Factor Graphs
Structured Prediction

Yann LeCun,

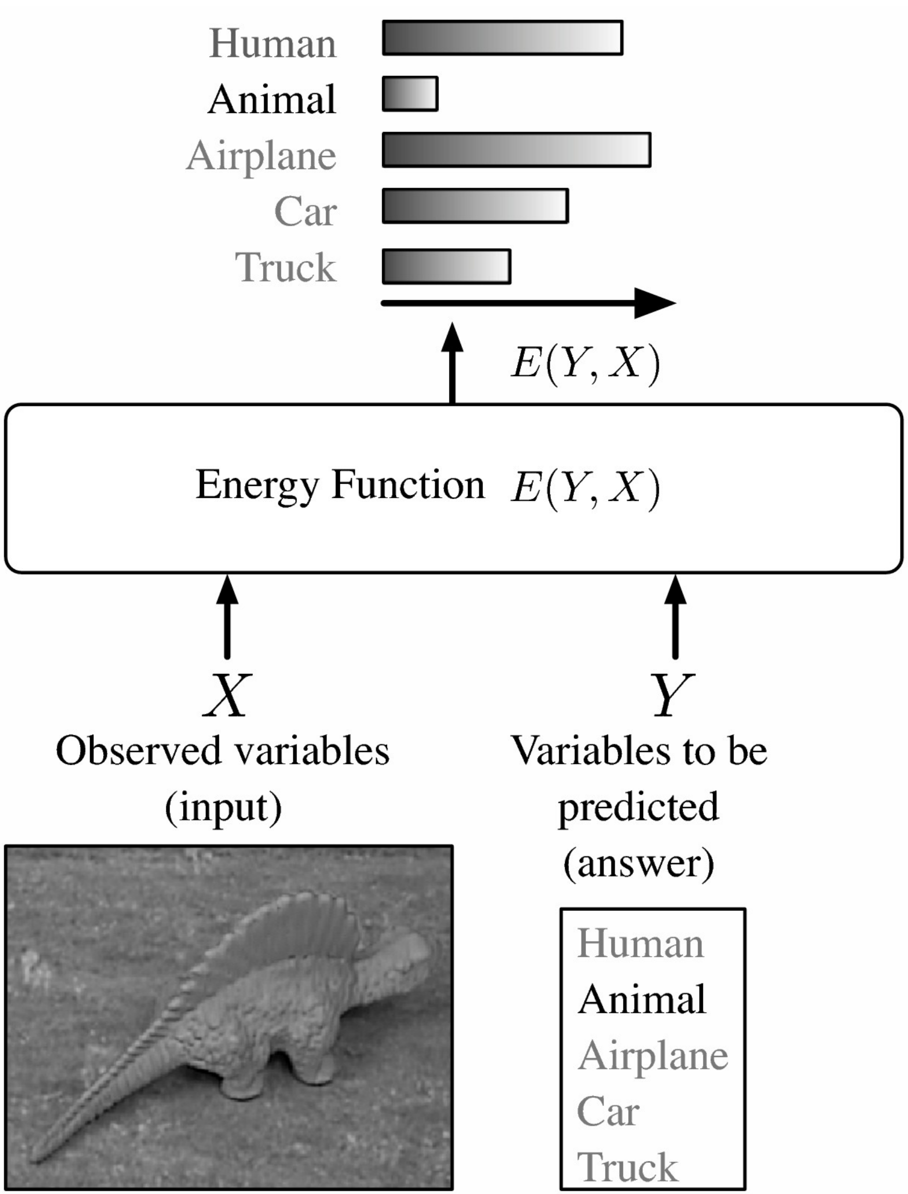
The Courant Institute of Mathematical Sciences

