Applications of Lattice Gauge Equivariant Neural Networks

Matteo Favoni

Institute for Theoretical Physics, TU Wien
Aug 1, 2022

QCHS 2022

Based on: M. Favoni, A. Ipp, D. I. Müller, D. Schuh, Phys.Rev.Lett. **128**, 032003

Code: gitlab.com/openpixi/lge-cnn

TECHNISCHE UNIVERSITÄT WIEN Vienna | Austria

Der Wissenschaftsfonds.

Doktoratskolleg Particles and Interactions

Outline

Introduction

- Neural networks (NNs) are a widely used tool in many scientific areas
- NNs are universal approximators of any given function
- In general, symmetries in data are learnt, therefore only approximated
- Good strategy: meet the requirements of the specific problem
- In quantum field theories, symmetries play a key role
- A desirable approach is to design NNs so that such symmetries are respected
- Example: translational symmetry is incorporated by construction in convolutional neural networks (CNNs) (under certain circumstances)
- Here: build NNs respecting gauge symmetries by construction

Lattice gauge theory

- Discretization of Yang-Mills theory
- Gauge links $\mathcal{U}$

$$U_{\mathbf{x},\mu} \simeq \exp\left(iga^{\mu} A_{\mu}(\mathbf{x}+\mathbf{a}^{\mu}/2)\right) \in \mathrm{SU}(N_c)$$

- 1×1 loops $\mathcal{W}$

$$W_{\mathbf{x},\mu\nu} = U_{\mathbf{x},\mu} U_{\mathbf{x}+\mu,\nu} U_{\mathbf{x}+\mu+\nu,-\mu} U_{\mathbf{x}+\nu,-\nu}$$

- Wilson action

$$S_W[U] = \frac{2}{g^2} \sum_{x\in\Lambda} \sum_{\mu<\nu} \mathrm{Re}\,\mathrm{tr}[\mathbf{1} - W_{x,\mu\nu}]$$

Gauge equivariance

- Lattice gauge transformations for $\mathcal{U}$ and $\mathcal{W}$

$$T_\Omega U_{\mathbf{x},\mu} = \Omega_{\mathbf{x}} U_{\mathbf{x},\mu} \Omega^\dagger_{\mathbf{x}+\mu}, \qquad \Omega_{\mathbf{x}} \in \mathrm{SU}(N_c)$$
$$T_\Omega W_{\mathbf{x},\mu\nu} = \Omega_{\mathbf{x}} W_{\mathbf{x},\mu\nu} \Omega^\dagger_{\mathbf{x}}$$

- Gauge equivariant function

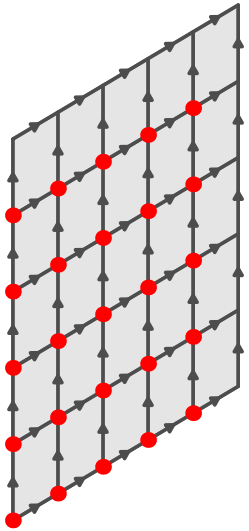
$$g(T_\Omega \mathcal{U}, T_\Omega \mathcal{W}) = T'_\Omega g(\mathcal{U}, \mathcal{W})$$

- Gauge invariant function (e.g. observables, action)

$$g(T_\Omega \mathcal{U}, T_\Omega \mathcal{W}) = g(\mathcal{U}, \mathcal{W})$$

- The individual layers of a lattice gauge equivariant convolutional neural network (L-CNN) are designed to respect gauge equivariance

Preprocessing layer



$\mathcal{U} = \{U_{\mathbf{x},\mu}\}$

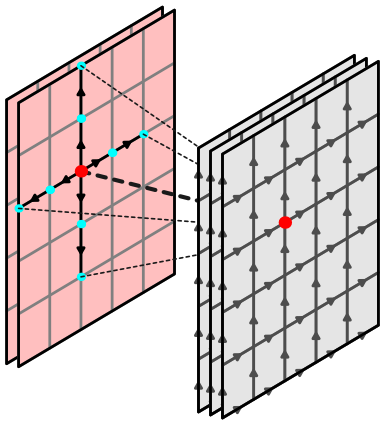
$\mathcal{W} = \{W_{\mathbf{x},\mu\nu}\}$

Preprocessing layers (**Plaq, Poly**)

- Preprocess input $\mathcal{U}$ to generate locally transforming objects $\mathcal{W}$, i.e. plaquettes and Polyakov loops $\mathcal{L}_{x,\mu}(\mathcal{U}) = \prod_k U_{x+k\mu,\mu}$

$$\textbf{Plaq, Poly} : \mathcal{U} \to (\mathcal{U}, \mathcal{W})$$

L-Conv



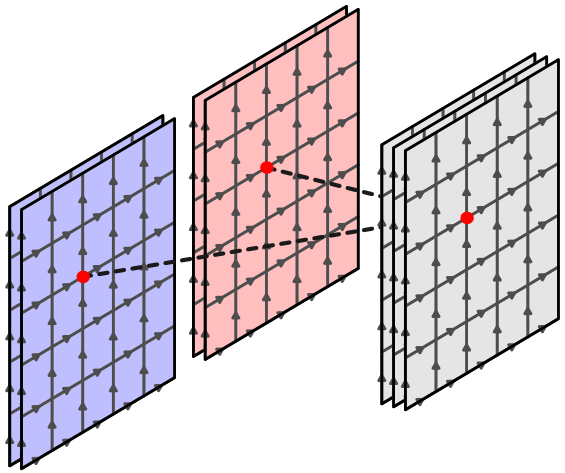
Lattice gauge equivariant convolutions (**L-Conv**)

