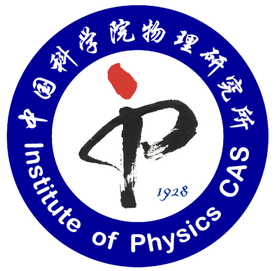
Generative AI & molecular simulation

Lei Wang (王磊)
Institute of Physics, CAS
https://wangleiphy.github.io



Generative AI

Lakeview with geese in the autumn panoramic artist

Generate image

https://huggingface.co/spaces/stabilityai/stable-diffusion

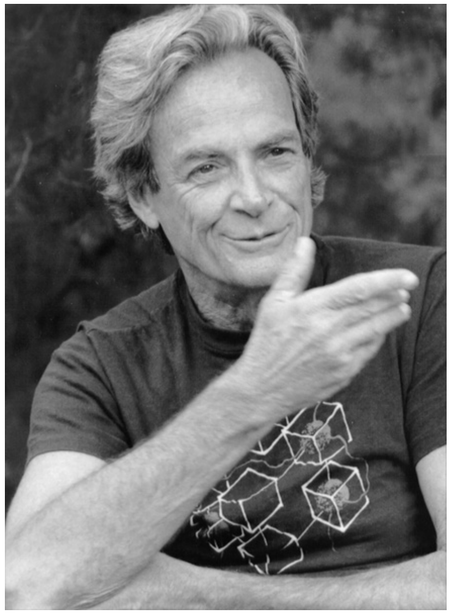
Discriminative learning

$$y = f(\boldsymbol{x})$$

or $p(y \mid \boldsymbol{x})$

Generative learning

$$p(\boldsymbol{x}, y)$$

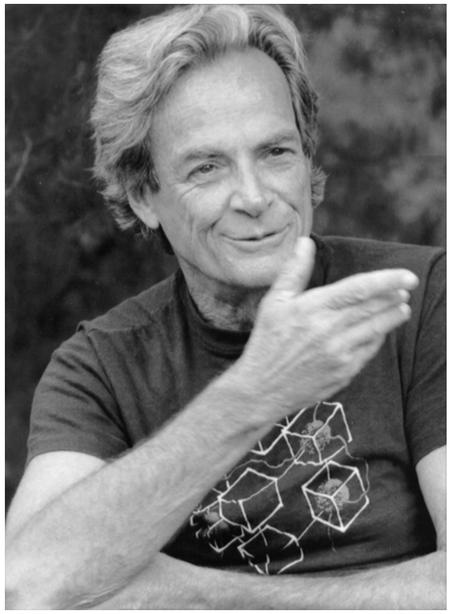


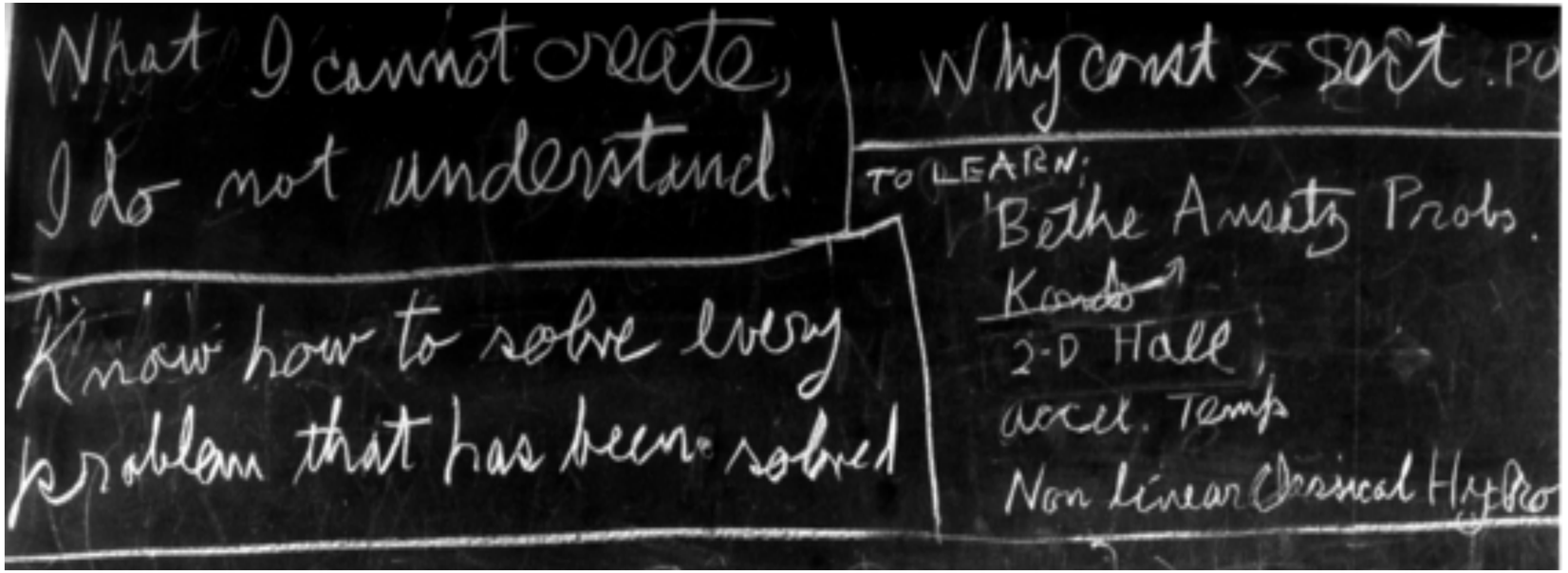
What I cannot create,
I do not understand.

Know how to solve every
problem that has been solved

Why const × sort . PO

TO LEARN:
Bethe Ansatz Probs.
Kondo
2-D Hall
accel. Temp
Non Linear Classical Hydro





“What I can not create, I do not understand”

Generative AI: a new buzz word in silicon valley

A Coming-Out Party for Generative A.I., Silicon Valley's New Craze

A celebration for Stability AI, the start-up behind the controversial Stable Diffusion image generator, represents the arrival of a new A.I. boom.

New York Times

Kevin Roose

Oct. 21, 2022

Protocol

Biz Carson

October 21, 2022

Sequoia's Sonya Huang: The generative AI hype is 'absolutely justified'

She's bullish on generative AI given the "superpowers" it gives humans who work with it.

https://www.sequoiacap.com/article/generative-ai-a-creative-new-world/

by Sonya Huang, Pat Grady and GPT-3

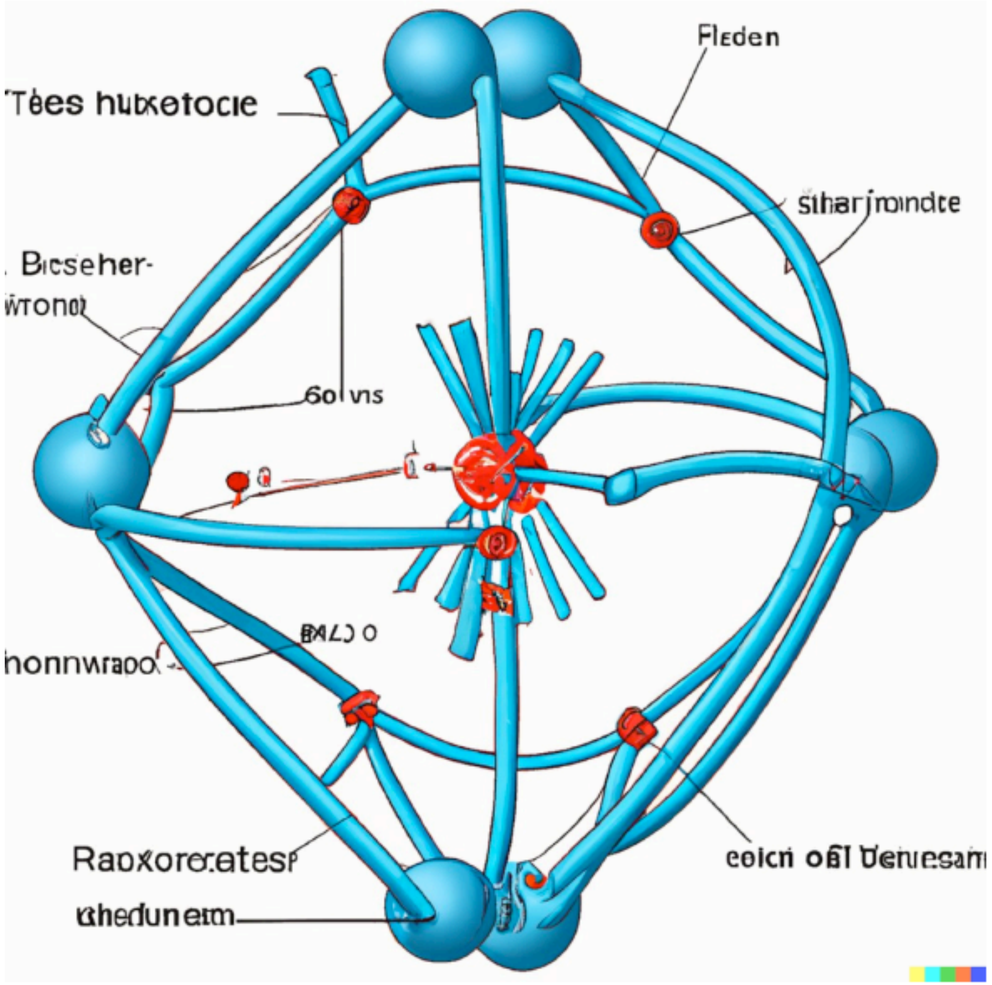
	PRE-2020	2020	2022	2023?	2025?	2030?
TEXT	Spam detection Translation Basic Q&A	Basic copy writing First drafts	Longer form Second drafts	Vertical fine tuning gets good (scientific papers, etc)	Final drafts better than the human average	Final drafts better than professional writers
CODE	1-line auto-complete	Multi-line generation	Longer form Better accuracy	More languages More verticals	Text to product (draft)	Text to product (final), better than full-time developers
IMAGES			Art Logos Photography	Mock-ups (product design, architecture, etc.)	Final drafts (product design, architecture, etc.)	Final drafts better than professional artists, designers, photographers)
VIDEO / 3D / GAMING			First attempts at 3D/video models	Basic / first draft videos and 3D files	Second drafts	AI Roblox Video games and movies are personalized dreams

Large model availability: First attempts Almost there Ready for prime time

https://huggingface.co/spaces/stabilityai/stable-diffusion

the inner structure of an electron

Generate image



https://future.com/how-to-build-gpt-3-for-science/

How to Build a GPT-3 for Science (scientific literature and data)

Josh Nicholson

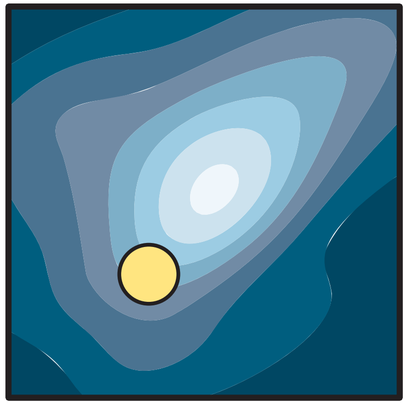
Posted August 18, 2022

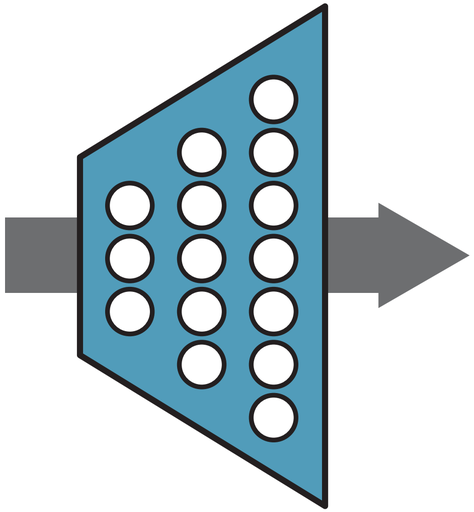
Some examples of complex potential prompts are:

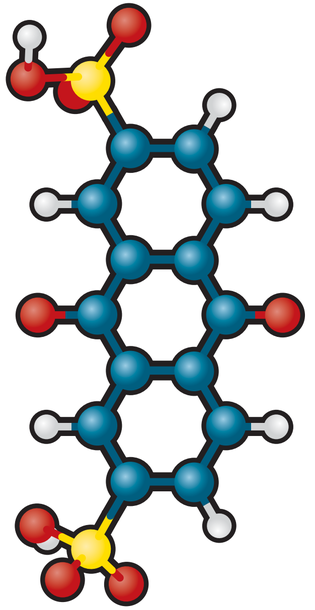
"Tell me why this hypothesis is wrong"
"Tell me why my treatment idea won't work"
"Generate a new treatment idea"
"What evidence is there to support social policy X?"
"Who has published the most reliable research in this field?"
"Write me a scientific paper based on my data"

Generative AI for matter engineering

latent space







chemical space

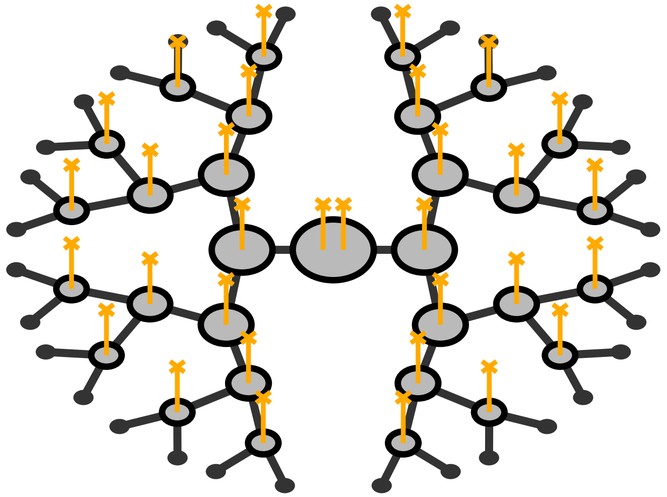
Generate

Inference

Review: “Inverse molecular design using machine learning”, Sanchez-Lengeling & Aspuru-Guzik, Science ’18

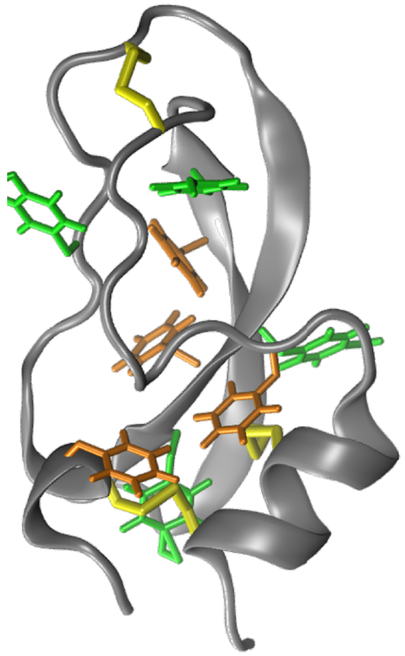
Generative AI for statistical physics

Renormalization group



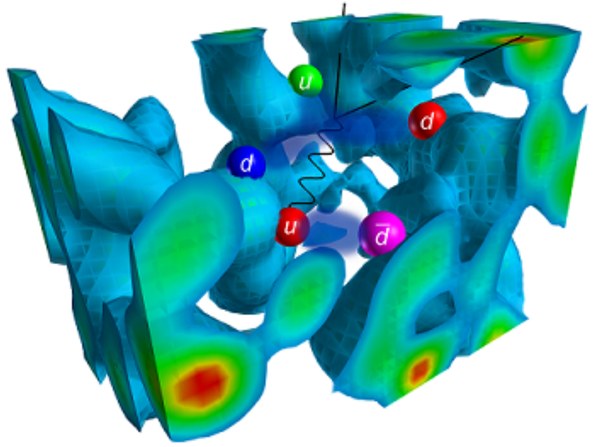
Li and LW, PRL ‘18
Li, Dong, Zhang, LW, PRX ‘20

Molecular simulation



Noe et al, Science ‘19
Wirnsberger et al, JCP ‘20

Lattice field theory



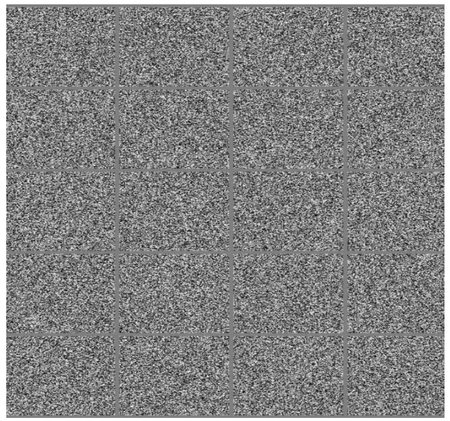
Albergo et al, PRD ‘19
Kanwar et al, PRL ‘20

These are principled computation: quantitatively accurate, interpretable, reliable, and generalizable even without data

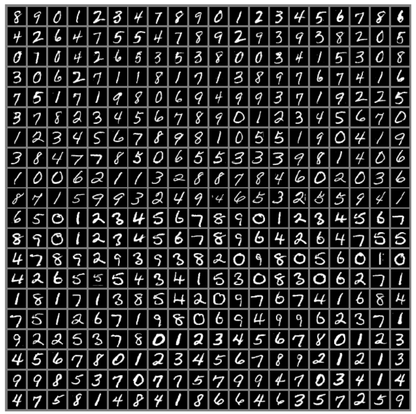
Probabilistic Generative Modeling

$$p(\boldsymbol{x})$$

How to express, learn, and sample from a high-dimensional probability distribution ?



“random” images

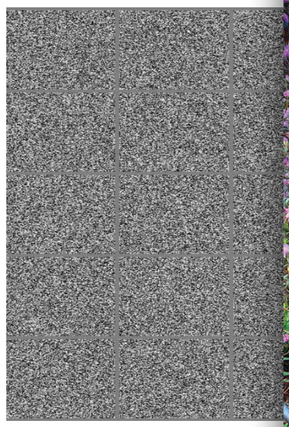


“natural” images

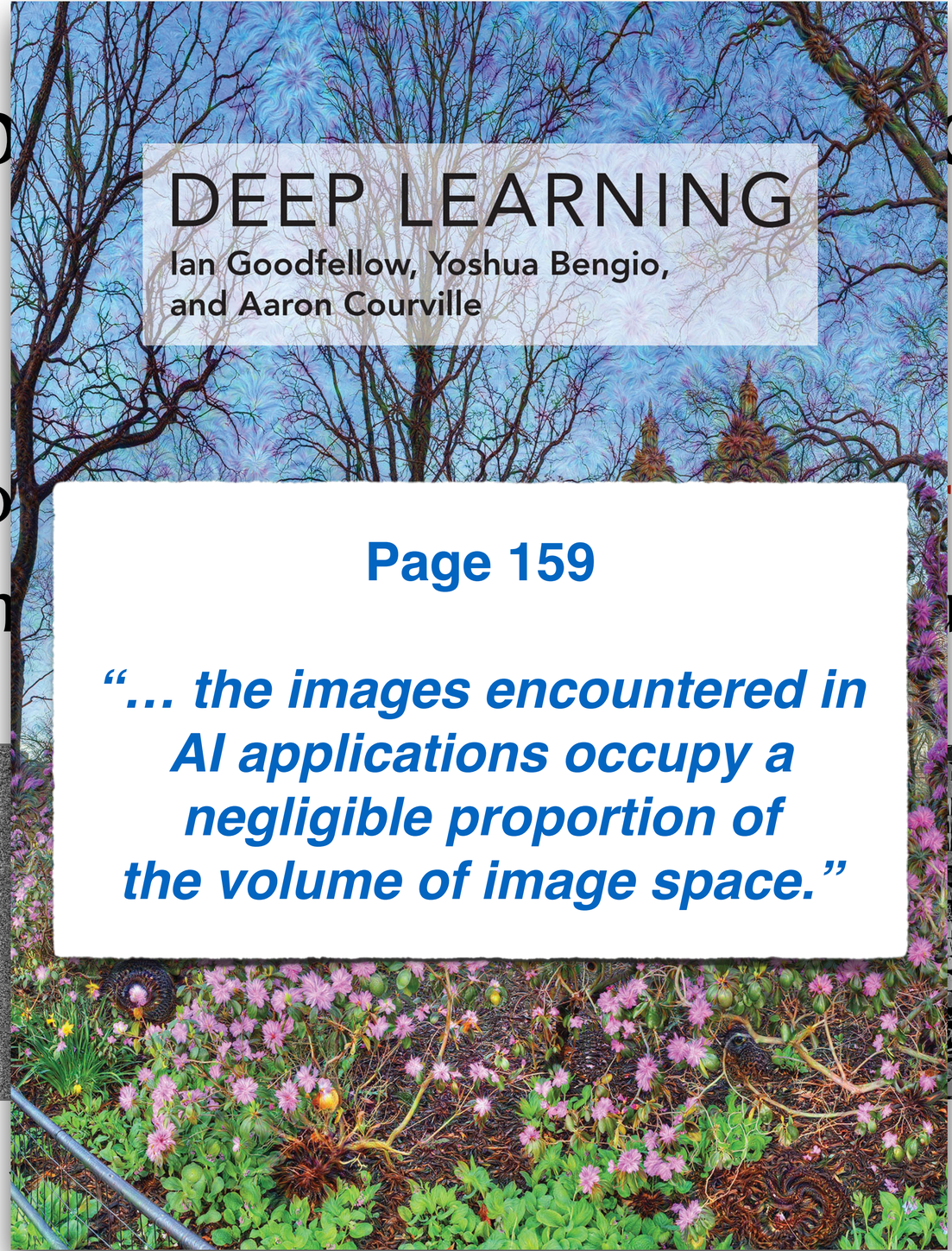
Probab deling

How to om a

high-din tion ?



