LieTransformer

Sam Bonnell, Junsu Kim, Patrick Chen

https://arxiv.org/abs/2012.10885

Outline

- Problem statement and Proposal
- Motivation
- Lie-Group
- Lifting
- Architecture
  - Lifting Layer
  - LieSelfAttention
    - Proof of Equivariance
- Experiments and Analysis
- Strengths and Weaknesses
- Question

Problem Statement and Proposal

Goal

- Design **self-attention** that preserves symmetry (**equivariance**) and output prediction under transformation (**invariance**).
- Build a Transformer architecture that works for **general Lie groups**, not only translations or rotations.

Limitations of Prior Work

- **Standard Transformer** [1]**:** ignores symmetry, must learn it from data.
- **TFN** [2] **/ SE(3)** [3] **models:** computationally heavy tensor operations.
- **LieConv** [4] **/ group CNNs** [5]**:** rely on convolution and complex neighborhood design.
- **Group-CNNs:** often limited to discrete or specific groups.

Why Build Group Equivariant Transformers?

Group equivariant transformers enable the observation of geometric symmetry through self-attention.

Many systems have implicit symmetry:

- point clouds,
- images.

[*] https://medium.com/data-science/group-equivariant-self-attention-19e47f0b786e

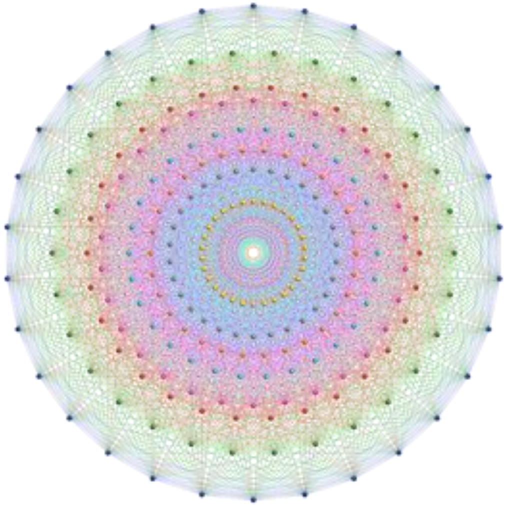
What is a Lie Group?

Lie groups are groups that encode continuous symmetries:

- transformations
- rotations,
- orientation,
- etc.

Lie groups provide a clean framework to deal with symmetries across a number of different problems.

- point clouds,
- image rotations,
- etc.



[*] https://en.wikipedia.org/wiki/Lie_group#/media/File:E8Petrie.svg

Lie Groups

A Lie group is a group that is also a smooth manifold, where group operations are smooth/differentiable maps. As a group, a Lie group must obey the following axioms:

Closure	$\forall g, h \in G, \quad g \cdot h \in G$
Associativity	$\forall g, h, k \in G, \quad (g \cdot h) \cdot k = g \cdot (h \cdot k)$
Identity	$\exists e \in G$ such that $e \cdot g = g \cdot e = g$ for all $g \in G$
Inverse	$\forall g \in G,\ \exists g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = e$

Lie Groups

To qualify as a Lie group, three additional conditions must be satisfied:

Smooth Manifold Structure:

Manifold	G is a smooth (differentiable) manifold

Compatibility Conditions:

Smooth Group Operation	The map $\mu : G \times G \to G$ given by $\mu(g, h) = g \cdot h$ is smooth
Smooth Inverse Map	The map $\iota : G \to G$ given by $\iota(g) = g^{-1}$ is smooth

Lie Groups - Example

The Special Orthogonal Group, SO(2) is a Lie Group.

First, we "show" it is a manifold:

$$\mathrm{SO}(2) = \{R \in \mathbb{R}^{2\times 2} \mid R^T R = I,\ \det\ R = 1\}$$

Each element of SO(2) can be uniquely written as:

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \quad \theta \in [0, 2\pi)$$

Maps between the S^1 manifold and SO(2) can be constructed:

$$\phi : S^1 \to \mathrm{SO}(2), \quad (x, y) \mapsto \begin{pmatrix} x & -y \\ y & x \end{pmatrix} \qquad R \mapsto (R_{11}, R_{21})$$

Therefore, $\mathrm{SO}(2) \cong S^1$

Lie Groups - Example

Next, we “show” that it is a group:

