

# MLMC: Machine Learning Monte Carlo

Sam Foreman

2023-07-31

# MLMC: Machine Learning Monte Carlo for Lattice Gauge Theory



Sam Foreman

Xiao-Yong Jin, James C. Osborn



[saforem2/lattice23](#)

# Overview

## 1. Background: {MCMC, HMC}

- Leapfrog Integrator
- Issues with HMC
- Can we do better?

## 2. L2HMC: Generalizing MD

- 4D  $SU(3)$  Model
- Results

## 3. References

## 4. Extras

# Background: MCMC

# Markov Chain Monte Carlo (MCMC)

## Goal

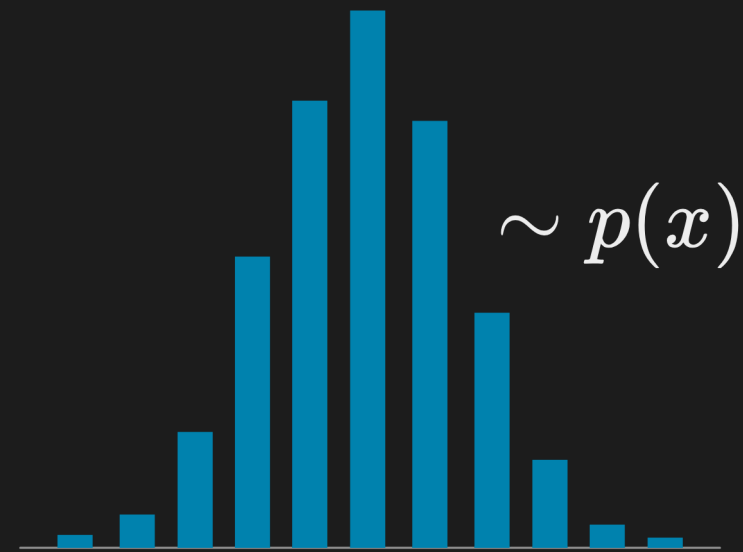
Generate **independent** samples  $\{x_i\}$ , such that<sup>1</sup>

$$\{x_i\} \sim p(x) \propto e^{-S(x)}$$

where  $S(x)$  is the *action* (or potential energy)

- Want to calculate observables  $\mathcal{O}$ :  

$$\langle \mathcal{O} \rangle \propto \int [\mathcal{D}x] \mathcal{O}(x) p(x)$$



If these were *independent*, we could approximate:  $\langle \mathcal{O} \rangle \simeq \frac{1}{N} \sum_{n=1}^N \mathcal{O}(x_n)$

$$\sigma_{\mathcal{O}}^2 = \frac{1}{N} \text{Var}[\mathcal{O}(x)] \implies \sigma_{\mathcal{O}} \propto \frac{1}{\sqrt{N}}$$

1. Here,  $\sim$  means “is distributed according to”

# Markov Chain Monte Carlo (MCMC)

## Goal

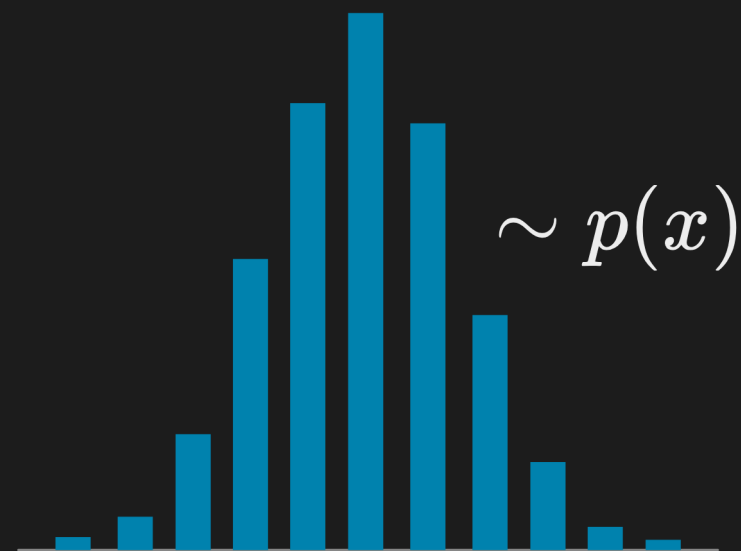
Generate **independent** samples  $\{x_i\}$ , such that<sup>1</sup>

$$\{x_i\} \sim p(x) \propto e^{-S(x)}$$

where  $S(x)$  is the *action* (or potential energy)

- Want to calculate observables  $\mathcal{O}$ :  

$$\langle \mathcal{O} \rangle \propto \int [\mathcal{D}x] \mathcal{O}(x) p(x)$$



Instead, nearby configs are **correlated**, and we incur a factor of  $\tau_{\text{int}}^{\mathcal{O}}$ :

$$\sigma_{\mathcal{O}}^2 = \frac{\tau_{\text{int}}^{\mathcal{O}}}{N} \text{Var}[\mathcal{O}(x)]$$

1. Here,  $\sim$  means “is distributed according to”

# Background: HMC

# Hamiltonian Monte Carlo (HMC)

- Want to (sequentially) construct a chain of states:

$$x_0 \rightarrow x_1 \rightarrow x_i \rightarrow \cdots \rightarrow x_N$$

such that, as  $N \rightarrow \infty$ :

$$\{x_i, x_{i+1}, x_{i+2}, \cdots, x_N\} \xrightarrow{N \rightarrow \infty} p(x) \propto e^{-S(x)}$$

## </> Trick

- Introduce **fictitious** momentum  $v \sim \mathcal{N}(0, 1)$ 
  - Normally distributed **independent** of  $x$ , i.e.

$$p(x, v) = p(x) p(v) \propto e^{-S(x)} e^{-\frac{1}{2}v^T v} = e^{-[S(x) + \frac{1}{2}v^T v]} = e^{-H(x, v)}$$



# Hamiltonian Monte Carlo (HMC)

- **Idea:** Evolve the  $(\dot{x}, \dot{v})$  system to get new states  $\{x_i\}$  !
- Write the **joint distribution**  $p(x, v)$ :

$$p(x, v) \propto e^{-S[x]} e^{-\frac{1}{2}v^T v} = e^{-H(x, v)}$$

## </> Hamiltonian Dynamics

$$H = S[x] + \frac{1}{2}v^T v \implies$$

$$\dot{x} = +\partial_v H, \quad \dot{v} = -\partial_x H$$

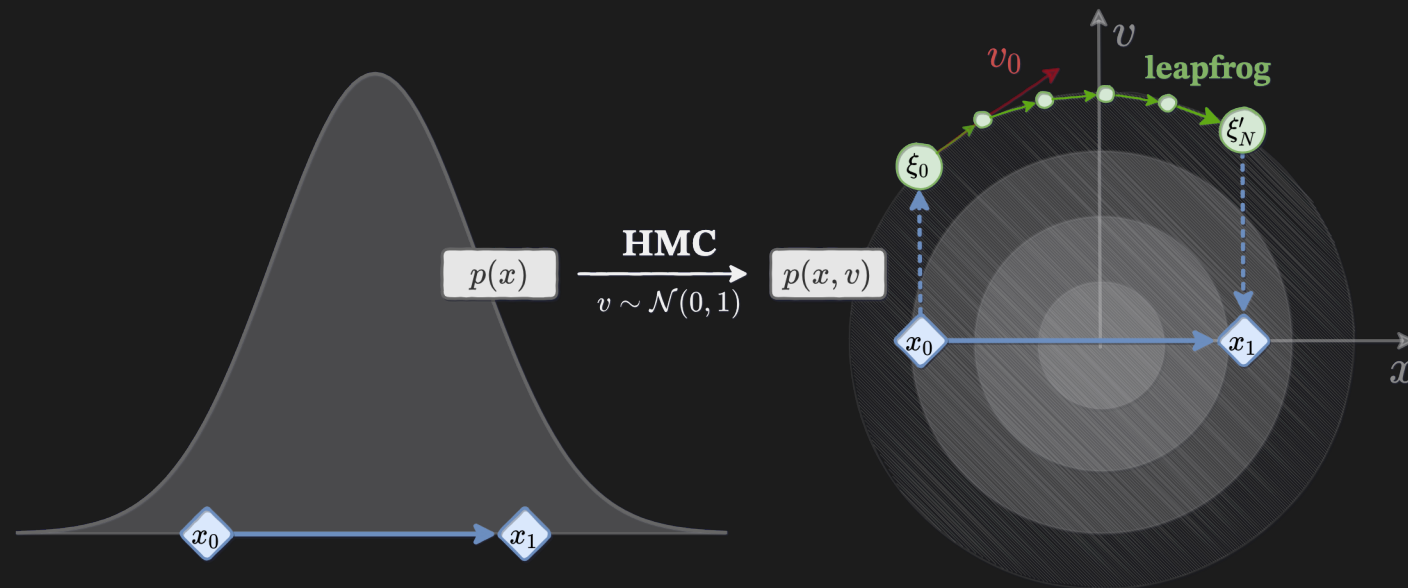


Figure 1: Overview of HMC algorithm

# Leapfrog Integrator (HMC)

## </> Hamiltonian Dynamics

$$(\dot{x}, \dot{v}) = (\partial_v H, -\partial_x H)$$

## ⌚ Leapfrog Step

input  $(x, v) \rightarrow (x', v')$  output

$$\tilde{v} := \Gamma(x, v) = v - \frac{\varepsilon}{2} \partial_x S(x)$$

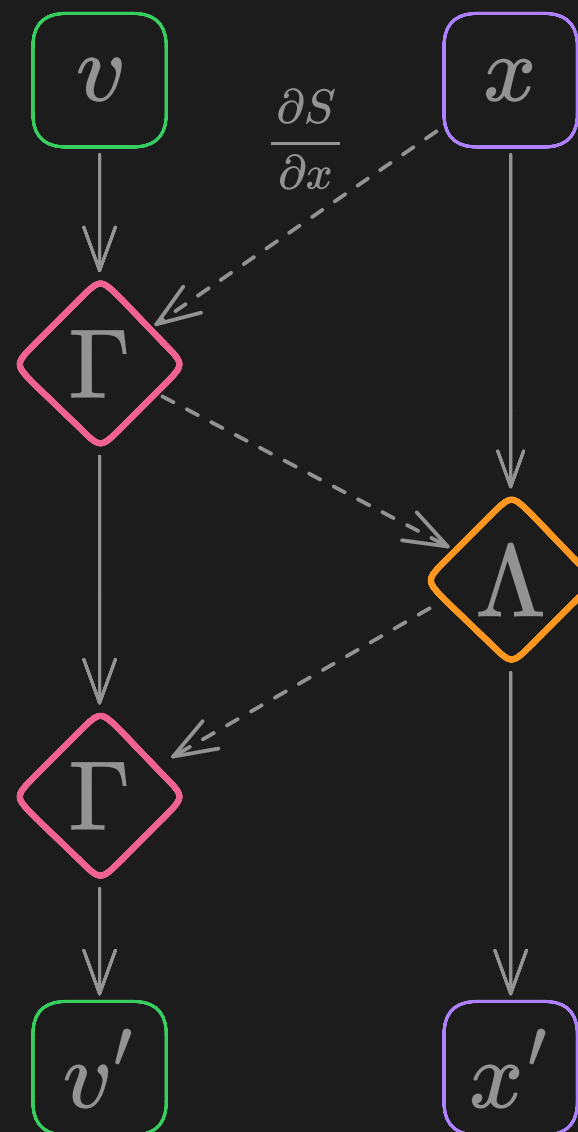
$$x' := \Lambda(x, \tilde{v}) = x + \varepsilon \tilde{v}$$

$$v' := \Gamma(x', \tilde{v}) = \tilde{v} - \frac{\varepsilon}{2} \partial_x S(x')$$

## 🚨 Warning!

Resample  $v_0 \sim \mathcal{N}(0, 1)$   
at the **beginning** of each trajectory

