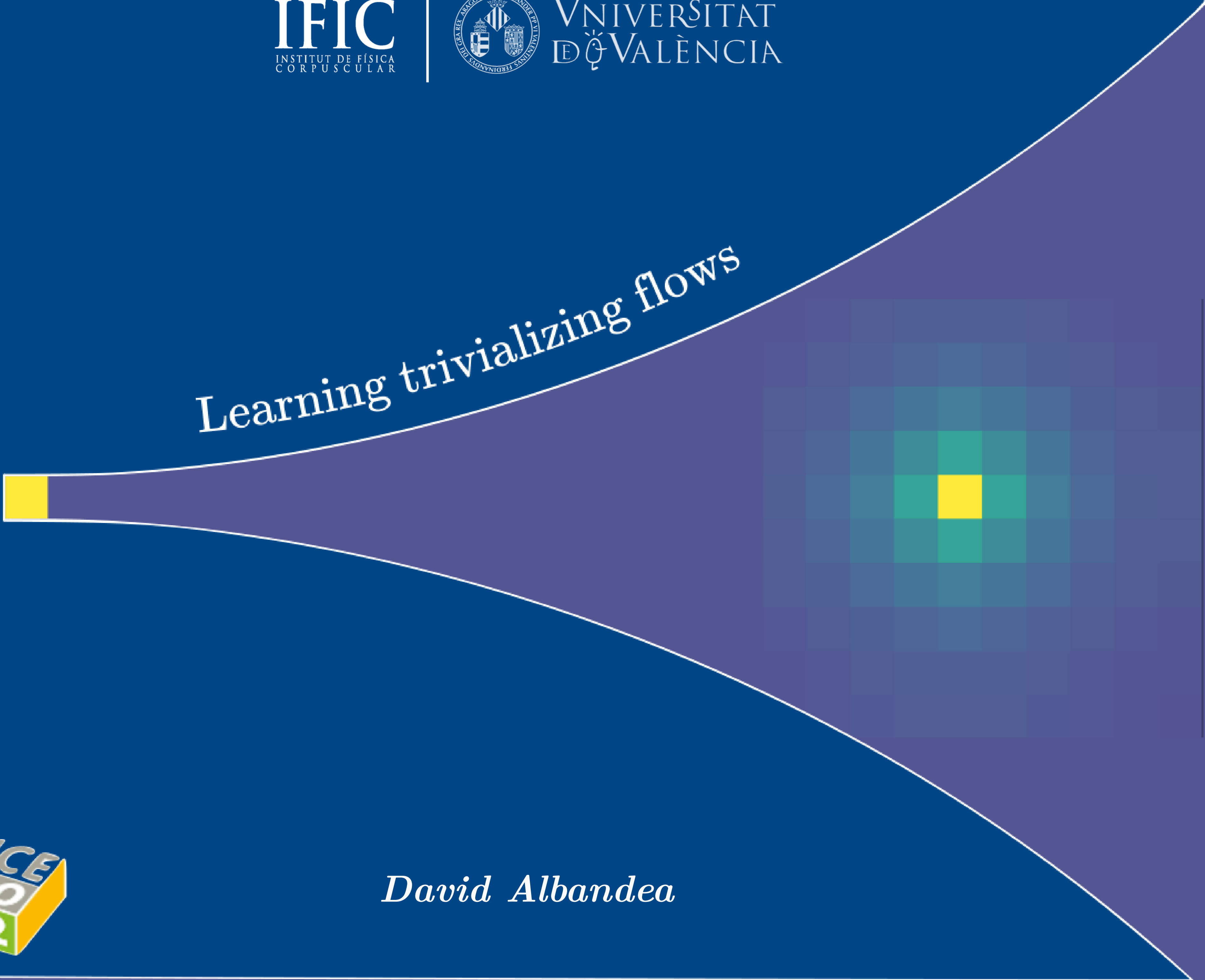
IFIC INSTITUT DE FÍSICA CORPUSCULAR | VNIVERSITAT DE VALÈNCIA

Learning trivializing flows



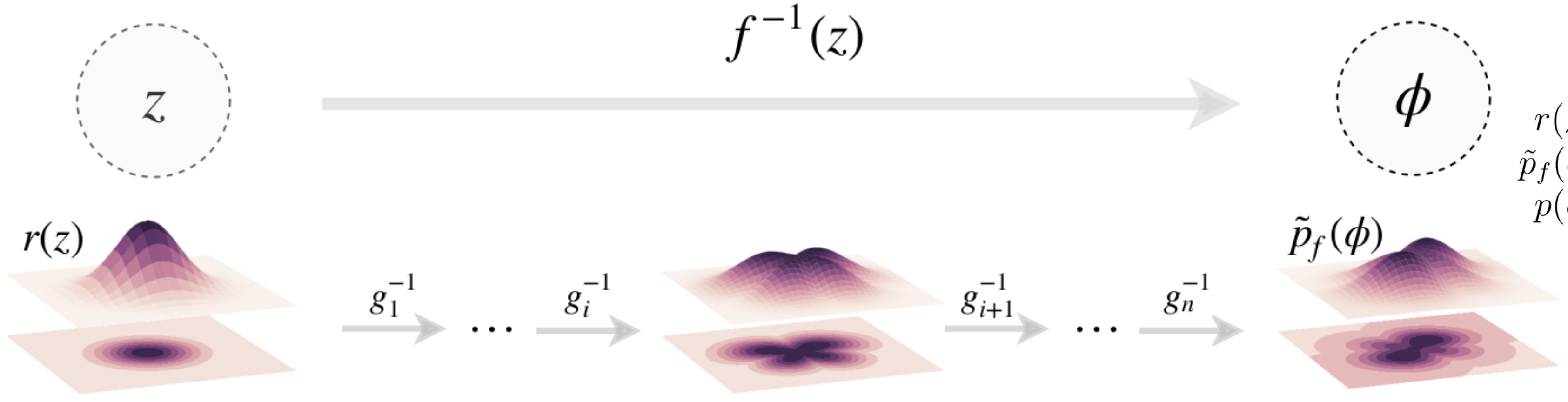
David Albandea

LATTICE 2022

Luigi del Debbio
Richard Kenway
Joe Marsh Rossney

David Albandea
Pilar Hernández
Alberto Ramos

Normalizing flows



(a) Normalizing flow between prior and output distributions

M. S. Albergo, G. Kanwar and P. E. Shanahan, Phys. Rev. D 100, 034515 (2019), 1904.12072

$r(z)$: input distribution
$\tilde{p}_f(\phi)$: output distribution
$p(\phi)$: target distribution

