Iterative SE(3)-Transformers

Fabian B. Fuchs [†1], Edward Wagstaff [†1], Justas Dauparas[2], and Ingmar Posner[1]

[1] Department of Engineering Science, University of Oxford, Oxford, UK
[2] Institute for Protein Design, University of Washington, WA, USA

† These authors contributed equally

{fabian,ed}@robots.ox.ac.uk

Abstract. When manipulating three-dimensional data, it is possible to ensure that rotational and translational symmetries are respected by applying so-called *SE(3)-equivariant* models. Protein structure prediction is a prominent example of a task which displays these symmetries. Recent work in this area has successfully made use of an SE(3)-equivariant model, applying an iterative SE(3)-equivariant attention mechanism. Motivated by this application, we implement an iterative version of the SE(3)-Transformer, an SE(3)-equivariant attention-based model for graph data. We address the additional complications which arise when applying the SE(3)-Transformer in an iterative fashion, compare the iterative and single-pass versions on a toy problem, and consider why an iterative model may be beneficial in some problem settings. We make the code for our implementation available to the community[3].

Keywords: Deep Learning · Equivariance · Graphs · Proteins

1 Introduction

Tasks involving manipulation of three-dimensional (3D) data often exhibit rotational and translational symmetry, such that the overall orientation or position of the data is not relevant to solving the task. One prominent example of such a task is protein structure refinement [1]. The goal is to improve on the initial 3D structure – the position and orientation of the structure, i.e. the frame of reference, is not important to the goal. We would like to find a mapping from the initial structure to the final structure such that if the initial structure is rotated and translated then the predicted final structure is rotated and translated in the same way. This symmetry between input and output is known as *equivariance*. More specifically, the group of translations and rotations in 3D is called the special Euclidean group and is denoted by SE(3). The relevant symmetry is known as SE(3) equivariance.

In the latest Community-Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction (CASP14) structure-prediction challenge, DeepMind's AlphaFold 2 team [2] successfully applied machine learning

[3] https://github.com/FabianFuchsML/se3-transformer-public

techniques to win the categories "regular targets" and "interdomain prediction" by a wide margin. This is a important achievement as it opens up new routes for understanding diseases and drug discovery in cases where protein structures cannot be experimentally determined [3]. At this time, the full implementation details of the AlphaFold 2 algorithm are not public. Consequently, there is a lot of interest in understanding and reimplementing AlphaFold 2 [4–8].

One of the core aspects that enabled AlphaFold 2 to produce very high quality structures is the end-to-end iterative refinement of protein structures [2]. Concretely, the inputs for the refinement task are the estimated coordinates of the protein, and the outputs are updates to these coordinates. This task is equivariant: when rotating the input, the update vectors are rotated identically. To leverage this symmetry, AlphaFold 2 uses an SE(3)-equivariant attention network. The first such SE(3)-equivariant attention network described in the literature is the SE(3)-Transformer [9]. However, in the original paper, the SE(3)-Transformer is only described as a single-pass predictor, and its use in an iterative fashion is not considered.

In this paper, we present an implementation of an iterative version of the SE(3)-Transformer, with a discussion of the additional complications which arise in this iterative setting. In particular, the backward pass is altered, as the gradient of the loss with respect to the model parameters could flow through basis functions. We conduct toy experiments to compare the iterative and single-pass versions of the architecture, draw conclusions about why this architecture choice has been made in the context of protein structure prediction, and consider in which other scenarios it may be a useful choice. The code will be made publicly available[3].

2 Background

In this section we provide a brief overview of SE(3) equivariance, with a description of some prominent SE(3)-equivariant machine learning models. To situate this discussion in a concrete setting, we take motivation from the task of protein structure prediction and refinement, which is subject to SE(3) symmetry.

