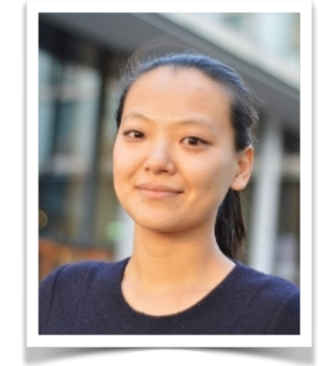
University of Amsterdam

with Pim de Haan,
Roberto Bondesan &
Miranda Cheng

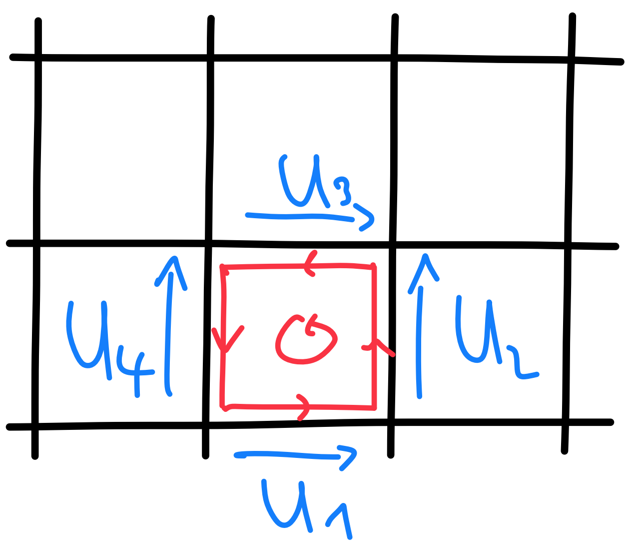
Continuous flows for gauge theories

[arxiv:2410.1316]

Mathis Gerdes — m.gerdes@uva.nl | Taipei 2024

Lattice gauge theory

Wilson action



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

Wilson loop trace

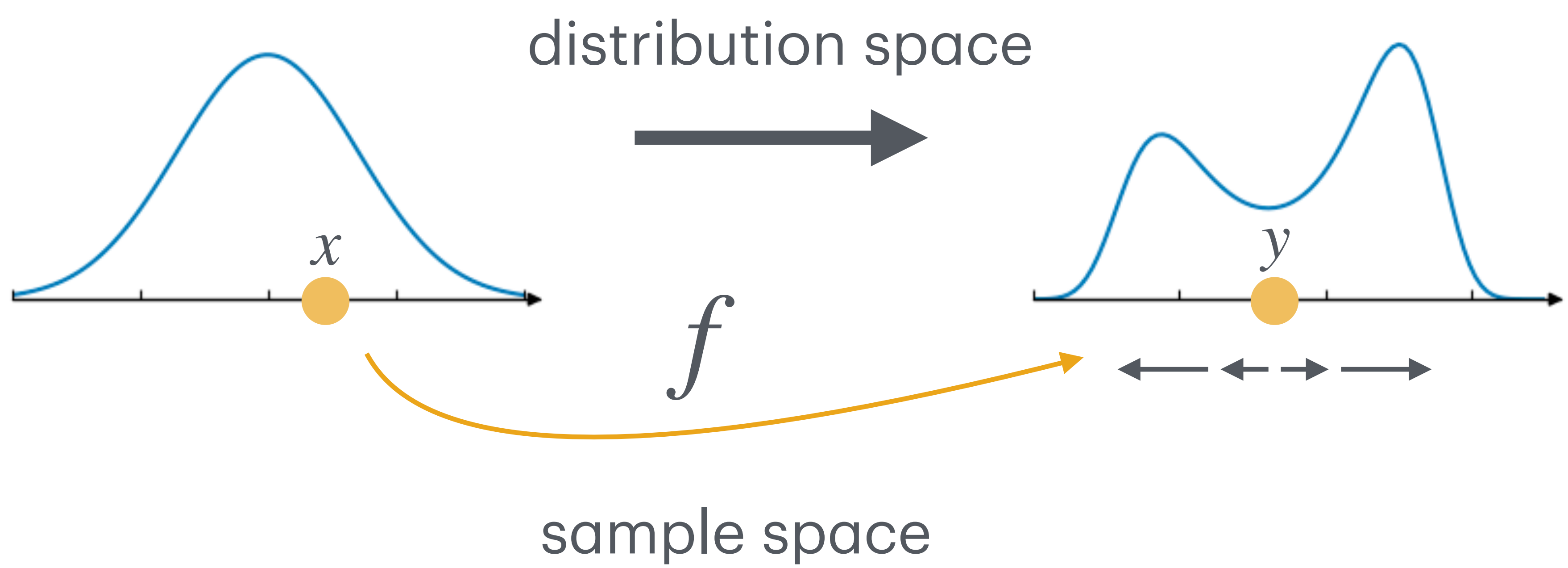
$$W = \mathrm{tr}(U_1 U_2 U_3^\dagger U_4^\dagger)$$

⟶ Want to sample $U \in SU(N)^{|E|}$

$$U \sim e^{-S[U]}$$

Change of variables

Transforming probability densities



Change of density

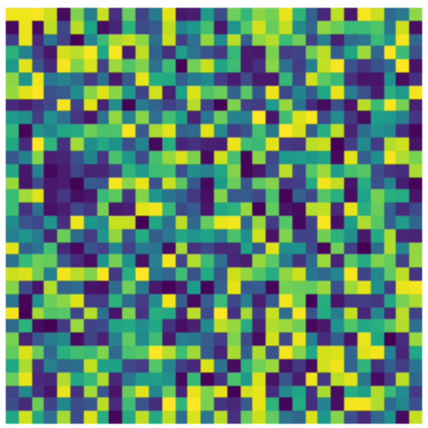
$$p(y) = p\left(f^{-1}(y)\right) \cdot \left|\det \frac{\partial f}{\partial x}\right|^{-1}$$

