Applications of Lattice Gauge Equivariant Neural Networks

Matteo Favoni[1,2,*], *Andreas* Ipp[1,**], and *David I.* Müller[1,***]

[1]Institute for Theoretical Physics, TU Wien,
 Wiedner Hauptstr. 8-10, 1040 Vienna, Austria
[2]Speaker and corresponding author

Abstract. The introduction of relevant physical information into neural network architectures has become a widely used and successful strategy for improving their performance. In lattice gauge theories, such information can be identified with gauge symmetries, which are incorporated into the network layers of our recently proposed Lattice Gauge Equivariant Convolutional Neural Networks (L-CNNs). L-CNNs can generalize better to differently sized lattices than traditional neural networks and are by construction equivariant under lattice gauge transformations. In these proceedings, we present our progress on possible applications of L-CNNs to Wilson flow or continuous normalizing flow. Our methods are based on neural ordinary differential equations which allow us to modify link configurations in a gauge equivariant manner. For simplicity, we focus on simple toy models to test these ideas in practice.

1 Introduction

In the past decade, neural networks (NNs) have been established as an essential tool with numerous applications in e.g. computer science and the natural sciences. The crucial role played by symmetries in a large amount of scientific problems has attracted the idea that the inclusion of such symmetries in the NN architecture could be beneficial in enhancing their performance. For example, convolutional neural networks (CNNs) are based on the inclusion of translational symmetry as an inherent property of their architecture. In computer vision problems such as image classification, this proved to be a powerful idea, due to the fact that the position of a particular feature that has to be detected is irrelevant and it is just the presence of the feature that matters. This approach has been generalized to include other symmetries in group equivariant convolutional neural networks (G-CNNs) [1], which take into account not just translational, but e.g. also rotational and reflection symmetry. Recently, this idea has been further extended to local symmetries [2]. The more general framework dealing with symmetries in neural networks is called geometric deep learning [3].

In theoretical physics, and more specifically in lattice field theories with global symmetries, CNNs or G-CNNs have been successfully applied to solving regression problems and detecting phase transitions [4–10]. In the context of Abelian and non-Abelian gauge theories, which exhibit local symmetries, there has been progress in the direction of incorporating

---

[*]e-mail: favoni@hep.itp.tuwien.ac.at
[**]e-mail: ipp@hep.itp.tuwien.ac.at
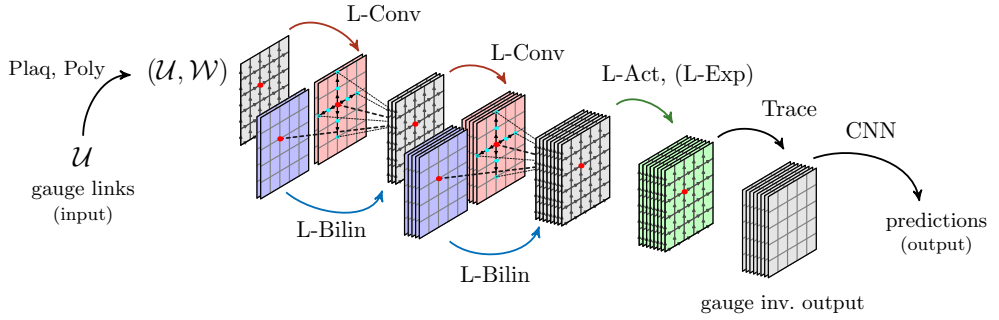[***]e-mail: dmueller@hep.itp.tuwien.ac.at

Figure 1: A possible L-CNN architecture. Figure from [17]

gauge symmetry in the network architecture [11–15]. For example, gauge equivariant normalizing flows [11, 12, 15, 16] can be used in place of Monte Carlo simulations to sample uncorrelated gauge configurations while retaining gauge symmetry. Similarly, a lattice gauge equivariant convolutional neural network (L-CNN) was proposed in our paper [17], in which the elementary layers of the architecture individually preserve gauge symmetry. L-CNNs have been used successfully for regression tasks and in principle can also be employed for the generation of gauge configurations. The continuous flow approach proposed in [18, 19] provides a continuous generalization of normalizing flows applied to lattice field theory. In contrast to normalizing flows, this continuous formulation allows for a straightforward inclusion of exact global symmetries. At its core, continuous flows are an application of neural ordinary differential equations (NODEs) [20], which are ordinary differential equations (ODEs) parametrized by NNs.

