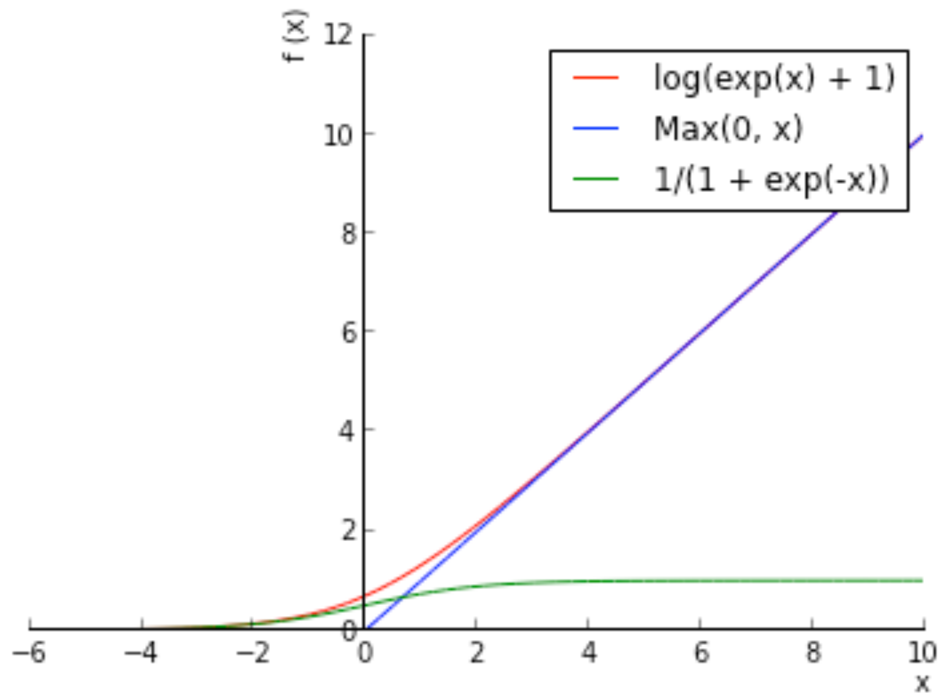


Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1+e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159\ \tanh\left(\frac{2}{3}x\right)$.

Image Credit: LeCun et al. '98

Rectified Linear Units (ReLU)





Convolutional Nets

- Basic Idea
  - On board
  - Assumptions:
    - Local Receptive Fields
    - Weight Sharing / Translational Invariance / Stationarity
  - Each layer is just a convolution!

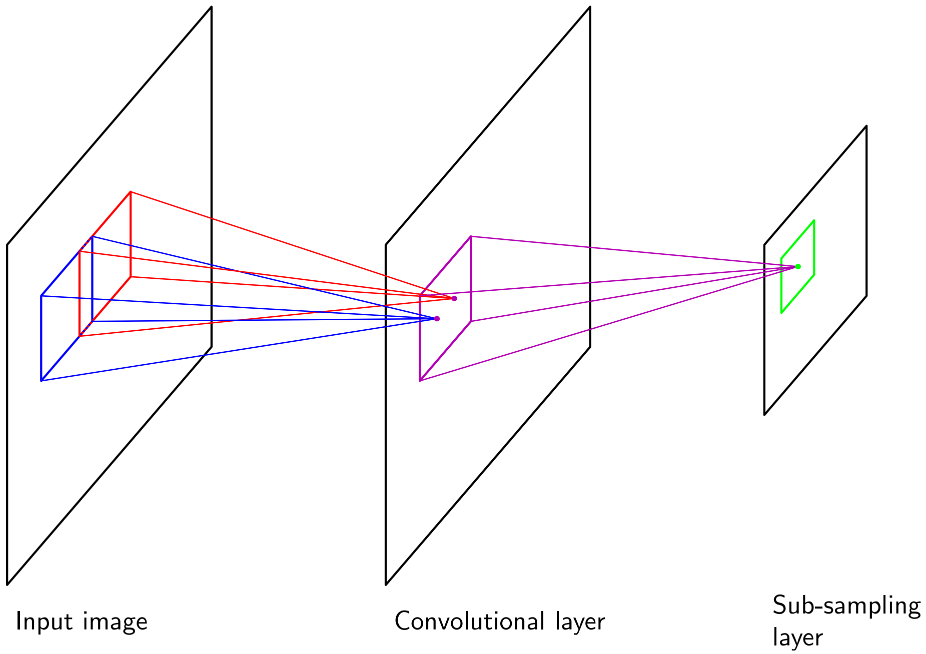


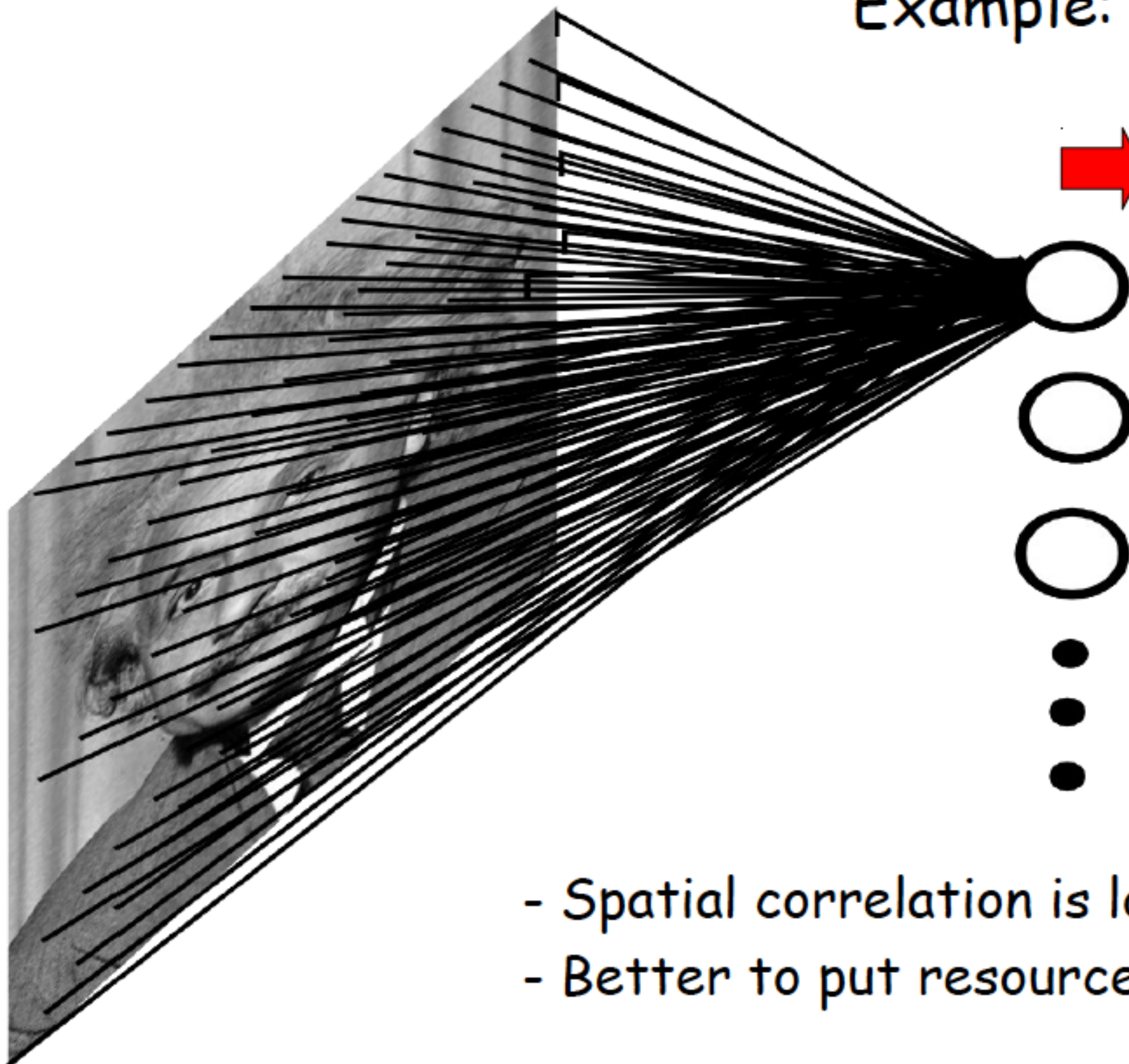
Image Credit: Chris Bishop

FULLY CONNECTED NEURAL NET

Example: 1000x1000 image

1M hidden units

→ 10^12 parameters!!!