Closure: $R_1, R_2 \in \mathrm{SO}(2)$

$$(R_1 R_2)^T (R_1 R_2) = R_2^T R_1^T R_1 R_2 = R_2^T I R_2 = I \qquad \det(R_1 R_2) = \det(R_1)\det(R_2) = 1$$

$$R_1 R_2 \in \mathrm{SO}(2)$$

Associativity: $(R_1 R_2) R_3 = R_1 (R_2 R_3)$

Identity:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad I^T I = I, \quad \det I = 1 \qquad I \in \mathrm{SO}(2)$$

Inverse:

$$R^T R = I \longrightarrow R^{-1} = R^T \qquad \det\left(R^{-1}\right) = (\det(R))^{-1} = 1$$

$$R^{-1} \in \mathrm{SO}(2)$$

Lifting to a Lie Group

Lifting is the process of transforming the data from the input domain onto a group element:

$$(x_i, \mathbf{f}_i) \mapsto (g, \mathbf{f}_i) \text{ for } g \in s(x_i)H$$

Mapping each input onto a Lie group element:

- symmetries become group operations on the representation,
- removes the requirement for data augmentation,
- enables equivariant operations on the data.

Lifting to a Lie Group

Formally, lifting defines cosets that partition the group into disjoint subsets, corresponding to the original spatial input data.

$$H = \{g \in G \mid gx_0 = x_0\} \qquad s(x) : x_0 \mapsto x, \quad s(x) \in G$$

$$s(x_i)H \cap s(x_j)H = \emptyset, \qquad x_i \neq x_j$$

Lifting onto a Lie Group

From the (not shown) example, lifted data is defined as:

$$g \in s(x_i)H = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta & t_{x_1} \\ \sin\theta & \cos\theta & t_{x_2} \\ 0 & 0 & 1 \end{pmatrix} : \theta \in [0, 2\pi) \right\}$$

Operating on the group, relative group elements are invariant under left group action:

$$M_{ij} = g_i^{-1} g_j = (ug_i)^{-1}(ug_j) = g_i^{-1} u^{-1} u g_j = g_i^{-1} g_j$$

The lifted coset is equivariant under group action:

$$s(ux_i)H = us(x_i)H \; \forall u \in G$$

This implies that lifting is equivariant under group action.

A rotated input will result in a rotated self-attention field.

LieTransformer Architecture

Phase One - Representation

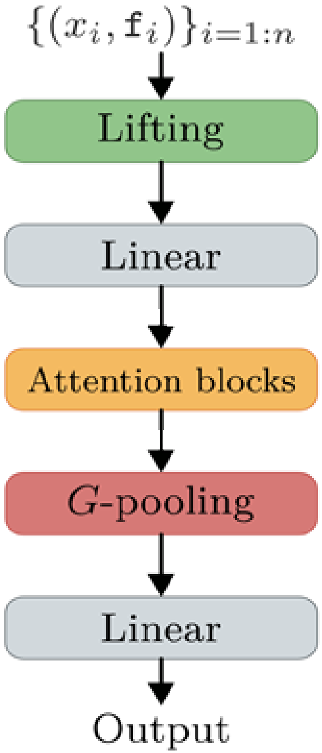
- Input
- Lifting

Phase Two - Processing

- Linear
- Self-attention

Phase Three - Prediction

- G-pooling - averaging over group features
- Linear
- Output

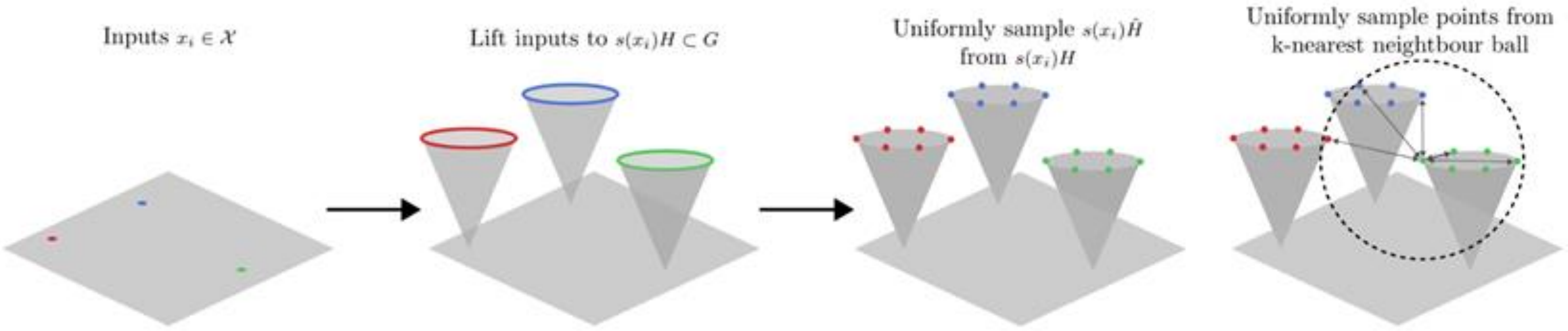


Lifting Layer

The lifting layer consists of three steps:

1. Lift data onto the group
2. Uniformly sample the cosets of each data point - **continuous** to **discrete**
3. Uniformly sample points using the k-nearest neighbour algorithm

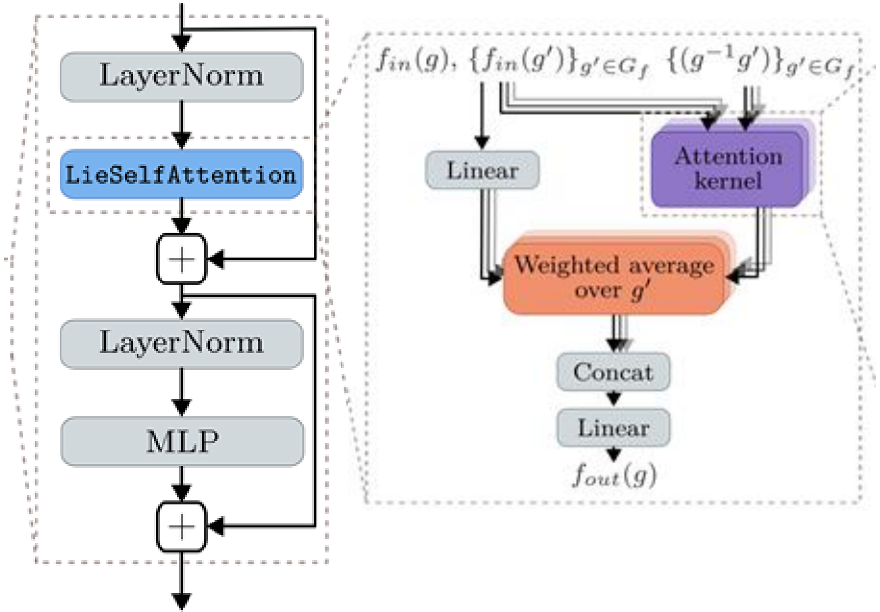
These k-nearest neighbours define the data for self-attention, etc.



LieTransformer Architecture - Attention Blocks

Basic Structure of LieSA Block

- Layer Normalization
- Lie Self-Attention
- Residual connection
- Feed-forward network (MLP)
- Residual connection



Self-Attention: SE(3) / TFN vs LieTransformer

SE(3) / TFN

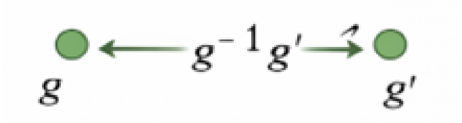
Point in Euclidean Space

Geometry indexed by Euclidean Coords

$$m_{ij} = \Phi(f_j, x_j - x_i)$$

LieTransformer

Features indexed by Group Elements



Geometry indexed by Group Elements

$$m_{g,g'} = \Phi(f(g'), g^{-1}g')$$

Key Difference

	SE(3) / TFN	**LieTransformer**
Input Space	Euclidean Points	Group Elements
Geometry	$r_{ij} = x_j - x_i$	$g^{-1}g'$

Lie Self-Attention

$$f_{\text{out}}(g) = \int_{G_f} w_f(g, g') W^V f(g')\, dg'$$

Steps:

1. Content attention

$$k_c(f(g), f(g'))$$

1. Location attention

$$k_l\left(g^{-1} g'\right)$$

1. Combine

$$\alpha_f(g, g') = F(k_c, k_l)$$

1. Normalize

$$w_f(g, g') = \text{norm}\left(\alpha_f(g, g')\right)$$

1. Output (Monte Carlo Sum)

$$f_{\text{out}}(g) = \int_{G_f} w_f(g, g') W^V f(g')\, dg' = \sum_{m=1}^{M} w_f(g, g_m') W^V f(g_m')$$

