cusp.ai

Continuous flows for SU(N)

Exploring general flow architectures for pure gauge theory

Pim de Haan

ML4PhysChem 2024.10.24

Collaborators

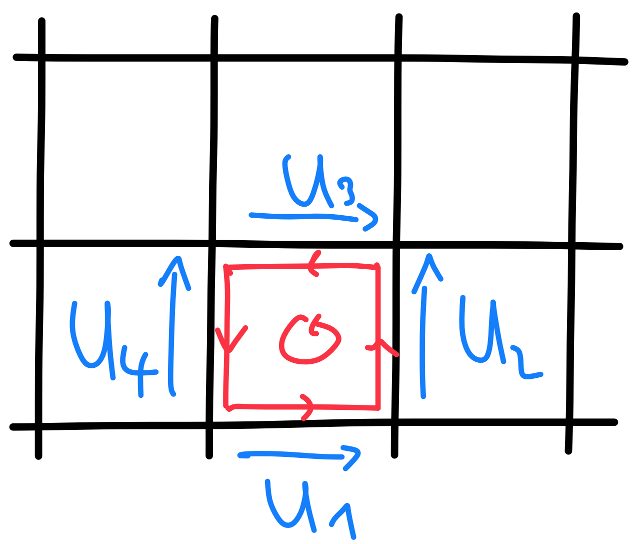
Mathis Gerdes

Roberto Bondesan

Miranda Cheng

Lattice gauge theory

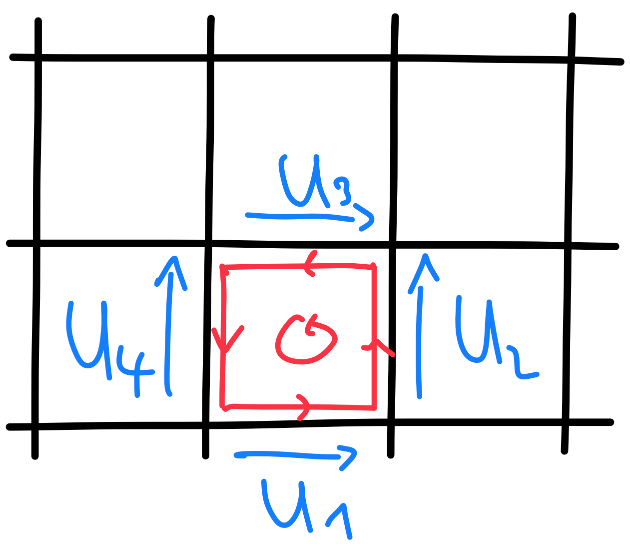
Wilson action



Group element at each edge $U_e \in \mathbf{SU(N)}$

Lattice gauge theory

Wilson action



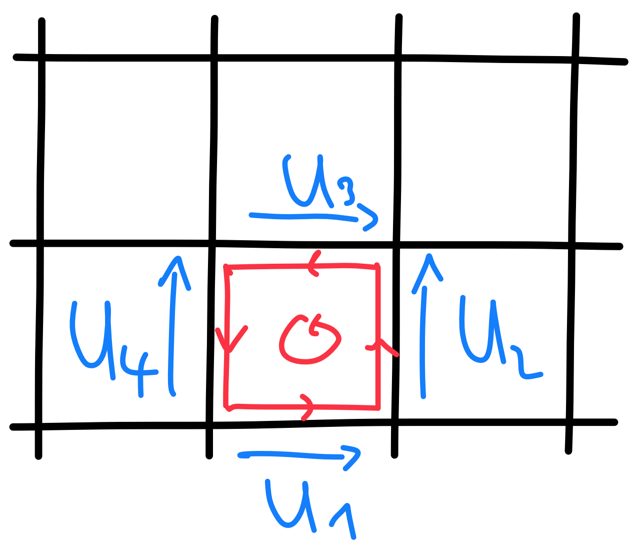
Group element at each edge $U_e \in \mathbf{SU(N)}$

Wilson loop trace

$$W = \mathrm{tr}(U_1 U_2 U_3^\dagger U_4^\dagger)$$

Lattice gauge theory

Wilson action



Group element at each edge $U_e \in \mathbf{SU(N)}$

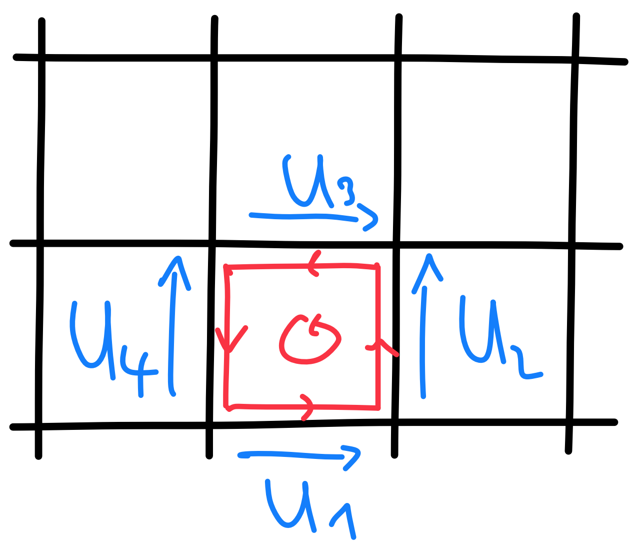
Wilson loop trace

$$W = \mathrm{tr}(U_1 U_2 U_3^\dagger U_4^\dagger)$$

Wilson action $S = -\dfrac{\beta}{N}\sum_x \mathrm{Re}\left[W(x)\right]$

Lattice gauge theory

Wilson action



Group element at each edge $U_e \in \mathrm{SU(N)}$

Wilson loop trace

$$W = \mathrm{tr}(U_1 U_2 U_3^\dagger U_4^\dagger)$$

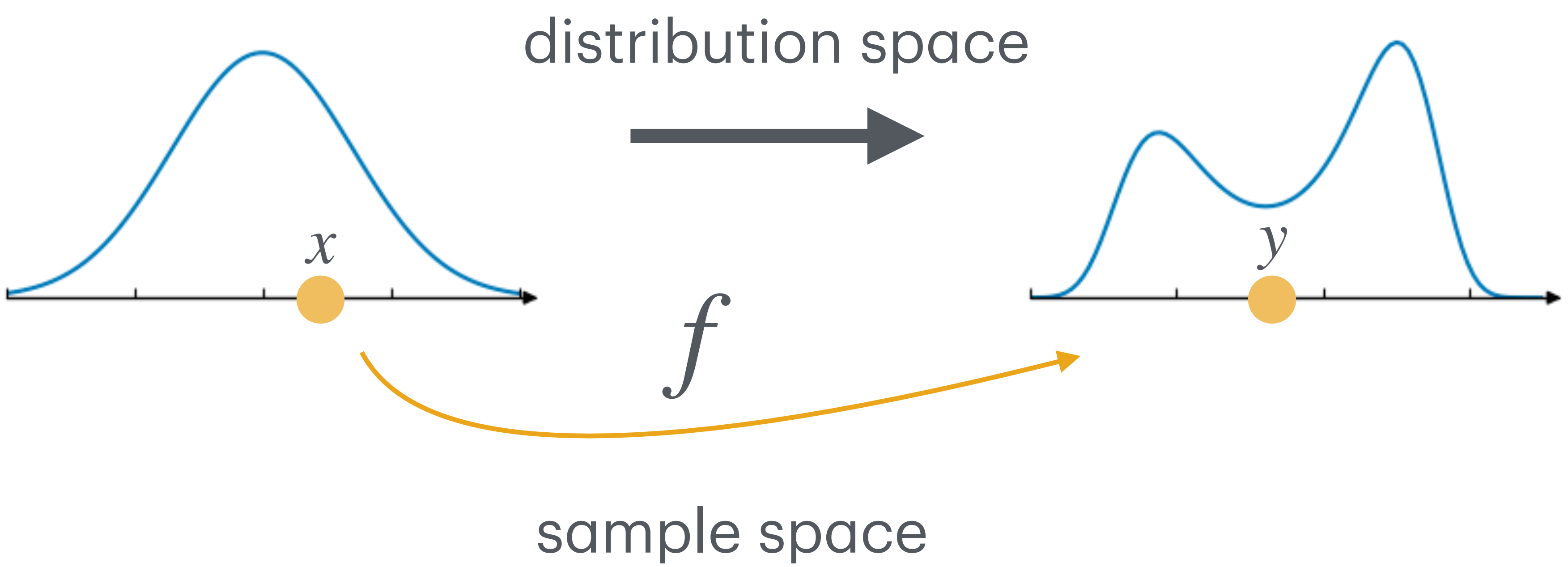
Wilson action $S = -\dfrac{\beta}{N} \sum_x \mathrm{Re}\left[W(x)\right]$

$\longrightarrow$ Want to sample U-configurations

$$p(U) \propto e^{-S[U]}$$

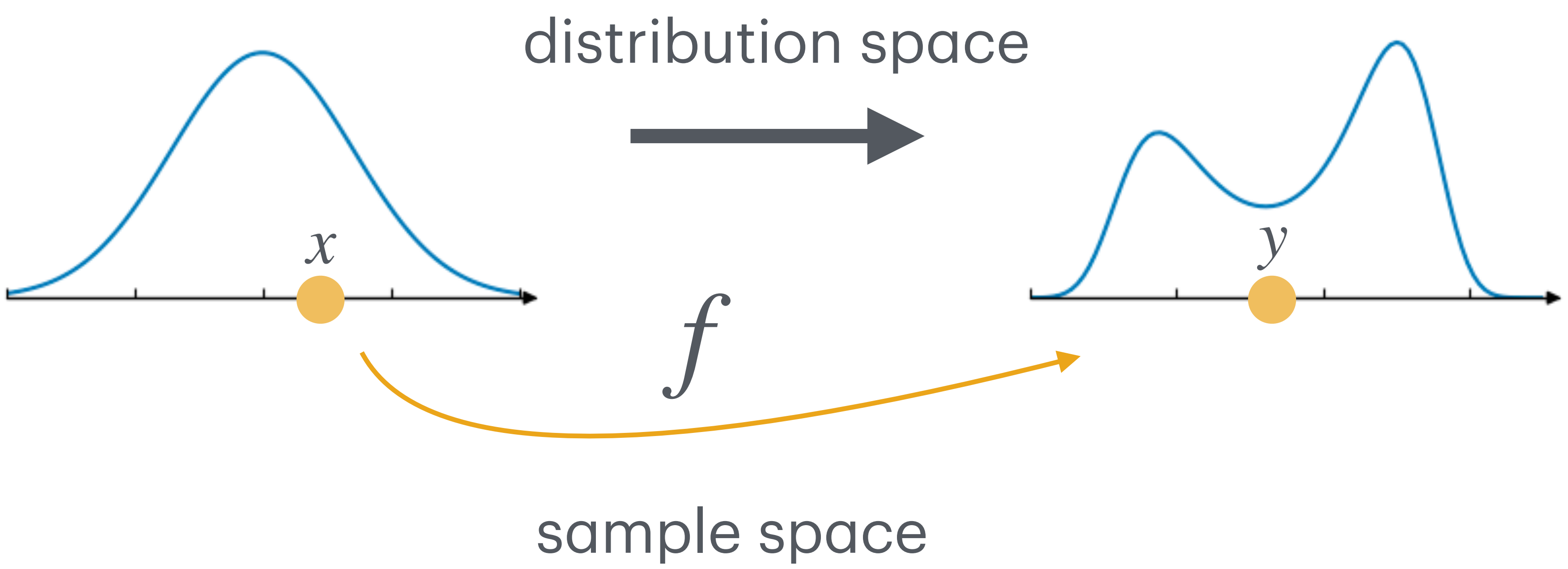
Change of variables

Transforming probability densities



Change of variables

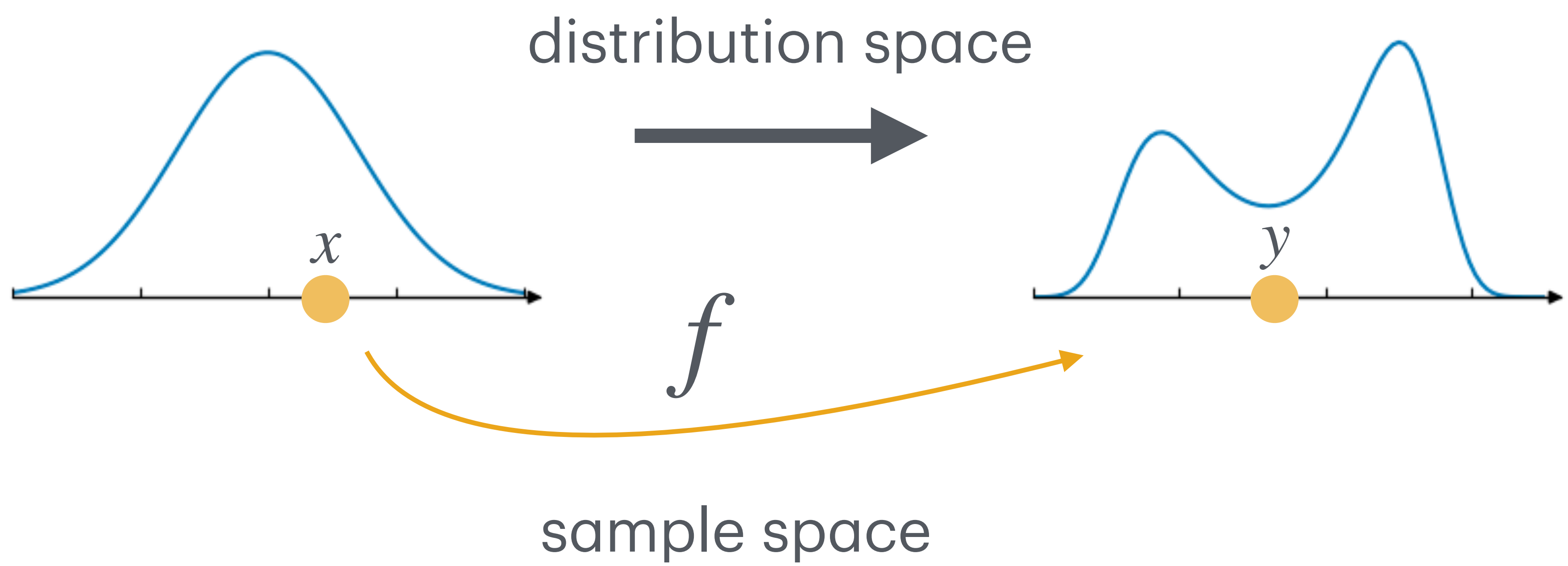
Transforming probability densities



$$q(y) = q\left(f^{-1}(y)\right) \cdot \left| \det \frac{\partial f}{\partial x} \right|^{-1}$$

