# Learning Invariant Representations in Neural Networks


Fabian Bernd Fuchs

Supervised by Prof. Ingmar Posner

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Trinity 2021

# Abstract

Symmetries and invariances are ubiquitous in machine learning tasks. While convolutional neural networks famously and successfully exploit translational symmetries, other symmetries have, until recently, often been neglected. Incorporating symmetries or invariances into the neural network architecture avoids costly data augmentation and alleviates the need for large datasets. The presented work focuses on invariant and equivariant neural network layers, putting symmetries at the centre of neural network architecture design. Concretely, this thesis covers three different invariances: permutation invariance, roto-translation invariance, and label invariance.

- For permutation invariance, this work presents an analysis of the capacity and constraints of sum aggregation, one of the most used permutation invariant neural network paradigms. Next, different permutation invariant relational reasoning modules in the context of multi-object tracking are proposed and compared.
- For roto-translation invariance, a new roto-translation equivariant graph network based on self-attention is introduced. In a subsequent step, this is adapted to enable iterative position refinement, a crucial feature for protein structure prediction.
- Lastly, the case is covered where there is no known symmetry group but instead a nuisance factor that the practitioner would like the neural network to be invariant to. We refer to this concept as label invariance. To that end, a neural network module is introduced that can promote or suppress the nuisance factor while simultaneously making the network more interpretable.

# Acknowledgements

First and foremost, I want to thank my supervisor, Ingmar Posner. You have been a great teacher and mentor for the entire course of my Ph.D. I also particularly appreciate your role in making A2I (the Applied Artificial Intelligence Lab) such a fantastic research group. Pre-Corona, I truly enjoyed every single day at the office - I could not have hoped for a better community. Thank you to everyone at A2I and ORI (the Oxford Robotics Institute) for contributing to this.

I thank my thesis examiners, Pawan Kumar and Michael Bronstein, for reading and evaluating my thesis, and for a profound and interesting discussion of my work during the examination.

I am grateful to all of my co-authors; I learned so much from every single one of you. I particularly want to thank the following people, each of whom fundamentally changed the way I conduct research, in chronological order: Adam Kosiorek, Oliver Groth, Edward Wagstaff, Daniel Worrall, and Max Welling. All of you have had such a unique and important role in my PhD that each deserve a paragraph on their own.

Very special thanks also to my parents, Gabriele Fuchs and Klaus Abraham-Fuchs, to Nela Cernota, to Adam Golinski, to Sasha Salter, to Wendy Poole, and to everyone in the AIMS CDT cohort. Each of you had an irreplaceable role in keeping me cheerful and sane.

I also thank the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems and Kellogg College, Oxford for their generosity in funding most of the research presented here.

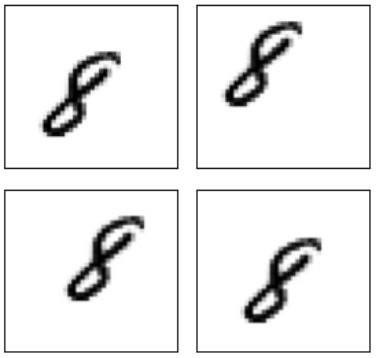
# Contents

# 1 Introduction

Symmetries are of paramount importance in the natural sciences and also arise in many machine learning tasks. Likely the most famous example is translation invariance in image classification: an image of an aircraft remains an image of an aircraft when moving the entire content several pixels to the right, up to boundary effects. The task of image *classification* is therefore said to be translation invariant or translation symmetric.

It is important to note that symmetry here is a property of the task, not the data sample. Concretely, the task of classifying airplanes is symmetric under translation of the input, but an image of an aircraft might not be symmetric in any meaningful way. The background section of this thesis (Chapter 2) will discuss the exact relationship between symmetry and invariance, but for now, the two terms can be understood as synonyms.

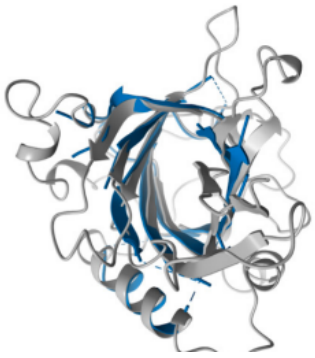
Convolutional neural networks exploit the translation symmetry mentioned above: the kernel convolved with a patch of the input image yields the same output regardless of the position of this patch. When considering tasks on other data, e.g., the 3D coordinates of a molecule, additional invariances may emerge (see Figure 1.1). For example, the energy or entropy of a molecular configuration does not change when



**(a)** A translated MNIST digit visualising the translation symmetry of image classification.



**(b)** 3D point cloud of the Stanford Bunny (Keselman and Hebert [36]).



**(c)** 3D protein structure predicted using SE(3)-Transformers (Baek et al. [1]).

**Figure 1.1:** Task symmetries in machine learning on different data domains. (a) Translational symmetries in image processing tasks have long been leveraged through CNNs. (b, c) Leveraging symmetries and invariances in other tasks, such as point cloud classification (b) or protein structure prediction (c) is an active area of research and the focus of this thesis. (c) shows a concrete application of this research: Baek et al. [1] use the Se(3)-Transformer, introduced in Sections 5 and 6, to predict the structure of a protein, exploiting roto-translational symmetries of the task.

rotating the system. Moreover, there is no intrinsic ordering of the atoms in a molecular graph, leading to permutation invariance. In machine learning, many such invariances arise. This raises two immediate questions: (1) Is there a benefit in incorporating those invariances in the design of our machine learning models? (2) If yes, what is the best way of doing so?

To answer the first question, a number of different perspectives can be taken. An empirical perspective is observing the pivotal role of convolutional neural networks (CNNs) in the advent of deep learning. CNNs are a special case of fully connected networks (FCNs). Moreover, convolutional layers are the most general case of a linear layer obeying translational symmetry (Cohen et al. [13]). Hence, the only difference between FCNs and CNNs is arguably the incorporation of a symmetry, making them a suitable case study for the effects of leveraging symmetries. In contrast to fully connected networks, CNNs have significantly fewer parameters yielding to less overfitting, smaller memory footprint, and faster training and inference times. Another perspective is robustness: deep learning is often referred to as a black box and seldom comes with guarantees. However, building invariances into models,

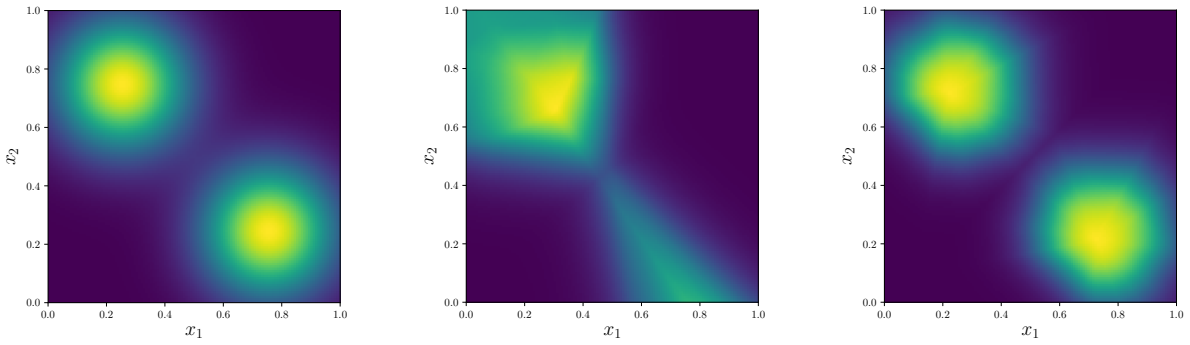
depending on the specific technique, does come with guarantees. As will be shown later, it is entirely possible to design a point cloud classification network that is *guaranteed* to not change its prediction under rotation of the point cloud. Robustness guarantees are not only intellectually pleasing but can also be vitally important in application domains like autonomous vehicles or medicine. A third perspective is an intuitive one about learning: when a piece of information, like the orientation of a molecule in space, is not relevant for predicting the quantity of interest, building this into the network should help the learning by forcing the network to focus on the information that is relevant for the task.

Assuming that the above yields sufficient motivation to leverage invariances and symmetries, the next question is how to best achieve this in deep learning. This is an active and growing field of research, and the work described in this thesis aims at contributing to answering this question.

## 1.1 Contributions

Irrespective of the specific type of symmetry under consideration, several overarching concepts are essential for an appropriate discussion. These include a mathematical definition of task symmetry and the connection to invariance, as well as the related concept of equivariance. These topics will be covered in Chapter 2. Equipped with these, this thesis makes contributions to the research area of designing invariant neural network layers. The contributions are clustered into three distinct topics, each covering a specific type of invariance. The sections below each cover one of the three invariances by first motivating why they are important in machine learning and then outlining the contributions made.

Each section features an exemplary visualisation of the respective symmetries and how neural networks can profit from leveraging these symmetries (Figures 1.2 to 1.4). These are intended to make the symmetry more tangible by visualising it in low dimensions and showcasing the performance benefit of incorporating the symmetry into the neural network design. To this end, the following setup is introduced: a target function $\phi(x_1, x_2)$ maps from $\mathbb{R}^2$ to $\mathbb{R}$ (left panel of Figures 1.2 to 1.4, respectively). In all plots, the two axes of the coordinate system are the input dimensions, whereas the (scalar) output is visualised via colour. First, a fully connected neural network is trained to approximate $\phi$. This learned approximation is referred to as $\hat{\phi}$ (centre panel). Theoretically, the network should be able to learn the symmetries of the task. However, due to the deliberately small number of training samples, the fully connected network learns only an imperfect approximation $\hat{\phi}$, which often does not exhibit the same symmetries as $\phi$. We then incorporate the knowledge about the symmetry into the network architecture (right panel) and refer to the learned symmetric function as $\hat{\phi}_s$. Leaving the number of training samples, training epochs, and trainable parameters constant, we show that $\hat{\phi}_s$ is often the more accurate approximation to $\phi$.




**(a)** Symmetric target function: $\phi(x_1, x_2) = \phi(x_2, x_1)$.

**(b)** Predictions $\hat{\phi}(x_1, x_2)$ of a fully connected network.

**(c)** Predictions $\hat{\phi}_s(x_1, x_2)$ of a permutation inv. network.

**Figure 1.2:** A visualisation of *permutation* invariance in 2D. The colour reflects the output value of the ground truth (a) and learned (b,c) functions. (b,c) Two neural networks are trained on a small set of random data points. (b) The fully connected network does not leverage the symmetry of the task and therefore does not model both modes equally well. (c) Incorporating the symmetry into the network architecture can greatly improve accuracy.

### 1.1.1 Permutation Invariance

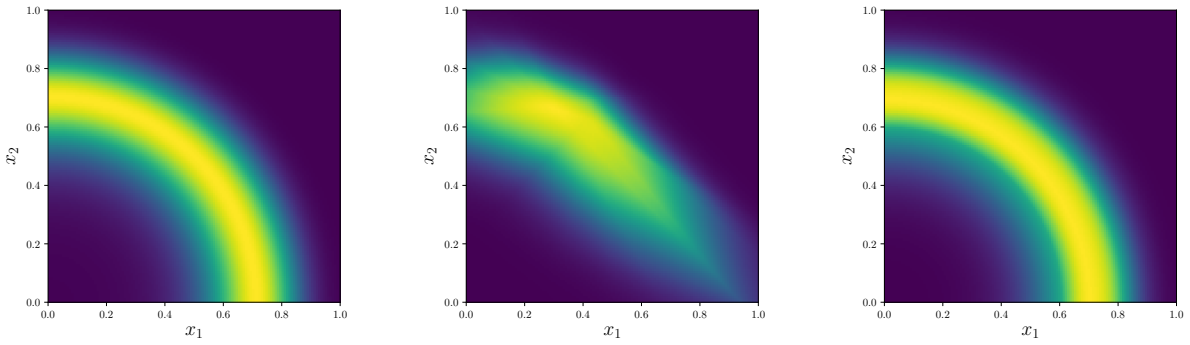
Examples of permutation invariant tasks in deep learning are plentiful: point cloud classification, relational reasoning about multiple objects in an image or regression tasks on a group of atoms that together form a molecule. In all of these tasks, the practitioner has to feed a collection of inputs - e.g., the points of a point cloud - into a neural network. Critically, these collections have no intrinsic ordering. In other words, these tasks are symmetric with respect to all permutations, which together form a symmetry group. Feeding a list of points to a fully connected layer would not respect this symmetry. The network would have to *learn* to become permutation invariant at the cost of additional compute, training data, and/or data augmentation. This raises the question of how to construct neural networks which are intrinsically permutation invariant instead.

Figure 1.2 provides a visualisation of permutation invariance in low dimensions. A permutation invariant function with two inputs exhibits an axis symmetry with respect to the $x_1 = x_2$ axis (see left panel). (b) and (c) use the same fully connected network and training procedure. In (c), the network is made invariant by ordering the input such that $x'_2 > x'_1$. Due to the low dimensionality of the problem, this

simple strategy suffices to achieve good performance and to successfully exploit the symmetry.

In practice, the method of choice for building permutation invariant operations into neural networks is often sum aggregation, typically computing the sum over a set of latent feature vectors. As the sum as a mathematical operator is commutative, the output of a sum aggregation layer is trivially permutation invariant. The downside of this simplicity is the self-evident loss of information, which goes beyond the ordering of the inputs: $5+4$ is not only the same as $4+5$ but also the same as $6+3$. This raises the question of whether there are permutation invariant functions that cannot be approximated with a network using sum aggregation. In other words, are neural networks using sum aggregation universal function approximators? Chapter 3 provides a theoretical analysis of this question. Concretely, the Deep Sets framework (Zaheer et al. [72]) is taken as a concrete instantiation of set-based learning using sum aggregations. The core finding is that Deep Sets is indeed a universal function approximator if, and only if, the number of latent dimensions is at least as large as the number of inputs (i.e., the cardinality of the input set). The takeaway is two-fold: (1) Sum aggregation is a valid choice for achieving permutation invariance in a general setting. (2) Depending on the function to be learned, sum aggregation might introduce a scaling issue with respect to the cardinality of the input set.

Chapter 4 examines an application of permutation invariant network architectures: relational reasoning in the context of multi-object tracking. If there is no intrinsic ordering between objects, relational reasoning between objects is a permutation invariant problem. Following the hypothesis from further above, a relational reasoning module should therefore profit from leveraging this invariance. The experiments in Chapter 4 provide empirical evidence that this is indeed the case. Furthermore, different permutation invariant architectures are compared, showing that self-attention is a particularly suitable module for relational reasoning.




**(a)** Rotation symmetric target funciton $\phi(x_1, x_2)$.

**(b)** Predictions of a rotation sensitive network $\hat{\phi}(x_1, x_2)$.

**(c)** Predictions of a rotation invariant network $\hat{\phi}_s(x_1, x_2)$.

**Figure 1.3:** A visualisation of rotation invariance in 2D. The colour reflects the output value of the ground truth (a) and learned (b,c) functions. Two neural networks are trained on a small set of random data points. (c) Analogous to Figure 1.2, incorporating the symmetry into the network architecture can greatly improve accuracy.

### 1.1.2 Roto-Translation Invariance

The previous section discussed permutation invariance, which, for example, arises in tasks on point clouds or molecules. However, these tasks often exhibit additional symmetries: rotational and translational symmetries. Interestingly, the most popular deep learning algorithms for point clouds do not fully leverage these additional symmetries (e.g. Qi et al. [52, 53], Wang et al. [64], and Zaheer et al. [72]) and hence fail to incorporate an inductive bias about geometry into the network architecture which goes beyond the concept of spatial proximity. More concretely, these networks handle x-y-z coordinate no differently than any other 3-tuple of scalars. However, in the same way that CNNs have an inductive bias about image pixels (discrete 2D coordinates) by being translation invariant, networks for point clouds could obtain an inductive bias about 3D coordinates.

A schematic visualisation of the effects of leveraging rotation symmetry is provided in Figure 1.3. Here, invariance is achieved by feeding the norm $\sqrt{x_1^2 + x_2^2}$ instead of the vector $(x_1, x_2)$ into the network. In practice, the input typically consists of more than just one point in space, leading to a variety of mathematically complex approaches for leveraging roto-translational symmetries.

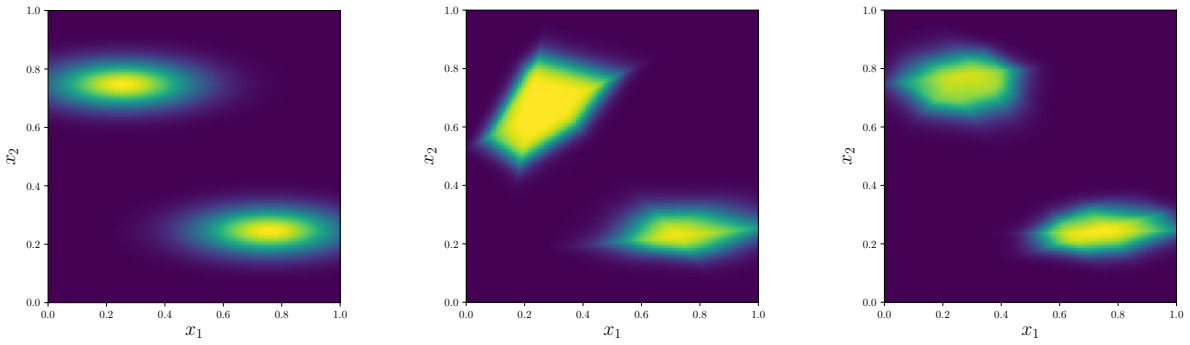
Chapter 5 proposes a roto-translational *equivariant* architecture, the SE(3)-Transformer. Equivariance is a more general form of invariance, which allows for passing on information (e.g., about orientation) to the next layer without increasing the number of parameters or allowing to overfit to this information. The SE(3)-Transformer is a graph neural network based on so-called *irreducible* representations (Thomas et al. [59] and Weiler et al. [66]) and self-attention (Lee et al. [40] and Vaswani et al. [61]). The SE(3)-Transformer has, for example, been used for predicting structures of proteins (Baek et al. [1]), a central challenge in biochemistry.

Specifically, when predicting structures of proteins, networks often iteratively refine the position predictions for the atoms (Baek et al. [1] and Jumper et al. [34]). This iterative application, which the original SE(3)-Transformer paper did not discuss, imposes an interesting additional challenge connected to gradient flow through non-trivial mathematical operations. Chapter 6 discusses how to address this and provides an implementation.

### 1.1.3 Label Invariance

Chapter 7 introduces neural stethoscopes, which enable suppression of features if they are connected to a label which comes with the dataset (or can be easily obtianed, e.g., by using a pre-trained classifier).

In the previous two sections, the invariances were connected to known symmetry operators leading to mathematical constraints on the construction of invariant and equivariant layers. However, some machine learning tasks have invariances that are not connected to a (known) symmetry operator. For example, an autonomous vehicle might be required to operate during the day as well as during nighttime. Despite substantial changes in the sensor input, the vehicle might even be required to behave the same during the night as during the day. In other words, the behaviour should be, to some degree, invariant with respect to the amount of light present in the scene. Moreover, the training data might not be equally distributed across domains. Hence, it would be beneficial to leverage training data from a domain with abundant data to




**(a)** Target function $\phi(x_1, x_2)$: two similar, disjoint domains.

**(b)** Predictions $\hat{\phi}$ of a fully connected network.

**(c)** Predictions $\hat{\phi}_s$ of a label invariant network.

**Figure 1.4:** A visualisation of label invariance. Here, two disjoint domains have similar properties. When labels for the domains are provided, an adversarial setup can be employed to approximate this invariance in a learned fashion, such as in this example. Intuitively, by forcing the network to process examples from both domains identically, it can use a sample from one domain to learn about all domains simultaneously.

simultaneously also learn about domains with less data. Both arguments motivate the implementation of invariant neural network layers in such cases. However, it is likely impossible to find and solve mathematical constraints for a neural network layer to respect the invariance with respect to lighting in the example above.

Chapter 7 focuses on such cases and discusses how adversarial training can be used to impose a learned invariance on a latent representation of a network. The proposed module is called a *neural stethoscope* as it probes the information content of the latent representation. As this technique leverages additional labels of an auxiliary task, we refer to this invariance as *label invariance*. Figure 1.4 visualises the potential benefit of label invariance in a toy example: two domains (the top left and the bottom right corner of the image) share similar properties. If labels are available for the auxiliary task of predicting the domain of the data sample, these labels can be used in an adversarial setup to enforce invariance with respect to the domain label. This in turn forces the network to treat both domains equally and therefore, ideally, to leverage each training sample twice—once for each domain.

### 1.1.4 Summary of Contributions

The contributions to the field of learning invariant representation with neural networks can be summarised as follows:

- Analysing the capacity and constraints of sum aggregation, one of the most used permutation invariant network paradigms (Chapter 3).
- Proposing and comparing different permutation invariant relational reasoning modules for multi-object tracking (Chapter 4).
- Designing a roto-translation equivariant graph network based on self-attention (Chapter 5).
- Expanding the roto-translation equivariant graph network to be suitable for iterative position refinement, which is crucial for protein structure prediction (Chapter 6).
- Introducing a neural network module that can promote or suppress a nuisance factor (label invariance) while simultaneously making the network more interpretable (Chapter 7).

## 1.2 Publications

The following is the list of papers which are included in this thesis. The stars indicate equal contribution of authors.

1. E. Wagstaff*, F. B. Fuchs*, M. Engelcke*, I. Posner, and M. A. Osborne. "On the Limitations of Representing Functions on Sets". In: *International Conference on Machine Learning*. 2019.
2. F. B. Fuchs, A. R. Kosiorek, L. Sun, O. P. Jones, and I. Posner. "Relational Reasoning and Permutation Invariance in Multi-Object Tracking". In: *Workshop on Sets and Partitions, NeurIPS*. 2019.
3. F. B. Fuchs*, D. E. Worrall*, V. Fischer, and M. Welling. "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2020.
4. F. B. Fuchs*, E. Wagstaff*, J. Dauparas, I. Posner. "Iterative SE(3)-Transformers". In: *International Conference on Geometric Science of Information, Paris*. 2021.
5. F. B. Fuchs, O. Groth, A. R. Kosiorek, A. Bewley, M. Wulfmeier, A. Vedaldi, and I. Posner. "Scrutinizing and De-Biasing Intuitive Physics with Neural Stethoscopes." In: *British Machine Vision Conference*. 2019.

The following work was also conducted during the course of the DPhil but is not included in this thesis:

1. O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. "ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking". In: *The European Conference on Computer Vision (ECCV)*. 2018.
2. V. G. Satorras, H. Emiel, F. B. Fuchs, I. Posner, and M. Welling. "E(n) Equivariant Normalizing Flows for Molecule Generation in 3D". in: *Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

3. E. Wagstaff, F. B. Fuchs, M. Engelcke, M. A. Osborne, and I. Posner. “Universal Approximation of Functions on Sets”. In: *arXiv:2107.01959*. 2021.

## 1.3 Outline

Chapter 2 will introduce the background necessary for an in-depth discussion of the contributions of this thesis. Specifically, it includes both mathematical and intuitive descriptions of invariance, equivariance, task symmetries, and the connections thereof. It also discusses the related literature, which puts the contributions of this thesis into context. Chapters 3-7 each contain one article on the topic of learning invariant representations. Chapter 3 and 4 both discuss *permutation invariance*, an invariance that is particularly relevant in set- and graph-based machine learning. Chapter 5 and 6 expand on this by adding another invariance that can be important in graph-based learning, especially when considering spatial information: *roto-translation invariance*. Chapter 7 rounds off the thesis by examining how a neural network can be made invariant even when there is no concrete symmetry group. Using an adversarial training method, a neural network layer is made *label invariant*. The final chapter of this thesis, Chapter 8, discusses limitations of the proposed approaches, outlines future work and describes applications of invariant and equivariant neural networks.

# 2
# Background

## 2.1 General Concepts

This section introduces general concepts around symmetries, invariance, equivariance, and group theory, which are relevant for the rest of this thesis. Inspiration for this section was drawn from Bronstein et al. [6], Fuchs et al. [26], Mallat [45], and Weiler et al. [66] and notation is borrowed from Fuchs et al. [26]. However, many of the topics of this section are long-established standard concepts in mathematics and are described in many textbooks.

### 2.1.1 Task Symmetries

The following presents the concept of task symmetries as described in Mallat [45]. Here, a global symmetry is defined as an invertible operator $g : \mathcal{X} \to \mathcal{X}$ which, applied to the input of a function $\phi : \mathcal{X} \to \mathcal{Y}$, does not change its result:

$$\phi(g \circ x) = \phi(x) \quad \forall x \in \mathcal{X} \tag{2.1}$$

In other words, $g$ is a symmetry of $\phi$. Concrete examples of such symmetries are permutations of a set or rotations of a point cloud. It is worth pointing out that the above defines a symmetry as a property of a task, not a dataset. For instance, let the

dataset be a collection of indoor objects represented as point clouds. Each point is simply an x-y-z coordinate. If the task is to predict whether the object is a table or a chair, the task is symmetric or *invariant* with respect to global rotations of the points. However, if the task was to predict whether the object is in an upright position or upside-down, rotating the point cloud could certainly change the desired output. If symmetries are present in a machine learning task, leveraging these can be beneficial in terms of accuracy, computational efficiency and/or robustness (Chapters 3-6).

### 2.1.2 Group Theory

If Equation (2.1) holds for invertible operators $g_1$ and $g_2$, then the composition $g_1 \circ g_2$ is also an invertible symmetry operator. Mathematically, this implies that the operators form a *group* under composition (Bronstein et al. [6] and Mallat [45]). Exactly for that reason, group theory plays a central role in leveraging symmetries in machine learning. This section covers basic group theory, providing sufficient background for most of the content of Chapters 3-6. Chapter 5, which goes deeper into representation theory, has its own background section in its appendix about the relevant representation theory specifically for rotation symmetries in three dimensions.

A group $(G, \circ)$, which consists of a set $G$ and a composition operator $\circ$, is an abstract mathematical construct with the following properties:

- **Closure**: $g \circ h \in G \quad \forall g, h \in G$
  *The group is closed under composition.*
- **Associativity**: $f \circ (g \circ h) = (f \circ g) \circ h \, \forall g, h \in G$
- **Identity**: $\exists e \in G \mid e \circ g = g \circ e = g \quad \forall g \in G$
  *An identity element exists.*
- **Inverse**: $\exists g^{-1} \in G \mid g^{-1} \circ g = g \circ g^{-1} = e \quad \forall g \in G$
  *Inverse elements exist.*

In order to make this practically relevant, it is important to define how the respective (symmetry) group acts on data points and/or feature vectors $x \in \mathcal{X}$. To that end,

two additional concepts are important: group actions and group representations. Actions, also referred to as transformations, are functions $T_g$ that map from $\mathcal{X}$ to $\mathcal{X}$ and are parameterised by a group element. These actions of course need to observe the group structure, so a composition of two group elements must be equivalent to the composition of the respective group actions: $T_{g \circ h}[x] = T_g[T_h[x]]$ (Bronstein et al. [6]).

We are interested in linear actions on finite dimensional vector-spaces, which brings us to *representations*. An N-dimensional real representation $\rho(g)$ of a group maps an element $g \in G$ to an $N \times N$ dimensional invertible matrix, often denoted as $GL(N)$ (Bronstein et al. [6]). Representations, too, have to obey the group structure: $\rho(g) \cdot \rho(h) = \rho(g \circ h)$ where '$\cdot$' indicates matrix multiplication.

To summarise, representations map from a group element to an invertible matrix, whereas an action maps a group element together with a vector to a new vector. A standard way to define the group action is to define a representation and have the action be the multiplication of the resulting matrix and the vector. For example, when considering the group of rotations in 3D, the group actions are often defined via multiplying a $3 \times 3$ rotation matrix $\rho(g) \in SO(3) \subset GL(3)$ with a vector. However, group actions can also be defined without considering representations. Permutations, for instance, are often defined and used without explicitly using permutation matrices.

It is important to note that this meaning of representation is a different one than the one referred to in the title of this thesis. The title refers to the machine learning concept of 'representation learning', which describes the extraction of useful features from input data and is not generally linked to group theory.

### 2.1.3 Invariance and Equivariance

When exploiting task symmetries in machine learning, two concepts are particularly important: invariance and equivariance. In machine learning, a function or neural network layer is invariant if the output of a layer, also referred to as a feature, does not

change under certain transformations of the input (note the similarity to the definition of a task symmetry in Equation (2.1)). In contrast, a function $\phi : \mathcal{X} \to \mathcal{Y}$ is called equivariant if for every $g$ there exists a transformation $S_g : \mathcal{Y} \to \mathcal{Y}$ such that

$$S_g[\phi(x)] = \phi(T_g[x]) \qquad \text{for all } g \in G, x \in \mathcal{X}. \tag{2.2}$$

where $T_g : \mathcal{X} \to \mathcal{X}$ for $g \in G$ is a set of transformations and $G$ is a symmetry group. To obtain an intuition for the implications of this, it is useful to consider a convolutional neural network (CNN). A single convolutional layer of a CNN is translation equivariant, not invariant. If the input is shifted by 2 pixels to the right, the output is also shifted by two pixels to the right – assuming the stride is set to 1. This ensures that a translated version of the input does not impose an entirely new problem to the network. In fact, the network outputs something very similar, just shifted by a transformation which corresponds exactly to the transformation of the input.

Stacking equivariant layers preserves equivariance. Crucially, equivariant features can be transformed into invariant features as well. A network comprised of multiple convolutional layers followed by global pooling and fully connected layers is translation invariant - up to boundary effects. For a general framework for how to construct complex neural networks from linear equivariant layers, local and global pooling layers, and equivariant non-linearities, see Section 3.5 *'The Blueprint of Geometric Deep Learning'* in Bronstein et al. [6].

It is important to note that the transformations $S_g$ and $T_g$ in Equation (2.2) can be the same but do not have to be. Chapter 5 features examples of different representations of the same symmetry group.

### 2.1.4 Aside: Connection to Noether's Theorem

Despite not being directly related to the work covered in this thesis, it is worthwhile to examine the connection to Noether's theorem about symmetries and invariances due to its fundamental role in physics. Noether's theorem builds on Lagrangian mechanics, which frames a classical mechanics problem in terms of a set of generalised

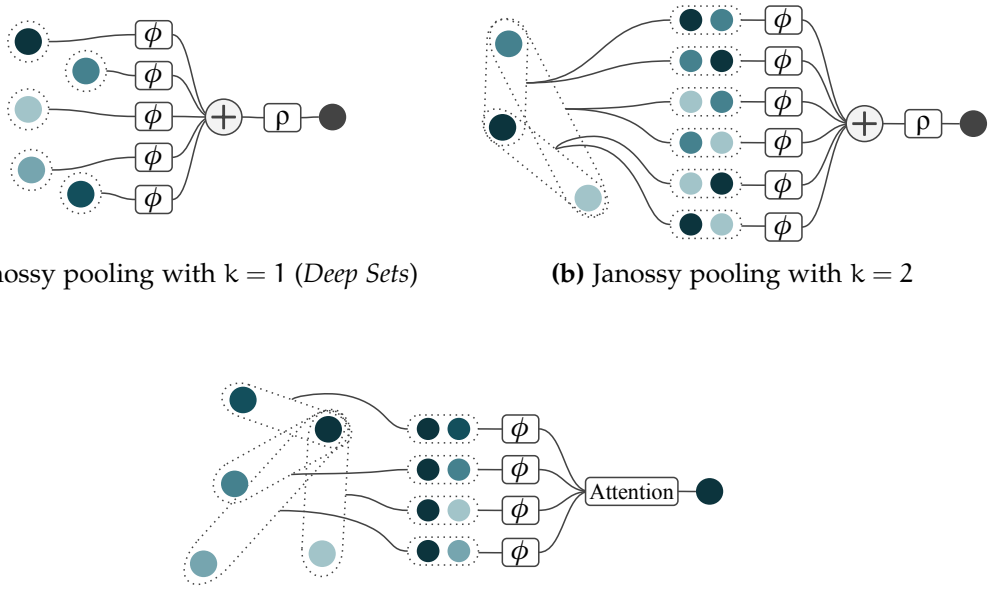
coordinates which entirely characterise the system and the possible motions of the particles. The Lagrangian, a scalar quantity defined as the difference between the kinetic and the potential energy of the system, can be used to derive the equations of motion. Noether [50] found that any differentiable symmetry of the Lagrangian is connected to a conserved quantity, i.e. an invariance of the system. For instance, a translational symmetry of the Lagrangian leads to conservation of momentum; a symmetry of the Lagrangian with respect to shifts in time leads to conservation of energy. Hamiltonian neural networks (Greydanus et al. [29]) and Lagrangian neural networks (Cranmer et al. [16]) make use of such formulations of classical mechanics to solve physical particle systems, building energy conservation directly into the neural network. Interestingly, invariance of energy over time is not always practical, particularly in the presence of friction or external forces, a limitation addressed by Desai et al. [19]. Most equivariant and invariant neural network architectures, such as the ones discussed in this thesis, are not directly connected to Noether's theorem or classical mechanics in general. However, many invariant and equivariant neural networks, such as the approaches discussed in Chapters 3-6, Noether's theorem, and, in fact, many of the physical theories of the 20th century (Chapter 13 in Penrose [51]) do share the same sentiment of putting symmetries at centre stage.

## 2.2 Related Work

The following provides an overview of the related work on the individual types of invariance covered in this thesis. For each invariance, the aim is to divide the rich and diverse set of existing approaches into a small number of categories and motivate the respective underlying paradigms.

### 2.2.1 Permutation Invariance

This section provides an overview of permutation invariant neural network architectures. Permutation invariance is needed for approximating functions that map from a set to a global property. To keep a clear scope, permutation equivariant methods (mapping from sets to sets) will not be covered in this section, despite being




**(a)** Janossy pooling with k = 1 (*Deep Sets*)

**(b)** Janossy pooling with k = 2

**(c)** Self-Attention

**Figure 2.1:** Janossy Pooling considering sub-sets with one and two elements, respectively. (a, b) By processing all combinations of sets with k distinct elements and aggregating the results with a sum, the output is permutation invariant. (c) Self-attention can be seen as a variant of Janossy pooling with k = 2 where one element of the tuple is kept fixed (the dark green node in this case). In practice, self-attention is often computed for all nodes in parallel, mapping sets of points to sets of points and guaranteeing equivariance (Lee et al. [40]).

closely related: see Zaheer et al. [72] as an example for a permutation invariant model that can be turned into an equivariant one with straight-forward modifications. The first part of this section will cover different methods of achieving permutation invariance, which is relevant background for Chapters 3 and 4. Most of the literature on exactly permutation invariant approaches can be classified into two paradigms: (1) permuting & averaging and (2) sorting. After explaining these two paradigms and providing examples, a connection is drawn from sets to graphs, an important concept for Chapters 4-6 of this thesis.[1]

**Invariance through Permuting and Averaging** In this first paradigm for constructing invariant networks, the core idea is to consider all permutations of the inputs, process them with a neural network and then average the results. One framework

[1]Much of the content in Section 2.2.1 is strongly based on an analysis of the literature I conducted for Wagstaff et al. [63] with help of my co-authors. Particularly the figures are taken from this work. I warmly thank my co-authors for their contributions.

describing this idea is Janossy pooling (Murphy et al. [49]), which can be expressed mathematically as follows:

$$\widehat{f}(\mathbf{x}) = \frac{1}{|S_M|} \sum_{\pi \in S_M} \phi\big(\pi(\mathbf{x})\big) \tag{2.3}$$

where $\mathbf{x}$ has $M$ elements, $S_M$ is the group of all permutations $\pi$ of $M$ elements, and $\phi$ is a permutation sensitive neural network. This results in a permutation-invariant network $\widehat{f}$, regardless of the choice of $\phi$.

Typically, the output of such a Janossy pooling layer is fed into another neural network $\rho$:

$$f(\mathbf{x}) = \rho\big(\widehat{f}(\mathbf{x})\big) \tag{2.4}$$

As the input $\widehat{f}(\mathbf{x})$ is already permutation invariant, $\rho$ does not need to follow any constraints. An advantage of this scheme is that it is universally applicable and, at least in theory, very expressive (Wagstaff et al. [63]). The main drawback is that the computational complexity scales at least linearly with the cardinality of $S_M$, which is $M!$. The computational burden can be alleviated by not processing the entire set at once but instead only all $k$-tuples. For example, with $M = 4$, $k = 2$, and an input set $\{w, x, y, z\}$, the sum is then over all 2-tuples from the set, namely:

$$(w,x),\ (x,w),(w,y),\ (y,w),\ (w,z),\ (z,w),$$
$$(x,y),\ (y,x),\ (x,z),\ (z,x),\ (y,z),\ (z,y)$$

with $k < M$. The tuples here consist of distinct elements of the set, so pairs such as $(w,w)$ were ignored.

Neural network architectures with $k = 1$ and $k = 2$ are depicted in Figure 2.1a and Figure 2.1b, respectively. Mathematically, letting $\mathbf{x}_{\{k\}}$ denote a $k$-tuple from $\mathbf{x}$, this reads

$$\widehat{f}(\mathbf{x}) = \frac{1}{P(M,k)} \sum_{\mathbf{x}_{\{k\}}} \phi(\mathbf{x}_{\{k\}}) \text{ , where } P(M,k) = \frac{M!}{(M-k)!}. \tag{2.5}$$

If $k$ is sufficiently small, the sum now has far fewer terms than $M!$, yielding a tractable computational cost in most cases. Concretely, the sum now has $\mathcal{O}(M^k)$ terms, where $k$ is fixed.

Interestingly, many of the existing neural network architectures for sets can be seen as variations Janossy pooling. Deep Sets (Zaheer et al. [72]), PointNet (Qi et al. [52]), and PointNet++ (Qi et al. [53]) resemble Janossy pooling with $k = 1$. Analogously, applying self-attention (Vaswani et al. [61]) to sets (Lee et al. [40]) or graphs (Veličković et al. [62]) has strong parallels with binary Janossy pooling ($k = 2$). Some of the differences are the choice of aggregation function (sum-pooling, max-pooling, softmax, ...) and whether the mapping is invariant or equivariant. However, the core principle of considering all $k$-tuples, processing them individually, and aggregating the results remains.

While $k = 1$ has lower computational cost than $k = 2$ (linear vs. quadratic), the main advantage of $k = 2$ is the explicit modelling of binary interactions. Wagstaff et al. [63] use a cooking recipe example to create an intuition. The hypothetical task is to predict whether a set of ingredients will likely result in a good meal. Hypothetically setting the second neural network in Figure 2.1 to the identity, all that can be done with a Deep Sets architecture is to score the tastiness of each ingredient and add or average the scores. While likely achieving some predictive prowess, this approach cannot penalise clashes between ingredients, such as between garlic and vanilla. Binary Janossy pooling, on the other hand, can at least directly test all pairwise compatibilities. Chapter 4 shows empirical evidence that self-attention can indeed help with relational reasoning.

**Invariance through Sorting** This part covers the second category of exactly permutation invariant neural networks. When representing a set as an ordered structure, the order is assumed to be arbitrary. That is, there may be two representations $\mathbf{x}$ and $\mathbf{y}$ of the same set, where the elements are ordered differently, and both of these are regarded as valid representations of the set. The purpose of permutation invariant neural networks is to not overfit to a specific representation and instead produce

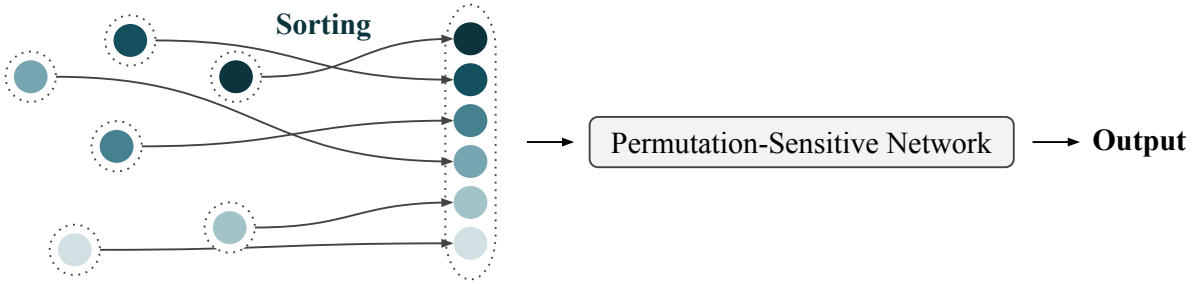



**Figure 2.2:** Permutation invariance through sorting. If the sorting does not depend on the order of the input, the model as a whole is permutation invariant regardless of the permutation sensitivity of the neural network itself.

the same result for all representations. Alternatively, one could define a canonical ordering which maps each set to a single valid representation. Put differently: if the neural network includes a canonical sorting step in the very beginning, then the entire model is permutation invariant as any permutation of the original set (before the sorting) will yield the same neural network output (see Figure 2.2).

