What do Graph Neural Networks learn? Insights from Tropical Geometry

Tuan Anh Pham
School of Mathematics
University of Edinburgh
Edinburgh, United Kingdom
`tuan.pham@ed.ac.uk`

Vikas Garg
YaiYai Ltd and Aalto University
`vgarg@csail.mit.edu`

Abstract

Graph neural networks (GNNs) have been analyzed from multiple perspectives, including the WL-hierarchy, which exposes limits on their expressivity to distinguish graphs. However, characterizing the class of functions that they learn has remained unresolved. We address this fundamental question for message passing GNNs under ReLU activations, i.e., the de-facto choice for most GNNs.

We first show that such GNNs learn tropical rational signomial maps or continuous piecewise linear functions, establishing an equivalence with feedforward networks (FNNs). We then elucidate the role of the choice of aggregation and update functions, and derive the first general upper and lower bounds on the geometric complexity (i.e., the number of linear regions), establishing new results for popular architectures such as GraphSAGE and GIN. We also introduce and theoretically analyze several new architectures to illuminate the relative merits of the feedforward and the message passing layers, and the tradeoffs involving depth and number of trainable parameters. Finally, we also characterize the decision boundary for node and graph classification tasks.

1 Introduction

Message passing has been a cornerstone of machine learning, from inference in graphical models [1, 2] to embedding of graphs [3, 4, 5, 6, 7]. Message passing neural networks (MPNNs) are easy to implement, and can handle large scale, heterogeneous, and dynamic real world data effectively; so continue to be an active area of research [8, 9, 10]. Indeed, several popular GNNs are usually cast and implemented as MPNNs, see e.g., [11, 12, 13, 14, 15].

Unsurprisingly, GNNs as MPNNs have been analyzed from multiple theoretical perspectives. A major theme is inspired by connections to the so-called 1-WL (Weisfeiler-Leman) test for group isomorphism and its higher order extensions, where nodes in a (hyper-)graph repeatedly refine their *colors* based on the messages from their neighbors [16]. MPNNs are known to be bounded in power according to the WL-hierarchy [15, 17, 18], implying their inability to distinguish some non-isomorphic graphs. Unlike WL that strives to expose what GNNs *cannot* do, here we seek to unravel what they *can*.

Notably, the WL formalism implicitly relies on injective hash functions; in contrast, most successful GNNs typically use ReLU activations that violate injectivity. Bounding the expressivity of such GNNs via their injective surrogates is theoretically valid; however, it does not illuminate what they learn. Indeed, several fundamental questions remain elusive for these practical models; e.g., (a) what class of functions can they represent, (b) how does their expressivity vary with the choice of message aggregation and update functions, (c) what complexity tradeoffs (e.g., in terms of the number of

Theoretical contributions of this work	
Characterizing the class of functions learned by ReLU MPNNs:	
Equivalence with ReLU FNNs, TRSMs and CPLMs	Proposition 1
Estimating the number of linear regions, and complexity tradeoffs:	
First general lower bound for ReLU MPNNs	Theorem 3
First general upper bound for ReLU MPNNs	Theorem 4
Max aggregation has greater geometric complexity than sum	Proposition 5
Recovery of existing upper bounds for FNN and GCN	Corollary 1, 2
New upper bounds for GraphSAGE and GIN	Corollary 3, 4
New ReLU MPNNs and complexity tradeoffs:	
New architectures that can all learn CPLMs, and their tradeoffs	Proposition 6
Characterizing the decision boundary:	
Decision boundary of ReLU MPNNs for graph classification	Proposition 7
Decision boundary of ReLU MPNNs for node classification	Proposition 8

Figure 1: **Overview of our results**. Formulating ReLU MPNNs as tropical geometric objects allows us to shed light on several important aspects where WL falls short.

layers or parameters) exist for models of comparable expressivity, and (d) what decision boundary emerges for node and graph classification tasks?

We appeal to tropical algebra and geometry to address these questions in the context of ReLU MPNNs. Specifically, casting these networks as tropical geometric objects, we analyze their ability to represent different weighted sums of *tropical monomials* (the basic objects of interest in tropical algebra, akin to monomials for standard polynomials) at different nodes. We first show that the family of functions represented by these networks is precisely the family of *tropical rational signomial maps* (TRSMs). Since TRSMs are known to be equivalent to ReLU FNNs [19], this reveals an equivalence between ReLU MPNNs and ReLU FNNs: they represent exactly the same class of functions, namely, continuous piecewise linear maps (CPLMs).

This equivalence in terms of expressivity does not, however, translate into parity in *efficiency*, e.g., as quantified in terms of their respective requirements for the number of layers and trainable parameters to represent an arbitrary TRSM. Indeed, MPNNs have some strengths and limitations relative to FNNs. On the positive side, MPNNs benefit from parallel computation and streamlined communication between the nodes and their neighbors. On the flip side, MPNNs are constrained by permutation-equivariant layers that employ permutation-invariant aggregation operators [20], which impedes their ability to represent arbitrary combinations of tropical monomials.

In order to better understand the efficiency of different MPNN architectures, we investigate their *geometric complexity*, the number of linear regions that they can distinguish. Towards that end, we establish the first general upper and lower bounds on the geometric complexity of ReLU MPNNs, recovering the existing results [19, 21] for FNNs and graph convolutional networks (GCNs) as special cases. Importantly, we also unravel new results for two of the most prominent MPNNs whose complexity was previously unknown, namely, GraphSAGE [13] and GIN [15].

A particularly attractive aspect of our bounds is manifested in segregation of the contribution from different components, such as aggregation and update steps. In particular, they reveal that selecting *coordinate-wise max* as the aggregation operator affords greater geometric complexity than *sum* for ReLU MPNNs. In order to provide further insights about various complexity tradeoffs, we introduce and analyze four novel MPNN abstractions. Notably, they can all represent arbitrary TRSMs, i.e., they are as expressive as ReLU FNNs, but differ in terms of layers as well as total learnable parameters in their corresponding architectures. Fundamentally, we expose a general trend about the relative merits of the feedforward and message passing paradigms: fewer layers for MPNNs (particularly with coordinate-wise max aggregation) but fewer trainable parameters for FNNs.

Finally, we study the decision boundaries for node and graph classification, uncovering the underlying connections with *tropical hypersurfaces*; i.e., the set of points where two or more tropical monomials achieve the same value.

We summarize our key contributions in Figure 1. We now proceed to reviewing some related works.

1.1 Related work

Tropical Geometry and Machine Learning. Tropical geometry [22, 23] provides tools to study the algebraic geometry and combinatorics of continuous piecewise linear functions, and finds several applications (e.g., in optimization). Two seminal works [19, 24] initiated the analysis of deep learning models via tropical geometry, establishing the link between ReLU FNNs and tropical rational functions. The connection was then extended to maxout-layers in [25]. The decision boundaries of FNNs through a tropical lens were studied in [26]. Other aspects of deep neural networks have also be analyzed [27, 28, 29, 30, 31, 32]. We refer to [33] for a survey on the current use of tropical geometry in deep learning.

Expressivity and WL. Much work on GNNs has been inspired by noticing the parallels between MPNNs and the WL test for isomorphism [15, 34, 35]. Standard MPNNs are no more powerful than 1-WL (equivalently, 2WL), so higher order models that consider tuples of vertices have been proposed [17, 18]. Several adaptations of WL such as geometric WL [36], cellular WL [37],temporal WL [10], and persistent WL [38] have been introduced in different contexts. Limited expressivity and symmetry considerations have led to the design of new MPNN architectures [15, 39, 40, 41, 42, 43], inclusion of different types of features [39, 44], and integration with topological descriptors [45, 46, 47]. Other notions of expressivity have also been proposed, e.g., equivariant polynomials [48] and homomorphism counts [49].

Other results about GNNs. GNNs as MPNNs are also known to be prone to information bottlenecks [50, 51, 52], oversmoothing and oversquashing [53], and heterophily [54, 55]. Some works have also established their inability to count substructures, compute graph properties, or learn topology [49, 56, 57]; connections to algebraic topology [58], biconnectivity [59], and communication complexity [60, 61], behavior in overparametrized regimes [62, 63]; power of recursion in counting substructures [64]; benefits of positional encoding [65]; and ability to generalize and achieve universal approximation [56, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75].

However, neither expressivity in terms of WL nor these other results characterize the class of functions learned by GNNs, and in this work we bridge this glaring gap for ReLU MPNNs.

