Energy-Based Learning. Structured Output Models

Yann LeCun,
The Courant Institute of Mathematical Sciences
New York University

http://yann.lecun.com
http://www.cs.nyu.edu/~yann

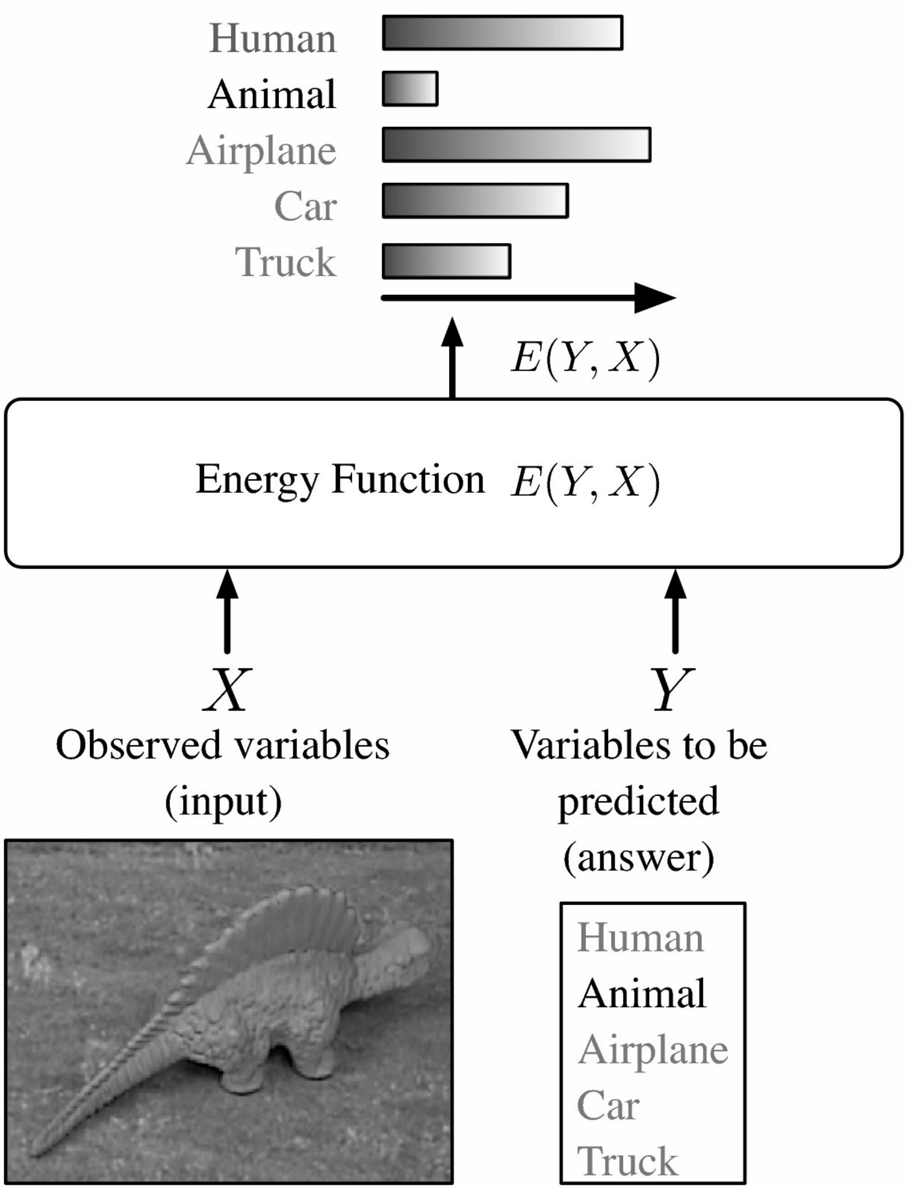
Two Big Problems in Machine Learning

- **1. The "Deep Learning Problem"**
  - "Deep" architectures are necessary to solve the invariance problem in vision (and perception in general)
- **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**
- **Paper: LeCun et al. : "A tutorial on energy-based learning"**
  - http://yann.lecun.com/exdb/publis
  - http://www.cs.nyu.edu/~yann/research/ebm

Plan of the Tutorial

- **Introduction to Energy-Based Models**
  - Energy-Based inference
  - Examples of architectures and applications, structured outputs
- **Training Energy-Based Models**
  - Designing a loss function. Examples of loss functions
  - Which loss functions work, and which ones don't work
  - Getting around the partition function problem with EB learning
- **2. Architectures for Structured Outputs**
  - Energy-Based Graphical Models (non-probabilistic factor graphs)
  - Latent variable models
  - Linear factors: Conditional Random Fields and Maximum Margin Markov Nets
  - Gradient-based learning with non-linear factors
- **Applications: supervised and unsupervised learning**
  - Integrated segmentation/recognition in vision, speech, and OCR.
  - Invariant feature learning, manifold learning

Energy-Based Model for Decision-Making

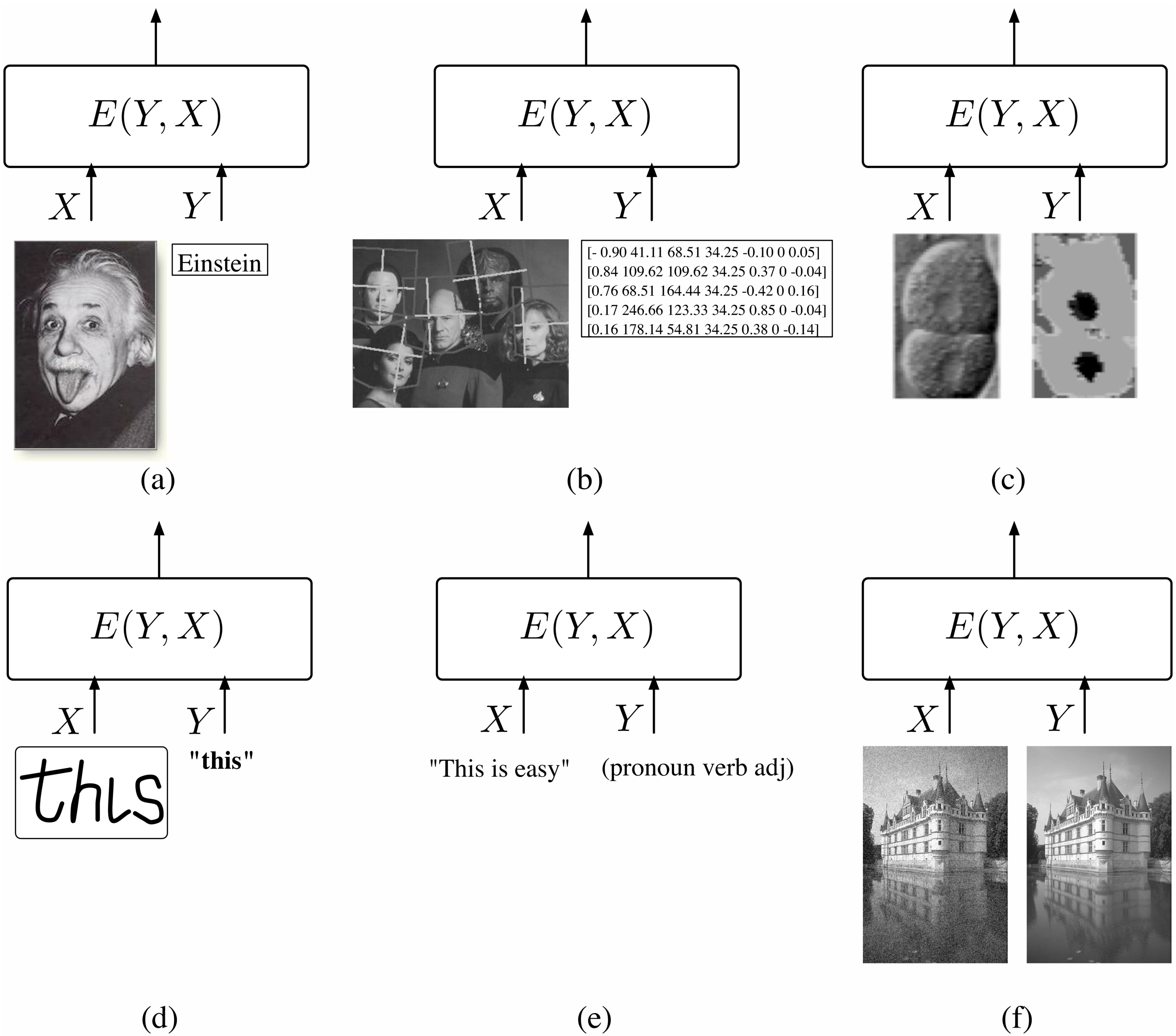


- **Model:** **Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).**

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

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

Complex Tasks: Inference is non-trivial



- **When the cardinality or dimension of Y is large, exhaustive search is impractical.**
- **We need to use 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 Y1 or Y2 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 untis)
- **How do we calibrate energies?**
  - We turn them into probabilities (positive numbers that sum to 1).
  - Simplest way: Gibbs distribution
  - Other ways can be reduced to Gibbs by a suitable redefinition of the energy.

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

Partition function

Inverse temperature

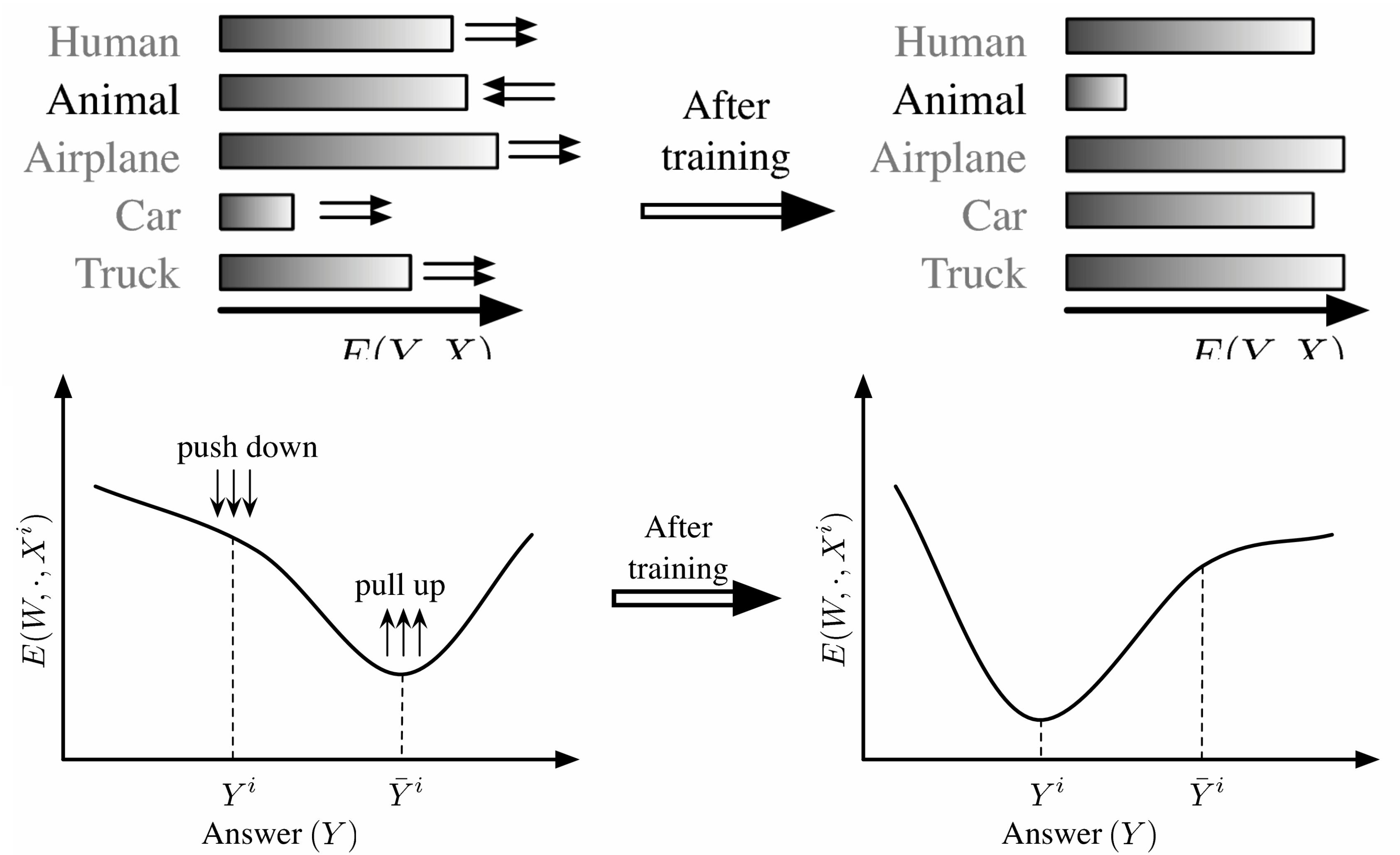
Architecture and Loss Function

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

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

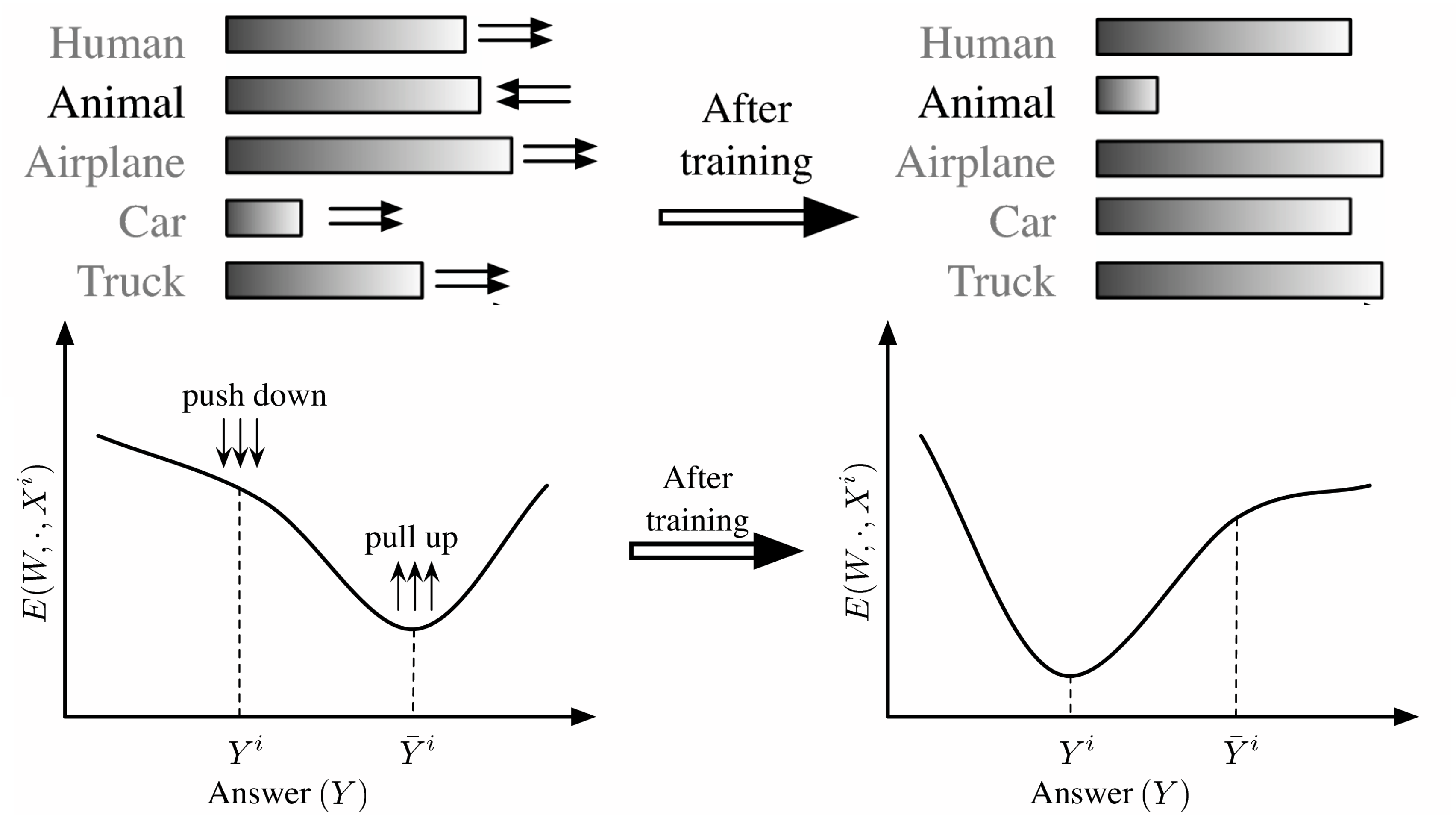
Per-sample loss — Desired answer — Energy surface for a given Xi as Y varies — Regularizer

Designing a Loss Functional



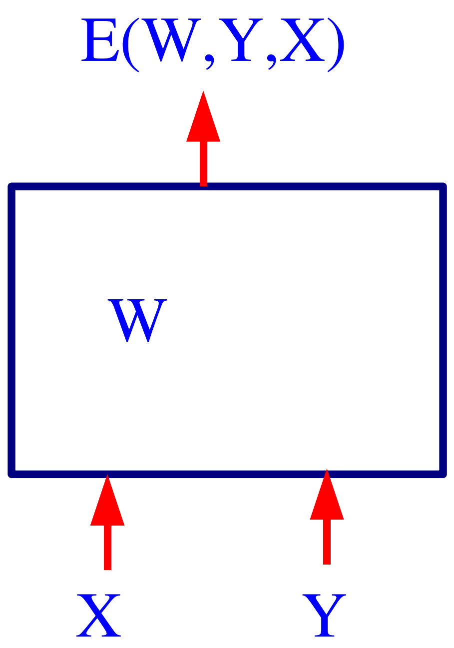
- Correct answer has the lowest energy -> LOW LOSS
- Lowest energy is not for the correct answer -> HIGH LOSS

Designing a Loss Functional



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

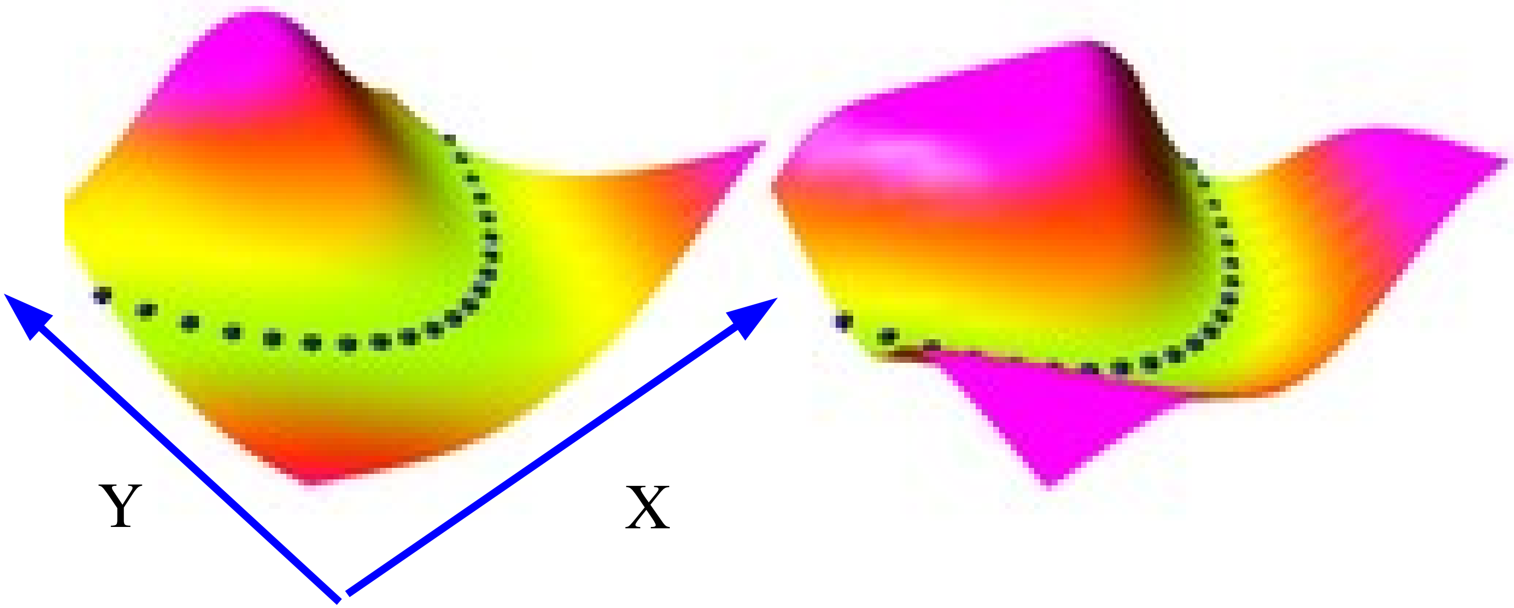
Architecture + Inference Algo + Loss Function = Model



