Universal Approximation of Functions on Sets

Edward Wagstaff   ED@ROBOTS.OX.AC.UK
Fabian B. Fuchs   FABIAN@ROBOTS.OX.AC.UK
Martin Engelcke   MARTIN@ROBOTS.OX.AC.UK
Michael A. Osborne   MOSB@ROBOTS.OX.AC.UK
Ingmar Posner   INGMAR@ROBOTS.OX.AC.UK
Department of Engineering Science
University of Oxford
Oxford, UK

Editor: Ruslan Salakhutdinov

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.

Keywords: Deep learning, sets, permutation invariance, equivariance, universal function approximation, Janossy pooling, self-attention

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 in a stateful fashion. 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. (2017) 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. Similar models are also explored in Qi et al. (2017a) and Yarotsky (2022).

While processing each input individually and pooling globally is easy to implement and parallelise, it hinders relational reasoning (Santoro et al., 2017; Battaglia et al., 2018). 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 inbuilt 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 (Vaswani et al., 2017), it has also been applied to sets (Lee et al., 2019), and is popular in the literature on graph learning (Veličković et al., 2018).

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* (Murphy et al., 2019). 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 dimension of a Deep Sets model's latent space affects its expressive capacity. This extends the partial characterisation provided in Zaheer et al. (2017).

First, we follow Zaheer et al. (2017) in considering the conditions under which Deep Sets is capable of *exactly representing* any continuous target function (that is, we consider *universal representation*). We show that the sufficient condition on the model's latent space dimension given in Zaheer et al. (2017) 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. Second, 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 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 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 3 proves a necessary and sufficient condition for *representing* arbitrary continuous functions on sets with the Deep Sets architecture;

- Section 4 furthers our analysis of the Deep Sets architecture by proving a necessary and sufficient condition for *approximating* arbitrary continuous functions on sets;

- Section 5 discusses other models for deep learning on sets;

- Section 6 discusses universal approximation criteria for architectures beyond Deep Sets, and identifies some open questions of interest.

Section 3 of this paper is a revised and abridged version of work previously presented in Wagstaff et al. (2019). This paper provides a more extensive and comprehensive discussion of the subject, and extends the work in Wagstaff et al. (2019) by providing additional results for architectures other than Deep Sets and a theoretical characterisation of universal function approximation.

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 operating on 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 (2021) 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 maintaining maximum expressivity (in the sense of being able to model a broad class of functions) while also ensuring permutation invariance. Murphy et al. (2019) 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 takes place as the model's *latent space*.

Mathematically, the procedure outlined above can be written as

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

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
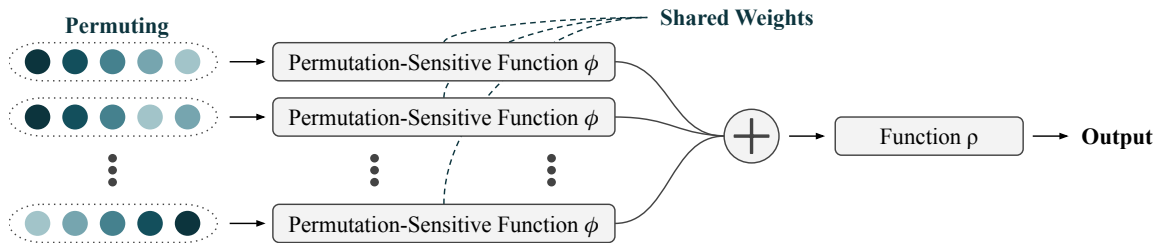
Figure 1: The Janossy pooling paradigm applies the same permutation-sensitive network to each possible permutation of the input set. When using a permutation-invariant pooling operation, such as averaging or summation, information about the ordering is eliminated. A second neural network may be used to predict the final output. This paradigm guarantees permutation invariance of the output without imposing restrictions on any of the neural network components.

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 1. This figure also illustrates that a second function may optionally be used to post-process the output of the aggregation operation:

$$f(\mathbf{x}) = \rho\big(\widehat{f}(\mathbf{x})\big).$$ (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 before ρ is reached in Figure 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 (1). The computational complexity scales at least linearly in the cardinality of S_M, which is $M!$. To remedy this, Murphy et al. (2019) discuss several options for reducing the computational complexity: (i) sorting; (ii) sampling; (iii) restricting permutation to k-tuples.

Rather than considering all permutations, sorting considers only 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 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. (2019) show that this works reasonably well empirically. Nevertheless, most models that are used in practice (e.g., Zaheer et al., 2017; Qi et al., 2017b; Lee et al., 2019) 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.
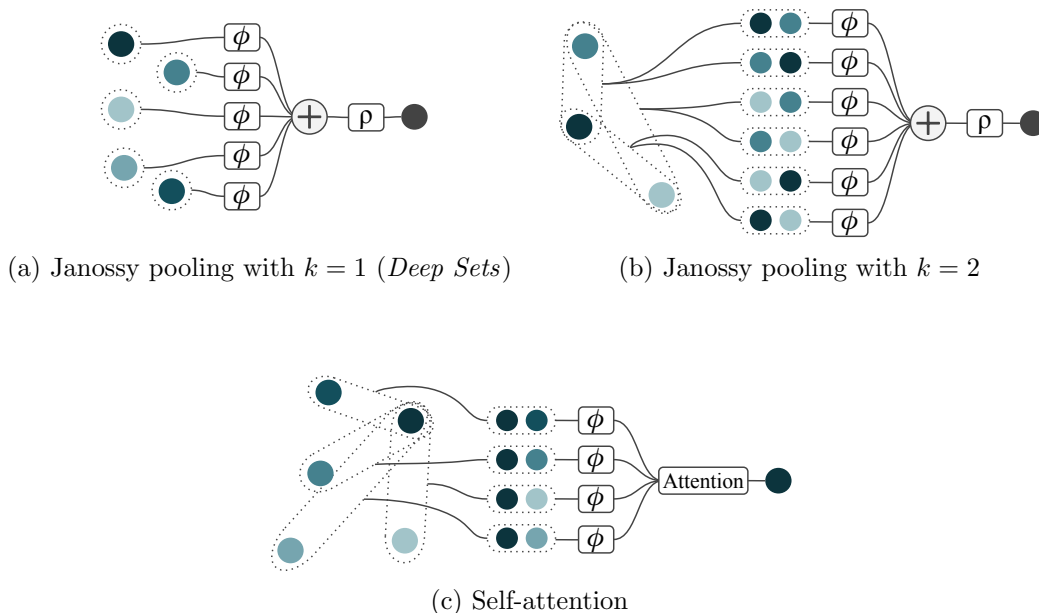
(a) Janossy pooling with $k = 1$ (*Deep Sets*)

(b) Janossy pooling with $k = 2$

(c) Self-attention

Figure 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 a softmax, in the case of attention). Self-attention, a variant of Janossy pooling with $k = 2$, focuses on one node at a time, computing an output for this specific node—in Figure 2c, this is represented by the darkest node. It is often employed for all nodes in parallel in a permutation-equivariant manner, mapping sets of points to sets of points (Lee et al., 2019).

2.1 Limiting the Number of Elements in Permutations

Equation (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 2. Mathematically, letting $\mathbf{x}_{\{k\}}$ denote the set of all k-tuples from $\mathbf{x}$, we have

$$\widehat{f}(\mathbf{x}) = \frac{1}{P(M, k)} \sum_{\mathbf{t} \in \mathbf{x}_{\{k\}}} \phi(\mathbf{t}) \ , \text{ where } P(M, k) = \frac{M!}{(M - k)!}. \tag{3}$$

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:

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

---

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.

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 also allows for more explicit relational reasoning—we will discuss this point in more detail in the following subsection.

2.1.1 EXPRESSIVITY AND INTERACTIONS

The terms *relational reasoning* and *interactions* are often encountered in the deep learning literature (Battaglia et al., 2016; Santoro et al., 2017; Fuchs et al., 2019). 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 appearance of some combination of elements *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. (2017) 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. (2019) 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 2c) compare two elements of the set at a time, typically by performing a scalar product (Vaswani et al., 2017; Lee et al., 2019). The results of the scalar products are used as attention weights for 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.

---

2. 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.

2.1.2 Deep Sets

The model obtained by setting $k = 1$ is in fact a popular and well-known special case of Janossy pooling—Deep Sets (Zaheer et al., 2017). 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 (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b). 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. Bueno and Hylton (2021) provide a unified mathematical analysis of the expressive power of both Deep Sets and PointNet. However, the conditions under which universality holds are not fully characterised, either for Deep Sets specifically or for Janossy pooling more broadly. Zaheer et al. (2017) and Han et al. (2019) 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.

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. (2017). 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 (for instance, the class of all permutation-invariant functions). The original analysis from Zaheer et al. (2017) demonstrates the universality of Deep Sets by considering universal representation, and in this section we present a direct continuation and refinement of this analysis. It must be noted that function representation is a stronger property than is required in practice, and we will return to analyse the weaker property of function *approximation* in Section 4. Proofs of all novel results from this section are provided in Appendix B.

3.1 Preliminaries

We begin by introducing the necessary definitions and notation, before outlining our contribution and its relationship to the original results from Zaheer et al. (2017). Importantly, we will establish the concept of *sum-decomposition*, which is a concise mathematical description of the Deep Sets architecture.

3.1.1 Definitions and Notation

Notation 1 *We denote sets of inputs in boldface with subscript-indexed elements. For example, the set $\mathbf{x}$ has elements $x_1, \ldots, x_M$.*

Notation 2 *Throughout, the variable M refers to the number of elements in the input sets under consideration.*

Notation 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 the set $\{1, \ldots, M\}$ to itself.*

Definition 4 *A function $f(\mathbf{x})$ is* permutation invariant *if $f(x_1, \ldots, x_M) = f(x_{\pi(1)}, \ldots, x_{\pi(M)})$ for all $\pi \in S_M$.*

Definition 5 *We say that a function f is* sum-decomposable *if there are functions ρ and ϕ such that*

$$f(\mathbf{x}) = \rho\Big(\sum_i \phi(x_i)\Big). \tag{4}$$

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

Note that sum-decomposition is equivalent to Janossy pooling with $k = 1$—that is, it is equivalent to Deep Sets. Comparing Equation (4) with Equations (2) and (3), these expressions differ only in dividing ϕ by an integer, and can be made equal simply by rescaling ϕ.

Notation 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) 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 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). We refer to Z as the* latent space *of the sum-decomposition (ρ, ϕ).*

Definition 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 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 10 *Denote the set of all subsets of a space X, known as the* power set *of X, by 2^X.*

Notation 11 *Denote the n-cube $[-1, 1]^n$ by I_n.*

3.1.2 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 dimension of the latent space Z. Specifically, we show that Z must be at least M-dimensional, where M is the number of elements in the input sets. We refine the analysis from Zaheer et al. (2017), 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 on the dimension of Z. Because this section directly builds on the analysis given in Zaheer et al. (2017), we first give a brief reproduction of the statements and proofs of two key theorems from that work.

3.1.3 BACKGROUND THEOREMS

Zaheer et al. (2017) consider two cases. First, where $\mathbf{x}$ is a subset of, or is drawn from, a *countably infinite* universe $\mathfrak{U}$. Second, the case where $\mathfrak{U}$ is *uncountably infinite*.

Theorem 12 (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

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

so 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 $r := \Phi(\mathbf{x})$. The number r 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 13 (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 13, which brings the terminology more in line with Deep Sets, is as follows:

Theorem 13 (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. (2017) of Theorem 13 is more involved than that of Theorem 12. 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. 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.
4. In the original proof, Φ is denoted E.

