Machine Learning the Density of States: Neural Wang-Landau Sampling

Martin Weigel

Institut für Physik, Technische Universität Chemnitz, Germany

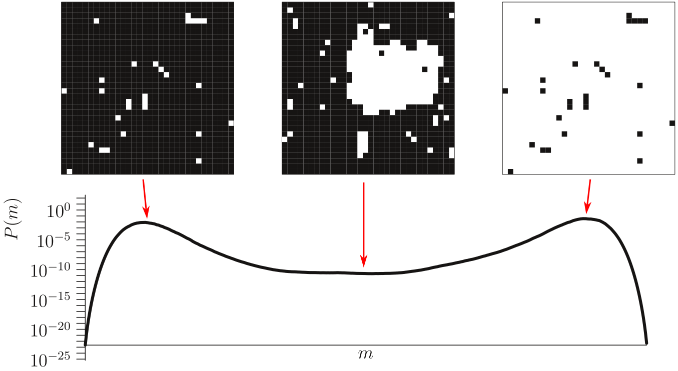
with Moritz Riedel (TU Chemnitz) and Johannes Zierenberg (MPI-DS Göttingen)

25th International NTZ-Workshop on
New Developments in Computational Physics
Universität Leipzig, November 28, 2024.

TECHNISCHE UNIVERSITÄT
IN DER KULTURHAUPTSTADT EUROPAS
CHEMNITZ

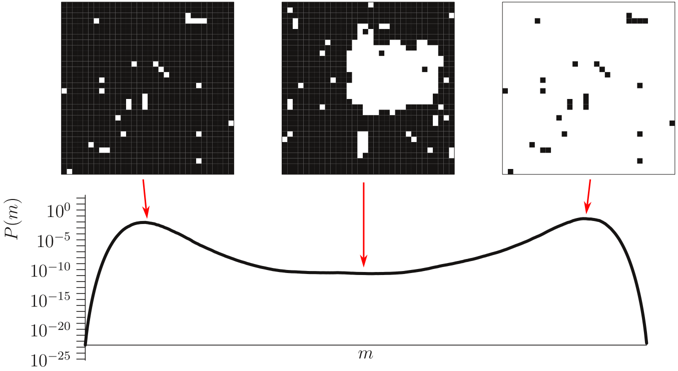
Rare events in statistical physics

Consider for example states in the coexistence region of a first-order phase transition



Rare events in statistical physics

Consider for example states in the coexistence region of a first-order phase transition



These are practically invisible in standard Monte Carlo simulations.

Rare events in statistical physics

Consider for example states in the coexistence region of a first-order phase transition

Other examples:

- transition states in biomolecules
- rare events in distributions of random graphs
- distribution tails in Griffiths phases
- ...

Multicanonical and Wang-Landau simulations

Generalized ensembles

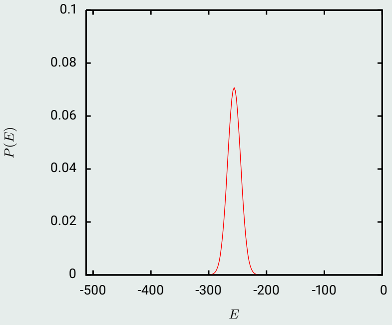
Instead of simulating the canonical distribution,

$$P_\beta(E) = \frac{1}{Z_\beta}\Omega(E)e^{-\beta E},$$

consider using a more general distribution

$$P_{\rm mc}(E) = \frac{\Omega(E)/W(E)}{Z_{\rm muca}} = \frac{\Omega(E)e^{-\omega(E)}}{Z_{\rm muca}},$$

engineered to overcome barriers, improve sampling speed and extend the reweighting range.

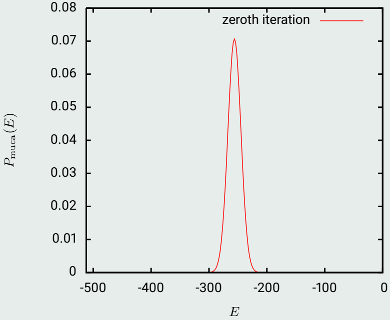


Multicanonical and Wang-Landau simulations

Muca iteration

Determine muca weights/density of states iteratively:

1. Use, e.g., a $K = 0$ canonical simulation to get initial estimate $\hat{S}_0(E) = \ln \hat{\Omega}_0(E)$.

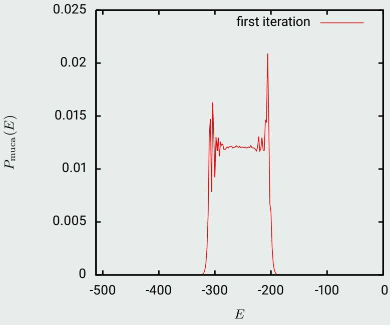


Multicanonical and Wang-Landau simulations

Muca iteration

Determine muca weights/density of states iteratively:

1. Use, e.g., a $K = 0$ canonical simulation to get initial estimate $\hat{S}_0(E) = \ln \hat{\Omega}_0(E)$.
2. Choose multicanonical weights $\omega_1(E) = \hat{S}_0(E)$ for next simulation.