New York University

http://yann.lecun.com

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

Energy-Based Model for Decision-Making

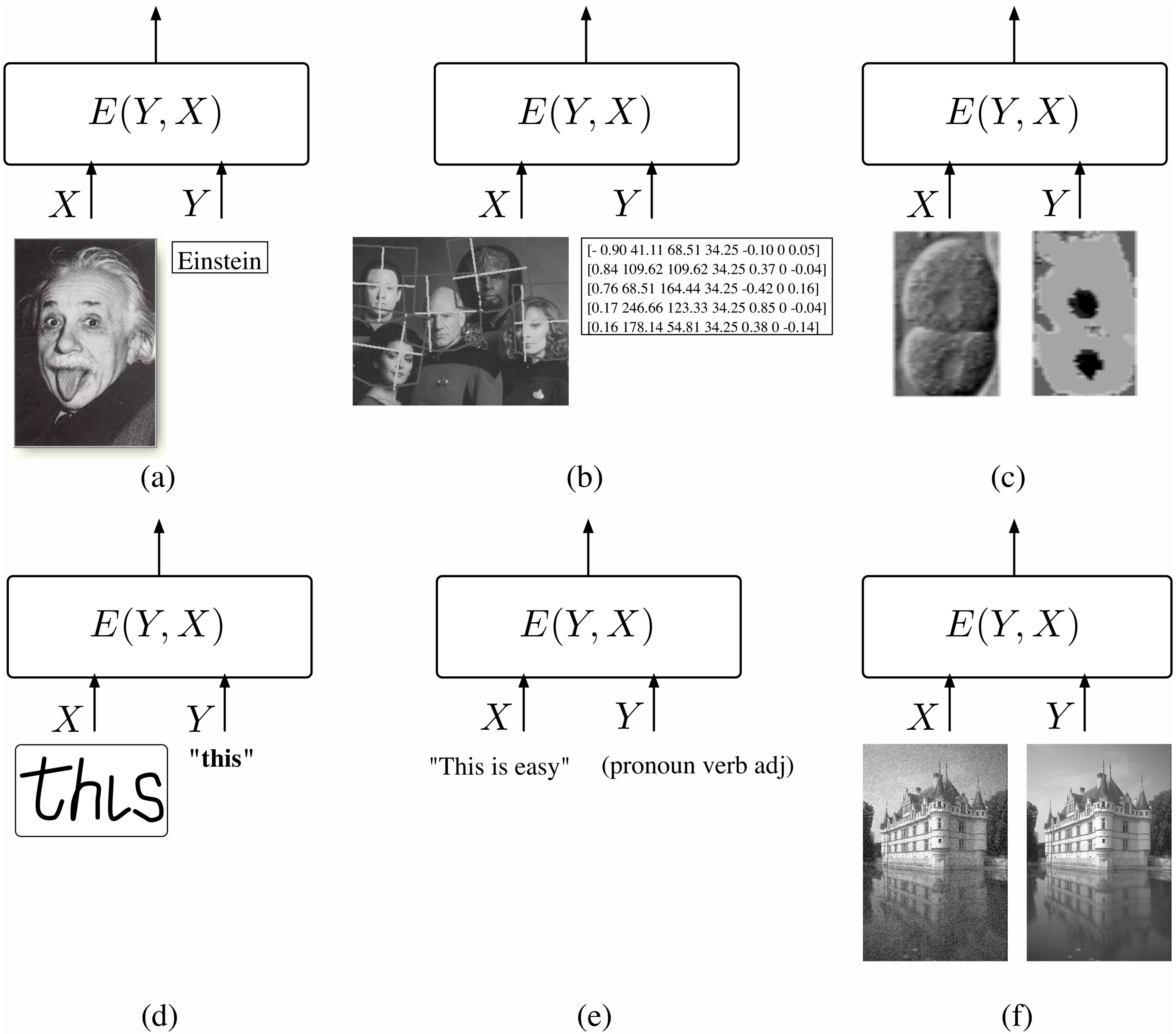


- **Model:** **Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).**

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

- **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, ...**

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

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:*

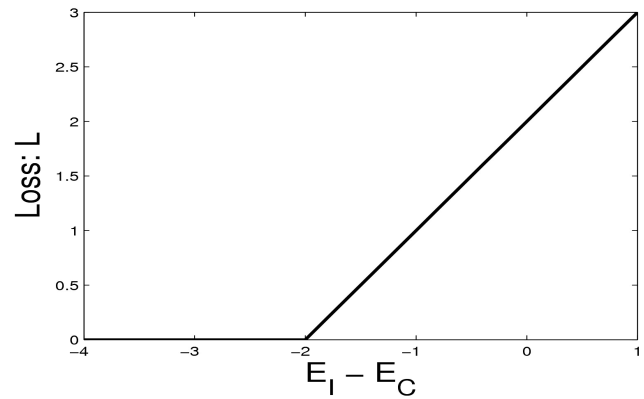
$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \quad (9)$$

Examples of 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

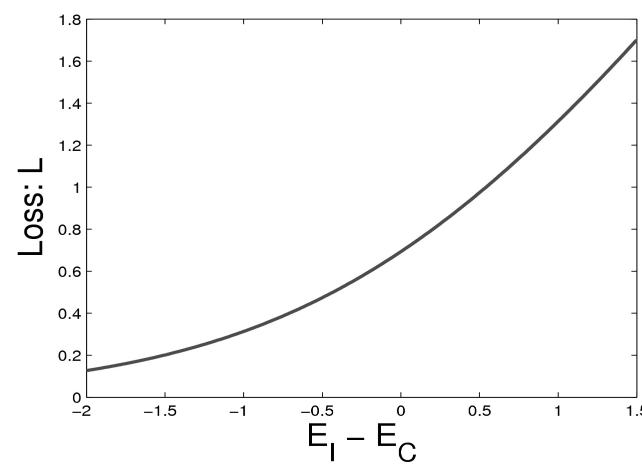
- [Altun et al. 2003], [Taskar et al. 200
- With the linearly-parameterized binary classifier architecture, we get linear SV