Complexity of Neural Networks. The complexity of deep neural networks is typically studied in terms of the number of linear regions [76, 77, 78, 79], and the intricacy of decision boundary [19], though other notions such as trajectory length [80] have also been considered recently. The number of linear regions quantifies the flexibility of the function class, thus bounding the number of regions relates closely to both expressivity and generalization [79]. To our knowledge, only the geometric complexity upper bounds for GCNs are known in the context of GNNs [21]. We provide the first general lower and upper bounds for ReLU MPNNs, establishing complexity of popular architectures such as GraphSage and GIN, and recovering the results for GCNs and FNNs as special cases.

2 Preliminaries

2.1 Message Passing Neural Networks (MPNNs)

A general T-layer MPNN χ is defined as a sequence of layers $\{\varphi^{(t)}\}_{t=1}^{T}$, and takes as input a (directed) graph $G = (V, E, X)$ with vertices $v \in V$, edges $(u, v) \in E \subseteq V \times V$, and features $X = (X_V, X_E)$ consisting of node attributes $X_V \in \mathbb{R}^{|V|\times d_0}$ and edge attributes $X_E \in \mathbb{R}^{|E|\times d'_0}$. It produces as output a refined *embedding* for each vertex and edge. Formally, χ acts on X as

$$\chi(X) = \varphi^{(T)} \circ ... \circ \varphi^{(1)}(X) \text{ , where each } \varphi^{(t)} : \mathbb{R}^{|V|\times d_{t-1}} \times \mathbb{R}^{|E|\times d'_{t-1}} \to \mathbb{R}^{|V|\times d_t} \times \mathbb{R}^{|E|\times d'_t}$$

is *permutation-equivariant* [20]. Subsequently, depending on the task, the embedding for each node produced by χ is fed into another neural network, e.g., $\eta : \mathbb{R}^{d_T} \to \mathbb{R}^{out}$ for node classification or regression (and similarly for edges). An additional readout step $\varphi_{\text{Readout}} : \mathbb{R}^{|V|\times d_T} \times \mathbb{R}^{|E|\times d'_T} \to \mathbb{R}^{d_G}$ is usually required to obtain a single vector from the output of χ for graph-level prediction.

We thus proceed to describing the working of each layer $\varphi^{(t)}$ that yields a representation $h_v^{(t)} \in \mathbb{R}^{d_t}$ for each vertex $v \in V$ and a representation $e_{uv}^{(t)} \in \mathbb{R}^{d'_t}$ for each edge $(u, v) \in E$. Assuming node embeddings are updated before edge embeddings (the converse works analogously), we have

$$\begin{aligned} h_v^{(t)} &= \varphi^{(t)}_{\text{Update}}(h_v^{(t-1)}, m_v^{(t)}), \quad \text{where} \\ m_v^{(t)} &= \varphi^{(t)}_{\text{Agg}}(h_v^{(t-1)}, \{\{h_u^{(t-1)}, e_{uv}^{(t-1)}, e_{vu}^{(t-1)} | u \in \text{Ne}(v)\}\}) \;. \end{aligned}$$

Aggregation functions $\{\varphi^{(t)}_{Agg}\}$ are typically *permutation-invariant*, and we use an operator $\square$ such as sum, average, or coordinate-wise max/min to combine the *messages* [15, 81]:

$$\varphi^{(t)}_{Agg}(h_v^{(t-1)}, \{\{h_u^{(t-1)}, e_{uv}^{(t-1)}, e_{vu}^{(t-1)}\}\}_{u\in\text{Ne}(v)}) = \phi_1^{(t)}(\square^{(t)}_{u\in\text{Ne}(v)} \phi_2^{(t)}(h_v^{(t-1)}, h_u^{(t-1)}, e_{uv}^{(t-1)}, e_{vu}^{(t-1)})), \tag{1}$$

where $\{\phi_1^{(t)}\}$ and $\{\phi_2^{(t)}\}$ are usually implemented as FNNs, $\text{Ne}(v)$ denotes the set of neighboring vertices of v, and $\{\{\cdot\}\}$ denotes a multiset. When there is no confusion, we usually write $\square^{(t)}$ instead. Let $L_1^{(t)}$ and $L_2^{(t)}$ denote, respectively, the number of layers in $\phi_1^{(t)}$ and $\phi_2^{(t)}$. We use $n_1^{t,\ell}$ (and $n_2^{t,\ell}$) to denote the dimension of layer ℓ in $\phi_1^{(t)}$ (and $\phi_2^{(t)}$). On the other hand, the *update functions* take the form

$$\varphi^{(t)}_{\text{Update}}(h_v^{(t-1)}, m_v^{(t)}) = \sigma^{(t)}_{\text{Update}}(W^{(t)}_{\text{self}} h_v^{(t-1)} + W^{(t)}_{\text{neigh}} m_v^{(t)}).$$

Henceforth, we focus on sum and coordinate-wise max, since coordinate-wise min and average can be obtained from coordinate-max and sum respectively. We shall also number the vertices in V with $A_1, ..., A_{|V|}$ and edges in E with $e_1, ..., e_{|E|}$, and use the lexical order for edges, i.e.,
$e_{uv} < e_{u'v'} \Leftrightarrow u < u'$ or $(u = u'$ and $v < v')$.

2.2 Tropical Algebra

Here we adopt the notation from [19]. Let $\mathbb{T} = \mathbb{R} \cup \{-\infty\}$ be an extended set of real numbers. We equip $\mathbb{T}$ with two binary operators **tropical sum** $\oplus$ and **tropical multiplication** $\odot$:

$$\begin{aligned} &a \oplus b = \max\{a, b\} \;; \quad a \odot b = a + b \;; \\ &a \oplus -\infty = -\infty \oplus a = a \;; \quad a \odot -\infty = -\infty \odot a = -\infty \;, \end{aligned}$$

where $\max$ and $+$ are the usual operators in $\mathbb{R}$. Thus, $(\mathbb{T}, \oplus, \odot)$ is a semi-ring with additive identity $-\infty$ and multiplicative identity 0. We also define the **tropical power** $x^{\odot a}$ for each $x \in \mathbb{R}$:

$$x^{\odot a} = \begin{cases} x \odot ... \odot x = a \cdot x & \text{if } a \in \mathbb{N} \\ (-x)^{\odot(-a)} & \text{if } a \in \mathbb{Z} \setminus \mathbb{N} \end{cases} , \quad -\infty^{\odot a} = \begin{cases} \infty & \text{if } a \in \mathbb{N} \setminus \{0\} \\ 0 & \text{if } a = 0 \end{cases} ;$$

where $\cdot$ is the standard product over $\mathbb{R}$, and often abbreviate $x^{\odot a}$ to x^a without inducing any confusion. A **tropical monomial** in m variables takes the form $c \odot x_1^{a_1} \odot ... \odot x_m^{a_m}$ for $c \in \mathbb{T}$ and $a_1, ..., a_m \in \mathbb{N}$, and is often denoted by multi-index shorthand cx^α, where $\alpha = (a_1, \ldots, a_m) \in \mathbb{Z}^m$ and $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$. Note that this may be interpreted as the affine combination $x^\top \alpha + c$.

A **tropical polynomial** $f(x) = c_1 x^{\alpha_1} \oplus ... \oplus c_r x^{\alpha_r}$ is a finite tropical sum of tropical monomials, and amounts to a max over finitely many terms, i.e., $f(x) = \max_{i=1}^r \{x^\top \alpha_i + c_i\}$. Without loss of generality, we assume $\alpha_i \neq \alpha_j$ for $i \neq j$ since we can combine monomials with the same α into one. A **tropical rational function** $f \oslash g(x) = f(x) - g(x)$ is the difference between two tropical polynomials. A map $F : \mathbb{R}^m \to \mathbb{R}^p$ with each component a tropical polynomial [resp. tropical rational function] is called a **tropical polynomial map** [resp. **tropical rational map**], and belongs to the set $Pol(m, p)$ [resp. $Rat(m, p)$].

When we allow $\alpha_i \in \mathbb{R}^m$ instead of restricting its components to integers ($\alpha_i \in \mathbb{Z}^m$), we obtain a **tropical signomial function/map** [resp. **tropical rational signomial function/map**] instead of a tropical polynomial/map [resp. tropical rational function/map]. It is known that each tropical rational signomial map (TRSM) is a continuous piecewise linear map (CPLM) and vice versa [19].

3 Tropical Algebra of MPNNs

In this section, we characterize the class of functions learned by ReLU MPNNs, whose activation functions for both nodes and edges (i.e., those used for η, $\phi_1^{(t)}$, $\phi_2^{(t)}$, $\sigma_{\text{Update}}^{(t)}$, etc.) are of the form:

$$\sigma^{(l)}(x) = \max\{x, t^{(l)}\}, \text{where } t^{(l)} \in (\mathbb{R} \cup -\infty)^{n_l} \; .$$

In particular, note that $\sigma^{(l)}(x) = x$ when $t^{(l)} = -\infty$. In contrast, $t^{(l)} = 0$ purges all negative inputs.