- Properties of CNNs: compact kernels, weight sharing
- Parallel transport of data $\mathcal{W}$ to common point using $\mathcal{U}$
- Avoid path dependence by restricting the kernel along the axis

$$\textbf{L-Conv} : W'_{\mathbf{x},i} = \sum_{j,\mu,k} \omega_{i,j,\mu,k} U_{\mathbf{x},k\cdot\mu} W_{\mathbf{x}+k\cdot\mu,j} U^{\dagger}_{\mathbf{x},k\cdot\mu}$$

- Equivariant convolutions: $(\mathcal{U},\mathcal{W}) \to (\mathcal{U},\mathcal{W}')$

L-BL



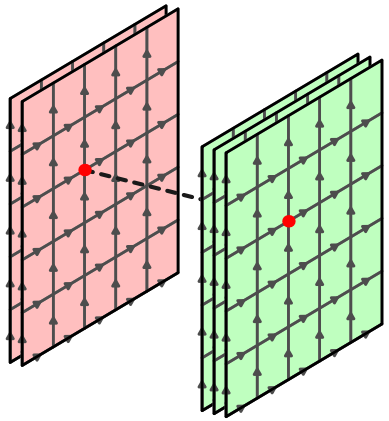
Equivariant bilinear layers (**L-BL**):

- Multiplication of $\mathcal{W}$'s at same lattice point is equivariant
- Allow the network to grow larger loops

$$\textbf{L-BL} : W''_{\mathbf{x},i} = \sum_{j,k} \alpha_{ijk} W_{\mathbf{x},j} W'_{\mathbf{x},k}$$

- Bilinear layers: $(\mathcal{U}, \mathcal{W}) \times (\mathcal{U}, \mathcal{W}') \to (\mathcal{U}, \mathcal{W}'')$

L-Act



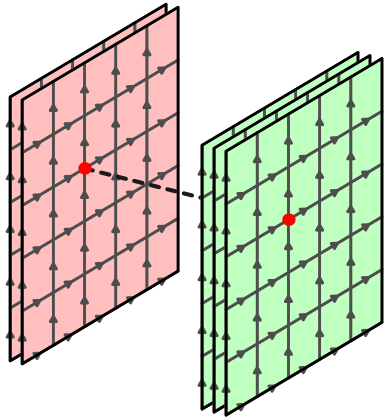
Gauge equivariant activation functions (**L-Act**):

- Multiplication of $\mathcal{W}$ with gauge invariant scalar functions a

$$\textbf{L-Act} : W'_{\mathbf{x}} = a(\mathrm{Tr}\, W_{\mathbf{x}})\, W_{\mathbf{x}}$$

- Activation functions: $(\mathcal{U}, \mathcal{W}) \to (\mathcal{U}, \mathcal{W}')$

L-Exp



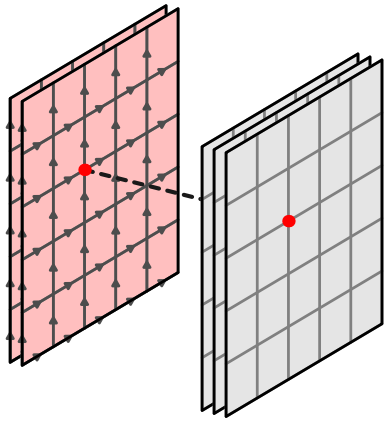
Equivariant exponential layers (**L-Exp**):

- Equivariant method to modify links $\mathcal{U} \to \mathcal{U}'$
- Multiplication of $\mathcal{U}$ with locally transforming $\mathrm{SU}(N_c)$

$$U'_{\mathbf{x},\mu} = \exp\left(i \sum_i \beta_{\mu,i} \left[W_{\mathbf{x},i}\right]_{\mathrm{ah}}\right) U_{\mathbf{x},\mu}$$

- Equivariant exponential layer: $(\mathcal{U}, \mathcal{W}) \to (\mathcal{U}', \mathcal{W})$

Trace



Generate gauge invariant output (**Trace**)