⇨ *f(z)* is a network trained to minimize the Kullbach-Leibler divergence:

$$\boxed{D_{\mathrm{KL}}(\tilde{p}_f \,||\, p) = \int \mathcal{D}\phi\, \tilde{p}_f(\phi) \log \frac{\tilde{p}_f(\phi)}{p(\phi)}}$$

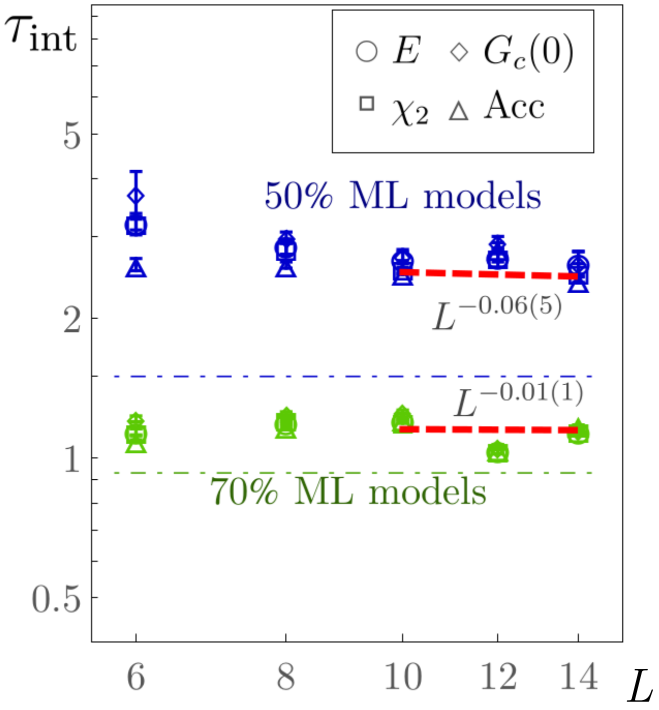
☆ $D_{\mathrm{KL}}(\tilde{p}_f \,||\, p) \geq 0$

☆ $D_{\mathrm{KL}}(\tilde{p}_f \,||\, p) = 0 \iff \tilde{p}_f = p$ ~ Trivializing map

⇨ Once *f* is trained, build a Markov chain with Metropoils-Hastings reweighting

Exploding training costs

M. S. Albergo, G. Kanwar and P. E. Shanahan,
Phys. Rev. D 100, 034515 (2019), 1904.12072



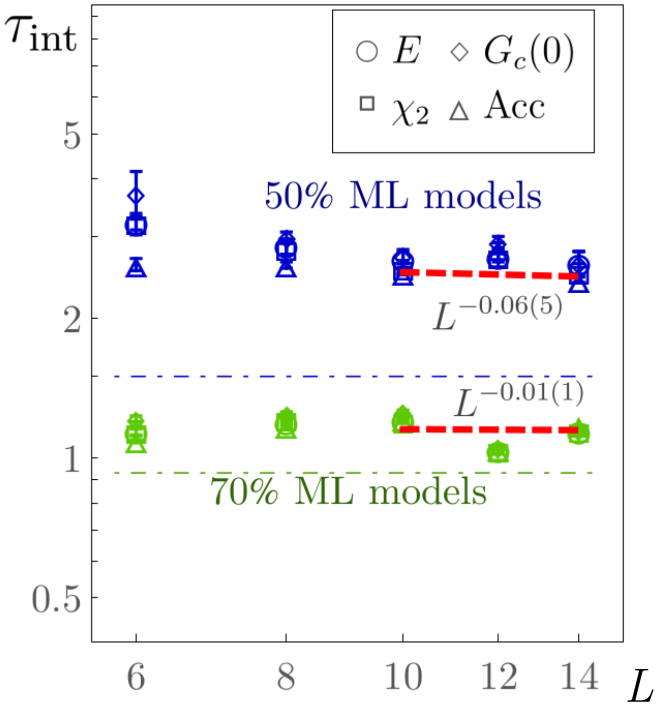
For equal acceptance, autocorrelation times do not scale towards the continuum

vs HMC: $\sim a^2$

Exploding training costs

Total cost $=$ configuration production cost $+$ network training cost

M. S. Albergo, G. Kanwar and P. E. Shanahan, Phys. Rev. D 100, 034515 (2019), 1904.12072



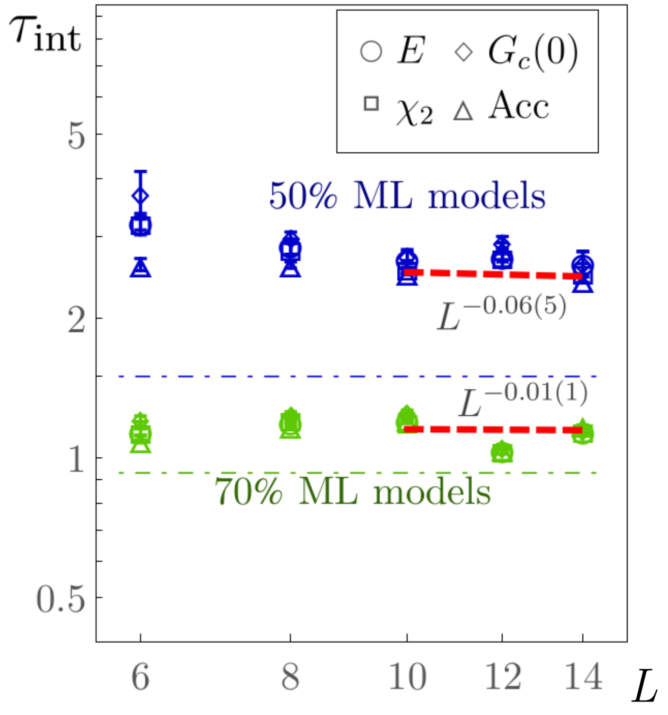
For equal acceptance, autocorrelation times do not scale towards the continuum

