Sumformer: Universal Approximation for Efficient Transformers

Silas Alberti [1 2] **Niclas Dern** [3] **Laura Thesing** [1] **Gitta Kutyniok** [1]

Abstract

Natural language processing (NLP) made an impressive jump with the introduction of Transformers. ChatGPT is one of the most famous examples, changing the perception of the possibilities of AI even outside the research community. However, besides the impressive performance, the quadratic time and space complexity of Transformers with respect to sequence length pose significant limitations for handling long sequences. While efficient Transformer architectures like Linformer and Performer with linear complexity have emerged as promising solutions, their theoretical understanding remains limited. In this paper, we introduce Sumformer, a novel and simple architecture capable of universally approximating equivariant sequence-to-sequence functions. We use Sumformer to give the first universal approximation results for Linformer and Performer. Moreover, we derive a new proof for Transformers, showing that just one attention layer is sufficient for universal approximation.

1. Introduction

The introduction of the Transformer architecture in 2017 (Vaswani et al., 2017) commenced a new revolution in the field of deep learning. It not only revolutionized Natural Language Processing with famous models like BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020) but also other areas like computer vision (Dosovitskiy et al., 2020) and biology (Jumper et al., 2021).

However, Transformers can become computationally expensive at scale. In many cases, the primary performance bottleneck is the attention mechanism that needs to compute a $n \times n$-Matrix, where $n \in \mathbb{N}$ is the length of the input sequence. Therefore, the computational complexity of a forward pass grows $O(n^2)$ with the sequence length. This establishes the sequence length as one of the major bottlenecks when using Transformers for long sequences, which are encountered in many fields, such as NLP for processing longer documents like books, time series (Wen et al., 2022), genomics (Eraslan et al., 2019), and reinforcement learning (Chen et al., 2021).

To address this problem, many new architectures have been proposed (Child et al., 2019; Wang et al., 2020; Choromanski et al., 2020; Katharopoulos et al., 2020; Kitaev et al., 2020; Zaheer et al., 2020; Tay et al., 2022; Beltagy et al., 2020). These can be roughly divided into *sparse Transformers* and *efficient Transformers* (Tay et al., 2022). In some cases, the complexity can be reduced to as low as $\mathcal{O}(n)$. While, in practice, these new architectures do not match the performance of Transformers, the performance relative to the decrease in computational cost makes them promising.

Besides their empirical performance, little is known about the theoretical properties of these new architectures. Particularly, they have not yet been studied from the perspective of expressivity. This paper shows that the efficient Transformers, Linformer and Performer, are universal approximators of equivariant continuous sequence-to-sequence functions on compact sets.

1.1. Summary of contributions

In this paper, we introduce the *Sumformer* architecture. This architecture serves as a simple tool that we can use to investigate the expressive power of Transformers and two selected efficient Transformer architectures: Linformer (Wang et al., 2020) and Performer (Choromanski et al., 2020). We chose the latter two architectures since they performed best in the Long Range Arena benchmark (Tay et al., 2020).

First, we show that the Sumformer architecture is able to approximate all continuous equivariant sequence-to-sequence functions on compact sets (Sec. 3). We give two different proofs: A continuous proof based on the algebra of multisymmetric polynomials, and a discrete proof based on a piecewise constant approximation.

[1]Department of Mathematics, Ludwig-Maximilians-Universität München, Germany [2]Department of Electrical Engineering, Stanford University, United States [3]TUM School of Computation, Information and Technology, Technical University of Munich, Munich, Germany. Correspondence to: Silas Alberti <salberti@stanford.edu>.

Proceedings of the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

Using this result, we give a new proof of the universal approximation theorem for Transformers (Sec. 4.2). This proof improves significantly upon the previous result from (Yun et al., 2019), by reducing the number of necessary attention layers. Our proof only needs one attention layer, whereas the number of attention layers in (Yun et al., 2019) grows exponentially with the token dimension.

Based on this proof, we give the first proof that Linformer and Performer are universal approximators (Sec. 4.3). This is the first universal approximation theorem for efficient Transformers, showing that despite using the efficient attention mechanisms we do not suffer from a loss in expressivity.

Our numerical experiments (Sec. 5) using the Sumformer architecture show that the Sumformer architecture is not only theoretically useful but can indeed be used to learn functions using gradient descent. Furthermore, we find an exponential relation between the token dimension and the necessary latent dimension.

1.2. Related work

This paper analyses the expressive power of newly evolving efficient Transformer architectures. Expressivity is a natural first question when investigating the possibilities and limitations of network architectures. Therefore, the question of which functions can be approximated (uniformly) with neural networks and their variance is of great interest.

The publications mentioned in the following are by no means exhaustive but rather a selection: The first universal approximation result for neural networks dates back to 1989 with the universal approximation theorem in (Hornik et al., 1989). Further investigations also for deeper networks were made in (Barron, 1994; Mhaskar, 1996; Shaham et al., 2018). These results were extended to functions with the rectified linear unit (ReLU) activation function in (Petersen & Voigtlaender, 2018; Lu et al., 2017; Yarotsky, 2017; Gühring et al., 2020) and convolutional neural networks in (Yarotsky, 2022). Feed forward neural networks with fewer non-zero coefficients and values that can be stored with fewer bits and therefore improve memory efficiency are investigated in (Bolcskei et al., 2019).

The Transformer architecture has not been explored as much in the literature. We know from (Yun et al., 2019) that Transformers are universal approximators in L_p, for $1 \leqslant p < \infty$ for continuous sequence-to-sequence functions. Moreover, it has been shown in (Yun et al., 2020) that under certain assumptions on the sparsity pattern, sparse Transformers form universal approximators in the same setting. The expressivity of the self-attention mechanism has also been examined from a complexity theory perspective in (Likhosherstov et al., 2021). For efficient Transformer architectures, no such universal approximation results exist to our knowledge.

The main inspiration for this work is the Deep Sets architecture which shows a universal approximation theorem for invariant functions on sets (Zaheer et al., 2017; Wagstaff et al., 2019). We expand on their theorems in the continuous case (Theorem 7 & 9) and expand the theory from invariant functions on sets to equivariant functions on sequences. A similar model to Sumformer was proposed, and universality was proven in (Hutter, 2020). However, the connection to (efficient) Transformers was not made. We build upon their proof and propose an alternative discontinuous version. Concurrent work has given the continuous proof in higher dimension, but neither considers the expansion to equivariant sequence-to-sequence functions nor to Transformers (Chen et al., 2022).

2. Preliminaries

This section describes the setting and states helpful theorems for our proofs and experiments.

We first recall the definition of *attention heads* and the *Transformer block* from (Vaswani et al., 2017). Afterwards, we describe how they can be changed to be more efficient with Linformer and Performer.

Furthermore, we define *equivariant*, *semi-invariant* functions, *multisymmetric polynomials*, and *multisymmetric power sums* (Briand, 2004). We also state important theorems about the relations between these concepts from (Hutter, 2020) and (Briand, 2004). Lastly, we recall an important theorem from (Zaheer et al., 2017).

2.1. Transformer

The central part of the Transformer is the (self-)attention layer, which is able to connect every element in a sequence with every other element.