An upside of this method is computational efficiency as only one copy of the set has to be processed by the network. The additional cost of sorting the elements typically scales with $O(n \log n)$. The challenge of this procedure is how to choose the canonical ordering as this may affect the performance of the model. Instead of manually choosing one, the ordering can also be learned, for example by training a scoring neural network that maps each element to a scalar. This procedure comes with its own challenges. A function mapping from a sorting score to a rank is piece-wise constant. E.g., for alphabetical sorting, this function would map $(\text{car}, \text{tent}, \text{ball}) \mapsto (2, 3, 1)$. If labels were available for the rank during training time, then the scoring network could be trained via backpropagation in a similar fashion as any classification or regression network. However, those labels are typically not available and a good ordering is instead determined by whatever yields a good performance of the permutation sensitive network. The straight-through estimator (Bengio et al. [2] and Yin et al. [71]) produces artificial gradients for the piece-wise constant functions and therefore allows for such an end-to-end training of the scoring function. Alternatively, the differential sorting operator by Blondel et al. [4], which scales with $O(n \log n)$, could also be employed.


**Sets:** Information is only assigned to individual nodes

**Graphs:** Information is assigned to edges, i.e. pairs of nodes

**Hyper-Graphs:** Information is assigned to hyper-edges, e.g. triplets of nodes

**Figure 2.3:** Sets, graphs, and hyper-graphs. (a) Sets can be seen as graphs without edges where information is only attached to nodes. (b) Graphs can have information attached to nodes as well as pairs of nodes (edges). (c) This idea can further be extended to hyper-graphs, where features are attached to triplets of nodes (Wei et al. [65] and Yadati et al. [70]).

Interestingly, the gradients for the scoring network do not need to come from the sorting itself: Liu et al. [41] propose to sort the values according to a learned score function (as described above), followed by a 1D convolution over the ordered values. To get gradients for the score function, the inputs to the 1D convolution are multiplied with the score function itself. This follows a similar principle as one of the architectures proposed in Y. Zhang et al. [73], where features are sorted according to the features value themselves. However, Y. Zhang et al. [73] also propose a variant where they obtain explicit sorting gradients through continuous relaxation (Grover et al. [31]), which is closer in spirit to the straight-through estimator approach described above.

Overall, the different variants of Janossy pooling, particularly sum-aggregation and self-attention, have been more prevalent in neural network architectures than the sorting paradigm. However, the specifics of the task will ultimately determine how effectively the challenges above can be addressed and which paradigm is best suited.

**Extending Sets to Graphs** There are tasks in machine learning which are permutation invariant, but the input data is not a set. Graphs are a famous example. Fortunately, much of the machinery discussed above can be used for tasks on graphs as well. Figure 2.3 visualises the relationship of sets, graphs, and hyper-graphs. A permutation of the input nodes is now coupled with a respective permutation of the

edges, for example by applying the same permutation to both rows and columns of the adjacency matrix. A wide range of graph neural networks leverage this specific form of permutation symmetry, often using multiple permutation equivariant layers followed by a permutation invariant pooling layer (Bruna et al. [8], Defferrard et al. [17], Kipf and Welling [37], and Satorras et al. [57]). Amongst the many studies of graph neural networks in the literature, Maron et al. [46] stand out by analysing the vectorised adjacency tensors and searching for the full set of independent linear, permutation-invariant or equivariant functions (i.e. matrices) transforming these tensors. Indeed, they find an orthogonal basis of such transformations. Remarkably, the number of basis vectors does not depend on the number of nodes in the (hyper-) graph and is instead connected to the Bell number. For instance, a bi-directional graph with $M$ nodes has an adjacency matrix of size $M \times M$. Vectorising this matrix yields a vector of length $M^2$. As Maron et al. [46] and Finzi et al. [22] show, for any $M > 3$, there is an orthogonal basis of exactly 15 $M^2 \times M^2$ matrices which linearly transform the adjacency matrices while preserving equivariance, where 15 is the fourth Bell number. This is helpful especially for machine learning on large graphs as it gives an upper bound to the rank of the linear, equivariant mapping and thus the number of learnable parameters.

### 2.2.2 Roto-Translation Invariance and Equivariance

In computer vision, the crucial importance of spatial invariances in signal processing has long been recognised. Zisserman et al. [75] summarised this as follows: "Invariance overcomes one of the fundamental difficulties in recognising objects from images: that the appearance of an object depends on viewpoint." In the domain of machine learning with neural networks, on the other hand, particularly for tasks on 3D point clouds, spatial symmetries have been ignored almost entirely by a majority of the most popular algorithms (e.g. Qi et al. [52, 53], Rao et al. [55], Wang et al. [64], and Zaheer et al. [72]). Especially more recently, some approaches have been using rotation invariant local descriptors to obtain robustness against rotations (Chen et al. [12], Deng et al. [18], and Z. Zhang et al. [74]). In contrast, Chapter 5 and 6 as

well as this section focus on leveraging roto-translational[2] symmetries through the coupling of equivariant and invariant layers. This paradigm is both expressive (Dym and Maron [20]) and domain-agnostic by not relying on hand-crafted features.

With the adoption of CNNs, translational symmetries have long been leveraged in many deep learning tasks through equivariance. The extension of purely translational to roto-translational symmetries in CNNs was investigated in the seminal work by Cohen and Welling [14]. This approach was restricted to 2D images and a small, discrete set of rotations. Since then, a wide variety of approaches for incorporating rotational symmetries into neural networks have been proposed. This section focuses on roto-translation equivariant approaches for point clouds and graphs in 3D. The approaches discussed here can be divided into three categories: (1) Simple transformations that trivially commute with rotations and can therefore be proven to be equivariant without using sophisticated mathematical machinery. This will be referred to as *type-1 representations*, although the meaning of the name will only be made clear later on in this section. (2) An expansion of the first category using the theory of *irreducible representations*, a mathematical concept based on group theory. (3) *Regular representations*, a fundamentally different approach that is also based on group theory and relies on keeping track of multiple versions of each feature vector, where each version is connected to a single element of the symmetry group.

**Type-1 Representations** The simplest possible approach for achieving rotation invariance in neural networks is to only feed rotation invariant information into the network in the first place, as done for example in Schütt et al. [58]. Any function with invariant inputs will yield invariant outputs. Therefore the network output is trivially invariant. The next step in complexity is to also consider vector features $\mathbf{v} \in \mathbb{R}^3$, such as the vector connecting one point to another (relative position) or the velocity of a particle. In this case, the input, the output, and the hidden representations of the network can have both scalar and vector components. Here, a vector is rotated

[2] A roto-translation is any transformation that can be decomposed into a rotation followed by a translation.

with an ordinary $3 \times 3$ rotation matrix. For a network to be rotation equivariant, a rotation of all vector inputs $\mathbf{v}_i$ with a rotation matrix $\mathbf{R}$ needs to result in a rotation of the vector outputs with the same rotation matrix $\mathbf{R}$. This can easily be achieved by constructing the layers of the neural network only from operations that (a) change the length of a vector by multiplying it with a rotation invariant scalar $\alpha$ or (b) linearly combine vector inputs with invariant weights $w_i$. The reason is that both operations commute with rotations:

$$\alpha \cdot \mathbf{R} \cdot \mathbf{v} = \mathbf{R} \cdot \alpha \cdot \mathbf{v} \tag{2.6}$$

$$w_1 \cdot \mathbf{R} \cdot \mathbf{v}_1 + w_2 \cdot \mathbf{R} \cdot \mathbf{v}_2 = \mathbf{R} \cdot (w_1 \mathbf{v}_1 + w_2 \mathbf{v}_2) \tag{2.7}$$

Here, $w_i$ and $\alpha$ could be learned parameters. Alternatively, they could be outputs of neural network layers themselves, in which case $w_i$ and $\alpha$ could be seen as invariant/scalar features. Scalar features do not change under rotation. The complexity and variety of the mathematical operations can further be increased by adding mappings from vector features to scalar features. Taking the norm of a vector results in an invariant feature. The same is true for scalar products of vector features:

$$||\mathbf{R} \cdot \mathbf{v}|| = ||\mathbf{v}|| \tag{2.8}$$

$$(\mathbf{R}\mathbf{v}_1)^\top \cdot \mathbf{R}\mathbf{v}_1 = \mathbf{v}_1^\top \cdot \mathbf{v}_2 \tag{2.9}$$

When designing an equivariant graph neural network, one can almost arbitrarily combine the operations above to design equivariant message passing and node updating steps. Examples of such approaches are Jumper et al. [34], Köhler et al. [38], and Satorras et al. [56, 57].

**Irreducible Representations** The scalar and vector features discussed above rotate according to the identity $\mathbf{I}$ and ordinary $3 \times 3$ rotation matrices, respectively. However, those are not the only valid representations of the group $SO(3)$ of 3D rotations. Importantly, all 3D rotation matrices $\rho(g)$ can be decomposed into a simpler, block-diagonal form:

$$\rho(g) = \mathbf{Q}^\top \left[ \bigoplus_\ell \mathbf{D}_\ell(g) \right] \mathbf{Q}, \tag{2.10}$$

where $g$ can be seen as an index for the specific rotation (for example, expressed as angle and rotation axis) and $\mathbf{Q}$ is an orthogonal change-of-basis matrix (Weiler et al. [66]). $\mathbf{D}_\ell$ are so-called irreducible representations, and, specifically for $SO(3)$, they bear the name Wigner-D matrices, where D stands for *Darstellung*, German for representation. Vectors that rotate according to $\mathbf{D}_\ell$ are called type-$\ell$ vectors with $\ell \in \{0, 1, 2, \ldots\}$ and have length $(2\ell + 1)$. The scalar and vector features from the previous section are therefore type-0 and type-1 features, respectively, in the framework of irreducible representations. Interestingly, when augmenting hidden representations of neural networks with features of higher types, additional equivariant transformations can be found adding to the complexity of the transformations discussed in the previous section (Kondor [39], Thomas et al. [59], and Weiler et al. [66]). Those transformations, constructed from spherical harmonics and Clebsch-Gordan coefficients, are discussed in detail in Chapter 5. Thomas et al. [59] used such an irreducible representation approach to build a convolutional network for point clouds. Weiler et al. [66] concurrently developed a similar approach for 3D voxel grids. Chapter 5 builds on top of these findings to build a graph neural network, the SE(3)-Transformer, based on irreducible representations and self-attention (Vaswani et al. [61]).

**Regular Representations** The seminal work in the field of rotation equivariance by Cohen and Welling [14], preceding all of the works mentioned above, uses a fundamentally different approach. The authors design convolutional neural network layers that are equivariant with respect to 90 degree rotations of the input image. Hence the group actions are rotations of 0, 90, 180, or 270 degrees, respectively. To achieve equivariance, Cohen and Welling [14] store copies of each group action in the hidden network layers. This approach is known as regular representations. Using (stochastic) sampling, this can be extended to continuous groups. Finzi et al. [21] and Hutchinson et al. [32] additionally use a mapping to the respective Lie Group to obtain a smoother features space. Finzi et al. [21] propose a graph neural network based on convolutions. In contrast, Hutchinson et al. [32] base their approach on

self-attention (Vaswani et al. [61]). An advantage of these regular representation approaches is that they are more flexible with respect to incorporating different kinds of symmetry groups.

**Iterative Equivariance** Potentially the most prominent application domain of equivariant graph neural networks is protein structure prediction. Only very recently, end-to-end deep learning approaches have been proposed for this task (Baek et al. [1] and Jumper et al. [34]). Both Baek et al. [1] and Jumper et al. [34] first predict an initial position estimate for each atom and then iteratively refine the estimate, all trained in an end-to-end fashion. Iterative position updates had previously not been at the focus of equivariant graph neural networks. The two concurrently developed approaches Satorras et al. [57] and Fuchs et al. [25] (Chapter 6) propose neural network architectures that perform iterative, SE(3)-equivariant position updates. Satorras et al. [57] use type-1 representations, where, after each layer, a velocity feature at each node updates the position of that node. The approach of Chapter 6 uses a similar principle but is based on the more general irreducible representation approach and works with arbitrary types.

### 2.2.3 Label Invariance

In the previous sections, the invariance was always connected to a known symmetry group allowing for the construction of guaranteeably invariant and equivariant neural network layers. If the symmetry group is not known or does not exist, an alternative is to let the network learn and/or approximate the invariance. This section covers different such approaches and therefore provides context for the work on learned invariance with neural stethoscopes in Chapter 7.

**Learned Invariance through Data Augmentation** A particularly popular variant of learned invariance is data augmentation. By training not just with the original images but also with, e.g., rotated versions of the input, the neural network learns to process rotated versions of the same image similarly (Quiroga et al. [54]). Benton et al. [3] incorporate the data augmentation into the forward pass and let the network

learn how to augment (e.g., rotate) the input within the forward pass. Intuitively, this could be interpreted as looking at an object from different angles. This approach has similarities with the Janossy pooling framework of Murphy et al. [49], when interpreting the permutations of the input as augmentations. In fact, Murphy et al. [49] also introduce a variant that approximates invariance with the goal of increasing computational efficiency through limiting the number of permutations processed by the network.

**Label Invariance for Domain Transfer** Label invariance is a specific type of learned invariance and is applicable in scenarios where labels for an auxiliary task are available, at least during training time. A popular application for label invariance is robustness versus domain transfer. Assuming there are two domains, $D_A$ and $D_B$ (e.g. day time and night time), the goal is typically to learn a supervised task (e.g. object detection from an RGB camera in an autonomous vehicle). There might be significantly more training data for domain $D_A$, but $D_B$ might be an equally or even more important domain at test time. Clearly, the sensor input itself differs strongly in this example. However, when training a neural network, in order to transfer knowledge between domains, it might be desirable to obtain a hidden representation that is domain agnostic. Interestingly, the domain label is often readily available, in this case via the timestamp of the images. The question, therefore, is how to force the network to be label invariant in a hidden layer. One valid approach is to compare the distributions of hidden representations for both labels and to enforce closeness on them explicitly (Long et al. [42] and Tzeng et al. [60]). Alternatively, similarly to Generative Adversarial Networks (Goodfellow et al. [28]), a discriminator can be applied to the hidden feature representation of each example, where the task of the discriminator is to identify the domain. In a minimax game setup, the encoder is trained to confuse the discriminator and is hence encouraged to extract domain invariant features (Bousmalis et al. [5], Ganin et al. [27], and Wulfmeier et al. [67, 68]).

**Label Invariance for General Nuisance Factors** The idea of robustness to domain shifts can be generalised to the concept of nuisance factors. In order to prevent a model overfitting to a specific nuisance factor, Louizos et al. [43] minimise the Maximum Mean Discrepancy between conditional distributions with different values of a binary nuisance variable in the conditioning set. This approach has two drawbacks: it is limited to discrete nuisance variables, and its computational cost scales exponentially with the number of states. Xie et al. [69] address both issues via adversarial training, optimising an encoding to confuse an additional discriminator, which aims to determine the values of nuisance parameters. This method assumes that the nuisance variable is known and is an input to the main model during training and deployment. Louppe et al. [44] follow a similar approach, applying the discriminator to the output of the main model instead of its intermediate representations.

**Label Invariance with Neural Stethoscopes** Inspired by many of the works above, as well as by related work on interpretability (Mirowski et al. [48]), auxiliary objectives (Jaderberg et al. [33] and Mirowski et al. [48]), and multi-task learning ( e. g., Caruana [9, 10]), Chapter 7 introduces *neural stethoscopes*. Neural stethoscopes are a general-purpose framework for quantifying the degree of importance of specific factors of influence in deep neural networks as well as for actively promoting and suppressing information as appropriate, in other words, enforcing label invariance.

# 3

# On the Limitations of Representing Functions on Sets

# On the Limitations of Representing Functions on Sets

**Edward Wagstaff** [*1] **Fabian B. Fuchs** [*1] **Martin Engelcke** [*1] **Ingmar Posner** [1] **Michael Osborne** [1]


## Abstract

Recent work on the representation of functions on sets has considered the use of summation in a latent space to enforce permutation invariance. In particular, it has been conjectured that the dimension of this latent space may remain fixed as the cardinality of the sets under consideration increases. However, we demonstrate that the analysis leading to this conjecture requires mappings which are highly discontinuous and argue that this is only of limited practical use. Motivated by this observation, we prove that an implementation of this model via continuous mappings (as provided by e.g. neural networks or Gaussian processes) actually imposes a constraint on the dimensionality of the latent space. Practical universal function representation for set inputs can only be achieved with a latent dimension at least the size of the maximum number of input elements.


## 1. Introduction

Machine learning models have had great success in taking advantage of structure in their input spaces: recurrent neural networks are popular models for sequential data (Sutskever et al., 2014) and convolutional neural networks are the state-of-the-art for many image-based problems (He et al., 2016). Recently, however, models for unstructured inputs in the form of sets have rapidly gained attention (Ravanbakhsh et al., 2016; Zaheer et al., 2017; Qi et al., 2017a; Lee et al., 2018; Murphy et al., 2018; Korshunova et al., 2018).

Importantly, a range of machine learning problems can naturally be formulated in terms of sets; e.g. parsing a scene composed of a set of objects (Eslami et al., 2016; Kosiorek et al., 2018), making predictions from a set of points forming a 3D point cloud (Qi et al., 2017a;b), or training a set of agents in reinforcement learning (Sunehag et al., 2017).

[*]Equal contribution [1]Department of Engineering Science, University of Oxford, Oxford, United Kingdom. Correspondence to: <{ed, fabian, martin}@robots.ox.ac.uk>.

*Proceedings of the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

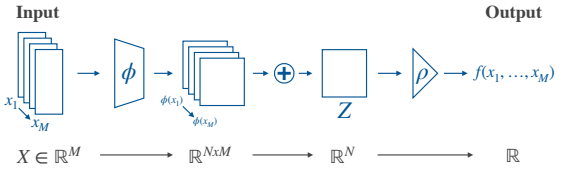



Figure 1: Illustration of the model structure proposed in several works (Zaheer et al., 2017; Qi et al., 2017a) for representing permutation-invariant functions. The sum operation enforces permutation invariance for the model as a whole. $\phi$ and $\rho$ can be implemented by e.g. neural networks.

Furthermore, attention-based models perform a weighted summation of a set of features (Vaswani et al., 2017; Lee et al., 2018). Hence, understanding the mathematical properties of set-based models is valuable both in terms of set-structured applications as well as better understanding the capabilities and limitations of attention-based models.

Many popular machine learning models, including neural networks and Gaussian processes, are fundamentally based on vector inputs[1] rather than set inputs. In order to adapt these models for use with sets, we must enforce the property of *permutation invariance*, i.e. the output of the model must not change if the inputs are reordered. Multiple authors, including Ravanbakhsh et al. (2016), Zaheer et al. (2017) and Qi et al. (2017a), have considered enforcing this property using a technique which we term *sum-decomposition*, illustrated in Figure 1. Mathematically speaking, we say that a function $f$ defined on sets of size $M$ is *sum-decomposable via* $Z$ if there are functions $\phi : \mathbb{R} \to Z$ and $\rho : Z \to \mathbb{R}$ such that[2]

$$f(X) = \rho\big(\Sigma_{x \in X} \phi(x)\big) \tag{1}$$

We refer to $Z$ here as the *latent space*. Since summation is permutation-invariant, a sum-decomposition is also permutation-invariant. Ravanbakhsh et al. (2016), Zaheer et al. (2017) and Qi et al. (2017b) have also considered the idea of enforcing permutation invariance using other operations, e.g. $\texttt{max}(\cdot)$. In this paper we concentrate on a detailed analysis of sum-decomposition, but some of the limitations we discuss also apply when $\texttt{max}(\cdot)$ is used instead of summation.

[1]Or inputs of higher rank, i.e. matrices and tensors.

[2]We use $\mathbb{R}$ here for brevity – see Definition 2.2 for the fully general definition.

Our main contributions can be summarised as follows.

1. Recent proofs, e.g. in Zaheer et al. (2017), consider functions on countable domains. We explain why considering countable domains can lead to results of limited practical value (i.e. cannot be implemented with a neural network), and why considering continuity on uncountable domains such as $\mathbb{R}$ is necessary. With reference to neural networks, we ground this discussion in the universal approximation theorem, which relies on continuity on uncountable domains $[0, 1]^M$.
2. In contrast to previous work (Zaheer et al., 2017; Qi et al., 2017a), which considers sufficient conditions for universal function representation, we establish a *necessary* condition for a sum-decomposition-based model to be capable of universal function representation. Additionally, we provide weaker sufficient conditions which imply a stronger version of universality. Specifically, we show that the dimension of the latent space being at least as large as the maximum number of input elements is both necessary and sufficient for universal function representation.

While primarily targeted at neural networks, these results hold for any implementation of sum-decomposition, e.g. using Gaussian processes, as long as it provides universal function approximation for continuous functions. Proofs of all novel results are available in Appendix B.

## 2. Preliminaries

In this section we recount the theorems and proofs on sum-decomposition from Zaheer et al. (2017). We begin by introducing important definitions and the notation used throughout our work. Note that we focus on permutation-invariant functions and do not discuss permutation equivariance which is also considered in Zaheer et al. (2017).

### 2.1. Definitions

**Definition 2.1.** A function $f(\mathbf{x})$ is *permutation-invariant* if $f(x_1, \ldots, x_M) = f\big(x_{\pi(1)}, \ldots, x_{\pi(M)}\big)$ for all $\pi$.

**Definition 2.2.** We say that a function $f$ is *sum-decomposable* if there are functions $\rho$ and $\phi$ such that

$$f(X) = \rho\big(\Sigma_{x \in X} \phi(x)\big).$$

In this case, we say that $(\rho, \phi)$ is a *sum-decomposition* of $f$.

Given a latent space $Z$, we say that $f$ is *sum-decomposable via* $Z$ when this expression holds for some $\phi$ whose codomain is $Z$, i.e. $\phi : \mathfrak{X} \to Z$.

We say that $f$ is *continuously sum-decomposable* when this expression holds for some continuous functions $\rho$ and $\phi$.

We will also consider sum-decomposability where the inputs to $f$ are vectors rather than sets - in this context, the sum is over the elements of the input vector.

**Definition 2.3.** A set $\mathfrak{X}$ is *countable* if its number of elements, i.e. the cardinality, is smaller or equal to the number of elements in $\mathbb{N}$. This includes both finite and countably infinite sets; e.g. $\mathbb{N}$, $\mathbb{Q}$, and subsets thereof.

**Definition 2.4.** A set $\mathfrak{X}$ is *uncountable* if its number of elements is greater than the number of elements in $\mathbb{N}$, e.g. $\mathbb{R}$ and certain subsets thereof.

**Notation 2.5.** Denote the power set of a set $\mathfrak{X}$ by $2^{\mathfrak{X}}$.

**Notation 2.6.** Denote the set of *finite* subsets of a set $\mathfrak{X}$ by $\mathfrak{X}^{\mathcal{F}}$.

**Notation 2.7.** Denote the set of subsets of a set $\mathfrak{X}$ containing at most $M$ elements by $\mathfrak{X}^{\leq M}$.

*Remark.* Throughout, we discuss expressions of the form $\Phi(X) = \Sigma_{x \in X} \phi(x)$, where $X$ is a set. Note that care must be taken in interpreting this expression when $X$ is not finite – we discuss this issue fully in Appendix A.1.

### 2.2. Background Theorems

Zaheer et al. (2017) consider the two cases where $X$ is a subset of, or drawn from, a *countable* and an *uncountable* universe $\mathfrak{X}$. We now outline the theorems and proofs relating to these two cases.

**Theorem 2.8** (Countable case)**.** *Let $f : 2^{\mathfrak{X}} \to \mathbb{R}$ where $\mathfrak{X}$ is countable. Then $f$ is permutation-invariant if and only if it is sum-decomposable via $\mathbb{R}$.*

*Proof.* Since $\mathfrak{X}$ is countable, each $x \in \mathfrak{X}$ can be mapped to a unique element in $\mathbb{N}$ by a function $c(x) : \mathfrak{X} \to \mathbb{N}$. Let $\Phi(X) = \sum_{x \in X} \phi(x)$. If we can choose $\phi$ so that $\Phi$ is injective, then we can set $\rho = f \circ \Phi^{-1}$, giving

$$f = \rho \circ \Phi$$
$$f(X) = \rho\big(\Sigma_{x \in X} \phi(x)\big)$$

i.e. f is sum-decomposable via $\mathbb{R}$.

Now consider $\phi(x) = 4^{-c(x)}$. Under this mapping, each $X \subset \mathfrak{X}$ corresponds to a unique real number expressed in base 4. Therefore $\Phi$ is injective, and the conclusion follows. □

*Remark.* This construction works for any set size $M$, and even for sets of infinite size. However, it assumes that $X$ is a set with no repeated elements, i.e. multisets are not supported. Specifically, the construction will fail with multisets because $\Phi$ fails to be injective if its domain includes multisets. In Appendix A.3, we extend Theorem 2.8 to also support multisets, with the restriction that infinite sets are no longer supported.

**Theorem 2.9** (Uncountable case)**.** *Let $M \in \mathbb{N}$, and let $f : [0,1]^M \to \mathbb{R}$ be a continuous function. Then $f$ is permutation-invariant if and only if it is continuously sum-decomposable via $\mathbb{R}^{M+1}$.*

The proof by Zaheer et al. (2017) of Theorem 2.9 is more involved than for Theorem 2.8. We do not include it here in full detail, but briefly summarise below.

1. Show that the mapping $\Phi : [0,1]^M \to \mathbb{R}^{M+1}$ defined by $\Phi_q(\mathbf{x}) = \sum_{m=1}^{M}(x_m)^q$ for $q = 0, \dots, M$ is injective and continuous.[3]
2. Show that $\Phi$ has a continuous inverse.
3. Define $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$ by $\rho = f \circ \Phi^{-1}$.
4. Define $\phi(x) : \mathbb{R} \to \mathbb{R}^{M+1}$ by $\phi_q(x) = x^q$.
5. Note that, by definition of $\rho$ and $\phi$, $(\rho, \phi)$ is a continuous sum-decomposition of $f$ via $\mathbb{R}^{M+1}$. □

*Remark.* Zaheer et al. (2017) conjecture that any continuous permutation-invariant function $f$ on $2^{[0,1]}$, the power set of $[0,1]$, is continuously sum-decomposable. In Section 3, we show that this is not possible, and in Section 4 we show that even if the domain of $f$ is restricted to $[0,1]^{\leq \mathcal{F}}$, the finite subsets of $[0,1]$, then $N \geq M$ is a *necessary condition* for arbitrary functions $f$ to be continuously sum-decomposable. Additionally, we prove that $N = M$ is a *sufficient condition* – implying together with the above that it is not possible to do better than this.

## 3. The Importance of Continuity

In this section, we argue that continuity is essential to discussions of function representation, that it has been neglected in prior work on permutation-invariant functions, and that this neglect has implications for the strength and generality of existing results.

Intuitively speaking a function is continuous if, at every point in the domain, the variation of the output can be made arbitrarily small by limiting the variation in the input. Continuity is the reason that, for instance, working to machine precision usually produces sensible results. Truncating to machine precision alters the input to a function slightly, but continuity ensures that the change in output is also slight.

In Zaheer et al. (2017), the authors demonstrate that when $\mathfrak{X}$ is a countable set, e.g. the rational numbers, any function $f : 2^{\mathfrak{X}} \to \mathbb{R}$ is sum-decomposable via $\mathbb{R}$. This is taken as a hopeful indication that sum-decomposability may extend to uncountable domains, e.g. $\mathfrak{X} = \mathbb{R}$. Extending to the uncountable case may appear, at first glance, to be a mere formality – we are, after all, ultimately interested in implementing algorithms on finite hardware. Nevertheless, it is not true that a theoretical result for a countably infinite domain must be strong enough for practical purposes. In fact, considering functions on uncountably infinite domains such as $\mathbb{R}^N$ is of real importance.

[3] In the original proof, $\Phi$ is denoted $E$.

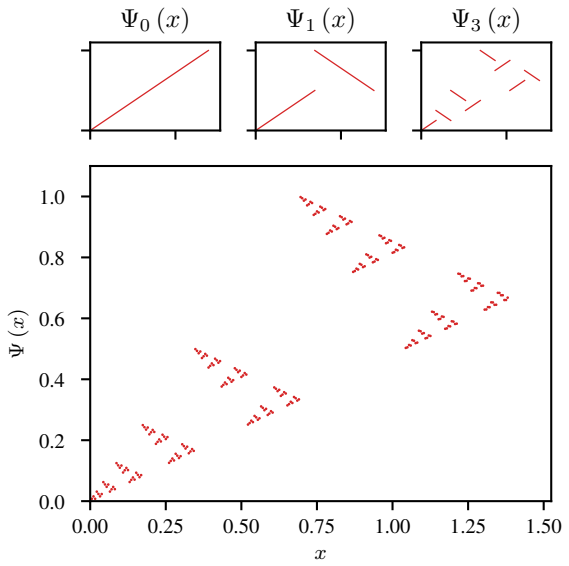



Figure 2: The function $\Psi$ shown here is continuous at every rational point in $[0, \ln 4]$. Intuitively, this is because all jumps occur at irrational values, namely at certain fractions of $\ln 4$. It defies our intuitions for what continuity should mean, and illustrates the fact that continuity on $\mathbb{Q}$ is a much weaker property than continuity on $\mathbb{R}$. The latter property is required to satisfy the universal approximation theorem for neural networks. $\Psi$ is defined and discussed in Appendix C.

Turning specifically to neural networks, the universal approximation theorem says that any *continuous* function can be approximated by a neural network, but not that *any* function can be approximated by a neural network (Cybenko, 1989). A similar statement is true for other approximators, such as some Gaussian processes (Rasmussen & Williams, 2006). The notion of continuity required here is specifically that of continuity on compact subsets of $\mathbb{R}^N$.

Crucially, if we wish to work mathematically with continuity in a way that closely matches our intuitions, we *must* consider uncountable domains. To illustrate this point, consider the rational numbers $\mathbb{Q}$. $\mathbb{Q}$ is dense in $\mathbb{R}$, and it is tempting to think that $\mathbb{Q}$ is therefore "all we need". However, a theoretical guarantee of continuity on $\mathbb{Q}$ is weak, and does not imply continuity on $\mathbb{R}$. The universal approximation theorem for neural networks relies on continuity on $\mathbb{R}$, and we cannot usefully take continuity on $\mathbb{Q}$ as a proxy for this

property. Figure 2 shows a function which is continuous on $\mathbb{Q}$, and illustrates that a continuous function on $\mathbb{Q}$ may not extend continuously to $\mathbb{R}$. This figure also illustrates that continuity on $\mathbb{Q}$ defies our intuitions about what continuity should mean, and is too weak for the universal approximation theorem for neural networks. We require the stronger notion of continuity on $\mathbb{R}$.

In light of the above, it is clear that continuity is a key property for function representation, and also that there is a crucially important difference between countable and uncountable domains. This raises two problems for Theorem 2.8. First, the theorem does not consider the continuity of the sum-decomposition when the domain $\mathfrak{X}$ has some non-trivial topological structure (e.g. $\mathfrak{X} = \mathbb{Q}$). Second, we still care about continuity on $\mathbb{R}$, and there is no guarantee that this is possible given continuity on $\mathbb{Q}$.

In fact, the continuity issue cannot be overcome – we can demonstrate that in general the sum-decomposition of Theorem 2.8, which goes via $\mathbb{R}$, cannot be made continuous for $\mathfrak{X} = \mathbb{Q}$:

**Theorem 3.1.** *There exist functions $f : 2^{\mathbb{Q}} \to \mathbb{R}$ such that, whenever $(\rho, \phi)$ is a sum-decomposition of $f$ via $\mathbb{R}$, $\phi$ is discontinuous at every point $q \in \mathbb{Q}$.*

We can actually say something more general than the above. Our proof can easily be adapted to demonstrate that if $f$ is injective, or if we want a fixed $\phi$ to suffice for any $f$, then $\phi$ can only be continuous at isolated points of the underlying set $\mathfrak{X}$, regardless of whether $\mathfrak{X} = \mathbb{Q}$. I.e., it is not specifically due to the structure of $\mathbb{Q}$ that continuous sum-decomposability fails. In fact, it fails whenever we have a non-trivial topological structure. For functions which we want to model using a neural network, this is worrying.

It is not possible to represent an everywhere-discontinuous $\phi$ with a neural network. We therefore view Theorem 2.8 as being of limited practical relevance and as not providing a reliable intuition for what should be possible in the uncountable case. We do however see this result as mathematically interesting, and have obtained the following result extending it to the case where the domain $\mathfrak{X}$ is uncountable. This result is slightly weaker than the countable case, in that the domain of $f$ can contain arbitrarily large finite sets, but not infinite sets.

**Theorem 3.2.** *Let $f : \mathbb{R}^{\mathcal{F}} \to \mathbb{R}$. Then $f$ is sum-decomposable via $\mathbb{R}$.*

Once again, the sum-decomposition is highly discontinuous. The limitation that $f$ is not defined on infinite sets cannot be overcome:

**Theorem 3.3.** *If $\mathfrak{X}$ is uncountable, then there exist functions $f : 2^{\mathfrak{X}} \to \mathbb{R}$ which are not sum-decomposable. Note that this holds even if the sum-decomposition $(\rho, \phi)$ is allowed to be discontinuous.*

To summarise, we show why considering countable domains can lead to results of limited practical value and why considering continuity on uncountable domains is necessary. We point out that some of the previous work is therefore of limited practical relevance, but regard it as mathematically interesting. In this vein, we extend the analysis of sum-decomposability when continuity is not required.

## 4. Practical Function Representation

Having established the necessity of considering continuity on $\mathbb{R}$, we now explore the implications for sum-decomposability of permutation-invariant functions. These considerations lead to concrete recommendations for model design and provide theoretical support for elements of current practice in the area. Specifically, we present three theorems whose implications can be summarised as follows.

1. A latent dimensionality of $M$ is *sufficient* for representing all continuous permutation-invariant functions on sets of size $\leq M$.
2. To guarantee that all continuous permutation-invariant functions can be represented for sets of size $\leq M$, a latent dimensionality of at least $M$ is *necessary*.

The key result which is the basis of the second statement and which underpins this discussion is as follows.

**Theorem 4.1.** *Let $M > N \in \mathbb{N}$. Then there exist permutation invariant continuous functions $f : \mathbb{R}^M \to \mathbb{R}$ which are* **not** *continuously sum-decomposable via $\mathbb{R}^N$.*

Restated in more practical terms, this implies that for a sum-decomposition-based model to be capable of representing *arbitrary* continuous functions on sets of size $M$, the latent space in which the summation happens must be chosen to have dimension at least $M$. A similar statement is true for the analogous concept of *max-decomposition* – details are available in Appendix B.6.

To prove this theorem, we first need to state and prove the following lemma.

**Lemma 4.2.** *Let $M, N \in \mathbb{N}$, and suppose $\phi : \mathbb{R} \to \mathbb{R}^N$, $\rho : \mathbb{R}^N \to \mathbb{R}$ are functions such that:*

$$max(X) = \rho\left(\Sigma_{x \in X} \phi(x)\right) \tag{2}$$

*Now let $\Phi(X) = \Sigma_{x \in X} \phi(x)$, and write $\Phi_M$ for the restriction of $\Phi$ to sets of size $M$.*

*Then $\Phi_M$ is injective for all $M$.*

*Proof.* We proceed by induction. The base case $M = 1$ is clear.

Now let $M \in \mathbb{N}$, and suppose that $\Phi_{M-1}$ is injective. Now suppose there are sets $X, Y$ such that $\Phi_M(X) = \Phi_M(Y)$. First note that, by (2), we must have:

$$\max(X) = \max(Y) \tag{3}$$

So now write:

$$X = \{x_{\max}\} \cup X_{\text{rem}} \; ; \; Y = \{y_{\max}\} \cup Y_{\text{rem}} \tag{4}$$

where $x_{\max} = \max(X)$, and similarly for $y_{\max}$.

But now:

$$\begin{aligned} \Phi_M(X) &= \Phi_{M-1}(X_{\text{rem}}) + \phi(x_{\max}) \\ &= \Phi_{M-1}(Y_{\text{rem}}) + \phi(y_{\max}) \\ &= \Phi_M(Y) \end{aligned}$$

From the central equality, and (3), we have:

$$\Phi_{M-1}(X_{\text{rem}}) = \Phi_{M-1}(Y_{\text{rem}})$$

Now by injectivity of $\Phi_{M-1}$, we have $X_{\text{rem}} = Y_{\text{rem}}$. Combining this with (3) and (4), we must have $X = Y$, and so $\Phi_M$ is injective. □

Equipped with this lemma, we can now prove Theorem 4.1.

*Proof.* We proceed by contradiction. Suppose that functions $\phi$ and $\rho$ exist satisfying (2). Define $\Phi_M : \mathbb{R}^M \to \mathbb{R}^N$ by:

$$\Phi_M(\mathbf{x}) = \Sigma_{i=1}^M \phi(x_i)$$

Denote the set of all $x \in \mathbb{R}^M$ with $x_1 < x_2 < ... < x_M$ by $\mathbb{R}^M_{\text{ord}}$, and let $\Phi^{\text{ord}}_M$ be the restriction of $\Phi_M$ to $\mathbb{R}^M_{\text{ord}}$. Since $\Phi^{\text{ord}}_M$ is a sum of continuous functions, it is also continuous, and by Lemma 4.2, it is injective.

Now note that $\mathbb{R}^M_{\text{ord}}$ is a convex open subset of $\mathbb{R}^M$, and is therefore homeomorphic to $\mathbb{R}^M$. Therefore, our continuous injective $\Phi^{\text{ord}}_M$ can be used to construct a continuous injection from $\mathbb{R}^M$ to $\mathbb{R}^N$. But it is well known that no such continuous injection exists when $M > N$. Therefore our decomposition (2) cannot exist. □

It is crucial to note that functions $f$ for which a lower-dimensional sum-decomposition does not exist need not be "badly-behaved" or difficult to specify. The limitation extends to functions of genuine interest. For our proof, we have specifically demonstrated that even $\texttt{max}(X)$ is not continuously sum-decomposable when $N < M$.

From Theorem 2.9, we also know that for a fixed input set size $M$, any continuous permutation-invariant function is continuously sum-decomposable via $\mathbb{R}^{M+1}$. It is, however, possible to adapt the construction of Zaheer et al. (2017) to strengthen the result in two ways. Firstly, we can perform the sum-decomposition via $\mathbb{R}^M$:

**Theorem 4.3** (Fixed set size)**.** *Let $f : \mathbb{R}^M \to \mathbb{R}$ be continuous. Then $f$ is permutation-invariant if and only if it is continuously sum-decomposable via $\mathbb{R}^M$.*

Secondly, we can deal with variable set sizes $\leq M$:

**Theorem 4.4** (Variable set size)**.** *Let $f : \mathbb{R}^{\leq M} \to \mathbb{R}$ be continuous. Then $f$ is permutation-invariant if and only if it is continuously sum-decomposable via $\mathbb{R}^M$.*

Note that we must take some care over the notion of continuity in this theorem – see Appendix A.2.

### 4.1. Discussion

Theorem 4.1 does not imply *all* functions require $N = M$. Some functions, such as the mean, can be represented in a lower dimensional space. The statement rather says that if we do not want to impose any limitations on the complexity of the function, the latent space needs to have dimensionality at least $M$.