“random”



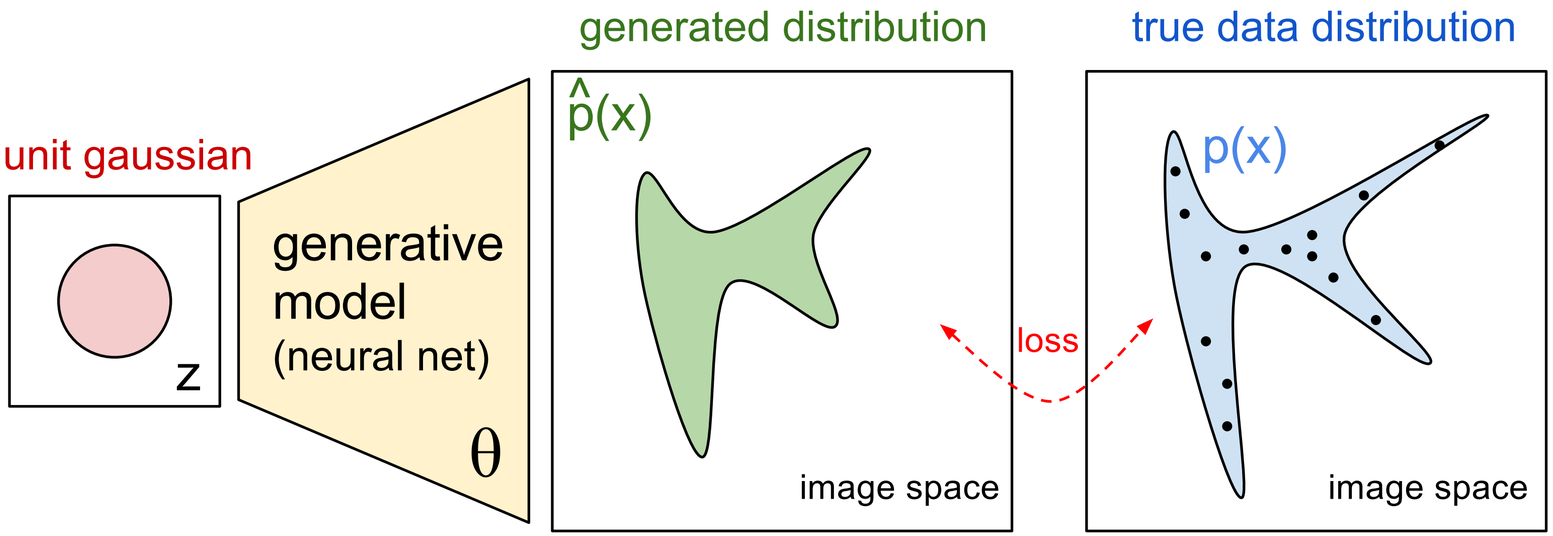
Page 159

“… the images encountered in AI applications occupy a negligible proportion of the volume of image space.”

Probabilistic Generative Modeling

$$p(\boldsymbol{x})$$

How to express, learn, and sample from a high-dimensional probability distribution ?



https://blog.openai.com/generative-models/

Generative models and their physics genes

Goodfellow,
NIPS tutorial, 1701.00160

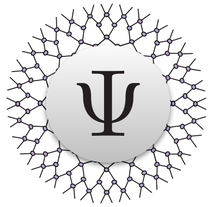
$p(\boldsymbol{x})$

Direct

GAN

Explicit density

Implicit density



Tensor Networks

Tractable density

-Fully visible belief nets

-NADE

-MADE

-PixelRNN

-Change of variables models (nonlinear ICA)

Approximate density

Variational

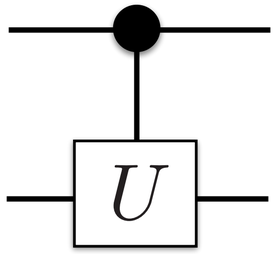
Variational autoencoder

Markov Chain

Boltzmann machine

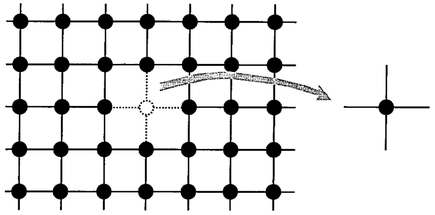
Markov Chain

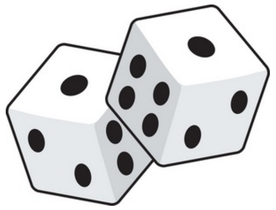
GSN

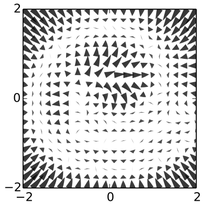


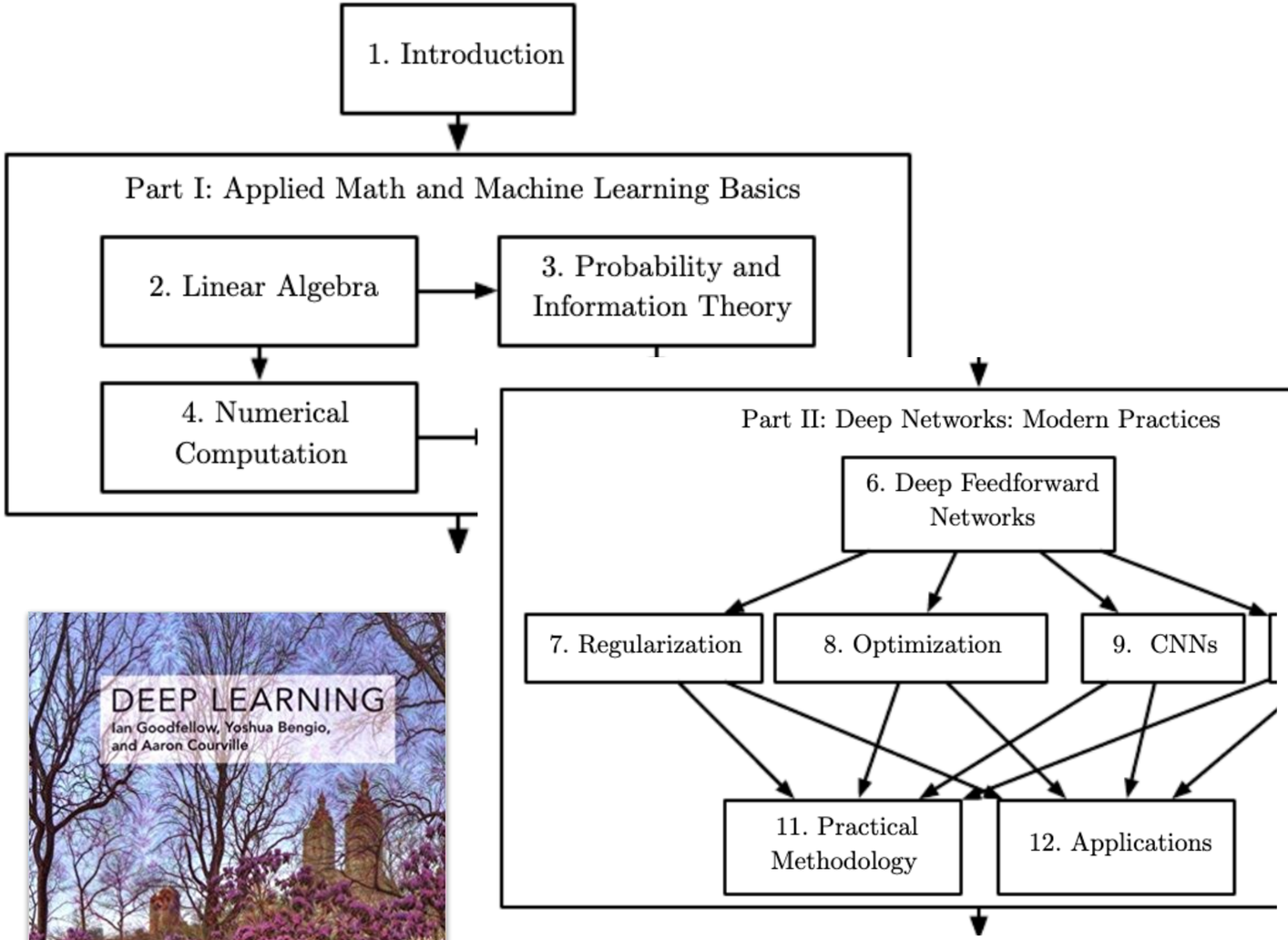
Quantum Circuits

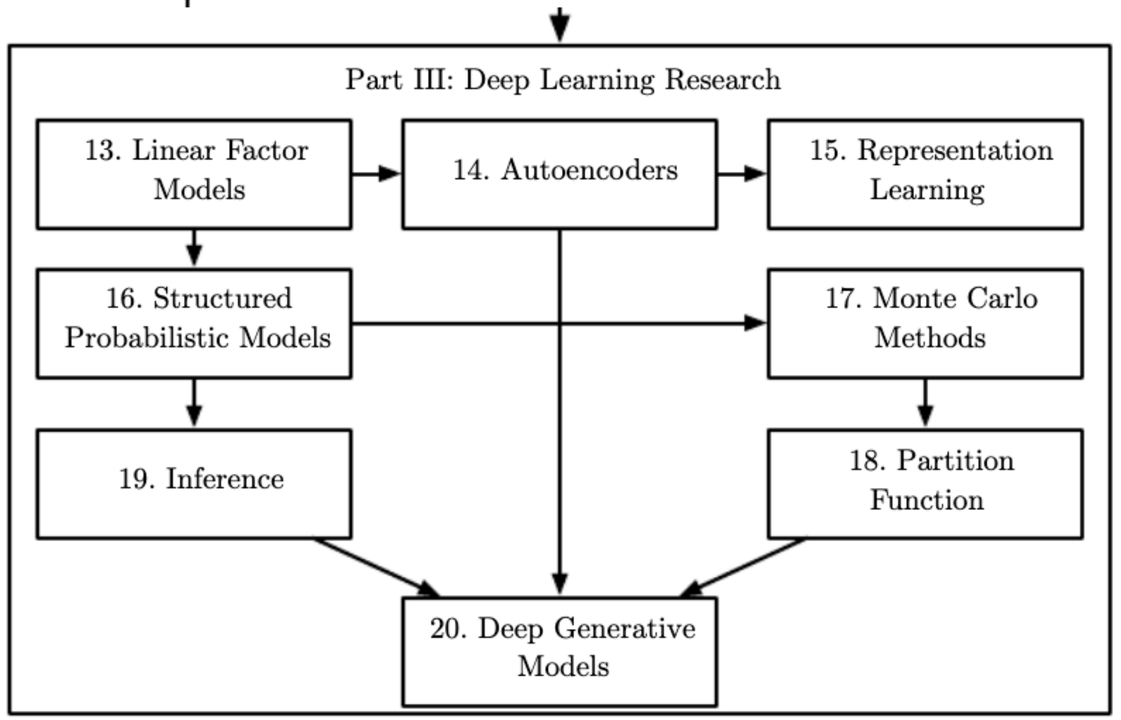
+Diffusion models



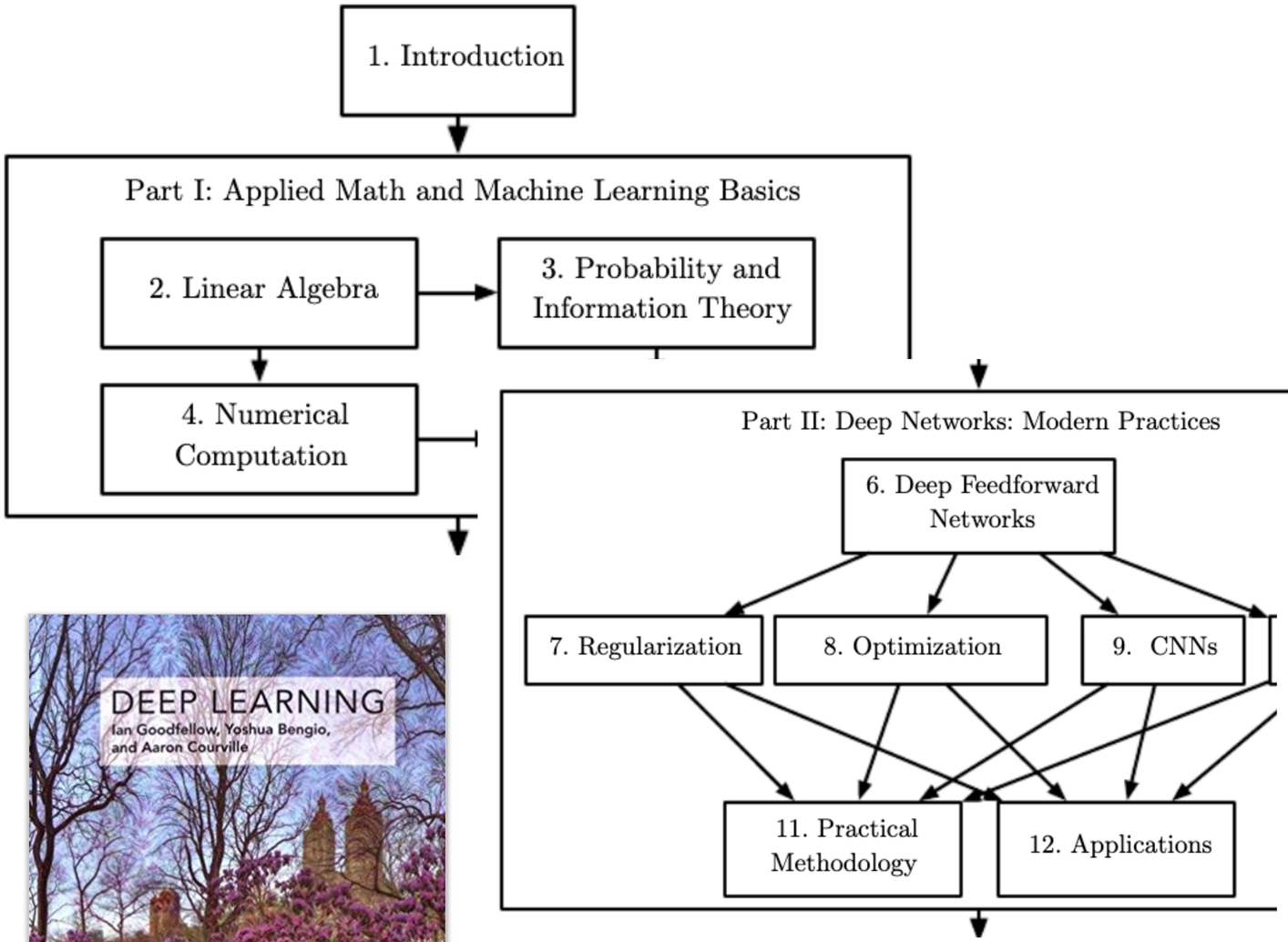


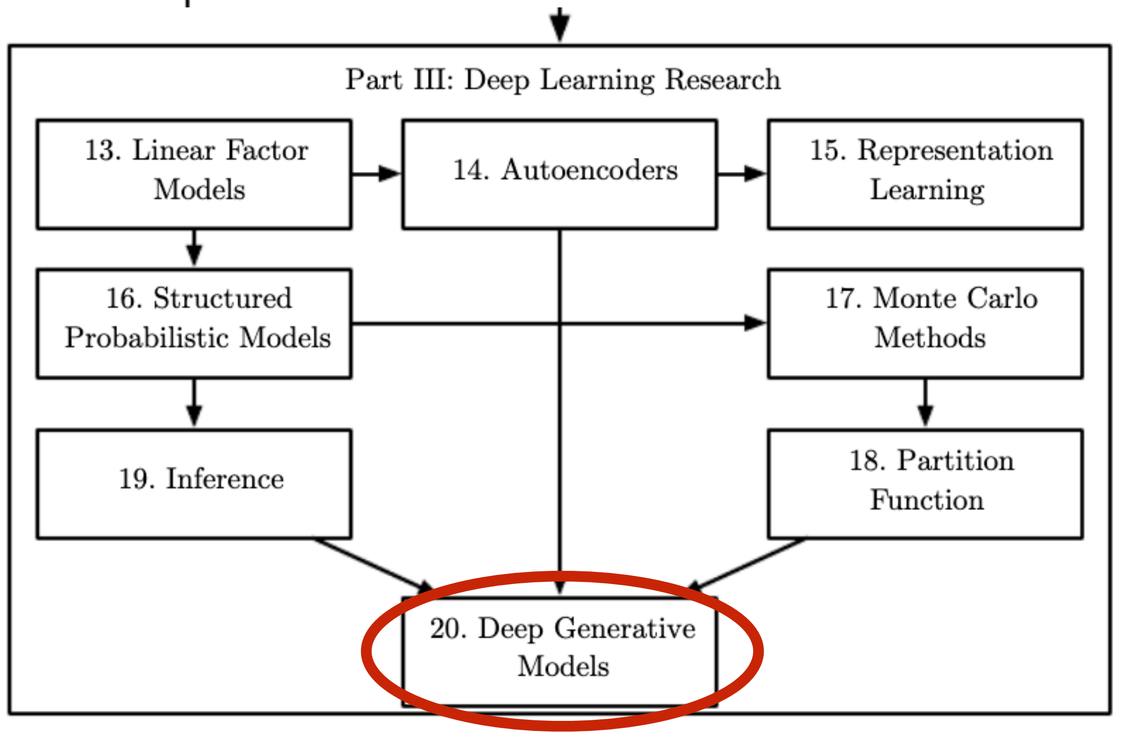






"Part III is the most important for a researcher—someone who wants to understand the breadth of perspectives that have been brought to the field of deep learning, and push the field forward towards true artificial intelligence."





"Part III is the most important for a researcher—someone who wants to understand the breadth of perspectives that have been brought to the field of deep learning, and push the field forward towards true artificial intelligence."

Lecture Note http://wangleiphy.github.io/lectures/PILtutorial.pdf

Generative Models for Physicists

Lei Wang*

Institute of Physics, Chinese Academy of Sciences
Beijing 100190, China

October 28, 2018

Abstract

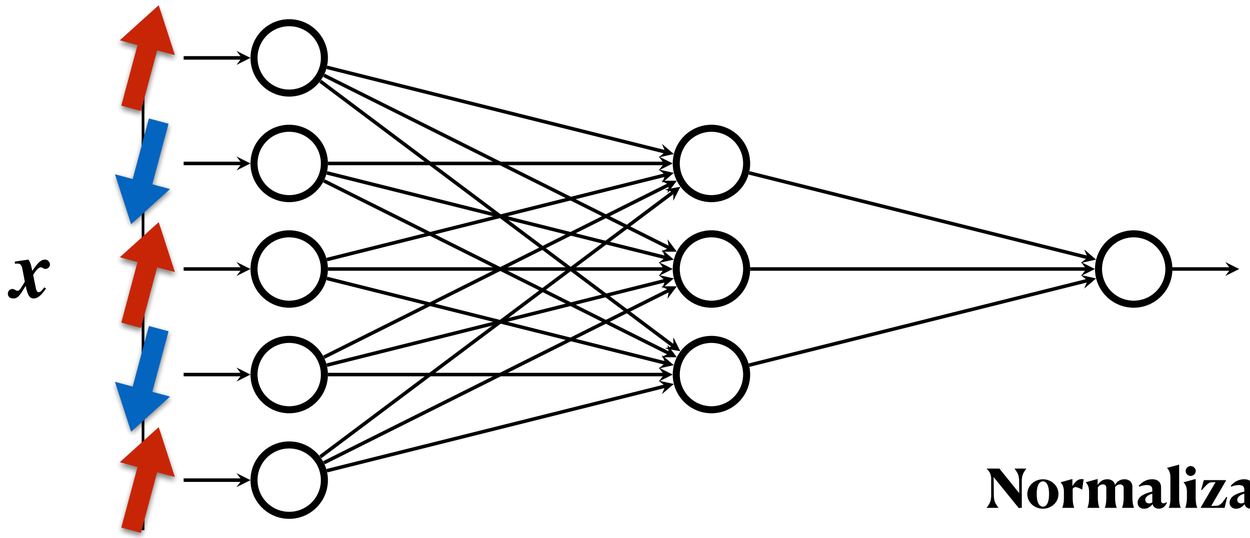
Generative models generate unseen samples according to a learned joint probability distribution in the high-dimensional space. They find wide applications in density estimation, variational inference, representation learning and more. Deep generative models and associated techniques (such as differentiable programing and representation learning) are cutting-edge technologies physicists can learn from deep learning.

This note introduces the concept and principles of generative modeling, together with applications of modern generative models (autoregressive models, normalizing flows, variational autoencoders etc) as well as the old ones (Boltzmann machines) to physics problems. As a bonus, this note puts some emphasize on physics-inspired generative models which take insights from statistical, quantum, and fluid mechanics.

The latest version of the note is at http://wangleiphy.github.io/. Please send comments, suggestions and corrections to the email address in below.

CONTENTS

So, what is the fuss ?



$$p(\boldsymbol{x}) \geq 0$$

Normalization ?

$$\int d\boldsymbol{x}\, p(\boldsymbol{x}) = 1$$

Sampling ?