3. Define $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$ by $\rho := f \circ \Phi^{-1}$.

4. Define $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ by $\phi_q(x) := x^q, \; q = 0, \ldots, M$.

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 3.2, we argue that the guarantee of sum-decomposability via $\mathbb{R}$ given by Theorem 12 cannot hold in practice. This is because the necessary mappings are highly discontinuous and cannot be computed.

- In Section 3.3, we prove that the guarantee of sum-decomposability via $\mathbb{R}^{M+1}$ given by Theorem 13 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 Appendix B.

3.2 The Importance of Continuity

Theorem 12 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 12, with the consequence that Theorem 12 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 $\iota := c^{-1}(2^{2^{16}})$, and suppose that, given a set of images $\mathbf{x}$, our target function f needs to compute whether $\iota \in \mathbf{x}$. Under the encoding/decoding procedure from the proof of Theorem 12, the decoder function ρ is given as input the real number $r := \Phi(\mathbf{x})$. Recalling that the construction has $\phi(x) := 4^{-c(x)}$, 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

---

5. One of our proofs depends on the axiom of choice, bringing us firmly beyond the realm of computability!
6. That is, images which are not just random noise, but have meaningful visual content.

insensitive to small variations in the input. In the example above, a successful decoder ρ must be 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 properties of machine learning models, including the universal approximation theorem for neural networks (Cybenko, 1989), the universality of Gaussian processes with certain kernels (Rasmussen and Williams, 2006), the literature on approximating functions on sets (Han et al., 2019; Segol and Lipman, 2020), and indeed in function approximation results outside machine learning (such as the well-known Stone-Weierstrass theorem, Stone, 1948). 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 of Theorem 13, and in the next section we show that the conclusion of Theorem 13 is essentially the strongest result that can be achieved under these assumptions.

3.3 Function Representation With Continuous Mappings

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

Theorem 14 *Let $M, N \in \mathbb{N}$, with $M > N$. Then there exist continuous permutation-invariant 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. (2017a)—details are provided in Appendix B.6.

In terms of concrete recommendations for practitioners, Theorem 14 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 back to our depiction of Deep Sets in Figure 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 14 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, where we prove that this requirement on the latent space dimension is also present for function *approximation* with Deep Sets. We discuss the practical implications of Theorem 14 in more detail in Section 3.3.2.

In addition to showing that an M-dimensional latent space is necessary, we adapt the proof of Theorem 13 to strengthen the result in two ways. Proofs of these two results can be

---

7. Essentially all popular activation functions are continuous, including ReLU, ELU, sigmoid, and tanh.

found in Appendix B. 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 15 *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 16 (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 $\mathbb{R}^M$.*

3.3.1 PROOF OF THEOREM 14

Our proof of Theorem 14 relies on showing that the permutation-invariant function `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 17 *Let $M, N \in \mathbb{N}$, and suppose $\phi_M : [\text{-}1, 1] \to \mathbb{R}^N$, $\rho_M : \mathbb{R}^N \to \mathbb{R}$ are functions such that*

$$\mathtt{max}(\mathbf{x}) = \rho_M\Big(\sum_{i=1}^M \phi_M(x_i)\Big) \tag{5}$$

for all $\mathbf{x} \in I_M$ (where $I_M := [\text{-}1, 1]^M$). That is, suppose (ϕ_M, ρ_M) is a sum-decomposition of `max` via $\mathbb{R}^N$. Then $\Phi_M(\mathbf{x}) := \sum_{i=1}^M \phi_M(x_i)$ is injective up to permutation.

Proof Although we consider only injectivity up to permutation, for brevity we will simply refer to "injectivity" in this proof. Similarly, we may say that $\mathbf{x}, \mathbf{y} \in I_M$ are "equal" when they are equal up to permutation.

We proceed by induction on M. The base case $M = 1$ is trivial, since `max` is injective when $M = 1$. Now let $M \in \mathbb{N}$, with ϕ_M and ρ_M satisfying Equation (5). Consider the subset of I_M defined by holding the minimum element fixed to -1. We denote this subset X_*. We first claim that Φ_M is injective on X_*.

To see this, set

$$\phi_{M-1} := \phi_M, \quad \rho_{M-1}(\mathbf{x}) := \rho_M\big(\mathbf{x} + \phi_M(\text{-}1)\big).$$

This gives a sum-decomposition (ρ_{M-1}, ϕ_{M-1}) of `max` on I_{M-1}, so by induction, Φ_{M-1} is injective. But on X_*, Φ_M is just Φ_{M-1} plus the constant $\phi_M(\text{-}1)$, so Φ_M is injective on X_*.

Now let $a \in [\text{-}1, 1]$, and consider the set X_a formed by holding the *maximum* element of $\mathbf{x}$ fixed to a. We will show that Φ_M is injective on X_a. Suppose $\mathbf{x}, \mathbf{y} \in X_a$ with

---

8. Note that we must take some care with the notion of continuity here—see Appendix A.2.

$\Phi_M(\mathbf{x}) = \Phi_M(\mathbf{y})$. Now consider the sets $\mathbf{x}', \mathbf{y}'$ obtained by replacing a with -1. Then $\mathbf{x}', \mathbf{y}' \in X_*$, and

$$\begin{aligned}
\Phi_M(\mathbf{x}') &= \Phi_M(\mathbf{x}) - \phi_M(a) + \phi_M(\text{-}1) \\
&= \Phi_M(\mathbf{y}) - \phi_M(a) + \phi_M(\text{-}1) \\
&= \Phi_M(\mathbf{y}').
\end{aligned}$$

Since Φ_M is injective on X_*, we therefore have $\mathbf{x}' = \mathbf{y}'$, and therefore $\mathbf{x} = \mathbf{y}$. That is, Φ_M is injective on X_a.

Now note that any $\mathbf{x} \in I_M$ belongs to X_a for some a. Specifically, $\mathbf{x} \in X_{\mathtt{max}(\mathbf{x})}$. So suppose $\mathbf{x}, \mathbf{y} \in I_M$ with $\mathbf{x} \neq \mathbf{y}$. There are two cases to consider. First, we may have $\mathbf{x}, \mathbf{y} \in X_a$ for some a, in which case $\Phi_M(\mathbf{x}) \neq \Phi_M(\mathbf{y})$ by injectivity of Φ_M on X_a. Second, we may have $\mathbf{x} \in X_a$, $\mathbf{y} \in X_b$ for some $a \neq b$, in which case $\mathtt{max}(\mathbf{x}) \neq \mathtt{max}(\mathbf{y})$. Then $\Phi_M(\mathbf{x}) \neq \Phi_M(\mathbf{y})$ by Equation (5). In either case, $\Phi_M(\mathbf{x}) \neq \Phi_M(\mathbf{y})$, so Φ_M is injective. ∎

Equipped with this lemma, we can now prove Theorem 14.

Proof Let $M, N \in \mathbb{N}$, with $M > N$, and let $\phi_M : [\text{-}1, 1] \to \mathbb{R}^N$ be continuous. Denote by Δ_M the set of all $\mathbf{x} \in I_M$ such that $x_1 \geq x_2 \geq \cdots \geq x_M$.

$\Phi_M : I_M \to \mathbb{R}^N$ is continuous, because it is a sum of continuous functions. Because $M > N$, it therefore cannot be injective on Δ_M—since Δ_M has non-empty interior, it cannot be continuously injected into a lower-dimensional Euclidean space. But by Lemma 17, any sum-decomposition of $\mathtt{max}$ has Φ_M injective up to permutation, which in particular means that any such Φ_M is injective on Δ_M. Since our Φ_M is *not* injective here, ϕ_M therefore cannot be used to sum-decompose $\mathtt{max}$. The only condition on ϕ_M was that it is continuous, so $\mathtt{max}$ therefore has no continuous sum-decomposition via $\mathbb{R}^N$. ∎

3.3.2 Implications and Limitations

In light of Theorems 14 and 15, 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 14 and 15 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 14 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 15 shows that an M-dimensional latent space suffices to model any target function, we also know from Theorem 14 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 14 is that, in following the mathematical framework developed in Zaheer et al. (2017), 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 14 does indeed hold for universal approximation, and M dimensions are still required.

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 Zaheer et al. (2017) 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:

$$\phi(x) := e^{ax}, \qquad\qquad \rho(x) := \frac{\log x}{a}. \qquad (6)$$

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

$$\widehat{f}_a(\mathbf{x}) = \frac{1}{a} \log \left(\sum_{x \in \mathbf{x}} e^{ax} \right). \qquad (7)$$

For an input set $\mathbf{x}$ of size M, it is easily shown that

$$\mathtt{max}(\mathbf{x}) \leq \widehat{f}_a(\mathbf{x}) \leq \mathtt{max}(\mathbf{x}) + \frac{\log M}{a}.$$

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.

4.1 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 18 *Let $\epsilon > 0$. (ϕ, ρ) is a* within-ϵ sum-decomposition *of f if $\big|f(\mathbf{u}) - \rho\big(\Phi(\mathbf{u})\big)\big| < \epsilon$ for every $\mathbf{u} \in U$.*

For example, if we let $a = \frac{\log M}{\epsilon}$, then the sum-decomposition defined by Equation (6) is a within-ϵ continuous sum-decomposition of `max` via $\mathbb{R}$.

Definition 19 *A sequence $(\phi, \rho)_k := \big\{(\phi_k, \rho_k) : k \in \mathbb{N}\big\}$ 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 (6) is a continuous approximate sum-decomposition of `max` via $\mathbb{R}$.

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

---

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 (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989). This compactness requirement is a fundamental constraint which is also necessary for other function approximation results, for example the Stone-Weierstrass theorem (Stone, 1948).

Theorem 20 *Let $M, N \in \mathbb{N}$ with $M > N$, and recall that $I_M = [-1, 1]^M \subset \mathbb{R}^M$. Then there exists a continuous permutation-invariant function $f : I_M \to \mathbb{R}$ which has no continuous approximate sum-decomposition via $\mathbb{R}^N$.*

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 C.

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 $E := \frac{y_{\max} - y_{\min}}{2}$ 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 $U = I_M$, this f_* has a simple form:[11]

$$
\begin{aligned}
f_*(\mathbf{u}) &:= \mathbf{w}^\mathsf{T}\mathtt{sort}(\mathbf{u}) + b\,, \\
w_i &:= (\text{-}1)^{i+1}\,, \\
b &:= \begin{cases} \text{-}1 & M \text{ even}\,, \\ 0 & M \text{ odd}\,. \end{cases}
\end{aligned}
\tag{8}
$$

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 4.2 and provide some intuition for why it is hard to approximate in Section 4.3. The function f_* is trivial to represent using the sorting framework[12] mentioned in Section 2, 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 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 20. We omit some details, particularly in the proof of Lemma 24. Full details are included in Appendix C.2.

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

---

10. By the *worst-case error* of a function $\widehat{f}$ approximating a function f, we mean $\mathtt{max}_{\mathbf{u} \in U}|\widehat{f}(\mathbf{u}) - f(\mathbf{u})|$.
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 (8) without affecting our result, but reasoning about f_* is simpler if its range is fixed.
12. We discuss sorting in more detail in Section 5.1.

(a) The boundary of Δ_3.

(b) The image of Δ_3 under f_*.

(c) The image of $[-1, 1]$ under ϕ.

