Energy-Based Models in Document Recognition and Computer Vision

Yann LeCun

The Courant Institute of Mathematical Sciences

New York University

Collaborators:

Marc'Aurelio Ranzato, Sumit Chopra, Fu-Jie Huang, Y-Lan Boureau

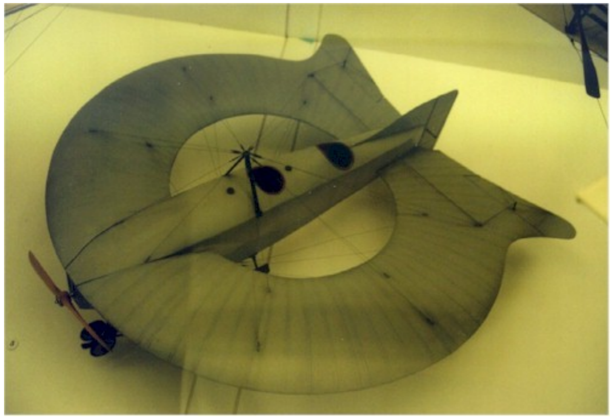
See: [LeCun et al. 2006]: “A Tutorial on Energy-Based Learning”

[Ranzato et al. AI-Stats'07], [Ranzato et al. NIPS06], [Ranzato et al. ICDAR '07]

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

The Challenges of Pattern Recognition, Computer Vision, and Visual Neuroscience

- **How do we learn "invariant representations"?**
  - From the image of an airplane, how do we extract a representation that is invariant to pose, illumination, background, clutter, object instance....
  - How can a human (or a machine) learn those representations by just looking at the world?
- **How can we learn visual categories from just a few examples?**
  - I don't need to see many airplanes before I can recognize every airplane (even really weird ones)



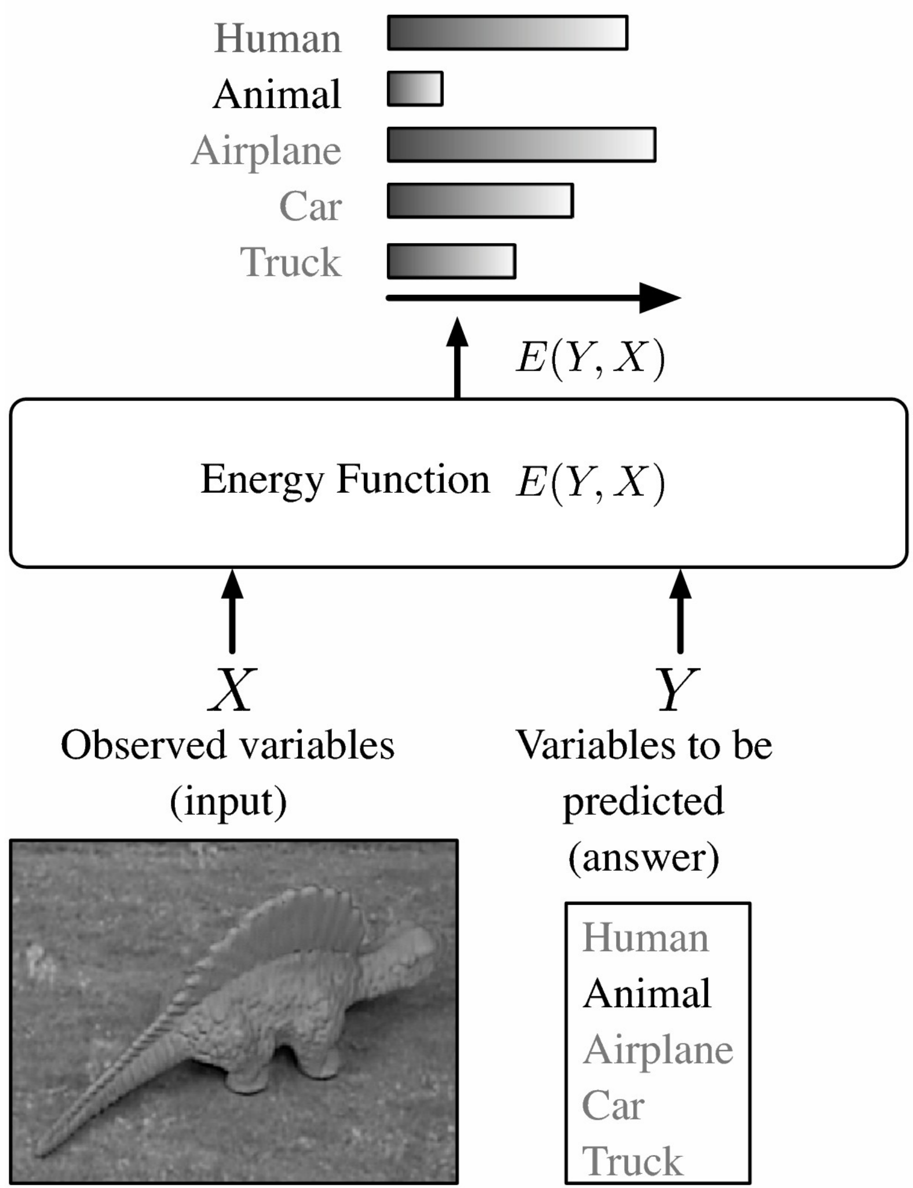
Two Big Problems in Learning and Recognition

- **The "Normalization Problem" (aka 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 visual recognition (and perception in general).
  - How do we train deep architectures with lots of non-linear stages?
- **This talks has three parts:**
  - Energy-Based learning: circumventing the intractable partition function problem.
  - Supervised methods for deep visual learning: convolutional nets
  - Unsupervised methods to learn deep, invariant feature hierarchies: "Deep belief networks".

Part 1: Energy-Based Learning.
circumventing the intractable partition function problem

- **Highly popular methods in the Machine Learning and Natural Language Processing Communities have their roots in Handwriting Recognition**
  - Conditional Random Fields, and related learning models with "structured outputs" are descendants of discriminative learning methods for word-level handwriting recognition.
- **A Tutorial and Energy-Based Learning:**
  - [LeCun & al., 2006]
- **Discriminative Training for "Structured Output" models**
  - The whole literature on discriminative speech recognition [1987-]
  - The whole literature on neural-net/HMM hybrids for speech [Bottou 1991, Bengio 1993, Haffner 1993, Bourlard 1994]
  - Graph Transformer Networks [LeCun & al. Proc IEEE 1998]
  - Conditional Random Fields [Lafferty & al 2001]
  - Max Margin Markov Nets [Altun & al 2003, Taskar & al 2003]

Energy-Based Model for Decision-Making

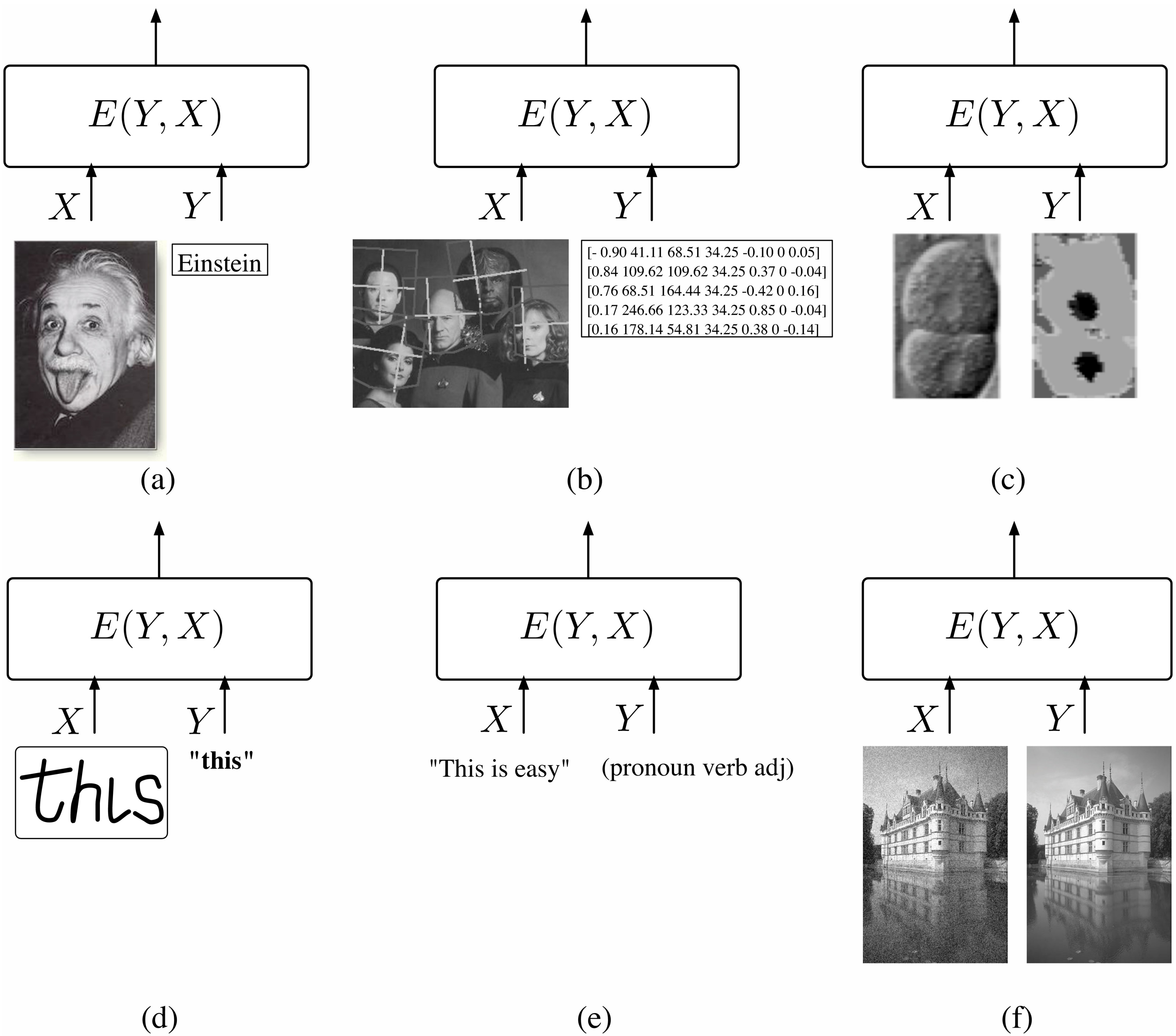


- **Model:** **Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).**

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

- **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, belief propagation, gradient decent.....**

Converting Energies to Probabilities

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.

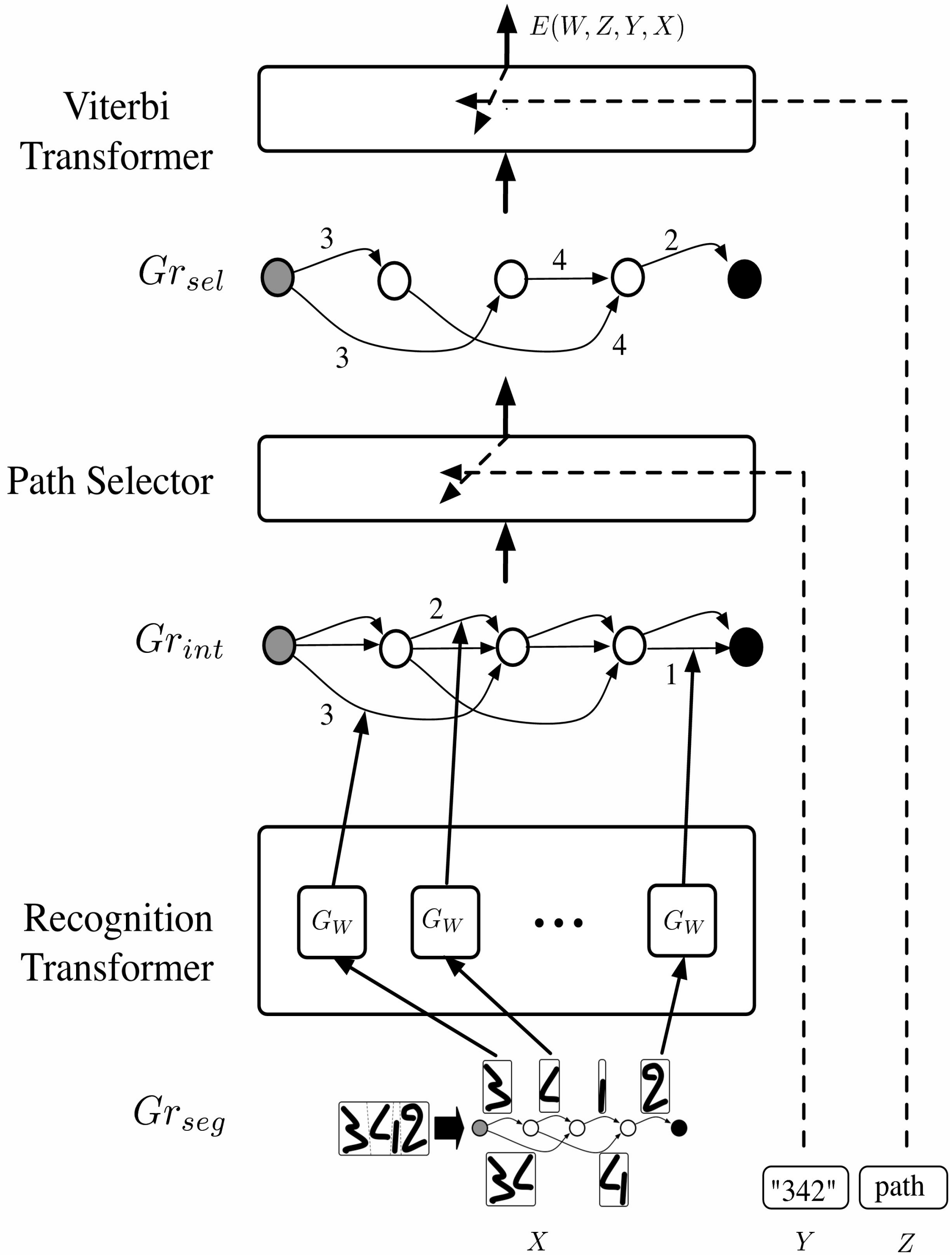
$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}},$$

Partition function

Inverse temperature

Handwriting recognition
Sequence labeling

- **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]

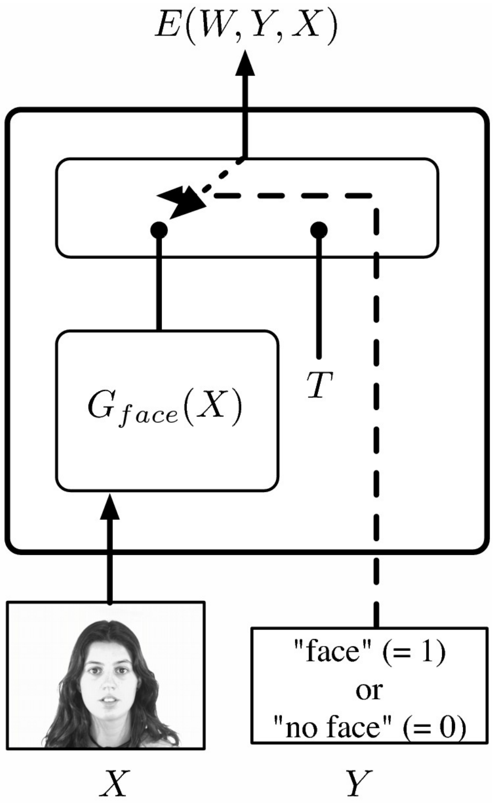


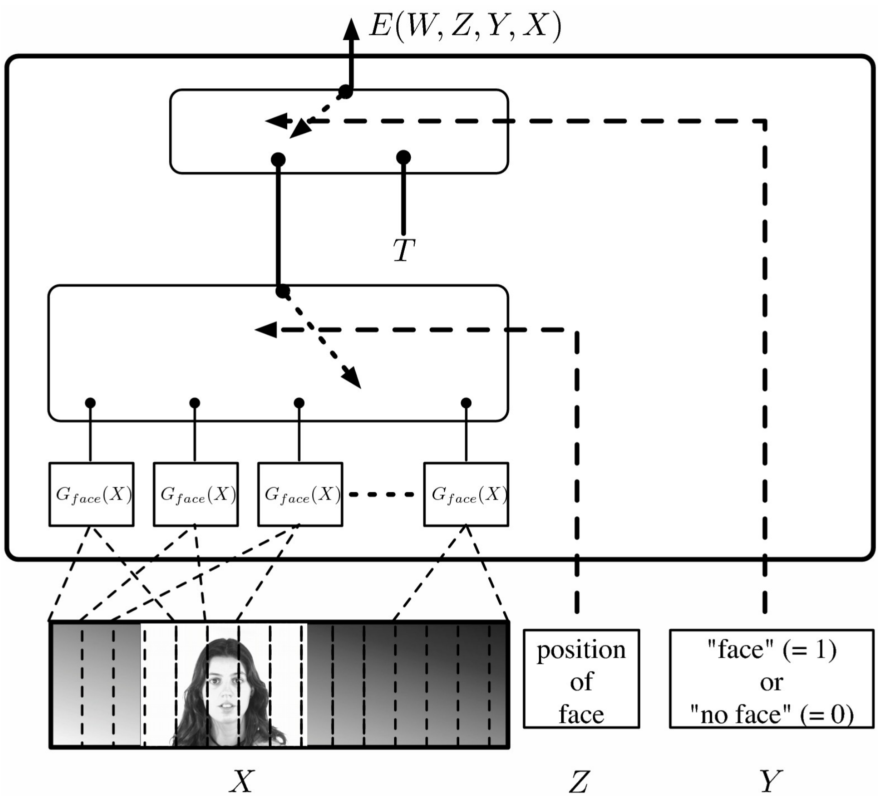
Latent Variable Models

- **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:**

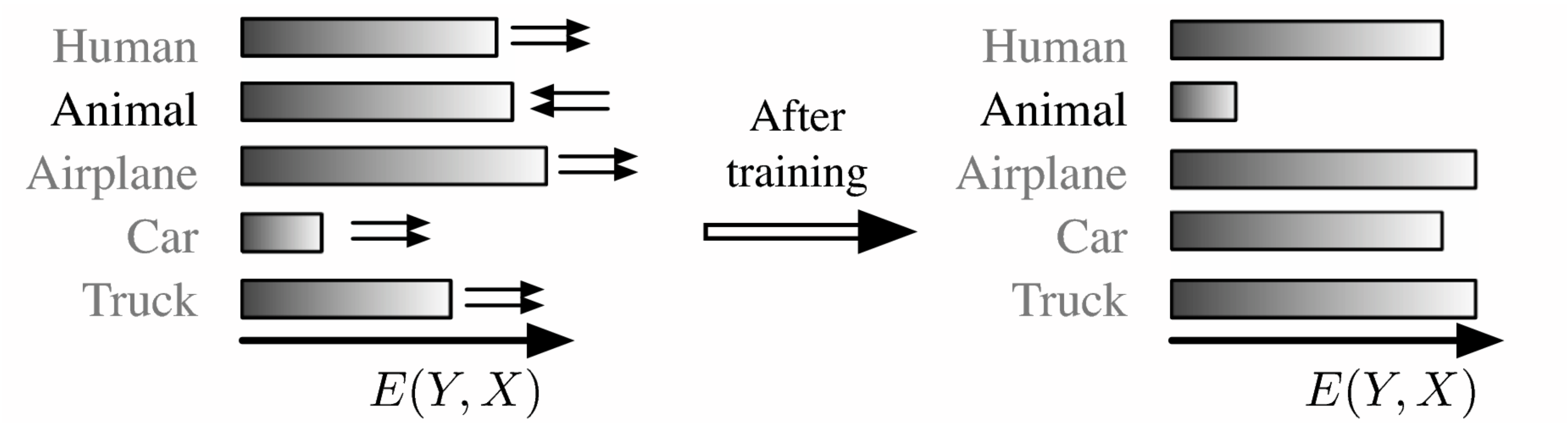
$$Y^* = \text{argmin}_{Y\in\mathcal{Y}} - \frac{1}{\beta} \log \int_{z\in\mathcal{Z}} e^{-\beta E(z,Y,X)}.$$

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

Training an EBM

- **Training an EBM consists in shaping the energy function so that the energies of the correct answer is lower than the energies of all other answers.**
  - Training sample: X = image of an animal, Y = "animal"

$$E(\text{animal}, X) < E(y, X) \ \forall y \neq \text{animal}$$



Architecture and Loss Function

- **Family of energy functions** $\mathcal{E} = \{E(W, Y, X) : \quad W \in \mathcal{W}\}.$
- **Training set** $\mathcal{S} = \{(X^i, Y^i) : i = 1 \dots P\}$
- **Loss functional / Loss function** $\mathcal{L}(E, \mathcal{S}) \qquad \mathcal{L}(W, \mathcal{S})$
  - 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

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

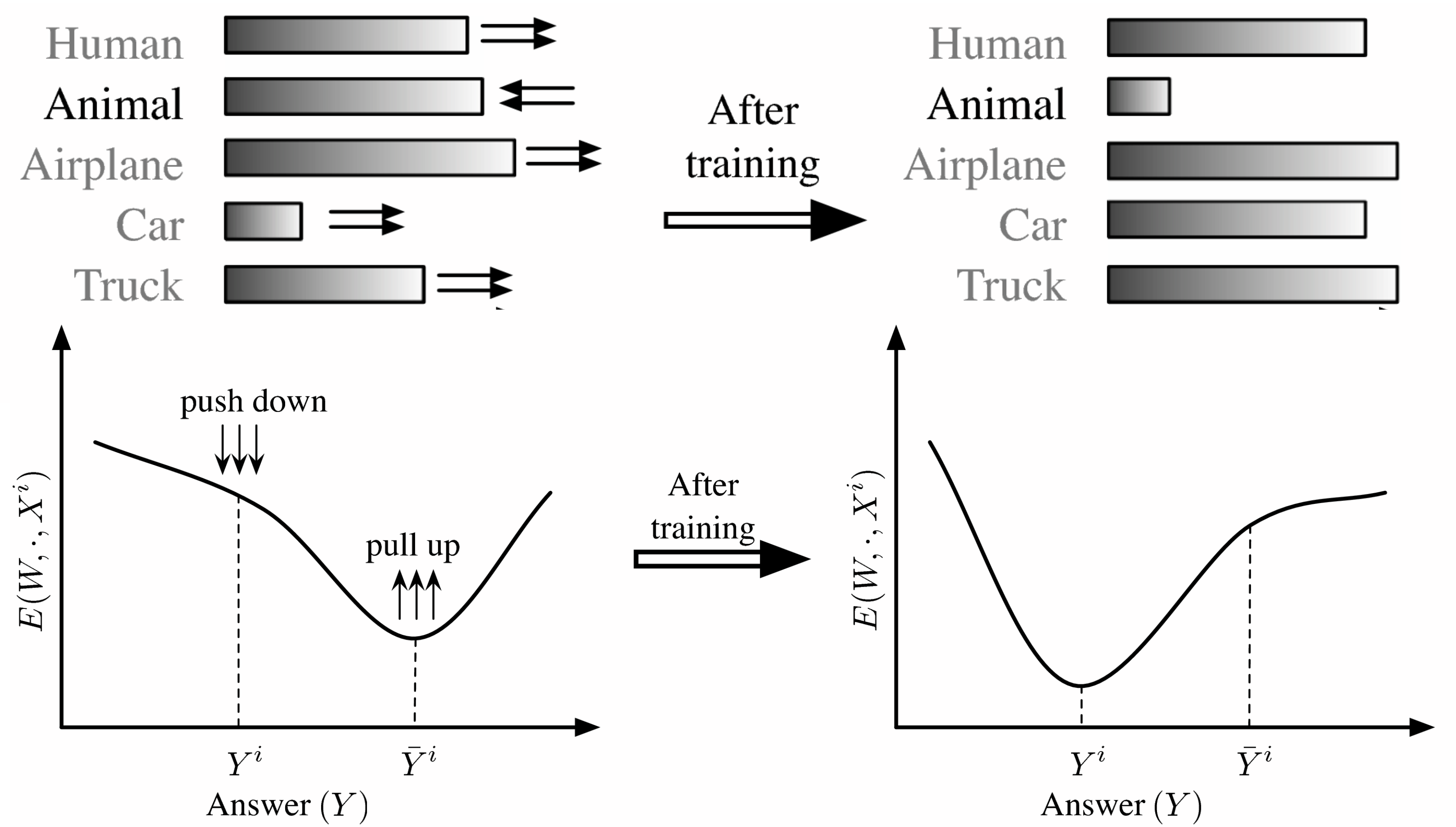
Per-sample loss

Desired answer

Energy surface for a given Xi as Y varies

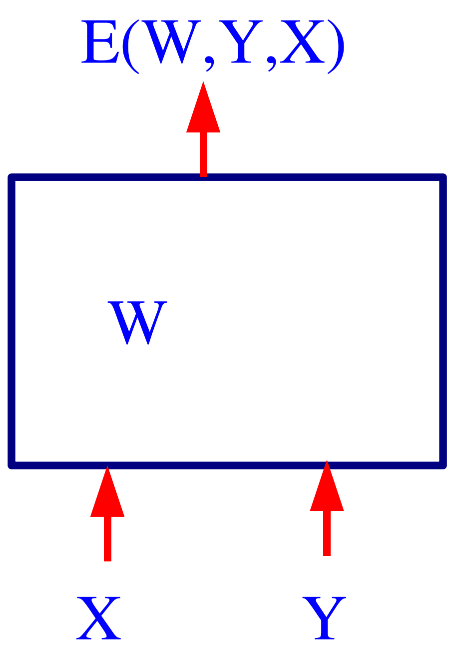
Regularizer

Designing a Loss Functional



- **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

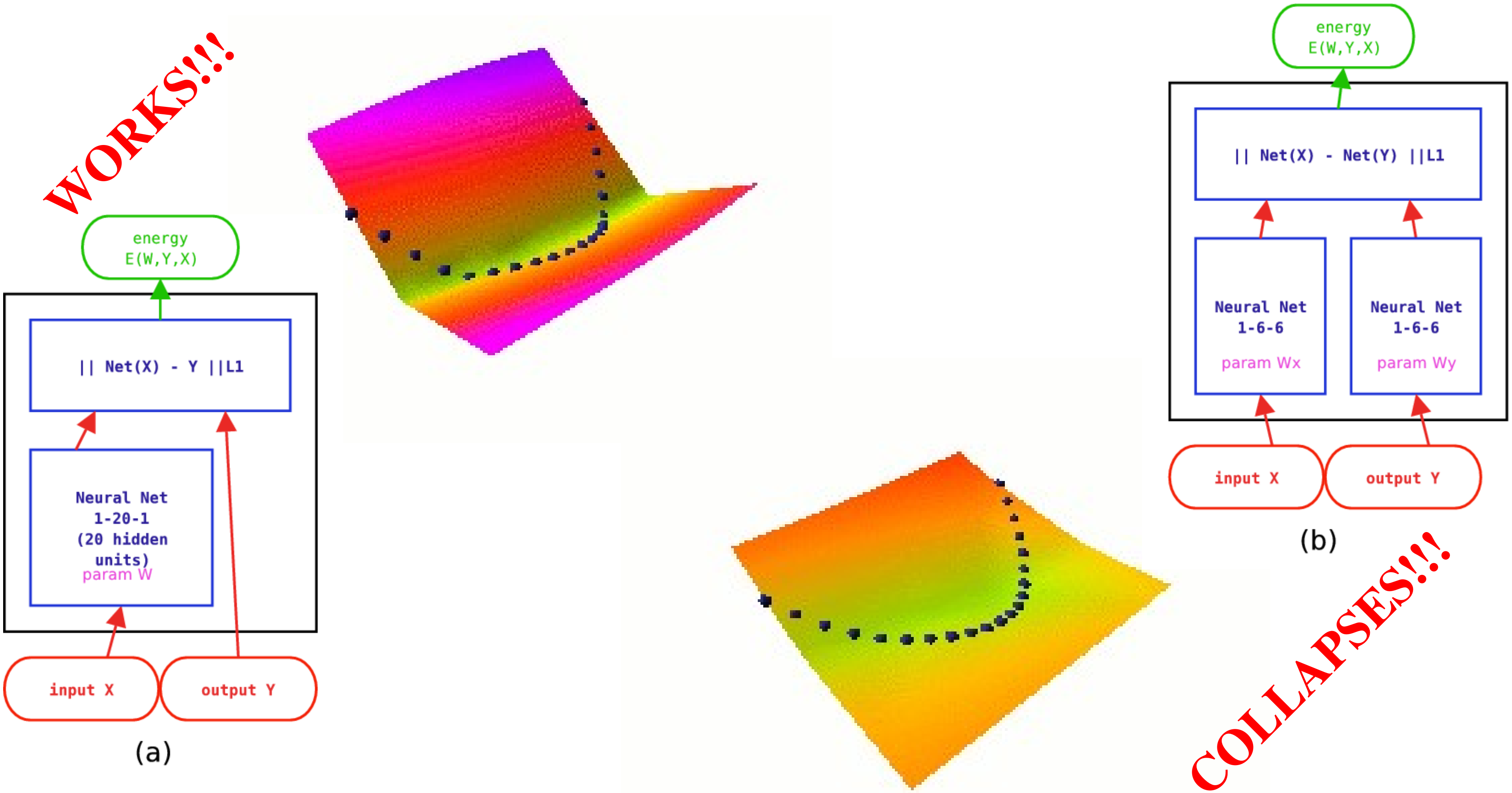


- **1. Design an architecture:** a particular form for E(W,Y,X).
- **2. Pick an inference algorithm for Y:** MAP or conditional distribution, belief prop, min cut, variational methods, gradient descent, MCMC, HMC.....
- **3. Pick a loss function:** in such a way that minimizing it with respect to W over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

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



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:**

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

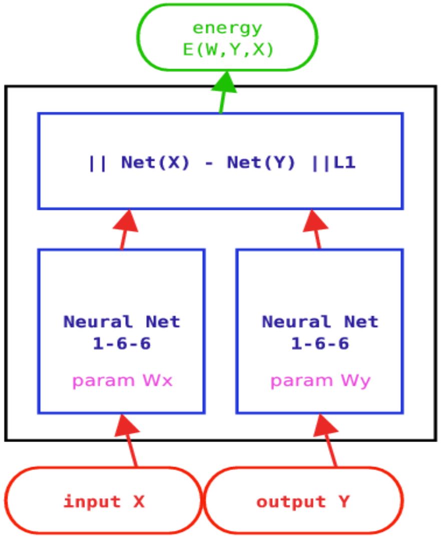
- **Reduces to the perceptron loss when Beta->infinity**

Negative Log-Likelihood Loss

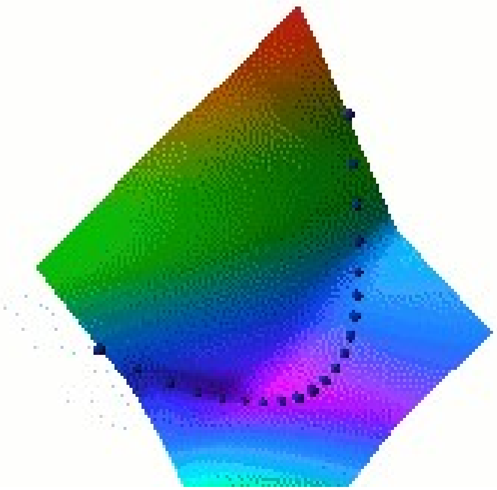
- Pushes down on the energy of the correct answer
- Pulls up on the energies of all answers in proportion to their probability

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

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



(b)



Negative Log-Likelihood Loss

- **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.