Change of variables

Transforming probability densities

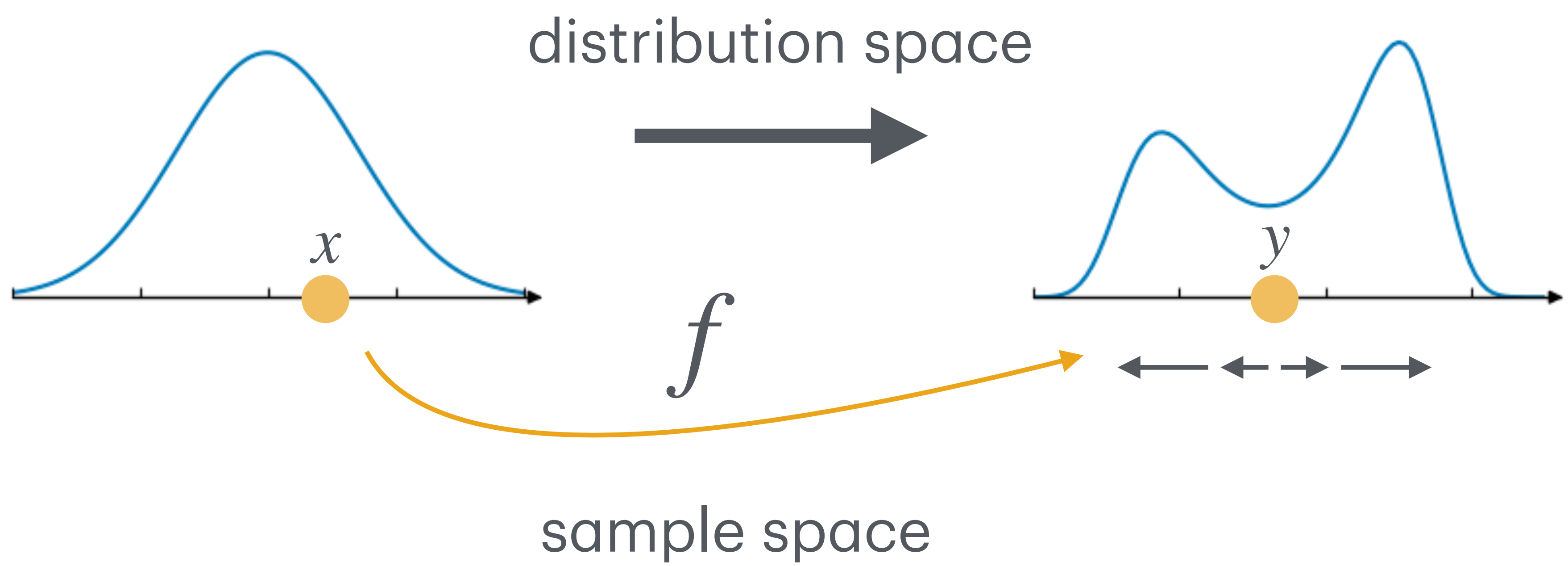


$$q(y) = q\left(f^{-1}(y)\right) \cdot \left|\det \frac{\partial f}{\partial x}\right|^{-1}$$

Source point

Change of variables

Transforming probability densities



Change of density

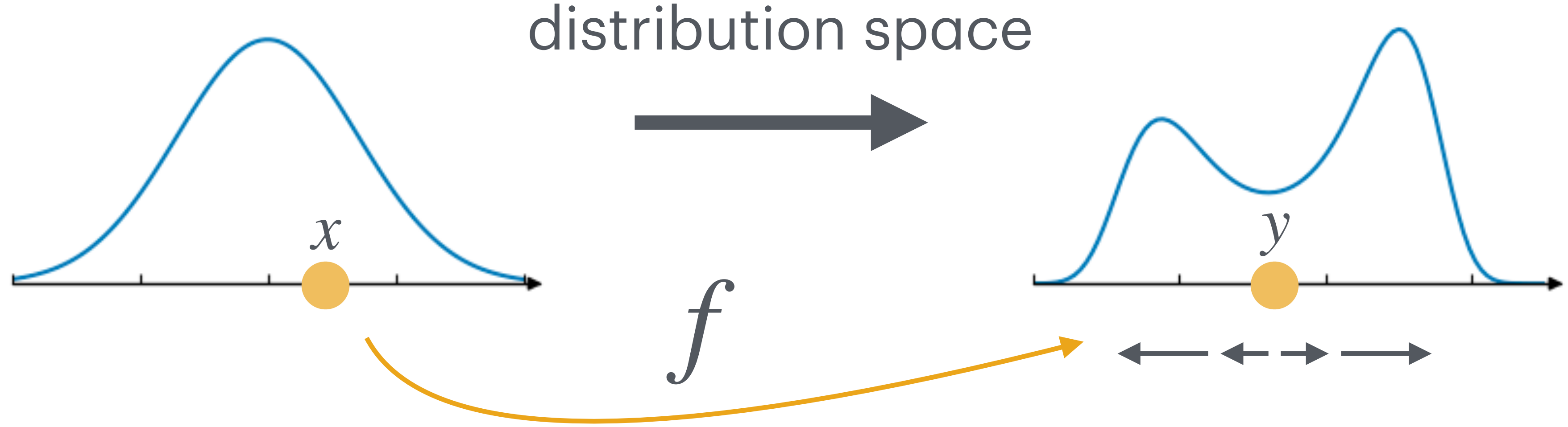
$$q(y) = q\left(f^{-1}(y)\right) \cdot \left|\det \frac{\partial f}{\partial x}\right|^{-1}$$

Source point

Change of variables

Transforming probability densities

distribution space



sample space

Change of density

$$q(y) = q\left(f^{-1}(y)\right) \cdot \left|\det \frac{\partial f}{\partial x}\right|^{-1}$$

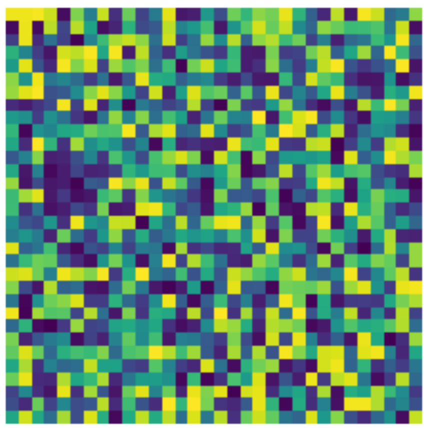
Source point

$$\mathbb{E}_{y \sim q}[\log q(y) - \log p(y)] = \mathbb{E}_{y \sim q}[S(y) + \log p(y)] + \text{const}$$

Normalizing flows

Learning f

trivial theory



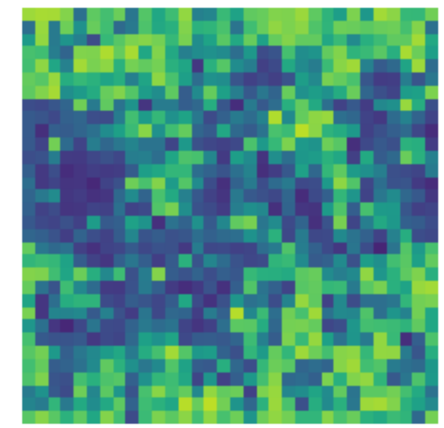
$\mathcal{N}$

bijection f

$\longleftrightarrow$

"Normalizing flow"

interacting theory



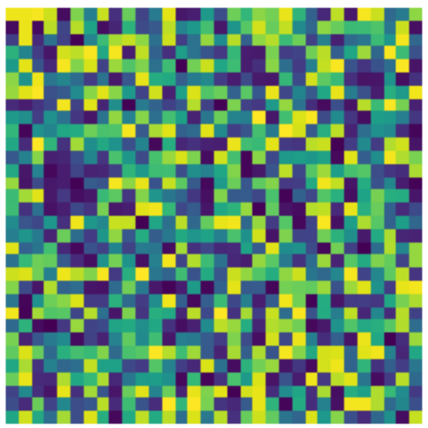
$e^{-S[\phi]}$

We want to ***learn*** a *trivializing map* f .

Normalizing flows

Learning f

trivial theory

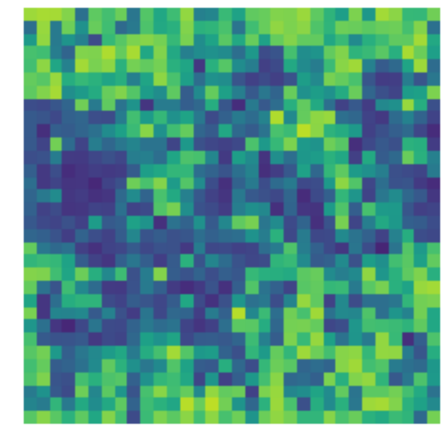


$\mathcal{N}$

bijection f

"Normalizing flow"

interacting theory



$e^{-S[\phi]}$

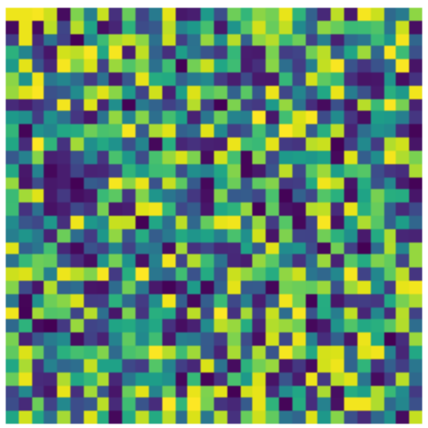
<u>We want to ***learn*** a *trivializing map* f .</u>

To compute model probability:

Normalizing flows

Learning f

trivial theory



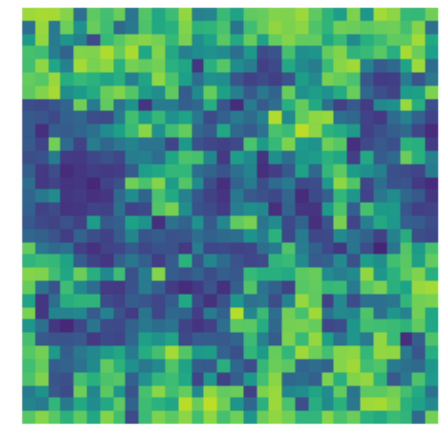
$\mathcal{N}$

bijection f

$\longleftrightarrow$

"Normalizing flow"

interacting theory



$e^{-S[\phi]}$

We want to ***learn*** a *trivializing map* f .

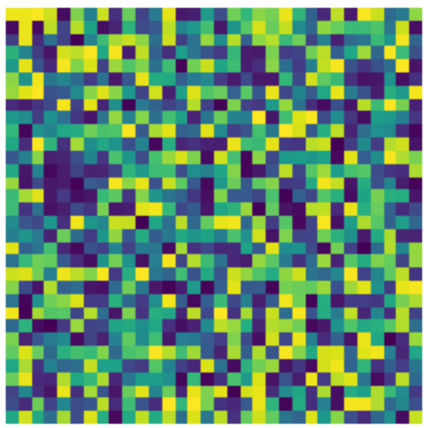
To compute model probability:

- f must be bijective.