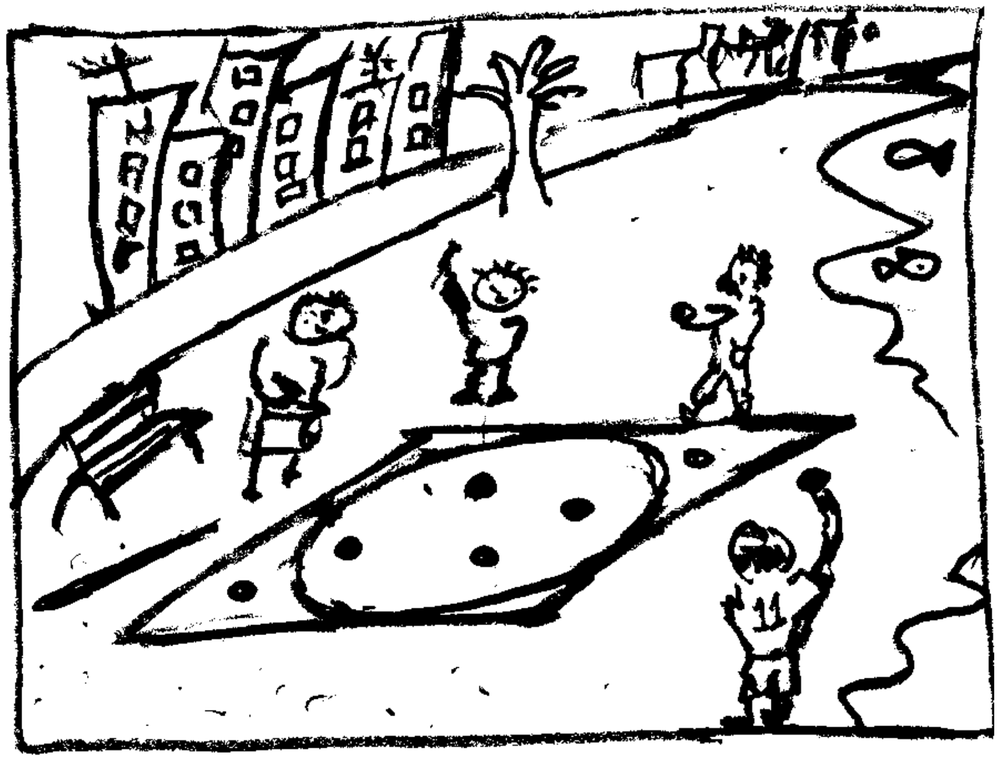
$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim p(\boldsymbol{x})}$$

Lecture 1: sampling distributions in vanilla MD

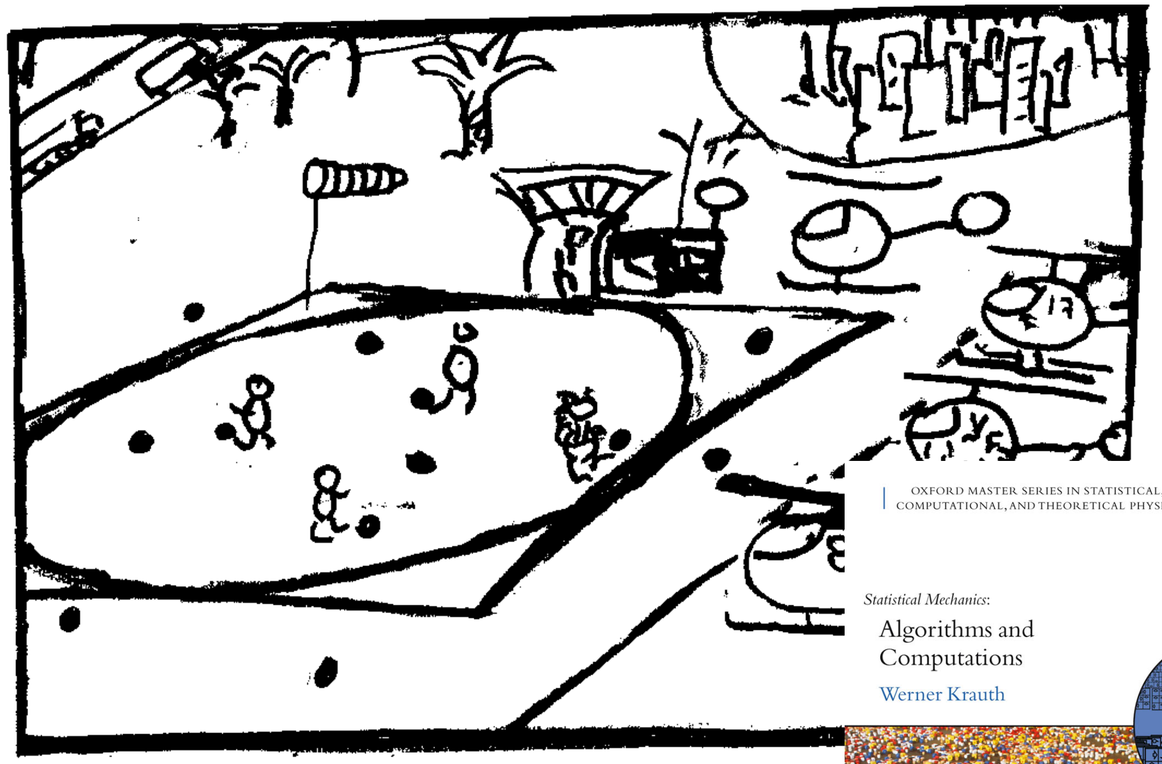
Lecture 2: free energy calculation (no rare event)

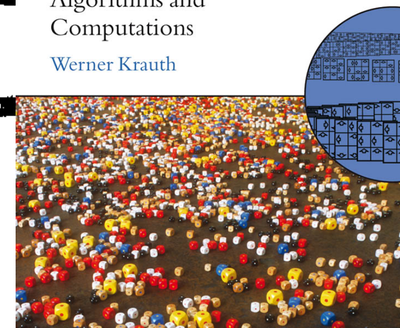
Perfect versus imperfect sampling

Children computing the number π on the Monte Carlo beach.



Adults computing the number π at the Monte Carlo heliport.





Generative modeling	Statistical physics
Negative log-likelihood	Energy function
Score function	Force
Latent variables	Collective variables/coarse graining/renormalization group
Partition function	Free energy calculation
Sample diversity	Enhanced sampling

Two sides of the same coin

Generative modeling

Known: samples

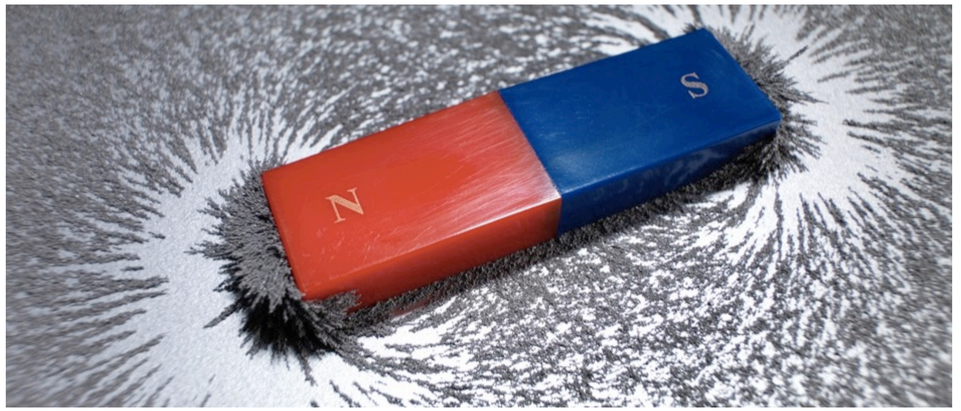
Unknown: generating distribution

Maximum likelihood estimation

"learn from data"

$$\mathscr{L} = -\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right]$$

Statistical physics



Known: energy function

Unknown: partition function, samples

Variational free energy

"learn from Hamiltonian"

$$F = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[k_B T \ln p(\boldsymbol{x}) + H(\boldsymbol{x})\right]$$

The variational free energy principle

$$F[p] = \int dx\, p(x) \left[H(x) + k_B T \ln p(x) \right] \geq F$$

$F[p]$: variational density

$H(x)$: energy

$k_B T \ln p(x)$: entropy 😱

Difficulties in Applying the Variational Principle to Quantum Field Theories[1]

Richard P. Feynman

[1] transcript of his talk in 1987

Generative models!

Deep variational free energy approach

Use deep generative models as the variational density

$$F[p] = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \Big[H(\boldsymbol{x}) + k_B T \ln p(\boldsymbol{x}) \Big]$$

energy ↓ entropy ↓ 😄

Li and LW, PRL '18
Wu, LW, Zhang, PRL '19

with normalizing flow & autoregressive models

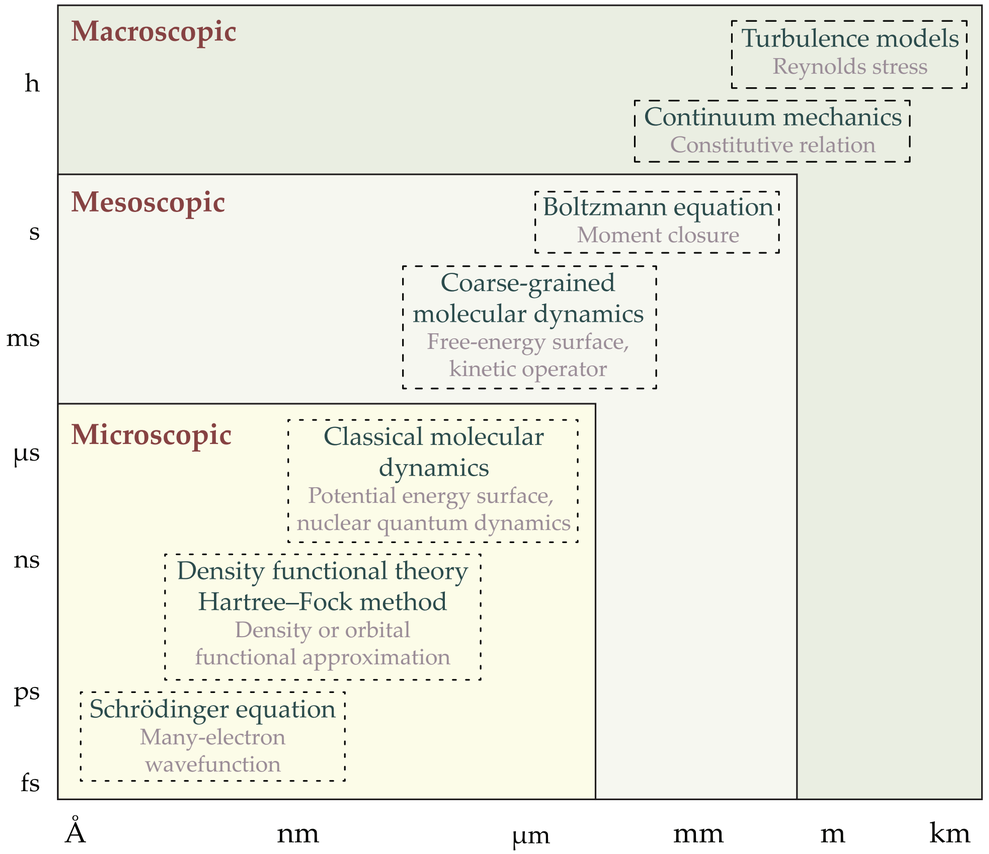
✓ Tractable entropy

✓ Direct sampling

✓ Turning a sampling problem to an optimization problem better leverages the deep learning engine:

Deep variational free-energy in the context

E, Han,Zhang, Physics Today 2020

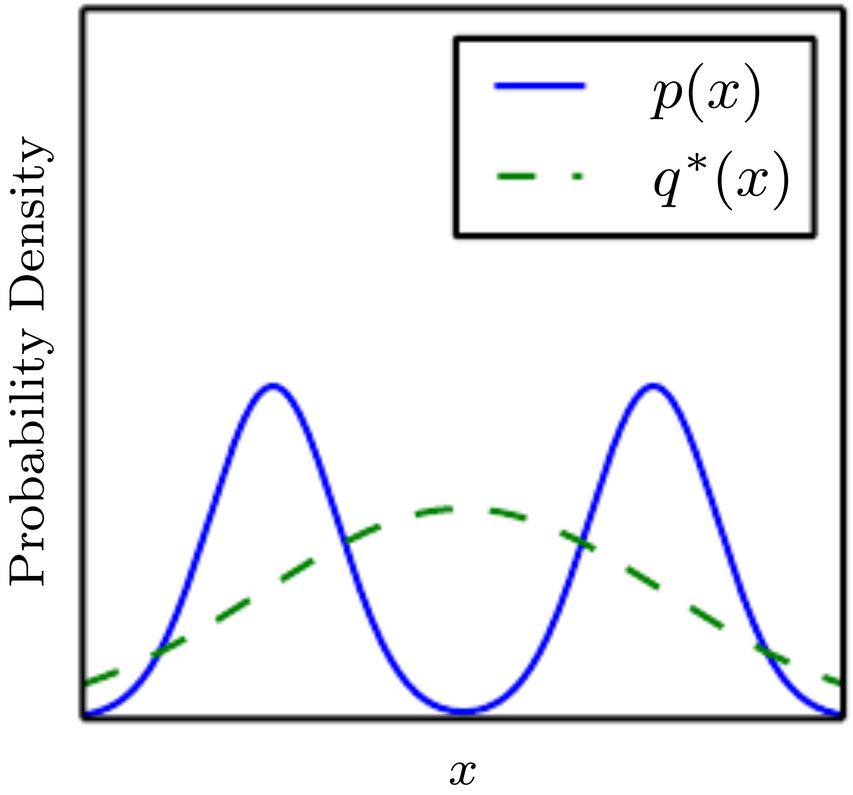


Objective	Model	Data	Task
MD potential energy surface	3N-dim function	DFT energy/ force	Generalization
DFT xc energy functional	3-dim functional	QMC/ CCSD/...	
Variational free-energy	3N-dim functional	No	Optimization

Forward KL or Reverse KL ?

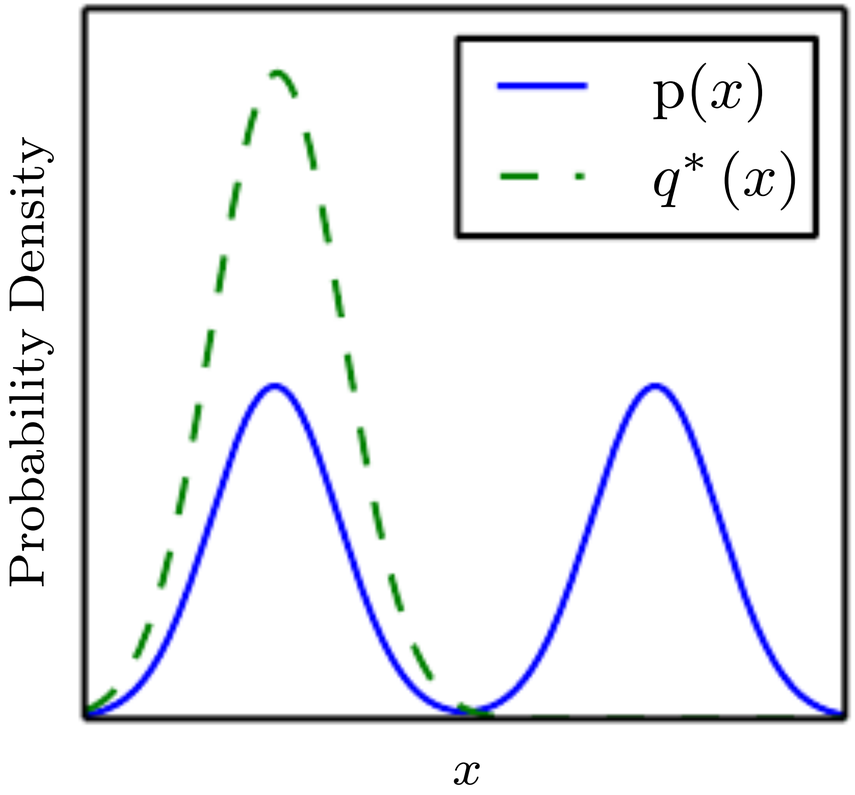
Maximum likelihood estimation

$$q^* = \text{argmin}_q D_{\text{KL}}(p\|q)$$



Variational free energy

$$q^* = \text{argmin}_q D_{\text{KL}}(q\|p)$$



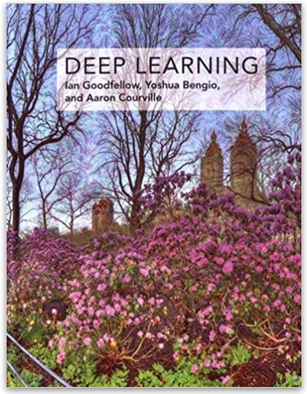


Fig. 3.6, Goodfellow, Bengio, Courville, http://www.deeplearningbook.org/

α-divergence

Minka, Microsoft Research Technical Report 2005

$$D_\alpha(p \,||\, q) = \frac{\int_x \alpha p(x) + (1-\alpha)q(x) - p(x)^\alpha q(x)^{1-\alpha} dx}{\alpha(1-\alpha)}$$

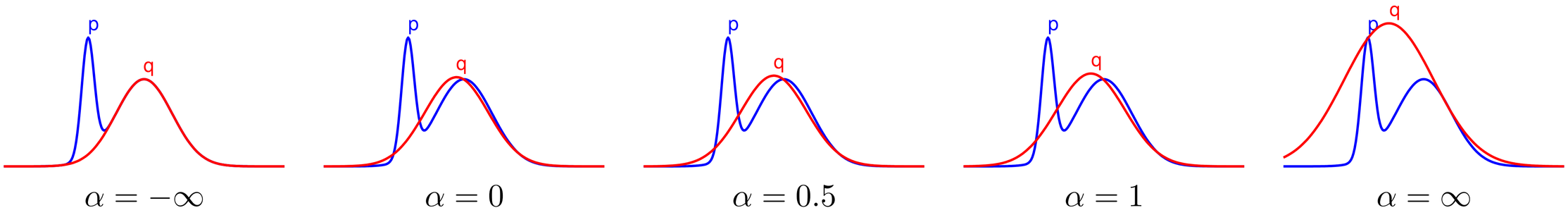
$$D_{-1}(p \,||\, q) = \frac{1}{2}\int_x \frac{(q(x)-p(x))^2}{p(x)} dx$$

$$\lim_{\alpha \to 0} D_\alpha(p \,||\, q) = \mathrm{KL}(q \,||\, p)$$

$$D_{\frac{1}{2}}(p \,||\, q) = 2\int_x \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 dx$$

$$\lim_{\alpha \to 1} D_\alpha(p \,||\, q) = \mathrm{KL}(p \,||\, q)$$

$$D_2(p \,||\, q) = \frac{1}{2}\int_x \frac{(p(x)-q(x))^2}{q(x)} dx$$



$\alpha = -\infty$ $\alpha = 0$ $\alpha = 0.5$ $\alpha = 1$ $\alpha = \infty$

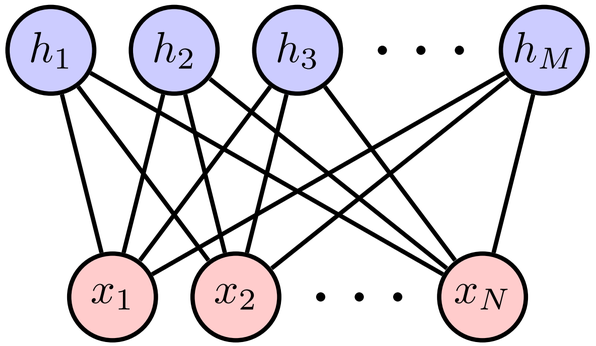
Fisher divergence, defined as

$$F(q,p) = \int_{\mathbb{R}^d} \|\nabla \log q(\theta) - \nabla \log p(\theta)\|^2 \, q(\theta)\, d\theta,$$

Boltzmann machines

$$\mathscr{L} = -\,\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right] \qquad p(\boldsymbol{x}) = e^{-E(\boldsymbol{x})}/Z$$

Learn



$$\nabla\mathscr{L} = \langle \nabla E\rangle_{\text{data}} - \langle \nabla E\rangle_{\text{model}}$$

2210.10318

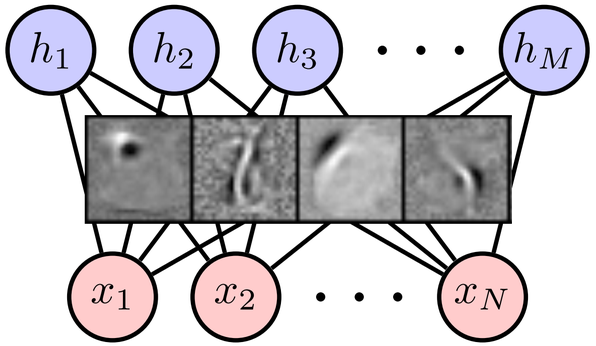
GAUSSIAN-BERNOULLI RBMS WITHOUT TEARS

Renjie Liao[*1], **Simon Kornblith**[2], **Mengye Ren**[3], **David J. Fleet**[2,4,5], **Geoffrey Hinton**[2,4,5]

Boltzmann machines

$$\mathscr{L} = -\,\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right] \qquad p(\boldsymbol{x}) = e^{-E(\boldsymbol{x})}/Z$$

Learn



$$\nabla\mathscr{L} = \langle \nabla E\rangle_{\text{data}} - \langle \nabla E\rangle_{\text{model}}$$

2210.10318

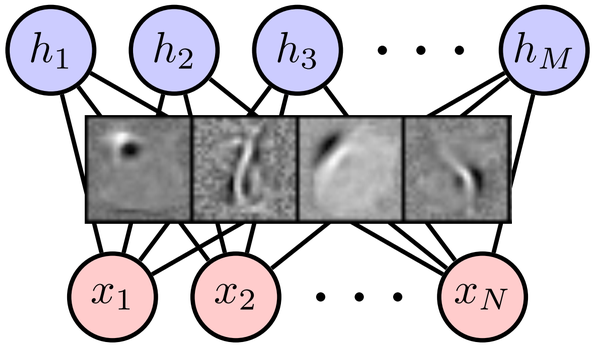
GAUSSIAN-BERNOULLI RBMS WITHOUT TEARS

Renjie Liao[*1], **Simon Kornblith**[2], **Mengye Ren**[3], **David J. Fleet**[2,4,5], **Geoffrey Hinton**[2,4,5]

Boltzmann machines

$$\mathscr{L} = -\,\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right] \qquad p(\boldsymbol{x}) = e^{-E(\boldsymbol{x})}/Z$$

Learn



Generate

$$\nabla\mathscr{L} = \langle \nabla E\rangle_{\text{data}} - \langle \nabla E\rangle_{\text{model}}$$

2210.10318

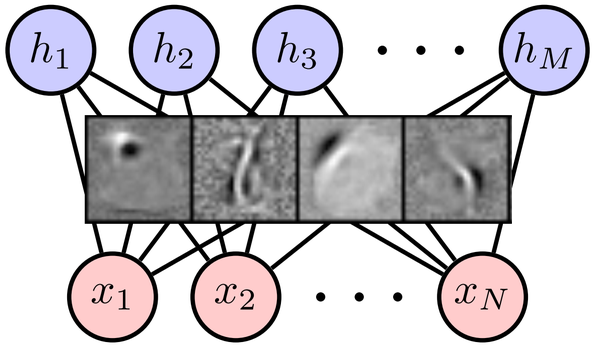
Gaussian-Bernoulli RBMs Without Tears

Renjie Liao[*1], **Simon Kornblith**[2], **Mengye Ren**[3], **David J. Fleet**[2,4,5], **Geoffrey Hinton**[2,4,5]

Boltzmann machines

$$\mathscr{L} = -\,\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right] \qquad p(\boldsymbol{x}) = e^{-E(\boldsymbol{x})}/Z$$

Learn



Generate

$$\nabla\mathscr{L} = \langle \nabla E\rangle_{\text{data}} - \langle \nabla E\rangle_{\text{model}}$$

2210.10318

Gaussian-Bernoulli RBMs Without Tears

Renjie Liao[*1], **Simon Kornblith**[2], **Mengye Ren**[3], **David J. Fleet**[2,4,5], **Geoffrey Hinton**[2,4,5]

Autoregressive models

$$p(\boldsymbol{x}) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)\cdots$$

Language: GPT 2005.14165

"... quick brown fox jumps ..."