Let $\mathcal{F}_{\text{ReLU MPNN}}, \mathcal{F}_{\text{ReLU FNN}}, \mathcal{F}_{\text{CPLM}}, \mathcal{F}_{\text{TRSM}}$ be the set of functions represented by all ReLU MPNNs, ReLU FNNs, CPLMs and TRSMs respectively. [19] established the equivalence of ReLU FNNs, CPLMs and TRSMs.

Lemma 1 ([19]). $\mathcal{F}_{\textit{ReLU FNN}} = \mathcal{F}_{\textit{CPLM}} = \mathcal{F}_{\textit{TRSM}}$.

We will now extend this result to establish equivalence with $\mathcal{F}_{\text{ReLU MPNN}}$. Equating ReLU MPNNs with CPLMs is rather nuanced, since nodes in each layer share the weights. Therefore, we employ two reductions showing (1) every ReLU FNN can be cast as a ReLU MPNN, and (2) every ReLU MPNN, in turn, can be expressed as a TRSM.

Proposition 1. *[Equivalence of ReLU MPNNs, ReLU FNNs, TRSMs and CPLMs]* $\mathcal{F}_{\textit{ReLU MPNN}} = \mathcal{F}_{\textit{ReLU FNN}} = \mathcal{F}_{\textit{CPLM}} = \mathcal{F}_{\textit{TRSM}}$*. In other words, the following families are equivalent (with* $m = |V|d + |E|d'$ *and* $p = |V|d_{\textit{out}} + |E|d'_{\textit{out}}$*).*

1. *ReLU MPNNs* $\chi : \mathbb{R}^{|V| \times d} \times \mathbb{R}^{|E| \times d'} \to \mathbb{R}^{|V| \times d_{\textit{out}}} \times \mathbb{R}^{|E| \times d'_{\textit{out}}}$*;*
2. *Tropical rational signomial maps (TRSMs)* $F \oslash G : \mathbb{R}^m \to \mathbb{R}^p$ *;*
3. *Continuous piecewise linear maps (CPLMs)* $\psi : \mathbb{R}^m \to \mathbb{R}^p$ *;*
4. *ReLU FNNs* $\nu : \mathbb{R}^m \to \mathbb{R}^p$*.*

Remark 1. *In contrast to the WL test, which exposes the limitations of MPNNs via injective hash functions, Proposition 1 characterizes the class of functions that can be represented by MPNNs with ReLU activations (that are non-injective). Note, however, that Proposition 1 does not quantify how effective ReLU MPNNs are in representing CPLMs. Moreover, the equivalence between ReLU MPNNs and ReLU FNNs (according to Proposition 1) does not explain the observed empirical discrepancy between ReLU MPNNs and ReLU FNNs in practice. This motivates our subsequent analysis and results (Section 5 and Table 1), which investigate the benefits of ReLU MPNNs in terms of both the number of learnable parameters and the number of layers required to represent the same CPLM.*

4 Geometric Complexity of ReLU MPNNs

We now invoke tools from tropical algebraic geometry to study the geometric complexity of ReLU MPNNs, and provide some insights into the model architecture. Following [19], for a CPLM $f : \mathbb{R}^m \to \mathbb{R}^p$, we define its *linear degree* $\mathcal{N}(f)$ to be K, where K is the least number of connected regions Ω_k of $\mathbb{R}^m$ such that the restriction $f|_{\Omega_k}$ is affine. Equivalently, following [82], we can also define $K = \mathcal{N}(f)$ as follows: $f : \mathbb{R}^m \to \mathbb{R}^p$ is a CPLM if f is continuous and there exists a set $\{f_k : k \in \{1, ..., K\}\}$ of affine functions and maximal connected subsets $(\Omega_k)_{k=1}^K$ satisfying the following conditions:

$$\Omega_i \cap \Omega_j = \emptyset; \quad \bigsqcup_{k=1}^{K} \Omega_k = \mathbb{R}^m; \quad f|_{\Omega_k} = f_k.$$

Similarly, we define the *convex degree* $\mathcal{N}_c(f)$ where we require additionally that Ω_k is convex. We further define $\mathcal{N}_c(f|m')$ as the maximum convex degree across restrictions of f to different m'-dimensional affine subspaces of $\mathbb{R}^m$, $m' \le m$. We will analyze general upper and lower bounds on $\mathcal{N}(\chi)$ for a ReLU MPNN χ. Recall our setting of MPNN layers from Section 2.1 consisting of $\phi_{\text{Agg}}^{(t)}$ and $\phi_{\text{Update}}^{(t)}$. Note that $n_1^{t,l}$ and $n_2^{t,l}$ denote the intermediate dimensions in $\phi_1^{(t)}$ and $\phi_2^{(t)}$. Let $\tilde{d}_t$ and $\bar{d}_t$ be the output dimension of $\phi_2^{(t)}$ and $\phi_1^{(t)}$, thus $n_2^{t,L_2^{(t)}} = \tilde{d}_t$.

To analyze the number of linear region for a ReLU MPNN χ, we make a simplifying assumption that all input graphs to χ have the same graph structure G. In particular, we denote the sum of degrees of all vertices in G by D, and the maximum degree by S. A key step in our analysis of geometric complexity is building a ReLU FNN that can be applied on the vectorized input $H^{(t-1)}$ of all node embeddings $h_v^{(t-1)}$. Using the notation indicated in Equation 1, we form a ReLU FNN $\Phi_2^{(t)}$ that achieves the same effect as $\phi_2^{(t)}$ for every adjacent node embedding. The aggregation operator can be seen as either a matrix multiplication for sum, or a FNN $\Phi_3^{(t)}$ for max aggregation - the difference between these two cases will be discussed in Section 4.3. Similarly, we form a ReLU FNN $\Phi_1^{(t)}$ for $\phi_1^{(t)}$, which in combination with $\Phi_2^{(t)}$ and $\Phi_3^{(t)}$ can be seen as a ReLU FNN $\Phi_{\text{Agg}}^{(t)}$.

Our next proposition is important in the analysis for geometric complexity of χ, as it relates the geometric complexity of the model up to the $t+1$-th layer to that of the $t-$th layer, and the update as well as the message components. We define the vectorized concatenated embedding of all the vertices in the t-th layer to be $H^{(t)} \in \mathbb{R}^{|V|d_t}$ and show that the result of t-th message aggregation step can be written as a result of a FNN $\Phi_{\text{Agg}}^{t}(H^{(t)})$. Modifying any aggregation or update component thus will affect the bound and in particular, the choice of aggregation operator has an impact on the geometric complexity (which we will discuss in Section 4.3).

Proposition 2. *[Recursive formula for geometric complexity]*

$$\mathcal{N}(\varphi^{(t+1)}) \leq \mathcal{N}_c(\varphi^{(t+1)}) \leq \mathcal{N}_c(\varphi_{\textit{Update}}^{(t+1)} || V|(d_t + \tilde{d}_{t+1}))\, \mathcal{N}_c(\Phi_{\textit{Agg}}^{(t+1)} ||V|d_t)\, \mathcal{N}_c(\varphi^{(t)}). \tag{2}$$

The ideas and proofs for Proposition 2 build on [19], and the details can be found in the Appendix.

4.1 Lower bound of geometric complexity

We note that the lower bound on the linear degree of χ depends heavily on the choice of weights and biases; e.g., setting their value to 0 trivially results in $\mathcal{N}(\chi) = 0$. Thus, a lower bound on the geometric complexity (i.e., maximal linear degree) is of greater interest. In this subsection, we will provide a general lower bound for the maximum number of linear regions for a ReLU MPNN, building on the work of [76].

Theorem 3. *[Lower bound on the maximum number of linear regions] Assume for all t, l, we have $n_1^{t,l}, n_2^{t,l} \geq d_0$ and let $n_{1,d_0}^{t,l} = \left\lfloor \frac{n_1^{t,l}}{d_0} \right\rfloor^{d_0}$ and $n_{2,d_0}^{t,l} = \left\lfloor \frac{n_2^{t,l}}{d_0} \right\rfloor^{d_0}$ then the maximum number of linear regions of functions computed by any ReLU MPNN is lower bounded by*

$$S^{t_0} \frac{\left(\prod_{t=1}^{T} \left(\prod_{l=1}^{L_1^{(t)}} n_{1,d_0}^{t,l} \prod_{l=1}^{L_2^{(t)}} n_{2,d_0}^{t,l}\right)\right)}{n_{1,d_0}^{T,L_1^{(T)}}} \sum_{j=0}^{d_0} \binom{d_T}{j},$$

where t_0 is the number of MPNN layer having max as aggregation operator and for each layer t, the index l runs through every layer in $\phi_2^{(t)}, \phi_1^{(t)}$.