Normalizing flows

Learning f

trivial theory

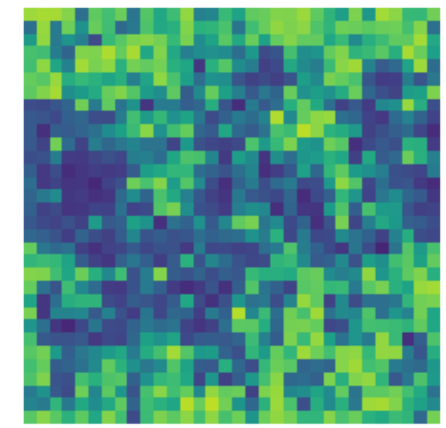


$\mathcal{N}$

bijection f

"Normalizing flow"

interacting theory



$e^{-S[\phi]}$

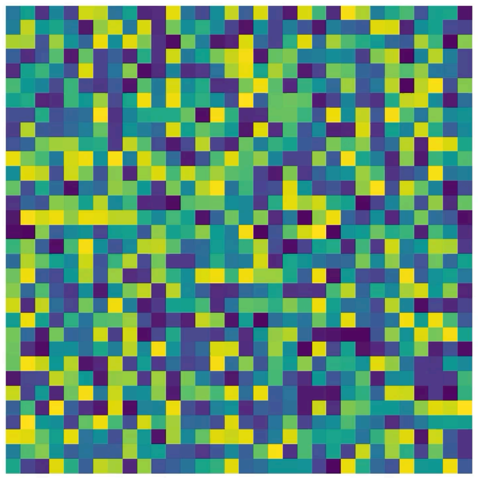
We want to ***learn*** *a trivializing map f.*

To compute model probability:

- f must be bijective.
- Computing the det-Jacobian must be tractable.

$$p(y) = p\left(f^{-1}(y)\right) \cdot \left|\det \frac{\partial f}{\partial x}\right|^{-1}$$

Continuous normalizing flows

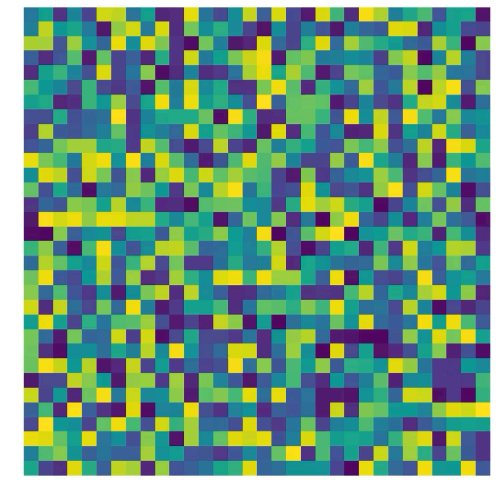


Sample $\phi^0 \sim \mathcal{N}$

Final proposal $\phi^{t=1}$

Solve $\frac{d}{dt}\phi = g_\theta(\phi, t)$

Continuous normalizing flows

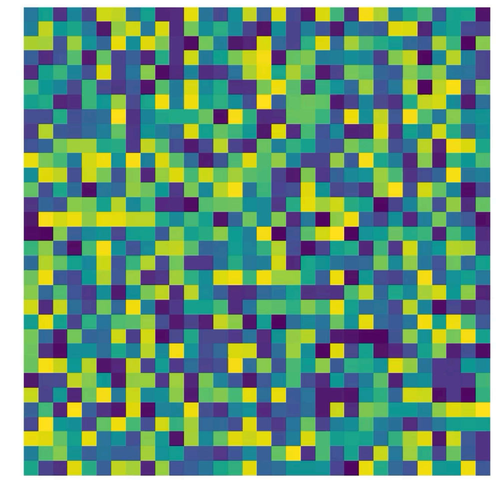


Sample $\phi^0 \sim \mathcal{N}$

Final proposal $\phi^{t=1}$

Solve $\frac{d}{dt}\phi = g_\theta(\phi, t)$

Continuous normalizing flows

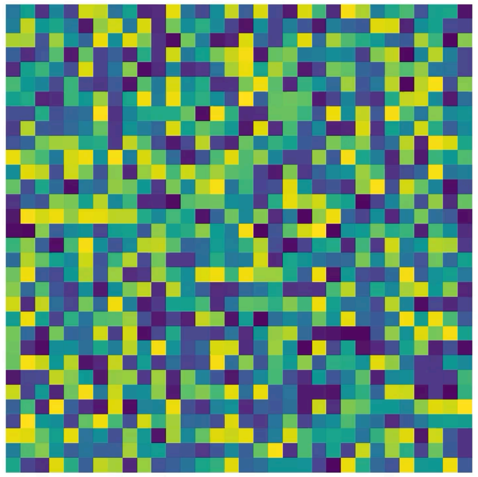


Sample $\phi^0 \sim \mathcal{N}$ Final proposal $\phi^{t=1}$

Solve $\frac{d}{dt}\phi = g_\theta(\phi, t)$

- ODE always invertible, architecture of g_θ unconstrained!

Continuous normalizing flows



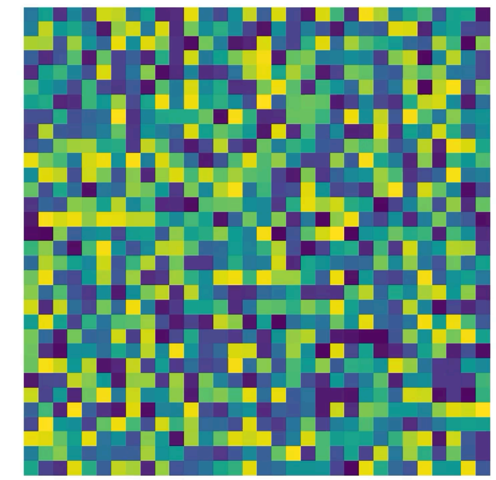
Sample $\phi^0 \sim \mathcal{N}$

Final proposal $\phi^{t=1}$

Solve $\dfrac{d}{dt}\phi = g_\theta(\phi, t)$

- ODE always invertible, architecture of g_θ unconstrained!
- ODE for $p(\phi^t)$ given by divergence: $\dfrac{d}{dt}\log p(\phi) = -\nabla \cdot \dot{\phi}$

Continuous normalizing flows



Sample $\phi^0 \sim \mathcal{N}$

Final proposal $\phi^{t=1}$

Solve $\dfrac{d}{dt}\phi = g_\theta(\phi, t)$

- ODE always invertible, architecture of g_θ unconstrained!
- ODE for $p(\phi^t)$ given by divergence: $\dfrac{d}{dt}\log p(\phi) = -\nabla \cdot \dot{\phi}$
  - Needs to be tractable

Continuous flows for ϕ^4

SciPost Phys. 15, 238 (2023)

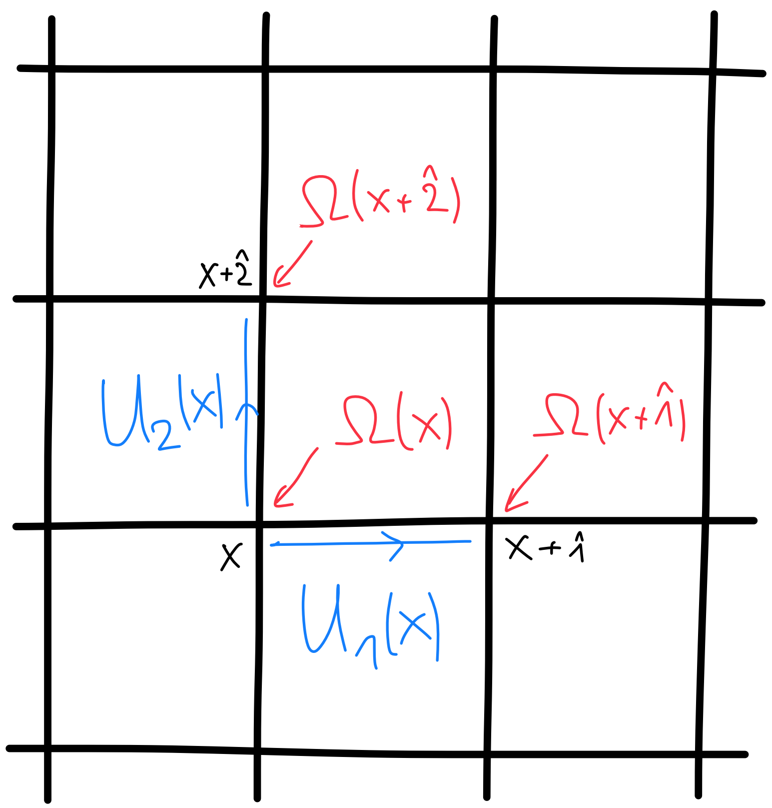
Learning lattice quantum field theories with equivariant continuous flows

Mathis Gerdes[1⋆∘], Pim de Haan[2,3†∘], Corrado Rainone[3],
Roberto Bondesan[3] and Miranda C. N. Cheng[1,4,5]

Gauge symmetry

How objects transform

$$U_\mu(x) \mapsto \Omega(x)\, U_\mu(x)\, \Omega(x+\hat{\mu})^\dagger$$



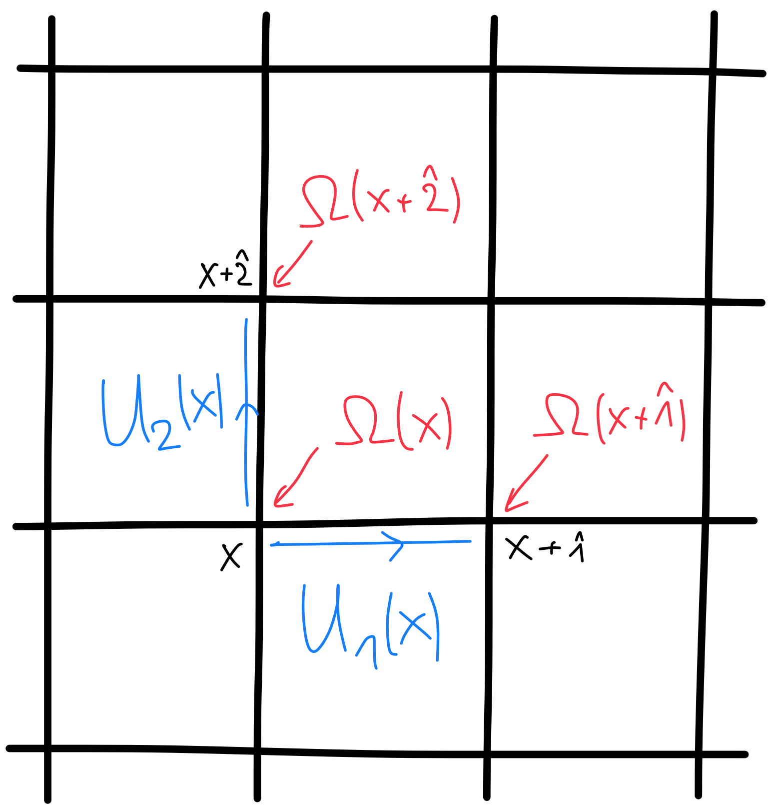
Wilson loop

$$P_{12} = U_1(x)U_2(x+\hat{1})U_1(x+\hat{2})^\dagger U_2(x)^\dagger$$

Gauge symmetry

How objects transform

$$U_\mu(x) \mapsto \Omega(x)\, U_\mu(x)\, \Omega(x+\hat{\mu})^\dagger$$



<u>Wilson loop</u>

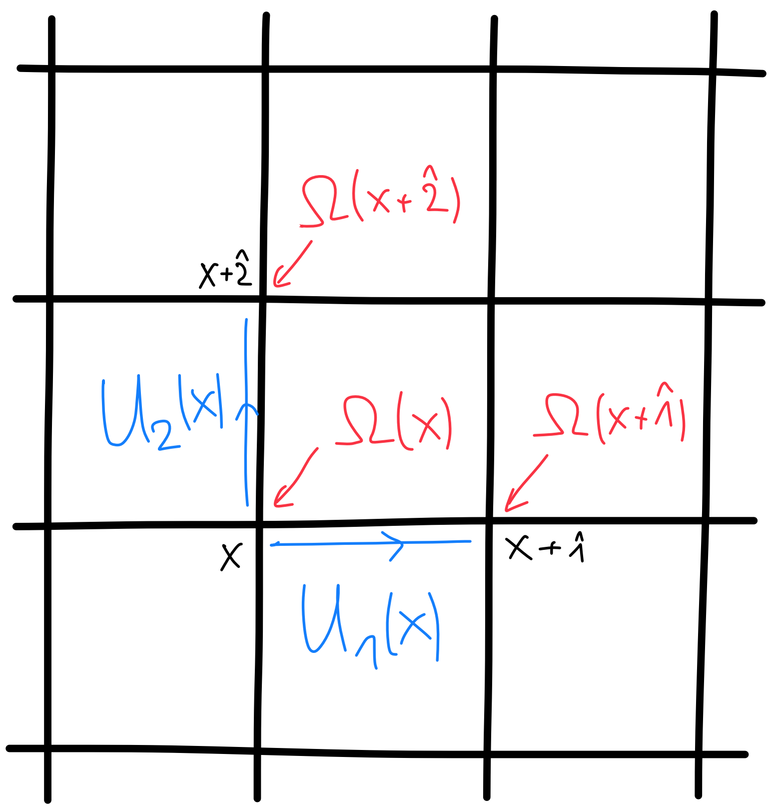
$$P_{12} = U_1(x)U_2(x+\hat{1})U_1(x+\hat{2})^\dagger U_2(x)^\dagger$$

are **equivariant** $P_{12} \mapsto \Omega(x)P_{12}\Omega(x)^\dagger$.

Gauge symmetry

How objects transform

$$U_\mu(x) \mapsto \Omega(x)\, U_\mu(x)\, \Omega(x+\hat{\mu})^\dagger$$



Wilson loop

$$P_{12} = U_1(x)U_2(x+\hat{1})U_1(x+\hat{2})^\dagger U_2(x)^\dagger$$

are **equivariant** $P_{12} \mapsto \Omega(x)P_{12}\Omega(x)^\dagger$.

