A Deeper Look into LieConv

An-Ya Olson [1] **Jonathan Palafoutas** [2]

Abstract

In this report we explore the architecture and model properties of LieConv, a convolutional layer that is equivariant to transformations from Lie groups developed by (Finzi et al., 2020). We generate a dynamical systems dataset for a Hamiltonian coupled spring-mass system and train four different models: a fully-connected network, and three group equivariant networks each with equivariance to a different Lie group. The dynamics model output by each of the four networks is used to generate predicted trajectories for a sample spring-mass system, and calculate total linear and angular momentum. We find that the group equivariant networks with T(2) and SO(2) equivariance (corresponding to translational and rotational equivariance, respectively), predict trajectories more accurately than the fully-connected and trivial group equivariant networks. In addition, we find that position-centering the data passed to a given group equivariant model leads to translation invariance and therefore conservation of linear momentum, even for models that are not inherently translation equivariant.

1. LieConv Overview

Our work is based on the LieConv convolutional layer developed in "Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data" (Finzi et al., 2020). In this section, we briefly outline the motivation for using LieConv and summarize the architecture of the original codebase which can be found at https://github.com/mfinzi/LieConv/tree/master.

[1]Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, New Jersey, USA [2]Program in Applied and Computational Mathematics, Princeton University, Princeton, New Jersey, USA. Correspondence to: An-Ya Olson <amolson@princeton.edu>, Jonathan Palafoutas <jpalafou@princeton.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1.1. Background

LieConv is a group convolutional layer that is equivariant to transformations from Lie groups which can be used in models applied to arbitrary continuous spatial data. Traditional convolutional layers are translation equivariant, meaning that when an input to the layer is translated, the output is translated in the same way. LieConv takes this a step further by introducing equivariance to other types of transformations such as rotations and scalings, many of which can be represented by Lie groups. Equivariance to such transformations is useful for learning and predicting symmetries in many kinds of spatial data, including images and molecular data, but here we focus on dynamical systems.

1.2. Datasets

LieConv contains built-in methods for generating different types of datasets, contained in `datasets.py`. To recreate the results from Finzi et. al, we generated a spring dynamics dataset that models $D = 10,000$ systems of $N = 6$ particles connected by springs. Each system's parameters, mass m and spring constant k, are randomly sampled from a uniform distribution. The state (consisting of a 2D position and 2D momentum) associated with each particle is randomly sampled from a normal distribution. Then, ground-truth trajectories for $t = [0, 5]$ seconds are generated using an RK4 numerical integration scheme that integrates the analytical Hamiltonian form of the dynamics over 500 timesteps. The states $\mathbf{z}_t$ for each of the 6 masses in each system are flattened into a single tensor with length 24 with the following form:

$$\mathbf{z}_t = \begin{bmatrix} q_{x,1} \\ q_{y,1} \\ \vdots \\ q_{x,N} \\ q_{y,N} \\ p_{x,1} \\ p_{y,1} \\ \vdots \\ p_{x,N} \\ p_{y,N} \end{bmatrix} \tag{1}$$

The final output of the dataset takes the form of 3 tensors: a state tensor `z` with shape [10000, 500, 24], a time tensor `t` with shape [10000, 500] and a system parameter tensor

`sysp` which contains the masses and spring constants for each of the sample systems with shape [10,000, 6, 2]. A sample trajectory of a spring-coupled mass system is depicted in Figure 1.

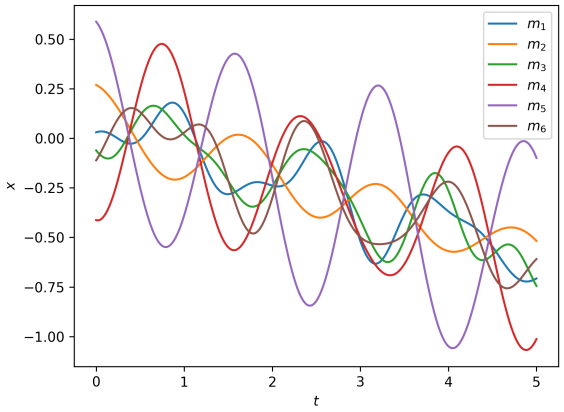

Figure 1. The evolution of the x coordinate of 6 masses connected by springs in two dimensions.