**Note:**  $\partial_x S(x)$  is the *force*



# HMC Update

- We build a trajectory of  $N_{\text{LF}}$  **leapfrog steps**<sup>1</sup>

$$(x_0, v_0) \rightarrow (x_1, v_1) \rightarrow \dots \rightarrow (x', v')$$

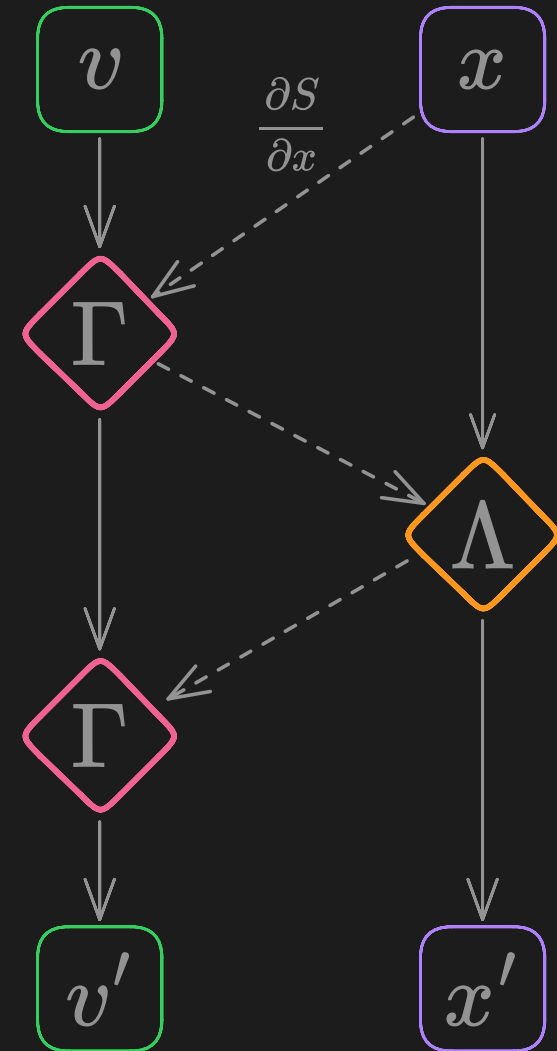
- And propose  $x'$  as the next state in our chain

$$\Gamma : (x, v) \rightarrow v' := v - \frac{\varepsilon}{2} \partial_x S(x)$$

$$\Lambda : (x, v) \rightarrow x' := x + \varepsilon v$$

- We then accept / reject  $x'$  using Metropolis-Hastings criteria,

$$A(x'|x) = \min \left\{ 1, \frac{p(x')}{p(x)} \left| \frac{\partial x'}{\partial x} \right| \right\}$$

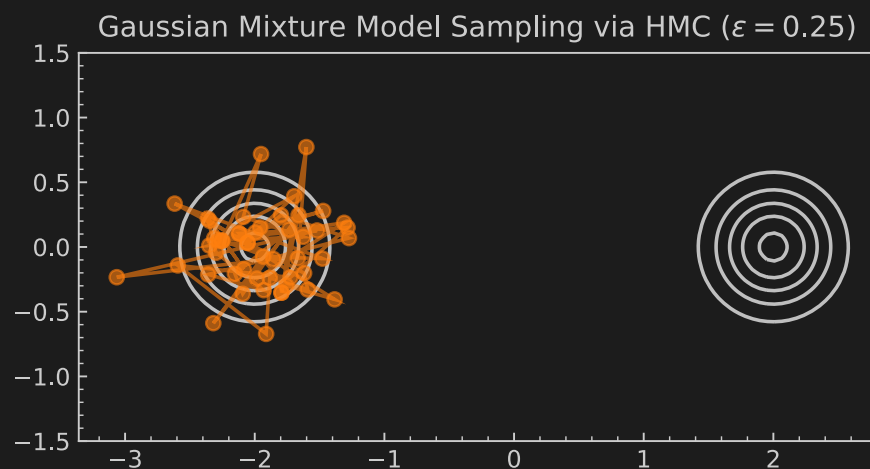


1. We **always** start by resampling the momentum,  $v_0 \sim \mathcal{N}(0, 1)$

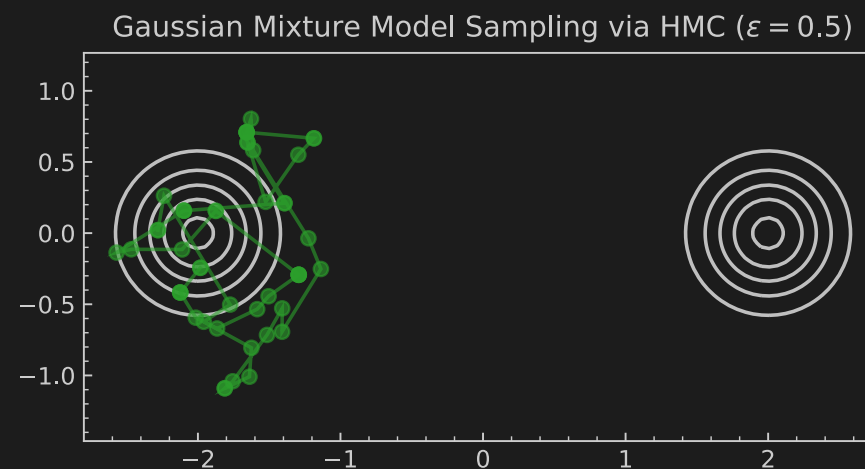
# HMC Demo

Figure 2: HMC Demo

- What do we want in a good sampler?
  - **Fast mixing** (small autocorrelations)
  - **Fast burn-in** (quick convergence)
- Problems with HMC:
  - Energy levels selected randomly → **slow mixing**
  - Cannot easily traverse low-density zones → **slow convergence**



*HMC Samples with  $\epsilon = 0.25$*



*HMC Samples with  $\epsilon = 0.5$*

Figure 3: HMC Samples generated with varying step sizes  $\epsilon$

# Topological Freezing

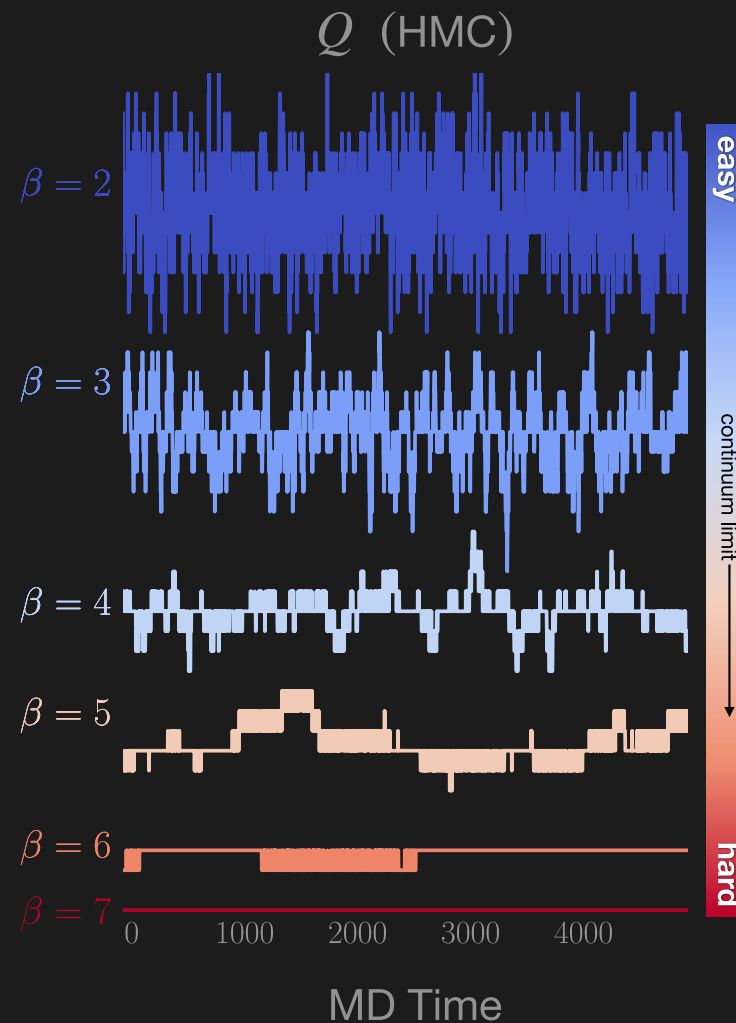
## Topological Charge:

$$Q = \frac{1}{2\pi} \sum_P [x_P] \in \mathbb{Z}$$

**note:**  $[x_P] = x_P - 2\pi \left\lfloor \frac{x_P + \pi}{2\pi} \right\rfloor$

### 🔥 Critical Slowing Down

- $Q$  gets stuck!
  - as  $\beta \rightarrow \infty$ :
    - $Q \rightarrow \text{const.}$
    - $\delta Q = (Q^* - Q) \rightarrow 0 \implies$
  - # configs required to estimate errors **grows exponentially:**  $\tau_{\text{int}}^Q \rightarrow \infty$



Note  $\delta Q \rightarrow 0$  at increasing  $\beta$

# Can we do better?

- Introduce two (**invertible NNs**) **vNet** and **xNet**<sup>1</sup>:
  - **vNet**:  $(x, F) \longrightarrow (s_v, t_v, q_v)$
  - **xNet**:  $(x, v) \longrightarrow (s_x, t_x, q_x)$
- Use these  $(s, t, q)$  in the *generalized* MD update:
  - $\Gamma_{\theta}^{\pm} : (x, v) \xrightarrow{s_v, t_v, q_v} (x, v')$
  - $\Lambda_{\theta}^{\pm} : (x, v) \xrightarrow{s_x, t_x, q_x} (x', v)$

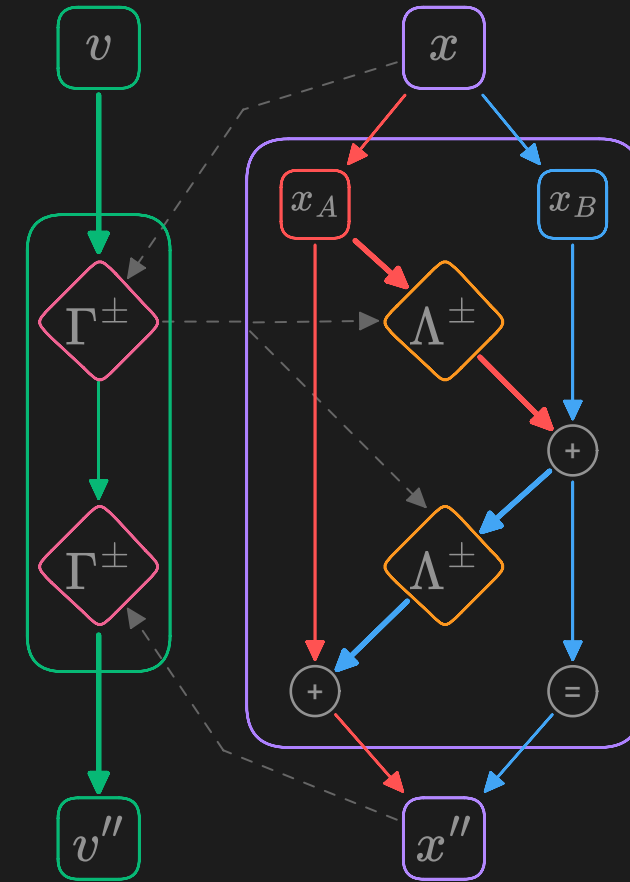



Figure 4: Generalized MD update where  $\Lambda_{\theta}^{\pm}, \Gamma_{\theta}^{\pm}$  are **invertible NNs**

1. L2HMC:  (Foreman, Jin, and Osborn 2021, 2022)

# L2HMC: Generalizing the MD Update

## L2HMC Update

- Introduce  $d \sim \mathcal{U}(\pm)$  to determine the direction<sup>1</sup> of our update

- $v' = \Gamma^\pm(x, v)$  update  $v$
- $x' = x_B + \Lambda^\pm(x_A, v')$  update first **half**:  $x_A$
- $x'' = x'_A + \Lambda^\pm(x'_B, v')$  update other half:  $x_B$
- $v'' = \Gamma^\pm(x'', v')$  update  $v$

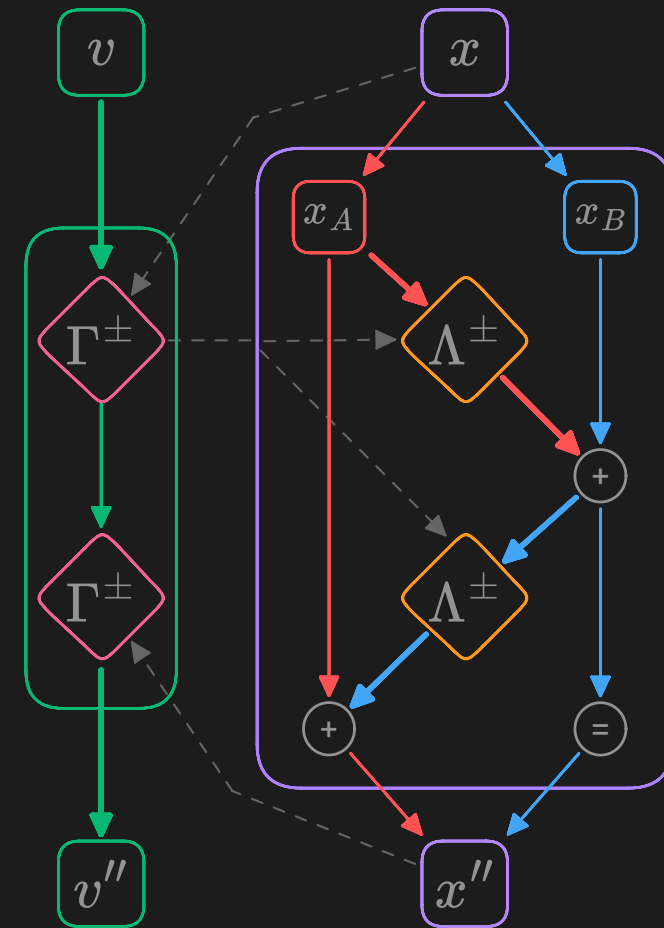


Figure 5: Generalized MD update with  $\Lambda_\theta^\pm, \Gamma_\theta^\pm$   
invertible NNs