$p(\textit{jumps} \mid \ldots)$

Speech: WaveNet 1609.03499

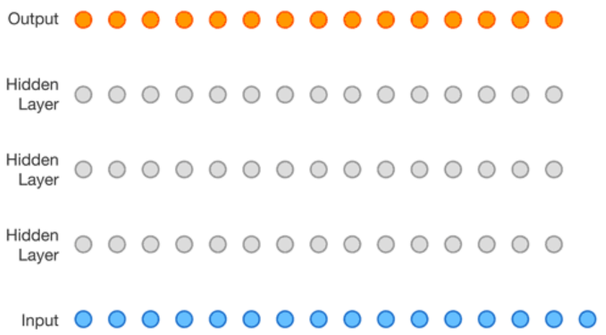
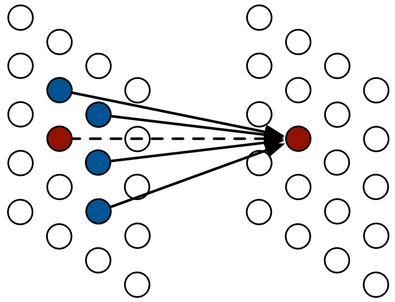
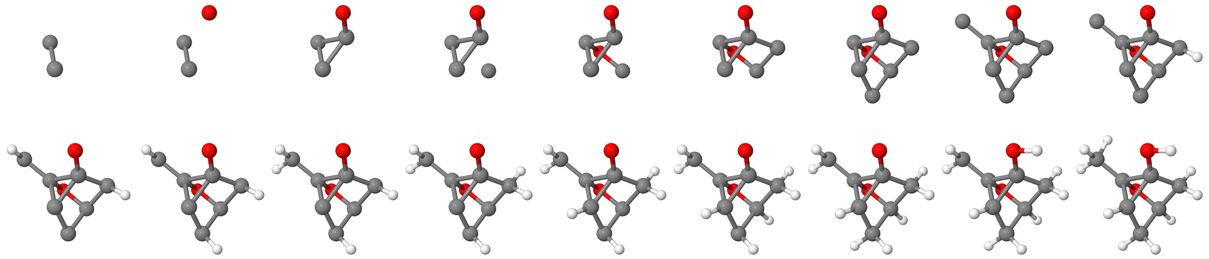


Image: PixelCNN 1601.06759



Molecular graph: 1810.11347



Autoregressive models

$$p(\boldsymbol{x}) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)\cdots$$

Language: GPT 2005.14165

"... quick brown fox jumps ..."

$p(\textit{jumps} \mid \ldots)$

Speech: WaveNet 1609.03499

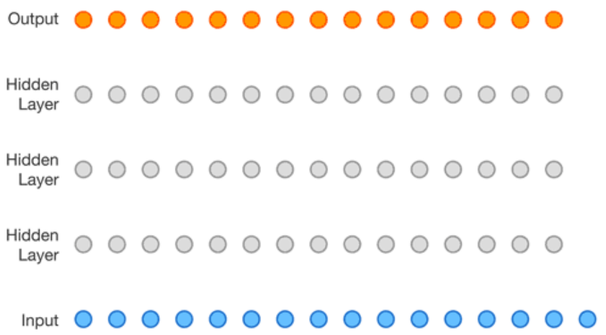
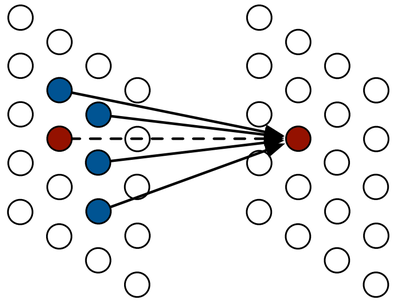
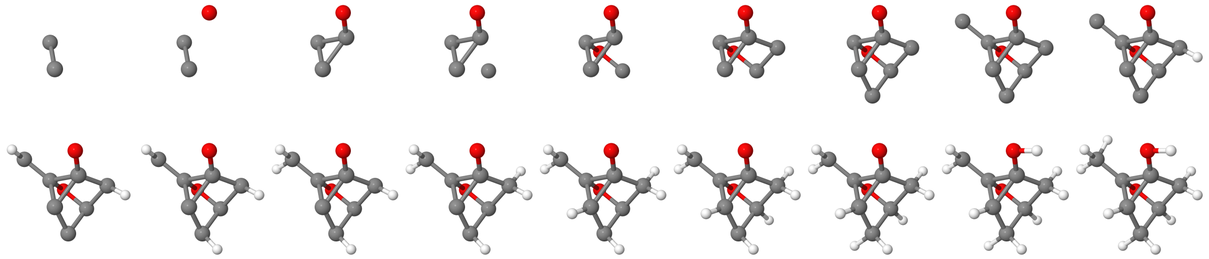


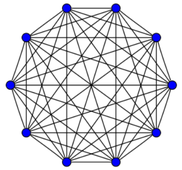
Image: PixelCNN 1601.06759



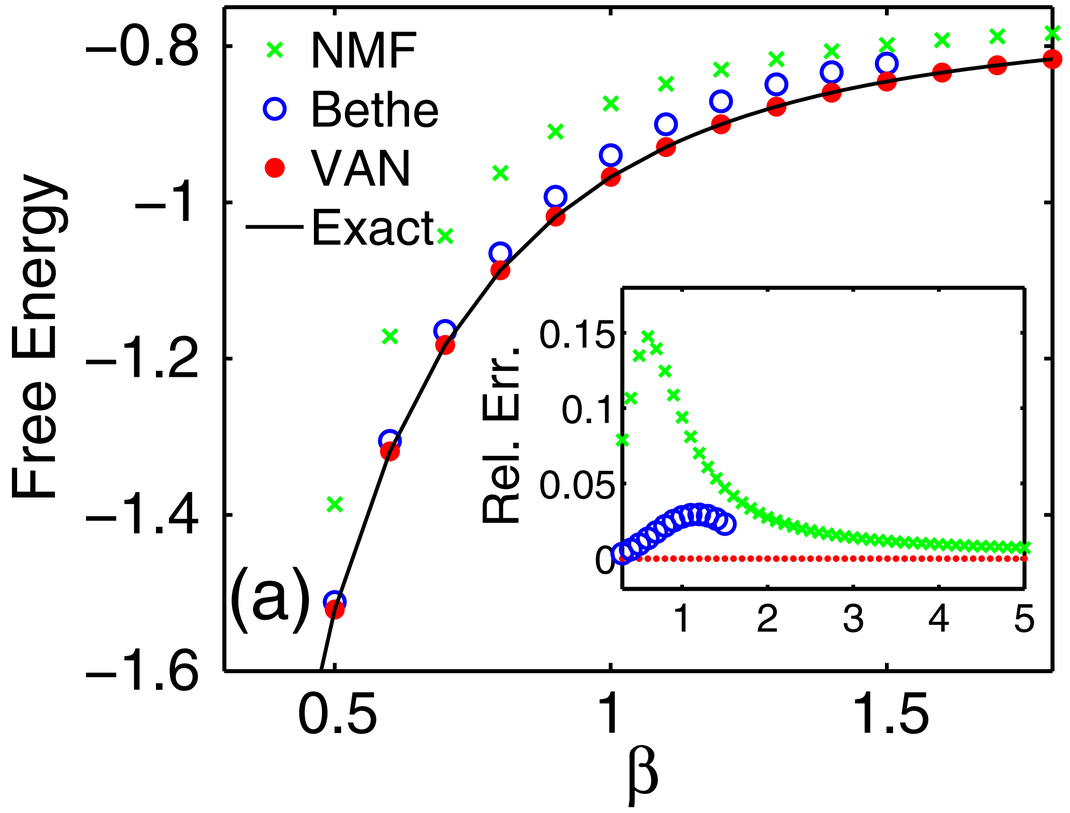
Molecular graph: 1810.11347



Variational autoregressive networks



Sherrington-Kirkpatrick spin glass



Variational autoregressive network

$$p(\boldsymbol{x}) = \prod_i p(x_i \,|\, \boldsymbol{x}_{<i})$$

github.com/wdphy16/stat-mech-van

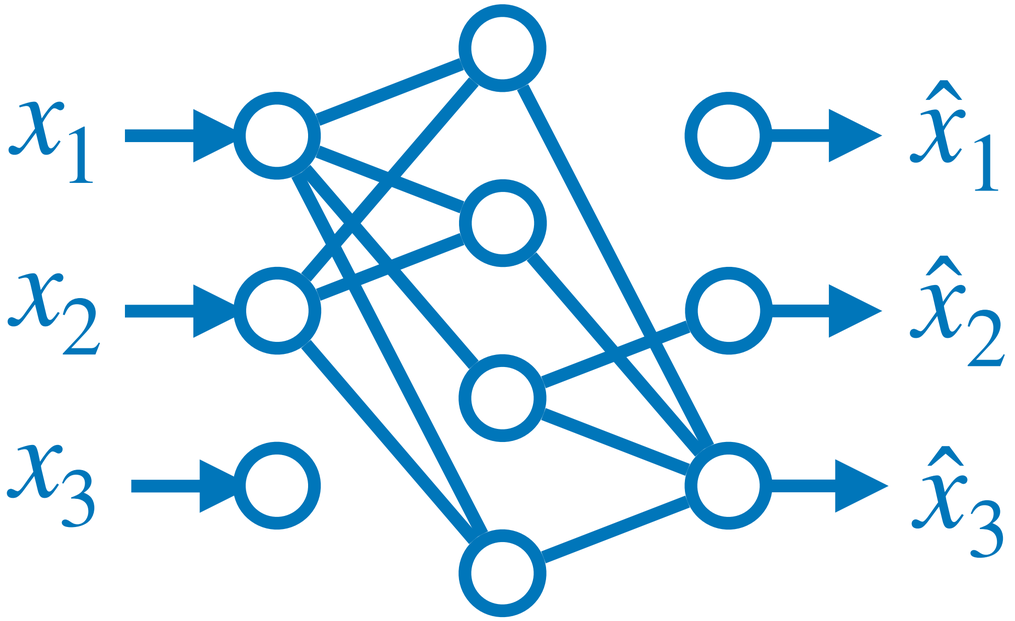
Wu, LW, Zhang, PRL '19

Conventional approaches

Naive mean-field factorized probability $p(\boldsymbol{x}) = \prod_i p(x_i)$

Bethe approximation pairwise interaction $p(\boldsymbol{x}) = \prod_i p(x_i) \prod_{(i,j)\in E} \frac{p(x_i, x_j)}{p(x_i)p(x_j)}$

Implementation: autoregressive masks



Masked Autoencoder
Germain et al, 1502.03509

$$p(x_1) = \text{Bern}(\hat{x}_1) \qquad p(x_2 \,|\, x_1) = \text{Bern}(\hat{x}_2) \qquad p(x_3 \,|\, x_1, x_2) = \text{Bern}(\hat{x}_3)$$

Other examples: PixelCNN, van den Oord et al, 1601.06759 Casual transformer, 1706.03762

Other ways to implement autoregressive models: recurrent networks

Normalizing flows

Parallel WaveNet 1711.10433

https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

Glow 1807.03039

https://blog.openai.com/glow/

Normalizing flows

Parallel WaveNet 1711.10433

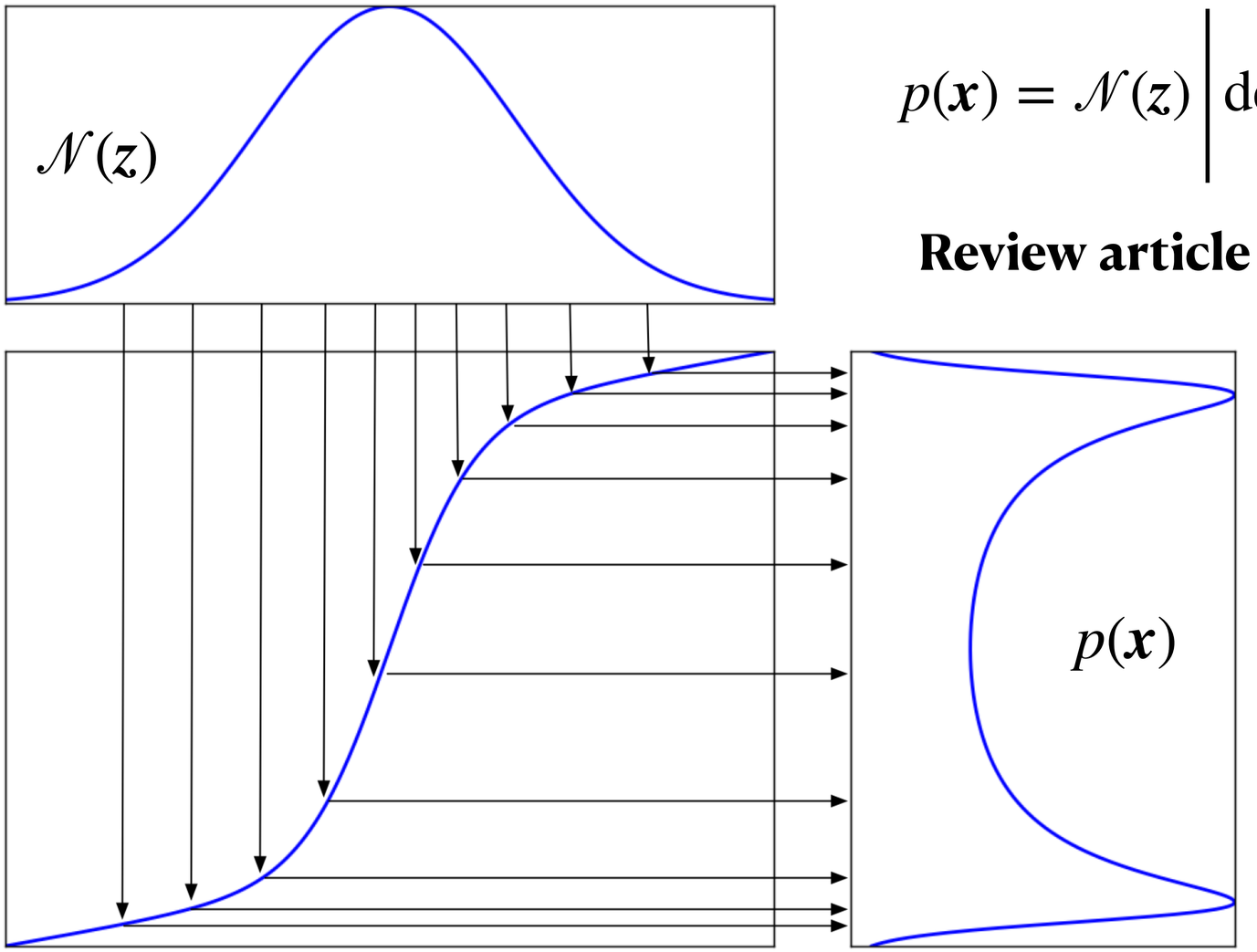
https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

Glow 1807.03039

https://blog.openai.com/glow/

Normalizing flow in a nutshell

Base density



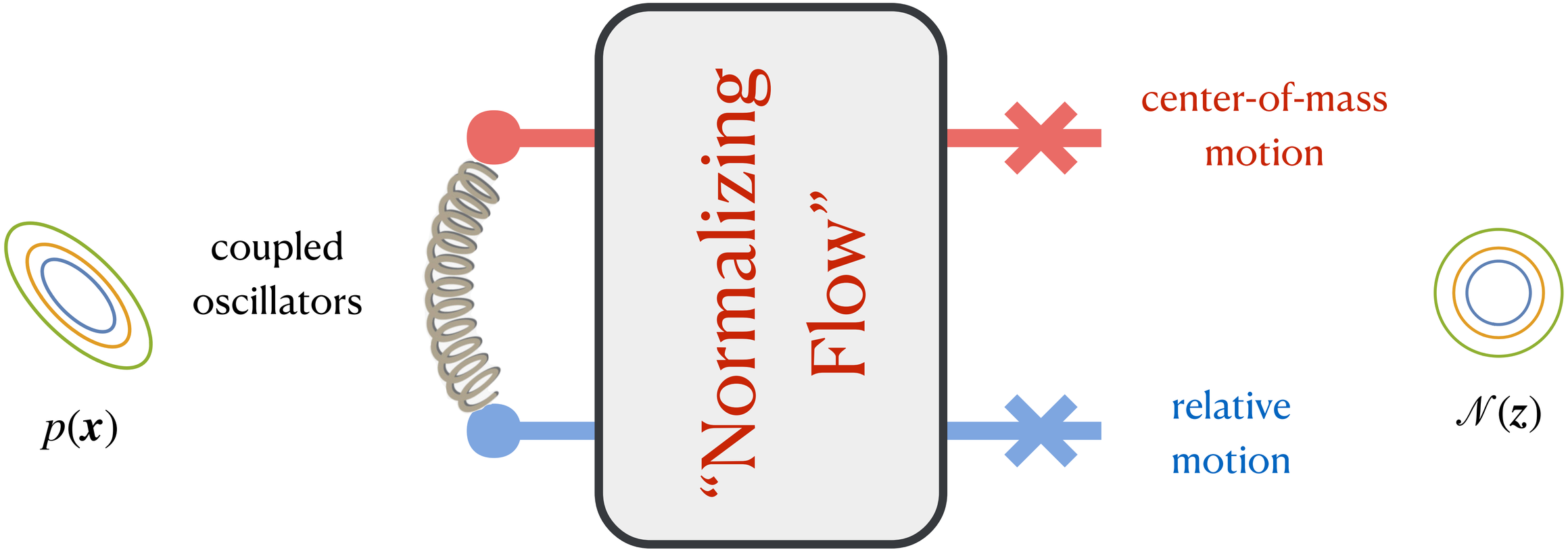
$$p(\boldsymbol{x}) = \mathcal{N}(z)\left|\det\left(\frac{\partial z}{\partial \boldsymbol{x}}\right)\right|$$

