

A Tutorial on Energy-Based Learning

**Yann LeCun, Sumit Chopra, Raia Hadsell,
Marc'Aurelio Ranzato, Fu-Jie Huang
The Courant Institute of Mathematical Sciences
New York University**

<http://yann.lecun.com>

<http://www.cs.nyu.edu/~yann>

Two Problems in Machine Learning

1. The “Deep Learning Problem”

- ▶ “Deep” architectures are necessary to solve the invariance problem in vision (and perception in general)
- ▶ How do we train deep architectures with lots of non-linear stages

2. The “Partition Function Problem”

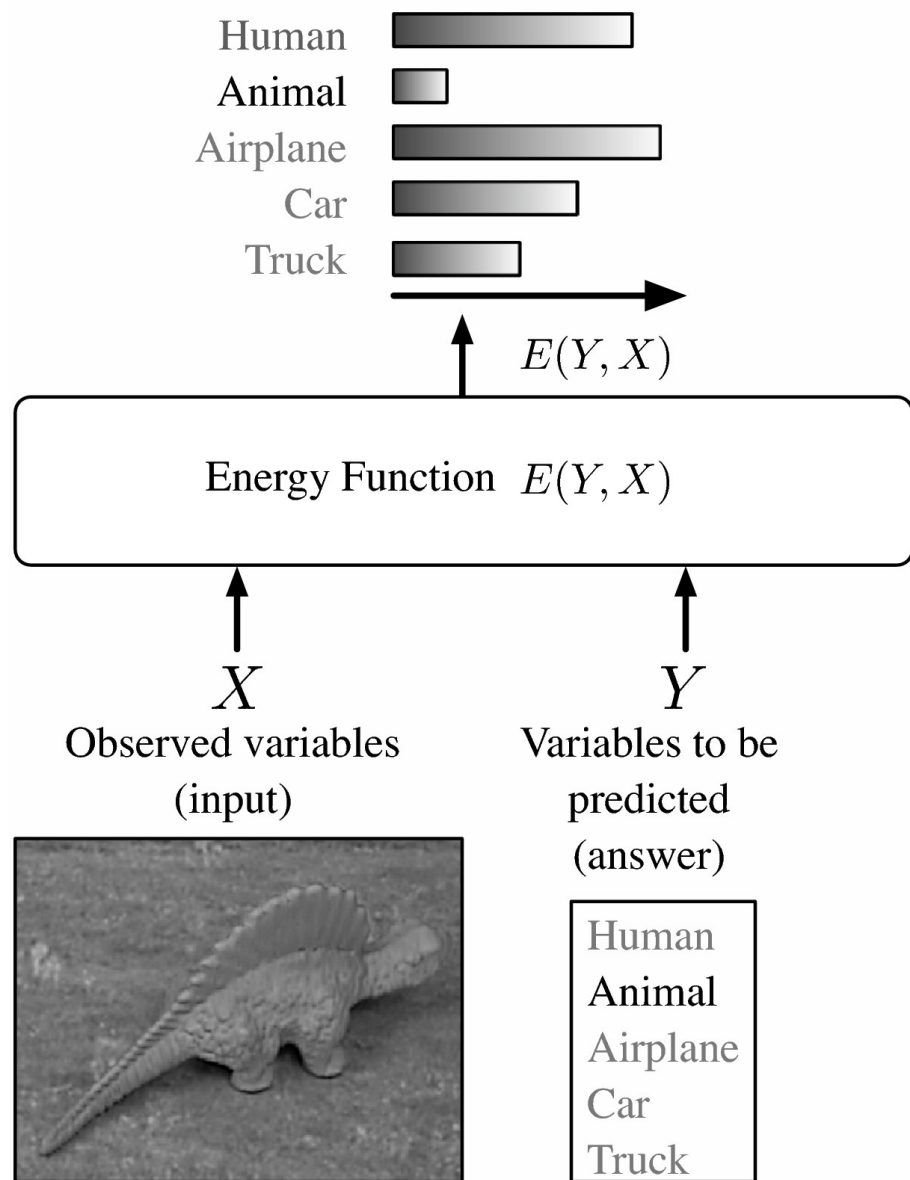
- ▶ Give high probability (or low energy) to good answers
- ▶ Give low probability (or high energy) to bad answers
- ▶ There are too many bad answers!

This tutorial discusses problem #2

- ▶ The partition function problem arises with probabilistic approaches
- ▶ Non-probabilistic approaches may allow us to get around it.

Energy-Based Learning provides a framework in which to describe probabilistic and non-probabilistic approaches to learning

Energy-Based Model for Decision-Making

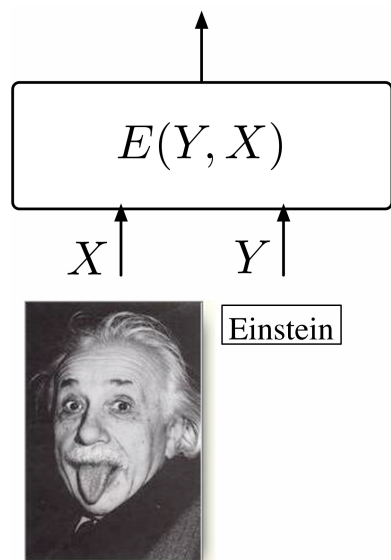


● **Model:** Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function $E(Y, X)$.

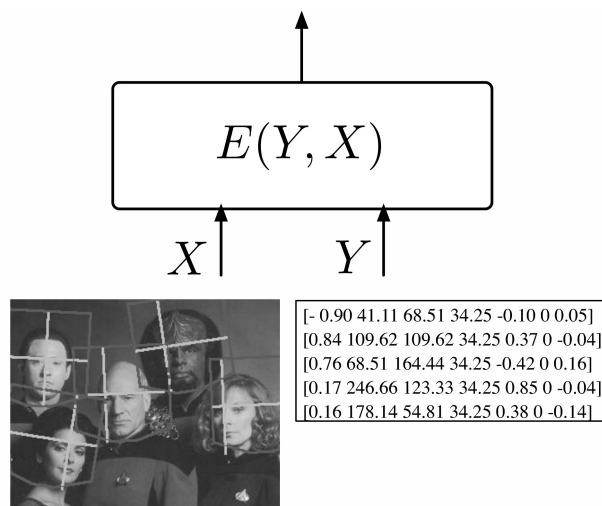
$$Y^* = \operatorname{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$

- **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.

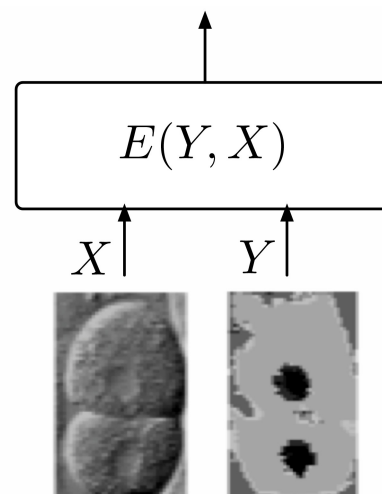
Complex Tasks: Inference is non-trivial



(a)

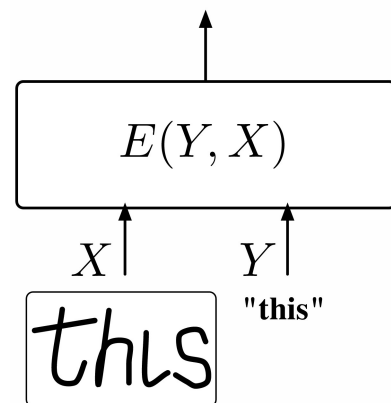


(b)

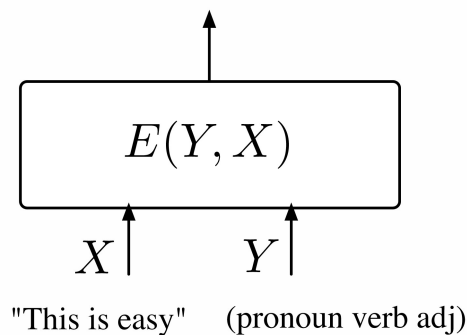


(c)

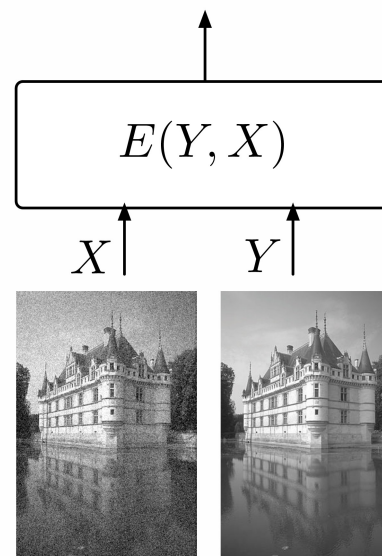
When the cardinality or dimension of Y is large, exhaustive search is impractical.



(d)



(e)



(f)

We need to use a “smart” inference procedure: min-sum, Viterbi,

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 Y_1 or Y_2 more compatible with X ?”
- ▶ Applications: Data-mining....
- ▶ Training: produce energies that rank the answers correctly

3. Detection:

- ▶ “Is this value of Y compatible with X ?”
- ▶ Application: face detection....
- ▶ Training: energies that increase as the image looks less like a face.

4. 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 units)

• 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** $\hat{\mathcal{S}} = \{(X^i, Y^i) : i = 1 \dots P\}.$

• **Loss functional / Loss function** $\mathcal{L}(E, \mathcal{S}) \quad \mathcal{L}(W, \mathcal{S})$

► Measures the quality of an energy function

• **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

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

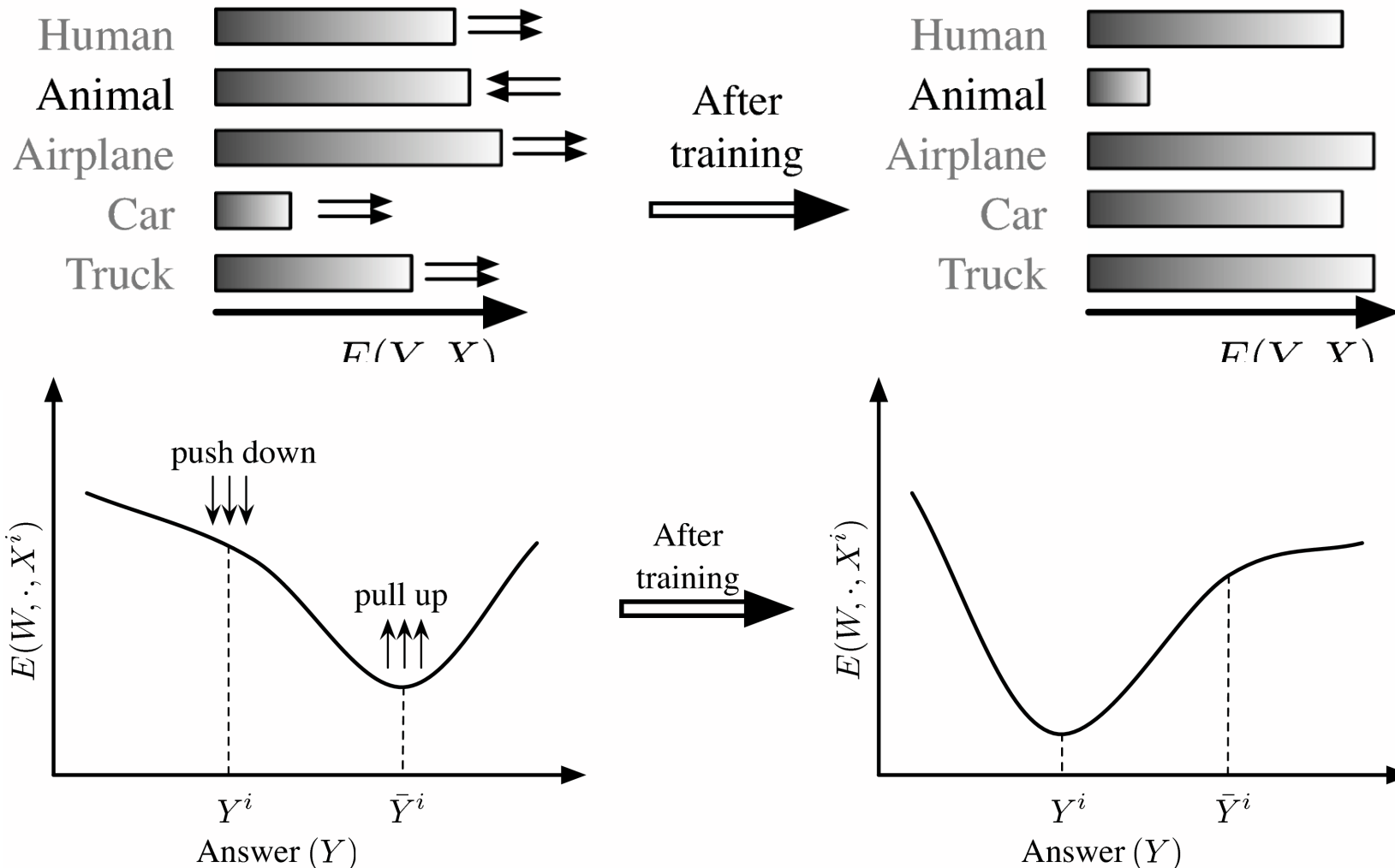
Per-sample
loss

Desired
answer

Energy surface
for a given X_i
as Y varies

Regularizer

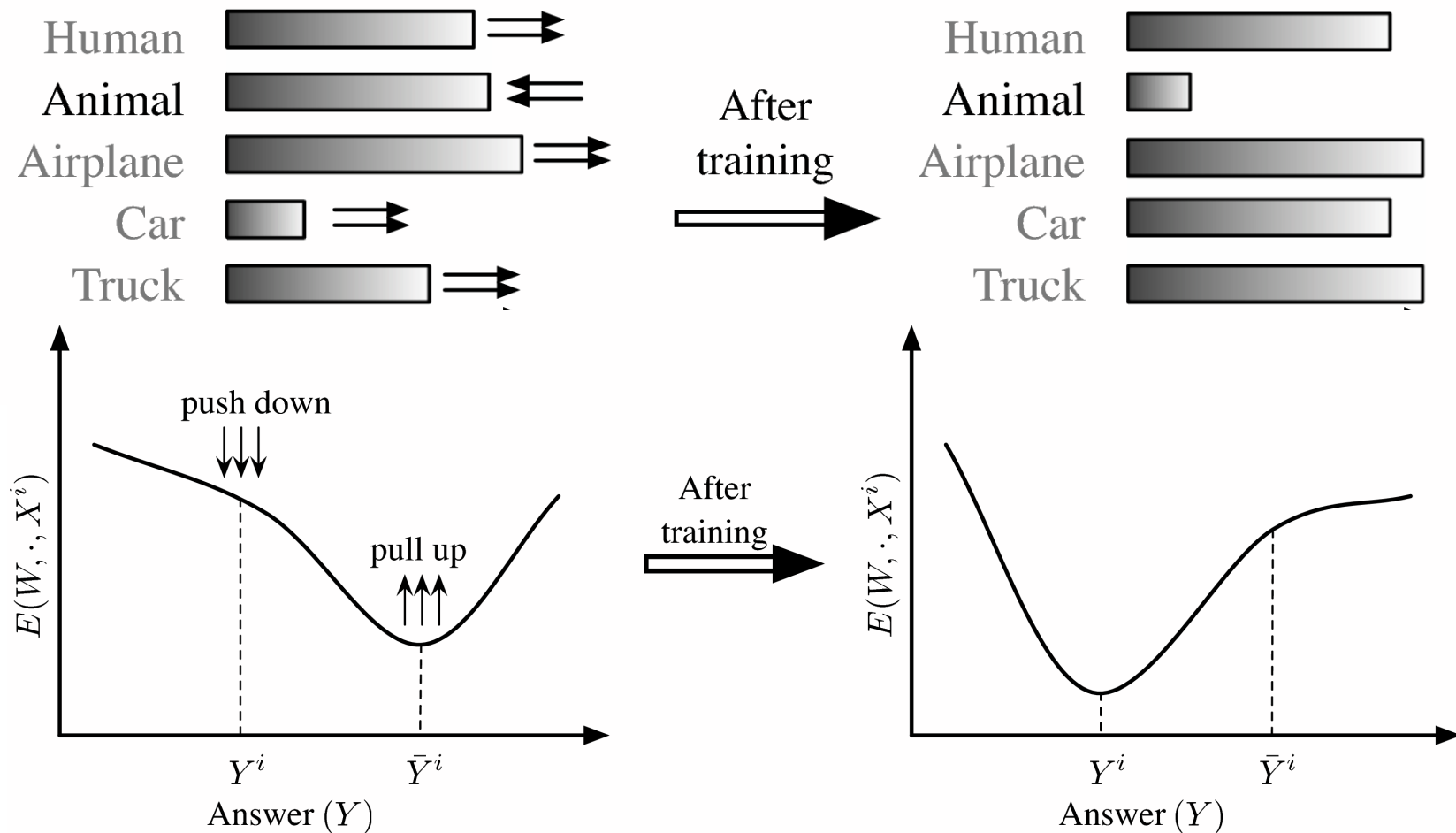
Designing a Loss Function



Correct answer has the lowest energy -> **LOW LOSS**

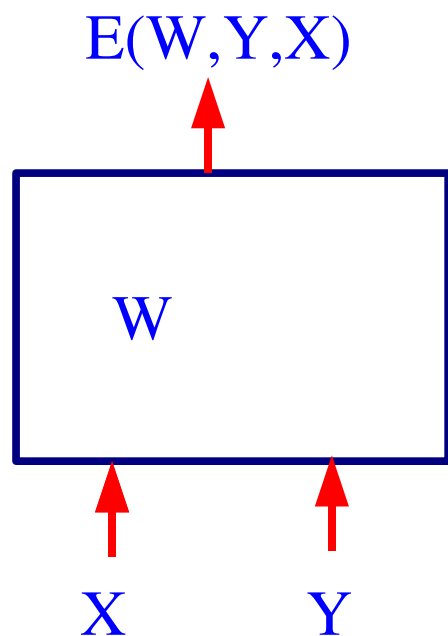
Lowest energy is not for the correct answer -> **HIGH LOSS**