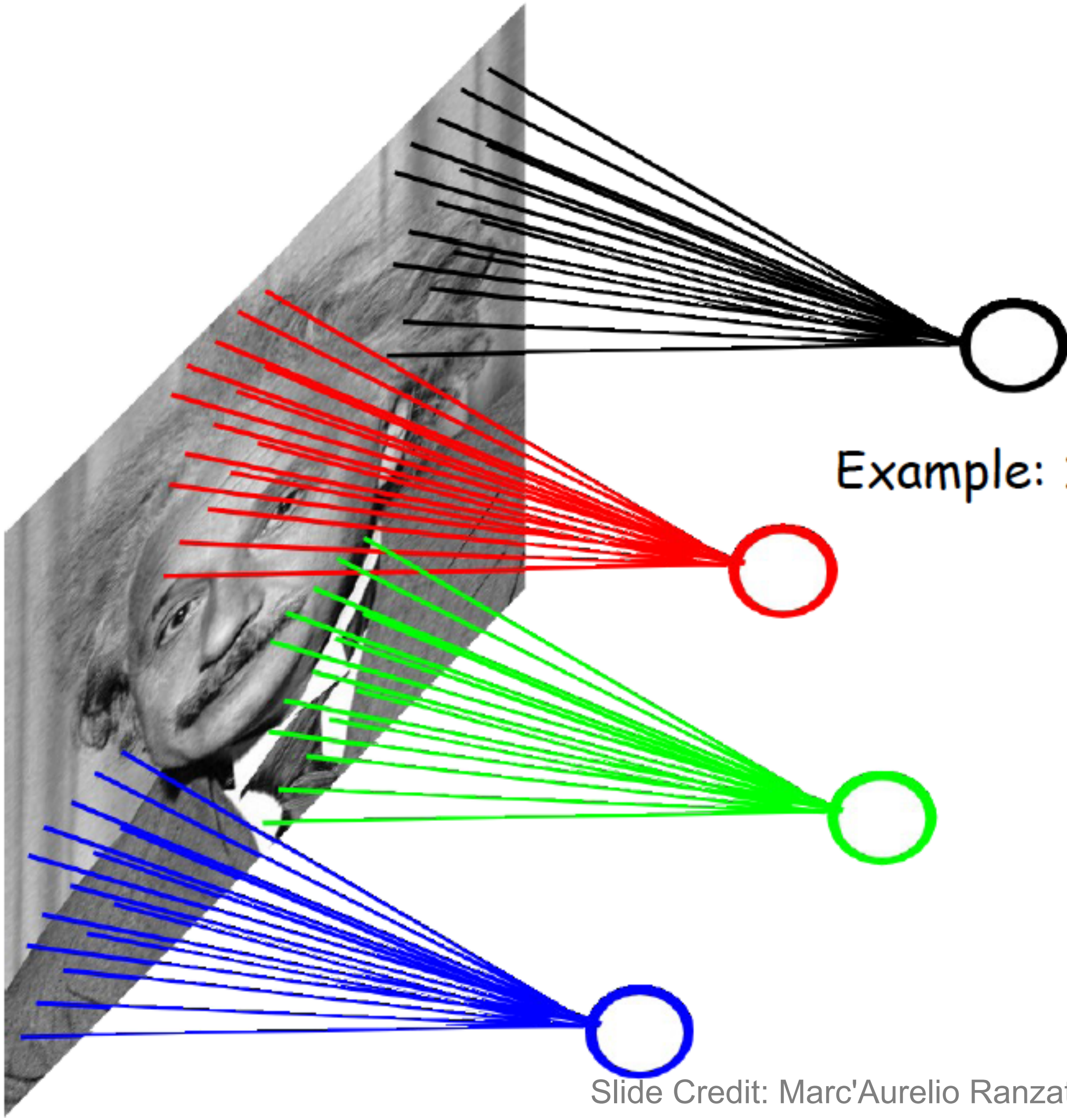


- Spatial correlation is local
- Better to put resources elsewhere!

Ranzato

Slide Credit: Marc'Aurelio Ranzato

LOCALLY CONNECTED NEURAL NET

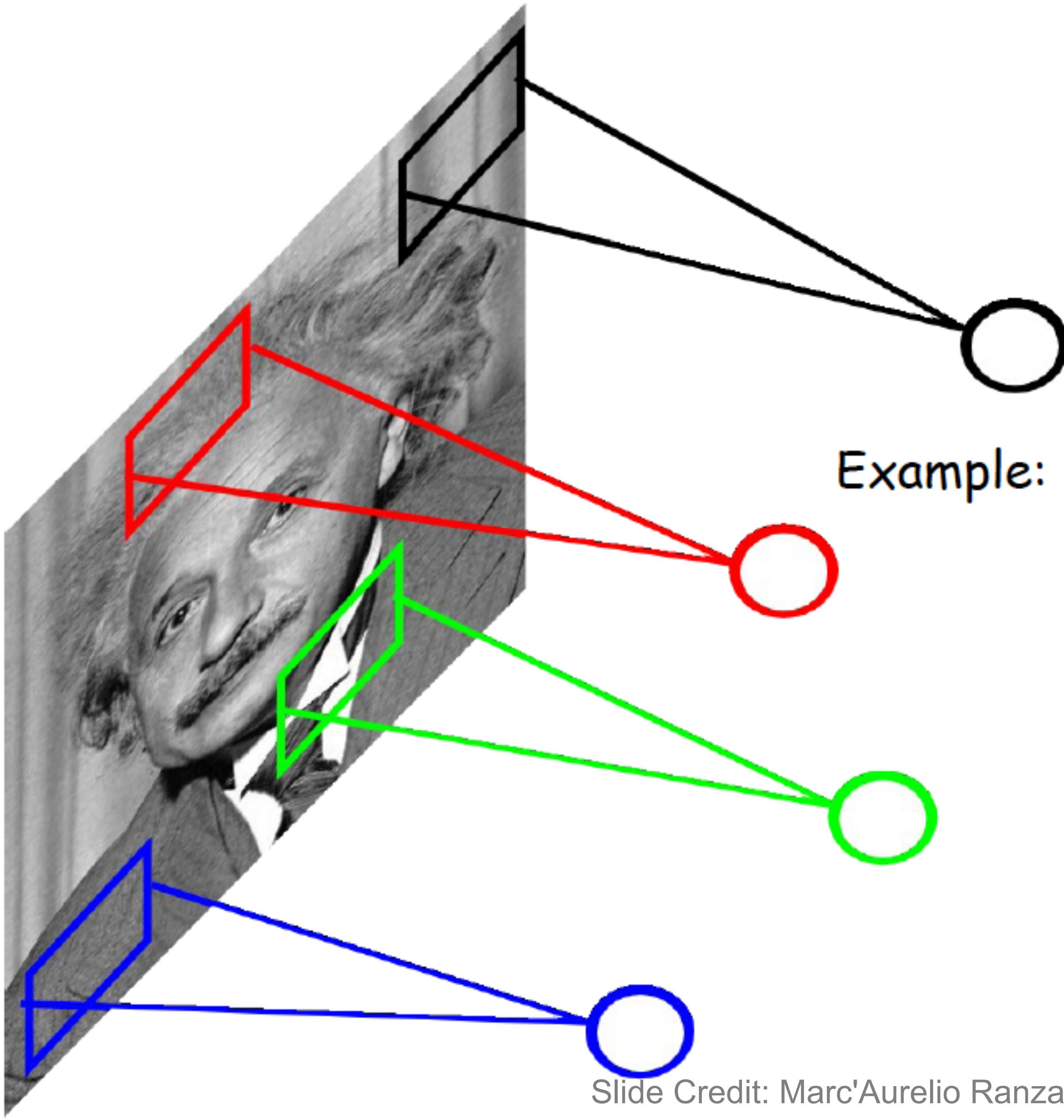


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Slide Credit: Marc'Aurelio Ranzato

