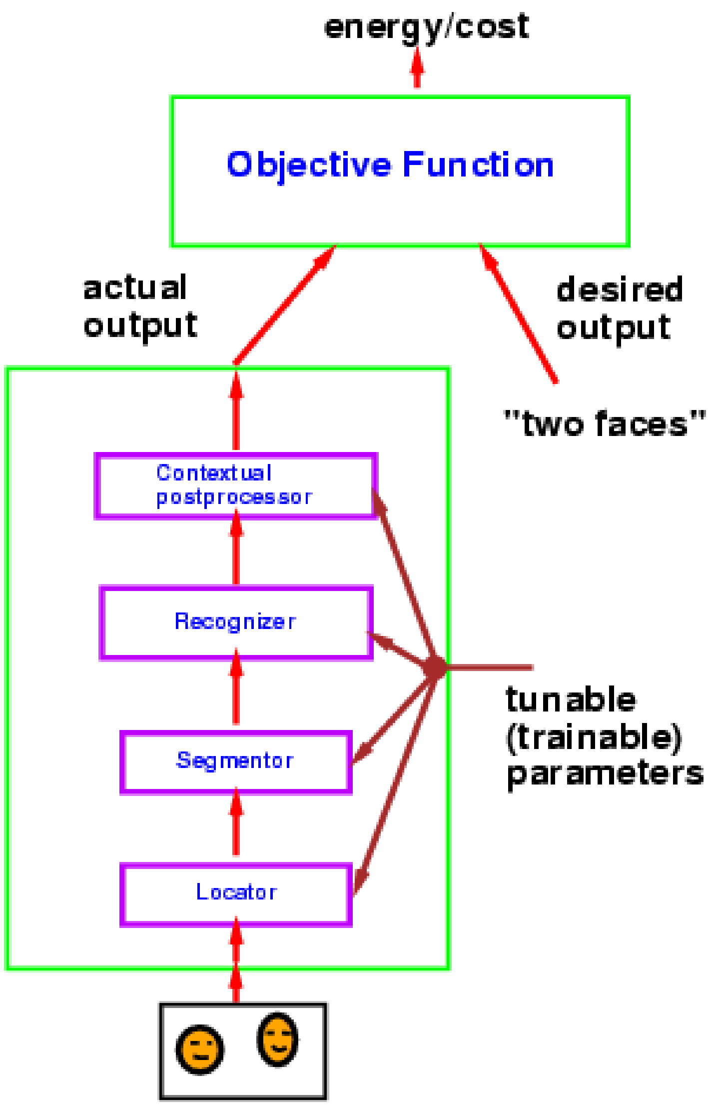
Deep Learning, Graphical Models, Energy-Based Models, Structured Prediction

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

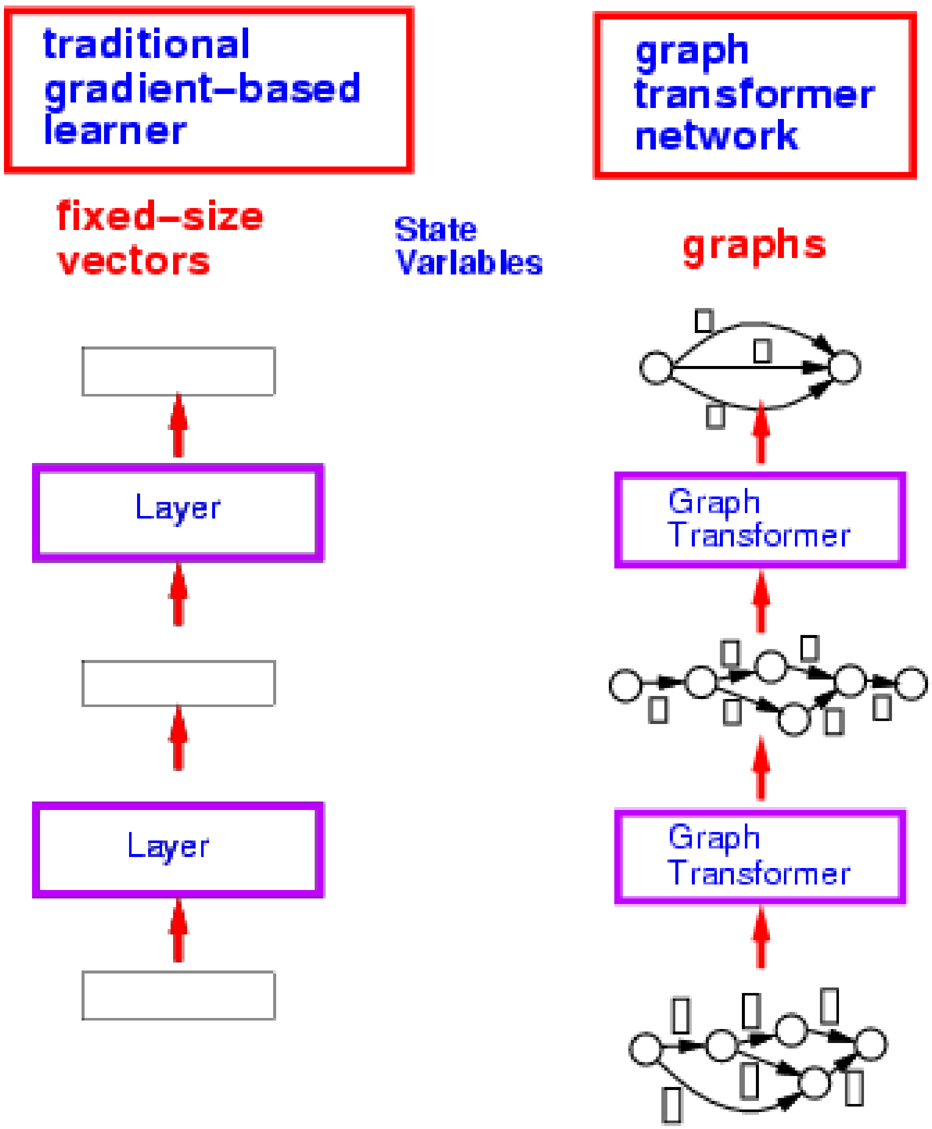
http://yann.lecun.com
http://www.cs.nyu.edu/~yann

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.

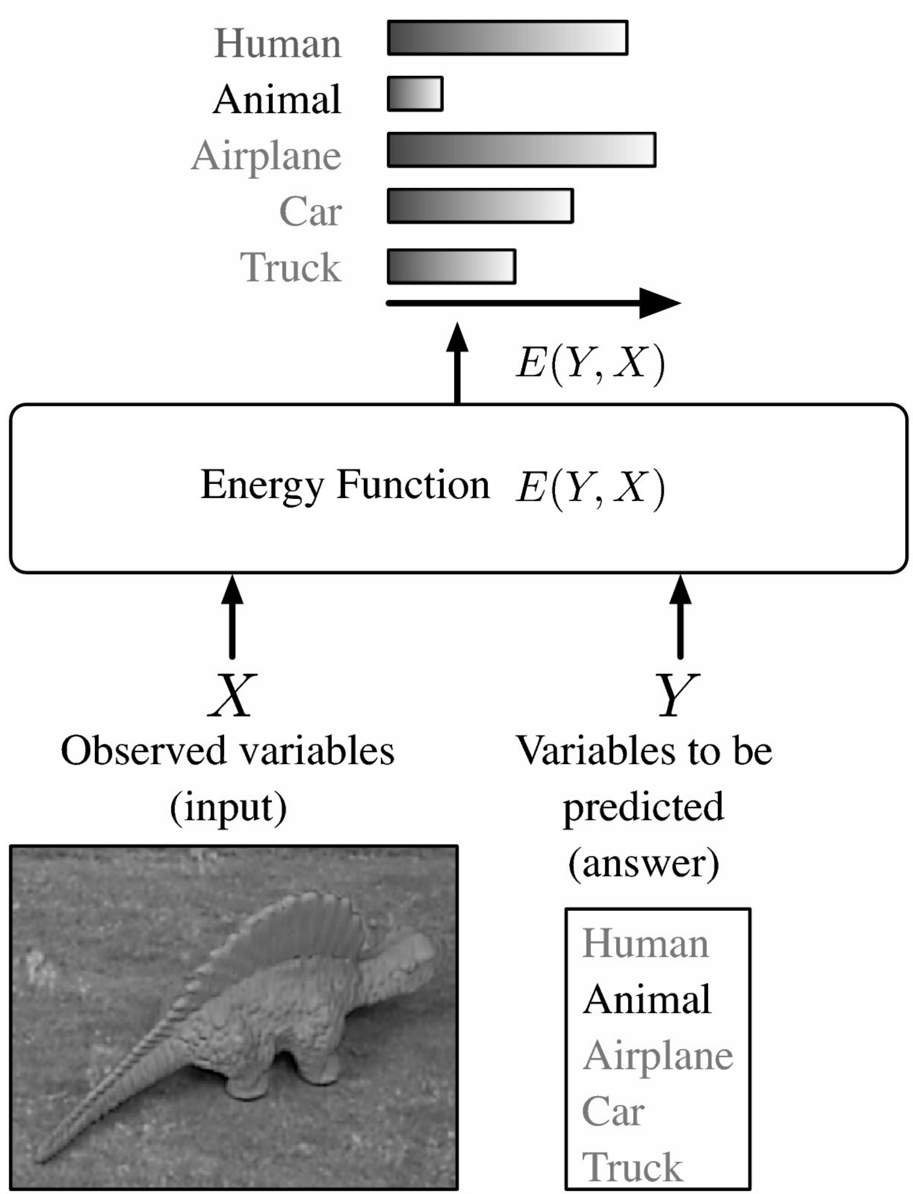


- Whereas traditional learning machines manipulate fixed-size vectors, Graph Transformer Networks manipulate graphs.

Energy-Based Model

- **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 Model for Decision-Making

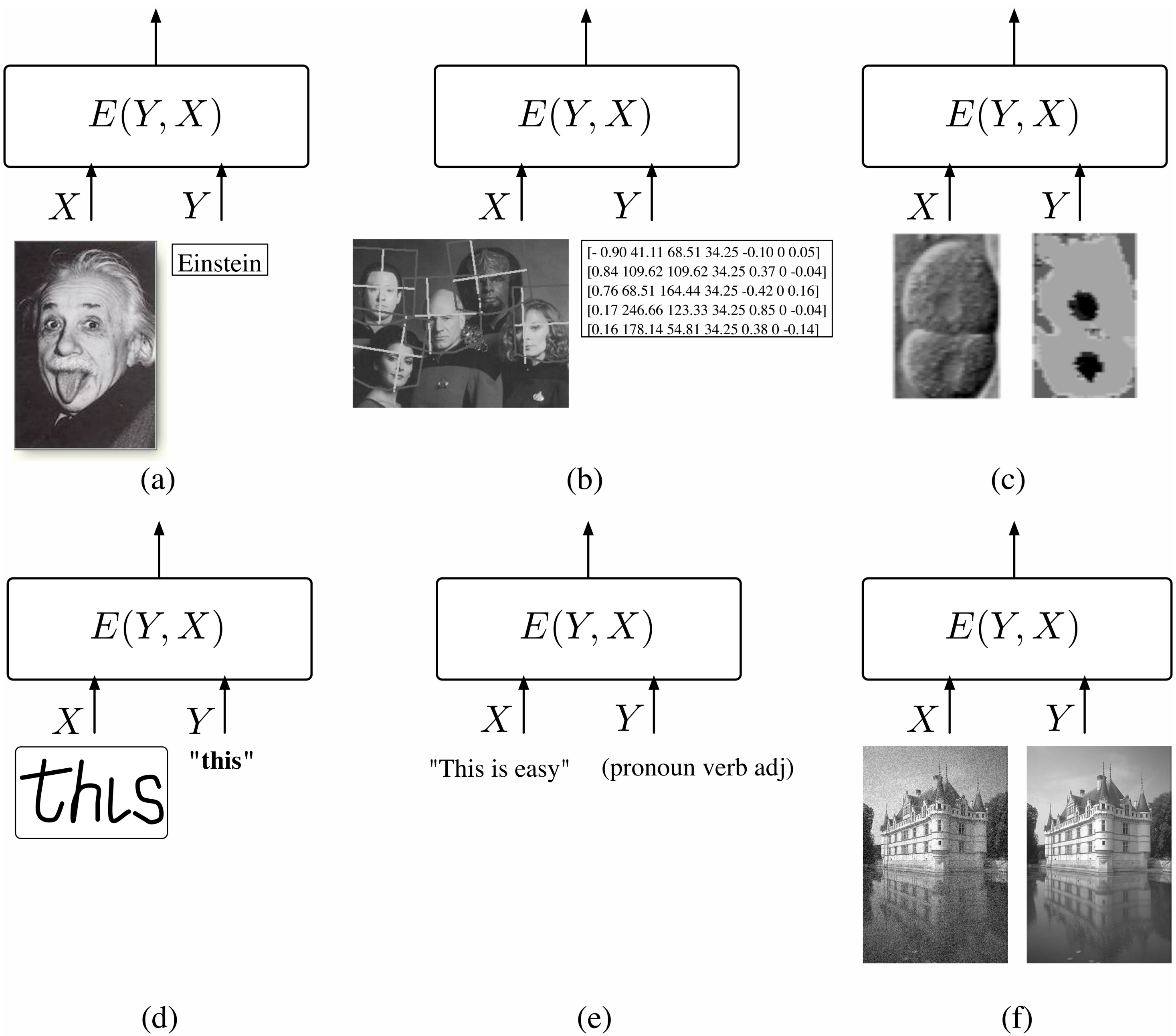


- **Model: Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).**

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

- **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



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

Converting Energies to Probabilities

Energies are uncalibrated

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

How do we calibrate energies?