Theorem 4.4 suggests that sum-decomposition via a latent space with dimension $N = M$ should suffice to model any function. Neural network models in the recent literature, however, deviate from these guidelines in several ways, indicating a disconnect between theory and practice. For example, the models in Zaheer et al. (2017) and Qi et al. (2017a) are considerably more complex than Equation (1), e.g. they apply several permutation-equivariant layers to the input before a permutation-invariant layer.

In light of Theorem 4.1, this disconnect becomes less surprising. We have shown that, for a target function of sufficient complexity, $N = M$ is the bare minimum required for the model to be capable of representing the target function. Achieving this would rely on the parameterisation of $\phi$ and $\rho$ being flexible enough and on the availability of a suitable optimisation method. In practice, we should not be surprised that more than the bare minimum capacity in our model is required for good performance. Even with $N > M$, the model might not converge to the desired solution. At the same time, when we are dealing with real datasets, the training data may contain noise and redundant information, e.g. in the form of correlations between elements in the input, inducing functions of limited complexity that may in fact be representable with $N < M$.

### 4.2. Illustrative Example

We now use a toy example to illustrate some practical implications of our results. Based on Theorem 4.1, we expect the number of input elements $M$ to have an influence on the required latent dimension $N$, and in particular, we expect that the required latent dimension may increase without bound.

We train a neural network with the architecture presented in Figure 1 to predict the median of a set of values. We choose the median as a function because it is relatively simple but cannot be trivially represented via a sum in a fixed-dimensional latent space, in contrast to e.g. the mean, which is sum-decomposable via $\mathbb{R}$.[4] $\phi$ and $\rho$ are parameterised by multi-layer perceptrons (MLPs). The input sets are randomly drawn from either a uniform, a Gaussian, or a Gamma distribution.

We vary the latent dimension $N$ and the input set size $M$ to investigate the link between these two variables and the predictive performance. The MLPs parameterising $\phi$ and $\rho$ are given comparatively many layers and hidden units, relative to the simplicity of the task, to ensure that the latent dimension is the bottleneck. Further details are described in Appendix D.

Figure 3(a) shows the RMSE depending on the latent dimension for different input sizes. We make three observations.

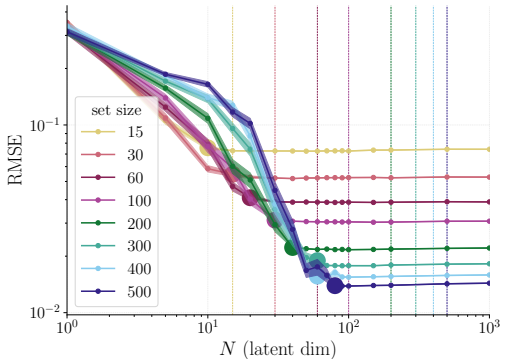
1. For each set size, the error decreases monotonically with the dimension of the latent space.

2. Beyond a certain point, increasing the dimension of the latent space does not further reduce the error. We denote this the "critical point".

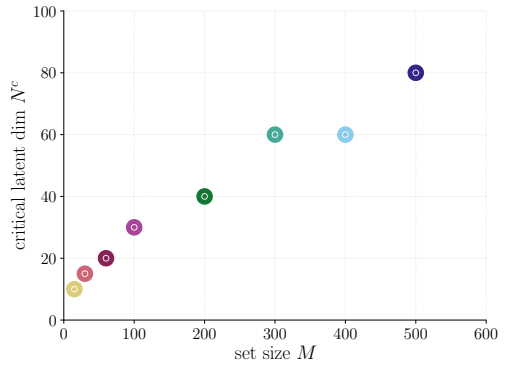
3. As the set size increases, so does the latent dimension at the critical point.

Figure 3(b) shows the critical points as a function of the input size, indicating a roughly linear relationship between the two. Note that the critical points occur at $N < M$. This can be explained by the fact that the models do not learn an algorithmic solution for computing the median, but rather to estimate it given samples drawn from the specific input distribution seen during training. Furthermore, estimating the median of a distribution, like other functions, renders some information in the input redundant. Therefore, the mapping from input to latent space does not need to be injective, allowing a model to solve the task with a smaller value of $N$.

[4]The construction for $Z = \mathbb{R}$ is not entirely trivial for variable set size, but going via $Z = \mathbb{R}^2$ is straightforward.




(a) Test performance on median estimation depending on latent dimension. Different colours depict different set sizes. Each data point is averaged over 500 runs with different seeds. Shaded areas indicate confidence intervals. Coloured dashed lines indicate $N = M$.




(b) Extracted 'critical points' from above graph. The coloured data points depict minimum latent dimension for optimal performance (RMSE less than 10% above minimum value for this set size) for different set sizes.

Figure 3: Illustrative toy example: a neural network is trained to predict the median of an unordered set.

## 5. Related Work

Much of the recent work on deep learning with unordered sets follows the paradigm discussed in (Ravanbakhsh et al., 2016), Zaheer et al. (2017), and Qi et al. (2017a) which leverage the structure illustrated in Figure 1. Zaheer et al. (2017) provide an in-depth theoretical analysis which is discussed in detail in Section 2. Qi et al. (2017a) also derive a sufficiency condition for universal function approximation. In their proof, however, they set the latent dimension $N$ to $\lceil 1/\delta_\epsilon \rceil$ where $\delta_\epsilon$ depends on the error tolerance for how closely the target function has to be approximated. As a result, the latent dimension $N$ goes to infinity for exact representation. In similar vain, Herzig et al. (2018) consider permutation-invariant functions on graphs.

A key application domain of set-based methods is the processing of point clouds, as the constituent points do not

have an intrinsic ordering. The work by Qi et al. (2017a) on 3D point clouds, one of the first to use a permutation-invariant neural networks, is extended in Qi et al. (2017b) by sampling and grouping points in a hierarchical fashion to model the interaction between nearby points in the input space more explicitly. Qi et al. (2018) combine RGB and lidar data for object detection by using image detectors to generate bounding box proposals which are then further processed by a set-based model. Achlioptas et al. (2018) and Yi et al. (2018) show that set-based models can also be used to learn generative models of point clouds.

Vinyals et al. (2015) suggest that even though recurrent networks are universal approximators, the ordering of the input is crucial for good performance. Hence, they propose model that relies on attention to achieve permutation invariance in order to solve a sorting task. In general, it is worth noting that there exists a connection between the model in Zaheer et al. (2017) and recent attention-based models such as the one proposed in Vaswani et al. (2017). In this case, the aggregation layer includes a weighting parameter which is computed based on a key-query system which is also permutation invariant. Since the value of the weighting parameters could be learned to be 1.0, it is trivial to show that such an attention algorithm is also in principle able to approximate any permutation-invariant function, of course depending on the remaining parts of the architecture. Inspired by inducing point methods, Set Transformer (Lee et al., 2018) propose a computationally more efficient attention-module and demonstrate better performance on a range of set-based tasks. While stacking several of attention-modules can capture higher order dependencies, a more general treatment of this is offered by permutation-invariant, learnable Janossy Pooling (Murphy et al., 2018).

Similar to the methods considered here, Neural Processes (Garnelo et al., 2018b) and Conditional Neural Processes (Garnelo et al., 2018a) also rely on aggregation via summation in order to infer a distribution from a set of data points. Kim et al. (2019) add an attention mechanism to neural processes to improve empirical performance. Generative Query Networks (Eslami et al., 2018; Kumar et al., 2018) can be regarded as an instantiation of neural processes to learn useful representations of 3D scenes from multiple 2D views. Yang et al. (2018) also aggregate information from multiple views to compute representations of 3D objects.

Bloem-Reddy & Teh (2019) and Korshunova et al. (2018) consider exchangeable sequences – sequences consisting of random variables with a joint likelihood which is invariant under permutations. Bloem-Reddy & Teh (2019) provide a theorem that describes distribution-invariant models. Korshunova et al. (2018) use RealNVP (Dinh et al., 2016) as a bijective function which sequentially computes the parameters of a Student-t process.

## 6. Conclusions

This work derives theoretical limitations on the representation of *arbitrary* functions on sets via a finite latent space. We demonstrate why continuity requires statements on uncountable domains, as opposed to countable domains, to ensure the practical usefulness of those statements. Under this constraint, we prove that a latent space whose dimension is at least as large as the maximum input set size is both sufficient and necessary to achieve *universal* function representation. The models covered in this analysis are popular for a range of practical applications and can be implemented e.g. by neural networks or Gaussian processes. In future work, we would like to investigate the effect of constructing models with both permutation-equivariant and permutation-invariant modules on the required dimension of the latent space. Examining the implications of using self-attention, e.g. as in Lee et al. (2018), would be of similar interest.

## Acknowledgements

This research was funded by the EPSRC AIMS Centre for Doctoral Training at the University of Oxford, an EPSRC DTA studentship, a Google studentship, and an EPSRC Programme Grant (EP/M019918/1). The authors acknowledge use of Hartree Centre resources in this work. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. The authors thank Sudhanshu Kasewa and Olga Isupova for proof reading a draft of the paper.

## References

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. Learning Representations and Generative Models for 3D Point Clouds. *International Conference on Machine Learning*, 2018.

Bloem-Reddy, B. and Teh, Y. W. Probabilistic Symmetry and Invariant Neural Networks. *arXiv preprint arXiv:1901.06082*, 2019.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., and Hinton, G. E. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. *Advances in Neural Information Processing Systems*, 2016.

Eslami, S. M. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., Reichert, D. P., Buesing, L., Weber, T., Vinyals, O., Rosenbaum, D., Rabinowitz, N. C., King, H., Hillier, C., Botvinick, M. M., Wierstra, D., Kavukcuoglu, K., and Hassabis, D. Neural scene representation and rendering. *Science*, 360:1204–1210, 2018.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. Conditional Neural Processes. *International Conference on Machine Learning*, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Teh, Y. W. Neural Processes. *International Conference on Machine Learning*, 2018b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Herzig, R., Raboh, M., Chechik, G., Berant, J., and Globerson, A. Mapping Images to Scene Graphs with Permutation-Invariant Structured Prediction. *arXiv preprint arXiv:1802.05451*, 2018.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive Neural Processes. *arXiv preprint arXiv:1901.05761*, 2019.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

Korshunova, I., Degrave, J., Huszár, F., Gal, Y., Gretton, A., and Dambre, J. BRUNO: A Deep Recurrent Model for Exchangeable Data. *Advances in Neural Information Processing Systems*, 2018.

Kosiorek, A. R., Kim, H., Posner, I., and Teh, Y. W. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. *Advances in Neural Information Processing Systems*, 2018.

Kumar, A., Eslami, S., Rezende, D. J., Garnelo, M., Viola, F., Lockhart, E., and Shanahan, M. Consistent Generative Query Networks. *arXiv preprint arXiv:1807.02033*, 2018.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. Set Transformer. *arXiv preprint arXiv:1810.00825*, 2018.

Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs. *arXiv preprint arXiv:1811.01900*, 2018.

Qi, C. R., Su, H., Mo, K., and Guibas, L. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017a.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems*, 2017b.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. Frustum PointNets for 3D Object Detection from RGB-D Data. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN 026218253X.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Deep Learning with Sets and Point Clouds. *arXiv preprint arXiv:1611.04500*, 2016.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *International Conference on Autonomous Agents and MultiAgent Systems*, 2017.

Sutskever, I., Vinyals, O., and Le, Q. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, 2014.

Vaswani, A., Shazeer, N., Parmer, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Yang, B., Wang, S., Markham, A., and Trigoni, N. Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. *arXiv preprint arXiv:1808.00758*, 2018.

Yi, L., Zhao, W., Wang, H., Sung, M., and Guibas, L. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. *arXiv preprint arXiv:1812.03320*, 2018.

Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. Deep Sets. In *Advances in Neural Information Processing Systems*, 2017.

# A. Mathematical Remarks

## A.1. Infinite Sums

Throughout this paper we consider expressions of the following form:

$$\Phi(X) = \Sigma_{x \in X} \phi(x) \tag{5}$$

Where $X$ is an arbitrary set. The meaning of this expression is clear when $X$ is finite, but when $X$ is infinite, we must be precise about what we mean.

### A.1.1. Countable Sums

We usually denote countable sums as e.g. $\Sigma_{i=1}^{\infty} x_i$. Note that there is an ordering of the $x_i$ here, whereas there is no ordering in our expression (5). The reason that we consider sums is for their permutation invariance in the finite case, but note that in the infinite case, permutation invariance of sums does not necessarily hold! For instance, the alternating harmonic series $\Sigma_{i=1}^{\infty} \frac{(-1)^i}{i}$ can be made to converge to any real number simply by reordering the terms of the sum. For expressions like (5) to make sense, we must require that the sums in question are indeed permutation invariant. This property is known as *absolute convergence*, and it is equivalent to the property that the sum of absolute values of the series converges. So for (5) to make sense, we will require everywhere that $\Sigma_{x \in X} |\phi(x)|$ is convergent. For any $X$ where this is not the case, we will set $\Phi(X) = \infty$.

### A.1.2. Uncountable Sums

It is well known that a sum over an uncountable set of elements only converges if all but countably many elements are 0. Allowing sums over uncountable sets is therefore of little interest, since it essentially reduces to the countable case.

## A.2. Continuity of Functions on Sets

We are interested in functions on subsets of $\mathbb{R}$, i.e. elements of $2^{\mathbb{R}}$, and the notion of continuity on $2^{\mathbb{R}}$ is not straightforward. As a convenient shorthand, we discuss "continuous" functions $f$ on $2^{\mathbb{R}}$, but what we mean by this is that the function $f_M$ induced by $f$ on $\mathbb{R}^M$ by $f_N(x_1, ..., x_M) = f(\{x_1, ..., x_M\})$ is continuous for every $M \in \mathbb{N}$.

## A.3. Remark on Theorem 2.8

The proof for Theorem 2.8 from Zaheer et al. (2017) can be extended to dealing with multi sets, i.e. sets with repeated elements. To that end, we replace the mapping to natural numbers $c(X) : \mathbb{R}^M \to \mathbb{N}$ with a mapping to prime numbers $p(X) : \mathbb{R}^M \to \mathbb{P}$. We then choose $\phi(x_m) = -\log p(x_m)$. Therefore,

$$\Phi(X) = \sum_{m=1}^{M} \phi(x_m) = \log \prod_{m=1}^{M} \frac{1}{p(x_m)} \tag{6}$$

which takes a unique value for each distinct $X$ therefore extending the validity of the proof to multi-sets. However, unlike the original series, this choice of $\phi$ diverges with infinite set size.

In fact, it is straightforward to show that there is no function $\phi$ for which $\Phi$ provides a unique mapping for arbitrary multi-sets while at same time guaranteeing convergence for infinitely large sets. Assume a function $\phi$ and an arbitrary point $x$ such that $\phi(x) = a \neq 0$. Then, the multiset comprising infinitely many identical members $x$ would give:

$$\Phi(X) = \sum_{i=1}^{\infty} \phi(x_m) = \sum_{i=1}^{\infty} a = \pm\infty \tag{7}$$

# B. Proofs of Theorems

## B.1. Theorem 3.1

**Theorem 3.1.** *There exist functions $f : 2^{\mathbb{Q}} \to \mathbb{R}$ such that, whenever $(\rho, \phi)$ is a sum-decomposition of $f$ via $\mathbb{R}$, $\phi$ is discontinuous at every point $q \in \mathbb{Q}$.*

*Proof.* Consider $f(X) = \sup(X)$, the least upper bound of $X$. Write $\Phi(X) = \Sigma_{x \in X} \phi(x)$. So we have:

$$\sup(X) = \rho(\Phi(X))$$

First note that $\phi(q) \neq 0$ for any $q \in \mathbb{Q}$. If we had $\phi(q) = 0$, then we would have, for every $X \subset \mathbb{Q}$:

$$\Phi(X) = \Phi(X) + \phi(q) = \Phi(X \cup \{q\})$$

But then, for instance, we would have:

$$q = \sup(\{q-1, q\}) = \sup(\{q-1\}) = q - 1$$

This is a contradiction, so $\phi(q) \neq 0$.

Next, note that $\Phi(X)$ must be finite for every upper-bounded $X \subset \mathbb{Q}$ (since sup is undefined for unbounded $X$, we do not consider such sets, and may allow $\Phi$ to diverge). Even if we allowed the domain of $\rho$ to be $\mathbb{R} \cup \{\infty\}$, suppose $\Phi(X) = \infty$ for some upper-bounded set $X$. Then:

$$\begin{aligned}\sup(X) &= \rho(\Phi(X)) \\ &= \rho(\infty) \\ &= \rho(\infty + \phi(\sup(X) + 1)) \\ &= \rho(\Phi(X \cup \{sup(X) + 1\})) \\ &= \sup(X \cup \{sup(X) + 1\}) \\ &= sup(X) + 1\end{aligned}$$

This is a contradiction, so $\Phi(X) < \infty$ for any upper-bounded set $X$.

Now from the above it is immediate that, for any upper-bounded set $X$, only finitely many $x \in X$ can have $\phi(x) > \frac{1}{n}$. Otherwise we can find an infinite upper-bounded set $Y \subset X$ with $\phi(y) > \frac{1}{n}$ for every $y \in Y$, and $\Phi(Y) = \infty$.

Finally, let $q \in \mathbb{Q}$. We have already shown that $\phi(q) \neq 0$, and we will now construct a sequence $q_n$ with:

1. $q_n \to q$
2. $\phi(q_n) \to 0$

If $\phi$ were continuous at $q$, we would have $\phi(q_n) \to \phi(q)$, so the above two points together will give us that $\phi$ is discontinuous at $q$.

So now, for each $n \in \mathbb{N}$, consider the set $B_n$ of points which lie within $\frac{1}{n}$ of $q$. Since only finitely many points $p \in B_n$ have $\phi(p) > \frac{1}{n}$, and $B_n$ is infinite, there must be a point $q_n \in B_n$ with $\phi(q_n) < \frac{1}{n}$. The sequence of such $q_n$ clearly satisfies both points above, and so $\phi$ is discontinuous everywhere. □

### B.2. Theorem 3.2

**Theorem 3.2.** *Let $f : \mathbb{R}^{\mathcal{F}} \to \mathbb{R}$. Then $f$ is sum-decomposable via $\mathbb{R}$.*

*Proof.* Define $\Phi : \mathbb{R}^{\mathcal{F}} \to \mathbb{R}$ by $\Phi(X) = \Sigma_{x \in X} \phi(x)$. If we can demonstrate that there exists some $\phi$ such that $\Phi$ is injective, then we can simply choose $\rho = f \circ \Phi^{-1}$ and the result is proved.

Say that a set $X \subset \mathbb{R}$ is *finite-sum-distinct* if, for any finite subsets $A, B \subset X$, $\Sigma_{a \in A} a \neq \Sigma_{b \in B} b$. Now, if we can show that there is a finite-sum-distinct set $D$ with the same cardinality as $\mathbb{R}$ (we denote $|\mathbb{R}|$ by $\mathfrak{c}$), then we can simply choose $\phi$ to be a bijection from $\mathbb{R}$ to $D$. Then, by finite-sum-distinctness, $\Phi$ will be injective, and the result is proved.

Now recall the statement of Zorn's Lemma: suppose $\mathcal{P}$ is a partially ordered set (or *poset*) in which every totally ordered subset has an upper bound. Then $\mathcal{P}$ has a maximal element.

The set of f.s.d. subsets of $\mathbb{R}$ (which we will denote $\mathcal{D}$) forms a poset ordered by inclusion. Supposing that $\mathcal{D}$ satisfies the conditions of Zorn's Lemma, it must have a maximal element, i.e. there is a f.s.d. set $D_{\max}$ such that any set $E$ with $D_{\max} \subsetneq E$ is not f.s.d. We claim that $D_{\max}$ has cardinality $\mathfrak{c}$.

To see this, let $D$ be a f.s.d. set with infinite cardinality $\kappa < \mathfrak{c}$ (any maximal $D$ clearly cannot be finite). We will show that $D \neq D_{\max}$. Define the *forbidden elements* with respect to $D$ to be those elements $x$ of $\mathbb{R}$ such that $D \cup \{x\}$ is not f.s.d. We denote this set of forbidden elements $F_D$. Now note that, if $D$ is maximal, then $D \cup F_D = \mathbb{R}$. In particular, this implies that $|F_D| = \mathfrak{c}$. But now consider the elements of $F_D$. By definition of $F_D$, we have that $x \in F_D$ if and only if $\exists c_1, ..., c_m, d_1, ..., d_n \in D$ such that $c_1 + ... + c_m + x = d_1 + ... + d_n$. So we can write $x$ as a sum of finitely many elements of $D$, minus a sum of finitely many other elements of $D$. So there is a surjection from pairs of finite sets of $D$ to elements of $F_D$. i.e.:

$$|F_D| \leq |D^{\mathcal{F}} \times D^{\mathcal{F}}|$$

But since $D$ is infinite:

$$|D^{\mathcal{F}} \times D^{\mathcal{F}}| = |D| = \kappa < \mathfrak{c}$$

So $|F_D| < \mathfrak{c}$, and therefore $|D|$ is not maximal. This demonstrates that $D_{\max}$ must have cardinality $\mathfrak{c}$.

To complete the proof, it remains to show that $\mathcal{D}$ satisfies the conditions of Zorn's Lemma, i.e. that every totally ordered subset (or *chain*) $\mathcal{C}$ of $\mathcal{D}$ has an upper bound. So consider:

$$C_{\text{ub}} = \bigcup \mathcal{C} = \bigcup_{C \in \mathcal{C}} C$$

We claim that $C_{\text{ub}}$ is an upper bound for $\mathcal{C}$. It is clear that $C \subset C_{\text{ub}}$ for every $C \in \mathcal{C}$, so it remains to be shown that $C_{\text{ub}} \in \mathcal{D}$, i.e. that $C_{\text{ub}}$ is f.s.d.

We proceed by contradiction. Suppose that $C_{\text{ub}}$ is not f.s.d. Then:

$$\exists c_1, ..., c_m, d_1, ..., d_n \in C_{\text{ub}} : \Sigma_i c_i = \Sigma_j d_j \qquad (8)$$

But now by construction of $C_{\text{ub}}$ there must be sets $C_1, ..., C_m, D_1, ..., D_m \in \mathcal{C}$ with $c_i \in C_i, d_j \in D_j$. Let $\mathcal{B} = \{C_i\}_{i=1}^m \cup \{D_j\}_{j=1}^n$. $\mathcal{B}$ is totally ordered by inclusion and all sets contained in it are f.s.d., since it is a subset of $\mathcal{C}$. Since $\mathcal{B}$ is finite it has a maximal element $B_{\max}$. By maximality, we have $c_i, d_j \in B_{\max}$ for all $c_i, d_j$. But then by (8), $B_{\max}$ is not f.s.d., which is a contradiction. So we have that $C_{\text{ub}}$ is f.s.d.

In summary:

1. $\mathcal{D}$ satisfies the conditions of Zorn's Lemma.
2. Therefore there exists a maximal f.s.d. set, $D_{\text{max}}$.
3. We have shown that any such set must have cardinality $\mathfrak{c}$.
4. Given an f.s.d. set $D_{\text{max}}$ with cardinality $\mathfrak{c}$, we can choose $\phi$ to be a bijection between $\mathbb{R}$ and $D_{\text{max}}$.
5. Given such a $\phi$, we have that $\Phi(X) = \Sigma_{x \in X} \phi(x)$ is injective on $R^{\mathcal{F}}$.
6. Given injective $\Phi$, choose $\rho = f \circ \Phi^{-1}$.
7. This choice gives us $f(X) = \rho(\Sigma_{x \in X} \phi(x))$ by construction.

This completes the proof. □

### B.3. Theorem 3.3

**Theorem 3.3.** *If $\mathfrak{X}$ is uncountable, then there exist functions $f : 2^{\mathfrak{X}} \to \mathbb{R}$ which are not sum-decomposable. Note that this holds even if the sum-decomposition $(\rho, \phi)$ is allowed to be discontinuous.*

*Proof.* Consider $f(X) = \sup(X)$.

As discussed above, a sum over uncountably many elements can converge only if countably many elements are non-zero. But as in the proof of Theorem 3.1, $\phi(x) \neq 0$ for any $x$. So it is immediate that sum-decomposition is not possible for functions operating on uncountable subsets of $\mathfrak{X}$.

Even restricting to countable subsets is not enough. As in the proof of Theorem 3.1, we must have that for each $n \in \mathbb{N}$, $\phi(x) > \frac{1}{n}$ for only finitely many $x$. But then if this is the case, let $\mathfrak{X}_n$ be the set of all $x \in \mathfrak{X}$ with $\phi(x) > \frac{1}{n}$. Since $\phi(x) \neq 0$, we know that $\mathfrak{X} = \bigcup \mathfrak{X}_n$. But this is a countable union of finite sets, which is impossible because $\mathfrak{X}$ is uncountable.

□

### B.4. Theorem 4.3

**Theorem 4.3** (Fixed set size)**.** *Let $f : \mathbb{R}^M \to \mathbb{R}$ be continuous. Then $f$ is permutation-invariant if and only if it is continuously sum-decomposable via $\mathbb{R}^M$.*

*Proof.* The reverse implication is clear. The proof relies on demonstrating that the function $\Phi \to \mathbb{R}^{M+1}$ defined as follows is a homeomorphism onto its image:

$$\Phi_q(X) = \sum_{m=1}^{M} \phi_q(x_m), \quad q = 0, \ldots, M$$
$$\phi_q(x) = x^q, \quad q = 0, \ldots, M$$

Now define $\widetilde{\Phi} \to \mathbb{R}^M$ by:

$$\widetilde{\Phi}_q(X) = \sum_{m=1}^{M} \widetilde{\phi}_q(x_m), \quad q = 1, \ldots, M$$
$$\widetilde{\phi}_q(x) = x^q, \quad q = 1, \ldots, M$$

Note that $\Phi_0(X) = M$ for all $X$, so $\text{Im}(\Phi) = \{M\} \times \text{Im}(\widetilde{\Phi})$. Since $\{M\}$ is a singleton, these two images are homeomorphic, with a homeomorphism given by:

$$\gamma : \text{Im}(\widetilde{\Phi}) \to \text{Im}(\Phi)$$
$$\gamma(x_1, \ldots, x_M) = (M, x_1, \ldots, x_M)$$

Now by definition, $\widetilde{\Phi} = \gamma^{-1} \circ \Phi$. Since this is a composition of homeomorphisms, $\widetilde{\Phi}$ is also a homeomorphism. Therefore $(f \circ \widetilde{\Phi}^{-1}, \widetilde{\phi})$ is a continuous sum-decomposition of $f$ via $\mathbb{R}^M$. □

### B.5. Theorem 4.4

**Theorem 4.4** (Variable set size)**.** *Let $f : \mathbb{R}^{\leq M} \to \mathbb{R}$ be continuous. Then $f$ is permutation-invariant if and only if it is continuously sum-decomposable via $\mathbb{R}^M$.*

*Proof.* We use the adapted sum-of-power mapping $\widetilde{\Phi}$ from above, denoted in this section by $\Phi$.

$$\Phi_q(X) = \sum_{m=1}^{M} \phi_q(x_m), \quad q = 1, \ldots, M$$
$$\phi_q(x_m) = (x_m)^q, \quad q = 1, \ldots, M$$

which is shown above to be injective. Without loss of generality, let $\mathfrak{X} = [0, 1]$ as in Theorem 2.9.

We separate $\Phi_q(X)$ into two terms:

$$\Phi_q(X) = \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(x_m) \tag{9}$$

For an input set $X$ with $M' = M - P$ elements and $0 \leq M', P \leq M$, we say that the set contains $M'$ "actual elements" as well as $P$ "empty" elements which are not in fact part of the input set. Those $P$ "empty elements" can

be regarded as place fillers when the size of the input set is smaller than $M$, i.e. $M' < M$.

We map those $P$ elements to a constant value $k \notin \mathfrak{X}$, preserving the injectiveness of $\Phi_q(X)$ for input sets $X$ of arbitrary size $M'$:

$$\Phi_q(X) = \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(k) \tag{10}$$

Equation (10) is no longer strictly speaking a sum-decomposition. This can be overcome by re-arranging it:

$$\begin{aligned}\Phi_q(X) &= \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(k) \\ &= \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=1}^{M} \phi_q(k) - \sum_{m=1}^{M'} \phi_q(k) \\ &= \sum_{m=1}^{M'} [\phi_q(x_m) - \phi_q(k)] + \sum_{m=1}^{M} \phi_q(k)\end{aligned} \tag{11}$$

The last term in Equation (11) is a constant value which only depends on the choice of $k$ and is independent of $X$ and $M'$. Hence, we can replace $\phi_q(x)$ by $\widehat{\phi_q}(x) = \phi_q(x) - \phi_q(k)$. This leads to a new sum-of-power mapping $\widehat{\Phi}_q(X)$ with:

$$\begin{aligned}\widehat{\Phi}_q(X) &= \sum_{m=1}^{M'} \widehat{\phi}_q(x_m) \\ &= \Phi_q(X) - M \cdot \phi_q(k)\end{aligned} \tag{12}$$

$\widehat{\Phi}$ is injective since $\Phi$ is injective, $k \notin \mathfrak{X}$, and the last term in the above sum is constant. $\widehat{\Phi}$ is also in the form of a sum-decomposition.

For each $m < M$, we can follow the reasoning used in the rest of the proof of Theorem 2.9 to note that $\widehat{\Phi}$ is a homeomorphism when restricted to sets of size $m$ – we denote these restricted functions by $\widehat{\Phi}_m$. Now each $\widehat{\Phi}_m^{-1}$ is a continuous function into $\mathbb{R}^m$. We can associate with each a continuous function $\widehat{\Phi}_{m,M}^{-1}$ which maps into $\mathbb{R}^M$, with the $M - m$ tailing dimensions filled with the value $k$.

Now the domains of the $\widehat{\Phi}_{m,M}^{-1}$ are compact and disjoint since $k \notin \mathfrak{X}$. We can therefore find a function $\widehat{\Phi}_C^{-1}$ which is continuous on $\mathbb{R}^N$ and agrees with each $\widehat{\Phi}_{m,M}^{-1}$ on its domain.

To complete the proof, let $\mathcal{Y}$ be a connected compact set with $k \in \mathcal{Y}, \mathfrak{X} \subset \mathcal{Y}$. Let $\widehat{f}$ be a function on subsets of $\mathcal{Y}$ of size exactly $M$ satisfying:

$$\widehat{f}(X) = f(X); \quad X \subset \mathfrak{X}$$
$$\widehat{f}(X) = f(X \cap \mathfrak{X}); \quad X \subset \mathfrak{X} \cup \{k\}$$

We can choose $\widehat{f}$ to be continuous under the notion of continuity in Appendix A.2. Then $(\widehat{f} \circ \widehat{\Phi}_C^{-1}, \widehat{\phi})$ is a continuous sum-decomposition of $f$.

□

### B.6. Max-Decomposition

Analogously to sum-decomposition, we define the notion of *max-decomposition*. A function $f$ is max-decomposable if there are functions $\rho$ and $\phi$ such that:

$$f(\mathbf{x}) = \rho\big(\max_i(\phi(x_i))\big).$$

where the max is taken over each dimension independently in the latent space. Our definitions of decomposability via $Z$ and continuous decomposability also extend to the notion of max-decomposition.

We now state and prove a theorem which is closely related to Theorem 4.1, but which establishes limitations on max-decomposition, rather than sum-decomposition.

**Theorem B.1.** *Let $M > N \in \mathbb{N}$. Then there exist permutation invariant continuous functions $f : \mathbb{R}^M \to \mathbb{R}$ which are not max-decomposable via $\mathbb{R}^N$.*

Note that this theorem rules out any max-decomposition, whether continuous or discontinuous. We specifically demonstrate that summation is not max-decomposable – as with Theorem 4.1, this theorem applies to ordinary well-behaved functions.

*Proof.* Consider $f(\mathbf{x}) = \sum_{i=1}^{M} x_m$. Let $\phi : \mathbb{R} \to \mathbb{R}^N$, and let $\mathbf{x} \in \mathbb{R}^M$ such that $x_i \neq x_j$ when $i \neq j$.

For $n = 1, \ldots, N$, let $\mu(n) \in \{1, \ldots, M\}$ such that:

$$\max_i(\phi(x_i)_n) = \phi(x_{\mu(n)})_n$$

That is, $\phi(x_{\mu(q)})$ attains the maximal value in the $q$-th dimension of the latent space among all $\phi(x_i)$. Now since $N < M$, there is some $m \in \{1, \ldots, M\}$ such that $\mu(n) \neq m$ for any $n \in \{1, \ldots, N\}$. So now consider $\widetilde{\mathbf{x}}$ defined by:

$$\widetilde{x}_i = x_i; i \neq m \tag{13}$$
$$\widetilde{x}_m = x_{\mu(1)} \tag{14}$$

Then:

$$\max_i(\phi(x_i)) = \max_i(\phi(\widetilde{x}_i))$$

But since we chose $\mathbf{x}$ such that all $x_i$ were distinct, we have $\sum_{i=1}^{M} x_i \neq \sum_{i=1}^{M} \widetilde{x}_i$ by the definition of $\widetilde{\mathbf{x}}$. This shows that $\phi$ cannot form part of a max-decomposition for $f$. But $\phi$ was arbitrary, so no max-decomposition exists.

□

## C. A Continuous Function on $\mathbb{Q}$

This section defines and analyses the function $\Psi$ shown in Figure 2, which is continuous on $\mathbb{Q}$ but not on $\mathbb{R}$. $\Psi$ is defined as the pointwise limit of a sequence of functions $\Psi_n$, illustrated in Figure 4. We proceed as follows:

1. Define a sequence of functions $\widetilde{\Psi}_n$ on $[0, 1]$.
2. Show that the pointwise limit $\widetilde{\Psi}$ is continuous except at points of the form $k \cdot 2^{-m}$ for some integers $k$ and $m$, i.e. except at the *dyadic rationals*.
3. Define the function $\Psi$ on $[0, A]$ by $\Psi(x) = \widetilde{\Psi}(\frac{x}{A})$.
4. Note that $\Psi$ is continuous except at points of the form $A \cdot k \cdot 2^{-m}$ for some integers $k$ and $m$.
5. Choose $A$ to be irrational, so that all points of discontinuity are also irrational, to obtain a function which is continuous on $\mathbb{Q}$. (In all figures, we have chosen $A = \log(4)$).

Informally, we set $\widetilde{\Psi_0}(x) = x$, and at iteration $n$, we split the unit interval into $2^n$ even subintervals. In every even-numbered subinterval, we reflect the function horizontally around the midpoint of the subinterval. We may write this formally as follows.

Let $x \in [0, 1], n \in \mathbb{N}$. Let:

$$a_n(x) = \frac{\lceil x \cdot 2^n \rceil + \frac{1}{2}}{2^n}$$

That is, $a_n(x)$ is the midpoint of the unique half-open interval containing $x$:

$$\left(k \cdot 2^{-n}, (k+1) \cdot 2^{-n}\right]; \quad k \in \mathbb{N}$$

Write $b_n(x)$ for the $n$-th digit in the binary expansion of $x$, and write $c_n(x)$ for the number of $b_m(x), m \leq n$ with $b_m(x) = 1$.

Importantly, $b_n(x)$ is ambiguous if $x$ is a dyadic rational, since in this case $x$ has both a terminating and a non-terminating expansion. For consistency with our choice of the upward-closed interval for the definition of $a_n(x)$, we choose the non-terminating expansion in this case.

Then:

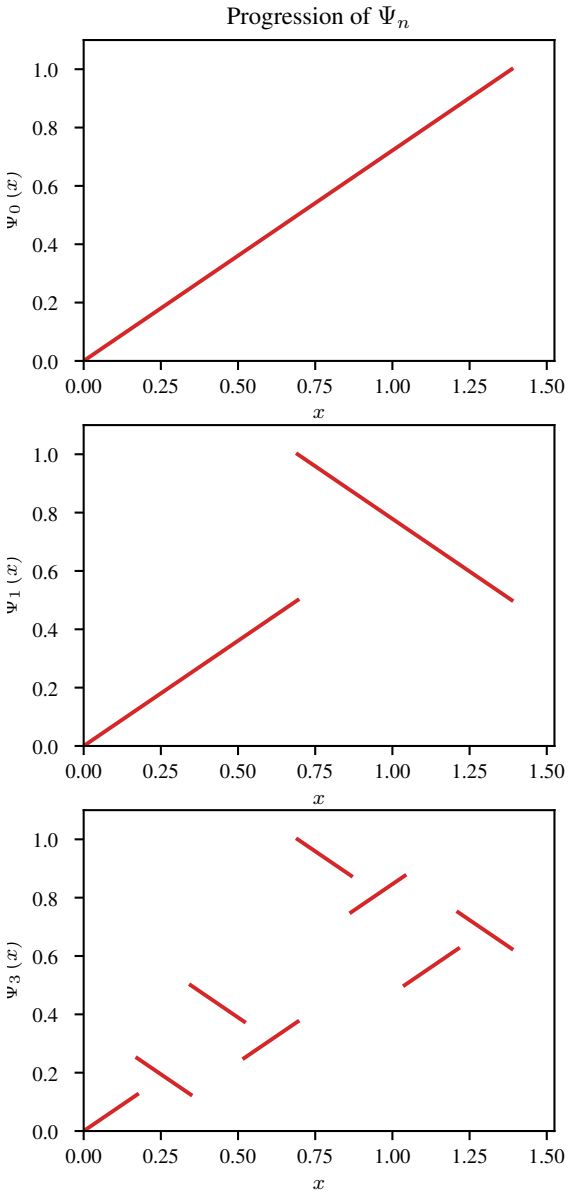



Figure 4: Several iterations of $\Psi_n$

$$\widetilde{\Psi}_n(x) = x + \sum_{i=1}^{n} (-1)^{c_i(x)} \cdot b_i(x) \cdot 2(x - a_i(x))$$
$$\widetilde{\Psi}(x) = x + \sum_{i=1}^{\infty} (-1)^{c_i(x)} \cdot b_i(x) \cdot 2(x - a_i(x))$$

First, it is clear that the series for $\widetilde{\Psi}(x)$ converges absolutely at every $x$, since $|2(x - a_i(x))| \leq 2^{-i}$. So this function is well defined. Also note that:

$$|\widetilde{\Psi}_n(x) - \widetilde{\Psi}(x)| \leq 2^{-n} \tag{15}$$

Note further that $\widetilde{\Psi}_n$ is continuous except at points of the form $k \cdot 2^{-n}$, since $a_m$, $b_m$ and $c_m$ are continuous at these points for all $m \leq n$.

Now consider a point $x_*$ which is not a dyadic rational. We wish to show that $\widetilde{\Psi}$ is continuous at $x_*$. So let $\epsilon > 0$. Choose $n$ so that $2^{-n} < \frac{\epsilon}{3}$. Since $x_*$ is not a dyadic rational, $\widetilde{\Psi}_n$ is continuous at $x_*$, i.e. there is some $\delta > 0$ such that $|x_* - x| < \delta \implies |\widetilde{\Psi}_n(x_*) - \widetilde{\Psi}_n(x)| < \frac{\epsilon}{3}$. But now, by Equation (15):

$$|\widetilde{\Psi}(x_*) - \widetilde{\Psi}_n(x_*)| < 2^{-n} < \frac{\epsilon}{3}$$
$$|\widetilde{\Psi}(x) - \widetilde{\Psi}_n(x)| < 2^{-n} < \frac{\epsilon}{3}$$

And so, whenever $|x_* - x| < \delta$, we have:

$$\begin{aligned} |\widetilde{\Psi}(x_*) - \widetilde{\Psi}(x)| &\leq |\widetilde{\Psi}(x_*) - \widetilde{\Psi}_n(x_*)| \\ &\quad + |\widetilde{\Psi}_n(x_*) - \widetilde{\Psi}_n(x)| \\ &\quad + |\widetilde{\Psi}_n(x) - \widetilde{\Psi}(x)| \\ &< \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} \\ |\widetilde{\Psi}(x_*) - \widetilde{\Psi}(x)| &< \epsilon \end{aligned}$$

Thus, $\widetilde{\Psi}$ is continuous at $x_*$.