A Simpler 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.
  - This is often called **"discriminative Viterbi training"** in the speech and handwriting literature

A Better Loss Function: 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:*

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

- **Most Offending Incorrect Answer: continuous case**

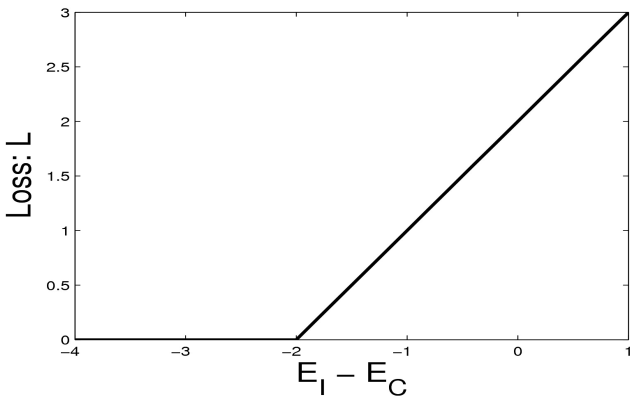
Definition 2 *Let Y be a continuous variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are at least ϵ away from the correct answer:*

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

Examples of Generalized Margin Losses

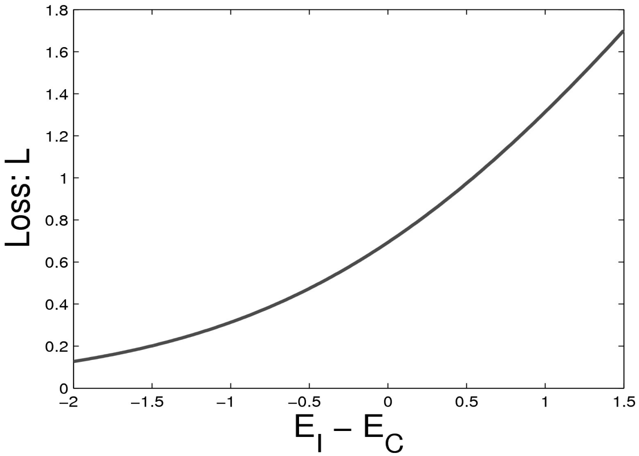
$$L_{\text{hinge}}(W, Y^i, X^i) = \max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right),$$

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



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

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

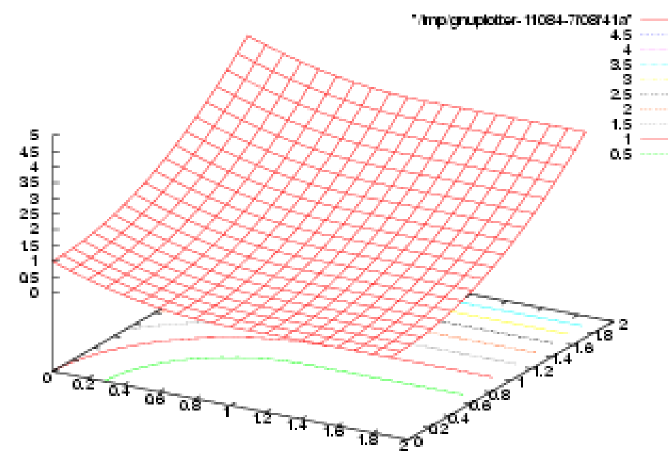


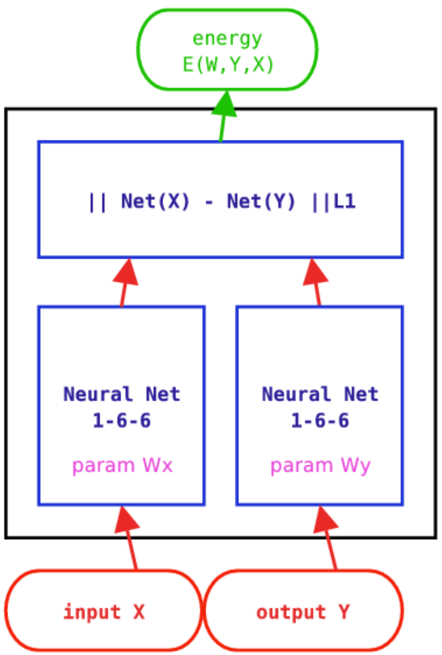
Examples of Margin Losses: Square-Square Loss

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

Square-Square Loss

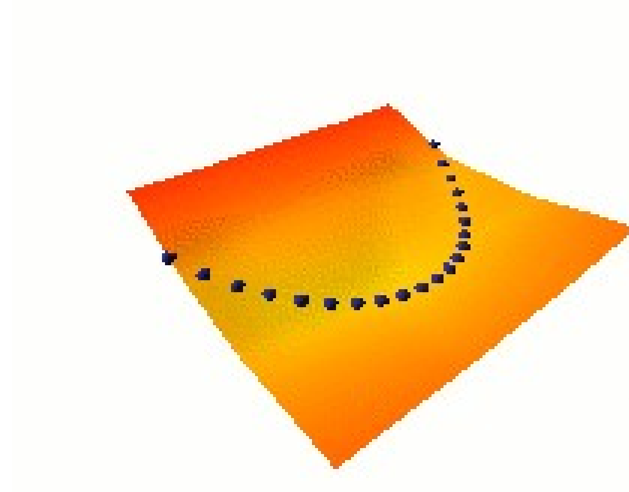
- [LeCun-Huang 2005]
- Appropriate for positive energy functions





(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]

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

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

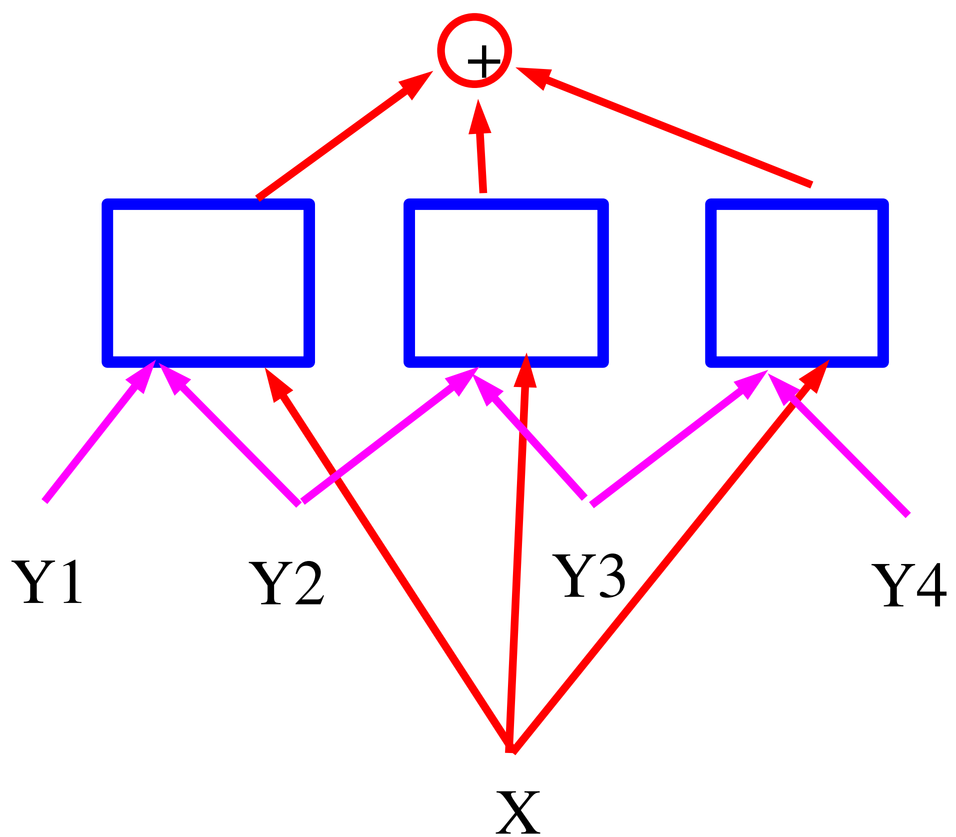
- **Loss functions differ in how they pick the point(s) whose energy is pulled up, and how much they pull them up**
- **Losses with a log partition function in the contrastive term pull up all the bad answers simultaneously.**
  - This may be good if the gradient of the contrastive term can be computed 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

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

Sequence Labeling

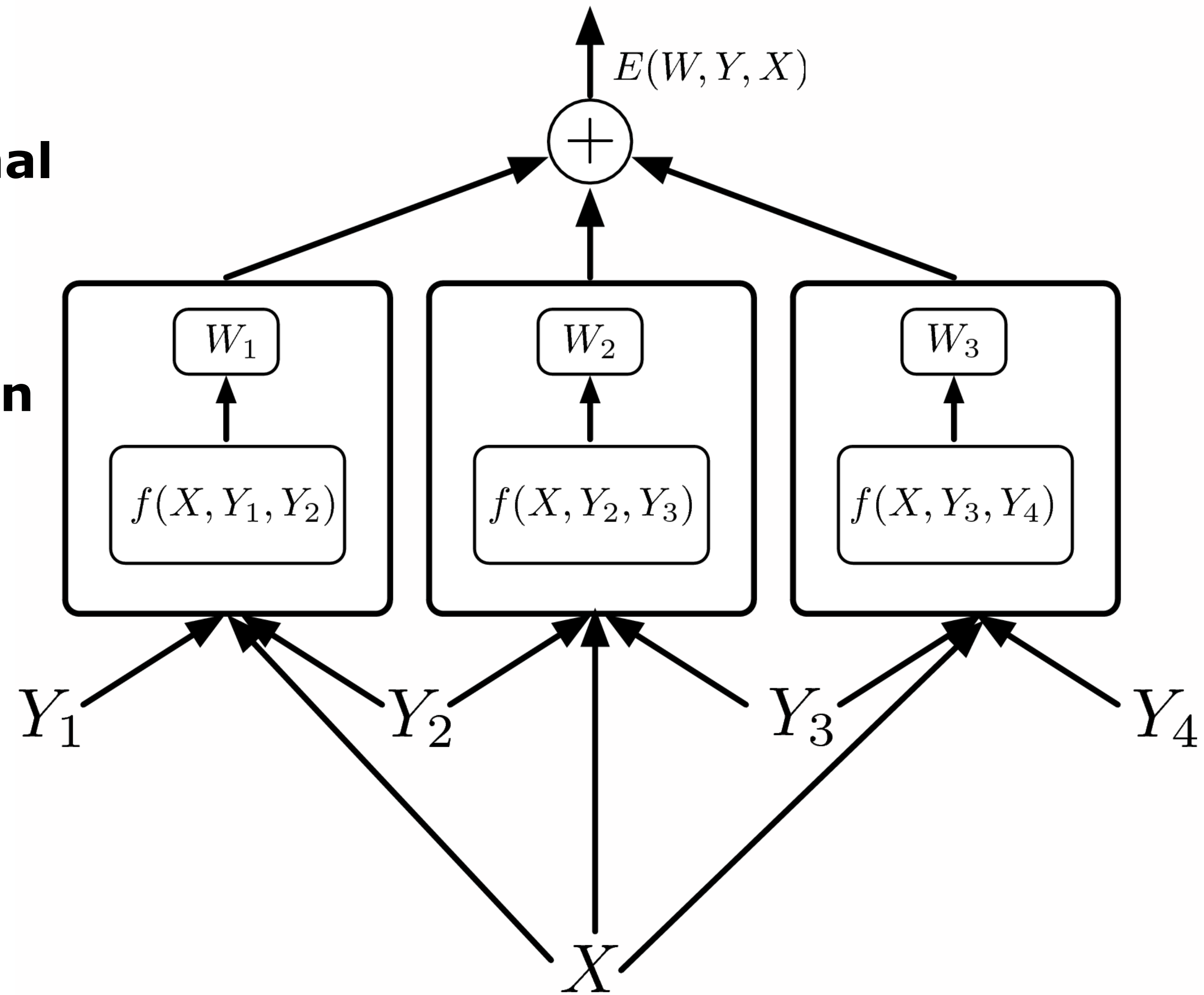
- Output is a sequence Y1,Y2,Y3,Y4......
- NLP parsing, MT, speech/handwriting recognition, biological sequence analysis
- The factors ensure grammatical consistency
- They give low energy to consistent sub-sequences of output symbols
- The graph is generally simple (chain or tree)
- Inference is easy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

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



Simple Energy-Based Factor Graphs with "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



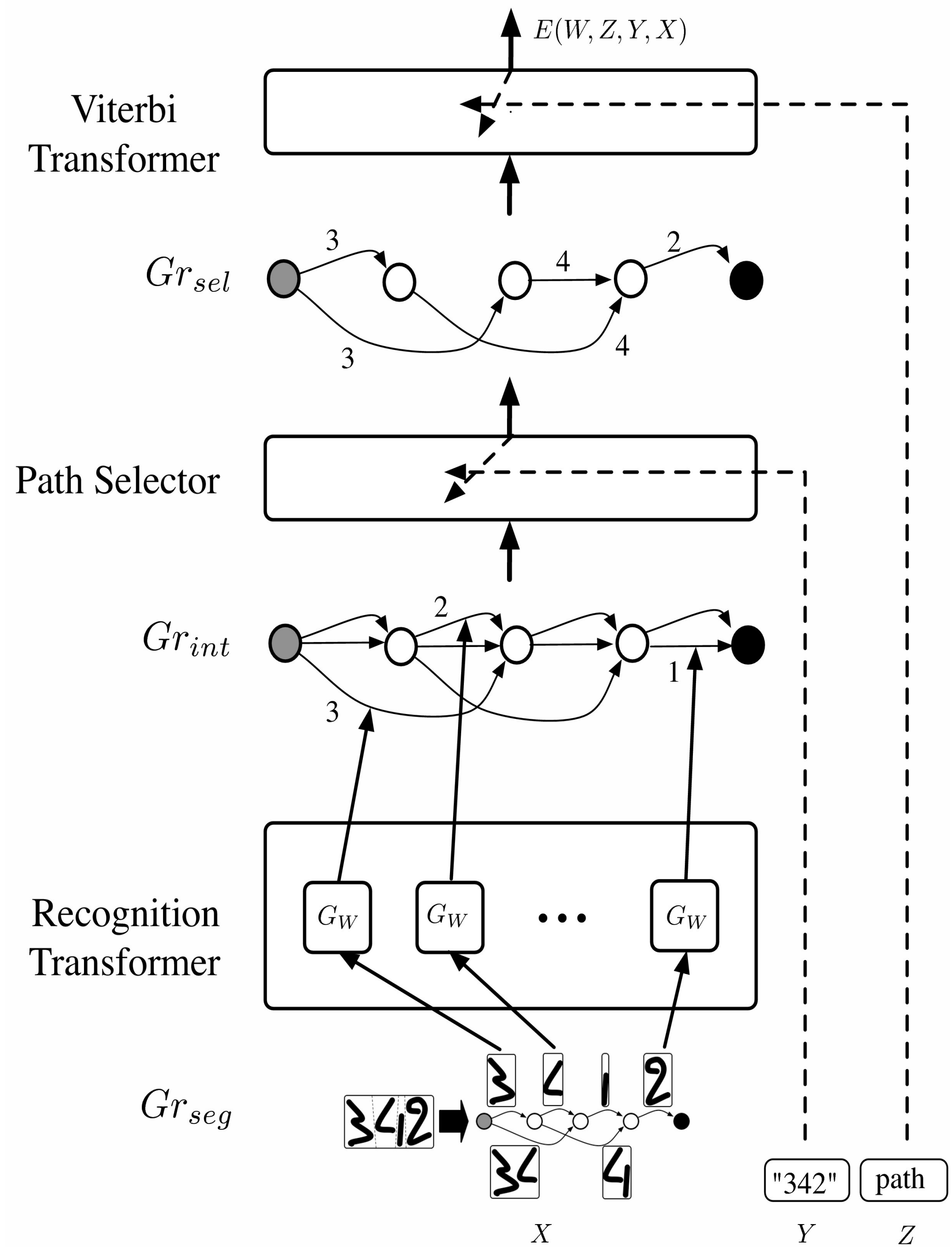
Deep/non-linear Factors for Speech and Handwriting

- 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)

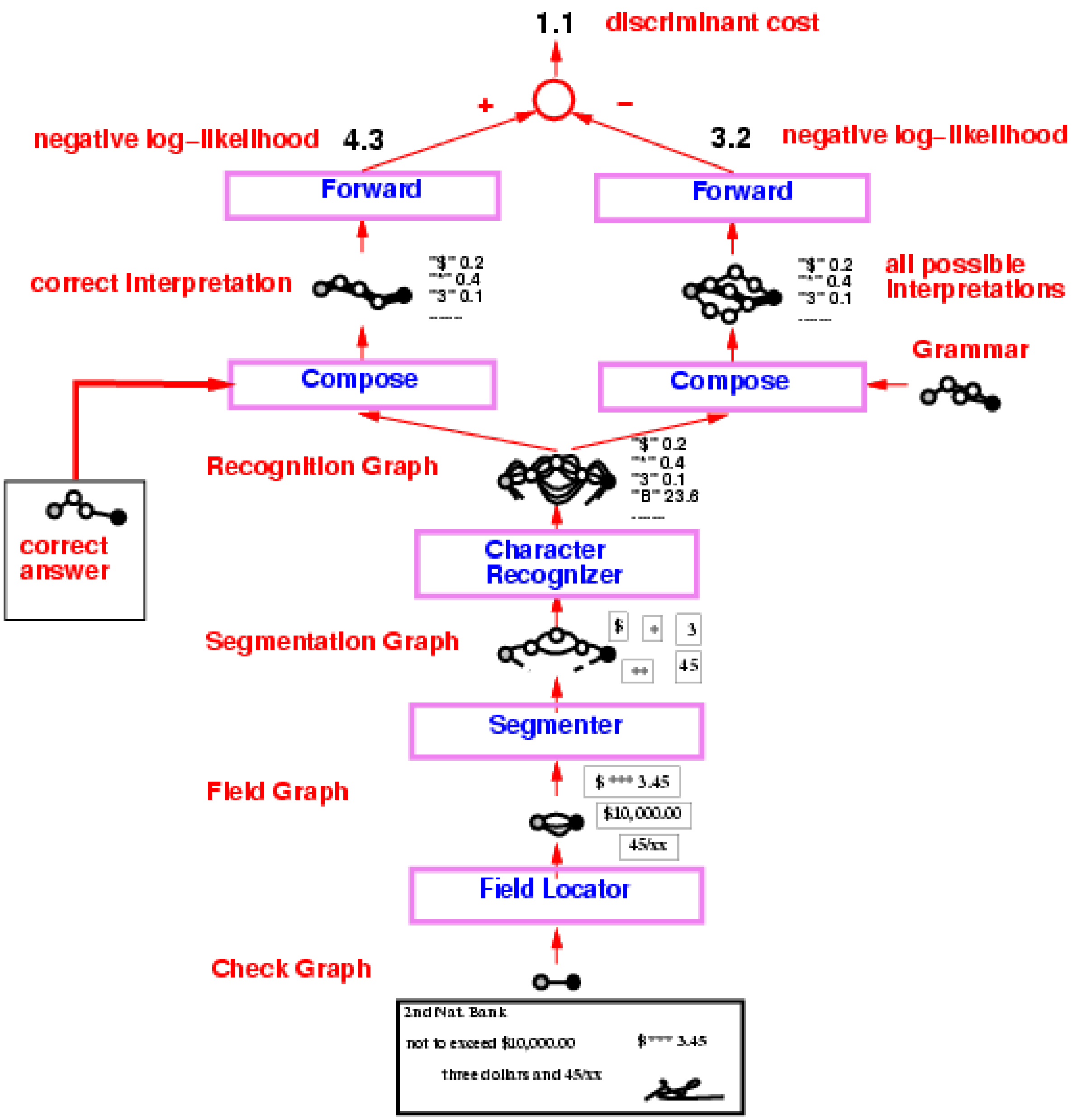
Really Deep Factors & implicit graphs: GTN

- **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.



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.**

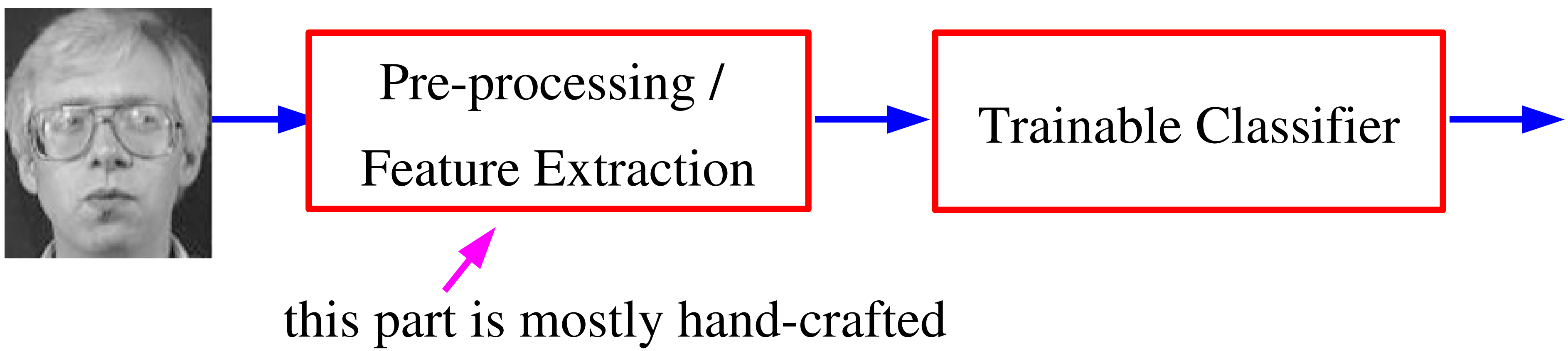
EBM

- 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

Part 2: Deep Supervised Learning for Vision: The Convolutional Network Architecture

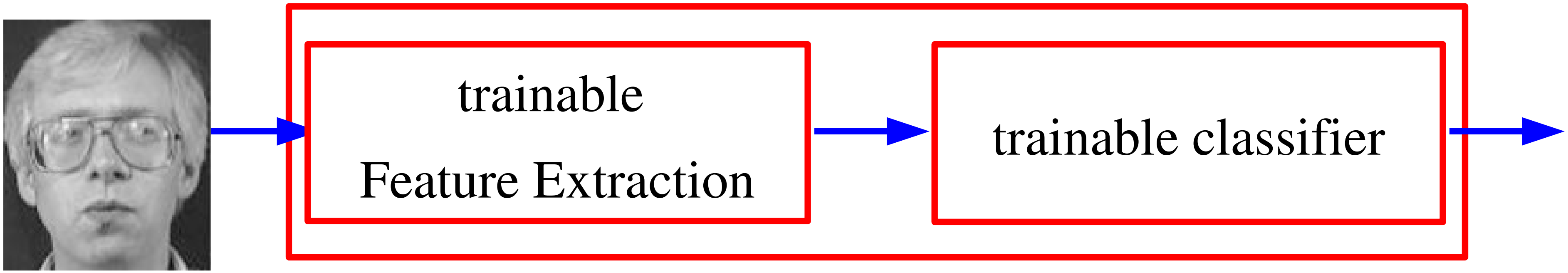
- **Convolutional Networks:**
  - [LeCun et al., Neural Computation, 1988]
  - [LeCun et al., Proc IEEE 1998]
- **Face Detection and pose estimation with convolutional networks:**
  - [Vaillant, Monrocq, LeCun, IEE Proc Vision, Image and Signal Processing, 1994]
  - [Osadchy, Miller, LeCun, JMLR vol 8, May 2007]
- **Category-level object recognition with invariance to pose and lighting**
  - [LeCun, Huang, Bottou, CVPR 2004]
  - [Huang, LeCun, CVPR 2005]
- **autonomous robot driving**
  - [LeCun et al. NIPS 2005]

The Traditional Architecture for Recognition



- **The raw input is pre-processed through a hand-crafted feature extractor**
- **The trainable classifier is often generic (task independent)**

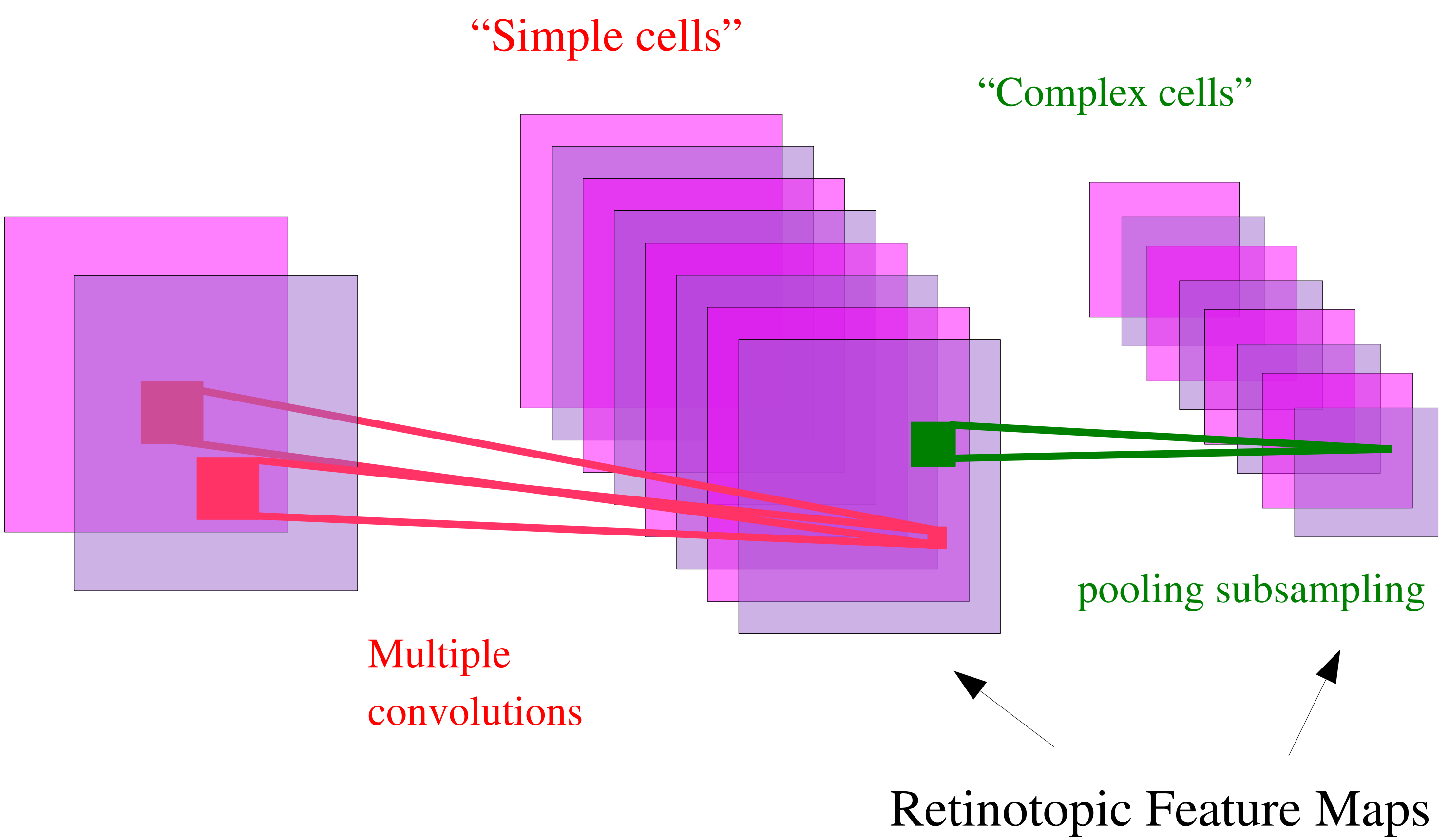
End-to-End Learning



- **The entire system is integrated and trainable "end-to-end".**
- **In some of the models presented here, there will be no discernible difference between the feature extractor and the classifier.**
- **We can embed general prior knowledge about images into the architecture of the system.**

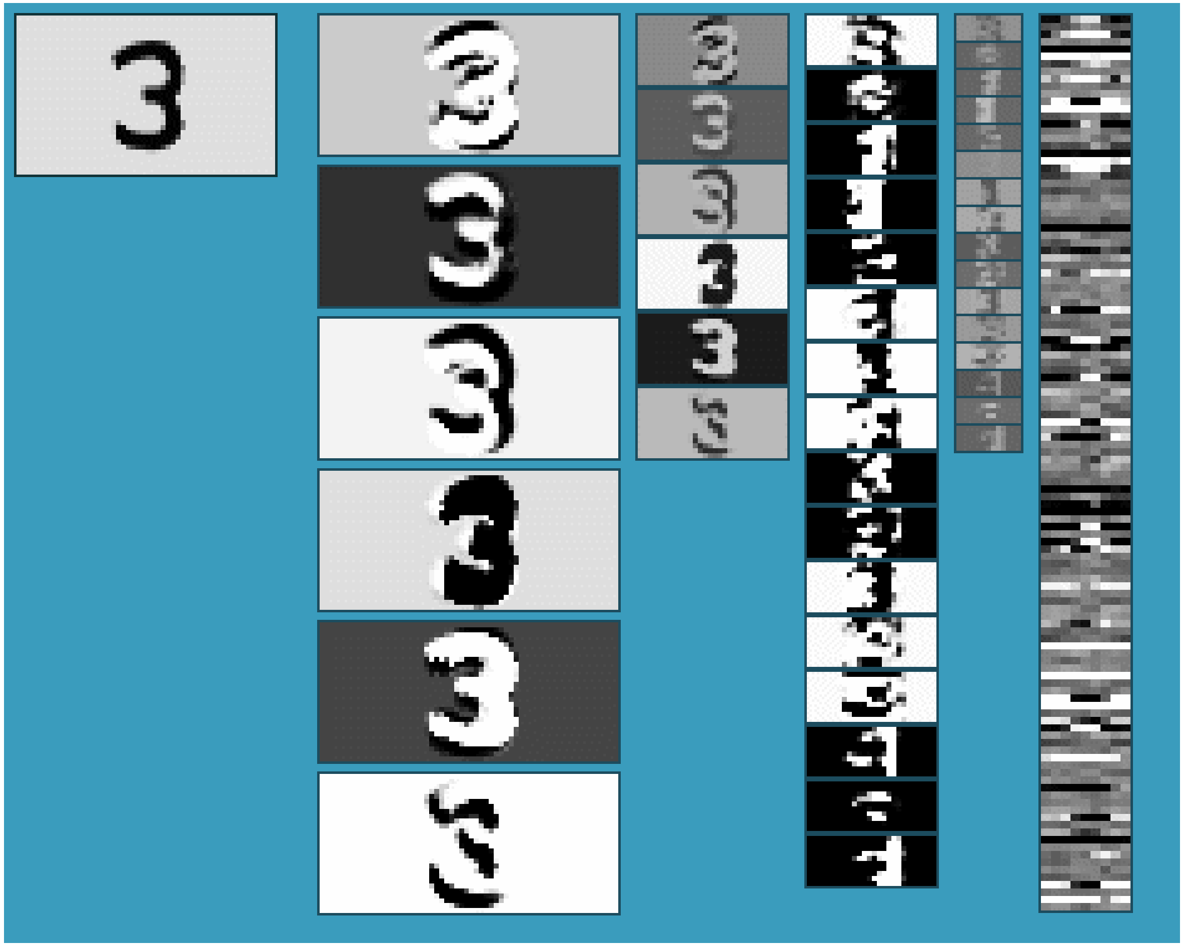
An Old Idea for Local Shift Invariance

- **[Hubel & Wiesel 1962]: architecture of the cat's visual cortex**
  - simple cells detect local features
  - complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.



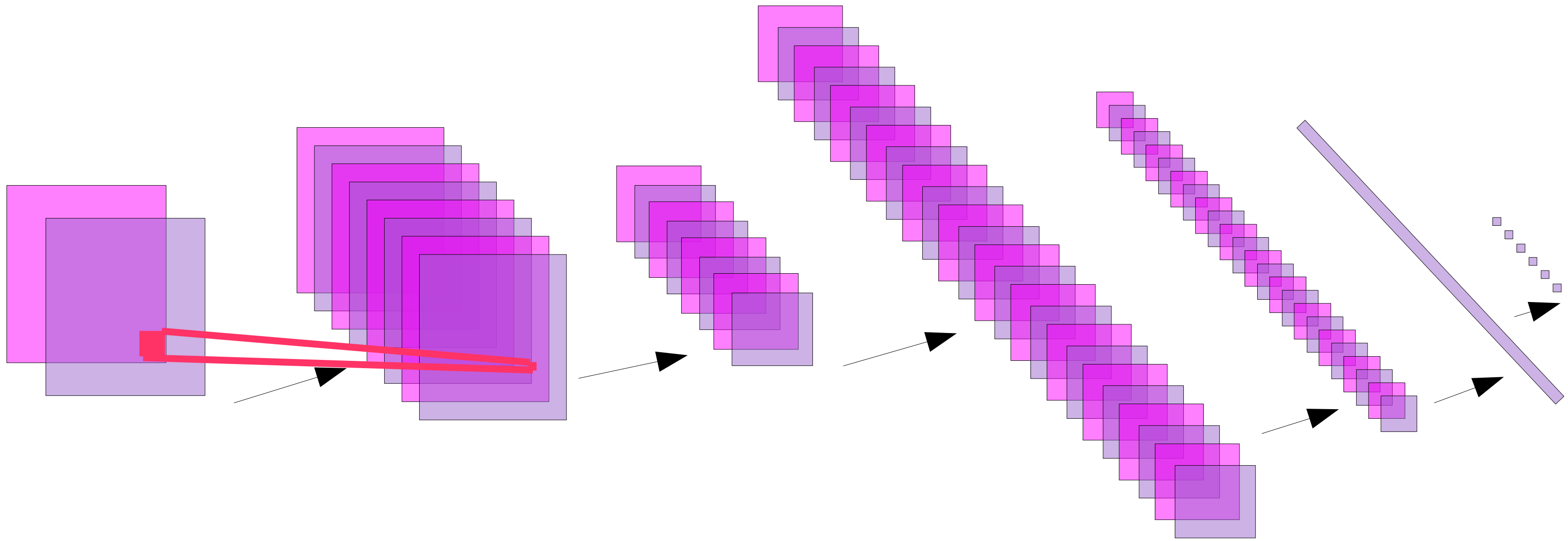
The Multistage Hubel-Wiesel Architecture

- **Building a complete artificial vision system:**
  - Stack multiple stages of simple cells / complex cells layers
  - Higher stages compute more global, more invariant features
  - Stick a classification layer on top
  - [Fukushima 1971-1982]
    - neocognitron
  - [LeCun 1988-2007]
    - convolutional net
  - [Poggio 2002-2006]
    - HMAX
  - [Ullman 2002-2006]
    - fragment hierarchy
  - [Lowe 2006]
    - HMAX



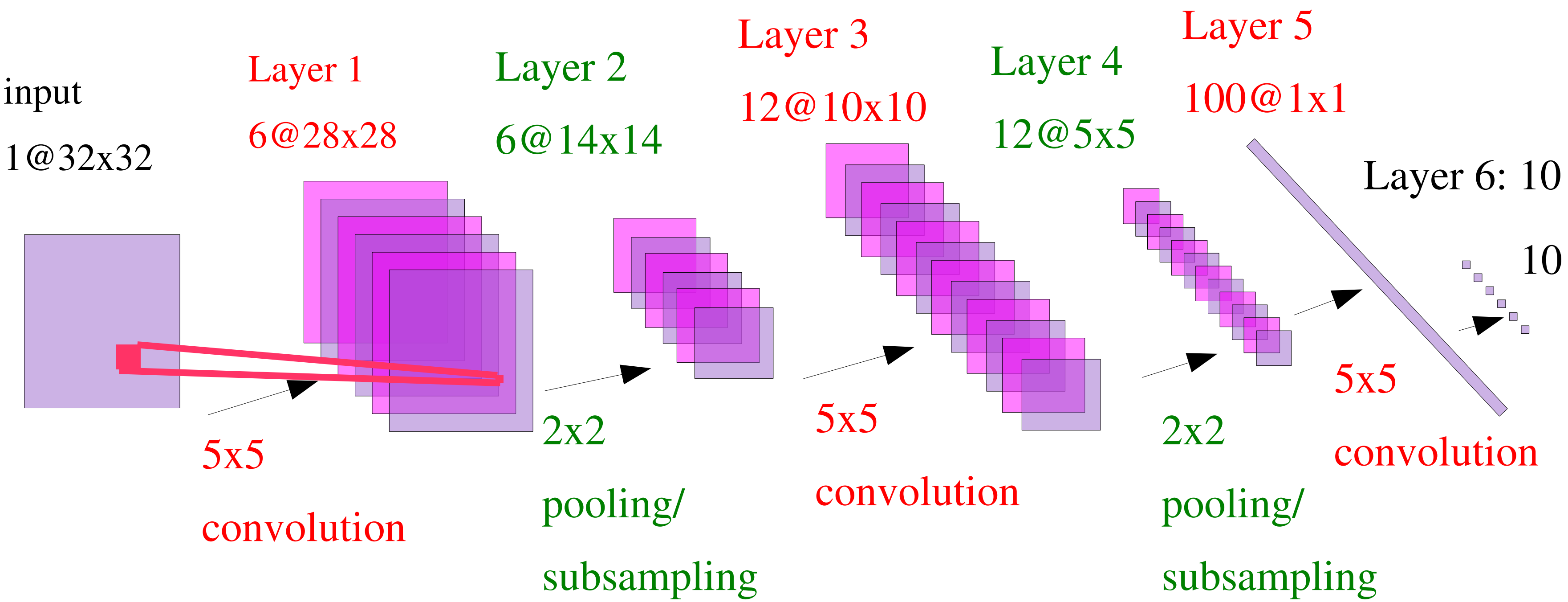
- **QUESTION: How do we find (or learn) the filters?**

Convolutional Network



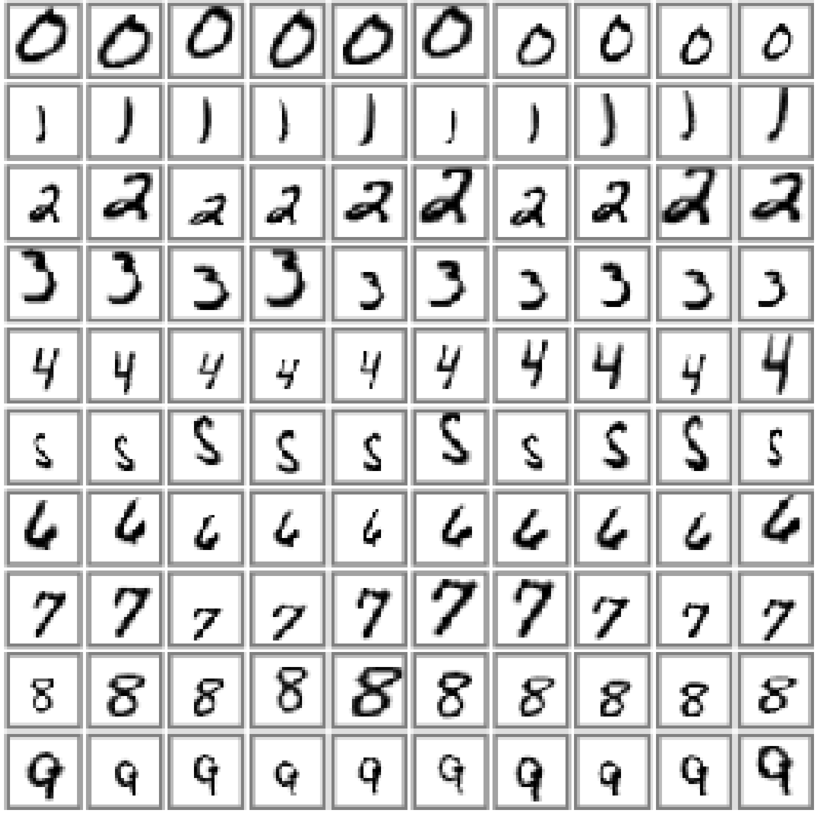
- **Hierarchical/multilayer:** features get progressively more global, invariant, and numerous
- **dense features:** features detectors applied everywhere (no interest point)
- **broadly tuned (possibly invariant) features:** sigmoid units are on half the time.
- **Global discriminative training:** The whole system is trained "end-to-end" with a gradient-based method to minimize a global loss function
- **Integrates segmentation, feature extraction, and invariant classification in one fell swoop.**

Convolutional Net Architecture



- **Convolutional net for handwriting recognition** (400,000 synapses)
- Convolutional layers (simple cells): all units in a feature plane share the same weights
- Pooling/subsampling layers (complex cells): for invariance to small distortions.
- **Supervised gradient-descent learning using back-propagation**
- **The entire network is trained end-to-end. All the layers are trained simultaneously**.

