Sampling states of spin models using generative neural networks

Moritz Riedel

Supervisor: Prof. Dr. Martin Weigel

Institut für Physik, TU Chemnitz

20.12.2023

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Motivation

Motivation

- Discrete statistical models important for studying phase ordering phenomena and phase transitions

Motivation

- Discrete statistical models important for studying phase ordering phenomena and phase transitions
- Conventional sampling algorithms (e.g. Metropolis) fail due to diverging autocorrelation times

Motivation

- Discrete statistical models important for studying phase ordering phenomena and phase transitions
- Conventional sampling algorithms (e.g. Metropolis) fail due to diverging autocorrelation times
- Networks with exactly known proposal probability allow unbiased computation of observables

Motivation

- Discrete statistical models important for studying phase ordering phenomena and phase transitions
- Conventional sampling algorithms (e.g. Metropolis) fail due to diverging autocorrelation times
- Networks with exactly known proposal probability allow unbiased computation of observables
- Supervised and Reinforcement learning compared
- Temperature dependent networks

Ising model

- Discrete model for ferromagnetism with spins pointing in two possible directions

Ising model

- Discrete model for ferromagnetism with spins pointing in two possible directions
- Exchange interaction favours parallel alignment
- Hamiltonian:

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j$$

with $s_i = \pm 1$

Ising model

- Discrete model for ferromagnetism with spins pointing in two possible directions
- Exchange interaction favours parallel alignment
- Hamiltonian:

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j$$

  with $s_i = \pm 1$
- Two dimensional case with $J = 1$ shows a continuous phase transition at $\beta_c = 1/2 \ln\left(1 + \sqrt{2}\right) \approx 0.44$

Generative neural networks

- Randomly generate samples $\mathbf{s}$ using neural networks
- Goal: Distribution of $\mathbf{s}$ should equal the Boltzmann-Distribution

$$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta \mathcal{H}(\mathbf{s})}$$

Generative neural networks

- Randomly generate samples $\mathbf{s}$ using neural networks
- Goal: Distribution of $\mathbf{s}$ should equal the Boltzmann-Distribution

$$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta \mathcal{H}(\mathbf{s})}$$

- Sampling probability of $q(\mathbf{s})$ is known for some types of networks $\rightarrow$ assign relative weight $w = p(\mathbf{s})/q(\mathbf{s}) \propto \exp(-\beta \mathcal{H}(\mathbf{s}))/q(\mathbf{s})$

Generative neural networks

- Randomly generate samples $\mathbf{s}$ using neural networks
- Goal: Distribution of $\mathbf{s}$ should equal the Boltzmann-Distribution

$$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta \mathcal{H}(\mathbf{s})}$$

- Sampling probability of $q(\mathbf{s})$ is known for some types of networks $\rightarrow$ assign relative weight $w = p(\mathbf{s})/q(\mathbf{s}) \propto \exp(-\beta \mathcal{H}(\mathbf{s}))/q(\mathbf{s})$
- Asymptotically unbiased expectation values estimated by

Reweighting A. M. Ferrenberg, R. H. Swendsen, 1988; K. A. Nicoli, N. Strodthoff et al., 2020

$$\langle \mathcal{A} \rangle \approx \frac{1}{M} \sum_{\mathbf{s} \sim q} \frac{p(\mathbf{s})}{q(\mathbf{s})} \mathcal{A}(\mathbf{s}) \approx \frac{\sum_{\mathbf{s} \sim q} \mathcal{A}(\mathbf{s}) e^{-\beta \mathcal{H}(\mathbf{s})}/q(\mathbf{s})}{\sum_{\mathbf{s} \sim q} e^{-\beta \mathcal{H}(\mathbf{s})}/q(\mathbf{s})}$$

Reweighting example

- Example: Reweight Metropolis-generated samples at $\beta_1 = 0.40$ to $\beta_2 = 0.44$

Reweighting example

- Example: Reweight Metropolis-generated samples at $\beta_1 = 0.40$ to $\beta_2 = 0.44$
- Apply relative weight

$$w(\mathbf{s}) = \frac{p(\mathbf{s})}{q(\mathbf{s})} \propto \frac{e^{-\beta_2 E(\mathbf{s})}}{e^{-\beta_1 E(\mathbf{s})}}$$

