Supervised and Unsupervised Methods for Learning Invariant Feature Hierarchies

Yann LeCun
The Courant Institute of Mathematical Sciences
New York University
Collaborators:
Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, Sumit Chopra

See: [LeCun et al. 2006]: "A Tutorial on Energy-Based Learning"

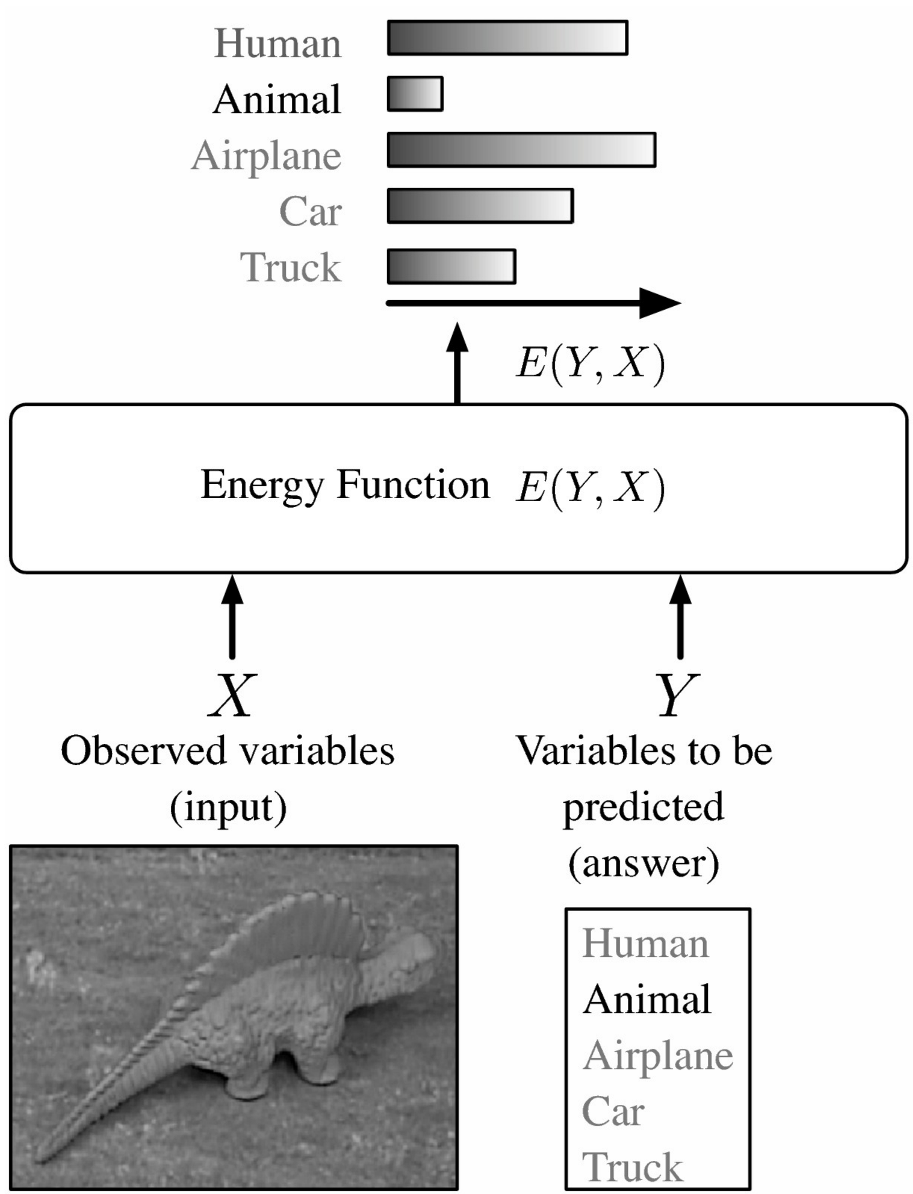
[Ranzato et al. AI-Stats 2007], [Ranzato et al. NIPS 2006]

http://yann.lecun.com/exdb/publis/

Two Big Problems in Machine Learning

- **1. The "Intractable Partition Function Problem"**
  - Give high probability (or low energy) to good answers
  - Give low probability (or high energy) to bad answers
  - There are too many bad answers!
  - The normalization constant of probabilistic models is a sum over too many terms.
- **2. The "Deep Learning Problem"**
  - Training "Deep Belief Networks" is a necessary step towards solving the invariance problem in vision (and perception in general).
  - How do we train deep architectures with lots of non-linear stages?
- **This talks addresses those two problems:**
  - The partition function problem arises with probabilistic approaches. Non-probabilistic **Energy-Based Models** may allow us to get around it.
  - How far can we go with traditional deep learning methods (backprop)
  - How unsupervised feature learning can help guide deep learning.

Energy-Based Model for Decision-Making

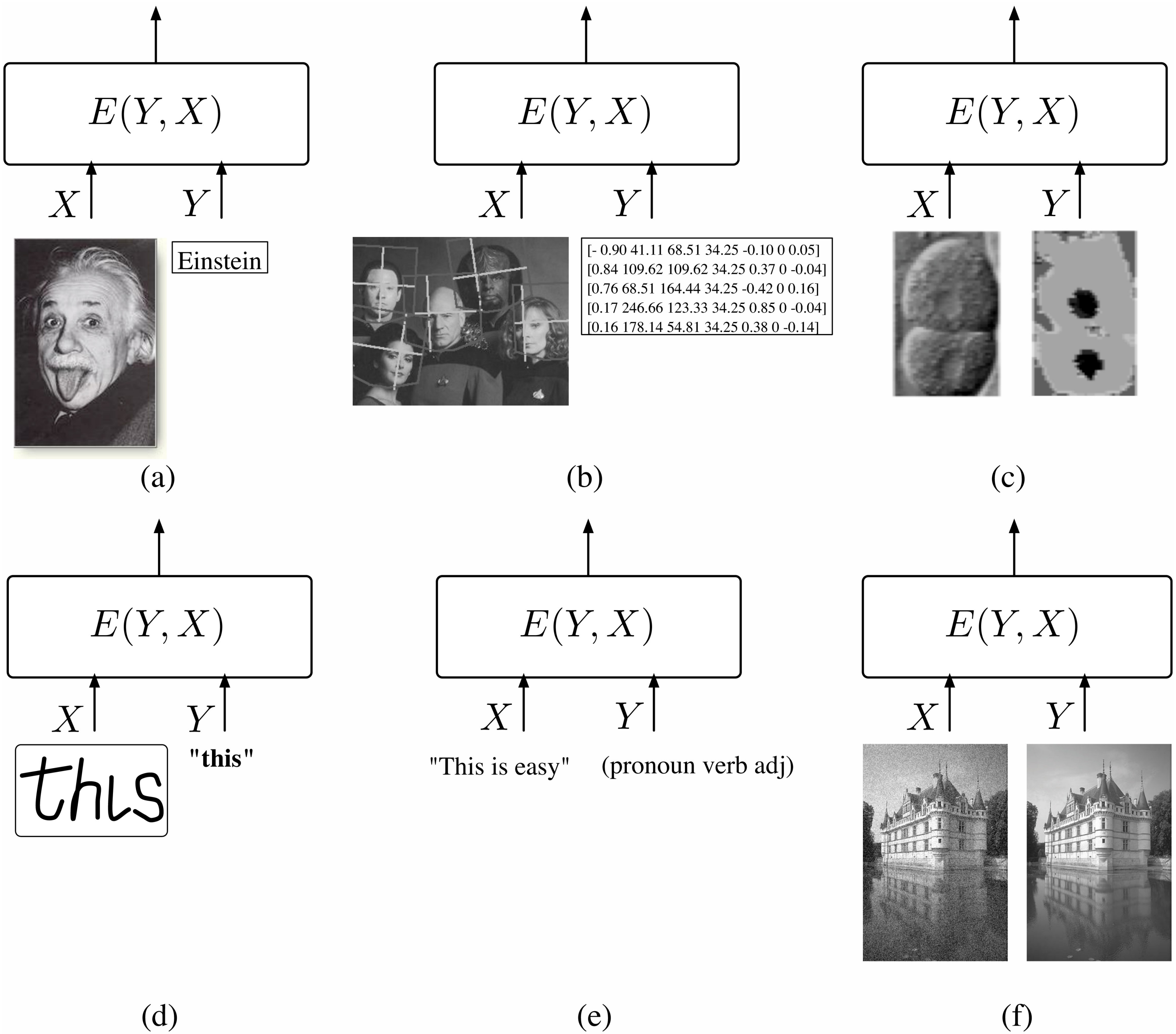


- **Model:** Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$

- **Inference:** Search for the Y that minimizes the energy within a set $\mathcal{Y}$
- **If the set has low cardinality, we can use exhaustive search.**

Complex Tasks: Inference is non-trivial

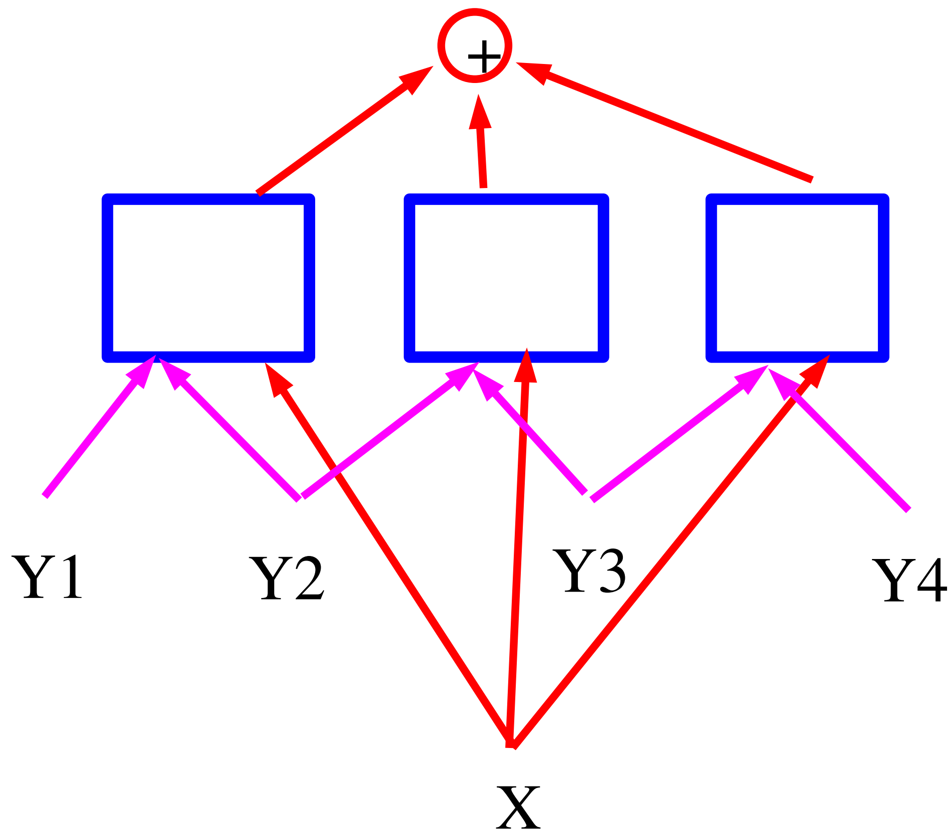


- **When the cardinality or dimension of Y is large, exhaustive search is impractical.**
- **We need to use "smart" inference procedures: min-sum, Viterbi, min cut, gradient decent.....**

Energy-Based Factor Graphs: Energy = Sum of "factors"

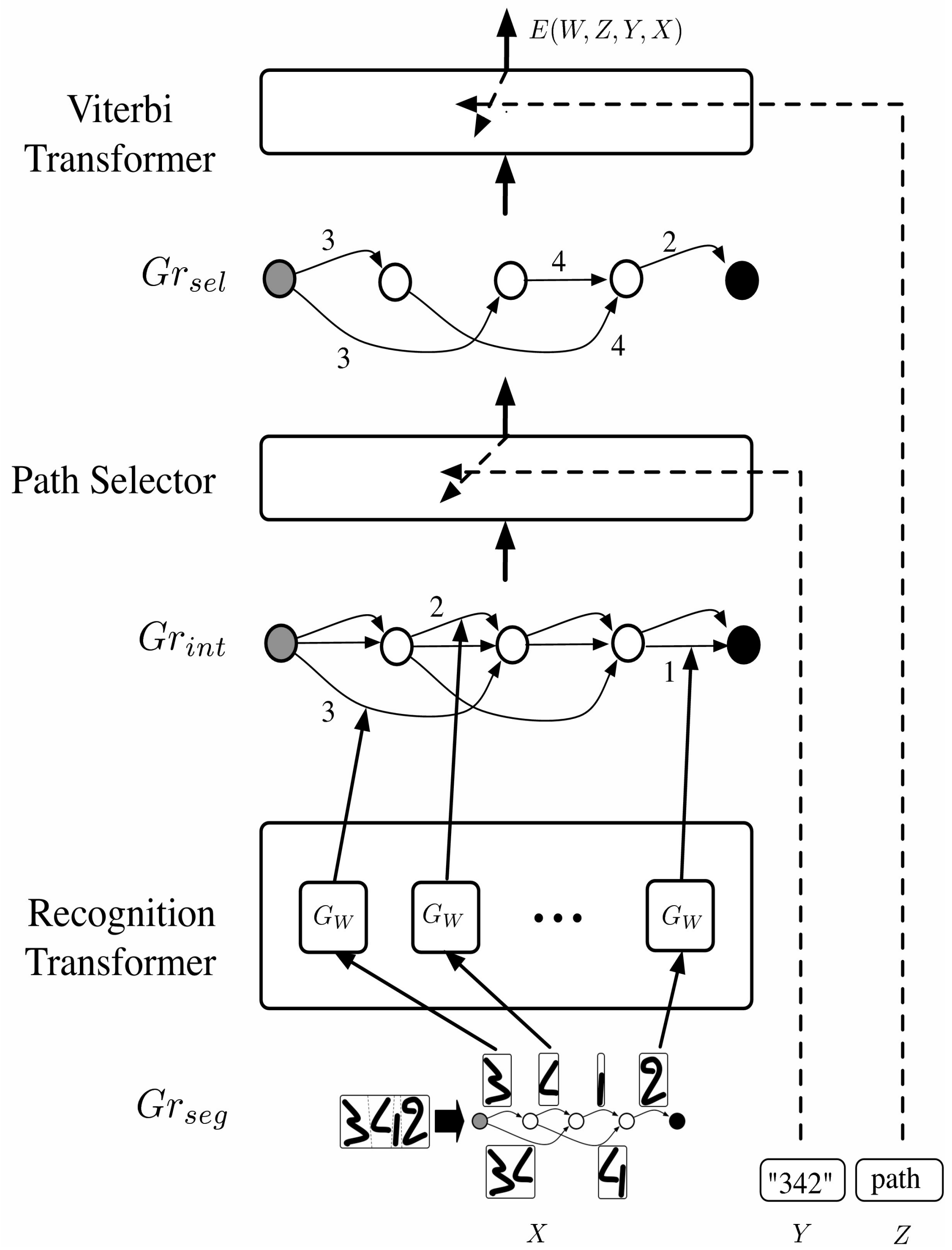
- **Sequence Labeling**
  - Output is a sequence Y1,Y2,Y3,Y4......
  - NLP parsing, MT, speech/handwriting recognition, biological sequence analysis
  - The factors ensure grammatical consistency
  - They give low energy to consistent sub-sequences of output symbols
  - The graph is generally simple (chain or tree)
  - Inference is easy (dynamic programming, min-sum)

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}, Z \in \mathcal{Z}} E(Z, Y, X).$$



Sequence Labeling, OCR

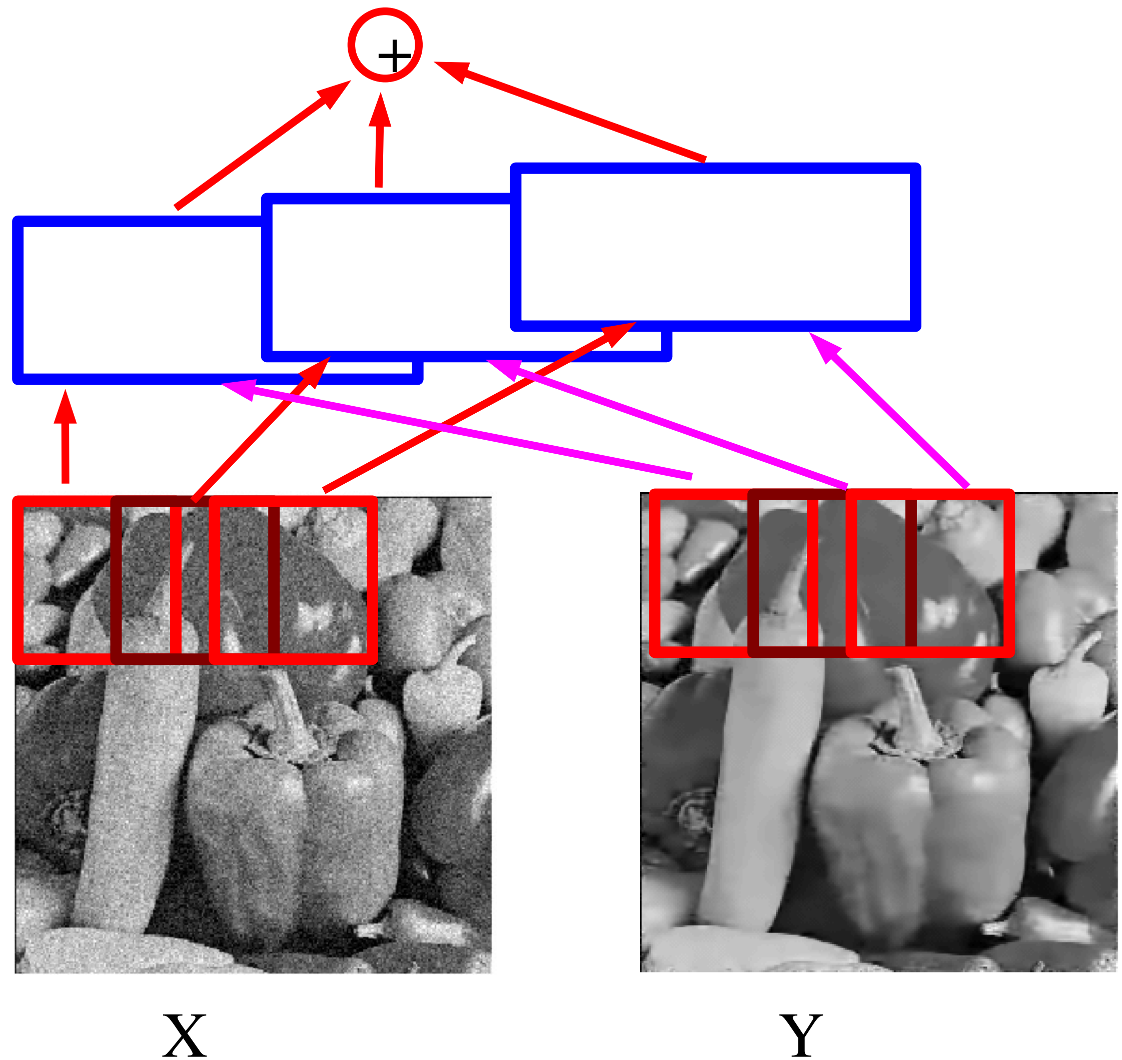
- **integrated segmentation and recognition of sequences.**
- **Each segmentation and recognition hypothesis is a path in a graph**
- **inference = finding the shortest path in the interpretation graph.**
- **Un-normalized hierarchical HMMs a.k.a. Graph Transformer Networks**
  - [LeCun, Bottou, Bengio, Haffner 1998]



Energy-Based Factor Graphs: complex/loopy graphs

- **Image restoration**
  - The factors ensure local consistency on small overlapping patches
  - They give low energy to "clean" patches, given the noisy versions
  - The graph is loopy when the patches overlap.
  - Inference is difficult, particularly when the patches are large,and when the number of greyscale values is large

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$



What Questions Can a Model Answer?

1. Classification & Decision Making:

- "which value of Y is most compatible with X?"
- Applications: Robot navigation,.....
- Training: give the lowest energy to the correct answer

2. Ranking:

- "Is Y1 or Y2 more compatible with X?"
- Applications: Data-mining....
- Training: produce energies that rank the answers correctly

3. Conditional Density Estimation:

- "What is the conditional distribution P(Y|X)?"
- Application: feeding a decision-making system
- Training: differences of energies must be just so.

Decision-Making versus Probabilistic Modeling

Energies are uncalibrated

- The energies of two separately-trained systems cannot be combined
- The energies are uncalibrated (measured in arbitrary untis)

How do we calibrate energies?

- We turn them into probabilities (positive numbers that sum to 1).
- Simplest way: Gibbs distribution
- Other ways can be reduced to Gibbs by a suitable redefinition of the energy.

$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}},$$

Partition function

Inverse temperature

Architecture and Loss Function

- **Family of energy functions** $\mathcal{E} = \{E(W,Y,X) : \ W \in \mathcal{W}\}.$
- **Training set** $\mathcal{S} = \{(X^i, Y^i) : i = 1 \dots P\}$
- **Loss functional / Loss function** $\mathcal{L}(E,\mathcal{S})$ $\mathcal{L}(W,\mathcal{S})$
  - Measures the quality of an energy function on training set
- **Training** $W^* = \min_{W \in \mathcal{W}} \mathcal{L}(W,\mathcal{S}).$
- **Form of the loss functional**
  - invariant under permutations and repetitions of the samples

$$\mathcal{L}(E,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P} L(Y^i, E(W,\mathcal{Y},X^i)) + R(W).$$

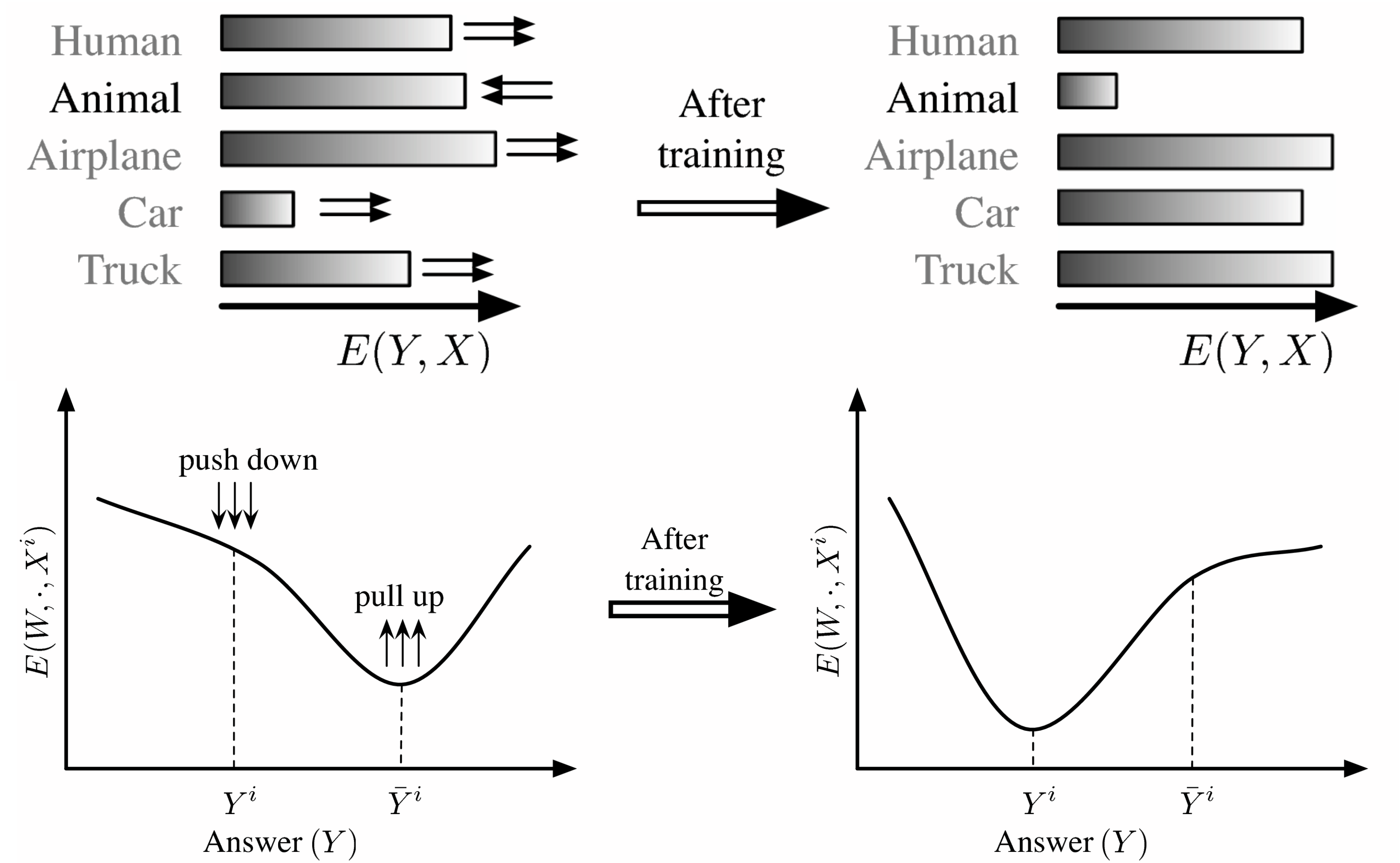
Per-sample loss

Desired answer

Energy surface for a given Xi as Y varies

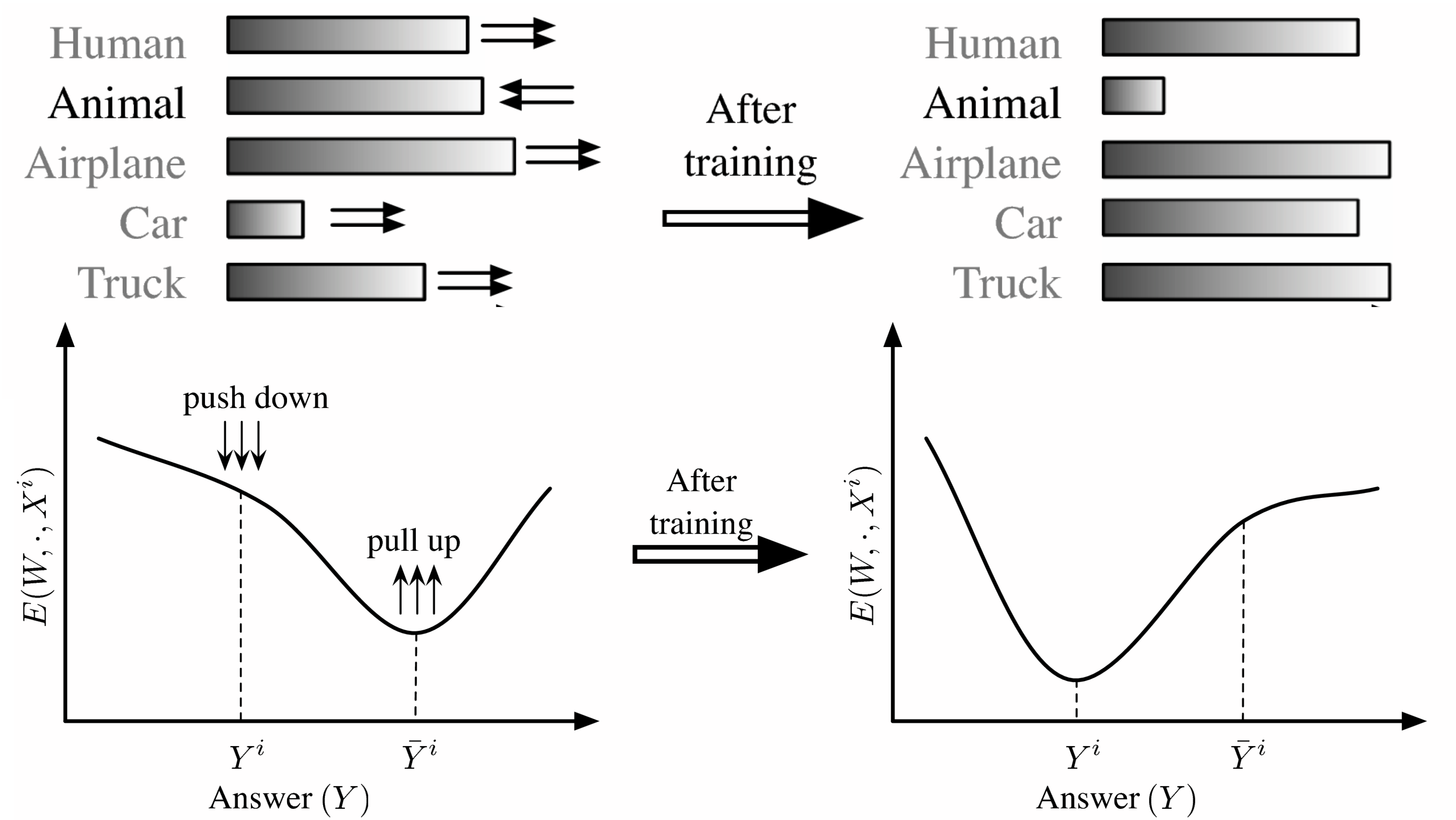
Regularizer

Designing a Loss Functional



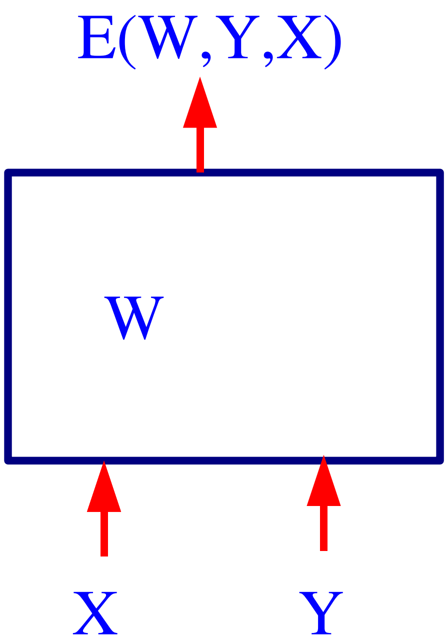
- **Correct answer has the lowest energy -> LOW LOSS**
- **Lowest energy is not for the correct answer -> HIGH LOSS**

Designing a Loss Functional



- **Push down on the energy of the correct answer**
- **Pull up on the energies of the incorrect answers, particularly if they are smaller than the correct one**

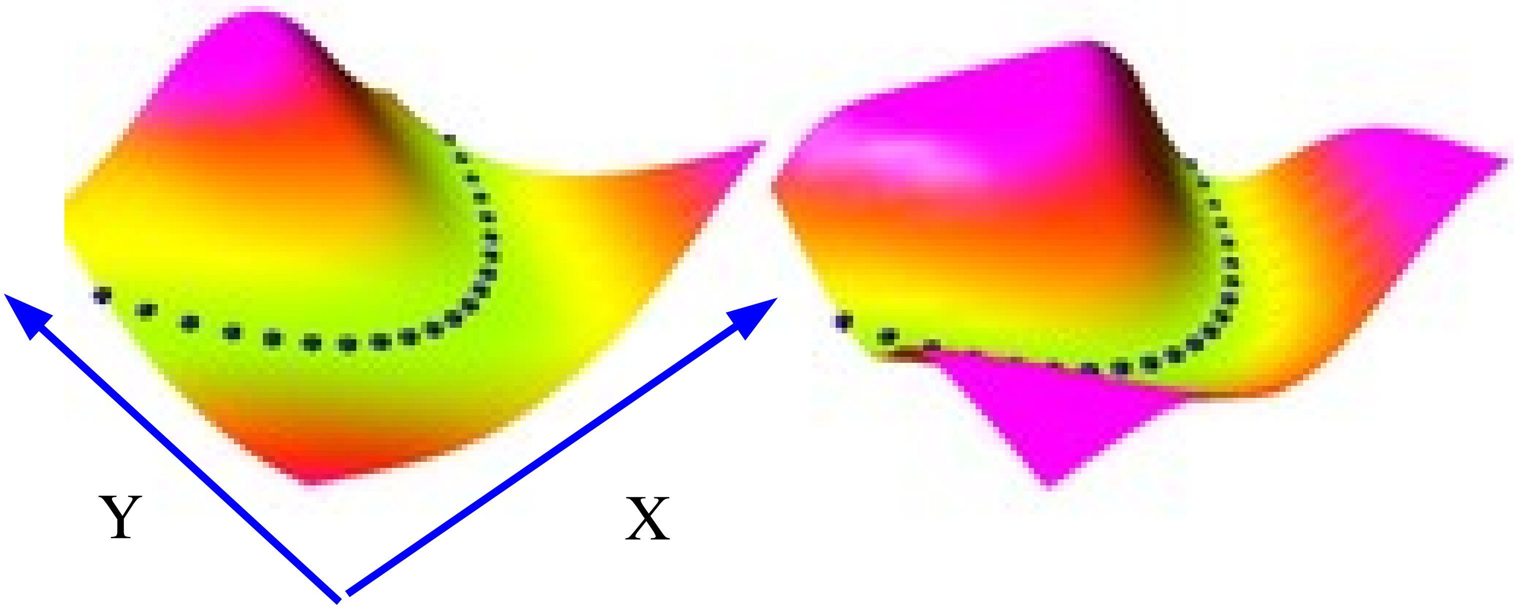
Architecture + Inference Algo + Loss Function = Model



- **1. Design an architecture:** a particular form for E(W,Y,X).
- **2. Pick an inference algorithm for Y:** MAP or conditional distribution, belief prop, min cut, variational methods, gradient descent, MCMC, HMC.....
- **3. Pick a loss function:** in such a way that minimizing it with respect to W over a training set will make the inference algorithm find the correct Y for a given X.
- **4. Pick an optimization method.**

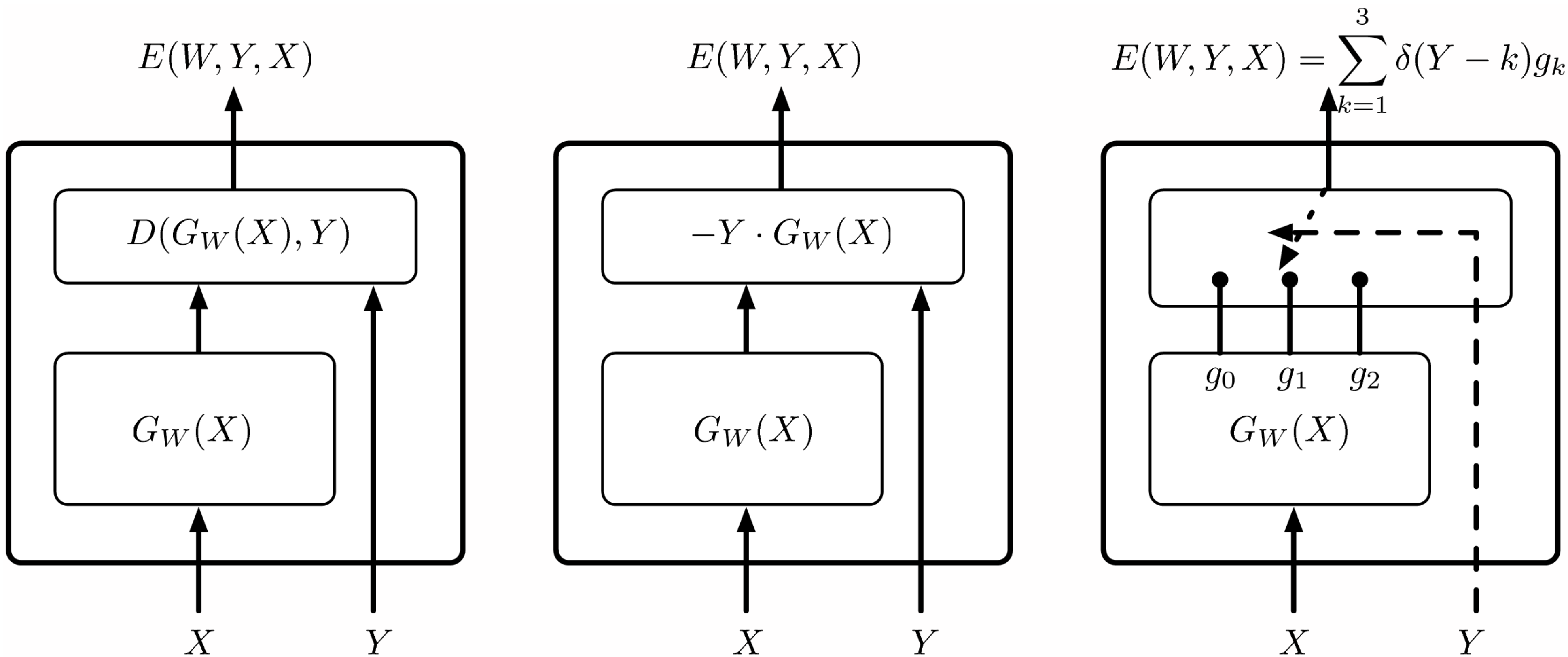
- **PROBLEM: What loss functions will make the machine approach the desired behavior?**

Several Energy Surfaces can give the same answers



- Both surfaces compute Y=X^2
- MINy E(Y,X) = X^2
- Minimum-energy inference gives us the same answer

Simple Architectures



- **Regression**

$$E(W, Y, X) = \frac{1}{2}||G_W(X) - Y||^2.$$

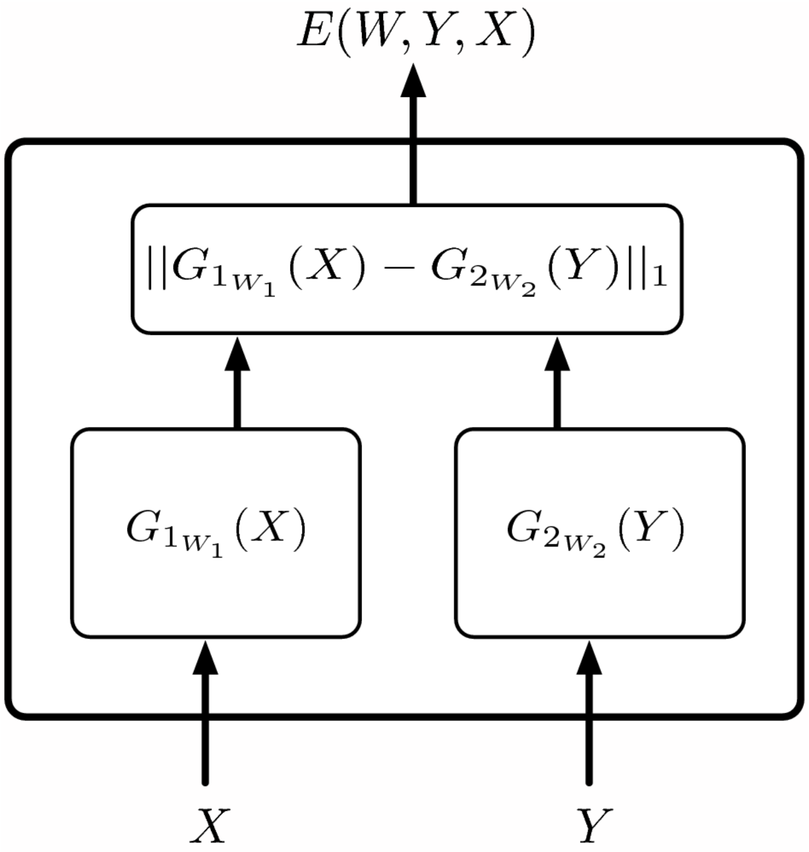
- **Binary Classification**

$$E(W, Y, X) = -Y G_W(X),$$

- **Multi-class Classification**

Simple Architecture: Implicit Regression

$$E(W,X,Y) = ||G_{1_{W_1}}(X) - G_{2_{W_2}}(Y)||_1,$$



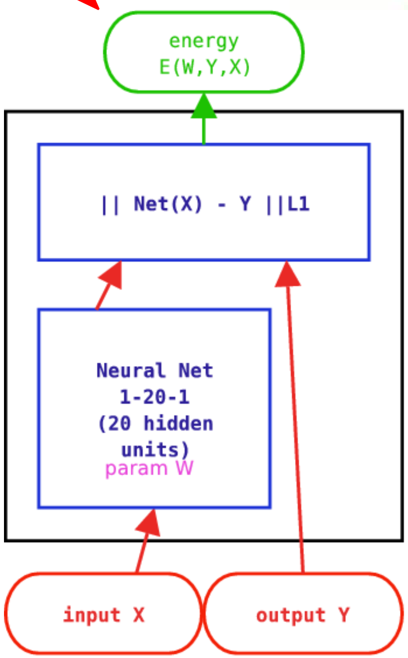
The Implicit Regression architecture

- allows multiple answers to have low energy.
- Encodes a constraint between X and Y rather than an explicit functional relationship
- This is useful for many applications
- Example: sentence completion: "The cat ate the {mouse,bird,homework,...}"
- [Bengio et al. 2003]
- But, inference may be difficult.