Definition 2.1 (Attention Head (Vaswani et al., 2017))**.** Let $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$ be weight matrices and let $\rho : \mathbb{R}^d \to \mathbb{R}^d$ be the softmax function. A *(self-)attention head* is a function $\mathrm{AttHead} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ with

$$\mathrm{AttHead}(X) := \underbrace{\rho\Big((XW_Q)(XW_K)^\top/\sqrt{d}\Big)}_{A} XW_V \quad (1)$$

where ρ is applied row-wise. We call $A \in \mathbb{R}^{n\times n}$ the *attention matrix*.

Computing the attention matrix A has a computational complexity of $\mathcal{O}(n^2)$, thereby forming the highest cost in evaluating the Transformer.

In the next step, we combine the attention heads to an attention layer by concatenating h attention heads and multiplying them with another weight matrix W_O.

Definition 2.2 (Attention Layer (Vaswani et al., 2017))**.** Let $h \in \mathbb{N}$, let $\mathrm{AttHead}_1, \dots, \mathrm{AttHead}_h$ be attention heads and let $W_O \in \mathbb{R}^{hd \times d}$. A *(multi-head) (self-)attention layer* $\mathrm{Att} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ is defined as

$$\mathrm{Att}(X) := [\mathrm{AttHead}_1(X), \dots, \mathrm{AttHead}_h(X)]W_O. \tag{2}$$

For the Transformer architecture the attention layer is combined with fully-connected layers that are applied token-wise. Moreover, there are residual connections between all the layers (He et al., 2016). Those three components together yield the *Transformer block*.

Definition 2.3 (Transformer Block (Vaswani et al., 2017))**.** A *Transformer block* $\mathrm{Block} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ is an attention layer $\mathrm{Att} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ followed by a fully-connected feed forward layer $\mathrm{FC} : \mathbb{R}^d \to \mathbb{R}^d$ with residual connections

$$\mathrm{Block}(X) := X + \mathrm{FC}(X + \mathrm{Att}(X)) \tag{3}$$

where the fully-connected feed-forward layer FC is applied row-wise.

Similar to the concept of feed-forward neural networks, we stack several Transformer blocks after each other by concatenation. The Transformer architecture is then defined as follows.

Definition 2.4 (Transformer Network (Vaswani et al., 2017))**.** Let $\ell \in \mathbb{N}$ and $\mathrm{Block}_1, \dots, \mathrm{Block}_\ell$ be Transformer blocks. A *Transformer network* $\mathcal{T} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ is a composition of Transformer blocks:

$$\mathcal{T}(X) := (\mathrm{Block}_\ell \circ \mathrm{Block}_{\ell-1} \circ \cdots \circ \mathrm{Block}_1)(X). \tag{4}$$

2.2. Efficient Transformer

To address the $O(n^2)$ bottleneck of computing the attention matrix A, various efficient Transformers were introduced. We chose to investigate Linformer and Performer since they stood out in the Long Range Arena benchmark (Tay et al., 2020). Both architectures only replace the attention mechanism and do not change the rest of the architecture.

2.2.1. Linformer

The Linformer architecture is motivated by the observation that the attention matrix A is effectively low rank. This is supported by empirical evidence in actual language models and theoretical results in (Wang et al., 2020).

The Linformer architecture utilizes the Johnson-Lindenstrauss Lemma by using linear projections $E, F \in \mathbb{R}^{k \times n}$ to project the key and value matrix $K = XW_K$ and $V = XW_V$ from $\mathbb{R}^{n \times d}$ to $\mathbb{R}^{k \times d}$. The entries of E and F are sampled from a normal distribution. The precise definition of a Linformer Attention Head is as follows:

Definition 2.5 (Linformer Attention Head (Wang et al., 2020))**.** Let $k \in \mathbb{N}$ with $k < n$ and let $E, F \in \mathbb{R}^{k \times n}$ be linear projection matrices. Furthermore, let $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$, $\rho : \mathbb{R}^d \to \mathbb{R}^d$ be as in the Definition of a Transformer attention head 2.1. A *Linformer attention head* is a function $\mathrm{LinAttHead} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ with

$$\mathrm{LinAttHead}(X) := \rho\Big((XW_Q)(EXW_K)^\top/\sqrt{d}\Big)FXW_V \tag{5}$$

where ρ is applied row-wise.

Then, the new attention matrix $\overline{A} = \rho((XW_Q)(EXW_K)^\top/\sqrt{d})$ will be in $\mathbb{R}^{n \times k}$, giving a computational complexity of $\mathcal{O}(nk)$ instead of $\mathcal{O}(n^2)$. Using the Johnson-Lindenstrauss Lemma it is shown that when k is chosen on the order of $\mathcal{O}(d/\varepsilon^2)$, the attention mechanism is approximated with ε error. Since $\mathcal{O}(d/\varepsilon^2)$ is independent of n, the complexity of Linformer Attention is $\mathcal{O}(n)$ as n increases.

2.2.2. Performer

The key insight that motivates the Performer architecture is the fact that the attention mechanism could be more efficient if the attention matrix had no non-linearity:

$$(QK^T)V = Q(K^TV) \tag{6}$$

This reduces the computational complexity from $O(n^2d)$ to $O(nd^2)$. By interpreting the attention matrix as a kernel matrix, this non-linearity can be replaced by a dot product in a kernel space, enabling the following efficient attention algorithm:

Definition 2.6 (Performer Attention Head (Choromanski et al., 2020))**.** Let $k \in \mathbb{N}$ with $k < n$, let $\omega_1, \dots, \omega_k \sim \mathcal{N}(0, I_d)$ and define $a : \mathbb{R}^d \to \mathbb{R}^k$ as

$$a(x) := \frac{1}{\sqrt{k}} \exp\left(-\frac{\|x\|^2}{2}\right)\big[\exp(\omega_1^\top x), \dots, \exp(\omega_k^\top x)\big]. \tag{7}$$

Furthermore, let $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ be weight matrices. A *Performer attention head* is a function $\mathrm{PerAttHead} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ with

$$\mathrm{PerAttHead}(X) := a(XW_Q)\Big(a(XW_K)^\top(XW_V)\Big) \tag{8}$$

where a is applied row-wise.

With this definition, we avoid the computation of the full attention matrix, which reduces the computational complexity from $O(n^2d)$ to $O(nkd)$.

2.3. Equivariant and Semi-Equivariant Functions

Let $\mathcal{X}$ and $\mathcal{Y}$ be the domain and range of a function, e.g., $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ or $\mathcal{X}, \mathcal{Y} = [0, 1]^d$ in the compact case. We

call an element $X \in \mathcal{X}^n$ a *sequence of n elements* and denote $X = [x_1, \dots, x_n]$. Often, we refer to the elements x_i of the sequence as *points*. In the canonical case $\mathcal{X} \subseteq \mathbb{R}^d$, we can represent sequences $X \in \mathbb{R}^{n \times d}$ as matrices. We call functions of type $f : \mathcal{X}^n \to \mathcal{Y}$ *sequence-to-point* functions.

Definition 2.7 (Equivariance)**.** A sequence-to-point function $f : \mathcal{X}^n \to \mathcal{Y}$, with $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$ is *equivariant* to the order of elements in a sequence if for each permutation $\pi : [n] \to [n]$:

$$f([x_{\pi(1)}, \dots, x_{\pi(n)}]) = [f_{\pi(1)}(X), \dots, f_{\pi(n)}(X)]. \quad (9)$$

We write that $f \in \mathcal{F}^n_{\text{equi}}(\mathcal{X}, \mathcal{Y})$.