We now sketch some intuition about this result. We can add to $\Phi_1^{(t)}$ (constructed in Algorithm 3 in the Appendix) an initial layer to calculate $\square^{(t)} = \sum$, while the aggregation $\square^{(t)} = \max$ can be seen as a FNN-layer with rank S max activation, thus we can identify S input regions (indicated in red). By setting $\varphi_{\text{Update}}^{(t)}(h_v^{(t-1)}, m_v^{(t)}) = m_v$, we can express the whole MPNN χ as a FNN applied to input X (details in Algorithm 1, 2 and 3 in the Supplementary). We build on the analysis in [76] to construct intermediate layers that identify $\left\lfloor \frac{n^{t,l}}{d_0} \right\rfloor^{d_0}$ input regions. Our procedure amounts to sequentially folding the input space until the last layer (indicated in blue), and then replicating the hyperplane arrangement in the last layer (indicated in green).

4.2 Geometric complexity - Aggregation and Update steps

We now provide a general upper bound for the geometric complexity of ReLU MPNNs when all the weights take integer values - this assumption is mild and holds without loss of generality (details in the Appendix). We call these models *integer-weighted ReLU MPNNs*. A result similar to Proposition 1 establishes the equivalence of integer-weight ReLU MPNNs with tropical rational maps.

For our analysis, we require a technical condition that the network "does not shrink" the representation in the following sense: each intermediate dimension $n_1^{t,l}$ and $n_2^{t,l}$ of $\phi_1^{(t)}$ and $\phi_2^{(t)}$ should be sufficiently large; and the dimension of the new embedding is at least the dimension of the aggregated message plus the dimension of the previous embedding. These conditions are standard in the analysis using tropical geometry, see e.g., [19].

Theorem 4. *[Upper bound on the geometric complexity] Let $\chi : \mathbb{R}^{|V|d_0} \to \mathbb{R}^{|V|d_T}$ be an integer-weight ReLU MPNN. If $\varphi^{(t)}$ satisfies the following conditions for all MPNN layer $t = 1, .., T$*

- $n_2^{t,l} \geq \frac{D}{|V|} d_{t-1}$ *for all* $l = 1, ..., L_2^{(t)}$;
- $n_1^{t,l} \geq d_{t-1}$ *for all* $l = 1, ..., L_1^{(t)}$;
- $n_1^{t,L_1^{(t)}} + d_{t-1} \leq d_t$;

then the linear degree of $H^{(T)}$ is at most

$$\prod_{t=1}^{T} \underbrace{\left(\prod_{l=1}^{L_1^{(t)}-1} \sum_{i=0}^{|V|d_{t-1}} \binom{|V|n_1^{t,l}}{i} \right)}_{\text{from } \Phi_1^{(t)}} \underbrace{\left(\prod_{l=1}^{L_2^{(t)}-1} \sum_{i=0}^{|V|\times d_{t-1}} \binom{Dn_2^{t,l}}{i} \right)}_{\text{from } \Phi_2^{(t)}} \underbrace{\left(\sum_{i=0}^{|V|(\tilde{d}_t+d_{t-1})} \binom{|V|d_t}{i} \right)}_{\text{from } \varphi_{Update}^{(t)}} \mathcal{N}_c(\square^{(t)}),$$

where $\mathcal{N}_c(\square^{(t)}) \leq \begin{cases} 1 & \text{if } \square^{(t)} \text{ is sum,} \\ \frac{1}{2}(8S)^{D\tilde{d}_t} & \text{if } \square^{(t)} \text{ is coordinate-wise max/min .} \end{cases}$

We emphasize that, to the best of our knowledge, this the first upper bound for general ReLU message passing architectures. Furthermore, we recover the upper bounds for FNNs and GCNs (with ReLU activations and integer-weights) established in [19] and [21] respectively as special cases.

Corollary 1. *The linear degree of an integer-weight FNN is at most* $\prod_{t=1}^{T} \left(\sum_{i=0}^{d} \binom{d_t}{i} \right)$.

Corollary 2. *The linear degree of a GCN χ with T hidden layers, ReLU activation, sum aggregation and integer weight is at most* $\prod_{t=1}^{T} \left(\sum_{i=0}^{|V|d} \binom{|V|d_t}{i} \right)$.

On the other hand, we obtain new bounds for popular GNN models, particularly **GraphSAGE**[13] and **GIN**[15]. For **GraphSAGE**, note that the normalization steps do not change the linear degree.

Corollary 3. *Let $\tilde{d}_t$ be the output dimension of Aggregate$_t$ in GraphSAGE [13]. If $d_t \geq d_{t-1} + \tilde{d}_t \geq 2d_{t-1}$ for all MPNN layers $t = 1, ..., T$, then the linear degree of integer-weighted ReLU GraphSAGE described in [13] is upper bounded by*

$$\mathcal{N}_c(\varphi^T) \dots \mathcal{N}_c(\varphi^1),$$

where

$$\mathcal{N}_c(\varphi^t) \leq \begin{cases} \sum_{i=0}^{2|V|d_{t-1}} \binom{|V|d_t}{i} & \text{if the aggregation step is mean,} \\ \left(\sum_{i=0}^{|V|\tilde{d}_{-1}} \binom{|D|\tilde{d}_t}{i} \right) \left(\sum_{i=0}^{|V|(\tilde{d}_t+d_{t-1})} \binom{|V|d_t}{i} \right) & \text{if the aggregation step is pooling.} \end{cases}$$

Corollary 4. *Node embedding of integer-weighted ReLU GIN can be written as [15, Equation 4.1]. Let $n_{t,i}$ be the dimension of each of the intermediate layer in the $MLP^{(t)}$, then its linear degree*

$$\mathcal{N}_c(\chi) \leq \prod_{t=1}^{T} \left(\prod_{l=1}^{L^{(t)}-1} \sum_{i=0}^{|V|d_t} \binom{|V|n^{t,l}}{i} \right).$$

4.3 Coordinate-wise max vs. sum for aggregation

Proposition 5. *[Coordinate-wise max has greater geometric complexity than sum]*

$$\begin{aligned} & \mathcal{N}(\square^{(t)}) = 1 & & \text{if } \square^{(t)} = \textstyle\sum, \\ S^{\min\{|V|,D\}\tilde{d}_t} \leq \ & \mathcal{N}(\square^{(t)}) \leq \min\left\{ \textstyle\sum_{i=0}^{|D|d_{t+1}} \binom{S^2|V|\tilde{d}_t}{i}, S^{|V|\tilde{d}_t} \right\} & & \text{if } \square^{(t)} = \max. \end{aligned}$$

To establish Proposition 5 we again adapt [76]. Note that if $D \geq |V|$ (the graph is not too sparse, and we have enough messages between the vertices), we have that $\mathcal{N}(\square^{(t)}) = S^{|V|\tilde{d}_t}$ if $\square^{(t)} = \max$. In that case, the geometric complexity will grow polynomially with S (maximum degree).

Remark 2. *Interestingly, in [15], the authors point out that if ϕ_1 and ϕ_2 are injective, then χ is as powerful as the WL test. Under that assumption, coordinate-wise is less "expressive" than mean, which is less "expressive" than sum.*

In contrast, ϕ_1 and ϕ_2 are not injective in our case (since ReLU activation is not injective), max is more "expressive" (as measured by the notion of geometric complexity), thus providing another novel insight. Here, the connectivity of the graph plays a particularly important role.

5 New ReLU MPNNs architectures and complexity tradeoffs

Note that while both ReLU FNNs and ReLU MPNNs learn TRSMs/CPLMs (Proposition 1), they might differ vastly in terms of their resource requirements (e.g., the number of layers and parameters). Therefore, we proceed to comparing the complexity of representing a TRSM under the two paradigms (we do not need integer-weight assumption in this section). We simplify our analysis by noting that each component of a TRSM results from the difference of two tropical signomial functions (TSFs) and these TSFs can be computed in parallel using shared layers. Thus, hereafter, we shall focus on TSFs, i.e., functions $f : \mathbb{R}^m \to \mathbb{R}$ of the form $f(x) = \oplus_{i=1}^{r} c_i \odot x^{\odot \alpha_i}$, where $c_i \in \mathbb{R}, \alpha_i \in \mathbb{R}^m$.