- $d$  resampled at start of each trajectory, to ensure reversibility



# L2HMC: Leapfrog Layer

1. Update  $\mathbf{v}$ :

$$\mathbf{v}' = \Gamma^\pm[\mathbf{v}; \zeta_{\mathbf{v}}]$$

2. Update half of  $\mathbf{x}$  via  $\bar{\mathbf{m}}_k \odot \mathbf{x}_k$ :

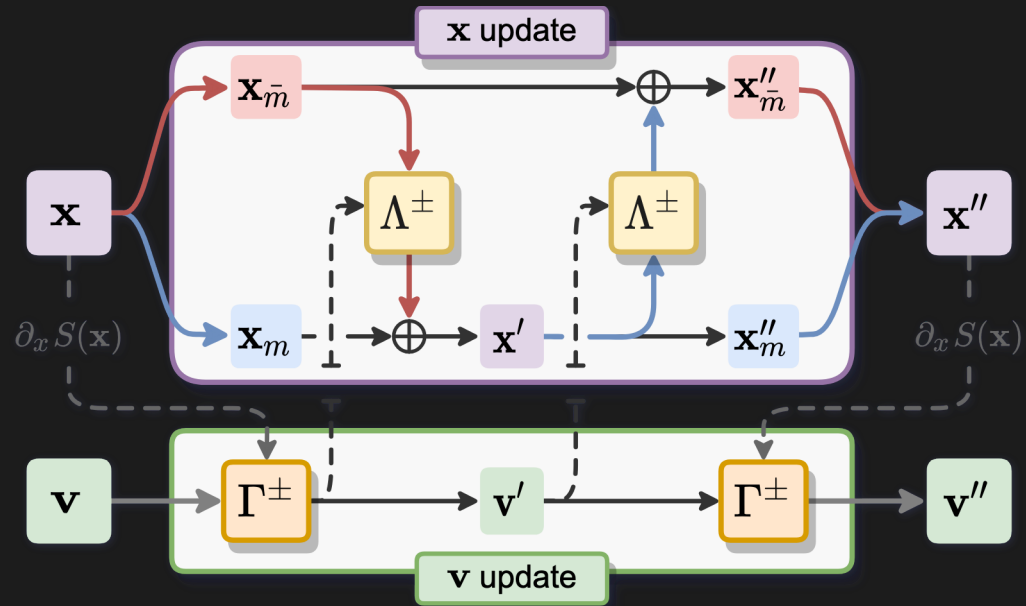
$$\mathbf{x}' = \mathbf{x}_m + \bar{\mathbf{m}} \odot \Lambda^\pm[\mathbf{x}_{\bar{\mathbf{m}}}; \zeta_{\bar{\mathbf{x}}_k}]$$

3. Update (other) half via  $\mathbf{m}^k \odot \mathbf{x}'_k$ :

$$\mathbf{x}'' = \mathbf{x}'_m + \bar{\mathbf{m}} \odot \Lambda^\pm[\mathbf{x}'_m; \zeta_{\mathbf{x}'}]$$

4. Half-step full  $\mathbf{v}$  update:

$$\mathbf{v}'' = \Gamma^\pm[\mathbf{v}'; \zeta_{\mathbf{v}'}]$$



$$\Gamma^+[\mathbf{v}_k; \zeta_{\mathbf{v}}] \equiv \mathbf{v}_k \odot \exp\left(\frac{\varepsilon_{\mathbf{v}}^k}{2} s_{\mathbf{v}}^k(\zeta_{\mathbf{v}_k})\right) - \frac{\varepsilon_{\mathbf{v}}^k}{2} \left[ \partial_x S(x_k) \odot \exp(\varepsilon_{\mathbf{v}}^k q_{\mathbf{v}}^k(\zeta_{\mathbf{v}_k})) + t_{\mathbf{v}}^k(\zeta_{\mathbf{v}_k}) \right]$$

trainable step sizes

$$\Lambda^+[\bar{\mathbf{x}}_k; \zeta_{\bar{\mathbf{x}}_k}] \equiv \bar{\mathbf{x}}_k \odot \exp(\varepsilon_{\bar{\mathbf{x}}}^k s_{\bar{\mathbf{x}}}^k(\zeta_{\bar{\mathbf{x}}_k})) + \varepsilon_{\bar{\mathbf{x}}}^k \left[ v'_k \odot \exp(\varepsilon_{\bar{\mathbf{x}}}^k q_{\bar{\mathbf{x}}}^k(\zeta_{\bar{\mathbf{x}}_k})) + t_{\bar{\mathbf{x}}}^k(\zeta_{\bar{\mathbf{x}}_k}) \right]$$

## Algorithm

1. **input:**  $x$

- Resample:  $v \sim \mathcal{N}(0, 1)$ ;  $d \sim \mathcal{U}(\pm)$
- Construct initial state:  $\xi = (x, v, \pm)$

2. **forward:** Generate proposal  $\xi'$  by passing initial  $\xi$  through  $N_{\text{LF}}$  leapfrog layers

$$\xi \xrightarrow{\text{LF layer}} \xi_1 \longrightarrow \cdots \longrightarrow \xi_{N_{\text{LF}}} = \xi' := (x'', v'')$$

- Accept / Reject:

$$A(\xi'|\xi) = \min \left\{ 1, \frac{\pi(\xi')}{\pi(\xi)} |\mathcal{J}(\xi', \xi)| \right\}$$

3. **backward** (if training):

- Evaluate the **loss function**<sup>1</sup>  $\mathcal{L} \leftarrow \mathcal{L}_\theta(\xi', \xi)$  and backprop

4. **return:**  $x_{i+1}$

Evaluate MH criteria (1) and return accepted config,

$$x_{i+1} \leftarrow \begin{cases} x'' & \text{w/ prob } A(\xi''|\xi) \quad \checkmark \\ x & \text{w/ prob } 1 - A(\xi''|\xi) \quad \otimes \end{cases}$$

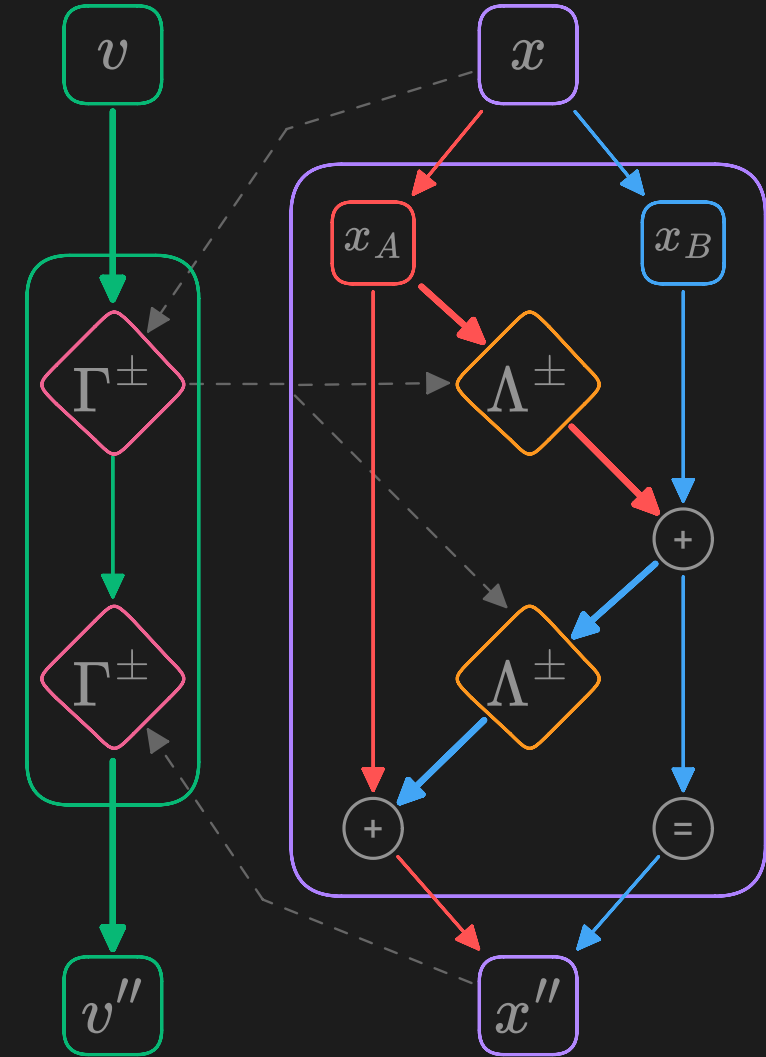


Figure 6: **Leapfrog Layer** used in generalized MD update

1. For simple  $\mathbf{x} \in \mathbb{R}^2$  example,  $\mathcal{L}_\theta = A(\xi^*|\xi) \cdot (\mathbf{x}^* - \mathbf{x})^2$

# 4D $SU(3)$ Model

## 🌀 Link Variables

- Write link variables  $U_\mu(x) \in SU(3)$ :

$$\begin{aligned} U_\mu(x) &= \exp \left[ i \omega_\mu^k(x) \lambda^k \right] \\ &= e^{iQ}, \quad \text{with } Q \in \mathfrak{su}(3) \end{aligned}$$

where  $\omega_\mu^k(x) \in \mathbb{R}$ , and  $\lambda^k$  are the generators of  $SU(3)$

## </> Conjugate Momenta

- Introduce  $P_\mu(x) = P_\mu^k(x) \lambda^k$  conjugate to  $\omega_\mu^k(x)$

## 🔥 Wilson Action

$$S_G = -\frac{\beta}{6} \sum \text{Tr} [U_{\mu\nu}(x) + U_{\mu\nu}^\dagger(x)]$$

where  $U_{\mu\nu}(x) = U_\mu(x) U_\nu(x + \hat{\mu}) U_\mu^\dagger(x + \hat{\nu}) U_\nu^\dagger(x)$

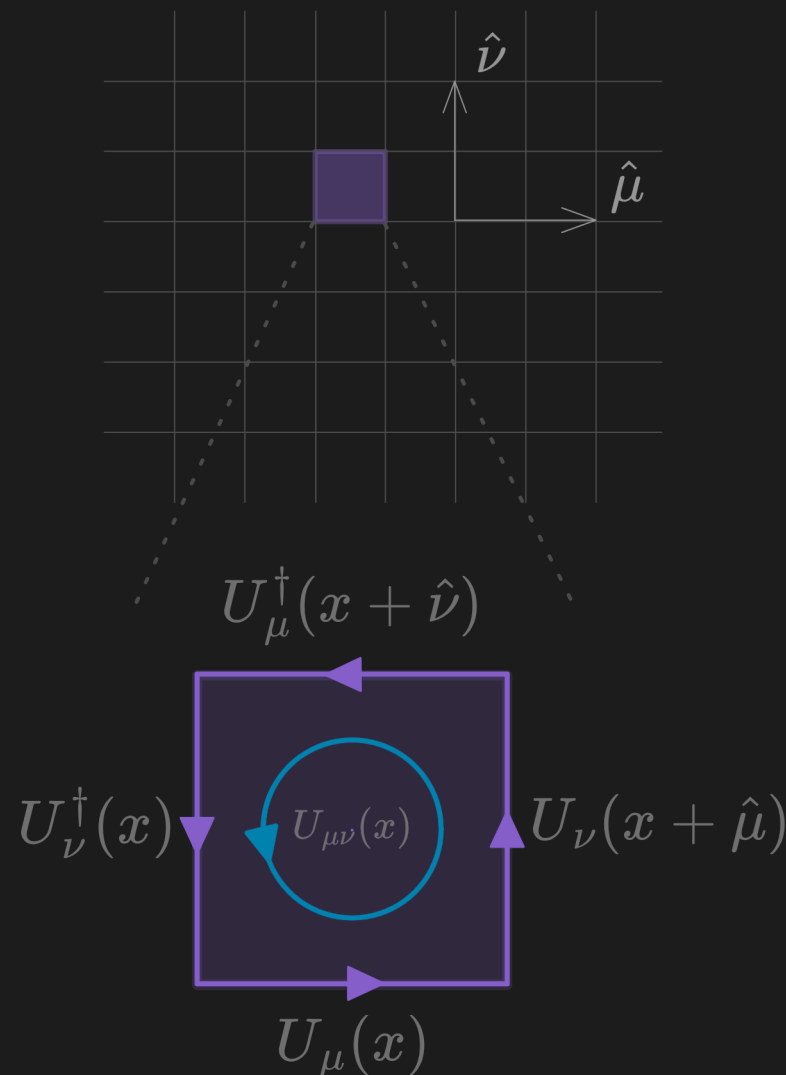


Figure 7: Illustration of the lattice



# HMC: 4D $SU(3)$

Hamiltonian:  $H[P, U] = \frac{1}{2}P^2 + S[U] \implies$

- $U$  update:  $\frac{d\omega^k}{dt} = \frac{\partial H}{\partial P^k}$

$$\frac{d\omega^k}{dt} \lambda^k = P^k \lambda^k \implies \frac{dQ}{dt} = P$$

$$Q(\epsilon) = Q(0) + \epsilon P(0) \implies$$

$$-i \log U(\epsilon) = -i \log U(0) + \epsilon P(0)$$

$$U(\epsilon) = e^{i\epsilon P(0)} U(0) \implies$$

$$\Lambda : U \longrightarrow U' := e^{i\epsilon P'} U$$

$\epsilon$  is the step size

- $P$  update:  $\frac{dP^k}{dt} = -\frac{\partial H}{\partial \omega^k}$

$$\frac{dP^k}{dt} = -\frac{\partial H}{\partial \omega^k} = -\frac{\partial H}{\partial Q} = -\frac{dS}{dQ} \implies$$

