PML 11. Energy-Based Models. Chapter 24, Probabilistic Machine Learning: Advanced Topics

Probabilistic Machine Learning Reading Group

Víctor Gallego

April 8, 2026

Komorebi AI, ICMAT

Contents

Contents

Why Energy-Based Models?

The generative–model zoo so far:

- VAEs, autoregressive models, normalizing flows: all *directed* graphical models.
- They generate data one step at a time, using locally normalized distributions.

Sometimes it is easier to specify *constraints* than a generation process.

- Example: proteins that are *thermally stable* **and** *bind to a target receptor*.
- Each constraint is easy to model; combining them in a directed model is hard.

$\Rightarrow$ **Idea:** specify an *unnormalized* distribution via an energy function.

EBM: Definition

An **Energy-Based Model** defines a Gibbs distribution:

$$\boxed{p_{\boldsymbol{\theta}}(\boldsymbol{x}) \;=\; \frac{\exp\bigl(-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\bigr)}{Z_{\boldsymbol{\theta}}}}$$

- $\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})$: the **energy function**, *any* function $\mathbb{R}^d \to \mathbb{R}$ parameterized by $\boldsymbol{\theta}$.
- $Z_{\boldsymbol{\theta}} = \int \exp\bigl(-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\bigr)\,\mathrm{d}\boldsymbol{x}$: the **partition function** (intractable!).

Advantage: $\mathcal{E}_{\boldsymbol{\theta}}$ is *unrestricted*, any neural network that outputs a scalar.

No invertibility, no autoregressive ordering, no tractable Jacobian required.

Challenge: we can never compute $Z_{\boldsymbol{\theta}}$ (or even its gradient) exactly.

Example: Product of Experts

Suppose we have two "expert" distributions $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$.

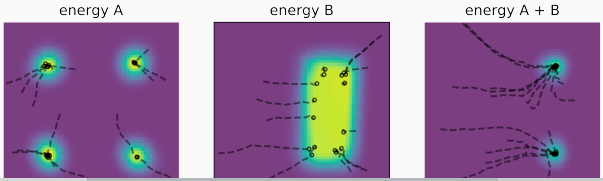
Product of Experts (PoE) [Hinton, 2002]:

$$p_{12}(\boldsymbol{x}) = \frac{1}{Z_{12}} p_1(\boldsymbol{x})\, p_2(\boldsymbol{x}).$$

If each expert is an EBM, the combined model is also an EBM with

$$\mathcal{E}_{12}(\boldsymbol{x}) = \mathcal{E}_1(\boldsymbol{x}) + \mathcal{E}_2(\boldsymbol{x}).$$

Each energy acts as a **soft constraint** on the data.

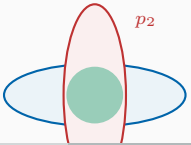


Product of Experts vs. Mixture of Experts

Product of Experts (PoE):

$$p_{\text{PoE}}(\boldsymbol{x}) = \frac{1}{Z} \prod_m p_m(\boldsymbol{x})$$

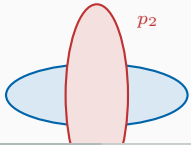
- High probability only where all experts agree
- Energies **add**: $\mathcal{E}_{12} = \mathcal{E}_1 + \mathcal{E}_2$
- Result is **sharper** than any single expert
- Natural for EBMs



Mixture of Experts (MoE):

$$p_{\text{MoE}}(\boldsymbol{x}) = \sum_m \pi_m \, p_m(\boldsymbol{x})$$

- High probability where any expert assigns mass
- Densities **average**: covers the union
- Result is **broader** than any single expert
- Natural for directed models



The Central Challenge

Everything about EBMs is easy...

except for the partition function $Z_{\boldsymbol{\theta}}$.

What $Z_{\boldsymbol{\theta}}$ blocks:

- Evaluating $p_{\boldsymbol{\theta}}(\boldsymbol{x})$
- Computing the likelihood
- Drawing exact samples

Strategies (this talk):

1. Maximum likelihood + MCMC
2. Score matching
3. Adversarial training
4. Other methods (NCE, Stein, ...)

Contents

MLE for EBMs: Setup

Standard MLE: maximize expected log-likelihood

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{D}}(\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] = -\mathrm{D}_{\mathrm{KL}}\big(p_{\mathcal{D}}(\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{x})\big) + \text{const.}$$

MLE for EBMs: Setup

Standard MLE: maximize expected log-likelihood

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{D}}(\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] = -\mathrm{D}_{\mathrm{KL}}\big(p_{\mathcal{D}}(\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{x})\big) + \text{const.}$$

Problem: $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log Z_{\boldsymbol{\theta}}$. We can't compute $\log Z_{\boldsymbol{\theta}}$.

MLE for EBMs: Setup

Standard MLE: maximize expected log-likelihood

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{D}}(\boldsymbol{x})}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] = -\mathrm{D}_{\mathrm{KL}}\big(p_{\mathcal{D}}(\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{x})\big) + \text{const.}$$

Problem: $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log Z_{\boldsymbol{\theta}}$. We can't compute $\log Z_{\boldsymbol{\theta}}$.

