Stochastic normalizing flows for lattice field theory

Alessandro Nada

Università degli Studi di Torino

8th June 2022

University of Plymouth - Zoom seminar

Based on:
M. Caselle, E. Cellini, A. N., M. Panero, arXiv:2201.08862

Outline

Lattice field theory - a (very) quick primer

Simple case: scalar field theory on a lattice:

- discretize space-time into a square lattice of spacing a
- scalar field variables placed on sites, action discretized in a consistent way
- compute v.e.v. as in statistical mechanics

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \prod_i \mathrm{d}\phi_i \; \underbrace{\mathcal{O}(\phi)}_{\text{measure}} \underbrace{\exp(-S(\phi))}_{\text{sample}}$$

  with the very complicated probability distribution $p(\phi) = \exp(-S(\phi))/Z$
- perform continuum extrapolation $a \to 0$

Lattice field theories need an efficient way to generate configurations ϕ according to $p(\phi)$

Lattice field theory - a (very) quick primer

Simple case: scalar field theory on a lattice:

- discretize space-time into a square lattice of spacing a
- scalar field variables placed on sites, action discretized in a consistent way
- compute v.e.v. as in statistical mechanics

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \prod_i \mathrm{d}\phi_i \; \underbrace{\mathcal{O}(\phi)}_{\text{measure}} \underbrace{\exp(-S(\phi))}_{\text{sample}}$$

  with the very complicated probability distribution $p(\phi) = \exp(-S(\phi))/Z$
- perform continuum extrapolation $a \to 0$

Lattice field theories need an efficient way to generate configurations ϕ according to $p(\phi)$

Elegant numerical solution: generate a (thermalized) Markov chain

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

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

the MCMC algorithm of choice (Metropolis...) is defined by the transition probability P_p

Critical slowing down

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

$$\ldots \to \phi^{(t)} \to \phi^{(t+1)} \to \ldots \to \phi^{(t+n)}$$

The measure of this autocorrelation is given by τ_{int}

$\to$ # effectively independent configurations $= n/2\tau_{\text{int}}$

Critical slowing down

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

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

The measure of this autocorrelation is given by τ_{int}

$\to$ # effectively independent configurations $= n/2\tau_{\text{int}}$

When a critical point is approached τ_{int} diverges with the correlation length of the system

$\to$ **critical slowing down**

The continuum limit $a \to 0$ is a critical point, so

$$\tau_{\text{int}} \simeq a^{-z}$$

where z depends on the algorithm and on the observable under study

Configurations become more and more autocorrelated with each other as the lattice spacing gets finer

What if every new configuration is sampled independently from the previous one?

Deep generative models in lattice field theory

What if every new configuration is sampled independently from the previous one?

Deep generative models might be flexible enough to model the target $p(\phi)$ by a mapping with some tractable distribution $q_0(z)$

Normalizing Flows are an efficient architecture that can provide this mapping

Deep generative models in lattice field theory

What if every new configuration is sampled independently from the previous one?

Deep generative models might be flexible enough to model the target $p(\phi)$ by a mapping with some tractable distribution $q_0(z)$

Normalizing Flows are an efficient architecture that can provide this mapping

$\rightarrow$ 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]**, ...

Related to the idea of trivializing maps **[Lüscher; 2009]**

Normalizing flows: structure

Normalizing flows are a deterministic mapping

$$g_\theta(y_0) = (g_N \circ \cdots \circ g_1)(y_0) \qquad\qquad y_0 \sim q_0$$

composed of N invertible transformations $\rightarrow$ the **coupling layers** g_i

Normalizing flows: structure

Normalizing flows are a deterministic mapping

$$g_\theta(y_0) = (g_N \circ \cdots \circ g_1)(y_0) \qquad\qquad y_0 \sim q_0$$

composed of N invertible transformations $\rightarrow$ the **coupling layers** g_i

At each layer the field variables y are transformed

$$y_{n+1} = g_n(y_n)$$

The generated distribution for y_N is

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

and depends on the **prior** distribution q_0 (e.g. a normal distribution of unit variance) and on the Jacobian of each coupling layer

Normalizing flows: affine layers

Transformations g_n must be invertible + the Jacobian has to be efficiently computable

A class of coupling layers called **affine layers** meets this criteria

- The variables y are divided into two partitions A and B
- For each layer, one is kept "frozen" while the other is transformed following

$$g_n : \begin{cases} y_{\mathrm{A}}^{n+1} = y_{\mathrm{A}}^{n} \\ y_{\mathrm{B}}^{n+1} = e^{-s(y_{\mathrm{A}}^{n})} y_{\mathrm{B}}^{n} + t(y_{\mathrm{A}}^{n}) \end{cases}$$

- s and t are the neural networks where the trainable parameters θ are
- **RealNVP** architecture **[Dinh et al.; 2016]**

Natural choice for lattice variables: checkerboard (i.e. even-odd) partitioning

Affine block = even c. layer + odd c. layer

Normalizing flows: training

Training = iterative procedure that brings the generated distribution q_N as close as possible to the target p

loss $\rightarrow$ the quantity the training algorithm minimizes in order to reach the target

Typical choice is the **Kullback-Leibler** divergence: measure of the "similarity" between two distributions

$$\tilde{D}_{\mathrm{KL}}(q_N \| p) = \int \mathrm{d}\phi \, q_N(\phi) \left[\ln q_N(\phi) - \ln p(\phi)\right]$$

Normalizing flows: training

Training = iterative procedure that brings the generated distribution q_N as close as possible to the target p

loss $\rightarrow$ the quantity the training algorithm minimizes in order to reach the target

Typical choice is the **Kullback-Leibler** divergence: measure of the "similarity" between two distributions

$$\tilde{D}_{\mathrm{KL}}(q_N \| p) = \int \mathrm{d}\phi \, q_N(\phi) \left[\ln q_N(\phi) - \ln p(\phi)\right]$$

Also needed: an efficient way of computing the gradient of the loss with respect to the flow parameters θ

$$\nabla_\theta \tilde{D}_{\mathrm{KL}}(q_N \| p)$$

$\rightarrow$ **backpropagation** algorithm: the overall gradient is calculated combining the intermediate gradients at each layer n, which can be stored in memory during a forward pass through the flow

$$\nabla_\theta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial y_N} \frac{\partial y_N}{\partial y_{N-1}} \dots$$

Normalizing flows and the free energy

How do we use a trained flow g_θ and the distribution q_N?

Normalizing flows and the free energy

How do we use a trained flow g_θ and the distribution q_N?