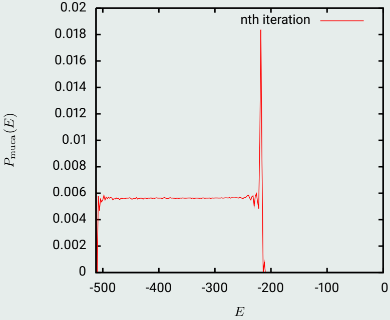


Multicanonical and Wang-Landau simulations

Muca iteration

Determine muca weights/density of states iteratively:

1. Use, e.g., a $K = 0$ canonical simulation to get initial estimate $\hat{S}_0(E) = \ln \hat{\Omega}_0(E)$.
2. Choose multicanonical weights $\omega_1(E) = \hat{S}_0(E)$ for next simulation.
3. Iterate.

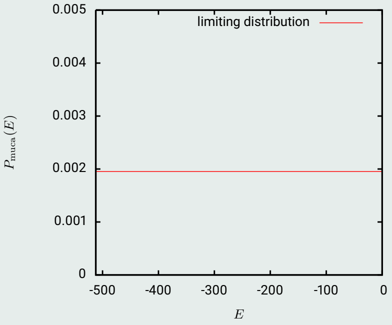


Multicanonical and Wang-Landau simulations

Muca iteration

Advantages:

- always in equilibrium
- arbitrary distributions possible
- system ideally performs an unbiased random walk in energy space $\rightarrow$ fast(er) dynamics

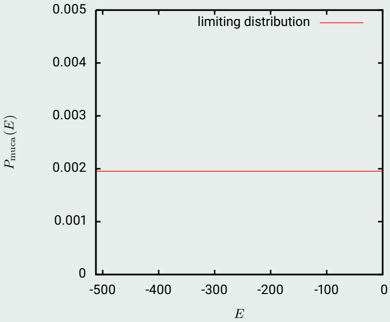


Multicanonical and Wang-Landau simulations

Multicanonical simulations

Variants:

- umbrella sampling, entropic sampling (identical)
- multiple Gaussian modified ensemble
- (broad histogram method)
- transition-matrix Monte Carlo
- metadynamics
- Wang-Landau sampling
- ...



Multicanonical and Wang-Landau simulations

Wang-Landau sampling

Muca weights are updated as

$$\omega_{i+1}(E) - \omega_i(E) = \mathrm{const} + \ln \hat{H}_i(E),$$

i.e., if an energy E is visited more often than others, it receives *less* weight in future iterations.

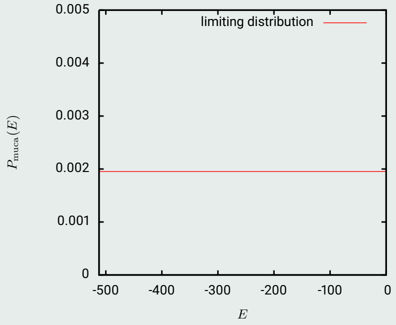
This behavior can be imitated in a one-step iteration: simulate

$$P_{\mathrm{WL}}(E) \propto \Omega(E) e^{-\omega(E)},$$

but update

$$\omega_{i+1}(E) - \omega_i(E) = \phi,$$

each time an energy E is seen.



Multicanonical and Wang-Landau simulations

Wang-Landau sampling

Muca weights are updated as

$$\omega_{i+1}(E) - \omega_i(E) = \text{const} + \ln \hat{H}_i(E),$$

i.e., if an energy E is visited more often than others, it receives *less* weight in future iterations.

This behavior can be imitated in a one-step iteration: simulate

$$P_{\text{WL}}(E) \propto \Omega(E) e^{-\omega(E)},$$

but update

$$\omega_{i+1}(E) - \omega_i(E) = \phi,$$

each time an energy E is seen.

WL iteration

1. Start with $\omega(E) = 0 \; \forall E$.
2. Simulate "sufficiently long" while continuously updating $\omega(E)$.
3. Reduce modification factor, e.g.,

$$\phi \to \phi/2$$

4. Iterate till $\phi < \phi_{\text{thres}}$.

Multicanonical and Wang-Landau simulations

Use and justification

Different possible interpretations of this scheme:

- Rather efficient way of calculating muca weights.
- Standalone algorithm for estimating the density of states (convergence?).
- Violates detailed balance for any $\phi > 0$, but convergence can be proved as a stochastic approximation (instead of MCMC) algorithm for

$$\phi = \frac{t_0}{\max(t, t_0)} \sim \frac{1}{t}, \quad t > t_0$$

instead of

$$\phi = \phi_0 \, 2^{-t}$$

(cf. simulated annealing)