1.3. LieConv Model

Finzi designed a group convolutional layer, LieConv, which is equivariant to transformations from a specified Lie group (Finzi et al., 2020). The layer is implemented in a public GitHub repository and used to construct HLieResNet, a Lie group equivariant network that predicts the Hamiltonian dynamics of a system given a state. HLieResNet can be summarized as an estimator function

$$\hat{f}\left(\begin{bmatrix} q \\ p \end{bmatrix}\right) \sim \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial p} \\ -\frac{\partial \mathcal{H}}{\partial q} \end{bmatrix} \tag{2}$$

where the column vector is actually a batch of states at particular instances in time. System physical parameters such as masses and spring constants for the spring-coupled mass system may also be required to compute the Hamiltonian of a state $\mathcal{H}$. For the physical systems considered in this work, the Hamiltonian can be expressed as the sum of kinetic energy $\mathcal{T}$ and potential $\mathcal{V}$

$$\mathcal{H}(q,p) = \mathcal{T}(p) + \mathcal{V}(q). \tag{3}$$

HLieResNet can be evaluated in a loop with a numerical integrator to estimate the trajectory of a system given an initial state.

2. Our Code

In order to reproduce the results from (Finzi et al., 2020), we made several modifications to the original LieConv code, as well as wrote our own custom functions in Python to streamline the process of generating training data, training models with different Lie groups, evaluating the performance of the models, and plotting results. Our custom functions and scripts are described here:

- `dynamics.py` defines a function which returns a time-series of momentum from a trajectory input for each system
- `model_config.py` defines model and training parameters
- `model_integrator.py` defines a function which returns the predicted trajectory of a system provided a neural network as the dynamics function and an initial state
- `spring_trainer.py` defines a model trainer which generates and saves spring-coupled mass data if it isn't present
- `train_springs_FC.py` trains and saves a fully connected network from the spring data
- `train_springs_HLieResNet.py` trains and saves an HLieResNet from the spring data
- `run.py` choose a sample system from the dataset, plot the classically simulated ground truth, plot trajectory and momentum using the trained models' dynamics predictions

3. Results and Equivariance Demo

3.1. Trajectory Predictions

We trained and evaluated the performance four different models for comparison:

1. FC, a fully connected network
2. HLieResNet-Trivial, with group convolutional layers that are equivariant to the trivial group $\{\text{id}\}$
3. HLieResNet-T(2), with group convolutional layers that are equivariant to the 2D translations group T(2)
4. HLieResNet-SO(2), with group convolutional layers that are equivariant to the 2D rotations group SO(2)

All models consisted of two layers, each with 150 nodes. During training, we used a batch size of 1000 for 20 epochs for HLieResNet and 100 epochs for FC. The trajectory predictions for each of the models are shown in Figure 2.

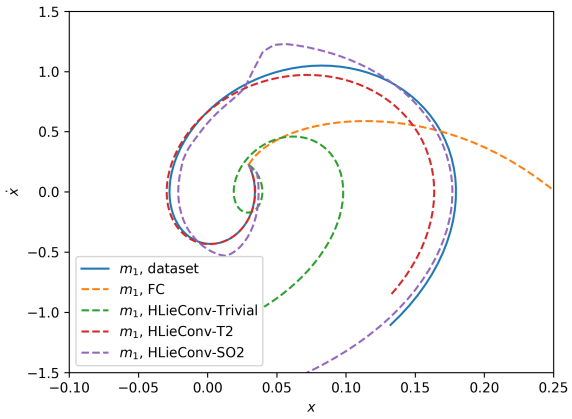

Figure 2. Trajectories simulated using dynamics from four different neural networks with centering enabled on the HLieConv networks.

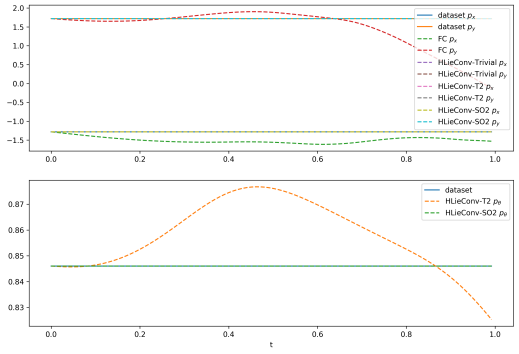

Figure 3. Momentum sums $\sum p_x$, $\sum p_y$ and $\sum p_\theta$ of the trajectories simulated using dynamics from four different neural networks with centering enabled on the HLieConv networks.