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

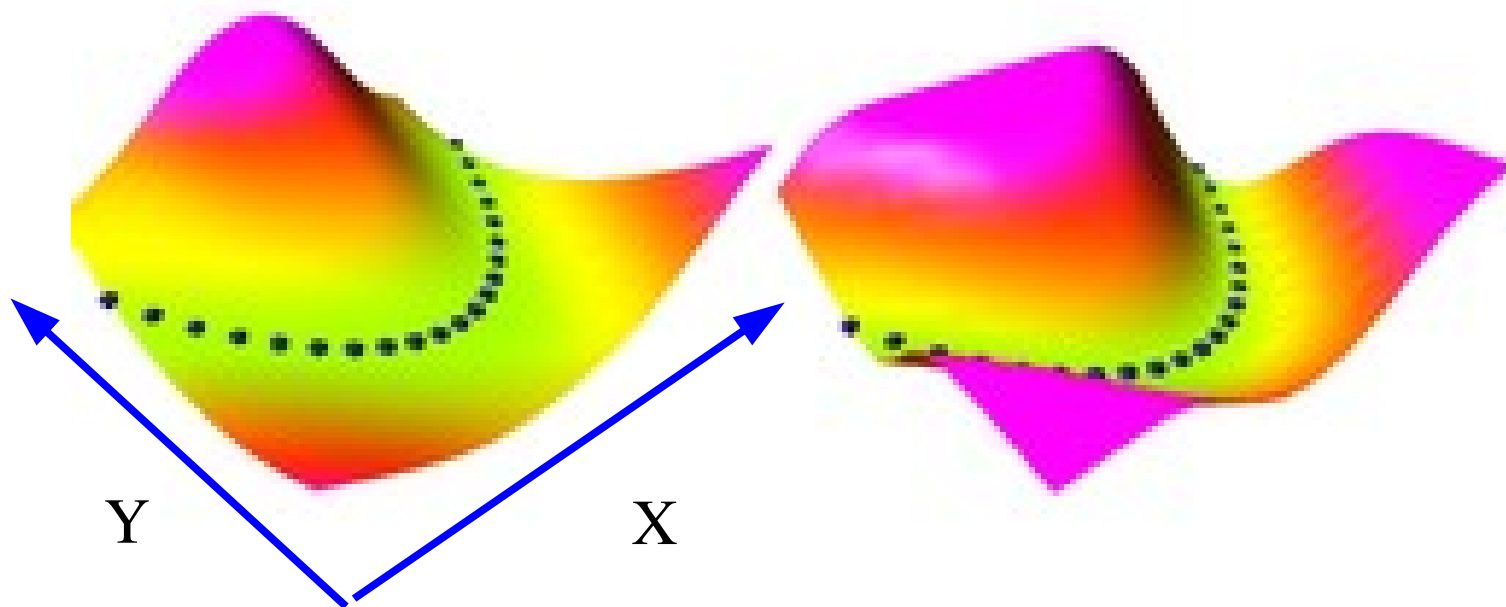
Architecture + Inference Algo + Loss Function = Model



- 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.**

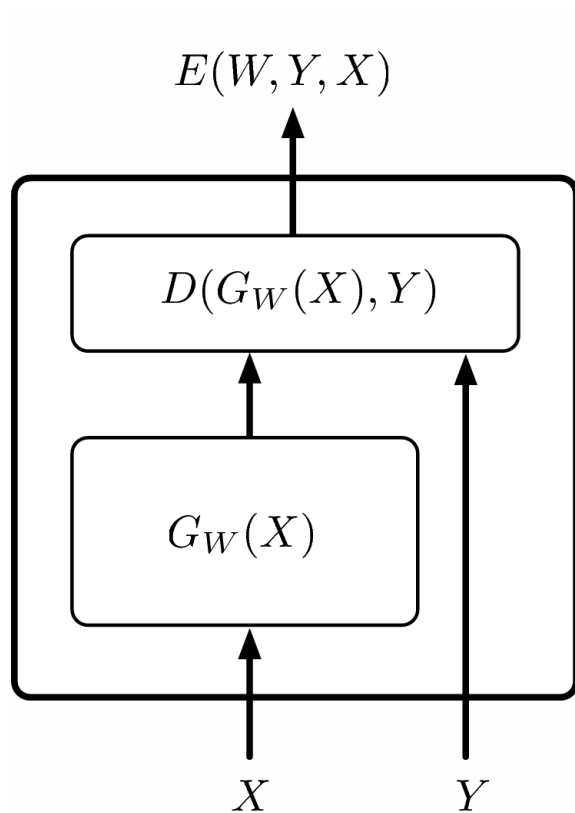
PROBLEM: What loss functions will make the machine approach the desired behavior?

Several Energy Surfaces can give the same answers



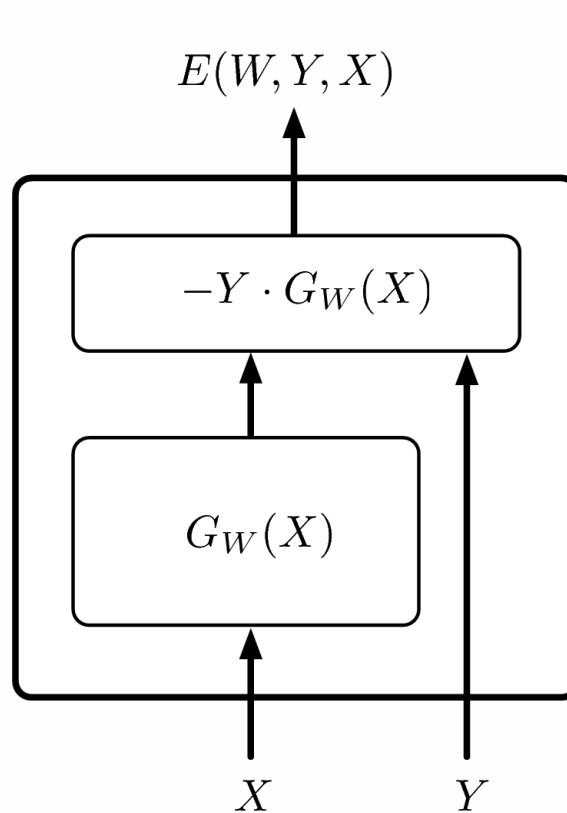
- Both surfaces compute $Y = X^2$
- $\text{MIN}_Y E(Y, X) = X^2$
- Minimum-energy inference gives us the same answer

Simple Architectures



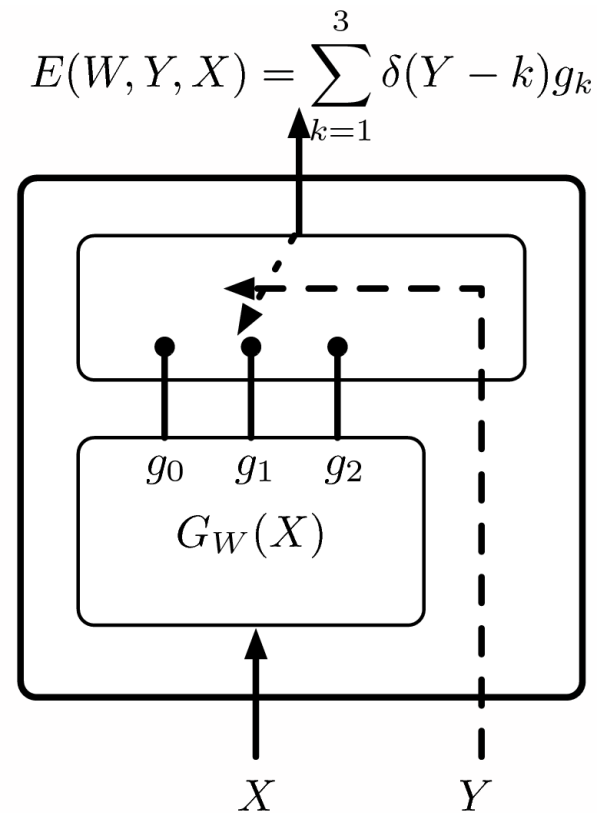
Regression

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



Binary Classification

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



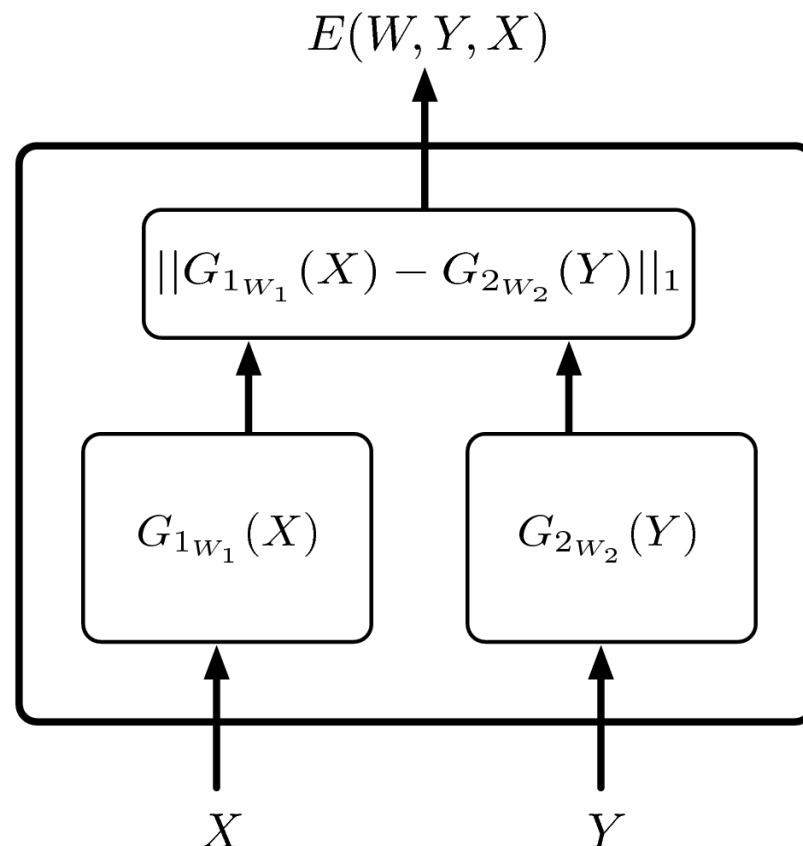
Multi-class
Classification

Simple Architecture: Implicit Regression

$$E(W, X, Y) = \|G_{1_{w_1}}(X) - G_{2_{w_2}}(Y)\|_1,$$

■ The Implicit Regression architecture

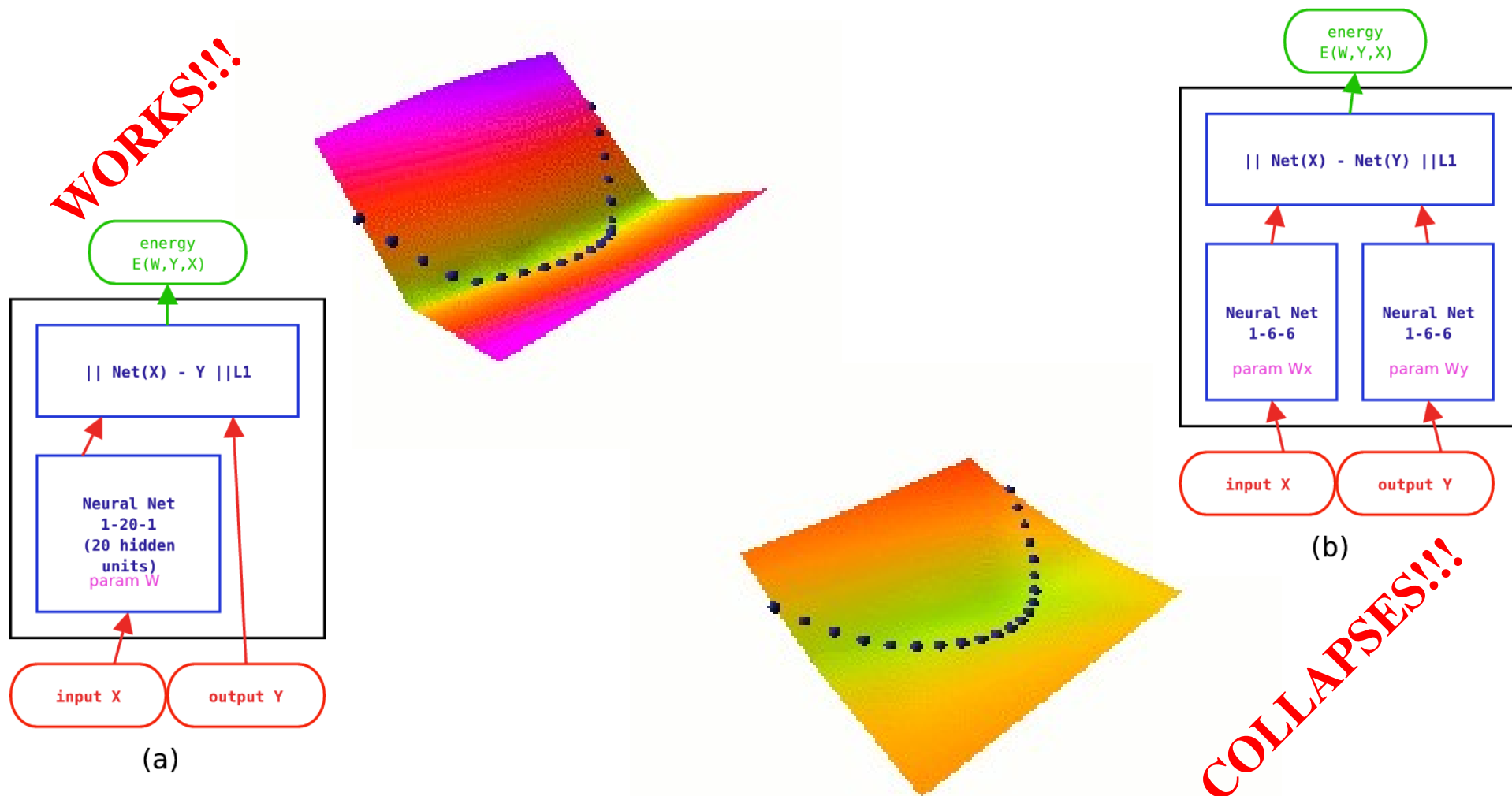
- ▶ allows multiple answers to have low energy.
- ▶ Encodes a constraint between X and Y rather than an explicit functional relationship
- ▶ This is useful for many applications
- ▶ Example: sentence completion: “The cat ate the {mouse,bird,homework,...}”
- ▶ [Bengio et al. 2003]
- ▶ But, inference may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



Examples of Loss Functions: Perceptron Loss

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

■ 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

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

• **Energy:** $E(W, Y, X) = -Y G_W(X),$

• **Inference:** $Y^* = \operatorname{argmin}_{Y \in \{-1, 1\}} -Y G_W(X) = \operatorname{sign}(G_W(X)).$

• **Loss:** $\mathcal{L}_{\text{perceptron}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^P (\operatorname{sign}(G_W(X^i)) - Y^i) G_W(X^i).$

• **Learning Rule:** $W \leftarrow W + \eta (Y^i - \operatorname{sign}(G_W(X^i))) \frac{\partial G_W(X^i)}{\partial W},$

• **If $G_W(X)$ is linear in W :** $E(W, Y, X) = -Y W^T \Phi(X)$

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

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:

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

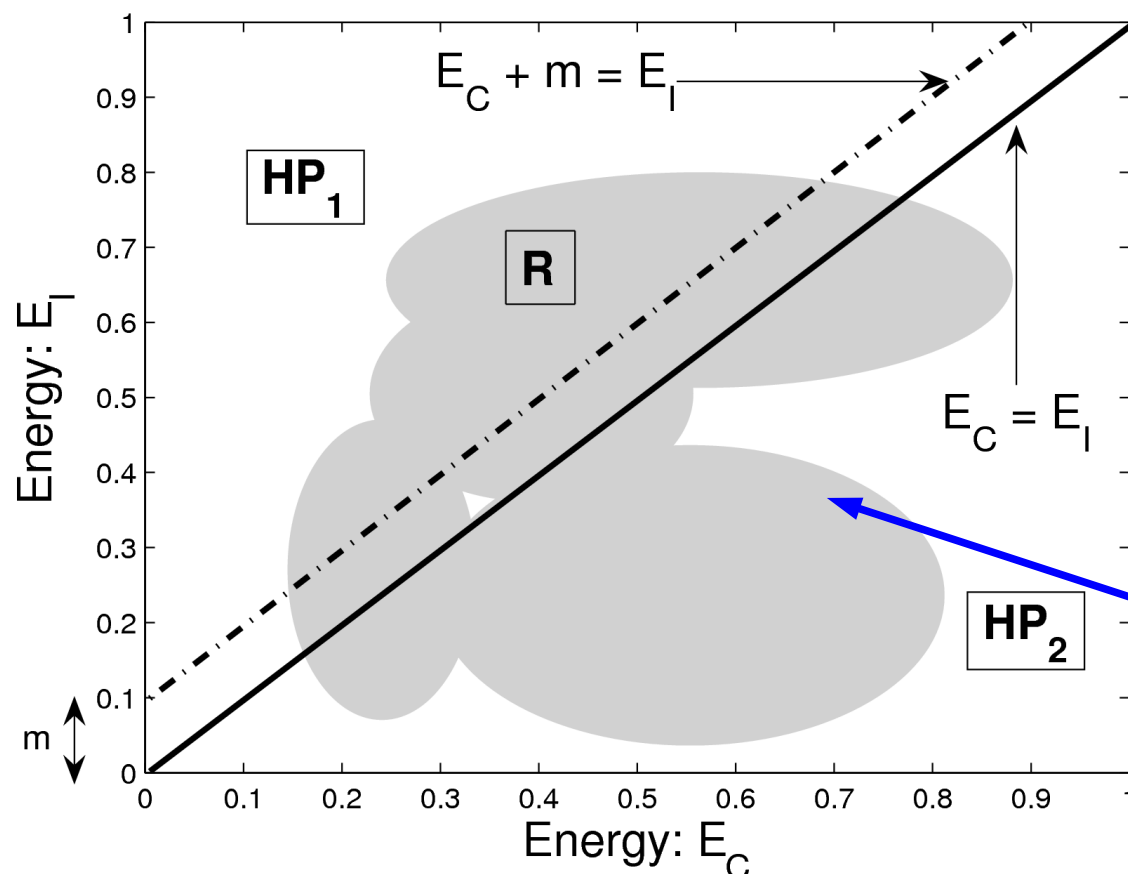
• **Most Offending Incorrect Answer: continuous case**

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:

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

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

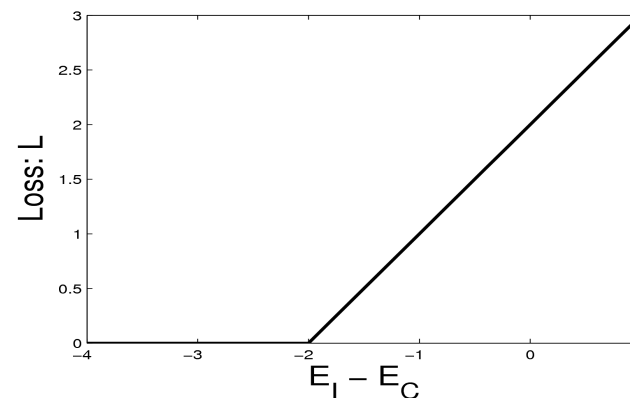
- ▶ Q_m increases with the energy of the correct answer
- ▶ Q_m decreases with the energy of the **most offending incorrect answer**
- ▶ whenever it is less than the energy of the correct answer plus a **margin m** .

