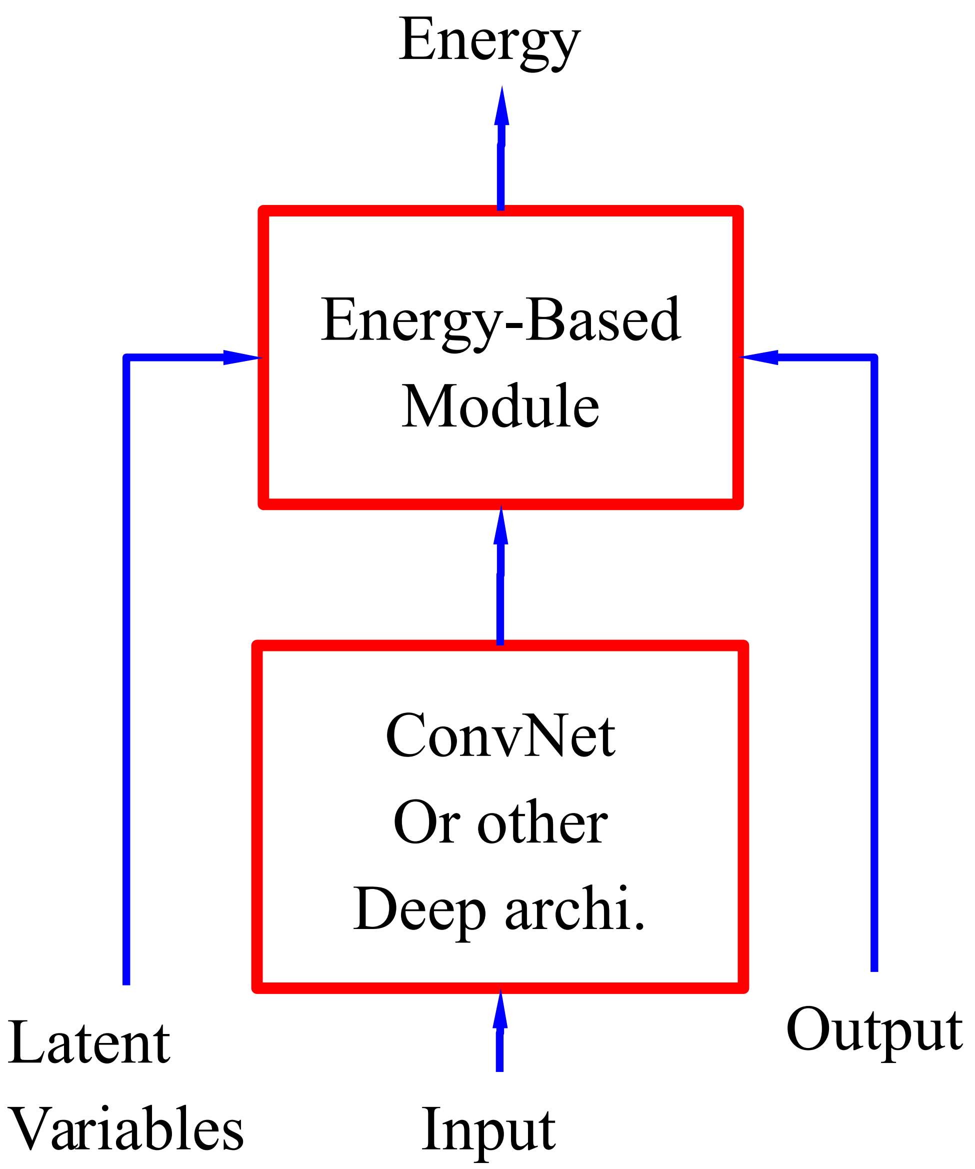
Energy-Based Learning

Lecture 06

Yann Le Cun
Facebook AI Research,
Center for Data Science, NYU
Courant Institute of Mathematical Sciences, NYU
http://yann.lecun.com

Global (End-to-End) Learning: Energy-Based Models.



Making every single module in the system trainable.

Every module is trained simultaneously so as to optimize a global loss function.

Includes the feature extractor, the recognizer, and the contextual post-processor (graphical model)

Problem: back-propagating gradients through the graphical model.

Energy-Based Models

Highly popular methods in the Machine Learning and Natural Language Processing Communities have their roots in Speech and Handwriting Recognition

- Structured Perceptron, Conditional Random Fields, and related learning models for “structured prediction” are descendants of discriminative learning methods for speech recognition and word-level handwriting recognition methods from the early 90's

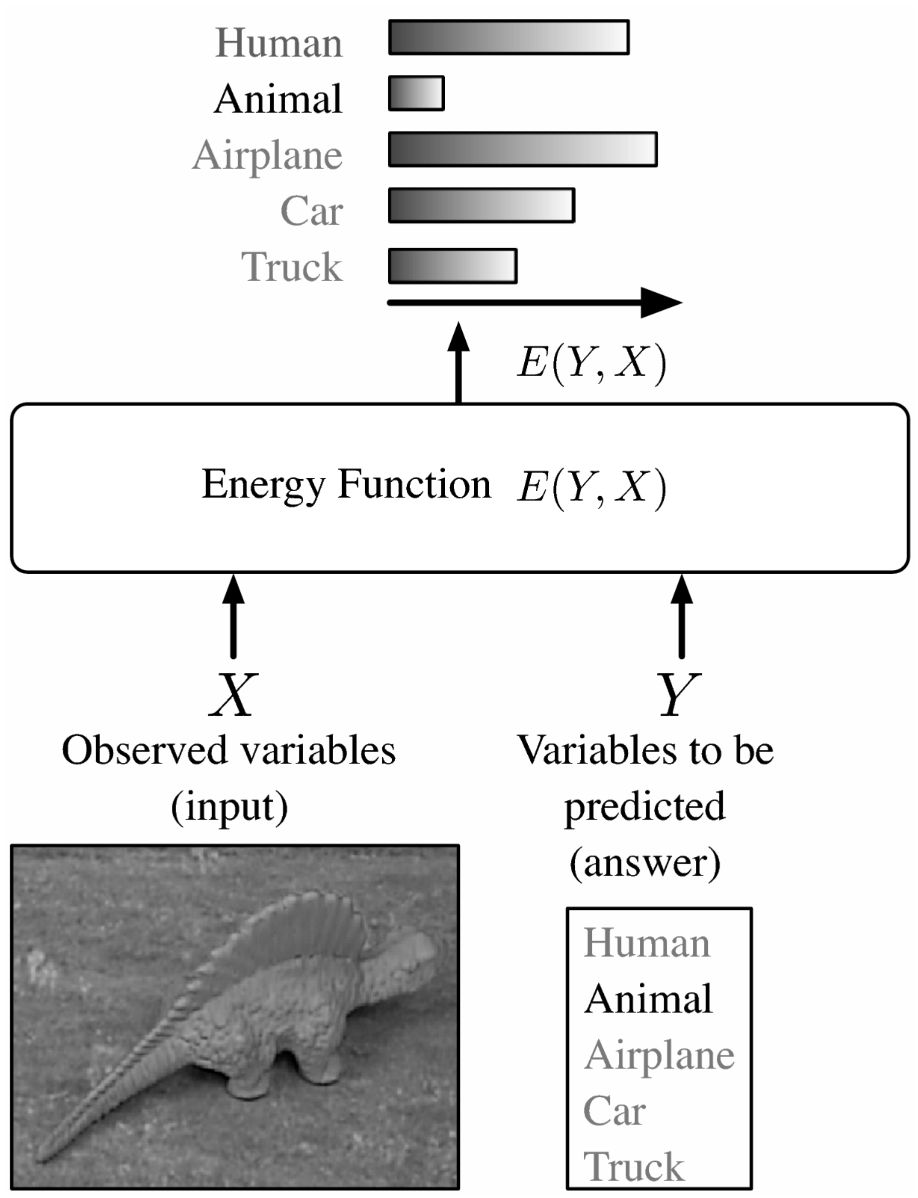
A Tutorial and Energy-Based Learning:

- [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]
- Structured Perceptron [Collins 2001]
- Conditional Random Fields [Lafferty & al 2001]
- Max Margin Markov Nets [Altun & al 2003, Taskar & al 2003]

Energy-Based Models for Decision Making

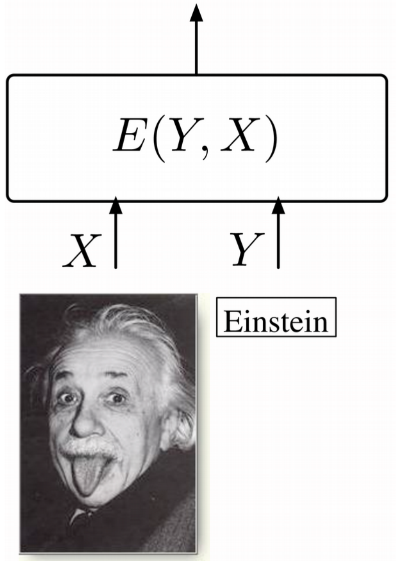

Human
Animal
Airplane
Car
Truck
$E(Y, X)$
Energy Function $E(Y, X)$
X
Observed variables (input)
Y
Variables to be predicted (answer)
Human
Animal
Airplane
Car
Truck

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

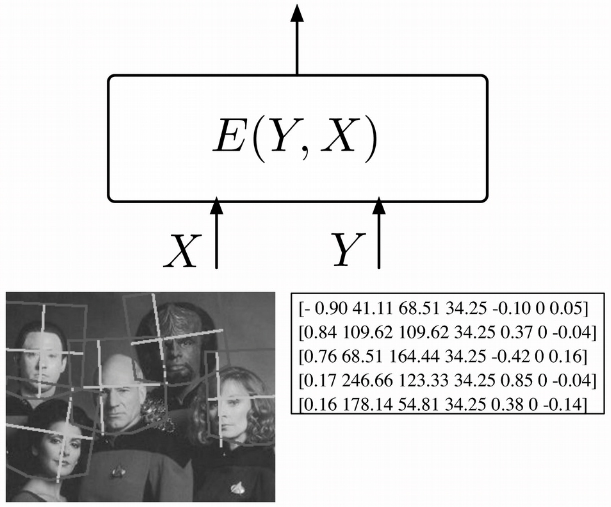
$$Y^* = \operatorname{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$

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

But Inference Might Be Complicated

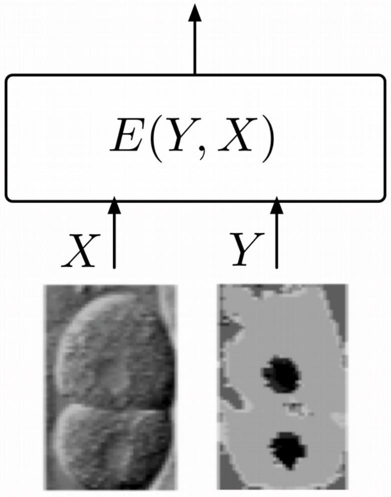


(a)

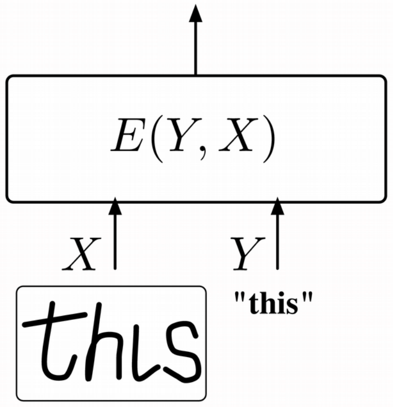


```
[- 0.90 41.11 68.51 34.25 -0.10 0 0.05]
[0.84 109.62 109.62 34.25 0.37 0 -0.04]
[0.76 68.51 164.44 34.25 -0.42 0 0.16]
[0.17 246.66 123.33 34.25 0.85 0 -0.04]
[0.16 178.14 54.81 34.25 0.38 0 -0.14]
```

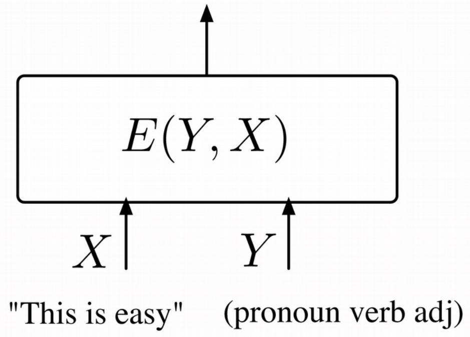
(b)



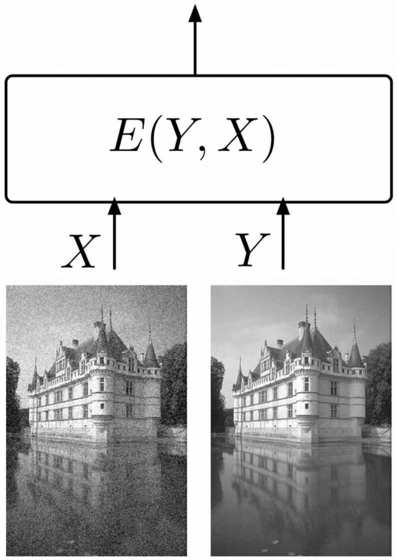
(c)



(d)



(e)



(f)

- 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.