MNIST Handwritten Digit Dataset



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

Results on MNIST Handwritten Digits

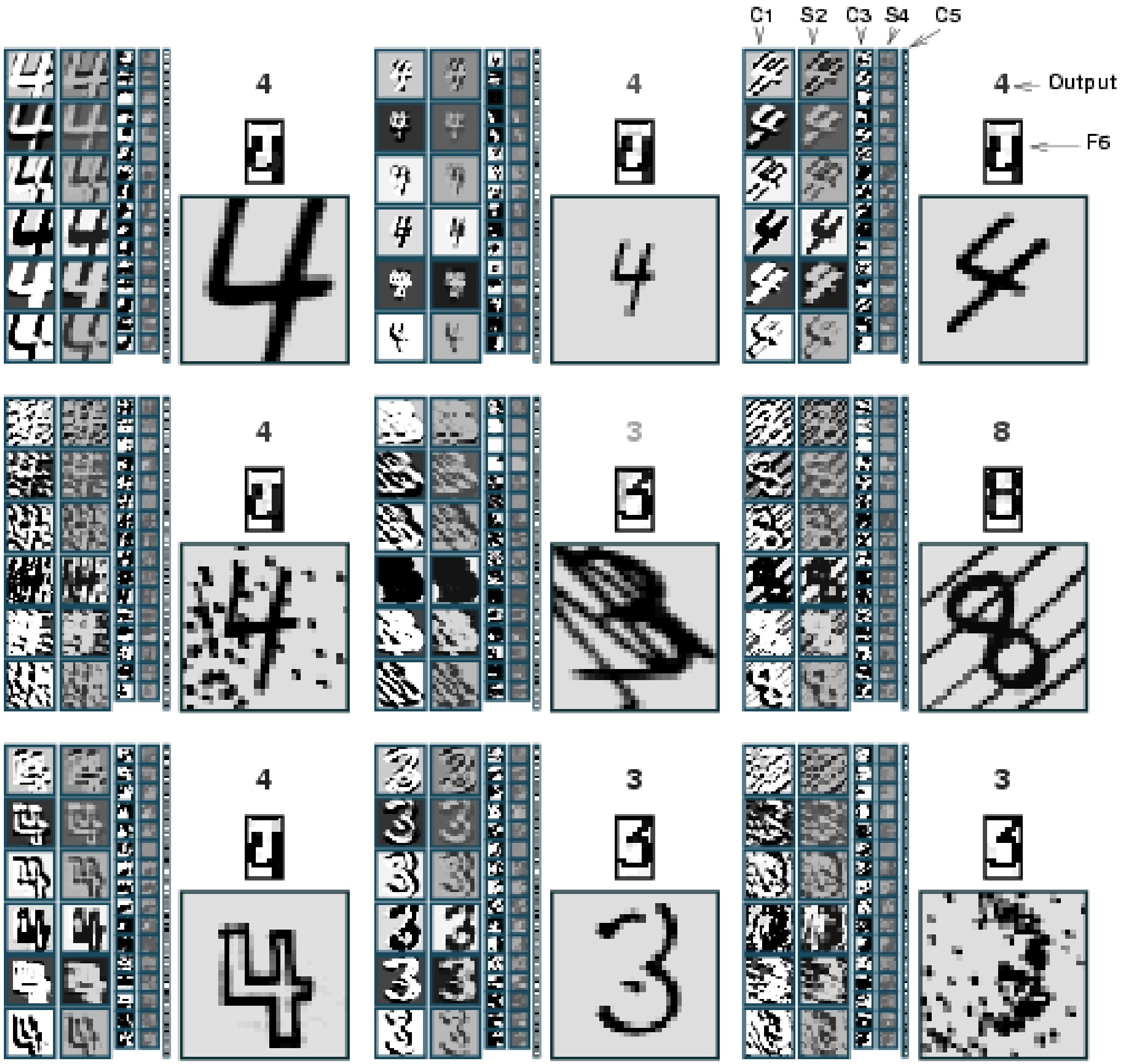
CLASSIFIER	DEFORMATION	PREPROCESSING	ERROR (%)	Reference
linear classifier (1-layer NN)		none	12.00	LeCun et al. 1998
linear classifier (1-layer NN)		deskewing	8.40	LeCun et al. 1998
pairwise linear classifier		deskewing	7.60	LeCun et al. 1998
K-nearest-neighbors, (L2)		none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, (L2)		deskewing	2.40	LeCun et al. 1998
K-nearest-neighbors, (L2)		deskew, clean, blur	1.80	Kenneth Wilder, U. Chicago
K-NN L3, 2 pixel jitter		deskew, clean, blur	1.22	Kenneth Wilder, U. Chicago
K-NN, shape context matching		shape context feature	0.63	Belongie et al. IEEE PAMI 2002
40 PCA + quadratic classifier		none	3.30	LeCun et al. 1998
1000 RBF + linear classifier		none	3.60	LeCun et al. 1998
K-NN, Tangent Distance		subsamp 16x16 pixels	1.10	LeCun et al. 1998
SVM, Gaussian Kernel		none	1.40	
SVM deg 4 polynomial		deskewing	1.10	LeCun et al. 1998
Reduced Set SVM deg 5 poly		deskewing	1.00	LeCun et al. 1998
Virtual SVM deg-9 poly	Affine	none	0.80	LeCun et al. 1998
V-SVM, 2-pixel jittered		none	0.68	DeCoste and Scholkopf, MLJ 2002
V-SVM, 2-pixel jittered		deskewing	0.56	DeCoste and Scholkopf, MLJ 2002
2-layer NN, 300 HU, MSE		none	4.70	LeCun et al. 1998
2-layer NN, 300 HU, MSE,	Affine	none	3.60	LeCun et al. 1998
2-layer NN, 300 HU		deskewing	1.60	LeCun et al. 1998
3-layer NN, 500+150 HU		none	2.95	LeCun et al. 1998
3-layer NN, 500+150 HU	Affine	none	2.45	LeCun et al. 1998
3-layer NN, 500+300 HU, CE, reg		none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, CE		none	1.60	Simard et al., ICDAR 2003
2-layer NN, 800 HU, CE	Affine	none	1.10	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE	Elastic	none	0.90	Simard et al., ICDAR 2003
2-layer NN, 800 HU, CE	Elastic	none	0.70	Simard et al., ICDAR 2003
Convolutional net LeNet-1		subsamp 16x16 pixels	1.70	LeCun et al. 1998
Convolutional net LeNet-4		none	1.10	LeCun et al. 1998
Convolutional net LeNet-5,		none	0.95	LeCun et al. 1998
Conv. net LeNet-5,	Affine	none	0.80	LeCun et al. 1998
Boosted LeNet-4	Affine	none	0.70	LeCun et al. 1998
Conv. net, CE	Affine	none	0.60	Simard et al., ICDAR 2003
Comv net, CE	Elastic	none	0.40	Simard et al., ICDAR 2003

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

CLASSIFIER	DEFORMATION	ERROR	Reference
Knowledge-free methods (a fixed permutation of the pixels would make no difference)			
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
???		0.95	
Convolutional nets			
Convolutional net LeNet-5,		0.80	Ranzato et al. NIPS 2006
Convolutional net LeNet-6,		0.70	Ranzato et al. NIPS 2006
???		0.60	
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
???		0.39	

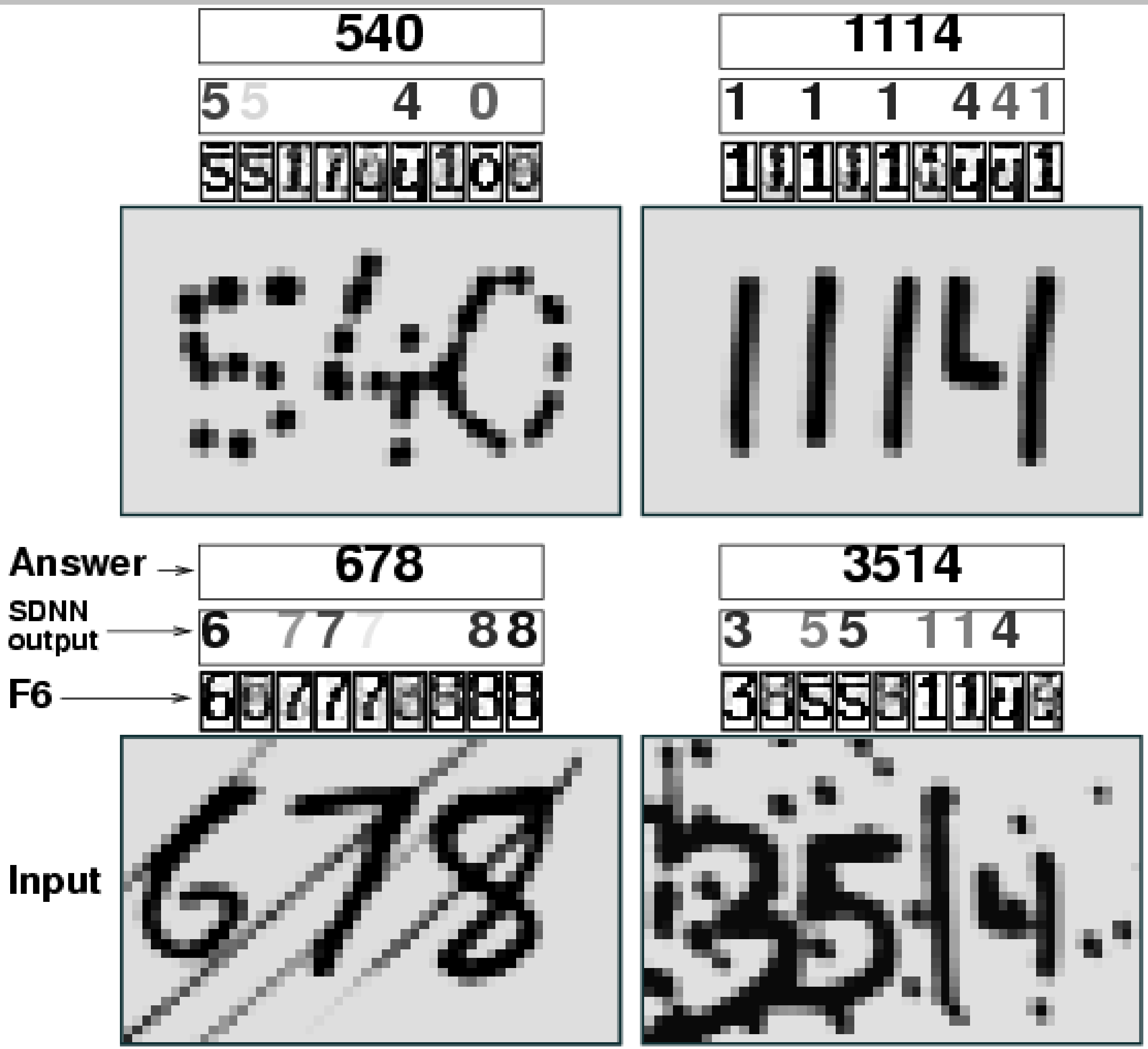
Note: some groups have obtained good results with various amounts of preprocessing: [deCoste and Schoelkopf] get 0.56% with an SVM on deskewed images; [Belongie] get 0.63% with “shape context” features; [CENPARMI] get below 0.4% with features and SVM; [Liu] get 0.42% with features and SVM.

Invariance and Robustness to Noise



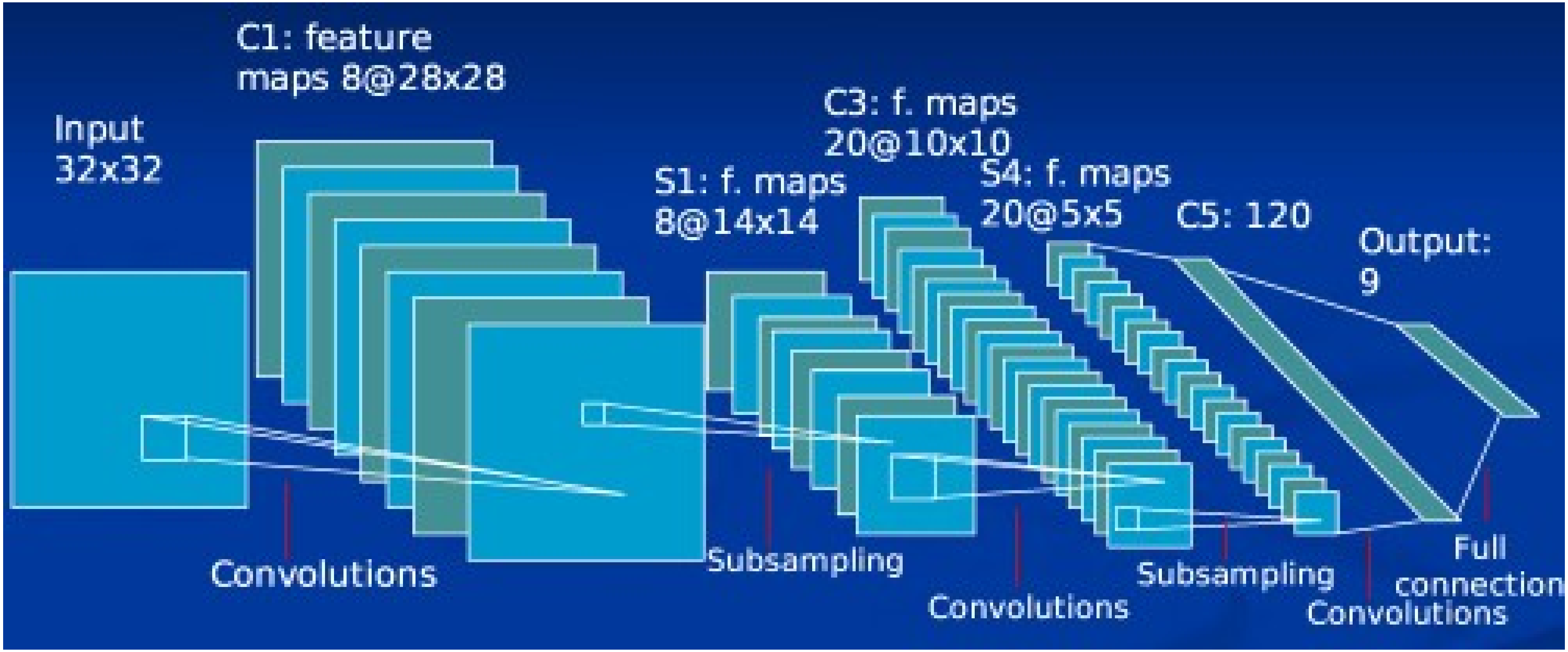
Recognizing Multiple Characters with Replicated Nets

Handwriting Recognition



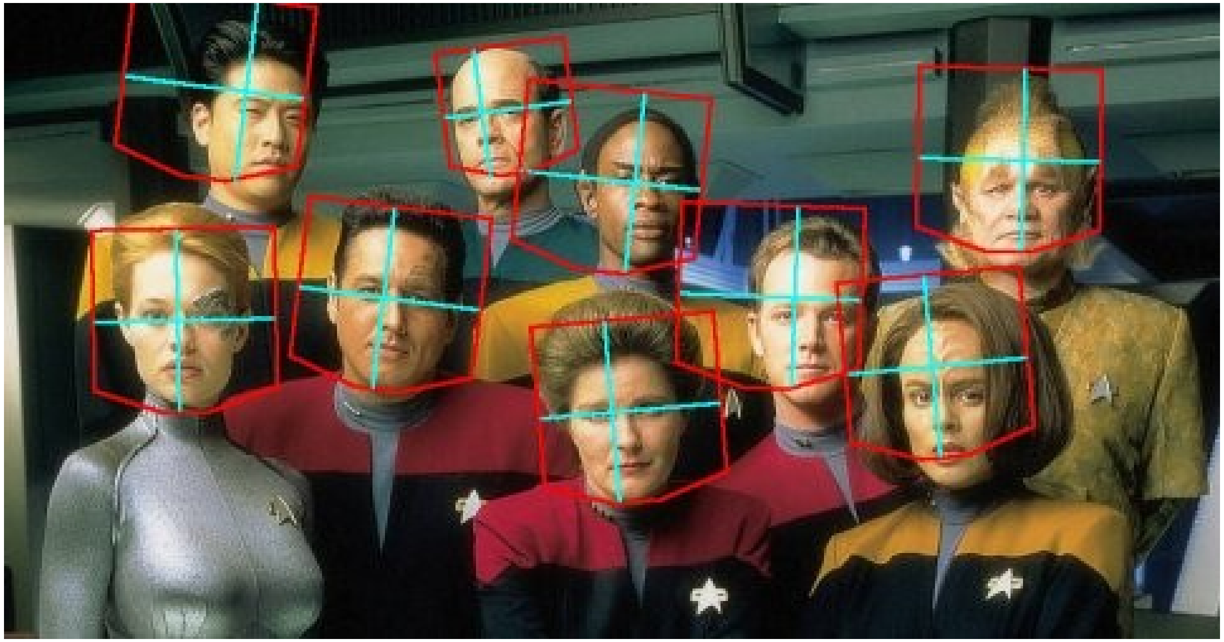
Face Detection and Pose Estimation with Convolutional Nets

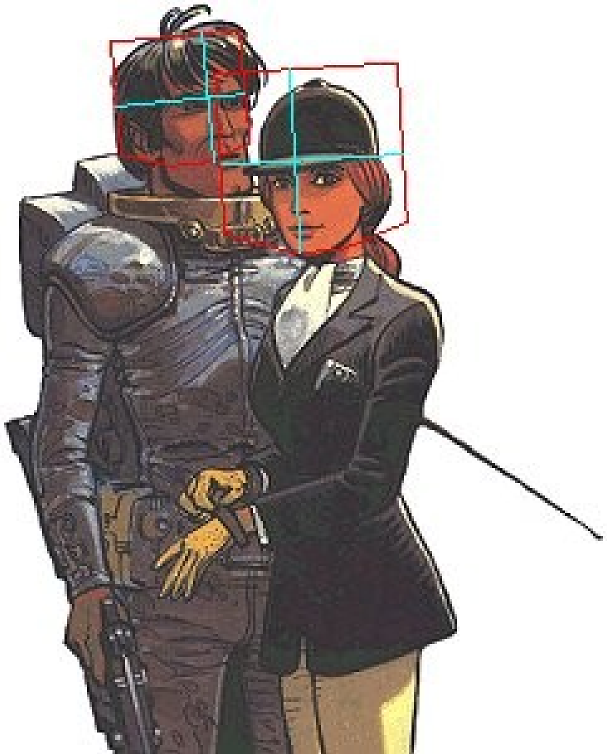
- **Training:** 52,850, 32x32 grey-level images of faces, 52,850 non-faces.
- **Each sample:** used 5 times with random variation in scale, in-plane rotation, brightness and contrast.
- **2nd phase:** half of the initial negative set was replaced by false positives of the initial version of the detector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



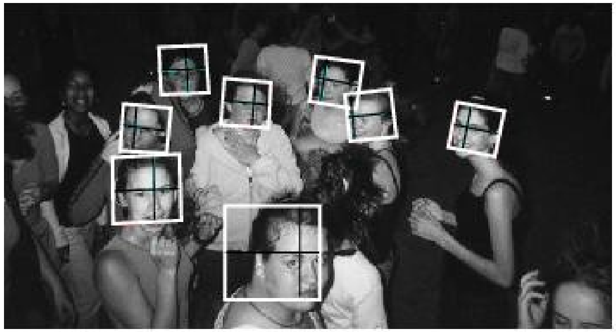
Face Detection: Results

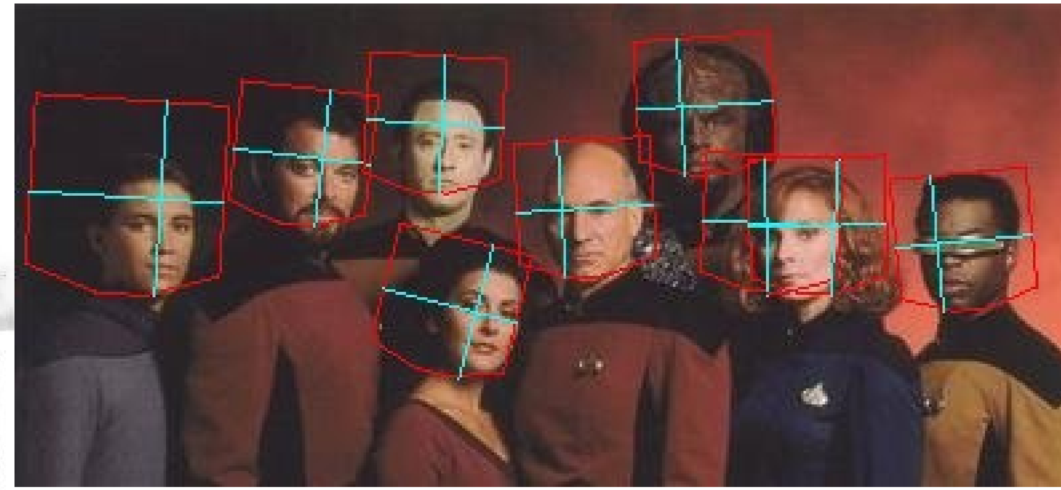
Data Set->	TILTED		PROFILE		MIT+CMU	
False positives per image->	4.42	26.9	0.47	3.36	0.5	1.28
Our Detector	90%	97%	67%	83%	83%	88%
Jones & Viola (tilted)	90%	95%	x		x	
Jones & Viola (profile)	x		70%	83%	x	

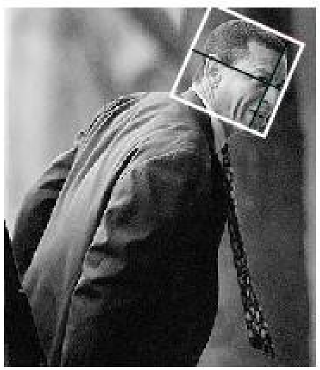


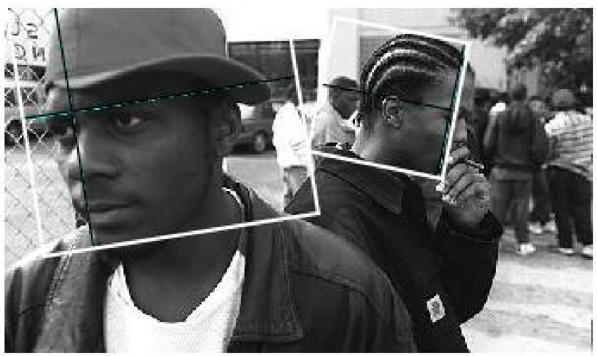


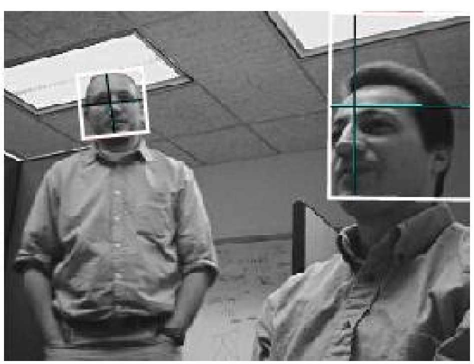
Face Detection and Pose Estimation: Results

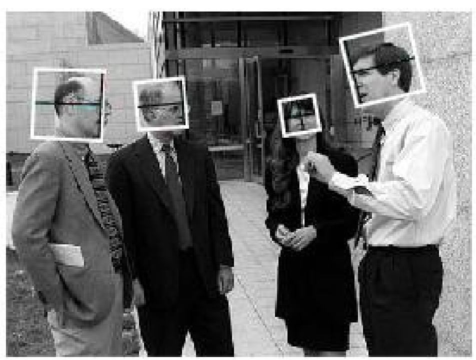


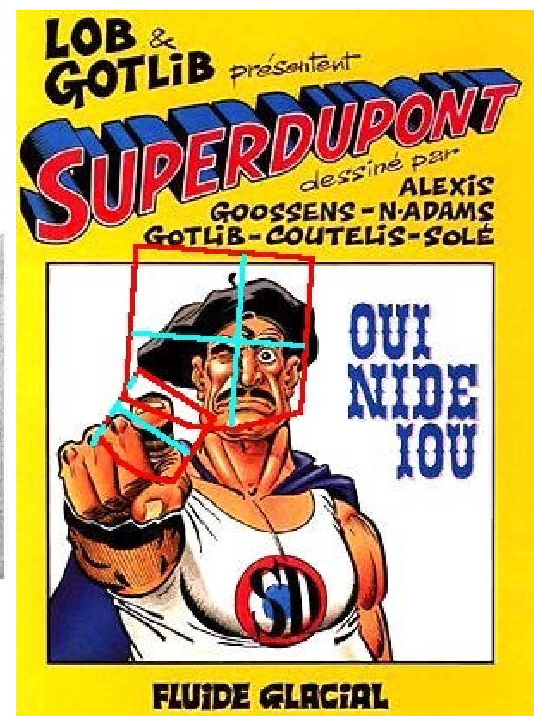


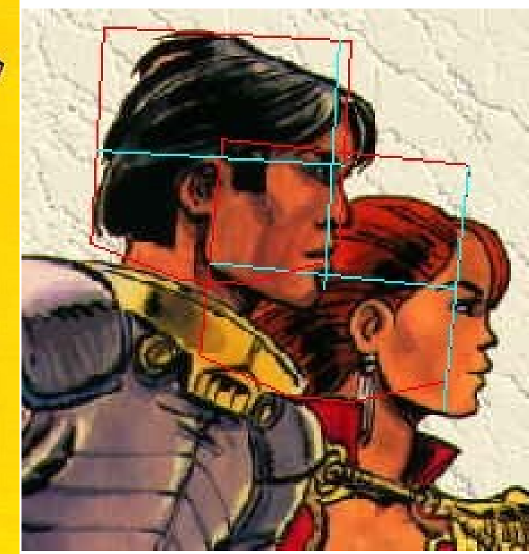




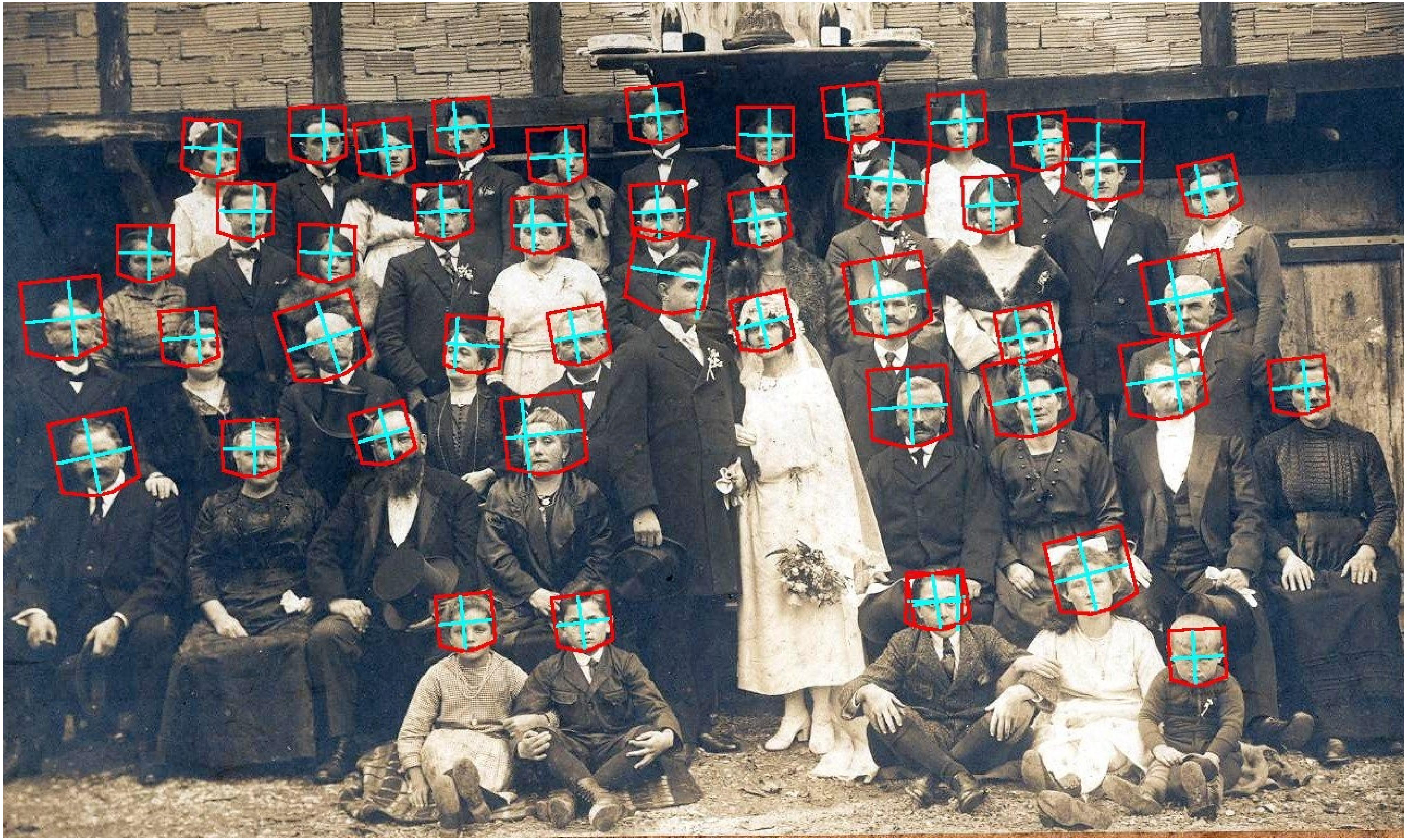




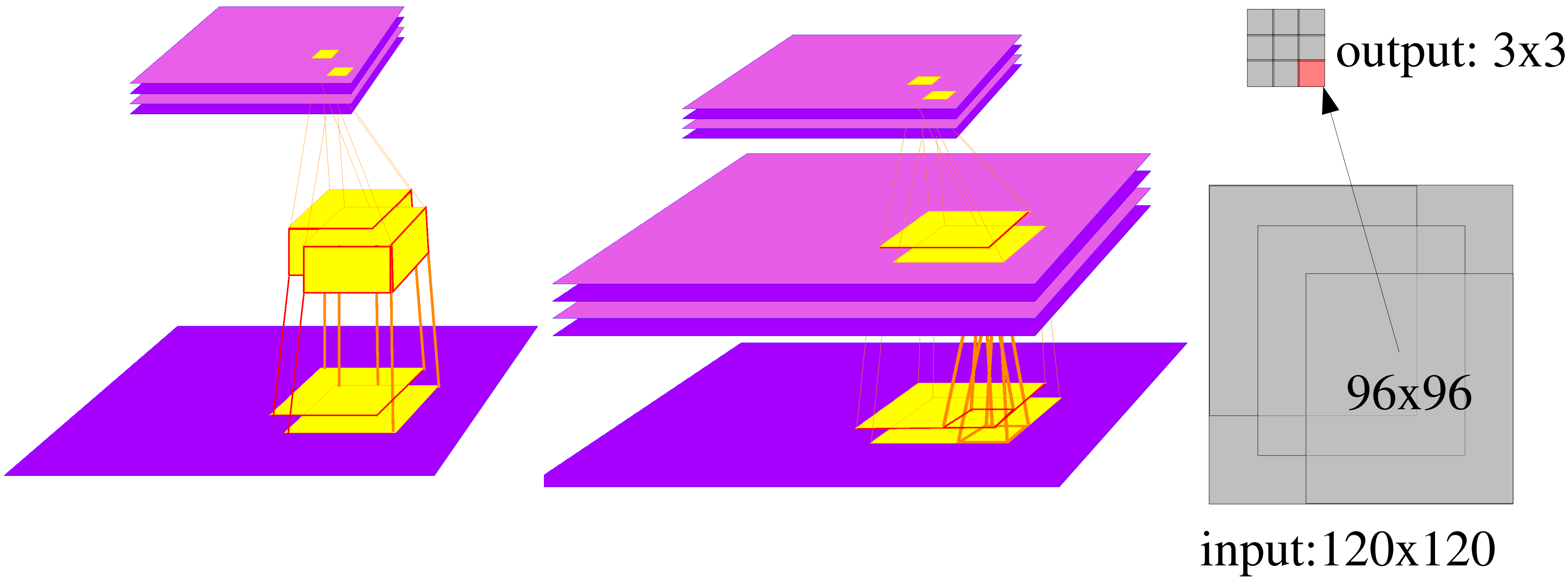




Face Detection with a Convolutional Net



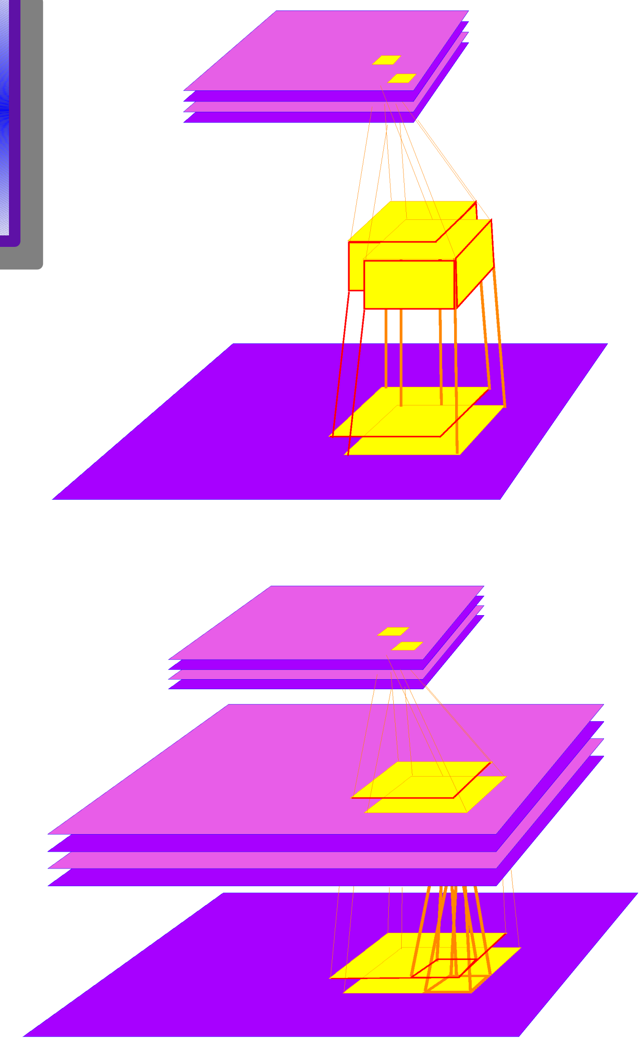
Applying a ConvNet on Sliding Windows is Very Cheap!