Review article 1912.02762

"neural net" with 1 neuron

Target density

Normalizing flow for physics: an intuition

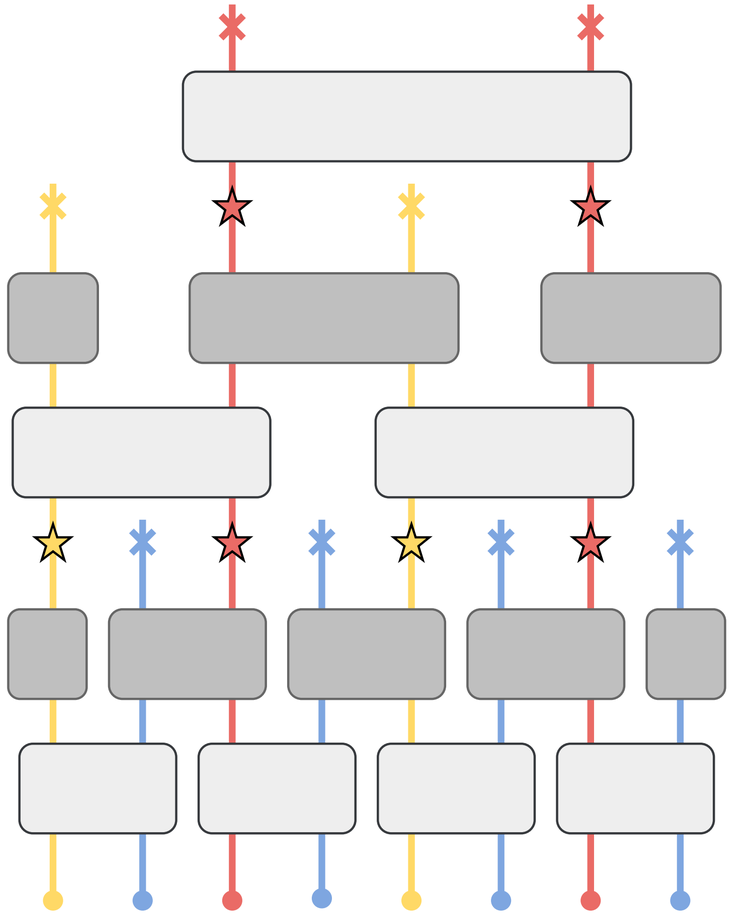


High-dimensional, composable, learnable, nonlinear transformations

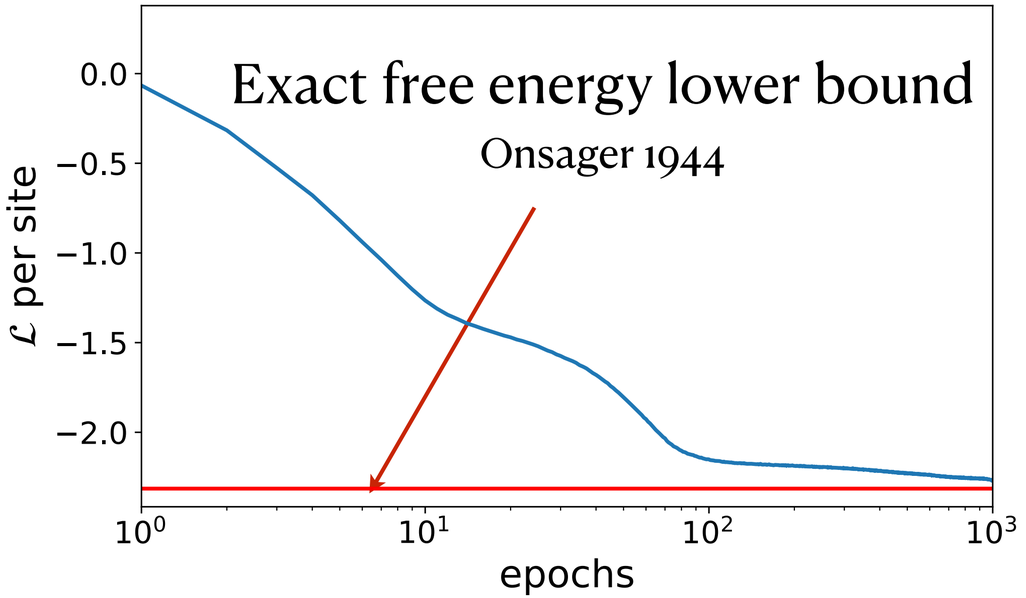
Neural network renormalization group

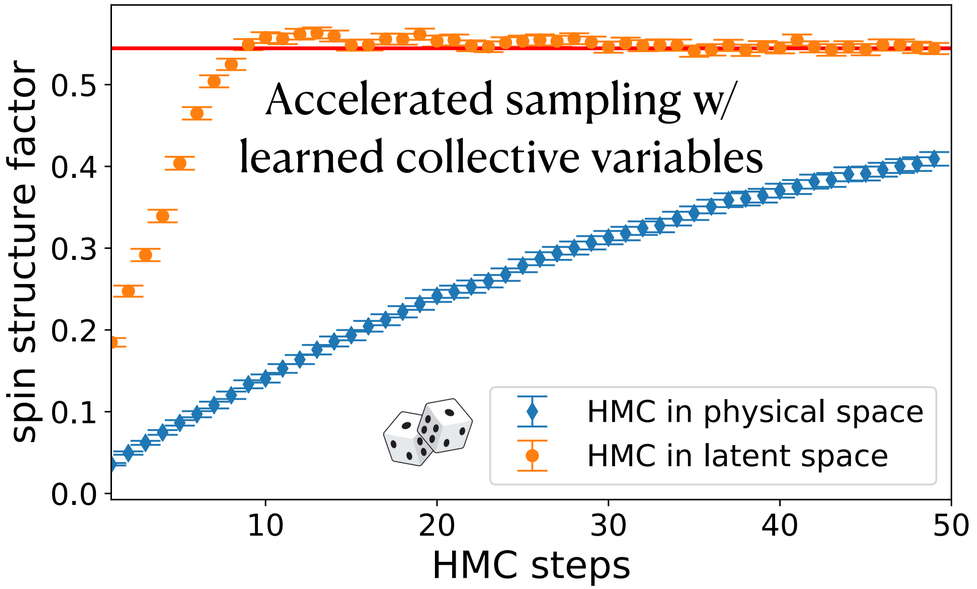
Li, LW, PRL '18 [lio12589/NeuralRG]

Collective variables



Physical variables

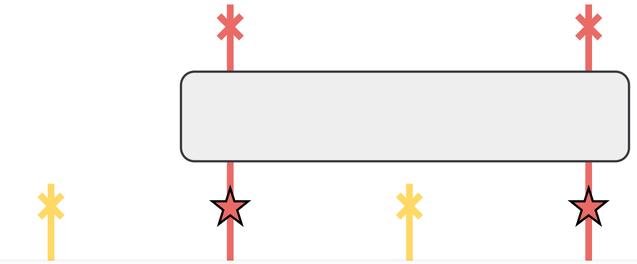




Neural network renormalization group

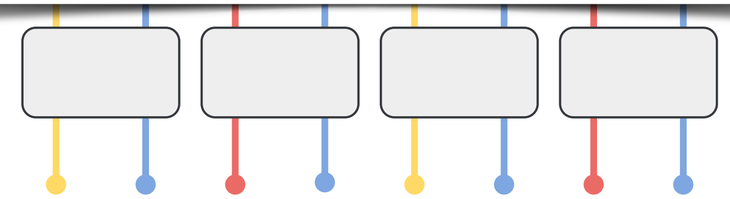
Li, LW, PRL '18 [li012589/NeuralRG]

Collective variables

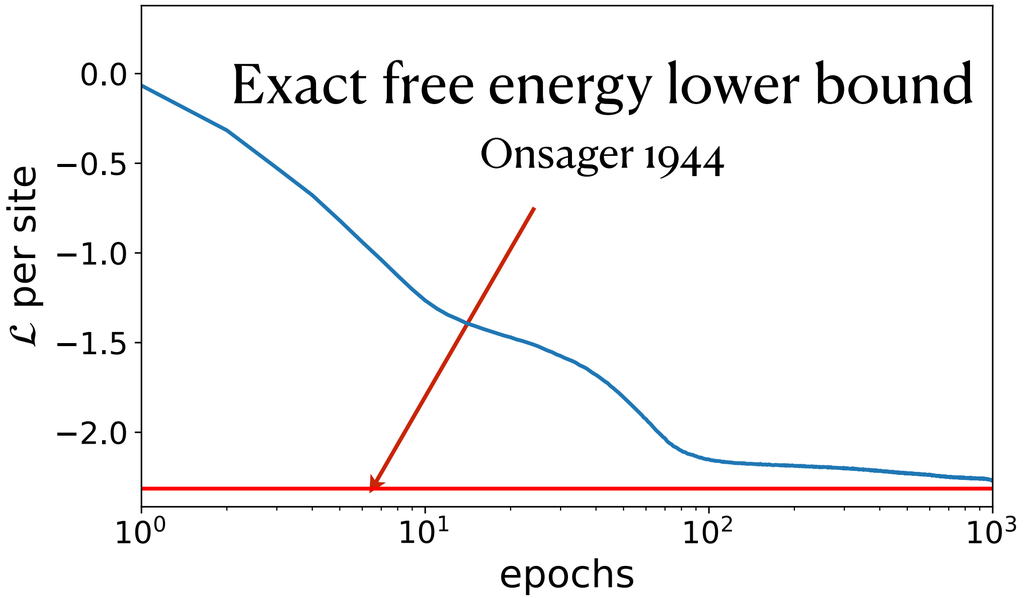


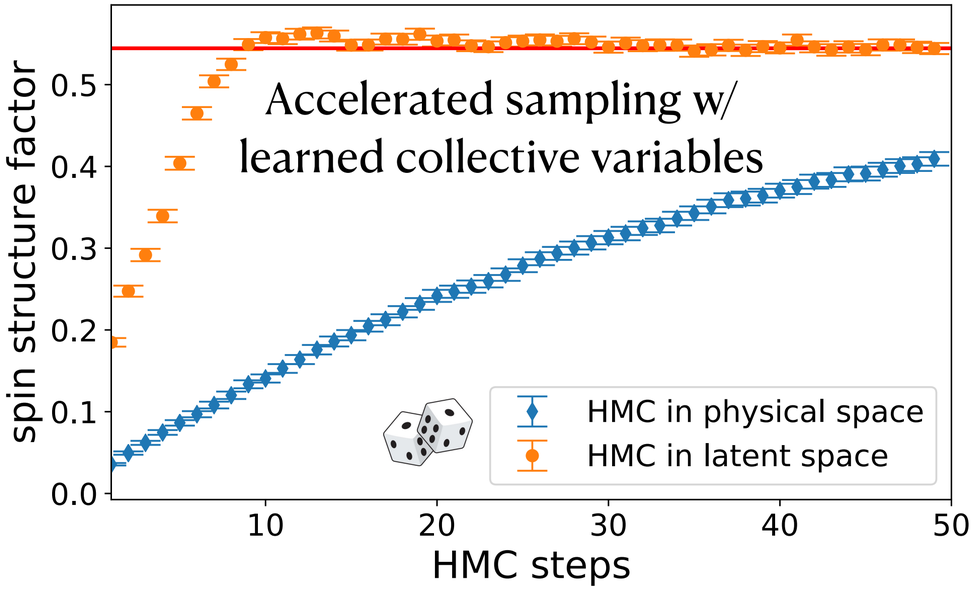
Probability Transformation

$$\ln p(\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{z}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|$$



Physical variables





Continuous normalizing flows

$$\ln p(\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{z}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|$$

Consider infinitesimal change-of-variables Chen et al 1806.07366

$$\boldsymbol{x} = \boldsymbol{z} + \varepsilon \boldsymbol{v} \qquad \ln p(\boldsymbol{x}) - \ln \mathcal{N}(\boldsymbol{z}) = -\ln \left| \det \left(1 + \varepsilon \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} \right) \right|$$

$\varepsilon \to 0$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v} \qquad \frac{d \ln p(\boldsymbol{x}, t)}{dt} = -\nabla \cdot \boldsymbol{v}$$

Continuous normalizing flows

$$\ln p(\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{z}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|$$

Consider infinitesimal change-of-variables Chen et al 1806.07366

$$\boldsymbol{x} = \boldsymbol{z} + \varepsilon \boldsymbol{v}$$

$$\ln p(\boldsymbol{x}) - \ln \mathcal{N}(\boldsymbol{z}) = - \ln \left| \det \left(1 + \varepsilon \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} \right) \right|$$

$\varepsilon \to 0$

$t = 1$ $t = 0$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

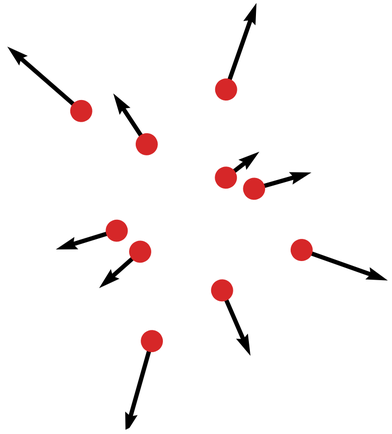
$$\frac{d \ln p(\boldsymbol{x}, t)}{dt} = - \nabla \cdot \boldsymbol{v}$$

Fluid physics behind flows

Zhang, E, LW 1809.10188
wangleiphy/MongeAmpereFlow

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

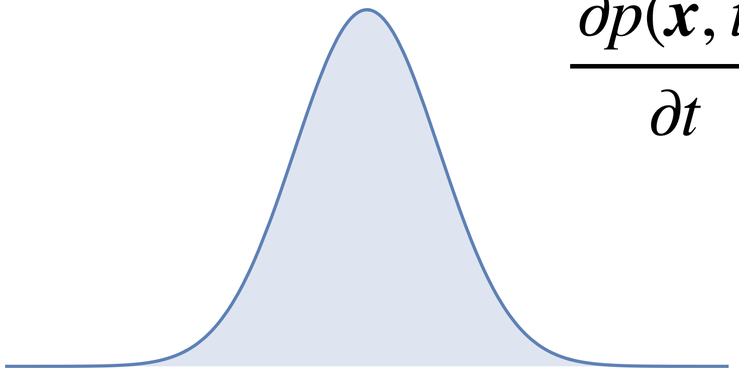
$$\frac{d \ln p(\boldsymbol{x}, t)}{dt} = -\nabla \cdot \boldsymbol{v}$$



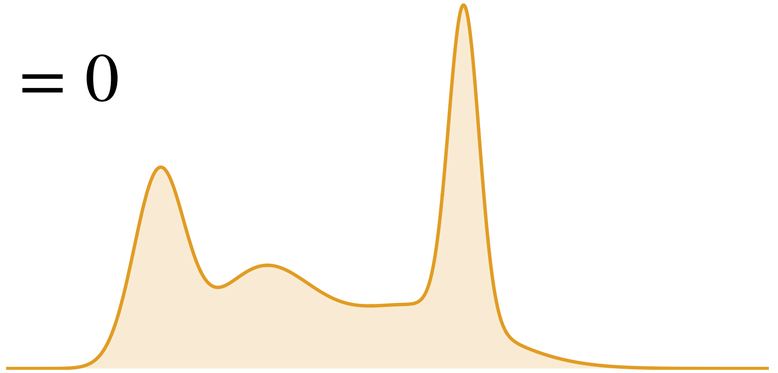
$$\frac{d}{dt} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla$$

"material derivative"

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot \left[p(\boldsymbol{x}, t) \boldsymbol{v} \right] = 0$$



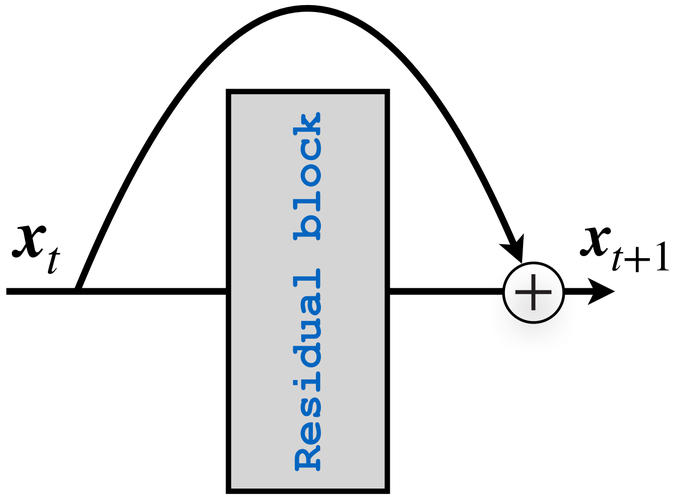
Simple density



Complex density

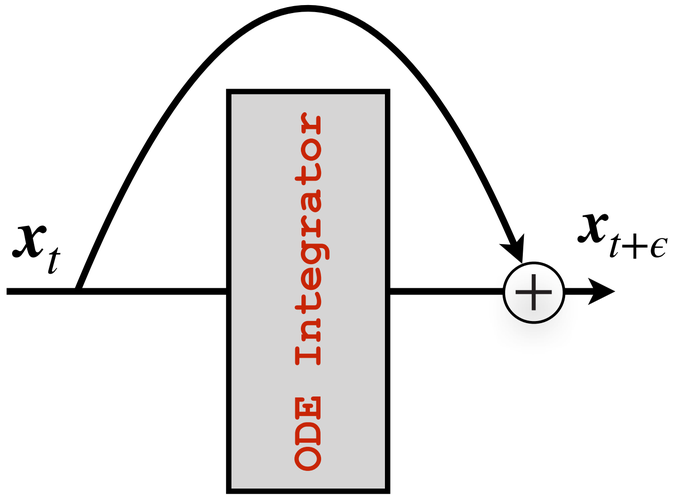
Neural Ordinary Differential Equations

Residual network



$$x_{t+1} = x_t + v(x_t)$$

ODE integration



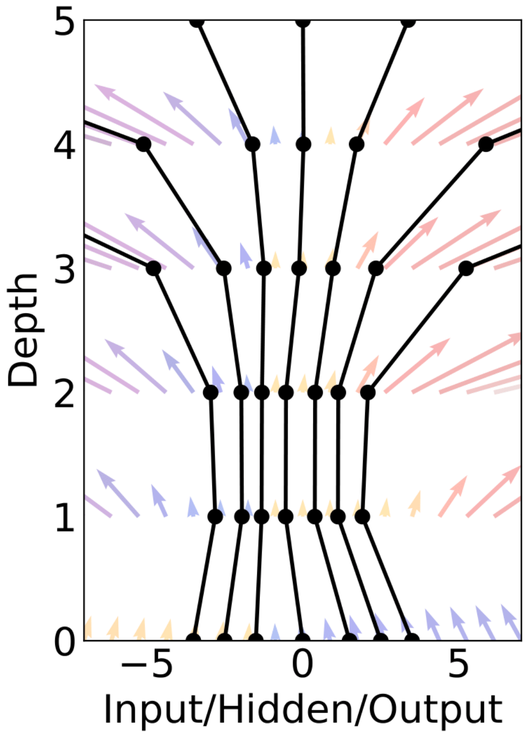
$$dx/dt = v(x)$$

Chen et al, 1806.07366

Harbor el al 1705.03341
Lu et al 1710.10121,
E Commun. Math. Stat 17'...

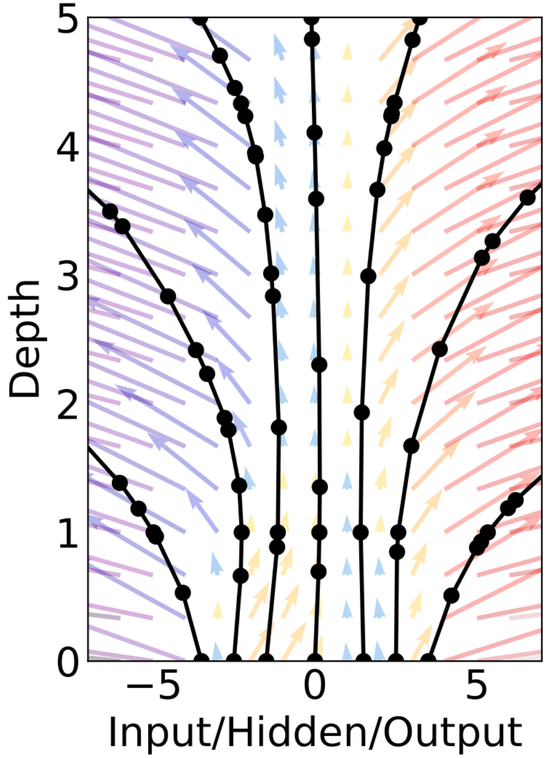
Neural Ordinary Differential Equations

Residual network



$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{v}(\boldsymbol{x}_t)$$

ODE integration



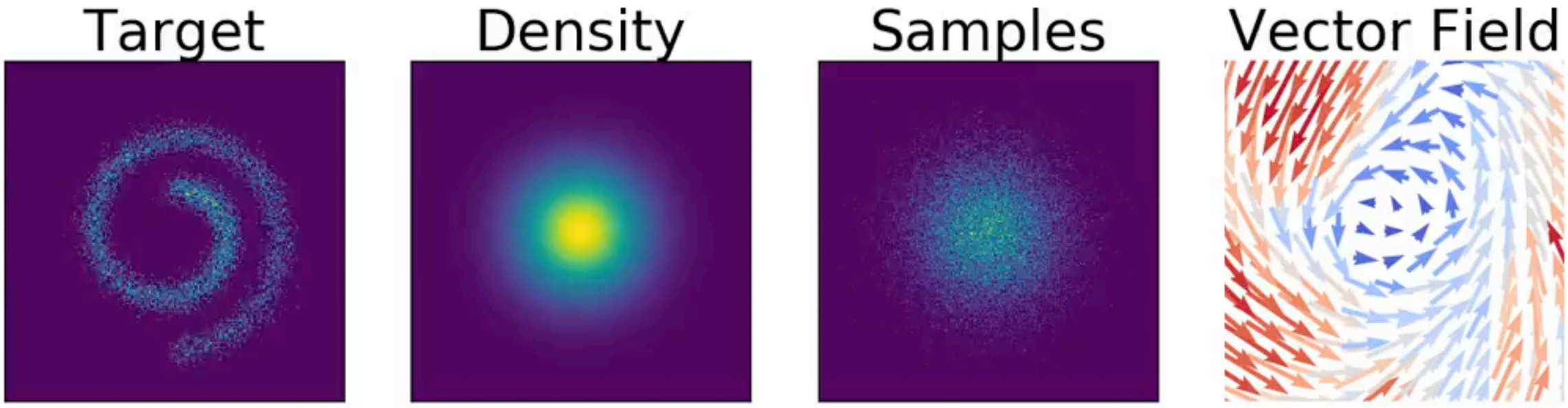
$$d\boldsymbol{x}/dt = \boldsymbol{v}(\boldsymbol{x})$$

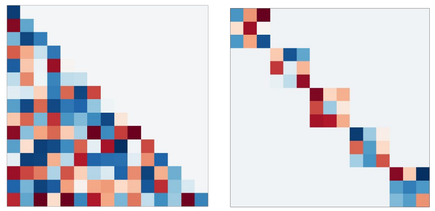
Chen et al, 1806.07366

Harbor el al 1705.03341
Lu et al 1710.10121,
E Commun. Math. Stat 17'...