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

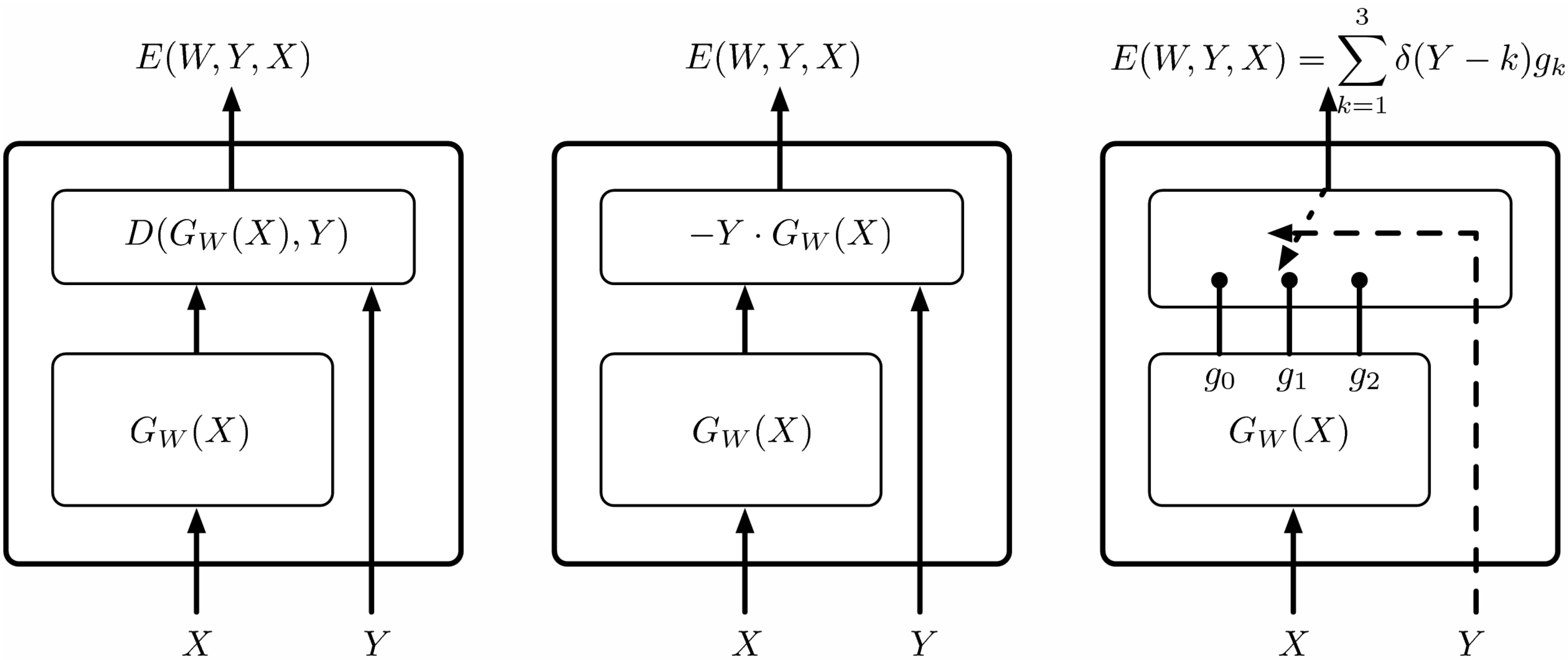
- **PROBLEM: What loss functions will make the machine approach the desired behavior?**

Several Energy Surfaces can give the same answers



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

Simple Architectures



- **Regression**
- **Binary Classification**
- **Multi-class Classification**

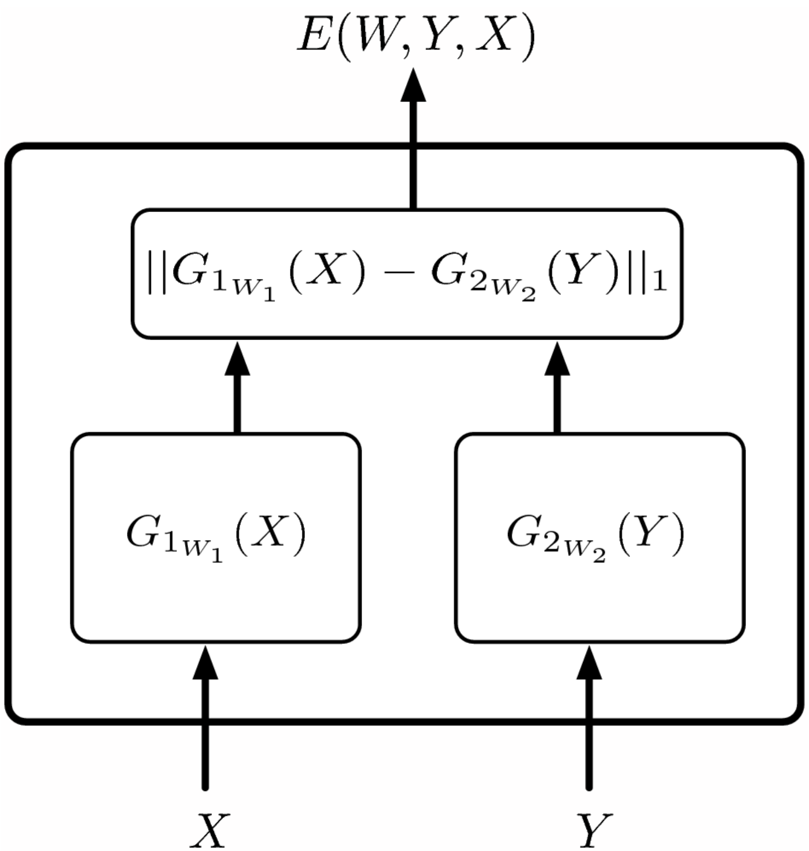
$$E(W,Y,X) = \frac{1}{2}||G_W(X) - Y||^2.$$

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

Simple Architecture: Implicit Regression

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

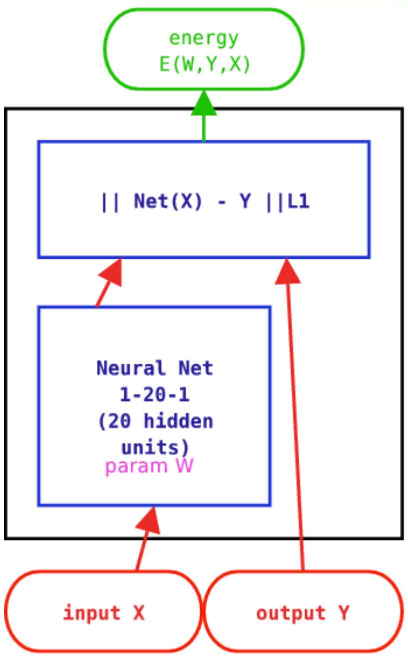
- **The Implicit Regression architecture**
  - allows multiple answers to have low energy.
  - Encodes a constraint between X and Y rather than an explicit functional relationship
  - This is useful for many applications
  - Example: sentence completion: "The cat ate the {mouse,bird,homework,...}"
 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



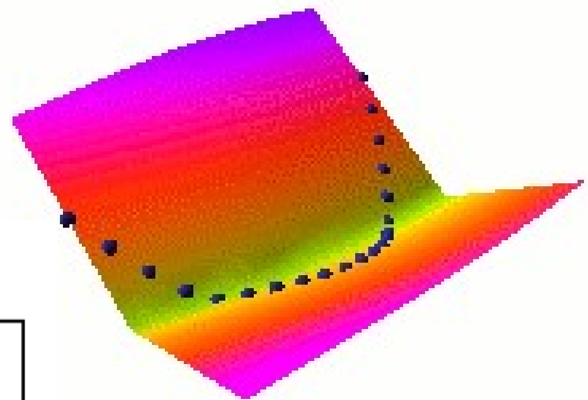
Examples of Loss Functions: Energy Loss

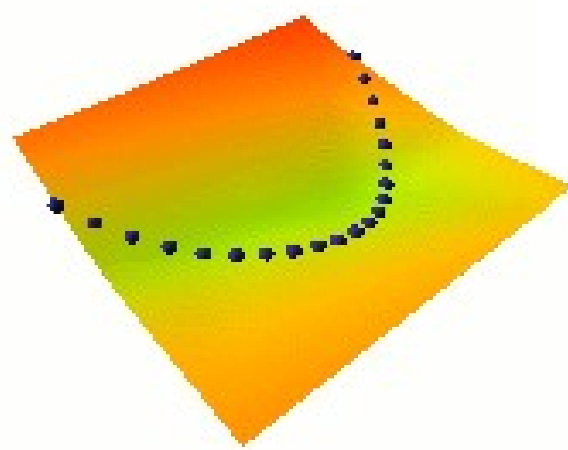
- **Energy Loss** $L_{energy}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i).$
  - Simply pushes down on the energy of the correct answer

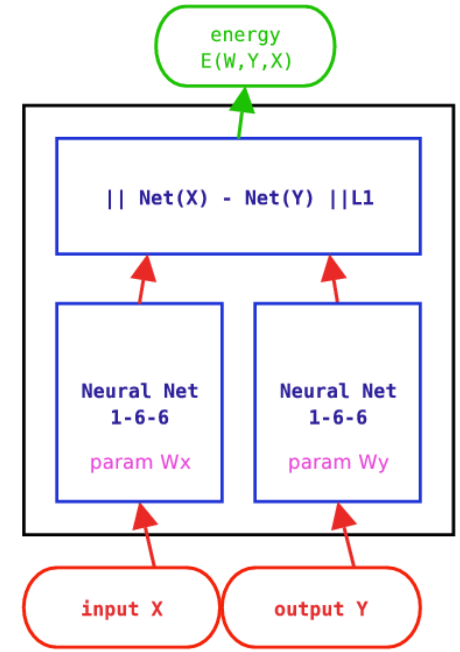
WORKS!!!



(a)







(b)

COLLAPSES!!!

Examples of Loss Functions:Perceptron Loss

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

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

Perceptron Loss for Binary Classification

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

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

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

Examples of Loss Functions: Generalized Margin Losses

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

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

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

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

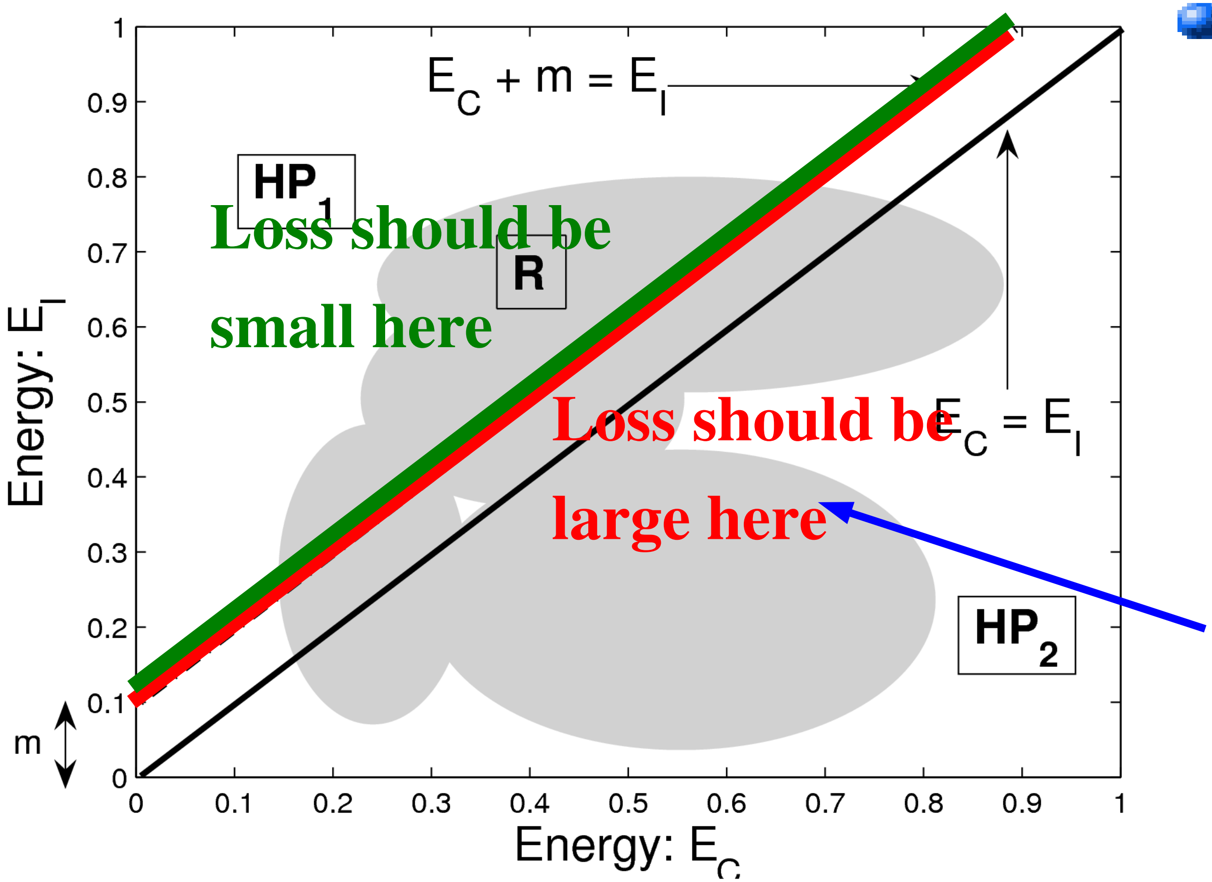
- **Most Offending Incorrect Answer: continuous case**

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

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

Examples of Loss Functions: Generalized Margin Losses

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



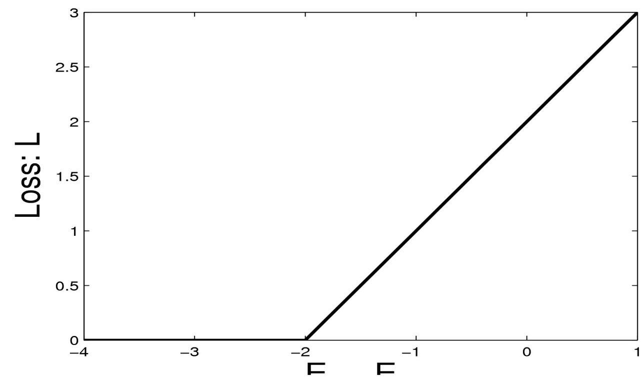
- **Generalized Margin Loss**
  - Qm increases with the energy of the correct answer
  - Qm decreases with the energy of the most offending incorrect answer
  - whenever it is less than the energy of the correct answer plus a margin m.

Examples of Generalized Margin Losses

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

Hinge Loss

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

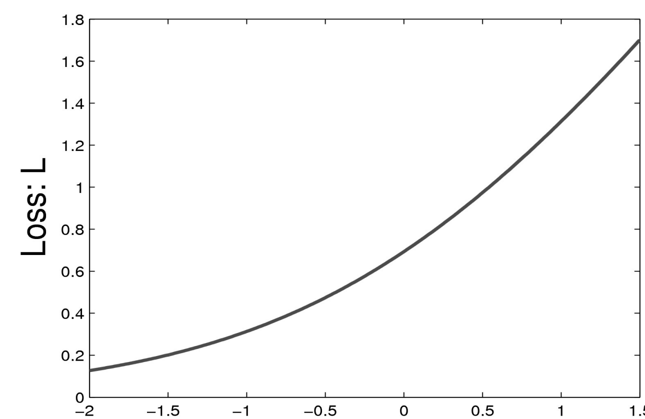


E_correct - E_incorrect

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

Log Loss

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



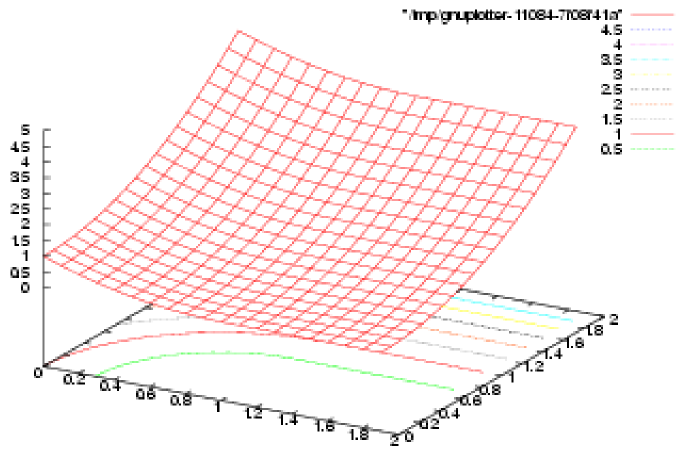
E_correct - E_incorrect

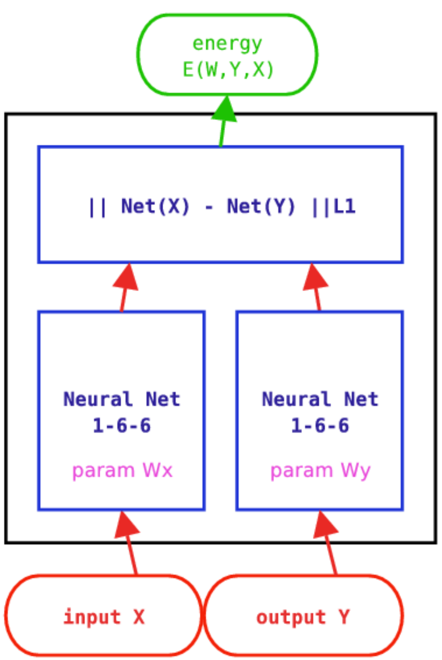
Examples of Margin Losses: Square-Square Loss

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

Square-Square Loss

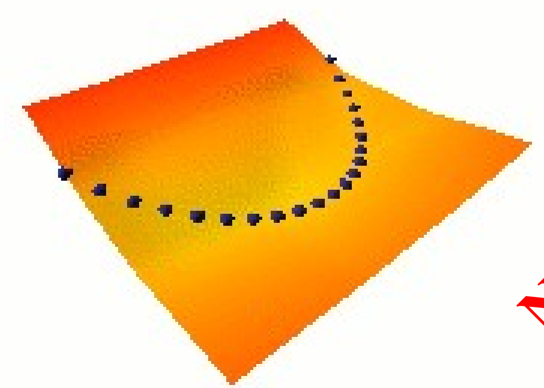
- [LeCun-Huang 2005]
- Appropriate for positive energy functions





(b)

Learning Y = X^2



NO COLLAPSE!!!

Other Margin-Like Losses

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

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

- **Minimum Classification Error Loss** [Juang, Chou, Lee 1997] <- speech r.

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

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

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

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

Negative Log-Likelihood Loss

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

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

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

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

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

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

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

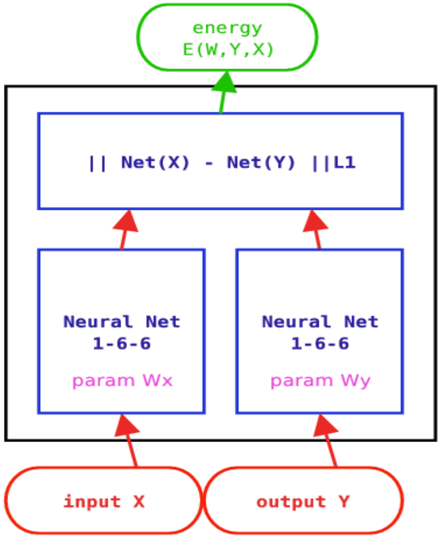
- **Reduces to the perceptron loss when Beta->infinity**

Negative Log-Likelihood Loss

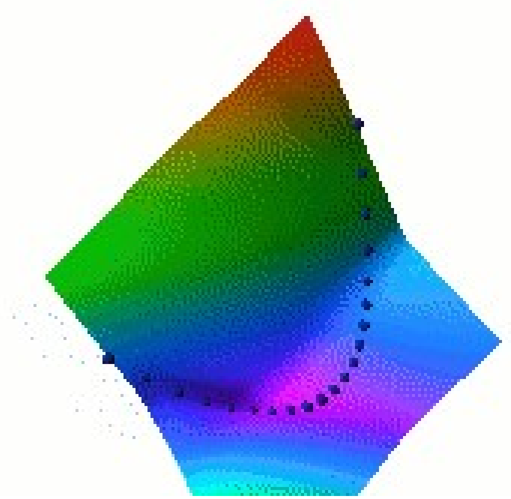
- Pushes down on the energy of the correct answer
- Pulls up on the energies of all answers in proportion to their probability

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

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



(b)



Negative Log-Likelihood Loss: Binary Classification

- **Binary Classifier Architecture:**

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

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

- **Linear Binary Classifier Architecture:**

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

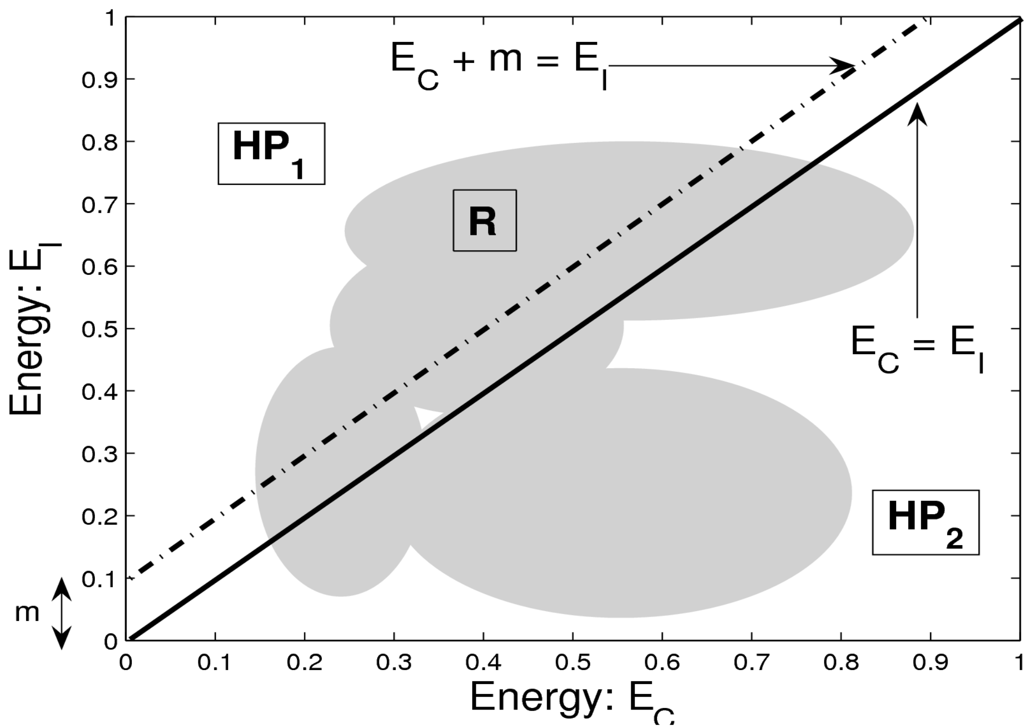
- **Learning Rule in the linear case: logistic regression**
- **NLL is used by lots of speech recognition systems (they call it Maximum Mutual Information), lots of handwriting recognition systems (e.g. Bengio, LeCun 94] [LeCun et al. 98]), CRF [Lafferty et al 2001]**

Negative Log-Likelihood Loss

- **Negative Log Likelihood Loss has been used for a long time in many communities for discriminative learning with structured outputs**
  - Speech recognition: many papers going back to the early 90's [Bengio 92], [Bourlard 94]. They call “Maximum Mutual Information”
  - Handwriting recognition [Bengio LeCun 94], [LeCun et al. 98]
  - Bio-informatics [Haussler]
  - Conditional Random Fields [Lafferty et al. 2001]
  - Lots more......
  - In all the above cases, it was used with non-linearly parameterized energies.