Transformers represent sequence-to-sequence functions, but sometimes it is more convenient to work with sequence-to-point functions. To facilitate that, we recall the concept of a *semi-invariant* function (see: (Hutter, 2020)).

Definition 2.8 (Semi-invariance)**.** A sequence-to-point function $g : \mathcal{X}^n \to \mathcal{Y}$ is *semi-invariant* if for each permutation $\pi : [n] \backslash \{1\} \to [n] \backslash \{1\}$:

$$g([x_1, x_2, \dots, x_n]) = g([x_1, x_{\pi(2)}, \dots, x_{\pi(n)}]). \quad (10)$$

In this context, the following insight from [(Hutter, 2020), Lemma 10] is important because it enables us to deal with equivariant sequence-to-sequence functions by looking at semi-invariant sequence-to-point functions instead:

Lemma 2.9 (Equivalence of Equivariance and Semi-invariance (Hutter, 2020))**.** *A sequence-to-sequence function $f : \mathcal{X}^n \to \mathcal{Y}^n$ is equivariant if and only if there exists a semi-invariant sequence-to-point function $g : \mathcal{X}^n \to \mathcal{Y}$ such that*

$$\begin{aligned} &f([x_1, \dots, x_n]) \\ &\quad = [g(x_1, \{x_2, x_3 \dots\}), g(x_2, \{x_1, x_3, \dots\}), \dots]. \end{aligned} \quad (11)$$

2.4. Multisymmetric Polynomials

We discuss two different proofs for the universality of Sumformer. For the continuous proof, we use multisymmetric polynomials, which we introduce now. Our definitions are based on (Briand, 2004).

Definition 2.10 (Multisymmetric Polynomial)**.** Let $\mathcal{X} \subset \mathbb{R}^d$. A (real) multisymmetric polynomial in a sequence of length n is a polynomial $p : \mathcal{X}^n \to \mathbb{R}$ in the variables $x_1^{(1)}, x_2^{(1)}, ..., x_d^{(n)}$ which is invariant in permutations of $x^{(1)}, ..., x^{(n)}$.

Definition 2.11 (Multisymmetric Power Sum)**.** A multisymmetric power sum of multidegree $\alpha = (\alpha_1, ..., \alpha_d) \in \mathbb{N}^d \backslash \{0\}$ is a multisymmetric polynomial of the form:

$$p_\alpha : \mathcal{X}^n \to \mathbb{R}, [x^{(1)}, ..., x^{(n)}] \mapsto \sum_{i=1}^{n} (x^{(i)})^\alpha \quad (12)$$

where $(x^{(i)})^\alpha = (x_1^{(i)})^{\alpha_1} \cdots (x_d^{(i)})^{\alpha_d}$.

The multisymmetric power sums are of interest because they can generate any multisymmetric polynomial. The following theorem which follows directly from [(Briand, 2004), Theorem 3 & Corollary 5] shows this relationship:

Theorem 2.12 (Multisymmetric Power Sums generate Multisymmetric Polynomials)**.** *The real multisymmetric power sums in a sequence of length n with multidegree $|\alpha| := \alpha_1 + \cdots + \alpha_d \leqslant n$ generate all real multisymmetric polynomials (in a sequence of length n), i.e. every multisymmetric polynomial p can be represented by*

$$p = \sigma(p_{\alpha^{(1)}}, ..., p_{\alpha^{(z)}}) \quad (13)$$

with a (real) polynomial σ and the multisymmetric power sums $p_{\alpha^{(1)}}, ..., p_{\alpha^{(z)}}$.

2.5. Deep sets

As discussed in Section 1.2, the concept of a Sumformer, which we introduce in section 3, is related to the concept of deep sets introduced in (Zaheer et al., 2017). We also utilize the following theorem for the discontinuous proof:

Theorem 2.13 ((Zaheer et al., 2017), Theorem 2)**.** *Let $Z = \{z_1, ..., z_M\}, z_m \in E$, E countable and $\mathcal{Z}$ be the power set of Z. A function $f : \mathcal{Z} \to \mathbb{R}$ operating on Z can be permutation invariant to the elements in Z, if and only if it can be decomposed in the form $\psi(\sum_{z \in Z} \phi(x))$, for suitable transformations ϕ and ψ.*

3. Sumformer

We now introduce the new architecture *Sumformer*. The name stems from the inherent dependence on the sum of a function evaluation of every token separately.

Definition 3.1 (Sumformer)**.** Let $d' \in \mathbb{N}$ and let there be two functions $\phi : \mathcal{X} \to \mathbb{R}^{d'}, \psi : \mathcal{X} \times \mathbb{R}^{d'} \to \mathcal{Y}$. A *Sumformer* is a sequence-to-sequence function $\mathcal{S} : \mathcal{X}^n \to \mathcal{Y}^n$ which is evaluated by first computing

$$\Sigma := \sum_{k=1}^{n} \phi(x_k), \quad (14)$$

and then

$$\mathcal{S}([x_1, \dots, x_n]) := [\psi(x_1, \Sigma), \dots, \psi(x_n, \Sigma)]. \quad (15)$$

The Sumformer architecture is simple and can be approximated with Transformers, Linformers, and Performers. The simplicity of the architecture and the ability to prove the universality of multiple architectures using it suggests that Sumformers can also be approximated by other architectures and thereby give universal approximation theorems for them.

4. Universal approximation

In this section, we give the main theorems of this paper. We first show that Sumformers are universal approximators for continuous sequence-to-sequence functions. This result can be used to give a new proof for the universality of Transformers and the first universal approximation results for Linformer and Performer.

Before continuing, we make an important assumption: For the rest of this paper, let $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^d$ and let $\mathcal{X}$ be a compact set. Note that $\mathcal{X}$ and $\mathcal{Y}$ do not need to have the same dimensionality in the following theorems. This only simplifies our notation.

4.1. Sumformer

We show two different proof ideas for the universal approximation by Sumformer.

The second relies on a local approximation with a piecewise constant function. This approximation allows us to choose the inherent dimension $d' = 1$. Hence, we are able to choose a very small attention matrix. However, due to the discontinuous structure, we need exponentially many feed-forward layers in the sequence lengths n and the token size d.

This problem can be circumvented with an approximation with continuous ψ and ϕ using multisymmetric power sums from Definition 2.11. In this case, four feed-forward layers and one attention or summing layer are sufficient. However, the inherent dimension d' scales with n^d - for a fixed d - in this case. Therefore, the related attention matrices also scale with n^d.

We investigate this trade-off further in Section 5 with numerical experiments.

Theorem 4.1 (Universal Approximation by Sumformer)**.** *For each function $f \in \mathcal{F}^n_{\text{equi}}(\mathcal{X}, \mathcal{Y})$ and for each $\varepsilon > 0$ there exists a Sumformer $\mathcal{S}$ such that*

$$\sup_{X \in \mathcal{X}^n} \|f(X) - \mathcal{S}(X)\|_\infty < \varepsilon. \tag{16}$$

Proof sketch continuous. We aim to use Theorem 2.12. Therefore, for every $i \in [d]$, we approximate coordinate i of f with an equivariant vector of polynomials $p_i : \mathcal{X}^n \mapsto \mathbb{R}^n$ with an accuracy of ϵ/d (as done in (Hutter, 2020)). This is possible using a version of the Stone-Weierstrass theorem from (Hutter, 2020). Because p_i is equivariant we can use Theorem 2.9 to represent p_i by a semi-invariant polynomial $q_i : \mathcal{X}^n \mapsto \mathbb{R}$, such that $p_i([x_1, \dots, x_n]) = [q_i(x_1, \{x_2, \dots, x_n\}), \dots, q_i(x_n, \{x_1, \dots, x_{n-1}\})$.