Examples of Generalized Margin Losses

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

● Hinge Loss

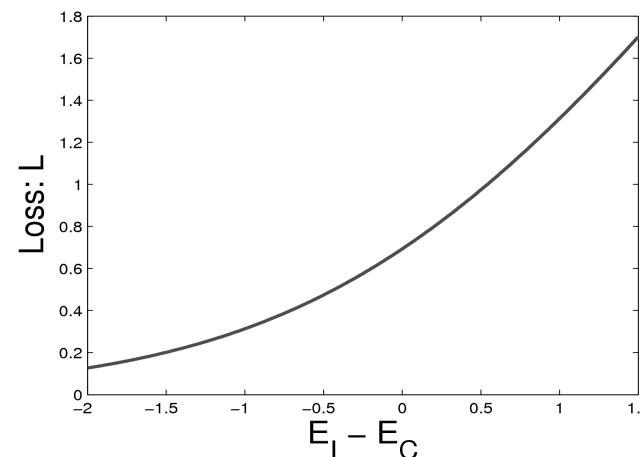
- ▶ [Altun et al. 2003], [Taskar et al. 2003]
- ▶ With the linearly-parameterized binary classifier architecture, we get linear SVM



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

● Log Loss

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

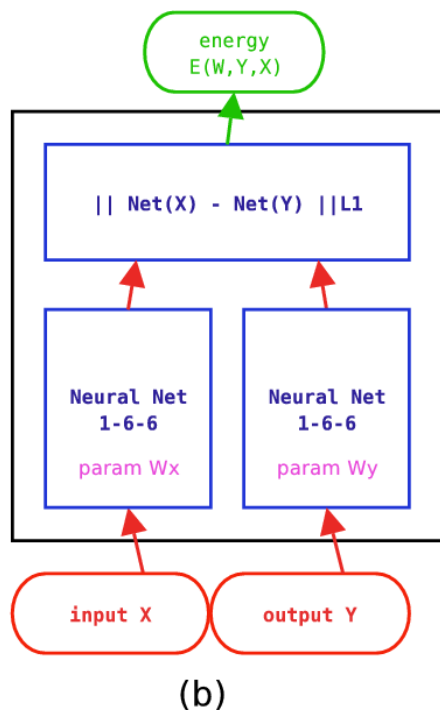


Examples of Margin Losses: Square-Square Loss

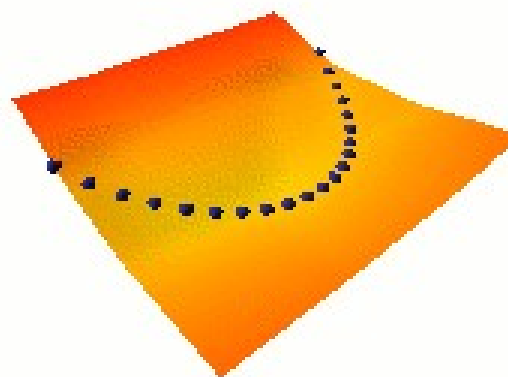
$$L_{\text{sq-sq}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \left(\max(0, m - E(W, \bar{Y}^i, X^i)) \right)^2.$$

■ Square-Square Loss

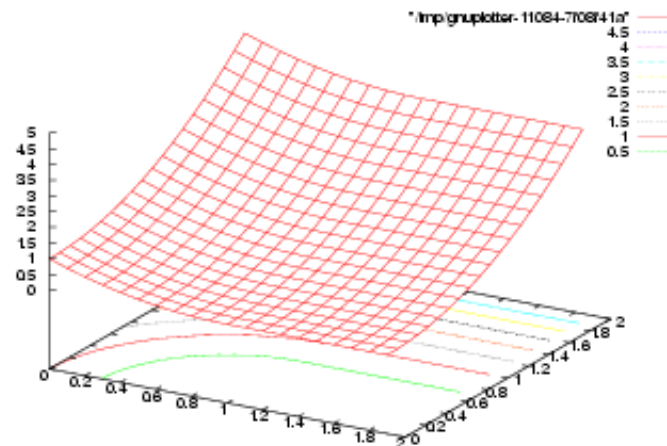
- ▶ [LeCun-Huang 2005]
- ▶ Appropriate for positive energy functions



Learning $Y = X^2$



NO COLLAPSE!!!



Other Margin-Like Losses

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

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

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

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

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

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

Negative Log-Likelihood Loss

- Conditional probability of the samples (assuming independence)

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

- 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:

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

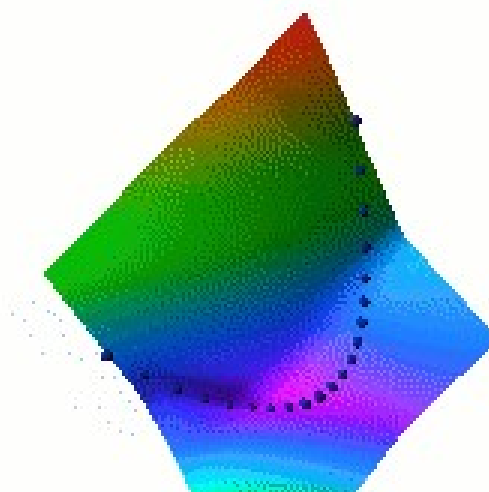
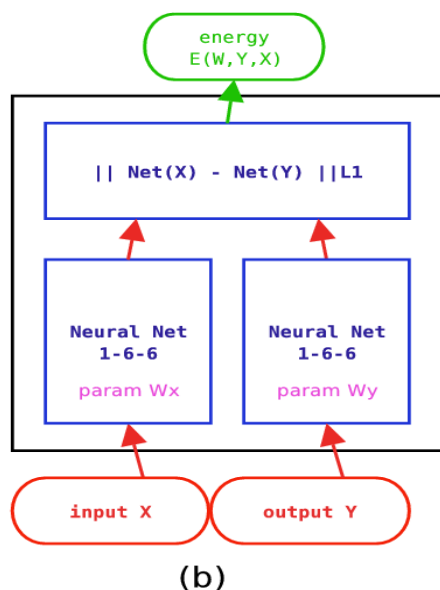
- Reduces to the perceptron loss when Beta->infinity

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

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

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



Negative Log-Likelihood Loss: Binary Classification

Binary Classifier Architecture:

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

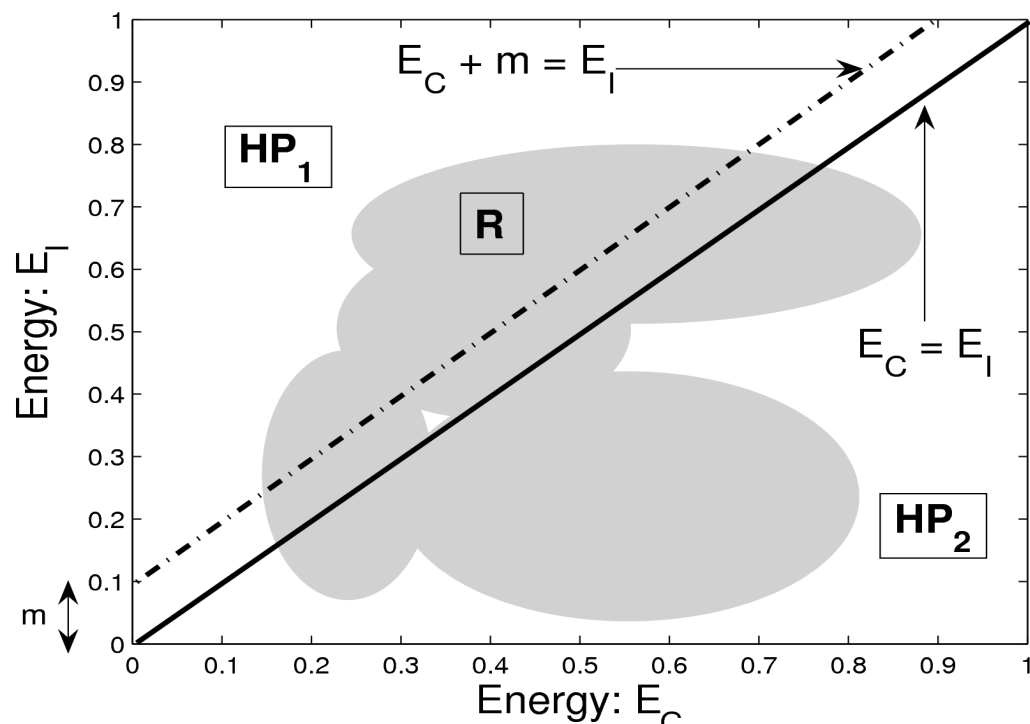
Linear Binary Classifier Architecture:

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

Learning Rule: logistic regression

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	$\max(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i))$	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	$\left(1 + e^{-(E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i))} \right)^{-1}$	> 0
square-square	$E(W, Y^i, X^i)^2 - (\max(0, m - E(W, \bar{Y}^i, X^i)))^2$	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

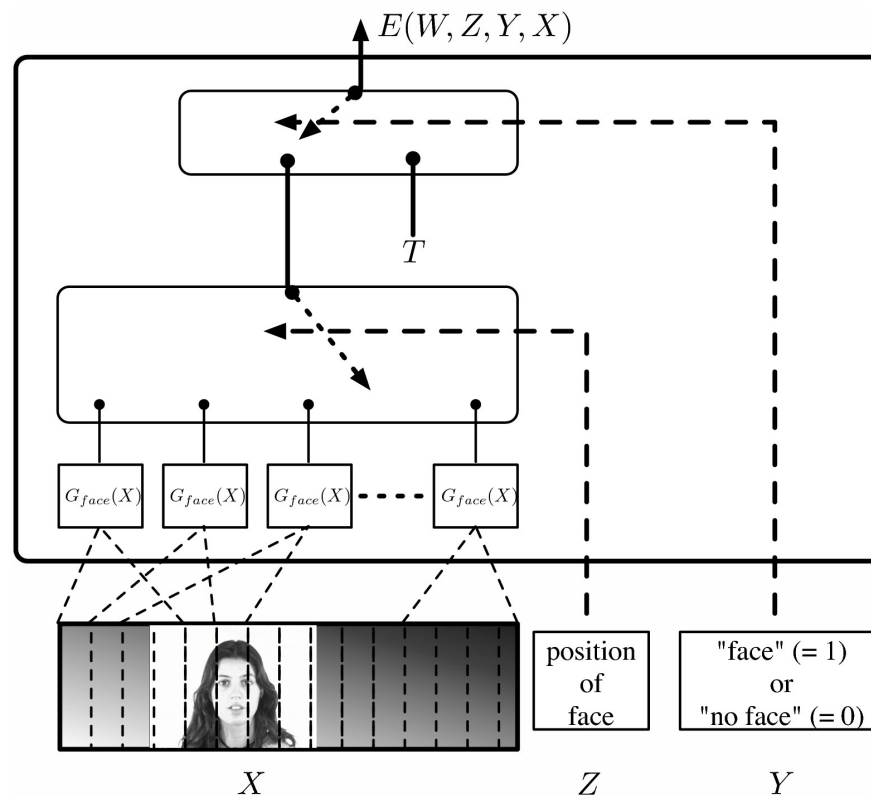
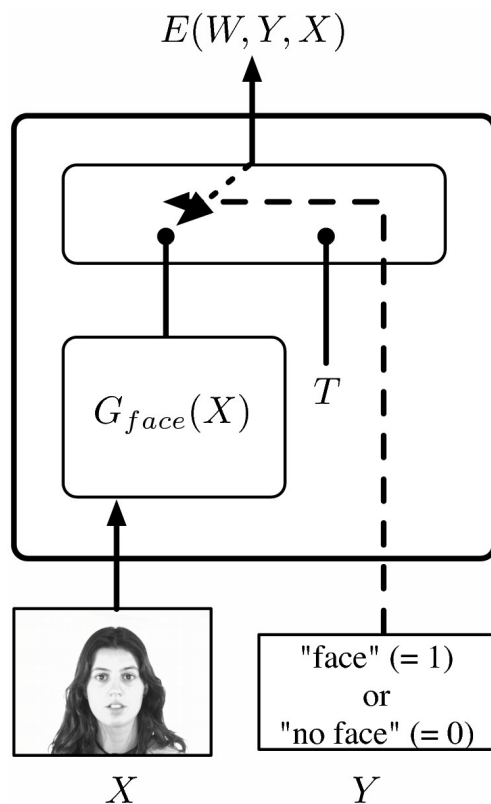
- Loss functions differ in how they pick the point(s) whose energy is pulled up, and how much they pull them up
- Losses with a log partition function in the contrastive term pull up all the bad answers simultaneously.
 - ▶ This may be good if the gradient of the contrastive term can be computed efficiently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

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

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



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:

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

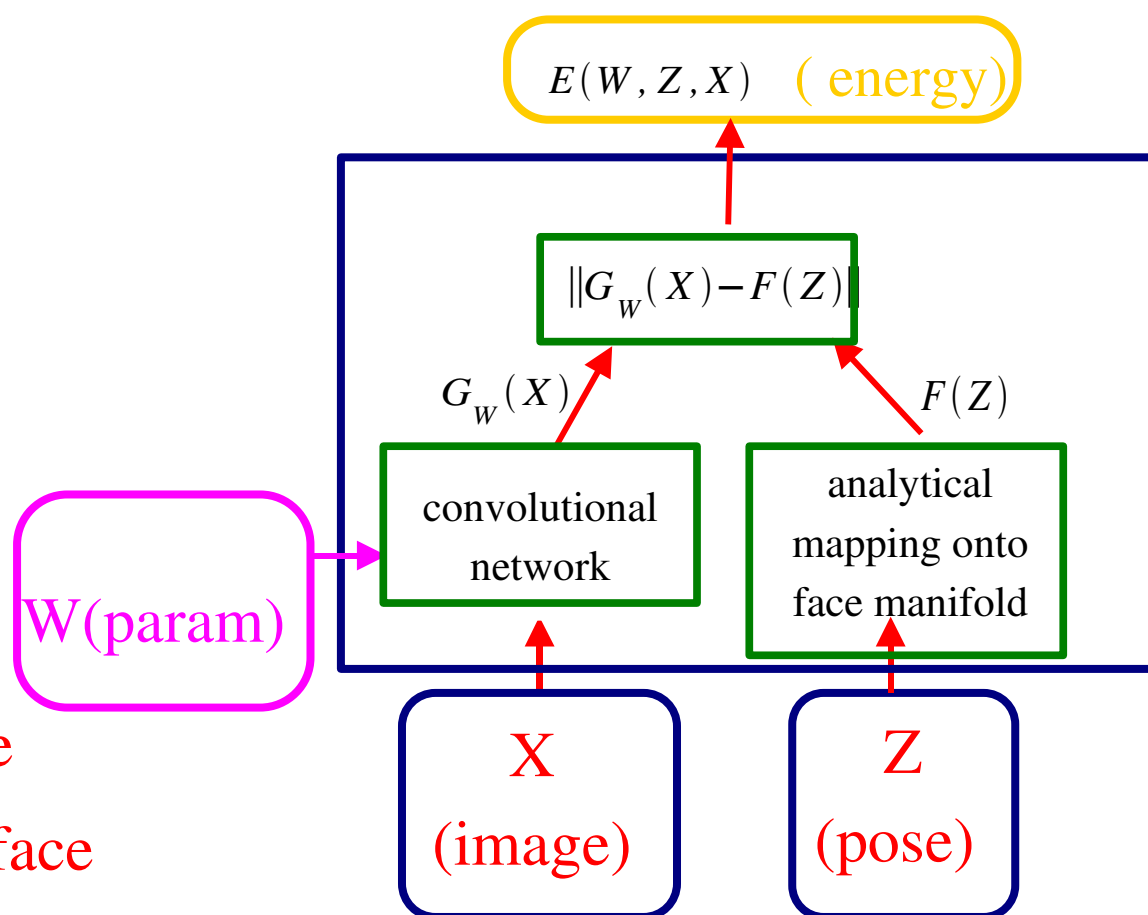
- Reduces to traditional minimization when Beta->infinity

Face Detection and Pose Estimation with a Convolutional EBM

- **Training:** 52,850, 32x32 grey-level images of faces, 52,850 selected 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 .