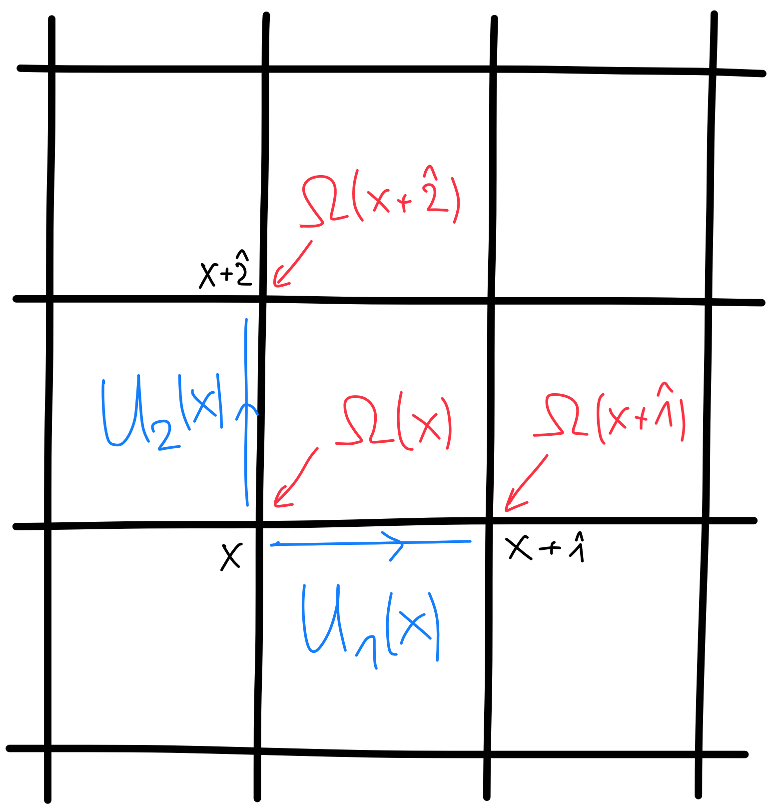
Trace of Wilson loops

$W = \operatorname{tr} P_{12}$ are **invariant**.

Gauge symmetry

How objects transform

$$U_\mu(x) \mapsto \Omega(x)\, U_\mu(x)\, \Omega(x+\hat{\mu})^\dagger$$



Wilson loop

$$P_{12} = U_1(x)U_2(x+\hat{1})U_1(x+\hat{2})^\dagger U_2(x)^\dagger$$

are **equivariant** $P_{12} \mapsto \Omega(x)P_{12}\Omega(x)^\dagger$.

Trace of Wilson loops

$W = \mathrm{tr}\, P_{12}$ are **invariant**.

Gradients of invariants

e.g. $V = \nabla_U W$ are **equivariant**

$$V \mapsto \Omega(x)V\Omega(x)^\dagger$$

Discrete normalizing flows

How to define gauge equivariant flows

Map $P_{\mu\nu} \mapsto P'_{\mu\nu} = f(P_{\mu\nu})$ to update edge in $P_{\mu\nu}$ conditioned on unmodified invariant quantities.

Get an equivariant flow, if map transform under conjugation:
$f(\Omega P \Omega^\dagger) = \Omega f(P) \Omega^\dagger$

xiv:2008.05456

Sampling using SU(N) gauge equivariant flows

Denis Boyda,[1,*] Gurtej Kanwar,[1,†] Sébastien Racanière,[2,‡] Danilo Jimenez Rezende,[2,§]
Michael S. Albergo,[3] Kyle Cranmer,[3] Daniel C. Hackett,[1] and Phiala E. Shanahan[1]

arxiv:2305.02402

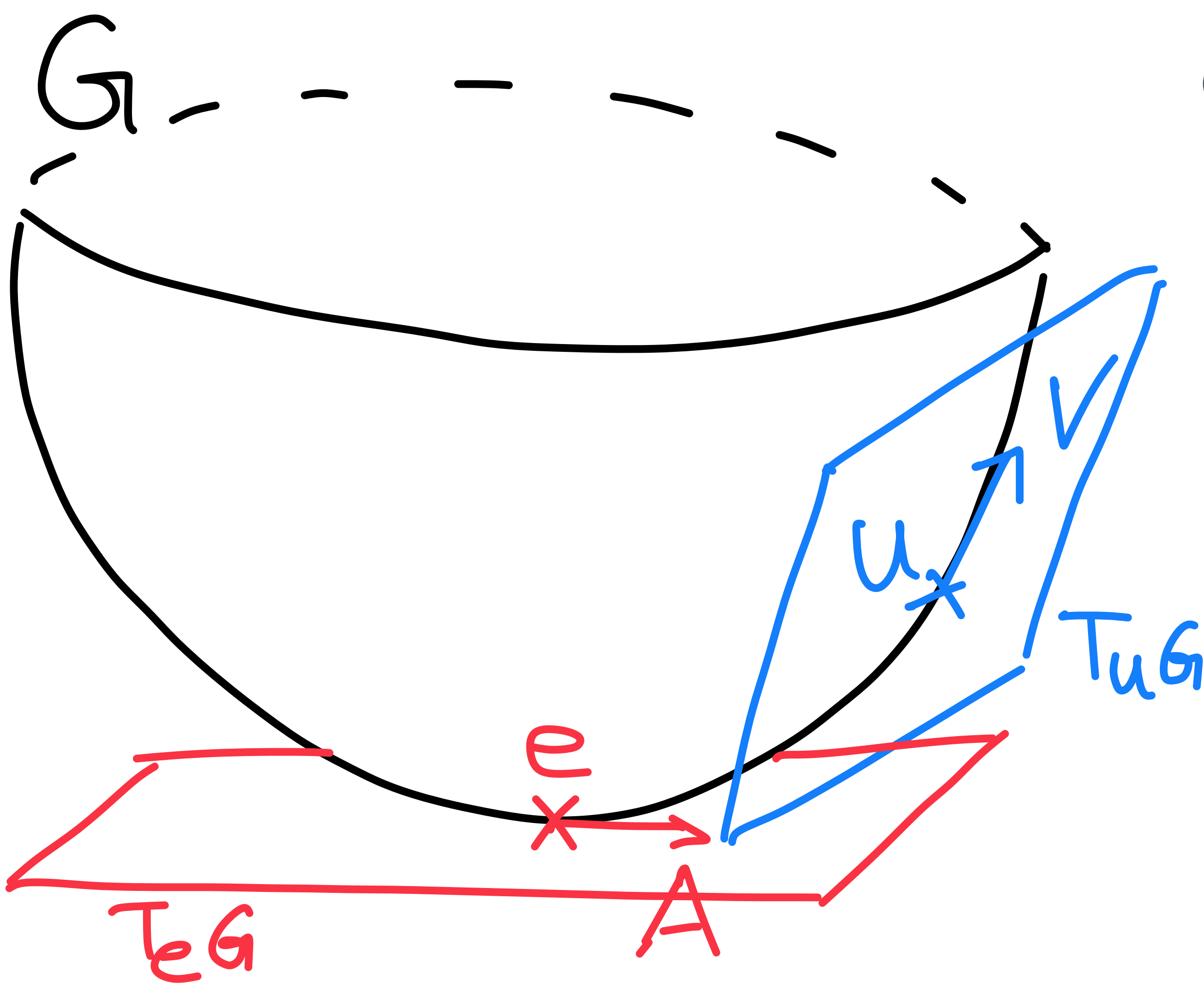
Normalizing flows for lattice gauge theory in arbitrary space-time dimension

Ryan Abbott,[1,2] Michael S. Albergo,[3] Aleksandar Botev,[4] Denis Boyda,[1,2] Kyle Cranmer,[5]
Daniel C. Hackett,[1,2] Gurtej Kanwar,[6,1,2] Alexander G.D.G. Matthews,[4] Sébastien Racanière,[4] Ali
Razavi,[4] Danilo J. Rezende,[4] Fernando Romero-López,[1,2] Phiala E. Shanahan,[1,2] and Julian M. Urban[1,

Continuous flows for gauge theories

Lie groups

A brief reminder



We can parametrize the vector space at U via the Lie algebra:

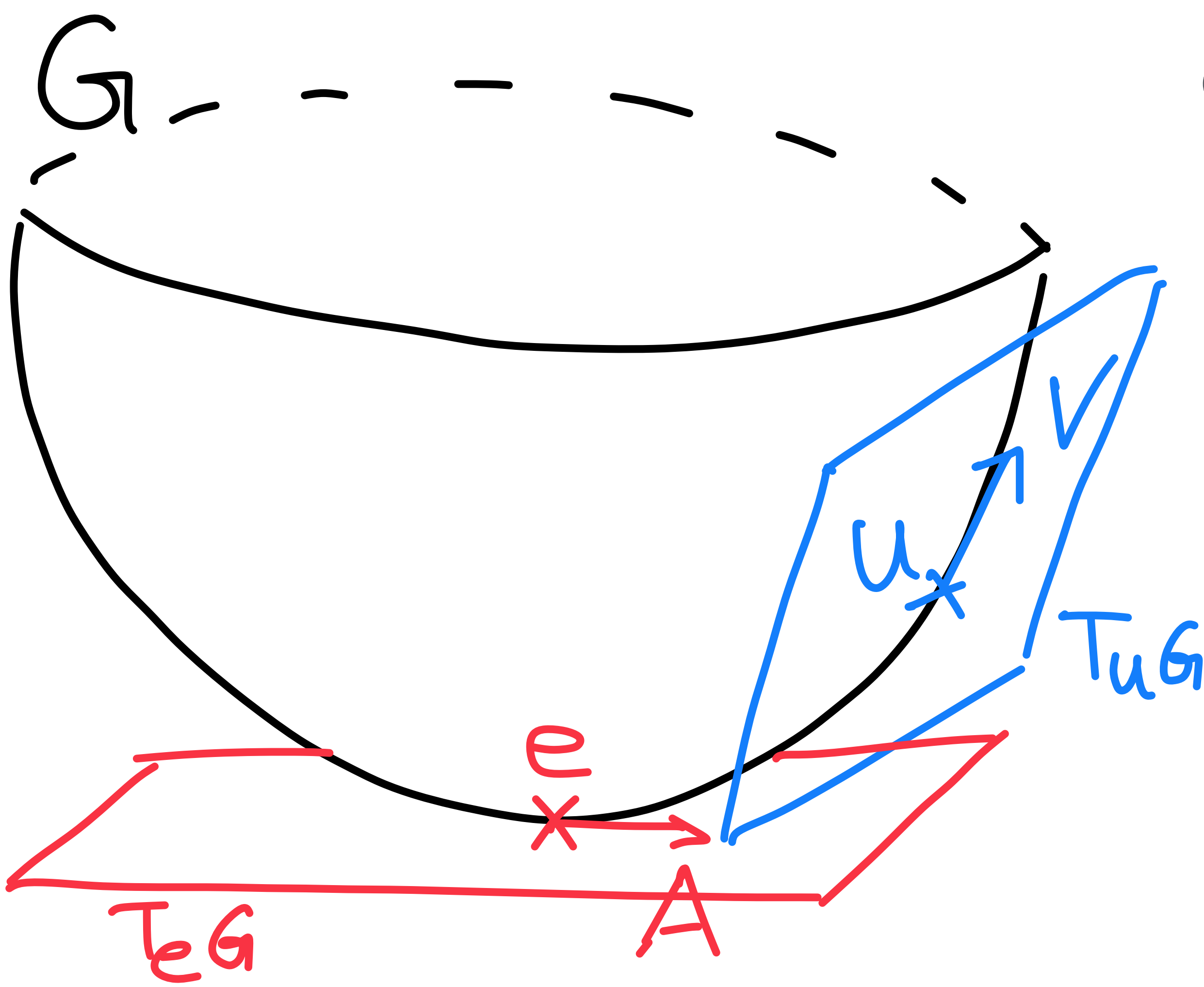
$$V \in T_U G \qquad A := VU^\dagger \in \mathfrak{g} = T_e G$$

$$\boxed{V = AU}$$

Transporting A to vector space at U

Lie groups

A brief reminder



We can parametrize the vector space at U via the Lie algebra:

$$V \in T_U G \qquad A := VU^\dagger \in \mathfrak{g} = T_e G$$

$$\boxed{V = AU}$$

Transporting A to vector space at U

Lie algebra is spanned by generators T^a

In components, $V = A^a T^a U$

Continuous flows for SU(N)

Defining an ODE

In coordinates A^a, general vector at U is: $V = (T^a A^a)U$.

Continuous flows for SU(N)

Defining an ODE

In coordinates A^a, general vector at U is: $V = (T^a A^a)U$.

Path derivative $\partial^a f(U) = \frac{d}{ds}\Big|_{s=0} f(e^{sT^a}U) = Df(T^a U)$.

Then, the gradient is $\nabla f(U) = \partial^a f(U)\, T^a U$.

Continuous flows for SU(N)

Defining an ODE

In coordinates A^a, general vector at U is: $V = (T^a A^a)U$.

Path derivative $\partial^a f(U) = \left.\frac{d}{ds}\right|_{s=0} f(e^{sT^a}U) = Df(T^a U)$.

Then, the gradient is $\nabla f(U) = \partial^a f(U)\, T^a U$.

To define our flow, the network should output an algebra element:

$$\frac{d}{dt}U = A^a(U)\, T^a U$$

Continuous flows for SU(N)

Defining an ODE

In coordinates A^a, general vector at U is: $V = (T^a A^a)U$.

Path derivative $\partial^a f(U) = \frac{d}{ds}\Big|_{s=0} f(e^{sT^a}U) = Df(T^a U)$.

Then, the gradient is $\nabla f(U) = \partial^a f(U)\, T^a U$.

