Find your inner peace, minimize your Energy🧘🏻

A gentle introduction to Generative Energy-Based learning

Luca Miglior

Department of Computer Science
University of Pisa

luca.miglior@phd.unipi.it

Why generative learning 📚

A modern paradigm to (probabilistic) deep learning

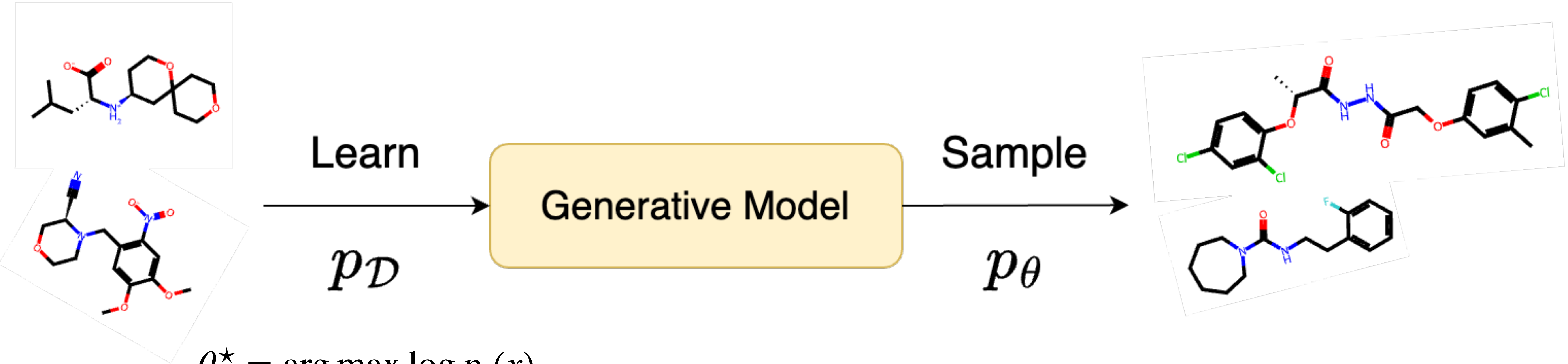
- Commonly defined as a paradigm where models directly learn to generate data, rather than classify them
- Powerful paradigm: capture salient patterns of the data distribution p_D into model's parameters θ
- Key idea: *If you can model how data is generated, you can also understand and manipulate it better!*

$$\theta^\star = \arg\max_\theta \log p_\theta(x)$$

A two-steps approach

Train and make predictions

- Modern generative learning consists of a two-steps approach
- Learn the model from the actual data distribution (e.g., your dataset)
- Find a good sampling procedure to generate new data that belong to the original distribution



$$\theta^\star = \arg\max_\theta \log p_\theta(x)$$

Training Maximum Likelihood estimation

Tackling the optimization problem

- Solving the optimization problem to find the optimal parameters is a non trivial, generally hard task
- A typical solution is Maximum Likelihood Estimation (MLE)
- Find the correct parameters configuration that better approximates the empirical distribution of the dataset $D = \{x_1, \dots x_n\}$

$$\text{MLE objective} ::= \theta^\star = \arg\max_\theta \mathbb{E}_{x \sim D}\left[\log p_\theta(x)\right]$$

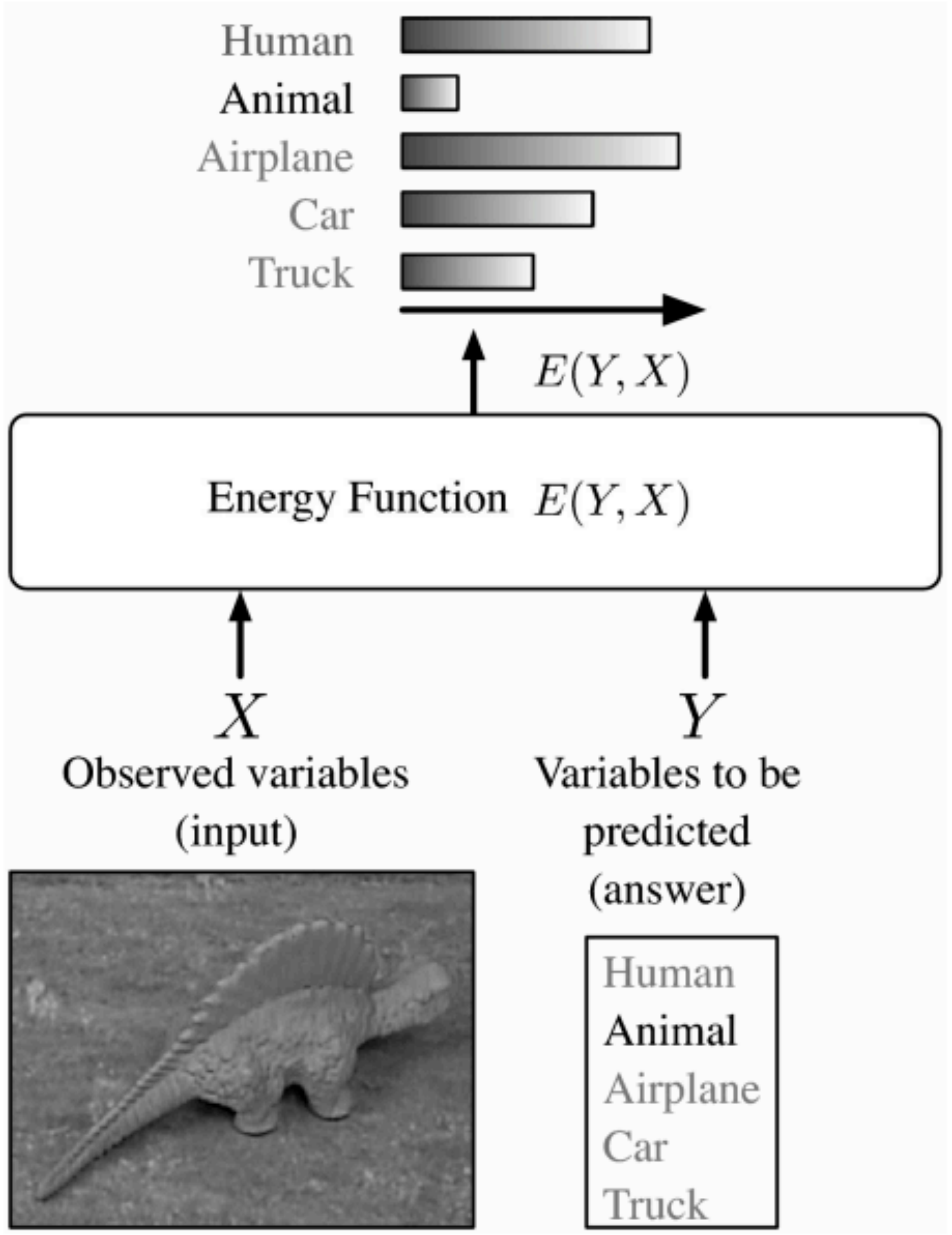
EBMs: a unifying framework for learning

A matter of compatibility 💞

- The model: EBMs are a particular category of models akin to measure the compatibility between an observed variable x and a variable *to be predicted* y