2.1 Protein Structure Prediction and Refinement

In protein structure prediction, we are given a target sequence of amino acids, and our task is to return 3D coordinates of all the atoms in the encoded protein. Additional information is often needed to solve this problem – the target sequence may be used to find similar sequences and related structures in protein databases first [10, 11]. Such coevolutionary data can be used to predict likely interresidue distances using deep learning – an approach that has been dominating protein structure prediction in recent years [12–14]. Coevolutionary information is encoded in a multiple sequence alignment (MSA) [15], which can be used to learn pairwise features such as residue distances and orientations [14]. These pairwise features are constraints on the structure of the protein, which

inform the prediction of the output structure. One could start with random 3D coordinates for the protein chain and use constraints from learnt pairwise features to find the best structure according to those constraints. The problem can then be approached in an iterative way, by feeding the new coordinates back in as inputs and further improving the structure. This iterative approach can help to improve predictions [16].

Importantly, MSA and pairwise features do not include a global orientation of the protein – in other words, they are invariant under rotation of the protein. This allows for the application of SE(3)-equivariant networks, which respect this invariance by design, as done in AlphaFold 2 [2]. The predictions of an SE(3)-equivariant network, in this case predicted shifts to backbone and side chain atoms, are always relative to the arbitrary input frame of reference, without the need for data augmentation. SE(3)-equivariant networks may also be applied in an iterative fashion, and when doing so it is possible to propagate gradients through the whole structure prediction pipeline. This full gradient propagation contrasts with the disconnected structure refinement pipeline of the first version of AlphaFold [12].

2.2 Equivariance and the SE(3)-Transformer

A function, task, feature[4], or neural network is *equivariant* if transforming the input results in an equivalent transformation of the output. Using rotations $\mathbf{R}$ as an example, this condition reads:

$$f(\mathbf{R} \cdot \vec{x}) = \mathbf{R} \cdot f(\vec{x}) \tag{1}$$

In the following, we will focus on 3D rotations only – this group of rotations is denoted SO(3). Adding translation equivariance, i.e. going from SO(3) to SE(3), is easily achieved by considering relative positions or by subtracting the center of mass from all coordinates.

A set of data relating to points in 3D may be represented as a graph. Each node has spatial coordinates, as well as an associated feature vector which encodes further relevant data. A node could represent an atom, with a feature vector describing its momentum. Each edge of the graph also has a feature vector, which encodes data about interactions between pairs of nodes. In an equivariant problem, it is crucial that equivariance applies not only to the positions of the nodes, but also to all feature vectors – for SO(3) equivariance, the feature vectors must rotate to match any rotation of the input. To distinguish such equivariant features from ordinary neural network features, we refer to them as *fibers*.

A fiber could, for example, encode momentum and mass as a 4 dimensional vector, formed by concatenating the two components. A momentum vector would typically be 3 dimensional (also called *type-1*), and is rotated by a 3x3 matrix. The mass is scalar information (also called *type-0*), and is invariant to rotation.

[4] A *feature* of a neural network is an input or output of any layer of the network.

The concept of *types* comes from representation theory, where a type-ℓ feature is rotated by a $(2\ell+1) \times (2\ell+1)$ Wigner-D matrix.[5] Because a fiber is a concatenation of features of different types, the entire fiber is rotated by a block diagonal matrix, where each block is a Wigner-D matrix. [17–19]

At the input and output layers, the fiber structure is determined by the task at hand. For the intermediate layers, arbitrary fiber structures can be chosen. In the following, we denote the structure of a fiber as a dictionary. E.g. a fiber with 3 scalar values (e.g. RGB colour channels) and one velocity vector has 3 type-0 and 1 type-1 feature: `{0:3, 1:1}`. This fiber is a feature vector of length 6.

There is a large and active literature on machine learning methods for graph data, most importantly graph neural networks [20–24]. The SE(3)-Transformer [9] in particular is a graph neural network explicitly designed for SE(3)-equivariant tasks, making use of the fiber structure and Wigner-D matrices discussed above to enforce equivariance of all features at every layer of the network.

Alternative Approaches to Equivariance: The Wigner-D matrix approach[6] at the core of the SE(3)-Transformer is based on closely related earlier works [17–19]. In contrast, Cohen et al. [25] introduced rotation equivariance by storing copies corresponding to each element of the group in the hidden layers – an approach called regular representations. This was constrained to 90 degree rotations of images. Two recent regular representation approaches [26, 27] extend this to continuous data by sampling the (infinite) group elements and map the group elements to the corresponding Lie group to achieve a smoother representation.