Main insight: we don't need $Z_{\boldsymbol{\theta}}$ itself, only its *gradient* w.r.t. $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underbrace{-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\text{easy (autodiff)}} - \underbrace{\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}}_{\text{hard}}$$

The Log-Partition Gradient

Theorem. The gradient of $\log Z_{\boldsymbol{\theta}}$ equals a model expectation:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})}\big[-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big].$$

The Log-Partition Gradient

Theorem. The gradient of $\log Z_{\boldsymbol{\theta}}$ equals a model expectation:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})}\big[-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big].$$

Proof sketch:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} &= \frac{1}{Z_{\boldsymbol{\theta}}} \int \exp\big(-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\big(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\, \mathrm{d}\boldsymbol{x} \\ &= \int p_{\boldsymbol{\theta}}(\boldsymbol{x})\big(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\, \mathrm{d}\boldsymbol{x} = \mathbb{E}_{p_{\boldsymbol{\theta}}}\big[-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big].\end{aligned}$$

The Log-Partition Gradient

Theorem. The gradient of $\log Z_{\boldsymbol{\theta}}$ equals a model expectation:

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})}\big[-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big].$$

Proof sketch:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} &= \frac{1}{Z_{\boldsymbol{\theta}}} \int \exp\big(-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\big(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\, \mathrm{d}\boldsymbol{x} \\ &= \int p_{\boldsymbol{\theta}}(\boldsymbol{x})\big(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\, \mathrm{d}\boldsymbol{x} = \mathbb{E}_{p_{\boldsymbol{\theta}}}\big[-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big].\end{aligned}$$

Full gradient of log-likelihood:

$$\nabla_{\boldsymbol{\theta}} \ell = \underbrace{-\mathbb{E}_{p_{\mathcal{D}}}\big[\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big]}_{\text{data term (“clamped”)}} + \underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}}}\big[\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big]}_{\text{model term (“unclamped”)}}$$

Push data energies **down**, push model-sample energies **up**.

Sampling with Langevin MCMC

To estimate the model expectation, we need samples $\tilde{\boldsymbol{x}}_s \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Observation, the score function:

$$\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) \underbrace{- \nabla_{\boldsymbol{x}} \log Z_{\boldsymbol{\theta}}}_{=0} = -\nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}).$$

The score is *free*: no partition function involved!

Sampling with Langevin MCMC

To estimate the model expectation, we need samples $\tilde{\boldsymbol{x}}_s \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Observation, the score function:

$$\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) \underbrace{- \nabla_{\boldsymbol{x}} \log Z_{\boldsymbol{\theta}}}_{=0} = -\nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}).$$

The score is *free*: no partition function involved!

Langevin MCMC (overdamped Langevin diffusion, discretized):

$$\boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k \underbrace{- \frac{\epsilon^2}{2} \nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^k)}_{\text{gradient step}} + \underbrace{\epsilon \boldsymbol{z}^k}_{\text{noise}}, \qquad \boldsymbol{z}^k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}).$$

As $\epsilon \to 0$ and $K \to \infty$, the iterates converge to $p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Persistent MCMC-SGD (PCD)

Running MCMC to convergence at every step is expensive. [Younes, 1989; Tieleman, 2008]: parameters change slowly, so the chain from step $t-1$ is still a good starting point at step t.

Algorithm (Persistent MCMC-SGD):

1. Initialize parameters $\boldsymbol{\theta}$ and chains $\tilde{\boldsymbol{x}}_{1:S}$ randomly.
2. For each training step t:
   - Compute data gradients: $\boldsymbol{g}_b = \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}_b)$
   - **Continue** each chain $\tilde{\boldsymbol{x}}_s$ for N Langevin steps (don't restart!)
   - Compute model gradients: $\tilde{\boldsymbol{g}}_s = \nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_s)$
   - Update: $\boldsymbol{\theta} \mathrel{+}= \eta\big(-\frac{1}{B}\sum \boldsymbol{g}_b - \frac{1}{S}\sum \tilde{\boldsymbol{g}}_s\big)$

Replay buffer variant [Du & Mordatch, 2019]:

- Store historical MCMC states in a buffer
- Init new chains by sampling from buffer (95%) or noise (5%)
- Encourages diversity, avoids mode collapse

Caveat: truncated MCMC $\Rightarrow$ biased gradients. Corrections exist (coupled MCMC, entropy terms).

Historical Note: CD and RBMs

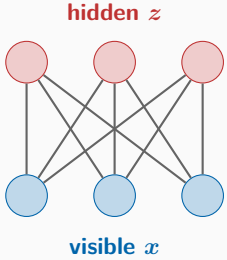
Contrastive Divergence [Hinton, 2002]:

- Initialize chain at the *data point* $\boldsymbol{x}_n$, run only T steps (often $T=1$)
- Minimizes $\mathrm{CD}_T = \mathrm{D}_{\mathrm{KL}}(p_0 \| p_\infty) - \mathrm{D}_{\mathrm{KL}}(p_T \| p_\infty)$
- Biased but fast; historically important for training **Restricted Boltzmann Machines**

RBMs: latent-variable EBM with bipartite structure and block Gibbs sampling:

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}) = \sum_{d,k} x_d z_k W_{dk} + \sum_d x_d b_d + \sum_k z_k c_k$$

CD-1 gradient: $\nabla_{\boldsymbol{W}} \ell \approx \boldsymbol{x}_n \boldsymbol{\mu}_n^\top - \boldsymbol{x}'_n (\boldsymbol{\mu}'_n)^\top$ (Hebbian − anti-Hebbian).