Source point

Normalizing flows

Learning f

trivial theory

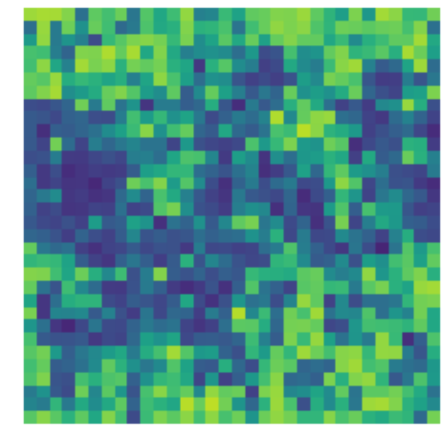


$\mathcal{N}$

bijection f

"Normalizing flow"

interacting theory



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

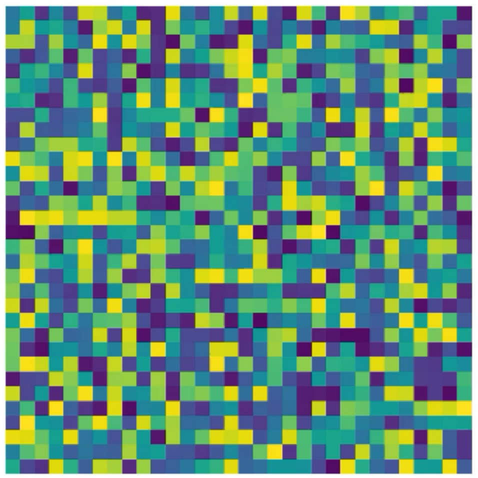
We want to ***learn*** a *trivializing map* f.

To compute model probability:

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

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

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:

$$\frac{d}{dt}\log p(\phi) = -\nabla \cdot \dot{\phi}$$

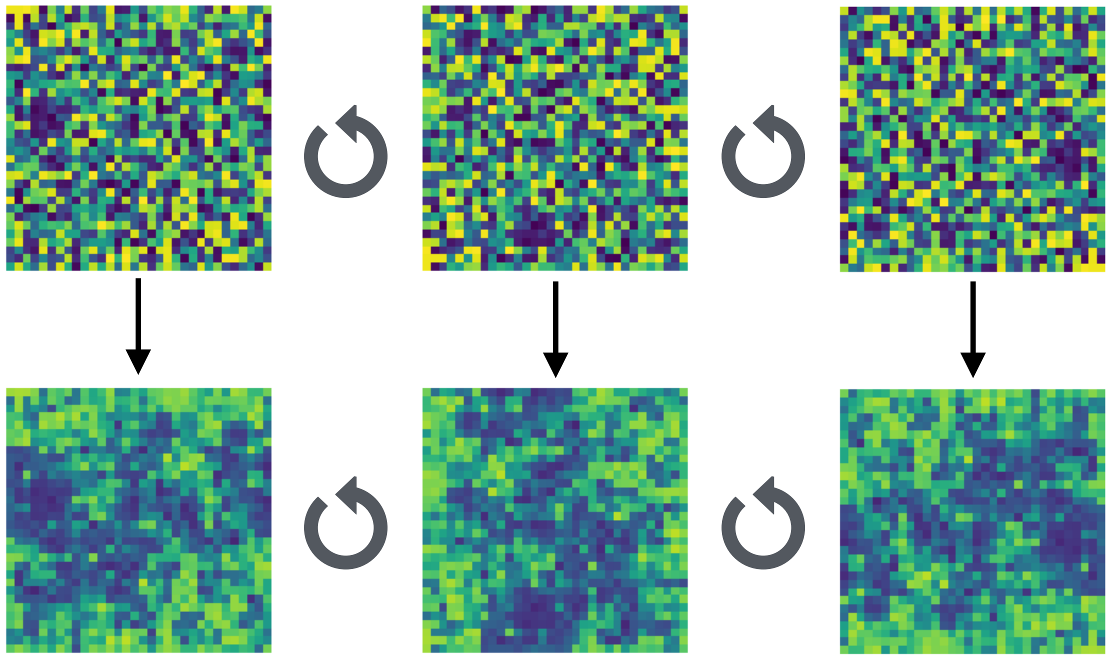
Symmetries

And the need for equivariant flows

If action is invariant under transformation $S(\phi) = S(g \cdot \phi)$

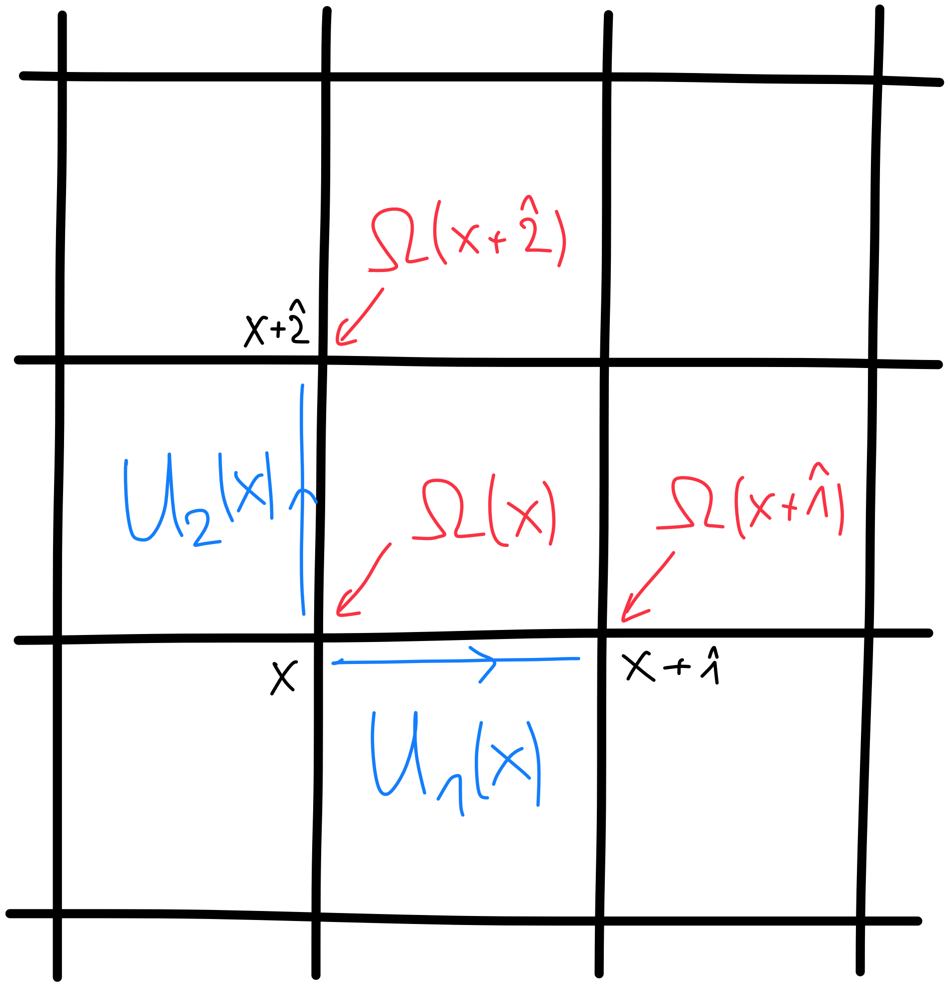
then $p(\phi) = p(g \cdot \phi)$, should be proposed equally likely.