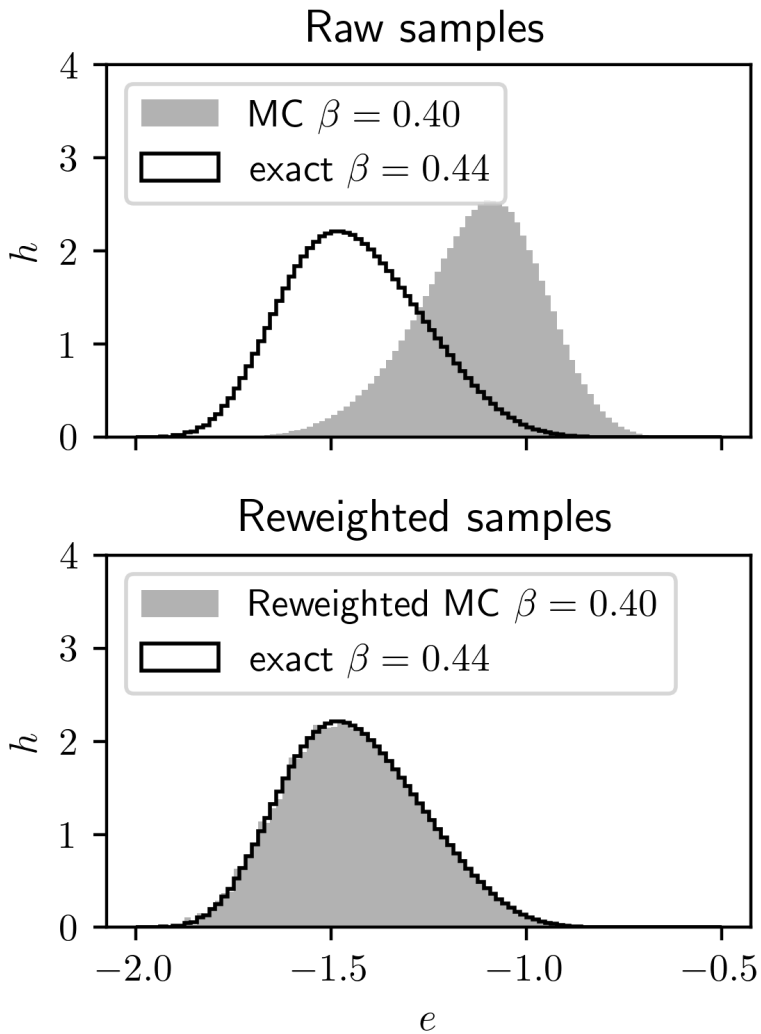
to each sample

Reweighting example

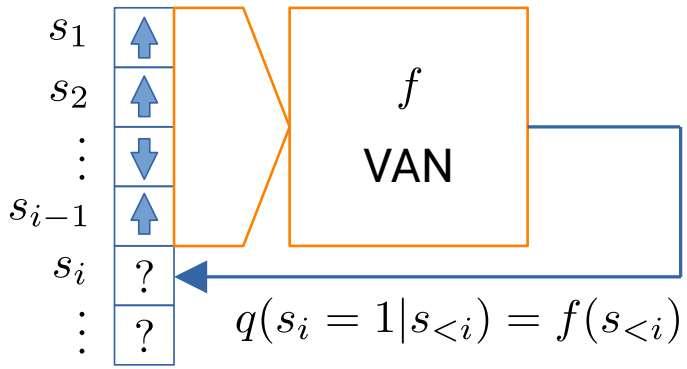
- Example: Reweight Metropolis-generated samples at $\beta_1 = 0.40$ to $\beta_2 = 0.44$
- Apply relative weight

$$w(\mathbf{s}) = \frac{p(\mathbf{s})}{q(\mathbf{s})} \propto \frac{e^{-\beta_2 E(\mathbf{s})}}{e^{-\beta_1 E(\mathbf{s})}}$$

to each sample

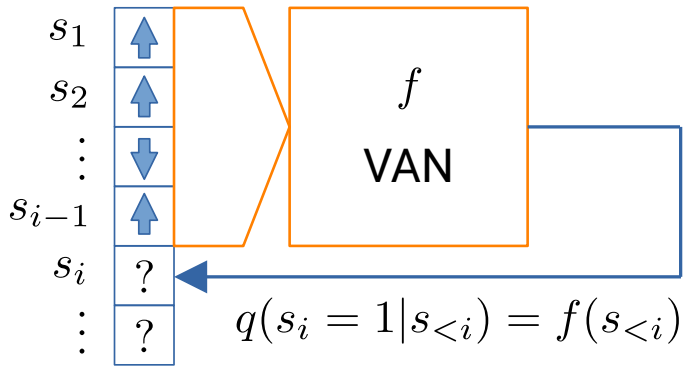


Variational autoregressive networks (VANs)



Generate a sample using VAN D. Wu, L. Wang, P. Zhang, 2018

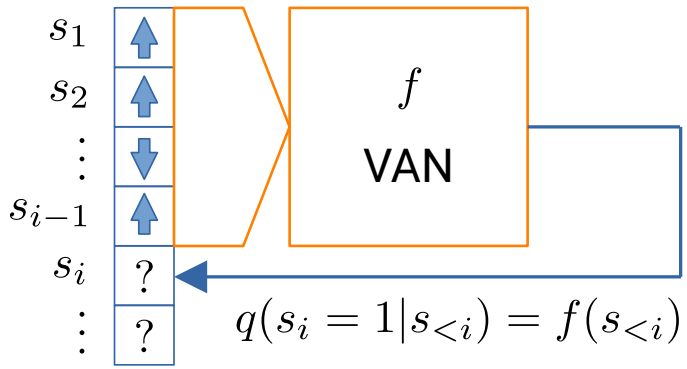
Variational autoregressive networks (VANs)



Generate a sample using VAN D. Wu, L. Wang, P. Zhang, 2018

1. Spin s_1 points upward with probability $q(s_1 = 1) = 1/2$

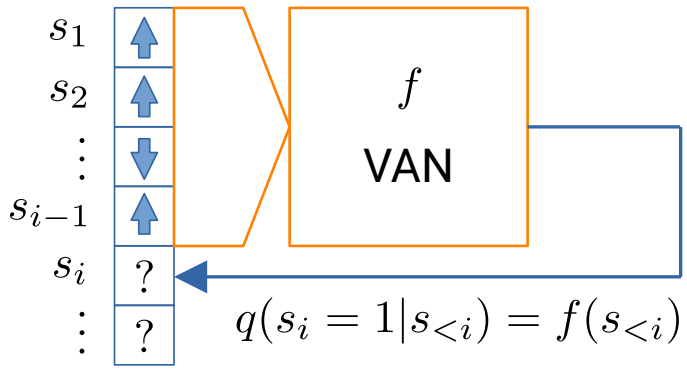
Variational autoregressive networks (VANs)



Generate a sample using VAN D. Wu, L. Wang, P. Zhang, 2018

1. Spin s_1 points upward with probability $q(s_1 = 1) = 1/2$
2. Pass already known spins $s_{<i}$ through the network and obtain $q_i = q(s_i = 1|s_{<i})$