What Makes a "Good" Loss Function

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



Sufficient Condition on the Loss

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

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

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

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

What Make a "Good" Loss Function

Good and bad loss functions

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

Advantages/Disadvantages of various losses

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

Latent Variable Models

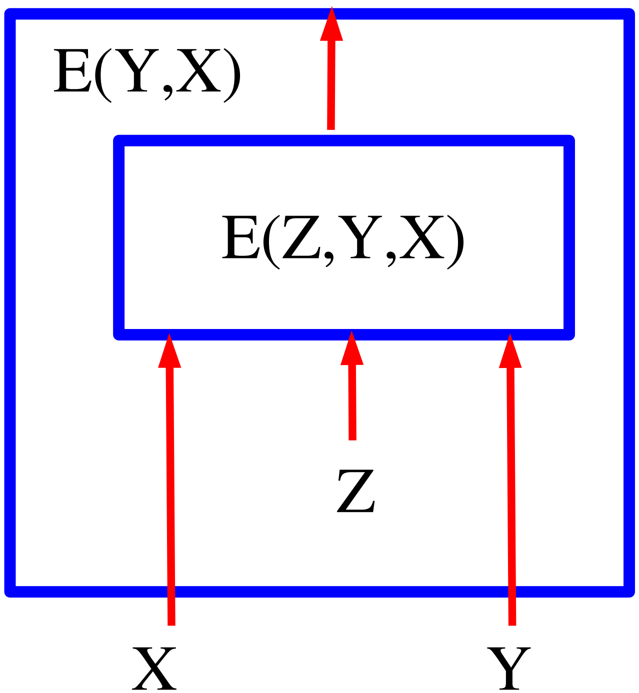
- **The energy includes "hidden" variables Z whose value is never given to us**
  - We can minimize the energy over those latent variables
  - We can also "marginalize" the energy over the latent variables

Minimization over latent variables:

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

Marginalization over latent variables:

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



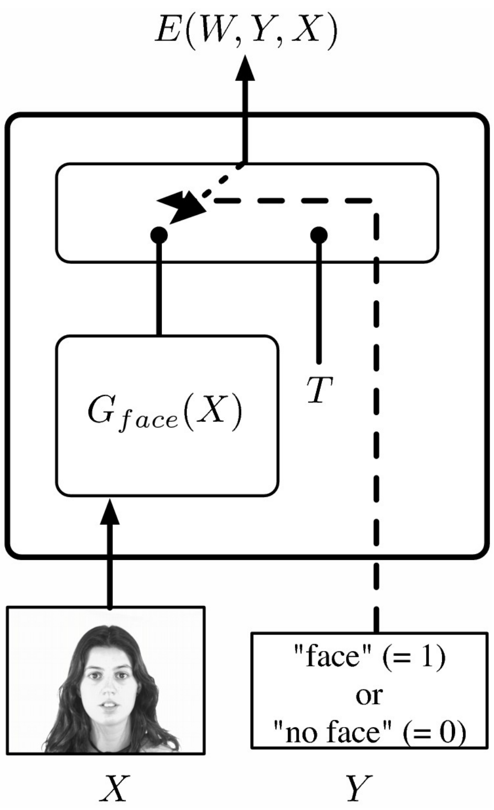
Estimation this integral may require some approximations (sampling, variational methods,....)

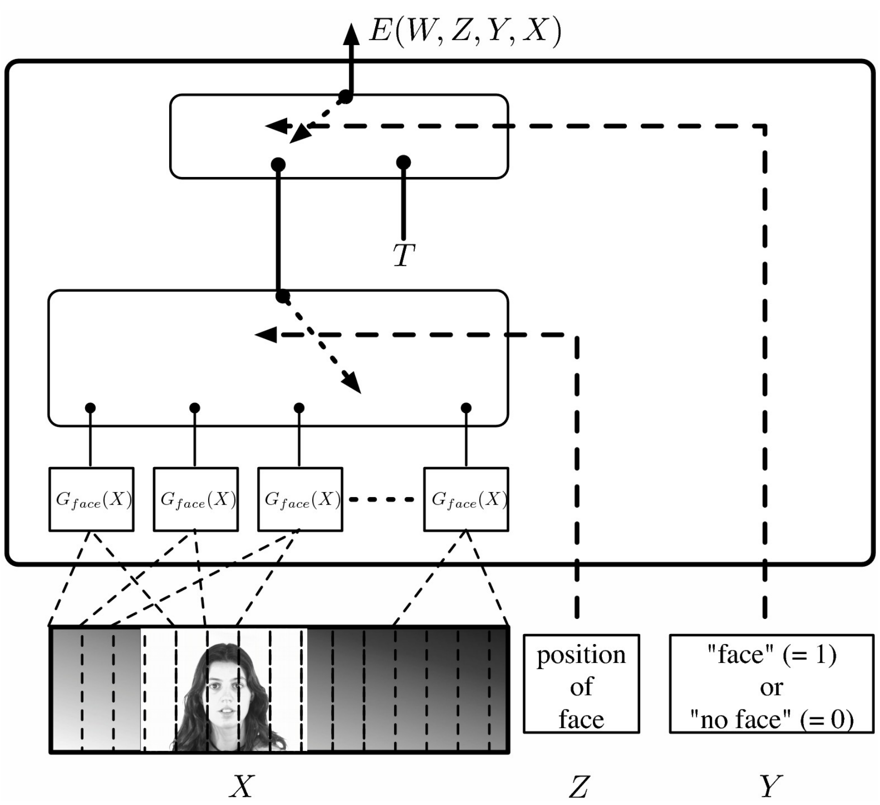
Latent Variable Models

- **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:**

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

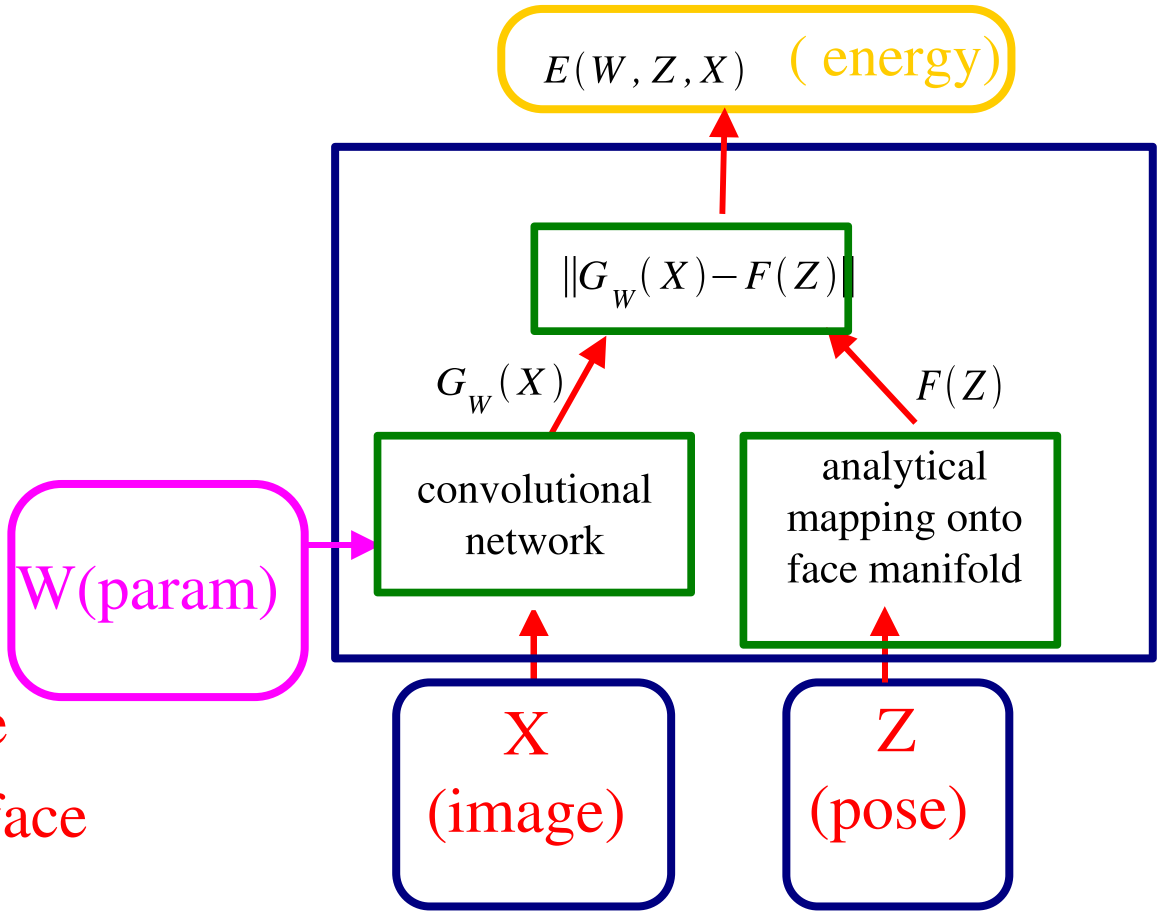
- **Reduces to minimization when Beta->infinity**

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 .

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

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

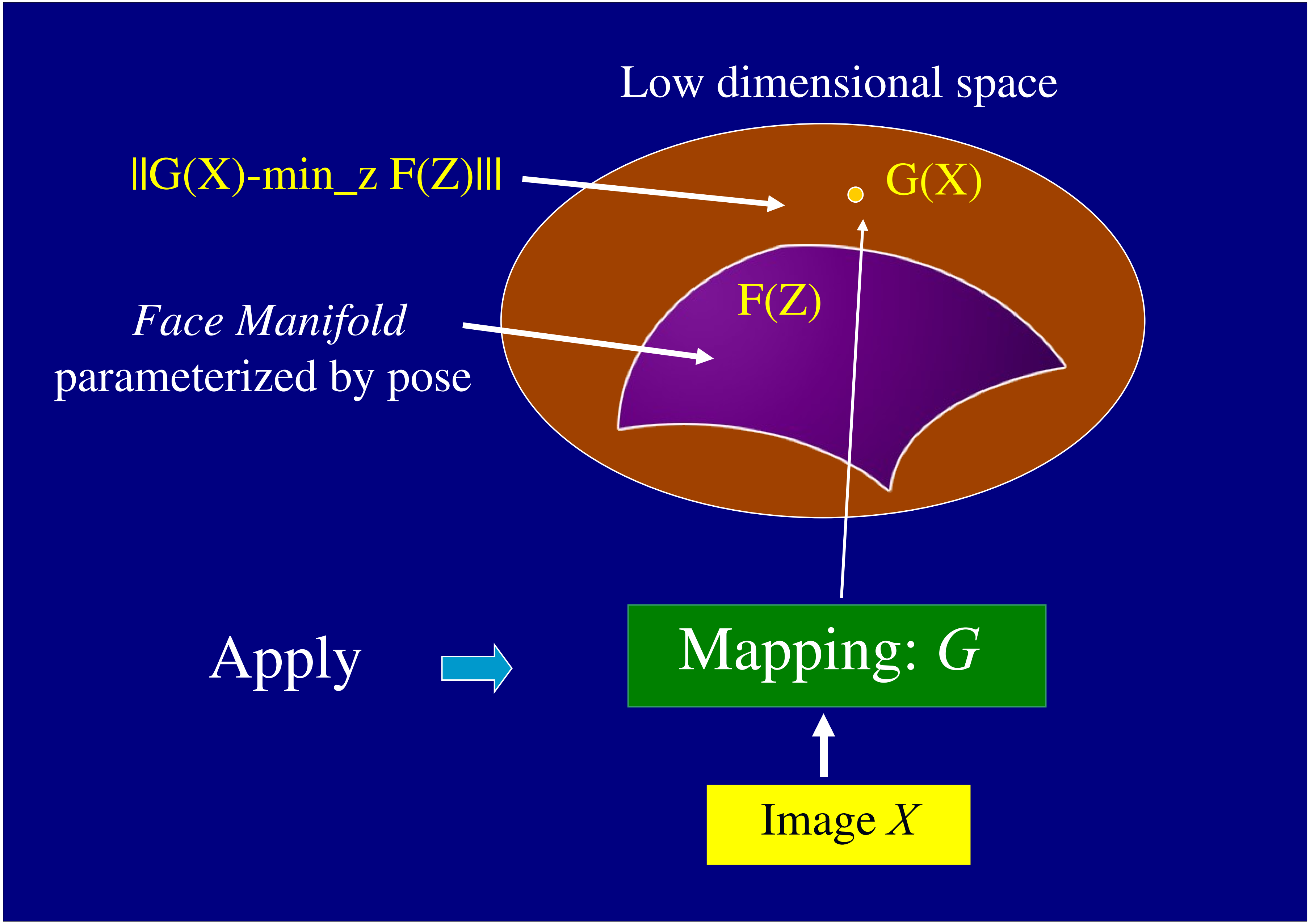


Small E*(W,X): face

Large E*(W,X): no face

[Osadchy, Miller, LeCun, NIPS 2004]

Face Manifold



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

Probability that image X is a face with pose Z

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

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

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

Equivalently, minimize the negative log likelihood:

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

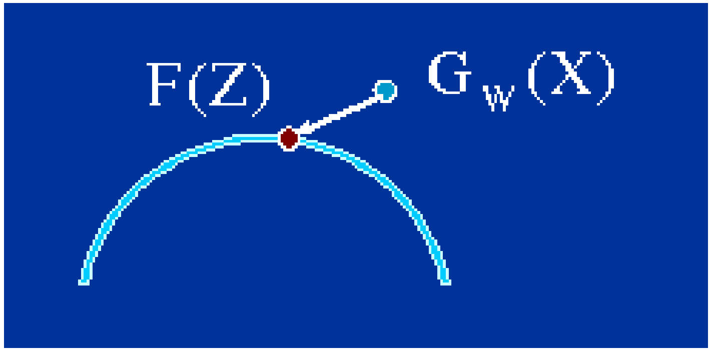
↑

COMPLICATED

Energy-Based Contrastive Loss Function

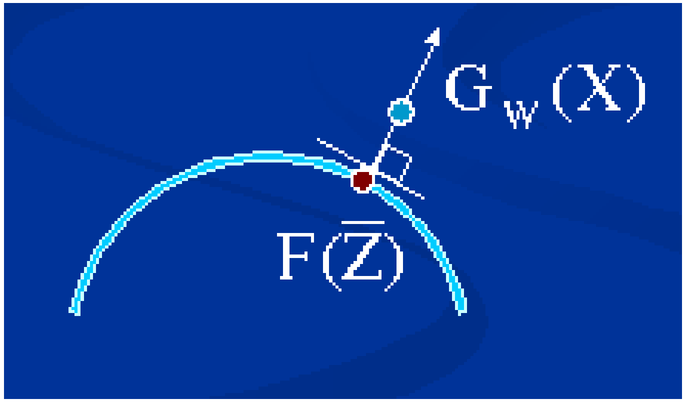
$$\mathcal{L}(W) = \frac{1}{|\mathrm{f}+\mathrm{p}|} \sum_{X,Z \in \mathrm{faces+pose}} \left[L^{+}\left(E(W,Z,X)\right)\right] + L^{-}\left(\min_{X,Z \in \mathrm{bckgnd,poses}} E(W,Z,X)\right)$$

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



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

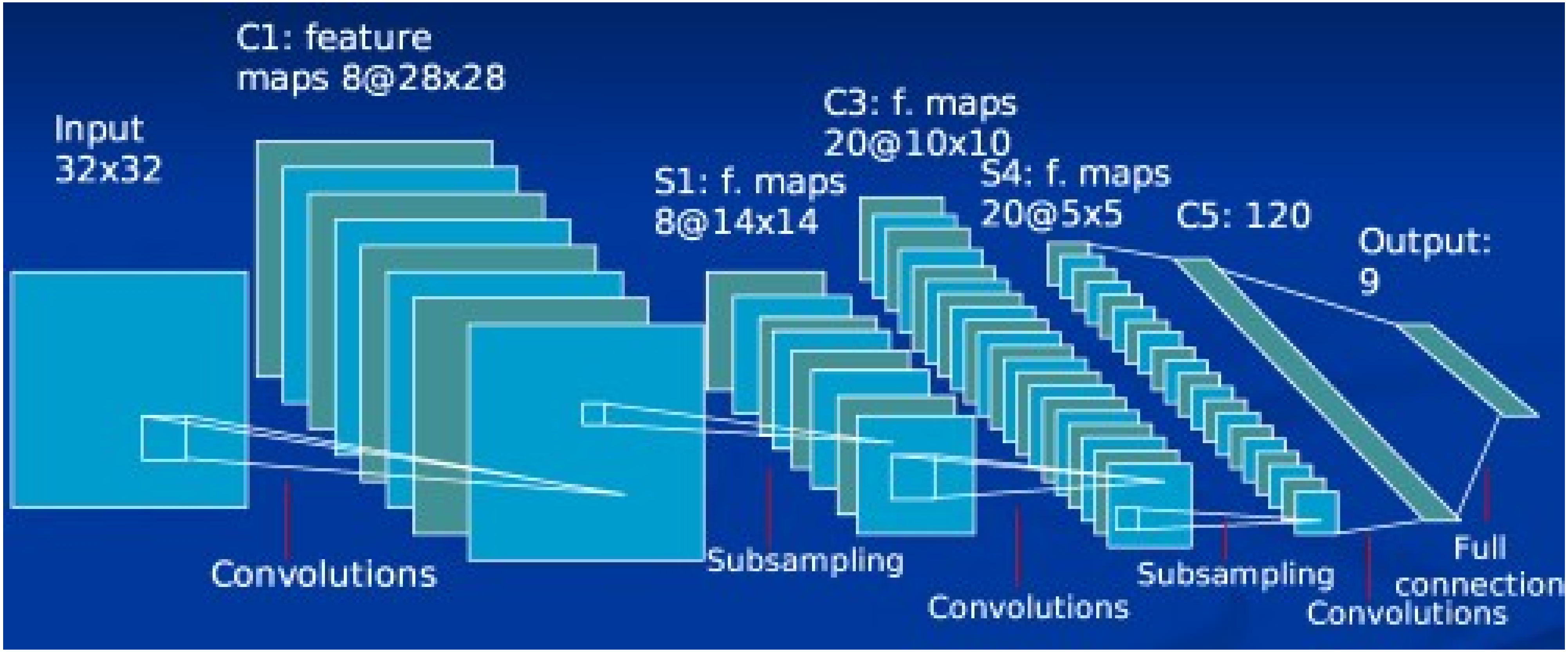
$$L^{-}\left(\min_{X,Z \in \mathrm{bckgnd,poses}} E(W,Z,X)\right) = K \exp\left(-\min_{X,Z \in \mathrm{bckgnd,poses}} ||G_W(X) - F(Z)||\right)$$



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

Convolutional Network Architecture

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

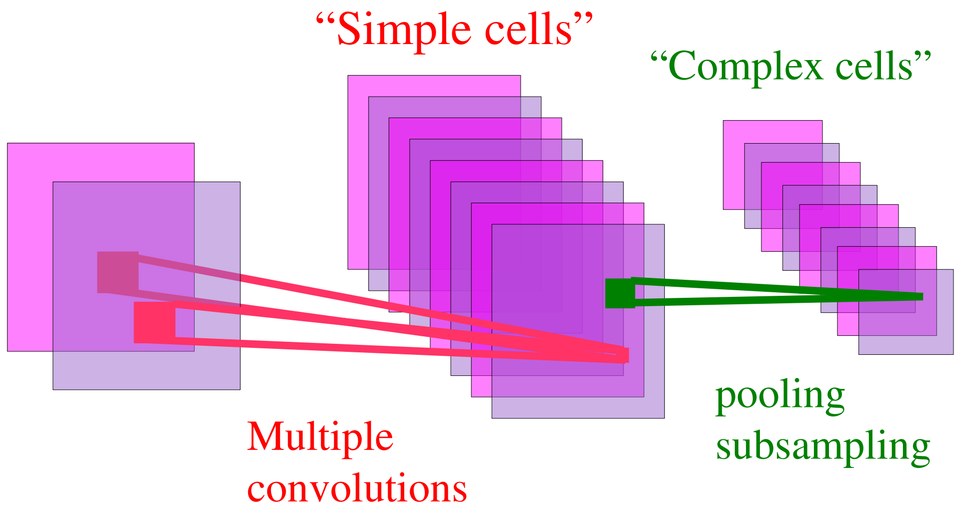


Hierarchy of local filters (convolution kernels),

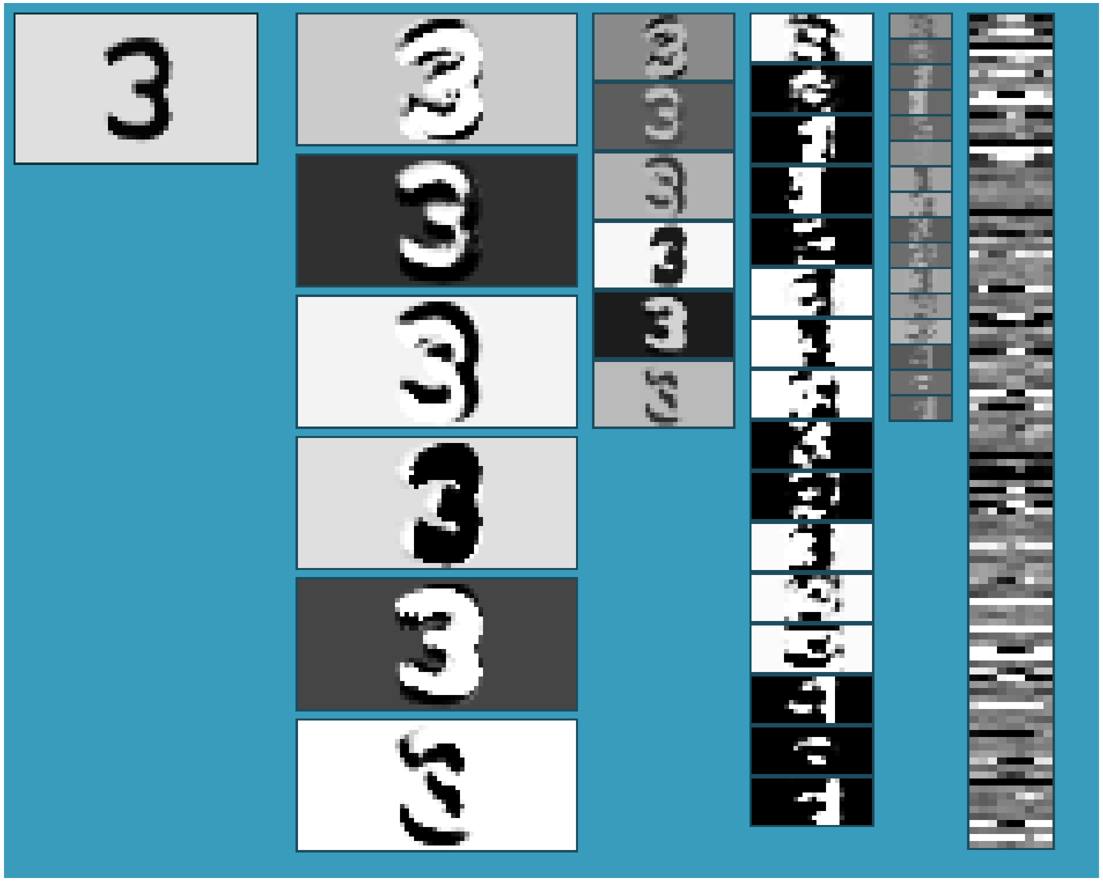
sigmoid pointwise non-linearities, and spatial subsampling

