Exploiting Prior Knowledge in Machine Learning Model Design

Edward Wagstaff

Worcester College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2021

Abstract

The design of models which are appropriate for specific tasks is an important activity in machine learning. This thesis considers multiple ways in which knowledge about the task at hand can be incorporated into the design of a machine learning model: (i) by using Bayesian models, which incorporate prior knowledge in probability distributions; (ii) by designing models to respect the symmetries of the task; (iii) by adapting models in a practical setting. For Bayesian models, we propose a probabilistic numeric method for computing integrals which arise in the context of Bayesian inference. Respecting the symmetries of a task comes under the framework of group equivariance and invariance. We provide a theoretical investigation of Deep Sets, a prominent permutation-invariant model, and propose an extension of the SE(3)-Transformer, an SE(3)-equivariant model. Finally, we present an investigation of the application of machine learning techniques to problems in atmospheric physics, in which the relevant models are adapted to specific known features of the problem.

Exploiting Prior Knowledge in Machine Learning Model Design



Edward Wagstaff

Worcester College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

Dedicated to Mum, Dad and Tom

Acknowledgements

Thanks first and foremost to my supervisor Mike Osborne, and to all my co-authors. Mike has been a great source of advice and support throughout, and has always helped me to keep a level head in difficult times. I've been blessed with excellent co-authors, all of whom have been a genuine pleasure to work with.

Enormous thanks to my partner Yingxin for her infinite patience, support and encouragement.

Thanks to my dear friends Anne Ploin, Adam Golinski, and Fabian Fuchs, for maintaining my sanity; to all of my wonderful cohort on the AIMS CDT; to Wendy Poole, for keeping the show on the road; and to everyone else who has supported me along the way.

Thanks to the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems for funding the bulk of this research.

Abstract

The design of models which are appropriate for specific tasks is an important activity in machine learning. This thesis considers multiple ways in which knowledge about the task at hand can be incorporated into the design of a machine learning model: (i) by using Bayesian models, which incorporate prior knowledge in probability distributions; (ii) by designing models to respect the symmetries of the task; (iii) by adapting models in a practical setting. For Bayesian models, we propose a probabilistic numeric method for computing integrals which arise in the context of Bayesian inference. Respecting the symmetries of a task comes under the framework of group equivariance and invariance. We provide a theoretical investigation of Deep Sets, a prominent permutation-invariant model, and propose an extension of the SE(3)-Transformer, an SE(3)-equivariant model. Finally, we present an investigation of the application of machine learning techniques to problems in atmospheric physics, in which the relevant models are adapted to specific known features of the problem.

Contents

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**

1.1 Bayesian Quadrature 2

1.2 Invariant and Equivariant Models 2

1.3 Adapting Models in a Practical Setting 3

2 Background **4**

2.1 Bayesian Quadrature 4

2.2 Group Invariance and Equivariance 6

2.3 A Practical Setting – GNSS Scintillations 9

3 Batch Selection for Parallelisation of Bayesian Quadrature **11**

3.1 Introduction 11

3.2 Bayesian Quadrature 13

3.3 Batch methods 15

3.4 Method 17

3.5 Experiments 23

3.6 Conclusion 25

4 Universal Approximation of Functions on Sets **28**

4.1 Introduction 29

4.2 Models For Learning on Sets 31

4.3 Universal Function Representation with Deep Sets 36

4.4 Universal Function Approximation with Deep Sets 46

4.5 Other Methods for Achieving Permutation Invariance 63

4.6 Universality Beyond Deep Sets 67

4.7 Conclusion 72

5 Iterative SE(3)-Transformers 74
5.1 Introduction . . . 75
5.2 Background . . . 76
5.3 Implementation of an Iterative SE(3)-Transformer . . . 79
5.4 Experiments . . . 82

6 Auroral Dynamics and GNSS Phase Scintillations: A Machine Learning Approach to Prediction and Clustering 88
6.1 Introduction . . . 89
6.2 Predicting Phase Scintillations . . . 91
6.3 Unsupervised Clustering of Auroral Images . . . 94
6.4 Conclusions . . . 99

7 Conclusion 102
7.1 Discussion and Outlook . . . 102
7.2 Avenues for Future Work . . . 103

Appendices

A Appendix to Chapter 4 106
A.1 Mathematical Remarks . . . 106
A.2 The Importance of Continuity . . . 111
A.3 Proofs and Additional Theorems . . . 114
A.4 A Continuous Function on $\mathbb{Q}$ . . . 132

B Appendix to Chapter 5 136
B.1 Network Architecture and Training Details . . . 136

References 137

List of Figures

3.1 Penalisation of the acquisition function. 19
3.2 Convergence on an in-model integrand for different batch sizes and batch selection methods. 22
3.3 Convergence on a synthetic integrand for different batch sizes and batch selection methods. 22
3.4 Computing the model evidence for a Gaussian Process. 24

4.1 Janossy pooling. 33
4.2 Different versions and variations of Janossy pooling. 34
4.3 A visualisation of sum-decomposition when the input sets are of size 3 and the latent space is 2-dimensional. 51
4.4 Beginning the construction of ν_n from ν_{n-1}. 56
4.5 Continuing the construction of ν_n from ν_{n-1}. 57
4.6 Contours of sum-decomposition approximations to `max` compared to the true `max` function. 61
4.7 Contours of the function f_* as defined by Equation (4.10). 62
4.8 Achieving permutation invariance via sorting. 63
4.9 From sets to graphs and hyper-graphs. 66
4.10 Necessary and sufficient conditions for the size of the latent dimension for Janossy pooling. 69

5.1 Gradient flow in a conventional single-pass SE(3)-Transformer. . . . 80
5.2 Gradient flow in an iterative SE(3)-Transformer. 81
5.3 Illustration of the experimental setup. 83

6.1 Proposed time-convolutional architecture. 93
6.2 Predicted and true time series of GNSS scintillations. 95
6.3 Predicted time series of GNSS scintillations. 95
6.4 Res-AE Encoder. 97
6.5 Original and reconstructed Themis images. 98
6.6 Clustering of encoded images, with labels. 98

A.1 The 6-dimensional simplex Δ_6 visualised as the complete graph K_7. 108

A.2 Illustration of a quasi-sorting function g on $\mathbb{R}^{2\times 2}$. 110
A.3 A continuous function on $\mathbb{Q}$ with no continuous extension to $\mathbb{R}$. . . 112
A.4 An illustration of the interpolation scheme defining ν_n. 125
A.5 Several iterations of Ψ_n 133

List of Tables

5.1 Average energy of the system after optimisation. 84
5.2 Performance comparison of single-pass and iterative SE(3)-Transformers. 84

1 Introduction

In this thesis, we investigate a number of ways in which prior knowledge about a problem may be incorporated into the design of a machine learning model to be applied to that problem. This thesis follows an integrated format, in which each chapter is a self-contained paper. The relevant background material and literature review is therefore provided in each chapter individually, and only general background is given in Chapter 2. The original contributions of this thesis can be divided into three sections:

1. **Chapter 3. Bayesian Quadrature**

   This chapter is unpublished work.

2. **Chapters 4 and 5. Invariant and Equivariant Models**

   Chapter 4 is an extension of Wagstaff et al. [1], which was published at ICML 2019.

   Chapter 5 was published at the International Conference on Geometric Science of Information 2021.

3. **Chapter 6. Adapting Models in a Practical Setting**

   Chapter 6 is a combined version of two papers presented at the Workshop on Machine Learning and the Physical Sciences at NeurIPS 2019.

1.1 Bayesian Quadrature

A central challenge in Bayesian inference is the computation of integrals with respect to probability distributions. This is common in the context of computing marginal likelihoods, and for evaluating expectations. These integrals are usually intractable, and must therefore be approximated. Probabilistic numerics places this approximation itself into a Bayesian context, treating the problem of approximation as one of inference from limited evidence - in particular, Bayesian Quadrature is a probabilistic numeric view on approximate integration. Since this is a Bayesian method, we can naturally incorporate prior knowledge about the integrand into our priors.

In Chapter 3, we investigate a method for parallelising Bayesian Quadrature. Ordinarily, such methods operate by sequentially choosing points to maximise information gain for the integral. This sequential process means that parallel computation cannot straightforwardly be used to improve wall-clock-time performance. Inspired by methods in Bayesian Optimisation, we investigate the selection of points in batches rather than in sequence, so that evaluations of expensive integrands can be parallelised.

1.2 Invariant and Equivariant Models

Tasks may exhibit symmetries between input and output data. We may want outputs to remain invariant to transformations of the input data, or we may want the output to undergo a corresponding transformation (i.e. to be equivariant). If we know about these symmetries at the time of designing a model, we can ensure that these symmetries are respected by the model, cutting down the size of the hypothesis space represented by the model and allowing for better-performing and more easily trained models.

In Chapter 4, we undertake a theoretical investigation of the expressivity of Deep Sets, a popular architecture for modelling permutation-invariant functions. The architecture works by mapping inputs into an encoding space, aggregating them, and then decoding. It has been shown to be universal for the modelling of

continuous permutation-invariant functions, but this universality was proved based on some additional assumptions on the dimensionality of the encoding space. It was previously unknown whether these assumptions were necessary. We undertake a mathematical analysis of the architecture, demonstrating that the encoding space dimension required for universality grows linearly with the number of inputs.

In Chapter 5, we investigate an iterative version of the SE(3)-Transformer. The SE(3)-Transformer is an SE(3)-equivariant model for 3D graph data. This model was introduced as a single-pass model. We analyse the changes that must be made, particularly in the backward pass of the model, in order to apply the model in an iterative fashion while allowing end-to-end training and avoiding information bottlenecks. This iterative version of the model may iteratively modify the graph, updating node positions and other graph features. Such an iterative approach has been applied in protein structure refinement.

1.3 Adapting Models in a Practical Setting

In Chapter 6, we apply machine learning techniques to a problem in atmospheric physics. This work is concerned with the relationship between the aurora borealis (in particular, images thereof taken from the ground) and ionospheric disturbances which interfere with satellite transmissions (instances of such interference are termed "scintillations"). The primary focus of this work is an attempt to predict scintillations based on other physical data including images of the aurora. We also investigate whether unsupervised clustering of auroral images results in clusters which bear a relationship to human-labeled aurora classes. We make adaptations to standard machine learning models which reflect our prior knowledge of the data. Most prominently, the target time series for the prediction problem is not smooth and consists of sparse, high peaks, and we therefore train the model with a custom loss function which is designed to encourage such outputs.

2 Background

2.1 Bayesian Quadrature

Bayesian inference is a task commonly encountered in the context of probabilistic machine learning methods. In these methods, prior knowledge about the problem at hand may be incorporated by specifying a prior probability distribution, and Bayesian inference is then performed to produce a posterior probability distribution on the basis of observed data. Mathematically, given a prior $p(x)$ and data D, the posterior $p(x|D)$ is

$$p(x|D) = \frac{p(D|x)p(x)}{p(D)}.$$

The denominator $p(D)$ is known as the marginal likelihood, and may be written as follows:

$$p(D) = \int p(D|x)p(x)\ \mathrm{d}x\,.$$

The computation of such integrals is of central importance to Bayesian inference. Traditional methods for approximating these integrals, such as Markov Chain Monte Carlo (MCMC) [2], usually produce point estimates, with convergence guarantees which provide some loose estimates of the uncertainty in the final estimate. Probabilistic numeric approaches [3], by contrast, treat the approximation itself as

an inference problem. In particular, Bayesian Quadrature [4] (or, in some sources [5], Bayesian Probabilistic Numerical Integration) proceeds by using established methods for Bayesian regression, most prominently Gaussian Processes [6], to obtain a posterior distribution for the integrand. The posterior distribution on the integrand induces a posterior distribution on the value of the integral, thus giving not simply a point estimate, but a full probability distribution representing our posterior beliefs about the value of the integral. Note that the integrand may be a known deterministic function – for instance, this is the case when computing the marginal likelihood as shown above. In this case, the probability distribution over the integrand should not be interpreted in terms of some underlying stochasticity, but should instead be interpreted in a Bayesian sense as representing incomplete knowledge of the behaviour of the integrand. That is, the probability distribution should be interpreted in terms of *epistemic* uncertainty, rather than in terms of *aleatory* uncertainty.

In both Bayesian Quadrature and MCMC, we make observations of the integrand by evaluating it at many points. An advantage of Bayesian Quadrature over MCMC is that, in Bayesian Quadrature, the choice of where next to evaluate the integrand may depend on the current state of knowledge about the integrand. This means that, similar to Bayesian optimisation [7], function evaluations may be chosen so as to maximise some criterion such as information gain or variance reduction. A disadvantage of this approach is that, in order to make use of knowledge gained from previous evaluations, function evaluations must be made sequentially rather than in parallel. In contrast, MCMC is straightforwardly parallelisable in part because the choice of sample x_t taken at time step t has only minimal dependence on the state of the algorithm at previous time steps, depending only on x_{t-1}. The focus of Chapter 3 is on addressing this disadvantage, by proposing a method for selecting points for evaluation in batches rather than one at a time. This batching allows for parallelisation of the function evaluations, allowing for improved performance in terms of wall-clock time.

2.2 Group Invariance and Equivariance

2.2.1 Motivation

Group invariance and equivariance is most naturally thought of in terms of symmetry, mathematically described by *group theory*. When we know that a task is symmetric with respect to a certain transformation, we may build this symmetry into our model, and group equivariance provides the mathematical framework for this. Perhaps the most prominent example of equivariance in machine learning is the translational symmetry of convolutional neural networks (CNNs) [8] applied to images. In order to discuss equivariance in more detail, we must first establish some basic mathematical definitions and notation, before giving a brief overview of how these mathematical concepts apply to machine learning. For further detail on group theory, see for example Armstrong [9], Pinter [10], or Rotman [11].

2.2.2 Definitions

Definition 2.1. A *group* G is a set X together with a binary operation $*$ on X, satisfying the following properties:

1. There is an *identity* element $e \in X$, so that for any $g \in X$, $e * g = g * e = g$
2. For every $g \in X$, there is an *inverse* element g^{-1}, with $g * g^{-1} = g^{-1} * g = e$
3. For every $g, h \in X$, $g * h \in X$
4. $*$ is associative. For every $g, h, k \in X$, we have $g * (h * k) = (g * h) * k$

Usually we ignore the distinction between X and G, and refer to elements of X as being elements of G. We will also write gh for $g * h$.

Definition 2.2. Given a group G and a set X, an *action* of G on X is a function $f : G \times X \to X$ satisfying the following properties:

1. $f\Big(g, f(h, x)\Big) = f(gh, x)$.
2. $f(e, x) = x$.

We usually write gx for $f(g, x)$.

Definition 2.3. Let X, Y be sets, and let $f : X \to Y$. Let G be a group which acts on both X and Y. We say that f is *equivariant* with respect to the action of G, or G-*equivariant*, if for every $x \in X$:

$$f(g * x) = g * f(x)\,.$$

Definition 2.4. Let X, Y be sets, and let $f : X \to Y$. Let G be a group which acts on X. We say that f is *invariant* with respect to the action of G, or G-*invariant*, if for every $x \in X$:

$$f(g * x) = f(x)\,.$$

Definition 2.5. A *representation* of a group G on a vector space V is a function $f : G \to \mathrm{GL}(V)$, where $\mathrm{GL}(V)$ is the set of invertible $n \times n$ matrices representing linear maps $V \to V$, which satisfies:

$$f(g * h) = f(g)f(h)\,.$$

In other words, the group operation on G becomes matrix multiplication under the map f. For each element g of G there is a corresponding matrix $f(g)$, and multiplying matrices after applying f is the same as composing group elements before applying f.

2.2.3 Discussion

A group G is an abstract mathematical object which captures the compositional structure of a set of symmetries. The concept of a group *action* allows the abstract group elements to be applied to other, more concrete objects. For example, letting T be the group of translations in the integer plane $\mathbb{Z}^2$, T has the structure of vector addition on $\mathbb{Z}^2$. But there is a natural action of T on the set of images $\mathcal{I}$, allowing us to mathematically capture the translation of images using the language of groups.

Note that one group may have many actions on a given set. For example, the 2D translation group T may act on $\mathbb{Z}^3$ by acting in the natural way on the first two coordinates, or on the last two coordinates (or in many other ways). In these cases, the group action preserves all the structure of the group, but this is not necessarily the case. For example, there is always a trivial action of a group G on a set X, where every group element corresponds to the identity on X. This is still an action, but all the structure of G is lost.

The notion of an action allows us to talk about symmetric operations applying to sets of objects (e.g. translations applying to images). The notions of equivariance and invariance allow us to talk precisely about what it means for a function to respect those symmetries. For instance, image classification may be viewed as a translation-invariant function from images to labels. Supposing that we wish to learn this image classification function, we may exploit our knowledge that it is translation-invariant by working with a model, such as a CNN, which is also translation-invariant. [1] One benefit of this approach is that it restricts the search space, ruling out all translation-sensitive functions, which are already known to be incorrect.

Invariance is, in fact, just a special case of equivariance. This is easily seen by considering the trivial action mentioned above, in which all group elements go to the identity function. If G acts on the output space trivially in this way, then a function which is G-equivariant with respect to this action is clearly G-invariant. In Chapter 4, we will be concerned entirely with the property of invariance – specifically, *permutation*-invariance, as guaranteed by the *Deep Sets* architecture [12], which enforces permutation invariance by use of a sum pooling operation. Our work is closely related to Bloem-Reddy and Teh [13], which investigates the representation of *stochastic* permutation-invariant functions, and also to Bueno and Hylton [14], which, like our work, investigates the universality of pooling-based architectures for *deterministic* permutation-invariant functions.

[1] Though note that CNNs are only *approximately* translation-invariant – a translation of the input image may affect the output. Exact invariance fails for two main reasons. Firstly, while *global* pooling operations are translation-invariant, *local* pooling operations are not. Secondly, the convolutional layers themselves fail to be exactly translation-equivariant, due to edge effects when a translation causes part of the input image to leave the frame.

Group equivariance in machine learning applies in two main ways. First, there are problems which are genuinely equivariant rather than invariant. This is the case in Chapter 5, where we consider a model which works with 3-dimensional input data and produces 3-dimensional outputs. In this model, rotations and translations of the input result in identical rotations and translations of the output. Here, the group action on the input space is the same as the group action on the output space. Second, G-equivariant modules may be used in the course of building a G-invariant model. For instance, the convolutional layers in a CNN are translation *equivariant*, and translation *invariance* is achieved by applying pooling operations.

2.3 A Practical Setting – GNSS Scintillations

A global navigation satellite system (GNSS) is a constellation of satellites used for navigation. The global positioning system (GPS) is one instance of such a system. GNSS satellites transmit signals to Earth, and these signals (and therefore the services which the satellites provide) are subject to disturbance by the movement of charged particles in the Earth's ionosphere. For an introduction to the physics governing the ionosphere, see Kivelson and Russell [15]. These ionospheric disturbances to satellite signals are known as *scintillations*. Forecasting the occurrence of scintillations would enable users of GNSS systems to be aware ahead of time of likely disturbances to GNSS services. However, directly modelling the physical dynamics underlying these scintillations is computationally intractable with current models, as the models would need to be run at a fine spatial resolution. In Chapter 6, we therefore investigate the application of machine learning techniques to this problem, with the ultimate goal of developing predictive models which are computationally tractable.

Preface to Chapter 3

In this section of the thesis, we address the incorporation of prior knowledge into machine learning models in a probabilistic manner, by way of Bayesian inference. More specifically, we address one of the core numerical challenges in performing Bayesian inference – integration against a prior probability distribution. Our approach is grounded in *probabilistic numerics*, a family of methods which casts problems of numerical approximation as problems of inference from data.

Applying probabilistic numerics to the numerical problems arising in Bayesian inference is in some sense "doubly Bayesian", giving us two opportunities to encode prior knowledge in the model. One opportunity, closer to the problem that the model is being applied to, in choosing a prior probability distribution over the object of inference (usually some model parameters). And a second opportunity, more distant from the model's end goal, in specifying a probability distribution over integrands.

3

Batch Selection for Parallelisation of Bayesian Quadrature

Edward Wagstaff, Saad Hamid, Michael A. Osborne

Abstract

Integration over non-negative integrands is a central problem in machine learning (e.g. for model averaging, (hyper-)parameter marginalisation, and computing posterior predictive distributions). Bayesian Quadrature is a probabilistic numerical integration technique that performs promisingly when compared to traditional Markov Chain Monte Carlo methods. However, in contrast to easily-parallelised MCMC methods, Bayesian Quadrature methods have, thus far, been essentially serial in nature, selecting a single point to sample at each step of the algorithm. We deliver methods to select batches of points at each step, based upon those recently presented in the Batch Bayesian Optimisation literature. Such parallelisation significantly reduces computation time, especially when the integrand is expensive to sample.

3.1 Introduction

Both within and without machine learning, we encounter integrals of the form

$$Z = \int \ell(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}, \tag{3.1}$$

where both $\ell(\boldsymbol{\theta})$ (e.g. a likelihood), and $\pi(\boldsymbol{\theta})$ (e.g a prior) are non-negative. These integrals are usually intractable, and we must therefore turn to approximation to solve them.

One possibility is the Laplace approximation (see for example MacKay [16], chapter 27). For multimodal distributions it can be repeated on the residuals – the difference between the approximation and the true distribution – to obtain a better approximation [17]. However, this requires optimisation and is not invariant to changes of representation. Another possibility is variational inference which provides a lower bound for the value of the integral, but there is limited underpinning theory and optimisation of the lower bound can be difficult [18]. Markov Chain Monte Carlo (MCMC) methods [2] have also seen broad use and success, but they are subject to a number of objections [19], the most pertinent of which is that they ignore the locations of samples from the likelihood when computing the integral. They therefore suffer from low sample efficiency and a poor $\frac{1}{\sqrt{N}}$ convergence rate. All of these approximation methods are unsuitable for applications where evaluating the desired likelihood is expensive (e.g. if we must make an estimate based on only a few dozen evaluations).

Bayesian Quadrature (BQ) offers an alternative framework for approaching the same integral problem. It belongs to a family of methods termed *probabilistic numerics* [3], which view the approximation of intractable quantities as a problem of inference from limited data, to which the tools of probability theory can be applied. In BQ, a probabilistic model of the integrand (typically a Gaussian Process) is maintained, from which the integral is inferred. Gunter et al. [20] showed that BQ can achieve faster convergence – both in wall-clock time and sample complexity – than state of the art MCMC methods. Some deterministic quadrature techniques can be reinterpreted in terms of Bayesian Quadrature – for instance, the trapezoid rule (with $\frac{1}{N}$ convergence) arises from BQ with a linear spline kernel [21].

One drawback of BQ as developed thus far is that is not easily parallelised, in contrast to the comparatively straightforward situation with MCMC methods where, for instance, several independent Markov chains may be run in parallel. While

the probabilistic model allows us to explore the space more efficiently, determining where next to evaluate the integrand depends on the current state of the model, i.e. on all evaluations made so far. Nonetheless, parallelisation is essential to realising competitive performance using modern computational hardware.

Using ideas applied in Batch Bayesian Optimisation, particularly those by González et al. [22], we propose a novel technique for parallelising BQ by selecting new points in batches rather than one-by-one.

3.2 Bayesian Quadrature

In Bayesian Quadrature, we work with a probabilistic model of an integrand which induces a probability distribution over the value of the integral. The early work on BQ [4, 23–25] identified that Gaussian Processes offer a well-studied method for placing a probability distribution over functions, so we work with these to provide our probabilistic model.

Recall from Equation (3.1) that we are interested in the quantity $\int \ell(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$. We typically choose to place our Gaussian process prior on the likelihood ℓ, rather than on the product $\ell \cdot \pi$ (though we could choose to do either).

Given a Gaussian prior π and a GP model for our likelihood,

$$\ell \mid \mathbf{y} \sim \mathcal{GP}(m_{\mathbf{y}}, C_{\mathbf{y}})\,, \tag{3.2a}$$

$$m_{\mathbf{y}}(x) = k(x, X)k(X, X)^{-1}\mathbf{y}\,, \tag{3.2b}$$

$$C_{\mathbf{y}}(x, x') = k(x, x') - k(x, X)k(X, X)^{-1}k(X, x')\,, \tag{3.2c}$$

the value of the integral $Z = \int \ell(x)\pi(x)\mathrm{d}x$ is Gaussian distributed – since Gaussian Processes are closed under linear operators (see e.g. Rasmussen and Williams [26]), the GP on the function induces a normal distribution on the value of the integral Z. By linearity of expectation and bilinearity of covariance, the mean and variance of Z are simply [27]:

$$\mathbb{E}\left(Z \mid \mathbf{y}\right) = \int m_{\mathbf{y}}\left(x\right) \pi(x) \mathrm{d}x \,, \tag{3.3a}$$

$$\mathrm{var}\left(Z \mid \mathbf{y}\right) = \iint C_{\mathbf{y}}\left(x, x'\right) \pi(x) \pi(x') \mathrm{d}x \mathrm{d}x' \,. \tag{3.3b}$$

These integrals are, for certain choices of prior, π, and kernel, K, analytic. In particular, this is true when π is a Gaussian and K is the squared exponential covariance, and we use this combination throughout this paper.

At a high level, the advantages of Bayesian Quadrature are two-fold. First, with the stronger prior of BQ, evaluations provide stronger constraints on the integrand, and therefore more information about the integral. Secondly, we can use the uncertainty of the GP model to inform our choice of where next to evaluate the integrand, thus exploring the space more efficiently. We do this by defining and maximising an *acquisition function* which is high-valued at points which, according to our model, will reduce the uncertainty in the integral. In the standard BQ model, this second advantage does not apply – note that the posterior covariance only depends on the *locations* sampled, not on the function value at those points [28]. However, as we shall see below, it is possible to develop more complex models where, among other advantages, the covariance of better-informed posteriors does depend on the sampled function values.

3.2.1 Warped Bayesian Quadrature

The Gaussian process framework has some restrictions which interfere with our ability to model our true beliefs about the functions we wish to work with. Standard Bayesian Quadrature does not allow us to express our belief that the function to be modelled (in our case a likelihood) is non-negative, since the distribution at any point is a Gaussian. Recent work has investigated applying a warping to the output space of the GP to address this [20, 28, 29]. Rather than modelling the likelihood ℓ itself as a GP, we model a transformed likelihood $g(\ell)$ such that the true likelihood $g^{-1}(g(\ell))$ is non-negative (i.e. such that the range of g^{-1} is the positive reals). For

example, $g = \log(x)$ and $g = \sqrt{x}$ have been investigated. Though the final posterior on ℓ is not a GP, it is possible to derive a GP which closely approximates it.

For the purposes of this investigation we follow the method of Gunter et al. [20], dubbed WSABI, which makes use of the transformation $g = \sqrt{x}$. In particular we use the approximation dubbed WSABI-L. WSABI-L was introduced alongside WSABI in Gunter et al. [20], as an analytically tractable approximation to the analytically intractable WSABI model. It relies on a linearised approximation to the inverse transform g^{-1}, and has the virtue of providing a closed-form estimate for the final integral rather than requiring Monte Carlo estimation.

In WSABI, the acquisition function is simply the variance of the posterior model – due to the form of the model, this automatically balances exploring areas of high uncertainty and exploiting in areas where the objective function is believed to be high-valued. WSABI has so far been considered only as a sequential process, selecting and evaluating one point at a time. We propose its parallelisation by turning to batch methods.

3.3 Batch methods

Bayesian Quadrature as described above is an essentially serial process. That is, we must sample a single point at each step, before updating our model (and therefore our acquisition function) accordingly in order to recommend the next point. Since we expect each sample to be expensive, this is a serious bottleneck – we would rather be able to work in parallel, taking several samples at once in order to gain the maximum amount of information about our integrand. In order to enable this, we turn to ideas which have been successfully used in Bayesian Optimisation.

The field of Bayesian Optimisation [7] has a clear relationship to Bayesian Quadrature. In particular, both fields use a probabilistic model to guide exploration of a search space by defining an acquisition function to be maximised. In Bayesian Optimisation, so-called batch methods have been proposed as an effective way of allowing parallelisation of this exploration (see e.g. Ginsbourger, Le Riche, and Carraro [30], González et al. [22], Nguyen et al. [31]).

In batch methods, rather than selecting one point to sample at each step, we select a batch of points, usually of a fixed size n. Naively one could try to select the n highest points of the acquisition function, but this has the obvious issue that these will all be essentially equal to the maximum of the acquisition function, leading to an uninformative batch – the quality of the batch as a whole is more complex than simply the sum of the quality of each point individually. One way of viewing this issue is that we require an informative batch, and therefore we require a diverse batch.

To find the optimal batch we could, in principle, define an acquisition function over all sets of n points (e.g. by considering the expected reduction in variance of our integral estimate) and maximise it, but in practice this is too expensive – it increases the dimensionality of the problem by a factor of n and the acquisition function is also likely to be more costly to compute. It is therefore common to turn to heuristics where each point is selected sequentially, with the acquisition function modified after each point is selected.

A simple method for performing this sequential selection is the so-called Kriging Believer strategy [30]. We "sample" our chosen point by setting its value to the posterior mean, update our Gaussian process model to take the new evidence into account, and select the most informative point from the posterior. We repeat this process until we have a sufficiently large batch, and then make the (real) evaluations at selected points and input them as observations into our model. This naturally forces further exploration of the space, as the region immediately around each "sampled" point will no longer be considered informative.

We also consider a *local penalisation* strategy which directly penalises the acquisition function around the chosen point. In Bayesian Optimisation, local penalisation was introduced by González et al. [22]. With this strategy, we don't use our full model, as in Kriging Believer, but instead directly modify the acquisition function to lower its value around each selected point, encouraging diversity in the final batch.

We address some properties of local penalisation proposed by González et al. [22] which make it well-suited to optimisation but poorly suited to quadrature.

We propose a novel method of local penalisation which is designed to select more informative batches in the context of quadrature.

In the next section, we will consider the application of batch methods to Bayesian Quadrature.

3.4 Method

In this section we introduce, to our knowledge, the first Batch Bayesian Quadrature routine. We also introduce a novel version of local penalisation.

The core algorithm is simple – we repeatedly select a batch of points, evaluate the integrand at these points, and input the returned data into our model. At each iteration, we can calculate an estimate of the integral using BQ. There is a question here of the stopping criterion. Theoretically, Bayesian Quadrature offers a principled stopping criterion since it returns a variance for its estimate of the integral, and we can simply evaluate the variance at each iteration and stop once it has reached a certain threshold. For the specific case of WSABI-L, the model is overconfident away from the data, and we expect that the BQ variance will therefore be erroneously low and unsuitable as a stopping criterion. This is a question that clearly merits further investigation, but for the purposes of this paper we simply set a fixed computational budget.

The most important step of the algorithm is the batch selection process. Above we described the Kriging Believer approach, and in general terms described local penalisation. We now outline our approach to local penalisation.

3.4.1 Local Penalisation

With local penalisation methods, there is a balance to be struck in how wide the penalisers are – if our penalisation affects a broad region, we may erroneously exclude genuinely informative points. If, however, our penalisation affects a narrow region, we may not sufficiently encourage diversity, and produce a batch of points which are clustered too closely together. González et al. [22] propose a multiplicative penaliser with a minimum at the sampled point. To determine how wide the penaliser should

be, they make use of a Lipschitz constant, a measure of the maximum absolute slope of the acquisition function. In their method the Lipschitz constant is incorporated into a probability density function which is used as part of the penaliser.

The method of González et al. [22] is well suited to optimisation, but has some weaknesses when applied to the problem of quadrature. It is not sufficiently exploratory for two reasons. First, it heavily penalises regions surrounding a selection which has a high variance. Second, it only softly penalises the points which are added to the batch, allowing for the possibility of returning to highly similar points multiple times in the same batch.

In the context of optimisation, this behaviour makes sense – finding the precise location of a global optimum may require thorough exploration of a small region in one batch. In quadrature, by contrast, the result depends on the global behaviour of the function rather than a single optimum. We therefore consider an alternative approach, attempting to place an upper bound on the true acquisition function (i.e. the acquisition function had the integrand been evaluated at the previous selections).