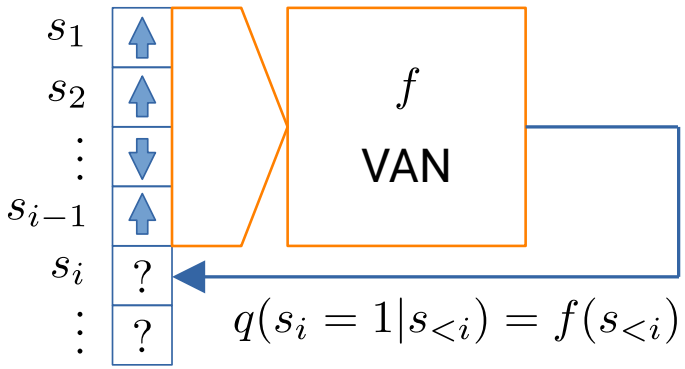
Variational autoregressive networks (VANs)



Generate a sample using VAN D. Wu, L. Wang, P. Zhang, 2018

1. Spin s_1 points upward with probability $q(s_1 = 1) = 1/2$
2. Pass already known spins $s_{<i}$ through the network and obtain $q_i = q(s_i = 1|s_{<i})$
3. Decide whether s_i points upward with that probability

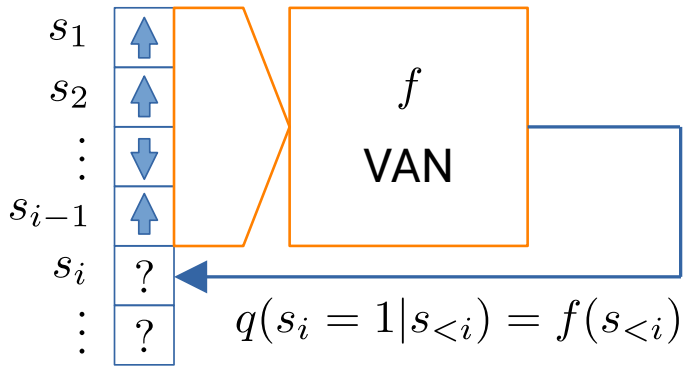
Variational autoregressive networks (VANs)



Generate a sample using VAN D. Wu, L. Wang, P. Zhang, 2018

1. Spin s_1 points upward with probability $q(s_1 = 1) = 1/2$
2. Pass already known spins $s_{<i}$ through the network and obtain $q_i = q(s_i = 1|s_{<i})$
3. Decide whether s_i points upward with that probability
4. Repeat for next spin from step 2 with $i \leftarrow i + 1$ until all spins are known

Variational autoregressive networks (VANs)



Generate a sample using VAN D. Wu, L. Wang, P. Zhang, 2018

1. Spin s_1 points upward with probability $q(s_1 = 1) = 1/2$
2. Pass already known spins $s_{<i}$ through the network and obtain $q_i = q(s_i = 1|s_{<i})$
3. Decide whether s_i points upward with that probability
4. Repeat for next spin from step 2 with $i \leftarrow i + 1$ until all spins are known
5. Return s and $q(\mathrm{s}) = \prod_{i=1}^{N} q(s_i|s_{<i})$

VAN architecture

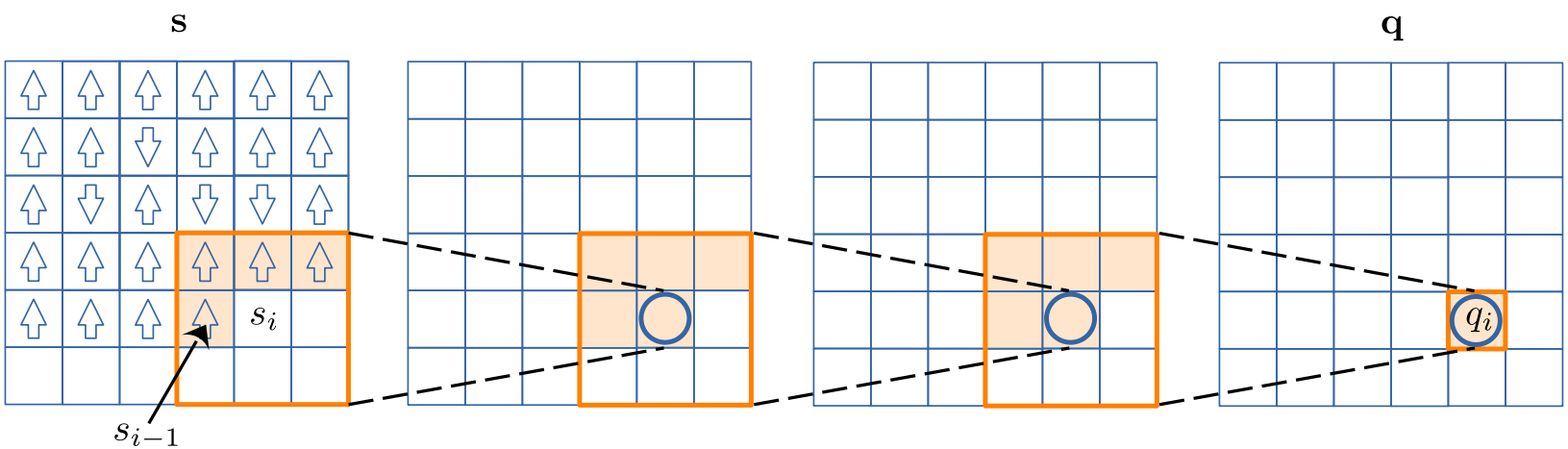
- Causal convolutional layers to fulfill the autoregressive property