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

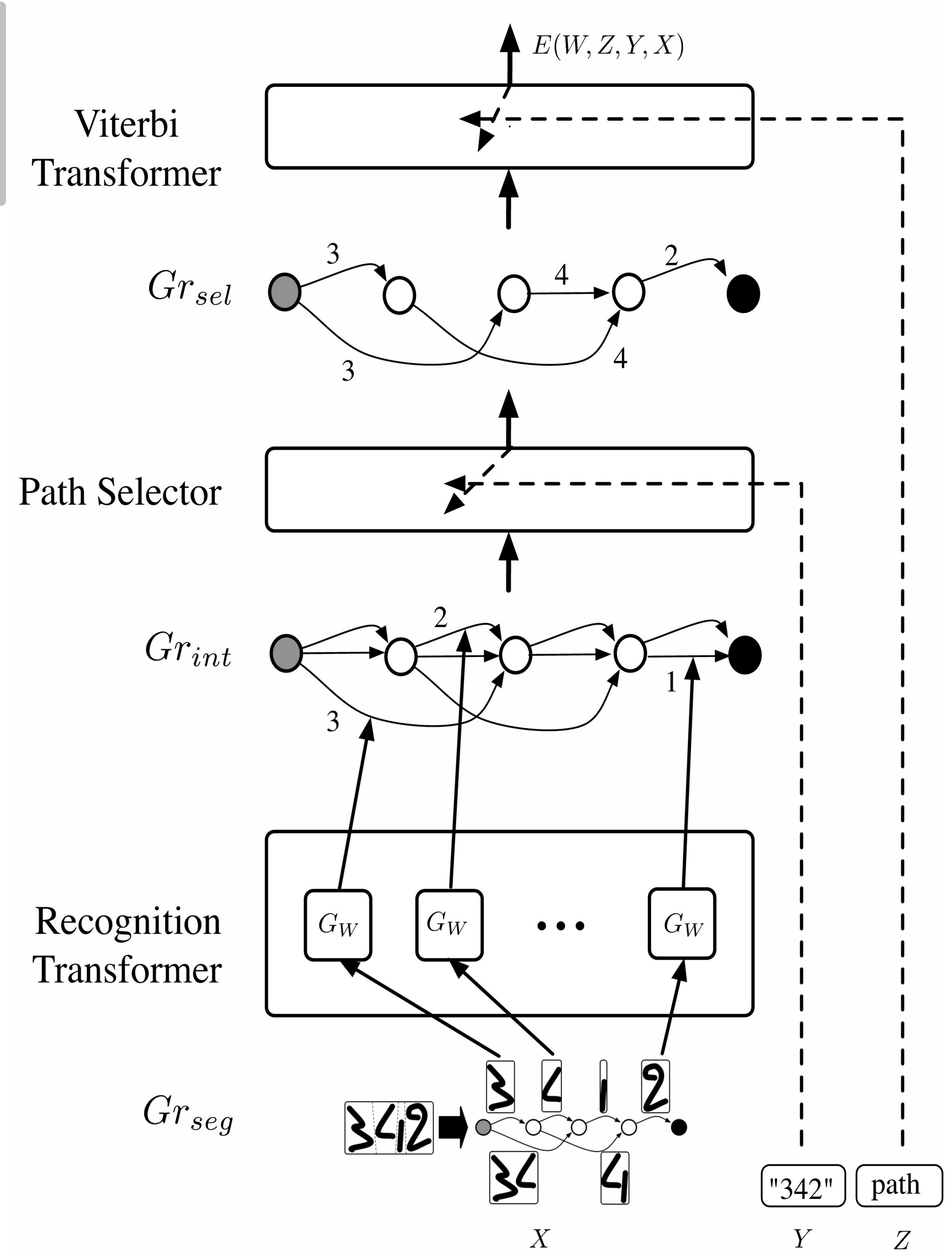
$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}},$$

Partition function

Inverse temperature

Handwriting recognition
Sequence labeling

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

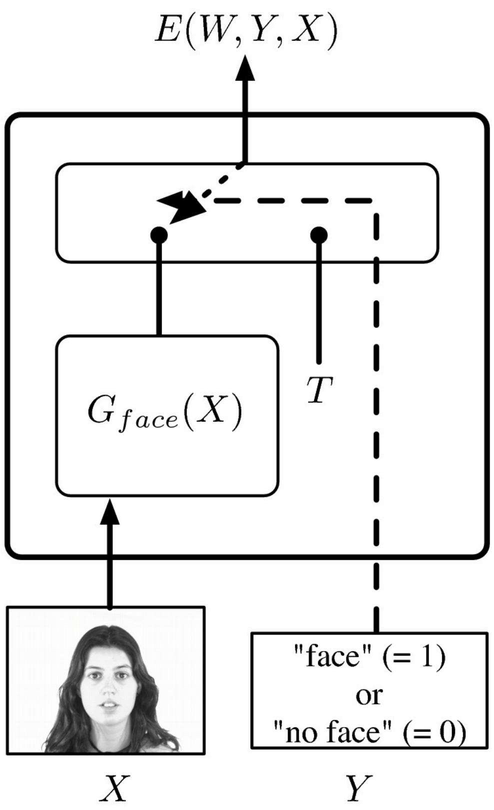


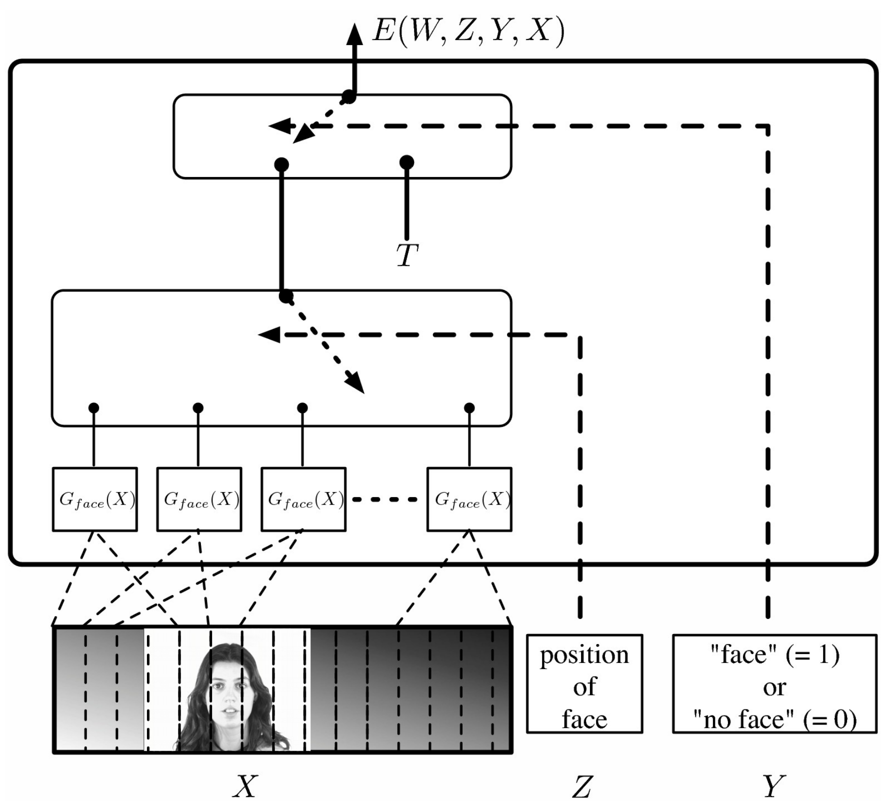
Latent Variable Models

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

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

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





What can the latent variables represent?

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

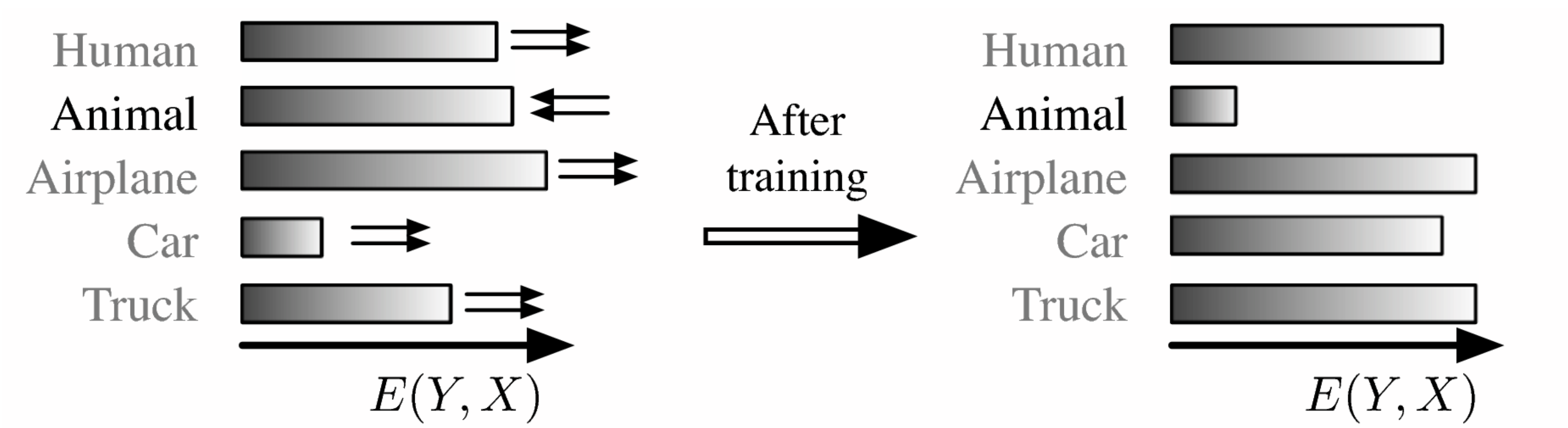
$$Y^* = \text{argmin}_{Y\in\mathcal{Y}} - \frac{1}{\beta}\log\int_{z\in\mathcal{Z}} e^{-\beta E(z,Y,X)}.$$

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

$$\mathcal{L}(E, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \underbrace{L}_{\text{Per-sample loss}}(\underbrace{Y^i}_{\text{Desired answer}}, \underbrace{E(W, \mathcal{Y}, X^i)}_{\text{Energy surface for a given Xi as Y varies}}) + \underbrace{R(W)}_{\text{Regularizer}}.$$

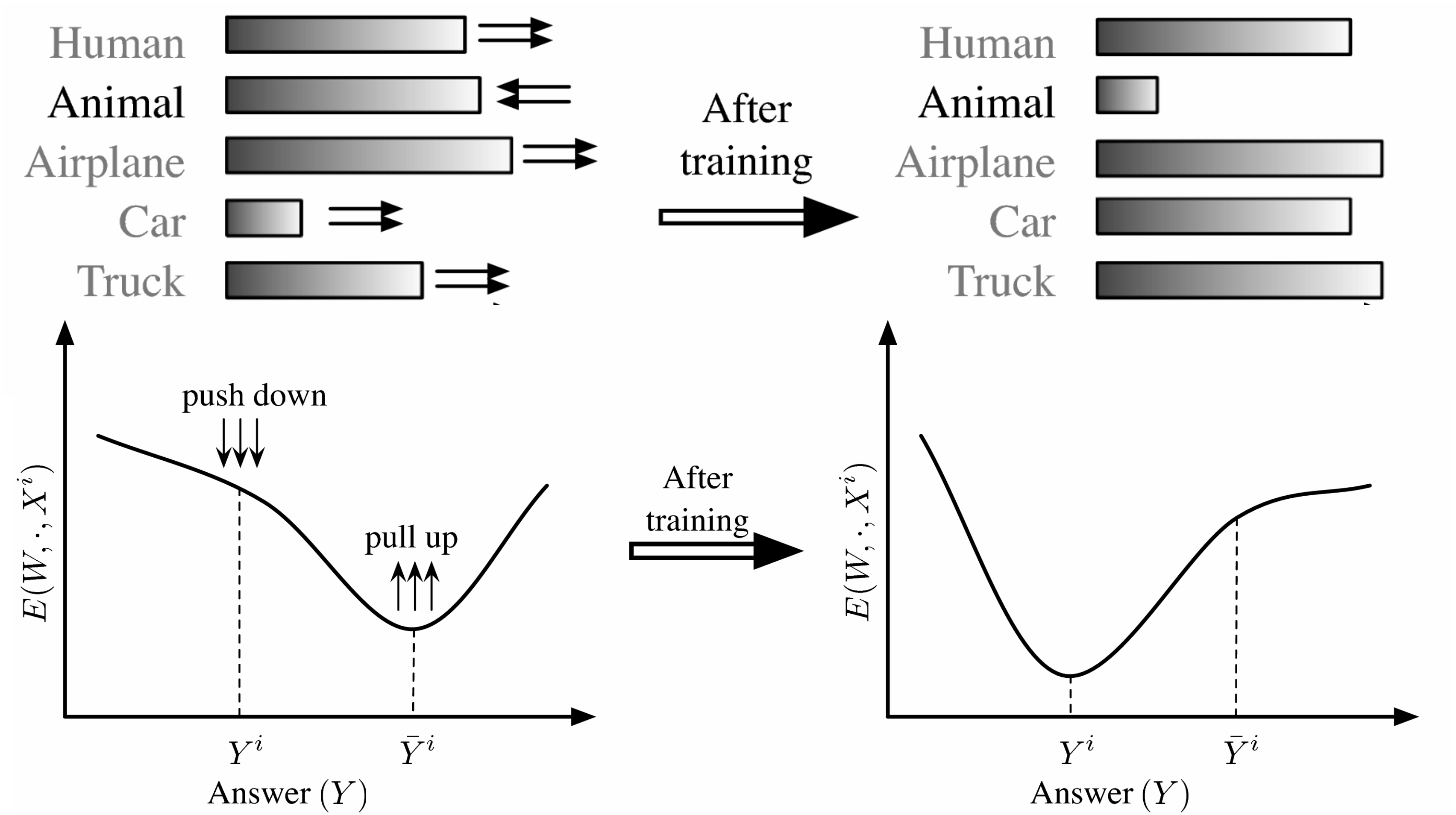
Per-sample loss

Desired answer

Energy surface for a given Xi as Y varies

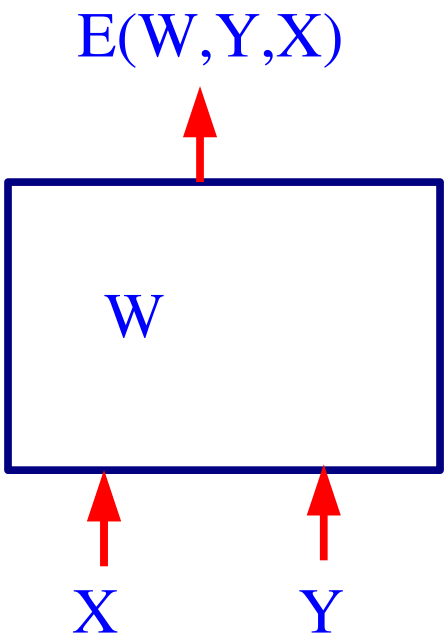
Regularizer

Designing a Loss Functional



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

Architecture + Inference Algo + Loss Function = Model

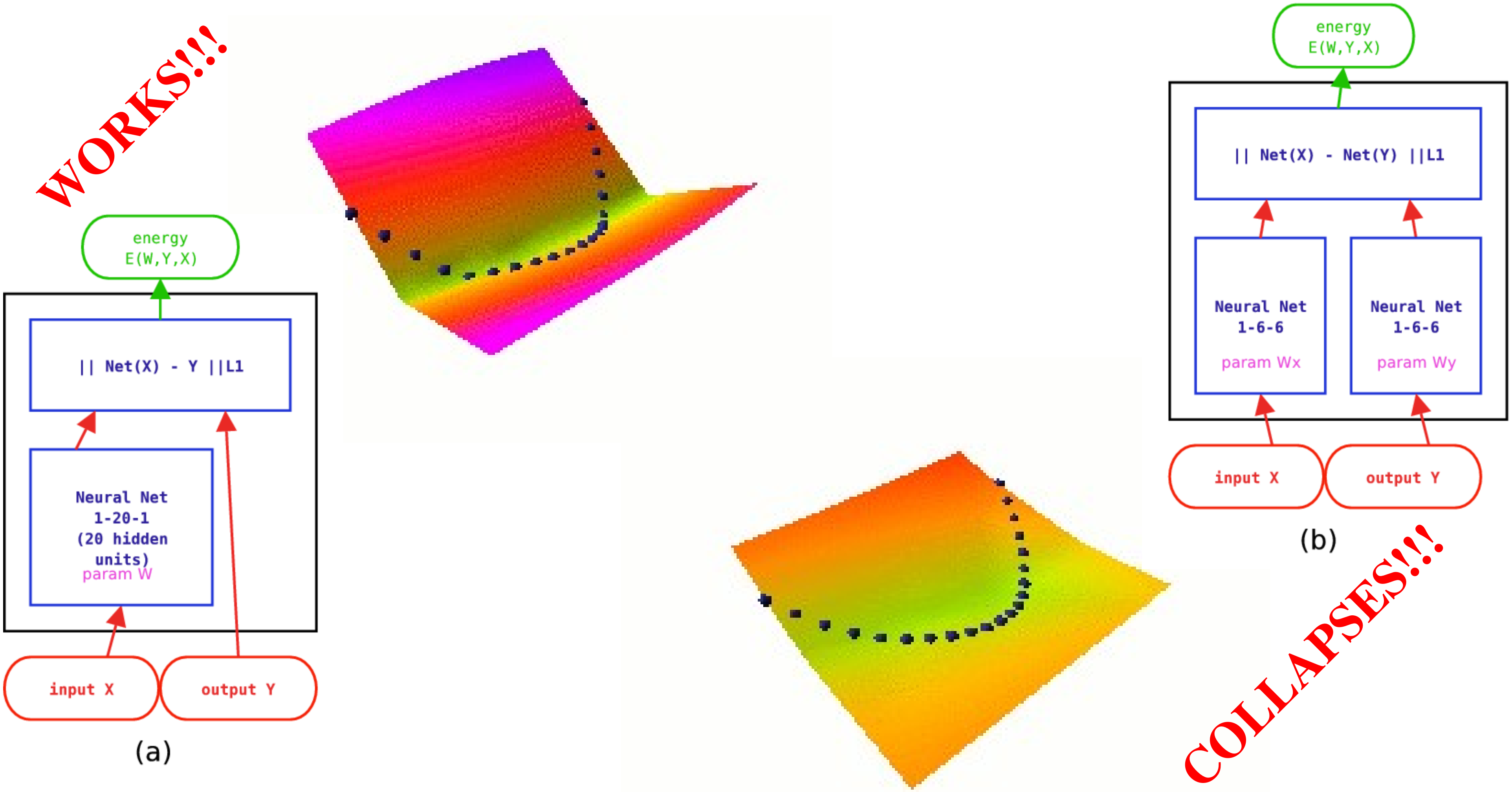


- **1. Design an architecture:** a particular form for E(W,Y,X).
- **2. Pick an inference algorithm for Y:** MAP or conditional distribution, belief prop, min cut, variational methods, gradient descent, MCMC, HMC.....
- **3. Pick a loss function:** in such a way that minimizing it with respect to W over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