PCD superseded CD: persistent chains explore better and don't need the bipartite trick.

Contents

Score Matching: Idea

Observation: if two log-pdfs have the *same first derivatives* everywhere, they are equal (up to normalization).

Define the **(Stein) score function**:

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) \triangleq \nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}). \qquad \text{(no } Z_{\boldsymbol{\theta}}\text{!)}$$

Score Matching: Idea

Observation: if two log-pdfs have the *same first derivatives* everywhere, they are equal (up to normalization).

Define the **(Stein) score function**:

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) \triangleq \nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}). \qquad \text{(no } Z_{\boldsymbol{\theta}}\text{!)}$$

Score Matching [Hyvärinen, 2005] minimizes the **Fisher divergence**:

$$\boxed{D_F\big(p_{\mathcal{D}} \,\|\, p_{\boldsymbol{\theta}}\big) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right\|^2\right].}$$

Problem: the data score $\nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}(\boldsymbol{x})$ is unknown!

Hyvärinen's Trick: Integration by Parts

Under regularity conditions, the Fisher divergence can be rewritten as:

$$D_F(p_{\mathcal{D}} \,\|\, p_{\boldsymbol{\theta}}) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\left[\frac{1}{2}\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})\right\|^2 + \operatorname{tr}\left(\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right] + \text{const}$$

- The unknown $\nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}$ has **disappeared**!
- Only involves the model score $\boldsymbol{s}_{\boldsymbol{\theta}}$ and its Jacobian.
- The constant doesn't depend on $\boldsymbol{\theta}$, just drop it.

Hyvärinen's Trick: Integration by Parts

Under regularity conditions, the Fisher divergence can be rewritten as:

$$D_F\big(p_{\mathcal{D}} \,\|\, p_{\boldsymbol{\theta}}\big) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\left[\frac{1}{2}\big\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})\big\|^2 + \operatorname{tr}\big(\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x})\big)\right] + \text{const}$$

- The unknown $\nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}$ has **disappeared**!
- Only involves the model score $\boldsymbol{s}_{\boldsymbol{\theta}}$ and its Jacobian.
- The constant doesn't depend on $\boldsymbol{\theta}$, just drop it.

Downside: computing $\operatorname{tr}(\boldsymbol{J}_{\boldsymbol{x}}\boldsymbol{s}_{\boldsymbol{\theta}})$ costs $O(d^2)$: one backprop per input dimension.

$\Rightarrow$ Only practical for low-dimensional or specially structured energy functions.

Denoising Score Matching (DSM)

[Vincent, 2011] — an elegant workaround to avoid second derivatives entirely.

Idea: corrupt data with noise $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}, \;\; \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$.

The noisy conditional has a *known* score:

$$\nabla_{\tilde{\boldsymbol{x}}} \log q(\tilde{\boldsymbol{x}} \,|\, \boldsymbol{x}) = \frac{\boldsymbol{x} - \tilde{\boldsymbol{x}}}{\sigma^2}.$$

Denoising Score Matching (DSM)

[Vincent, 2011] — an elegant workaround to avoid second derivatives entirely.

Idea: corrupt data with noise $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$.

The noisy conditional has a *known* score:

$$\nabla_{\tilde{\boldsymbol{x}}} \log q(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \frac{\boldsymbol{x} - \tilde{\boldsymbol{x}}}{\sigma^2}.$$

DSM objective:

$$\boxed{\frac{1}{2}\, \mathbb{E}_{q(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left[\left\| \boldsymbol{s}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) - \frac{\boldsymbol{x} - \tilde{\boldsymbol{x}}}{\sigma^2} \right\|^2 \right]}$$

Train the score network to **denoise**: predict the direction back to clean data.

Pro: no Jacobian traces, simple to implement.

Con: learns the score of the *noisy* distribution $q(\tilde{\boldsymbol{x}})$, not $p_{\mathcal{D}}(\boldsymbol{x})$.

Sliced Score Matching (SSM)

[Song et al., 2019] : project scores onto random directions.

Sliced Fisher divergence:

$$D_{SF}(p_{\mathcal{D}} \,\|\, p_{\boldsymbol{\theta}}) = \mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}\, \mathbb{E}_{p(\boldsymbol{v})} \left[\frac{1}{2} \left(\boldsymbol{v}^\top \nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}(\boldsymbol{x}) - \boldsymbol{v}^\top \nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \right)^2 \right]$$

After integration by parts (same trick!), with $p(\boldsymbol{v}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$:

$$D_{SF} = \mathbb{E}_{p_{\mathcal{D}}}\, \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}} \left[\frac{1}{2} \left(\boldsymbol{v}^\top \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right)^2 + \boldsymbol{v}^\top \boldsymbol{J}\, \boldsymbol{v} \right] + C$$

Autodiff theory: the Hessian-vector product $\boldsymbol{J}\boldsymbol{v}$ costs $O(d)$, not $O(d^2)$.