  D. Wu, L. Wang, P. Zhang, 2018

VAN architecture

- Causal convolutional layers to fulfill the autoregressive property

  D. Wu, L. Wang, P. Zhang, 2018
- Kernel values are learnable parameters



Training VANs

Supervised learning

Training VANs

Supervised learning

- Boltzmann-distributed training samples required

Training VANs

Supervised learning

- Boltzmann-distributed training samples required
- Probability $q(\mathbf{s})$ for proposal of this sample is obtained by passing it through the network
- Maximize likelihood for a batch of samples:

$$\log \mathcal{L} \propto \log \prod_{\mathbf{s} \sim p} q(\mathbf{s})$$

Training VANs

Supervised learning

- Boltzmann-distributed training samples required
- Probability $q(\mathbf{s})$ for proposal of this sample is obtained by passing it through the network
- Maximize likelihood for a batch of samples:

$$\log \mathcal{L} \propto \log \prod_{\mathbf{s} \sim p} q(\mathbf{s})$$

Reinforcement learning

Training VANs

Supervised learning

- Boltzmann-distributed training samples required
- Probability $q(\mathbf{s})$ for proposal of this sample is obtained by passing it through the network
- Maximize likelihood for a batch of samples:

$$\log \mathcal{L} \propto \log \prod_{\mathbf{s} \sim p} q(\mathbf{s})$$

Reinforcement learning

- The network generates a batch of samples

Training VANs

Supervised learning

- Boltzmann-distributed training samples required
- Probability $q(\mathbf{s})$ for proposal of this sample is obtained by passing it through the network
- Maximize likelihood for a batch of samples:

$$\log \mathcal{L} \propto \log \prod_{\mathbf{s} \sim p} q(\mathbf{s})$$

Reinforcement learning

- The network generates a batch of samples
- Kullback-Leibler divergence is a measure of deviations between distributions
- Minimize $D_{\mathrm{KL}}(q \| p)$:

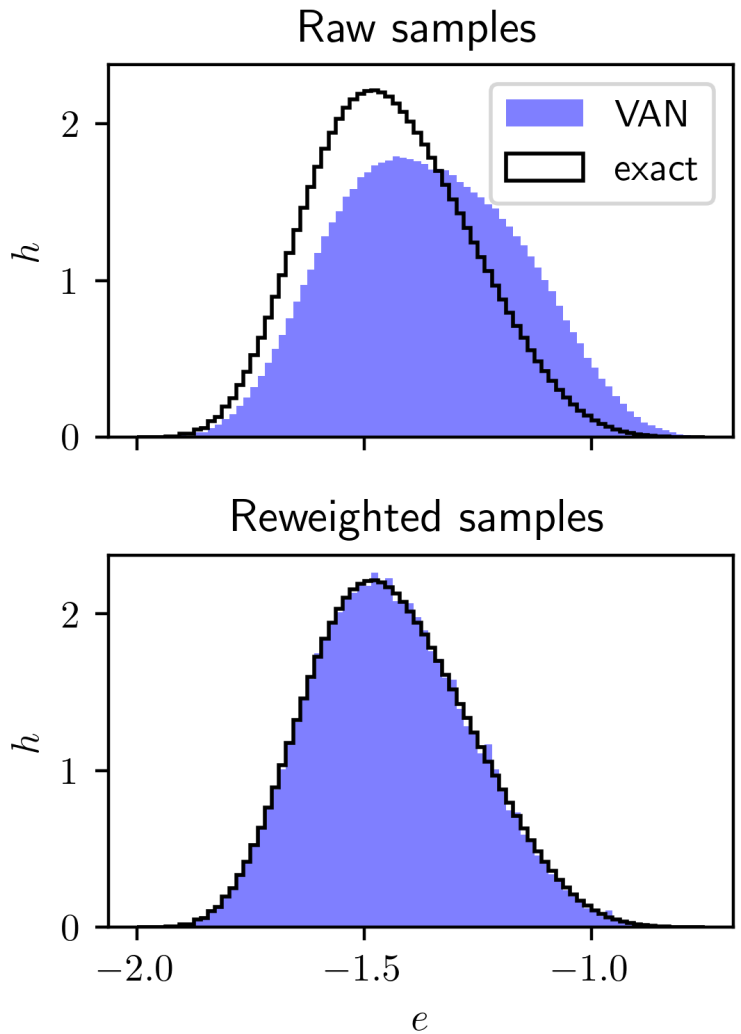
$$D_{\mathrm{KL}}(q \| p) \propto \sum_{\mathbf{s} \sim q} \log \left(\frac{q(\mathbf{s})}{p(\mathbf{s})} \right)$$

Energy histogram of sampled states

- VAN supervised trained for $L = 16$ and $\beta = 0.44$

Energy histogram of sampled states

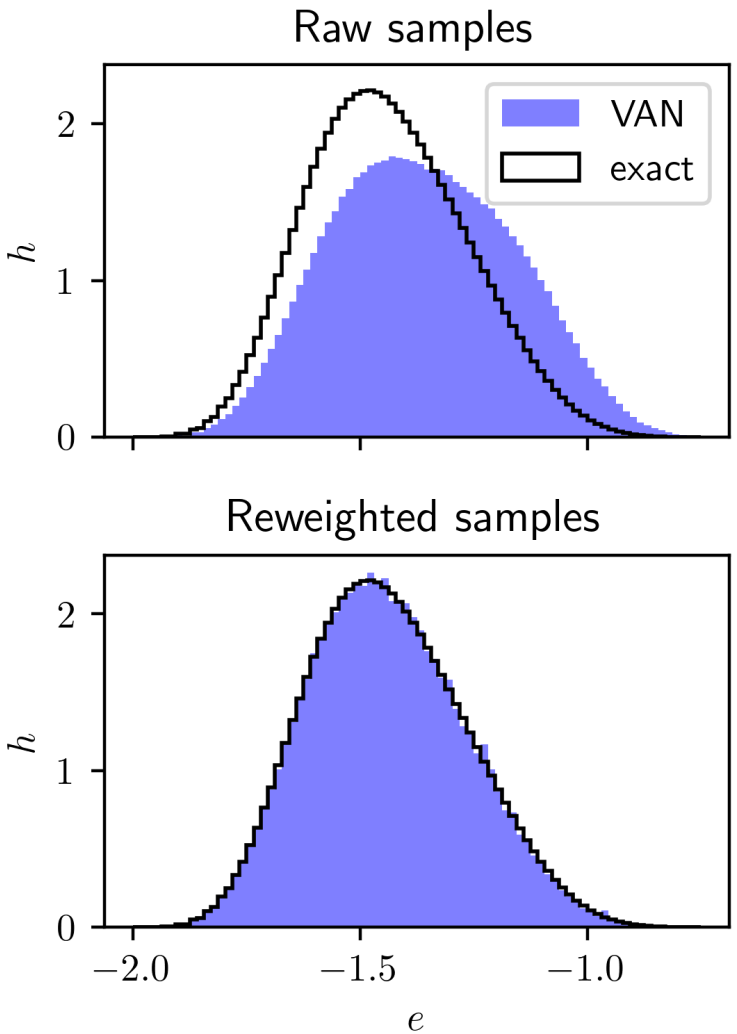
- VAN supervised trained for $L = 16$ and $\beta = 0.44$