Compute (not the only possibility: see independent MH)

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathrm{d}\phi \, \mathcal{O}(\phi) q_N(\phi) \frac{p(\phi)}{q_N(\phi)} = \frac{Z_0}{Z} \int \mathrm{d}\phi \, \underbrace{q_N(\phi)}_{\text{sample}} \underbrace{\mathcal{O}(\phi)\tilde{w}(\phi)}_{\text{measure}} = \frac{\langle \mathcal{O}(\phi)\tilde{w}(\phi) \rangle_{\phi \sim q_N}}{\langle \tilde{w}(\phi) \rangle_{\phi \sim q_N}}$$

as in a reweighting step with weight

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

with

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

Normalizing flows and the free energy

How do we use a trained flow g_θ and the distribution q_N?

Compute (not the only possibility: see independent MH)

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathrm{d}\phi \, \mathcal{O}(\phi) q_N(\phi) \frac{p(\phi)}{q_N(\phi)} = \frac{Z_0}{Z} \int \mathrm{d}\phi \, \underbrace{q_N(\phi)}_{\text{sample}} \underbrace{\mathcal{O}(\phi)\tilde{w}(\phi)}_{\text{measure}} = \frac{\langle \mathcal{O}(\phi)\tilde{w}(\phi) \rangle_{\phi \sim q_N}}{\langle \tilde{w}(\phi) \rangle_{\phi \sim q_N}}$$

as in a reweighting step with weight

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

with

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

Get Z directly by sampling from q_N **[Nicoli et al.; 2020]**

$$Z = \int \mathrm{d}\phi \, \exp(-S[\phi]) = Z_0 \int \mathrm{d}\phi \, q_N(\phi) \tilde{w}(\phi) = Z_0 \langle \tilde{w}(\phi) \rangle_{\phi \sim q_N}$$

Normalizing flows and the free energy

How do we use a trained flow g_θ and the distribution q_N?

Compute (not the only possibility: see independent MH)

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \mathrm{d}\phi \, \mathcal{O}(\phi) q_N(\phi) \frac{p(\phi)}{q_N(\phi)} = \frac{Z_0}{Z} \int \mathrm{d}\phi \, \underbrace{q_N(\phi)}_{\text{sample}} \underbrace{\mathcal{O}(\phi)\tilde{w}(\phi)}_{\text{measure}} = \frac{\langle \mathcal{O}(\phi)\tilde{w}(\phi) \rangle_{\phi \sim q_N}}{\langle \tilde{w}(\phi) \rangle_{\phi \sim q_N}}$$

as in a reweighting step with weight

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

with

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

Get Z directly by sampling from q_N **[Nicoli et al.; 2020]**

$$Z = \int \mathrm{d}\phi \, \exp(-S[\phi]) = Z_0 \int \mathrm{d}\phi \, q_N(\phi) \tilde{w}(\phi) = Z_0 \langle \tilde{w}(\phi) \rangle_{\phi \sim q_N}$$

And the loss:

$$\tilde{D}_{\mathrm{KL}}(q_N \| p) = -\langle \ln \tilde{w}(\phi) \rangle_{\phi \sim q_N} + \ln \frac{Z}{Z_0}$$

Some possible issues with NFs

- **multi-modal distributions**
  in the presence of multiple vacua the training procedure "picks" only one

  "mode-collapse": only one mode of the distribution is sampled by the flow

  see **[Hackett et al.; 2021]**

- **scalability**
  measurements of v.e.v. are statistically independent (no autocorrelation)

  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

Free-energy differences (at equilibrium) directly calculated with an average over **non-equilibrium processes [Jarzynski; 1997]**:

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

Jarzynski's equality

Free-energy differences (at equilibrium) directly calculated with an average over **non-equilibrium processes [Jarzynski; 1997]**:

$$\frac{Z}{Z_0} = \langle \exp(-W) \rangle_f$$

For an MCMC

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

$$q_0 = \exp(-S_0)/Z_0 \to \cdots \to p = \exp(-S)/Z$$

- happens over N intermediate steps
- the system evolves using regular Monte Carlo updates with transition probability P_{η_n}
- but the $P_{\eta_n}(y_n \to y_{n+1})$ change along the evolution (i.e.: the action S_{η_n} changes)
- η_n is a **protocol** that interpolates the parameters of the theory between q_0 and p

$$q_0 \simeq e^{-S_{\eta_0}} \to e^{-S_{\eta_1}} \to \cdots \to p \simeq e^{-S_{\eta_N}}$$

Jarzynski's equality

Free-energy differences (at equilibrium) directly calculated with an average over **non-equilibrium processes [Jarzynski; 1997]**:

$$\frac{Z}{Z_0} = \langle \exp(-W) \rangle_f$$

For an MCMC

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

$$q_0 = \exp(-S_0)/Z_0 \to \cdots \to p = \exp(-S)/Z$$

- happens over N intermediate steps
- the system evolves using regular Monte Carlo updates with transition probability P_{η_n}
- but the $P_{\eta_n}(y_n \to y_{n+1})$ change along the evolution (i.e.: the action S_{η_n} changes)
- η_n is a **protocol** that interpolates the parameters of the theory between q_0 and p

$$q_0 \simeq e^{-S_{\eta_0}} \to e^{-S_{\eta_1}} \to \cdots \to p \simeq e^{-S_{\eta_N}}$$

Along the process we compute the **work**

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

Out-of-equilibrium stochastic evolutions

Closer look at the average on the processes in the equality:

$$\frac{Z}{Z_0} = \langle \exp(-W) \rangle_f = \int \mathrm{d}y_0 \, \mathrm{d}y_1 \ldots \mathrm{d}y_N \, q_0(y_0) \, P_\mathrm{f}[y_0, y_1, \ldots, y_N] \, \exp(-W)$$

with

$$P_\mathrm{f}[y_0, y_1, \ldots, y_N] = \prod_{n=0}^{N-1} P_{\eta_n}(y_n \to y_{n+1})$$

- the *actual* probability distribution at each step is NOT the equilibrium distribution $\sim \exp(-S_{\eta_n})$: it's a non-equilibrium process!
- the $\langle \ldots \rangle_f$ average is taken over as many evolutions as possible (all independent from each other!)

A common framework: Stochastic Normalizing Flows

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

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

as for deterministic mappings $\langle \dots \rangle_{\phi \sim q_N} = \langle \dots \rangle_{\mathrm{f}}$.

The "work" is simply

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

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

normalizing flows

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

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

stochastic non-equilibrium evolutions

$$y_0 \overset{P_{\eta_1}}{\to} y_1 \overset{P_{\eta_2}}{\to} \cdots \overset{P_{\eta_N}}{\to} y_N$$

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

A common framework: Stochastic Normalizing Flows

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

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

as for deterministic mappings $\langle \dots \rangle_{\phi \sim q_N} = \langle \dots \rangle_{\mathrm{f}}$.