vs HMC: $\sim a^2$

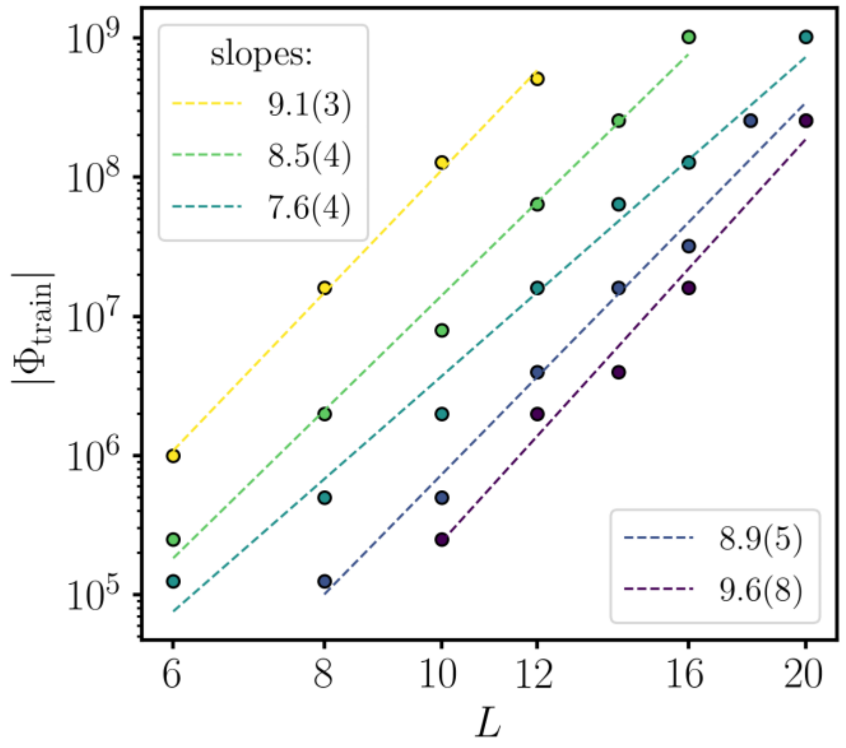
Exploding training costs

Total cost $=$ configuration production cost $+$ network training cost

M. S. Albergo, G. Kanwar and P. E. Shanahan, Phys. Rev. D 100, 034515 (2019), 1904.12072



Luigi Del Debbio, Joe Marsh Rossney, and Michael Wilson Phys. Rev. D 104, 094507



For equal acceptance, autocorrelation times do not scale towards the continuum

vs HMC: $\sim a^2$

Training costs to achieve equal acceptance explode towards the continuum as $\sim a^8$

Can we benefit from normalizing flows keeping training costs low?

Learning trivializing flows

☆ Idea: use the normalizing flow f to **help** HMC sampling

$$Z = \int D\phi \, e^{-S(\phi)} \xrightarrow{\tilde{\phi} = f(\phi)} \int D\tilde{\phi} \, e^{-S(f^{-1}(\tilde{\phi})) + \log \det J[f]} \equiv \int D\tilde{\phi} \, e^{-\tilde{S}(\tilde{\phi})}$$

⇨ $\tilde{S}$ might be easier to sample from using HMC

↳ lower autocorrelation times!

Learning trivializing flows

☆ Idea: use the normalizing flow f to **help** HMC sampling

$$Z = \int D\phi\, e^{-S(\phi)} \xrightarrow{\tilde{\phi}=f(\phi)} \int D\tilde{\phi}\, e^{-S(f^{-1}(\tilde{\phi}))+\log\det J[f]} \equiv \int D\tilde{\phi}\, e^{-\tilde{S}(\tilde{\phi})}$$

⇨ $\tilde{S}$ might be easier to sample from using HMC

↳ lower autocorrelation times!

The algorithm

1. Train the network f minimizing the KL divergence.
2. Use HMC to build a Markov chain following $\tilde{p} = e^{-\tilde{S}(\tilde{\phi})}$

$$\{\tilde{\phi}_1,\ \tilde{\phi}_2,\ \tilde{\phi}_3,\ \ldots,\ \tilde{\phi}_N\} \sim e^{-\tilde{S}(\tilde{\phi})}$$

3. Apply f^{-1} to the Markov chain to obtain configurations following $p(\phi) = e^{-S(\phi)}$

$$\{f^{-1}(\tilde{\phi}_1),\ f^{-1}(\tilde{\phi}_2),\ f^{-1}(\tilde{\phi}_3),\ \ldots,\ f^{-1}(\tilde{\phi}_N)\} = \{\phi_1,\ \phi_2,\ \phi_3,\ \ldots,\ \phi_N\} \sim e^{-S(\phi)}$$

⇨ The acceptance of HMC with the new action $\tilde{S}$ **does not depend on** f!

Learning trivializing flows

☆ Lüscher: an exact trivializing flow is not known, but can be constructed via power series (Wilson flow) Lüscher, M. Trivializing Maps, the Wilson Flow and the HMC Algorithm. Commun. Math. Phys. 293, 899 (2010)

It was not good enough to improve autocorrelation scaling towards the continuum on a CP(N) theory G. P. Engel, S. Schaefer, Testing trivializing maps in the Hybrid Monte Carlo algorithm, Comput.Phys.Commun. 182 (2011) 2107-2114

Can normalizing flows be helpful as trivializing flows for HMC?

Xiao-Yong Jin, Neural Network Field Transformation and Its Application in HMC, PoS LATTICE2021 (2022) 600

S. Foreman *et al.*, HMC with Normalizing Flows, PoS LATTICE2021 (2022) 073

The algorithm

1. Train the network f minimizing the KL divergence.
2. Use HMC to build a Markov chain following $\tilde{p} = e^{-\tilde{S}(\tilde{\phi})}$

$$\{\tilde{\phi}_1,\ \tilde{\phi}_2,\ \tilde{\phi}_3,\ \dots,\ \tilde{\phi}_N\} \sim e^{-\tilde{S}(\tilde{\phi})}$$

3. Apply f^{-1} to the Markov chain to obtain configurations following $p(\phi) = e^{-S(\phi)}$

$$\{f^{-1}(\tilde{\phi}_1),\ f^{-1}(\tilde{\phi}_2),\ f^{-1}(\tilde{\phi}_3),\ \dots,\ f^{-1}(\tilde{\phi}_N)\} = \{\phi_1,\ \phi_2,\ \phi_3,\ \dots,\ \phi_N\} \sim e^{-S(\phi)}$$

The acceptance of HMC with the new action $\tilde{S}$ **does not depend on** f !