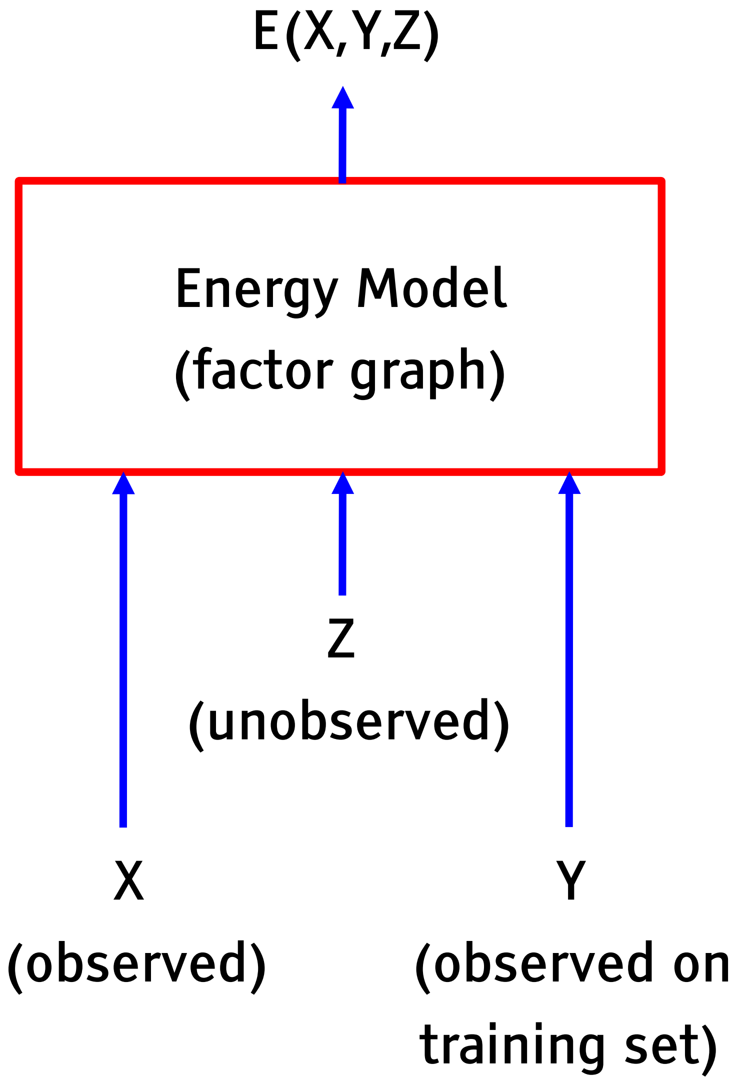
$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}},$$

Partition function

Inverse temperature

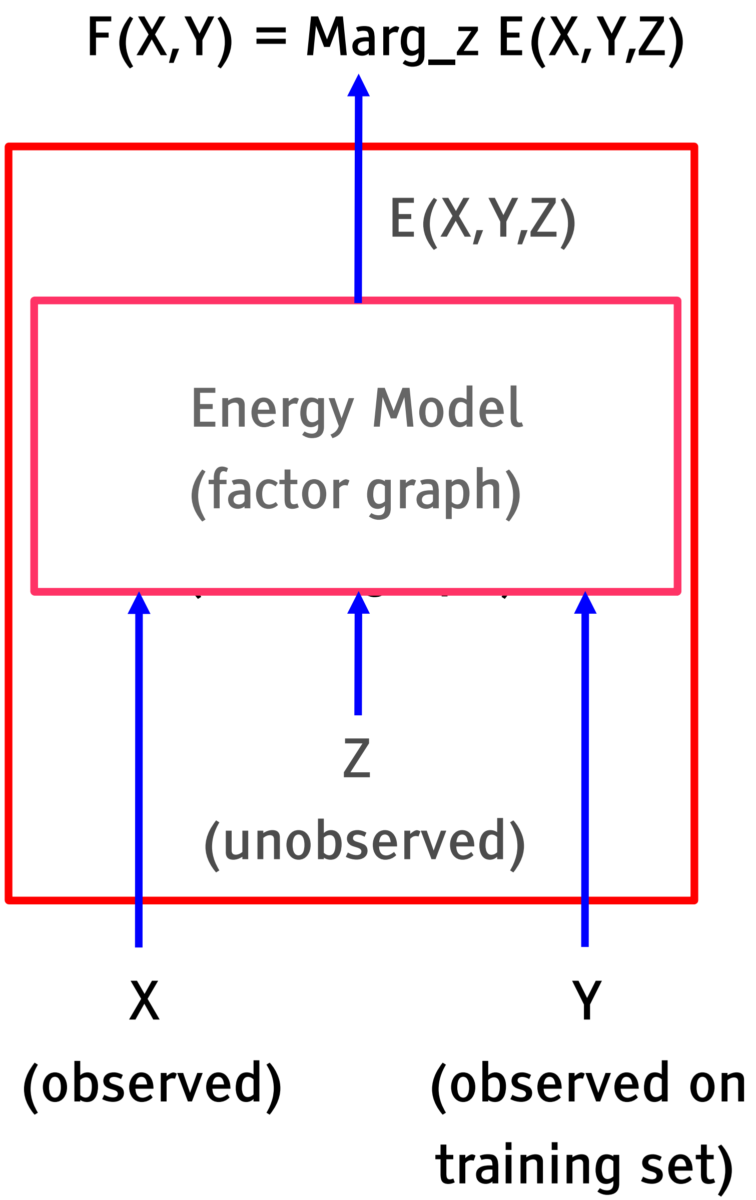
Integrating Deep Learning and Structured Prediction

- **Deep Learning systems can be assembled into factor graphs**
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- **Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X**



Integrating Deep Learning and Structured Prediction

- **Deep Learning systems can be assembled into factor graphs**
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- **Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X**
  - F(X,Y) = MIN_z E(X,Y,Z)
  - F(X,Y) = -log SUM_z exp[-E(X,Y,Z)]



Handwriting recognition Sequence labeling

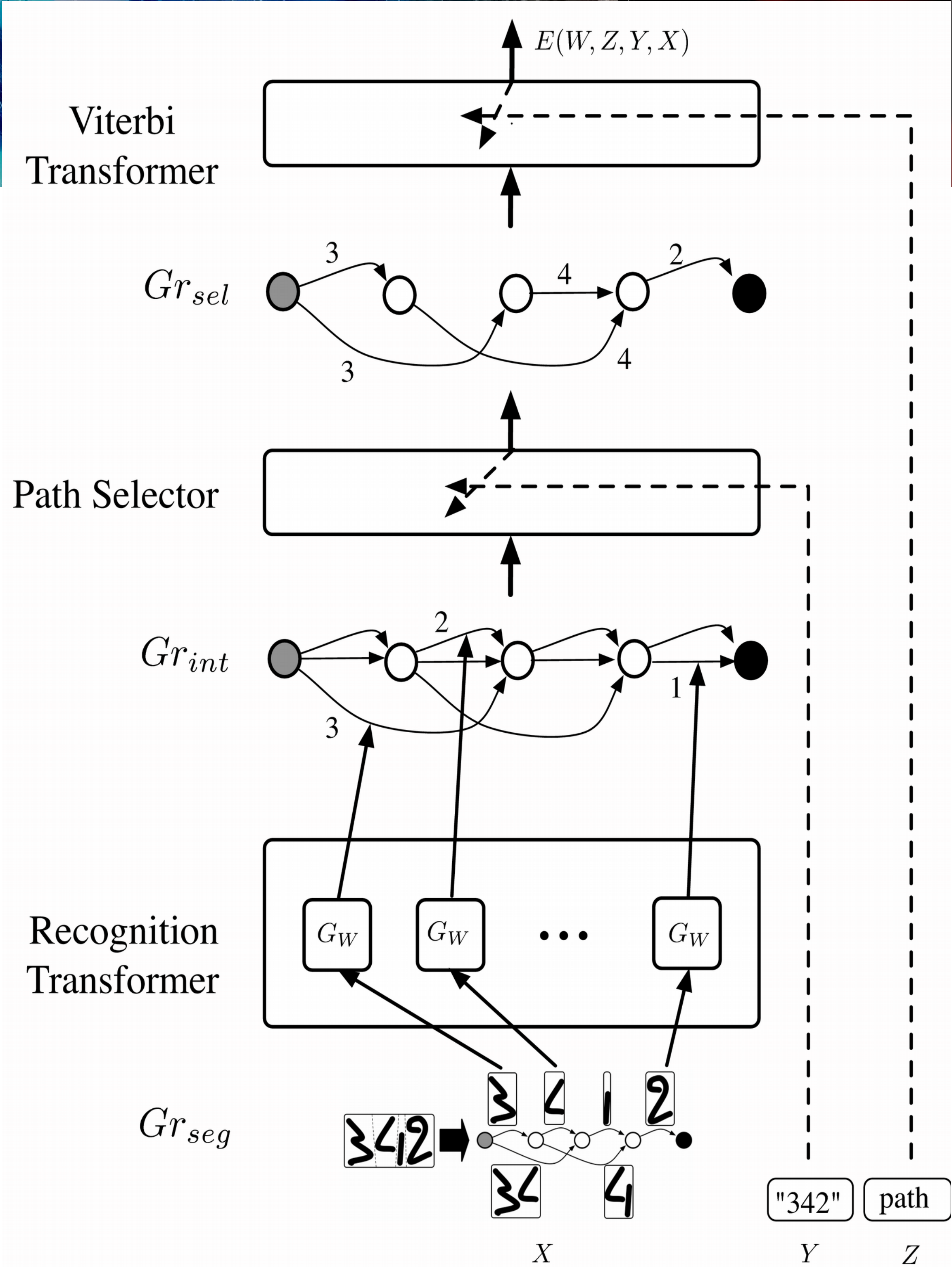
integrated segmentation and recognition of sequences.

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, Proc IEEE 1998]

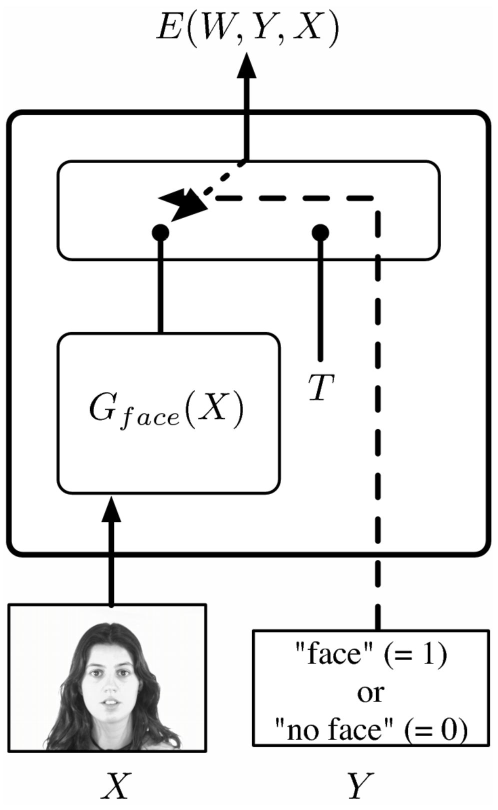


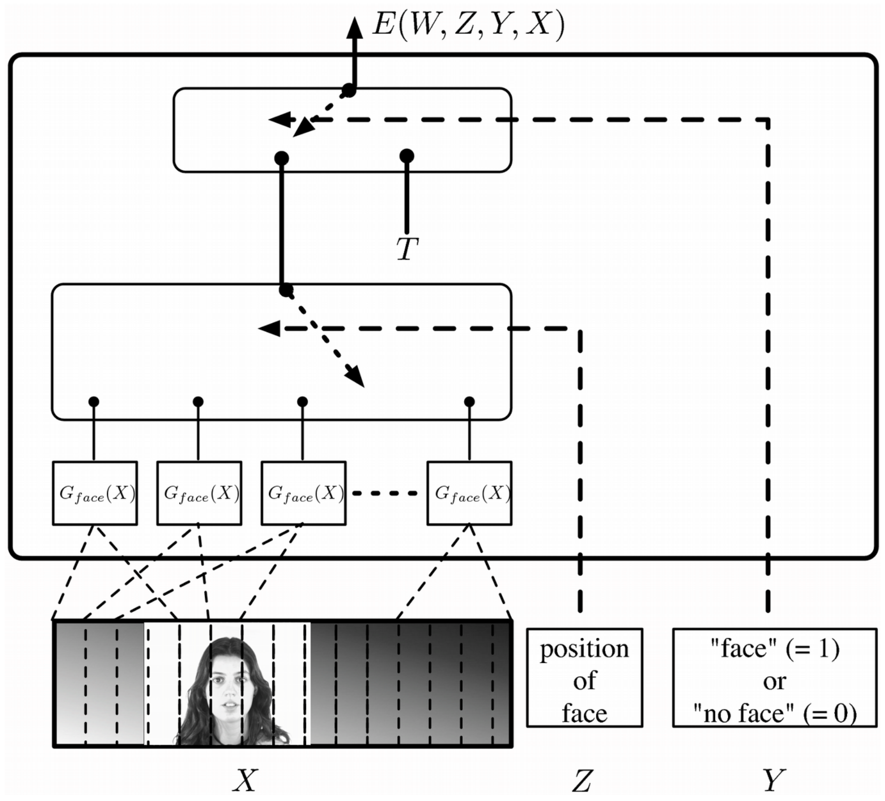
Example of Latent Variable Models: object detection

The energy includes “hidden” variables Z whose value is never given to us

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

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





Latent Variable Models

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

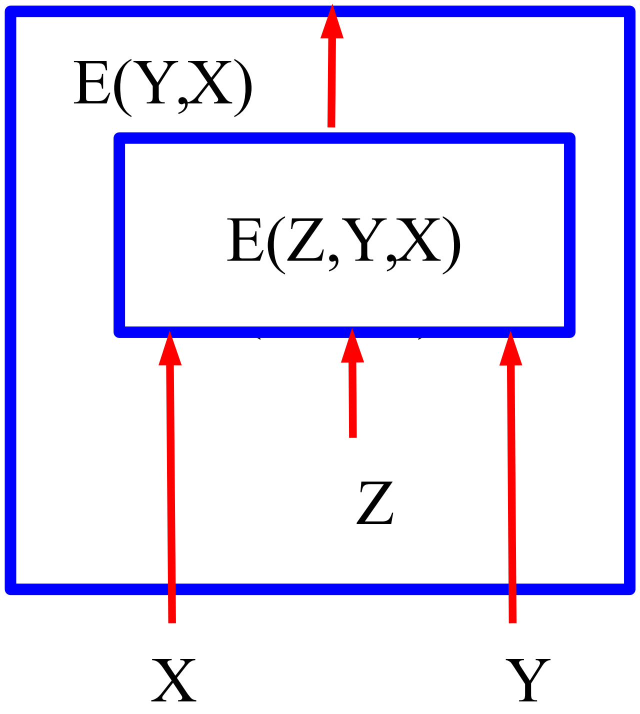
- We can minimize the energy over those latent variables
- We can also "marginalize" the energy over the latent variables

