Stochastic normalizing flows for lattice field theory

Elia Cellini

Università degli Studi di Torino/Istituto Nazionale Fisica Nucleare

8th August 2022

Lattice 2022

Based on:
M. Caselle, E. C., A. Nada., M. Panero, *JHEP* 07 (2022) 015

Outline

Critical slowing down

The main numerical method to compute observables $\mathcal{O}$ in lattice field theories is generate a (thermalized) Markov chain

$$\underbrace{\phi^{(0)} \xrightarrow{P_p} \phi^{(1)} \xrightarrow{P_p} \dots \xrightarrow{P_p}}_{\text{thermalization}} \underbrace{\phi^{(t)} \xrightarrow{P_p} \phi^{(t+1)} \xrightarrow{P_p} \dots \to \phi^{(t+n)}}_{\text{equilibrium}}$$

and then measure $\mathcal{O}$ on sampled equilibrium configurations

Critical slowing down

The main numerical method to compute observables $\mathcal{O}$ in lattice field theories is generate a (thermalized) Markov chain

$$\underbrace{\phi^{(0)} \xrightarrow{P_p} \phi^{(1)} \xrightarrow{P_p} \dots \xrightarrow{P_p}}_{\text{thermalization}} \underbrace{\phi^{(t)} \xrightarrow{P_p} \phi^{(t+1)} \xrightarrow{P_p} \dots \rightarrow \phi^{(t+n)}}_{\text{equilibrium}}$$

and then measure $\mathcal{O}$ on sampled equilibrium configurations

The configurations sampled sequentially in a Markov Chain are **autocorrelated**

$$\dots \rightarrow \phi^{(t)} \rightarrow \phi^{(t+1)} \rightarrow \dots \rightarrow \phi^{(t+n)}$$

When a critical point is approached the autocorrelation diverges with the correlation length of the system $\rightarrow$ **critical slowing down** $\rightarrow$ drastic increase of computational cost

Deep Generative Models: Normalizing Flows

How can we sample uncorrelated configurations?

One way is use Normalizing flows (NFs) **[Rezende and Mohamed; 2015]**, a class of deep generative models able to model the target $p(\phi)$ by mapping with some tractable prior distribution $q_0(z)$

Deep Generative Models: Normalizing Flows

How can we sample uncorrelated configurations?

One way is use Normalizing flows (NFs) **[Rezende and Mohamed; 2015]**, a class of deep generative models able to model the target $p(\phi)$ by mapping with some tractable prior distribution $q_0(z)$

- successfully applied in LFTs, in particular ϕ^4 scalar field theory: **[Albergo et al.; 2019]**, **[Kanwar et al.; 2020]**, **[Nicoli et al.; 2020]**, **[Boyda et al.; 2020]**, **[Del Debbio et al.; 2021]**,**[Hackett et al.; 2021]**, **[Yamauchi et al.; 2021]**, **[Foreman et al.; 2021]**, **[de Haan et al.; 2021]**, **[Albergo et al.; 2022]**, **[Lawrence et al.; 2022]**, **[Gerdes et al.; 2022]**, **[Pawlowski and Urban; 2022]**, **[Singha et al.; 2022]**, **[Abbott et al.; 2022]**, **[Vaitl et al.; 2022]**

Normalizing Flows: structure

NFs learn families of compositions of diffeomorphisms (i.e. invertible and differentiable transformations):

$$y_N = g_\theta(y_0) = (g_N \circ \cdots \circ g_1)(y_0) \qquad y_0 \sim q_0 \qquad \theta : \textit{parameters}$$

The maps g_i are called **bijectors**

Normalizing Flows: structure

NFs learn families of compositions of diffeomorphisms (i.e. invertible and differentiable transformations):

$$y_N = g_\theta(y_0) = (g_N \circ \cdots \circ g_1)(y_0) \qquad y_0 \sim q_0 \qquad \theta : \textit{parameters}$$

The maps g_i are called **bijectors**

The generated distribution for y_N is

$$q_N(y_N) = q_0(g_\theta^{-1}(y_N)) \prod_n |\det J_n(y_n)|^{-1}$$

Normalizing Flows: structure

NFs learn families of compositions of diffeomorphisms (i.e. invertible and differentiable transformations):

$$y_N = g_\theta(y_0) = (g_N \circ \cdots \circ g_1)(y_0) \qquad y_0 \sim q_0 \qquad \theta : \textit{parameters}$$