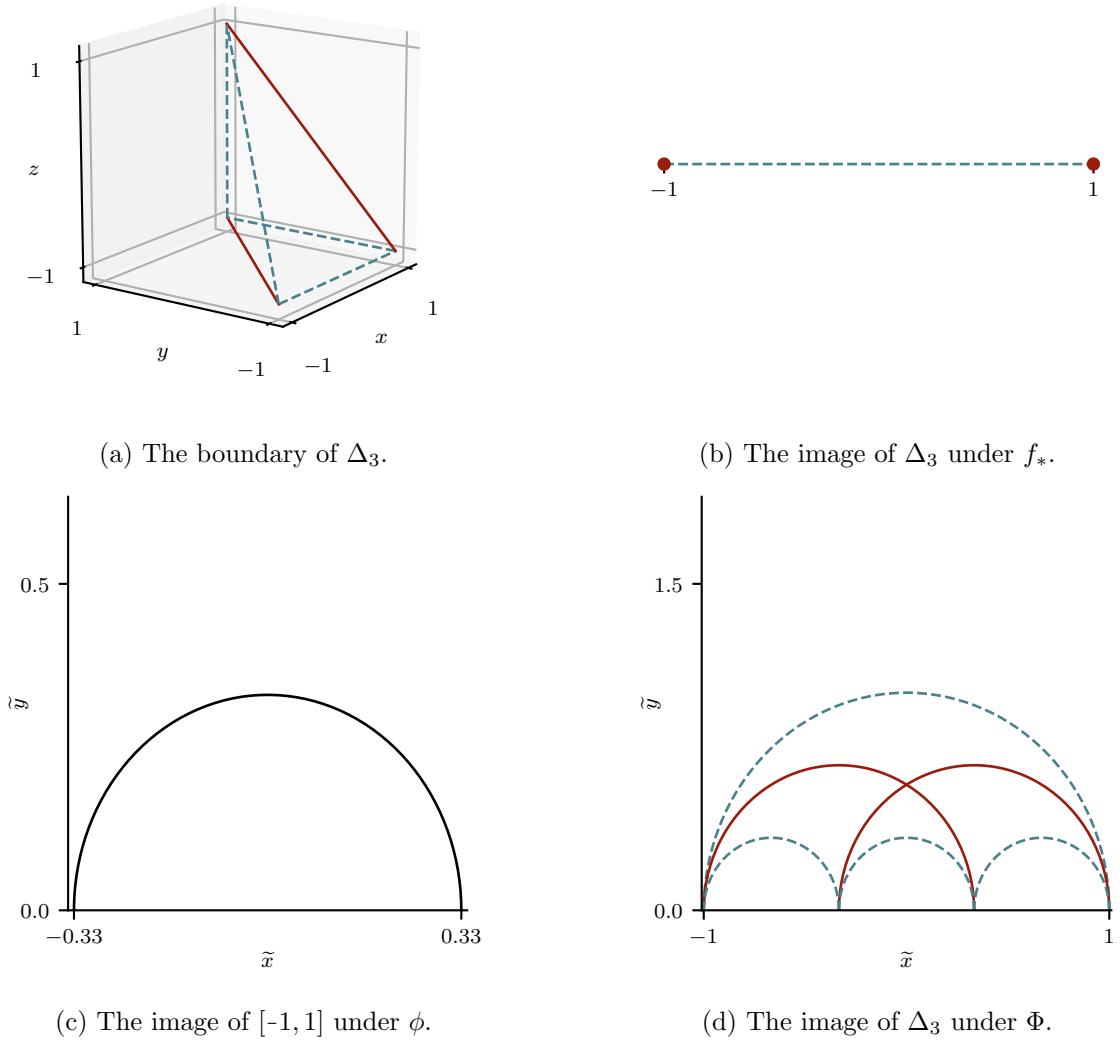(d) The image of Δ_3 under Φ.

Figure 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 here 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, as long as the dimension of the latent space is less than the number of inputs, a similar problem occurs in any number of dimensions. That is, there is always a collision under Φ which renders sum-decomposition of f_* impossible.

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 $f : I_3 \to \mathbb{R}$ 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,

$$\Delta_n := \{\mathbf{x} \in I_n : x_1 \geq x_2 \geq \ldots \geq x_n\}. \tag{9}$$

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 I_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 $\texttt{sort}$ in Equation (8), 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 plane onto a line under ρ. Figures 3a, 3c and 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 3b, then the sum-decomposition $\rho \circ \Phi$ must fail with worst-case error at least 1.

Figure 3 only shows that this intersection problem occurs for our example choice of ϕ and M, but we prove below that the general case also holds: for every M, there are two disjoint regions of the surface of Δ_M such that, for any continuous ϕ which maps into $\mathbb{R}^{M-1}$, these regions intersect under Φ. We denote the two regions X_M^{+1} and X_M^{-1}—the solid red regions in Figure 3a are X_3^{+1} and X_3^{-1}. Our f_* from Equation (8) 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:[13]

$$X_M^{+1} := \{\mathbf{x} \in \Delta_M : 1 = x_1 \geq x_2 = x_3 \geq \ldots \stackrel{=}{\geq} x_M \stackrel{\geq}{=} -1\}, \tag{10}$$

$$X_M^{-1} := \{\mathbf{x} \in \Delta_M : 1 \geq x_1 = x_2 \geq x_3 = \ldots \stackrel{\geq}{=} x_M \stackrel{=}{\geq} -1\}. \tag{11}$$

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 9) by changing either the odd- or the even-numbered $\geq$ signs into $=$ signs. It follows directly from the definition of f_* (Equation 8) that $f_*(X_M^{+1}) = 1$ and $f_*(X_M^{-1}) = -1$.

The above discussion can be summarised by the following claim:

---

13. The symbols $\stackrel{\geq}{=}$ and $\stackrel{=}{\geq}$ should be interpreted as representing the top sign when M is odd, and the bottom sign when M is even.

Claim 21 *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 $f_*(X_M^{-1}) = \text{-}1$.*

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

Theorem 20 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 20. The proof of point 2 has two parts:

1. **Lemma 22**: 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 24:** Show that Γ_N has a zero.

The subscript on Γ is N, rather than M, because there are a total of $N = M - 1$ degrees of freedom in $X_M^{+1} \times X_M^{-1}$—consider for example $M = 2$, which gives a total of one degree of freedom.[14] 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. 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 24.

Importantly, the following statement of Lemma 22 requires that $\phi(\text{-}1) = \mathbf{0}$. From now on we will assume this without loss of generality, since otherwise we can reason identically about $\widetilde{\phi}(x) := \phi(x) - \phi(\text{-}1)$, which does satisfy $\widetilde{\phi}(\text{-}1) = \mathbf{0}$.

Lemma 22 *Let $\phi : [\text{-}1, 1] \to \mathbb{R}^N$, with $\phi(\text{-}1) = \mathbf{0}$. Define $\Gamma_N : \Delta_N \to \mathbb{R}^N$ by*

$$\Gamma_N(\mathbf{x}) := \sum_{i=1}^{N}(\text{-}1)^i \phi(x_i) + \frac{\phi(1)}{2} \,. \tag{12}$$

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 $\mathbf{x}^- \in X_M^{-1}$ by

$$\begin{aligned} x_i^+ &:= z_i \quad \text{for even } i \,, \\ x_i^- &:= z_i \quad \text{for odd } i \,. \end{aligned} \tag{13}$$

---

14. Equations (10) and (11) imply that X_2^{+1} consists of the single point $(1, \text{-}1)$, contributing zero degrees of freedom, and X_2^{-1} consists of the line $\{(x, x) : x \in [\text{-}1, 1]\}$, contributing one degree of freedom.

The remaining coordinates of $\mathbf{x}^+$ and $\mathbf{x}^-$ are fixed by the equalities in Equations (10) and (11). Note in particular that Equation (10) implies that $x_1^+ = 1$. Taking these constraints together with the condition that $\phi(\text{-}1) = \mathbf{0}$, we obtain

$$\Phi(\mathbf{x}^+) = \phi(1) + 2 \sum_{i \text{ even}} \phi(x_i^+) \,,$$

$$\Phi(\mathbf{x}^-) = 2 \sum_{i \text{ odd}} \phi(x_i^-) \,. \tag{14}$$

Plugging $\mathbf{z}$ into Equation (12) and pulling the odd terms of the sum to the left hand side (and recalling that $\Gamma_N(\mathbf{z}) = \mathbf{0}$), we obtain

$$\sum_{i \text{ odd}} \phi(z_i) = \sum_{i \text{ even}} \phi(z_i) + \frac{\phi(1)}{2} \,. \tag{15}$$

Now we apply Equation (13) to obtain

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

This equates the right hand sides of the two lines of Equation (14), 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 (12), 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 itself a simplex[15] of dimension $N-1$, defined by changing one of the inequalities in Equation (9) into an equality—that is, by applying the constraint $x_i = x_{i+1}$ for some i. 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 (12). 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:

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

---

15. At this stage of the argument, it is helpful to have a way of visualising high-dimensional simplices and their faces. We discuss such a visualisation in Appendix A.4.

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 $(1, \frac{2}{3}, 0, -\frac{1}{2}) \in \Delta_4^{(0)}$:

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

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

$$\Gamma_4\left((1, \tfrac{2}{3}, 0, -\tfrac{1}{2})\right) = -\phi(1) + \phi(\tfrac{2}{3}) - \phi(0) + \phi(-\tfrac{1}{2}) + \frac{\phi(1)}{2},$$

$$\Gamma_4\left((\tfrac{2}{3}, 0, -\tfrac{1}{2}, -1)\right) = \qquad -\phi(\tfrac{2}{3}) + \phi(0) - \phi(-\tfrac{1}{2}) + \phi(-1) + \frac{\phi(1)}{2}.$$

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

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

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

$$\Gamma_N\big(\alpha(\mathbf{x})\big) = -\Gamma_N\big(\mathbf{x}\big). \tag{16}$$

We can exploit this symmetry to show that Γ_N has a zero. To do this, we will need the Borsuk-Ulam theorem (Borsuk, 1933—see Lloyd, 1978 for a detailed English-language treatment). One statement of the theorem is as follows.

Theorem 23 (Borsuk-Ulam) *Let ∂I_n be the surface of I_n—written mathematically, let $\partial I_n := \{\mathbf{x} \in I_n : x_i = \pm 1 \text{ for some } i\}$. Suppose $f : I_n \to \mathbb{R}^n$ is continuous, and for every $\mathbf{x} \in \partial I_n$ we have*

$$f(-\mathbf{x}) = -f(\mathbf{x}). \tag{17}$$

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

There is a clear similarity between Equations (16) and (17), but we cannot directly apply Borsuk-Ulam to Γ_N. We need to bridge the two equations, relating the left shift α on the left hand side of Equation (16) to the negation on the left hand side of Equation (17). 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 $\partial \Delta_N$:[16]

$$\nu_N(-\mathbf{x}) = \alpha(\nu_N(\mathbf{x})). \tag{18}$$

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

$$f(-\mathbf{x}) := \Gamma_N\big(\nu_N(-\mathbf{x})\big) = \Gamma_N\Big(\alpha\big(\nu_N(\mathbf{x})\big)\Big)$$
$$= -\Gamma_N\big(\nu_N(\mathbf{x})\big) = -f(\mathbf{x}).$$

Showing that a suitable ν_N exists will complete the proof, since Theorem 23 then immediately implies that Γ_N has a zero.