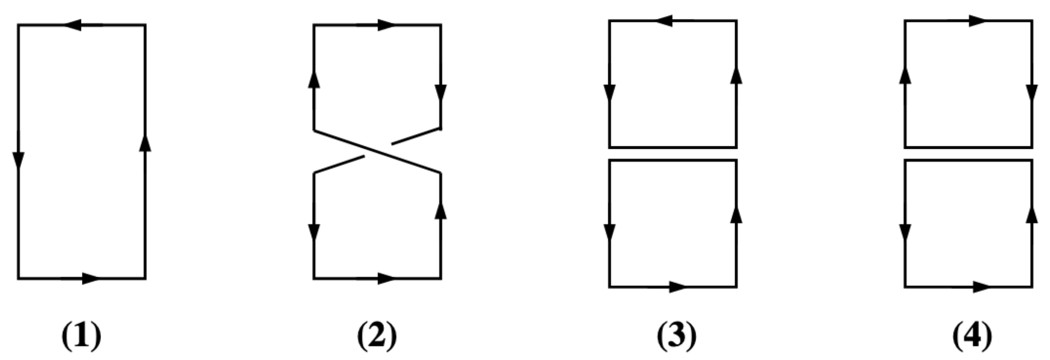
To define our flow, the network should output an algebra element:

$$\frac{d}{dt}U = A^a(U)\, T^a U$$

Continuous flows for SU(N) lattices

Gradient flows

Define $A^a = \partial^a S$ as the gradient of some potential, given as sums and products of Wilson loops.



Can extend/do better by learning coefficients by gradient descent

Trivializing maps, the Wilson flow and the HMC algorithm

Martin Lüscher

Learning Trivializing Gradient Flows for Lattice Gauge Theories

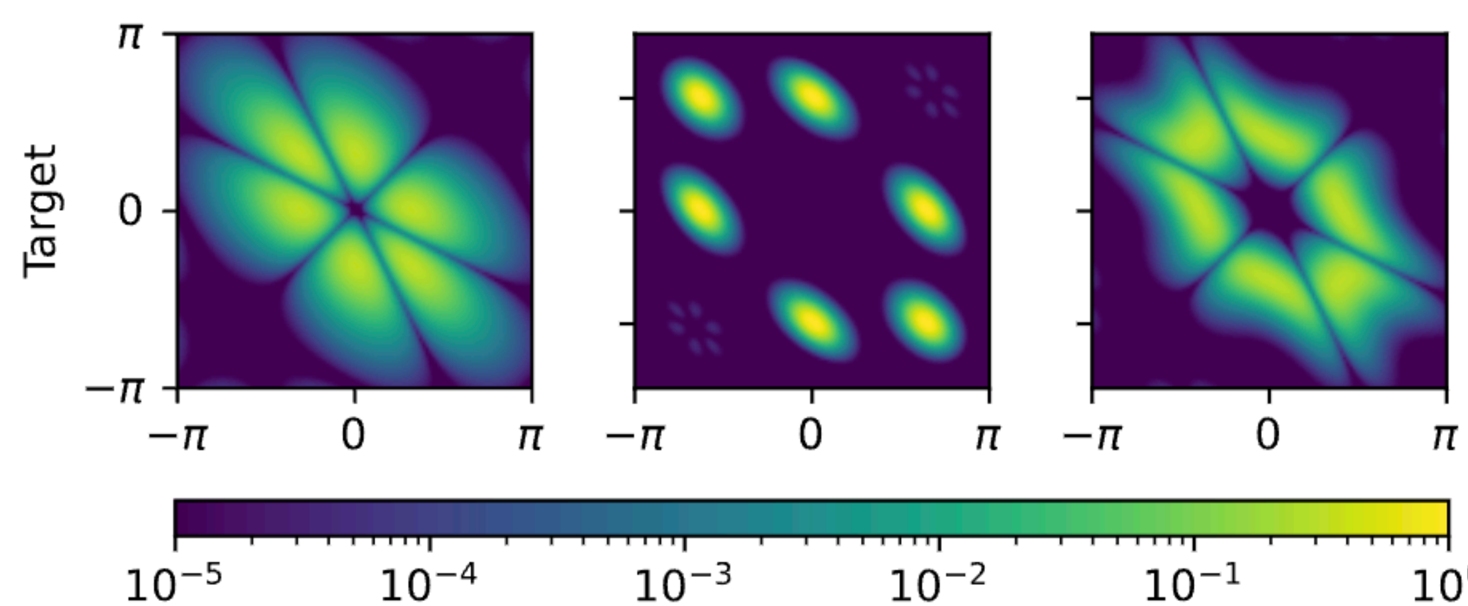
Simone Bacchio,[1] Pan Kessel,[2,3] Stefan Schaefer,[4] and Lorenz Vaitl[2]

General ML architecture?

Single edge

Conjugation-equivariant flow on SU(3)

$$S^{(i)}(U) = -\frac{\beta}{N}\,\mathrm{Re}\,\mathrm{tr}\left(c_n^{(i)} U^n\right)$$

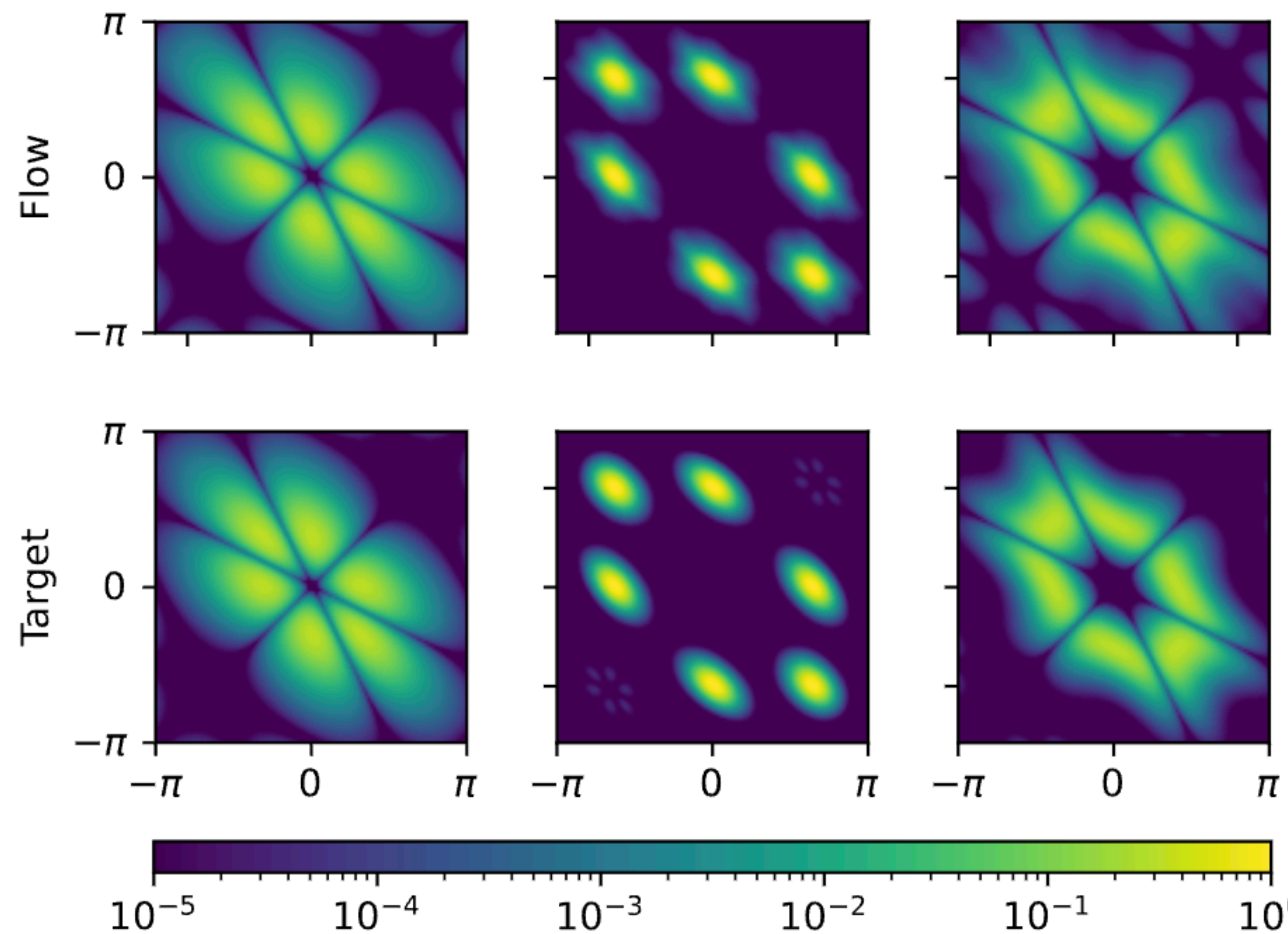


Single edge

Conjugation-equivariant flow on SU(3)

$$\dot{U} = \nabla P_\theta(t, U)$$

$$S^{(i)}(U) = -\frac{\beta}{N}\,\mathrm{Re}\,\mathrm{tr}\left(c_n^{(i)} U^n\right)$$



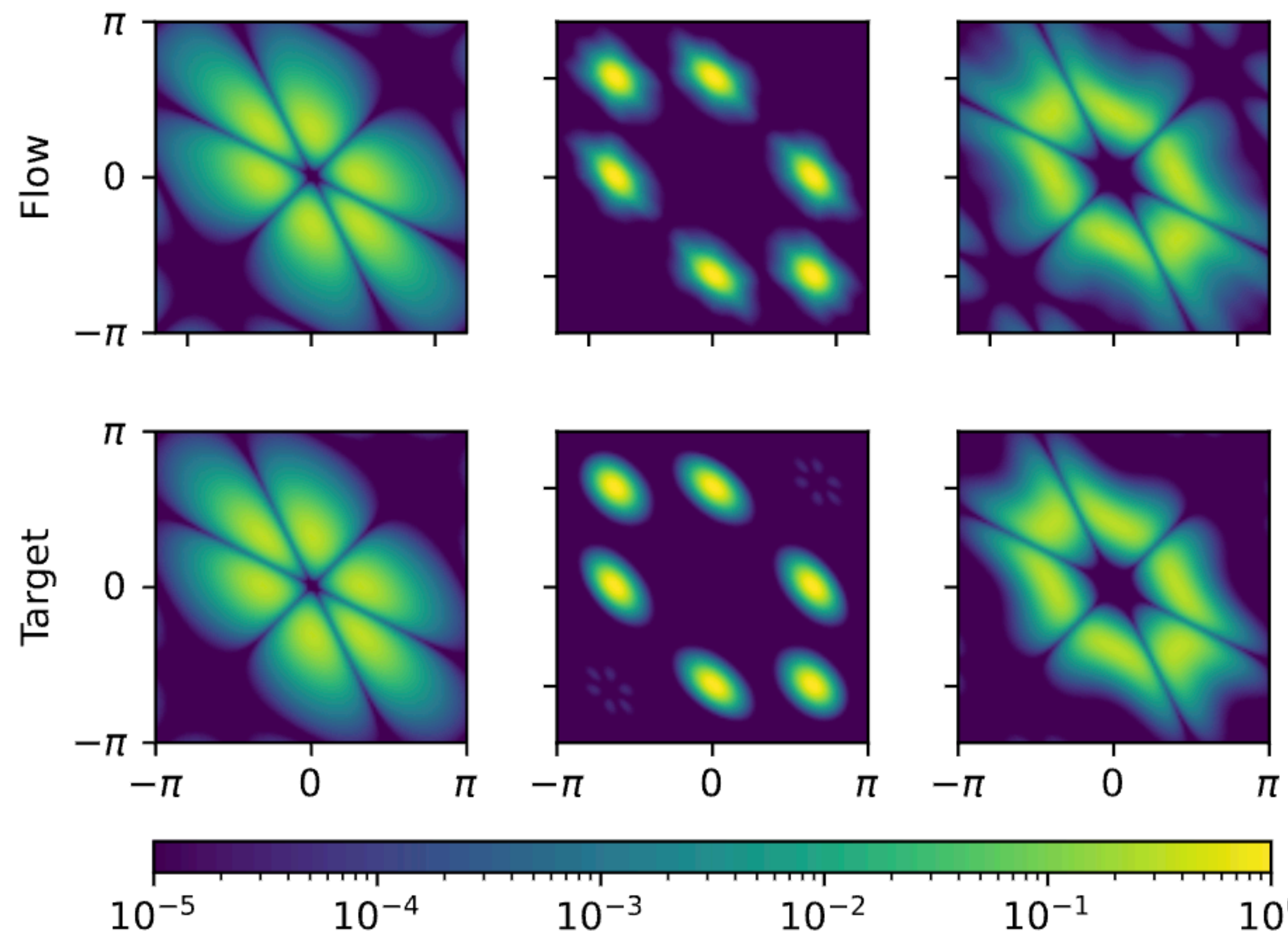
Single edge

Conjugation-equivariant flow on SU(3)

MLP based on traces of U^n

$$\dot{U} = \nabla P_\theta(t, U)$$

$$S^{(i)}(U) = -\frac{\beta}{N} \operatorname{Re} \operatorname{tr}\left(c_n^{(i)} U^n\right)$$