Ranzato

LOCALLY CONNECTED NEURAL NET

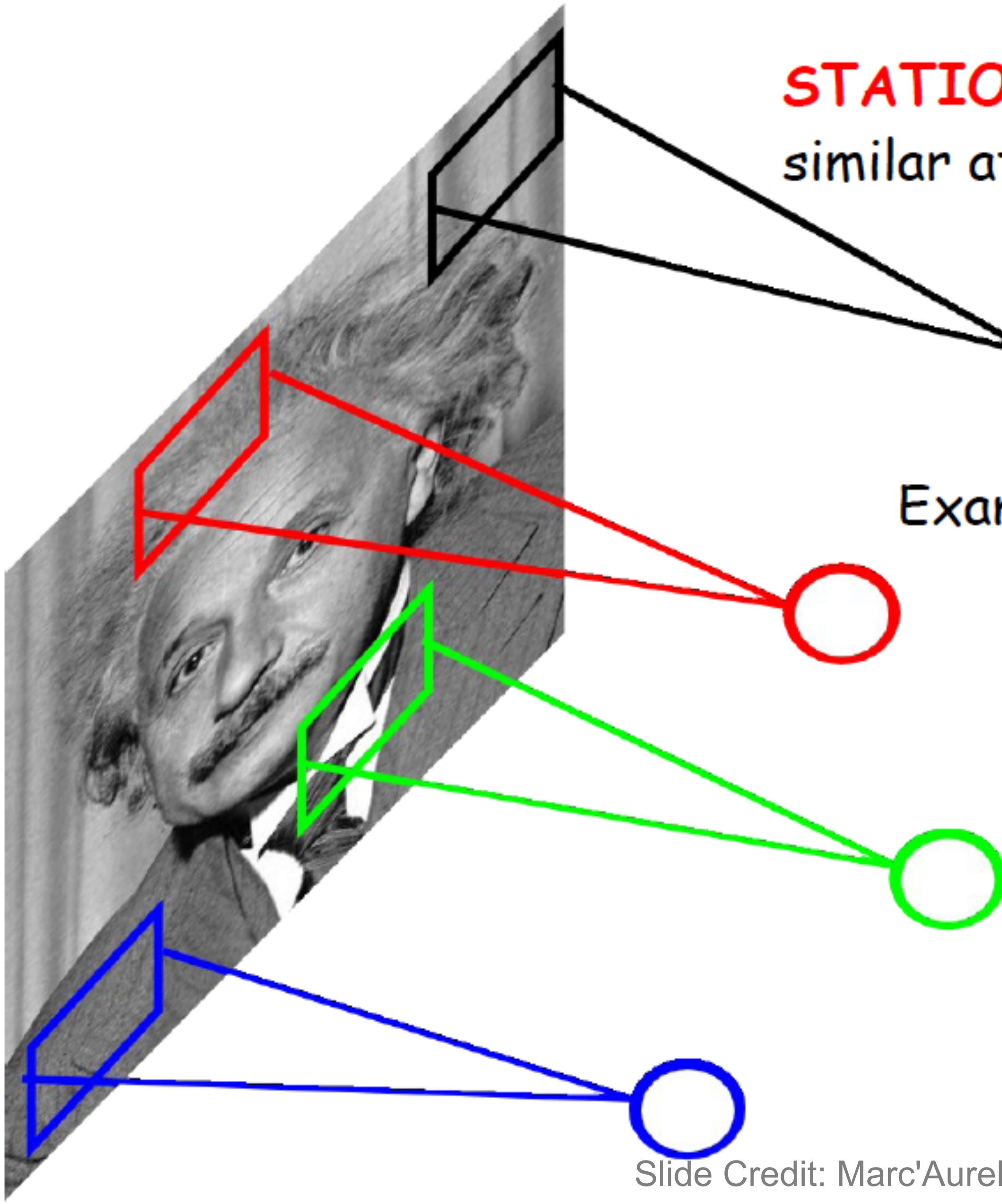


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Slide Credit: Marc'Aurelio Ranzato

Ranzato

LOCALLY CONNECTED NEURAL NET



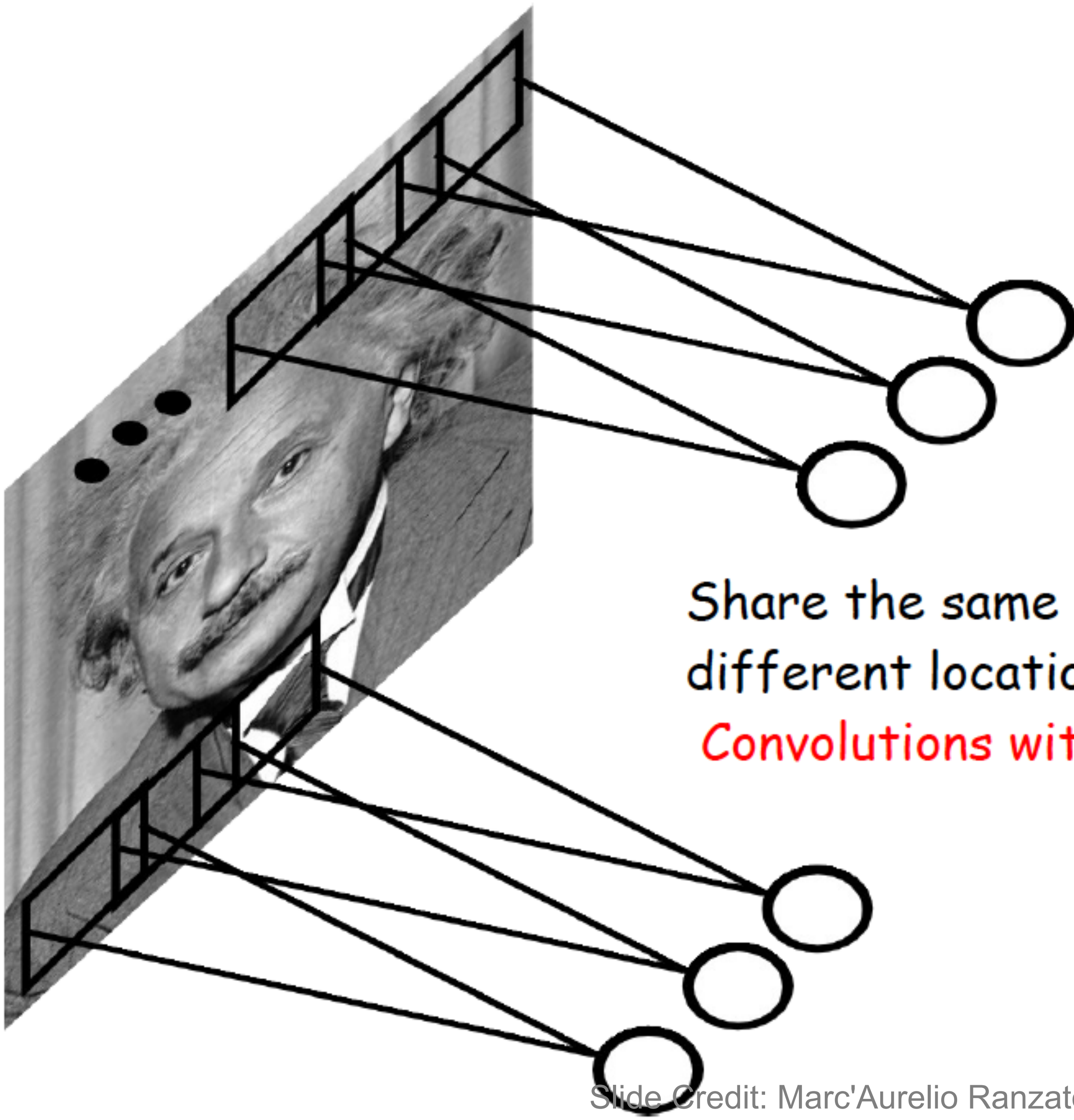
STATIONARITY? Statistics is similar at different locations

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Slide Credit: Marc'Aurelio Ranzato

Ranzato

CONVOLUTIONAL NET

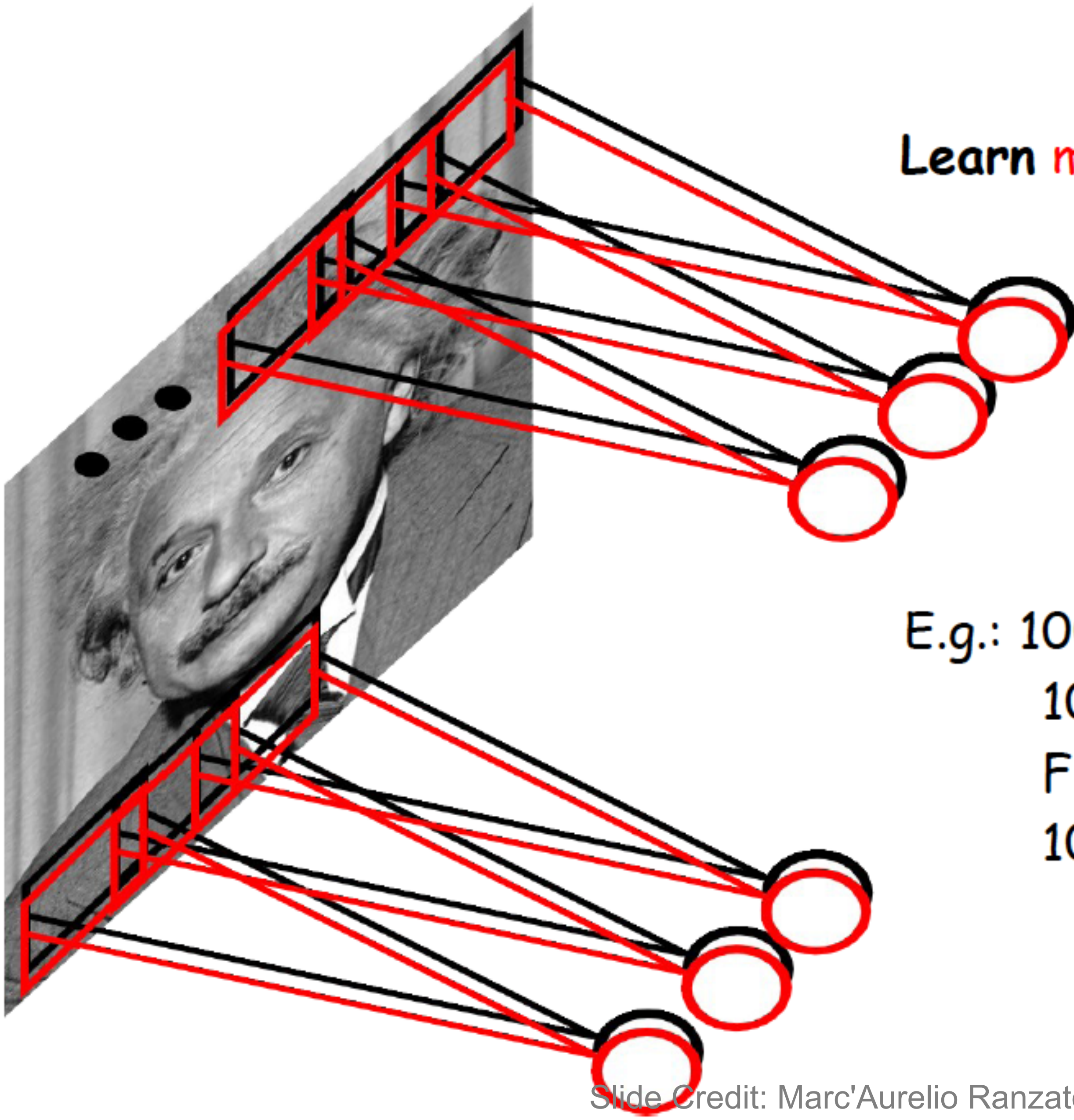


Share the same parameters across different locations:
Convolutions with learned kernels