Energy histogram of sampled states

- VAN supervised trained for $L = 16$ and $\beta = 0.44$
- Estimates for $\langle e \rangle$ obtained from 10^6 samples:

Exact	$-1.4477...$
Raw samples	-1.3679
Reweighted	-1.4465 ± 0.0027
Relative error	8.5×10^{-4}

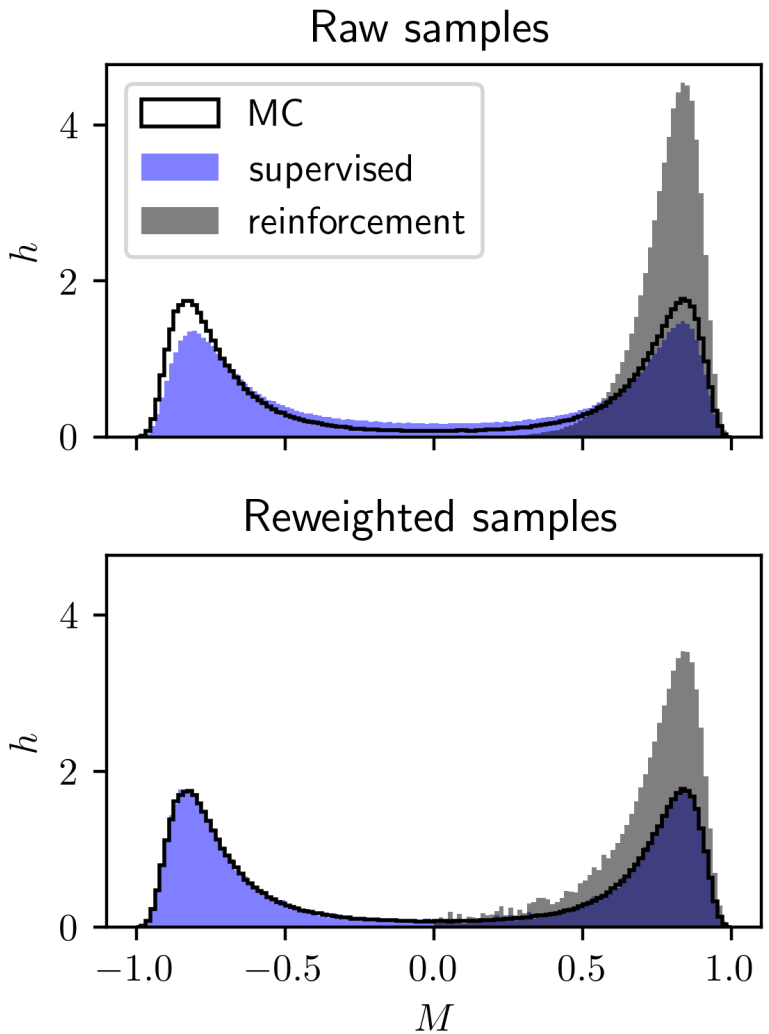


Supervised and Reinforcement Learning

- Both VANs trained for $L = 16$ and $\beta = 0.44$ with equal hyperparameters

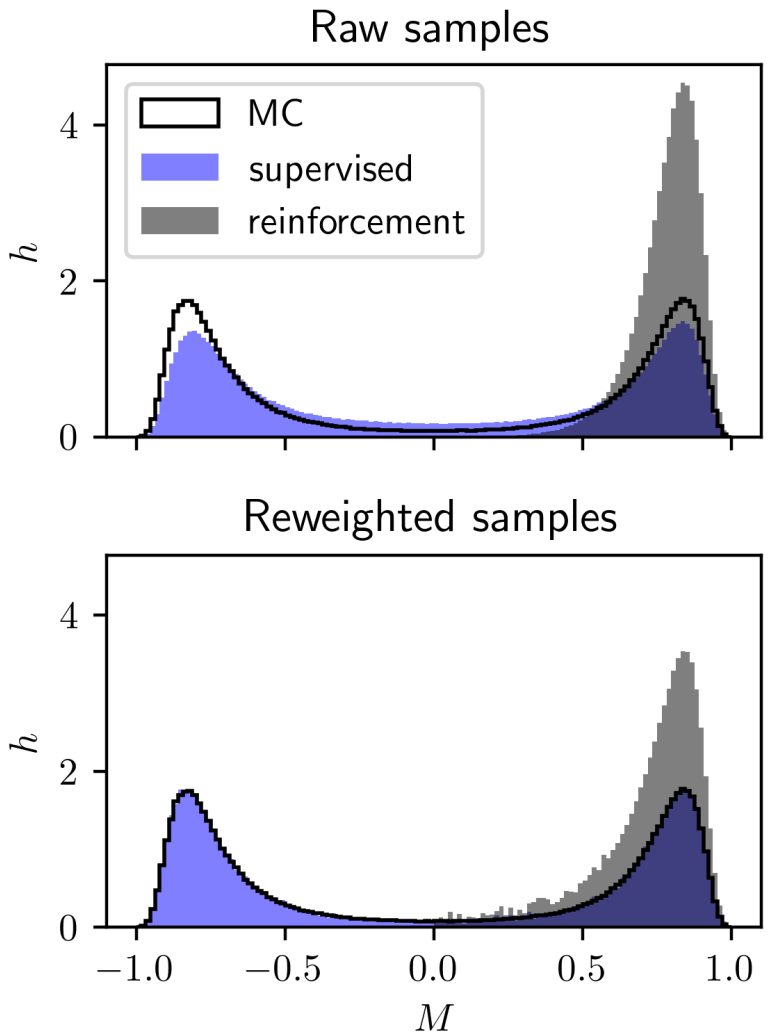
Supervised and Reinforcement Learning

- Both VANs trained for $L = 16$ and $\beta = 0.44$ with equal hyperparameters
- Reinforced trained VAN does not require training data, but not capturing entire phase space



Supervised and Reinforcement Learning

- Both VANs trained for $L = 16$ and $\beta = 0.44$ with equal hyperparameters
- Reinforced trained VAN does not require training data, but not capturing entire phase space
- Might be problematic for more complex systems

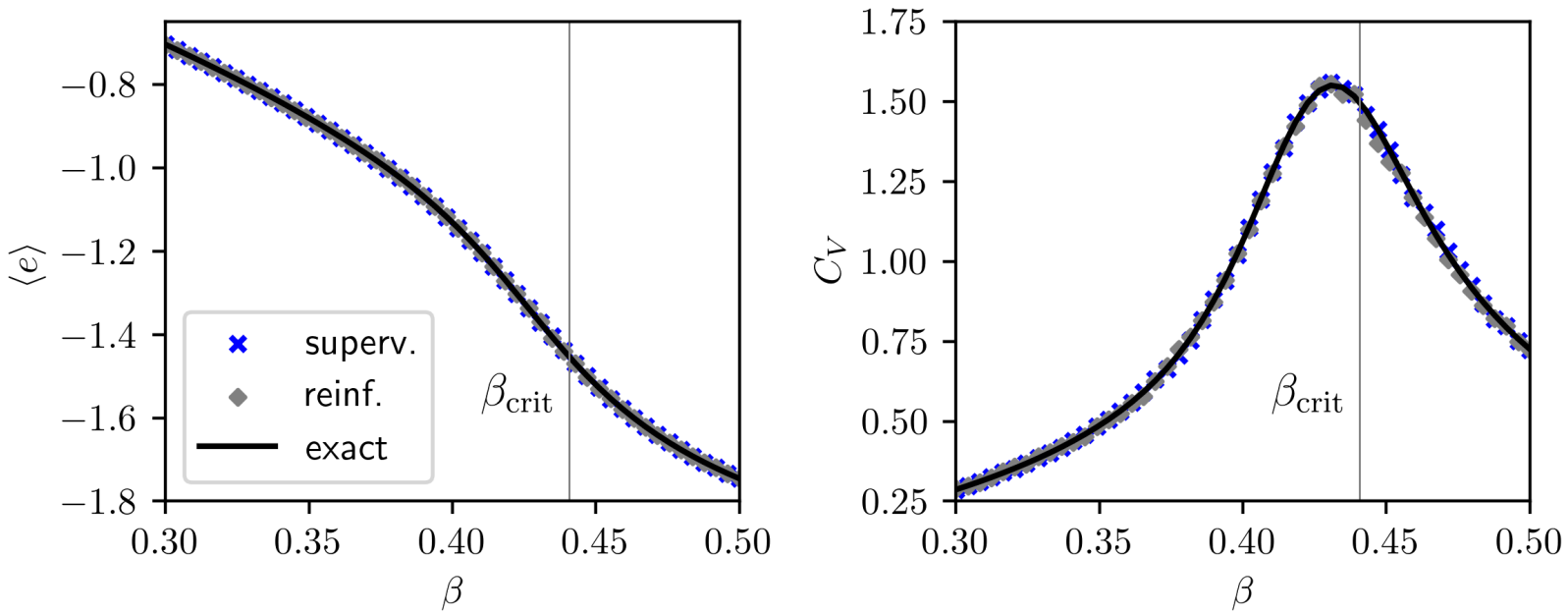


Observables from temperature dependent VANs

- Additional input parameter for inverse temperature β
- Randomly chosen $\beta \in (0, 1]$ for each training epoch

Observables from temperature dependent VANs

- Additional input parameter for inverse temperature β
- Randomly chosen $\beta \in (0, 1]$ for each training epoch
- Precise results for $\langle e \rangle$ and C_V in temperature range



Conclusion

- VANs allow for computation of sampling probabilities

Conclusion

- VANs allow for computation of sampling probabilities
- Yield precise thermodynamic properties by reweighting (relative error of $\langle e \rangle$ was $< 10^{-3}$ for $L = 16$)

Conclusion

- VANs allow for computation of sampling probabilities
- Yield precise thermodynamic properties by reweighting (relative error of $\langle e \rangle$ was $< 10^{-3}$ for $L = 16$)
- Reinforced trained VANs do not cover entire phase space

Conclusion

- VANs allow for computation of sampling probabilities
- Yield precise thermodynamic properties by reweighting (relative error of $\langle e \rangle$ was $< 10^{-3}$ for $L = 16$)
- Reinforced trained VANs do not cover entire phase space
- Experiments with temperature dependent networks

Conclusion

- VANs allow for computation of sampling probabilities
- Yield precise thermodynamic properties by reweighting (relative error of $\langle e \rangle$ was $< 10^{-3}$ for $L = 16$)
- Reinforced trained VANs do not cover entire phase space
- Experiments with temperature dependent networks