The model

⇨ We study a ϕ^4 theory in 2 dimensions

$$S(\phi) = \sum_x \left[-\beta \sum_{\mu=1}^{2} \phi_{x+\mu}\phi_x + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 \right]$$

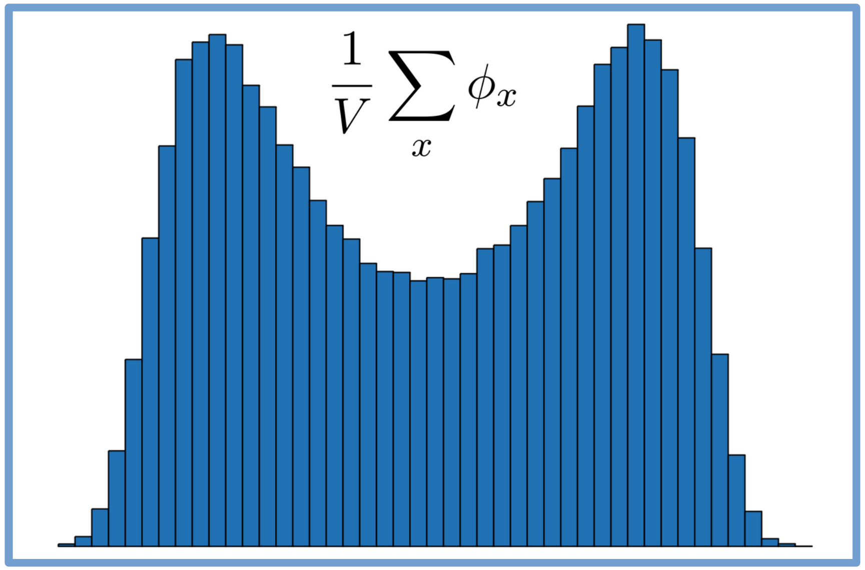
☆ $\mathbb{Z}_2$ symmetry: action invariant under $\phi \to -\phi$

☆ Bimodal probability density

☆ Non-trivial correlation length ξ

↳ HMC scaling: $\tau_{\text{int}} \propto \xi^2$

☆ No topology freezing

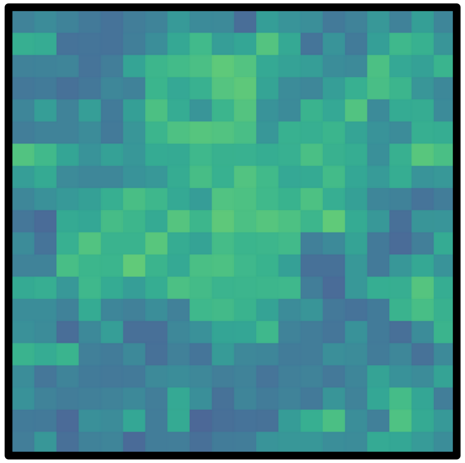


Keeping training costs low

$$\boxed{\text{Total cost} \approx \text{configuration production cost}}$$

☆ Translational symmetry $\Rightarrow$ use convolutional networks

configuration



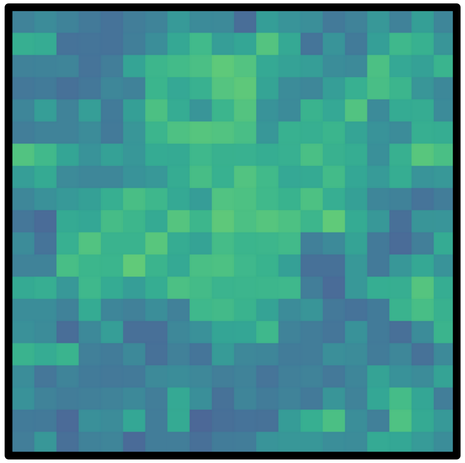
Keeping training costs low

Total cost $\approx$ configuration production cost

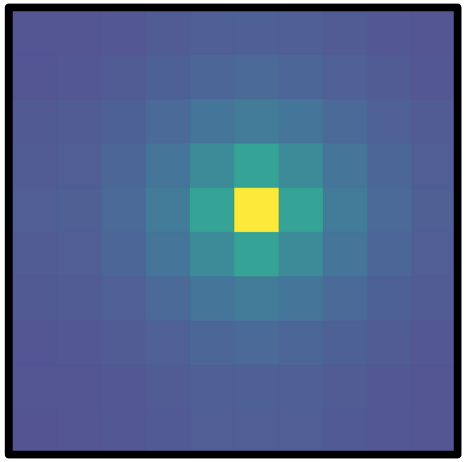
☆ Translational symmetry $\Rightarrow$ use convolutional networks

☆ Information within correlation length $\Rightarrow$ control network footprint

configuration



2-point correlation



Keeping training costs low

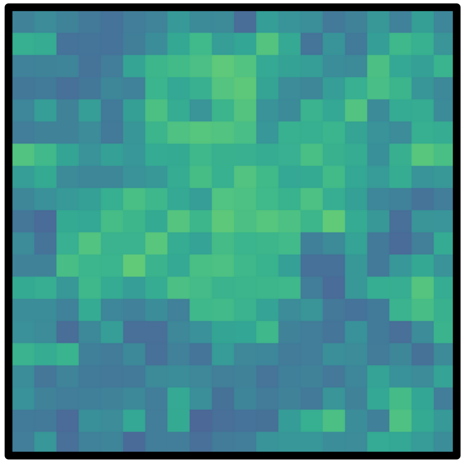
Total cost ≈ configuration production cost

- Translational symmetry ⇨ use convolutional networks
- Information within correlation length ⇨ control network footprint
  - simple affine coupling layer with no hidden layers

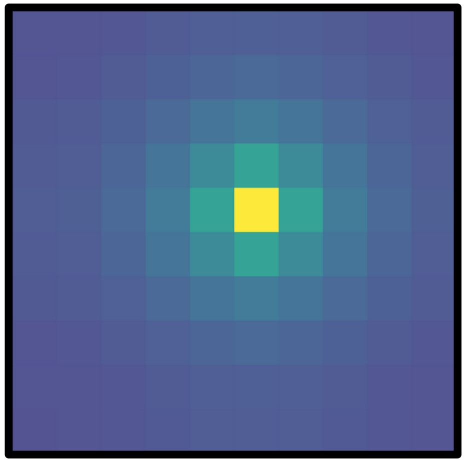
$$\phi_x \to e^{s(\phi)}\phi_x + t(\phi)$$