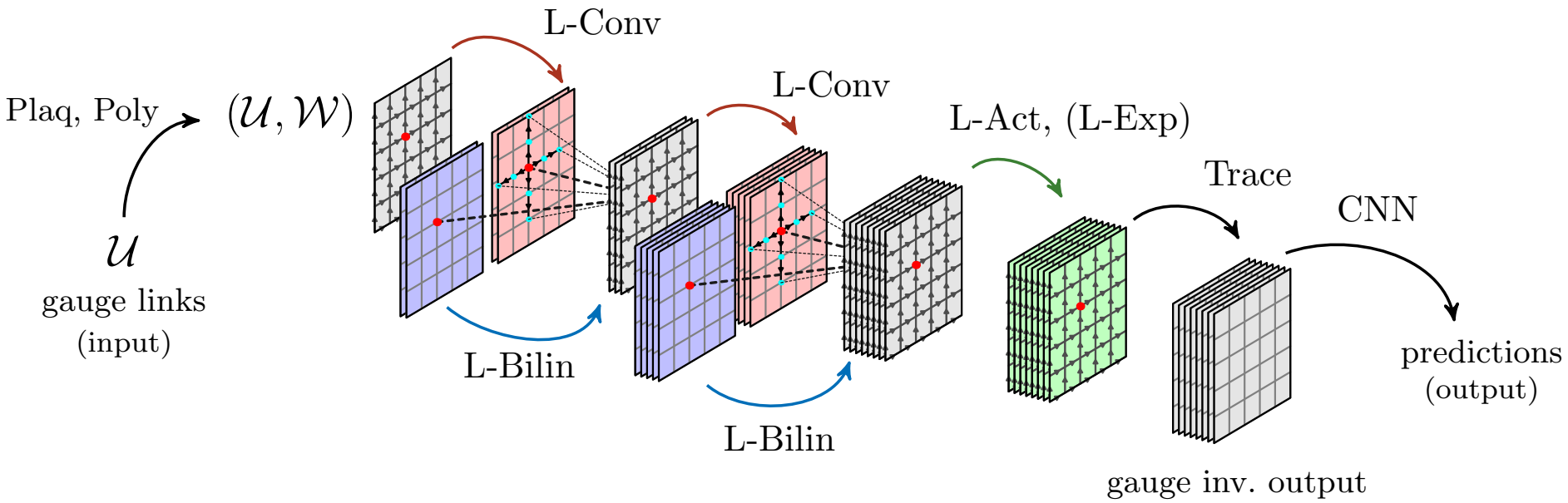
- Computes traces of $\mathcal{W}$'s: gauge invariant complex numbers

$$\textbf{Trace} : w_{\mathbf{x},i} = \mathrm{tr}\, W_{\mathbf{x},i} \in \mathbb{C}$$

- No trainable parameters ("postprocessing")
- Gauge invariant output can be passed to traditional CNN

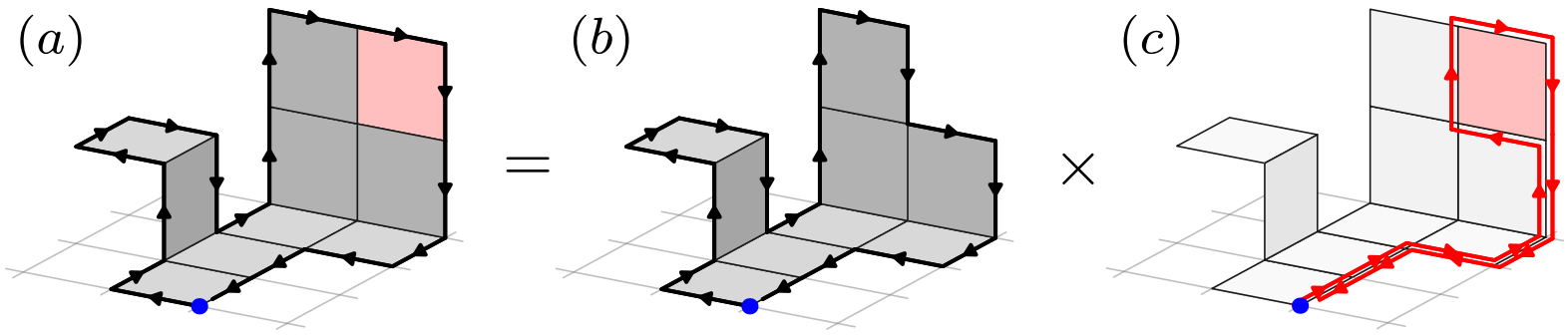
Lattice gauge equivariant convolutional neural networks

L-CNNs: A collection of gauge equivariant layers for lattice gauge configurations



Construction of arbitrary Wilson loops

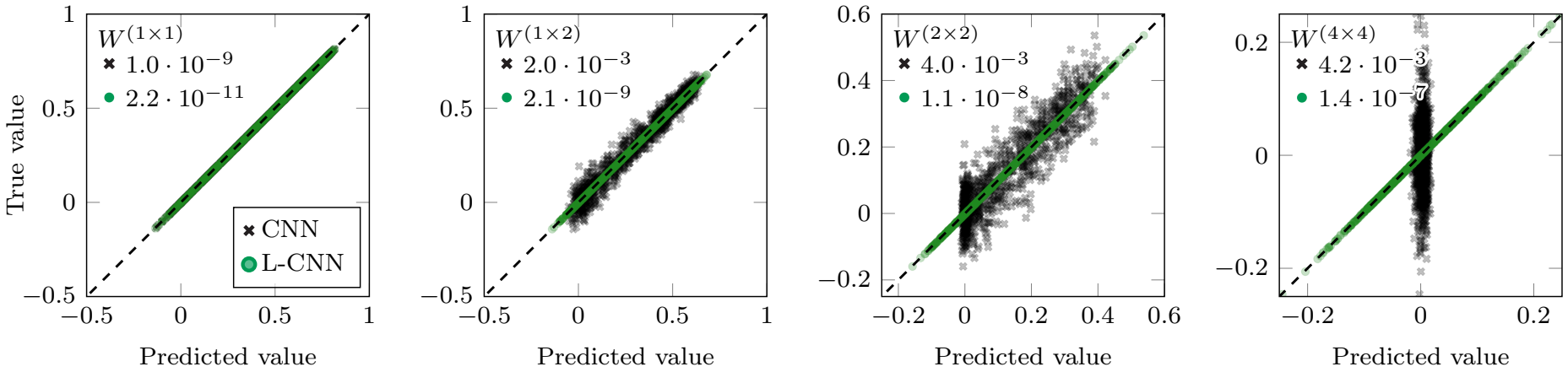
- Repeated applications of **L-Conv** and **L-Bilin** operations can be used to generate arbitrarily sized Wilson loops if input $\mathcal{W}$ consists of plaquettes (preprocessing layer **Plaq**)