The maps g_i are called **bijectors**

The generated distribution for y_N is

$$q_N(y_N) = q_0(g_\theta^{-1}(y_N)) \prod_n |\det J_n(y_n)|^{-1}$$

Training of NFs is done by minimizing the **Kullback-Leibler divergence**:

$$D_{KL}(q_\theta || p) = \int d\phi q_\theta(\phi) \log \frac{q_\theta(\phi)}{p(\phi)} = \int d\phi q_\theta(\phi) \log q_\theta(\phi) + S[\phi] + \log Z$$

Normalizing Flows: Partition function

A trained NF g_θ can be used to compute directly the partition function of the target:

$$Z = \int D\phi e^{-S[\phi]} = \int D\phi q_\theta(\phi) \frac{e^{-S[\phi]}}{q_\theta(\phi)} = Z_0 \int D\phi q_\theta(\phi) \tilde{w}(\phi) \approx Z_0 \langle \tilde{w} \rangle_{\phi \sim q_\theta}$$

[Nicoli et al.; 2020]

Normalizing Flows: Partition function

A trained NF g_θ can be used to compute directly the partition function of the target:

$$Z = \int D\phi e^{-S[\phi]} = \int D\phi q_\theta(\phi) \frac{e^{-S[\phi]}}{q_\theta(\phi)} = Z_0 \int D\phi q_\theta(\phi) \tilde{w}(\phi) \approx Z_0 \langle \tilde{w} \rangle_{\phi \sim q_\theta}$$

[Nicoli et al.; 2020]

Unnormalized weight:

$$\tilde{w}(\phi) = \exp\left(-\left\{S[\phi] - S_0[g_\theta^{-1}(\phi)] - Q\right\}\right) = \frac{\exp(-S[\phi])}{Z_0 q_\theta(\phi)}$$

with

$$q_\theta(\phi) = q_0(g_\theta^{-1}(\phi)) \exp(-Q) \qquad \underbrace{q_0(y_0) = \exp(-S_0[y_0])/Z_0}_{\text{e.g. normal distribution}}$$

Normalizing Flows: Observables

Observables can be computed using a reweighting procedure or a MCMC algorithm:

Reweighting

$$\langle \mathcal{O} \rangle_{\phi \sim p} = \frac{1}{\hat{Z}} \langle \mathcal{O} \tilde{w} \rangle_{\phi \sim q_\theta}$$

[Nicoli et al.; 2020]

Metropolis-Hastings

$$A(\phi^{(i-1)}, \phi') = \min\left(1, \frac{q_\theta(\phi)}{p(\phi)} \frac{p(\phi')}{q_\theta(\phi')}\right)$$

[Albergo et al.; 2019]

Normalizing Flows: Weakness

- **Multimodal-distribution**
  Training procedure could "pick" just one mode of the target
  $\rightarrow$ **equivariant normalizing flows [Nicoli et al.; 2020]**, **[Kanwar et al.; 2020]**,**[de Haan et al.; 2021]**, **[Gerdes et al.; 2022]**, **[Abbott et al.; 2022]**, ...
- **Scalability**
  measurements of observables are statistically independent
  not clear however how the training times scale when approaching the continuum limit

comprehensive discussion in **[Del Debbio et al.; 2021]**

Jarzynski's equality and Stochastic Normalizing Flows

Jarzynski's equality

Very general, intriguing relation in non-equilibrium statistical mechanics.
Free-energy differences (at equilibrium) directly calculated with an average over **non-equilibrium processes [Jarzynski; 1997]**:

$$\frac{Z}{Z_0} = \left\langle \exp\left(-\frac{\mathcal{W}}{T}\right)\right\rangle_f$$

Stochastic non-equilibrium evolutions

For an MCMC:

- the stochastic **non-equilibrium** evolution starts from a configuration sampled from the initial distribution q_0 and reaches the target (final) distribution p

$$q_0 \simeq e^{-S_{\eta_0}} \xrightarrow{P_{\eta_1}} e^{-S_{\eta_1}} \xrightarrow{P_{\eta_2}} \dots \xrightarrow{P_{\eta_N}} e^{-S_{\eta_N}} \simeq p$$