$$P(\epsilon) = P(0) - \epsilon \left. \frac{dS}{dQ} \right|_{t=0}$$

$$= P(0) - \epsilon F[U]$$

$$\Gamma : P \longrightarrow P' := P - \frac{\epsilon}{2} F[U]$$

$F[U]$  is the force term

# HMC: 4D $SU(3)$

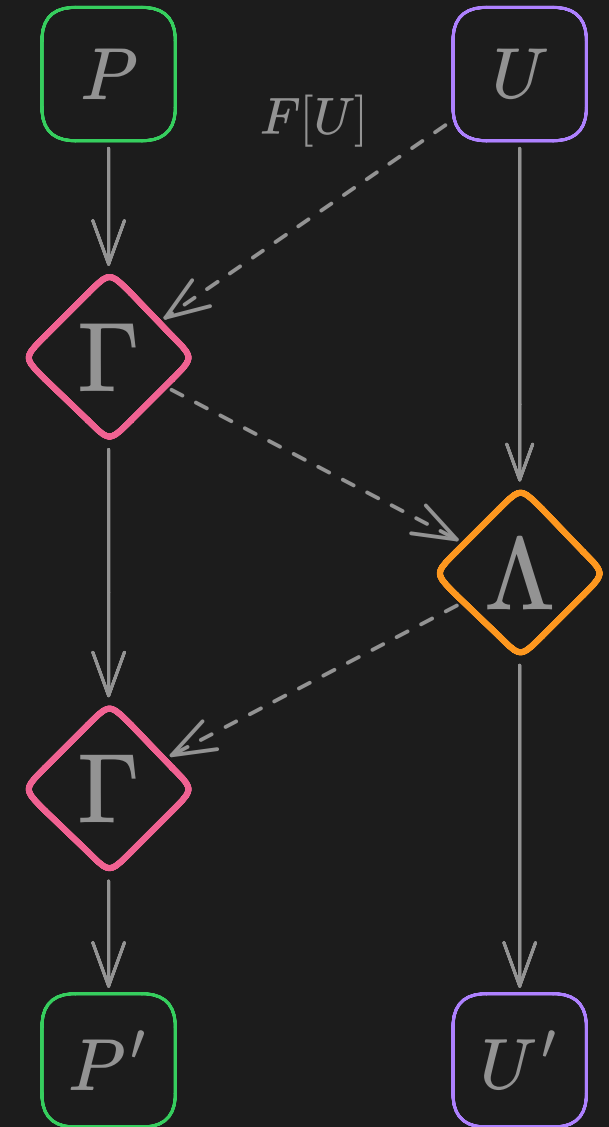
- Momentum Update:

$$\Gamma : P \longrightarrow P' := P - \frac{\varepsilon}{2} F[U]$$

- Link Update:

$$\Lambda : U \longrightarrow U' := e^{i\varepsilon P'} U$$

- We maintain a batch of  $N_b$  lattices, all updated in parallel
  - $U.dtype = complex128$
  - $U.shape = [N_b, 4, N_t, N_x, N_y, N_z, 3, 3]$



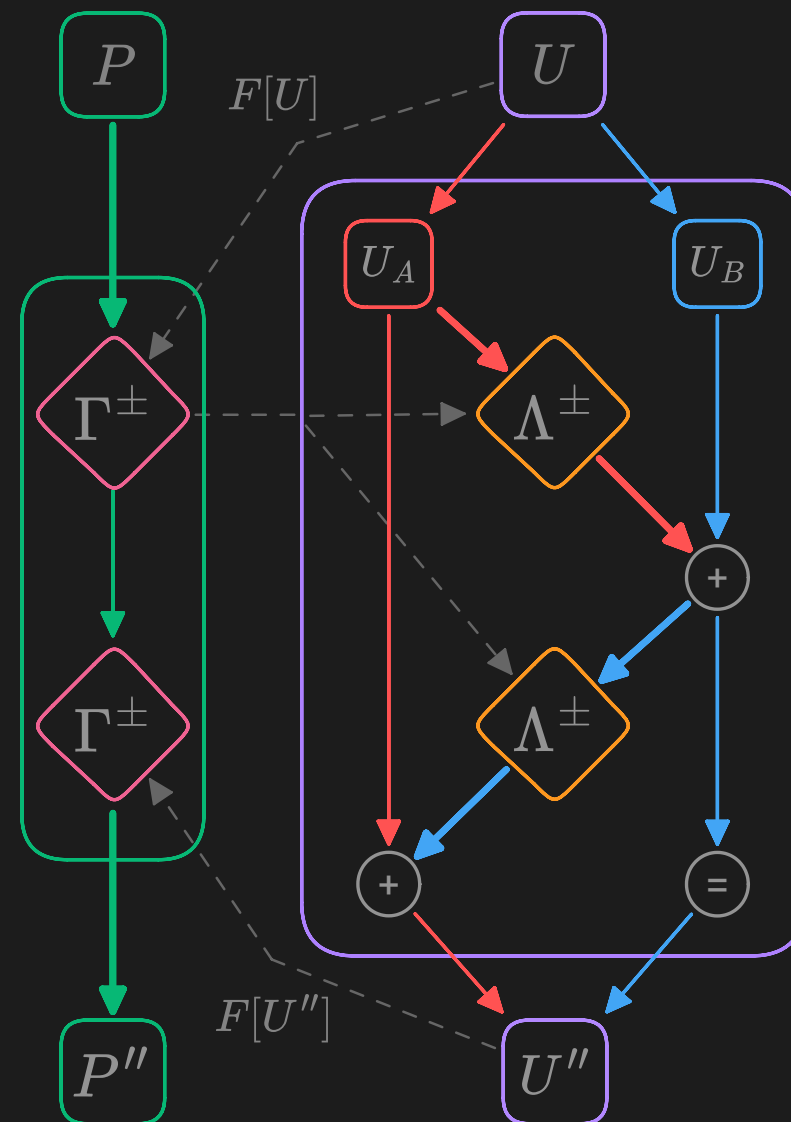
# Networks 4D $SU(3)$

$U$ -Network:

$$\text{UNet: } (U, P) \longrightarrow (s_U, t_U, q_U)$$

$P$ -Network:

$$\text{PNet: } (U, P) \longrightarrow (s_P, t_P, q_P)$$



# Networks 4D $SU(3)$

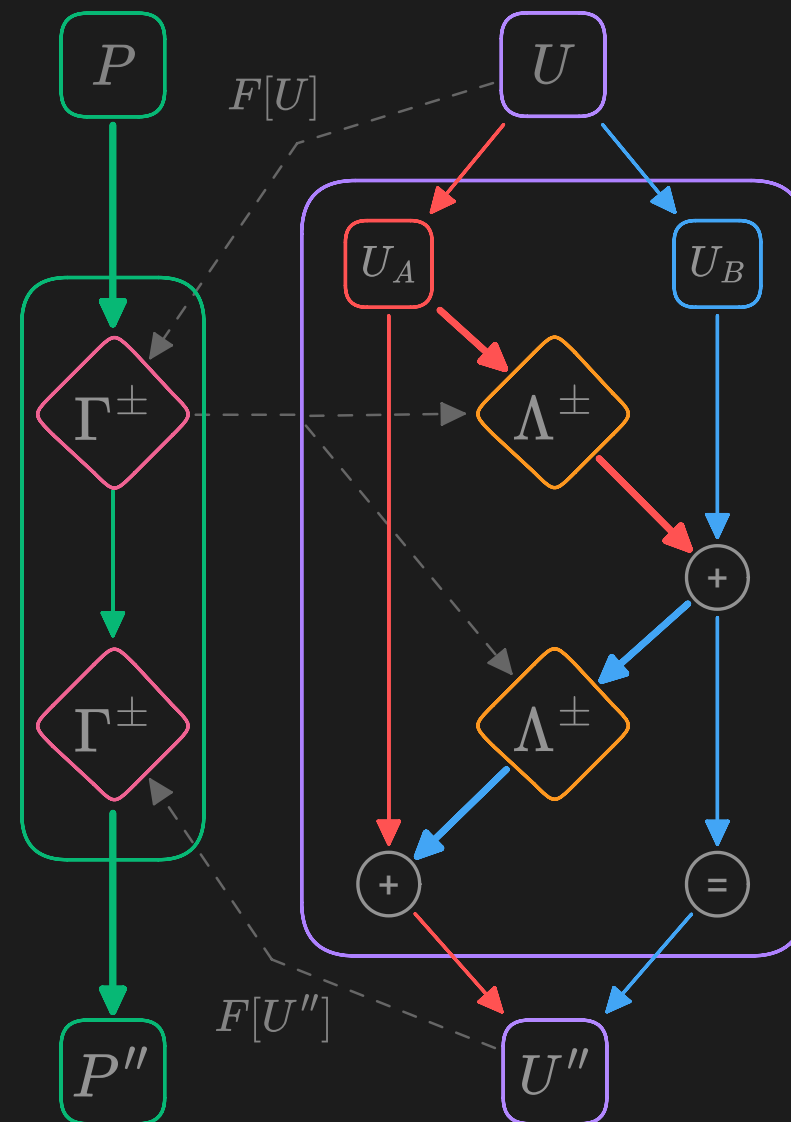
$U$ -Network:

$$\text{UNet: } (U, P) \longrightarrow (s_U, t_U, q_U)$$

$P$ -Network:

$$\text{PNet: } (U, P) \longrightarrow (s_P, t_P, q_P)$$

↑  
let's look at this





# $P$ -Network (pt. 1)



- input<sup>1</sup>:  $(U, F) := (e^{iQ}, F)$

$$h_0 = \sigma(w_Q Q + w_F F + b)$$

$$h_1 = \sigma(w_1 h_0 + b_1)$$

$$\vdots$$

$$h_n = \sigma(w_{n-1} h_{n-2} + b_n)$$

$$z := \sigma(w_n h_{n-1} + b_n) \longrightarrow$$

- output<sup>2</sup>:  $(s_P, t_P, q_P)$

$$\blacksquare s_P = \lambda_s \tanh(w_s z + b_s)$$

$$\blacksquare t_P = w_t z + b_t$$

$$\blacksquare q_P = \lambda_q \tanh(w_q z + b_q)$$

1.  $\sigma(\cdot)$  denotes an activation function

2.  $\lambda_s, \lambda_q \in \mathbb{R}$  are trainable parameters

## $P$ -Network (pt. 2)



- Use  $(s_P, t_P, q_P)$  to update  $\Gamma^\pm : (U, P) \rightarrow (U, P_\pm)^1$ :

- **forward** ( $d = +$ ):