- Non-contractible loops can also be generated by including Polyakov loops in the input $\mathcal{W}$ (preprocessing layer **Poly**)
- Non-linear functions of Wilson loops are possible through **L-Act**, **Trace** and passing gauge invariant output to traditional CNNs
- L-CNNs are universal approximators for gauge invariant functions on the lattice

Performance of standard CNNs vs LCNNs

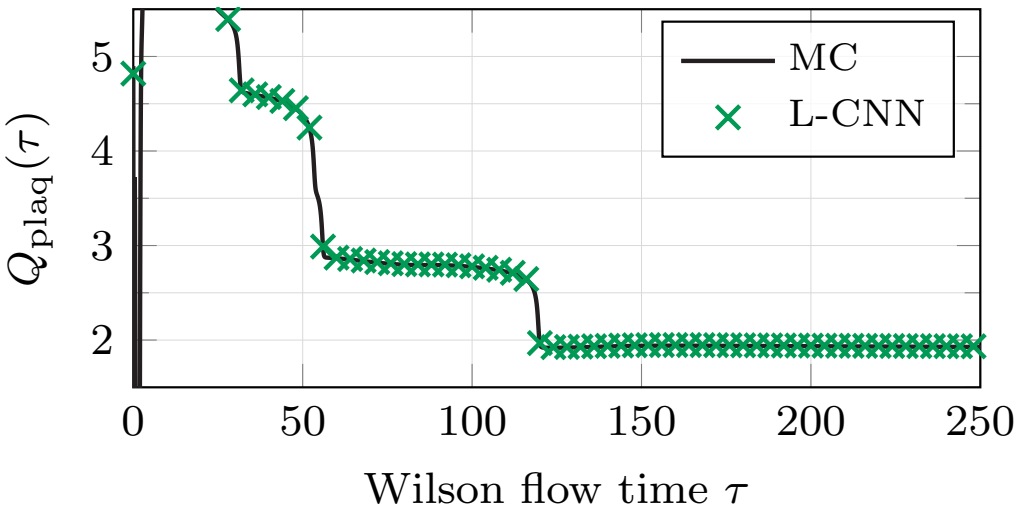
Benchmark problem: regression of Wilson loops from 1×1 to 4×4 on 2D lattice



True vs. predicted values for CNNs and L-CNNs for $n \times m$ Wilson loops (best models)

- Increasing loop size from left to right $\rightarrow$ increasingly harder problem
- Deteriorating performance of baseline CNNs with increased loop size
- Best L-CNN **always** beats best baseline CNN
- Consistent performance of L-CNNs across all loop and lattice sizes

Regression on topological charge



- L-CNN predictions vs. true values (MC) for $Q_{\mathrm{plaq}} = \sum_{x\in\Lambda} q_x^{\mathrm{plaq}} = \sum_{x\in\Lambda} \frac{\epsilon_{\mu\nu\rho\sigma}}{32\pi^2} \mathrm{tr}\left[\frac{U_{x,\mu\nu} - U_{x,\mu\nu}^\dagger}{2i} \frac{U_{x,\rho\sigma} - U_{x,\rho\sigma}^\dagger}{2i}\right]$ on an 8×24^3 configuration
- Training on 4×8^3 lattices and testing performed also on larger lattices with cooled configurations

Neural ODEs

Neural ordinary differential equations (NODEs) are ODEs parametrized by NNs (see e.g. arXiv:1806.07366)

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \mathbf{f}(\mathbf{z}(t), \theta, t)$$

- $\mathbf{z}(\mathbf{t})$: time-dependent D-dimensional vector
- $\mathbf{f}(\mathbf{z}(t), \theta, t)$: NN parametrized by the weights θ with a D-dimensional output
- Input: boundary value $\mathbf{z}_0 = \mathbf{z}(t_0)$
- Label: desired output vector $\mathbf{z}_1^*$
- Prediction: final state vector $\mathbf{z}(t_1) = \mathbf{z}_0 + \int_{t_0}^{t_1} \mathrm{d}t' \, \mathbf{f}(\mathbf{z}(t'), \theta, t')$
  An ODE integrator (e.g. Euler, Runge-Kutta) is used for the state evolution
- Training: minimization of the loss function $\mathcal{L}(\theta) = (\mathbf{z}_1^* - \mathbf{z}(t_1))^2$