- Inference: as “simple” as searching the right y for our input x

- Clearly infeasible for large search spaces



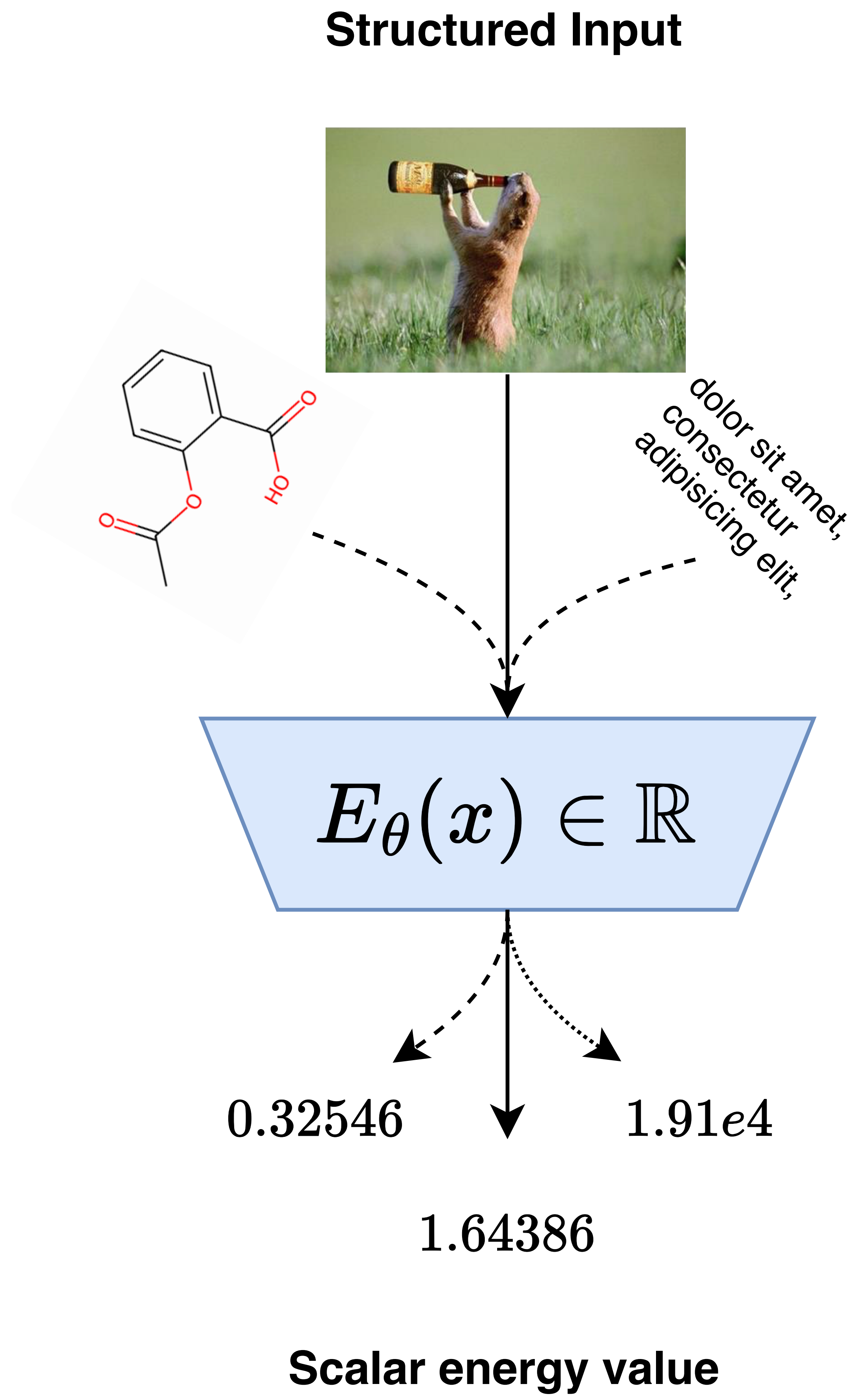
$$y^\star = \arg\min_y E(x, y)$$

Image credits: A Tutorial on Energy-Based Learning, Yann LeCun, Sumit Chopra, Raia Hadsell, Marc’Aurelio Ranzato, and Fu Jie Huang

EBMs: a unifying framework for learning

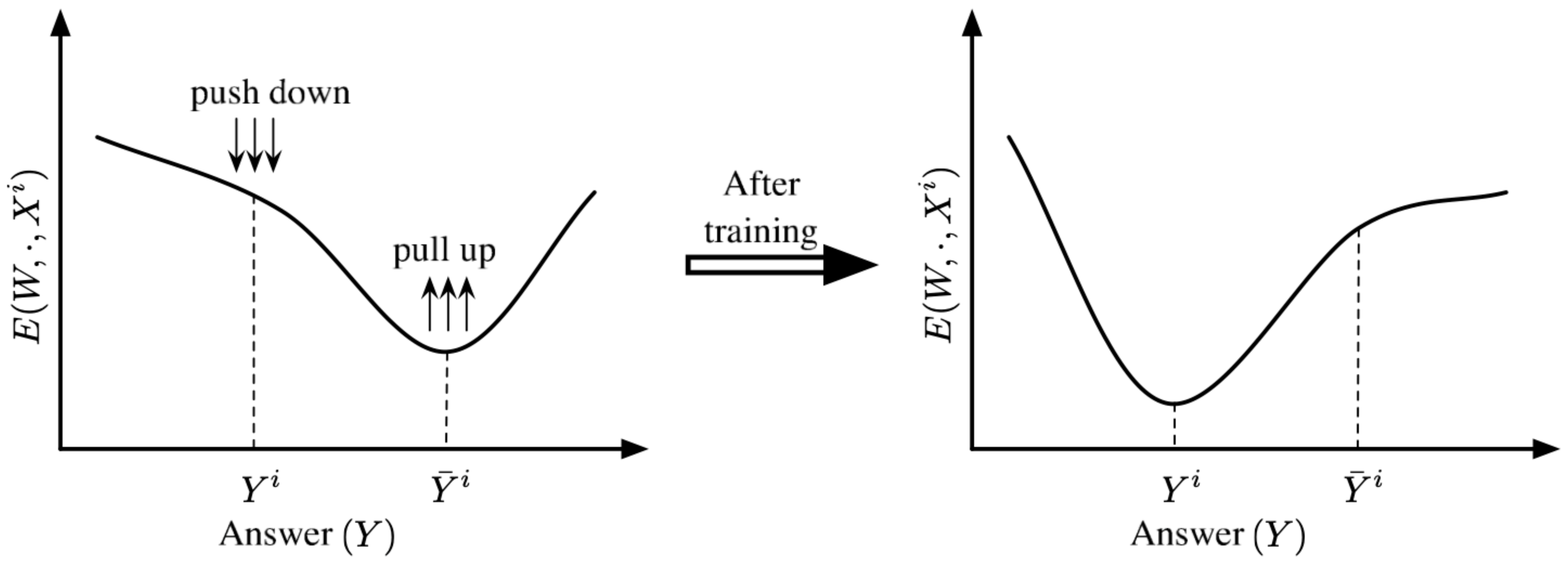
Learn energy values for any possible input