Continuous normalizing flows implemented with NeuralODE

Chen et al, 1806.07366, Grathwohl et al 1810.01367

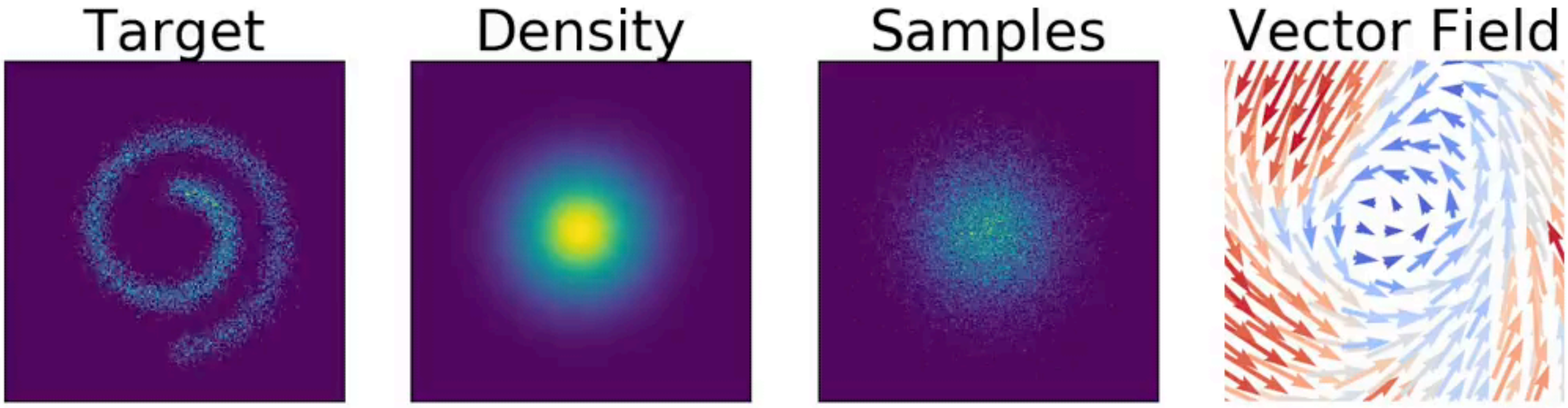


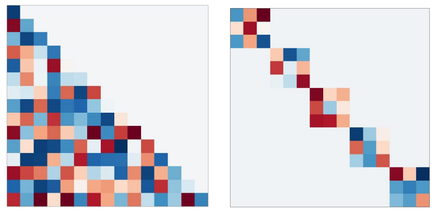


Continuous normalizing flow have no structural constraints on the transformation Jacobian

Continuous normalizing flows implemented with NeuralODE

Chen et al, 1806.07366, Grathwohl et al 1810.01367





Continuous normalizing flow have no structural constraints on the transformation Jacobian

The two use cases

(a) Density estimation

$$\mathcal{N}(\boldsymbol{x}) \longleftarrow \int_T^0 \dot{\boldsymbol{x}}\, dt$$

"learn from data"

$$\mathscr{L} = -\,\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right]$$

(b) Variational free energy

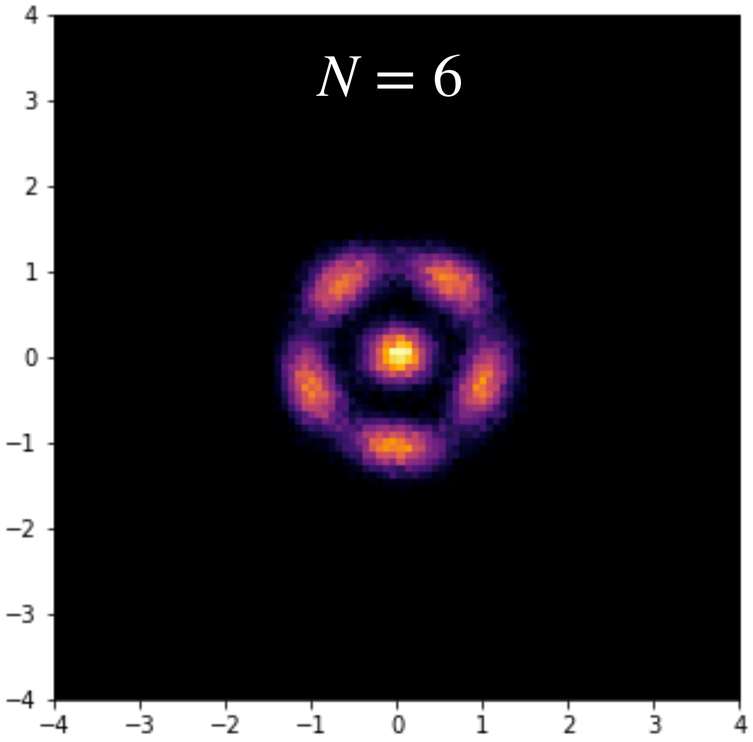
$$\mathcal{N}(\boldsymbol{x}) \longrightarrow \int_0^T \dot{\boldsymbol{x}}\, dt \longrightarrow \frac{e^{-E(\boldsymbol{x})}}{Z}$$

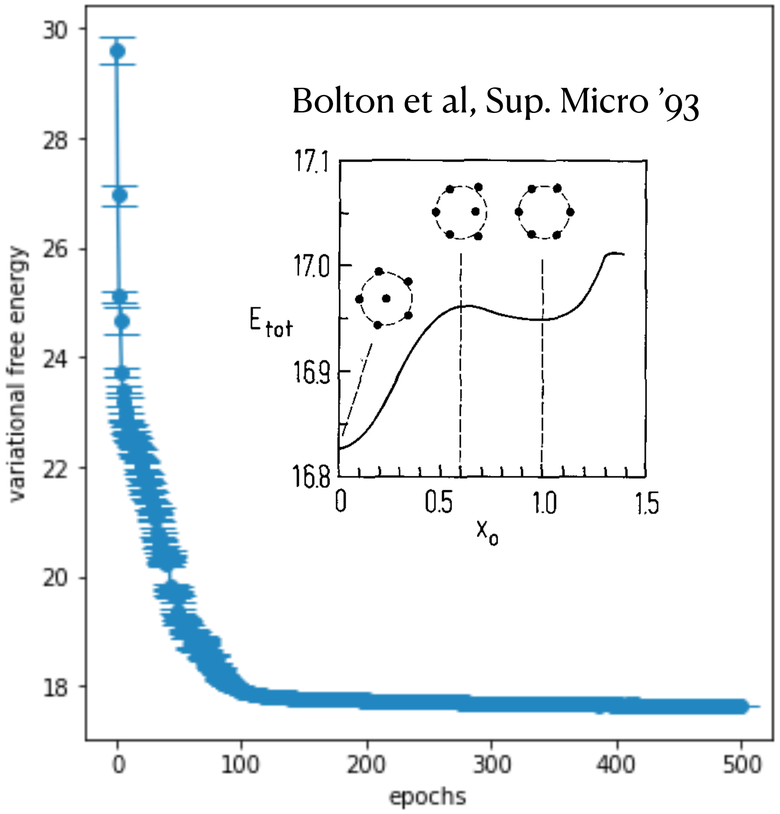
"learn from Hamiltonian"

$$F = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[k_B T \ln p(\boldsymbol{x}) + H(\boldsymbol{x})\right]$$

Example: Classical Coulomb gas in a harmonic trap

$$H = \sum_{i<j} \frac{1}{|\boldsymbol{x}_i - \boldsymbol{x}_j|} + \sum_i^N \boldsymbol{x}_i^2$$



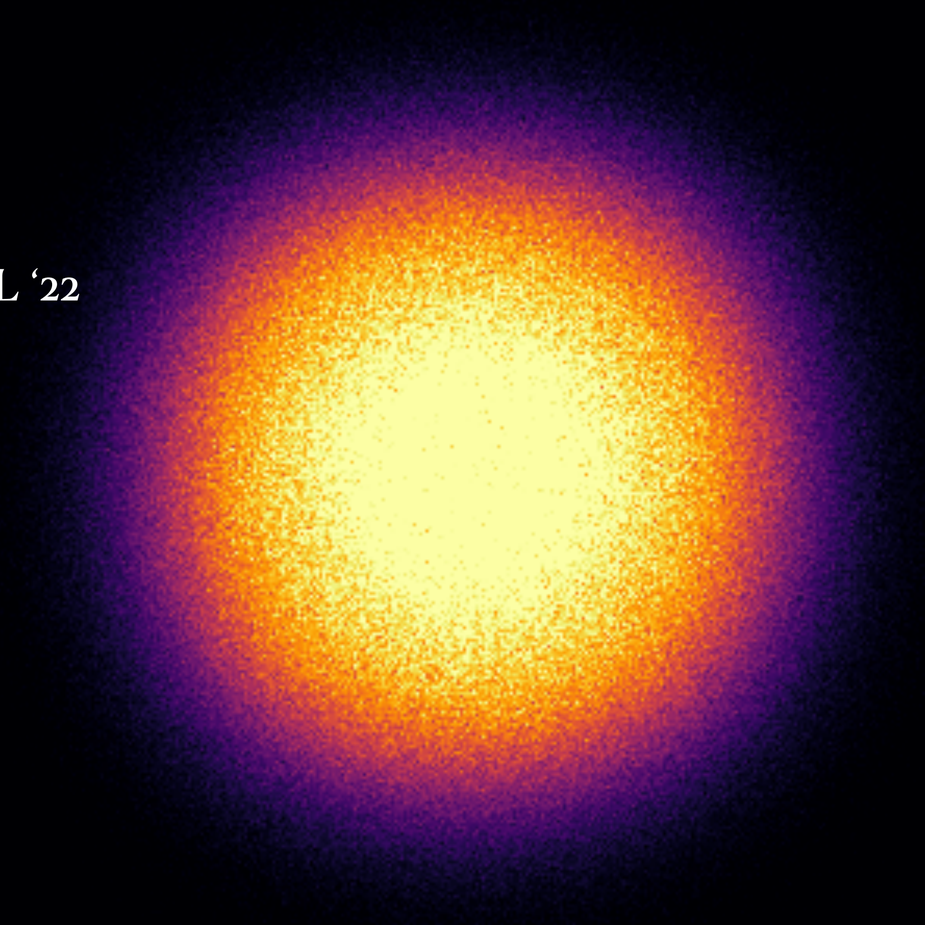


https://github.com/fermiflow/FermiFlow/blob/github/classical_coulomb_gas.ipynb

Fermi Flow

Xie, Zhang, LW, 2105.08644, JML ‘22

github.com/fermiflow

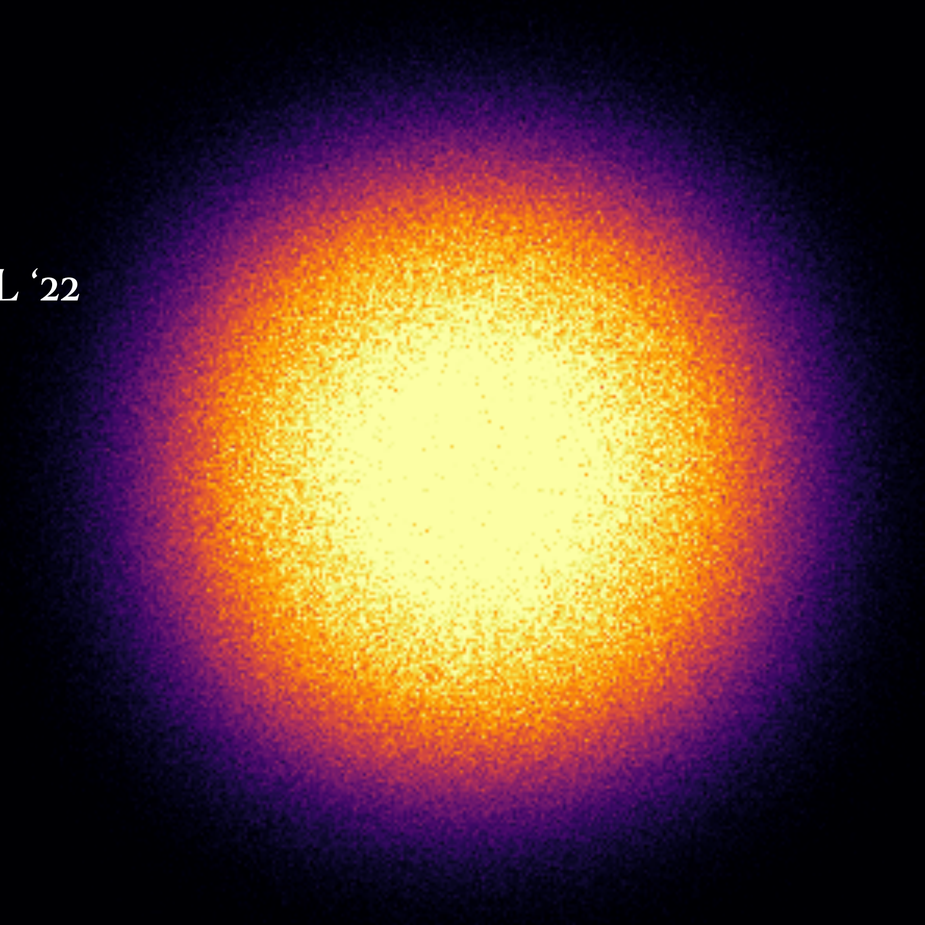


Continuous flow of electron density in a quantum dot

Fermi Flow

Xie, Zhang, LW, 2105.08644, JML ‘22

github.com/fermiflow



Continuous flow of electron density in a quantum dot

Training: Monte Carlo Gradient Estimators

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\theta}}} \left[f(\boldsymbol{x}) \right]$$

Review: 1906.10652

Reinforcement learning
Variational inference
Variational Monte Carlo
Variational quantum algorithms
...

Score function estimator (REINFORCE)

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x} \sim p_\theta} \left[f(\boldsymbol{x}) \right] = \mathbb{E}_{\boldsymbol{x} \sim p_\theta} \left[f(\boldsymbol{x}) \nabla_\theta \ln p_\theta(\boldsymbol{x}) \right]$$

Pathwise estimator (Reparametrization trick) $\boldsymbol{x} = g_\theta(\boldsymbol{z})$

$$\nabla_\theta \mathbb{E}_{\boldsymbol{x} \sim p_\theta} \left[f(\boldsymbol{x}) \right] = \mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z})} \left[\nabla_\theta f(g_\theta(\boldsymbol{z})) \right]$$

10.1 Guidance in Choosing Gradient Estimators

With so many competing approaches, we offer our rules of thumb in choosing an estimator, which follow the intuition we developed throughout the paper:

- If our estimation problem involves continuous functions and measures that are continuous in the domain, then using the pathwise estimator is a good default. It is relatively easy to implement and a default implementation, one without other variance reduction, will typically have variance that is low enough so as not to interfere with the optimisation.
- If the cost function is not differentiable or a black-box function then the score-function or the measure-valued gradients are available. If the number of parameters is low, then the measure-valued gradient will typically have lower variance and would be preferred. But if we have a high-dimensional parameter set, then the score function estimator should be used.
- If we have no control over the number of times we can evaluate a black-box cost function, effectively only allowing a single evaluation of it, then the score function is the only estimator of the three we reviewed that is applicable.
- The score function estimator should, by default, always be implemented with at least a basic variance reduction. The simplest option is to use a baseline control variate estimated with a running average of the cost value.
- When using the score-function estimator, some attention should be paid to the dynamic range of the cost function and its variance, and to find ways to keep its value bounded within a reasonable range, e.g., transforming the cost so that it is zero mean, or using a baseline.
- For all estimators, track the variance of the gradients if possible and address high variance by using a larger number of samples from the measure, decreasing the learning rate, or clipping the gradient values. It may also be useful to restrict the range of some parameters to avoid extreme values, e.g., by clipping them to a desired interval.
- The measure-valued gradient should be used with some coupling method for variance reduction. Coupling strategies that exploit relationships between the positive and negative components of the density decomposition, and which have shared sampling paths, are known for the commonly-used distributions.
- If we have several unbiased gradient estimators, a convex combination of them might have lower variance than any of the individual estimators.
- If the measure is discrete on its domain then the score-function or measure-valued gradient are available. The choice will again depend on the dimensionality of the parameter space.
- In all cases, we strongly recommend having a broad set of tests to verify the unbiasedness of the gradient estimator when implemented.

Mohamed et al, 1906.10652

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\theta}}} \left[f(\boldsymbol{x}) \right]$$

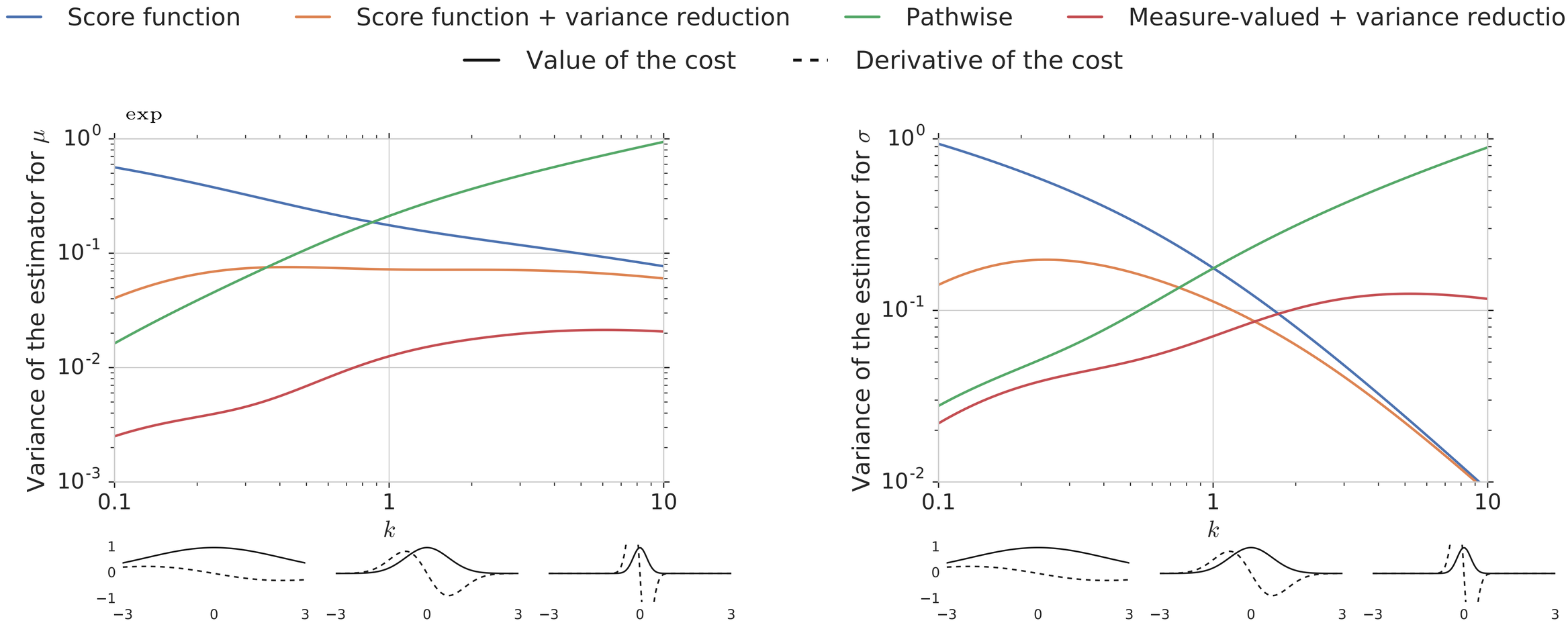
When to use which ?