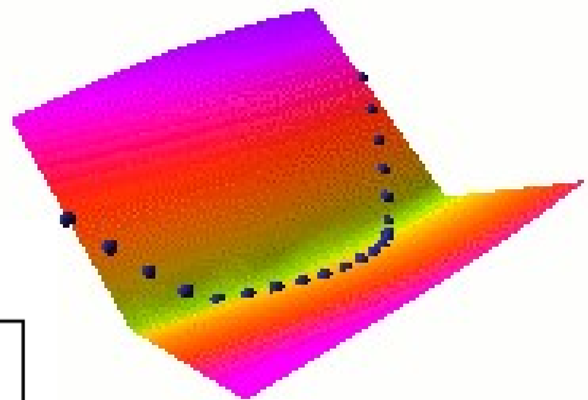
Examples of Loss Functions: Energy Loss

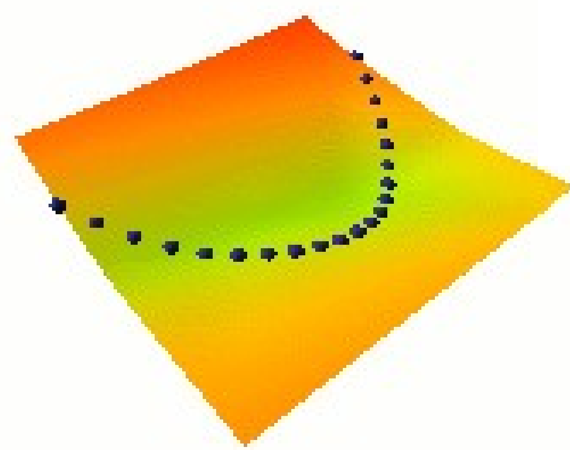
- **Energy Loss** $L_{energy}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i).$
  - Simply pushes down on the energy of the correct answer

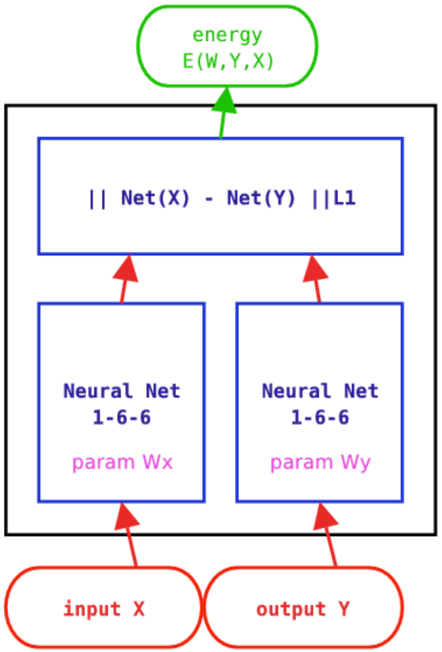
WORKS!!!



(a)







(b)

COLLAPSES!!!

Examples of Loss Functions:Perceptron Loss

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i).$$

- **Perceptron Loss [LeCun et al. 1998], [Collins 2002]**
  - Pushes down on the energy of the correct answer
  - Pulls up on the energy of the machine's answer
  - Always positive. Zero when answer is correct
  - No “margin”: technically does not prevent the energy surface from being almost flat.
  - Works pretty well in practice, particularly if the energy parameterization does not allow flat surfaces.

Perceptron Loss for Binary Classification

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i).$$

- **Energy:** $E(W, Y, X) = -Y G_W(X),$
- **Inference:** $Y^* = \text{argmin}_{Y \in \{-1,1\}} - Y G_W(X) = \text{sign}(G_W(X)).$
- **Loss:** $\mathcal{L}_{\text{perceptron}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \left(\text{sign}(G_W(X^i)) - Y^i\right) G_W(X^i).$
- **Learning Rule:** $W \leftarrow W + \eta \left(Y^i - \text{sign}(G_W(X^i))\right) \frac{\partial G_W(X^i)}{\partial W},$
- **If Gw(X) is linear in W:** $E(W, Y, X) = -Y W^T \Phi(X)$

$$W \leftarrow W + \eta \left(Y^i - \text{sign}(W^T \Phi(X^i))\right) \Phi(X^i)$$

Examples of Loss Functions: Generalized Margin Losses

- First, we need to define the Most Offending Incorrect Answer

- Most Offending Incorrect Answer: discrete case

Definition 1 *Let Y be a discrete variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are incorrect:*

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y} and Y \neq Y^i} E(W, Y, X^i). \quad (8)$$

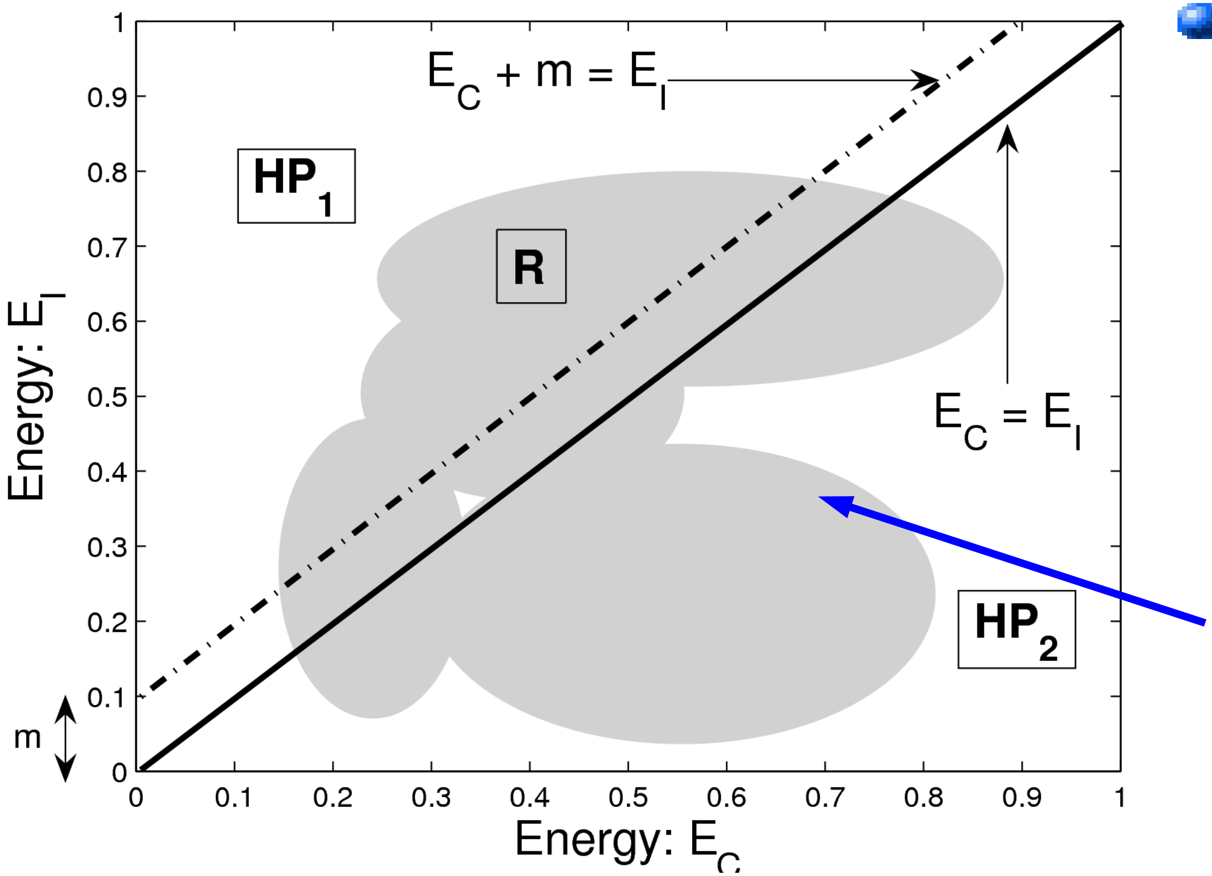
- Most Offending Incorrect Answer: continuous case

Definition 2 *Let Y be a continuous variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are at least ϵ away from the correct answer:*

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \quad (9)$$

Examples of Loss Functions: Generalized Margin Losses

$$L_{\text{margin}}(W, Y^i, X^i) = Q_m\left(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)\right).$$

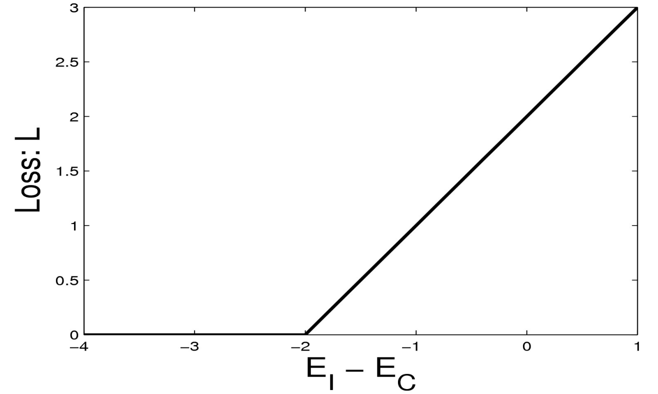


Generalized Margin Loss

- Qm increases with the energy of the correct answer
- Qm decreases with the energy of the most offending incorrect answer
- whenever it is less than the energy of the correct answer plus a margin m.

Examples of Generalized Margin Losses

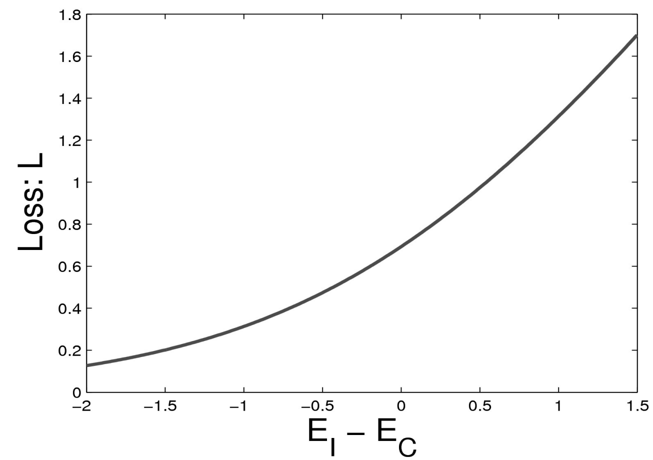
$$L_{\text{hinge}}(W, Y^i, X^i) = \max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right),$$



Hinge Loss

- [Altun et al. 2003], [Taskar et al. 2003
- With the linearly-parameterized binary classifier architecture, we get linear SV

$$L_{\log}(W, Y^i, X^i) = \log\left(1 + e^{E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)}\right).$$



Log Loss

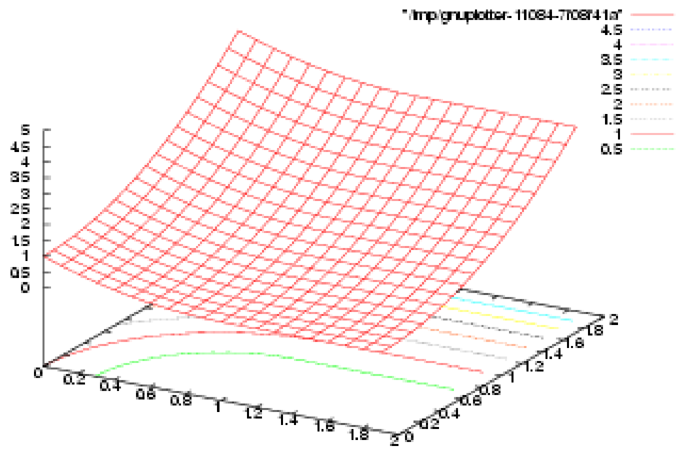
- "soft hinge" loss
- With the linearly-parameterized binary classifier architecture, we get linear Logistic Regression

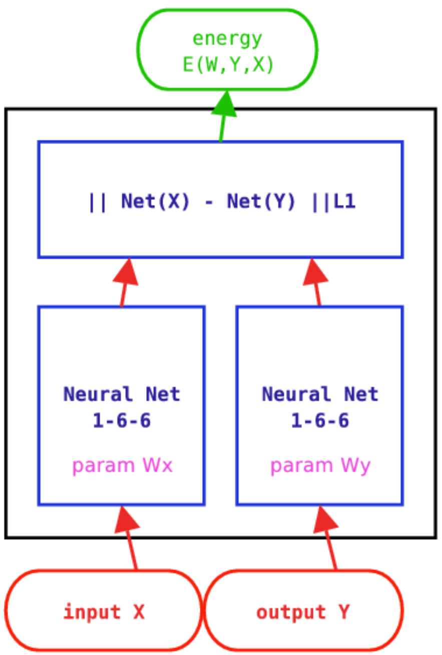
Examples of Margin Losses: Square-Square Loss

$$L_{\text{sq-sq}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2.$$

Square-Square Loss

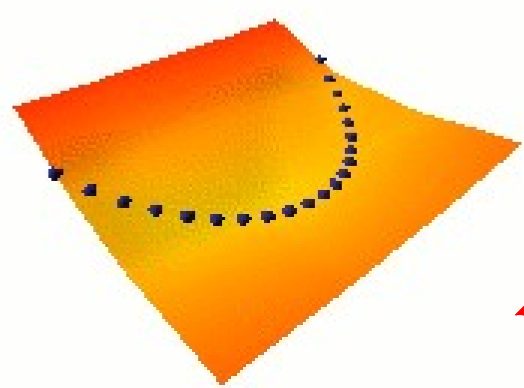
- [LeCun-Huang 2005]
- Appropriate for positive energy functions





(b)

Learning Y = X^2



NO COLLAPSE!!!

Other Margin-Like Losses

- **LVQ2 Loss** [Kohonen, Oja], Driancourt-Bottou 1991]

$$L_{\text{lvq2}}(W, Y^i, X^i) = \min\left(1, \max\left(0, \frac{E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)}{\delta E(W, \bar{Y}^i, X^i)}\right)\right),$$

- **Minimum Classification Error Loss** [Juang, Chou, Lee 1997]

$$L_{\text{mce}}(W, Y^i, X^i) = \sigma\left(E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right),$$
$$\sigma(x) = (1 + e^{-x})^{-1}$$

- **Square-Exponential Loss** [Osadchy, Miller, LeCun 2004]

$$L_{\text{sq-exp}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \gamma e^{-E(W, \bar{Y}^i, X^i)},$$

Negative Log-Likelihood Loss

- **Conditional probability of the samples (assuming independence)**

$$P(Y^1, \ldots, Y^P | X^1, \ldots, X^P, W) = \prod_{i=1}^{P} P(Y^i | X^i, W).$$

$$-\log \prod_{i=1}^{P} P(Y^i | X^i, W) = \sum_{i=1}^{P} -\log P(Y^i | X^i, W).$$

- **Gibbs distribution:** $$P(Y | X^i, W) = \frac{e^{-\beta E(W,Y,X^i)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}}.$$

$$-\log \prod_{i=1}^{P} P(Y^i | X^i, W) = \sum_{i=1}^{P} \beta E(W, Y^i, X^i) + \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}.$$

- **We get the NLL loss by dividing by P and Beta:**

$$\mathcal{L}_{\text{nll}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \left(E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)} \right).$$

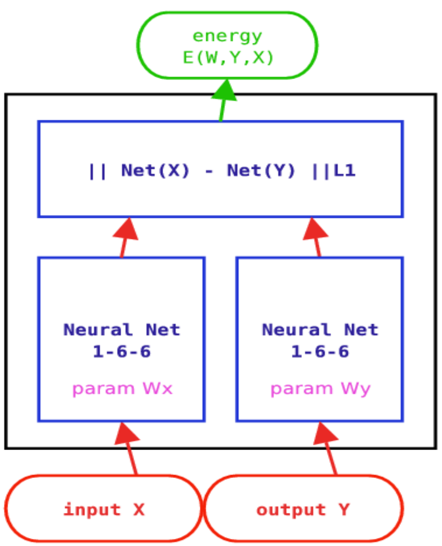
- **Reduces to the perceptron loss when Beta->infinity**

Negative Log-Likelihood Loss

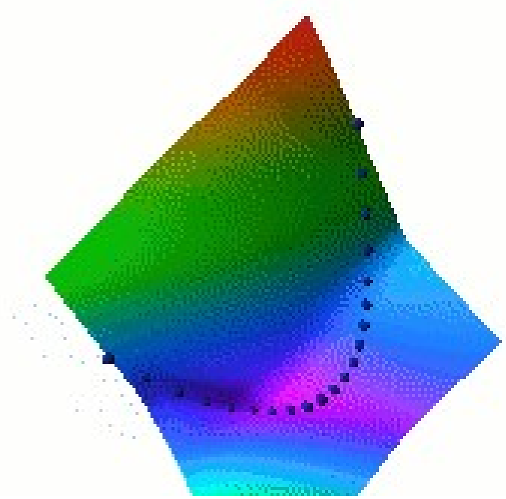
- Pushes down on the energy of the correct answer
- Pulls up on the energies of all answers in proportion to their probability

$$\mathcal{L}_{\text{nll}}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P}\left(E(W,Y^i,X^i) + \frac{1}{\beta}\log\int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)}\right).$$

$$\frac{\partial L_{\text{nll}}(W,Y^i,X^i)}{\partial W} = \frac{\partial E(W,Y^i,X^i)}{\partial W} - \int_{Y\in\mathcal{Y}} \frac{\partial E(W,Y,X^i)}{\partial W} P(Y|X^i,W),$$



(b)



Negative Log-Likelihood Loss: Binary Classification

- **Binary Classifier Architecture:**

$$\mathcal{L}_{\text{nll}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \left[-Y^i G_W(X^i) + \log \left(e^{Y^i G_W(X^i)} + e^{-Y^i G_W(X^i)} \right) \right].$$

$$\mathcal{L}_{\text{nll}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \log \left(1 + e^{-2Y^i G_W(X^i)} \right),$$

- **Linear Binary Classifier Architecture:**

$$\mathcal{L}_{\text{nll}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \log \left(1 + e^{-2Y^i W^T \Phi(X^i)} \right).$$

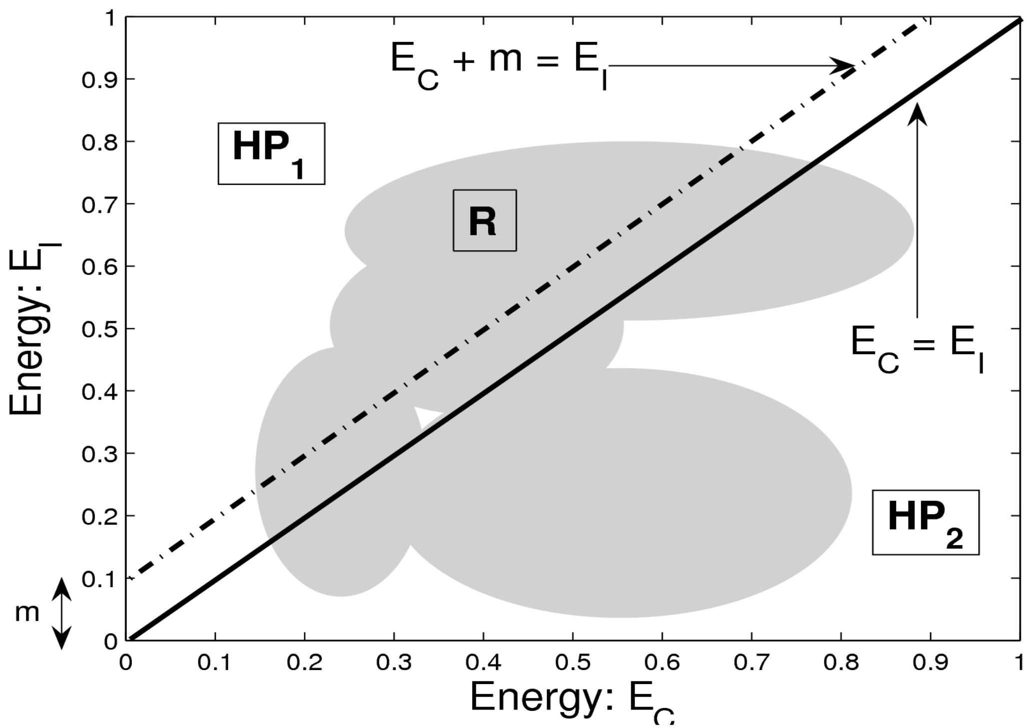
- **Learning Rule: logistic regression**

Negative Log-Likelihood Loss

- **A probabilistic model is an EBM in which:**
  - The energy can be integrated over Y (the variable to be predicted)
  - The loss function is the negative log-likelihood
- **Negative Log Likelihood Loss has been used for a long time in many communities for discriminative learning with structured outputs**
  - Speech recognition: many papers going back to the early 90's [Bengio 92], [Bourlard 94]. They call "Maximum Mutual Information"
  - Handwriting recognition [Bengio LeCun 94], [LeCun et al. 98]
  - Bio-informatics [Haussler]
  - Conditional Random Fields [Lafferty et al. 2001]
  - Lots more......
  - In all the above cases, it was used with non-linearly parameterized energies.

What Makes a "Good" Loss Function

- **Good loss functions make the machine produce the correct answer**
  - Avoid collapses and flat energy surfaces



Sufficient Condition on the Loss

Let (X^i, Y^i) be the i^{th} training example and m be a positive margin. Minimizing the loss function L will cause the machine to satisfy $E(W, Y^i, X^i) < E(W, Y, X^i) - m$ for all $Y \neq Y^i$, if there exists at least one point (e_1, e_2) with $e_1 + m < e_2$ such that for all points (e'_1, e'_2) with $e'_1 + m \geq e'_2$, we have

$$Q_{[E_y]}(e_1, e_2) < Q_{[E_y]}(e'_1, e'_2),$$

where $Q_{[E_y]}$ is given by

$$L(W, Y^i, X^i) = Q_{[E_y]}(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)).$$

What Make a "Good" Loss Function

Good and bad loss functions

Loss (equation #)	**Formula**	**Margin**
energy loss	$E(W, Y^i, X^i)$	none
perceptron	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0
LVQ2	$\min\left(M, \max(0, E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	0
MCE	$\left(1 + e^{-\left(E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)\right)}\right)^{-1}$	> 0
square-square	$E(W, Y^i, X^i)^2 - \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2$	m
square-exp	$E(W, Y^i, X^i)^2 + \beta e^{-E(W,\bar{Y}^i,X^i)}$	> 0
NLL/MMI	$E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0
MEE	$1 - e^{-\beta E(W,Y^i,X^i)} / \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0

Advantages/Disadvantages of various losses

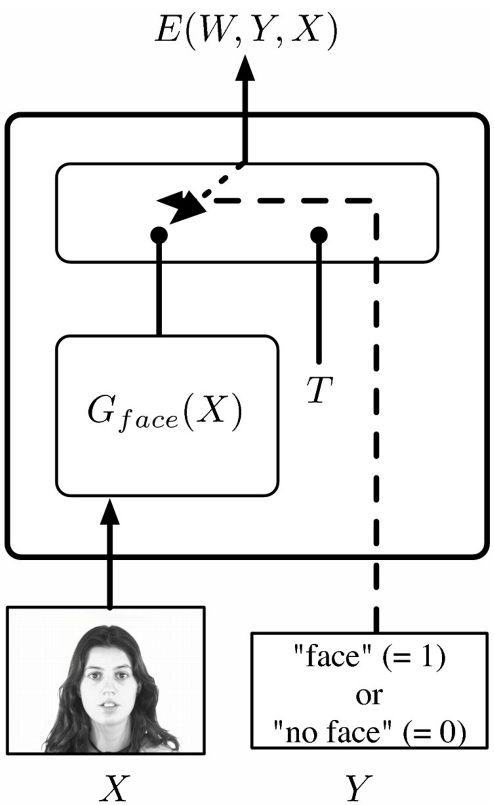
- **Loss functions differ in how they pick the point(s) whose energy is pulled up, and how much they pull them up**
- **Losses with a log partition function in the contrastive term pull up all the bad answers simultaneously.**
  - This may be good if the gradient of the contrastive term can be computed efficiently
  - This may be bad if it cannot, in which case we might as well use a loss with a single point in the contrastive term
- **Variational methods pull up many points, but not as many as with the full log partition function.**
- **Efficiency of a loss/architecture: how many energies are pulled up for a given amount of computation?**
  - The theory for this is to be developed

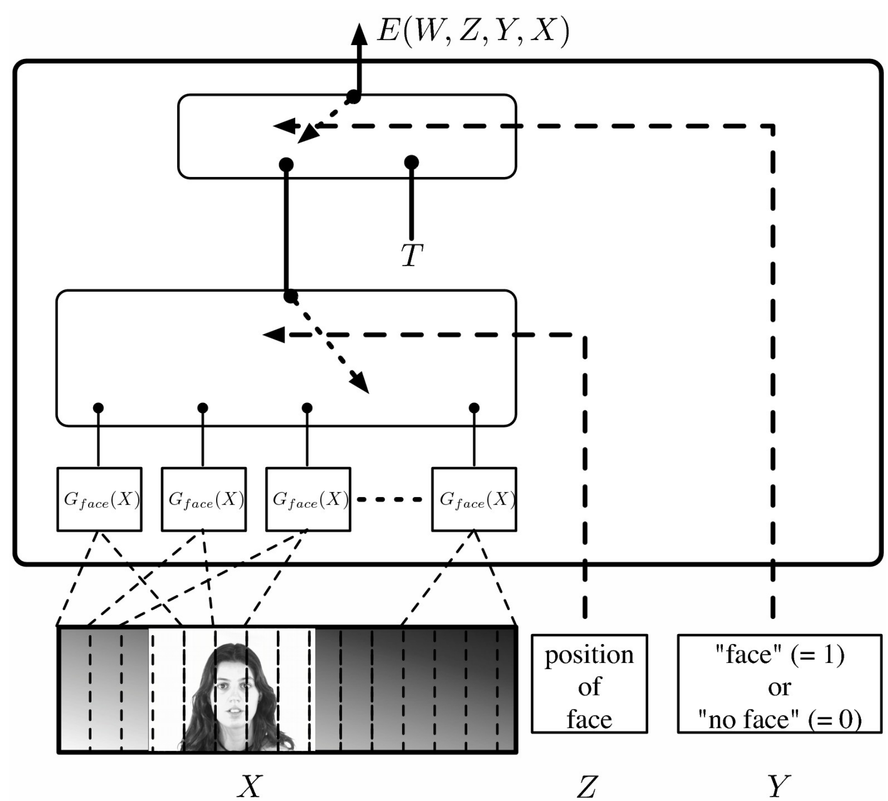
Latent Variable Models

- **The energy includes "hidden" variables Z whose value is never given to us**

$$E(Y,X) = \min_{Z \in \mathcal{Z}} E(Z,Y,X).$$

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}, Z \in \mathcal{Z}} E(Z,Y,X).$$





What can the latent variables represent?

- **Variables that would make the task easier if they were known:**
  - **Face recognition**: the gender of the person, the orientation of the face.
  - **Object recognition**: the pose parameters of the object (location, orientation, scale), the lighting conditions.
  - **Parts of Speech Tagging**: the segmentation of the sentence into syntactic units, the parse tree.
  - **Speech Recognition**: the segmentation of the sentence into phonemes or phones.
  - **Handwriting Recognition**: the segmentation of the line into characters.

- **In general, we will search for the value of the latent variable that allows us to get an answer (Y) of smallest energy.**

Probabilistic Latent Variable Models

- **Marginalizing over latent variables instead of minimizing.**

$$P(Z,Y|X) = \frac{e^{-\beta E(Z,Y,X)}}{\int_{y\in\mathcal{Y},\ z\in\mathcal{Z}} e^{-\beta E(y,z,X)}}.$$

$$P(Y|X) = \frac{\int_{z\in\mathcal{Z}} e^{-\beta E(Z,Y,X)}}{\int_{y\in\mathcal{Y},\ z\in\mathcal{Z}} e^{-\beta E(y,z,X)}}.$$

- **Equivalent to traditional energy-based inference with a redefined energy function:**

$$Y^* = \text{argmin}_{Y\in\mathcal{Y}} - \frac{1}{\beta}\log \int_{z\in\mathcal{Z}} e^{-\beta E(z,Y,X)}.$$

- **Reduces to traditional minimization when Beta->infinity**

What's so bad about probabilistic models?

- Why bother with a normalization since we don't use it for decision making?
- Why insist that P(Y|X) have a specific shape, when we only care about the position of its minimum?
- When Y is high-dimensional (or simply combinatorial), normalizing becomes intractable (e.g. Language modeling, image restoration, large DoF robot control...).
- A tiny number of models are pre-normalized (Gaussian, exponential family)
- A very small number are easily normalizable
- A large number have intractable normalization
- A huuuge number can't be normalized at all (examples will be shown).
- Normalization forces us to take into account areas of the space that we don't actually care about because our inference algorithm never takes us there.
- **If we only care about making the right decisions, maximizing the likelihood solves a much more complex problem than we have to.**

EBM

- Unlike traditional classifiers, EBMs can represent multiple alternative outputs
- The normalization in probabilistic models is often an unnecessary aggravation, particularly if the ultimate goal of the system is to make decisions.
- EBMs with appropriate loss function avoid the necessity to compute the partition function and its derivatives (which may be intractable)
- EBMs give us complete freedom in the choice of the architecture that models the joint “incompatibility” (energy) between the variables.
- We can use architectures that are not normally allowed in the probabilistic framework (like neural nets).
- **The inference algorithm that finds the most offending (lowest energy) incorrect answer does not need to be exact:** our model may give low energy to far-away regions of the landscape. But if our inference algorithm never finds those regions, **they do not affect us.** But they do affect normalized probabilistic models

Face Detection and Pose Estimation with a Convolutional EBM

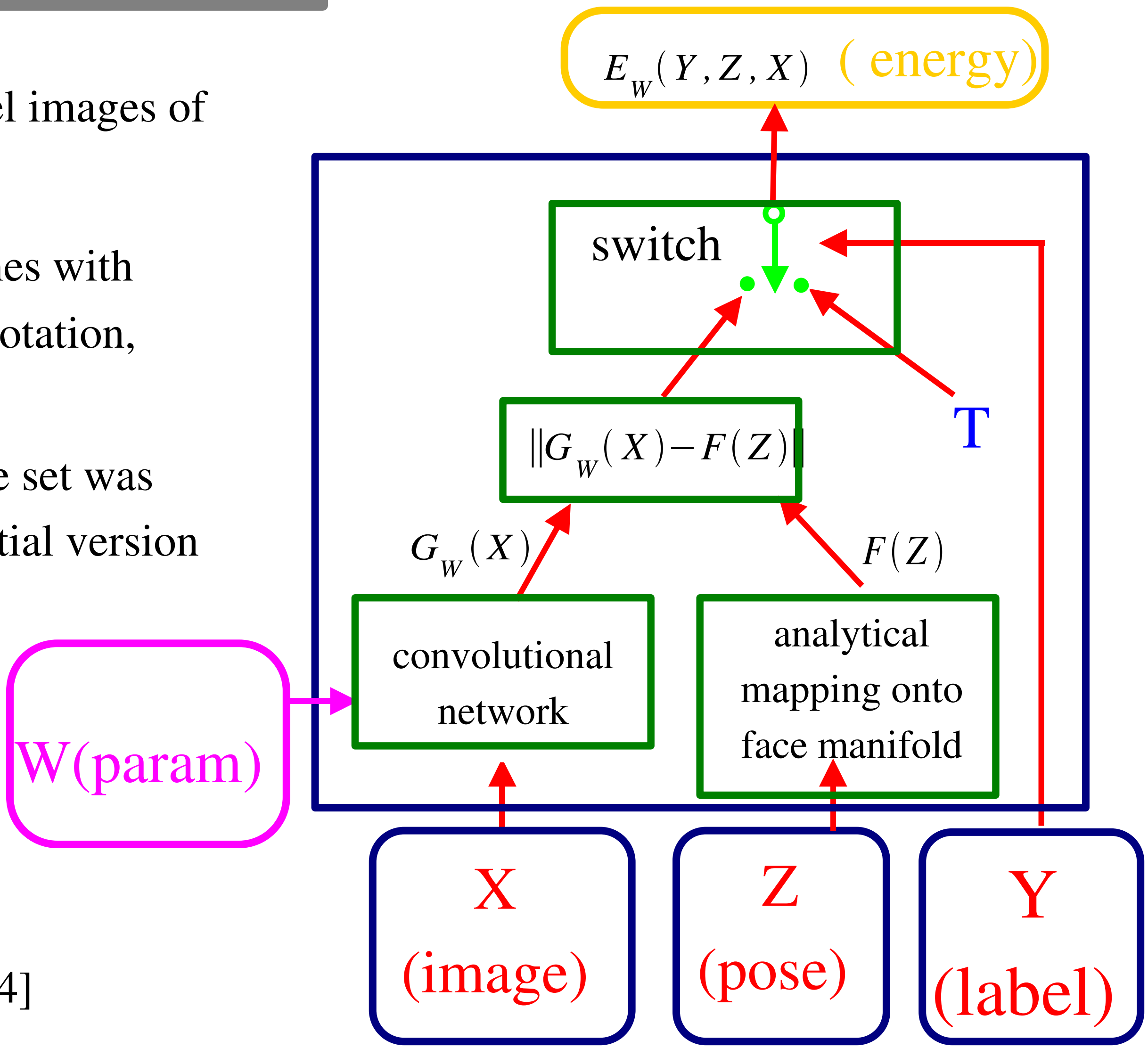
$$E^*(W,X) = \min_Z ||G_W(X) - F(Z)||$$

$$Z^* = \text{argmin}_Z ||G_W(X) - F(Z)||$$

- **Training:** 52,850, 32x32 grey-level images of faces, 52,850 non-faces.
- Each training image was used 5 times with random variation in scale, in-plane rotation, brightness and contrast.
- **2nd phase:** half of the initial negative set was replaced by false positives of the initial version of the detector .