Method	Consistent?	Cost
Basic SM	✓	$O(d^2)$
DSM	$\times$ (noisy target)	$O(d)$
SSM	✓	$O(d)$

Connection: Score Matching $\longleftrightarrow$ Contrastive Divergence

These are more related than they look! [Hyvärinen, 2007a]

Consider 1-step Langevin CD with step size ϵ:

$$\boldsymbol{x}^1 = \boldsymbol{x} - \frac{\epsilon}{2}\nabla_{\boldsymbol{x}}\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) + \sqrt{\epsilon}\,\boldsymbol{z}, \qquad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}).$$

Connection: Score Matching $\longleftrightarrow$ Contrastive Divergence

These are more related than they look! [Hyvärinen, 2007a]

Consider 1-step Langevin CD with step size ϵ:

$$\boldsymbol{x}^1 = \boldsymbol{x} - \frac{\epsilon}{2}\nabla_{\boldsymbol{x}}\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) + \sqrt{\epsilon}\,\boldsymbol{z}, \qquad \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}).$$

Taylor-expanding the CD gradient in ϵ gives:

$$\mathbb{E}_{p_{\mathcal{D}}}\big[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] \;\approx\; \frac{\epsilon^2}{2}\,\nabla_{\boldsymbol{\theta}} D_F\big(p_{\mathcal{D}} \,\|\, p_{\boldsymbol{\theta}}\big) + o(\epsilon^2).$$

Takeaway: contrastive divergence (KL-based) and score matching (Fisher-based) are linked by **de Bruijn's identity**:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{D}_{\mathrm{KL}}\big(q_t \,\|\, p_{\boldsymbol{\theta},t}\big) = -\frac{1}{2}\,D_F\big(q_t \,\|\, p_{\boldsymbol{\theta},t}\big),$$

where q_t, $p_{\boldsymbol{\theta},t}$ are Gaussian-smoothed versions (variance t) of $p_{\mathcal{D}}$, $p_{\boldsymbol{\theta}}$.

Contents

Motivation: Avoiding MCMC with a Learned Sampler

Recall: MLE requires samples $\tilde{\boldsymbol{x}} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ at every gradient step.

Problems with MCMC sampling:

- Multiple MCMC steps per training iteration $\Rightarrow$ expensive
- Requires careful tuning of step size, chain length, ...
- Truncated chains $\Rightarrow$ biased gradients

Motivation: Avoiding MCMC with a Learned Sampler

Recall: MLE requires samples $\tilde{\boldsymbol{x}} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ at every gradient step.

Problems with MCMC sampling:

- Multiple MCMC steps per training iteration $\Rightarrow$ expensive
- Requires careful tuning of step size, chain length, ...
- Truncated chains $\Rightarrow$ biased gradients

Idea: learn an **auxiliary model** $q_{\boldsymbol{\phi}}(\boldsymbol{x})$ that approximates $p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

- Replace MCMC sampling with **direct sampling** from $q_{\boldsymbol{\phi}}$
- $q_{\boldsymbol{\phi}}$ acts as a *variational approximation* to the EBM
- Leads naturally to a **minimax** (adversarial) objective

Deriving the Minimax Objective (I)

Goal: maximize the expected log-likelihood $\mathbb{E}_{p_{\mathcal{D}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]$.

Introduce a variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{x})$ and expand:

$$\begin{aligned}
\mathbb{E}_{p_{\mathcal{D}}}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] &= \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \log Z_{\boldsymbol{\theta}} \\
&= \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \log \int e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}\, \mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \log \int q_{\boldsymbol{\phi}}(\boldsymbol{x})\, \frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})}\, \mathrm{d}\boldsymbol{x}
\end{aligned}$$

Deriving the Minimax Objective (I)

Goal: maximize the expected log-likelihood $\mathbb{E}_{p_{\mathcal{D}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]$.

Introduce a variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{x})$ and expand:

$$\begin{aligned}\mathbb{E}_{p_{\mathcal{D}}}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] &= \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \log Z_{\boldsymbol{\theta}} \\ &= \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \log \int e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x} \\ &= \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \log \int q_{\boldsymbol{\phi}}(\boldsymbol{x})\,\frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

Apply **Jensen's inequality** ($-\log$ is convex):

$$-\log \int q_{\boldsymbol{\phi}}(\boldsymbol{x})\,\frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x} \;\leq\; -\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \log \frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}$$

Deriving the Minimax Objective (II)

Expanding the RHS of Jensen's bound:

$$
\begin{aligned}
-\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \log \frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x} &= -\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \Big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{x}) \Big] \mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\big[\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] \underbrace{- H(q_{\boldsymbol{\phi}})}_{\int q \log q}
\end{aligned}
$$

Deriving the Minimax Objective (II)

Expanding the RHS of Jensen's bound:

$$
\begin{aligned}
-\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \log \frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x} &= -\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \Big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{x}) \Big] \mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\big[\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] \underbrace{- H(q_{\boldsymbol{\phi}})}_{\int q \log q}
\end{aligned}
$$

Combining with the data term, we get an **upper bound on expected log-likelihood**:

$$
\mathbb{E}_{p_{\mathcal{D}}}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] \;\leq\; \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big]}_{\text{energy gap (sampler vs. data)}} \quad \underbrace{- H(q_{\boldsymbol{\phi}})}_{\text{entropy of sampler}}
$$

Deriving the Minimax Objective (II)

Expanding the RHS of Jensen's bound:

$$-\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \log \frac{e^{-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})}}{q_{\boldsymbol{\phi}}(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x} = -\int q_{\boldsymbol{\phi}}(\boldsymbol{x}) \Big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{x}) \Big] \mathrm{d}\boldsymbol{x}$$
$$= \mathbb{E}_{q_{\boldsymbol{\phi}}}\big[\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] \underbrace{- H(q_{\boldsymbol{\phi}})}_{\int q \log q}$$