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

Examples of Loss Functions: Energy Loss

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



Negative Log-Likelihood Loss

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

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

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

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

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

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

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

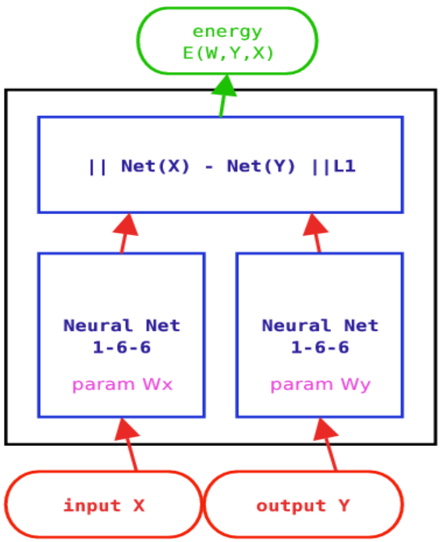
- **Reduces to the perceptron loss when Beta->infinity**

Negative Log-Likelihood Loss

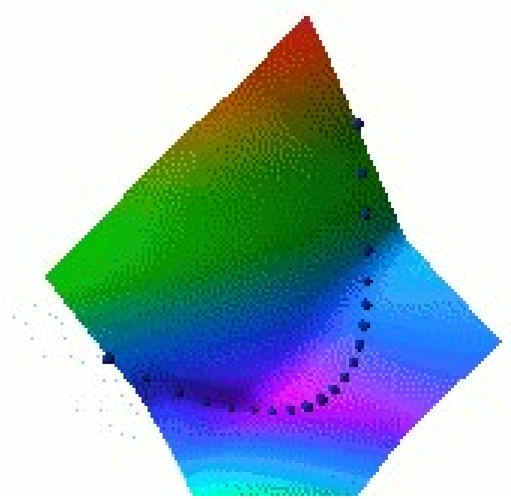
- **Pushes down on the energy of the correct answer**
- **Pulls up on the energies of all answers in proportion to their probability**

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

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



(b)



Negative Log-Likelihood Loss

- **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:*

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

- 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:*

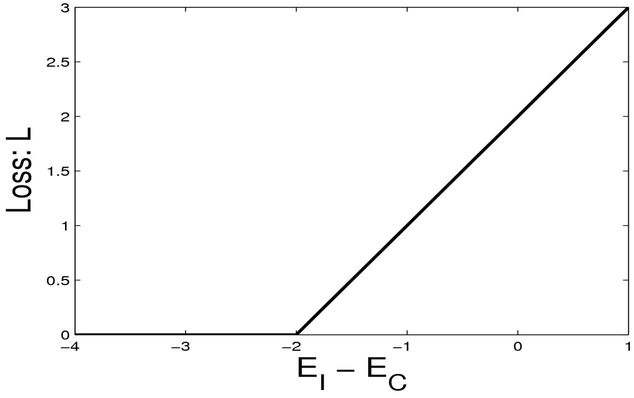
$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \quad (9)$$

Examples of Generalized Margin Losses

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

Hinge Loss

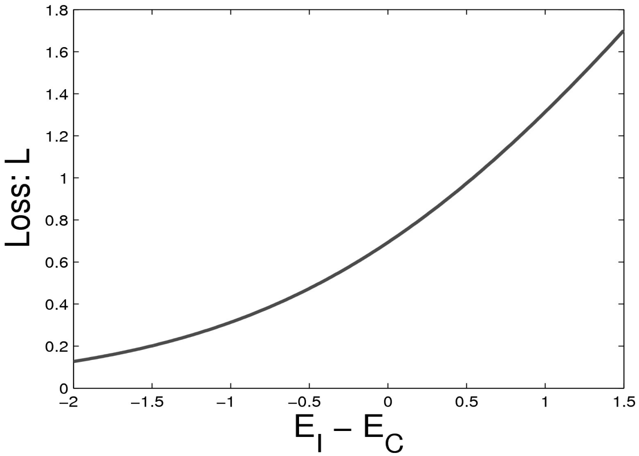
- [Altun et al. 2003], [Taskar et al. 2003]
- With the linearly-parameterized binary classifier architecture, we get linear SVMs



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

Log Loss

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

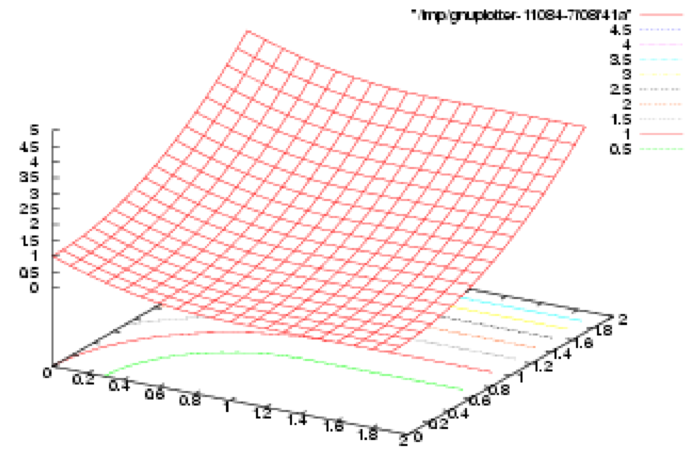


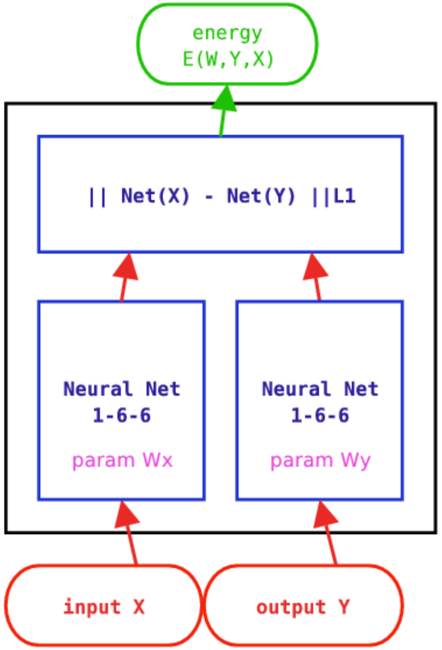
Examples of Margin Losses: Square-Square Loss

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

Square-Square Loss

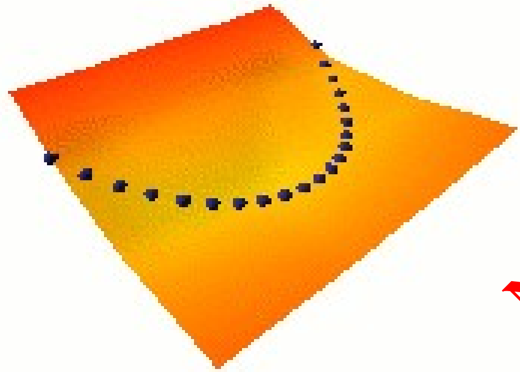
- [LeCun-Huang 2005]
- Appropriate for positive energy functions





(b)

Learning Y = X^2



NO COLLAPSE!!!

Other Margin-Like Losses

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

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

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

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

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

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

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

What Make a "Good" Loss Function

Good and bad loss functions

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

Slightly more general form:

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

Advantages/Disadvantages of various losses

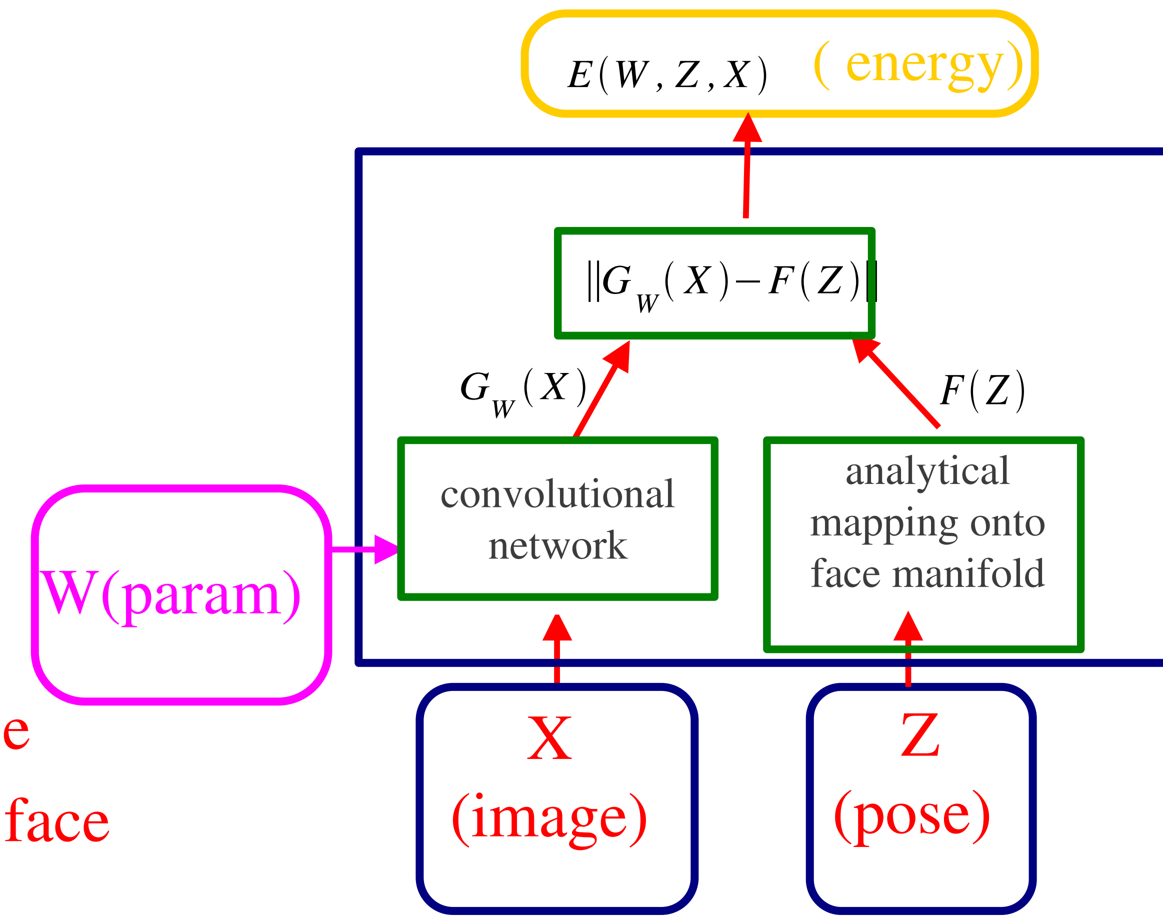
- **Loss functions differ in how they pick the point(s) whose energy is pulled up, and how much they pull them up**
- **Losses with a log partition function in the contrastive term pull up all the bad answers simultaneously.**
  - This may be good if the gradient of the contrastive term can be computed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

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 .

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

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

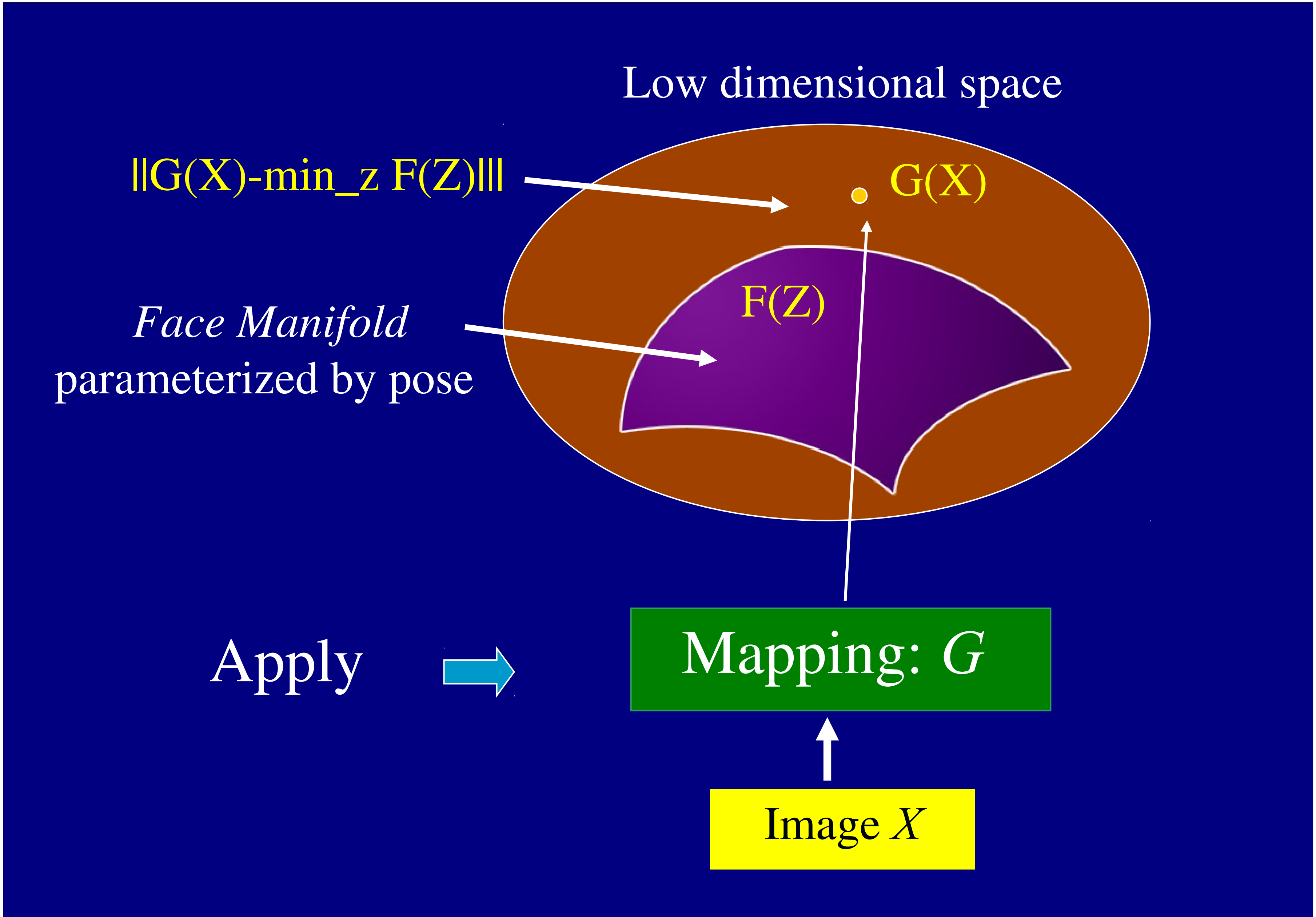


Small E*(W,X): face

Large E*(W,X): no face

[Osadchy, Miller, LeCun, NIPS 2004]