In these proceedings, we first review the basics of lattice gauge theory and the L-CNN architecture, and we show how NODEs can be modified to study Wilson flow [21] and exemplify it with an SU(2) toy model.

2 Lattice gauge equivariant neural networks

Lattice gauge theory is a discretized version of SU(N_c) Yang-Mills theory, in which spacetime is approximated by a periodic hypercubic lattice in $D + 1$ dimensions with imaginary time and with lattice spacing a. In the lattice discretizations, the continuous gauge fields A_μ are replaced by the link variables $U_{x,\mu}$ via the following definition:

$$U_{x,\mu} = \mathcal{P} \exp\left(ig \int_x^{x+\hat{\mu}} \mathrm{d}x'^\nu A_\nu(x')\right),$$

(1)

where $\mathcal{P}$ denotes path ordering, g is the coupling constant and the integral is performed over the straight line connecting the site x to the site $x + \hat{\mu}$. The gauge fields are elements of the $\mathfrak{su}(N_c)$ algebra, while the links can be interpreted as the parallel transporters along the lattice edges and live in the SU(N_c) group. In practice, we employ the fundamental representation of $U_{x,\mu}$, where links are represented as complex $N_c \times N_c$ matrices.

It is possible to multiply adjacent links, the repetition of which leads to arbitrary Wilson lines. If the start and end point of a Wilson line coincide, closed loops are formed and are called Wilson loops. The simplest Wilson loop on a hypercubic lattice is the plaquette given by

$$U_{x,\mu\nu} = U_{x,\mu} U_{x+\hat{\mu},\nu} U_{x+\hat{\mu}+\hat{\nu},\mu}^\dagger U_{x+\hat{\nu},\mu}^\dagger.$$

(2)

The Wilson action [22], formulated in terms of plaquettes,

$$S_W[U] = \frac{2}{g^2} \sum_{x \in \Lambda} \sum_{\mu < \nu} \mathrm{Re}\,\mathrm{Tr}\left[\mathbf{1} - U_{x,\mu\nu}\right],\tag{3}$$

is equivalent to the Yang-Mills action in the continuum limit $a \to 0$. A general lattice gauge transformation applied to links

$$U_{x,\mu} \to \Omega_x U_{x,\mu} \Omega_{x+\hat{\mu}}^{\dagger}\tag{4}$$

induces a local transformation of the plaquettes

$$U_{x,\mu\nu} \to \Omega_x U_{x,\mu\nu} \Omega_x^{\dagger}.\tag{5}$$

These transformations leave the Wilson action unchanged, meaning the theory is invariant under SU(N_c) lattice gauge transformations.

In order to build up an L-CNN [17], we can consider its individual layers, each of which is designed to respect gauge equivariance. First, the input consists of the set of gauge links $\mathcal{U}$ of a particular lattice configuration and locally transforming objects $\mathcal{W}$, which in practice we choose to be the plaquettes, but can also be the Polyakov loops (closed Wilson lines wrapping around the periodic boundary of the lattice). The L-Conv layer is a gauge equivariant convolution, which acts as a parallel transporter of locally transforming objects $\mathcal{W}$, while L-Bilin performs a multiplication of such objects (more specifically, it is a bilinear layer). We proved that the repeated application of these two operations can grow arbitrarily sized Wilson (or Polyakov) loops. Moreover, it is possible to introduce non-linearity via L-Act layers, which behave like activation functions in traditional CNNs. The Trace layer yields a gauge invariant output that can be passed to a traditional CNN. A possible realization of such a network is depicted in Fig. 1. By virtue of the ability of generating any loop and the non-linearity, L-CNNs can be seen as universal approximators of gauge-equivariant functions on the lattice.