Combining with the data term, we get an **upper bound on expected log-likelihood**:

$$\mathbb{E}_{p_{\mathcal{D}}}\big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\big] \;\leq\; \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big] - \mathbb{E}_{p_{\mathcal{D}}}\big[-\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x})\big]}_{\text{energy gap (sampler vs. data)}} \quad \underbrace{- H(q_{\boldsymbol{\phi}})}_{\text{entropy of sampler}}$$

Tightness: the bound is tight when $q_{\boldsymbol{\phi}}(\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x})$, i.e. when the sampler perfectly matches the EBM.

The Minimax Game

Minimize the upper bound w.r.t. ϕ, maximize likelihood w.r.t. θ:

$$\max_{\theta} \min_{\phi} \; \mathbb{E}_{q_\phi}\big[\mathcal{E}_\theta(x)\big] - \mathbb{E}_{p_\mathcal{D}}\big[\mathcal{E}_\theta(x)\big] - H(q_\phi)$$

Inner min (over ϕ):

- q_ϕ learns to generate **low-energy** samples
- Entropy $H(q_\phi)$ prevents collapse to a single mode
- $q_\phi \to p_\theta$ at optimality

Outer max (over θ):

- Assign low energy to **real data**
- Assign high energy to **sampler outputs**
- Widens the energy gap

Analogy with GANs (Ch. 26): q_ϕ plays the role of the *generator*, $\mathcal{E}_\theta$ the *discriminator*.
But here $\mathcal{E}_\theta$ is an **explicit energy model**: $p_\theta \propto e^{-\mathcal{E}_\theta}$ defines a proper density.

Implementing q_ϕ: Normalizing Flows vs. Neural Samplers

Requirements for q_ϕ: (1) **fast sampling** (replace MCMC), and (2) **tractable entropy** $H(q_\phi)$.

Normalizing flows [Dai+19]

- Invertible $\boldsymbol{x} = f_\phi(\boldsymbol{z})$, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Exact density via change-of-variables
- $H(q_\phi)$ tractable via $\log|\det \boldsymbol{J}_{f_\phi}|$

✓ Both requirements satisfied

× Invertibility constrains architecture

Neural samplers [KB16; Zha+16]

- Deterministic $\boldsymbol{x} = g_\phi(\boldsymbol{z})$, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Like a GAN generator

✓ Unrestricted architecture

✓ Fast sampling

× $H(q_\phi)$ **intractable**

Implementing q_ϕ: Normalizing Flows vs. Neural Samplers

Requirements for q_ϕ: (1) **fast sampling** (replace MCMC), and (2) **tractable entropy** $H(q_\phi)$.

Normalizing flows [Dai+19]

- Invertible $\boldsymbol{x} = f_\phi(\boldsymbol{z})$, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Exact density via change-of-variables
- $H(q_\phi)$ tractable via $\log|\det \boldsymbol{J}_{f_\phi}|$
- ✓ Both requirements satisfied
- × Invertibility constrains architecture

Neural samplers [KB16; Zha+16]

- Deterministic $\boldsymbol{x} = g_\phi(\boldsymbol{z})$, $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- Like a GAN generator
- ✓ Unrestricted architecture
- ✓ Fast sampling
- × $H(q_\phi)$ **intractable**

Entropy estimation heuristics for neural samplers:

- Mutual information: $H(q_\phi(\boldsymbol{x})) = I(g_\phi(\boldsymbol{z}), \boldsymbol{z}) + H(\boldsymbol{z})$ [Kumar+19]
- Variational lower bounds [Nguyen+10; Nowozin+16]
- With fixed base p_0: replace $-H(q_\phi)$ by $\mathrm{D}_{\mathrm{KL}}(q_\phi \| p_0)$ [Dai+19]

Noisy Neural Samplers [Grathwohl et al., 2020a]

Main idea: add noise to make the entropy gradient tractable.

$$\boxed{\boldsymbol{x} = g_{\boldsymbol{\phi}}(\boldsymbol{z}) + \sigma\,\boldsymbol{\epsilon}, \qquad \boldsymbol{z},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}$$

This defines $q_{\boldsymbol{\phi}}(\boldsymbol{x}) = \int \mathcal{N}\big(\boldsymbol{x} \,\big|\, g_{\boldsymbol{\phi}}(\boldsymbol{z}),\, \sigma^2 \boldsymbol{I}\big)\, p(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}$.

Noisy Neural Samplers [Grathwohl et al., 2020a]

Main idea: add noise to make the entropy gradient tractable.

$$\boxed{\boldsymbol{x} = g_{\boldsymbol{\phi}}(\boldsymbol{z}) + \sigma\,\boldsymbol{\epsilon}, \qquad \boldsymbol{z},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}$$

This defines $q_{\boldsymbol{\phi}}(\boldsymbol{x}) = \int \mathcal{N}\big(\boldsymbol{x} \,\big|\, g_{\boldsymbol{\phi}}(\boldsymbol{z}),\, \sigma^2 \boldsymbol{I}\big)\, p(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}$.