- Learn the data distribution through a parametrized energy function $E_\theta(x) : X \to \mathbb{R}$
- Energy assigns **unnormalized probabilities** to any point in the input space
- Any parametrized function satisfying the above requirements can be used as $E_\theta(x) \to$ any deep neural network
- We still need a way to estimate probabilities in the case of decision making problems



Training Energy Functions 🏋️‍♀️

- Key idea: the goal is to learn a smooth energy landscape on the dataset
- If the energy is smooth enough, we can perform local optimization (generally gradient-based) and make predictions
- Typically made through contrastive learning ⬆️⬇️



Good Loss functions to learn EBMs

Loss (equation #)	Formula	Margin
energy loss (6)	$E(W, Y^i, X^i)$	none
perceptron (7)	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge (11)	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log (12)	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0
LVQ2 (13)	$\min\left(M, \max(0, E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	0
MCE (15)	$\left(1 + e^{-\left(E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)\right)}\right)^{-1}$	> 0
square-square (16)	$E(W, Y^i, X^i)^2 - \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2$	m
square-exp (17)	$E(W, Y^i, X^i)^2 + \beta e^{-E(W,\bar{Y}^i,X^i)}$	> 0
NLL/MMI (23)	$E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0
MEE (27)	$1 - e^{-\beta E(W,Y^i,X^i)} / \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0

A tutorial on Energy Based Learning — Yann LeCun, Sumit Chopra, Raia Hadsell, Marc' Aurelio Ranzato and Fu Jie Huang

Boltzmann-Gibbs Distribution

Turning energy functions into probabilities

Energy of a sample x

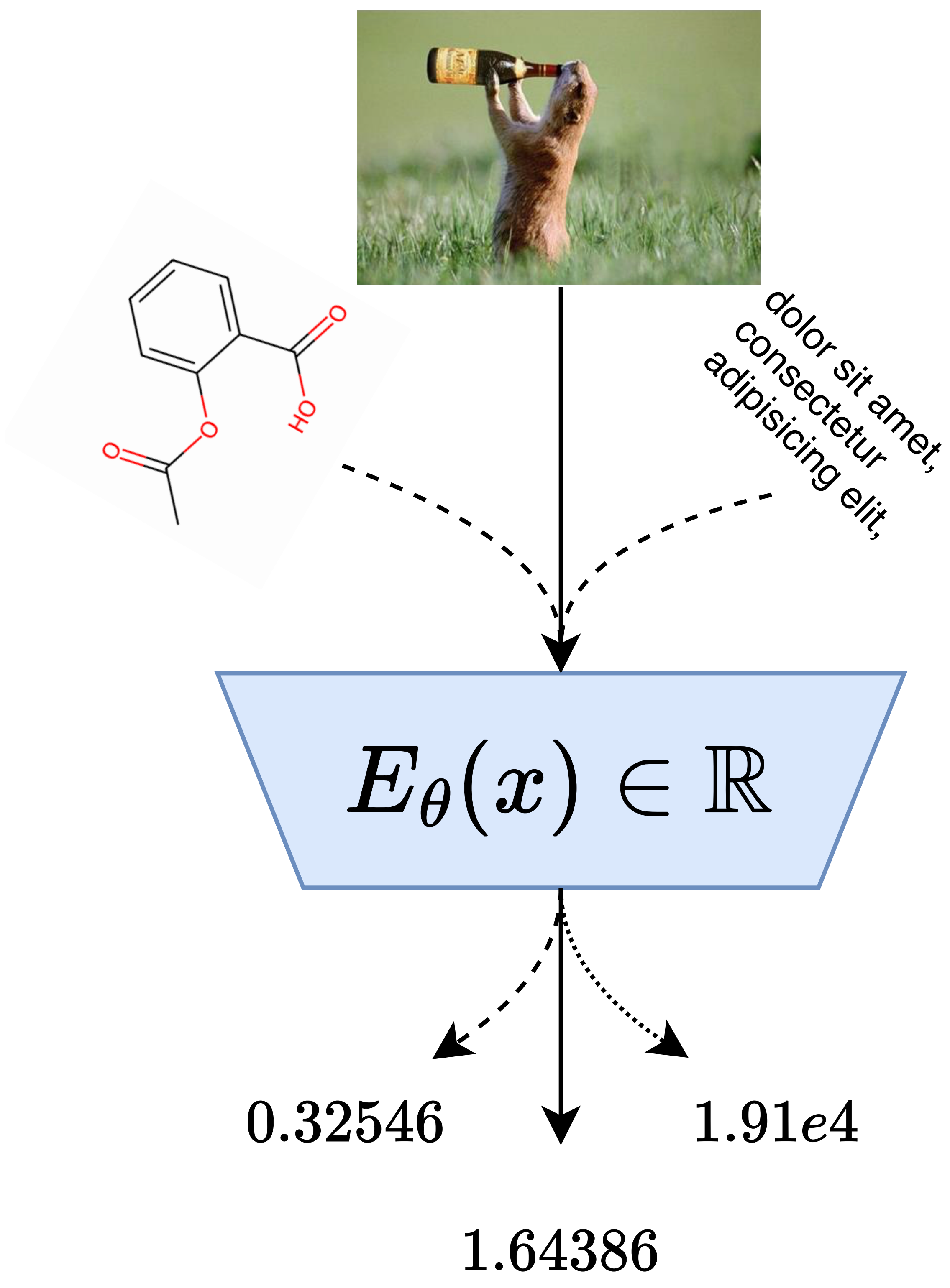
Likelihood of a sample

$$p_\theta(x) = \frac{\exp\{-E_\theta(x)\}}{Z_\theta}$$

$$Z_\theta = \int_x \exp\{-E_\theta(x)\}dx$$

Again, intractable for large spaces

Structured Input



Scalar energy value

Training Boltzmann Machines

With Maximum Likelihood Estimation

1. Rewrite the Boltzmann-Gibbs likelihood as the log-likelihood

$$L_\theta = \log p_\theta(x) = -\log(Z_\theta) - E_\theta(x)$$

2. Therefore, the gradient:

$$\nabla L_\theta = \nabla_\theta \log p_\theta(x) = \underbrace{-\nabla_\theta \log(Z_\theta)}_{\text{unfeasible}} \underbrace{-\nabla_\theta E_\theta(x)}_{\text{easy}}$$