More discussions

Roeder et al, 1703.09194

Vaitl et al 2206.09016, 2207.08219

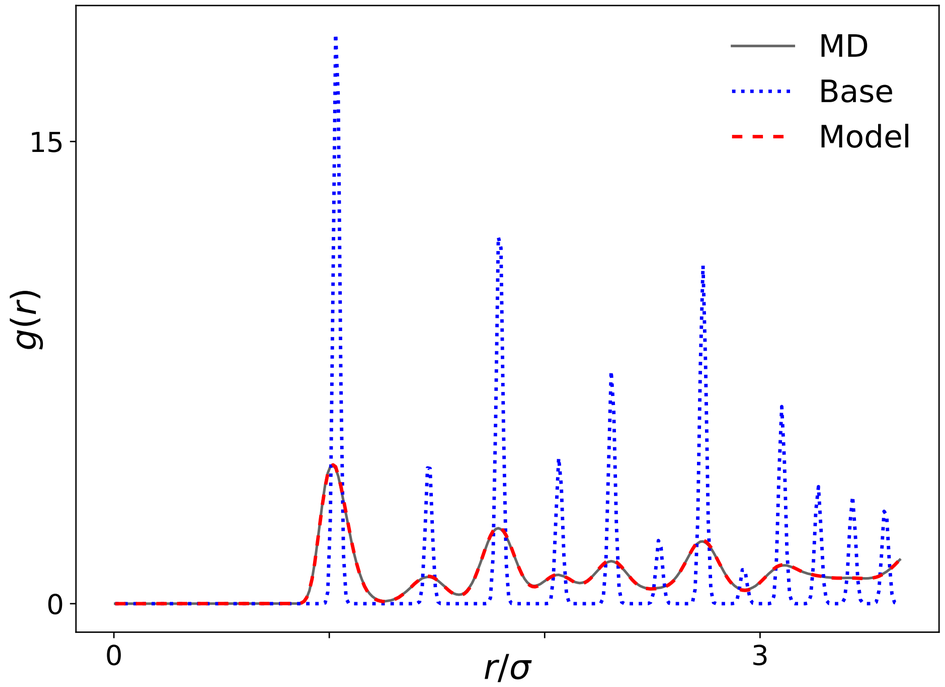
$$\eta = \nabla_\theta \int \mathcal{N}(x|\mu, \sigma^2) f(x; k) dx; \quad \theta \in \{\mu, \sigma\}$$



https://github.com/deepmind/mc_gradients Mohamed et al, 1906.10652

Case study: Normalizing flow for atomic solids

Variational free energy with a really deep (and a bit awkward) permutation equivariant flow



System	N	LFEP	LBAR	MBAR
LJ	256	3.10800(28)	3.10797(1)	3.10798(9)
LJ	500	3.12300(41)	3.12264(2)	3.12262(10)

$$\ln Z = \ln \mathbb{E}_{x\sim p(x)} \left[e^{-\beta E(x) - \ln p(x)} \right]$$

free energy perturbation (Zwanzig 1954)

$$\ln Z_B - \ln Z_A = \ln \mathbb{E}_A \left[e^{-\beta (E_B - E_A)} \right]$$

Wirnsberger et al, 2111.08696 https://github.com/deepmind/flows_for_atomic_solids

Normalizing flow for atomic solids

F. Hardware details and computational cost

For our flow experiments, we used 16 A100 GPUs to train each model on the bigger systems (512-particle mW and 500-particle LJ). It took approximately 3 weeks of training to reach convergence of the free-energy estimates. Obtaining 2M samples for evaluation took approximately 12 hours on 8 V100 GPUs for each of these models.

For each baseline MBAR estimate, we performed 100 separate simulations for LJ and 200 for mW, corresponding to the number of stages employed. These simulations were performed with LAMMPS [8] and each of them ran on multiple CPU cores communicating via MPI. We used 4 cores for the 64-particle and 216-particle mW experiments and 8 cores for all other systems. The MD simulations completed after approximately 11 and 14 hours for LJ (256 and 500 particles), and 7, 20 and 48 hours for mW (64, 216 and 512 particles). To evaluate the energy matrix for a single MBAR

Will the score-function gradient estimator do better?

Heavy lifting is mostly due to preserving permutation. But, does it really matter?

Diffusion models

Denoising score matching Vincent 2011

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})}[\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2].$$

Diffusion generative model

Sohl-Dickstein et al, 1503.03585

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]$$

Song et al, 1907.05600, Ho et al, 2006.11239

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t}\underbrace{\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0}\underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t}||\underbrace{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)}_{\text{score of diffused data sample}}||_2^2$$

A tale of three equations

Langevin equation (SDE)

$$q(\boldsymbol{x}_{t+dt}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_t + \boldsymbol{f}dt, 2TdtI) \qquad \text{or} \quad \boldsymbol{x}_{t+dt} - \boldsymbol{x}_t = \boldsymbol{f}dt + \sqrt{2Tdt}\,\mathcal{N}(0,1)$$

Fokker-Planck equation (PDE)

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \left[p(\boldsymbol{x},t)\boldsymbol{f}\right] - T\nabla^2 p(\boldsymbol{x},t) = 0$$

"Particle method" (ODE)

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f} - T\nabla \ln p(\boldsymbol{x},t)$$

(Another way to reverse the diffusion is via the reverse-time SDE Anderson 1982)

Maoutsa et al, 2006.00702
Song et al, 2011.13456

$$\mathcal{P}(\vec{x},t)=\int \mathrm{d}^3\vec{x}\,'\left(\frac{1}{4\pi D\epsilon}\right)^{3/2}\exp\left[-\frac{\left(\vec{x}-\vec{x}\,'-\epsilon\vec{v}(\vec{x}\,')\right)^2}{4D\epsilon}\right]\mathcal{P}(\vec{x}\,',t-\epsilon), \tag{9.18}$$

and simplified by the change of variables,

$$\begin{aligned}\vec{y}\ &=\vec{x}\,'+\epsilon\vec{v}(\vec{x}\,')-\vec{x}\quad\Longrightarrow\\ \mathrm{d}^3\vec{y}\ &=\mathrm{d}^3\vec{x}\,'\left(1+\epsilon\nabla\cdot\vec{v}(\vec{x}')\right)=\mathrm{d}^3\vec{x}\,'\left(1+\epsilon\nabla\cdot\vec{v}(\vec{x})+\mathcal{O}(\epsilon^2)\right).\end{aligned} \tag{9.19}$$

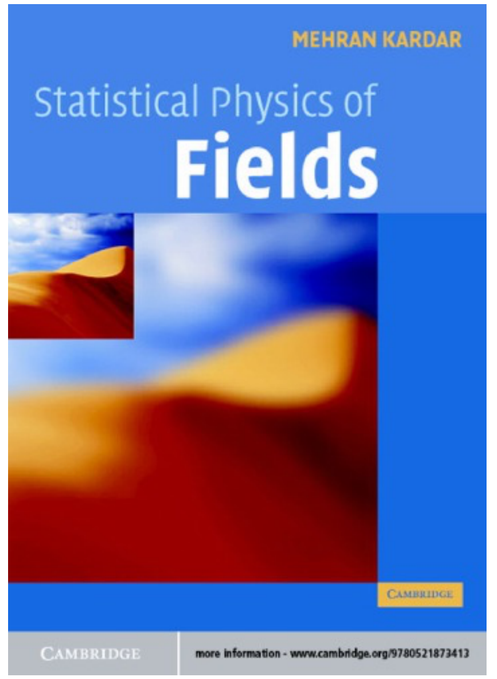
Keeping only terms at order of ϵ, we obtain

$$\begin{aligned}\mathcal{P}(\vec{x},t)&=\left[1-\epsilon\nabla\cdot\vec{v}(\vec{x})\right]\int \mathrm{d}^3\vec{y}\left(\frac{1}{4\pi D\epsilon}\right)^{3/2}\mathrm{e}^{-\frac{y^2}{4D\epsilon}}\ \mathcal{P}(\vec{x}+\vec{y}-\epsilon\vec{v}(\vec{x}),t-\epsilon)\\ &=\left[1-\epsilon\nabla\cdot\vec{v}(\vec{x})\right]\ \int \mathrm{d}^3\vec{y}\left(\frac{1}{4\pi D\epsilon}\right)^{3/2}\mathrm{e}^{-\frac{y^2}{4D\epsilon}}\\ &\times\left[\mathcal{P}(\vec{x},t)+(\vec{y}-\epsilon\vec{v}(\vec{x}))\cdot\nabla\mathcal{P}+\frac{y_iy_j-2\epsilon y_iv_j+\epsilon^2v_iv_j}{2}\nabla_i\nabla_j\mathcal{P}-\epsilon\frac{\partial\mathcal{P}}{\partial t}+\mathcal{O}(\epsilon^2)\right]\\ &=\left[1-\epsilon\nabla\cdot\vec{v}(\vec{x})\right]\left[\mathcal{P}-\epsilon\vec{v}\cdot\nabla+\epsilon D\nabla^2\mathcal{P}-\epsilon\frac{\partial\mathcal{P}}{\partial t}+\mathcal{O}(\epsilon^2)\right].\end{aligned} \tag{9.20}$$

Equating terms at order of ϵ leads to the *Fokker–Planck equation*,

$$\frac{\partial\mathcal{P}}{\partial t}+\nabla\cdot\vec{J}=0,\quad\text{with}\quad\vec{J}=\vec{v}\,\mathcal{P}-D\nabla\mathcal{P}. \tag{9.21}$$

from Langevin to Fokker-Planck



Lessons from diffusion models

Continuous normalizing flow still has great potential!

Going beyond maximum likelihood estimation training (even if we can)

Break the loss into small pieces, sample them (kind of layer-wise training)

https://blog.alexalemi.com/diffusion.html

The conditional trick (originated from denoising score matching Vincent 2011)

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t} || \underbrace{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)}_{\text{score of diffused data sample}} ||_2^2$$

https://cvpr2022-tutorial-diffusion-models.github.io/

Lessons from diffusion models

Continuous normalizing flow still has great potential!

Going beyond maximum likelihood estimation training (even if we can)

Break the loss into small pieces, sample them (kind of layer-wise training)

https://blog.alexalemi.com/diffusion.html

The conditional trick (originated from denoising score matching Vincent 2011)

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t)}}_{\text{diffused data } \mathbf{x}_t} \underbrace{||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{\text{score of diffused data (marginal)}}||_2^2$$

https://cvpr2022-tutorial-diffusion-models.github.io/

Claim:

$$\mathbb{E}_{x\sim q(x)}\,|\,s_\theta(x)-\nabla_x \ln q(x)\,|^2=\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{x\sim q(x|x_0)}\,|\,s_\theta(x)-\nabla_x \ln q(x\,|\,x_0)\,|^2+\text{const}.$$

↓

Independent of θ

$$q(x)=\int q(x\,|\,x_0)q_0(x_0)dx_0$$

Proof:

$$\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{x\sim q(x|x_0)}\,|\,s\,|^2=\int dx_0\int dx q_0(x_0)q(x\,|\,x_0)\,|\,s\,|^2=\int dx q(x)\,|\,s\,|^2=\mathbb{E}_{x\sim q(x)}\,|\,s^2\,|$$

$$\begin{aligned}\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{x\sim q(x|x_0)}[s\cdot\nabla \ln q(x\,|\,x_0)]&=\int dx_0\int dx q_0(x_0)q(x\,|\,x_0)\frac{s\cdot\nabla q(x\,|\,x_0)}{q(x\,|\,x_0)}\\&=\int dx_0\int dx q_0(x_0)s\cdot\nabla q(x\,|\,x_0)\\&=\int dx s\cdot\nabla q(x)=\mathbb{E}_{x\sim q(x)}[s\cdot\nabla \ln q(x)]\end{aligned}$$

要让$\boldsymbol{s_\theta(x_t,t)}$等于$\nabla_{\boldsymbol{x_t}}\log p(\boldsymbol{x_t}|\boldsymbol{x_0})$的加权平均【即$\nabla_{\boldsymbol{x_t}}\log p(\boldsymbol{x_t})$】只需要最小化$\|\boldsymbol{s_\theta(x_t,t)}-\nabla_{\boldsymbol{x_t}}\log p(\boldsymbol{x_t}|\boldsymbol{x_0})\|^2$的加权平均 https://spaces.ac.cn/archives/9209

Flow matching

$p(\boldsymbol{x},0) = \mathcal{N}(0,I)$

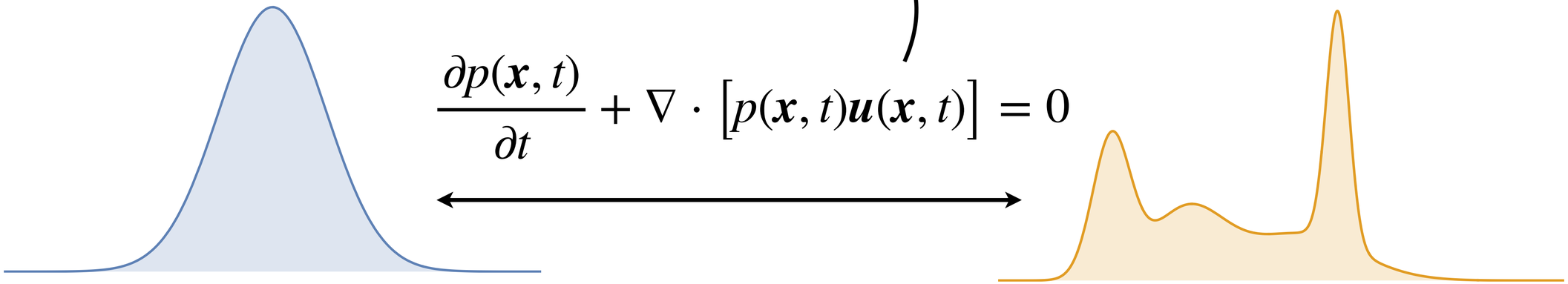
base distribution

ground truth

velocity field

$p(\boldsymbol{x},1) = q(\boldsymbol{x})$

data distribution



$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \left[p(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t) \right] = 0$$

$$\mathscr{L}_{\text{FM}} = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x},t)} \left| \boldsymbol{v}_\theta(\boldsymbol{x},t) - \boldsymbol{u}(\boldsymbol{x},t) \right|^2$$

Albergo et al, 2209.15571, Lipman et al, 2210.02747

Conditional flow matching

$$\frac{\partial p(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t)}{\partial t} + \nabla \cdot \left[p(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t) \boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t) \right] = 0$$

$$p(\boldsymbol{x}, t) = \int p(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t)\, q(\boldsymbol{x}_1) d\boldsymbol{x}_1 \qquad p(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) = \int p(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t)\boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t)\, q(\boldsymbol{x}_1) d\boldsymbol{x}_1$$

$$\mathscr{L}_{\text{CFM}} = \mathbb{E}_{t\sim\mathscr{U}(0,1)} \mathbb{E}_{\boldsymbol{x}_1\sim q(\boldsymbol{x}_1)} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{x}_1,t)} \left| \boldsymbol{v}_\theta(\boldsymbol{x}, t) - \boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t) \right|^2$$

$$\nabla_\theta \mathscr{L}_{\text{FM}} = \nabla_\theta \mathscr{L}_{\text{CFM}}$$

an example:

$$\boldsymbol{x} = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$$
$$\boldsymbol{x}_0 \sim \mathscr{N}(0, I)$$

$$p(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t) = \mathscr{N}\left(t\boldsymbol{x}_1, (1-t)^2\right)$$
$$\boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1, t) = d\boldsymbol{x}/dt = \boldsymbol{x}_1 - \boldsymbol{x}_0$$

Claim: $\nabla_\theta \mathscr{L}_{\text{FM}} = \nabla_\theta \mathscr{L}_{\text{CFM}}$

where $\mathscr{L}_{\text{FM}} = \mathbb{E}_{t\sim\mathscr{U}(0,1)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x},t)} \left|\boldsymbol{v}_\theta(\boldsymbol{x},t) - \boldsymbol{u}(\boldsymbol{x},t)\right|^2$

$$\mathscr{L}_{\text{CFM}} = \mathbb{E}_{t\sim\mathscr{U}(0,1)}\mathbb{E}_{\boldsymbol{x}_1\sim q(\boldsymbol{x}_1)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{x}_1,t)} \left|\boldsymbol{v}_\theta(\boldsymbol{x},t) - \boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\right|^2$$

$$p(\boldsymbol{x},t) = \int p(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\, q(\boldsymbol{x}_1) d\boldsymbol{x}_1 \qquad p(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t) = \int p(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\, q(\boldsymbol{x}_1) d\boldsymbol{x}_1$$

Proof:

$$\mathbb{E}_{\boldsymbol{x}_1\sim q(\boldsymbol{x}_1)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{x}_1,t)} \left|\boldsymbol{v}_\theta\right|^2 = \int d\boldsymbol{x}_1 \int d\boldsymbol{x} q(\boldsymbol{x}_1) p(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\,|\,\boldsymbol{v}_\theta|^2 = \int d\boldsymbol{x} p(\boldsymbol{x},t)\,|\,\boldsymbol{v}_\theta|^2 = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x},t)} \left|\boldsymbol{v}_\theta\right|^2$$

$$\mathbb{E}_{\boldsymbol{x}_1\sim q(\boldsymbol{x}_1)}\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}|\boldsymbol{x}_1,t)} \left[\boldsymbol{v}_\theta \cdot \boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\right] = \int d\boldsymbol{x}_1 \int d\boldsymbol{x} q(\boldsymbol{x}_1) p(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t) \left[\boldsymbol{v}_\theta \cdot \boldsymbol{u}(\boldsymbol{x}\,|\,\boldsymbol{x}_1,t)\right]$$

$$= \int d\boldsymbol{x} p(\boldsymbol{x},t) \boldsymbol{v}_\theta \cdot \boldsymbol{u}(\boldsymbol{x},t) = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x},t)} \left[\boldsymbol{v}_\theta \cdot \boldsymbol{u}(\boldsymbol{x},t)\right]$$

Flow matching is all you need!

This framework contains various diffusion models as special cases

Greater freedom (and optimal transport theory) in the interpolation path

400x speedup compared to continuous normalizing flow (Albergo et al, 2209.15571)

Surpasses diffusion model on Imagenet in likelihood and sample quality (Lipman et al, 2210.02747)

Fun to try: flow matching for computing free energy difference

Fun to try: Train Riemannian flows with this

Part II Optimal transport and Riemannian geometry

GAN

Likelihood free simulator

Prone to mode collapse

More tricky to train than others

Performance have been surpassed by diffusion models

I found it is less useful to scientific applications

VAE

Close connection to variational calculus we have just learned

$$p(x) = \frac{e^{-E(x)}}{Z}$$

Variational free energy

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

Variational Bayes

2209.10423

Approximate sampling and estimation of partition functions using neural networks

George T. Cantwell
Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501
gcant@umich.edu

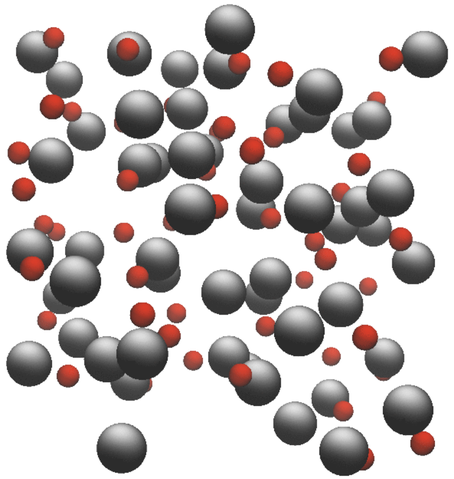
We consider the closely related problems of sampling from a distribution known up to a normalizing constant, and estimating said normalizing constant. We show how variational autoencoders (VAEs) can be applied to this task. In their standard applications, VAEs are trained to fit data drawn from an unknown and intractable distribution. We invert the logic and train the VAE to fit a simple and tractable distribution, on the assumption of a complex and intractable latent distribution, specified up to normalization. This procedure constructs approximations without the use of training data or Markov chain Monte Carlo sampling. We illustrate our method on three examples: the Ising model, graph clustering, and ranking.