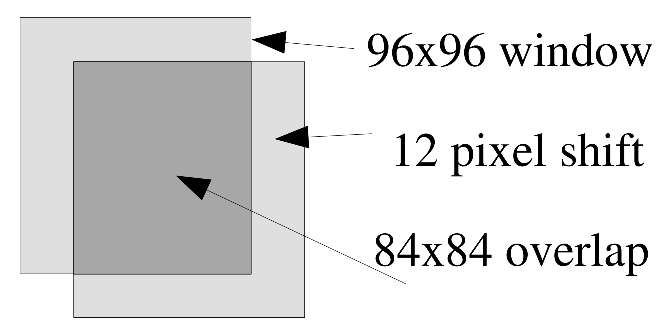


- Traditional Detectors/Classifiers must be applied to every location on a large input image, at multiple scales.
- Convolutional nets can replicated over large images very cheaply.
- The network is applied to multiple scales spaced by 1.5.

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

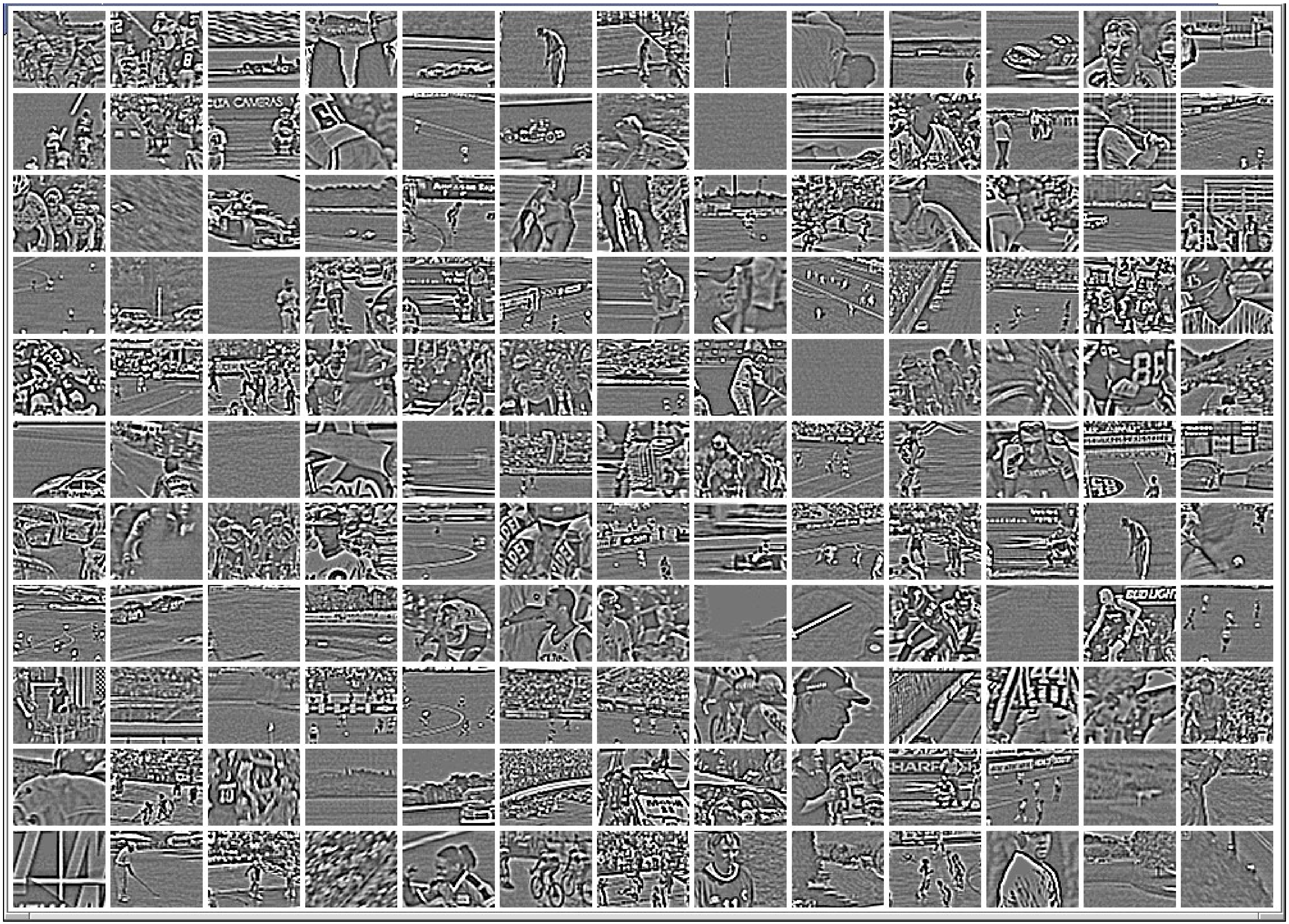




TV sport categorization (with Alex Niculescu, Cornell)

- Classifying TV sports snapshots into 7 categories: auto racing, baseball, basketball, bicycle, golf, soccer, football.
- 123,900 training images (300 sequence with 59 frames for each sport)
- 82,600 test images (200 sequences with 59 frames for each sport)
- Preprocessing: convert to YUV, high-pass filter the Y component, crop, subsample to 72x60 pixels
- Results:
  - frame-level accuracy: 61% correct
  - Sequence-level accuracy 68% correct (simple voting scheme).

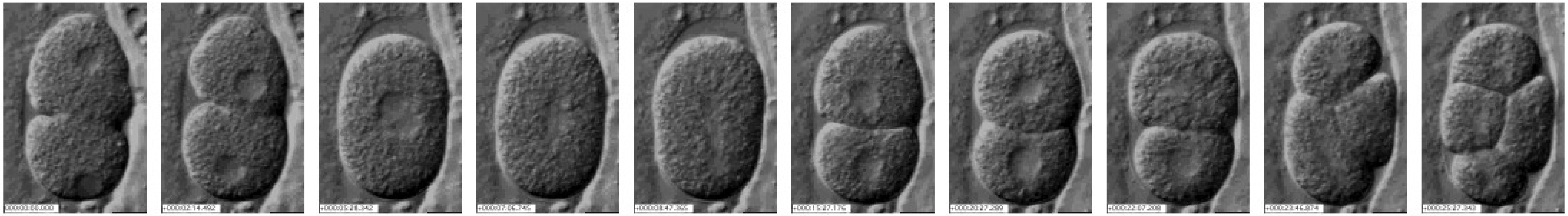
TV sport categorization (with Alex Niculescu, Cornell)



C. Elegans Embryo Phenotyping

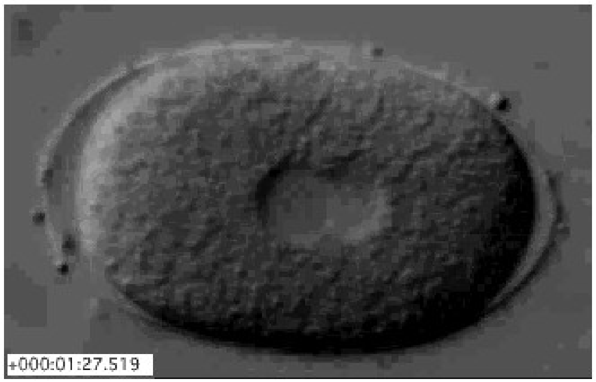
[Ning et al. IEEE Trans. Image Processing, Nov 2005]

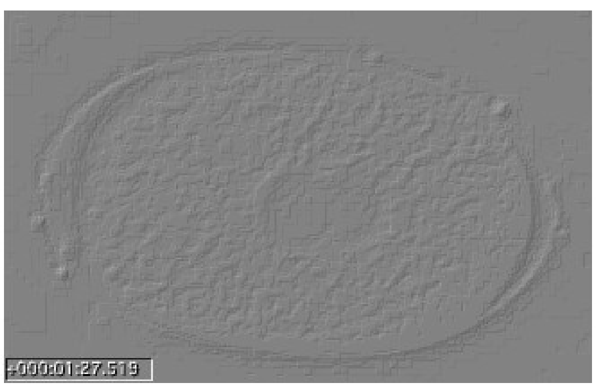
- **Analyzing results for Gene Knock-Out Experiments**

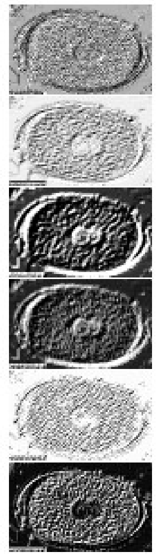


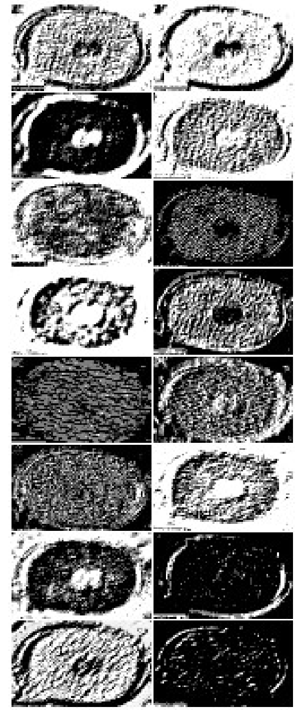
C. Elegans Embryo Phenotyping

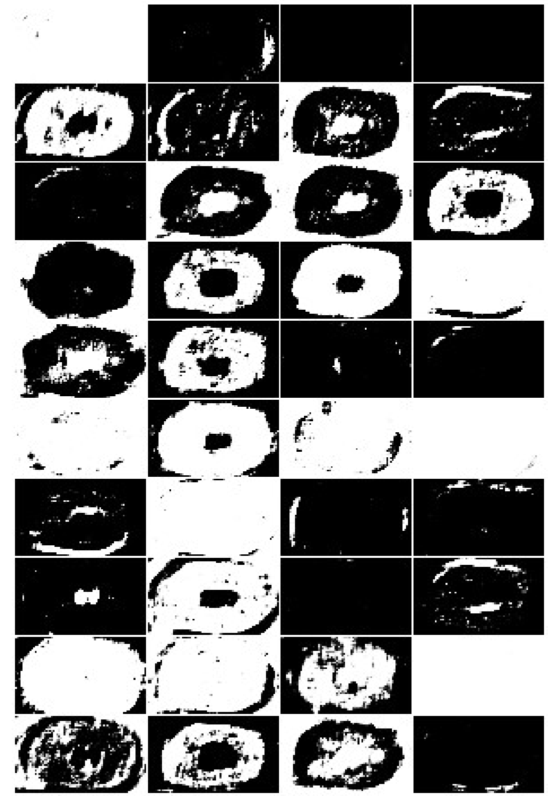
Analyzing results for Gene Knock-Out Experiments

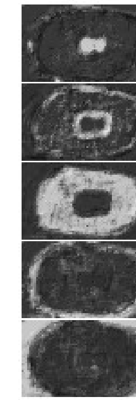




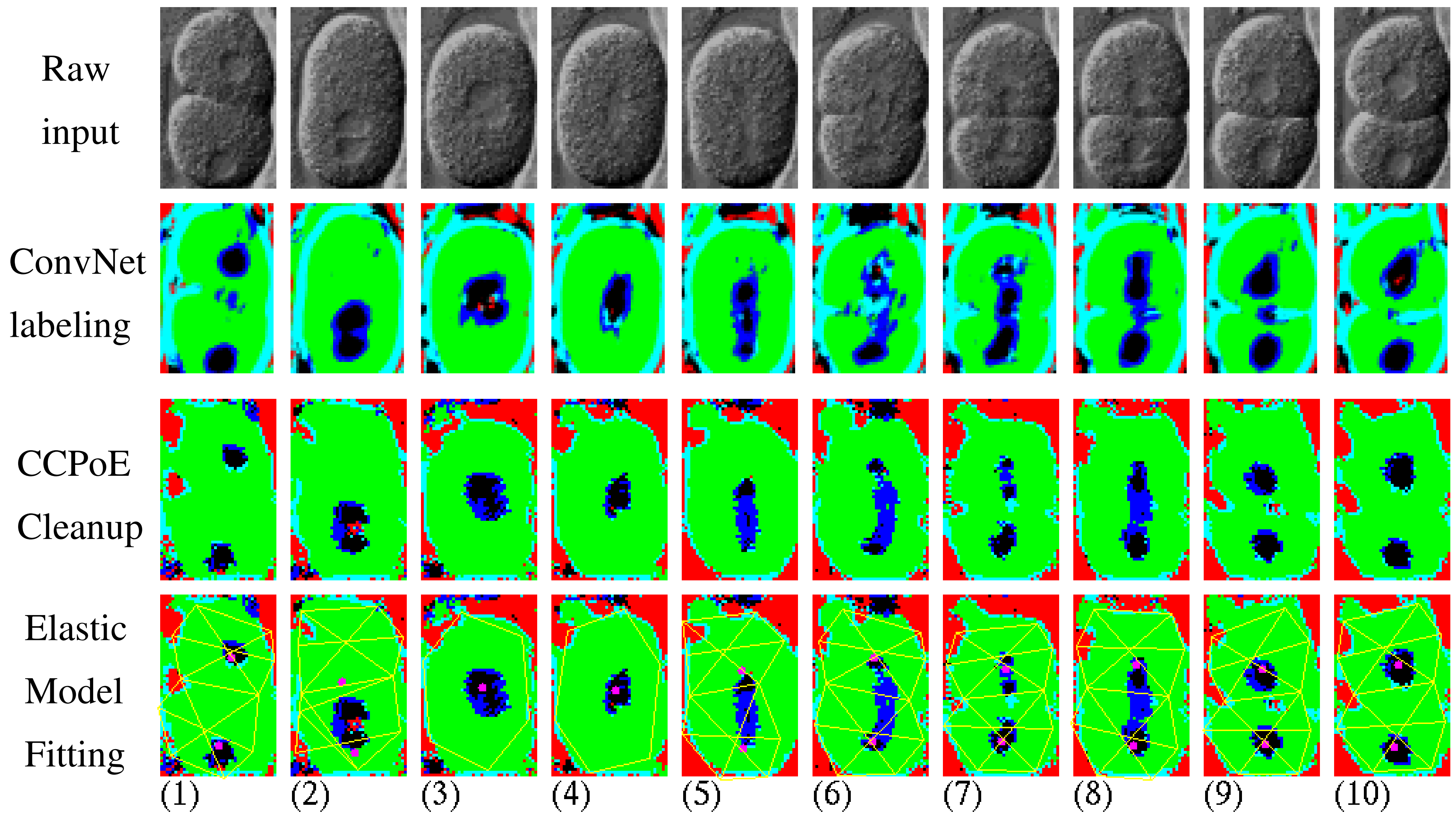








C. Elegans Embryo Phenotyping



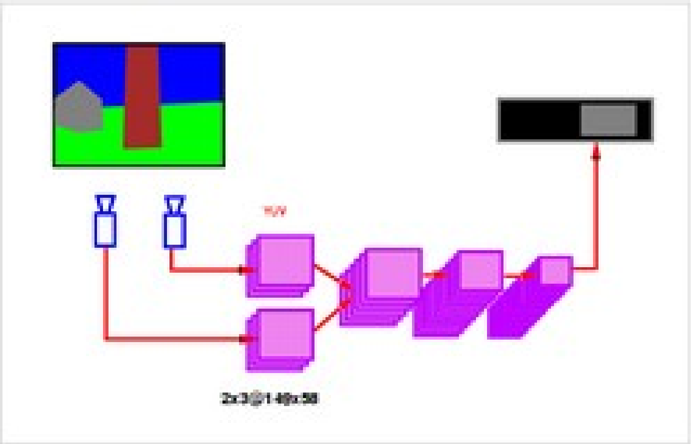
CCPoE = Convolutional Conditional Product of Experts [Ning et al, IEEE TIP 2005]

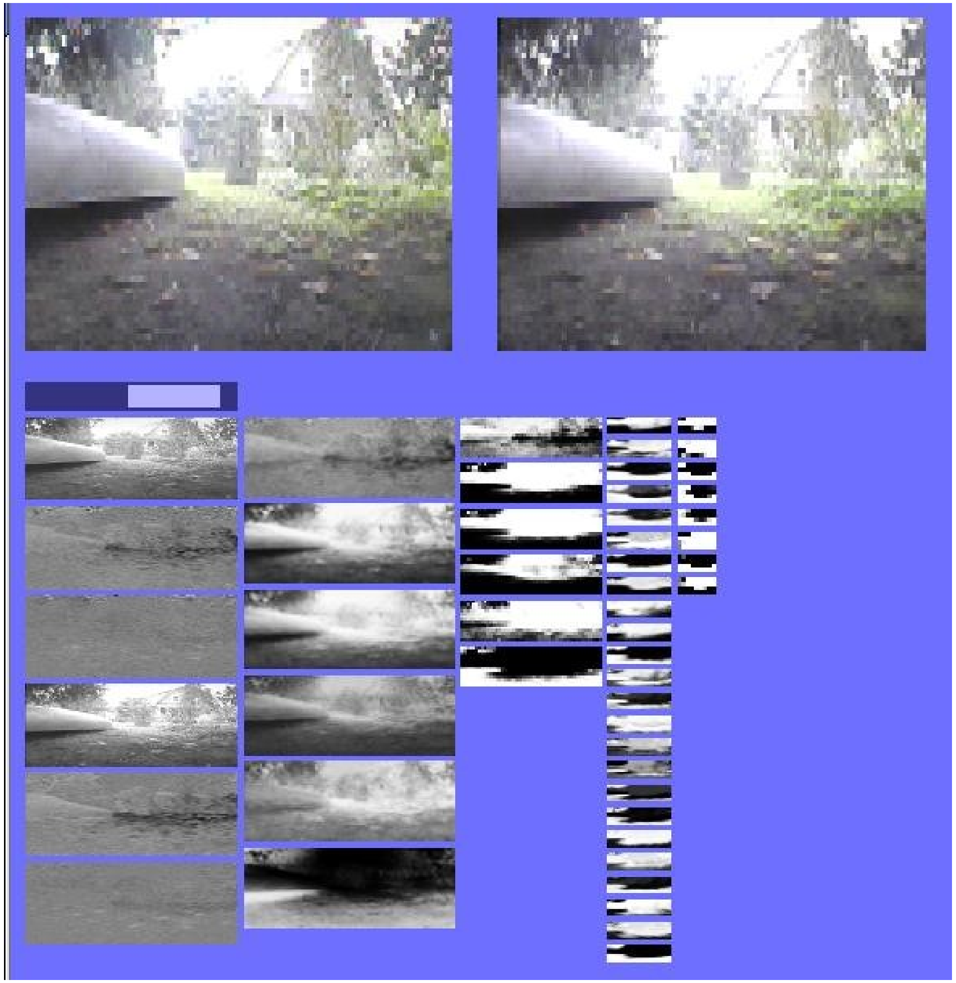
(similar to Field of Experts [Roth & Black, CVPR 2005])

Visual Navigation for a Mobile Robot

[LeCun et al. NIPS 2005]

- 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





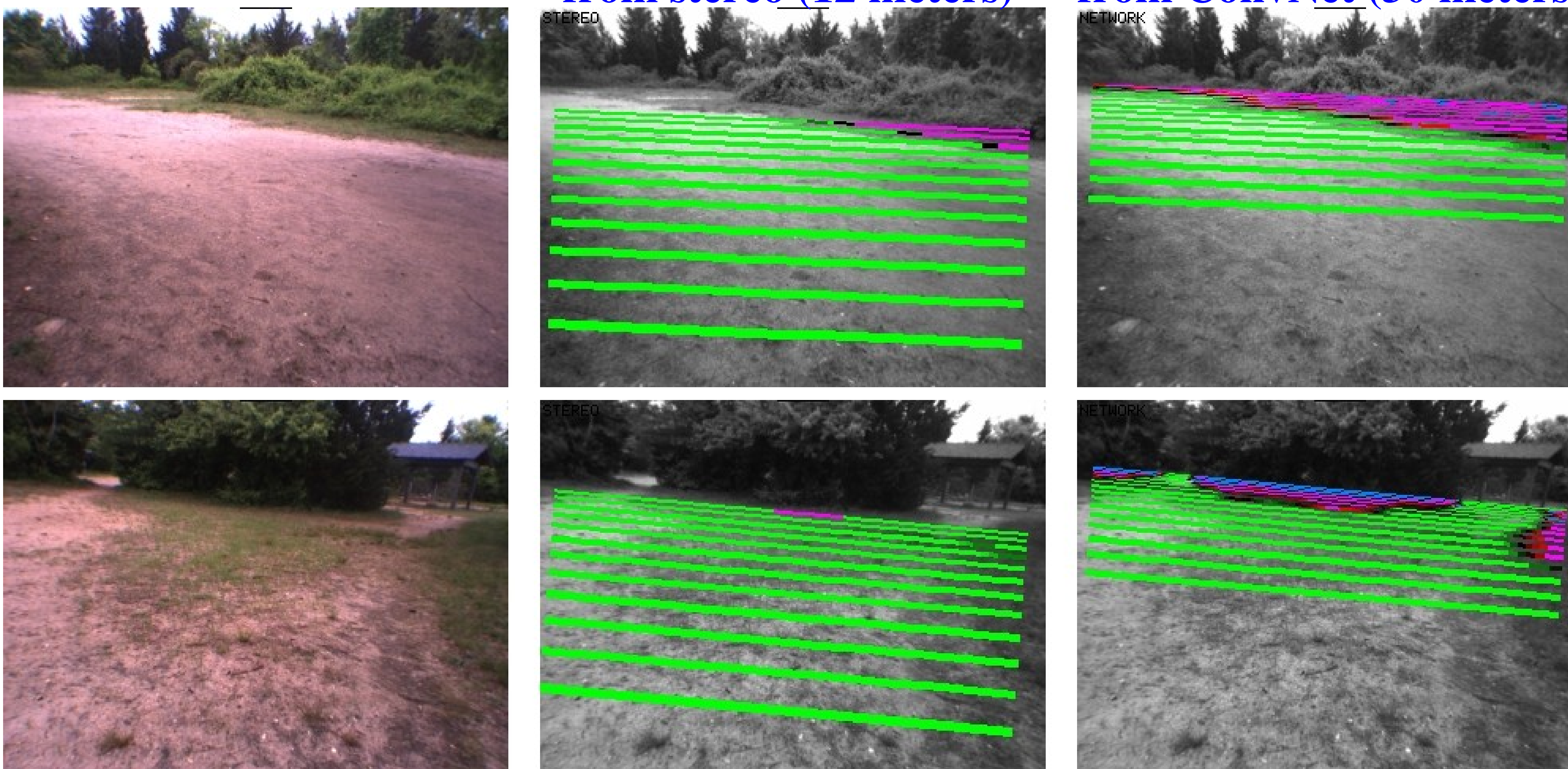
LAGR: Learning Applied to Ground Robotics

- Getting a robot to drive autonomously in unknown terrain solely from vision (camera input).
- Our team (NYU/Net-Scale Technologies Inc.) is one of 8 participants funded by DARPA
- All teams received identical robots and can only modify the software (not the hardware)
- The robot is given the GPS coordinates of a goal, and must drive to the goal as fast as possible. The terrain is unknown in advance. The robot is run 3 times through the same course.

Training a ConvNet On-line to detect obstacles

[Hadsell et al. Robotics Science and Systems 2007]

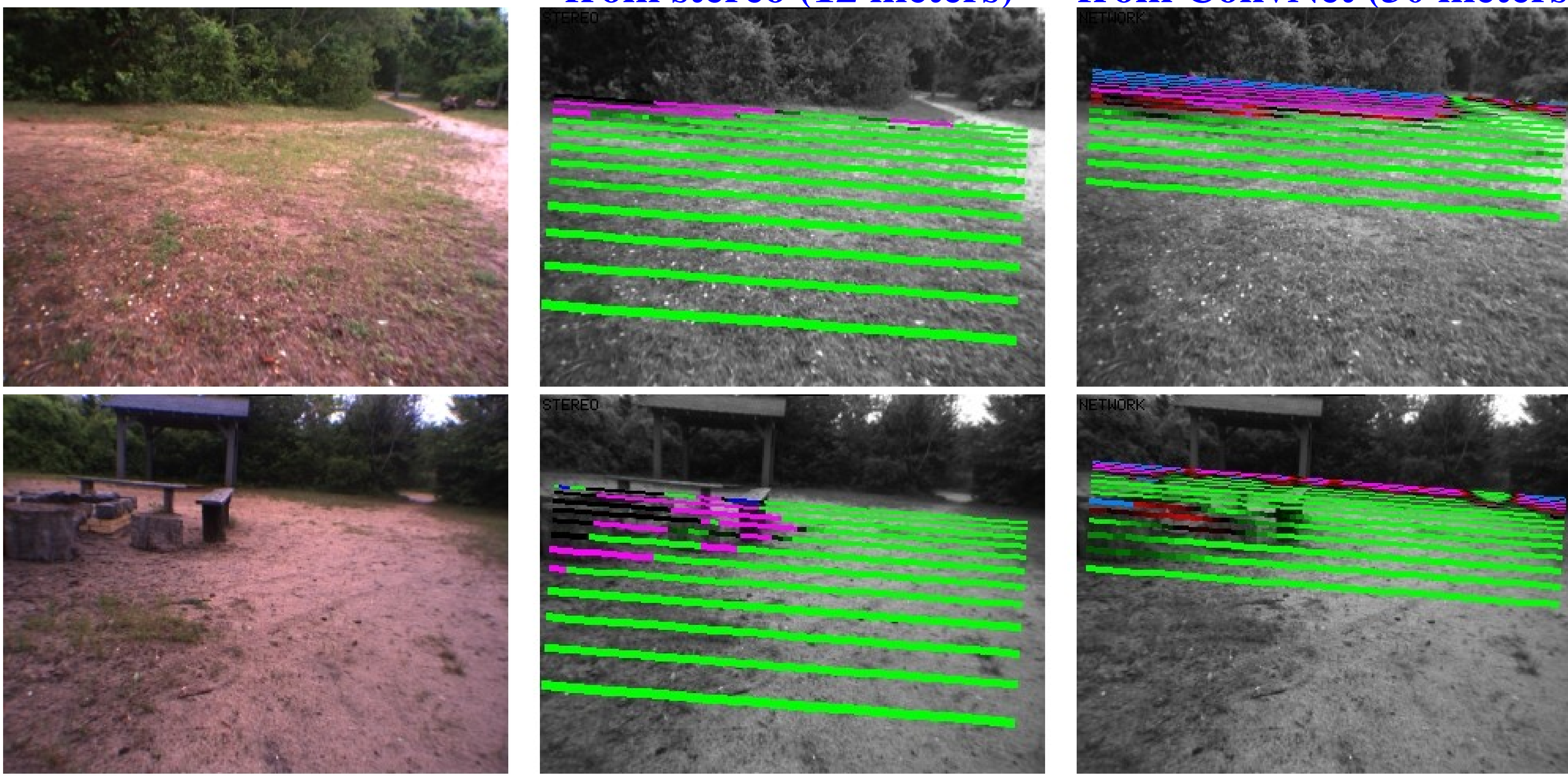
Raw image | **Traversability labels from stereo (12 meters)** | **Traversability labels from ConvNet (30 meters)**



Training a ConvNet On-line to detect obstacles

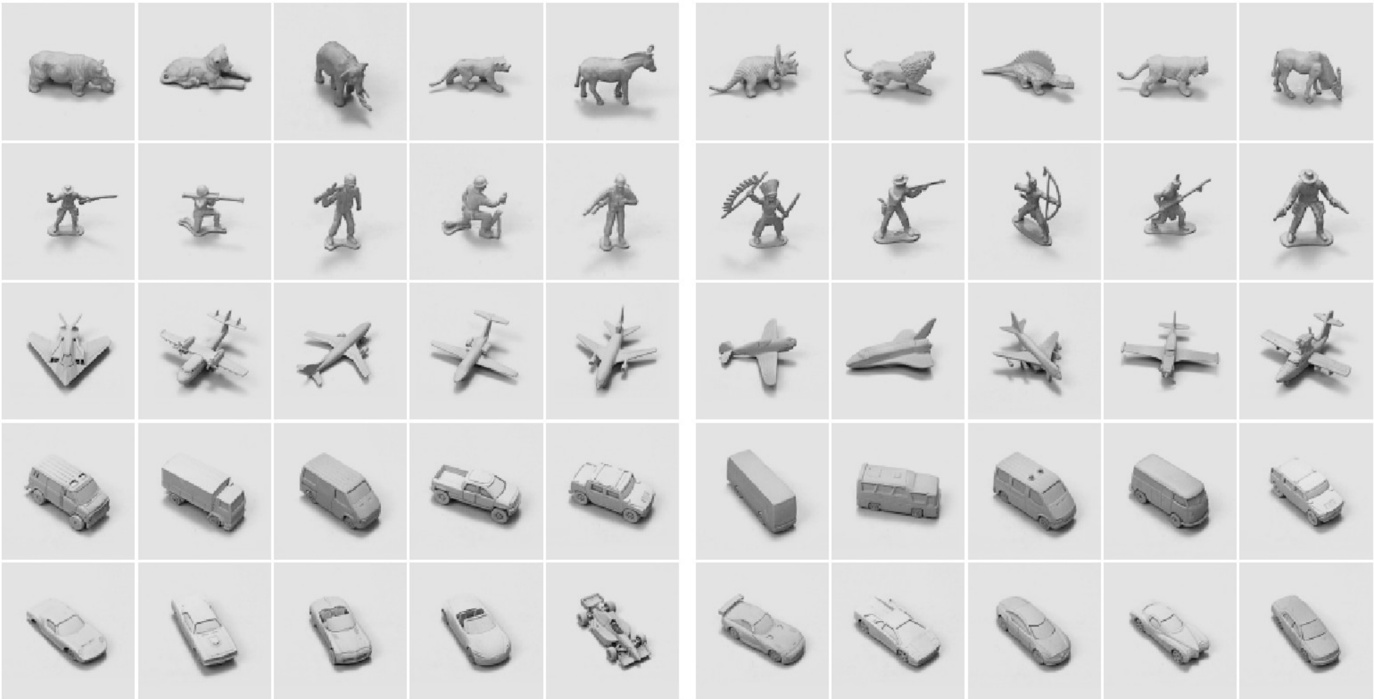
[Hadsell et al. Robotics Science and Systems 2007]

Raw image | **Traversability labels from stereo (12 meters)** | **Traversability labels from ConvNet (30 meters)**



Generic Object Detection and Recognition with Invariance to Pose and Illumination

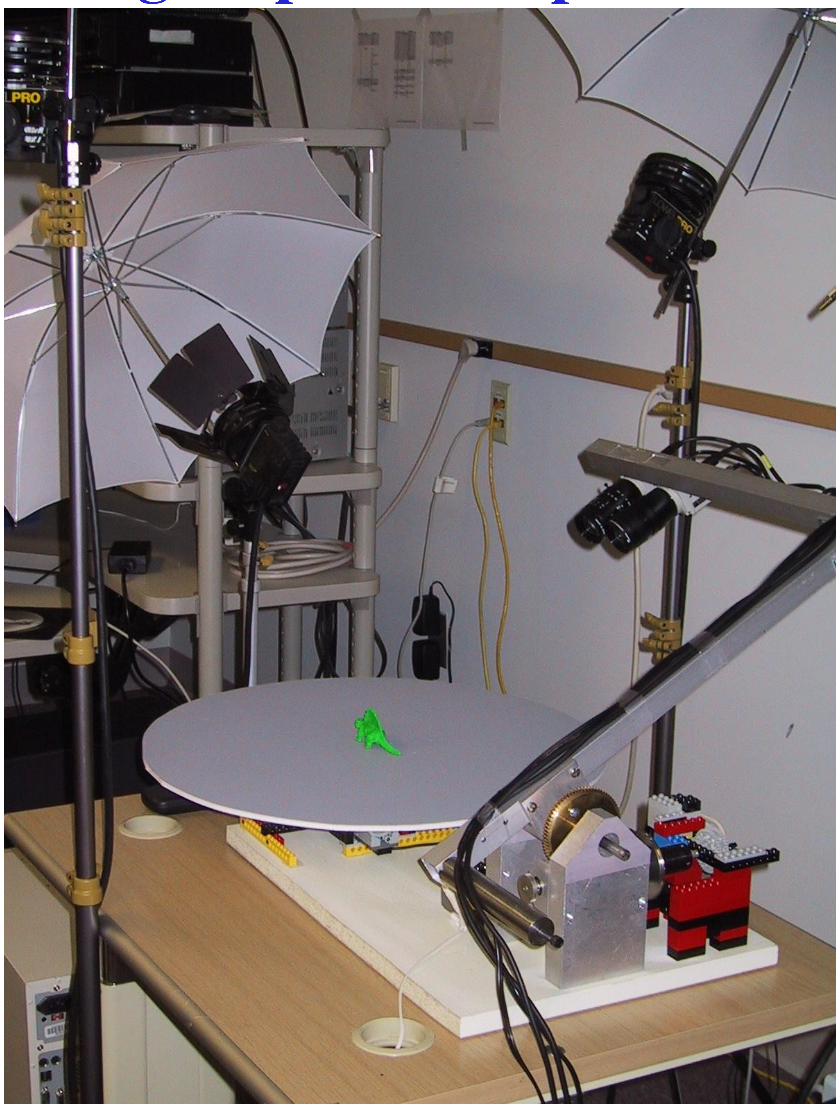
- **50** toys belonging to 5 categories: **animal, human figure, airplane, truck, car**
- **10** instance per category: 5 instances used for training, 5 instances for testing
- **Raw dataset: 972** stereo pair of each object instance. **48,600** image pairs total.
- **For each instance:**
- **18 azimuths**
  - 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**

Data Collection, Sample Generation

Image capture setup

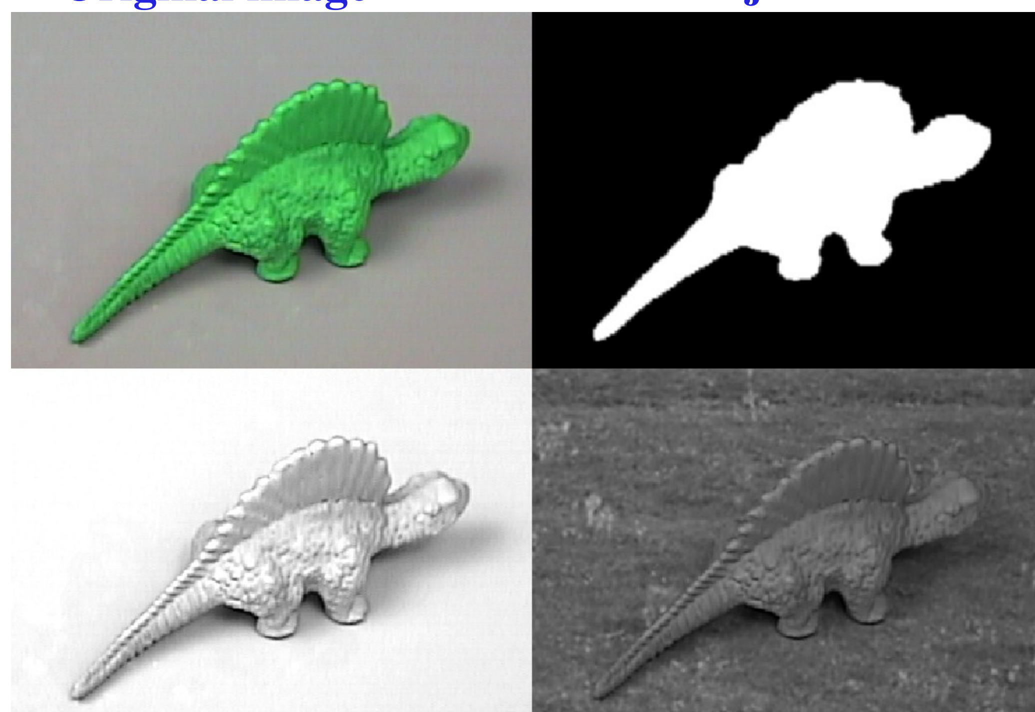


Objects are painted green so that:

- all features other than shape are removed

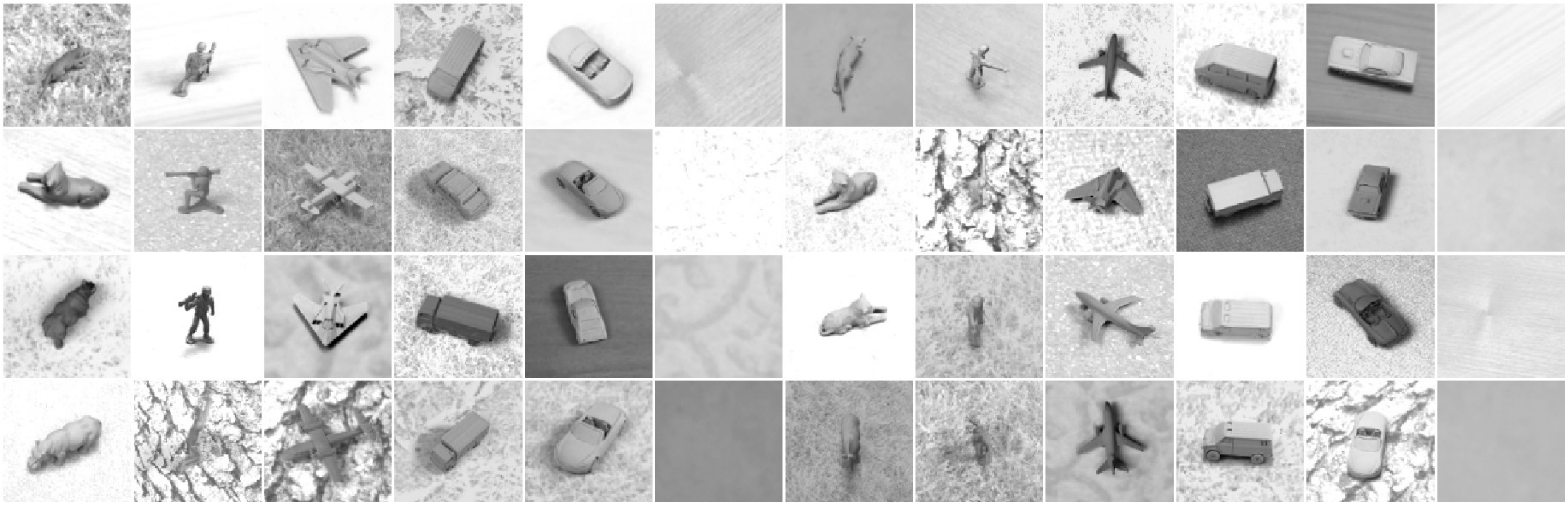
- objects can be segmented, transformed,
  and composited onto various backgrounds

Original image **Object mask**



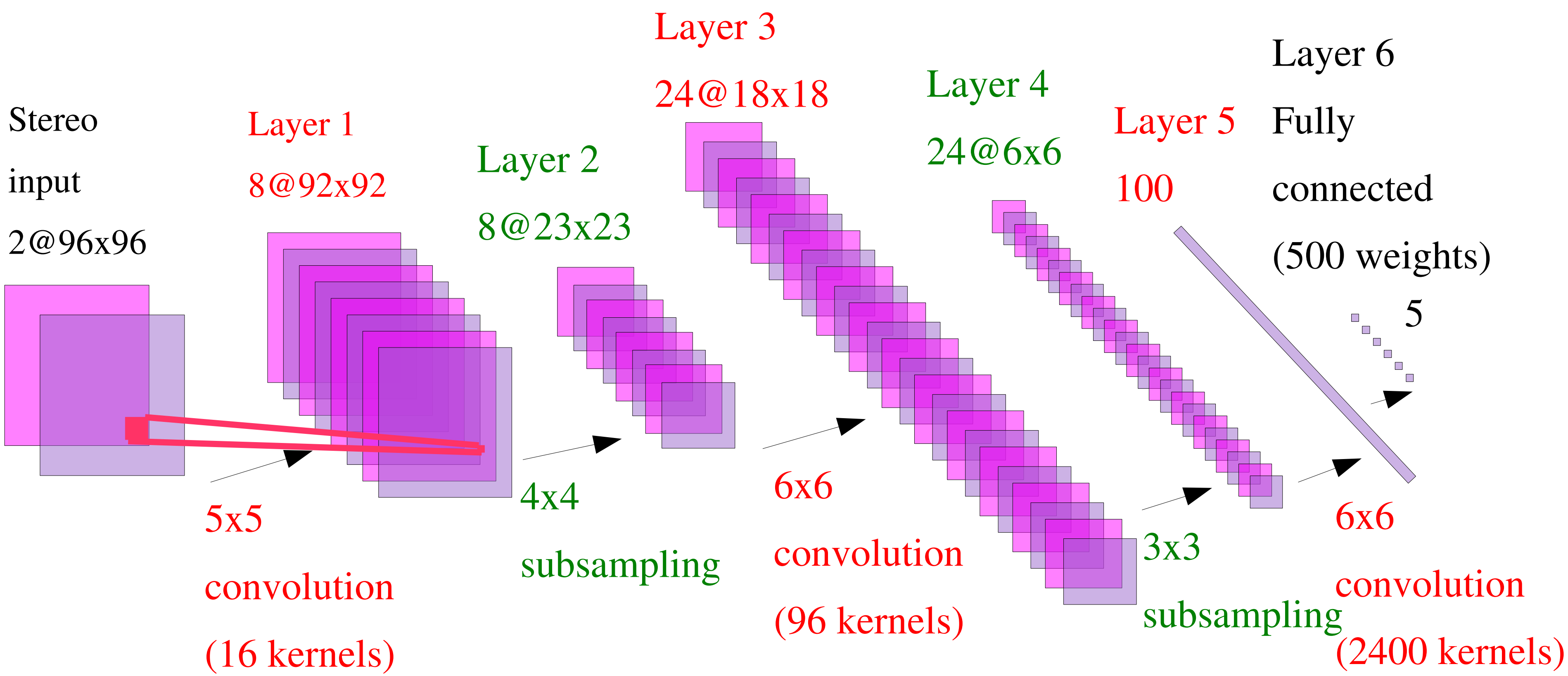
Shadow factor **Composite image**

Textured and Cluttered Datasets



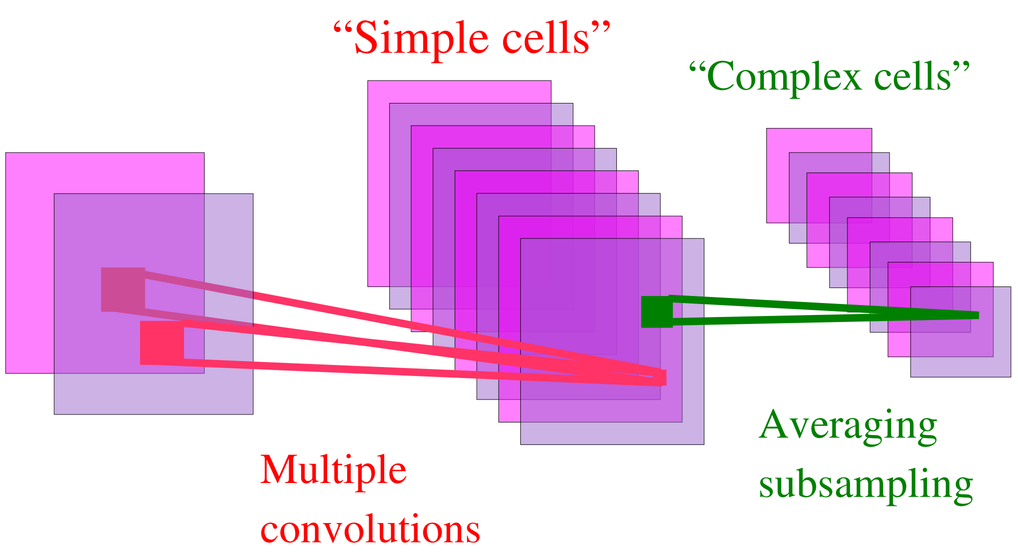
Yann LeCun

Convolutional Network

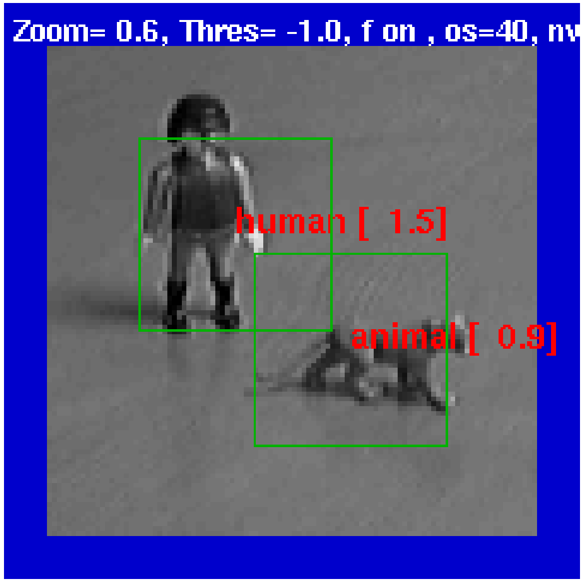


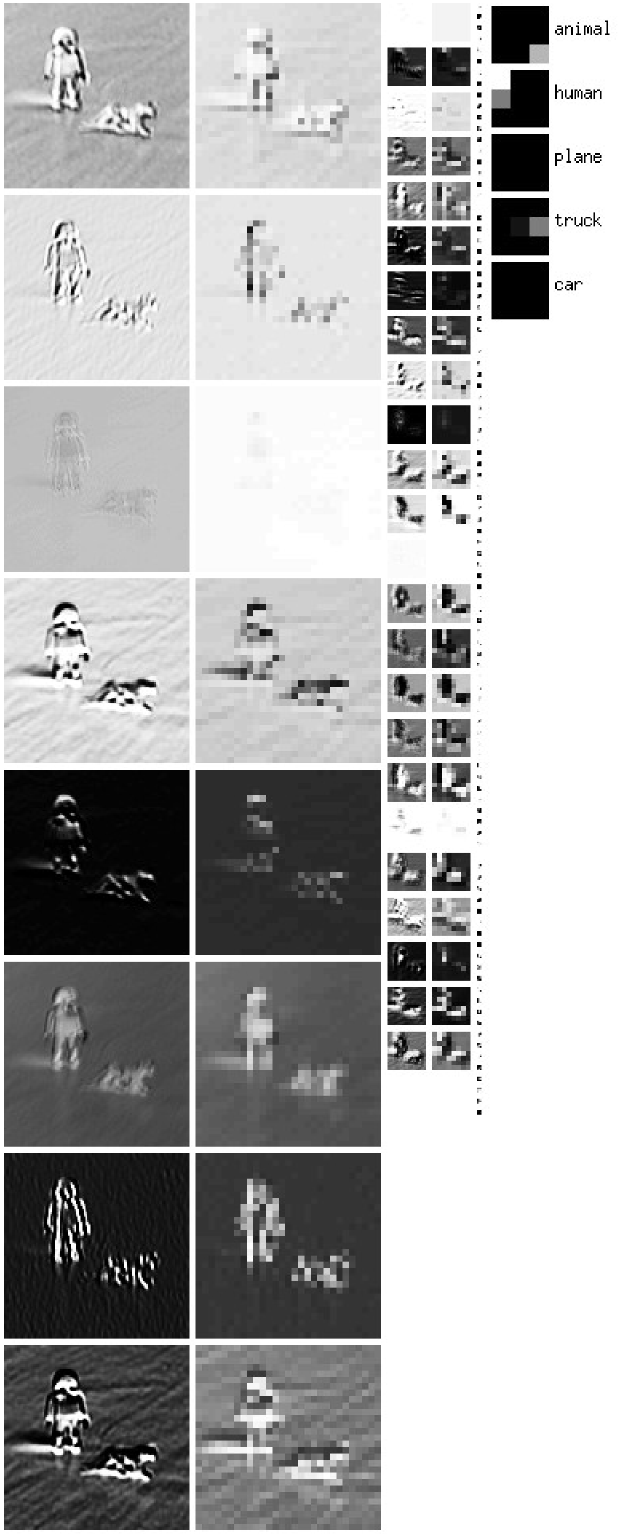
- **90,857 free parameters, 3,901,162 connections.**
- The architecture alternates convolutional layers (feature detectors) and subsampling layers (local feature pooling for invariance to small distortions).
- **The entire network is trained end-to-end** (all the layers are trained simultaneously).
-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

Alternated Convolutions and Subsampling



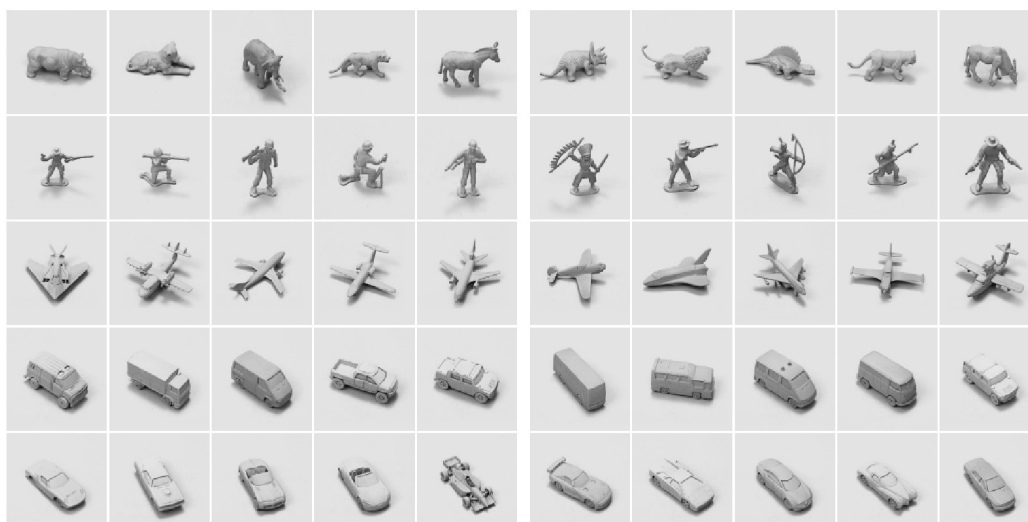
- Local features are extracted everywhere.
- averaging/subsampling layer builds robustness to variations in feature locations.
- Hubel/Wiesel'62, Fukushima'71, LeCun'89, Riesenhuber & Poggio'02, Ullman'02,....





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

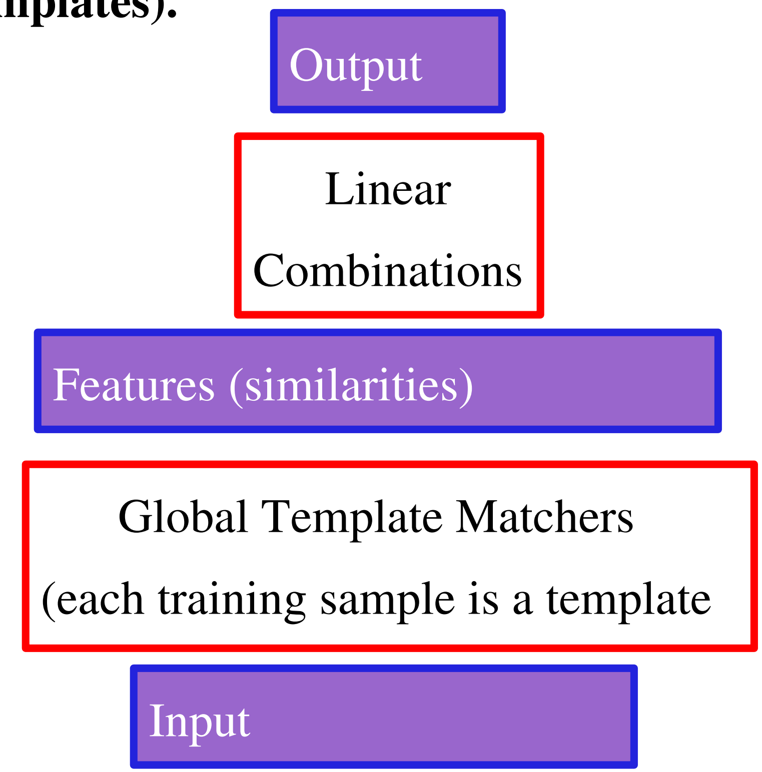
OUCH!

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!

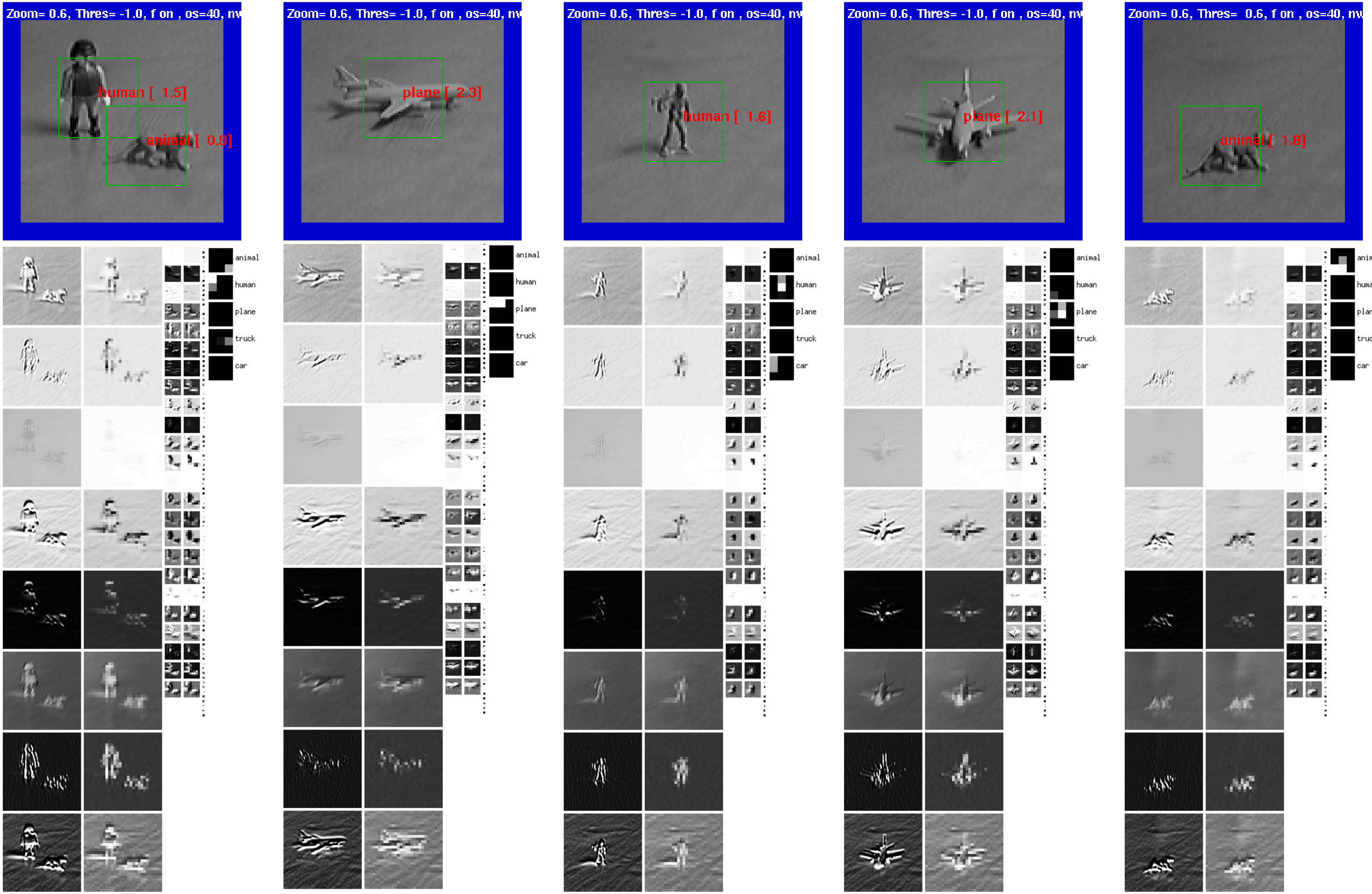
What's wrong with SVMs? they are shallow!

- **SVM with Gaussian kernels is based on matching global templates**
- **It is a "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, ...**

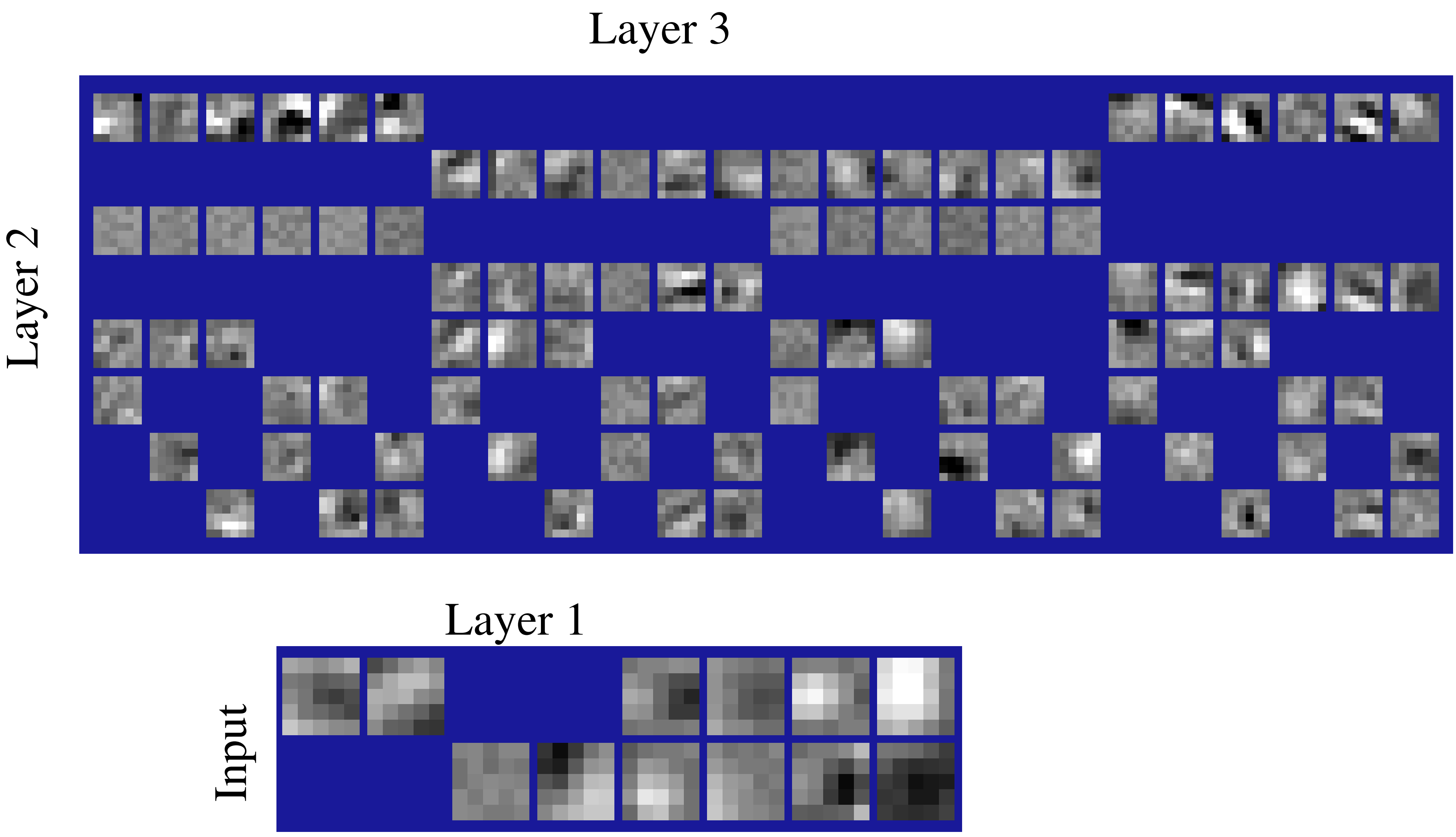


SVM is glorified template matching

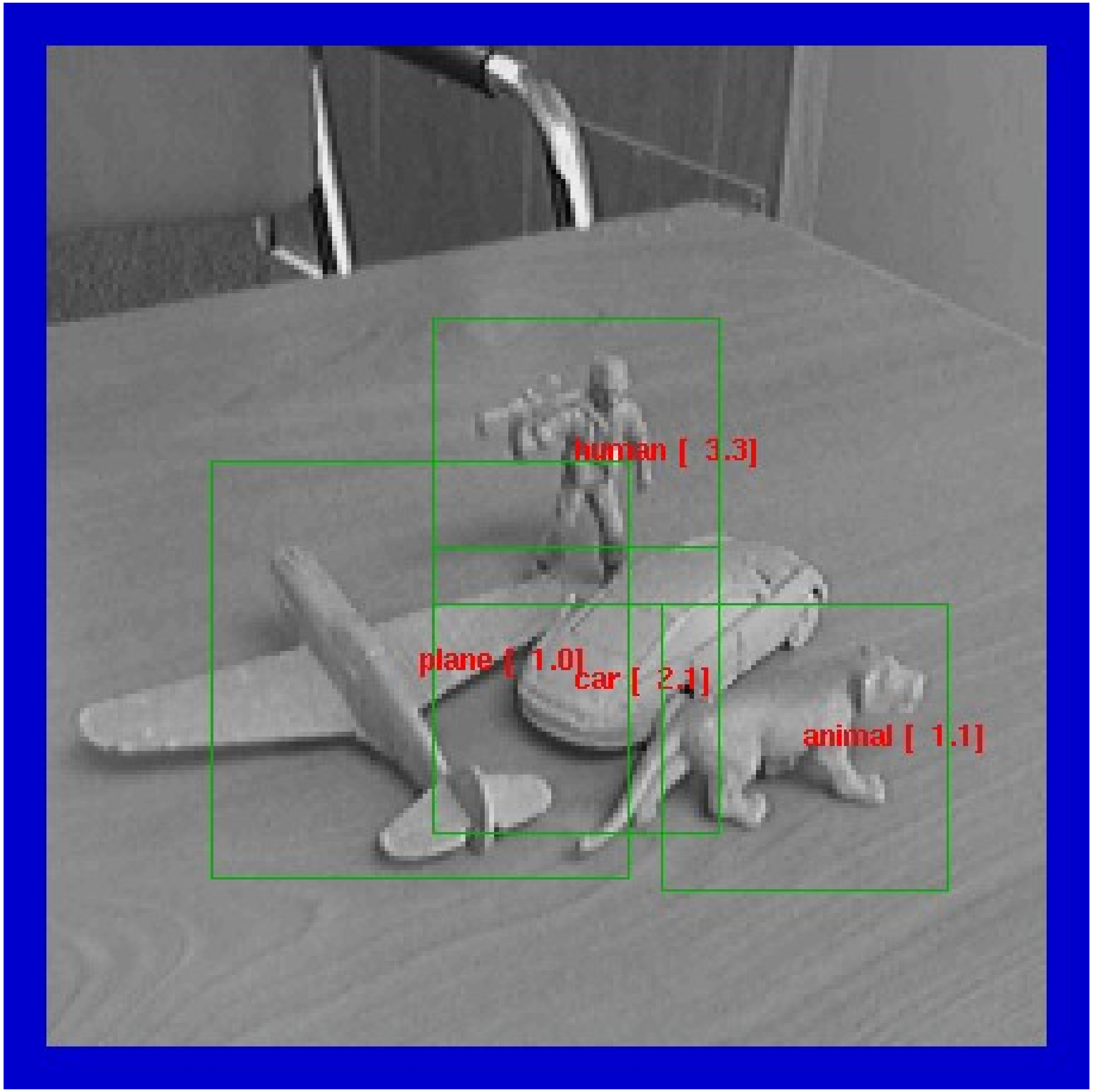
Examples (Monocular Mode)



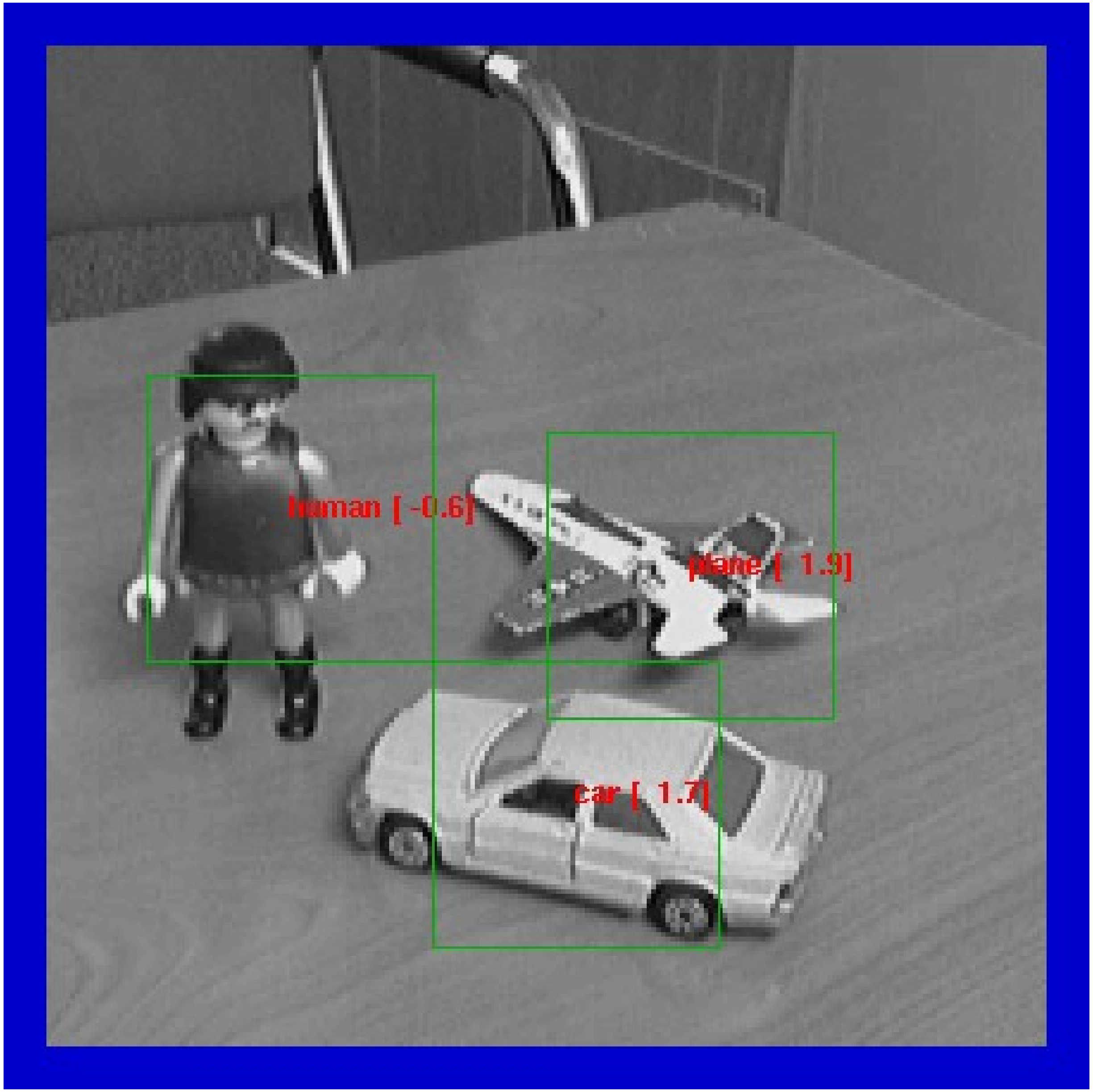
Learned Features



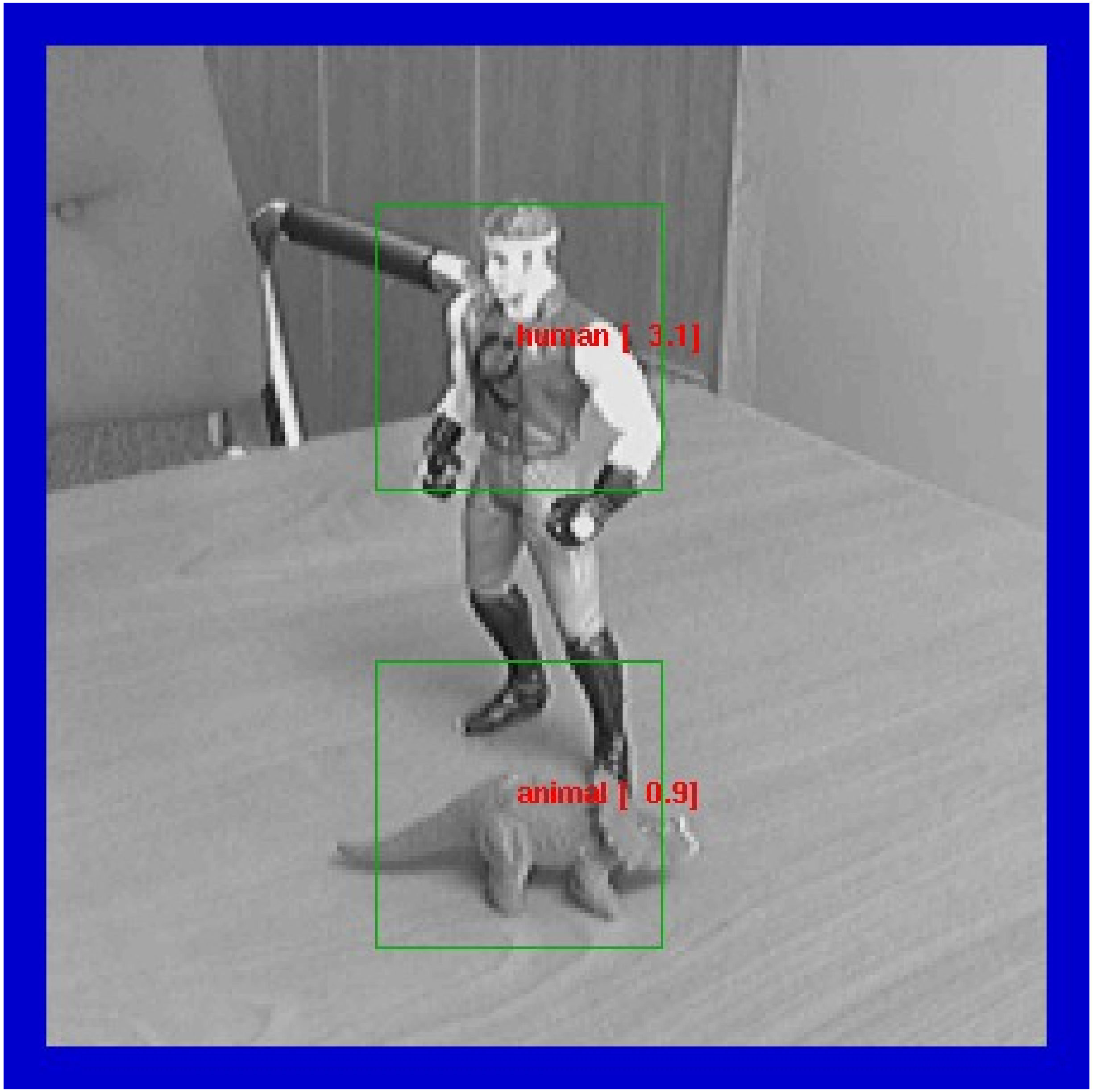
Examples (Monocular Mode)



