# NEURAL NETS FOR VISION

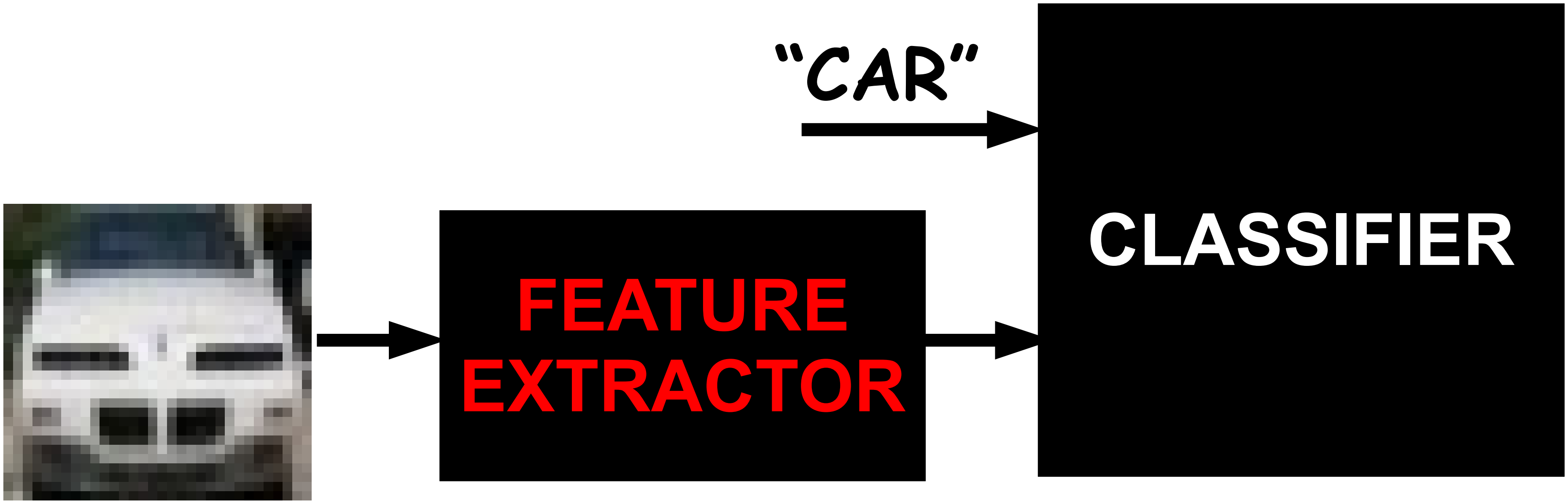
## CVPR 2012 Tutorial on Deep Learning
## Part II


**Marc'Aurelio Ranzato** - Google

ranzato@google.com

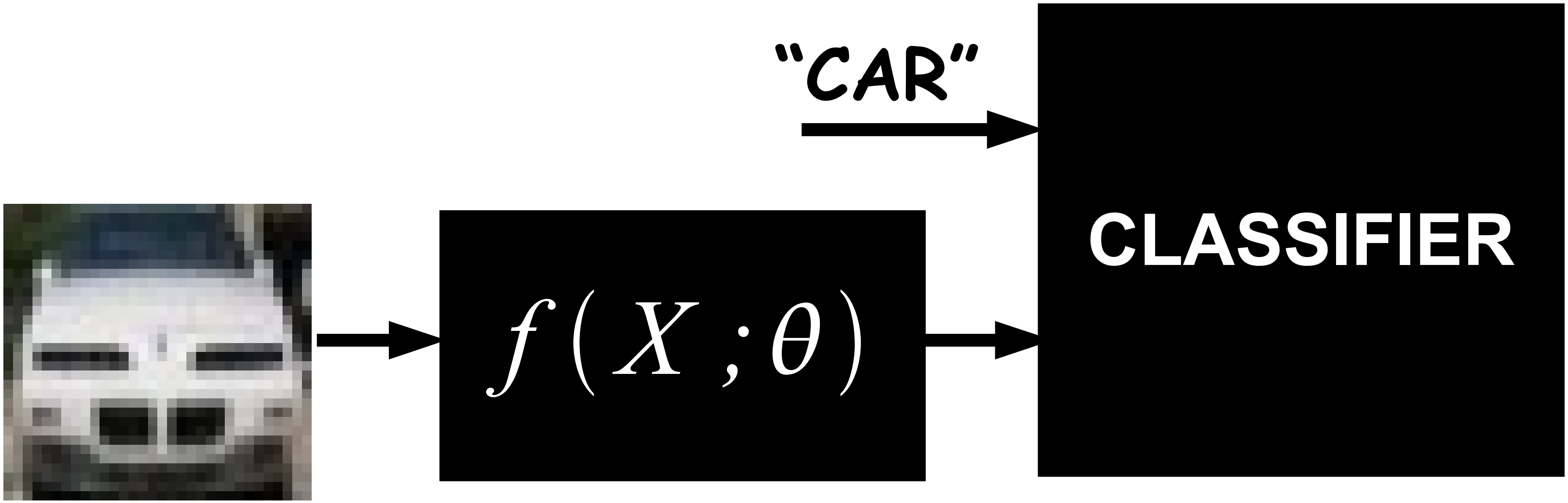
www.cs.toronto.edu/~ranzato

# Building an Object Recognition System




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

# 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

"CAR"

**END-TO-END RECOGNITION SYSTEM**

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

# Building an Object Recognition System


**Q:** How can we build such a highly non-linear system?

# Building an Object Recognition System

"CAR"

END-TO-END
RECOGNITION
SYSTEM

**Q:** How can we build such a highly non-linear system?

**A:** By combining simple building blocks we can make more and more complex systems.

# Building A Complicated Function

Simple Functions

$\sin(x)$

$\log(x)$

$\cos(x)$

$x^3$

$\exp(x)$

One Example of Complicated Function

$\log(\cos(\exp(\sin^3(x))))$

# Building A Complicated Function

Simple Functions

$\sin(x)$

$\log(x)$

$\cos(x)$

$x^3$

$\exp(x)$

One Example of Complicated Function

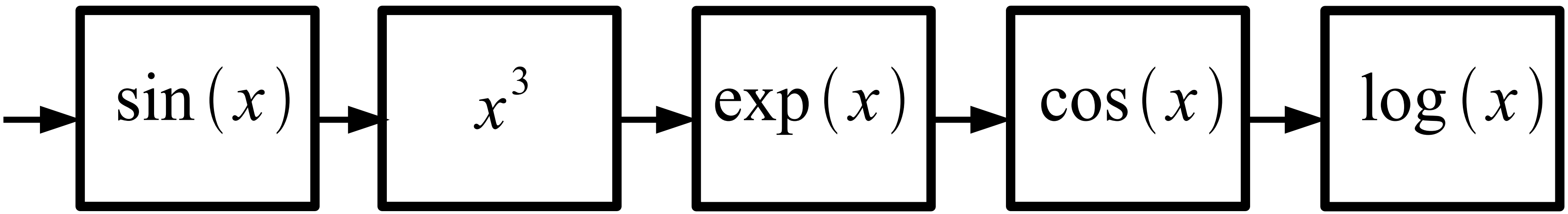
$\log(\cos(\exp(\sin^3(x))))$

- Function composition is at the core of deep learning methods.
- Each "simple function" will have parameters subject to training.

# Implementing A Complicated Function

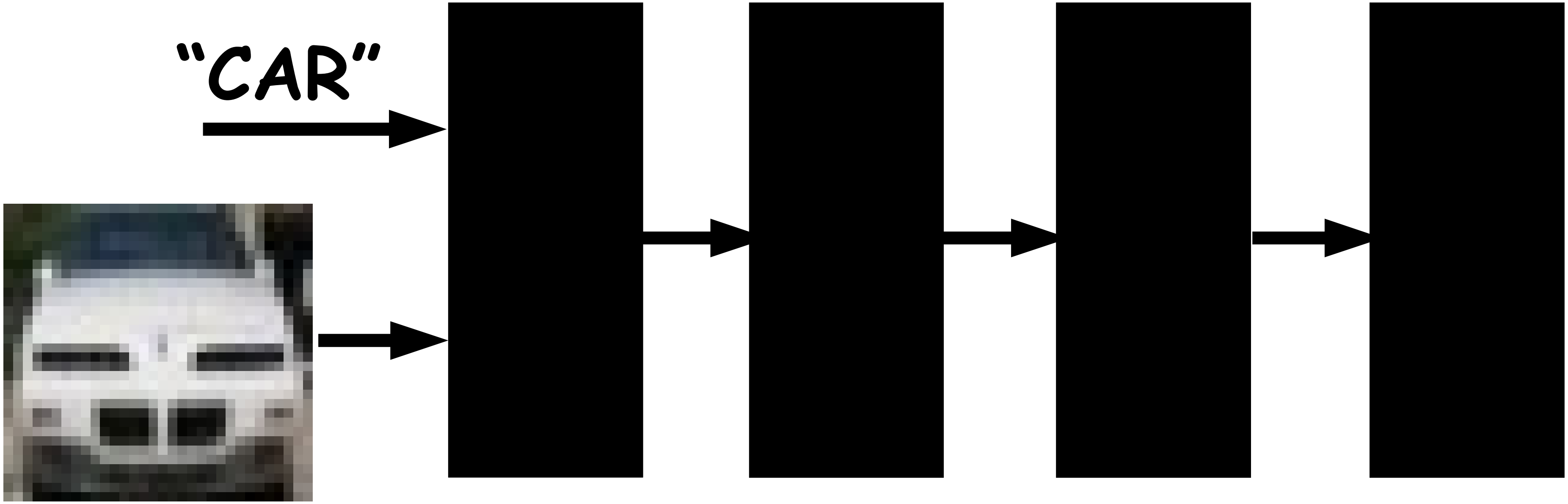
Complicated Function

$$\log(\cos(\exp(\sin^3(x))))$$



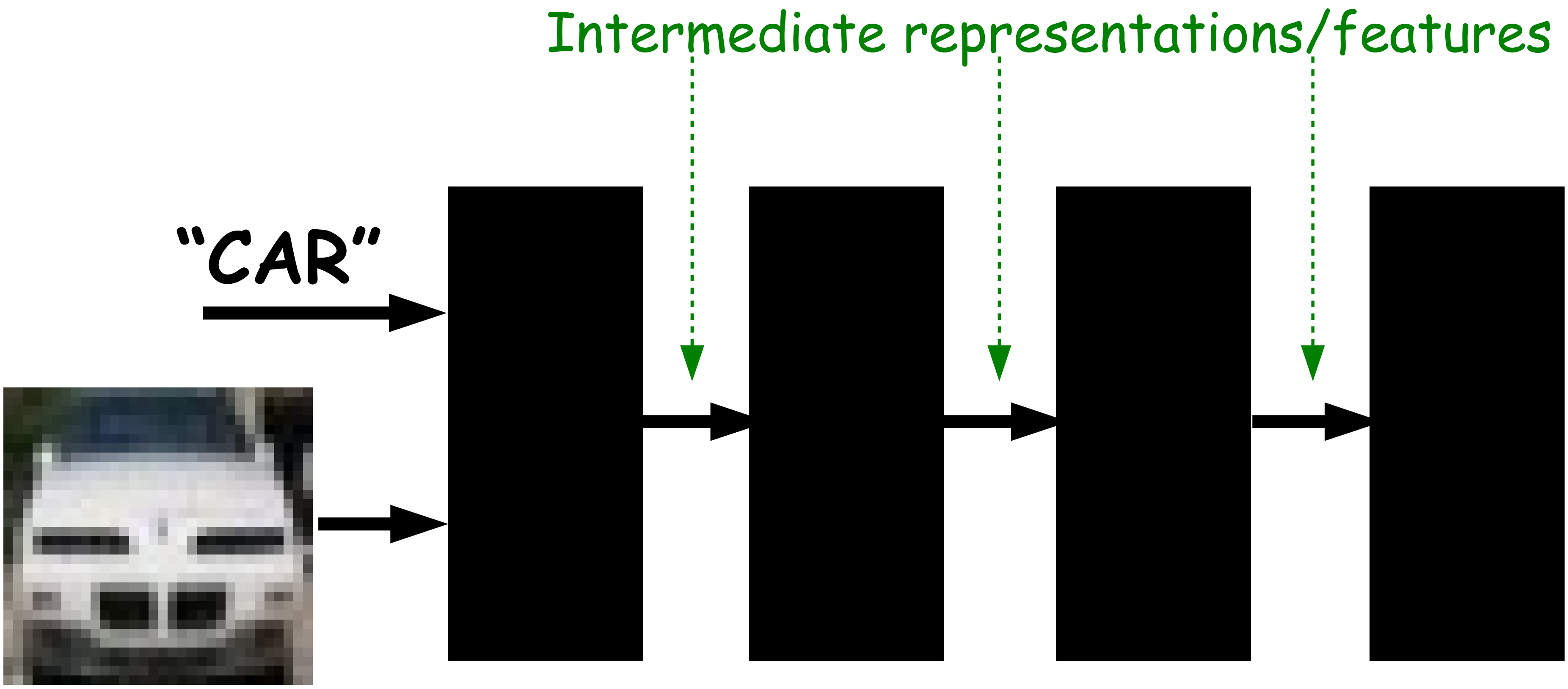
# Intuition Behind Deep Neural Nets

# Intuition Behind Deep Neural Nets



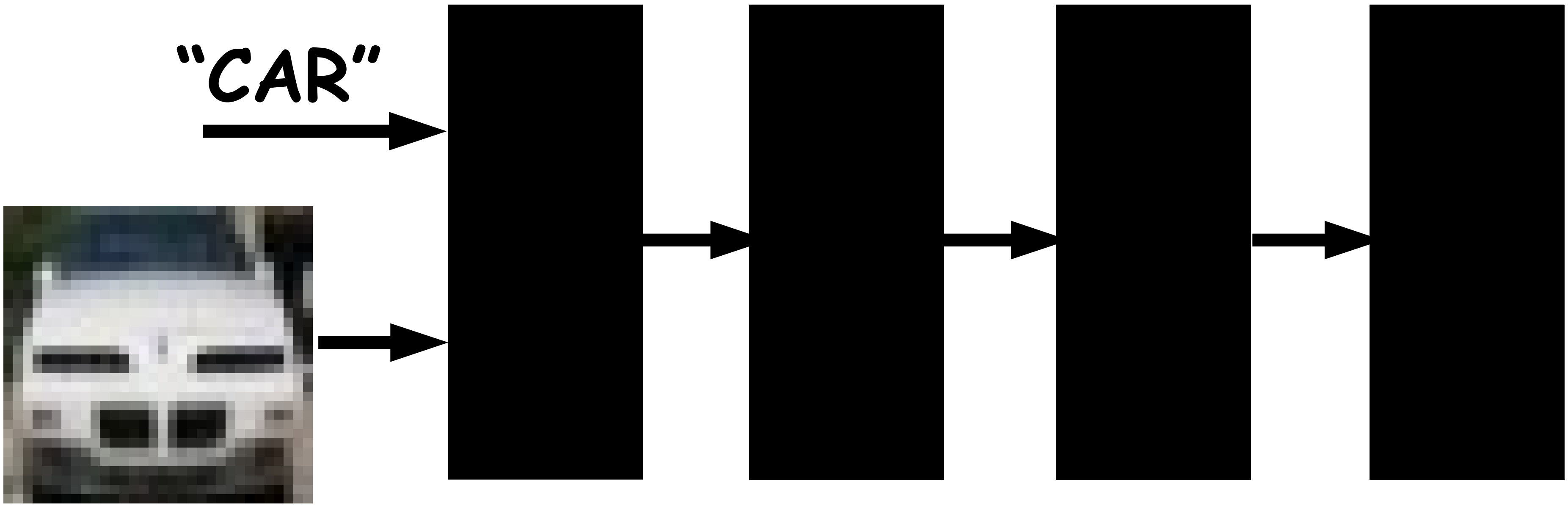

**NOTE:** Each black box can have trainable parameters.
Their composition makes a highly non-linear system.

# Intuition Behind Deep Neural Nets



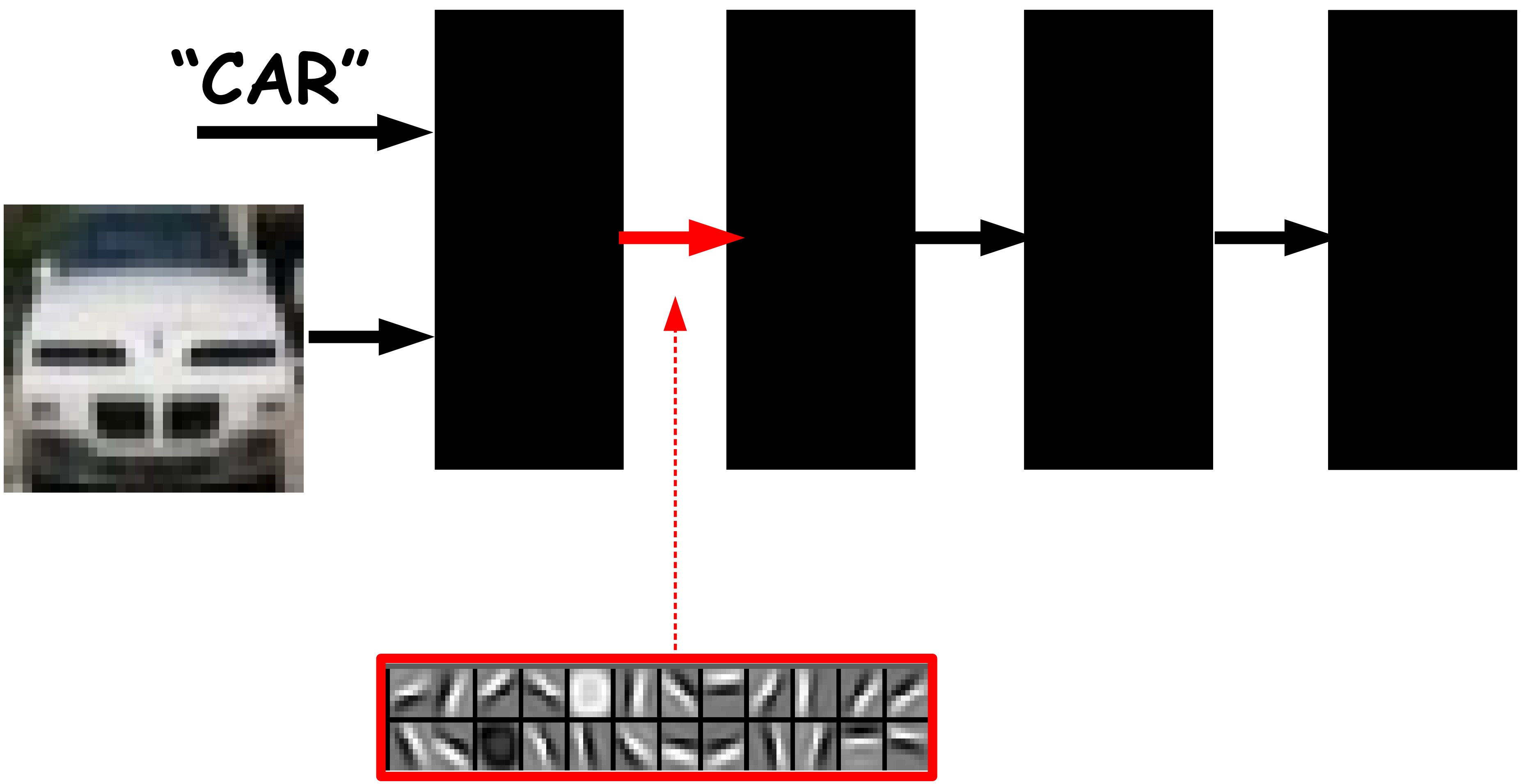

**NOTE:** System produces a hierarchy of features.

# Intuition Behind Deep Neural Nets




**Q:** What do the intermediate representations do?

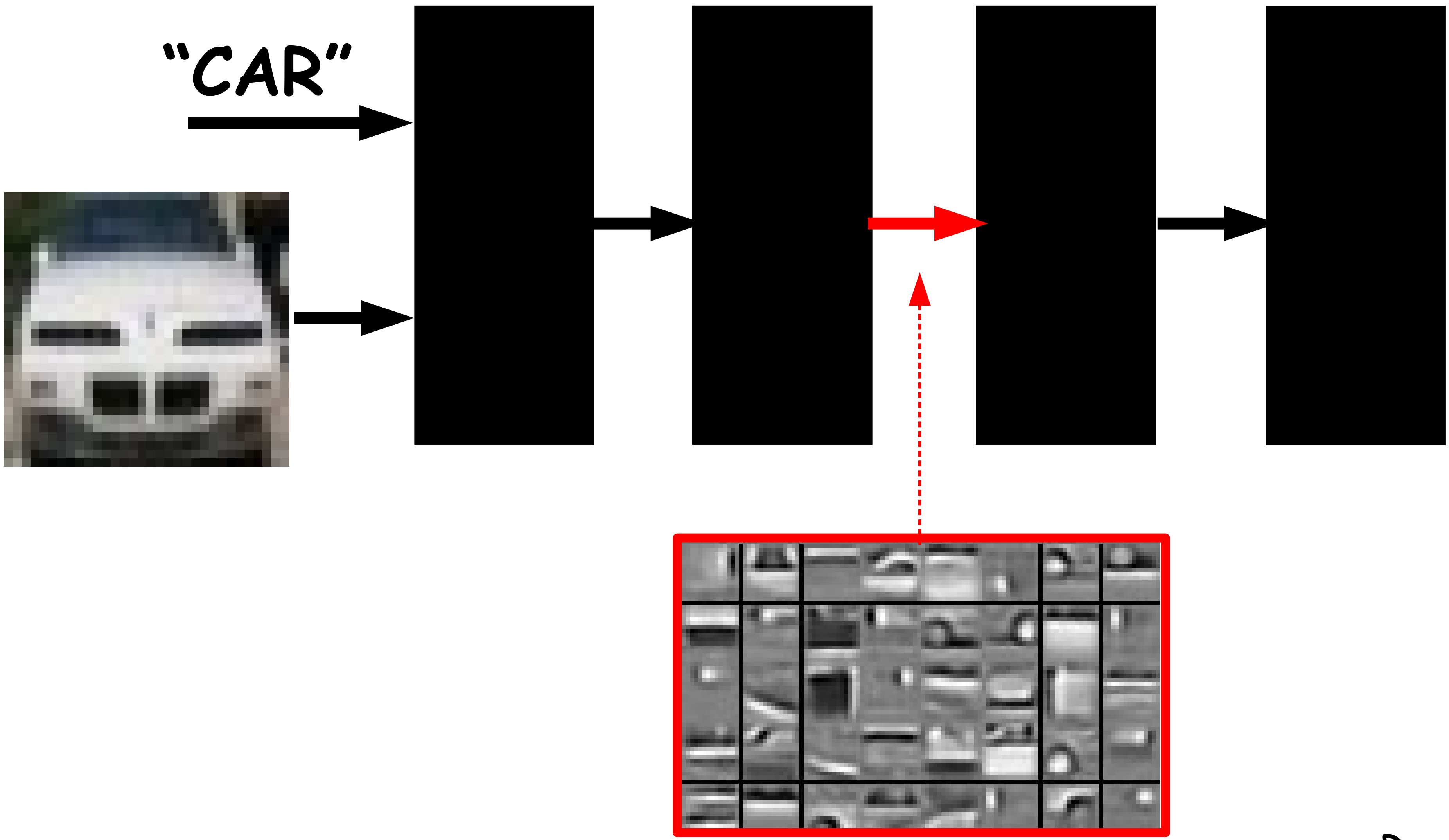
# Intuition Behind Deep Neural Nets





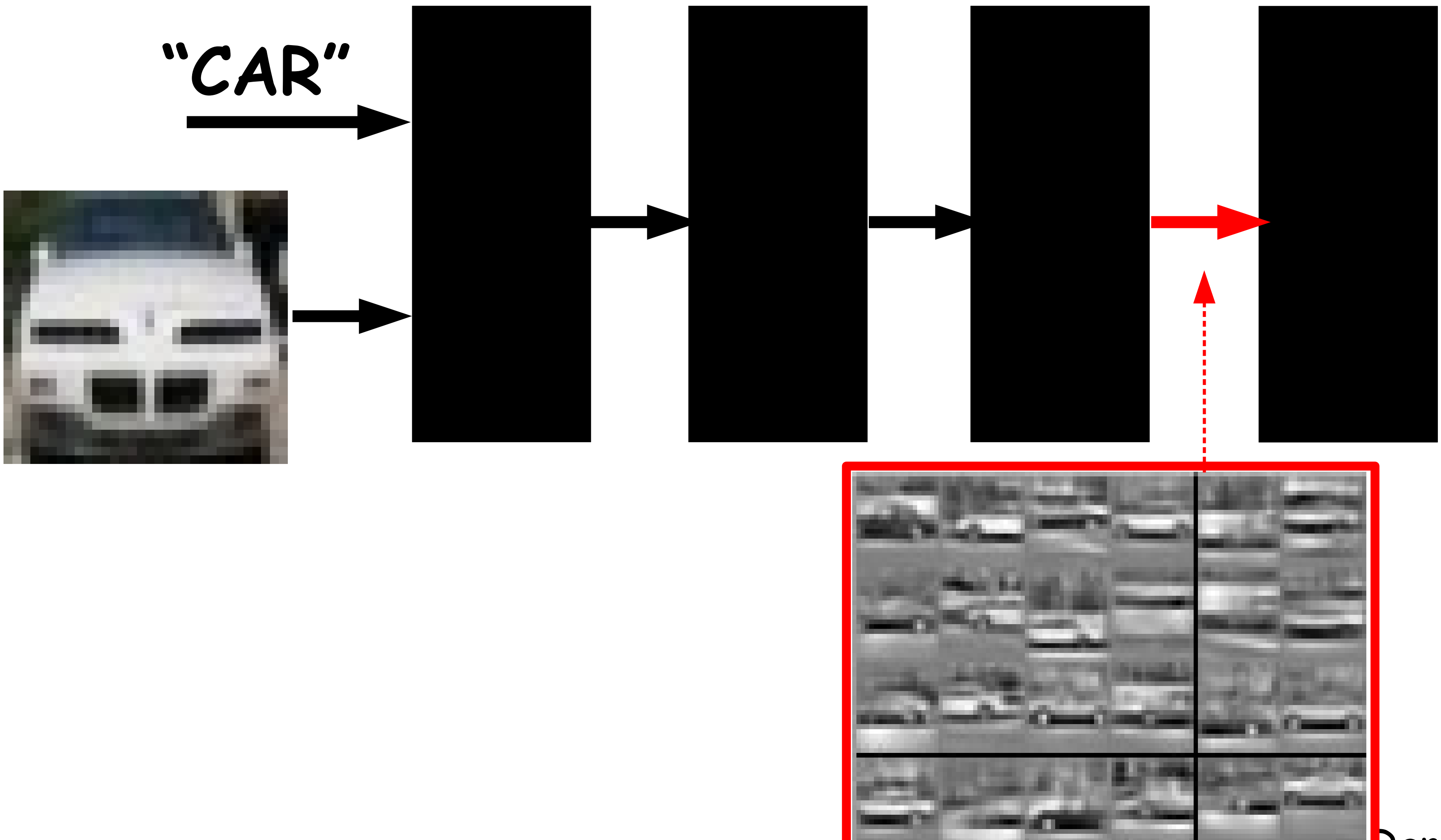

*Lee et al. "Convolutional DBN's for scalable unsup. learning..." ICML 2009*

# Intuition Behind Deep Neural Nets




Lee et al. ICML 2009

# Intuition Behind Deep Neural Nets




Lee et al. ICML 2009

# KEY IDEAS OF NEURAL NETS

## IDEA # 1

Learn features from data

## IDEA # 2

Use differentiable functions that produce features efficiently

## IDEA # 3

End-to-end learning:
no distinction between feature extractor and classifier

## IDEA # 4

"Deep" architectures:
cascade of simpler non-linear modules

# KEY QUESTIONS

- What is the input-output mapping?

- How are parameters trained?

- How computational expensive is it?

- How well does it work?

# Outline

- **Neural Networks for Supervised Training**
  - Architecture
  - Loss function
- **Neural Networks for Vision: Convolutional & Tiled**
- **Unsupervised Training of Neural Networks**
- **Extensions**:
  - semi-supervised / multi-task / multi-modal
- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models
- **Large-Scale Learning with Deep Neural Nets**

# Outline

- **Neural Networks for Supervised Training**
  - Architecture
  - Loss function
- **Neural Networks for Vision: Convolutional & Tiled**
- **Unsupervised Training of Neural Networks**
- **Extensions:**
  - semi-supervised / multi-task / multi-modal
- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models
- **Large-Scale Learning with Deep Neural Nets**

# Linear Classifier: SVM

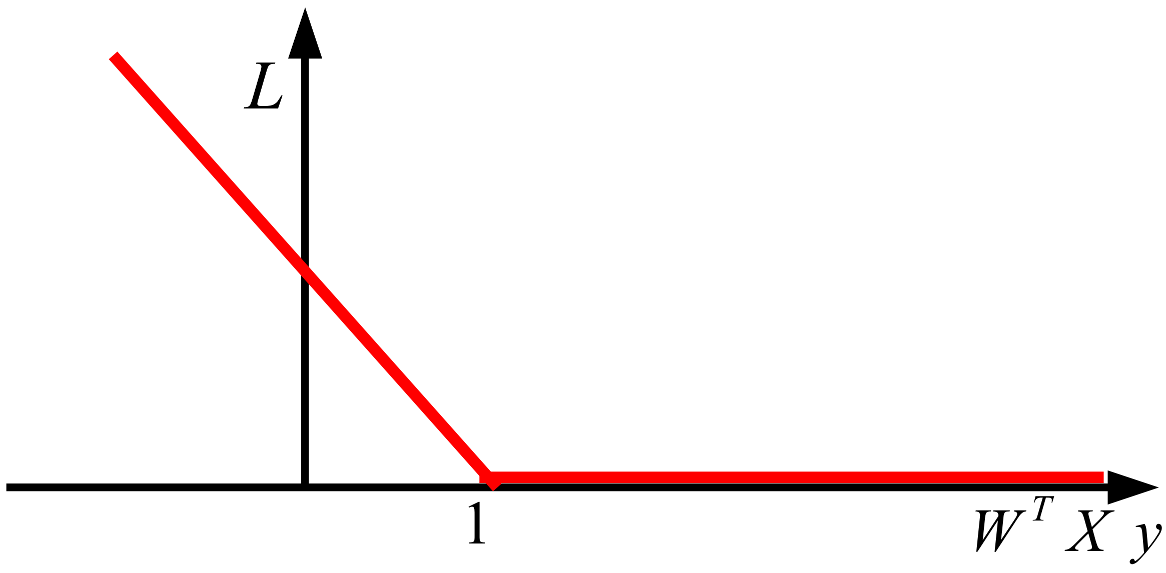
Input: $X \in R^D$

Binary label: $y$

Parameters: $W \in R^D$

Output prediction: $W^T X$

Loss: $L = \frac{1}{2} \| W \|^2 + \lambda \, max[0, 1 - W^T X \, y]$




Hinge Loss

# Linear Classifier: Logistic Regression

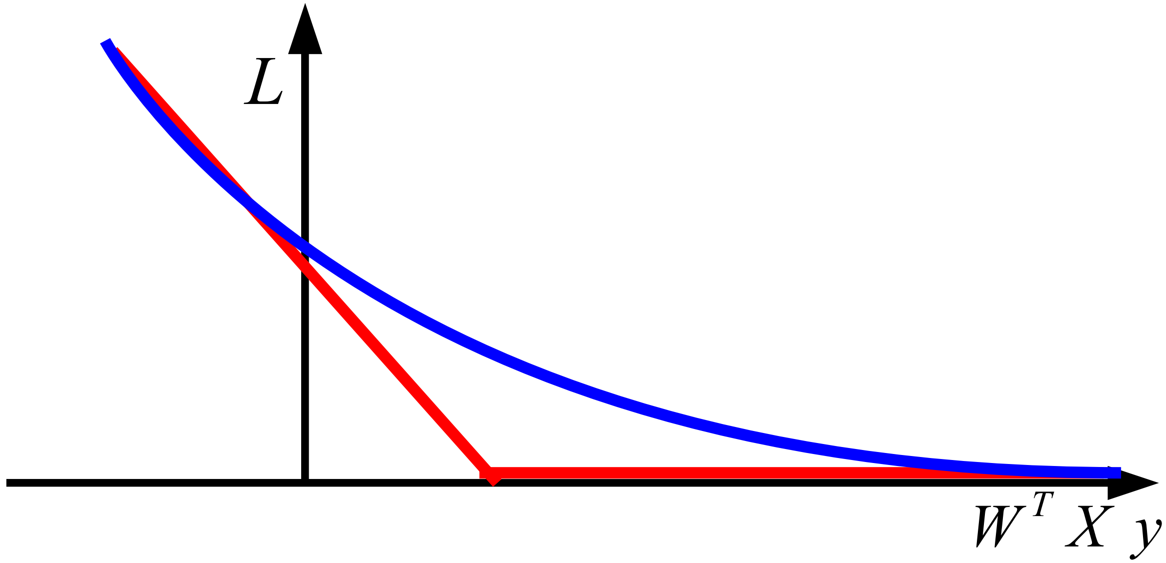
Input: $X \in R^D$

Binary label: $y$

Parameters: $W \in R^D$

Output prediction: $W^T X$

Loss: $L = \frac{1}{2} \|W\|^2 + \lambda \log(1 + \exp(-W^T X y))$




Log Loss



Ranzato

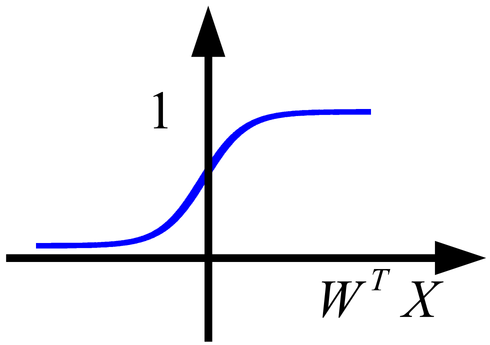
# Logistic Regression: Probabilistic Interpretation

Input: $X \in R^D$

Binary label: $y$

Parameters: $W \in R^D$

Output prediction: $p(y=1|X) = \frac{1}{1+e^{-W^T X}}$




Loss: $L = -\log(p(y|X))$

**Q:** What is the gradient of $L$ w.r.t. $W$?

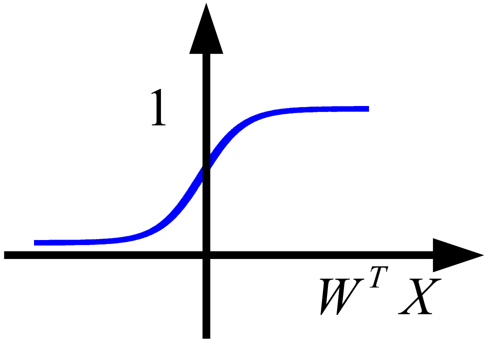
# Logistic Regression: Probabilistic Interpretation

Input: $X \in R^D$

Binary label: $y$

Parameters: $W \in R^D$

Output prediction: $p(y=1|X) = \frac{1}{1+e^{-W^T X}}$




Loss: $L = \log(1+\exp(-W^T X y))$

**Q:** What is the gradient of $L$ w.r.t. $W$?

Simple Functions

$\sin(x)$

$\log(x)$

$\cos(x)$

$x^3$

$\exp(x)$

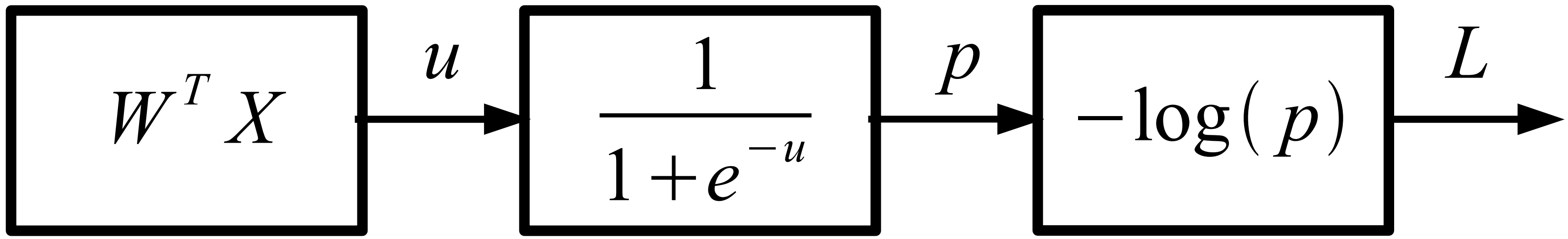
Complicated Function

$$-\log\left(\frac{1}{1+e^{-W^T X}}\right)$$

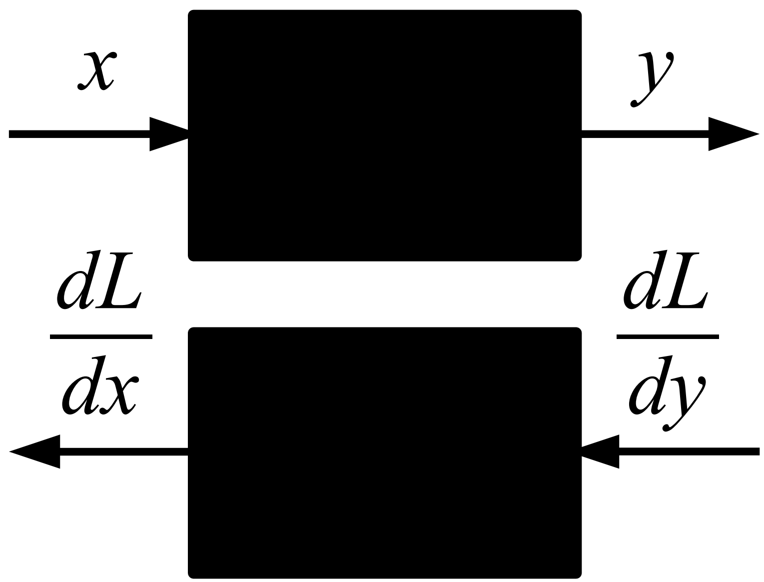
# Logistic Regression: Computing Loss

Complicated Function

$$-\log\left(\frac{1}{1+e^{-W^T X}}\right)$$



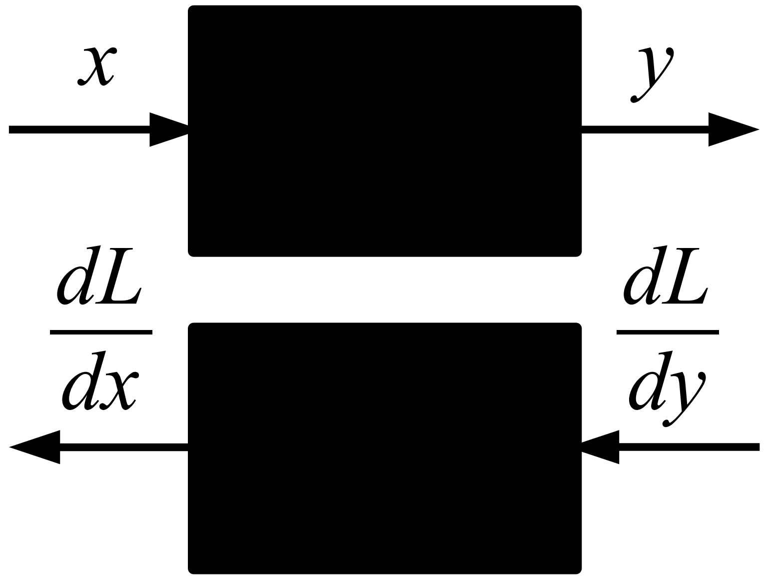
# Chain Rule




Given $y(x)$ and $dL/dy$,

What is $dL/dx$ ?