The "work" is simply

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

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

normalizing flows

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

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

stochastic non-equilibrium evolutions

$$y_0 \overset{P_{\eta_1}}{\to} y_1 \overset{P_{\eta_2}}{\to} \cdots \overset{P_{\eta_N}}{\to} y_N$$

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

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

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

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

A common framework: Stochastic Normalizing Flows

The proper KL divergence is

$$\tilde{D}_{\mathrm{KL}}(q_0 P_{\mathrm{f}} \| p P_{\mathrm{r}}) = \int \mathrm{d}y_0 \, \mathrm{d}y_1 \ldots \mathrm{d}y_N \, q_0(y_0) P_{\mathrm{f}}[y_0, y_1, \ldots, y_N] \ln \frac{q_0(y_0) P_{\mathrm{f}}[y_0, y_1, \ldots, y_N]}{p(y_N) P_{\mathrm{r}}[y_N, y_{N-1}, \ldots, y_0]}$$

$\rightarrow$ measure of how reversible the process is!

A common framework: Stochastic Normalizing Flows

The proper KL divergence is

$$\tilde{D}_{\mathrm{KL}}(q_0 P_{\mathrm{f}} \| p P_{\mathrm{r}}) = \int \mathrm{d}y_0 \, \mathrm{d}y_1 \ldots \mathrm{d}y_N \, q_0(y_0) P_{\mathrm{f}}[y_0, y_1, \ldots, y_N] \ln \frac{q_0(y_0) P_{\mathrm{f}}[y_0, y_1, \ldots, y_N]}{p(y_N) P_{\mathrm{r}}[y_N, y_{N-1}, \ldots, y_0]}$$

$\rightarrow$ measure of how reversible the process is!

In general we have simply

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

If we go back to NFs: the same definition simplifies (using the change of variables theorem) to

$$\tilde{D}_{\mathrm{KL}}(q_0 P_{\mathrm{f}} \| p P_{\mathrm{r}}) \rightarrow \tilde{D}_{\mathrm{KL}}(q_N \| p) = -\langle \ln \tilde{w}(\phi) \rangle_{\phi \sim q_N} + \ln \frac{Z}{Z_0}$$

due to the deterministic nature of the mapping.

Some comparisons

	normalizing flows	stochastic evolutions	SNFs
preparation	training	setting the protocol η_n	both
forward prob. P_f	$P_\mathrm{f} = \prod_n P_n(y_n \to y_{n+1})$		
transition prob. P_n	$\delta(y_{n+1} - g_n(y_n))$	$P_{\eta_n}(y_n \to y_{n+1})$	uses both
KL divergence	$\tilde{D}_\mathrm{KL}(q_N \Vert p)$	$\tilde{D}_\mathrm{KL}(q_0 P_\mathrm{f} \Vert p P_\mathrm{r})$	
"work"	$W = S - S_0 - Q = -\ln \tilde{w}$		
"heat" Q	$\sum_{n=0}^{N-1} \ln \lvert\det J_n(y_n)\rvert$	$\sum_{n=0}^{N-1} S_{\eta_{n+1}}(y_{n+1}) - S_{\eta_{n+1}}(y_n)$	both
e.v. $\langle \mathcal{O} \rangle$	$\frac{\langle \mathcal{O}(y_N)\tilde{w}(y_N)\rangle_{y_N \sim q_N}}{\langle \tilde{w}(y_N)\rangle_{y_N \sim q_N}}$	$\frac{\langle \mathcal{O}(y_N)\exp(-W(y_0 \to y_N))\rangle_\mathrm{f}}{\langle \exp(-W(y_0 \to y_N))\rangle_\mathrm{f}}$	

Testing Stochastic Normalizing Flows

SNFs for the ϕ^4 2d model

Typical toy model for tests: ϕ^4 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 for the ϕ^4 2d model

Typical toy model for tests: ϕ^4 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

Protocol

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

SNFs for the ϕ^4 2d model

Typical toy model for tests: ϕ^4 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

Protocol

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

Coupling layers and NN

- $n_{ab} = \#$ of affine blocks
- inside each affine layer neural networks are CNNs with 1 hidden layer, 3×3 kernel and 1 feature map

Testing SNFs

Goals

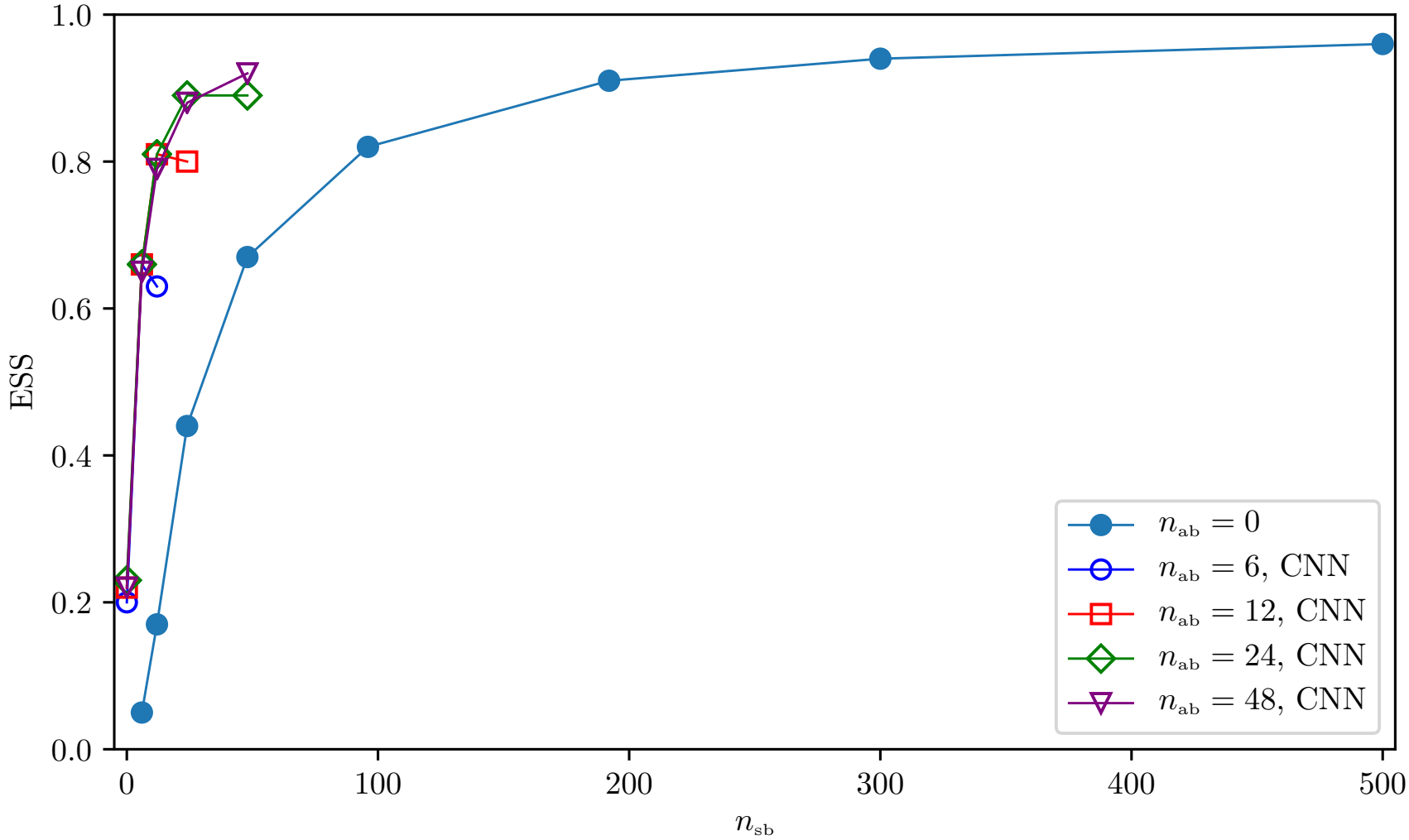
- can we train SNFs efficiently?
- can we improve both on NFs and on stochastic evolutions?
- how do the SNFs behave for a given neural network architecture?
- previous experience with stochastic evolutions with JE: the $\mathrm{SU}(3)$ equation of state in $(3+1)$D **[Caselle et al.; 2018]**. Can we learn something from it?

