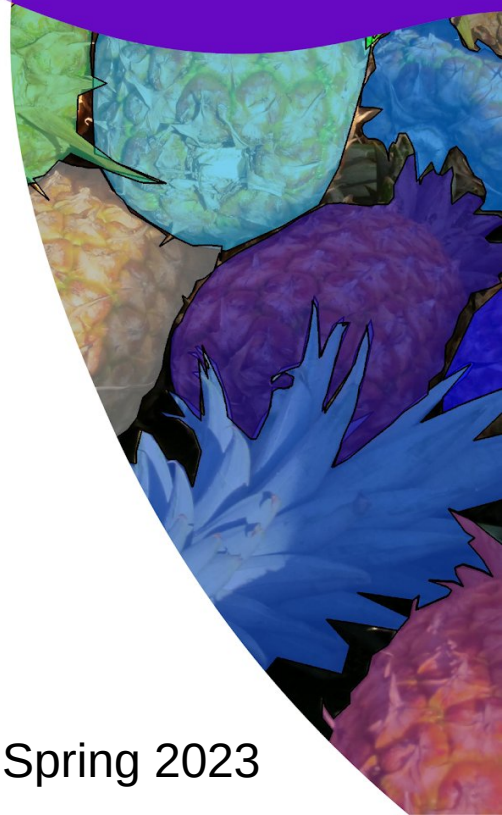
NEW YORK UNIVERSITY

Energy-Based Models Part 1

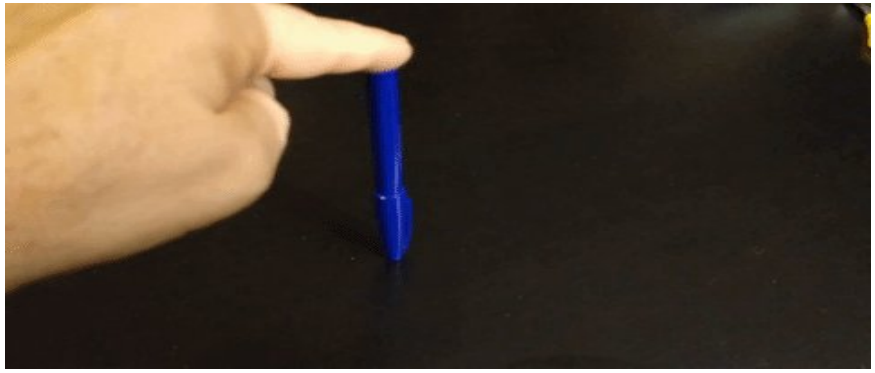
Alfredo Canziani, Yann LeCun
NYU - Courant Institute & Center for Data Science

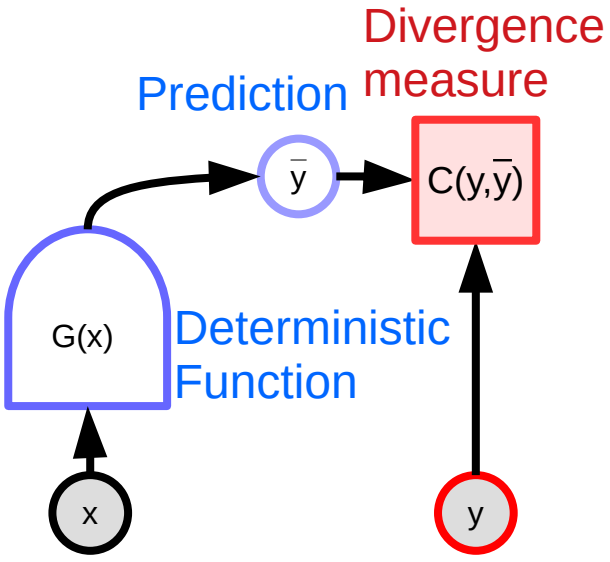


Deep Learning, NYU Spring 2023

The world is stochastic

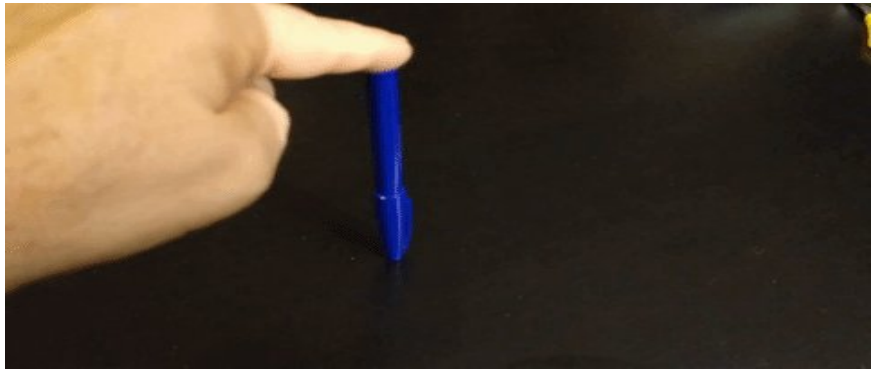
- Training a system to make a single prediction makes it predict the average of all plausible predictions
- **Blurry predictions!**





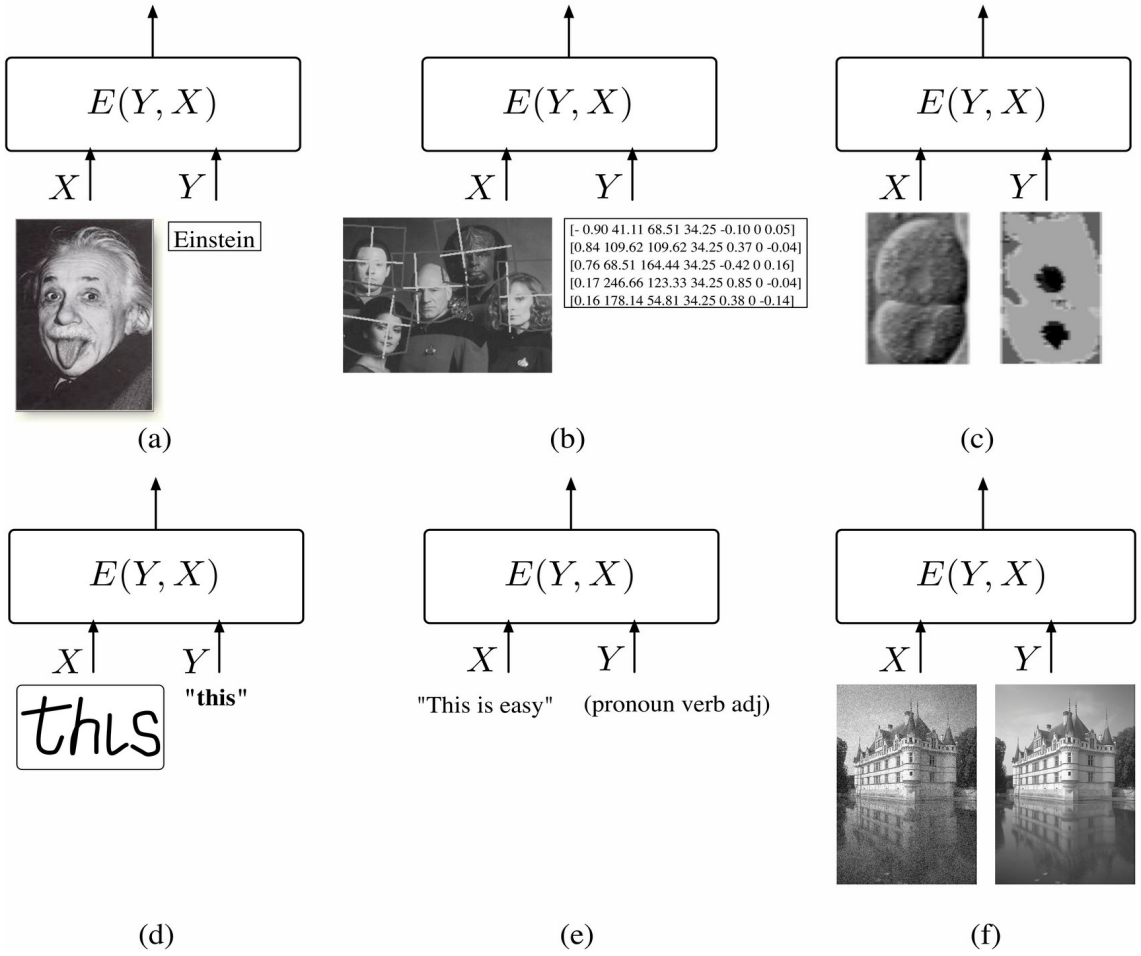
The world is unpredictable. Output must be multimodal.

- Training a system to make a single prediction makes it predict the average of all plausible predictions
- **Blurry predictions!**



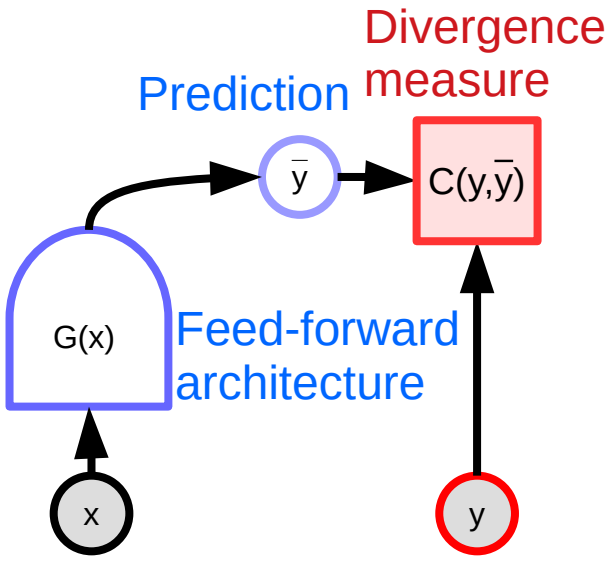
When inference is hard: search for the correct output.

- **Cases where inference is hard:**
  - Output is a high-dimensional object with structure:
    - Sequence, image, video,…
  - Output has compositional structure:
    - Text, action sequence,…
  - Output results from a search, or a chain of reasoning...
    - ...That can be reduced to an optimization problem

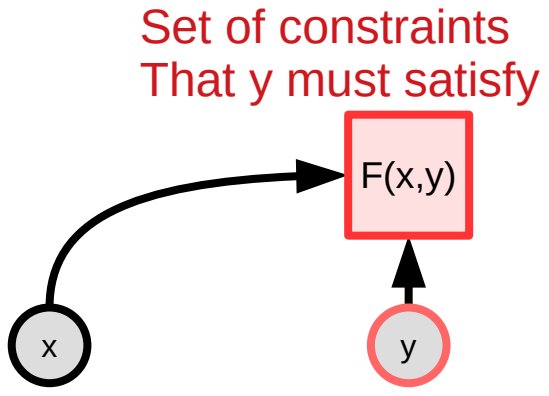


Energy-Based Models

- **Feed-forward nets use a finite number of steps to produce a single output.**
- **What if…**
  - The problem requires a complex computation to produce its output? (complex inference)
  - There are multiple possible outputs for a single input? (e.g. predicting future video frames)



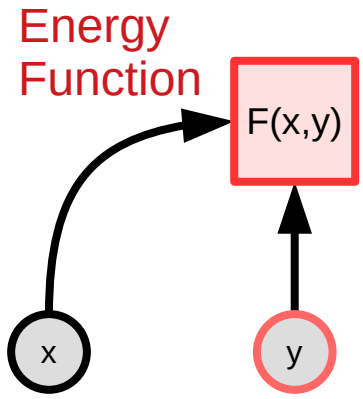
- **Inference through constraint satisfaction**
  - Finding an output that satisfies constraints: e.g a linguistically correct translation or speech transcription.
  - Maximum likelihood inference in graphical models



Energy-Based Models (EBM)

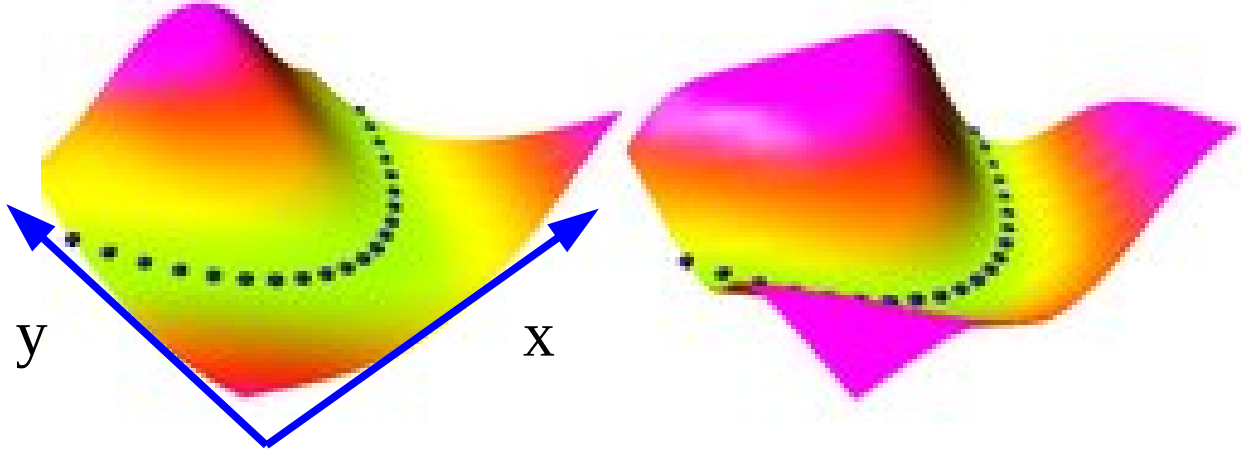
- **Energy function F(x,y) scalar-valued.**
  - Takes low values when y is compatible with x and higher values when y is less compatible with x
- **Inference: find values of y that make F(x,y) small.**
  - There may be multiple solutions

$$\check{y} = \text{argmin}_y F(x, y)$$



- **Note: the energy is used for inference, not for learning**

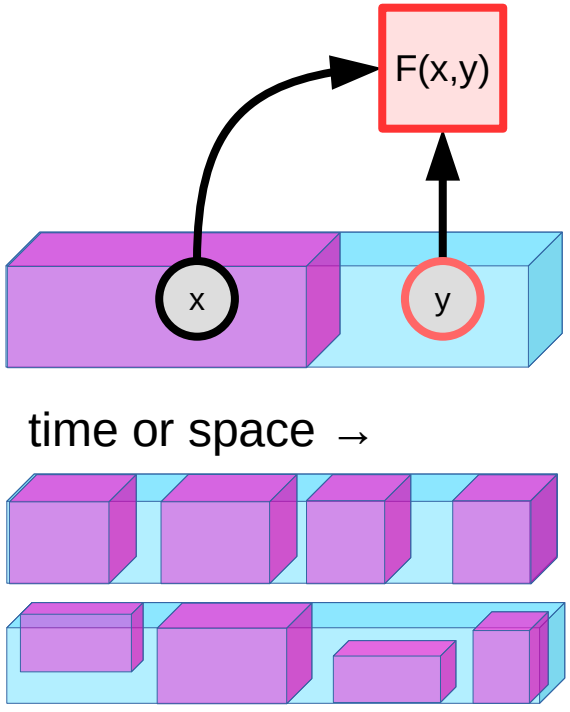
- **Example**
  - Blue dots are data points

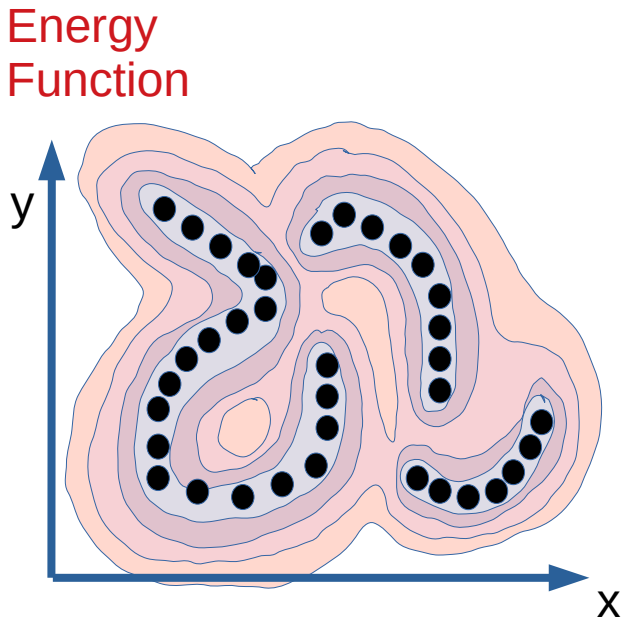


Energy-Based Models: Implicit function

- **Gives low energy for compatible pairs of x and y**
- **Gives higher energy for incompatible pairs**

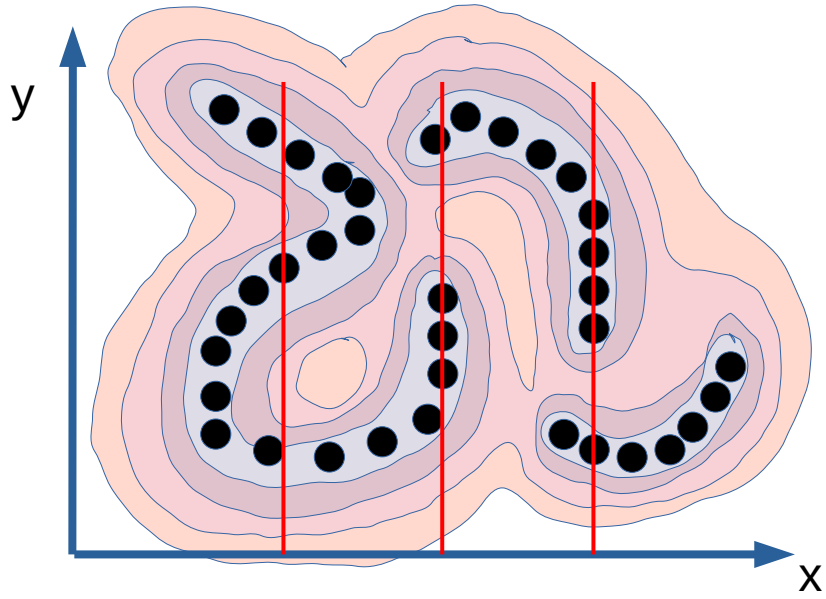
$$\check{y} = \text{argmin}_y F(x, y)$$



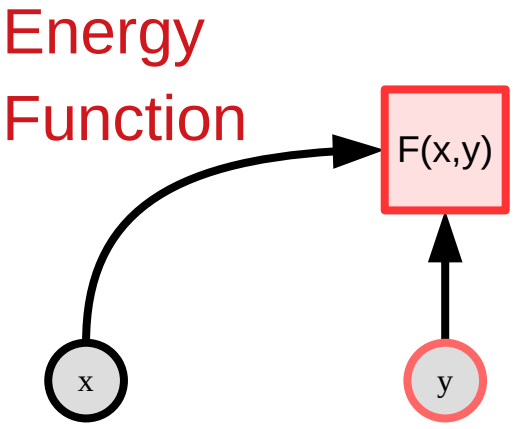


Energy-Based Model: implicit function

- **Energy function that captures the x,y dependencies:**
  - Low energy near the data points. Higher energy everywhere else.
  - If y is continuous, F should be smooth and differentiable, so we can use gradient-based inference algorithms.