As part of our approach, we apply a *hard penaliser*. Whereas the acquisition function in González et al. [22] may remain high at a sampled point, we enforce that the acquisition function at a sampled point must be zero, thus preventing over-exploitation.

We also make use of a *local* Lipschitz constant, rather than a global constant as employed by González et al. [22]. Furthermore, we consider a Lipschitz constant for the acquisition function, rather than the objective function. In the optimisation context, a global Lipschitz constant for the objective function is used to place bounds on the location of the global optimum. By contrast, we wish to thoroughly explore the space as directed by our acquisition function. In our context, a high global Lipschitz constant could lead to pathological behaviour, where the acquisition function is penalised very narrowly in a region where a wider penaliser would be more appropriate.

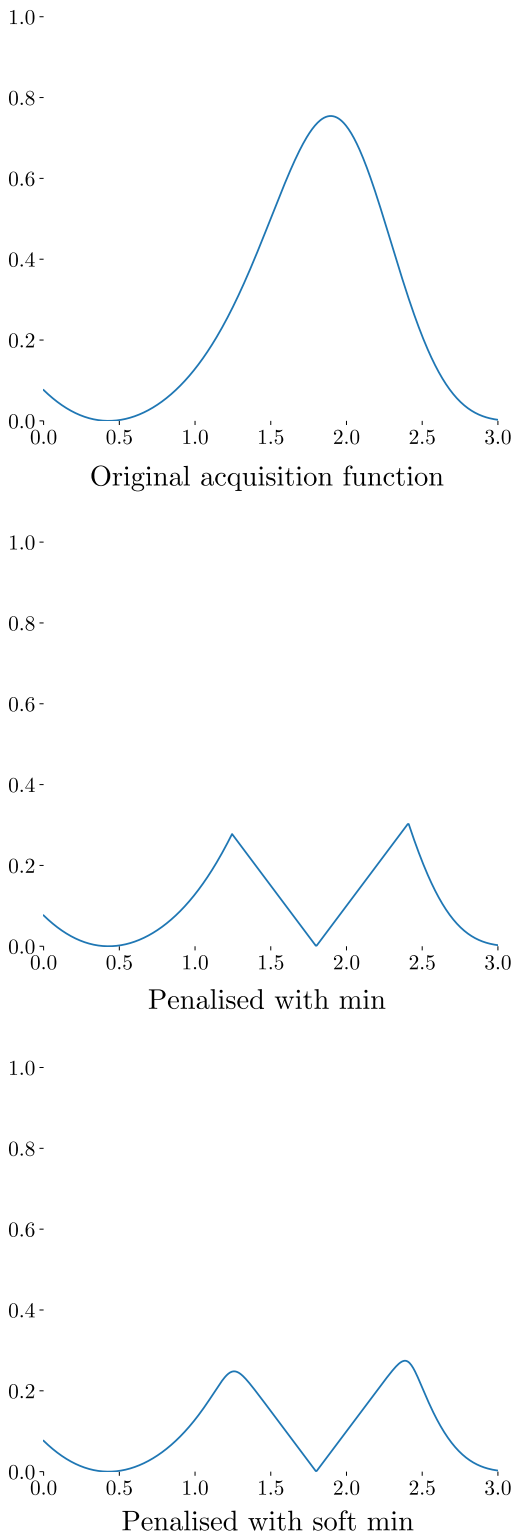


Figure 3.1: The penalisation of the acquisition function, showing the non-differentiable minimum, and the differentiable soft minimisation achieved by applying the ℓ_p-norm for $p = -6$. Note that the maxima after penalisation are stationary points with the soft minimisation, but not with the true minimisation.

Hard Local Penalisation

Recall that a function f is said to be Lipschitz continuous if there is a constant L such that, for all x, y,

$$||f(x) - f(y)|| \leq L||x - y||. \tag{3.4}$$

We expect that the true acquisition function after sampling a point x_0 will be zero at x_0 and increase away from it. We wish for our penalised function to reflect this. The role of the local Lipschitz constant here is to estimate the rate at which the true acquisition function will increase. We find the local Lipschitz constant L for a given point by locally optimising the norm of the gradient of the acquisition function.

Viewing this local L as a bound on the slope of the acquisition function around x_0, we then define a *Lipschitz Cone* which is zero at the point and increases linearly away from the point. For a Lipschitz constant L and point x_0, we define the Lipschitz Cone by:

$$C(x) = L||x - x_0||. \tag{3.5}$$

Our acquisition function is then taken to be the minimum of the original acquisition function and the Lipschitz Cones of all points so far added to the batch.

This minimum, however, is not differentiable everywhere. In particular, it is not differentiable where the cone intersects the original acquisition function (or where two cones intersect). These intersections may in fact be maxima of the penalised acquisition function, so ideally the penalised function would have stationary points on these intersection surfaces. We address this by applying a soft minimisation, rather than the true minimum, which leaves us with an acquisition function which is differentiable everywhere except at the centres of the Lipschitz cones (i.e. the already selected batch points).

Recall that the ℓ_p norm $||\cdot||_p$ is defined by

$$||x||_p = \left(\sum_i x_i^p\right)^{\frac{1}{p}}. \tag{3.6}$$

It is well known that the ℓ_p-norm for large p approaches the ℓ_∞ norm, $||x||_\infty = \max_i(x_i)$. Large values of p can therefore be viewed as applying a kind of soft maximisation. Large negative values of p, on the other hand, apply a soft minimisation to the input vector.[1] We therefore take our final acquisition function to be an ℓ_p-norm of the vector whose components are the function evaluation for each of the penalisers and the original acquisition function. There is a balance to be struck in the choice of p here – large p will more closely resemble the true minimum of all the functions, but will also be more prone to numerical errors (these errors will appear particularly around the important regions where the true minimum is not differentiable). We found that small values of p already came reasonably close to the true minimum function, and selected $p = -6$ for our experiments.

Since we expect the true acquisition function to be smooth, rather than resembling a sharp cone near evaluated points, we set the gradient of our Lipschitz cone to be a fraction of the Lipschitz constant – this has the effect of encouraging (or rather not discouraging) exploration. We found $\frac{1}{2}$ to work well in practice. In principle, this parameter could be tuned to help determine the degree to which the algorithm explores diverse regions of the space, with smaller values encouraging exploration more strongly, but we did not investigate this possibility as part of this work.

3.4.2 Optimising the acquisition function

To optimise the acquisition function, we run a local optimiser multiple times from randomly selected starting locations. We sampled these locations according to the prior π against which we are integrating. We chose the number of starting locations to be 10 times the dimensionality of the integrand.

Implementing Multi-start Optimisation

Our code for this method was written in python, using numpy and scipy. The most straightforward way to implement multi-start optimisation is to simply loop over

[1] For negative p this is no longer a norm, but we refer to it as an ℓ_p-norm for brevity.

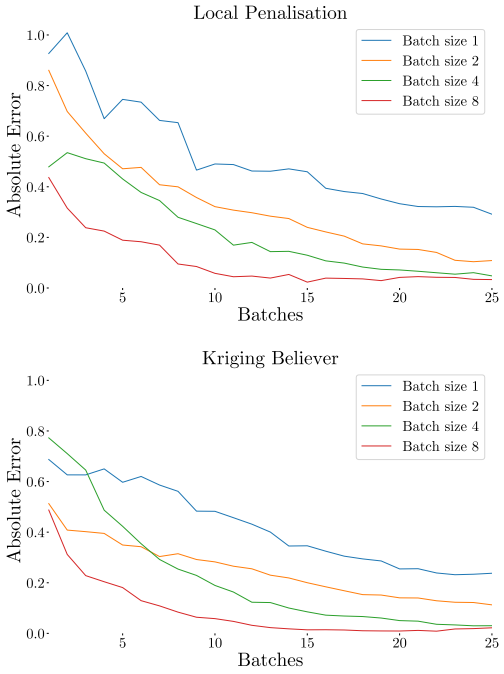


Figure 3.2: Convergence on an in-model integrand for different batch sizes and batch selection methods.

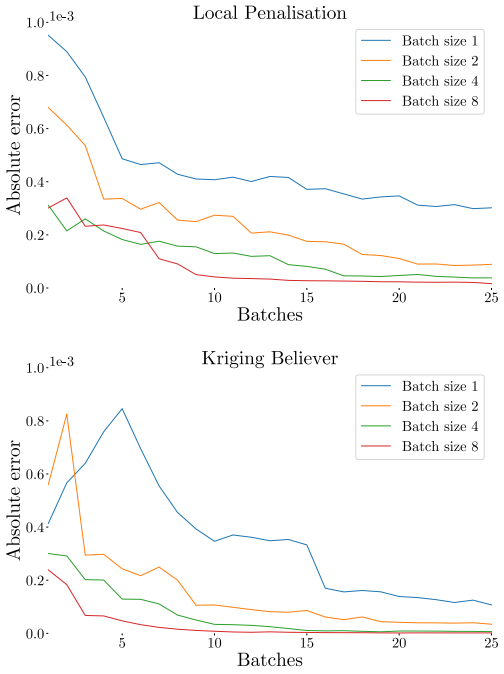


Figure 3.3: Convergence on a synthetic integrand for different batch sizes and batch selection methods.

the starting points and perform the optimisation many times. It is however possible to speed up this process when using numpy by taking advantage of fast vectorised computation in numpy rather than using slow python loops. The acquisition function can be evaluated at many points by passing these points as a numpy array, and this is many times faster than passing the points one at a time in a python loop. To take advantage of this in the optimisation process, we posed our multi-start optimisation problem as a single-start optimisation problem in higher dimensions. Each point in our high-dimensional space is the concatenation of multiple points in the original low-dimensional space. Denoting concatenation by $|$, we define a new function $g(x_1|\dots|x_n) = f(x_1) + \cdots + f(x_n)$. The Jacobian of this function is then obtained by concatenating the Jacobian of f at each x_i, and we eliminate the python loop over starting points by simply starting this high-dimensional optimisation at $x_1|\dots|x_n$.

3.5 Experiments

Prior work [20, 28, 29] has demonstrated that in terms of wall-clock time, BQ methods are competitive with MCMC. Here we investigate the degree to which adding batch methods can improve the performance of BQ in terms of the number of batches required (which, with an objective function which can be effectively parallelised, is a better indicator of actual time taken than the number of samples taken). For a batch of size n, the best we can reasonably hope for is a factor-n decrease in batch complexity against the sequential (i.e. batch size 1) method. We test both of our proposed batch selection methods on synthetic problems to assess the improvement achieved by batching, and compare both batch selection methods against MCMC methods in a test computing the model evidence of a Gaussian Process. Bayesian quadrature is known to scale poorly to integrals in high-dimensional spaces [4]. Our technique does not attempt to resolve this issue, and we therefore follow [20] in restricting our experiments to low-dimensional problems, in up to 4-dimensional spaces.

Our first experiment is on an in-model integral, i.e. a sample from a Gaussian Process prior with the WSABI warping applied to it. We integrated this function with WSABI for a number of different batch sizes. Each sample was taken in 2 dimensions, and integrated against an isotropic Gaussian prior. Because we are working in 2 dimensions, the ground truth was obtained by simply evaluating the function on a fine grid and evaluating the integral naively. We plot the average absolute error over 10 runs of this experiment in Figure 3.2.

The in-model test demonstrates effective parallelisation of the process – we can clearly see that larger batch sizes converge much more quickly to the true integral. We also see that both methods are similarly effective – Kriging Believer and Local Penalisation show comparable performance.

Our second experiment is on a synthetic integral. Here we construct our integrand as a mixture of isotropic Gaussians. We do this in 4 dimensions, varying the number of components between 10 and 15, setting their variance uniformly at random between 1 and 4, and setting their means uniformly at random within

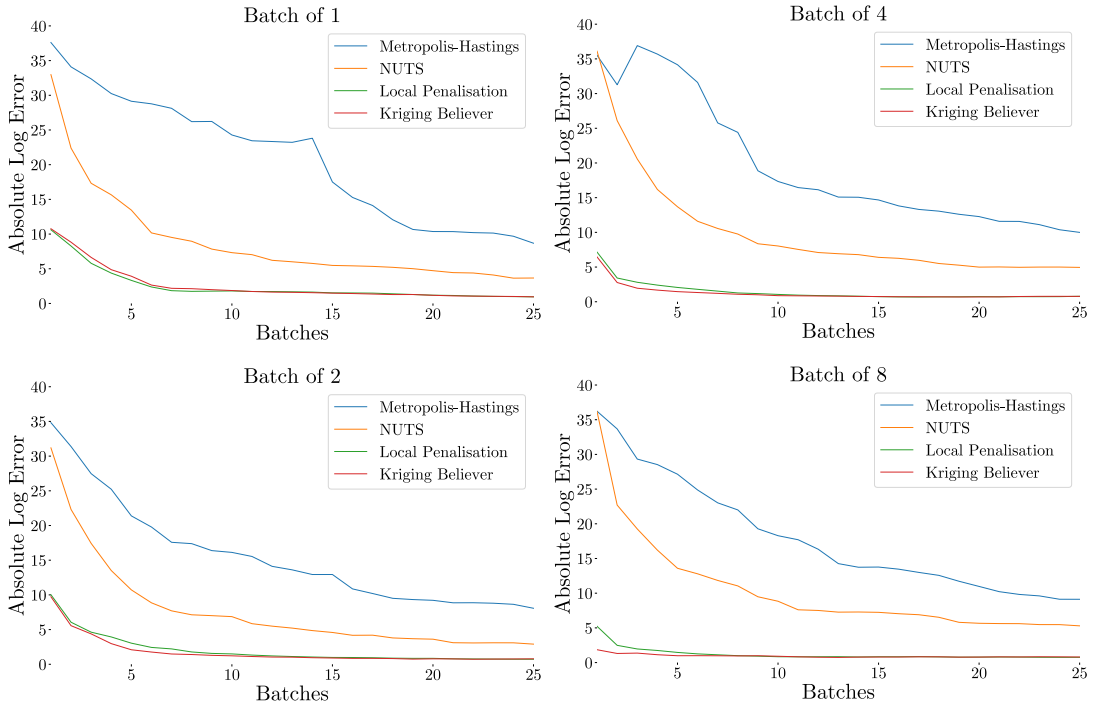


Figure 3.4: Computing the model evidence for a Gaussian Process.

the box bounded by $[-3, 3]$ in all dimensions. We set the hyperparameters of the underlying GP in our model (which uses an isotropic squared exponential kernel) by optimising the marginal likelihood.

Again we test both of our batch selection methods, with a varying number of batch sizes. We plot the average absolute error over 10 runs of this experiment in Figure 3.3. The results are similar to the in-model test, demonstrating an effective speedup with increasing batch size, and again showing comparable performance between the two batch selection methods.

Our final experiment is computing the model evidence for a Gaussian Process with an isotropic squared exponential covariance function. This GP has a fixed dataset $\mathcal{D}$, collected by running 20 iterations of Bayesian Optimisation on the Branin-Hoo function, a common benchmark function for Bayesian Optimisation [32]. We marginalise $p(\mathcal{D}|\mu, \sigma)$ against a Gaussian prior on the two GP hyperparameters μ and σ. Since this integral is 2D, we once again obtained ground truth by evaluating on a fine grid.

In this experiment we also compare against two MCMC methods – Metropolis-Hastings (Metropolis et al. [33]; Hastings [34]), a popular standard method, and NUTS [35], a newer state-of-the-art method. Both of these methods were run with 500 samples of burn-in, and when comparing against our batched method with a batch size of n, we take the result obtained from running n parallel chains. Figure 3.4 shows the average absolute error in the log of the model evidence, over 50 runs of this experiment. Once again it is clear that larger batch sizes converge more quickly for our method, and our method clearly outperforms the MCMC benchmarks.

All three experiments show improved performance as batch size increases, demonstrating that the selected batches are genuinely informative sets of points.

3.6 Conclusion

In this paper we have presented a routine for performing Bayesian Quadrature as a batched method, and introduced a novel form of local penalisation as part of the batch selection process. Our method allows parallelisation of methods which have so far only been available as serial processes. We have demonstrated that the performance of our method improves as the degree of parallelism increases, and shown that the method is competitive with state-of-the-art MCMC when the sampling budget is small.

Acknowledgements

This research was funded by the EPSRC AIMS Centre for Doctoral Training at the University of Oxford.

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).**

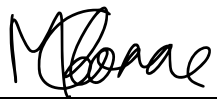
Title of Paper	Batch Selection for Parallelisation of Bayesian Quadrature
Publication Status	□Published □ Accepted for Publication □Submitted for Publication ☑Unpublished and unsubmitted work written in a manuscript style
Publication Details	

Student Confirmation

Student Name:	Edward Wagstaff		
Contribution to the Paper	Wrote the majority of the implementing code, responsible for the majority of the algorithm design, and most of the work in preparing the manuscript.		
Signature 		Date	09/06/2021

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:	Michael Osborne, Professor of Machine Learning		
Supervisor comments	All above seems accurate and fair.		
Signature 		Date	10 June 2021

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

Preface to Chapters 4 and 5

In this section of the thesis, we address the incorporation of prior knowledge into machine learning models by way of models which are known to satisfy the same symmetries as the task to be solved. For example, if we wish to train a model with respect to a task which we know to be rotation-invariant, we may exploit this knowledge by making use of a rotation-invariant model.

We address two different types of symmetry. First, in Chapter 4, we discuss permutation invariance, which arises most naturally in the context of unordered collections, or sets. We provide an analysis of arguably the most prominent permutation-invariant architecture, *Deep Sets* [12], and show that the model's expressivity is determined by the relationship between the size of the input sets and the dimension of latent vectors used by the model. Second, in Chapter 5, we discuss SE(3) equivariance – that is, equivariance to the operations of rotation and translation in 3D space. Motivated by the problems of protein structure prediction and refinement, which we know to satisfy this symmetry, we propose an extension to an existing SE(3)-equivariant method, the SE(3)-Transformer [36], allowing this model to be applied in an iterative fashion.

4

Universal Approximation of Functions on Sets

Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Michael A. Osborne, Ingmar Posner

Abstract

Modelling functions of sets, or equivalently, *permutation-invariant* functions, is a long-standing challenge in machine learning. Deep Sets is a popular method which is known to be a universal approximator for continuous set functions. We provide a theoretical analysis of Deep Sets which shows that this universal approximation property is only guaranteed if the model's latent space is sufficiently high-dimensional. If the latent space is even one dimension lower than necessary, there exist piecewise-affine functions for which Deep Sets performs no better than a naïve constant baseline, as judged by worst-case error. Deep Sets may be viewed as the most efficient incarnation of the Janossy pooling paradigm. We identify this paradigm as encompassing most currently popular set-learning methods. Based on this connection, we discuss the implications of our results for set learning more broadly, and identify some open questions on the universality of Janossy pooling in general.

4.1 Introduction

Many applications of machine learning work with collections of inputs or features which are best modelled as *sets*. Crucially, these collections have no intrinsic ordering, which distinguishes them from other commonly-encountered types of data such as images or audio. Examples are plentiful: a point cloud obtained from a LIDAR sensor; a group of atoms which together form a molecule; or a collection of objects appearing in an image. Although the data for these problems is not ordered, standard machine learning techniques usually impose an ordering on the data, either by representing it as a vector or matrix and applying order-sensitive operations, or by iterating over data points one by one. Any function with set-valued inputs is insensitive to this imposed ordering, in the sense that the output does not change when the input is reordered. This property is known as *permutation invariance*.

Tasked with learning a permutation-invariant target function, we may build this invariance into our model – we can regard this as giving the model an inductive bias. A family of techniques have been proposed in recent years to build and train deep permutation-invariant machine learning models. Most notably, the seminal work by Zaheer et al. [12] introduces *Deep Sets*. The central idea of Deep Sets is to process the individual elements of a set in parallel using a shared encoding function before aggregating these encodings using a *symmetric function* – for example summation, averaging, or max-pooling. We refer to the vector space in which this aggregation happens as the model's *latent space*. This design ensures that the model is exactly permutation-invariant. A similar model is also explored in Qi et al. [37].

While easy to implement and parallelise, processing each input individually and pooling globally hinders relational reasoning [38, 39]. Arguably the most popular alternatives are self-attention mechanisms, which perform weighted summation to aggregate information via input-dependent attention weights. Self-attention explicitly performs relational reasoning by processing elements in pairs, rather than individually – working with relationships between pairs of elements is therefore an in-built feature of the model. This stands in contrast to Deep Sets, where no facility for relational reasoning is built into the model (though such reasoning can

still be learned). While self-attention has famously been widely applied in natural language processing [40], it has also been applied to sets [41] and is popular in the literature on graph learning [42].

Although Deep Sets and self-attention may not immediately appear to be closely related, both can be viewed as special cases of k-ary *Janossy pooling* [43]. Deep Sets is the most restricted instance of Janossy pooling, which makes it a good starting point for fuller theoretical characterisation of these methods. In this work, we contribute to this theoretical understanding by considering how the dimensionality of the Deep Sets model's latent space affects its expressive capacity. This extends the partial characterisation provided in Zaheer et al. [12].

Firstly, we follow Zaheer et al. [12] in considering the conditions under which Deep Sets is capable of *exactly representing* any continuous target function (i.e. *universal representation*). We show that the sufficient condition on the model's latent space dimension given in Zaheer et al. [12] is essentially the best possible – it can be lowered, but only by 1, and this weakened sufficient condition is in fact also a *necessary* condition. Secondly, we consider universal *approximation*, a weaker and more practically important property. We demonstrate that this is subject to the same necessary and sufficient conditions as universal representation. In fact, we show that if the latent dimension is even 1 lower than necessary, there exist piecewise-affine functions for which the model's worst-case error is no better than the worst-case error of a model which simply outputs 0. Finally, we consider the implications of our findings for other related architectures such as self-attention, and discuss the universality properties of other set-learning methods.

In summary, this work makes the following contributions:

- Section 4.2 provides an overview of the most popular deep learning architectures for learning functions on sets, including Deep Sets, and discusses their unification as instances of the *k-ary Janossy pooling* paradigm;
- Section 4.3 proves a necessary and sufficient condition for *representing* arbitrary continuous functions on sets with the Deep Sets architecture;

- Section 4.4 furthers our analysis of the Deep Sets architecture by proving a necessary and sufficient condition for *approximating* arbitrary continuous functions on sets;
- Section 4.5 discusses other models for deep learning on sets;
- Section 4.6 discusses universal approximation criteria for architectures beyond Deep Sets, and identifies some open questions of interest.

Section 4.3 of this paper is a revised and abridged version of work previously presented in Wagstaff et al. [1]. This paper provides a more extensive and comprehensive discussion of the subject and extends the work in Wagstaff et al. [1] by providing additional results for architectures other than Deep Sets and a theoretical characterisation of universal function approximation.

4.2 Models For Learning on Sets

In this section, we examine deep learning architectures for data that consists of unordered sets of elements. We are centrally concerned with the property of permutation invariance as introduced above, but we briefly note that functions working with set-valued data may also be *permutation-equivariant*. In the simplest case, equivariance appears in the context of mapping sets to sets. Although the ordering of elements is arbitrary, we may still want the ordering of the input set to be consistent with the ordering of the output set – that is, any permutation of the input results in a corresponding permutation of the output. This consistency of ordering is *permutation equivariance*. Romero and Cordonnier [44] provide a full mathematical definition of permutation equivariance, along with a detailed analysis of the equivariance properties of self-attention. We do not consider permutation-equivariant set functions in detail in this paper.

Returning to permutation *invariance*, a core consideration for the design of permutation-invariant models is how to maintain maximum expressivity (in the sense of being able to model a broad class of functions) while also ensuring permutation

invariance. Murphy et al. [43] introduce *Janossy pooling* as a unifying framework of methods that learn either strictly permutation-invariant functions or suitable approximations. Janossy pooling is known to be highly expressive, and in fact it is trivially shown to be universal in the sense that any permutation-invariant function may be represented within the Janossy pooling framework.

In its most general form, Janossy pooling considers all possible permutations π of the input elements x_i. Each permutation $\pi(\mathbf{x})$ is separately passed through the same permutation-sensitive function ϕ. The outputs for the different permutations are then aggregated by computing the average (or by another global pooling operation). If two inputs $\mathbf{x}$ and $\mathbf{y}$ are permutations of one another, this process will give the same output for both inputs. As noted above, we refer to the space in which the aggregation happens as the model's *latent space*.

Mathematically, the procedure outlined above can be written as

$$\widehat{f}(\mathbf{x}) = \frac{1}{|S_M|} \sum_{\pi \in S_M} \phi\Big(\pi\left(\mathbf{x}\right)\Big), \tag{4.1}$$

where $\mathbf{x}$ has M elements, and S_M is the group of all permutations π of M elements. A permutation-invariant function $\widehat{f}$ is thereby constructed from a permutation-sensitive function ϕ. This permutation-sensitive function is typically implemented as a neural network, but other function approximators such as Gaussian processes may also be used. The architecture is illustrated in Figure 4.1. This figure also illustrates that a second function may optionally be used to post-process the output of the aggregation operation:

$$f(\mathbf{x}) = \rho\Big(\widehat{f}\left(\mathbf{x}\right)\Big). \tag{4.2}$$

This second function ρ does not need to follow any constraints to guarantee invariance because its input, $\widehat{f}(\mathbf{x})$, is already permutation-invariant. In other words, the ordering information is already lost by the point that ρ is reached in Figure 4.1.

One drawback of this scheme is that it can become prohibitively expensive for large set sizes M, because of the large number of terms in the sum in Equation (4.1). The computational complexity scales at least linearly with the cardinality of

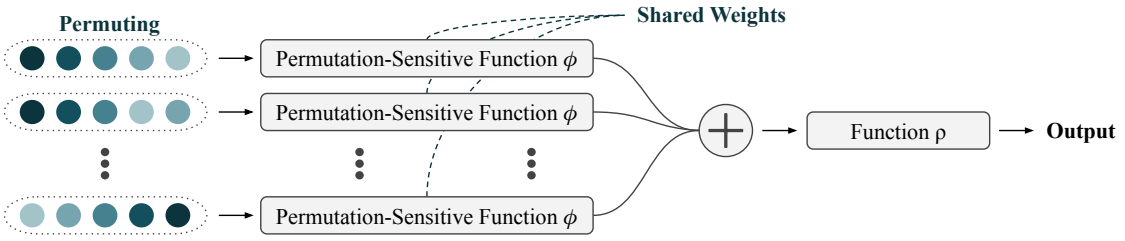


Figure 4.1: The Janossy pooling paradigm applies the same permutation-sensitive network to each possible permutation of the input set. When using a permutation-invariant pooling operation, such as averaging or summation, information about the ordering is eliminated. A second neural network may be used to predict the final output. This paradigm guarantees permutation invariance of the output without imposing restrictions on any of the neural network components.

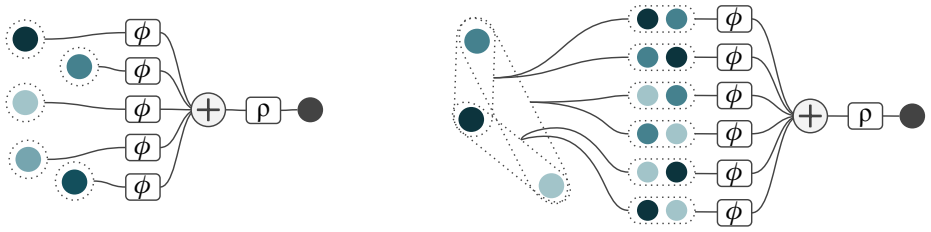
S_M, which is $M!$. To remedy this, Murphy et al. [43] discuss several options for reducing the computational complexity: (i) sorting; (ii) sampling; (iii) restricting permutations to k-tuples.

Rather than considering all permutations, sorting only considers a single canonical permutation, which is obtained by sorting the input. The sorting operation may be hand-specified, or it may be learned. Sampling, by contrast, aggregates over a randomly-sampled subset of permutations (so that the sum in Equation 4.1 is over a randomly chosen subset of S_M). While the output in this case is only approximately rather than strictly permutation invariant, Murphy et al. [43] show that this works reasonably well empirically. Nevertheless, most models that are used in practice [e.g. 12, 41, 45] fall into the third category of restricting permutation to k-tuples. The theoretical analysis conducted in our work focuses on this third category, which we describe in more detail in the remainder of this section.

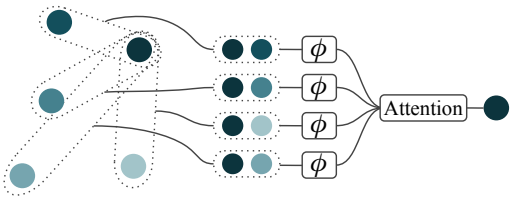
4.2.1 Limiting the Number of Elements in Permutations

Equation (4.1) considers all possible M-tuples from an M element set. To save computation, one can instead consider all k-tuples[1] with $k < M$. The models obtained by setting $k = 1$ and $k = 2$ are visualised in Figure 4.2. Mathematically,

[1]We require that the tuples we discuss here consist of distinct elements from our input set. So in this case, $(2, 2)$ is not a valid 2-tuple from the set $\{1, 2, 3\}$. This restriction is not always observed, for instance self-attention *does* allow such 2-tuples, but for brevity we restrict ourselves to discussing the case of distinct elements.



(a) Janossy pooling with $k = 1$ (*Deep Sets*) **(b)** Janossy pooling with $k = 2$



(c) Self-attention

Figure 4.2: Different versions and variations of Janossy pooling. Permutation invariance is guaranteed by processing all combinations of k elements and then aggregating via a sum (or softmax in the case of attention). Self-attention, a variant of Janossy pooling with $k = 2$, focuses on one node at a time (the darkest node here), computing an output for this specific node. It is often employed for all nodes in parallel in a permutation-equivariant manner, mapping sets of points to sets of points [41].

letting $\mathbf{x}_{\{k\}}$ denote the set of all k-tuples from $\mathbf{x}$, we have

$$\widehat{f}(\mathbf{x}) = \frac{1}{P(M,k)} \sum_{x \in \mathbf{x}_{\{k\}}} \phi(x) \text{ , where } P(M,k) = \frac{M!}{(M-k)!}. \tag{4.3}$$

For clarity, we provide an example for $M = 4$ and $k = 2$. If the input set is $\{w, x, y, z\}$, the sum will be over all 2-tuples from the set, namely:

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

For sufficiently small k, this gives a sum with far fewer than $M!$ terms, leading to a computationally tractable method in many practical cases. For fixed k and varying M, the number of terms in the sum is $\mathcal{O}(M^k)$. Setting $k = 1$ therefore provides a model whose cost is linear in the size of the input set. Increasing k comes at the cost of additional computational complexity, but as we will discuss in the following subsection, also allows for more explicit relational reasoning.

Expressivity and Interactions

The terms *relational reasoning* and *interactions* are often encountered in the deep learning literature [38, 46, 47]. We use the expression *interactions between elements in a set* to refer to the fact that the output may depend not only on the individual contribution of each element, but may also depend on the fact that multiple elements appear *together in the same set*. Relational reasoning describes the act of modelling and using these interactions. To illustrate this with a simple example, consider the task of assessing how well a set of ingredients go together for cooking a meal. If we set $k = 1$, the function ϕ can take into account relevant individual attributes, but will be unable to spot any clashes between ingredients (like garlic and vanilla). Increasing k allows ϕ to see multiple elements at once, and therefore perform relational reasoning about pairs of ingredients, enabling a more expressive model of what tastes good. If we view ϕ as an "encoder" and ρ as a "decoder", ϕ is capable of encoding information about interactions, which ρ can then make use of during decoding.

If we set $k = 1$, this is no longer the case. Zaheer et al. [12] show that the encoder ϕ can encode the entire set when $k = 1$, and therefore the decoder ρ can in principle recover all the input information and perform relational reasoning from there. However, any such relational reasoning is not built into the model, and must be entirely learned. Murphy et al. [43] speculate that this additional burden on what must be learned by ρ can contribute to difficulties in training the $k = 1$ model on some tasks which depend heavily on relational reasoning.