As noted above, we may now set $\Psi(x) = \widetilde{\Psi}(\frac{x}{A})$ to obtain a function which is continuous at all rational points.

## D. Implementation Details for Illustrative Example

The network setup follows Equation (1) with 3 fully connected layers before the summation (acting on each input independently) and 2 fully connected layers after. Each fully connected layer has 1000 hidden units and is followed by a ReLU non-linearity. However, the third hidden layer, which creates the latent space in which the summation is executed, has a variable dimension $N$. $N$ is varied for each experiment in order to examine the influence of the latent dimension on the performance.

Training was conducted using the ADAM optimizer (Kingma & Ba, 2015) with an initial learning rate of $0.001$ and an exponential decay after each batch of $0.99$. Training was ended after convergence at 500 batches with a batch size of 32. Samples were continuously drawn from the respective distributions. Therefore, there is no notion of training vs. test data or epoch sizes. The results $r^t$, measured as RMSE, were smoothed using exponential smoothing with $\alpha = 0.95$:

$$r^t_{\text{smooth}} = (1 - \alpha) \cdot r^t + \alpha \cdot r^{t-1} \tag{16}$$

The last, smoothed RMSE is extracted from each experiment, averaged over 500 different runs with different seeds and plotted in Figure 3(a). The confidence intervals are calculated assuming a Gaussian distribution. The critical points are extracted by taking the smallest latent dimension which produces an RMSE of less than $10\%$ above the global minimum for this set size.

*Out of distribution samples:* We tested the performance of a trained model (500 inputs, 100 latent dimensions) on out-of-distribution samples to see to what extent the model exploits the statistical properties of the training examples. Below is a list with examples:

- Input: $[1.0, 1.0, \ldots, 1.0]$, output: $0.933$, true label: $1.0$
- Input: 200 times $1.0$ and 300 times $0.0$, output: $0.413$, true label: $0.0$
- Input: $[0.002, 0.004, 0.006, ..., 1.0]$, output: $0.5006$, true label: $0.5000$

The distributions the samples were drawn from during test time were the uniform distribution, a Gaussian and a Gamma distribution. Each sample consisting of 500 values was randomly drawn from one of these distributions. Hence, the first two are very unlikely samples from the provided distributions. The poor performance of the model on these two examples can therefore be taken as an indication that the model does utilize information about the underlying distributions when estimating the median. The third sample is much closer to a realistic sample from, e.g., the uniform distribution (in our case between $0.0$ and $1.0$), which makes it unsurprising that the model performs much better on this task. It is worth noting that the notion of 'likely' examples of uniform distributions is of course an intuitive one. Given that no value is repeated, every specific set of numbers is of course equally likely as long as all numbers lie within the interval of the uniform distribution.

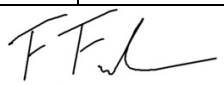
## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

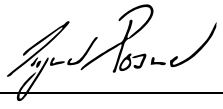
| | |
|---|---|
| Title of Paper | On the Limitations of Representing Functions on Sets |
| Publication Status | x Published □ Accepted for Publication<br>□Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Edward Wagstaff*, Fabian B. Fuchs*, Martin Engelcke*, Ingmar Posner, Michael Osborne<br>*On the Limitations of Representing Functions on Sets*<br>International Conference on Machine Learning (ICML), 2019. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Fabian Fuchs | | |
| Contribution to the Paper | Led the following efforts:<br>- Proposing the project and bringing together the team; managing and coordinating the project<br>- The design, implementation, and analysis of the experiments<br>- Writing a guest blog post on Ferenc Huszar's website about the topic<br>Co-led the following efforts:<br>- Writing the paper<br>Made significant contributions to the following efforts:<br>- Developing the Theory | | |
| Signature |  | Date | 2021/10/06 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Professor Ingmar Posner | | |
| Supervisor comments<br>This is an accurate description of the candidate's contributions. | | |
| Signature  | Date | 8/10/2021 |

This completed form should be included in the thesis, at the end of the relevant chapter.

# 4

# Relational Reasoning and Permutation Invariance in Multi-Object Tracking

# Relational Reasoning and Permutation Invariance in Multi-Object Tracking

**Fabian B. Fuchs, Adam R. Kosiorek, Li Sun, Oiwi Parker Jones, Ingmar Posner**
Applied AI Lab, University of Oxford
{fabian, adamk, kevin, oiwi, ingmar}@robots.ox.ac.uk

## Abstract

Relational reasoning is the ability to model interactions between objects and is critical for a number of cognitive functions and machine learning applications. In this paper, we explore the modelling of relational reasoning in the context of multi-object tracking with a specific focus on permutation invariance. Specifically, we find that permutation-invariant models outperform non-permutation-invariant alternatives. We also find that architectures using a single permutation invariant operation like DeepSets, despite, in theory, being universal function approximators, are nonetheless outperformed by a more complex architecture based on multi-headed attention. Furthermore, we show on three real-world tracking datasets that adding relational reasoning capabilities in our proposed way increases the tracking performance.

## 1 Introduction

The majority of contemporary object-tracking approaches used in autonomous vehicles does not model interactions between objects. This contrasts with the fact that object paths are not independent. For example, a cyclist might abruptly deviate from a previously planned trajectory in order to avoid colliding with a car. To fully leverage the potential of relational reasoning for real-world applications, like multi-object tracking, we propose that a relational-reasoning module ought to access both appearance features and previous motions of all objects involved. In order to provide the relational reasoning module with the maximum richness of appearance and motion information, we built on an end-to-end single object tracker, hierarchical attentive recurrent tracking (HART) [15], and extent it to multi-object tracking (MOHART) with relational reasoning.

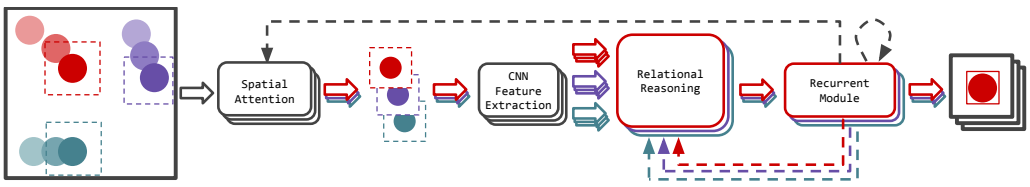



FIGURE 1: Multi-object hierarchical attentive recurrent tracking (MOHART). A glimpse is extracted for each object using a (fully differentiable) spatial attention algorithm. These glimpses are further processed with a CNN and fed into a relational reasoning module. A recurrent module which iterates over time steps allows for capturing of complex motion patterns. The recurrent module also outputs spatial attention parameters and a feature vector per object for the relational reasoning module. Dashed lines indicate temporal connections (from time step $t$ to $t+1$). The entire pipeline operates in parallel for the different objects, only the relational reasoning module allows for exchange of information between tracking states of each object. MOHART is an extension of HART (a single-object tracker), which features the same pipeline sans the relational reasoning module.

Sets and Partitions Workshop at the 33rd Conference on Neural Information Processing Systems, 2019.

Importantly, the entire MOHART system, including the way it models interactions and relations between objects, is class-agnostic and is trained end-to-end (Figure 1). While HART only tracks a single object, MOHART connects multiple HART modules, performing relational reasoning using a self-attention mechanism. Concretely, the relational-reasoning module receives a list of feature vectors, one per object, as input. This part of the problem is *permutation invariant*, as the list-order of object representations carries no meaning for the task at hand.

The paper is organised as follows. In Section 3, we describe a set of methods, including self-attention, for modelling relational reasoning. In Section 4, we show that processing the information in a non-permutation-invariant manner shows no performance improvement compared to single object tracking (i.e. no relational reasoning), despite being theoretically capable of *learning* permutation invariance and having more capacity. In this setting, we demonstrate that the alternative DeepSets architecture [32, 13, 20, 21, 30] is inferior to multi-headed self-attention [28, 16], despite the DeepSets architecture in theory allowing for universal approximation of all permutation invariant functions [32, 30]. In Section 5, we apply MOHART to real world datasets and show that permutation invariant relational reasoning leads to consistent performance improvement compared to HART.

## 2 Related Work

**Visual Object Tracking VOT** In visual object tracking (VOT), a ground-truth bounding box is provided to the algorithm in the first frame and the tracker is evaluated on all future frames. The performance is typically measured as intersection-over-union (IOU) averaged across all frames in which the object is present. On many VOT datasets, SIAMRCNN [29] is currently state-of-the-art (SOTA). In each frame, it employs a pre-trained region proposal network (RPN) providing candidate bounding boxes, which are then compared to the bounding box in the first frame. Previous SOTA models include SIAMRPN++ [17] ECO [4], and DIMP [3]. These models are highly engineered achieving excellent results: They often use RPNs pre-trained on large amounts of data and fine-tune their model online on the target dataset. All of these models are tracking one object at a time (although SIAMRCNN does track distractor objects) and therefore do not perform relational reasoning. They also do not have internal motion models of the objects.

**End-to-End VOT** A newly established and much less explored stream of work approaches region proposal, feature extraction and tracking in an end-to-end fashion with gradients propagated through all parts of the model and across the time axis. This allows for complex motion models as well as efficiency (learning where to look). A key difficulty here is that extracting an image crop (according to bounding-boxes provided by a detector), is non-differentiable and results in high-variance gradient estimators. Kahou et al. [11] propose an end-to-end tracker with soft spatial-attention using a 2D grid of Gaussians instead of a hard bounding-box. HART draws inspiration from this idea, employs an additional attention mechanism, and shows promising performance on the real-world KITTI dataset [15]. HART forms the foundation of this work. It has also been extended to incorporate depth information from RGBD cameras [22]. Gordon et al. [7] propose an approach in which the crop corresponds to the scaled-up previous bounding-box. This simplifies the approach but does not allow the model to learn where to look—i.e. no gradient is backpropagated through crop coordinates. To the best of our knowledge, there are no successful implementations of any such end-to-end approaches for multi-object tracking from vision beyond generative modelling works [14, 27, 10] which work only on relatively simple datasets, and Frossard and Urtasun [6], which relies on costly LIDAR sensors.

**Tracking-by-Detection** Here, traditionally, objects are first detected in each frame independently. A tracking algorithm then links the detections from different frames to propose a coherent trajectory [33, 19, 2, 12]. Often, tracking-by-detection benchmark suites provide external detections for both training and test sets, turning it into a data association task. Recently, some approaches started reusing the same networks for detecting objects and generating re-identification features [35, 34]. MOHART currently does not use external detections (provided or from a private detector) and hence cannot be quantitatively compared to these approaches. However, incorporating external detections via the proposed attention framework is a promising future direction of research.

**Pedestrian trajectory prediction** We draw inspiration from this stream of work for developing a relational reasoning module. Social-LSTM [1] employs a long short-term memory (LSTM) to predict pedestrian trajectories and uses max-pooling to model global social context. Attention

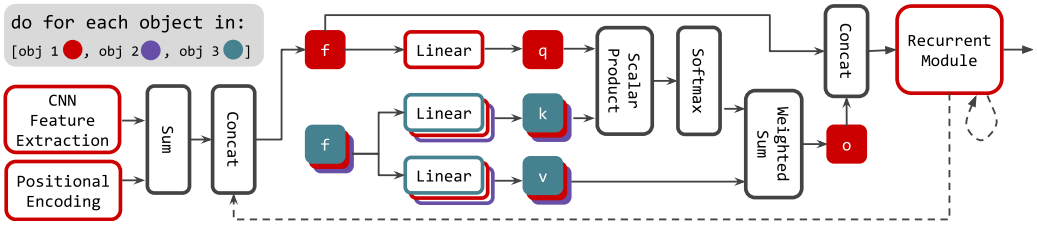



FIGURE 2: The relational reasoning module in MOHART based on multi-headed self-attention. Here, we show the computation of the interaction of the red object with all other objects. Object representations $f_{t,m}$ are computed using visual features, positional encoding and the hidden state from the recurrent module. These are linearly projected onto keys (k), queries (q), and values (v) to compute a weighted sum of interactions between objects, yielding an interaction vector $o_{t,m}$. Subscripts $t, m$ are dropped from all variables for clarity of presentation, so is the splitting into multiple heads.

mechanisms have also been employed to query the most relevant information, such as neighbouring pedestrians, in a learnable fashion [25, 5, 24]. Our work stands apart from this prior art by not relying on ground truth tracklets. It addresses the more challenging task of working directly with visual input, performing tracking, modelling interactions, and, depending on the application scenario, simultaneously predicting future motions.

## 3 Recurrent Multi-Object Tracking with Self-Attention

This section describes the model architecture in Figure 1. We start by describing the hierarchical attentive recurrent tracking (HART) algorithm (Kosiorek et al. [15]), and then follow with an extension of HART to tracking multiple objects, where multiple instances of HART communicate with each other using multi-headed attention to facilitate relational reasoning. We also explain how this method can be extended to trajectory prediction instead of just tracking.

### 3.1 Hierarchical Attentive Recurrent Tracking (HART)

HART is an attention-based recurrent algorithm, which can efficiently track single objects in a video. It uses a spatial attention mechanism to extract a *glimpse* $\mathbf{g}_t$, which corresponds to a small crop of the image $\mathbf{x}_t$ at time-step $t$, containing the object of interest. This allows it to dispense with the processing of the whole image and can significantly decrease the amount of computation required. HART uses a convolutional neural network (CNN) to convert the glimpse $\mathbf{g}_t$ into features $\mathbf{f}_t$, which then update the hidden state $\mathbf{h}_t$ of a LSTM core. The hidden state is used to estimate the current bounding-box $\mathbf{b}_t$, spatial attention parameters for the next time-step $\mathbf{a}_{t+1}$, as well as object appearance. Importantly, the recurrent core can learn to predict complicated motion conditioned on the past history of the tracked object, which leads to relatively small attention glimpses—contrary to CNN-based approaches [9, 26], HART does not need to analyse large regions-of-interest to search for tracked objects. In the original paper, HART processes the glimpse with an additional ventral and dorsal stream. Early experiments have shown that this does not improve performance on the MOTChallenge dataset, presumably due to the oftentimes small objects and overall small amount of training data. Further details are provided in Appendix B.

The algorithm is initialised with a bounding-box[1] $\mathbf{b}_1$ for the first time-step, and operates on a sequence of raw images $\mathbf{x}_{1:T}$. For time-steps $t \geq 2$, it recursively outputs bounding-box estimates for the current time-step and predicted attention parameters for the next time-step. The performance is measured as IOU averaged over all time steps in which an object is present, excluding the first time step.

HART is limited to tracking one object at a time. While it can be deployed on several objects in parallel, different HART instances have no means of communication. This results in performance loss, as it is more difficult to identify occlusions, ego-motion and object interactions. Below, we propose an extension of HART which remedies these shortcomings.

[1] We can use either a ground-truth bounding-box or one provided by an external detector; the only requirement is that it contains the object of interest.

### 3.2 Multi-Object Hierarchical Attentive Recurrent Tracking

Multi-object support in HART requires the following modifications. Firstly, in order to handle a dynamically changing number of objects, we apply HART to multiple objects in parallel, where all parameters between HART instances are shared. We refer to each HART instance as a *tracker*. Secondly, we introduce a presence variable $p_{t,m}$ for object $m$. It is used to mark whether an object should interact with other objects, as well as to mask the loss function (described in Kosiorek et al. [15]) for the given object when it is not present. In this setup, parallel trackers cannot exchange information and are conceptually still single-object trackers, which we use as a baseline, referred to as HART (despite it being an extension of the original algorithm). Finally, to enable communication between trackers, we augment HART with an additional step between feature extraction and the LSTM.

For each object, a glimpse is extracted and processed by a CNN (see Figure 1). Furthermore, spatial attention parameters are linearly projected on a vector of the same size and added to this representation, acting as a positional encoding. This is then concatenated with the hidden state of the recurrent module of the respective object (see Figure 2). Let $\mathbf{f}_{t,m}$ denote the resulting feature vector corresponding to the $\mathrm{m}^{\mathrm{th}}$ object, and let $\mathbf{f}_{t,1:M}$ be the set of such features for all objects. Since different objects can interact with each other, it is necessary to use a method that can inform each object about the effects of their interactions with other objects. Moreover, since features extracted from different objects comprise a set, this method should be permutation-invariant, i. e., the results should not depend on the order in which object features are processed. Therefore, we use the multi-head self-attention block (SAB, Lee et al. [16]), which is able to account for higher-order interactions between set elements when computing their representations. Intuitively, in our case, SAB allows any of the trackers to query other trackers about attributes of their respective objects, e. g., distance between objects, their direction of movement, or their relation to the camera. This is implemented as follows,

$$Q = W_q \mathbf{f}_{1:M} + b_q\,, \qquad K = W_k \mathbf{f}_{1:M} + b_k\,, \qquad V = W_v \mathbf{f}_{1:M} + b_v\,, \tag{1}$$

$$O_i = \mathrm{softmax}\left(Q_i K_i^T / \sqrt{d_q}\right) V_i\,, \qquad i = 1, \ldots, H\,, \tag{2}$$

$$o_{1:M} = O = \mathrm{concat}(O_1, \ldots, O_H)\,, \tag{3}$$

where $o_m$ is the output of the relational reasoning module for object $m$. Time-step subscripts are dropped to decrease clutter. In Eq. 1, each of the extracted features $\mathbf{f}_{t,m}$ is linearly projected into a triplet of key $\mathbf{k}_{t,m}$, query $\mathbf{q}_{t,m}$ and value $\mathbf{v}_{t,m}$ vectors. Together, they comprise $K, Q$ and $V$ matrices with $M$ rows and $d_q, d_q, d_v$ columns, respectively. $K, Q$ and $V$ are then split up into multiple heads $H \in \mathbb{N}_+$, which allows to query different attributes by comparing and aggregating different projection of features. Multiplying $Q_i K_i^T$ in Eq. 2 allows to compare every query vector $\mathbf{q}_{t,m,i}$ to all key vectors $\mathbf{k}_{t,1:M,i}$, where the value of the corresponding dot-products represents the degree of similarity. Similarities are then normalised via a $\mathrm{softmax}$ operation and used to aggregate values $V$. Finally, outputs of different attention heads are concatenated in Eq. 3. SAB produces $M$ output vectors, one for each input, which are then concatenated with corresponding inputs and fed into separate LSTMs for further processing, as in HART—see Figure 1.

We have been deliberately careless about making a precise distinction between invariance and equivariance but rectify this now. A function is *equivariant* if a transformation (in this case permutation) of the input results in an equivalent transformation of the output. A single output vector of the relational reasoning module $o_i$ is *invariant* with respect to permutations of the list of all other objects. In contrast, when considering all outputs at once, i.e. the vector $o_{1:M}$, the relational reasoning module is *equivariant* with respect to a permutation of the inputs. Similarly, the bounding box predictions of the network as a whole are permutation *equivariant*.

MOHART is trained fully end-to-end, contrary to other approaches. It maintains a hidden state, which can contain information about the object's motion. One benefit is that one can simply feed black frames into the model to predict future trajectories. Our experiments show that the model learns to fall back on the motion model captured by the LSTM in this case.

## 4 Validation on Simulated Data

We first evaluate the relational reasoning capabilities of the proposed algorithms on a toy domain—a two-dimensional square box filled with bouncing balls. We train the model to predict future object locations (in contrast to simply tracking), to see how well the models understand motion patterns and interactions between objects. For details about the experimental setup, see Appendix C.

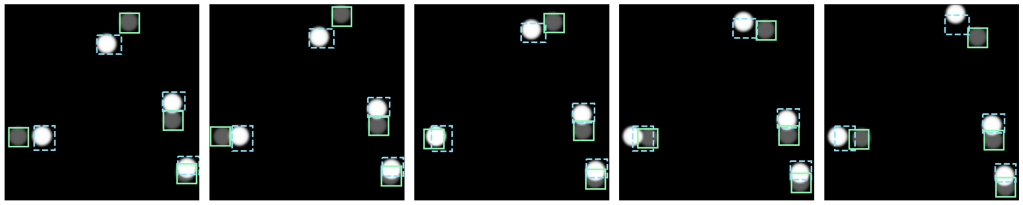


FIGURE 3: HART single object tracking applied four times in parallel and trained to predict the location of each circle three time steps into the future. Dashed lines indicate spatial attention, solid lines are predicted bounding boxes, faded circles show ground truth location at $T + 3$. Each circle exerts repulsive forces on each other, where the force scales with $1/r$, $r$ being their distance.

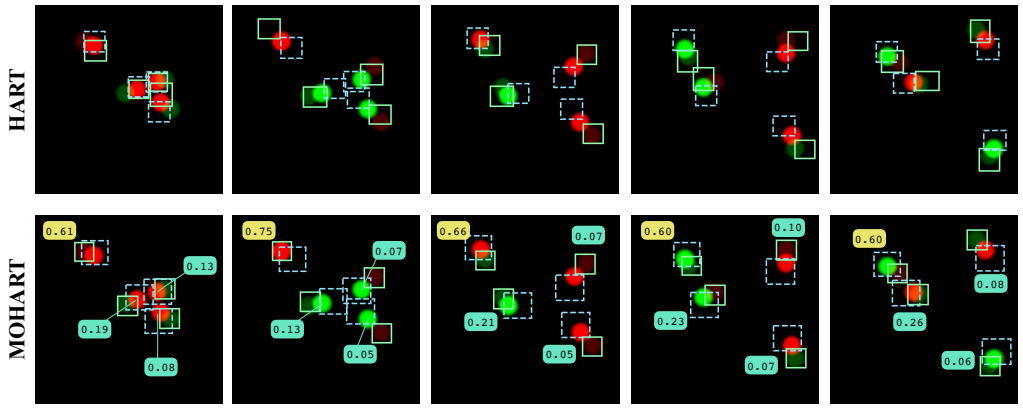



FIGURE 4: A scenario constructed to be impossible to solve without relational reasoning. Circles of the same colour repel each other, circles of different colour attract each other. Crucially, each circle is randomly assigned its identity in each time step. Hence, the algorithm can not infer the forces exerted on one object without knowledge of the state of the other objects in the current time step. The forces in this scenario scale with $1/\sqrt{r}$ and the algorithm was trained to predict one time step into the future. HART (top) is indeed unable to predict the future location of the objects accurately. The achieved average IoU is $47\%$, which is only slightly higher than predicting the objects to have the same position in the next time step as in the current one ($34\%$). Using the relational reasoning module, MOHART (bottom) is able to make meaningful predictions ($76\%$ IoU). The numbers in the bottom row indicate the self-attention weights from the perspective of the top left tracker (yellow number box). Interestingly, the attention scores have a strong correlation with the interaction strength (which scales with distance) without receiving supervision.

### 4.1 First Experiment: Deterministic Domain

In the first experiment in the toy domain (Figure 3), four balls, which can be thought of as 'protons in a box', repel each other. HART is applied four times in parallel and is trained to predict the location of each ball three time steps into the future. Different forces from different objects lead to a non-trivial force field at each time step. Accurately predicting the future location of an object using only its previous motion is therefore challenging (Figure 3 shows that each attention glimpse covers only the current object). Surprisingly, the single object tracker solves this task with an average of $95\%$ IoU over sequences of 15 time steps, which shows the efficacy of end-to-end tracking to capture complex motion patterns and use them to predict future locations. This, of course, could also be used to generate good-quality bounding boxes for a tracking task.

### 4.2 Second Experiment: Stochastic Domain

In the second experiment, we introduce randomness, rendering the scenario not solvable for a single object tracker as it requires knowledge about the state of the other objects and relational reasoning (see Figure 4). In each time step, we assign a colour-coded identity to the objects. Objects of the same identity repel each other, object of different identities attract each other (the objects can be thought of as electrons and protons). We compare our proposed attention-based relational reasoning module to the following baselines:

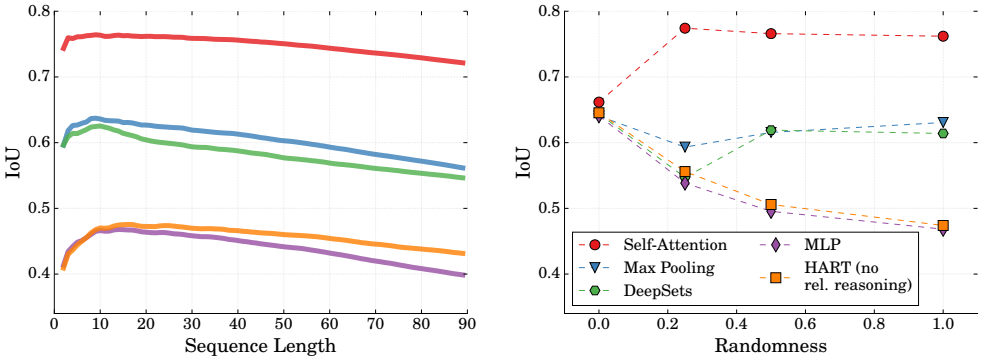



FIGURE 5: Left: average IoU over sequence length for different implementations of relational reasoning on the toy domain shown in Figure 4 (randomness $= 1.0$). Right: performance depends on *randomness*—the frequency with which ball identities are randomly changed (sequence length 15). Higher randomness puts more pressure on relational reasoning. For randomness $= 0$, identities still have to be reassigned in some cases in order to prevent deadlocks, this leads to a performance loss for all models, which explains lower performance of self-attention for randomness $= 0$.

**Multilayer perceptron (MLP)** In this version, the representations of all objects are concatenated and fed into a fully connected layer followed by ELU activations. The output is then again concatenated to the unaltered feature vector of each object. This concatenated version is then fed to the recurrent module of HART. This way of exchanging information allows for universal function approximation (in the limit of infinite layer sizes) but does not impose permutation invariance.

**DeepSets** Here, the learned representations of the different objects are summed up instead of concatenated and then divided by total number of objects. This is closely related to DeepSets (Zaheer et al. [32]) and allows for universal function approximation of all permutation invariant functions (Wagstaff et al. [30]).

**Max-Pooling** Similar to DeepSets, but using max-pooling as the permutation invariant operation. This way of exchanging information is used, e.g., by Alahi et al. [1] who predict future pedestrian trajectories from ground truth tracklets in coordinate space.

Figure 5 (left) shows a quantitative comparison of augmenting HART with different relational reasoning modules when identities are re-assigned in every timestep (randomness $= 1.0$). Exchanging information between different trackers with an MLP leads to slightly worse performance than the baseline, while simple max-pooling performs significantly better ($\Delta$IoU $\sim 17\%$). This can be explained through the permutation invariance of the problem: latent representation of different objects have no meaningful order; therefore the output of the model for a specific object should be invariant to the ordering of the other objects. The MLP is in itself not permutation invariant and therefore prone to overfitting to the (meaningless) order of the objects in the training data. Max-pooling, however, is permutation invariant and can in theory, despite its simplicity, be used to approximate any permutation invariant function given a sufficiently large latent space (Wagstaff et al. [30]). Max-pooling is often used to exchange information between different tracklets, e. g., in the trajectory prediction domain (Alahi et al. [1], Gupta et al. [8]). However, self-attention, allowing for learned querying and encoding of information, solves the relational reasoning task much more accurately.

In Figure 5 (right), the frequency with which object identities are reassigned randomly is varied. The results show that, in a deterministic environment, tracking does not necessarily profit from relational reasoning - even in the presence of long-range interactions. The less random, the more static the force field is and a static force field can be inferred from a small number of observations (see Figure 3). This does of course not mean that all stochastic environments profit from relational reasoning. What these experiments indicate is that tracking can not be expected to profit from relational reasoning by default in any environment, but instead in environments which feature (potentially non-deterministic) dynamics and predictable interactions.

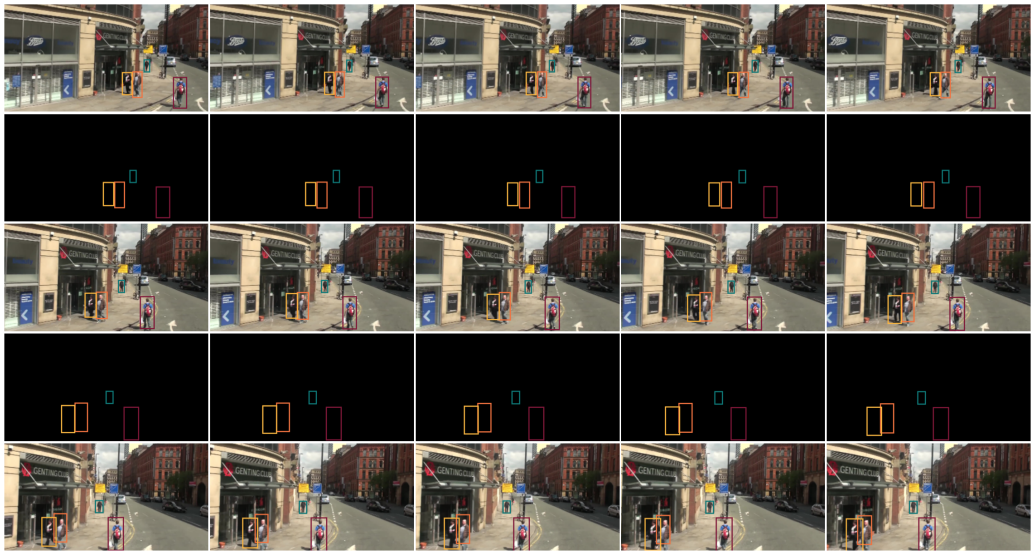


FIGURE 6: Camera blackout experiment on a street scene from the MOTChallenge dataset with strong ego-motion. Solid boxes are MOHART predictions (for $t \geq 2$), faded bounding boxes indicate object locations in the first frame. As the model is trained end-to-end, MOHART learns to fall back onto its internal motion model if no new observations are available (black frames). As soon as new observations come in, the model 'snaps' back onto the tracked objects.

## 5 Relational Reasoning in Real-World Tracking

Having established that MOHART is capable of performing complex relational reasoning, we now test the algorithm on three real-world datasets and analyse the effects of relational reasoning on performance depending on dataset and task. We find consistent improvements of MOHART compared to HART throughout. Relational reasoning yields particularly high gains for scenes with ego-motion, crowded scenes, and simulated faulty sensor inputs.

### 5.1 Experimental Details

We investigate three qualitatively different datasets: the MOTChallenge dataset (Milan et al. [18]), the UA-DETRAC dataset (Wen et al. [31]), and the Stanford Drone dataset (Robicquet et al. [23]). To increase scene dynamics and make the tracking/prediction problems more challenging, we sub-sample some of the high framerate scenes with a stride of two, resulting in scenes with 7-15 frames per second. Training and architecture details are given in Appendices A and B. We conduct experiments in three different modes:

**Tracking.** The model is initialised with the ground truth bounding boxes for a set of objects in the first frame. It then consecutively sees the following frames and predicts the bounding boxes. The sequence length is 30 time steps and the performance is measured as intersection-over-union (IOU) averaged over the entire sequence excluding the first frame. This algorithm is either applied to the entire dataset or subsets of it to study the influence of certain properties of the data.

**Camera Blackout.** This simulates unsteady or faulty sensor inputs. The setup is the same as in *Tracking*, but sub-sequences of the input are replaced with black images. The algorithm is expected to recognise that no new information is available and that it should resort to its internal motion model.

**Prediction.** Testing MOHART's ability to capture motion patterns, only two frames are shown to the model followed by three black frames. IoU is measured separately for each time step.

### 5.2 Results and Analysis

On the MOTChallenge dataset, HART achieves $66.6\%$ IOU (see Table 1), which in itself is impressive given the small amount of training data of only 5225 training frames and no pre-training. MOHART

TABLE 1: Tracking performance on the MOTChallenge dataset measured in IoU.

| | Entire Dataset | Only Ego-Motion | No Ego-Motion | Crowded Scenes | Camera Blackout |
|---|---|---|---|---|---|
| **MOHART** | **68.5%** | **66.9%** | **64.7%** | **69.1%** | **63.6%** |
| HART | 66.6% | 64.0% | 62.9% | 66.9% | 60.6% |
| $\Delta$ | 1.9% | 2.9% | 1.8% | 2.2% | 3.0% |

TABLE 2: UA-DETRAC Dataset

| | All | Crowded Scenes | Camera Blackout |
|---|---|---|---|
| **MOHART** | 68.1% | **69.5%** | **64.2%** |
| HART | **68.4%** | 68.6% | 53.8% |
| $\Delta$ | -0.3% | 0.9% | 0.4% |

TABLE 3: Stanford Drone Data

| | All | Camera Blackout | CamBlack Bikes |
|---|---|---|---|
| **MOHART** | **57.3%** | **53.3%** | **53.3%** |
| HART | 56.1% | 52.6% | 50.7% |
| $\Delta$ | 1.2% | 0.7% | 2.6% |

achieves $68.5\%$ (both numbers are averaged over 5 runs, independent samples $t$-test resulted in $p < 0.0001$). The performance gain increases when only considering ego-motion data. This is readily explained: movements of objects in the image space due to ego-motion are correlated and can therefore be better understood when combining information from movements of multiple objects, i.e. performing relational reasoning. In another ablation, we filtered for only crowded scenes by requesting five objects to be present for, on average, 90% of the frames in a sub-sequence. For the MOT-Challenge dataset, this only leads to a minor increase of the performance gain of MOHART indicating that the dataset exhibits a sufficient density of objects to learn interactions. The biggest benefit from relational reasoning can be observed in the *camera blackout* experiments (setup explained in Section 5.1). Both HART and MOHART learn to rely on their internal motion models when confronted with black frames and propagate the bounding boxes according to the previous movement of the objects. It is unsurprising that this scenario profits particularly from relational reasoning. Qualitative tracking and *camera blackout* results are shown in Figure 6 and Appendix E.

Tracking performance on the UA-DETRAC dataset only profits from relational reasoning when filtering for crowded scenes (see Table 2). The fact that the performance of MOHART is slightly worse on the vanilla dataset ($\Delta = -0.3\%$) can be explained with more overfitting. As there is no exchange between trackers for each object, each object constitutes an independent training sample.

The Stanford drone dataset (see Table 3) is different to the other two—it is filmed from a birds-eye view. The scenes are more crowded and each object covers a small number of pixels, rendering it a difficult problem for tracking. The dataset was designed for trajectory prediction—a setup where an algorithm is typically provided with ground-truth tracklets in coordinate space and potentially an image as context information. The task is then to extrapolate these tracklets into the future. The performance gain of MOHART on the *camera blackout* experiments is particularly strong when only considering cyclists.

In the *prediction* experiments (see Appendix D), MOHART consistently outperforms HART. On both datasets, the model outperforms a baseline which uses momentum to linearly extrapolate the bounding boxes from the first two frames. This shows that even from just two frames, the model learns to capture motion models which are more complex than what could be observed from just the bounding boxes (i.e. momentum), suggesting that it uses visual information (HART & MOHART) as well as relational reasoning (MOHART).

## Acknowledgements

We thank Stefan Saftescu for his contributions, particularly for integrating the Stanford Drone Dataset. We thank Adam Golinski as well as Stefan Saftescu for proof-reading. This research was funded by the EPSRC AIMS Centre for Doctoral Training at Oxford University and an EPSRC Programme Grant (EP/M019918/1). We acknowledge use of Hartree Centre resources in this work. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. The Centre aims to develop and demonstrate next generation software, optimised to take advantage of the move towards exa-scale computing.

## References


[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2016.

[2] Seung-Hwan Bae and Kuk-Jin Yoon. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[3] G. Bhat, M. Danelljan, L. Van Gool, and R. Timofte. Learning discriminative model prediction for tracking. *International Conference on Computer Vision (ICCV)*, 2019.

[4] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *Neural networks*, 2018.

[6] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3d tracking by detection. *ICRA*, 2018.

[7] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re3 : Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects. *IEEE Robotics and Automation Letters*, 2018.

[8] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: socially acceptable trajectories with generative adversarial networks. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2018.

[9] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[10] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations (ICLR)*, 2020.

[11] Samira Ebrahimi Kahou, Vincent Michalski, and Roland Memisevic. RATM: recurrent attentive tracking model. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2017.

[12] Margret Keuper, Siyu Tang, Bjorn Andres, Thomas Brox, and Bernt Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[13] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow networks: deep sets for particle jets. *Journal of High Energy Physics*, 2019(1), Jan 2019.

[14] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8606–8616, 2018.

[15] Adam R. Kosiorek, Alex Bewley, and Ingmar Posner. Hierarchical attentive recurrent tracking. *Neural Information Processing Systems (NeurIPS)*, 2017.

[16] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer. In *International Conference on Machine Learning (ICML)*, 2019.

[17] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv*, 2016.

[19] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[22] Maryam Rasoulidanesh, Srishti Yadav, Sachini Herath, Yasaman Vaghei, and Shahram Payandeh. Deep attention models for human tracking using rgbd. *Sensors*, 19:750, 02 2019.

[23] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savaresei. Learning social etiquette: Human trajectory prediction in crowded scenes. *European Conference on Computer Vision (ECCV)*, 2016.

[24] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2019.

[25] Hang Su, Yinpeng Dong, Jun Zhu, Haibin Ling, and Bo Zhang. Crowd scene understanding with coherent recurrent neural networks. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2016.

[26] Jack Valmadre, Luca Bertinetto, João F. Henriques, Andrea Vedaldi, and Philip Hilaire Sean Torr. End-to-end representation learning for correlation filter based tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] 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.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems (NeurIPS)*, 2017.

[29] Paul Voigtlaender, Jonathon Luiten, Philip Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. *International Conference on Machine Learning (ICML)*, 2019.

[31] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. DETRAC: A new benchmark and protocol for multi-object tracking. *arXiv*, 2015.

[32] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[33] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2008.

[34] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wen-Jun Zeng, and Wen-Yu Liu. A simple baseline for multi-object tracking. *ArXiv*, 2020.

[35] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *arXiv*, 2020.

## A Experimental Details

The MOTChallenge and the UA-DETRAC dataset discussed in this section are intended to be used as a benchmark suite for multi-object-tracking in a tracking-by-detection paradigm. Therefore, ground truth bounding boxes are only available for the training datasets. The user is encouraged to upload their model which performs tracking in a data association paradigm leveraging the provided bounding box proposals from an external object detector. As we are interested in a different analysis (IoU given inital bounding boxes), we divide the training data further into training and test sequences. To make up for the smaller training data, we extend the MOTChallenge 2017 dataset with three sequences from the 2015 dataset (ETH-Sunnyday, PETS09-S2L1, ETH-Bahnhof). We use the first 70% of the frames of each of the ten sequences for training and the rest for testing. Sequences with high frame rates (30Hz) are sub-sampled with a stride of two. For the UA-DETRAC dataset, we split the 60 available sequences into 44 training sequences and 16 test sequences. For the considerably larger Stanford Drone dataset we took three videos of the scene *deathCircle* for training and the remaining two videos from the same scene for testing. The videos of the drone dataset were also sub-sampled with a stride of two to increase scene dynamics.

## B Architecture Details

The architecture details were chosen to optimise HART performance on the MOTChallenge dataset. They deviate from the original HART implementation (Kosiorek et al. [15]) as follows: A presence variable predicts whether an object is in the scene and successfully tracked. This is trained with a binary cross entropy loss. The maximum number of objects to be tracked simultaneously was set to 5 for the UA-DETRAC and MOTChallenge dataset. For the more crowded Stanford drone dataset, this number was set to 10. The feature extractor is a three layer convolutional network with a kernel size of 5, a stride of 2 in the first and last layer, 32 channels in the first two layers, 64 channels in the last layer, ELU activations, and skip connections. This converts the initial $32 \times 32 \times 3$ glimpse into a $7 \times 7 \times 64$ feature representation. This is followed by a fully connected layer with a 128 dimensional output and an elu activation. The spatial attention parameters are linearly projected onto 128 dimensions and added to this feature representation serving as a positional encoding. The LSTM has a hidden state size of 128. The self-attention unit in MOHART comprises linear projects the inputs to dimensionality 128 for each keys, queries and values. For the real-world experiments, in addition to the extracted features from the glimpse, the hidden states from the previous LSTM state are also fed as an input by concatinating them with the features. In all cases, the output of the attention module is concatenated to the input features of the respective object.