WL iteration

1. Start with $\omega(E) = 0 \, \forall E$.
2. Simulate "sufficiently long" while continuously updating $\omega(E)$.
3. Reduce modification factor, e.g.,

$$\phi \to \phi/2$$

4. Iterate till $\phi < \phi_{\rm thres}$.

Multicanonical and Wang-Landau simulations

Use and justification

Different possible interpretations of this scheme:

- Rather efficient way of calculating muca weights.
- Standalone algorithm for estimating the density of states (convergence?).
- Violates detailed balance for any $\phi > 0$, but convergence can be proved as a stochastic approximation (instead of MCMC) algorithm for

$$\phi = \frac{t_0}{\max(t, t_0)} \sim \frac{1}{t}, \quad t > t_0$$

instead of

$$\phi = \phi_0 \, 2^{-t}$$

(cf. simulated annealing)

WL iteration

1. Start with $\omega(E) = 0 \; \forall E$.
2. Simulate "sufficiently long" while continuously updating $\omega(E)$.
3. Reduce modification factor, e.g.,

$$\phi \to \phi/2$$

4. Iterate till $\phi < \phi_{\mathrm{thres}}$.

Problem

Multicanonical simulations are very expensive. Samples in Markov chains suffer from autocorrelations of consecutive samples.

Sampling with neural networks

Early approaches: used systems such as (restricted) Boltzmann machines to generate states approximately according to Boltzmann distribution (e.g., `Torlai + Melko, 2016`).

Sampling with neural networks

Early approaches: used systems such as (restricted) Boltzmann machines to generate states approximately according to Boltzmann distribution (e.g., `Torlai + Melko, 2016`).

Problem: exact probability $P_{\mathrm{gen}}(\{s_i\})$ of generated states is not known.

Sampling with neural networks

Early approaches: used systems such as (restricted) Boltzmann machines to generate states approximately according to Boltzmann distribution (e.g., `Torlai + Melko, 2016`).

Problem: exact probability $P_{\rm gen}(\{s_i\})$ of generated states is not known.

If $P_{\rm gen}(\{s_i\})$ known, we can use **reweighting** to remove biases,

$$\langle O \rangle_\beta = \frac{\sum_t O(\{s_t\}) P_\beta(\{s_t\})/P_{\rm gen}(\{s_t\})}{\sum_t P_\beta(\{s_t\})/P_{\rm gen}(\{s_t\})}$$

Sampling with neural networks

Early approaches: used systems such as (restricted) Boltzmann machines to generate states approximately according to Boltzmann distribution (e.g., `Torlai + Melko, 2016`).

Problem: exact probability $P_{\mathrm{gen}}(\{s_i\})$ of generated states is not known.

If $P_{\mathrm{gen}}(\{s_i\})$ known, we can use **reweighting** to remove biases,

$$\langle O \rangle_\beta = \frac{\sum_t O(\{s_t\}) P_\beta(\{s_t\}) / P_{\mathrm{gen}}(\{s_t\})}{\sum_t P_\beta(\{s_t\}) / P_{\mathrm{gen}}(\{s_t\})}$$

Standard architectures achieving this: **autoregressive networks** and **normalizing flows**.

Autoregressive networks

Autoregressive networks (`Wu, Wang + Zhang, 2019`): based on factorization property

$$q(\{s_i\}) = \prod_{i=1}^{N} P(s_i|s_{<i}), \quad P(s_i|s_{<i}) = \begin{cases} q_i & s_i = +1, \\ 1 - q_i & s_i = -1. \end{cases}$$

Autoregressive networks

Autoregressive networks (`Wu, Wang + Zhang, 2019`): based on factorization property

$$q(\{s_i\}) = \prod_{i=1}^{N} P(s_i|s_{<i}), \quad P(s_i|s_{<i}) = \begin{cases} q_i & s_i = +1, \\ 1 - q_i & s_i = -1. \end{cases}$$

Mostly relevant for *discrete* variables (Ising and friends).

Autoregressive networks

Autoregressive networks (`Wu, Wang + Zhang, 2019`): based on factorization property

$$q(\{s_i\}) = \prod_{i=1}^{N} P(s_i|s_{<i}), \quad P(s_i|s_{<i}) = \begin{cases} q_i & s_i = +1, \\ 1 - q_i & s_i = -1. \end{cases}$$

Mostly relevant for *discrete* variables (Ising and friends).

Proposal probability $q(\{s_i\})$ explicitly known.

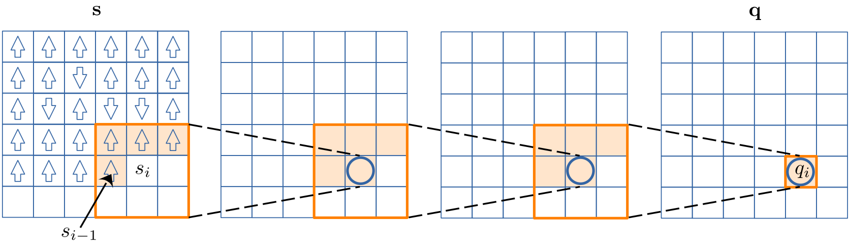
Autoregressive networks

Autoregressive networks (`Wu, Wang + Zhang, 2019`): based on factorization property

$$q(\{s_i\}) = \prod_{i=1}^{N} P(s_i|s_{<i}), \quad P(s_i|s_{<i}) = \begin{cases} q_i & s_i = +1, \\ 1 - q_i & s_i = -1. \end{cases}$$

Mostly relevant for *discrete* variables (Ising and friends).