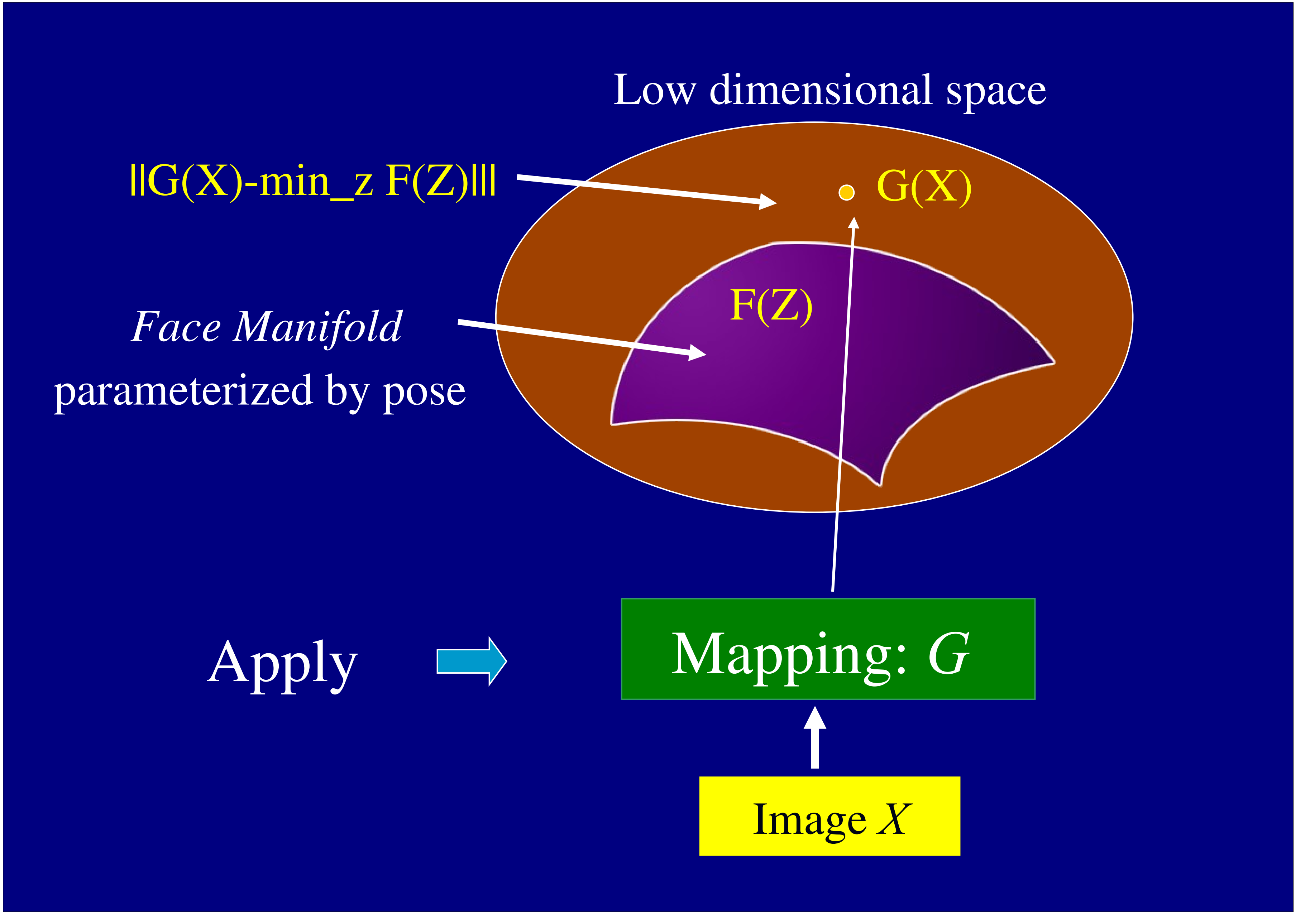
Small E*(W,X): face

Large E*(W,X): no face



[Osadchy, Miller, LeCun, NIPS 2004]

Face Manifold



Probabilistic Approach: Density model of joint P(face,pose)

Probability that image X is a face with pose Z

$$P(X,Z) = \frac{\exp(-E(W,Z,X))}{\int_{X,Z \in \text{images,poses}} \exp(-E(W,Z,X))}$$

Given a training set of faces annotated with pose, find the W that maximizes the likelihood of the data under the model:

$$P(\text{faces} + \text{pose}) = \prod_{X,Z \in \text{faces+pose}} \frac{\exp(-E(W,Z,X))}{\int_{X,Z \in \text{images,poses}} \exp(-E(W,Z,X))}$$

Equivalently, minimize the negative log likelihood:

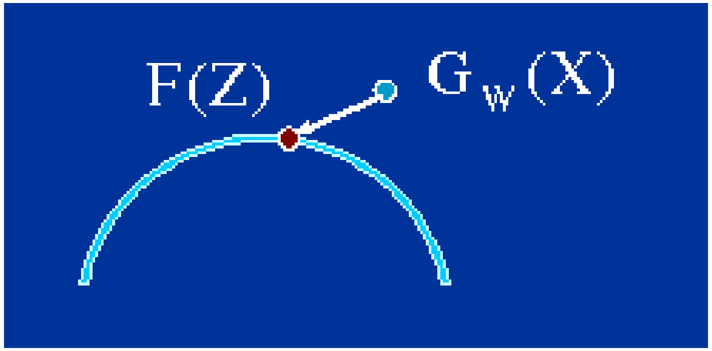
$$\mathcal{L}(W, \text{faces} + \text{pose}) = \sum_{X,Z \in \text{faces+pose}} E(W,Z,X) + \log \left[\int_{X,Z \in \text{images,poses}} \exp(-E(W,Z,X)) \right]$$

COMPLICATED

Energy-Based Contrastive Loss Function

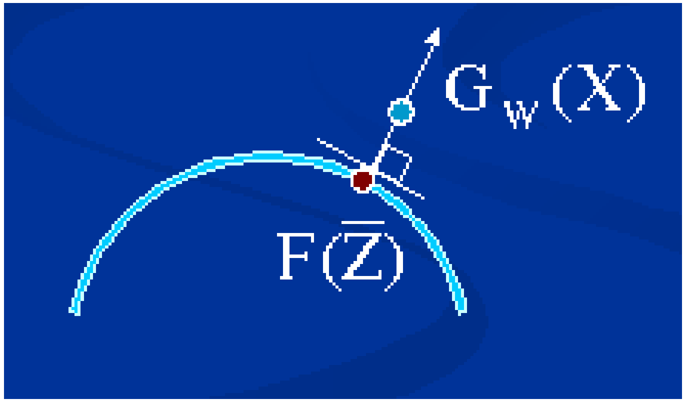
$$\mathcal{L}(W) = \frac{1}{|\mathrm{f}+\mathrm{p}|} \sum_{X,Z \in \mathrm{faces+pose}} [L^{+}(E(W,Z,X))] + L^{-}\left(\min_{X,Z \in \mathrm{bckgnd,poses}} E(W,Z,X)\right)$$

$$L^{+}(E(W,Z,X)) = E(W,Z,X)^2 = ||G_W(X) - F(Z)||^2$$



Attract the network output Gw(X) to the location of the desired pose F(Z) on the manifold

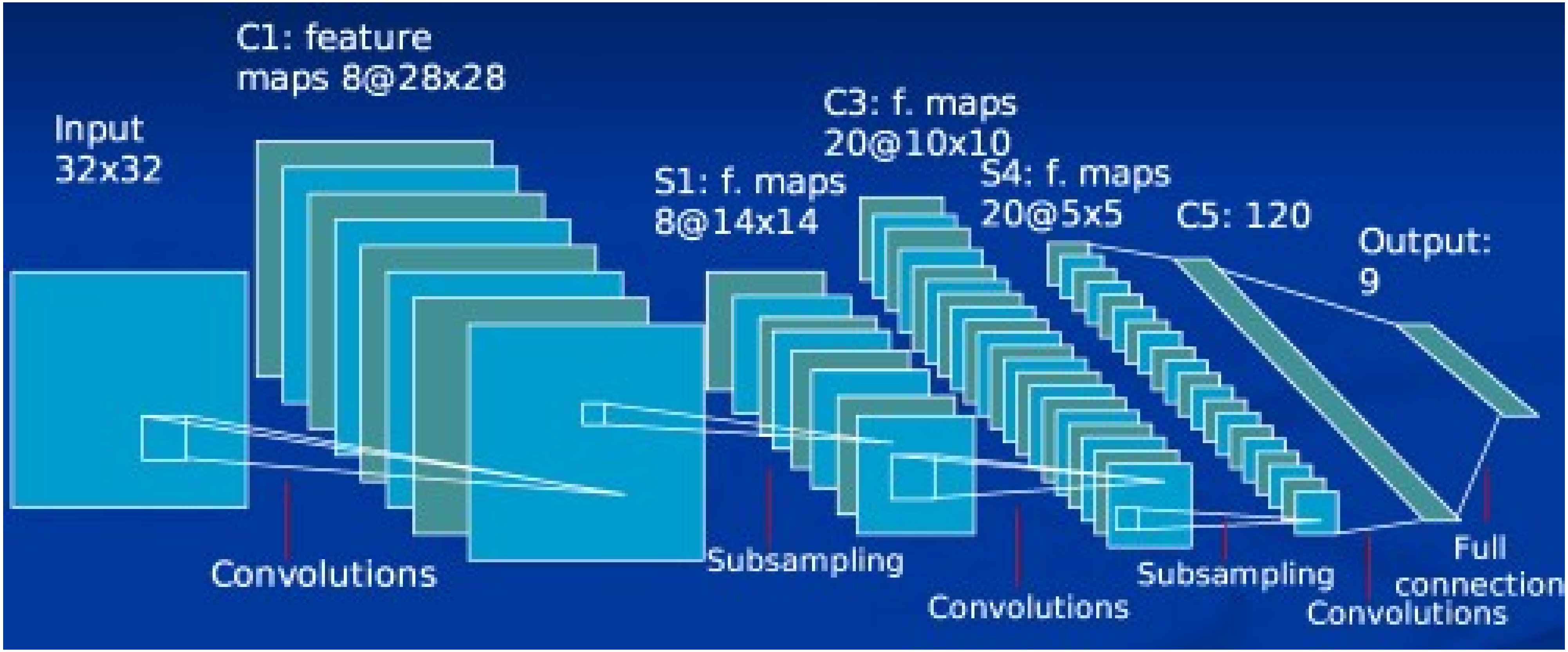
$$L^{-}\left(\min_{X,Z \in \mathrm{bckgnd,poses}} E(W,Z,X)\right) = K \exp\left(-\min_{X,Z \in \mathrm{bckgnd,poses}} ||G_W(X) - F(Z)||\right)$$



Repel the network output Gw(X) away from the face/pose manifold

Convolutional Network Architecture

[LeCun et al. 1988, 1989, 1998, 2005]

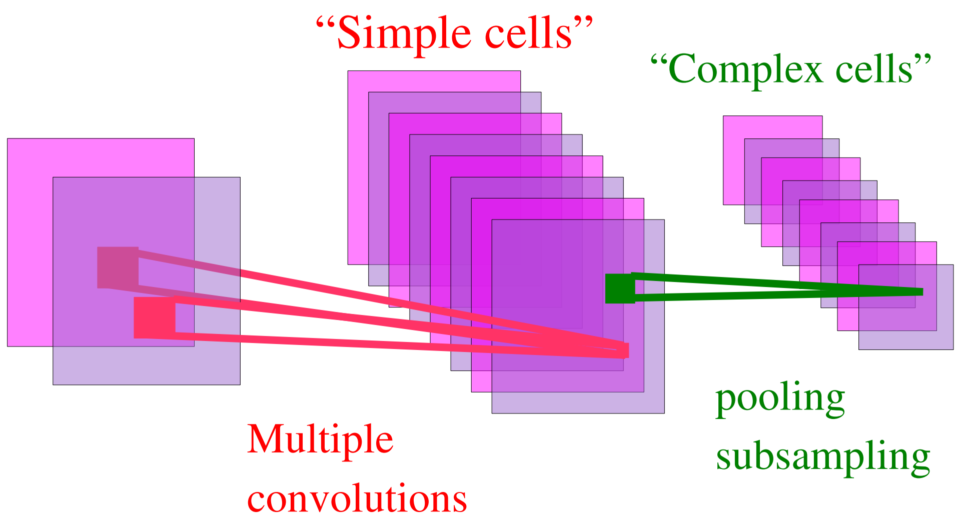


Hierarchy of local filters (convolution kernels),

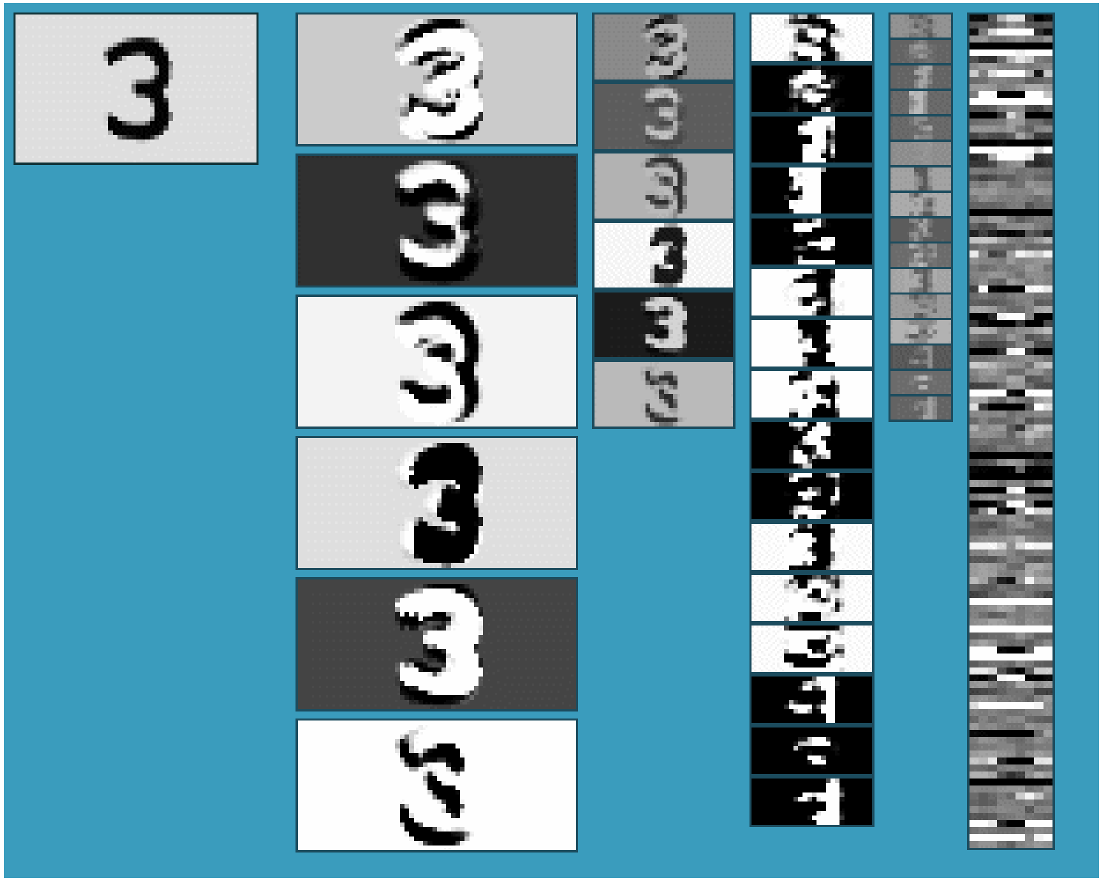
sigmoid pointwise non-linearities, and spatial subsampling

All the filter coefficients are learned with gradient descent (back-prop)

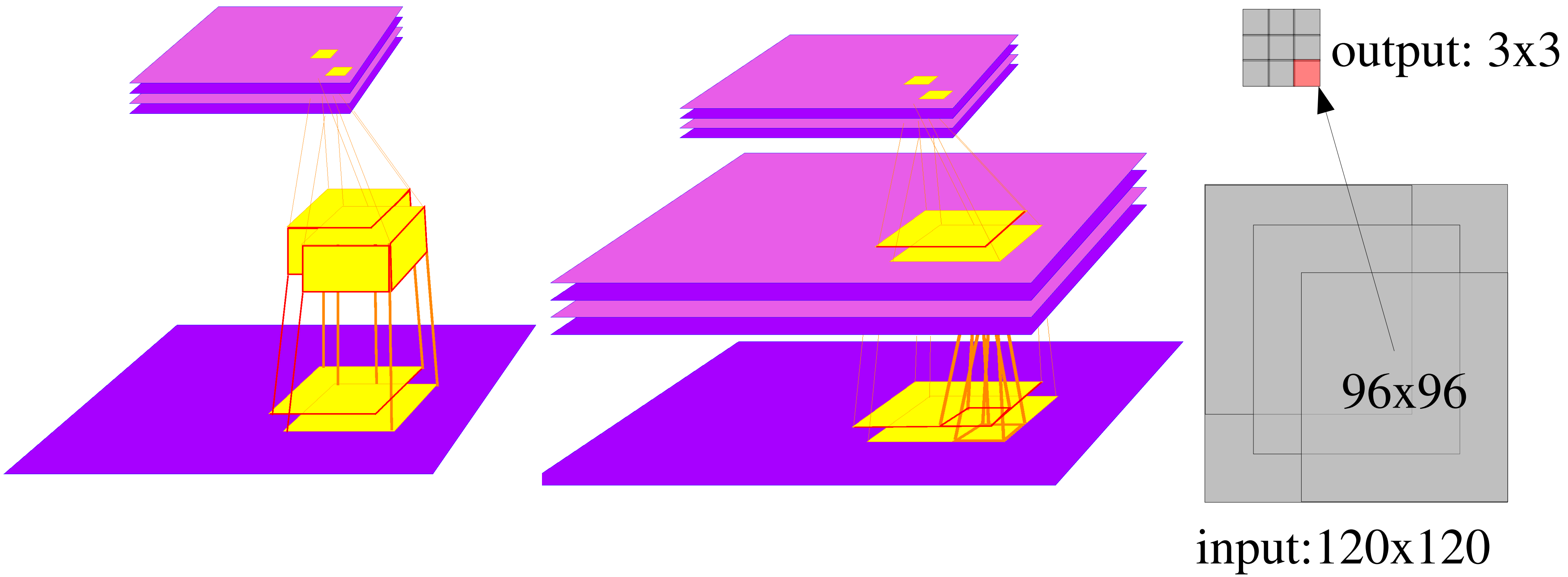
Alternated Convolutions and Pooling/Subsampling



- **Local features are extracted everywhere.**
- **pooling/subsampling layer builds robustness to variations in feature locations.**
- **Long history in neuroscience and computer vision:**
  - **Hubel/Wiesel 1962,**
  - **Fukushima 1971-82,**
  - **LeCun 1988-06**
  - **Poggio, Riesenhuber, Serre 02-06**
  - **Ullman 2002-06**
  - **Triggs, Lowe,....**



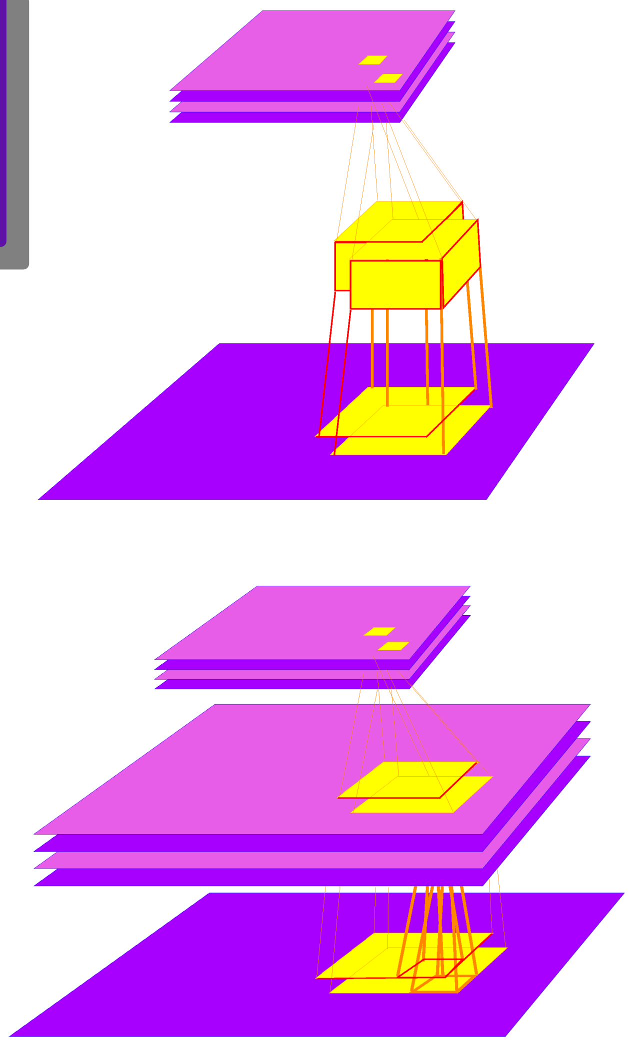
Building a Detector/Recognizer: Replicated Conv. Nets

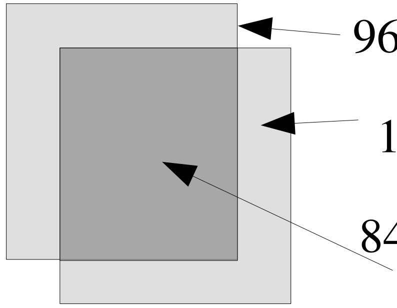


- Traditional Detectors/Classifiers must be applied to every location on a large input image, at multiple scales.
- Convolutional nets can replicated over large images very cheaply.
- The network is applied to multiple scales spaced by 1.5.

Building a Detector/Recognizer: Replicated Convolutional Nets

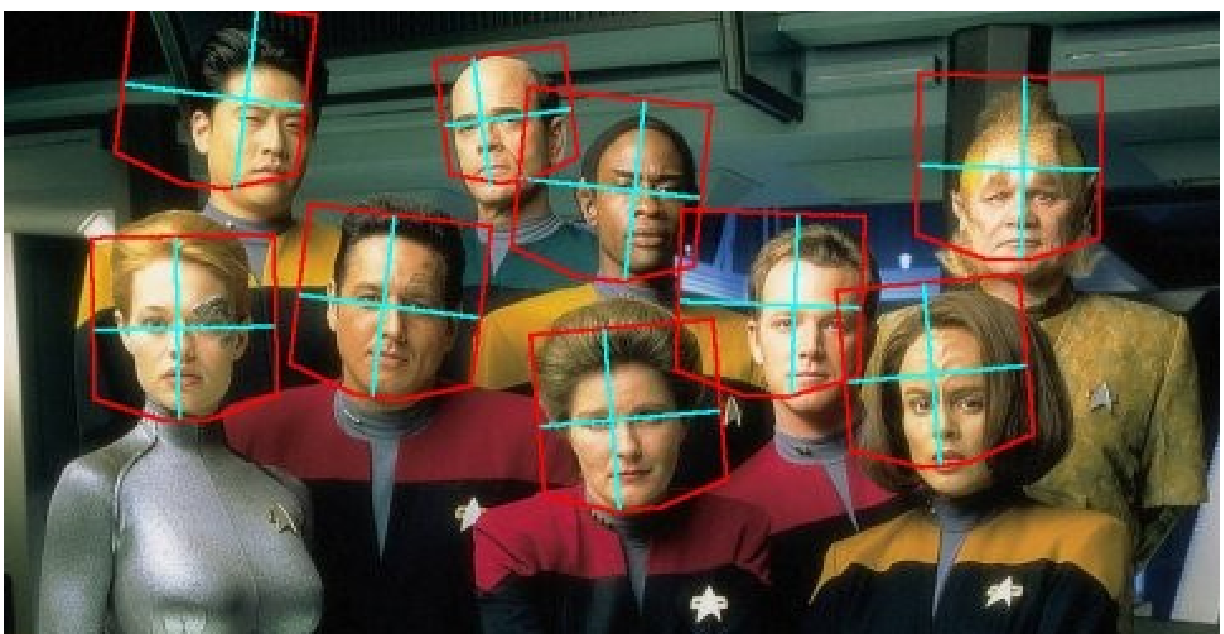
- Computational cost for replicated convolutional net:
  - 96x96 -> 4.6 million multiply-accumulate operations
  - 120x120 -> 8.3 million multiply-accumulate operations
  - 240x240 -> 47.5 million multiply-accumulate operations
  - 480x480 -> 232 million multiply-accumulate operations
- Computational cost for a non-convolutional detector of the same size, applied every 12 pixels:
  - 96x96 -> 4.6 million multiply-accumulate operations
  - 120x120 -> 42.0 million multiply-accumulate operations
  - 240x240 -> 788.0 million multiply-accumulate operations
  - 480x480 -> 5,083 million multiply-accumulate operations

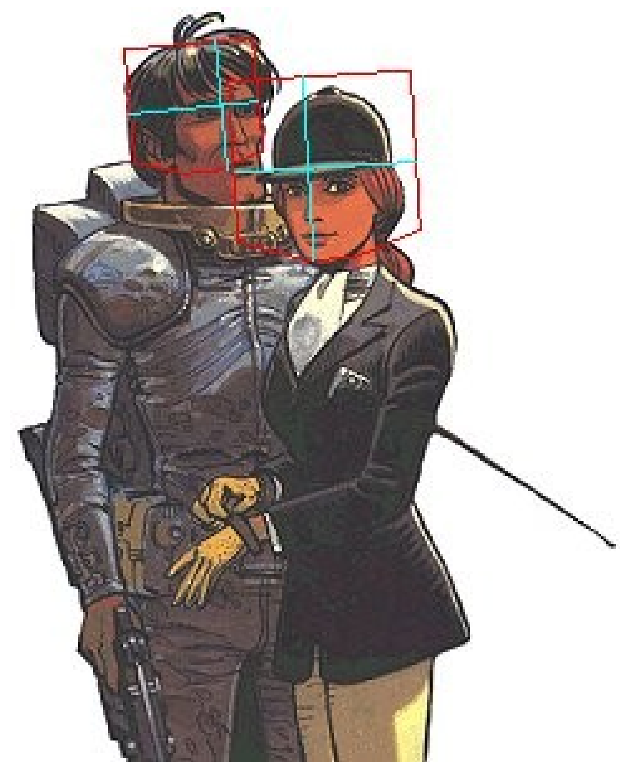




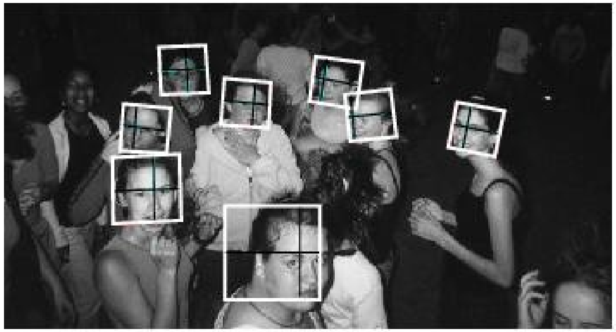
Face Detection: Results

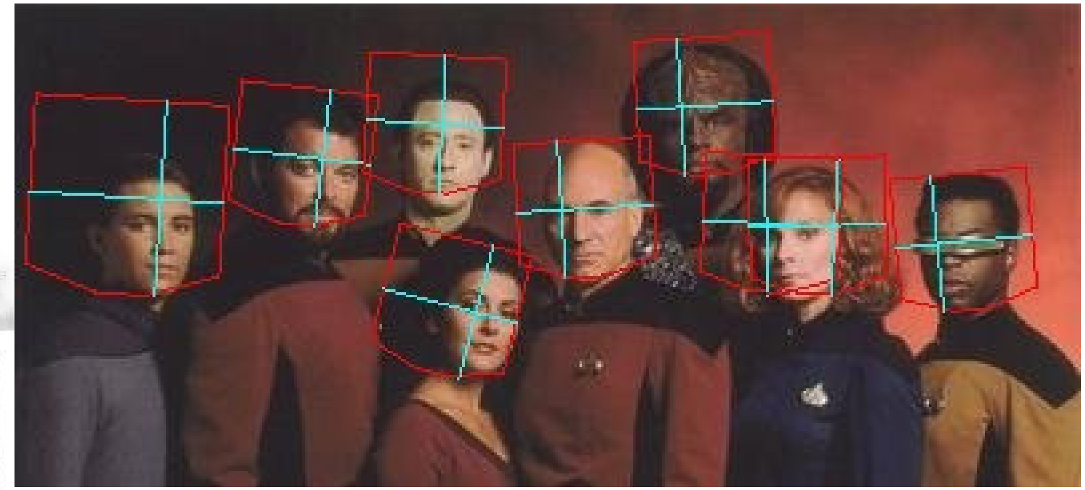
Data Set-> / *False positives per image->*	TILTED		PROFILE		MIT+CMU	
	4.42	26.9	0.47	3.36	0.5	1.28
Our Detector	90%	97%	67%	83%	83%	88%
Jones & Viola (tilted)	90%	95%	x		x	
Jones & Viola (profile)	x		70%	83%	x	

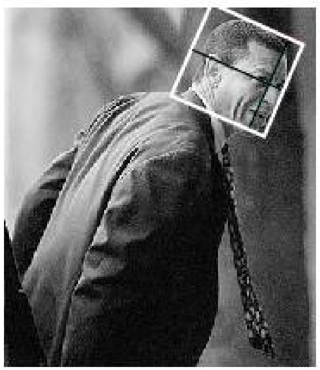


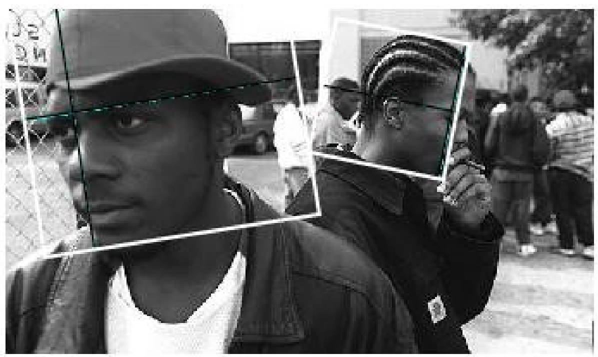


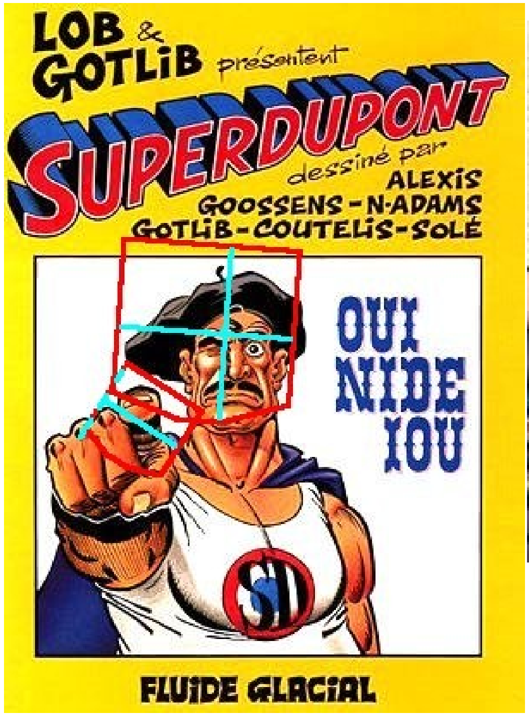
Face Detection and Pose Estimation: Results

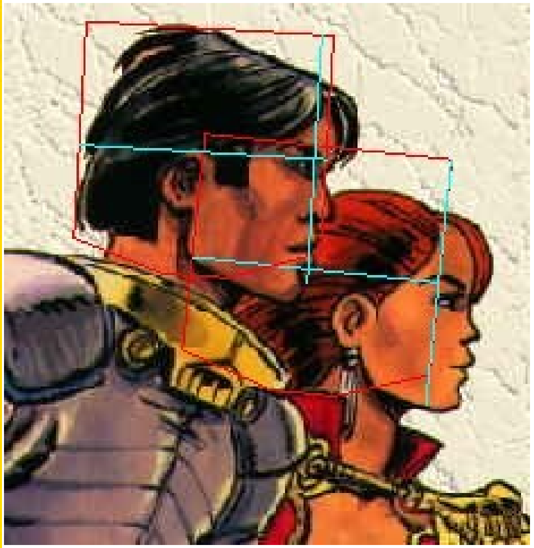


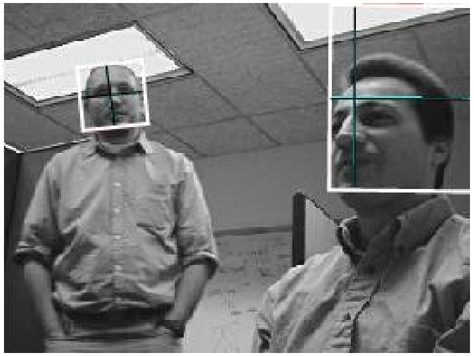


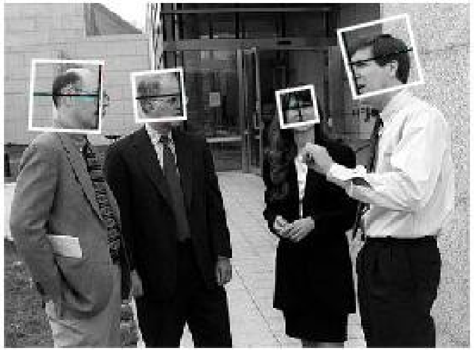




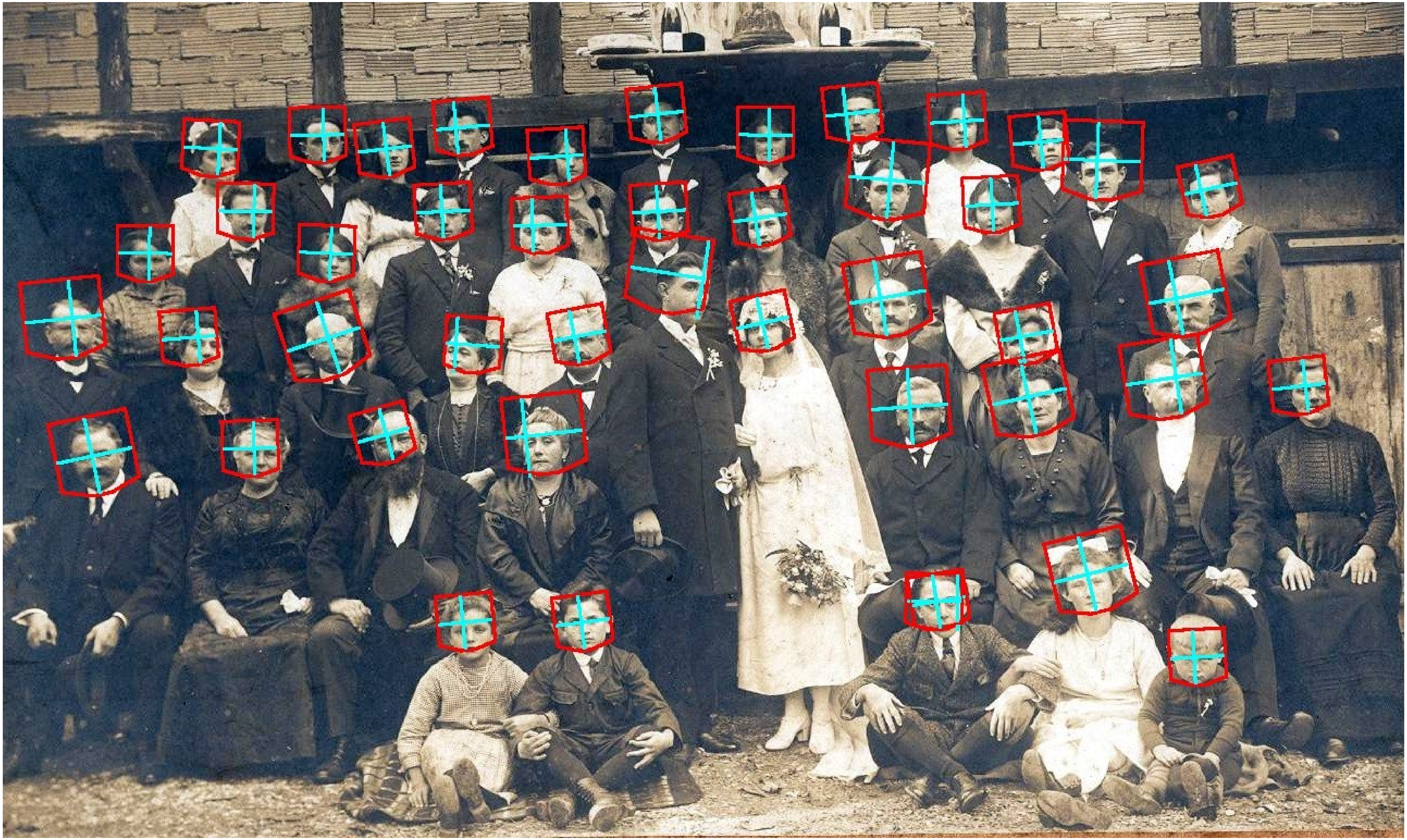








Face Detection with a Convolutional Net



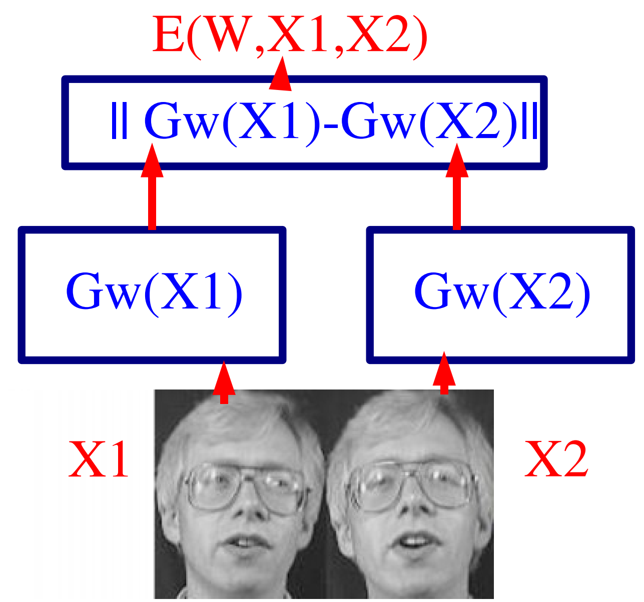
How do we Handle Lots of Classes?