2.3 Equivariant Attention

The second core aspect of an SE(3)-Transformer layer is the self-attention [28] mechanism. This is widely used in machine learning [21,29–34] and based on the principle of keys, queries and values – where each is a learned embedding of the input. The word 'self' describes the fact that keys, queries and values are derived from the same context. In graph neural networks, this mechanism can be used to have nodes attend to their neighbours [21,28,35–37]. Each node serves as a focus point and queries information from the surrounding points. That is, the feature vector f_i of node i is transformed via an equivariant mapping into a query q_i. The feature vectors f_j of the surrounding points j are mapped to equivariant keys k_{ij}[7]. A scalar product between key and query – together with a softmax normalisation – gives the attention weight. The scalar product of two rotation equivariant features of the same type gives an invariant feature. Multiplying the invariant weights w_{ij} with the equivariant values v_{ij} gives an equivariant output.

[5] We can think of type-0 features as rotating by the 1×1 rotation matrix (1).

[6] This approach rests on the theory of *irreducible representations [17, 18]*.

[7] Note that often, the keys and values do not depend on the query node, i.e. $k_{ij} = k_j$. However, in the SE(3)-Transformer, keys and values depend on the relative position between query i and neighbour j as well as on the feature vector f_j.

3 Implementation of an Iterative SE(3)-Transformer

Here, we describe the implementation of the iterative SE(3)-Transformer covering multiple aspects such as gradient flow, equivariance, weight sharing and avoidance of information bottlenecks.

3.1 Gradient flow in Single-Pass vs. Iterative SE(3)-Transformers

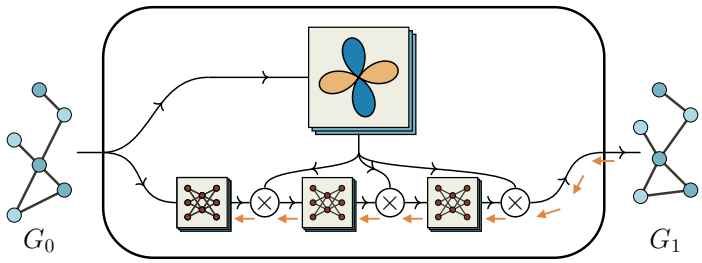

Fig. 1: Gradient flow (orange) in a conventional, single-pass SE(3)-Transformer mapping from a graph (left) to an updated graph (right). The equivariant basis kernels (top) do not have to be differentiated as there is no gradient flow through them.

At the core of the SE(3)-Transformer are the kernel matrices $\mathbf{W}(\vec{x}_j - \vec{x}_i)$ which form the equivariant linear mappings used to obtain keys, queries and values in the attention mechanism. These matrices are a linear combination of basis matrices. The weights for this linear combination are the output of trainable neural networks. Importantly, the basis matrices are not learnable. They are defined by spherical harmonics and Clebsch-Gordan coefficients, and depend only on the relative positions in the input graph G_0 [9].

Typically, equivariant networks have been applied to tasks where 3D coordinates in the input are mapped to an invariant or equivariant output in a single pass. This means that the relative positions of nodes do not change until the final output, and the basis matrices therefore remain constant. Gradients do not flow through them, and the spherical harmonics do not have to be differentiated, as can be seen in Fig. 1.

When applying the SE(3)-Transformer in an iterative fashion (see Fig. 2), each block i outputs an updated graph G_i. This allows for, e.g., re-evaluating the interactions or binary potentials between two nodes. Now the relative positions, and therefore the basis matrices, are no longer constant until the final output, and gradients flow through the basis. The spherical harmonics used to construct the basis are smooth functions, and therefore backpropagating through them is possible. We provide code which implements this backpropagation.