Why this helps:

- The gradient $\nabla_{\boldsymbol{\phi}} H(q_{\boldsymbol{\phi}})$ admits unbiased Monte Carlo estimates
- The noisy sampler is a latent variable model with Gaussian conditionals
- Enables **gradient-based optimization** of the full minimax objective

Noisy Neural Samplers [Grathwohl et al., 2020a]

Main idea: add noise to make the entropy gradient tractable.

$$\boxed{\boldsymbol{x} = g_{\boldsymbol{\phi}}(\boldsymbol{z}) + \sigma\,\boldsymbol{\epsilon}, \qquad \boldsymbol{z},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}$$

This defines $q_{\boldsymbol{\phi}}(\boldsymbol{x}) = \int \mathcal{N}\big(\boldsymbol{x} \,\big|\, g_{\boldsymbol{\phi}}(\boldsymbol{z}),\, \sigma^2 \boldsymbol{I}\big)\, p(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}$.

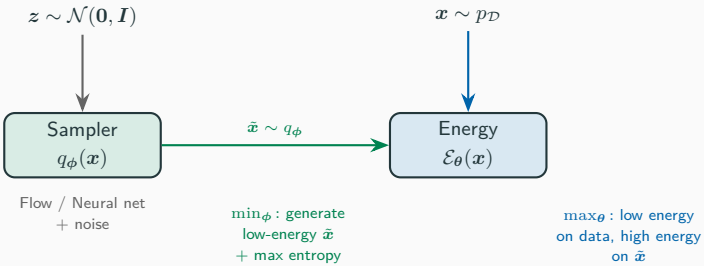
Why this helps:

- The gradient $\nabla_{\boldsymbol{\phi}} H(q_{\boldsymbol{\phi}})$ admits unbiased Monte Carlo estimates
- The noisy sampler is a latent variable model with Gaussian conditionals
- Enables **gradient-based optimization** of the full minimax objective

Connection to normalizing flows:

- A normalizing flow is a *deterministic* invertible sampler ($\sigma = 0$)
- A noisy neural sampler is a *stochastic, non-invertible* sampler ($\sigma > 0$)
- The noise trades off fidelity for architectural freedom and entropy tractability

Adversarial Training: The Full Picture



- **No MCMC** at training time: q_ϕ provides fast approximate samples
- **Explicit density model:** unlike GANs, $\mathcal{E}_\theta$ defines $p_\theta \propto e^{-\mathcal{E}_\theta}$
- Can combine with short-run MCMC for refinement [Xie+18]
- Extends to arbitrary f-divergences beyond KL [Yu+20]

Contents

Other Methods at a Glance

Noise Contrastive Estimation [GH10]
Reduce density estimation to binary classification (data vs. noise p_n). At the optimum $p_{\boldsymbol{\theta}^*} \equiv p_{\mathcal{D}}$. Bonus: $Z_{\boldsymbol{\theta}}$ is a learnable scalar.
With $p_n = p_{\mathcal{D}}(\cdot - \boldsymbol{v})$, $\|\boldsymbol{v}\| \to 0$: recovers SSM.

KL Divergence Tricks [Mov08; SDBD11; Lyu11]
$Z_{\boldsymbol{\theta}}$ cancels in *differences/derivatives* of KL:

- Min. velocity / probability flow: minimize $\frac{\mathrm{d}}{\mathrm{d}t} \mathrm{D}_{\mathrm{KL}}(p_t \| p_{\boldsymbol{\theta}})\big|_{t=0}$
- Min. KL contraction: $\mathrm{D}_{\mathrm{KL}}(p_{\mathcal{D}} \| p_{\boldsymbol{\theta}}) - \mathrm{D}_{\mathrm{KL}}(\Phi\{p_{\mathcal{D}}\} \| \Phi\{p_{\boldsymbol{\theta}}\})$

Stein Discrepancy [GM15; LLJ16]

$$D_{\text{Stein}} := \sup_{\boldsymbol{f} \in \mathcal{F}} \mathbb{E}_{p_{\mathcal{D}}}\left[\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}^\top \boldsymbol{f} + \operatorname{tr}(\nabla_{\boldsymbol{x}} \boldsymbol{f})\right]$$

- Only needs the **score**: no $Z_{\boldsymbol{\theta}}$
- $\mathcal{F}$ = RKHS ball $\Rightarrow$ **KSD**: trace is free, scalable

Common theme: every method finds a clever way to *avoid computing* the intractable partition function $Z_{\boldsymbol{\theta}}$.

Contents

IGEBM: Scaling EBMs to High Dimensions

[Du & Mordatch, NeurIPS 2019] — the theory we've seen, made to work on real images.

Recipe: MLE + Langevin dynamics + three practical tricks:

1. **Replay buffer:** store past MCMC samples; reinitialize 95% of chains from the buffer, 5% from uniform noise. Improves mixing and diversity.
2. **Spectral normalization:** constrain the Lipschitz constant of $\mathcal{E}_{\boldsymbol{\theta}}$ to stabilize Langevin dynamics (smooth energy landscape $\Rightarrow$ stable gradients).
3. **Energy regularization:** weakly penalize $\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^+)^2 + \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^-)^2$ to keep energy magnitudes bounded.