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

Alternated Convolutions and Pooling/Subsampling

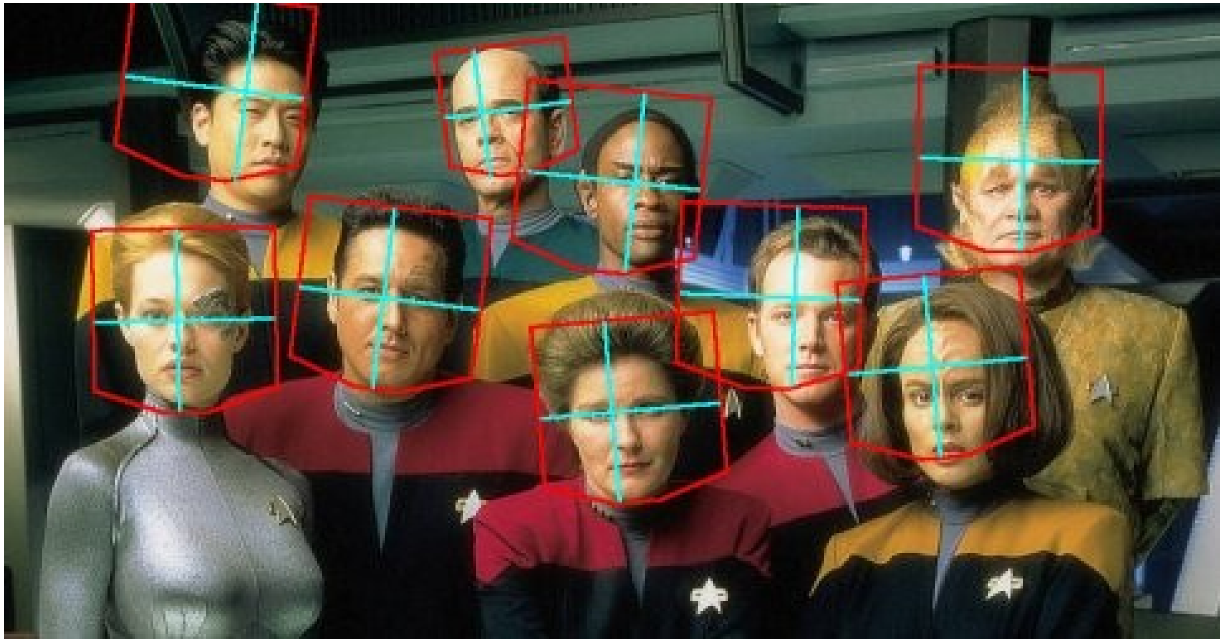


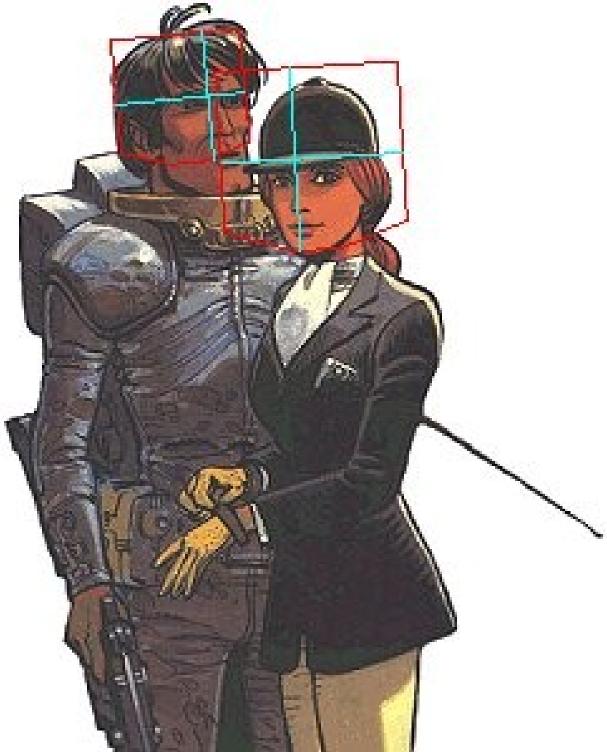
- Local features are extracted everywhere.
- pooling/subsampling layer builds robustness to variations in feature locations.
- Long history in neuroscience and computer vision:
  - Hubel/Wiesel 1962,
  - Fukushima 1971-82,
  - LeCun 1988-06
  - Poggio, Riesenhuber, Serre 02-06
  - Ullman 2002-06
  -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



Face Detection: Results

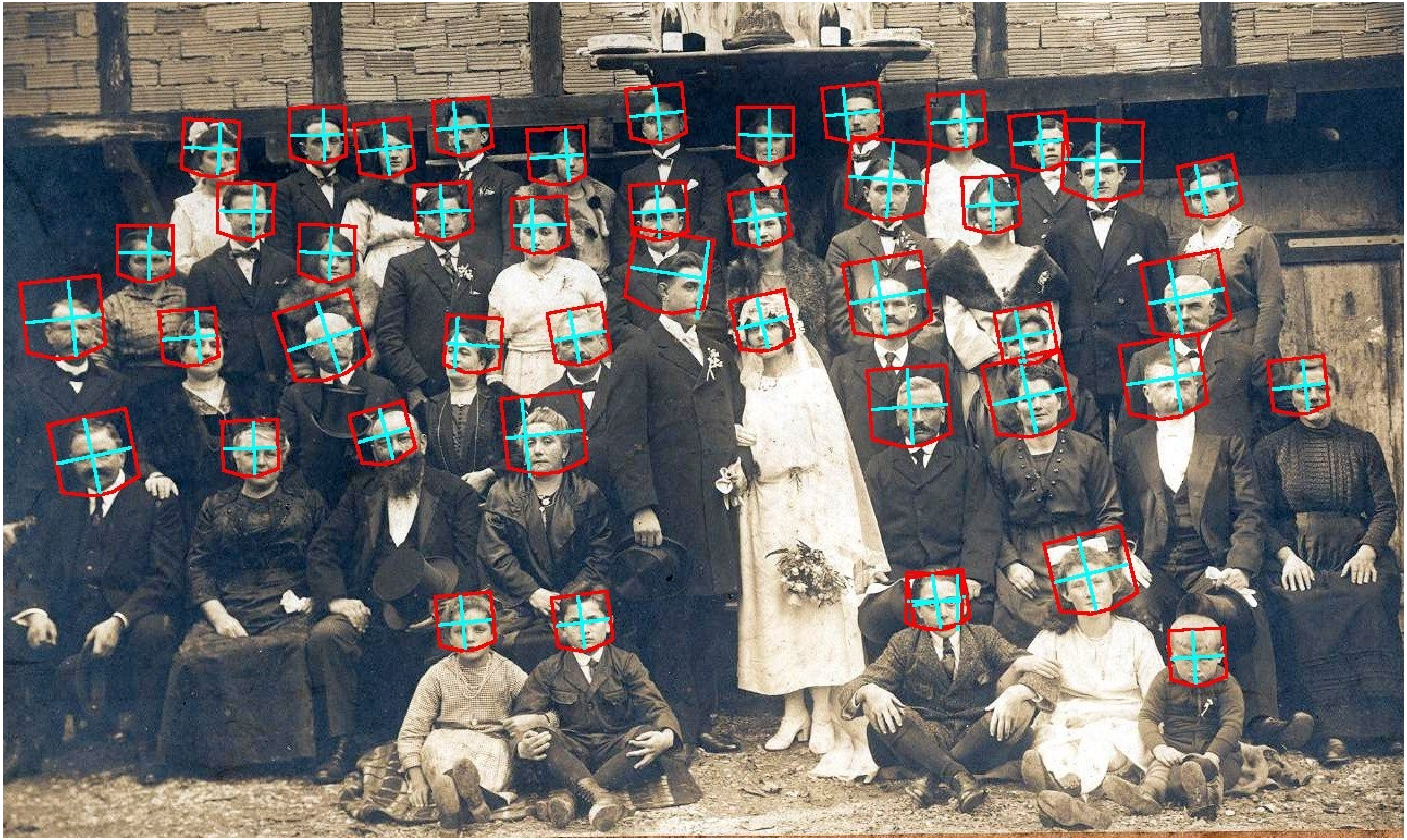
Data Set->	TILTED		PROFILE		MIT+CMU	
False positives per image->	4.42	26.9	0.47	3.36	0.5	1.28
Our Detector	90%	97%	67%	83%	83%	88%
Jones & Viola (tilted)	90%	95%	x		x	
Jones & Viola (profile)	x		70%	83%	x	





Face Detection and Pose Estimation: Results

Face Detection with a Convolutional Net

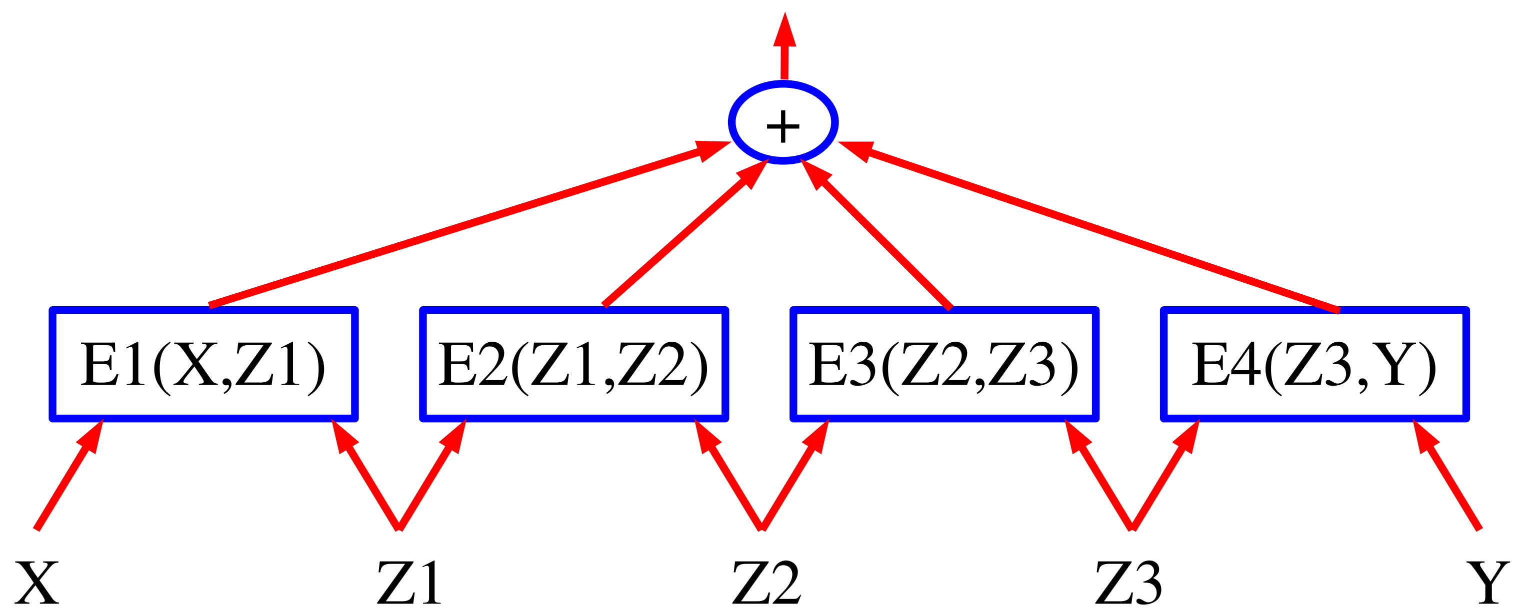


Efficient Inference: Energy-Based Factor Graphs

- **Graphical models have given us efficient inference algorithms, such as belief propagation and its numerous variations.**
- **Traditionally, graphical models are viewed as probabilistic models**
- **At first glance, is seems difficult to dissociate graphical models from the probabilistic view (think "Bayesian networks").**
- **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.**
- **Basically, most algorithms for probabilistic factor graphs (such as belief prop) have a counterpart for EBFG:**
  - Operations are performed in the log domain
  - The normalization steps are left out.

Energy-Based Factor Graphs

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

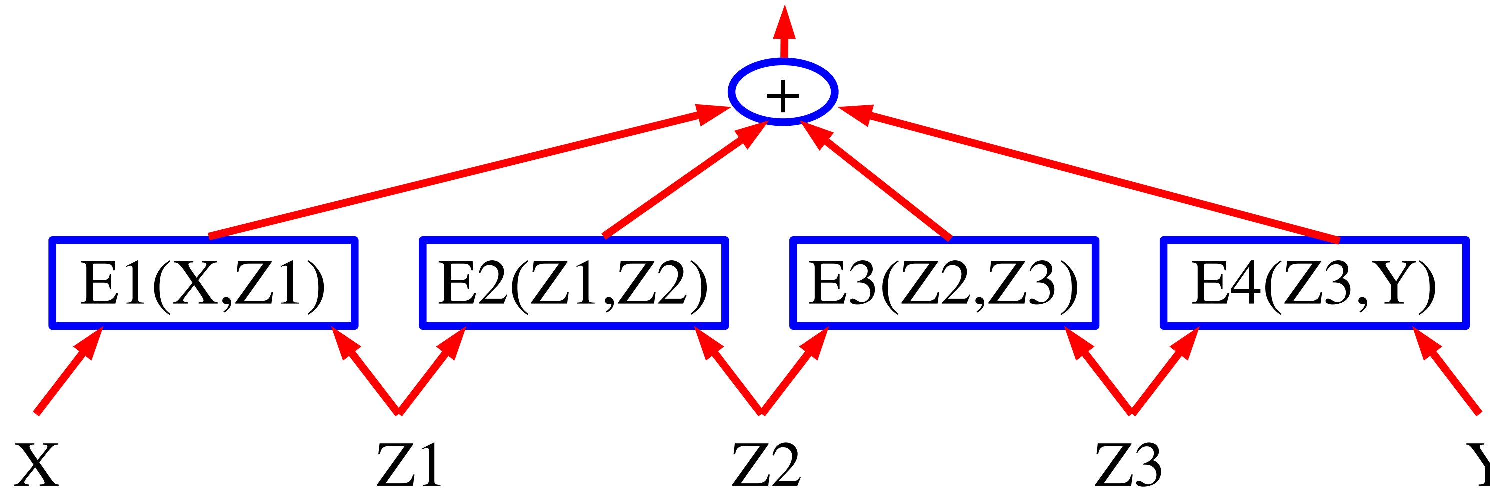


EBFG for Structured Outputs: Sequences, Graphs, Images

- **Structured outputs**
  - When Y is a complex object with components that must satisfy certain constraints.
- **Typically, structured outputs are sequences of symbols that must satisfy "grammatical" constraints**
  - spoken/handwritten word recognition
  - spoken/written sentence recognition
  - DNA sequence analysis
  - Parts of Speech tagging
  - Automatic Machine Translation
- **In General, structured outputs are collections of variables in which subsets of variables must satisfy constraints**
  - Pixels in an image for image restoration
  - Labels of regions for image segmentations
- **We represent the constraints using an Energy-Based Factor Graph.**

Energy-Based Factor Graphs: Three Inference Problems

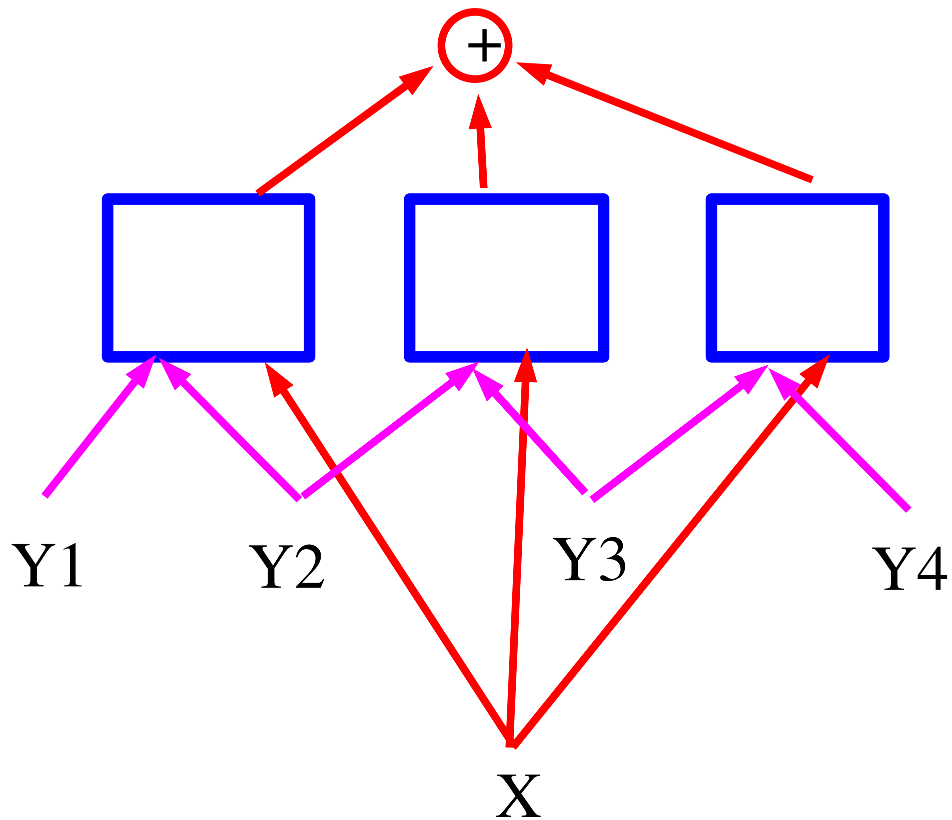
- **X: input, Y: output, Z: latent variables, Energy: E(Z,Y,X)**
- **Minimization over Y and Z**
  - $$E(Y,X) = \min_{Z \in \mathcal{Z}} E(Z,Y,X). \qquad Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y,X).$$
- **Min over Y, marginalization over Z (E(X,Y) is a "free energy")**
  - $$E(X,Y) = -\frac{1}{\beta} \log \int_{z \in \mathcal{Z}} e^{-\beta E(z,Y,X)} \quad Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y,X).$$
- **Marginal Distribution over Y**
  - $$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}},$$



Energy-Based Factor Graphs: simple graphs

- **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)

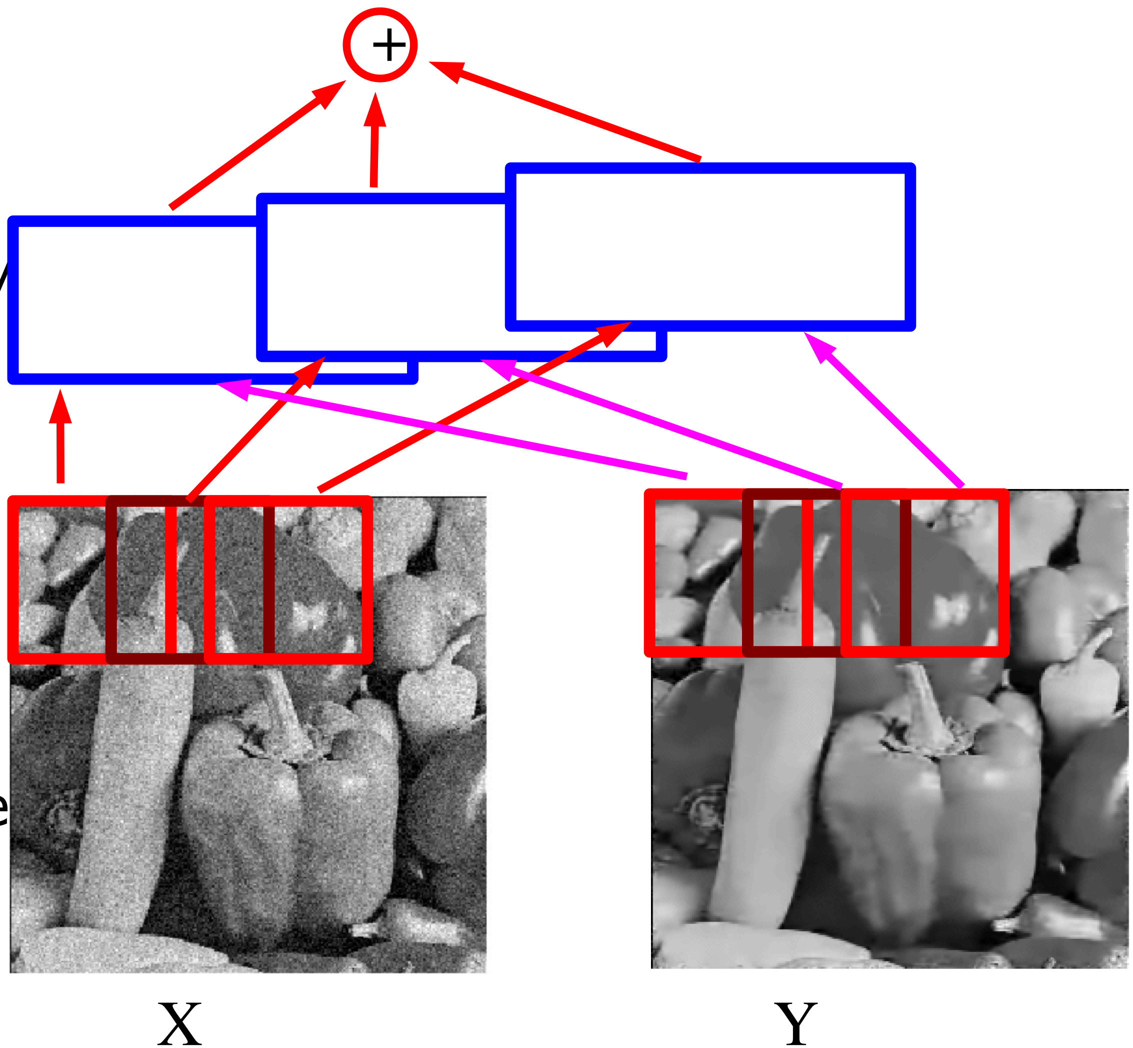
$$Y^* = \text{argmin}_{Y \in \mathcal{Y}, Z \in \mathcal{Z}} E(Z, Y, X).$$