---

16. 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 24.
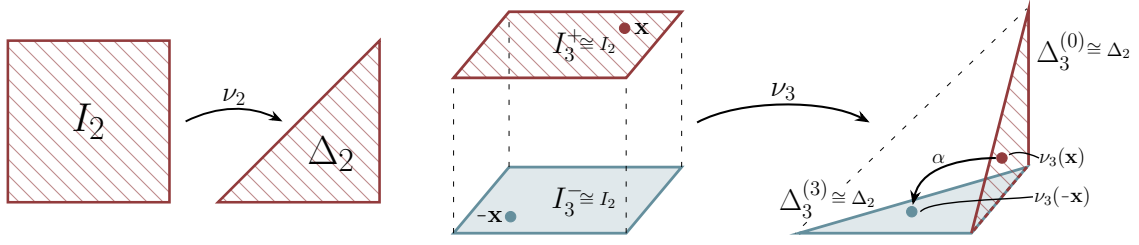
Figure 4: For the first part of the construction of ν_n, we use ν_{n-1} to define ν_n on the top and bottom faces of I_n. The $n = 3$ case is illustrated here. ν_n uses ν_{n-1} to send the top face of I_n to $\Delta_n^{(0)}$, and the bottom face to $\Delta_n^{(n)}$. We construct ν_n so that opposite points $\mathbf{x}$ and $\text{-}\mathbf{x}$ on the top and bottom faces go to points in Δ_n which are related by the left-shift function α—that is, $\nu_n(\text{-}\mathbf{x}) = \alpha(\nu_n(\mathbf{x}))$.
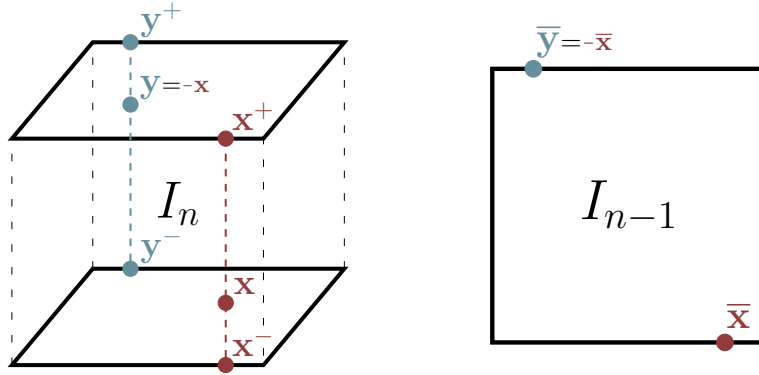

Figure 5: For the second part of the construction of ν_n, we note that any point $\mathbf{x}$ lies on a "vertical" line, i.e., an n-th-axis-aligned line, meeting the top face at $\mathbf{x}^+$ and the bottom face at $\mathbf{x}^-$. We complete the construction of ν_n by interpolating along these lines. 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}.

Lemma 24 *Let $n \in \mathbb{N}$. Let Γ_n be defined as in Equation (12). Then there exists a continuous function $\nu_n : I_n \to \Delta_n$ such that, for $\mathbf{x} \in \partial I_n$,*

$$\Gamma_n\big(\nu_n(\text{-}\mathbf{x})\big) = -\Gamma_n\big(\nu_n(\mathbf{x})\big). \tag{19}$$

We present a partial proof of Lemma 24 here, with full details available in Appendix C.2.

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 (18) for every $\mathbf{x}$ on ∂I_n. Nevertheless, we can still satisfy Equation (19) for every $\mathbf{x}$ on ∂I_n, and we reach this conclusion by showing that Equation (18) holds on part of ∂I_n.

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

$$
\begin{aligned}
I_n^+ &:= \{\mathbf{x} \in I_n : x_n = 1\}, && \text{the top face,}\\
I_n^- &:= \{\mathbf{x} \in I_n : x_n = \text{-}1\}, && \text{the bottom face.}
\end{aligned}
$$

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 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. 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 (18) 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 (18) 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:

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

So by definition, ν_n satisfies Equation (18) for $\mathbf{x} \in I_n^+$. This immediately implies Equation (19) on I_n^+ and, by symmetry, on I_n^-.

We have established that Equation (19) 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^-, where 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 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 (19) 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 (19), consider a pair of antipodal[17] points $\mathbf{x}$ and $\mathbf{y}$ on $\partial I_n \backslash (I_n^+ \cup I_n^-)$. As illustrated in Figure 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 (19) holds:

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

---

17. That is, such that $\mathbf{y} = \text{-}\mathbf{x}$.

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 (19) 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 $\Gamma_n\big(\nu_n(\mathbf{x}^+)\big)$:

$$\begin{aligned}
\Gamma_n\big(\nu_n(\mathbf{x}^+)\big) &= \Gamma_n\Big(\big(1, \nu_{n-1}(\overline{\mathbf{x}})\big)\Big) \\
&= -\phi(1) - \Big(\Gamma_{n-1}\big(\nu_{n-1}(\overline{\mathbf{x}})\big) - \frac{\phi(1)}{2}\Big) + \frac{\phi(1)}{2} \qquad (20) \\
&= -\Gamma_{n-1}\big(\nu_{n-1}(\overline{\mathbf{x}})\big).
\end{aligned}$$

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

$$\Gamma_n\big(\nu_n(\mathbf{x}^-)\big) = \Gamma_{n-1}\big(\nu_{n-1}(\text{-}\overline{\mathbf{x}})\big). \qquad (21)$$

But note that $\overline{\mathbf{x}} \in \partial I_{n-1}$, as illustrated in Figure 5. By induction, Equation (19) therefore applies:

$$\Gamma_{n-1}\big(\nu_{n-1}(\text{-}\overline{\mathbf{x}})\big) = -\Gamma_{n-1}\big(\nu_{n-1}(\overline{\mathbf{x}})\big).$$

Equations (20) and (21) therefore imply that

$$\Gamma_n\big(\nu_n(\mathbf{x}^-)\big) = \Gamma_n\big(\nu_n(\mathbf{x}^+)\big). \qquad (22)$$

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 C.2.
$\blacksquare$

With ν_N in hand, the proof of Theorem 20 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 I_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.
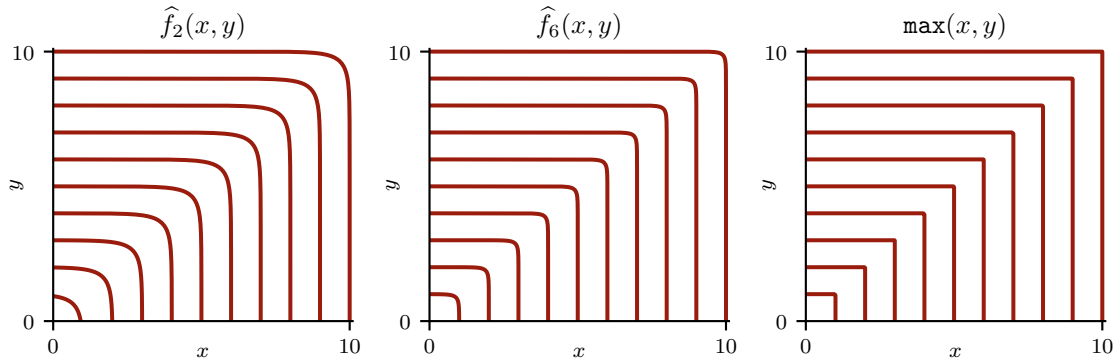
Figure 6: Contours of sum-decomposition approximations to `max` compared to the true `max` function. The sum-decompositions $\widehat{f}_2$ and $\widehat{f}_6$ are defined by Equation (7), with $a = 2, 6$ respectively. As a increases, the corners become sharper, but never perfectly sharp.

3. Because of the way Γ_N was constructed, this implies that there are two disjoint regions of the input space[18] I_M which collide under the application of Φ.

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 (8).

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 $\Phi(\mathbf{x}) = \sum_i \phi(x_i)$. 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 $\Phi(\mathbf{x}) = \Phi(\mathbf{y})$, then $\rho \circ \Phi(\mathbf{x}) = \rho \circ \Phi(\mathbf{y})$.

We plot the contours of `max` in Figure 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 contours of `max` have sharp corners on the line $y = x$, whereas the contours of the approximations have soft corners 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

---

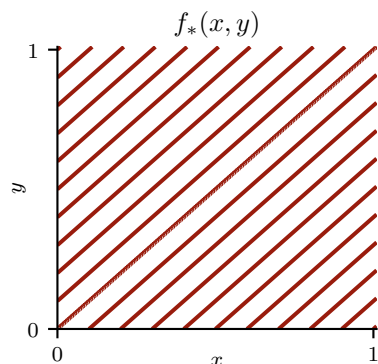18. Strictly speaking, there are two disjoint regions of the ordered input space Δ_M.

Figure 7: Contours of the function f_* as defined by Equation (8).

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[19] 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 6. `max` does not violate this condition so badly that we cannot approximate it—its contours do at least cross $y = x$. But as Figure 7 shows, the function f_* from our proof of Theorem 20 violates this condition very badly, having a contour along $y = x$, perpendicular to the contours of any sum-decomposition.

Again, what we show in Figures 6 and 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.

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 2, which are also discussed in Murphy et al. (2019): 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.

---

19. Neural networks can represent non-differentiable functions, but if we need to construct a ϕ which is non-differentiable in infinitely many places, things get more complicated.
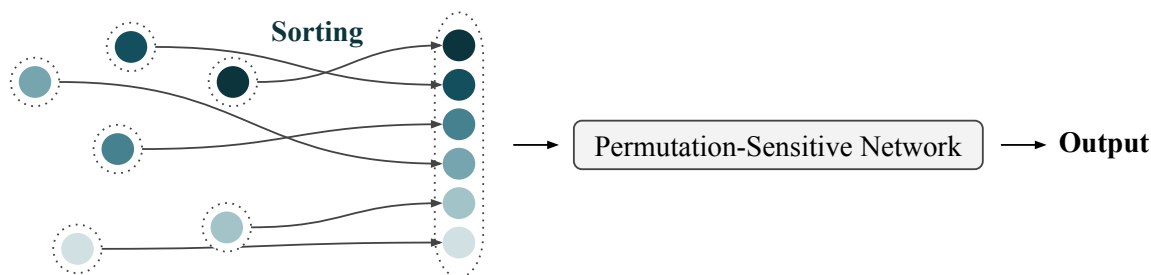
Figure 8: Achieving permutation invariance via sorting. Finding or learning a canonical way of sorting the elements of a set before feeding it into a permutation-sensitive network yields a permutation-invariant output, as long as the sorting does not depend on the initial order of the elements.

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 the elements are first sorted, ensuring that only this valid representation is seen by the model, 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 8, does have some drawbacks.

First, there is the 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. More straightforwardly, a function giving a score to each element can be learned, and the elements can then be sorted 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[20] is a piecewise-constant function, 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 at 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

---

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

ranking. A good ranking is whatever allows the decoder (that is, the permutation-sensitive network in Figure 8) 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 (Bengio et al., 2013; Yin et al., 2019). Recently, a cheap differentiable sorting operation with a computational cost of $O(n \log n)$ was also proposed by Blondel et al. (2020). 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.

Liu et al. (2021) propose yet another approach: after ranking the inputs according to a learned score function, a 1-dimensional convolution is applied to the ranked values. However, in order to get gradients for the score function, Liu et al. (2021) 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.

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. (2019) 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. (2019) 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 the fidelity of the 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 (2019) approach approximate permutation invariance in a different way. Inspired by generative adversarial networks (Goodfellow et al., 2014), 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.

5.3 From Sets to Graphs

Although this paper is focused on sets, not graphs, there is an obvious link between these two topics, which we briefly discuss. Graphs typically consist of nodes (which form sets) and edges, where each edge connects two nodes. This can be extended to hypergraphs, which include information attached to triplets of nodes (Yadati et al., 2019; Wei et al., 2020). From this perspective, it becomes clear that a set can be seen as a graph without edges (Figure 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, hypergraph data can be represented with higher-order adjacency tensors. The concept of permutations can now easily be extended

Sets: Information is only assigned to individual nodes.