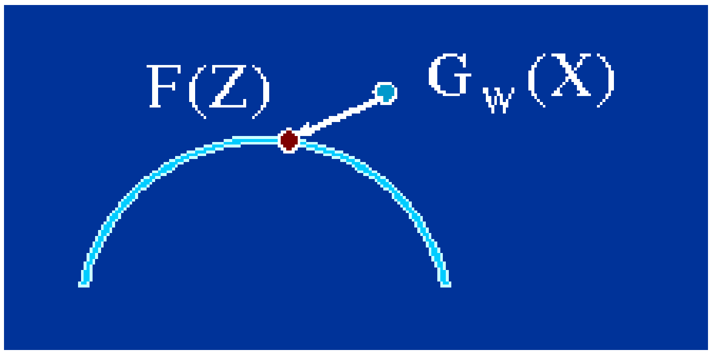
Face Manifold



Energy-Based Contrastive Loss Function

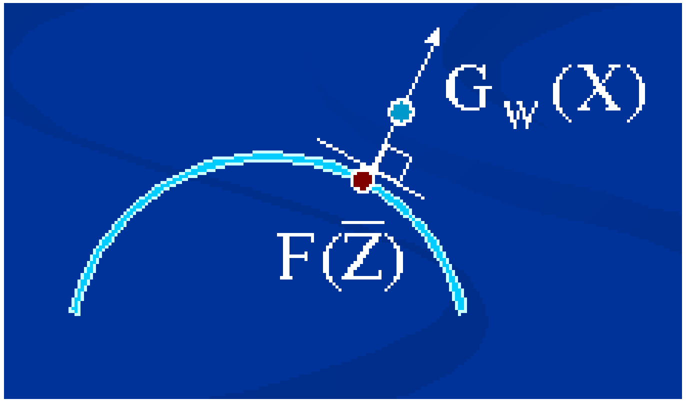
$$\mathcal{L}(W) = \frac{1}{|\mathrm{f}+\mathrm{p}|} \sum_{X,Z \in \mathrm{faces+pose}} \left[L^{+}\left(E(W,Z,X)\right)\right] + L^{-}\left(\min_{X,Z \in \mathrm{bckgnd,poses}} E(W,Z,X)\right)$$

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



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

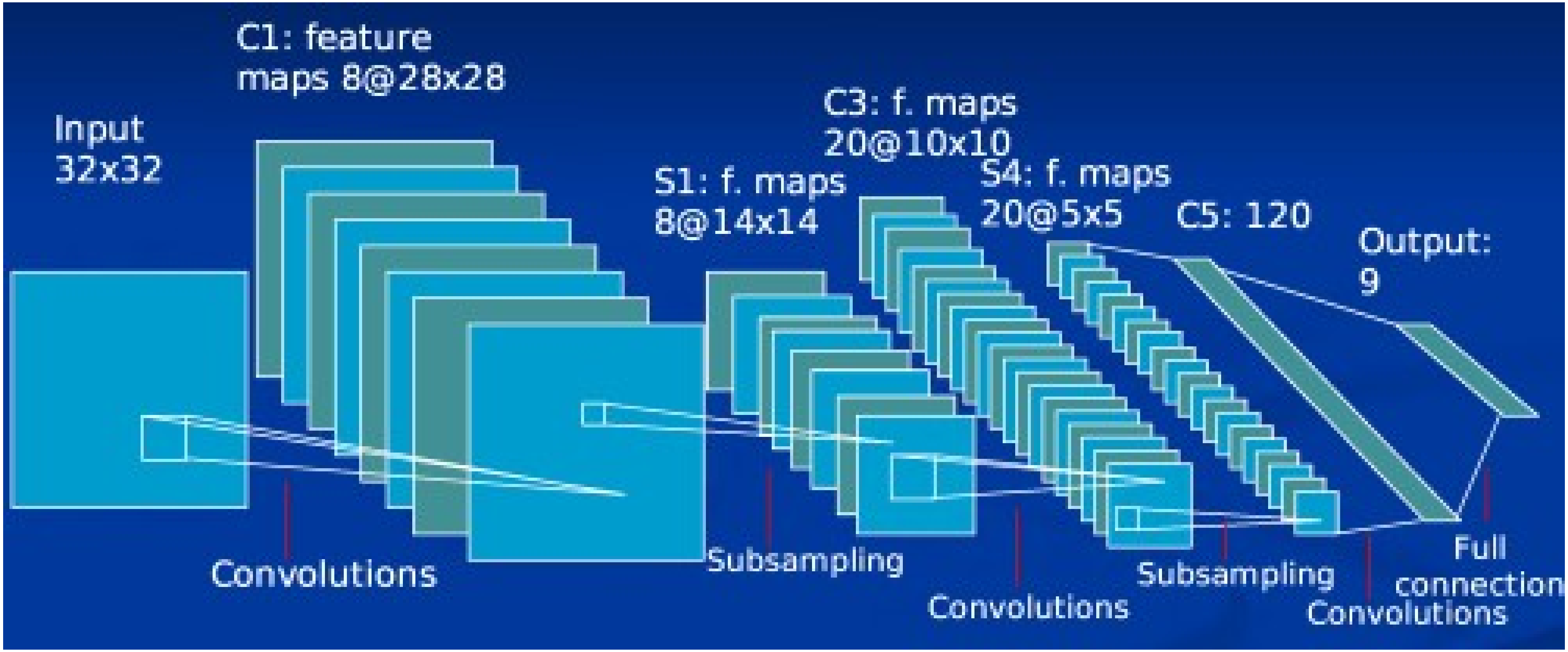
$$L^{-}\left(\min_{X,Z \in \mathrm{bckgnd,poses}} E(W,Z,X))\right) = K \exp\left(-\min_{X,Z \in \mathrm{bckgnd,poses}} ||G_W(X) - F(Z)||\right)$$



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

Convolutional Network Architecture

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

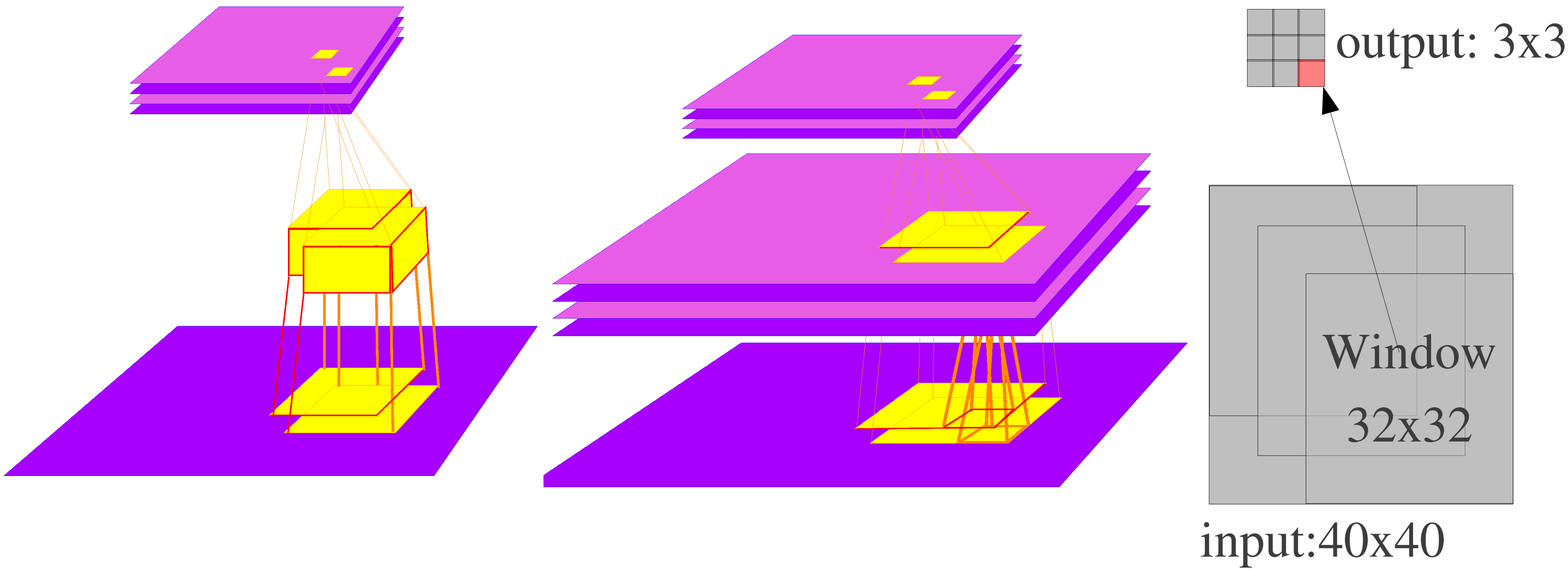


Hierarchy of local filters (convolution kernels),

sigmoid pointwise non-linearities, and spatial subsampling

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

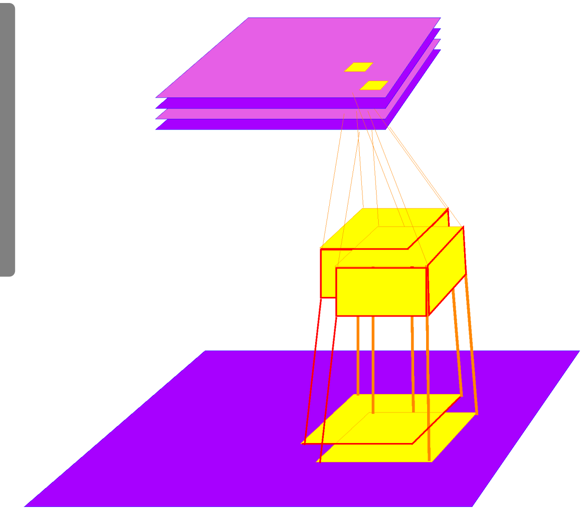
Building a Detector/Recognizer: Replicated Conv. Nets

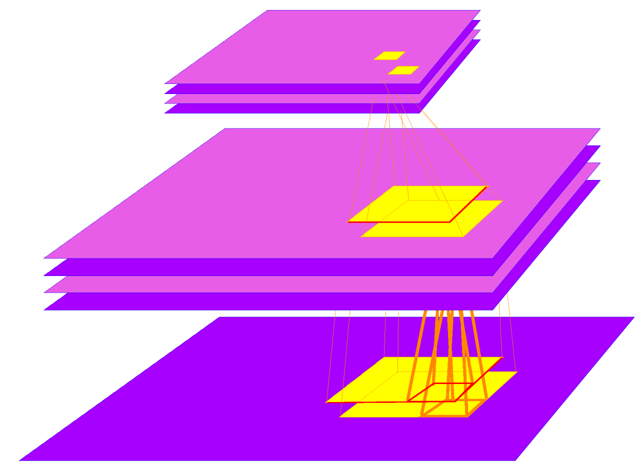


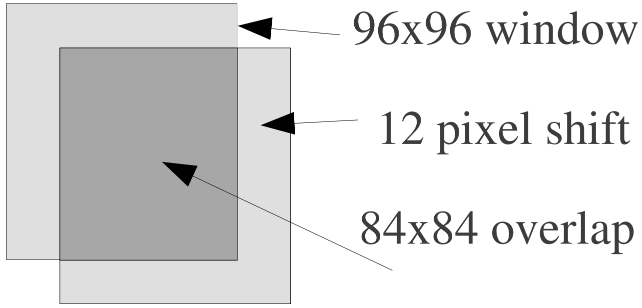
- Traditional Detectors/Classifiers must be applied to every location on a large input image, at multiple scales.
- Convolutional nets can replicated over large images very cheaply.
- The network is applied to multiple scales spaced by sqrt(2)
- Non-maximum suppression with exclusion window