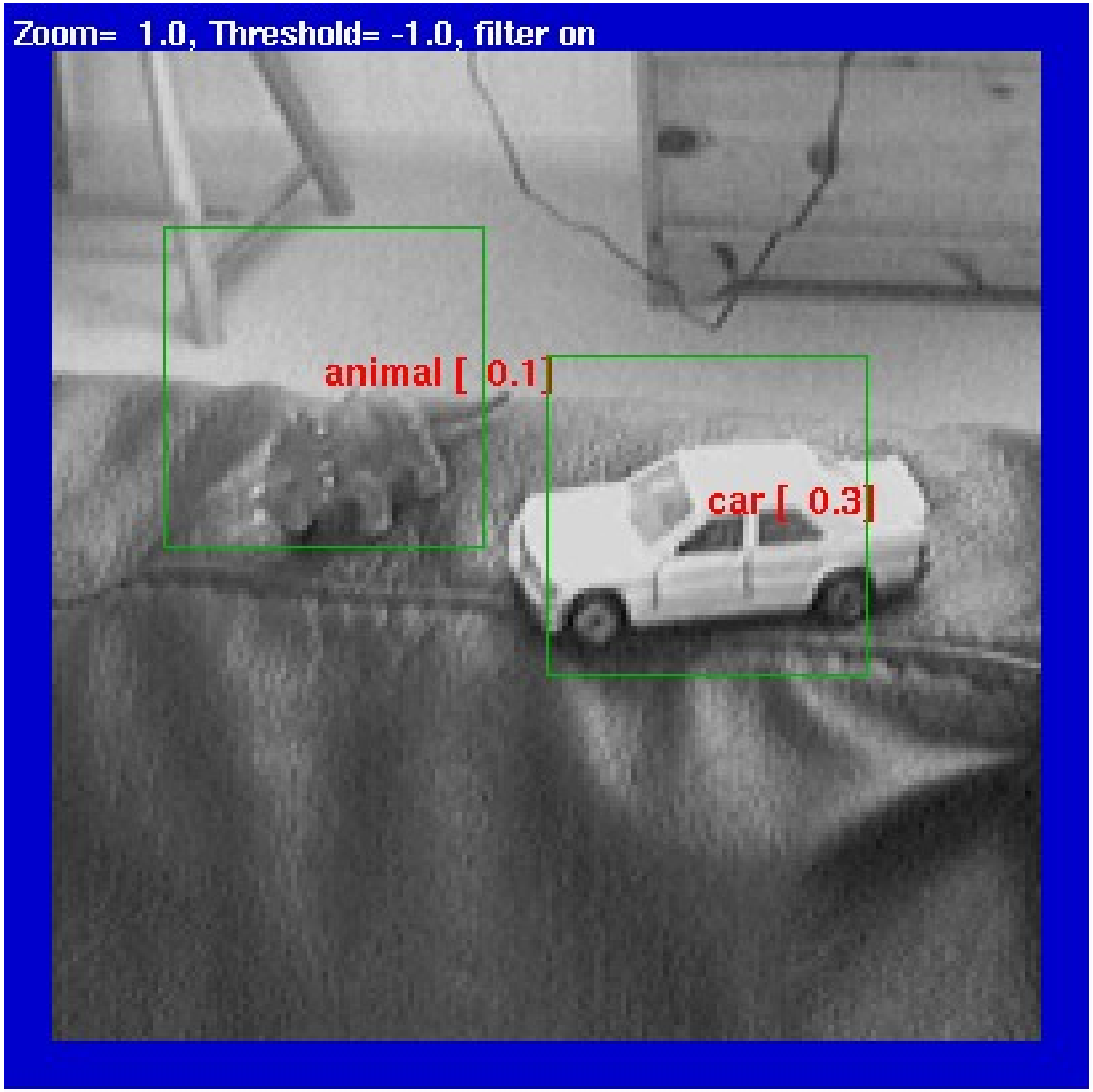
Examples (Monocular Mode)



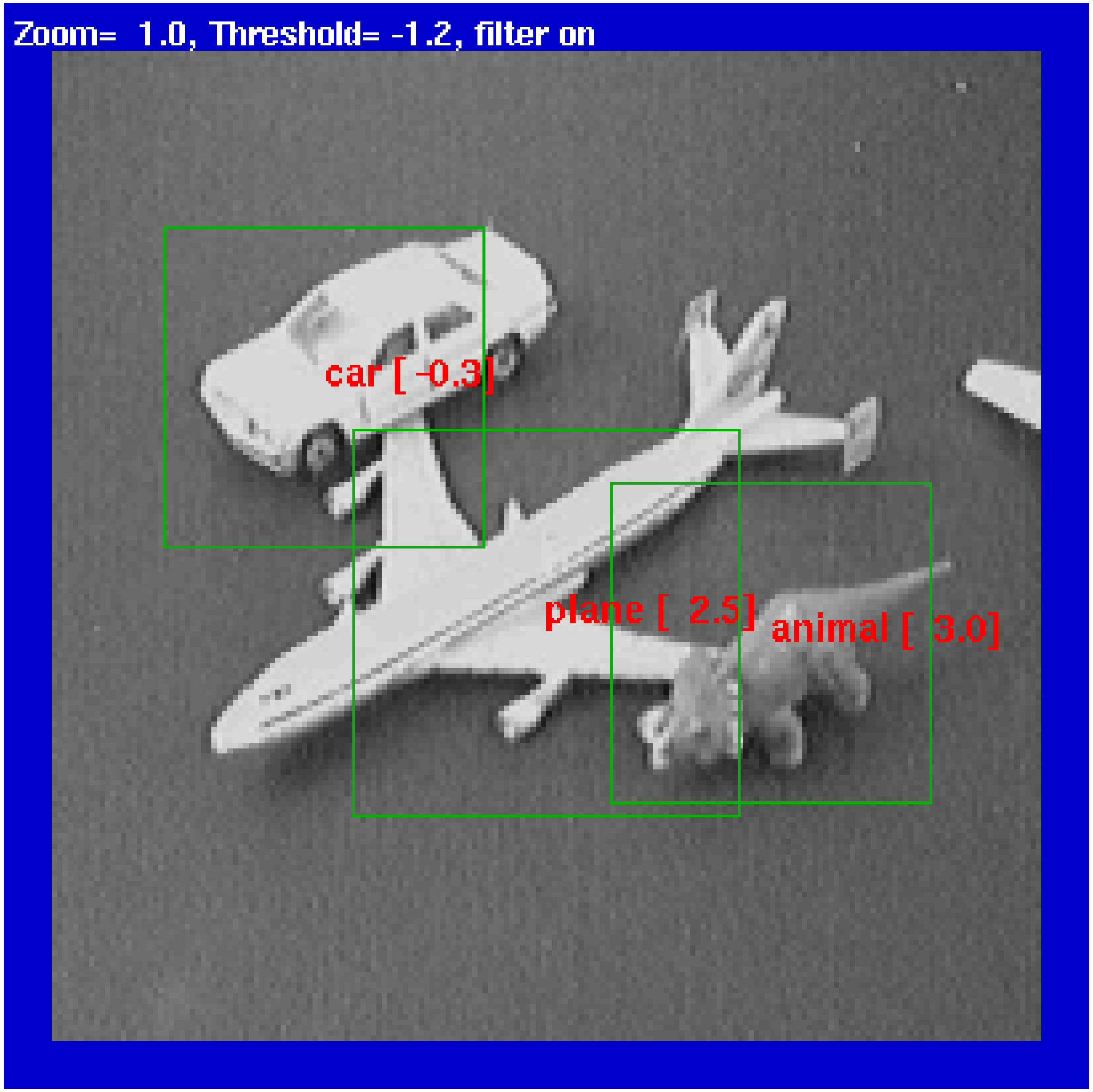
Examples (Monocular Mode)



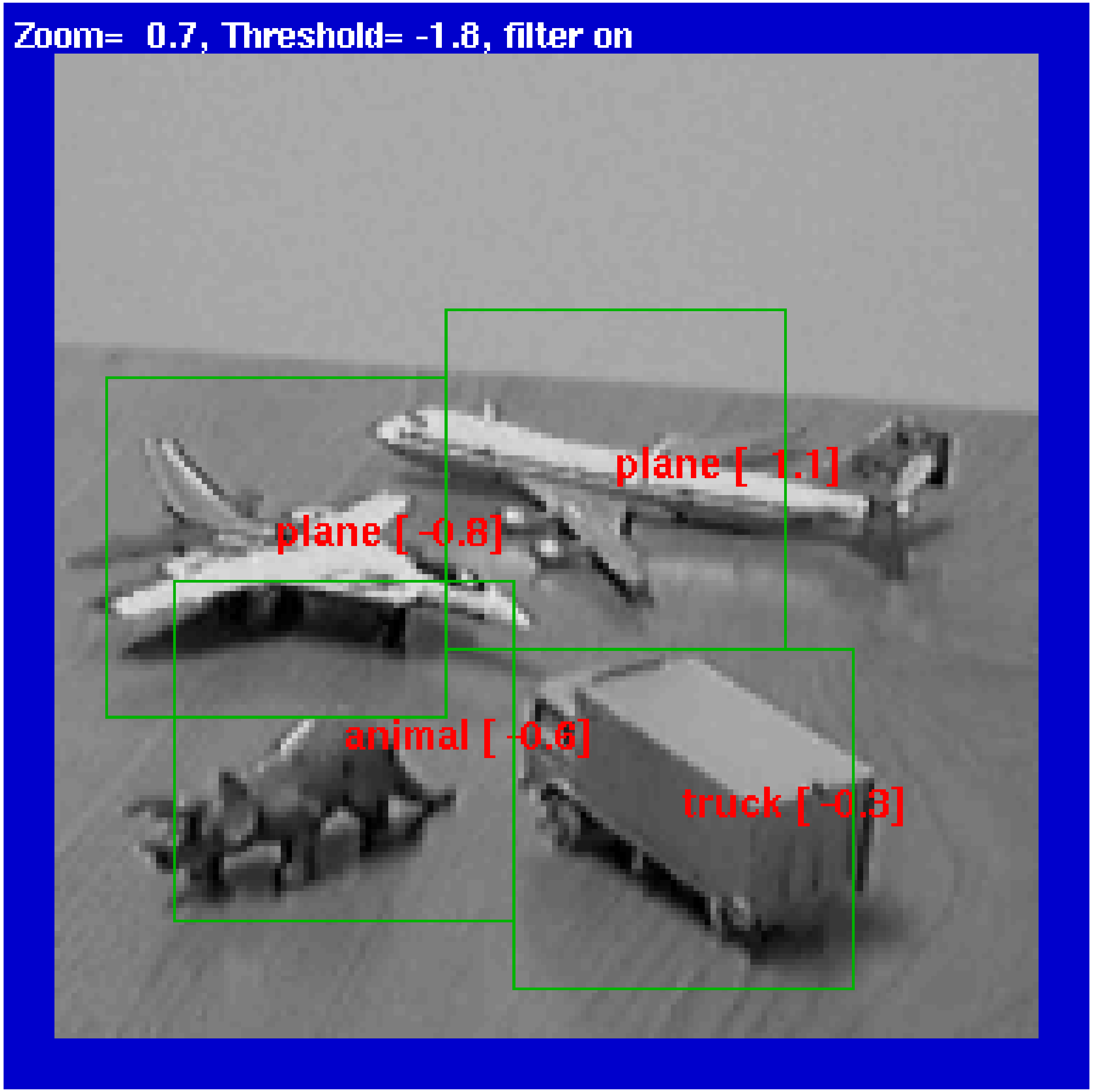
Examples (Monocular Mode)



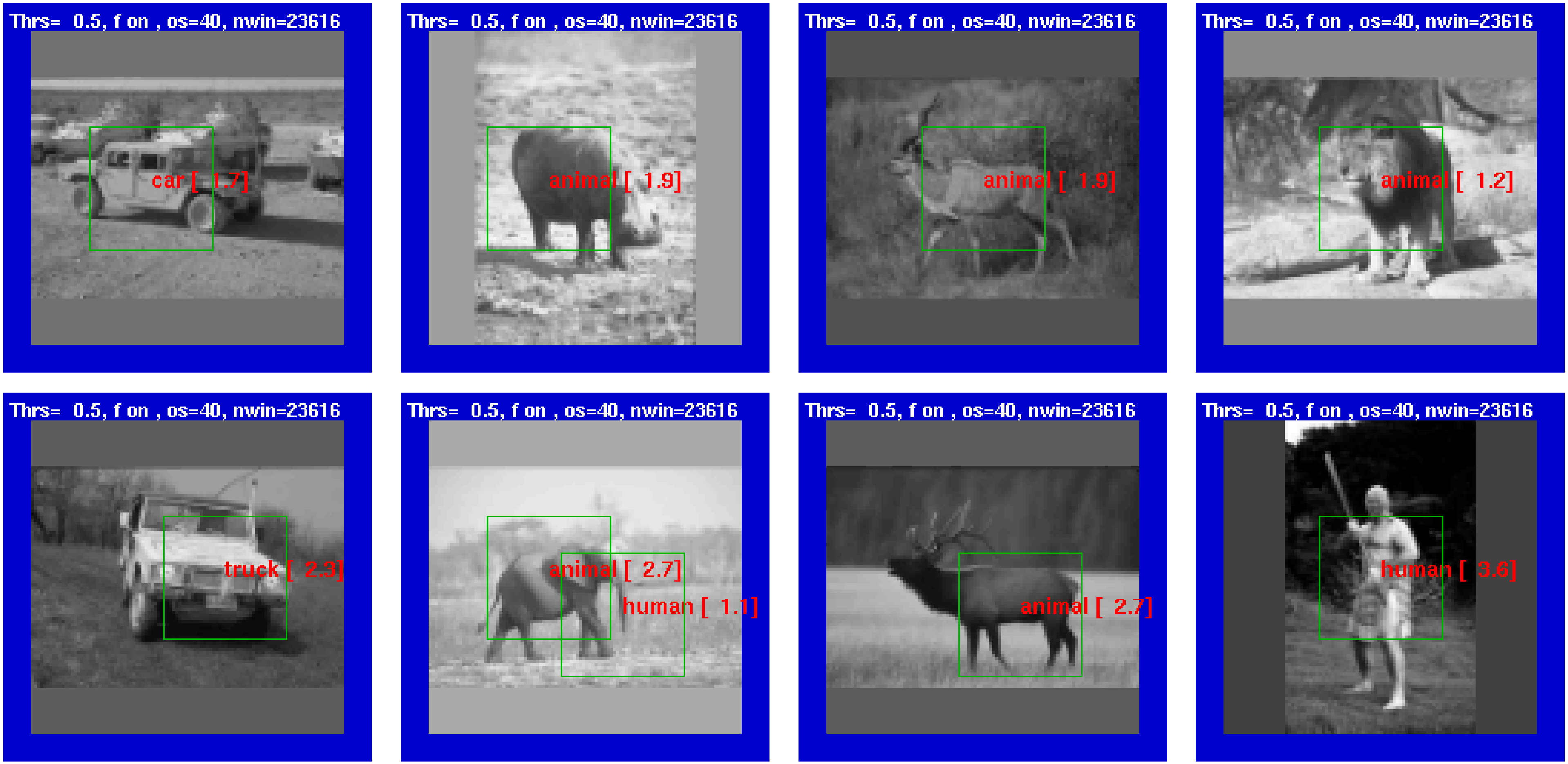
Examples (Monocular Mode)



Examples (Monocular Mode)



Natural Images (Monocular Mode)



Commercially Deployed applications of Convolutional Nets

Faxed form reader

- Developed at AT&T Bell Labs in the early 90's
- Commercially deployed in 1994

Check Reading system:

- Developed at AT&T Bell Labs in the mid 90's
- Commercially deployed by NCR in 1996
- First practical system for reading handwritten checks
- Read 10 to 20% of all the checks in the US in the late 90's

Face detector / Person detector / Intrusion detector

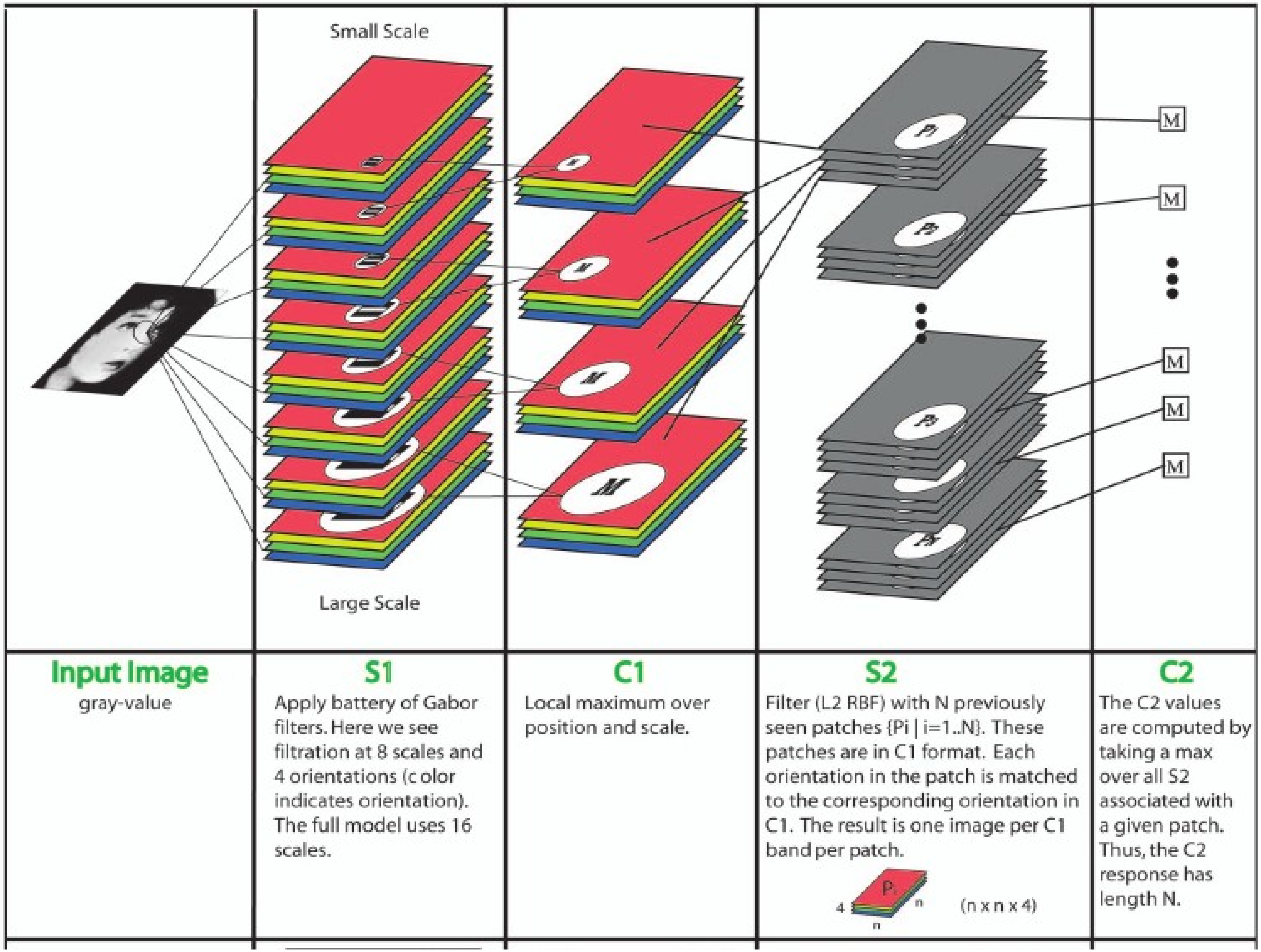
- Developed at NEC Research Institute in 2002/2003
- Commercially deployed in 2004 by Vident Technologies
- Used at San Francisco Airport (among others).

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.**

Models Similar to ConvNets

- **HMAX**
  - [Poggio & Riesenhuber 2003]
  - [Serre et al. 2007]
  - [Mutch and Low CVPR 2006]
- **Difference?**
  - the features are not learned
- **HMAX is very similar to Fukushima's Neocognitron**



[from Serre et al. 2007]

Part 3:
Unsupervised Training of "Deep" Energy-Based Models, Learning Invariant Feature Hierarchies

- **Why do we need Deep Learning?**
  - "scaling learning algorithms towards AI" [Bengio and LeCun 2007]
- **Deep Belief Networks, Deep Learning**
  - Stacked RBM [Hinton, Osindero, and Teh, Neural Comp 2006]
  - Stacked autoencoders [Bengio et al. NIPS 2006]
  - Stacked sparse features [Ranzato & al., NIPS 2006]
  - Improved stacked RBM [Salakhutdinov & Hinton, AI-Stats 07]
- **Unsupervised Learning of Invariant Feature Hierarchies**
  - learning features for Caltech-101 [Ranzato et al. CVPR 2006]
  - learning features hierarchies for hand-writing [Ranzato et al ICDAR'07]

[See Mar'cAurelio Ranzato's poster on Wednesday]

Why do we need "Deep" Architectures?

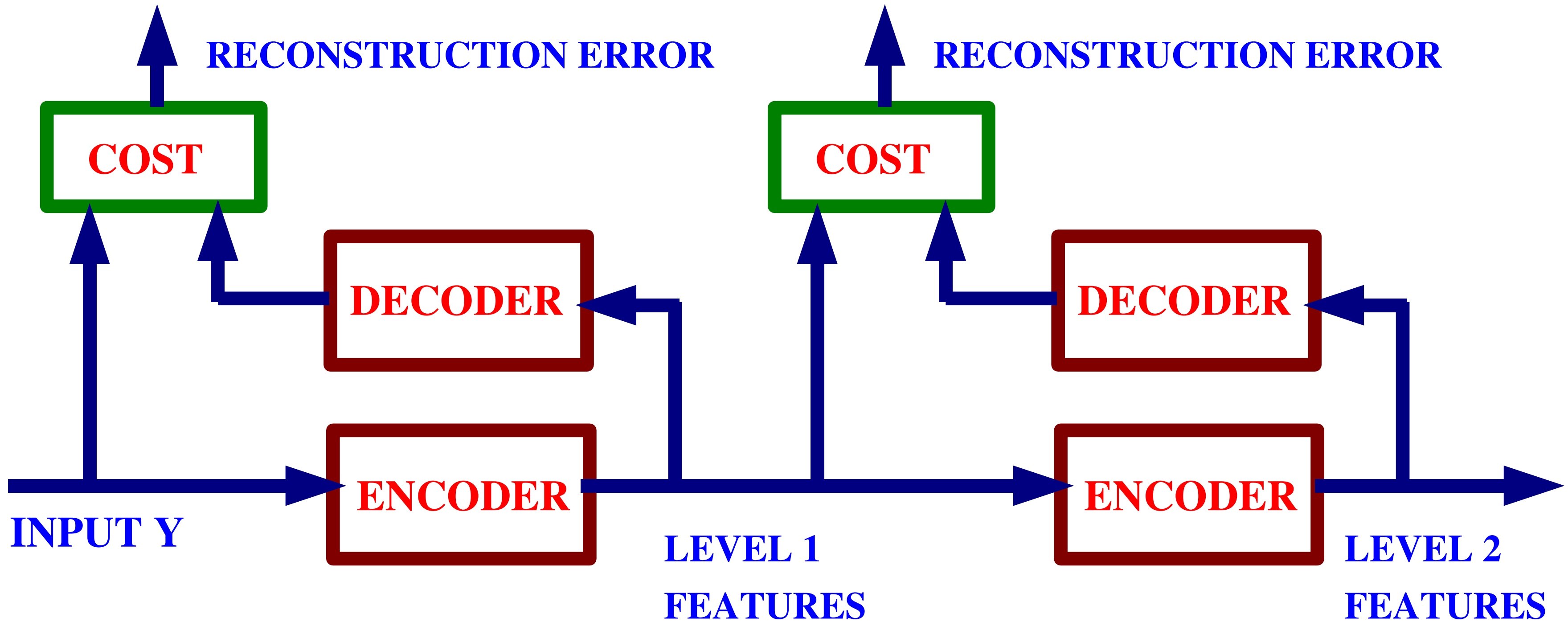
[Bengio & LeCun 2007]

- **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

The Basic Idea of Deep Learning

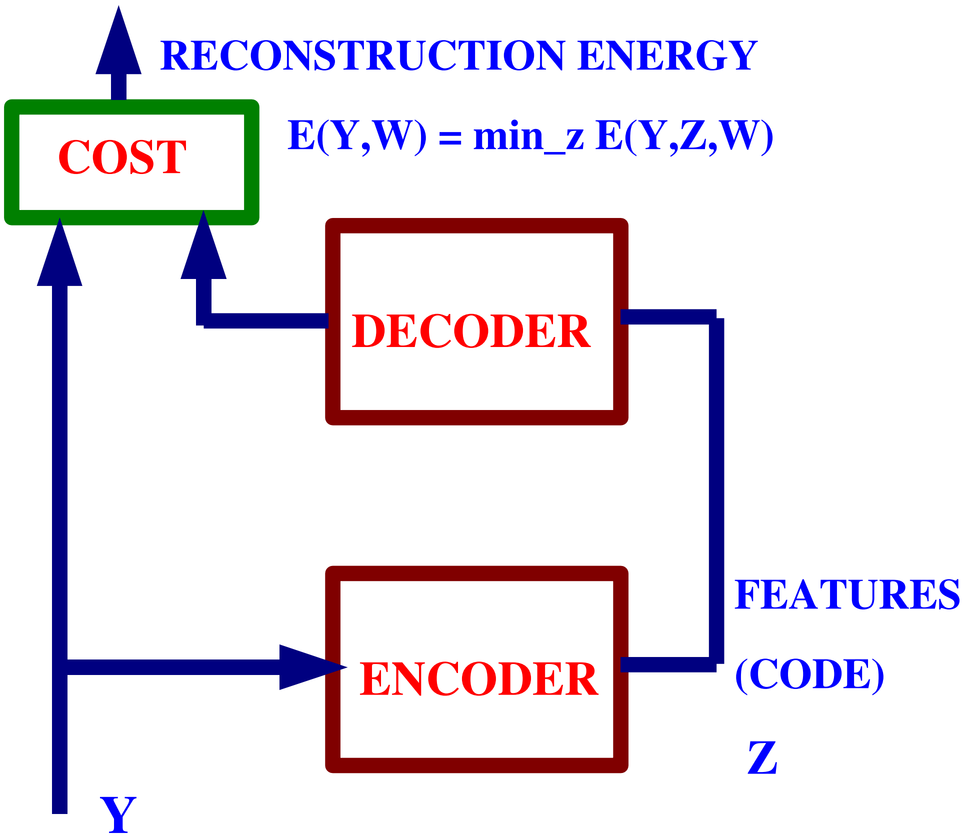
[Hinton et al. 2005 - 2007]

- **Unsupervised Training of Feature Hierarchy [Hinton et al. 2005 – 2007]**
  - Each layer is designed to extract higher-level features from lower-level ones
  - Each layer is trained unsupervised with a reconstruction criterion
  - The layers are trained one after the other, in sequence.



Encoder-Decoder Architecture for Unsupervised Learning

- **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], [Ranzato et al NIPS 06].
  - approximation of data likelihood: Restricted Boltzmann Machine [Hinton 2005-...]



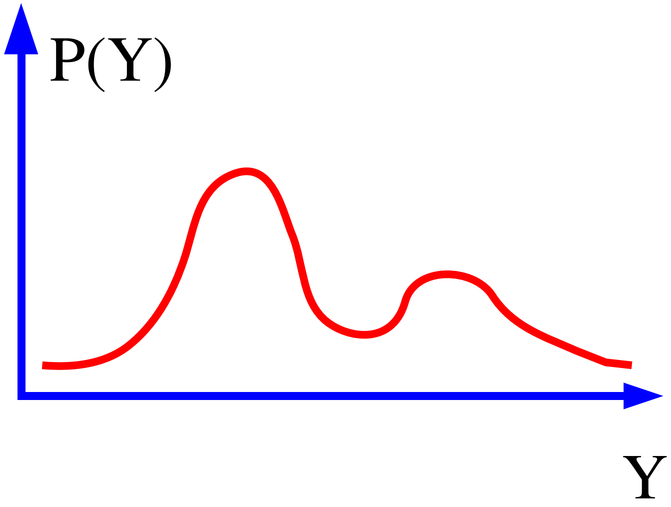
$$E(Y,W) = min_Z E(Y,Z,W)$$

$$\bar{Z}_Y = argmin_Z E(Y,Z,W)$$

What is Energy-Based Unsupervised Learning?

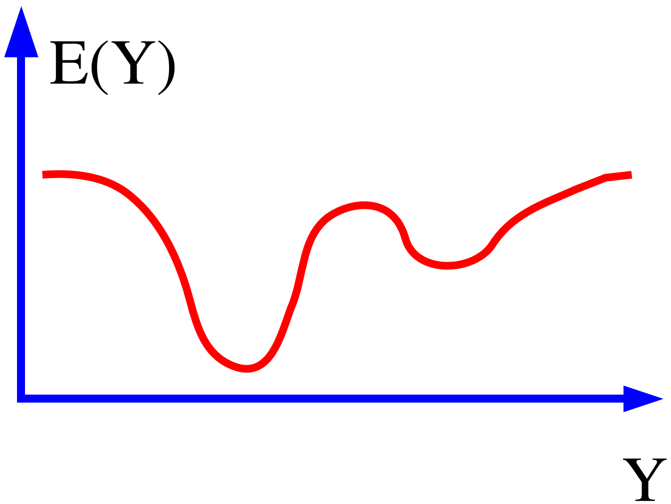
Probabilistic View:

- Produce a probability density function that:
- has high value in regions of high sample density
- has low value everywhere else (integral=1)
- **Training:** maximize the data likelihood **(intractable)**
- $$P(Y,W) = \frac{e^{-\beta E(Y,W)}}{\int_y e^{-\beta E(y,W)}}$$



Energy-Based View:

- produce an energy function E(Y) that:
- has low value in regions of high sample density
- has high(er) value everywhere else



Unsupervised Training of Energy-Based Models

- **Basic Idea:**
  - push down on the energy of training samples
  - pull up on the energy of everything else
  - **but this is often intractable**
- **Approximation #1: Contrastive Divergence [Hinton et al 2005]**
  - Push down on the energy of the training samples
  - Pull up on the energies of configuration that have low energy **near the training samples** (to create local minima of the energy surface)
- **Approximation #2: Minimizing the information content of the code [Ranzato et al. AI-Stats 2007]**
  - Reduce the information content of the code **by making it sparse**
  - This has the effect of increasing the reconstruction error for non-training samples.

Deep Learning for Non-Linear Dimensionality Reduction

- **Restricted Boltzmann Machine.**
  - simple energy function

$$E(Y,Z,W)=\sum_{ij} -Y_i W_{ij} Z_j$$

  - code units are binary stochastic
  - training with contrastive divergence

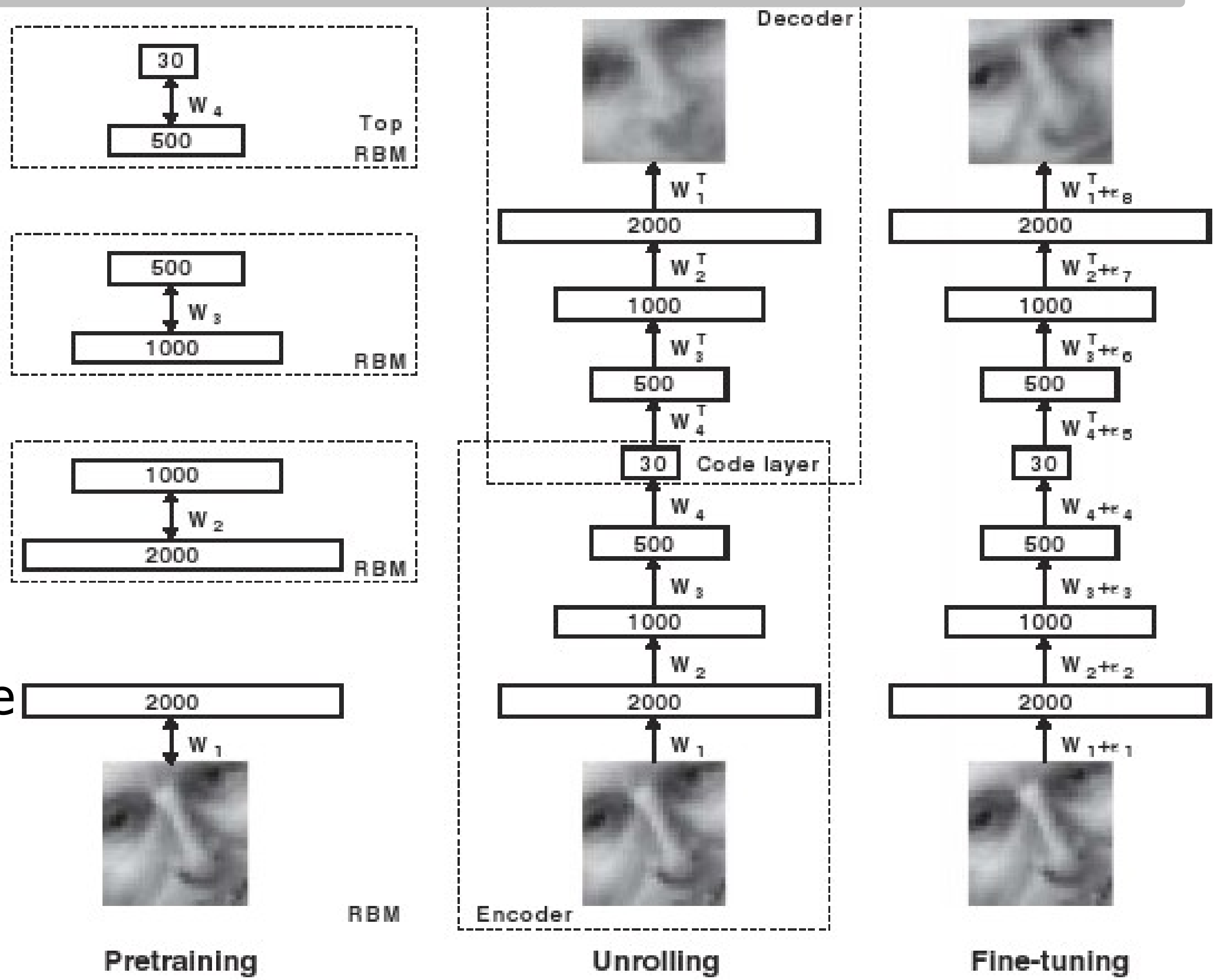
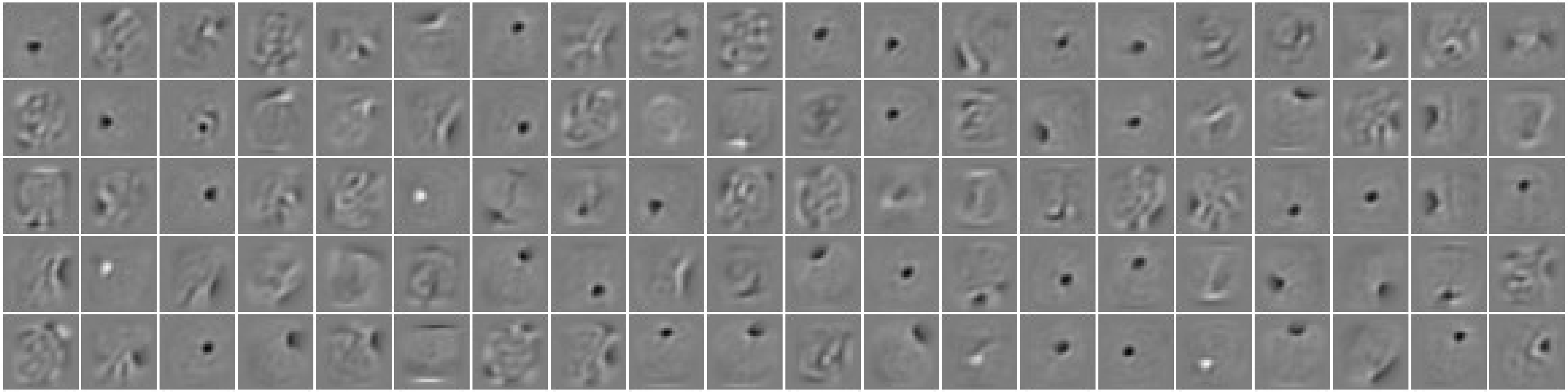


Fig. 1. Pretraining consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack. After the pretraining, the RBMs are "unrolled" to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.