3. With tedious manipulations, it can be proved that:

$$\nabla_\theta \log Z_\theta = \mathbb{E}_{x \sim p_\theta(x)}\left[-\nabla_\theta E_\theta(x)\right]$$

Tedious manipulations of the partition function

$$\nabla_\theta \log Z_\theta = \nabla_\theta \log \int \exp\{-E_\theta(x)\} dx$$

(Chain rule) $$= \left(\int \exp\{-E_\theta(x)\} dx \right)^{-1} \int \nabla_\theta \exp\{-E_\theta(x)\} dx$$

(Chain rule) $$= \left(\int \exp\{-E_\theta(x)\} dx \right)^{-1} \int \exp\{-E_\theta(x)\} \left(-\nabla_\theta E_\theta(x) \right) dx$$

$$= \int \underbrace{\frac{\exp\{-E_\theta(x)\}}{Z_\theta}}_{p_\theta(x)} \left(-\nabla_\theta E_\theta(x) \right) dx$$

(Def.) $$= \mathbb{E}_{x \sim p_\theta(x)} \left[-\nabla_\theta E_\theta(x) \right]$$

Contrastive Divergence ☯️

Optimizing the likelihood while dreaming 😴

Mini-batch Version

$$\nabla_\theta L_{CD} = \frac{1}{N}\sum_{i=1}^{N}\left[-\nabla_\theta E_\theta(x_i) + \nabla_\theta E_\theta(x_i^{(k)})\right]$$

In summary, the gradient of the likelihood is

$$\nabla_\theta \log p_\theta(x) = \mathbb{E}_{x\sim p_D}\left[-\nabla_\theta E_\theta(x)\right] - \mathbb{E}_{x\sim p_\theta(x)}\left[\nabla_\theta E_\theta(x)\right]$$

$\longrightarrow$

Practically, we approximate it as

$$\nabla_\theta \log p_\theta(x) \approx -\nabla_\theta E_\theta(x \sim D) - \nabla_\theta E_\theta(x \sim p_\theta)$$

"Wake phase" "Dream phase"

Learning is a two-steps process, where we compute the energy of the dataset samples (the wake phase) and compare it to the "negative" samples during the dream phase

However, while we can easily have access to dataset samples x_D, how can we obtain samples $x \sim p_\theta$?

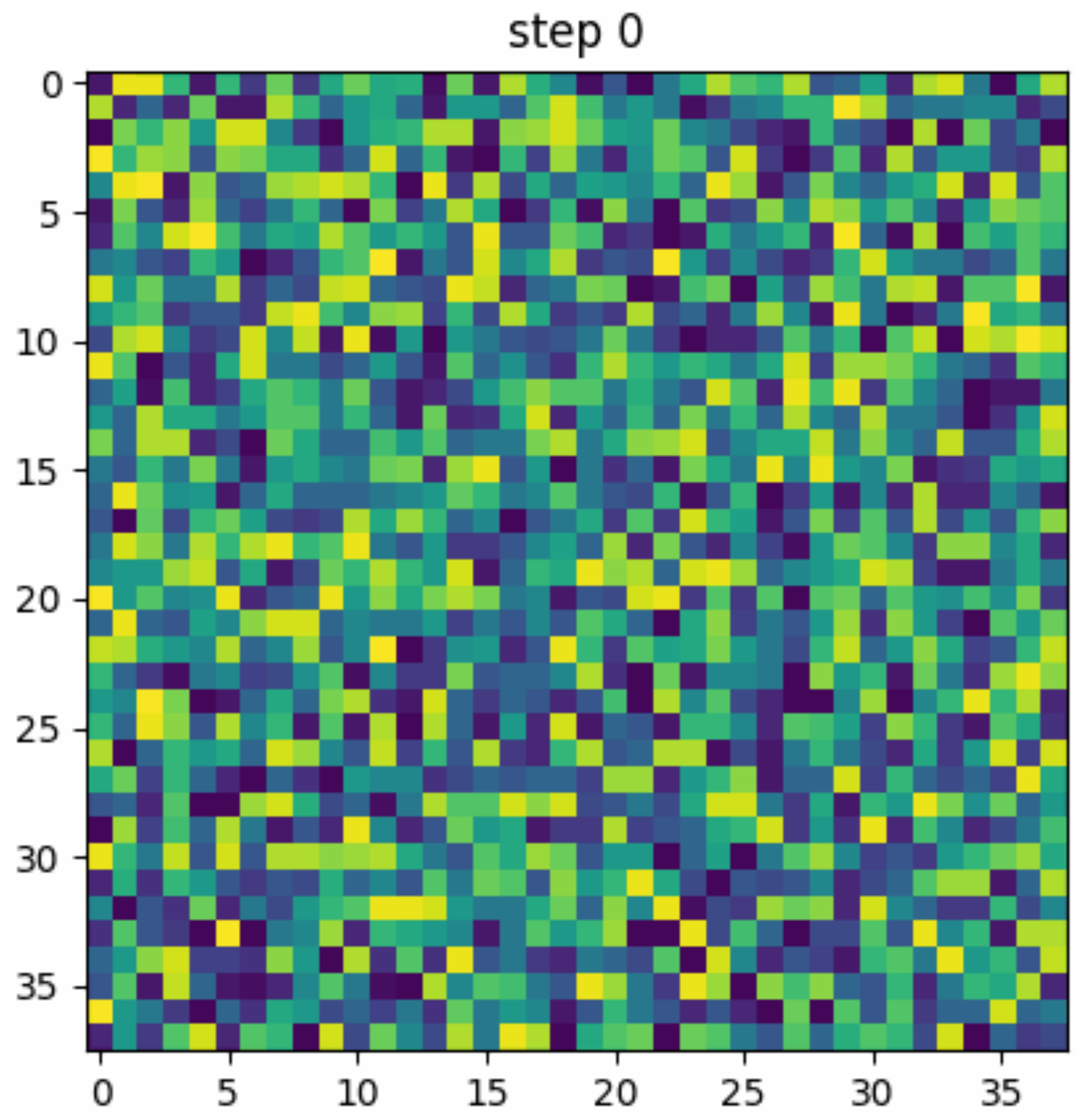
Sampling from EBMs

Langevin Dynamics MCMC denoising process

We can get samples x from the model's distribution $p_\theta(x)$ with Langevin Dynamics

$$x_t = x_{t-1} - \epsilon \nabla_x E_\theta(x_{t-1}) + \omega$$

Where $\omega \sim \mathcal{N}(0,\epsilon)$ and ϵ is the step size, and x_0 is typically drawn from Gaussian noise



As long as $\epsilon \to 0$ is sufficiently small and $t \to \infty$ the chain will converge to the true distribution