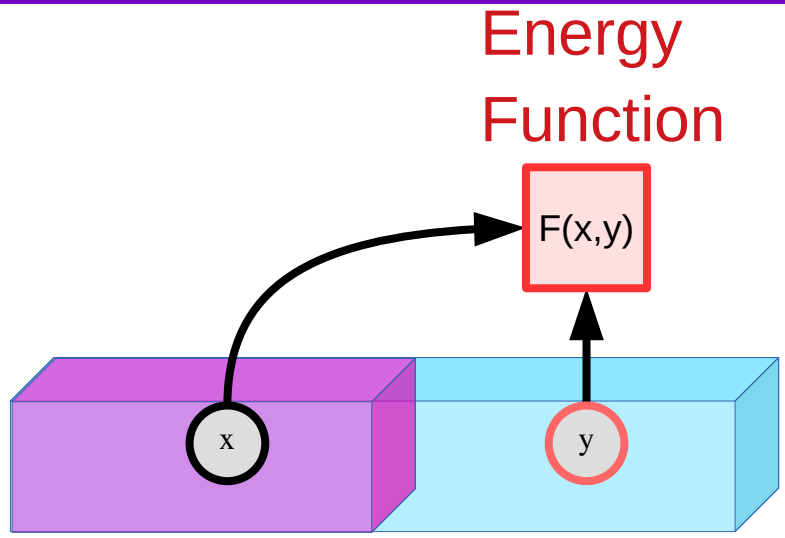


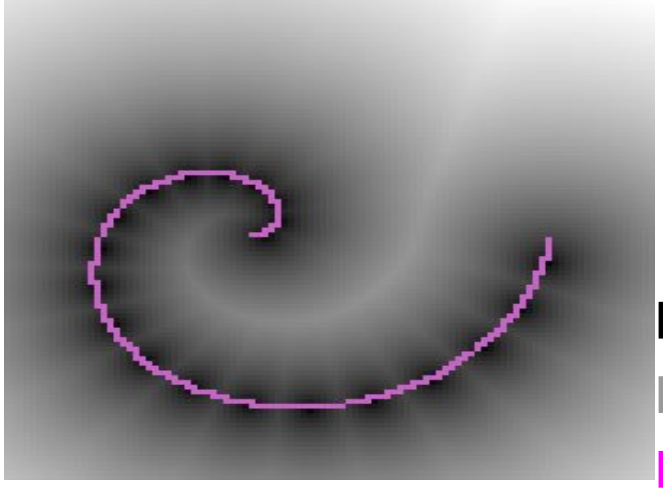
$$\check{y} = \text{argmin}_y F(x, y)$$



Energy-Based Model: unconditional version

- **Conditional EBM: F(x,y)**
- **Unconditional EBM: F(y)**
  - measures the compatibility between the components of y
  - If we don't know in advance which part of y is known and which part is unknown

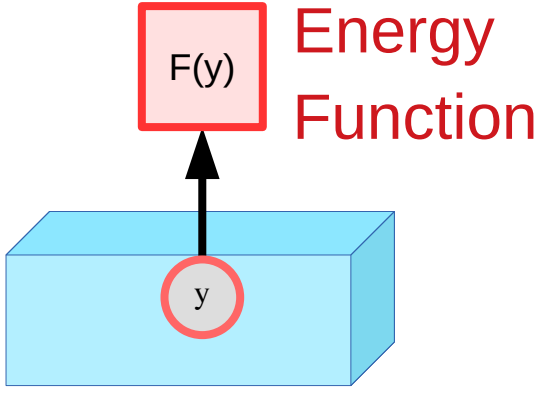




Dark = low energy (good)

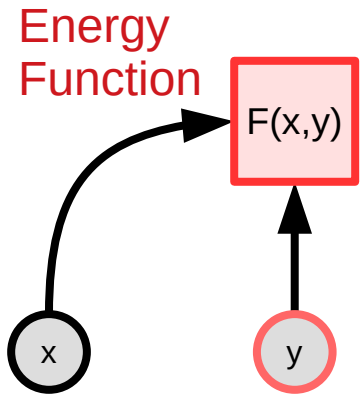
Bright = high energy (bad)

Purple = data manifold



Energy-Based Models vs Probabilistic Models

- **Probabilistic models are a special case of EBM**
  - Energies are like un-normalized negative log probabilities
- **Why use EBM instead of probabilistic models?**
  - EBM gives more flexibility in the choice of the scoring function.
  - More flexibility in the choice of objective function for learning
- **From energy to probability: Gibbs-Boltzmann distribution**
  - Beta is a positive constant

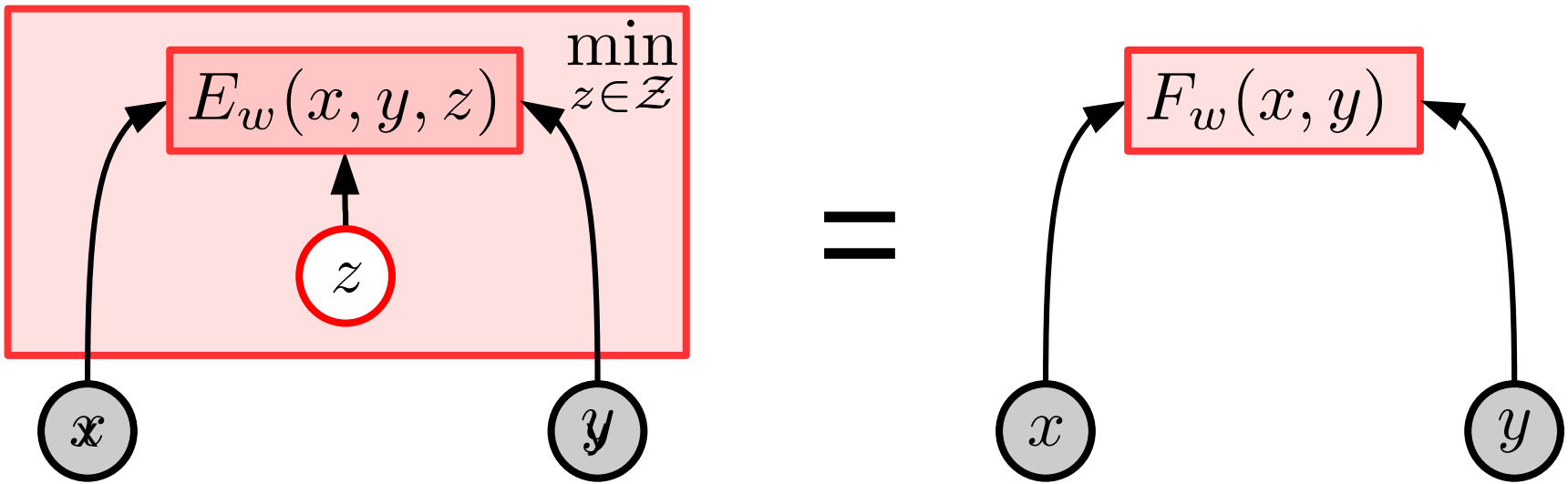


$$P(y|x) = \frac{e^{-\beta F(x,y)}}{\int_{y'} e^{-\beta F(x,y')}}$$

Latent-Variable EBM

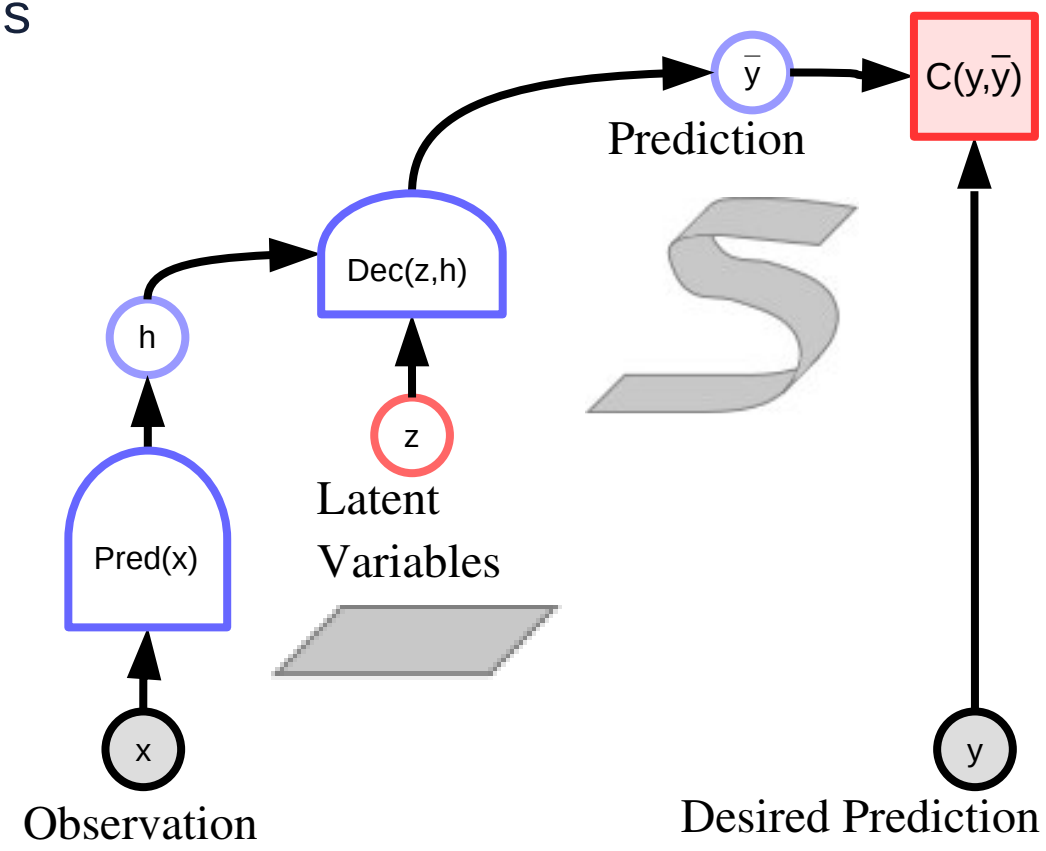
- **Latent variable z:**
  - Captures the information in y that is not available in x
  - Computed by minimization

$$\check{z} = \text{argmin}_{z \in \mathcal{Z}} E_w(x, y, z) \qquad F_w(x, y) = E_w(x, y, \check{z})$$



Latent-Variable Generative EBM Architecture

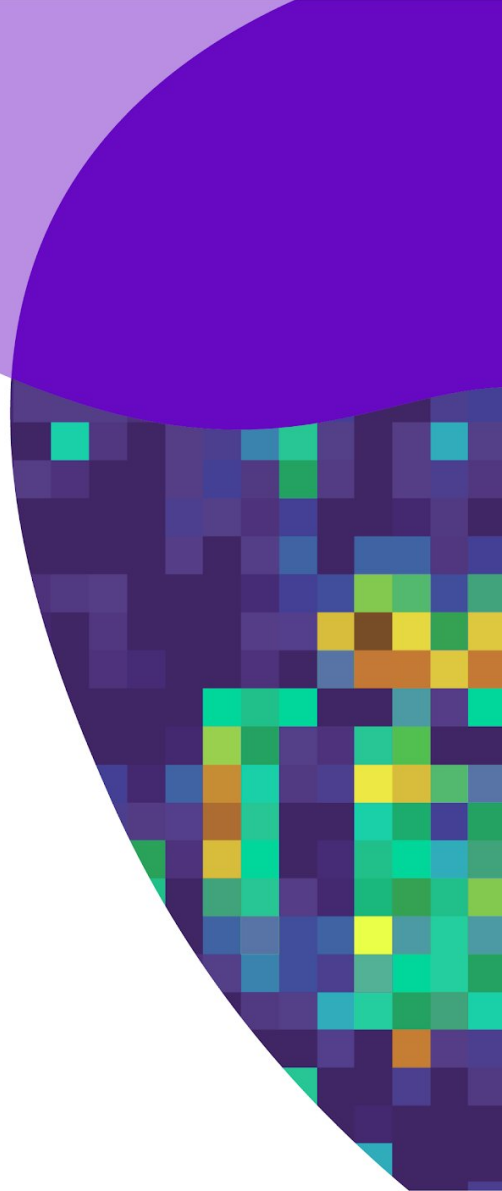
- **Latent variables:**
  - parameterize the set of predictions
- **Ideally, the latent variable represents independent explanatory factors of variation of the prediction.**
- **The information capacity of the latent variable must be minimized.**
  - Otherwise all the information for the prediction will go into it.



FACEBOOK AI

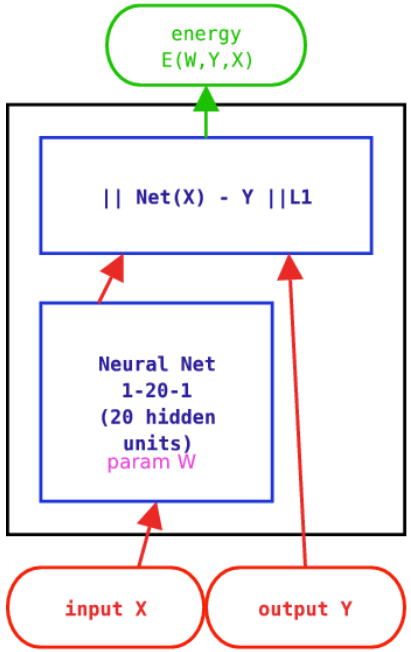
EBM Training

1. Contrastive methods
2. Regularized & Architectural methods

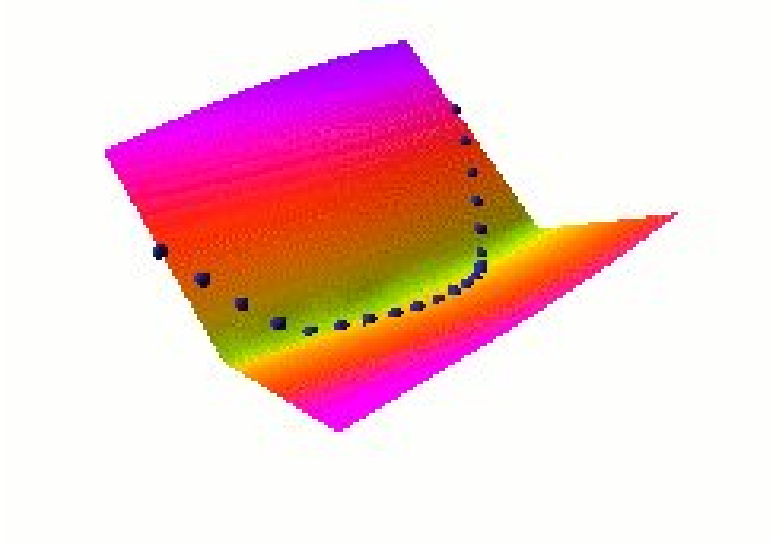


Regularized and Architectural EBM Training

- **With some architecture, simply pushing down on the energy of data points will make the energy function take the right shape.**

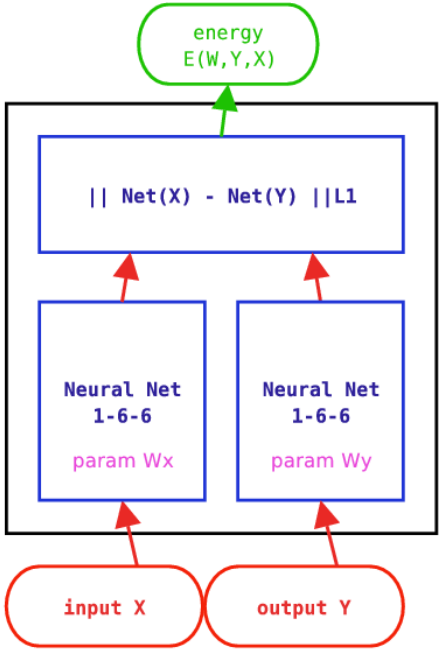


(a)

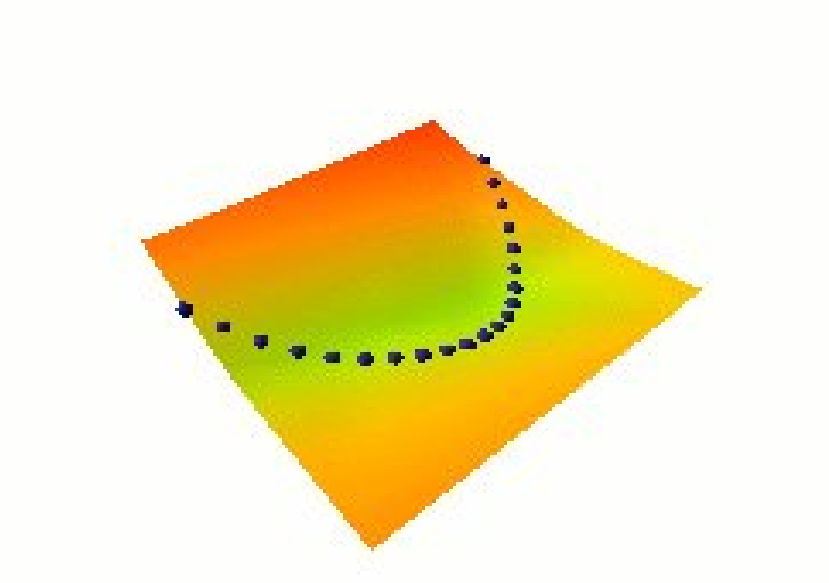


Architectures that can collapse!