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

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

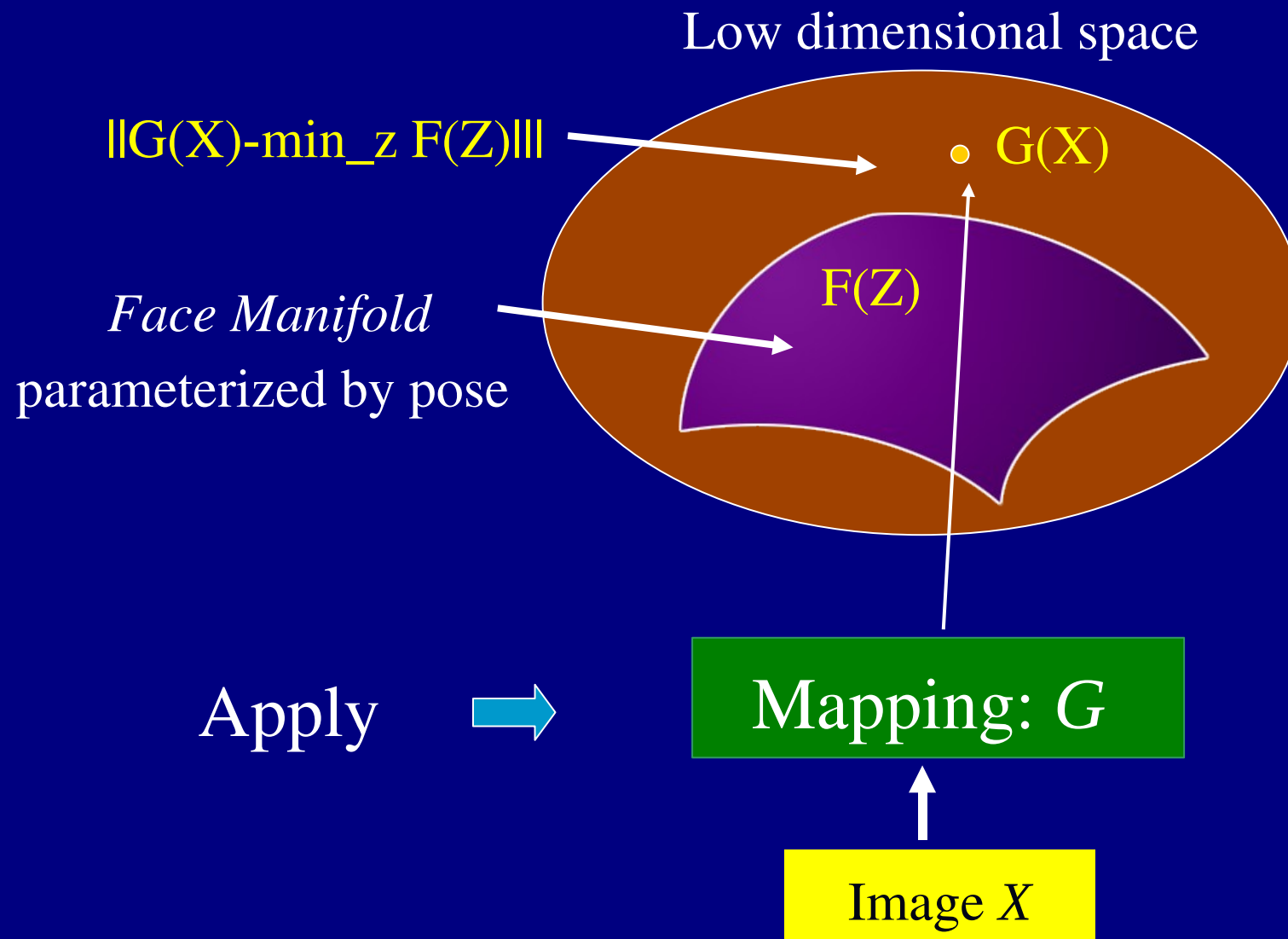


Small $E^*(W, X)$: face

Large $E^*(W, X)$: no face

[Osadchy, Miller, LeCun, NIPS 2004]

Face Manifold



Probabilistic Approach: Density model of joint $P(\text{face}, \text{pose})$

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:

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

Equivalently, minimize the negative log likelihood:

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

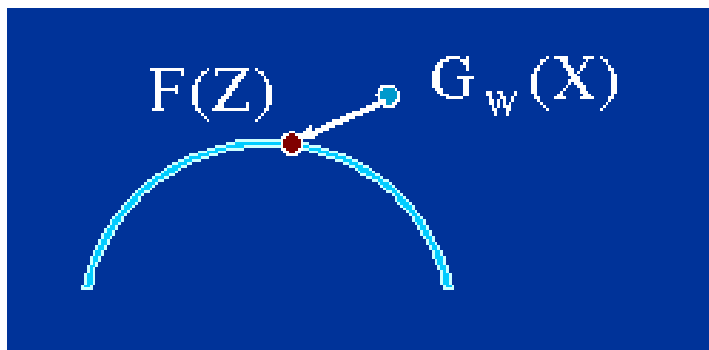


COMPLICATED

Energy-Based Contrastive Loss Function

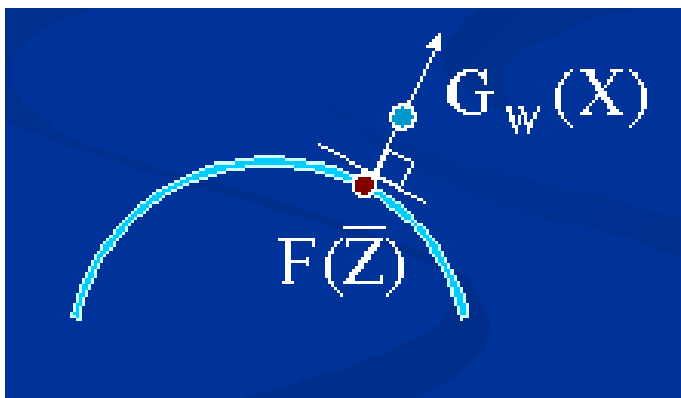
$$\mathcal{L}(W) = \frac{1}{|f + p|} \sum_{X, Z \in \text{faces} + \text{pose}} [L^+(E(W, Z, X))] + L^- \left(\min_{X, Z \in \text{bckgnd}, \text{poses}} E(W, Z, X) \right)$$

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



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

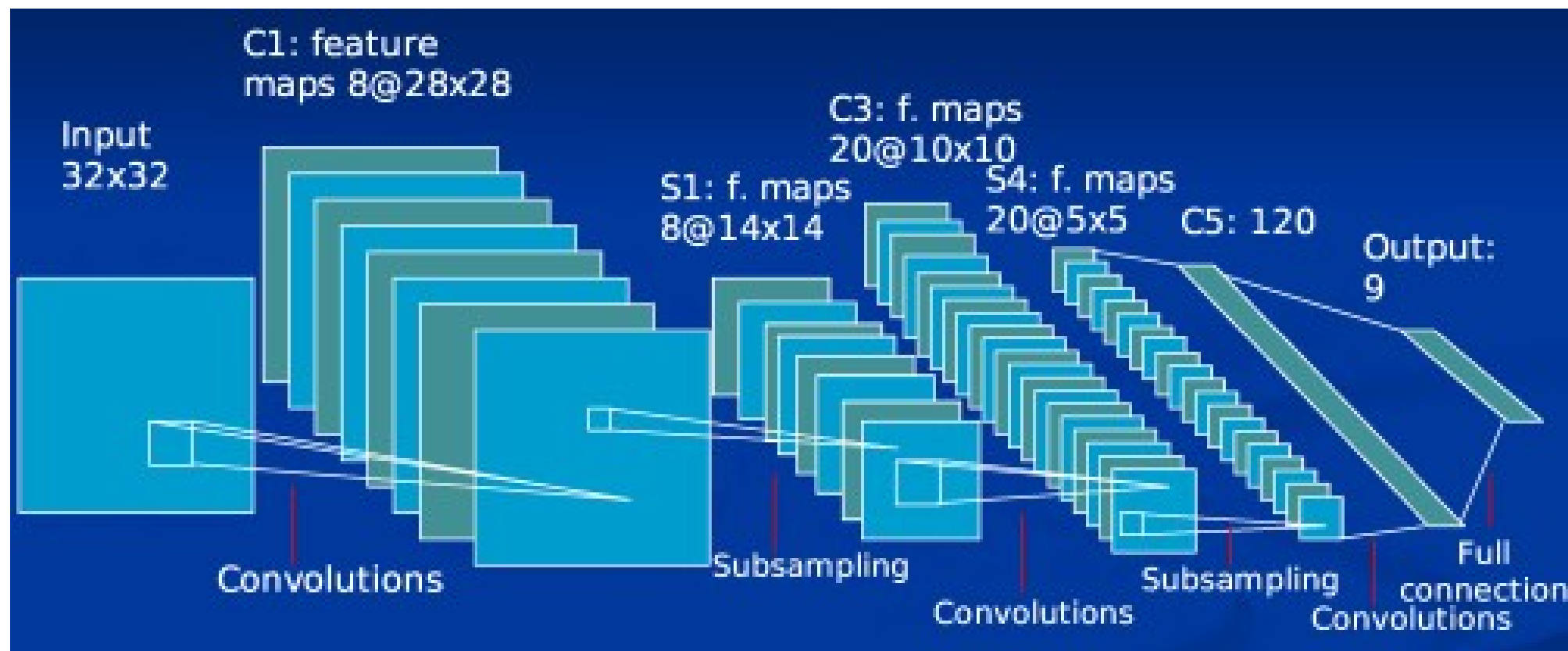
$$L^- \left(\min_{X, Z \in \text{bckgnd}, \text{poses}} E(W, Z, X) \right) = K \exp \left(-\min_{X, Z \in \text{bckgnd}, \text{poses}} \|G_W(X) - F(Z)\| \right)$$



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

Convolutional Network Architecture

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



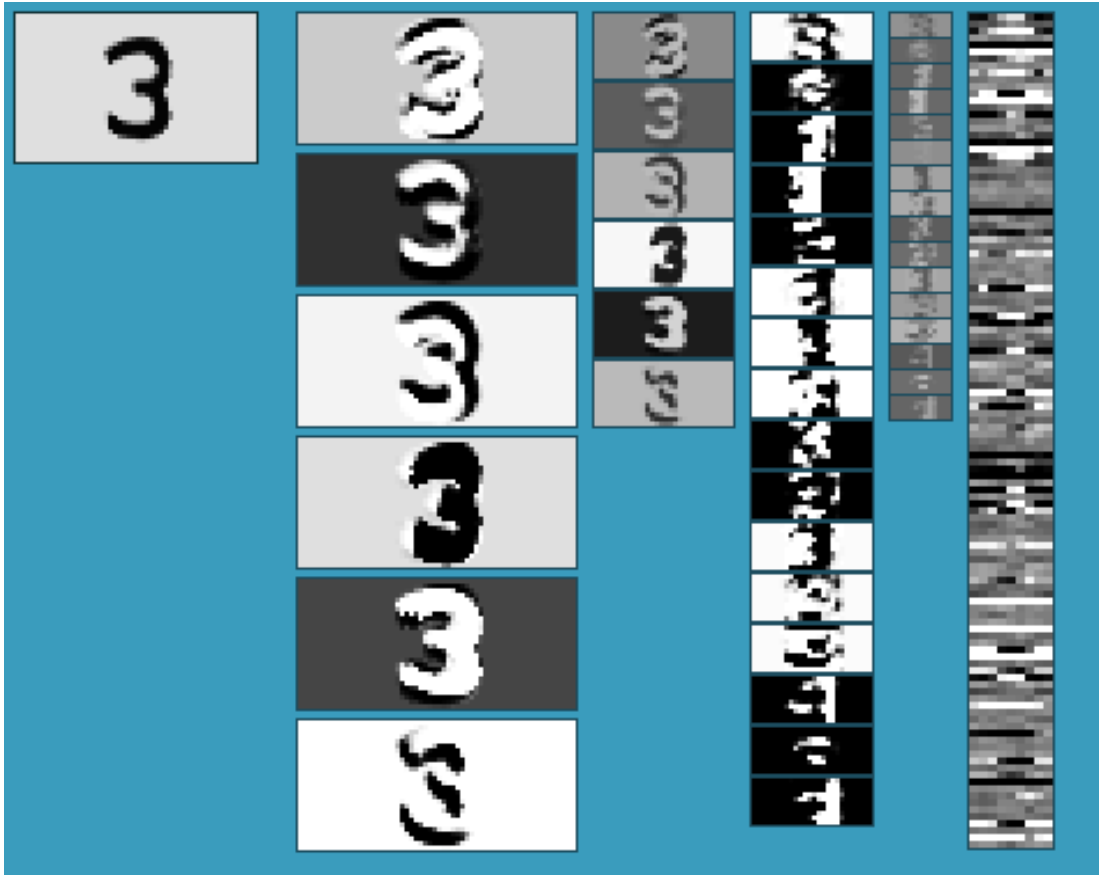
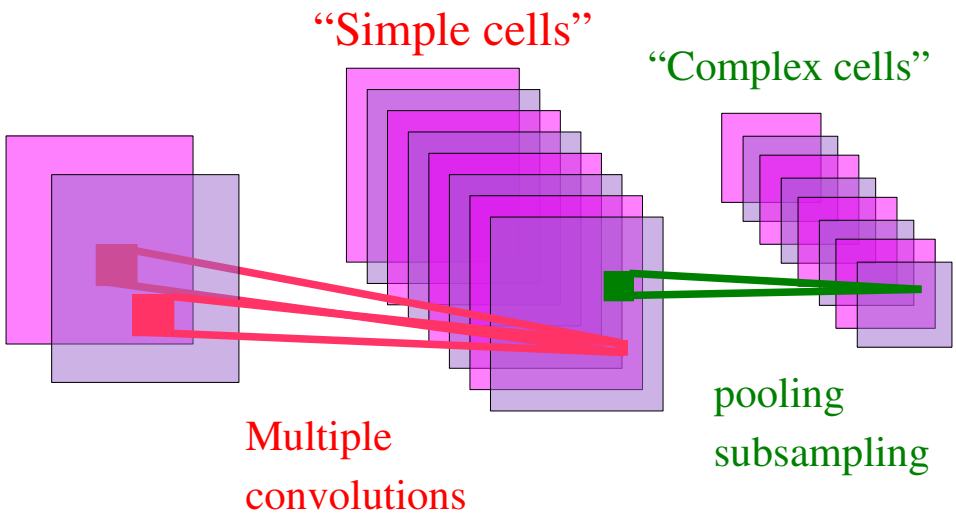
Hierarchy of local filters (convolution kernels),

sigmoid pointwise non-linearities, and spatial subsampling

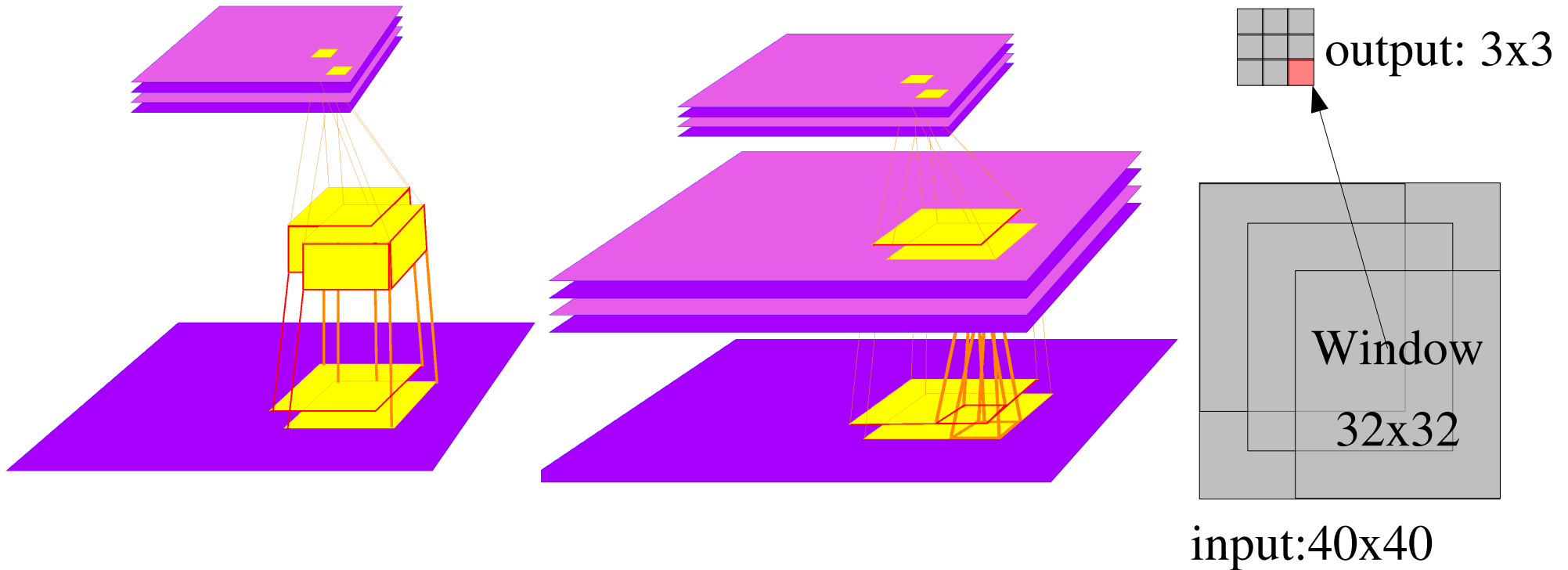
All the filter coefficients are learned with gradient descent (back-prop)

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 $\sqrt{2}$
- Non-maximum suppression with exclusion window

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

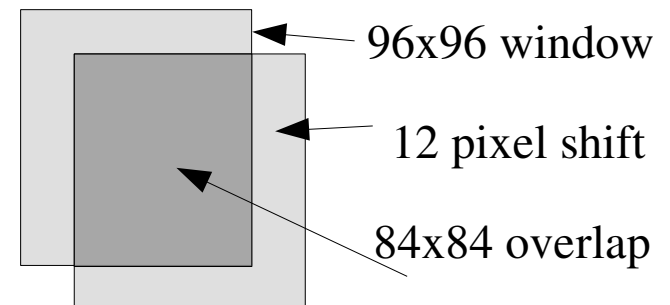
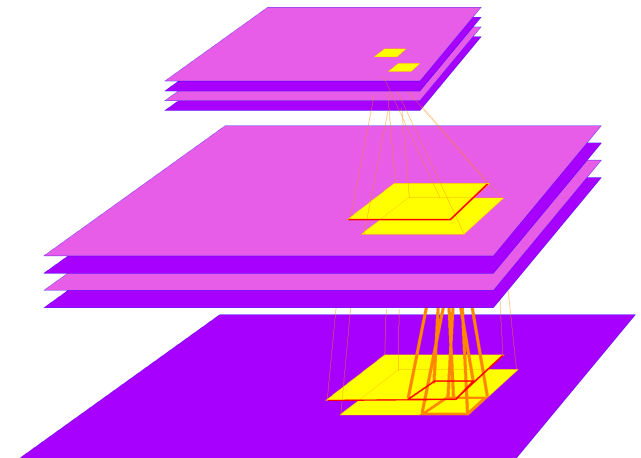
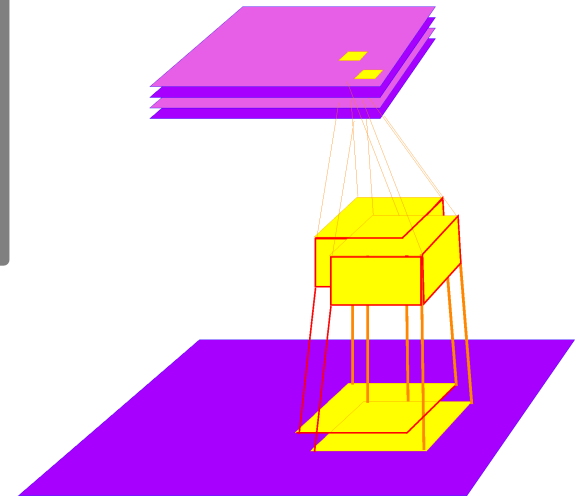
● Computational cost for a non-convolutional detector of the same size, applied every 12 pixels:

● 96x96 -> 4.6 million multiply-accumulate operations

● 120x120 -> 42.0 million multiply-accumulate operations

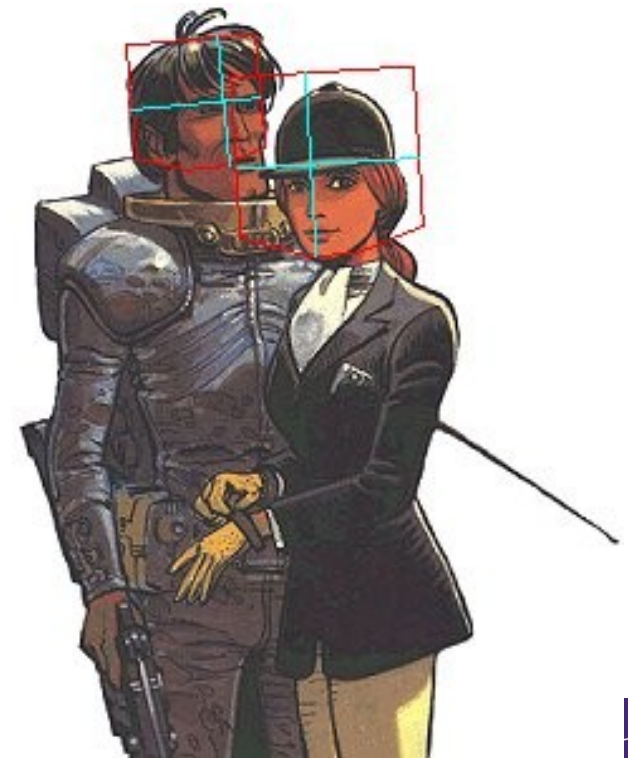
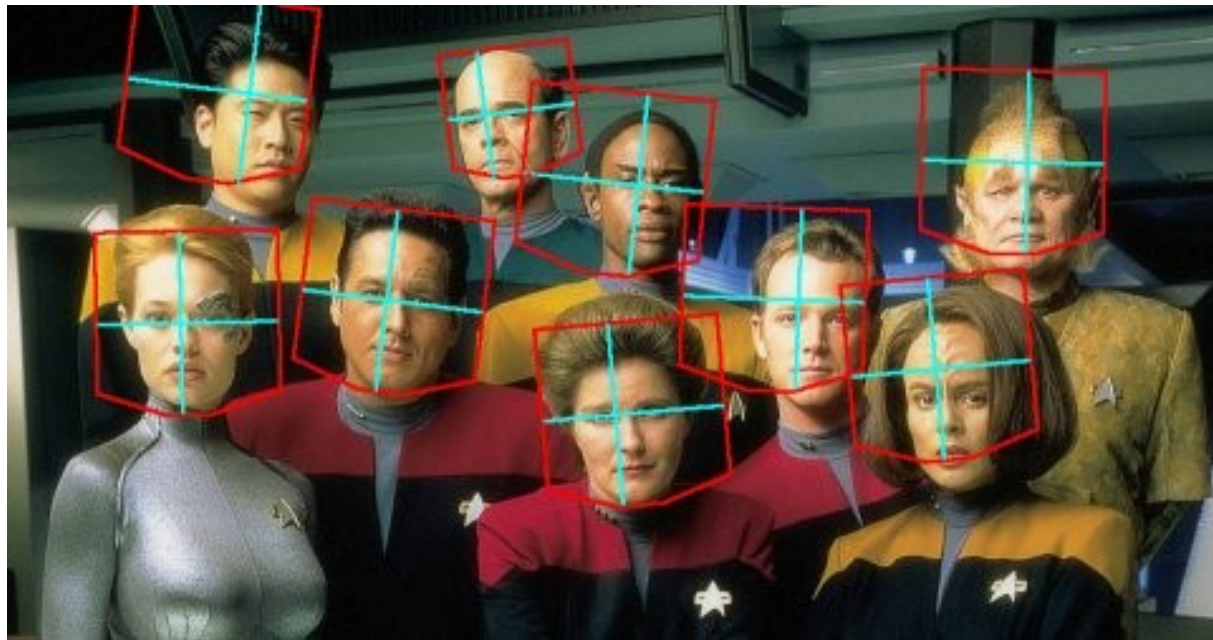
● 240x240 -> 788.0 million multiply-accumulate operations

● 480x480 -> 5,083 million multiply-accumulate operations

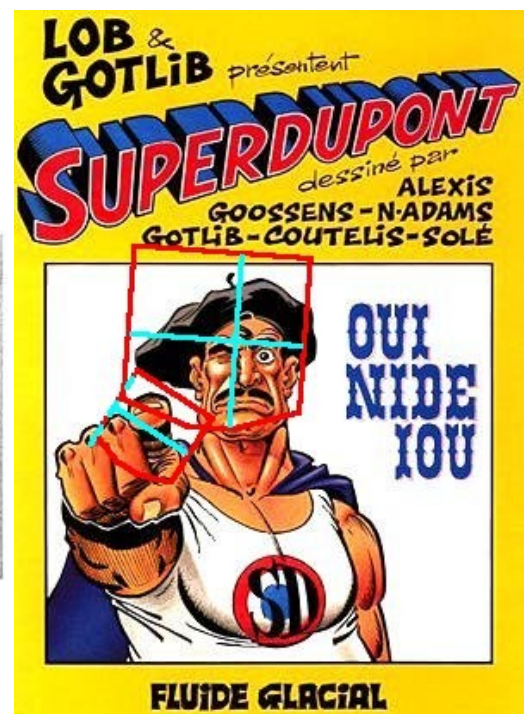
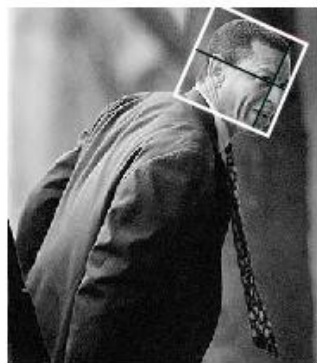
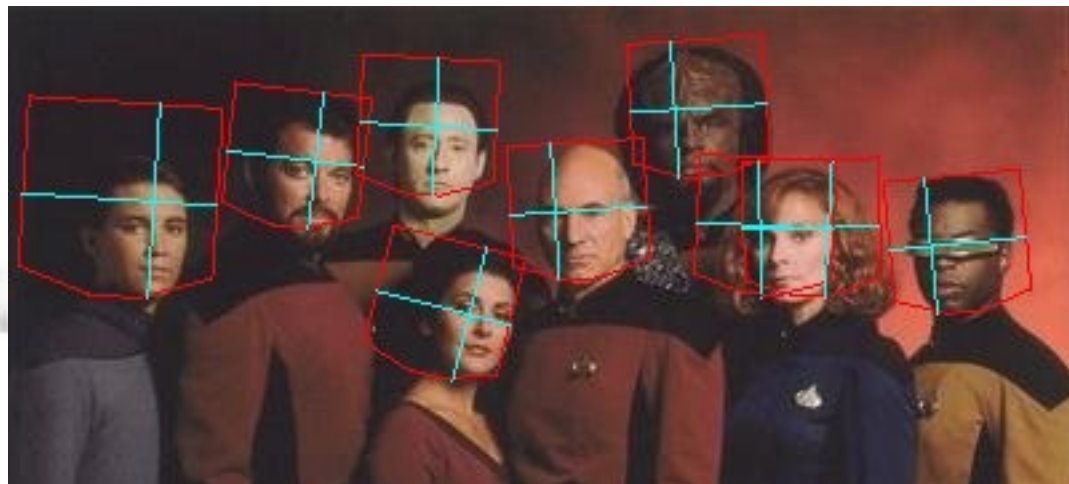


Face Detection: Results

<i>Data Set-></i>	TILTED		PROFILE		MIT+CMU	
	<i>False positives per image-></i>					
	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

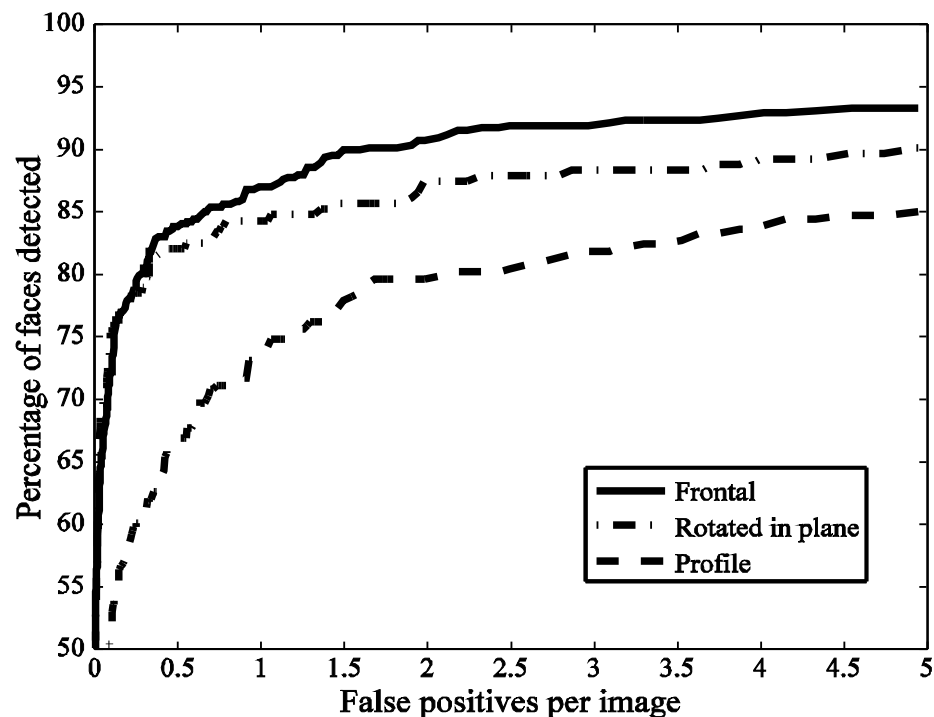


Face Detection with a Convolutional Net

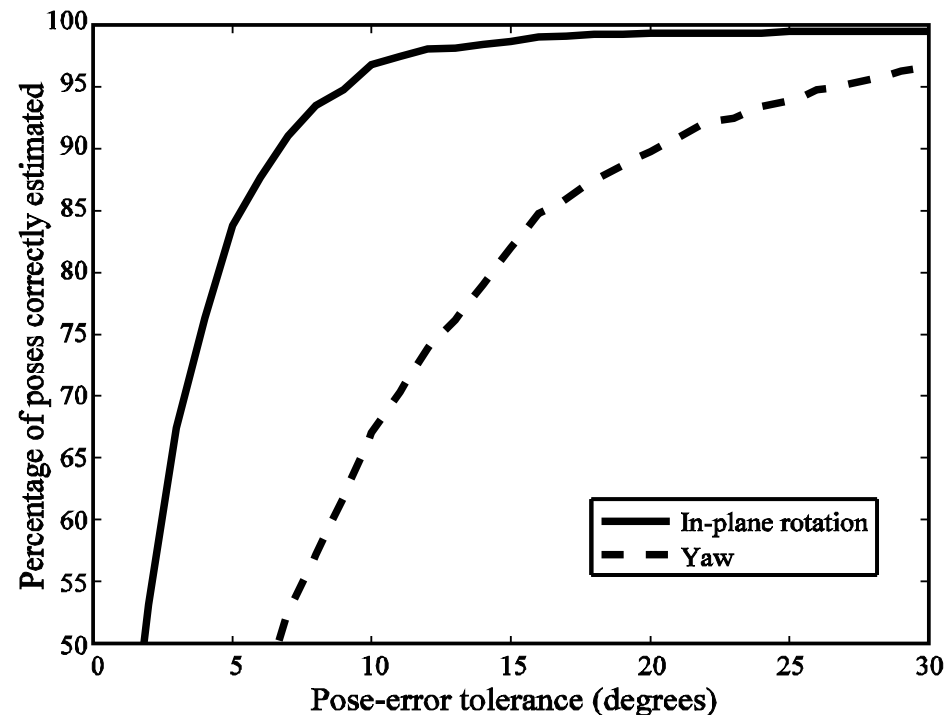


Performance on standard dataset

Detection



Pose estimation

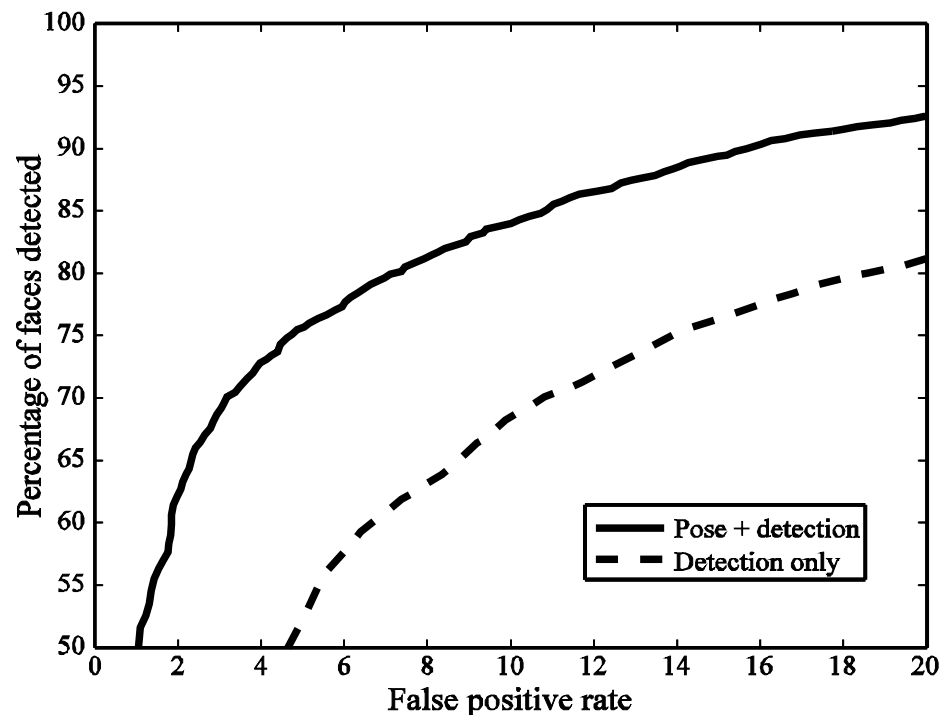


Pose estimation is performed on faces located automatically by the system

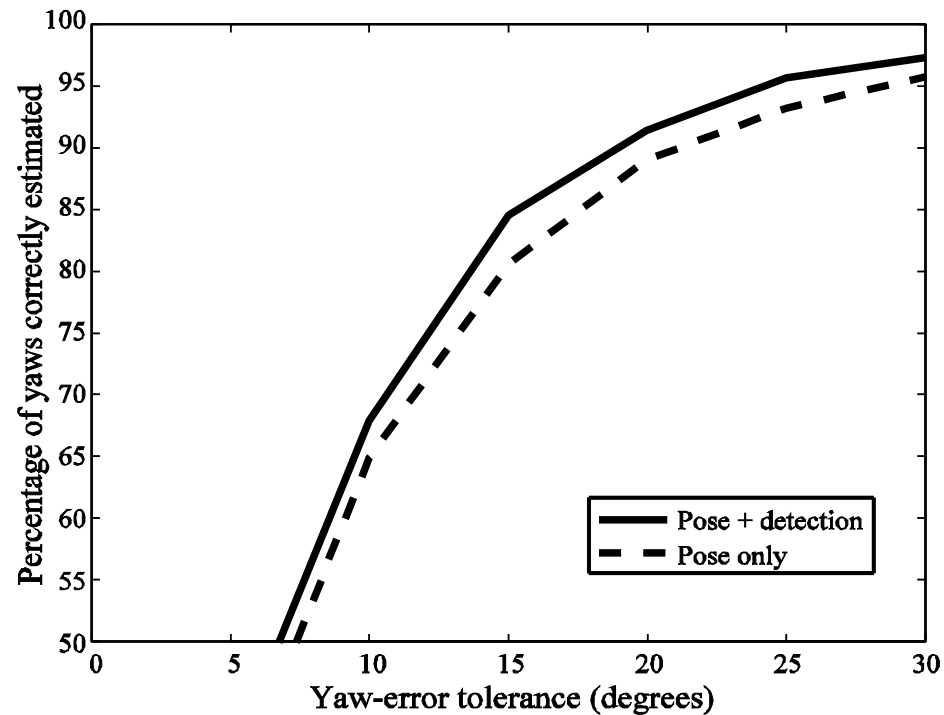
when the faces are localized by hand we get: 89% of yaw and 100% of in-plane rotations within 15 degrees.

Synergy Between Detection and Pose Estimation

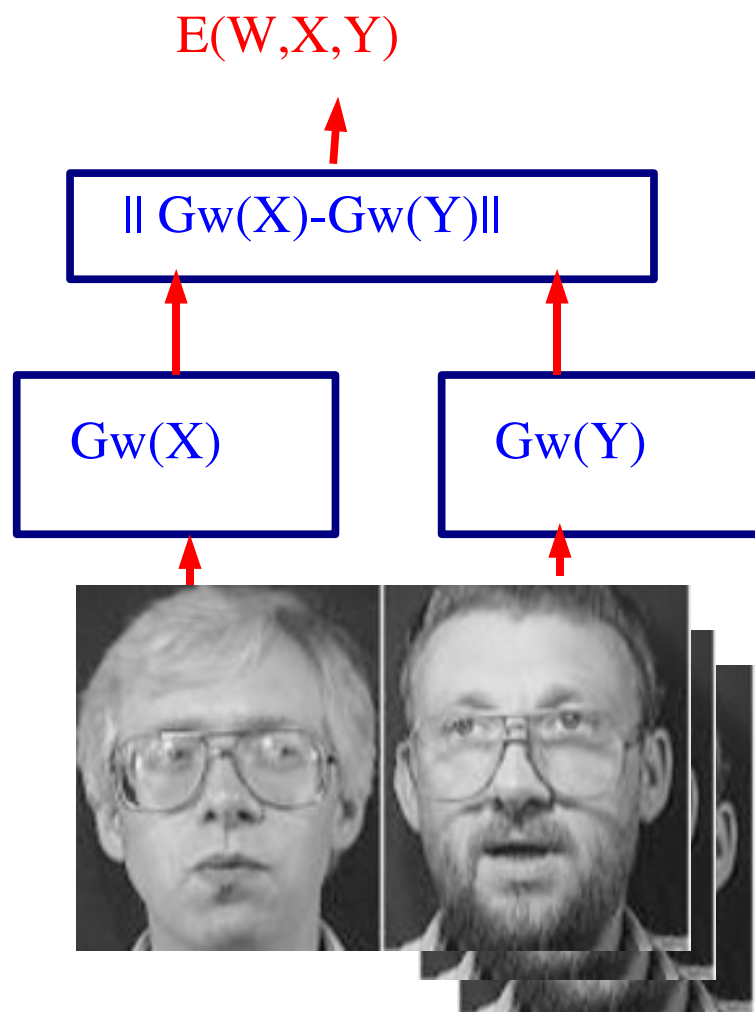
Pose Estimation Improves Detection



Detection improves pose estimation



EBM for Face Recognition



- **X and Y are images**

- **Y is a discrete variable with many possible values**

- ▶ All the people in our gallery

- **Example of architecture:**

- ▶ A function $G(X)$ maps input images into a low-dimensional space in which the Euclidean distance measures dissimilarity.