Ranzato

Slide Credit: Marc'Aurelio Ranzato

CONVOLUTIONAL NET



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

Ranzato

Slide Credit: Marc'Aurelio Ranzato

NEURAL NETS FOR VISION

A standard neural net applied to images:

- scales quadratically with the size of the input
- does not leverage stationarity

Solution:

- connect each hidden unit to a small patch of the input
- share the weight across hidden units

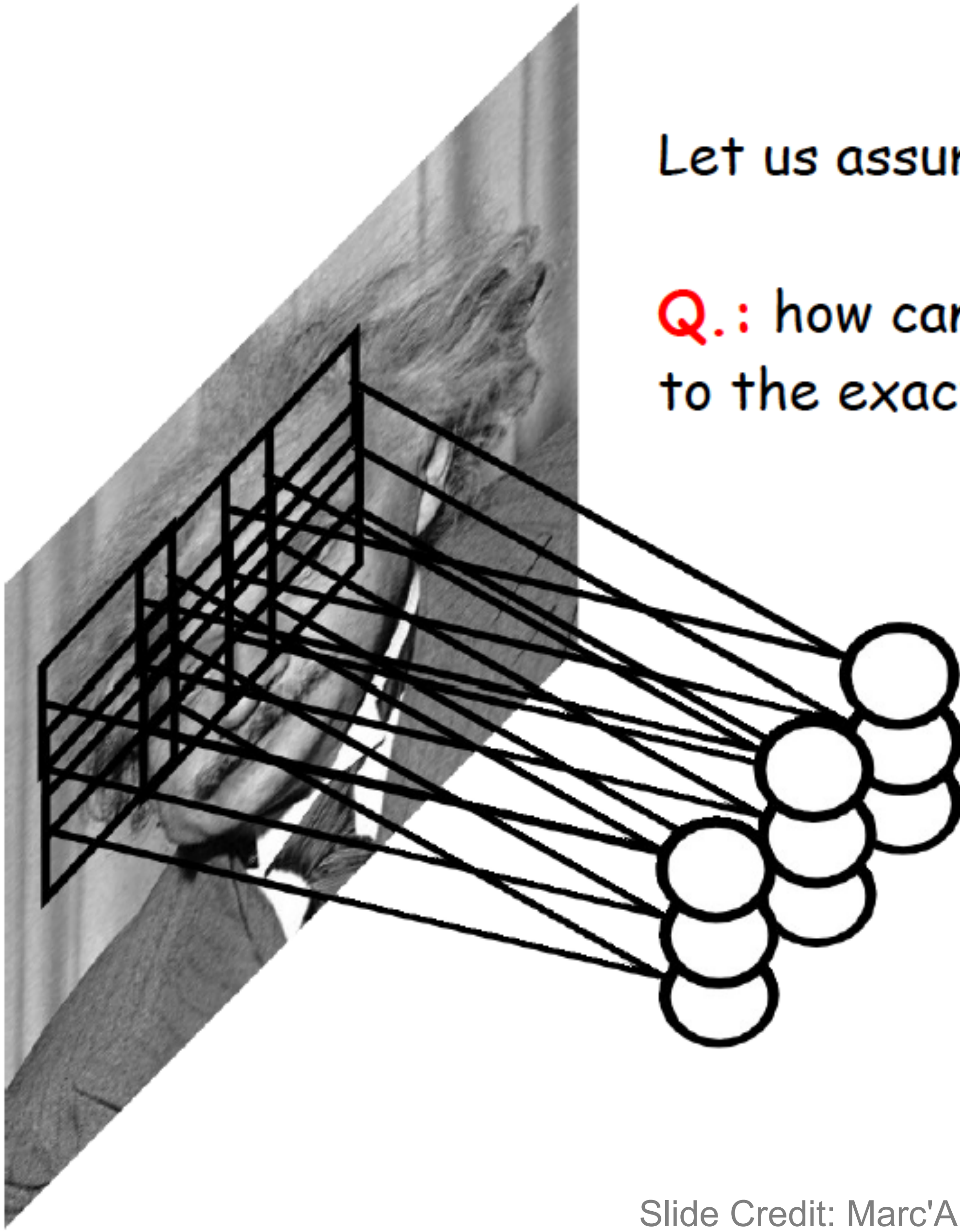
This is called: **convolutional network.**

LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

Slide Credit: Marc'Aurelio Ranzato

Ranzato

CONVOLUTIONAL NET



Let us assume filter is an "eye" detector.

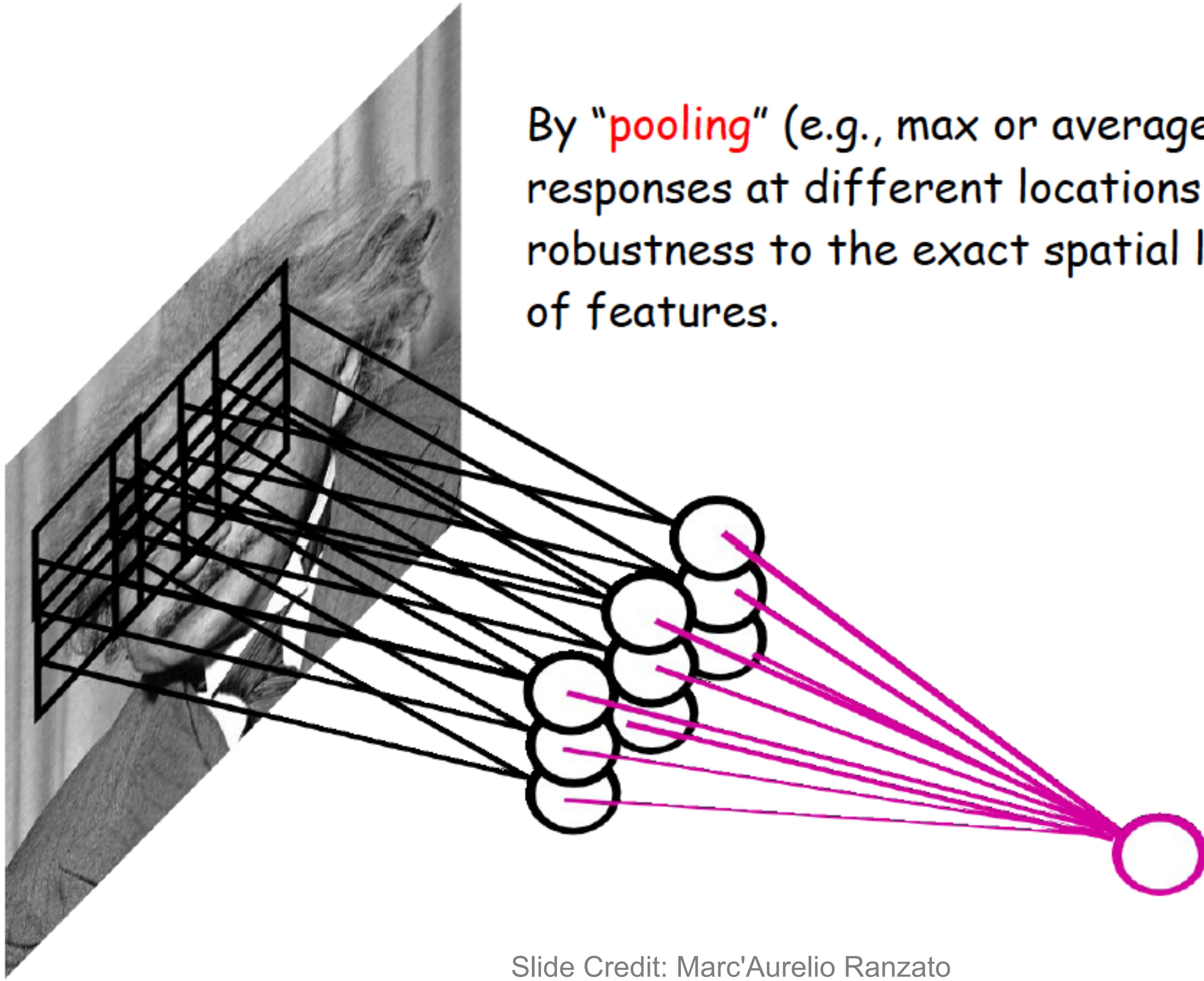
Q.: how can we make the detection robust to the exact location of the eye?