- **With architectures like joint embedding, pushing down on the training sample energy can make the energy landscape flat**
  - The networks ignores the input and produce identical constant outputs



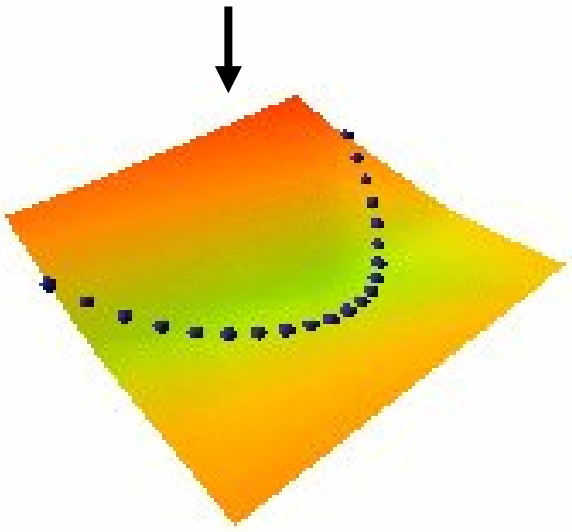
(b)



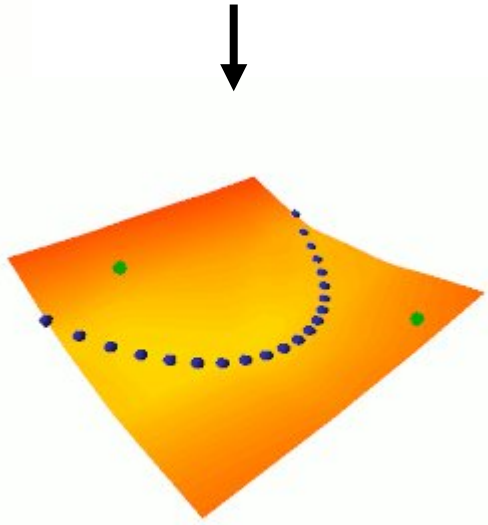
Shaping the energy surface / preventing collapse

- **A flexible energy surface can take any shape.**
- **We need a loss function that shapes the energy surface so that:**
  - Data points have low energies
  - Points outside the regions of high data density have higher energies.

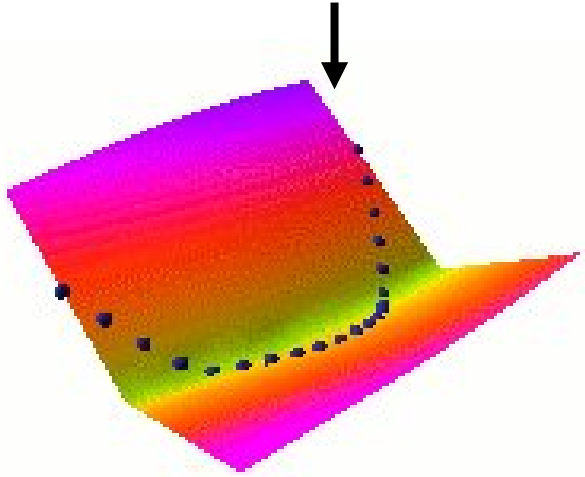
Collapse!



Contrastive Method

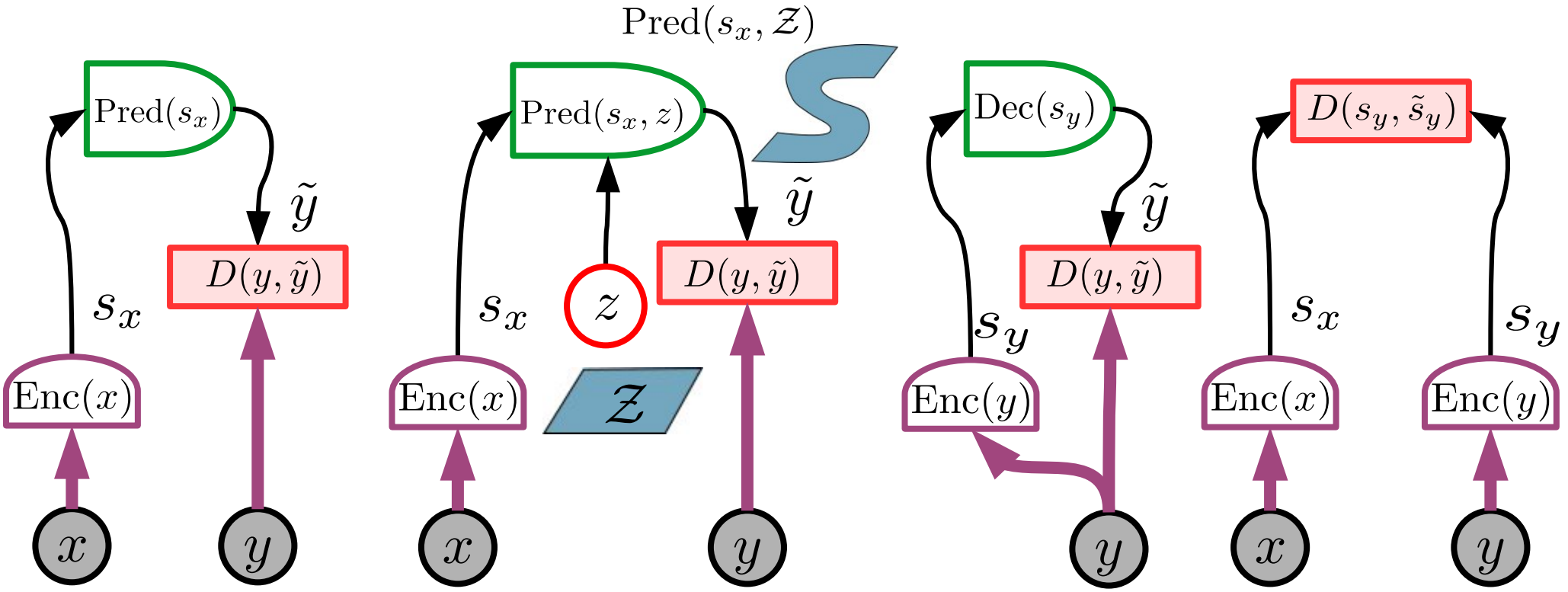


Regularized Methods



EBM Architectures

- **Some architectures can lead to a collapse of the energy surface**



a) Prediction / regression
NO COLLAPSE

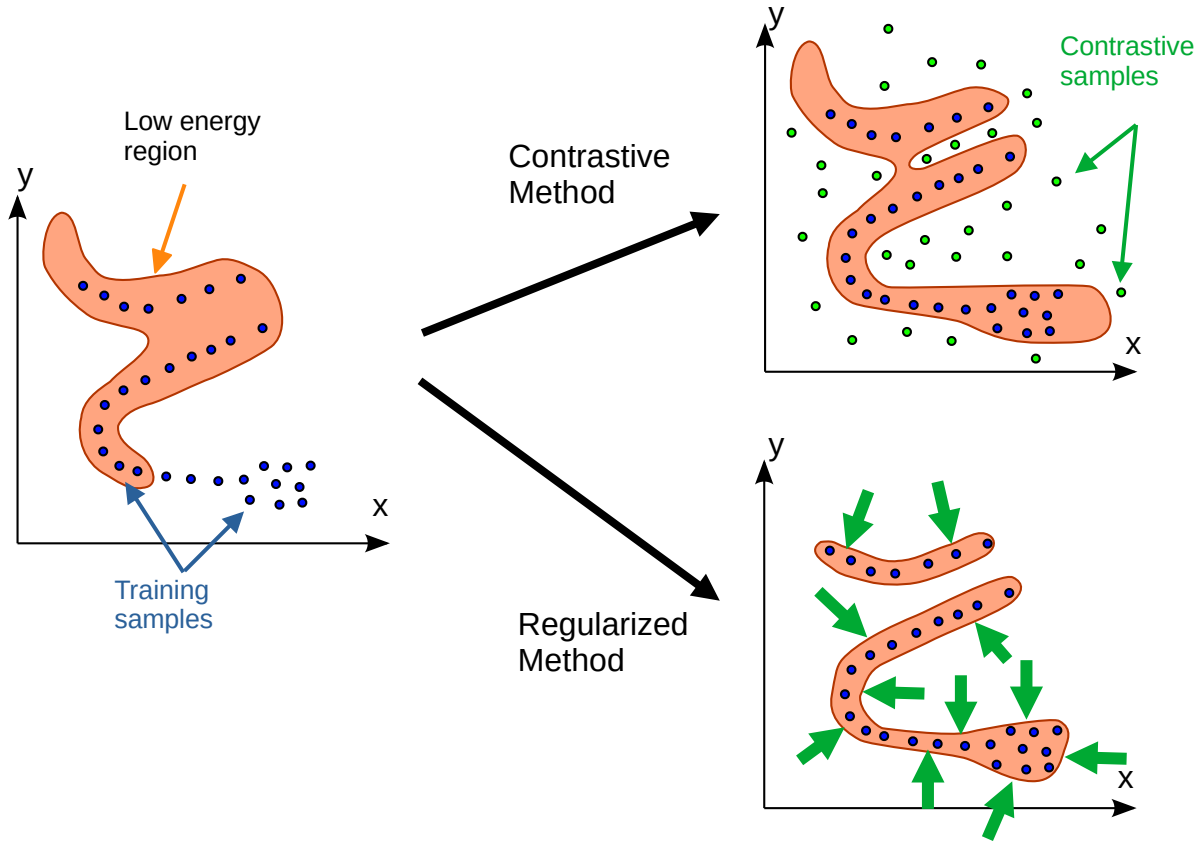
b) Generative latent-variable Architecture
CAN COLLAPSE

c) Auto-Encoder
CAN COLLAPSE

d) Joint Embedding Architecture
CAN COLLAPSE

EBM Training: two categories of methods

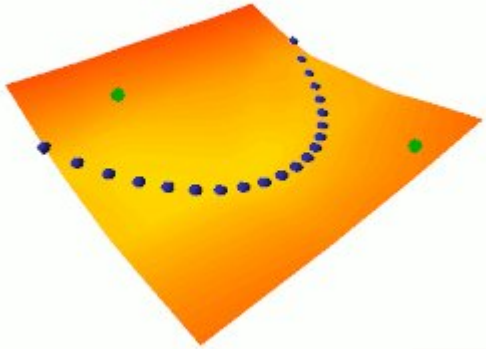
- **Contrastive methods**
  - Push down on energy of training samples
  - Pull up on energy of suitably-generated contrastive samples
  - Scales very badly with dimension
- **Regularized Methods**
  - Regularizer minimizes the volume of space that can take low energy



Contrastive methods vs Regularized Methods

- **Contrastive methods: works with any architecture**
  - Expensive in high dimension
  - Example of contrastive loss: pick a ŷ to push up.

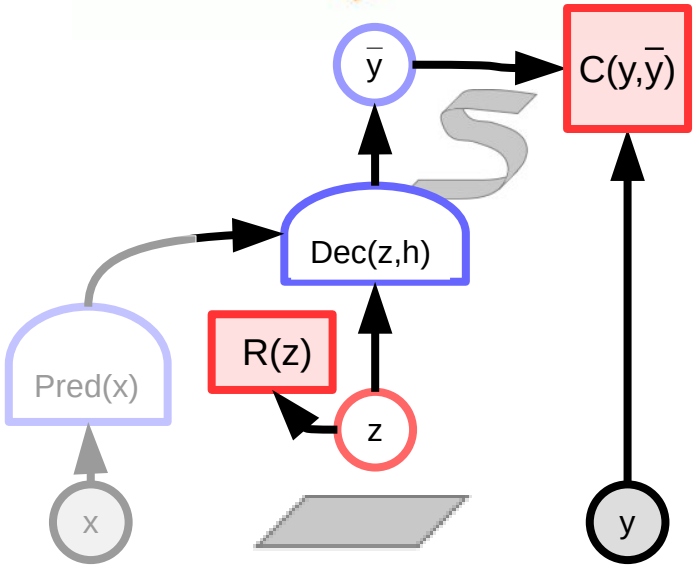
$$\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+$$



- **Regularized methods: minimizing the volume of low-energy space**
  - E.g. by limiting the capacity of the latent

$$\mathcal{L}(x, y, w) = F_w(x, y)$$

$$F_w(x, y) = \min_z [C(\mathrm{Dec}(\mathrm{Pred}(x), z), y) + R(z)]$$

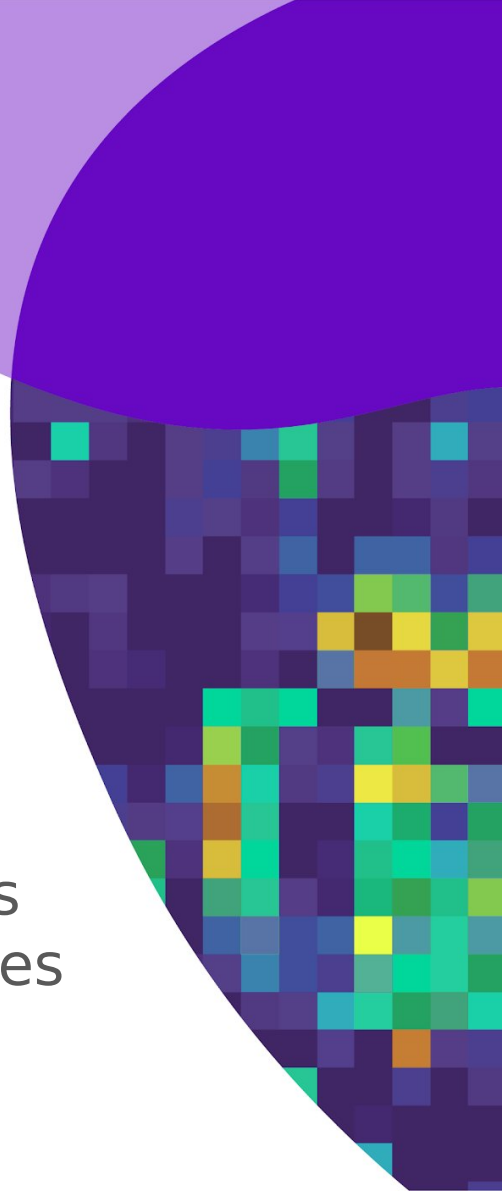


Contrastive Methods vs Regularized/Architectural Methods

- **Contrastive: [they all are different ways to pick which points to push up]**
  - C1: push down of the energy of data points, push up everywhere else: Max likelihood (needs tractable partition function or variational approximation)
  - C2: push down of the energy of data points, push up on chosen locations: max likelihood with MC/MMC/HMC, Contrastive divergence, Metric learning/Siamese nets, Ratio Matching, Noise Contrastive Estimation, Min Probability Flow, adversarial generator/GANs
  - C3: train a function that maps points off the data manifold to points on the data manifold: denoising auto-encoder, masked auto-encoder (e.g. BERT)
- **Regularized/Architectural: [Different ways to limit the information capacity of the latent representation]**
  - A1: build the machine so that the volume of low energy space is bounded: PCA, K-means, Gaussian Mixture Model, Square ICA, normalizing flows…
  - A2: use a regularization term that measures the volume of space that has low energy: Sparse coding, sparse auto-encoder, LISTA, Variational Auto-Encoders, discretization/VQ/VQVAE.
  - A3: F(x,y) = C(y, G(x,y)), make G(x,y) as "constant" as possible with respect to y: Contracting auto-encoder, saturating auto-encoder
  - A4: minimize the gradient and maximize the curvature around data points: score matching