- the system evolves using regular Monte Carlo updates with transition probability P_{η_n}
- η_n is a **protocol** that interpolates the parameters of the theory between q_0 and p

Stochastic non-equilibrium evolutions

For an MCMC:

- the stochastic **non-equilibrium** evolution starts from a configuration sampled from the initial distribution q_0 and reaches the target (final) distribution p

$$q_0 \simeq e^{-S_{\eta_0}} \xrightarrow{P_{\eta_1}} e^{-S_{\eta_1}} \xrightarrow{P_{\eta_2}} \dots \xrightarrow{P_{\eta_N}} e^{-S_{\eta_N}} \simeq p$$

- the system evolves using regular Monte Carlo updates with transition probability P_{η_n}
- η_n is a **protocol** that interpolates the parameters of the theory between q_0 and p

Along the process we compute the dimensionless **work**

$$W = \sum_{n=0}^{N-1} \left\{ S_{\eta_{n+1}} [\phi_n] - S_{\eta_n} [\phi_n] \right\}$$

Stochastic non-equilibrium evolutions for LFTs

Applied in: **[Chatelain et al.; 2006]**, **[Chatelain; 2007]**, **[Hijar et al.; 2007]**, **[Caselle et al.; 2016]**, **[Francesconi et al.; 2020]**

SU(3) equation of state in 4 dimensions: **[Caselle et al.; 2018]**

Related to Annealed Importance Sampling **[Neal; 1998]**: procedure equivalent to Jarzynski's equality. Very popular in machine learning community.

A common framework

We realized that Jarzynski's relation is the same formula used to extract Z in NFs:

$$\frac{Z}{Z_0} = \langle \tilde{w}(\phi) \rangle_{\phi \sim q_\theta} = \langle \exp(-W) \rangle_{\mathrm{f}}$$

General dimensionless "work" :

$$W(y_0, \ldots, y_N) = S(y_N) - S_0(y_0) - Q(y_1, \ldots, y_N) = -\ln \tilde{w}(\phi)$$

while the "heat" Q depends on the type of flow:

A common framework

We realized that Jarzynski's relation is the same formula used to extract Z in NFs:

$$\frac{Z}{Z_0} = \langle \tilde{w}(\phi) \rangle_{\phi \sim q_\theta} = \langle \exp(-W) \rangle_{\mathrm{f}}$$

General dimensionless "work" :

$$W(y_0, \ldots, y_N) = S(y_N) - S_0(y_0) - Q(y_1, \ldots, y_N) = -\ln \tilde{w}(\phi)$$

while the "heat" Q depends on the type of flow:

normalizing flows

$$y_0 \rightarrow y_1 = g_1(y_0) \rightarrow \cdots \rightarrow y_N$$

$$Q = \sum_{n=0}^{N-1} \ln |\det J_n(y_n)|$$

$$D_{KL}(q_\theta \| p) = -\langle \ln \tilde{w}(\phi) \rangle_{\phi \sim q_\theta} + \ln \frac{Z}{Z_0}$$

stochastic non-equilibrium evolutions

$$y_0 \overset{P_{\eta_1}}{\rightarrow} y_1 \overset{P_{\eta_2}}{\rightarrow} \ldots \overset{P_{\eta_N}}{\rightarrow} y_N$$

$$Q = \sum_{n=0}^{N-1} S_{\eta_{n+1}}(y_{n+1}) - S_{\eta_{n+1}}(y_n)$$

$$D_{KL}(q_0 P_{\mathrm{f}} \| p P_{\mathrm{r}}) = \langle W \rangle_{\mathrm{f}} + \ln \frac{Z}{Z_0}$$

Stochastic Normalizing flow

Stochastic Normalizing Flows (SNFs) (introduced in **[Wu et al.; 2020]**)

$$y_0 \to g_1(y_0) \overset{P_{\eta_1}}{\to} y_1 \to g_2(y_1) \overset{P_{\eta_2}}{\to} \dots \overset{P_{\eta_N}}{\to} y_N$$

$$Q = \sum_{n=0}^{N-1} S_{\eta_{n+1}}(y_{n+1}) - S_{\eta_{n+1}}(g_n(y_n)) + \ln |\det J_n(y_n)|$$