Ranzato

Slide Credit: Marc'Aurelio Ranzato

CONVOLUTIONAL NET

By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

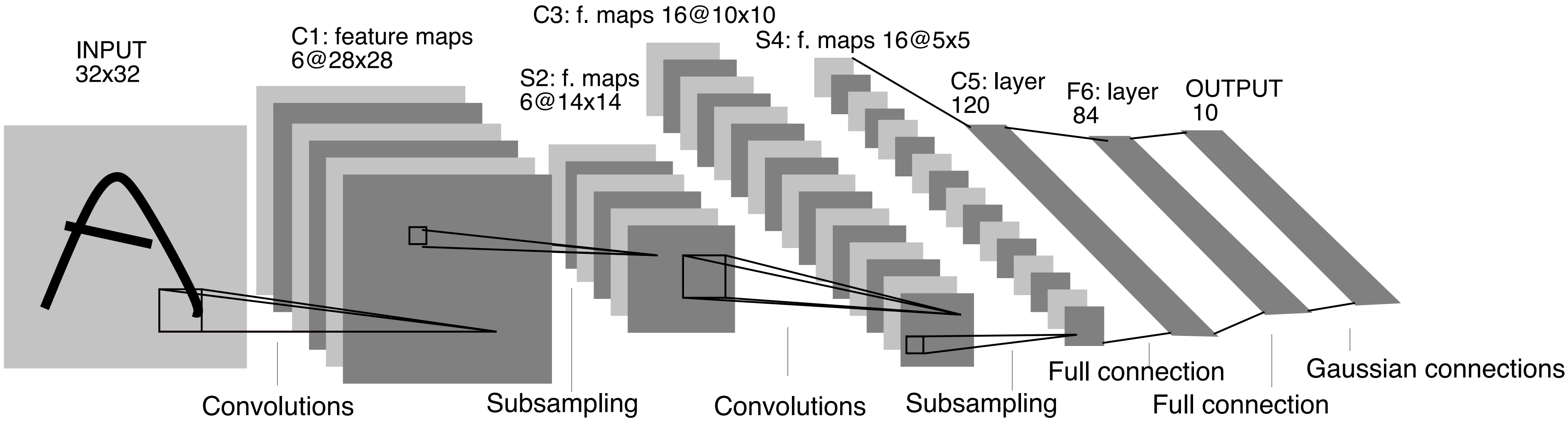


Ranzato

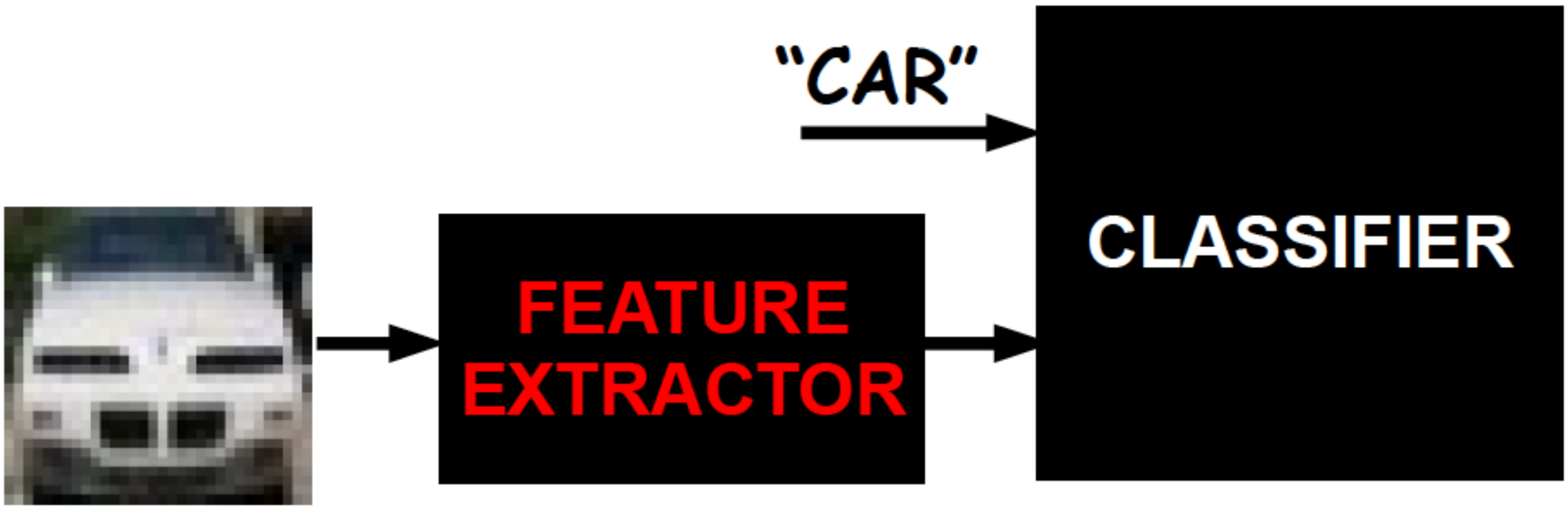
Slide Credit: Marc'Aurelio Ranzato

Convolutional Nets

- Example:
  - http://yann.lecun.com/exdb/lenet/index.html



Building an Object Recognition System

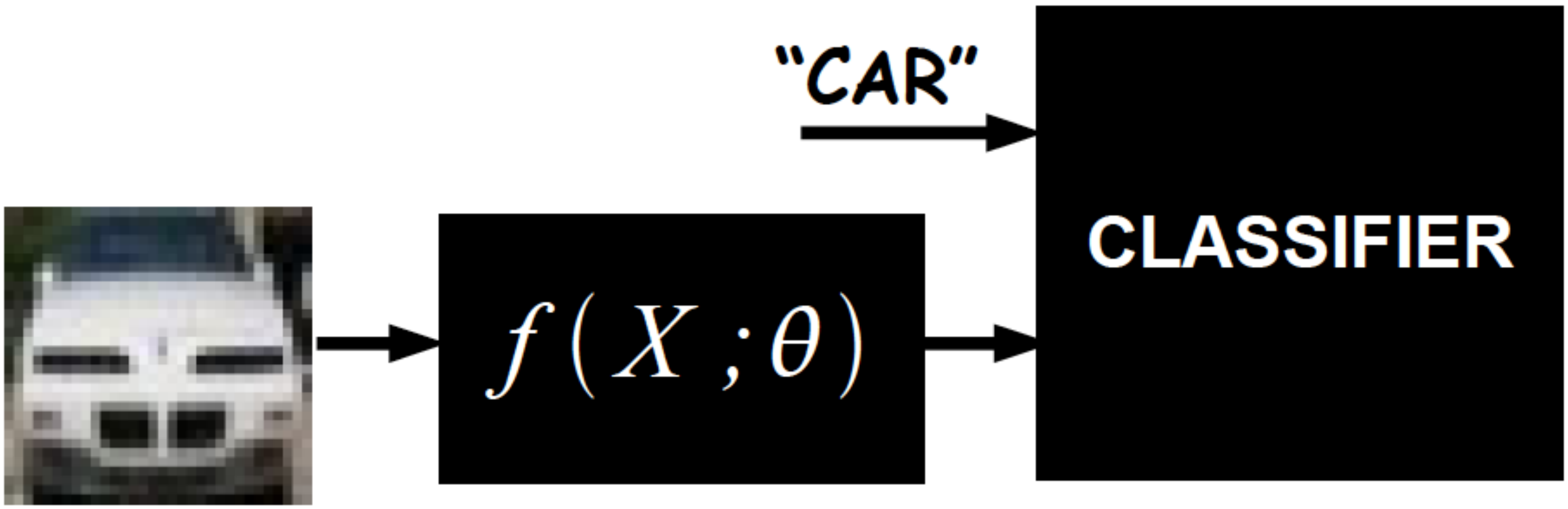


IDEA: Use data to optimize features for the given task.