Training:

$$\Delta\boldsymbol{\theta} \;\propto\; \underbrace{\nabla_{\boldsymbol{\theta}}\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^+)}_{\text{push data energy down}} \;-\; \underbrace{\nabla_{\boldsymbol{\theta}}\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^-)}_{\text{push sample energy up}} \;+\; \alpha\,\nabla_{\boldsymbol{\theta}}\big(\mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^+)^2 + \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}^-)^2\big)$$

where $\boldsymbol{x}^+ \sim p_{\mathcal{D}}$ and $\boldsymbol{x}^-$ is obtained via K steps of Langevin dynamics.

IGEBM: Generation Results

Competitive image generation on CIFAR-10 and ImageNet:

Model	IS↑	FID↓
CIFAR-10 Unconditional		
PixelCNN	4.60	65.93
WGAN+GP	6.50	36.4
EBM (single)	**6.02**	**40.58**
SNGAN	8.22	21.7
ImageNet 32×32 Conditional		
PixelIQN	10.18	22.99
EBM (single)	**18.22**	**14.31**

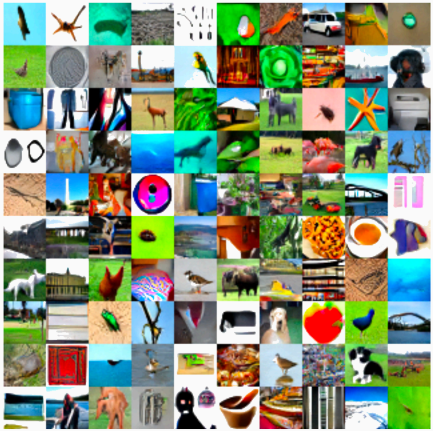


Figure 2: Conditional ImageNet32x32 EBM samples

IGEBM: Beyond Image Generation

EBMs as a versatile building block:

Adversarial robustness

- Conditional EBMs on CIFAR-10 achieve 49.6% accuracy under PGD attack, *without* explicit adversarial training
- Outperform L_∞ SOTA
- Implicit generation "refines" inputs toward high-density regions

Trajectory modeling

- Robot hand manipulation: EBMs capture multimodal transitions

Continual learning

- Split MNIST: **64.99%** (EBM) vs. 40.04% (VAE)
- Replay buffer acts as a natural episodic memory
- No task-specific regularization needed

Some advantages over GANs/VAEs:

✓ Single network (no generator/discriminator)

✓ Adaptive computation (more MCMC = better samples)

JEM: Your Classifier Is Secretly an EBM

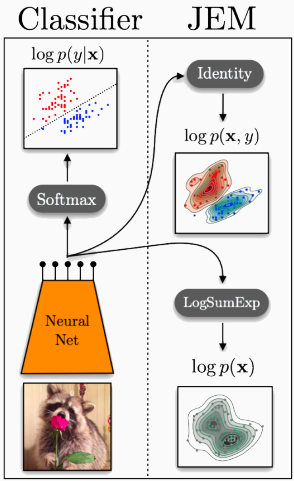
[Grathwohl et al., ICLR 2020] — a **nice reinterpretation** of standard classifiers.

A classifier with logits $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^K$ defines $p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ via softmax. But the *same* logits also define:

Joint: $p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]) \,/\, Z(\boldsymbol{\theta})$
Marginal: $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_y \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]) \,/\, Z(\boldsymbol{\theta})$
$\Rightarrow \; \mathcal{E}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\text{LogSumExp}_y \, f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]$

Punchline: dividing $p_{\boldsymbol{\theta}}(\boldsymbol{x}, y)$ by $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, $Z(\boldsymbol{\theta})$ **cancels** $\Rightarrow$ we recover the standard softmax $p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$!

$\Rightarrow$ Every classifier has a **generative model hidden inside**. Grathwohl et al. propose to *actually train it* — the **Joint Energy-based Model (JEM)**.



JEM: Training and the Hybrid Objective

Factor the joint log-likelihood:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = \underbrace{\log p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})}_{\text{cross-entropy (standard classification)}} + \underbrace{\log p_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\text{EBM training (SGLD / PCD)}}$$

- The discriminative term is **exact**: just standard cross-entropy.
- The generative term uses SGLD with persistent chains and replay buffer (following Du & Mordatch).
- Architecture: Wide ResNet.

JEM: Training and the Hybrid Objective

Factor the joint log-likelihood:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = \underbrace{\log p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})}_{\text{cross-entropy (standard classification)}} + \underbrace{\log p_{\boldsymbol{\theta}}(\boldsymbol{x})}_{\text{EBM training (SGLD / PCD)}}$$

- The discriminative term is **exact**: just standard cross-entropy.
- The generative term uses SGLD with persistent chains and replay buffer (following Du & Mordatch).
- Architecture: Wide ResNet.

CIFAR-10 results: best of both worlds:

Model	**Accuracy** ↑	**IS** ↑	**FID** ↓
Wide-ResNet (disc. only)	95.8%	N/A	N/A
SNGAN (gen. only)	N/A	8.59	25.5
JEM (ours)	**92.9%**	**8.76**	**38.4**

JEM: Calibration, OOD Detection, and Robustness

Calibration (CIFAR-100):

- Baseline WRN: ECE = 22.32%
- **JEM: ECE = 4.87%**
- Nearly perfectly calibrated, *without* post-hoc methods (Platt scaling, etc.)