Among the relevant results found in [17], it is worth mentioning that the L-CNNs performed very well on the regression of Wilson loops up to a size of 4×4 and simple observables such as the topological charge density, beating traditional CNNs on the same task.

3 Adaptation of NODEs to lattice gauge theory

NODEs are ODEs parametrized by neural networks [20]. As in the original paper, we will focus on first order ODEs:

$$\frac{d\mathbf{z}}{dt} = \mathbf{f}(\mathbf{z}(t), \theta, t).\tag{6}$$

The unknown function $\mathbf{z}(t)$ is a time-dependent D-dimensional vector and $\mathbf{f}(\mathbf{z}(t), \theta, t)$ is a D-dimensional function parametrized by a priori unknown weights θ. In particular one can choose $\mathbf{f}(\mathbf{z}(t), \theta, t)$ to be represented by a NN. NODEs can be understood as generalizations of residual networks [23] with continuous depth, where the time coordinate t is used in place of the discrete depth of the network. Starting with an input state $\mathbf{z}_0 = \mathbf{z}(t_0)$ at $t = t_0$, the NODE can be formally solved by

$$\mathbf{z}(t_1) = \mathbf{z}_0 + \int_{t_0}^{t_1} dt'\,\mathbf{f}(\mathbf{z}(t'), \theta, t'),\tag{7}$$

which provides predicted states $\mathbf{z}(t_1)$ at some final time $t = t_1$. In this manner, NODEs map arbitrary input states to output states similar to generic NNs. The mapping depends on the NN architecture and the weights θ. NODEs can thus be used to solve regression problems:

given a dataset characterized by the initial conditions $\mathbf{z}_0^i = \mathbf{z}^i(t_0)$ (where $i \in \{1, \ldots, N_{\text{samples}}\}$), which are used as input, and the desired output vectors $\tilde{\mathbf{z}}_1^i$, which represent the labels, the weights θ can be optimized such that the final states approximate the labels as accurately as possible. In practice, this is done with the aid of an ODE integrator, such as Euler or Runge-Kutta. We can require that the discrepancy between the labels and the predicted final states is minimized by introducing a loss function such as the mean squared error (MSE), $\mathcal{L}(\theta) = \sum_i (\tilde{\mathbf{z}}_1^i - \mathbf{z}^i(t_1))^2 / N_{\text{samples}}$, and run the training procedure in order to optimize the weights θ.

While the approach described above only uses the final state labels in the optimization problem, it is also possible to include the discrepancies of the whole state evolution (i.e. more points t_j along the trajectory $\mathbf{z}(t_j)$) in the loss function for successful training. If we only use the final states at t_1, then it is crucial that the dataset provides sufficient information to reconstruct the underlying dynamics.

We can adapt the previous scheme to study continuous flow applied to lattice gauge configurations

$$\frac{\mathrm{d}U_{x,\mu}(\tau)}{\mathrm{d}\tau} = iH_\mu[U_{x,\mu}(\tau), \theta, \tau]\, U_{x,\mu}(\tau), \tag{8}$$

where $U_{x,\mu} \in \mathrm{SU}(N_c)$ are gauge links, τ is flow time and $H_\mu[U(\tau), \theta, \tau]$ is a NN parametrized by the weights θ with a traceless and Hermitian output. This last requirement guarantees that the gauge links do not leave the group during the evolution. In order to retain gauge equivariance, H_μ can be modeled with an L-CNN.