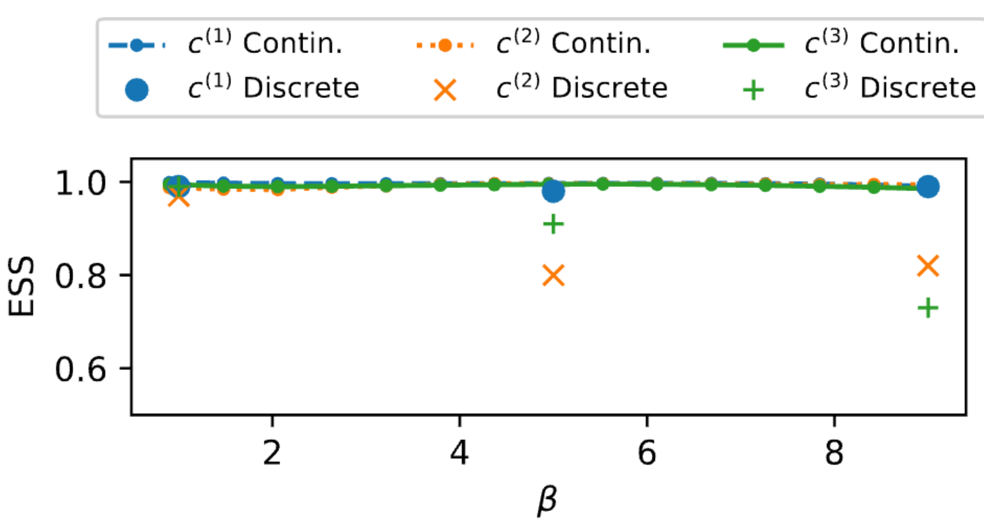
Single edge

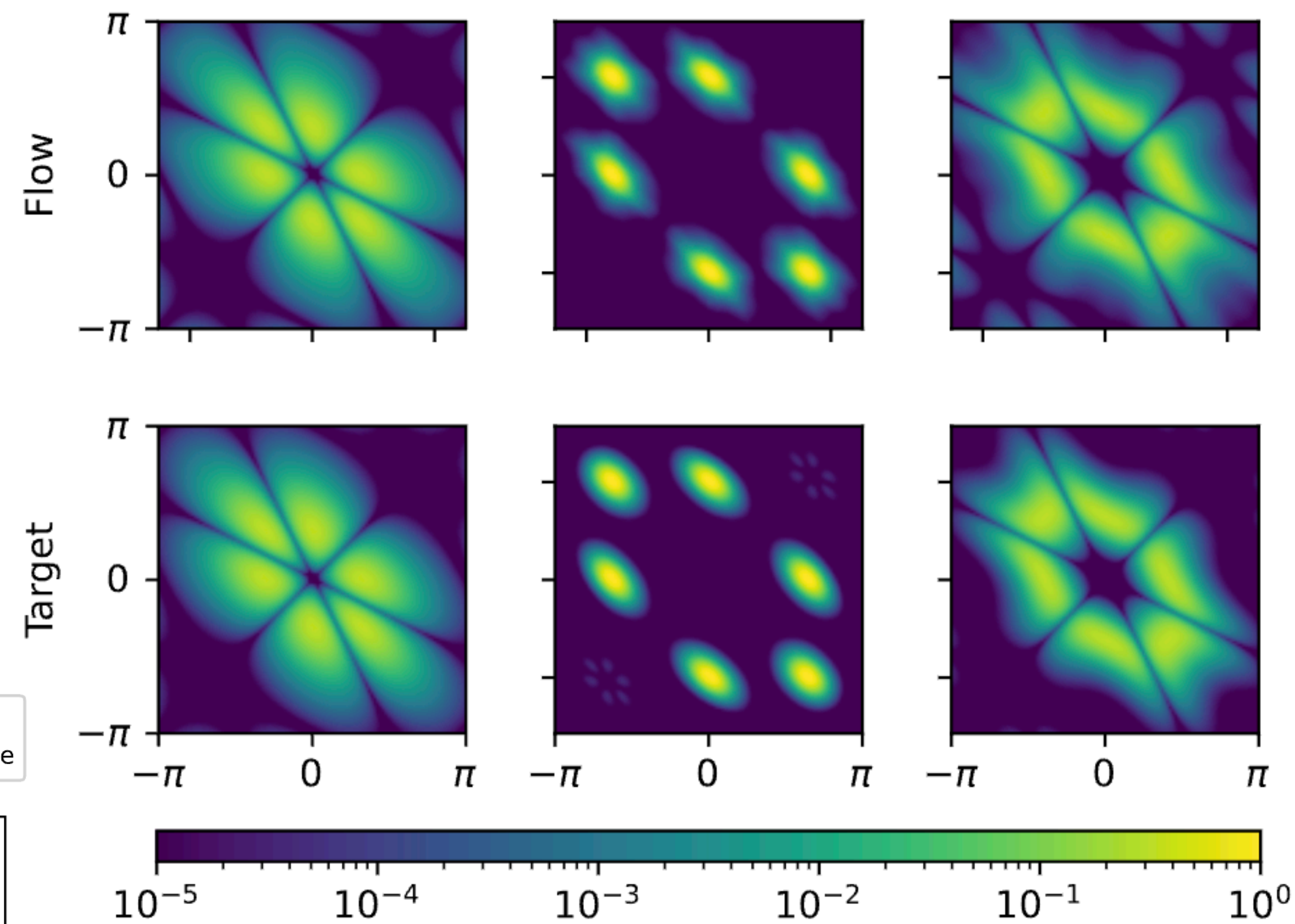
Conjugation-equivariant flow on SU(3)

MLP based on traces of U^n

$$\dot{U} = \nabla P_\theta(t, U)$$

$$S^{(i)}(U) = -\frac{\beta}{N} \operatorname{Re} \operatorname{tr} \left(c_n^{(i)} U^n \right)$$



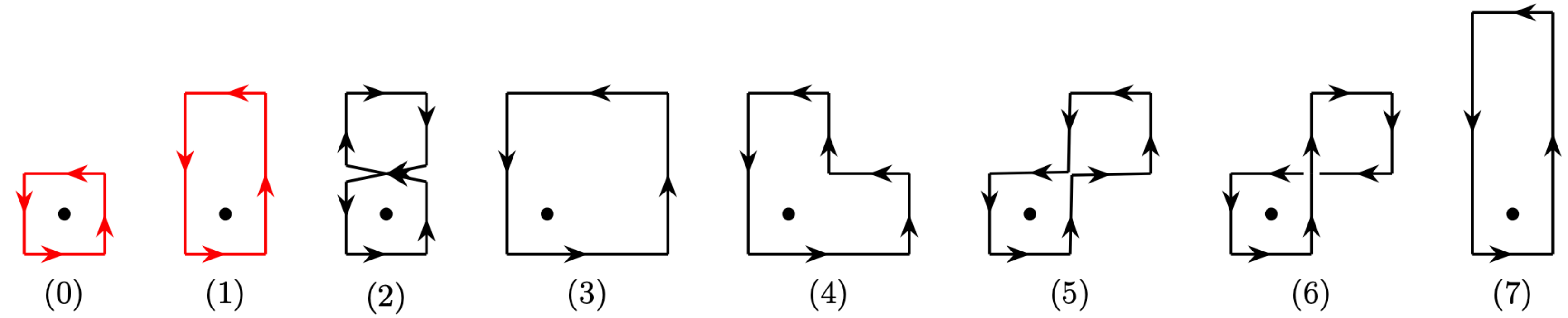


Network

Equivariant
vector field

$$A_e^a(U) =$$

Network

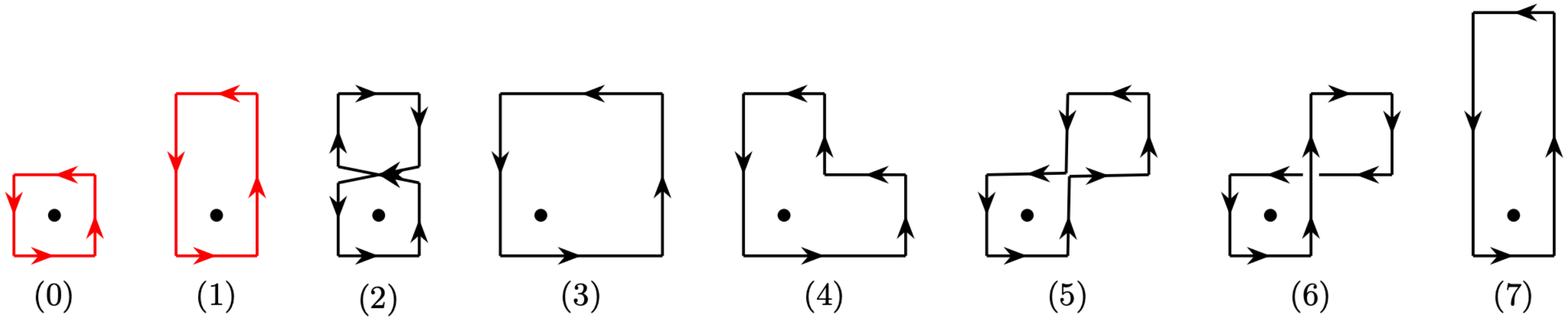


Equivariant vector field

"Basis" vectors: Built to be gauge equivariant

$$A_e^a(U) = \sum_{k,x} \partial^a_{U_e} W_x^{(k)}$$

Network



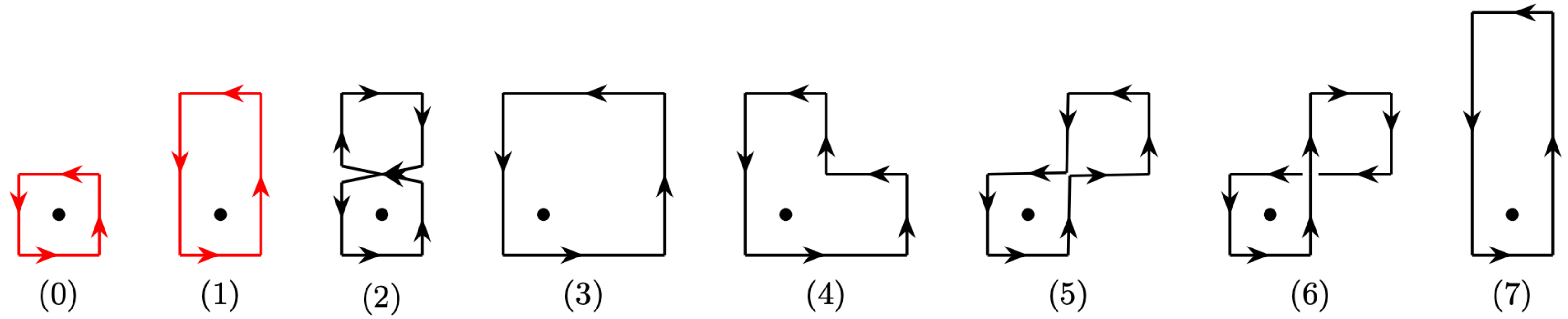
Equivariant vector field

"Basis" vectors: Built to be gauge equivariant

Superposition function: Built out of invariant quantities

$$A_e^a(U) = \sum_{k,x} \partial^a_{U_e} W_x^{(k)} \cdot \Lambda_x^k(W^{(1)}, W^{(2)}, \ldots)$$

Network



Equivariant vector field

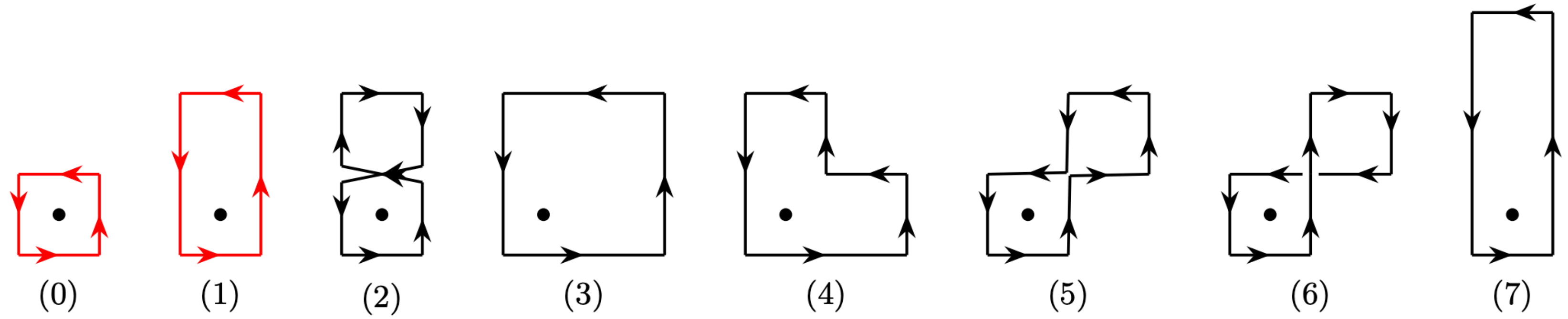
"Basis" vectors: Built to be gauge equivariant

Superposition function: Built out of invariant quantities

$$A_e^a(U) = \sum_{k,x} \partial^a_{U_e} W_x^{(k)} \cdot \Lambda_x^k(W^{(1)}, W^{(2)}, \dots)$$

- Divergence must not be too expensive.