3.2. Conservation of Linear and Angular Mometum

A key result of (Finzi et al., 2020) is that group equivariance can directly inform the model's ability to preserve physical quantities like linear and angular momentum. The angular momentum of the i^{th} mass in a two-dimensional system p_θ is defined as

$$p_{\theta,i} = \vec{q_i} \times \vec{p_i} = q_{x,i}p_{y,i} - q_{y,i}p_{x,i} \tag{4}$$

The total linear and angular momentum of the sample system for each model are depicted in Figure 3. It was expected that only the T2-equivariant network would conserve the system's linear momentum. Yet, all the HLieConv networks, equivariant in the trivial group, T2, and SO2, conserved the system's linear momentum. Unsurpringly, only the SO2-equivariant network conserves angular momentum.

3.3. Disabling Centering

The linear momentum conservation of all HLieConv networks is due to mean centering of the position vectors, a standard operation for normalizing data in machine learning. A mean-centered position vector, denoted by a * with $q_i^* = q_i - \bar{q}_i$, is a T2-invariant transformation. Finzi shows that a Hamiltonian-predicting model is linear-momentum-conserving if and only if it is T2-invariant (Finzi et al., 2020). Thus, in order to see the expected conservation behavior, the same models were trained on the same data but with mean centering disabled in Figures 4 and 5. Note that the fully connected model in the implementation always has mean centering enabled, but it still fails to conserve either momentum.

Now, only the T2-equivariant model conserves linear momentum. T2-invariance was the requirement, but in this particular system the potenital depends only on the distances between masses. T2-equivariance preserves these distances and satisfies the requirement for linear momentum conservation.

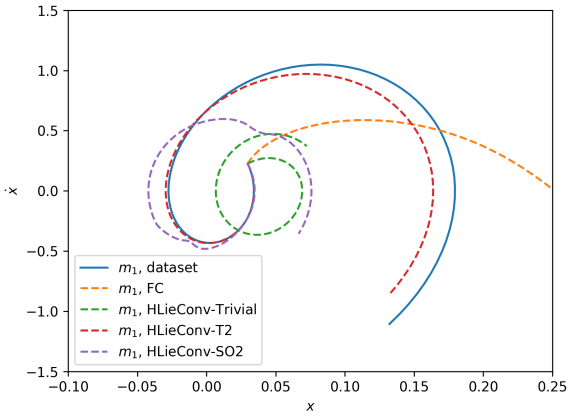

Figure 4. Trajectories simulated using dynamics from four different neural networks with centering disabled on the HLieConv networks.

In Figure 5, the SO2-equivariant model now fails to conserve linear momentum because SO2 is not equivariant to transformations from the T2 group. It correctly conserves angular momentum. Finzi similarly shows that a Hamiltonian-predicting model is angularly-momentum-conserving if and only if it is invariant to a global rotation of both the positions and momentums of the masses (Finzi et al., 2020).

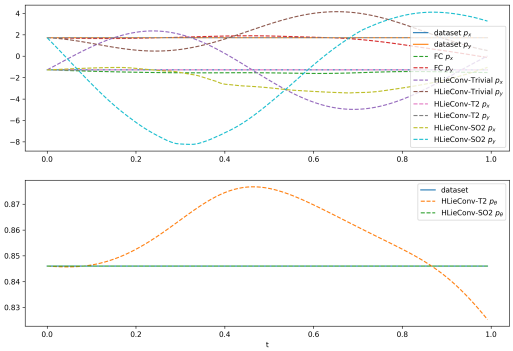

Figure 5. Momentum sums $\sum p_x$, $\sum p_y$ and $\sum p_\theta$ of the trajectories simulated using dynamics from four different neural networks with centering disnabled on the HLieConv networks.

Because SO2 is orthogonal, the SO2-equivariant model still preserves distances between the masses even if it isn't translation-invariant, thus satisfying the requirements for angular momentum conservation.

We reproduce Finzi's finding that complete momentum conservation is achieved by a SO2-equivariant model with mean centering enabled.

References

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data, 2020.