Many current neural network architectures on sets and graphs resemble Janossy pooling with $k = 2$. This choice of k represents a tradeoff between computational complexity (lower k is better) and ease of training for relational reasoning (higher k is better[2]). Most famously, self-attention algorithms (schematically depicted in Figure 4.2c) compare two elements of the set at a time, typically by performing a scalar product [40, 41]. The results of the scalar products are used as attention weights for

[2]Though we note that high k being "better" here is, at least theoretically speaking, a speculative statement. Nevertheless, the fact that $k = 2$ is widely used in practice despite the increased computational complexity strongly suggests that increasing k does indeed improve performance in some contexts.

aggregating information from different points via a weighted, permutation-invariant sum. While this mechanism is very similar to Janossy pooling with $k = 2$, some additional architecture choices are often made. For example, a softmax might be used to ensure that the attention weights are normalised.

Deep Sets

The model obtained by setting $k = 1$ is in fact a popular and well-known special case of Janossy pooling – Deep Sets [12]. As discussed above, choosing $k = 1$ may hinder relational reasoning, but Deep Sets is nevertheless a widely-used architecture for permutation-invariant models, for example forming the basis of two well-known point cloud classification approaches, PointNet [37] and PointNet++ [45]. As the computationally cheapest instance of Janossy pooling, its linear scaling in the number of inputs makes it particularly well suited to problems with large input sets – a point cloud for example may consist of thousands of points. Importantly, it is known to be universal, guaranteeing (at least in principle) that it is capable of computing any permutation-invariant target function. However, the conditions under which universality holds are not fully characterised, either for Janossy pooling more broadly or for Deep Sets specifically. Zaheer et al. [12] and Han et al. [48] provide sufficient conditions for Deep Sets to be universal, but do not show whether universality fails if these conditions are violated – that is, they do not give *necessary* conditions for universality, and leave open the possibility that universality is preserved under weaker conditions. In the following two sections, we provide proofs of necessary conditions for the universality of Deep Sets.

4.3 Universal Function Representation with Deep Sets

This first section of our theoretical investigation focuses on function *representation*, building directly on the proofs and results from Zaheer et al. [12]. By function representation, we mean the ability of a model to *exactly represent* a given target function. By *universal* representation, we mean the ability of a model to represent *all*

target functions from a given function class (e.g. the class of all permutation-invariant functions). The original analysis from Zaheer et al. [12] demonstrates the universality of Deep Sets by considering universal representation, and in this section we present a direct continuation and refinement of this analysis. Nevertheless, it must be noted that function representation is a stronger property than required in practice, and we will return to analyse the weaker property of function *approximation* in Section 4.4. Proofs of all novel results from this section are provided in Appendix A.3.

4.3.1 Preliminaries

We begin by introducing the necessary definitions and notation, before detailing our contribution and its relationship to the original results from Zaheer et al. [12]. Importantly, we will establish the concept of *sum-decomposition*, which is a concise mathematical description of the Deep Sets architecture.

Definitions and Notation

Notation 4.1. We denote sets of inputs in boldface with subscript-indexed elements, e.g. the set $\mathbf{x}$ has elements $x_1, \ldots, x_M$.

Notation 4.2. Throughout, the variable M refers to the number of elements in the input sets under consideration.

Notation 4.3. We denote the group of all permutations of M elements by S_M. A permutation $\pi \in S_M$ may be thought of as a bijection from $\{1, \ldots, M\}$ to itself.

Definition 4.4. 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 $\pi \in S_M$.

Definition 4.5. We say that a function f is *sum-decomposable* if there are functions ρ and ϕ such that

$$f(\mathbf{x}) = \rho\Big(\sum_i \phi(x_i)\Big). \tag{4.4}$$

In this case, we say that (ρ, ϕ) is a *sum-decomposition* of f.

Note that sum-decomposition is equivalent to Janossy pooling with $k = 1$, i.e. Deep Sets. Comparing Equation (4.4) with Equations (4.2) and (4.3), these expressions differ only in dividing ϕ by an integer, and can be made equal simply by rescaling ϕ.

Notation 4.6. Given a sum-decomposition (ρ, ϕ), we write $\Phi(\mathbf{x}) \coloneqq \sum_i \phi(x_i)$. With this notation, we can write Equation (4.4) as $f(\mathbf{x}) = \rho\Big(\Phi(\mathbf{x})\Big)$. We may also refer to the function $\rho \circ \Phi$ as a sum-decomposition.

Definition 4.7. Let (ρ, ϕ) be a sum-decomposition. Write Z for the domain of ρ (which is also the codomain of ϕ, and the space in which the summation happens in Equation 4.4). We refer to Z as the *latent space* of the sum-decomposition (ρ, ϕ).

Definition 4.8. Given a space Z, we say that f is *sum-decomposable via* Z if f has a sum-decomposition whose latent space is Z.

Definition 4.9. We say that f is *continuously sum-decomposable* when there exists a sum-decomposition (ρ, ϕ) of f such that both ρ and ϕ are continuous. (ρ, ϕ) is then a *continuous sum-decomposition* of f.

Notation 4.10. Denote the power set of a space X (that is, the set of all subsets of X) by 2^X.

Our Contribution

As noted above, the Deep Sets model uses sum-decomposition to represent permutation-invariant functions. Our contribution is to show that the universality of sum-decomposition is dependent on the dimensionality of the latent space Z. Specifically, we show that $\dim(Z)$ must be at least M, where M is the number of elements in the input sets. We refine the analysis from Zaheer et al. [12], relying on the observation that some of the mappings used for the proofs of universality in the original analysis are highly discontinuous, and cannot be computed in practice. By considering continuous mappings[3], we provide a proof of the above lower bound

[3]This function class includes, for instance, all functions which can be computed by neural networks with continuous activation functions, or by Gaussian processes with continuous kernels.

on $\dim(Z)$. Because this section directly follows on from the analysis given in Zaheer et al. [12], we first give a brief reproduction of the statement and proof of two key theorems from that work.

Background Theorems

Zaheer et al. [12] consider two cases. First, where $\mathbf{x}$ is a subset of, or drawn from, a *countably infinite* universe $\mathfrak{U}$. Second, the case where $\mathfrak{U}$ is *uncountably infinite*.

Theorem 4.11 (Countable case)**.** *Let $f : 2^{\mathfrak{U}} \to \mathbb{R}$ where $\mathfrak{U}$ is countable. Then f is sum-decomposable via $\mathbb{R}$.*

Proof. Since $\mathfrak{U}$ is countable, each $x \in \mathfrak{U}$ can be mapped to a unique element in $\mathbb{N}$ by a bijective function $c : \mathfrak{U} \to \mathbb{N}$. If we can choose ϕ so that Φ is invertible, then we can set $\rho = f \circ \Phi^{-1}$, giving

$$f = \rho \circ \Phi \,,$$

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

Now consider $\phi(x) = 4^{-c(x)}$. Under this mapping, each set $\mathbf{x} \subset \mathfrak{U}$ corresponds to a unique real number. The real number $r \coloneqq \Phi(\mathbf{x})$ can be decoded to the set $\mathbf{x}$ by looking at the base 4 expansion of r. The element $c^{-1}(n) \in \mathfrak{U}$ belongs to $\mathbf{x}$ if and only if the n-th digit of r is 1. This decoding procedure shows that Φ is invertible, and the conclusion follows. □

For the uncountable case, we consider M-element sets from the universe $\mathfrak{U} = [0, 1]$.

Theorem 4.12 (Uncountable case)**.** *Let $M \in \mathbb{N}$, and let $f : [0, 1]^M \to \mathbb{R}$ be a continuous permutation-invariant function. Then f is continuously sum-decomposable via $\mathbb{R}^{M+1}$.*

Another way of stating Theorem 4.12, which brings the terminology more in line with Deep Sets, is as follows:

Theorem 4.12 (Deep Sets terminology)**.** *Deep Sets can represent any continuous permutation-invariant function of M elements if the dimension of the model's latent space is at least $M + 1$.*

Proof. The proof by Zaheer et al. [12] of Theorem 4.12 is more involved than for Theorem 4.11. We do not include it here in detail, but we summarise the main points as follows.

1. Show that the mapping $\Phi : [0, 1]^M \to \mathbb{R}^{M+1}$ defined by $\Phi_q(\mathbf{x}) = \sum_{i=1}^{M} (x_i)^q$ for $q = 0, \ldots, M$ is injective and continuous[4].
2. Show that Φ 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 ρ and ϕ, (ρ, ϕ) is a continuous sum-decomposition of f via $\mathbb{R}^{M+1}$.

□

In light of these theorems, our key conclusions can be summarised as follows:

- In Section 4.3.2 we argue that the guarantee of sum-decomposability via $\mathbb{R}$ given by Theorem 4.11 cannot hold in practice. This is because the necessary mappings are highly discontinuous and cannot be computed.
- In Section 4.3.3 we prove that the guarantee of sum-decomposability via $\mathbb{R}^{M+1}$ given by Theorem 4.12 is essentially the best possible. The dimension of the latent space must be at least M, or universality is lost.

Additionally, we contribute some results on universal representation with discontinuous sum-decompositions. These results are not relevant for practice[5] but are included for mathematical interest, and can be found in Appendices A.2 and A.3.

[4] In the original proof, Φ is denoted E.

[5] One of our proofs depends on the axiom of choice, bringing us firmly beyond the realm of computability!

4.3.2 The Importance of Continuity

Theorem 4.11 appears to give a strong guarantee on the universality of sum-decomposition via $\mathbb{R}$. The theorem applies even when the domain is allowed to be infinite, and so any task implemented on real hardware, where inputs belong to large but finite domains, satisfies the conditions of the theorem. Nevertheless, an insurmountable problem arises when trying to apply the construction from the proof of Theorem 4.11, with the consequence that Theorem 4.11 cannot be applied on real hardware. That is, sum-decomposition via $\mathbb{R}$ does *not* provide a guarantee of universal function representation in practice. We illustrate the problem by considering an example with the domain of all 256×256 pixel images in 8-bit greyscale, which we denote $\mathcal{I}$.

Suppose, as required by the construction used in the proof, that we have a counting function c which assigns an integer to each element of $\mathcal{I}$. There are $2^{2^{19}}$ elements in $\mathcal{I}$, so even if c counts all "interesting" images[6] first, there must be "interesting" images mapping to very large numbers under c. Consider for instance the $2^{2^{16}}$th image $\iota \coloneqq c^{-1}(2^{2^{16}})$, and suppose that, given a set of images $\mathbf{x}$, our target function f needs to compute whether or not $\iota \in \mathbf{x}$. Under the encoding/decoding procedure from the proof of Theorem 4.11, the decoder function ρ is given as input the real number $r \coloneqq \Phi(\mathbf{x})$. Recalling that the construction has $\phi(x) \coloneqq 4^{-c(x)}$, the presence or absence of the image ι in the set $\mathbf{x}$ will affect the value of r by one part in $4^{2^{2^{16}}}$. Any decoder ρ implemented on real hardware will therefore be unable to distinguish between sets containing ι and sets without ι.

The essential problem here is an extreme degree of *discontinuity*. Intuitively speaking, a function is continuous if, at every point in the domain, the variation of the output is insensitive to small variations in the input. In the example above, the decoder ρ is sensitive to vanishingly small perturbations in the input, across its entire input space, and this property renders it impossible to compute in practice. We can avoid this problem by requiring that the functions under consideration are all continuous. This requirement is ubiquitous in work on the function approximation

[6] That is, images which are not just random noise, but have meaningful visual content.

properties of machine learning models, including the universal approximation theorem for neural networks [49], the universality of Gaussian processes with certain kernels [6], the literature on approximating functions on sets [48, 50], and indeed in function approximation results outside machine learning (such as the well-known Stone-Weierstrass theorem [51]). In practice, neural networks with continuous activation functions[7] can only compute continuous functions, and the same is true of Gaussian processes with continuous kernels.

To continue our analysis of sum-decomposition, we therefore adopt the assumption that the target functions and sum-decompositions under consideration are all continuous. This aligns with the conditions in Theorem 4.12, and in the next section we show that the conclusion of Theorem 4.12 is essentially the strongest result that can be achieved under these assumptions.

4.3.3 Function Representation With Continuous Mappings

The following theorem is our central result on the function representation properties of continuous sum-decomposition.

Theorem 4.13. *Let $M, N \in \mathbb{N}$, with $M > N$. 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 Deep Sets to be capable of representing *arbitrary* continuous functions on sets of size M, the dimension of the model's latent space (denoted N in the theorem statement above) must be at least M. A similar statement is also true for models based on the similar concept of *max-decomposition*, as considered by Qi et al. [37] – details are provided in Appendix A.3.6.

In terms of concrete recommendations for practitioners, Theorem 4.13 suggests that when deploying a model based on Deep Sets, the choice of latent space should depend on the number of elements in the input sets under consideration. Referring

[7] I.e. essentially all popular activation functions, including ReLU, ELU, sigmoid, and tanh.

back to our depiction of Deep Sets in Figure 4.2a, this means that the cardinality of the input sets should be taken into account when choosing the length of the vectors output by the ϕ blocks. Choosing a latent space which is not sufficiently high-dimensional may mean that the target function cannot be successfully modelled. It must be noted, however, that Theorem 4.13 places restrictions on function *representation*, but not on function *approximation*. Function approximation comes closer to describing what is needed in practice, and we will return to address this point in Section 4.4, where we prove that this restriction on the latent space dimension also holds for function *approximation* with Deep Sets. We will discuss the practical implications of Theorem 4.13 in more detail after the proof.

In addition to showing that an M-dimensional latent space is necessary, we adapt the proof from Zaheer et al. [12] of Theorem 4.12 to strengthen the result in two ways. Proofs of these two results can be found in Appendix A.3. Firstly, we lower the bound on the sufficient latent space dimension by 1, concluding that an M-dimensional latent space is *necessary and sufficient* for universal representation.

Theorem 4.14. *Let $M \in \mathbb{N}$, and let $f : \mathbb{R}^M \to \mathbb{R}$ be a continuous permutation-invariant function. Then f is continuously sum-decomposable via $\mathbb{R}^M$.*

Secondly, we show that the model can deal with variable set sizes $\leq M$. That is, the model is not only capable of modelling functions on sets of size *exactly* M, but also on sets of size *up to* M. At least in principle, the model is therefore applicable in settings where the input sets may contain variable numbers of points.

Theorem 4.15 (Variable set size)**.** *Denote the set of subsets of $[0, 1]$ containing at most M elements by $[0, 1]^{\leq M}$. Let $f : [0, 1]^{\leq M} \to \mathbb{R}$ be continuous*[8] *and permutation-invariant. Then f is continuously sum-decomposable via $\mathbb{R}^M$.*

[8]Note that we must take some care with the notion of continuity here – see Appendix A.1.2.

Proof of Theorem 4.13

Our proof of Theorem 4.13 relies on showing that the permutation-invariant function $\texttt{max}$, applied to M-element sets of real numbers, is not sum-decomposable via $\mathbb{R}^{M-1}$. To show this, we need the following lemma.

Lemma 4.16. *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*

$$\texttt{max}(\mathbf{x}) = \rho\Big(\sum_i \phi(x_i)\Big). \tag{4.5}$$

Recall that $\Phi(\mathbf{x}) = \sum_i \phi(x_i)$, and write Φ_M for the restriction of Φ to sets of size M.

Then Φ_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 Φ_{M-1} is injective. Suppose there are sets $\mathbf{x}, \mathbf{y}$ such that $\Phi_M(\mathbf{x}) = \Phi_M(\mathbf{y})$. First note that, by Equation (4.5), we must have

$$\texttt{max}(\mathbf{x}) = \texttt{max}(\mathbf{y}). \tag{4.6}$$

So now write

$$\mathbf{x} = \{x_{\texttt{max}}\} \cup \mathbf{x}_{\text{rem}} \ ; \ \mathbf{y} = \{y_{\texttt{max}}\} \cup \mathbf{y}_{\text{rem}}, \tag{4.7}$$

where $x_{\texttt{max}} = \texttt{max}(\mathbf{x})$, and $y_{\texttt{max}} = \texttt{max}(\mathbf{y})$.

But now we have

$$\begin{aligned} \Phi_M(\mathbf{x}) &= \Phi_{M-1}(\mathbf{x}_{\text{rem}}) + \phi(x_{\texttt{max}}) \\ &= \Phi_{M-1}(\mathbf{y}_{\text{rem}}) + \phi(y_{\texttt{max}}) \\ &= \Phi_M(\mathbf{y}). \end{aligned}$$

From the central equality, and Equation (4.6), we have

$$\Phi_{M-1}(\mathbf{x}_{\text{rem}}) = \Phi_{M-1}(\mathbf{y}_{\text{rem}}).$$

Now by injectivity of Φ_{M-1}, we have $\mathbf{x}_{\text{rem}} = \mathbf{y}_{\text{rem}}$. Combining this with Equations (4.6) and (4.7), we must have $\mathbf{x} = \mathbf{y}$, and so Φ_M is injective. □

Equipped with this lemma, we can now prove Theorem 4.13.

Proof. We proceed by contradiction. Suppose that functions ϕ and ρ exist satisfying Equation (4.5). Define $\Phi_M : \mathbb{R}^M \to \mathbb{R}^N$ by

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

Denote by $\mathbb{R}^M_{\text{ord}}$ the set of all $\mathbf{x} \in \mathbb{R}^M$ such that $x_1 < x_2 < \cdots < x_M$. Let Φ^{ord}_M be the restriction of Φ_M to $\mathbb{R}^M_{\text{ord}}$. Since Φ^{ord}_M is a sum of continuous functions, it is also continuous, and by Lemma 4.16, 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 Φ^{ord}_M can be used to construct a continuous injection from $\mathbb{R}^M$ to $\mathbb{R}^N$. But it is known that no such continuous injection exists when $M > N$. Therefore our decomposition from Equation (4.5) cannot exist. □

Implications and Limitations

In light of Theorems 4.13 and 4.14, we now have a necessary and sufficient condition for universal function representation with Deep Sets. A Deep Sets-based model is capable of representing all continuous permutation-invariant functions on sets of size at most M if and only if the model's latent space is at least M-dimensional. We now provide a brief discussion of the implications of this result and the limitations which must be taken into account when considering how the result applies in practice.

We can concisely summarise the practical implications of Theorems 4.13 and 4.14 as follows. When deploying a Deep Sets-based model, the choice of latent space should depend on the cardinality of the sets being processed. **Larger input sets demand a larger latent space**. We emphasise that, although our results give a precise figure for the necessary and sufficient latent dimension M, this should not be interpreted as meaning that exactly M dimensions should always be used in practice, for the following reasons.

First note that Theorem 4.13 does not imply that *all* functions require an M-dimensional latent space. Some functions can be represented in a lower dimensional space – take for instance the sum of all elements, which is trivially represented by

setting both ϕ and ρ to be the identity function. The statement rather says that *some* functions require an M-dimensional latent space. It is still possible that, in a given application, the target function does not require M dimensions. While we do not characterise exactly which functions require an M-dimensional sum-decomposition, we do note that such functions need not be "badly-behaved" or difficult to specify. Our proof specifically demonstrates that even `max`, which is trivial to specify, is not continuously sum-decomposable with a latent space dimension less than M.

Second, although Theorem 4.14 shows that an M-dimensional latent space suffices to model any target function, we also know from Theorem 4.13 that M dimensions is the *bare minimum* needed to guarantee this property. In practice, a model using only this minimum capacity of M dimensions is not guaranteed to provide good results. One reason for this is that, while the necessary functions ϕ and ρ certainly *exist*, we have no guarantee that they can be successfully *learned*. It is entirely possible, even in light of our results, that increasing the latent dimension above M leads to more efficient training and superior results in practice.

A final, major weakness of Theorem 4.13 is that, in following the mathematical framework developed in Zaheer et al. [12], it addresses function *representation* rather than function *approximation*. This leaves open the possibility that, while an M-dimensional latent space is needed for exact representation, we can achieve arbitrarily good approximation of any target function using only a 1-dimensional latent space. Addressing this weakness requires significantly more complicated mathematics, and we devote the next section to proving that an analogue of Theorem 4.13 does indeed hold for universal approximation, and M dimensions are still required.

4.4 Universal Function Approximation with Deep Sets

In the previous section, we considered the problem of function *representation* – showing that there exists an instance of a given model which exactly computes a target function. But exactly representing target functions is generally *not* our goal – typically, we are concerned with *approximating* a target function. In this

section, we extend our analysis of the Deep Sets architecture to assess its ability to perform approximation.

To see concretely why function approximation requires a separate analysis, we return to the `max` function on a set of M elements. As proved above, for an exact sum-decomposition, the latent space must be at least M-dimensional. But as noted by [12] in an appendix to their work, `max` can be approximated arbitrarily well with only a 2-dimensional latent space. In fact, one dimension will suffice. Let $a \in \mathbb{R}$ and set ϕ and ρ as follows:

$$\phi(x) = e^{ax}, \qquad \rho(x) = \frac{\log x}{a}. \tag{4.8}$$

Thus, our sum-decomposition approximation for `max` is

$$\widehat{f}_a(X) = \frac{1}{a} \log \left(\sum_{x \in X} e^{ax} \right). \tag{4.9}$$

For input sets of size M, it is easily shown that

$$\texttt{max}(X) \leq \widehat{f}_a(X) \leq \texttt{max}(X) + \frac{\log M}{a}.$$

That is, the worst-case error in this approximation is $\frac{\log M}{a}$. For fixed set size M, we can therefore achieve arbitrarily good approximation by increasing the value of a, even though we have only used a 1-dimensional latent space.

In the specific case of `max`, this demonstrates that there is a wide gap between what is necessary for representation and what is necessary for approximation. `max` is in some sense "as hard as possible" to represent, requiring M latent dimensions, but "as easy as possible" to approximate, requiring only 1 latent dimension. This naturally raises the question of whether the approximation problem is somehow fundamentally easier. We state a more precise form of this question as follows: *is it the case that every permutation-invariant function of M elements has an approximate sum-decomposition via fewer than M dimensions*? Our analysis answers this question in the negative – there exist functions which cannot be closely approximated with a lower-dimensional latent space. Moreover, low-dimensional sum-decomposition is guaranteed to fail very badly for these functions – in terms of worst-case error, approximation with sum-decomposition performs as badly as approximation with a constant function.

4.4.1 A Necessary Condition For Function Approximation by Sum-Decomposition

The question above mentions *approximate sum-decomposition* – in order to state and prove our result, we must define precisely what we mean by this. In the following definitions, let M be a positive integer, $U \subset \mathbb{R}^M$ be compact,[9] and $f : U \to \mathbb{R}$. We maintain the convention that $\Phi(\mathbf{x}) = \sum_{i=1}^{M} \phi(x_i)$.

Definition 4.17. Let $\epsilon > 0$. (ϕ, ρ) is a *within-ϵ sum-decomposition* of f if $|f(\mathbf{u}) - \rho(\Phi(\mathbf{u}))| < \epsilon$ for every $\mathbf{u} \in U$.

For example, if we let $a = \frac{\log M}{\epsilon}$, then the sum-decomposition defined by Equation (4.8) is a within-ϵ continuous sum-decomposition of `max` via $\mathbb{R}$.

Definition 4.18. A sequence $(\phi, \rho)_k = \{(\phi_k, \rho_k); k \in \mathbb{N}\}$ is an *approximate sum-decomposition* of f if, for any $\epsilon > 0$, there is some $K \in \mathbb{N}$ such that (ϕ_K, ρ_K) is a within-ϵ sum-decomposition of f. We also require that (ϕ_k, ρ_k) is a within-ϵ sum-decomposition of f for every $k \geq K$. Put more loosely, (ϕ_k, ρ_k) is a sequence of ever-closer approximations to f. The existence of an approximate sum-decomposition of f guarantees that f can be approximated arbitrarily closely by sum-decomposition.

Given a set Y, we say that the approximate sum-decomposition $(\phi, \rho)_k$ is *via* Y if (ϕ_k, ρ_k) is via Y for every k.

For example, letting $a = 1, 2, \ldots$, the sequence $(\phi, \rho)_a$ as defined by Equation (4.8) is a continuous approximate sum-decomposition of `max` via $\mathbb{R}$.

With these definitions in hand, we now state our main result on approximation.

Theorem 4.19. *Let $M, N \in \mathbb{N}$ with $M > N$, and let $I_M \coloneqq [-1, 1]^M \subset \mathbb{R}^M$. Then there exists a continuous permutation-invariant function $f : I_M \to \mathbb{R}$ which has no continuous approximate sum-decomposition via $\mathbb{R}^N$.*

[9] This compactness requirement is necessary when discussing universal function approximation. Of particular relevance here is the fact that the universal approximation theorem for neural networks also requires a compact domain [49, 52, 53]. This compactness requirement is a fundamental constraint which is also necessary for other function approximation results, for example the Stone-Weierstrass theorem [51].

Our proof of this theorem actually provides stronger conclusions than stated above. The following paragraphs explain these additional conclusions in detail, and a theorem statement incorporating these conclusions is included in Appendix A.3.7.

Since the target function f is continuous on a compact domain, we know that its output is bounded, say by $y_{\min} \leq f(\mathbf{u}) \leq y_{\max}$. A worst-case error[10] of $E \coloneqq \frac{y_{\max}-y_{\min}}{2}$ can trivially be achieved by approximating f with the constant function $\widehat{f}(\mathbf{u}) = \frac{y_{\max}+y_{\min}}{2}$. We show that there exist permutation-invariant functions f_* on $\mathbb{R}^M$ which have no within-E continuous sum-decomposition via $\mathbb{R}^N$. In other words, any attempt to sum-decompose f_* via $\mathbb{R}^N$ is doomed to have the same worst-case error as simply approximating f_* by a constant value.

In addition, we explicitly construct a permutation-invariant function f_* for which sum-decomposition via $\mathbb{R}^N$ must have worst-case error E as defined above. Letting $U = [\text{-}1, 1]^M$, this f_* has a simple form:[11]

$$\begin{aligned} f_*(\mathbf{u}) &\coloneqq \mathbf{w}^\intercal \texttt{sort}(\mathbf{u}) + b\,, \\ w_i &\coloneqq (\text{-}1)^{i+1}\,, \\ b &\coloneqq \begin{cases} \text{-}1 & M \text{ even} \\ 0 & M \text{ odd} \end{cases}. \end{aligned} \tag{4.10}$$

Here, `sort` sorts the elements of $\mathbf{u}$ in descending order. This definition may look a little arbitrary, but we will justify it in Section 4.4.2 and provide some intuition for why it is hard to approximate in Section 4.4.3. The function f_* is trivial to represent using the sorting framework mentioned in Section 4.2,[12] but impossible to approximate using a continuous sum-decomposition via $\mathbb{R}^N$. This gives mathematical support to the proposition that there is no true "one size fits all" approach – for a given task, performance may vary greatly between different methods of achieving permutation invariance.

Finally we note that poor approximation behaviour is not confined to a single point. This follows from the fact that both f_* and the sum-decomposition are

[10] By worst-case error we mean $\max_{\mathbf{u}} |f(\mathbf{u}) - \widehat{f}(\mathbf{u})|$.

[11] The extra -1 for even M is to ensure that f_* is bounded between -1 and 1. This isn't necessary, indeed the bias term could be removed from Equation (4.10) without affecting our result, but reasoning about f_* is simpler if its range is fixed.

[12] We discuss sorting in more detail in Section 4.5.1.

continuous, and so the approximation error also varies continuously – if the error is high at one point, it must also be high in some region around that point.

Having stated the key conclusions, we now present the proof of Theorem 4.19. We omit some details, particularly in the proof of Lemma 4.23. Full details are included in Appendix A.3.8.

4.4.2 Proving Necessity for Universal Approximation

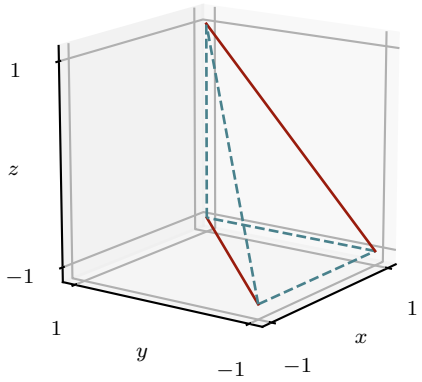
The overall proof strategy is best understood by thinking topologically – that is, by considering how the encoding map Φ and the target function f deform the input space, squashing it onto lower-dimensional output spaces. To understand this perspective we will consider an example with $M = 3$ and $N = 2$, since this is high-dimensional enough to illustrate the key points and low-dimensional enough to visualise. For this example, we choose ϕ to send $[-1, 1]$ to a semicircle. This choice is made purely for easy visualisation of ϕ and Φ.

Any continuous target function $f : [-1, 1]^3 \to \mathbb{R}$ may be thought of as deforming a cube into a line segment. We actually do not need to consider the whole cube – since f is permutation-invariant, it is fully determined by its behaviour on the set of points with descending-ordered coordinates. In general we will denote this set Δ_n,

$$\Delta_n \coloneqq \{\mathbf{x} \in \mathbb{R}^n : 1 \geq x_1 \geq x_2 \geq \ldots \geq x_n \geq -1\}\,. \tag{4.11}$$

Considering Δ_3 instead of the whole cube allows us to think more freely about f, without worrying about permutation invariance – as long as f is continuous on Δ_3, it can be extended in a continuous and permutation-invariant way to $[-1, 1]^3$. In general, if we want to define a function f by its behaviour on Δ_n and implicitly extend it to the rest of the space, we can simply write it as a function of $\texttt{sort}(\mathbf{x})$. This accounts for the appearance of `sort` in Equation (4.10), which can be thought of as saying "$f_*(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x} + b$ on Δ_M, and f_* is permutation-invariant".

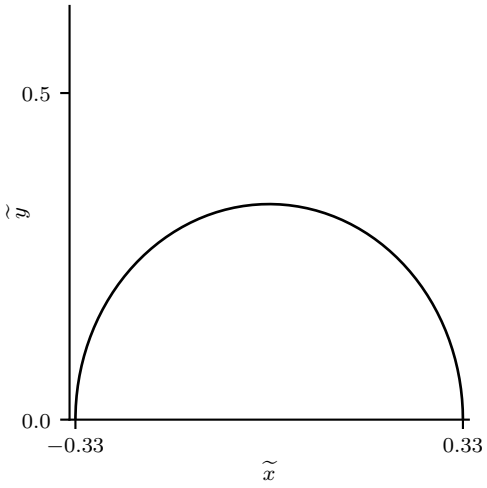
In general Δ_n is an n-simplex, so Δ_3 is a tetrahedron. From our topological perspective, sum-decomposition via $\mathbb{R}^2$ attempts to break down the tetrahedron-to-line deformation f by first sending Δ_3 onto the plane under Φ, then sending the



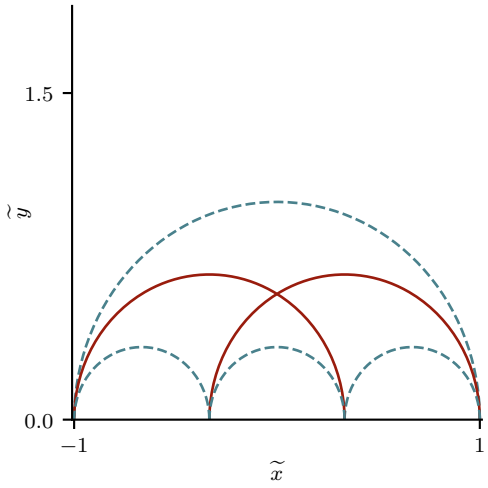
(a) The boundary of Δ_3.



(b) The image of Δ_3 under f_*.



(c) The image of [-1, 1] under ϕ.



(d) The image of Δ_3 under Φ.