Proposal probability $q(\{s_i\})$ explicitly known.



Normalizing flows

Normalizing flows represent an *invertible* map $T : \Omega \to \Omega$ that transports samples from the base or *latent space* to the target space (and vice versa).

Normalizing flows

Normalizing flows represent an *invertible* map $T : \Omega \to \Omega$ that transports samples from the base or *latent space* to the target space (and vice versa).

Probability densities transform in the usual way,

$$\rho(x) = \rho_L(T^{-1}(x)) \det |\nabla_x T^{-1}|,$$

where $\det |\nabla_x T^{-1}|$ is the Jacobi determinant. ρ_L is usually a multivariate Gaussian.

Normalizing flows

Normalizing flows represent an *invertible* map $T : \Omega \to \Omega$ that transports samples from the base or *latent space* to the target space (and vice versa).

Probability densities transform in the usual way,

$$\rho(x) = \rho_L(T^{-1}(x)) \det |\nabla_x T^{-1}|,$$

where $\det |\nabla_x T^{-1}|$ is the Jacobi determinant. ρ_L is usually a multivariate Gaussian.

First application to statistical mechanics: `Noé et al., 2019`.

Normalizing flows

Normalizing flows represent an *invertible* map $T : \Omega \to \Omega$ that transports samples from the base or *latent space* to the target space (and vice versa).

Probability densities transform in the usual way,

$$\rho(x) = \rho_L(T^{-1}(x)) \det |\nabla_x T^{-1}|,$$

where $\det |\nabla_x T^{-1}|$ is the Jacobi determinant. ρ_L is usually a multivariate Gaussian.

First application to statistical mechanics: `Noé et al., 2019`.

Training:

- with samples
- with (inverse) KL divergence w.r.t. target distribution (e.g., Boltzmann), "training by energy"

Normalizing flows

Normalizing flows represent an *invertible* map $T : \Omega \to \Omega$ that transports samples from the base or *latent space* to the target space (and vice versa).

Probability densities transform in the usual way,

$$\rho(x) = \rho_L(T^{-1}(x)) \det |\nabla_x T^{-1}|,$$

where $\det |\nabla_x T^{-1}|$ is the Jacobi determinant. ρ_L is usually a multivariate Gaussian.

First application to statistical mechanics: `Noé et al., 2019`.

Training:

- with samples
- with (inverse) KL divergence w.r.t. target distribution (e.g., Boltzmann), "training by energy"

does not work for discrete variables

Wang-Landau sampling with autoregressive network

We consider the 2D Potts model with Hamiltonian

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \quad s_i = 0, \ldots, q-1.$$

as test system: first-order transitions of increasing strength for $q > 4$.

Wang-Landau sampling with autoregressive network

We consider the 2D Potts model with Hamiltonian

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \quad s_i = 0, \ldots, q-1.$$

as test system: first-order transitions of increasing strength for $q > 4$.

Autoregressive network can be trained quite well to generate states according to canonical distribution.

Wang-Landau sampling with autoregressive network

We consider the 2D Potts model with Hamiltonian

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \quad s_i = 0, \ldots, q-1.$$

as test system: first-order transitions of increasing strength for $q > 4$.

Autoregressive network can be trained quite well to generate states according to canonical distribution.

How about the task of learning the density of states?

Wang-Landau sampling with autoregressive network

We consider the 2D Potts model with Hamiltonian

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \quad s_i = 0, \ldots, q-1.$$

as test system: first-order transitions of increasing strength for $q > 4$.

Autoregressive network can be trained quite well to generate states according to canonical distribution.

How about the task of learning the density of states? Possible algorithm:

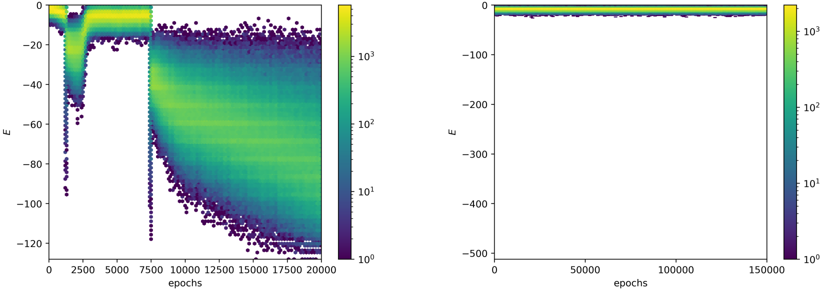
1. Initialize a preliminary estimate for the DoS $g(E) \equiv 1$ and a constant $f > 1$.
2. Generate a batch of M states $\{\mathbf{s}^{(k)}\}$ with probabilities $\{q(\mathbf{s}^{(k)})\}$.
3. Minimize $D_{\mathsf{KL}}(q\|p)$ for the current batch with $p(\mathbf{s}) \propto g(H(\mathbf{s}))^{-1}$.
4. Compute relative, not normalized weights $\hat{w}(\mathbf{s}^{(k)}) = g(H(\mathbf{s}^{(k)}))^{-1}/q(\mathbf{s}^{(k)})$ and update $g(H(\mathbf{s}^{(k)})) \leftarrow g(H(\mathbf{s}^{(k)})) \cdot f^{w(\mathbf{s}^{(k)})}$ with $w(\mathbf{s}^{(k)}) = \hat{w}(\mathbf{s}^{(k)}) / \sum_{k=1}^{M} \hat{w}(\mathbf{s}^{(k)})$.
5. Repeat from step 2 until histogram “broad” or “flat”.

Wang-Landau sampling with autoregressive network (2)

How well does it work?

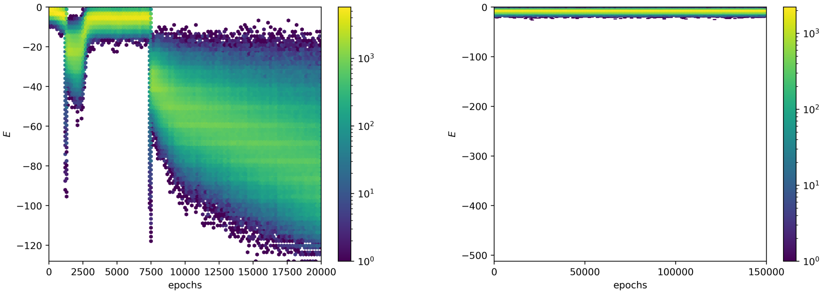
Wang-Landau sampling with autoregressive network (2)

How well does it work?



Wang-Landau sampling with autoregressive network (2)

How well does it work?



Observations:

- is not able to sample broad energy range
- shows signs of **catastrophic forgetting**
- shows very bad scaling behavior in system size

Autoencoder for the Potts model

We consider the 2D Potts model with Hamiltonian

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \quad s_i = 0, \ldots, q-1.$$

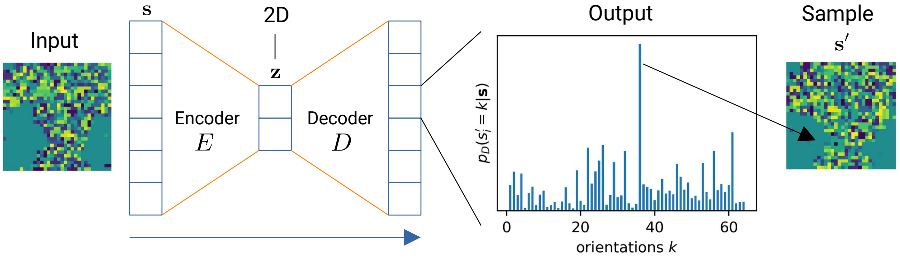
as test system: first-order transitions of increasing strength for $q > 4$.

Autoencoder for the Potts model

We consider the 2D Potts model with Hamiltonian

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{s_i,s_j}, \quad s_i = 0, \ldots, q-1.$$

as test system: first-order transitions of increasing strength for $q > 4$.



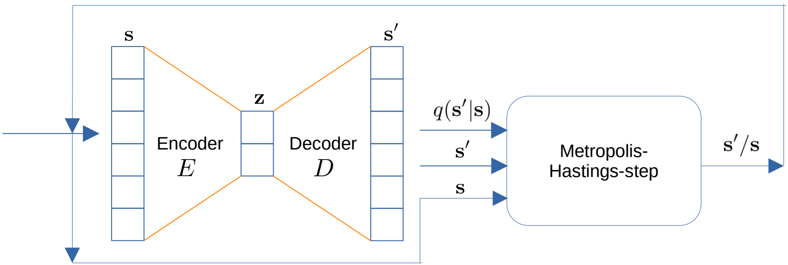
Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture



Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture

- encoder is deterministic, decoder contains VAN

Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture

- encoder is deterministic, decoder contains VAN
- train VAE to optimize *reconstruction accuracy* to improve locality,

$$L(\mathbf{s}) = q(\mathbf{s}|\mathbf{s}) = q_{D_\theta}(\mathbf{s}|E_\phi(\mathbf{s}))$$

Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture

- encoder is deterministic, decoder contains VAN
- train VAE to optimize *reconstruction accuracy* to improve locality,

$$L(\mathbf{s}) = q(\mathbf{s}|\mathbf{s}) = q_{D_\theta}(\mathbf{s}|E_\phi(\mathbf{s}))$$

- accept with Metropolis-Hastings probability

$$p_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') = \min\left[1, \frac{g(E)q(\mathbf{s}|\mathbf{s}')}{g(E')q(\mathbf{s}'|\mathbf{s})}\right].$$

Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture

- encoder is deterministic, decoder contains VAN
- train VAE to optimize *reconstruction accuracy* to improve locality,

$$L(\mathbf{s}) = q(\mathbf{s}|\mathbf{s}) = q_{D_\theta}(\mathbf{s}|E_\phi(\mathbf{s}))$$

- accept with Metropolis-Hastings probability

$$p_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') = \min\left[1, \frac{g(E)q(\mathbf{s}|\mathbf{s}')}{g(E')q(\mathbf{s}'|\mathbf{s})}\right].$$

- modify $g(E) \to g(E) \cdot f$ until histogram flat

Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture

- encoder is deterministic, decoder contains VAN
- train VAE to optimize *reconstruction accuracy* to improve locality,

$$L(\mathbf{s}) = q(\mathbf{s}|\mathbf{s}) = q_{D_\theta}(\mathbf{s}|E_\phi(\mathbf{s}))$$

- accept with Metropolis-Hastings probability

$$p_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') = \min\left[1, \frac{g(E)q(\mathbf{s}|\mathbf{s}')}{g(E')q(\mathbf{s}'|\mathbf{s})}\right].$$

- modify $g(E) \to g(E) \cdot f$ until histogram flat
- reduce modification factor $f \to \sqrt{f}$ (WL type update)

Autoregressive autoencoder

Non-locality of sampling with ANNs can also be a *disadvantage*, for example for multicanonical (density-of-states, Wand-Landau) simulations, where local exploration is important.

Consider hybrid architecture

- encoder is deterministic, decoder contains VAN
- train VAE to optimize *reconstruction accuracy* to improve locality,

$$L(\mathbf{s}) = q(\mathbf{s}|\mathbf{s}) = q_{D_\theta}(\mathbf{s}|E_\phi(\mathbf{s}))$$

- accept with Metropolis-Hastings probability

$$p_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') = \min\left[1, \frac{g(E)q(\mathbf{s}|\mathbf{s}')}{g(E')q(\mathbf{s}'|\mathbf{s})}\right].$$

- modify $g(E) \to g(E) \cdot f$ until histogram flat
- reduce modification factor $f \to \sqrt{f}$ (WL type update)
- possibly complement with supervised training

Neural Wang-Landau sampling

This leads to the following algorithm:

Neural Wang-Landau sampling

1. Generate a batch of M initial states $\{s_i\}$ and initialize $g(E) = 1\ \forall E$, $F > 1$.
2. Propose new states $\{s_i'\}$ through autoencoder and accept with

$$p_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') = \min\left[1, \frac{g(E)q(\mathbf{s}|\mathbf{s}')}{g(E')q(\mathbf{s}'|\mathbf{s})}\right].$$

3. Update $g(E) \leftarrow g(E) \cdot f$ for each state.
4. Train the network on the current batch.
5. Repeat from step 2 until full energy range has been explored.

Neural Wang-Landau sampling

This leads to the following algorithm:

Neural Wang-Landau sampling

1. Generate a batch of M initial states $\{s_i\}$ and initialize $g(E) = 1 \, \forall E$, $F > 1$.
2. Propose new states $\{s_i'\}$ through autoencoder and accept with

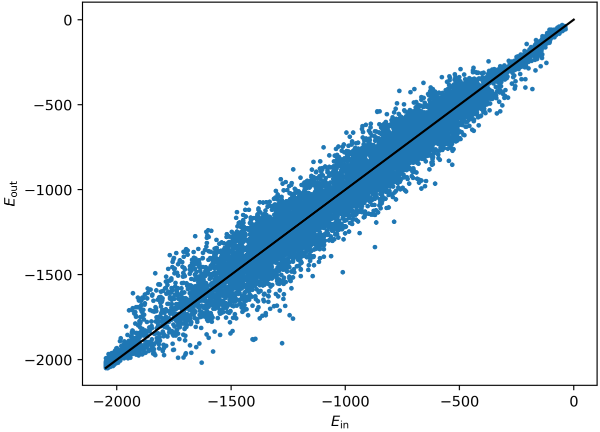
$$p_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') = \min\left[1, \frac{g(E)q(\mathbf{s}|\mathbf{s}')}{g(E')q(\mathbf{s}'|\mathbf{s})}\right].$$

3. Update $g(E) \leftarrow g(E) \cdot f$ for each state.
4. Train the network on the current batch.
5. Repeat from step 2 until full energy range has been explored.

Notably, at this stage it is *not* required to systematically reduce f.