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

Log Loss

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



Negative Log-Likelihood Loss

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

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

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

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

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

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

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

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

What Make a "Good" Loss Function

Good and bad loss functions

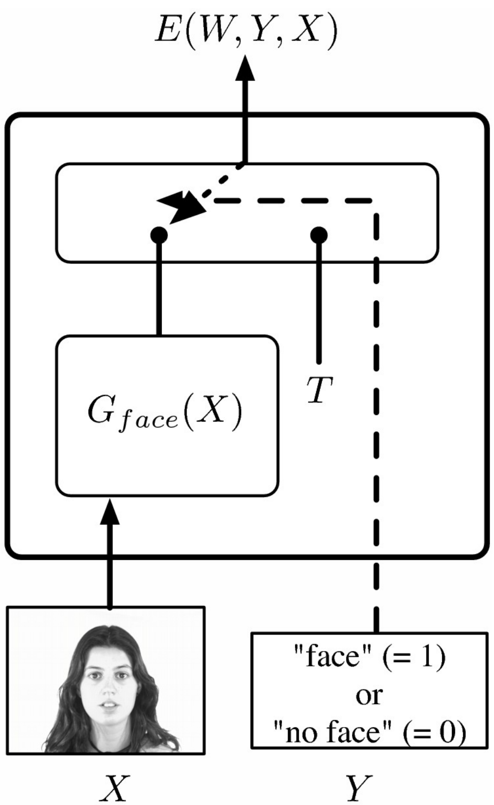
Loss (equation #)	**Formula**	**Margin**
energy loss	$E(W, Y^i, X^i)$	none
perceptron	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0
LVQ2	$\min\left(M,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

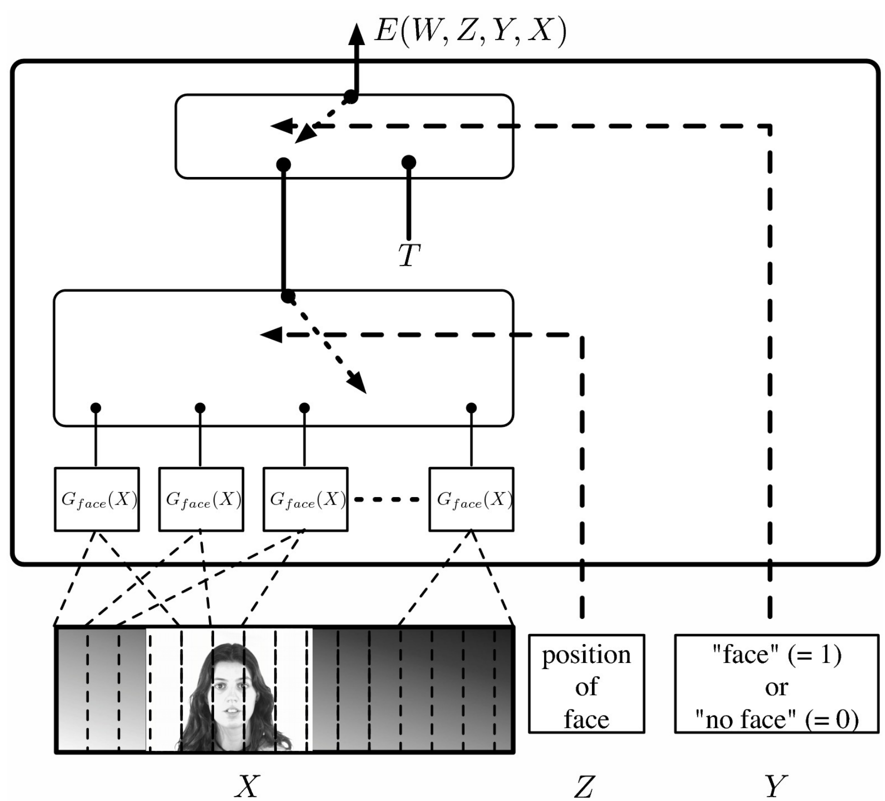
Latent Variable Models

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

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

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





Latent variables in Weakly Supervised Learning

- **Variables that would make the task easier if they were known:**
  - **Scene Analysis**: segmentation of the scene into regions or objects.
  - **Parts of Speech Tagging**: the segmentation of the sentence into syntactic units, the parse tree.
  - **Speech Recognition**: the segmentation of the sentence into phonemes or phones.
  - **Handwriting Recognition**: the segmentation of the line into characters.
- **In general, we will search for the value of the latent variable that allows us to get an answer (Y) of smallest energy.**

Probabilistic Latent Variable Models

- Marginalizing over latent variables instead of minimizing.