  - footprint can be controlled with the kernel size k of the CNNs s and t

configuration



2-point correlation



Keeping training costs low

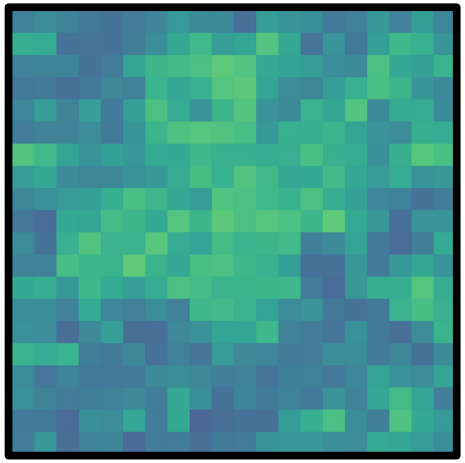
Total cost ≈ configuration production cost

- Translational symmetry ⇨ use convolutional networks
- Information within correlation length ⇨ control network footprint
  - simple affine coupling layer with no hidden layers

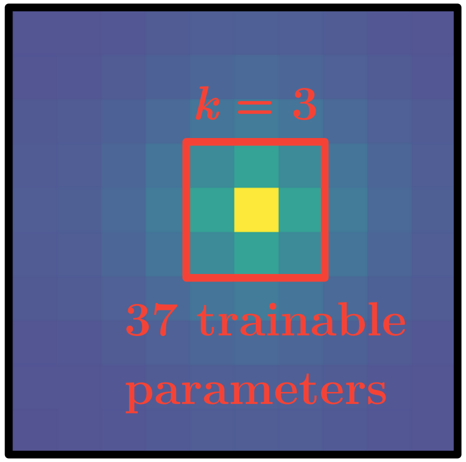
$$\phi_x \to e^{s(\phi)}\phi_x + t(\phi)$$

  - footprint can be controlled with the kernel size k of the CNNs s and t

configuration



2-point correlation

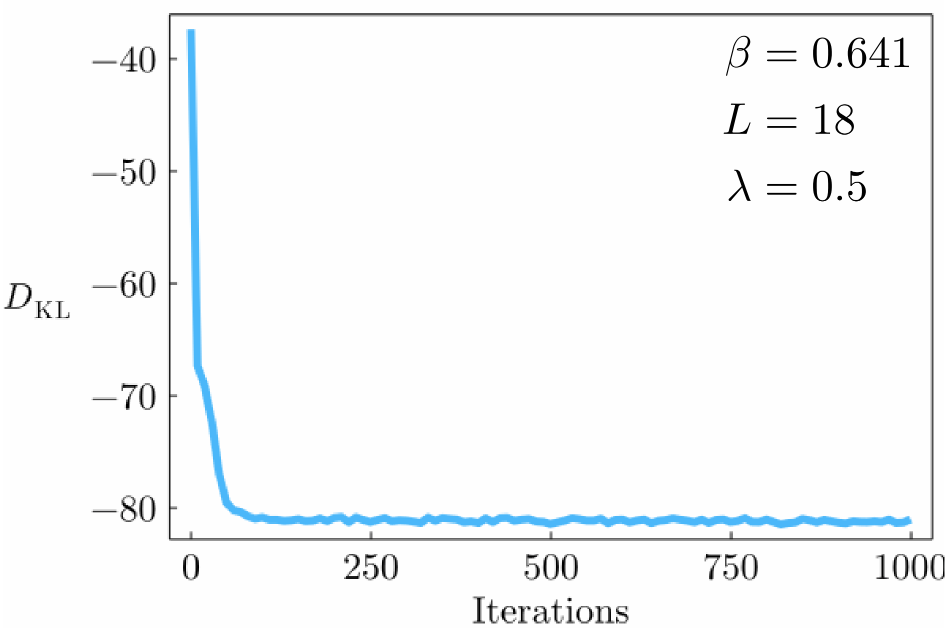


Can this simple network learn something?

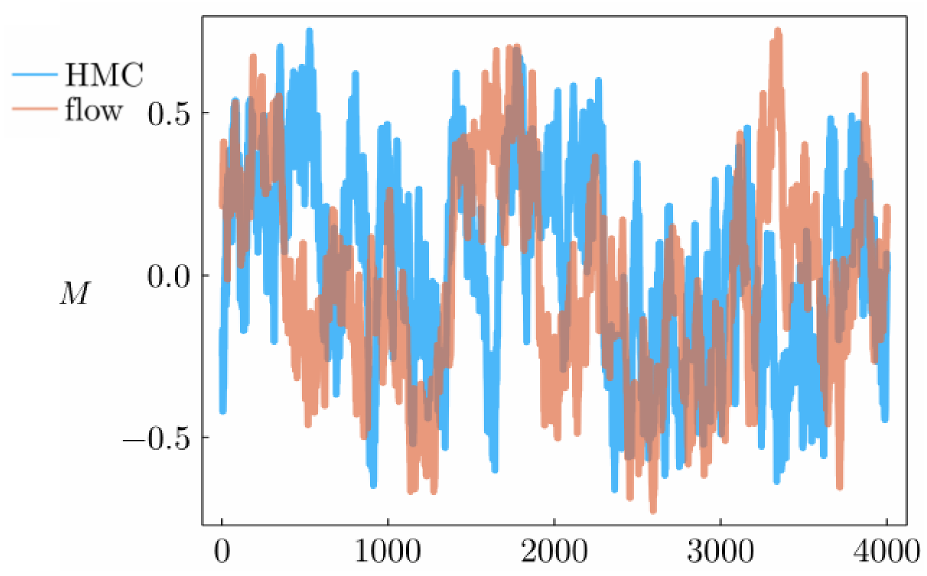
Check 1: minimal network

Minimal architecture
- 1 affine coupling layer
- $k = 3$

1. Train network minimizing KL



2. Compare magnetization history with HMC



Algorithm	$\lvert M\rvert$	τ_M
Flow	0.24472(53)	74.4(3)
HMC	0.24436(22)	100.4(2)

☆ KL divergence saturates fast

☆ Results from both algorithms are consistent with each other

☆ Learned trivializing flow reduces autocorrelations even with simple architectures