Our idea is to construct as input a clique (i.e., a fully-connected graph) with m nodes $A_1, \ldots, A_m$ and distribute the r monomials (almost) evenly such that each node A_i contains $r' = \lceil \frac{r}{m} \rceil$ monomials (padding with zero monomials if $m \cdot r' > r$). Our construction makes use of a comparison gadget which is already introduced in [19] and [83], and introduces a novel selection gadget (details in the Appendix) which proves to be useful with the permutation equivariance restriction of MPNNs.

The two MPNNs below differ in terms of the way they compare the monomials.

Global comparison: Compare m monomials, one from each A_i, simultaneously using coordinate-wise max aggregation. Now redistribute (using the selection gadget) the resulting $O(r')$ maximum monomials (each coordinate yields one such monomial) evenly across the nodes, and recur until the maximum across all monomials is obtained.

Local comparison: First employ a recursive procedure to compare r' monomials assigned to each A_i locally in order to find a maximum monomial for A_i (breaking ties arbitrarily). Now compare the maximum monomials across the nodes using a single global comparison described above.

One limitation of both these architectures is that they can make no more than m comparisons at a time but r can be much larger. Fortunately, a theorem due to [84], also exploited previously by [83], motivates our construction of a **Constant MPNN** consisting only a constant number of layers to represent any TSF. However, the number of learnable parameters in this MPNN can be exponential in the worst case.

Remark 3 (Nature of MPNN and FNN)**.** *Without the equivariance constraint on the message component of MPNN, FNN is much more efficient at constructing affine combinations or tropical monomials. On the other hand, thanks to the parallel MP paradigm of MPNN, it is more effective at comparing these monomials, or in other words, increases the geometric complexity of the model. This benefit of MPNN can be seen in Theorem 4, where one has to build much bigger $\Phi_1^{(t)}$ and $\Phi_2^{(t)}$ to reconstruct the result of the parallel $\phi_1^{(t)}$ and $\phi_2^{(t)}$.*

We therefore proceed to our final algorithm that combines the strengths of these two paradigms. **Hybrid MPNN**: We first use a layer of FNN to calculate r tropical monomials and then employ a single layer of MPNN over an r-clique with coordinate-max to learn f.

Proposition 6. *There exist ReLU MPNN algorithms (Local, Global, Constant, and Hybrid that can learn any TSF $f : \mathbb{R}^m \to \mathbb{R}$ with r monomials. Their respective complexity and tradeoffs are summarized in Table 1.*

We provide details about these algorithms and proofs of their correctness as well as complexity in the Supplementary. Local architecture: Algorithm 6 and Proposition 16, Global architecture: Algorithm 7 and Proposition 17 architecture, Constant architecture: Algorithm 8 and Proposition 18, Hybrid architecture: Algorithm 9 and Proposition 19.

Previously	**Message layers**	**Feedforward layers**	**Learnable parameters**
Deep NN in [19]	None	$\lceil \log_2(r) \rceil + 1$	$\mathcal{O}(rm)$
Deep NN in [83]	None	$\lceil \log_2(m) \rceil + 1$	$\mathcal{O}(rm)$
New (in this work)			
Local (Algorithm 6)	2	$\lceil \log_2(r/m) \rceil + 5$	$\mathcal{O}(rm)$
Global (Algorithm 7)	$\lceil \log_2(r) \rceil + 1$	$3 \lceil \log_m(r) \rceil + 2$	$\mathcal{O}(rm)$
Constant (Algorithm 8)	2	7	$\mathcal{O}(mr^{m+2})$
Hybrid (Algorithm 9)	1	1	$\mathcal{O}(rm)$

Table 1: Complexity of representing any tropical signomial function (TSFs) $f : \mathbb{R}^m \to \mathbb{R}$ consisting of r tropical monomials with different architectures. One more layer is required to compute any tropical rational signomial map (TRSM). The four new methods introduced here construct a graph (based on m and r) and leverage message passing to efficiently compare these monomials.

6 Decision boundary

Lastly, we proceed to characterizing the decision boundary of integer-weighted ReLU MPNNs for classification. We focus on binary classification tasks with a single output to keep the exposition transparent and explicit, albeit one can adapt our construction and analysis accordingly to accommodate multiple classes and outputs. We analyze the decision boundary for both graph and node/edge predictions.

We begin with the analysis for graph classification. Recall from Section 2 that we need an additional readout step $\varphi_{\text{readout}} : \mathbb{R}^{|V| \times d_T} \times \mathbb{R}^{|E| \times d'_T} \to \mathbb{R}^{d_G}$ prior to classification η, i.e.

$$\chi : \mathbb{R}^{d \times |V|} \times \mathbb{R}^{d' \times |E|} \xrightarrow{\varphi^{(T)} \circ \ldots \circ \varphi^{(1)}} \mathbb{R}^{d_T \times |V|} \times \mathbb{R}^{d'_T \times |E|} \xrightarrow{\varphi_{\text{Readout}}} \mathbb{R}^{d_G} \xrightarrow{\eta} \mathbb{R}.$$

Let $\gamma : \mathbb{R} \to \mathbb{R}$ be an injective score function. For $c \in \mathbb{R}$ in $\operatorname{Im}(\gamma)$, we define the decision boundary of χ as $\mathcal{B} = \{z \in \mathbb{R}^m : \chi(z) = \gamma^{-1}(c)\}$, where $m = d|V| + d'|E|$. The tropical hypersurface $\mathcal{T}(f)$ is precisely the set of points x where f is not linear; i.e., two or more monomials in f achieve the value of f at x. We adapt a result from [19, Proposition 6.1] to arrive at the following proposition.

Proposition 7 (Decision boundary for graph classification)**.** *Let $\chi : \mathbb{R}^{d \times |V|} \times \mathbb{R}^{d' \times |E|} \to \mathbb{R}$ be a ReLU MPNN and $\gamma : \mathbb{R} \to \mathbb{R}$ be an injective score function with c in its range. Then χ can be viewed as a tropical rational function, $f \oslash g$ and the decision boundary $\mathcal{B}$ defined above divides $\mathbb{R}^m$ into at most $\mathcal{N}(f)$ connected positive regions and at most $\mathcal{N}(g)$ connected negative regions. Furthermore, $\mathcal{B}$ is contained in the tropical hypersurface of a specific tropical polynomial, namely,*

$$\mathcal{B} \subseteq \mathcal{T}(\gamma^{-1}(c) \odot g \oplus f) .$$

However, the characterization for node and edge classification requires a more nuanced analysis. We focus on node classification, since the treatment for edges is analogous. In this setting the neural network $\eta : \mathbb{R}^{d_T} \to \mathbb{R}$ is applied to the embedding of each node simultaneously, and a scoring function $\gamma_i : \mathbb{R} \to \mathbb{R}$ is then employed for each node to predict its class based on its score c_i. Generally, different scoring functions γ_i may be applied, but in practice, we often use the same score function for all the nodes. Viewed individually, the decision boundary for each vertex is similar to Proposition 7; so we pursue a more interesting problem, namely, characterizing the decision boundary of the vertices resulting from a vectorized score function $\Gamma : \mathbb{R}^{|V|} \to \mathbb{R}^{|V|}$, $[\Gamma(z)]_i = \gamma_i(z)$. However, we immediately hit a roadblock, since choosing a meaningful order for c_i is problematic: the product or subset order is not total whereas the lexical order relies heavily on the ordering of the vertices and thus violates permutation equivariance of the MPNN paradigm. Therefore, we introduce the following notion for the decision boundary.

Definition 1. *The decision boundary $\mathcal{B}$ is defined as $\bigcup_{i=1}^{|V|} \mathcal{B}_i$, where $\mathcal{B}_i = \{z \in \mathbb{R}^m : \nu_i(z) = \gamma_i^{-1}(c_i)\}$.*

In order to proceed, we need to generalize the analysis to tropical hypersurfaces. Specifically, we relate $\mathcal{B}$ with the tropical hypersurface of the components of the corresponding tropical rational map. Invoking Proposition 6.1 in [19], we establish the following proposition.