Adversarial robustness:

- Competitive with explicit adversarial training under L_∞ and L_2 PGD attacks
- MCMC refinement at test time "undoes" adversarial perturbations
- Doesn't confidently classify noise images (unlike standard CNNs)

OOD detection (AUROC):

Score	Glow	JEM
$s = \log p_{\boldsymbol{\theta}}(\boldsymbol{x})$		
SVHN	.05	**.67**
CelebA	.57	**.75**
$s = -\|\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}\|$ (approx. mass)		
SVHN	.27	**.95**
CelebA	.29	**.79**

New OOD score: norm of the score function distinguishes typical-set points from high-density anomalies.

JEM consistently outperforms Glow and IGEBM across all OOD metrics.

Lessons from IGEBM and JEM

What works:

- Langevin dynamics + replay buffer is a practical MCMC recipe for images
- Spectral norm + energy regularization stabilize training
- Classifier logits provide a free energy parameterization
- Joint generative + discriminative training is feasible and beneficial

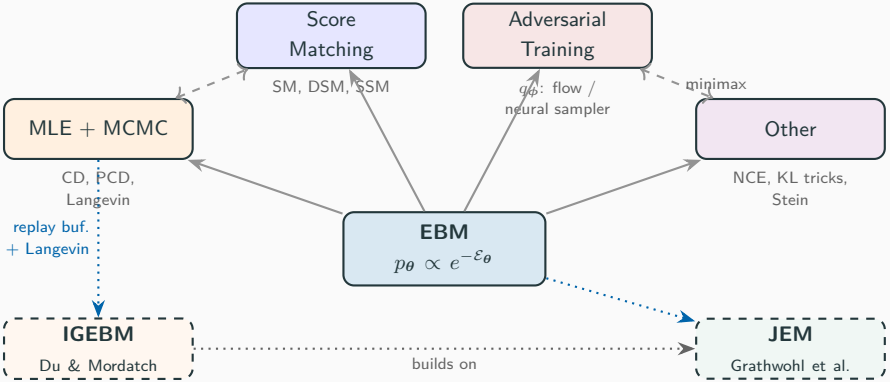
Open challenges:

- Training stability remains fragile (JEM required restarts with lower LR)
- Cannot compute normalized likelihoods: hard to verify learning
- MCMC mixing in high dimensions is slow (biased short-run chains)
- Scaling beyond CIFAR/small ImageNet still difficult

*These papers demonstrated that EBMs are not just theoretically elegant, they are **practically competitive** and bring unique capabilities (compositionality, robustness, calibration) that other generative models lack.*

Contents

The Big Picture



Theory: all methods sidestep intractable Z_θ.

Connections: CD $\leftrightarrow$ SM (de Bruijn), adversarial $\approx$ variational MLE.

Practice: IGEBM scales to images; JEM unifies gen. + disc. learning.

Takeaways

1. **EBMs are maximally flexible** generative models: $\mathcal{E}_{\boldsymbol{\theta}}$ can be *any* scalar-valued network. The price is an intractable $Z_{\boldsymbol{\theta}}$.
2. **MLE + MCMC** gives unbiased gradients in principle, but requires sampling from the model: expensive and tricky (CD, PCD, replay buffers).
3. **Score matching** avoids $Z_{\boldsymbol{\theta}}$ by matching *score functions*. DSM adds noise and learns to denoise; SSM projects onto random directions.
4. **Adversarial training** sidesteps MCMC entirely via a learned sampler $q_{\boldsymbol{\phi}}$ (normalizing flow or noisy neural sampler), yielding a minimax game analogous to GANs but with an explicit energy model.
5. These methods are **deeply connected**: CD $\leftrightarrow$ SM (de Bruijn), NCE $\leftrightarrow$ SSM (local limit), adversarial training $\leftrightarrow$ variational MLE.
6. **EBMs scale to real data** (IGEBM) and can **unify generative & discriminative learning** (JEM).

Thank you!

Questions?

Next Session

Diffusion Models (Ch. 25)

Apr 15, 2026

Alberto Suárez

References i

- Hinton, G.E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*.
- Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *JMLR*.
- Hyvärinen, A. (2007). Some extensions of score matching. *Computational Statistics & Data Analysis*.
- Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*.
- Gutmann, M.U., Hyvärinen, A. (2010). Noise-contrastive estimation. *AISTATS*.
- Song, Y., Garg, S., Shi, J., Ermon, S. (2019). Sliced score matching. *UAI*.
- Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. *ICML*.
- Du, Y., Mordatch, I. (2019). Implicit generation and modeling with energy-based models. *NeurIPS*.

References ii

- Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Swersky, K., Norouzi, M. (2020). Your classifier is secretly an energy based model and you should treat it like one. *ICLR*.
- Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Zemel, R. (2020). Learning the Stein discrepancy for training and evaluating energy-based models without sampling. *ICML*.
- Kim, T., Bengio, Y. (2016). Deep directed generative models with energy-based probability estimation. *arXiv:1606.03439*.
- Dai, Z., et al. (2019). Exponential family estimation via adversarial dynamics embedding. *NeurIPS*.
- Sohl-Dickstein, J., Battaglino, P., DeWeese, M.R. (2011). Minimum probability flow learning. *ICML*.
- Lyu, S. (2011). Unifying non-maximum-likelihood learning objectives with minimum KL contraction. *NeurIPS*.
- Murphy, K.P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.