Check 2: reusability on bigger volumes

- Convolutional networks can be reused for bigger volumes

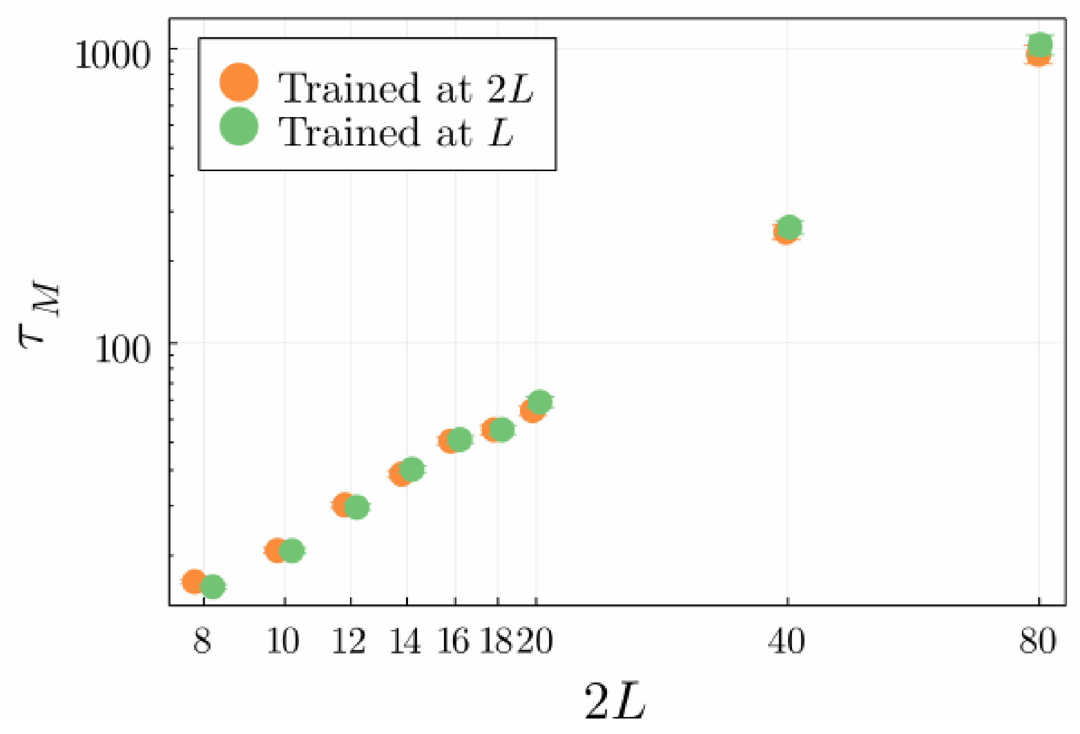
L	Acc. at L	Acc. at $2L$
3	0.3	0.2
4	0.04	0.001
5	0.002	0.00003
6	0.002	0.000007
7	0.0001	$< 10^{-7}$
8	0.0001	-
9	0.00007	-
10	0.00004	-

- The network acceptance decreases (the action is extensive)

Check 2: reusability on bigger volumes

☆ Convolutional networks can be reused for bigger volumes

L	Acc. at L	Acc. at $2L$
3	0.3	0.2
4	0.04	0.001
5	0.002	0.00003
6	0.002	0.000007
7	0.0001	$< 10^{-7}$
8	0.0001	-
9	0.00007	-
10	0.00004	-



☆ The network acceptance decreases (the action is extensive)

☆ Autocorrelation times remain the same on bigger volumes

↳ Training should be done at the correlation length

Scaling of the computational cost

$$S(\phi) = \sum_x \left[-\beta \sum_{\mu=1}^{2} \phi_{x+\mu}\phi_x + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 \right]$$

☆ Lattice with fixed physical size

Continuum

λ	L	β	Network acc.
0.5	6	0.537	0.3
0.5	8	0.576	0.04
0.5	10	0.601	0.002
0.5	12	0.616	0.002
0.5	14	0.626	0.0001
0.5	16	0.634	0.0001
0.5	18	0.641	0.00007
0.5	20	0.645	0.00004
0.5	40	0.667	-
0.5	80	0.677	-

☆ Simple network architectures: 1 affine layer

☆ Networks trained until saturation

☆ Training cost negligible w.r.t. production cost

Total cost $\approx$ configuration production cost

☆ Metropolis acceptance of the networks decreases rapidly towards the continuum

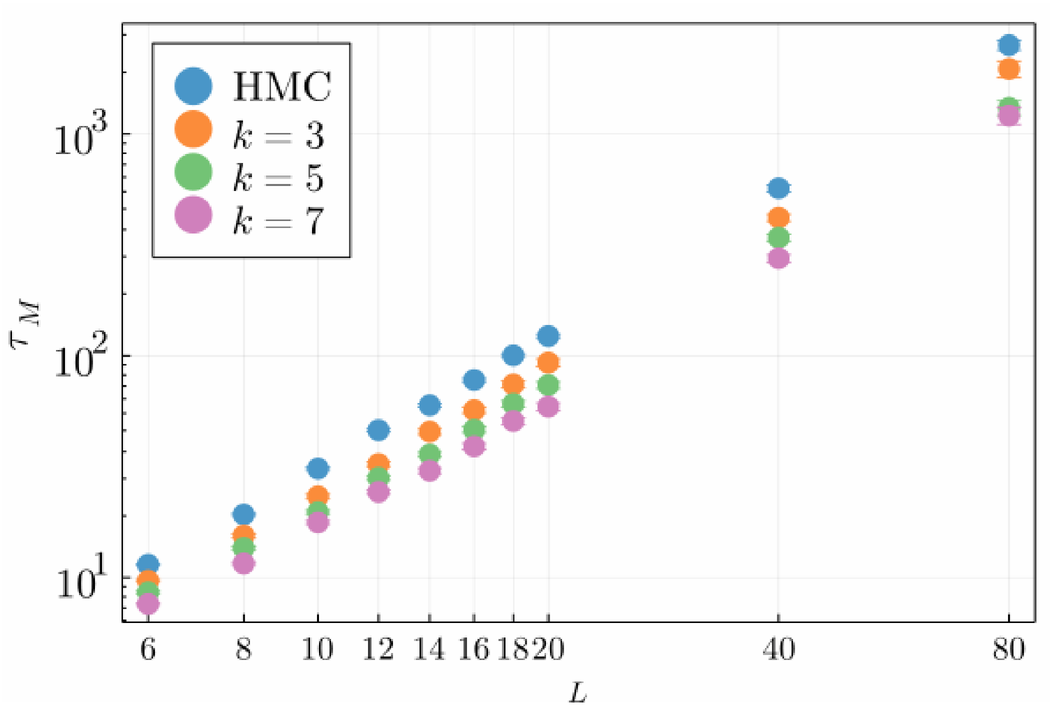
☆ But remember: we'll just use the network to change variables! $\tilde{\phi} = f(\phi)$