Now, we use Theorem 2.12 and a representation similar to (Hutter, 2020) to represent q_i using multisymmetric monomials and polynomials of multisymmetric power sums. For this, we define a function mapping to the power sums: Let $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d'}$ be the map to all d' multisymmetric monomials with order $0 < |\alpha| \leqslant n$. The sum in the Sumformer is then represented as $\Sigma = \sum_{i=1}^n \phi(x^{(i)})$. We represent q_i by

$$\psi_i(x^{(j)}, \Sigma) = \sum_{\alpha \in P} (x^{(j)})^\alpha \cdot \sigma_\alpha(\Sigma - \phi(x^{(j)})) \tag{17}$$

with $P \subseteq \mathbb{N}_0^d, |P| < \infty$ and σ_α are polynomials. Finally, by setting $\psi = [\psi_1, ..., \psi_d]$, we obtain a Sumformer $\mathcal{S}$ with $\mathcal{S}(x) = [p_1(x), ..., p_d(x)]$ which therefore also fulfills the required goodness of fit. □

Proof sketch discontinuous. Instead of approximating the equivariant function f, we approximate the semi-invariant and uniformly continuous (since $\mathcal{X}$ is compact) function g, which represents every component as described in Theorem 2.9. To be able to use Theorem 2.13 with a countable input, we approximate g with a locally constant function $\overline{g}$. The used grid is of size $(1/\delta)^{nd}$ for some $\delta > 0$, which depends on ε. The new function $\overline{g}$ is also semi-invariant.

Now, we can assign every grid point $p \in G$ a coordinate $\chi(p) = (a, b) \in [\Delta]^d \times [\Delta]^{(n-1)\times d}$ where $\Delta = \frac{1}{\delta}$. Furthermore, we can find a function $\lambda : [\Delta]^{(n-1)\times d} \to \mathbb{N}$ with a finite range which yields the same output if and only if the input sequences are permutations of each other.

In the next step, we can use Theorem 2.13 to find ϕ^* and ψ^* so that $\lambda(b) = \psi^*\left(\sum_{i=1}^{n-1} \phi^*(b_i)\right)$.

Let $q : \mathcal{X} \to [\Delta]^d$ be the function mapping tokens to the corresponding cube-coordinate. Then by defining

$$\Sigma = \sum_{i=1}^n \phi^*(q(x)) \tag{18}$$

and

$$\begin{aligned} &\psi(x_1, \Sigma) \\ &:= \overline{g}\Big(\chi^{-1}\Big(q(x_1),\ \lambda^{-1}\Big(\psi^*\big(\Sigma - \phi(x_1)\big)\Big)\Big)\Big) \end{aligned} \tag{19}$$

we yield a Sumformer with the required goodness of fit. Note that even though λ^{-1}, in general, might not be invertible, we can find an inverse of a restriction of λ to a subset of the domain such that properties necessary for our proof are given. □

4.2. Transformer

With the approximation result for Sumformers, we can now present two new proofs for the universality of Transformers. Before we give these proofs, we want to highlight the first universality theorem for Transformers from (Yun et al., 2019) and discuss the similiarities and differences.

Theorem 4.2 (Universal Approximation by Transformer (Yun et al., 2019)). *Let $\varepsilon > 0$ and let $1 \leqslant p < \infty$. Then, for any continuous, permutation-equivariant function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ with compact support, there exists a Transformer Network $\mathcal{T}$ such that*

$$\left(\int \|\mathcal{T}(X) - f(X)\|_p^p dX\right)^{1/p} \leqslant \varepsilon. \tag{20}$$

The first noticeable difference is the fact that (Yun et al., 2019) uses the L_p norm to measure the accuracy. In our setting, we aim to understand the worst-case behavior and therefore use the supremum norm. Furthermore, (Yun et al., 2019) also gives proofs for functions that are not equivariant by using positional encoding. Because the positional encoding is added only to the input and does not change any further points about the architecture, this can probably be applied also in our case.

Beyond the difference in the theorem setup, we also have a very different proof strategy. The proof in (Yun et al., 2019) relies on the concept of contextual mappings. To implement these mappings, the Transformer needs ε^{-d} many attention layers, where d is the token size and ε is the desired approximation accuracy. With our proof, we improve upon this result by showing that we only need one attention layer, which is used to represent the sum in the Sumformer.

With this information, we can now state our theorem for the universal approximation by Transformers.

Theorem 4.3 (Universal Approximation by Transformer). *For each function $f \in \mathcal{F}^n_{\text{equi}}(\mathcal{X}, \mathcal{Y})$ and for each $\varepsilon > 0$ there exists a Transformer $\mathcal{T}$ such that*

$$\sup_{X\in\mathcal{X}^n} \|f(X) - \mathcal{T}(X)\|_\infty < \varepsilon. \tag{21}$$

Proof Sketch. First, note that the weights in the attention matrix can be set to zero; this way, we can get feed-forward networks only. In the continuous case, ϕ is also continuous and can therefore be approximated with a 2-layer network by (Hornik et al., 1989). For the discontinuous proof, we know from (Yun et al., 2019) that we need $\mathcal{O}(n(1/\varepsilon)^{nd}/n!)$ many layers for the approximation.

In the following steps, we approximate the sum with an attention head. This step is equal for the continuous and discontinuous settings. However, in the discontinuous case, we can set $d' = 1$. This step is also the only step we need to investigate for the Linformer and Performer proof.

We first use a feed-forward neural network to have as input the matrix:

$$\begin{bmatrix} 1 & x_1 & \phi(x_1) & \mathbf{0}_{d'} \\ & \dots & & \\ 1 & x_n & \phi(x_n) & \mathbf{0}_{d'} \end{bmatrix} \in \mathbb{R}^{n\times 1+d+2d'} \tag{22}$$

Then, we choose

$$W_Q = W_K = [e_1, \mathbf{0}_{(1+d+2d')\times(1+d+2d')}] \tag{23}$$

with $e_1 = [1, \mathbf{0}_{d+2d'}]^\top \in \mathbb{R}^{1+d+2d'}$ such that $A = \frac{1}{n}1_{n\times n}$ and W_V such that we get together with the skip connection:

$$\begin{bmatrix} 1 & x_1 & \phi(x_1) & \Sigma \\ & \dots & & \\ 1 & x_n & \phi(x_n) & \Sigma \end{bmatrix} \in \mathbb{R}^{n\times 1+d+2d'} \tag{24}$$

We can then, in the continuous case, apply another two layers for the approximation of the continuous ψ, or we need another $\mathcal{O}(n(1/\varepsilon)^{nd}/n!)$ many feed-forward layers to approximate the ψ build in the discontinuous case. □

4.2.1. Network size

Using Sumformer, we were able to give two different constructions for the Transformer as universal approximators. We note that the construction of the attention head remains the same except for the possible choice of d'. When we approximate ϕ and ψ with smooth functions, we need a larger latent dimension d'. In the discontinuous construction, we need more layers to approximate ϕ and ψ but can approximate the function of interest f using only $d' = 1$. The same situation can be observed for the efficient Transformers as we only replace the attention heads but keep the functions ϕ and ψ from the proof of the Transformer. There might be another way of representing functions with Sumformers. However, the current proofs suggest a trade-off between the size of the latent dimension d' and the number of necessary layers. In Section 5, we test the dependence of the validation loss on the relationship of d' to the sequence length n and the token size d.