3.2 Hidden Representations Between Blocks

In the simplest case, each SE(3)-Transformer block outputs a single type-1 feature per point, which is then used as a relative update to the coordinates before

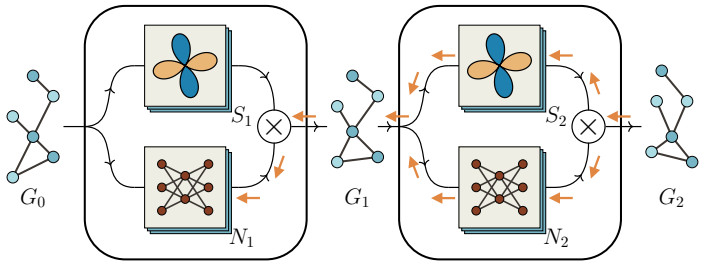

Fig. 2: Gradient flow (orange) in an iterative SE(3)-Transformer with multiple position updates. Now the gradients do flow through the basis construction (S_2) meaning this part has to be differentiated.

applying the second block. This, however, introduces a bottleneck in the information flow. Instead, we choose to maintain the dimensionality of the hidden features. A typical choice would be to have the same number of channels (e.g. 4) for each feature type (e.g., up to type 3). In this case the hidden representation reads {`0:4,1:4,2:4,3:4`}. We then choose this same fiber structure for the outputs of each SE(3)-Transformer block (except the last) and the inputs to the blocks (except the first). This way, the amount of information saved in the hidden representations is constant throughout the entire network.

3.3 Weight Sharing

If each iteration is expected to solve the same sub-task, weight sharing can make sense in order to reduce overfitting. The effect on memory consumption and speed should, however, be negligible during training, as the basis functions still have to be evaluated and activations have to evaluated and stored separately for each iteration. In our implementation, we chose not to share weights between different SE(3)-Transformer blocks. The number of trainable parameters hence scales linearly with the number of iterations. In theory, this enables the network to leverage information gained in previous steps for deciding on the next update. One benefit of this choice is that it facilitates using larger fibers between the blocks as described in 3.2 to avoid information bottlenecks. A downside is that it fixes the number of iterations of the SE(3)-Transformer.

3.4 Gradient Descent

In this paper, we will apply the SE(3)-Transformer to an energy optimisation problem, loosely inspired by the protein structure prediction problem. For convex optimsation problems, gradient descent is a simple yet effective algorithm. However, long amino acid chains with a range of different interactions can be assumed to create many local minima. Our hypothesis is that the SE(3)-Transformer is better at escaping the local minimum of the starting configuration and likely to find a better global minimum. We add an optional post-processing step of gradient descent, which optimises the configuration within the minimum that the

SE(3)-Transformer found. In real-world protein folding, these two steps do not have to use the same potential. Gradient descent needs a differentiable potential, whereas the SE(3)-Transformer is not subject to that constraint.

4 Experiments

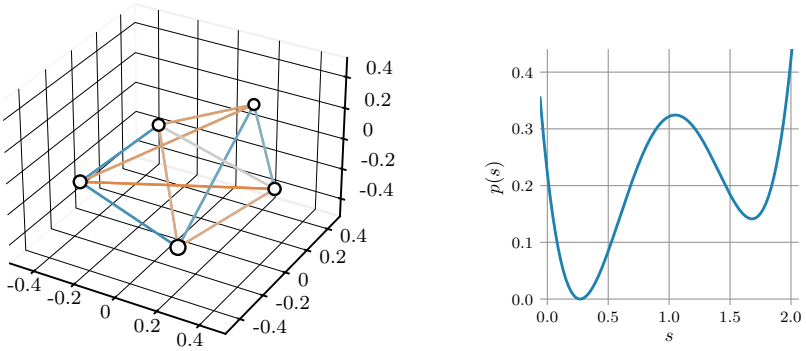

Fig. 3: The left plot shows a configuration of five nodes, with the gradient of the potential between each pair represented by the colour of the edges. Blue edges indicate repulsion and orange edges indicate attraction. Stronger colour represents stronger interaction. The right plot shows the double-minimum potential $p(s)$, with parameter $a = 0$.

We study a physical toy problem as a proof-of-concept and to get insights into what type of tasks could benefit from iterative predictions. We consider an energy minimisation problem with 10 particles, which we will refer to as nodes. Each pair (n_i, n_j) of nodes in the graph interacts according to a potential $p_{ij}(r_{ij})$, where r_{ij} is the distance between the nodes. The goal is to minimise the total value of the potential across all pairs of nodes in the graph.