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

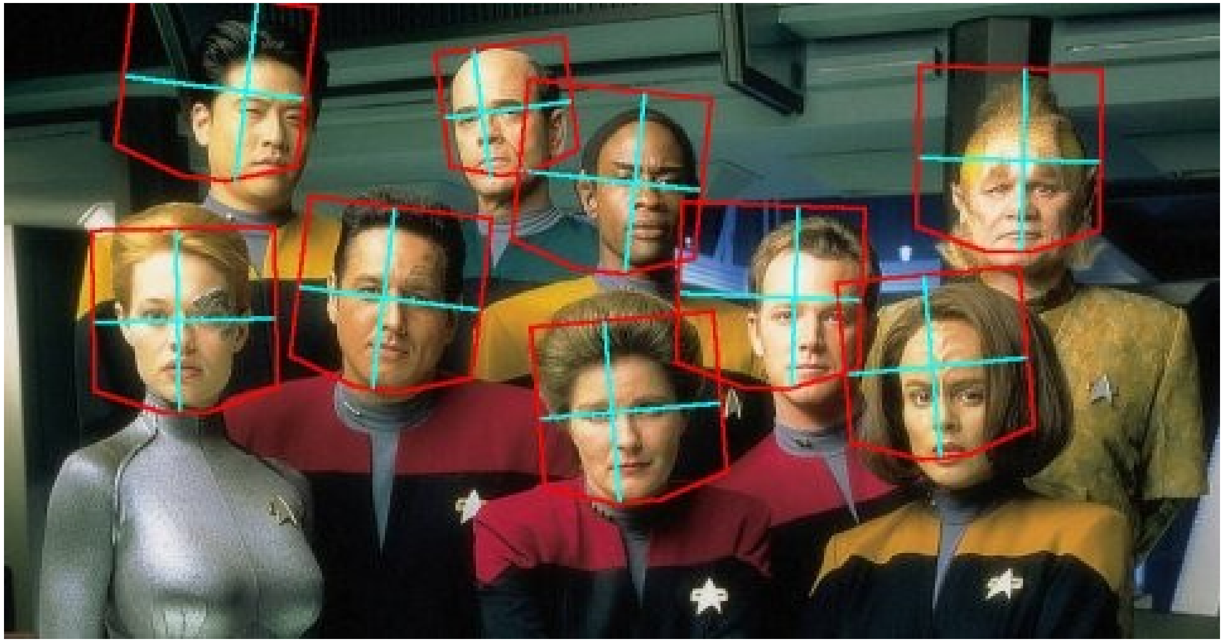


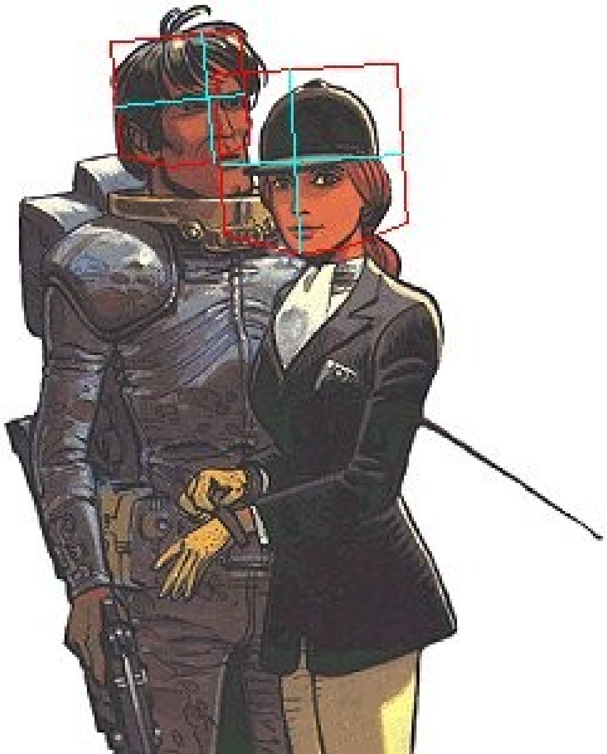




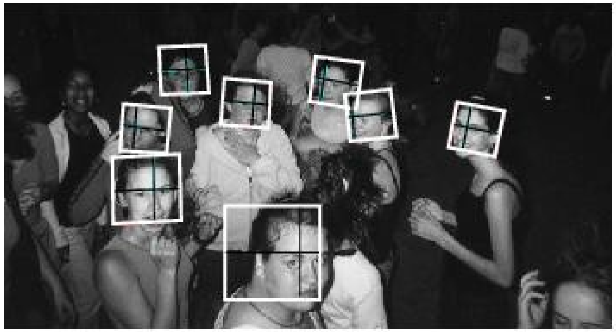
Face Detection: Results

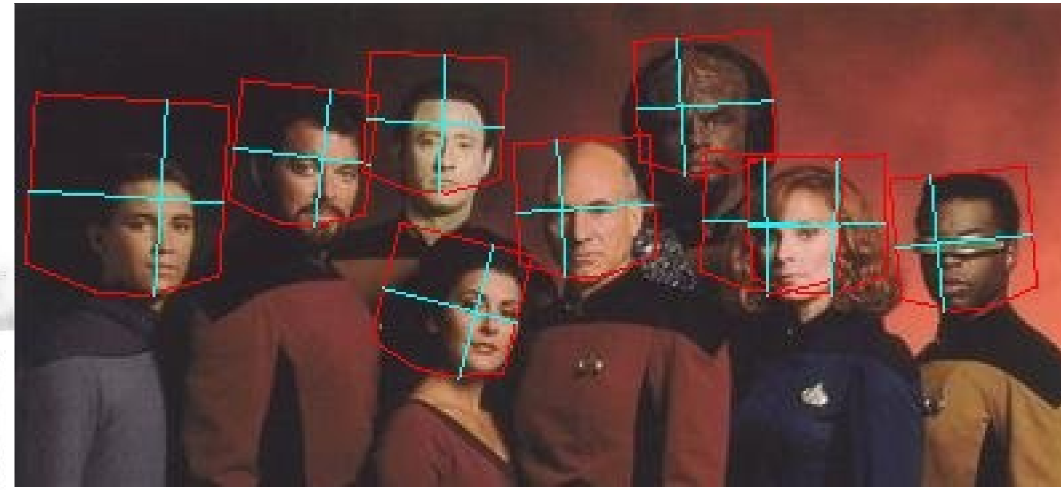
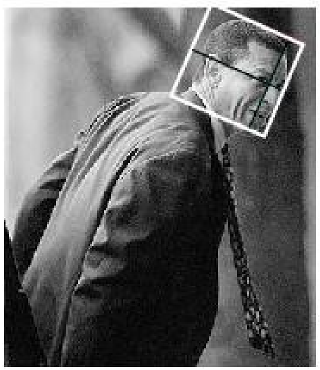
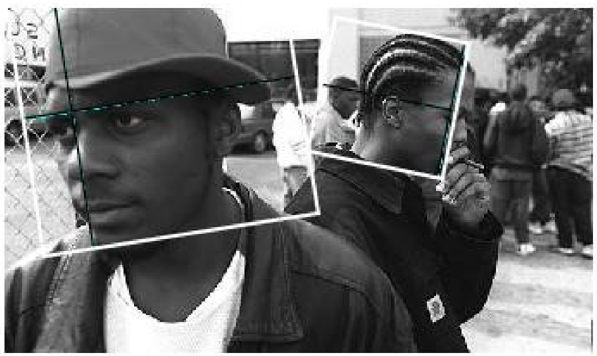
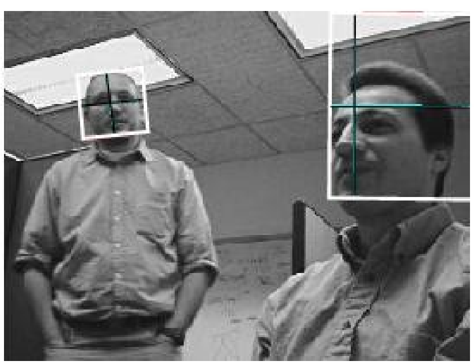
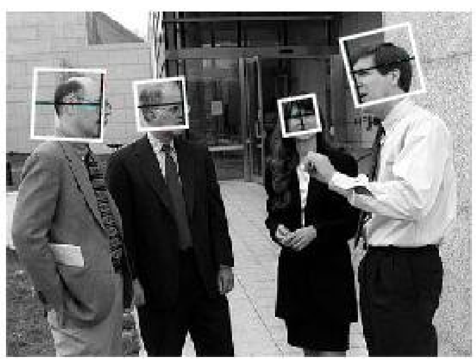
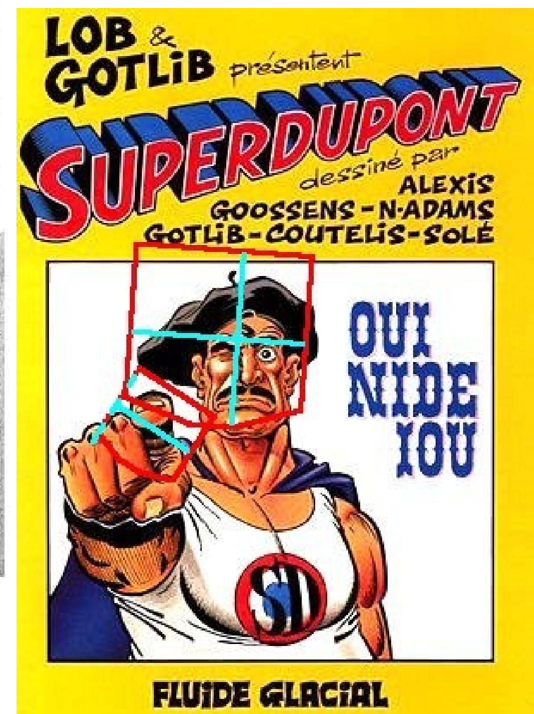
Data Set-> / *False positives per image->*	TILTED		PROFILE		MIT+CMU	
	4.42	26.9	0.47	3.36	0.5	1.28
Our Detector	90%	97%	67%	83%	83%	88%
Jones & Viola (tilted)	90%	95%	x		x	
Jones & Viola (profile)	x		70%	83%	x	

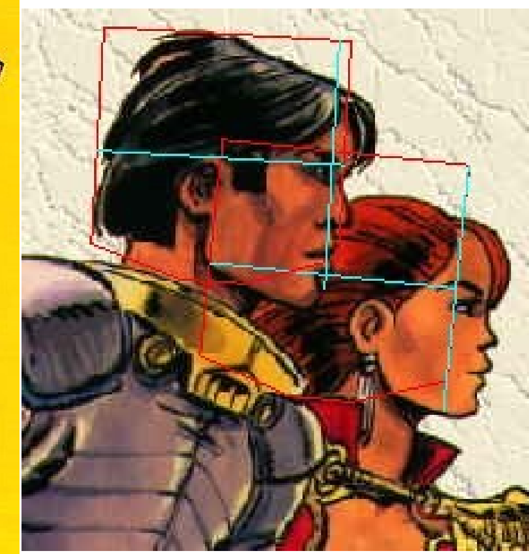




Face Detection and Pose Estimation: Results

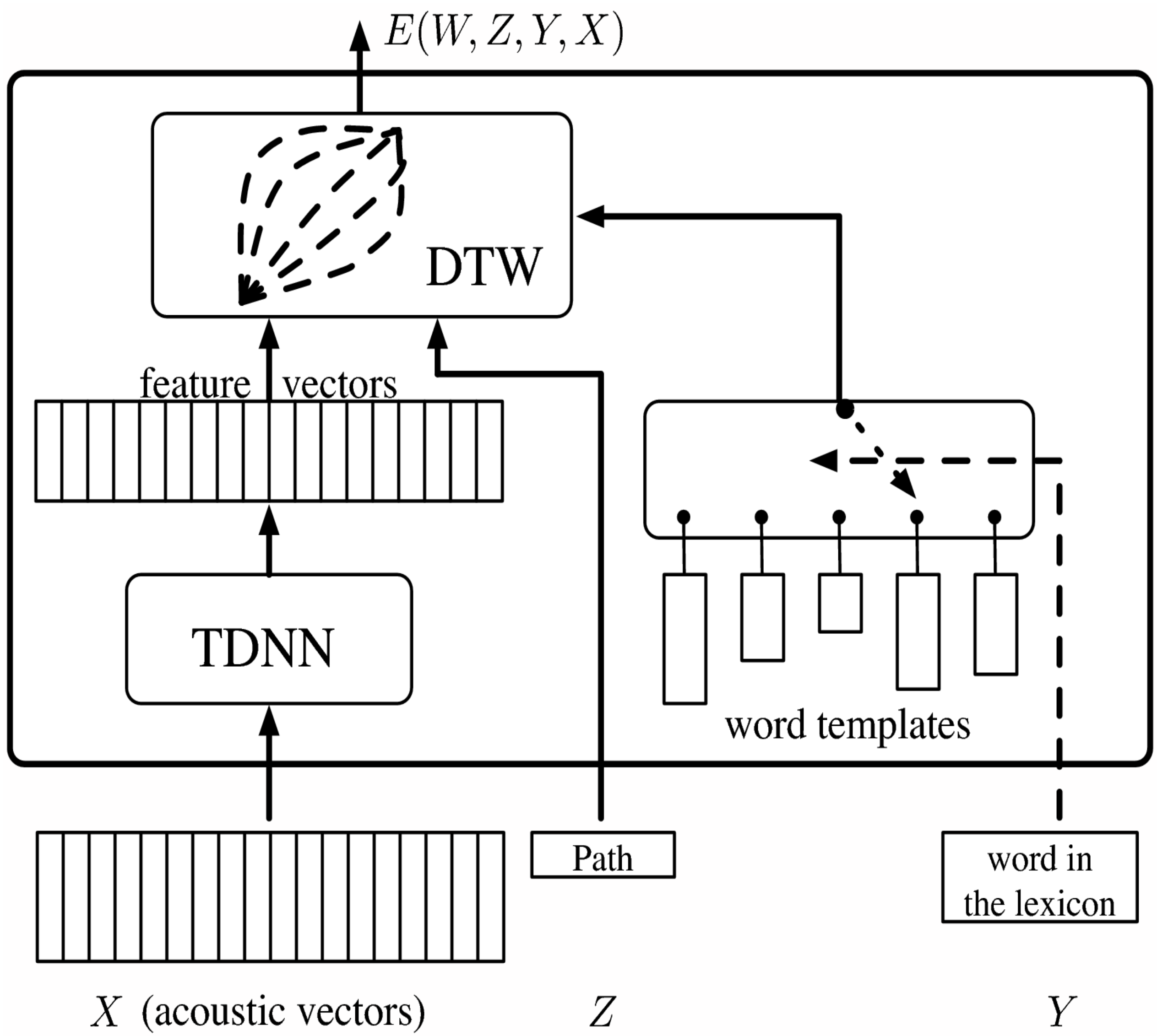




The Oldest Example of Structured Prediction

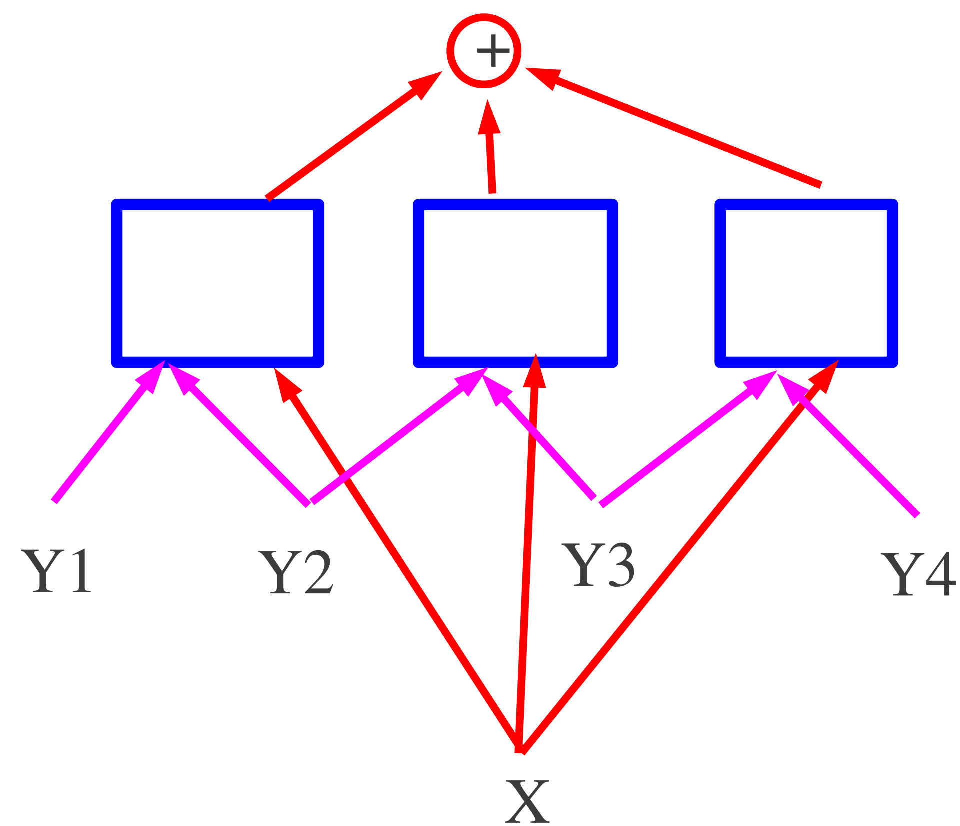
- **Trainable Automatic Speech Recognition system with a convolutional net (TDNN) and dynamic time warping (DTW)**
- **The feature extractor and the structured classifier are trained simultanesously in an integrated fashion.**
- **with the LVQ2 Loss :**
  - Driancourt and Bottou's speech recognizer (1991)
- **with NLL:**
  - Bengio's speech recognizer (1992)
  - Haffner's speech recognizer (1993)



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

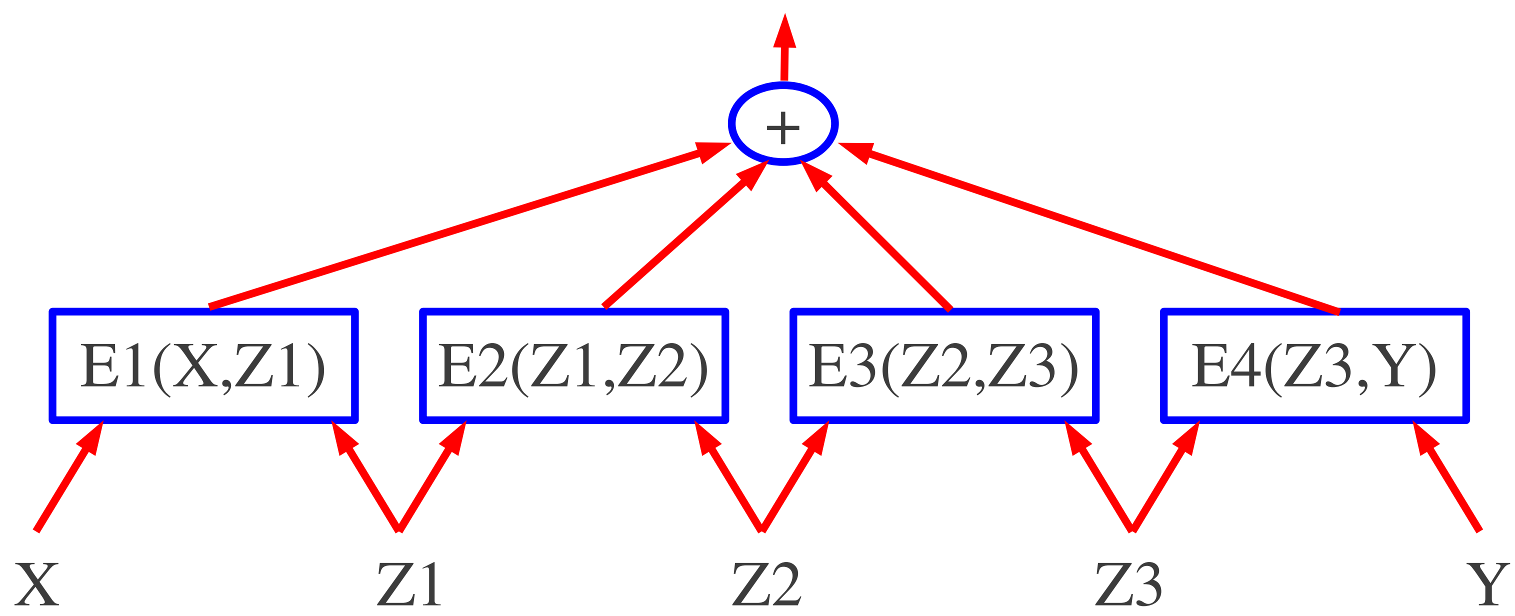
- **Sequence Labeling**
  - Output is a sequence Y1,Y2,Y3,Y4......
  - NLP parsing, MT, speech/handwriting recognition, biological sequence analysis
  - The factors ensure grammatical consistency
  - They give low energy to consistent sub-sequences of output symbols
  - The graph is generally simple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

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



Energy-Based Factor Graphs

- **When the energy is a sum of partial energy functions (or when the probability is a product of factors):**
  - Efficient inference algorithms can be used for inference (without the normalization step).