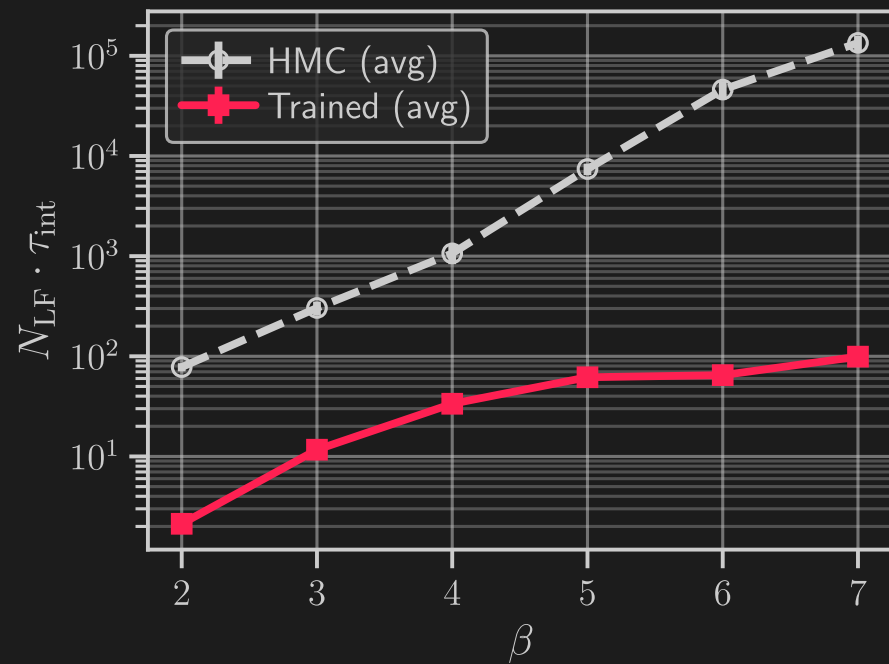
$$\Gamma^+(U, P) := P_+ = P \cdot e^{\frac{\varepsilon}{2}s_P} - \frac{\varepsilon}{2} [F \cdot e^{\varepsilon q_P} + t_P]$$

- **backward** ( $d = -$ ):

$$\Gamma^-(U, P) := P_- = e^{-\frac{\varepsilon}{2}s_P} \left\{ P + \frac{\varepsilon}{2} [F \cdot e^{\varepsilon q_P} + t_P] \right\}$$

1. Note that  $(\Gamma^+)^{-1} = \Gamma^-$ , i.e.  $\Gamma^+ [\Gamma^-(U, P)] = \Gamma^- [\Gamma^+(U, P)] = (U, P)$

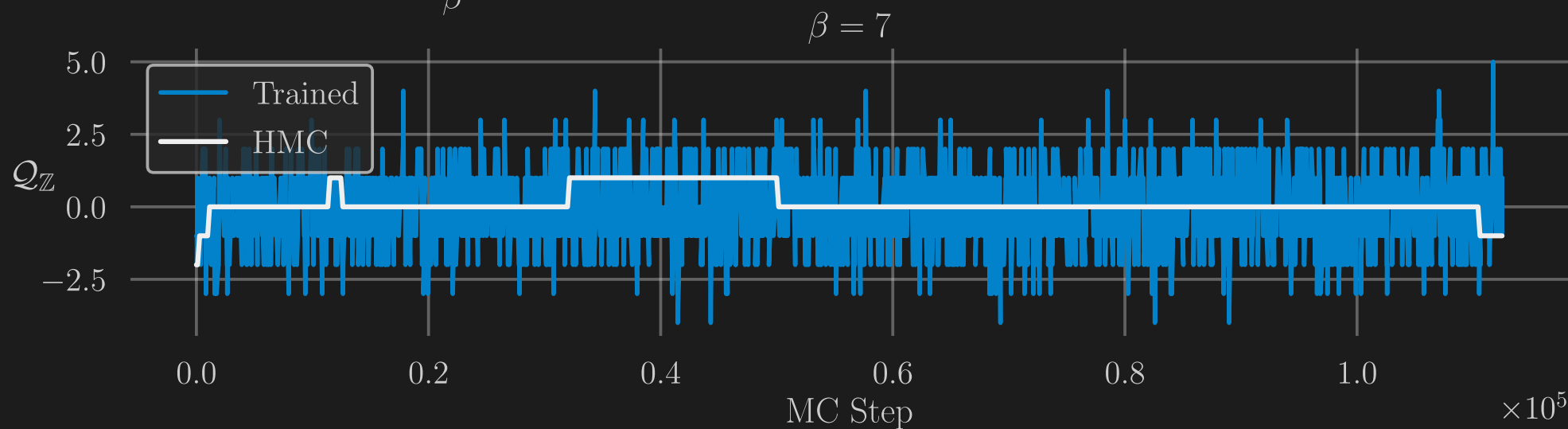
# Results: 2D $U(1)$



## Improvement

We can measure the performance by comparing  $\tau_{int}$  for the **trained model** vs. **HMC**.

**Note:** lower is better



# Interpretation

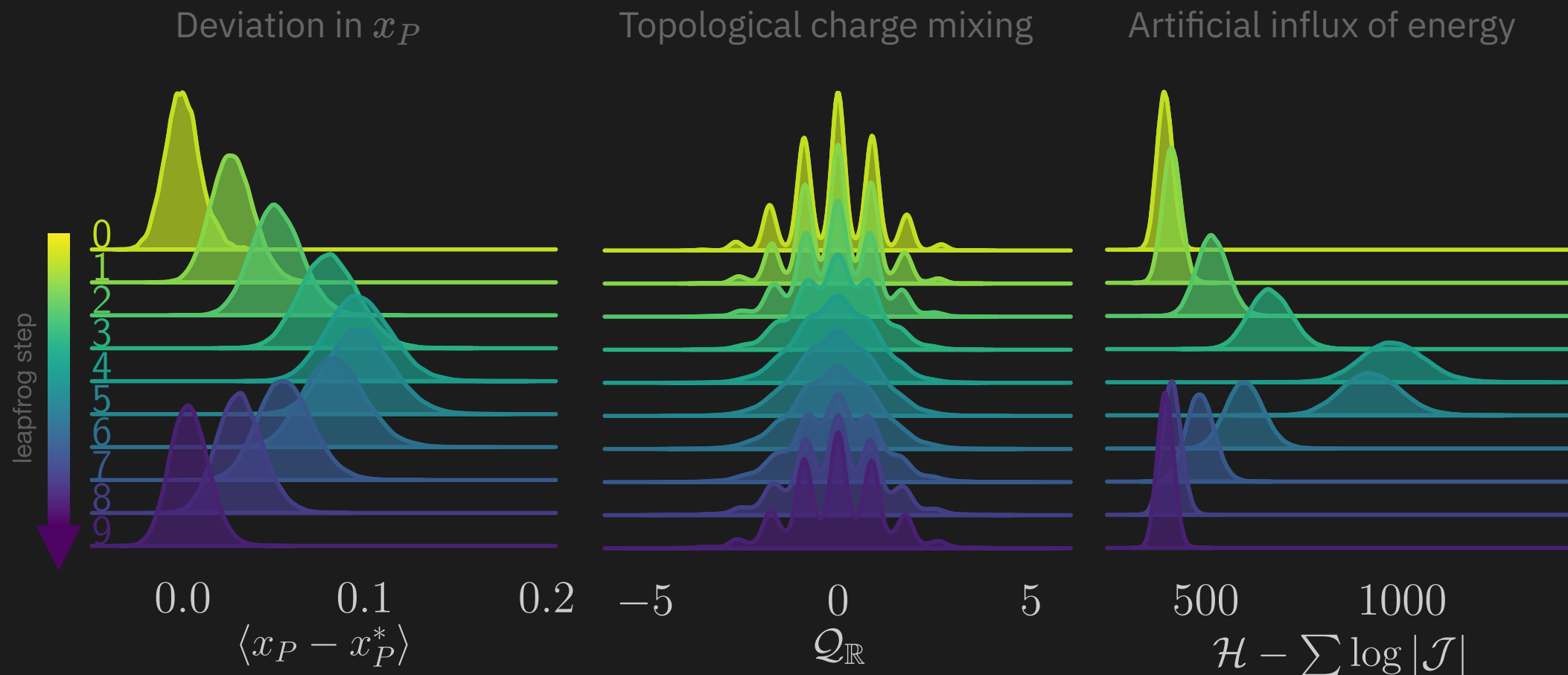
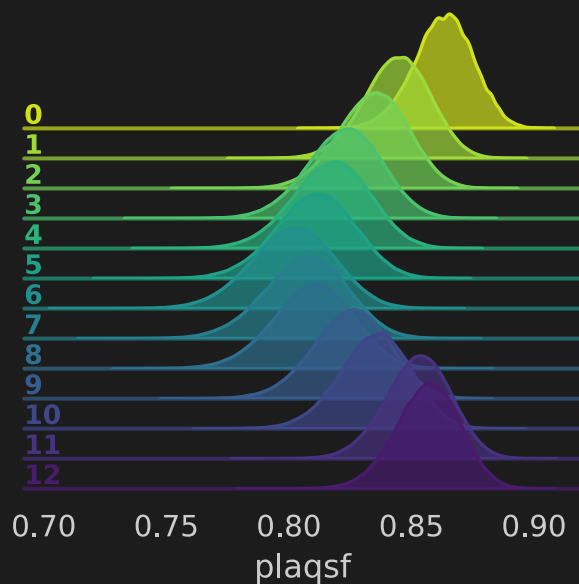
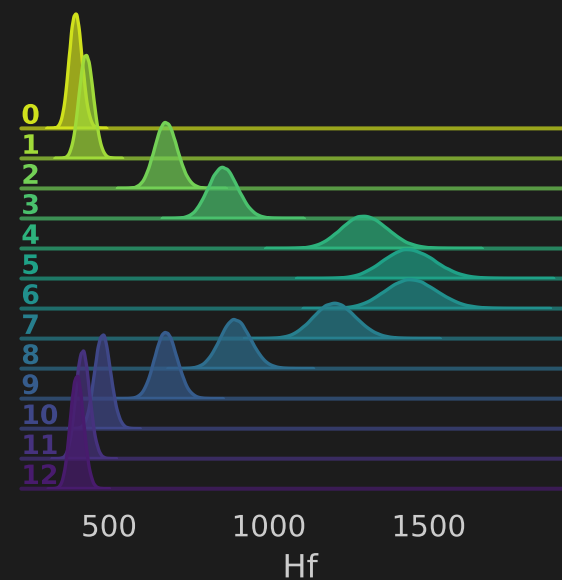


Figure 8: Illustration of how different observables evolve over a single L2HMC trajectory.

# Interpretation



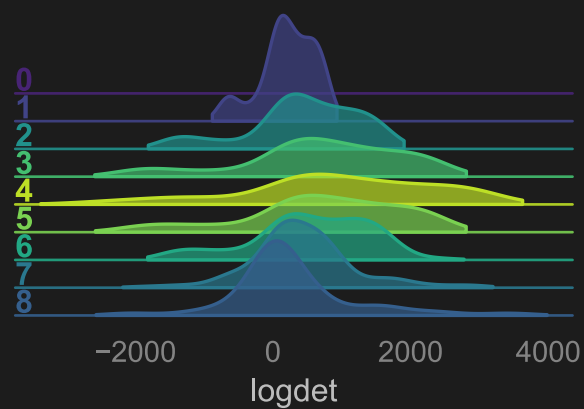
Average plaquette:  $\langle x_P \rangle$  vs LF step



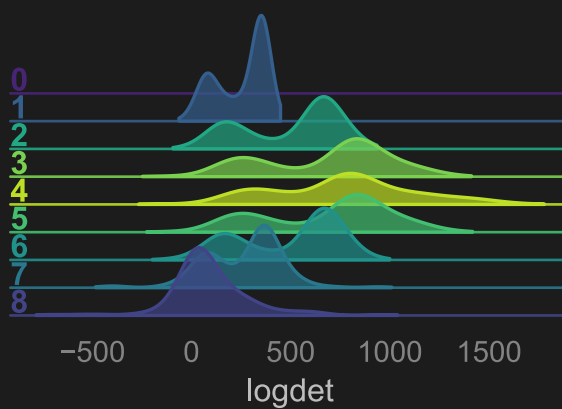
Average energy:  $H - \sum \log |\mathcal{J}|$

Figure 9: The trained model artificially increases the energy towards the middle of the trajectory, allowing the sampler to tunnel between isolated sectors.