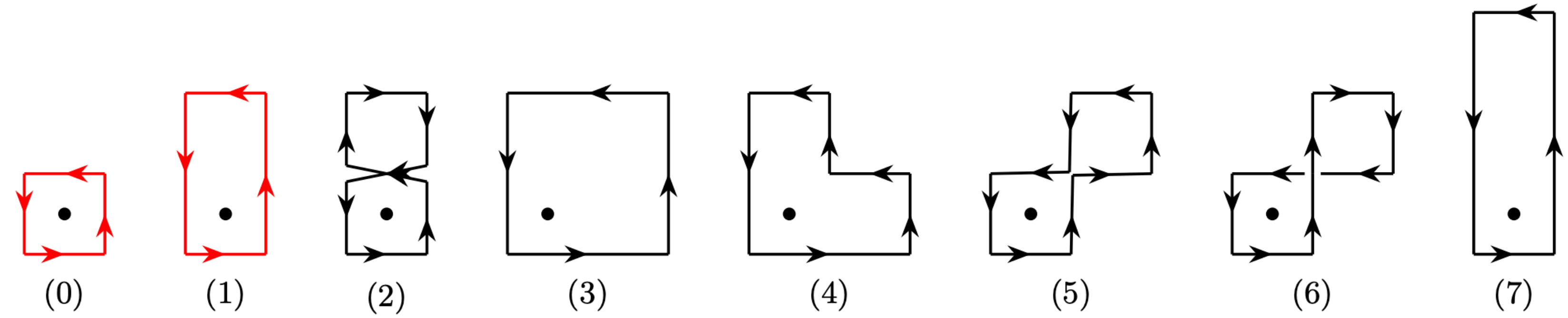
Network



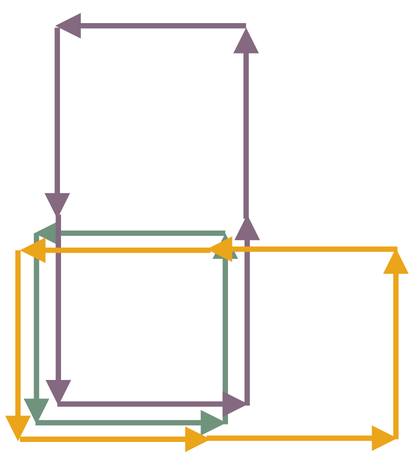
$$A_e^a(U) = \sum_{k,x} \partial^a_{U_e} W_x^{(k)} \cdot \Lambda_x^k(W^{(1)}, W^{(2)}, \ldots)$$

$$\Lambda_x^k = \sum_y C_{x,y}^{k,l} \, \mathrm{NN}_y^l(\{W_y^{(m)}\})$$

Network



$$A_e^a(U) = \sum_{k,x} \boxed{\partial^a_{U_e} W_x^{(k)}} \cdot \boxed{\Lambda_x^k(W^{(1)}, W^{(2)}, \dots)}$$

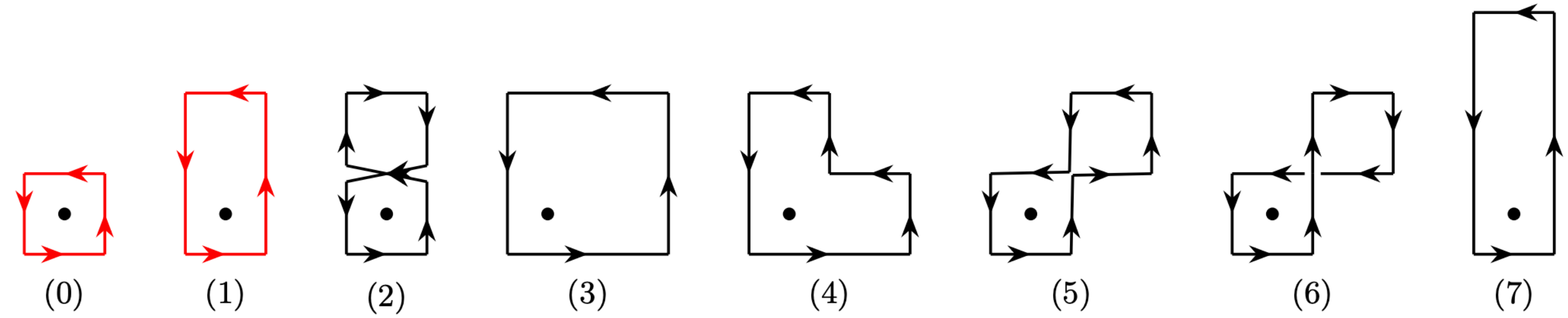


$W_x^{(k)}$

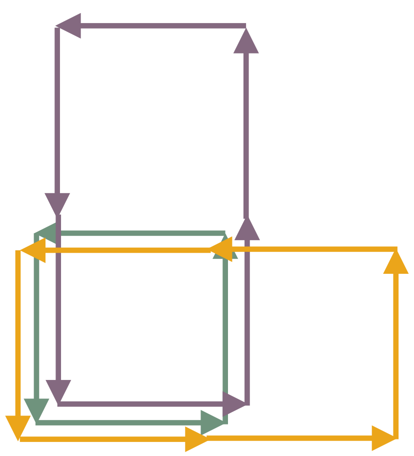
Local “stack” of Wilson loops

$$\Lambda_x^k = \sum_y C_{x,y}^{k,l}\, \mathrm{NN}_y^l(\{W_y^{(m)}\})$$

Network



$$A_e^a(U) = \sum_{k,x} \partial^a_{U_e} W_x^{(k)} \cdot \Lambda_x^k(W^{(1)}, W^{(2)}, \ldots)$$



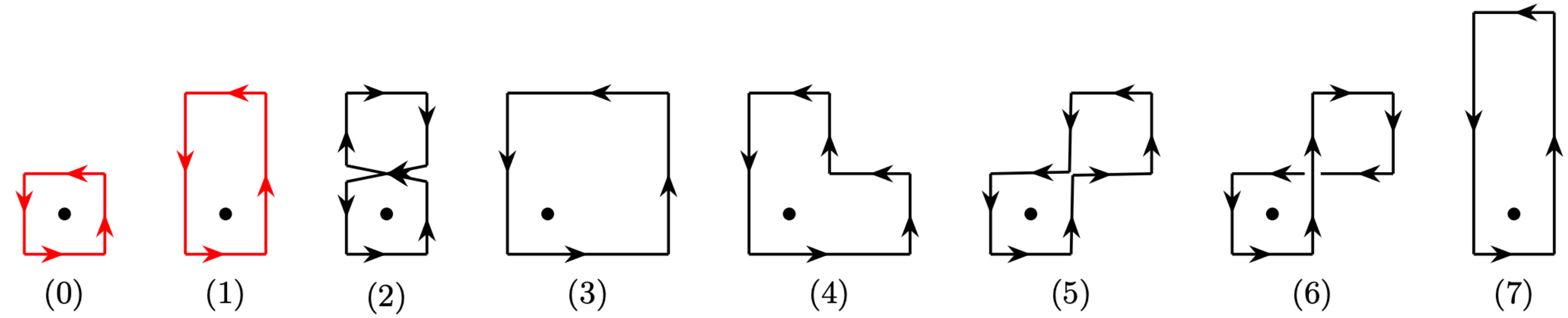
$$W_x^{(k)}$$

Local “stack” of Wilson loops

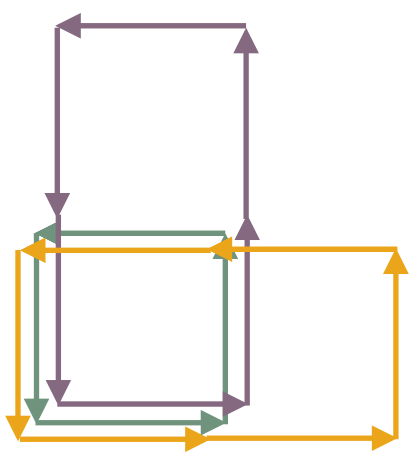
$$\Lambda_x^k = \sum_y C^{k,l}_{x,y}\, \mathrm{NN}^l_y(\{W^{(m)}_y\})$$

Arbitrary (non-linear) “local” neural network

Network



$$A_e^a(U) = \sum_{k,x} \boxed{\partial^a_{U_e} W_x^{(k)}} \cdot \boxed{\Lambda_x^k(W^{(1)}, W^{(2)}, \ldots)}$$