Energy-Based Factor Graphs: complex/loopy graphs

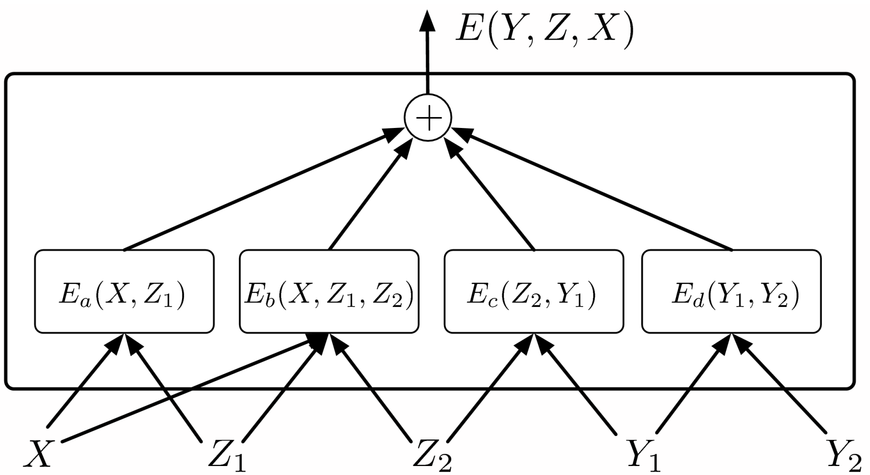
- **Image restoration**
  - The factors ensure local consistency on small overlapping patches
  - They give low energy to "clean" patches, given the noisy versions
  - The graph is loopy when the patches overlap.
  - Inference is difficult, particularly when the patches are large,and when the number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

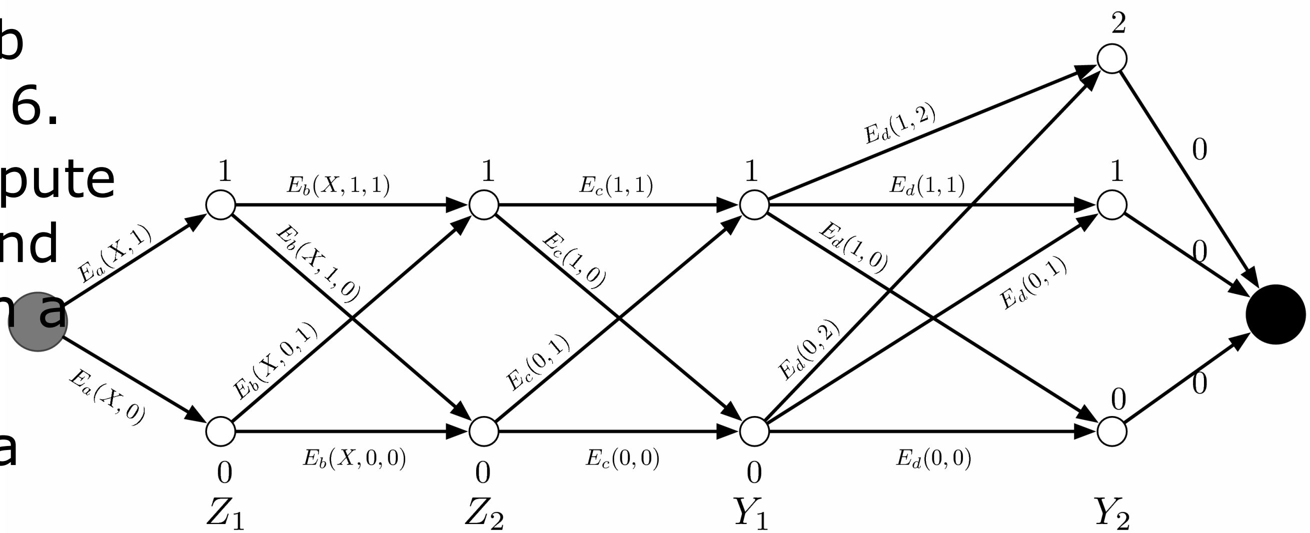
$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$



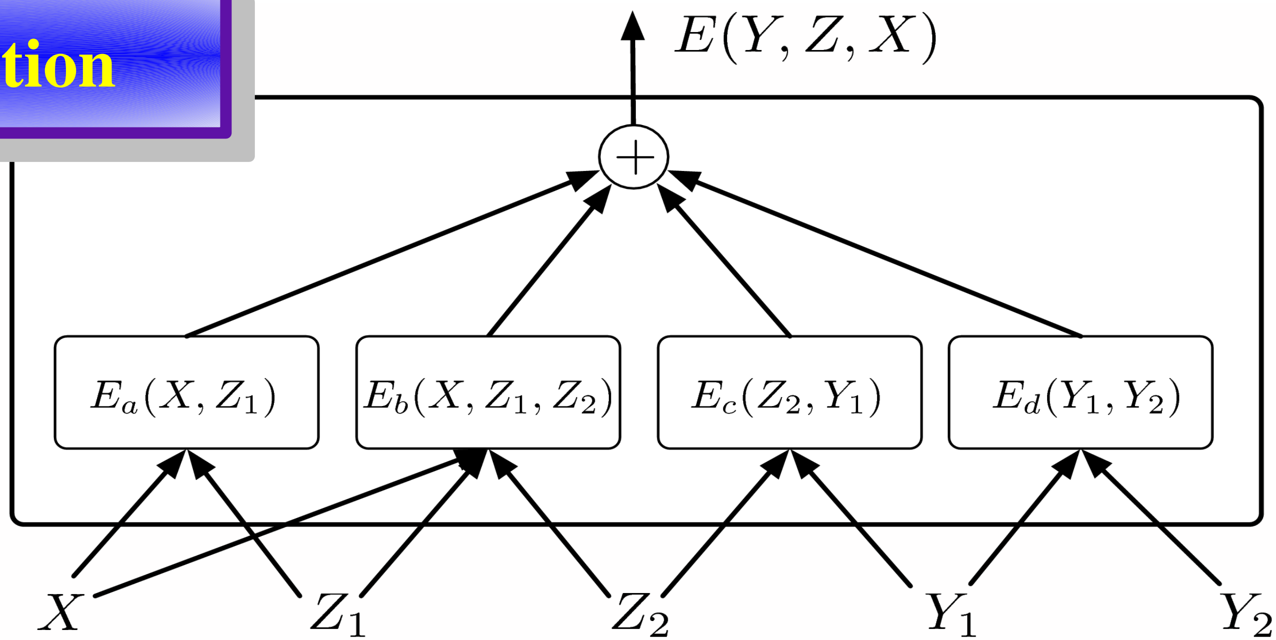
Efficient Inference in simple EBFG

- **The energy is a sum of "factor" functions, the graph is a chain**
- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 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



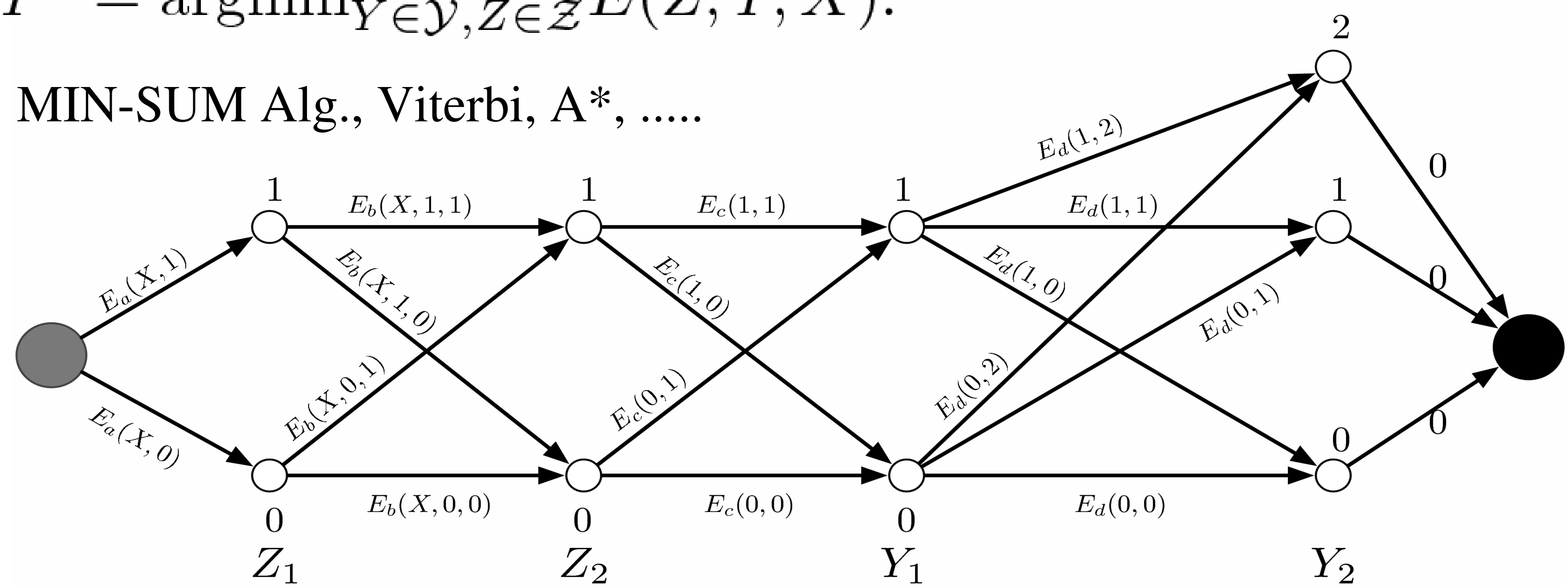


Minimization



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

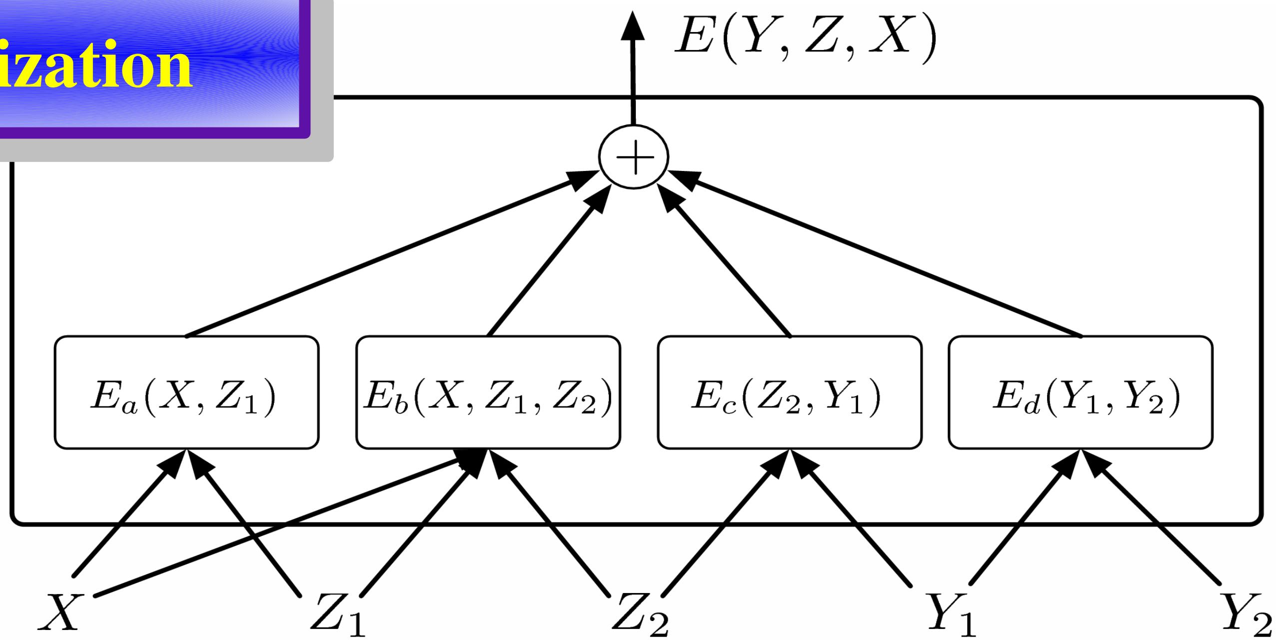
MIN-SUM Alg., Viterbi, A*, ...



Energy-Based Belief Prop: Minimization over Latent Variables

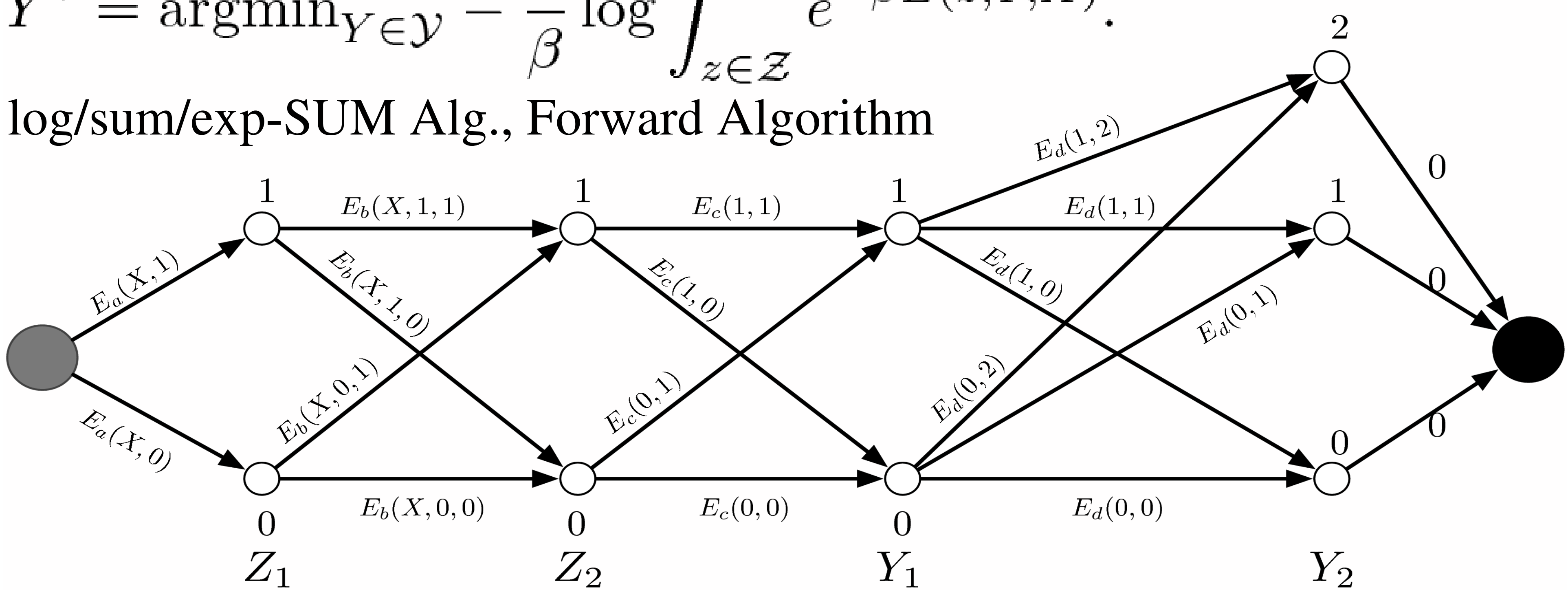
- **The previous picture shows a chain graph of factors with 2 inputs.**
- **The extension of this procedure to trees, with factors that can have more than 2 inputs is the “min-sum” algorithm (a non-probabilistic form of belief propagation)**
- **Basically, it is the sum-product algorithm with a different semi-ring algebra (min instead of sum, sum instead of product), without the normalization step.**
  - [Kschischang, Frey, Loeliger, 2001][McKay's book]

Marginalization



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

log/sum/exp-SUM Alg., Forward Algorithm



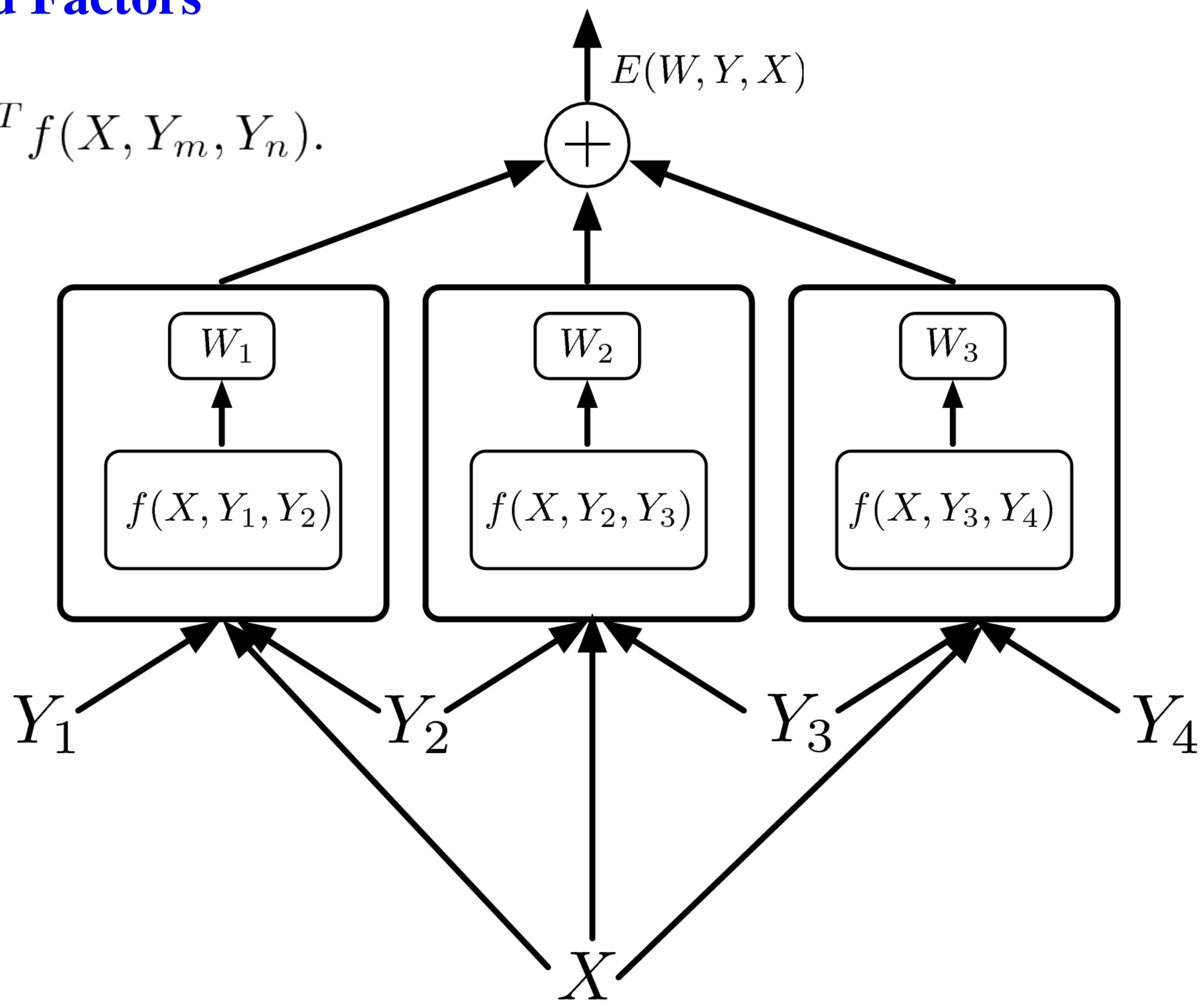
Energy-Based Belief Prop: Marginalization over Latent Variables

- **The previous picture shows a chain graph of factors with 2 inputs.**
  - Going along a path: add up the energies
  - When several paths meet: compute $-\frac{1}{\beta}\log\sum_i e^{-\beta E_{ji}}$
- **The extension of this procedure to trees, with factors that can have more than 2 inputs is the "[log/sum/exp]-sum" algorithm (a non-probabilistic form of belief propagation)**
- **Basically, it is the sum-product algorithm with a different semi-ring algebra (log/sum/exp instead of sum, sum instead of product), and without the normalization step.**
  - [Kschischang, Frey, Loeliger, 2001][McKay's book]