$$f_\theta(g \cdot \phi) = g \cdot f_\theta(\phi)$$



etc.

Gauge symmetry



Under gauge symmetry links transform as

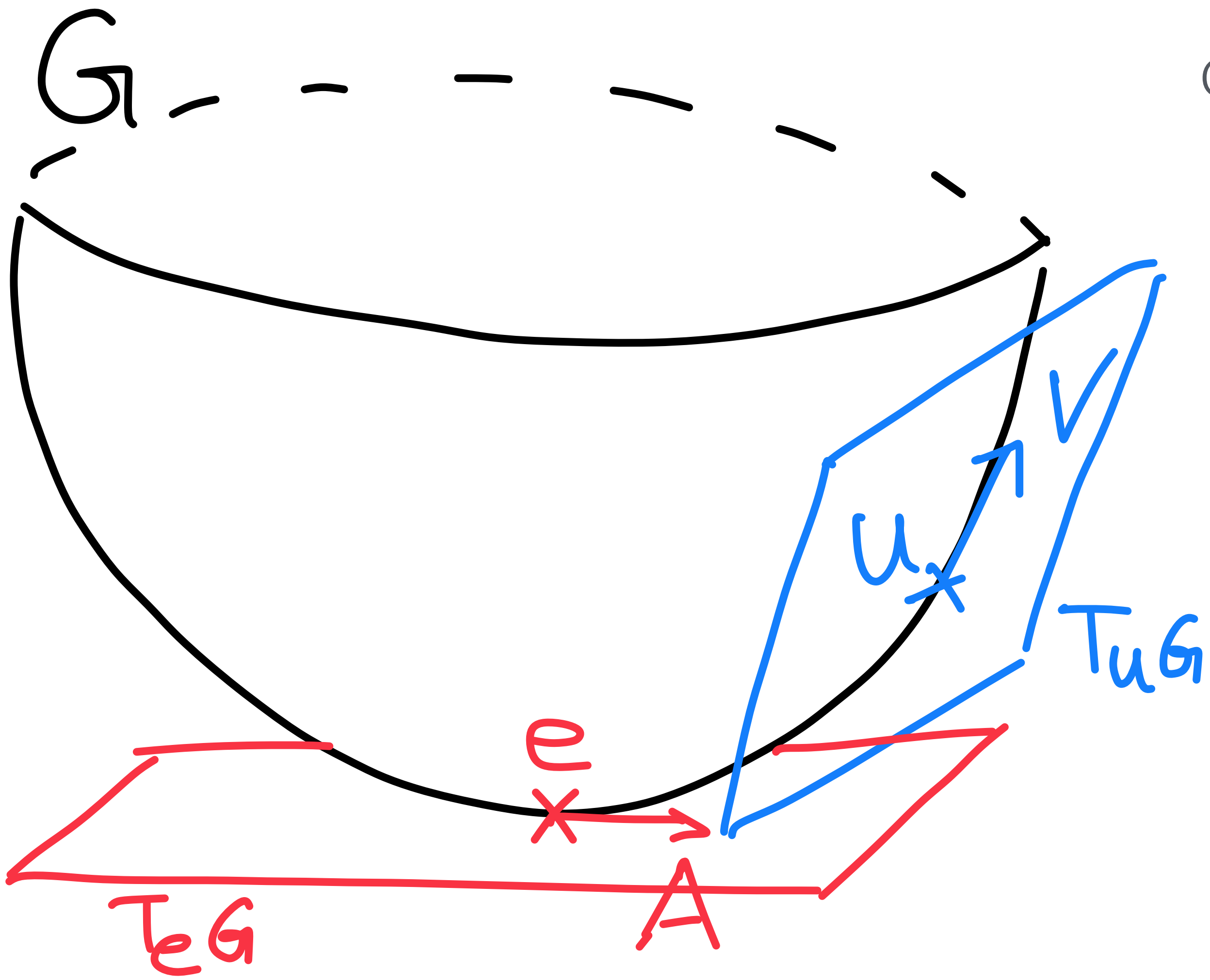
$$U_\mu(x) \mapsto \Omega(x)\, U_\mu(x)\, \Omega(x+\hat{\mu})$$

How can we define gauge equivariant transformations?