Outlook

- Population-based reinforcement training to prevent mode collapse

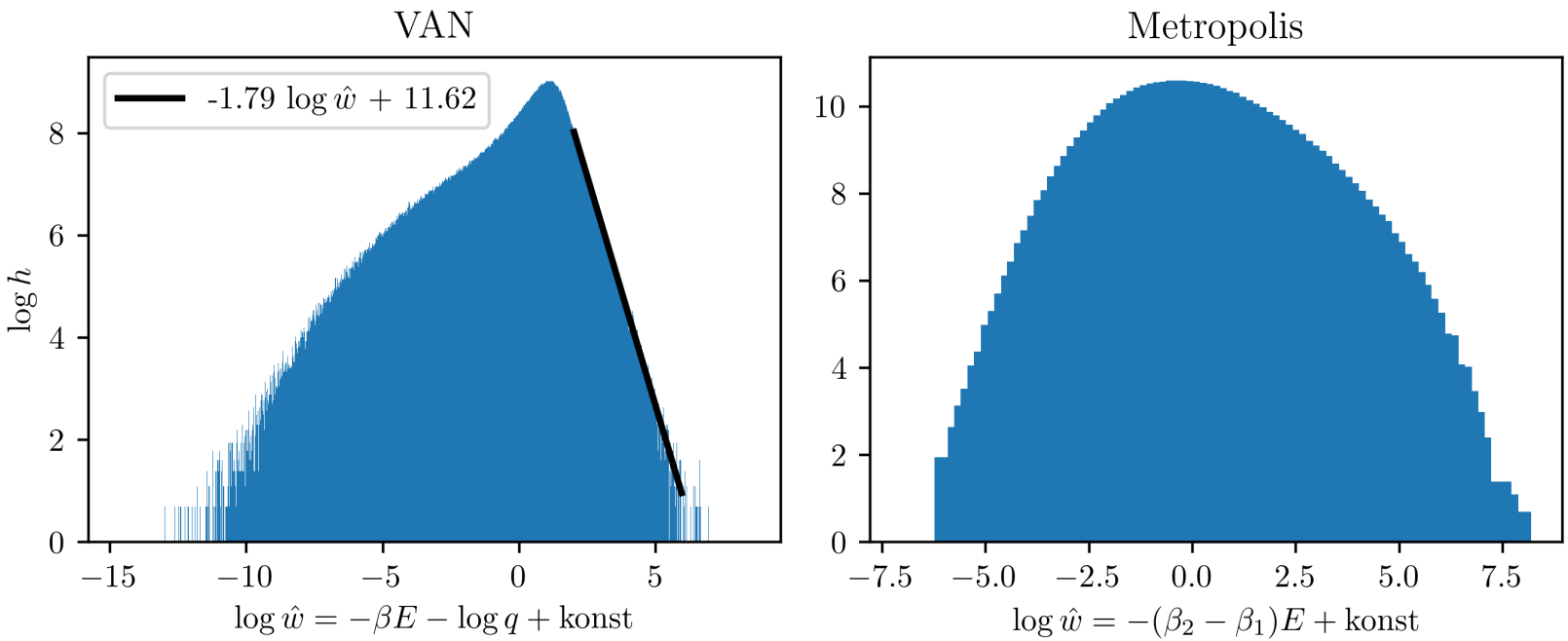
Conclusion

- VANs allow for computation of sampling probabilities
- Yield precise thermodynamic properties by reweighting (relative error of $\langle e \rangle$ was $< 10^{-3}$ for $L = 16$)
- Reinforced trained VANs do not cover entire phase space
- Experiments with temperature dependent networks