Ranzato

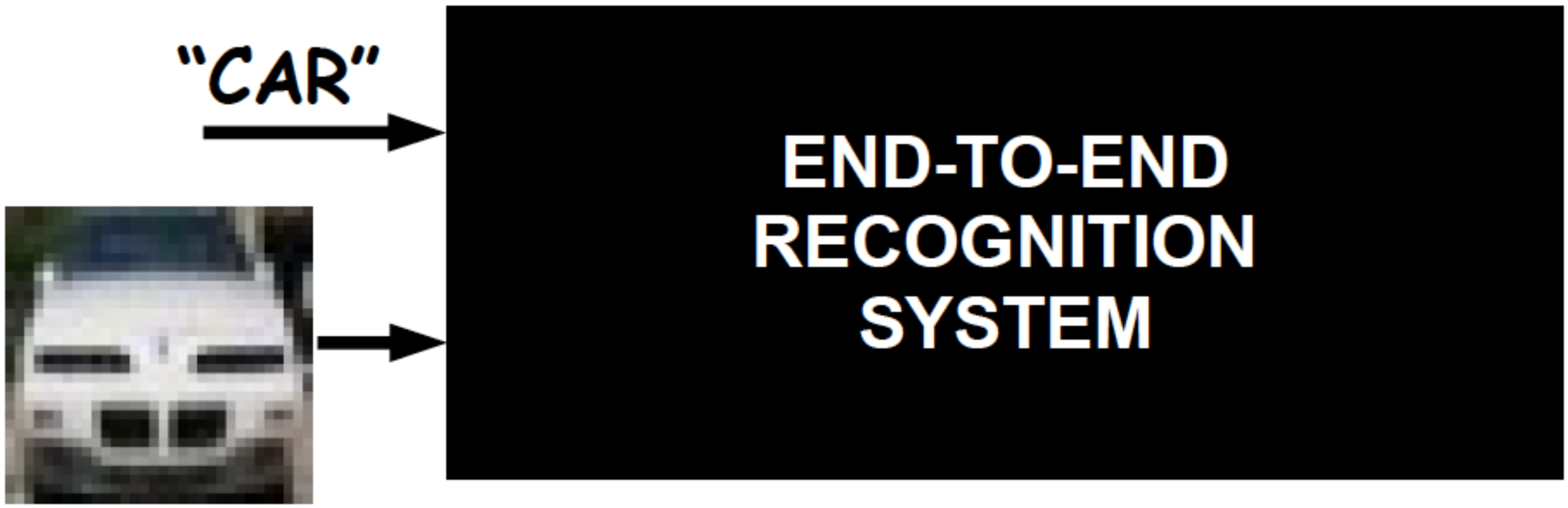
Slide Credit: Marc'Aurelio Ranzato

Building an Object Recognition System



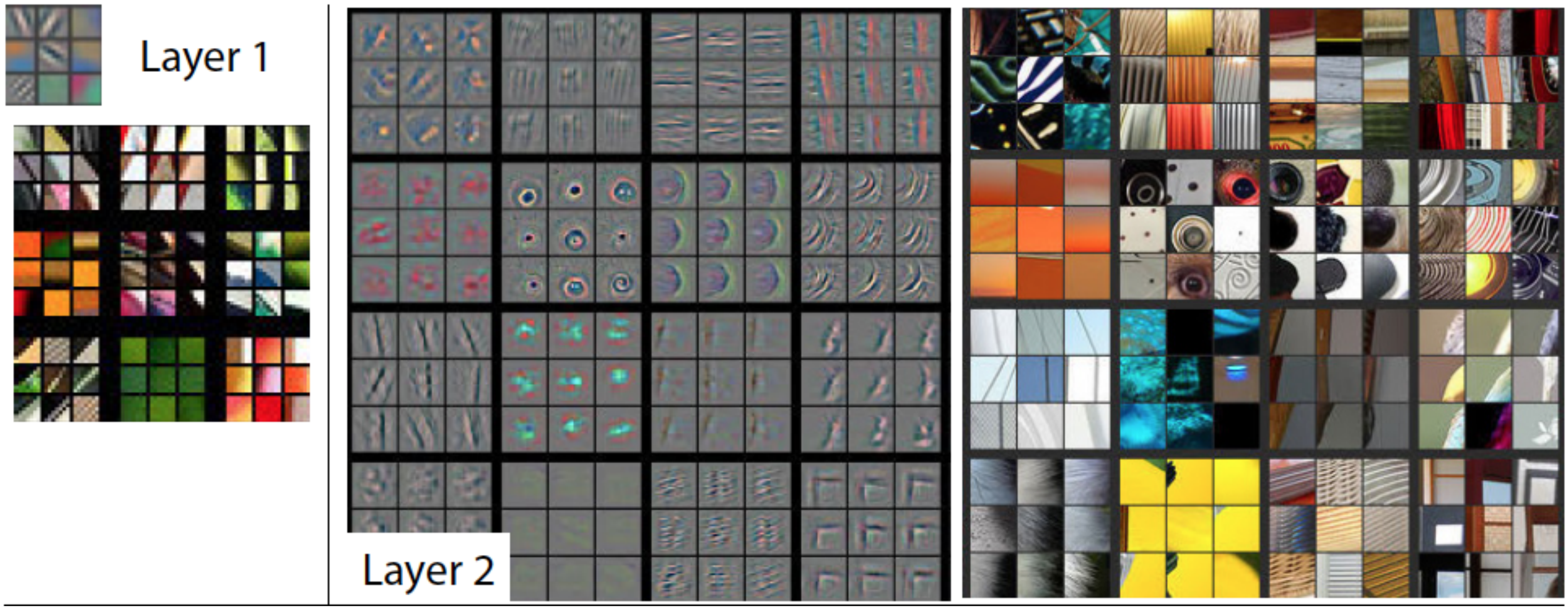
What we want: Use parameterized function such that
a) features are computed efficiently
b) features can be trained efficiently

Building an Object Recognition System

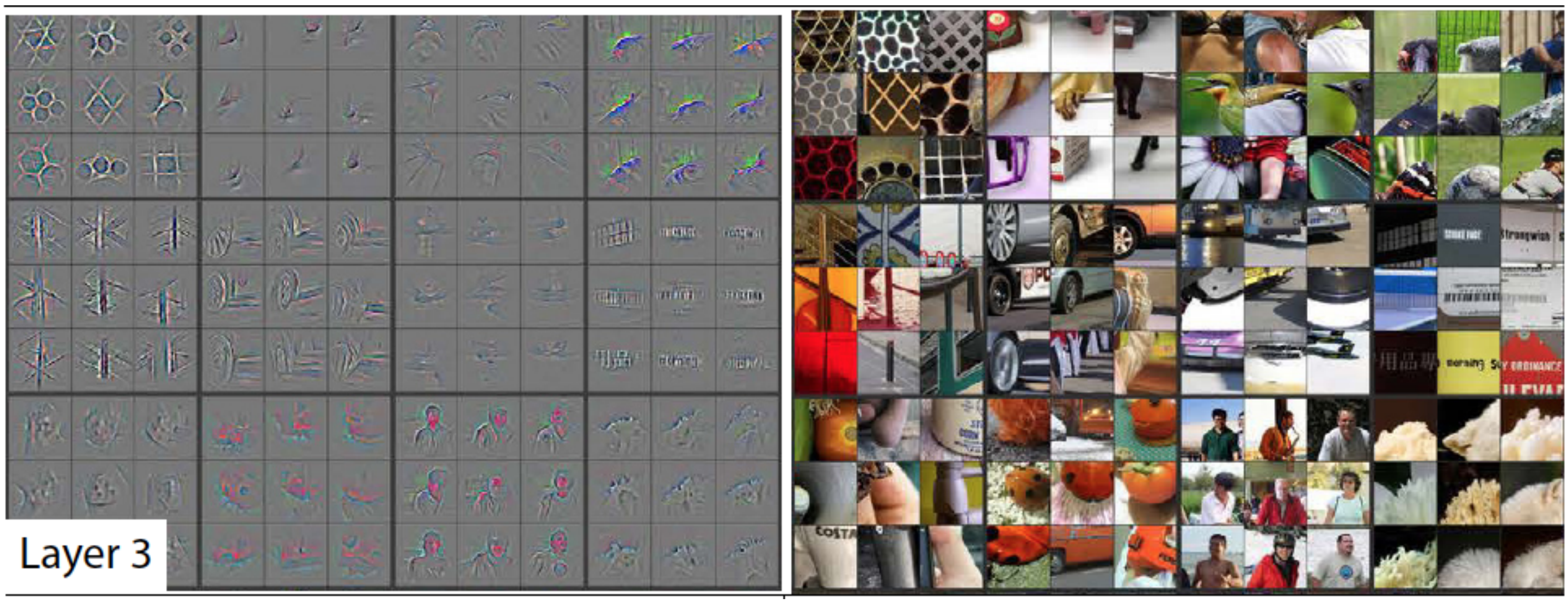


- Everything becomes adaptive.
- No distiction between feature extractor and classifier.
- Big non-linear system trained from raw pixels to labels.