Continuous flows for gauge theories

Lie groups

A brief reminder



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

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

$$V = AU$$

Transporting A to vector space at U

Lie algebra is spanned by generators T^a

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

Challenges

Training gauge-neural ODEs

Define equivariant
$\dot{U} = Z_\theta(U, t)U$

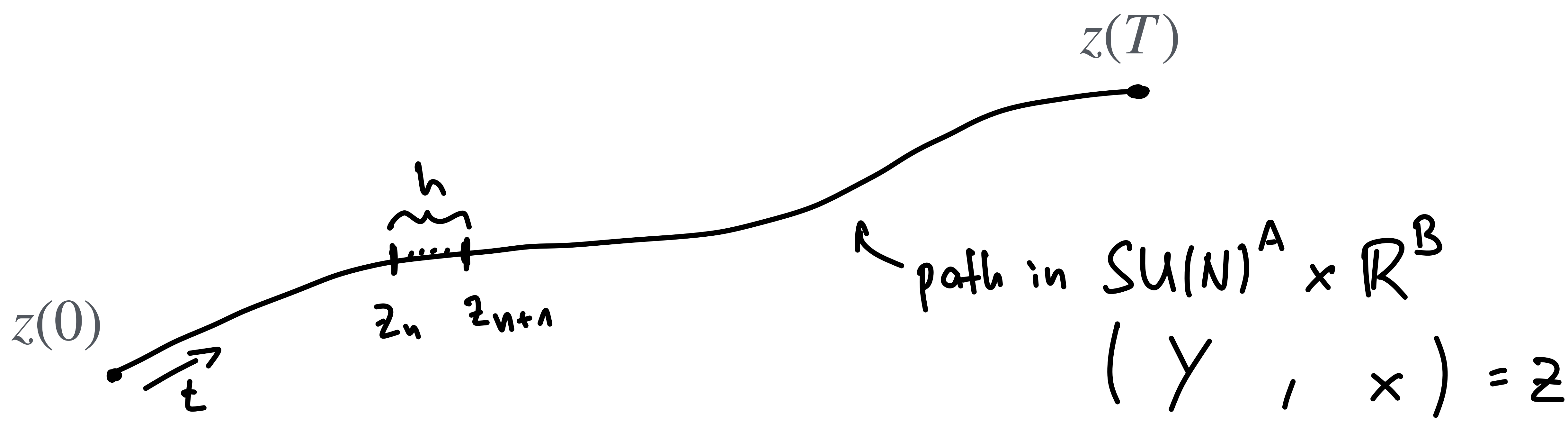
Solve ODE & gradients
$\partial_\theta L_{KL}[U(T)]$

Efficient divergence
$\partial_a Z^a(U, t)$

Actually $L[p(U(T))]$ $\quad \frac{d}{dt} \log p(U) = -\partial_a Z^a$

Crouch-Grossmann

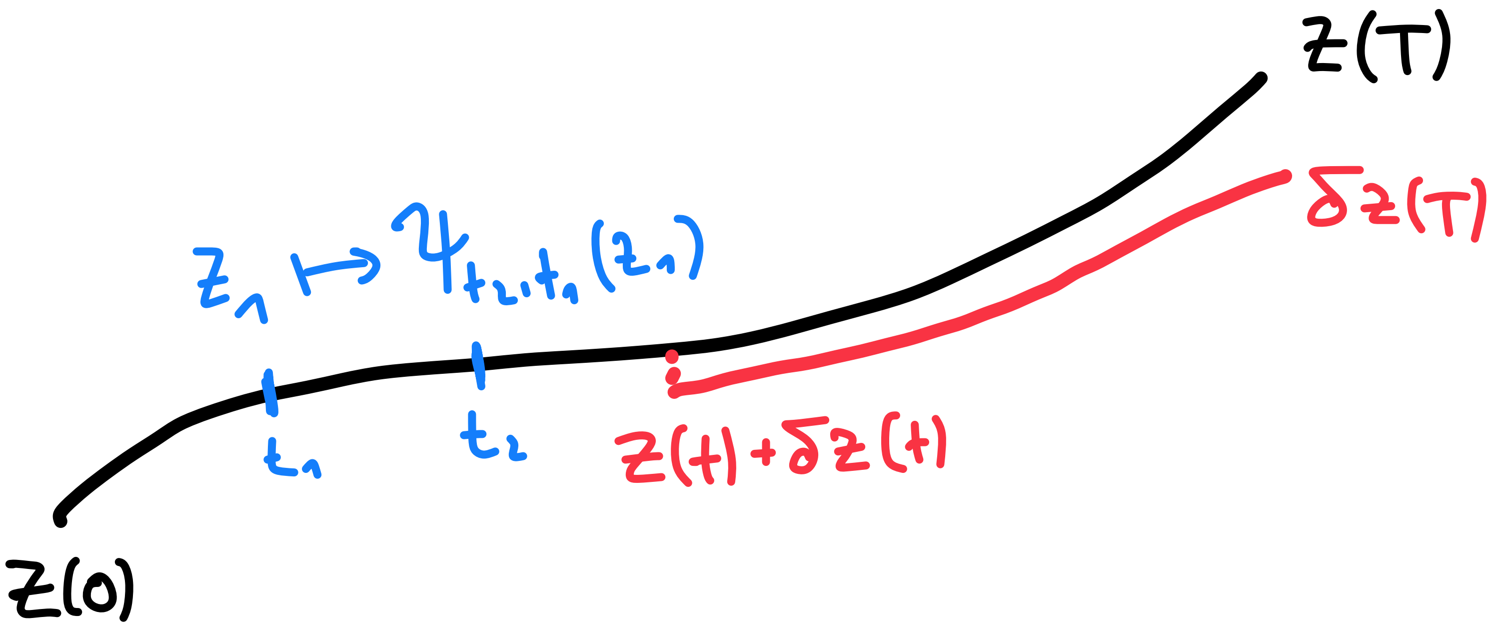
Discretize integration



Real $$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i f(x_i^{(n)}, Y_i^{(n)})$$

Matrix Lie group $$Y_{n+1} = \exp\left(h b_s Z_s^{(n)}\right) \cdots \exp\left(h b_1 Z_1^{(n)}\right) Y_n$$

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

In words: maps $\delta z(t)$ to δL.