# Chain Rule

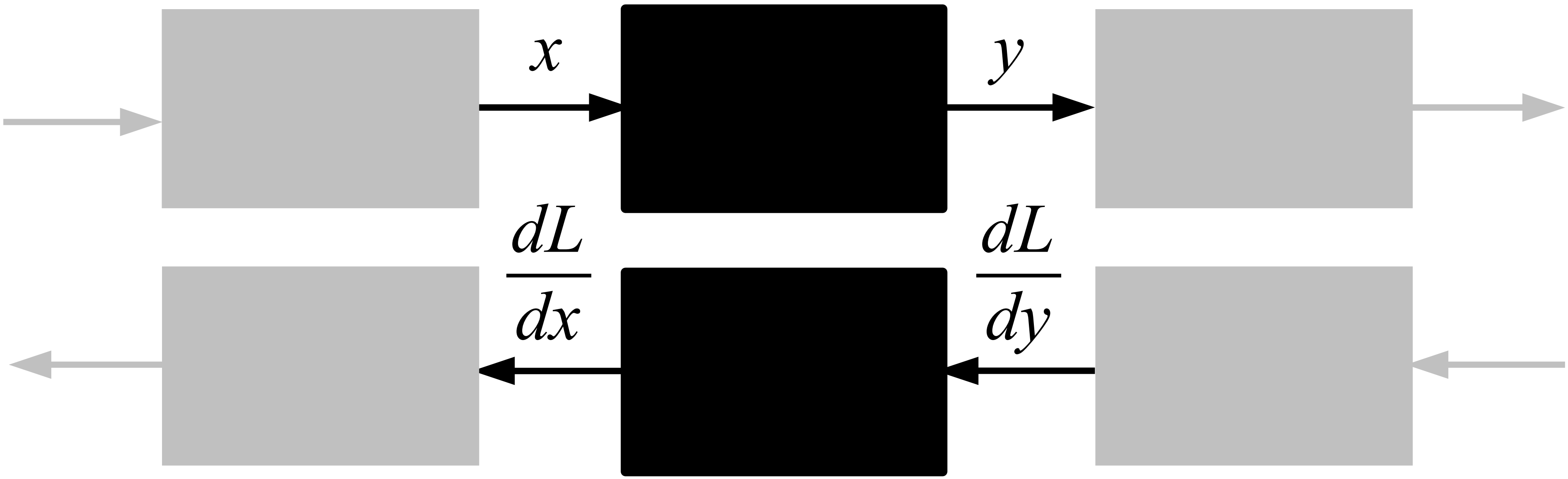



Given $y(x)$ and $dL/dy$,

What is $dL/dx$ ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

# Chain Rule



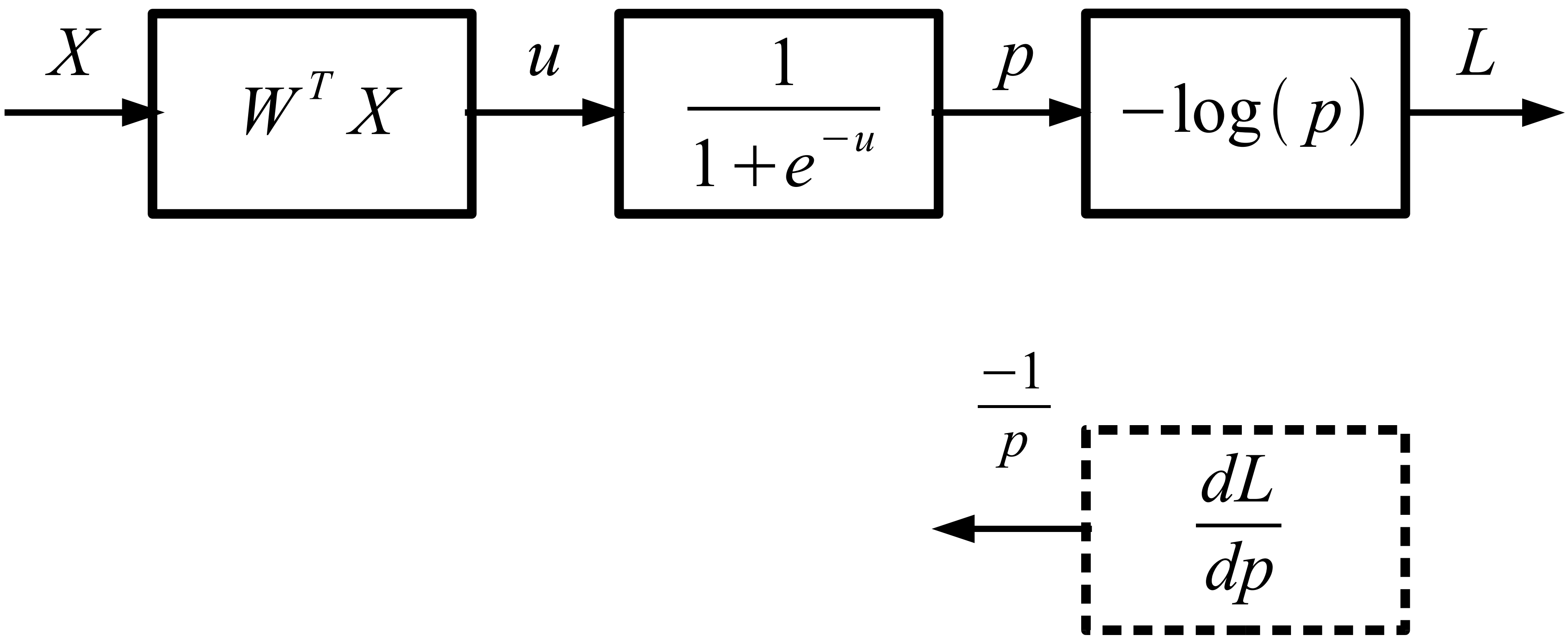

Given $y(x)$ and $dL/dy$,

What is $dL/dx$ ?

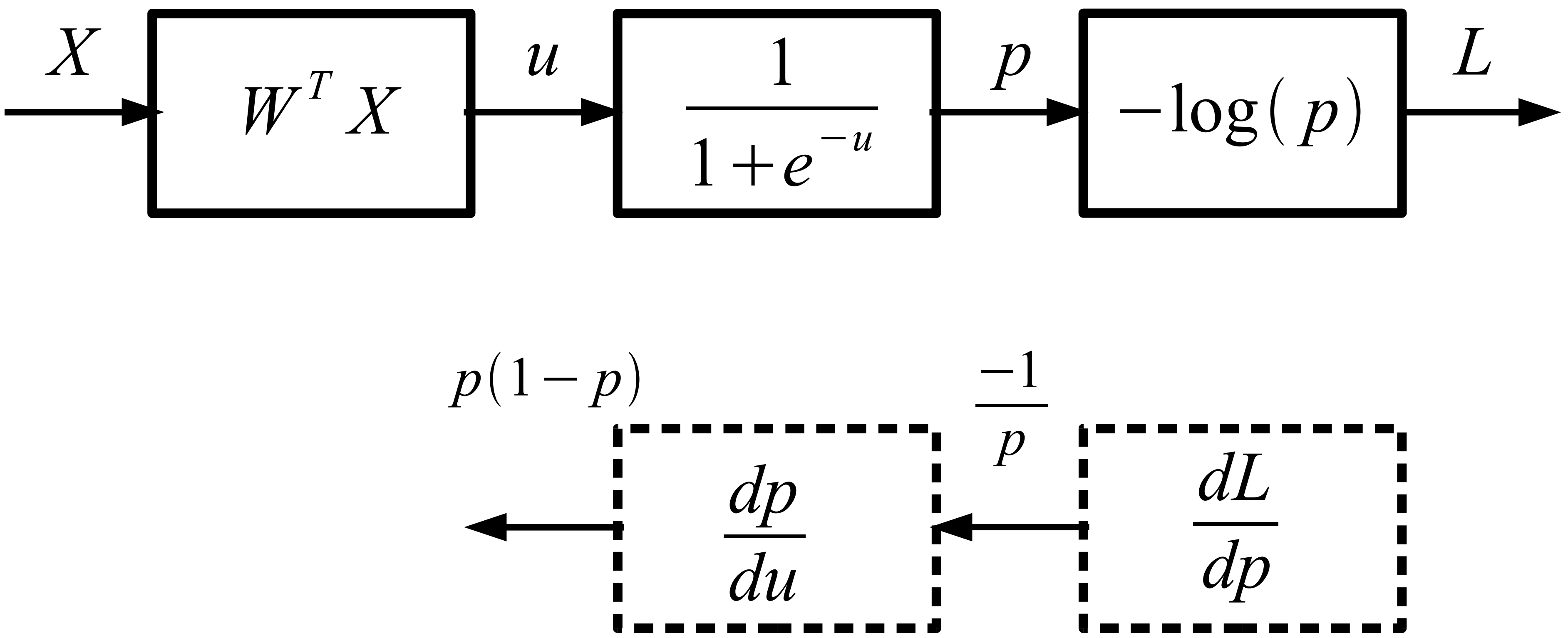
$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

**All needed information is local!**

# Logistic Regression: Computing Gradients



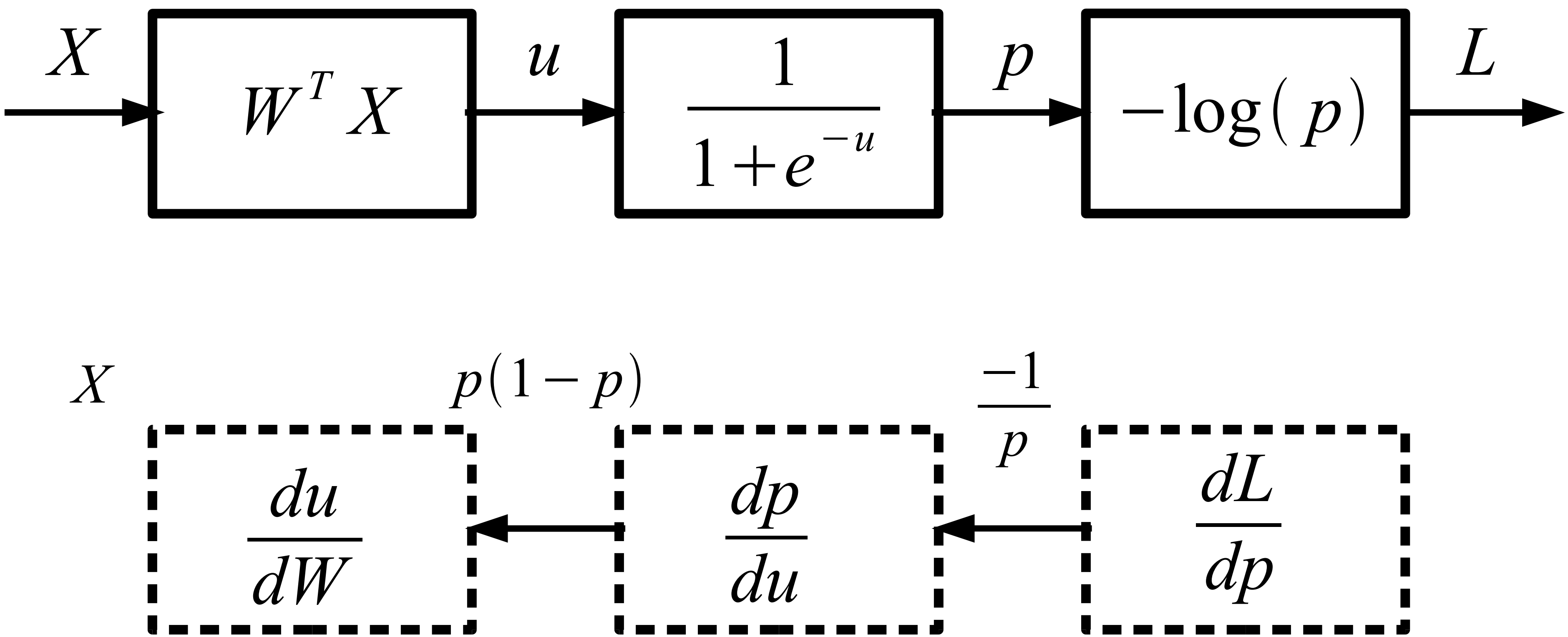
# Logistic Regression: Computing Gradients






Ranzato

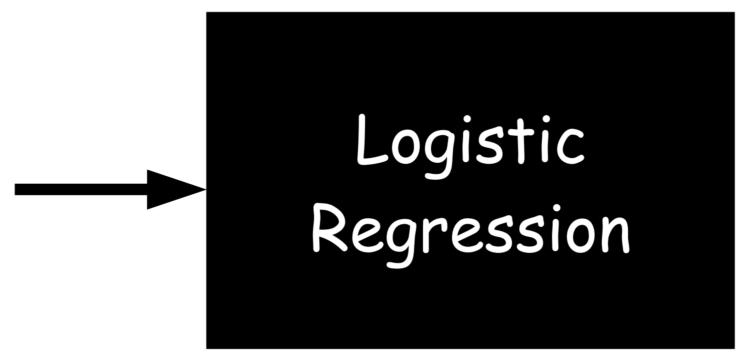
# Logistic Regression: Computing Gradients



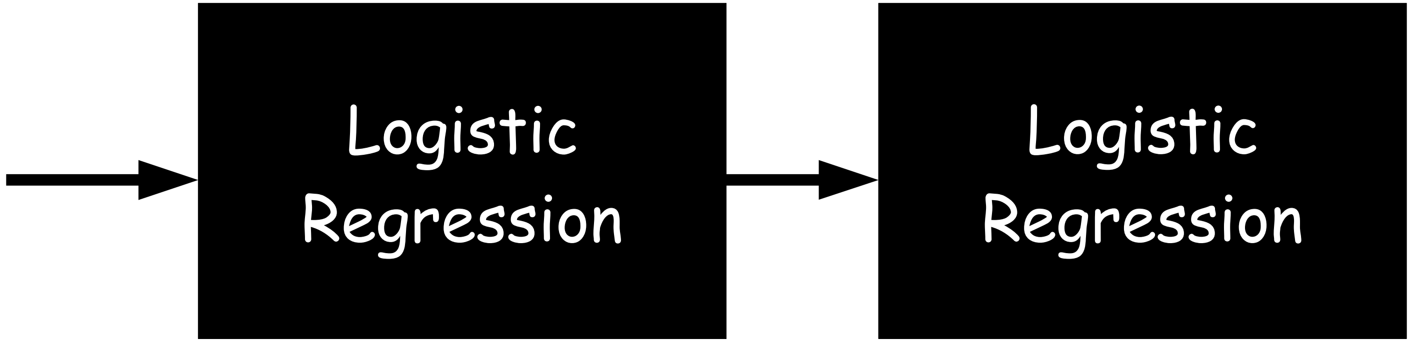

$$\frac{dL}{dW} = \frac{dL}{dp} \cdot \frac{dp}{du} \cdot \frac{du}{dW} = (p-1)X$$

# What Did We Learn?

- Logistic Regression
- How to compute gradients of complicated functions



# Neural Network



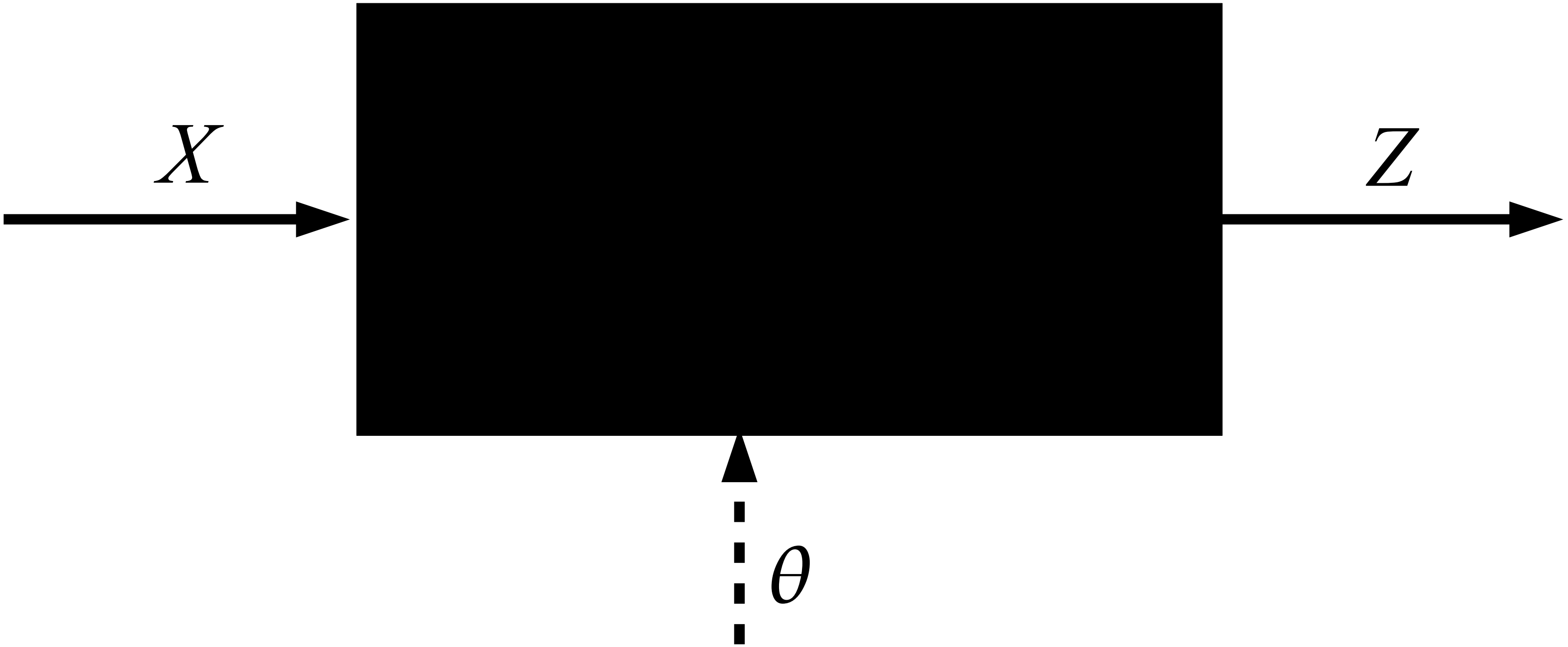
# Neural Network

- A **neural net** can be thought of as a stack of logistic regression classifiers. Each input is the output of the previous layer.


**NOTE:** intermediate units can be thought of as linear classifiers trained with implicit target values.

# Key Computations: F-Prop / B-Prop

## F-PROP



# Key Computations: F-Prop / B-Prop

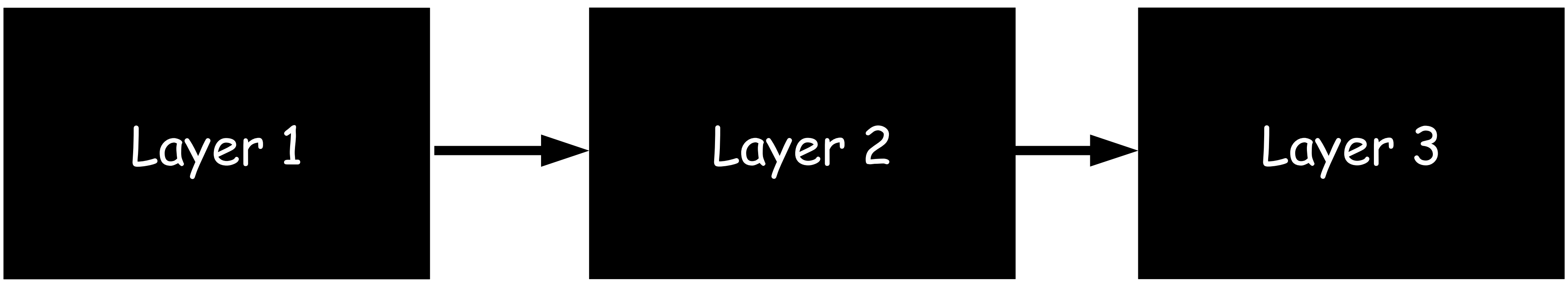
## B-PROP

$$\frac{\partial L}{\partial X} \longleftarrow \left\{ \frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial \theta} \right\} \longleftarrow \frac{\partial L}{\partial Z}$$

$$\frac{\partial L}{\partial \theta}$$

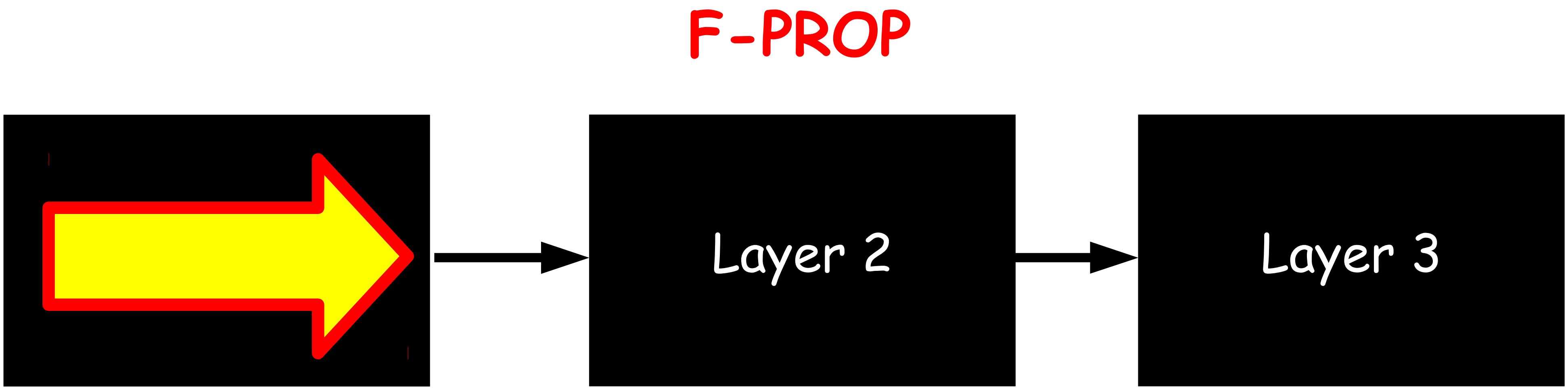
# Neural Net: Training

A) Compute loss on small mini-batch



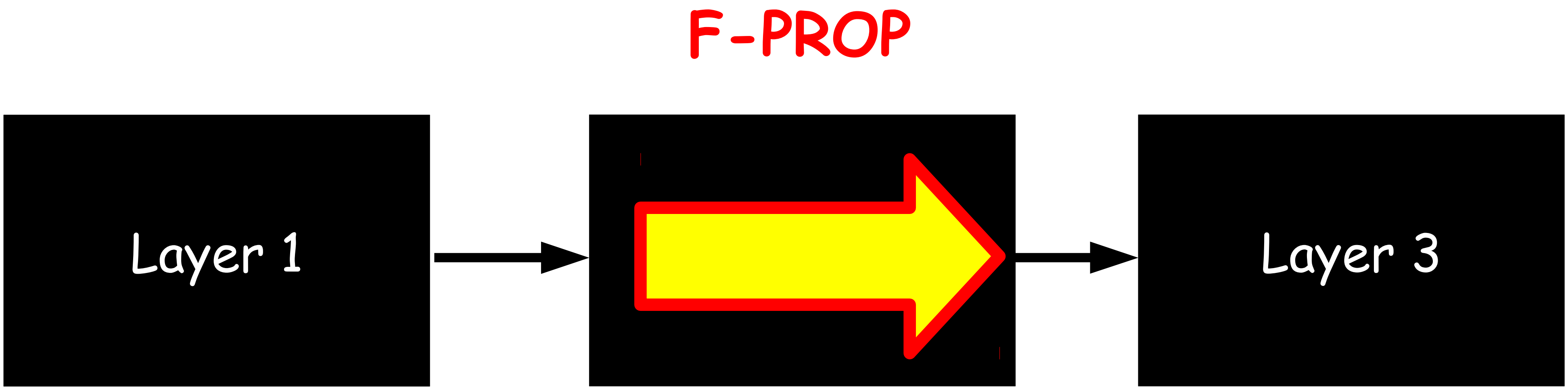
# Neural Net: Training

A) Compute loss on small mini-batch



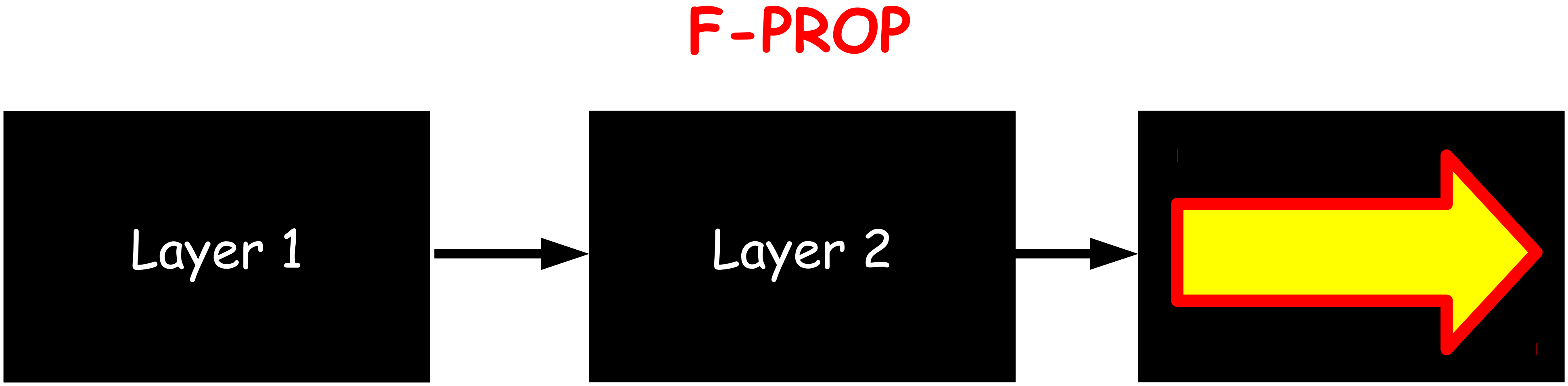
# Neural Net: Training

A) Compute loss on small mini-batch



# Neural Net: Training

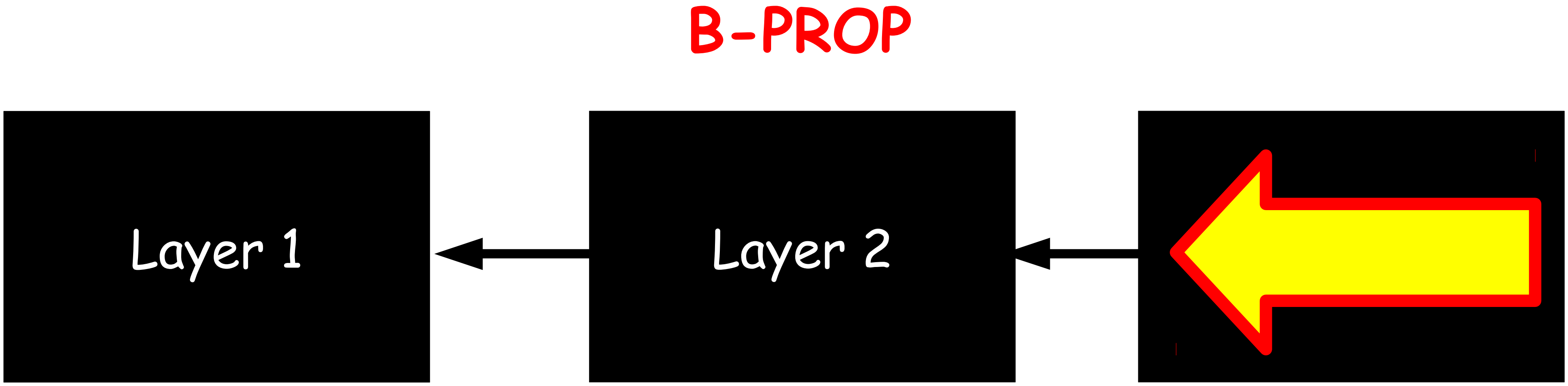
A) Compute loss on small mini-batch



# Neural Net: Training

A) Compute loss on small mini-batch

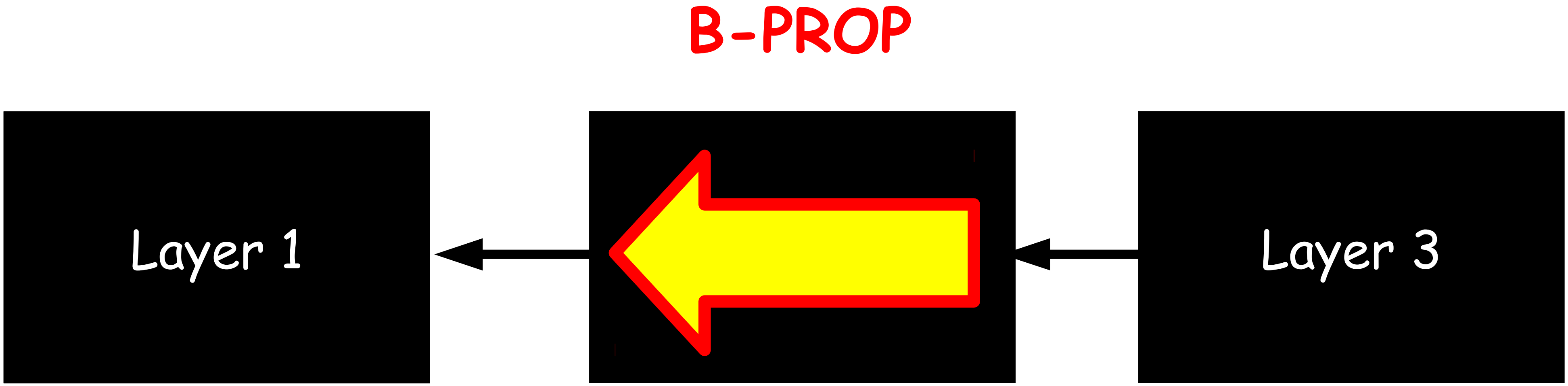
B) Compute gradient w.r.t. parameters



# Neural Net: Training

A) Compute loss on small mini-batch

B) Compute gradient w.r.t. parameters

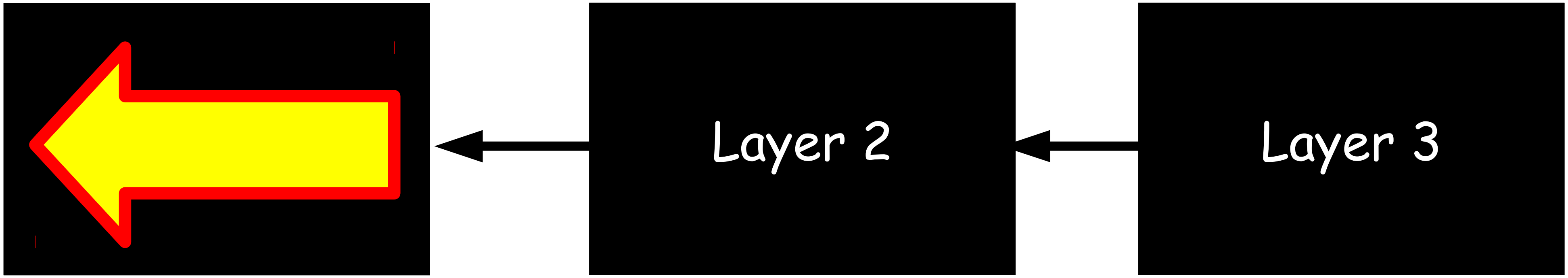


# Neural Net: Training

A) Compute loss on small mini-batch

B) Compute gradient w.r.t. parameters

**B-PROP**

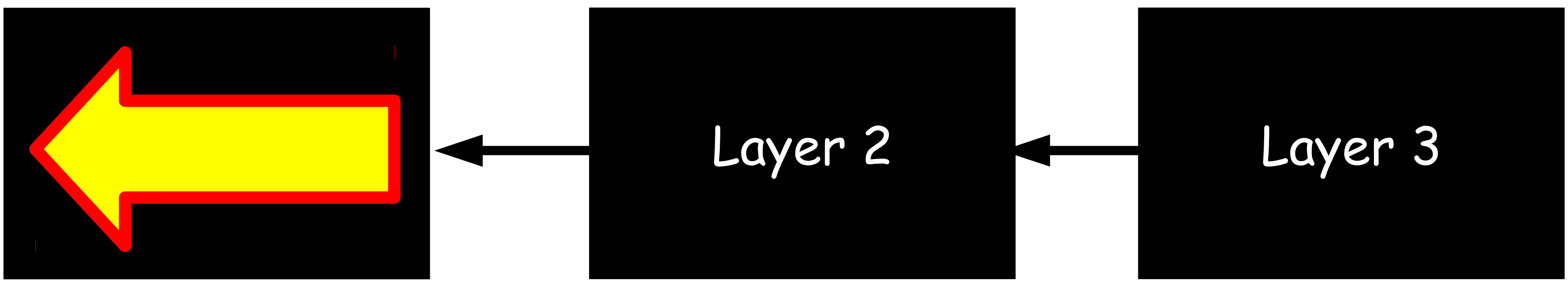


# Neural Net: Training

A) Compute loss on small mini-batch

B) Compute gradient w.r.t. parameters

C) Use gradient to update parameters $\theta \leftarrow \theta - \eta \frac{dL}{d\,\theta}$