Algorithm 1 LieSelfAttention

Input: $(g, f(g))_{g \in G_f}$

for $g \in G_f$

  for $g' \in G_f$ (or $\text{nbhd}_\eta(g)$)

    ▷ Compute content/location attention

    $k_c(f(g), f(g')), k_l(g^{-1} g')$

    ▷ Compute unnormalised weights

    $\alpha_f(g, g') = F(k_c(f(g), f(g')), k_l(g^{-1} g'))$

  ▷ Compute normalised weights and output

  $\{w_f(g, g')\}_{g' \in G_f} = \text{norm}\{\alpha_f(g, g')\}_{g' \in G_f}$

  $f_{out}(g) = \int_{G_f} w_f(g, g') W^V f(g') dg'$

Output: $(g, f_{out}(g))_{g \in G_f}$

Attention Kernel

$$f_{\text{out}}(g) = \int_{G_f} w_f(g, g') W^V f(g') \, dg'$$

Instead of tokens at positions, we now have: $(g, f(g))$

- **g** = group element (location/orientation)
- **f(g)** = feature

Content attention: $k_c(f(g), f(g'))$

- Do features match?

Location attention: $k_l(g^{-1}g')$

- How are positions related?

Combine & Normalization

$$\alpha(g, g') = F(k_c, k_l)$$

$$w_f(g, g') = \text{norm}(\alpha(g, g'))$$

$k_c(f(g), f(g'))$

Options:

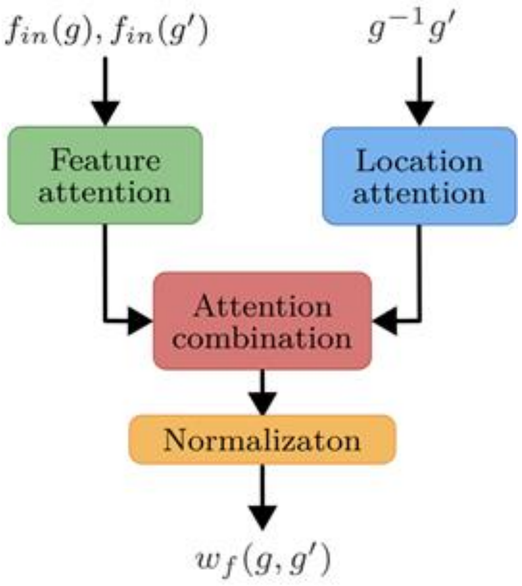
- Dot product: $\frac{1}{\sqrt{d_v}} (W^Q f(g))^T (W^K f(g'))$
- Concat: $Concat(W^Q f(g), W^K f(g'))$

$k_l(g^{-1} g')$

Options:

- Plain: $\nu[\log(g^{-1} g')]$
- Learned: $MLP(\nu[\log(g^{-1} g')])$

LieTransformer Architecture - Attention Kernel



$$\alpha(g, g') = F(k_c, k_l)$$

Combining Content & Location Attention

Kernel Combination *F(...)*

- Additive: $k_c + k_l$
- MLP: $\mathrm{MLP}([k_c, k_l])$
- Multiplicative $k_c \times k_l$

Attention depends on feature similarity and relative group transformations

Proof of Equivariance

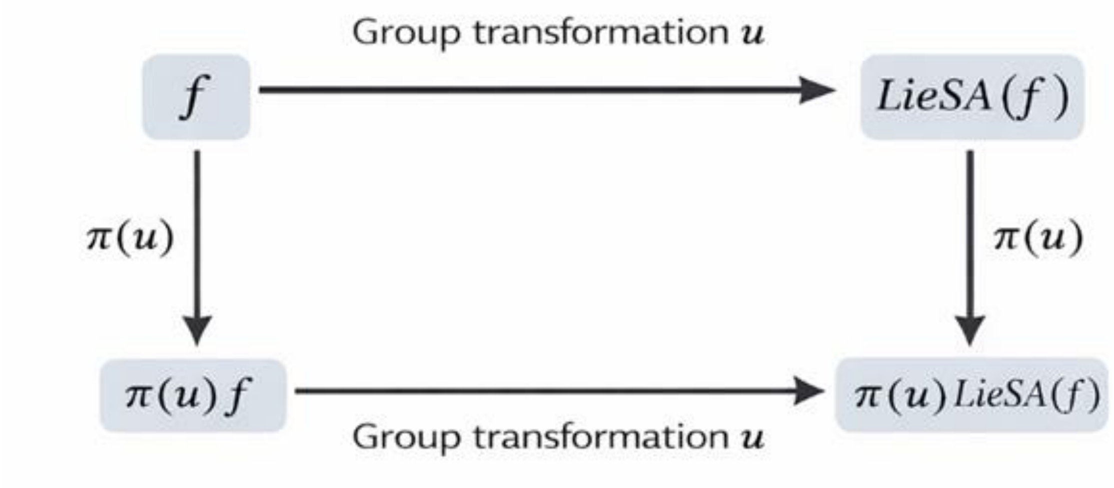
$$(\mathrm{LieSA}(f))(x) = \sum_{h} \mathrm{norm}\Big(F\big(k_c(f(x), f(h)), k_l(x^{-1}h)\big)\Big) W f(h).$$