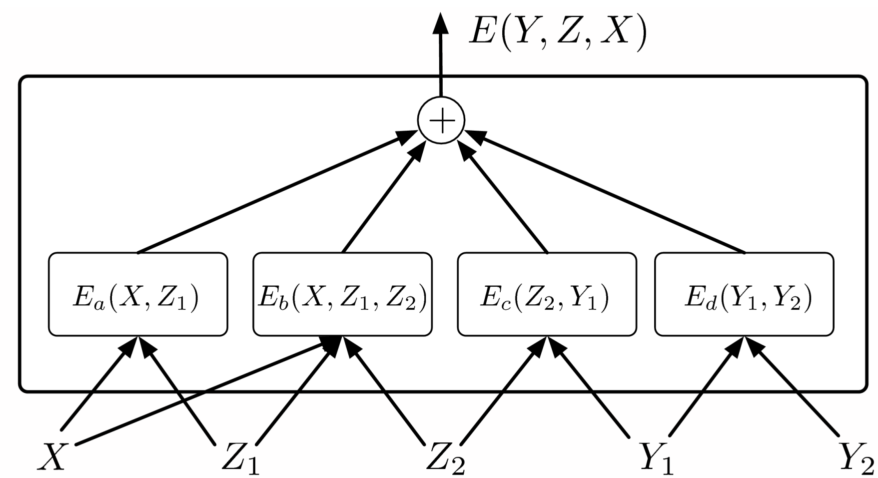


Efficient Inference: Energy-Based Factor Graphs

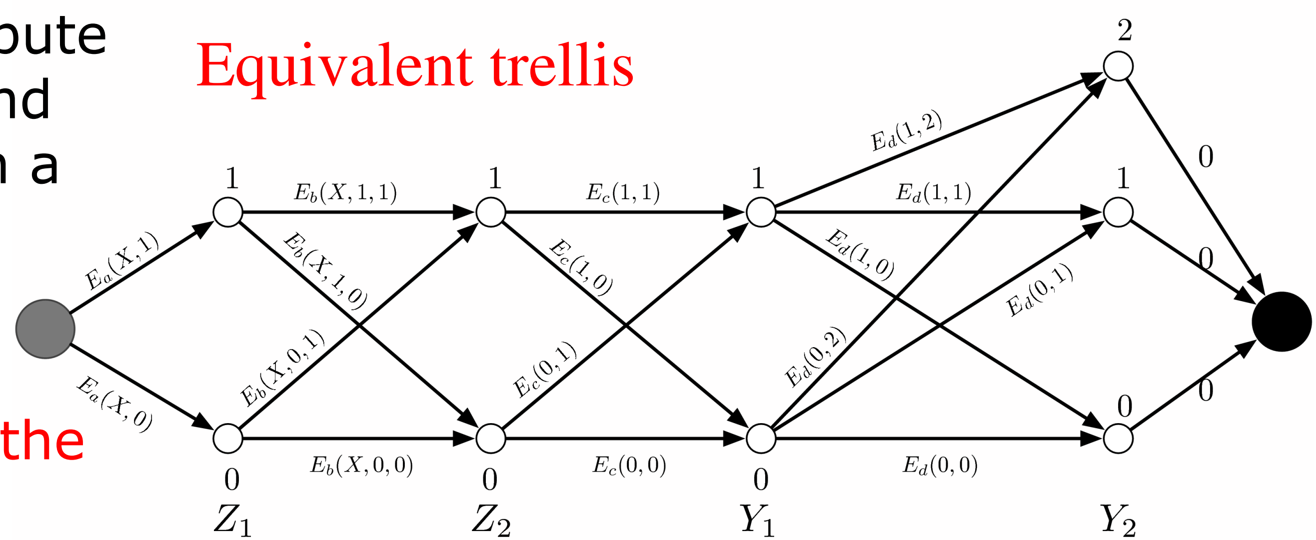
- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3=24 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 6.
  - Hence, we can precompute the 16 factor values, and put them on the arcs in a trellis.
  - A path in the trellis is a config of variable
  - The cost of the path is the energy of the config

- **The energy is a sum of "factor" functions**

Factor graph



Equivalent trellis

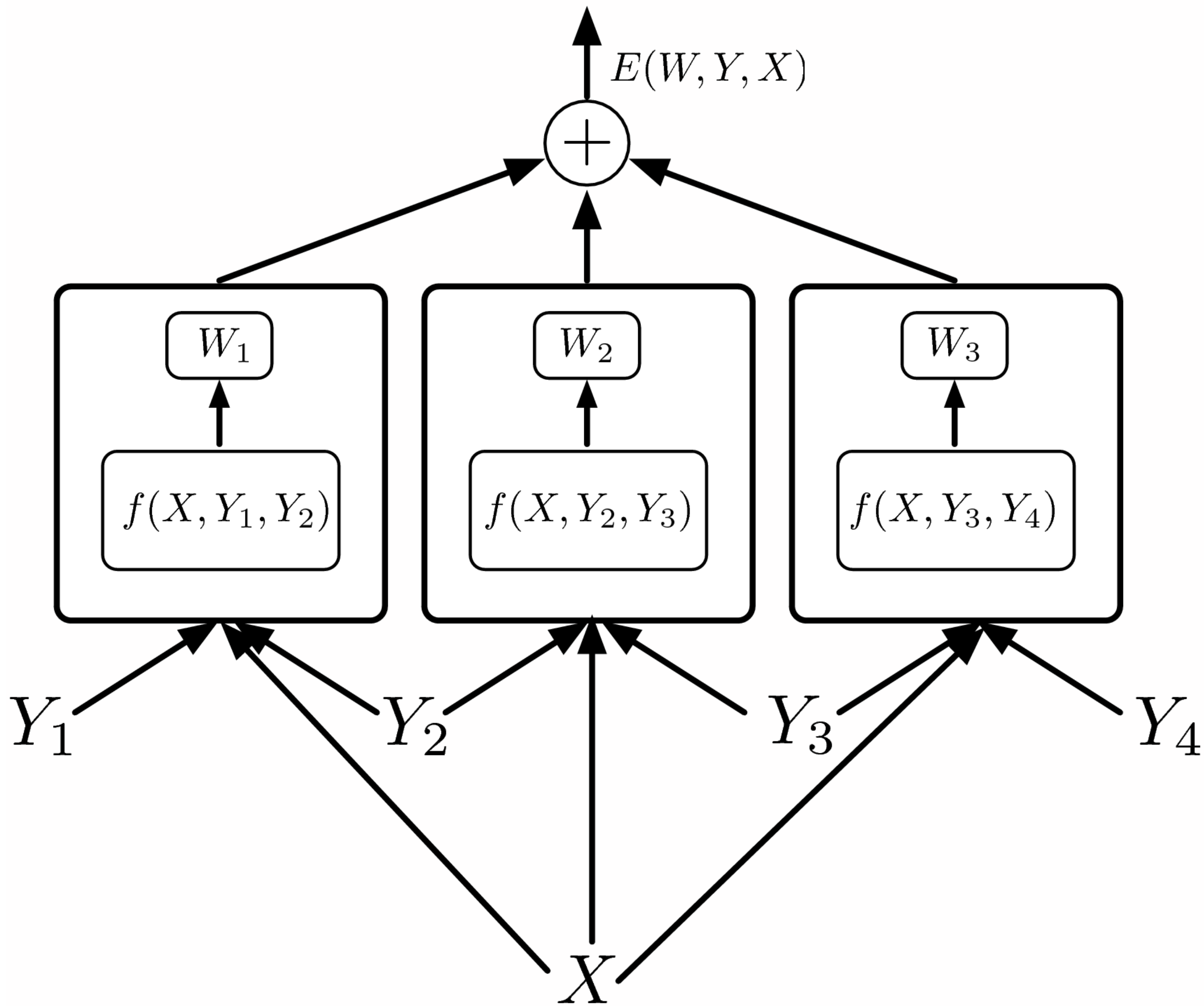


Energy-Based Belief Prop

- **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]

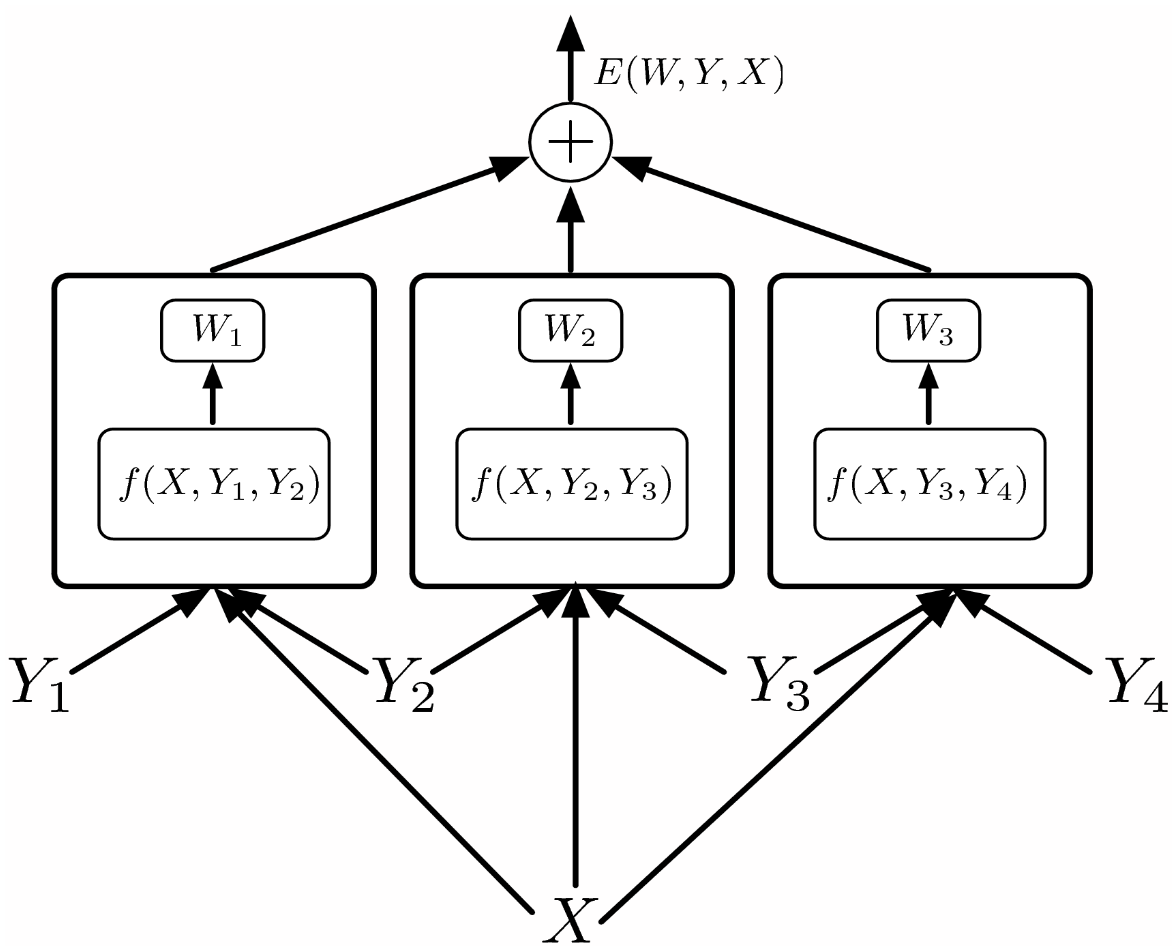
Simple Energy-Based Factor Graphs with "Shallow" Factors

- **Linearly Parameterized Factors**
- **with the NLL Loss :**
  - Lafferty's **Conditional Random Field**
- **with Hinge Loss:**
  - Taskar and Altun/Hofmann's **Max Margin Markov Nets** and **Latent SVM**
- **with Perceptron Loss**
  - Collins's **Structured Perceptron** model



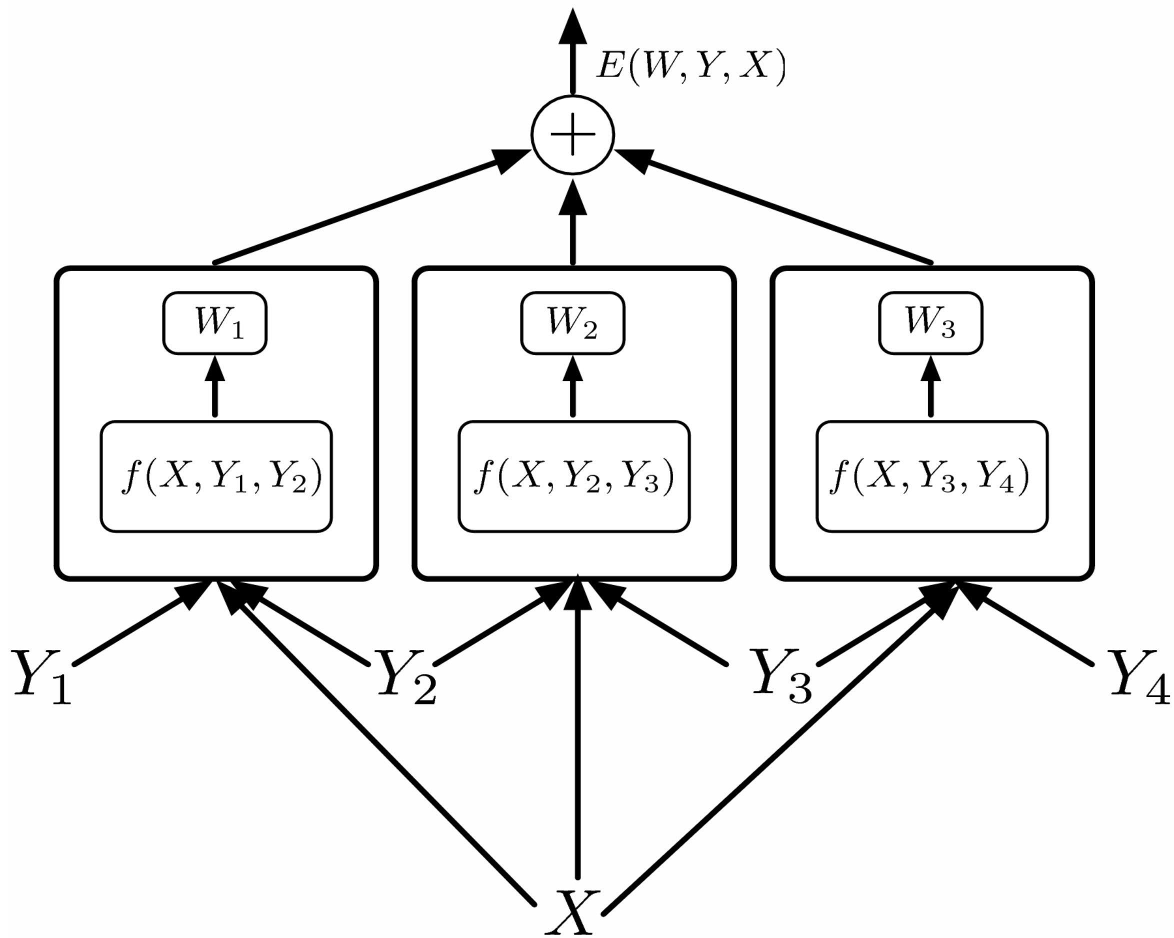
Example : The Conditional Random Field Architecture

- **A CRF is an energy-based factor graph in which:**
  - the factors are **linear in the parameters** **(shallow factors)**
  - The factors take neighboring output variables as inputs
  - The factors are often all identical



Example : The Conditional Random Field Architecture

- **Applications:**
  - X is a sentence, Y is a sequence of Parts of Speech Tags (there is one Yi for each possible group of words).
  - X is an image, Y is a set of labels for each window in the image (vegetation, building, sky....).