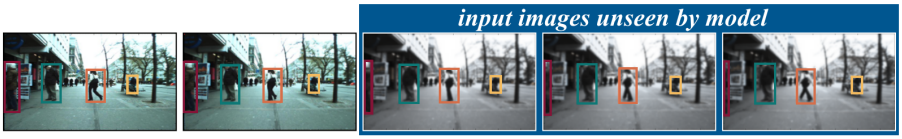
As an optimizer, we used RMSProp with momentum set to $0.9$ and learning rate $5 * 10^{-6}$. For the MOTChallenge dataset and the UA-DETRAC dataset, the models were trained for 100,000 iterations of batch size 10 and the reported IoU is exponentially smoothed over iterations to achieve lower variance. For the Stanford Drone dataset, the batch size was increased to 32, reducing time to convergence and hence model training to 50,000 iterations.

## C Toy Domain

The toy domain consists of a square two-dimensional box filled with bouncing balls. The balls move and can collide with each other with approximated elastic collisions (energy and momentum conservation). Additionally, balls exert either repulsive force (first experiment) or repulsive/attractive force (second experiment, colour-coded), which scales with $1/r$, $r$ being the distance between centres of the balls.

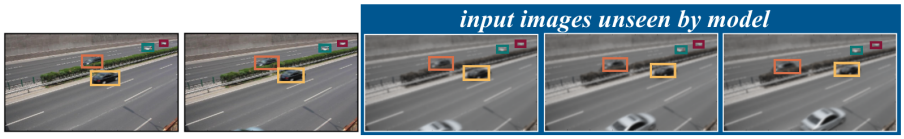
## D Prediction Experiments

In the results from the *prediction* experiments (see Figure 7) MOHART consistently outperforms HART. On both datasets, the model outperforms a baseline which uses momentum to linearly extrapolate the bounding boxes from the first two frames. This shows that even from just two frames, the model learns to capture motion models which are more complex than what could be observed from just the bounding boxes (i.e. momentum), suggesting that it uses visual information (HART & MOHART) as well as relational reasoning (MOHART). The strong performance gain of MOHART compared to




| | | | | | |
|---|---|---|---|---|---|
| **MOHART** | – | **86.4%** | **80.0%** | **74.0%** | **68.4%** |
| HART | – | 85.3% | 79.2% | 73.1% | 67.4% |
| Momentum | – | – | 78.2% | 70.9% | 63.9% |

(A) Prediction results on the MOTChallenge dataset [18].




| | | | | | |
|---|---|---|---|---|---|
| **MOHART** | – | **87.8%** | **81.3%** | **75.5%** | **70.0%** |
| HART | – | 86.6% | 79.6% | 72.6% | 66.1% |
| Momentum | – | - | 80.5% | 73.6% | 67.2% |

(B) Prediction results on the UA-DETRAC dataset (crowded scenes only) [31].

FIGURE 7: Peeking into the future. Only the first two frames are shown to the tracking algorithm followed by three black frames. MOHART learns to fall back on its internal motion model when no observation (i.e. only a black frame) is available. The reported IoU scores show the performance for the respective frames 0, 1, 2, and 3 time steps into the future.

HART on the UA-DETRAC dataset, despite the small differences for tracking on this dataset, can be explained as follows: this dataset features little interactions but strong correlations in motion. Hence when only having access to the first two frames, MOHART profits from estimating the velocities of multiple cars simultaneously.

## E Qualitative Tracking Results

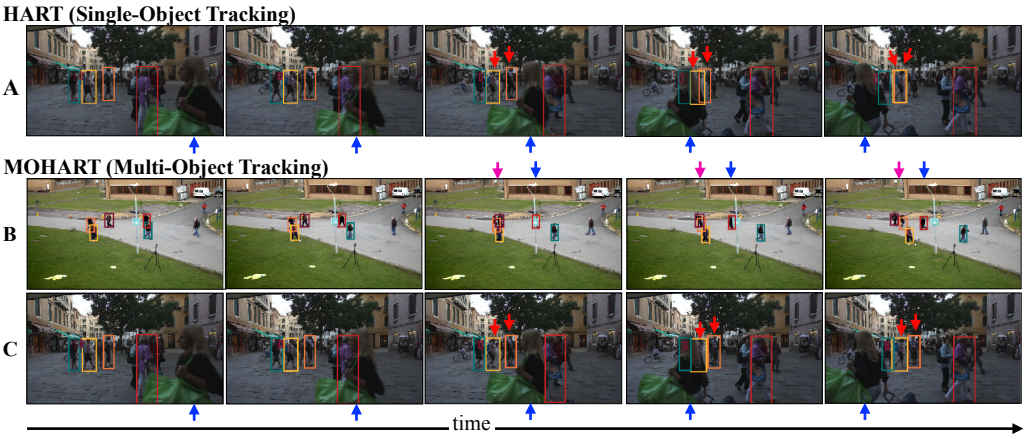



FIGURE 8: Tracking examples of both HART and MOHART. Coloured boxes are bounding boxes predicted by the model, arrows point at challenging aspects of the scenes. (A) & (C): Each person being tracked is temporarily occluded by a woman walking across the scene (blue arrows). MOHART, which includes a relational reasoning module, handles this more robustly (compare red arrows).

In Section 5, we tested MOHART on three different real world data sets and in a number of different setups. Figure 8 shows qualitative results both for HART and MOHART on the MOTChallenge dataset.

Furthermore, we conducted a set of camera blackout experiments to test MOHART's capability of dealing with faulty sensor inputs. While traditional pipeline methods require careful consideration of different types of corner cases to properly handle erroneous sensor inputs, MOHART is able to capture these automatically, especially when confronted with similar issues in the training scenarios. To simulate this, we replace subsequences of the images with black frames. Figure 9 and Figure 6

FIGURE 9: Camera blackout experiment on a pedestrian street scene from the MOTChallenge dataset without ego-motion. Subsequent frames are displayed going from top left to bottom right. Shown are the inputs to the model (some of them being black frames, i.e. arrays of zeroes) and bounding boxes predicted by MOHART (coloured boxes). This scene is particularly challenging as occlusion and missing sensor input coincide (fourth row).

show two such examples from the test data together with the model's prediction. MOHART learns not to update its internal model when confronted with black frames and instead uses the LSTM to propagate the bounding boxes. When proper sensor input is available again, the model uses this to make a rapid adjustment to its predicted location and 'snap' back onto the object. This works remarkably well in both the presence of occlusion (Figure 9) and ego-motion (Figure 6). Tables 1 to 3 show that the benefit of relational reasoning is particularly high in these scenarios specifically. These experiments can also be seen as a proof of concept of MOHART's capabalities of predicting future trajectories—and how this profits from relational reasoning.

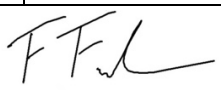
## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

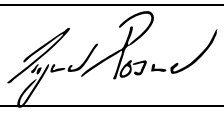
| | |
|---|---|
| Title of Paper | Relational Reasoning and Permutation Invariance in Multi-Object Tracking |
| Publication Status | (x) Published (Workshop only) □ Accepted for Publication<br>□Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Fabian B. Fuchs, Adam R. Kosiorek, Li Sun, Oiwi Parker Jones, Ingmar Posner<br>*Permutation Invariance and Relational Reasoning in Multi-Object Tracking*<br>Sets and Partitions Workshop at NeurIPS, 2019. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Fabian Fuchs | | |
| Contribution to the Paper | Led the following efforts:<br>- Designing and implementing the phsyical toy simulations<br>- Designing, running, and analysing experiments<br>- Writing the paper<br>- Making and presenting the poster<br>Co-led the following efforts:<br>- Altering the HART algorithm to track multiple objects while performing relational reasoning | | |
| Signature |  | Date | 2021/10/01 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Professor Ingmar Posner | | |
| Supervisor comments<br>This is an accurate description of the candidate's contributions. | | |
| Signature  | Date | 8/10/2021 |

This completed form should be included in the thesis, at the end of the relevant chapter.

# 5

# SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks

# SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks

**Fabian B. Fuchs**[*†]
Bosch Center for Artificial Intelligence
A2I Lab, Oxford University
fabian@robots.ox.ac.uk

**Daniel E. Worrall**[*]
Amsterdam Machine Learning Lab, Philips Lab
University of Amsterdam
d.e.worrall@uva.nl

**Volker Fischer**
Bosch Center for Artificial Intelligence
volker.fischer@de.bosch.com

**Max Welling**
Amsterdam Machine Learning Lab
University of Amsterdam
m.welling@uva.nl

## Abstract

We introduce the SE(3)-Transformer, a variant of the self-attention module for 3D point clouds, which is *equivariant* under continuous 3D roto-translations. Equivariance is important to ensure stable and predictable performance in the presence of nuisance transformations of the data input. A positive corollary of equivariance is increased weight-tying within the model, leading to fewer trainable parameters and thus decreased sample complexity (i.e. we need less training data). The SE(3)-Transformer leverages the benefits of self-attention to operate on large point clouds with varying number of points, while guaranteeing SE(3)-equivariance for robustness. We evaluate our model on a toy $N$-body particle simulation dataset, showcasing the robustness of the predictions under rotations of the input. We further achieve competitive performance on two real-world datasets, ScanObjectNN and QM9. In all cases, our model outperforms a strong, non-equivariant attention baseline and an equivariant model without attention.

## 1 Introduction

Self-attention mechanisms [28] have enjoyed a sharp rise in popularity in the last few years. Their relative implementational simplicity coupled with high efficacy on a wide range of tasks such as language modeling [28], image recognition [16], or graph-based problems [29], make them an attractive component to use. However, their generality of application means that for specific tasks, knowledge of existing underlying structure is unused. In this paper, we propose the *SE(3)-Transformer* shown in Fig. 1, a self-attention mechanism specifically for 3D point cloud data, which adheres to *equivariance constraints*, improving robustness to nuisance transformations and general performance.

Point cloud data is ubiquitous across many fields, presenting itself in diverse forms such as 3D object scans [26], 3D molecular structures [19], or $N$-body particle simulations [12]. Finding neural structures which can adapt to the varying number of points in an input, while respecting the irregular sampling of point positions, is challenging. Furthermore, an important property is that these structures should be invariant to global changes in overall input pose; that is, 3D translations and rotations of the input point cloud should not affect the output. In this paper, we find that the explicit imposition of equivariance constraints on the self-attention mechanism addresses these challenges. The SE(3)-Transformer uses the self-attention mechanism as a data-dependent filter particularly suited for sparse, non-voxelised point cloud data, while respecting and leveraging the symmetries of the task at hand.

[*]equal contribution
[†]work done while at the Bosch Center for Artificial Intelligence

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

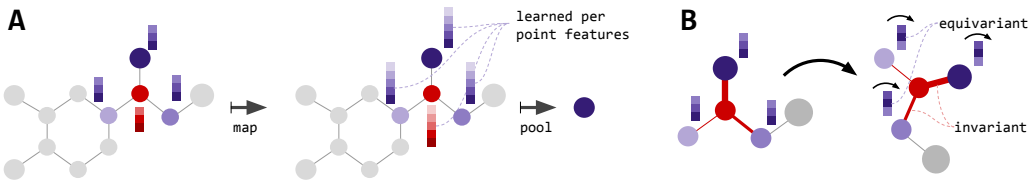



Figure 1: **A**) Each layer of the SE(3)-Transformer maps from a point cloud to a point cloud while guaranteeing equivariance. For classification, this is followed by an invariant pooling layer and an MLP. **B**) In each layer, for each node, attention is performed. Here, the red node attends to its neighbours. Attention weights (indicated by line thickness) are invariant w.r.t. rotation of the input.

Self-attention itself is a pseudo-linear map between sets of points. It can be seen to consist of two components: input-dependent *attention weights* and an embedding of the input, called a *value embedding*. In Fig. 1, we show an example of a molecular graph, where attached to every atom we see a value embedding vector and where the attention weights are represented as edges, with width corresponding to the attention weight magnitude. In the SE(3)-Transformer, we explicitly design the attention weights to be invariant to global pose. Furthermore, we design the value embedding to be equivariant to global pose. Equivariance generalises the translational weight-tying of convolutions. It ensures that transformations of a layer's input manifest as equivalent transformations of the output. SE(3)-equivariance in particular is the generalisation of translational weight-tying in 2D known from conventional convolutions to roto-translations in 3D. This restricts the space of learnable functions to a subspace which adheres to the symmetries of the task and thus reduces the number of learnable parameters. Meanwhile, it provides us with a richer form of invariance, since relative positional information between features in the input is preserved.

Our contributions are the following:

- We introduce a novel self-attention mechanism, guaranteeably invariant to global rotations and translations of its input. It is also equivariant to permutations of the input point labels.
- We show that the SE(3)-Transformer resolves an issue with concurrent SE(3)-equivariant neural networks, which suffer from angularly constrained filters.
- We introduce a `Pytorch` implementation of spherical harmonics, which is 10x faster than `Scipy` on CPU and $100 - 1000\times$ faster on GPU.

## 2 Background And Related Work

In this section we introduce the relevant background materials on self-attention, graph neural networks, and equivariance. We are concerned with point cloud based machine learning tasks, such as object classification or segmentation. In such a task, we are given a point cloud as input, represented as a collection of $n$ coordinate vectors $\mathbf{x}_i \in \mathbb{R}^3$ with optional per-point features $\mathbf{f}_i \in \mathbb{R}^d$.

### 2.1 The Attention Mechanism

The standard *attention mechanism* [28] can be thought of as consisting of three terms: a set of query vectors $\mathbf{q}_i \in \mathbb{R}^p$ for $i = 1, ..., m$, a set of key vectors $\mathbf{k}_j \in \mathbb{R}^p$ for $j = 1, ..., n$, and a set of value vectors $\mathbf{v}_j \in \mathbb{R}^r$ for $j = 1, ..., n$, where $r$ and $p$ are the dimensions of the low dimensional embeddings. We commonly interpret the key $\mathbf{k}_j$ and the value $\mathbf{v}_j$ as being 'attached' to the same point $j$. For a given query $\mathbf{q}_i$, the attention mechanism can be written as

$$\text{Attn}\left(\mathbf{q}_i, \{\mathbf{k}_j\}, \{\mathbf{v}_j\}\right) = \sum_{j=1}^{n} \alpha_{ij}\mathbf{v}_j, \qquad \alpha_{ij} = \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_j)}{\sum_{j'=1}^{n} \exp(\mathbf{q}_i^\top \mathbf{k}_{j'})} \tag{1}$$

where we used a softmax as a nonlinearity acting on the weights. In general, the number of query vectors does not have to equal the number of input points [14]. In the case of *self-attention* the query, key, and value vectors are embeddings of the input features, so

$$\mathbf{q} = h_Q(\mathbf{f}), \qquad \mathbf{k} = h_K(\mathbf{f}), \qquad \mathbf{v} = h_V(\mathbf{f}), \tag{2}$$

where $\{h_Q, h_K, h_V\}$ are, in the most general case, neural networks [27]. For us, query $\mathbf{q}_i$ is associated with a point $i$ in the input, which has a geometric location $\mathbf{x}_i$. Thus if we have $n$ points, we have $n$ possible queries. For query $\mathbf{q}_i$, we say that node $i$ *attends* to all other nodes $j \neq i$.

Motivated by a successes across a wide range of tasks in deep learning such as language modeling [28], image recognition [16], graph-based problems [29], and relational reasoning [27, 7], a recent stream of work has applied forms of self-attention algorithms to point cloud data [39, 37, 14]. One such example is the Set Transformer [14]. When applied to object classification on ModelNet40 [36], the input to the Set Transformer are the cartesian coordinates of the points. Each layer embeds this positional information further while dynamically querying information from other points. The final per-point embeddings are downsampled and used for object classification.

**Permutation equivariance** A key property of self-attention is *permutation equivariance*. Permutations of point labels $1, ..., n$ lead to permutations of the self-attention output. This guarantees the attention output does not depend arbitrarily on input point ordering. Wagstaff et al. [30] recently showed that this mechanism can theoretically approximate *all* permutation equivariant functions. The SE(3)-transformer is a special case of this attention mechanism, inheriting permutation equivariance. However, it limits the space of learnable functions to rotation and translation equivariant ones.

### 2.2 Graph Neural Networks

Attention scales quadratically with point cloud size, so it is useful to introduce neighbourhoods: instead of each point attending to *all* other points, it only attends to its nearest neighbours. Sets with neighbourhoods are naturally represented as graphs. Attention has previously been introduced on graphs under the names of intra-, self-, vertex-, or graph-attention [15, 28, 29, 10, 23]. These methods were unified by Wang et al. [31] with the non-local neural network. This has the simple form

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\{\mathbf{f}_j \in \mathcal{N}_i\})} \sum_{j \in \mathcal{N}_i} w(\mathbf{f}_i, \mathbf{f}_j) h(\mathbf{f}_j) \tag{3}$$

where $w$ and $h$ are neural networks and $\mathcal{C}$ normalises the sum as a function of all features in the neighbourhood $\mathcal{N}_i$. This has a similar structure to attention, and indeed we can see it as performing attention per neighbourhood. While non-local modules do not explicitly incorporate edge-features, it is possible to add them, as done in Veličković et al. [29] and Hoshen [10].

### 2.3 Equivariance

Given a set of transformations $T_g : \mathcal{V} \to \mathcal{V}$ for $g \in G$, where $G$ is an abstract group, a function $\phi : \mathcal{V} \to \mathcal{Y}$ is called equivariant if for every $g$ there exists a transformation $S_g : \mathcal{Y} \to \mathcal{Y}$ such that

$$S_g[\phi(v)] = \phi(T_g[v]) \qquad \text{for all } g \in G, v \in \mathcal{V}. \tag{4}$$

The indices $g$ can be considered as parameters describing the transformation. Given a pair $(T_g, S_g)$, we can solve for the family of equivariant functions $\phi$ satisfying Equation 4. Furthermore, if $(T_g, S_g)$ are linear and the map $\phi$ is also linear, then a very rich and developed theory already exists for finding $\phi$ [5]. In the equivariance literature, deep networks are built from interleaved linear maps $\phi$ and equivariant nonlinearities. In the case of 3D roto-translations it has already been shown that a suitable structure for $\phi$ is a *tensor field network* [25], explained below. Note that Romero et al. [21] recently introduced a 2D roto-translationally equivariant attention module for pixel-based image data.

**Group Representations** In general, the transformations $(T_g, S_g)$ are called *group representations*. Formally, a group representation $\rho : G \to GL(N)$ is a map from a group $G$ to the set of $N \times N$ invertible matrices $GL(N)$. Critically $\rho$ is a *group homomorphism*; that is, it satisfies the following property $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$ for all $g_1, g_2 \in G$. Specifically for 3D rotations $G = SO(3)$, we have a few interesting properties: 1) its representations are orthogonal matrices, 2) all representations can be decomposed as

$$\rho(g) = \mathbf{Q}^\top \left[ \bigoplus_\ell \mathbf{D}_\ell(g) \right] \mathbf{Q}, \tag{5}$$

where $\mathbf{Q}$ is an orthogonal, $N \times N$, change-of-basis matrix [4]; each $\mathbf{D}_\ell$ for $\ell = 0, 1, 2, ...$ is a $(2\ell+1) \times (2\ell+1)$ matrix known as a Wigner-D matrix[3]; and the $\bigoplus$ is the *direct sum* or concatenation

[3]The 'D' stands for *Darstellung*, German for representation

of matrices along the diagonal. The Wigner-D matrices are *irreducible representations* of SO(3)—think of them as the 'smallest' representations possible. Vectors transforming according to $\mathbf{D}_\ell$ (i.e. we set $\mathbf{Q} = \mathbf{I}$, $i = \ell$), are called *type-$\ell$* vectors. Type-0 vectors are invariant under rotations and type-1 vectors rotate according to 3D rotation matrices. Note, type-$\ell$ vectors have length $2\ell + 1$. They can be stacked, forming a feature vector $\mathbf{f}$ transforming according to Eq. (5).

**Tensor Field Networks** Tensor field networks (TFN) [25] are neural networks, which map point clouds to point clouds under the constraint of SE(3)-equivariance, the group of 3D rotations and translations. For point clouds, the input is a vector field $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^d$ of the form

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{f}_j \delta(\mathbf{x} - \mathbf{x}_j), \tag{6}$$

where $\delta$ is the Dirac delta function, $\{\mathbf{x}_j\}$ are the 3D point coordinates and $\{\mathbf{f}_j\}$ are point features, representing such quantities as atomic number or point identity. For equivariance to be satisfied, the features of a TFN transform under Eq. (5), where $\mathbf{Q} = \mathbf{I}$. Each $\mathbf{f}_j$ is a concatenation of vectors of different *types*, where a subvector of type-$\ell$ is written $\mathbf{f}_j^\ell$. A TFN layer computes the convolution of a continuous-in-space, learnable weight kernel $\mathbf{W}^{\ell k} : \mathbb{R}^3 \to \mathbb{R}^{(2\ell+1)\times(2k+1)}$ from type-$k$ features to type-$\ell$ features. The type-$\ell$ output of the TFN layer at position $\mathbf{x}_i$ is

$$\mathbf{f}_{\text{out},i}^\ell = \sum_{k\geq 0} \underbrace{\int \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\mathbf{f}_{\text{in}}^k(\mathbf{x}')\,\mathrm{d}\mathbf{x}'}_{k\to\ell \text{ convolution}} = \sum_{k\geq 0}\sum_{j=1}^{n} \underbrace{\mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^k}_{\text{node } j \to \text{node } i \text{ message}}, \tag{7}$$

We can also include a sum over input channels, but we omit it here. Weiler et al. [33], Thomas et al. [25] and Kondor [13] showed that the kernel $\mathbf{W}^{\ell k}$ lies in the span of an equivariant basis $\{\mathbf{W}_J^{\ell k}\}_{J=|k-\ell|}^{k+\ell}$. The kernel is a linear combination of these basis kernels, where the $J^{\text{th}}$ coefficient is a learnable function $\varphi_J^{\ell k} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ of the radius $\|\mathbf{x}\|$. Mathematically this is

$$\mathbf{W}^{\ell k}(\mathbf{x}) = \sum_{J=|k-\ell|}^{k+\ell} \varphi_J^{\ell k}(\|\mathbf{x}\|)\mathbf{W}_J^{\ell k}(\mathbf{x}), \qquad \text{where } \mathbf{W}_J^{\ell k}(\mathbf{x}) = \sum_{m=-J}^{J} Y_{Jm}(\mathbf{x}/\|\mathbf{x}\|)\mathbf{Q}_{Jm}^{\ell k}. \tag{8}$$

Each basis kernel $\mathbf{W}_J^{\ell k} : \mathbb{R}^3 \to \mathbb{R}^{(2\ell+1)\times(2k+1)}$ is formed by taking a linear combination of Clebsch-Gordan matrices $\mathbf{Q}_{Jm}^{\ell k}$ of shape $(2\ell + 1) \times (2k + 1)$, where the $J, m^{\text{th}}$ linear combination coefficient is the $m^{\text{th}}$ dimension of the $J^{\text{th}}$ spherical harmonic $Y_J : \mathbb{R}^3 \to \mathbb{R}^{2J+1}$. Each basis kernel $\mathbf{W}_J^{\ell k}$ completely constrains the form of the learned kernel in the angular direction, leaving the only learnable degree of freedom in the radial direction. Note that $\mathbf{W}_J^{\ell k}(\mathbf{0}) \neq \mathbf{0}$ only when $k = \ell$ and $J = 0$, which reduces the kernel to a scalar $w$ multiplied by the identity, $\mathbf{W}^{\ell\ell} = w^{\ell\ell}\mathbf{I}$, referred to as *self-interaction* [25]. As such we can rewrite the TFN layer as

$$\mathbf{f}_{\text{out},i}^\ell = \underbrace{w^{\ell\ell}\mathbf{f}_{\text{in},i}^\ell}_{\text{self-interaction}} + \sum_{k\geq 0}\sum_{j\neq i}^{n} \mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^k, \tag{9}$$

Eq. (7) and Eq. (9) present the convolution in message-passing form, where messages are aggregated from all nodes and feature types. They are also a form of nonlocal graph operation as in Eq. (3), where the weights are functions on edges and the features $\{\mathbf{f}_i\}$ are node features. We will later see how our proposed attention layer unifies aspects of convolutions and graph neural networks.

## 3 Method

Here, we present the *SE(3)-Transformer*. The layer can be broken down into a procedure of steps as shown in Fig. 2, which we describe in the following section. These are the construction of a graph from a point cloud, the construction of equivariant edge functions on the graph, how to propagate SE(3)-equivariant messages on the graph, and how to aggregate them. We also introduce an alternative for the self-interaction layer, which we call *attentive self-interaction*.

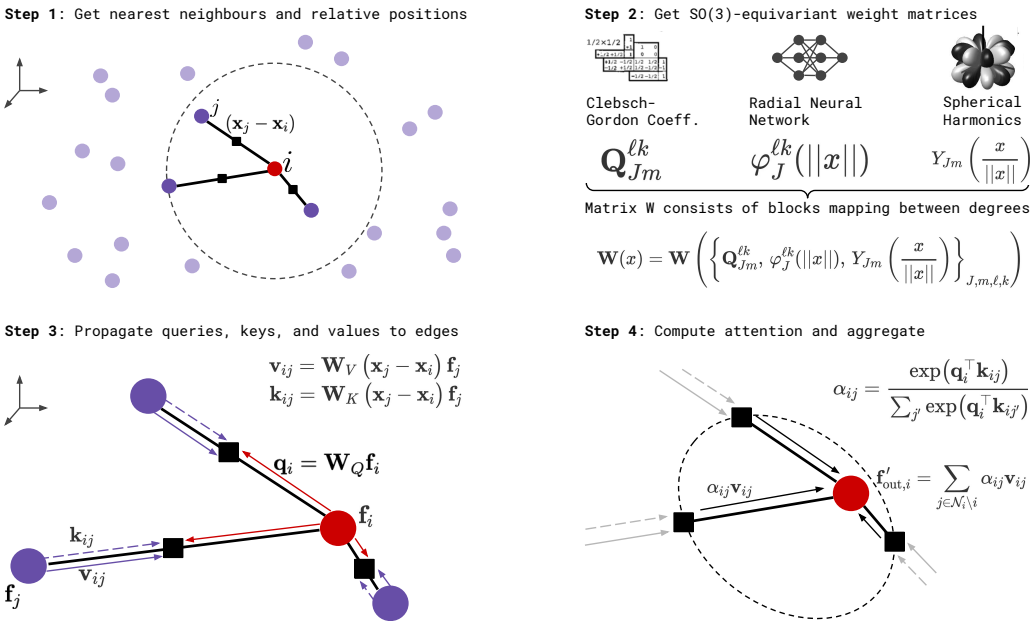



Figure 2: Updating the node features using our equivariant attention mechanism in four steps. A more detailed description, especially of step 2, is provided in the Appendix. Steps 3 and 4 visualise a graph network perspective: features are passed from nodes to edges to compute keys, queries and values, which depend both on features and relative positions in a rotation-equivariant manner.

### 3.1 Neighbourhoods

Given a point cloud $\{(\mathbf{x}_i, \mathbf{f}_i)\}$, we first introduce a collection of neighbourhoods $\mathcal{N}_i \subseteq \{1, ..., N\}$, one centered on each point $i$. These neighbourhoods are computed either via the nearest-neighbours methods or may already be defined. For instance, molecular structures have neighbourhoods defined by their bonding structure. Neighbourhoods reduce the computational complexity of the attention mechanism from quadratic in the number of points to linear. The introduction of neighbourhoods converts our point cloud into a graph. This step is shown as Step 1 of Fig. 2.

### 3.2 The SE(3)-Transformer

The SE(3)-Transformer itself consists of three components. These are 1) edge-wise attention weights $\alpha_{ij}$, constructed to be SE(3)-invariant on each edge $ij$, 2) edge-wise SE(3)-equivariant value messages, propagating information between nodes, as found in the TFN convolution of Eq. (7), and 3) a linear/attentive self-interaction layer. Attention is performed on a per-neighbourhood basis as follows:

$$\mathbf{f}_{\text{out},i}^{\ell} = \underbrace{\mathbf{W}_V^{\ell\ell}\mathbf{f}_{\text{in},i}^{\ell}}_{\text{③ self-interaction}} + \sum_{k \geq 0} \sum_{j \in \mathcal{N}_i \setminus i} \underbrace{\alpha_{ij}}_{\text{① attention}} \underbrace{\mathbf{W}_V^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}}_{\text{② value message}}. \tag{10}$$

These components are visualised in Fig. 2. If we remove the attention weights then we have a tensor field convolution, and if we instead remove the dependence of $\mathbf{W}_V$ on $(\mathbf{x}_j - \mathbf{x}_i)$, we have a conventional attention mechanism. Provided that the attention weights $\alpha_{ij}$ are invariant, Eq. (10) is equivariant to SE(3)-transformations. This is because it is just a linear combination of equivariant value messages. Invariant attention weights can be achieved with a dot-product attention structure shown in Eq. (11). This mechanism consists of a normalised inner product between a query vector $\mathbf{q}_i$ at node $i$ and a set of key vectors $\{\mathbf{k}_{ij}\}_{j \in \mathcal{N}_i}$ along each edge $ij$ in the neighbourhood $\mathcal{N}_i$ where

$$\alpha_{ij} = \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_{ij})}{\sum_{j' \in \mathcal{N}_i \setminus i} \exp(\mathbf{q}_i^\top \mathbf{k}_{ij'})}, \quad \mathbf{q}_i = \bigoplus_{\ell \geq 0} \sum_{k \geq 0} \mathbf{W}_Q^{\ell k}\mathbf{f}_{\text{in},i}^{k}, \quad \mathbf{k}_{ij} = \bigoplus_{\ell \geq 0} \sum_{k \geq 0} \mathbf{W}_K^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}. \tag{11}$$

$\bigoplus$ is the direct sum, i.e. vector concatenation in this instance. The linear embedding matrices $\mathbf{W}_Q^{\ell k}$ and $\mathbf{W}_K^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)$ are of TFN type (c.f. Eq. (8)). The attention weights $\alpha_{ij}$ are invariant for the following reason. If the input features $\{\mathbf{f}_{\text{in},j}\}$ are SO(3)-equivariant, then the query $\mathbf{q}_i$ and key vectors $\{\mathbf{k}_{ij}\}$ are also SE(3)-equivariant, since the linear embedding matrices are of TFN type.

The inner product of SO(3)-equivariant vectors, transforming under the same representation $\mathbf{S}_g$ is invariant, since if $\mathbf{q} \mapsto \mathbf{S}_g\mathbf{q}$ and $\mathbf{k} \mapsto \mathbf{S}_g\mathbf{k}$, then $\mathbf{q}^\top\mathbf{S}_g^\top\mathbf{S}_g\mathbf{k} = \mathbf{q}^\top\mathbf{k}$, because of the orthonormality of representations of SO(3), mentioned in the background section. We follow the common practice from the self-attention literature [28, 14], and chosen a softmax nonlinearity to normalise the attention weights to unity, but in general any nonlinear function could be used.

**Aside: Angular Modulation** The attention weights add extra degrees of freedom to the TFN kernel in the angular direction. This is seen when Eq. (10) is viewed as a convolution with a data-dependent kernel $\alpha_{ij}\mathbf{W}_V^{\ell k}(\mathbf{x})$. In the literature, SO(3) equivariant kernels are decomposed as a sum of products of learnable radial functions $\varphi_J^{\ell k}(\|\mathbf{x}\|)$ and non-learnable angular kernels $\mathbf{W}_J^{\ell k}(\mathbf{x}/\|\mathbf{x}\|)$ (c.f. Eq. (8)). The fixed angular dependence of $\mathbf{W}_J^{\ell k}(\mathbf{x}/\|\mathbf{x}\|)$ is a strange artifact of the equivariance condition in noncommutative algebras and while necessary to guarantee equivariance, it is seen as overconstraining the expressiveness of the kernels. Interestingly, the attention weights $\alpha_{ij}$ introduce a means to modulate the angular profile of $\mathbf{W}_J^{\ell k}(\mathbf{x}/\|\mathbf{x}\|)$, while maintaining equivariance.

**Channels, Self-interaction Layers, and Non-Linearities** Analogous to conventional neural networks, the SE(3)-Transformer can straightforwardly be extended to multiple channels per representation degree $\ell$, so far omitted for brevity. This sets the stage for self-interaction layers. The attention layer (c.f. Fig. 2 and circles 1 and 2 of Eq. (10)) aggregates information over nodes and input representation degrees $k$. In contrast, the self-interaction layer (c.f. circle 3 of Eq. (10)) exchanges information solely between features of the same degree and within one node—much akin to 1x1 convolutions in CNNs. Self-interaction is an elegant form of learnable skip connection, transporting information from query point $i$ in layer $L$ to query point $i$ in layer $L+1$. This is crucial since, in the SE(3)-Transformer, points do not attend to themselves. In our experiments, we use two different types of self-interaction layer: (1) linear and (2) attentive, both of the form

$$\mathbf{f}_{\text{out},i,c'}^{\ell} = \sum_c w_{i,c'c}^{\ell\ell}\mathbf{f}_{\text{in},i,c}^{\ell}. \tag{12}$$

**Linear:** Following Schütt et al. [22], output channels are a learned linear combination of input channels using one set of weights $w_{i,c'c}^{\ell\ell} = w_{c'c}^{\ell\ell}$ per representation degree, shared across all points. As proposed in Thomas et al. [25], this is followed by a norm-based non-linearity.

**Attentive**: We propose an extension of linear self-interaction, *attentive self-interaction*, combining self-interaction and nonlinearity. We replace the learned scalar weights $w_{c'c}^{\ell\ell}$ with attention weights output from an MLP, shown in Eq. (13) ($\bigoplus$ means concatenation.). These weights are SE(3)-invariant due to the invariance of inner products of features, transforming under the same representation.

$$w_{i,c'c}^{\ell\ell} = \text{MLP}\left(\bigoplus_{c,c'} \mathbf{f}_{\text{in},i,c'}^{\ell\top}\mathbf{f}_{\text{in},i,c}^{\ell}\right) \tag{13}$$

### 3.3 Node and Edge Features

Point cloud data often has information attached to points (node-features) and connections between points (edge-features), which we would both like to pass as inputs into the first layer of the network. Node information can directly be incorporated via the tensors $\mathbf{f}_j$ in Eqs. (6) and (10). For incorporating edge information, note that $\mathbf{f}_j$ is part of multiple neighbourhoods. One can replace $\mathbf{f}_j$ with $\mathbf{f}_{ij}$ in Eq. (10). Now, $\mathbf{f}_{ij}$ can carry different information depending on which neighbourhood $\mathcal{N}_i$ we are currently performing attention over. In other words, $\mathbf{f}_{ij}$ can carry information both about node $j$ but also about edge $ij$. Alternatively, if the edge information is scalar, it can be incorporated into the weight matrices $\mathbf{W}_V$ and $\mathbf{W}_K$ as an input to the radial network (see step 2 in Fig. 2).

## 4 Experiments

We test the efficacy of the SE(3)-Transformer on three datasets, each testing different aspects of the model. The N-body problem is an equivariant task: rotation of the input should result in rotated predictions of locations and velocities of the particles. Next, we evaluate on a real-world object classification task. Here, the network is confronted with large point clouds of noisy data with symmetry only around the gravitational axis. Finally, we test the SE(3)-Transformer on a molecular property regression task, which shines light on its ability to incorporate rich graph structures. We compare to publicly available, state-of-the-art results as well as a set of our own baselines. Specifically,

Table 1: Predicting future locations and velocities in an electron-proton simulation.

| | | Linear | DeepSet [40] | Tensor Field [25] | Set Transformer [14] | **SE(3)-Transformer** |
|---|---|---|---|---|---|---|
| **Position** | MSE $x$ | 0.0691 | 0.0639 | 0.0151 | 0.0139 | **0.0076** |
| | std | - | 0.0086 | 0.0011 | 0.0004 | 0.0002 |
| | $\Delta_{EQ}$ | - | 0.038 | $1.9 \cdot 10^{-7}$ | 0.167 | $3.2 \cdot 10^{-7}$ |
| **Velocity** | MSE $v$ | 0.261 | 0.246 | 0.125 | 0.101 | **0.075** |
| | std | - | 0.017 | 0.002 | 0.004 | 0.001 |
| | $\Delta_{EQ}$ | - | 1.11 | $5.0 \cdot 10^{-7}$ | 0.37 | $6.3 \cdot 10^{-7}$ |

we compare to the Set-Transformer [14], a non-equivariant attention model, and Tensor Field Networks [25], which is similar to SE(3)-Transformer but does not leverage attention.

Similar to [24, 34], we measure the exactness of equivariance by applying uniformly sampled SO(3)-transformations to input and output. The distance between the two, averaged over samples, yields the equivariance error. Note that, unlike in Sosnovik et al. [24], the error is not squared:

$$\Delta_{EQ} = \|L_s \Phi(f) - \Phi L_s(f)\|_2 \,/\, \|L_s \Phi(f)\|_2 \tag{14}$$

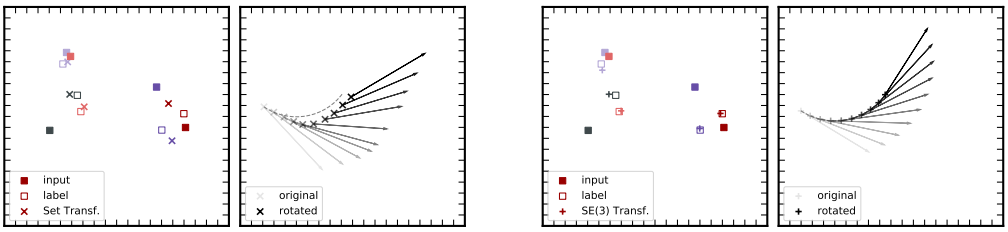
### 4.1 N-Body Simulations

In this experiment, we use an adaptation of the dataset from Kipf et al. [12]. Five particles each carry either a positive or a negative charge and exert repulsive or attractive forces on each other. The input to the network is the position of a particle in a specific time step, its velocity, and its charge. The task of the algorithm is then to predict the relative location and velocity 500 time steps into the future. We deliberately formulated this as a regression problem to avoid the need to predict multiple time steps iteratively. Even though it certainly is an interesting direction for future research to combine equivariant attention with, e.g., an LSTM, our goal here was to test our core contribution and compare it to related models. This task sets itself apart from the other two experiments by not being invariant but equivariant: When the input is rotated or translated, the output changes respectively (see Fig. 3).

We trained an SE(3)-Transformer with 4 equivariant layers, each followed by an attentive self-interaction layer (details are provided in the Appendix). Table 1 shows quantitative results. Our model outperforms both an attention-based, but not rotation-equivariant approach (Set Transformer) and a equivariant approach which does not levarage attention (Tensor Field). The equivariance error shows that our approach is indeed fully rotation equivariant up to the precision of the computations.