Proposition 8 (Decision boundary for node classification)**.** *Let $\chi : \mathbb{R}^{d \times |V|} \times \mathbb{R}^{d' \times |E|} \to \mathbb{R}^{|V|}$ be a ReLU MPNN and $\gamma_i : \mathbb{R} \to \mathbb{R}, i = 1, 2, ..., |V|$ be injective score functions with c_i in their range.*

Then χ can be viewed as a tropical rational map $F \oslash G$ and its decision boundary is contained in the tropical hypersurface of a specific tropical polynomial, namely,

$$\mathcal{B} \subseteq \bigcup_{i=1}^{|V|} \mathcal{T}(\gamma^{-1}(c_i) \odot G_i \oplus F_i).$$

Conclusion, Broader Impact, and Limitations

In this paper, we characterize the class of functions learned by ReLU MPNNs through the lens of Tropical geometry. Thus, beyond previous works that are limited to utilizing tropical geometry in the context of ReLU FNNs, our analysis expands the scope to a widely employed class of GNNs, laying the groundwork for further work on the connections with other machine learning models.

We provide both the lower and upper bounds for the number of linear regions of ReLU MPNNs. The upper bound makes some simplifying assumptions; however, Theorem 4 is still general enough to recover existing bounds for FNNs, GCNs, and provide new bounds for widely used GNN architectures such as GraphSAGE and GIN. Our bound is rather analytical; a numerical approach to counting the number of linear regions can be found in [85]. Adapting the method in [85] for MPNNs, and comparing to our analytical bounds, is an interesting future direction.

We also show that the max aggregation operator is more expressive than the sum operator in terms of geometric complexity of ReLU MPNNs (see Remark 2 on how this result contrasts with the implications of the WL test for injective aggregation functions). We thus showcase the dependence of expressivity on message aggregation operator, and furthermore, the connectivity of the graph structure (max and total degree). It remains open whether spectral quantities such as the spectrum of Laplacian have any effect on the geoemetric complexity of ReLU MPNNs.

The theoretical result on equivalence of class of functions learned by ReLU MPNNs and ReLU FNNs is usually not reflected in practice, where ReLU MPNNs typically outperform FNNs. This motivates our results in Section 5, where we consider several ReLU MPNN architectures to represent CPLMs and compare them to ReLU FNNs. Remark 3, Proposition 6 and Table 1 show that ReLU MPNNs can be more efficient in terms of both the number of learnable parameters as well as the number of layers required in order to be able to represent the same CPLMs. We however, sidestepped other important and practical considerations such as the difference in training time, prediction error and the role of optimisation algorithm (e.g. SGD/Adam). Our focus here has been purely theoretical, and we believe that (at least some of) proposed new architectures would benefit from a comprehensive empirical evaluation. We also characterize the decision boundary for graph classification and node classification, explaining the difference between the two.

Overall, we hope that this work fosters further research on design and analysis of modern deep architectures through the fruitful machinery of tropical geometry.

Acknowledgments

VG acknowledges support from the Research Council of Finland for the project “Human-steered next-generation machine learning for reviving drug design” (grant decision 342077), the Jane and Aatos Erkko Foundation for the project “Biodesign: Use of artificial intelligence in enzyme design for synthetic biology” (grant 7001703), and a Saab-WASP initiative (grant 411025). TAP wants to thank Johanna Immonen, Jannis Halbey, Negar Soltan Mohammadi, Yunseon (Sunnie) Lee, Bruce Nguyen and Nahal Mirzaie for their discussion, support and wonderful experience during his research internship at Aalto University in 2022.

References

[1] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *AAAI*, pages 133–136, 1982.

[2] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009.

[3] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2005.

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[5] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2702–2711. JMLR.org, 2016.

[6] Z. Zhang, F. Wu, and W. S. Lee. Factor graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[7] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[8] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, E. Monti, and M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[9] Y. Wang, Y. Chang, Y. Liu, J. Leskovec, and P. Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations (ICLR)*, 2021.

[10] Amauri Souza, Diego Mesquita, Samuel Kaski, and Vikas Garg. Provably expressive temporal graph networks. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[13] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[15] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[16] B. Weisfeiler and A. A. Leman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[17] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[18] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[19] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In Jennifer G. Dy and Andreas Krause, editors, *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 5819–5827. PMLR, 2018.

[20] Stefanie Jegelka. Theory of graph neural networks: Representation and learning, 2022.

[21] Hao Chen, Yu Guang Wang, and Huan Xiong. Lower and upper bounds for numbers of linear regions of graph convolutional networks, 2022.

[22] D. Maclagan and B. Sturmfels. *Introduction to Tropical Geometry*. Graduate Studies in Mathematics. American Mathematical Society, 2015.

[23] M. Joswig. *Essentials of Tropical Combinatorics*. Graduate Studies in Mathematics. American Mathematical Society, 2021.

[24] Vasileios Charisopoulos and Petros Maragos. A tropical approach to neural networks with piecewise linear activations. *ArXiv*, abs/1805.08749, 2018.

[25] Guido Montúfar, Yue Ren, and Leon Zhang. Sharp bounds for the number of regions of maxout networks and vertices of minkowski sums. *SIAM Journal on Applied Algebra and Geometry*, 6(4):618–649, 2022.

[26] Motasem Alfarra, Adel Bibi, Hasan Hammoud, Mohamed Gaafar, and Bernard Ghanem. On the decision boundaries of neural networks: A tropical geometry perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):5027–5037, 2023.

[27] Georgios Smyrnis and Petros Maragos. Tropical polynomial division and neural networks, 2019.

[28] Marie-Charlotte Brandenburg, Georg Loho, and Guido Montúfar. The real tropical geometry of neural networks, 2024.

[29] Matthew Trager, Kathlén Kohn, and Joan Bruna. Pure and spurious critical points: a geometric study of linear networks, 2020.

[30] Dhagash Mehta, Tianran Chen, Tingting Tang, and Jonathan D. Hauenstein. The loss surface of deep linear networks viewed through the algebraic geometry lens. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5664–5680, 2022.

[31] J. Elisenda Grigsby and Kathryn Lindsey. On transversality of bent hyperplane arrangements and the topological expressiveness of relu neural networks. *SIAM Journal on Applied Algebra and Geometry*, 6(2):216–242, 2022.

[32] Francis Williams, Matthew Trager, Daniele Panozzo, Cláudio T. Silva, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Edward A. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8376–8385, 2019.

[33] Petros Maragos, Vasileios Charisopoulos, and Emmanouil Theodosis. Tropical geometry and machine learning. *Proceedings of the IEEE*, 109(5):728–755, 2021.

[34] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: the story so far. *J. Mach. Learn. Res.*, 24(1), mar 2024.

[35] Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond weisfeiler-lehman: A quantitative framework for GNN expressiveness. In *The Twelfth International Conference on Learning Representations*, 2024.

[36] Chaitanya K. Joshi, Cristian Bodnar, Simon V. Mathis, Taco Cohen, and Pietro Liò. On the expressive power of geometric graph neural networks. In *International Conference on Machine Learning*, 2023.

[37] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2625–2640. Curran Associates, Inc., 2021.

[38] B. Rieck, C. Bock, and K. Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning (ICML)*, 2019.

[39] Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. *ICML*, 2022.

[40] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6790–6802. Curran Associates, Inc., 2021.

[41] Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In *International Conference on Learning Representations*, 2022.

[42] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

[43] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[44] R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks. In *SIAM International Conference on Data Mining (SDM)*, 2021.

[45] M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. PersLay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[46] Yuzhou Chen, Baris Coskunuzer, and Yulia Gel. Topological relational learning on graphs. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27029–27042. Curran Associates, Inc., 2021.

[47] Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations*, 2022.

[48] Omri Puny, Derek Lim, Bobak T. Kiani, Haggai Maron, and Yaron Lipman. Equivariant polynomials for graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[49] Z. Chen, L. Chen, S. Villar, and J. Bruna. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[50] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

[51] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Letourneau, and P. Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[52] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1407–1418. PMLR, 18–24 Jul 2021.

[53] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

[54] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

[55] Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. ACMP: allen-cahn message passing with attractive and repulsive forces for graph neural networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[56] V. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning (ICML)*, 2020.

[57] Nima Dehmamy, Albert-Laszlo Barabasi, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[58] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Lio, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[59] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023.

[60] A. Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations (ICLR)*, 2020.

[61] A. Loukas. How hard is to distinguish graphs with graph neural networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[62] S. S. Du, K. Hou, R. Salakhutdinov, B. Póczos, R. Wang, and K. Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[63] Sanjukta Krishnagopal and Luana Ruiz. Graph neural tangent kernel: Convergence on large graphs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17827–17841. PMLR, 2023.

[64] Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. The power of recursion in graph neural networks for counting substructures. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 11023–11042. PMLR, 25–27 Apr 2023.

[65] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022.

[66] S. Verma and Z.-L. Zhang. Stability and generalization of graph convolutional neural networks. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

[67] R. Liao, R. Urtasun, and R. Zemel. A PAC-bayesian approach to generalization bounds for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[68] K. Xu, J. Li, M. Zhang, S. S. Du, K.-I. Kawarabayashi, and S. Jegelka. What can neural networks reason about? In *International Conference on Learning Representations (ICLR)*, 2020.