Our dataset consists of the initial conditions $U_{x,\mu,0}^i = U_{x,\mu}^i(\tau_0)$ and the desired output configurations $\tilde{U}_{x,\mu,1}^i$, which define input and labels respectively. A standard ODE integrator would in general break the group structure, so we make use of the iterative application of the exponential map

$$U_{x,\mu}^i(\tau_{j+1}) = \exp\left(iH_\mu[U^i(\tau_j), \theta, \tau_j]\Delta\tau\right) U_{x,\mu}^i(\tau_j) \tag{9}$$

for time evolution. Since H_μ is traceless and Hermitian, i.e. it can be understood as a $\mathfrak{su}(N_c)$ algebra element, the links remain in the group manifold. The final configuration is then used in a loss function, such as

$$\mathcal{L}(\theta) = \frac{1}{N_{\text{samples}}} \sum_{x,\mu} \sum_i \|\tilde{U}_{x,\mu,1} - U_{x,\mu}^i(\tau_1)\|^2, \tag{10}$$

where $\|\ldots\|$ denotes the Frobenius norm.

4 SU(2) Wilson flow toy model

We test the adaptation of the NODE approach to Wilson flow [21] using a toy model consisting of a single SU(2) link characterized by the action $S[U] = \operatorname{Re} \operatorname{Tr}(U^2)$. Starting with randomly distributed initial matrices $U_0^i \in \mathrm{SU}(2)$, we generate a dataset of flowed matrices $\tilde{U}_1^i$ by applying gradient descent on $S[U]$ using group derivatives akin to Wilson flow in lattice gauge theory. The action exhibits two minima, $\pm\mathbf{1}$, toward which Wilson flow lets the links evolve depending on the initial conditions. If $\operatorname{Tr} U > 0$, the link is flowed toward the north pole $(+\mathbf{1})$, otherwise the dynamics is directed toward the south pole $(-\mathbf{1})$. For links with $\operatorname{Tr} U = 0$, the dynamics are stuck. Our goal is to use NODEs to reconstruct these dynamics via the flow Eq. (8).

In order to visualize the dataset, we use the following parametrization of SU(2)

$$U = u_0\mathbf{1} + i\sigma_a u_a, \qquad u_0 = \frac{1}{2}\operatorname{Tr}(U), \qquad u_i = \frac{1}{2i}\operatorname{Tr}(U\sigma_i), \tag{11}$$
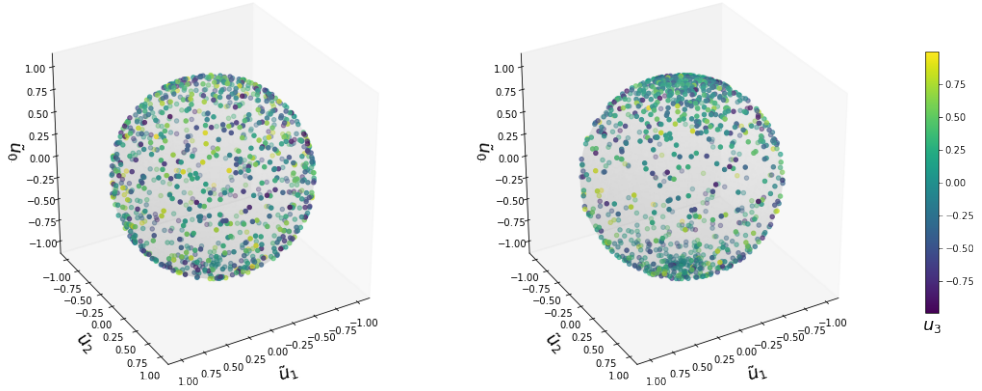
Figure 2: Visualization of 1000 samples from the dataset. Left: distribution of the initial conditions U_0^i. Right: distribution of the labels $\tilde{U}_1^i$, found by evolving the initial conditions according to the action $S[U] = \mathrm{Re}\,\mathrm{Tr}\,(U^2)$ up to $\tau_1 = 1$.