FACEBOOK AI

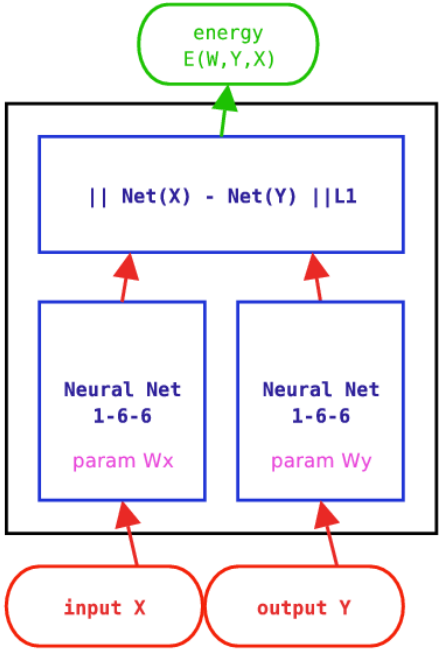
Contrastive Methods



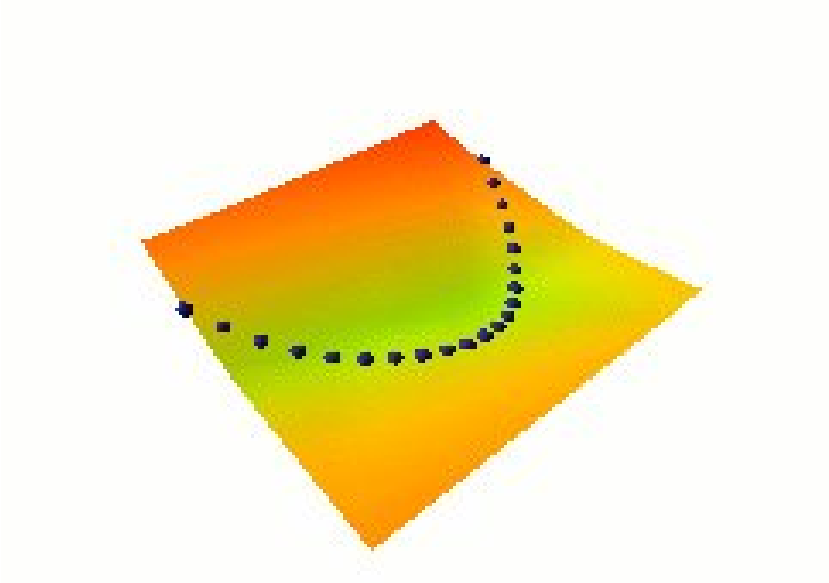
Push down on the energy of training samples
Pull up on the energy of “contrastive” samples

Architectures that can collapse!

- **With architectures like joint embedding, pushing down on the training sample energy can make the energy landscape flat**
  - The networks ignores the input and produce identical constant outputs



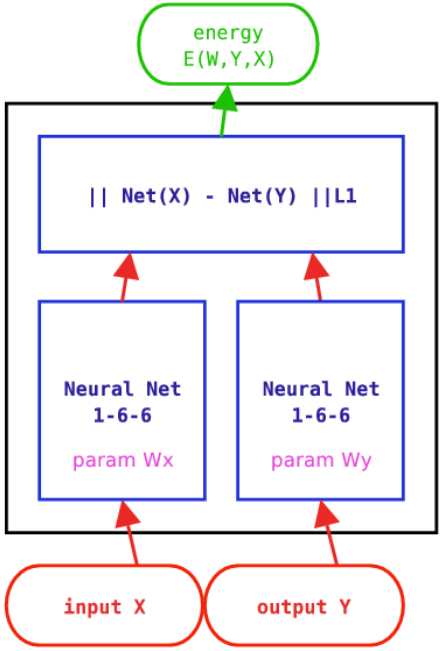
(b)



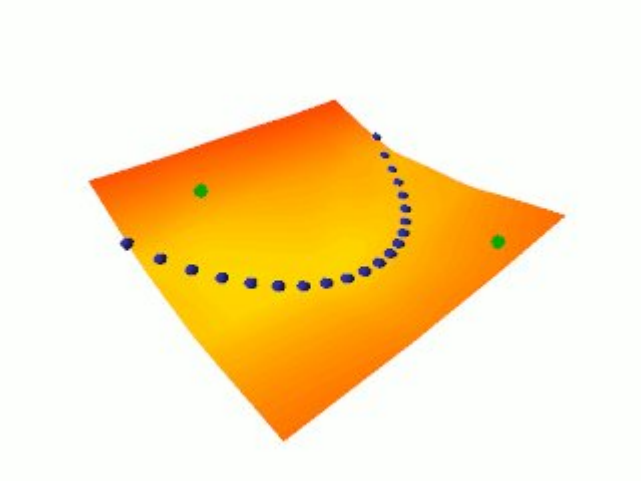
Contrastive Methods (to prevent collapse)

- **Push down on the energy of training samples**
- **Push up on the energy of other judiciously-chosen samples**

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



(b)

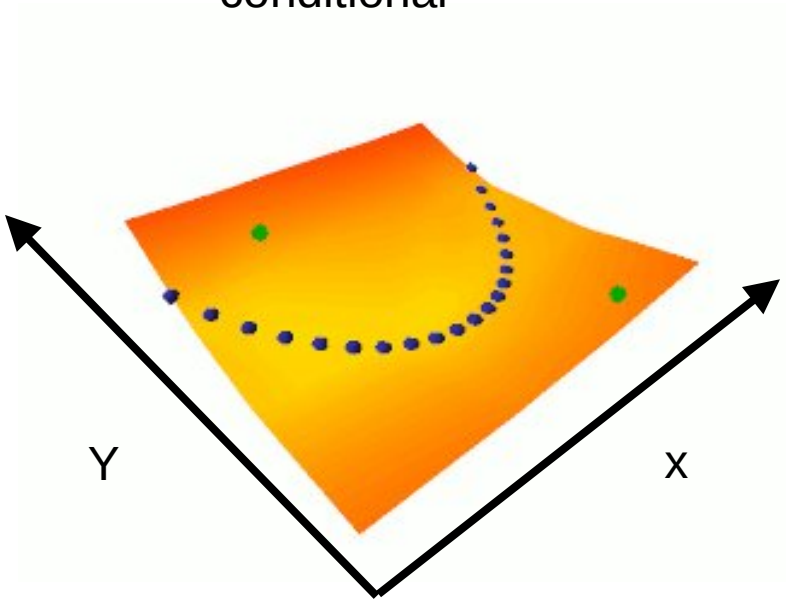


Contrastive energy-based learning

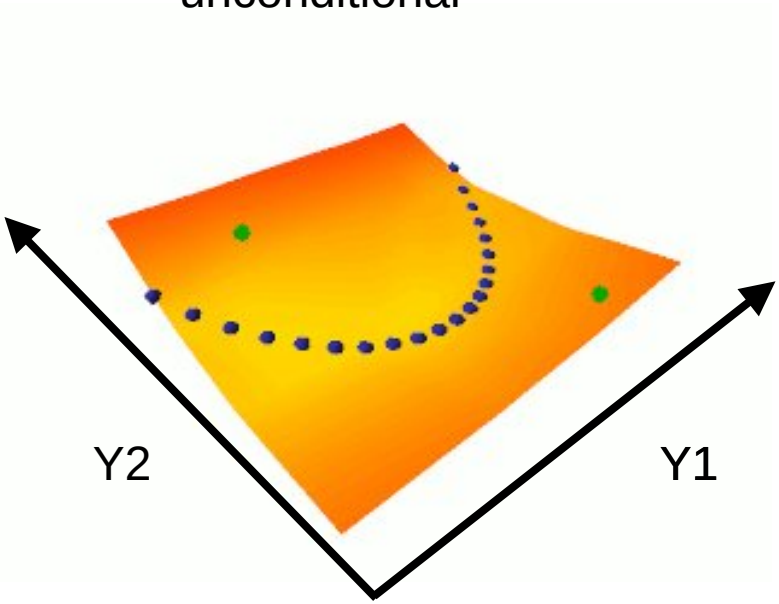
- **Push down on the energy of training samples**
- **Pull up on the energy of other "well-chosen" points**

$$\mathcal{L}(x_1 \dots x_{p^+}, y_1 \dots y_{p^+}, \hat{y}_1 \dots \hat{y}_{p^-}, w) = H\left(E(x_1, y_1), \dots E(x_{p^+}, y_{p^+}), E(x_1, \hat{y_1}), \dots E(x_{p^+}, \hat{y_{p^+}}), M(Y_{1\dots p^+}, \hat{Y}_{1\dots p^-})\right)$$

conditional

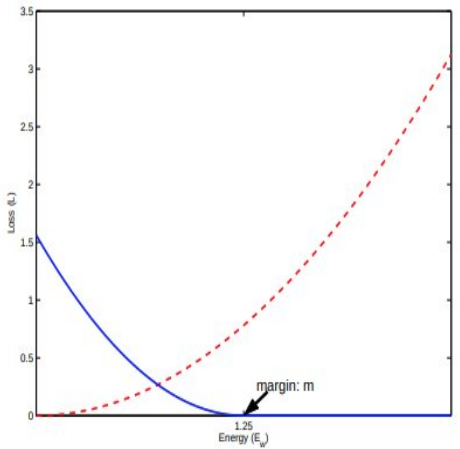


unconditional



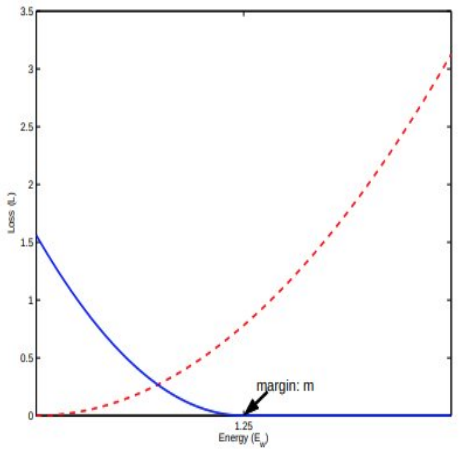
Contrastive losses: pairwise margin losses

- **Push down on data points, push up of other points**
  - well chosen contrastive points
  - General margin loss: $\mathcal{L}(x, y, \hat{y}, w) = H(F_w(x, y), F_w(x, \hat{y}), m(y, \hat{y}))$
    - Where H(F^+,F^-,m) is a strictly increasing function of F^+ and a strictly decreasing function of F^- , at least whenever F^- - F^+ < m.
  - Examples:
    - Simple [Bromley 1993]:
      $\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y)]^+ + [m(y, \hat{y}) - F_w(x, \hat{y})]^+$
    - Hinge pair loss [Altun 2003], Ranking loss [Weston 2010]:
      $\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+$
    - Square-Square: [Chopra CVPR 2005] [Hadsell CVPR 2006]:
      $\mathcal{L}(x, y, \hat{y}, w) = \left([F_w(x, y)]^+\right)^2 + \left([m(y, \hat{y}) - F_w(x, \hat{y})]^+\right)^2$



Contrastive Methods: many possible losses

- **Push down on data points, push up on other points**
  - well chosen contrastive points
  - General margin loss: $\mathcal{L}(x, y, \hat{y}, w) = H(F_w(x, y), F_w(x, \hat{y}), m(y, \hat{y}))$
    - Where H(F^+,F^-,m) is a strictly increasing function of F^+ and a strictly decreasing function of F^- , at least whenever F^- - F^+ < m.
  - Examples:
    - Simple [Bromley 1993]:
      $\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y)]^+ + [m(y, \hat{y}) - F_w(x, \hat{y})]^+$
    - Hinge pair loss [Altun 2003], Ranking loss [Weston 2010]:
      $\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+$
    - Square-Square: [Chopra CVPR 2005] [Hadsell CVPR 2006]:
      $\mathcal{L}(x, y, \hat{y}, w) = \left([F_w(x, y)]^+\right)^2 + \left([m(y, \hat{y}) - F_w(x, \hat{y})]^+\right)^2$



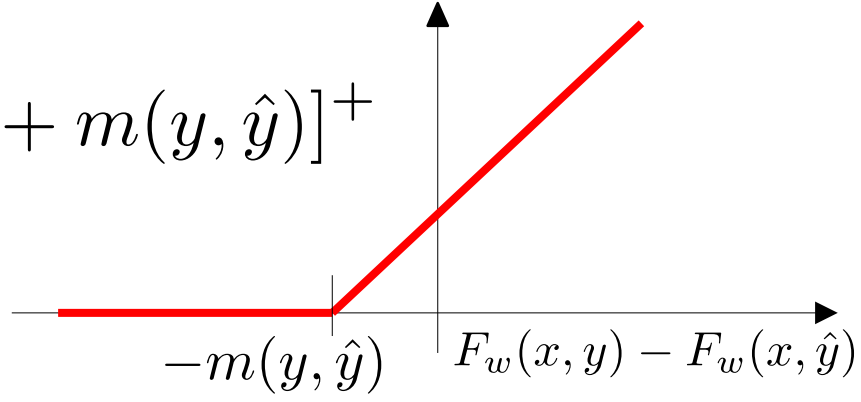
General margin loss

- **Considers all possible outputs (or a well-chosen subset)**

$$\mathcal{L}(x, y, w) = \sum_{\hat{y} \in \mathcal{Y}} H(F_w(x, y), F_w(x, \hat{y}), m(y, \hat{y}))$$

- **Hinge loss that makes F(x,y) lower than F(x,y') by a quantity (margin) that depends on the distance between y and y'**
- **Example:**

$$\mathcal{L}(x, y, w) = \sum_{\hat{y} \in \mathcal{Y}} [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+$$



Plenty of Contrastive Loss Functions to Choose From

Good and bad loss functions: good ones have non-zero margin

Loss	**Formula**	**Margin**
energy loss	$F(x,y)$	0
perceptron	$F(x,y) - \min_{\check{y}\in} F(x,\check{y})$	0
hinge	$\max(0, m + F(x,y) - F(x,\hat{y}))$	m
log	$\log\left(1 + e^{F(x,y)-F(x,\hat{y})}\right)$	∞
LVQ2	$\min(M, \max(0, F(x,y) - F(x,\hat{y}))$	0
MCE	$\left(1 + e^{-(F(x,y)-F(x,\hat{y}))}\right)^{-1}$	∞
square-square	$F(x,y)^2 - (\max(0, m(y,\hat{y}) - F(x,\hat{y})))^2$	m
square-exp	$F(x,y)^2 + \beta e^{-F(x,\hat{y})}$	∞
NLL/MMI	$F(x,y) + \frac{1}{\beta}\log\int_{\hat{y}\in} e^{-\beta F(x,\hat{y})}$	∞
MEE	$1 - e^{-\beta F(x,y)} / \int_{\hat{y}\in} e^{-\beta F(x,\hat{y})}$	∞

Loss function zoo for contrastive EBM training

	Method	Energy	$\hat{y}$ Generation	Loss
1	Max Likelihood	discrete y	exhaustive	$F_w(x,y) + \log \sum_{y' \in \mathcal{Y}} \exp(-F_w(x,y'))$
2	Max Likelihood	tractable	exhaustive	$F_w(x,y) + \log \int_{y' \in \mathcal{Y}} \exp(-F_w(x,y'))$
3	Max likelihood	any	MC or MCMC	$F_w(x,y) - F_w(x,\hat{y})$
4	Contr. Divergence	any	trunc'd MCMC	$F_w(x,y) - F_w(x,\hat{y})$
5	Pairwise Hinge	any	most offending	$[F_w(x,y) - F_w(x,\hat{y}) + m(y,\hat{y})]^+$
6	Min-Hinge	positive	most offending	$F_w(x,y) + [m(y,\hat{y}) - F_w(x,\hat{y})]^+$
6	Square-Hinge	divergence	most offending	$F_w(x,y)^2 + \left([m(y,\hat{y}) - F_w(x,\hat{y})]^+\right)^2$
7	Square-Exp	any	most offending	$F_w(x,y)^2 + \exp(-\beta F_w(x,\hat{y}))$
8	Logistic	any	most offending	$\log(1 + \exp(F_w(x,y) - F_w(x,\hat{y}))$
9	GAN	any	$\hat{y} = g_u(z)$	$H(F_w(x,y), F_w(x,\hat{y}), m(y,\hat{y}))$
10	Denoising AE	$D(y, g_w(y))$	$\hat{y} = N(y)$	$D(y, g_w(\hat{y})$

Contrastive Methods: group losses

- **Push down on a group of data points, push up on a group of contrastive points**
  - General group loss on p^+ data points and p^- contrastive points:

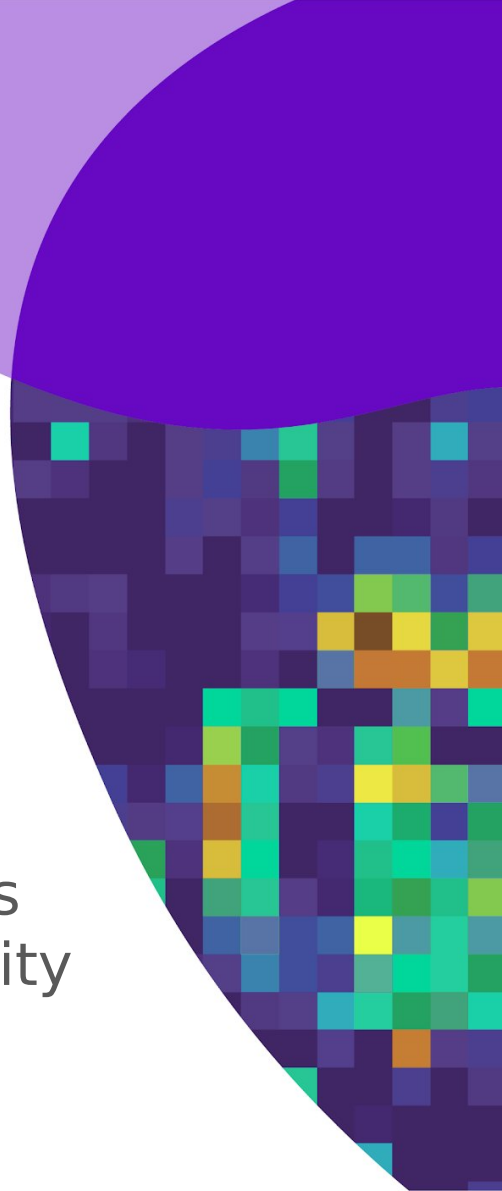
$$\mathcal{L}(x_1 \dots x_{p^+}, y_1 \dots y_{p^+}, \hat{y}_1 \dots \hat{y}_{p^-}, w) = H\left(F(x_1, y_1), \dots F(x_{p^+}, y_{p^+}), F(x_1, \hat{y_1}), \dots F(x_{p^+}, \hat{y_{p^+}}), M(Y_{1\dots p^+}, \hat{Y}_{1\dots p^-})\right)$$

    - Where H must be an increasing fn of the data energies and decreasing fn of the contrastive point energies within the margin.
    - M is a margin matrix for all pairs of y and ŷ in the group.
  - Example: Neighborhood Component Analysis, Noise Contrastive Estimation, InfoNCE (implicit infinite margin) [Goldberger 2005] [Gutmann 2010]... [Misra 2019] [Chen 2020]

$$\mathcal{L}(x, y, \hat{y}_1, \dots \hat{y}_{p^-}, w) = -\log \frac{e^{-F_w(x,y)}}{e^{-F_w(x,y)} + \sum_{i=1}^{p^-} e^{-F_w(x,\hat{y}_i,w)}}$$

FACEBOOK AI

Maximum Likelihood as a special case of Contrastive Method



Push down on the energy of training samples
Pull up the energy of everything else to infinity

Refresher on turning energies to probabilities

- **Gibbs distribution (a.k.a. softmax, should be called softargmax)**
- **Discrete / Continuous**

$$P_w(y) = \frac{e^{-\beta F_w(y)}}{\sum_{y'} e^{-\beta F_w(y')}} \qquad P_w(y) = \frac{e^{-\beta F_w(y)}}{\int_{y'} e^{-\beta F_w(y')}}$$

- **Joint distribution**

$$P_w(y,z) = \frac{e^{-\beta E_w(y,z)}}{\int_{y'} \int_{z'} e^{-\beta E_w(y',z')}}$$

Partition function

Inverse temperatur

- **Conditional distribution**

$$P_w(y,z|x) = \frac{e^{-\beta E_w(x,y,z)}}{\int_{y'} \int_{z'} e^{-\beta E_w(x,y',z')}}$$

- **Marginal distribution**

$$P_w(y|x) = \int_{z'} P_w(y,z'|x) = \frac{\int_{z'} e^{-\beta E_w(x,y,z')}}{\int_{y'} \int_{z'} e^{-\beta E_w(x,y',z')}}$$

Refresher on turning energies to probabilities

- **Joint distribution**

$$P_w(y,z) = \frac{e^{-\beta E_w(y,z)}}{\int_{y'} \int_{z'} e^{-\beta E_w(y',z')}}$$

- **Conditional distribution**

$$P_w(y|z) = \frac{e^{-\beta E_w(y,z)}}{\int_{y'} e^{-\beta E_w(y',z)}}$$

- **Marginal distribution**

$$P_w(z) = \frac{\int_{y'} e^{-\beta E_w(y',z)}}{\int_{z'} \int_{y'} e^{-\beta E_w(y',z')}}$$

- **Bayes rules!**

$$P_w(y,z) = P_w(y|z)P_w(z) = P_w(z|y)P_w(y)$$

Negative log-likelihood loss

$$L(x, y, w) = -\frac{1}{\beta} \log P_w(y|x) = F_w(x, y) + \frac{1}{\beta} \log \left[\int_{y'} e^{-\beta F_w(x,y')} \right]$$

- **Gradient of log partition function**

Minus log partition functio

Like a free energy over y.

$$\frac{\partial \left[-\frac{1}{\beta} \log \left[\int_{y'} e^{-\beta F_w(x,y')} \right] \right]}{\partial w} = \int_{y'} P_w(y'|x) \frac{\partial F_w(x, y')}{\partial w}$$

- **Monte Carlo methods: sample y from P(y|x)**
  - The integral is an expectation of the gradient over the distribution of y
  - Sample y from the distribution and average the corresponding gradients.

Max Likelihood is (generally) a (bad) Contrastive Method

- **Push down on data points,**
- **Push up on all points**
- **Max likelihood / probabilistic models**

$$P_w(y|x) = \frac{e^{-\beta F_w(x,y)}}{\int_{y'} e^{-\beta F_w(x,y')}}$$

- Loss: $$\mathcal{L}(x,y,w) = F_w(x,y) + \frac{1}{\beta}\log\int_{y'} e^{-\beta F_w(x,y')}$$
- Gradient: $$\frac{\partial \mathcal{L}(x,y,w)}{\partial w} = \frac{\partial F_w(x,y)}{\partial w} - \int_{y'} P_w(y'|x)\frac{\partial F_w(x,y')}{\partial w}$$
- 2nd term is intractable: MC/MCMC/HMC/CD: $\hat{y}$ sampled from $P_w(y|x)$

$$\frac{\partial \mathcal{L}(x,y,w)}{\partial w} = \frac{\partial F_w(x,y)}{\partial w} - \frac{\partial F_w(x,\hat{y})}{\partial w}$$

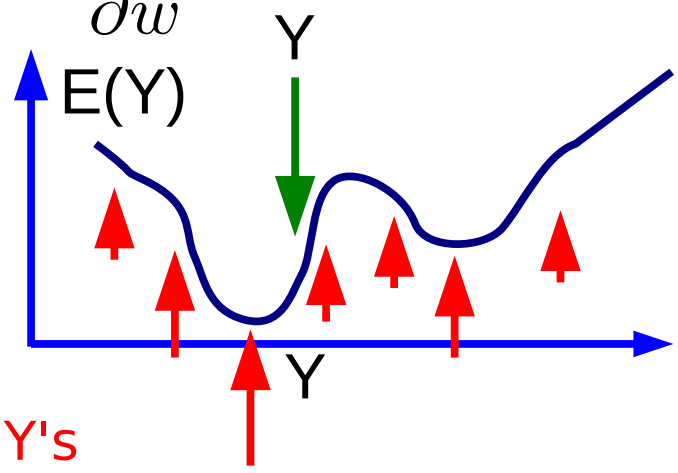
push down of the energy of data points, push up everywhere else

Gradient of the negative log-likelihood loss for one sample Y:

$$\frac{\partial \mathcal{L}(x,y,w)}{\partial w} = \frac{\partial F_w(x,y)}{\partial w} - \int_{y'} P_w(y'|x) \frac{\partial F_w(x,y')}{\partial w}$$

Gradient descent:

$$w \leftarrow w - \eta \frac{\partial \mathcal{L}(x,y,w)}{\partial w}$$



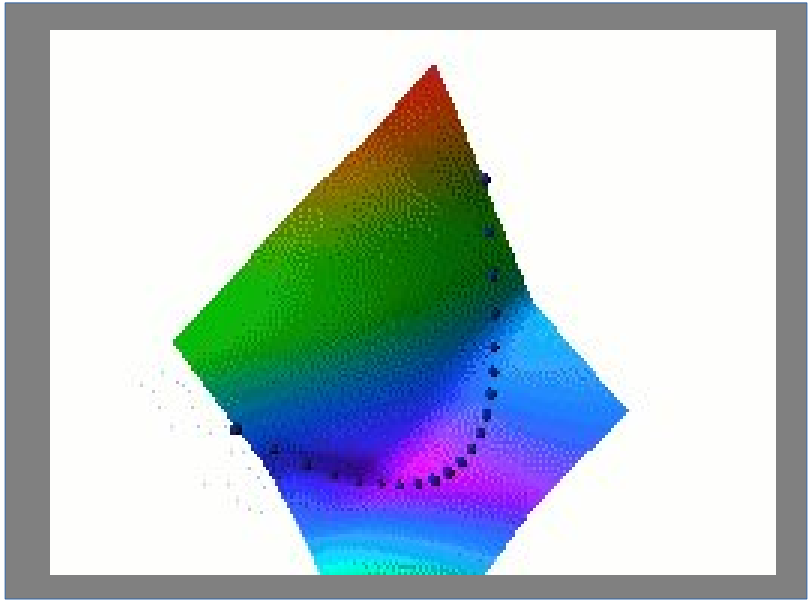
Pushes down on the energy of the samples

Pulls up on the energy of low-energy Y's

$$w \leftarrow w - \eta \frac{\partial F_w(x,y)}{\partial w} + \eta \int_{y'} P_w(y'|x) \frac{\partial F_w(x,y')}{\partial w}$$

Problem with Max Likelihood / Probabilistic Methods

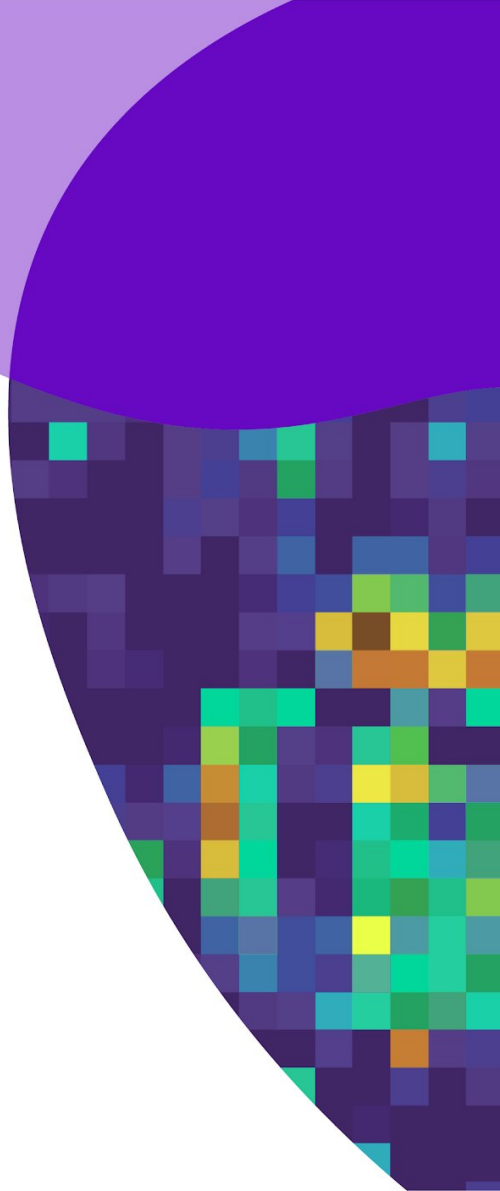
- **It wants to make the difference between the energy on the data manifold and the energy just outside of it infinitely large!**
- **It wants to make the data manifold an infinitely deep and infinitely narrow canyon.**
- **The loss must be regularized to keep the energy smooth**
  - e.g. with Bayesian prior or by limiting weight sizes à la Wasserstein GAN.
  - So that gradient-based inference works
  - Equivalent to a Bayesian prior
  - But then, why use a probabilistic model?



FACEBOOK AI

Latent Variable Energy-Based Models

Minimize or marginalize the energy with respect to the latent variable



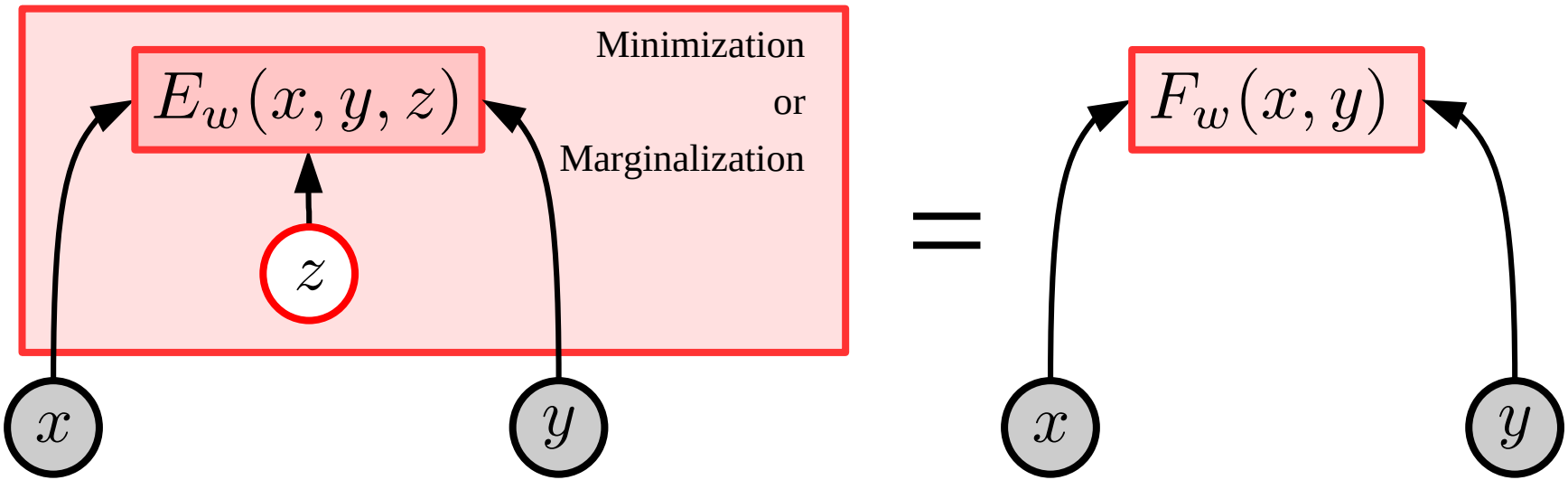
Latent-Variable Energy-Based Model

- **Zero temperature limit**

$$\check{z} = \text{argmin}_{z \in \mathcal{Z}} E_w(x, y, z) \qquad F_w(x, y) = E_w(x, y, \check{z})$$

- **Marginalization**

$$F_w(x, y) = -\frac{1}{\beta} \log \left[\int_{z'} e^{-\beta E_w(x,y,z')} \right]$$



Marginalization for Latent-Variable EBM

- **Gibbs distribution (a.k.a. softmax, should be called softargmax)**
- **Marginalization over latent variable z**

$$P_w(y|x) = \frac{\int_{z'} e^{-\beta E_w(x,y,z')}}{\int_{y'} \int_{z'} e^{-\beta E_w(x,y',z')}}$$

- **Definition of free energy over latent variable z**

$$F_w(x,y) = -\frac{1}{\beta} \log \left[\int_{z'} e^{-\beta E_w(x,y,z')} \right]$$

- **Marginal distribution = Gibbs formula with free energy**

$$P_w(y|x) = \frac{e^{-\beta F_w(x,y)}}{\int_{y'} e^{-\beta F_w(x,y')}}$$

Marginalizing over a latent variable

$$P(y,z|x) = \frac{e^{-\beta E(x,y,z)}}{\int_y \int_z e^{-\beta E(x,y,z)}} \qquad P(y|x) = \int_z P(y,z|x)$$

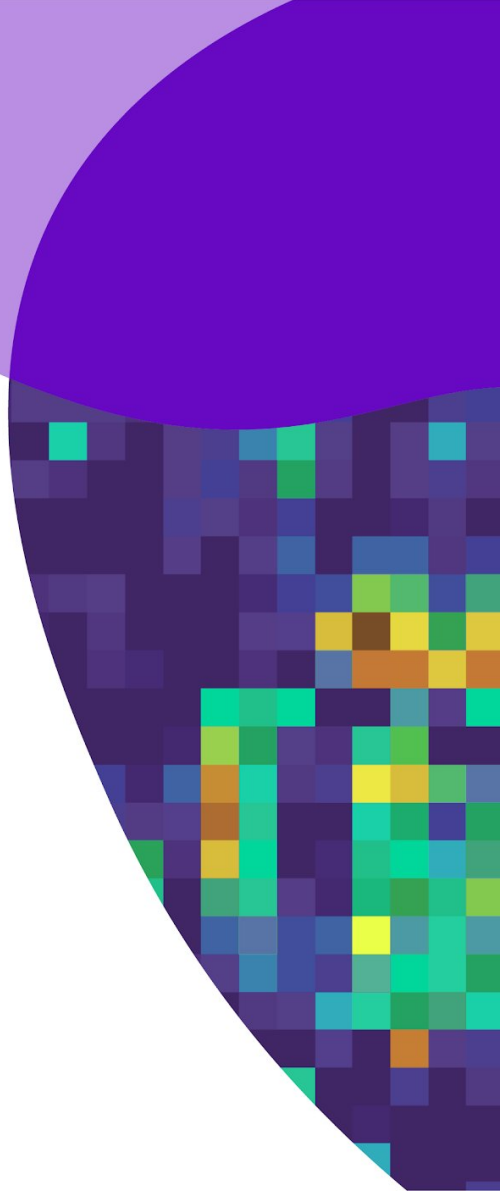
$$P(y|x) = \frac{\int_z e^{-\beta E(x,y,z)}}{\int_y \int_z e^{-\beta E(x,y,z)}} = \frac{e^{-\beta\left[-\frac{1}{\beta}\log \int_z e^{-\beta E(x,y,z)}\right]}}{\int_y e^{-\beta\left[-\frac{1}{\beta}\log \int_z e^{-\beta E(x,y,z)}\right]}} = \frac{e^{-\beta F_\beta(x,y)}}{\int_y e^{\beta F_\beta(x,y)}}$$