Problem:

Let G be a Lie group, $u, g, g' \in G$

Let $f : G \rightarrow R^d$, Define the group action on functions by $(\pi(u)f)(g) = f(u^{-1}g)$

Prove equivariance: $LieSA(\pi(u)f) = \pi(u)\, LieSA(f)$



Proof of Equivariance

$$(\mathrm{LieSA}(f))(x) = \sum_h \mathrm{norm}\big(F\big(k_c(f(x), f(h)), k_l(x^{-1}h)\big)\big)\, W f(h).$$

Prove equivariance: $LieSA(\pi(u) f) = \pi(u)\, LieSA(f)$

LHS: $$\big(\mathrm{LieSA}(\pi(u)f)\big)(g) = \sum_{g'} w_{\pi(u)f}(g, g')\, W^v \big(\pi(u)f\big)(g')$$

$$= \sum_{g'} \mathrm{norm}\big(F\big(k_c(f(u^{-1}g), f(u^{-1}g')), k_\ell(g^{-1}g')\big)\big)\, W^v f(u^{-1}g')$$

$Let\ h = u^{-1}g'\ (equivalently\ g' = uh).\ Then\ f(u^{-1}g') = f(h)$

$$= \sum_h \mathrm{norm}\big(F\big(k_c(f(u^{-1}g), f(h)), k_\ell((u^{-1}g)^{-1}h)\big)\big)\, W^v f(h)$$

$$= \big(\mathrm{LieSA}(f)\big)(u^{-1}g)$$

$$= \pi(u)\, \mathrm{LieSA}(f)$$

Experiments Overview

Datasets and Tasks

- Counting Shapes in 2D Point Clouds: **T(2)** and **SE(2)** Invariance
- QM9 Molecular Property Regression: **T(3)** and **SE(3)** Invariance
- Modeling Particle Trajectories under Hamiltonian Dynamics: **SE(2)** Invariance

Experimental Setup: 2D Point Clouds

Task: Classify the number of instances for various shapes

Dataset Details:

- Input: 2D coordinate information
- Include triangles, squares, pentagons, “L” shapes
- 10k training and 1k test samples with T(2) and SE(2) augmentation

Baseline: SetTransformer [6], ~1M params

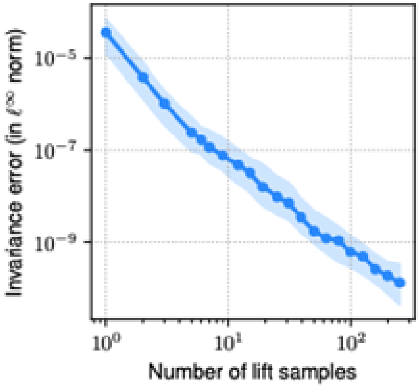
Architecture:

- T(2) and SE(2) variants
- 10 attention layers, 8 heads, 128 feature dimension, ~1M params
- 1 lift sample per point for SE(2)

[*] https://github.com/oxcsml/lie-transformer

Results: 2D Point Clouds

Training data / Test data	D_{train} / D_{test}	D_{train} / D_{test}^{T2}	D_{train} / D_{test}^{SE2}	D_{train}^{T2} / D_{test}^{T2}	D_{train}^{T2} / D_{test}^{SE2}	D_{train}^{SE2} / D_{test}^{SE2}
SetTransformer	0.58 ± 0.07	0.44 ± 0.02	0.44 ± 0.02	0.61 ± 0.02	0.51 ± 0.01	0.55 ± 0.01
LieTransformer-T2	$\mathbf{0.75} \pm 0.03$	$\mathbf{0.75} \pm 0.03$	0.63 ± 0.06	$\mathbf{0.75} \pm 0.03$	0.63 ± 0.06	0.70 ± 0.03
LieTransformer-SE2	0.71 ± 0.01	0.71 ± 0.01	$\mathbf{0.69} \pm 0.02$	0.71 ± 0.01	$\mathbf{0.69} \pm 0.02$	$\mathbf{0.72} \pm 0.04$