Using the Effective Sample Size as metric to evaluate architectures

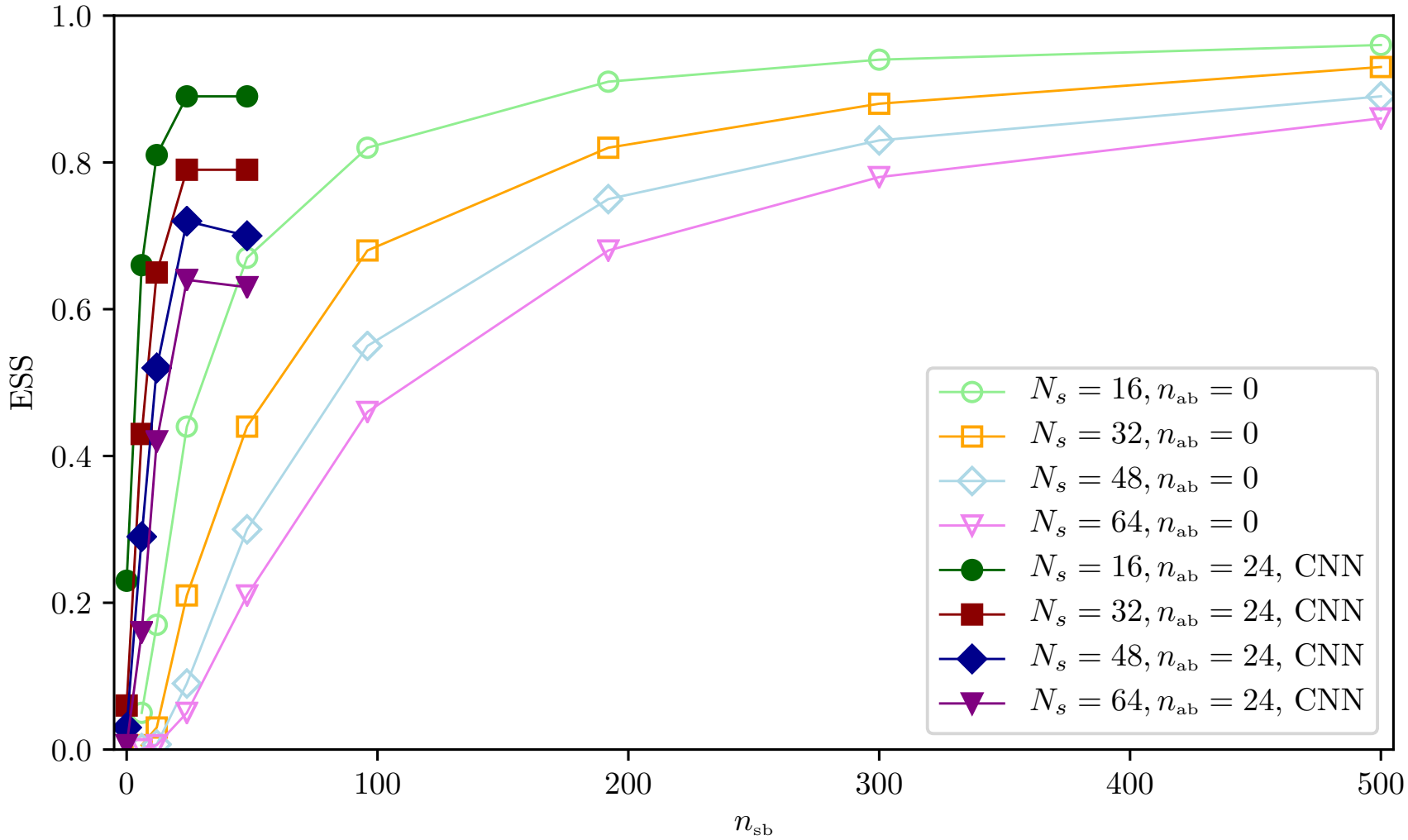
$$\mathrm{ESS} = \frac{\langle \tilde{w} \rangle_{\mathrm{f}}^2}{\langle \tilde{w}^2 \rangle_{\mathrm{f}}}$$