From: [Hinton and Salakhutdinov, Science 2006]

RBM: filters trained on MNIST

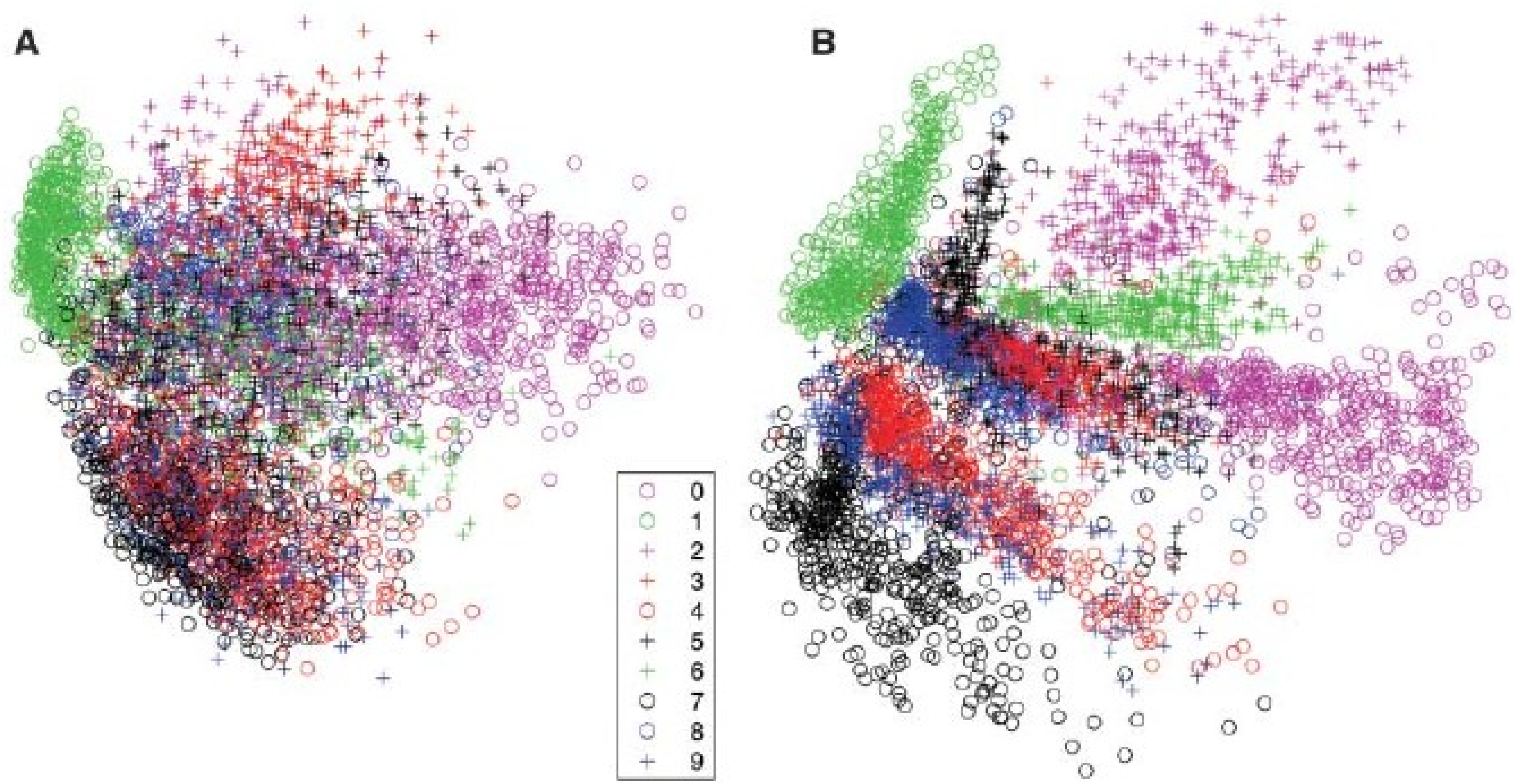
- "bubble" detectors



Non-Linear Dimensionality Reduction: MNIST

- **[Hinton and Salakhutdinov, Science 2006]**

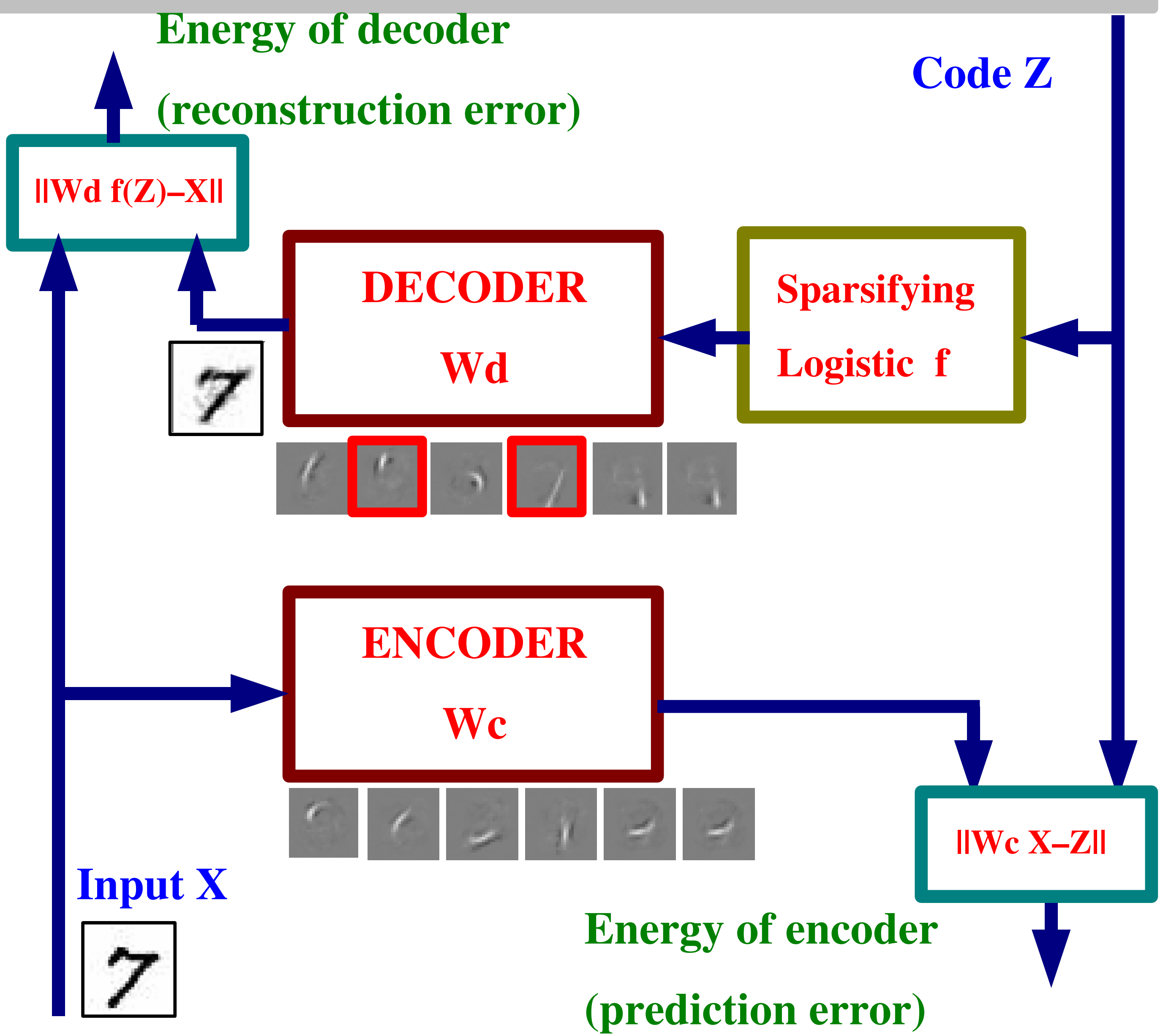
Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



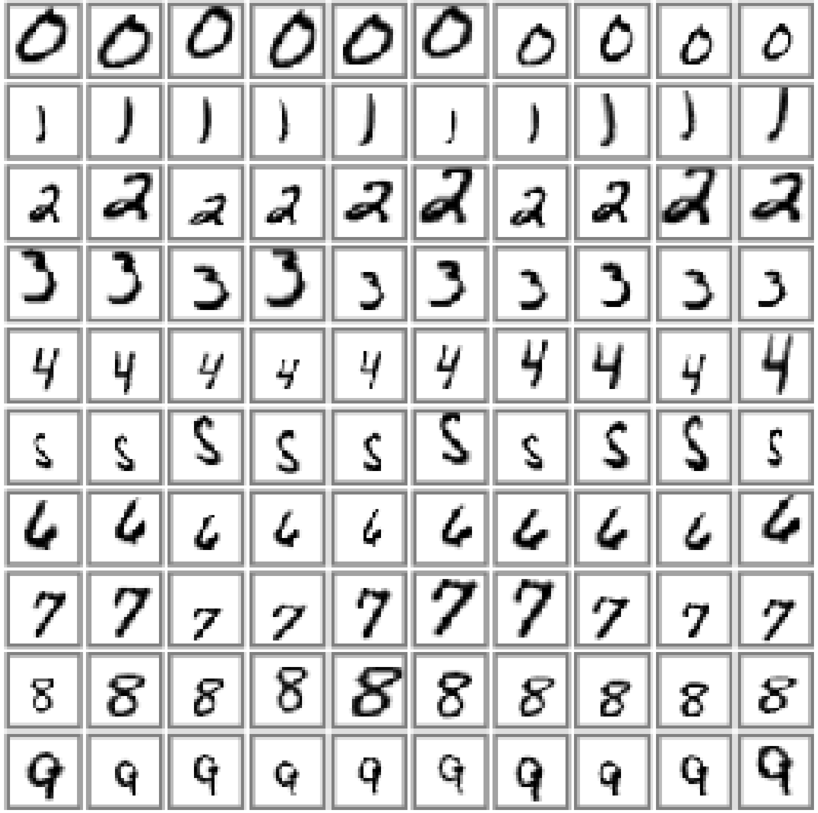
- **[Salakhutdinov and Hinton, AI-Stats 2007]:**
  - < 1.00% error on MNIST using K-NN on 30 dimensions:
  - **BEST ERROR RATE OF ANY KNOWEDGE-FREE METHODS!!!**

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

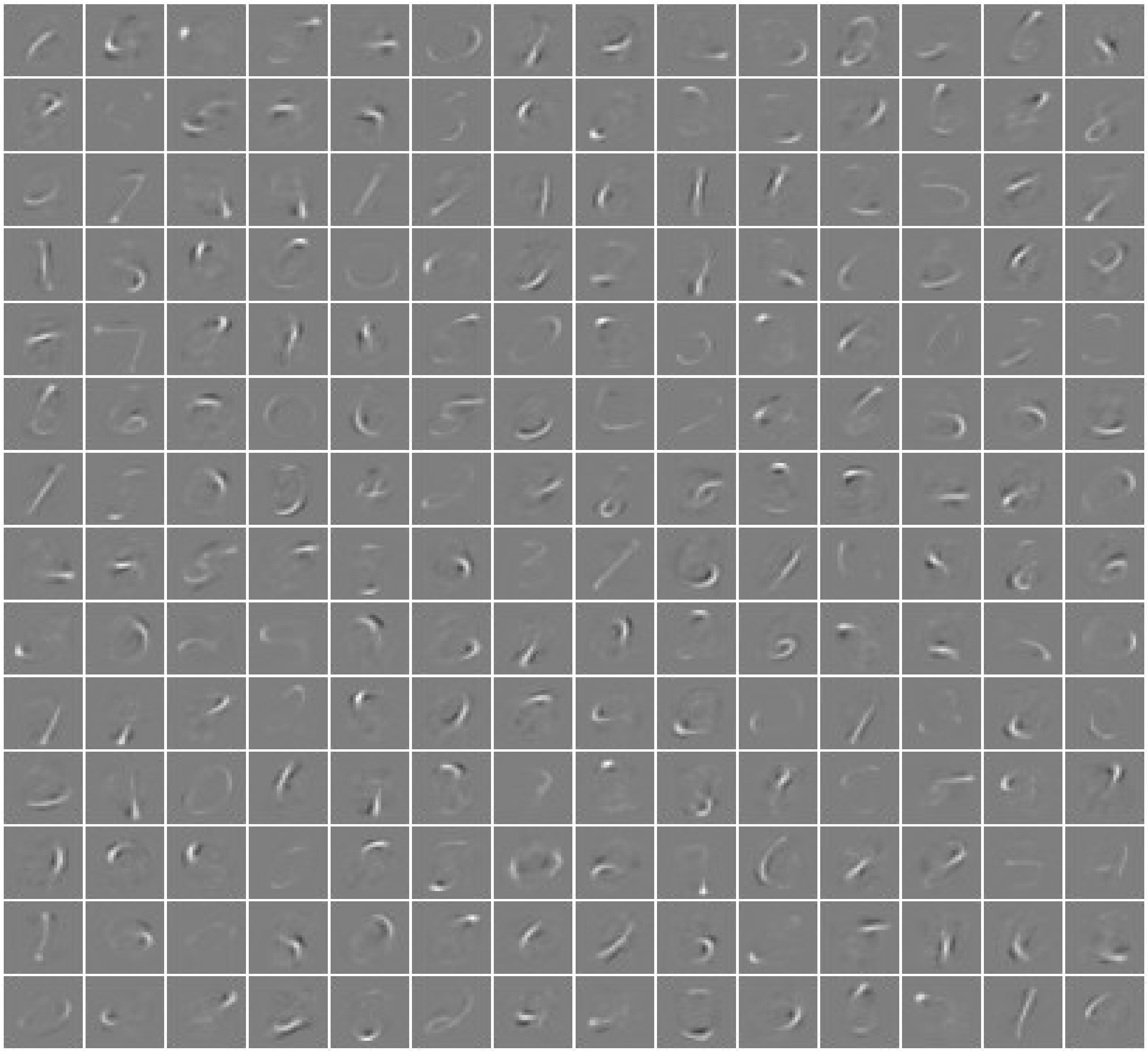


MNIST Dataset



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

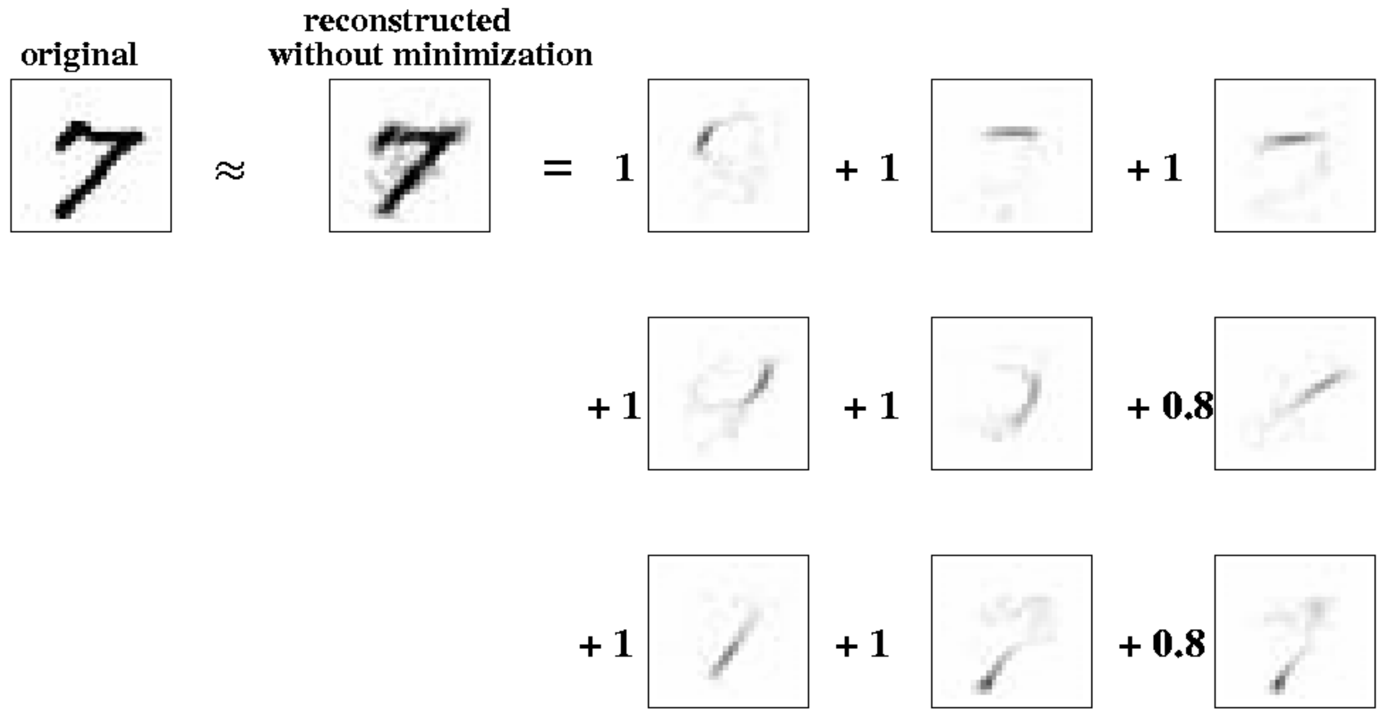
Training on handwritten digits

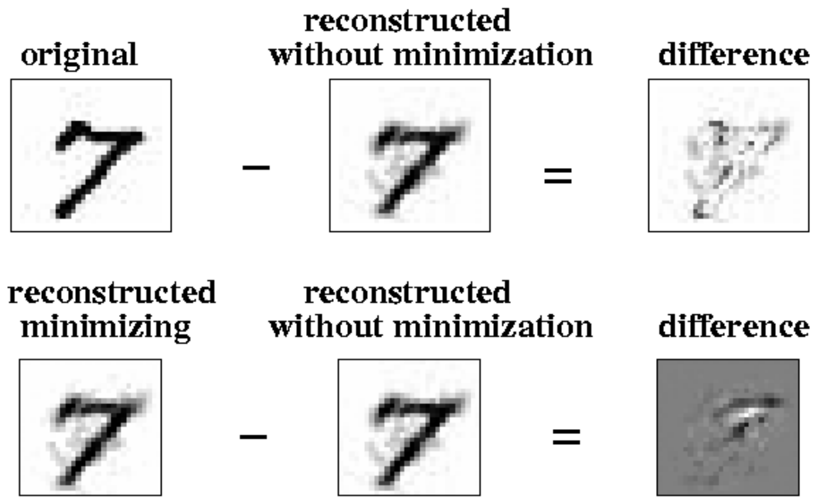


Encoder *direct* filters

- 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

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.

Unsupervised Training of Convolutional Filters

CLASSIFICATION EXPERIMENTS

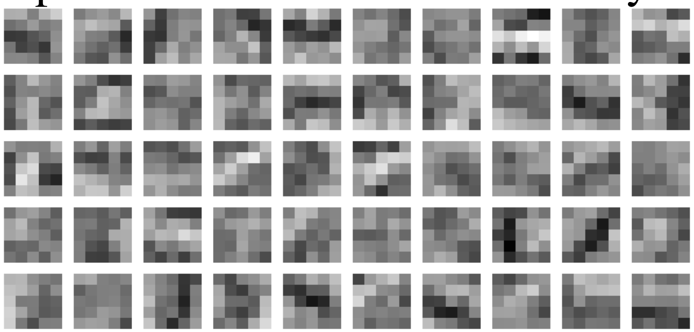
IDEA: improving supervised learning by pre-training
with the unsupervised method (*)

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

- The baseline: *lenet6* initialized randomly

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

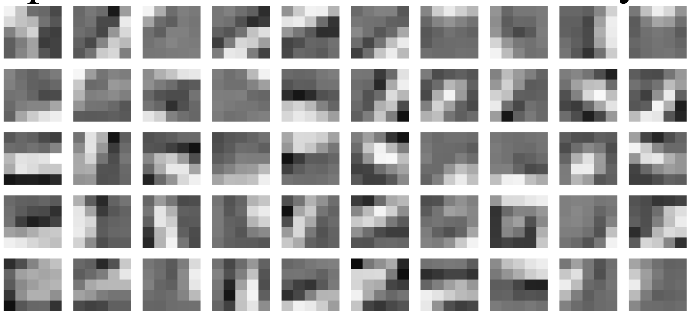
supervised filters in first conv. layer



- *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%.

unsupervised 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%**).

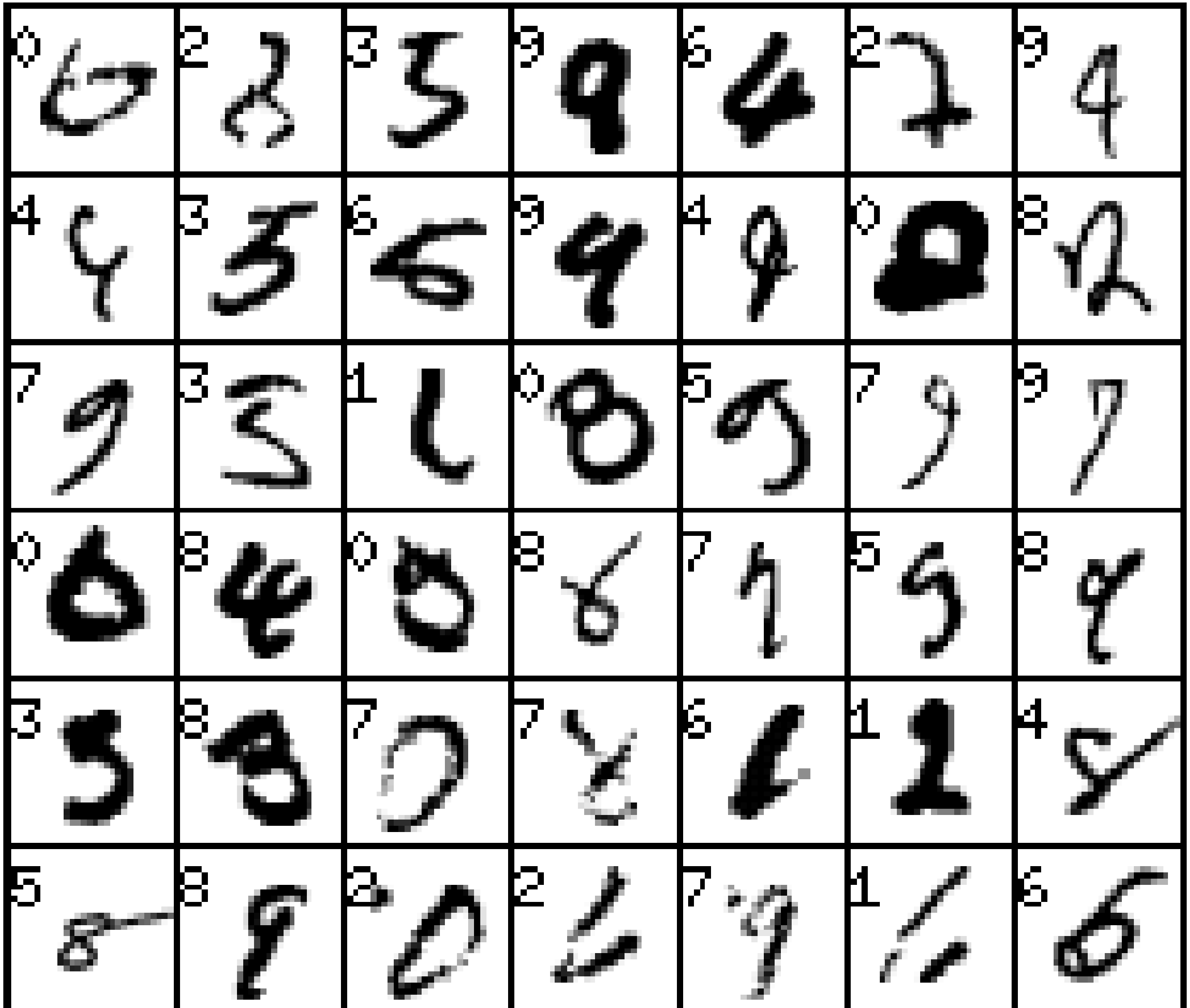
Test error rate: 0.39%. Training error rate: 0.23%.

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

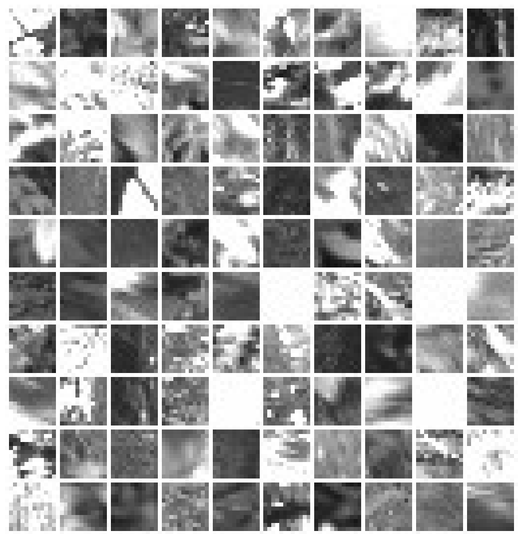
Best Results on MNIST (from raw images: no preprocessing)

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

MNIST Errors (0.42% error)



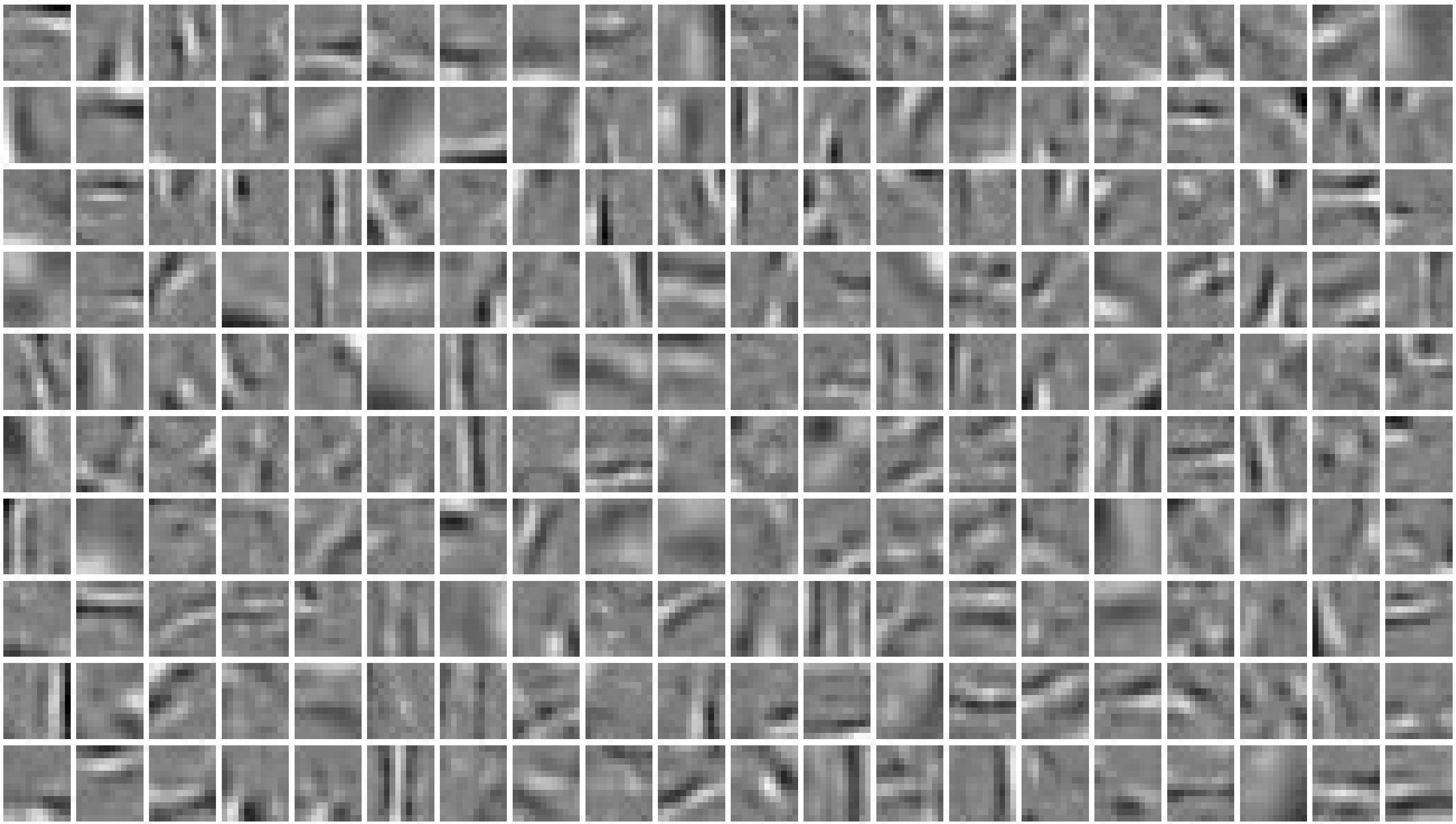
Training on natural image patches



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.

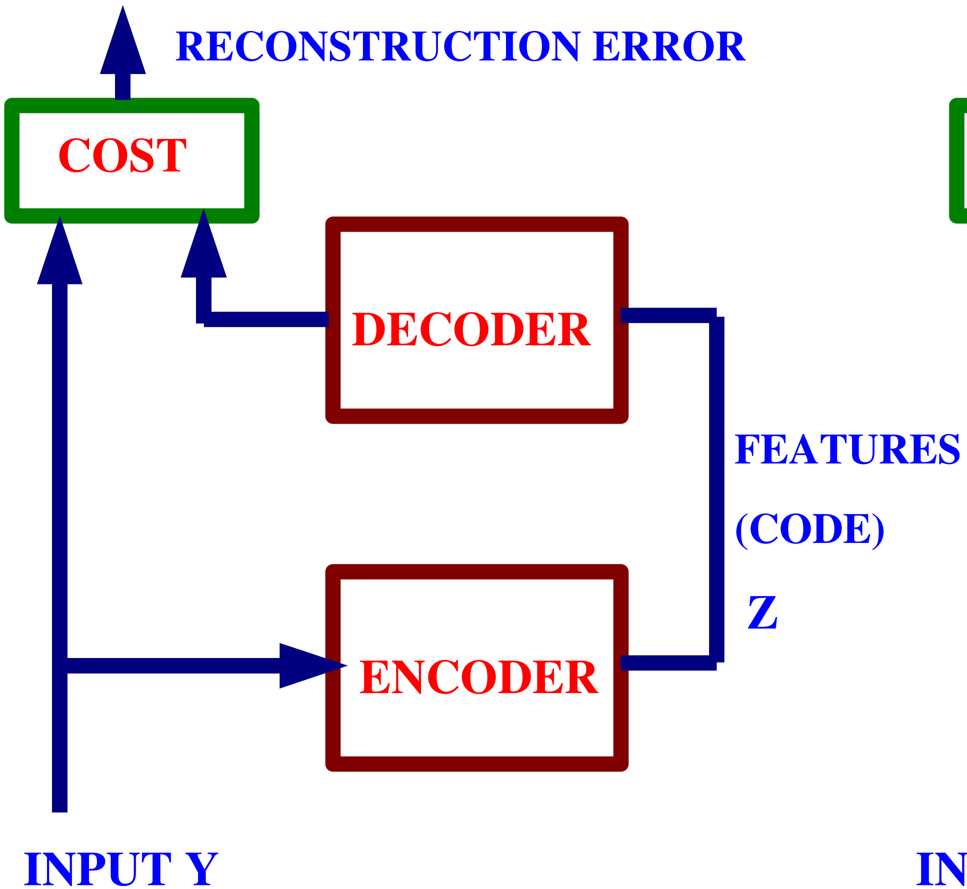
Natural image patches: Filters



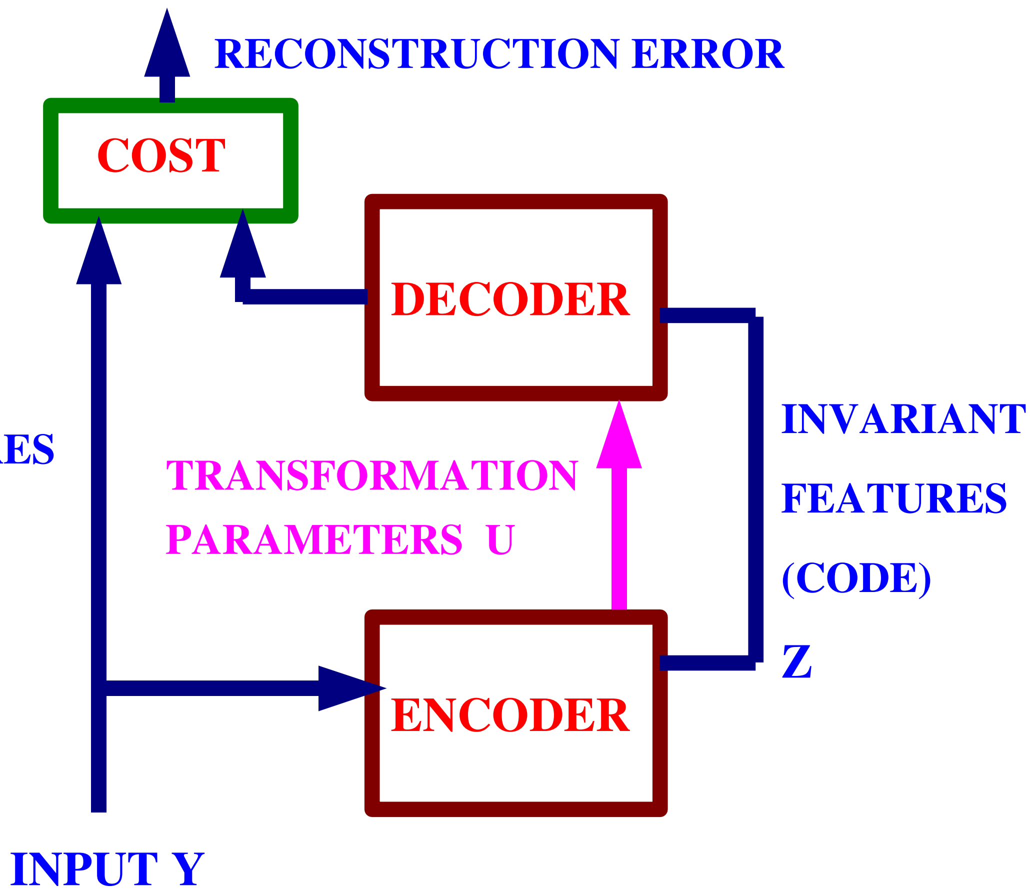
200 decoder filters (reshaped columns of matrix $\mathbf{W_d}$)