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

Hypergraphs: Information is assigned to hyperedges, e.g., triplets of nodes.
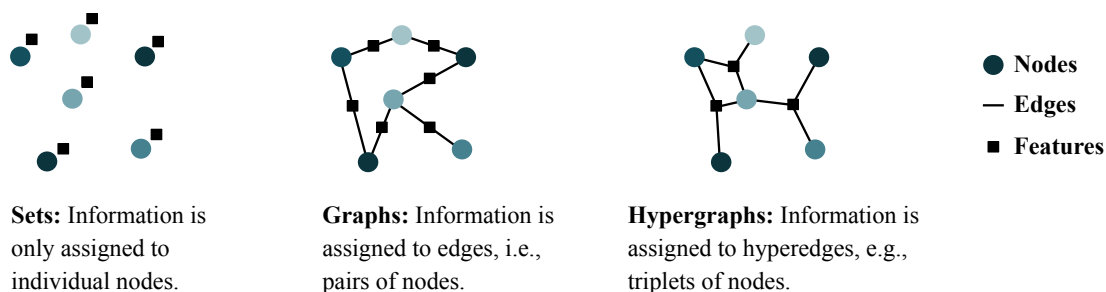
Figure 9: From sets to graphs and hypergraphs. Sets can be extended to graphs by introducing the notion of edges. A permutation of the input now corresponds to a permutation of the nodes together with the respective edges.

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 of 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. Permutation-*equivariant* models, by contrast, are applicable to graphs without requiring additional complications. With self-attention algorithms (Vaswani et al., 2017), for example, it is straightforward to include edge information (Veličković et al., 2018; Fuchs et al., 2020). Permutation-equivariant models have also been studied in the context of learning functions from sets to graphs (Serviansky et al., 2020).

Notably, Maron et al. (2019b) study the set of linear permutation-invariant or -equivariant functions 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 graph (or hypergraph). 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. (2019b) show that there is an orthogonal basis of at most 15 different $M^2 \times M^2$ matrices[21] 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 3. Our main result in Section 4 shows that this is the case even if we do not consider the whole function space, but only a dense subspace.

---

21. As noted in Finzi et al. (2021), for small M this basis may actually have fewer than 15 elements.

6. Universality Beyond Deep Sets

We have seen that Deep Sets is theoretically capable of representing all permutation-invariant functions if the dimension of the latent space is at least as high 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, theoretical universality does not guarantee that this is the best choice in practice. For example, from relational reasoning experiments (Fuchs et al., 2019), 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 known sufficiency and necessity criteria for universal function representation for different categories of set-based learning approaches.

6.1 Janossy Pooling

We consider k-ary Janossy pooling as depicted in Figure 1 and described in Section 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 25 *Let $M, N, k \in \mathbb{N}$, $\mathbf{x} \in \mathbb{R}^M$. Let $f : \mathbb{R}^M \to \mathbb{R}$. Write $\mathbf{x}_{\{k\}}$ 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*

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

6.1.1 CASE $k = 1$

This case is Deep Sets. In Sections 3 and 4, we provided an in-depth analysis of this case. In brief, Janossy pooling for $k = 1$ can approximate all permutation-invariant functions if the latent space dimension 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.

6.1.2 CASE $1 < k < M$

This inherits sufficiency from the case $k = 1$: the network processing tuples of length k can ignore all coordinates 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 tuples having the same first element. In Janossy pooling, this number is $(M-1)!/(M-k-1)!$. This gives the following statement:

Theorem 26 *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.*

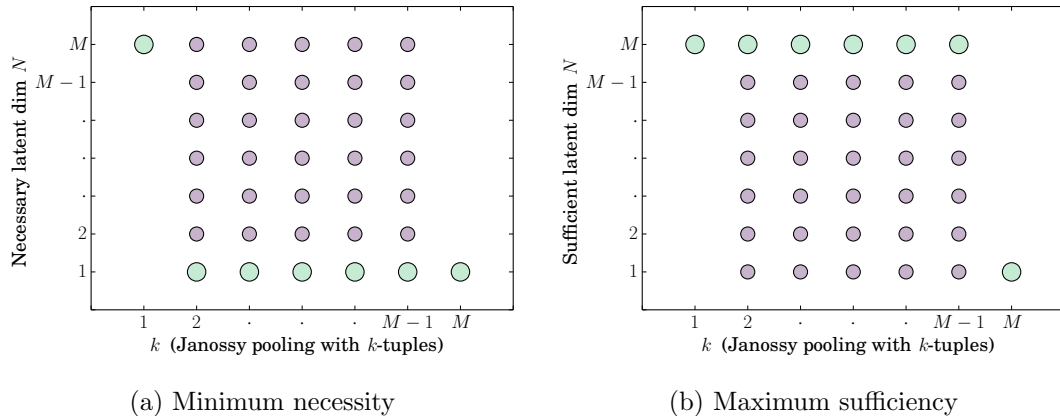(a) Minimum necessity (b) Maximum sufficiency

Figure 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 according to our proofs. The smaller purple circles show possible true necessary and sufficient conditions for universal function representation.

6.1.3 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 longer any requirement on the dimension of the latent space, and we can make the following statement:

Theorem 27 *Let $f : \mathbb{R}^M \to \mathbb{R}$ be continuous and permutation invariant. Then f has a continuous M-ary Janossy representation via $\mathbb{R}$.*

6.1.4 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 highest known necessary condition and the lowest known sufficient condition. Figure 10 visualises the conditions we derived and what the true, strongest statements might be. For $k = 2$, for example, 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 further investigation to future work.

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,[22] and indeed sufficient conditions on the latent space dimension are known for both invariant (Han et al., 2019) and equivariant (Segol and Lipman, 2020; Maron et al., 2019a) set functions. It is unknown, however, whether these known sufficiency bounds are also necessary—indeed, no nontrivial necessary bound is known, other than the 1-dimensional case presented in Section 4.

6.2 Sorting

In Section 5.1, we described a category of methods which achieve permutation invariance by 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 28 *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.

6.3 Approximate Permutation Invariance

In Section 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 (2019), 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 (2019) 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., used for the function f in the notation of Equation 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 $\mathbf{x}$,

---

22. Higher-dimensional inputs are also considered in the context of a similar sum-aggregation-based permutation-invariant model proposed in Yarotsky (2022). This model is also shown to be universal.

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 $p \to M!$.

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. (2017) 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 dimension is at least as high as the number of inputs. Further, we relate these insights back to the broader paradigm of Janossy pooling, as introduced by Murphy et al. (2019).

Continuing our focus on continuous transformations, we adopt a topological perspective to answer the question of whether requiring universal function *representation* is an overly strong criterion, and whether universal function *approximation* would be possible even with lower-dimensional latent spaces. We turn again to the Deep Sets architecture and answer this question in the negative. We show that functions exist which cannot be approximated when the latent space dimension is lower than the number of inputs.

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. We also believe that a deeper exploration of the connections between our results and other aggregation-based methods, particularly self-attention, would be valuable. Finally, practitioners would benefit from stronger and more specific practical guidance on the application of Deep Sets. We see two main directions for producing such guidance. Firstly, further theoretical work, particularly on the training-time behaviour of Deep Sets, could strengthen the link between theoretical knowledge and practical recommendations. Secondly, a detailed empirical investigation would be very valuable, particularly for assessing how the performance of Deep Sets varies in response to changes in the task, the latent dimension, and the number of input points.

We hope that our work will inspire future theoretical treatments of the subject, while also providing practical guidance for machine learning practitioners.

Acknowledgments

Appendix A. Mathematical Remarks

In this section we discuss some considerations arising from the mathematics in the main text.

A.1 Infinite Sums

Throughout this paper we consider expressions of the form

$$\Phi(\mathbf{x}) = \sum_{x \in \mathbf{x}} \phi(x), \tag{23}$$

where $\mathbf{x}$ is an arbitrary set. The meaning of this expression is clear when $\mathbf{x}$ is finite, but when $\mathbf{x}$ is infinite, we must be precise about what we mean.

A.1.1 COUNTABLE SUMS

We usually denote countable sums as, for example, $\sum_{i=1}^{\infty} x_i$. Note that there is an ordering of the x_i here, whereas there is no ordering in Equation (23). 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 $\sum_{i=1}^{\infty} \frac{(-1)^i}{i}$ can be made to converge to any real number simply by reordering the terms of the sum. For expressions like Equation (23) 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 $\sum_{x \in \mathbf{x}} |\phi(x)|$ is convergent. For any $\mathbf{x}$ where this is not the case, we will set $\Phi(\mathbf{x}) = \infty$.

A.1.2 UNCOUNTABLE SUMS

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

A.2 Continuity on Variable-Size Sets

In Theorem 16, we consider functions f on $[0,1]^{\leq M}$, the set of all subsets of $[0,1]$ containing at most M elements. When we say that such a function is "continuous", we mean that the function f_m induced by f on $[0,1]^m$ by $f_m(x_1, \ldots, x_m) \coloneqq f(\{x_1, \ldots, x_m\})$ is continuous for every $m \leq M$.

A.3 Remark on Theorem 12

The proof of Theorem 12 from Zaheer et al. (2017) can be extended to deal with multisets, which are sets with repeated elements. To that end, we replace the mapping to natural numbers $c(x) : \mathfrak{U} \to \mathbb{N}$ with a mapping to prime numbers $p(x) : \mathfrak{U} \to \mathbb{P}$. We then choose $\phi(x) = -\log p(x)$. Therefore,

$$\Phi(\mathbf{x}) = \sum_{m=1}^{M} \phi(x_m) = \log \prod_{m=1}^{M} \frac{1}{p(x_m)},$$
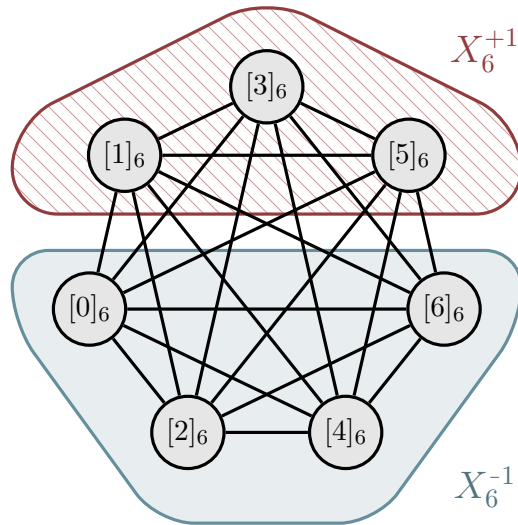
Figure 11: The 6-dimensional simplex Δ_6, and the low-dimensional faces X_6^{+1} and X_6^{-1}, visualised using the complete graph K_7. Here, $[m]_6$ denotes the vertex of Δ_6 whose coordinates in $\mathbb{R}^6$ consist of m ones followed by $6-m$ negative ones.

which takes a unique value for each distinct $\mathbf{x}$, therefore extending the validity of the proof to multisets.

This choice of ϕ diverges with infinite multiset size. In fact, it is straightforward to show that there is no function ϕ for which Φ provides a unique mapping for arbitrary multisets while at same time guaranteeing convergence for infinitely large multisets. To see this, let ϕ be a non-zero function, and let $x \in \mathfrak{U}$ such that $\phi(x) = a \neq 0$. Then the multiset $\mathbf{x}$ comprising infinitely many identical members x would give

$$\Phi(\mathbf{x}) = \sum_{i=1}^{\infty} \phi(x_m) = \sum_{i=1}^{\infty} a = \pm\infty \,.$$

A.4 Visualising High-Dimensional Simplices

In the proof of Theorem 20, 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, as shown in Figure 11.

An N-dimensional simplex can be defined as the convex hull of $N+1$ points in general position.[23] 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,

---

23. That is, no three points are collinear, no four points are coplanar, etc.

and this can be viewed as an induced K_N subgraph of K_{N+1}. Lower-dimensional faces can be viewed similarly.

Under 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 such vertex can be written in shorthand as $[m]_N$, indicating the point 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 11 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. The faces $\Delta_6^{(0)}$ and $\Delta_6^{(6)}$ correspond to the subgraphs obtained by removing $[0]_6$ and $[6]_6$ respectively.