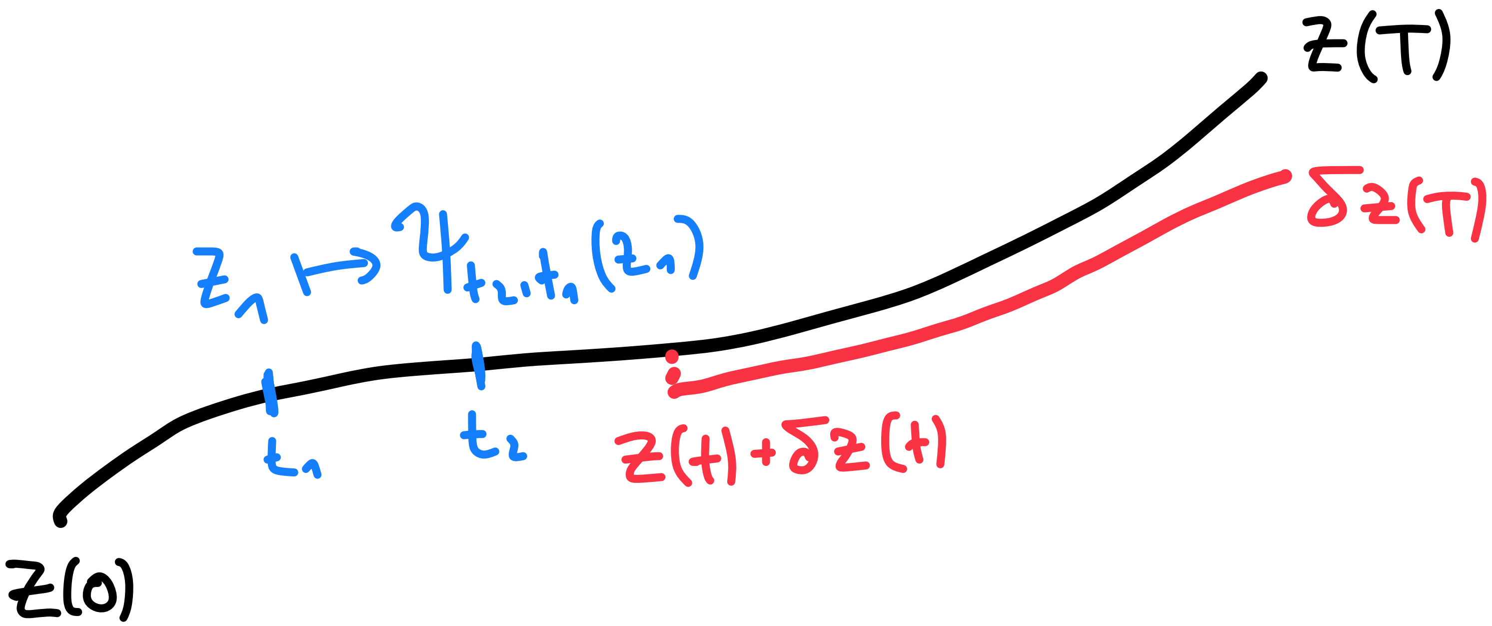


$$\Lambda_x^k = \sum_y C_{x,y}^{k,l} \, \mathrm{NN}_y^l(\{W_y^{(m)}\})$$

$W_x^{(k)}$ → Arbitrary (non-linear) “local” neural network → Convolution

Local “stack” of Wilson loops

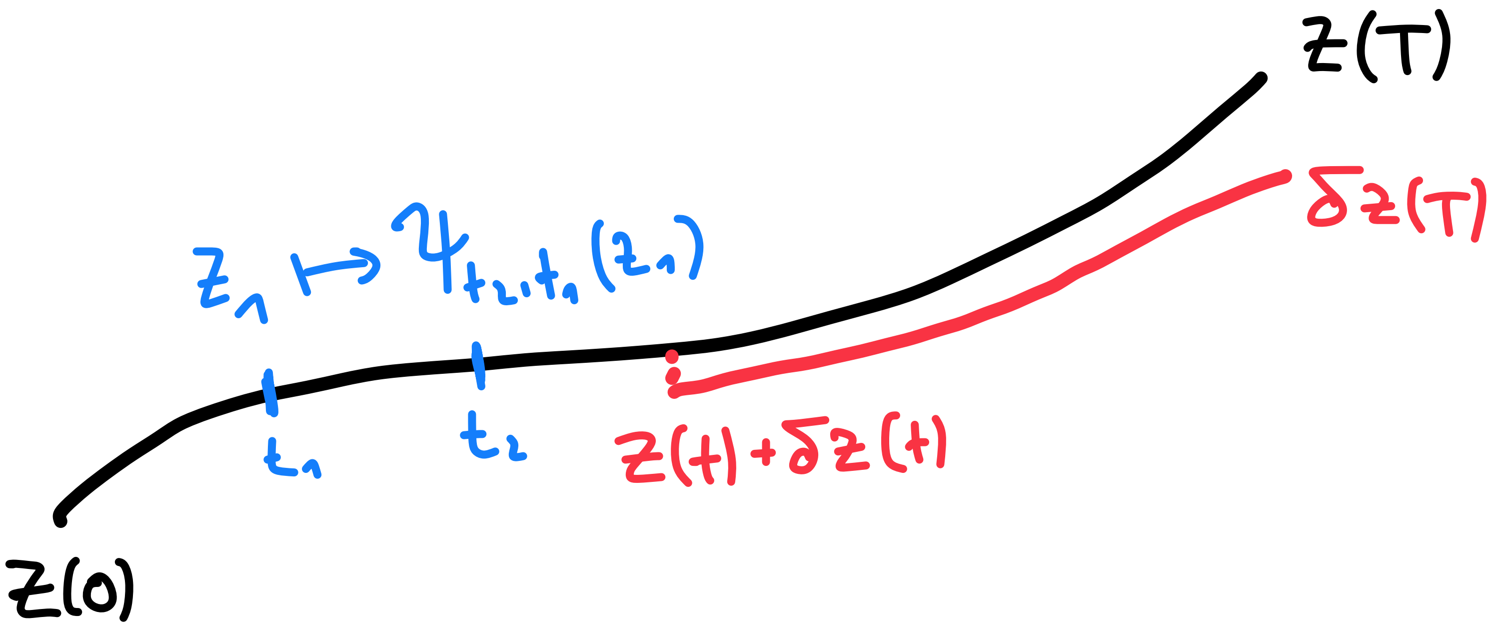
Adjoint sensitivity method



Continuous flow

ODE $\dot{z} = f_\theta(z, t)$

Adjoint sensitivity method

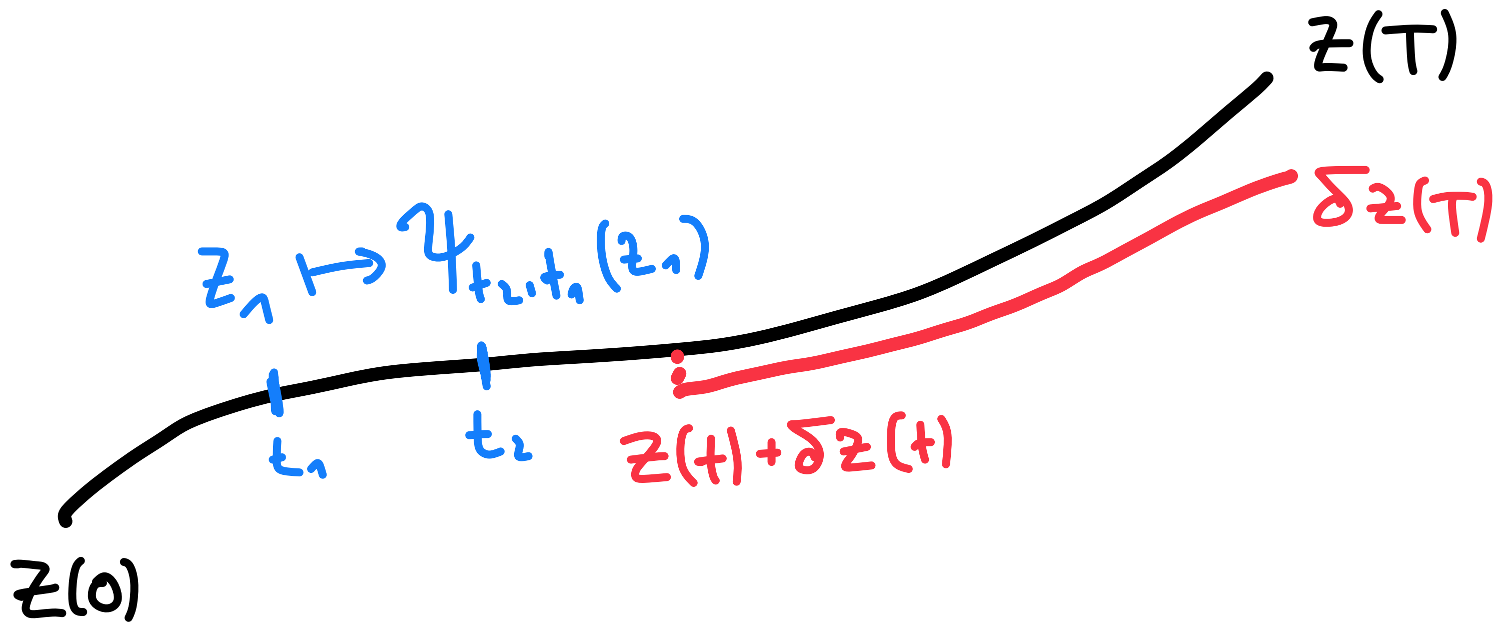


Continuous flow

ODE $\dot{z} = f_\theta(z, t)$

We have a loss function $L : M \to \mathbb{R}$, so $dL_z \in T_z^*M$

Adjoint sensitivity method



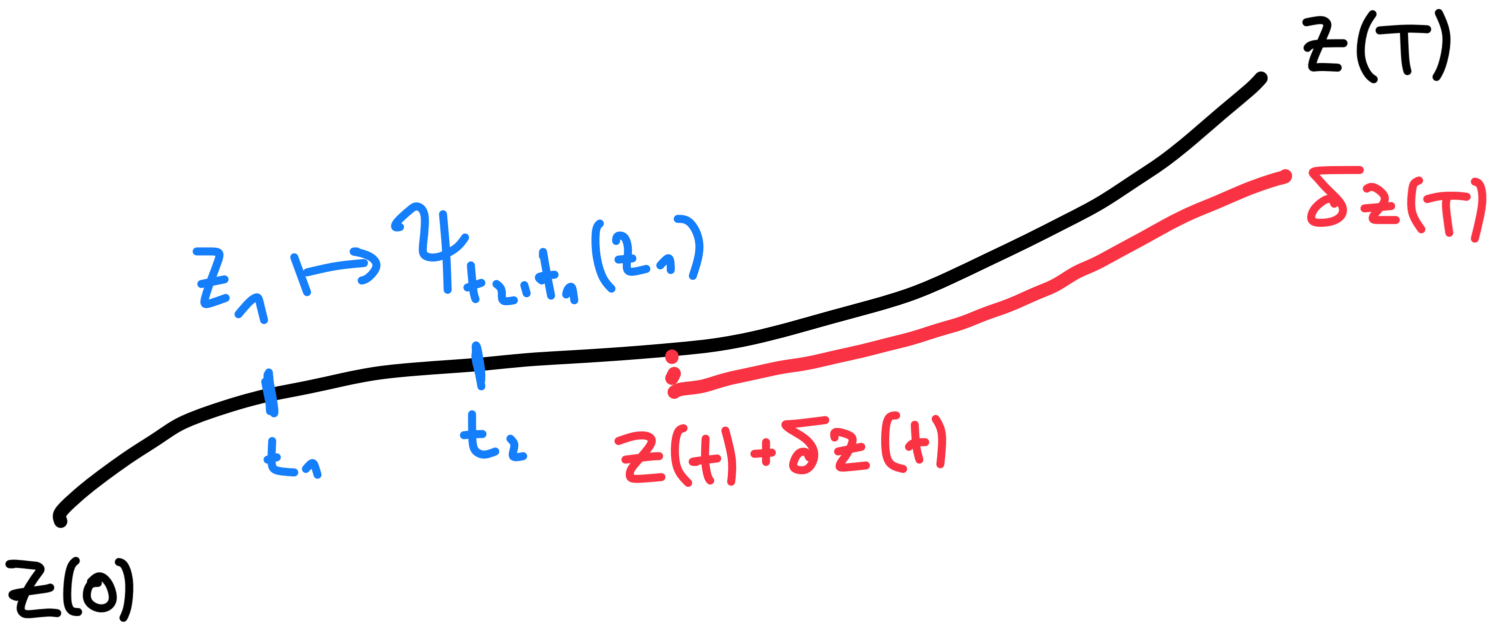
Continuous flow

ODE $\dot{z} = f_\theta(z, t)$

We have a loss function $L : M \to \mathbb{R}$, so $dL_z \in T_z^* M$

Adjoint state: $a(t) = \psi^*_{T,t}\, dL_{z(T)}$.

Adjoint sensitivity method



Continuous flow

ODE $\dot{z} = f_\theta(z, t)$

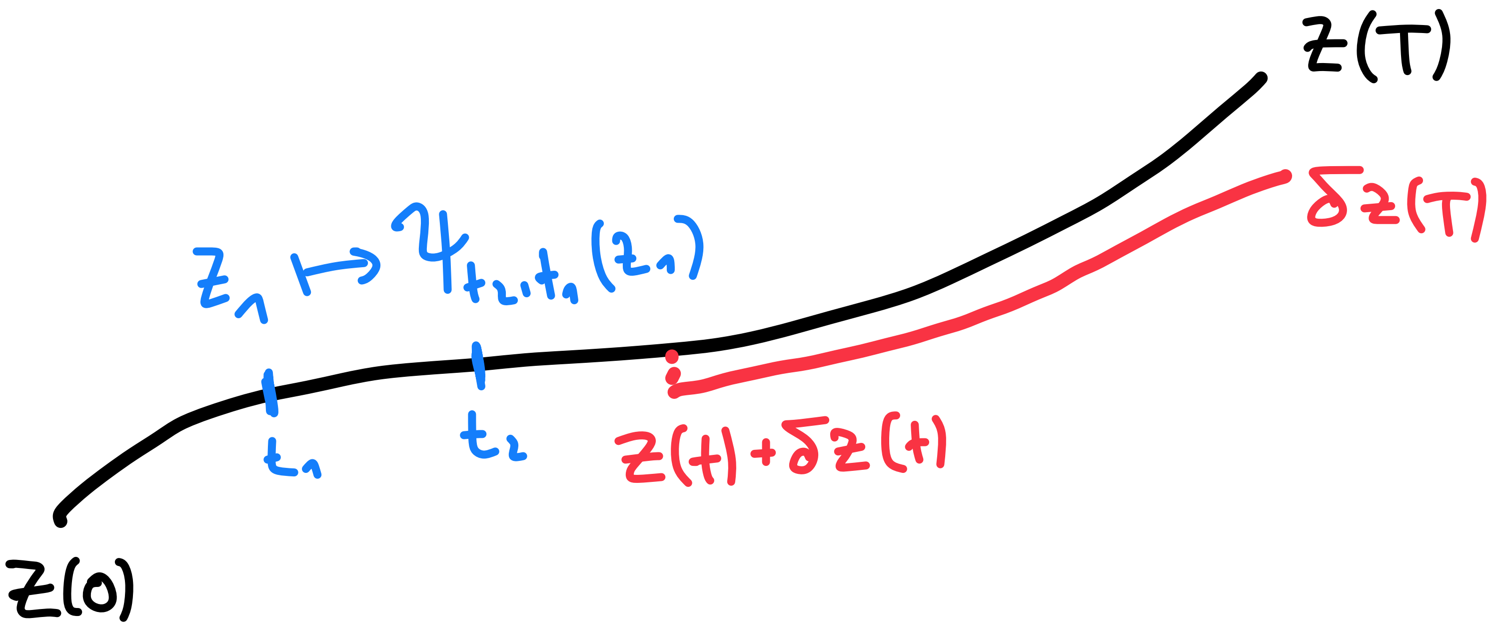
We have a loss function $L : M \to \mathbb{R}$, so $dL_z \in T^*_z M$

Adjoint state: $a(t) = \psi^*_{T,t}\, dL_{z(T)}\,.$

"Compute gradients by back-integrating"

$$\frac{da(t)}{dt} = -\,a(t)\frac{\partial f_\theta(z,t)}{\partial z} \qquad \frac{dL}{d\theta} = -\int_T^0 a(t)\frac{\partial f(z,t)}{\partial \theta}\,dt$$

Adjoint sensitivity method



Continuous flow

ODE $\dot{z} = f_\theta(z, t)$

We have a loss function $L : M \to \mathbb{R}$, so $dL_z \in T_z^* M$

Adjoint state: $a(t) = \psi_{T,t}^* \, dL_{z(T)}$.

"Compute gradients by back-integrating"

$$\frac{da(t)}{dt} = -\, a(t) \frac{\partial f_\theta(z,t)}{\partial z} \qquad \frac{dL}{d\theta} = - \int_T^0 a(t) \frac{\partial f(z,t)}{\partial \theta} \, dt$$

$$\dot{U}(t) = Z_\theta(U, t) U(t) \qquad \dot{A} = [Z, A] + \nabla(\nabla \cdot \dot{U}) U^{-1} - \sum_a A_a (\nabla Z^a) U^{-1}$$

Results

16×16

ESS [%]	SU(2) $\beta = 2.2$	SU(2) $\beta = 2.7$	SU(3) $\beta = 5$	SU(3) $\beta = 6$	SU(3) $\beta = 8$
Continuous flow	**87**	**68**	86	**76**	**23**
Bacchio et al [13]	–	–	**88**	70	–
Boyda et al [8]	80	56	75	48	–

Results

16×16

ESS [%]	SU(2) $\beta = 2.2$	SU(2) $\beta = 2.7$	SU(3) $\beta = 5$	SU(3) $\beta = 6$	SU(3) $\beta = 8$
Continuous flow	**87**	**68**	86	**76**	**23**
Bacchio et al [13]	–	–	**88**	70	–
Boyda et al [8]	80	56	75	48	–

8×8

ESS [%]	SU(3) $\beta = 8$	SU(3) $\beta = 12$
Continuous flow	**64**	**27**
Multiscale + flow [23]	35	13
Haar + flow [23]	25	3

Takeaways

Takeaways

- More general architecture improves sample quality

Takeaways

- More general architecture improves sample quality
- Tractable divergence constraint still limits architecture, complicates implementation

Takeaways

- More general architecture improves sample quality
- Tractable divergence constraint still limits architecture, complicates implementation
- Incorporating spatial rotation/mirror symmetries not yet implemented

Takeaways

- More general architecture improves sample quality
- Tractable divergence constraint still limits architecture, complicates implementation
- Incorporating spatial rotation/mirror symmetries not yet implemented
- General framework for normalising flows in JAX

Continuous normalizing flows for lattice gauge theories

Mathis Gerdes*
Institute of Physics, University of Amsterdam

Pim de Haan
CuspAI

Roberto Bondesan
Department of Computing, Imperial College London

Miranda C. N. Cheng†
Institute of Physics, University of Amsterdam
Institute for Mathematics, Academia Sinica, Taiwan and
Korteweg-de Vries Institute for Mathematics, University of Amsterdam

cusp.ai