Minimization over latent variables:

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

Marginalization over latent variables:

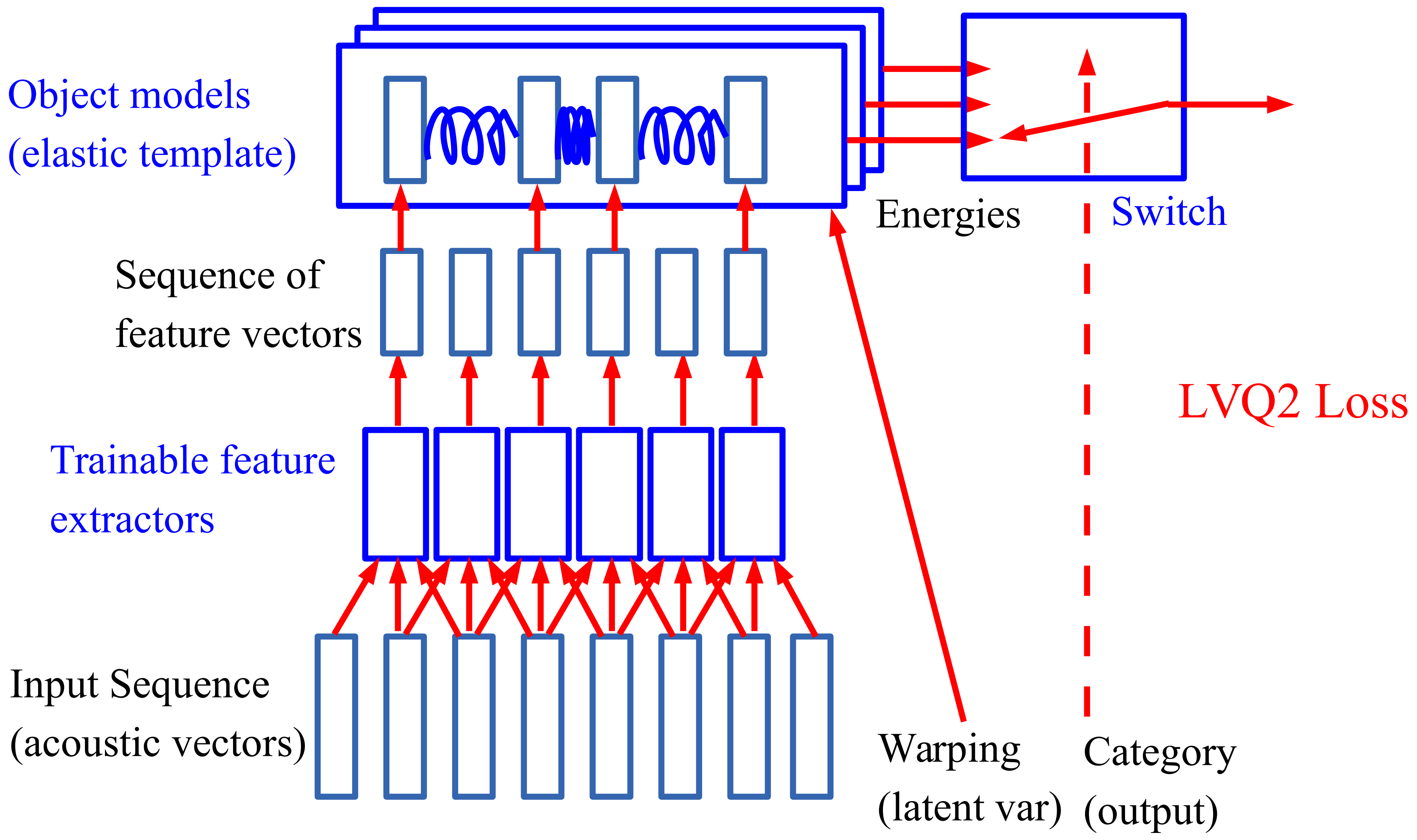
$$E(X, Y) = -\frac{1}{\beta} \log \int_{z \in \mathcal{Z}} e^{-\beta E(z, Y, X)}$$



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

Example 1: Integrated Training with Sequence Alignment

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

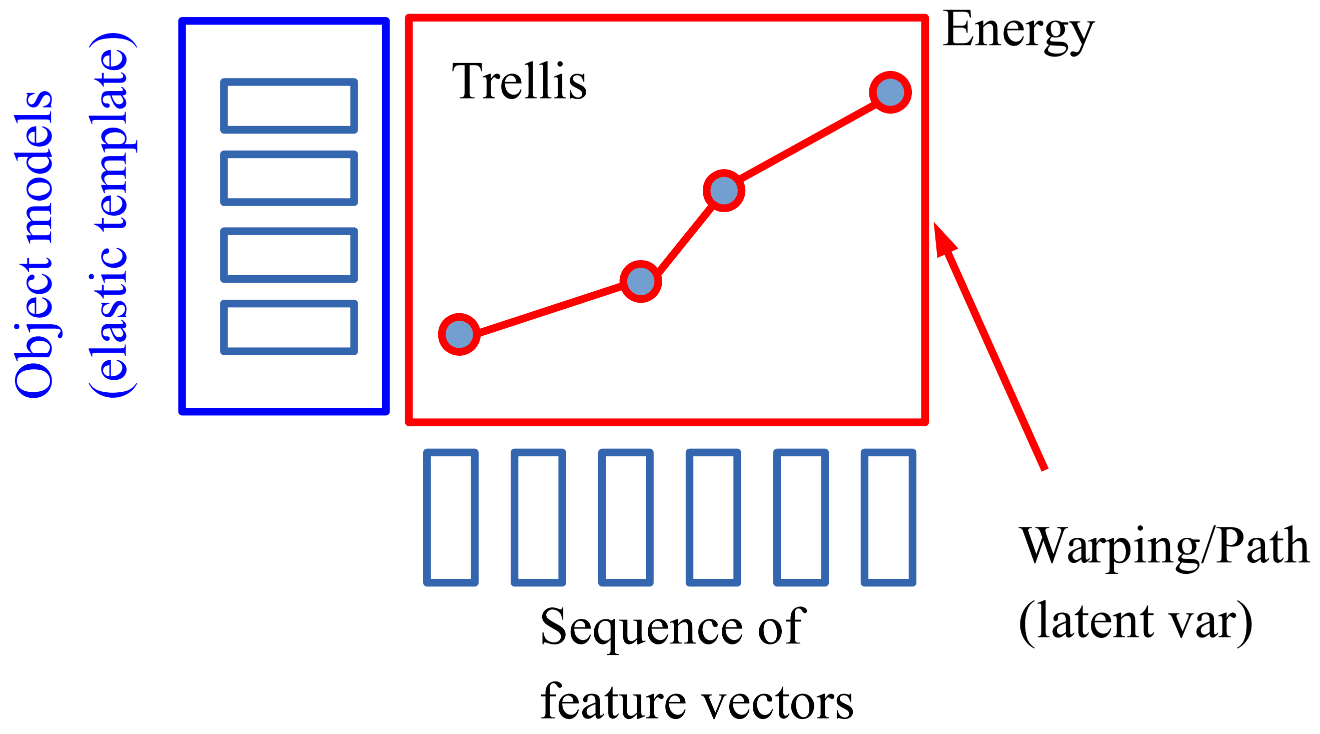


Example: 1-D Constellation Model (a.k.a. Dynamic Time Warping)

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).



The Oldest Example of Structured Prediction

Trainable Automatic Speech Recognition system with a convolutional net (TDNN) and dynamic time warping (DTW)

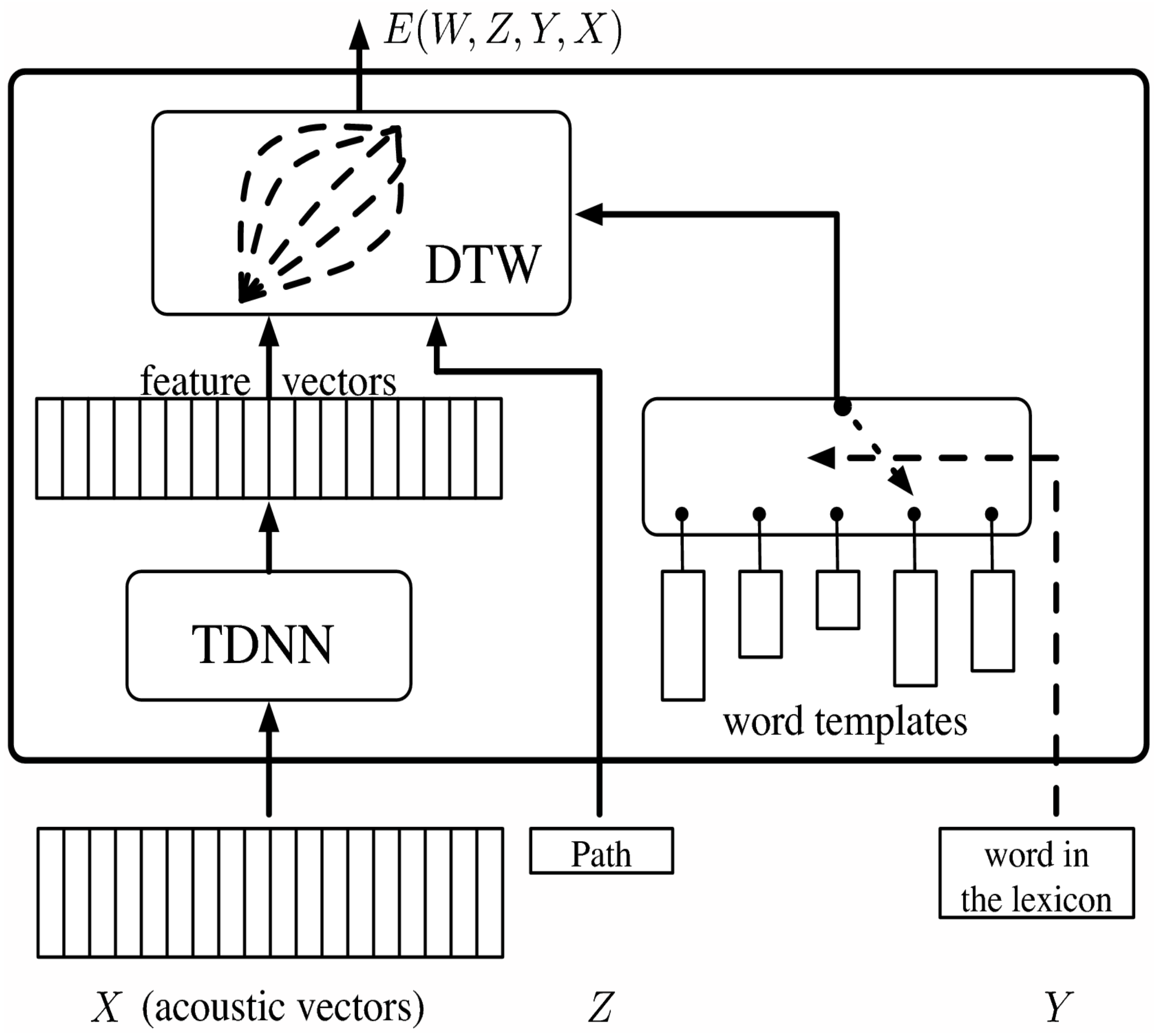
The feature extractor and the structured classifier are trained simultanesously in an integrated fashion.

with the LVQ2 Loss :

- Driancourt and Bottou's speech recognizer (1991)

with NLL:

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



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.
- **Object Recognition/Scene Parsing:** the segmentation of the image into components (objects, parts,...)

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:

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

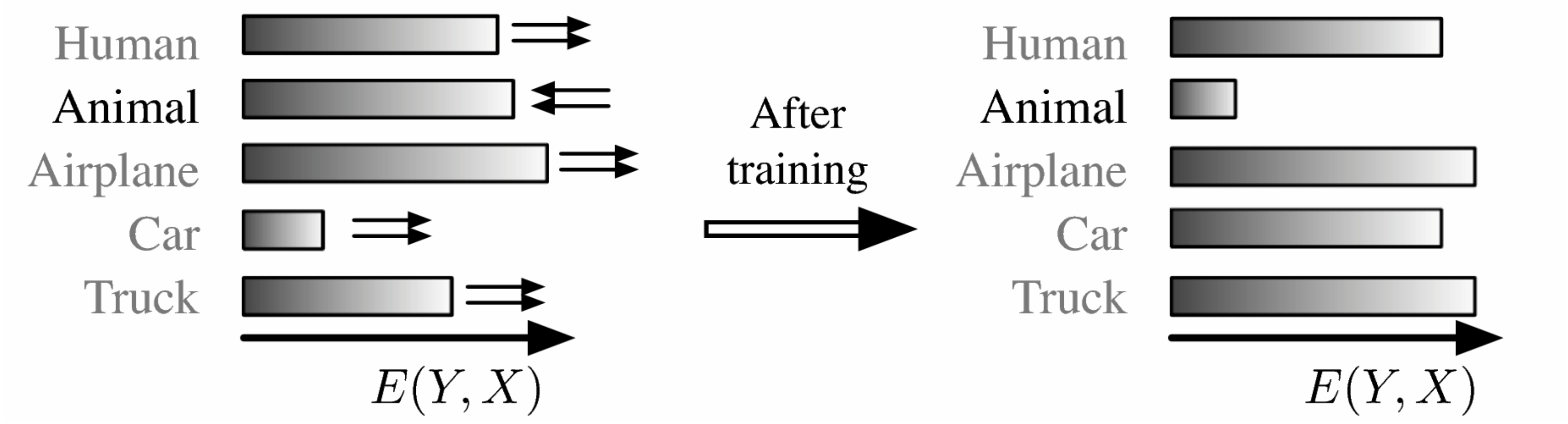
Reduces to traditional minimization when Beta->infinity