"Compute gradients by back-integrating"

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

Defining an ODE

Continuous flows for SU(N)

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

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

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

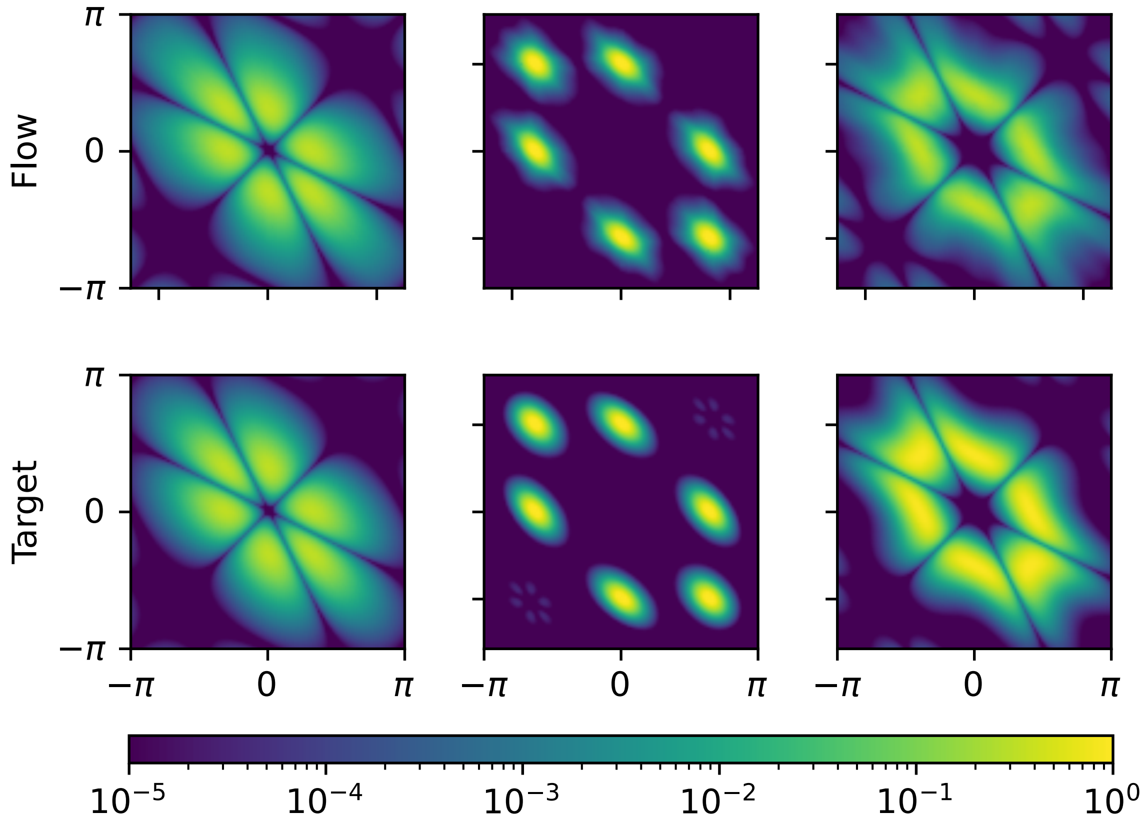
To define our flow, the network should output an algebra element:

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

General implementation

Thanks to JAX

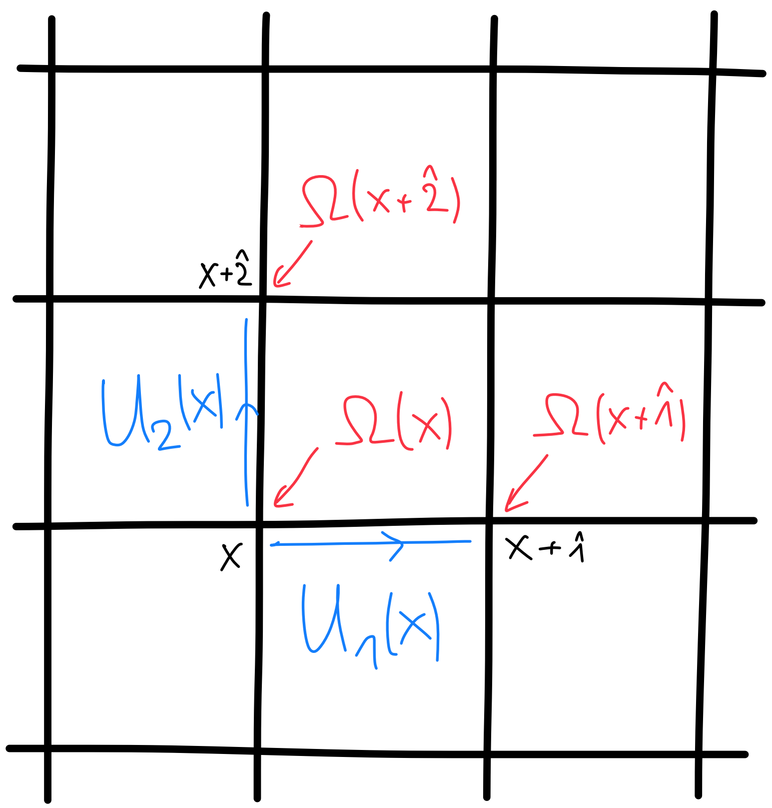
- Integration & gradients for any "real × matrix" Lie group d.o.f.
- Test on single $SU(N)$: targets $p(U) = p(V^\dagger U V)$.
- Define $Z^a = \partial_a \Phi_\theta(U, t)$ & use autograd.



Gauge symmetry

Object transformations

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



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**.