Deep/non-linear Factors for Speech and Handwriting

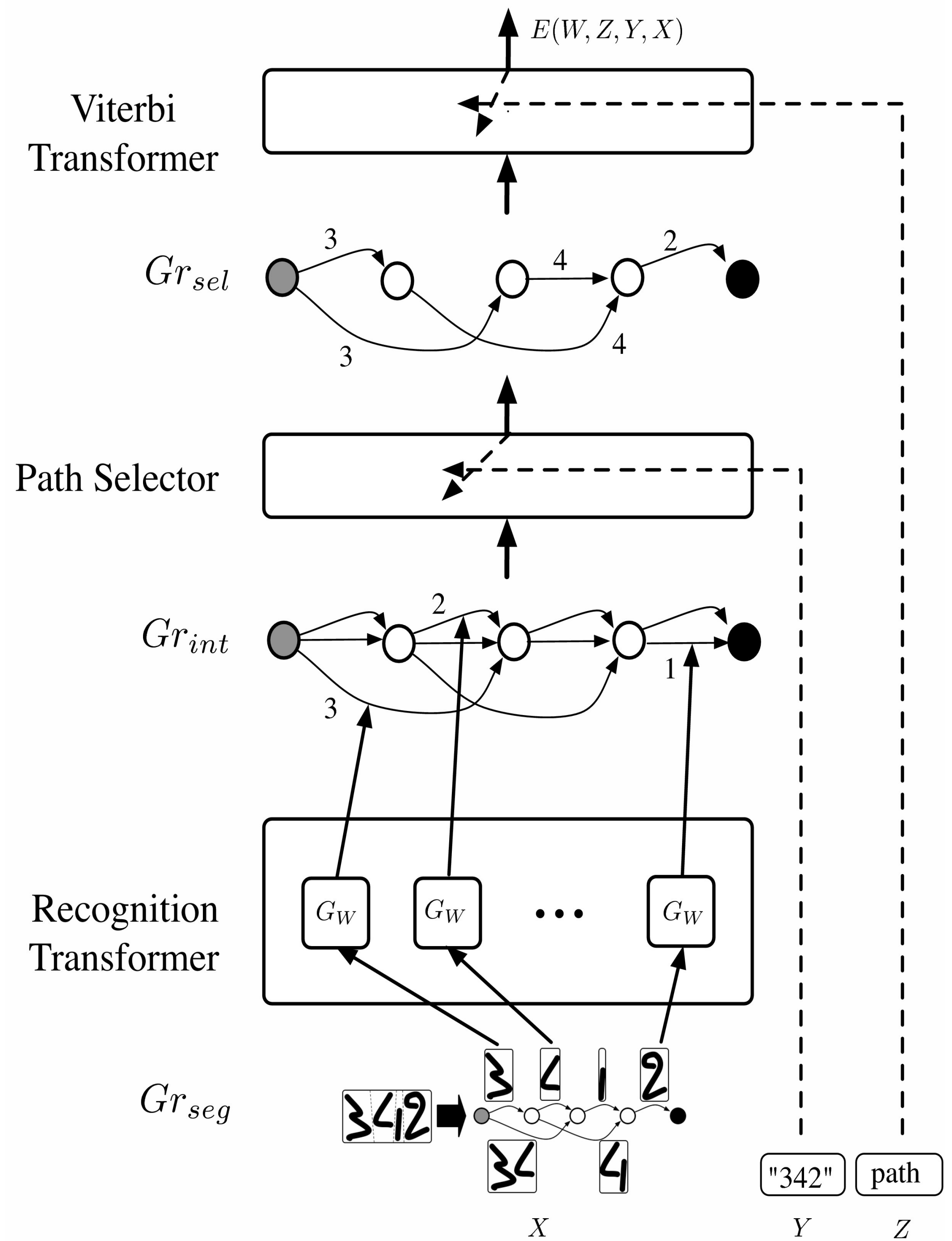
- Trainable Speech/Handwriting Recognition systems that integrate Neural Nets (or other "deep" classifiers) with dynamic time warping, Hidden Markov Models, or other graph-based hypothesis representations

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

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

Deep Factors & implicit graphs: GTN

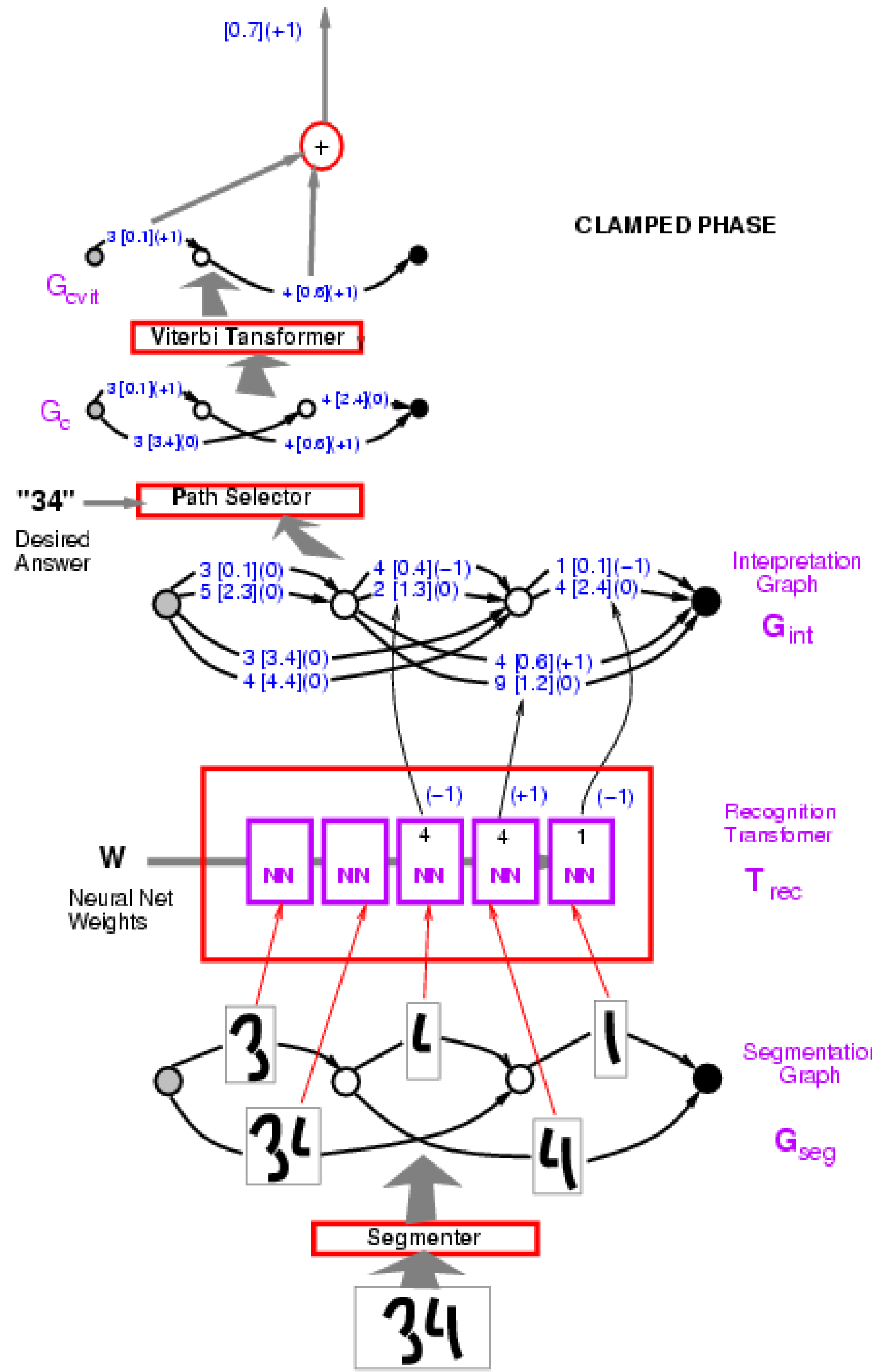
- **Handwriting Recognition with Graph Transformer Networks**
- **Un-normalized hierarchical HMMs**
  - Trained with Perceptron loss [LeCun, Bottou, Bengio, Haffner 1998]
  - Trained with NLL loss [Bengio, LeCun 1994], [LeCun, Bottou, Bengio, Haffner 1998]
- **Answer = sequence of symbols**
- **Latent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

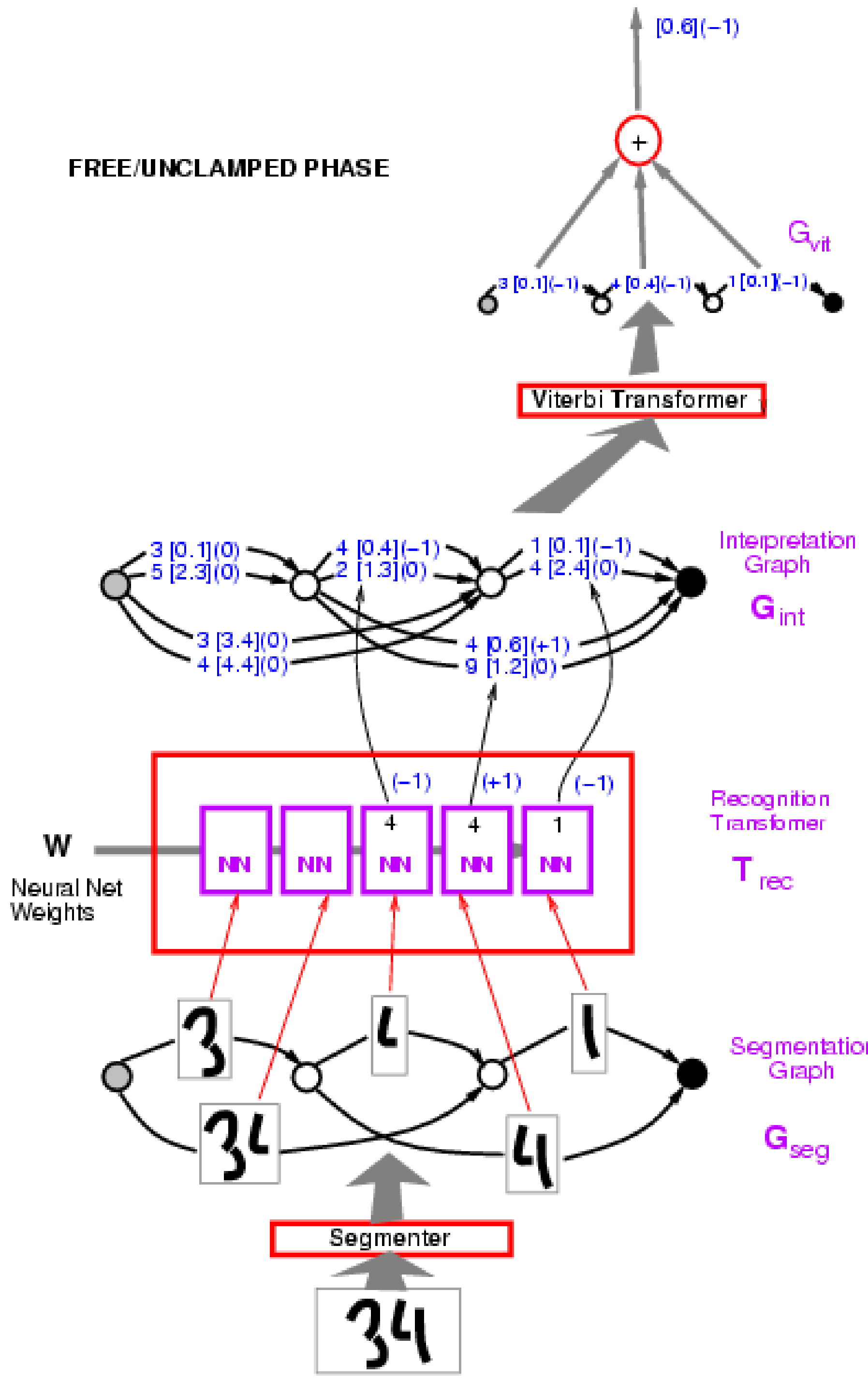
$$E(W, Y, X)$$



Graph Transformer Networks

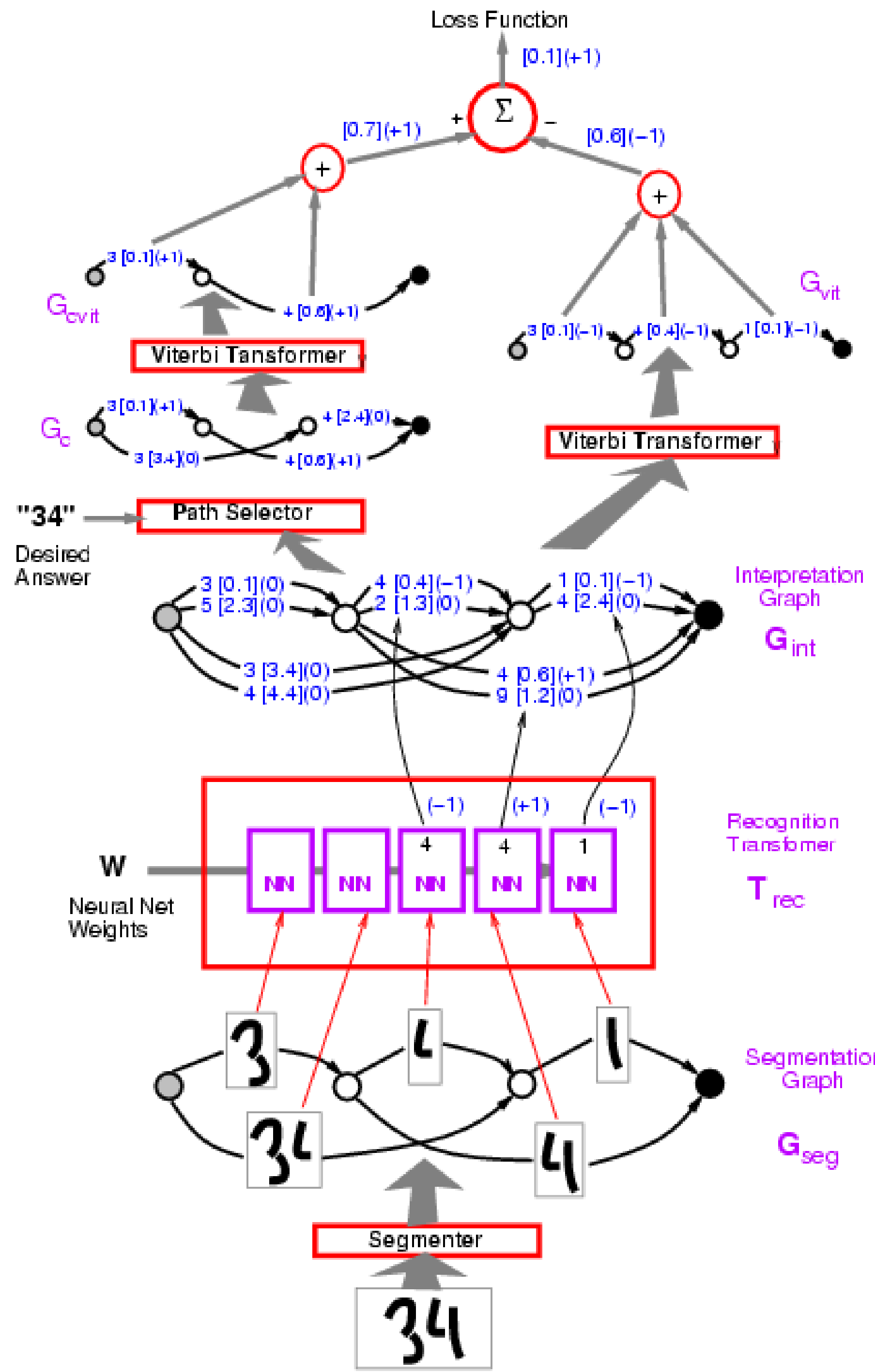
- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the constrastive term:**