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

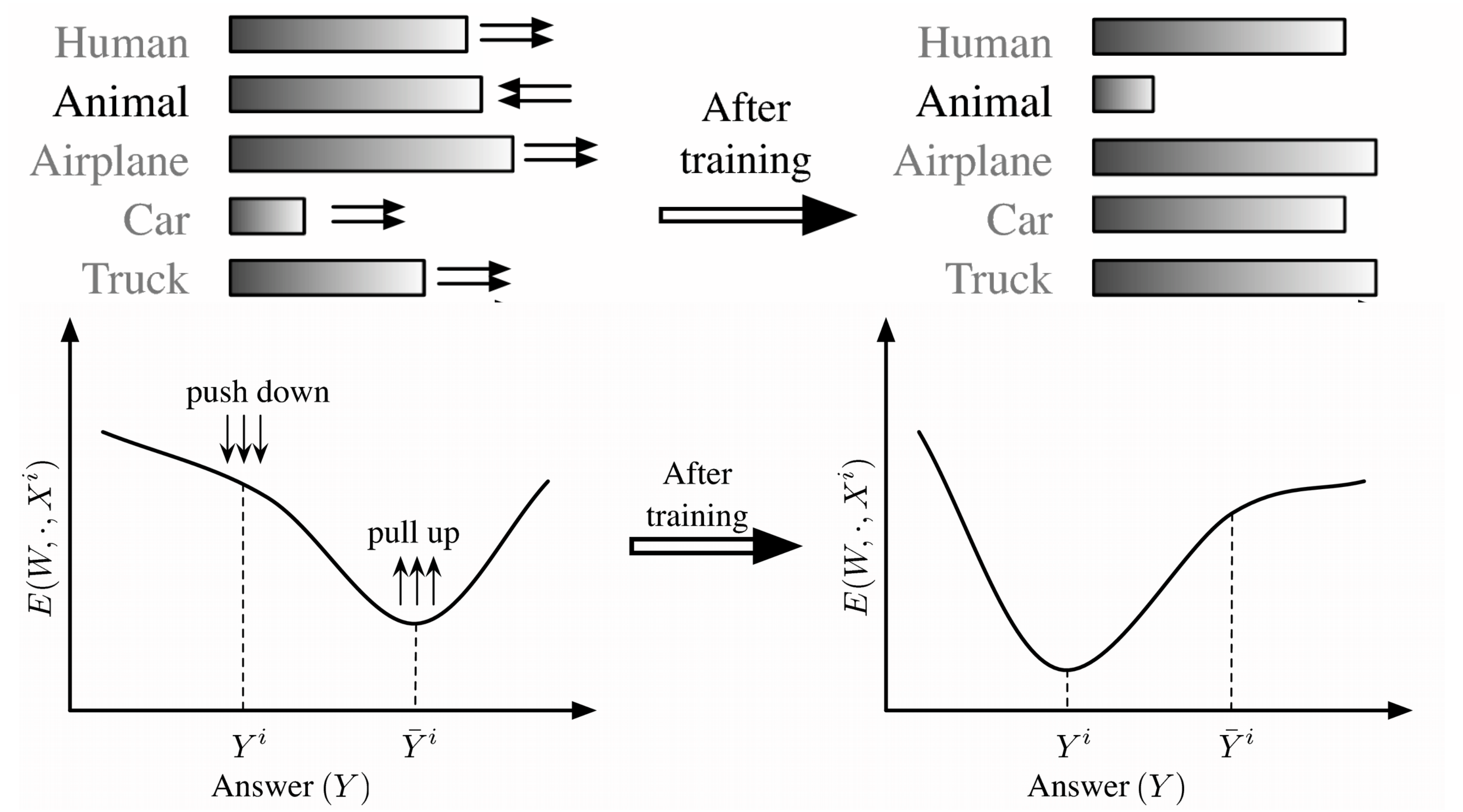
Desired answer

Energy surface for a given Xi as Y varies

Regularizer

Designing a Loss Functional

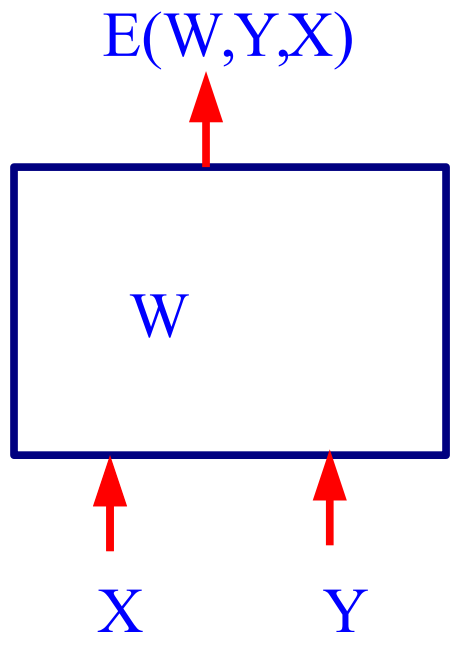
Y LeCun



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 Functional = EBM



- **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.**

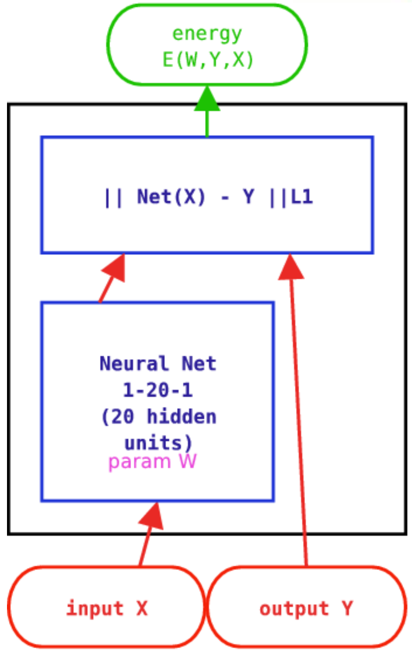
- **PROBLEM: What loss functions will make the machine approach the desired behavior?**

Examples of Loss Functions: Energy Loss

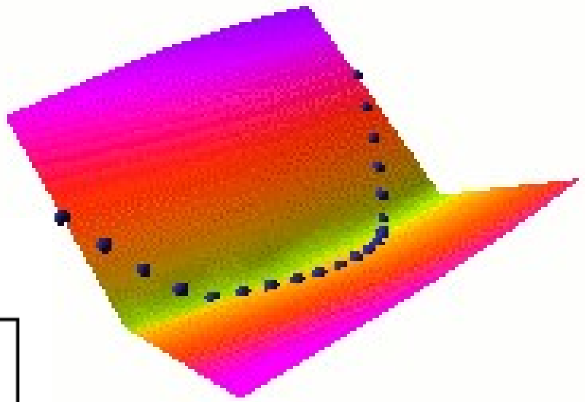
Energy Loss
$$L_{energy}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i).$$

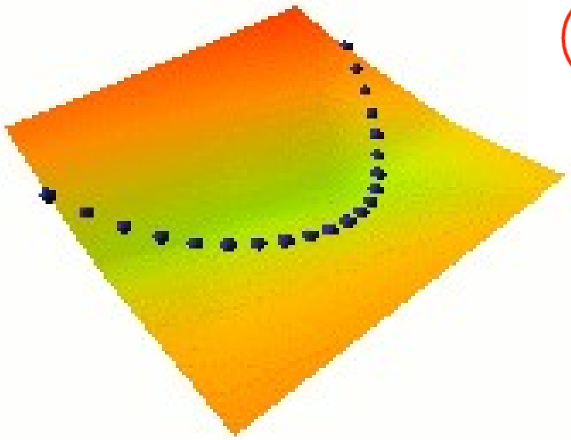
– Simply pushes down on the energy of the correct answer

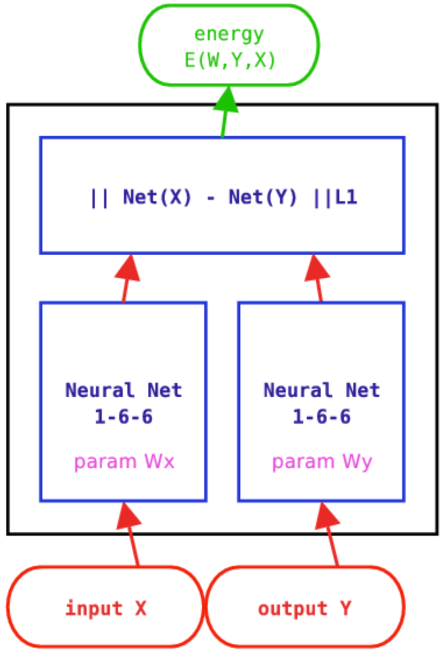
WORKS!!!



(a)







(b)

COLLAPSES!!!

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:

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

Reduces to the perceptron loss when Beta->infinity

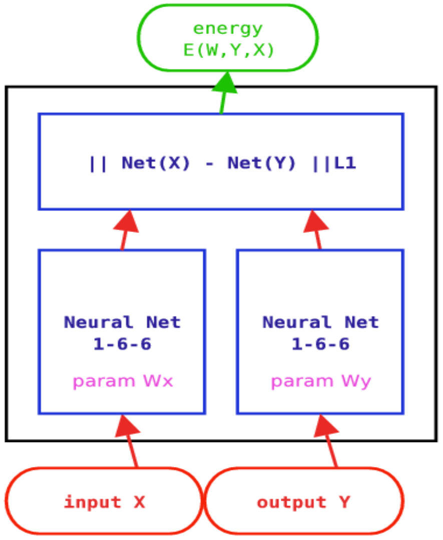
Negative Log-Likelihood Loss

Pushes down on the energy of the correct answer

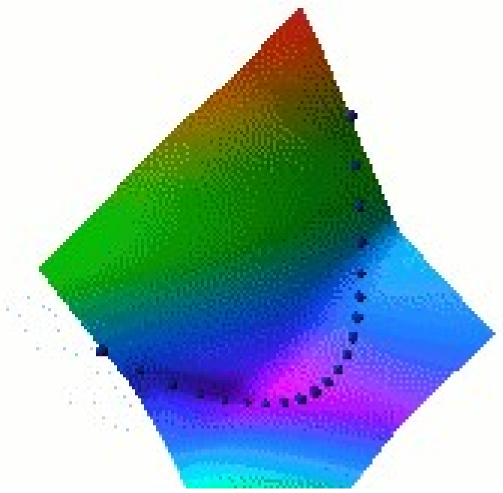
Pulls up on the energies of all answers in proportion to their probability

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

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



(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

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: $E(W, Y, X) = -Y W^T \Phi(X)$

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

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:*

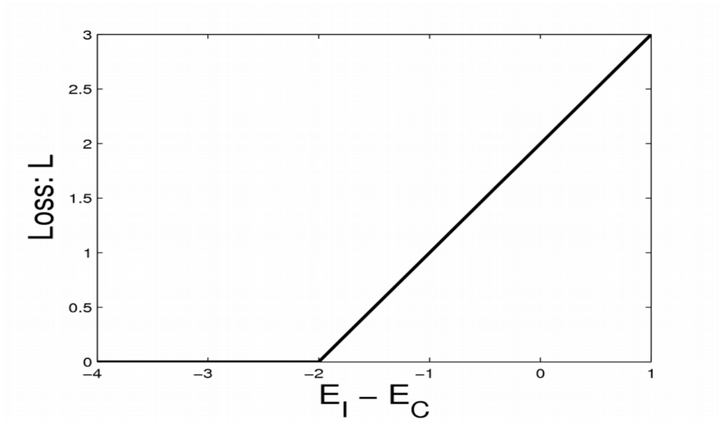
$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \tag{9}$$

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

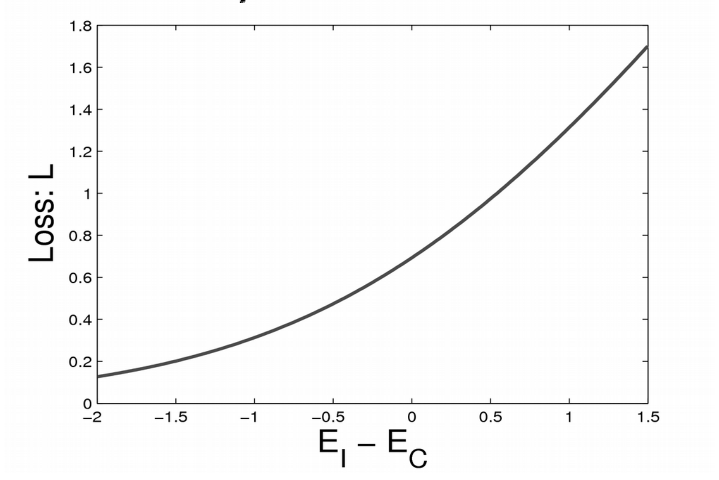
- [Altun et al. 2003], [Taskar et al. 2003]
- With the linearly-parameterized binary classifier architecture, we get linear SVMs



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

Log Loss

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

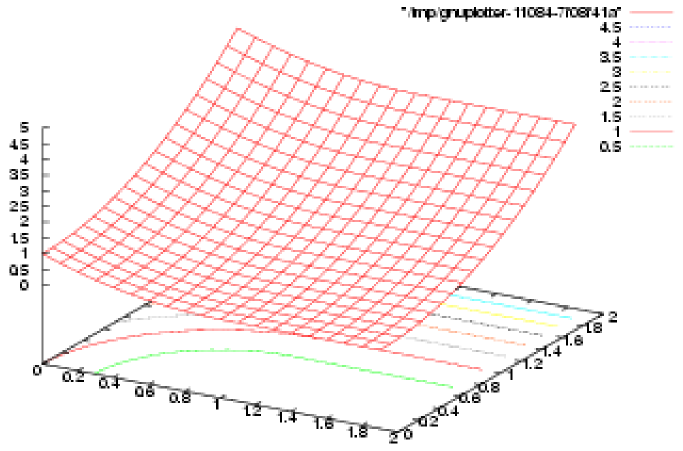


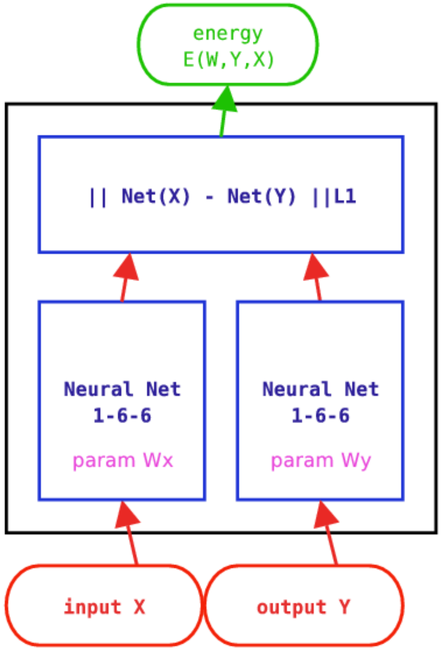
Examples of Margin Losses: Square-Square Loss

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

Square-Square Loss

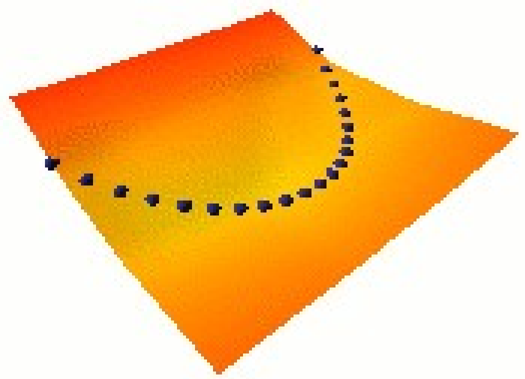
- [LeCun-Huang 2005]
- Appropriate for positive energy functions





(b)

Learning Y = X^2



NO COLLAPSE!!!