Learning Invariant Feature Hierarchies

- **Learning Shift Invariant Features**



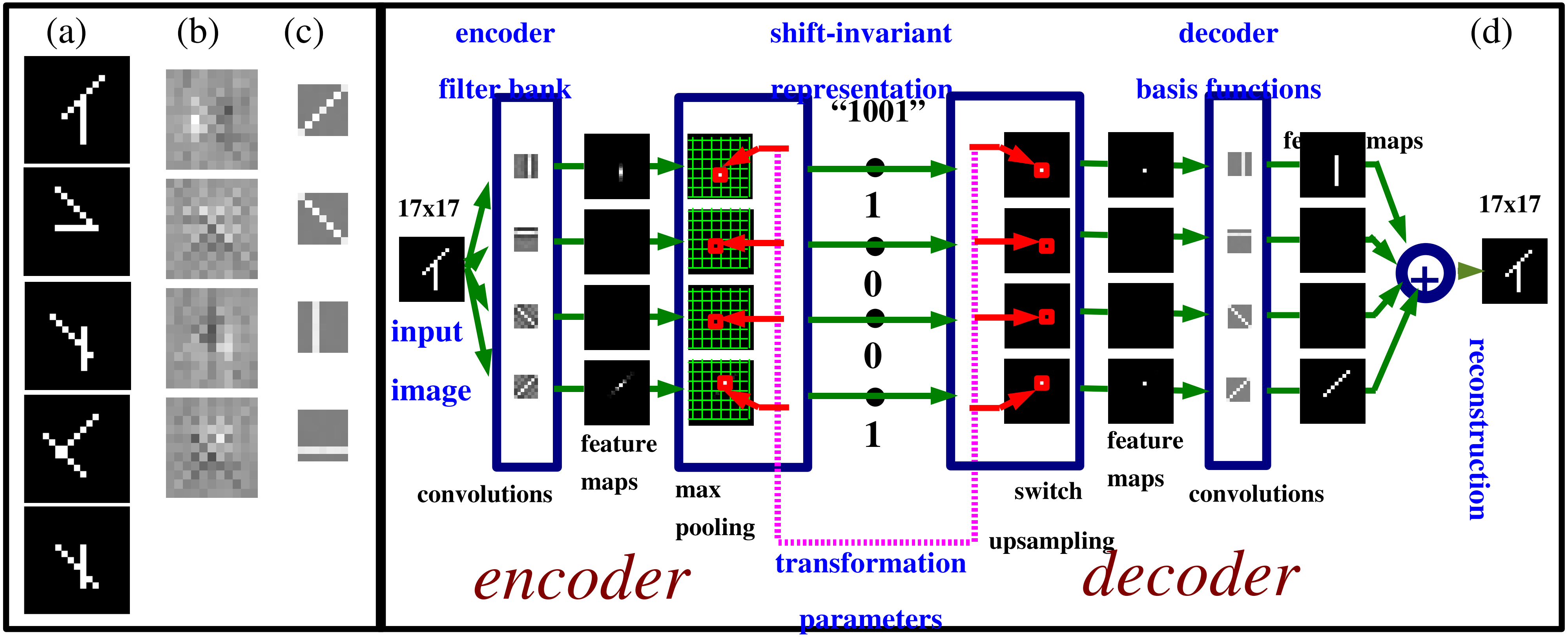
Standard Feature Extractor



Invariant Feature Extractor

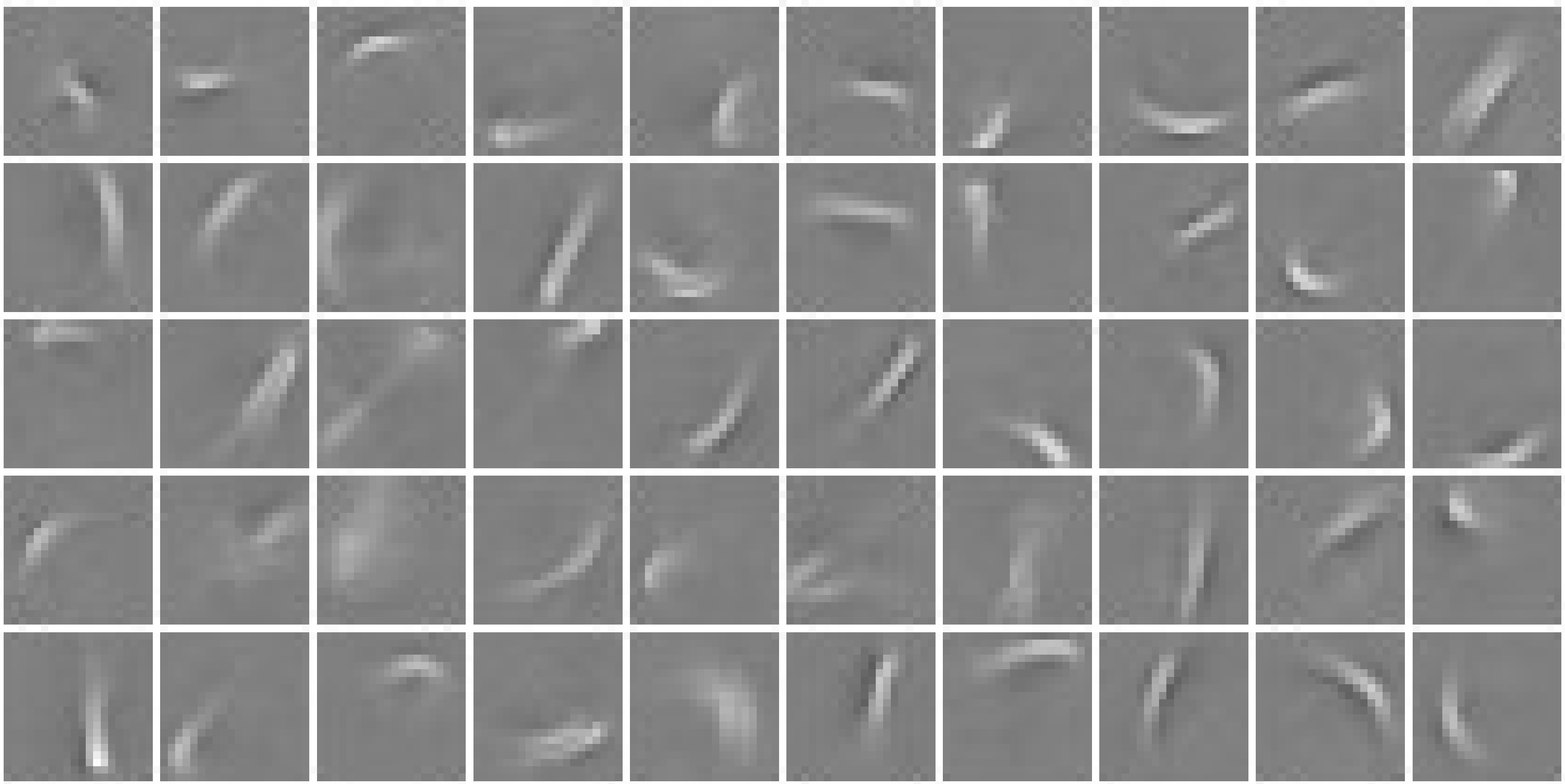
Learning Invariant Feature Hierarchies

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!



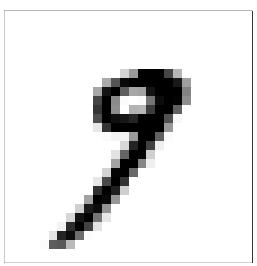
Example of Reconstruction

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

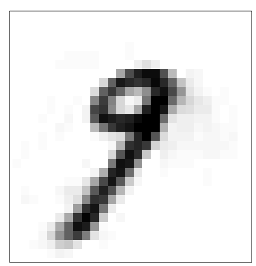
ORIGINAL DIGIT

RECONS-TRUCTION

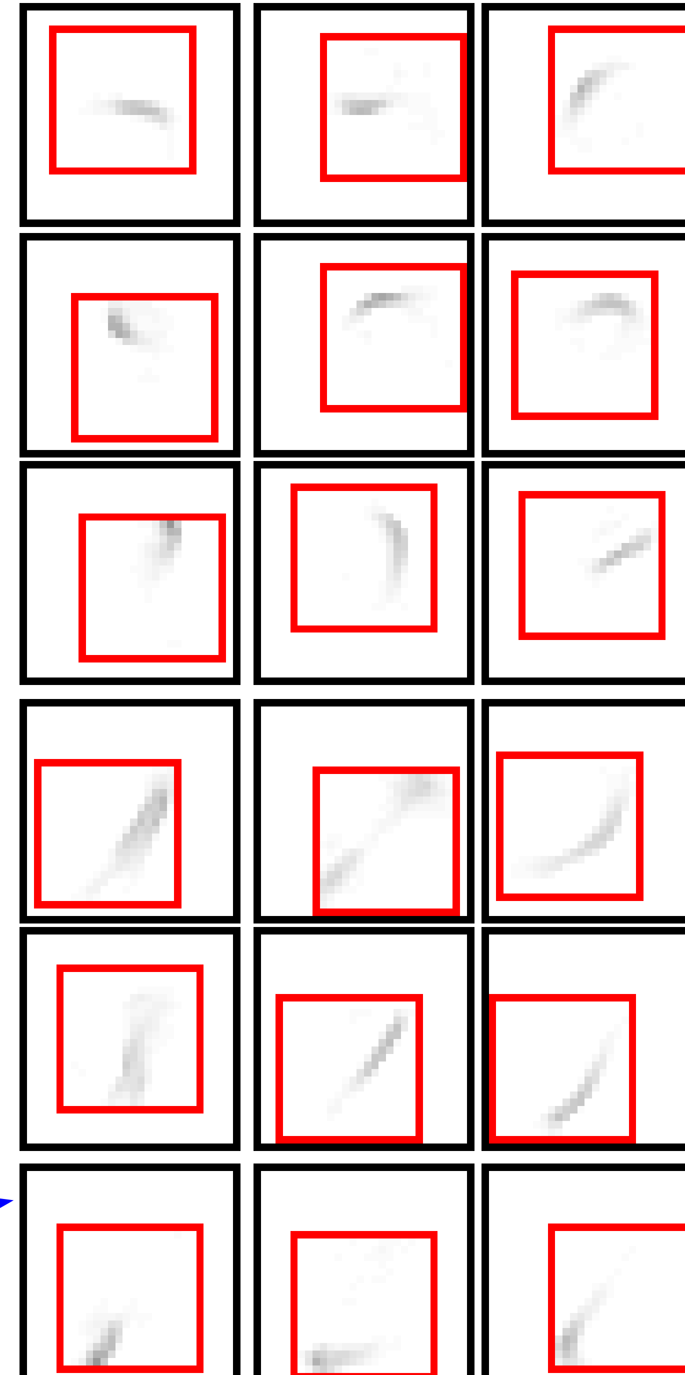
$\sum$



$\cong$



$=$



ACTIVATED DECODER BASIS FUNCTIONS

(in feed-back layer)

red squares: decoder bases

Learning Invariant Filters in a Convolutional Net

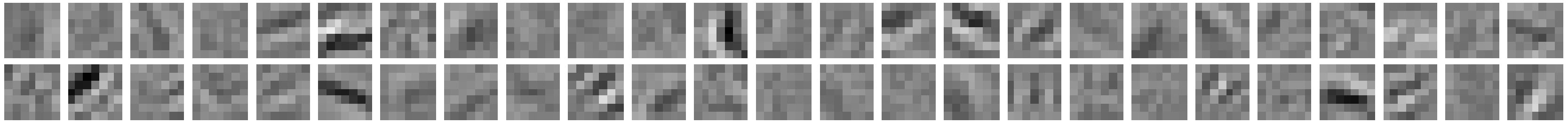


Figure 1: 50 7x7 filters in the first convolutional layer that were learned by the network trained supervised from *random* initial conditions with 600K digits.

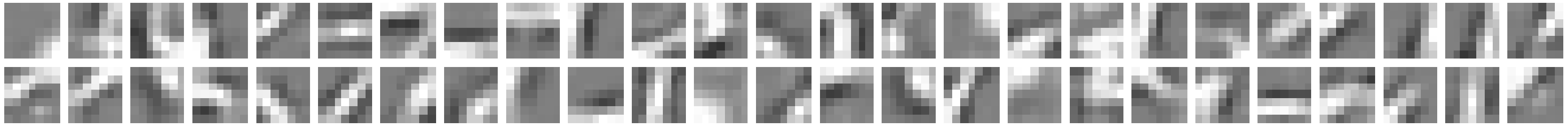


Figure 2: 50 7x7 filters that were learned by the unsupervised method (on 60K digits), and that are used to initialize the first convoltional layer of the network.

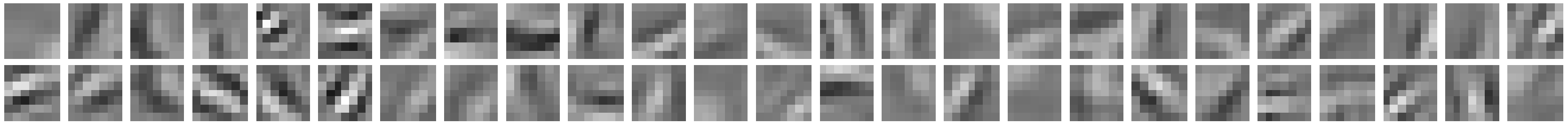
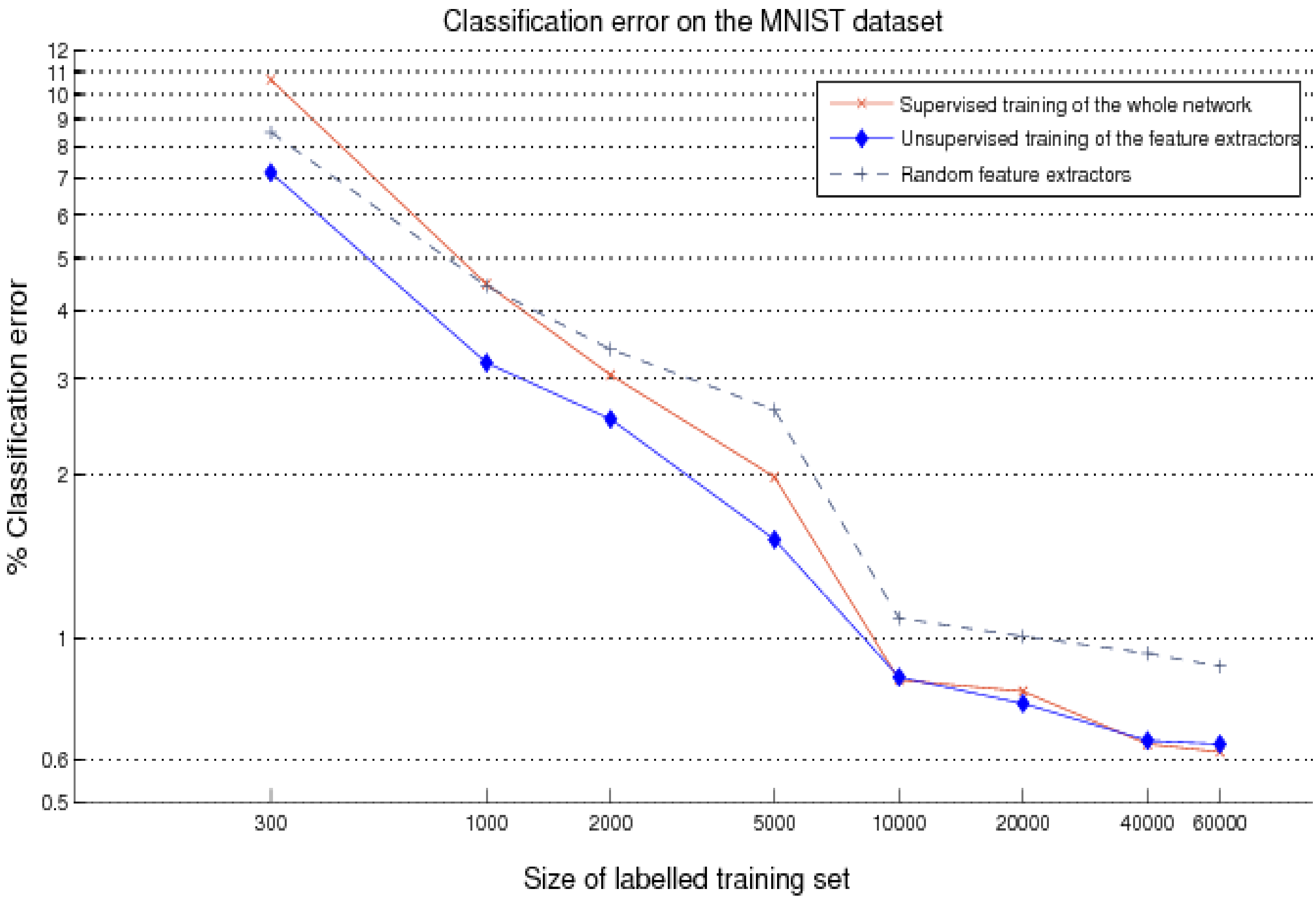


Figure 3: 50 7x7 filters in the first convolutional layer that were learned by the network trained supervised from the initial conditions given by the *unsupervised method* (see fig.2) with 600K digits.

Influence of Number of Training Samples

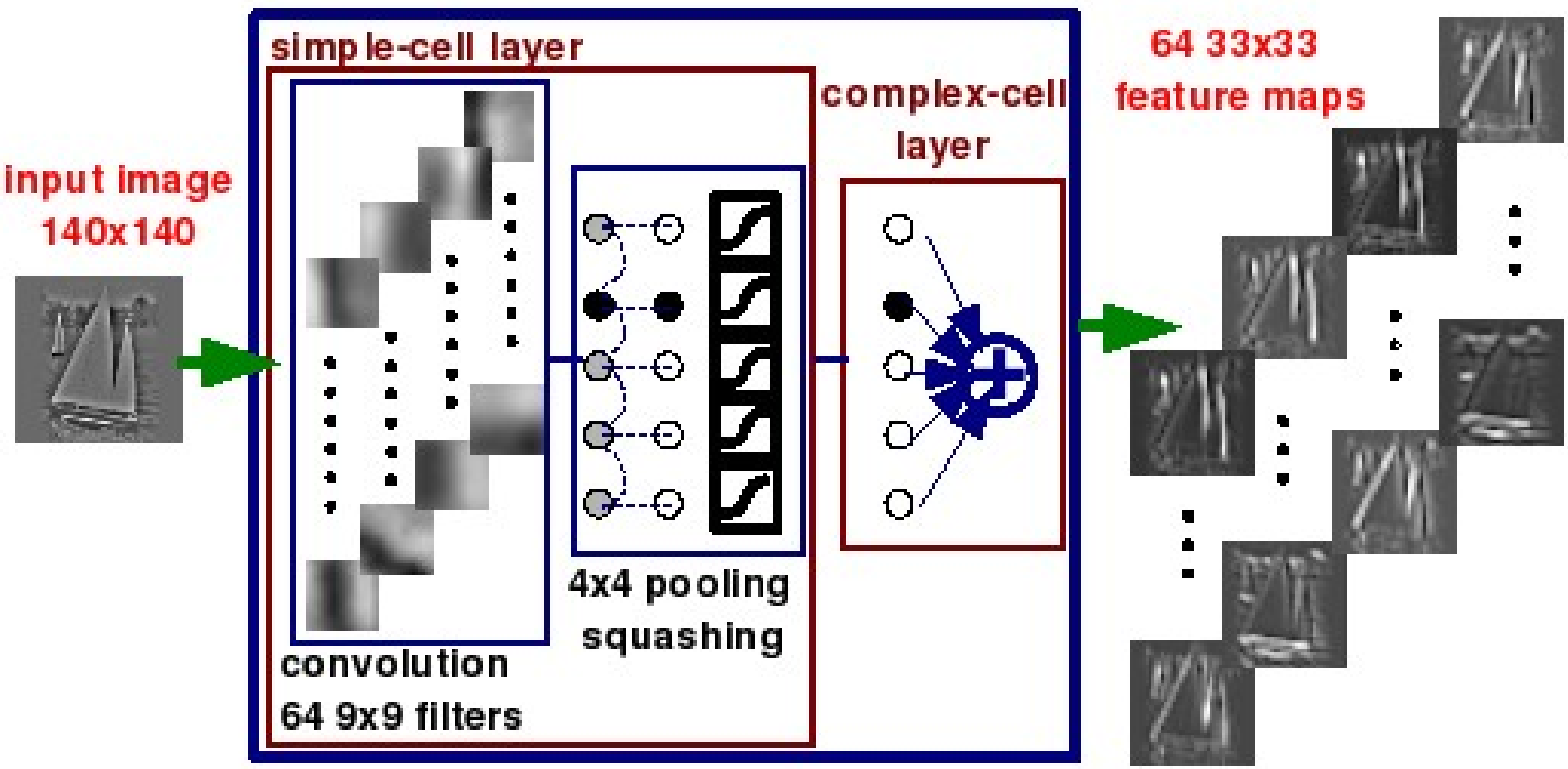


Generic Object Recognition: 101 categories + background

- **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**

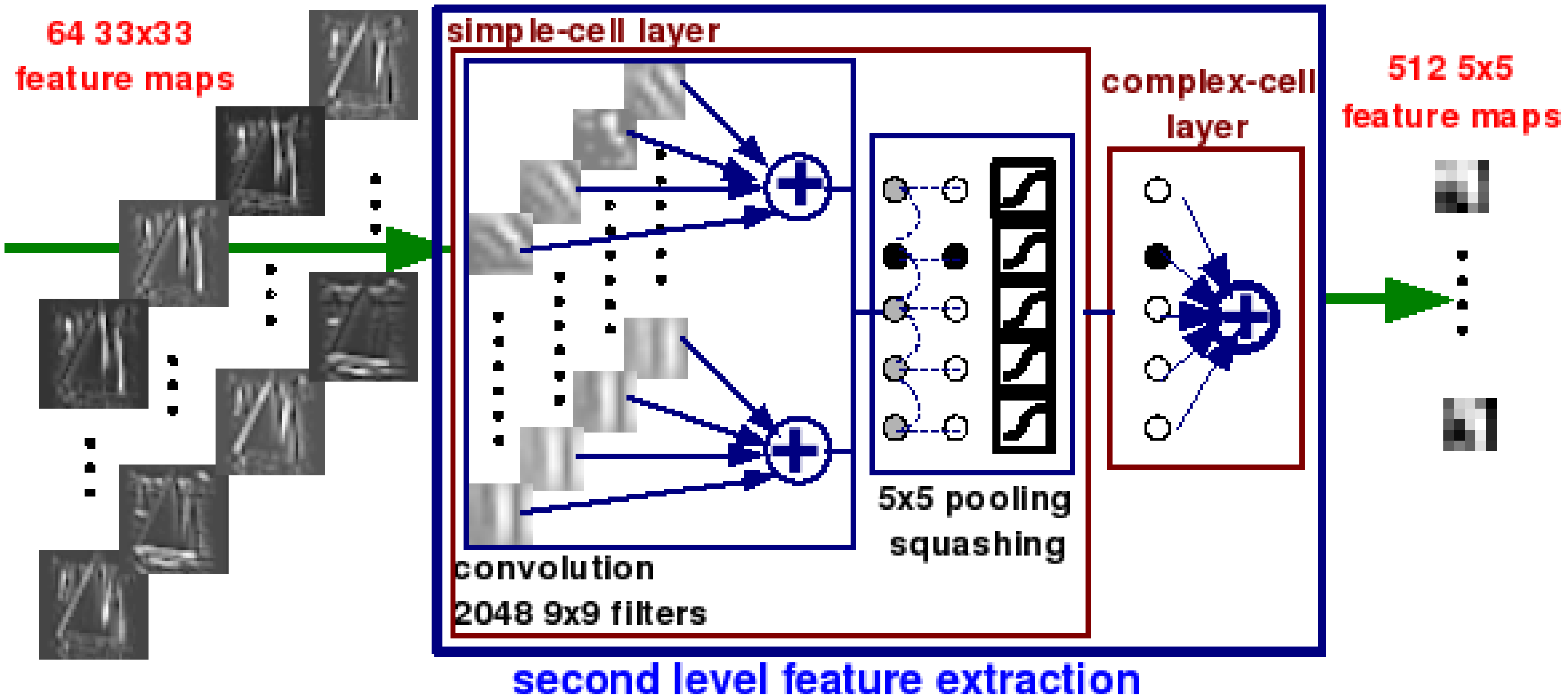
Training the 1st stage filters

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



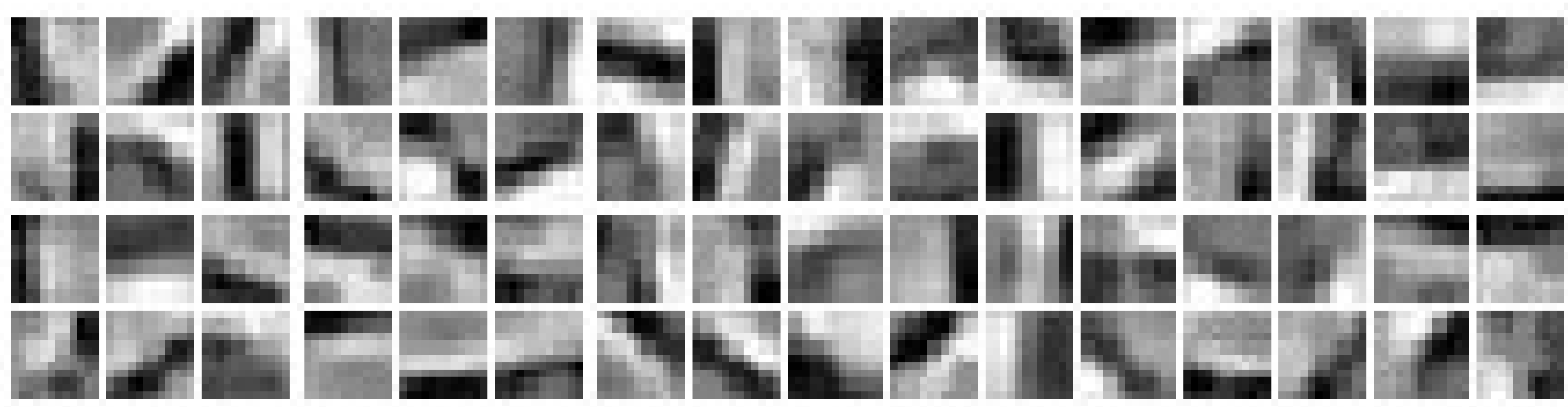
Training the 2nd stage filters

- 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

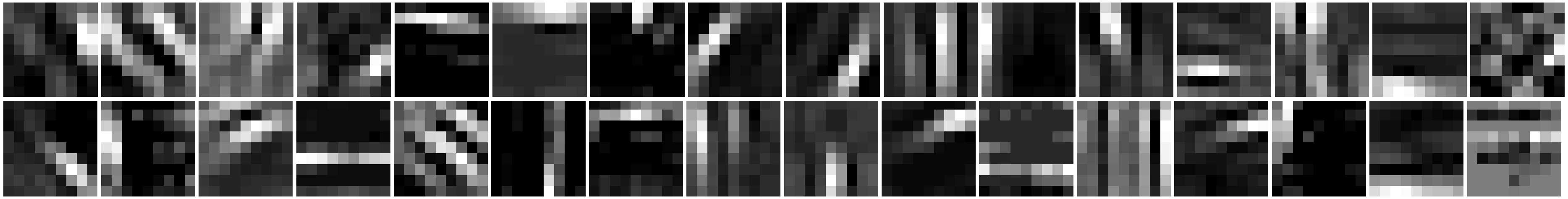


Generic Object Recognition: 101 categories + background

- 9x9 filters at the first level

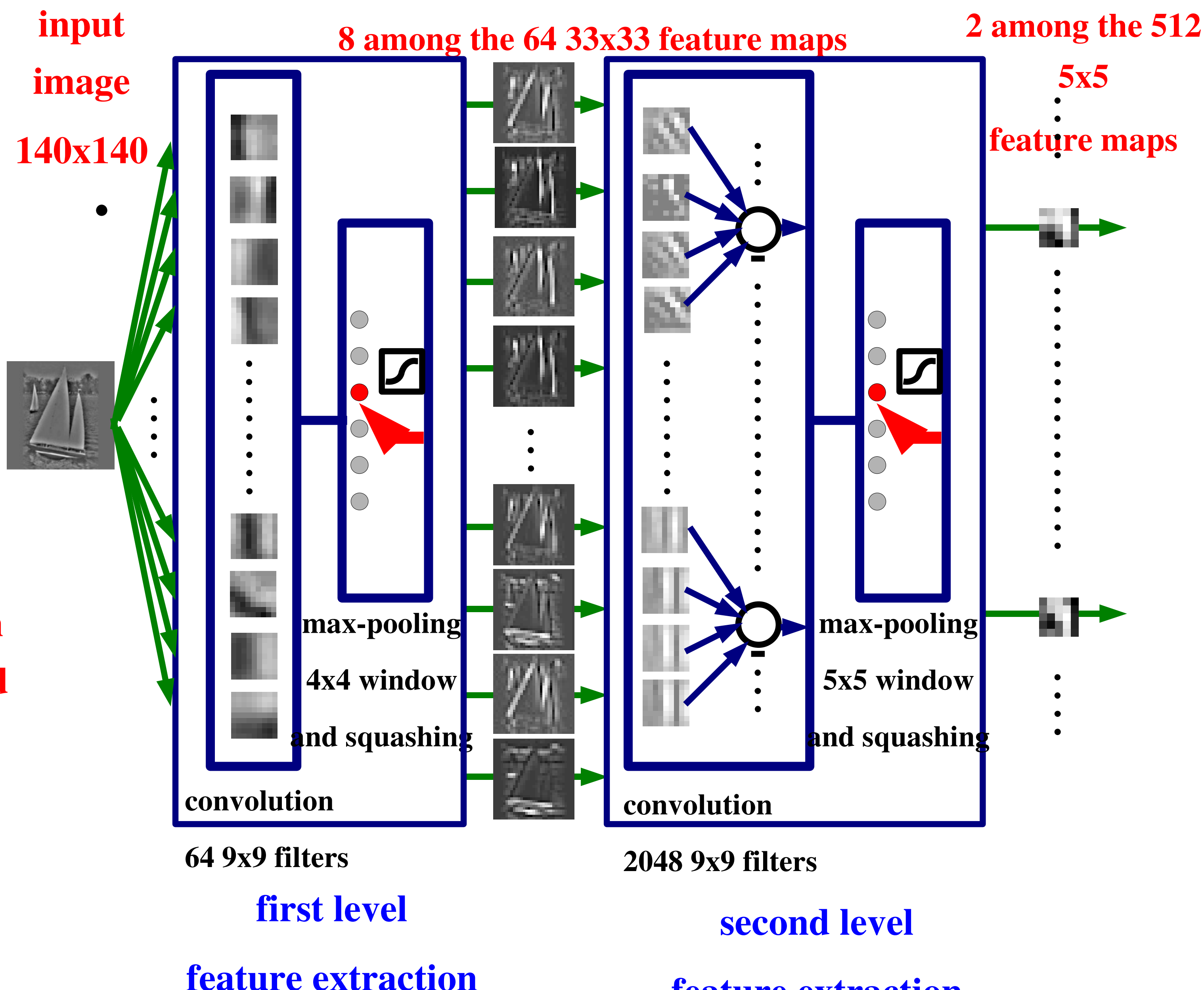


- 9x9 filters at the second level

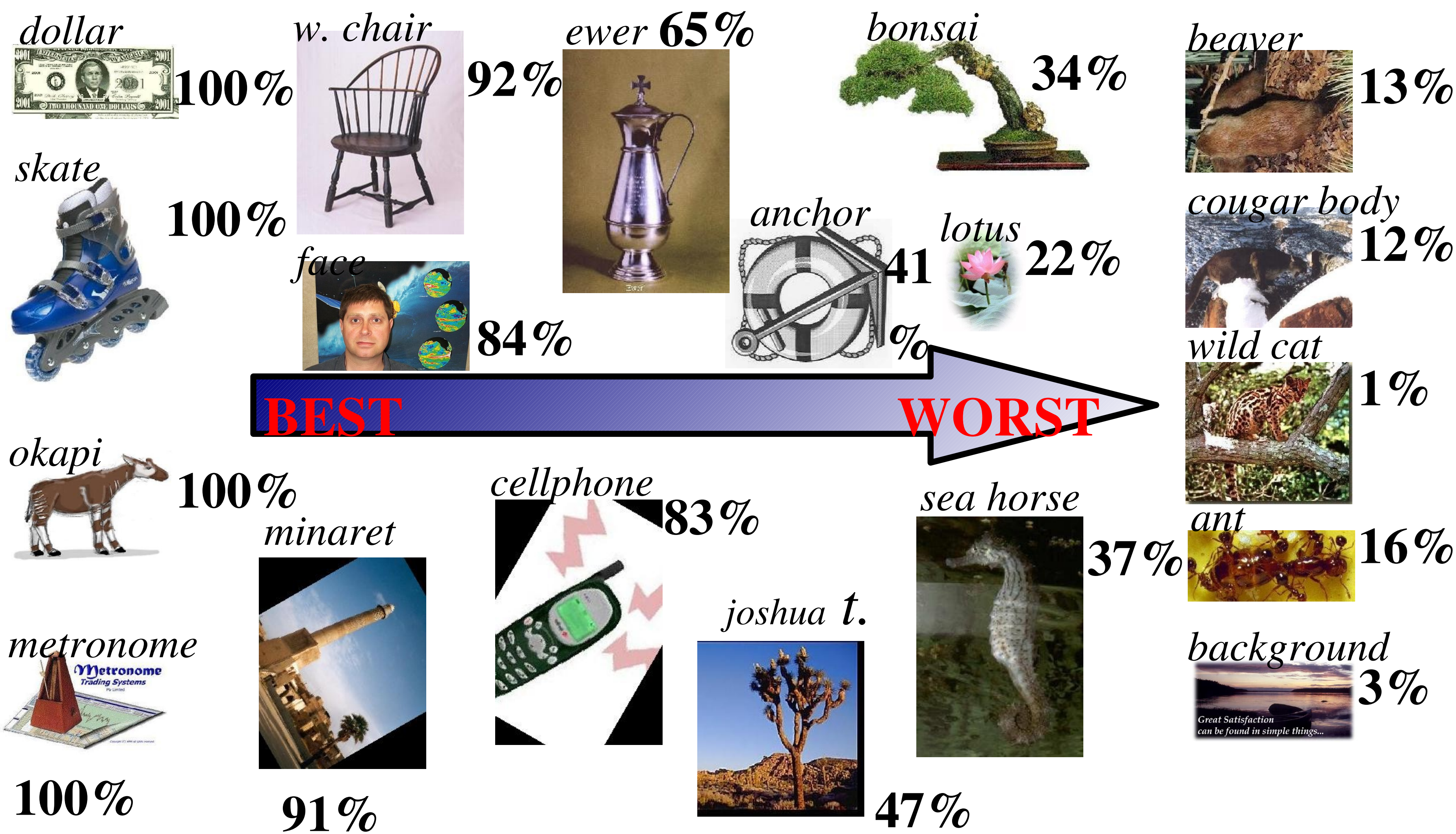


Shift-Invariant Feature Hierarchies on Caltech-101

- 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

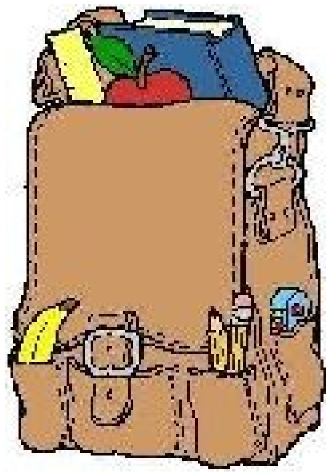


Recognition Rate on Caltech 101

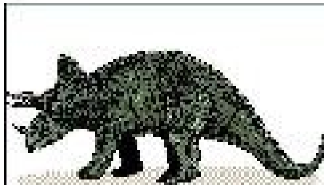


Caltech 256















Conclusion

- **Energy-Based Models is a general framework for probabilistic and non-probabilistic learning**
  - Make the energy of training samples low, make the energy of everything else high (e.g. Discriminant HMM, Graph Transformer Networks, Conditional Random Fields, Max Margin Markov Nets,...)
- **Invariant vision tasks require deep learning**
  - shallow models such as SVM can't learn complicated invariances.
- **Deep Supervised Learning works well with lots of samples**
  - Convolutional nets have record accuracy on handwriting recognition and face detection, and can be applied to many tasks.
- **Unsupervised Learning can reduce the need for labeled samples**
  - Stacks of sequentially-trained RBMs or sparse encoder-decoder layers learn good feature without requiring labeled samples
- **Learning invariant feature hierarchies**
  - yields excellent accuracy for shape recognition

Thank You