# NEURAL NET: ARCHITECTURE


$$h_{j+1} = \sigma\left(W_{j+1}^T h_j + b_{j+1}\right)$$

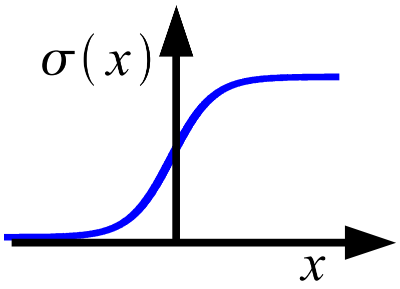
$$W_j \in R^{M \times N}, \; b_j \in R^N$$

$$h_j \in R^M, \; h_{j+1} \in R^N$$

# NEURAL NET: ARCHITECTURE


$$h_{j+1} = \sigma(W_{j+1}^T h_j + b_{j+1})$$

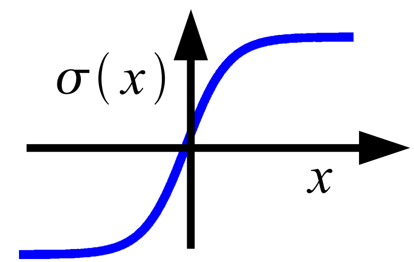
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



# NEURAL NET: ARCHITECTURE

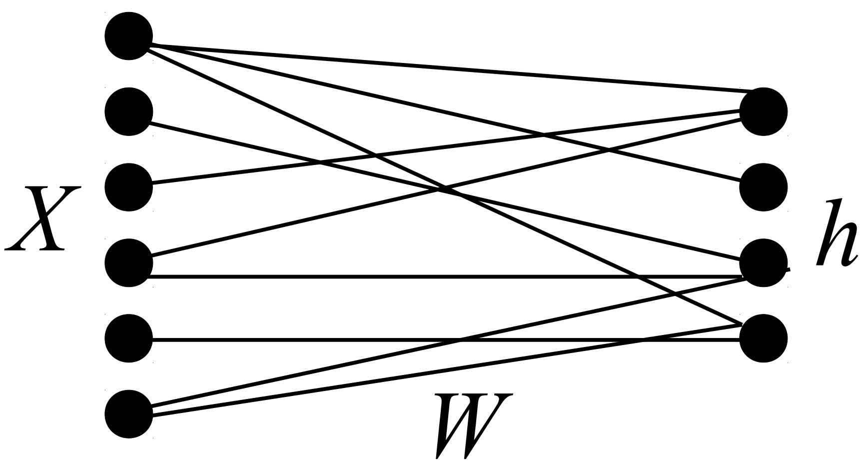

$$h_{j+1} = \sigma(W^T_{j+1} h_j + b_{j+1})$$

$$\sigma(x) = \tanh(x)$$



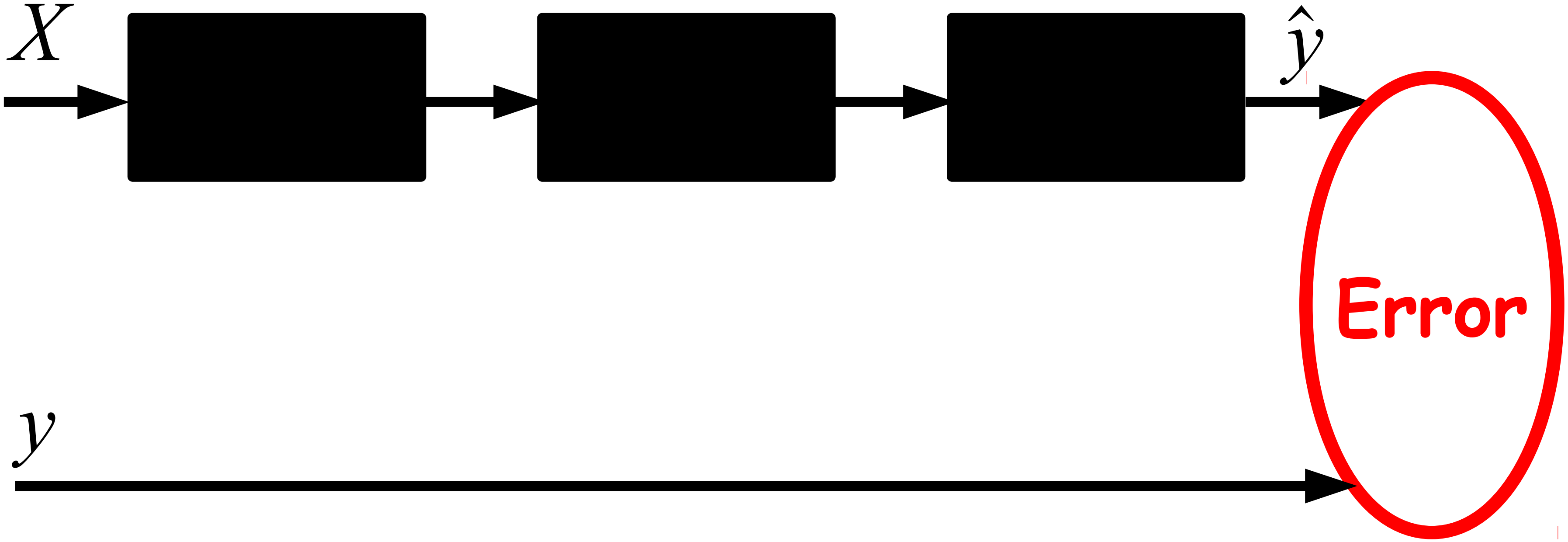
# Graphical Notations


**is equivalent to**




$h_k$ is called feature, hidden unit, neuron or code unit

# MOST COMMON ARCHITECTURE

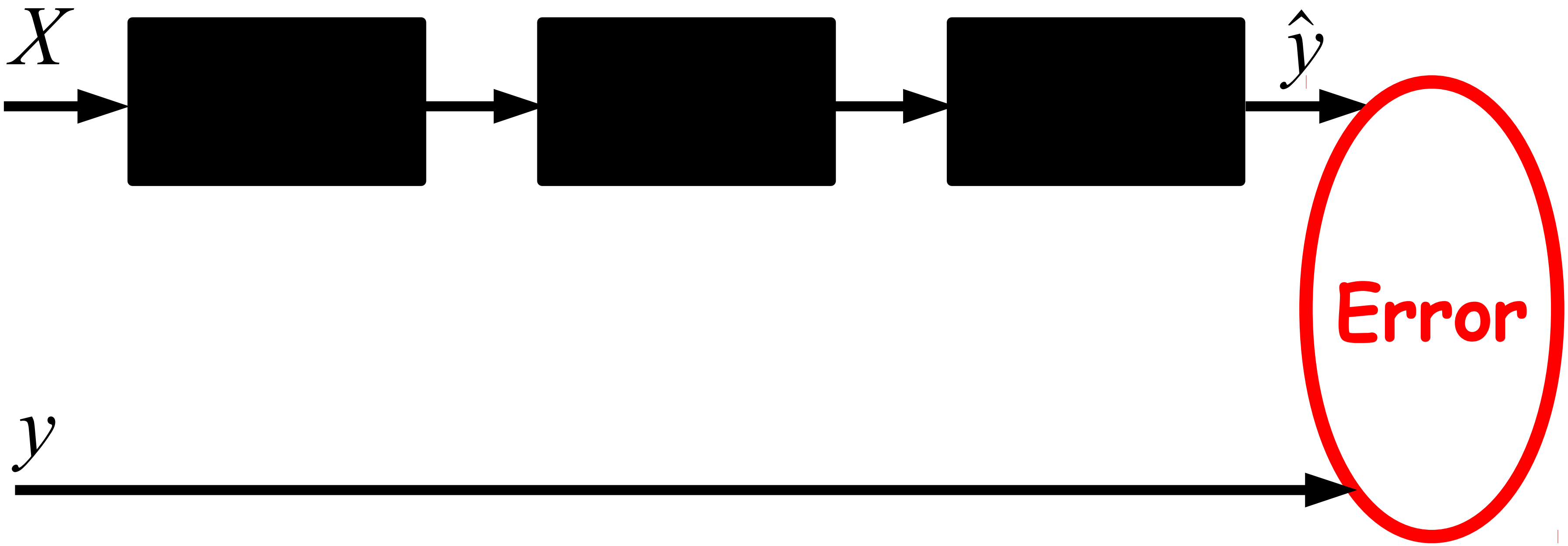



**NOTE:** Multi-layer neural nets with more than two layers are nowadays called **deep nets**!!



Ranzato

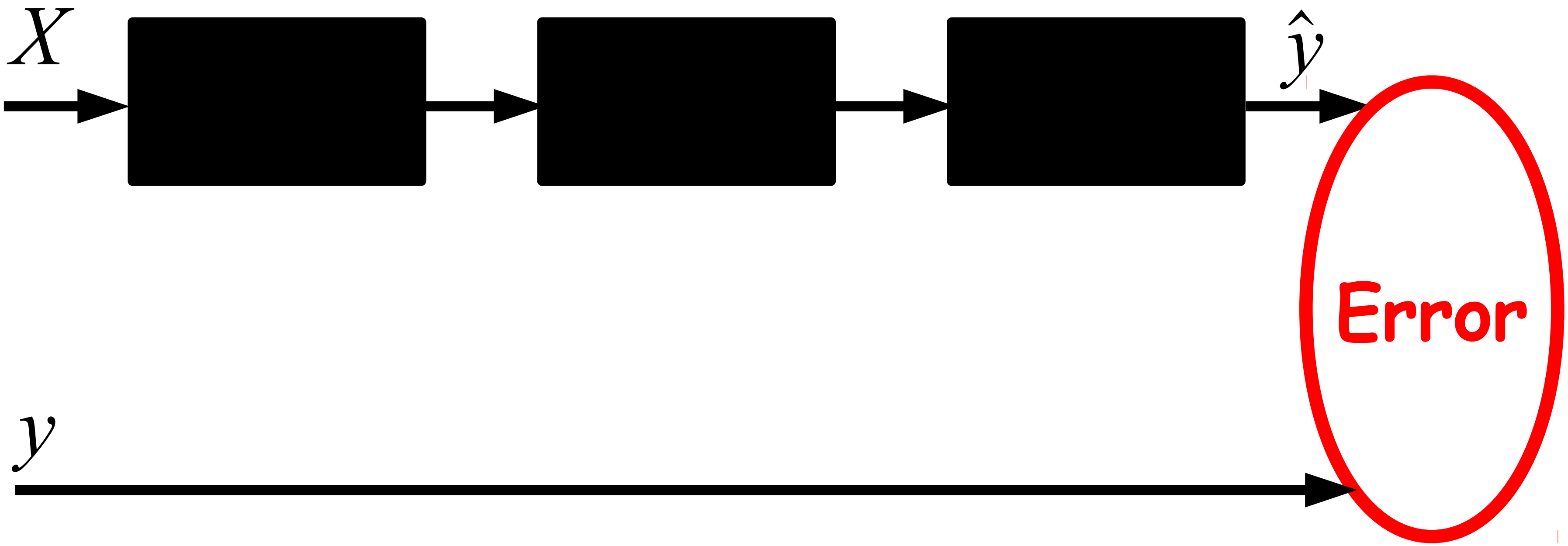
# MOST COMMON ARCHITECTURE




**NOTE:** Multi-layer neural nets with more than two layers are nowadays called **deep nets**!!

**NOTE:** User must specify number of layers, number of hidden units, type of layers and loss function.

# MOST COMMON LOSSES

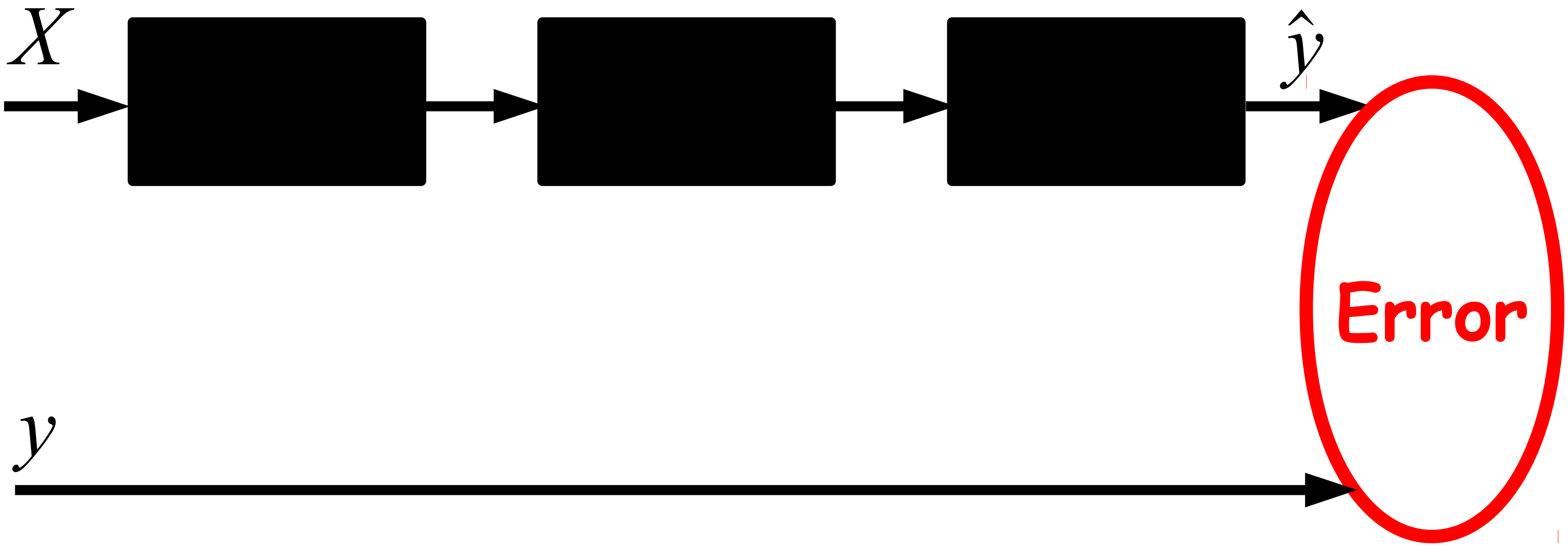



**Square Euclidean Distance (regression):**

$y, \hat{y} \in R^N$

$$L = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

# MOST COMMON LOSSES

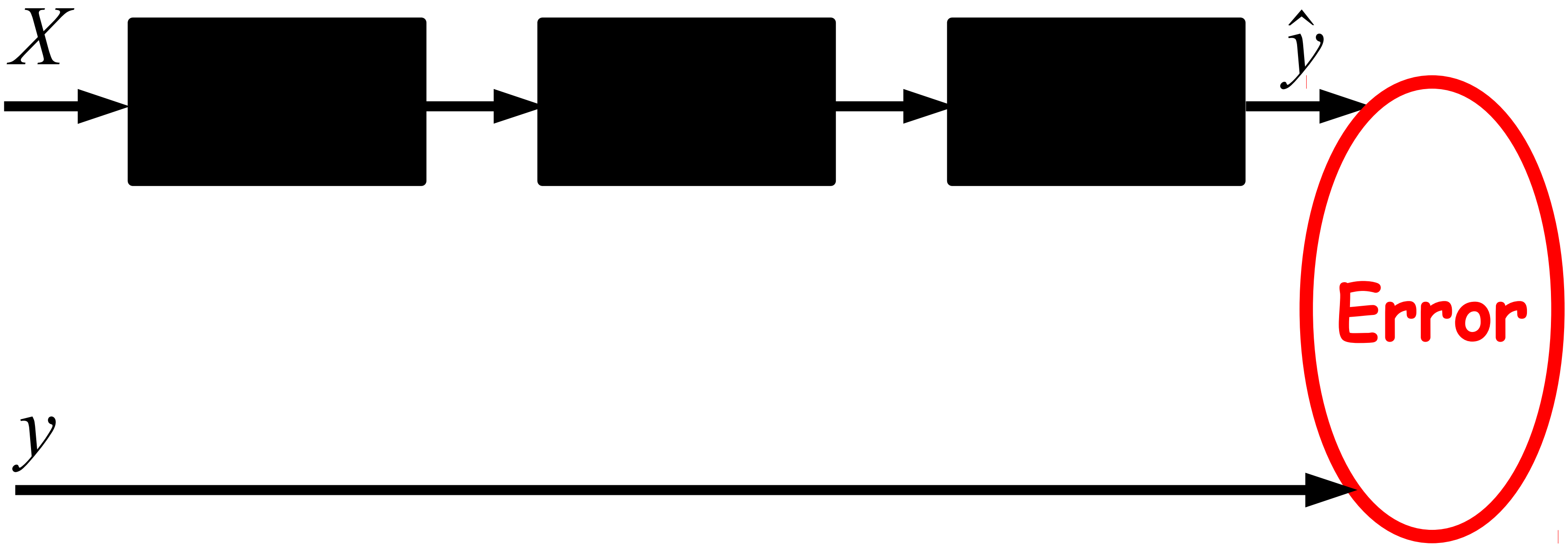



**Cross Entropy (classification):**

$$y\,,\hat{y} \in [0,1]^N,\ \sum_{i=1}^{N} y_i = 1,\ \sum_{i=1}^{N} \hat{y}_i = 1$$

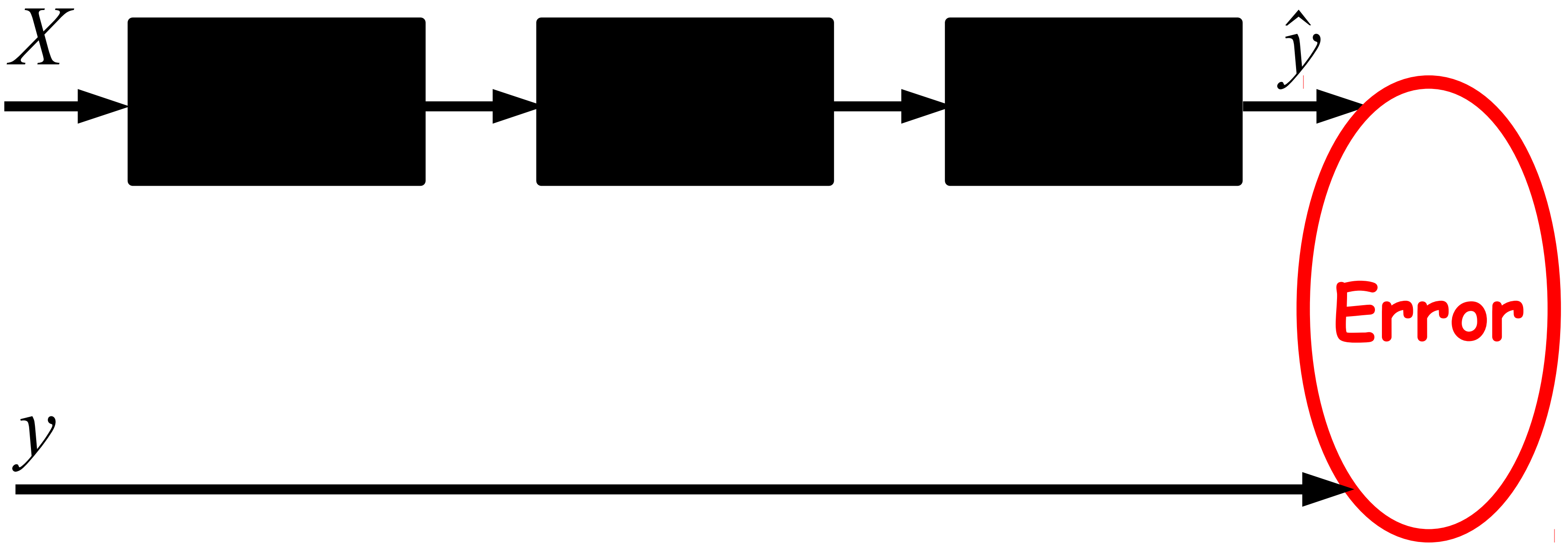
$$L = -\sum_{i=1}^{N} y_i \log \hat{y}_i$$

# MOST COMMON LOSSES




**NOTE:** User specifies loss based on the task.

# TRAINING




**Algorithm (SGD):**

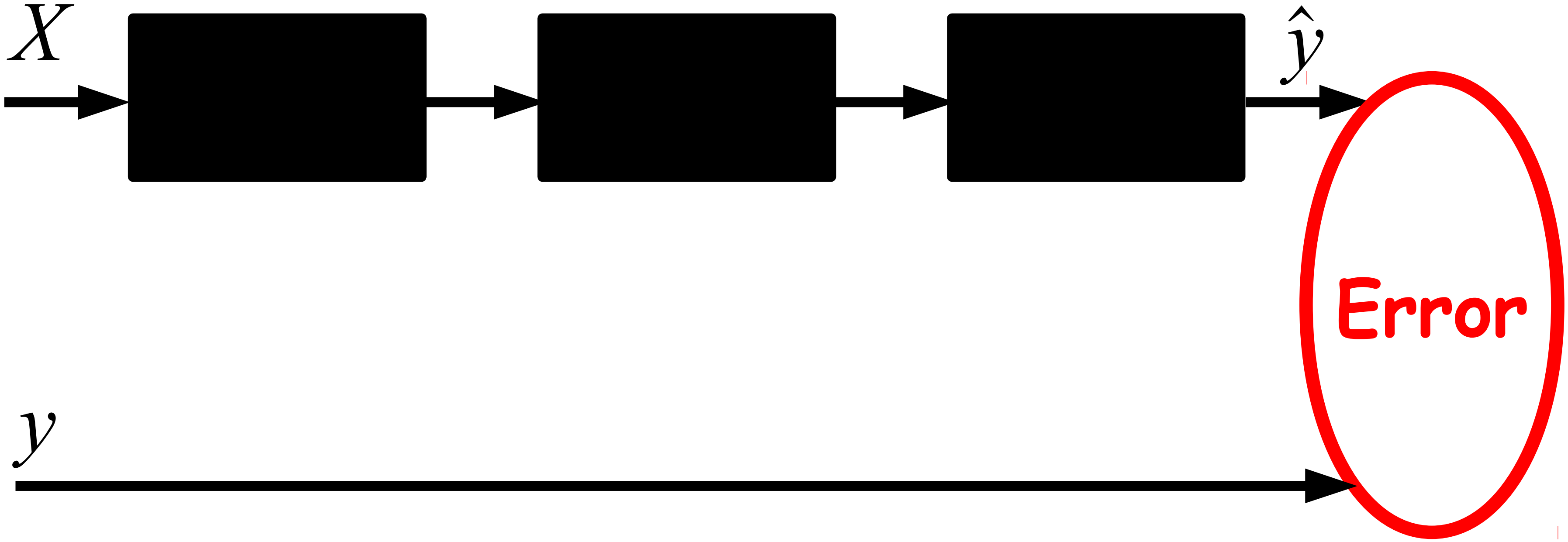
**Given a small mini-batch**

**- FPROP**

**- BPROP**

**- PARAMETER UPDATE** $$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}$$

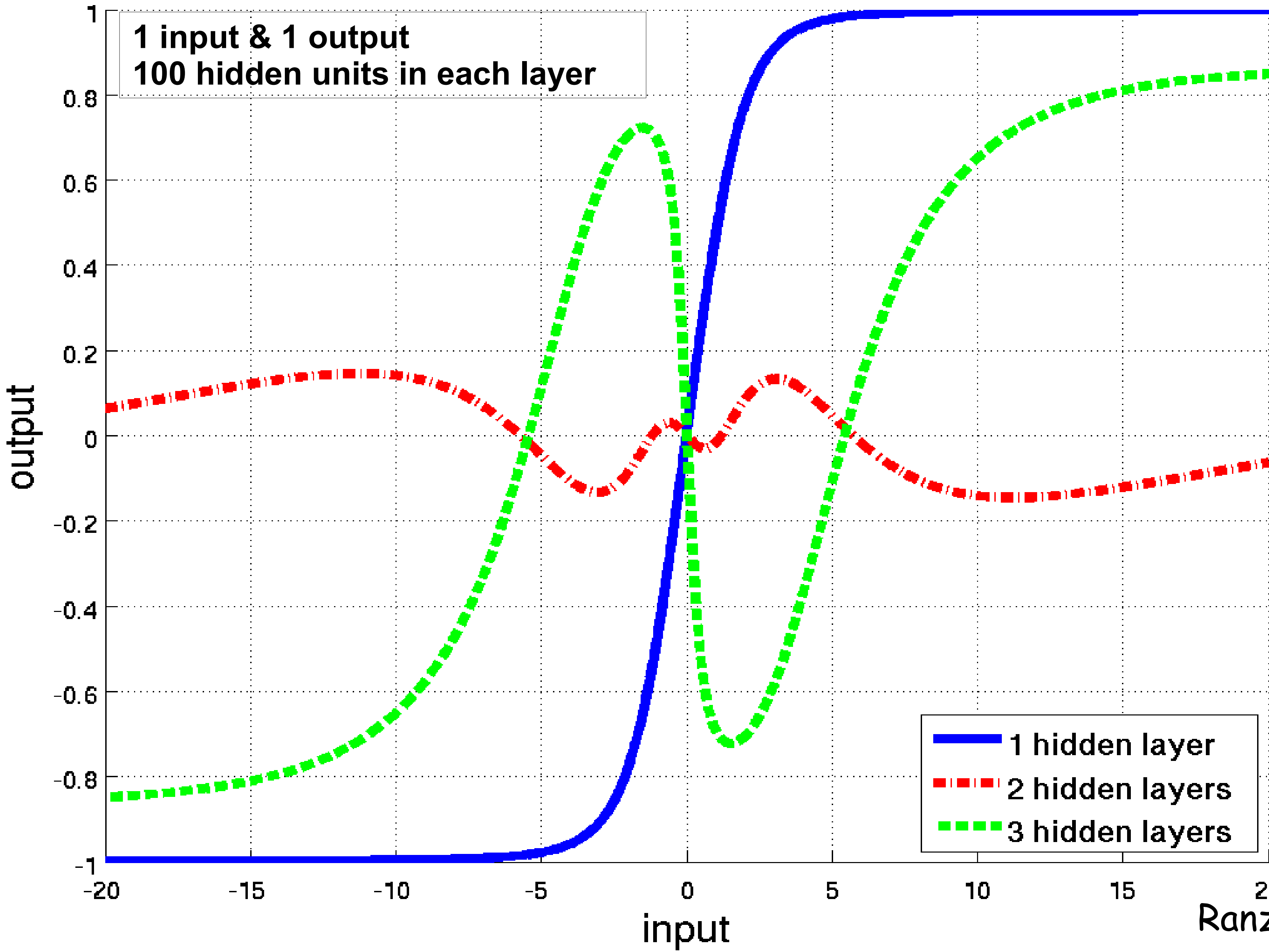
# TRAINING




## NOTES

- User chooses optimization algorithm.
- Computational cost of F-PROP & B-PROP is similar.

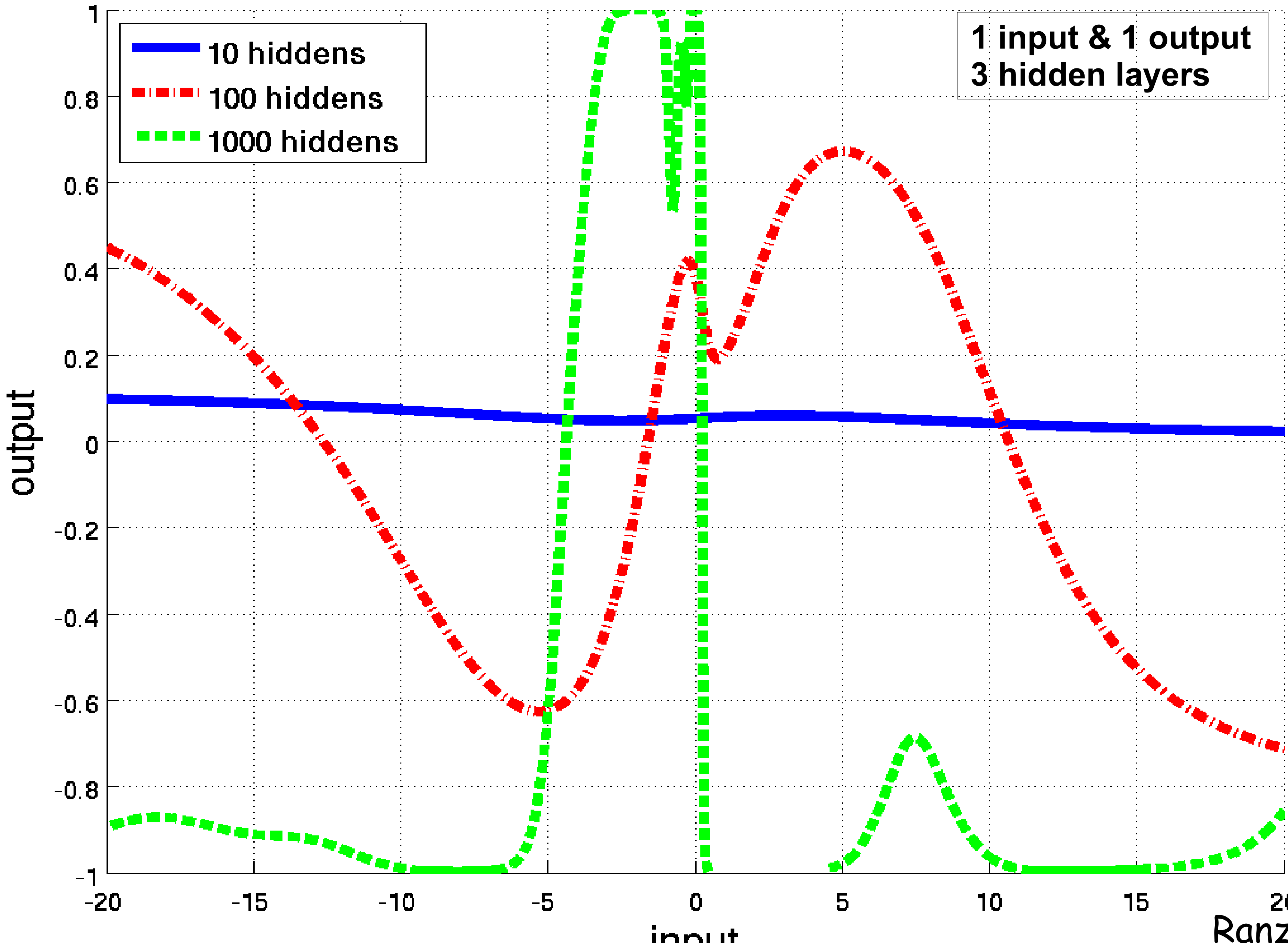
# TOY EXAMPLE: SYNTHETIC DATA






Ranzato

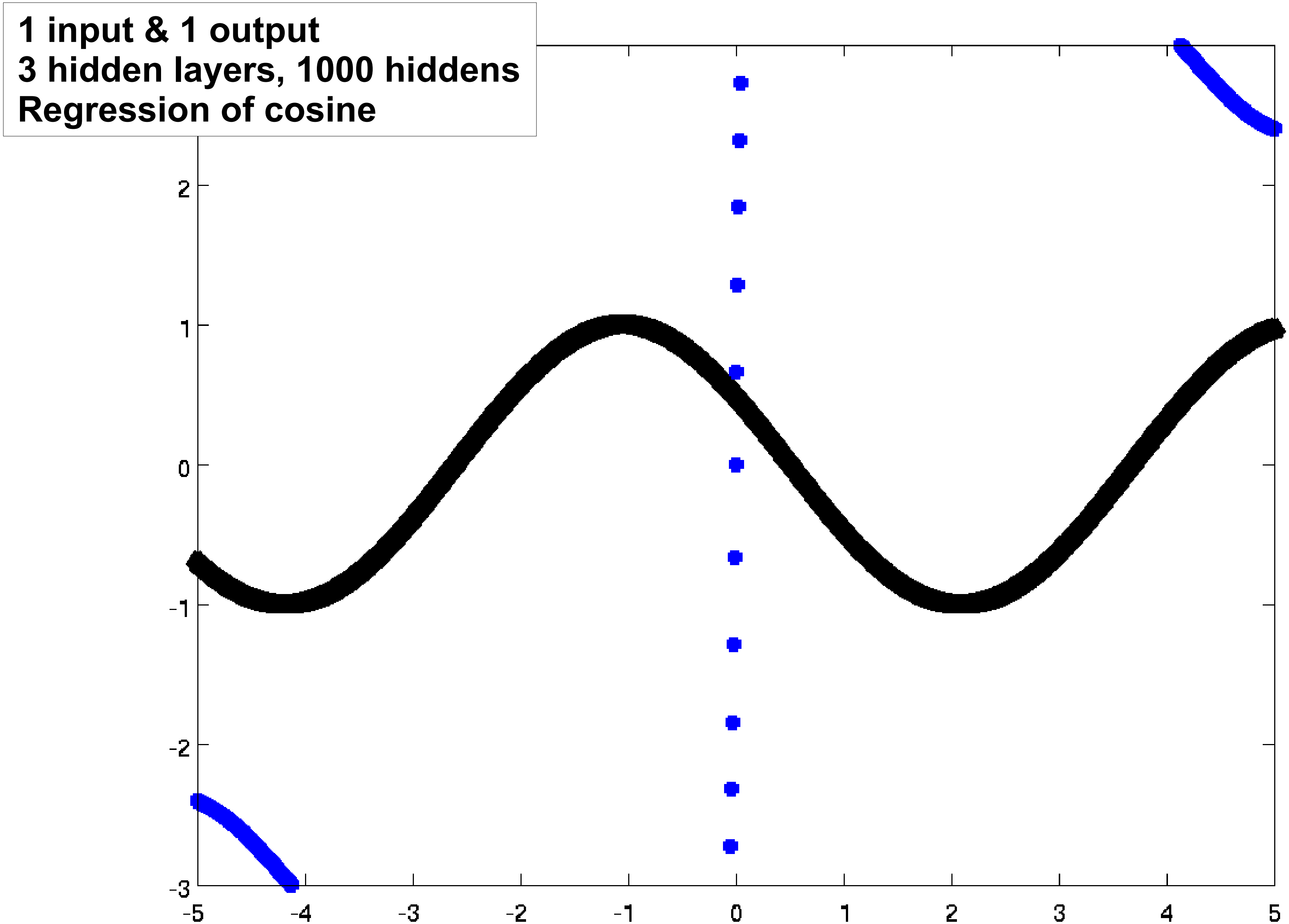
# TOY EXAMPLE: SYNTHETIC DATA






Ranzato

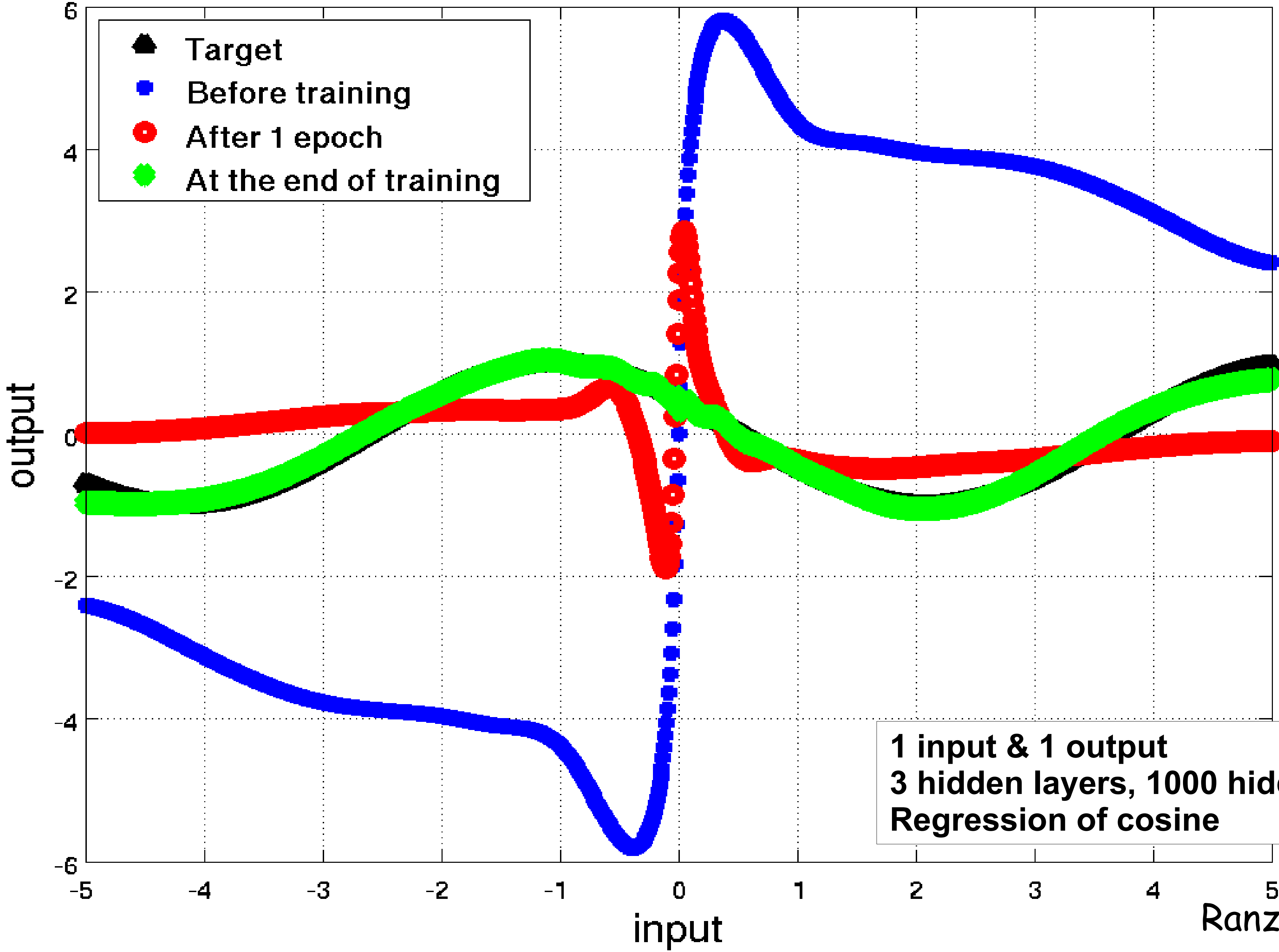
# TOY EXAMPLE: SYNTHETIC DATA






Ranzato

# TOY EXAMPLE: SYNTHETIC DATA




**1 input & 1 output**
**3 hidden layers, 1000 hiddens**
**Regression of cosine**

# TOY EXAMPLE: MNIST

Linear Classifier.........................................................12.0%

Boosted stumps..........................................................7.7%
Product of boosted stumps...................................1.3%
Boosted trees...........................................................1.5%
Stumps on Haar features.......................................0.9%