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

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

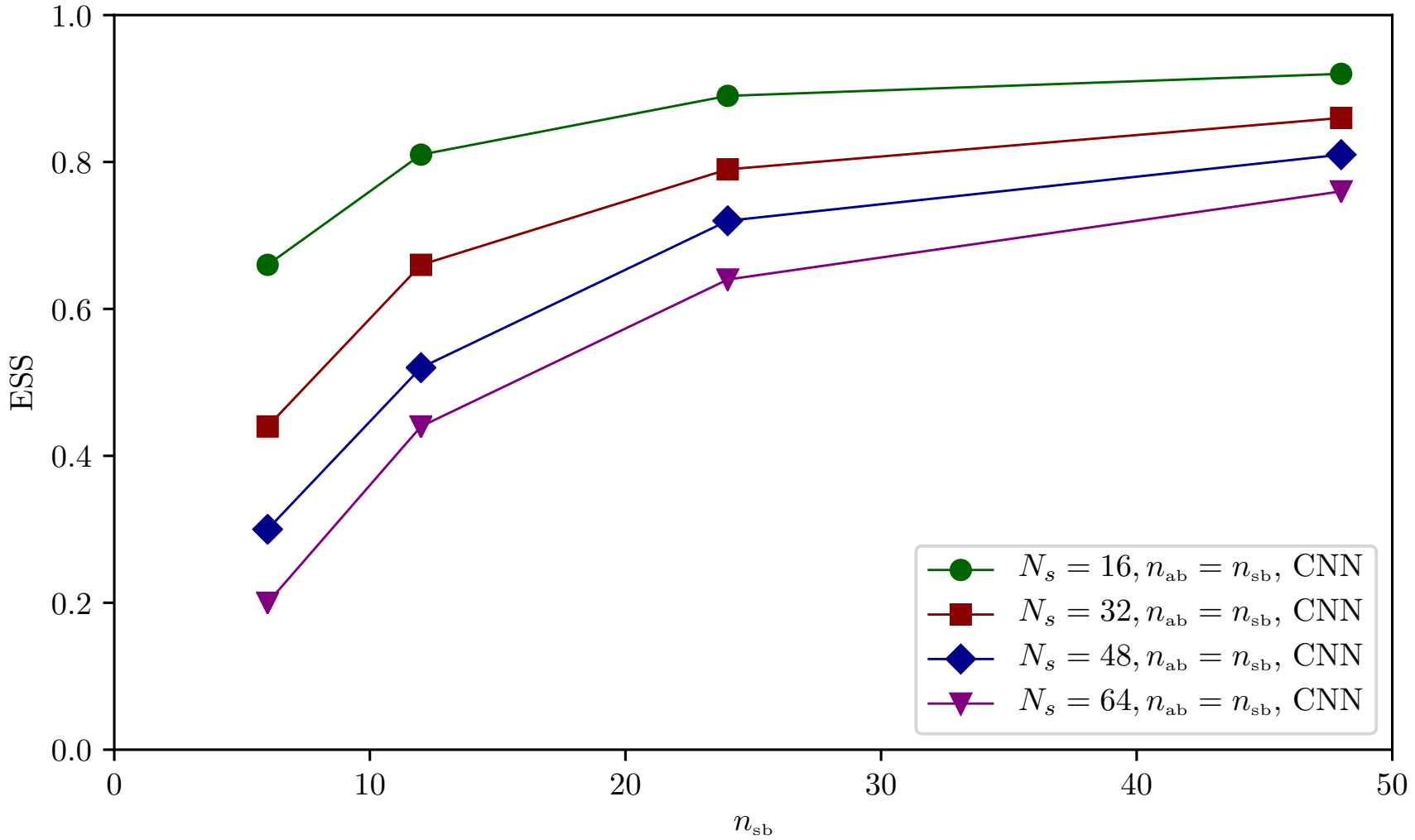


Training length: 10^4 epochs for all volumes. Slowly-improving regime reached fast



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

SNFs with $n_{sb} = n_{ab}$ as a possible recipe for efficient scaling



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

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 vs. stochastic evolutions

- Jarzynski's equality provides a way to compute Z and $\langle O \rangle$ (which works well also in LGTs, see $\mathrm{SU}(3)$ e.o.s. **[Caselle et al.; 2018]**)
- SNFs might be an even better method!
- trade-off: training for less MCMC updates
- very interesting for thermodynamic applications (or similar)

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 vs. stochastic evolutions

- Jarzynski's equality provides a way to compute Z and $\langle O \rangle$ (which works well also in LGTs, see $\mathrm{SU}(3)$ e.o.s. **[Caselle et al.; 2018]**)
- SNFs might be an even better method!
- trade-off: training for less MCMC updates
- very interesting for thermodynamic applications (or similar)

SNFs vs. normalizing flows

- improve scalability and interpretability?
- SNFs with CNNs and $n_{sb} = n_{ab}$ have a promising volume scaling at fixed training length
- training could be qualitatively "guided" towards the target by the protocol, but ultimately might also be limited by it

Thank you for your attention!

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.

Taking cues from the $\mathrm{SU}(3)$ e.o.s.

Is there anything we can learn from out-of-equilibrium stochastic processes that we can apply to stochastic normalizing flows?

Relevant application: large-scale computation of the $\mathrm{SU}(3)$ equation of state **[Caselle et al.; 2018]**

goal: extract the pressure with Jarzynski's equality

$$\frac{p(T)}{T^4} - \frac{p(T_0)}{T_0^4} = \left(\frac{N_t}{N_s}\right)^3 \log\langle e^{-W_{\mathrm{SU}(N_c)}}\rangle_{\mathrm{f}}$$

evolution in β (inverse coupling) $\rightarrow$ changes lattice spacing $a \rightarrow$ changes temperature $T = 1/(aN_t)$ in $[\mathrm{f} = T_0 \rightarrow T]$ process

Taking cues from the $\mathrm{SU}(3)$ e.o.s.

Is there anything we can learn from out-of-equilibrium stochastic processes that we can apply to stochastic normalizing flows?

Relevant application: large-scale computation of the $\mathrm{SU}(3)$ equation of state **[Caselle et al.; 2018]**

goal: extract the pressure with Jarzynski's equality

$$\frac{p(T)}{T^4} - \frac{p(T_0)}{T_0^4} = \left(\frac{N_t}{N_s}\right)^3 \log\langle e^{-W_{\mathrm{SU}(N_c)}}\rangle_{\mathrm{f}}$$

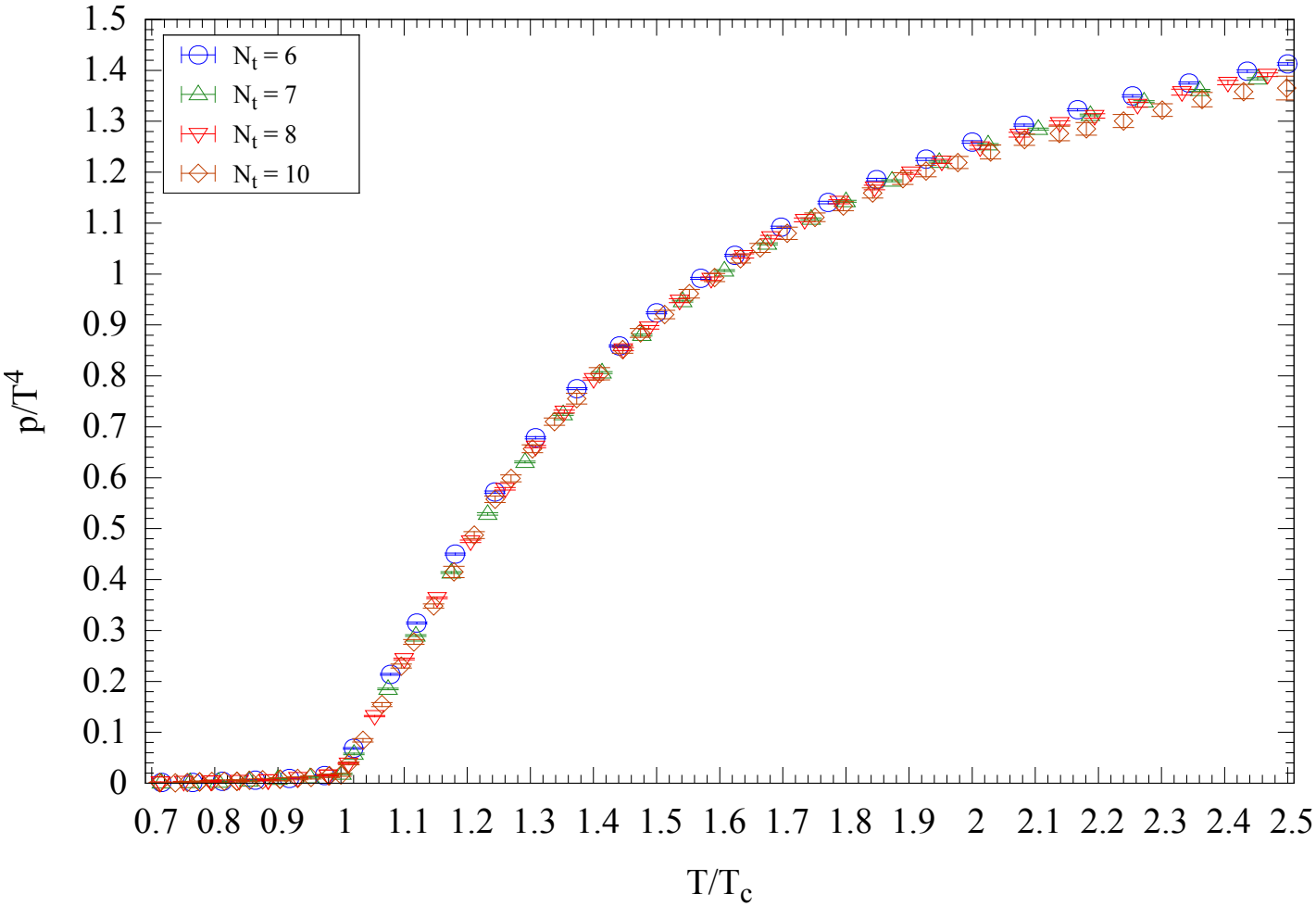
evolution in β (inverse coupling) $\rightarrow$ changes lattice spacing a $\rightarrow$ changes temperature $T = 1/(aN_t)$ in $[\mathrm{f} = T_0 \rightarrow T]$ process