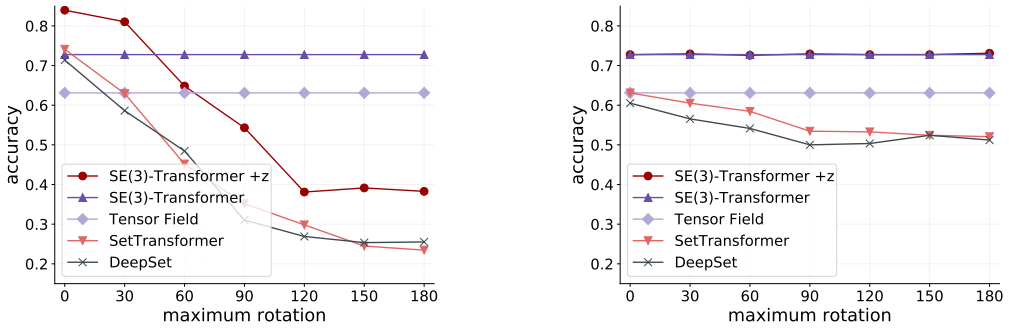
### 4.2 Real-World Object Classification on ScanObjectNN

ScanObjectNN is a recently introduced dataset for real-world object classification. The benchmark provides point clouds of 2902 objects across 15 different categories. We only use the coordinates of the points as input and object categories as training labels. We train an SE(3)-Transformer with 4 equivariant layers with linear self-interaction followed by max-pooling and an MLP. Interestingly, the task is not fully rotation invariant, in a statistical sense, as the objects are aligned with respect to the gravity axis. This results in a performance loss when deploying a fully SO(3) invariant model (see Fig. 4a). In other words: when looking at a new object, it helps to know where 'up' is. We create an SO(2) invariant version of our algorithm by additionally feeding the $z$-component as an




(a) Set Transformer (b) SE(3)-Transformer

Figure 3: A model based on conventional self-attention (left) and our rotation-equivariant version (right) predict future locations and velocities in a 5-body problem. The respective left-hand plots show input locations at time step $t = 0$, ground truth locations at $t = 500$, and the respective predictions. The right-hand plots show predicted locations and velocities for rotations of the input in steps of 10 degrees. The dashed curves show the predicted locations of a perfectly equivariant model.




(a) Training *without* data augmentation.

(b) Training *with* data augmentation.

Figure 4: ScanObjectNN: $x$-axis shows data augmentation on the test set. The $x$-value corresponds to the maximum rotation around a random axis in the $x$-$y$-plane. If both training and test set are not rotated ($x = 0$ in **a**), breaking the symmetry of the SE(3)-Transformer by providing the $z$-component of the coordinates as an additional, scalar input improves the performance significantly. Interestingly, the model learns to ignore the additional, symmetry-breaking input when the training set presents a rotation-invariant problem (strongly overlapping dark red circles and dark purple triangles in **b**).

Table 2: Classification accuracy on the 'object only' category of the ScanObjectNN dataset[4]. The performance of the SE(3)-Transformer is averaged over 5 runs (standard deviation 0.7%).

| | DeepSet | 3DmFV | Set Transformer | PointNet | SpiderCNN | Tensor Field +z | PointNet++ | **SE(3)-Transf.+z** | PointCNN | DGCNN | PointGLR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| No. Points | 1024 | 1024 | 1024 | 1024 | 1024 | 128 | 1024 | **128** | 1024 | 1024 | 1024 |
| Accuracy | 71.4% | 73.8% | 74.1% | 79.2% | 79.5% | 81.0% | 84.3% | **85.0%** | 85.5% | 86.2% | 87.2% |

type-0 field and the $x$, $y$ position as an additional type-1 field (see Appendix). We dub this model *SE(3)-Transformer +z*. This way, the model can 'learn' which symmetries to adhere to by suppressing and promoting different inputs (compare Fig. 4a and Fig. 4b). In Table 2, we compare our model to the current state-of-the-art in object classification[4]. Despite the dataset not playing to the strengths of our model (full SE(3)-invariance) and a much lower number of input points, the performance is competitive with models specifically designed for object classification.

### 4.3 QM9

The QM9 regression dataset [19] is a publicly available chemical property prediction task. There are 134k molecules with up to 29 atoms per molecule. Atoms are represented as a 5 dimensional one-hot node embeddings in a molecular graph connected by 4 different chemical bond types (more details in Appendix). 'Positions' of each atom are provided. We show results on the test set of Anderson et al. [1] for 6 regression tasks in Table 3. Lower is better. The table is split into non-equivariant (top) and equivariant models (bottom). Our nearest models are Cormorant and TFN (own implementation). We see that while not state-of-the-art, we offer competitive performance, especially against Cormorant and TFN, which transform under irreducible representations of SE(3) (like us), unlike LieConv(T3), using a left-regular representation of SE(3), which may explain its success.

Table 3: QM9 Mean Absolute Error. Top: Non-equivariant models. Bottom: Equivariant models

| TASK<br>UNITS | $\alpha$<br>bohr$^3$ | $\Delta\varepsilon$<br>meV | $\varepsilon_{\text{HOMO}}$<br>meV | $\varepsilon_{\text{LUMO}}$<br>meV | $\mu$<br>D | $C_\nu$<br>cal/mol K |
|---|---|---|---|---|---|---|
| WaveScatt [9] | .160 | 118 | 85 | 76 | .340 | .049 |
| NMP [8] | .092 | 69 | 43 | 38 | .030 | .040 |
| SchNet [22] | .235 | 63 | 41 | 34 | .033 | .033 |
| Cormorant [1] | .085 | 61 | 34 | 38 | .038 | .026 |
| LieConv(T3) [6] | .084 | 49 | 30 | 25 | .032 | .038 |
| TFN [25] | .223 | 58 | 40 | 38 | .064 | .101 |
| Us | .148 | 53 | 36 | 33 | .053 | .057 |

[4]PointGLR is a recently published preprint [20]. The performance of the following models was taken from the official benchmark of the dataset as of June 4th, 2020 (`https://hkust-vgd.github.io/benchmark/`): 3DmFV [3], PointNet [17], SpiderCNN [38], PointNet++ [18], DGCN [32].

## 5 Conclusion

We have presented an attention-based neural architecture designed specifically for point cloud data. This architecture is guaranteed to be robust to rotations and translations of the input, obviating the need for training time data augmentation and ensuring stability to arbitrary choices of coordinate frame. The use of self-attention allows for anisotropic, data-adaptive filters, while the use of neighbourhoods enables scalability to large point clouds. We have also introduced the interpretation of the attention mechanism as a data-dependent nonlinearity, adding to the list of equivariant nonlinearties which we can use in equivariant networks. Furthermore, we provide pseudocode in the Appendix for a speed up of spherical harmonics computation of up to 3 orders of magnitudes. This speed-up allowed us to train significantly larger versions of both the SE(3)-Transformer and the Tensor Field network [25] and to apply these models to real-world datasets.

Our experiments showed that adding attention to a roto-translation-equivariant model consistently led to higher accuracy and increased training stability. Specifically for large neighbourhoods, attention proved essential for model convergence. On the other hand, compared to convential attention, adding the equivariance constraints also increases performance in all of our experiments while at the same time providing a mathematical guarantee for robustness with respect to rotations of the input data.

## Broader Impact

The main contribution of the paper is a mathematically motivated attention mechanism which can be used for deep learning on point cloud based problems. We do not see a direct potential of *negative* impact to the society. However, we would like to stress that this type of algorithm is inherently suited for classification and regression problems on molecules. The SE(3)-Transformer therefore lends itself to application in drug research. One concrete application we are currently investigating is to use the algorithm for early-stage suitability classification of molecules for inhibiting the reproductive cycle of the coronavirus. While research of this sort always requires intensive testing in wet labs, computer algorithms can be and are being used to filter out particularly promising compounds from large databases of millions of molecules.

## Acknowledgements

We would like to express our gratitude to the Bosch Center for Artificial Intelligence and Koninklijke Philips N.V. for their support and contribution to open research in publishing our paper.

## References

[1] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[2] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv Preprint*, 2016.

[3] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in realtime using convolutional neural networks. *IEEE Robotics and Automation Letters*, 2018.

[4] Gregory S Chirikjian, Alexander B Kyatkin, and AC Buckingham. Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups. *Appl. Mech. Rev.*, 54(6):B97–B98, 2001.

[5] Taco S. Cohen and Max Welling. Steerable cnns. *International Conference on Learning Representations (ICLR)*, 2017.

[6] 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.

[7] 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.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, and Oriol Vinyals George E. Dahl. Neural message passing for quantum chemistry. *Proceedings of the International Conference on Machine Learning, ICML*, 2020.

[9] Matthew J. Hirn, Stéphane Mallat, and Nicolas Poilvert. Wavelet scattering regression of quantum chemical energies. *Multiscale Model. Simul.*, 15(2):827–863, 2017.

[10] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

[12] 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.

[13] Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint*, 2018.

[14] 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.

[15] 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.

[16] 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.

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[18] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[19] Raghunathan Ramakrishnan, Pavlo Dral, Matthias Rupp, and Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 08 2014.

[20] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] David W. Romero, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. Attentive group equivariant convolutional networks. 2020.

[22] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela1, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[23] 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.

[24] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. *International Conference on Learning Representations (ICLR)*, 2020.

[25] 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.

[26] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

[27] 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.

[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] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

[30] Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. *International Conference on Machine Learning (ICML)*, 2019.

[31] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[33] 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.

[34] Daniel E. Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[35] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance, 2016.

[36] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[37] 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.

[38] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *European Conference on Computer Vision (ECCV)*, 2018.

[39] 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.

[40] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

# A Group Theory and Tensor Field Networks

**Groups** A *group* is an abstract mathematical concept. Formally a group $(G, \circ)$ consists of a set $G$ and a binary composition operator $\circ : G \times G \to G$ (typically we just use the symbol $G$ to refer to the group). All groups must adhere to the following 4 axioms

- **Closure**: $g \circ h \in G$ for all $g, h \in G$
- **Associativity**: $f \circ (g \circ h) = (f \circ g) \circ h = f \circ g \circ h$ for all $f, g, h \in G$
- **Identity**: There exists an element $e \in G$ such that $e \circ g = g \circ e = g$ for all $g \in G$
- **Inverses**: For each $g \in G$ there exists a $g^{-1} \in G$ such that $g^{-1} \circ g = g \circ g^{-1} = e$

In practice, we omit writing the binary composition operator $\circ$, so would write $gh$ instead of $g \circ h$. Groups can be finite or infinite, countable or uncountable, compact or non-compact. Note that they are not necessarily *commutative*; that is, $gh \neq hg$ in general.

**Actions/Transformations** Groups are useful concepts, because they allow us to describe the structure of *transformations*, also sometimes called *actions*. A transformation (operator) $T_g : \mathcal{X} \to \mathcal{X}$ is an injective map from a space into itself. It is parameterised by an element $g$ of a group $G$. Transformations obey two laws:

- **Closure**: $T_g \circ T_h$ is a valid transformation for all $g, h \in G$
- **Identity**: There exists at least one element $e \in G$ such that $T_e[\mathbf{x}] = \mathbf{x}$ for all $\mathbf{x} \in \mathcal{X}$,

where $\circ$ denotes composition of transformations. For the expression $T_g[\mathbf{x}]$, we say that $T_g$ *acts* on $\mathbf{x}$. It can also be shown that transformations are associative under composition. To codify the structure of a transformation, we note that due to closure we can always write

$$T_g \circ T_h = T_{gh}, \tag{15}$$

If for any $x, y \in \mathcal{X}$ we can always find a group element $g$, such that $T_g[x] = y$, then we call $\mathcal{X}$ a *homogeneous space*. Homogeneous spaces are important concepts, because to each pair of points $x, y$ we can always associate at least one group element.

**Equivariance and Intertwiners** As written in the main body of the text, equivariance is a property of functions $f : \mathcal{X} \to \mathcal{Y}$. Just to recap, given a set of transformations $T_g : \mathcal{X} \to \mathcal{X}$ for $g \in G$, where $G$ is an abstract group, a function $f : \mathcal{X} \to \mathcal{Y}$ is called equivariant if for every $g$ there exists a transformation $S_g : \mathcal{Y} \to \mathcal{Y}$ such that

$$S_g[f(\mathbf{x})] = f(T_g[\mathbf{x}]) \qquad \text{for all } g \in G, \mathbf{x} \in \mathcal{X}. \tag{16}$$

If $f$ is linear and equivariant, then it is called an intertwiner. Two important questions arise: 1) How do we choose $S_g$? 2) once we have $(T_g, S_g)$, how do we solve for $f$? To answer these questions, we need to understand what kinds of $S_g$ are possible. For this, we review representations.

**Representations** A group representation $\rho : G \to GL(N)$ is a map from a group $G$ to the set of $N \times N$ invertible matrices $GL(N)$. Critically $\rho$ is a *group homomorphism*; that is, it satisfies the following property $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$ for all $g_1, g_2 \in G$. Representations can be used as transformation operators, acting on $N$-dimensional vectors $\mathbf{x} \in \mathbb{R}^N$. For instance, for the group of 3D rotations, known as $SO(3)$, we have that 3D rotation matrices, $\rho(g) = \mathbf{R}_g$ act on (i.e., rotate) 3D vectors, as

$$T_g[\mathbf{x}] = \rho(g)\mathbf{x} = \mathbf{R}_g\mathbf{x}, \qquad \text{for all } \mathbf{x} \in \mathcal{X}, g \in G. \tag{17}$$

However, there are many more representations of $SO(3)$ than just the 3D rotation matrices. Among representations, two representations $\rho$ and $\rho'$ of the same dimensionality are said to be *equivalent* if they can be connected by a similarity transformation

$$\rho'(g) = \mathbf{Q}^{-1}\rho(g)\mathbf{Q}, \qquad \text{for all } g \in G. \tag{18}$$

We also say that a representation is *reducible* if is can be written as

$$\rho(g) = \mathbf{Q}^{-1}(\rho_1(g) \oplus \rho_2(g))\mathbf{Q} = \mathbf{Q}^{-1}\begin{bmatrix}\rho_1(g) & \\ & \rho_2(g)\end{bmatrix}\mathbf{Q}, \qquad \text{for all } g \in G. \tag{19}$$

If the representations $\rho_1$ and $\rho_2$ are not reducible, then they are called *irreducible representations* of $G$, or *irreps*. In a sense, they are the atoms among representations, out of which all other representations can be constructed. Note that each irrep acts on a separate subspace, mapping vectors from that space back into it. We say that subspace $\mathcal{X}_\ell \in \mathcal{X}$ is *invariant under* irrep $\rho_\ell$, if $\{\rho_\ell(g)\mathbf{x} \mid \mathbf{x} \in \mathcal{X}_\ell, g \in G\} \subseteq \mathcal{X}_\ell$.

**Representation theory of $SO(3)$** As it turns out, all linear representations of compact groups[5] (such as $SO(3)$) can be decomposed into a direct sum of irreps, as

$$\rho(g) = \mathbf{Q}^\top \left[ \bigoplus_J \mathbf{D}_J(g) \right] \mathbf{Q}, \tag{20}$$

where $\mathbf{Q}$ is an orthogonal, $N \times N$, change-of-basis matrix [4]; and each $\mathbf{D}_J$ for $J = 0, 1, 2, ...$ is a $(2J+1) \times (2J+1)$ matrix known as a Wigner-D matrix. The Wigner-D matrices are the irreducible representations of $SO(3)$. We also mentioned that vectors transforming according to $\mathbf{D}_J$ (i.e. we set $\mathbf{Q} = \mathbf{I}$), are called *type-$J$* vectors. Type-0 vectors are invariant under rotations and type-1 vectors rotate according to 3D rotation matrices. Note, type-$J$ vectors have length $2J+1$. In the previous paragraph we mentioned that irreps act on orthogonal subspaces $\mathcal{X}_0, \mathcal{X}_1, ....$ The orthogonal subspaces corresponding to the Wigner-D matrices are the space of *spherical harmonics*.

**The Spherical Harmonics** The spherical harmonics $\mathbf{Y}_J : S^2 \to \mathbb{C}^{2J+1}$ for $J \geq 0$ are square-integrable complex-valued functions on the sphere $S^2$. They have the satisfying property that they are rotated directly by the Wigner-D matrices as

$$\mathbf{Y}_J(\mathbf{R}_g^{-1}\mathbf{x}) = \mathbf{D}_J^*(g)\mathbf{Y}_J(\mathbf{x}), \qquad \mathbf{x} \in S^2, g \in G, \tag{21}$$

where $\mathbf{D}_J$ is the $J^{\text{th}}$ Wigner-D matrix and $\mathbf{D}_J^*$ is its complex conjugate. They form an orthonormal basis for (the Hilbert space of) square-integrable functions on the sphere $L^2(S^2)$, with inner product given as

$$\langle f, h \rangle_{S^2} = \int_{S^2} f(\mathbf{x}) h^*(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{22}$$

So $\langle Y_{Jm}, Y_{J'm'} \rangle_{S^2} = \delta_{JJ'}\delta_{mm'}$, where $Y_{Jm}$ is the $m^{\text{th}}$ element of $\mathbf{Y}_J$. We can express any function in $L^2(S^2)$ as a linear combination of spherical harmonics, where

$$f(\mathbf{x}) = \sum_{J \geq 0} \mathbf{f}_J^\top \mathbf{Y}_J(\mathbf{x}), \qquad \mathbf{x} \in S^2, \tag{23}$$

where each $\mathbf{f}_J$ is a vector of coefficients of length $2J+1$. And in the opposite direction, we can retrieve the coefficients as

$$\mathbf{f}_J = \int_{S^2} f(\mathbf{x}) \mathbf{Y}_J^*(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{24}$$

following from the orthonormality of the spherical harmonics. This is in fact a Fourier transform on the sphere and the the vectors $\mathbf{f}_J$ can be considered Fourier coefficients. Critically, we can represent rotated functions as

$$f(\mathbf{R}_g^{-1}\mathbf{x}) = \sum_{J \geq 0} \mathbf{f}_J^\top \mathbf{D}_J^*(g) \mathbf{Y}_J(\mathbf{x}), \qquad \mathbf{x} \in S^2, g \in G. \tag{25}$$

**The Clebsch-Gordan Decomposition** In the main text we introduced the *Clebsch-Gordan coefficients*. These are used in the construction of the equivariant kernels. They arise in the situation where we have a tensor product of Wigner-D matrices, which as we will see is part of the equivariance constraint on the form of the equivariant kernels. In representation theory a tensor product of representations is also a representation, but since it is not an easy object to work with, we seek to decompose it into a direct sum of irreps, which are easier. This decomposition is of the form of Eq. (20), written

$$\mathbf{D}_k(g) \otimes \mathbf{D}_\ell(g) = \mathbf{Q}^{\ell k \top} \left[ \bigoplus_{J=|k-\ell|}^{k+\ell} \mathbf{D}_J(g) \right] \mathbf{Q}^{\ell k}. \tag{26}$$

In this specific instance, the change of basis matrices $\mathbf{Q}^{\ell k}$ are given the special name of the Clebsch-Gordan coefficients. These can be found in many mathematical physics libraries.

[5] Over a field of characteristic zero.

**Tensor Field Layers** In Tensor Field Networks [25] and 3D Steerable CNNs [33], the authors solve for the intertwiners between SO(3) equivariant point clouds. Here we run through the derivation again in our own notation.

We begin with a point cloud $f(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{f}_j \delta(\mathbf{x} - \mathbf{x}_j)$, where $\mathbf{f}_j$ is an equivariant point feature. Let's say that $\mathbf{f}_j$ is a type-$k$ feature, which we write as $\mathbf{f}_j^k$ to remind ourselves of the fact. Now say we perform a convolution $*$ with kernel $\mathbf{W}^{\ell k} : \mathbf{R}^3 \to \mathbb{R}^{(2\ell+1)\times(2k+1)}$, which maps from type-$k$ features to type-$\ell$ features. Then

$$\begin{aligned}
\mathbf{f}_{\text{out,i}}^{\ell} &= [\mathbf{W}^{\ell k} * \mathbf{f}_{\text{in}}^{k}](\mathbf{x}) && (27)\\
&= \int_{\mathbb{R}^3} \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\mathbf{f}_{\text{in}}^{k}(\mathbf{x}')\,\mathrm{d}\mathbf{x}' && (28)\\
&= \int_{\mathbb{R}^3} \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\sum_{j=1}^{N}\mathbf{f}_{\text{in},j}^{k}\delta(\mathbf{x}' - \mathbf{x}_j)\,\mathrm{d}\mathbf{x}' && (29)\\
&= \sum_{j=1}^{N}\int_{\mathbb{R}^3} \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}\delta(\mathbf{x}' - \mathbf{x}_j)\,\mathrm{d}\mathbf{x}' \qquad \text{change of variables } \mathbf{x}'' = \mathbf{x}' - \mathbf{x}_j && (30)\\
&= \sum_{j=1}^{N}\int_{\mathbb{R}^3} \mathbf{W}^{\ell k}(\mathbf{x}'' + \mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}\delta(\mathbf{x}'')\,\mathrm{d}\mathbf{x}'' \qquad \text{sifting theorem} && (31)\\
&= \sum_{j=1}^{N}\mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}. && (32)
\end{aligned}$$

Now let's apply the equivariance condition to this expression, then

$$\begin{aligned}
\mathbf{D}_\ell(g)\mathbf{f}_{\text{out,i}}^{\ell} &= \sum_{j=1}^{N}\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}(\mathbf{x}_j - \mathbf{x}_i))\mathbf{D}_k(g)\mathbf{f}_{\text{in},j}^{k} && (33)\\
\implies \mathbf{f}_{\text{out,i}}^{\ell} &= \sum_{j=1}^{N}\mathbf{D}_\ell(g)^{-1}\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}(\mathbf{x}_j - \mathbf{x}_i))\mathbf{D}_k(g)\mathbf{f}_{\text{in},j}^{k} && (34)
\end{aligned}$$

Now we notice that this expression should also be equal to Eq. (32), which is the convolution with an unrotated point cloud. Thus we end up at

$$\boxed{\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x}) = \mathbf{D}_\ell(g)\mathbf{W}^{\ell k}(\mathbf{x})\mathbf{D}_k(g)^{-1},} \tag{35}$$

which is sometimes refered to as the *kernel constraint*. To solve the kernel constraint, we notice that it is a linear equation and that we can rearrange it as

$$\text{vec}(\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})) = (\mathbf{D}_k(g) \otimes \mathbf{D}_\ell(g))\text{vec}(\mathbf{W}^{\ell k}(\mathbf{x})) \tag{36}$$

where we used the identity $\text{vec}(\mathbf{AXB}) = (\mathbf{B}^\top \otimes \mathbf{A})\text{vec}(\mathbf{X})$ and the fact that the Wigner-D matrices are orthogonal. Using the Clebsch-Gordan decomposition we rewrite this as

$$\text{vec}(\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})) = \mathbf{Q}^{\ell k\top}\left[\bigoplus_{J=|k-\ell|}^{k+\ell}\mathbf{D}_J(g)\right]\mathbf{Q}^{\ell k}\text{vec}(\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})). \tag{37}$$

Lastly, we can left multiply both sides by $\mathbf{Q}^{\ell k}$ and denote $\boldsymbol{\eta}^{\ell k}(\mathbf{x}) \triangleq \mathbf{Q}^{\ell k}\text{vec}(\mathbf{W}^{\ell k}(\mathbf{x}))$, noting the the Clebsch-Gordan matrices are orthogonal. At the same time we

$$\boldsymbol{\eta}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x}) = \left[\bigoplus_{J=|k-\ell|}^{k+\ell}\mathbf{D}_J(g)\right]\boldsymbol{\eta}^{\ell k}(\mathbf{x}). \tag{38}$$

Thus we have that $\boldsymbol{\eta}_J^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})$ the $J^{\text{th}}$ subvector of $\boldsymbol{\eta}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})$ is subject to the constraint

$$\boldsymbol{\eta}_J^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x}) = \mathbf{D}_J(g)\boldsymbol{\eta}_J^{\ell k}(\mathbf{x}), \tag{39}$$

which is exactly the transformation law for the spherical harmonics from Eq. (21)! Thus one way how $\mathbf{W}^{\ell k}(\mathbf{x})$ can be constructed is

$$\text{vec}\left(\mathbf{W}^{\ell k}(\mathbf{x})\right) = \mathbf{Q}^{\ell k\top} \bigoplus_{J=|k-\ell|}^{k+\ell} \mathbf{Y}_J(\mathbf{x}). \tag{40}$$

# B Recipe for Building an Equivariant Weight Matrix

One of the core operations in the SE(3)-Transformer is multiplying a feature vector $\mathbf{f}$, which transforms according to $SO(3)$, with a matrix $\mathbf{W}$ while preserving equivariance:

$$S_g[\mathbf{W} * \mathbf{f}](\mathbf{x}) = [\mathbf{W} * T_g[\mathbf{f}]](\mathbf{x}), \tag{41}$$

where $T_g[\mathbf{f}](\mathbf{x}) = \rho_{\text{in}}(g)\mathbf{f}(\mathbf{R}_g^{-1}\mathbf{x})$ and $S_g[\mathbf{f}](\mathbf{x}) = \rho_{\text{out}}(g)\mathbf{f}(\mathbf{R}_g^{-1}\mathbf{x})$. Here, as in the previous section we showed how such a matrix $\mathbf{W}$ could be constructed when mapping between features of type-$k$ and type-$\ell$, where $\rho_{\text{in}}(g)$ is a block diagonal matrix of type-$k$ Wigner-D matrices and similarly $\rho_{\text{in}}(g)$ is made of type-$\ell$ Wigner-D matrices. $\mathbf{W}$ is dependent on the relative position $\mathbf{x}$ and underlies the linear equivariance constraints, but is also has learnable components, which we did not show in the previous section. In this section, we show how such a matrix is constructed in practice.

Previously we showed that

$$\text{vec}\left(\mathbf{W}^{\ell k}(\mathbf{x})\right) = \mathbf{Q}^{\ell k\top} \bigoplus_{J=|k-\ell|}^{k+\ell} \mathbf{Y}_J(\mathbf{x}), \tag{42}$$

which is an equivariant mapping between vectors of types $k$ and $\ell$. In practice, we have multiple input vectors $\{\mathbf{f}_c^k\}_c$ of type-$k$ and multiple output vectors of type-$\ell$. For simplicity, however, we ignore this and pretend we only have a single input and single output. Note that $\mathbf{W}^{\ell k}$ has no learnable components. Note that the kernel constraint only acts in the angular direction, but not in the radial direction, so we can introduce scalar radial functions $\varphi_J^{\ell k} : \mathbf{R}_{\geq 0} \to \mathbf{R}$ (one for each $J$), such that

$$\text{vec}\left(\mathbf{W}^{\ell k}(\mathbf{x})\right) = \mathbf{Q}^{\ell k\top} \bigoplus_{J=|k-\ell|}^{k+\ell} \varphi_J^{\ell k}(\|\mathbf{x}\|)\mathbf{Y}_J(\mathbf{x}), \tag{43}$$

The radial functions $\varphi_J^{\ell k}(\|\mathbf{x}\|)$ act as an independent, learnable scalar factor for each degree $J$. The vectorised matrix has dimensionality $(2\ell+1)(2k+1)$. We can unvectorise the above yielding

$$\mathbf{W}^{\ell k}(\mathbf{x}) = \text{unvec}\left(\mathbf{Q}^{\ell k\top} \bigoplus_{J=|k-\ell|}^{k+\ell} \varphi_J^{\ell k}(\|\mathbf{x}\|)\mathbf{Y}_J(\mathbf{x})\right) \tag{44}$$

$$= \sum_{J=|k-\ell|}^{k+\ell} \varphi_J^{\ell k}(\|\mathbf{x}\|)\text{unvec}\left(\mathbf{Q}_J^{\ell k\top}\mathbf{Y}_J(\mathbf{x})\right) \tag{45}$$

where $\mathbf{Q}_J^{\ell k}$ is a $(2\ell+1)(2k+1) \times (2J+1)$ slice from $\mathbf{Q}^{\ell k}$, corresponding to spherical harmonic $\mathbf{Y}_J$. As we showed in the main text, we can also rewrite the unvectorised Clebsch-Gordan–spherical harmonic matrix-vector product as

$$\text{unvec}\left(\mathbf{Q}_J^{\ell k\top}\mathbf{Y}_J(\mathbf{x})\right) = \sum_{m=-J}^{J} \mathbf{Q}_{Jm}^{\ell k\top} Y_{Jm}(\mathbf{x}). \tag{46}$$

In contrast to Weiler et al. [33], we do not voxelise space and therefore $\mathbf{x}$ will be different for each pair of points in each point cloud. However, the same $\mathbf{Y}_J(\mathbf{x})$ will be used multiple times in the network and even multiple times in the same layer. Hence, precomputing them at the beginning of each forward pass for the entire network can significantly speed up the computation. The Clebsch-Gordan coefficients do not depend on the relative positions and can therefore be precomputed once and stored on disk. Multiple libraries exist which approximate those coefficients numerically.

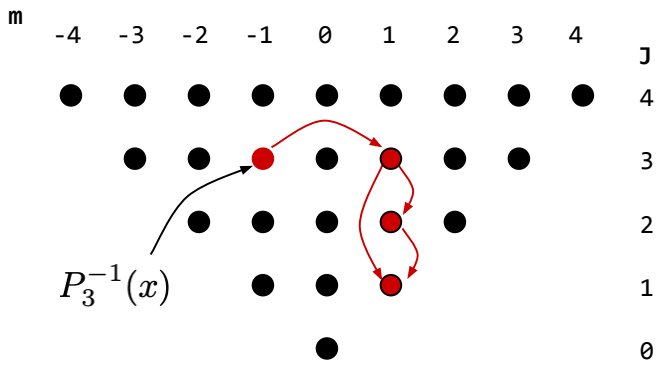



Figure 5: Subproblem graph for the computatin of the associated Legendre polynomials. To compute $P_3^{-1}(x)$, we compute $P_3^1(x)$, for which we need $P_2^1(x)$ and $P_1^1(x)$. We store each intermediate computation, speeding up average computation time by a factor of $\sim 10$ on CPU.

## C Accelerated Computation of Spherical Harmonics

We wrote our own spherical harmonics library in `Pytorch`, which can generate spherical harmonics on the GPU. We found this critical to being able to run the SE(3)-Transformer and Tensor Field network baselines in a reasonable time. This library is accurate to within machine precision against the `scipy` counterpart `scipy.special.sph_harm` and is 10x faster on CPU and 100-1000x on GPU. Here we outline our method to generate them.

The tesseral/real spherical harmonics are given as

$$Y_{Jm}(\theta, \phi) = \sqrt{\frac{2J+1}{4\pi} \frac{(J-m)!}{(J+m)!}} P_J^{|m|}(\cos\theta) \cdot \begin{cases} \sin(|m|\phi) & m < 0, \\ 1 & m = 0, \\ \cos(m\phi) & m > 0, \end{cases} \tag{47}$$

where $P_J^{|m|}$ is the associated Legendre polynomial (ALP), $\theta \in [0, 2\pi)$ is azimuth, and $\phi \in [0, \pi]$ is a polar angle. The term $P_J^{|m|}$ is by far the most expensive component to compute and can be computed recursively. To speed up the computation, we use dynamic programming storing intermediate results in a memoization.

We make use of the following recursion relations in the computation of the ALPs:

$$P_J^{|J|}(x) = (-1)^{|J|} \cdot (1-x^2)^{|J|/2} \cdot (2|J|-1)!! \qquad \text{boundary: } J = m \tag{48}$$

$$P_J^{-m}(x) = (-1)^J \frac{(\ell - m)!}{(\ell + m)!} P_J^m(x) \qquad \text{negate } m \tag{49}$$

$$P_J^{|m|}(x) = \frac{2J-1}{J-|m|} x P_{J-1}^m(x) + \mathbb{I}[J - |m| > 1] \frac{J + |m| - 1}{J - |m|} P_{J-2}^m(x) \qquad \text{recurse} \tag{50}$$

where the *semifactorial* is defined as $x!! = x(x-2)(x-4)\cdots$, and $\mathbb{I}$ is the indicator function. These relations are helpful because they define a recursion.

To understand how we recurse, we consider an example. Fig. 5 shows the space of $J$ and $m$. The black vertices represent a particular ALP, for instance, we have highlighted $P_3^{-1}(x)$. When $m < 0$, we can use Eq. (49) to compute $P_3^{-1}(x)$ from $P_3^1(x)$. We can then use Eq. (50) to compute $P_3^1(x)$ from $P_2^1(x)$ and $P_1^1(x)$. $P_2^1(x)$ can also be computed from Eq. (50) and the boundary value $P_1^1(x)$ can be computed directly using Eq. (48). Crucially, all intermediate ALPs are stored for reuse. Say we wanted to compute $P_4^{-1}(x)$, then we could use Eq. (49) to find it from $P_4^{-1}(x)$, which can be recursed from the stored values $P_3^1(x)$ and $P_2^1(x)$, without needing to recurse down to the boundary.

We intend to make the code for the spherical harmonics computation publicly available.

# D Experimental Details

## D.1 ScanObjectNN

**SE(3)-Transformer and Tensor Field Network** A particularity of object classification from point clouds is the large number of points the algorithm needs to handle. We use up to 200 points out of the available 2024 points per sample and create neighbourhoods with up to 40 nearest neighbours. It is worth pointing out that especially in this setting, adding self-attention (i.e. when comparing the SE(3) Transformer to Tensor Field Networks) significantly increased the stability. As a result, whenever we swapped out the attention mechanism for a convolution to retrieve the Tensor Field network baseline, we had to decrease the model size to obtain stable training. However, we would like to stress that all the Tensor Field networks we trained were significantly bigger than in the original paper [25], mostly enabled by the faster computation of the spherical harmonics.

For the ablation study in Fig. 4, we trained networks with 4 hidden equivariant layers with 5 channels each, and up to representation degree 2. This results in a hidden feature size per point of

$$5 \cdot \sum_{\ell=0}^{2} (2\ell + 1) = 45 \tag{51}$$

We used 200 points of the point cloud and neighbourhood size 40. For the Tensor Field network baseline, in order to achieve stable training, we used a smaller model with 3 instead of 5 channels, 100 input points and neighbourhood size 10, but with representation degrees up to 3.

We used 1 head per attention mechanism yielding one attention weight for each pair of points but across all channels and degrees (for an implementation of multi-head attention, see Appendix D.3). For the query embedding, we used the identity matrix. For the key embedding, we used a quadratic equivariant matrix preserving the number of degrees and channels per degree.

For the quantitative comparison to the start-of-the-art in Table 2, we used 128 input points and neighbourhood size 10 for both the Tensor Field network baseline and the SE(3)-Transformer. We used farthest point sampling with a random starting point to retrieve the 128 points from the overall point cloud. We used degrees up to 3 and 5 channels per degree, which we again had to reduce to 3 channels for the Tensor Field network to obtain stable training. We used a norm based non-linearity for the Tensor Field network (as in [25]) and no extra non-linearity (beyond the *softmax* in the self-attention algorithm) for the SE(3) Transformer.

For all experiments, the final layer of the equivariant encoder maps to 64 channels of degree 0 representations. This yields a 64-dimensional SE(3) *invariant* representation per point. Next, we pool over the point dimension followed by an MLP with one hidden layer of dimension 64, a ReLU and a 15 dimensional output with a cross entropy loss. We trained for 60000 steps with batch size 10. We used the Adam optimizer [11] with a start learning of `1e-2` and a reduction of the learning rate by 70% every 15000 steps. Training took up to 2 days on a system with 4 CPU cores, 30 GB of RAM, and an NVIDIA GeForce GTX 1080 Ti GPU.

The input to the Tensorfield network and the Se(3) Transformer are relative x-y-z positions of each point w.r.t. their neighbours. To guarantee equivariance, these inputs are provided as fields of degree 1. For the '+z' versions, however, we deliberately break the SE(3) equivariance by providing additional and relative z-position as two additional scalar fields (i.e. degree 0), as well as relative x-y positions as a degree 1 field (where the z-component is set to 0).

**DeepSet Baseline** We originally replicated the implementation proposed in [40] for their object classification experiment on ModelNet40 [36]. However, most likely due to the relatively small number of objects in the ScanObjectNN dataset, we found that reducing the model size helped the performance significantly. The reported model had 128 units per hidden layer (instead of 256) and no dropout but the same number of layers and type of non-linearity as in [40].

**Set Transformer Baseline** We used the same architecture as [14] in their object classification experiment on ModelNet40 [36] with an ISAB (induced set attention block)-based encoder followed by PMA (pooling by multihead attention) and an MLP.

## D.2 Relational Inference

Following Kipf et al. [12], we simulated trajectories for $5$ charged, interacting particles. Instead of a 2d simulation setup, we considered a 3d setup. Positive and negative charges were drawn as Bernoulli

trials ($p = 0.5$). We used the provided code base `https://github.com/ethanfetaya/nri` with the following modifications: While we randomly sampled initial positions inside a $[-5, 5]^3$ box, we removed the bounding-boxes during the simulation. We generated $5$k simulation samples for training and $1$k for testing. Instead of phrasing it as a time-series task, we posed it as a regression task: The input data is positions and velocities at a random time step as well as the signs of the charges. The labels (which the algorithm is learning to predict) are the positions and velocities 500 simulation time steps into the future.

**Training Details** We trained each model for 100,000 steps with batch size 128 using an Adam optimizer [11]. We used a fixed learning rate throughout training and conducted a separate hyper parameter search for each model to find a suitable learning rate.

**SE(3)-Transformer Architecture** We trained an SE(3)-Transformer with 4 equivariant layers, where the hidden layers had representation degrees $\{0, 1, 2, 3\}$ and 3 channels per degree. The input is handled as two type-1 fields (for positions and velocities) and one type-0 field (for charges). The learning rate was set to `3e-3`. Each layer included attentive self-interaction.

We used 1 head per attention mechanism yielding one attention weight for each pair of points but across all channels and degrees (for an implementation of multi-head attention, see Appendix D.3). For the query embedding, we used the identity matrix. For the key embedding, we used a quadratic equivariant matrix preserving the number of degrees and channels per degree.

**Baseline Architectures** All our baselines fulfill permutation invariance (ordering of input points), but only the Tensor Field network and the linear baseline are SE(3) equivariant. For the **Tensor Field Network**[25] baseline, we used the same hyper parameters as for the SE(3) Transformer but with a linear self-interaction and an additional norm-based nonlinearity in each layer as in Thomas et al. [25]. For the **DeepSet**[40] baseline, we used 3 fully connected layers, a pooling layer, and two more fully connected layers with 64 units each. All fully connected layers act pointwise. The pooling layer uses max pooling to aggregate information from all points, but concatenates this with a skip connection for each point. Each hidden layer was followed by a LeakyReLU. The learning rate was set to `1e-3`. For the **Set Transformer**[14], we used 4 self-attention blocks with 64 hidden units and 4 heads each. For each point this was then followed by a fully connected layer (64 units), a LeakyReLU and another fully connected layer. The learning rate was set to `3e-4`.

For the **linear baseline**, we simply propagated the particles linearly according to the simulation hyperparamaters. The linear baseline can be seen as removing the interactions between particles from the prediction. Any performance improvement beyond the linear baseline can therefore be interpreted as an indication for relational reasoning being performed.

### D.3 QM9

The QM9 regression dataset [19] is a publicly available chemical property prediction task consisting of 134k small drug-like organic molecules with up to 29 atoms per molecule. There are 5 atomic species (Hydrogen, Carbon, Oxygen, Nitrogen, and Flourine) in a molecular graph connected by chemical bonds of 4 types (single, double, triple, and aromatic bonds). 'Positions' of each atom, measured in ångtröms, are provided. We used the exact same train/validation/test splits as Anderson et al. [1] of sizes 100k/18k/13k.

The architecture we used is shown in Table 4. It consists of 7 multihead attention layers interspersed with norm nonlinearities, followed by a TFN layer, max pooling, and two linear layers separated by a ReLU. For each attention layer, shown in Fig. 6, we embed the input to half the number of feature channels before applying multiheaded attention [28]. Multiheaded attention is a variation of attention, where we partition the queries, keys, and values into $H$ *attention heads*. So if our embeddings have dimensionality $(4, 16)$ (denoting 4 feature types with 16 channels each) and we use $H = 8$ attention heads, then we partition the embeddings to shape $(4, 2)$. We then combine each of the 8 sets of shape $(4, 2)$ queries, keys, and values individually and then concatenate the results into a single vector of the original shape $(4, 16)$. The keys and queries are edge embeddings, and thus the embedding matrices are of TFN type (c.f. Eq. (8)). For TFN type layers, the radial functions are learnable maps. For these we used neural networks with architecture shown in Table 5.