K-NN...........................................................................5.0%

SVM Gaussian kernel..............................................1.4%

2 layer nnet 800 hiddens......................................1.6%
4 layer nnet (pre-trained)......................................1.0%
Conv. Net (pre-trained)..........................................0.6%

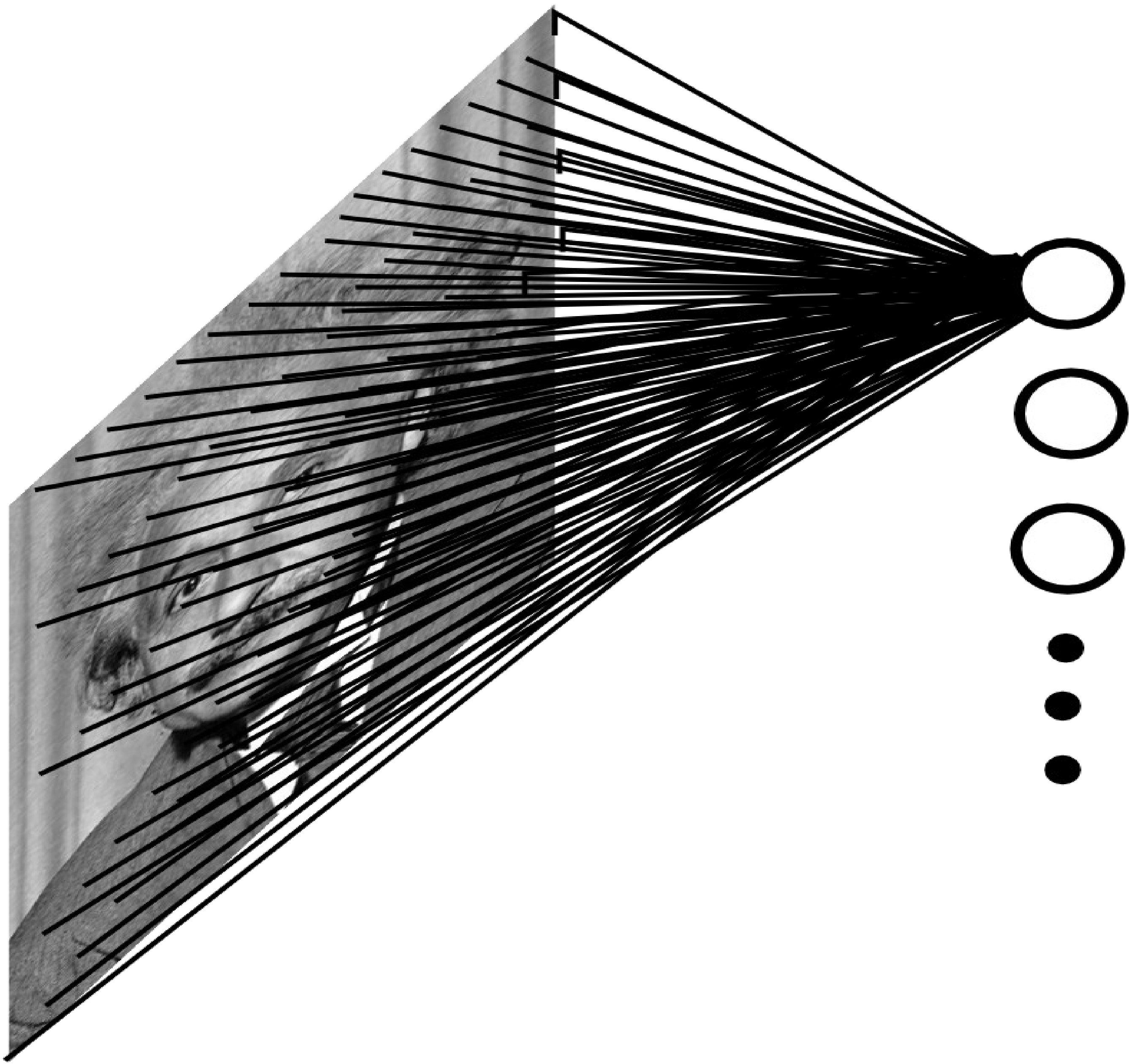


*http://yann.lecun.com/exdb/mnist/*

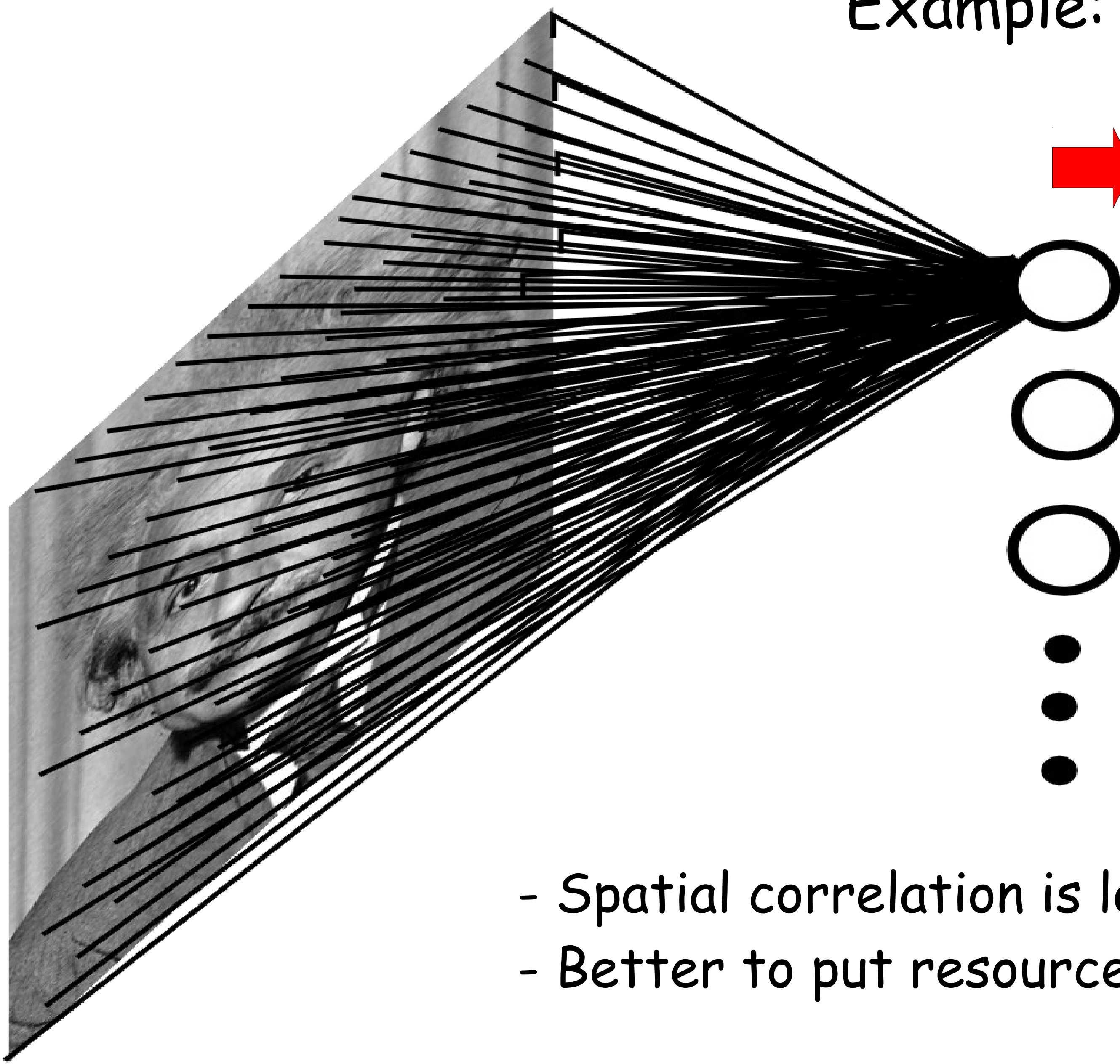
# Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function
- **Neural Networks for Vision: Convolutional & Tiled**
- Unsupervised Training of Neural Networks
- Extensions:
  - semi-supervised / multi-task / multi-modal
- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models
- Large-Scale Learning with Deep Neural Nets

# FULLY CONNECTED NEURAL NET



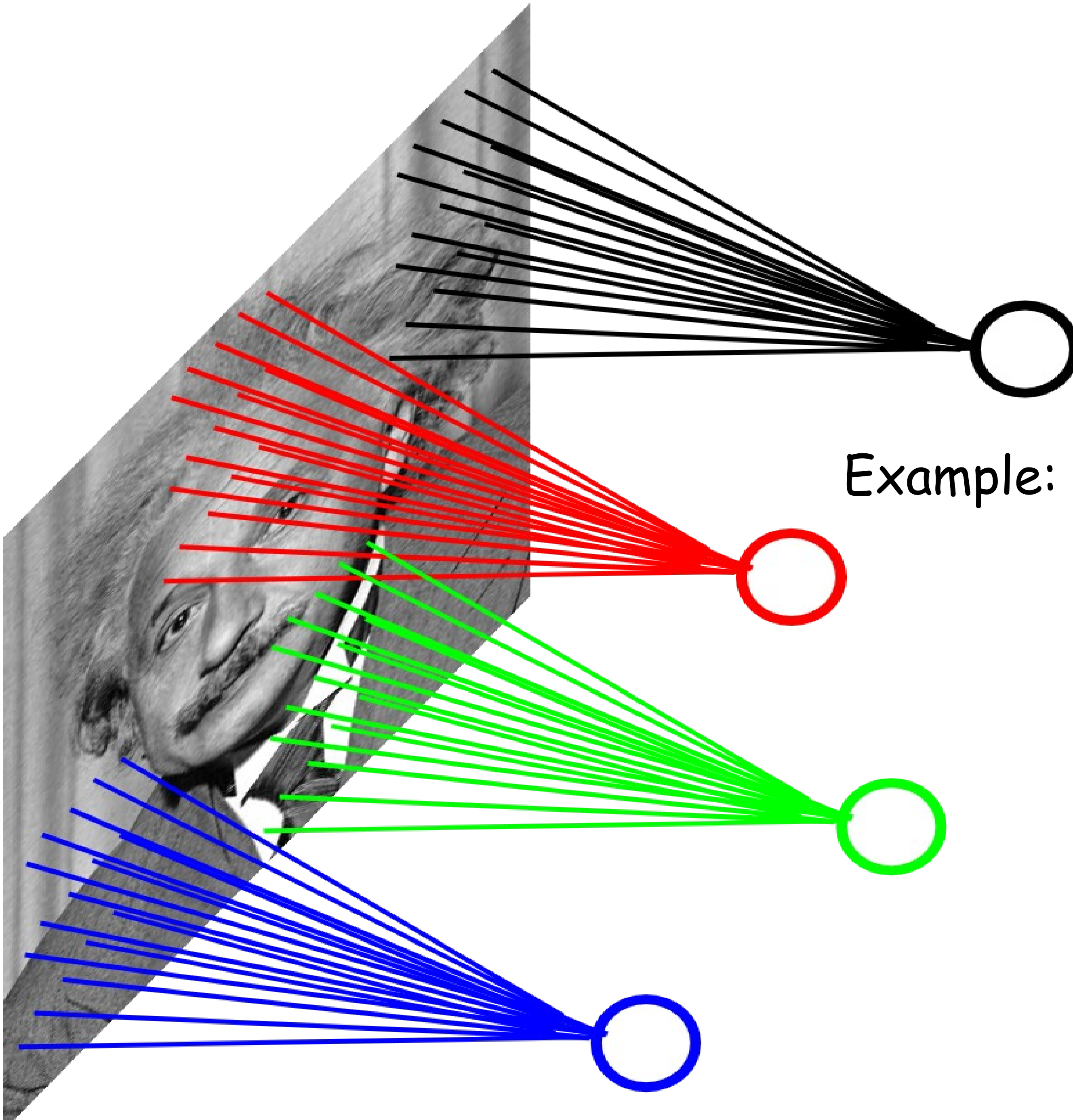
# FULLY CONNECTED NEURAL NET



Example: 1000x1000 image
1M hidden units
**1B parameters**!!!

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

# LOCALLY CONNECTED NEURAL NET

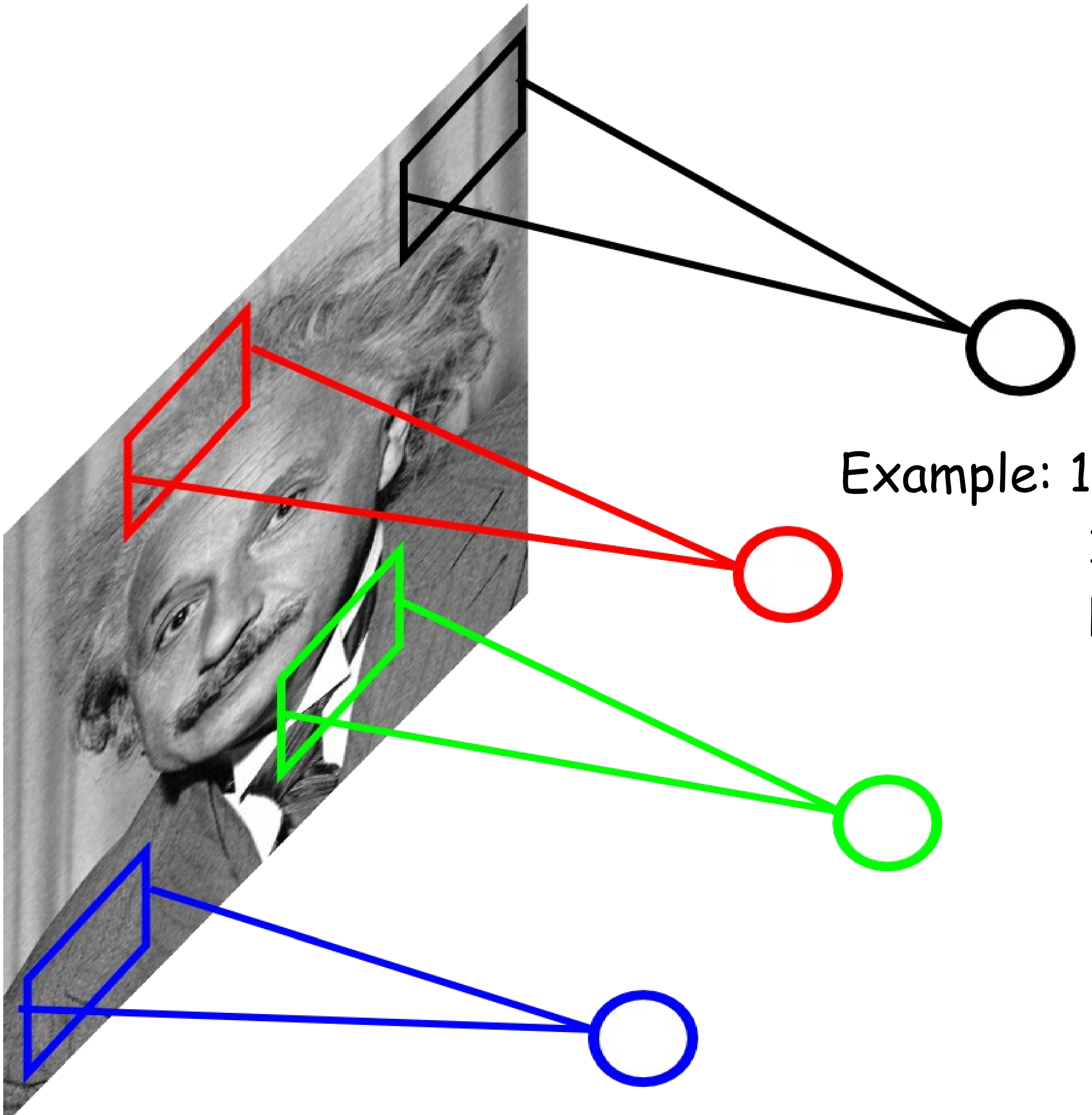


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters



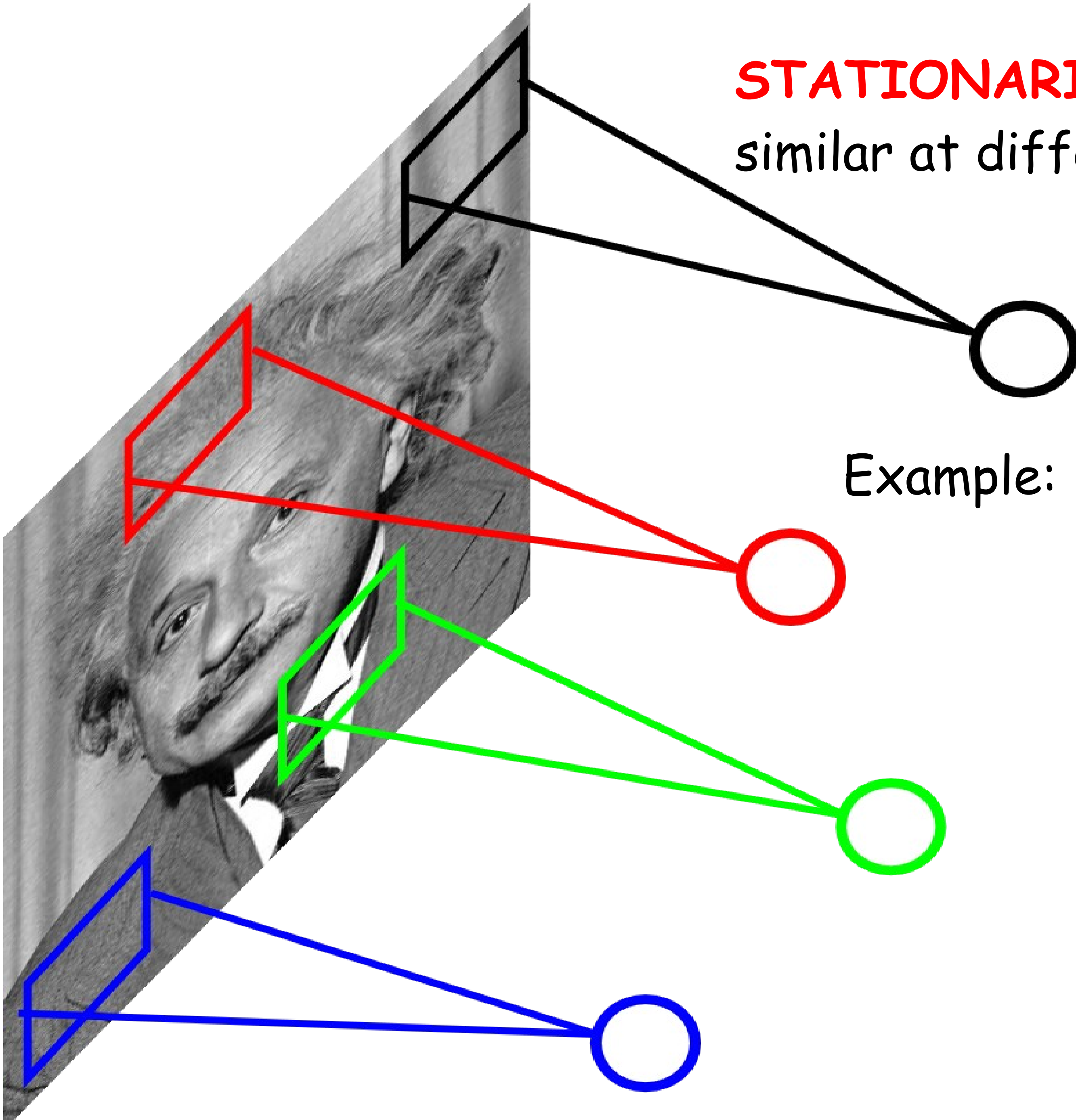
Ranzato

# LOCALLY CONNECTED NEURAL NET



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

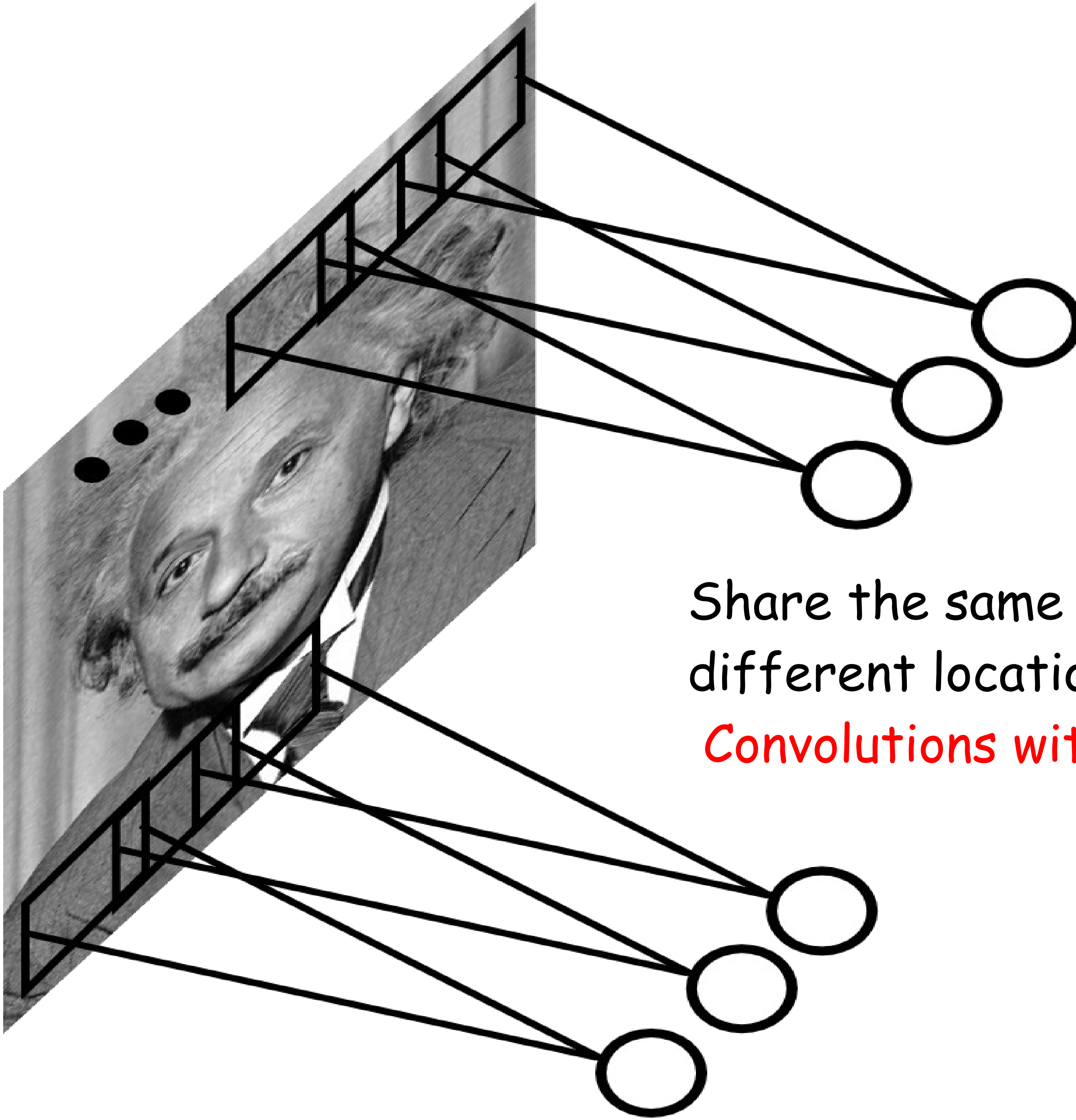
# LOCALLY CONNECTED NEURAL NET



STATIONARITY? Statistics is similar at different locations

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

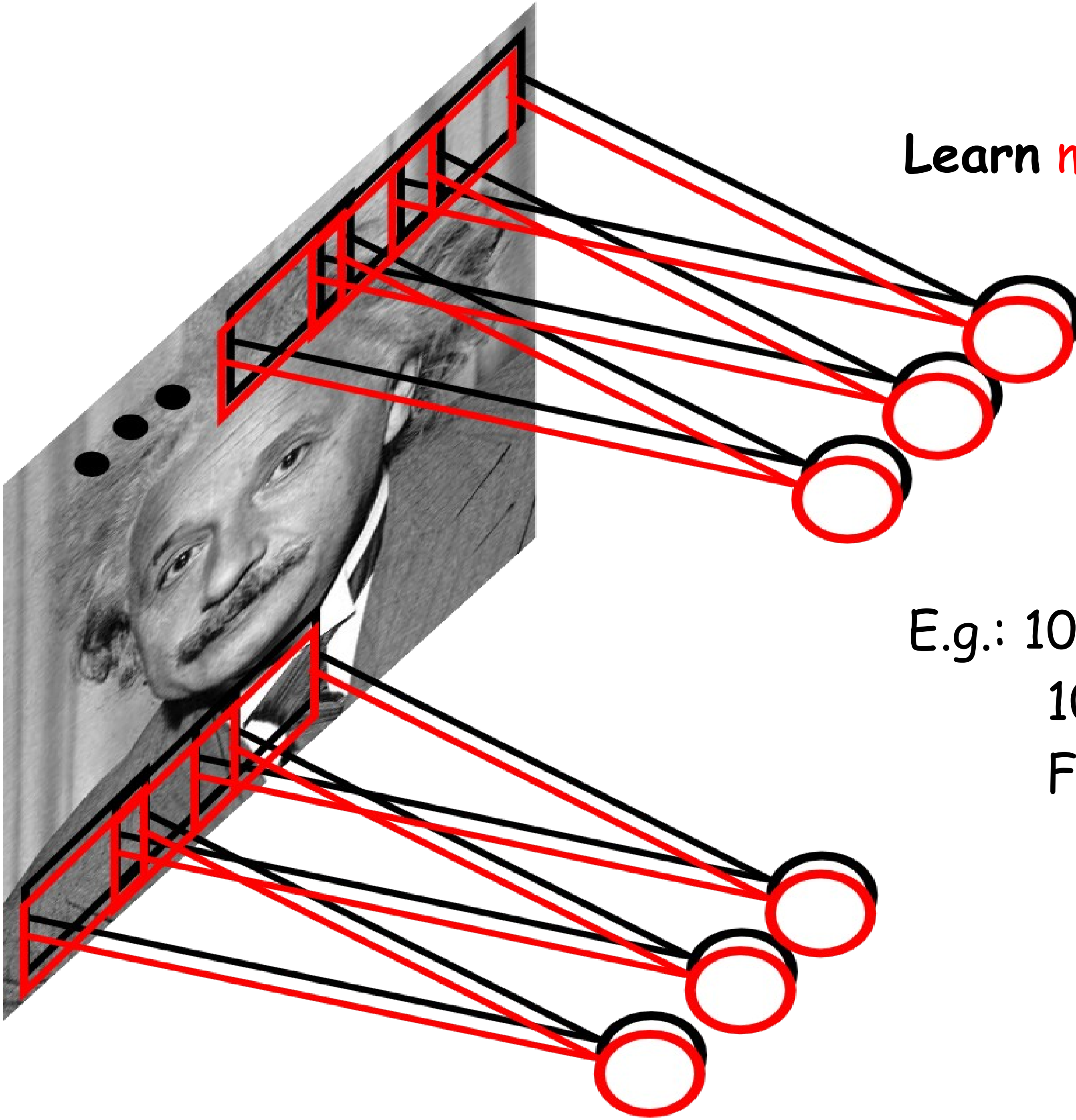
# CONVOLUTIONAL NET



Share the same parameters across different locations:

Convolutions with learned kernels

# CONVOLUTIONAL NET



Learn multiple filters.

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



Ranzato

# NEURAL NETS FOR VISION

A standard neural net applied to images:

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

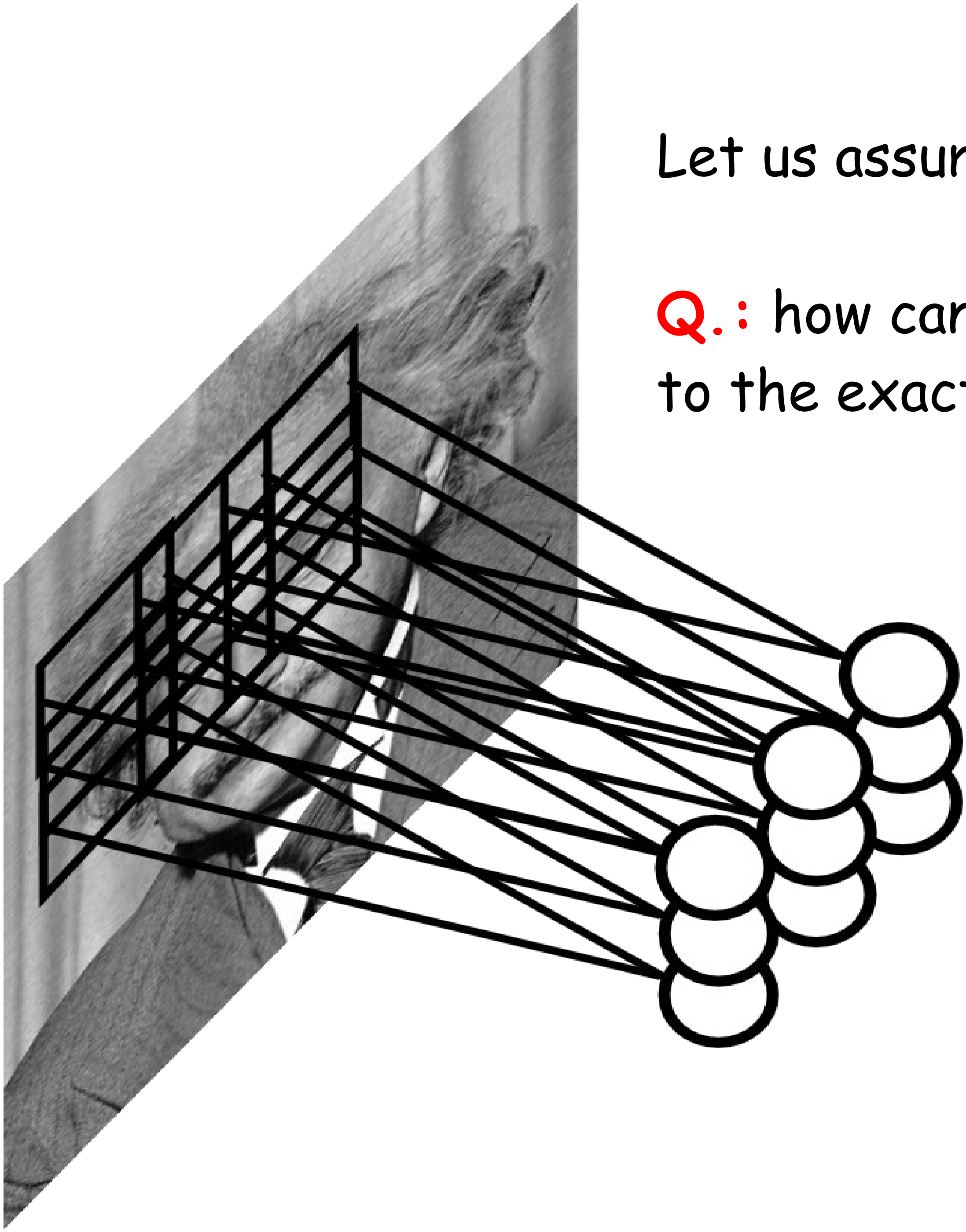
Solution:

- 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***

# CONVOLUTIONAL NET

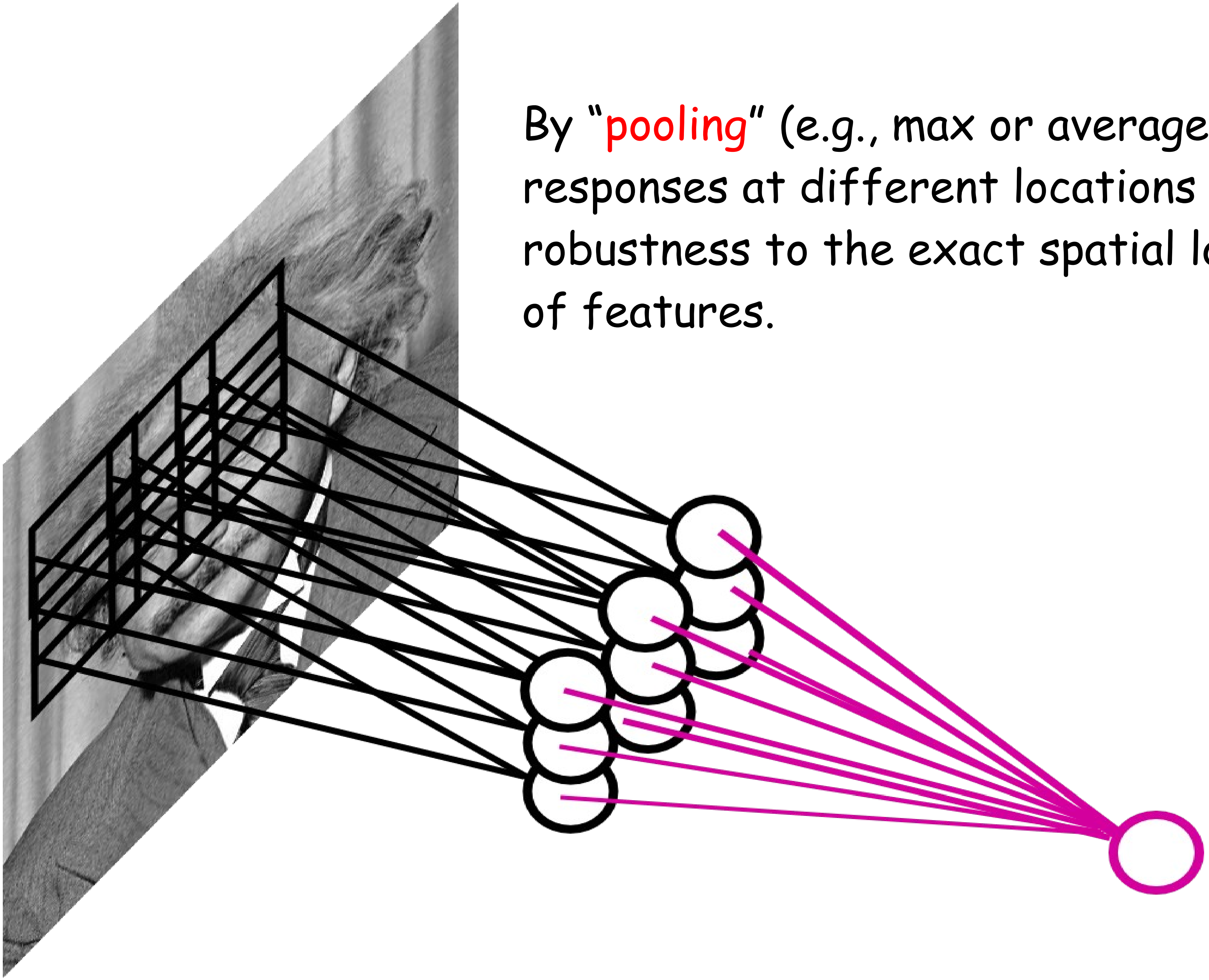


Let us assume filter is an “eye” detector.