where σ_i are the Pauli matrices. We then normalize u_0, u_1 and u_2 by introducing $\tilde{u}_j = u_j / \sqrt{u_0^2 + u_1^2 + u_2^2}$ for $j \in \{0, 1, 2\}$, so that each $\tilde{u}_j$ lies on a three-dimensional sphere, while the remaining parameter u_3 determines the color of a point, as shown in Fig. 2. As anticipated, the link variables, which are initially homogeneously distributed on the sphere, flow toward one of the two minima, which in Fig. 2 correspond to the north and south pole of the sphere.

Training NODEs in principle needs backpropagation through the ODE solver. The problem with standard backpropagation is that it requires to keep the whole evolution of the system, which leads to increased memory consumption. For more complicated systems, this can easily saturate memory. A solution to this problem lies in the adjoint sensitivity method [20], which avoids having to store the entire history of the Wilson flow by solving the evolution backwards. Our implementation of this method is still a work in progress, so for this simple system we rely on standard backpropagation.

For our model, the matrix H in Eq. (8) is constructed with the following steps: the complex entries of U are split into real and imaginary parts. They are then fed into a multi-layer perceptron with real weights. The output is recombined into a complex matrix, which is generally neither Hermitian nor traceless. Therefore, we take its anti-hermitian traceless part, $[C]_{\mathrm{ah}} = \left(C - C^\dagger\right)/(2i) - \mathbf{1}\,\mathrm{Tr}\left(C - C^\dagger\right)/(2iN_c)$, which projects the output onto the $\mathfrak{su}(2)$ algebra. The application of the exponential map yields a matrix in SU(2). This guarantees that the evolution of U takes place without leaving the group.

We choose the Frobenius norm averaged over the lattice as our loss function and train on a dataset of 50000 samples using a batch size of 100 and a learning rate of 10^{-3} for 100 epochs. The multi-layer perceptron we employed has four hidden layers with 16, 64, 32 and 16 nodes respectively. We use $\tanh(x)$ as an activation function after every layer except the last.

After training, we test on 4000 samples with the same final Wilson flow time $\tau = 1$. The results are shown in Fig. 3. The left panel shows the trajectories of the ground truth (blue) and the predicted trajectories (red) during the NODE flow. The right panel shows the MSE as a function of flow time. Since the loss of $5 \cdot 10^{-6}$ is very small, the two evolutions are visually indistinguishable.

(a) Ground truth and predicted trajectories

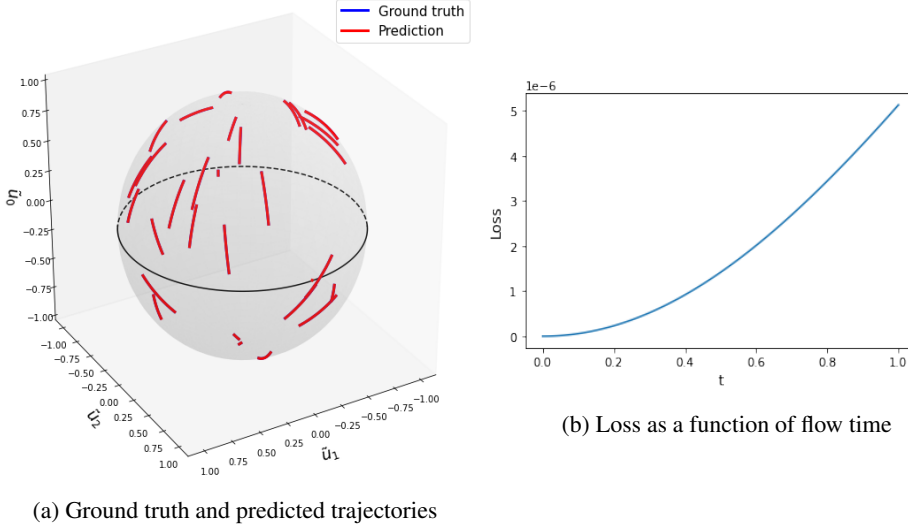(b) Loss as a function of flow time

Figure 3: Test results. (a) Evolution of 30 samples projected on the three-dimensional sphere. The ground truth and the prediction lie on top of each other. (b) The corresponding Frobenius loss as a function of flow time.