- **Free energy F(x,y)** $F_\beta(x,y) = -\frac{1}{\beta}\log \int_z e^{-\beta E(x,y,z)}$

FACEBOOK AI

Hopfield Nets and Boltzmann Machines

Simple concepts of historical relevance



Hopfield Nets (Hopfield 1982)

- **Energy-based model**
- **Fully-connected recurrent network with symmetric connections**
- **Binary activations**

$$E(y) = -\sum_{ij} y_i w_{ij} y_j$$

- **Inference: update neuron states with:** $y_i \leftarrow \text{sign}(\sum_j w_{ij} y_j)$
  - This makes the energy go down
- **Learning rule: minimize energy of training samples**
  - Dig holes around training samples.
  - no contrastive term! which is why it doesn't work very well

$$L(y, w) = E(y) \qquad \frac{\partial L(y,w)}{\partial w_{ij}} = -y_i y_j \qquad w_{ij} \leftarrow w_{ij} + y_i y_j$$

Boltzmann Machine [Hinton & Sejnowski 1983]

- **A Hopfield net with hidden units**

$$E(y,z) = -\sum_{ij} y_i w_{ij}^{yy} y_j - \sum_{ij} z_i w_{ij}^{zz} z_j - \sum_{ij} y_i w_{ij}^{yz} z_j$$

- **Free energy: marginalizes over z**

$$F(y) = -\log \sum_z \exp(-E(y,z))$$

- **Loss: Negative log-likelihood (with MCMC contrastive samples)**

$$L(y,w) = F_w(y) + \log \sum_{y'} \exp(-F_w(y'))$$

$$\frac{\partial L(y,w)}{\partial w_{ij}} = \frac{\partial F_w(y)}{\partial w} - \sum_{y'} P(y') \frac{\partial F_w(y')}{\partial w} \qquad \text{with} \quad P(y) = \frac{\exp(-F_w(y)}{\sum_{y'} \exp(-F_w(y'))}$$

Boltzmann Machine [Hinton & Sejnowski 1983]

- **But how do we marginalize on z? → MCMC sampling**
  - MCMC sampling: on z for first term; on z and y for second term

$$L(y,w) = F_w(y) + \log \sum_{y'} \exp(F_w(y')) \qquad F(y) = -\log \sum_{z} \exp(-E(y,z))$$

$$\frac{\partial L(y,w)}{\partial w_{ij}} = \sum_{y',z'} P(z'|y) \frac{\partial E_w(y,z')}{\partial w} - \sum_{y',z'} P(y',z') \frac{\partial E_w(y',z)}{\partial w} \qquad \text{with}$$

$$P(z|y) = \frac{\exp(-E_w(y,z)}{\sum_{z'} \exp(-E_w(y,z'))} \qquad P(y,z) = \frac{\exp(-E_w(y,z)}{\sum_{y',z'} \exp(-E_w(y',z'))}$$

$\check{z}$ sampled from $P(z|y)$ $\qquad$ $\hat{y}, \hat{z}$ sampled from $P(y,z)$

$$\frac{\partial L(y,w)}{\partial w_{ij}^{yz}} \approx -y_i \check{z}_j + \hat{y}_i \hat{z}_j \qquad w_{ij} \leftarrow w_{ij} + \eta(y_i z_j - \hat{y}_i \hat{z}_j)$$

Boltzmann Machine [Hinton & Sejnowski 1983]

- **But how do we get MCMC samples of z (and y)?**

$\check{z}$ sampled from $P(z|y)$ $\qquad$ $\hat{y}, \hat{z}$ sampled from $P(y,z)$

$$P(z|y) = \frac{\exp(-E_w(y,z)}{\sum_{z'} \exp(-E_w(y,z'))} \qquad P(y,z) = \frac{\exp(-E_w(y,z)}{\sum_{y',z'} \exp(-E_w(y',z'))}$$

$$E(y,z) = -\sum_{ij} y_i w_{ij}^{yy} y_j - \sum_{ij} z_i w_{ij}^{zz} z_j - \sum_{ij} z_i w_{ij}^{yz} y_j$$

$$E(y,z)_{z_i=1} - E(y,z)_{z_i=0} = -\sum_j 1 w_{ij}^{yz} y_j + \sum_j 0 w_{ij}^{yz} y_j = \sum_j w_{ij}^{yz} y_j$$

$$\check{z}_i \text{ sampled from } P(z_i|y, z_{i\neq i}) = \frac{1}{1+\exp(-\sum_j w_{ij}^{yz} y_i)}$$

Boltzmann Machine [Hinton & Sejnowski 1983]

$$E(y,z)_{z_i=1} - E(y,z)_{z_i=0} = -\sum_j 1 w_{ij}^{yz} y_j + \sum_j 0 w_{ij}^{yz} y_j = \sum_j w_{ij}^{yz} y_j$$

$$P(z_i = 1|y,z) \propto \exp(-E(y,z,z_i=1)) \qquad P(z_i = 0|y,z) \propto \exp(-E(y,z,z_i=0))$$

$$P(z_i = 1|y,z) = \frac{\exp(-E(y,z)_{z_i=1})}{\exp(-E(y,z)_{z_i=1}) + \exp(-E(y,z)_{z_i=0}))}$$

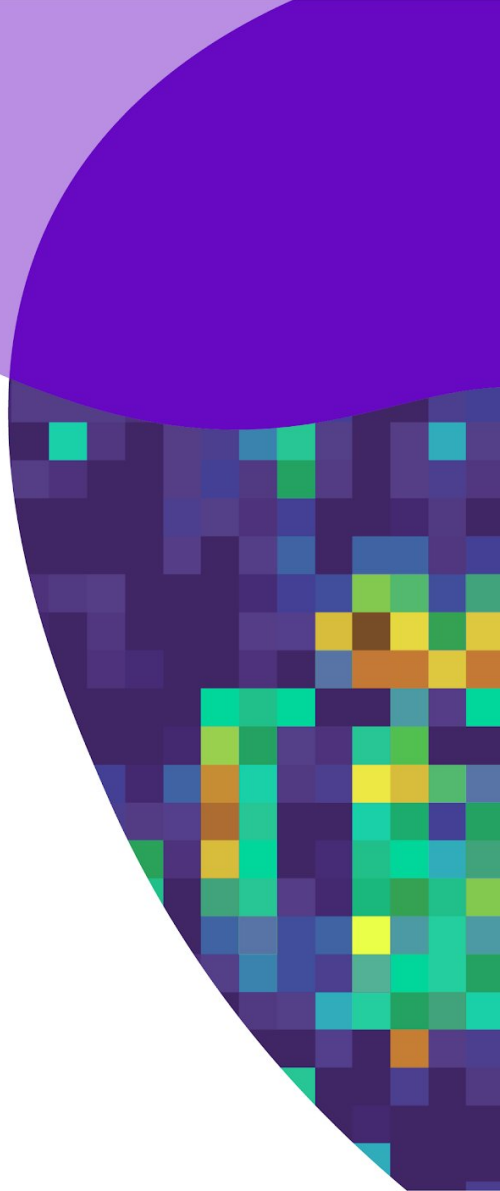
$$P(z_i = 1|y,z) = \frac{1}{1 + \exp(E(y,z)_{z_i=1} - E(y,z)_{z_i=0})}$$

$$\check{z}_i \text{ sampled from } P(z_i|y,z_{i\neq i}) = \frac{1}{1 + \exp(-\sum_j w_{ij}^{yz} y_i)}$$

FACEBOOK AI

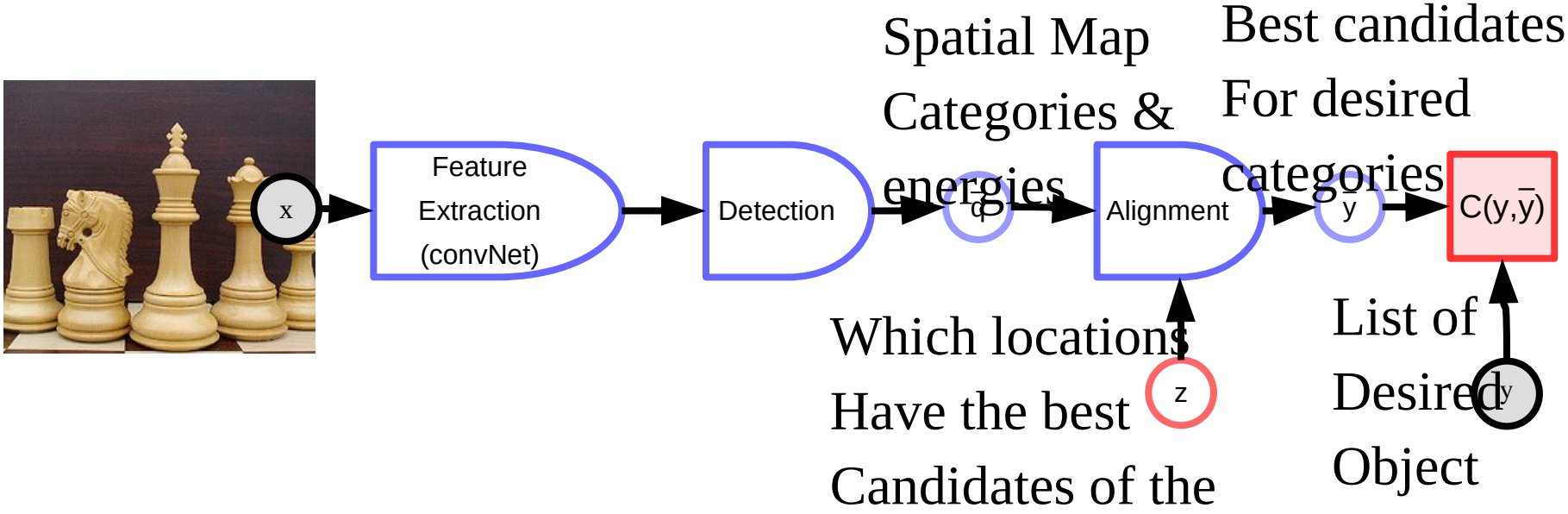
Latent Variable Models in Practice

Structured prediction with latent variable models



Structured Prediction

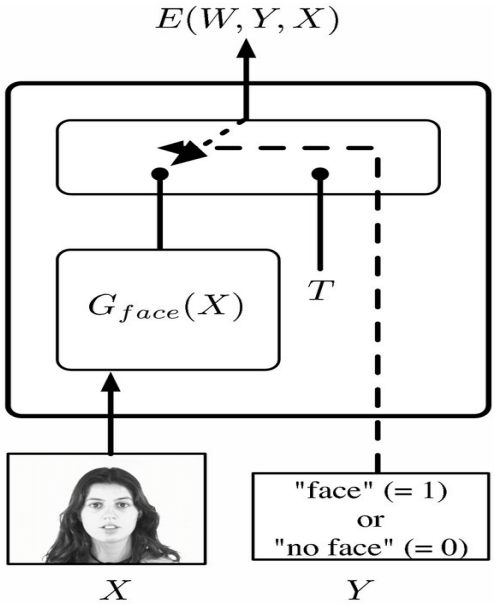
- **Complex output with weak/partial supervision**
  - Speech recognition: alignment of audio to text transcription
  - Handwriting recognition: alignment of image to character sequence
  - Object detection: alignment of image features to labeled categories

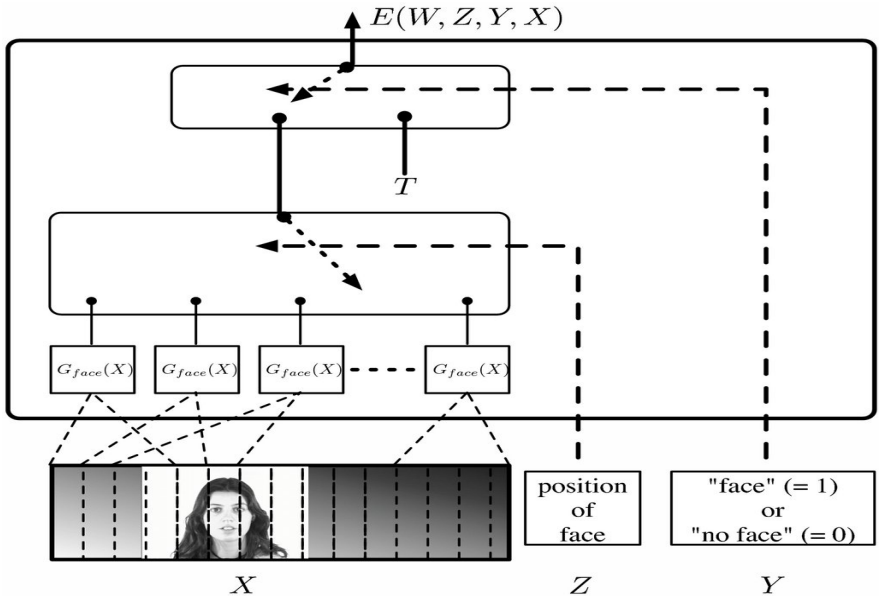


Example of Latent Variable Models: object detection

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

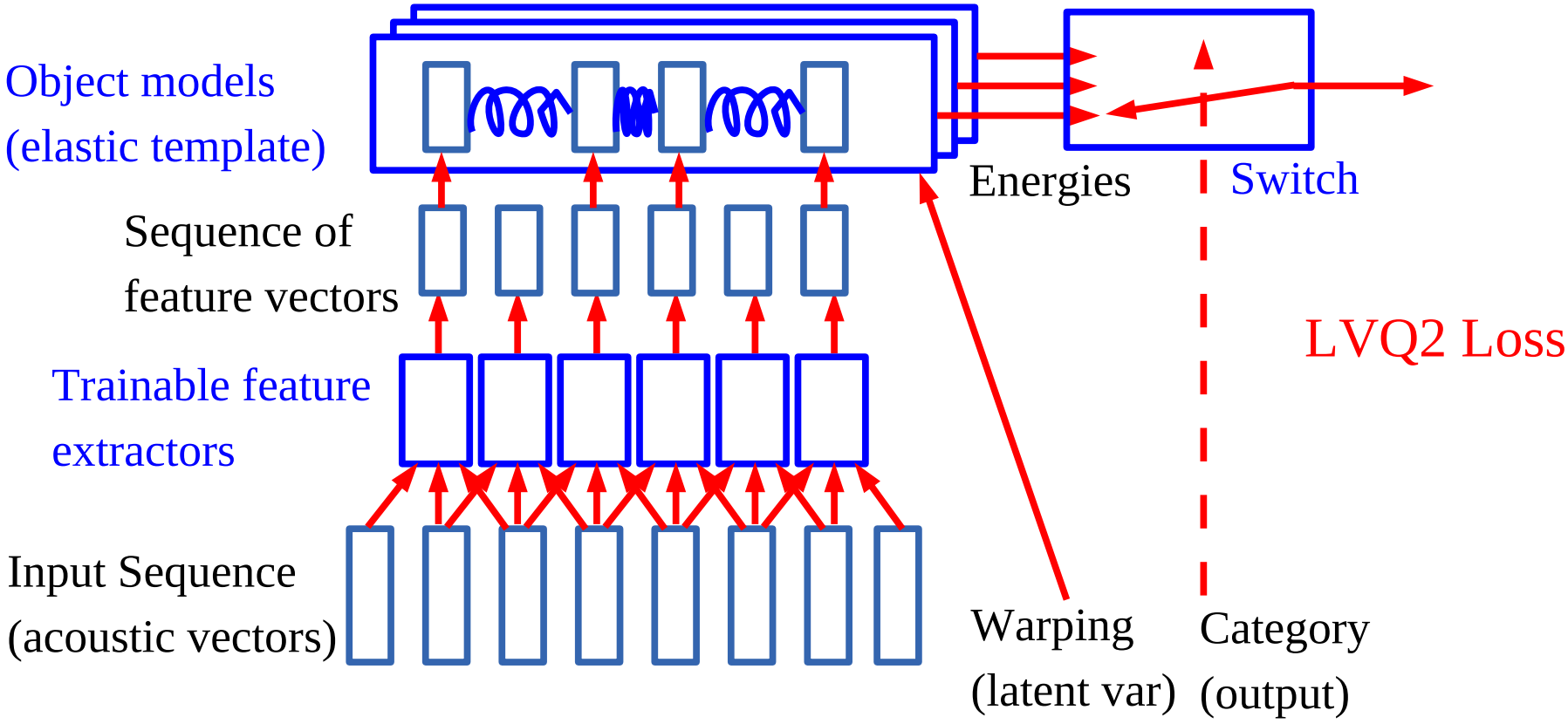
$$Y^* = \text{argmin}_{Y \in \mathcal{Y}, Z \in \mathcal{Z}} E(Z, Y, X).$$





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]