Gradients of invariants

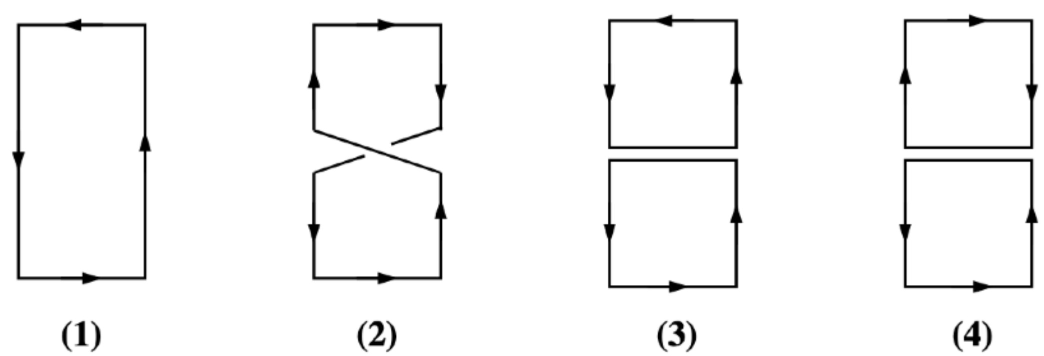
e.g. $V = \nabla_U W$ are **equivariant**

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

Continuous flows for SU(N)

Gradient flows

Define $Z^a = \partial^a S$ as the gradient of potential:
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

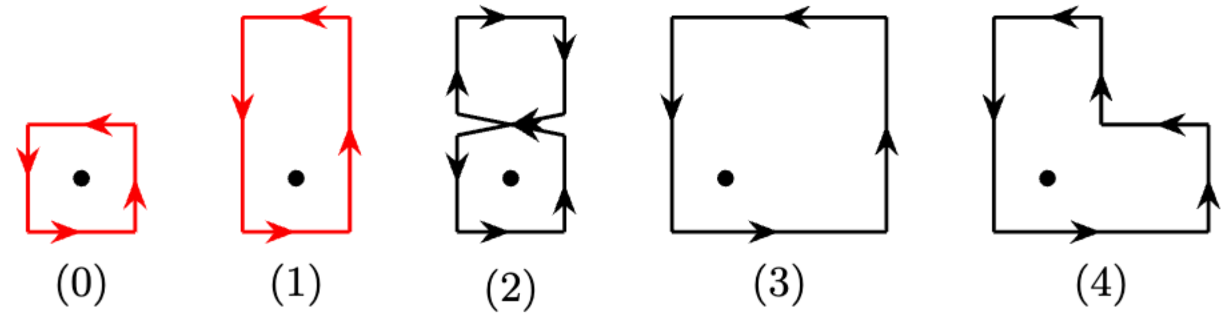
Learning Trivializing Gradient Flows for Lattice Gauge Theories

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

Can we define a more general ML architecture?

Network

Idea for construction



Equivariant vector field

"Basis" vectors: Built to be gauge equivariant

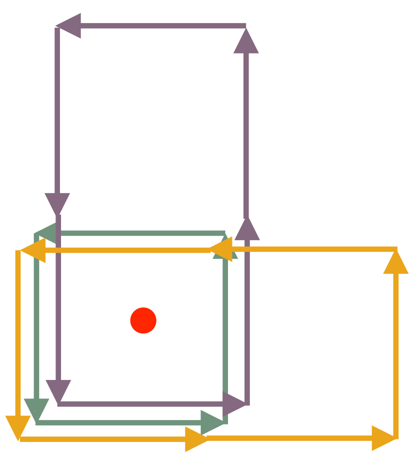
Superposition function: Built out of invariant quantities

$$Z_e^a(U) = \sum_{k,\bar{x}} \partial_{e,a} W_{\bar{x}}^{(k)} \cdot S_{\bar{x}}^k(W^{(1)}, W^{(2)}, \ldots)$$

Network

Idea for construction

$$Z_e^a(U) = \sum_{k,\bar{x}} \partial_{e,a} W_{\bar{x}}^{(k)} \cdot S_{\bar{x}}^k(W^{(1)}, W^{(2)}, \ldots)$$



$W_{\bar{x}}^{(k)}$

$$S_{\bar{x}}^k = \sum_{\bar{y},l} C_{\bar{x}\bar{y}}^{k,l} \, \mathrm{NN}_{\bar{y}}^l(\{W_{\bar{y}}^{(m)}\})$$

Local “stack” of Wilson loops → Non-linear *local* neural network → (Equivariant) Convolution

Divergence computation

$$\sum_{e,a} \partial_{a,e} Z_e^a(U,t)$$

JVP forward-mode

$$v \mapsto Df \cdot v$$

VJP backward-mode

$$w^\dagger \mapsto w^\dagger \cdot Df$$

$$\partial_i Z^i = \hat{e}_i^\dagger (DZ) \hat{e}_i$$

$\longrightarrow$ scales with extra $|E|$.

Divergence computation

$$\sum_{e,a} \partial_{a,e} Z_e^a(U,t)$$

$$= \sum_{k,\bar{x}} \partial_{e,a}^2 W_{\bar{x}}^{(k)} \cdot S_{\bar{x}}^k(\{W\}) + \partial_{e,a} W_{\bar{x}}^{(k)} \cdot \partial_{e,a} S_{\bar{x}}^k(\{W\})$$

$$\partial_e^a S_e^k = \sum_{l,x} C_{e,x}^{k,l} D(\mathrm{NN}_x^l)(\{\partial_e^a W_x^{(m)}\})$$

$\longrightarrow$ start with $\partial_{e,a} W$ and apply forward mode!

Divergence computation

$$Z_e^a(U) = \sum_{k,\bar{x}} \partial_{e,a} W_{\bar{x}}^{(k)} \cdot S_{\bar{x}}^k(W^{(1)}, W^{(2)}, \ldots)$$

```
conv = algebra.Convolution(vec_count, kernel_size)
superpos = conv(trace_stack.no_first_grad(), t_emb)
potential = superpos @ trace_stack
# take first & second derivative
vect, div = potential.sum_all_appearances().vect_div_origin(lat_dim)
return vect, div
```

$\longrightarrow$ start with $\partial_{e,a}W$ and apply forward mode!