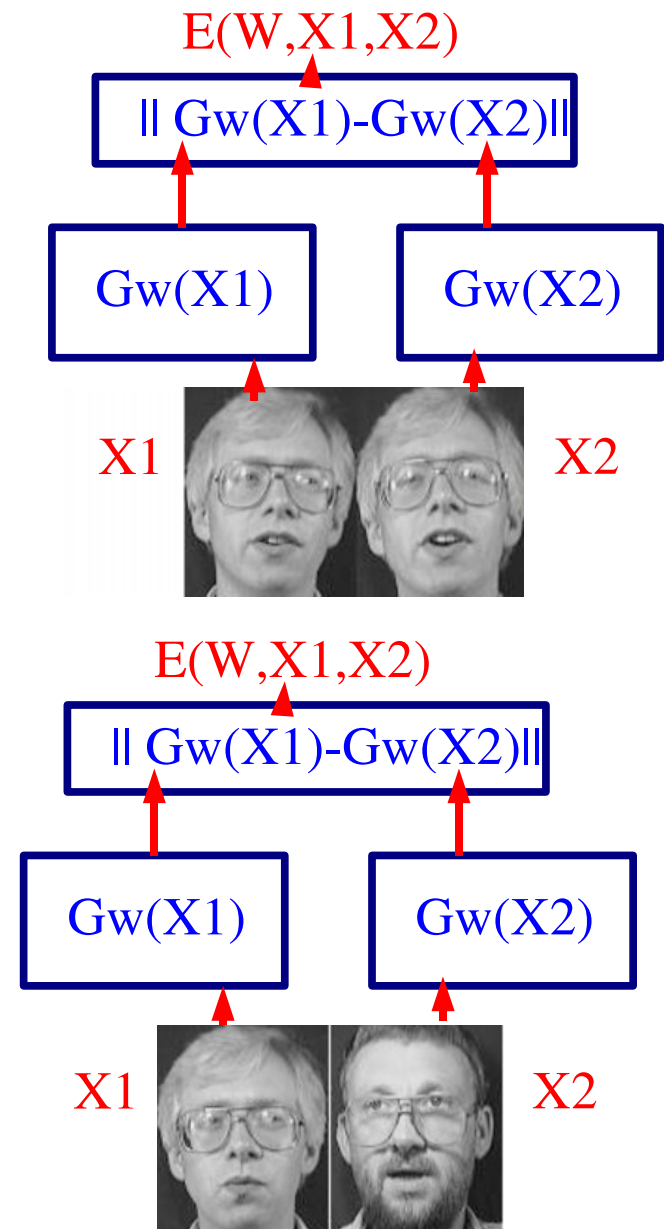
- **Inference:**

- ▶ Find the Y in the gallery that minimizes the energy (find the Y that is most similar to X)
 - ▶ Minimization through exhaustive search.

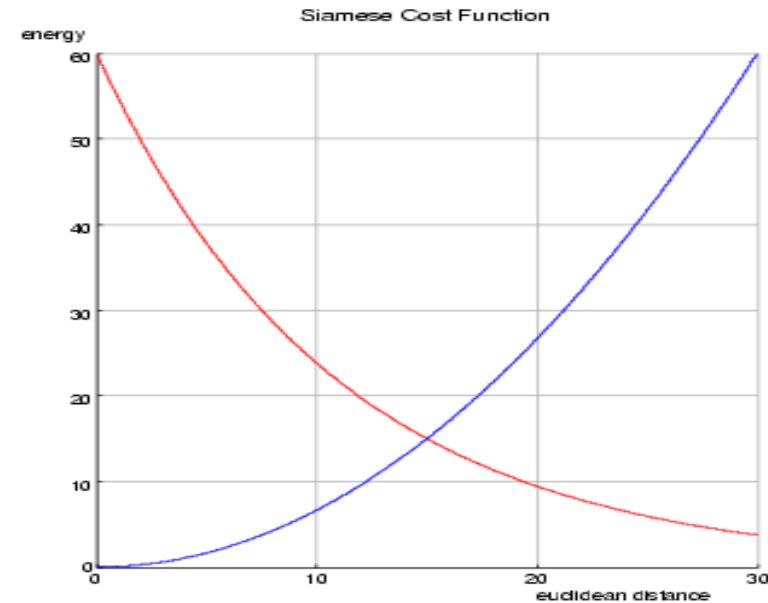
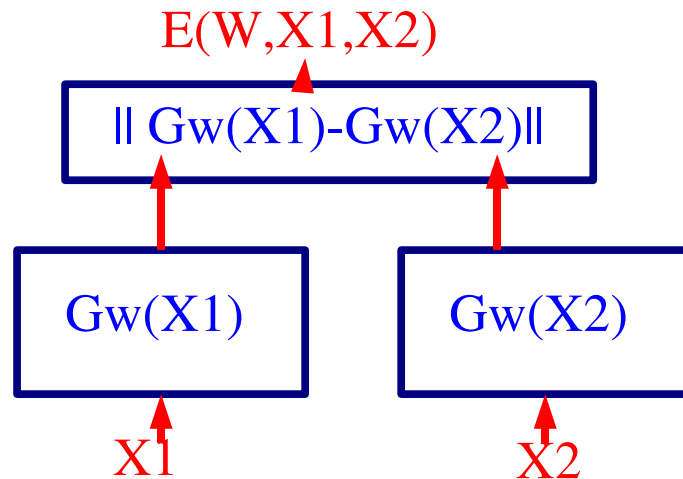
Learning an Invariant Dissimilarity Metric with EBM

[Chopra, Hadsell, LeCun CVPR 2005]

- Training a **parameterized, invariant dissimilarity metric** may be a solution to the **many-category problem**.
- Find a mapping **$G_w(X)$** such that the Euclidean distance **$\|G_w(X1) - G_w(X2)\|$** 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



Loss Function



- **Siamese models:** distance between the outputs of two identical copies of a model.
- **Energy function:** $E(W,X1,X2) = \|Gw(X1)-Gw(X2)\|$
- If $X1$ and $X2$ are from the **same category (genuine pair)**, train the two copies of the model to produce **similar outputs (low energy)**
- If $X1$ and $X2$ are from **different categories (impostor pair)**, train the two copies of the model to produce **different outputs (high energy)**
- **Loss function:** **increasing function of genuine pair energy, decreasing function of impostor pair energy.**

Loss Function

Our Loss function for a single training pair (X1,X2):

$$L(W, X_1, X_2) = (1 - Y) L_G(E_W(X_1, X_2)) + Y L_I(E_W(X_1, X_2))$$

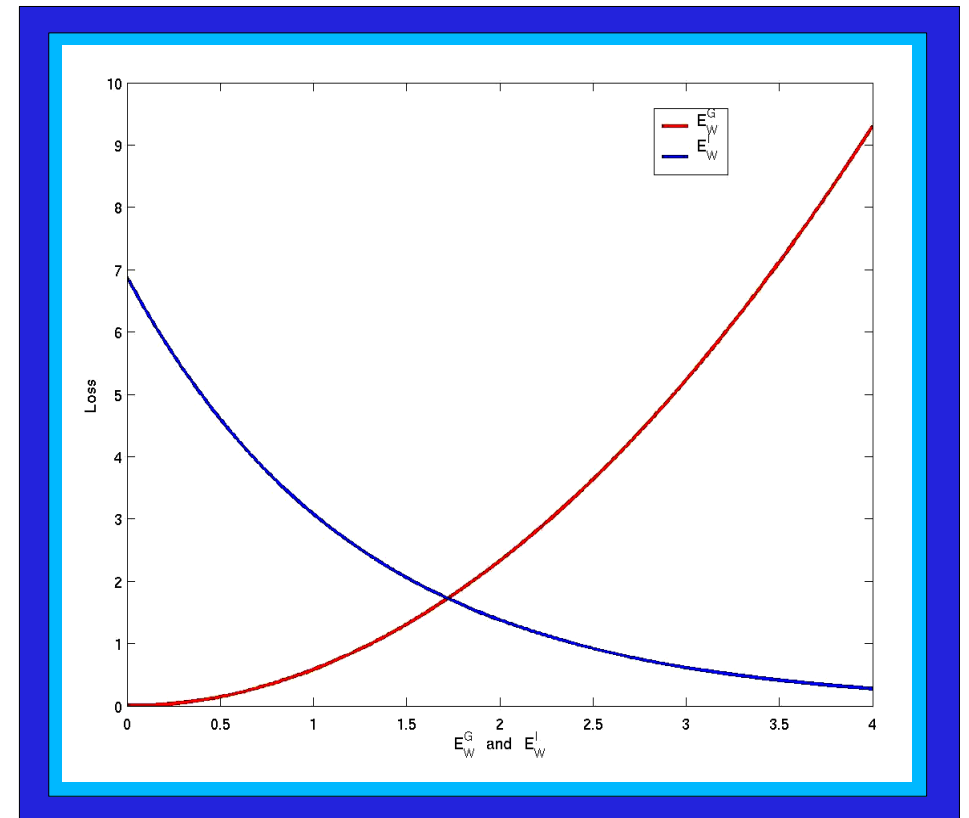
$$= (1 - Y) \frac{2}{R} (E_W(X_1, X_2))^2 + (Y) 2R e^{-2.77 \frac{E_W(X_1, X_2)}{R}}$$

$$E_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_{L1}$$

And R is the largest possible value of

$$E_W(X_1, X_2)$$

Y=0 for a genuine pair, and Y=1 for an impostor pair.



Face Verification datasets: AT&T, FERET, and AR/Purdue

- 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**



Face Verification datasets: AT&T, FERET, and AR/Purdue

- **The FERET dataset.** part of the dataset was used only for training.
- Total subjects: **96**. Images per subject: **6**. Total images: **1122**.
- Images had **high** degree of variation in pose, lighting, expression and head position.
- The images were used for **training only**.
- <http://www.itl.nist.gov/iad/humanid/feret/>



FERET Dataset



Face Verification datasets: AT&T, FERET, and AR/Purdue

- The AR/Purdue dataset
- Total subjects: **136**. Images per subject: **26**. Total images: **3536**.
- Each subject has 2 sets of 13 images taken 14 days apart.
- Images had **very high** degree of variation in pose, lighting, expression and position. Within each set of 13, there are 4 images with expression variation, 3 with lighting variation, 3 with dark sun glasses and lighting variation, and 3 with face obscuring scarfs and lighting variation.
- Images from **96** subjects were used for training. The remaining **40** subjects were used for testing.
- Training set drawn from: **64896** genuine and **6165120** impostor pairs.
- Test set drawn from: **27040** genuine and **1054560** impostor pairs.
- http://rv11.ecn.purdue.edu/aleix/aleix_face_DB.html



Face Verification dataset: AR/Purdue



Preprocessing

The 3 datasets each required a small amount of preprocessing.

FERET: Cropping, subsampling, and centering (see below)

AR/PURDUE: Cropping and subsampling

AT&T: Subsampling only



crop



subsample



center



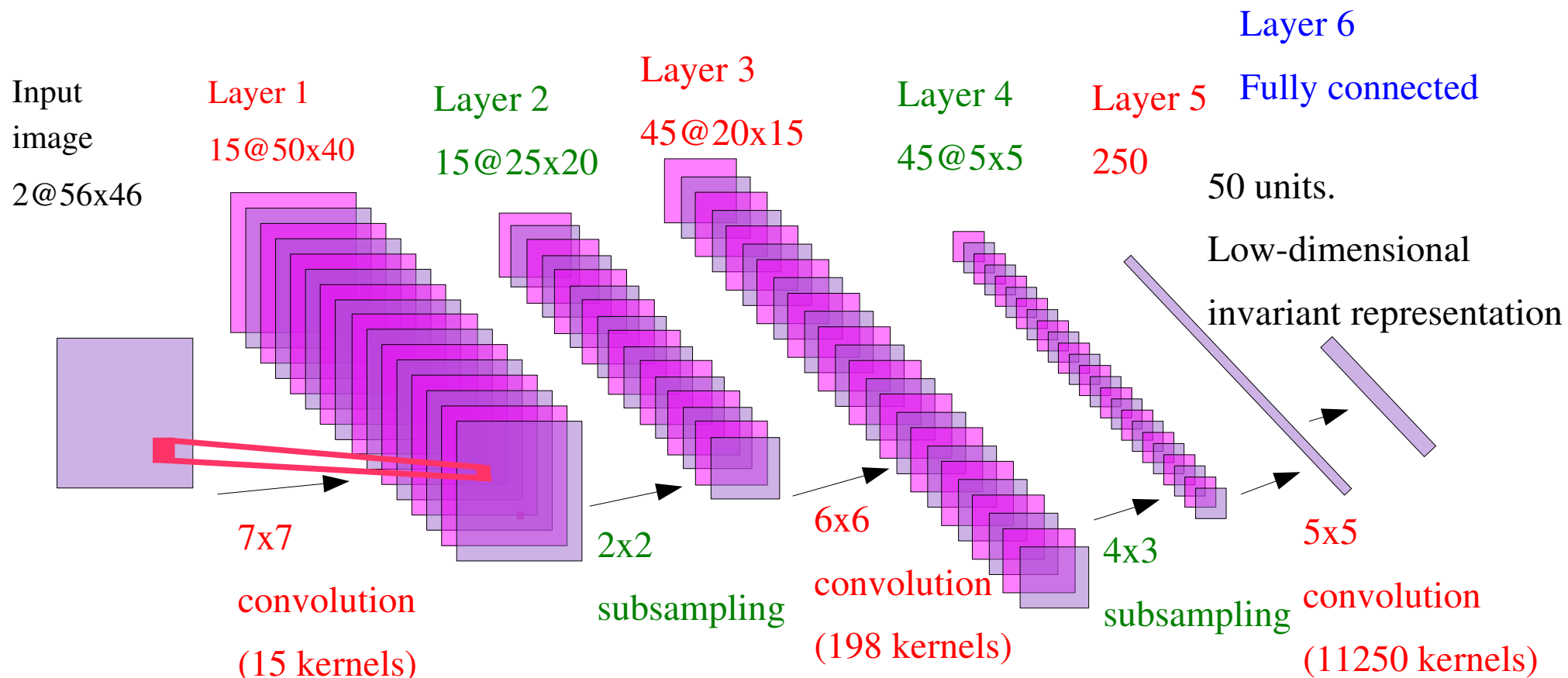
Centering with a Gaussian-blurred face template

- Coarse centering was done on the FERET database images
1. Construct a template by blurring a well-centered face.
 2. Convolve the template with an uncentered image.
 3. Choose the 'peak' of the convolution as the center of the image.



Architecture for the Mapping Function $G_w(X)$

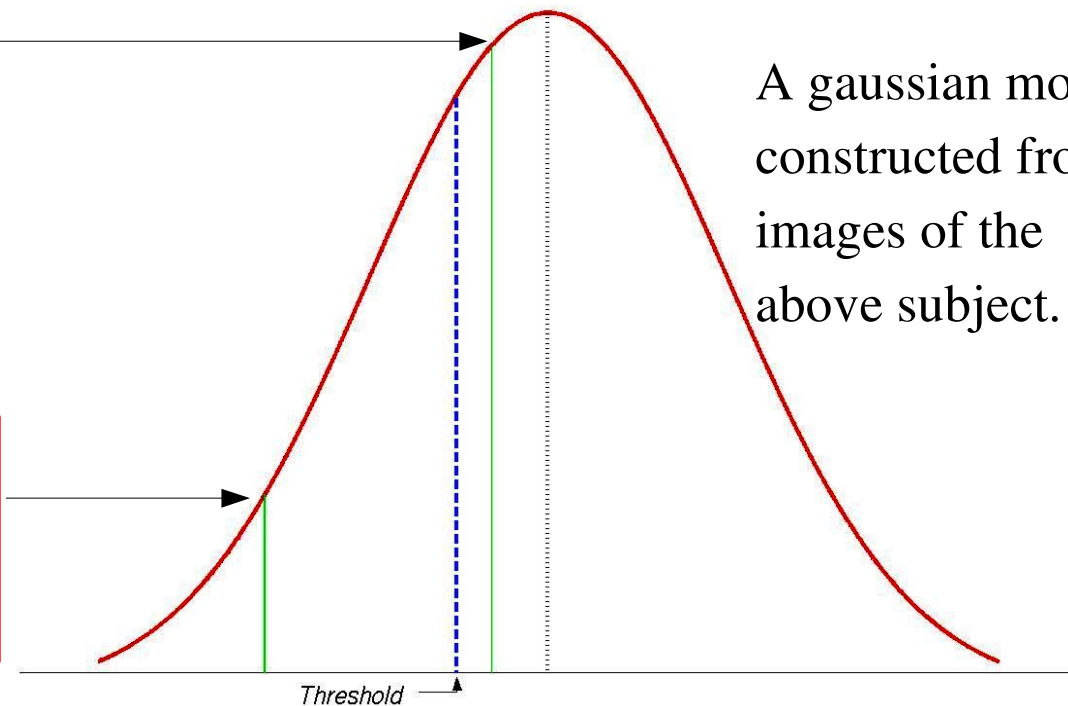
Convolutional net



Internal state for genuine and impostor pairs



Gaussian Face Model in the output space



A gaussian model constructed from 5 images of the above subject.