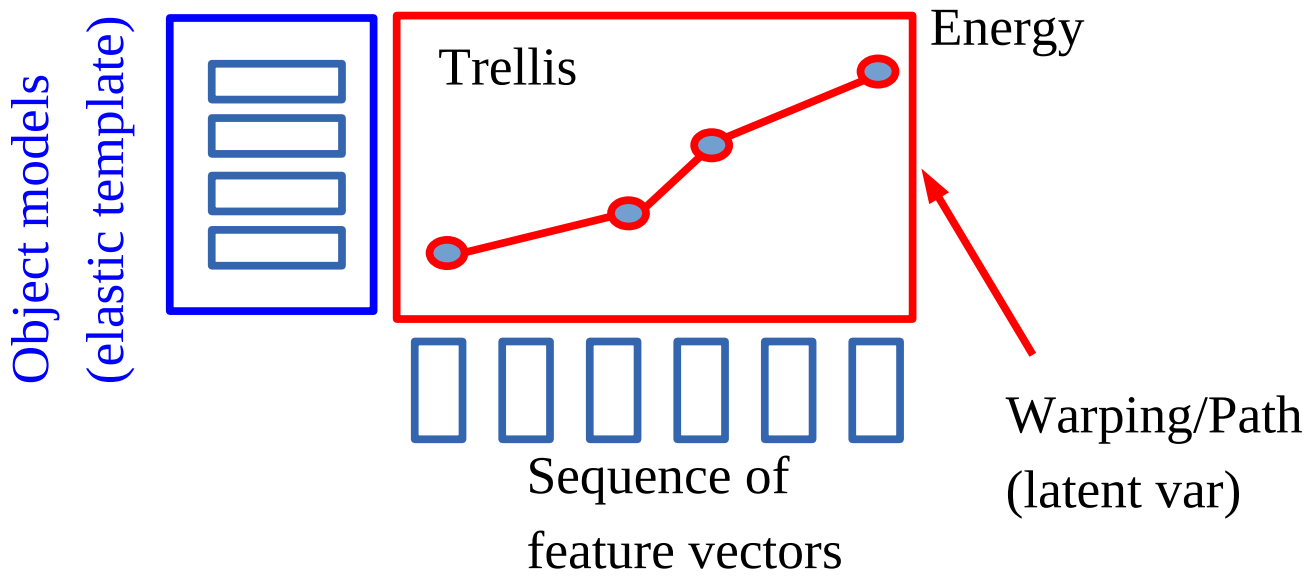


Example: 1-D Constellation Model (a.k.a. Dynamic Time Warping)

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)

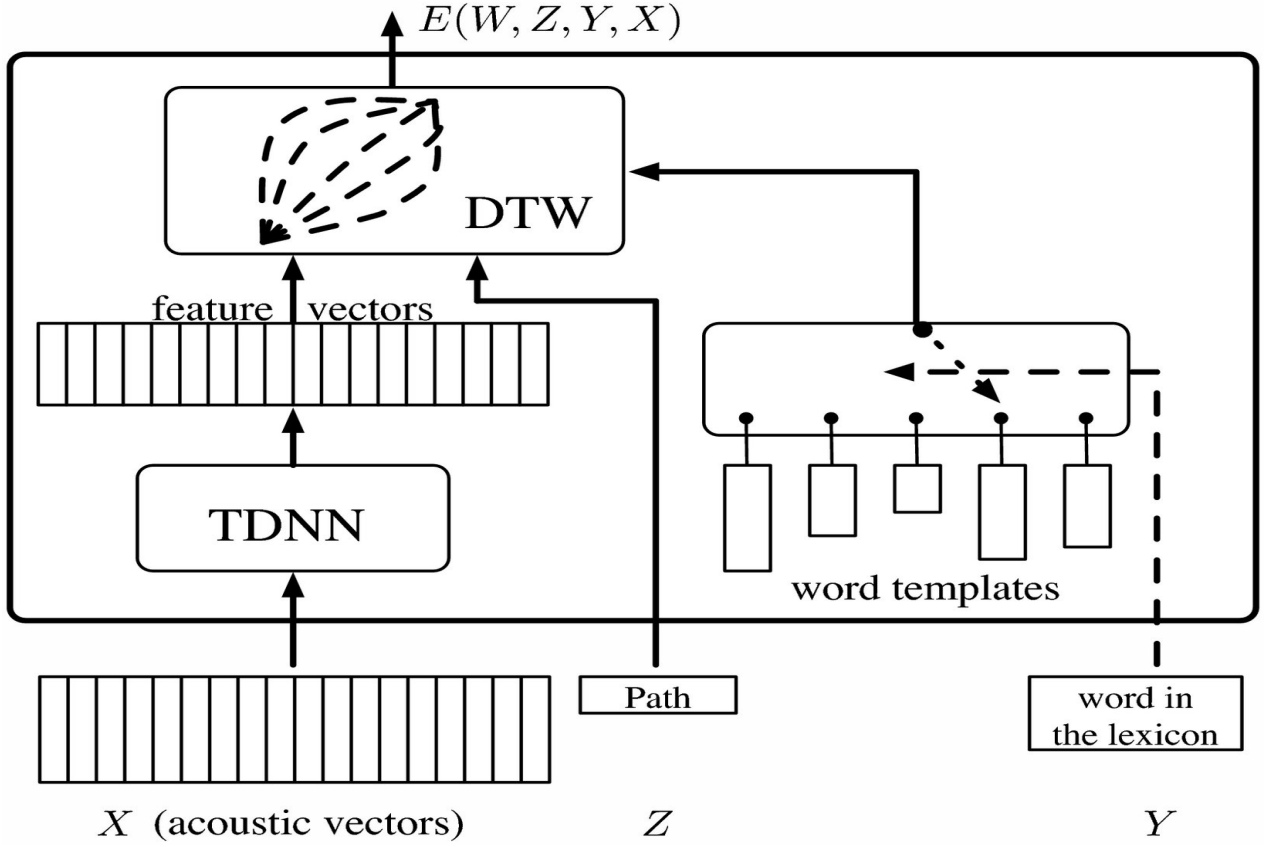
The feature extractor and the structured classifier are trained simultaneously in an integrated fashion.

with the LVQ2 Loss :

- Driancourt and Bottou's speech recognizer (1991)

with NLL:

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



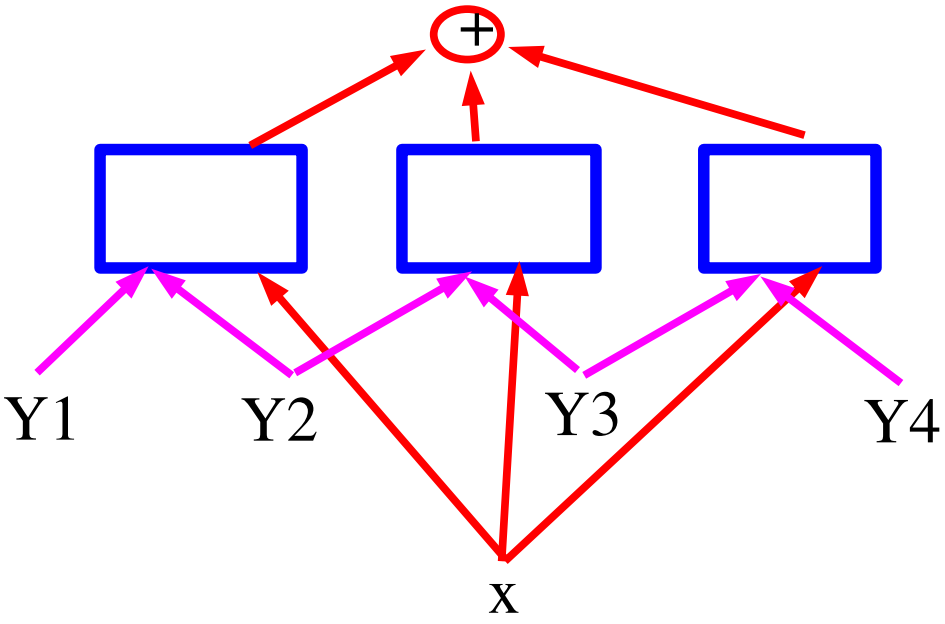
What can the latent variables represent?

- **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,…), assignment of labels to objects.
- **In general, we will search for the value of the latent variable that allows us to get an answer (y) of smallest energy.**

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)

$$\check{y} = \text{argmin}_{z \in \mathcal{Z}, y \in \mathcal{Y}} E(x, y, z)$$

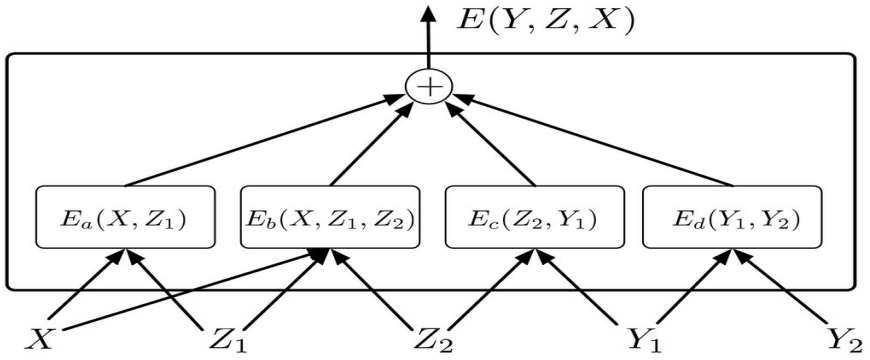


Efficient Inference: Energy-Based Factor Graphs

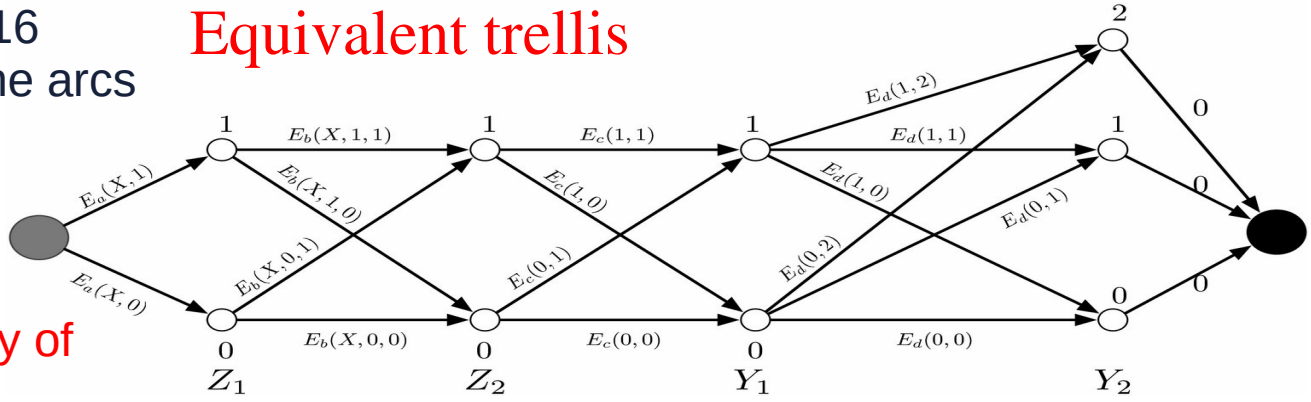
- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3=24 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 6.
  - Hence, we can precompute the 16 factor values, and put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

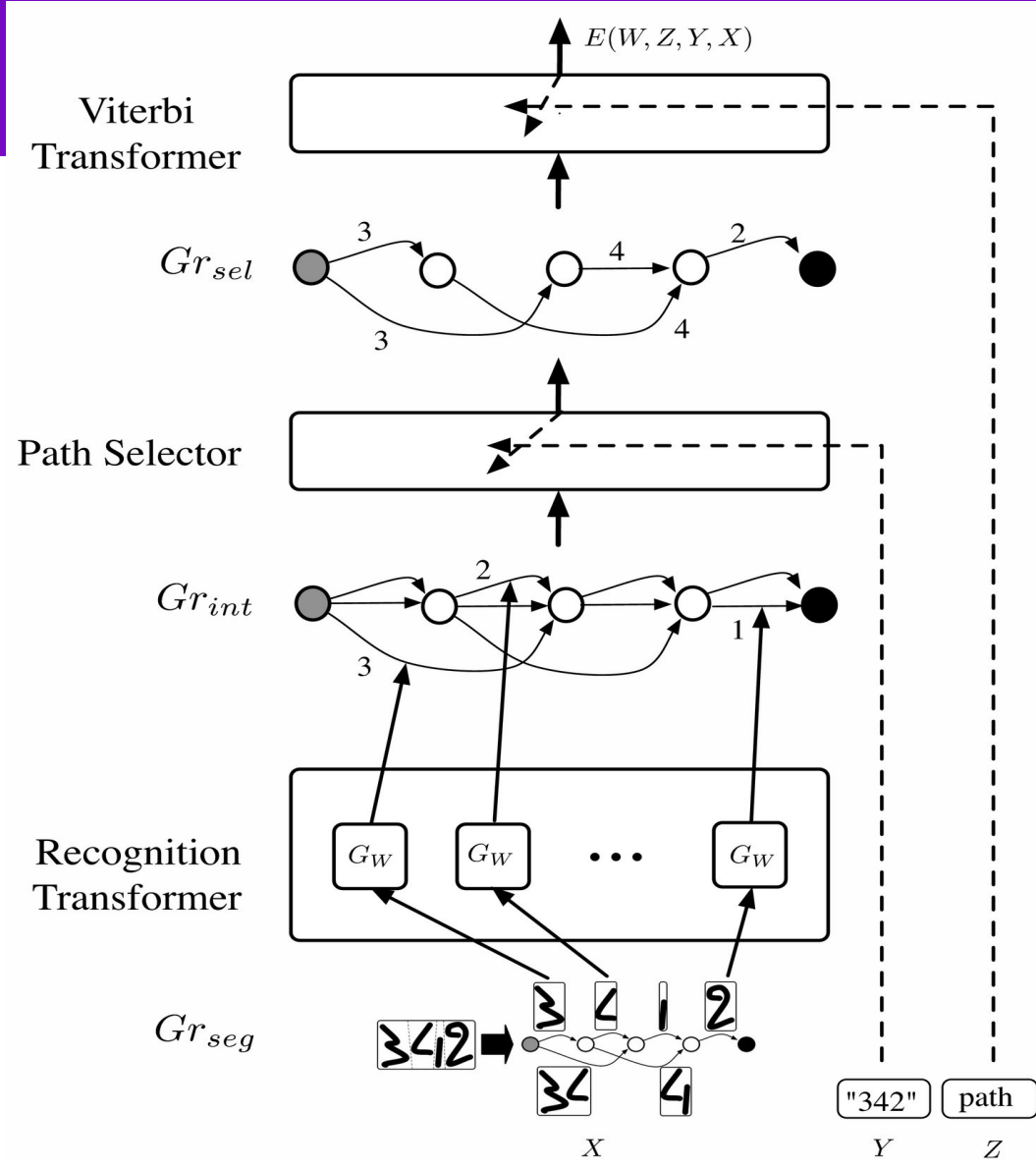


Equivalent trellis



Deep Factors & implicit graphs: GTN

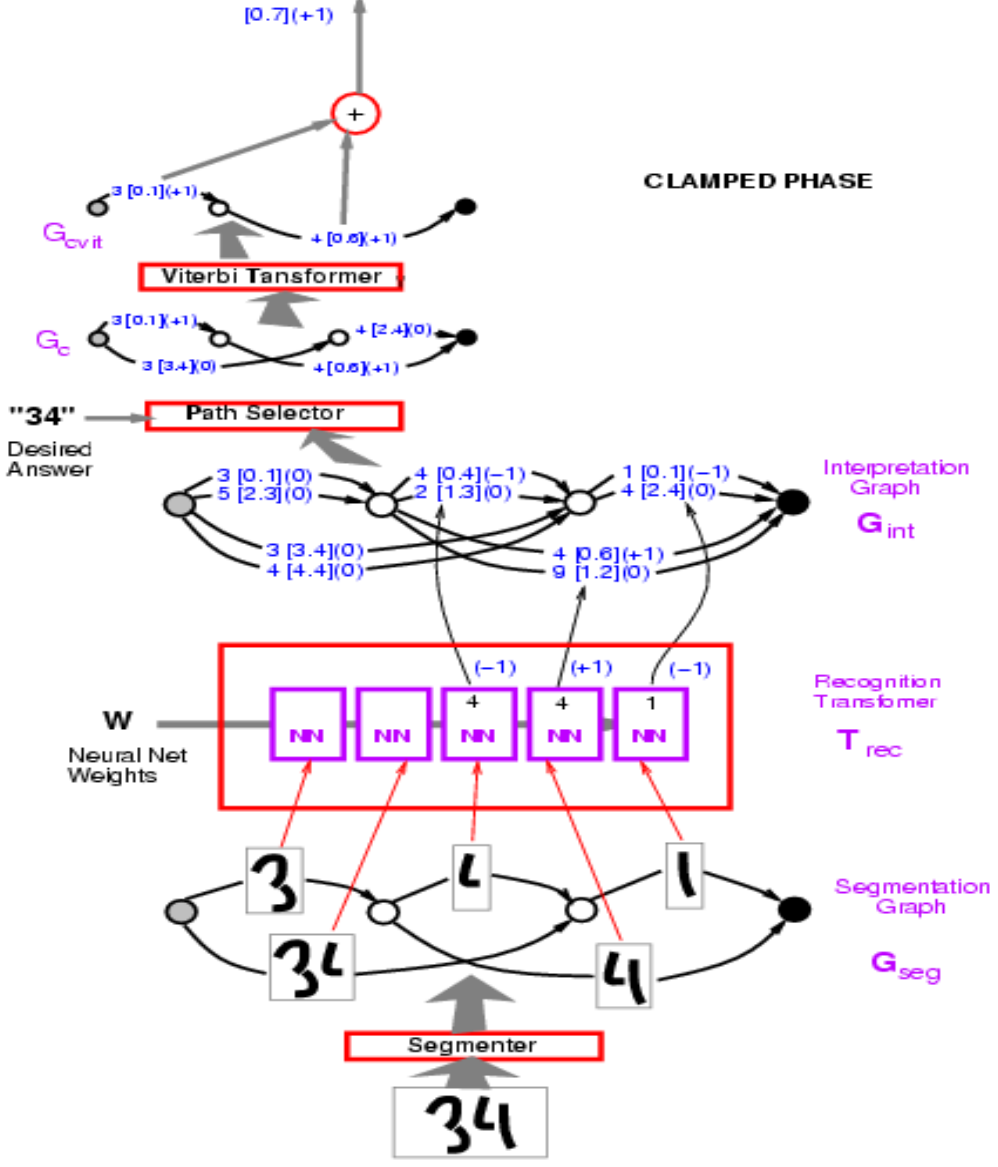
- **Handwriting Recognition with Graph Transformer Networks**
- **Un-normalized hierarchical HMMs**
  - Trained with Perceptron loss [LeCun, Bottou, Bengio, Haffner 1998]
  - Trained with NLL loss [Bengio, LeCun 1994], [LeCun, Bottou, Bengio, Haffner 1998]
- **Answer = sequence of symbols**
- **Latent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