[69] Sohir Maskey, Ron Levie, Yunseok Lee, and Gitta Kutyniok. Generalization analysis of message passing neural networks on large random graphs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[70] Pascal Esser, Leena Chennuru Vankadara, and Debarghya Ghoshdastidar. Learning theory can (sometimes) explain generalisation in graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27043–27056. Curran Associates, Inc., 2021.

[71] Haotian Ju, Dongyue Li, Aneesh Sharma, and Hongyang R Zhang. Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion. *Artificial Intelligence and Statistics*, 2023.

[72] Yangze Zhou, Gitta Kutyniok, and Bruno Ribeiro. OOD link prediction generalization capabilities of message-passing GNNs in larger test graphs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[73] K. Xu, M. Zhang, J. Li, S. S. Du, K.-I. Kawarabayashi, and S. Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[74] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In Marina Meila and Tong Zhang, editors, *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 837–851. PMLR, 2021.

[75] Andrew Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[76] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[77] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2596–2604. PMLR, 2019.

[78] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10514–10523. PMLR, 2020.

[79] Itay Safran, Gal Vardi, and Jason D. Lee. On the effective number of linear regions in shallow univariate reLU networks: Convergence guarantees and implicit bias. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[80] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 06–11 Aug 2017.

[81] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[82] Alexis Goujon, Arian Etemadi, and Michael Unser. On the number of regions of piecewise linear neural networks. *Journal of Computational and Applied Mathematics*, 441:115667, 2024.

[83] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[84] Shuning Wang and Xusheng Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51:4425–4431, 12 2005.

[85] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4558–4566. PMLR, 10–15 Jul 2018.

Technical Appendix - Supplementary Material

We now describe in detail all the algorithms introduced in the main paper, as well as proofs for all our results. We begin with the results from the section on Tropical algebra.

Tropical algebra

Lemma 2. *Every ReLU FNN $\nu : \mathbb{R}^m \to \mathbb{R}^p$ can be represented by a ReLU MPNN $\chi : \mathbb{R}^{1\times m} \to \mathbb{R}^{1\times p}$.*

Proof. This is obvious when we set G to be a one-clique (i.e., one vertex graph with a self-edge) and $\phi_2^{(1)} = \nu$, with max or sum aggregation, $\phi_1^{(1)} = \text{id}$, $W_{\text{self}}^{(1)} = 0$ and $W_{\text{neigh}}^{(1)} = \text{Id}_{p\times p}$. □

Lemma 3. *Any ReLU MPNN $\chi : \mathbb{R}^{|V|\times d} \times \mathbb{R}^{|E|\times d'} \to \mathbb{R}^{|V|\times d_{out}} \times \mathbb{R}^{|E|\times d'_{out}}$ can be expressed as a TRSM $F \oslash G : \mathbb{R}^m \to \mathbb{R}^p$, with $m = |V|d + |E|d'$ and $p = |V|d_{out} + |E|d'_{out}$.*

Proof. To prove the proposition, we will show that each layer $\varphi^{(t)}$ is a TRSM on its input, thus χ is a TRSM as well. $\phi_2^{(t)} : \mathbb{R}^{d_{t-1}} \times \mathbb{R}^{d_{t-1}} \times \mathbb{R}^{d'_{t-1}} \times \mathbb{R}^{d'_{t-1}} \to \mathbb{R}^{\tilde{d}_t}$ and $\phi_1^{(t)} : \mathbb{R}^{\tilde{d}_t} \to \mathbb{R}^{\bar{d}_t}$ are ReLU FNNs, hence by [19, Proposition 5.6], they are TRSMs. The aggregation operators (sum and coordinate-wise max for our purpose) are tropical operators pertaining to tropical multiplication and addition respectively. Thus, each component of $\varphi_{\text{Agg}}^{(t)}$ is a TRSF. On the other hand, $\sigma_{\text{Update}}^{(t)}$ is a ReLU activation, the update function $\varphi_{\text{Update}}^{(t)}$, and thus, each MPNN layer $\varphi^{(t)}$ is also a TRSM on its input. □

Proof of Proposition 1. Note that [19, Corollary 5.3] established the equivalence of $(2), (3), (4)$. Lemma 2 and Lemma 3, imply $(4) \Rightarrow (1)$, and $(1) \Rightarrow (2)$ respectively and hence we are done. □

Bounds on geometric complexity

We here provide more detailed analysis for the Geometric complexity and the proof of Theorem 3 and Theorem 4. To simplify the analysis, we will assume that we have no edge embedding throughout MPNN layers - the analysis for the most general case follows the same idea. In this analysis, we also assume that inputs of the model follows the same graph G. Let S and D be the maximum and total degree of all vertices of G.

We will consider the input $H^{(t-1)}$ of each message passing layer $\varphi^{(t)}$ as a stacked vector of edge node embedding $h_v^{(t-1)}$. As we do not want to calculate unused messages between non-adjacency nodes, with the vectorized form (instead of matrix form), we could consider each degree separately and thus reduce the dimension of intermediate layers from $S|V|$ to D. Forming the vectorized form requires assumption of an order between the nodes and the induced lexical order for the edges introduced in Section 2. However, we can neutralize this effect by multiplying with the appropriate permutation matrix P. We now outline the changes happen in each layer.

1. The vector $H^{(t)} \in \mathbb{R}^{|V|d_t}$ will be the input to $\varphi^{(t)} : \mathbb{R}^{|V|d_{t-1}} \to \mathbb{R}^{|V|d_t}$ ($H^{(0)} = X$).
2. We then apply $\phi_2^{(t)} : \mathbb{R}^{|V|d_{t-1}} \times \mathbb{R}^{|V|d_{t-1}} \to \mathbb{R}^{|V|\tilde{d}_t}$ to every pair of adjacent node embedding A_i and A_j to obtain the message from node A_i to node A_j. The idea is to form a big FNN $\Phi_2^{(t)}$ applied on the whole vector $H^{(t-1)}$ to obtain the output $X^{(t)} \in \mathbb{R}^{D\tilde{d}_t}$. The FNN $\Phi_2^{(t)}$ in general depends *depends on D and connectedness of the graph.*
3. After that, we perform the operator $\square^{(t)} = \square_{u\in\mathcal{N}(v)}^{(t)}$ (sum or coordinate-wise max/min) of all the neighboring messages to obtain $Y^{(t)} \in \mathbb{R}^{|V|\tilde{d}_t}$. We will see later that there is a *significant difference* in the two cases of aggregation operator and suggest that the *coordinate-wise max/min* could indeed *increase our geometric complexity*.
4. Analogously, we then form $Z^{(t)} \in \mathbb{R}^{|V|\bar{d}_t}$ by forming a bigger network $\Phi_1^{(t)}$ (based on $\phi_1^{(t)} : \mathbb{R}^{\tilde{d}_t} \to \mathbb{R}^{\bar{d}_t}$), applied to $Y^{(t)}$, i.e. $Z^{(t)} = \Phi_1^{(t)}(Y^{(t)})$.

5. We combine step 2, 3 and 4 to form a big FNN $\Phi_{\text{Agg}}^{(t)}$ applied to $H^{(t-1)}$
6. Lastly, we obtain $H^{(t)} \in \mathbb{R}^{|V| \times d_t}$ by performing the $\varphi_{\text{Update}}^{(t)}$.

We proceed with the first step. The algorithm 1 shows us how to build $\Phi_2^{(t)}$ from $\phi_2^{(t)}$ by replicating the neural network $\phi_2^{(t)}$ for neighboring nodes and putting 0 for the nodes that are not adjacent. Without no edge embeddings, our $\phi_2^{(t)}$ only depends on the node embedding $h_{A_i}^{(t-1)}$ and $h_{A_j}^{(t-1)}$. Our notations from now on always indicate that $\phi_1^{(t)}$, $\phi_2^{(t)}$ are applied to each (pair of) embeddings whereas $\Phi_1^{(t)}$, $\Phi_2^{(t)}$ ($\Phi_3^{(t)}$) are applied to the corresponding stacked vector as a whole.

Algorithm 1 Building $\Phi_2^{(t)}$

Input: Stacked vector $H^{(t-1)} \in \mathbb{R}^{|V|d_{t-1}}$ of embedding of all vertices.