4.3. Efficient Transformers are Universal Approximators

Using the concept of Sumformer, we can show that Linformer and Performer are universal approximators for continuous functions on a compact support. We are able to utilize the proof for Transformers as the architecture is only changed in the attention head, which forms the main computational cost of Transformer. As the rest of the architecture stays the same, this part of the proof does not need to be adapted. We start with Linformer as introduced in Definition 2.5.

Theorem 4.4 (Universal Approximation by Linformer). *For each function $f \in \mathcal{F}^n_{\text{equi}}(\mathcal{X}, \mathcal{Y})$ and for each $\varepsilon > 0$ there exist $k \in \mathcal{O}(d/\varepsilon^2)$ and there exist matrices $E, F \in \mathbb{R}^{k\times n}$ and a Linformer $\mathcal{T}_{\text{Lin}}$ such that*

$$\sup_{X\in\mathcal{X}^n} \|f(X) - \mathcal{T}_{\text{Lin}}(X)\|_\infty < \varepsilon. \tag{25}$$

Proof. By Definition 2.5, Linformer $\mathcal{T}_{\text{Lin}}$ have the same architecture as Transformer $\mathcal{T}$ except for the attention head. Therefore, we can use the same construction for ψ and ϕ as in the proof of Theorem 4.2. It remains to show that we can represent the sum in the Sumformer with the linear attention head as well. We now discuss how the weight and projection matrices are chosen for the approximation. Let $E = \frac{1}{n}\mathbf{1}_{k\times n}$ and $F = \frac{1}{k}\mathbf{1}_{k\times n}$, W_Q, W_K, W_V as in Equation (23) and we get that the Linformer attention layers maps to

$$\rho((XW_Q)(EXW_K)^T)\cdot(FXW_V) = [\mathbf{0}_{n\times 1+d+d'}, \Sigma] \tag{26}$$

After applying the skip connection, we get the same output as in Equation (24) in Theorem 4.2. Therefore, we can apply the same representation for ψ and get the desired approximation. □

Now, even though the structure and idea of Performer differ a lot from Linformer, we can use a similar strategy to show the universal approximation.

Theorem 4.5 (Universal Approximation by Performer)**.** *Let $k \in \mathbb{N}$ with $k < n$. For each function $f \in \mathcal{F}^n_{\text{equi}}(\mathcal{X}, \mathcal{Y})$ and for each $\varepsilon > 0$ there exists a Performer $\mathcal{T}_{\text{Per}}$ such that*

$$\sup_{X\in\mathcal{X}^n} \|f(X) - \mathcal{T}_{\text{Per}}(X)\|_\infty < \varepsilon. \tag{27}$$

Proof. As in the proof for the Linformer attention layer we use the fact that the Performer $\mathcal{T}_{\text{Per}}$ only differs from a Transformer $\mathcal{T}$ by the choice of the attention head. Therefore, we now build a Performer attention head which is able to approximate the sum for the Sumformer.

We choose the same W_Q and W_K as in Equation (23). Next, we fix the vectors $w_1, \dots, w_k$ in a in the Performer Definition 2.6. Then, because all rows are the same and a is applied row-wise, $a(XW_Q)a(XW_K)^\top = \lambda \cdot \mathbf{1}_{n\times n}$ for some $\lambda \in \mathbb{R}$.

In contrast, to the previous proof, we need to add another feed-forward layer after the attention layer. We choose the weight matrix to be $W = \frac{1}{\lambda n} I_{(1+d+2d')}$ and the bias $b = \mathbf{0}_{1+d+2d'}$. Then, we get an output of

$$\begin{aligned} &Wa(XW_Q)a(XW_K)^\top(XW_V) + b \\ &= [\mathbf{0}_{n\times 1+d+d'}, \Sigma]^T. \end{aligned} \tag{28}$$

With the skip connection we get the desired input for ψ and are able to use the same approximation for ψ as in Theorem 4.3. □

5. Numerical Experiments

We implemented two different Sumformer architectures and tested them on approximating analytically given (i.e., non-real-world) functions. Both architectures consist of three components: one representing ϕ, one representing ψ, and the last combining the two as described in Definition 3.1.

The function ψ is represented by a Multi-layer perceptron (MLP) in both architectures. The representation of ϕ differs: The first model uses the ϕ we constructed in the proof of Theorem 4.1 (Polynomial Sumformer), whereas the second one uses an MLP again (MLP Sumformer).

Each MLP we used consisted of five hidden layers of 50 nodes. We use the ReLU activation function.

We trained our two models (using the same latent dimension d') on approximating multiple equivariant functions (assuming $\mathcal{X} = [0,1]^d$): two polynomial-type and two non-polynomial-type functions. The results (Fig.1) show that the previous results are not just theoretical: Sumformer architectures can approximate a variety of functions.

It is interesting to note that the two Sumformers perform approximately equally well on most functions we approximated (polynomial & non-polynomial type). Based on this, we observe that the construction used in the continuous proof of Theorem 4.1 is indeed able to learn our benchmark functions using gradient descent.

Furthermore, we observe that the validation loss of the Polynomial Sumformer is smoother and decreases in a more stable way than that of the MLP Sumformer. In contrast, the validation loss of the MLP Sumformer often jumps to drastically lower levels over just a few epochs and is relatively flat apart from that. This phenomenon could be explained by the interaction of the two disjoint trainable components (MLPs).

We also tested how changing the dimension d' (see Definition 3.1) in the MLP Sumformer impacts the best validation loss over a fixed number of epochs while holding n, d and the function to approximate constant. The results (Fig. 2) show - as expected - that higher dimensions d' generally lead to better approximation. Furthermore, when changing d linearly, we have to make non-linear - presumably exponential - changes to the size of d' to achieve significantly diminishing returns on further increasing d'.

This finding is particularly interesting as the continuous proof of Theorem 4.1 needs $d' = \binom{n+d}{d} - 1 = \frac{(n+d)!}{d!n!} - 1 = \prod_{i=1}^{d} \frac{n+i}{i} - 1$ in ϕ for a fixed n. This suggests that the empirical performance aligns with the theory.