For the norm nonlinearities [35], we use

$$\texttt{Norm ReLU}(\mathbf{f}^\ell) = \texttt{ReLU}(\texttt{LN}\left(\|\mathbf{f}^\ell\|\right)) \cdot \frac{\mathbf{f}^\ell}{\|\mathbf{f}^\ell\|}, \qquad \text{where } \|\mathbf{f}^\ell\| = \sqrt{\sum_{m=-\ell}^{\ell} (f_m^\ell)^2}, \tag{52}$$

where LN is layer norm [2] applied across all features within a feature type. For the TFN baseline, we used the exact same architecture but we replaced each of the multiheaded attention blocks with a TFN layer with the same output shape.

The input to the network is a sparse molecular graph, with edges represented by the molecular bonds. The node embeedings are a 6 dimensional vector composed of a 5 dimensional one-hot embedding of the 5 atomic species and a 1 dimension integer node embedding for number of protons per atom. The edges embeddings are a 5 dimensional vector consisting of a 4 dimensional one-hot embedding of bond type and a positive scalar for the Euclidean distance between the two atoms at the ends of the bond. For each regression target, we normalised the values by mean and dividing by the standard deviation of the training set.

We trained for 50 epochs using Adam [11] at initial learning rate `1e-3` and a single-cycle cosine rate-decay to learning rate `1e-4`. The batch size was 32, but for the TFN baseline we used batch size 16, to fit the model in memory. We show results on the 6 regression tasks not requiring thermochemical energy subtraction in Table 3. As is common practice, we optimised architectures and hyperparameters on $\varepsilon_{\text{HOMO}}$ and retrained each network on the other tasks. Training took about 2.5 days on an NVIDIA GeForce GTX 1080 Ti GPU with 4 CPU cores and 15 GB of RAM.

Table 4: QM9 Network architecture: $d_{\text{out}}$ is the number of feature types of degrees $0, 1, ..., d_{\text{out}} - 1$ at the output of the corresponding layer and $C$ is the number of multiplicities/channels per feature type. For the norm nonlinearity we use ReLUs, preceded by equivariant layer norm [33] with learnable affine transform.

| NO. REPEATS | LAYER TYPE | $d_{\text{out}}$ | $C$ |
|---|---|---|---|
| 1x | Input | 1 | 6 |
| 1x | Attention: 8 heads | 4 | 16 |
| | Norm Nonlinearity | 4 | 16 |
| 6x | Attention: 8 heads | 4 | 16 |
| | Norm Nonlinearity | 4 | 16 |
| 1x | TFN layer | 1 | 128 |
| 1x | Max pool | 1 | 128 |
| 1x | Linear | 1 | 128 |
| 1x | ReLU | 1 | 128 |
| 1x | Linear | 1 | 1 |

Table 5: QM9 Radial Function Architecture. $C$ is the number of output channels at each layer. Layer norm [2] is computed per pair of input and output feature types, which have $C_{\text{in}}$ and $C_{\text{out}}$ channels each.

| LAYER TYPE | $C$ |
|---|---|
| Input | 6 |
| Linear | 32 |
| Layer Norm | 32 |
| ReLU | 32 |
| Linear | 32 |
| Layer Norm | 32 |
| ReLU | 32 |
| Linear | $d_{\text{out}} * C_{\text{in}} * C_{\text{out}}$ |

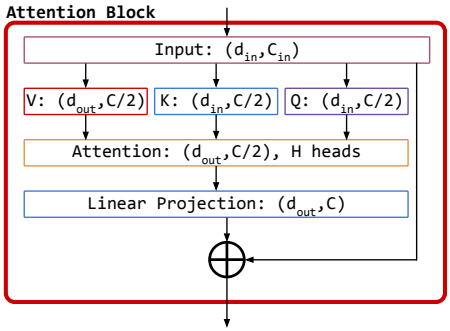



Figure 6: Attention block for the QM9 dataset. Each component is listed with a tuple of numbers representing the output feature types and multiplicities, so $(4, 32)$ means feature types $0, 1, 2, 3$ (with dimensionalities $1, 3, 5, 7$), with 32 channels per type.

### D.4 General Remark

Across experiments on different datasets with the SE(3)-Transformer, we made the observation that the number of representation degrees have a significant but saturating impact on performance. A big improvement was observed when switching from degrees $\{0, 1\}$ to $\{0, 1, 2\}$. Adding type-3 latent representations gave small improvements, further representation degrees did not change the performance of the model. However, higher representation degrees have a significant impact on memory usage and computation time. We therefore recommend representation degrees up to 2, when computation time and memory usage is a concern, and 3 otherwise.

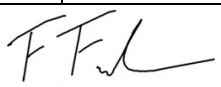
## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

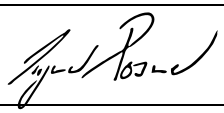
| | |
|---|---|
| Title of Paper | SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks |
| Publication Status | x Published □ Accepted for Publication<br>□Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Fabian B. Fuchs*, Daniel E. Worrall*, Volker Fischer, Max Welling<br>*SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks*<br>Conference on Neural Information Processing Systems (NeurIPS), 2020. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Fabian Fuchs | | |
| Contribution to the Paper | Led the following efforts:<br>- Proposing the project and setting up the collaboration<br>- Implementing the first iteration of the equivariant attention algorithm<br>- Designing, running, and analysing experiments in Sections 4.1 and 4.2<br>- Making and presenting the poster & video presentation<br>Co-led the following efforts:<br>- Designing an equivariant attention algorithm<br>- Writing the paper<br>- Further improving the code and releasing it publicly | | |
| Signature |  | Date | 2021/10/01 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Professor Ingmar Posner | | |
| Supervisor comments<br>This is an accurate description of the candidate's contributions. | | |
| Signature  | Date | 8/10/2021 |

This completed form should be included in the thesis, at the end of the relevant chapter.

# 6

# Iterative SE(3)-Transformers

# 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-$\ell$ 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 $1 \times 1$ 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

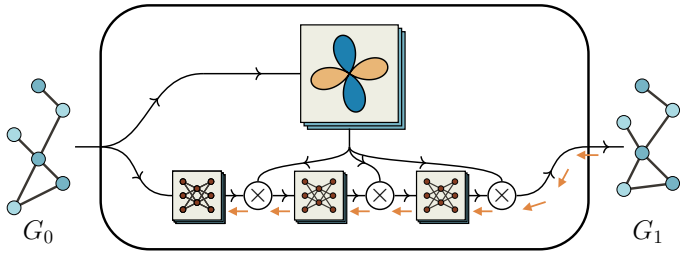



Fig. 1: Gradient flow (orange) in a conventional, single-pass SE(3)-Transformer mapping from a graph (left) to an updated graph (right). The equivariant basis kernels (top) do not have to be differentiated as there is no gradient flow through them.

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

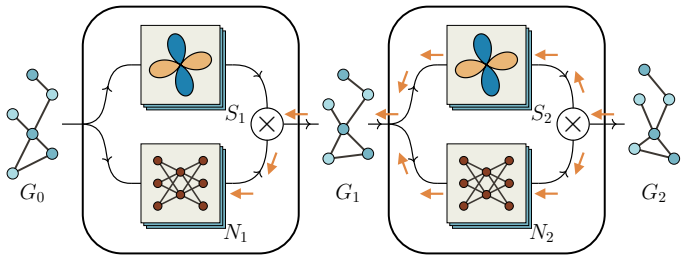



Fig. 2: Gradient flow (orange) in an iterative SE(3)-Transformer with multiple position updates. Now the gradients do flow through the basis construction ($S_2$) meaning this part has to be differentiated.

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

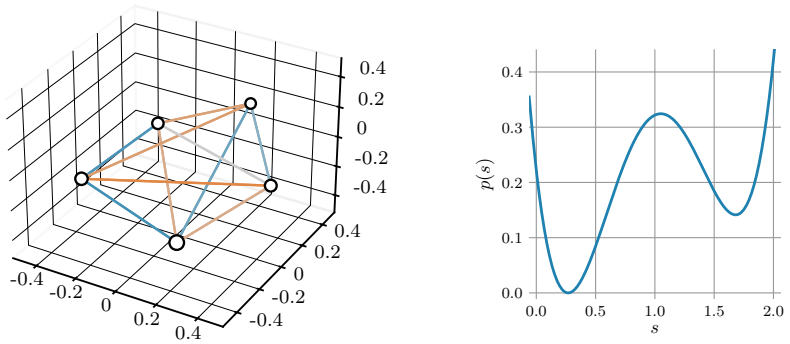



Fig. 3: The left plot shows a configuration of five nodes, with the gradient of the potential between each pair represented by the colour of the edges. Blue edges indicate repulsion and orange edges indicate attraction. Stronger colour represents stronger interaction. The right plot shows the double-minimum 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 ($4 \times 3$ 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 $\sigma$ values in Tables 1 and 2 represent 1-$\sigma$ 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 |
| $\sigma$ | ±0.0001 | ±0.0002 | ±0.0025 | ±0.0011 | ±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 |
| $\sigma$ | ±0.0002 | ±0.0011 | ±0.0003 | ±0.0050 | ±0.0001 | ±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 Georgy Derevyanko, 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.

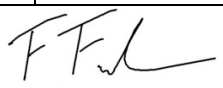
## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

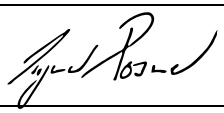
| | |
|---|---|
| Title of Paper | Iterative SE(3)-Transformers |
| Publication Status | x Published □ Accepted for Publication<br>□Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Fabian B. Fuchs*, Edward Wagstaff*, Justas Dauparas, Ingmar Posner<br>*Iterative SE(3)-Transformers*<br>International Conference on Geometric Science of Information, Paris, 2021. (selected for oral presentation) |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Fabian Fuchs | | |
| Contribution to the Paper | Led the following efforts:<br>- Proposing the project and suggesting the collaboration<br>Co-led the following efforts:<br>- Designing, implementing and evaluating the experiments<br>- Designing and implementing the algorithm<br>- Writing the paper<br>- Making the video presentation<br>- Preparing public code release | | |
| Signature |  | Date | 2021/10/01 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Professor Ingmar Posner | | |
| Supervisor comments<br>This is an accurate description of the candidate's contributions. | | |
| Signature  | Date | 8/10/2021 |

This completed form should be included in the thesis, at the end of the relevant chapter.

# 7

# Neural Stethoscopes: Unifying Analytic, Auxiliary and Adversarial Network Probing

# Scrutinizing and De-Biasing Intuitive Physics with Neural Stethoscopes

Fabian B. Fuchs[1]
fabian@robots.ox.ac.uk
Oliver Groth[12]
ogroth@robots.ox.ac.uk
Adam R. Kosiorek[1]
adamk@robots.ox.ac.uk
Alex Bewley*[13]
bewley@google.com
Markus Wulfmeier*[14]
mwulfmeier@google.com
Andrea Vedaldi[2]
vedaldi@robots.ox.ac.uk
Ingmar Posner[1]
ingmar@robots.ox.ac.uk

[1] Applied AI Lab, University of Oxford, UK
[2] Visual Geometry Group, University of Oxford, UK
[3] Google Research, Zürich, Switzerland
[4] Google Deepmind, London, UK

**Abstract**

Visually predicting the stability of block towers is a popular task in the domain of intuitive physics. While previous work focusses on prediction accuracy, a one-dimensional performance measure, we provide a broader analysis of the learned physical understanding of the final model and how the learning process can be guided. To this end, we introduce *neural stethoscopes* as a general purpose framework for quantifying the degree of importance of specific factors of influence in deep neural networks as well as for actively promoting and suppressing information as appropriate. In doing so, we unify concepts from multitask learning as well as training with auxiliary and adversarial losses. We apply neural stethoscopes to analyse the state-of-the-art neural network for stability prediction. We show that the baseline model is susceptible to being misled by incorrect visual cues. This leads to a performance breakdown to the level of random guessing when training on scenarios where visual cues are inversely correlated with stability. Using stethoscopes to promote meaningful feature extraction increases performance from 51% to 90% prediction accuracy. Conversely, training on an easy dataset where visual cues are positively correlated with stability, the baseline model learns a bias leading to poor performance on a harder dataset. Using an adversarial stethoscope, the network is successfully de-biased, leading to a performance increase from 66% to 88%.

* Work done while at Oxford University.

# 1 Introduction

In the deep learning community, intuitive physics describes the concept of training neural networks to solve physics-related tasks in a data-driven as opposed to rule-based manner. However, what kind of function the trained network represents highly depends on the types of scenarios it is confronted with and the task it is trying to solve. Furthermore, it depends on the network architecture, on regularisation techniques, on the training procedure, *etc*. As a result, in contrast to a rule-based approach, it is often hard to assess what form of physical understanding a neural network has developed. We are specifically interested in whether the network uses visual cues as shortcuts which reflect correlations in the dataset but are incommensurate with the underlying laws of physics the network was intended to learn.

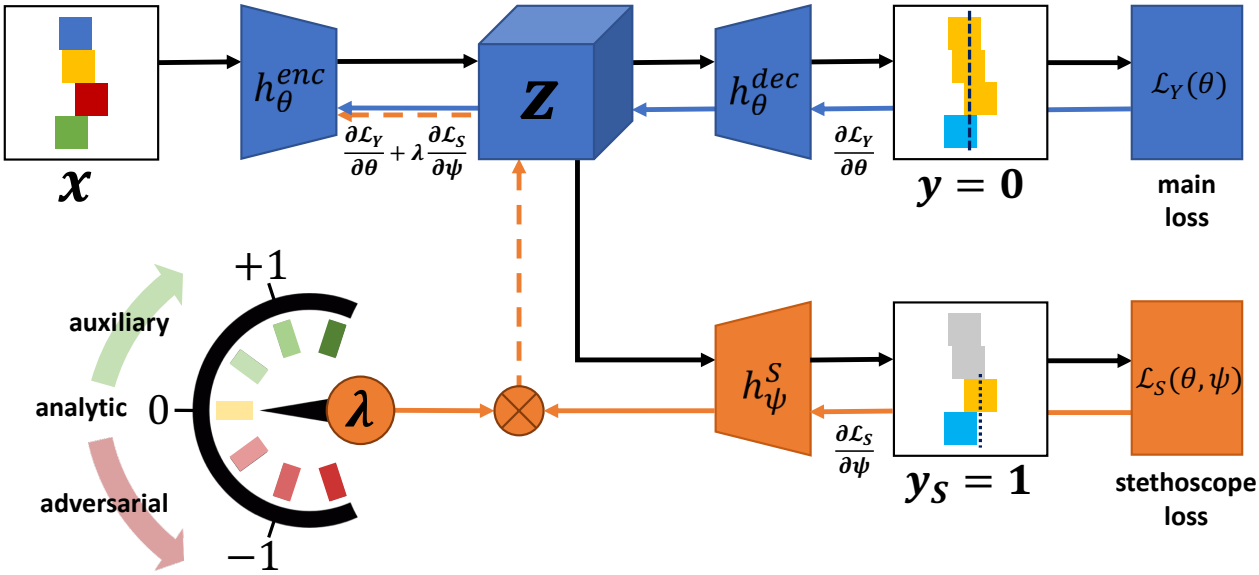



Figure 1: The stethoscope framework. The main network (blue), comprised of an encoder and a decoder, is trained for global stability prediction of block towers. The stethoscope (orange), a two layered perceptron, is trained to predict a nuisance parameter (local stability) where the input is $Z$, an arbitrary layer of the main network. The stethoscope loss is back-propagated with weighting factor $\lambda$ to the main network. The value of $\lambda$ determines whether the stethoscope operates in analytic ($\lambda = 0$), auxiliary ($\lambda > 0$) or adversarial manner ($\lambda < 0$).

In this paper, we specifically focus on stability prediction of block towers, a task which has gained interest in both the deep learning [9, 11, 24] and the robotics community in recent years [12, 13]. Images of towers of blocks stacked on top of each other are shown to a neural network. Its task is to predict whether the tower will fall over or not resulting in a binary classification problem. End-to-end learning approaches as well as simulation-based approaches achieve super-human performance on a real dataset [9, 11, 24]. However, with investigation of trained deep learning models limited to occlusion-based attention analyses [9, 11], it is not clear to what extent neural networks trained on this task take into account physical principles such as centre-of-mass or whether they follow visual cues instead. To this end, we introduce a variation of the ShapeStacks dataset presented by [9] which facilitates the analysis of the effects of visual cues on the learning process.

Motivated by the need for an effective tool to understand and guide the physical reasoning of the neural network and inspired by prior research in interpretability, multi-task learning and adversarial training, we present *neural stethoscopes* as a unified framework for the interrogation and perturbation of task-specific information at any layer. A stethoscope can be

deployed in a purely *analytic* fashion whereby a question is posed via a stethoscope loss which is not propagated back into the main network. It can also be used to promote or suppress specific information by deploying either an auxiliary or an adversarial training mechanism. The concept is illustrated in Figure 1. We demonstrate that deploying an auxiliary stethoscope can be used to promote information conducive to the main task improving overall network performance. Conversely, we show that an adversarial stethoscope can mitigate a specific bias by effectively suppressing information. Moreover, the main network does not need to be changed in order to apply a neural stethoscope.

In this work, we present two contributions: (1) An in-depth analysis of the state-of-the-art approach for intuitive stability prediction. To that end, we also introduce an extension to the existing ShapeStacks dataset which will be made publicly available.[1] (2) A framework for interpreting, suppressing or promoting extraction of features specific to a secondary task unifying existing approaches from interpretability, auxiliary and adversarial learning. While we frame this work in the context of intuitive physics, questions regarding model interpretability and, consequently, systematic, targeted model adaptation find applicability in all domains of deep learning. For a study of two MNIST toy problems with neural stethoscopes, please see Appendix C.

## 2 Related Work

This work touches on two fields within deep learning: intuitive physics, specifically stability prediction, and targeted model adaptation/interpretation.

**Stability Prediction** Vision-based stability prediction of block towers has become a popular task for teaching and showcasing physical understanding of algorithms. Approaches include end-to-end deep learning algorithms [9, 11] as well as pipelines using computer vision to create an abstract state representation which can then be used by a physics simulator [6, 24]. [6] achieve impressive results with a pipeline approach to robotic stone stacking. Interestingly, on a publicly available real-world dataset of block tower images [11], the state-of-the-art is shared between an end-to-end learning approach [9] and a pipeline method using a physics simulator [24]. While being much easier to implement, end-to-end approaches have the downside of being significantly harder to interpret. Interpretability brings at least two advantages: (1) Trust by understanding the model's reasoning and therefore also its potential failure cases. (2) Identification of potentials to improve the model. Both Lerer et al. [11] and Groth et al. [9] conduct occlusion-based attention analyses. Groth et al. [9] find that the focus of the algorithm's attention lies within a bounding box around the stability violation in 80% of the cases. While encouraging, conclusions which can be drawn regarding the network's understanding of physical principles are limited. Moreover, Selvaraju et al. [18] shows that attention analyses can be misleading: The Grad-CAM visualisation does not change even for artificially crafted adversarial examples which maximally confuse the classifier.

**Neural Stethoscopes** The notion of passively interrogating and actively influencing feature representations in hidden layers of neural networks connects disparate fields including interpretability, auxiliary losses and multitask learning as well as adversarial training. We note that much of the required machinery for neural stethoscopes already exists in a number of sub-domains of deep learning. Mirowski et al. [17] use a small probing network to predict

[1] Dataset available at https://shapestacks.robots.ox.ac.uk/

an agent's position from a latent space for analytical purposes without backpropagating to the main network. Work on auxiliary objectives [10, 17] as well as multi-task learning (*e.g.* Caruana [4, 5]) commonly utilises dedicated modules with losses targeted towards a variety of tasks in order to optimise a representation shared across tasks. Based on this notion both *deep supervision* [23] and *linear classifier probes* [2] reinforce the original loss signal at various levels throughout a network stack. Although their work is restricted to reinforcing the learning signal via the same loss applied to the global network [2] in particular demonstrate that the accessibility of information at a given layer can be determined - and promoted - by formulating a loss applied locally to that layer.

Conversely, in order to encourage representations invariant to components of the input signal, regularisation techniques are commonly utilised (*e.g.* Srivastava et al. [19]). To obtain invariance with respect to known systematic changes between training and deployment data, [7, 25] propose methods for domain adaptation. In order to prevent a model fitting to a specific nuisance factor, Louizos et al. [14] minimise the Maximum Mean Discrepancy between conditional distributions with different values of a binary nuisance variable in the conditioning set. This method is limited to discrete nuisance variables and its computational cost scales exponentially with the number of states. Xie et al. [26] address both issues via adversarial training, optimising an encoding to confuse an additional discriminator, which aims to determine the values of nuisance parameters. This approach assumes that the nuisance variable is known and is an input to the main model during training and deployment. Louppe et al. [15] follow a similar approach, applying the discriminator to the output of the main model instead of its intermediate representations.

## 3 Neural Stethoscopes

We use stethoscopes to analyse and influence the learning process for stability prediction, but present it in the following as a general framework which can be applied to any set of tasks.

In supervised deep learning, we typically look for a function $f_\theta : X \to Y$ with parameters $\theta$ that maps an input $\boldsymbol{x} \in X$ to its target $\boldsymbol{y} \in Y$. Often the function internally computes one or more intermediate representations $\boldsymbol{z} \in Z$ of the data. In this case, we rewrite $f_\theta$ as the composition of the encoder $h^{\text{enc}}_\theta : X \to Z$, which maps the input to the corresponding features $\boldsymbol{z} \in Z$, and the decoder $h^{\text{dec}}_\theta : Z \to Y$, which maps features to the output.

Let the stethoscope be defined as an arbitrary function $h^{\text{s}}_\psi : Z \to S$ with parameters $\psi$. $h^{\text{s}}_\psi$ is trained on a supplemental task with targets $s \in S$ and not for the main objective. The loss function for the stethoscope is defined as $\mathcal{L}_s(\theta, \psi)$, which measures the performance on the supplemental task. The weights of the stethoscope are updated as

$$-\Delta\psi \propto \nabla_\psi \mathcal{L}_s(\theta, \psi) \tag{1}$$

to minimise $\mathcal{L}_s(\theta, \psi)$.

The loss function measuring the performance on the main task, i.e., the discrepancy between predictions $f_\theta$ and the true task $y$, is defined as $\mathcal{L}_y(\theta)$. Crucially, the weights of the main network are updated not just as a function of $\mathcal{L}_y$, but also of $\mathcal{L}_s$:

$$\mathcal{L}_{y,s}(\theta, \psi) = \mathcal{L}_y(\theta) + \lambda \cdot \mathcal{L}_s(\theta, \psi). \tag{2}$$

$$-\Delta\theta \propto \nabla_\theta \mathcal{L}_{y,s}(\theta, \psi) \tag{3}$$

By choosing different values for the constant $\lambda$ we obtain three very different use cases:
**Analytic Stethoscope** ($\lambda = 0$) Here, the gradients of the stethoscope, which acts as a passive observer, are not used to alter the main model. This setup can be used to interrogate learned feature representations: if the stethoscope predictions are accurate, the features can be used to solve the task.
**Auxiliary Stethoscope** ($\lambda > 0$) The encoder is trained with respect to the stethoscope objective, hence enforcing correlation between main network and supplemental task. This setup is related to learning with auxiliary tasks, and helpful if we expect the two tasks to be beneficially related.
**Adversarial Stethoscope** ($\lambda < 0$) By setting $\lambda < 0$, we train the encoder to maximise the stethoscope loss (which the stethoscope *still* tries to minimise), thus encouraging independence between main network and supplemental tasks. This is effectively an adversarial training framework and is useful if features required to solve the stethoscope task are a detrimental nuisance factor.

For the analytic stethoscope, to fairly compare the accessibility of information with respect to a certain task in different feature representations, we set two criteria: (1) The capacity of the stethoscope architecture has to be constant regardless of the dimensions of its input. (2) The stethoscope has to be able to access each neuron of the input separately. We guarantee this by fully connecting the input with the first layer of the stethoscope using a sparse matrix. This matrix has a constant number of non-zero entries (criterion 1) and connects every input unit as well as every output unit at least once (criterion 2). For more details, see Appendix A.

In auxiliary and adversarial mode, we attach the stethoscope to the main network's last layer before the logits in a fully connected manner. This setup proved to have the highest impact on the main network. The stethoscope itself is implemented as a two-layer perceptron with ReLU activation and trained with sigmoid or softmax cross-entropy loss on its task $\mathcal{S}$.

For numerical stability, the loss of the encoder in the adversarial setting is rewritten as

$$\tilde{\mathcal{L}}_{y,s}(\theta, \psi) = \mathcal{L}_y(\theta) + |\lambda| \cdot \mathcal{L}_{\bar{s}}(\theta, \psi) \tag{4}$$

where $\mathcal{L}_{\bar{s}}(\theta, \psi)$ is the stethoscope loss with flipped labels. The objective is similar to the confusion loss formulation utilised in GANs to avoid vanishing gradients when the discriminator's performance is high [8].

# 4 Vision-Based Stability Prediction of Block Towers

Previous work has shown that neural networks are highly capable of learning physical tasks such as stability prediction. However, unlike approaches using physics simulators [6, 24], with pure-learning based approaches, it is hard to assess what reasoning they follow and whether they gain a sound understanding of the physical principles or whether they learn to take short-cuts following visual cues based on correlations in the training data.

In this section, we follow the state-of-art approach on visual stability prediction of block towers and examine as well as influence its learning behaviour. We introduce a variation of the ShapeStacks dataset from [9] which is particularly suited to study the dependence of network predictions on visual cues. We examine how suppressing or promoting the extraction of certain features influences the performance of the network using neural stethoscopes.

**Dataset** As shown in [9], a single-stranded tower of blocks is stable if, and only if, at every interface between two blocks the centre of mass of the entire tower above is supported by the convex hull of the contact area. If a tower satisfies this criterion, *i.e.*, it does not collapse, we

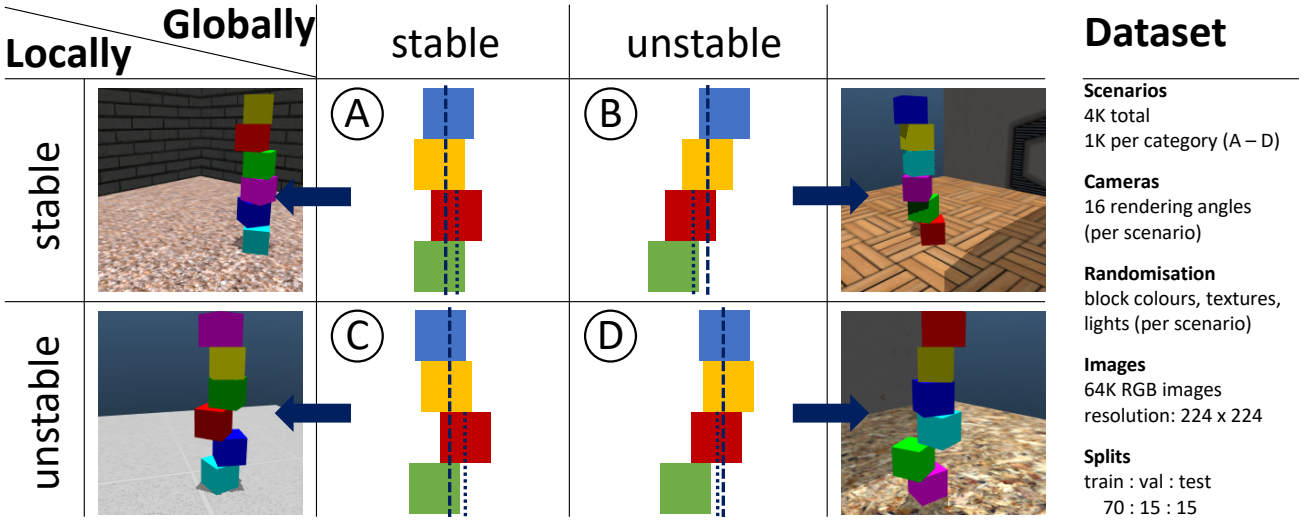



Figure 2: We have four qualitative scenarios: (A) Globally stable towers which also do not exhibit any local stability violation. (B) Towers without any local instability which are globally unstable because of the skew. (C) Towers in which one local stability violation is counterbalanced by blocks above. (D) Globally unstable towers where the site of local and global violation is perfectly correlated. The dashed line shows the projection of the cumulative centre of mass of the upper tower (red, yellow and blue block) whereas the dotted line depicts the projection of the local centre of mass of the red block. A tower is globally stable, if and only if the global centre of mass is always supported whereas the individual local centre of masses are not indicative of global structure stability. Global and local centre of masses for the green, yellow and blue block have been omitted for clarity of presentation.

call it *globally stable*. To be able to quantitatively assess how much the algorithm follows visual cues, we introduce a second label: We call a tower *locally stable* if, and only if, at every interface between two blocks, the centre of mass of the block immediately above is supported by the convex hull of the contact area. Intuitively, this measure describes, if taken on its own without any blocks above, each block would be stable. We associate binary prediction tasks $y_G$ and $y_L$ to respective global and local stability where label $\boldsymbol{y} = 0$ indicates *stability* and $\boldsymbol{y} = 1$ *instability*. Global and local instability are neither mutually necessary nor sufficient, but can easily be confused visually which is demonstrated by our experimental results. Based on the two factors of local and global stability, we create a dataset[2] with 4,000 block tower scenarios divided into four qualitative categories (cf. Figure 2). The dataset is divided into an *easy* subset, where local and global stability are always positively correlated, and a *hard* subset, where this correlation is always negative. The dataset will be made available online.[3]

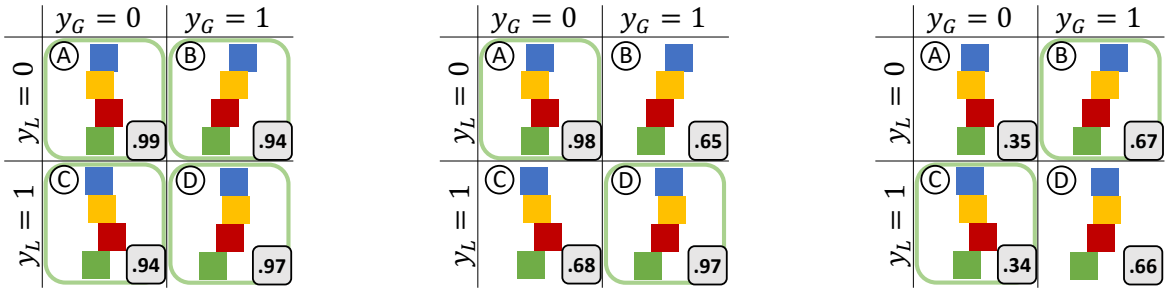
**Model** We choose the Inception-v4 network [20] as it yields state-of-the-art performance on stability prediction [9]. The model is trained in a supervised setting using example tuples $(\boldsymbol{x}, \boldsymbol{y}_G, \boldsymbol{y}_L)$ consisting of an image $\boldsymbol{x}$ and its global and local stability labels $\boldsymbol{y}_G$ and $\boldsymbol{y}_L$. Classification losses use sigmoid cross entropy. We use the RMSProp optimiser [21] throughout all our experiments.[4]

**Local Stability as a Visual Cue** Based on the four categories of scenarios described in Figure 2, we conduct an initial set of experiments to gauge the influence of local stability on the network predictions. If local stability had, as it would be physically correct, no influence on the network's prediction of global stability, the performance of the network should be

[2] We use the MuJoCo physics engine [22] for rendering and stability checking.

[3] Dataset available at https://shapestacks.robots.ox.ac.uk/

[4] We use the Tensorflow [1] implementation of RMSProp without momentum, gradient history decay of 0.9, epsilon of 1.0, a learning rate of 0.045 and a learning rate decay of 0.975 after every epoch.




(a) Trained on All: $\varnothing_{acc} = 0.96$ (b) Trained on Easy: $\varnothing_{acc} = 0.82$ (c) Trained on Hard: $\varnothing_{acc} = 0.51$

Figure 3: The influence of local instability on global stability prediction. In setup (a) we train on all 4 tower categories (indicated by green frames). Global stability prediction accuracies on per-category test splits are reported in the bottom right grey boxes. In (b) we train solely on easy scenarios (A & D) where global and local stability are positively correlated. In (c) we only present hard scenarios during training featuring a negative correlation between global and local stability. The performance differences clearly show that local stability influences the network's prediction for global stability.

equal for *easy* and *hard* scenarios, regardless of the training split. However, Figure 3 shows a strong influence of local stability on the prediction performance. When trained on the entire, balanced data set, the error rate is three times higher for hard than for easy scenarios (6% vs. 2%). When trained on easy scenarios only, the error rate even differs by a factor of 13. Trained on hard scenarios only, the average performance across all four categories is on the level of random chance (51%), indicating that negatively correlated local and global stability imposes a much harder challenge on the network.

# 5 Using Neural Stethoscopes to Guide the Learning Process

After demonstrating the influence of local stability on global stability prediction we turn our attention to the use of neural stethoscopes to quantify and actively mitigate this influence.

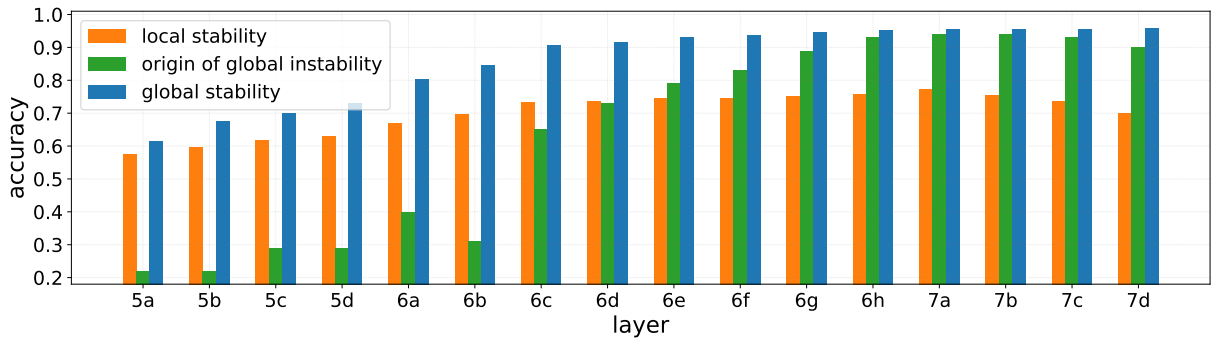



Figure 4: Analysis of the prediction performance throughout the Inception-v4 network trained on predicting global stability. We report average performances on balanced test data after 50 epochs of stethoscope training. All stethoscopes have been attached to the respective activation tensors[5] with sparse connection matrices as described in Section 3.

**Task Relationship Analysis** We seek to quantify the correlation of features extracted for global stability prediction with the task of local instability detection. We train the main

network on global stability prediction while attaching stethoscopes to multiple layers. The stethoscope is trained on three tasks: global stability prediction (binary), local stability (binary) and origin of global instability, particularly the interface at which this occurs (n-way one-hot). Figure 4 reveals that the network gradually builds up features which are conducive to both global and local stability prediction. However, the performance of local stability prediction peaks at the activation tensor after layer 7a whereas the performance for global stability prediction (both binary and n-way) keeps improving. This is in line with the governing physical principles of the scenarios and yields the conjecture that the information about local stability can be considered a *nuisance factor* whereas information about the global site of stability violation can serve as a *complementary factor* for the main task.

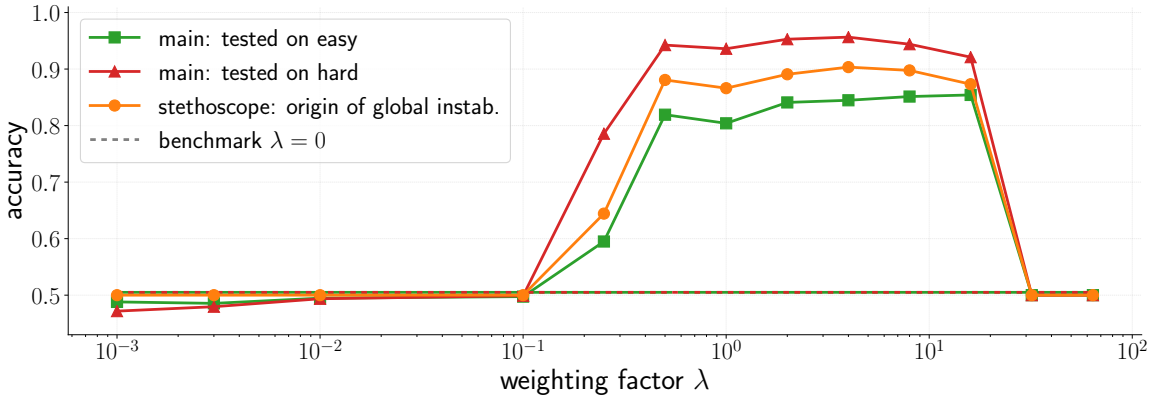


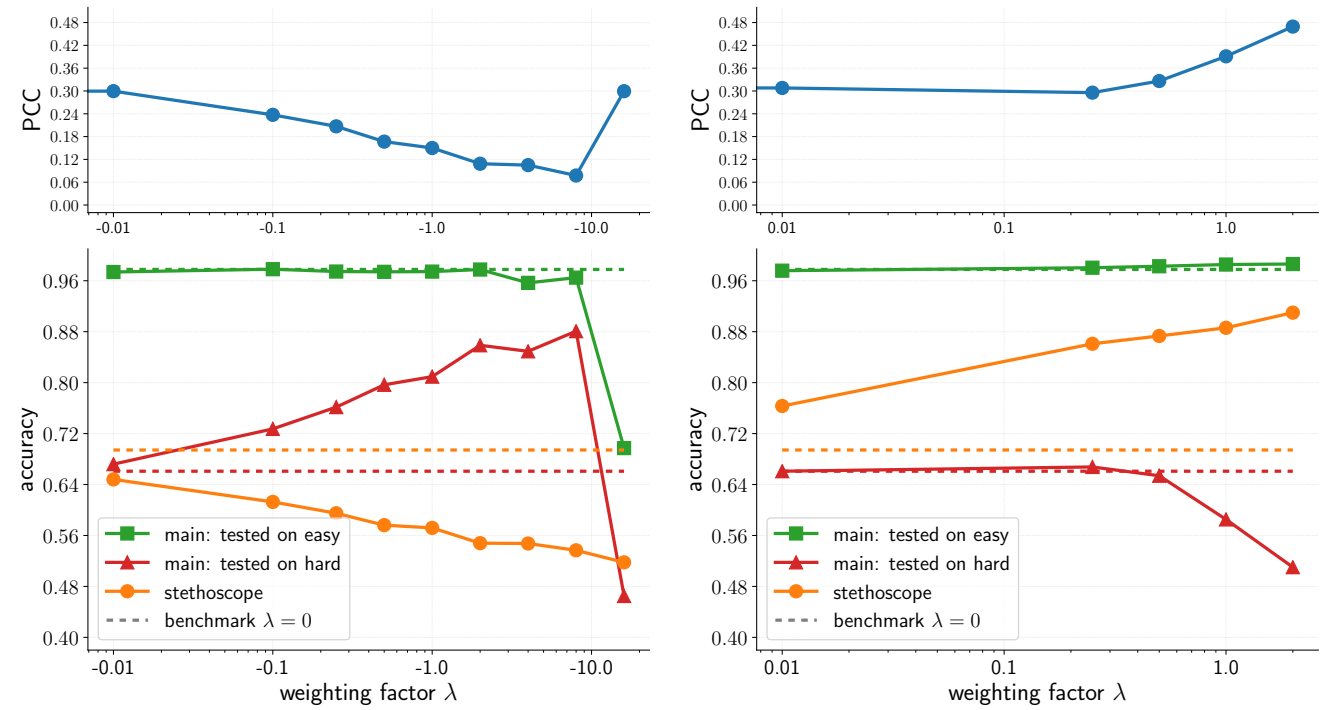

Figure 5: Performance gains by promoting complementary feature extraction with auxiliary stethoscopes. The main network is trained on binary global stability labels while the stethoscope is trained on more fine grained labels, origin of global stability (n-way). The network was trained on *hard* scenarios only but evaluated on all. Dashed lines are baselines ($\lambda = 0$).