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

# CONVOLUTIONAL NET

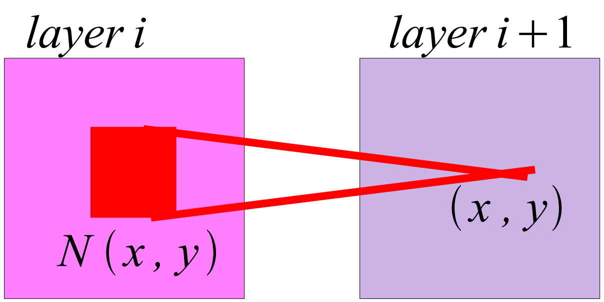
By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.



# CONV NETS: EXTENSIONS

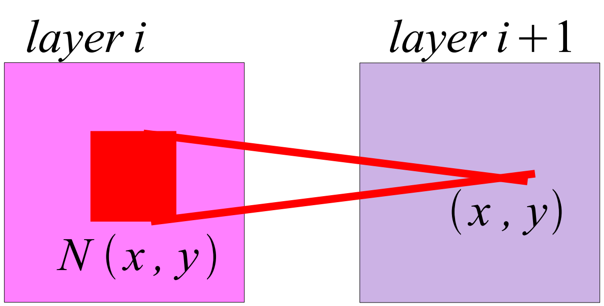
Over the years, some new modules have proven to be very effective when plugged into conv-nets:

## - L2 Pooling




$$h_{i+1,x,y} = \sqrt{\sum_{(j,k) \in N(x,y)} h_{i,j,k}^2}$$

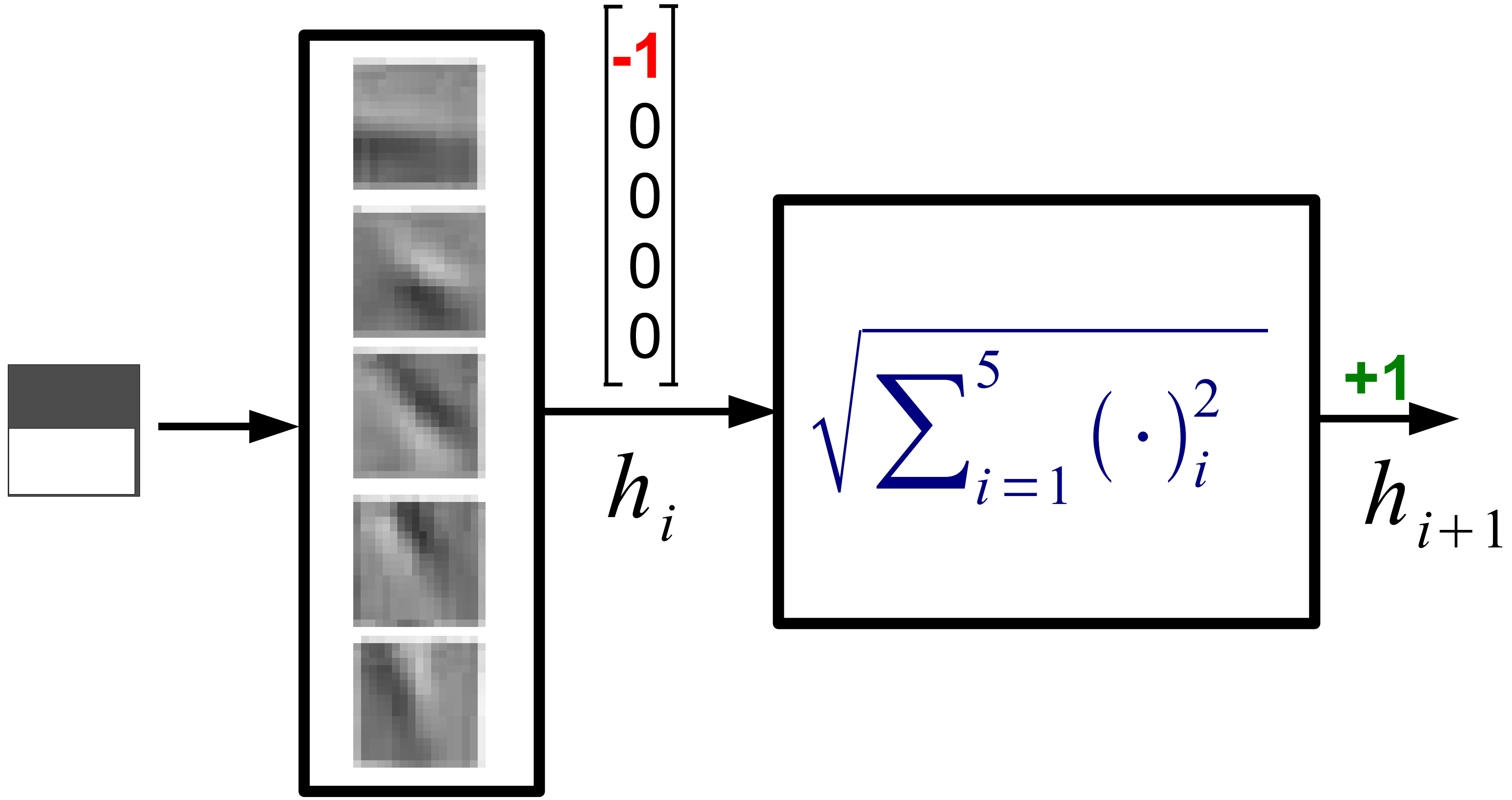
## - Local Contrast Normalization




$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$

*Jarrett et al. "What is the best multi-stage architecture for object recognition?" ICCV 2009*

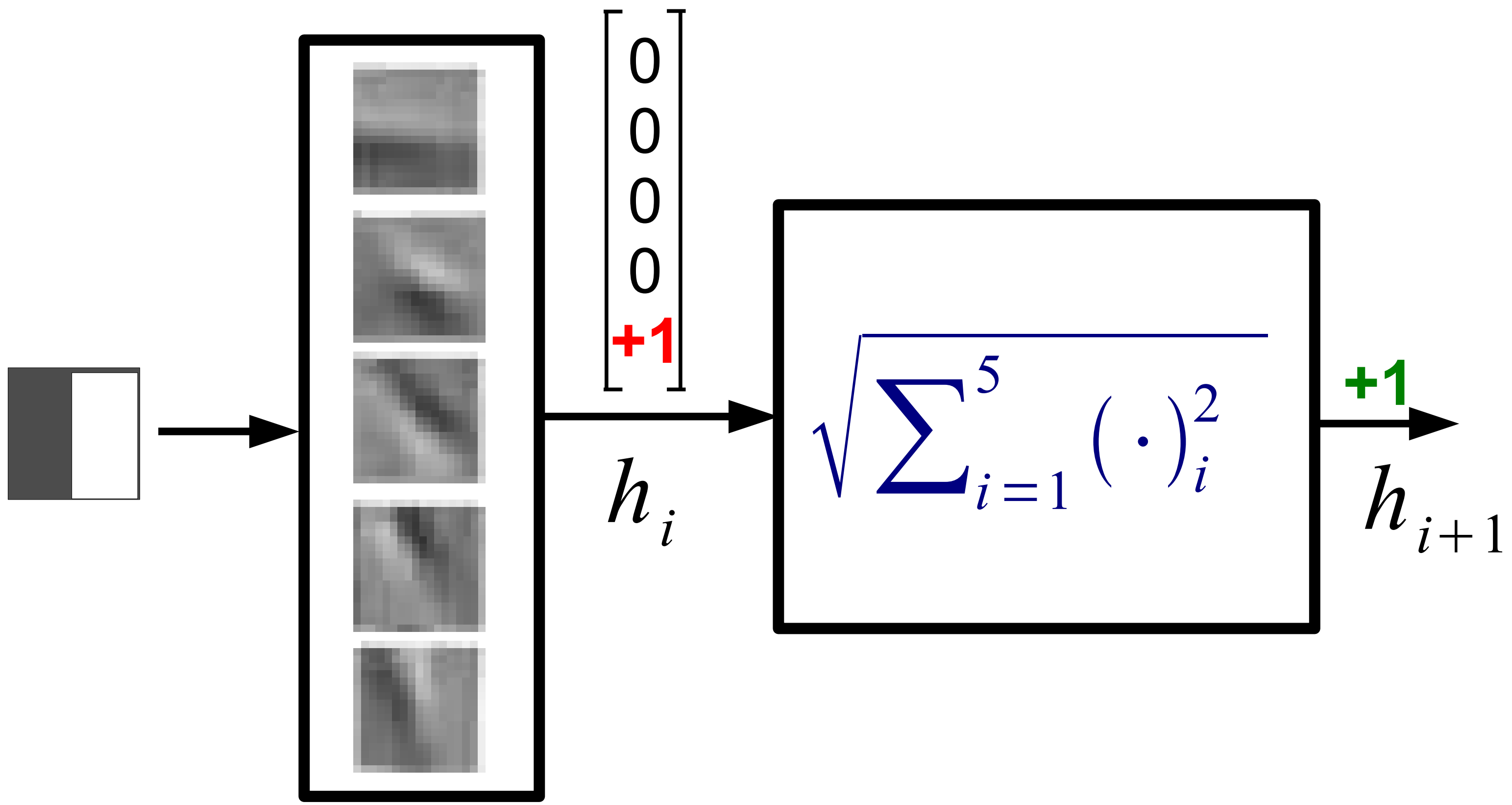
# CONV NETS: L2 POOLING






Kavukguoglu et al. "Learning invariant features ..." CVPR 2009

# CONV NETS: L2 POOLING

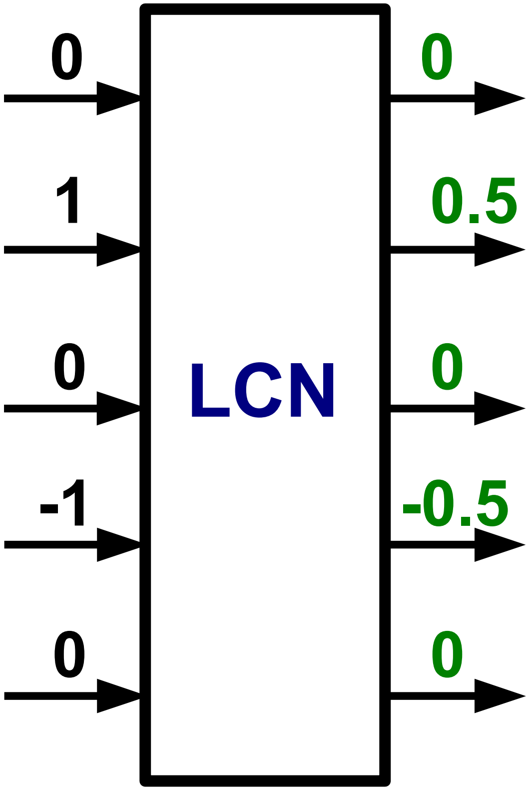





*Kavukguoglu et al. "Learning invariant features ..." CVPR 2009*

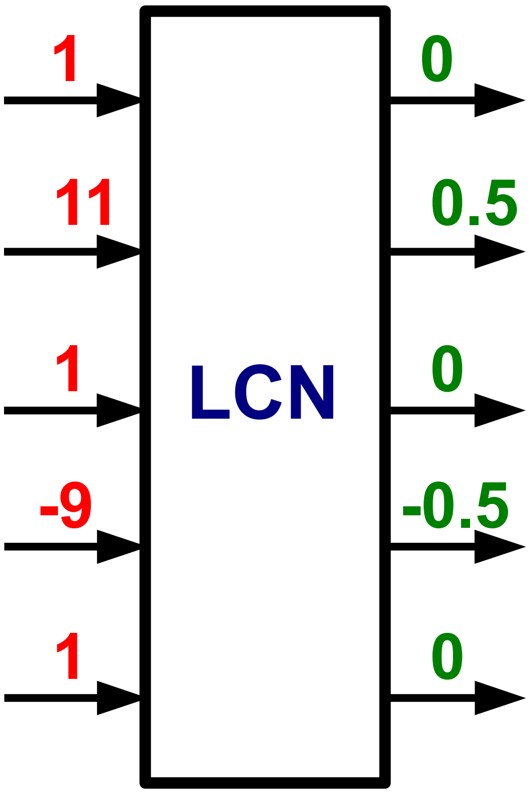
# LOCAL CONTRAST NORMALIZATION

$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$



# LOCAL CONTRAST NORMALIZATION

$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$

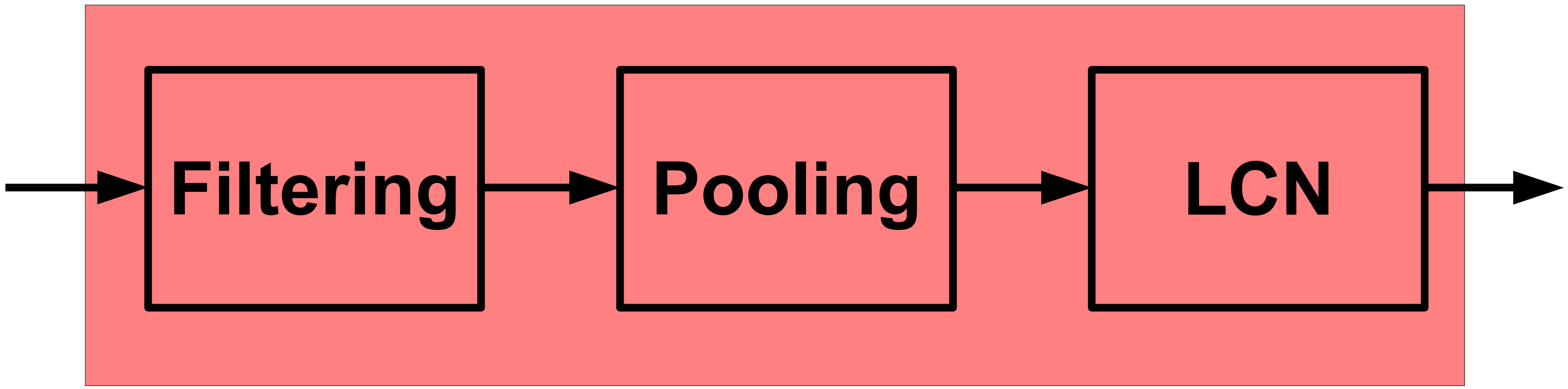


# CONV NETS: EXTENSIONS

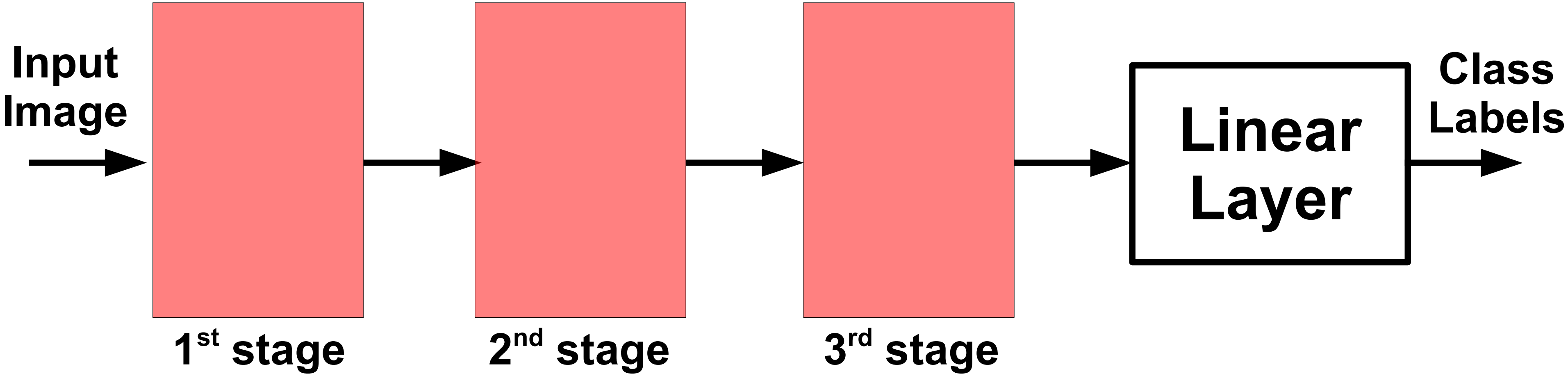
**L2 Pooling & Local Contrast Normalization**
**help learning more invariant representations!**

# CONV NETS: TYPICAL ARCHITECTURE

One stage (zoom)




Whole system






Ranzato

# CONV NETS: TRAINING

Since convolutions and sub-sampling are differentiable, we can use standard back-propagation.

**Algorithm:**

**Given a small mini-batch**

**- FPROP**

**- BPROP**

**- PARAMETER UPDATE**

# CONV NETS: EXAMPLES

- **Object category recognition**
  Boureau et al. "Ask the locals: multi-way local pooling for image recognition" ICCV 2011
- **Segmentation**
  Turaga et al. "Maximin learning of image segmentation" NIPS 2009
- **OCR**
  Ciresan et al. "MCDNN for Image Classification" CVPR 2012
- Pedestrian **detection**
  Kavukcuoglu et al. "Learning convolutional feature hierarchies for visual recognition" NIPS 2010
- **Robotics**
  Sermanet et al. "Mapping and planning..with long range perception" IROS 2008

# CONV NETS: LIMITATIONS

- requires lots of labeled data to train

- difficult optimization

- scalability

# LIMITATIONS & SOLUTIONS

- requires lots of labeled data to train
+ unsupervised learning

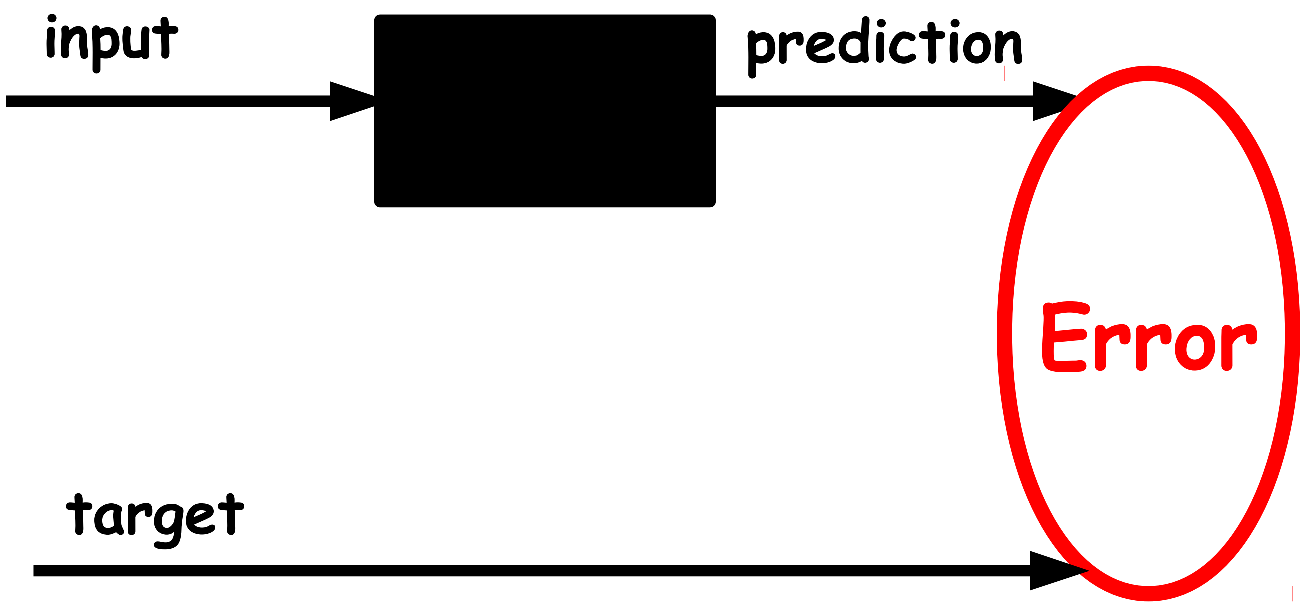
- difficult optimization
+ layer-wise training

- scalability
+ distributed training

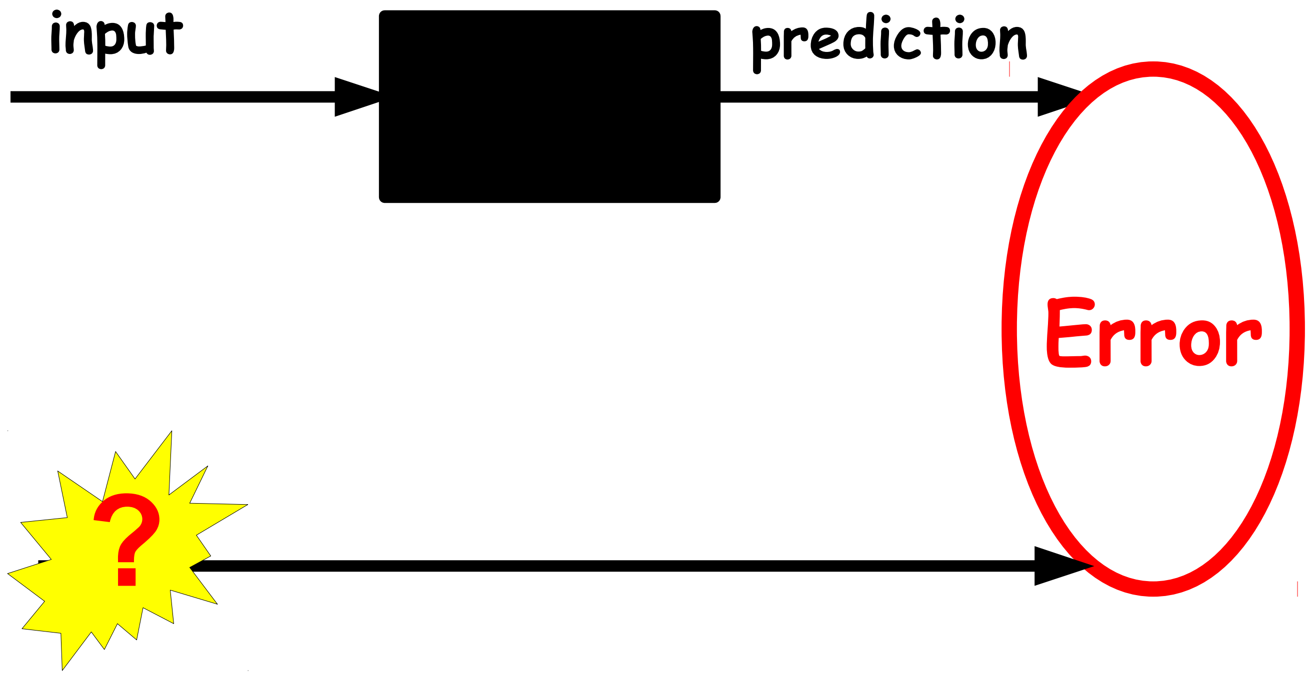
# Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function
- Neural Networks for Vision: Convolutional & Tiled
- **Unsupervised Training of Neural Networks**
- Extensions:
  - semi-supervised / multi-task / multi-modal
- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models
- Large-Scale Learning with Deep Neural Nets

# BACK TO LOGISTIC REGRESSION



# Unsupervised Learning



# Unsupervised Learning

**Q**: How should we train the input-output mapping if we do not have target values?


**A:** Code has to retain information from the input but only if this is similar to training samples.

By better representing only those inputs that are similar to training samples we hope to extract interesting structure (e.g., structure of manifold where data live).

# Unsupervised Learning

**Q**: How to constrain the model to represent training samples better than other data points?

# Unsupervised Learning

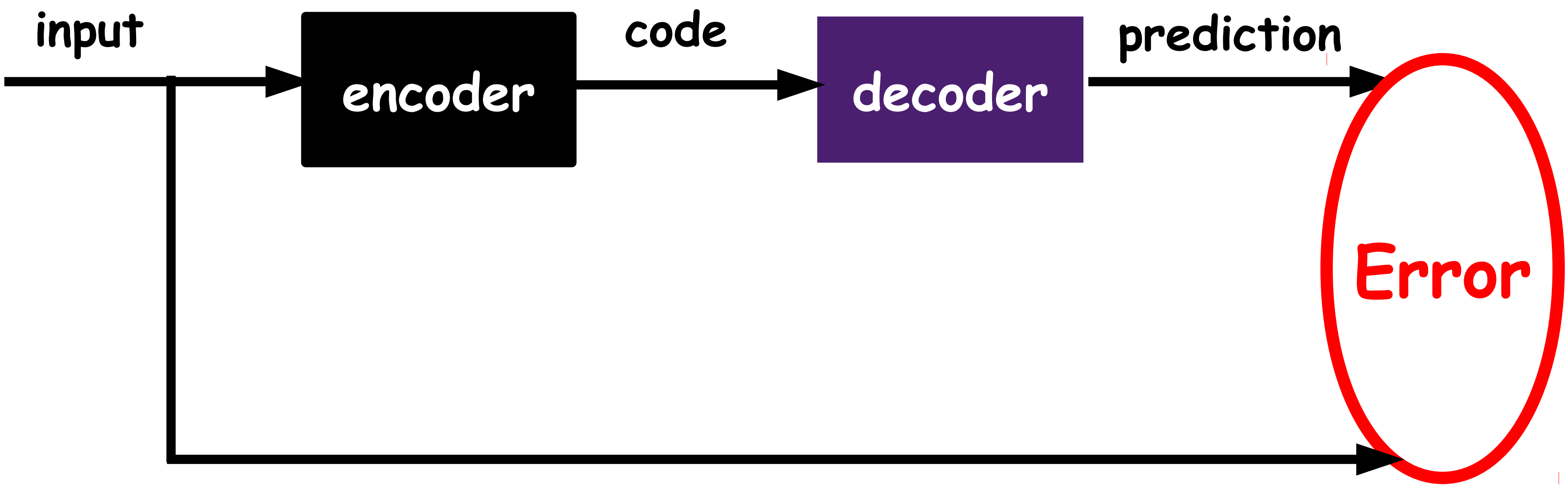
- reconstruct the **input** from the **code** & make code compact
  (auto-encder with bottle-neck).

- reconstruct the input from the code & make code sparse
  (sparse auto-encoders)
  *see work in LeCun, Ng, Fergus, Lee, Yu's labs*

- add noise to the input or code (denoising auto-encoders)
  *see work in Y. Bengio, Lee's lab*

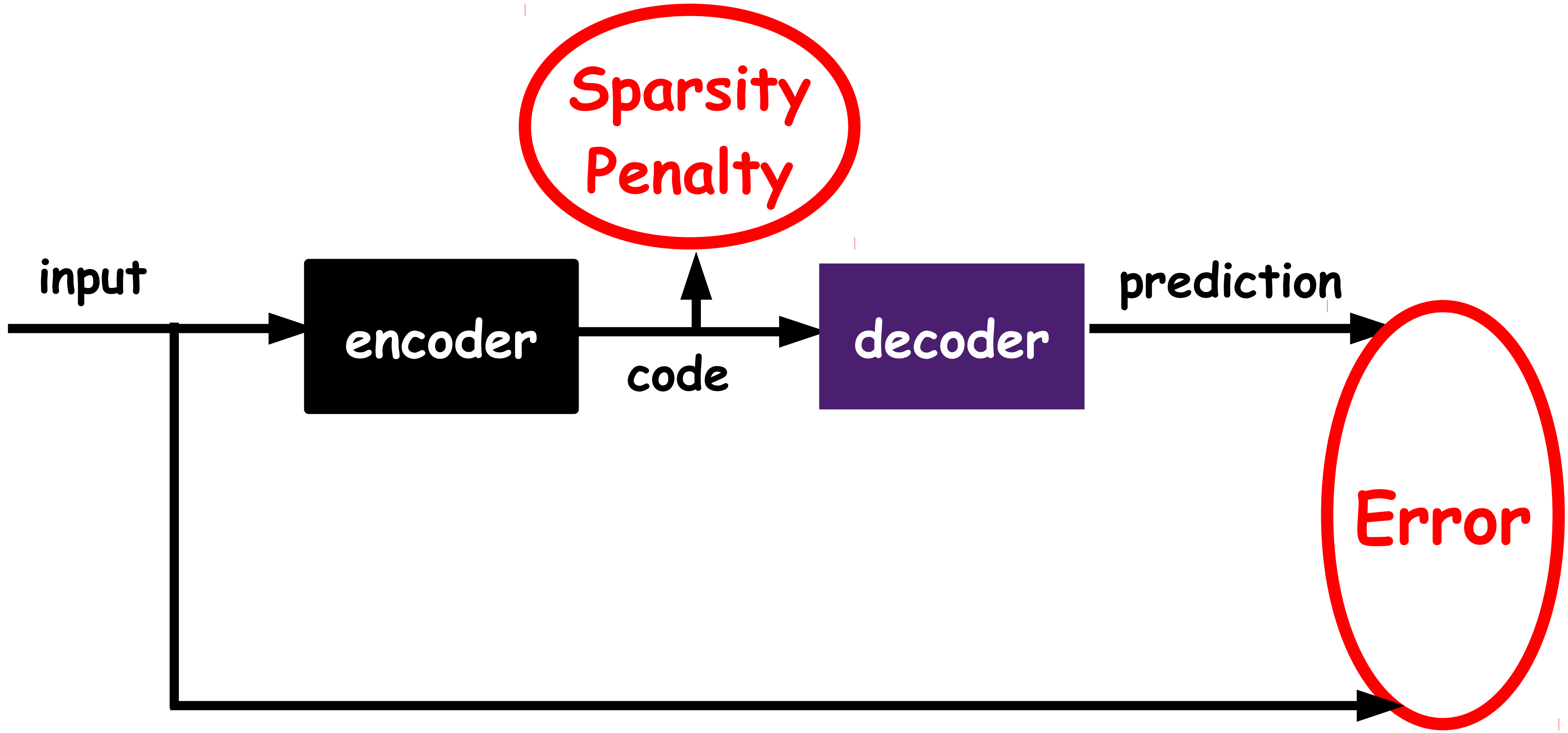
- make sure that the model defines a distribution that normalizes
  to 1 (RBM).
  *see work in Y. Bengio, Hinton, Lee, Salakthudinov's lab*

# AUTO-ENCODERS NEURAL NETS



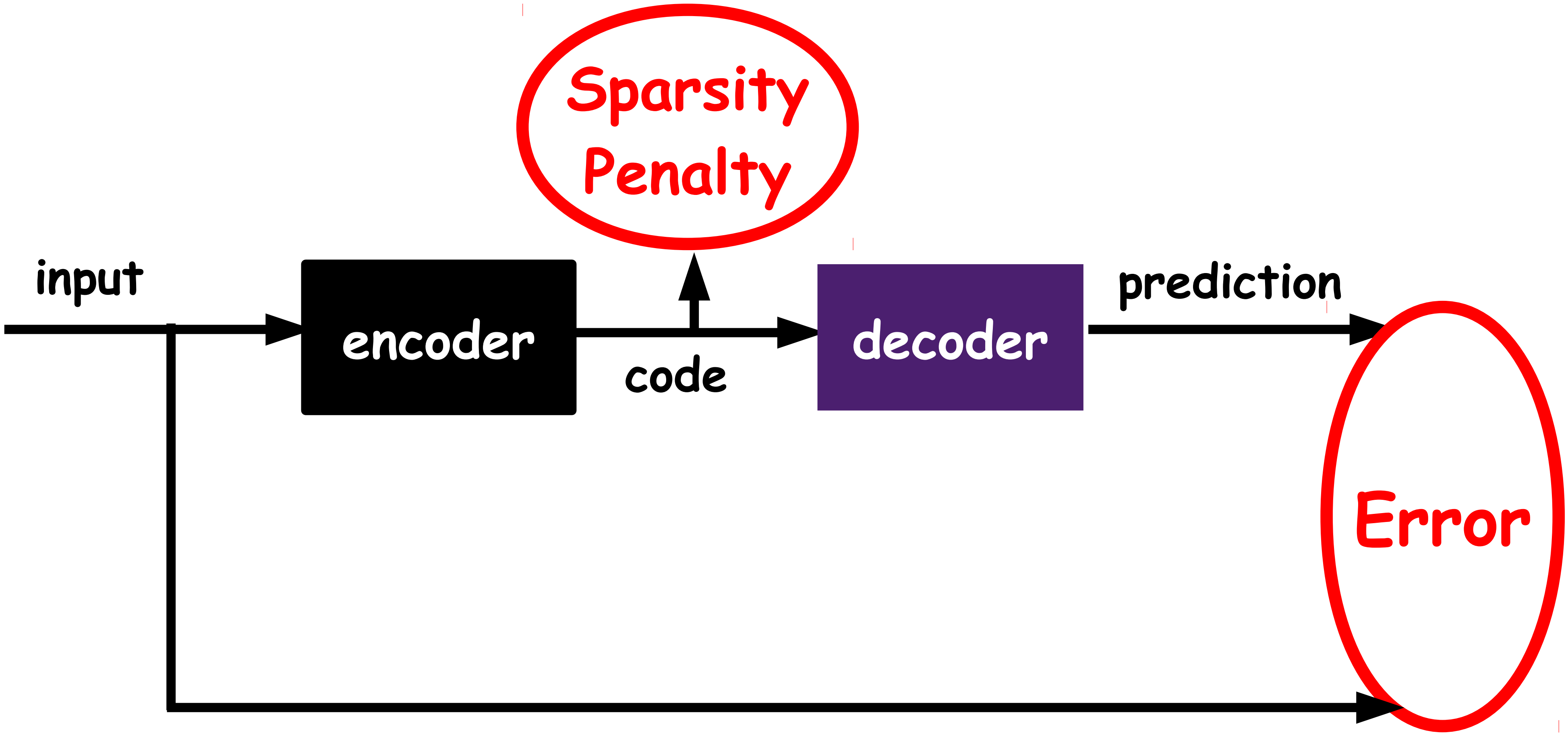

- input higher dimensional than code
- error: $||\text{prediction} - \text{input}||^2$
- training: back-propagation