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

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

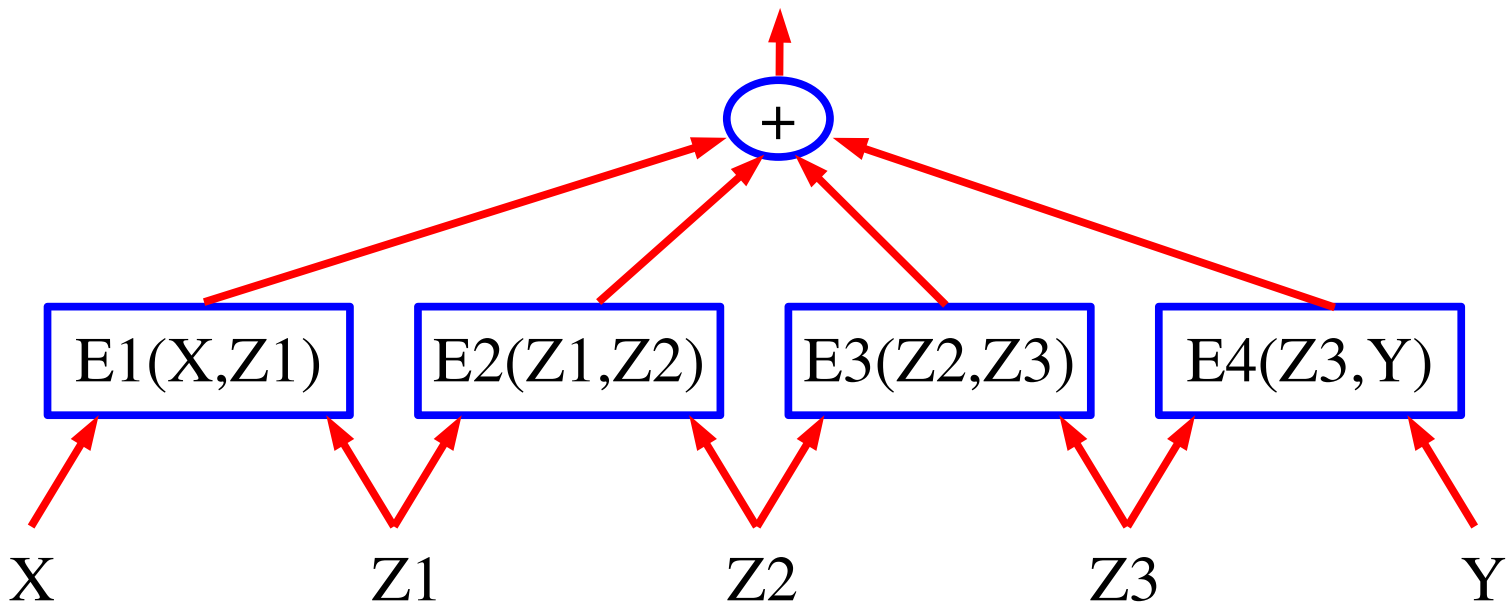
- Equivalent to traditional energy-based inference with a redefined energy function:

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

- Reduces to traditional minimization when Beta->infinity

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).

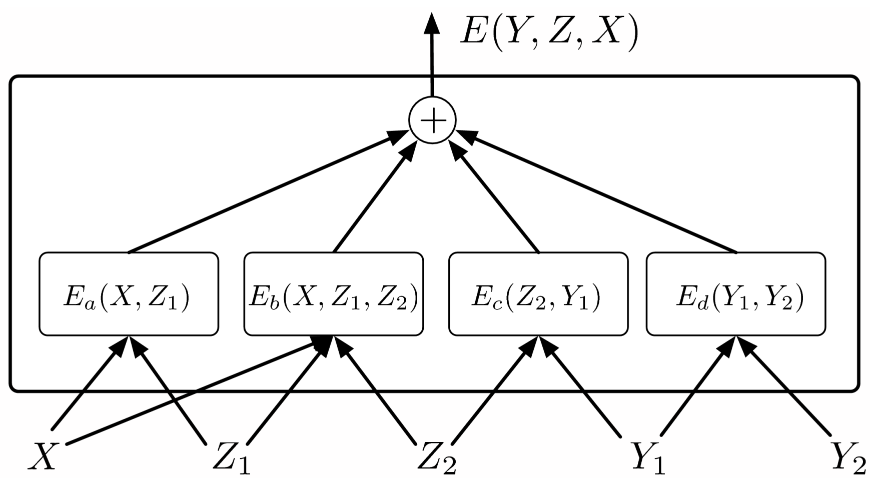


Efficient Inference: Energy-Based Factor Graphs

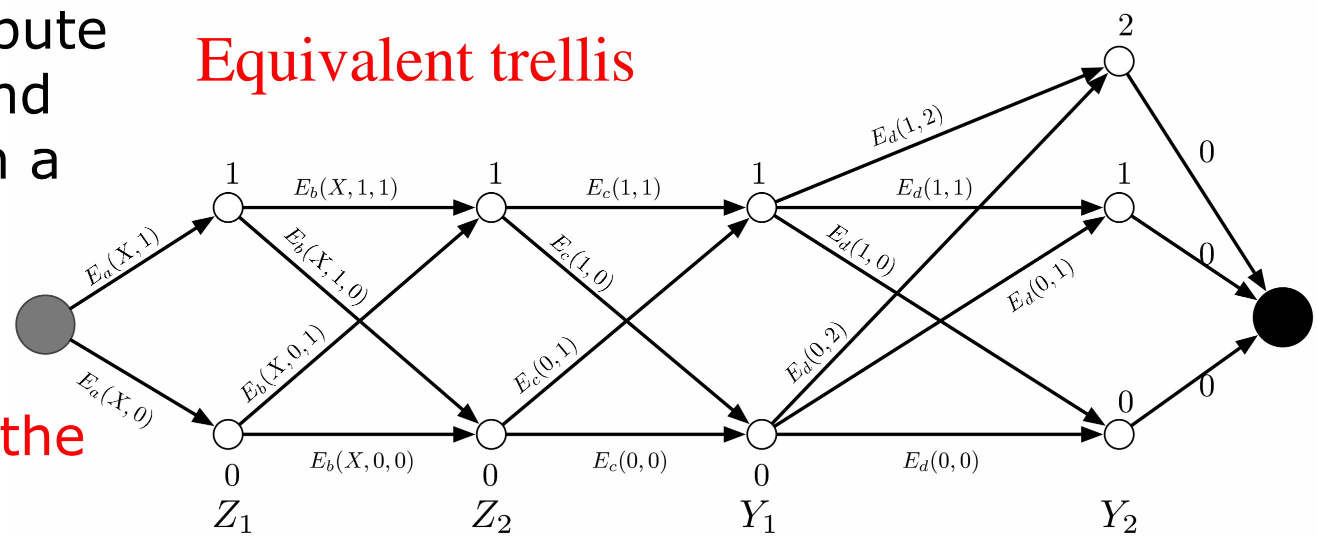
- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3=24 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 6.
  - Hence, we can precompute the 16 factor values, and put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