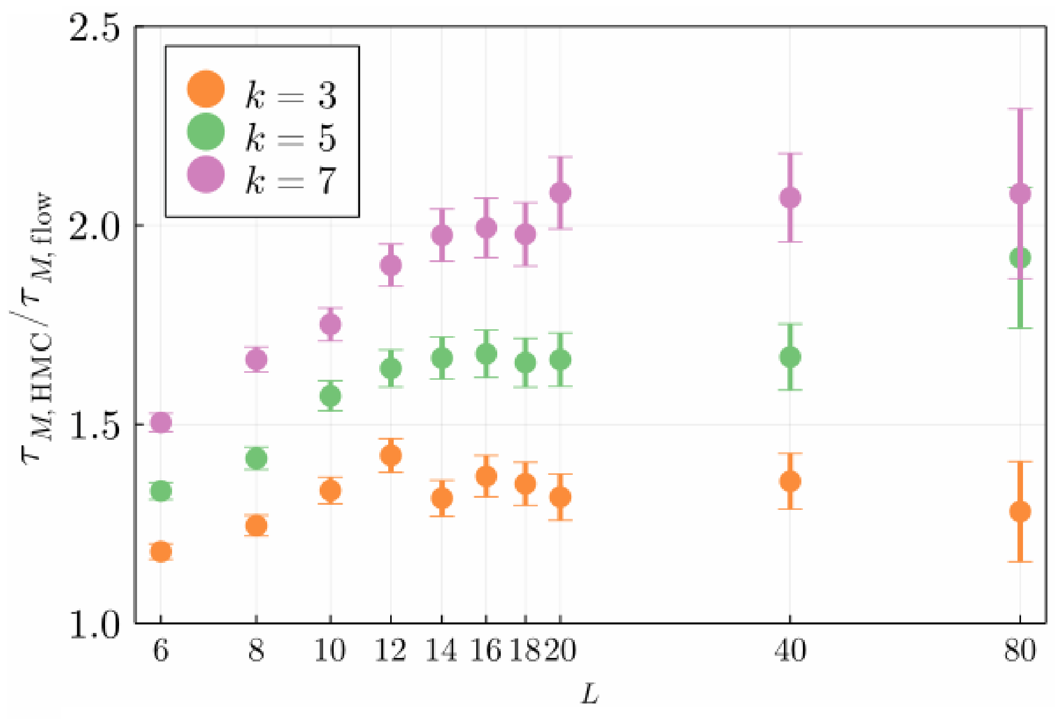
$$Z = \int D\phi\, e^{-S(\phi)} \xrightarrow{\tilde{\phi}=f(\phi)} \int D\tilde{\phi}\, e^{-S(f^{-1}(\tilde{\phi}))+\log\det J[f]} \equiv \int D\tilde{\phi}\, e^{-\tilde{S}(\tilde{\phi})}$$

The acceptance of HMC with the new action $\tilde{S}$ **does not depend on** f !

Scaling with fixed architecture

Magnetization: $M = \frac{1}{V}\sum_x \phi_x$





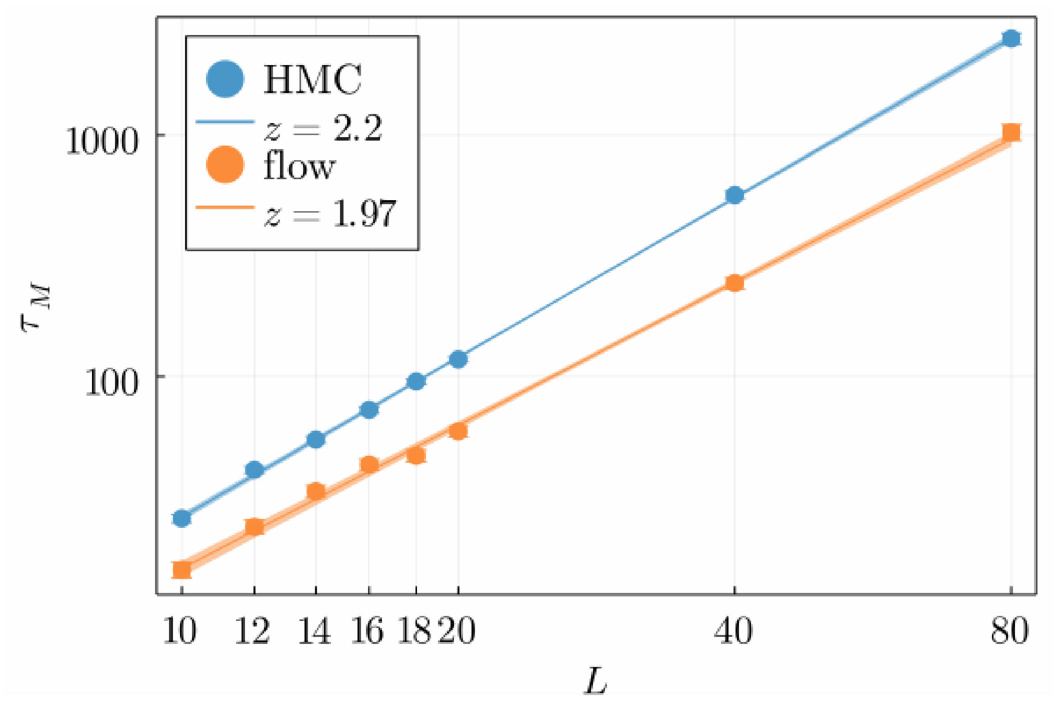
Autocorrelation times are decreased compared to HMC

For a fixed architecture the scaling does not improve

Should we scale the kernel size going to the continuum?