# SPARSE AUTO-ENCODERS




- sparsity penalty: $||\text{code}||_1$
- error: $||\text{prediction - input}||^2$
- loss: sum of square reconstruction error and sparsity
- training: back-propagation

# SPARSE AUTO-ENCODERS



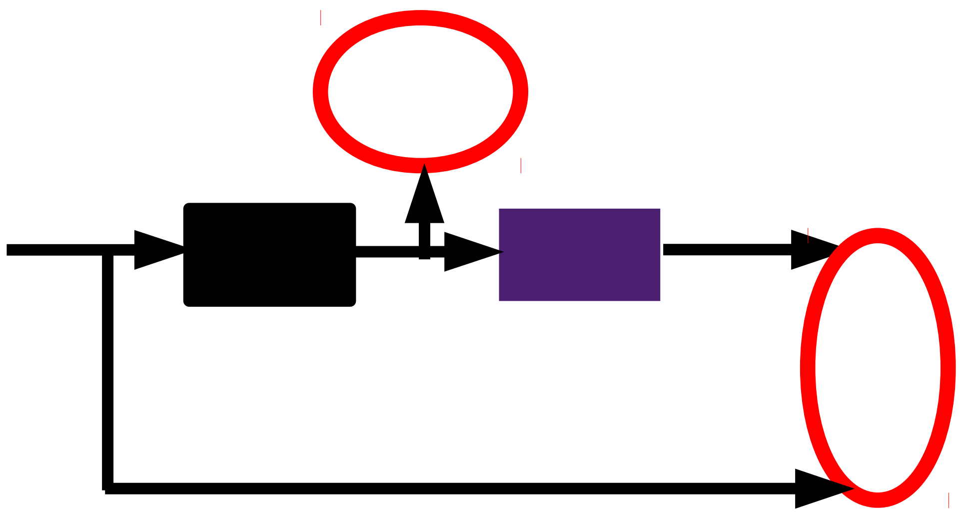

- input: $X$ code: $h = W^T X$

- loss: $L(X; W) = \|W h - X\|^2 + \lambda \sum_j |h_j|$

*Le et al. "ICA with reconstruction cost.." NIPS 2011*

# How To Use Unsupervised Learning

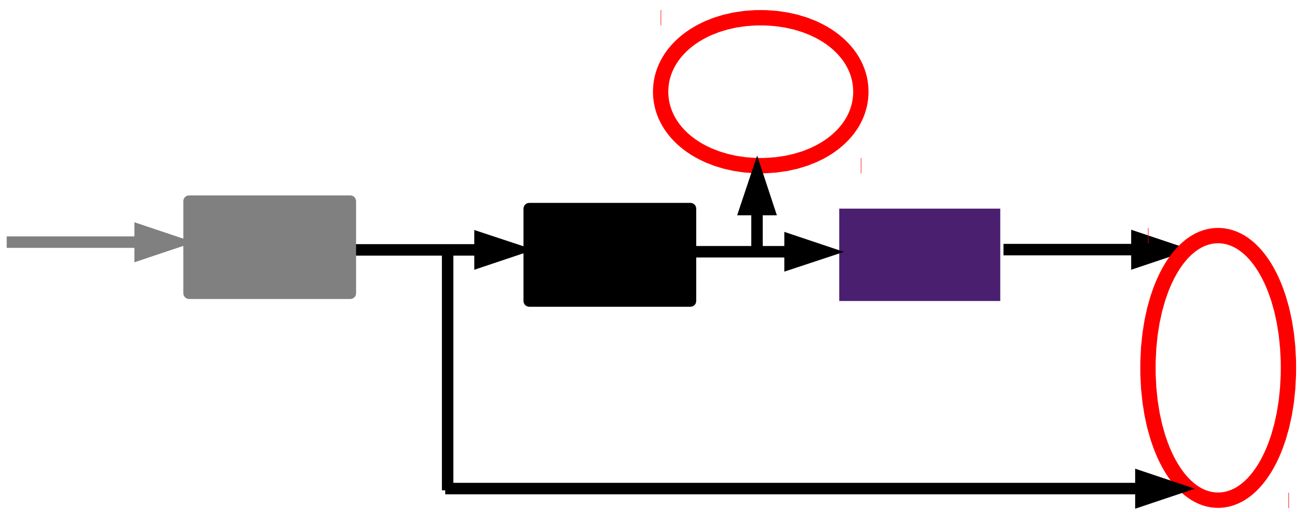
1) Given unlabeled data, learn features



# How To Use Unsupervised Learning

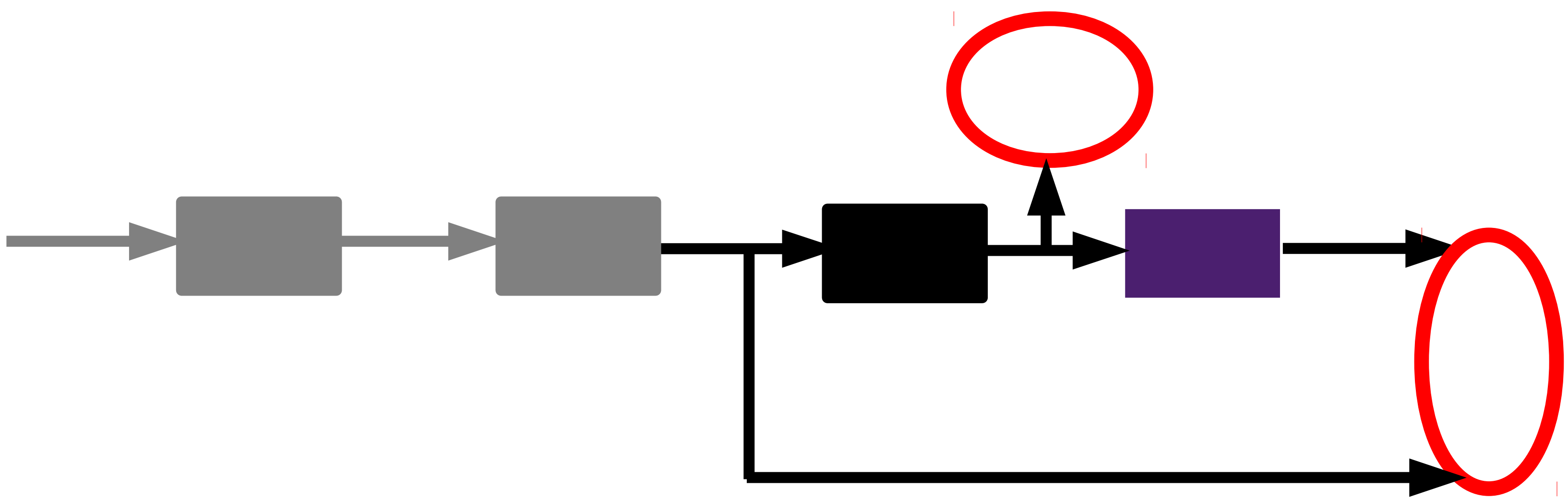
1) Given unlabeled data, learn features

2) Use encoder to produce features and train another layer on the top



# How To Use Unsupervised Learning

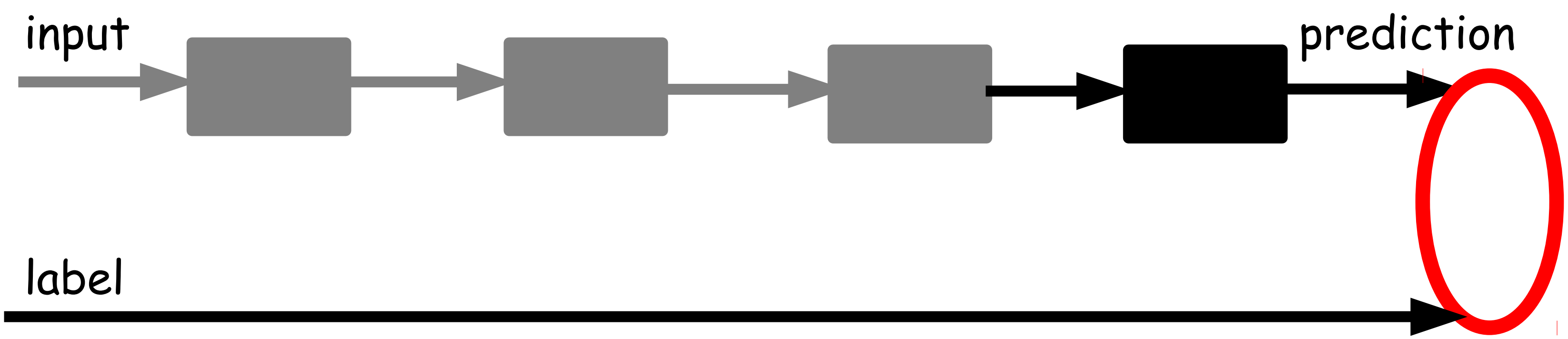
1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top



**Layer-wise training of a feature hierarchy**

# How To Use Unsupervised Learning

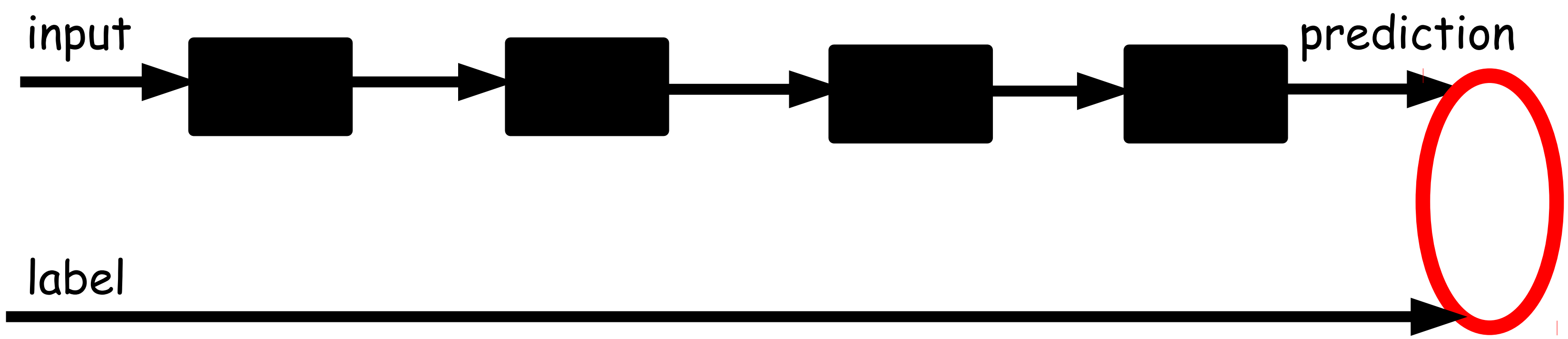
1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top
3) feed features to classifier & train just the classifier




**Reduced overfitting since features are learned in unsupervised way!**

# How To Use Unsupervised Learning

1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top
3) feed features to classifier & jointly train the whole system




**Given enough data, this usually yields the best results: end-to-end learning!**

# Visualizing Learned Features

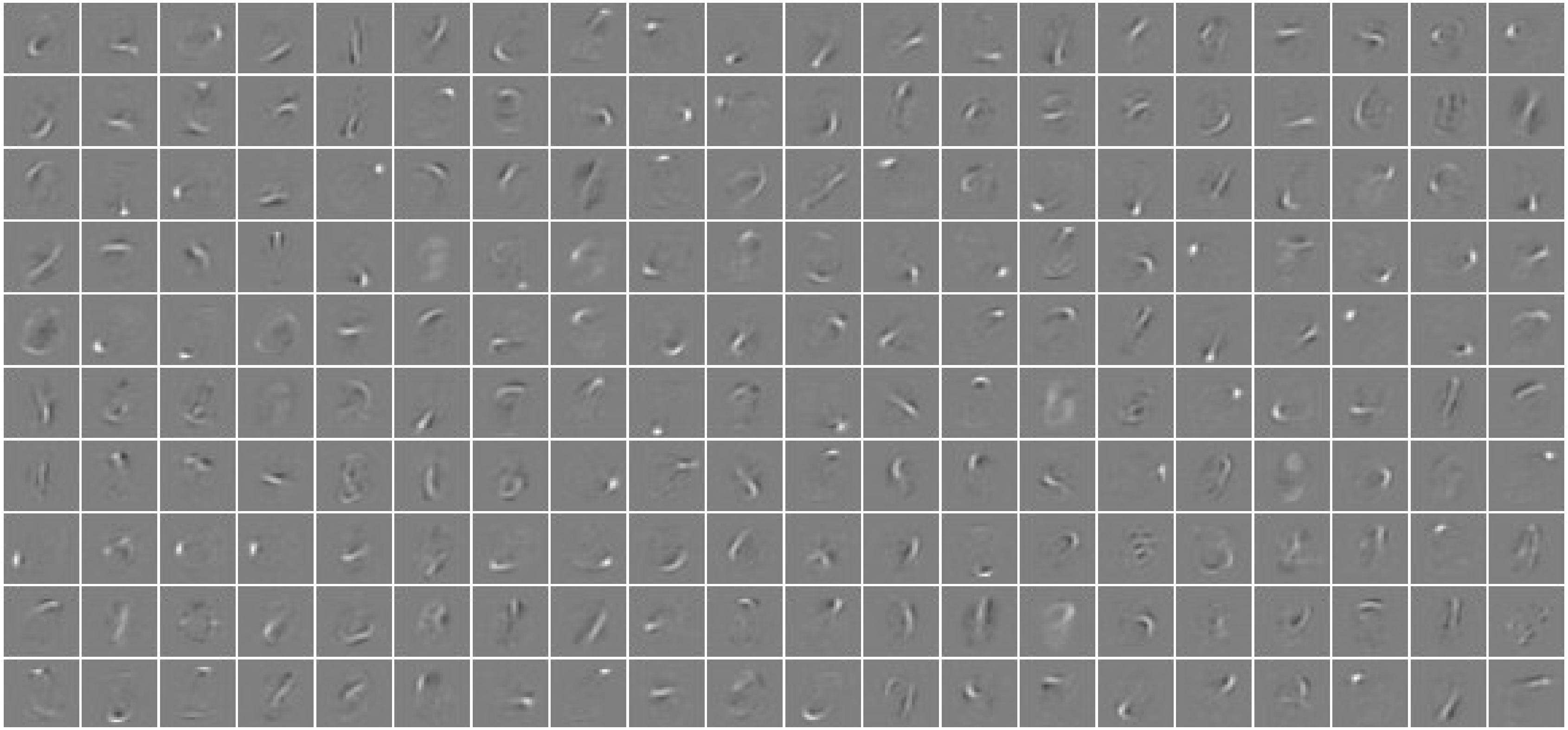
**Q:** can we interpret the learned features?

*Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007*

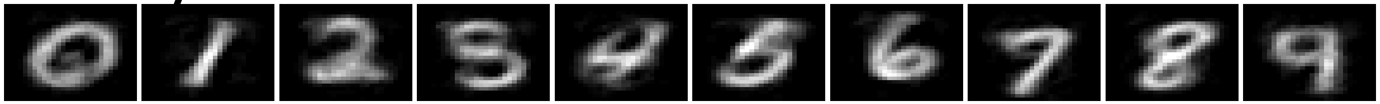
# Visualizing Learned Features

**Q:** how are these images computed?

1st layer features



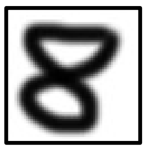
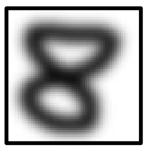
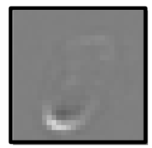
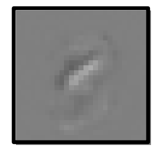
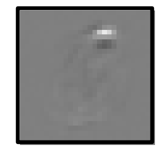
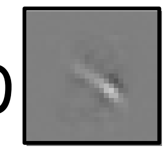
2nd layer features



*Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007*

# Visualizing Learned Features

reconstruction: $W\,h = W_1 h_1 + W_2 h_2 + \ldots$

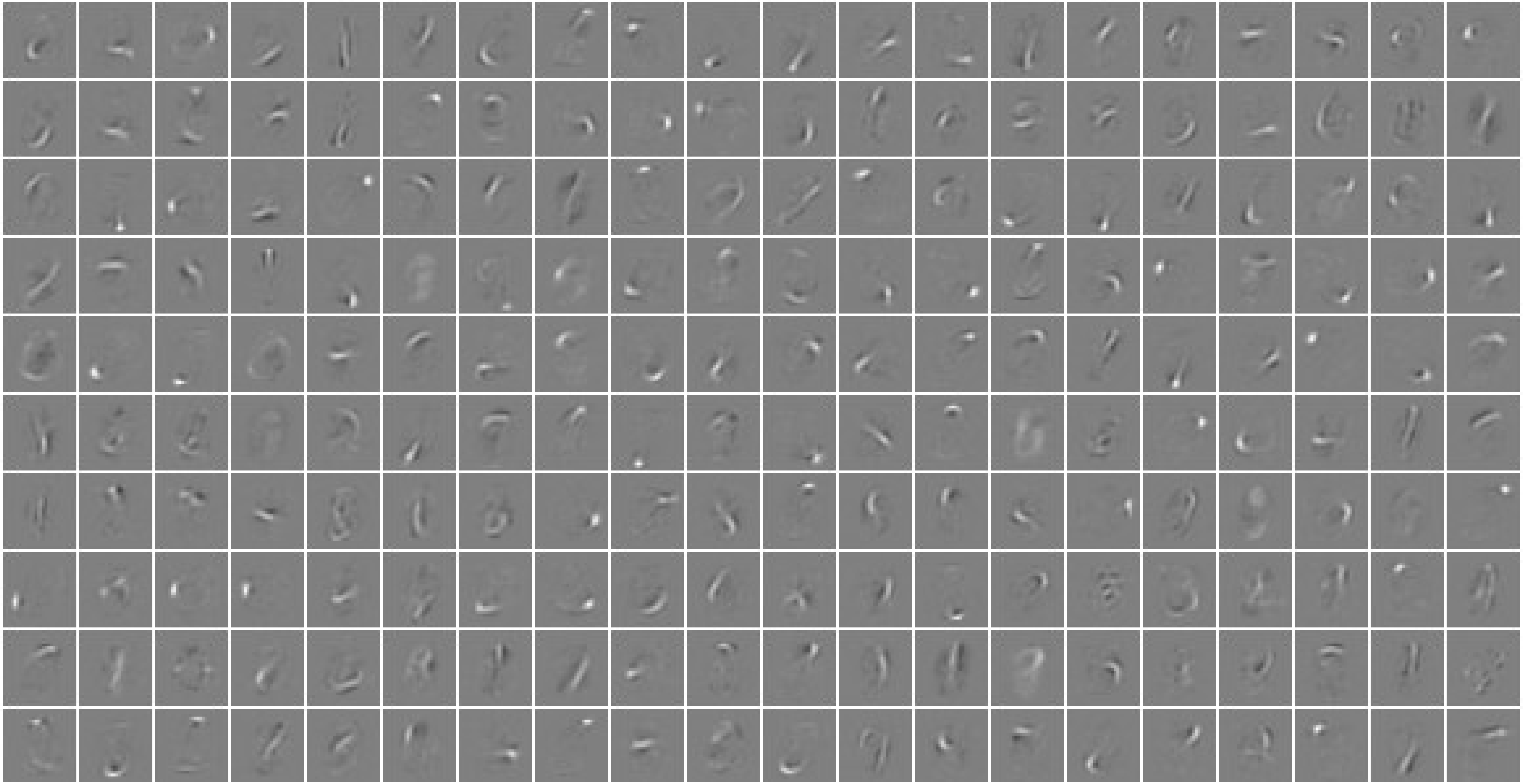
 $\approx$  $=$ 0.9  + 0.7  + 0.5  + 1.0  + ...

Columns of $W$ show what each code unit represents.

***Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007***

# Visualizing Learned Features

1st layer features



*Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007*

# Visualizing Learned Features

**Q:** How about the second layer features?

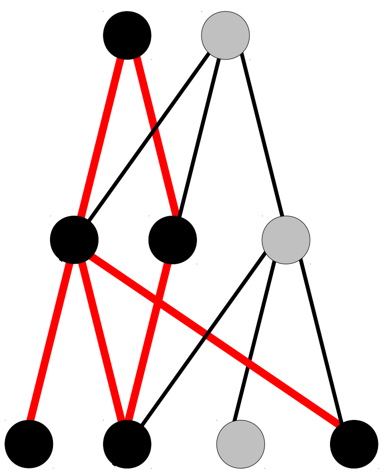
**A:** Similarly, each second layer code unit can be visualized by taking its bases and then projecting those bases in image space through the first layer decoder.

*Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007*

# Visualizing Learned Features

**Q:** How about the second layer features?

**A:** Similarly, each second layer code unit can be visualized by taking its bases and then projecting those bases in image space through the first layer decoder.

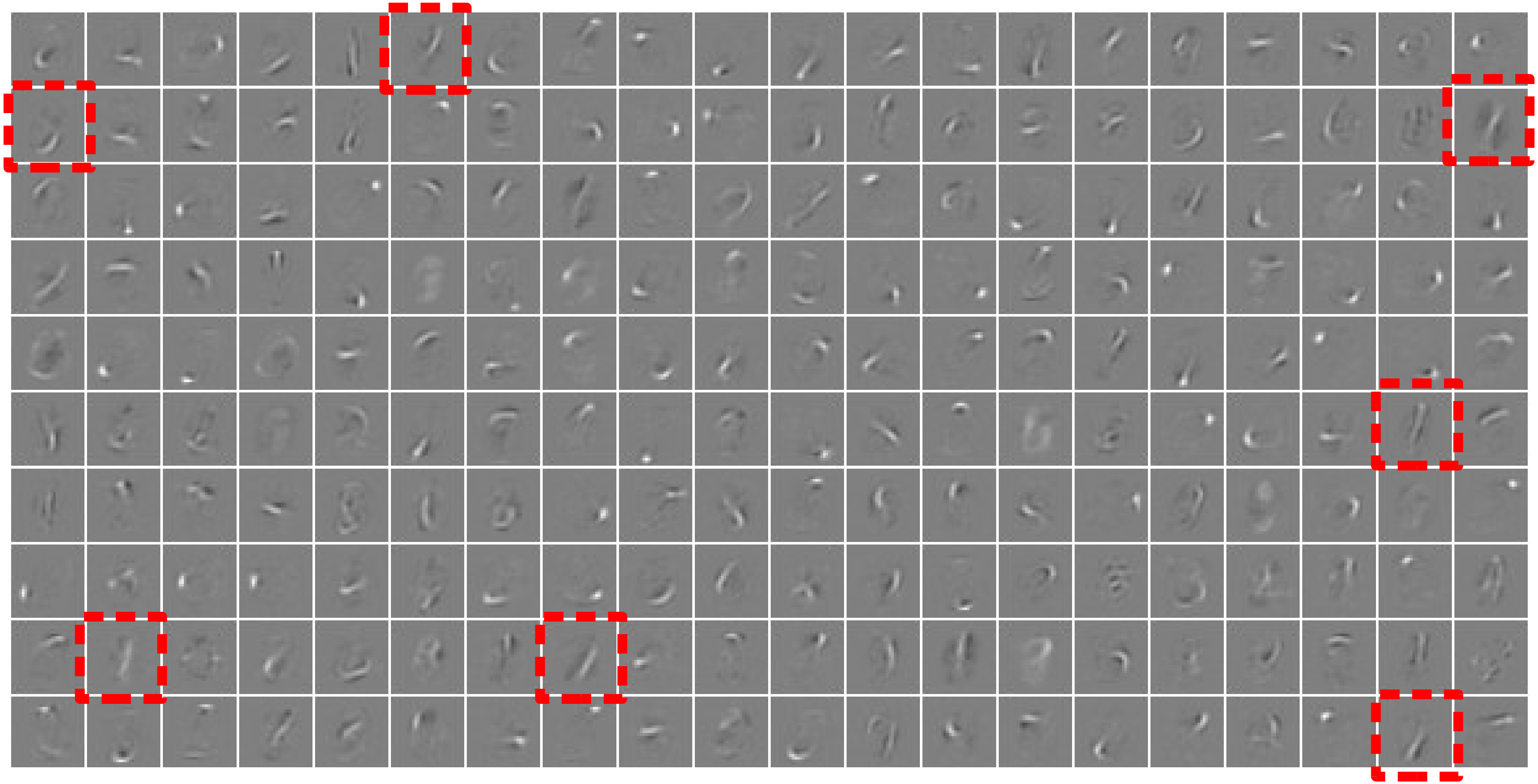


Missing edges have 0 weight.
Light gray nodes have zero value.

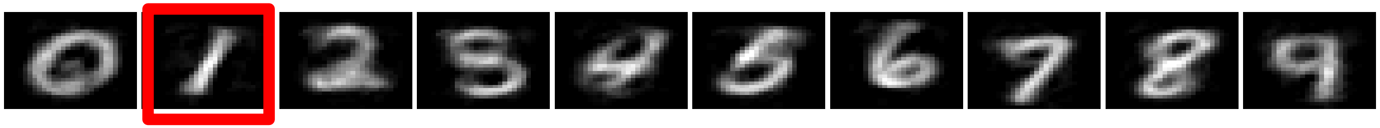
*Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007*

# Example of Feature Learning

1$^{st}$ layer features



2$^{nd}$ layer features



*Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007*



Ranzato

# Outline

- **Neural Networks for Supervised Training**
  - Architecture
  - Loss function
- **Neural Networks for Vision: Convolutional & Tiled**
- **Unsupervised Training of Neural Networks**
- **Extensions**:
  - semi-supervised / multi-task / multi-modal
- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models
- **Large-Scale Learning with Deep Neural Nets**

# Semi-Supervised Learning



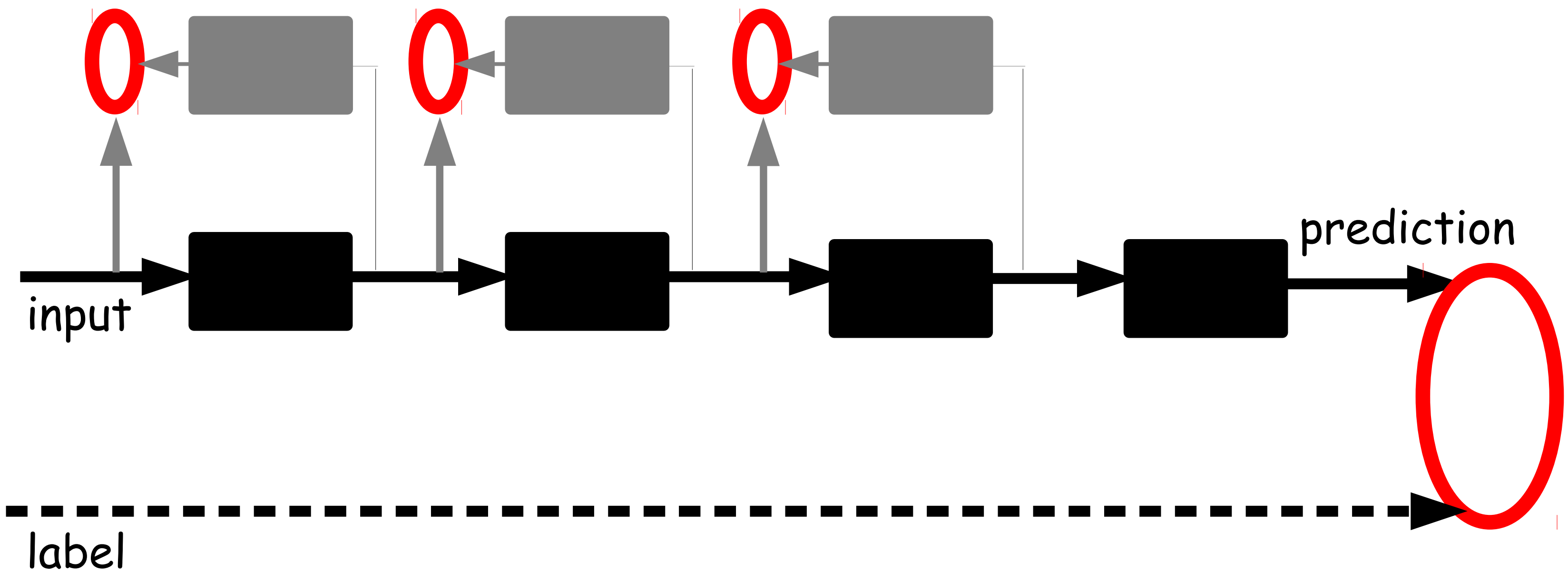

Ranzato

# Semi-Supervised Learning


**LOTS & LOTS OF UNLABELED DATA!!!**

# Semi-Supervised Learning

Loss = supervised_error + unsupervised_error






*Weston et al. "Deep learning via semi-supervised embedding" ICML 2008*

# Multi-Task Learning

Face detection is hard because of lighting, pose, but also occluding goggles.

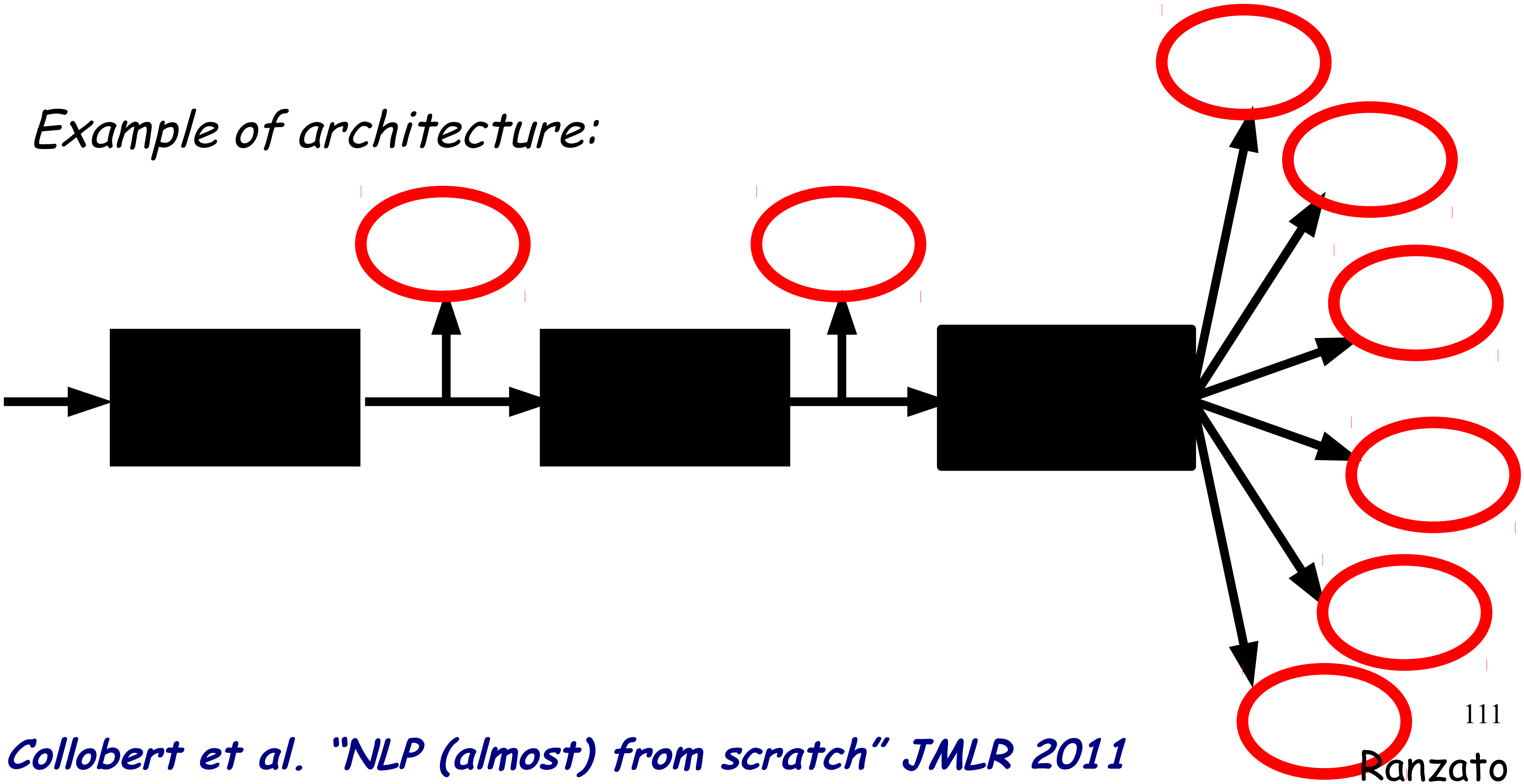
Face detection could made be easier by face identification.

The identification task may help the detection task.

# Multi-Task Learning

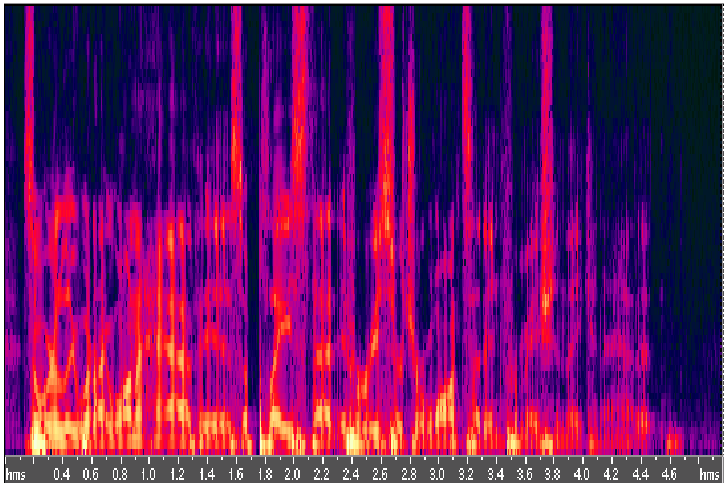
- Easy to add many error terms to loss function.
- Joint learning of related tasks yields better representations.