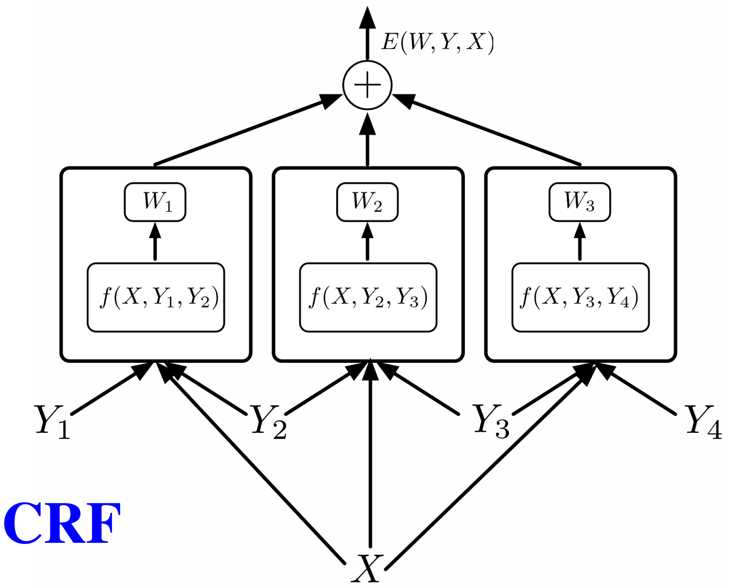
A Simple Case: Linearly Parameterized Factors: CRF, MMMN

Linearly Parameterized Factors

$$E(W,Y,X) = \sum_{(m,n)\in\mathcal{F}} W^T f(X,Y_m,Y_n).$$



Linearly Parameterized Factors + Negative Log Likelihood Loss = Conditional Random Fields



- **Linearly Parameterized Factors + NLL loss = CRF**
  - [Lafferty, McCallum, Pereira, 2001]

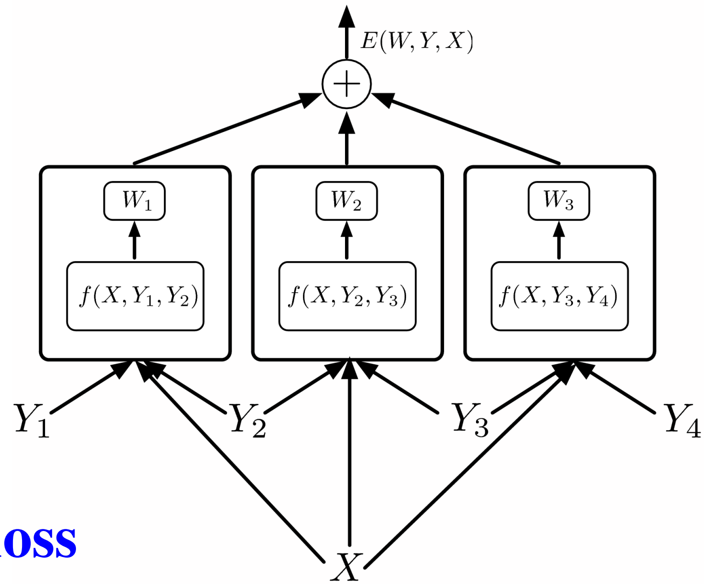
$$\mathcal{L}_{\text{nll}}(W) = \frac{1}{P}\sum_{i=1}^{P} W^T F(X^i,Y^i) + \frac{1}{\beta}\log \sum_{y\in\mathcal{Y}} e^{-\beta W^T F(X^i,y)}.$$

$$\frac{\partial \mathcal{L}_{\text{nll}}(W)}{\partial W} = \frac{1}{P}\sum_{i=1}^{P} F(X^i,Y^i) - \sum_{y\in\mathcal{Y}} F(X^i,y)P(y|X^i,W),$$

$$P(y|X^i,W) = \frac{e^{-\beta W^T F(X^i,y)}}{\sum_{y'\in\mathcal{Y}} e^{-\beta W^T F(X^i,y')}}.$$

simplest/best learning procedure: stochastic gradient

Linearly Parameterized Factors + Perceptron Loss = Sequence Perceptron



- **Linearly Parameterized Factors + Perceptron loss**
  - [LeCun, Bottou, Bengio, Haffner 1998, Collins 2000, Collins 2001]

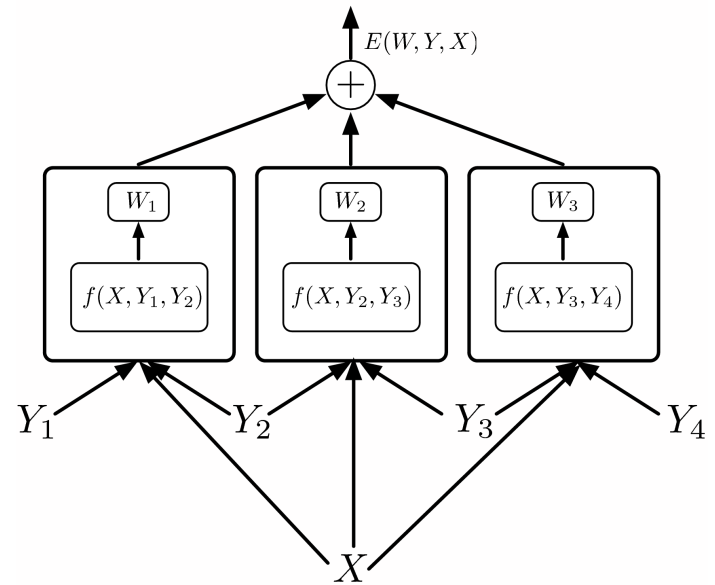
$$\mathcal{L}_{\text{perceptron}}(W) = \frac{1}{P}\sum_{i=1}^{P} E(W,Y^i,X^i) - E(W,Y^{*i},X^i),$$

$$\mathcal{L}_{\text{perceptron}}(W) = \frac{1}{P}\sum_{i=1}^{P} W^T\left(F(X^i,Y^i) - F(X^i,Y^{*i})\right).$$

$$W \leftarrow W - \eta\left(F(X^i,Y^i) - F(X^i,Y^{*i})\right).$$

(but [LeCun et al. 1998] used non-linear factors)

Linearly Parameterized Factors + Hinge Loss = Max Margin Markov Networks



- **Linearly Parameterized Factor + Hinge loss**
  - [Altun et a. 2003, Taskar et al. 2003]

$$\mathcal{L}_{\text{hinge}}(W) = \frac{1}{P}\sum_{i=1}^{P} \max(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)) + \gamma ||W||^2.$$

$$\mathcal{L}_{\text{hinge}}(W) = \frac{1}{P}\sum_{i=1}^{P} \max\left(0, m + W^T \Delta F(X^i, Y^i)\right) + \gamma ||W||^2,$$

$$\Delta F(X^i, Y^i) = F(X^i, Y^i) - F(X^i, \bar{Y}^i)$$

Simple gradient descent rule:

$$\text{If } \Delta F(X^i, Y^i) > -m \text{ then } W \leftarrow W - \eta \Delta F(X^i, Y^i) - 2\gamma W$$

Can be performed in the dual (like an SVM)

Non-Linear Factors

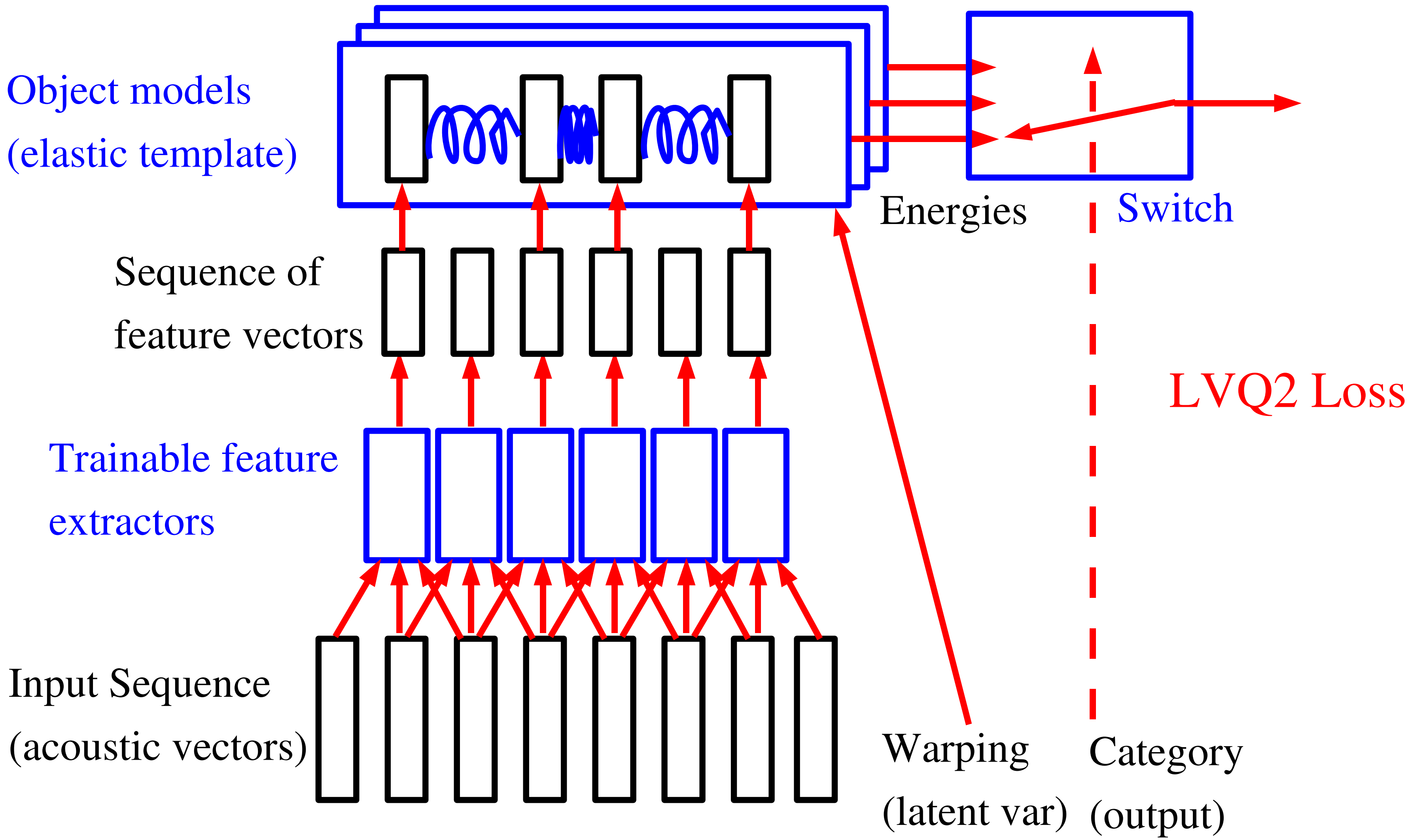
- **Energy-Based sequence labeling systems trained discriminatively have been used since the early 1990's**
- **Almost all of them used non-linear factors, such as multi-layer neural nets or mixtures of Gaussians.**
- **They were used mostly for speech and handwriting recognition**
- **There is a huge literature on the subject that has been somewhat ignored or forgotten by the NIPS and NLP communities.**
- **Why use non linear factors?**
  - :-(the loss function is non-convex
  - :-o You have to use simple gradient-based optimization algorithms, such as stochastic gradient descent (but that's what works best anyway, even in the convex case)
  - :-) linear factors simply don't cut it for speech and handwriting (including SVM-like linear combinations of kernel functions)

Deep Factors / Deep Graph: ASR with TDNN/HMM

- **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.
- **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)

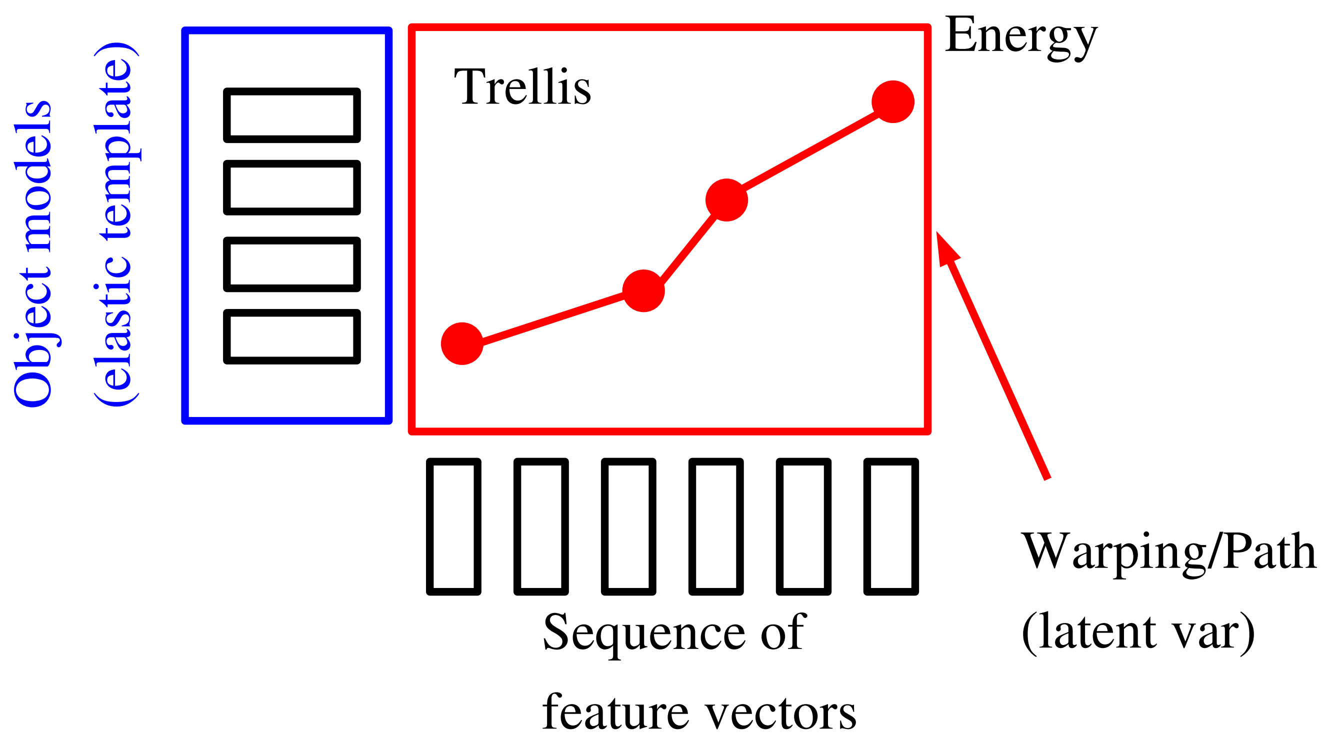
Example 1: Integrated Disc. Training with Sequence Alignment

- **Spoken word recognition with trainable elastic templates and trainable feature extraction [Driancourt&Bottou 1991, Bottou 1991, Driancourt 1994]**



Example: 1-D Constellation Model (a.k.a. Dynamic Time Warping)

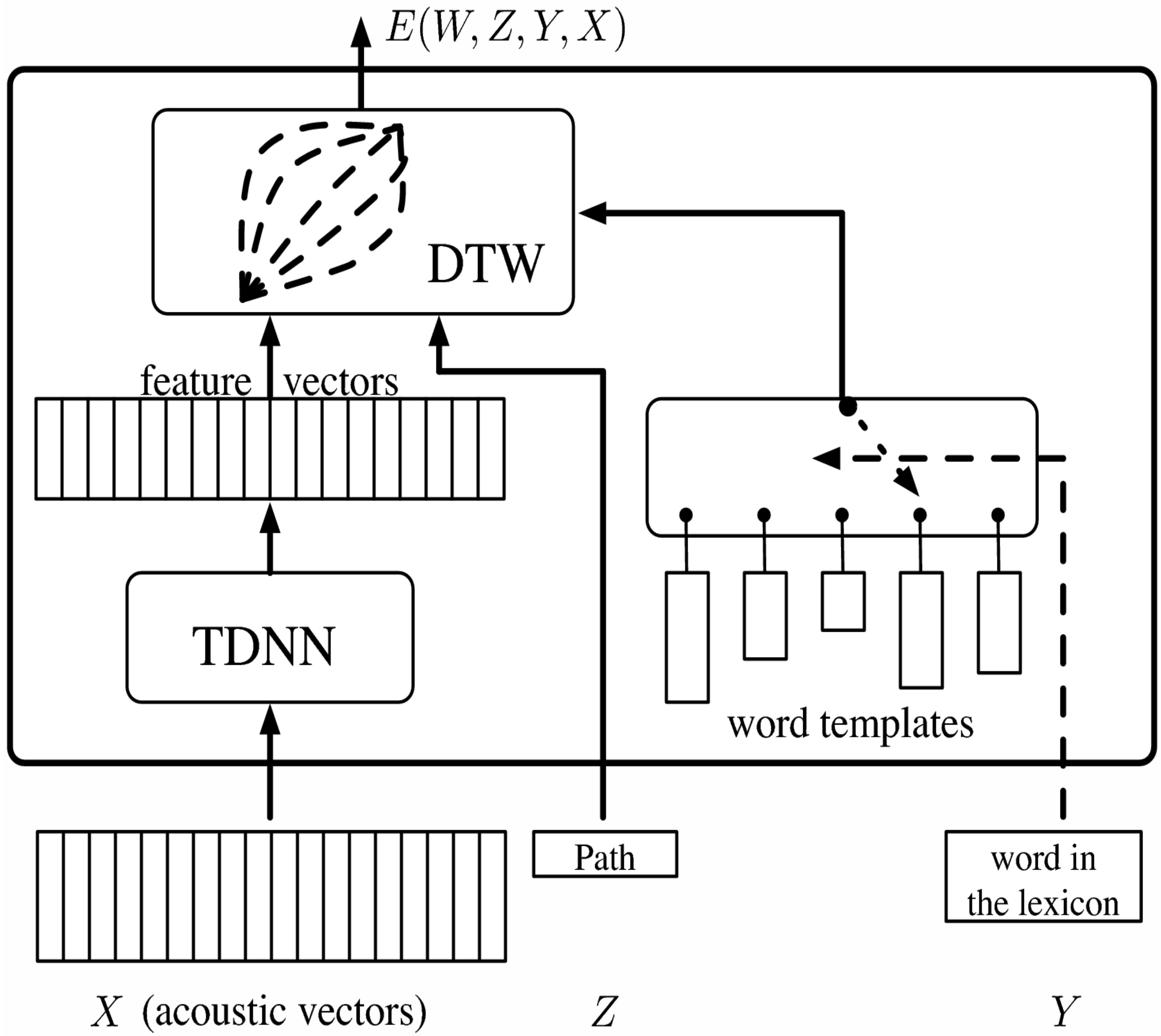
- **Spoken word recognition with trainable elastic templates and trainable feature extraction [Driancourt&Bottou 1991, Bottou 1991, Driancourt 1994]**
- **Elastic matching using dynamic time warping (Viterbi algorithm on a trellis).**
- **The corresponding EBFG is implicit (it changes for every new sample).**



Deep Factors / Deep Graph: ASR with TDNN/DTW

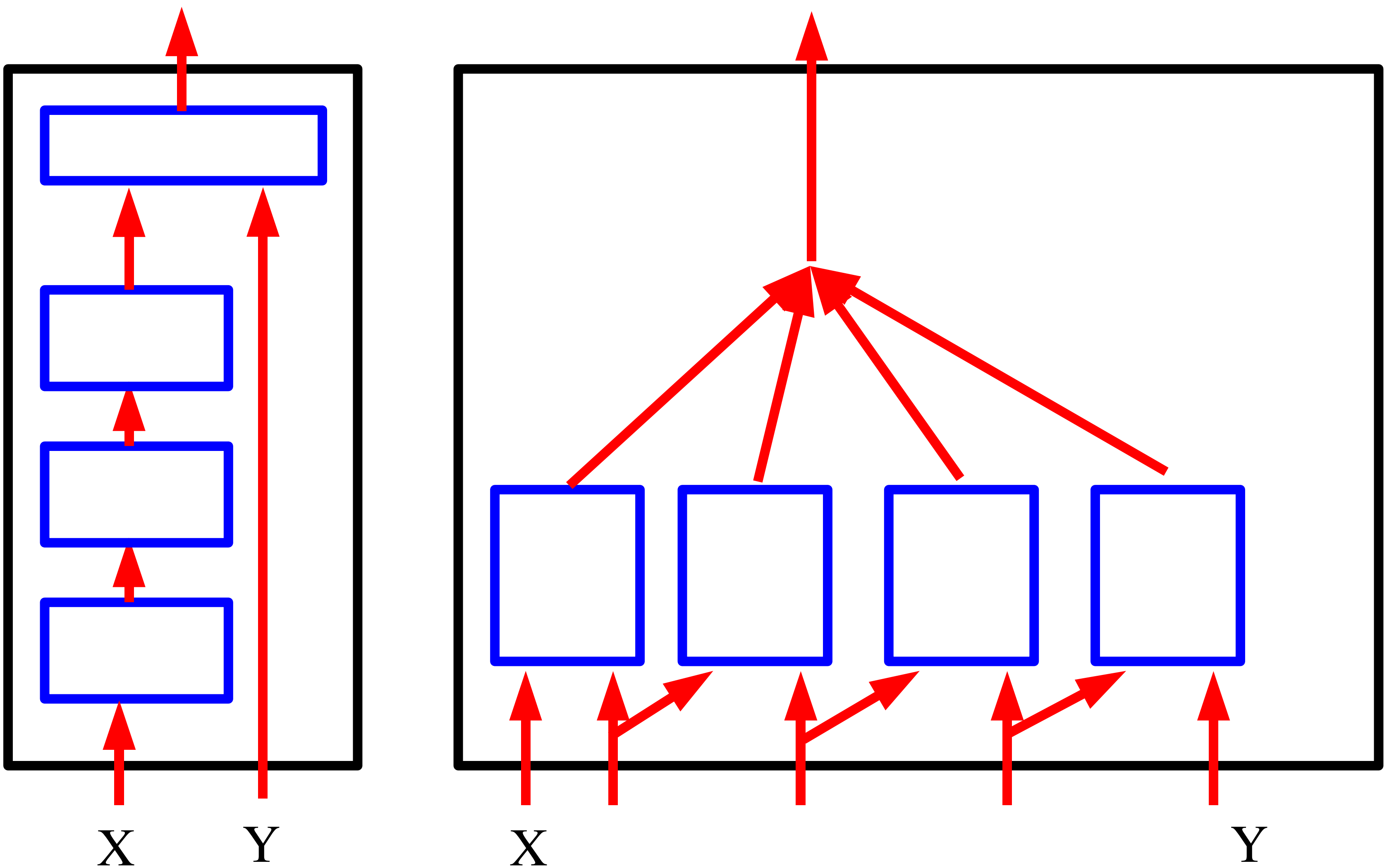
- **Trainable Automatic Speech Recognition system with convolutional nets (TDNN) and dynamic time warping (DTW)**