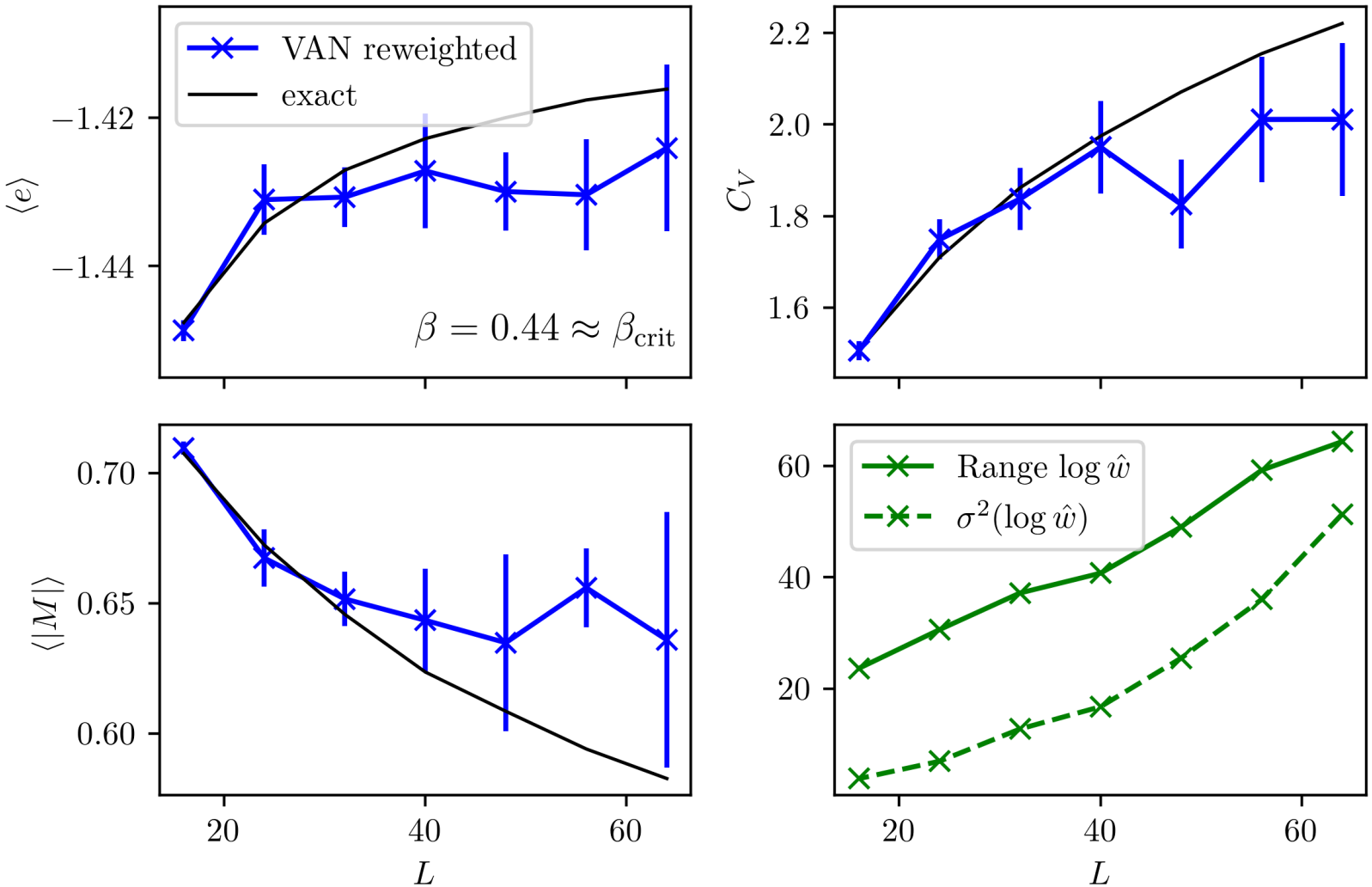
Outlook

- Population-based reinforcement training to prevent mode collapse
- Multicanonical sampling

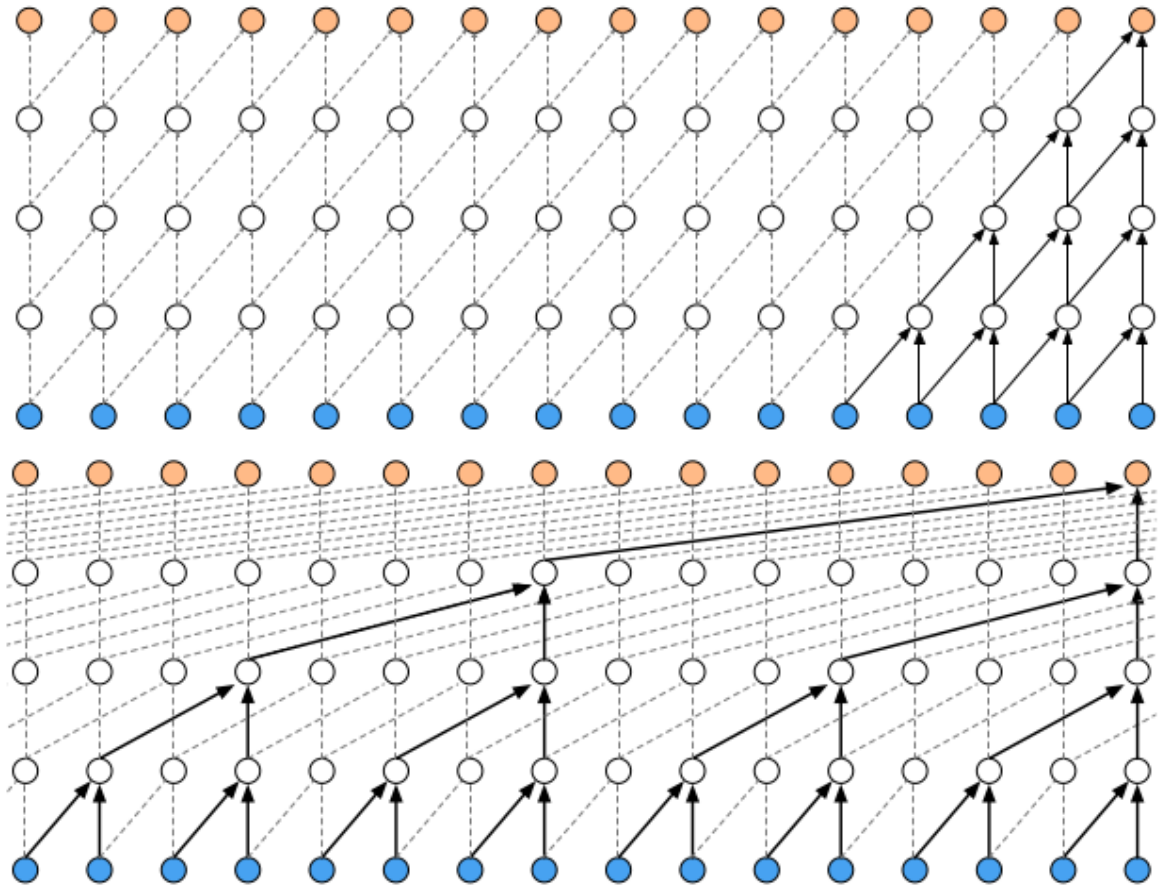
Discussion


VAN
-1.79 log ŵ + 11.62
log h
log ŵ = −βE − log q + konst
Metropolis
log ŵ = −(β2 − β1)E + konst

Discussion


VAN reweighted
exact
$\beta = 0.44 \approx \beta_{\text{crit}}$
$\langle e \rangle$
−1.42
−1.44
C_V
2.2
2.0
1.8
1.6
$\langle |M| \rangle$
0.70
0.65
0.60
Range $\log \hat{w}$
$\sigma^2(\log \hat{w})$
60
40
20
20
40
60
L
L

Discussion



A. van den Oord, S. Dieleman, H.Zen et. al, 2016