6. Conclusion

We have seen that the efficient Transformers, Linformer, and Performer, are able to represent all equivariant continuous sequence-to-sequence functions on compact sets arbitrarily well. Due to the simplicity of the Sumformer

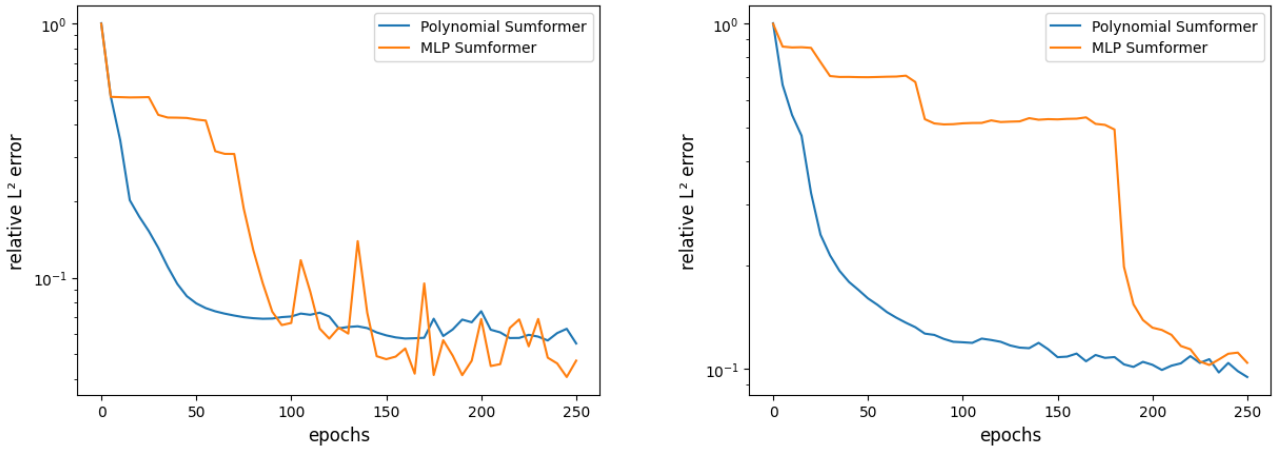

Figure 1. Examples of validation errors (every 5 epochs; relative L^2 error) of the two Sumformer models over a training run on 2000 data points with $n = 5, d = 4$. The equivariant functions were defined by semi-invariant functions of different kinds: *(left)* polynomial-type, *(right)* non-polynomial-type.

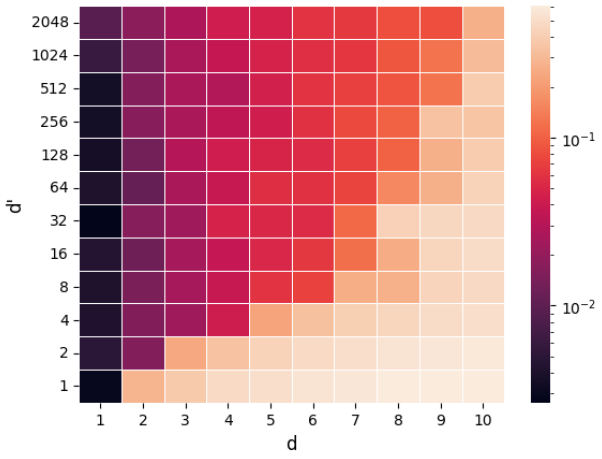

Figure 2. Development of best validation errors (relative L^2 error) over multiple training runs (2000 data points; 200 epochs; $n = 3$) with exponentially increasing latent dimension d' for ten dimensions d. The equivariant functions (for every d) were defined by the following polynomial-type semi-invariant function $g : [x^{(1)}, ..., x^{(3)}] \mapsto x^{(1)} + 7 \cdot (x^{(1)})^2 + 3 \cdot x^{(1)} \cdot (x^{(2)} + x^{(3)})^3$ where products and sums are to be understood component-wise.

architecture on which the proofs are based, it seems likely that further research can use similar techniques to show that other Transformer architectures and state space models are also universal approximators.

In addition, we offered a new proof for universal approximation by Transformer and were able to reduce the necessary number of non-zero attention layers to only one.

In our experiments, we showed that the construction from our continuous proof of universal approximation by Sumformer is tractable and indeed able to approximate given functions using gradient descent. Furthermore, our numerical results about the impact of the latent dimension d' of a Sumformer in relation to the token size d nicely relate to the required size of the latent dimension in our continuous proof.

Lastly, we note that a significant limitation of our continuous proof is that (for a fixed token size d) the size of the attention matrix scales with n^d. In other words: Although for a *fixed* model dimension d' the computational cost scales linearly in n, for achieving universal approximation the required dimension d' grows polynomially in n and correspondingly the overall computational cost. In the discontinuous setting, we were able to keep the latent dimension small but had to scale the number of feed-forward layers accordingly. It would be interesting to improve on this result and analyze the trade-off further in future research.

Acknowledgements

LT and GK acknowledge support from the German Research Foundation in the frame of the priority programme SPP 2298. SA appreciates the support by the Stanford Graduate Fellowship. GK is also grateful for partial support by the Konrad Zuse School of Excellence in Reliable AI (DAAD), the Munich Center for Machine Learning (BMBF) as well as the German Research Foundation under Grants KU 1446/31-1 and KU 1446/32-1 and under Grant DFG-SFB/TR 109, Project C09 and the Federal Ministry of Education and Research under Grant MaGriDo.

References

Barron, A. R. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14:115–133, 1994.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Bolcskei, H., Grohs, P., Kutyniok, G., and Petersen, P. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019.

Briand, E. When is the algebra of multisymmetric polynomials generated by the elementary multisymmetric polynomials? *Beiträge zur Algebra und Geometrie: Contributions to Algebra and Geometry, 45 (2), 353-368.*, 2004.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, C., Chen, Z., and Lu, J. Representation theorem for multivariable totally symmetric functions. *arXiv preprint arXiv:2211.15958*, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Eraslan, G., Avsec, Ž., Gagneur, J., and Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

Gühring, I., Kutyniok, G., and Petersen, P. Error bounds for approximations with deep relu neural networks in w s, p norms. *Analysis and Applications*, 18(05):803–859, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Hutter, M. On representing (anti) symmetric functions. *arXiv preprint arXiv:2007.15298*, 2020.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Likhosherstov, V., Choromanski, K., and Weller, A. On the expressive power of self-attention matrices. *arXiv preprint arXiv:2106.03764*, 2021.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.

Mhaskar, H. N. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8 (1):164–177, 1996.

Petersen, P. and Voigtlaender, F. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 108:296–330, 2018.

Shaham, U., Cloninger, A., and Coifman, R. R. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers, 2020.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6): 1–28, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wagstaff, E., Fuchs, F., Engelcke, M., Posner, I., and Osborne, M. A. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pp. 6487–6494. PMLR, 2019.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

Yarotsky, D. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

Yarotsky, D. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1): 407–474, 2022.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Yun, C., Chang, Y.-W., Bhojanapalli, S., Rawat, A. S., Reddi, S., and Kumar, S. $o(n)$ connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794, 2020.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

A. Proofs of the universal approximation results for Sumformer

In this section we give the details of the continuous and discontinous proofs of Theorem 4.1.