SNF idea reworked in CRAFT approach **[Matthews et al.; 2022]**

Testing Stochastic Normalizing Flows

SNFs for the ϕ^4 2d model

Typical toy model for tests: ϕ^4 scalar field theory in 2 dimensions

$$S(\phi) = \sum_{x \in \Lambda} -2\kappa \sum_{\mu=0,1} \phi(x)\phi(x+\hat{\mu}) + (1-2\lambda)\phi(x)^2 + \lambda\phi(x)^4$$

target parameters $\kappa = 0.2$ and $\lambda = 0.022$ (as in **[Nicoli et al.; 2020]**): unbroken symmetry phase

SNFs architecture

Stochastic evolutions

η_n interpolates between the prior (normal distribution is recovered with $\kappa = \lambda = 0$) and target parameters

- linear protocol η_n
- heatbath algorithm for the stochastic updates
- $n_{sb} = \#$ of stochastic updates

Normalizing Flow

- Bijector = affine coupling layers as describe in the **RealNVP** architecture **[Dinh et al.; 2016]**
- each coupling layer has two convolutional neurons with 3×3 kernel and 1 feature map
- Affine block = odd coupling layer + even coupling layer
- $n_{ab} = \#$ of affine blocks

Testing SNFs

Goals

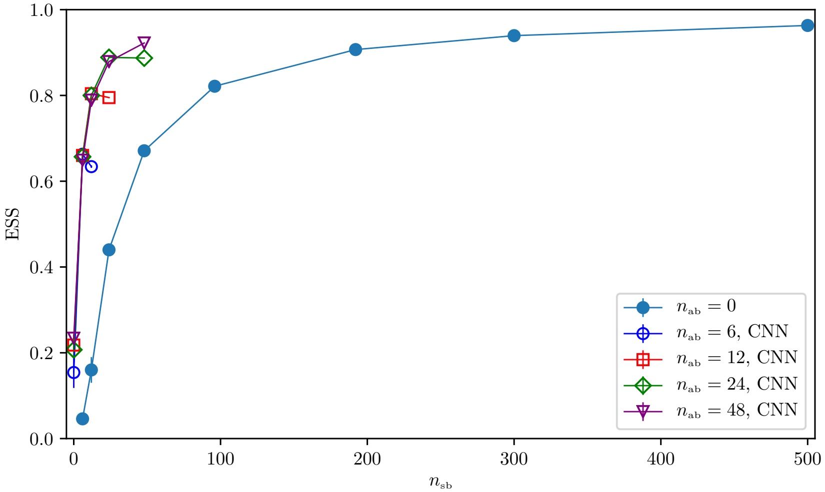
- can we train SNFs efficiently?
- can we improve both on NFs and on stochastic evolutions?

Using the Effective Sample Size as metric to evaluate architectures

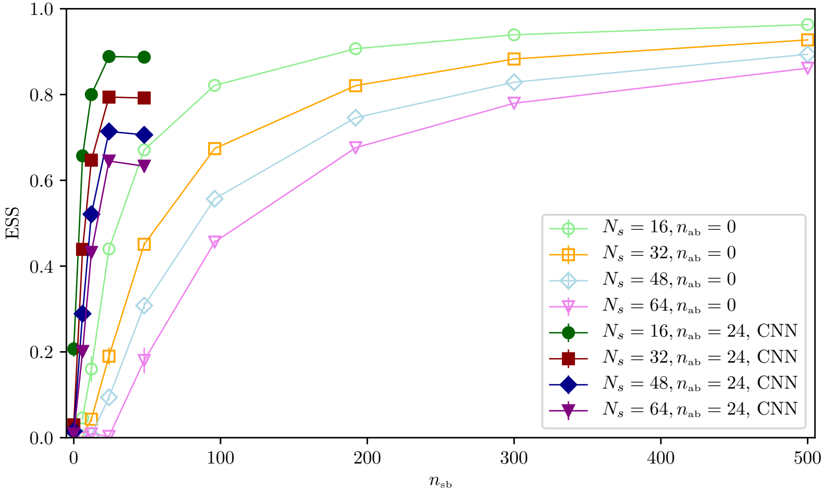
$$\mathrm{ESS} = \frac{\langle \tilde{w} \rangle_{\mathrm{f}}^2}{\langle \tilde{w}^2 \rangle_{\mathrm{f}}}$$