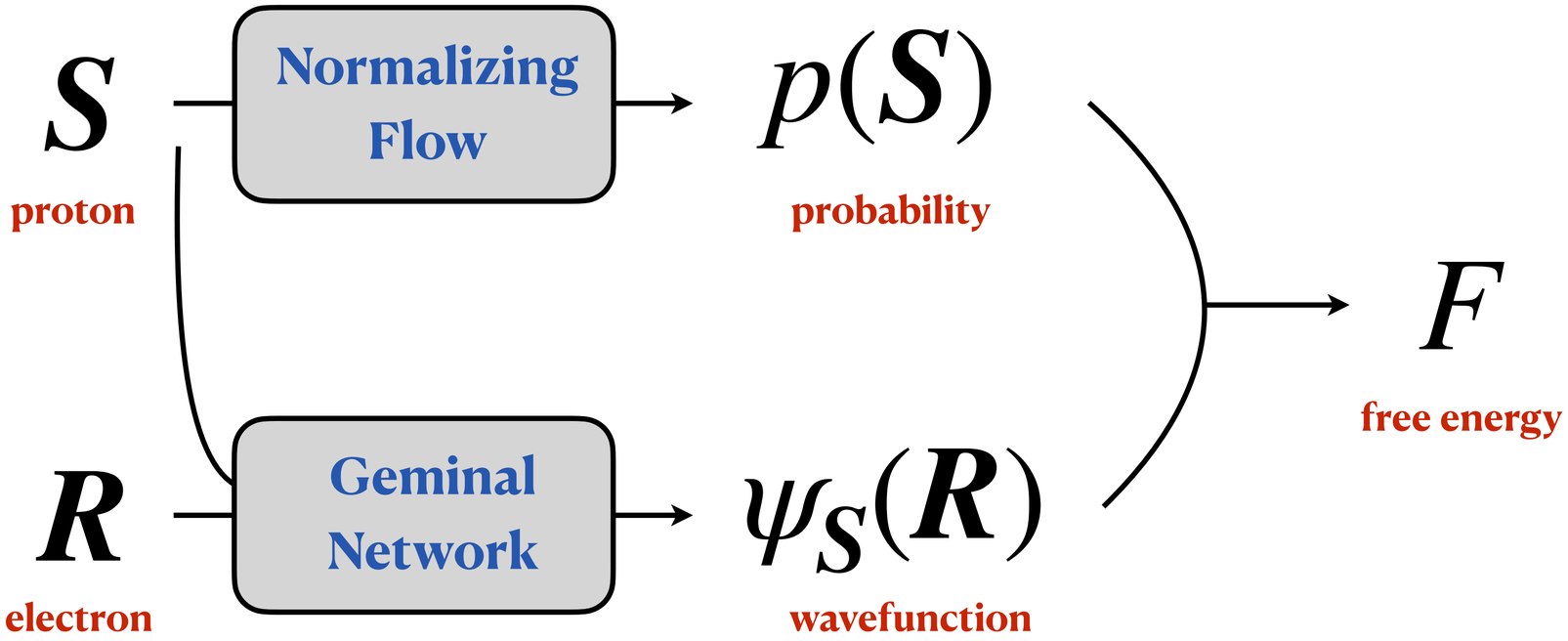
Deep variational free energy for dense hydrogen

Xie, Li, Wang, Zhang, LW, 2209.06095

$$F = \mathop{\mathbb{E}}_{S\sim p(S)} \left[k_B T \ln p(S) + \mathop{\mathbb{E}}_{R\sim|\psi_S(R)|^2} \left[\frac{H\psi_S(R)}{\psi_S(R)} \right] \right]$$

Classical protons coupled to ground state electrons



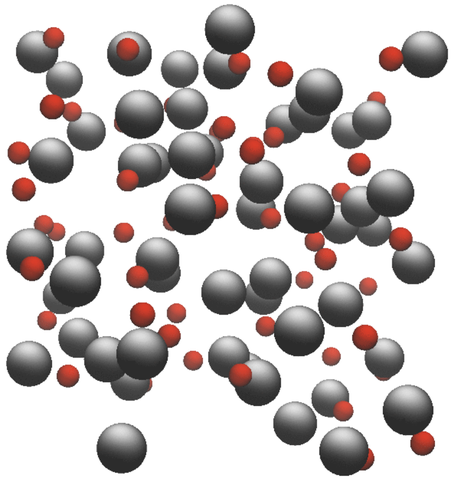


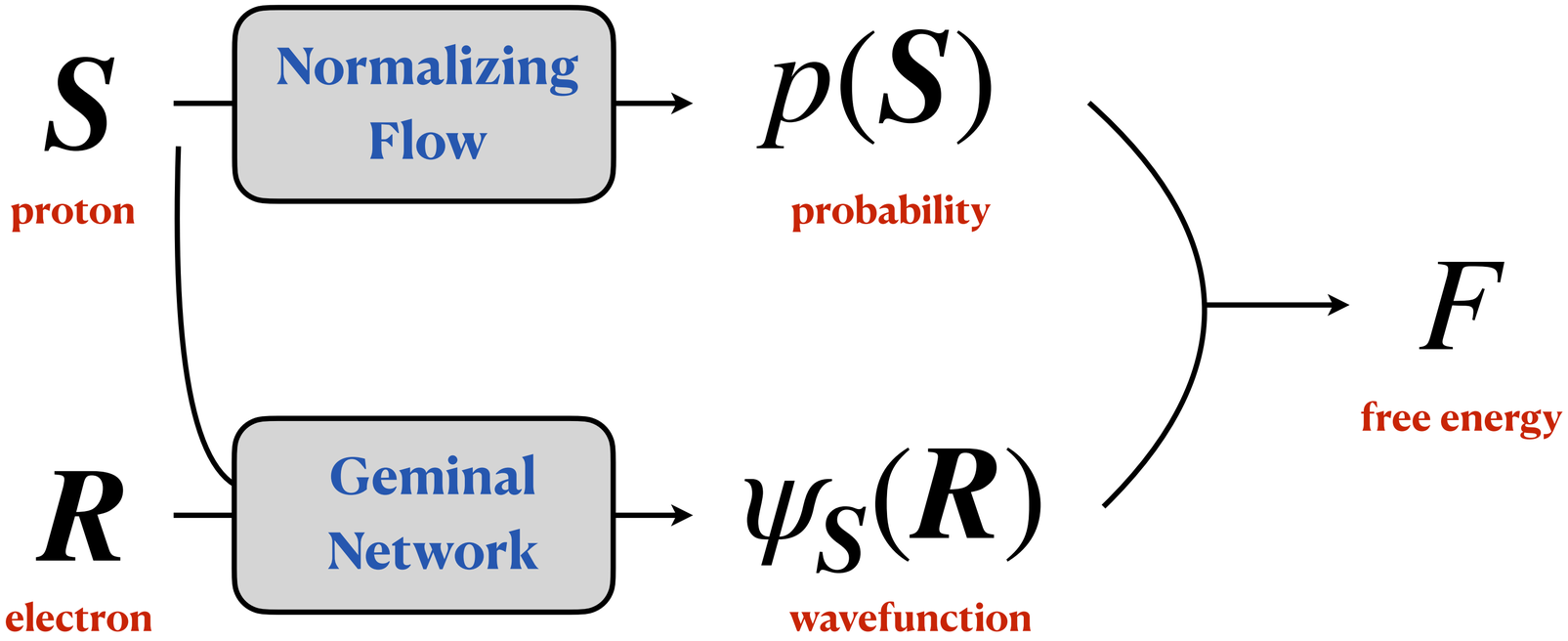
Deep variational free energy for dense hydrogen

Xie, Li, Wang, Zhang, LW, 2209.06095

$$F = \mathop{\mathbb{E}}_{S\sim p(S)} \left[k_B T \ln p(S) + \mathop{\mathbb{E}}_{R\sim|\psi_S(R)|^2} \left[\frac{H\psi_S(R)}{\psi_S(R)} \right] \right]$$

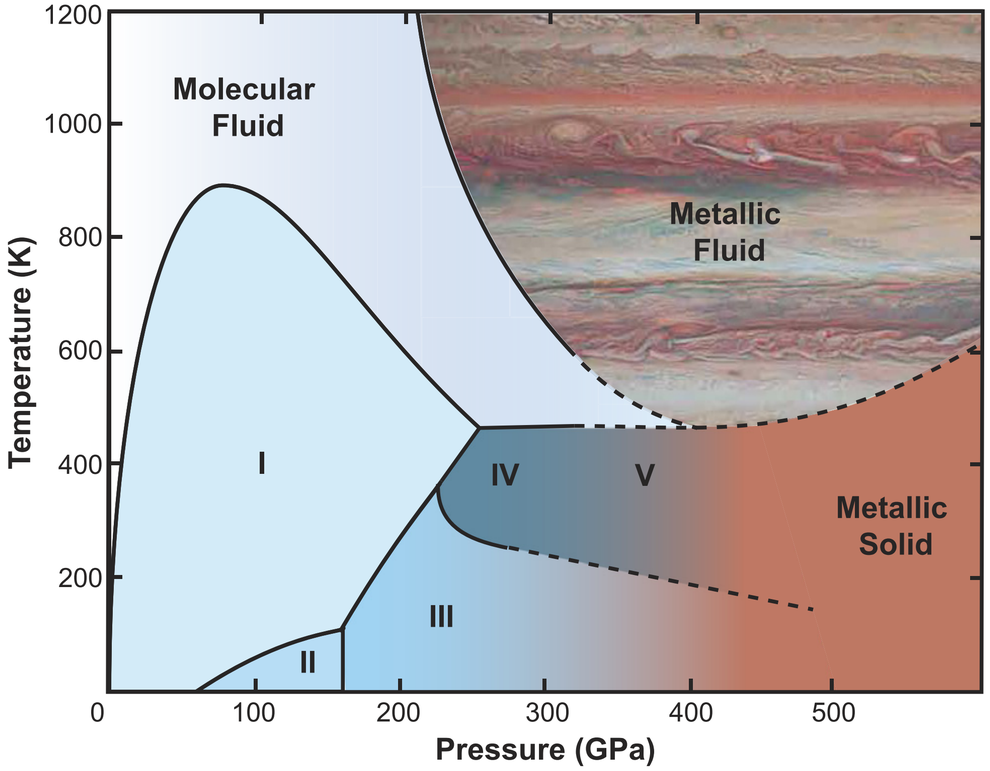
Classical protons coupled to ground state electrons



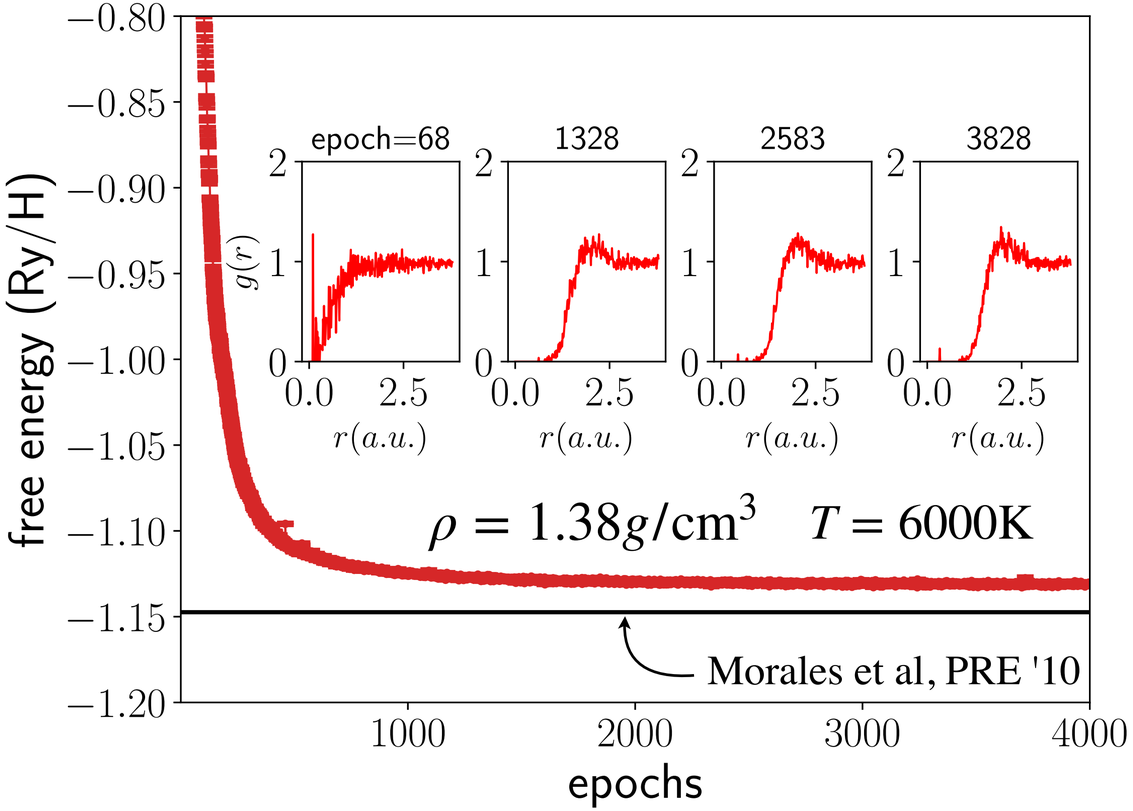


The dense hydrogen problem

Radiat et al, Extremes, 2020

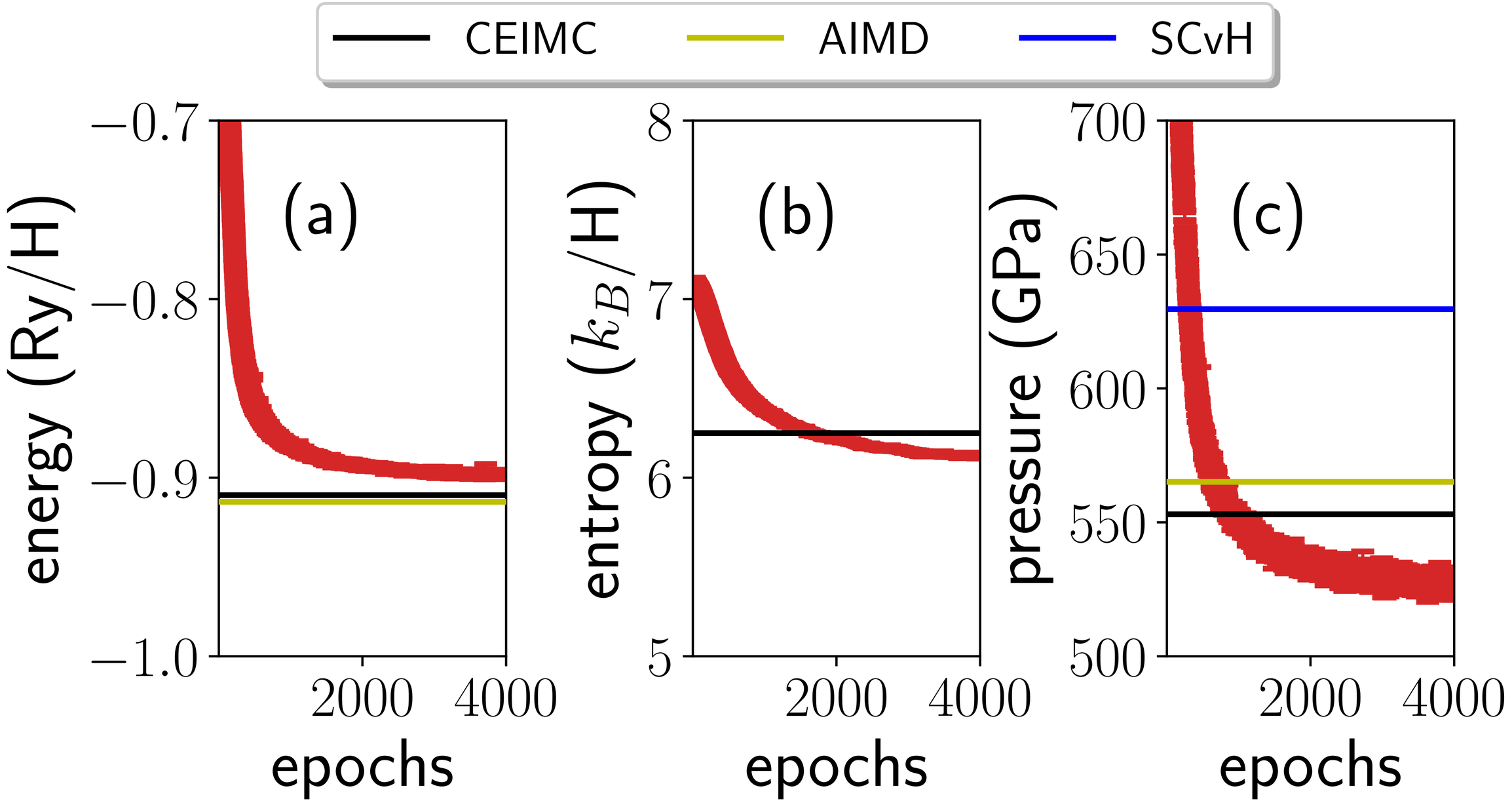


Xie, Li, Wang, Zhang, LW, 2209.06095



Generative model for proton probability density distribution

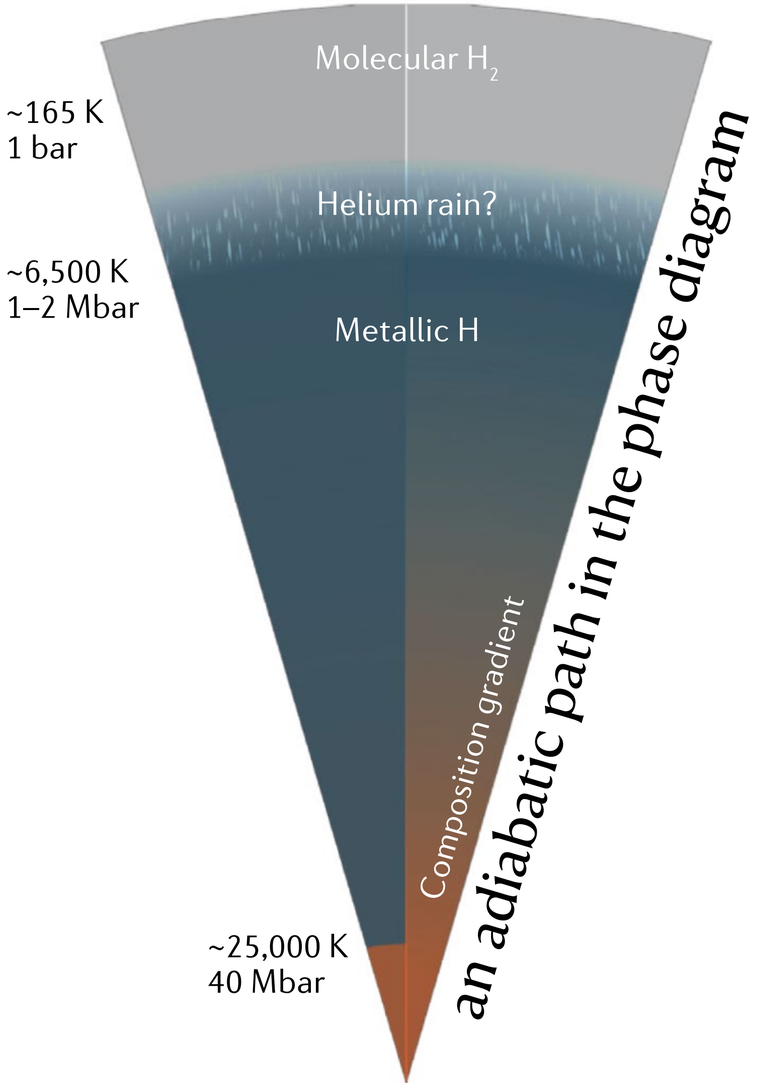
Deep neural network (Ferminet) for electron wavefunction



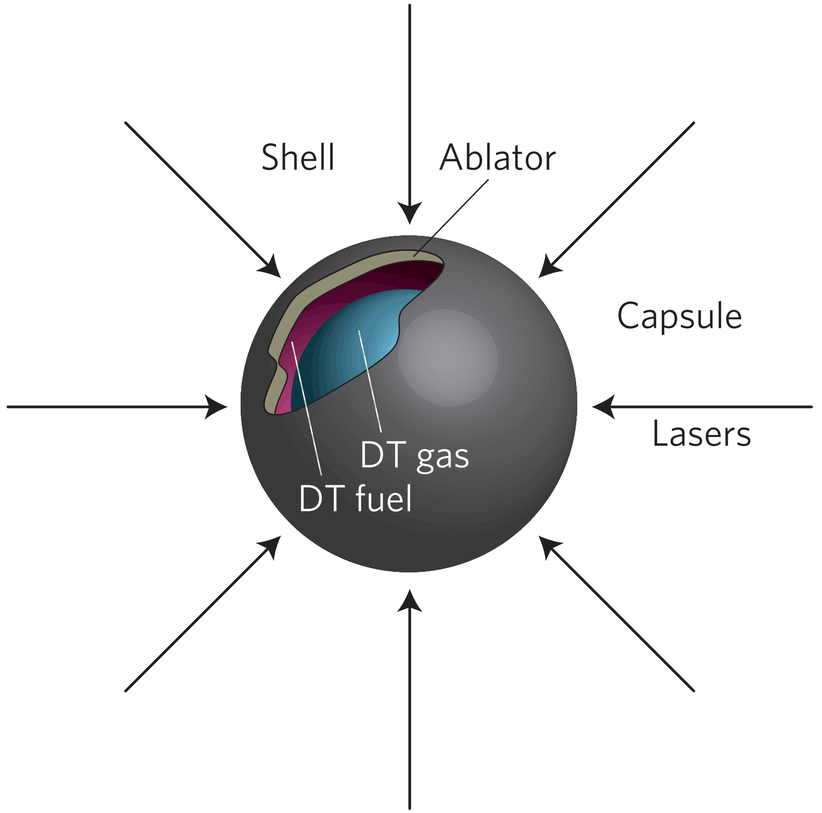
Our predictions will be systematically improved when lowering the variational free energy

Dense hydrogen in the sky and in the lab

Jupiter interior



Inertial confinement fusion



Equation-of-state is the input for hydrodynamics simulations

Issues

Symmetry in generative models

Generative models on a manifold

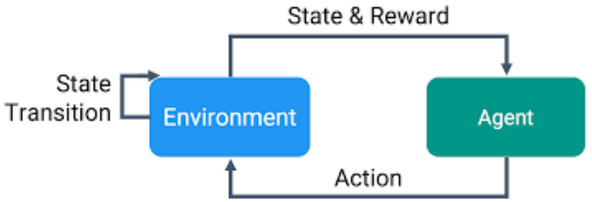
Coarse graining to slow/collective variables

Accelerated sampling in/with generative models

ML accelerated sampling

1. Cheap surrogate function for MC weight Neal 96' Jun. S Liu 01' A recommender engine for MC updates when the surrogate is a generative model: Huang, LW, 1610.02746, Liu, Qi, Meng, Fu, 1610.03137

2. Reinforcement learning the transition kernel: Song et al, 1706.07561, Levy et al 1711.09268, Cusumano-Towner et al 1801.03612, Bojesen, 1808.09095



3. Performs MC in the variationally learned disentangled representation: Wavelet MC, Ismail 03'