Visualizing Learned Filters



Visualizing Learned Filters



Visualizing Learned Filters

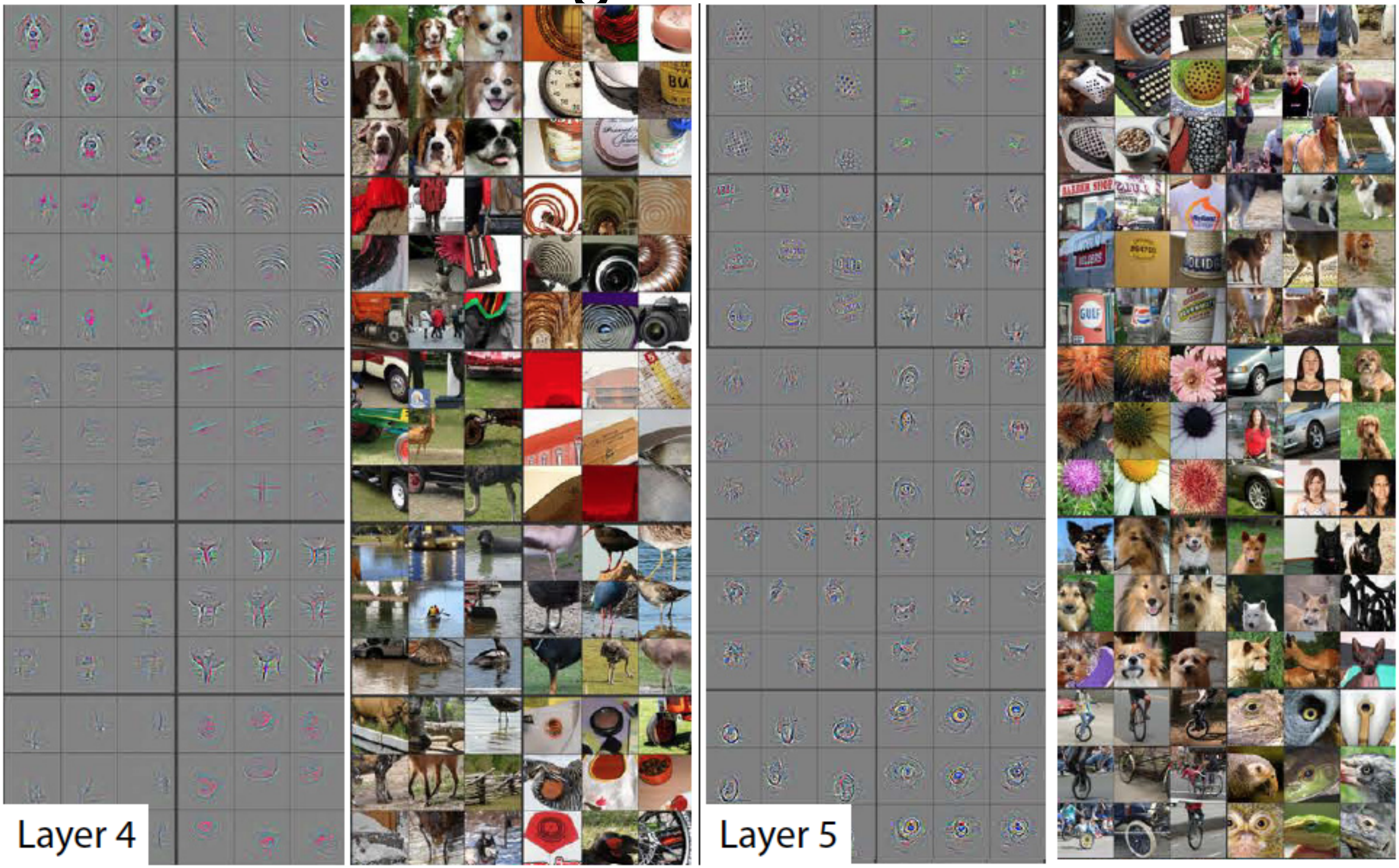
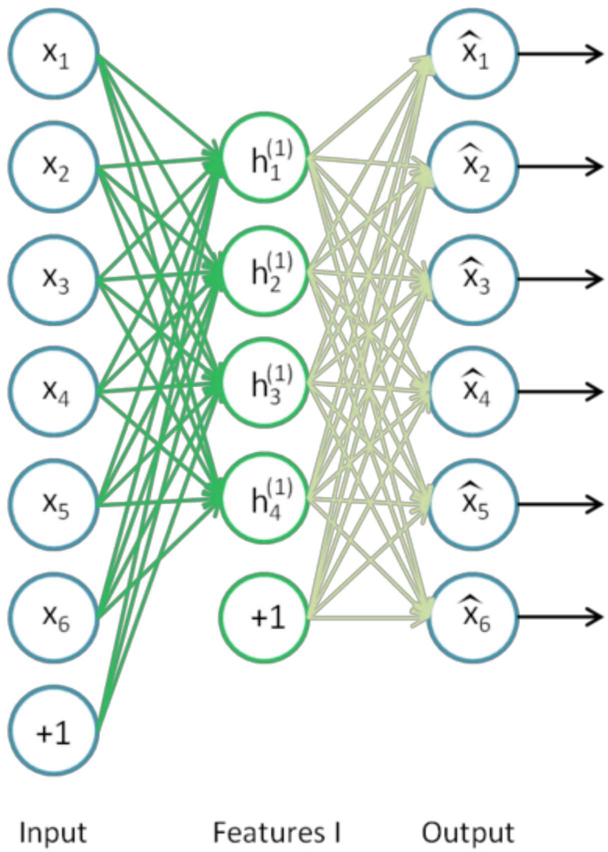


Figure Credit: [Zeiler & Fergus ECCV14]