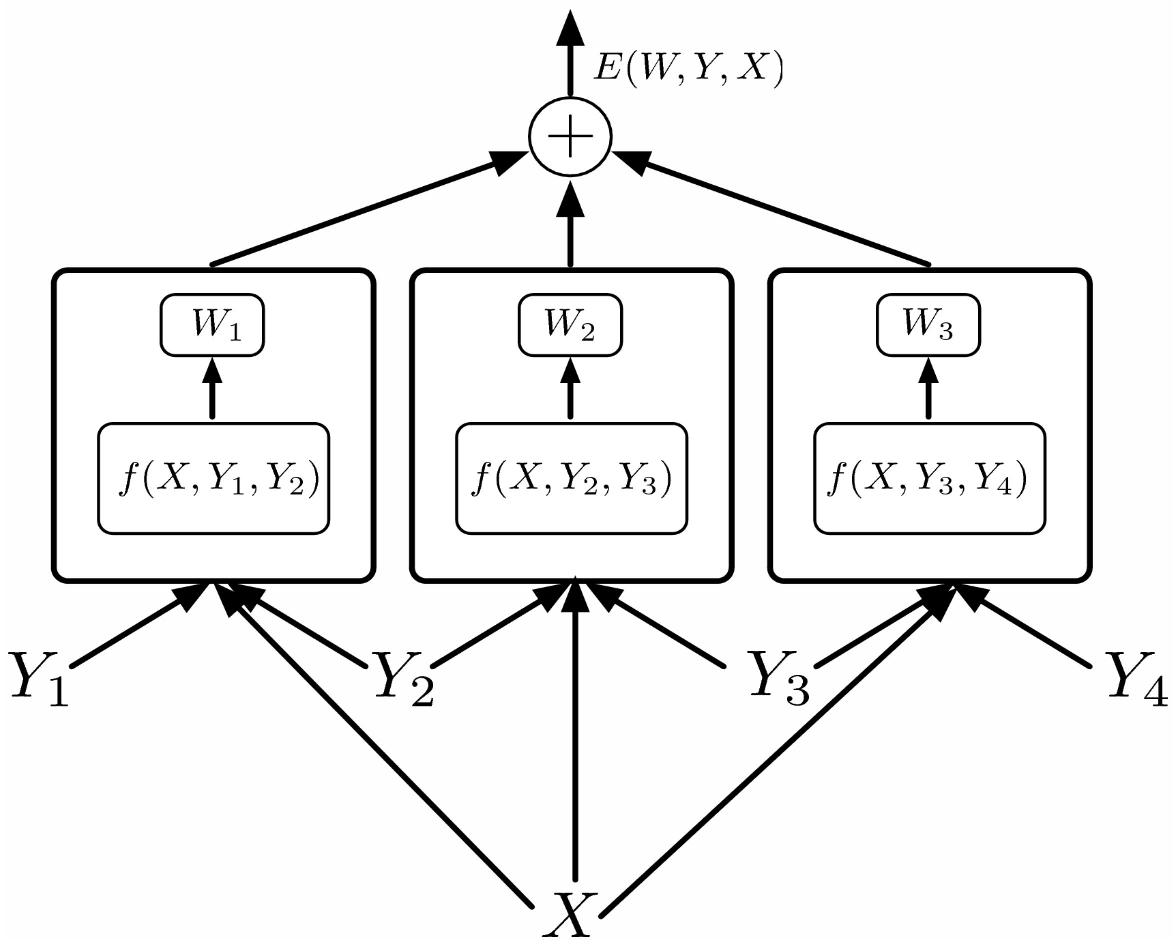


Equivalent trellis



Example : The Conditional Random Field Architecture

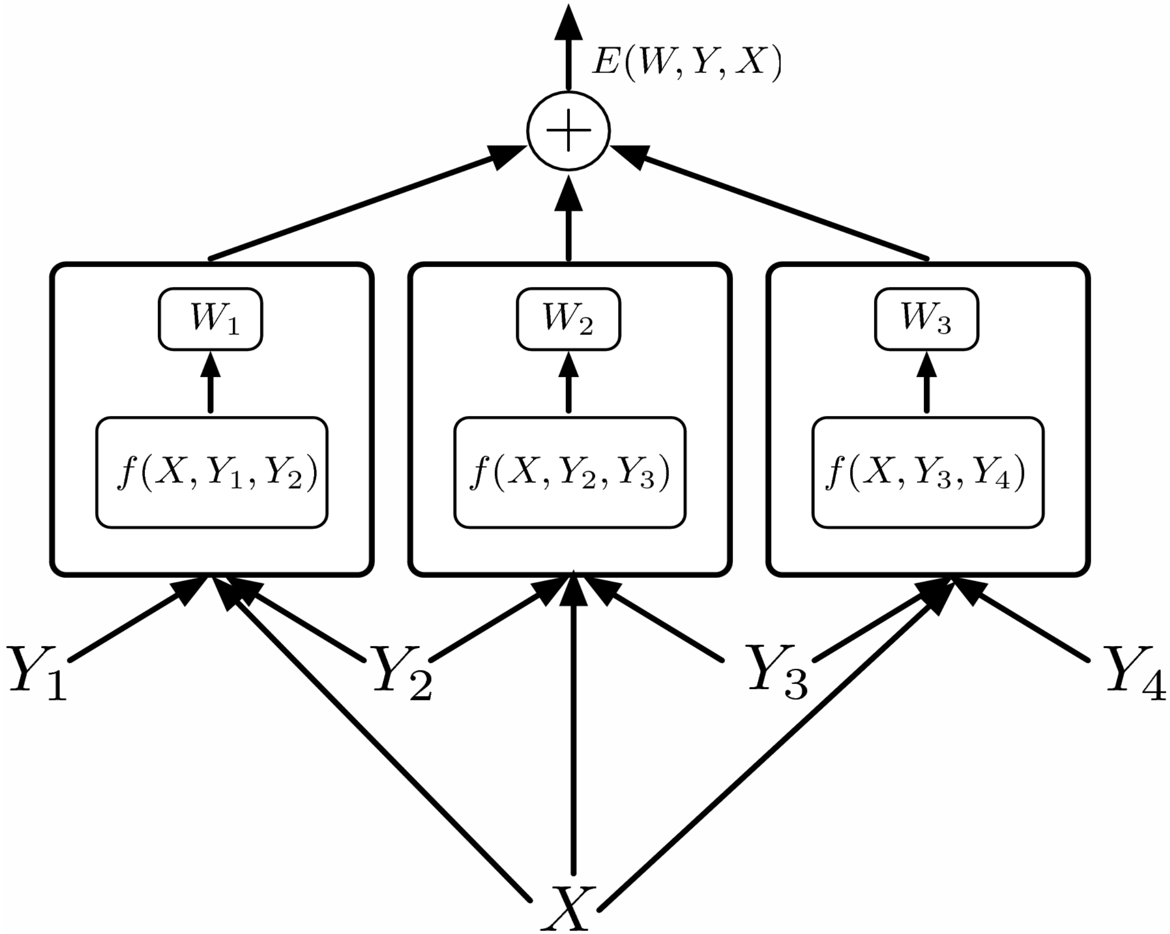
- **A CRF is an energy-based factor graph in which:**
  - the factors are **linear in the parameters** **(shallow factors)**
  - The factors take neighboring output variables as inputs