We choose a pairwise potential with two local minima. This creates a complex global potential landscape that is not trivially solvable for gradient descent:

$$s_{ij} = r_{ij} - a_{ij} - 1 \tag{2}$$

$$p_{ij}(s_{ij}) = s_{ij}^4 - s_{ij}^2 + \frac{s_{ij}}{10} + p_{\min} \tag{3}$$

Here $p_{\min} \approx 0.32$ ensures that $p_{ij}(s_{ij})$ attains a minimum value of zero. The parameter $a_{ij} = a_{ji}$ is a random number between 0.1 and 1.0 – this stochasticity is necessary to avoid the existence of an optimal solution for all examples.

We consider three models for solving this problem: (i) the **single-pass** SE(3)-Transformer (12 layers); (ii) a three-block **iterative** SE(3)-Transformer (4×3 layers); (iii) **gradient descent** (GD) on the positions of the nodes. We also evaluate a combination of first applying an SE(3)-Transformer and then running GD on the output. Additionally, we evaluate the iterative SE(3)-Transformer both with and without propagation of basis function gradients as described in

Section 3.1. We run GD until the update changes the potential by less than a fixed tolerance value. We train the SE(3)-Transformers for 100 epochs with 5000 examples per epoch, which takes about 90 minutes. We ran each model between 15 and 25 times – the σ values in Tables 1 and 2 represent 1-σ confidence intervals (computed using Student's t-distribution) for the mean performance achieved by each model. All models except GD have the same overall number of parameters.

Table 1: Average energy of the system after optimisation (lower is better).

	Gradient Descent	**Single-Pass**	**No Basis Gradients**	**Iterative**	**Iterative + GD**
Energy	0.0619	0.0942	0.0704	0.0592	0.0410
σ	$\pm$0.0001	$\pm$0.0002	$\pm$0.0025	$\pm$0.0011	$\pm$0.0003

The results in Table 1 show that the iterative model performs significantly better than the single-pass model, approximately matching the performance of gradient descent. The performance of the iterative model is significantly degraded if gradients are not propagated through the basis functions. The best performing method was the SE(3)-Transformer followed by gradient descent. The fact that the combination of SE(3)-Transformer and gradient descent outperforms pure gradient descent demonstrates that the SE(3)-Transformer finds a genuinely different solution to the one found by gradient descent, moving the problem into a better gradient descent basin.

Table 2: Performance comparison of single-pass and iterative SE(3)-Transformers for different neighbourhood sizes K and a fully connected version (FC).

	FC Single (K9)	**FC Iterative (K9)**	**K5 Single**	**K5 Iterative**	**K3 Single**	**K3 Iterative**
Energy	0.0942	0.0592	0.1321	0.0759	0.1527	0.0922
σ	$\pm$0.0002	$\pm$0.0011	$\pm$0.0003	$\pm$0.0050	$\pm$0.0001	$\pm$0.0036

So far, every node attended to all $N-1$ other nodes in each layer, hence creating a fully connected graph. In Table 2, we analyse how limited neighborhood sizes [9] affect the performance, where a neighborhood size of $K=5$ means that every node attends to 5 other nodes. This reduces complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(NK)$, which is important in many practical applications with large graphs, such as proteins. We choose the neighbors of each node by selecting the nodes with which it interacts most strongly. In the iterative SE(3)-Transformer, we update the neighbourhoods in each step as the interactions change.

Table 2 shows that the iterative version consistently outperforms the single-pass version across multiple neighborhood sizes, with the absolute difference being the biggest for the smallest K. In particular, it is worth noting that the iterative version with $K=3$ outperforms the fully connected single-pass network.

In summary, the iterative version consistently outperforms the single-pass version in finding low energy configurations. We emphasise that this experiment is no more than a proof-of-concept. However, we expect that when moving to larger graphs (e.g. proteins often have more than 10^3 atoms), being able to explore different configurations iteratively will only become more important for building an effective optimiser.

Acknowledgements

We thank Oiwi Parker Jones and Rob Weston for helpful discussions and feedback. This research was funded by the EPSRC AIMS Centre for Doctoral Training at the University of Oxford and The Open Philanthropy Project Improving Protein Design.