Figure 4.3: A visualisation of sum-decomposition when the input sets are of size 3 and the latent space is 2-dimensional. For the purposes of this illustration, we pick ϕ to send the interval [-1, 1] to a semicircle. With this choice of ϕ, we see that two disjoint regions of the surface of Δ_3, highlighted with solid red lines, intersect under the application of Φ. By contrast, f_* separates these highlighted regions, sending them to opposite ends of the interval [-1, 1]. Since these regions are already "mixed together" by Φ, there is no function ρ which can separate them out again to approximate f_*, so sum-decomposition of f_* fails. Our proof demonstrates that an intersection under Φ which causes this problem for f_* is forced for any continuous ϕ and in any number of dimensions, as long as the dimension of the latent space is less than the number of inputs.

plane onto a line under ρ. Figures 4.3a, 4.3c and 4.3d visualise the action of Φ, showing how the boundaries of Δ_3 are mapped to the plane for our example choice of ϕ. Crucially, note that some of these boundaries intersect under Φ. Once two regions have been brought together by Φ, they cannot be separated again – ρ cannot send the point of intersection to two places at once. Therefore if f *does* separate these regions, sending them to opposite ends of [-1, 1] as illustrated in Figure 4.3b, then the sum-decomposition $\rho \circ \Phi$ must fail with worst-case error at least 1.

Figure 4.3 only shows that this intersection problem occurs for our example choice of ϕ and M, but the general case also holds – for every M, and for any continuous ϕ which maps into $\mathbb{R}^{M-1}$, there are two disjoint regions of the surface of Δ_M which intersect under Φ. We denote the two regions X_M^{+1} and X_M^{-1} – the solid red regions in Figure 4.3a are X_3^{+1} and X_3^{-1}. Our f_* from Equation (4.10) is defined so that it separates X_M^{+1} and X_M^{-1}, and it is just one among many continuous functions which have this property. In some sense it is the simplest, since it is affine on Δ_M. Our proof that $\Phi(X_M^{+1})$ and $\Phi(X_M^{-1})$ always intersect relies on the structure of Φ as a sum over inputs.

We define X_M^{+1} and X_M^{-1} as follows:

$$X_M^{+1} \coloneqq \{\mathbf{x} \in \Delta_M : 1 = x_1 \geq x_2 = x_3 \geq \ldots \overset{=}{\geq} x_M \overset{\geq}{=} -1\} \,, \tag{4.12}$$

$$X_M^{-1} \coloneqq \{\mathbf{x} \in \Delta_M : 1 \geq x_1 = x_2 \geq x_3 = \ldots \overset{\geq}{=} x_M \overset{=}{\geq} -1\} \,. \tag{4.13}$$

The rightmost equality $x_M = -1$ applies to X_M^{+1} when M is even, and X_M^{-1} when M is odd. We can view these equations as modifying the definition of Δ_M (Equation 4.11) by changing either the odd- or the even-numbered $\geq$ signs into $=$ signs. It follows directly from the definition of f_* (Equation 4.10) that $f_*(X_M^{+1}) = 1$ and $f_*(X_M^{-1}) = -1$.

The above discussion can be summarised by the following claim:

Claim 4.20. *Let $N = M - 1$. There exist two subsets X_M^{+1} and X_M^{-1} of Δ_M such that:*

1. *f_* **separates** X_M^{+1} **from** X_M^{-1}**:** $f_*(X_M^{+1}) = 1$ and $f_*(X_M^{-1}) = -1$.*

2. Φ ***cannot separate*** X_M^{+1} ***from*** X_M^{-1}***:*** *For any continuous function* $\phi : [-1, 1] \to \mathbb{R}^N$*, the set* $\Phi(X_M^{-1}) \cap \Phi(X_M^{+1})$ *is non-empty.*

Theorem 4.19 follows immediately, since the above points imply that f_* *has no within-1 continuous sum-decomposition via* $\mathbb{R}^N$*.*

As noted, point 1 is true by definition, so we only need to prove point 2 to complete the proof of Theorem 4.19. The proof of point 2 has two parts:

1. **Lemma 4.21**: Construct a function Γ_N which has a zero only if the set $\Phi(X_M^{-1}) \cap \Phi(X_M^{+1})$ is non-empty.

2. **Lemma 4.23:** Show that Γ_N has a zero.

The main idea in constructing Γ_N is to subtract $\Phi(X_M^{-1})$ from $\Phi(X_M^{+1})$. This clearly evaluates to zero only where the two sets intersect. The subscript on Γ is N, rather than M, because there are a total of $N = M - 1$ degrees of freedom in X_M^{+1} and X_M^{-1} – this is most easily seen by considering $M = 1$, which gives zero degrees of freedom.[13] This subtraction idea is slightly obscured in the formal statement and proof of the following Lemma, but it is the key idea on which the Lemma is based.

Straightforwardly using this idea results in a function whose input space is $X_M^{+1} \times X_M^{-1}$, but we will instead define Γ_N to have input space Δ_N. As shown below, any point $\mathbf{z} \in \Delta_N$ may be used to construct a pair of points $(\mathbf{x}^+, \mathbf{x}^-) \in X_M^{+1} \times X_M^{-1}$, and the subtraction idea can then be applied to this pair of points. We define Γ_N in this way because Δ_N turns out to have useful structure which we can exploit for the proof of Lemma 4.23.

Importantly, the following statement of Lemma 4.21 requires that $\phi(-1) = \mathbf{0}$. From now on we will assume this without loss of generality, since otherwise we can reason identically about $\widetilde{\phi}(x) \coloneqq \phi(x) - \phi(-1)$, which does satisfy $\widetilde{\phi}(-1) = \mathbf{0}$.

[13] Equations (4.12) and (4.13) imply that $X_1^{+1} = \{1\}$ and $X_1^{-1} = \{-1\}$.

Lemma 4.21. *Let* $\phi : [-1, 1] \to \mathbb{R}^N$, *with* $\phi(-1) = \mathbf{0}$. *Define* $\Gamma_N : \Delta_N \to \mathbb{R}^N$ *by*

$$\Gamma_N(\mathbf{x}) \coloneqq \sum_{i=1}^{N} (-1)^i \phi(x_i) + \frac{\phi(1)}{2}. \tag{4.14}$$

Suppose Γ_N *has a zero in* Δ_N. *Then there exist* $\mathbf{x}^+ \in X_M^{+1}$, $\mathbf{x}^- \in X_M^{-1}$ *with*

$$\Phi(\mathbf{x}^+) = \Phi(\mathbf{x}^-).$$

Proof. Let $\mathbf{z} \in \Delta_N$ with $\Gamma_N(\mathbf{z}) = \mathbf{0}$. Define $\mathbf{x}^+ \in X_M^{+1}$ and $\mathbf{x}^- \in X_M^{-1}$:

$$\begin{aligned} x_i^+ &= z_i \quad \text{for even } i\,, \\ x_i^- &= z_i \quad \text{for odd } i\,. \end{aligned} \tag{4.15}$$

The remaining coordinates of $\mathbf{x}^+$ and $\mathbf{x}^-$ are fixed by the equalities in Equations (4.12) and (4.13). Note in particular that Equation (4.12) implies that $x_1^+ = 1$. Taking these constraints together with the condition that $\phi(-1) = \mathbf{0}$, we obtain

$$\begin{aligned} \Phi(\mathbf{x}^+) &= \phi(1) + 2 \sum_{i \text{ even}} \phi(x_i^+)\,, \\ \Phi(\mathbf{x}^-) &= 2 \sum_{i \text{ odd}} \phi(x_i^-)\,. \end{aligned} \tag{4.16}$$

Plugging $\mathbf{z}$ into Equation (4.14) and pulling the odd terms of the sum to the left hand side, we obtain

$$\sum_{i \text{ odd}} \phi(z_i) = \sum_{i \text{ even}} \phi(z_i) + \frac{\phi(1)}{2}. \tag{4.17}$$

Now we apply Equation (4.15) to obtain

$$\sum_{i \text{ odd}} \phi(x_i^-) = \sum_{i \text{ even}} \phi(x_i^+) + \frac{\phi(1)}{2}. \tag{4.18}$$

This equates the right hand sides of the two lines of Equation (4.16), so the left hand sides must also be equal, i.e. $\Phi(\mathbf{x}^+) = \Phi(\mathbf{x}^-)$.

□

All that remains is to show that Γ_N has a zero. To understand Γ_N, we exploit the alternating sum form of Equation (4.14), and in particular we will see how this constrains the behaviour of Γ_N on the surface of Δ_N. Each face of the surface is defined by changing one of the inequalities in Equation (4.11) into an equality, i.e. by applying the constraint $x_i = x_{i+1}$. We denote the corresponding face $\Delta_N^{(i)}$. For $1 \leq i < N$, this constraint leads to the cancellation of two consecutive terms of the sum in Equation (4.14). Each of these faces therefore has the same image under Γ_N – for example, the following points belong to different faces of Δ_4, but because equal pairs cancel out they all go to the same point under Γ_4.

$$\left(\tfrac{2}{3}, \tfrac{2}{3}, \tfrac{1}{2}, 0\right) \in \Delta_4^{(1)} \qquad \left(\tfrac{1}{2}, \tfrac{1}{3}, \tfrac{1}{3}, 0\right) \in \Delta_4^{(2)} \qquad \left(\tfrac{1}{2}, 0, \text{-}\tfrac{1}{2}, \text{-}\tfrac{1}{2}\right) \in \Delta_4^{(3)}$$

There are two more faces to consider – $\Delta_N^{(0)}$ and $\Delta_N^{(N)}$, corresponding to the constraints $x_1 = 1$ and $x_N = \text{-}1$ respectively. We can observe an interesting behaviour of Γ_N here, which is in fact a symmetry: if we apply a "coordinate left shift" to a point in $\Delta_N^{(0)}$, the output of Γ_N is multiplied by -1. By "coordinate left shift", we mean the bijection $\alpha : \Delta_N^{(0)} \to \Delta_N^{(N)}$ defined as follows:

$$\alpha(\mathbf{x})_i = \begin{cases} x_{i+1} & i < N \\ \text{-}1 & i = N \end{cases}.$$

Applying α moves each coordinate one place to the left, deleting the 1 in the leftmost place and padding with a -1 in the rightmost place. For example, consider the point $(1, \frac{2}{3}, 0, \text{-}\frac{1}{2}) \in \Delta_4^{(0)}$.

$$\alpha\left((1, \tfrac{2}{3}, 0, \text{-}\tfrac{1}{2})\right) = (\tfrac{2}{3}, 0, \text{-}\tfrac{1}{2}, \text{-}1) \in \Delta_4^{(4)}$$

Now apply Γ_4 to $(1, \frac{2}{3}, 0, \text{-}\frac{1}{2})$ and $(\frac{2}{3}, 0, \text{-}\frac{1}{2}, \text{-}1)$.

$$\begin{aligned} \Gamma_4\left((1, \tfrac{2}{3}, 0, \text{-}\tfrac{1}{2})\right) &= -\phi(1) + \phi(\tfrac{2}{3}) - \phi(0) + \phi(\text{-}\tfrac{1}{2}) + \frac{\phi(1)}{2} \\ \Gamma_4\left((\tfrac{2}{3}, 0, \text{-}\tfrac{1}{2}, \text{-}1)\right) &= \qquad -\phi(\tfrac{2}{3}) + \phi(0) - \phi(\text{-}\tfrac{1}{2}) + \phi(\text{-}1) + \frac{\phi(1)}{2} \end{aligned}$$

Simplifying the $\phi(1)$ terms, and recalling that $\phi(\text{-}1) = 0$, we obtain

$$\Gamma_4\left((\tfrac{2}{3}, 0, \text{-}\tfrac{1}{2}, \text{-}1)\right) = -\Gamma_4\left((1, \tfrac{2}{3}, 0, \text{-}\tfrac{1}{2})\right).$$

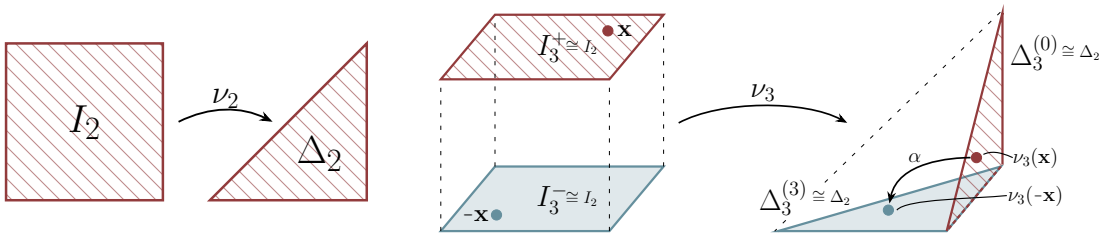


Figure 4.4: For the first part of the construction of ν_n, we use ν_{n-1} to define ν_n on the top and bottom faces of I_n. The $n = 3$ case is illustrated here. ν_n uses ν_{n-1} to send the top face of I_n to $\Delta_n^{(0)}$, and the bottom face to $\Delta_n^{(n)}$. We construct ν_n so that opposite points $\mathbf{x}$ and -$\mathbf{x}$ on the top and bottom faces go to points in Δ_n which are related by the left-shift function α – that is, $\nu_n(\text{-}\mathbf{x}) = \alpha(\nu_n(\mathbf{x}))$.

It is easily seen that this generalises to any $\mathbf{x} \in \Delta_N^{(0)}$:

$$\Gamma_N\Big(\alpha(\mathbf{x})\Big) = -\Gamma_N\Big(\mathbf{x}\Big)\,. \tag{4.19}$$

We can exploit this symmetry to show that Γ_N has a zero. To do this, we will need the Borsuk-Ulam theorem (Borsuk [54] – see Lloyd [55] for a detailed English-language treatment). One statement of the theorem is as follows.

Theorem 4.22 (Borsuk-Ulam)**.** *Let $I_n \coloneqq [\text{-}1, 1]^n$ be the unit n-cube. Let ∂I_n be the surface of I_n, $\partial I_n \coloneqq \{\mathbf{x} \in I_n : x_i = \pm 1 \text{ for some } i\}$. Suppose $f : I_n \to \mathbb{R}^n$ is continuous, and for every $\mathbf{x} \in \partial I_n$ we have*

$$f(\text{-}\mathbf{x}) = -f(\mathbf{x})\,. \tag{4.20}$$

Then there is some $\mathbf{z} \in I_n$ such that $f(\mathbf{z}) = 0$.

There is a clear similarity between Equations (4.19) and (4.20), but we cannot directly apply Borsuk-Ulam to Γ_N. We need to bridge the two equations, relating the left-shift α in Equation (4.19) to the negation on the left hand side of Equation (4.20). We can do this by finding a continuous function $\nu_N : I_N \to \Delta_N$ which turns negation on ∂I_N into a left-shift on $\partial\Delta_N$:[14]

$$\nu_N(\text{-}\mathbf{x}) = \alpha(\nu_N(\mathbf{x}))\,. \tag{4.21}$$

[14]Here, we are glossing over the fact that α is only defined on part of $\partial\Delta_N$. We will come back to this when proving Lemma 4.23.

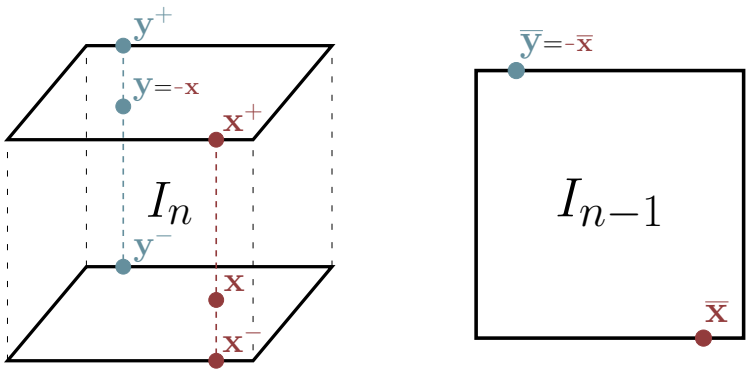


Figure 4.5: For the second part of the construction of ν_n, we note that any point $\mathbf{x}$ lies on a "vertical" line, i.e. an n-th-axis-aligned line, meeting the top face at $\mathbf{x}^+$ and the bottom face at $\mathbf{x}^-$. We complete the construction of ν_n by interpolating along these lines. For a pair of opposite points $\mathbf{y} = \text{-}\mathbf{x}$, we see that $\mathbf{y}^- = \text{-}\mathbf{x}^+$. To ensure that ν_n has the desired properties, we must pay particular attention to the surface of I_n without the top and bottom faces, i.e. $\partial I_n \backslash (I_n^+ \cup I_n^-)$. If $\mathbf{x}$ lies on this part of the surface (as in this figure), then the point $\overline{\mathbf{x}}$, which is the projection of $\mathbf{x}$ along the vertical, lies on the surface ∂I_{n-1}.

That is, we want to be able to apply Borsuk-Ulam to $\Gamma_N \circ \nu_N$:

$$\begin{aligned} f(\text{-}\mathbf{x}) \coloneqq \Gamma_N\Big(\nu_N(\text{-}\mathbf{x})\Big) &= \Gamma_N\Big(\alpha\big(\nu_N(\mathbf{x})\big)\Big) \\ &= -\Gamma_N\Big(\nu_N(\mathbf{x})\Big) = -f(\mathbf{x})\,. \end{aligned}$$

Showing that a suitable ν_N exists will complete the proof, since Theorem 4.22 then immediately implies that Γ_N has a zero.

Lemma 4.23. *Let $n \in \mathbb{N}$. Let Γ_n be defined as in Equation (4.14). Then there exists a continuous function $\nu_n : I_n \to \Delta_n$ such that, for $\mathbf{x} \in \partial I_n$*

$$\Gamma_n\Big(\nu_n(\text{-}\mathbf{x})\Big) = -\Gamma_n\Big(\nu_n(\mathbf{x})\Big)\,. \tag{4.22}$$

We present a partial proof here, with full details available in Appendix A.3.8.

Proof. First, we must point out that the proof strategy is not quite as simple as suggested above. Since α is only defined on $\Delta_n^{(0)}$, and not on the rest of $\partial\Delta_n$, we cannot in fact satisfy Equation (4.21) for every $\mathbf{x}$ on ∂I_n. Nevertheless, we can still

satisfy Equation (4.22) for every $\mathbf{x}$ on ∂I_n, and we reach this conclusion by showing that Equation (4.21) holds on part of ∂I_n.

We proceed by induction – the base case $\nu_1(\mathbf{x}) \coloneqq \mathbf{x}$ is trivial. Consider the "top" and "bottom" faces of I_n.

$$I_n^+ \coloneqq \{\mathbf{x} \in I_n : x_n = 1\} \qquad \text{The top face.}$$
$$I_n^- \coloneqq \{\mathbf{x} \in I_n : x_n = \text{-}1\} \qquad \text{The bottom face.}$$

Note that I_n^+ and I_n^- are naturally equivalent to I_{n-1} – we can map bijectively from either face to I_{n-1} by dropping the last coordinate of $\mathbf{x}$. For any point $\mathbf{x}$, we'll denote the corresponding point in I_{n-1} by $\overline{\mathbf{x}}$ – that is, $\overline{\mathbf{x}}$ is $\mathbf{x}$ with the last coordinate removed.

Note also that $\Delta_n^{(0)}$ and $\Delta_n^{(n)}$ are naturally equivalent to Δ_{n-1} – any point $\mathbf{x} \in \Delta_n^{(0)}$ can be written as $(1, \mathbf{x}')$, where $\mathbf{x}' \in \Delta_{n-1}$. Similarly any point in $\Delta_n^{(n)}$ can be written as $(\mathbf{x}', \text{-}1)$. We can use these equivalences to the lower-dimensional case to come up with a partial definition of ν_n in terms of ν_{n-1}.

The idea of how to begin constructing ν_n using these equivalences is illustrated in Figure 4.4. In the following, we will choose the definition of ν_n on I_n^+ and I_n^- to satisfy what is shown in this figure. That is, by viewing I_n^+ as a copy of I_{n-1}, and $\Delta_n^{(0)}$ as a copy of Δ_{n-1}, we can map from I_n^+ to $\Delta_n^{(0)}$ using $\nu_{n-1} : I_{n-1} \to \Delta_{n-1}$. Crucially, note that the truth of Equation (4.21) for $\mathbf{x} \in I_n^+$ depends on how ν_n behaves on I_n^-. There is only one way to define ν_n on I_n^- so that Equation (4.21) is true on I_n^+, and this is the definition that we choose.

Written mathematically, we define ν_n on I_n^+ and I_n^- as follows:

$$\nu_n(\mathbf{x}) \coloneqq \begin{cases} \Big(1, \nu_{n-1}(\overline{\mathbf{x}})\Big) & \text{for } \mathbf{x} \in I_n^+ \,. \\ \alpha\Big(\nu_n(\text{-}\mathbf{x})\Big) = \Big(\nu_{n-1}(\text{-}\overline{\mathbf{x}}), \text{-}1\Big) & \text{for } \mathbf{x} \in I_n^- \,. \end{cases}$$

So by definition, ν_n satisfies Equation (4.21) for $\mathbf{x} \in I_n^+$. This immediately implies Equation (4.22) on I_n^+ and, by symmetry, on I_n^-.

We have established that Equation (4.22) holds on $I_n^+ \cup I_n^-$ for our partially defined ν_n. We now need to extend ν_n to the rest of I_n. We will define this extension

by interpolating along the vertical lines joining I_n^+ and I_n^-. By "vertical", we mean parallel to the n-th coordinate axis.

Consider a point $\mathbf{x}$ lying on the part of ∂I_n that we have not yet considered, i.e. $\partial I_n \backslash (I_n^+ \cup I_n^-)$. Any such point lies on a vertical line which meets ∂I_n^+ and ∂I_n^- at points $\mathbf{x}^+$ and $\mathbf{x}^-$, as depicted in Figure 4.5. We refer to these vertical line segments on ∂I_n as *surface verticals*. Crucially, $\partial I_n \backslash (I_n^+ \cup I_n^-)$ is entirely covered by surface verticals.

We want to ensure that Equation (4.22) holds on all of ∂I_n, and because we have already shown this for $I_n^+ \cup I_n^-$, it only remains to ensure that it holds on all of the surface verticals. One way to achieve this is to make $\Gamma_n \circ \nu_n$ constant on each surface vertical. To see that this would imply Equation (4.22), consider a pair of antipodal[15] points $\mathbf{x}$ and $\mathbf{y}$ on $\partial I_n \backslash (I_n^+ \cup I_n^-)$. As illustrated in Figure 4.5, $\mathbf{x}^+$ and $\mathbf{y}^-$ are also antipodal. Since $\mathbf{x}^+$ and $\mathbf{y}^-$ belong to $I_n^+ \cup I_n^-$, we know that Equation (4.22) holds:

$$\Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big) = -\Gamma_n\Big(\nu_n(\mathbf{y}^-)\Big)\,.$$

If $\Gamma_n \circ \nu_n$ is constant on surface verticals, it follows that $\Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big) = \Gamma_n\Big(\nu_n(\mathbf{x})\Big)$ and $\Gamma_n\Big(\nu_n(\mathbf{y}^-)\Big) = \Gamma_n\Big(\nu_n(\mathbf{y})\Big)$, and therefore Equation (4.22) also holds for $\mathbf{x}$ and $\mathbf{y}$:

$$\Gamma_n\Big(\nu_n(\mathbf{x})\Big) = \Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big) = -\Gamma_n\Big(\nu_n(\mathbf{y}^-)\Big) = -\Gamma_n\Big(\nu_n(\mathbf{y})\Big)\,.$$

Because the ends of a surface vertical, $\mathbf{x}^+$ and $\mathbf{x}^-$, belong to I_n^+ and I_n^-, the behaviour here is already fixed by our partial definition of ν_n. Therefore, to have any hope of holding $\Gamma_n \circ \nu_n$ constant on surface verticals, we must first show that $\Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big) = \Gamma_n\Big(\nu_n(\mathbf{x}^-)\Big)$ under our existing partial definition of ν_n.

By expanding the definition of Γ_n, consider $\Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big)$:

$$\begin{aligned}
\Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big) &= \Gamma_n\Big(\Big(1, \nu_{n-1}(\overline{\mathbf{x}})\Big)\Big) \\
&= -\phi(1) - \Big(\Gamma_{n-1}\Big(\nu_{n-1}(\overline{\mathbf{x}})\Big) - \frac{\phi(1)}{2}\Big) + \frac{\phi(1)}{2} \\
&= -\Gamma_{n-1}\Big(\nu_{n-1}(\overline{\mathbf{x}})\Big)\,.
\end{aligned}$$

[15] That is, such that $\mathbf{y} = -\mathbf{x}$.

Similarly we find that for $\mathbf{x}^-$:

$$\Gamma_n\Big(\nu_n(\mathbf{x}^-)\Big) = \Gamma_{n-1}\Big(\nu_{n-1}(\text{-}\overline{\mathbf{x}})\Big)\,.$$

But note that $\overline{\mathbf{x}} \in \partial I_{n-1}$, as illustrated in Figure 4.5. By induction, Equation (4.22) therefore applies:

$$\begin{aligned}\Gamma_{n-1}\Big(\nu_{n-1}(\text{-}\overline{\mathbf{x}})\Big) &= -\Gamma_{n-1}\Big(\nu_{n-1}(\overline{\mathbf{x}})\Big)\,,\\ \Gamma_n\Big(\nu_n(\mathbf{x}^-)\Big) &= \Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big)\,.\end{aligned} \tag{4.23}$$

We have shown that $\Gamma_n \circ \nu_n$ takes the same value at either end of each surface vertical. It remains to extend ν_n so that $\Gamma_n \circ \nu_n$ remains constant along the entire surface vertical. We can perform this extension by interpolating along the vertical between I_n^+ and I_n^-. Linear interpolation does *not* respect this condition of constancy – the definition and justification of the correct interpolation scheme are given in Appendix A.3.8.

□

With ν_N in hand, the proof of Theorem 4.19 is complete. We briefly summarise the chain of reasoning from this section, which is now fully justified by the proof of the above Lemma.

1. We are attempting to model permutation-invariant functions on the input space $[\text{-}1, 1]^M$ using sum-decomposition, where the dimension of the latent space, N, is one less than that of the input space, M.

2. We have shown that, for any choice of continuous encoding map $\phi : [\text{-}1, 1] \to \mathbb{R}^N$, the function Γ_N (which depends on ϕ) has a zero.

3. Because of the way Γ_N was constructed, this implies that there are two disjoint regions of the input space[16] $[\text{-}1, 1]^M$ which collide under the application of Φ.

[16]Strictly speaking, two disjoint regions of the ordered input space Δ_M.

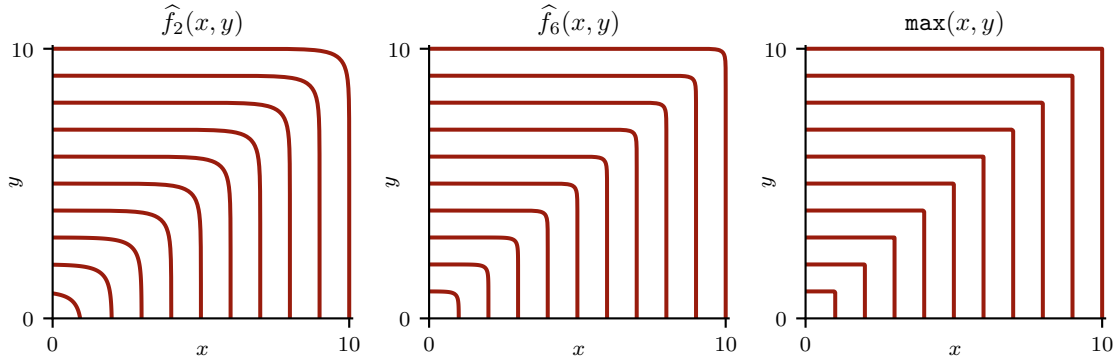


Figure 4.6: Contours of sum-decomposition approximations to `max` compared to the true `max` function. The sum-decompositions $\widehat{f_2}$ and $\widehat{f_6}$ are defined by Equation (4.9), with $a = 2, 6$ respectively. As a increases, the corners become sharper, but never perfectly sharp.

4. Any target function which widely separates these regions is therefore impossible to approximate by sum-decomposition – there will always be a pair of points which collide under the sum-decomposition but are widely separated by the target.

5. One example of such a function, which therefore cannot be approximated via $\mathbb{R}^N$, is given by Equation (4.10).

4.4.3 Some Visual Intuition

We have shown mathematically that there exist functions which cannot be approximately sum-decomposed via a low-dimensional latent space. In contrast to the heavy mathematical content above, we now provide some less formal intuition as to why this particular function is so hard to approximate. To that end, we will gloss over some details for the sake of readability.

We consider the contours of the encoding map $\Phi = \sum \phi$. This provides a useful visual perspective on sum-decomposition, helping to develop some intuition for the behaviour of functions represented in this form. Contours of Φ must also be contours of the full sum-decomposition $\rho \circ \Phi$: if $\Phi(X) = \Phi(Y)$ then $\rho \circ \Phi(X) = \rho \circ \Phi(Y)$.

We plot the contours of `max` in Figure 4.6, as well as the contours of two terms from an approximate sum-decomposition of `max` via $\mathbb{R}$. As can be seen here, the

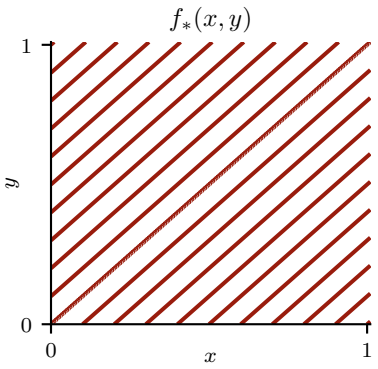


Figure 4.7: Contours of the function f_* as defined by Equation (4.10).

contours of `max` have sharp corners on the line $y = x$, whereas the contours of the approximations have a soft corner here. It is easily seen that a sum-decomposition via $\mathbb{R}$ cannot have the same sharp contours as `max` – we can push the contours as close to the corner as desired, providing ever-better approximation, but we cannot make the contours sharp like the true function. This picture of pushing the sum-decomposition contours closer to the true contours gives some intuition for why arbitrarily close approximation can be possible even when exact representation is not.

This contour perspective can also give some intuition for the case we have addressed in this section, where even approximation is not possible. As long as the sum-decomposition $\widehat{f}$ is differentiable and goes via $\mathbb{R}$, any contour of $\widehat{f}$ which meets the line $y = x$ must do so at right angles, as is the case for the sum-decompositions plotted in Figure 4.6.[17] `max` does not violate this condition so badly that we cannot approximate it – its contours do at least cross $y = x$. But as Figure 4.7 shows, the function f_* from our proof of Theorem 4.19 violates this condition very badly, having a contour along $y = x$, perpendicular to the contours of any sum-decomposition. Again, what we show in Figure 4.6 and Figure 4.7 does not prove anything, as we have glossed over many details – such as the ability of neural networks to represent non-differentiable functions. But perhaps this provides some intuition for why this function should be so difficult to model with sum-decomposition.

[17]Neural networks can, of course, represent non-differentiable functions, but if we need to construct a ϕ which is non-differentiable in infinitely many places, things get more complicated.

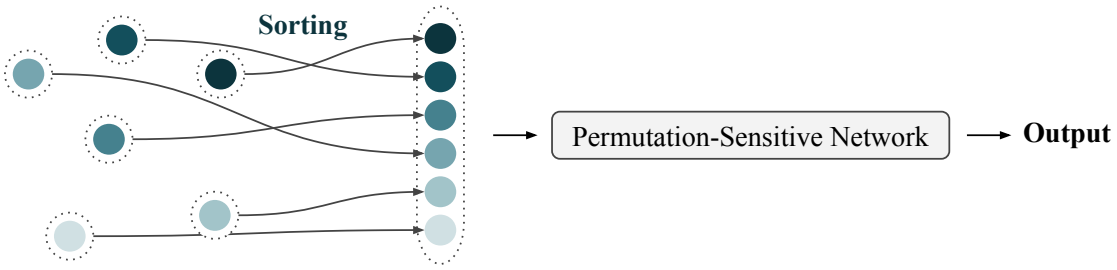


Figure 4.8: Achieving permutation invariance via sorting. Finding or learning a canonical way of sorting the elements of a set before feeding it into a permutation-sensitive network yields a permutation-invariant output – if the sorting does not depend on the initial order of the elements.

4.5 Other Methods for Achieving Permutation Invariance

In this section we discuss methods for achieving permutation invariance beyond k-ary Janossy pooling. In particular, we cover two paradigms mentioned in passing in Section 4.2, which are also discussed in Murphy et al. [43]: methods which achieve exact permutation invariance through sorting, and methods which do not achieve exact permutation invariance, but only approximate this property. We also provide a brief discussion of the relationship between learning permutation-invariant functions for sets and learning on graphs.