Other Margin-Like Losses

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

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

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

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

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

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

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

What Make a “Good” Loss Function

Y LeCun

Good and bad loss functions

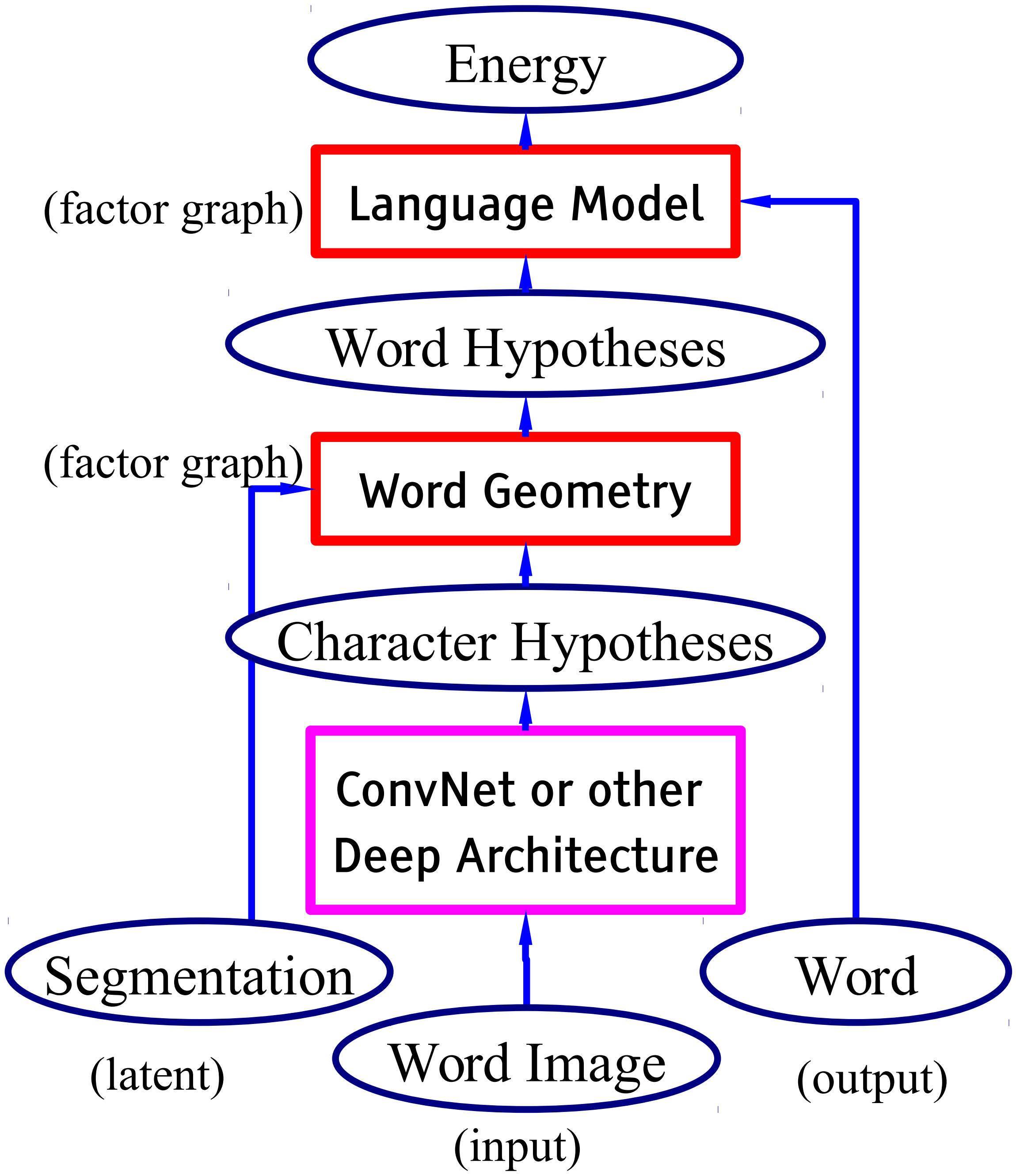
Loss (equation #)	Formula	Margin
energy loss	$E(W, Y^i, X^i)$	none
perceptron	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0
LVQ2	$\min\left(M, \max(0, E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	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

Slightly more general form:

$$L(W, X^i, Y^i) = \sum_y H\left(E(W, Y^i, X^i) - E(W, y, X^i) + C(Y^i, y)\right)$$

Recognizing Written Words With Deep Learning And Structured Prediction

End-to-End Learning – Word-Level Discriminative Training



Making every single module in the system trainable.

Every module is trained simultaneously so as to optimize a global loss function.

Includes the feature extractor, the recognizer, and the contextual post-processor (graphical model)

Problem: back-propagating gradients through the graphical model.

"Shallow" Structured Prediction

Energy function is linear in the parameters

$$E(X,Y,Z)=\sum_i W_i^T h_i(X,Y,Z)$$

with the NLL Loss :

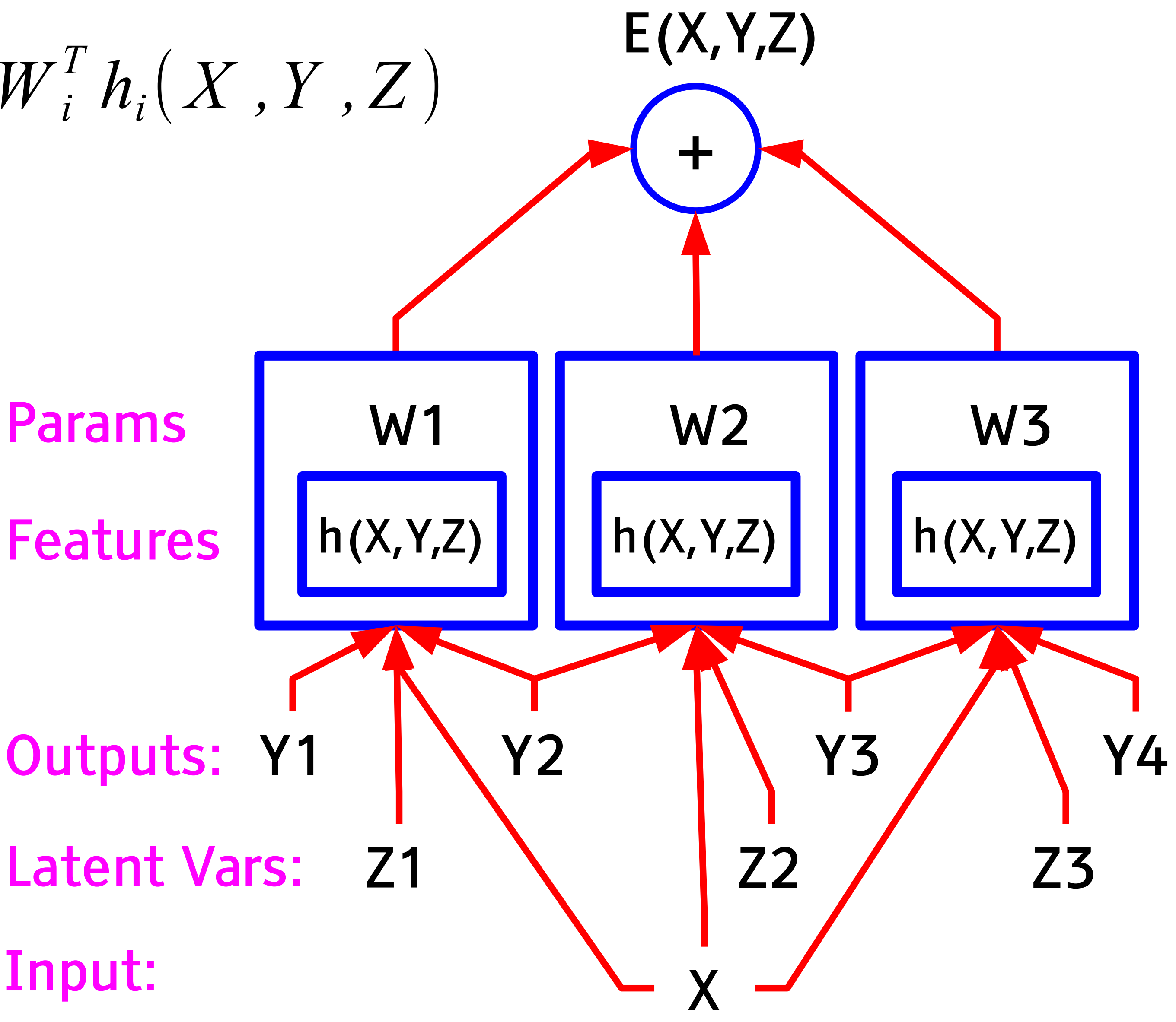
- **Conditional Random Field** [Lafferty, McCallum, Pereira 2001]

with Hinge Loss:

- **Max Margin Markov Nets** and **Latent SVM** [Taskar, Altun, Hofmann...]

with Perceptron Loss

- **Structured Perceptron** [Collins...]