Important difference: the prior is not a random distribution, but a thermalized Markov chain at a certain inverse coupling β_0 (or temperature T_0)

Observation: for systems with many d.o.f. (large volumes), Jarzynski's equality "converges" more easily to the right result when stochastic evolutions are very close to equilibrium (i.e. N is large, evolution is slow). "Easy" way to obtain reversibility.

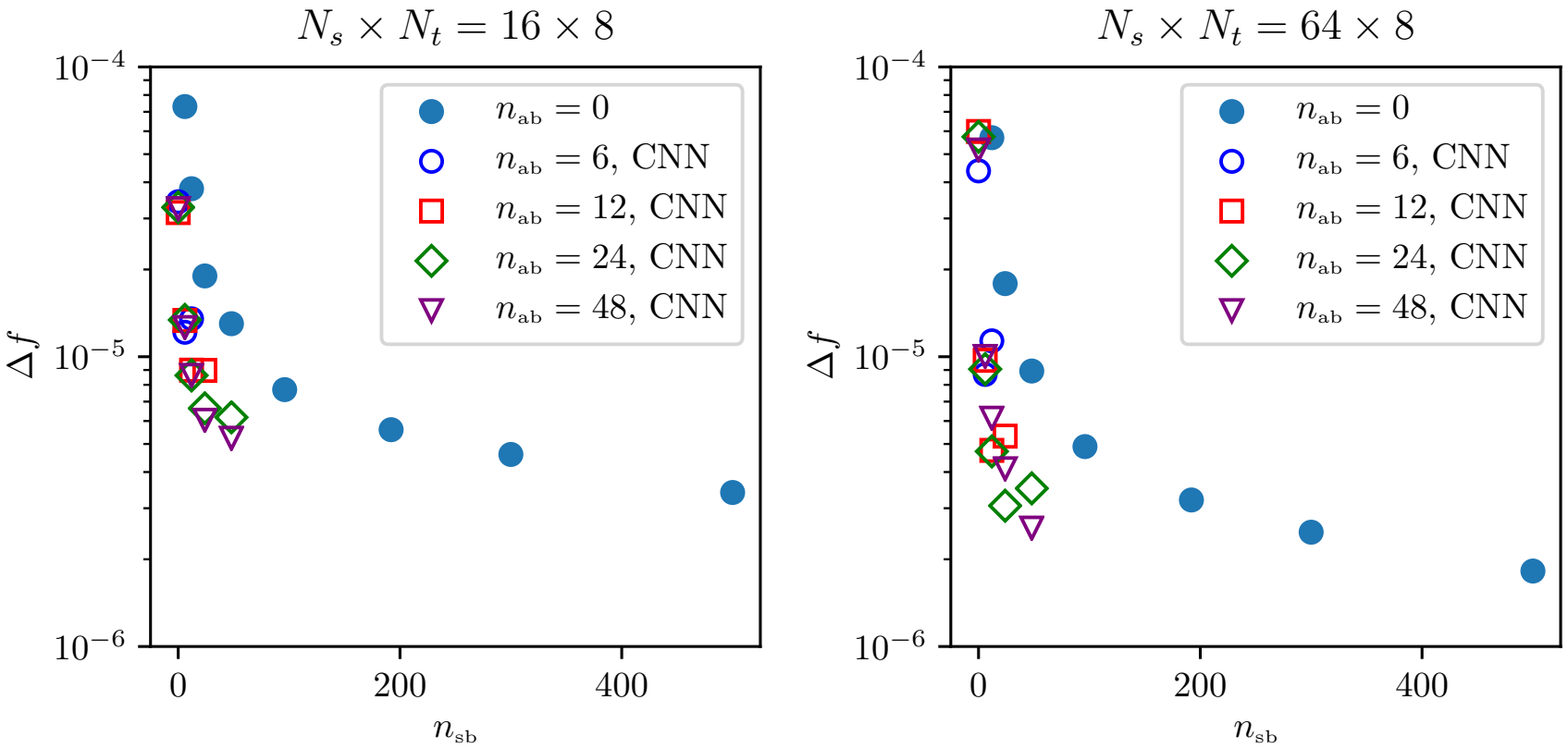
SU(3) e.o.s. with Jarzynski's equality

SU(3) pressure in $(3+1)d$ across the deconfinement transition with Jarzynski's equality