**Promotion of Complementary Information** We now test the hypothesis that fine-grained labels of instability locations help the main network to grasp the correct physical concepts. To that end, we consider the setup from Figure 3c where the training data only consists of *hard* scenarios with a baseline performance of 51%. The main network is trained on global stability while the stethoscope is trained on predicting the origin of global instability, namely the interface at which the instability occurs. Figure 5 shows that auxiliary training substantially improves the performance for weighting parameters $\lambda \in [0.5, 16]$. However, for very small values of $\lambda$, the contribution of the additional loss term is too small while for large values, performance deteriorates to the level of random chance as a result of the primary task being far out-weighted by the auxiliary task. Note that it is to be expected that performance on *hard* exceeds performance on *easy* as the latter type is not encountered during training.

**Suppression of Nuisance Information** Results from Figure 3 indicate that the network might use local stability as a visual cue to make biased assumptions about global stability. We now investigate whether it is possible to debias the network by forcing it to pay less attention to local stability. To that end, we focus on the scenario shown in Figure 3b, where we only train the network on global stability labels for *easy* scenarios. As shown in Figure 3b, the performance of the network suffers significantly when tested on *hard* scenarios where local and global stability labels are inversely correlated.

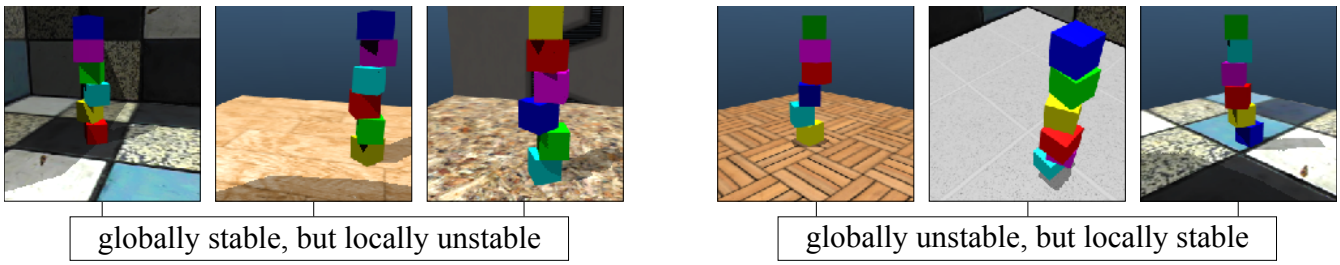
The hypothesis is that forcing the network not to focus on local stability weakens this bias.

[5]The stethoscopes are connected to outputs of inception and reduction modules of the Inception v4 network, cf. [20]. Note that this analysis is specific to this architecture and could yield very different results, *e.g.* for networks with residual connections.




(a) Adversarial Stethoscope

(b) Auxiliary Stethoscope




(c) Success cases where algorithm only predicted correctly after de-biasing ($\lambda = -8.0$).

Figure 6: Successful debiasing by suppressing a nuisance factor with adversarial training while auxiliary training worsens the bias. The main network is trained on global stability while the supplementary task is to predict presence of local instabilities. Training data for global stability only comprises *easy* scenarios while training data for the supplementary task comprises both *easy* and *hard* scenarios. **(a)** & **(b)** Green squares and red triangles show accuracies of the main network when predicting global stability of block towers for *easy* scenarios and *hard* scenarios, respectively. Orange circles depict the performance of the stethoscope on the task of local stability. Blue circles represent the Pearson Correlation Coefficient of predicted global stability and ground truth local stability which gives an indication of how much the network follows visual cues. **(c)** Images show *hard* scenarios which the algorithm classified correctly only when adversarial training was used.

In Figure 6, we use active stethoscopes ($\lambda \neq 0$) to test this hypothesis. We train a stethoscope on local stability on labels of all categories (in a hypothetical scenario where local labels are easier to obtain than global labels) and use both the adversarial and the auxiliary setup in order to test the influence of suppressing and promoting accessibility of information relevant for local stability in the encoded representation, respectively. In Figure 6, the results of both adversarial and auxiliary training are compared to the baseline of $\lambda = 0$.

Figure 6a shows that adversarial training does indeed partly remove the bias and significantly increases the performance of the main network on *hard* scenarios while maintaining its high accuracy on *easy* scenarios. The bias is removed more and more with increasing magnitude of the weighting factor $\lambda$ up to a point where further increasing $\lambda$ jeopardises

the performance on the main task as the encoder puts more focus on confusing the stethoscope than on the main task (in our experiments this happens at $\lambda \approx 10^1$). Interestingly, the correlation of local stability and global stability prediction rises at this point as for pushing the stethoscope below 50% (random guessing), the main network has to extract information about local stability. Naturally, the performance of the stethoscope continuously decreases with increasing $\lambda$ as the encoder puts more and more focus on confusing the stethoscope.

This scenario could also be seen from the perspective of feeding additional information into the network, which could profit from more diverse training data. However, Figure 6b shows that naively using an auxiliary setup to train the network on local stability worsens the bias. With increasing $\lambda$ and increasing performance of the stethoscope, the network slightly improves its performance on *easy* scenarios while accuracy deteriorates on *hard* scenarios. These observations are readily explained: local and global stability are perfectly correlated in the *easy* scenarios, *i.e.*, the (global) training data. The network is essentially free to chose whether to select local or global features when predicting global stability. Auxiliary training on local stability further shifts the focus to local features. When tested on *hard* scenarios, where local and global stability are inversely correlated, the network will therefore perform worse when it has learned to rely on local features.

# 6 Conclusion

We study the state-of-the-art approach for stability prediction of block towers and test its physical understanding. We create a new dataset and introduce the framework of *neural stethoscopes* unifying multiple threads of work in machine learning related to analytic, auxiliary and adversarial probing of neural networks. The analytic application of stethoscopes allows measuring relationships between different prediction tasks. We show that the network trained on stability prediction also obtains a more fine-grained physical understanding of the scene (origin of instability) but at the same time is susceptible to potentially misleading visual cues (*i.e.*, local stability). In addition to the analysis, the auxiliary and adversarial modes of the stethoscopes are used to support beneficial complementary information (origin of instability) and suppress harmful nuisance information (visual cues) without changing the network architecture of the main predictor. This yields substantial performance gains in unfavourable training conditions where data is biased or labels are partially unavailable. We encourage the use of neural stethoscopes for other application scenarios in the future as a general tool to analyse task relationships and suppress or promote extraction of specific features. This can be done by collecting additional labels or using existing multi-modal data.

# Acknowledgements

This research was funded by the EPSRC AIMS Centre for Doctoral Training at Oxford University and the European Research Council under grant ERC 677195-IDIU. We thank Adrian Penate-Sanchez, Oliver Bartlett and Triantafyllos Afouras for proofreading.

We acknowledge use of Hartree Centre resources in this work. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. The Centre aims to develop and demonstrate next generation software, optimised to take advantage of the move towards exa-scale computing.

# References


[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (ICLR) Workshop*, 2016.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *International Conference on Machine Learning*, 2017.

[4] Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems*, 1995.

[5] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[6] Fadri Furrer, Martin Wermelinger, Hironori Yoshida, Fabio Gramazio, Matthias Kohler, Roland Siegwart, and Marco Hutter. Autonomous robotic stone stacking with online next best object target pose planning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2350–2356, 2017.

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[8] Ian Goodfellow. Tutorial: GANs. *arXiv*, 2016. ISSN 0253-0465. doi: 10.1001/jamainternmed.2016.8245.

[9] Oliver Groth, Fabian Fuchs, Ingmar Posner, and Andrea Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. *ECCV*, 2018.

[10] Max Jaderberg, Volodymyr Mnih, Wojciech Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016.

[11] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning - Volume 48*, ICML, 2016.

[12] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[13] Wenbin Li, Jeannette Bohg, and Mario Fritz. Acquiring Target Stacking Skills by Goal-Parameterized Deep Reinforcement Learning. pages 1–10, 2017.

[14] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. The variational fair autoencoder. *ICLR*, 2015.

[15] Gilles Louppe, Michael Kagan, and Kyle Cranmer. Learning to pivot with adversarial networks. In *NIPS*, 2017.

[16] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems 30*. 2017.

[17] Piotr W. Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. *ICLR*, 2017.

[18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[19] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[20] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[21] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[22] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. *IEEE International Conference on Intelligent Robots and Systems*, 2012.

[23] Liwei Wang, Chen-Yu Lee, Zhuowen Tu, and Svetlana Lazebnik. Training deeper convolutional networks with deep supervision. *arXiv*, 1505.02496, 2015.

[24] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems 30*. 2017.

[25] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Addressing appearance change in outdoor robotics with adversarial domain adaptation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2017.

[26] Qizhe Xie, Zihang Dai, Yulun Du, Eduard H. Hovy, and Graham Neubig. Controllable invariance through adversarial feature learning. In *NIPS*, 2017.

# A Connection of Stethoscope to Main Network

This is a detailed explanation of how to connect the first layer of the stethoscope with any layer of the main network as described in Section 3. The goal is to guarantee equal capacity of the stethoscope regardless of the dimensions of the layer it is connected to in order to obtain comparable results. Let's call the flattened feature vector from the main network $\vec{x}$. In the following we describe how to compute the sparse matrix $\mathbf{M}$ which reduces the full vector $\vec{x}$ to a smaller vector $\vec{x'}$ while adding biases:

$$\vec{x'} = \mathbf{M} \cdot \vec{x} + \vec{b} \tag{5}$$

$\mathbf{M}$ is a sparse matrix with a constant number of non-zero entries (*n_non_zero*) which are trainable. The hyperparameter *n_non_zero*, which determines the capacity of this additional linear layer, is chosen as an integer multiple of the length of $\vec{x'}$ and has to be greater or equal than the number of features in any of the layers of the main network which the stethoscope is attached to. The matrix $\mathbf{M}$ is generated in a way so that it fulfils the following two guarantees:

- Each input is connected at least once. *I.e.*, no column only contains zeroes.
- Each output is connected an equal amount of times. *I.e.*, all rows have the same number of non-zero entries.

# B Appendix: Is Information Discarded in Adversarial Training?

Figure 6a shows that adversarial training increases performance given a biased training dataset. While hypothetically the network could learn to completely discard information about local stability when trying to confuse the adversarial stethoscope, it could also simply reduce its accessibility.

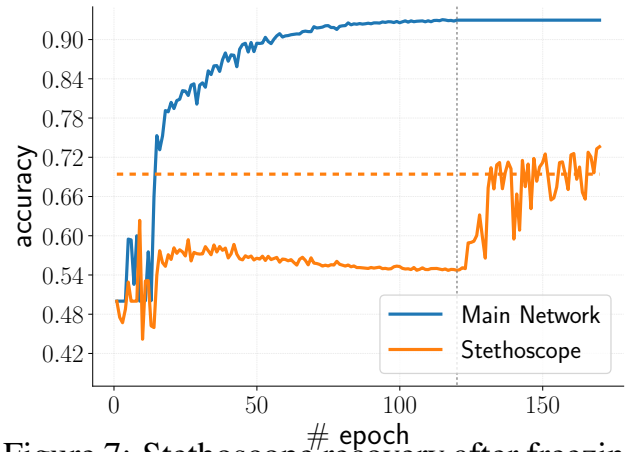
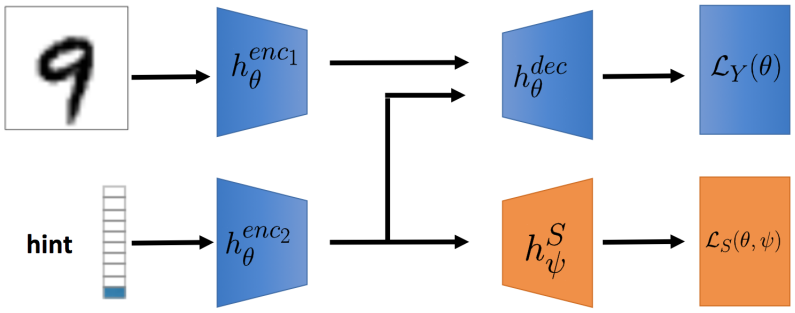



Figure 7: Stethoscope recovery after freezing the adversarially trained main network.

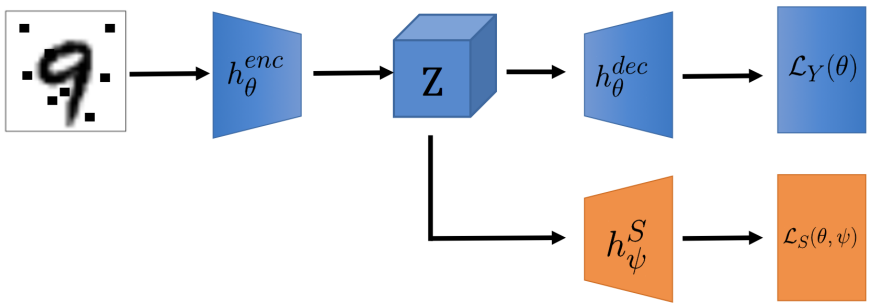
In an additional experiment (Figure 7), where the main network (blue line) is fixed after adversarial training for 120 epochs (grey dashed line), the performance of the stethoscope on local stability (orange) is fully recovered to the level of the baseline for a purely analytic stethoscope (dashed orange line). We therefore argue that the information is not removed, but instead made less accessible to the stethoscope by constantly shifting the representation as explained in [16]. However, the resulting performance increase for the main task indicates that this behaviour also makes information related to local stability less accessible to the decoder of the main network. Hence, although the adversarial setup does not reach final, stable equilibria (*cf.* [16]), it decreases the dependence of the decoder on features particular to local stability. Hence, further improvements for stabilising GAN training could additionally improve performance in adversarial stethoscope use-cases [3, 8, 16].

# C Appendix: Suppressing Nuisance Information in MNIST

This section introduces two intuitive examples and serves to illustrate the challenge with biased training datasets. We conduct two toy experiments built on the MNIST dataset to demonstrate the issue of varying correlation between nuisance parameters and ground truth labels in training and test datasets and to demonstrate the use of an adversarial stethoscope to suppress this information and mitigate overfitting. Both experiments build on the provision of additional *hints* as network input. In the first experiment, the hint is a separately handled input in the form of a one-hot vector whereas in the second experiment, the hint is directly incorporated in the pixel space of the MNIST images.




(a) First Toy Example: Hint as Separate One-Hot Vector

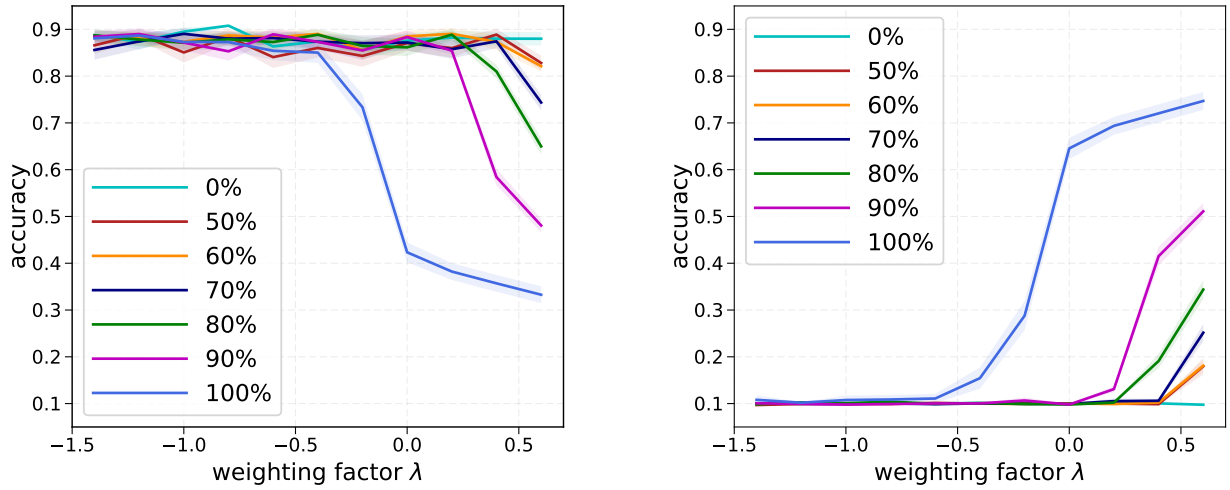



(b) Second Toy Example: Hint as Pixel Encoding

Figure 8: Setup of the toy experiments using the MNIST dataset with artificial hints. In (a), the hints are fed to the main network via a separate input and both input streams are encoded separately by neural networks. The stethoscope only sees the output of the hint encoder $h_\theta^{enc_2}$ while the decoder $h_\theta^{dec}$, which is trained on digit classification, sees the output of both encoders. In (b), the hints are incorporated as pixel modifications into the main images. The encoder $h_\theta^{enc}$ transforms the input into a latent representation Z which is then accessed by both the stethoscope $h_S^\Psi$ and the decoder $h_\theta^{dec}$.

In both experiments, the network is trained on the loss as defined in Equation (2) and the gradients are used to update the weights of the different parts of the network (encoder(s), decoder, stethoscope) as defined in Section 3. Hence, $\lambda < 0$ denotes adversarial training and $\lambda > 0$ auxiliary training.

**Hint as Separate One-Hot Vector** In the first example, the hint labels have the same format as the true labels of the input image and we run experiments with varying correlation between hint and true label in the training set. Given high correlation, the network can learn an identity transformation between hint and output and ignore the image. Now, at




(a) (b)

Figure 9: Influence of adversarial training on classifier performance for toy experiment variant 1 described in Figure 8a. Each curve represents the results for different hint qualities $h_q$. The bold lines indicate the mean of 100 runs and the shaded areas mark the student-t 1 $\sigma$ confidence intervals. In (a) the network is evaluated against the ground truth. In (b), the output of the network is compared to the hint labels during test time. Hence, a high accuracy in (b) shows strong overfitting while a high accuracy in (a) shows that the network learned the actual task of classifying digits from images.

test time, the hint is completely independent of the true label and hence contains no useful information. The experiment mimics challenges with biased datasets as the supplemental task $s \in \mathcal{S}$ (see Section 3) has labels $\boldsymbol{y_s}$ which are correlated to the labels $\boldsymbol{y}$ of the main objective during training but not at test time ($p_{train}(\boldsymbol{y_s}|\boldsymbol{y}) \neq p_{test}(\boldsymbol{y_s}|\boldsymbol{y})$). Hence, an approximation of the correlations in the training data will not generalise to data during deployment.

To that end, we introduce a parameter $q_h$ denoting the quality of the hints (i.e. their correlation with the true labels) during training time where for $q_h = 1.0$, the hints are always correct (during training), for $q_h = 0.5$, the hints are correct for 50% of the training examples and so on. Varying the hint quality enables us to investigate biases of increasing strength.

In this setup (Figure 8a), both input streams are separated via independent encoders. Let $h_\theta^{enc_1}$ and $h_\theta^{enc_2}$ respectively denote image and hint encoders. The stethoscope only accesses the hint encoding in this example, which simplifies its task and enables us to demonstrate the effect of active stethoscopes without affecting the image encoding. The hint encoder multiplies the one-hot hint vector with a scalar weight. Hence, a weight close to zero would effectively hide the hint information from the main network, explicitly making the task of the stethoscope $h_S^\Psi$ more difficult. The image encoder consists of two convolutional layers followed by two fully connected layers with 256 units where each layer is followed by leaky ReLU activations. The decoder $h_\theta^{dec}$, which acts as a digit classifier, is comprised of a single fully connected layer and receives the image encoding concatenated with the encoded hint vector as input.

With adversarial training of the stethoscope, the encoder learns to suppress information about the hints forcing the classifier to purely rely on image data. As can be observed in Figure 9b, for perfect hints ($q_h = 1.0$) and no adversarial training, the decoder is highly accurate in predicting the hint (instead of the actual label), suggesting that the classifier is

effectively independent of the input image and purely relies on the hint. With adversarial training, however, we can indeed force the encoder to hide the hint, therefore forcing it to learn to classify digits from images. As expected, with increasing hint quality, we need to increase the weight of adversarial training so as to discourage the encoder from relying on the hint.

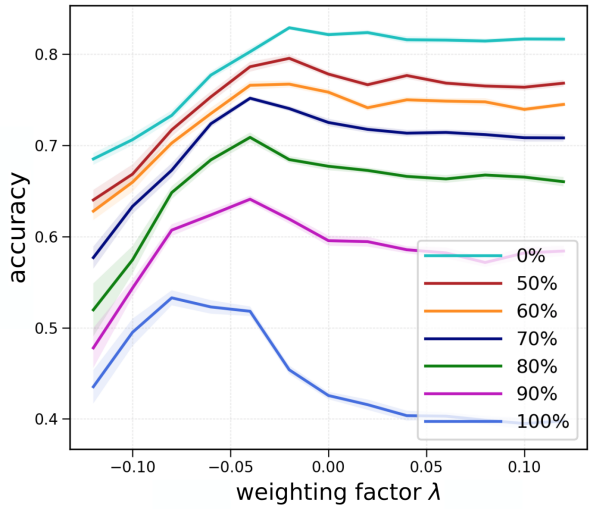



Figure 10: Influence of adversarial training on classifier performance for the second MNIST experiment. The adversarial training is less effective as both the stethoscope and the main network directly access a single encoding. High adversarial weights can strongly degrade performance on the main task as a trivial solution would be to suppress all information about the input. Each curve represents the results for different hint qualities $h_q$. The bold lines indicate the mean of 20 runs and the shaded areas mark the student-t 1 $\sigma$ confidence intervals.

**Hint as Pixel Encoding** In the second variant of the MNIST experiment, hints are provided at training time as a set of high intensity pixels in the input images as opposed to the explicit concatenation in the previous example. This better reflects the typical scenario where the goal is to disentangle the nuisance hints from the informative information related to the main task. Here, the main network is a two-layer multi-layer perceptron (MLP) with the stethoscope attached after the first hidden layer. The simpler architecture (compared to the one used in the first toy problem) was chosen in order to lower the performance on vanilla MNIST classification in order to see clearer effects. Note that in this setup, giving the same labels to the hints as for the main task in the training scenario makes the two tasks indistinguishable (in particular for perfect hint quality $h_q = 1.0$). We therefore introduce a higher number of hint classes with 100 different hints, each of them related to a single digit in a many-to-one mapping, such that theoretical suppression of hint information is possible without degrading main performance.

Figure 10 shows the digit classification accuracy for the main network where the stethoscope is applied with weights ranging in both the positive and negative directions to reflect auxiliary and adversarial use of the stethoscope respectively. When using a positive stethoscope loss, which encourages $h_\theta^{enc}$ to focus more on the artificial hints over the actual digit features, the test accuracy degrades. This is expected for this toy example as the hints have zero correlation with the digit labels at test time. In the negative case, $h_\theta^{enc}$ serves as an adversary to the stethoscope effectively benefiting the main objectives by removing the misleading hint information leading to an improved test accuracy. However, as the magnitude of the adversarial strength increases the gains quickly erode as the encoder begins to suppress information relevant for the main task.

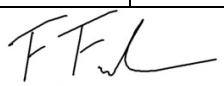
## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

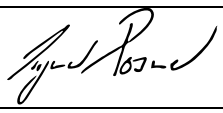
| | |
|---|---|
| Title of Paper | Neural Stethoscopes: Unifying Analytic, Auxiliary and Adversarial Network Probing |
| Publication Status | x Published □ Accepted for Publication<br>□Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Fabian B. Fuchs, Oliver Groth, Adam R. Kosiorek, Alex Bewley, Markus Wulfmeier, Andrea Vedaldi, Ingmar Posner<br>*Neural Stethoscopes: Unifying Analytic, Auxiliary and Adversarial Network Probing*<br>British Machine Vision Conference (BMVC), 2019. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Fabian Fuchs | | |
| Contribution to the Paper | Led the following efforts:<br>- Conception of the idea (stethoscopes & application to intuitive physics)<br>- Designing and analysing experiments<br>- Writing the paper<br>- Making and presenting the poster<br>Co-led the following efforts:<br>- Implementation of the algorithm and running experiments | | |
| Signature  | | Date | 2021/10/07 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Professor Ingmar Posner | | |
| Supervisor comments<br>This is an accurate description of the candidate's contributions. | | |
| Signature  | Date | 8/10/2021 |

This completed form should be included in the thesis, at the end of the relevant chapter.

# 8 Discussion

The guiding question for this work was how to leverage task symmetries and invariances with neural networks. Successfully exploiting symmetries is tantamount to introducing additional, structured bias while reducing variance and can increase accuracy, robustness, and sample-efficiency. The thesis made several contributions to this topic: Chapter 3 analysed the capacity and constraints of sum aggregation, one of the most used permutation invariant neural network paradigms. Chapter 4 proposed and compared different permutation invariant relational reasoning modules for multi-object tracking. Chapter 5 introduced a roto-translation equivariant graph network based on self-attention, while Chapter 6 expanded this to be suitable for iterative position refinement, which is crucial for protein structure prediction. Chapter 7 introduced a neural network module that can promote or suppress a nuisance factor (label invariance) while simultaneously making the network more interpretable. The following paragraphs will discuss limitations, future work and applications of these contributions.

## 8.1 Limitations and Future Work

This thesis covered multiple symmetries and invariances, analysing existing architectures and proposing new alternatives. A common denominator across all discussed architectures was some degree of task agnosticism. The SE(3)-Transformer from Chapter 5, for example, — developed for tasks on point clouds and graphs — was applied to indoor furniture classification, particle dynamics prediction and regression tasks on molecules. On the other hand, the flexibility of these approaches also has its limitations. For almost all of the discussed architectures, the symmetries to be leveraged were defined beforehand. Adding further invariances to the neural network, such as mirror- or scale-invariance, would likely result in fundamental changes of the layer designs. Regular representation approaches such as Finzi et al. [21] or Hutchinson et al. [32] tend to be more flexible with respect to which symmetry group to incorporate. However, with all of the above, symmetries do have to be known and defined before training the model. In the indoor object classification task in Chapter 5, the network is presented with symmetry-preserving and symmetry-breaking input features. It can therefore *learn* to make the predictions invariant or not. Although in this setting, this approach leads to performance improvements compared to only using either symmetry-preserving or symmetry-breaking input features, it remains to be explored in future work how this flexibility is incorporated in neural networks in the most principled and effective manner.

The above is related to another fundamental limitation of group invariant neural networks. A symmetry group has to be closed under composition of its elements. A (symmetry) group that includes small deformations therefore typically needs to also include large deformations. Hence, if a task is invariant only with respect to small deformations, this invariance can typically not be modelled by a symmetry (Section 3.3 in Bronstein et al. [6]). As a result, group invariant neural networks often only cover a small fraction of the space of transformations one would like to be invariant to (Bruna et al. [7]).

Another limitation of many networks leveraging symmetries is computational cost, specifically for Janossy pooling with large k (Murphy et al. [49]) as well as for the work in Chapters 5 and 6. Leveraging symmetries is partly motivated by avoiding costly data augmentation. In that sense, it does save compute. In contrast, a single forward and backward pass is typically significantly slower due to redundant computations (Cohen and Welling [15] and Murphy et al. [49]) or costly mathematical computations of equivariant kernels (Fuchs et al. [26], Thomas et al. [59], and Weiler et al. [66]). The specifics of this trade-off often determine whether such equivariant approaches are used in practice (Baek et al. [1] and Jumper et al. [35]) or not (Qi et al. [53], Rao et al. [55], and Wang et al. [64]). In particular, the SE(3)-Transformer as well as tensor field networks (TFNs) require computations of the spherical harmonics for each sample of the dataset separately, yielding a bottleneck in computation. Mainly through implementing a GPU-based spherical harmonics calculation, a single forward or backward pass of the SE(3)-Transformer is significantly faster than with the original TFN (see Appendix of Chapter 5), allowing for application of these networks to real-world tasks. More recently, Milesi [47] proposed further optimisations yielding additional speedups of over an order of magnitude, allowing to train the SE(3)-Transformer on the QM9 dataset in under 30 minutes. These developments are highly promising and could play a significant role in further popularising these equivariant approaches.

## 8.2 Applications

Many of the chapters in this thesis feature an experiment in a physics domain. This is not a coincidence. Symmetries play an important role in all areas of physics and the natural sciences in general (Chapter 13 in Penrose [51]). This is, for example, instantiated in Noether's theorem, which shows that a continuous symmetry of the Lagrangian is always connected to a conserved (i.e. invariant) property of the system (Noether [50]). In our everyday world, particularly the spatial symmetries are often destroyed by the earth's gravitation. This is the reason why the point cloud classification task of indoor furniture in Chapter 5 is not fully symmetric. A chair

is almost always in an upright position. Knowing where 'up' is therefore helps to understand and process the environment. In contrast, when going to the length scales of molecules and below, the effect of earth's gravitation can mostly be safely ignored. As a result, machine learning tasks on molecules tend to have more symmetries. It is, therefore, no surprise that the molecular domain, specifically protein structure prediction (Baek et al. [1] and Jumper et al. [34, 35]), is currently the main driver for the increased research interest in equivariant neural networks. It will be interesting to see how these recent advances will help with the understanding of human diseases and drug discovery and what role equivariance will play. Outside of biochemistry, similar models and techniques could aid with the discovery of new catalysts, enabling technologies like hydrogen-based energy storage (Chanussot et al. [11]).

# References


[1] M. Baek, F. Dimaio, I. Anishchenko, J. Dauparas, S. Ovchinnikov, G. R. Lee, J. Wang, Q. Cong, L. Kinch, R. D. Schaeffer, C. Millán, H. Park, C. Adams, C. R. Glassman, A. DeGiovanni, J. H. Pereira, A. V. Rodrigues, A. A. van Dijk, A. Ebrecht, D. Opperman, T. Sagmeister, C. Buhlheller, T. Pavkov-Keller, M. K. Rathinaswamy, U. Dalwadi, C. Yip, J. Burke, K. Garcia, N. Grishin, P. Adams, R. Read, and D. Baker. "Accurate prediction of protein structures and interactions using a three-track neural network". In: *Science* 373 (2021), pp. 871–876.

[2] Y. Bengio, N. Leonard, and A. Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." In: *arXiv: 1308.3432* (2013).

[3] G. Benton, M. Finzi, P. Izmailov, and A. G. Wilson. "Learning Invariances in Neural Networks". In: *arXiv:2010.11882* (2020).

[4] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. "Fast Differentiable Sorting and Ranking". In: *International Conference on Machine Learning (ICML)* (2020).

[5] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. "Domain Separation Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[6] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478.

[7] J. Bruna, A. Szlam, and Y. LeCun. "Learning Stable Group Invariant Representations with Convolutional Networks". In: *arXiv:1301.3537*. 2013.

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. "Spectral networks and locally connected networks on graphs". In: *arXiv:1312.6203* (2013).

[9] R. Caruana. "Learning many related tasks at the same time with backpropagation". In: *Advances in neural information processing systems (NeurIPS)*. 1994.

[10] R. Caruana. "Multitask learning". In: *Learning to learn*. Springer, 1998, pp. 95–133.

[11] L. Chanussot, A. Das, S. Goyal, T. Lavril, M. Shuaibi, M. Riviere, K. Tran, J. Heras-Domingo, C. Ho, W. Hu, A. Palizhati, A. Sriram, B. Wood, J. Yoon, D. Parikh, C. L. Zitnick, and Z. Ulissi. "Open Catalyst 2020 (OC20) Dataset and Community Challenges". In: *ACS Catalysis* (2021). DOI: 10.1021/acscatal.0c04525.

[12] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin. "ClusterNet: Deep Hierarchical Cluster Network With Rigorously Rotation-Invariant Representation for Point Cloud

Analysis". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 4989–4997.

[13] T. Cohen, M. Geiger, and M. Weiler. "A General Theory of Equivariant CNNs on Homogeneous Spaces". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[14] T. Cohen and M. Welling. "Group Equivariant Convolutional Networks". In: *Proceedings of the International Conference on Machine Learning, ICML*. 2016.

[15] T. Cohen and M. Welling. "Group Equivariant Convolutional Networks". In: *International Conference on Machine Learning*. 2016, pp. 2990–2999.

[16] M. Cranmer, S. Greydanus, S. Hoyer, P. W. Battaglia, D. N. Spergel, and S. Ho. "Lagrangian Neural Networks". In: *ArXiv* (2020).

[17] M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[18] H. Deng, T. Birdal, and S. Ilic. "PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors". In: *arXiv:1808.10322* (2018).

[19] S. Desai, M. Mattheakis, D. Sondak, P. Protopapas, and S. J. Roberts. "Port-Hamiltonian Neural Networks for Learning Explicit Time-Dependent Dynamical Systems". In: *Physical Review E* (2021).

[20] N. Dym and H. Maron. "On the Universality of Rotation Equivariant Point Cloud Networks". In: *International Conference on Learning Representations (ICLR)* (2021).

[21] M. Finzi, S. Stanton, P. Izmailov, and A. Wilson. "Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data". In: *Proceedings of the International Conference on Machine Learning, (ICML)* (2020).

[22] M. Finzi, M. Welling, and A. G. Wilson. "A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups". In: *International Conference on Machine Learning (ICML)* (2021).

[23] F. B. Fuchs, O. Groth, A. R. Kosiorek, A. Bewley, M. Wulfmeier, A. Vedaldi, and I. Posner. "Scrutinizing and De-Biasing Intuitive Physics with Neural Stethoscopes." In: *British Machine Vision Conference*. 2019.

[24] F. B. Fuchs, A. R. Kosiorek, L. Sun, O. P. Jones, and I. Posner. "Relational Reasoning and Permutation Invariance in Multi-Object Tracking". In: *Workshop on Sets and Partitions, NeurIPS*. 2019.

[25] F. B. Fuchs, E. Wagstaff, J. Dauparas, and I. Posner. "Iterative SE(3)-Transformers". In: *International Conference on Geometric Science of Information, Paris*. 2021.

[26] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling. "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks". In: *Conference on Neural Information Processing Systems (NeurIPS)* (2020).

[27] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. "Domain-Adversarial Training of Neural Networks". In: *Journal of Machine Learning Research* 17 (2016), 59:1–59:35.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[29] S. Greydanus, M. Dzamba, and J. Yosinski. "Hamiltonian Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[30] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. "ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking". In: *The European Conference on Computer Vision (ECCV)*. 2018.

[31] A. Grover, E. Wang, A. Zweig, and S. Ermon. "Stochastic Optimization of Sorting Networks via Continuous Relaxations". In: *International Conference on Learning Representations (ICLR)*. 2019.

[32] M. Hutchinson, C. L. Lan, S. Zaidi, E. Dupont, Y. W. Teh, and H. Kim. "LieTransformer: Equivariant self-attention for Lie Groups". In: *International Conference on Machine Learning (ICML)*. 2020.

[33] M. Jaderberg, V. Mnih, W. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. "Reinforcement Learning with Unsupervised Auxiliary Tasks". In: *International Conference on Learning Representations (ICLR)* (2016).

[34] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žıdek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (2021), pp. 583–589.

[35] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, O. Ronneberger, R. Bates, A. Žídek, A. Bridgland, C. Meyer, S. A. Kohl, A. Potapenko, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, M. Steinegger, M. Pacholska, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. "High accuracy protein structure prediction using deep learning". In: *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)* (2020).

[36] L. Keselman and M. Hebert. "Direct Fitting of Gaussian Mixture Models". In: *Conference on Computer and Robot Vision (CRV)* (2019).

[37] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[38] J. Köhler, L. Klein, and F. Noé. "Equivariant Flows: sampling configurations for multi-body systems with symmetric energies". In: *arXiv:1910.00753* (2019).

[39] R. Kondor. "N-body Networks: a Covariant Hierarchical Neural Network Architecture for Learning Atomic Potentials". In: *arXiv preprint* (2018).

[40] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. "Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks". In: *International Conference on Machine Learning (ICML)* (2019).

[41] M. Liu, Z. Wang, and S. Ji. "Non-Local Graph Neural Networks". In: *arXiv* (2020).

[42] M. Long, Y. Cao, J. Wang, and M. I. Jordan. "Learning Transferable Features with Deep Adaptation Networks". In: *International Conference on Machine Learning (ICML)* (2015).

[43] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. S. Zemel. "The Variational Fair Autoencoder". In: *International Conference on Learning Representations (ICLR)* (2015).

[44] G. Louppe, M. Kagan, and K. Cranmer. "Learning to Pivot with Adversarial Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[45] S. Mallat. "Understanding deep convolutional networks". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (2016).

[46] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman. "Invariant and Equivariant Graph Networks". In: *International Conference on Learning Representations (ICLR)* (2019).

[47] A. Milesi. *Accelerating SE(3)-Transformers Training Using an NVIDIA Open-Source Model Implementation*. `https://developer.nvidia.com/blog/accelerating-se3-transformers-training-using-an-nvidia-open-source-model-implementation/`. 2021.

[48] P. W. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. "Learning to Navigate in Complex Environments". In: *International Conference on Learning Representations (ICLR)* (2017).

[49] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. "Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs". In: *International Conference on Learning Representations (ICLR)* (2019).

[50] E. Noether. "Invariante Variationsprobleme". ger. In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* (1918), pp. 235–257.

[51] R. Penrose. "The road to reality : a complete guide to the laws of the universe". In: *Jonathan Cape London* (2005).

[52] C. R. Qi, H. Su, K. Mo, and L. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[53] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[54] F. Quiroga, F. Ronchetti, L. Lanzarini, and A. Bariviera. "Revisiting Data Augmentation for Rotational Invariance in Convolutional Neural Networks". In: *Modelling and Simulation in Management Sciences* (2018).

[55] Y. Rao, J. Lu, and J. Zhou. "Global-Local Bidirectional Reasoning for Unsupervised Representation Learning of 3D Point Clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[56] V. G. Satorras, H. Emiel, F. B. Fuchs, I. Posner, and M. Welling. "E(n) Equivariant Normalizing Flows for Molecule Generation in 3D". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

[57] V. G. Satorras, E. Hoogeboom, and M. Welling. "E(n) Equivariant Graph Neural Networks". In: *arXiv:2102.09844* (2021).

[58] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. *SchNet: A continuous-filter convolutional neural network for modeling quantum interactions*. 2017.

[59] N. Thomas, T. Smidt, S. M. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. "Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds". In: *arXiv Preprint* (2018).

[60] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. *Deep Domain Confusion: Maximizing for Domain Invariance*. 2014.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* (2017).

[62] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations (ICLR)* (2017).

[63] E. Wagstaff, F. B. Fuchs, M. Engelcke, M. A. Osborne, and I. Posner. "Universal Approximation of Functions on Sets". In: *arXiv:2107.01959*. 2021.

[64] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds". In: *ACM Transactions on Graphics (TOG)* (2019).

[65] J. Wei, M. Goyal, G. Durrett, and I. Dillig. "LambdaNet: Probabilistic Type Inference using Graph Neural Networks". In: *International Conference on Learning Representations (ICLR)* (2020).

[66] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[67] M. Wulfmeier, A. Bewley, and I. Posner. "Addressing appearance change in outdoor robotics with adversarial domain adaptation". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), pp. 1551–1558.

[68] M. Wulfmeier, A. Bewley, and I. Posner. "Incremental Adversarial Domain Adaptation for Continually Changing Environments". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1–9.

[69] Q. Xie, Z. Dai, Y. Du, E. H. Hovy, and G. Neubig. "Controllable Invariance through Adversarial Feature Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[70] N. Yadati, M. Nimishakavi, P. Yadav, A. Louis, and P. P. Talukdar. "HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Classification". In: *arXiv* (2018).

[71] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin. "Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets". In: *International Conference on Learning Representations (ICLR)* (2019).

[72] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, and A. Smola. "Deep Sets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[73] Y. Zhang, J. Hare, and A. Prügel-Bennett. "FSPool: Learning Set Representations with Featurewise Sort Pooling". In: *International Conference on Learning Representations (ICLR)* (2020).

[74] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung. "Rotation Invariant Convolutions for 3D Point Clouds Deep Learning". In: *International Conference on 3D Vision (3DV)*. 2019.

[75] A. Zisserman, D. Forsyth, J. Mundy, C. Rothwell, J. Liu, and N. Pillow. "3D Object Recognition Using Invariance". In: *Artifical Intelligence* 78 (1995), pp. 239–288.