The contrastive Divergence Algorithm 🎭

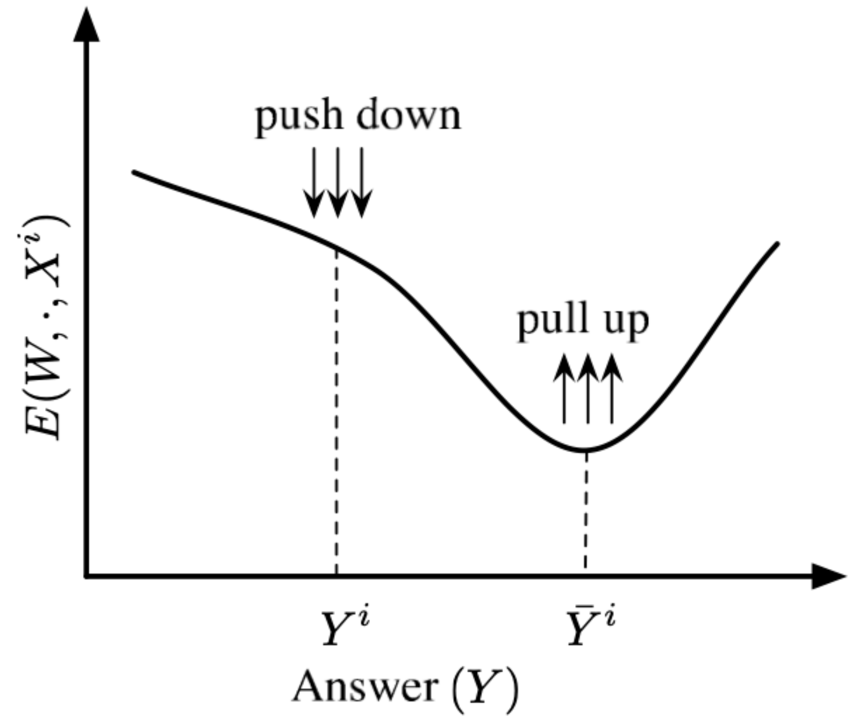
Algorithm 1 Outline of Contrastive Divergence Algorithm

Set k, the number of sampling steps, sufficiently long to allow the Markov Chain to converge
while not converged **do**
  $\mathbf{x}^+ \leftarrow$ sample from the dataset ▷ Positive phase
  $g^+ \leftarrow \nabla_\theta \log p_\theta(x^+)$
  $\mathbf{x}^- \leftarrow$ initial sample typically drawn from noise
  for $j = 1$ to k **do**
    $\mathbf{x}^- \leftarrow$ sample from the model distribution
  end for
  $g^- \leftarrow \nabla_\theta \log p_\theta(\mathbf{x}^-)$ ▷ Negative phase
  $g \leftarrow g^+ - g^-$
  Optimize weights θ using g and a gradient-based optimizer
end while

$$\boxed{\nabla_\theta \log p_\theta(x) \approx \nabla_\theta E_\theta(x_D) - \nabla_\theta E_\theta(x_{p_\theta})}$$

Wake 😀

Dream 😴



Why you should (or should not) use EBMs

- Pros ✅
  - Simplicity: the EBM is the only object that needs to be designed
  - Less parameters: EBM is the only trained object, and it requires less model parameters than approaches that use multiple networks (e.g., VAEs, GANs)
  - Implicitly learning the data distribution, and not constrained to hidden spaces manifolds

- Cons ❌
  - Training is often unstable and very susceptible to optimization strategies
  - CD algorithm is a crude approximation of the gradient (in fact, it does not follow the gradient of any function)
  - High mixing times for MCMC
  - High sensitive to hyperparameters choice

Examples

Restricted Boltzmann Machines

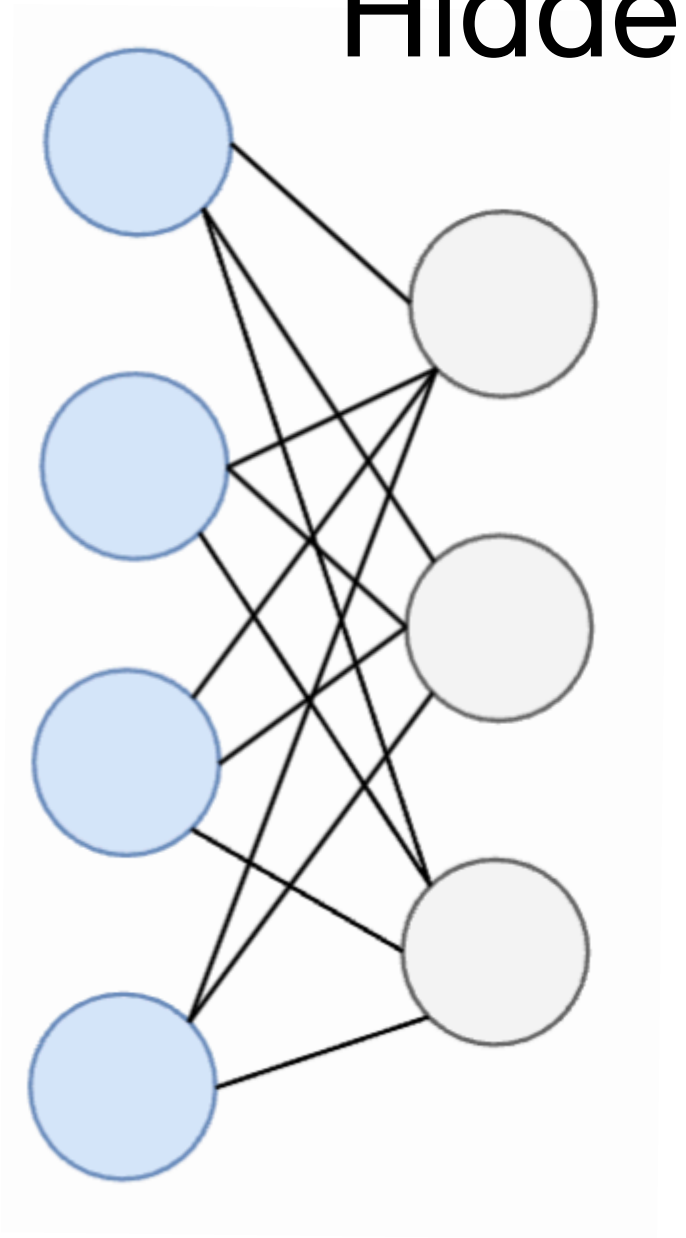
- A particular class of models that defines the joint distribution of two set of (binary) random variables: visible and hidden:

$$p(\mathbf{v}, \mathbf{h}) = \frac{\exp\{-E(\mathbf{v}, \mathbf{h})\}}{Z}$$

- The energy of the units is given by the following term

$$E(\mathbf{v}, \mathbf{h}) = -v^T b - c^T h - v^T \mathbf{W} h$$

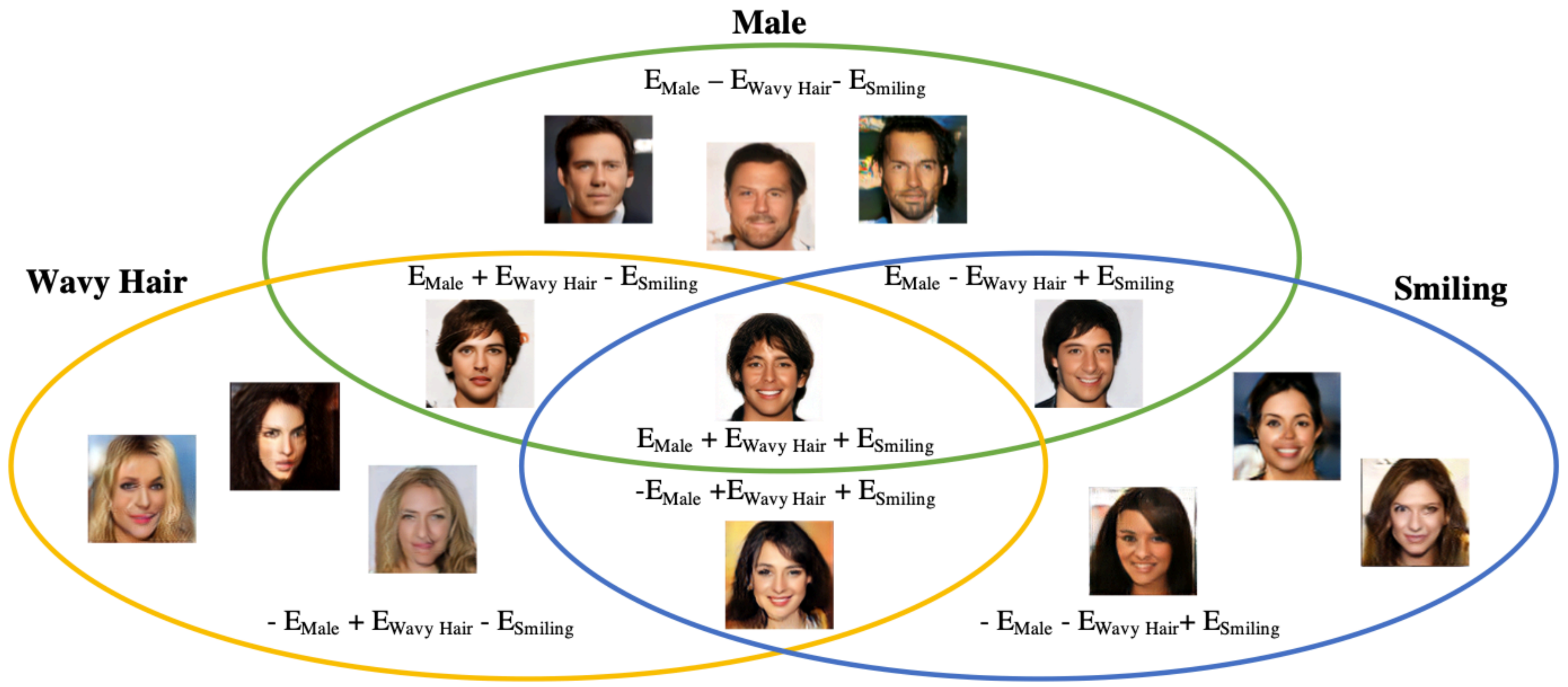
Hidden



Visible

Composing energy functions +

- EBMs present nice compositional properties
- In other words, given a set of k independently trained EBMs $E(x \mid c_1), E(x \mid c_2), \ldots E(x \mid c_k)$ it is possible to compose them and sample from the combination of the concepts
- Common combination include concept conjunction, negation and disjunction



$$p(x|c_1 \text{ and } c_2, \ldots, \text{ and } c_i) = \prod_i p(x|c_i) \propto e^{-\sum_i E(x|c_i)} \longrightarrow \tilde{\mathbf{x}}^k = \tilde{\mathbf{x}}^{k-1} - \frac{\lambda}{2} \nabla_{\mathbf{x}} \sum_i E_\theta(\tilde{\mathbf{x}}^{k-1}|c_i) + \omega^k.$$

Product of Experts

Sampling

Compositional Visual Generation with Energy Based Models — Yilun Du, Shuang Li, Igor Mordatch

Composing energy functions

Figure 3: Combinations of different attributes on CelebA via concept conjunction. Each row adds an additional energy function. Images on the first row are conditioned on young, while images on the last row are conditioned on young, female, smiling, and wavy hair.