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



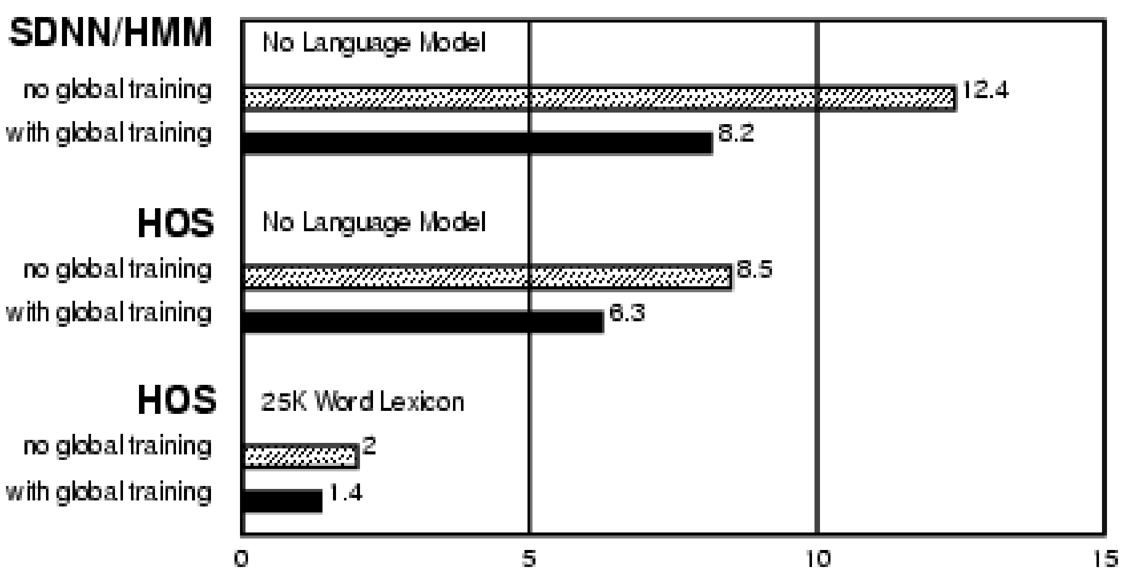
Graph Transformer Networks

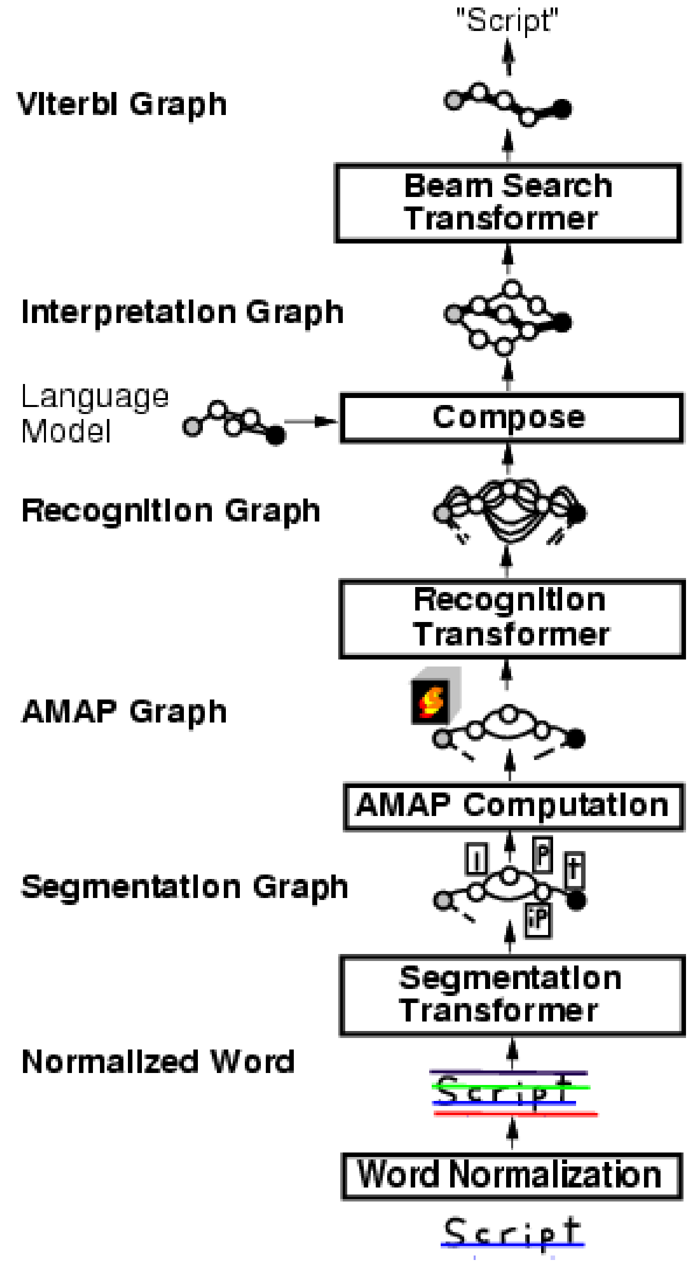
- **Example: Perceptron loss**
- **Loss = Energy of desired answer – Energy of best answer.**
  - (no margin)



Global Training Helps

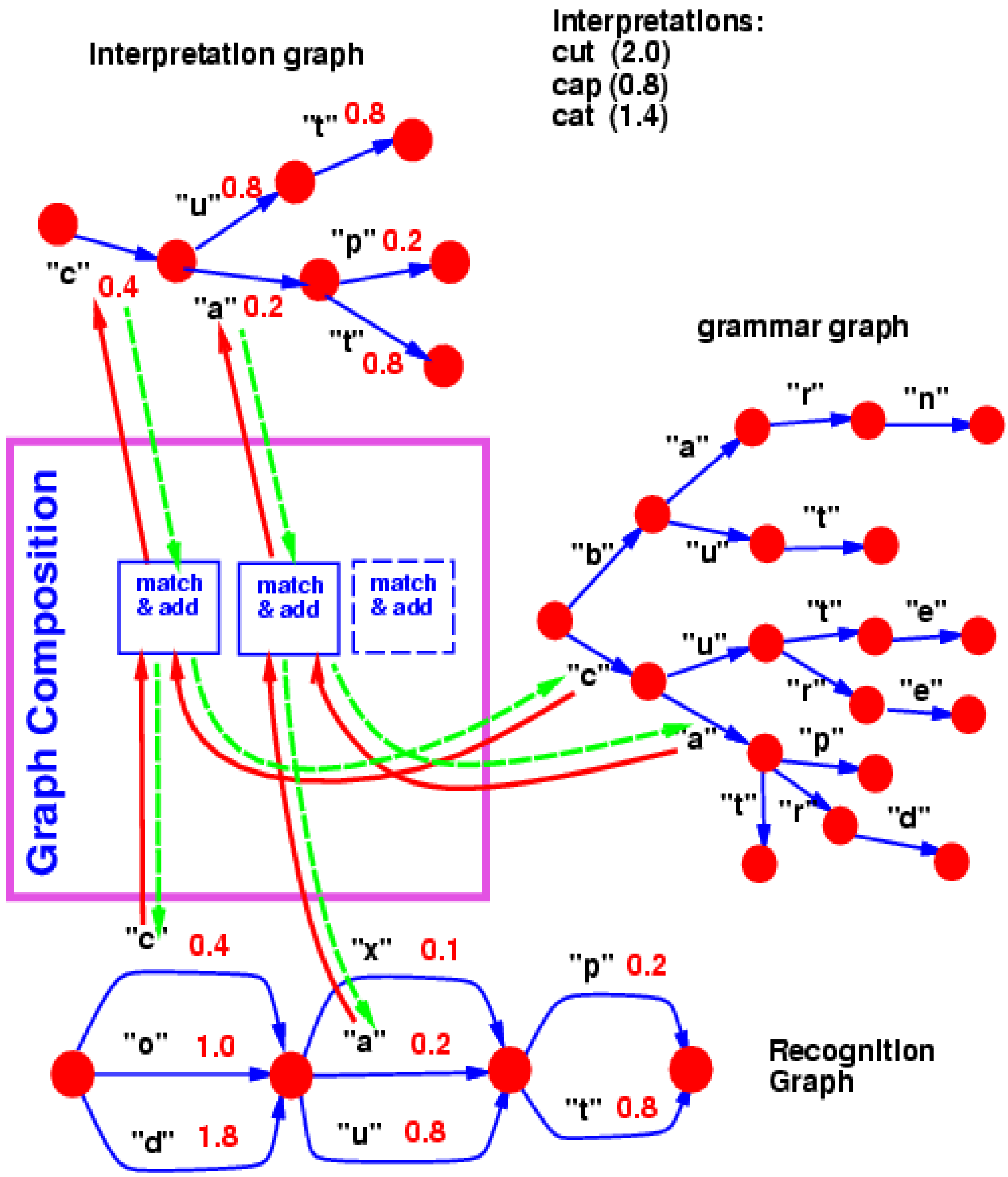
- **Pen-based handwriting recognition (for tablet computer)**
  - [Bengio&LeCun 1995]





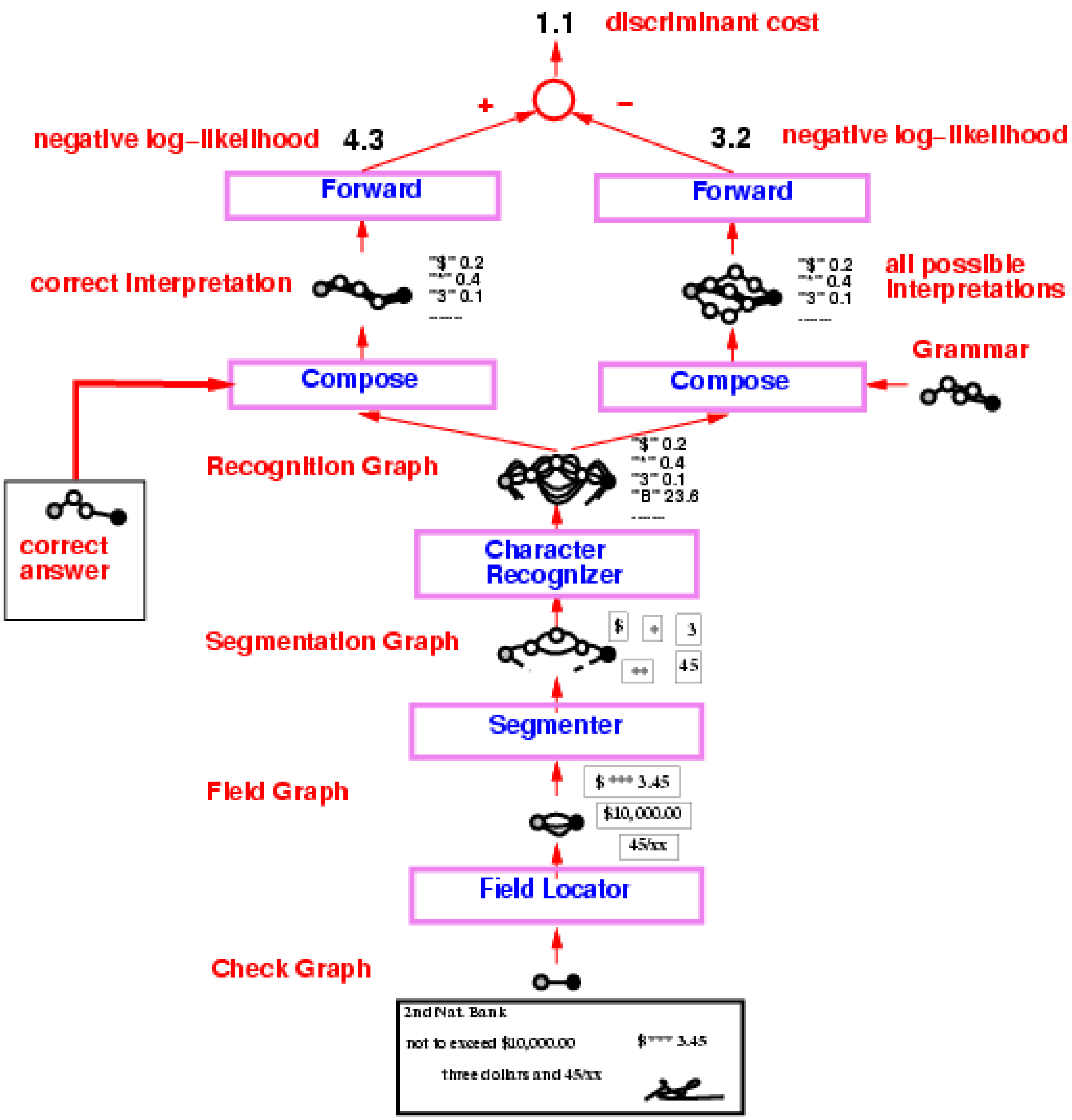
Graph Composition, Transducers.

- 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 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.

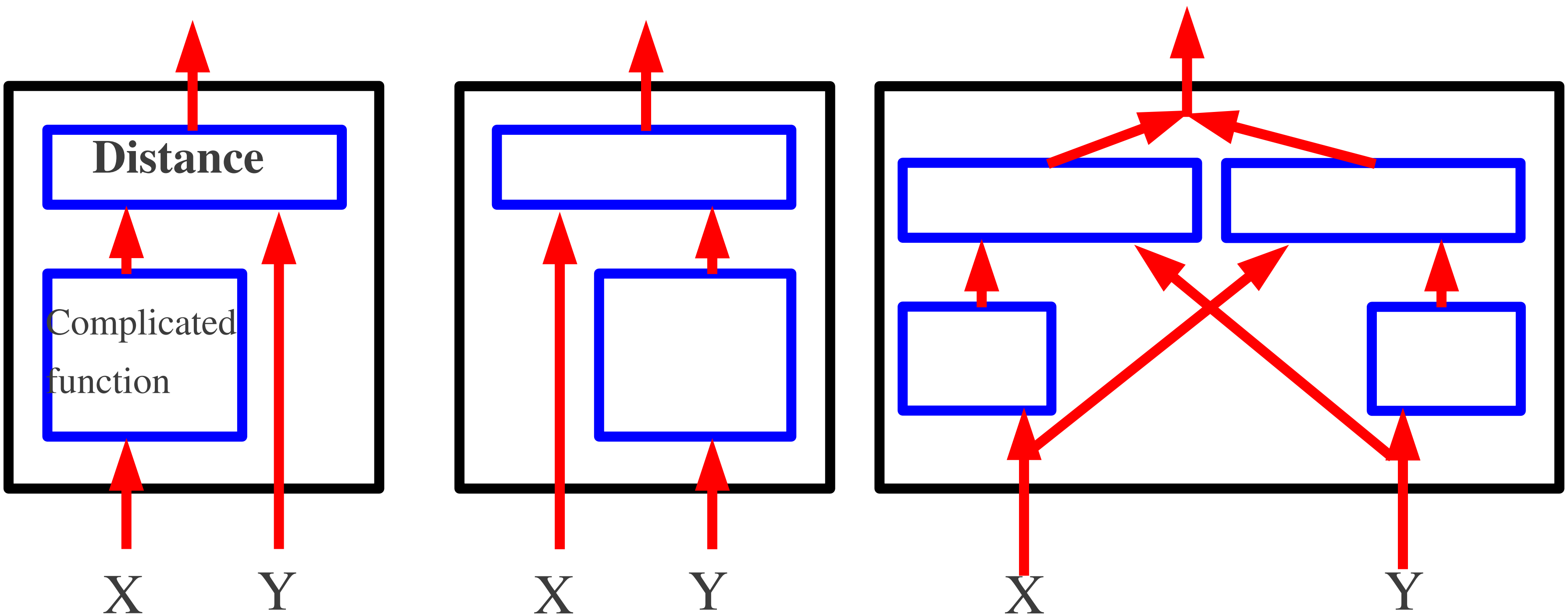


Deep Factors / Deep Graph: ASR with TDNN/HMM

- **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)

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