Discontinuous case. By Lemma 2.9, there exists a semi-invariant function $g : \mathcal{X}^n \to \mathcal{Y}$ such that $f(X) = [g(x_1, \{x_2, \dots, x_n\}), \dots, g(x_n, \{x_1, \dots, x_{n-1}\})]$. Since f is continuous, the component functions $f_1, \dots, f_n$ are also continuous and thus also g. The compactness of $\mathcal{X}$ implies that $\mathcal{X}^n$ is compact and therefore g is uniformly continuous. Without loss of generality, let the compact support of g be contained in $[0,1]^{n\times d}$. Then, we define a piece-wise constant function $\overline{g}$ by

$$\overline{g}(X) = \sum_{\boldsymbol{p}\in\mathcal{G}} g(\boldsymbol{p})\mathbf{1}\{X \in C_{\boldsymbol{p}}\}, \tag{29}$$

where the grid $\mathcal{G} := \{0, \delta, \dots, 1-\delta\}^{n\times d}$ for some $\delta := \frac{1}{\Delta}$ with $\Delta \in \mathbb{N}$ consists of cubes $C_{\boldsymbol{p}} = \prod_{i=1}^{n}\prod_{k=1}^{d}[\boldsymbol{p}_{i,k}, \boldsymbol{p}_{i,k}+\delta)$ with corresponding values $g(\boldsymbol{p}) \in \mathcal{Y}$ for each $\boldsymbol{p} \in \mathcal{G}$. Because g is uniformly continuous, there exists for each $\varepsilon > 0$ a $\delta > 0$ such that

$$\sup_{X\in\mathcal{X}^n} \|g(X) - \overline{g}(X)\|_\infty < \varepsilon. \tag{30}$$

We next show that that $\overline{g}$ is semi-invariant. Since g is semi-invariant, we have $g([x_1, x_{\pi(2)}, \dots, x_{\pi(n)}]) = g([x_1, x_2, \dots, x_n])$ for any permutation $\pi : [n]\backslash\{1\} \to [n]\backslash\{1\}$. With $\boldsymbol{p} = [p_1, \dots, p_n]$, we can write $\pi(\boldsymbol{p}) = [\boldsymbol{p}_1, \boldsymbol{p}_{\pi(2)}, \dots, \boldsymbol{p}_{\pi(n)}]$ and get $g(\boldsymbol{p}) = g(\pi(\boldsymbol{p}))$. Moreover, we get $X \in C_{\boldsymbol{p}} \Leftrightarrow \pi(X) \in C_{\pi(\boldsymbol{p})}$. Hence, for any $X \in C_{\boldsymbol{p}}$, we get

$$\overline{g}(X) = g(\boldsymbol{p}) = g(\pi(\boldsymbol{p})) = \overline{g}(\pi(X)). \tag{31}$$

Now, we want to to represent $\overline{g}$ using an appropriate $\mathcal{S}$. While it is trivial to match each X to its corresponding $\boldsymbol{p}$ such that $X \in C_{\boldsymbol{p}}$, it is more difficult to find the corresponding cube of X when only being able to use x_1 and the aggregated Σ.

To achieve this, we will use the following strategy: Recall that $\Delta \in \mathbb{N}$ is the number of cubes in each dimension. We can assign each grid point $\boldsymbol{p} \in \mathcal{G}$ a coordinate $\chi(\boldsymbol{p}) = (a, \boldsymbol{b}) \in [\Delta]^d \times [\Delta]^{(n-1)\times d}$. The map $\chi : \mathcal{G} \to [\Delta]^d \times [\Delta]^{(n-1)\times d}$ is bijective and the first part of the coordinate $a \in [\Delta]^d$ can be constructed from x_1 by quantizing it in each dimension. Let $q : \mathcal{X} \to [\Delta]^d$ be this quantization function such that $q(x_1) = a$.

Let us now find a way to choose ϕ and ψ such that we can reconstruct $\boldsymbol{b}$ from Σ. We can treat $\boldsymbol{b}$ as a sequence of length $n-1$ and write $\boldsymbol{b} = [b_1, \dots, b_{n-1}]$ with $b_i \in [\Delta]^d$. Since there are finitely many $\boldsymbol{b} \in [\Delta]^{(n-1)\times d}$, we can enumerate all $\boldsymbol{b}$ using a function $\lambda : [\Delta]^{(n-1)\times d} \to \mathbb{N}$. Moreover, let us choose λ to be invariant to permutations of $[b_1, \dots, b_{n-1}]$, i.e. for all permutations $\pi : [n-1] \to [n-1]$ we have $\lambda([b_1, \dots, b_{n-1}]) = \lambda([b_{\pi(1)}, \dots, b_{\pi(n-1)}])$, but we let λ always assign different values to $\boldsymbol{b}_1, \boldsymbol{b}_2$ if they are not a permutation of each other. Although this prevents λ from being injective, all cubes with the same value under λ have the same value under $\overline{g}$, due to semi-invariance, i.e. for a fixed $a \in [\Delta]^d$ and for all n in the range of λ the inverse is well defined and we can evaluate

$$\overline{g}\Big(\chi^{-1}\big(a, \lambda^{-1}(n)\big)\Big). \tag{32}$$

Now, λ is an invariant sequence-to-point function and since $[\Delta]^d$ is countable, we can utilize Theorem 2.13 (note that we use multisets of a fixed size here, to which the proof in (Zaheer et al., 2017) can be easily extended) to find $\phi^* : [\Delta]^{(n-1)\times d} \to \mathbb{R}$ and $\psi^* : \mathbb{R} \to \mathbb{N}$ such that

$$\lambda(\boldsymbol{b}) = \psi^*\left(\sum_{i=1}^{n-1}\phi^*(b_i)\right)$$

With the quantization function q we set $\phi(x) := \phi^*(q(x))$ and define

$$\Sigma = \sum_{i=1}^{n}\phi^*(q(x)). \tag{33}$$

We can than recover $\lambda(\boldsymbol{b})$ by

$$\lambda(\boldsymbol{b}) = \psi^*\left(\Sigma - \phi(x_1)\right). \tag{34}$$

Now, we can define ψ such that the related $\mathcal{S}$ is equal to $\overline{g}$:

$$\psi(x_1, \Sigma) := \overline{g}\Big(\chi^{-1}\Big(q(x_1),\ \lambda^{-1}\Big(\psi^*\big(\Sigma - \phi(x_1)\big)\Big)\Big)\Big). \tag{35}$$

Since we chose $\overline{g}$ to uniformly approximate g and thereby each component of f up to ε error, this implies that $\mathcal{S}$ uniformly approximates f up to ε error. □

Continuous case. As before we have that the compactness of $\mathcal{X}$ implies that $\mathcal{X}^n$ is compact and without loss of generality, we can assume that the compact support of f is contained in $[0,1]^{n\times d}$.

Now, for every $i \in [d]$, we approximate coordinate i of f with an equivariant vector of polynomials $p_i : \mathcal{X}^n \mapsto \mathbb{R}^n$ with an accuracy of ϵ/d (as done in (Hutter, 2020)). This is possible using a version of the Stone-Weierstrass theorem from (Hutter, 2020). Because p_i is equivariant we can use Theorem 2.9 to represent p_i by a semi-invariant polynomial $q_i : \mathcal{X}^n \mapsto \mathbb{R}$, such that $p_i([x_1, \ldots, x_n]) = [q_i(x_1, \{x_2, \ldots, x_n\}), \ldots, q_i(x_n, \{x_1, \ldots, x_{n-1}\})]$.

Now, we use Theorem 2.12 and a representation similar to (Hutter, 2020) to represent q_i using multisymmetric monomials and polynomials of multisymmetric power sums. For this, we define a function mapping to the power sums: Let

$$\phi : [0,1]^d \to \mathbb{R}^{d'}, x \mapsto \begin{pmatrix} x_1^1 x_2^0 \cdots x_d^0 \\ x_1^2 x_2^0 \cdots x_d^0 \\ \vdots \\ x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d} \\ \vdots \\ x_1^0 x_2^0 \cdots x_d^n \end{pmatrix} \tag{36}$$

where $\alpha = (\alpha_1, ..., \alpha_d)$ runs over all multidegrees with order $0 < |\alpha| \leqslant n$. The sum in the Sumformer is then represented as $\Sigma = \sum_{i=1}^n \phi(x^{(i)})$.