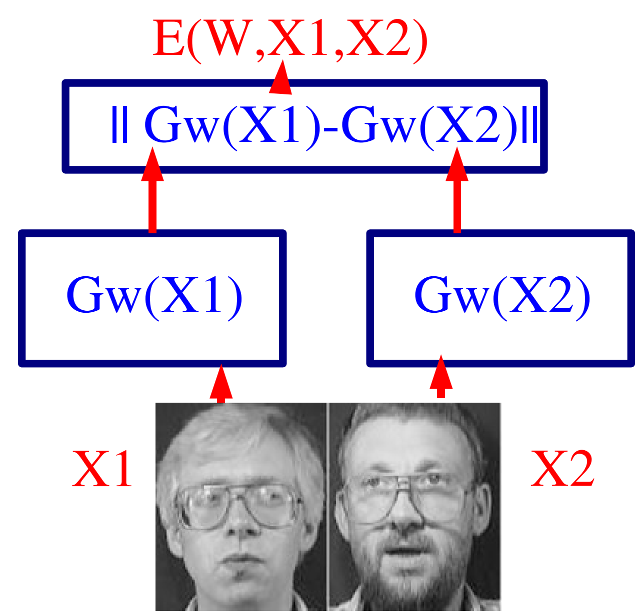
- **Example: face recognition**
  - We do not have pictures of every person
- **We must be able to learn something without seeing all the classes**
- **Solution: learn a similarity metric**
- **Map images to a low dimensional space in which**
  - Two images of the same person are mapped to nearby points
  - Two images of different persons are mapped to distant points

Comparing Objects: Learning an Invariant Dissimilarity Metric

[Chopra, Hadsell, LeCun CVPR 2005]

- Training a **parameterized, invariant dissimilarity metric** may be a solution to the **many-category problem**.
- Find a mapping **Gw(X)** such that the Euclidean distance **||Gw(X1)- Gw(X2)||** reflects the "semantic" distance between X1 and X2.
- Once trained, a trainable dissimilarity metric can be used to classify **new categories using a very small number of training samples** (used as prototypes).
- This is an example where probabilistic models are too constraining, because we would have to limit ourselves to models that can be normalized over the space of input pairs.
- With EBMs, we can put what we want in the box (e.g. A convolutional net).
- **Siamese Architecture**
- **Application:** face verification/recognition





Face Verification datasets: AT&T/ORL

- The AT&T/ORL dataset
- Total subjects: **40**. Images per subject: **10**. Total images: **400**.
- Images had a moderate degree of variation in pose, lighting, expression and head position.
- Images from **35** subjects were used for training. Images from **5** remaining subjects for testing.
- Training set was taken from: **3500** genuine and **119000** impostor pairs.
- Test set was taken from: **500** genuine and **2000** impostor pairs.
- http://www.uk.research.att.com/facedatabase.html

AT&T/ORL Dataset

Internal state for genuine and impostor pairs

Classification Examples

- **Example: Correctly classified genuine pairs**

energy: 0.3159

energy: 0.0043

energy: 0.0046

- **Example: Correctly classified impostor pairs**

energy: 20.1259

energy: 32.7897

energy: 5.7186

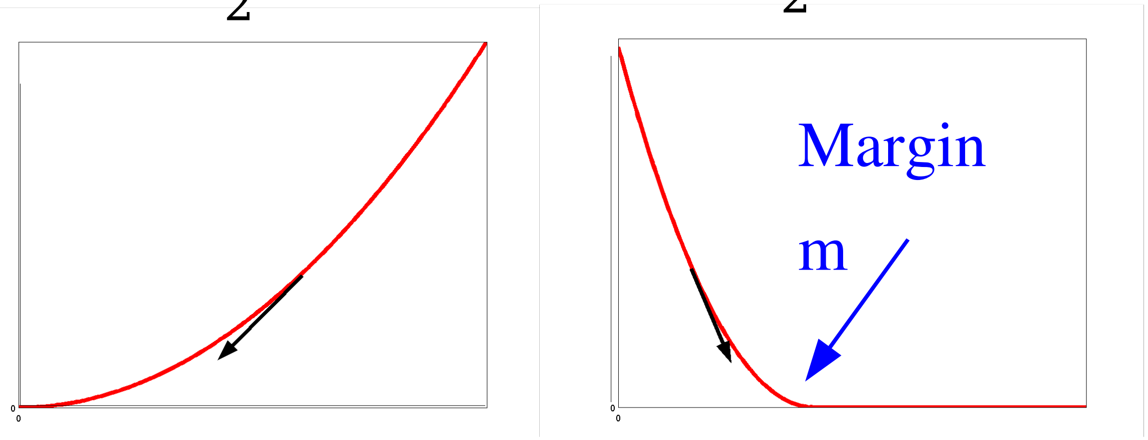
- **Example: Mis-classified pairs**

energy: 10.3209

energy: 2.8243

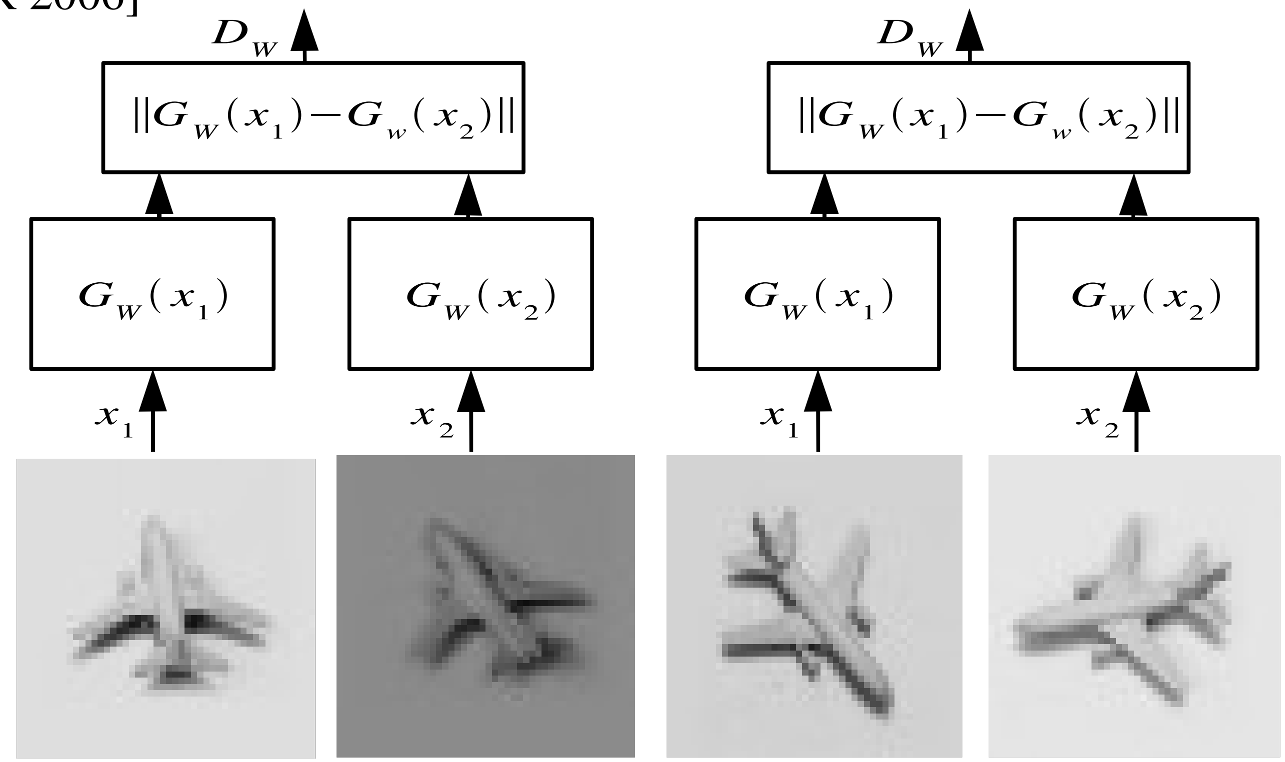
A similar idea for Learning a Manifold with Invariance Properties

$$L_{similar} = \frac{1}{2} D_W^2 \qquad L_{dissimilar} = \frac{1}{2} \{max(0, m - D_W)\}^2$$

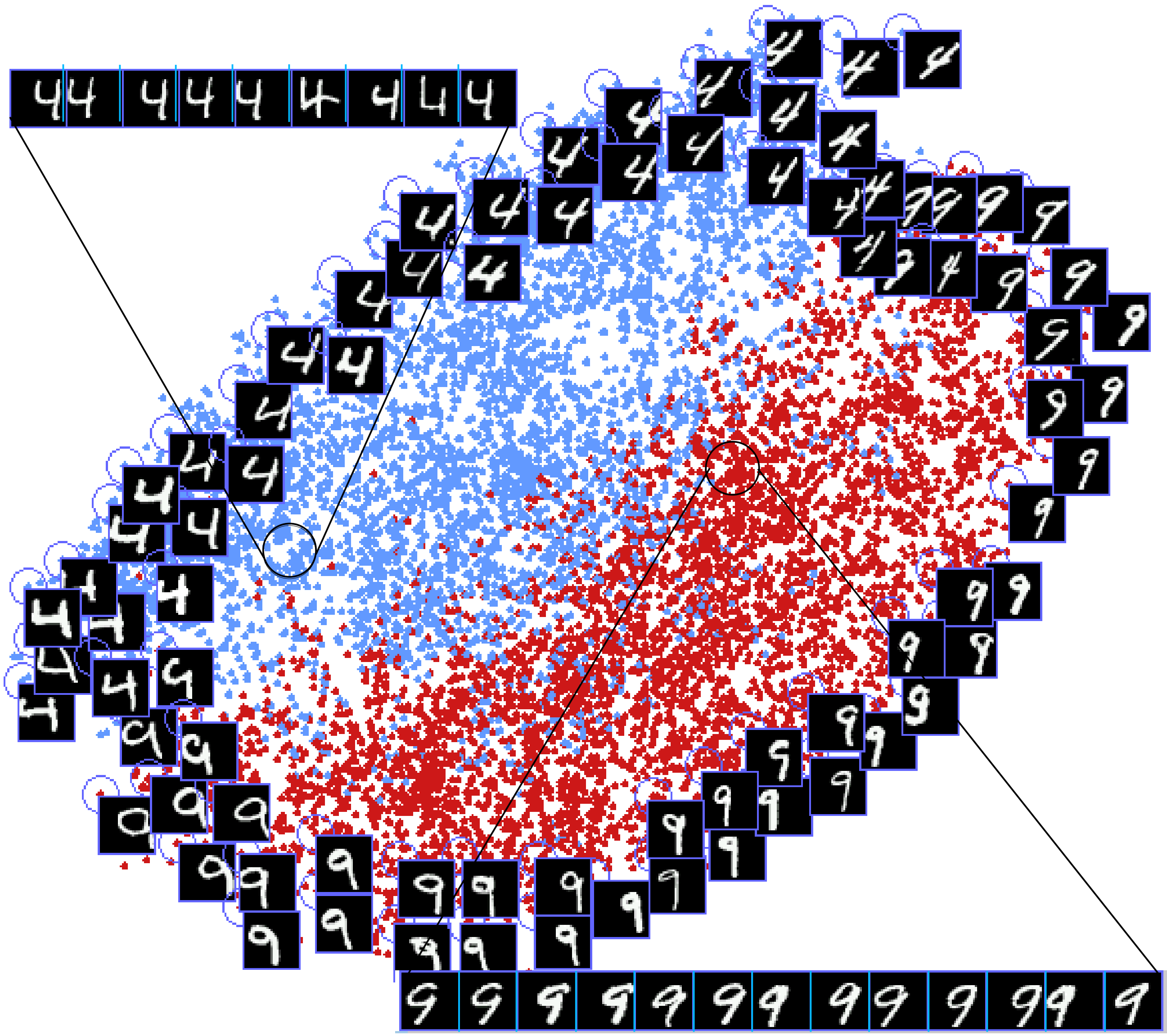


[Hadsell, Chopra, LeCun, CVPR 2006]

- **Loss function:**
  - Pay quadratically for making outputs of neighbors far apart
  - Pay quadratically for making outputs of non-neighbors smaller than a **margin** m

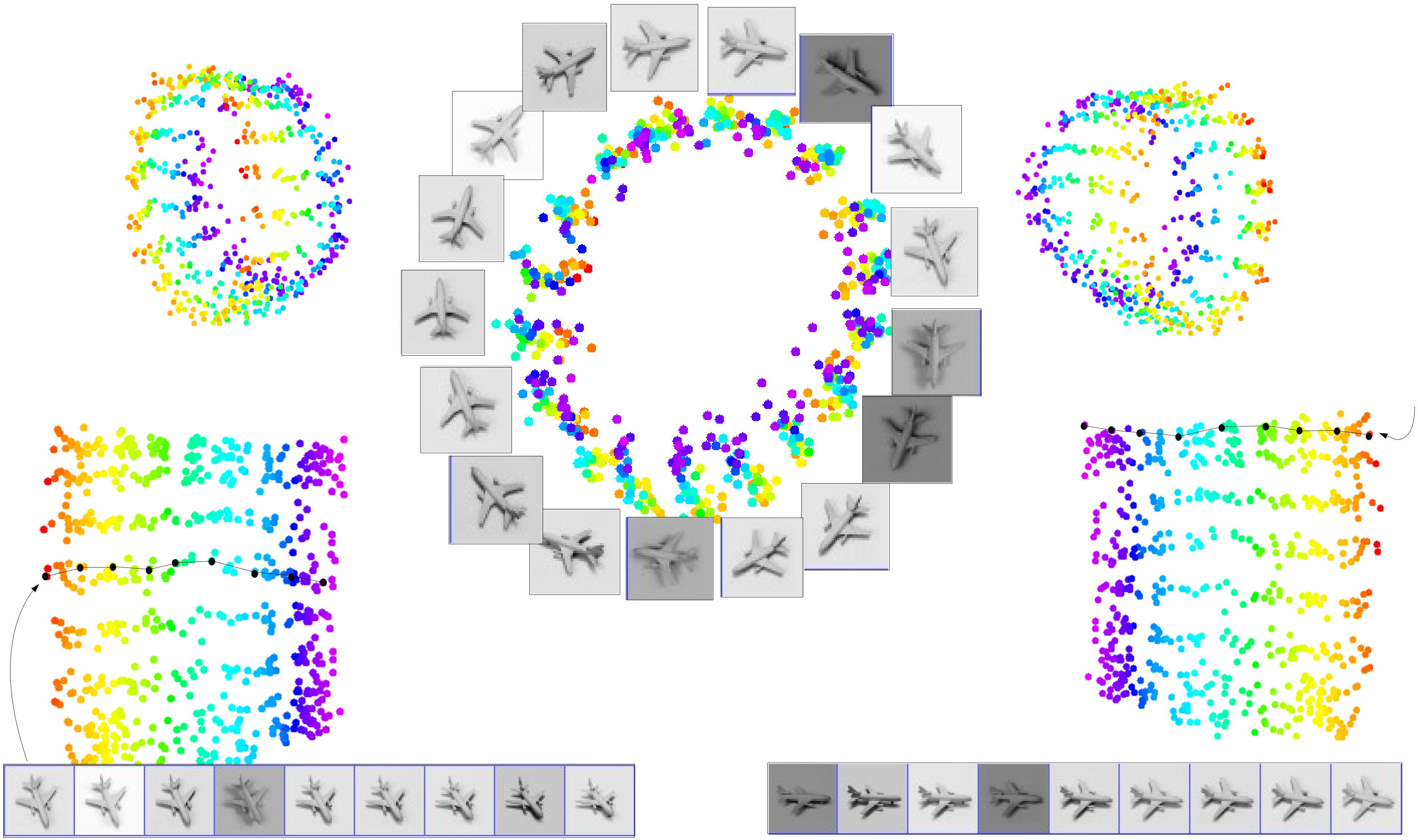


A Manifold with Invariance to Shifts



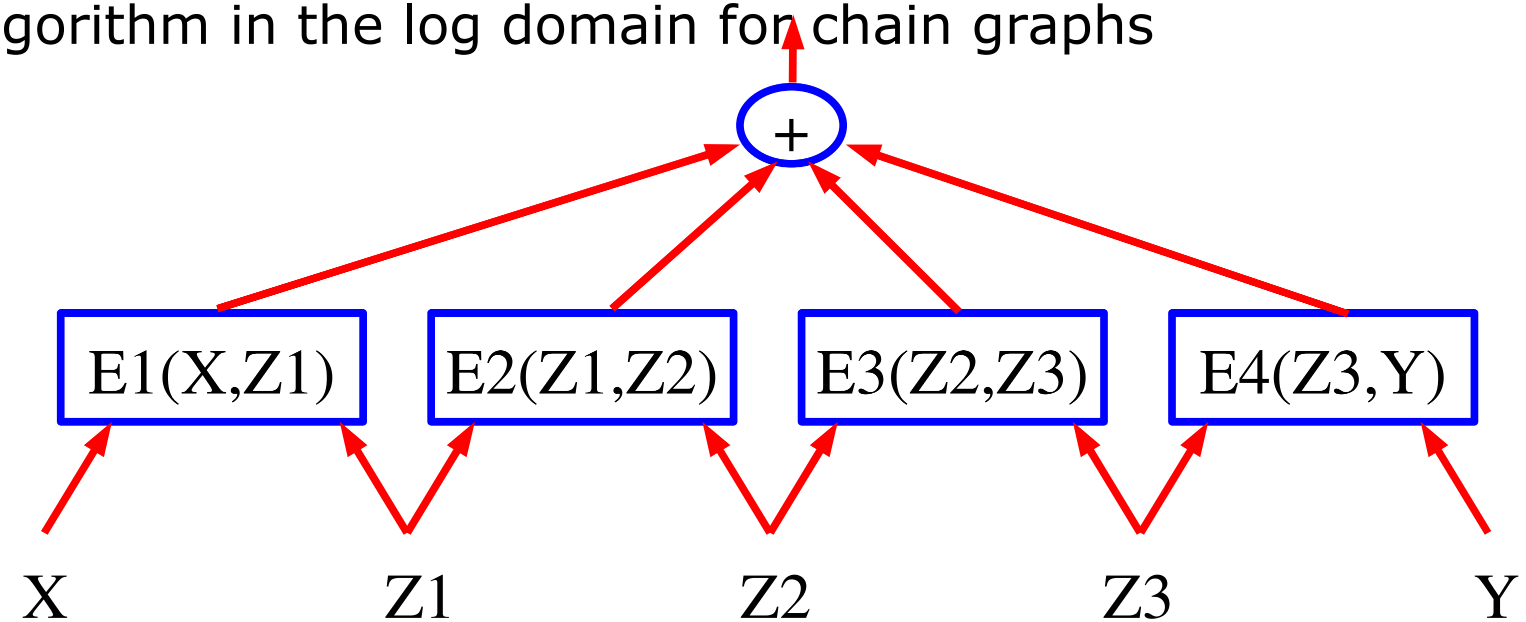
- Training set: 3000 “4” and 3000 “9” from MNIST. **Each digit is shifted horizontally by -6, -3, 3, and 6 pixels**
- Neighborhood graph: 5 nearest neighbors in Euclidean distance, **and shifted versions of self and nearest neighbors**
- Output Dimension: 2
- Test set (shown) 1000 “4” and 1000 “9”

Automatic Discovery of the Viewpoint Manifold with Invariant to Illumination



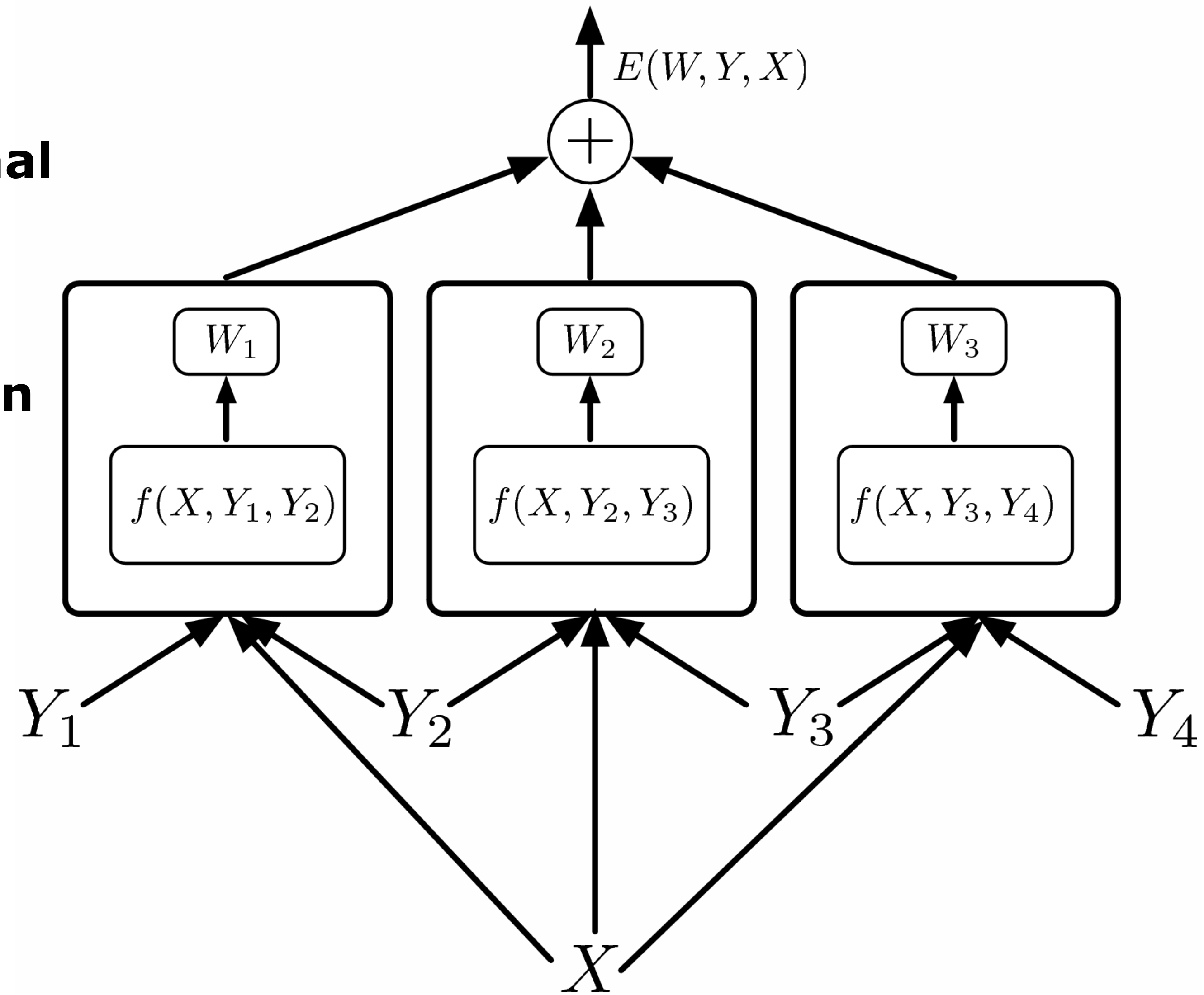
Non-Probabilistic Graphical Models: Energy-Based Factor Graphs

- **When the energy is a sum of partial energy functions (or when the probability is a product of factors):**
  - An EBM can be seen as an unnormalized factor graph in the log domain
  - Our favorite efficient inference algorithms can be used for inference (without the normalization step).
  - Min-sum algorithm (instead of max-product), Viterbi for chain graphs
  - (Log/sum/exp)-sum algorithm (instead of sum-product), Forward algorithm in the log domain for chain graphs



Example of EBFG: "Shallow" Factors

- **Linearly Parameterized Factors**
- **with the NLL Loss :**
  - Lafferty's **Conditional Random Field**
- **with Hinge Loss:**
  - Taskar's **Max Margin Markov Nets**
- **with Perceptron Loss**
  - Collins's sequence labeling model
- **With Log Loss:**
  - Altun/Hofmann sequence labeling model



Deep/non-linear Factors: ASR with TDNN/DTW

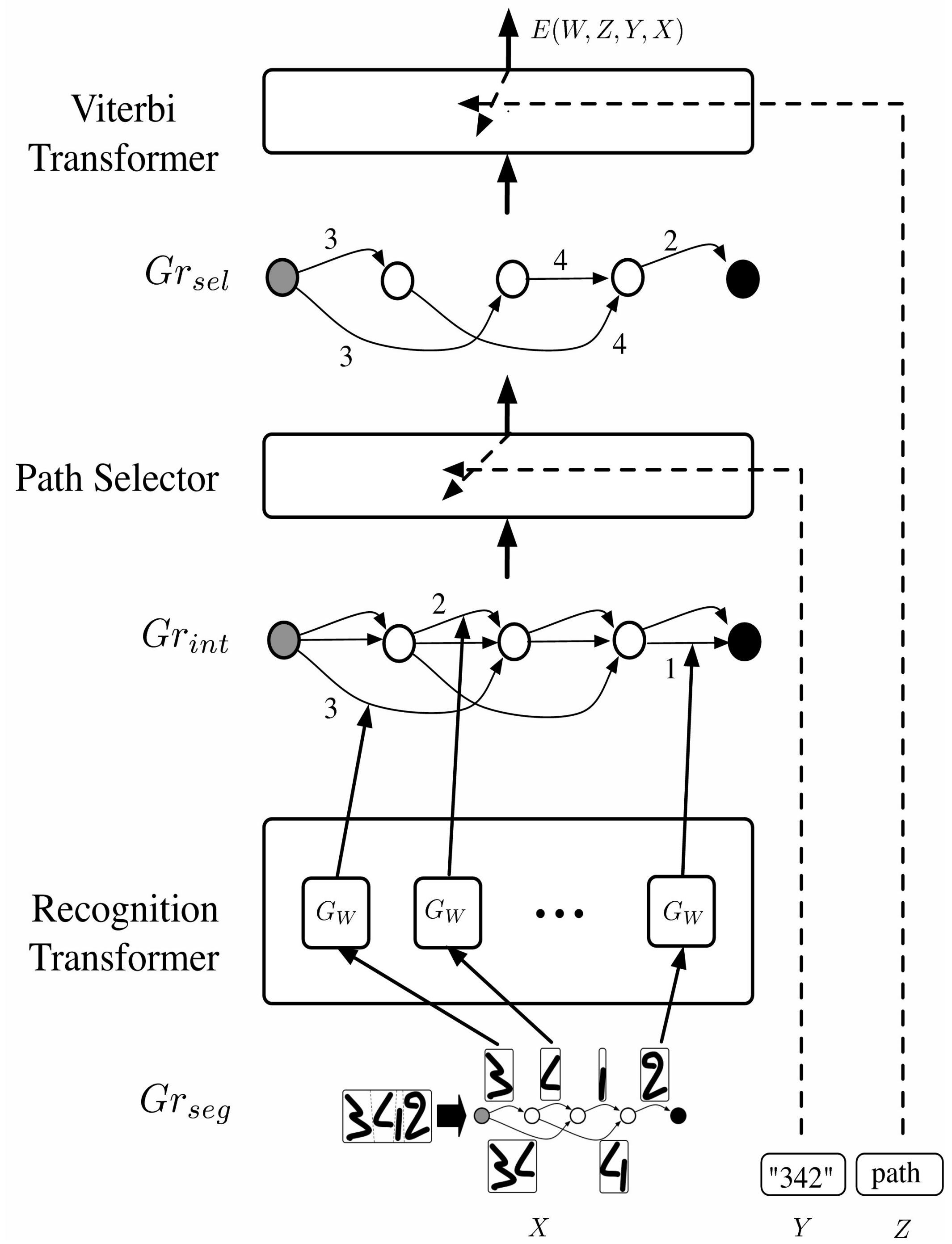
- Trainable Speech/Handwriting Recognition systems that integrate Neural Nets (or other "deep" classifiers) with dynamic time warping, Hidden Markov Models, or other graph-based hypothesis representations

- Training the feature extractor as part of the whole process.
- with the LVQ2 Loss :
  - Driancourt and Bottou's speech recognizer (1991)
- with NLL:
  - Bengio's speech recognizer (1992)
  - Haffner's speech recognizer (1993)

- With Minimum Empirical Error loss
  - Ljolje and Rabiner (1990)
- with NLL:
  - Bengio (1992), Haffner (1993), Bourlard (1994)
- With MCE
  - Juang et al. (1997)
- Late normalization scheme (un-normalized HMM)
  - Bottou pointed out the **label bias problem** (1991)
  - Denker and Burges proposed a solution (1995)

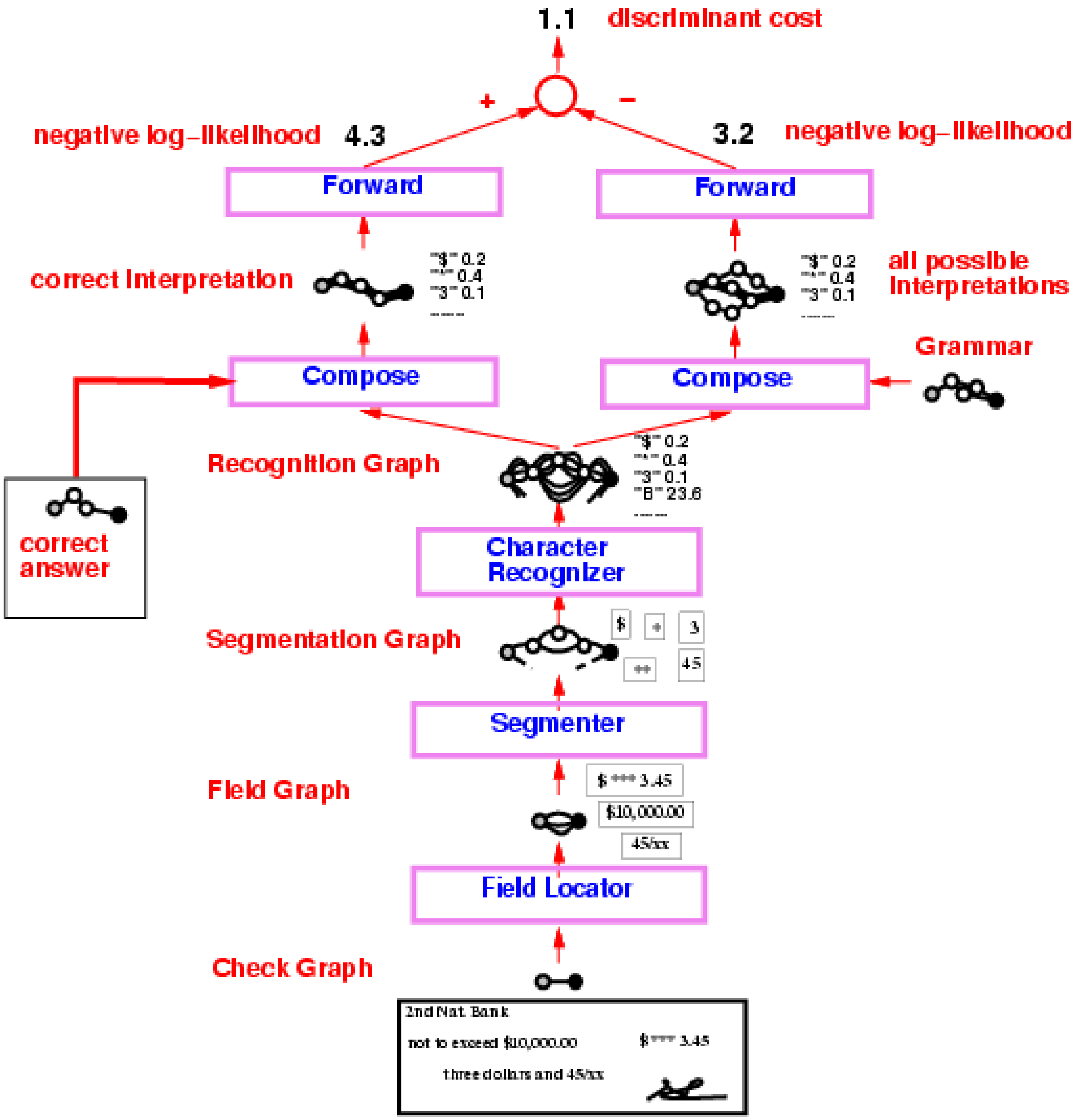
Really Deep Factors & implicit graphs: GTN

- **Handwriting Recognition with Graph Transformer Networks**
- **Un-normalized hierarchical HMMs**
  - Trained with Perceptron loss [LeCun, Bottou, Bengio, Haffner 1998]
  - Trained with NLL loss [Bengio, LeCun 1994], [LeCun, Bottou, Bengio, Haffner 1998]
- **Answer = sequence of symbols**
- **Latent variable = segmentation**



Check Reader

- Graph transformer network trained to read check amounts.
- Trained globally with Negative-Log-Likelihood loss.
- 50% percent corrent, 49% reject, 1% error (detectable later in the process.
- Fielded in 1996, used in many banks in the US and Europe.
- Processes an estimated 10% of all the checks written in the US.

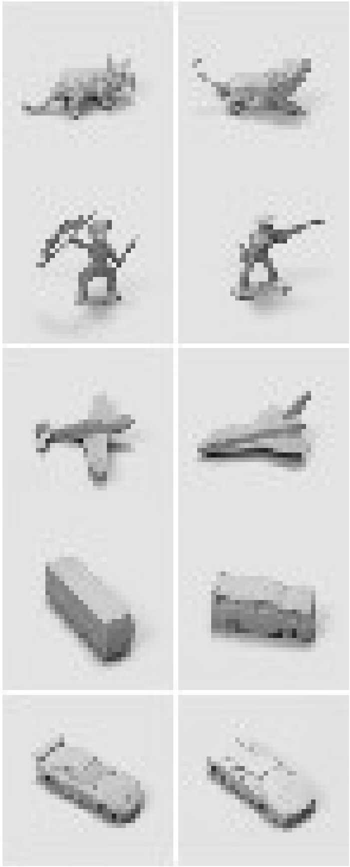


The "Deep Learning Problem": Generic Object Detection and Recognition with Invariance to Pose, Illumination and Clutter

[Huang, LeCun, CVPR 2006, CVPR 2004]

Generic Object Detection and Recognition with Invariance to Pose, Illumination and Clutter

- **Computer Vision and Biological Vision are getting back together again after a long divorce (Hinton, LeCun, Poggio, Ullman, Lowe, Triggs, S. Geman, Itti, Olshausen, Simoncelli, ...).**
- **What happened? (1) Machine Learning, (2) Moore's Law.**
- **Generic Object Recognition** is the problem of detecting and classifying objects into generic categories such as "cars", "trucks", "airplanes", "animals", or "human figures"
- **Appearances are highly variable within a category** because of shape variation, position in the visual field, scale, viewpoint, illumination, albedo, texture, background clutter, and occlusions.
- **Learning invariant representations is key.**
- **Understanding the neural mechanism behind invariant recognition is one of the main goals of Visual Neuroscience.**



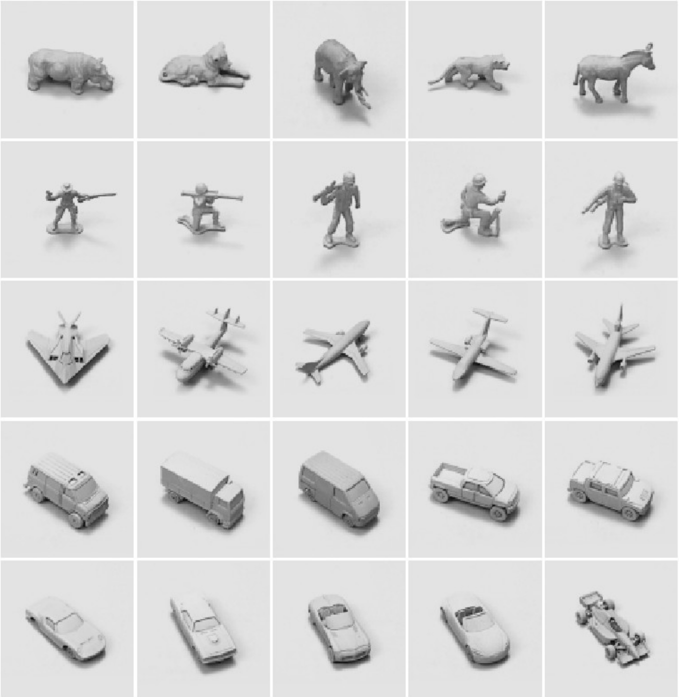
Why do we need "Deep" Architectures?

[Bengio & LeCun 2007]

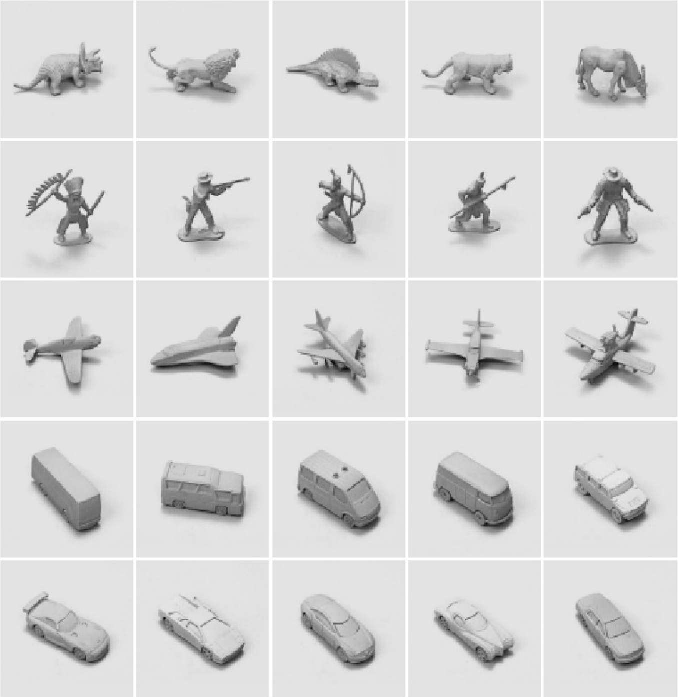
- **Conjecture: we won't solve the perception problem without solving the problem of learning in deep architectures [Hinton]**
  - Neural nets with lots of layers
  - Deep belief networks
  - Factor graphs with a "Markov" structure
- **We will not solve the perception problem with kernel machines**
  - Kernel machines are glorified template matchers
  - You can't handle complicated invariances with templates (you would need too many templates)
- **Many interesting functions are "deep"**
  - Any function can be approximated with 2 layers (linear combination of non-linear functions)
  - But many interesting functions a more efficiently represented with multiple layers
  - Stupid examples: binary addition

Generic Object Detection and Recognition with Invariance to Pose and Illumination

- **50** toys belonging to 5 categories: **animal, human figure, airplane, truck, car**
- **10** instance per category: 5 instances used for training, 5 instances for testing
- **Raw dataset: 972** stereo pair of each object instance. **48,600** image pairs total.
- **For each instance:**
- **18 azimuths**
  - 0 to 350 degrees every 20 degrees
- **9 elevations**
  - 30 to 70 degrees from horizontal every 5 degrees
- **6 illuminations**
  - on/off combinations of 4 lights
- **2 cameras (stereo)**
  - 7.5 cm apart
  - 40 cm from the object

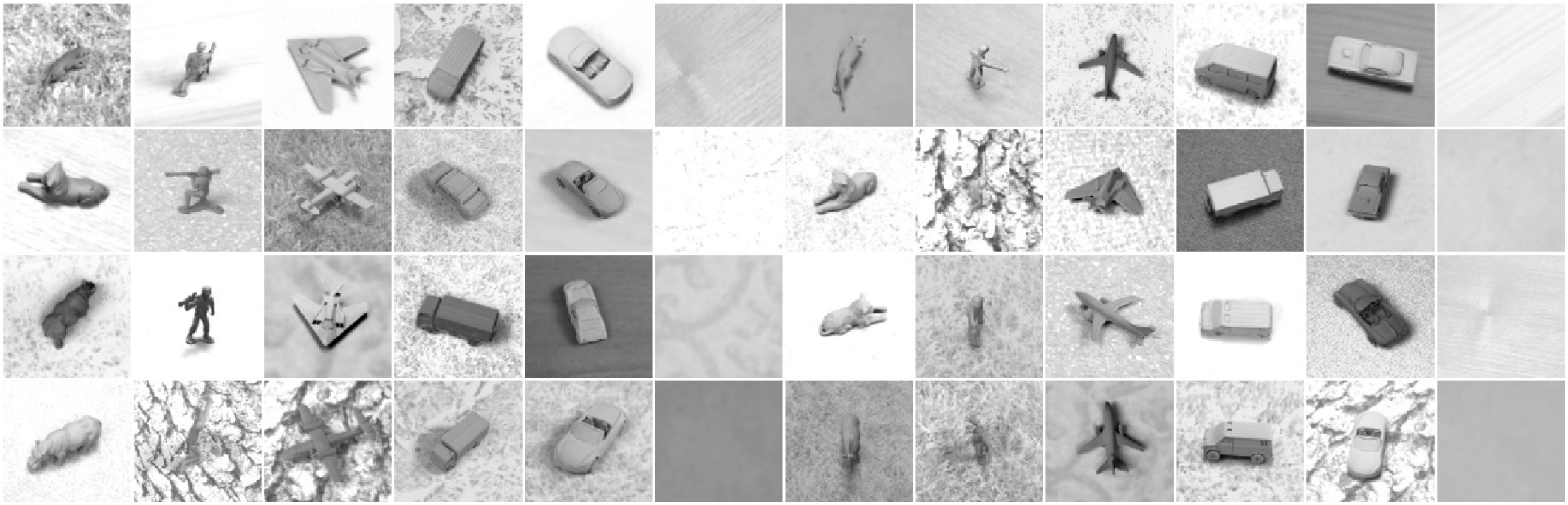


Training instances

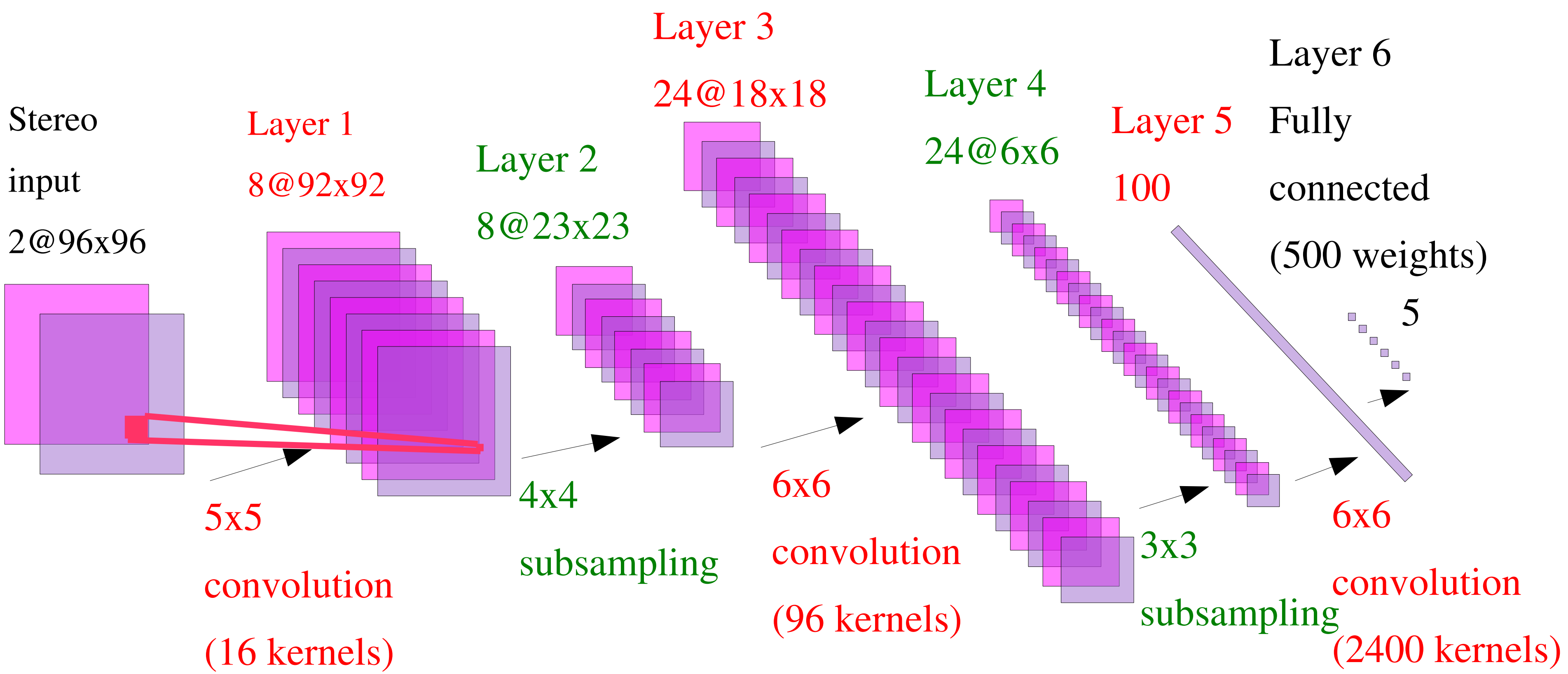


Test instances

Textured and Cluttered Datasets

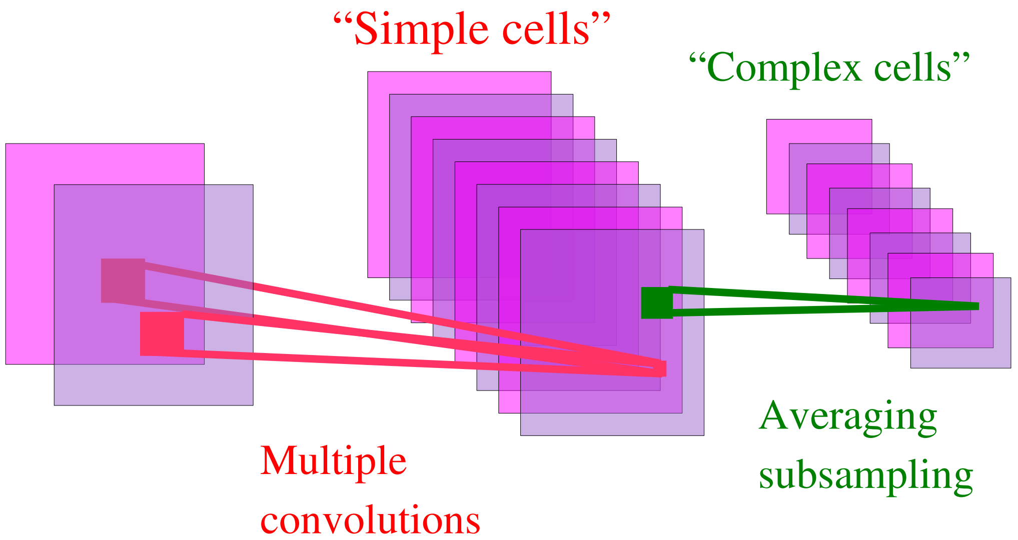


Convolutional Network

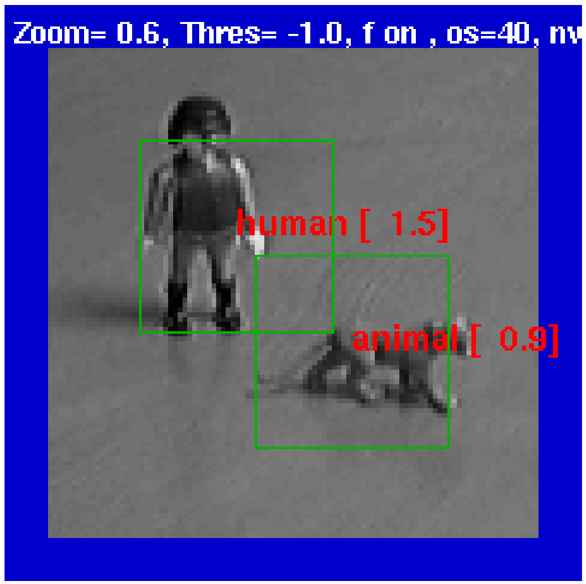


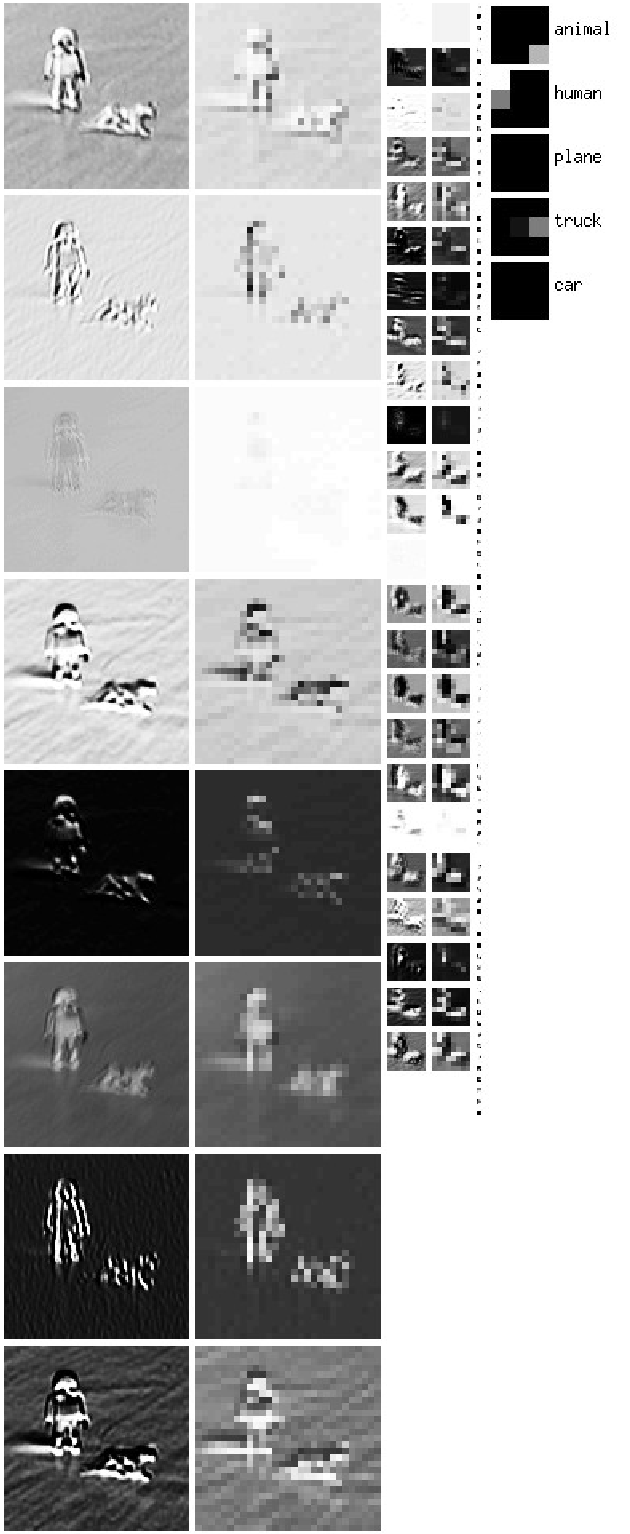
- **90,857 free parameters, 3,901,162 connections.**
- The architecture alternates convolutional layers (feature detectors) and subsampling layers (local feature pooling for invariance to small distortions).
- **The entire network is trained end-to-end** (all the layers are trained simultaneously).
- A gradient-based algorithm is used to minimize a supervised loss function.