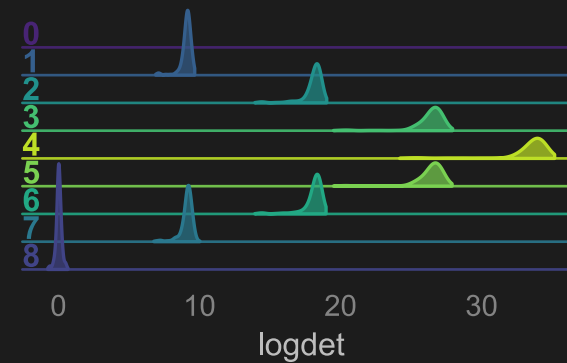
# 4D $SU(3)$ Results



(a) 100 train iters



(b) 500 train iters



(c) 1000 train iters

Figure 10:  $\log|\mathcal{J}|$  vs.  $N_{LF}$  during training

# 4D $SU(3)$ Results: $\delta U_{\mu\nu}$

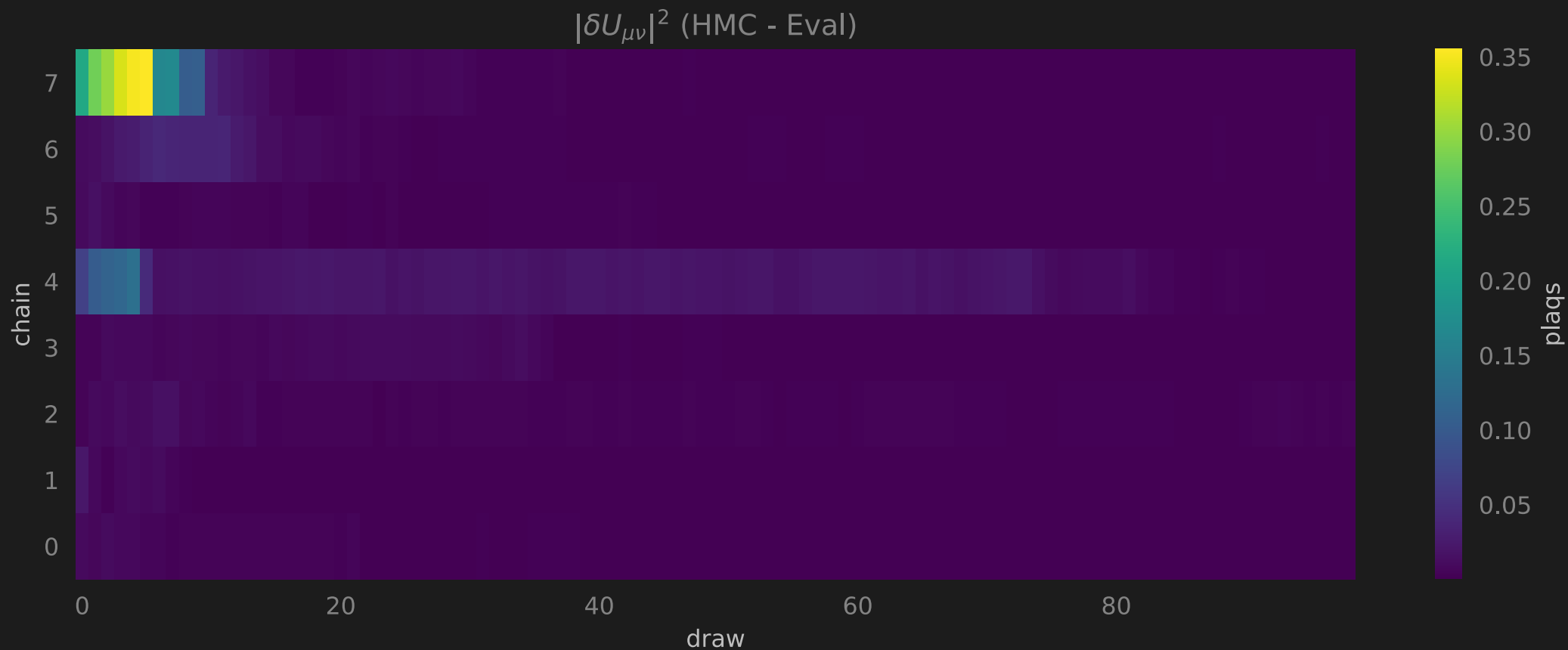


Figure 11: The difference in the average plaquette  $|\delta U_{\mu\nu}|^2$  between the trained model and HMC

# 4D $SU(3)$ Results: $\delta U_{\mu\nu}$

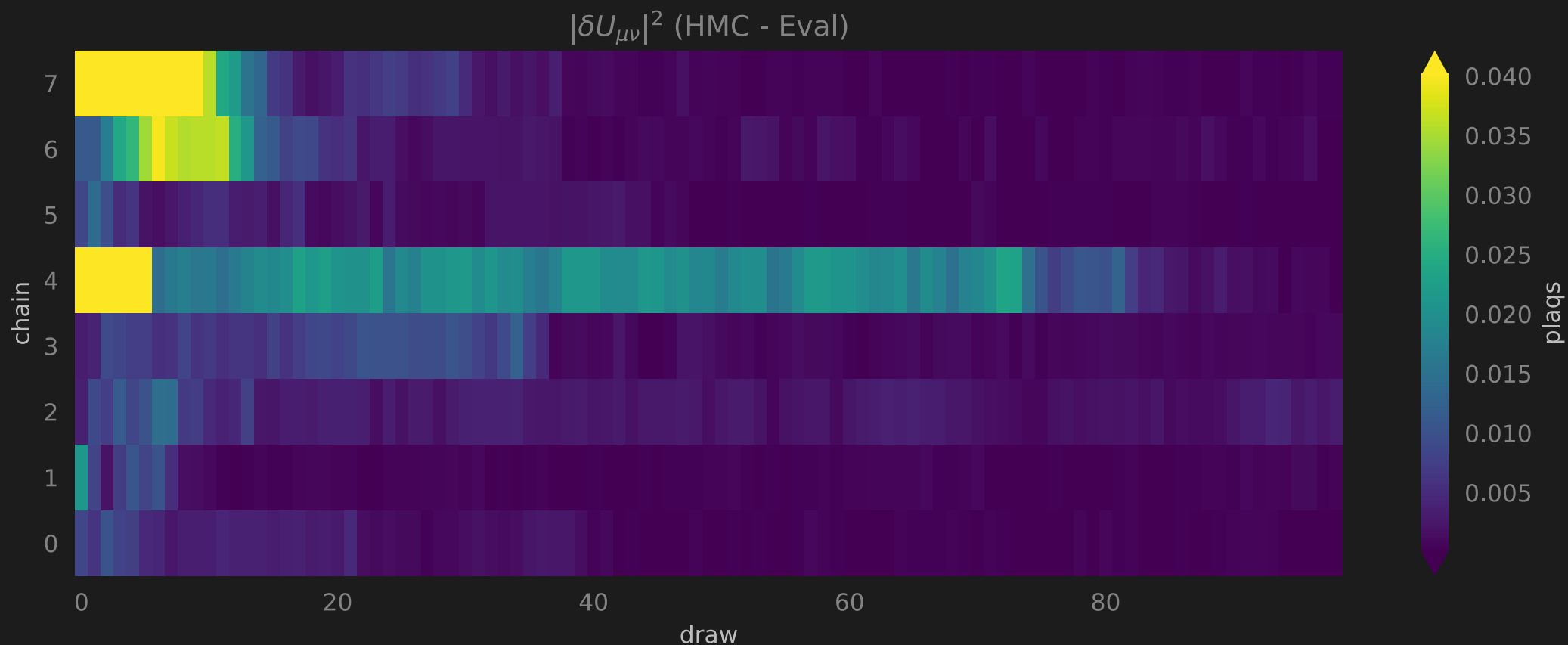



Figure 12: The difference in the average plaquette  $|\delta U_{\mu\nu}|^2$  between the trained model and HMC





# Next Steps


- Further code development
  -  [saforem2/12hmc-qcd](#)
- Continue to use / test different network architectures
  - Gauge equivariant NNs for  $U_\mu(x)$  update
- Continue to test different loss functions for training
- Scaling:
  - Lattice volume
  - Network size
  - Batch size
  - # of GPUs

# Thank you!

 [samforeman.me](https://samforeman.me)

 [saforem2](https://github.com/saforem2)

 [@saforem2](https://twitter.com/saforem2)

 [foremans@anl.gov](mailto:foremans@anl.gov)





 510 / 21402  l2hmc-qcd  codefactor  A

 arXiv 2112.01582  arXiv 2105.03418




 Config  Hydra  PyTorch  TensorFlow  Visualize in  W&B

# Acknowledgements

- **Links:**

-  Link to github
-  reach out!

- **References:**

- Link to HMC demo
- Link to slides
  -  link to github with slides
-  (Foreman et al. 2022; Foreman, Jin, and Osborn 2022, 2021)
-  (Boyda et al. 2022; Shanahan et al. 2022)




- Huge thank you to:

- Yannick Meurice
- Norman Christ
- Akio Tomiya
- Luchang Jin
- Chulwoo Jung
- Peter Boyle
- Taku Izubuchi
- ECP-CSD group
- **ALCF Staff + Datascience Group**

## Acknowledgements

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

## Links + References

- Code repo  [saforem2/12hmc-qcd](https://github.com/saforem2/12hmc-qcd)
- This talk:  [saforem2/lattice23](https://github.com/saforem2/lattice23)
  - Slides [ [saforem2.github.io/lattice23](https://saforem2.github.io/lattice23)]
- Title Slide Background (worms) animation

# References

- Boyda, Denis et al. 2022. “Applications of Machine Learning to Lattice Quantum Field Theory.” In *Snowmass 2021*. <https://arxiv.org/abs/2202.05838>.
- Foreman, Sam, Taku Izubuchi, Luchang Jin, Xiao-Yong Jin, James C. Osborn, and Akio Tomiya. 2022. “HMC with Normalizing Flows.” *PoS LATTICE2021*: 073. <https://doi.org/10.22323/1.396.0073>.
- Foreman, Sam, Xiao-Yong Jin, and James C. Osborn. 2021. “Deep Learning Hamiltonian Monte Carlo.” In *9th International Conference on Learning Representations*. <https://arxiv.org/abs/2105.03418>.
- . 2022. “LeapfrogLayers: A Trainable Framework for Effective Topological Sampling.” *PoS LATTICE2021* (May): 508. <https://doi.org/10.22323/1.396.0508>.
- Shanahan, Phiala et al. 2022. “Snowmass 2021 Computational Frontier CompF03 Topical Group Report: Machine Learning,” September. <https://arxiv.org/abs/2209.07559>.

# Extras

# Integrated Autocorrelation Time

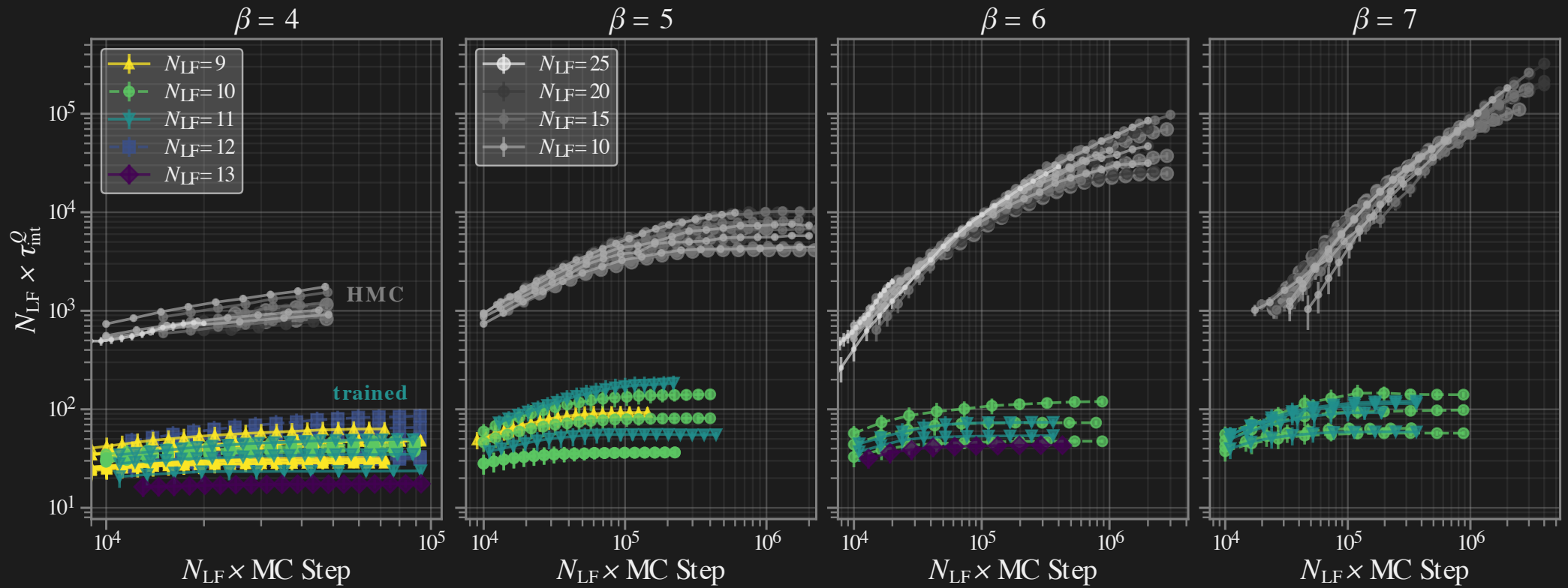
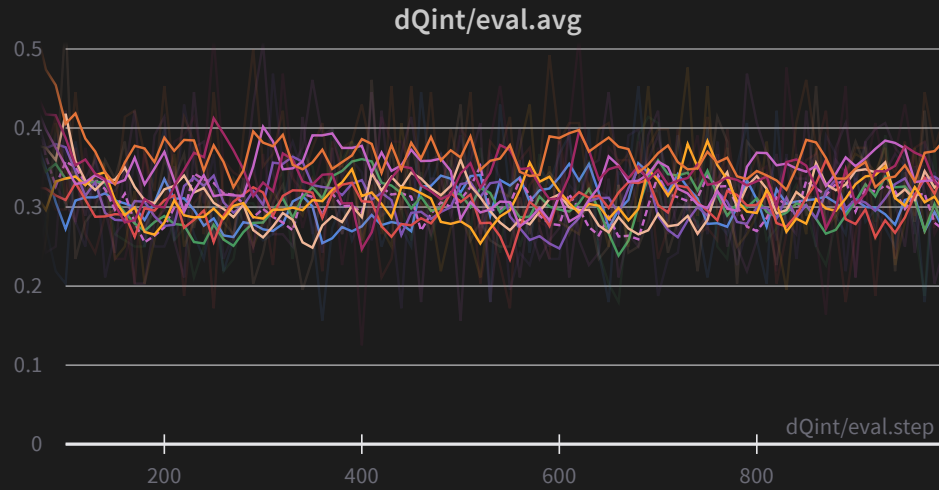