A.5 Extending Theorem 20 to Higher Dimensions

As discussed in the main text, an interesting line of future work lies in extending Theorem 20 to sets of higher-dimensional elements. We briefly make note of an obstacle to extending the arguments of this work to the higher-dimensional case.

Our proof of Theorem 20 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 (8). 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. We can codify this property of a function f as follows:

1. f is permutation invariant.

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

We refer to any such function as a *quasi-sorting function*. `sort` is a quasi-sorting function, but this definition also 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. 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 2-dimensional element. We can visualise a set of two 2-dimensional elements as being a pair of points in the plane.

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 12, 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.

$$g(\mathbf{x}) = \begin{pmatrix} -\tfrac{1}{2}\sqrt{2} & \tfrac{1}{2}\sqrt{2} \\ \tfrac{1}{2}\sqrt{2} & -\tfrac{1}{2}\sqrt{2} \end{pmatrix}$$

(a)

$$g(\mathbf{x}) = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$$

(b)

$$g(\mathbf{x}) = \begin{pmatrix} -\tfrac{1}{2}\sqrt{2} & -\tfrac{1}{2}\sqrt{2} \\ \tfrac{1}{2}\sqrt{2} & \tfrac{1}{2}\sqrt{2} \end{pmatrix}$$

(c)

$$g(\mathbf{x}) = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}$$

(d)

$$g(\mathbf{x}) = \begin{pmatrix} -\tfrac{1}{2}\sqrt{2} & \tfrac{1}{2}\sqrt{2} \\ \tfrac{1}{2}\sqrt{2} & -\tfrac{1}{2}\sqrt{2} \end{pmatrix}$$
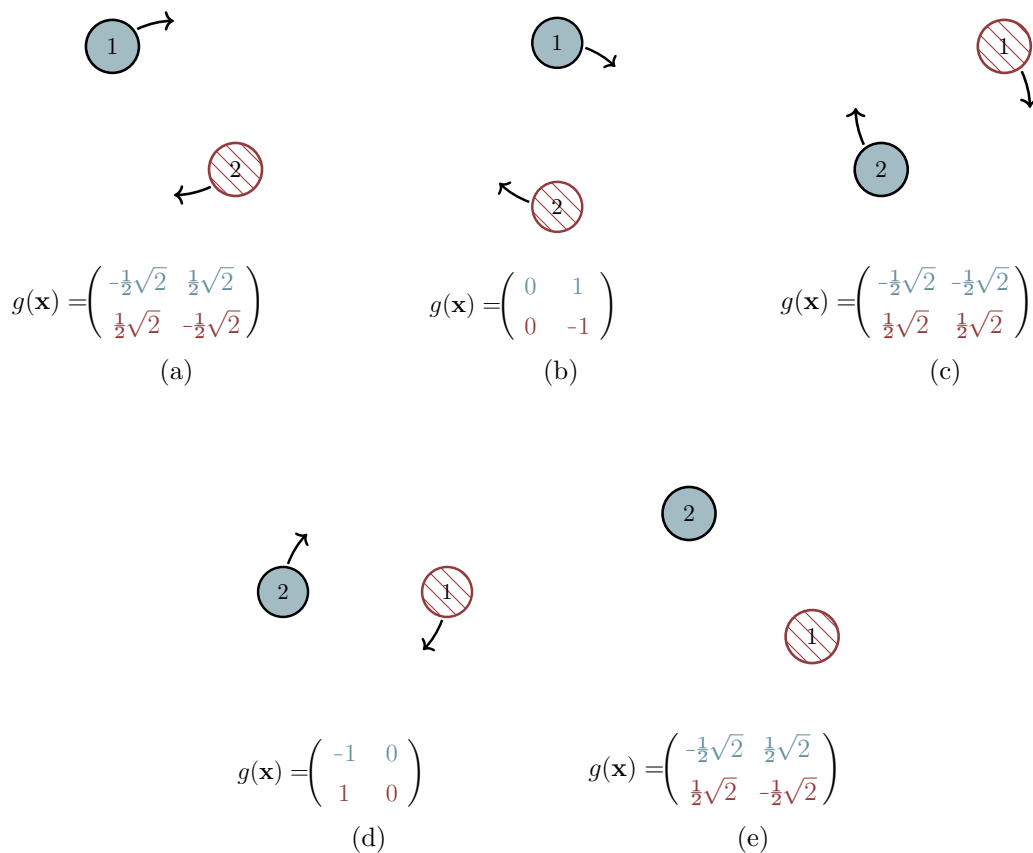
(e)

Figure 12: Rotating a pair of points around their midpoint, and seeing how the canonical ordering changes. Figures 12a and 12e 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 12a and 12e to match, point "1" must switch from blue to red at some point during the 180-degree rotation from 12a to 12e. This switching of colours represents swapping the rows of $g(\mathbf{x})$, which is a discontinuity.

After a 180-degree rotation, the set is the same as it was at the beginning. The canonical permutation—that is, the colouring—is therefore 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 12, this happens between 12b and 12c. This switching of colours corresponds to swapping the rows of the output matrix. Since the rows of the matrix are distinct (because the points represented by each row are a constant distance apart), this swapping is discontinuous, and therefore g is not continuous.

Appendix B. Function Representation

In this section we state and prove some additional theorems on function representation. We also provide proofs of some theorems stated in Section 3.

Theorem 29 *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* $q \in \mathbb{Q}$.

Theorem 30 *Given a space* X, *write* $X^{\mathcal{F}}$ *for the set of all finite subsets of* X. *Let* $f : \mathbb{R}^{\mathcal{F}} \to \mathbb{R}$. *Then* f *is sum-decomposable via* $\mathbb{R}$.

Theorem 31 *If* $\mathfrak{U}$ *is uncountable, then there exist functions* $f : 2^{\mathfrak{U}} \to \mathbb{R}$ *which are not sum-decomposable. This holds even if the sum-decomposition* (ρ, ϕ) *is allowed to be discontinuous.*

B.1 Proof of Theorem 29

Theorem 29 *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 $f(\mathbf{x}) = \mathtt{sup}(\mathbf{x})$, the least upper bound of $\mathbf{x}$. Let (ρ, ϕ) be a sum-decomposition of $\mathtt{sup}$ via $\mathbb{R}$, so we have

$$\mathtt{sup}(\mathbf{x}) = \rho\big(\Phi(\mathbf{x})\big). \tag{24}$$

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

$$\Phi(\mathbf{x}) = \Phi(\mathbf{x}) + \phi(q) = \Phi\big(\mathbf{x} \cup \{q\}\big).$$

So by Equation (24), $\mathtt{sup}(\mathbf{x}) = \mathtt{sup}\big(\mathbf{x} \cup \{q\}\big)$ for all $\mathbf{x} \subset \mathbb{Q}$. But then, for instance, we would have

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

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

Next, note that $\Phi(\mathbf{x})$ must be finite for every upper-bounded $\mathbf{x} \subset \mathbb{Q}$ (since $\mathtt{sup}$ is undefined for unbounded $\mathbf{x}$, 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 $\Phi(\mathbf{x}) = \infty$ for some upper-bounded set $\mathbf{x}$. Then

$$
\begin{aligned}
\sup(\mathbf{x}) &= \rho\big(\Phi(\mathbf{x})\big) \\
&= \rho(\infty) \\
&= \rho\Big(\infty + \phi\big(\sup(\mathbf{x}) + 1\big)\Big) \\
&= \rho\Big(\Phi\big(\mathbf{x} \cup \{\sup(\mathbf{x}) + 1\}\big)\Big) \\
&= \sup\big(\mathbf{x} \cup \{\sup(\mathbf{x}) + 1\}\big) \\
&= \sup(\mathbf{x}) + 1 \,.
\end{aligned}
$$

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

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

Finally, let $q \in \mathbb{Q}$. We will now construct a sequence q_n with:

1. $q_n \to q$,

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

Since we have already shown that $\phi(q) \neq 0$, point 2 implies that $\phi(q_n) \not\to \phi(q)$. By point 1, this implies 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 at q. Since q was arbitrary, ϕ is discontinuous everywhere. ∎

B.2 Proof of Theorem 30

Theorem 30 *Given a space X, write $X^{\mathcal{F}}$ for the set of all finite subsets of X. 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 $\Phi(\mathbf{x}) = \sum_{x \in \mathbf{x}} \phi(x)$. 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 $\mathbf{x} \subset \mathbb{R}$ is *finite-sum-distinct* (f.s.d.) if, for any finite subsets $\mathbf{a}, \mathbf{b} \subset \mathbf{x}$, $\sum_{a \in \mathbf{a}} a \neq \sum_{b \in \mathbf{b}} b$. Let $\mathfrak{c}$ be the cardinality of $\mathbb{R}$. If we can show that there is a finite-sum-distinct set $\mathbf{d} \subset \mathbb{R}$ with cardinality $\mathfrak{c}$, then we can simply choose ϕ to be an injection from $\mathbb{R}$ to $\mathbf{d}$. 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 $\mathbf{d}_{\max}$. Maximality of $\mathbf{d}_{\max}$ within $\mathcal{D}$ means that, for any set $\mathbf{e}$ with $\mathbf{d}_{\max} \subsetneq \mathbf{e}$, $\mathbf{e} \notin \mathcal{D}$. Equivalently, the maximality of $\mathbf{d}_{\max}$ means that adding any additional elements to $\mathbf{d}_{\max}$ causes it to lose the property of finite-sum-distinctness. We claim that any such maximal set $\mathbf{d}_{\max}$ must have cardinality $\mathfrak{c}$.

To see this, let $\mathbf{d}$ be a f.s.d. set with infinite[24] cardinality $\kappa < \mathfrak{c}$. We will show that $\mathbf{d} \neq \mathbf{d}_{\max}$. Define the *forbidden elements* with respect to $\mathbf{d}$ to be those elements x of $\mathbb{R}$ such that $\mathbf{d} \cup \{x\}$ is not f.s.d. We denote this set of forbidden elements $F_{\mathbf{d}}$. By definition, we have that $x \in F_{\mathbf{d}}$ if and only if there exist finite subsets $\mathbf{d}^A, \mathbf{d}^B$ of $\mathbf{d}$ such that

$$\sum_{d \in \mathbf{d}^A} d = x + \sum_{d' \in \mathbf{d}^B} d'.$$

So we can write x as a sum of finitely many elements of $\mathbf{d}$, minus a sum of finitely many other elements of $\mathbf{d}$. So there is a surjection from pairs of finite sets of $\mathbf{d}$ to elements of $F_{\mathbf{d}}$. That is,

$$|F_{\mathbf{d}}| \leq |\mathbf{d}^{\mathcal{F}} \times \mathbf{d}^{\mathcal{F}}|.$$

But since $\mathbf{d}$ is infinite,

$$|\mathbf{d}^{\mathcal{F}} \times \mathbf{d}^{\mathcal{F}}| = |\mathbf{d}| = \kappa < \mathfrak{c}.$$

So $|F_{\mathbf{d}}| < \mathfrak{c}$, and therefore $\mathbf{d} \cup F_{\mathbf{d}} \neq \mathbb{R}$. So there exists some $x \in \mathbb{R} \setminus (\mathbf{d} \cup F_{\mathbf{d}})$. $\mathbf{d} \cup \{x\}$ is f.s.d, and therefore $\mathbf{d}$ is not maximal. This demonstrates that $\mathbf{d}_{\max}$ must have cardinality $\mathfrak{c}$.