Results SU(2), SU(3)

In two dimensions

16×16

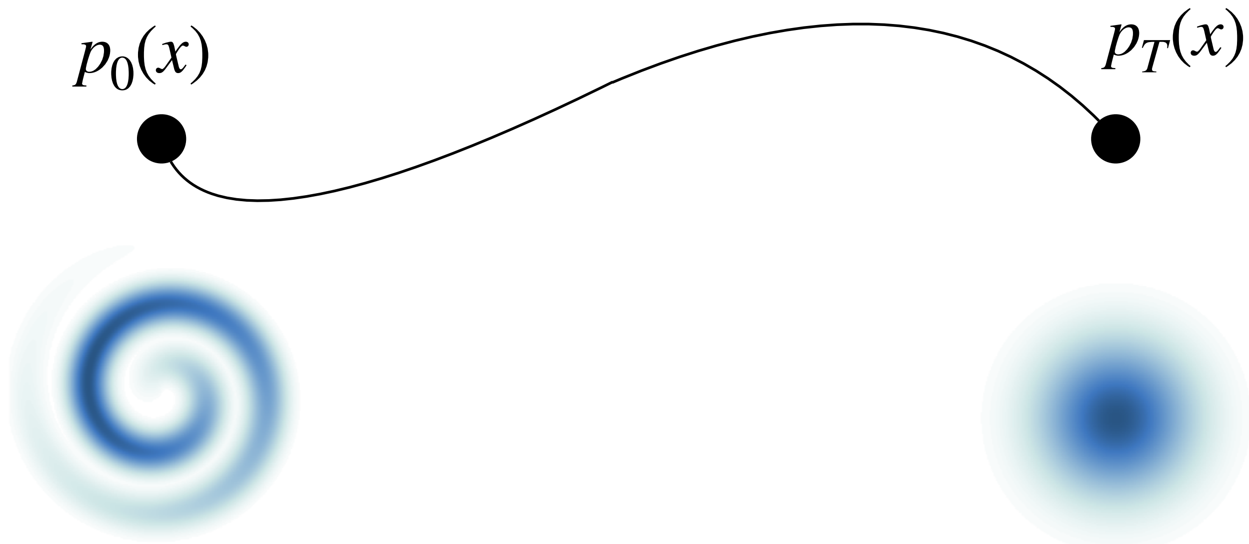
ESS [%]	SU(2) $\beta = 2.2$	SU(2) $\beta = 2.7$	SU(3) $\beta = 5$	SU(3) $\beta = 6$	SU(3) $\beta = 8$
Continuous flow	**87**	**68**	86	**76**	**23**
Bacchio et al [13]	–	–	**88**	70	–
Boyda et al [8]	80	56	75	48	–

8×8

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

- Shallow ResNet activation.
- Transform from Haar measure.
- 2nd order integrator.
- Switch to 64bit after some training.
- Standard reverse KL loss.

Paths in distribution space



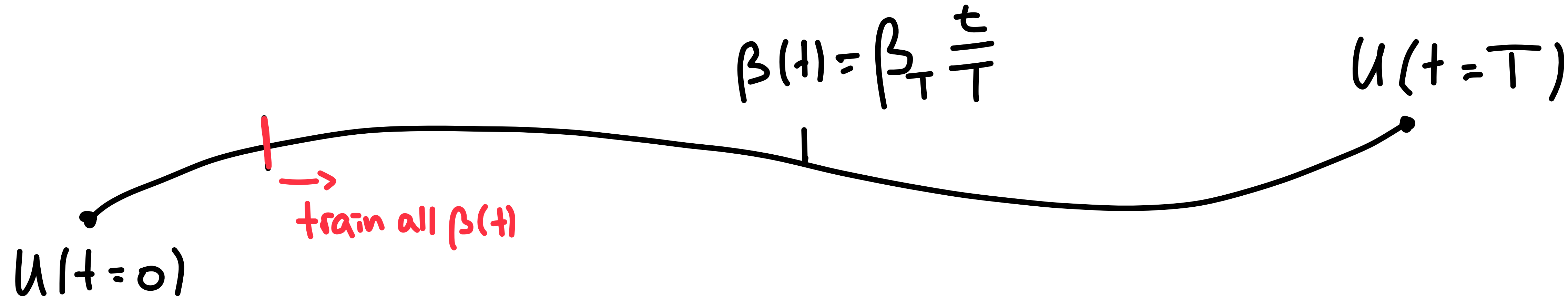
Path in distributions

drives
(Fokker-Planck
equation)