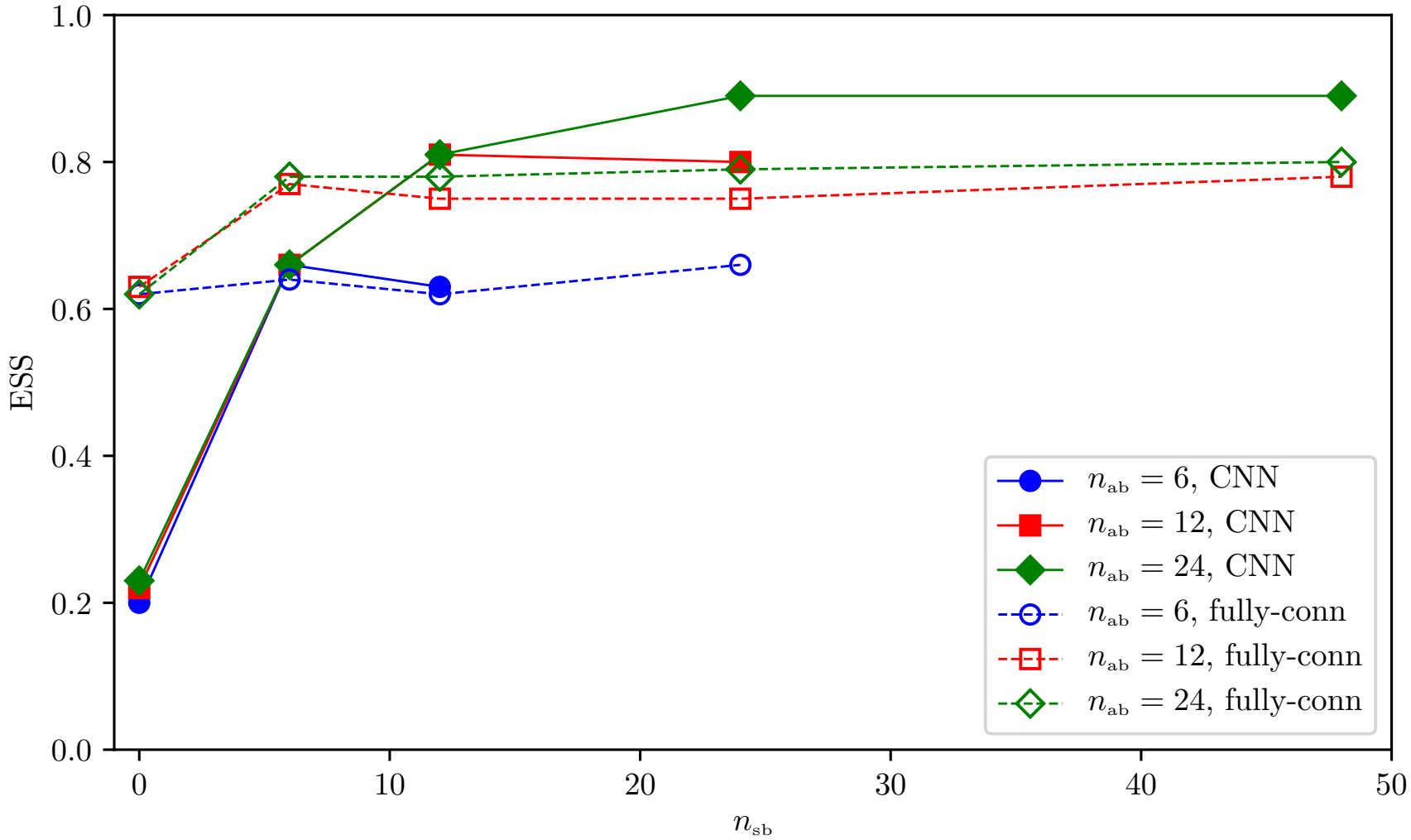
Does it work for SNFs?