  - The factors are often all identical



Example : The Conditional Random Field Architecture

Applications:

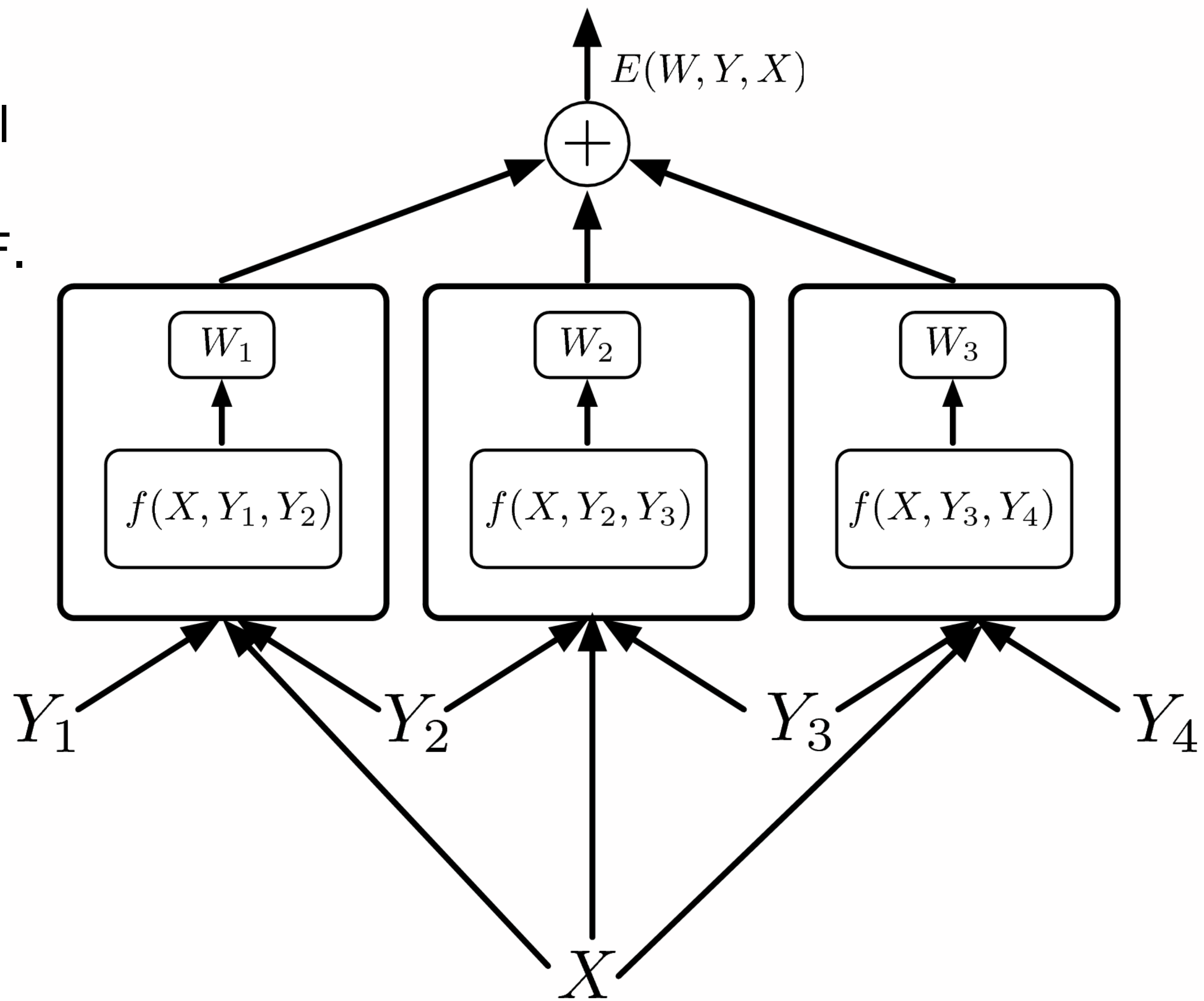
- X is a sentence, Y is a sequence of Parts of Speech Tags (there is one Yi for each possible group of words).
- X is an image, Y is a set of labels for each window in the image (vegetation, building, sky....).