Dataset for Verification

tested on AT&T and AR/Purdue

AT&T dataset

Number of subjects: 5
Images/subject: 10
Images/Model: 5
Total test size: 5000
Number of Genuine: 500
Number of Impostors: 4500

Purdue/AR dataset

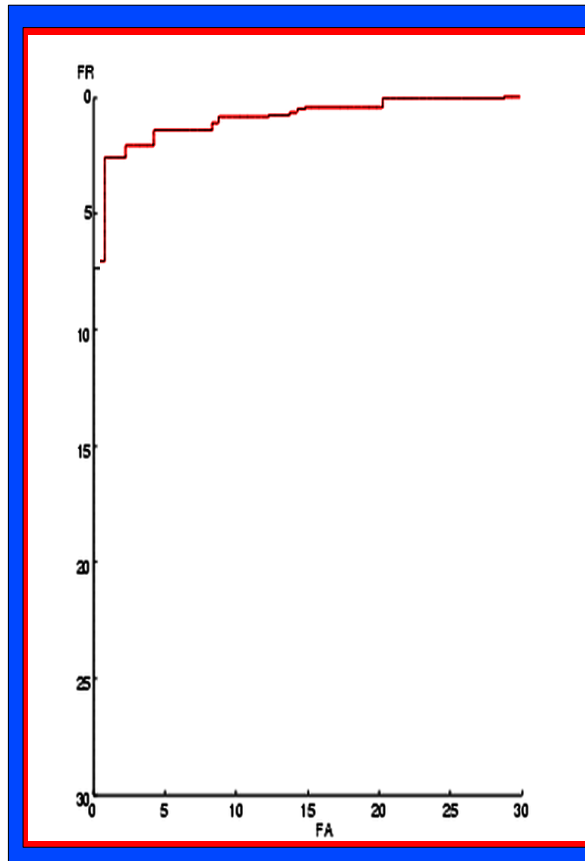
Number of subjects: 40
Images/subject: 26
Images/Model: 13
Total test size: 5000
Number of Genuine: 500
Number of Impostors: 4500

Verification Results

The AT&T dataset

False Accept False Reject

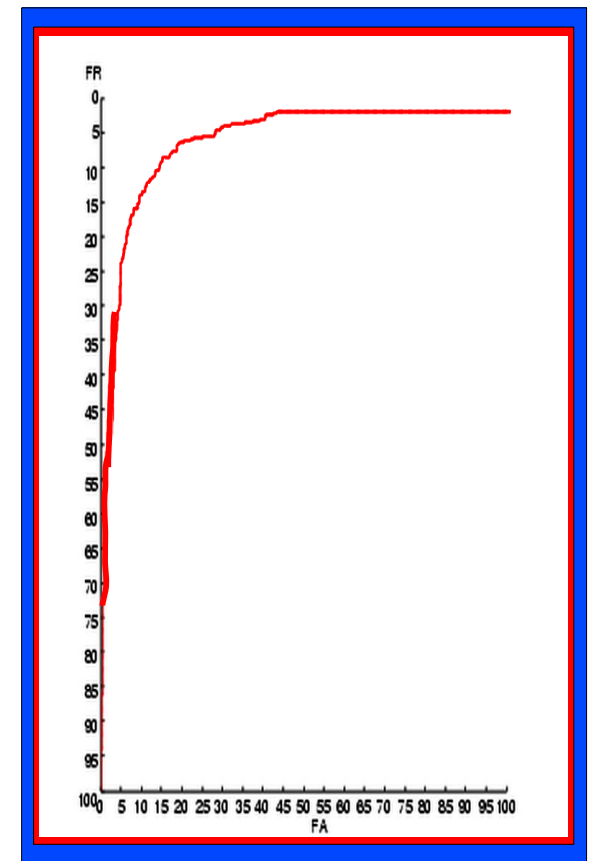
10.00% 0.00%
7.50% 1.00%
5.00% 1.00%



The AR/Purdue dataset

False Accept False Reject

10.00% 11.00%
7.50% 14.60%
5.00% 19.00%



Classification Examples

Example: Correctly classified genuine pairs



energy: 0.3159



energy: 0.0043



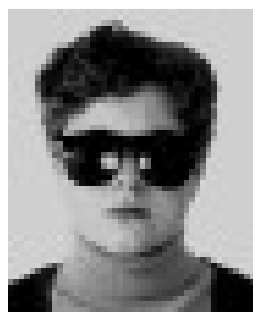
energy: 0.0046



Example: Correctly classified impostor pairs



energy: 20.1259



energy: 32.7897



energy: 5.7186



Example: Mis-classified pairs



energy: 10.3209



energy: 2.8243

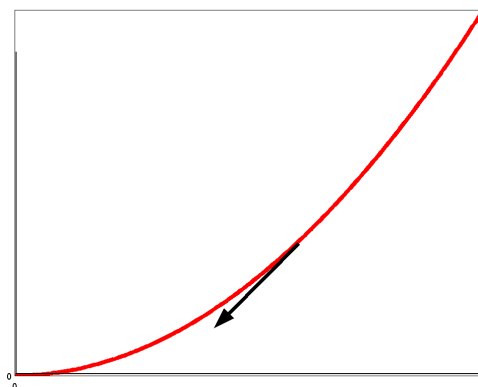


Internal State

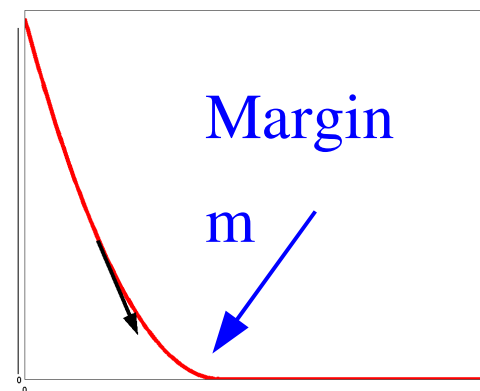


A similar idea for Learning a Manifold with Invariance Properties

$$L_{\text{similar}} = \frac{1}{2} D_w^2$$

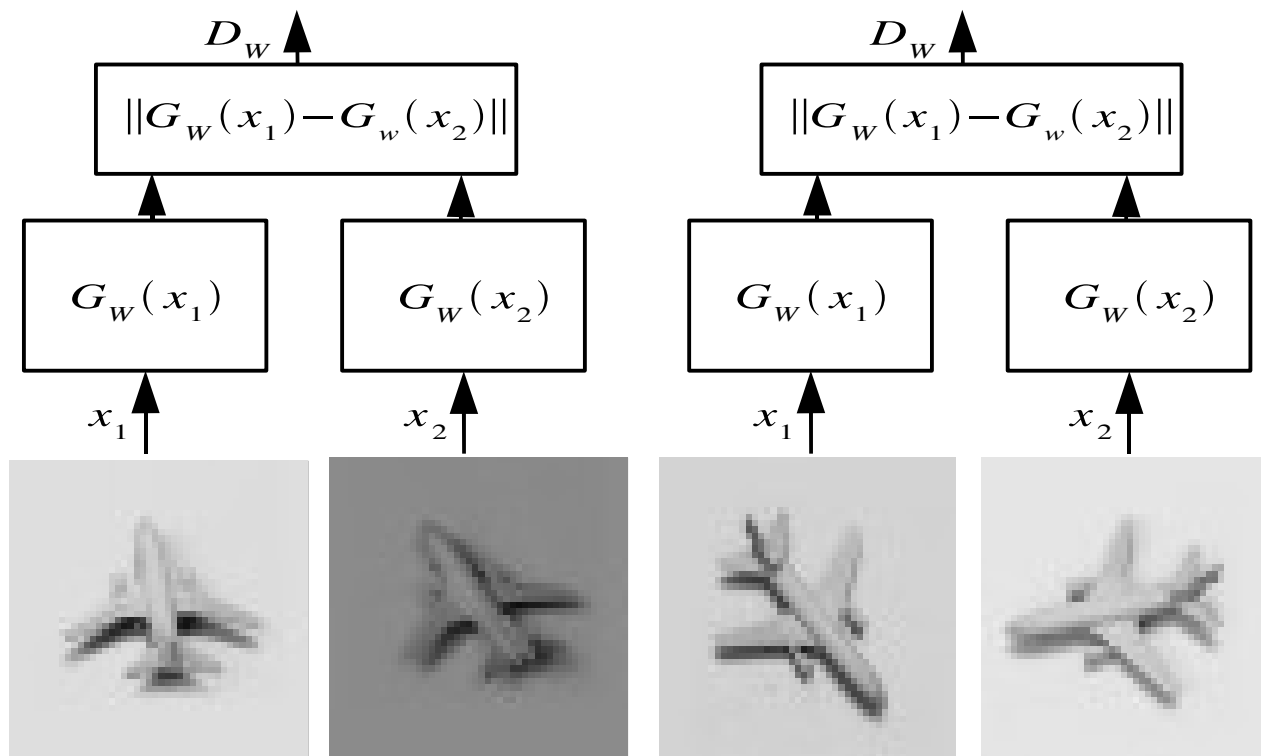


$$L_{\text{dissimilar}} = \frac{1}{2} \{ \max(0, m - D_w) \}^2$$

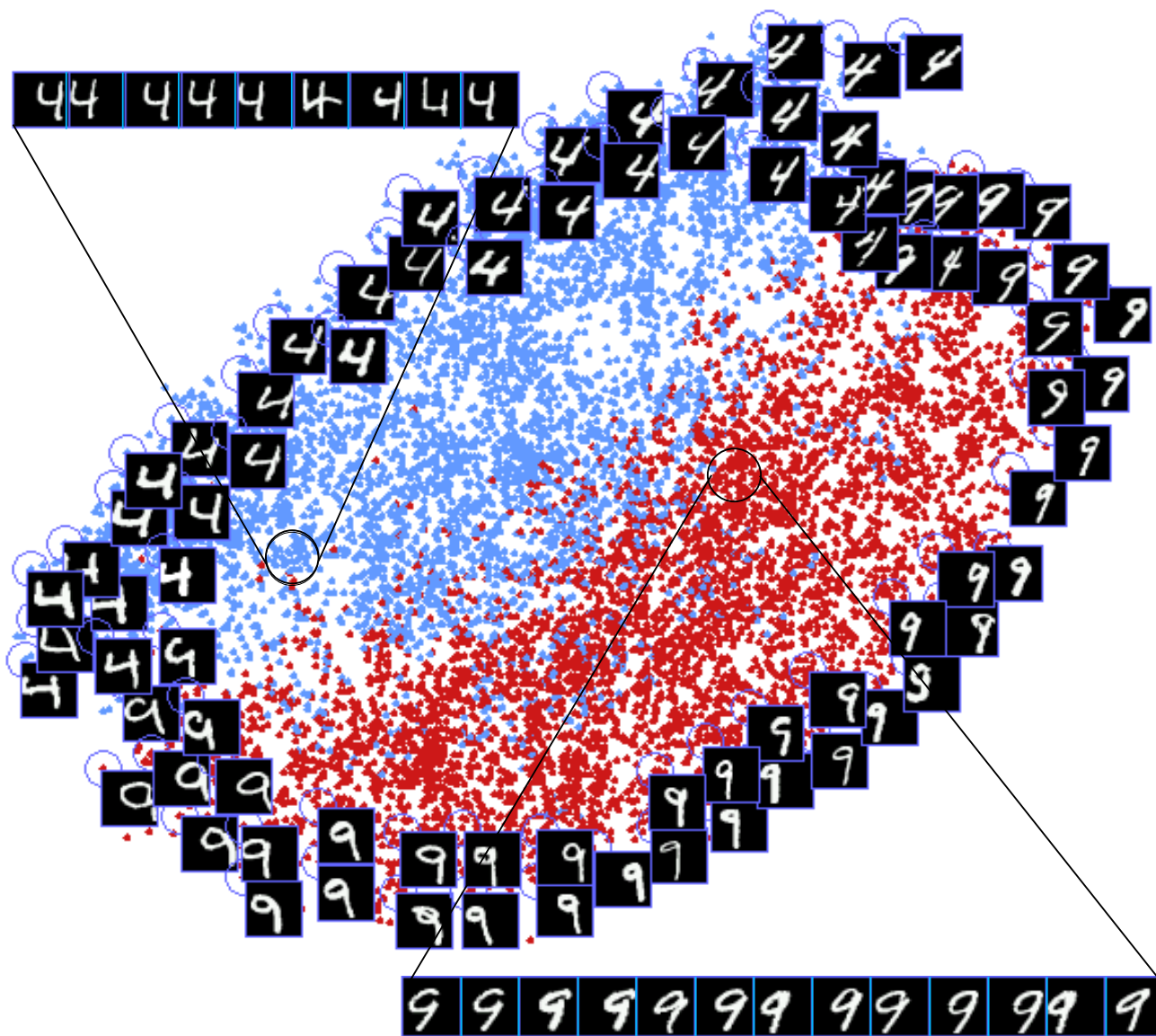


Loss function:

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



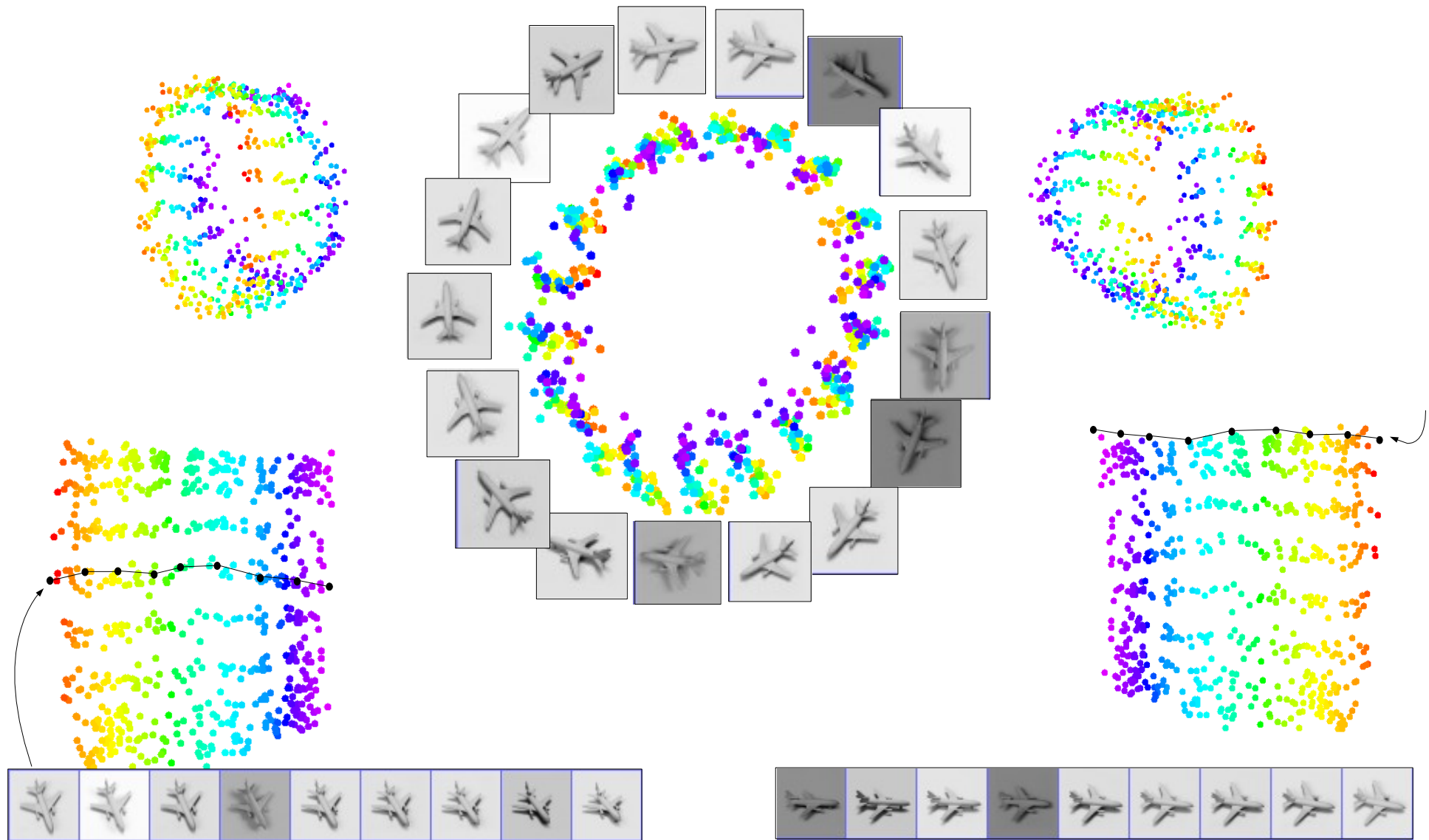
A Manifold with Invariance to Shifts



- 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



Efficient Inference: Energy-Based Factor Graphs

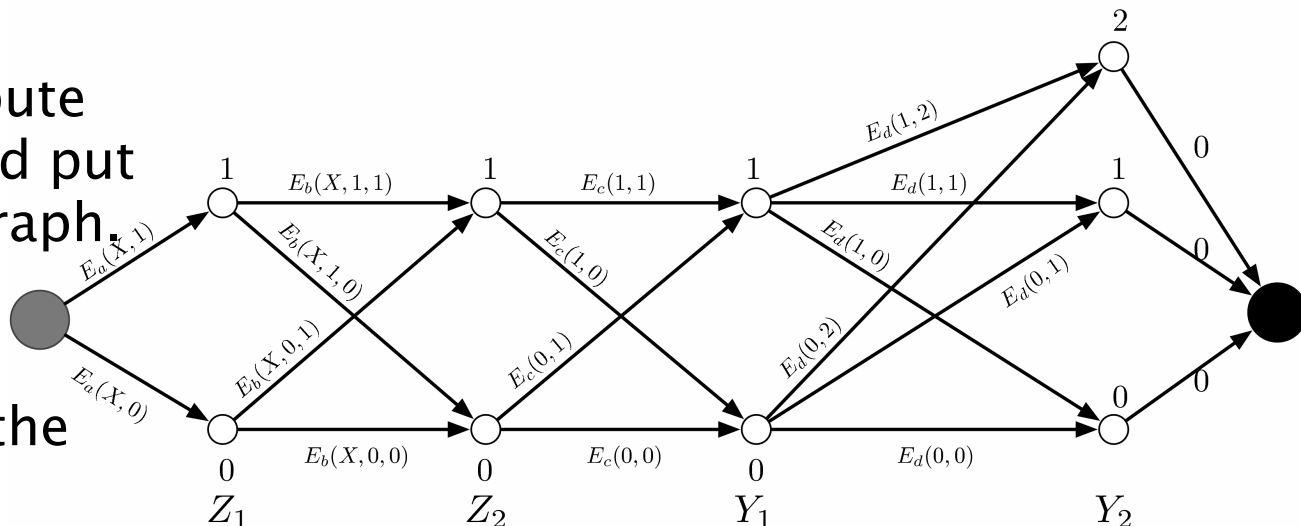
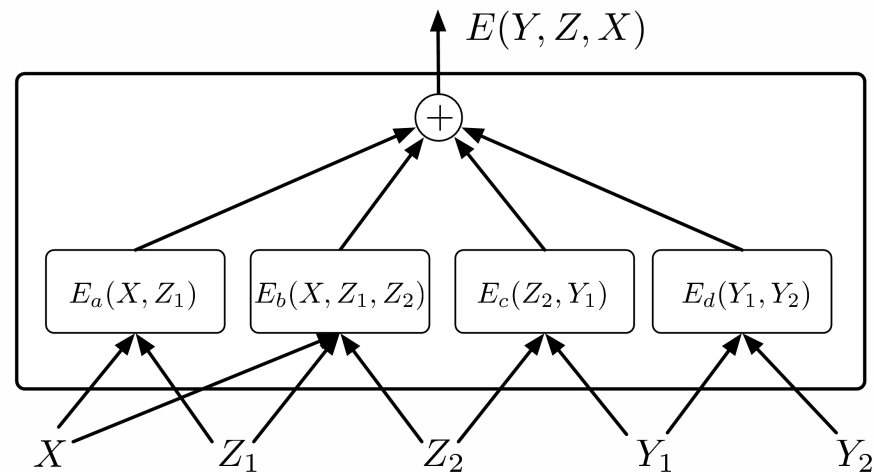
- Graphical models have brought us efficient inference algorithms, such as belief propagation and its numerous variations.
- Traditionally, graphical models are viewed as probabilistic models
- At first glance, it seems difficult to dissociate graphical models from the probabilistic view
- Energy-Based Factor Graphs are an extension of graphical models to non-probabilistic settings.
- An EBFG is an energy function that can be written as a sum of “factor” functions that take different subsets of variables as inputs.