Adaptation to Wilson flow

We study the equivariant flow equation

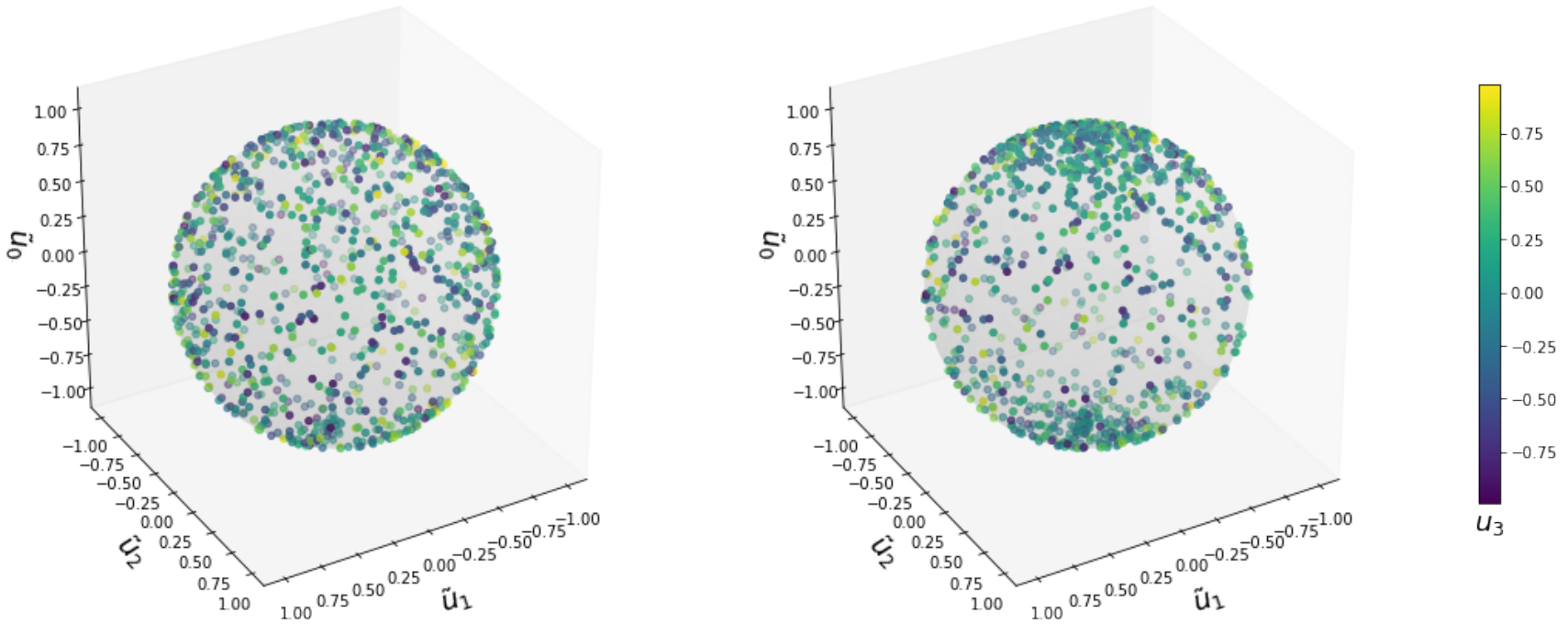
$$\frac{\mathrm{d}U_{x,\mu}(t)}{\mathrm{d}t} = iH_\mu[U(t),\theta,t]\,U_{x,\mu}(t)$$

- $U(t)$: gauge link configuration
- $H_\mu[U(t),\theta,t]$: NN parametrized by the weights θ with a traceless and hermitian output
- Input: boundary value $U_0 = U(t_0)$
- Label: desired output U_1^*
- Prediction: final configuration $U(t_1)$ found by the iterative application of the exponential map $U_{x,\mu}^{(i+1)} = \exp\left(iH_\mu[U^{(i)}]\Delta t\right)U_{x,\mu}^{(i)}$
- Training: minimization of the loss function $\mathcal{L}(\theta) = \|U_1^* - U(t_1)\|^2$ (e.g. Frobenius norm)

SU(2) toy model

We solve the Wilson flow equation on the previous slide for a single-link configuration in SU(2) with the action $S[U] = \mathrm{Re}\,\mathrm{tr}(U^2)$. This action has two minima: $\pm\mathbf{1}$. If $\mathrm{tr}U > 0$ the link flows to the north pole ($+\mathbf{1}$), otherwise it flows to the south pole ($-\mathbf{1}$).

$$u_0 = \frac{1}{2}\mathrm{tr}(U), \qquad u_i = \frac{1}{2}\mathrm{tr}(U\sigma_i), \qquad \tilde{u}_j = u_j/(u_0^2 + u_1^2 + u_2^2)^{1/2}$$