Alternated Convolutions and Subsampling



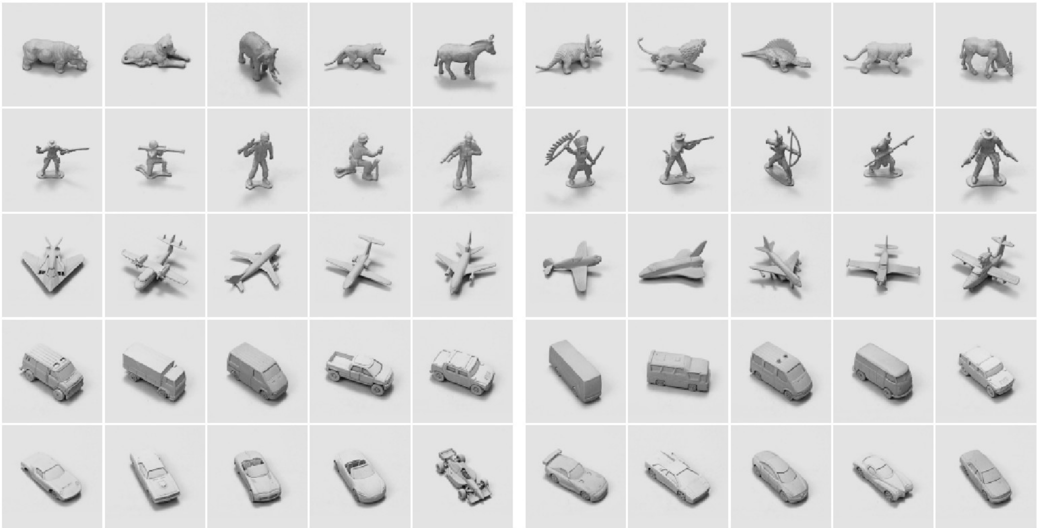
- Local features are extracted everywhere.
- averaging/subsampling layer builds robustness to variations in feature locations.
- Hubel/Wiesel'62, Fukushima'71, LeCun'89, Riesenhuber & Poggio'02, Ullman'02,....





Normalized-Uniform Set: Error Rates

- Linear Classifier on raw stereo images: 30.2% error.
- K-Nearest-Neighbors on raw stereo images: 18.4% error.
- K-Nearest-Neighbors on PCA-95: 16.6% error.
- Pairwise SVM on 96x96 stereo images: 11.6% error
- Pairwise SVM on 95 Principal Components: 13.3% error.
- Convolutional Net on 96x96 stereo images: 5.8% error.



Training instances Test instances

Normalized-Uniform Set: Learning Times

	SVM	Conv Net				SVM/Conv
test error	11.6%	10.4%	6.2%	5.8%	6.2%	5.9%
train time (min*GHz)	480	64	384	640	3,200	50+
test time per sample (sec*GHz)	0.95	0.03				0.04+
#SV	28%					28%
parameters	σ=2,000 C=40					dim=80 σ=5 C=0.01

SVM: using a parallel implementation by Graf, Durdanovic, and Cosatto (NEC Labs)

Chop off the last layer of the convolutional net and train an SVM on it

Jittered-Cluttered Dataset

- **Jittered-Cluttered Dataset:**
- **291,600** tereo pairs for training, **58,320** for testing
- Objects are jittered: position, scale, in-plane rotation, contrast, brightness, backgrounds, distractor objects,...
- Input dimension: 98x98x2 (approx 18,000)

Experiment 2: Jittered-Cluttered Dataset

- 291,600 training samples, 58,320 test samples
- SVM with Gaussian kernel 43.3% error
- Convolutional Net with binocular input: 7.8% error
- Convolutional Net + SVM on top: 5.9% error
- Convolutional Net with monocular input: 20.8% error
- Smaller mono net (DEMO): 26.0% error
- Dataset available from http://www.cs.nyu.edu/~yann

Jittered-Cluttered Dataset

	SVM	Conv Net			SVM/Conv
test error	43.3%	16.38%	7.5%	7.2%	5.9%
train time (min*GHz)	10,944	420	2,100	5,880	330+
test time per sample (sec*GHz)	2.2	0.04			0.06+
#SV	5%				2%
parameters	σ=10^4 C=40				dim=100 σ=5 C=1

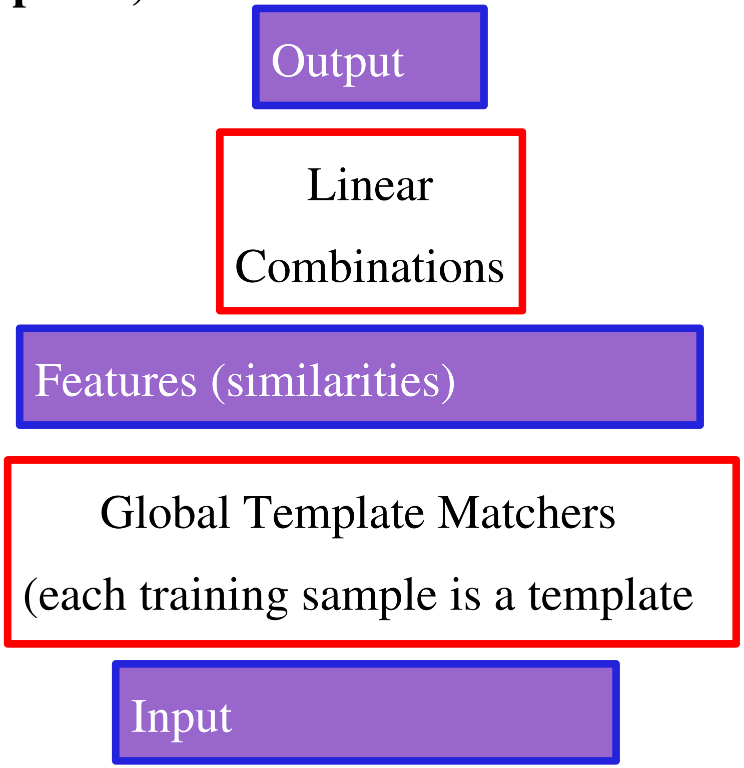
OUCH!

The convex loss, VC bounds and representers theorems don't seem to help

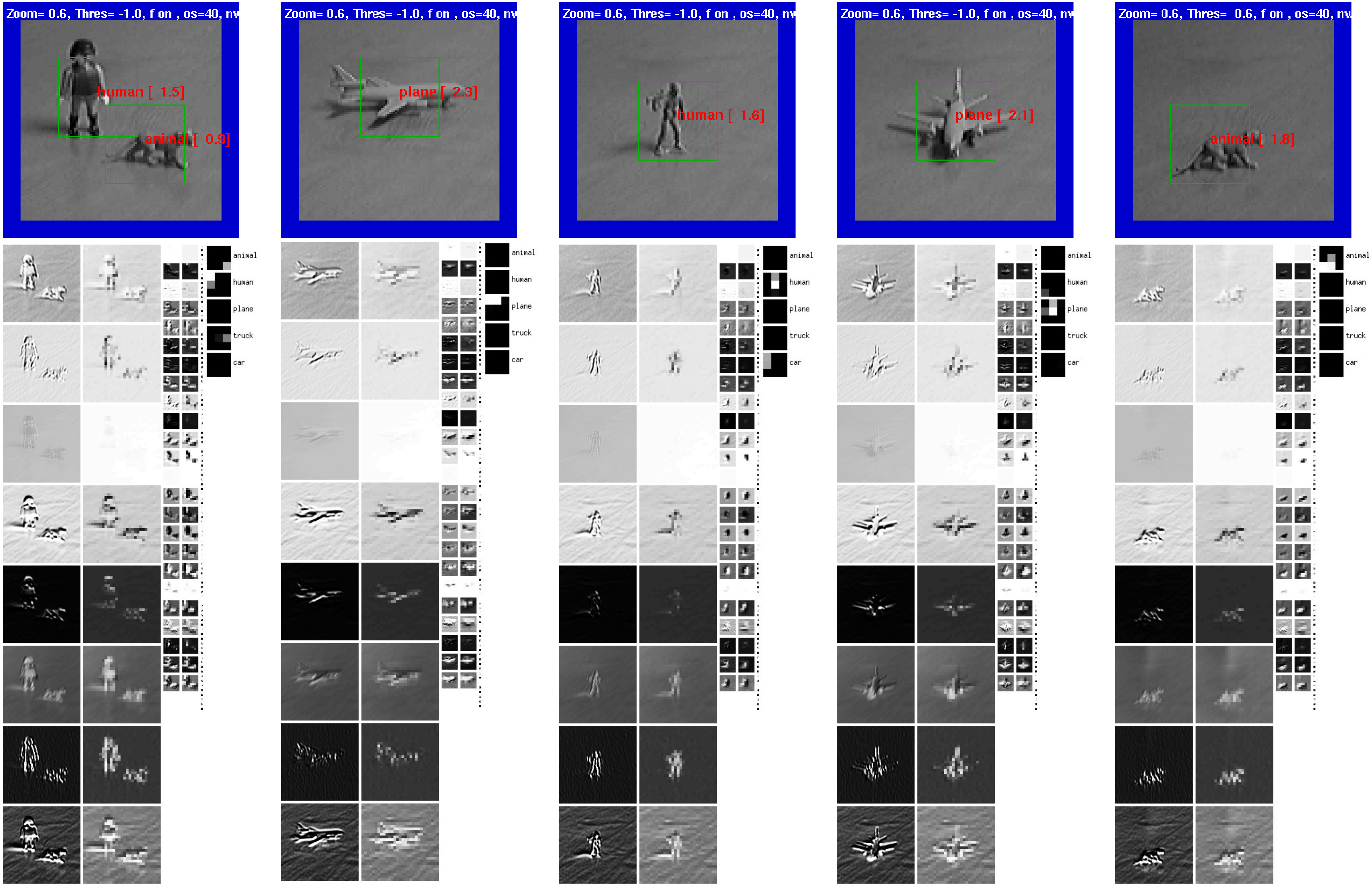
Chop off the last layer, and train an SVM on it it works!

What's wrong with K-NN and SVMs?

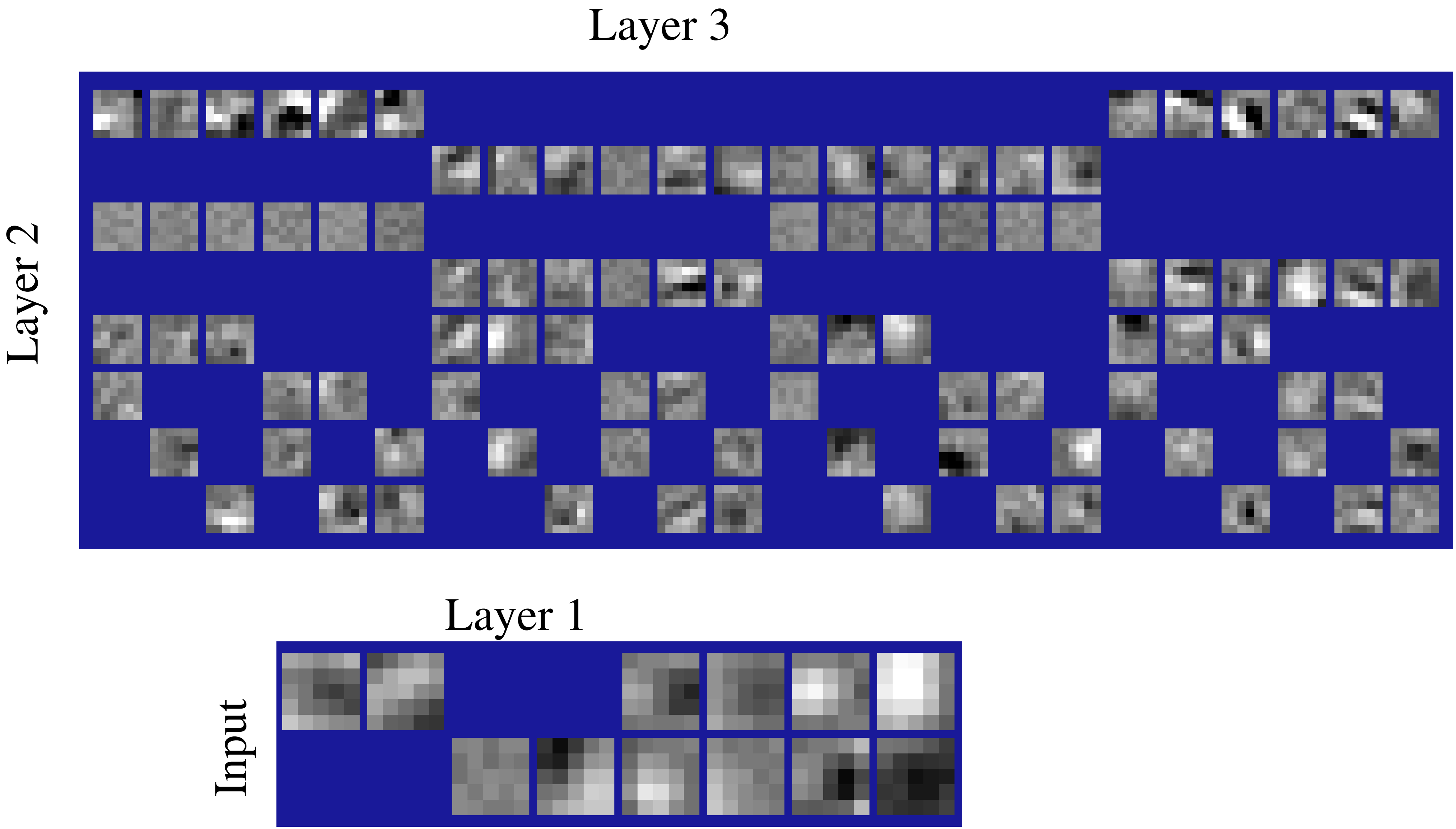
- **K-NN and SVM with Gaussian kernels are based on matching global templates**
- **Both are "shallow" architectures**
- **There is now way to learn invariant recognition tasks with such naïve architectures (unless we use an impractically large number of templates).**
  - **The number of necessary templates grows exponentially with the number of dimensions of variations.**
  - **Global templates are in trouble when the variations include: category, instance shape, configuration (for articulated object), position, azimuth, elevation, scale, illumination, texture, albedo, in-plane rotation, background luminance, background texture, background clutter, ...**



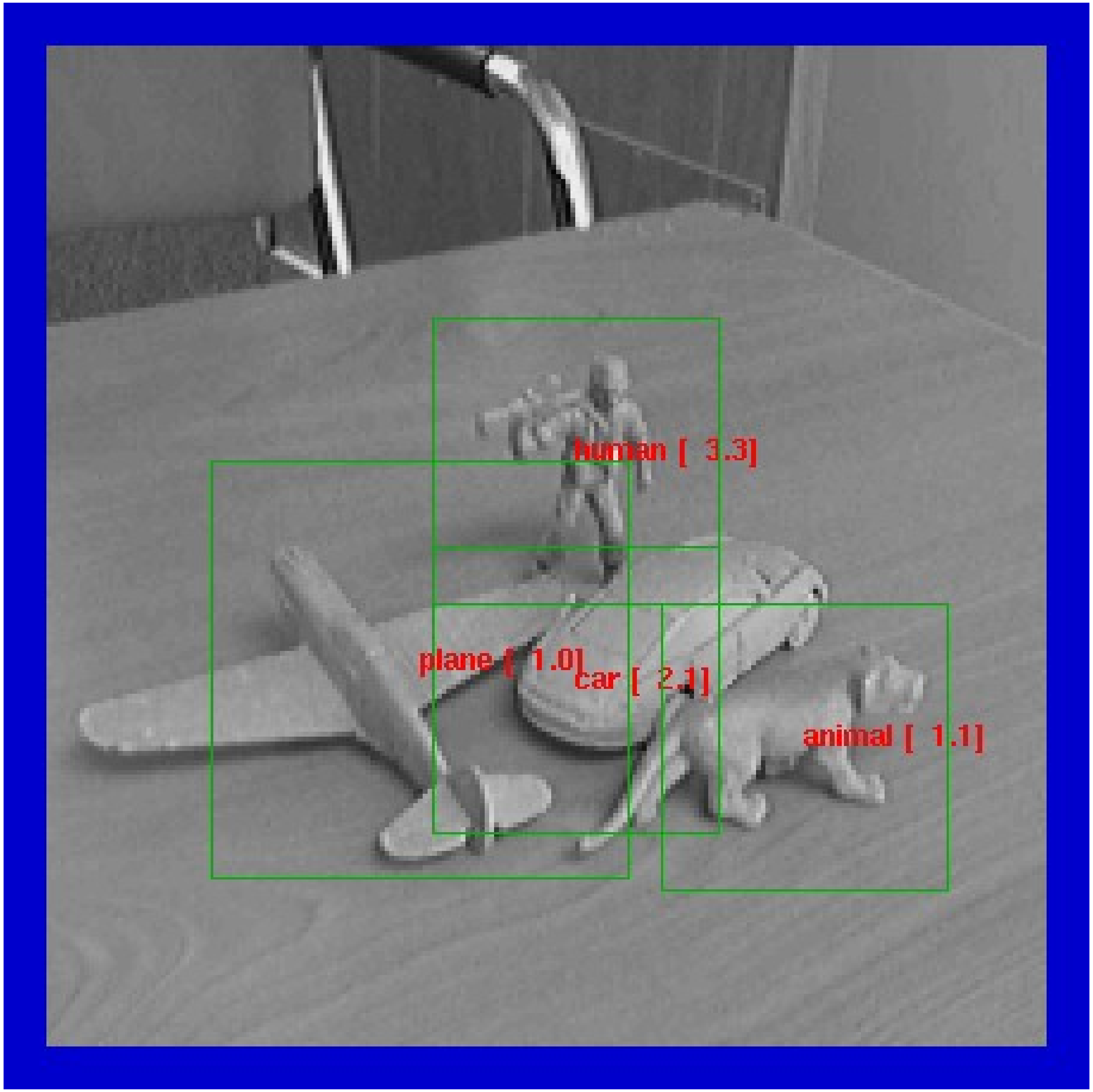
Examples (Monocular Mode)



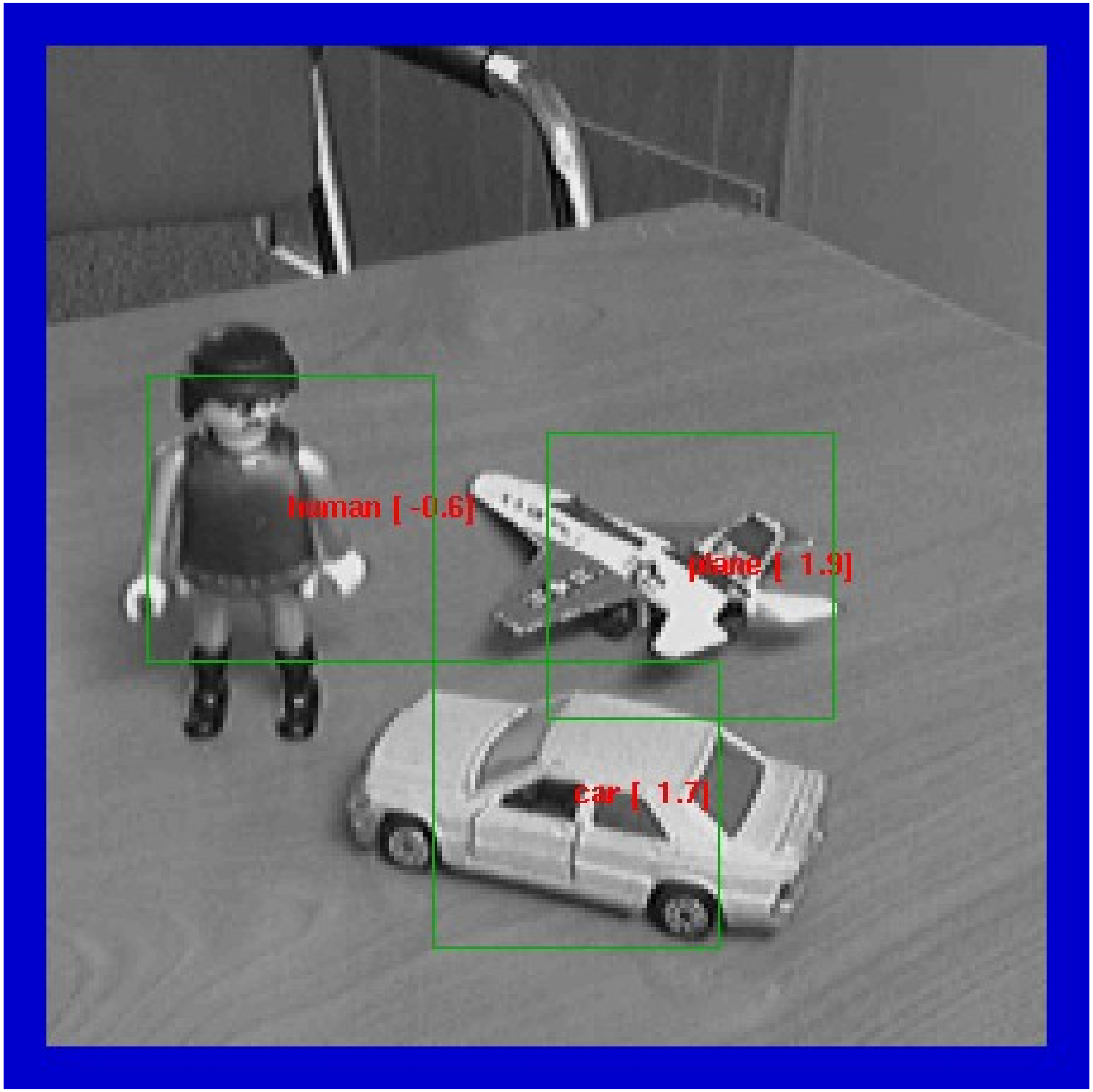
Learned Features



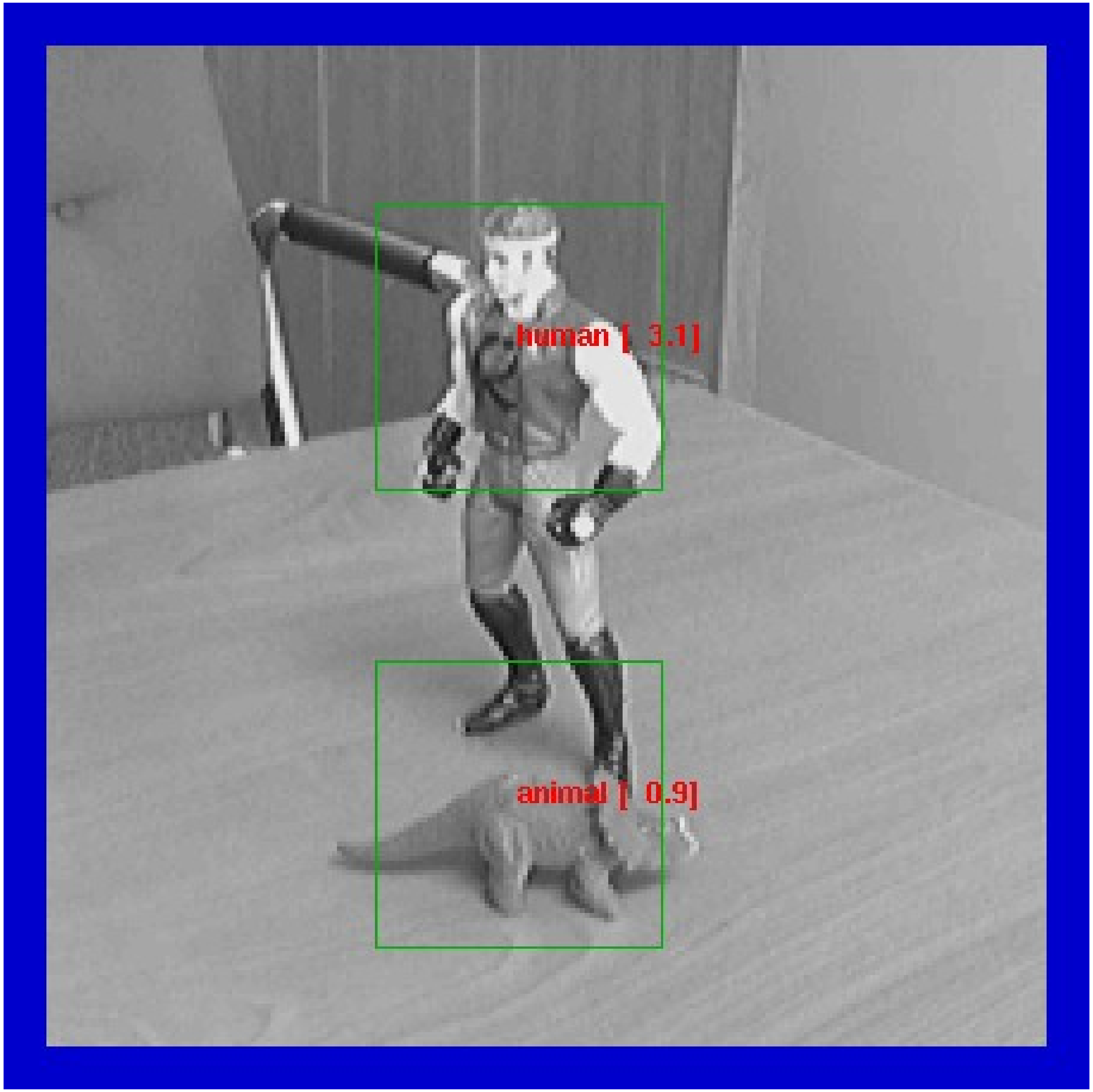
Examples (Monocular Mode)



Examples (Monocular Mode)

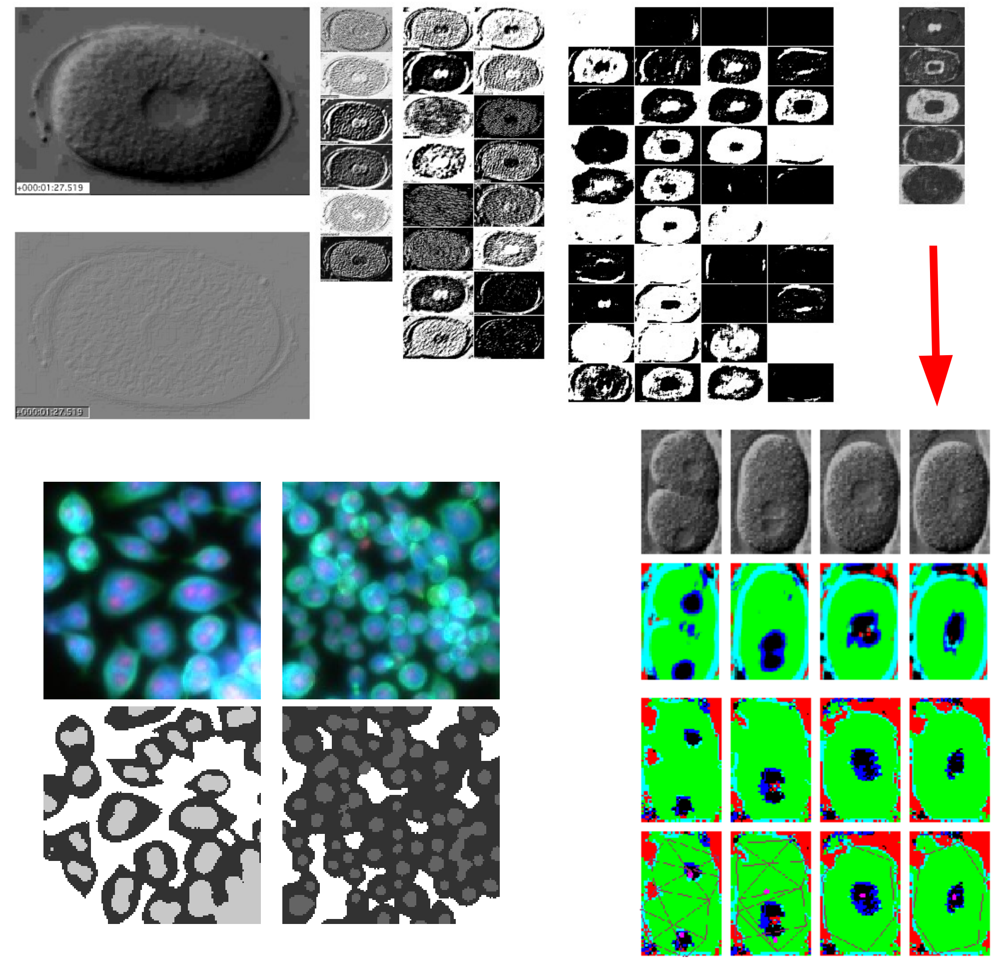


Examples (Monocular Mode)

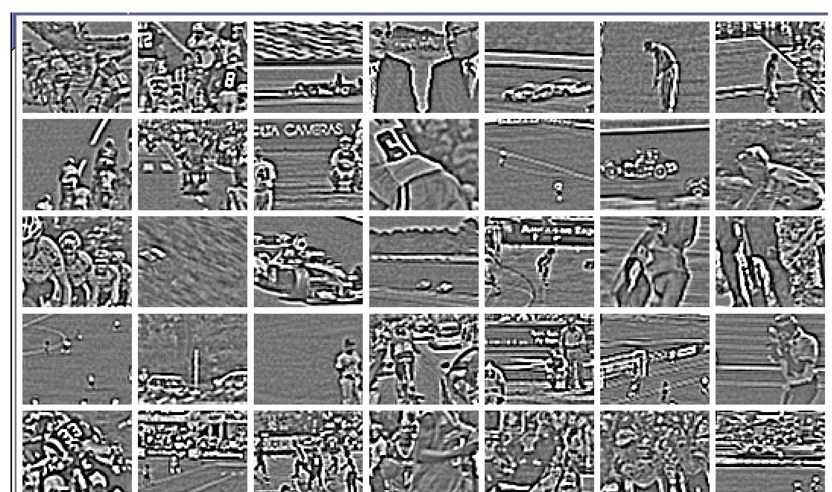


Other Applications of Convolutional Nets:

- **Analyzing Biological Images:**
  - Subcellular structure classification
  - Cancer cell detection

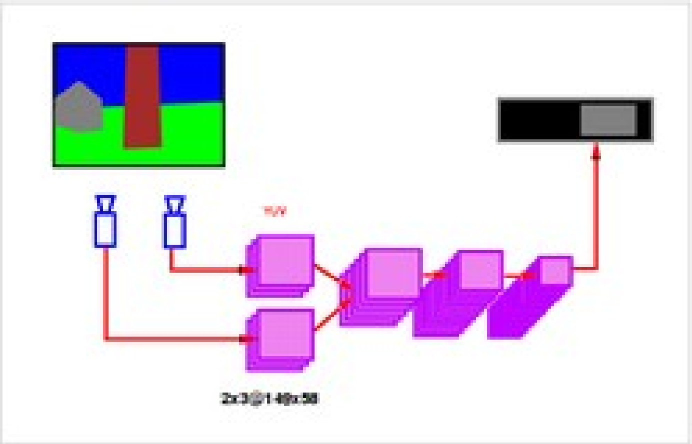


- **Classifying sports TV snapshots**
  - 7 categories: auto racing, baseball, basketball, bicycle, golf, soccer, football.
  - 61% correct frame by frame

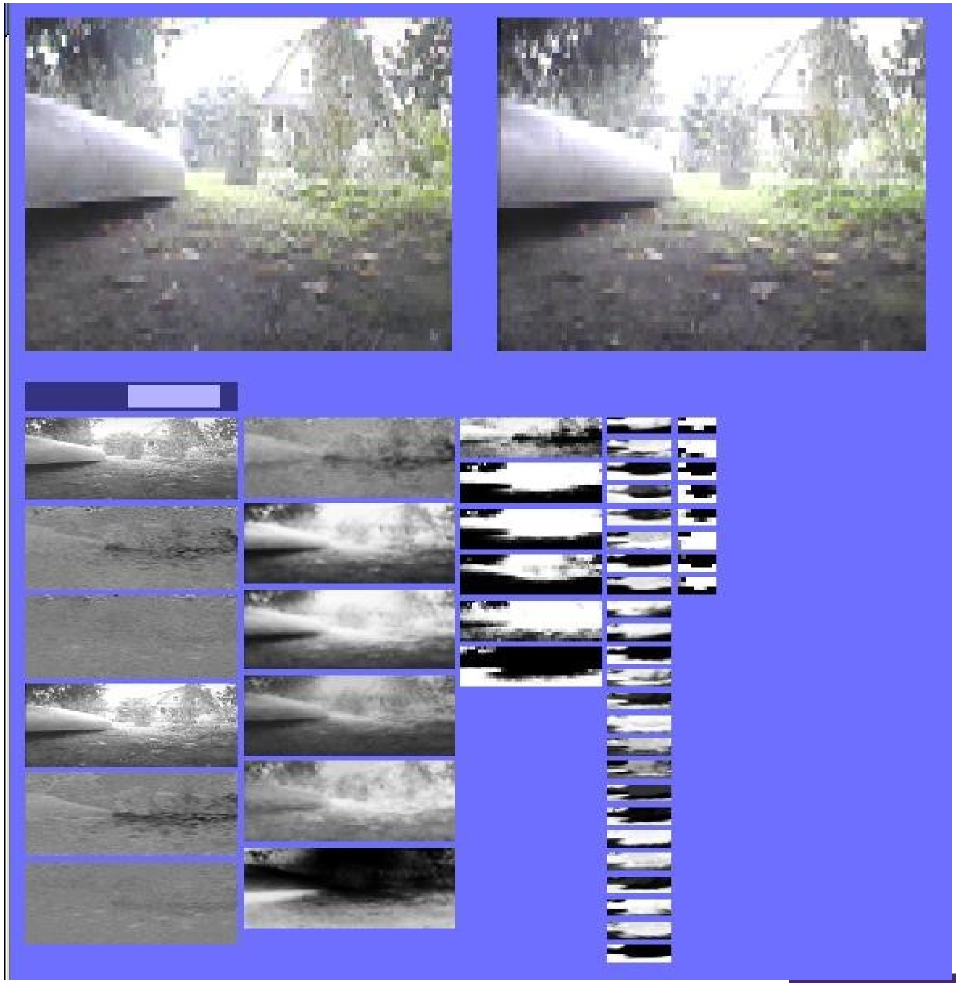


- **RF signal processing**
- **Face Recognition**

Visual Navigation for a Mobile Robot



- Mobile robot with two cameras
- The convolutional net is trained to emulate a human driver from recorded sequences of video + human-provided steering angles.
- The network maps stereo images to steering angles for obstacle avoidance