*Example of architecture:*



***Collobert et al. "NLP (almost) from scratch" JMLR 2011***

# Multi-Modal Learning



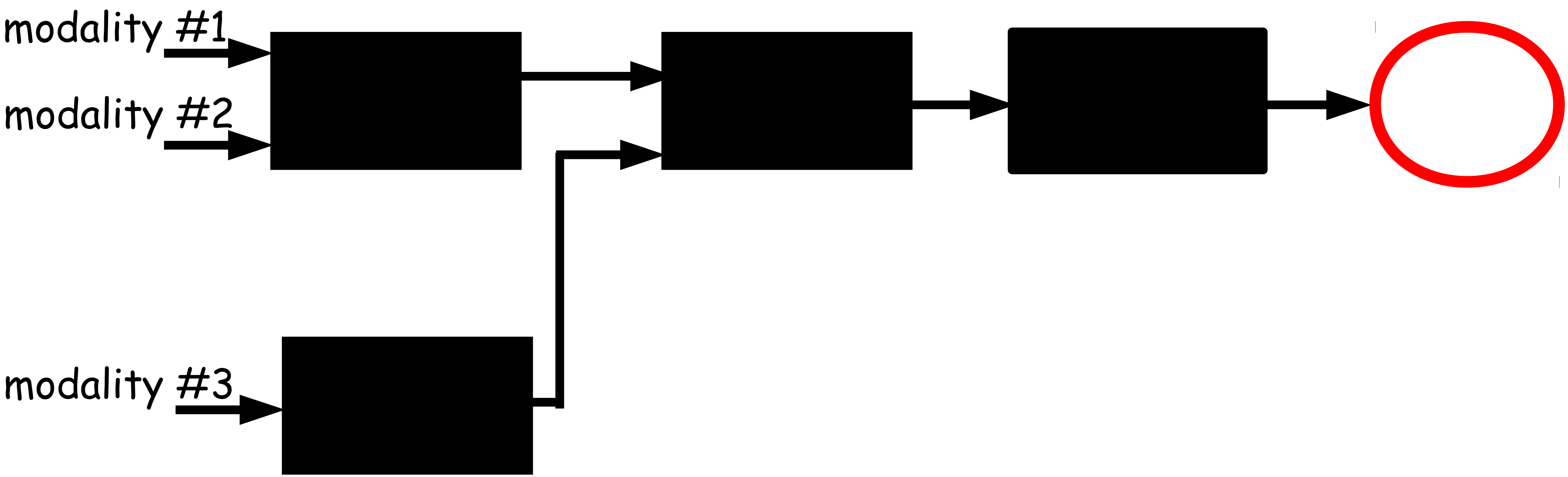

Audio and Video streams are often complimentary to each other.

E.g., audio can provide important clues to improve visual recognition, and vice versa.

# Multi-Modal Learning

- Weak assumptions on input distribution
- Fully adaptive to data

*Example of architecture:*




***Ngiam et al. "Multi-modal deep learning" ICML 2011***

# Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function
- Neural Networks for Vision: Convolutional & Tiled
- Unsupervised Training of Neural Networks
- Extensions:
  - semi-supervised / multi-task / multi-modal
- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models
- Large-Scale Learning with Deep Neural Nets

# Boosting & Forests

**Deep Nets**:

- single highly non-linear system
- "*deep*" stack of simpler modules
- all parameters are subject to learning

---

**Boosting & Forests**:

- sequence of "weak" (simple) classifiers that are linearly combined to produce a powerful classifier
- subsequent classifiers do not exploit representations of earlier classifiers, it's a "*shallow*" linear mixture
- typically features are not learned

| *Properties* | ***Deep Nets*** | ***Boosting*** |
|---|---|---|
| **Adaptive features** | ✓ | ✗ |
| **Hierarchical features** | ✓ | ✗ |
| **End-to-end learning** | ✓ | ✗ |
| **Leverage unlab. data** | ✓ | 😐 |
| **Easy to parallelize** | 😐 | ✓ |
| **Fast training** | 😐 | ✓ |
| **Fast at test time** | 🙂 | ✓ |

# Deep Neural-Nets VS Probabilistic Models

**Deep Neural Nets**:

- mean-field approximations of intractable probabilistic models
- usually more efficient
- typically more unconstrained (partition function has to be replaced by other constraints, e.g. sparsity).

---

**Hierarchical Probabilistic Models (DBN, DBM, etc.)**:

- in the most interesting cases, they are intractable
- they better deal with uncertainty
- they can be easily combined

# Example: Auto-Encoder

## Neural Net:

code $Z = \sigma ( W_e^T X + b_e )$

reconstruction $\hat{X} = W_d Z + b_d$

---

## Probabilistic Model (Gaussian RBM):

$$E [ Z | X ] = \sigma ( W^T X + b_e )$$

$$E [ X | Z ] = W Z + b_d$$

| *Properties* | ***Deep Nets*** | ***Probab. Models*** |
|---|---|---|
| **Adaptive features** | ✓ | ✓ |
| **Hierarchical features** | ✓ | ✓ |
| **End-to-end learning** | ✓ | ✓ |
| **Leverage unlab. data** | ✓ | ✓ |
| **Models uncertainty** | ✗ | ✓ |
| **Fast training** | ☺ | ✗ |
| **Fast at test time** | ☺ | ☹ |

# Outline

- **Neural Networks for Supervised Training**
  - Architecture
  - Loss function
- **Neural Networks for Vision: Convolutional & Tiled**
- **Unsupervised Training of Neural Networks**
- **Extensions:**
  - semi-supervised / multi-task / multi-modal
- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models
- **Large-Scale Learning with Deep Neural Nets**

# Tera-Scale Deep Learning @ Google

**Observation #1:** more features always improve performance unless data is scarce.

**Observation #2:** deep learning methods have higher capacity and have the potential to model data better.

**Q #1:** Given lots of data and lots of machines, can we scale up deep learning methods?

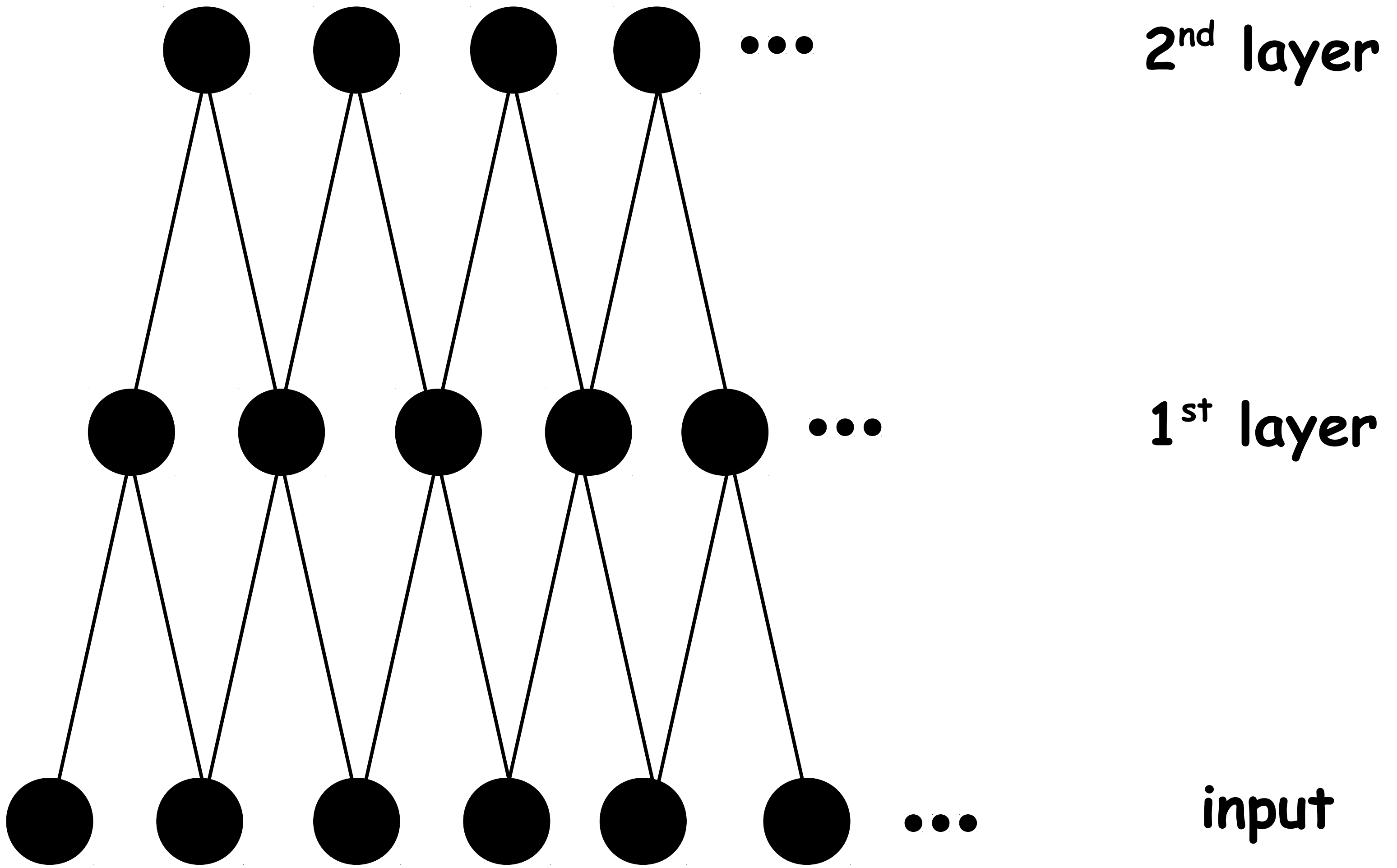
**Q #2:** Will deep learning methods perform much better?

# The Challenge

**A Large Scale problem has:**
- lots of training samples (>10M)
- lots of classes (>10K) and
- lots of input dimensions (>10K).

- best optimizer in practice is on-line SGD which is naturally sequential, hard to parallelize.
- layers cannot be trained independently and in parallel, hard to distribute
- model can have lots of parameters that may clog the network, hard to distribute across machines