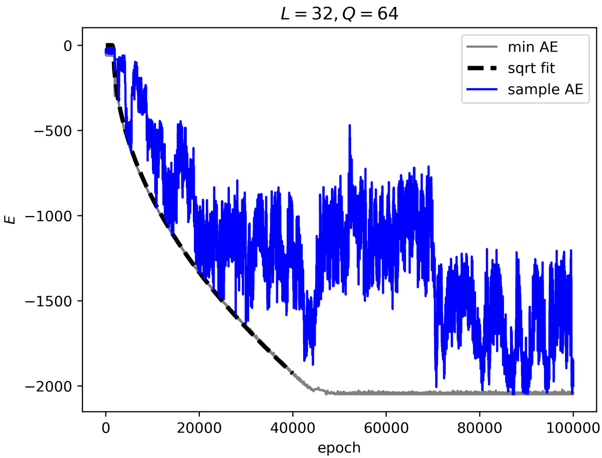
Network training

VAE leads to semi-local updates, ensuring reasonable acceptance rates.



Network training

VAE leads to semi-local updates, ensuring reasonable acceptance rates.



52 h GPU time

Density of states estimate

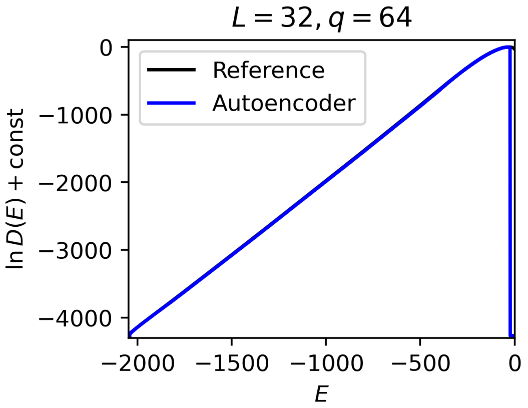
New estimator for density of states enabled by full knowledge of proposal probabilities,

$$\Omega(E) \propto \sum_{\mathbf{s}\in S_E} \frac{1}{q(\mathbf{s})}.$$

Density of states estimate

New estimator for density of states enabled by full knowledge of proposal probabilities,

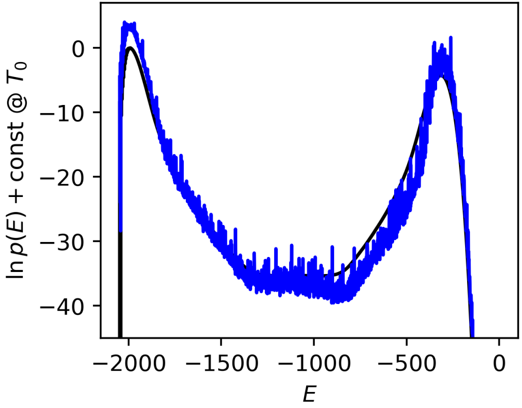
$$\Omega(E) \propto \sum_{\mathbf{s}\in S_E} \frac{1}{q(\mathbf{s})}.$$



Density of states estimate

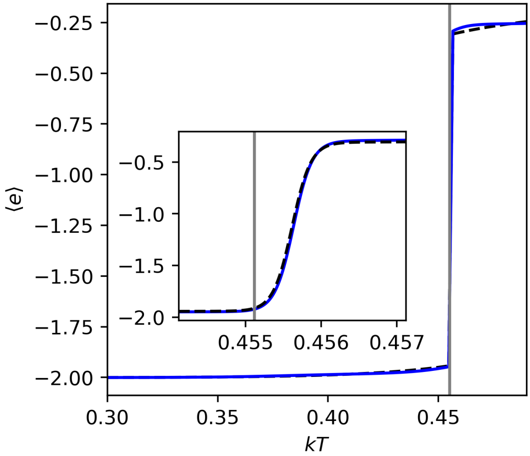
New estimator for density of states enabled by full knowledge of proposal probabilities,

$$\Omega(E) \propto \sum_{\mathbf{s}\in S_E} \frac{1}{q(\mathbf{s})}.$$



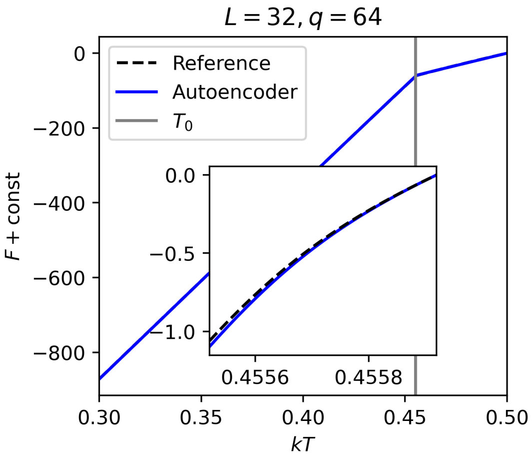
Canonical observables

We can easily estimate canonical averages, such as the mean energy and the free energy.



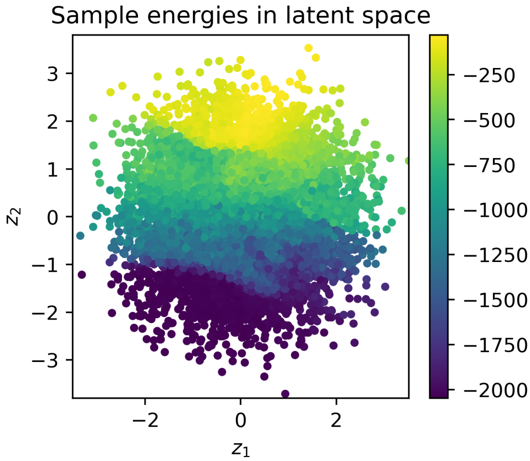
Canonical observables

We can easily estimate canonical averages, such as the mean energy and the free energy.