4.5.1 Sorting

When a set is represented as an ordered structure, it is generally assumed that the order can be arbitrary. That is, two representations $\mathbf{x}$ and $\mathbf{y}$ of the same set, where the elements are ordered differently, are both regarded as valid representations of the set. However, one could also say that each set has only one valid representation, corresponding to some canonical ordering of the elements. If it is ensured that only this valid representation is seen by the model by first sorting the elements, then permutation invariance is guaranteed. This is, in most cases, computationally cheaper than permuting and pooling. The only additional computation necessary is the sorting of the elements into their canonical representation, which is $O(n \log n)$. This method, visualised in Figure 4.8, does have some drawbacks.

First, there is an inherent ambiguity of how to pick the canonical ordering. This choice can be made manually, but the choice of ordering may in fact affect the performance of the model. This is because the ordering may or may not be meaningful in the context of the task. To illustrate this, consider two ways of sorting the integers. First, sorting according to the usual order relation on the integers ($1 < 2 < 3$), and second, sorting alphabetically by the representation of each integer as an English word ("one" < "three" < "two"). If we want to compute the `max` of a set of integers, the first choice of sort operation renders the task trivial, while the second choice of sort operation does not.

To avoid poorly-specified orderings, the ordering can instead be learned – or more straightforwardly, a function can be learned giving a score to each element, which is then used to sort the elements by their scores. This raises the issue of how to learn such a scoring function. More specifically, how are the gradients for this scoring function obtained? The ranking of elements according to their sorting score is a piece-wise constant function[18], meaning that the gradients are zero almost everywhere and undefined at the remaining locations. This would not be a problem if there were labels for the perfect ranking during training time – in that case the model could just predict the ranking and gradients could be obtained by comparing the model's predictions to the ground truth. But, in general, there are no labels for the perfect ranking. A good ranking is whatever allows the decoder (the permutation-sensitive network) to perform well. This makes getting proper gradients significantly more difficult.

Backpropagating through piecewise constant functions, however, is an established task in deep learning. The straight-through estimator, for example, is a viable tool to apply here [56, 57]. Recently, a cheap differentiable sorting operation with a computational cost of $O(n \log n)$ was also proposed by Blondel et al. [58]. To the best of our knowledge, this has not been applied to standard set-based deep learning tasks, but this area could certainly provide interesting applications of this method.

[18]That is, the function taking a list of elements to a list of ranks. For alphabetical sorting, for example, we have $(\text{bat}, \text{cat}, \text{ant}) \mapsto (2, 3, 1)$.

Liu, Wang, and Ji [59] propose yet another approach: after ranking the inputs according to a learned score function, a 1D convolution is applied to the ranked values. However, in order to get gradients for the score function, Liu, Wang, and Ji [59] multiply each value with its score before feeding it into the convolutional layer. Interestingly, while the gradients do backpropagate into the score function, they do not come from the sorting. The gradients come from the scores being used as features later on.

4.5.2 Approximate Permutation Invariance

One motivation for k-ary Janossy pooling is its low computational cost relative to full M-ary Janossy pooling. Recall that the high computational cost of M-ary Janossy pooling is due to the fact that S_M consists of $M!$ permutations, each of which must be computed. Murphy et al. [43] propose an alternative method of reducing this computational cost: instead of pooling over all permutations in S_M, a fixed number $p < M!$ of permutations are randomly sampled. The pooling is then performed only over those random samples, yielding approximate permutation invariance. Murphy et al. [43] show that this method can provide good empirical results even when setting $p = 1$. It is also possible to set, e.g., $p = 1$ during training and $p > 1$ at test time when fidelity of our predictions is more important, akin to ensemble methods. This could even be a potentially useful way to obtain uncertainty estimates for the model's predictions.

Pabbaraju and Jain [60] approach approximate permutation invariance in a different way. Inspired by Generative Adversarial Networks [61], they propose to use an adversary that tries to find the permutation of the input data which maximises the loss of the main task. This encourages the model to become close to permutation-invariant to minimise the impact of the adversary.

4.5.3 From Sets to Graphs

Even though this paper is focussed on sets, not graphs, there is an obvious link between these two topics, which we briefly discuss. Graphs typically consist of

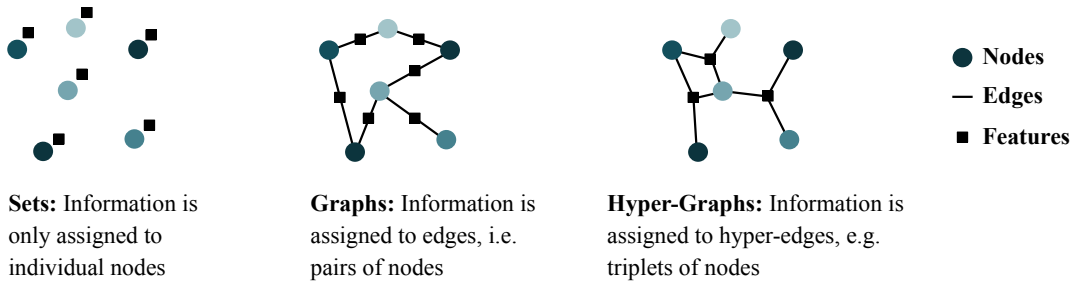


Figure 4.9: From sets to graphs and hyper-graphs. Sets can be extended to graphs by introducing the notion of edges. A permutation of the input now corresponds to a permutation of the nodes together with the respective edges.

nodes (which form sets) and edges, where each edge connects two nodes. This can be extended to hyper-graphs which include information attached to triplets of nodes [62, 63]. From this perspective, it becomes clear that a set can be seen as a graph without edges (Figure 4.9).

Whereas a set of M nodes (without edge information) can be represented as a vector of length M, a fully-connected bi-directional graph has $M(M-1)$ edges, which can be captured in an $M \times M$ adjacency matrix. Analogously, hyper-graph data can be represented with higher order adjacency tensors. The concept of permutations can now easily be extended from sets to graphs: in addition to nodes, the adjacency matrices capturing the edge information are transformed by the same permutation. This raises the question whether permutation invariant architectures from the sets literature can be applied to graphs as well. Interestingly, there is no canonical way to apply Deep Sets to graphs as it is not clear how to process the edge information. With self-attention algorithms [40], on the other hand, it is straightforward to include edge information [42, 64].

Notably, Maron et al. [65] study the set of independent linear, permutation-invariant or equivariant functions (i.e. matrices) transforming vectorised versions of the adjacency tensors. In fact, there is an orthogonal basis of such linear, permutation-equivariant functions. The number of basis vectors is connected to the *Bell number* and is, remarkably, independent of the number of nodes in the (hyper-)graph. As an example, the edge information of a bi-directional graph can be written as an $M \times M$ adjacency matrix or, after vectorising this matrix, as a vector

of length M^2. This vector can now be linearly transformed by multiplying it with a matrix of size $M^2 \times M^2$. Maron et al. [65] show that there is an orthogonal basis of at most 15 different $M^2 \times M^2$ matrices[19] which transform the (vectorised) adjacency matrix in an equivariant manner. The number 15 is the fourth Bell number, and remarkably this upper bound is independent of the size M of the graph.

This result shows a contrast between linear and nonlinear permutation-invariant functions. The complexity of the space of linear functions does not change as the number of inputs increases. By contrast, at least from the perspective of Deep Sets, the "complexity" of the nonlinear function space as measured by the necessary latent dimension increases with the number of inputs, as is shown in Section 4.3. Our main result in Section 4.4 shows that this is the case even if we consider not the whole function space, but also any dense subspace.

4.6 Universality Beyond Deep Sets

We have seen that Deep Sets is theoretically capable of representing all permutation-invariant functions if the latent space is at least as large as the cardinality of the sets being processed. For large input set sizes, this can be prohibitively slow and memory-consuming. Moreover, even for small set sizes, the theoretical universality does not guarantee that this is the best choice in practice. For example, from relational reasoning experiments [47], it is known that Deep Sets is not always best at learning about interactions between input elements. Especially when dependencies between elements are important for the task, other architectures, such as self-attention, are often preferred. This raises the question of how this empirical evidence of superior performance on some tasks relates to general theoretical limitations of these models. In the following, we provide a summary of the sufficiency and necessity criteria for universal function representation for different categories of set-based learning approaches.

[19] As noted in Finzi, Welling, and Wilson [66], for small M this basis may actually have fewer than 15 elements.

4.6.1 Janossy Pooling

We consider k-ary Janossy pooling as depicted in Figure 4.1 and described in Section 4.2. We assume a universal function approximator processing the permuted input subsets, a sum over the processed inputs, and a final component which is again a universal function approximator acting on the sum.

Definition 4.24. Let $M, N, k \in \mathbb{N}$, $\mathbf{x} \in \mathbb{R}^M$. Let $f : \mathbb{R}^M \to \mathbb{R}$. Write $T_k(\mathbf{x})$ for the set of all k-tuples of coordinates of $\mathbf{x}$. We say that f has a *continuous k-ary Janossy representation via $\mathbb{R}^N$* if there exist continuous functions $\phi : \mathbb{R}^k \to \mathbb{R}^N$ and $\rho : \mathbb{R}^N \to \mathbb{R}$ such that:

$$f(\mathbf{x}) = \rho\Big(\sum_{\mathbf{t} \in T_k(\mathbf{x})} \phi(\mathbf{t})\Big).$$

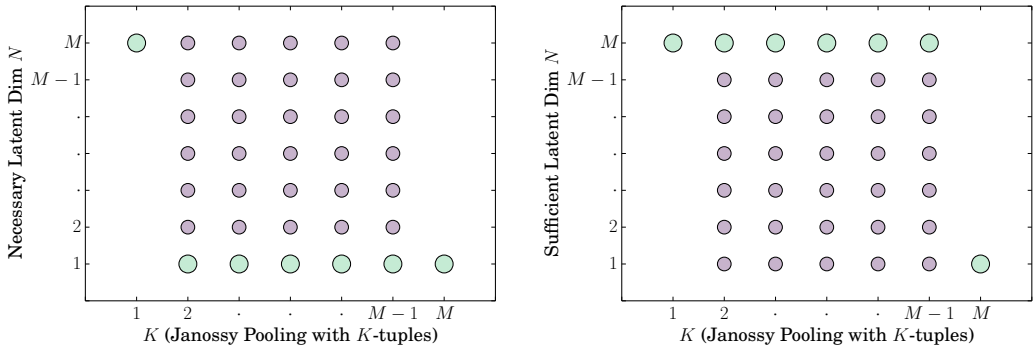
Case $k = 1$

This case is Deep Sets. In Sections 4.3 and 4.4, we provided an in-depth analysis of this case. In a nutshell, Janossy pooling for $k = 1$ can approximate all permutation-invariant functions if the latent space N is at least as large as the number of inputs M. Moreover, we show that a smaller latent space ($N < M$) does not suffice in general.

Case $1 < k < M$

This inherits sufficiency from the case $k = 1$: the network processing inputs of cardinality k can ignore all inputs except the first. This recovers Deep Sets, though each element is seen more than once, so the model must divide its output by the number of inputs having the same first element. In Janossy pooling, this number is $(M-1) \cdot (M-2) \cdot \ldots \cdot (M-2)(M-k) = (M-1)!/(M-k-1)!$. This gives the following statement:

Theorem 4.25. *Let $f : \mathbb{R}^M \to \mathbb{R}$ be continuous and permutation-invariant. Then f has a continuous k-ary Janossy representation via $\mathbb{R}^M$ for any choice of k.*



(a) Minimum Necessity **(b)** Maximum Sufficiency

Figure 4.10: Necessary and sufficient conditions for the size of the latent dimension for Janossy pooling. The large, green circles show the minimum and maximum boundaries as according to our proofs. The smaller, purple circles show possible true necessary and sufficient conditions for universal function representation.

Case $k = M$

In this case, the first part of the neural network sees the entire input, and can therefore directly approximate the desired function. The model must then divide by a factor which takes into account the sum over the permuted terms, namely $M!$. This means that there is no requirement anymore on the size of the latent space, and we can make the following statement:

Theorem 4.26. *Let $f : \mathbb{R}^M \to \mathbb{R}$ be continuous and permutation-invariant. Then f has a continuous M-ary Janossy representation via $\mathbb{R}$.*

Open Questions

For the extreme cases of k-ary Janossy pooling, $k = 1$ and $k = M$, we know a necessary and sufficient condition on the latent space dimension required for universal approximation. For $1 < k < M$, however, there is a gap between the largest known necessary condition and the lowest known sufficient condition. Figure 4.10 visualises the conditions we derived and what the true, strongest statements might be. As an example: for $k = 2$, we know that a latent dimension of $N = M$ is sufficient and a latent dimension of $N = 1$ is necessary in order to be able to represent all permutation-invariant functions. However, it is possible that $N = 2$ is sufficient,

or that $N = M - 1$ is necessary. This is an interesting open question and we leave it for future work to further investigate this.

The proofs of our results on Deep Sets rely on the input elements being 1-dimensional. It is known that Deep Sets is universal for approximation of set functions with higher-dimensional inputs, and indeed sufficient conditions on the latent space dimension are known for both invariant [48] and equivariant [50, 67] set functions. It is unknown, however, whether these known sufficiency bounds are also necessary – indeed, no non-trivial necessary bound is known, other than the 1-dimensional case presented in Section 4.4.

4.6.2 Sorting

In Section 4.5.1, we described a category of methods which achieve permutation invariance through sorting the inputs. We assume that a (potentially learned) scoring function is used to sort the inputs. A list of sorted inputs is then fed to a universal function approximator (e.g. a sufficiently large neural network). As the order of the sorted input is invariant with respect to permutations of the input, the whole architecture is automatically permutation-invariant. Since the second part of the model is a universal approximator, this architecture choice is therefore a universal function approximator for permutation-invariant functions and we arrive at the following conclusion:

Theorem 4.27. *Let $f : \mathbb{R}^M \to \mathbb{R}$ be continuous and permutation-invariant. Then for any sorted version of the inputs, f has a continuous representation which acts on these sorted inputs.*

Caution must be exercised in the case where the sorting does not operate directly on the inputs, but instead on features derived from the inputs. Universality still holds if the features are a smooth bijective mapping from the inputs.

4.6.3 Approximate Permutation Invariance

In Section 4.5.2, we mentioned two approaches to approximate permutation invariance. These models are in some sense *too* expressive – they may, depending on the exact implementation details, be capable of approximating the permutation-invariant target function, but they may also be capable of approximating functions which are not permutation-invariant.

In the approach of Pabbaraju and Jain [60], permutation invariance is encouraged by the training process, but the model class is not constrained to be permutation-invariant – if a permutation-sensitive function approximator is used to build the model, then the trained model may retain the ability to distinguish between different permutations of the same set. Whether the model is capable of approximating the target function depends entirely on the model class chosen – Pabbaraju and Jain [60] choose an LSTM, but other models could be trained with a similar adversarial method.

In the case of approximate Janossy pooling it is similarly clear that, if a universal approximator is used for the function whose outputs are pooled (i.e. for the function f in the notation of Equation 4.1), then the overall model inherits this universality. Although permutation invariance is not guaranteed, the model is guaranteed to approximate permutation invariance in the following sense. Write p for the number of permutations sampled, and write f for the permutation-sensitive function whose outputs are pooled. For a given set $\mathbf{x}$, there are $M!$ possible permutations π, and therefore $M!$ possible outputs $f\Big(\pi(\mathbf{x})\Big)$. When $p = 1$, we get an output sampled uniformly from these possible outputs, and we can view this output as a random variable with variance σ^2. For $p > 1$, the output is the mean of a random sample of size p from the set of outputs of f. The variance of this random variable is

$$\frac{M! - p}{M! - 1} \cdot \frac{\sigma^2}{p}.$$

This goes monotonically to 0 as $p \to M!$.

4.7 Conclusion

This work provides a theoretical characterisation of the representation and approximation of permutation-invariant functions. To this end, we focus on the Deep Sets architecture from Zaheer et al. [12] and derive sufficient and necessary conditions for continuous function representation. This is achieved by noting the importance of considering continuous transformations. We find that Deep Sets is only able to represent all continuous functions if the latent space is at least as large as the number of inputs. We relate these insights back to the broader paradigm of Janossy pooling, as introduced by Murphy et al. [43].

Continuing our focus on continuous transformations, we adopt a topological perspective to answer the question of whether requiring universal function *representation* is too strong of a criterion and whether universal function *approximation* would be possible even with smaller latent spaces. We turn again to the Deep Sets architecture and answer this question in the negative. We show that functions exist which can only be approximated very poorly (i.e. the worst case error is no better than a constant baseline) when making the latent space any smaller than the number of inputs.

We hope that the analytical arguments in this work inspire future theoretical treatments of the subject while also providing practical guidance for machine learning practitioners. Regarding future work, it would be interesting to extend the theoretical analysis beyond scalar-valued set elements, to vector-valued set elements, as commonly encountered in practical applications.

Acknowledgements

We thank André Henriques, Jan Steinebrunner and Tom Zeman for a helpful mathematical discussion, including proofs that the function Γ_N defined in Equation (4.14) has a zero when $N = 2$. This research was funded by the EPSRC AIMS Centre for Doctoral Training at the University of Oxford.

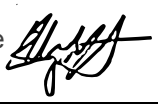
Statement of Authorship for joint/multi-authored papers for PGR thesis

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	Universal Approximation of Functions on Sets
Publication Status	□Published □ Accepted for Publication ☑Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	

Student Confirmation

Student Name:	Edward Wagstaff		
Contribution to the Paper	Responsible for essentially all of the mathematics and the wrietup thereof. The remainder of the writing was shared equally between the first three authors.		
Signature 		Date	09/06/2021

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:	Michael Osborne, Professor of Machine Learning		
Supervisor comments	All above seems accurate and fair.		
Signature 		Date	10 June 2021

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

5

Iterative SE(3)-Transformers

Fabian B. Fuchs,† Edward Wagstaff,† Justas Dauparas

† These authors contributed equally

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[1].

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

5.1 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 [68]. 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 [69] successfully applied machine learning 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 [70]. 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 [71–75].

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 [69]. 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 [36]. 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[1].

5.2 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.

5.2.1 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 [76, 77]. 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 [78–80]. Coevolutionary information is encoded in a multiple sequence alignment (MSA) [81], which can be used to learn pairwise features such as residue distances and orientations [80]. 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 [82].

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 symmetry by design, as done in AlphaFold 2 [69]. 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 [78].

5.2.2 Equivariance and the SE(3)-Transformer

A function, task, feature, 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

$$f(\mathbf{R} \cdot \vec{x}) = \mathbf{R} \cdot f(\vec{x})\,. \tag{5.1}$$

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 *filaments*.

We can view a filament as a concatenation of several vectors, where $SO(3)$ acts on each component vector. A filament could, for example, concatenate momentum and mass to form a combined 4-dimensional vector. The momentum vector is 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. The concept of *types* is used in the representation theory of SO(3), where a type-ℓ feature is rotated by a $(2\ell + 1) \times (2\ell + 1)$ Wigner-D matrix. [2] Because a filament is a concatenation of features of different types, the entire filament is rotated by a block diagonal matrix, where each block is a Wigner-D matrix [83–85].

At the input and output layers, the filament structure is determined by the task at hand. For the intermediate layers, arbitrary filament structures can be chosen. In the following, we denote the structure of a filament as a dictionary $\{t_i : n_i\}$, where each entry denotes the number n_i of features of type t_i present in the filament. E.g. a filament consisting of 3 scalar values (e.g. RGB colour channels) and one velocity vector has 3 type-0 features and 1 type-1 feature, which we denote `{0:3, 1:1}`. This filament 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 [86–90]. The SE(3)-Transformer [36] in particular is a graph neural network explicitly designed for SE(3)-equivariant tasks, making use of the filament 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[3] at the core of the SE(3)-Transformer is based on closely related earlier works [83–85]. In contrast, Cohen et al. [91] introduced rotation equivariance by storing copies corresponding to each element of the group in the hidden layers – an approach called

[2] We can think of type-0 features as rotating by the 1×1 rotation matrix (1).
[3] This approach rests on the theory of *irreducible representations [83, 84]*.

regular representations. This was constrained to 90-degree rotations of images. Recent regular representation approaches [92–94] extend this to continuous data by sampling the (infinite) group elements and map the group elements to the corresponding Lie algebra to achieve a smoother representation.

5.2.3 Equivariant Attention

The second core aspect of an SE(3)-Transformer layer is the self-attention [95] mechanism. This is widely used in machine learning [87, 96–101] 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 [87, 95, 102–104]. 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} and values v_{ij}[4]. 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.

5.3 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.

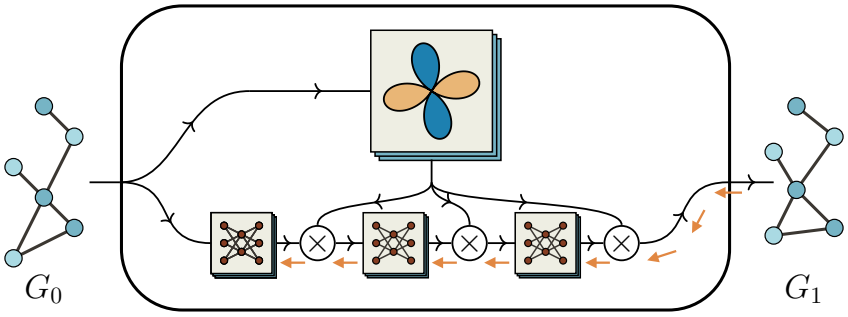


Figure 5.1: Gradient flow (orange) in a conventional single-pass SE(3)-Transformer mapping from a graph (left) to an updated graph (right). The equivariant basis kernels (top) need not be differentiated as gradients do not flow through them.

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

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 computed by trainable neural networks which take scalar edge features as input. 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 [36].

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. 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 need not be differentiated, as can be seen in Figure 5.1.

When applying an iterative SE(3)-Transformer (Figure 5.2), each block i outputs a graph G_i. This allows for re-evaluating the interactions between pairs of nodes. The relative positions, and thus the basis matrices, change before the final output, and gradients thus flow through the basis. The spherical harmonics used to construct

[4]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.

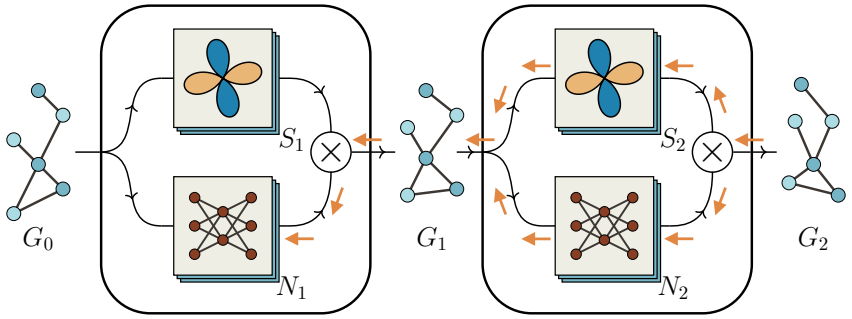


Figure 5.2: Gradient flow (orange) in an iterative SE(3)-Transformer with multiple position updates. Now the gradients do flow through the basis construction (S_2) meaning this part has to be differentiated.

the basis are smooth functions, and therefore backpropagating through them is possible. We provide code implementing this backpropagation.

5.3.2 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 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 filament 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.

5.3.3 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 filaments between the blocks as described in 5.3.2 to avoid information bottlenecks. A downside is that it fixes the number of iterations of the SE(3)-Transformer.

5.3.4 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 optimisation 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.

5.4 Experiments

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

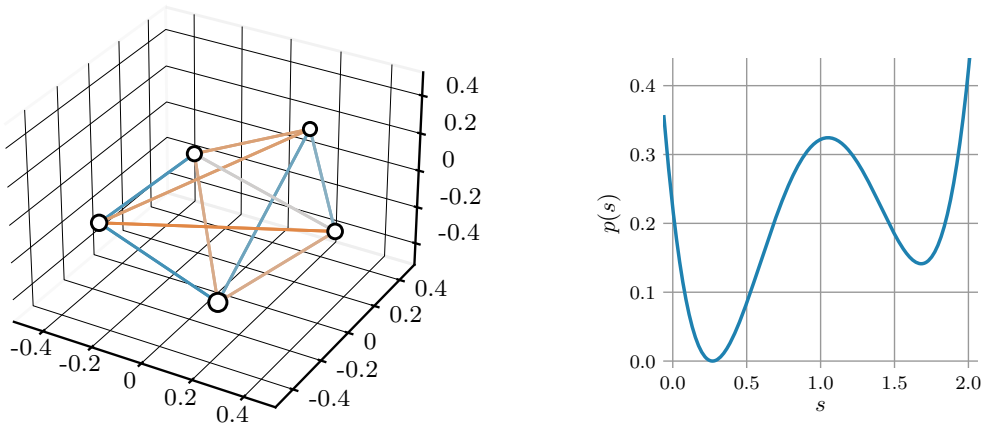


Figure 5.3: The left plot shows a configuration of nodes, with a potential between each pair. The gradient of each potential is represented by the colour of the edges. Blue edges indicate repulsion and orange edges indicate attraction. Stronger colour represents stronger interaction. Our experiment uses ten nodes, but we show only five here for visual clarity. The right plot shows the double-minimum, pairwise potential $p(s)$, with parameter $a = 0$.

global potential landscape that is not trivially solvable for gradient descent:

$$s_{ij} = r_{ij} - a_{ij} - 1\,, \tag{5.2}$$

$$p_{ij}(s_{ij}) = s_{ij}^4 - s_{ij}^2 + \frac{s_{ij}}{10} + p_{\min}\,. \tag{5.3}$$

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 a single optimal solution for all examples, which the network could simply memorise.

We consider three models for solving this problem: (i) the **single-pass** SE(3)-Transformer (12 layers); (ii) a three-block **iterative** SE(3)-Transformer (4×3 layers); (iii) **gradient descent** (GD) on the positions of the nodes. We also evaluate a combination of first applying an SE(3)-Transformer and then running GD on the output. We evaluate the iterative SE(3)-Transformer both with and without propagation of basis function gradients as described in Section 5.3.1 (the latter case corresponds to removing the gradient flow from the upper half of Figure 5.2). We run GD until the update changes the potential by less than a fixed tolerance value. We ran each model between 15 and 25 times – the σ values in Tables 5.1

Table 5.1: Average energy of the system after optimisation (lower is better).

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

Table 5.2: Performance comparison of single-pass and iterative SE(3)-Transformers for different neighbourhood sizes K and a fully connected version (FC).

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

and 5.2 represent 1-σ confidence intervals (computed using Student's t-distribution) for the mean performance achieved by each model. All models except GD have the same overall number of parameters.

The results in Table 5.1 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.

So far, every node attended to all $N-1$ other nodes in each layer, hence creating a fully connected graph. In Table 5.2, we analyse how limited neighborhood sizes [36] 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 5.2 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.

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).**

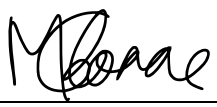
Title of Paper	Iterative SE(3)-Transformers
Publication Status	□Published ☑ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	To appear in the proceedings of the 5th International Conference on Geometric Science of Information, GSI 2021

Student Confirmation

Student Name:	Edward Wagstaff		
Contribution to the Paper	Coding, manuscript preparation, experimental design and algorithm design were shared equally between the first two authors, with the exception of the parts of the introduction and background dealing with protein structure prediction and refinement, which were largely the responsibility of the third author.		
Signature 		Date	09/06/2021

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:	Michael Osborne, Professor of Machine Learning		
Supervisor comments	All above seems accurate and fair.		
Signature 		Date	10 June 2021

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

Preface to Chapter 6

In the final section of this thesis, we turn to a direct practical application of machine learning. In particular, we investigate the relationship between the aurora borealis, as imaged from ground stations in Canada, and ionospheric disturbances to satellite transmissions (or *scintillations*). We attempt to build a predictive model of these scintillations based on auroral images and a number of physical measurements. We also investigate the relationship between human-labeled classes of auroral imagery and clusters discovered by unsupervised learning techniques. Since different human-labeled classes correspond to different physical behaviour, which may drive different scintillation behaviour, this opens the door to the future possibility of correlating different image clusters with different levels of scintillation activity.

In the course of developing the models used in this section, we incorporated prior knowledge of the tasks at hand in two main ways. First, based on the knowledge that the time series to be predicted in Chapter 6 is sparse and sharply peaked, we trained the model using a novel loss function which encourages outputs of this form. Second, in order to better exploit the translation invariance of CNNs, we transform the image data from its raw form, a distorted wide-angle view of the sky, to a projection onto a latitude-longitude grid, which more closely represents the actual spatial structure of the aurora.

Chapter 6 is a combined version of two papers presented at the Workshop on Machine Learning and the Physical Sciences at NeurIPS 2019, and as such there are two separate authorship statements following this chapter.

6

Auroral Dynamics and GNSS Phase Scintillations: A Machine Learning Approach to Prediction and Clustering

Kara Lamb†, Garima Malhotra†, Athanasios Vlontzos†, Edward Wagstaff†, Atilim Günes Baydin, Anahita Bhiwandiwalla, Yarin Gal, Alfredo Kalaitzis, Anthony Reina, Asti Bhatt

† These authors contributed equally

Abstract

A Global Navigation Satellite System (GNSS) uses a constellation of satellites around the earth for accurate navigation, timing, and positioning. Natural phenomena like space weather introduce irregularities in the Earth's ionosphere, disrupting the propagation of the radio signals that GNSS relies upon. Such disruptions affect both the amplitude and the phase of the propagated waves. No physics-based model currently exists to predict the time and location of these disruptions with sufficient accuracy and at relevant scales. In this paper, we focus on predicting the phase fluctuations of GNSS radio waves, known as *phase scintillations*. We propose a novel architecture and loss function to predict 1 hour in advance the magnitude of phase scintillations within a time window of ±5 minutes with state-of-the-art

performance. The physical processes underlying scintillations also drive the aurora borealis. We investigate the incorporation of images of the aurora into our predictive model, and also investigate the relationship between human-labelled classification of auroral images and unsupervised clustering of the same images.

6.1 Introduction

A global navigation satellite system (GNSS) is a constellation of satellites around the Earth used for accurate navigation, timing and positioning. GNSS refers to a large category of commercial products like the Global Positioning System (GPS). As a society, we are becoming increasingly dependent on GNSS technology, with a recent study estimating losses of $1 billion per day in case of an extended outage [105]. Therefore it becomes imperative to predict disruptions to GNSS signals with good spatial and temporal resolution.

GNSS signals are high frequency radio waves that propagate through the ionosphere before they reach ground-based receivers. The frequency of these signals is of the order of GHz and they therefore interact with small scale ionospheric irregularities (i.e. sharp gradients in ion and electron densities). This causes the signals to exhibit rapid amplitude and phase variations known as *scintillations*, causing uncertainty in position and loss of lock in severe cases [106]. Interactions between the sun and the Earth's ionosphere are extremely non-linear, which makes the prediction of space weather effects challenging. Because of the complex nature of the problem, a complete theory of ionospheric irregularities and signal scintillation does not yet exist, which limits the prediction capabilities of physics-based models [107].

In this study, we focus on predicting phase scintillations at high latitudes, using data from Global Positioning System (GPS) receivers in the Canadian High Arctic Ionospheric Network (CHAIN) obtained between 2015–2017. Phase scintillations are an uncommon yet severe phenomenon. Over the course of 2015–2016, only 0.0091% of the minute basis samples from CHAIN exhibited scintillations over the threshold of 0.1, which is the value above which the GNSS signal reliability

decreases [108]. Therefore, the event task is sparse, rendering standard techniques incapable of predicting accurately in time and magnitude. In addition, due to errors in hardware, 22% of data in the years 2015–2017 are missing, further complicating the prediction task.

We propose a method for predicting scintillations from time-series of physical measurements, incorporating two key novelties: (a) we account for the sparsity of scintillations with a custom loss function; (b) we handle missing data values with binary masks that inform our model which values are missing. We outperform the current state of the art by making predictions 1 hour in advance with a total skill score (TSS, discussed below) of 0.64. To the best of our knowledge we are the first to treat the prediction problem as a regression problem, rather than classifying the existence of phase scintillations, as per the current state of the art.