Supervised Convolutional Nets: Pros and Cons

- **Convolutional nets can be trained to perform a wide variety of visual tasks.**
  - Global supervised gradient descent can produce parsimonious architectures
- **BUT: they require lots of labeled training samples**
  - 60,000 samples for handwriting
  - 120,000 samples for face detection
  - 25,000 to 350,000 for object recognition
- **Since low-level features tend to be non task specific, we should be able to learn them unsupervised.**
- **Hinton has shown that layer-by-layer unsupervised "pre-training" can be used to initialize "deep" architectures**
  - [Hinton & Shalakhutdinov, Science 2006]
- **Can we use this idea to reduce the number of necessary labeled examples.**

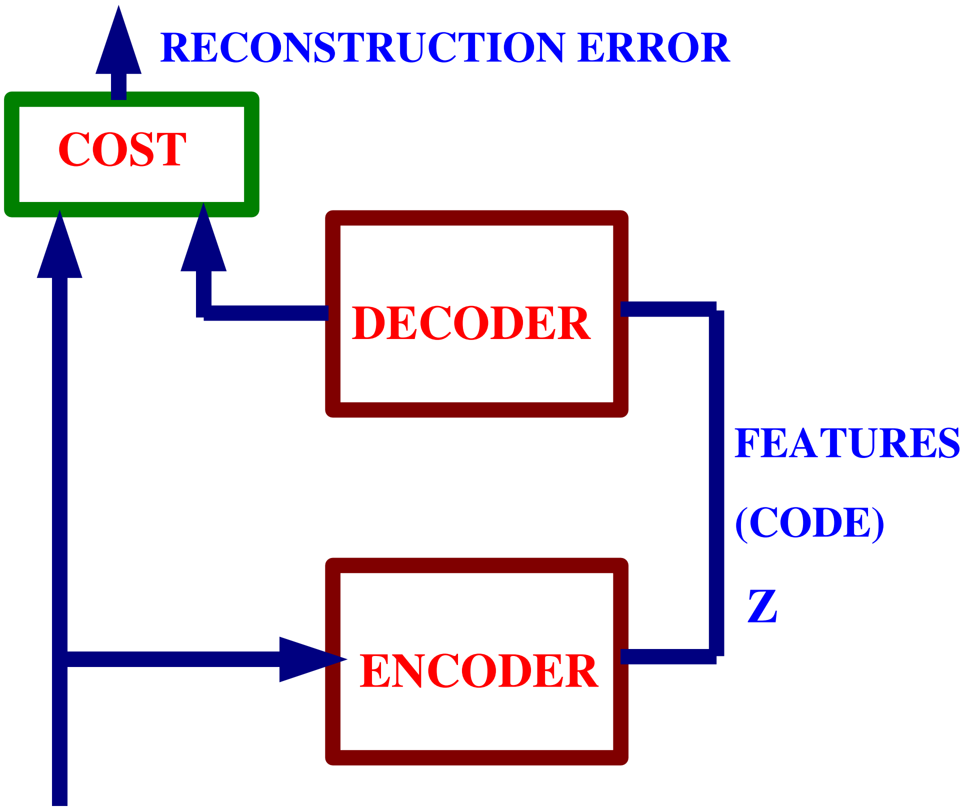
Unsupervised Learning of Sparse-Overcomplete Features

[Ranzato, Poultney, Chopra, LeCun, NIPS 2006]

[Ranzato et al. CVPR 2007]

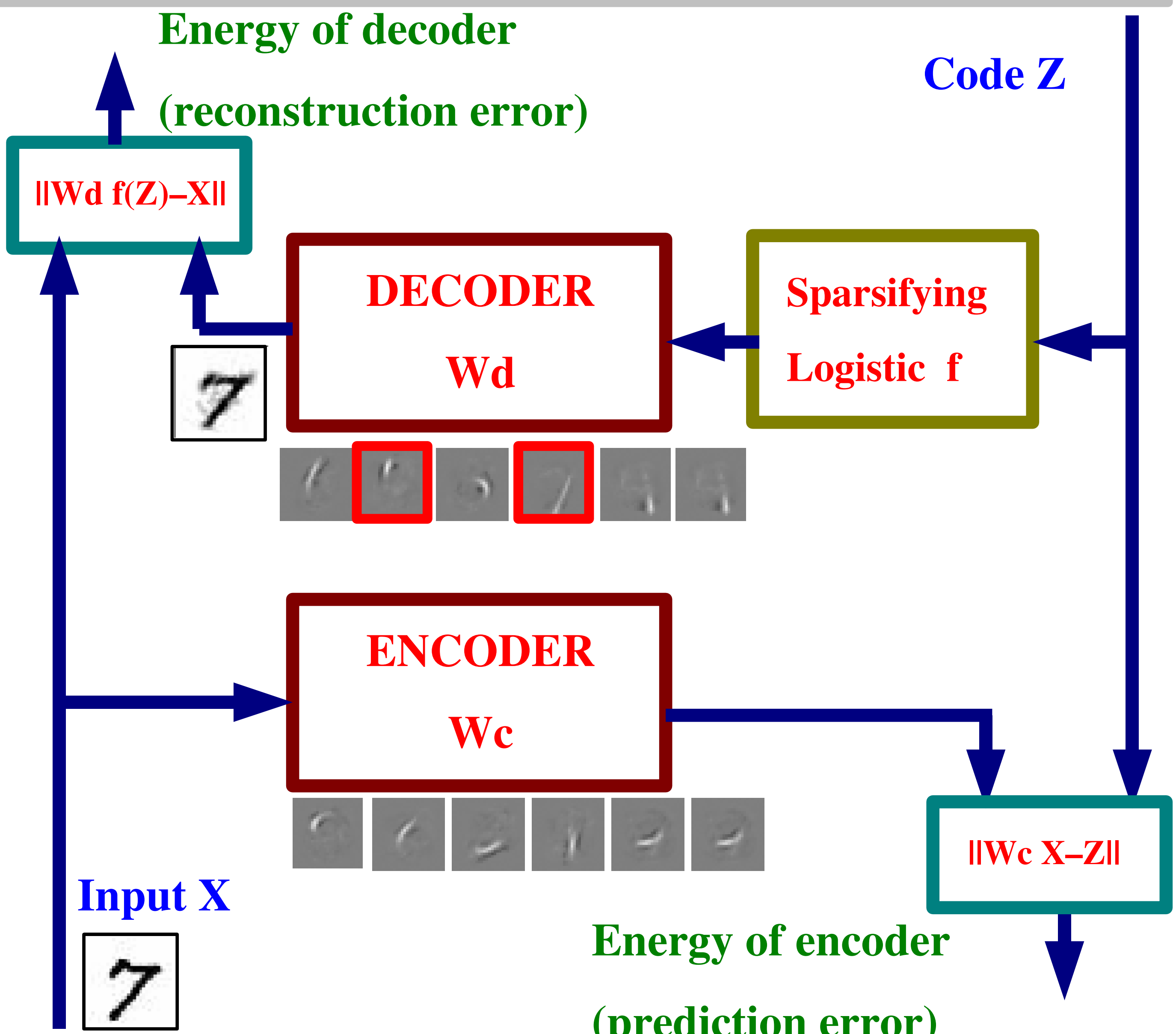
Layer-by-Layer Unsupervised Training

- **A principle on which unsupervised algorithms can be built is reconstruction of the input from a code (feature vector)**
  - reconstruction from compact feature vectors (e.g. PCA).
  - reconstruction from sparse overcomplete feature vectors (Olshausen & Field 1997)

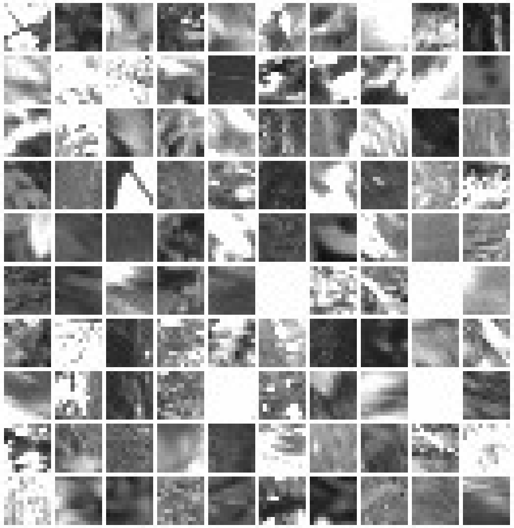


Encoder/Decoder Architecture for learning Sparse Feature Representations

- **Algorithm:**
  - 1. find the code Z that minimizes the reconstruction error AND is close to the encoder output
  - 2. Update the weights of the decoder to decrease the reconstruction error
  - 3. Update the weights of the encoder to decrease the prediction error



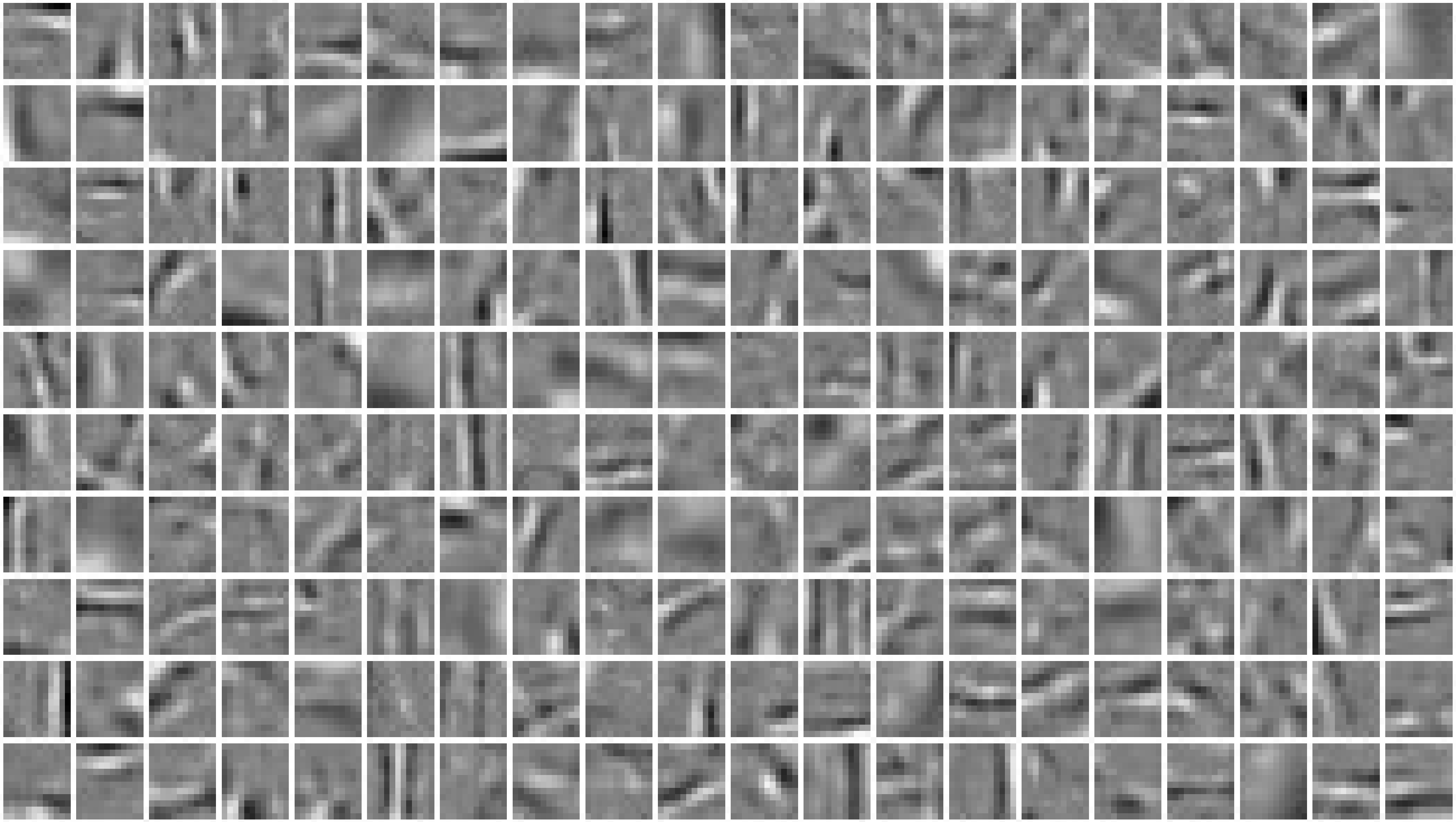
Training on natural image patches



Berkeley data set

- 100,000 12x12 patches
- 200 units in the code
- η β 0.02
- 1
- learning rate 0.001
- L1 regularizer 0.001
- fast convergence: < 30min.

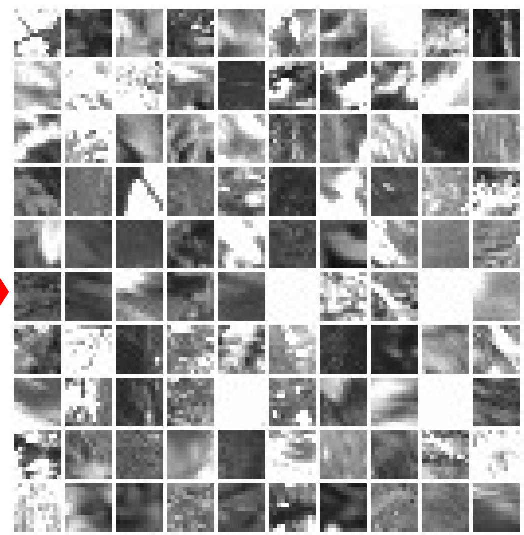
Natural image patches: Filters



200 decoder filters (reshaped columns of matrix $\mathbf{W_d}$)

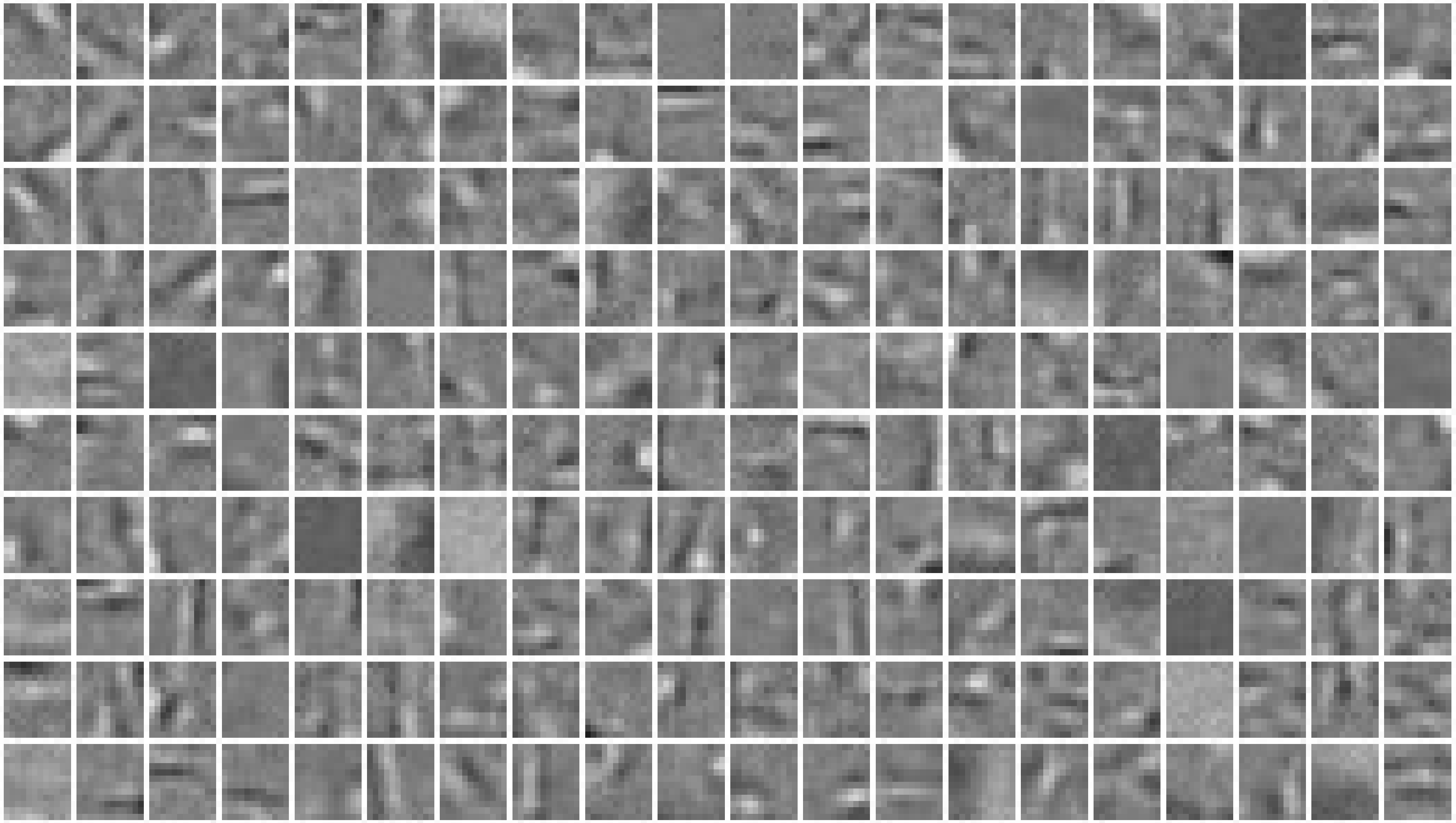
Natural image patches - Forest

Forest data set



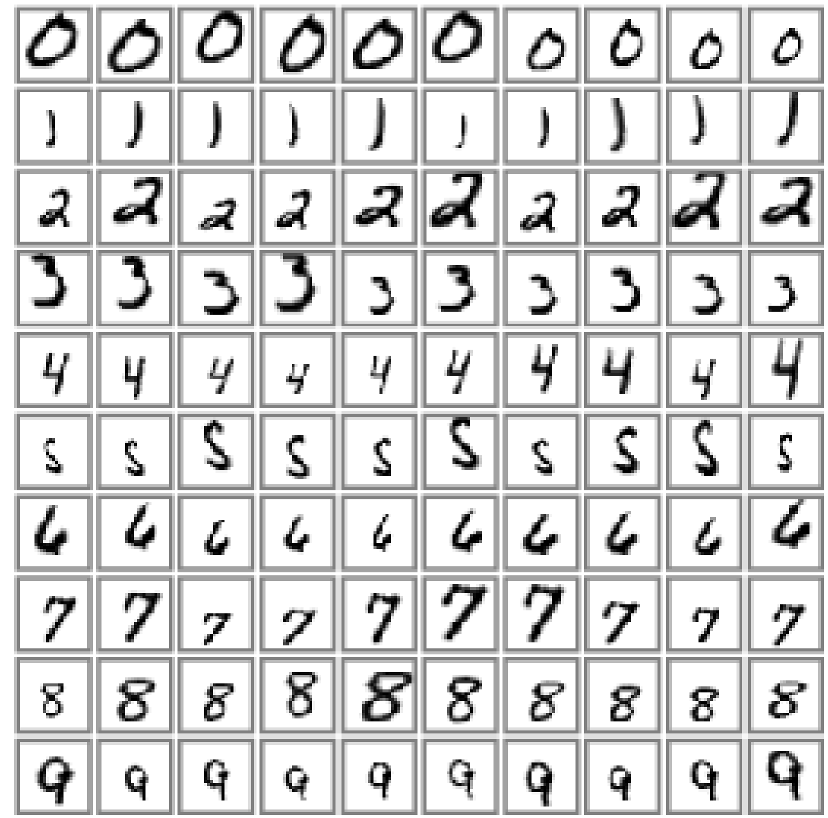
- 100,000 12x12 patches
- 200 units in the code
- η β 0.02
- 1
- learning rate 0.001
- L1, L2 regularizer 0.001
- fast convergence: < 30min.

Natural image patches - Forest



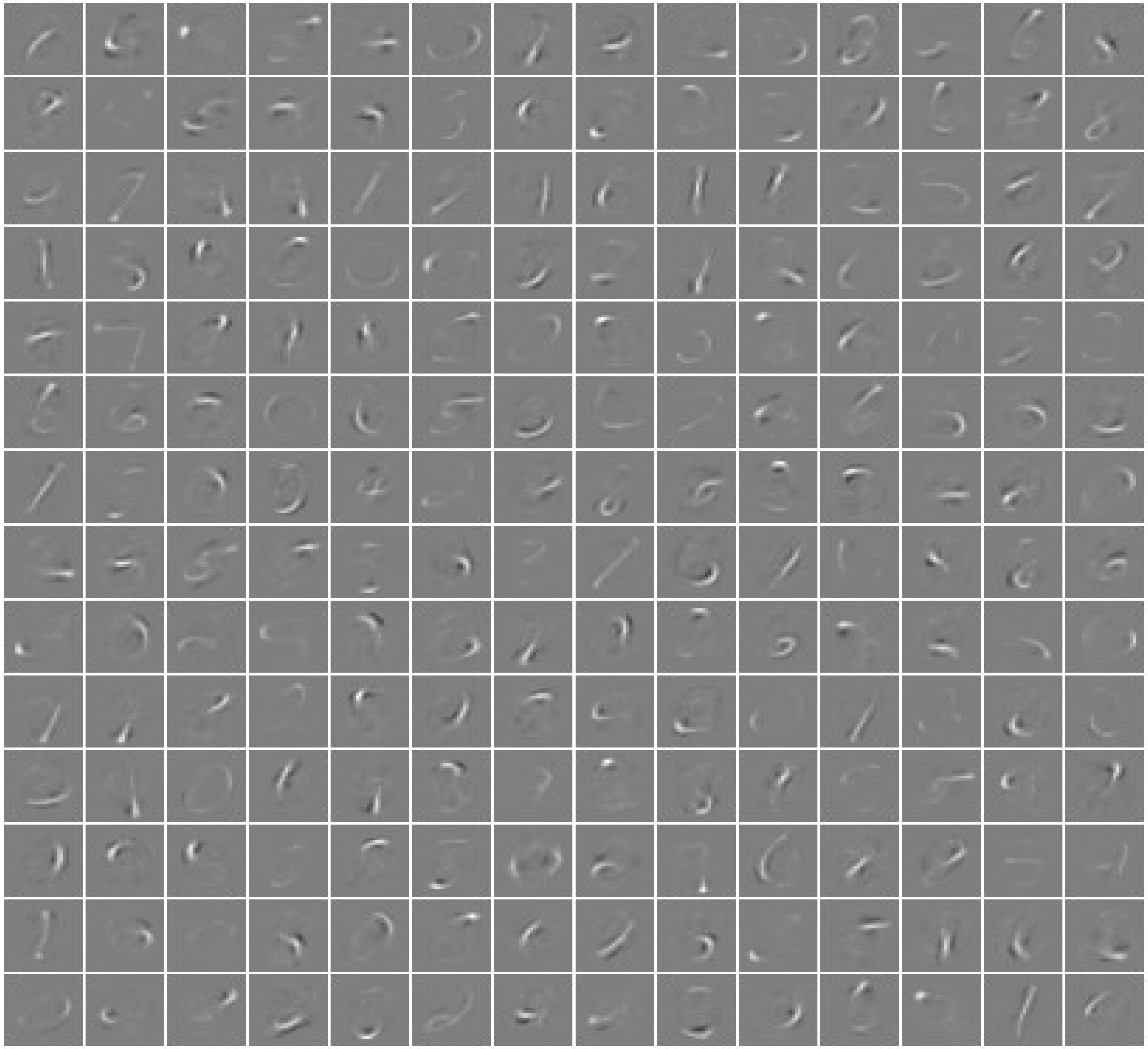
200 decoder filters (reshaped columns of matrix $\mathbf{W_d}$)

MNIST Dataset



- Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples

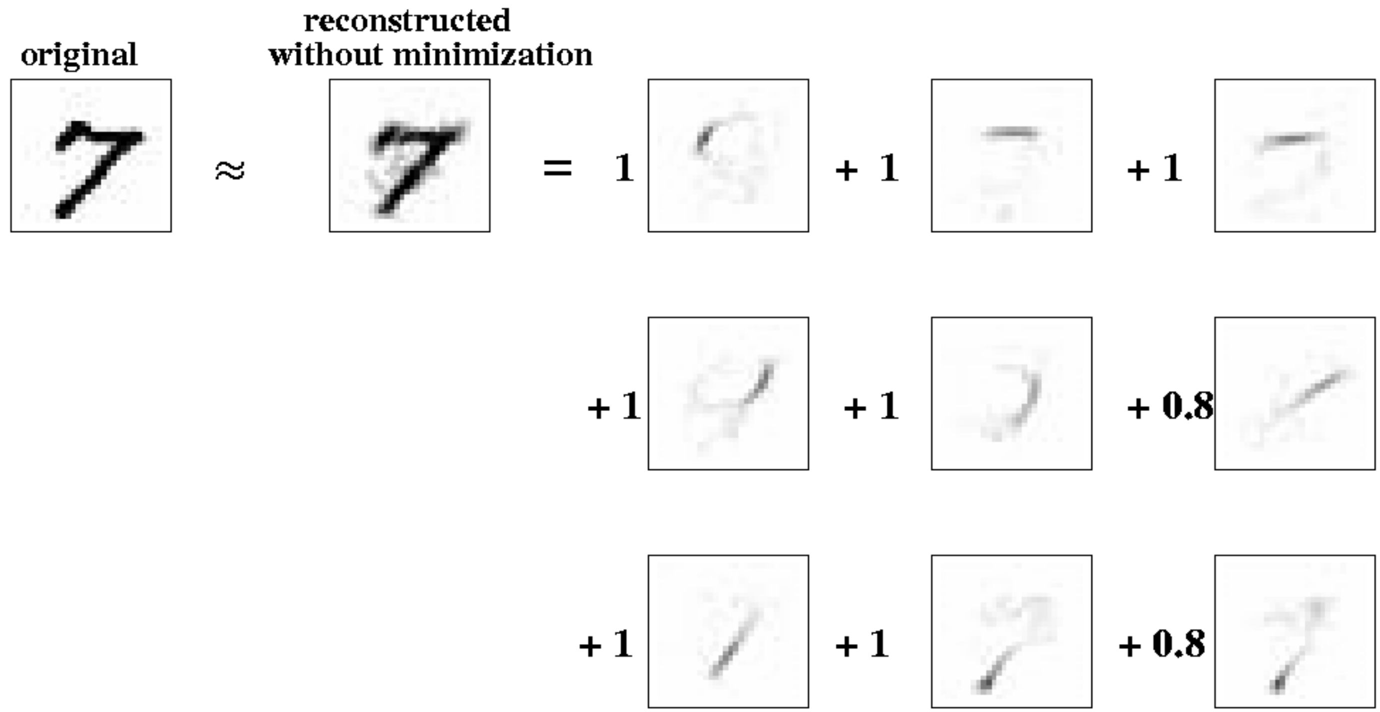
Training on handwritten digits

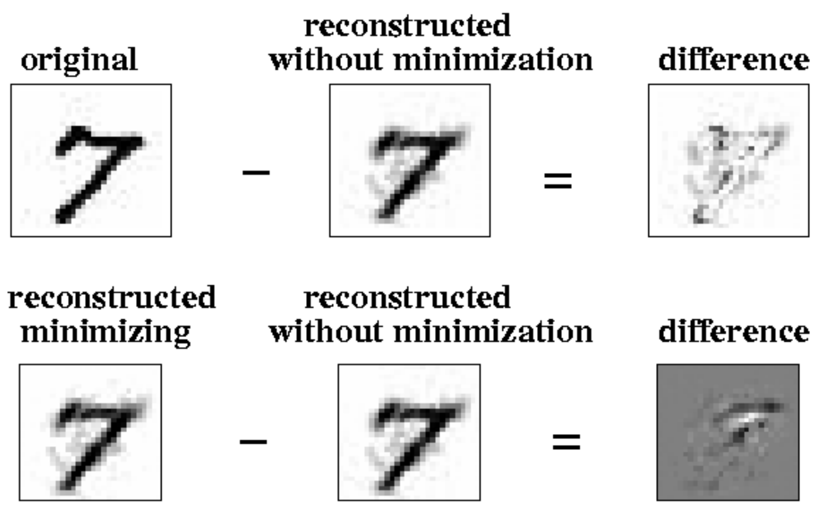


Encoder *direct* filters

- 60,000 28x28 images
- 196 units in the code
- η 0.01
- β 1
- learning rate 0.001
- L1, L2 regularizer 0.005

Handwritten digits - MNIST





forward propagation through encoder and decoder

after training there is no need to minimize in code space

Denoising

original image

noisy image

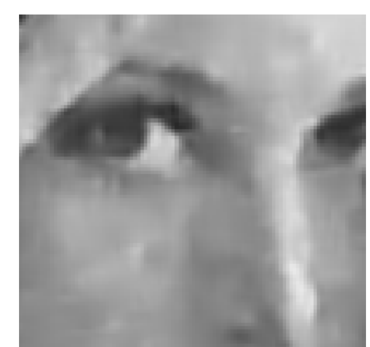
PSNR 14.15dB

(std. dev. noise 50)

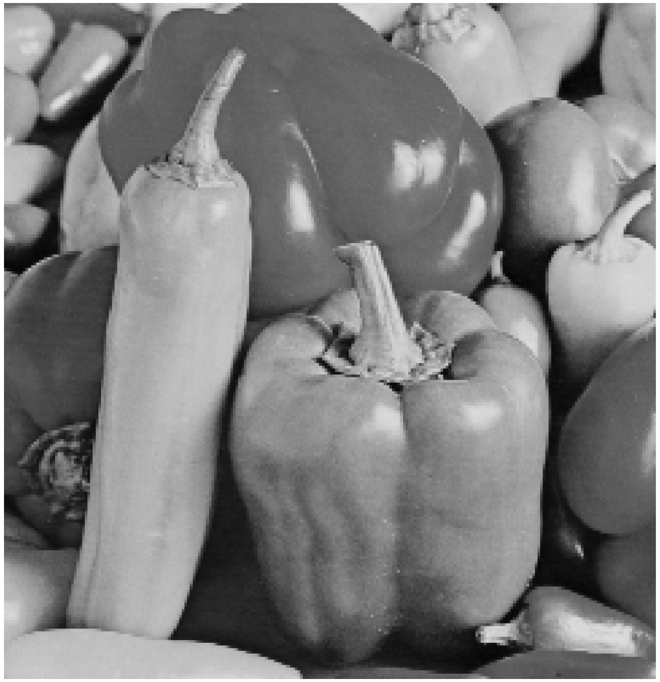
denoised image

PSNR 27.88dB

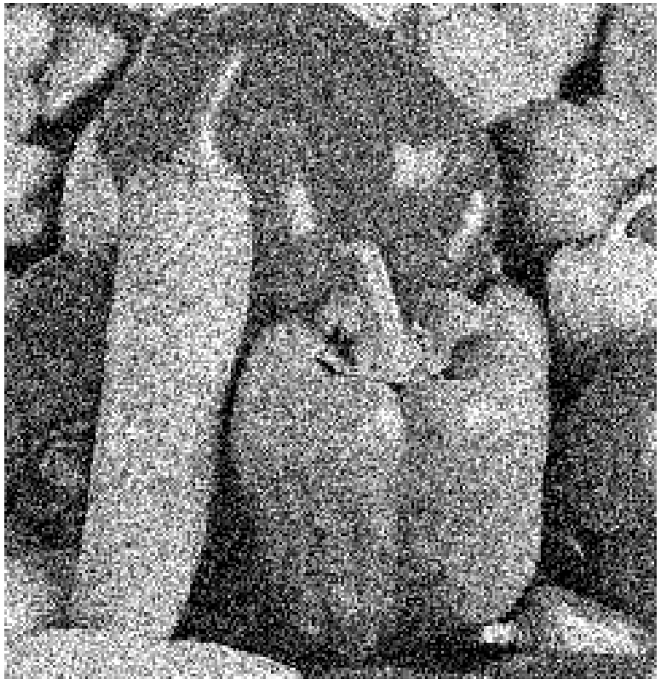
ZOOM ->



Denoising



original image



noisy image

PSNR 14.15dB

(std. dev. noise 50)

denoised image

PSNR 26.50dB

ZOOM ->

Denoising

s.d. / PSNR	Lena				Barbara				Boat				House				Peppers			
50 / 14.15	27.86	**28.61**	27.79	26.49	23.46	**25.48**	25.47	23.15	26.02	**26.38**	25.95	24.53	27.85	**28.26**	27.95	26.74	**26.35**	25.90	26.13	24.52
75 / 10.63	25.97	**26.84**	25.80	24.13	22.46	**23.65**	23.01	21.36	24.31	**24.79**	23.98	22.48	25.77	**26.41**	25.22	24.13	**24.56**	24.00	23.69	21.68
100 / 8.13	24.49	**25.64**	24.46	21.87	21.77	**22.61**	21.89	19.77	23.09	**23.75**	22.81	20.80	24.20	**25.11**	23.71	21.66	**23.04**	22.66	21.75	19.60

Comparison between:

- our method [first column]
- Portilla et al. IEEE Trans. Image Processing (2003) [second column]
- Elad and Aharon CVPR 2006 [third column]
- Roth and Black CVPR 2005 [fourth column]

Training The Layers of a Convolutional Net Unsupervised

- Extract windows from the MNIST images
- Train the sparse encoder/decoder on those windows
- Use the resulting encoder weights as the convolution kernels of a convolution network
- Repeat the process for the second layer
- Train the resulting network supervised.

Best Results on MNIST (from raw images: no preprocessing)

CLASSIFIER	DEFORMATION	ERROR	Reference
Knowledge-free methods			
2-layer NN, 800 HU, CE		1.60	Simard et al., ICDAR 2003
3-layer NN, 500+300 HU, CE, reg		1.53	Hinton, in press, 2005
SVM, Gaussian Kernel		1.40	Cortes 92 + Many others
Unsupervised Stacked RBM + backprop		0.95	Hinton, Neur Comp 2006
Convolutional nets			
Convolutional net LeNet-5,		0.80	Ranzato et al. NIPS 2006
Convolutional net LeNet-6,		0.70	Ranzato et al. NIPS 2006
Conv. net LeNet-6- + unsup learning		0.60	Ranzato et al. NIPS 2006
Training set augmented with Affine Distortions			
2-layer NN, 800 HU, CE	Affine	1.10	Simard et al., ICDAR 2003
Virtual SVM deg-9 poly	Affine	0.80	Scholkopf
Convolutional net, CE	Affine	0.60	Simard et al., ICDAR 2003
Training et augmented with Elastic Distortions			
2-layer NN, 800 HU, CE	Elastic	0.70	Simard et al., ICDAR 2003
Convolutional net, CE	Elastic	0.40	Simard et al., ICDAR 2003
Conv. net LeNet-6- + unsup learning	Elastic	0.39	Ranzato et al. NIPS 2006

Training Convolutional Filters

CLASSIFICATION EXPERIMENTS

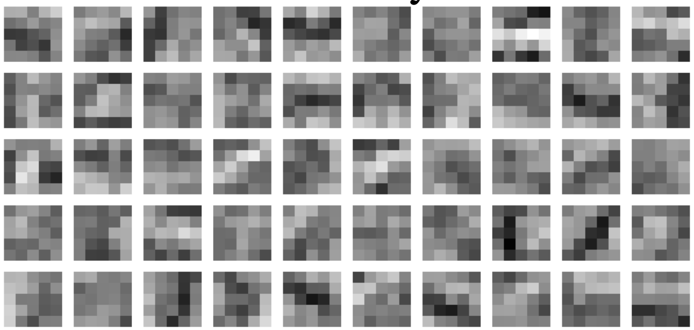
IDEA: improving supervised learning by pre-training
with the unsupervised method (*)

sparse representations & *lenet6* (1->50->50->200->10)

- The baseline: *lenet6* initialized randomly

Test error rate: 0.70%. Training error rate: 0.01%.

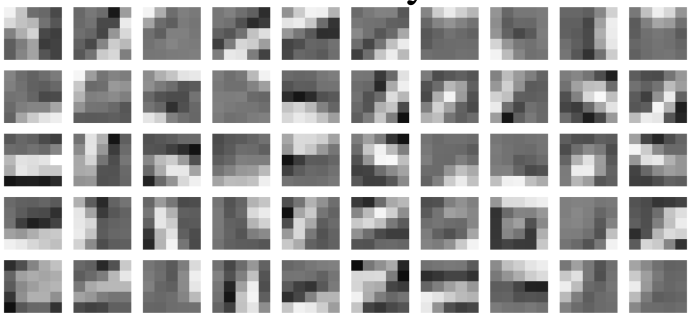
filters in first conv. layer



- *Experiment 1*
  - Train on 5x5 patches to find 50 features
  - Use the scaled filters in the encoder to initialize the kernels in the first convolutional layer

Test error rate: 0.60%. Training error rate: 0.00%.

filters in first conv. layer

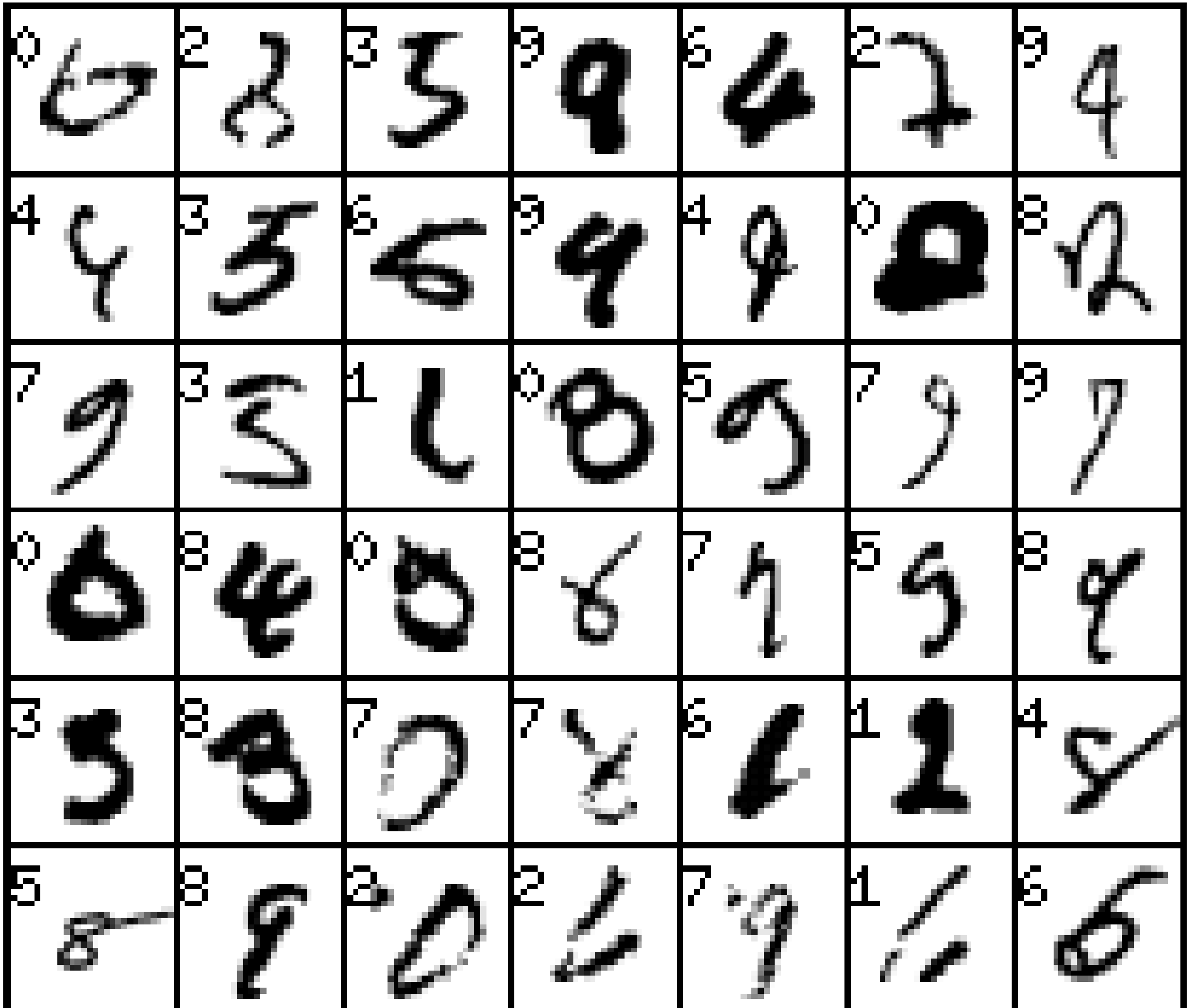


- *Experiment 2*
  - Same as experiment 1, but training set augmented by elastically distorted digits (random initialization gives test error rate equal to 0.49%).