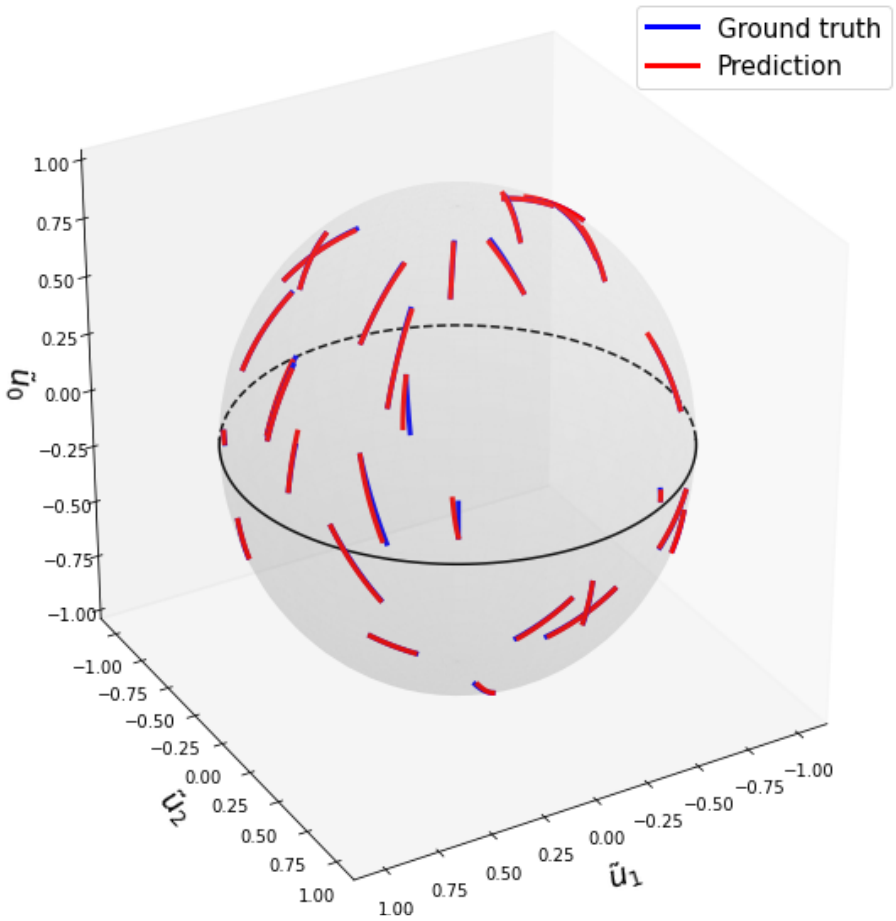
SU(2) toy model

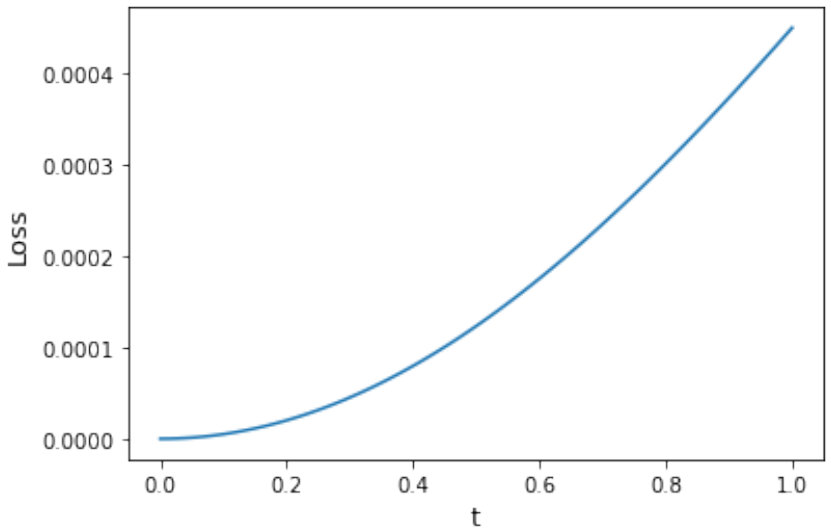
The traceless and hermitian matrix $H[U(t), \theta, t]$ is constructed with the following steps

- The complex entries of U are split into real and imaginary part
- They are fed into a multi-layer perceptron with real weights
- The output is recombined into a complex matrix
- Taking the anti-hermitian, $[C]_{\mathrm{ah}} = \frac{1}{2i}\left(C - C^\dagger\right) - \frac{1}{2iN_c}\mathbf{1}\,\mathrm{Tr}\left(C - C^\dagger\right)$, projects the output onto $\mathfrak{su}(2)$
- The application of the exponential map yields a matrix in SU(2)

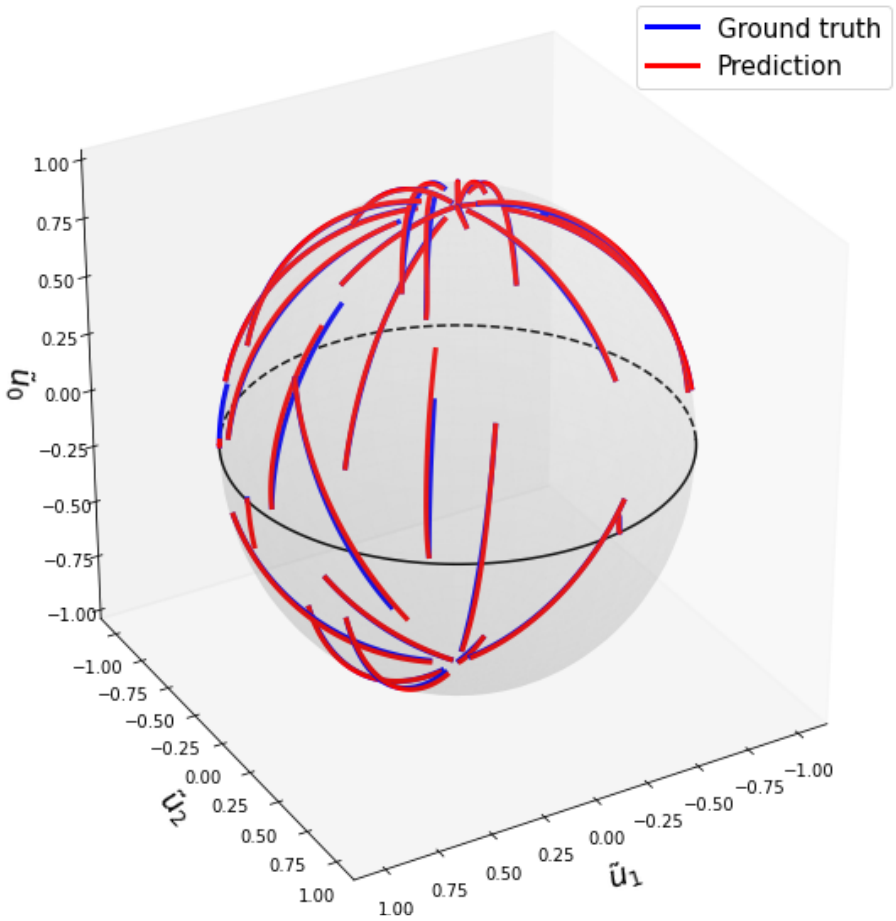
Training is performed by choosing the Frobenius norm as loss function