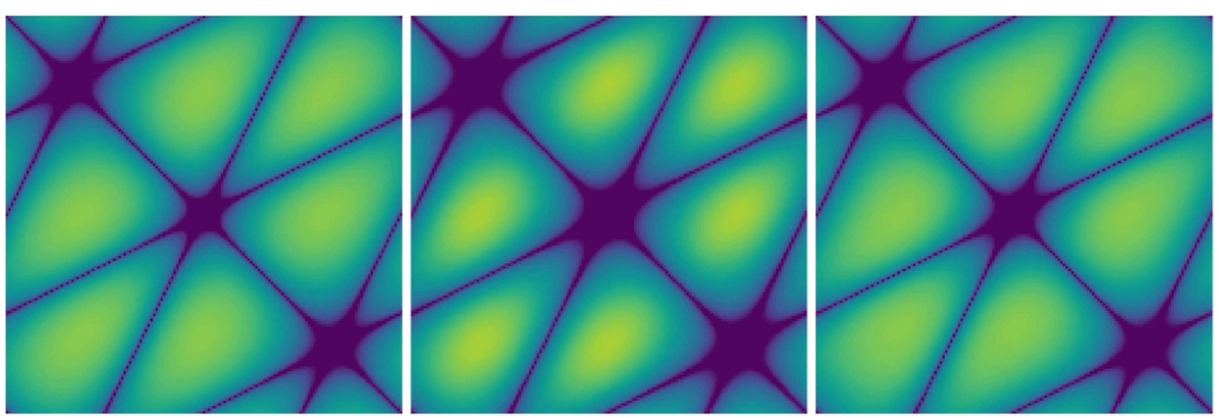
Flow on samples

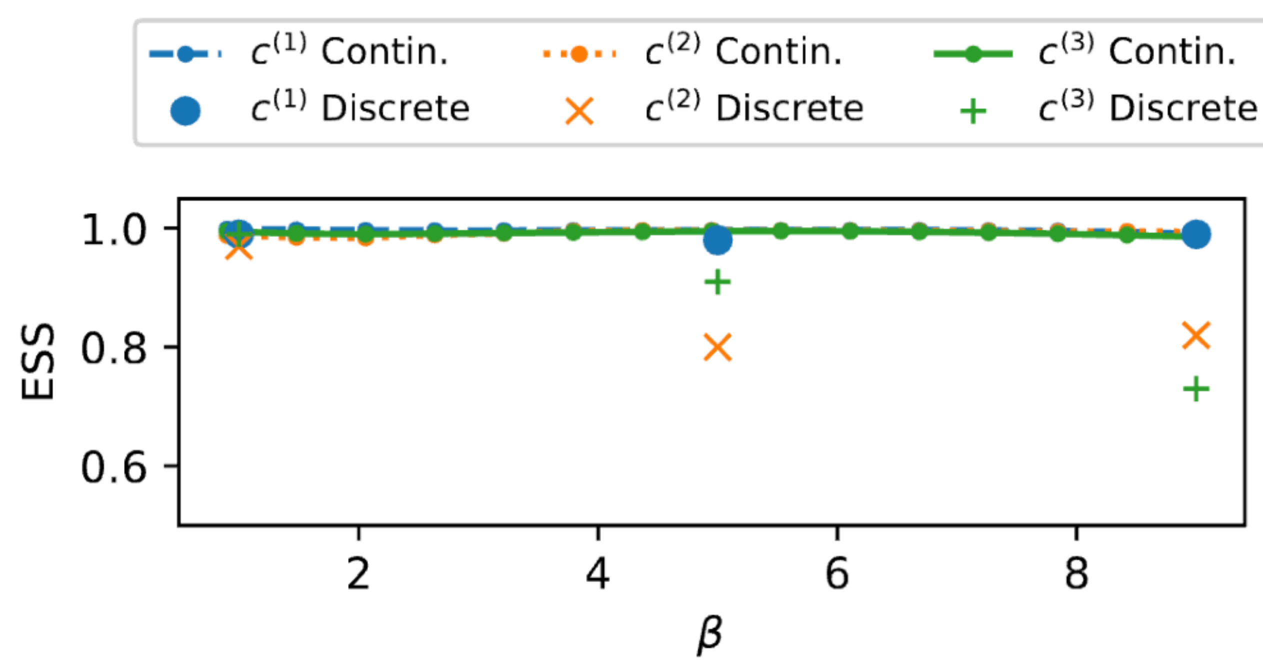
Identify β with flow time

Simplest theory-conditioned flow



$$Z^a = \partial_a \Phi_\theta(U, t)$$

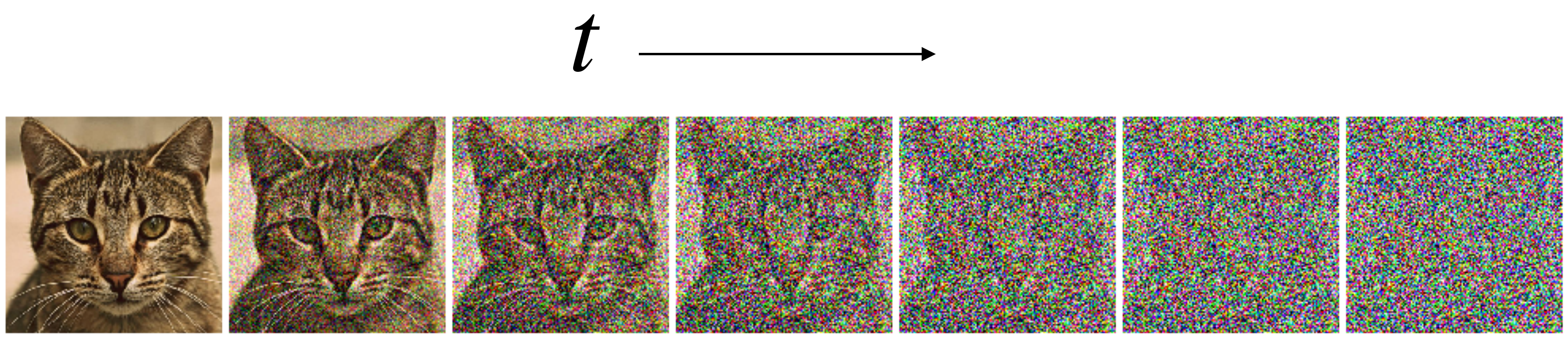




Takeaways

- Integration & adjoint sensitivity for any $\mathbb{R} \times SU(N)$.
- Tractable divergence computation.
- Experiments confirm architecture improvements.
- Straight-forward temperature-conditioning.

Diffusion Models



Brownian motion SDE from images to noise: $d\phi = -\frac{1}{2}\beta\,\phi\,dt + \beta\,dw$

Solving the SDE starting at $p_0(\phi)$
leads to a path in distributions $p_t(\phi)$.

All information about the flow is encoded in the Stein score:

Want to learn $s_\theta(\phi, t) \approx -\nabla_\phi \log p_t(\phi)$ $\longrightarrow$ Know inverse SDE!

Controlled destruction (GUD)

Forward OU:

$$\phi(t) = \alpha_t \phi(0) + \sigma_t \epsilon$$