Test error rate: 0.39%. Training error rate: 0.23%.

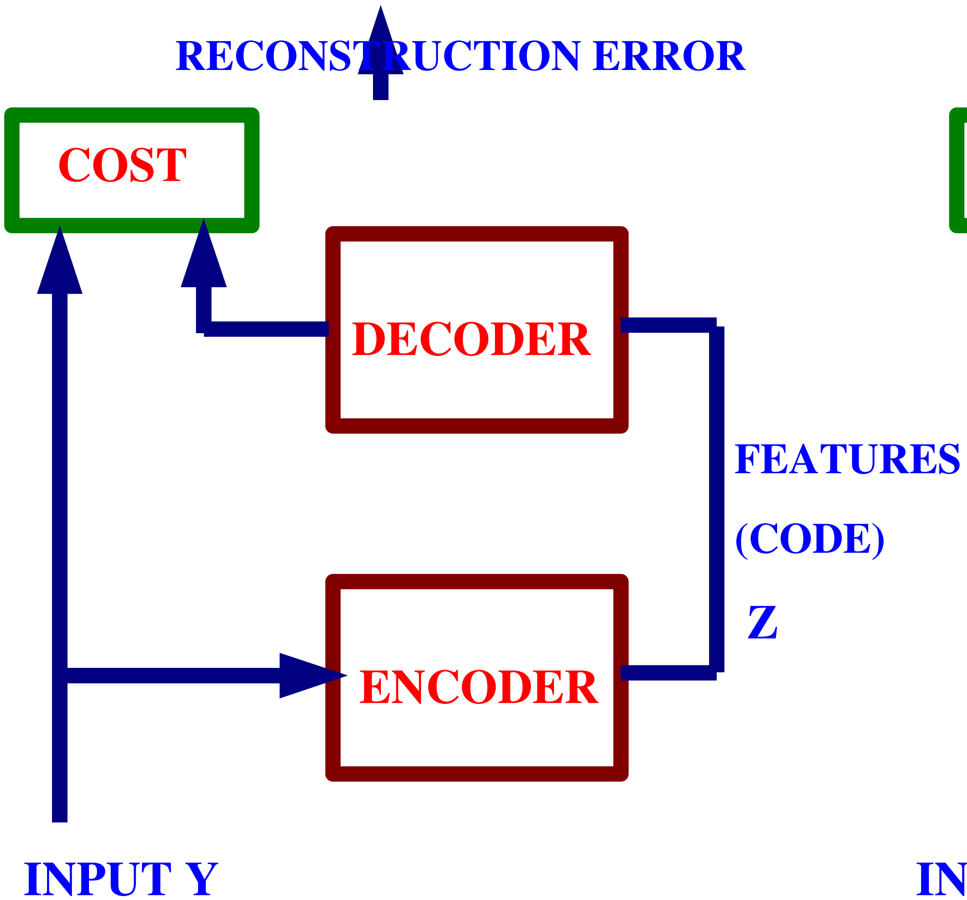
(*)[Hinton, Osindero, Teh "*A fast learning algorithm for deep belief nets*" Neural Computaton 2006]

MNIST Errors (0.42% error)

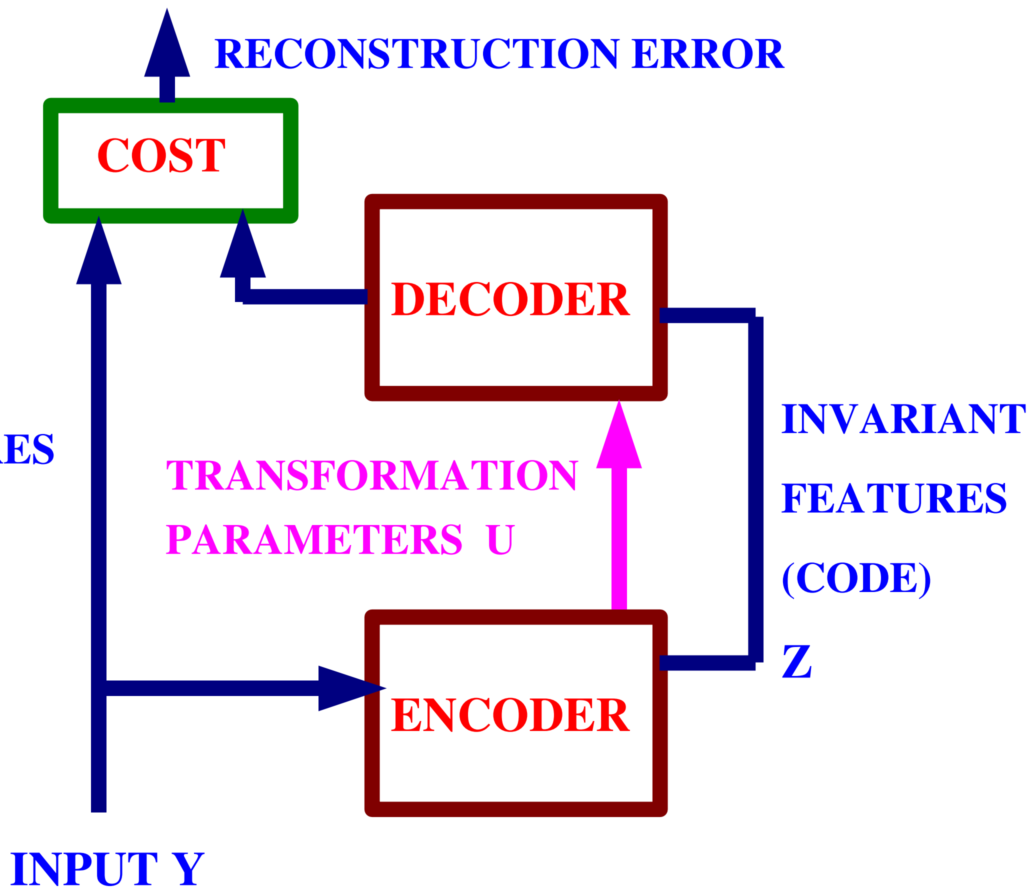


Learning Invariant Feature Hierarchies

- **Learning Shift Invariant Features**



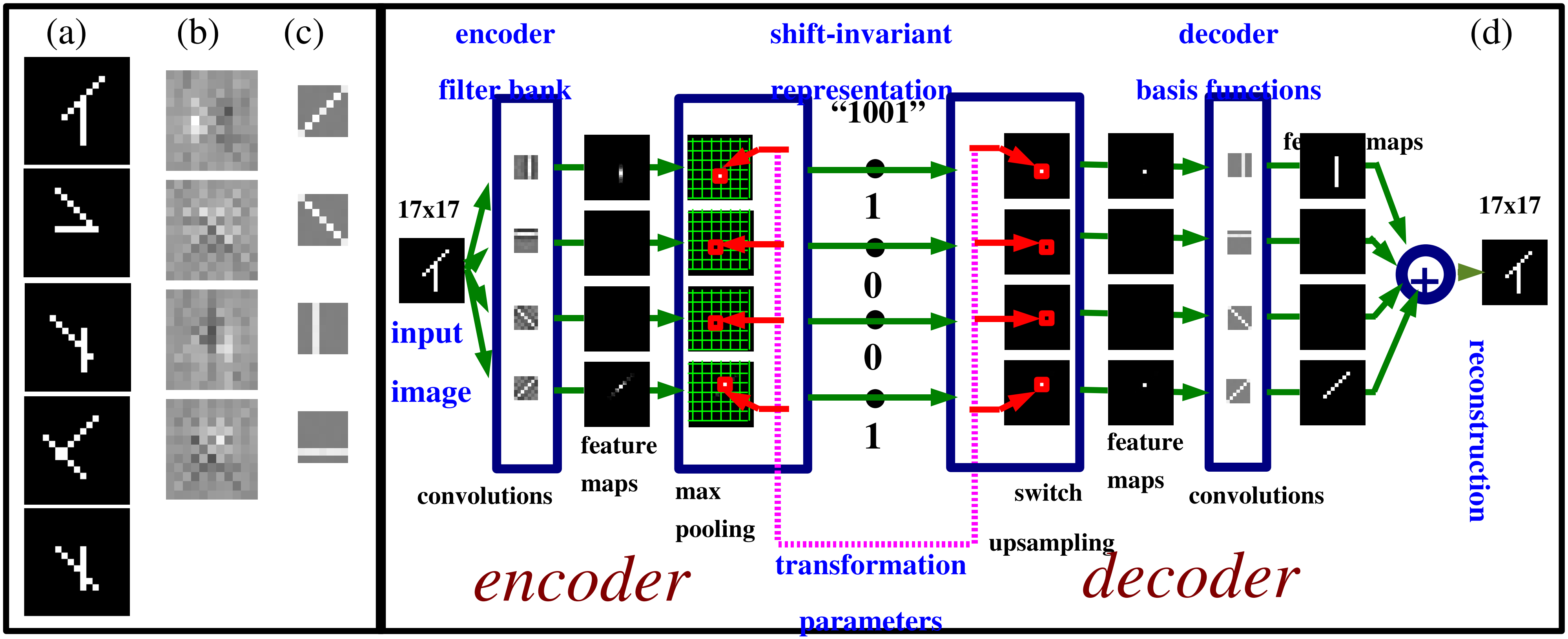
Standard Feature Extractor



Invariant Feature Extractor

Learning Invariant Feature Hierarchies

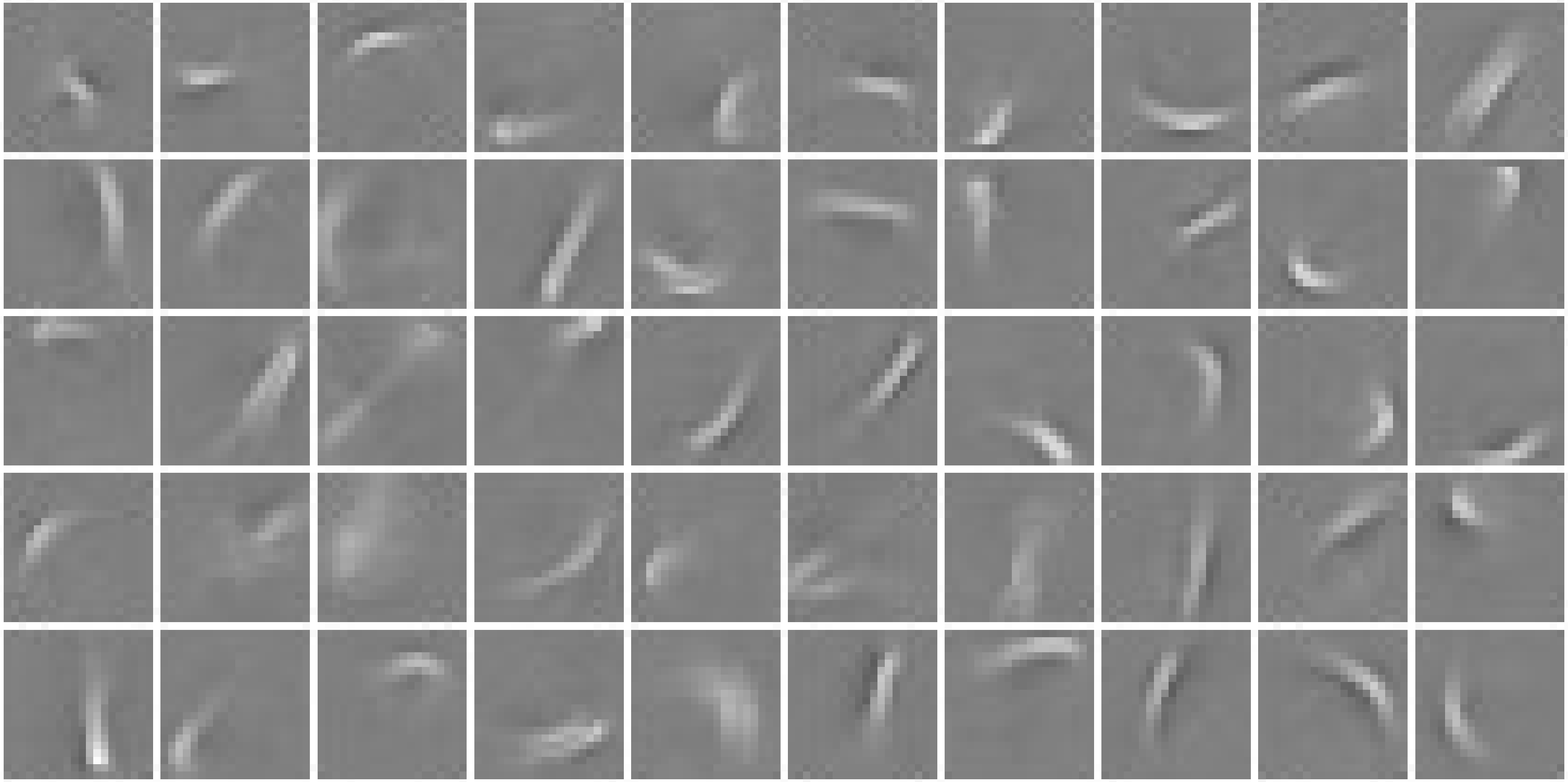
- **Learning Shift Invariant Features**



Shift Invariant Global Features on MNIST

Learning 50 Shift Invariant Global Features on MNIST:

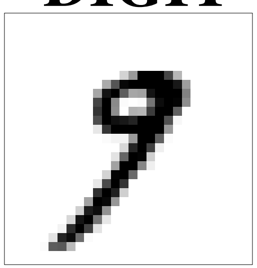
- 50 filters of size 20x20 movable in a 28x28 frame (81 positions)
- movable strokes!



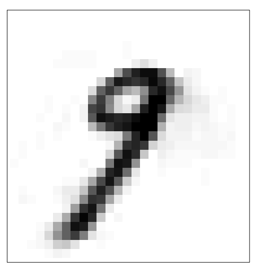
Example of Reconstruction

- **Any character can be reconstructed as a linear combination of a small number of basis functions.**

ORIGINAL DIGIT

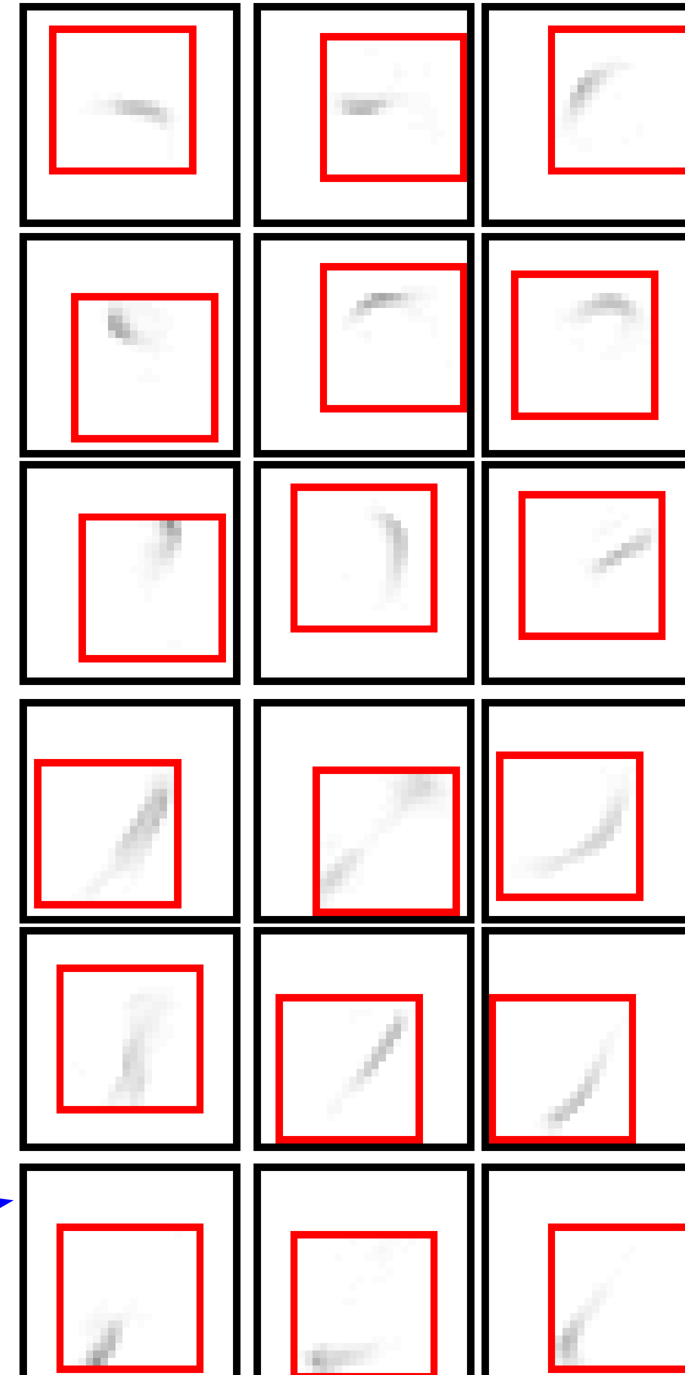
$\sum$  $\cong$

RECONS-TRUCTION

 $=$

ACTIVATED DECODER BASIS FUNCTIONS

(in feed-back layer)



red squares: decoder bases

Learning Invariant Filters in a Convolutional Net

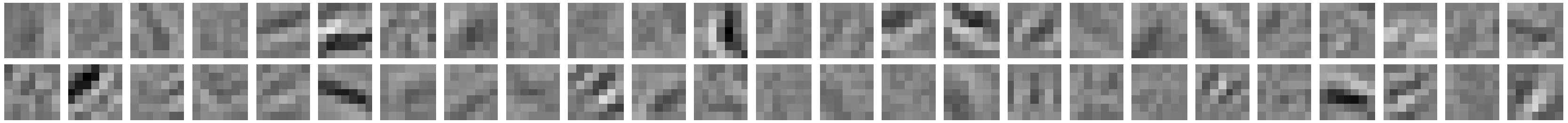


Figure 1: 50 7x7 filters in the first convolutional layer that were learned by the network trained supervised from *random* initial conditions with 600K digits.

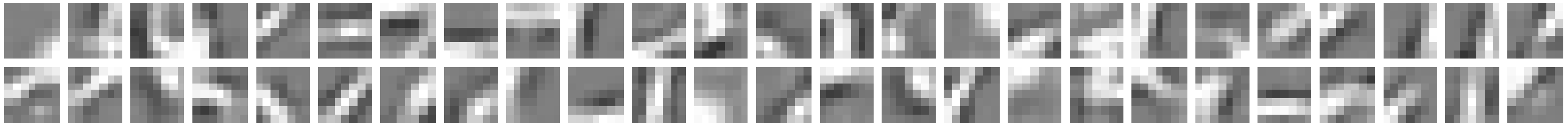


Figure 2: 50 7x7 filters that were learned by the unsupervised method (on 60K digits), and that are used to initialize the first convoltional layer of the network.

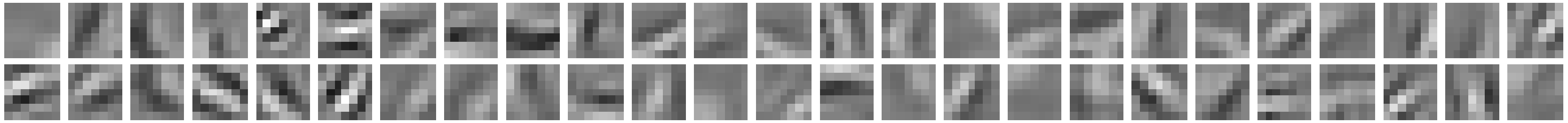
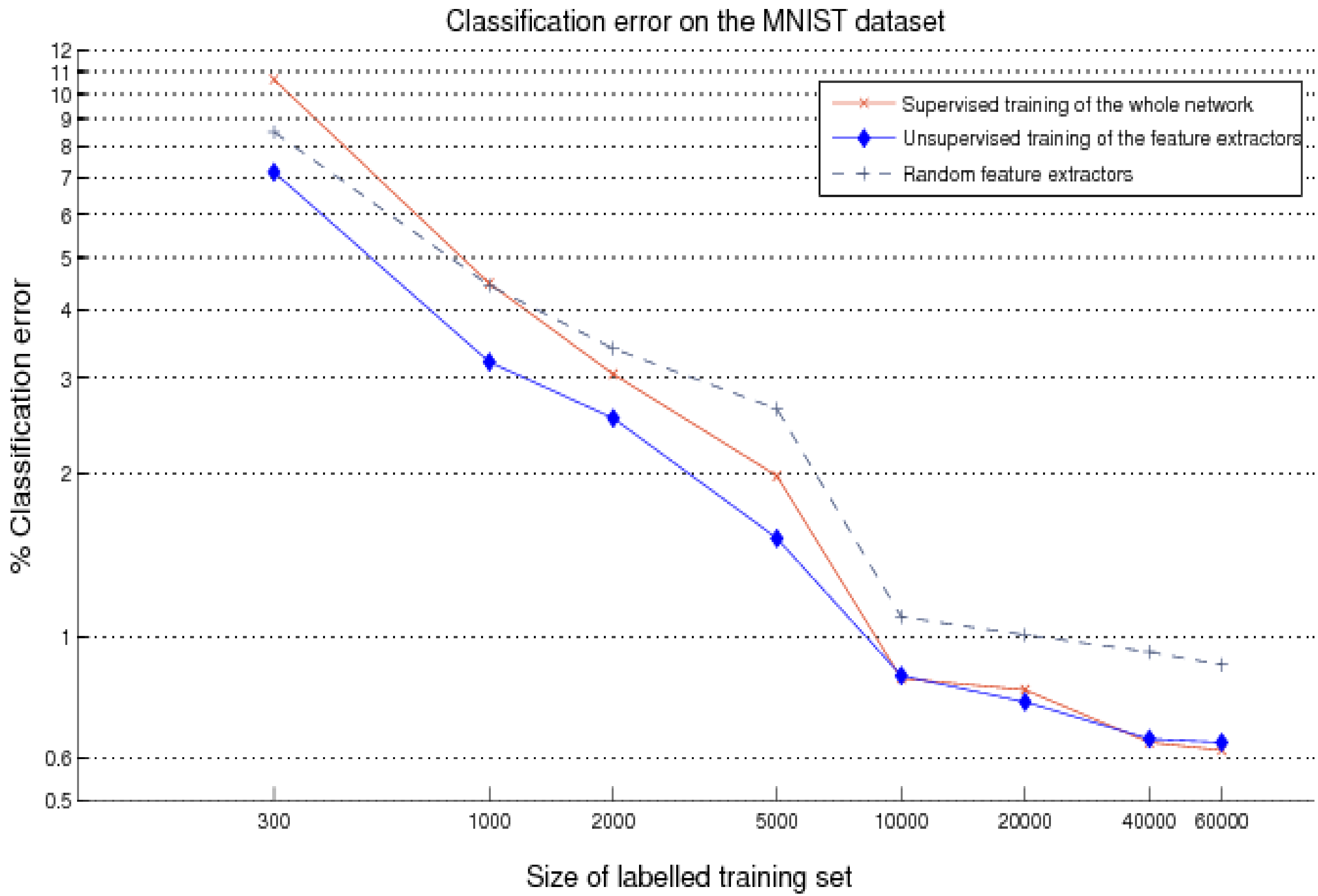


Figure 3: 50 7x7 filters in the first convolutional layer that were learned by the network trained supervised from the initial conditions given by the *unsupervised method* (see fig.2) with 600K digits.

Influence of Number of Training Samples

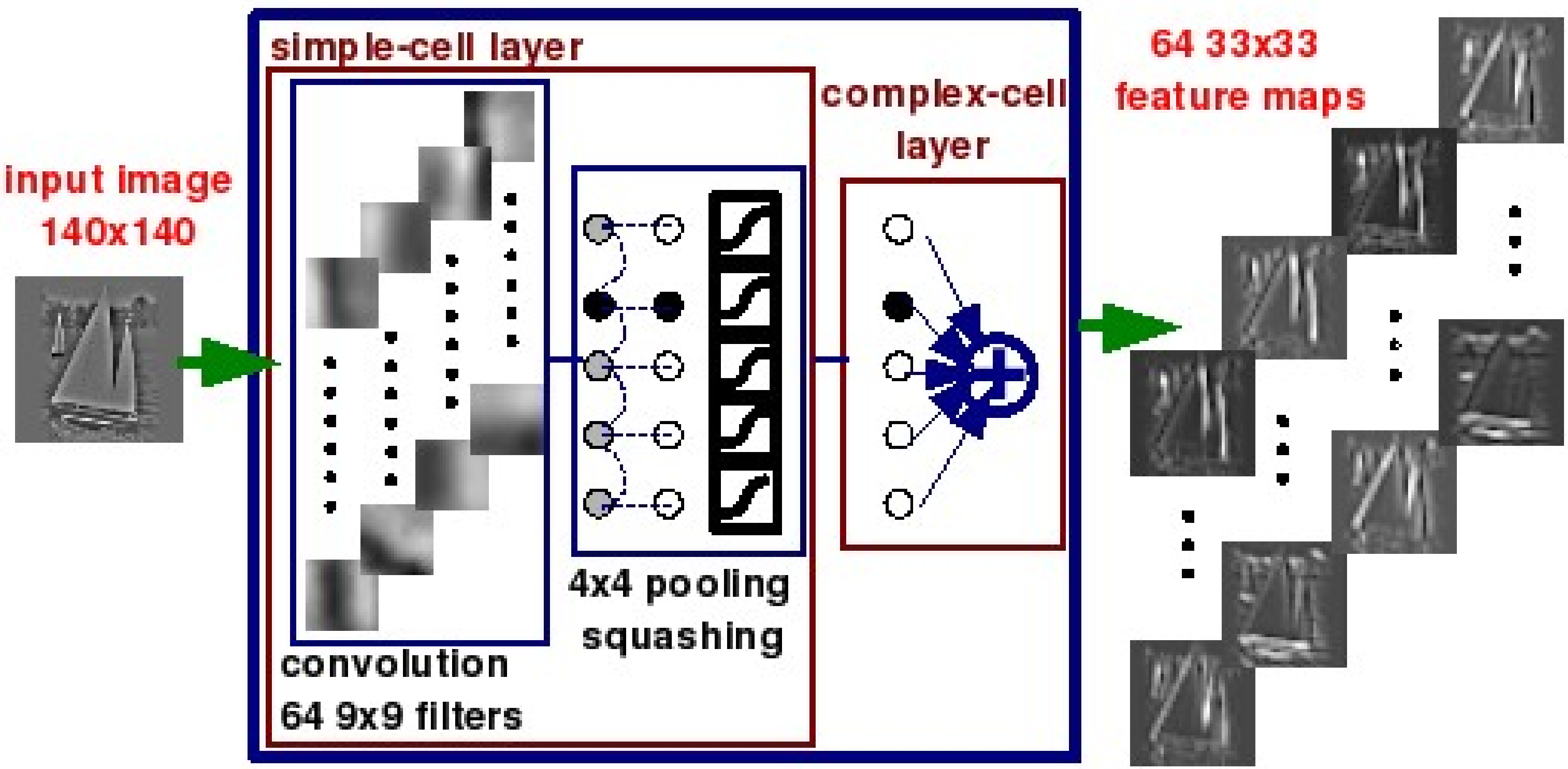


Generic Object Recognition: 101 categories + background

- **Caltech-101 dataset: 101 categories**
  - accordion airplanes anchor ant barrel bass beaver binocular bonsai brain brontosaurus buddha butterfly camera cannon car_side ceiling_fan cellphone chair chandelier cougar_body cougar_face crab crayfish crocodile crocodile_head cup dalmatian dollar_bill dolphin dragonfly electric_guitar elephant emu euphonium ewer Faces Faces_easy ferry flamingo flamingo_head garfield gerenuk gramophone grand_piano hawksbill headphone hedgehog helicopter ibis inline_skate joshua_tree kangaroo ketch lamp laptop Leopards llama lobster lotus mandolin mayfly menorah metronome minaret Motorbikes nautilus octopus okapi pagoda panda pigeon pizza platypus pyramid revolver rhino rooster saxophone schooner scissors scorpion sea_horse snoopy soccer_ball stapler starfish stegosaurus stop_sign strawberry sunflower tick trilobite umbrella watch water_lilly wheelchair wild_cat windsor_chair wrench yin_yang
- **Only 30 training examples per category!**
- **A convolutional net trained with backprop (supervised) gets 20% correct recognition.**
- **Training the filters with the sparse invariant unsupervised method**

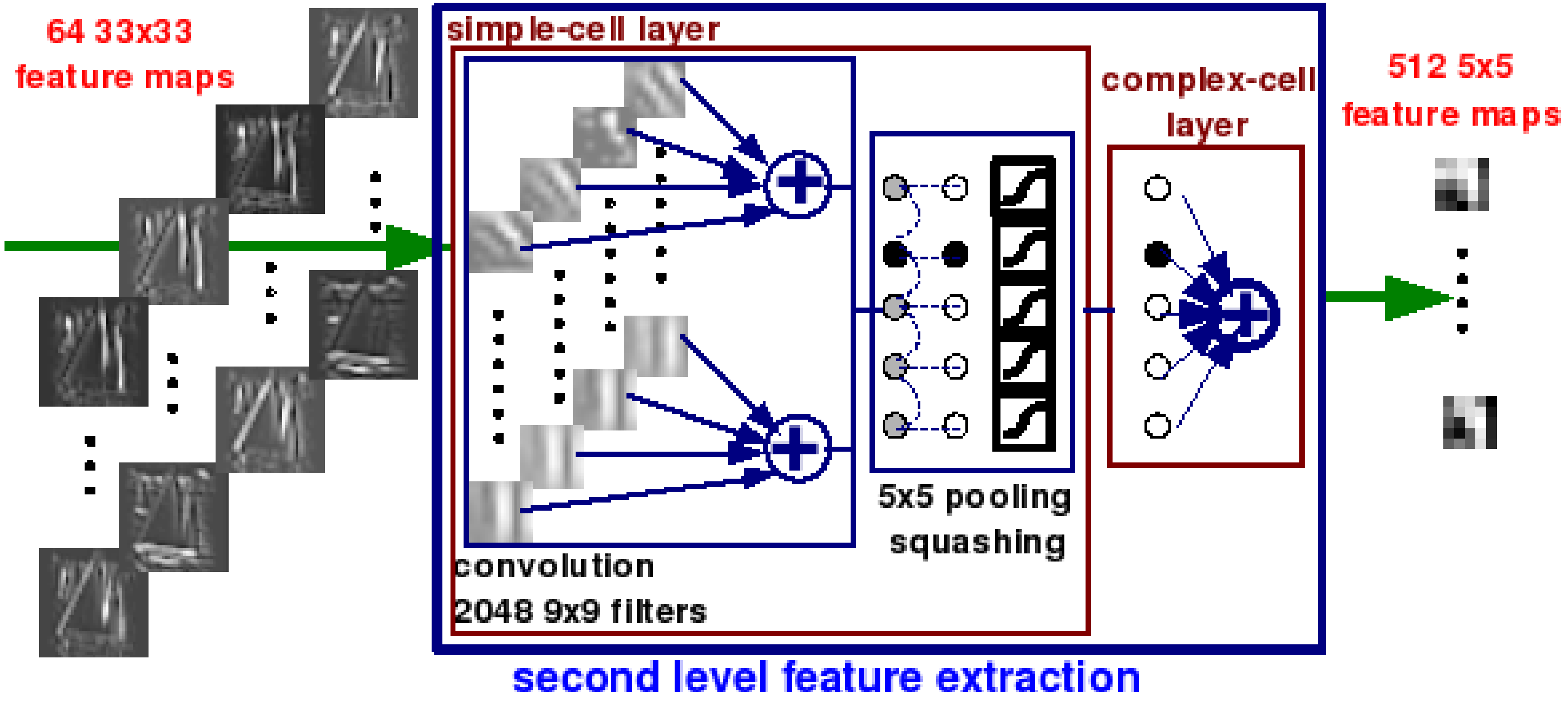
Training the 1st stage filters

- 12x12 input windows (complex cell receptive fields)
- 9x9 filters (simple cell receptive fields)
- 4x4 pooling



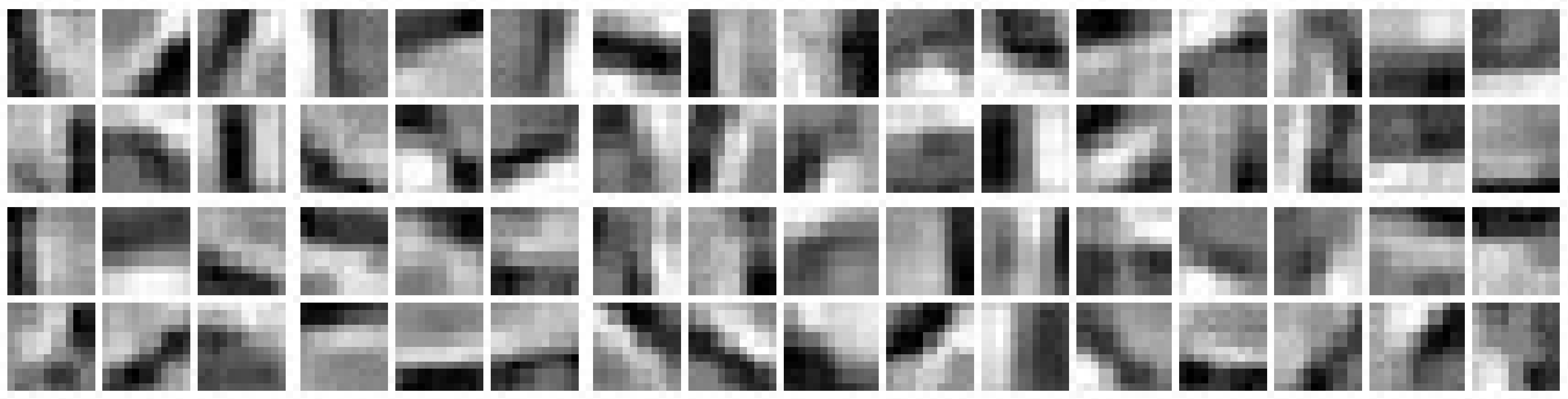
Training the 2nd stage filters

- 13x13 input windows (complex cell receptive fields on 1st features)
- 9x9 filters (simple cell receptive fields)
- Each output feature map combines 4 input feature maps
- 5x5 pooling