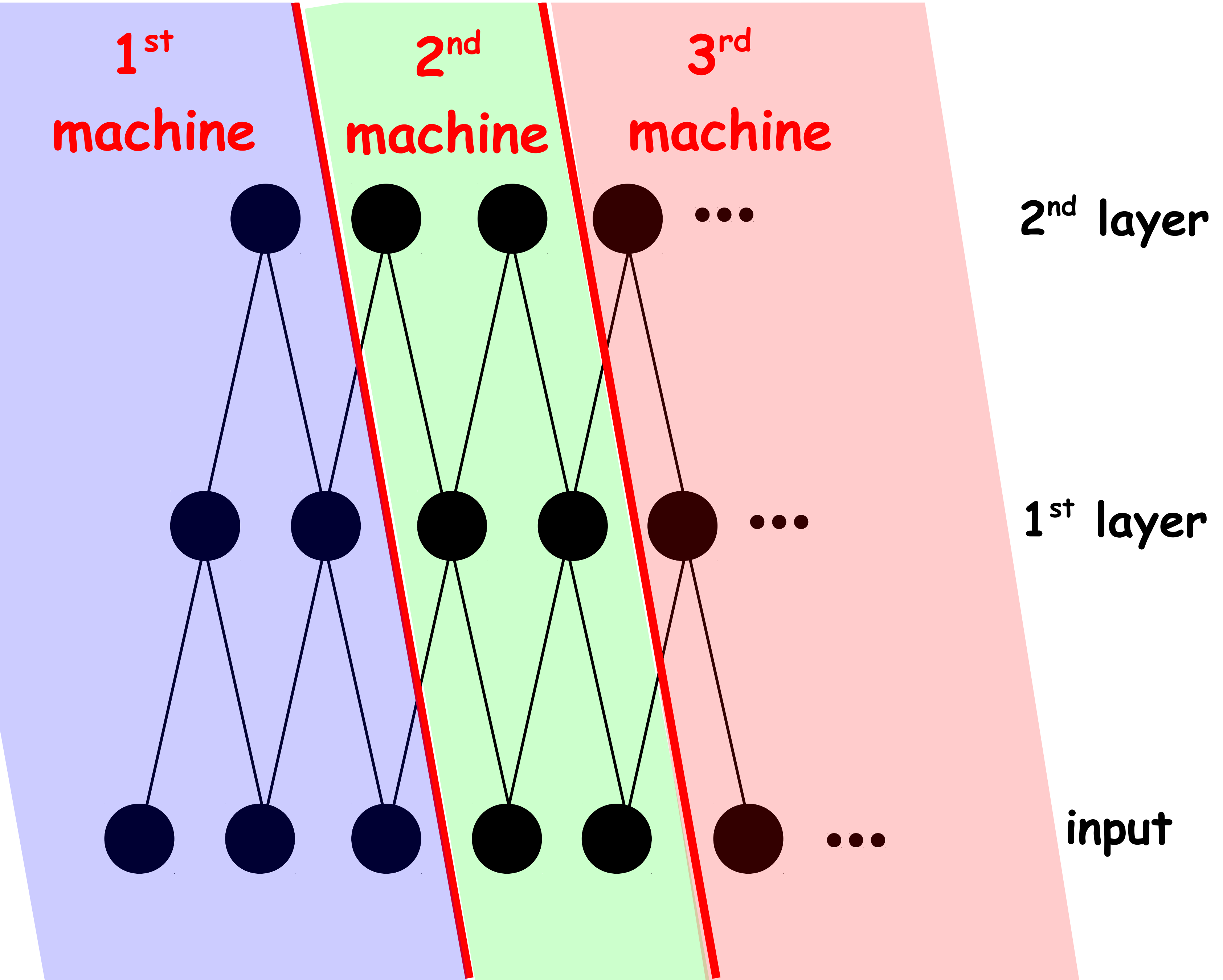
# Our Solution



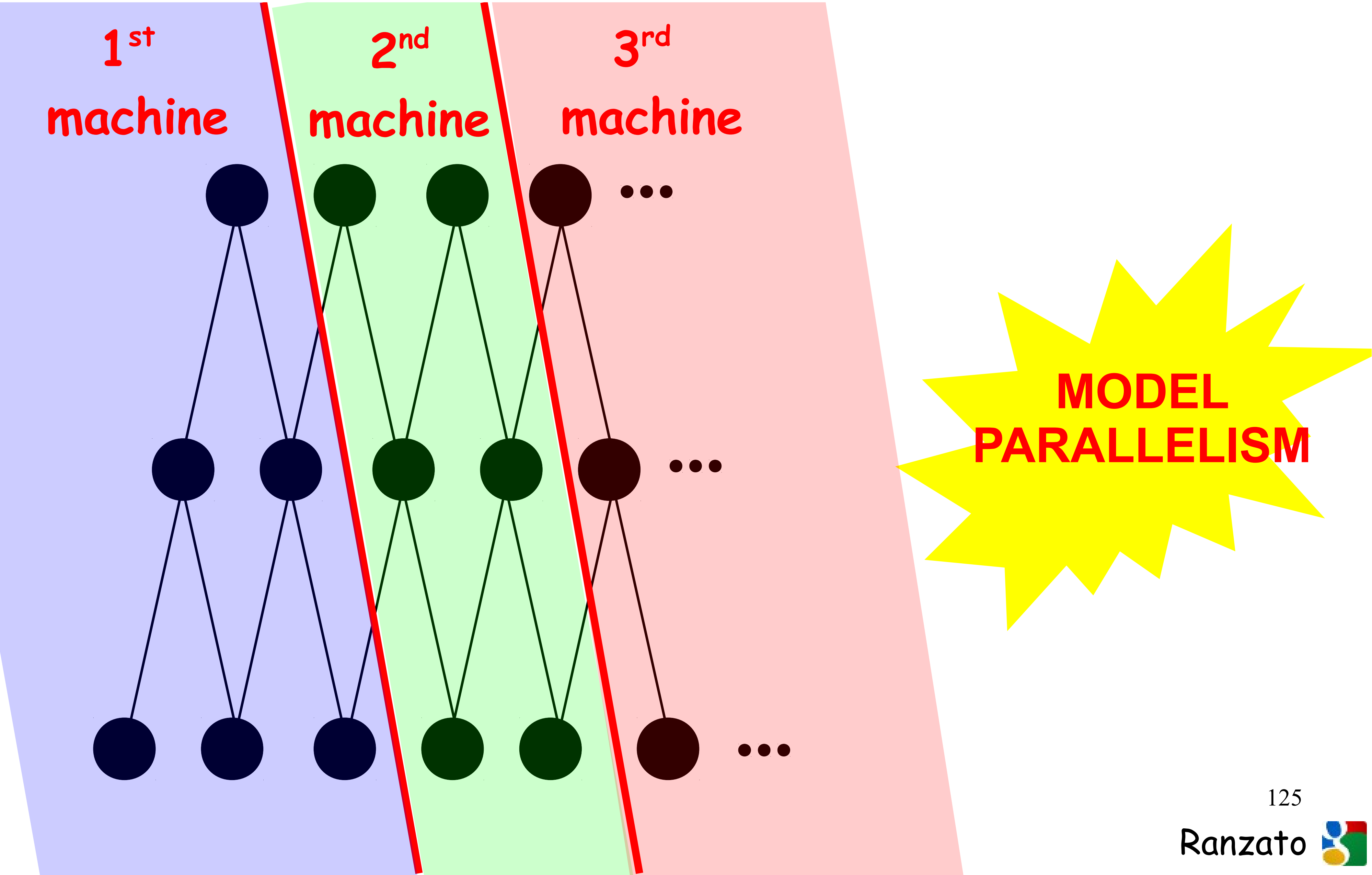



Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012

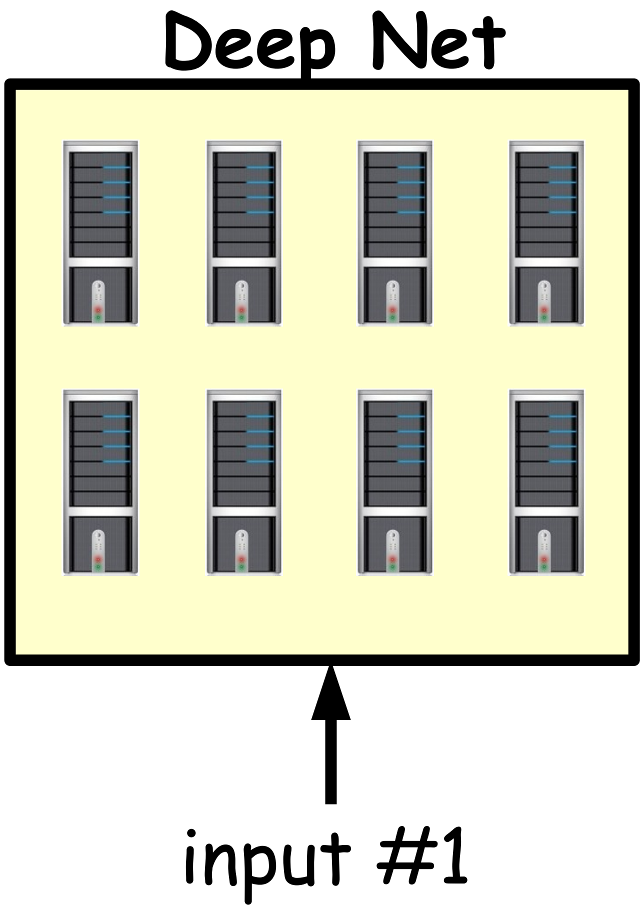
# Our Solution



# Our Solution



# Distributed Deep Nets

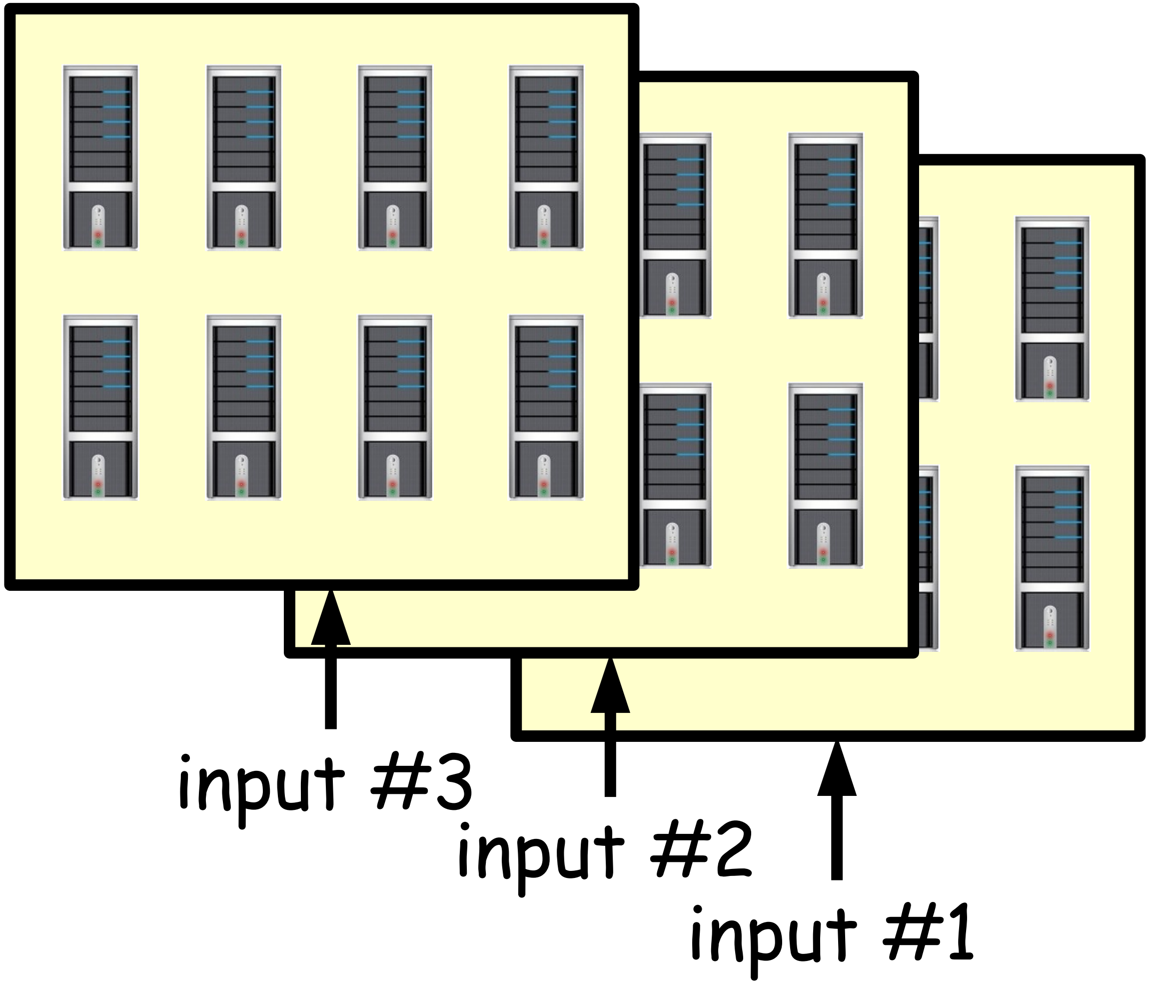



MODEL PARALLELISM



Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012

# Distributed Deep Nets




**MODEL PARALLELISM**

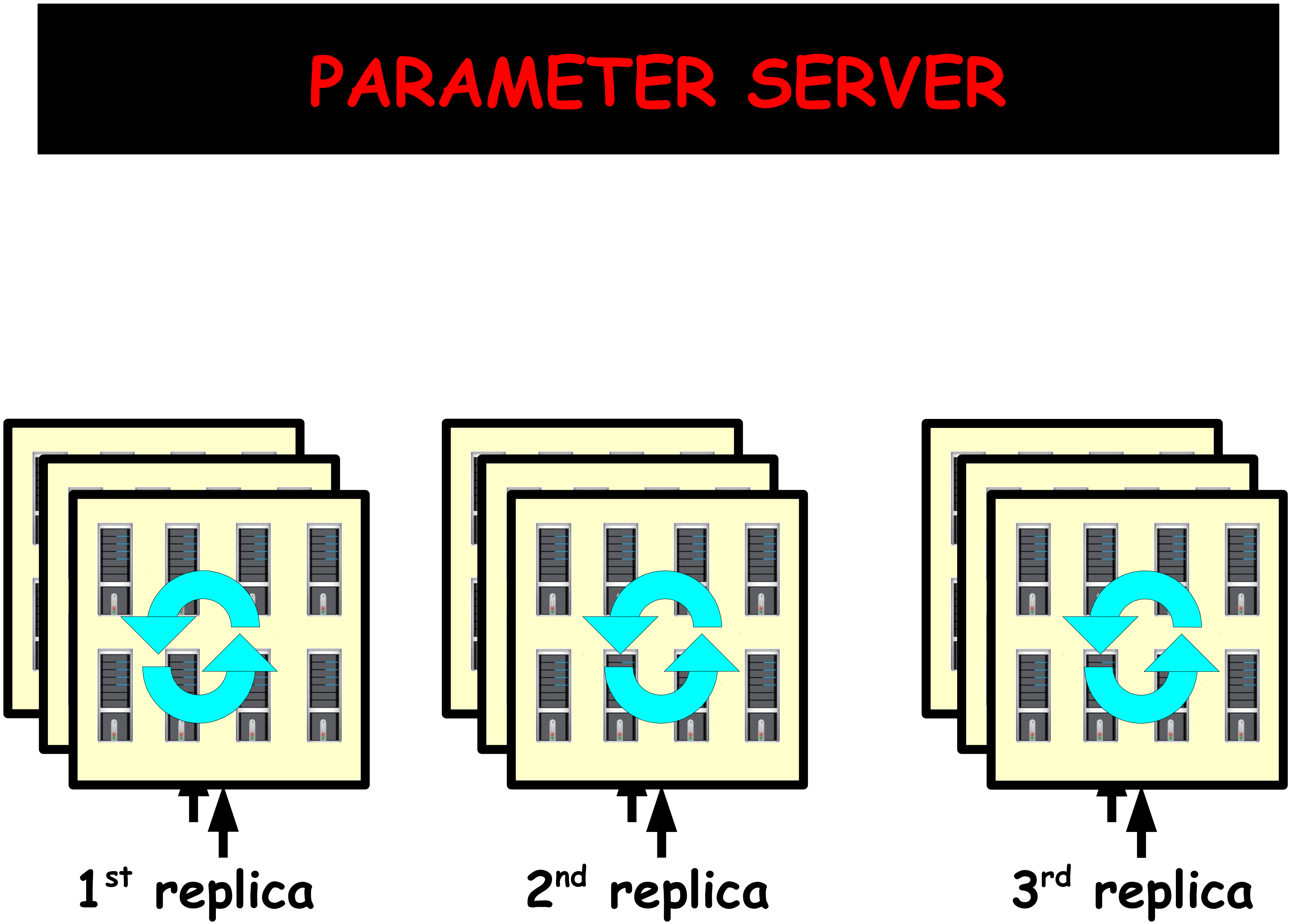
**+**

**DATA PARALLELISM**

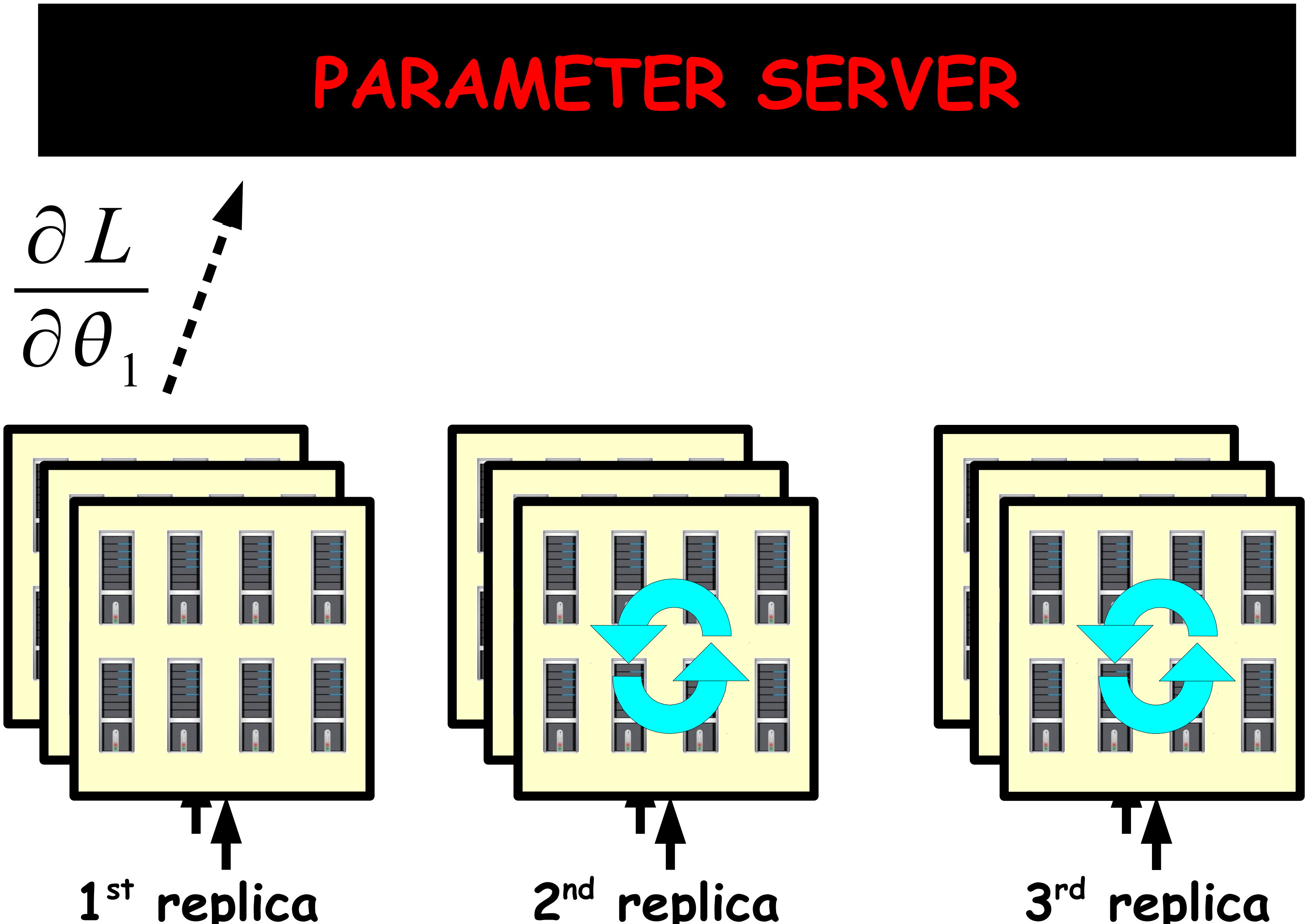


*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

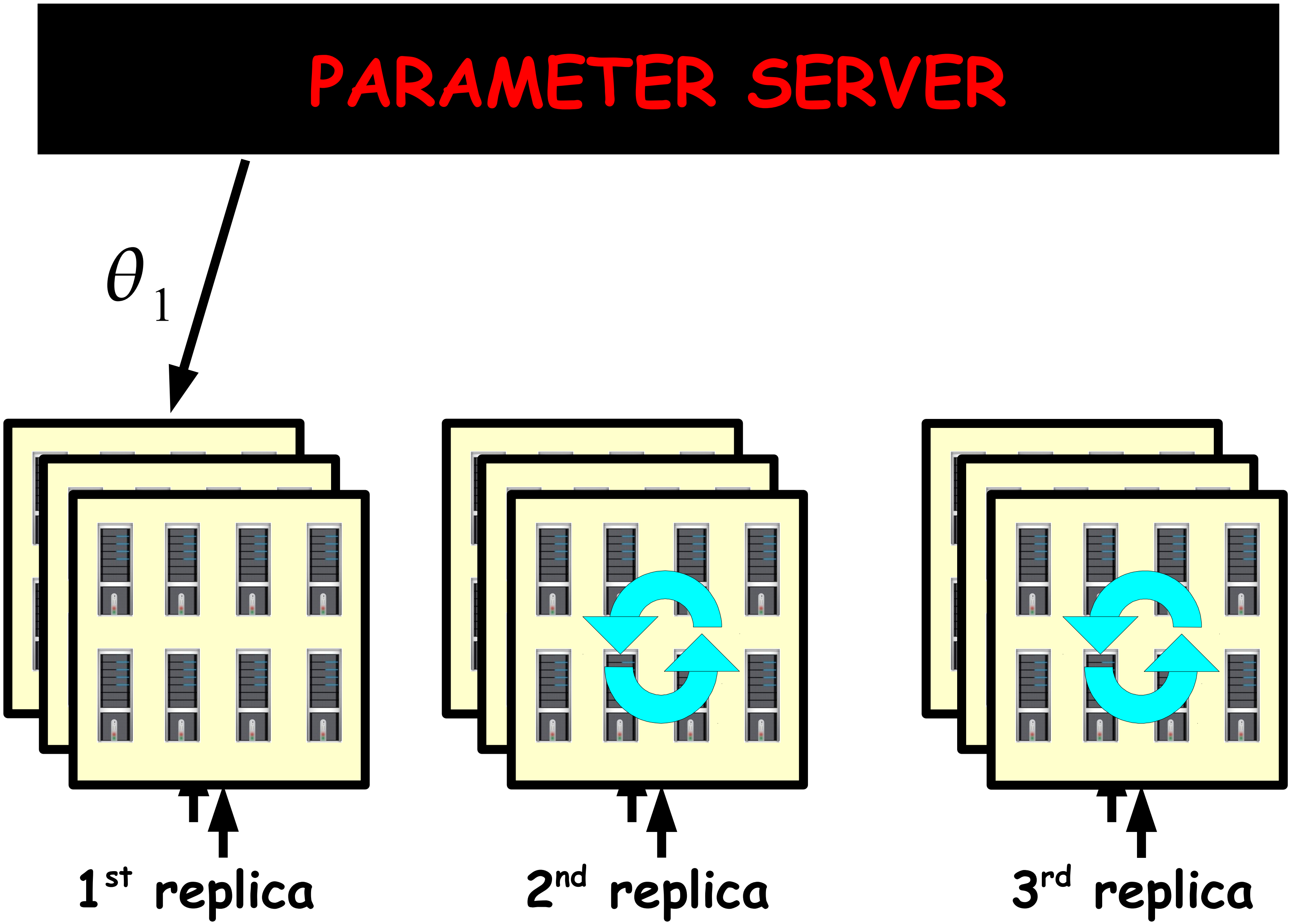
# Asynchronous SGD



# Asynchronous SGD

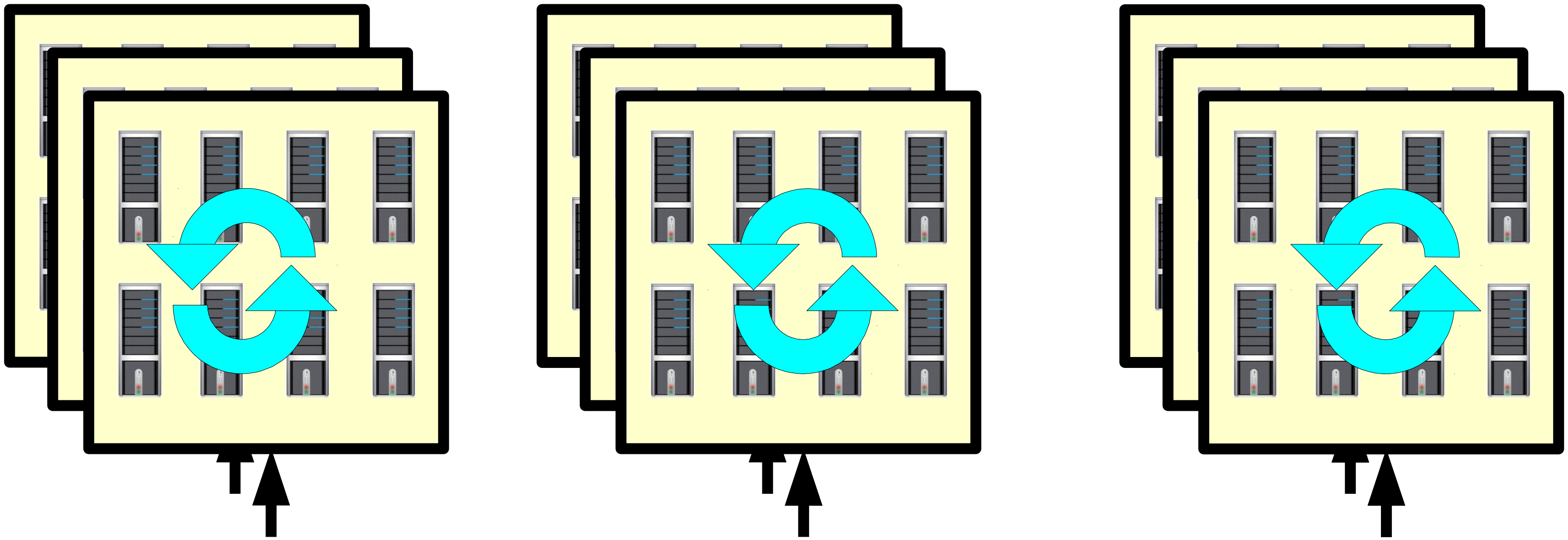


# Asynchronous SGD



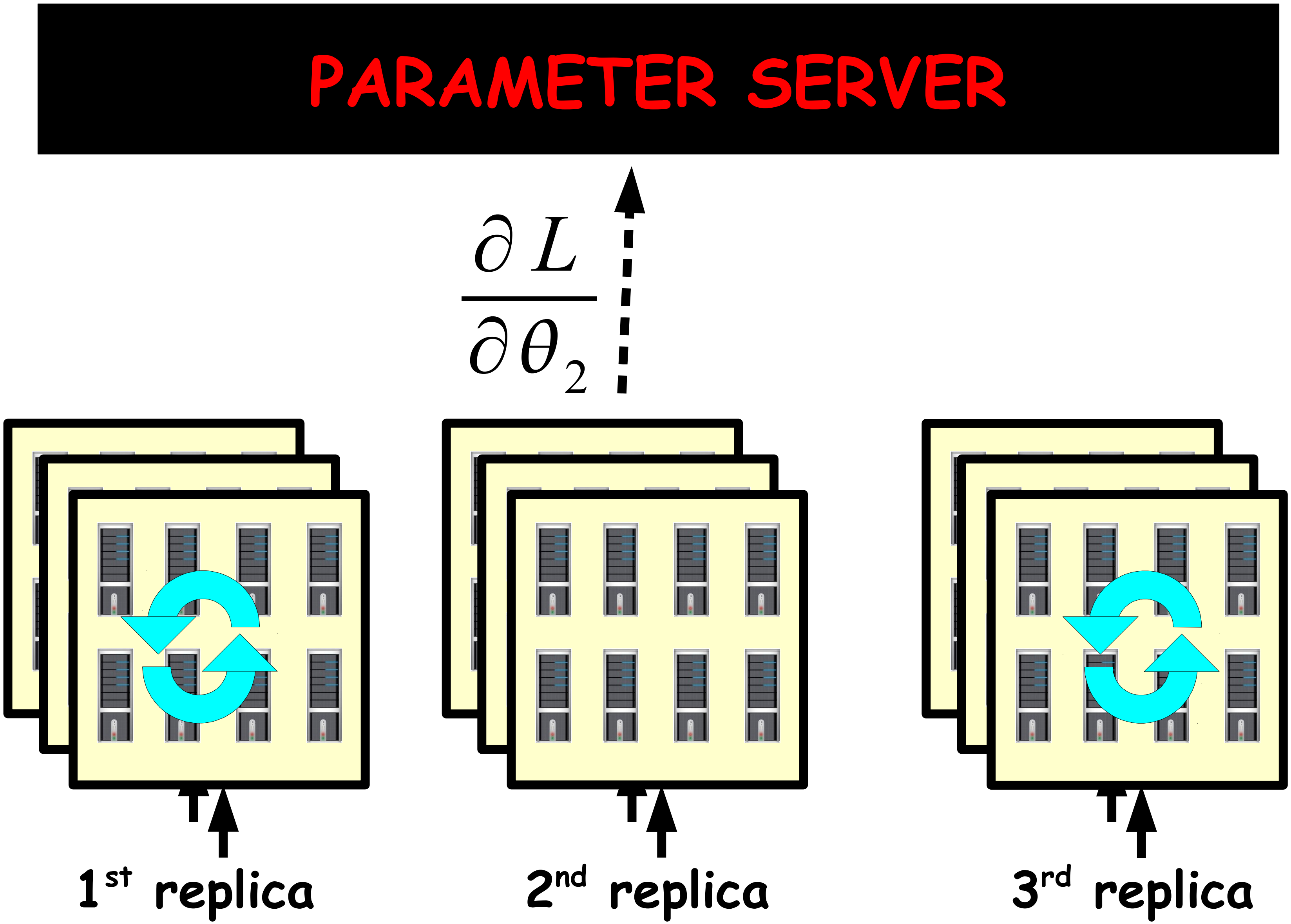
# Asynchronous SGD

**PARAMETER SERVER**
**(update parameters)**

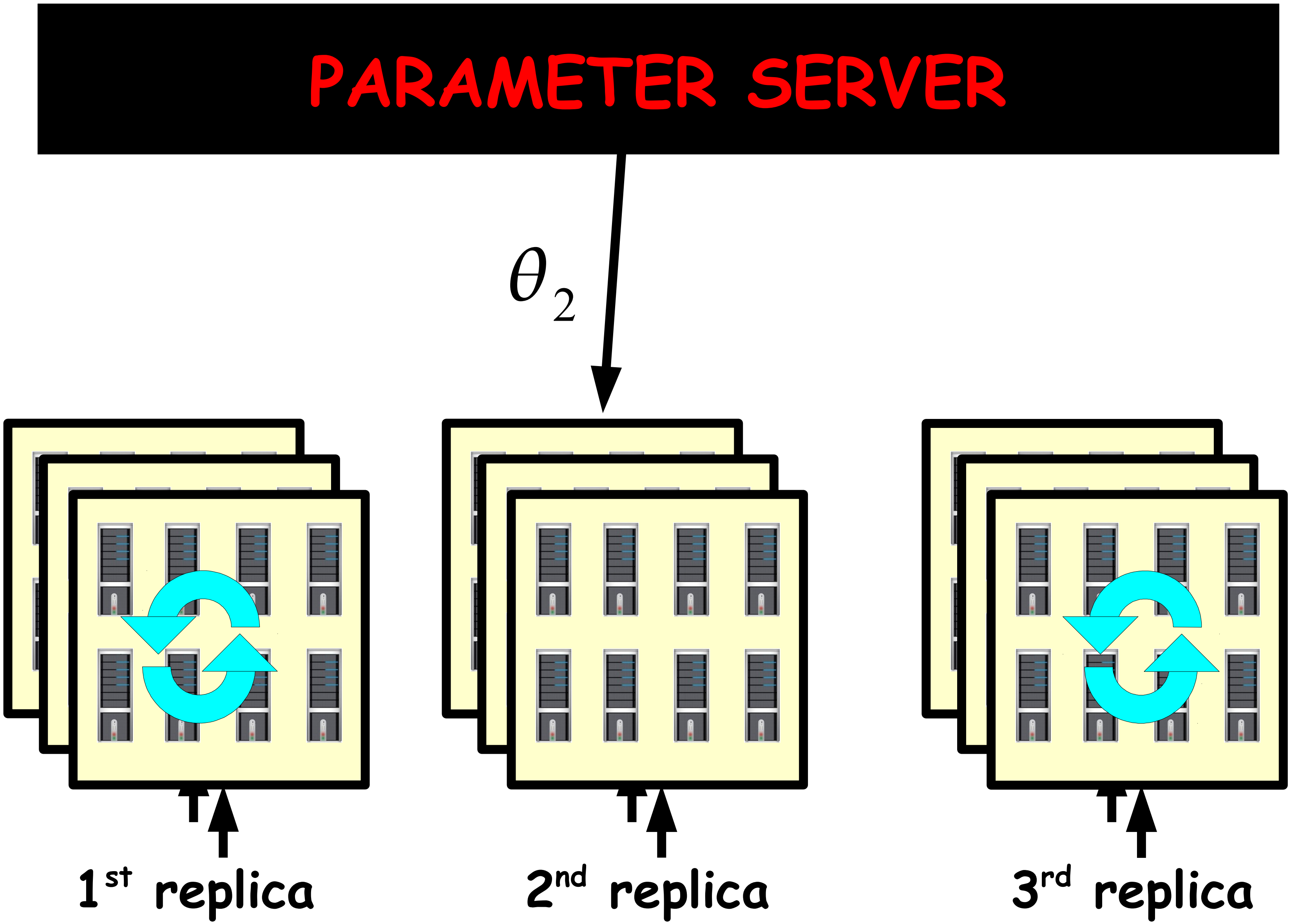


1$^{st}$ replica 2$^{nd}$ replica 3$^{rd}$ replica

# Asynchronous SGD

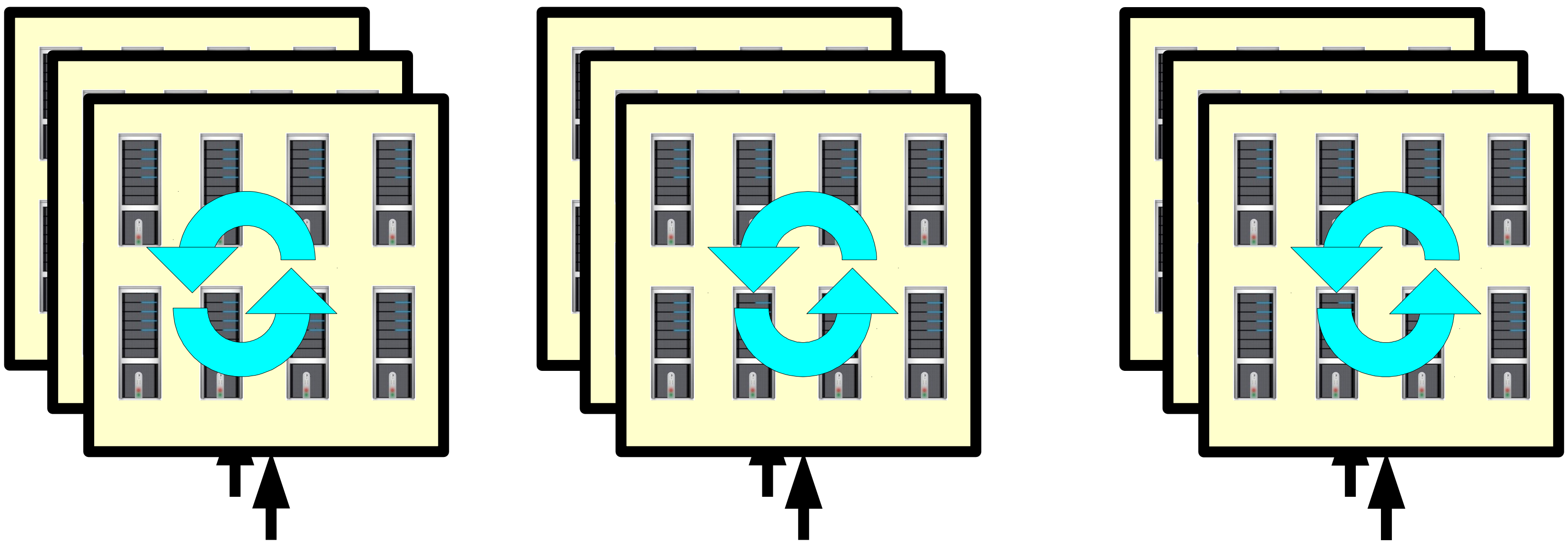


# Asynchronous SGD



# Asynchronous SGD

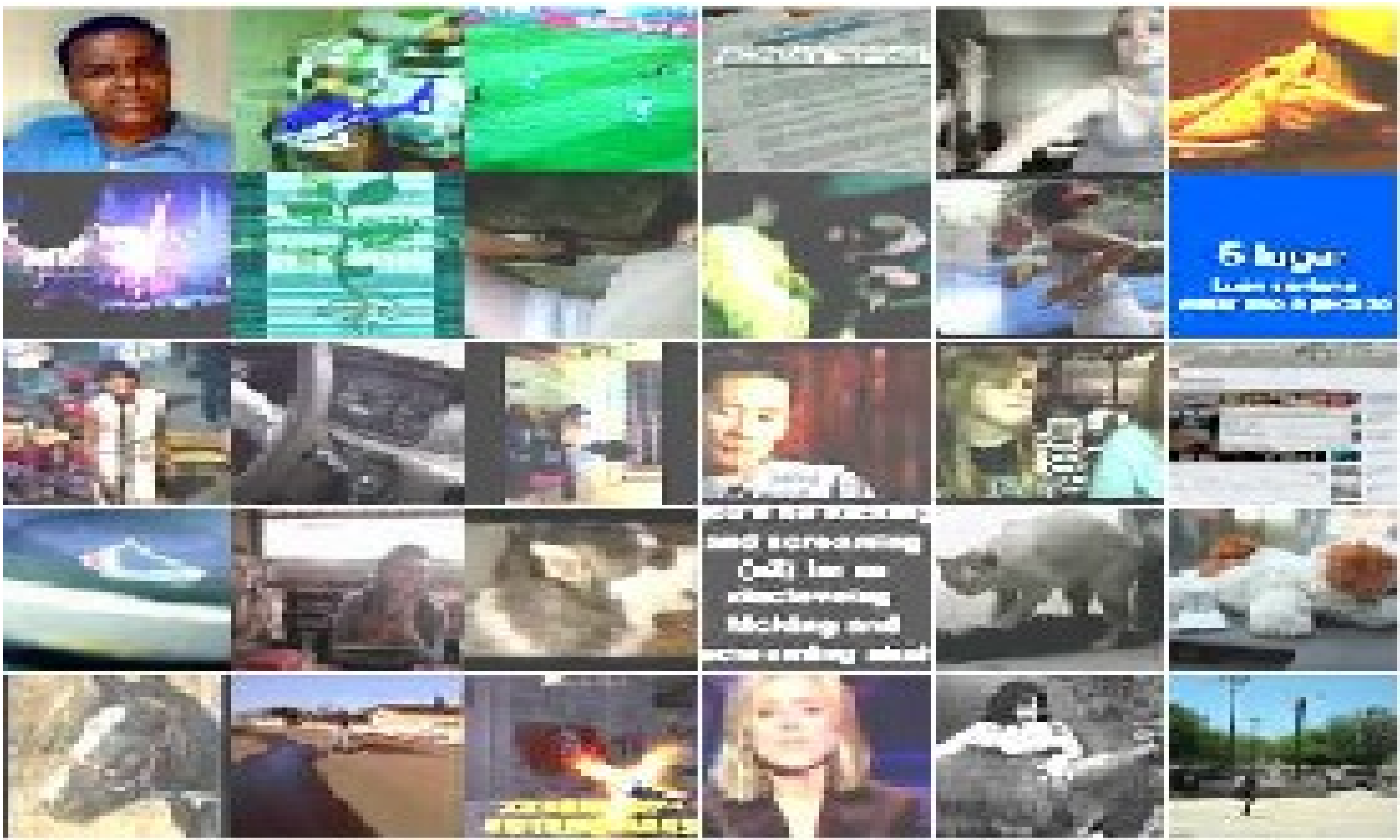
**PARAMETER SERVER**
**(update parameters)**



$1^{st}$ replica $\quad$ $2^{nd}$ replica $\quad$ $3^{rd}$ replica

# Unsupervised Learning With 1B Parameters

**DATA:** 10M youtube (unlabeled) frames of size 200x200.






*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Unsupervised Learning With 1B Parameters

## Deep Net:

- 3 stages
- each stage consists of local filtering, L2 pooling, LCN
  - 18x18 filters
  - 8 filters at each location
  - L2 pooling and LCN over 5x5 neighborhoods
- training jointly the three layers by:
  - reconstructing the input of each layer
  - sparsity on the code



*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Unsupervised Learning With 1B Parameters

**Deep Net:**

- 3 stages
- each stage consists of local filtering, L2 pooling, LCN
  - 18x18 filters
  - 8 filters at each location
  - L2 pooling and LCN over 5x5 neighborhoods
- training jointly the three layers by:
  - reconstructing the input of each layer
  - sparsity on the code

**1B parameters!!!**



*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Validating Unsupervised Learning

The network has seen lots of objects during training, but without any label.

**Q.:** how can we validate unsupervised learning?

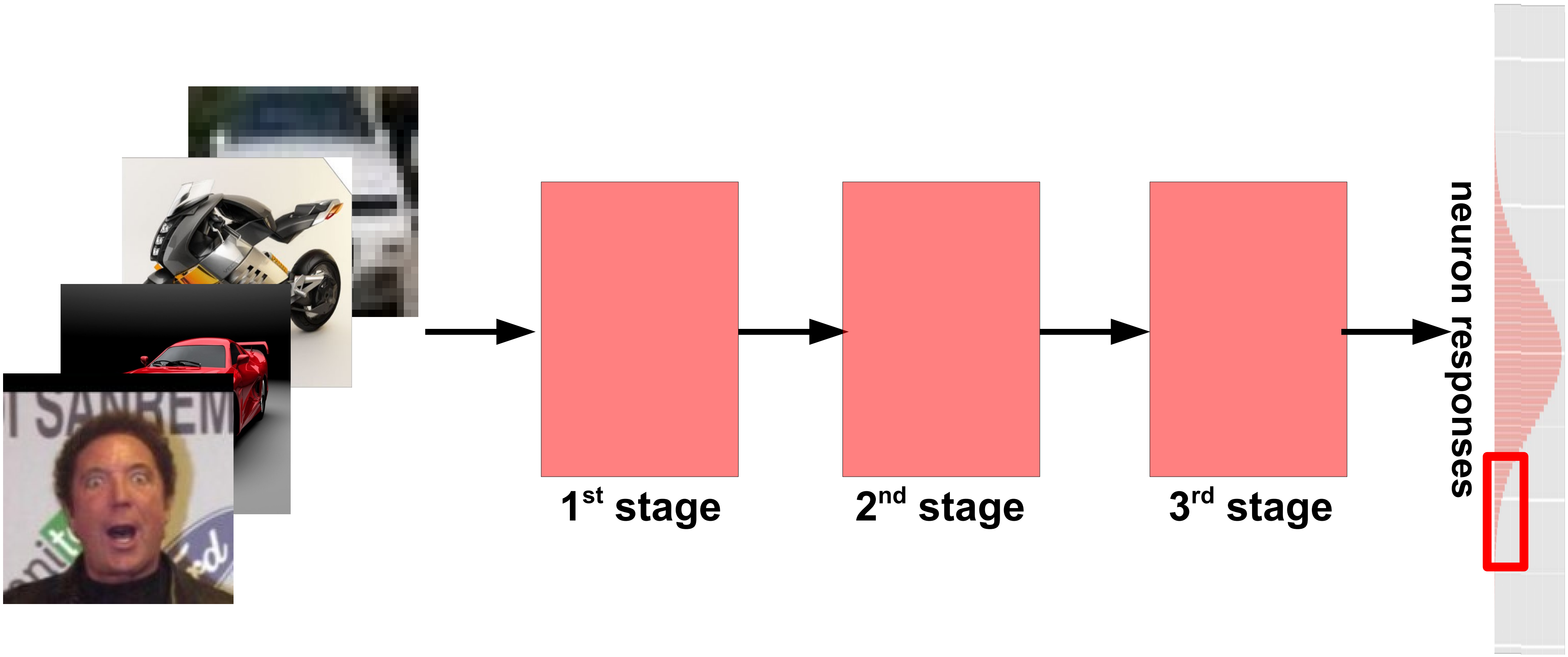
**Q.:** Did the network form any high-level representation?
E.g., does it have any neuron responding for faces?

- build validation set with 50% faces, 50% random images
- study properties of neurons



*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Validating Unsupervised Learning

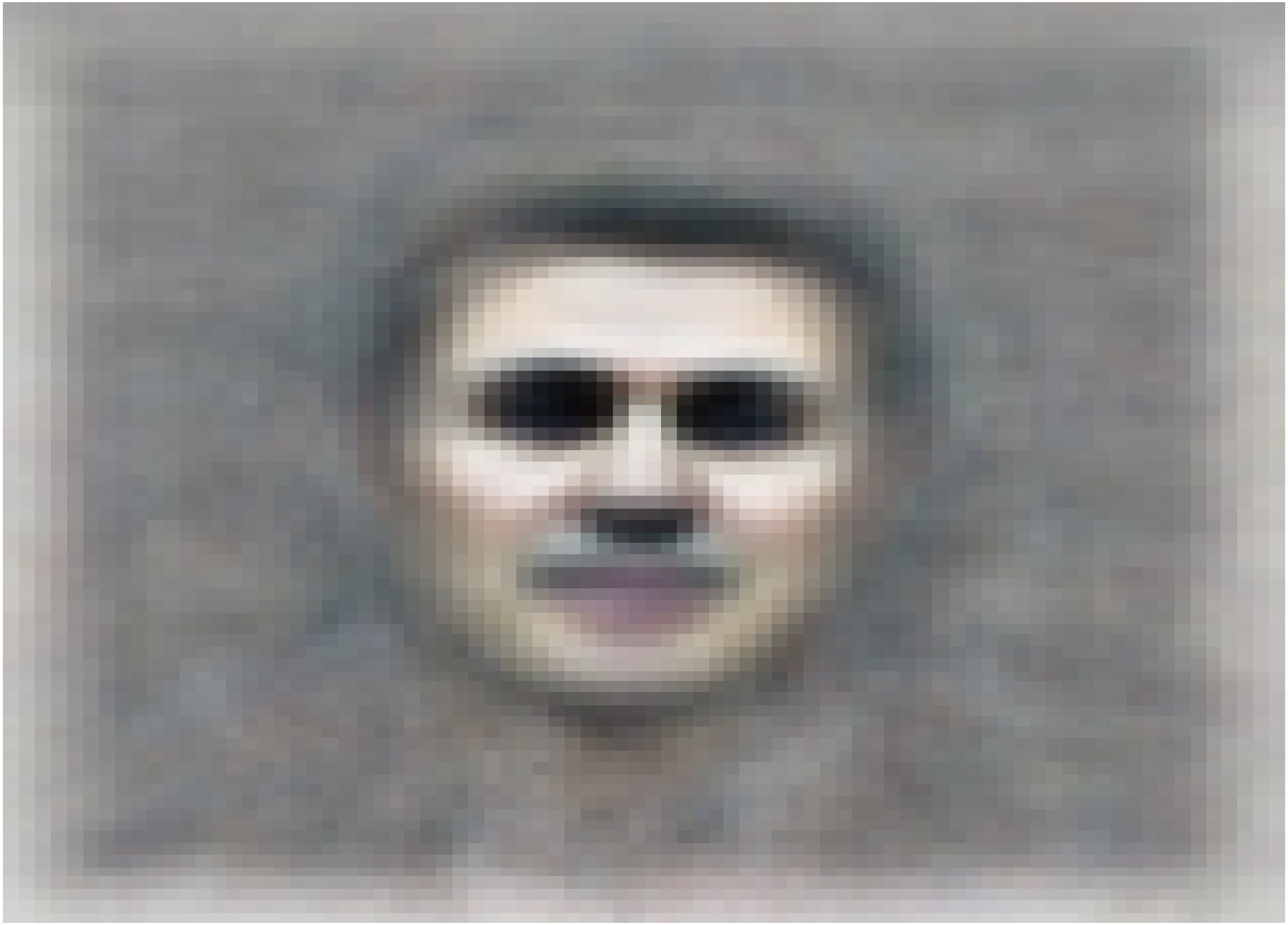





*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Top Images For Best Face Neuron

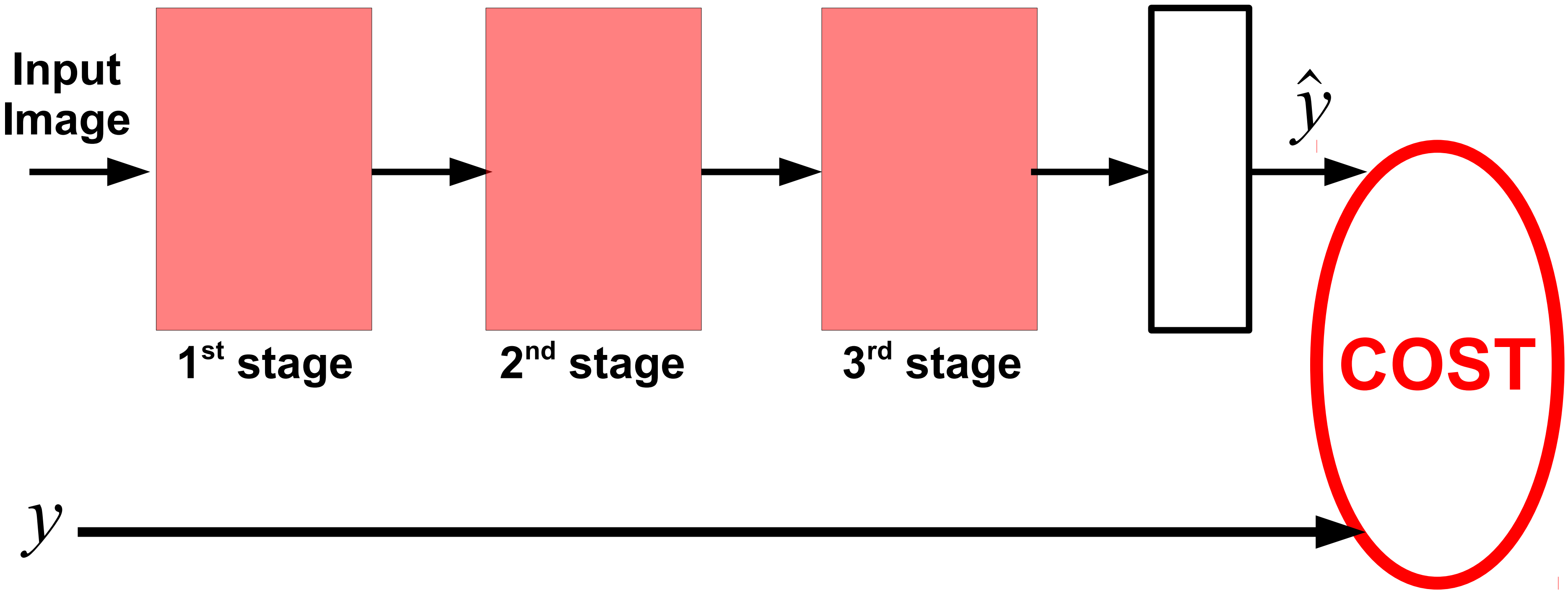
# Best Input For Face Neuron





*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Unsupervised + Supervised (ImageNet)






*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Object Recognition on ImageNet

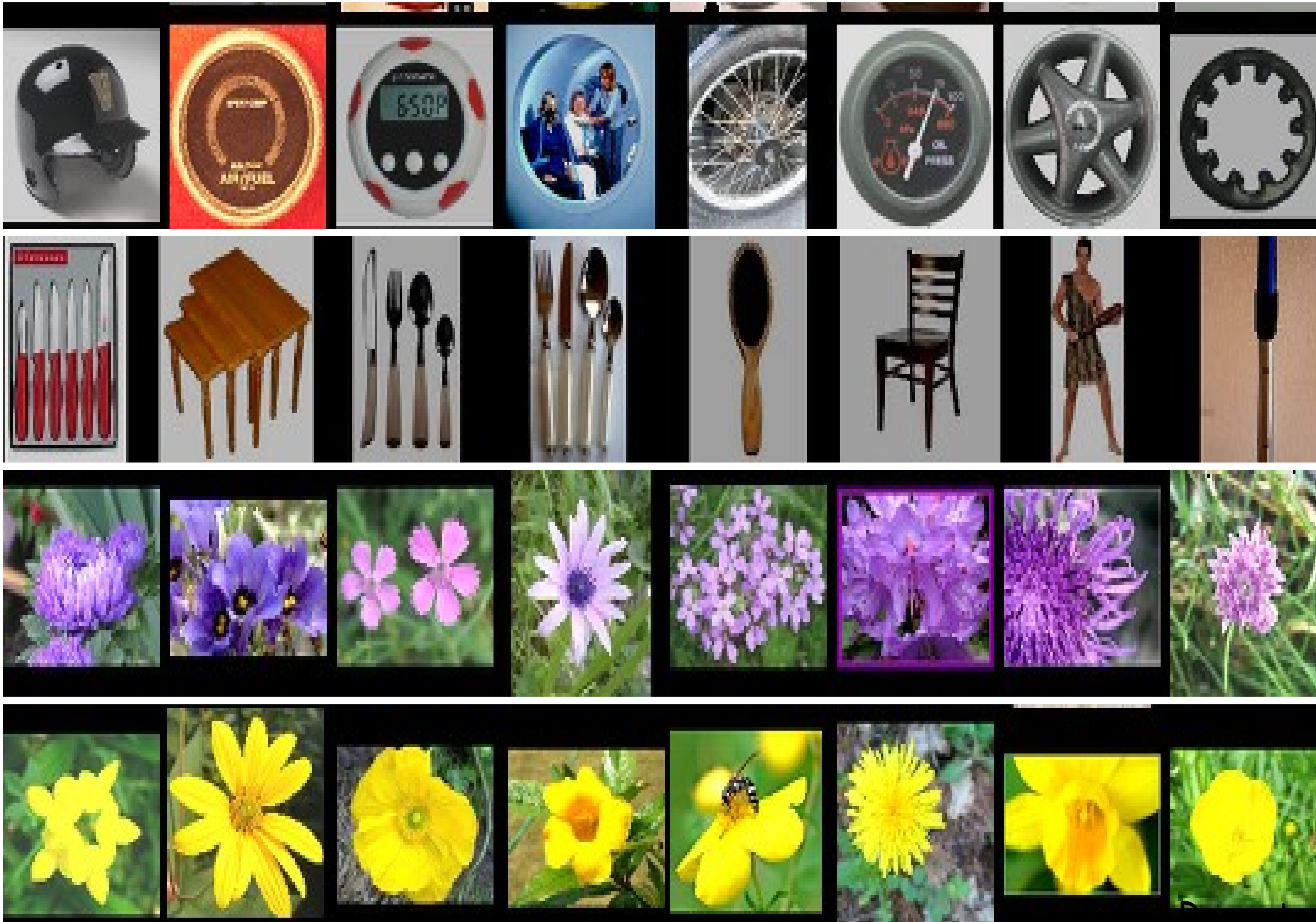
IMAGENET v.2011 (16M images, 20K categories)

| METHOD | ACCURACY % |
|---|---|
| Weston & Bengio 2011 | 9.3 |
| Linear Classifier on deep features | 13.1 |
| Deep Net (from random) | 13.6 |
| Deep Net (from unsup.) | **15.8** |

*Le et al. "Building high-level features using large-scale unsupervised learning" ICML 2012*

# Top Inputs After Supervision

# Top Inputs After Supervision



# Experiments: and many more...

- automatic speech recognition

- natural language processing

- biomed applications

- finance

**Generic learning algorithm!!**

# References

## Tutorials & Background Material


– Yoshua Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009.

– LeCun, Chopra, Hadsell, Ranzato, Huang: A Tutorial on Energy-Based Learning, in Bakir, G. and Hofman, T. and Schölkopf, B. and Smola, A. and Taskar, B. (Eds), Predicting Structured Data, MIT Press, 2006


## Convolutional Nets


– LeCun, Bottou, Bengio and Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

– Jarrett, Kavukcuoglu, Ranzato, LeCun: What is the Best Multi-Stage Architecture for Object Recognition?, Proc. International Conference on Computer Vision (ICCV'09), IEEE, 2009

- Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, LeCun: Learning Convolutional Feature Hierachies for Visual Recognition, Advances in Neural Information Processing Systems (NIPS 2010), 23, 2010

## Unsupervised Learning

- ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning. Le, Karpenko, Ngiam, Ng. In NIPS*2011

- Rifai, Vincent, Muller, Glorot, Bengio, Contracting Auto-Encoders: Explicit invariance during feature extraction, in: Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11), 2011

- Vincent, Larochelle, Lajoie, Bengio, Manzagol, Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, Journal of Machine Learning Research, 11:3371--3408, 2010.

- Gregor, Szlam, LeCun: Structured Sparse Coding via Lateral Inhibition, Advances in Neural Information Processing Systems (NIPS 2011), 24, 2011

- Kavukcuoglu, Ranzato, LeCun. "Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition". ArXiv 1010.3467 2008

- Hinton, Krizhevsky, Wang, Transforming Auto-encoders, ICANN, 2011

## Multi-modal Learning

- Multimodal deep learning, Ngiam, Khosla, Kim, Nam, Lee, Ng. In Proceedings of the Twenty-Eighth International Conference on Machine Learning, 2011.

## Locally Connected Nets

- Gregor, LeCun "Emergence of complex-like cells in a temporal product network with local receptive fields" Arxiv. 2009
- Ranzato, Mnih, Hinton "Generating more realistic images using gated MRF's" NIPS 2010
- Le, Ngiam, Chen, Chia, Koh, Ng "Tiled convolutional neural networks" NIPS 2010

## Distributed Learning

- Le, Ranzato, Monga, Devin, Corrado, Chen, Dean, Ng. "Building High-Level Features Using Large Scale Unsupervised Learning". International Conference of Machine Learning (ICML 2012), Edinburgh, 2012.

## Papers on Scene Parsing

- Farabet, Couprie, Najman, LeCun, "Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers", in Proc. of the International Conference on Machine Learning (ICML'12), Edinburgh, Scotland, 2012.

## Papers on Segmentation

- Turaga, Briggman, Helmstaedter, Denk, Seung Maximin learning of image segmentation. NIPS, 2009.

## Papers on Object Recognition

- Boureau, Le Roux, Bach, Ponce, LeCun: Ask the locals: multi-way local pooling for image recognition, Proc. International Conference on Computer Vision 2011
- Sermanet, LeCun: Traffic Sign Recognition with Multi-Scale Convolutional Networks, Proceedings of International Joint Conference on Neural Networks (IJCNN'11)
- Ciresan, Meier, Gambardella, Schmidhuber. Convolutional Neural Network Committees For Handwritten Character Classification. 11th International Conference on Document Analysis and Recognition (ICDAR 2011), Beijing, China.
- Ciresan, Meier, Masci, Gambardella, Schmidhuber. Flexible, High Performance Convolutional Neural Networks for Image Classification. International Joint Conference on Artificial Intelligence IJCAI-2011.

## Papers on Action Recognition

- Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis, Le, Zou, Yeung, Ng. In Computer Vision and Pattern Recognition (CVPR), 2011

## Papers on Vision for Robotics

- Hadsell, Sermanet, Scoffier, Erkan, Kavackuoglu, Muller, LeCun: Learning Long-Range Vision for Autonomous Off-Road Driving, Journal of Field Robotics, 26(2):120-144, February 2009,

## Deep Convex Nets & Deconv-Nets

- Deng, Yu. “Deep Convex Network: A Scalable Architecture for Speech Pattern Classification.” Interspeech, 2011.
- Zeiler, Taylor, Fergus "Adaptive Deconvolutional Networks for Mid and High Level Feature Learning." ICCV. 2011

## Papers on Biological Inspired Vision

- Serre, Wolf, Bileschi, Riesenhuber, Poggio. Robust Object Recognition with Cortex-like Mechanisms, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29, 3, 411-426, 2007.
- Pinto, Doukhan, DiCarlo, Cox "A high-throughput screening approach to discovering good forms of biologically inspired visual representation." {PLoS} Computational Biology. 2009

# Software & Links

## Deep Learning website

- http://deeplearning.net/

## C++ code for ConvNets

- http://eblearn.sourceforge.net/

## Matlab code for R-ICA unsupervised algorithm

- http://ai.stanford.edu/~quocle/rica_release.zip

## Python-based learning library

- http://deeplearning.net/software/theano/

## Lush learning library which includes ConvNets

- http://lush.sourceforge.net/

## Code used to generate demo for this tutorial

- http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

# Acknowledgements

Quoc Le, Andrew Ng

Jeff Dean, Kai Chen, Greg Corrado, Matthieu Devin, Mark Mao, Rajat Monga, Paul Tucker, Samy Bengio

Yann LeCun, Pierre Sermanet, Clement Farabet