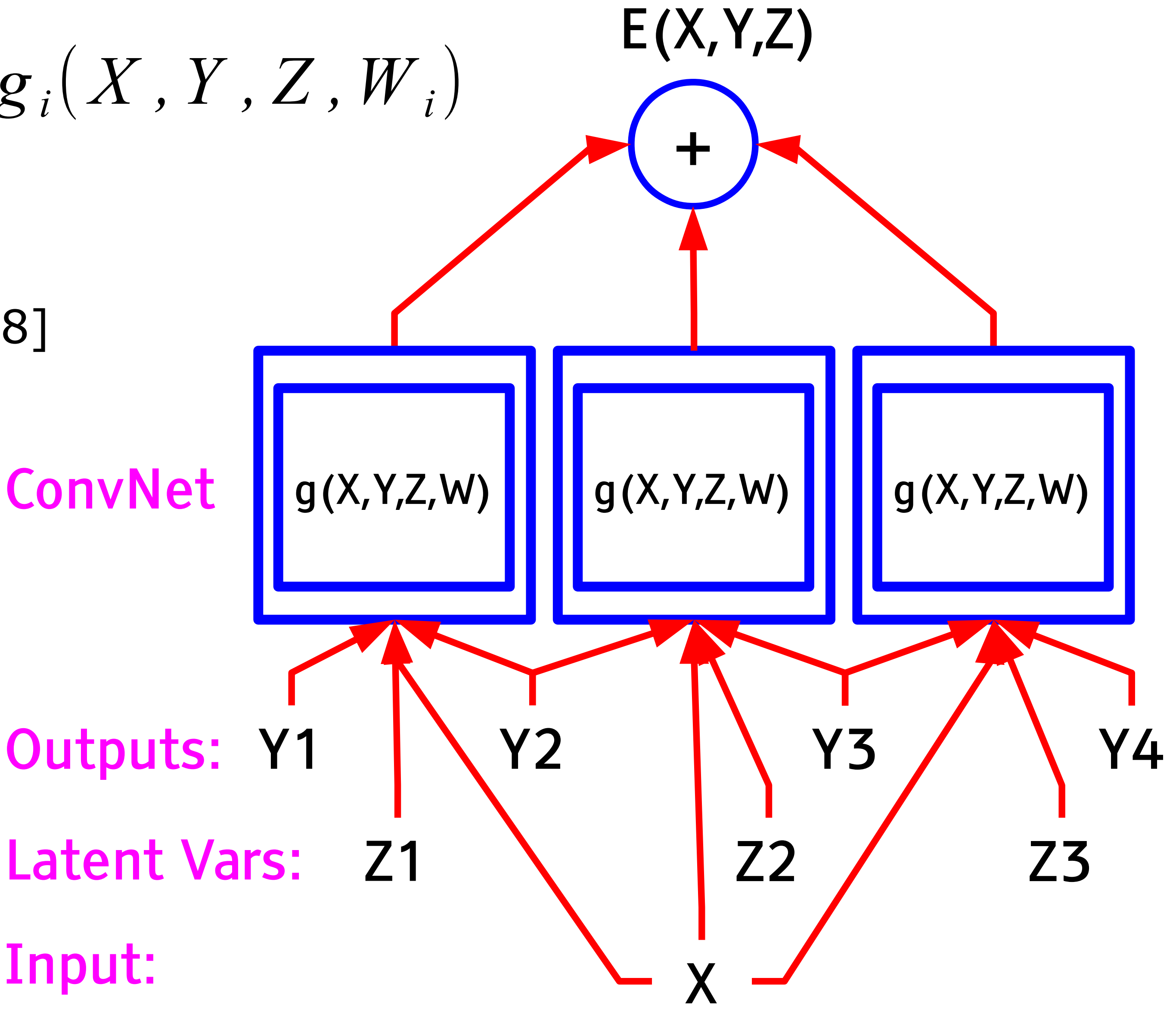
Deep Structured Prediction

Energy function is linear in the parameters

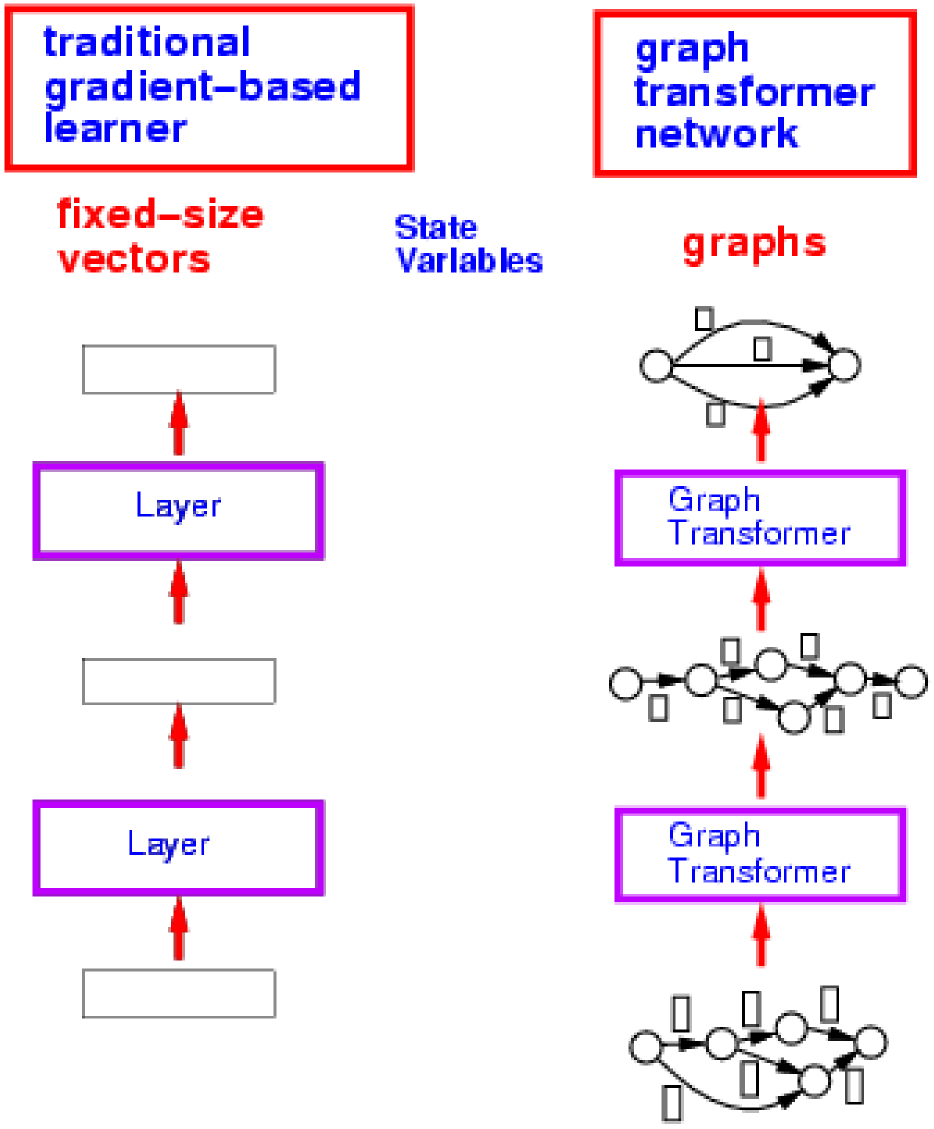
$$E(X,Y,Z)=\sum_i g_i(X,Y,Z,W_i)$$

Graph Transformer Networks

- [LeCun, Bottou, Bengio,Haffner 97,98]
- NLL loss
- Perceptron loss



Using Graphs instead of Vectors or Arrays.



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

Deep Factors & implicit graphs: GTN

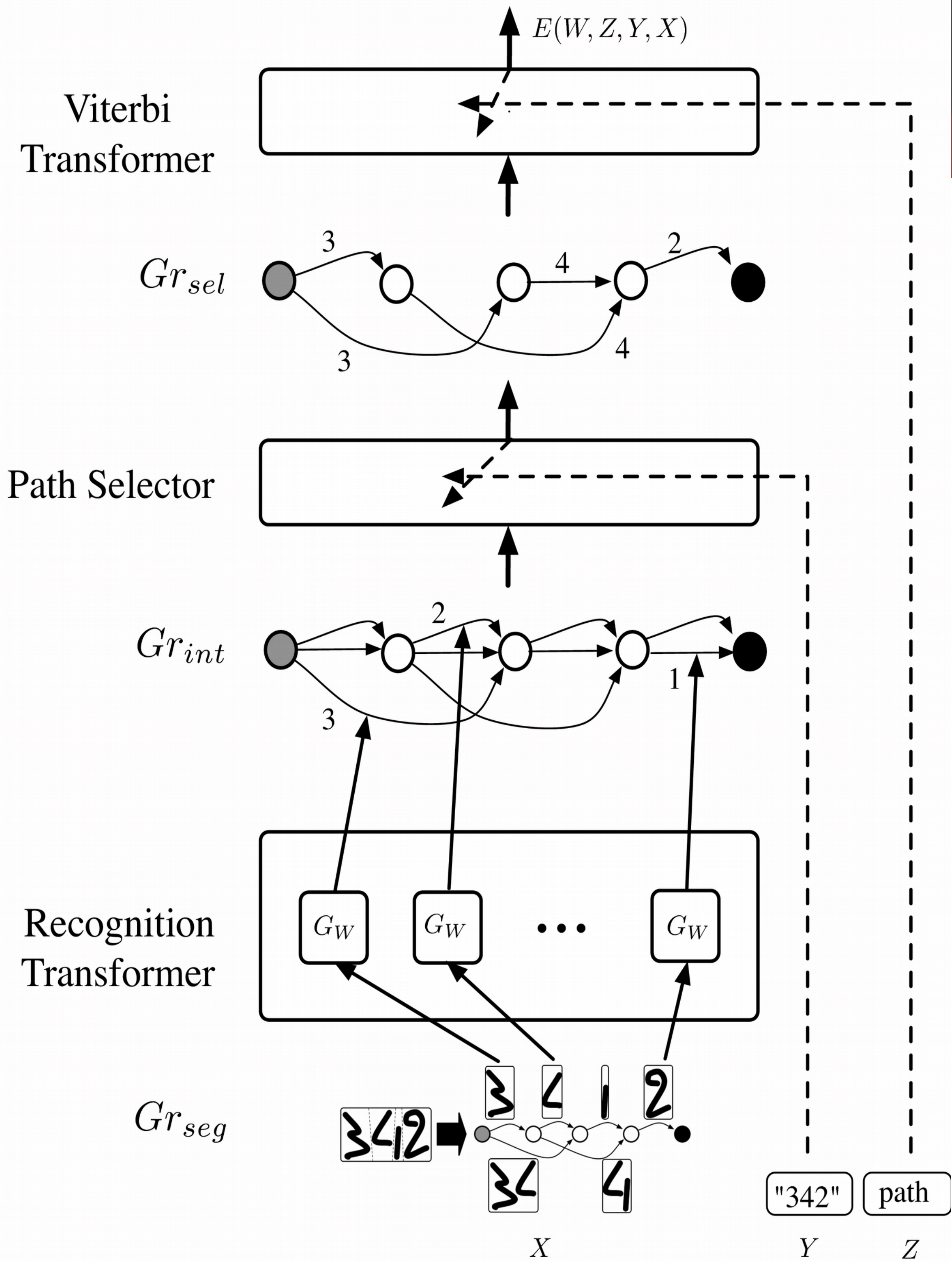
Handwriting Recognition with Graph Transformer Networks

Un-normalized hierarchical HMMs

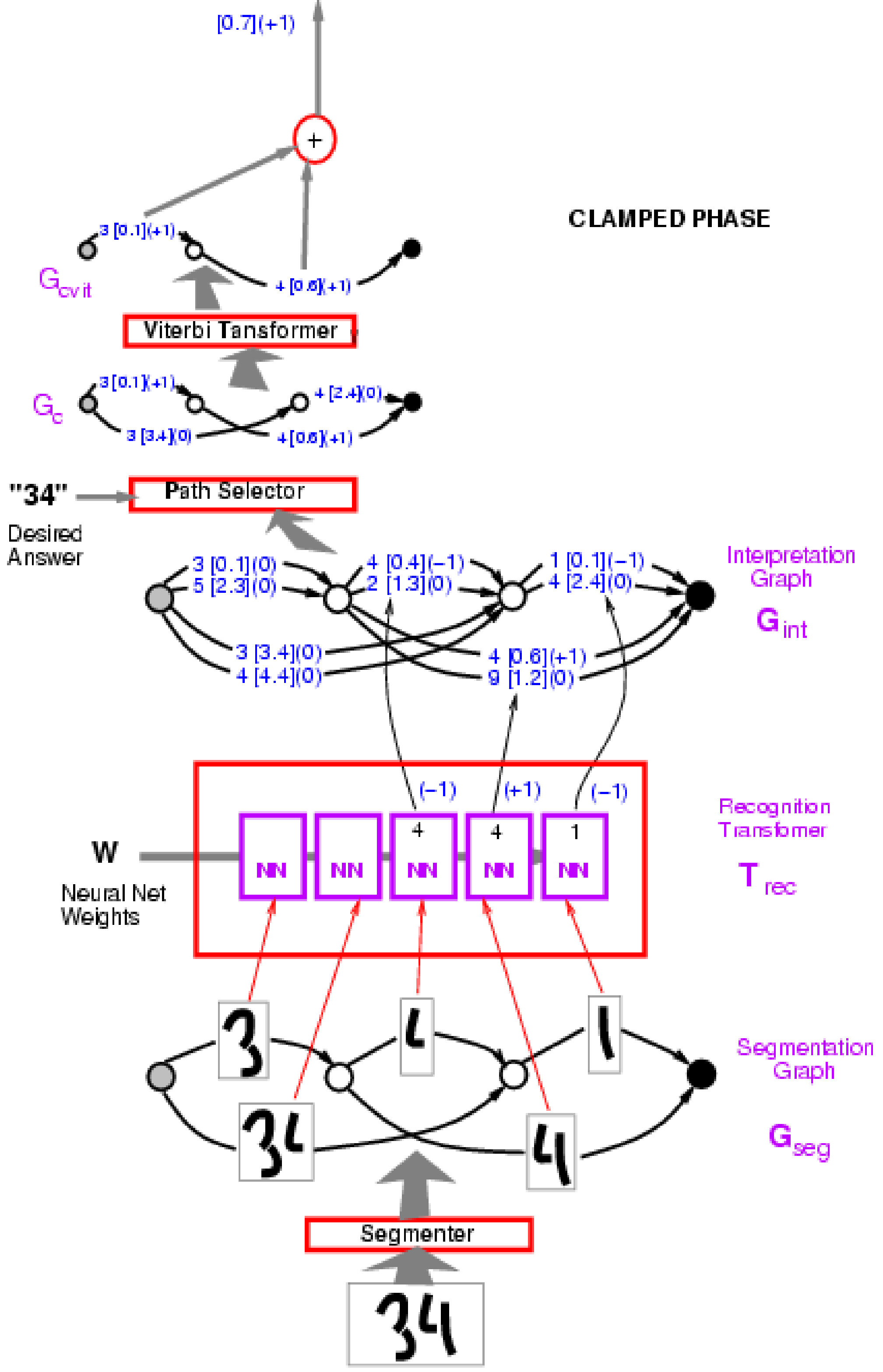
- Trained with Perceptron loss [LeCun, Bottou, Bengio, Haffner 1998]
- Trained with NLL loss [Bengio, LeCun 1994], [LeCun, Bottou, Bengio, Haffner 1998]

Answer = sequence of symbols

Latent variable = segmentation



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:

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

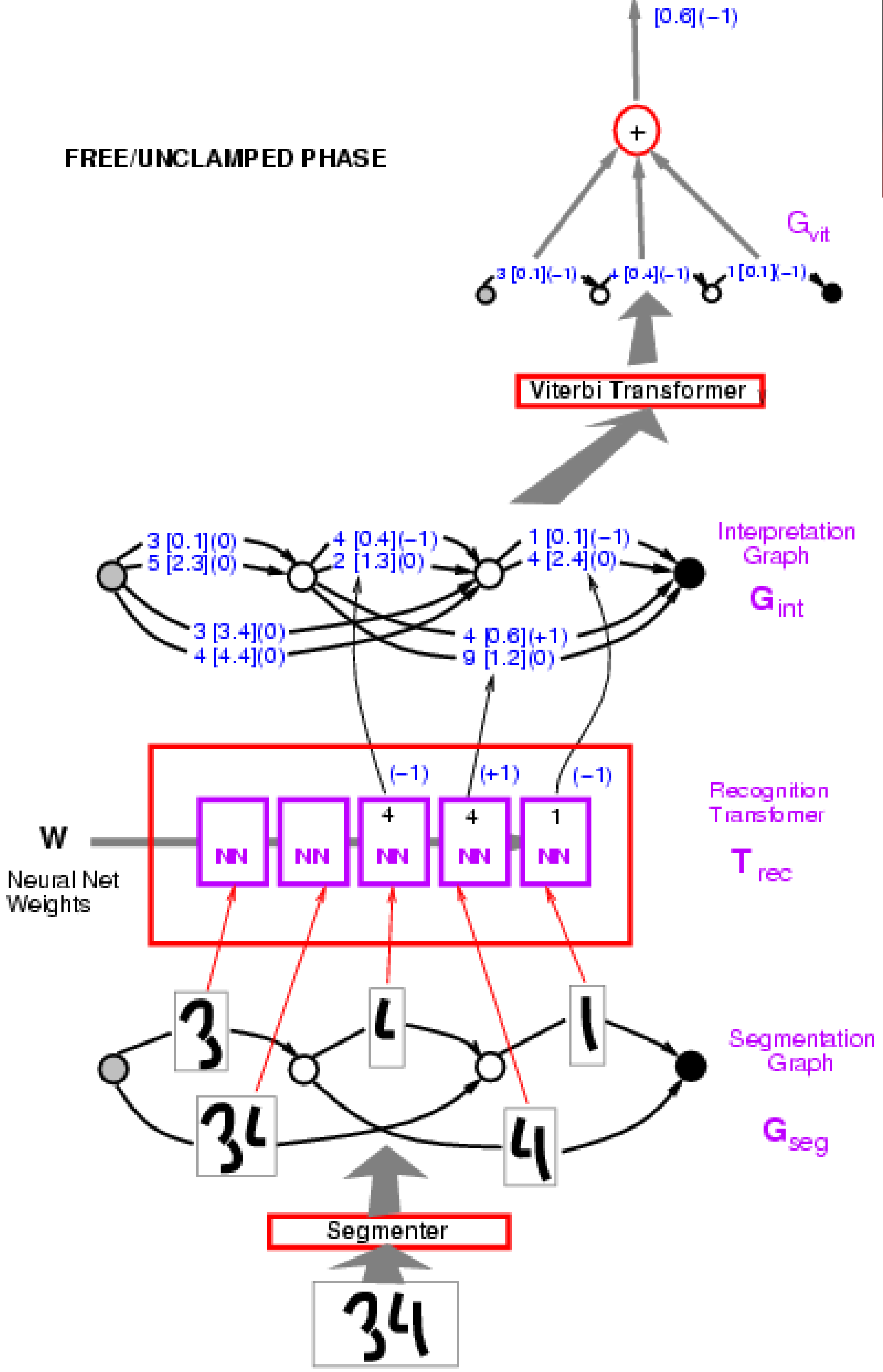
Graph Transformer Networks

Variables:

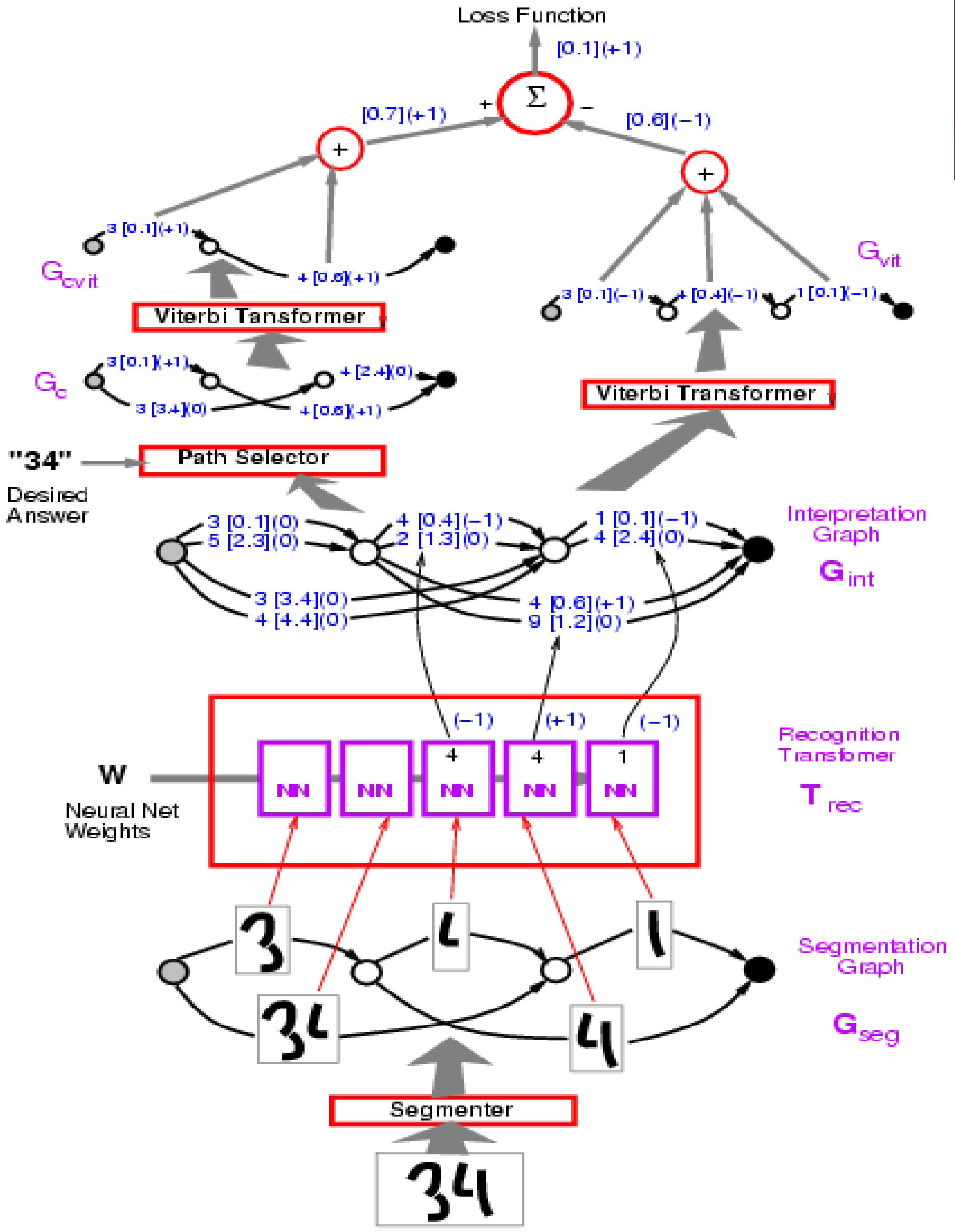
- X: input image
- Z: path in the interpretation graph/segmentation
- Y: sequence of labels on a path

Loss function: computing the constrastive term:

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



Graph Transformer Networks



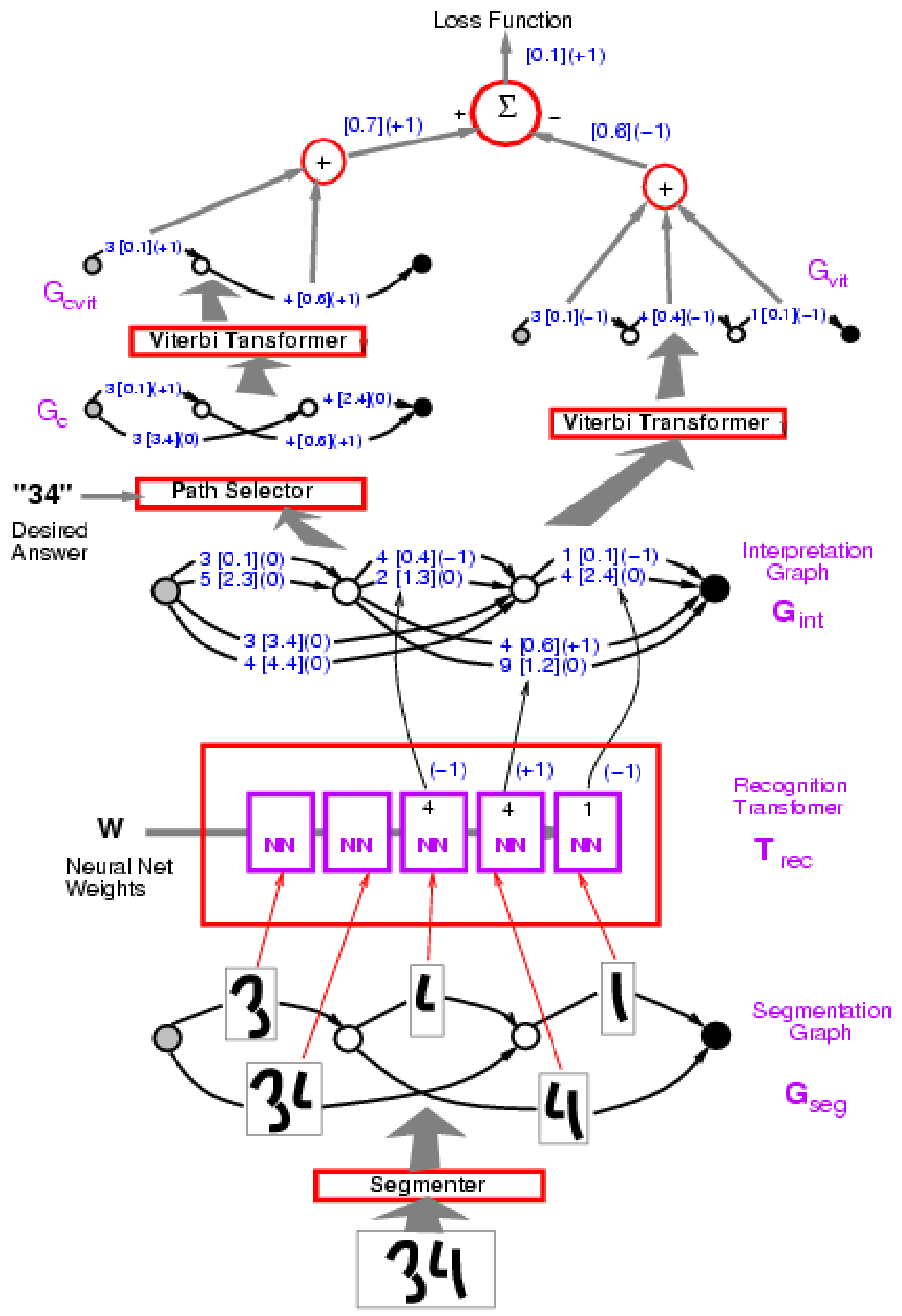
Example: Perceptron loss

Loss = Energy of desired answer – Energy of best answer.

– (no margin)

Integrating Deep Learning and Structured Prediction

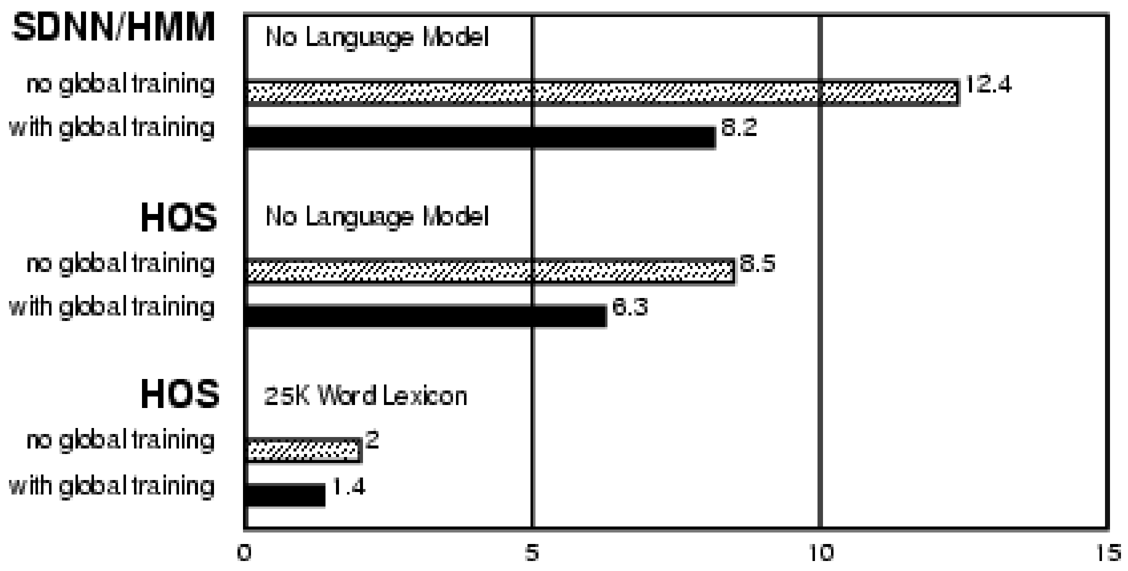
- **Structured prediction: when the output is structured: string, graph.....**
- **Integrating deep learning and structured prediction is an old idea**
  - In fact, it predates structured prediction [LeCun, Bottou, Bengio, Haffner 1998]
- **Globally-trained convolutional-net + graphical models for handwriting recognition**
  - trained discriminatively at the word level
  - Loss identical to CRF and structured perceptron
  - Compositional movable parts model