Scaling increasing the kernel size

Magnetization: $M = \frac{1}{V}\sum_x \phi_x$



- Fit autocorrelation to $\tau \propto \xi^z$

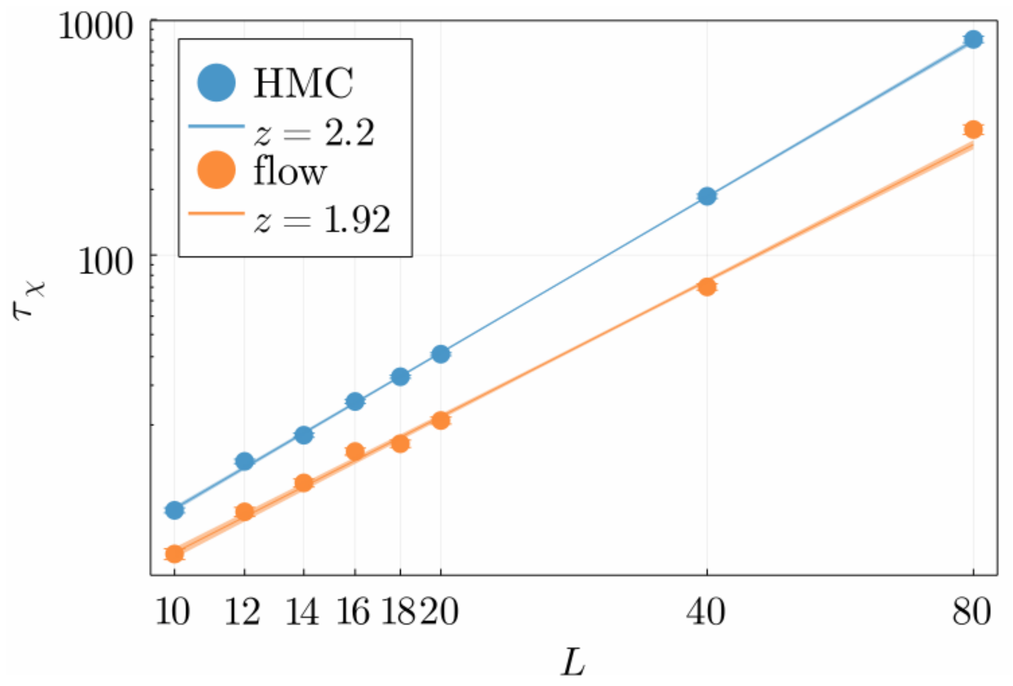
$$z_{\text{HMC}} = 2.20(4)$$
$$z_{\text{flow}} = 1.97(7)$$

- Scaling the kernel size leads to slight improvement in the autocorrelation scaling

Smeared susceptibility

Smeared one-point susceptibility: $\chi_t = \frac{1}{V}\sum_x \phi_{t,x}^2$

smeared with radius $\sim \xi$



- Fit autocorrelation to $\tau \propto \xi^z$

$$z_{\text{HMC}} = 2.20(2)$$
$$z_{\text{flow}} = 1.92(4)$$

- Scaling the kernel size leads to slight improvement in the autocorrelation scaling

Summary & Outlook

☆ This works with simple network architectures

☆ The algorithm improves the autocorrelation times of HMC, but the scaling is the same with fixed architecture

☆ The networks can be trained at a small lattice size and reused at a larger volume (with no further training)

  ⮡ In QCD: can we train at large values of quark masses?

☆ Scaling the kernel size of the convolutions slightly improves the scaling of autocorrelations

  ⮡ Can this algorithm help with topology freezing?

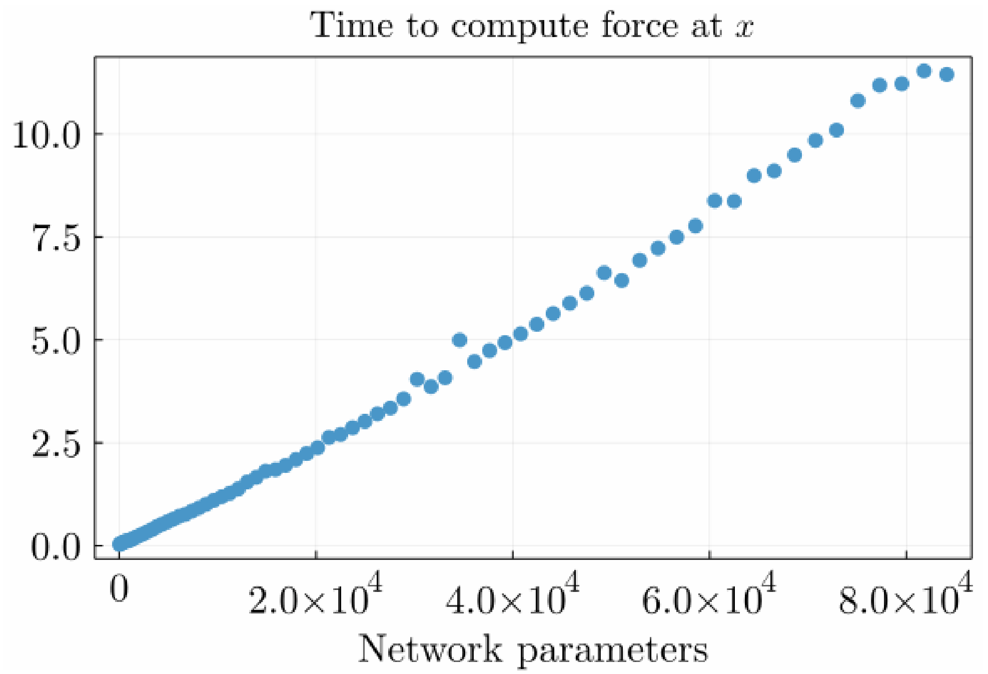
Backup

Automatic differentiation

$$Z = \int D\phi\, e^{-S(\phi)} \xrightarrow{\tilde{\phi}=f(\phi)} \int D\tilde{\phi}\, e^{-S(f^{-1}(\tilde{\phi}))+\log\det J[f]} \equiv \int D\tilde{\phi}\, e^{-\tilde{S}(\tilde{\phi})}$$

☆ We need to compute the force of the new variables: $\tilde{F}_x = \frac{\partial \tilde{S}[\tilde{\phi}]}{\partial \tilde{\phi}_x}$

↳ automatic differentiation



$N_{\text{params.}} \propto k^2$

☆ Scaling the kernel size also increases the number of operations to compute the HMC force