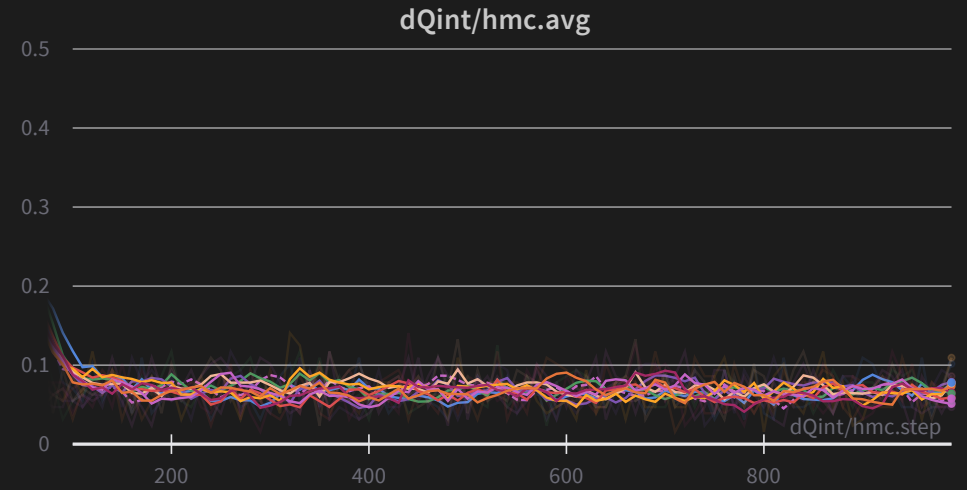
Figure 13: Plot of the integrated autocorrelation time for both the trained model (colored) and HMC (greyscale).



# Comparison



(a) *Trained model*



(b) *Generic HMC*

Figure 14: Comparison of  $\langle \delta Q \rangle = \frac{1}{N} \sum_{i=k}^N \delta Q_i$  for the trained model Figure 14 (a) vs. HMC Figure 14 (b)

# Plaquette analysis: $x_P$

Deviation from  $V \rightarrow \infty$  limit,  $x_P^*$

Average  $\langle x_P \rangle$ , with  $x_P^*$  (dotted-lines)

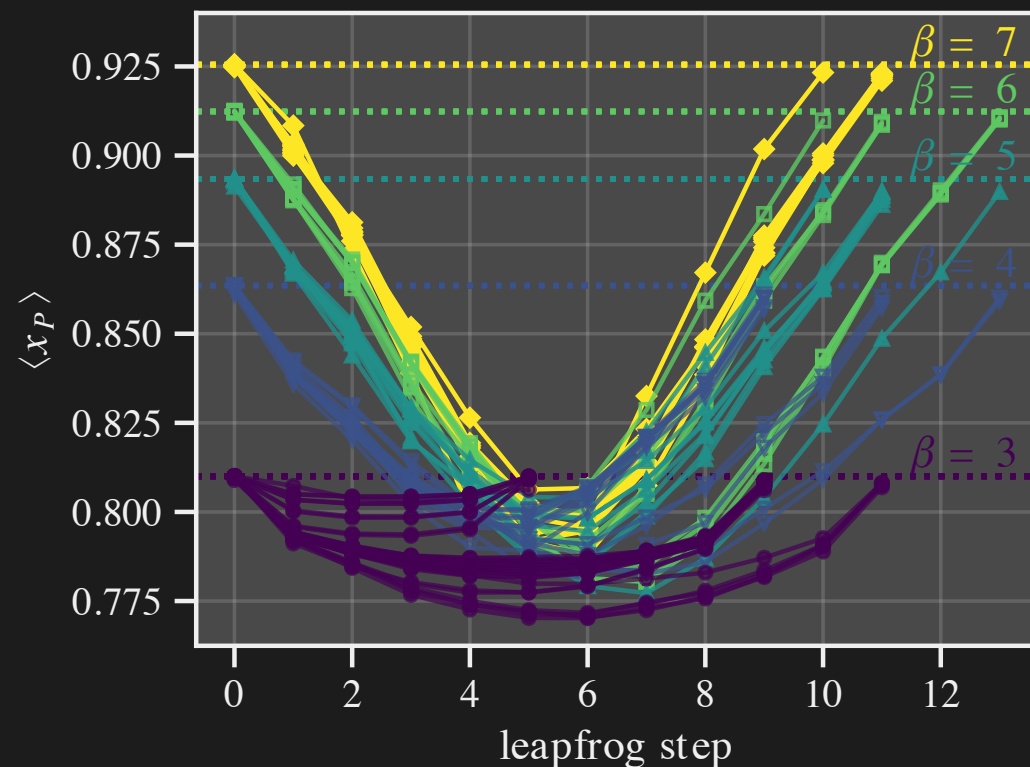
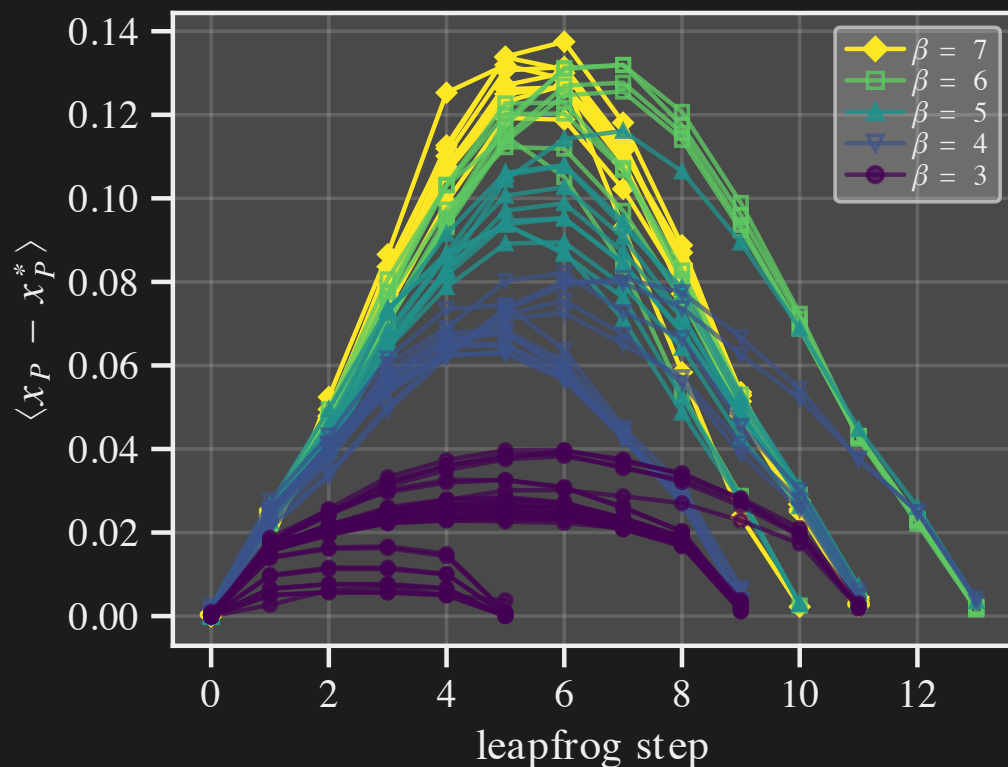


Figure 15: Plot showing how **average plaquette**,  $\langle x_P \rangle$  varies over a single trajectory for models trained at different  $\beta$ , with varying trajectory lengths  $N_{LF}$

# Loss Function

- Want to maximize the *expected* squared charge difference<sup>1</sup>:

$$\mathcal{L}_\theta (\xi^*, \xi) = \mathbb{E}_{p(\xi)} \left[ -\delta Q^2 (\xi^*, \xi) \cdot A(\xi^*|\xi) \right]$$

- Where:

- $\delta Q$  is the *tunneling rate*:

$$\delta Q(\xi^*, \xi) = |Q^* - Q|$$

- $A(\xi^*|\xi)$  is the probability<sup>2</sup> of accepting the proposal  $\xi^*$ :

$$A(\xi^*|\xi) = \min \left( 1, \frac{p(\xi^*)}{p(\xi)} \left| \frac{\partial \xi^*}{\partial \xi^T} \right| \right)$$

1. Where  $\xi^*$  is the *proposed* configuration (prior to Accept / Reject)

2. And  $\left| \frac{\partial \xi^*}{\partial \xi^T} \right|$  is the Jacobian of the transformation from  $\xi \rightarrow \xi^*$

## $v$ -Update<sup>1</sup>

- forward ( $d = +$ ):

$$\Gamma^+ : (x, v) \rightarrow v' := v \cdot e^{\frac{\varepsilon}{2} s_v} - \frac{\varepsilon}{2} [F \cdot e^{\varepsilon q_v} + t_v]$$

- backward ( $d = -$ ):

$$\Gamma^- : (x, v) \rightarrow v' := e^{-\frac{\varepsilon}{2} s_v} \left\{ v + \frac{\varepsilon}{2} [F \cdot e^{\varepsilon q_v} + t_v] \right\}$$

1. Note that  $(\Gamma^+)^{-1} = \Gamma^-$ , i.e.  $\Gamma^+ [\Gamma^-(x, v)] = \Gamma^- [\Gamma^+(x, v)] = (x, v)$

## $x$ -Update

- forward ( $d = +$ ):

$$\Lambda^{+}(x, v) = x \cdot e^{\frac{\varepsilon}{2} s_x} - \frac{\varepsilon}{2} [v \cdot e^{\varepsilon q_x} + t_x]$$

- backward ( $d = -$ ):

$$\Lambda^{-}(x, v) = e^{-\frac{\varepsilon}{2} s_x} \left\{ x + \frac{\varepsilon}{2} [v \cdot e^{\varepsilon q_x} + t_x] \right\}$$