$\mathrm{ESS} = 1 \rightarrow$ perfect training

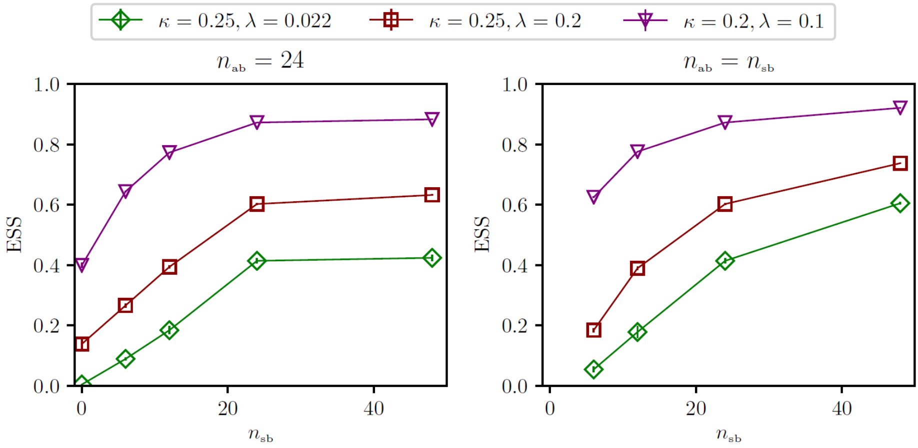
Comparing stochastic evolutions with (S)NFs on a $N_s \times N_t = 16 \times 8$ lattice,



Training length: 10^4 epochs for all volumes. Slowly-improving regime reached fast



Test with different action parameters (unbroken symmetry phase) on a $N_s \times N_t = 16 \times 8$ lattice



Interesting behaviour for all volumes: a peak for $n_{sb} = n_{ab}$?

Conclusions

The common framework between Jarzynski's equality and NFs is now explicit
General idea: use knowledge from non-equilibrium SM to create efficient SNFs

SNFs

- SNFs with CNNs and $n_{sb} = n_{ab}$ have a promising volume scaling at fixed training length

Conclusions

The common framework between Jarzynski's equality and NFs is now explicit

General idea: use knowledge from non-equilibrium SM to create efficient SNFs

SNFs

- SNFs with CNNs and $n_{sb} = n_{ab}$ have a promising volume scaling at fixed training length

SNFs vs. stochastic evolutions

- SNFs might be an even better method!
- trade-off: training for less MCMC updates

Conclusions

The common framework between Jarzynski's equality and NFs is now explicit
General idea: use knowledge from non-equilibrium SM to create efficient SNFs

SNFs

- SNFs with CNNs and $n_{sb} = n_{ab}$ have a promising volume scaling at fixed training length

SNFs vs. stochastic evolutions

- SNFs might be an even better method!
- trade-off: training for less MCMC updates

SNFs vs. normalizing flows

- improve scalability ?
- improve interpretability?

Thank you for your attention!

SNF backpropagation

- Accept/Reject step is not differentiable, we test smooth function instead of Heaviside theta with poor results
- Metropolis update:
  - Accept: $x' = x + \epsilon \rightarrow \frac{\partial x'}{\partial x} = 1$
  - Reject: $x' = x \rightarrow \frac{\partial x'}{\partial x} = 1$
- Heatbath update:
  - Accept: $x' = \epsilon + F(x) \rightarrow \frac{\partial x'}{\partial x} = \frac{\partial F(x)}{\partial x}$
  - Reject: $x' = x \rightarrow \frac{\partial x'}{\partial x} = 1$

Related works

- Annealed Importance Sampling **[Neal; 1998]**: procedure equivalent to JE. Very popular in ML community. Used in SNF paper **[Wu et al.; 2020]**
- AIS $\rightarrow$ generalized in Sequential Monte Carlo (SMC) samplers. Also well known in ML.
- SNF idea reworked in CRAFT approach **[Matthews et al.; 2022]**
- **[Vaikuntanathan and Jarzynski; 2011]**: related approach with deterministic mappings on top of non-equilibrium transformations. No neural networks.