Efficient Inference: Energy-Based Factor Graphs

• The energy is a sum of “factor” functions

• Example:

- ▶ Z_1, Z_2, Y_1 are binary
- ▶ Z_2 is ternary
- ▶ A naïve exhaustive inference would require $2 \times 2 \times 2 \times 3$ energy evaluations (= 96 factor evaluations)
- ▶ BUT: E_a only has 2 possible input configurations, E_b and E_c have 4, and E_d 6.
- ▶ Hence, we can precompute the 16 factor values, and put them on the arcs in a graph.
- ▶ A path in the graph is a config of variable
- ▶ The cost of the path is the energy of the config

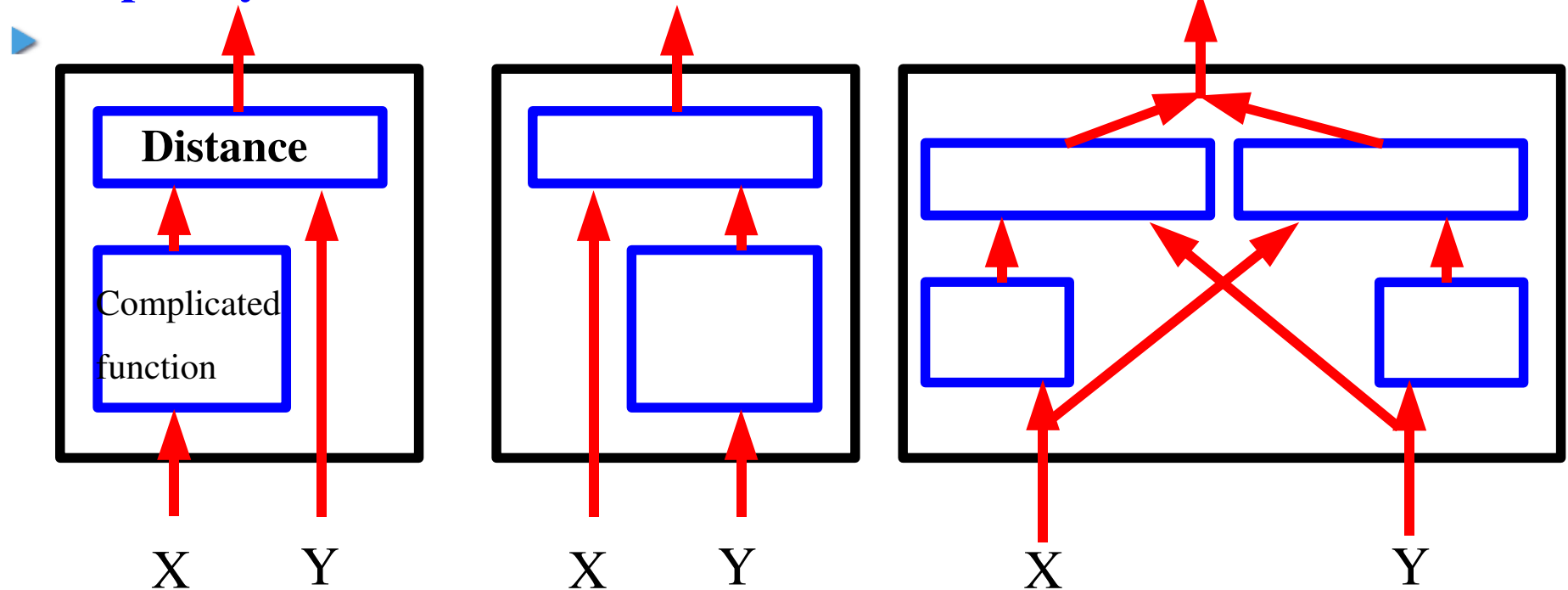


Energy-Based Belief Prop

- The previous picture shows a chain graph of factors with 2 inputs.
- The extension of this procedure to trees, with factors that can have more than 2 inputs the “min-sum” algorithm (a non-probabilistic form of belief propagation)
- Basically, it is the sum-product algorithm with a different semi-ring algebra (min instead of sum, sum instead of product), and no normalization step.
 - ▶ [Kschischang, Frey, Loeliger, 2001][McKay's book]

Feed-Forward, Causal, and Bi-directional Models

- EBFG are all “undirected”, but the architecture determines the complexity of the inference in certain directions



Feed-Forward

- Predicting Y from X is easy
- Predicting X from Y is hard

“Causal”

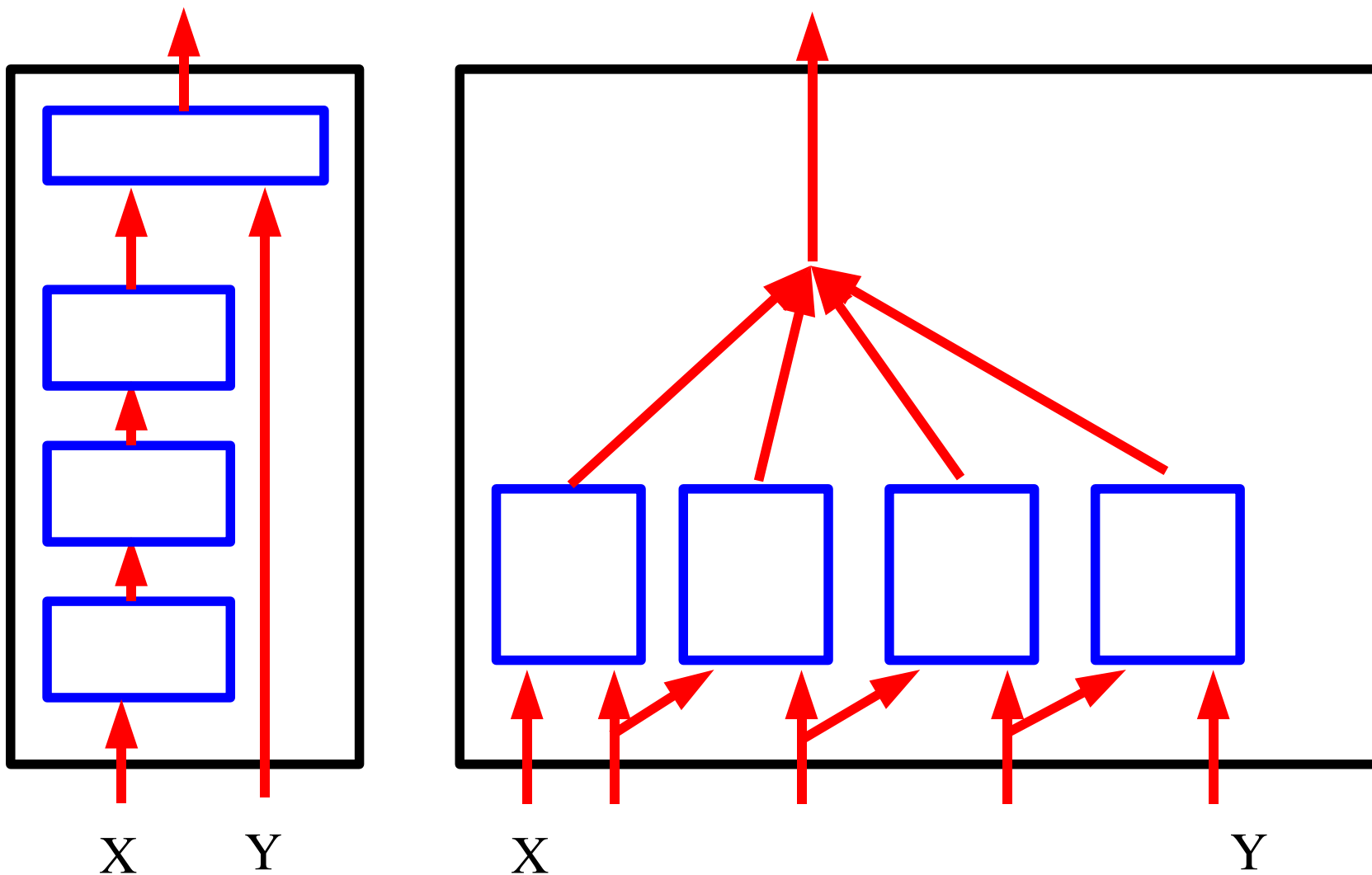
- Predicting Y from X is hard
- Predicting X from Y is easy

Bi-directional

- X $\rightarrow$ Y and Y $\rightarrow$ X are both hard if the two factors don't agree.
- They are both easy if the factors agree

Two types of “deep” architectures

- Factors are deep / graph is deep



Shallow Factors / Deep Graph

Linearly Parameterized Factors

with the NLL Loss :

- ▶ Lafferty's Conditional Random Field

with Hinge Loss:

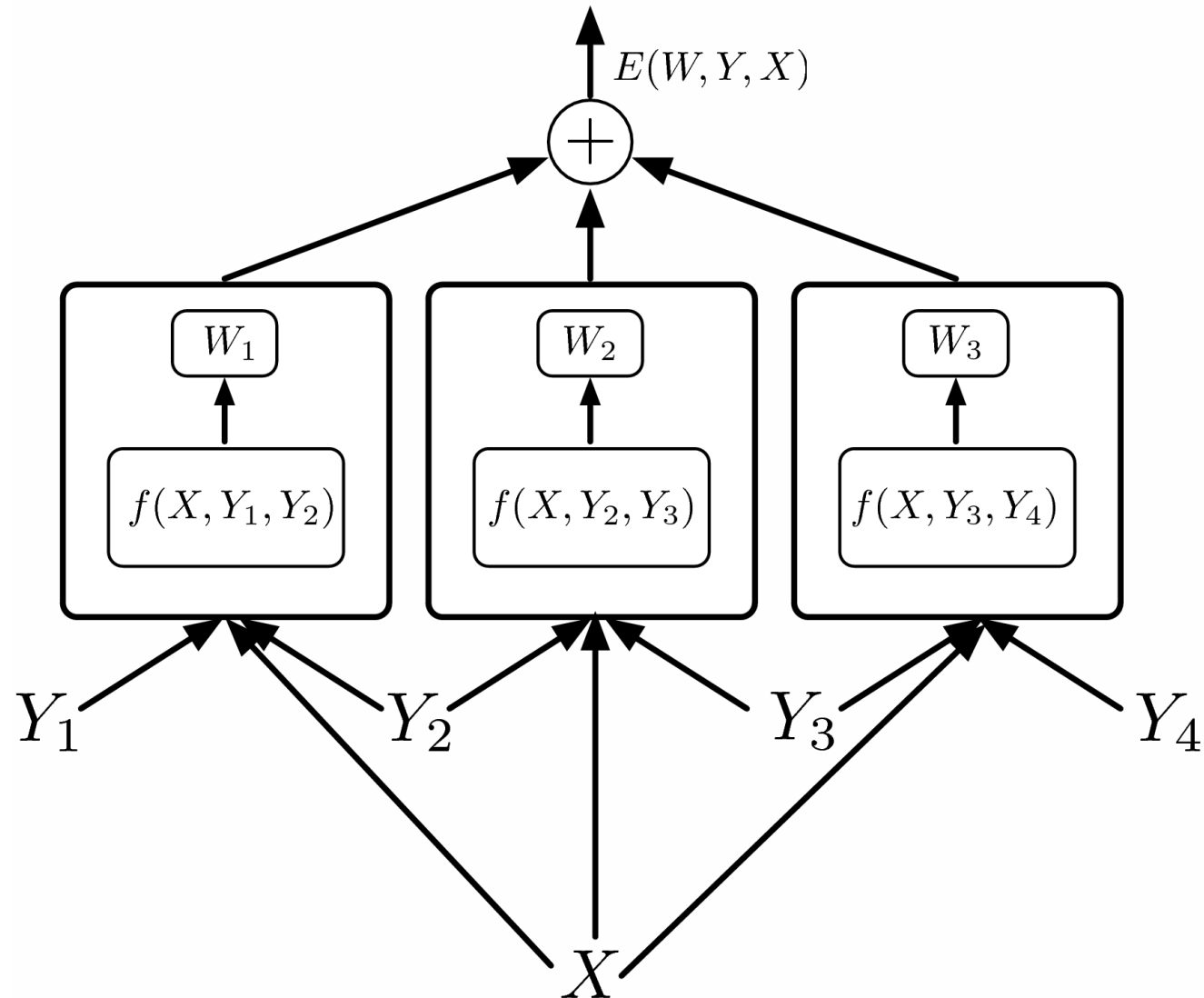
- ▶ Taskar's Max Margin Markov Nets

with Perceptron Loss

- ▶ Collins's sequence labeling model

With Log Loss:

- ▶ Altun/Hofmann sequence labeling model



Deep Factors / Deep Graph: ASR with TDNN/DTW

- Trainable Automatic Speech Recognition system with convolutional nets (TDNN) and dynamic time warping (DTW)

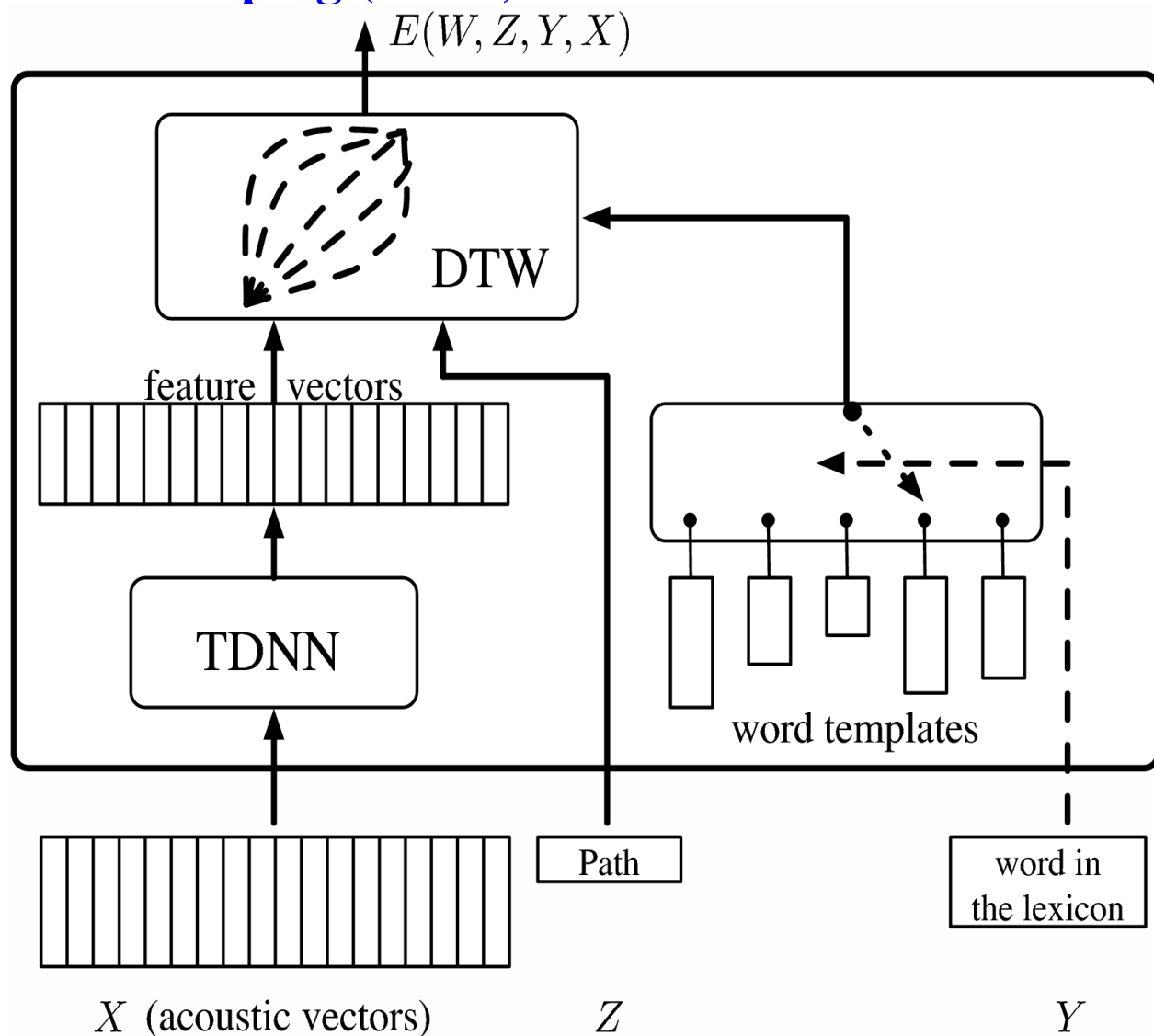
- Training the feature extractor as part of the whole process.

- with the LVQ2 Loss :

- ▶ Driancourt and Bottou's speech recognizer (1991)

- with NLL:

- ▶ Bengio's speech recognizer (1992)
- ▶ Haffner's speech recognizer (1993)



Deep Factors / Deep Graph: ASR with TDNN/HMM

- ❶ **Discriminative Automatic Speech Recognition system with HMM and various acoustic models**
 - ▶ Training the acoustic model (feature extractor) and a (normalized) HMM in an integrated fashion.
- ❷ **With Minimum Empirical Error loss**
 - ▶ Ljolje and Rabiner (1990)
- ❸ **with NLL:**
 - ▶ Bengio (1992)
 - ▶ Haffner (1993)
 - ▶ Bourlard (1994)
- ❹ **With MCE**
 - ▶ Juang et al. (1997)
- ❺ **Late normalization scheme (un-normalized HMM)**
 - ▶ Bottou pointed out the **label bias problem** (1991)
 - ▶ Denker and Burges proposed a solution (1995)

Really Deep Factors / Really Deep Graph

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