- **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)



Two types of “deep” architectures

- Factors are deep / graph is deep

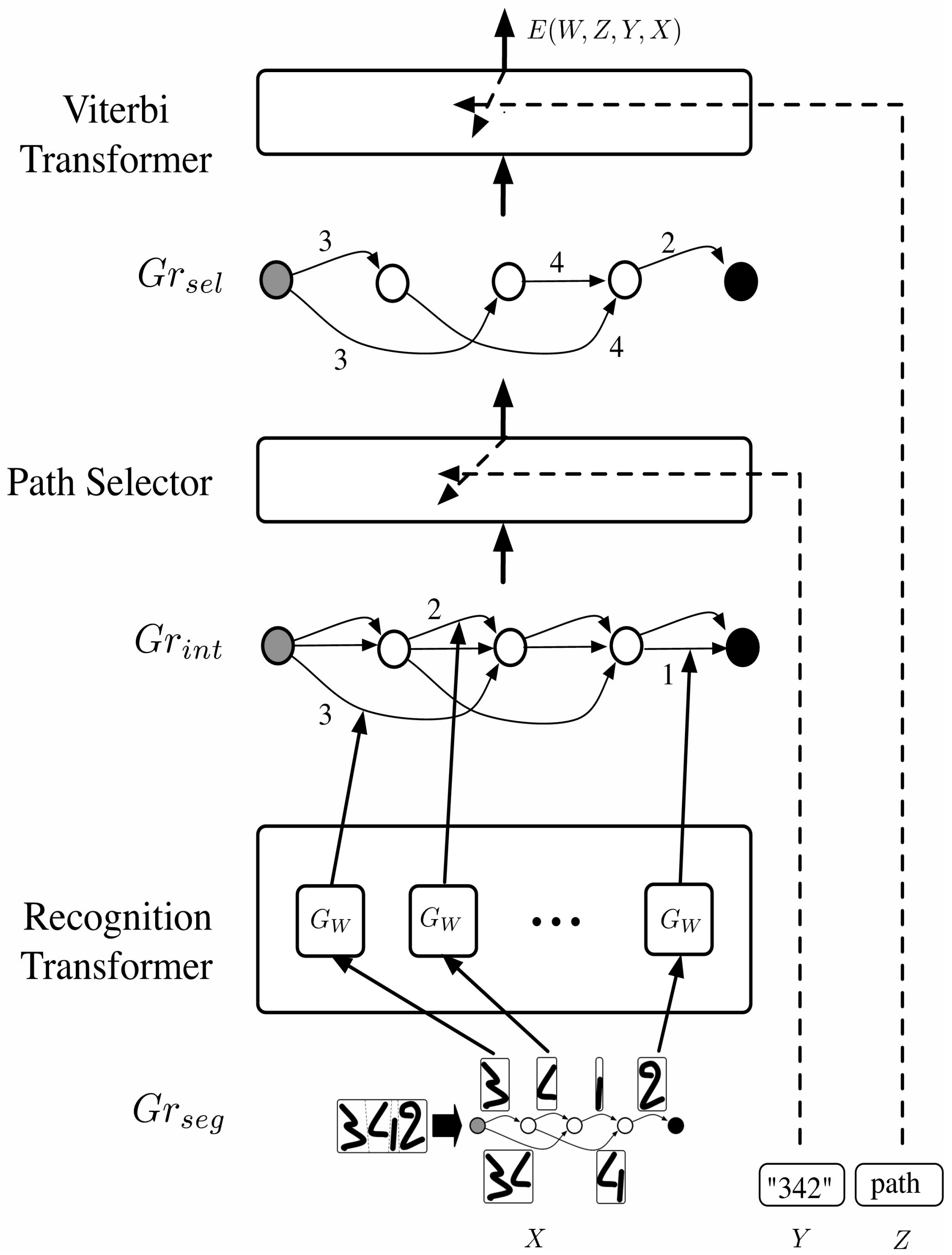


Complex Trellises: procedural representation of trellises

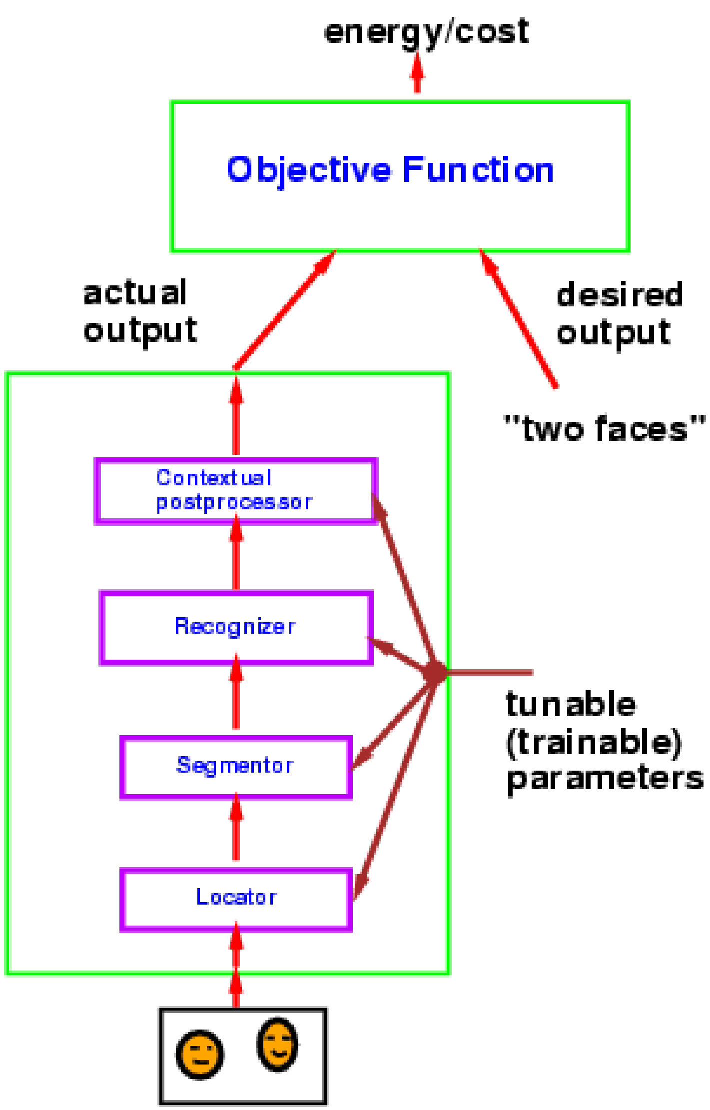
- **When the trellis is too large, we cannot store it in its entirety in memory.**
  - We must represent it proceduraly
- **The cleanest way to represent complex graphs proceduraly is through the formalism of finite-state transducer algebra**
  - [Mohri 1997, Pereira et al.]

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**

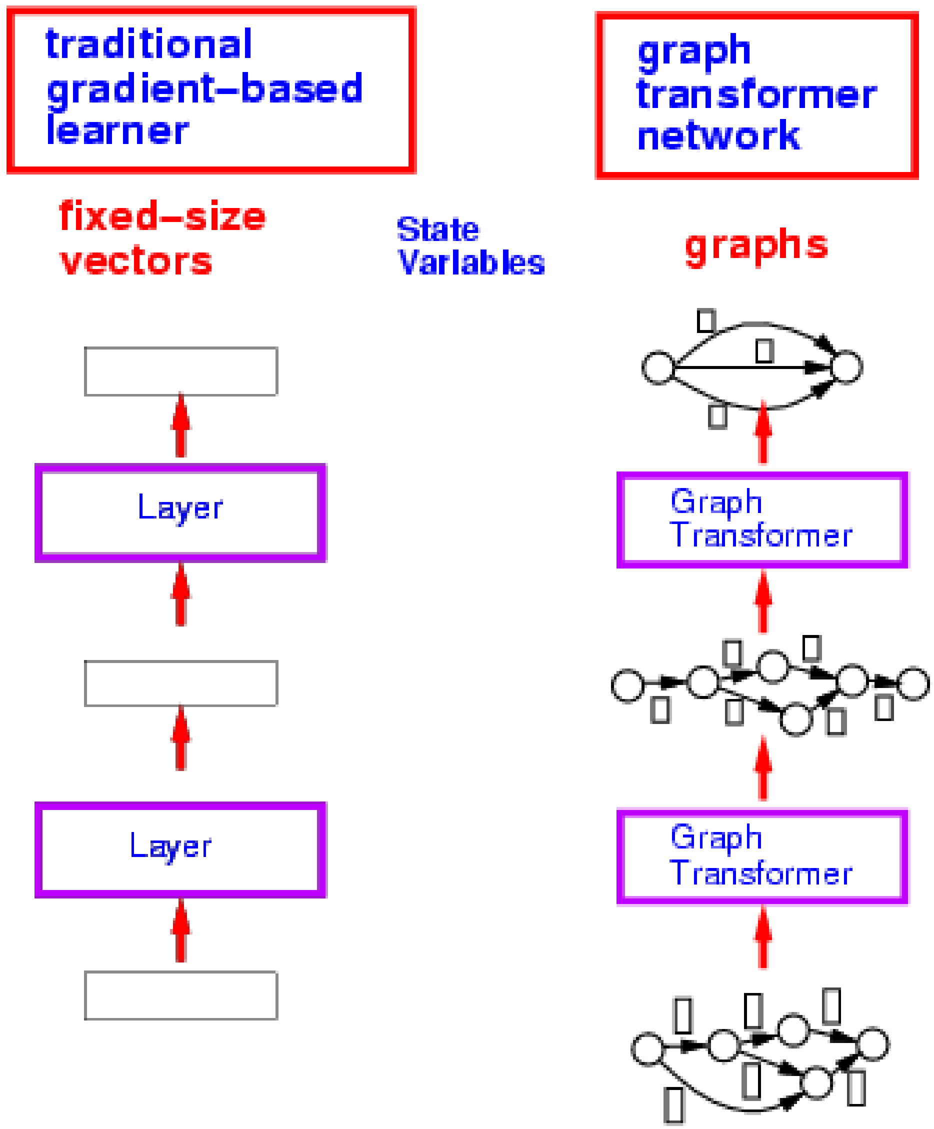


End-to-End Learning.



- **Making every single module in the system trainable.**
- **Every module is trained simultaneously so as to optimize a global loss function.**

Using Graphs instead of Vectors.

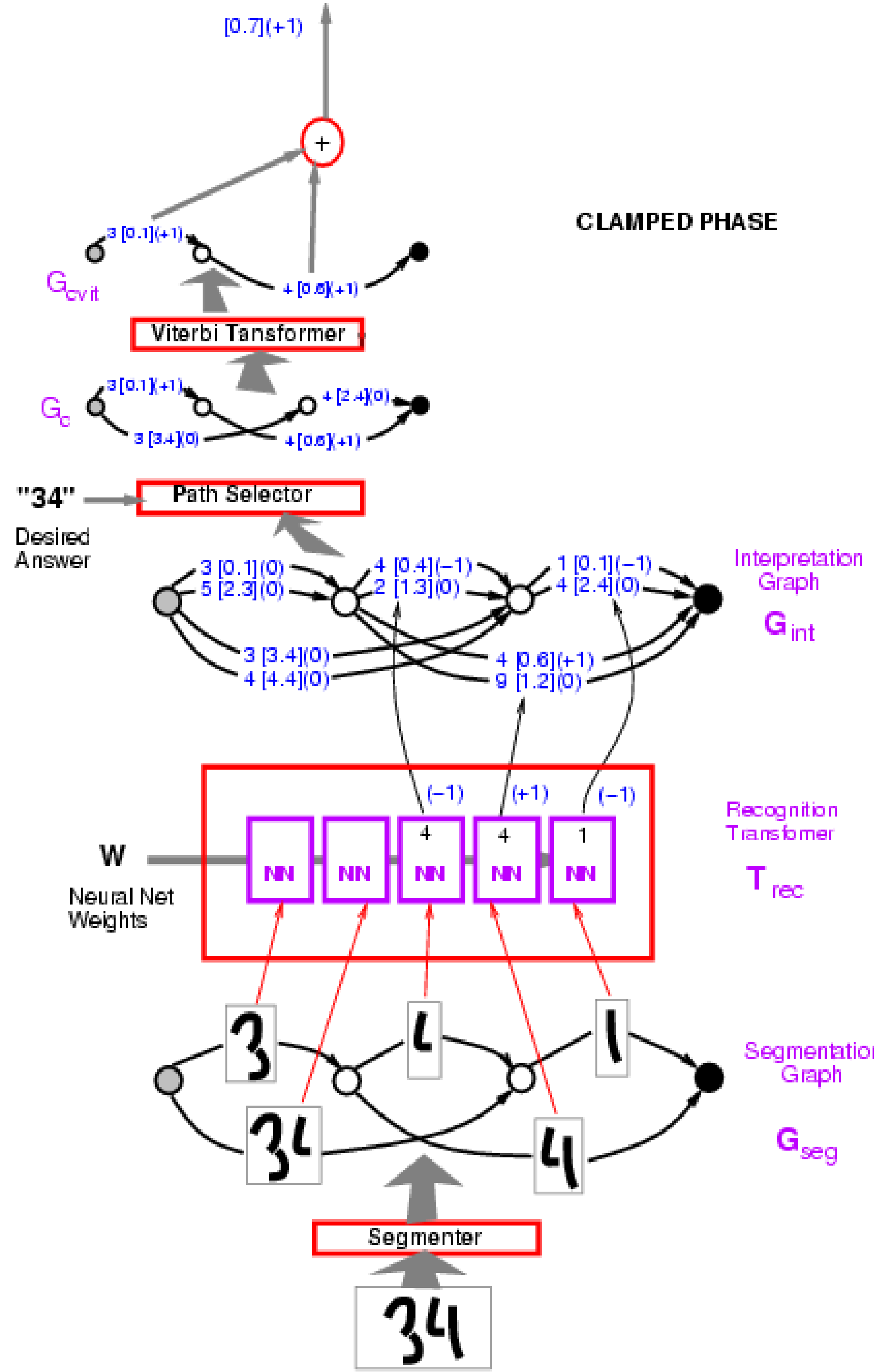


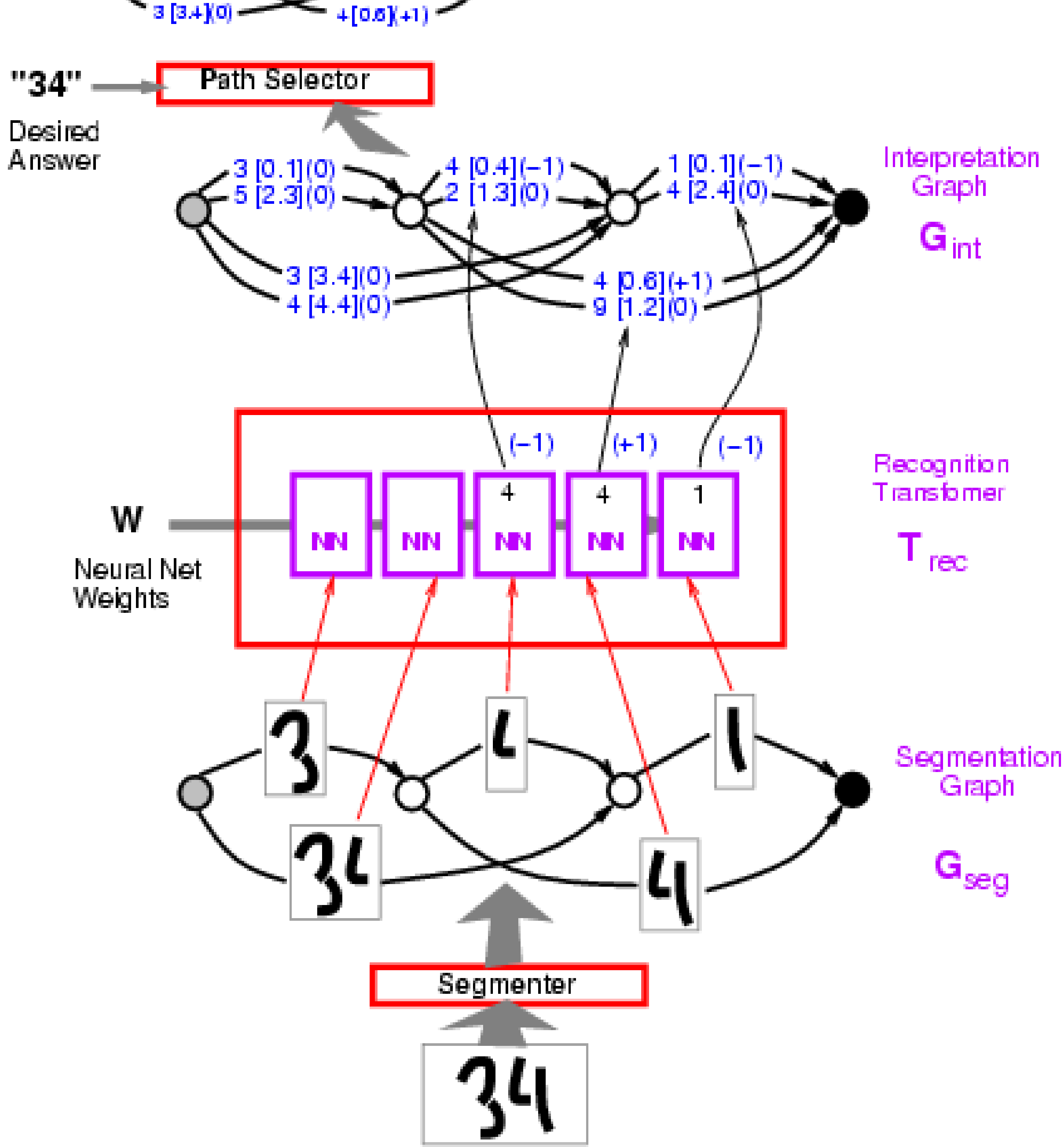
- Whereas traditional learning machines manipulate fixed-size vectors, **Graph Transformer Networks** manipulate graphs.

Graph Transformer Networks

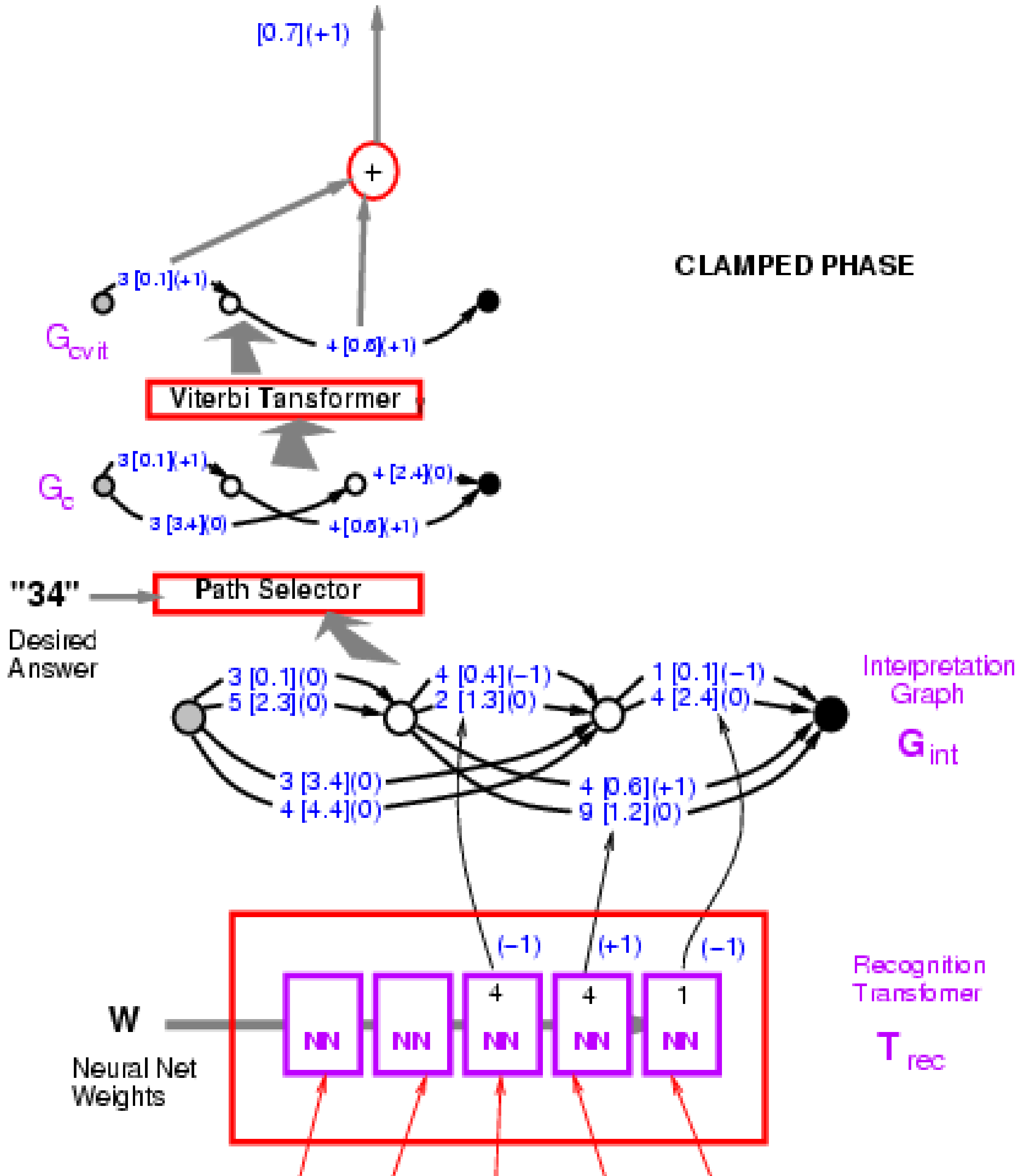
- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the energy of the desired answer:**