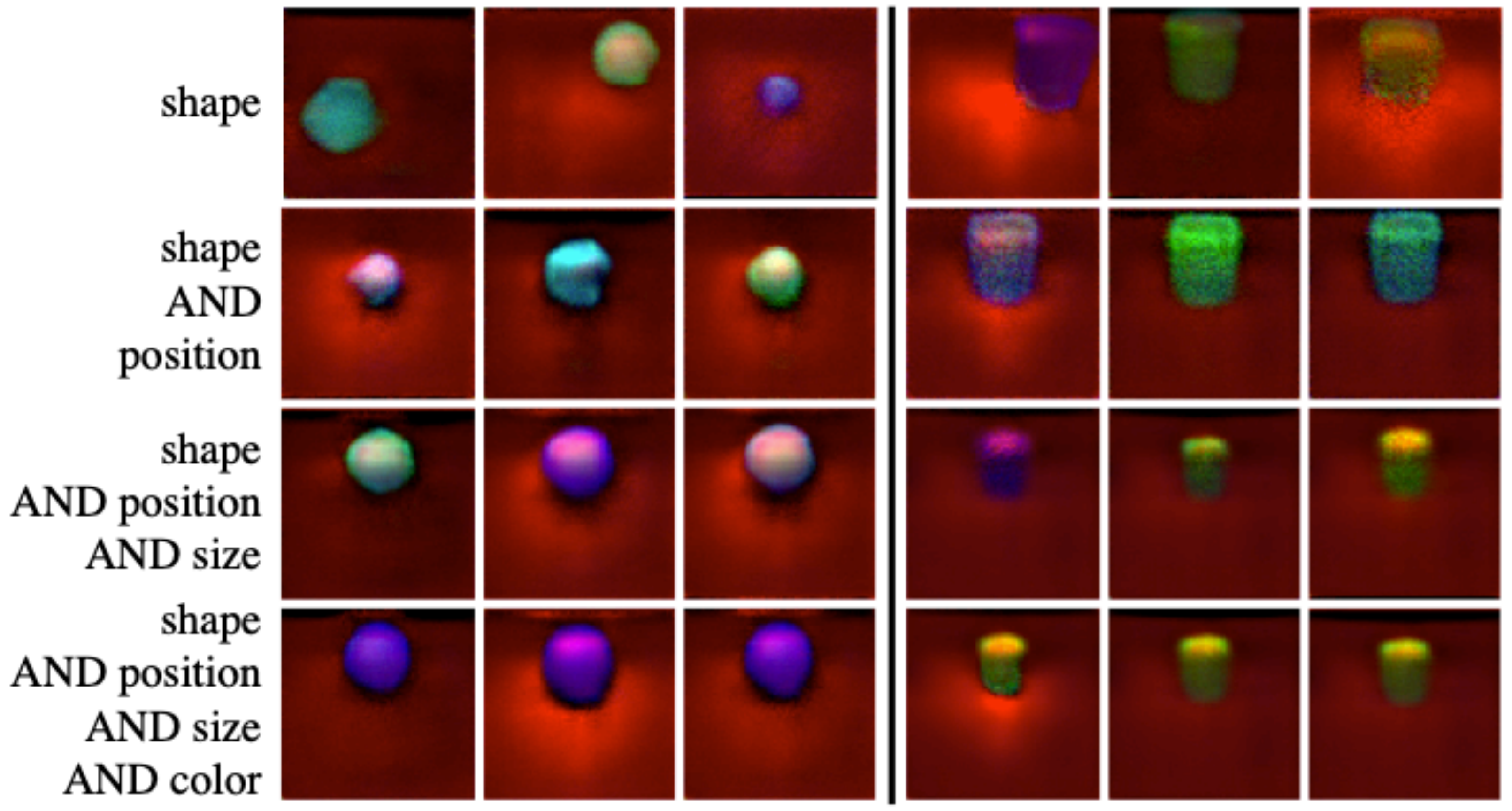


Figure 4: Combinations of different attributes on MuJoCo via concept conjunction. Each row adds an additional energy function. Images on the first row are only conditioned on shape, while images on the last row are conditioned on shape, position, size, and color. The left part is the generation of a sphere shape and the right is a cylinder.

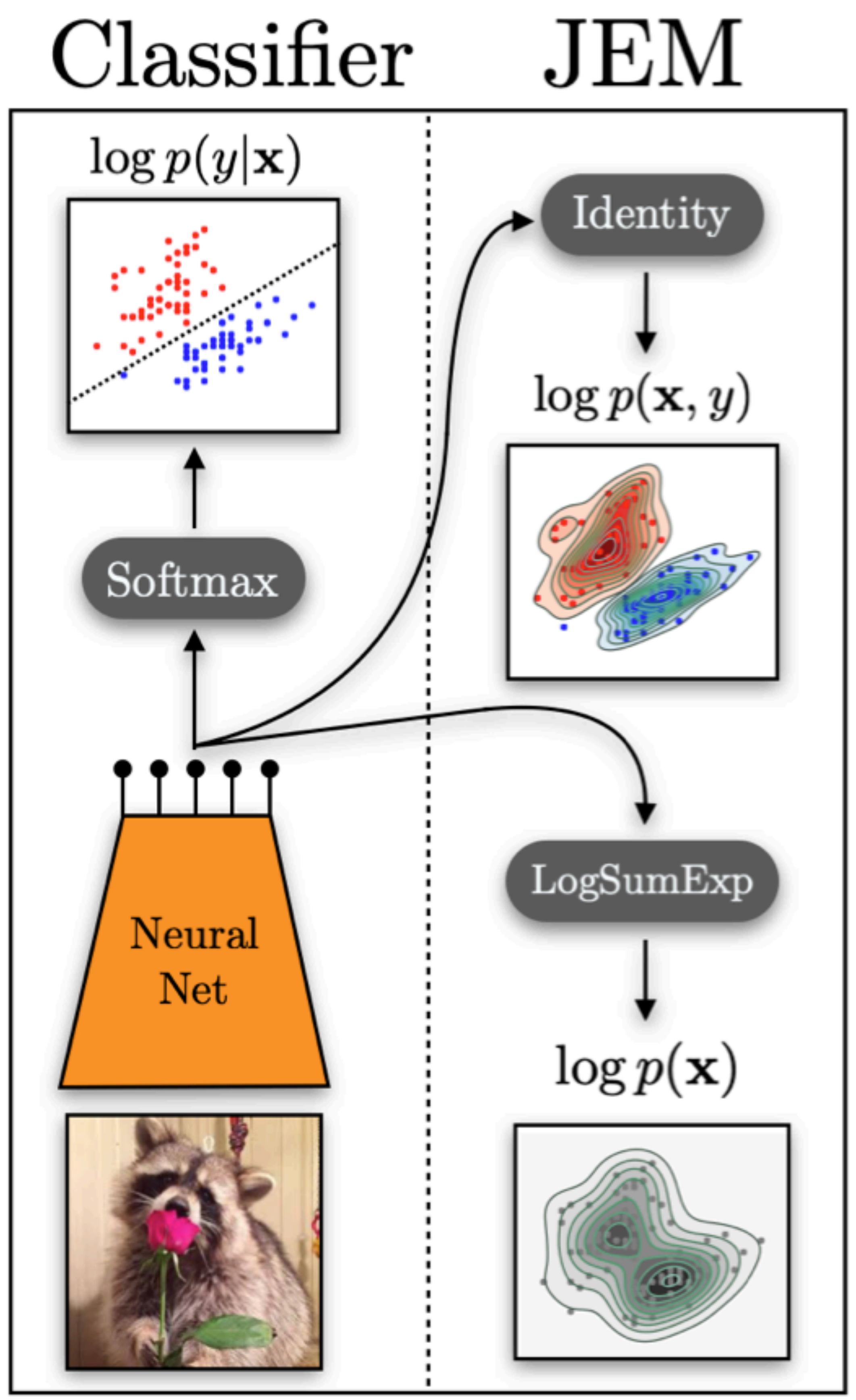
Compositional Visual Generation with Energy Based Models — Yilun Du, Shuang Li, Igor Mordatch

Your classifier is secretly an EBM!

- Reinterpretation of the "standard" classifiers $p(y \mid x)$
- In this setting, the standard class probabilities can be computed, as well as unnormalized values of $p(x)$
- Significant better results in classification, robustness and calibration w.r..t. normal discriminative models

$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{\sum_y \exp\left(f_\theta(\mathbf{x})[y]\right)}{Z(\theta)}$$