Accuracy remains unchanged with T(2) and minimal changes in SE(2) augmentation

Experimental Setup: QM9 Molecular Property Regression

Task: Regression on molecular property using position information

Dataset Details:

- Input: 3D atomic position and charge information
- 100K training samples with SO3 augmentation
- 10K test samples

Main Baseline: LieConv [4]

Architecture:

- T(3) and SE(3) variants
- 13 (T3) and 30 (SE3) attention layers, 8 heads, 848 feature dimension
- 2 lift samples for SE(3)

[*] https://github.com/oxcsml/lie-transformer

Results: QM9 Molecular Property Regression

Upper:
- Non-invariant
- Designed for QM9

Middle:
- Invariant
- Designed for QM9

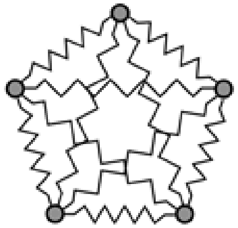
Lower:
- Invariant
- General-purpose

Task Units	α bohr^3	$\Delta\epsilon$ meV	ϵ_{HOMO} meV	ϵ_{LUMO} meV	μ D	C_ν cal/mol K	G meV	H meV	R^2 bohr^2	U meV	U_0 meV	ZPVE meV
WaveScatt (Hirn et al., 2017)	.160	118	85	76	.340	.049	–	–	–	–	–	–
NMP (Gilmer et al., 2017)	**.092**	69	43	38	**.030**	.040	19	17	.180	20	20	**1.50**
SchNet (Schütt et al., 2017)	.235	**63**	**41**	**34**	.033	**.033**	**14**	**14**	**.073**	**19**	**14**	1.70
Cormorant (Anderson et al., 2019)	.085	61	34	38	.038	.026	20	21	.961	21	22	2.03
DimeNet++ (Klicpera et al., 2020) *	**.049**	**34**	**26**	**20**	**.033**	**.024**	**8**	**7**	.387	**7**	**7**	**1.23**
L1Net (Miller et al., 2020)	.088	68	45	35	.043	.031	14	14	**.354**	14	13	1.56
TFN (Thomas et al., 2018)	.223	58	40	38	.064	.101	–	–	–	–	–	–
SE3-Transformer (Fuchs et al., 2020)	.148	53	36	33	.053	.057	–	–	–	–	–	–
LieConv-T3 (Finzi et al., 2020) †	.125	60	36	32	.057	.046	35	37	1.54	36	35	3.62
LieConv-T3 + SO3 Aug (Finzi et al., 2020)	.084	49	30	**25**	**.032**	.038	22	24	.800	19	19	2.28
LieConv-SE3 (Finzi et al., 2020)†	.097	**45**	**27**	**25**	.039	.041	39	46	2.18	49	48	3.27
LieConv-SE3 + SO3 Aug (Finzi et al., 2020)†	.088	**45**	**27**	**25**	.038	.043	47	46	2.12	44	45	3.25
LieTransformer-T3 (Us)	.179	67	47	37	.063	.046	27	29	.717	27	28	2.75
LieTransformer-T3 + SO3 Aug (Us)	**.082**	51	33	27	.041	**.035**	**19**	**17**	**.448**	**16**	**17**	**2.10**
LieTransformer-SE3 (Us)	.104	52	33	29	.061	.041	23	27	2.29	26	26	3.55
LieTransformer-SE3 + SO3 Aug (Us)	.105	52	33	29	.062	.041	22	25	2.31	24	25	3.67

Achieve the best performance on 8 tasks among general-purpose SE(3) invariant models

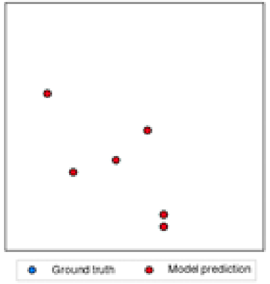
Experimental Setup: Particle Trajectories under Hamiltonian

Task: Predict particle trajectories with random position and momentum



Dataset Details:

- Input: 2D position, momentum, and mass at a timestep *t*
- Output: Hamiltonian, H(x(t))
- Simulate 500 timesteps and train using 5 random subsequences



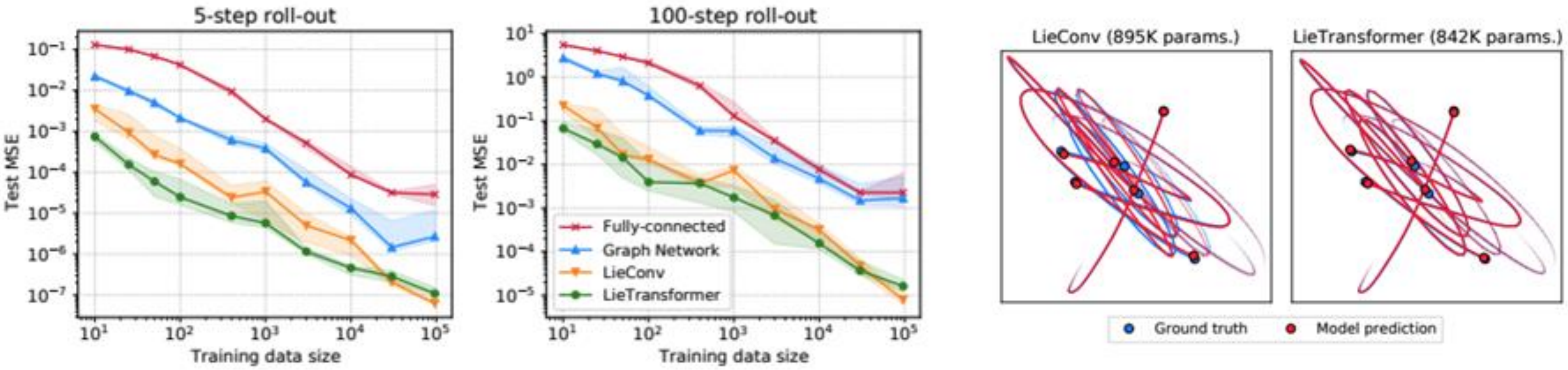
Baseline: LieConv [4] (Main), Fully Connected NN, Graph NN

Architecture:

- 5 attention layers, 8 heads, 160 feature dimension, ~0.9M params
- 2 lift samples for SE(2)

[*] https://people.eecs.berkeley.edu/~sequin/CS184/TOPICS/SpringMass/Spring_c.html
[**] https://github.com/oxcsml/lie-transformer

Results: Particle Trajectories under Hamiltonian



LieTransformer consistently outperforms prior work with various dataset size and the level of difficulty

Strengths and Weaknesses

Strengths

1. Generally applicable to all translation groups
2. Controllable invariance performance via lift sampling

Weaknesses

1. Pseudo-equivariance due to sampling
2. Computation & memory cost for Lie self-attention
    a. Time Complexity: from $O(nbhd^2)$ to $O(gnbhd^2)$
    b. Memory Complexity: $O(gnbhd)$ to store lift samples
3. Workload-specific Model Architecture: Painstaking manual architecture search

Q & A

Reference

[1] M. Hutchinson, C. L. Lan, S. Zaidi, E. Dupont, Y. W. Teh, and H. Kim, “LieTransformer: Equivariant self-attention for Lie Groups,” Jun. 16, 2021, *arXiv*: arXiv:2012.10885. doi: 10.48550/arXiv.2012.10885.

[2] M. Weiler, F. A. Hamprecht, and M. Storath, “Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds,” Dec. 3, 2018, arXiv: arXiv:1802.08219.

[3] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling, “SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks,” Jul. 14, 2020, arXiv: arXiv:2006.10503.

[4] Finzi M, Stanton S, Izmailov P, Wilson AG. “Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data”. International conference on machine learning (ICML) 2020 Nov 21 (pp. 3165-3176). PMLR.

[5] Cohen T, Welling M. :Group equivariant convolutional networks”. International conference on machine learning (ICML) 2016 Jun 11 (pp. 2990-2999). PMLR.

[6] Lee J, Lee Y, Kim J, Kosiorek A, Choi S, Teh YW. “Set transformer: A framework for attention-based permutation-invariant neural networks”. International conference on machine learning (ICML) 2019 May 24 (pp. 3744-3753). PMLR.