At high latitudes, the dominant source of ionospheric irregularities and therefore scintillations is solar-driven storms and substorms (geomagnetically active periods). One very visible manifestation of these high energy inputs from the sun is the aurora, which has also been shown to correlate with these scintillations [109, 110]. Variations in the visible aurora are manifestations of variations in the geophysical drivers. Specific structures in the aurora may be more correlated with high σ_ϕ values (e.g. Jin, Moen, and Miloch [111], Oksavik et al. [112], Meeren et al. [113], and Jin et al. [114]), and we therefore investigate incorporating images of the aurora into our prediction pipeline. These images originate from the THEMIS network of northern Canada [115].

Auroral images have long been classified in a few distinct classes pointing to their geophysical driving mechanisms. Several recent studies have applied deep learning methods to aurora image classification [116–118]. Such classification is relevant to the study of phase scintillations because the underlying geophysical driving mechanisms, which vary between classes, also have varying effects on scintillations. We investigate unsupervised approaches to clustering auroral images using a general-purpose approach, and the relationship of these clusters to human-specified image classes.

6.2 Predicting Phase Scintillations

There is no physics based model capable of performing accurate predictions of timing and magnitude of phase scintillations with relevant spatial and temporal scales. Previous data-driven approaches include Lima et al. [119], where the authors predict scintillations in equatorial latitudes; we note that their results are not directly comparable to our task, as the physics guiding ionospheric scintillations on high latitudes differ from the equatorial. In Jiao et al. [120], the authors characterise the climatology of scintillations using statistical analysis but do not predict them in the future. The only known predictive model, from McGranaghan et al. [108], only classifies the occurrence of scintillations 1 hour in advance. Our work in this section is the first to treat phase scintillation prediction as a regression problem, and the first to account for event sparsity and missing data.

6.2.1 Methodology

Dataset: We study the high latitude Canadian sector ($50° - 70°$ geographic latitude) between mid-October 2015 and end-February 2016. We use solar activity parameters such as solar wind speed (Vsw), interplanetary magnetic field components such as (IMF Bz, By), F10.7, Sym-H, etc. and geomagnetic activity indices such as Kp, AE to characterise the global influence of solar activity on Earth's magnetosphere and ionosphere [15]. For local ionospheric state information pertaining to the high latitude Canadian sector, we are using the CARISMA (Canadian Array for Realtime Investigations of Magnetic Activity) magnetometer dataset, the Canadian High Arctic Ionospheric Network for ionospheric total electron content (TEC) and scintillation index measurements. In summary, we have 39 features in a minute-cadence dataset. We use the first 75000 points (until mid-January) as our training data, while we reserve the latter 75000 points as our test set. We also considered the use of images of the aurora borealis originating from the THEMIS network of northern Canada [115], projected onto a latitude-longitude grid and then mapped to a lower-dimensional feature representation by a convolutional neural network. In our experiments, the use of these images did not significantly affect

the performance of the model, and the results we report here are from a model which does not make use of THEMIS images.

Masking: Due to measurement errors, faulty equipment and other natural phenomena, 22% of the data are corrupted and logged as `NaN`. For each feature, we adopt the following simple scheme for imputing such missing values. We first compute, for each feature, the mean value of all non-`NaN` observations. We then replace all missing values with the observed mean. Where a missing value has been imputed, we indicate such substitutions with a binary (1/0) mask, which is provided as an additional feature. Including a mask for each feature therefore doubles the total number of features.

More sophisticated imputation strategies are possible – for example, imputation methods based on Expectation Maximisation [121] and Markov Chain Monte Carlo [122] are popular. Our strategy of mean-imputation with masking resembles the more complex approach of Che et al. [123]. The key difference is that our imputed values are all set to the mean value, whereas in Che et al. [123], imputed values decay exponentially towards the mean from the most recent observed value. In our setting, almost all observations are very close to the mean, and it is this characteristic of the data which motivates our choice of the simple mean-imputation strategy, as opposed to any of the available more sophisticated techniques.

Sparsity and Loss Function: As discussed above, phase scintillations are a rare phenomenon. Given the extreme sparsity of our positive phase scintillation values, standard regression loss functions would find their minimum by predicting values around the mean, and failing to predict high phase scintillation events. We introduce a custom loss function defined in Equation (6.1). The loss function has two components: (i) the Mean Absolute error (MAE) between the predicted output and ground truth; (ii) a dynamic range penalty that incentivises the range of the output sequence to match the range of the true sequence. With both components, the model is encouraged to match both the mean and variance of the true sequence. In the following expression for the loss L, the sum, max and min are computed over a batch X, Y of predictions and ground truth respectively. Here, each input in

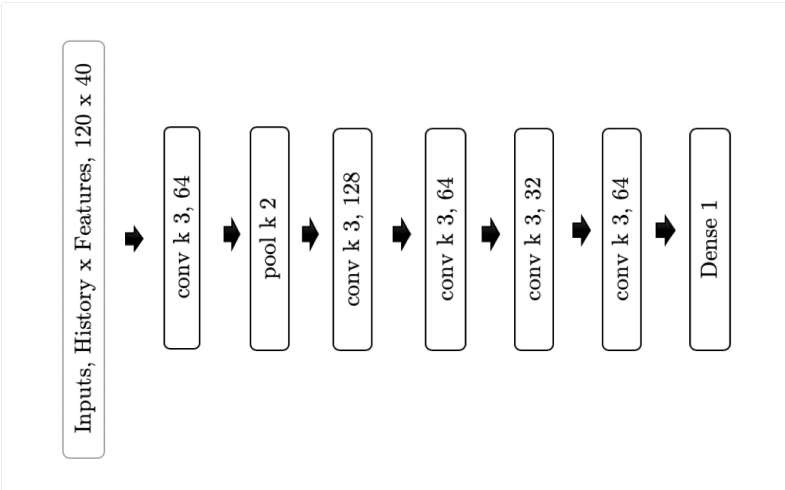


Figure 6.1: Proposed convolutional architecture. Input time series is passed through the convolutional layers and a single output neuron produces the prediction. `k`: is the size of the convolutional kernel, while the following number corresponds to the filters

the batch consists of a 120×40 array, representing a 120-element time series for each of 40 features, as shown in Figure 6.1. A batch of size k beginning at time T consists of the k such time series beginning at times $T, T+1, \ldots, T+k-1$.

$$L = \frac{\sum_i \|y_i - x_i\|}{N} + \lambda \left|\text{DynRange}(Y) - \text{DynRange}(X)\right| , \tag{6.1}$$

$$\text{DynRange}(X) = \max(X) - \min(X) . \tag{6.2}$$

In our experiments we found that setting $\lambda = 0.1$ provided the best performance.

Architecture: Figure 6.1 shows the proposed architecture. In addition to the time-series data for time t we also include observations from $t - 120$ min to t min to inform the model of the historical changes in the values. Thus, the dimensionality of our input is $History \times Features$; in our case 120×40. We condition our model to predict 1 hour in advance. We note that filter sizes decrease and then slightly increase in order to create a small bottleneck in feature space such that noisy information is excluded from the final layers. The aforementioned mask is appended to our input as an extra channel.

6.2.2 Results

Figure 6.2 shows our results over the whole testing dataset of 75000 minute samples, as well as a zoomed version over 5000 minute samples. The test time-points do not overlap with the training time-points. It is evident that the prediction follows closely the ground truth but fails to accurately predict the magnitude of high intensity phase scintillations. Figure 6.2b shows that despite the mismatch in absolute magnitude, our model successfully predicts a phase scintillation larger than the mean. We also note a small delay in the order of 2–3 minutes in our predictions. Figure 6.2a shows that the predicted sequence has the same peak behaviour as the ground truth. Figure 6.3 shows the predictions over the whole test set to make the predicted time series more clear without the scaling of Figure 6.2.

For ease of comparison against existing techniques such as McGranaghan et al. [108], which approached the prediction task as one of classification, we quantitatively assess the performance of our model by first casting it as a binary classification task and then applying two classification metrics. To cast as binary classification, we say that a scintillation occurs if a threshold of 0.1 in the scintillation index is exceeded (this choice of threshold follows that chosen in Jiao et al. [120]). The metrics we use to assess this are the Heidke skill score as discussed in Hyvärinen [124], and the total skill score (TSS) as defined in McGranaghan et al. [108] and Equation 6.3:

$$TSS = \frac{TP}{(TP + FN)} - \frac{FP}{(FP + TN)}\,. \tag{6.3}$$

The Heidke skill score takes values in $(-\inf, 1]$, and the TSS takes values in $[-1, 1]$. In both cases, a score of 0 means no predictive capability. Our model scores 0.34 on the Heidke skill score, showing some predictive ability. The model of McGranaghan et al. [108] performed classification with a TSS of 0.49, while our model achieves a TSS of 0.64, outperforming the current state of the art.

6.3 Unsupervised Clustering of Auroral Images

Although we found above that straightforwardly incorporating auroral images into our prediction pipeline did not improve performance, we believe that there may still

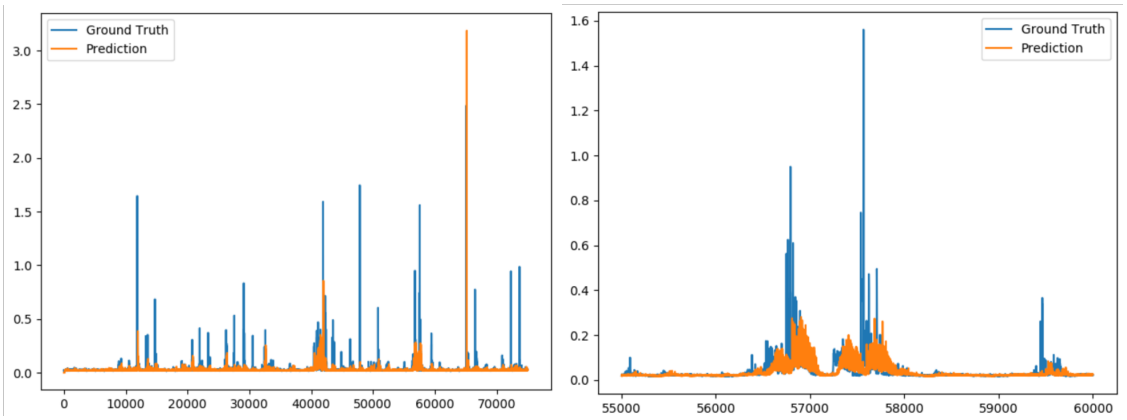


Figure 6.2: (a) Results over 75000 minute interval (b) Results zoomed in a 5000 min interval

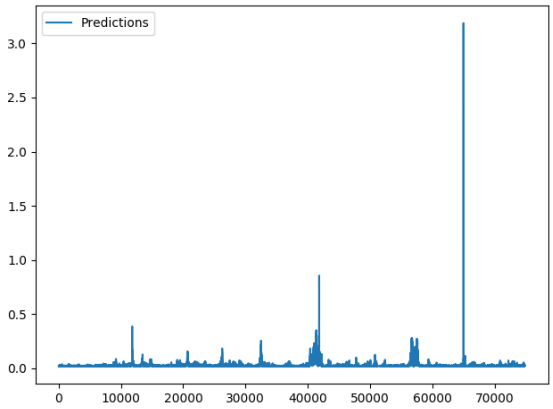


Figure 6.3: Predictions over 75000 points of test set; Note that predicted sequence exhibits same peak behaviour as ground truth.

be useful information in the images which could be extracted by more sophisticated methods. As a step in this direction, this section investigates clustering of auroral images into different classes. Previous studies on classifying aurorae with machine learning have used supervised learning approaches, which can be subjective, as they rely on the judgement of human experts to label auroral events. Human annotation can bias labels to identifying only the most prominent of features in the images, and it relies on pre-defining specific classes that may exclude some of the salient features associated with physical processes occurring in the ionosphere. In addition to being time-consuming, labelling typically focuses on identifying application-specific features. In this section, we investigate *unsupervised* clustering of auroral images,

and how the outputs of such algorithms relate to human-defined classes.

6.3.1 Method

Dataset Our dataset contains 35,277 images from THEMIS (Time History of Events and Macroscale Interactions during Substorms) All-Sky Imager (ASI) data from the Fort Simpson (FSIM) site in Northern Canada. The images are part of the THEMIS auroral dataset, taken ±3 hours from midnight and represent two months of data from January and February 2015. An additional 7700 manually annotated auroral images, classified among 6 classes (*arc, diffuse, discrete, moon, clear, clouds*) depending on the auroral structure [116] from March 2015, are used to evaluate embeddings in the low-dimensional feature space. We note that images are classified in the moon category only if the presence of the moon makes the aurora not visible (per the discretion of the expert annotating the images). We also use measurements from a PolarRxS GNSS receiver [125] at Fort Simpson from the Canadian High Arctic Ionospheric Network (CHAIN) to provide measurements of the scintillation indices at the same time as the aurora images.

Residual Autoencoder In this work we use a Residual Autoencoder (Res-AE) as introduced in Hou and Vlontzos [126]. Figure 6.4 illustrates the encoder, and the decoder mirrors the encoder. The overall architecture has a U-Net like structure, where sequential convolutions create multi-scale representations of the input [127]. Each resolution level includes a residual block [128] and a convolutional layer with a stride of 2. The resulting feature map is passed to the next resolution level. Furthermore, a series of convolutional layers map the features from all resolution levels to an encoding of size $L_1 \times L_2 \times 128$, adding them all together to create an intermediate representation L'. A final convolutional layer then maps the intermediate representation to the encoding L of size $L_1 \times L_2 \times L_3$. The size of the representation L is a critical hyperparameter as it defines the expressiveness of the encoder. In our work we found the optimal trade-off between feature space compression and expressiveness to be $32 \times 32 \times 3$. The multi-scale and residual nature of the autoencoder compresses the high-dimensional input images into a feature

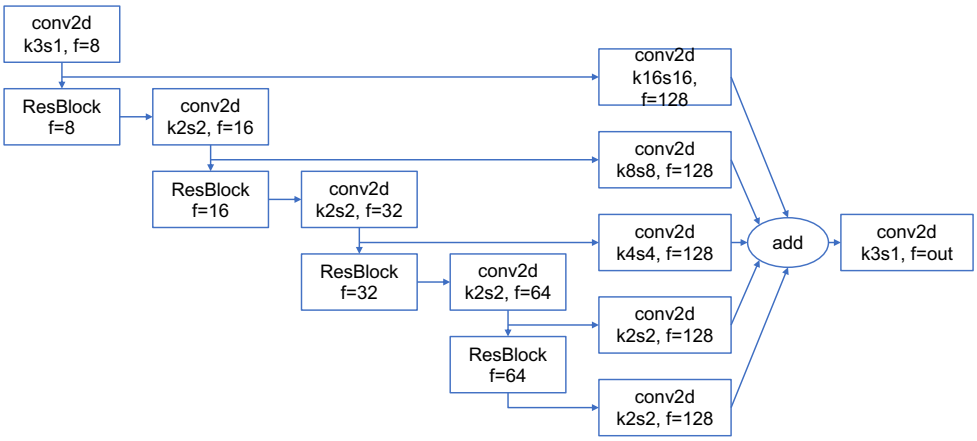


Figure 6.4: Res-AE Encoder.

representation that accounts for multi-scale features and dependencies while filtering out some of the noise. Figure 6.5 shows an example of the test and reconstructed images using the Res-AE trained on images from the FSIM Themis site.

Unsupervised clustering of embedded latent representations We investigate Barnes-Hut t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) for dimensionality reduction of the latent space [129, 130]. Both approaches provide lower dimensional representations of higher dimensional data, such that essential topological structures of the higher dimensional space are preserved in the density of embeddings in the lower dimensional space. In our approach, THEMIS images are mapped to the latent space L using the trained Res-AE and then clustered with t-SNE or UMAP to embed the encoded latent representation into two dimensions for easier visualisation and interpretation. In this case we constrain the local neighborhood from which UMAP learns the manifold structure of the data to 15 nearest neighbors, as this provides a reasonable separation between the clusters.

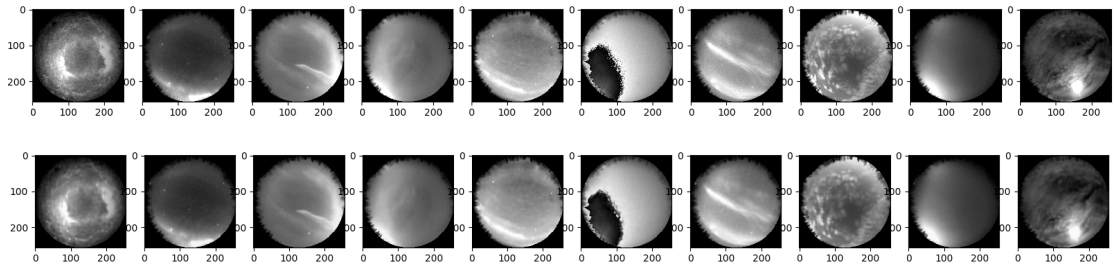
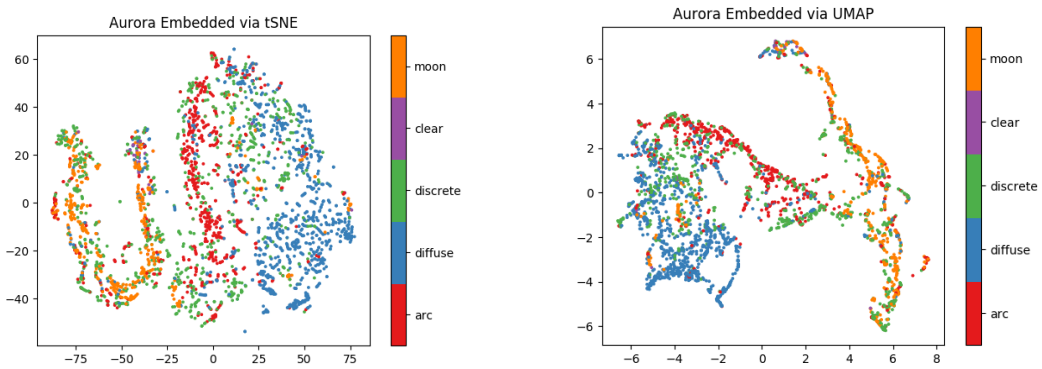


Figure 6.5: Original (top) and reconstructed (bottom) Themis images.



(a) (b)

Figure 6.6: Latent space colored by image classes projected by (a) t-SNE (b) UMAP

6.3.2 Results

We train the Res-AE on 2 months of THEMIS images from FSIM to encode our image class labelled dataset of 7700 images (we exclude images labelled as *cloudy*, as these do not contain physical information on the ionosphere). Figure 6.6 shows the embedded latent representations for both the t-SNE and UMAP projections, colored by the image class labels. Both t-SNE and UMAP generate similar clusters of the labelled image classes from the latent representations. They both most clearly cluster images identified as *moon*, *arc*, and *diffuse*, with subsets of the *discrete* images more closely associated with each of these clusters. Both t-SNE and UMAP are stochastic methods, so there is necessarily some variability between runs – different runs of these algorithms result in qualitatively similar results to those shown in Figure 6.6.

6.4 Conclusions

GNSS products like GPS are vital to modern day operations; from navigation to high-frequency trading in the stock markets – all of which can be disrupted by scintillations in the signals. In this paper we proposed a novel methodology for predicting GNSS phase scintillations 1 hour in advance. We introduced a custom loss function and a method of dealing with missing and incomplete data. We further improved the current state of the art by 0.15 in skill score and demonstrated our method's predictive capability. We believe that additional data sources driven by the same physical phenomena, like auroral images, can further improve the skill score. In particular, our investigation of unsupervised clustering techniques using a general-purpose autoencoder opens up a potentially useful alternative approach to incorporating image data into a predictive model. Our results indicate that non-linear dimensionality-reduction techniques such as UMAP and t-SNE can provide meaningful lower-dimensional projections of the latent representation of aurora images that correlate with clusters associated with human annotated image classes. Future work will focus on applying our methods to a larger image data set, and quantitative investigation of the correlation of clusters with high σ_ϕ values.

Acknowledgements

This project was conducted during the 2019 NASA Frontier Development Lab (FDL) program, a public-private partnership between NASA, the SETI Institute, and commercial partners. We wish to thank, in particular, NASA, Google, Intel, ElementAI and SRI, for supporting this project.

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).**

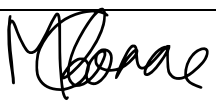
Title of Paper	Prediction of GNSS Phase Scintillations: A Machine Learning Approach
Publication Status	□Published □ Accepted for Publication □Submitted for Publication ☑Unpublished and unsubmitted work written in a manuscript style
Publication Details	Presented at the Workshop on Machine Learning and the Physical Sciences at the 33rd Conference on Neural Information Processing Systems

Student Confirmation

Student Name:	Edward Wagstaff		
Contribution to the Paper	Contributed heavily to the code, particularly the image pre-processing pipeline but also more broadly. Contributed to the design of the model architecture on an equal basis with the other first authors.		
Signature 		Date	09/06/2021

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:	Michael Osborne, Professor of Machine Learning		
Supervisor comments	All above seems accurate and fair.		
Signature 		Date	10 June 2021

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	Correlation of Auroral Dynamics and GNSS Scintillation with an Autoencoder
Publication Status	□Published □ Accepted for Publication □Submitted for Publication ☑Unpublished and unsubmitted work written in a manuscript style
Publication Details	Presented at the Workshop on Machine Learning and the Physical Sciences at the 33rd Conference on Neural Information Processing Systems

Student Confirmation

Student Name:	Edward Wagstaff		
Contribution to the Paper	Contributed substantially to the implementation code.		
Signature 		Date	09/06/2021

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:	Michael Osborne, Professor of Machine Learning		
Supervisor comments	All above seems accurate and fair.		
Signature 		Date	10 June 2021

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

7 Conclusion

7.1 Discussion and Outlook

In this thesis we have undertaken a number of investigations on the theme of incorporating prior knowledge in machine learning models. While we have necessarily focused on narrow instances of this theme, such considerations appear across machine learning.

Bayesian probabilistic models, of the type considered in Chapter 3, are popular throughout machine learning. Bayesian optimisation and probabilistic programming [131, 132] are especially prominent examples of this paradigm. The field of probabilistic programming in particular makes heavy use of numerical techniques to perform approximate inference in probabilistic models, typically implemented using Markov Chain Monte Carlo. Other numerical strategies, such as those considered in Chapter 3, can be applied in similar settings – Variational Bayesian Monte Carlo [133], for instance, directly employs Bayesian quadrature in the course of performing variational inference.

Chapters 4 and 5 both fit within the fields of graph neural networks and equivariant models. Graph neural networks have been employed in the study of, among other application areas, social networks [134], learning physical laws [135], and molecular modelling [69], the motivating application for Chapter 5. Equivariant

models, meanwhile, have also been employed in molecular modelling [69] and social networks [136], as well as in image classification [91]. Both of these paradigms have demonstrated applicability across a broad range of domains, and we hope that the work presented in this thesis contributes to the further advancement of these fields.

While most of the contributions in this thesis are theoretical or methodological in nature, in Chapter 6, we investigated an application of machine learning to a problem in the physical sciences. In recent years, applications of machine learning in the physical sciences have spanned from quantum mechanics (e.g. Unke et al. [137]) to cosmological simulations (e.g. Li et al. [138]). Applying machine learning in practice usually heavily leverages general-purpose models and approaches such as those investigated in Chapters 4 and 5. For example, Unke et al. [137] make use of SE(3)-equivariant models, and, moving beyond topics considered in this thesis, Li et al. [138] make use of generative adversarial networks [139]. As in Chapter 6, application-specific adaptations may be made to these general-purpose models in order to tailor methods to particular problems.

7.2 Avenues for Future Work

Each of the chapters in this thesis leaves open directions for future work.

In Bayesian Quadrature, the choice of surrogate used for modelling the integrand is of vital importance to the success of the method. The method discussed in Chapter 3 relies specifically on Gaussian Process surrogates. Gaussian Processes are amenable to theoretical treatment and, in many cases, allow for analytic expressions for the relevant integrals, but Gaussian Process priors may be inappropriate for many integrands. As recently investigated in Zhu et al. [5], Bayesian Quadrature methods may also make use of surrogates other than Gaussian Processes, allowing for potentially more appropriate and informative priors. One promising line of future work lies in developing similar batch selection methods for Bayesian Quadrature with surrogates other than Gaussian Processes, and another lies in further expanding the range of surrogates for which Bayesian Quadrature may be used.

Perhaps the most significant limitation of the theoretical results from Chapter 4 is that they apply to sets of one-dimensional data. This means that the lower bounds provided, while tight for one-dimensional data, are likely to be loose for higher-dimensional data. Our understanding of how the expressivity of Deep Sets behaves when the input sets consist of higher-dimensional data is therefore incomplete, and we view this as an interesting direction for future research.

Chapter 5 leaves open the possibility of further experiments with real data and varied model parameters, to investigate more deeply the behaviour of the iterated SE(3)-Transformer and the ways in which it may be tuned for better performance. Similar models have previously been used inside larger machine learning pipelines, e.g. in AlphaFold 2 [69], and it may be illuminating to isolate this component of the architecture and more fully understand its contribution to the success of the pipeline as a whole.

Chapter 6 leaves open the problem of usefully extracting quantitative information from auroral imagery and applying this to improve the accuracy of scintillation prediction. One potential avenue for this is first separately training a model to directly learn correlations between features in auroral images and high scintillation activity, and then incorporating this trained model as a component of the overall prediction model.

Appendices

A

Appendix to Chapter 4

A.1 Mathematical Remarks

A.1.1 Infinite Sums

Throughout this paper we consider expressions of the form

$$\Phi(X) = \Sigma_{x \in X} \phi(x)\,, \tag{A.1}$$

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.

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 Equation (A.1). 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 (A.1) 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.

We therefore require everywhere that $\Sigma_{x\in X}|\phi(x)|$ is convergent. For any X where this is not the case, we will set $\Phi(X) = \infty$.

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.1.2 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 $2^{\mathbb{R}}$, by which we mean that the function f_M induced by f on $\mathbb{R}^M$ by $f_N(x_1, \ldots, x_M) = f(\{x_1, \ldots, x_M\})$ is continuous for every $M \in \mathbb{N}$.

A.1.3 Remark on Theorem 4.11

The proof for Theorem 4.11 from Zaheer et al. [12] 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,

$$\Phi(X) = \sum_{m=1}^{M} \phi(x_m) = \log \prod_{m=1}^{M} \frac{1}{p(x_m)}\,, \tag{A.2}$$

which takes a unique value for each distinct X therefore extending the validity of the proof to multi-sets. This choice of ϕ diverges with infinite set size.

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

$$\Phi(X) = \sum_{i=1}^{\infty} \phi(x_m) = \sum_{i=1}^{\infty} a = \pm\infty\,. \tag{A.3}$$

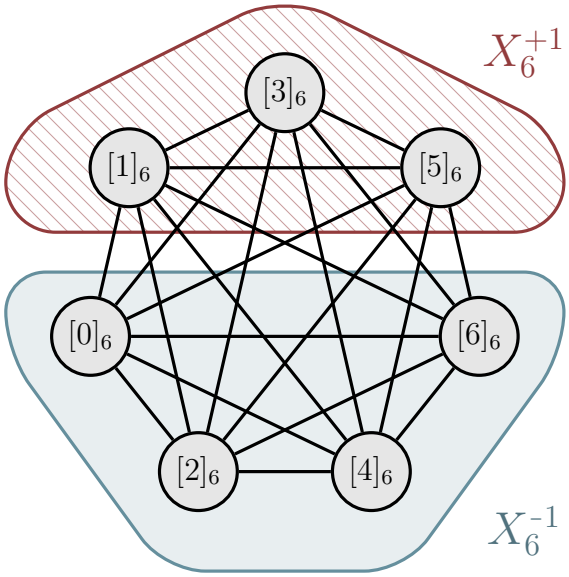


Figure A.1: The 6-dimensional simplex Δ_6, and the faces X_6^{+1} and X_6^{-1}, visualised using the complete graph K_7.

A.1.4 Visualising High-dimensional Simplices

In the proof of Theorem 4.19, we make heavy use of the N-dimensional simplex Δ_N, its $N-1$-dimensional faces $\Delta_N^{(i)}$, and the two opposite faces X_M^{+1} and X_M^{-1} of the M-dimensional simplex Δ_M. To mentally keep track of all these different objects, it is helpful to have a way of visualising them. Although these simplices are high-dimensional objects, they can be visualised easily as graphs.

An N-dimensional simplex can be defined as the convex hull of $N+1$ points in general position.[1] We can represent this as the complete graph K_{N+1}, where just as any pair of vertices of the simplex are joined by an edge, so are any pair of vertices of the graph. An $N-1$-dimensional face of the simplex is just the convex hull of N of the $N+1$ vertices, and this can be viewed as an induced K_N subgraph of our K_{N+1}. Lower-dimensional faces can be viewed similarly.

With our definition of Δ_N, the vertices of the simplex are the points whose coordinates are a (possibly empty) sequence of ones followed by a sequence of negative ones. Any vertex can be written in shorthand as $[m]_N$, indicating the point

[1] That is, no three points are collinear, no four points are coplanar, etc.

in dimension N consisting of m ones followed by all negative ones. For example, $[2]_3 = (1, 1, \text{-}1)$, the vertex of Δ_3 beginning with two ones.

Figure A.1 illustrates this way of visualising Δ_N, showing the 6-dimensional simplex Δ_6 represented as a complete graph with 7 vertices. With the vertices arranged appropriately, the faces X_6^{+1} and $X_6^{\text{-}1}$ correspond to the indicated subgraphs, and it is visually clear that it is sensible to describe them as "opposite" faces of the simplex.

A.1.5 Extending Theorem 4.19 to Higher Dimensions

As discussed in the main text, an interesting line of future work lies in extending Theorem 4.19 to sets of higher-dimensional elements. We briefly make note of a small obstacle to extending the arguments of this work to the higher-dimensional case.

Our proof of Theorem 4.19 makes use of the fact that an arbitrary continuous function f on $\mathbb{R}^M$ can be used to construct a continuous permutation-invariant function by composing with `sort`. This simplifies the process of defining a hard-to-approximate continuous permutation-invariant target function f_*, as in Equation (4.10). For d-dimensional inputs with $d > 1$, however, this simplification is not possible because there is no continuous sorting function on $\mathbb{R}^d$.

In fact we can say something slightly stronger than this. The important property of `sort` for the purpose above is that it maps all permutations of a given set to the same canonical permutation. That is, it has the following two properties:

1. It is permutation-invariant.

2. For any set $\mathbf{x}$, there is a permutation π such that $\texttt{sort}\big(\pi(\mathbf{x})\big) = \pi(\mathbf{x})$.

We refer to any function satisfying these properties as a *quasi-sorting function*, since this definition includes many functions which cannot be defined by sorting according to any order relation. For sets of d-dimensional elements, there is not only no continuous sorting function, there is also no continuous quasi-sorting function. To see this, we give an informal argument which can easily be made precise.