$$F_w(x, y) = \min_{z \in \text{paths}} E_w(x, y, z)$$

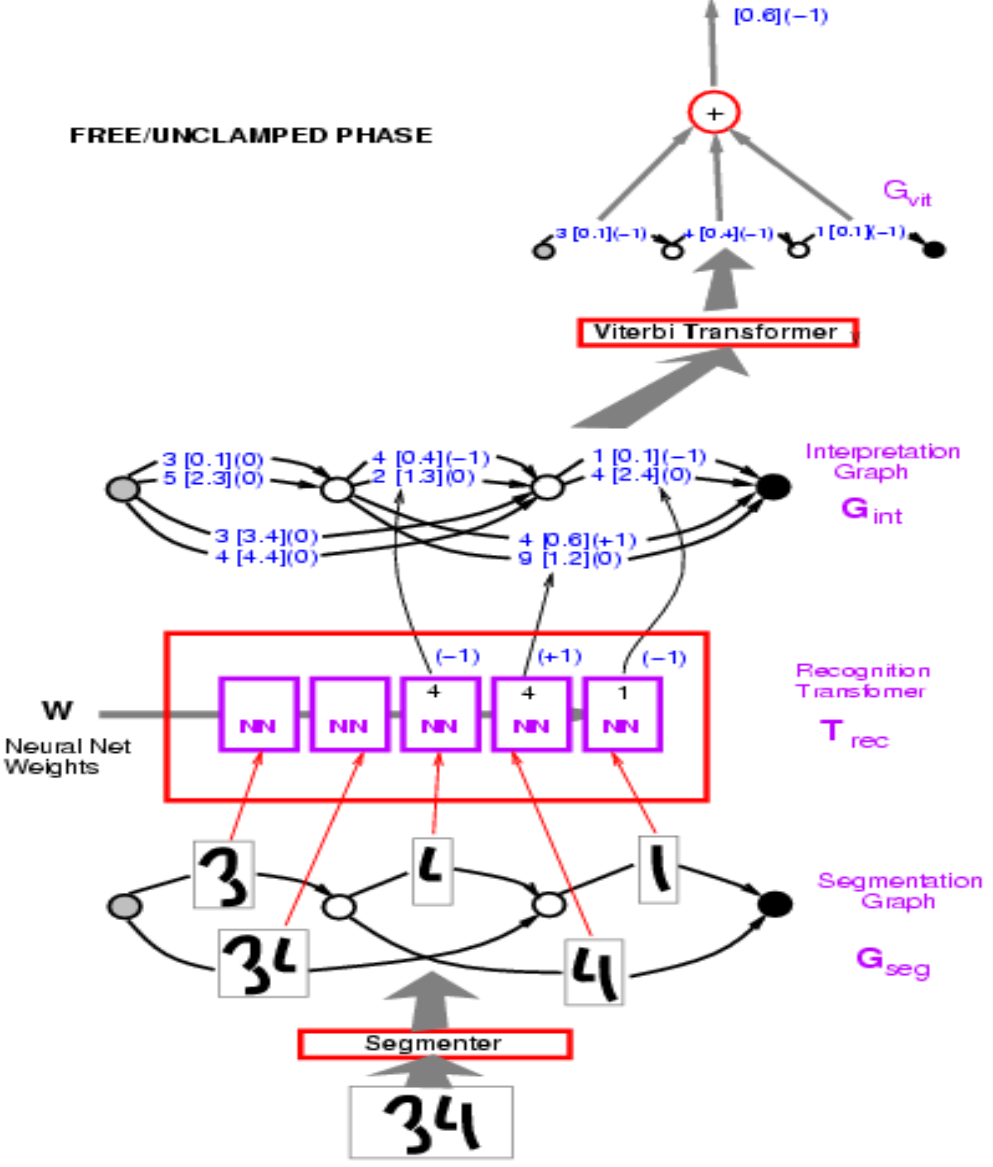


Graph Transformer Networks

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

$$\hat{y} = \text{argmin}_{\hat{y} \neq y} \min_{z \in \mathcal{Z}} E_w(x, y, z)$$

$$F_w(x, \hat{y}) = \min_{z \in \text{paths}} E_w(x, \hat{y}, z)$$



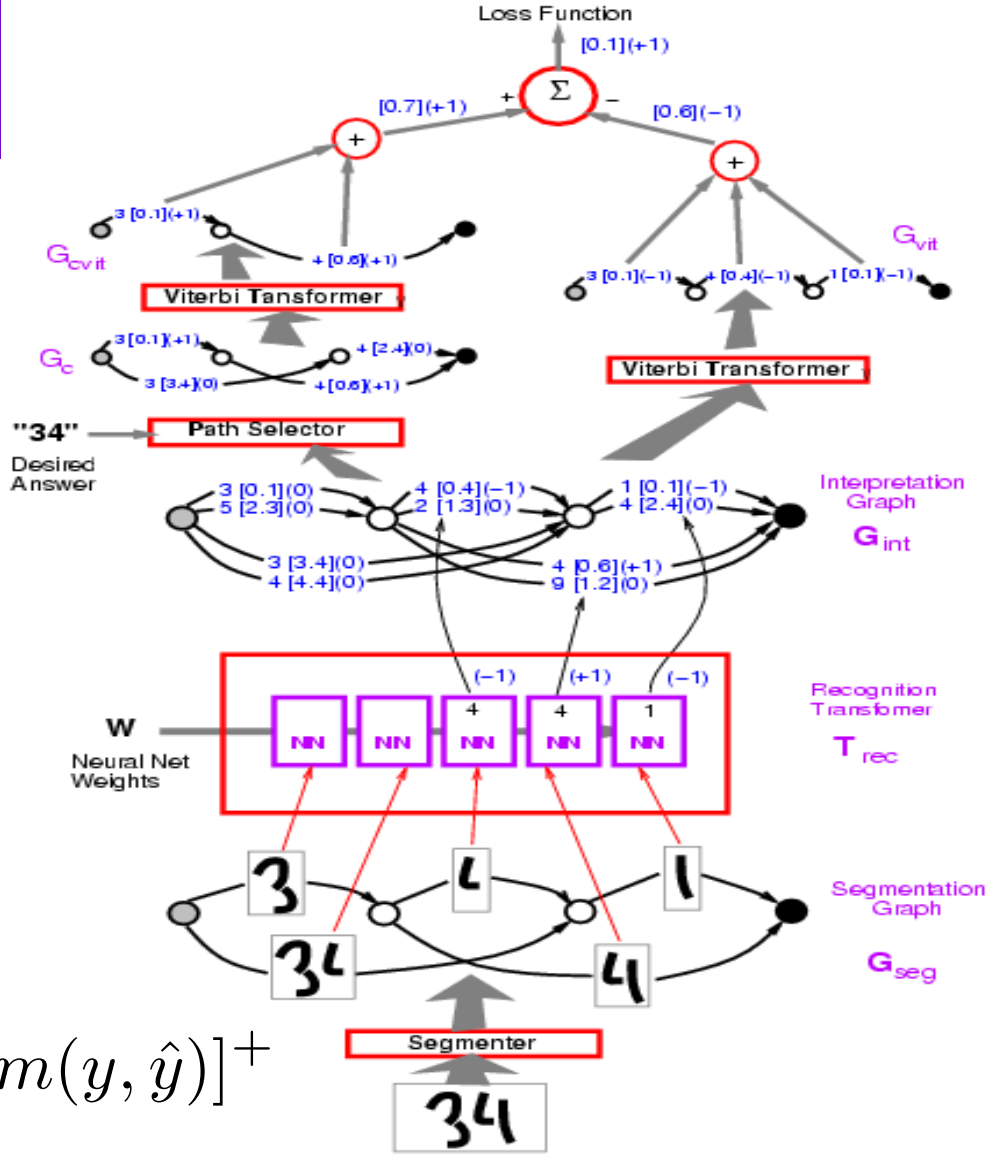
Graph Transformer Networks

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

$$L(x, y, w) = F_w(x, y) - F_w(x, \hat{y})$$

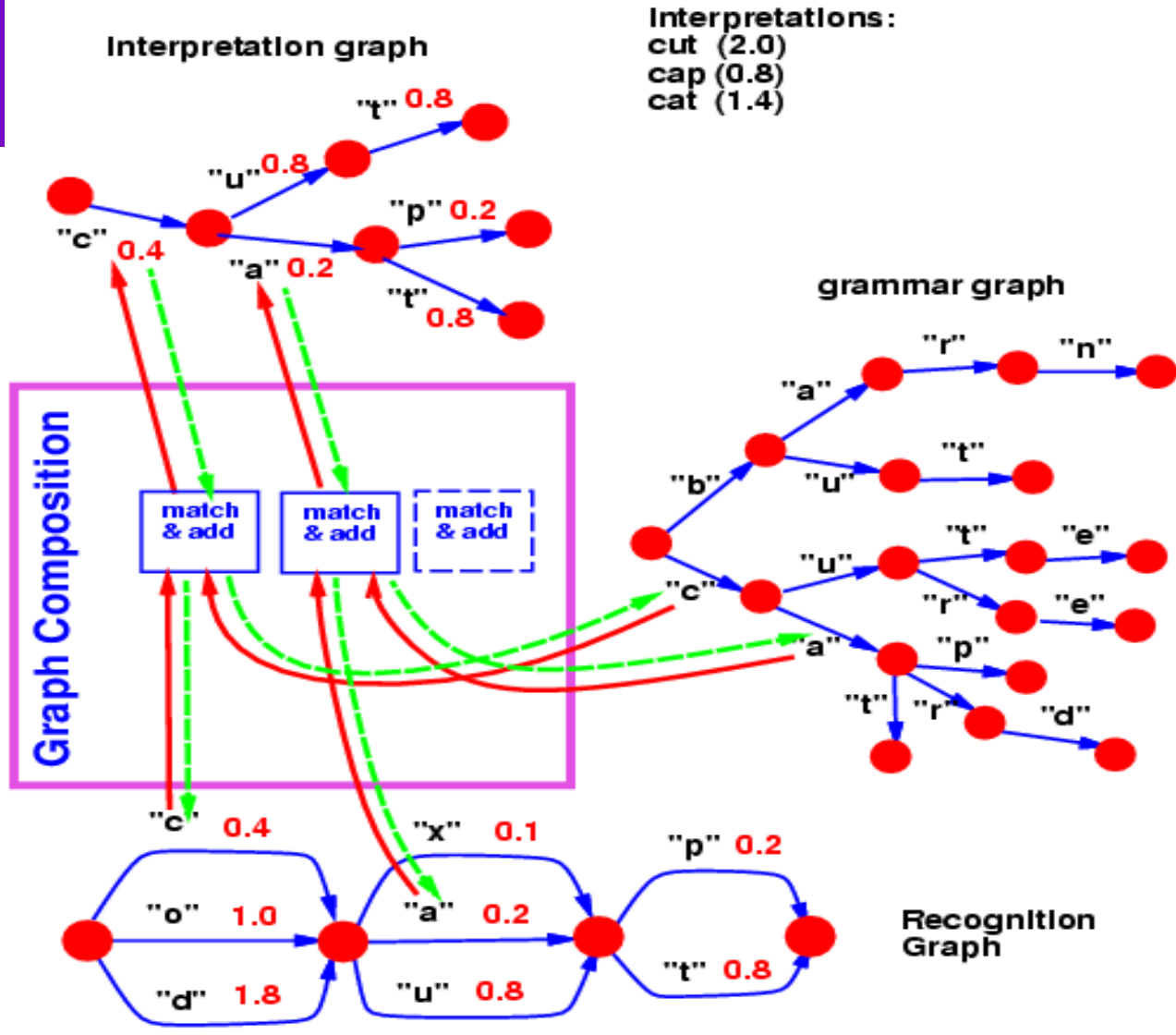
- **Hinge Loss**

$$L(x, y, w) = [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+$$



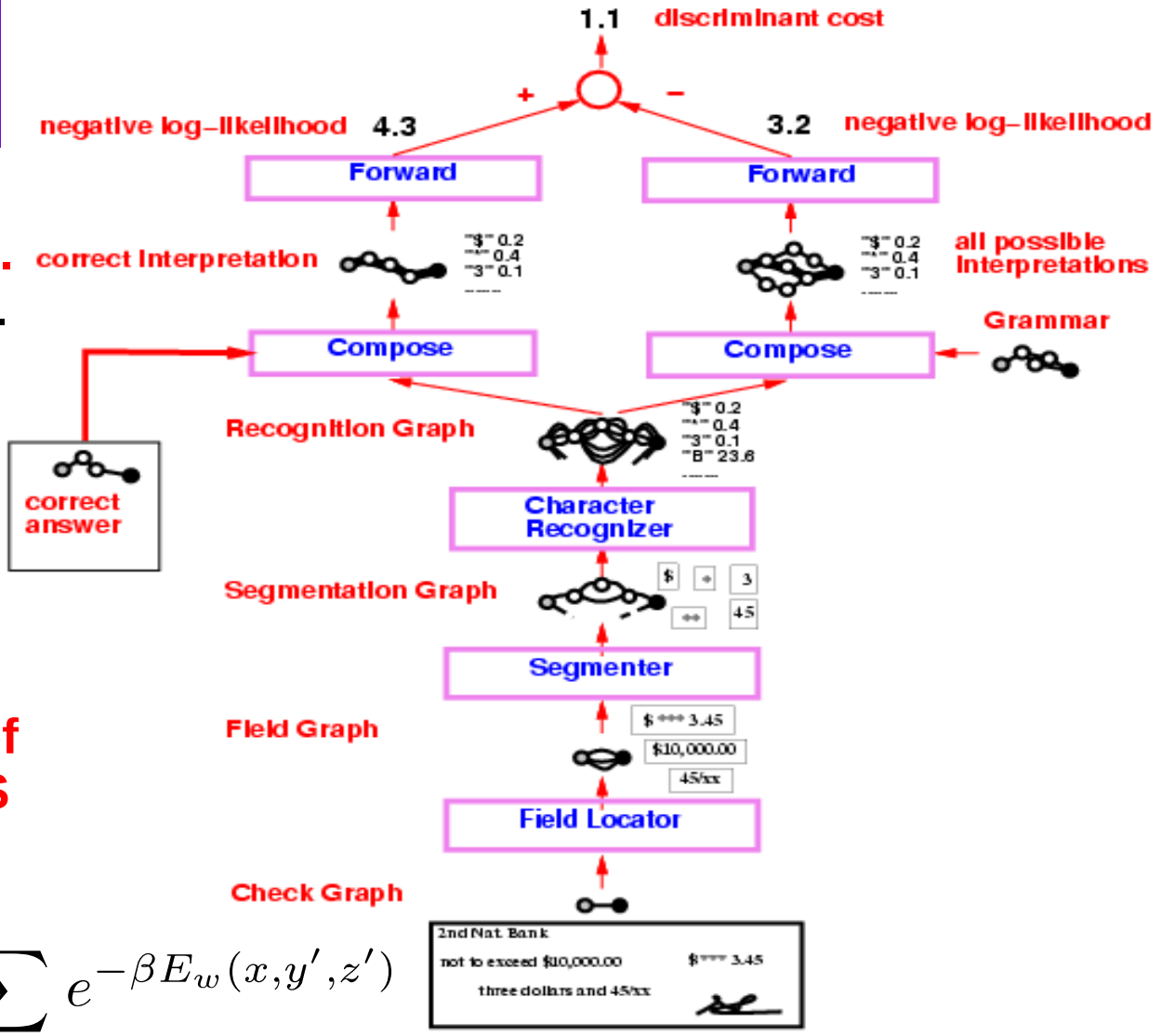
Graph Composition, Transducers.

- 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% of all the checks written in the US in the late 1990s**



$$L(x, y, w) = F_w(x, y) + \frac{1}{\beta} \log \sum_{y', z'} e^{-\beta E_w(x, y', z')}$$

- **End of lecture**