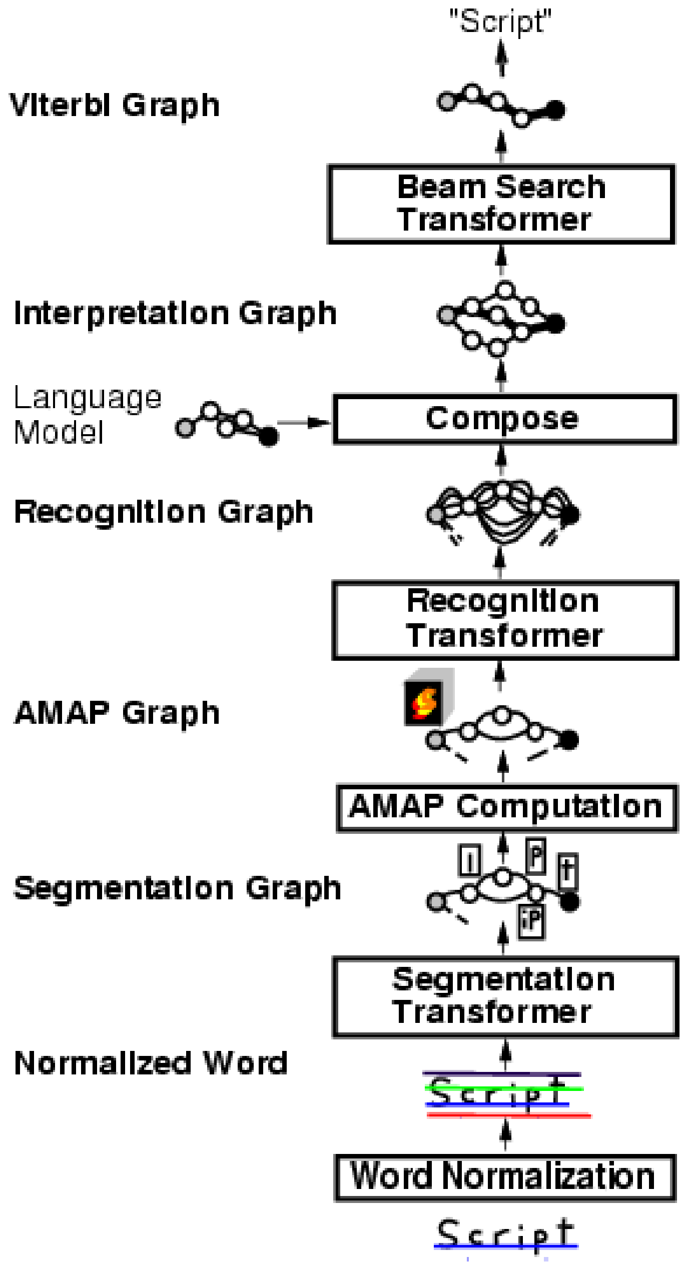


Global (word-level) Training Helps

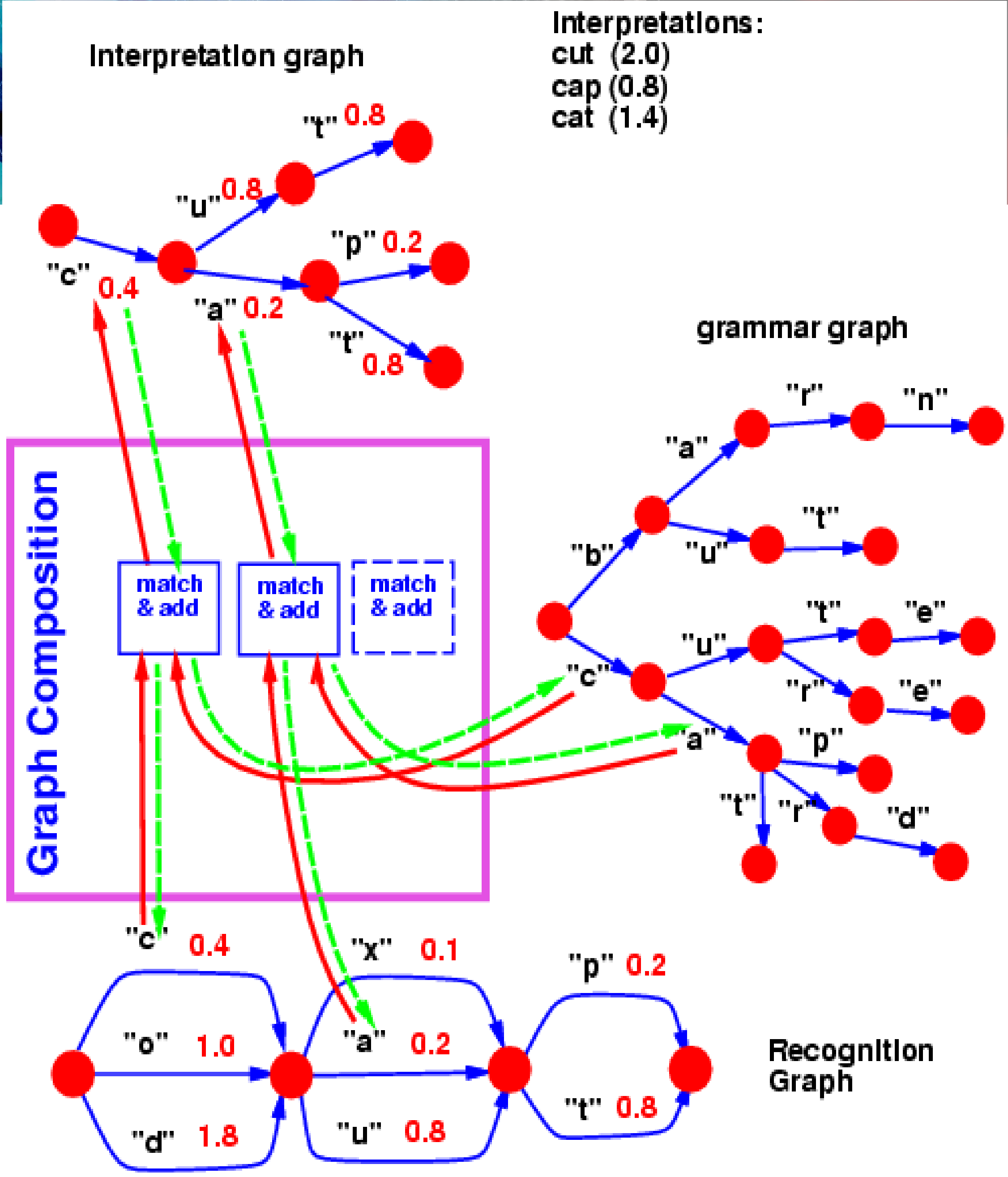
Pen-based handwriting recognition (for tablet computer)

- [Bengio&LeCun 1995]





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

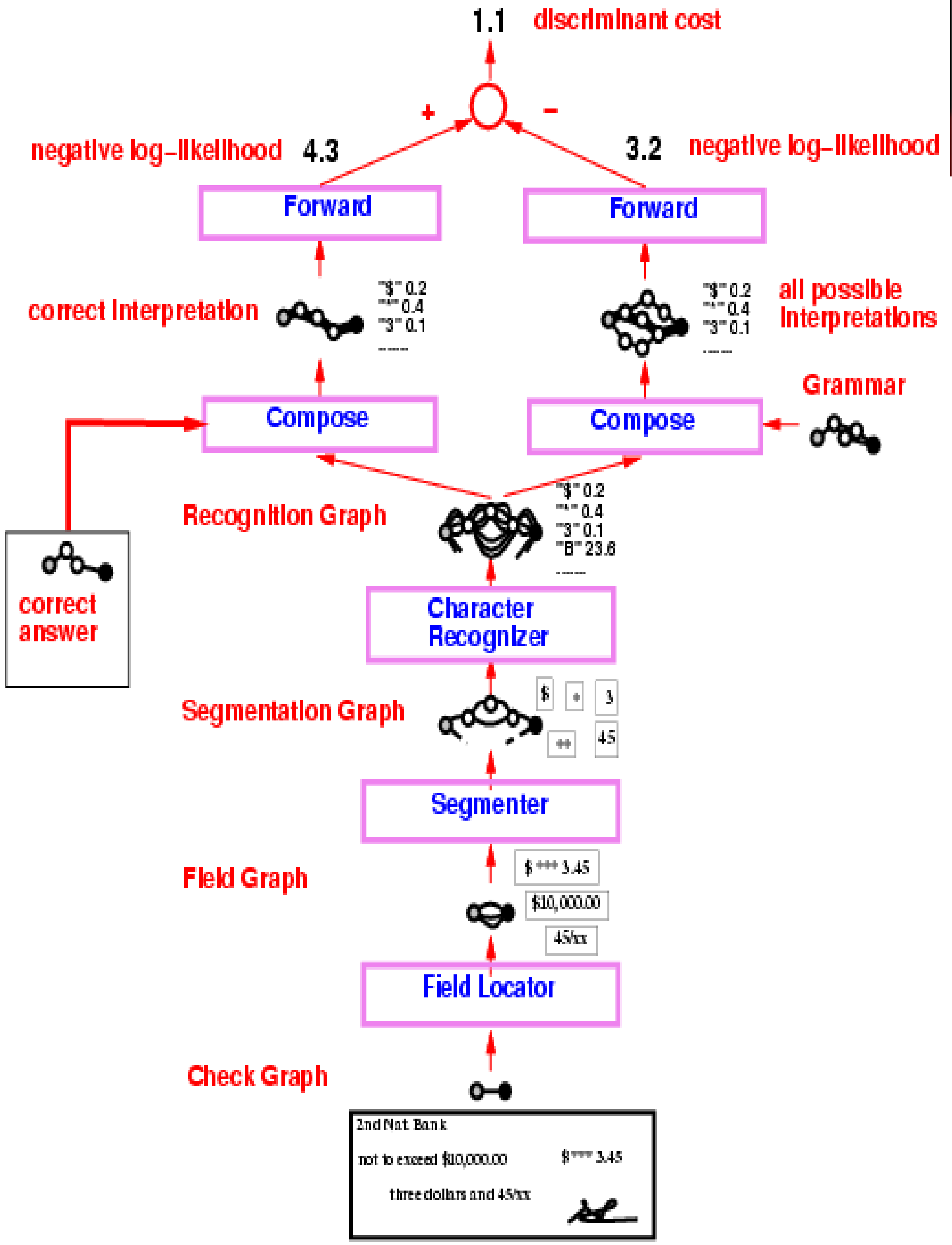
Graph transformer network trained to read check amounts.

Trained globally with Negative-Log-Likelihood loss.

50% percent correct, 49% reject, 1% error (detectable later in the process.

Fielded in 1996, used in many banks in the US and Europe.

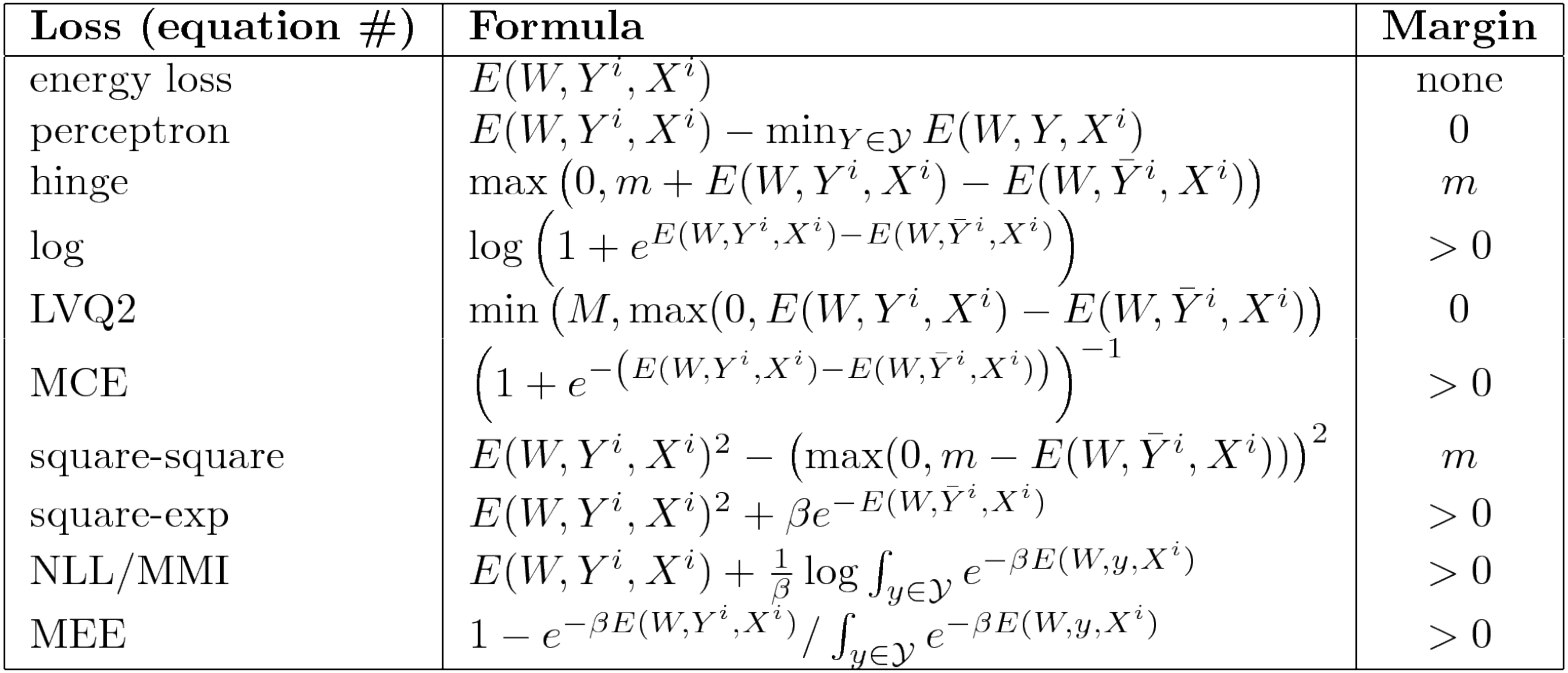
Processes an estimated 10% to 20% of all the checks written in the US.



Loss Function to train Energy-Based Models

Good and bad loss functions

A tutorial on Energy-Based Learning [LeCun et al 2006]



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

Deep Structured Predictions 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

Word-level global discriminative training with GMM:

With Minimum Empirical Error loss

- Ljolje and Rabiner (1990)

With MCE

- Juang et al. (1997)

Word-level global discriminative training with ConvNets:

with the LVQ2 Loss :

- Driancourt and Bottou's speech recognizer (1991)

with Neg Log Likelihood (aka MMI):

- Bengio (1992), Haffner (1993), Bourlard (1994)

CRF-like Late normalization

- un-normalized HMM
- Bottou pointed out the **label bias problem** (1991)
- Denker and Burges proposed a solution (1995)
- Implemented in (LeCun et al 1998)