By Theorem 2.12 the function

$$s_j(x^{(i\neq j)}) = \sigma\left(\sum_{i\neq j} \phi(x^{(i)})\right) \tag{37}$$

with σ being a polynomial function can fit any multisymmetric polynomial in the variables $x^{(i\neq j)} := \{x^{(1)}, ..., x^{(j-1)}, x^{(j+1)}, ..., x^{(n)}\}$ perfectly.

We can therefore represent q_i by

$$\psi_i(x^{(j)}, \Sigma) = \sum_{\alpha\in P} (x^{(j)})^\alpha \cdot \sigma_\alpha(\Sigma - \phi(x^{(j)})) \tag{38}$$

with $P \subseteq \mathbb{N}_0^d, |P| < \infty$ and σ_α are polynomials.

By setting $\psi = [\psi_1, ..., \psi_d]$, we obtain a Sumformer $\mathcal{S}$ with $\mathcal{S}(x) = [p_1(x), ..., p_d(x)]$ which is able to approximate f sufficiently well. □

B. Proofs of the universal approximation results for Transformer

Now we give the detailed proof of the universality of Transformers from Theorem 4.3.

Proof. We use the triangular inequality to divide the approximation in two steps. We first approximate f by a Sumformer $\mathcal{S}$ and then show that the Sumformer can be approximated by a Transformer $\mathcal{T}$, i.e.

$$\sup_{X\in\mathcal{X}^n} \|f(X) - \mathcal{T}(X)\|_\infty \leqslant \sup_{X\in\mathcal{X}^n} \|f(X) - \mathcal{S}(X)\|_\infty + \sup_{X\in\mathcal{X}^n} \|\mathcal{S}(X) - \mathcal{T}(X)\|_\infty \tag{39}$$

For the first summand we have from Theorem 4.1 that there is a Sumformer $\mathcal{S}$ which approximates f to an accuracy of $\varepsilon/2$. The Sumformer has the inherent latent dimension d'.

We now turn to the second summand and construct a Transformer that is able to approximate the Sumformer to $\varepsilon/2$ accuracy. Transformers are constructed as described in Definition 2.4. Because of the structure with $X + \mathrm{FC}(X + \mathrm{Att}(X))$, we can set the attention for the first layers to zero. Thereby, we obtain feed-forward layers without attention.

The Transformer is then constructed as follows. We have the input $X = [x_1, \dots, x_n]^\top \in \mathcal{X}^n$ with $x_i \in \mathbb{R}^{1\times d}$ and map it with a feed-forward from the right to

$$\begin{bmatrix} x_1, x_1 \\ \cdots \\ x_n, x_n \end{bmatrix} \in \mathbb{R}^{n\times 2d}. \tag{40}$$

We can then find a two layer feed-forward network such that it acts as the identity on the first n components and approximates the function ϕ. The approximation with two feed forward layers of ϕ is possible because of the universal approximation theorem (Hornik et al., 1989). In the discontinuous setting we need more layers to approximate ϕ. Therefore, after three feed-forward layers we get

$$\begin{bmatrix} x_1, \phi(x_1) \\ \cdots \\ x_n, \phi(x_n) \end{bmatrix} \in \mathbb{R}^{n\times(d+d')}. \tag{41}$$

Before, we get to the attention layer we add one more layer from the right $\mathrm{FC} : \mathbb{R}^{d+d'} \to \mathbb{R}^{1+d+2d'}$ with

$$W = \begin{bmatrix} \mathbf{0}_{d\times 1} & I_d & \mathbf{0}_{d\times d'} & \mathbf{0}_{d\times d'} \\ \mathbf{0}_{d'\times 1} & \mathbf{0}_{d'\times d} & I_{d'} & \mathbf{0}_{d'\times d'} \end{bmatrix} \in \mathbb{R}^{(d+d')\times(1+d+2d')} \tag{42}$$

and $b = [\mathbf{1}_n, 0_{n\times(d+2d')}]$. Using these transformations, we get as output after the first step:

$$X_1 = \begin{bmatrix} 1 & x_1 & \phi(x_1) & \mathbf{0}_{d'} \\ & \cdots & & \\ 1 & x_n & \phi(x_n) & \mathbf{0}_d \end{bmatrix} \in \mathbb{R}^{n\times 1+d+2d'} \tag{43}$$

Note that these steps are the same for the efficient Transformers.

Now, we turn to the attention head to represent the sum $\Sigma = \sum_{i=1}^n \phi(x_i) \in \mathbb{R}^{d'}$. First we choose $W_Q = W_K = [e_1, \mathbf{0}_{(1+d+2d')\times(1+d+2d')}] \in \mathbb{R}^{(1+d+2d')\times(1+d+2d')}$ for $e_1 = [1, \mathbf{0}_{d+2d'}]^\top \in \mathbb{R}^{1+d+2d'}$, such that

$$A = \rho((X_1 W_Q)(X_1 W_K)^\top) = \frac{1}{n}\mathbf{1}_{n\times n}. \tag{44}$$

The matrix A will then be multiplied with $X_1 W_V$. We can choose

$$W_V = \begin{bmatrix} \mathbf{0}_{(1+d)\times(1+d+d')} & \mathbf{0}_{(1+d)\times d'} \\ \mathbf{0}_{d'\times(1+d+d')} & n\cdot I_{d'} \\ \mathbf{0}_{d'\times(1+d+d')} & \mathbf{0}_{d'\times d'} \end{bmatrix} \in \mathbb{R}^{(1+d+2d')\times(1+d+2d')}. \tag{45}$$

The output of this attention layer is

$$[\mathbf{0}_{1+d+d'}, \Sigma]^\top. \tag{46}$$

Then, we apply a residual connection and obtain

$$[1, x_i, \phi(x_i), \Sigma]^\top. \tag{47}$$

Last, we implement ψ. For the discontinuous case, we first compute $q(x_i)$. Then, we map a finite set of values to another finite set of values for which we can use Lemma 7 in (Yun et al., 2019). Hence, we need to add another $O(n\left(\frac{1}{\varepsilon}\right)^{dn}/n!)$ feed-forward layers for the approximation of ψ. In the continuous case this can be avoided because of the continuity of ψ, we can approximate it with the universal approximation theorem (Hornik et al., 1989) with 2 feed-forward layers. □

C. Deep Sets

Sumformers are related to the concept of deep sets introduced in (Zaheer et al., 2017). For the discrete proof we use Theorem 2.13. However, there is also a version for uncountable inputs which we highlight here:

Theorem C.1 ((Zaheer et al., 2017), Theorem 9)**.** *Assume the elements are from a compact set in $\mathbb{R}^d$, i.e. possibly uncountable, and the set size is fixed to M. Then any continuous function operating on a set X, i.e. $f : \mathbb{R}^{d\times M} \to \mathbb{R}$ which is permutation invariant to the elements in X can be approximated arbitrarily close in the form of $\psi(\sum_{x\in X} \phi(x))$, for suitable transformations ϕ and ψ.*

The fundamental differences of the previous theorem to our work are that we consider *equivariant*, continuous sequence-to-*sequence* functions. This difference is the reason why we need a second parameter in ϕ.