Consider the simplest case, a set of two 2-dimensional elements, and let g be a quasi-sorting function. We can visualise a set of two 2-dimensional elements as being

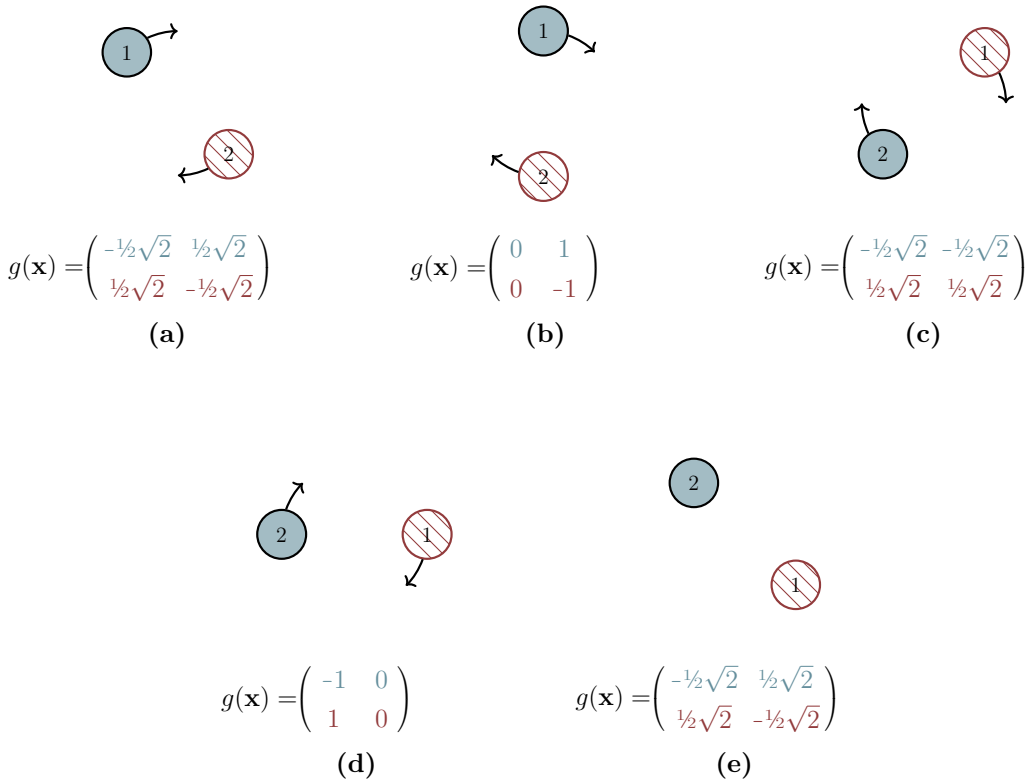


Figure A.2: Rotating a pair of points around their midpoint, and seeing how the canonical ordering changes. Figures A.2a and A.2e represent the same set, so the canonical ordering $g(\mathbf{x})$ must be the same in both cases. This is visually represented by the colours of the points, where the solid blue point occupies the first row of $g(\mathbf{x})$ and the striped red point occupies the second row of $g(\mathbf{x})$. In order for the colours in A.2a and A.2e to match, point "1" must switch from blue to red at some point during the 180-degree rotation from A.2a to A.2e. This switching of colours represents swapping the rows of $g(\mathbf{x})$, which is a discontinuity.

a pair of points in the plane. As is the case throughout this paper, we are working with an ordered representation of this set, in this case as an element $\mathbf{x}$ of $\mathbb{R}^{2\times 2}$. That is, we have a 2×2 matrix $\mathbf{x}$, where each row gives the coordinates of one point.

Now rotate this pair of points around their midpoint, in a continuous motion, through an angle of 180 degrees. This motion is illustrated in Figure A.2, with the point in the first row of $\mathbf{x}$ labelled "1" and the point in the second row labelled "2". The points are coloured according to which row they belong to in the canonical permutation $g(\mathbf{x})$, with solid blue indicating the first row and striped red indicating the second row.

After a 180-degree rotation, the set is the same as it was at the beginning. The canonical permutation, i.e. the colouring, is also the same as at the beginning. This means that at some point during the rotation, the colour of each point has switched – in Figure A.2, this happens between A.2b and A.2c. This switching of colours corresponds to swapping the rows of the output matrix. Since the rows of the matrix are distinct (because the points representing each row are a constant distance apart), this swapping is discontinuous, and therefore g is not continuous.

A.2 The Importance of Continuity

This section was previously presented in Wagstaff et al. [1], and is included in this appendix for completeness. Proofs of the theorems stated here can be found in Appendix A.3.

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. [12], the authors demonstrate that when $\mathfrak{U}$ is a countable set, e.g. the rational numbers, any function $f : 2^{\mathfrak{U}} \to \mathbb{R}$ is sum-decomposable via $\mathbb{R}$. This is taken as a hopeful indication that sum-decomposability may extend to uncountable domains, e.g. $\mathfrak{U} = \mathbb{R}$. 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.

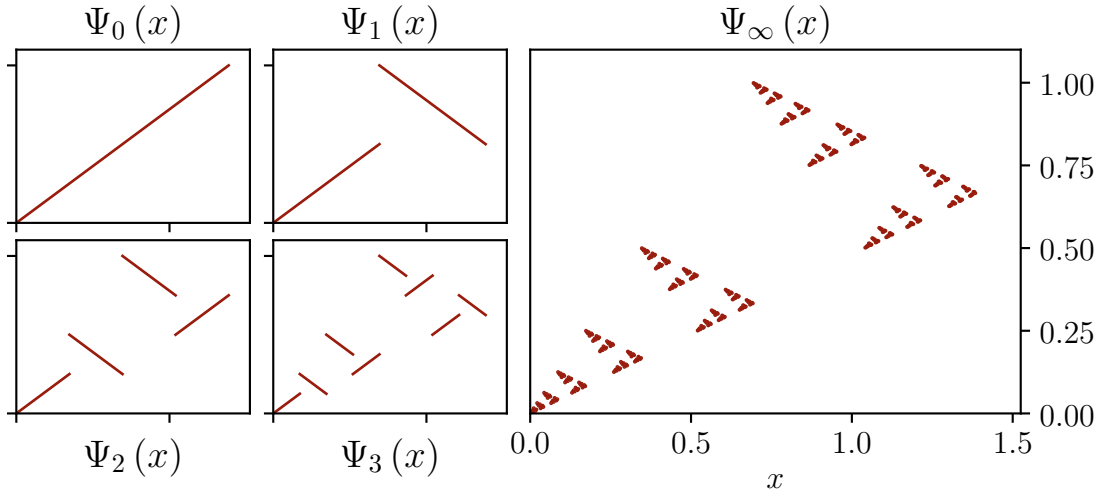


Figure A.3: The function Ψ shown here is continuous at every rational point in $[0, \ln 4]$. Intuitively, this is because all jumps occur at irrational values, namely at certain fractions of $\ln 4$. It defies our intuitions for what continuity should mean, and illustrates the fact that continuity on $\mathbb{Q}$ is a much weaker property than continuity on $\mathbb{R}$. The latter property is required to satisfy the universal approximation theorem for neural networks. Ψ is defined and discussed in Appendix A.4.

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 [49]. A similar statement is true for other approximators, such as some Gaussian processes [6]. 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 A.3 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 4.11. First, the theorem does not consider the continuity of the sum-decomposition when the domain $\mathfrak{U}$ has some non-trivial topological structure (e.g. $\mathfrak{U} = \mathbb{Q}$). 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 4.11, which goes via $\mathbb{R}$, cannot be made continuous for $\mathfrak{U} = \mathbb{Q}$:

Theorem A.1. *There exist functions $f : 2^{\mathbb{Q}} \to \mathbb{R}$ such that, whenever (ρ, ϕ) is a sum-decomposition of f via $\mathbb{R}$, ϕ 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 ϕ to suffice for any f, then ϕ can only be continuous at isolated points of the underlying set $\mathfrak{U}$, regardless of whether $\mathfrak{U} = \mathbb{Q}$. 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 ϕ with a neural network. We therefore view Theorem 4.11 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 $\mathfrak{U}$ 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 A.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 A.3. *If $\mathfrak{U}$ is uncountable, then there exist functions $f : 2^{\mathfrak{U}} \to \mathbb{R}$ which are not sum-decomposable. Note that this holds even if the sum-decomposition (ρ, ϕ) 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.

A.3 Proofs and Additional Theorems

A.3.1 Theorem A.1

Theorem A.1. *There exist functions $f : 2^{\mathbb{Q}} \to \mathbb{R}$ such that, whenever (ρ, ϕ) is a sum-decomposition of f via $\mathbb{R}$, ϕ is discontinuous at every point $q \in \mathbb{Q}$.*

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

$$\mathtt{sup}(X) = \rho(\Phi(X)) \,.$$

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:

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

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 $\mathtt{sup}$ is undefined for unbounded X, we do not consider such sets, and may allow Φ

to diverge). Even if we allowed the domain of ρ to be $\mathbb{R} \cup \{\infty\}$, suppose $\Phi(X) = \infty$ for some upper-bounded set X. Then:

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

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

A.3.2 Theorem A.2

Theorem A.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 ϕ such that Φ 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* (f.s.d.) 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 ϕ to be a bijection from $\mathbb{R}$ to D. Then, by finite-sum-distinctness, Φ 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 all 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 $\exists c_1, \ldots, c_m, d_1, \ldots, d_n \in D$ such that $c_1 + \cdots + c_m + x = d_1 + \cdots + d_n$. 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. That is:

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

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_{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_{ub} is f.s.d.

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

$$\exists c_1, \ldots, c_m, d_1, \ldots, d_n \in C_{\text{ub}} : \Sigma_i c_i = \Sigma_j d_j \,. \tag{A.4}$$

But now by construction of C_{ub} there must be sets $C_1, \ldots, C_m, D_1, \ldots, D_m \in \mathcal{C}$ 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 (A.4), $B_{\max}$ is not f.s.d., which is a contradiction. So we have that C_{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, $D_{\max}$.
3. We have shown that any such set must have cardinality $\mathfrak{c}$.
4. Given an f.s.d. set $D_{\max}$ with cardinality $\mathfrak{c}$, we can choose ϕ to be a bijection between $\mathbb{R}$ and $D_{\max}$.
5. Given such a ϕ, we have that $\Phi(X) = \Sigma_{x \in X} \phi(x)$ is injective on $R^{\mathcal{F}}$.
6. Given injective Φ, 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. □

A.3.3 Theorem A.3

Theorem A.3. *If $\mathfrak{U}$ is uncountable, then there exist functions $f : 2^{\mathfrak{U}} \to \mathbb{R}$ which are not sum-decomposable. Note that this holds even if the sum-decomposition (ρ, ϕ) is allowed to be discontinuous.*

Proof. Consider $f(X) = \texttt{sup}(X)$.

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 A.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 $\mathfrak{U}$.

Even restricting to countable subsets is not enough. As in the proof of Theorem A.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 $\mathfrak{U}_n$ be the set of all $x \in \mathfrak{U}$ with $\phi(x) > \frac{1}{n}$. Since $\phi(x) \neq 0$, we know that $\mathfrak{U} = \bigcup \mathfrak{U}_n$. But this is a countable union of finite sets, which is impossible because $\mathfrak{U}$ is uncountable.

□

A.3.4 Theorem 4.14

Theorem 4.14. *Let $M \in \mathbb{N}$, and let $f : \mathbb{R}^M \to \mathbb{R}$ be a continuous permutation-invariant function. Then f is continuously sum-decomposable via $\mathbb{R}^M$.*

Proof. The reverse implication is clear. We already know from Zaheer et al. [12] that the function $\Phi : \Delta_M \to \mathbb{R}^{M+1}$ 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 $\mathrm{Im}(\Phi) = \{M\} \times \mathrm{Im}(\widetilde{\Phi})$. Since $\{M\}$ is a singleton, these two images are homeomorphic, with a homeomorphism given by

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

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 $\mathbb{R}^M$. □

A.3.5 Theorem 4.15

Theorem 4.15 (Variable set size)**.** *Denote the set of subsets of* $[0,1]$ *containing at most* M *elements by* $[0,1]^{\leq M}$*. Let* $f : [0,1]^{\leq M} \to \mathbb{R}$ *be continuous*[2] *and permutation-invariant. Then* f *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 Φ,

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

which is shown above to be injective (up to reordering input sets). We separate $\Phi_q(X)$ into two terms:

$$\Phi_q(X) = \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(x_m)\,. \tag{A.5}$$

For an input set X with $M' = M - P$ elements, with $P \geq 0$, we say that the set contains M' "actual elements" as well as P "empty" elements which are not 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. when $M' < M$.

We fill in these P elements with a constant value $k \notin [0,1]$, preserving the injectivity of $\Phi_q(X)$ for input sets X of arbitrary size M':

$$\Phi_q(X) = \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(k)\,. \tag{A.6}$$

[2]Note that we must take some care with the notion of continuity here – see Appendix A.1.2.

Equation (A.6) is no longer strictly speaking a sum-decomposition. This can be fixed by re-arranging the expression:

$$\begin{aligned}\Phi_q(X) &= \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(k) \\ &= \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=1}^{M} \phi_q(k) - \sum_{m=1}^{M'} \phi_q(k) \\ &= \sum_{m=1}^{M'} [\phi_q(x_m) - \phi_q(k)] + \sum_{m=1}^{M} \phi_q(k)\,.\end{aligned} \tag{A.7}$$

The last term in Equation (A.7) 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

$$\begin{aligned}\widehat{\Phi}_q(X) &= \sum_{m=1}^{M'} \widehat{\phi}_q(x_m) \\ &= \Phi_q(X) - M \cdot \phi_q(k).\end{aligned} \tag{A.8}$$

$\widehat{\Phi}$ is injective since Φ is injective 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 4.12 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, since the domain of $\widehat{\Phi}_{m,M}^{-1}$ is just the image of the compact set $[0, 1]$ under the continuous function $\widehat{\Phi}_m$. The domains of the $\widehat{\Phi}_{m,M}^{-1}$ are also disjoint, since $k \notin [0, 1]$. We can therefore find a function $\widehat{\Phi}_C^{-1}$ which is continuous on $\mathbb{R}^M$ 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 $k \in \mathcal{Y}, [0, 1] \subset \mathcal{Y}$. Let $\widehat{f}$ be a function on subsets of $\mathcal{Y}$ of size exactly M satisfying

$$\widehat{f}(X) = f(X); \quad X \subset [0, 1]\,,$$
$$\widehat{f}(X) = f(X \cap [0, 1]); \quad X \subset [0, 1] \cup \{k\}\,.$$

We can choose $\widehat{f}$ to be continuous because f is continuous. Then $(\widehat{f} \circ \widehat{\Phi}_C^{-1}, \widehat{\phi})$ is a continuous sum-decomposition of f.

□

A.3.6 Max-Decomposition

Analogously to sum-decomposition, we define the notion of *max-decomposition*. A function f is max-decomposable if there are functions ρ and ϕ such that

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

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 4.13, but which establishes limitations on max-decomposition, rather than sum-decomposition.

Theorem A.4. *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.

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

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

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

$$\widetilde{x}_i = x_i; i \neq m\,, \tag{A.9}$$

$$\widetilde{x}_m = x_{\mu(1)}\,. \tag{A.10}$$

Then:

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

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 ϕ cannot form part of a max-decomposition for f. But ϕ was arbitrary, so no max-decomposition exists.

□

A.3.7 A stronger statement of Theorem 4.19

Given a function $f : [\text{-}1,1]^M \to \mathbb{R}$, write V_f for the *variation* of f:

$$V_f = \max_{\mathbf{x}}\Big(f(\mathbf{x})\Big) - \min_{\mathbf{x}}\Big(f(\mathbf{x})\Big)\,.$$

Theorem (Strong Form of Theorem 4.19)**.** *Let $M, N \in \mathbb{N}$ with $M > N$. There exists a continuous, non-constant, permutation-invariant function $f : [\text{-}1,1]^M \to \mathbb{R}$ such that:*

1. *f is affine on Δ_M.*
2. *For any continuous sum-decomposition $\rho\circ\Phi$ via $\mathbb{R}^N$, there is some $\mathbf{x} \in [\text{-}1,1]^M$ with $|\rho\Big(\Phi(\mathbf{x})\Big) - f(\mathbf{x})| \geq \frac{V_f}{2}$.*

A.3.8 Completing the proof of Lemma 4.23

Lemma 4.23. *Let $n \in \mathbb{N}$. Let Γ_n be defined as in Equation (4.14). Then there exists a continuous function $\nu_n : I_n \to \Delta_n$ such that, for $\mathbf{x} \in \partial I_n$*

$$\Gamma_n\Big(\nu_n(\text{-}\mathbf{x})\Big) = -\Gamma_n\Big(\nu_n(\mathbf{x})\Big)\,. \tag{4.22}$$

Proof. We proceed by induction. We state our induction hypotheses as follows, in the order in which we prove the corresponding conclusions:

Hypothesis 1 ν_{n-1} is continuous.

Hypothesis 2 Let $\mathbf{x} \in I_{n-1}$ and $1 \leq j < n-1$.
Then $\nu_{n-1}(\mathbf{x})_j \geq \nu_{n-1}(\text{-}\mathbf{x})_{j+1}$.

Hypothesis 3 The codomain of ν_{n-1} is Δ_{n-1}.

Hypothesis 4 Let $\mathbf{x} \in \partial I_{n-1}$. Then $\Gamma_{n-1}\Big(\nu_{n-1}(\text{-}\mathbf{x})\Big) = -\Gamma_{n-1}\Big(\nu_{n-1}(\mathbf{x})\Big)$.

Hypotheses 1, 3 and 4 come from the statement of the lemma. Hypothesis 2 is a technical condition which is required for the proof. We will refer to these statements as "hypotheses" when referring to the $n-1$ case, and "conclusions" when referring to the n case.

Base case $n = 1$

Let $\nu_1(\mathbf{x}) \coloneqq \mathbf{x}$. Conclusions 1 and 3 are trivial. Conclusion 2 is also trivial – there is no j with $1 \leq j < 1$, and so there are no inequalities to satisfy. $\partial I_1 = \{\text{-}1, 1\}$, so conclusion 4 reduces to checking that $\Gamma_1(\text{-}1) = -\Gamma_1(1)$. This is true by definition of Γ.

Inductive step

Conclusion 1 – Defining the continuous function ν_n

We define $\nu_n^+ : I_n^+ \to I_n$, $\nu_n^- : I_n^- \to I_n$ as follows:

$$\nu_n^+(\mathbf{x}) \coloneqq \Big(1, \nu_{n-1}(\overline{\mathbf{x}})\Big),$$
$$\nu_n^-(\mathbf{x}) \coloneqq \Big(\nu_{n-1}(\text{-}\overline{\mathbf{x}}), \text{-}1\Big) = \alpha\Big(\nu_n^+(\text{-}\mathbf{x})\Big).$$

These definitions follow the partial definition of ν_n given in Section 4.4.2. ν_n^+ and ν_n^- are both continuous by continuity of ν_{n-1} (hypothesis 1). We will define ν_n to agree with ν_n^+ and ν_n^- on their domains, and extend ν_n to the rest of I_n by interpolating between ν_n^+ and ν_n^-. As discussed in Section 4.4.2, any point $\mathbf{x} \in I_n$ lies on a line running parallel to the n-th coordinate axis, and this line meets I_n^+, I_n^- at $\mathbf{x}^+, \mathbf{x}^-$ respectively. We interpolate along these "vertical" lines to define ν_n on the rest of I_n. Simple linear interpolation does not define a function with the right properties, so we instead define the following piecewise linear interpolation scheme.

Divide the straight line segment from $\mathbf{x}^+$ to $\mathbf{x}^-$ into n sections of equal length, with endpoints $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(n+1)}$ (so $\mathbf{x}_{(1)} = \mathbf{x}^+$ and $\mathbf{x}_{(n+1)} = \mathbf{x}^-$). We fix the value of ν_n at each endpoint and linearly interpolate between them. For compactness, we write $\mathbf{y}^+ = \nu_n^+(\mathbf{x}^+)$ and $\mathbf{y}^- = \nu_n^-(\mathbf{x}^-)$, and make the following definition:

$$m_j \coloneqq \texttt{median}(y_j^+, y_{j-1}^-, y_j^-)\,.$$

Then we fix ν_n at the endpoints as follows:

$$\nu_n(\mathbf{x}_{(i)})_j \coloneqq \begin{cases} y_j^+ & i < j \\ m_j & i = j \\ y_j^- & i > j \end{cases}\,. \tag{A.11}$$

The special case $i = j = 1$ depends on $\nu_n^-(\mathbf{x}^-)_0$, which is undefined – for consistency with ν_n^+, we must set $\nu_n(\mathbf{x}_{(1)})_1 = y_1^+$. Figure A.4 gives an example of how this interpolation scheme works for $n = 5$.

This interpolation scheme defines a continuous function ν_n on I_n, by continuity of ν_n^+, ν_n^-, `median`, and linear interpolation. Conclusion 1 is thus established.

In the rest of the proof, we will often use the more verbose notation when referring to $\mathbf{y}^\pm$, writing $\nu_n^\pm(\mathbf{x}^\pm)$ to make the dependence on $\mathbf{x}$ explicit. We will use the compact notation $\mathbf{y}^\pm$ where this dependence is less important.

Conclusion 2 – A useful inequality

First we show that for every j, we have

$$y_j^+ \geq y_j^-\,. \tag{A.12}$$

By definition:

$$y_j^+ = \nu_n(\mathbf{x}^+)_j = \begin{cases} 1 & j = 1 \\ \nu_{n-1}(\overline{\mathbf{x}})_{j-1} & j > 1 \end{cases},$$
$$y_j^- = \nu_n(\mathbf{x}^-)_j = \begin{cases} \nu_{n-1}(\text{-}\overline{\mathbf{x}})_j & j < n \\ \text{-1} & j = n \end{cases}.$$

Equation (A.12) is therefore trivial for $j = 1$ and $j = n$, since hypothesis 3 implies that ν_{n-1} is bounded between -1 and 1. For $1 < j < n$, Equation (A.12) is exactly hypothesis 2. Note also from the above definitions that

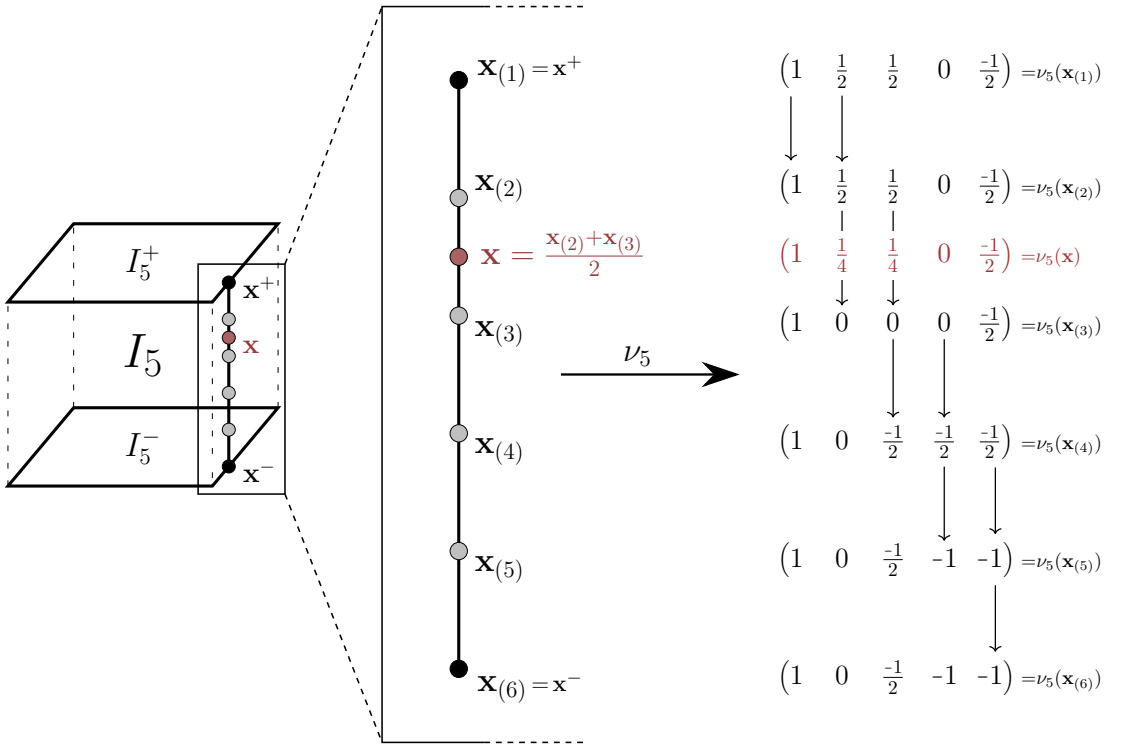


Figure A.4: An example of how the interpolation scheme works for $n = 5$, with $\mathbf{x}$ chosen to lie on a surface vertical. The specific values of $\nu_5(\mathbf{x}_{(1)})$ and $\nu_5(\mathbf{x}_{(6)})$ shown here are purely illustrative, but the remaining values are computed according to our interpolation scheme. Equation (A.11) directly implies that the value of ν_n at successive endpoints can differ only along the indicated vertical arrows – for instance, $\nu_n(\mathbf{x}_{(1)})$ and $\nu_n(\mathbf{x}_{(2)})$ can differ only in the first and second places. One perspective on this is to view the function values $\nu_n(\mathbf{x}_{(i)})_j$ as the entries of a matrix $\mathcal{N}_{ij}$ – Equation (A.11) fixes the above- and below-diagonal entries of $\mathcal{N}$ to match $\mathbf{y}^+$ and $\mathbf{y}^-$ respectively, so values can only change across the diagonal. Crossing the diagonal corresponds to moving along the vertical arrows shown above.

The final part of the proof of Lemma 4.23 demonstrates that, along a surface vertical, the values of ν_n either do not change from endpoint to endpoint, as seen here between $\mathbf{x}_{(1)}$ and $\mathbf{x}_{(2)}$, or they change in equal pairs, as seen here between $\mathbf{x}_{(2)}$ and $\mathbf{x}_{(3)}$. In either case, the value of $\Gamma_n \circ \nu_n$ is not affected – trivially in the former case, and in the latter case because equal pairs cancel to 0 in the alternating sum which defines Γ_n (Equation 4.14). Importantly, $\Gamma_n \circ \nu_n$ is constant not only at consecutive endpoints, but also when linearly interpolating between the endpoints. In the above, for instance, $\Gamma_5(\nu_5(\mathbf{x})) = \Gamma_5(\nu_5(\mathbf{x}_{(2)})) = \Gamma_5(\nu_5(\mathbf{x}_{(3)}))$, because the second and third coordinates are equal and cancel to 0 under Γ_n. To show that this Γ-equivalence property always holds, we rely on ν_n being sufficiently "well-behaved" on I_n^+ and I_n^-. In particular we will see that Equation (A.16), which relies on hypothesis 2, is crucial.

Here we have illustrated the interpolation scheme along a surface vertical. The same interpolation scheme applies across all of I_n, with the same behaviour of changing only along the vertical arrows. However, the Γ-equivalence property discussed above only holds on surface verticals, and does not hold on the rest of I_n.

$$\nu_n\Big(\mathbf{x}^-\Big)_j = \nu_n\Big((\text{-}\mathbf{x})^+\Big)_{j+1} \qquad 1 \le j < n. \tag{A.13}$$

Equation (A.12) implies, by definition of m_j, that $y_j^+ \ge m_j \ge y_j^-$ for all j. Therefore by Equation (A.11):

$$\nu_n(\mathbf{x}_{(i)})_j \ge \nu_n(\mathbf{x}_{(i+1)})_j \qquad 1 \le i \le n, \text{ all } j\,. \tag{A.14}$$

This implies that for all $\mathbf{x}$ and for all j, we have

$$\nu_n(\mathbf{x}^+)_j \ge \nu_n(\mathbf{x})_j \ge \nu_n(\mathbf{x}^-)_j\,.$$

Therefore for $1 \le j < n$, by Equation (A.13), we have

$$\begin{aligned}\nu_n(\mathbf{x})_j &\ge \nu_n(\mathbf{x}^-)_j \\ &= \nu_n((\text{-}\mathbf{x})^+)_{j+1} \\ &\ge \nu_n(\text{-}\mathbf{x})_{j+1}\,.\end{aligned}$$

This final inequality is exactly conclusion 2.

Conclusion 3 – The codomain of ν_n is Δ_n

We must show that $\nu_n(\mathbf{x})_j$ is a descending sequence in j. On I_n^+ and I_n^-, this follows immediately by definition and by hypothesis 3. Away from I_n^+ and I_n^-, we must consider the interpolation scheme. We want $\nu_n(\mathbf{x})_j \ge \nu_n(\mathbf{x})_{j+1}$ for all j. We only need to check this inequality at the endpoints $\mathbf{x}_{(i)}$ of the interpolation scheme – if it holds at the endpoints, then the function value at any other point is a convex combination of points from Δ_n, and Δ_n is convex.

The relevant inequality is therefore as follows:

$$\nu_n(\mathbf{x}_{(i)})_j \ge \nu_n(\mathbf{x}_{(i)})_{j+1} \qquad \text{all } i, 1 \le j < n\,. \tag{A.15}$$

To prove this, we will make use of the following inequality, which holds for $1 < j < n$:

$$y_{j-1}^- \ge y_{j+1}^+\,. \tag{A.16}$$

Expanding definitions, $y_{j-1}^- = \nu_{n-1}(\text{-}\overline{\mathbf{x}})_{j-1}$ and $y_{j+1}^+ = \nu_{n-1}(\overline{\mathbf{x}})_j$. Equation (A.16) therefore follows directly from hypothesis 2.

To prove Equation (A.15), there are five cases to check. For case 4, we make use of the fact that $a \geq \texttt{median}(a, b, c)$ if and only if $a \geq b$ or $a \geq c$. For case 5, we make use of the fact that $\texttt{median}(a, b, c) \geq \texttt{min}(a, b)$.

1. $i < j$. Expanding definitions, we need

$$y_j^+ \geq y_{j+1}^+.$$

   This is the I_n^+ case, which as already noted is true by definition and induction.

2. $i = j = 1$. This is also the I_n^+ case.

3. $i > j + 1$. Similarly, this is the I_n^- case.

4. $i = j + 1$. Expanding definitions, we need

$$y_j^- \geq \texttt{median}\Big(y_{j+1}^+, y_j^-, y_{j+1}^-\Big).$$

   This is true because $y_j^- \geq y_{j+1}^-$, which is the I_n^- case.

5. $i = j > 1$. Expanding definitions, we need

$$\texttt{median}\Big(y_j^+, y_{j-1}^-, y_j^-\Big) \geq y_{j+1}^+.$$

   This follows because $y_{j+1}^+ \leq \texttt{min}(y_{j-1}^-, y_j^+)$. Equation (A.16) implies $y_{j+1}^+ \leq y_{j-1}^-$, and $y_{j+1}^+ \leq y_j^+$ is the I_n^+ case.

Equation (A.15) therefore holds, so conclusion 3 is established.

4. Conclusion 4 – The Borsuk-Ulam condition

In this section we will freely make use of the following inequalities, which we have already shown to be true:

$$y_j^+ \geq y_{j+1}^+\,, \qquad y_j^- \geq y_{j+1}^-\,,$$
$$y_j^+ \geq y_j^-\,, \qquad y_{j-1}^- \geq y_{j+1}^+\,.$$

The top inequalities hold because $\nu_n(\mathbf{x}) \in \Delta_n$ (conclusion 3). The bottom left inequality is Equation (A.12). The bottom right inequality is Equation (A.16).

Conclusion 4 applies to $\mathbf{x} \in \partial I_n$. As shown in Section 4.4.2, for $\mathbf{x} \in I_n^+ \cup I_n^-$, we have

$$\Gamma_n\Big(\nu_n(\text{-}\mathbf{x})\Big) = -\Gamma_n\Big(\nu_n(\mathbf{x})\Big)\,.$$

The remaining points on the surface are the points $\mathbf{x} \in \partial I_n \backslash (I_n^+ \cup I_n^+)$, i.e. points lying on surface verticals, and for the remainder of the proof we consider only these points. We claim that $\Gamma_n(\nu_n(\mathbf{x})) = \Gamma_n(\nu_n(\mathbf{x}^+))$, i.e. that $\Gamma_n \circ \nu_n$ is constant along surface verticals. As noted in Section 4.4.2, this will imply that conclusion 4 holds on all of ∂I_n. To prove our claim, we first define the relation of Γ-*equivalence* on Δ_n, denoted by $\sim_\Gamma$. We define $\mathbf{x}_1 \sim_\Gamma \mathbf{x}_2$ if and only if $\Gamma_n(\mathbf{x}_1) = \Gamma_n(\mathbf{x}_2)$ for every choice of ϕ. For an alternative view on this relation, given $\mathbf{x} \in \Delta_n$, form $\mathbf{x}_\Gamma$ by deleting pairs of equal coordinates until no matching pairs remain. For example:

$$(1, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, 0, \text{-}\tfrac{1}{2}, \text{-}\tfrac{1}{2})_\Gamma = (1, \tfrac{1}{2}, 0)\,.$$

Then $\mathbf{x}_1 \sim_\Gamma \mathbf{x}_2$ if and only if $(\mathbf{x}_1)_\Gamma = (\mathbf{x}_2)_\Gamma$. The reverse implication follows from the alternating sum form of Γ, so that consecutive equal coordinates cancel to 0. For the forward implication, if $(\mathbf{x}_2)_\Gamma \neq (\mathbf{x}_2)_\Gamma$, then there is some coordinate x_i which appears in $(\mathbf{x}_1)_\Gamma$ but not in $(\mathbf{x}_2)_\Gamma$. Choosing ϕ to be non-zero at this co-ordinate, and zero at every other coordinate of $(\mathbf{x}_1)_\Gamma$ and $(\mathbf{x}_2)_\Gamma$, we obtain $\Gamma_n(\mathbf{x}_1) \neq \Gamma_n(\mathbf{x}_2)$.