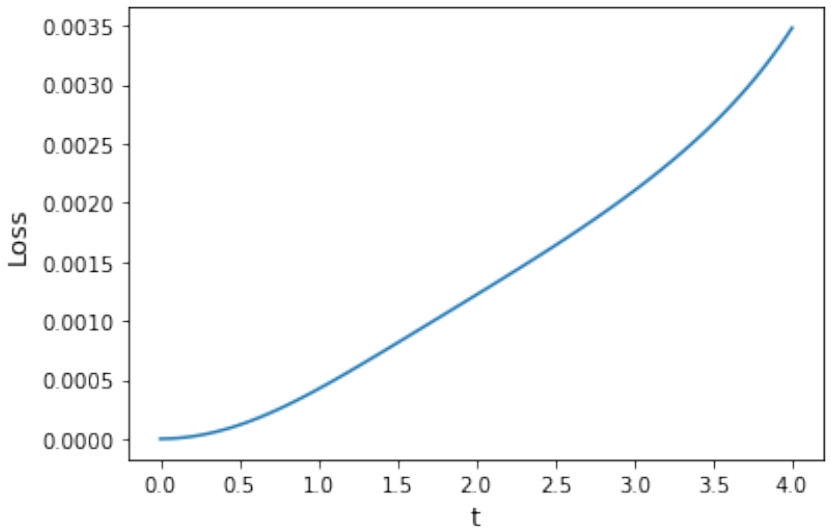
Testing





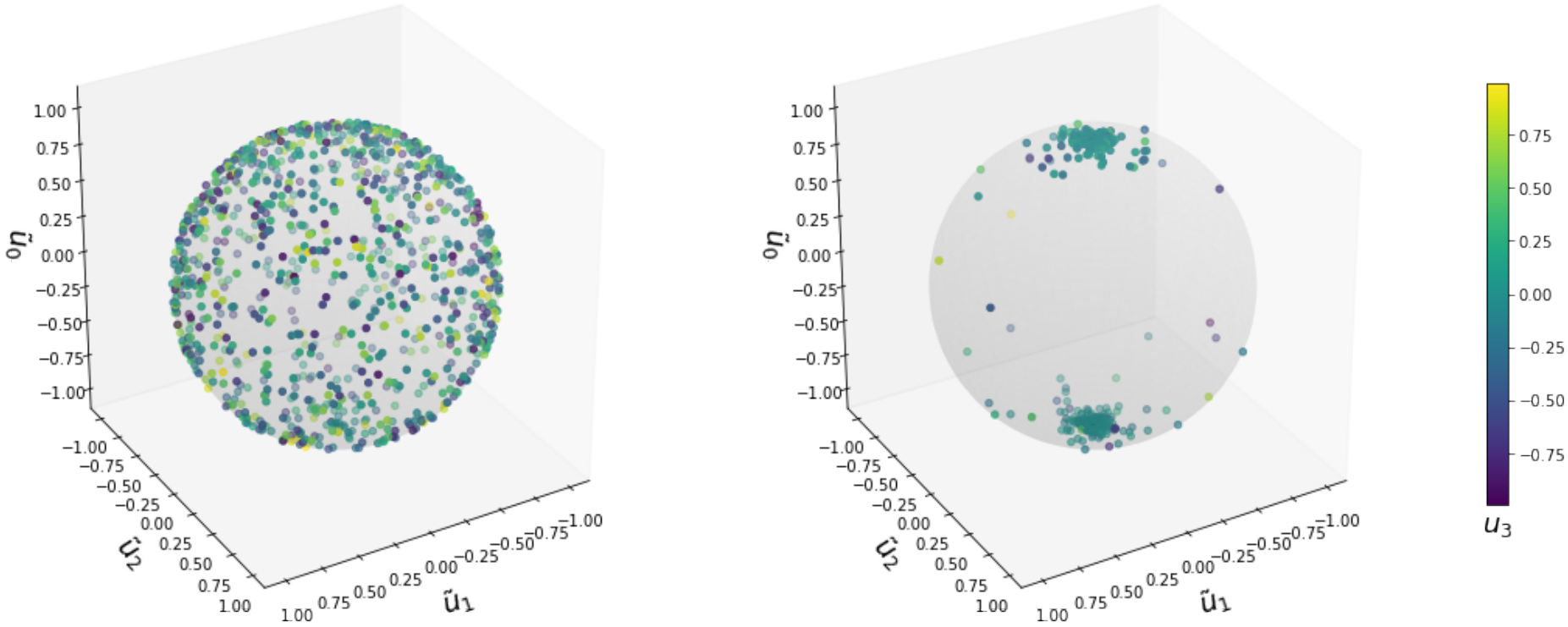
Extrapolation to larger times



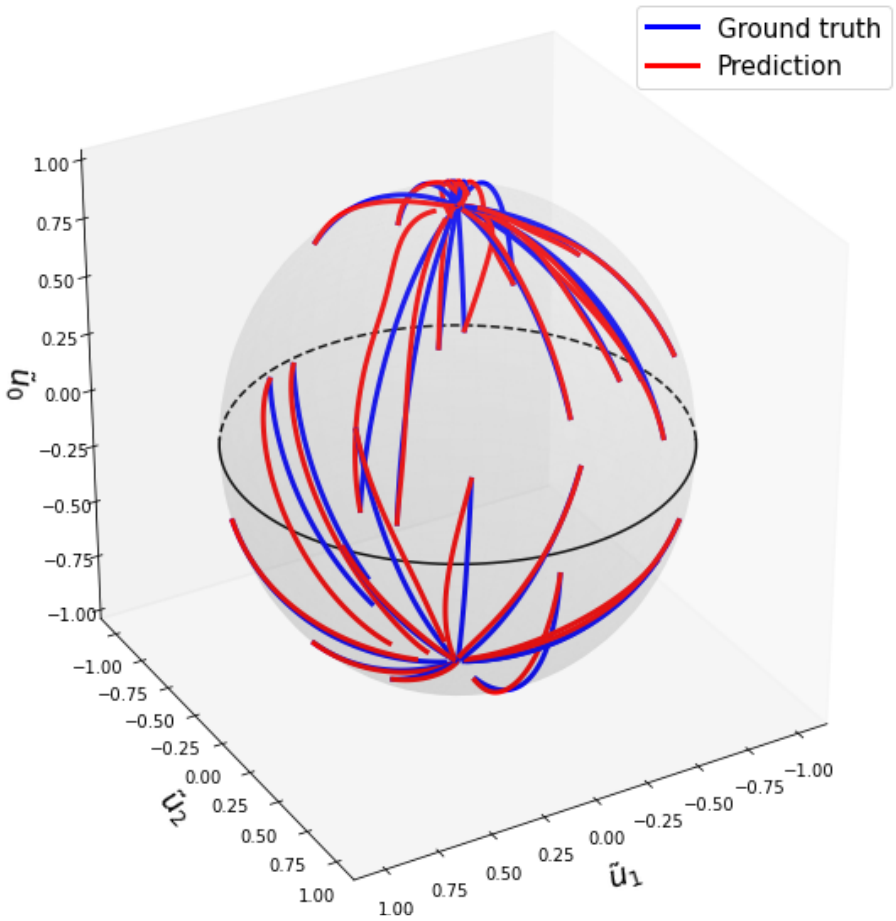


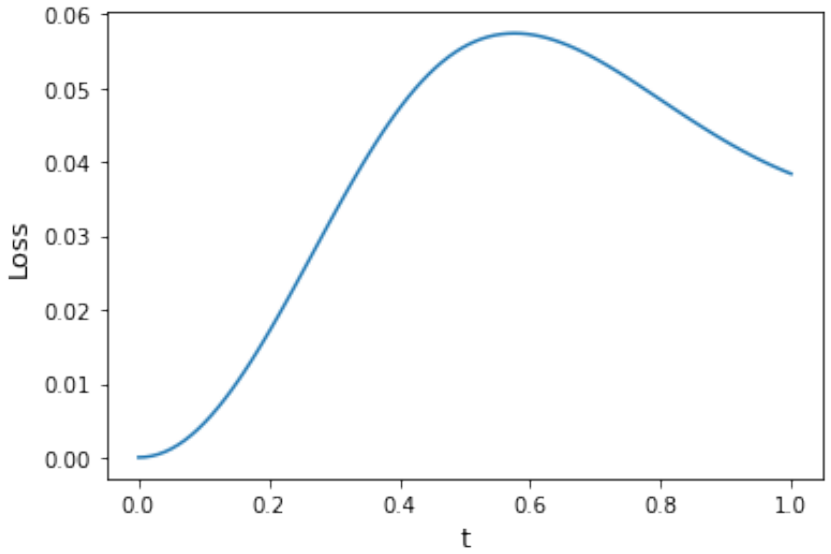
Dataset

If the time steps are too large most samples flow too close to the poles and learning becomes harder



Testing





Conclusions and outlook

Conclusions

- L-CNNs are gauge equivariant NNs that can be applied to lattice problems
- Better performance in regression tasks compared to traditional NNs
- They can be applied to the modification of gauge link configurations
- A minimal example of a Wilson flow with a single link has been successfully solved

Next steps

- Implement the adjoint sensivity method to save memory (done for U(1))
- Extend the toy model to proper lattices
- Apply the L-CNNs to Wilson flows