Autoencoders

- Goal
  - Compression: Output tries to predict input



Autoencoders

- Goal
  - Learns a low-dimensional “basis” for the data

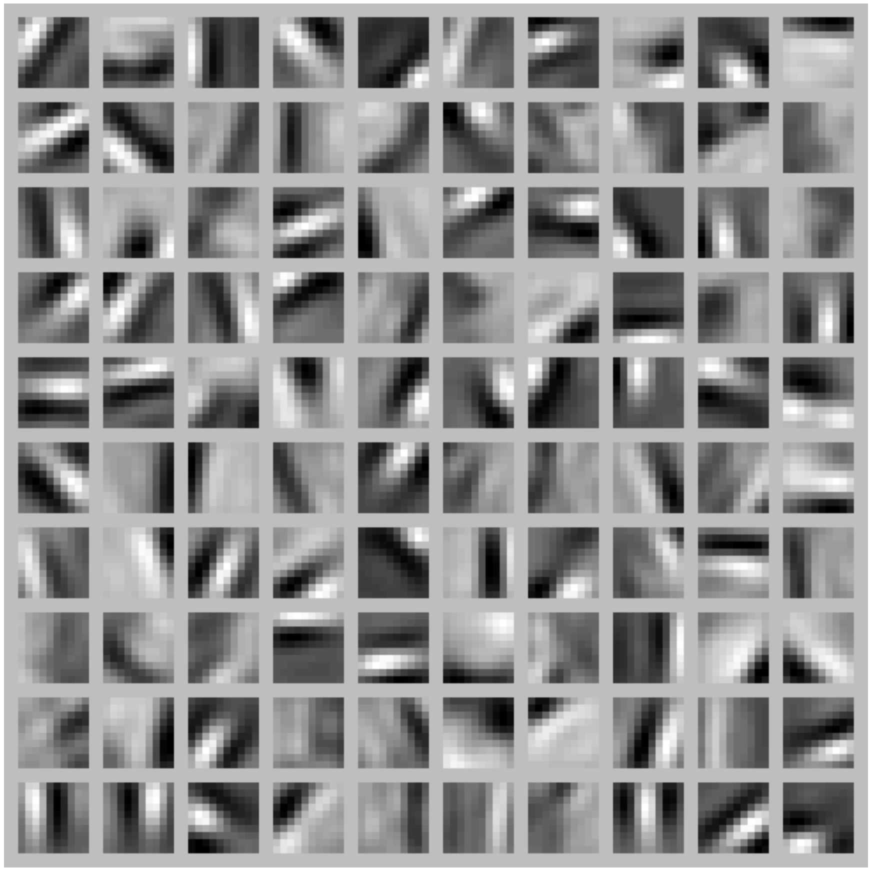
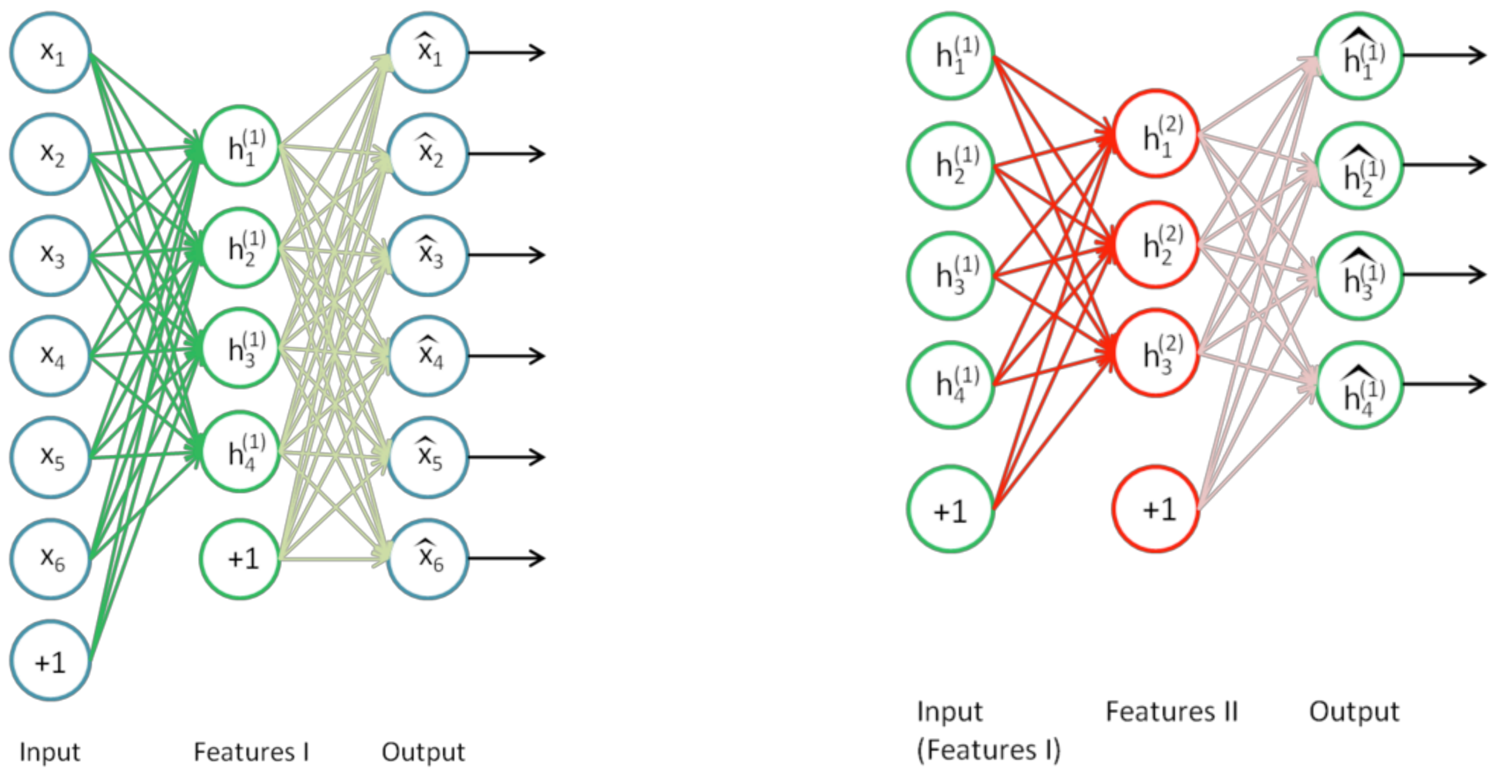
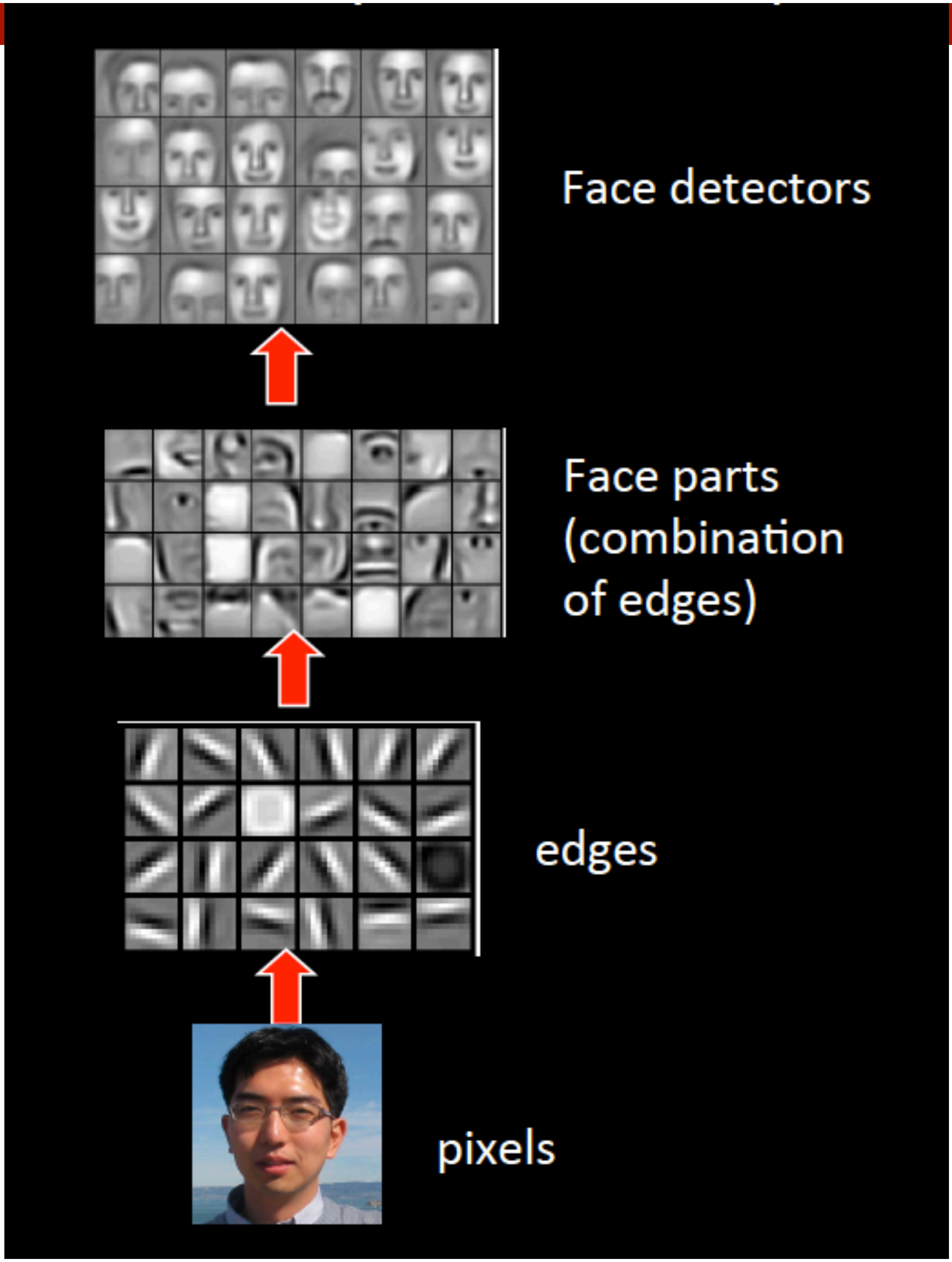


Image Credit: Andrew Ng

Stacked Autoencoders

- How about we compress the low-dim features more?





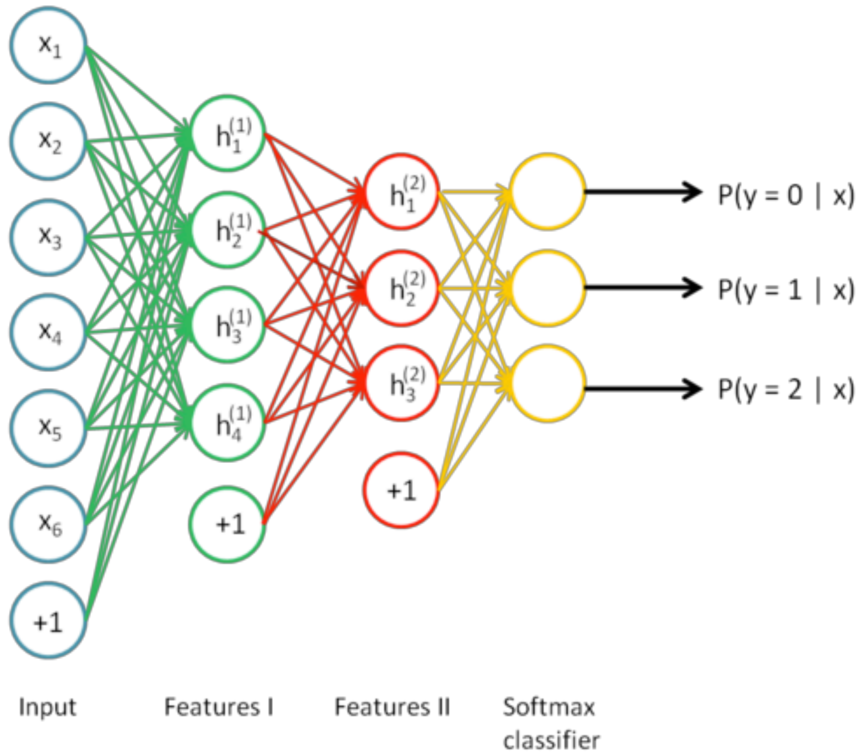
Sparse DBNs

[Lee et al. ICML '09]

Figure courtesy: Quoc Le

Stacked Autoencoders

- Finally perform classification with these low-dim features.



What you need to know about neural networks

- Perceptron:
  - Representation
  - Derivation

- Multilayer neural nets
  - Representation
  - Derivation of backprop
  - Learning rule
  - Expressive power