Shallow Factors / Deep Graph

- **Linearly Parameterized Factors (shallow factors)**

- **with the NLL Loss :**
  - Lafferty's Conditional Random Field
  - Kumar&Hebert's DRF.
- **with Hinge Loss:**
  - Taskar's Max Margin Markov Nets
- **with Perceptron Loss**
  - Collins's sequence labeling model
- **With Log Loss:**
  - Altun/Hofmann sequence labeling model

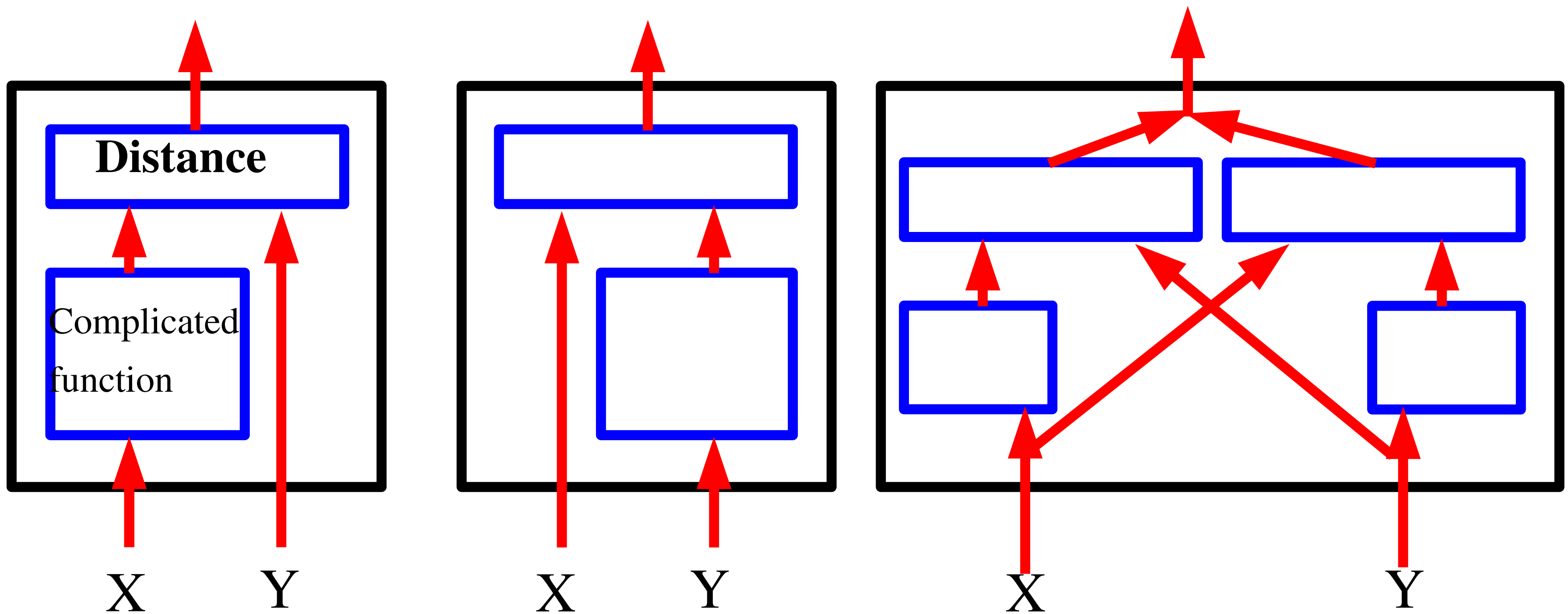


Energy-Based Belief Prop

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

Feed-Forward, Causal, and Bi-directional Models

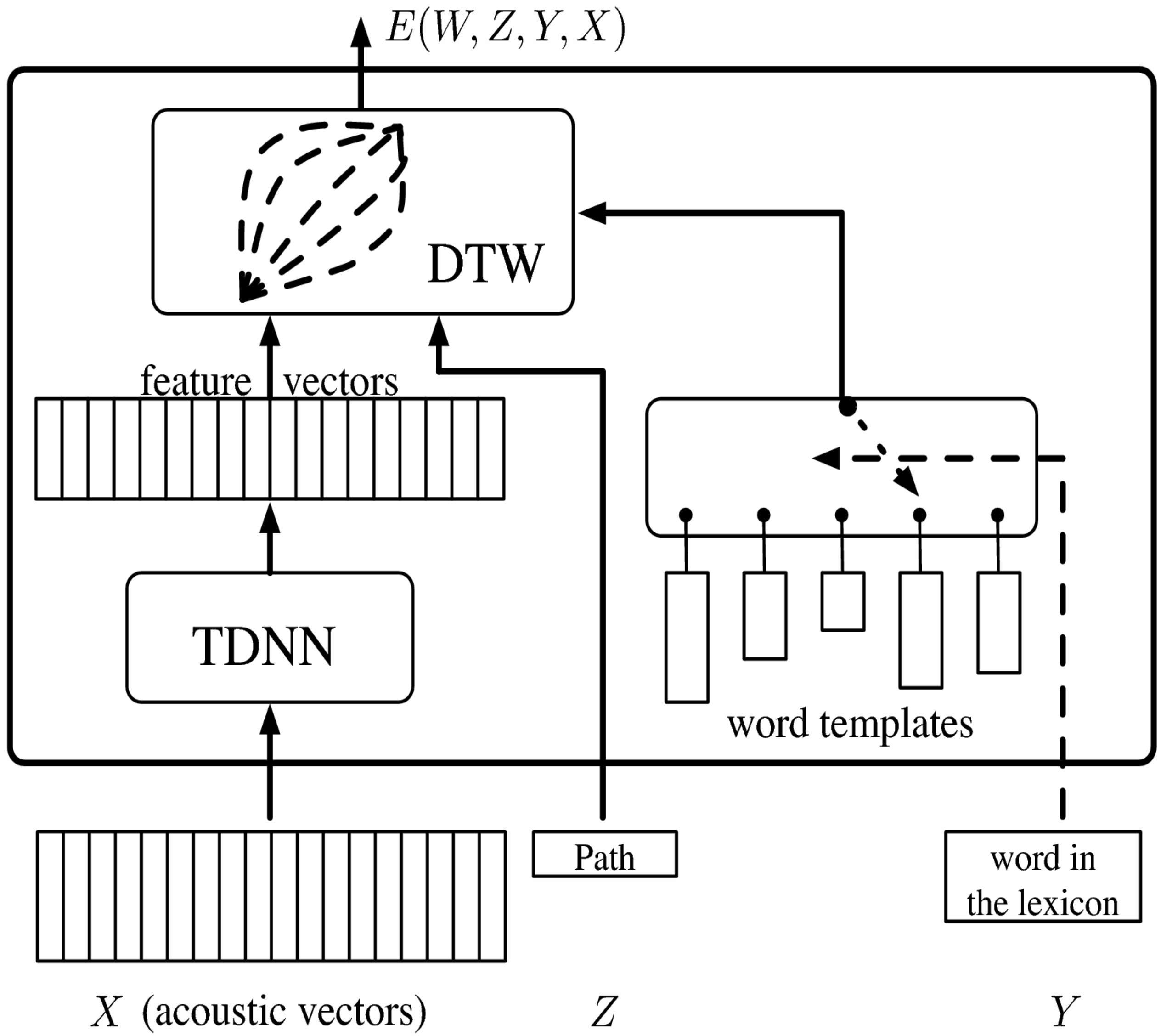
- **EBFG are all "undirected", but the architecture determines the complexity of the inference in certain directions**



- **Feed-Forward**
  - Predicting Y from X is easy
  - Predicting X from Y is hard
- **"Causal"**
  - Predicting Y from X is hard
  - Predicting X from Y is easy
- **Bi-directional**
  - X->Y and Y->X are both hard if the two factors don't agree.
  - They are both easy if the factors agree

Deep Factors / Deep Graph: ASR with TDNN/DTW

- **Trainable Automatic Speech Recognition system with convolutional nets (TDNN) and dynamic time warping (DTW)**
- **Training the feature extractor as part of the whole process.**
- **with the LVQ2 Loss :**
  - Driancourt and Bottou's speech recognizer (1991)
- **with NLL:**
  - Bengio's speech recognizer (1992)
  - Haffner's speech recognizer (1993)



Deep Factors / Deep Graph: ASR with TDNN/HMM

- **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)

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**