# Lattice Gauge Theory (2D $U(1)$ )

## Link Variables

$$U_\mu(n) = e^{ix_\mu(n)} \in \mathbb{C}, \quad \text{where}$$

$$x_\mu(n) \in [-\pi, \pi)$$

## Wilson Action

$$S_\beta(x) = \beta \sum_P \cos x_P,$$

$$x_P = [x_\mu(n) + x_\nu(n + \hat{\mu}) - x_\mu(n + \hat{\nu}) - x_\nu(n)]$$

**Note:**  $x_P$  is the product of links around  $1 \times 1$  square, called a “plaquette”

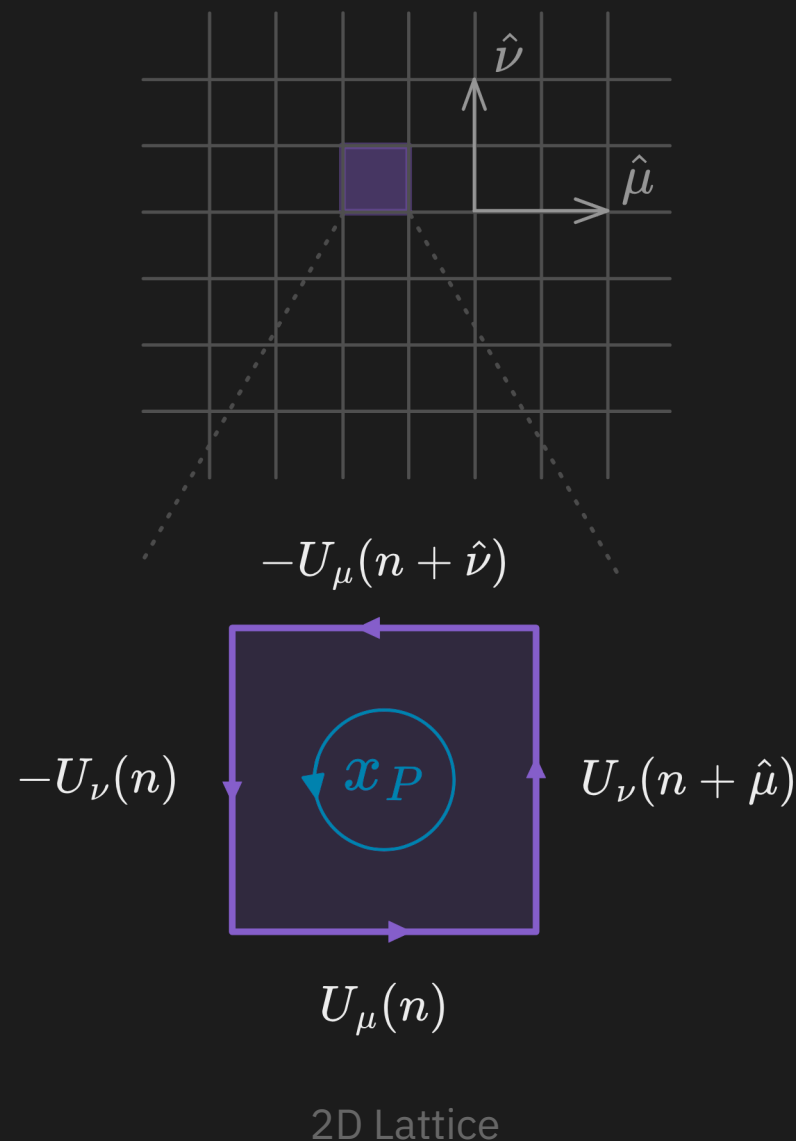


Figure 16: Jupyter Notebook



# Annealing Schedule

- Introduce an *annealing schedule* during the training phase:

$$\{\gamma_t\}_{t=0}^N = \{\gamma_0, \gamma_1, \dots, \gamma_{N-1}, \gamma_N\}$$

where  $\gamma_0 < \gamma_1 < \dots < \gamma_N \equiv 1$ , and  $|\gamma_{t+1} - \gamma_t| \ll 1$

- **Note:**
  - for  $|\gamma_t| < 1$ , this rescaling helps to reduce the height of the energy barriers  $\implies$
  - easier for our sampler to explore previously inaccessible regions of the phase space



# Networks 2D $U(1)$

- Stack gauge links as  $\text{shape}(U_\mu) = [\text{Nb}, 2, \text{Nt}, \text{Nx}] \in \mathbb{C}$

$$x_\mu(n) := [\cos(x), \sin(x)]$$

with  $\text{shape}(x_\mu) = [\text{Nb}, 2, \text{Nt}, \text{Nx}, 2] \in \mathbb{R}$

- $x$ -Network:
  - $\psi_\theta : (x, v) \longrightarrow (s_x, t_x, q_x)$
- $v$ -Network:
  - $\varphi_\theta : (x, v) \longrightarrow (s_v, t_v, q_v)$

## Networks 2D $U(1)$

- Stack gauge links as  $\text{shape}(U_\mu) = [\text{Nb}, 2, \text{Nt}, \text{Nx}] \in \mathbb{C}$

$$x_\mu(n) := [\cos(x), \sin(x)]$$

with  $\text{shape}(x_\mu) = [\text{Nb}, 2, \text{Nt}, \text{Nx}, 2] \in \mathbb{R}$

- $x$ -Network:

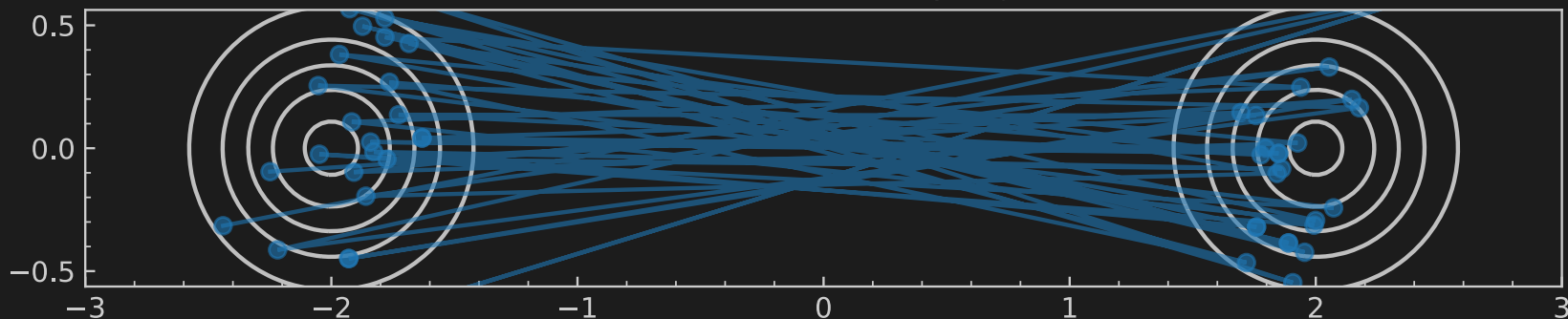
- $\psi_\theta : (x, v) \longrightarrow (s_x, t_x, q_x)$

- $v$ -Network:

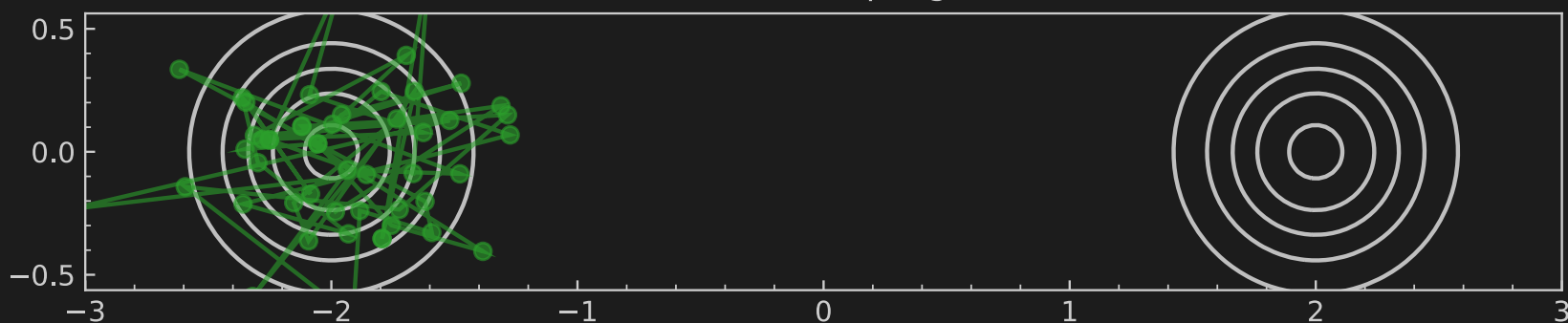
- $\varphi_\theta : (x, v) \longrightarrow (s_v, t_v, q_v) \longleftarrow$  lets look at this

# Toy Example: $\text{GMM} \in \mathbb{R}^2$

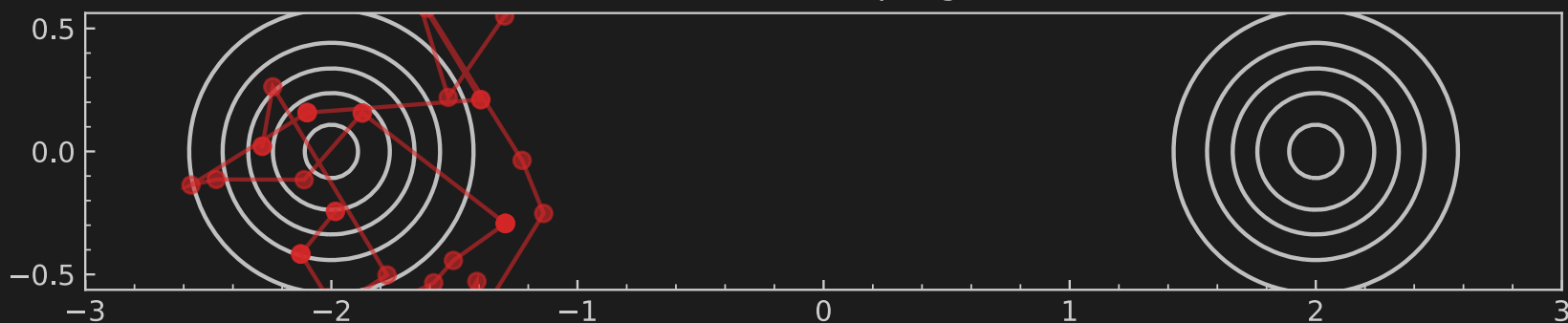
Gaussian Mixture Model Sampling via L2HMC



Gaussian Mixture Model Sampling via HMC ( $\varepsilon = 0.25$ )



Gaussian Mixture Model Sampling via HMC ( $\varepsilon = 0.5$ )



# Physical Quantities

- To estimate physical quantities, we:
  - calculate physical observables at **increasing** spatial resolution
  - perform extrapolation to continuum limit

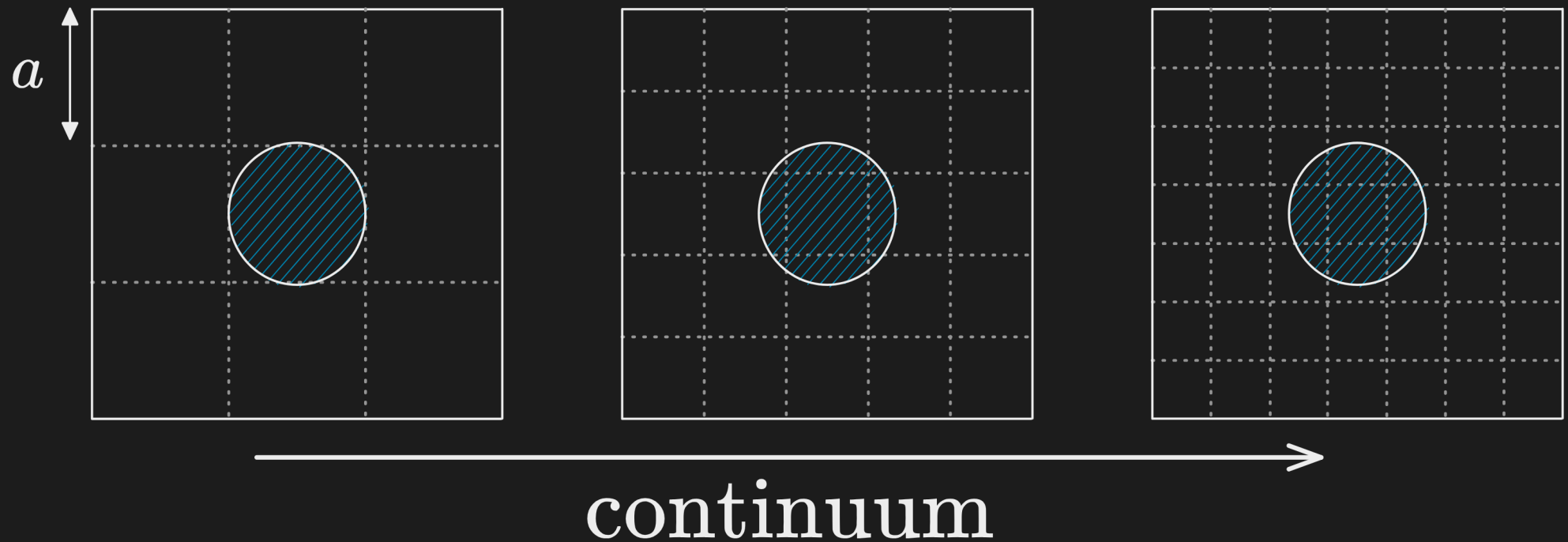


Figure 17: Increasing the physical resolution ( $a \rightarrow 0$ ) allows us to make predictions about numerical values of physical quantities in the continuum limit.