To complete the proof, it remains to show that $\mathcal{D}$ satisfies the conditions of Zorn's Lemma. That is, it remains to show that every totally ordered subset $\mathcal{C}$ of $\mathcal{D}$ has an upper bound. So let $\mathcal{C}$ be a totally ordered subset of $\mathcal{D}$, and consider

$$\mathbf{c}_{\mathrm{ub}} \coloneqq \bigcup_{\mathbf{c} \in \mathcal{C}} \mathbf{c}.$$

We claim that $\mathbf{c}_{\mathrm{ub}}$ is an upper bound for $\mathcal{C}$ within $\mathcal{D}$. It is clear that $\mathbf{c} \subset \mathbf{c}_{\mathrm{ub}}$ for every $\mathbf{c} \in \mathcal{C}$, so it remains to be shown that $\mathbf{c}_{\mathrm{ub}} \in \mathcal{D}$. That is, we must show that $\mathbf{c}_{\mathrm{ub}}$ is f.s.d.

Let $c_1, \ldots, c_m, c'_1, \ldots, c'_n$ be elements of $\mathbf{c}_{\mathrm{ub}}$. We wish to show that

$$\sum_{i=1}^{m} c_i \neq \sum_{j=1}^{n} c'_j. \tag{25}$$

By construction of $\mathbf{c}_{\mathrm{ub}}$, there must be sets $\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathbf{c}'_1, \ldots, \mathbf{c}'_n \in \mathcal{C}$ with $c_i \in \mathbf{c}_i$ and $c'_j \in \mathbf{c}'_j$. Since $\mathcal{C}$ is totally ordered by inclusion, there exists some set $\mathbf{c}_{\max} \in \{\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathbf{c}'_1, \ldots, \mathbf{c}'_n\}$, with $\mathbf{c}_i \subset \mathbf{c}_{\max}$ and $\mathbf{c}'_j \subset \mathbf{c}_{\max}$ for all i, j. This implies that $c_i \in \mathbf{c}_{\max}$, $c'_j \in \mathbf{c}_{\max}$, for all i, j. Since $\mathbf{c}_{\max} \in \mathcal{C}$, it is finite-sum-distinct. Therefore, Equation (25) holds, and $\mathbf{c}_{\mathrm{ub}}$ is finite-sum-distinct. In summary:

1. $\mathcal{D}$ satisfies the conditions of Zorn's Lemma.

2. Therefore there exists a maximal finite-sum-distinct set, $\mathbf{d}_{\max}$.

3. We have shown that $\mathbf{d}_{\max}$ must have cardinality $\mathfrak{c}$.

---

24. Any finite set $\mathbf{d}$ cannot be maximal within $\mathcal{D}$, since we can add new elements to any finite set without losing finite-sum-distinctness.

4. Since $\mathbf{d}_{\max}$ has cardinality $\mathfrak{c}$, we can choose ϕ to be an injection from $\mathbb{R}$ into $\mathbf{d}_{\max}$.

5. Given such a ϕ, finite-sum-distinctness of $\mathbf{d}_{\max}$ implies that $\Phi(\mathbf{x}) = \sum_{x \in \mathbf{x}} \phi(x)$ is injective on $\mathbb{R}^{\mathcal{F}}$.

6. Given this injective Φ, we choose $\rho = f \circ \Phi^{-1}$.

7. This choice gives us $f(\mathbf{x}) = \rho\big(\Phi(\mathbf{x})\big)$ by construction.

This completes the proof. ∎

B.3 Proof of Theorem 31

Theorem 31 *If $\mathfrak{U}$ is uncountable, then there exist functions $f : 2^{\mathfrak{U}} \to \mathbb{R}$ which are not sum-decomposable. This holds even if the sum-decomposition (ρ, ϕ) is allowed to be discontinuous.*

Proof Consider $f(\mathbf{x}) = \sup(\mathbf{x})$.

As discussed in Appendix A.1.2, a sum over uncountably many elements can converge only if only countably many elements are non-zero. But as in the proof of Theorem 29, $\phi(x) \neq 0$ 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 29, 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. ∎

B.4 Proof of Theorem 15

Theorem 15 *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 trivial. We already know from Zaheer et al. (2017) that the function $\Phi : \Delta_M \to \mathbb{R}^{M+1}$ defined as follows is a homeomorphism onto its image:

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

$$\phi_q(x) := x^q \quad q = 0, \dots, M \,.$$

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

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

$$\widetilde{\phi}_q(x) := x^q \quad q = 1, \dots, M \,.$$

Note that $\Phi_0(\mathbf{x}) = M$ for all $\mathbf{x}$, 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

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

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

B.5 Proof of Theorem 16

Theorem 16 (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 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 proof by Φ,

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

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

$$\Phi_q(\mathbf{x}) = \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(x_m)\,.$$

For an input set $\mathbf{x}$ 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 $\Phi_q(\mathbf{x})$ for input sets $\mathbf{x}$ of arbitrary size M':

$$\Phi_q(\mathbf{x}) = \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(k)\,. \tag{26}$$

Equation (26) is no longer strictly speaking a sum-decomposition. This can be fixed by rearranging the expression:

$$
\begin{aligned}
\Phi_q(\mathbf{x}) &= \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=M'+1}^{M} \phi_q(k) \\
&= \sum_{m=1}^{M'} \phi_q(x_m) + \sum_{m=1}^{M} \phi_q(k) - \sum_{m=1}^{M'} \phi_q(k) \\
&= \sum_{m=1}^{M'} [\phi_q(x_m) - \phi_q(k)] + \sum_{m=1}^{M} \phi_q(k) \, .
\end{aligned}
\tag{27}
$$

The last term in Equation (27) is a constant value which only depends on the choice of k and is independent of $\mathbf{x}$ 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 $\widehat{\Phi}(\mathbf{x})$ with

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

$\widehat{\Phi}$ is injective, since Φ is injective and the last term in the above sum is constant. $\widehat{\Phi}$ is also in the necessary form for a sum-decomposition.

For each $m < M$, we can follow the reasoning used in the rest of the proof of Theorem 13 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 $[0,1]^m$ under the continuous function $\widehat{\Phi}_m$. The domains of the $\widehat{\Phi}_{m,M}^{-1}$ can also be made disjoint, by taking k sufficiently large ($k \geq 2M$ suffices). 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 subset of $\mathbb{R}$ with $k \in \mathcal{Y}$, $[0,1] \subset \mathcal{Y}$. Let $\widehat{f}$ be a permutation-invariant function on $\mathcal{Y}^M$ satisfying

$$
\widehat{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \mathbf{x} \subset [0,1] \, , \\ f\big(\mathbf{x} \cap [0,1]\big) & \mathbf{x} \subset [0,1] \cup \{k\} \, . \end{cases}
$$

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.

■

B.6 Max-Decomposition

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

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

where the $\texttt{max}$ 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 14, but which establishes limitations on max-decomposition, rather than sum-decomposition.

Theorem 32 *Let $M > N \in \mathbb{N}$. Then there exist continuous permutation-invariant 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 $\texttt{sum}(\mathbf{x}) = \sum_{i=1}^{M} x_m$. 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

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

That is, $\phi(x_{\mu(j)})$ attains the maximal value in the j-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

$$\widetilde{x}_i := x_i \text{ for } i \neq m \, ,$$
$$\widetilde{x}_m := x_{\mu(1)} \, .$$

Then

$$\texttt{max}_i\big(\phi(x_i)\big) = \texttt{max}_i\big(\phi(\widetilde{x}_i)\big) \, .$$

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 $\texttt{sum}$. But ϕ was arbitrary, so no max-decomposition exists.

∎

Appendix C. Function Approximation

In this section we formalise the stronger statement of Theorem 20 described in Section 4, and we complete the proof of Lemma 24.

C.1 A Stronger Statement of Theorem 20

Given a function $f : I_M \to \mathbb{R}$, write V_f for the *variation* of f:

$$V_f := \max_{\mathbf{x} \in I_M} \big(f(\mathbf{x}) \big) - \min_{\mathbf{x} \in I_M} \big(f(\mathbf{x}) \big) .$$

Theorem 20 (Strong Form) *Let $M, N \in \mathbb{N}$ with $M > N$. There exists a continuous, non-constant, permutation-invariant function $f : I_M \to \mathbb{R}$ 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 $\mathbf{x} \in I_M$ with $\big| \rho\big(\Phi(\mathbf{x})\big) - f(\mathbf{x}) \big| \geq \frac{V_f}{2}$.*

C.2 Completing the Proof of Lemma 24

Lemma 24 *Let $n \in \mathbb{N}$. Let Γ_n be defined as in Equation (12). Then there exists a continuous function $\nu_n : I_n \to \Delta_n$ such that, for $\mathbf{x} \in \partial I_n$,*

$$\Gamma_n\big(\nu_n(\text{-}\mathbf{x})\big) = -\Gamma_n\big(\nu_n(\mathbf{x})\big) . \tag{19}$$

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. Conclusion 4 is the main claim of the lemma—Conclusions 1 to 3 will enable us to prove Conclusion 4.

For the base case $n = 1$, let $\nu_1(\mathbf{x}) := \mathbf{x}$. 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, so the base case holds.

For the inductive step, we will prove each conclusion in sequence.

C.2.1 CONCLUSION 1—DEFINING THE CONTINUOUS FUNCTION ν_n

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

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

These definitions follow the partial definition of ν_n given in Section 4.2. ν_n^+ and ν_n^- are both continuous by continuity of ν_{n-1} (Hypothesis 1). Both functions have codomain Δ_n by Hypothesis 3. 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 4.2, 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—Figure 13 gives an example of how this interpolation scheme works for $n = 5$.

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:

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

Then we fix ν_n at the endpoints as follows:

$$\nu_n(\mathbf{x}_{(i)})_j := \begin{cases} y_j^+ & i < j \,, \\ m_j & i = j \,, \\ y_j^- & i > j \,. \end{cases} \tag{28}$$

The special case $i = j = 1$ depends on $\nu_n^-(\mathbf{x}^-)_0$, which is undefined—for consistency with ν_n^+, we must set $\nu_n(\mathbf{x}_{(1)})_1 = y_1^+$. By continuity of ν_n^+, ν_n^-, $\texttt{median}$, and linear interpolation, the function ν_n defined by the interpolation scheme is continuous. 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.