Since the loss function in Fig. 3 (b) seems to increase quadratically as a function of flow time, we investigate the loss at larger times $\tau > 1$ outside the training interval. In Fig. 4 we test 4000 samples and extrapolate to flow times up to $\tau = 10$. The deterioration of the performance is clear, since the loss jumps up to values that are three orders of magnitude larger compared to the highest loss found during testing in the interval $\tau \in [0, 1]$. Investigating our data more closely, we found two types of mispredictions which can contribute to a large loss. For one specific sample in our test dataset, the predicted trajectory moves in the opposite direction of the actual one. There are also some trajectories at large times τ that tend to overshoot the ground truth values. In both cases, we found that these trajectories originate from points that lie within a thin neighborhood of the equator ($\mathrm{Tr}U \approx 0$) and are in general very difficult for the network to evolve correctly. Despite these flaws, the results are encouraging, considering that we employed a simple multi-layer perceptron which is not adapted to the symmetries of the problem. Therefore, a network structure incorporating additional symmetries ($U \rightarrow \Omega U \Omega^{\dagger}$ with $\Omega \in \mathrm{SU}(2)$), could further improve the performance in this toy model.

5 Conclusions and outlook

In these proceedings, we reviewed the structure of L-CNNs and their successful application in regression tasks. These architectures are very flexible and their layers can be composed to modify gauge link configurations. We discussed how this can be achieved in the context of NODEs and tested it on an SU(2) Wilson flow toy model with a single link. Based on these experiments, we intend extend the toy model to actual lattice configurations and apply L-CNNs to Wilson flow.

(a) Ground truth and predicted trajectories
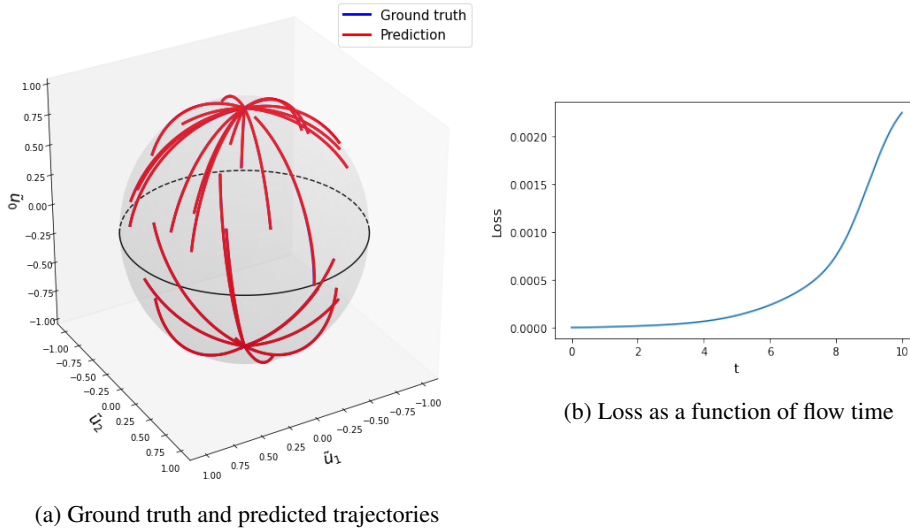

(b) Loss as a function of flow time

Figure 4: Results of our extrapolation up to $\tau = 10$ based on training up to $\tau = 1$. (a) Extrapolated evolution of 30 samples. The ground truth and the prediction are very close. (b) Loss function as a function of flow time. At large times the loss increases by up to three orders of magnitude compared to the original interval $\tau \in [0, 1]$.