More transparent comparison: error on the free-energy density

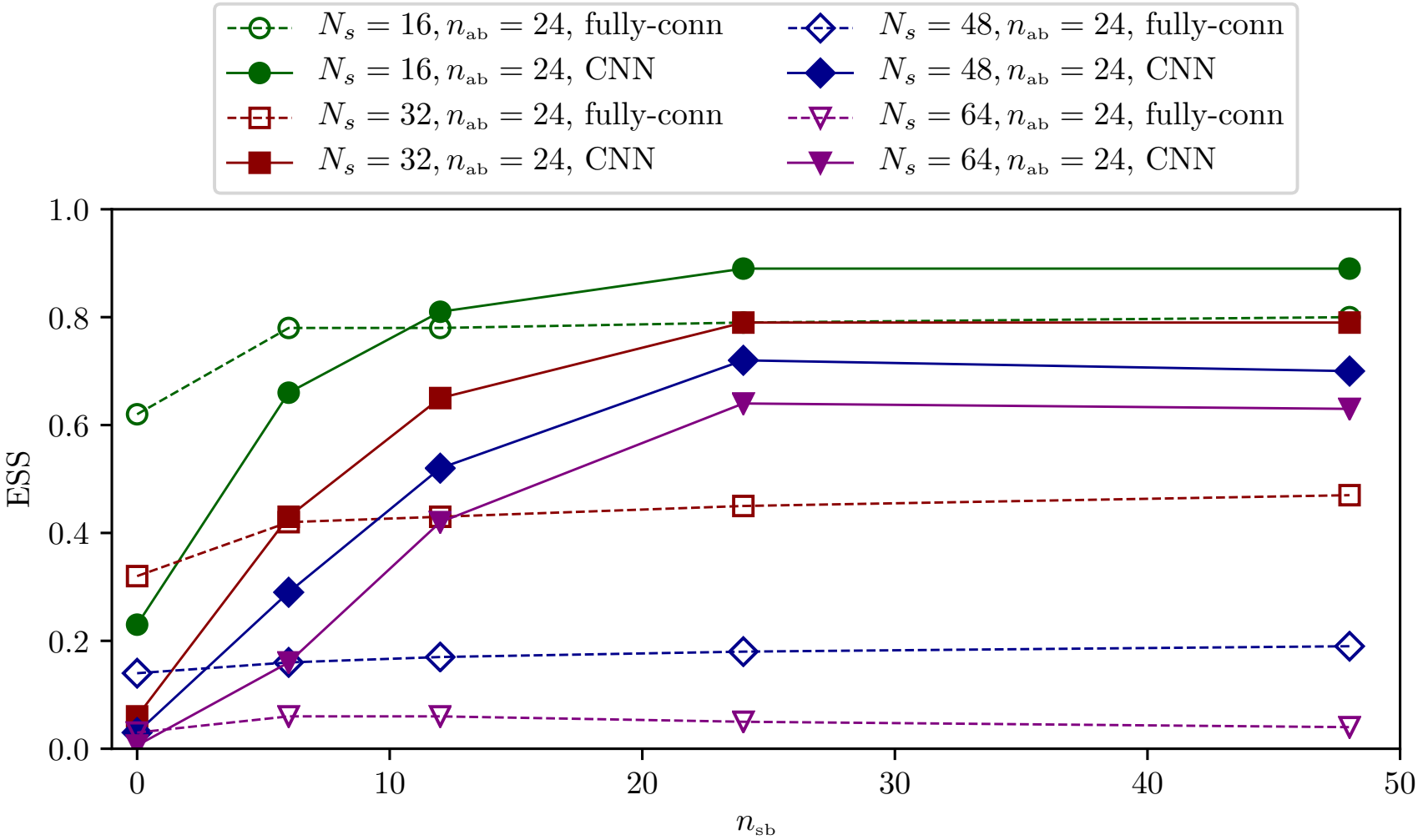


Overall computational cost difficult to assess

CNN vs fully-connected networks with $N_t \times N_s$ neurons, 16×8 lattice

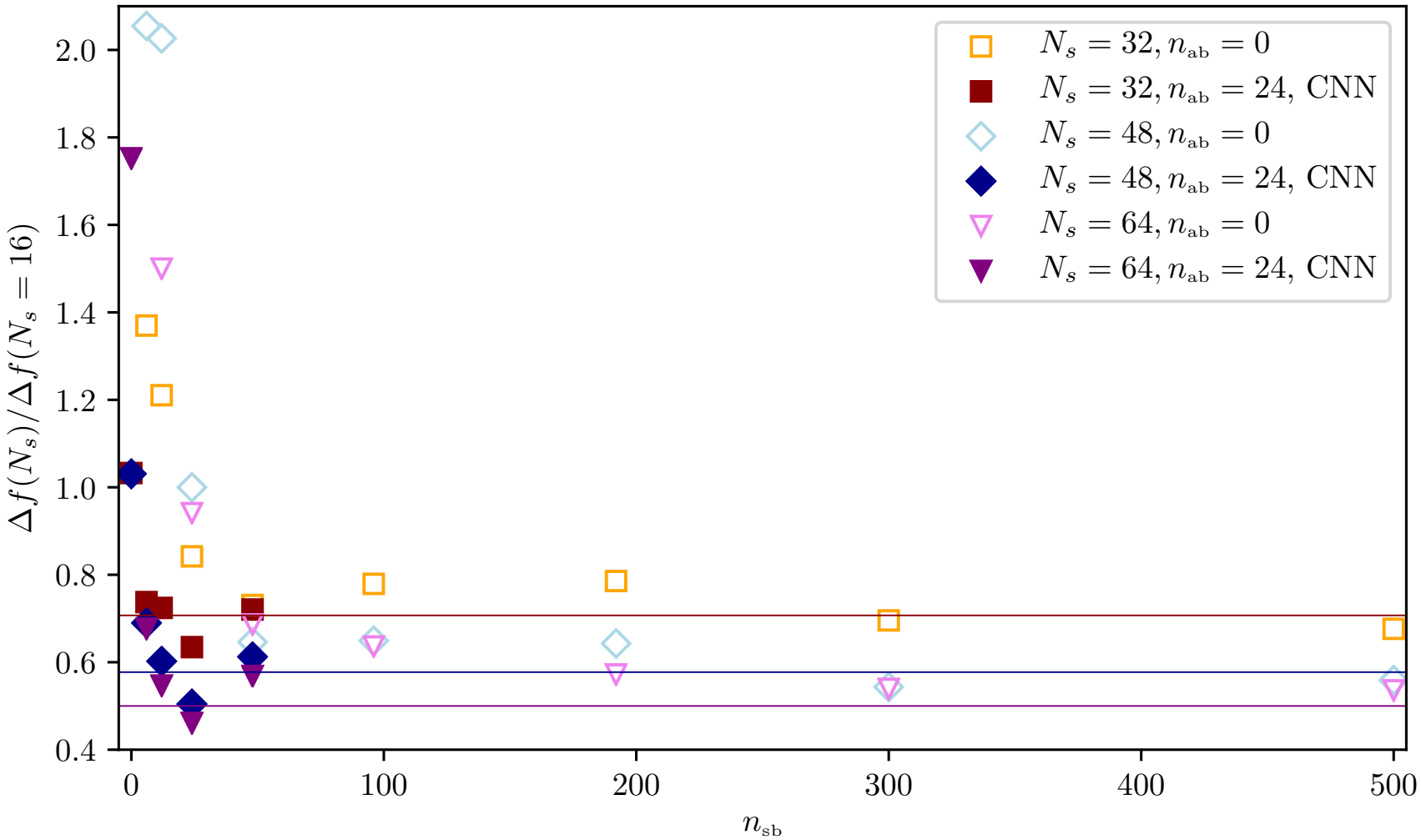


CNN vs fully-connected networks with $N_t \times N_s$ neurons, larger lattices



SNFs not necessarily convenient for any NFs: poor performance with fully-connected NNs

Error ratio


$N_s = 32, n_{\rm ab} = 0$
$N_s = 32, n_{\rm ab} = 24$, CNN
$N_s = 48, n_{\rm ab} = 0$
$N_s = 48, n_{\rm ab} = 24$, CNN
$N_s = 64, n_{\rm ab} = 0$
$N_s = 64, n_{\rm ab} = 24$, CNN
$\Delta f(N_s)/\Delta f(N_s = 16)$
$n_{\rm sb}$
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
100
200
300
400
500

The Second Law of Thermodynamics

We start from Clausius inequality

$$\int_A^B \frac{\mathrm{d}Q}{T} \leq \Delta S$$

that for isothermal transformations becomes

$$\frac{Q}{T} \leq \Delta S$$

If we use

$$\begin{cases} Q = & \Delta E - W \quad \text{(First Law)} \\ F \stackrel{\text{def}}{=} & E - ST \end{cases}$$

the Second Law becomes

$$W \geq \Delta F$$

where the equality holds for reversible processes.

Moving from thermodynamics to statistical mechanics we know that the former relation (valid for a *macroscopic* system) becomes

$$\langle W \rangle_f \geq \Delta F$$

JE and the Second Law

Starting from Jarzynski's equality

$$\left\langle \exp\left(-\frac{W}{T}\right)\right\rangle_f = \exp\left(-\frac{\Delta F}{T}\right)$$

and using *Jensen's inequality*

$$\langle \exp x\rangle \geq \exp\langle x\rangle$$

(valid for averages on real x) we get

$$\exp\left(-\frac{\Delta F}{T}\right) = \left\langle \exp\left(-\frac{W}{T}\right)\right\rangle_f \geq \exp\left(-\frac{\langle W\rangle_f}{T}\right)$$

from which we have

$$\langle W\rangle_f \geq \Delta F$$

In this sense Jarzynski's relation can be seen as a **generalization** of the Second Law.

Crooks fluctuation theorem

Crooks theorem **[Crooks; 1998]**: another relation deeply connected with Jarzynski's equality

$$\frac{\mathcal{P}_F(W)}{\mathcal{P}_R(-W)} = e^{(W-\Delta F)}$$

The $\mathcal{P}_{F,R}$ indicate the probability distribution of the work performed in the forward and reverse realizations of the transformation.

JE is easily recovered by moving the $\exp(-W)$ and $\mathcal{P}_R$ factors and integrating in W on both sides.

$W_d = W - \Delta F$ is the **dissipated** work.