Our claim is that the relation $\sim_\Gamma$ is preserved under the interpolation between $\nu_n(\mathbf{x}^+)$ and $\nu_n(\mathbf{x}^-)$. We will prove this by showing that the successive endpoints are Γ-equivalent, and that the interpolation between any two endpoints preserves Γ-equivalence.

Consider two successive endpoints in our interpolation scheme, $\mathbf{x}_{(I)}$ and $\mathbf{x}_{(I+1)}$. By definition, we have $\nu_n(\mathbf{x}_{(I)})_j = \nu_n(\mathbf{x}_{(I+1)})_j$ if $I > j$ or $I + 1 < j$. This leaves $I = j$ or $I + 1 = j$. That is, $\nu_n(\mathbf{x}_{(I)})$ can differ from $\nu_n(\mathbf{x}_{(I+1)})$ only in the I-th and

$I+1$-th coordinates. From the definition of the interpolation scheme, the relevant quantities are as follows:

$$\begin{aligned}
\nu_n(\mathbf{x}_{(I)})_I &= m_I \,, \\
\nu_n(\mathbf{x}_{(I)})_{I+1} &= y^+_{I+1} \,, \\
\nu_n(\mathbf{x}_{(I+1)})_I &= y^-_I \,, \\
\nu_n(\mathbf{x}_{(I+1)})_{I+1} &= m_{I+1} \,.
\end{aligned}$$

There are two special cases here. When $I=1$, $\nu_n(\mathbf{x}_{(1)})_1 = y^+_1$ instead of m_1. When $I=n$, the quantities y^+_{n+1} and m_{n+1} are not defined. For now, we assume $1<I<n$, and we address the special cases at the end of the proof.

We have already shown the following inequalities between the four quantities above – the left-to-right inequalities are because $\nu_n(\mathbf{x}) \in \Delta_n$, and the top-to-bottom inequalities are instances of Equation (A.14).

$$\begin{array}{ccc}
m_I & \geq & y^+_{I+1} \\
\rotatebox{90}{$\geq$} & & \rotatebox{90}{$\geq$} \\
y^-_I & \geq & m_{I+1}
\end{array} \tag{A.17}$$

Recalling the visualisation of Figure A.4, the four quantities above correspond to two pairs of coordinates joined by vertical arrows. For example, $I=3$ corresponds to the transition between $\nu_n(\mathbf{x}_{(3)})_{3,4}$ and $\nu_n(\mathbf{x}_{(4)})_{3,4}$. Another way of seeing the above inequalities is to consider the values $\nu_n(\mathbf{x}_{(i)})_j$ as a matrix $\mathcal{N}_{ij}$, where the matrix $\mathcal{N}$ is (non-strictly) decreasing downwards and decreasing to the right. See Figure A.4 for more on this perspective.

In the above array of four values, we wish to show that either both columns are equal, or both rows are equal. That is, we wish to show that one of the following two cases holds:

1. $m_I = y^+_{I+1}$ and $m_{I+1} = y^-_I$ (columns of Equation A.17 are equal).
2. $m_I = y^-_I$ and $m_{I+1} = y^+_{I+1}$ (rows of Equation A.17 are equal).

In case 1, linear interpolation between endpoints implies that $\nu_n(\mathbf{x})_I = \nu_n(\mathbf{x})_{I+1}$ for every $\mathbf{x}$ on the line segment between $\mathbf{x}_{(I)}$ and $\mathbf{x}_{(I+1)}$. Since these values are all equal in every other coordinate, and since Γ-equivalence is not affected by adding a pair of equal coordinates, all of the values $\nu_n(\mathbf{x})$ for $\mathbf{x}$ on this section of the surface vertical are Γ-equivalent. In case 2, we have $\nu_n(\mathbf{x}_{(I)}) = \nu_n(\mathbf{x}_{(I+1)})$. Linear interpolation between endpoints therefore trivially gives Γ-equivalence, because ν_n is constant between the endpoints $\mathbf{x}_{(I)}$ and $\mathbf{x}_{(I+1)}$. So in either case, Γ-equivalence is preserved by the interpolation, and therefore $\Gamma_n \circ \nu_n$ is constant along surface verticals.

To prove that either case holds, we just need to show that, in Equation (A.17), either the top or the left equality holds, and either the bottom or the right equality holds. If this were true with neither the rows nor the columns being equal, then we would be in one of the contradictory situations shown below.

$$\begin{array}{cc} m_I = y^+_{I+1} & m_I > y^+_{I+1} \\ \vee \quad \| & \| \quad \vee \\ y^-_I > m_{I+1} & y^-_I = m_{I+1} \end{array}$$

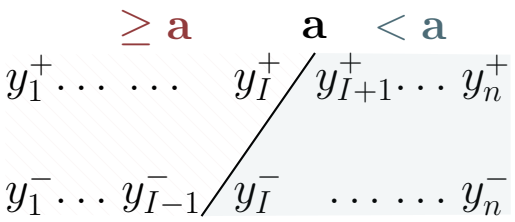
By definition of m_{I+1} and our inequalities on the y_I, it is immediate that we have either the bottom or the right equality – either $m_{I+1} = y^-_I$, or $m_{I+1} = y^+_{I+1}$.

The case for the top and left equalities is more complex. By definition of m_I, and our inequalities on the y_I, we have that $m_I = y^-_I$ if either $y^-_I = y^-_{I-1}$ or $y^-_I = y^+_I$. We have $m_I = y^+_{I+1}$ if either $y^+_{I+1} = y^+_I$ or $y^+_{I+1} = y^-_{I-1}$. So for neither the top nor the left equality to hold, all four of these statements must be false. That is, applying our known inequalities for the y_I, we must have:

$$\begin{array}{cc} y^+_I > y^-_I\,, & y^+_I > y^+_{I+1}\,, \\ y^-_{I-1} > y^-_I\,, & y^-_{I-1} > y^+_{I+1}\,. \end{array} \tag{A.18}$$

In the following representation, letting $a \coloneqq \mathtt{min}(y^+_I, y^-_{I-1})$, these inequalities say that everything to the left of the diagonal line is strictly greater than everything to

the right of the diagonal line. The special case $I = 1$ results in essentially the same picture, with y_1^+ being the only point to the left of the diagonal line.



This implies that there are exactly I coordinates of $\mathbf{y}^+$ which are at least a, and exactly $I - 1$ coordinates of $\mathbf{y}^-$ which are at least a. But now recall that we have shown in Section 4.4.2, specifically Equation (4.23), that

$$\Gamma_n\Big(\nu_n(\mathbf{x}^+)\Big) = \Gamma_n\Big(\nu_n(\mathbf{x}^-)\Big)\,.$$

Since this equation holds regardless of the choice of ϕ, it is equivalent to the statement that $\mathbf{y}^+ \sim_\Gamma \mathbf{y}^-$. We now note the basic but crucial fact that if $\mathbf{y}^+ \sim_\Gamma \mathbf{y}^-$, then for any a, the number of occurrences of a in $\mathbf{y}^+$ must have the same parity as the number of occurrences of a in $\mathbf{y}^-$. This parity matching is also true if we consider the number of occurrences of values which are at least a. But we have just shown that these numbers have differing parity. This is a contradiction, so Equation (A.18) must fail. As shown above, this implies that either $m_I = y_{I+1}^+$ or $m_I = y_I^-$, and this in turn implies that either the rows or the columns of Equation (A.17) must be equal. Therefore $\Gamma \circ \nu$ is constant along the line segment from $\mathbf{x}_{(I)}$ to $\mathbf{x}_{(I+1)}$.

As noted above, there is a final special case to consider, $I = n$. We have shown that $\Gamma \circ \nu$ is constant along the surface vertical between $\mathbf{x}_{(1)}$ and $\mathbf{x}_{(n)}$, but we have not shown that it is constant along the final section of the surface vertical, from $\mathbf{x}_{(n)}$ to $\mathbf{x}_{(n+1)}$. However, we have shown that $\nu_n(\mathbf{x}_{(1)}) \sim_\Gamma \nu_n(\mathbf{x}_{(n)})$ (by transitivity of $\sim_\Gamma$), and that $\nu_n(\mathbf{x}_{(1)}) \sim_\Gamma \nu_n(\mathbf{x}_{(n+1)})$. Applying transitivity of $\sim_\Gamma$ again, this implies that $\nu_n(\mathbf{x}_{(n)}) \sim_\Gamma \nu_n(\mathbf{x}_{(n+1)})$. We also know that $\nu_n(\mathbf{x}_{(n)})$ and $\nu_n(\mathbf{x}_{(n+1)})$ differ in at most one place (the n-th coordinate). It is easily seen that if two points are Γ-equivalent, they must differ in an even number of places. Therefore $\nu_n(\mathbf{x}_{(n)}) = \nu_n(\mathbf{x}_{(n+1)})$, so

$\Gamma \circ \nu$ is constant along the final section of the surface vertical, and conclusion 4 holds.

□

A.4 A Continuous Function on $\mathbb{Q}$

This section defines and analyses the function Ψ shown in Figure A.3, which is continuous on $\mathbb{Q}$ but not on $\mathbb{R}$. Ψ is defined as the pointwise limit of a sequence of functions Ψ_n, illustrated in Figure A.5. 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 Ψ on $[0, A]$ by $\Psi(x) = \widetilde{\Psi}\left(\frac{x}{A}\right)$.
4. Note that Ψ 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 subintervals of equal width. 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:

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

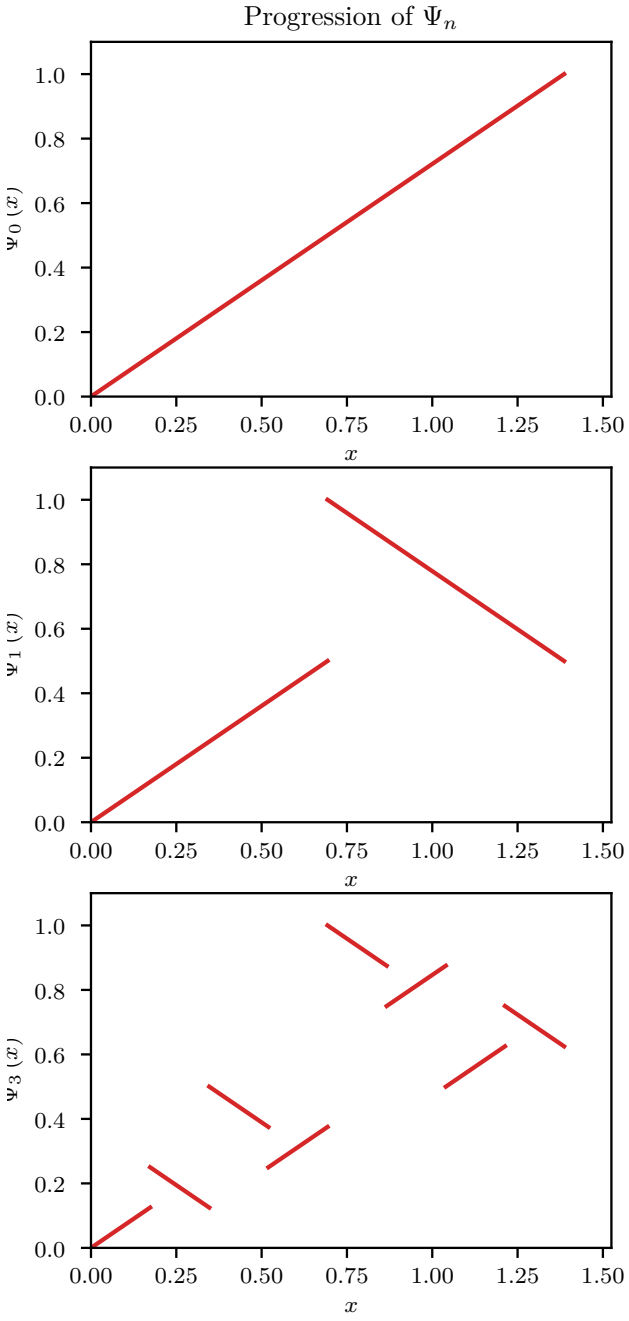


Figure A.5: Several iterations of Ψ_n

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:

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

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

$$|\widetilde{\Psi}_n(x) - \widetilde{\Psi}(x)| \leq 2^{-n}. \tag{A.19}$$

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 (A.19):

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

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 $\Psi(x) = \widetilde{\Psi}\left(\frac{x}{A}\right)$ to obtain a function which is continuous at all rational points.

B
Appendix to Chapter 5

B.1 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 optimiser 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 optimising when the norm of every position update is below 0.001. We train the SE(3)-Transformers for 100 epochs with 5000 examples per epoch, which takes about 90 to 120 minutes on an Nvidia Titan V GPU. We did not measure the runtime increase of backpropagating through the basis functions in a controlled environment, but an informal analysis suggested a single digit percentage increase in wall-clock time.

References

[1] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. "On the Limitations of Representing Functions on Sets". In: *International Conference on Machine Learning (ICML)* (2019).

[2] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2005.

[3] Philipp Hennig, Michael A Osborne, and Mark Girolami. "Probabilistic Numerics and Uncertainty in Computations". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2179 (June 2015). arXiv: 1506.01326. URL: http://dx.doi.org/10.1098/rspa.2015.0142.

[4] A. O'Hagan. "Bayes-Hermite quadrature". In: *Journal of Statistical Planning and Inference* 29.3 (1991), pp. 245–260.

[5] Harrison Zhu, Xing Liu, Ruya Kang, Zhichao Shen, Seth Flaxman, and Francois-Xavier Briol. "Bayesian Probabilistic Numerical Integration with Tree-Based Models". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 5837–5849. URL: https://proceedings.neurips.cc/paper/2020/file/3fe94a002317b5f9259f82690aeea4cd-Paper.pdf.

[6] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[7] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[8] Yann LeCun and Yoshua Bengio. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[9] M.A. Armstrong. *Groups and Symmetry*. Undergraduate texts in mathematics. Springer, 1988.

[10] C.C. Pinter. *A Book of Abstract Algebra: Second Edition*. Dover Books on Mathematics. Dover Publications, 2010.

[11] J. Rotman. *An Introduction to the Theory of Groups*. Graduate Texts in Mathematics. Springer New York, 1999.

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

[13] Benjamin Bloem-Reddy and Yee Whye Teh. "Probabilistic Symmetries and Invariant Neural Networks." In: *J. Mach. Learn. Res.* 21 (2020), pp. 90–1.

[14] Christian Bueno and Alan G. Hylton. "On the Representation Power of Set Pooling Networks". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021. URL: https://papers.nips.cc/paper/2021/file/8ea1e4f9f24c38f168d538c9cfc50a14-Paper.pdf.

[15] M.G. Kivelson and C.T. Russell. *Introduction to Space Physics*. Cambridge atmospheric and space science series. Cambridge University Press, 1995.

[16] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[17] Björn Bornkamp. "Approximating Probability Densities by Iterated Laplace Approximations". In: *Journal of Computational and Graphical Statistics* 20.3 (Mar. 2011), pp. 656–669. arXiv: 1103.3508. URL: http://dx.doi.org/10.1198/jcgs.2011.10099.

[18] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (Jan. 2016), pp. 859–877. arXiv: 1601.00670. URL: http://dx.doi.org/10.1080/01621459.2017.1285773.

[19] A. O'Hagan. "Monte Carlo is Fundamentally Unsound". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 36.2 (1987), pp. 247–249.

[20] Tom Gunter, Michael A. Osborne, Roman Garnett, Philipp Hennig, and Stephen J. Roberts. "Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature". In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 1–9. arXiv: 1411.0439. URL: http://arxiv.org/abs/1411.0439.

[21] Toni Karvonen and Simo Särkkä. "Classical quadrature rules via Gaussian processes". In: *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. 2017, pp. 1–6.

[22] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. "Batch bayesian optimization via local penalization". In: *Artificial Intelligence and Statistics*. 2016, pp. 648–657.

[23] Persi Diaconis. "Bayesian Numerical Analysis". In: *Statistical Decision Theory and Related Topics IV*. New York, NY: Springer New York, 1988, pp. 163–175. URL: http://link.springer.com/10.1007/978-1-4613-8768-8%7B%5C_%7D20.

[24] Thomas Minka. "Deriving quadrature Rules from Gaussian processes". In: *Technical report, Statistics Department, Carnegie Mellon University* (2000), pp. 1–21. URL: http://131.107.65.14/en-us/um/people/minka/papers/minka-quadrature.ps.gz.

[25] Carl E. Rasmussen and Zoubin Ghahramani. "Bayesian Monte Carlo". In: *Advances in Neural Information Processing Systems*. Oct. 2003.

[26] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. 2006, p. 248.

[27] François-Xavier Briol, Chris. J. Oates, Mark Girolami, Michael A. Osborne, and Dino Sejdinovic. "Probabilistic Integration: A Role in Statistical Computation?" In: (2015), pp. 1–49. arXiv: 1512.00933. URL: http://arxiv.org/abs/1512.00933.

[28] Michael A. Osborne, David Duvenaud, Roman Garnett, Carl E. Rasmussen, Stephen J. Roberts, and Zoubin Ghahramani. "Active Learning of Model Evidence Using Bayesian Quadrature". In: *Advances in Neural Information Processing Systems* 1 (2012), pp. 46–54.

[29] Henry Chai and Roman Garnett. "An Improved Bayesian Framework for Quadrature". 2018. arXiv: 1802.04782.

[30] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. "Kriging Is Well-Suited to Parallelize Optimization". In: *Computational Intelligence in Expensive Optimization Problems* 2 (2010), pp. 131–162.

[31] Vu Nguyen, Santu Rana, Sunil Gupta, Cheng Li, and Svetha Venkatesh. "Budgeted Batch Bayesian Optimization With Unknown Batch Sizes". In: *IEEE International Conference on Data Mining, ICDM*. Mar. 2017, pp. 1107–1112. arXiv: 1703.04842. URL: http://arxiv.org/abs/1703.04842.

[32] Donald R Jones. "A taxonomy of global optimization methods based on response surfaces". In: *Journal of global optimization* 21.4 (2001), pp. 345–383.

[33] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[34] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (1970), pp. 97–109. URL: http://dx.doi.org/10.1093/biomet/57.1.97.

[35] Matthew D Hoffman and Andrew Gelman. "The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1593–1623.

[36] 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.

[37] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017). arXiv: arXiv:1612.00593v2.

[38] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. "A simple neural network module for relational reasoning". In: *Advances in Neural Information Processing Systems* (June 2017).

[39] Peter W. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: (2018), pp. 1–38. arXiv: 1806.01261. URL: http://arxiv.org/abs/1806.01261.

[40] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* (2017). arXiv: arXiv:1706.03762v5.

[41] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. "Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks". In: *International Conference on Machine Learning (ICML)* (2019).

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". In: *ICLR* (2018).

[43] Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. "Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs". In: *International Conference on Learning Representations (ICLR)* (2019).

[44] David W. Romero and Jean-Baptiste Cordonnier. "Group Equivariant Stand-Alone Self-Attention For Vision". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=JkfYjnOEo6M.

[45] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems* (2017). arXiv: arXiv:1706.02413v1.

[46] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. "Interaction Networks for Learning about Objects, Relations and Physics". In: *Advances in Neural Information Processing Systems* (2016). arXiv: 1612.00222.

[47] Fabian B. Fuchs, Adam R. Kosiorek, Li Sun, Oiwi Parker Jones, and Ingmar Posner. "End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning". In: *Sets and Parts Workshop, NeurIPS* (2019).

[48] Jiequn Han, Yingzhou Li, Lin Lin, Jianfeng Lu, Jiefu Zhang, and Linfeng Zhang. *Universal approximation of symmetric and anti-symmetric functions*. 2019. arXiv: 1912.01765 [math.NA].

[49] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), pp. 303–314.

[50] Nimrod Segol and Yaron Lipman. *On Universal Equivariant Set Networks*. 2020. arXiv: 1910.02421 [cs.LG].

[51] MH Stone. "The Generalized Weierstrass Approximation Theorem". In: *Mathematics Magazine* 21.4 (1948), pp. 167–184.

[52] Ken-Ichi Funahashi. "On the approximate realization of continuous mappings by neural networks". In: *Neural networks* 2.3 (1989), pp. 183–192.

[53] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[54] Karol Borsuk. "Drei Sätze über die n-dimensionale euklidische Sphäre". ger. In: *Fundamenta Mathematicae* 20.1 (1933), pp. 177–190. URL: http://eudml.org/doc/212624.

[55] N.G. Lloyd. *Degree Theory*. Cambridge Tracts in Mathematics. Cambridge University Press, 1978.

[56] Yoshua Bengio, Nicholas Leonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." In: *ArXiv: 1308.3432* (2013).

[57] Penghang 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).

[58] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. "Fast Differentiable Sorting and Ranking". In: *International Conference on Machine Learning* (2020).

[59] Meng Liu, Zhengyang Wang, and Shuiwang Ji. "Non-Local Graph Neural Networks". In: *ArXiv* (2020).

[60] Chirag Pabbaraju and Prateek Jain. "Learning Functions over Sets via Permutation Adversarial Networks". In: *ArXiv* (2020).

[61] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *NeurIPS* (2016).

[62] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Pratim Talukdar. "HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Classification". In: (2018). URL: http://arxiv.org/abs/1809.02589.

[63] Jiayi Wei, Maruth Goyal, Greg Durrett, and Isil Dillig. "LambdaNet: Probabilistic Type Inference using Graph Neural Networks". In: *ICLR* (2020).

[64] 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 Systems 33 (NeurIPS)*. 2020.

[65] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. "Invariant and Equivariant Graph Networks". In: *ICLR* (2019).

[66] Marc Finzi, Max Welling, and Andrew Gordon Wilson. "A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 3318–3328. URL: https://proceedings.mlr.press/v139/finzi21a.html.

[67] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/bb04af0f7ecaee4aae62035497da1387-Paper.pdf.

[68] Recep Adiyaman and Liam James McGuffin. "Methods for the refinement of protein structure 3D models". In: *International journal of molecular sciences* 20.9 (2019), p. 2301.

[69] John Jumper et al. "High accuracy protein structure prediction using deep learning". In: *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)* 22 (2020), p. 24.

[70] Brian Kuhlman and Philip Bradley. "Advances in protein structure prediction and design". In: *Nature Reviews Molecular Cell Biology* 20.11 (2019), pp. 681–697.

[71] 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.

[72] Lupoglaz. *OpenFold2*. `https://github.com/lupoglaz/OpenFold2/tree/toy_se3`. 2021.

[73] Phil Wang. *SE3 Transformer - Pytorch*. `https://github.com/lucidrains/se3-transformer-pytorch`. 2021.

[74] Dale Markowitz. *AlphaFold 2 Explained: A Semi-Deep Dive*. `https://towardsdatascience.com/alphafold-2-explained-a-semi-deep-dive-fa7618c1a7f6`. 2020.

[75] 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.

[76] UniProt Consortium. "UniProt: a worldwide hub of protein knowledge". In: *Nucleic acids research* 47.D1 (2019), pp. D506–D515.

[77] "Protein Data Bank: the single global archive for 3D macromolecular structure data". In: *Nucleic acids research* 47.D1 (2019), pp. D520–D528.

[78] Andrew W Senior et al. "Improved protein structure prediction using potentials from deep learning". In: *Nature* 577.7792 (2020), pp. 706–710.

[79] Jinbo Xu. "Distance-based protein folding powered by deep learning". In: *Proceedings of the National Academy of Sciences* 116.34 (2019), pp. 16856–16865.

[80] Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. "Improved protein structure prediction using predicted interresidue orientations". In: *Proceedings of the National Academy of Sciences* 117.3 (2020), pp. 1496–1503.

[81] 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". In: *BMC bioinformatics* 20.1 (2019), pp. 1–15.

[82] 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". In: *Nature communications* 10.1 (2019), pp. 1–13.

[83] 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". In: *ArXiv Preprint* (2018).

[84] 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.

[85] Risi Kondor. "N-body Networks: a Covariant Hierarchical Neural Network Architecture for Learning Atomic Potentials". In: *ArXiv preprint* (2018).

[86] 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.

[87] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations (ICLR)* (2018).

[88] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. "Non-local Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[89] Peter Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv* (2018). URL: `https://arxiv.org/pdf/1806.01261.pdf`.

[90] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* (2020).

[91] Taco Cohen and Max Welling. "Group Equivariant Convolutional Networks". In: *Proceedings of the International Conference on Machine Learning, ICML*. 2016.

[92] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew 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).

[93] 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.

[94] Erik J Bekkers. "B-Spline CNNs on Lie groups". In: *International Conference on Learning Representations*. 2019.

[95] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[96] 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.

[97] 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.

[98] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. "Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions". In: *International Conference on Learning Representations (ICLR)* (2018).

[99] 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". In: *arXiv preprint* (2020).

[100] Jiancheng Yang, Qiang Zhang, and Bingbing Ni. "Modeling Point Clouds With Self-Attention and Gumbel Subset Sampling". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[101] Saining Xie, Sainan Liu, and Zeyu Chen Zhuowen Tu. "Attentional ShapeContextNet for Point Cloud Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[102] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. "A Structured Self-attentive Sentence Embedding". In: *International Conference on Learning Representations (ICLR)* (2017).

[103] Yedid Hoshen. "VAIN: Attentional Multi-agent Predictive Modeling". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[104] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations". In: *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)* (2018).

[105] A. C. O'Connor, M. P. Gallaher, K. Clark-Sutton, D. Lapidus, and Z. T. Oliver. "Economic Benefits of the Global Positioning System (GPS)". In: *National Institute of Standards and Technology Report* (2019).

[106] P. M. Kintner, B. M. Ledvina, and E. R. de Paula. "GPS and ionospheric scintillations". In: *Space Weather* 5 (2007).

[107] S. Priyadarshi. "A Review of Ionospheric Scintillation Models". In: *Surveys in Geophysics* 36 (2015).

[108] Ryan M. McGranaghan, Anthony J. Mannucci, Brian Wilson, Chris A Mattmann, and Richard Chadwick. "New Capabilities for Prediction of High-Latitude Ionospheric Scintillation: A Novel Approach With Machine Learning". In: *Space Weather* 16 (2018).

[109] J. Aarons, B. Lin, M. Mendillo, K. Liou, and M. Codrescu. "Global Positioning System phase fluctuations and ultraviolet images from the Polar satellite". In: *Journal of Geophysical Research: Space Physics* 105.A3 (2000), pp. 5201–5213.

[110] Christer van der Meeren, Kjellmar Oksavik, Dag A. Lorentzen, Michael T. Rietveld, and Lasse B. N. Clausen. "Severe and localized GNSS scintillation at the poleward edge of the nightside auroral oval during intense substorm aurora". In: *Journal of Geophysical Research: Space Physics* 120 (2015).

[111] Yaqi Jin, Jøran I Moen, and Wojciech J Miloch. "GPS scintillation effects associated with polar cap patches and substorm auroral activity: Direct comparison". In: *Journal of Space Weather and Space Climate* 4 (2014), A23.

[112] Kjellmar Oksavik, Christer van der Meeren, Dag Arne Lorentzen, LJ Baddeley, and J Moen. "Scintillation and loss of signal lock from poleward moving auroral forms in the cusp ionosphere". In: *Journal of Geophysical Research: Space Physics* 120.10 (2015), pp. 9161–9175.

[113] Christer van der Meeren, Kjellmar Oksavik, Dag A Lorentzen, Michael T Rietveld, and Lasse BN Clausen. "Severe and localized GNSS scintillation at the poleward edge of the nightside auroral oval during intense substorm aurora". In: *Journal of Geophysical Research: Space Physics* 120.12 (2015), pp. 10–607.

[114] Yaqi Jin, Jøran I Moen, Wojciech J Miloch, Lasse BN Clausen, and Kjellmar Oksavik. "Statistical study of the GNSS phase scintillation associated with two types of auroral blobs". In: *Journal of Geophysical Research: Space Physics* 121.5 (2016), pp. 4679–4697.

[115] SB Mende et al. "TheTHEMIS array of ground-based observatories for the study of auroral substorms". In: *The THEMIS Mission*. Springer, 2009, pp. 357–387.

[116] Lasse BN Clausen and Hannes Nickisch. "Automatic classification of auroral images from the Oslo Auroral THEMIS (OATH) data set using machine learning". In: *Journal of Geophysical Research: Space Physics* 123.7 (2018), pp. 5640–5647.

[117] Qiuju Yang, Dacheng Tao, Desheng Han, and Jimin Liang. "Extracting auroral key local structures from all-sky auroral images by artificial intelligence technique". In: *Journal of Geophysical Research: Space Physics* (2019).

[118] Xi Yang, Nannan Wang, Bin Song, and Xinbo Gao. "BoSR: A CNN-based aurora image retrieval method". In: *Neural Networks* 116 (2019), pp. 188–197.

[119] G. R. T. de Lima, S. Stephany, E. R. de Paula, I. S. Batista, and M. A. Abdu. "Prediction of the level of ionospheric scintillation at equatorial latitudes in Brazil using a neural network". In: *Space Weather* 13 (2015).

[120] Yu Jiao, Yu T. Morton, Steven Taylor, and Wouter Pelgrum. "Characterization of high-latitude ionospheric scintillation of GPS signals". In: *Radio Science* 48 (2013).

[121] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[122] Paul Zhang. "Multiple imputation: theory and method". In: *International Statistical Review/Revue Internationale de Statistique* (2003), pp. 581–592.

[123] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. "Recurrent Neural Networks for Multivariate Time Series with Missing Values". In: *Scientific Reports* 8 (2018).

[124] Otto Hyvärinen. "A Probabilistic Derivation of Heidke Skill Score". In: *Weather and Forecasting* 29 (2014).

[125] Septentrio. *PolaRxS Application Manual*. 2015.

[126] Benjamin Hou and Athanasios Vlontzos. *ResNetAE*. 2019. URL: `https://github.com/farrell236/ResNetAE`.

[127] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[128] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[129] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[130] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

[131] Zoubin Ghahramani. "Probabilistic machine learning and artificial intelligence". In: *Nature* 521.7553 (2015), pp. 452–459.

[132] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. "An introduction to probabilistic programming". In: *arXiv preprint arXiv:1809.10756* (2018).

[133] Luigi Acerbi. "Variational Bayesian Monte Carlo". In: *32nd Conference on Neural Information Processing Systems*. 2018, pp. 8223–8233.

[134] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. "Graph convolutional networks with markov random field reasoning for social spammer detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 01. 2020, pp. 1054–1061.

[135] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. "Interaction Networks for Learning about Objects, Relations and Physics". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4502–4510.

[136] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478* (2021).

[137] Oliver T. Unke, Mihail Bogojeski, Michael Gastegger, Mario Geiger, Tess Smidt, and Klaus-Robert Müller. *SE(3)-equivariant prediction of molecular wavefunctions and electronic densities*. 2021. arXiv: `2106.02347 [physics.chem-ph]`.

[138] Yin Li, Yueying Ni, Rupert AC Croft, Tiziana Di Matteo, Simeon Bird, and Yu Feng. "AI-assisted superresolution cosmological simulations". In: *Proceedings of the National Academy of Sciences* 118.19 (2021).

[139] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.