References

1. Recep Adiyaman and Liam James McGuffin. Methods for the refinement of protein structure 3d models. *International journal of molecular sciences*, 20(9):2301, 2019.
2. John Jumper, R Evans, A Pritzel, T Green, M Figurnov, K Tunyasuvunakool, O Ronneberger, R Bates, A Zidek, A Bridgland, et al. High accuracy protein structure prediction using deep learning. *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, 22:24, 2020.
3. Brian Kuhlman and Philip Bradley. Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology*, 20(11):681–697, 2019.
4. Carlos Outeiral Rubiera. Casp14: what google deepmind's alphafold 2 really achieved, and what it means for protein folding, biology and bioinformatics. `https://www.blopig.com/blog/2020/12/casp14-what-google-deepminds-alphafold-2-really-achieved-and-what-it-means-for-protein-folding-biology-and-bioinformatics/`, 2020.
5. Lupoglaz. Openfold2. `https://github.com/lupoglaz/OpenFold2/tree/toy_se3`, 2021.
6. Phil Wang. Se3 transformer - pytorch. `https://github.com/lucidrains/se3-transformer-pytorch`, 2021.
7. Dale Markowitz. Alphafold 2 explained: A semi-deep dive. `https://towardsdatascience.com/alphafold-2-explained-a-semi-deep-dive-fa7618c1a7f6`, 2020.
8. Mohammed AlQuraishi. Alphafold2 @ casp14: "it feels like one's child has left home.". `https://moalquraishi.wordpress.com/2020/12/08/alphafold2-casp14-it-feels-like-ones-child-has-left-home/`, 2020.
9. Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing System (NeurIPS)*, 2020.
10. UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2019.
11. Protein data bank: the single global archive for 3d macromolecular structure data. *Nucleic acids research*, 47(D1):D520–D528, 2019.
12. Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.
13. Jinbo Xu. Distance-based protein folding powered by deep learning. *Proceedings of the National Academy of Sciences*, 116(34):16856–16865, 2019.
14. Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, 117(3):1496–1503, 2020.

15. Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J Haunsberger, and Johannes Söding. Hh-suite3 for fast remote homology detection and deep protein annotation. *BMC bioinformatics*, 20(1):1–15, 2019.
16. Joe G Greener, Shaun M Kandathil, and David T Jones. Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints. *Nature communications*, 10(1):1–13, 2019.
17. Nathaniel Thomas, Tess Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *ArXiv Preprint*, 2018.
18. Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
19. Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *ArXiv preprint*, 2018.
20. Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *Proceedings of the International Conference on Machine Learning, ICML*, 2018.
21. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.
22. Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
23. Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018.
24. Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.
25. Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the International Conference on Machine Learning, ICML*, 2016.
26. Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. *Proceedings of the International Conference on Machine Learning, ICML*, 2020.
27. Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *ArXiv Preprint*, 2020.
28. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
29. Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the International Conference on Machine Learning, ICML*, 2019.

30. Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing System (NeurIPS)*, 2019.
31. Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *International Conference on Learning Representations (ICLR)*, 2018.
32. Fabian B. Fuchs, Adam R. Kosiorek, Li Sun, Oiwi Parker Jones, and Ingmar Posner. End-to-end recurrent multi-object tracking and prediction with relational reasoning. *arXiv preprint*, 2020.
33. Jiancheng Yang, Qiang Zhang, and Bingbing Ni. Modeling point clouds with self-attention and gumbel subset sampling. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
34. Saining Xie, Sainan Liu, and Zeyu Chen Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
35. Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *International Conference on Learning Representations (ICLR)*, 2017.
36. Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
37. Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

A Network Architecture and Training Details

For the single-pass SE(3)-Transformer we use 12 layers. For the iterative version we use 3 iterations with 4 layers in each iteration. In both cases, for all hidden layers, we use type-0, type-1, and type-2 representations with 4 channels each. The attention uses a single head. The model was trained using an Adam optimizer with a cosine annealing learning rate decay starting at 10^{-3} and ending at 10^{-4}. Our gradient descent implementation uses a step size of 0.02 and stops optimizing when the norm of every position update is below 0.001.