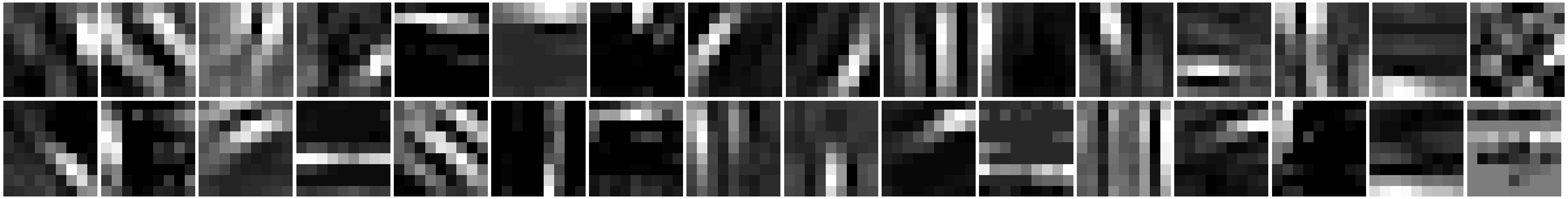


Generic Object Recognition: 101 categories + background

- 9x9 filters at the first level

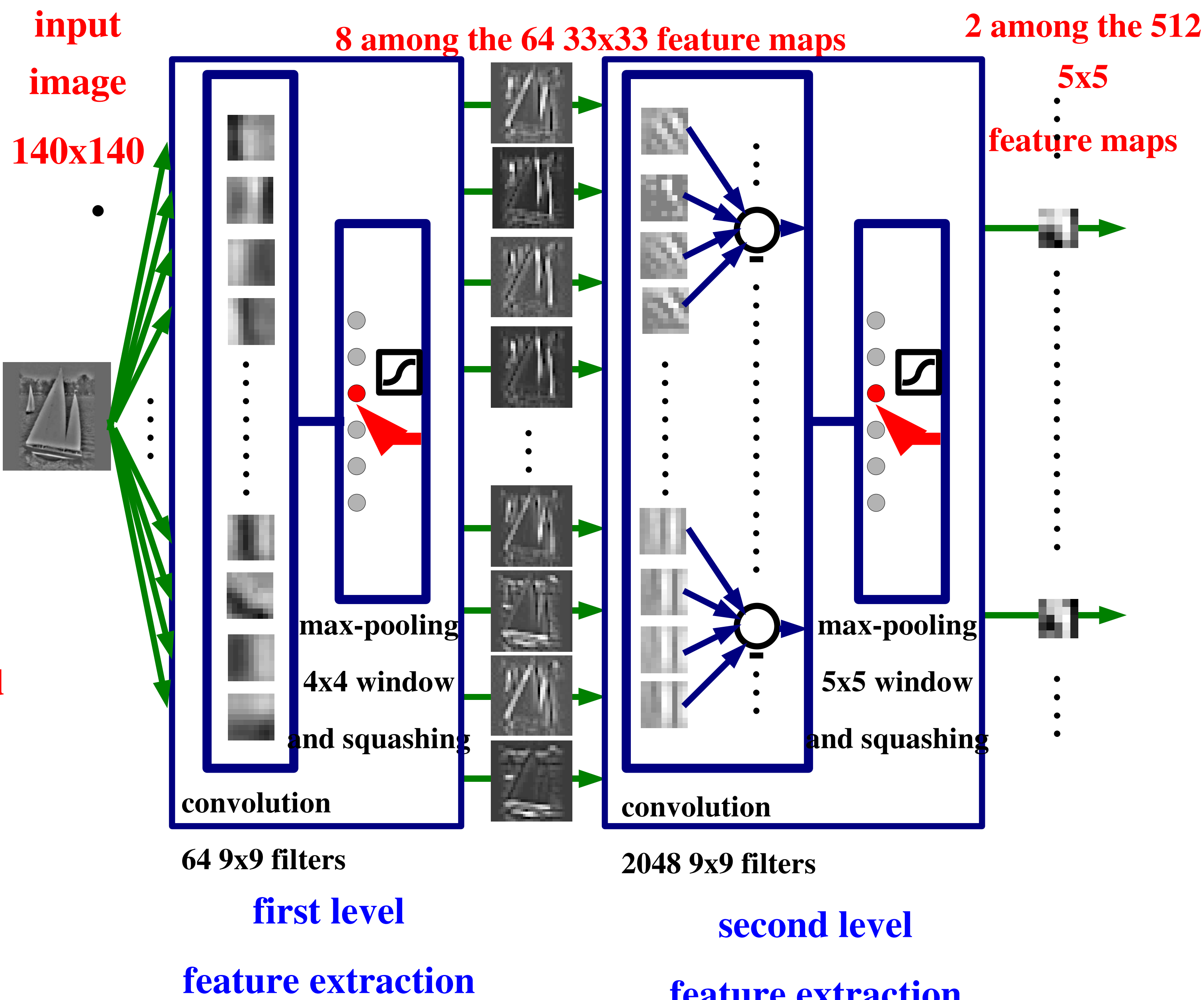


- 9x9 filters at the second level

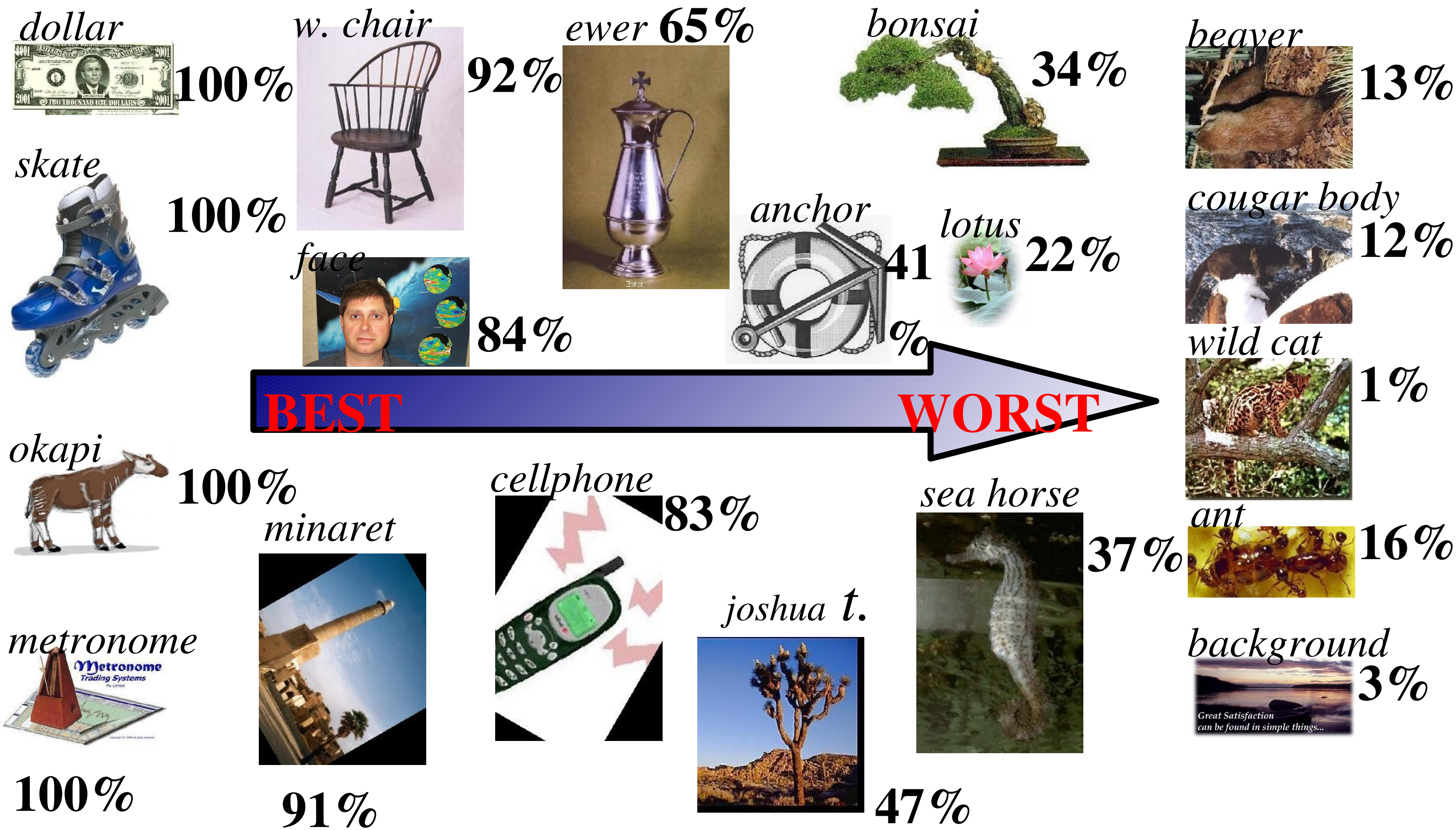


Shift-Invariant Feature Hierarchies on Caltech-101

- 2 layers of filters trained unsupervised
- supervised classifier on top.
- 54% correct on Caltech-101 with 30 examples per class
- 20% correct with purely supervised backprop

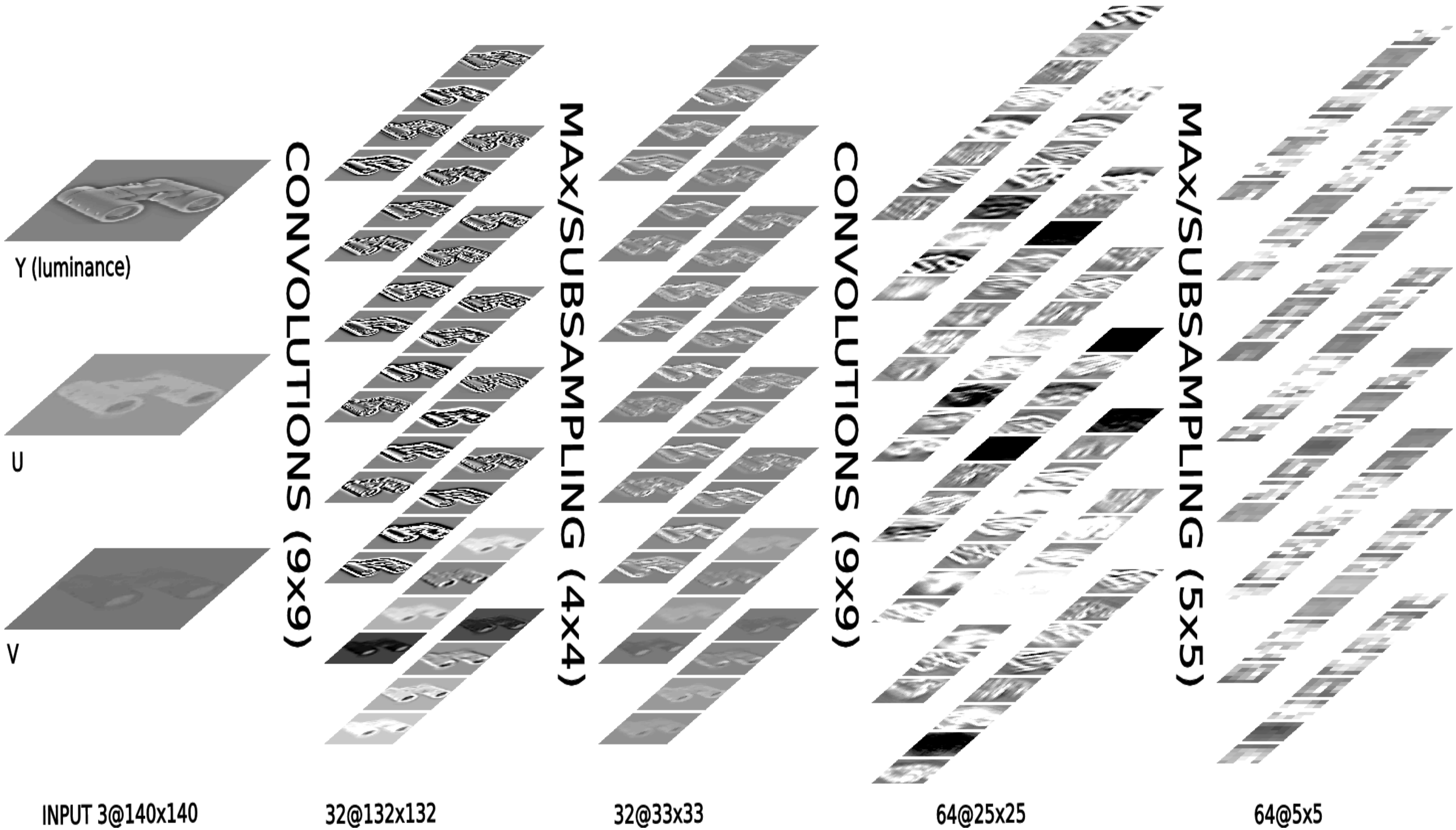


Recognition Rate on Caltech 101



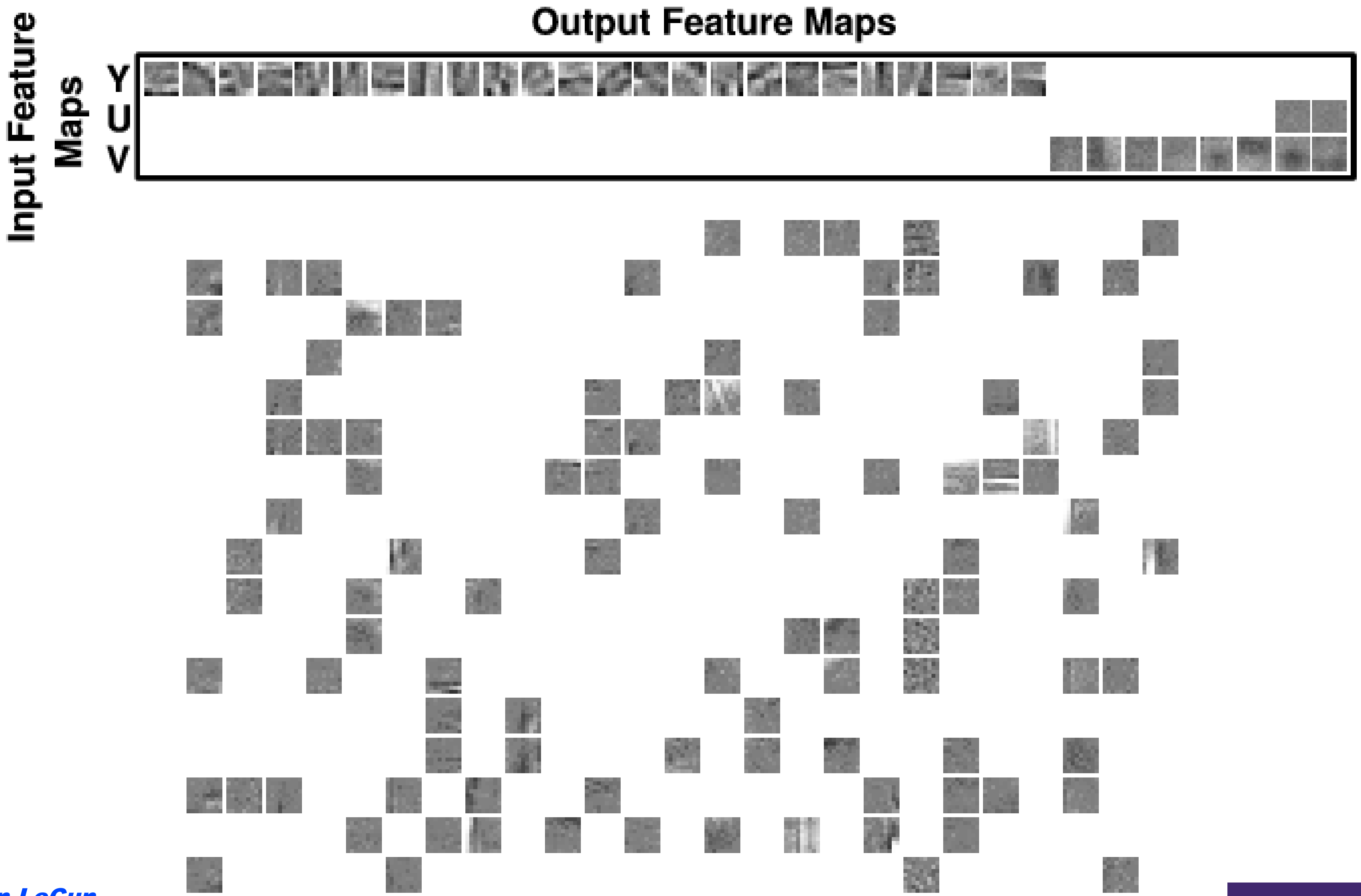
Caltech 256

Network



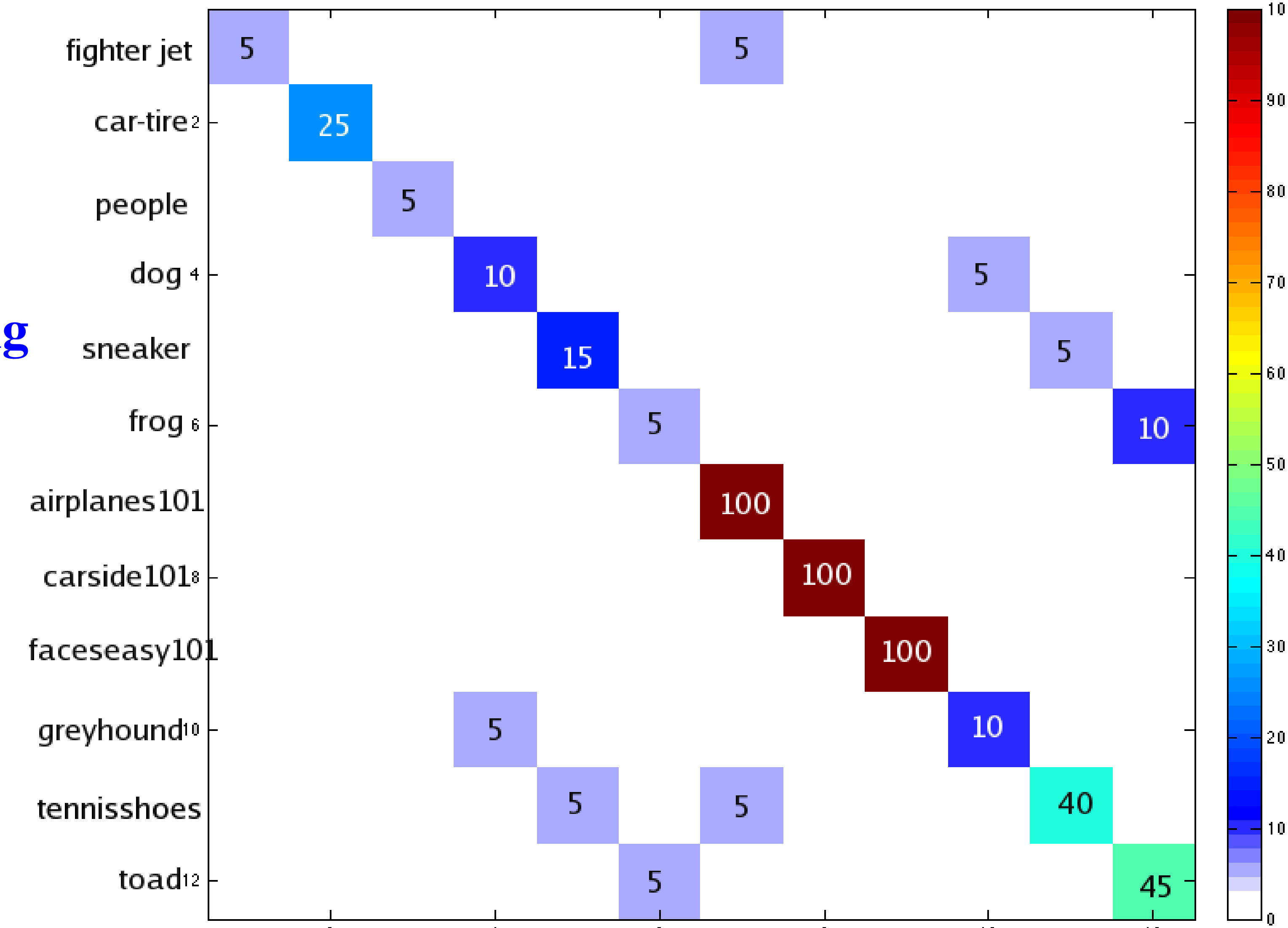
Y (luminance)
U
V
CONVOLUTIONS (9x9)
MAX/SUBSAMPLING (4x4)
CONVOLUTIONS (9x9)
MAX/SUBSAMPLING (5x5)
INPUT 3@140x140
32@132x132
32@33x33
64@25x25
64@5x5

Network: Learned Filters



Caltech 256: Results

31% correct recognition with 60 training samples per category



Practical Conclusion

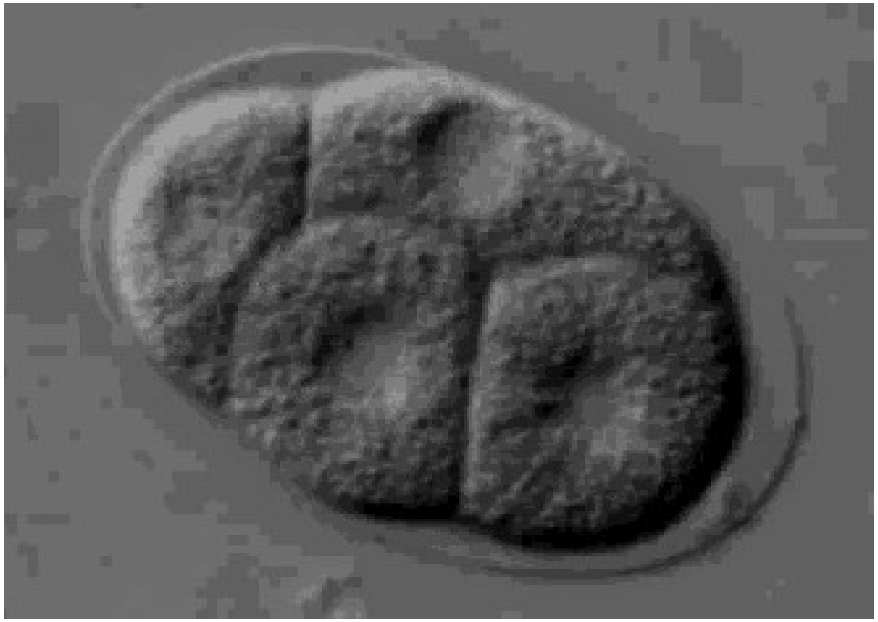
- **Deep architectures are better than shallow ones for vision**
- **The Multi-stage Hubel-Wiesel Architecture can be trained to recognize almost any set of objects.**
  - Supervised gradient descent learning requires too many examples
  - Unsupervised learning of each layer reduces the number of necessary training samples
- **Invariant feature learning preserves the nature of each feature, but throws away the instantiation parameters (position).**
- **Invariant feature hierarchies can be trained unsupervised**
  - on large training sets: the recognition rate is almost as good as supervised gradient descent learning
  - on small training sets: the recognition rate is much better.
- **We haven't solved the deep learning problem yet!**

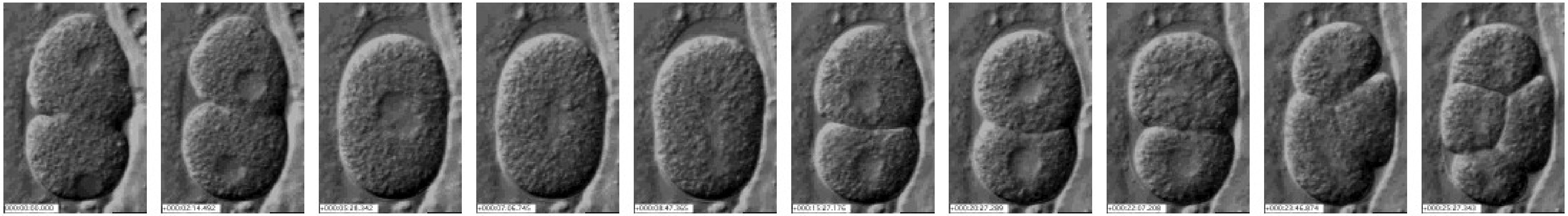
C. Elegans Embryo Phenotyping

[Ning, Delhome, LeCun, Piano, Bottou, Barbano

IEEE Trans. Image Processing, October 2005]

- **Analyzing results for Gene Knock-Out Experiments**
- **Automatically determining if a roundworm embryo is developing normally after a gene has been knocked out.**

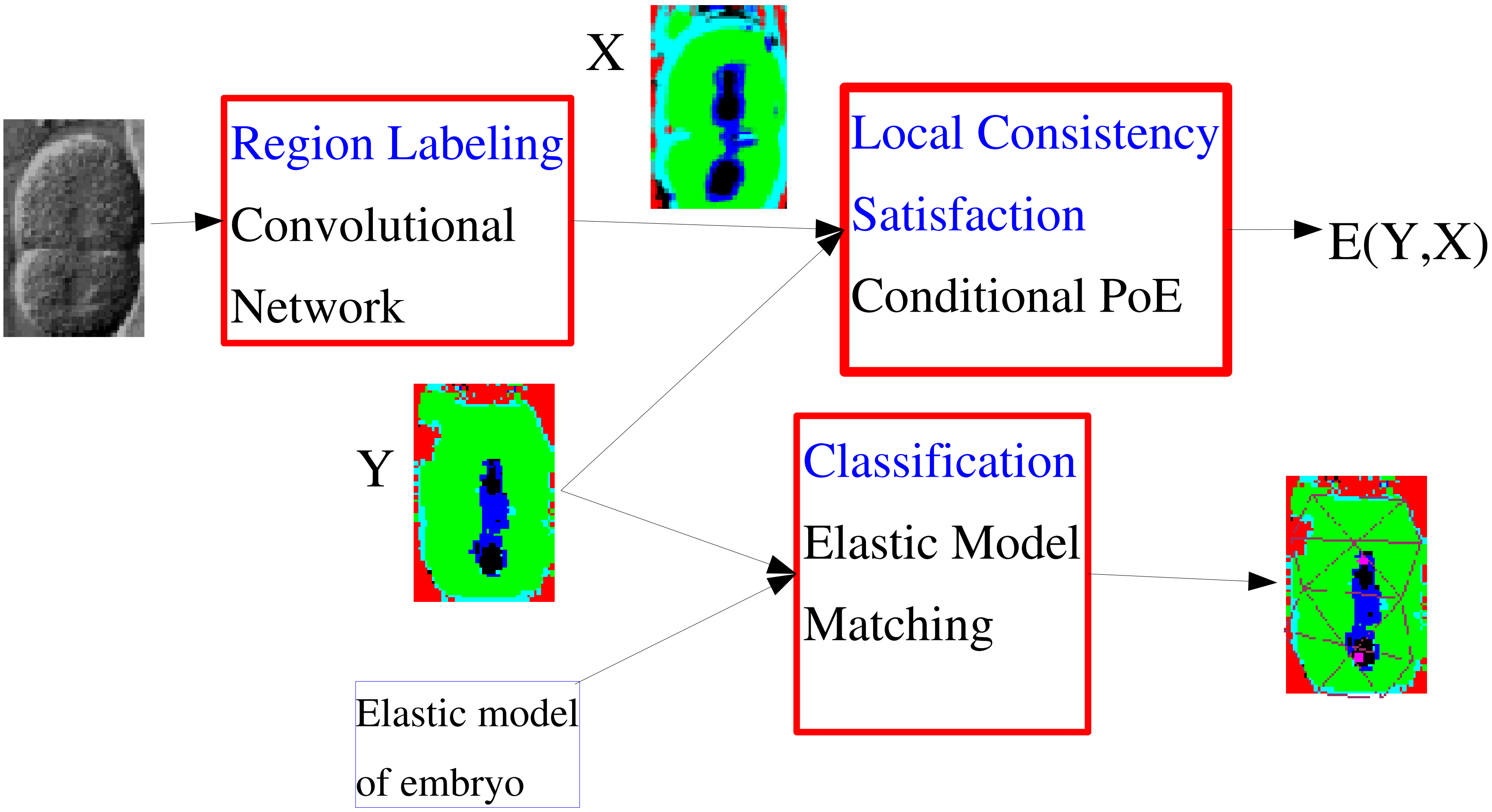




Time-lapse movie

Architecture

- **Region Classification with a convolutional network**
- **Local Consistency with a Conditional Product of Experts**
- **Embryo classification with elastic model matching**



Region Labeling with a Convolutional Net

- Supervised training fromhand-labeled images
- 5 categories:
  - nucleus, nuclear membrane, cytoplasm, cell wall, external medium

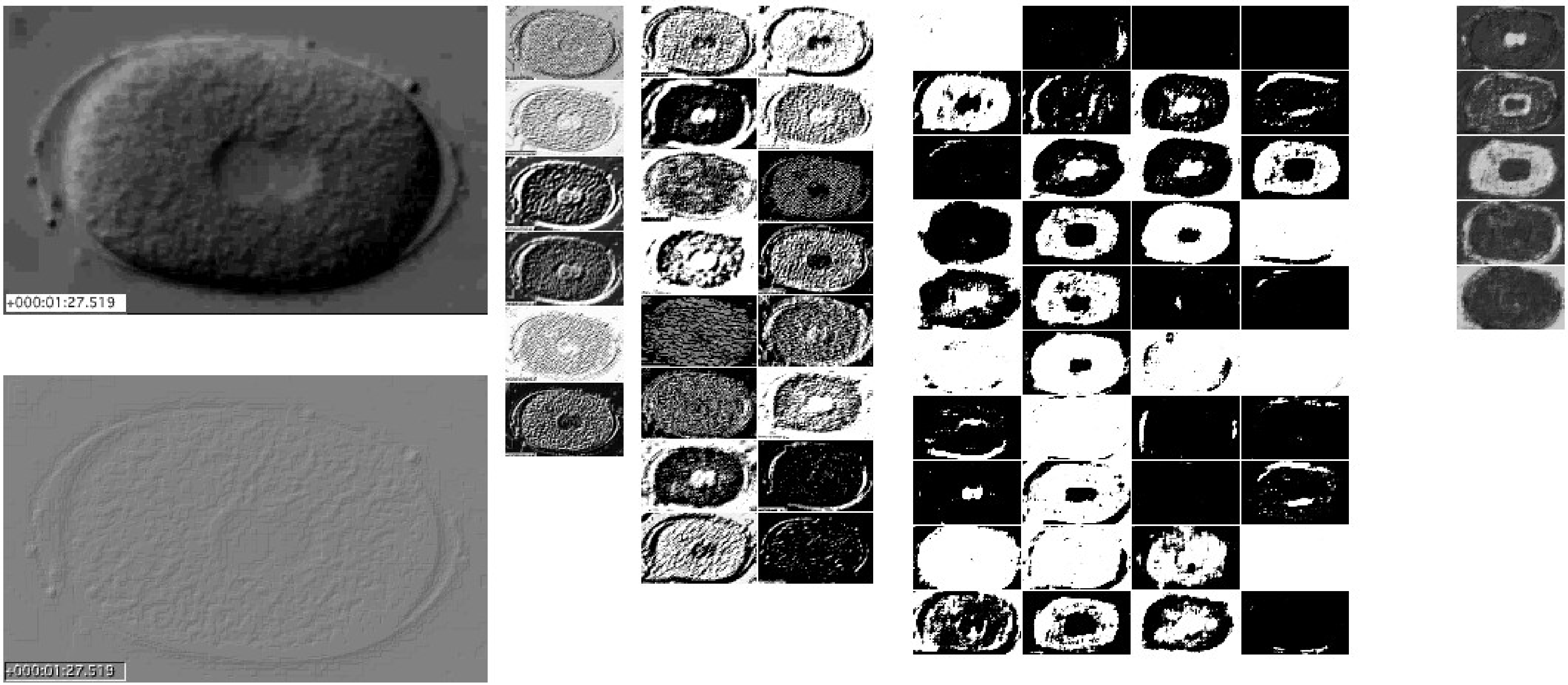
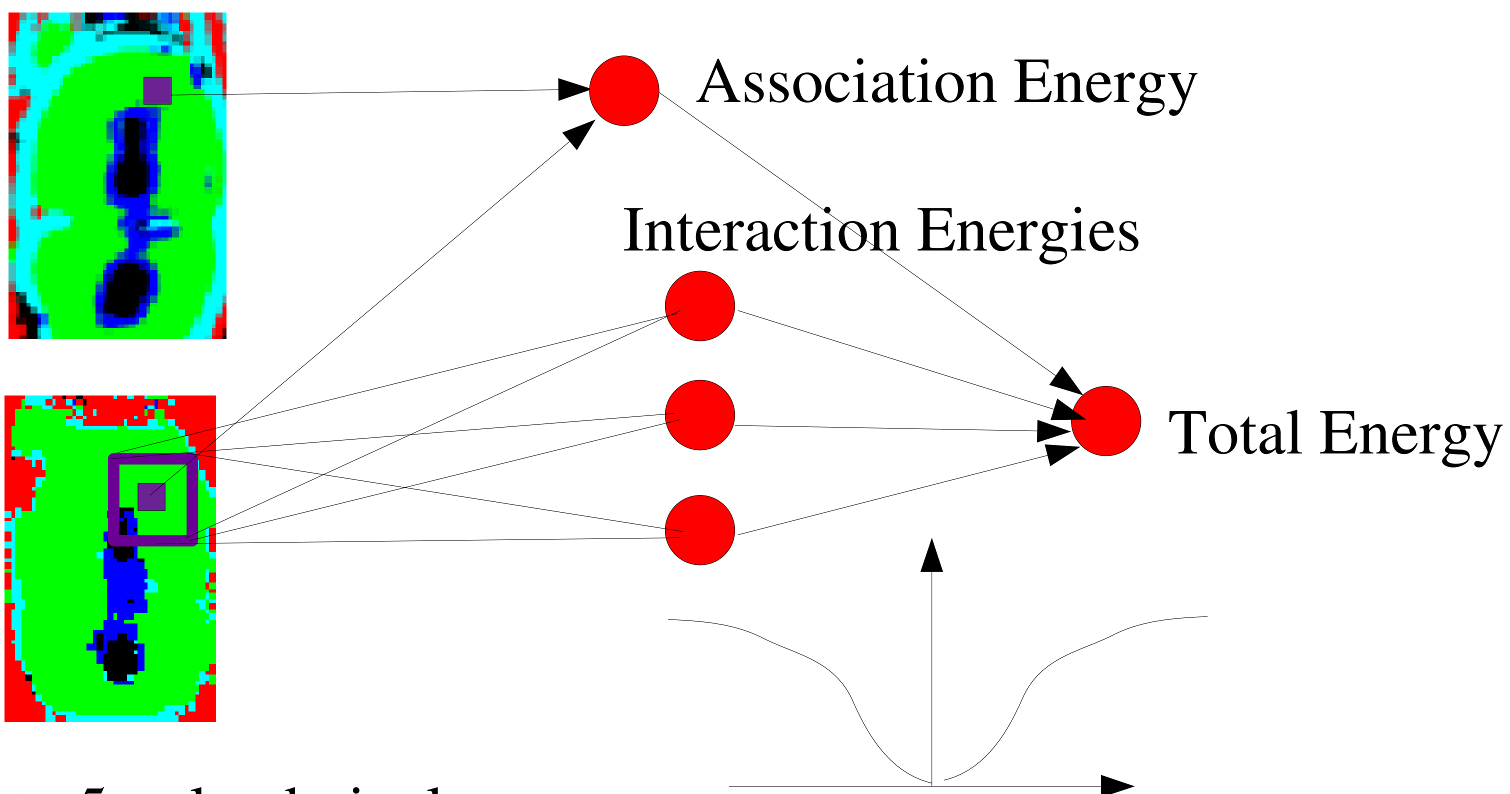


Image Segmentation with Local Consistency Constraints

[Teh, Welling, Osindero, Hinton, 2001], [Kumar, Hebert 2003], [Zemel 2004]

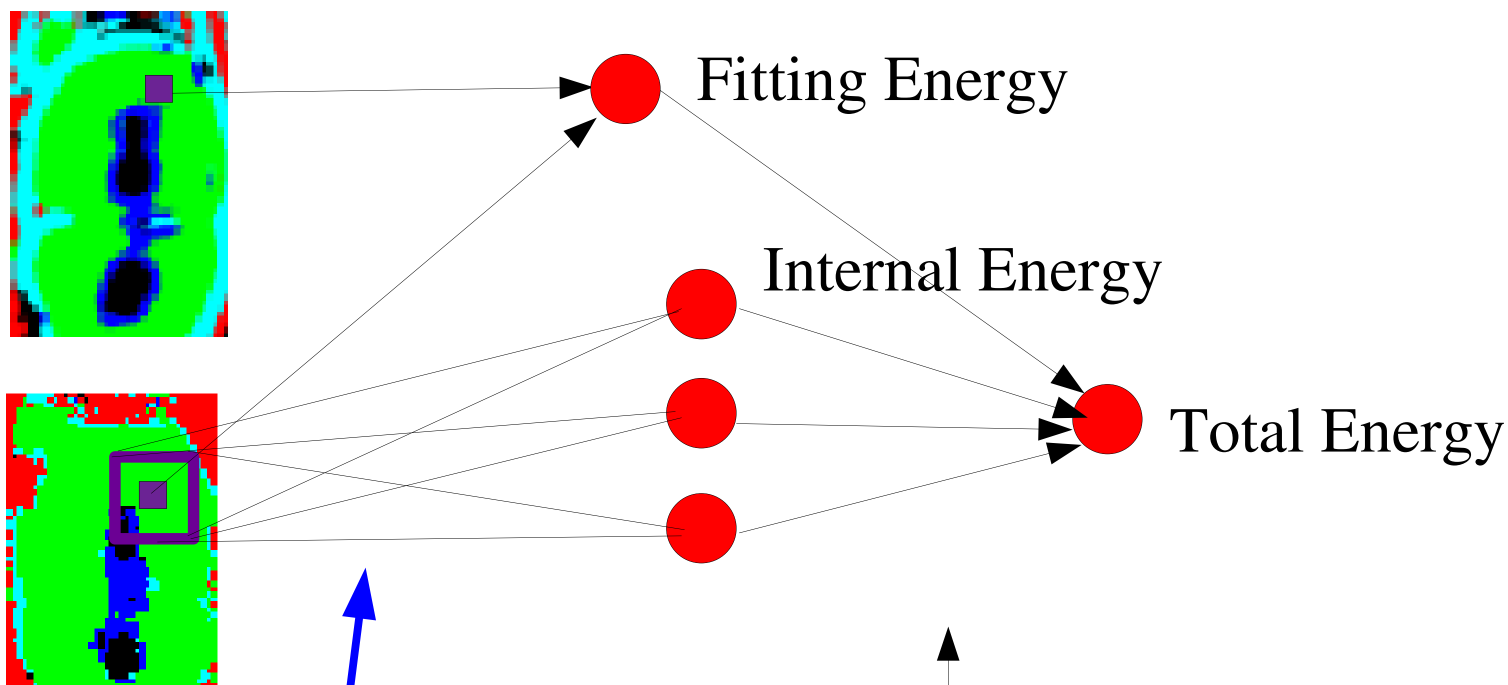
- **Learn local consistency constraints with an Energy-Based Model so as to clean up images produced by the segmentor.**



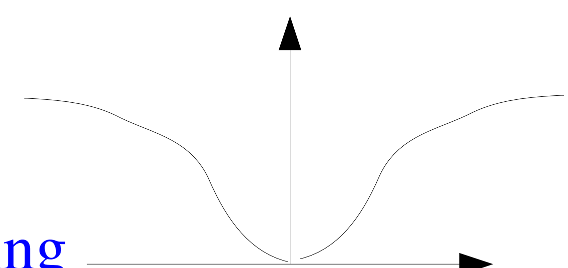
Discrete 5-valued pixels

Convolutional Conditional PoE

$$E(Y,X,W) = \sum_{ij} C_{Y_{ij},X_{ij}} + \sum_{k=1}^{10} \sum_{ij} g\left(\sum_{lpq=(1,-2,-2)}^{(5,+2,+2)} W_{klpq}.Y_{l,i-p,j-q} \right)$$



Inference with Gibbs sampling



$$g(u) = \frac{u^2}{1+u^2}$$

C. Elegans Embryo Phenotyping

- **Analyzing results for Gene Knock-Out Experiments**

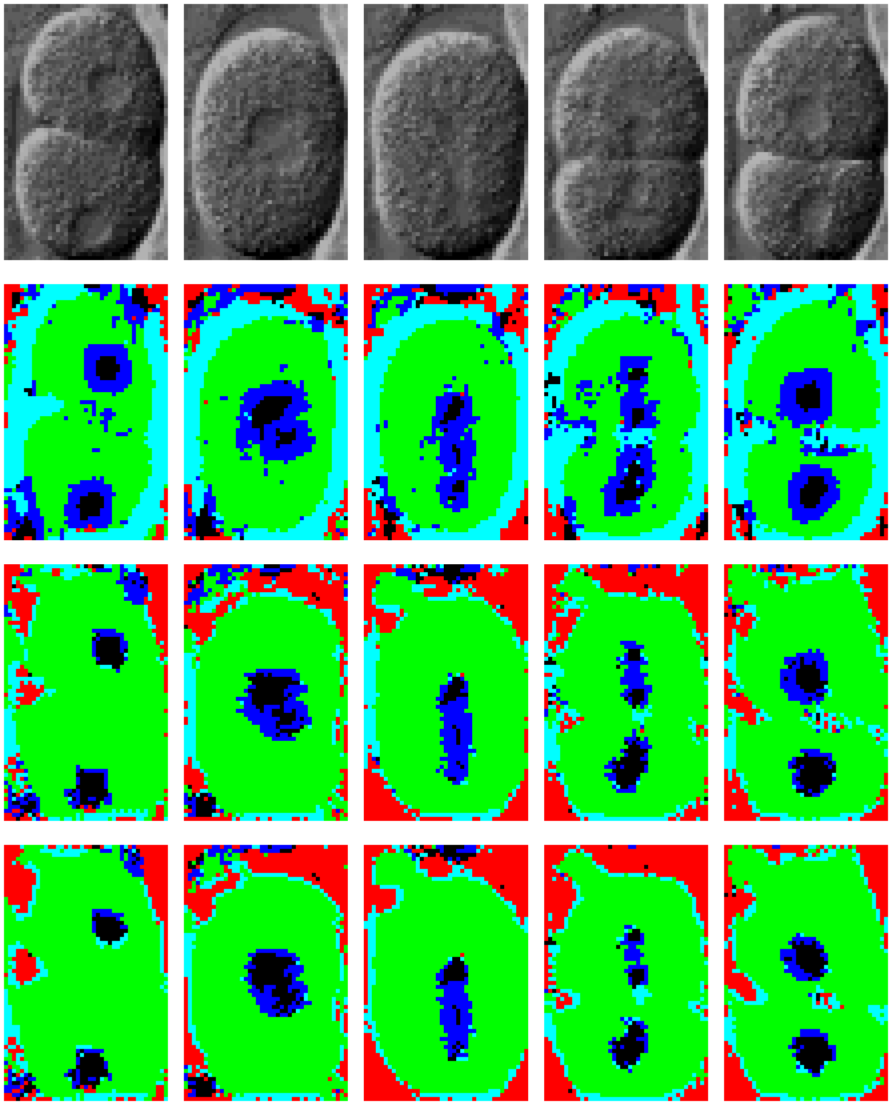
Original Images

Segmentation #1

Segmentation #2

CCPoE

Cleanup



(1) (2) (3) (4) (5)

C. Elegans Embryo Phenotyping

Analyzing results for Gene Knock-Out Experiments