$$E(W, Y, X)$$





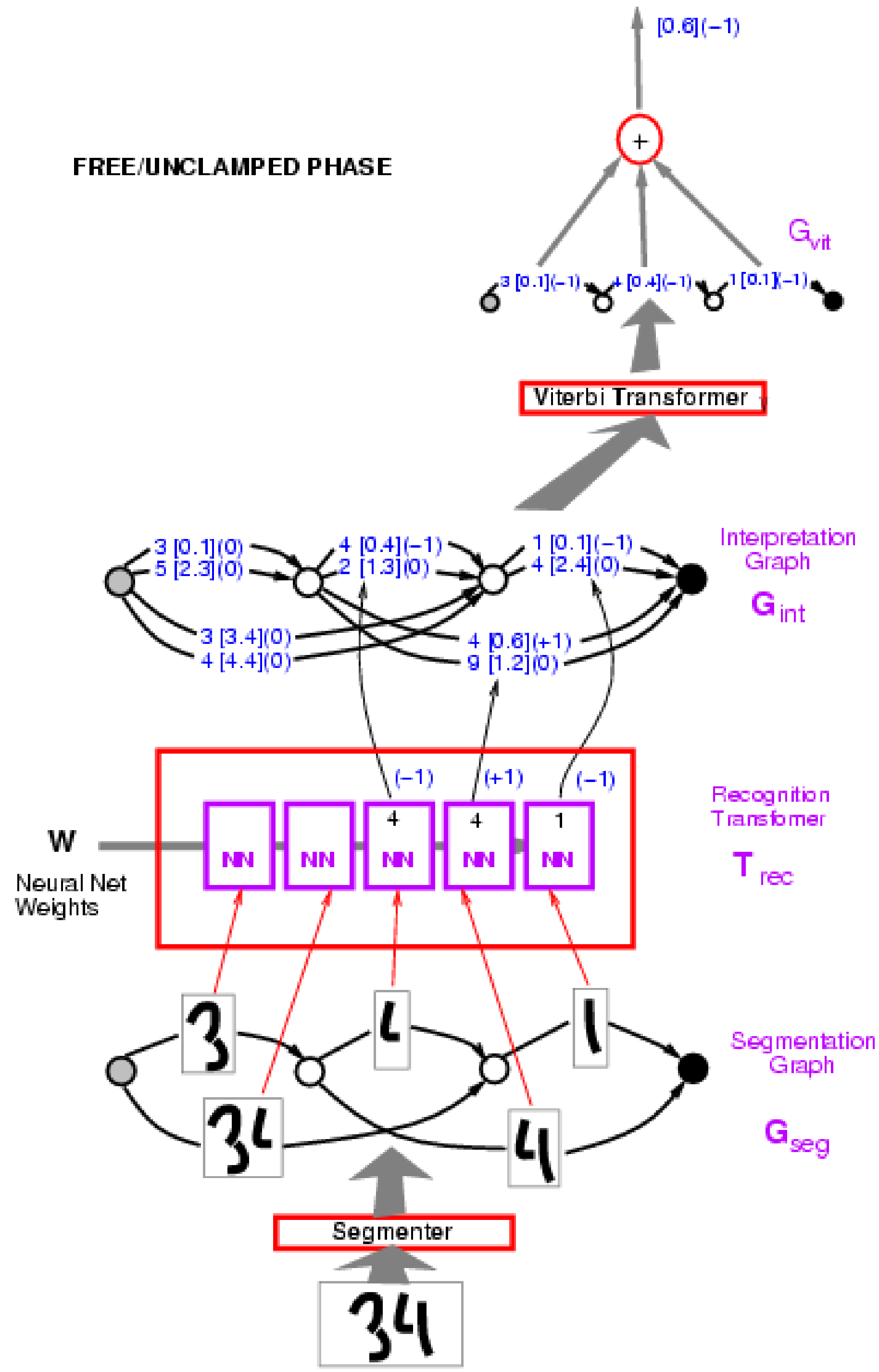
"34"
Desired Answer
Path Selector
3 [0.1](0)
5 [2.3](0)
4 [0.4](−1)
2 [1.3](0)
1 [0.1](−1)
4 [2.4](0)
3 [3.4](0)
4 [4.4](0)
4 [0.6](+1)
9 [1.2](0)
Interpretation Graph
G int
W
Neural Net Weights
(−1)
(+1)
(−1)
4
4
1
NN
NN
NN
NN
NN
Recognition Transfomer
T rec
Segmentation Graph
G seg
Segmenter


[0.7](+1)
+
CLAMPED PHASE
3 [0.1](+1)
4 [0.6](+1)
G_{cvit}
Viterbi Tansformer
3 [0.1](+1)
4 [2.4](0)
G_c
3 [3.4](0)
4 [0.6](+1)
"34"
Path Selector
Desired Answer
3 [0.1](0)
5 [2.3](0)
4 [0.4](−1)
2 [1.3](0)
1 [0.1](−1)
4 [2.4](0)
Interpretation Graph
G_{int}
3 [3.4](0)
4 [4.4](0)
4 [0.6](+1)
9 [1.2](0)
(−1)
(+1)
(−1)
4
4
1
NN
NN
NN
NN
NN
W
Neural Net Weights
Recognition Transfomer
T_{rec}

Graph Transformer Networks

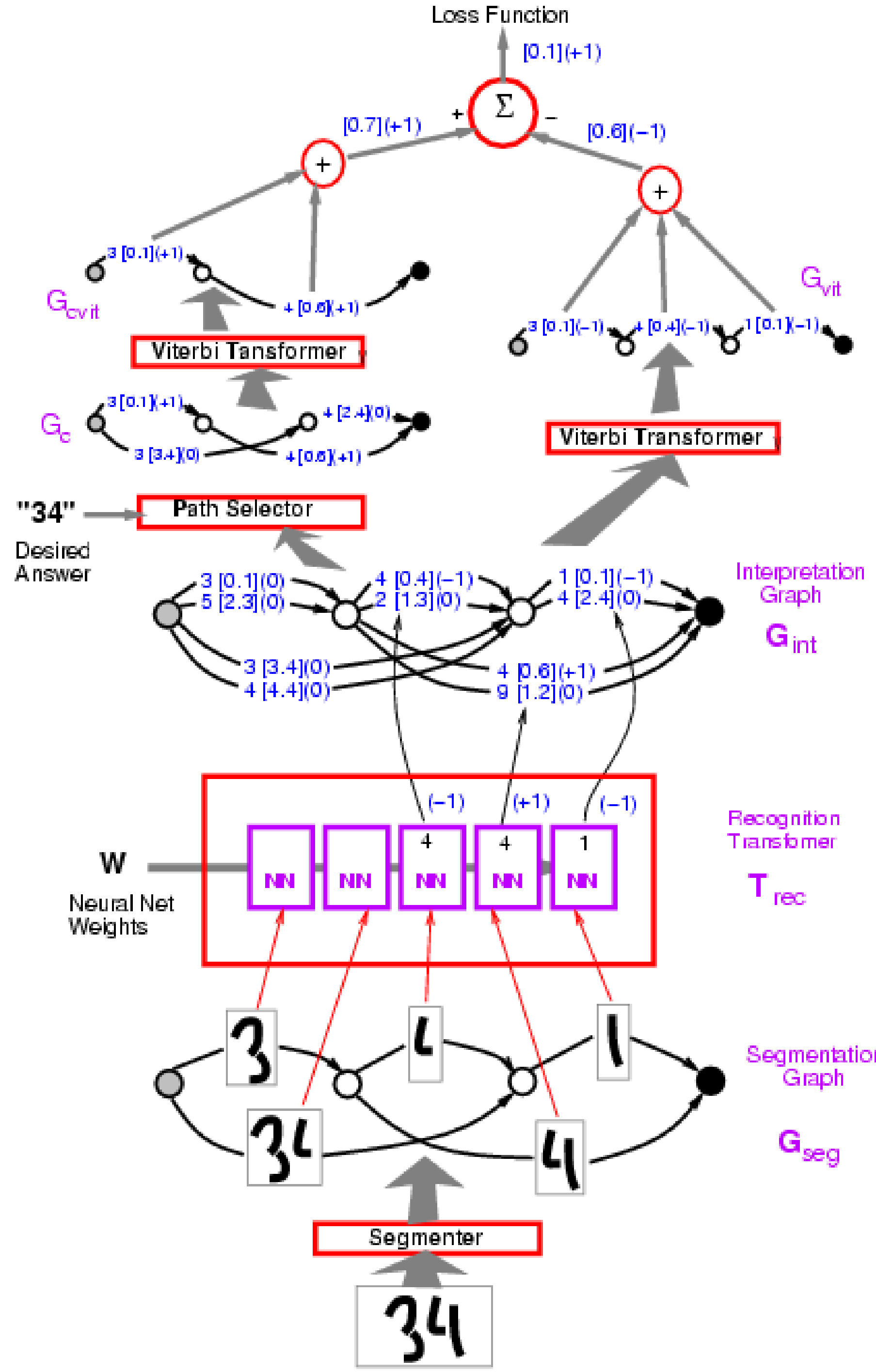
- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the constrastive term:**

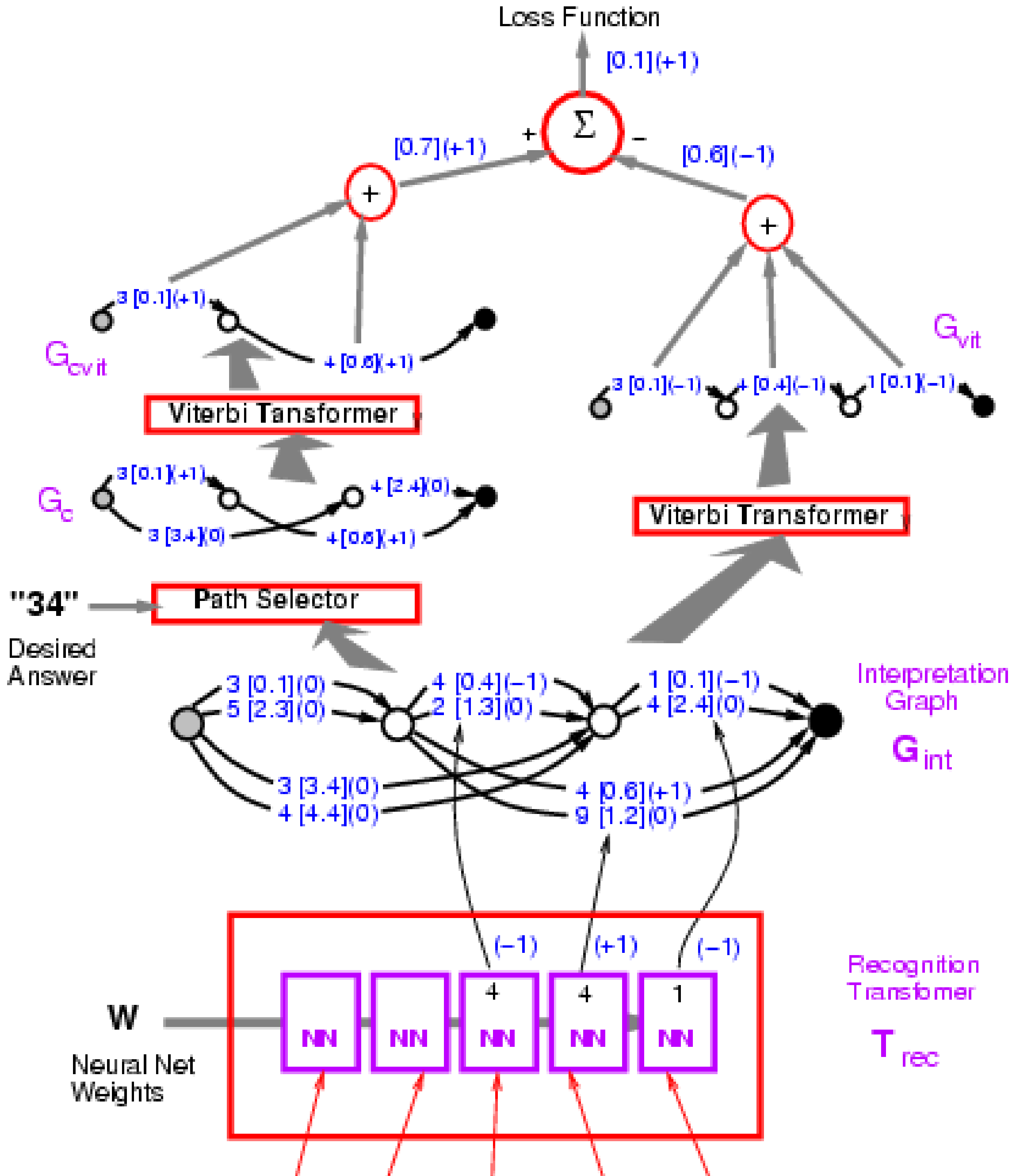
$$E(W, \check{Y}, X)$$



Graph Transformer Networks

- **Example: Perceptron loss**
- **Loss = Energy of desired answer – Energy of best answer.**
  - (no margin)

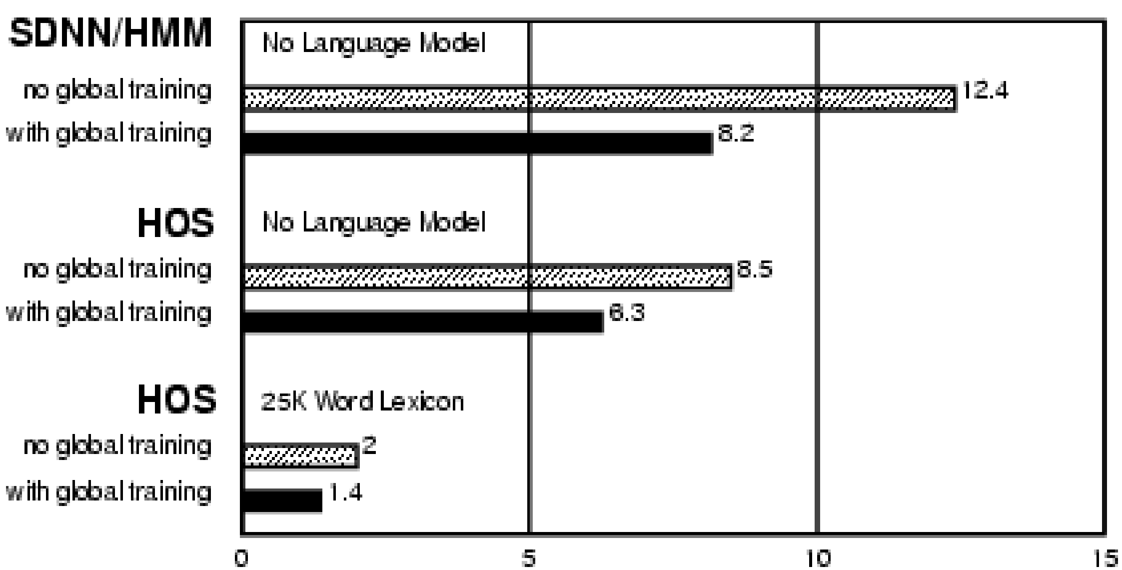


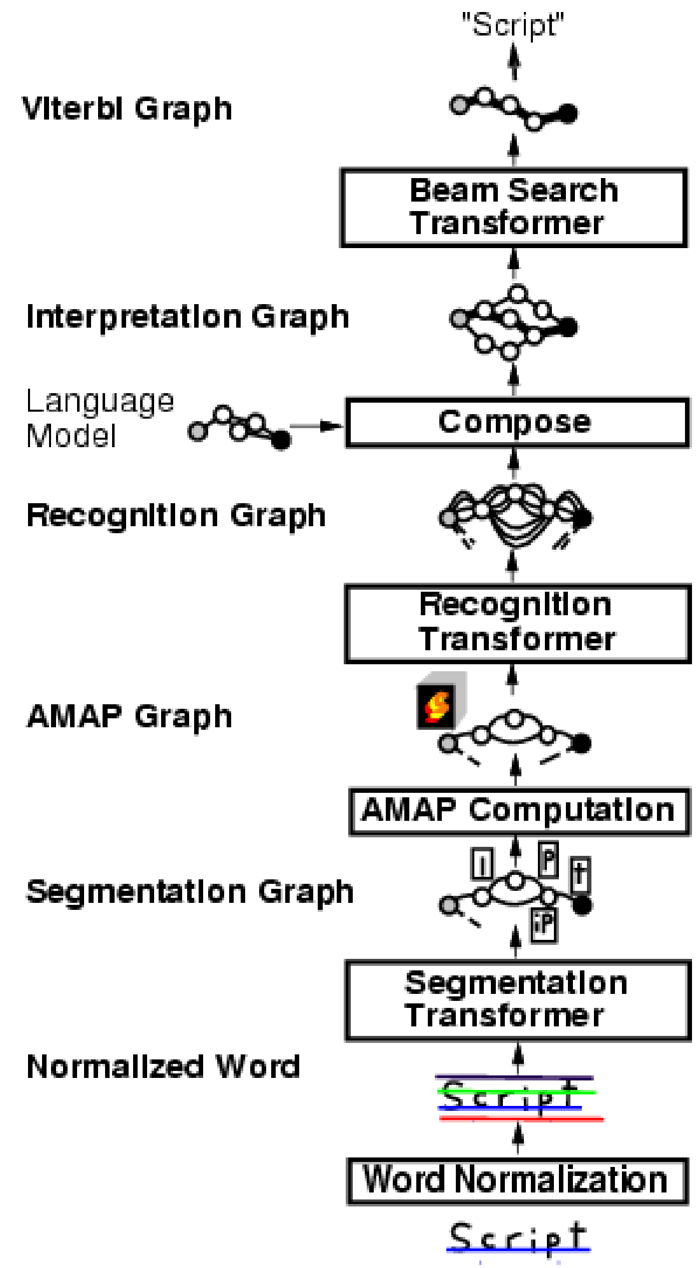


Loss Function
[0.1](+1)
Σ
[0.7](+1)
[0.6](−1)
Gcvit
Viterbi Tansformer
Gc
"34"
Desired Answer
Path Selector
Gvit
Viterbi Transformer
Interpretation Graph
Gint
3 [0.1](0)
5 [2.3](0)
4 [0.4](−1)
2 [1.3](0)
1 [0.1](−1)
4 [2.4](0)
3 [3.4](0)
4 [4.4](0)
4 [0.6](+1)
9 [1.2](0)
W
Neural Net Weights
(−1)
(+1)
(−1)
NN
Recognition Transfomer
T rec

Global Training Helps

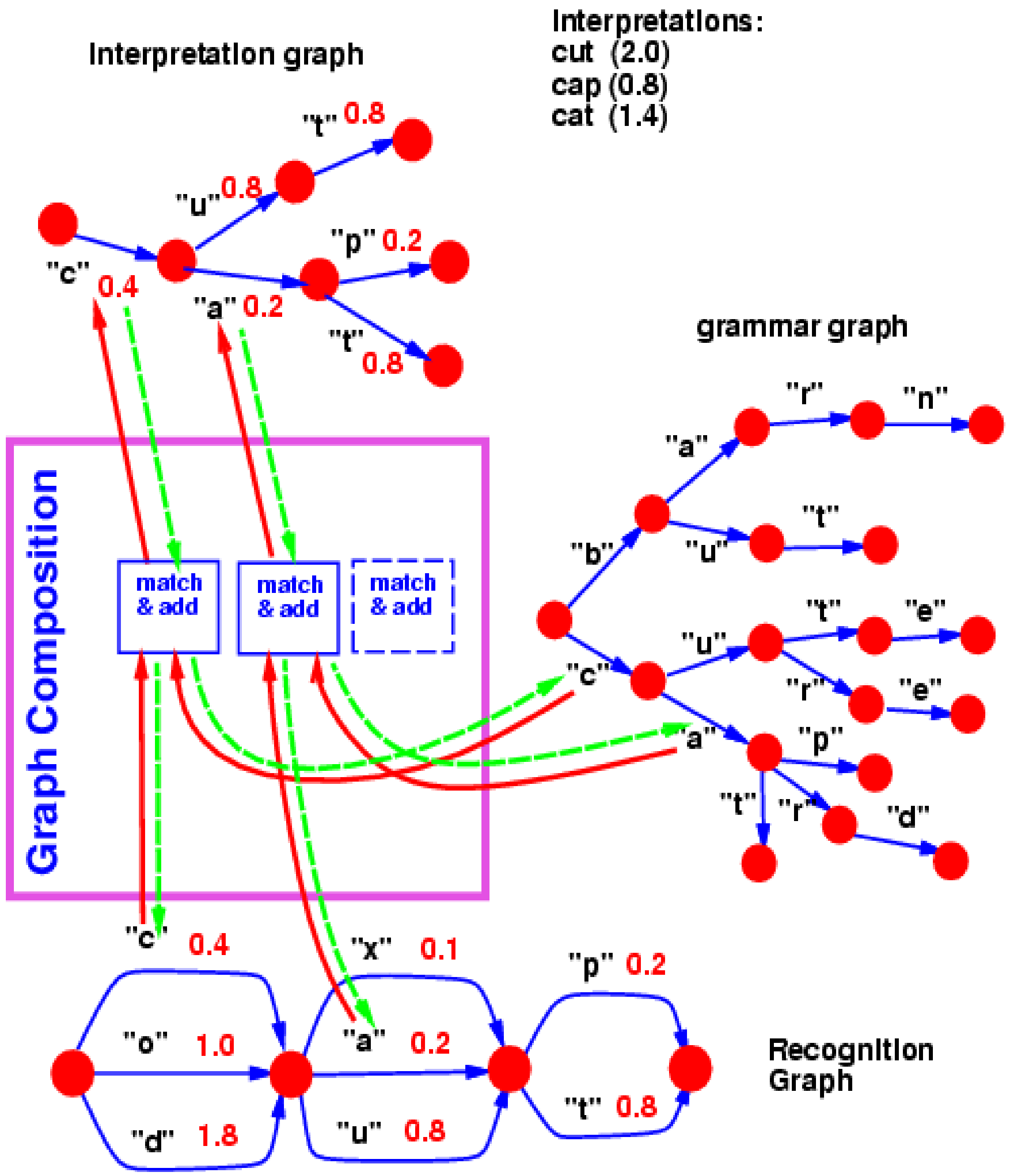
- **Pen-based handwriting recognition (for tablet computer)**
  - [Bengio&LeCun 1995]
  - Trained with NLL loss (aka MMI)





Graph Composition, Transducers.

- The composition of two graphs can be computed, the same way the dot product between two vectors can be computed.
- General theory: semi-ring algebra on weighted finite-state transducers and acceptors.



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.**