References

[1] T.S. Cohen, M. Welling, *Group Equivariant Convolutional Networks*, in *Proceedings of The 33rd International Conference on Machine Learning* (JMLR, 2016), Vol. 48, pp. 2990–2999, `1602.07576`

[2] T.S. Cohen, M. Weiler, B. Kicanaoglu, M. Welling, *Gauge Equivariant Convolutional Networks and the Icosahedral CNN*, in *Proceedings of the 36th International Conference on Machine Learning* (JMLR, 2019), Vol. 97, pp. 1321–1330, `1902.04615`

[3] J.E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Petersson, D. Persson (2021), `2105.13926`

[4] K. Zhou, G. Endrődi, L.G. Pang, H. Stöcker, Phys. Rev. D **100**, 011501 (2019), `1810.12879`

[5] D.L. Boyda, M.N. Chernodub, N.V. Gerasimeniuk, V.A. Goy, S.D. Liubimov, A.V. Molochkov, Phys. Rev. D **103**, 014509 (2021), `2009.10971`

[6] S. Blücher, L. Kades, J.M. Pawlowski, N. Strodthoff, J.M. Urban, Phys. Rev. D **101**, 094507 (2020), `2003.01504`

[7] D. Bachtis, G. Aarts, B. Lucini, Phys. Rev. E **102**, 053306 (2020), `2007.00355`

[8] S. Bulusu, M. Favoni, A. Ipp, D.I. Müller, D. Schuh, Phys. Rev. D **104**, 074504 (2021), `2103.14686`

[9] D. Bachtis, G. Aarts, B. Lucini, Phys. Rev. D **103**, 074510 (2021), `2102.09449`

[10] D. Bachtis, G. Aarts, F. Di Renzo, B. Lucini, Phys. Rev. Lett. **128**, 081603 (2022), `2107.00466`

[11] G. Kanwar, M.S. Albergo, D. Boyda, K. Cranmer, D.C. Hackett, S. Racanière, D.J. Rezende, P.E. Shanahan, Phys. Rev. Lett. **125**, 121601 (2020), `2003.06413`

[12] D. Boyda, G. Kanwar, S. Racanière, D.J. Rezende, M.S. Albergo, K. Cranmer, D.C. Hackett, P.E. Shanahan, Phys. Rev. D **103**, 074504 (2021), `2008.05456`

[13] A. Tomiya, Y. Nagai (2021), `2103.11965`

[14] D. Luo, G. Carleo, B.K. Clark, J. Stokes, Phys. Rev. Lett. **127**, 276402 (2021), `2012.05232`

[15] R. Abbott et al., Phys. Rev. D **106**, 074506 (2022), `2207.08945`

[16] M.S. Albergo, D. Boyda, D.C. Hackett, G. Kanwar, K. Cranmer, S. Racanière, D.J. Rezende, P.E. Shanahan (2021), `2101.08176`

[17] M. Favoni, A. Ipp, D.I. Müller, D. Schuh, Phys. Rev. Lett. **128**, 032003 (2022), `2012.12901`

[18] P. de Haan, C. Rainone, M.C.N. Cheng, R. Bondesan (2021), `2110.02673`

[19] M. Gerdes, P. de Haan, C. Rainone, R. Bondesan, M.C.N. Cheng (2022), `2207.00283`

[20] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, *Neural Ordinary Differential Equations*, in *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Curran Associates, Inc., 2018), Vol. 31, `1806.07366`

[21] M. Lüscher, JHEP **08**, 071 (2010), [Erratum: JHEP 03, 092 (2014)], `1006.4518`

[22] K.G. Wilson, Phys. Rev. D **10**, 2445 (1974)

[23] K. He, X. Zhang, S. Ren, J. Sun, *Deep Residual Learning for Image Recognition*, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778