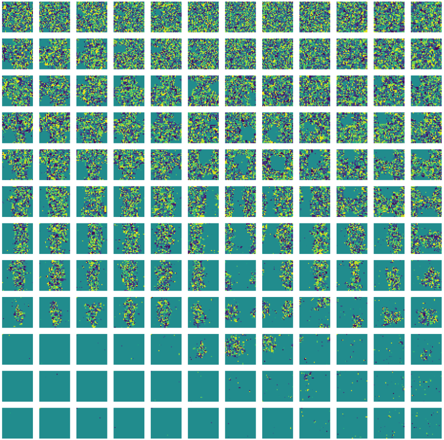
Conditioning in latent space

Use additional term in loss function to tie, e.g., energy to latent space variable.



Conditioning in latent space

Use additional term in loss function to tie, e.g., energy to latent space variable.



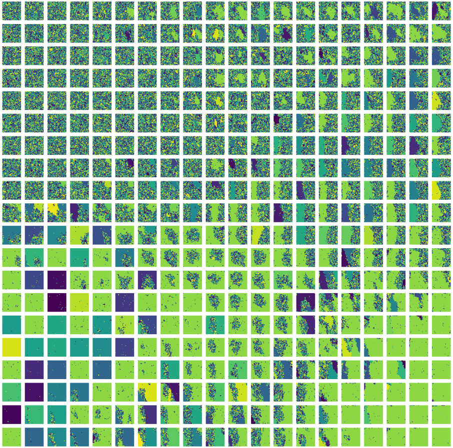
Mode collapse

A general problem with such ANN architectures is contraction to one or a few modes of the distribution: mode collapse

Mode collapse

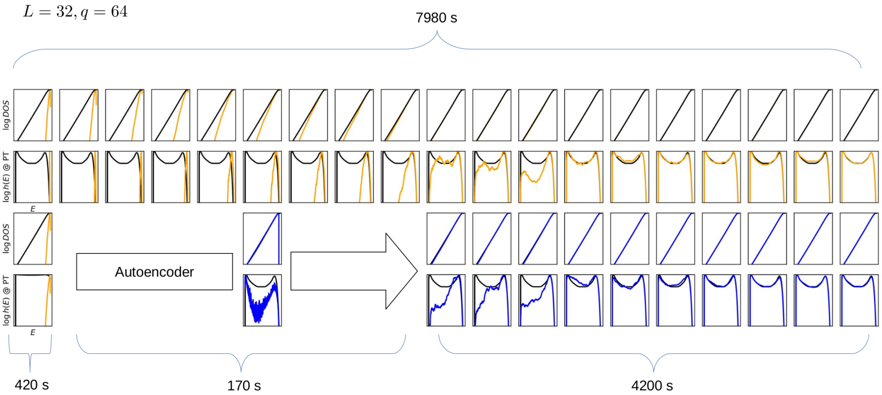
A general problem with such ANN architectures is contraction to one or a few modes of the distribution: mode collapse

Can be improved by explicitly symmetrizing the training samples.



Hybrid sampling

Combine advantages of traditional and neural WL sampling.



Conclusions

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach
- use semi-local technique with Markov chain and ANN as move proposer

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach
- use semi-local technique with Markov chain and ANN as move proposer
- combination of autoregressive network and autoencoder

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach
- use semi-local technique with Markov chain and ANN as move proposer
- combination of autoregressive network and autoencoder
- can imprint reaction coordinates onto latent-space variables

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach
- use semi-local technique with Markov chain and ANN as move proposer
- combination of autoregressive network and autoencoder
- can imprint reaction coordinates onto latent-space variables
- not (yet) computationally competitive as standalone approach (52 h on GPU vs. 3.5 h on CPU)

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach
- use semi-local technique with Markov chain and ANN as move proposer
- combination of autoregressive network and autoencoder
- can imprint reaction coordinates onto latent-space variables
- not (yet) computationally competitive as standalone approach (52 h on GPU vs. 3.5 h on CPU)
- hybrid approach combining strength of both approaches faster

Conclusions

- neural networks (ANNs) can be used for sampling stat. mech. systems *without systematic bias*
- broad range require for rare events difficult to achieve with purely global approach
- use semi-local technique with Markov chain and ANN as move proposer
- combination of autoregressive network and autoencoder
- can imprint reaction coordinates onto latent-space variables
- not (yet) computationally competitive as standalone approach (52 h on GPU vs. 3.5 h on CPU)
- hybrid approach combining strength of both approaches faster

Thank you for your attention!

Mode collapse (2)