C.2.2 CONCLUSION 2—A USEFUL INEQUALITY

First we show that for every j, we have

$$y_j^+ \geq y_j^- \,. \tag{29}$$

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 (29) 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 (29) is exactly Hypothesis 2, and is therefore true by induction. Note also from the above definitions that

$$\nu_n(\mathbf{x}^-)_j = \nu_n((\text{-}\mathbf{x})^+)_{j+1} \qquad 1 \leq j < n \,. \tag{30}$$
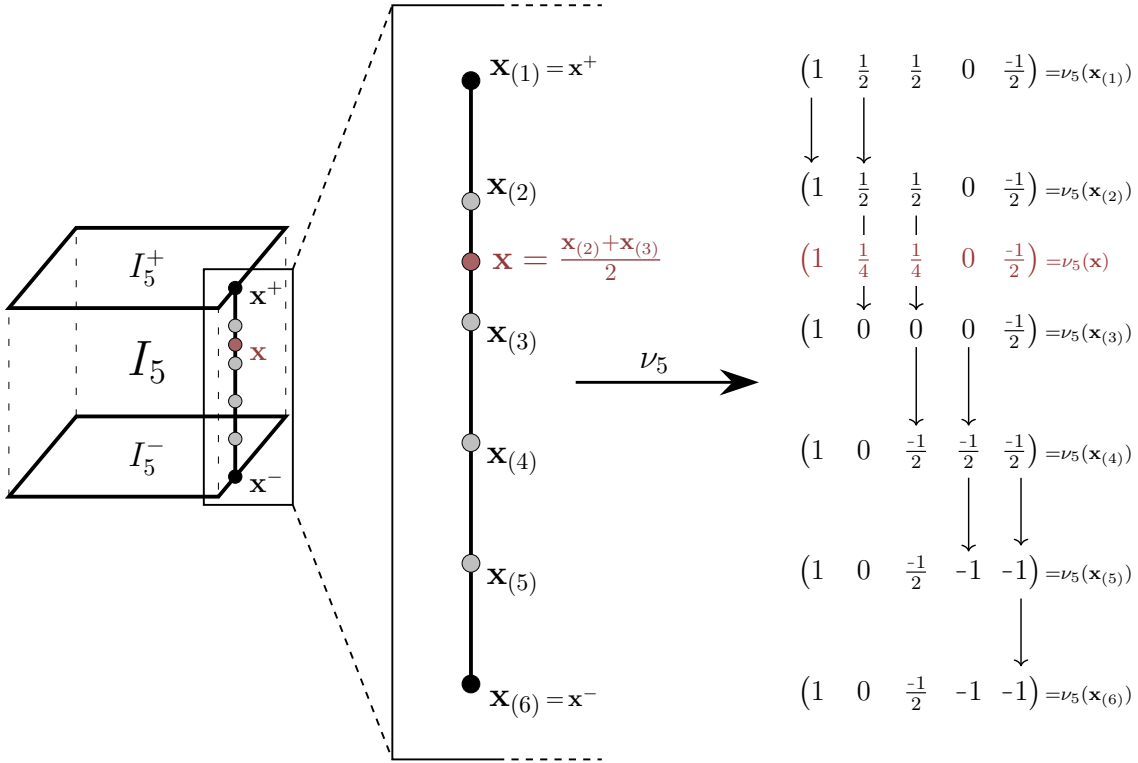
Figure 13: An example of how the interpolation scheme works for $n = 5$, with $\mathbf{x}$ chosen to lie on a surface vertical. The specific values of $\nu_5(\mathbf{x}_{(1)})$ and $\nu_5(\mathbf{x}_{(6)})$ shown here are purely illustrative, but the remaining values are computed according to our interpolation scheme. Equation (28) 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 (28) 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 main diagonal. Crossing the main diagonal corresponds to moving along the vertical arrows shown above.

The final part of the proof of Lemma 24 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 $\mathbf{0}$ in the alternating sum which defines Γ_n (Equation 12). 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, $\Gamma_5\big(\nu_5(\mathbf{x})\big) = \Gamma_5\big(\nu_5(\mathbf{x}_{(2)})\big) = \Gamma_5\big(\nu_5(\mathbf{x}_{(3)})\big)$, because the second and third coordinates are equal and cancel to $\mathbf{0}$ 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 (33), which relies on Hypothesis 2, is crucial.

47

Figure 13 (cont.): 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 (or equivalently, only changing across the main diagonal of the matrix $\mathcal{N}$). However, the Γ-equivalence property discussed above only holds on surface verticals, and does not hold on the rest of I_n.

Equation (29) implies, by definition of m_j, that $y_j^+ \geq m_j \geq y_j^-$ for all j. Therefore, by Equation (28),

$$\nu_n(\mathbf{x}_{(i)})_j \geq \nu_n(\mathbf{x}_{(i+1)})_j \qquad 1 \leq i \leq n, \text{ all } j. \tag{31}$$

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

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

Therefore for $1 \leq j < n$, by Equation (30), we have

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

This final inequality is exactly Conclusion 2.

C.2.3 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 $\nu_n(\mathbf{x})_j \geq \nu_n(\mathbf{x})_{j+1}$ 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 desired inequality is therefore as follows:

$$\nu_n(\mathbf{x}_{(i)})_j \geq \nu_n(\mathbf{x}_{(i)})_{j+1} \qquad \text{all } i, \ 1 \leq j < n. \tag{32}$$

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

$$y_{j-1}^- \geq y_{j+1}^+. \tag{33}$$

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 (33) therefore follows directly from Hypothesis 2.

To prove Equation (32), there are five cases to check, enumerated below. For case 4, we will make use of the fact that $a \geq \mathtt{median}(a, b, c)$ if and only if $a \geq b$ or $a \geq c$. For case 5, we will make use of the fact that $\mathtt{median}(a, b, c) \geq \mathtt{min}(a, b)$.

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

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

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

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

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

This follows because $y_{j+1}^+ \leq \mathtt{min}(y_{j-1}^-, y_j^+)$. Equation (33) implies $y_{j+1}^+ \leq y_{j-1}^-$, and $y_{j+1}^+ \leq y_j^+$ is the I_n^+ case.

Equation (32) therefore holds, so Conclusion 3 is established.

C.2.4 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:

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

The top inequalities hold because $\nu_n(\mathbf{x}) \in \Delta_n$ (Conclusion 3). The bottom left inequality is Equation (29). The bottom right inequality is Equation (33).

Conclusion 4 applies to $\mathbf{x} \in \partial I_n$. As shown in Section 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 $\mathbf{x} \in \partial I_n \backslash (I_n^+ \cup I_n^-)$—that is, points lying on surface verticals—and for the remainder of the proof we consider only these points. We claim that $\Gamma_n\big(\nu_n(\mathbf{x})\big) = \Gamma_n\big(\nu_n(\mathbf{x}^+)\big)$. That is, $\Gamma_n \circ \nu_n$ is constant along surface verticals. As noted in Section 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:

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

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 $\mathbf{0}$. 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:

$$\nu_n(\mathbf{x}_{(I)})_I = m_I, \qquad\qquad \nu_n(\mathbf{x}_{(I)})_{I+1} = y_{I+1}^+,$$
$$\nu_n(\mathbf{x}_{(I+1)})_I = y_I^-, \qquad\qquad \nu_n(\mathbf{x}_{(I+1)})_{I+1} = m_{I+1}.$$

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 $\nu_n(\mathbf{x}) \in \Delta_n$ (from Conclusion 3), and the top-to-bottom inequalities are instances of Equation (31).

$$
\begin{array}{ccc}
m_I & \geq & y_{I+1}^+ \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
y_I^- & \geq & m_{I+1}
\end{array}
\tag{34}
$$

Recalling the visualisation of Figure 13, 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 entries of $\mathcal{N}$ are (non-strictly) decreasing downwards and decreasing to the right—see Figure 13 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 34 are equal);

2. $m_I = y_I^-$ and $m_{I+1} = y_{I+1}^+$ (rows of Equation 34 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 (34), either the top or the left equality holds, and either the bottom or the right equality holds. This will imply that either the rows or the columns are equal, since otherwise 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

$$
\begin{aligned}
y_I^+ &> y_I^-, & y_I^+ &> y_{I+1}^+, \\
y_{I-1}^- &> y_I^-, & y_{I-1}^- &> y_{I+1}^+.
\end{aligned}
\tag{35}
$$

In the following representation, letting $a := \mathtt{min}(y_I^+, y_{I-1}^-)$, 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.

$$
\begin{array}{c}
\overset{\geq \mathbf{a} \qquad\qquad \mathbf{a} \quad < \mathbf{a}}{y_1^+ \cdots \;\; \cdots \quad y_I^+ \Big/ y_{I+1}^+ \cdots y_n^+} \\[2pt]
y_1^- \cdots y_{I-1}^- \Big/ y_I^- \quad \cdots\;\cdots\; y_n^-
\end{array}
$$

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 4.2, specifically Equation (22), 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. We have just shown that Equation (35) implies that these numbers

have differing parity, so Equation (35) is therefore false. 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 (34) 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 $\Gamma_n \circ \nu_n$ 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 $\Gamma_n \circ \nu_n$ is constant along the final section of the surface vertical, and Conclusion 4 holds.

■

References

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 4502–4510, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/3147da8ab4a0437c15ef51a5cc7f2dc4-Abstract.html.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. URL http://arxiv.org/abs/1806.01261.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. URL http://arxiv.org/abs/1308.3432.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 950–959, 2020. URL https://proceedings.mlr.press/v119/blondel20a.html.

Karol Borsuk. Drei Sätze über die n-dimensionale euklidische Sphäre. *Fundamenta Mathematicae*, 20(1):177–190, 1933. URL http://eudml.org/doc/212624.

Christian Bueno and Alan Hylton. On the representation power of set pooling networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 17170–17182, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/8ea1e4f9f24c38f168d538c9cfc50a14-Abstract.html.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989. URL https://doi.org/10.1007/BF02551274.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 3318–3328, 2021. URL https://proceedings.mlr.press/v139/finzi21a.html.

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. *Sets and Parts Workshop, Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1907.12887.

Fabian B. Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages

1970–1981, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/15231a7ce4ba789d13b722cc5c955834-Abstract.html`.

Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989. URL `https://doi.org/10.1016/0893-6080(89)90003-8`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, pages 2672–2680, 2014. URL `https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html`.

Jiequn Han, Yingzhou Li, Lin Lin, Jianfeng Lu, Jiefu Zhang, and Linfeng Zhang. Universal approximation of symmetric and anti-symmetric functions. *arXiv preprint arXiv:1912.01765*, 2019. URL `http://arxiv.org/abs/1912.01765`.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. URL `https://doi.org/10.1016/0893-6080(89)90020-8`.

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 *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753, 2019. URL `https://proceedings.mlr.press/v97/lee19d.html`.

Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. URL `https://doi.org/10.1109/tpami.2021.3134200`.

Noel G. Lloyd. *Degree Theory*. Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge, 1978.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 2156–2, 2019a. URL `https://papers.nips.cc/paper/2019/hash/bb04af0f7ecaee4aae62035497da1387-Abstract.html`.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations (ICLR)*, 2019b. URL `https://openreview.net/forum?id=Syx72jC9tm`.

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. URL `https://openreview.net/forum?id=BJluy2RcFm`.

Chirag Pabbaraju and Prateek Jain. Learning functions over sets via permutation adversarial networks. *arXiv preprint arXiv:1907.05638*, 2019. URL `https://arxiv.org/abs/1907.05638`.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. URL `https://openaccess.thecvf.com/content_cvpr_2017/html/Qi_PointNet_Deep_Learning_CVPR_2017_paper.html`.

Charles Ruizhongtai 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(NeurIPS)*, volume 30, pages 5099–5108, 2017b. URL `https://papers.nips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html`.

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

David W. Romero and Jean-Baptiste Cordonnier. Group equivariant stand-alone self-attention for vision. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=JkfYjnOEo6M`.

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 (NeurIPS)*, volume 30, pages 4967–4976, 2017. URL `https://papers.nips.cc/paper/2017/hash/e6acf4b0f69f6f6e60e9a815938aa1ff-Abstract.html`.

Nimrod Segol and Yaron Lipman. On universal equivariant set networks. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=HkxTwkrKDB`.

Hadar Serviansky, Nimrod Segol, Jonathan Shlomi, Kyle Cranmer, Eilam Gross, Haggai Maron, and Yaron Lipman. Set2Graph: Learning graphs from sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 22080–22091, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/fb4ab556bc42d6f0ee0f9e24ec4d1af0-Abstract.html`.

Marshall H. Stone. The generalized Weierstrass approximation theorem. *Mathematics Magazine*, 21(4):167–184, 1948. URL `https://doi.org/10.2307/3029750`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008, 2017. URL `https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 6487–6494, 2019. URL `https://proceedings.mlr.press/v97/wagstaff19a.html`.

Jiayi Wei, Maruth Goyal, Greg Durrett, and Isil Dillig. LambdaNet: Probabilistic type inference using graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=Hkx6hANtwH`.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 1511–1522, 2019. URL `https://papers.nips.cc/paper/2019/hash/1efa39bcaec6f3900149160693694536-Abstract.html`.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022. URL `https://doi.org/10.1007/s00365-021-09546-1`.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://openreview.net/forum?id=Skh4jRcKQ`.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Poczós, Russ R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 3391–3401, 2017. URL `https://papers.nips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html`.