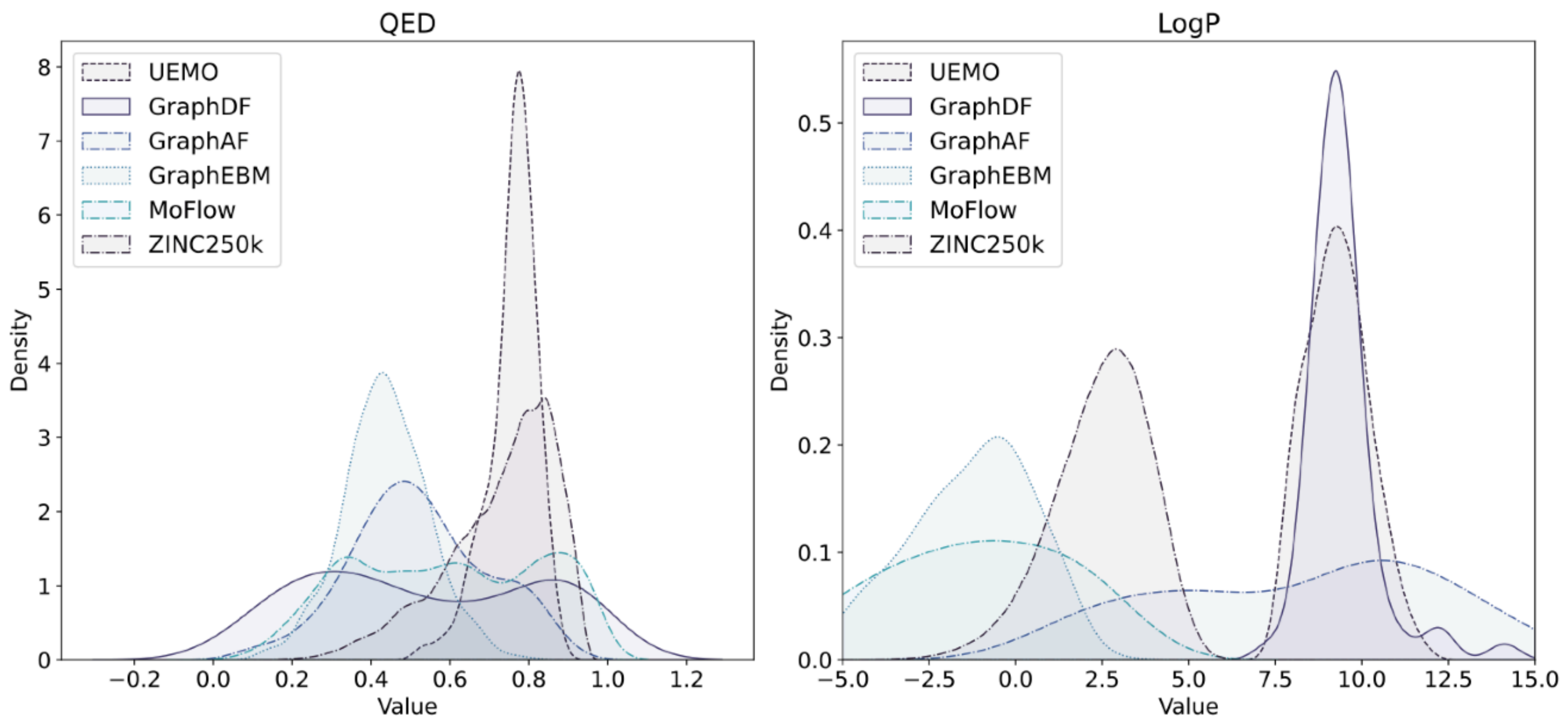
$$E_\theta(\mathbf{x}) = -\mathrm{LogSumExp}_y(f_\theta(\mathbf{x})[y]) = -\log \sum_y \exp(f_\theta(\mathbf{x})[y])$$



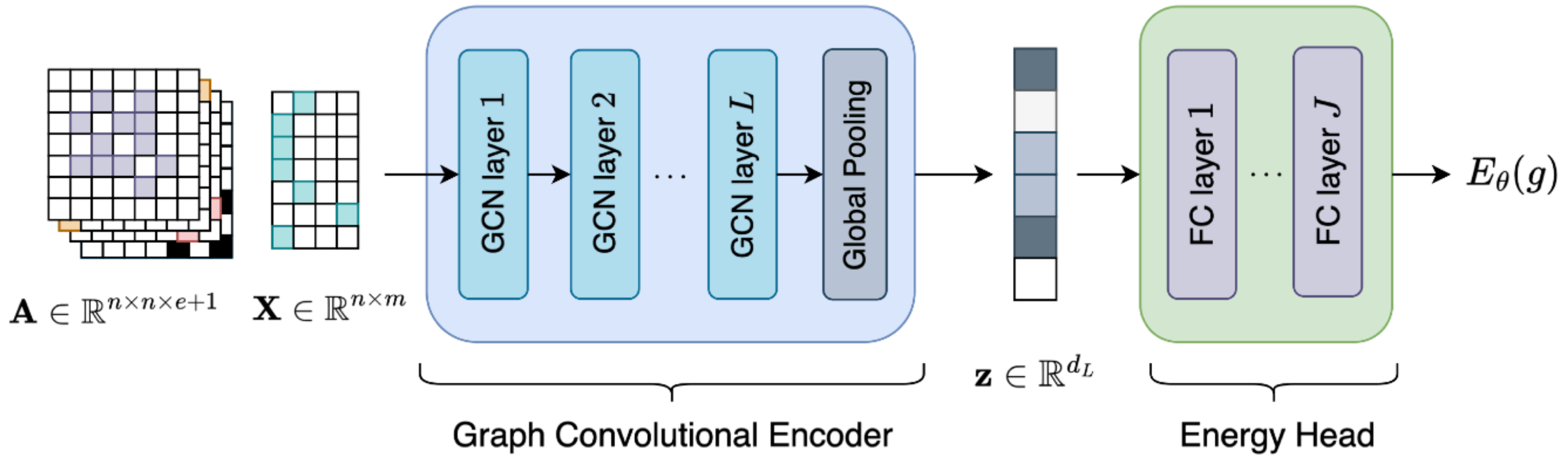
Your classifier is secretly an Energy Based Model and you should treat it like one — Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, Kevin Swersk

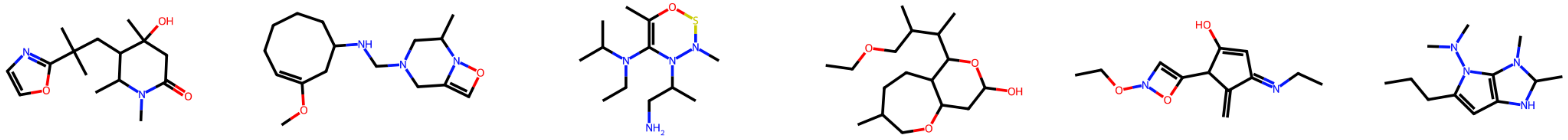
EBMs for Graph Generation

Implicit chemical property optimization



- Smaller EBMs allow for smaller training datasets
- We can train a single expert on a subset of molecules satisfying our desired objective
- Implicitly learning to generate new graphs having the original dataset properties





Towards Efficient Molecular Property Optimization with Graph Energy Based Models —Luca Miglior, Lorenzo Simone, Marco Podda and Davide Bacciu

Conclusion

Why should you give them a try 😔

- Unified framework for learning, handling seamlessly supervised, unsupervised and self supervised tasks through Energy minimization
- No more need for fixed output structures
- Focus on the energy landscaper rather than explicit variables correlations and probabilities
- Still open questions! Reduce sampling times, integrating latent variables efficiently in BMs and challenges to address the issues deriving from the intractability of the partition function

Thank You!