We will only describe the construction for weight, the bias could be done analogously. Let $W_2^{t,l}$ and $n_2^{t,l}$ be the weight and number of output nodes of the ℓ-th layer of $\phi_2^{(t)}$ respectively ($n_2^{t,0} = d_{t-1}$). Thus, $W_2^{t,l} \in \mathbb{R}^{n_2^{t,l} \times n_2^{t,l-1}}$. For the first layer $l = 1$, let $W_2^{t,1} = \left[W_{2,1}^{t,1}, W_{2,2}^{t,1}\right]$, where $W_{2,1}^{t,1}, W_{2,2}^{t,1} \in \mathbb{R}^{n_2^{t,1} \times d_{t-1}}$ correspond to the weight of the first d_{t-1} nodes (of the embedding $h_{A_i}^{(t-1)}$) and last d_{t-1} nodes (of the $h_{A_j}^{(t-1)}$) respectively.

We now define the weight for $\Phi_2^{(t)}$. In the first layer of $\Phi_2^{(t)}$, $\tilde{W}_2^{t,1} \in \mathbb{R}^{Dn_2^{t(1)} \times |V|d_{t-1}}$ by breaking it into blocks $[\tilde{W}_2^{t,1}]_{\kappa,\kappa'}$ of size $n_2^{t,1} \times d_{t-1}$. For each edge κ-th (according to the ordering of edges), assume it is (A_i, A_j), then, we set

$$[\tilde{W}_2^{t,1}]_{\kappa,\kappa'} = \begin{cases} W_{2,1}^{t,1} + W_{2,2}^{t,1} & \text{if } \kappa' = i = j \\ W_{2,1}^{t,1} & \text{if } \kappa' = i \\ W_{2,2}^{t,1} & \text{if } \kappa' = j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The remaining layers ℓ are easy to build: let $\tilde{W}_2^{t,l} \in \mathbb{R}^{Dn_2^{t,l} \times Dn_2^{t,l-1}}$ and $\tilde{W}_2^{t,l} = \begin{bmatrix} W_2^{t,l} & 0 & ... & 0 \\ 0 & W_2^{t,l} & ... & 0 \\ 0 & 0 & ... & W_2^{t,l} \end{bmatrix}$ and applied the activation accordingly.

Proposition 9. *ReLU FNN $\Phi_2^{(t)}$ in Algorithm 1, applied to $H^{(t-1)}$, will result in the same output as we apply $\phi_2^{(t)}$ separately for neighboring nodes $h_{A_i}^{(t-1)}$, $h_{A_j}^{(t-1)}$ and combine the result.*

Proof. We first assume that $\phi_2^{(t)}$ has only one layer and the action of ReLU activation is applied coordinate-wise, we can assume that

$$\phi_2^{(t)}(h_{A_i}^{(t-1)}, h_{A_j}^{(t-1)}) = W_2^{(t),1} \begin{bmatrix} h_{A_i}^{(t-1)} \\ h_{A_j}^{(t-1)} \end{bmatrix} = W_{2,1}^{t,1} h_{A_i}^{(t-1)} + W_{2,2}^{(t,1} h_{A_j}^{(t-1)}.$$

There are D edges in total, thus $X^{(t)} = \begin{bmatrix} X_1^{(t)} \\ ... \\ X_D^{(t)} \end{bmatrix} \in \mathbb{R}^{Dn_2^{t,2}}$ and $H^{(t-1)} = \begin{bmatrix} h_{A_1}^{(t-1)} \\ ... \\ h_{A_{|V|}}^{(t-1)} \end{bmatrix}$. For the κ-th edge (A_i, A_j), we want $X_\kappa^{(t)} = \phi_2^{(t)}(h_{A_i}^{(t-1)}, h_{A_j}^{(t-1)})$, the formula (3) will yield the desired result.

Then applying the following layers $\Phi_2^{(t)}$ is equivalent to applying the corresponding layers of $\phi_2^{(t)}$ on each result. □

Proposition 10 (The matrix to take the sum). *If $\square_{u \in \mathcal{N}(v)}^{(t)}$ is sum, then*

$$Y^{(t)} = MX^{(t)},$$

where M is a matrix that generally depends on the adjacency matrix A of G. Thus, it does not change the linear/convex degree and its effect can be absorbed by the first layer of $\Phi_1^{(t)}$.

Proof. We first build $M \in \mathbb{R}^{|V|\tilde{d}_t \times D\tilde{d}_t}$, by breaking it into blocks of size $\tilde{d}_t \times \tilde{d}_t$. Then each block $M_{\kappa,\kappa'} = \begin{cases} \mathrm{Id}_{\tilde{d}_t \times \tilde{d}_t} & \text{if edge } \kappa' = (A_\kappa, A_j) \in E \\ 0 & \text{otherwise} \end{cases}$.

It is easy to see that $Y^{(t)} = MX^{(t)}$, since the matrix M is the same as the adjacency matrix A of G replacing 1 with $\mathrm{Id}_{\tilde{d}_t \times \tilde{d}_t}$. □

On the other hand, when the operator $\square^{(t)}$ is coordinate-wise max-min, the analysis becomes a little bit more complicated. In particular, we can create a neural network $\Phi_3^{(t)}$ to represent its effect. In the Algorithm 2, we want to make use of the comparing architecture in [83]. A minor blockage is the appearance of the weight $1/2$ in the comparing layer. Fortunately, by scaling the input by a factor of $1/2$ (which does not change the number of linear regions) and scaling the weights by a factor of 2, we could solve this problem. We note that $\Phi_3^{(t)}$ depends heavily on the graph G.

Algorithm 2 Building $\Phi_3^{(t)}$

Input: $X^{(t)} \in \mathbb{R}^{D\tilde{d}_t}$.

We first scale all entries in the input by 1/2. Then we create a FNN that resembles the construction in [83], however, we will not compare two consecutive nodes, but the comparison is done according to the edges of the graph.

Proposition 11. *If $\square^{(t)}$ is coordinate-wise max/min, then*

$$Y^{(t)} = \Phi_3^{(t)}(X^{(t)}),$$

where $\Phi_3^{(t)}$ is built according to Algorithm 2. □

Remark 4. *Here, we surmise that if we have a coordinate-wise max/min aggregation operator, the geometric complexity of our model grows exponentially with D and polynomially with S.*

After that, we continue with step 4, where we apply the neural network $\phi_1^{(t)}$ to form the transformation of the aggregated message. Similar to step 2, we would want to build $\Phi_1^{(t)}$ that is applied on the whole $Y^{(t)}$ as a vector. However, this is much easier to form $\phi_1^{(t)}$ than that of $\Phi_2^{(t)}$ as

$$([\Phi_1^{(t)}(Y^{(t)})^\top]_{(i-1)\bar{d}_t+1,\ldots,i\bar{d}_t}) = \phi_1^{(t)}(Y^{(t)}_{(i-1)\tilde{d}_t+1,\ldots,i\tilde{d}_t})^\top. \tag{4}$$

Algorithm 3 Building $\Phi_1^{(t)}$

Input: $Y^{(t)} \in \mathbb{R}^{|V|\tilde{d}_t}$.

Let $W_1^{t,l} \in \mathbb{R}^{n_1^{t,l+1} \times n_1^{t,l}}$ and $\tilde{W}_1^{t,l} \in \mathbb{R}^{|V|n_1^{t,l+1} \times |V|n_1^{t,l}}$ be the weight of the ℓ-th layer of $\phi_2^{(t)}, \Phi_2^{(t)}$ respectively. Then $\tilde{W}_1^{t,l} = \begin{bmatrix} W_1^{t,l} & 0 & \ldots & 0 \\ 0 & W_1^{t,l} & \ldots & 0 \\ 0 & 0 & \ldots & W_1^{t,l} \end{bmatrix}$ and applied the activation accordingly.

Obviously, since each component is applied separately, we have the following proposition.

Proposition 12.

$$([\Phi_1^{(t)}(Y^{(t)})^\top]_{(i-1)\bar{d}_t+1,\ldots,i\bar{d}_t}) = \phi_1^{(t)}(Y^{(t)}_{(i-1)\tilde{d}_t+1,\ldots,i\tilde{d}_t})^\top. \tag{5}$$

□

Lower bound on geometric complexity

Proof of Theorem 3. By Proposition 9, Proposition 13 and Proposition 11, we note that each Aggregation step $\varphi_{Agg}^{(t)}$ can be written as a FNNs $\Phi_{\text{Agg}}^{(t)}$ applied to $H^{(t)}$. Note that $\square^{(t)} = \sum$ can be absorbed by the first layer of $\Phi_1^{(t)}$ (constructed in Algorithm 3). On the other hand, $\square^{(t)} = \max$ can be seen as a max-out layer with rank S). Thus, by [76, Theorem 8 proof], it can identify S regions of the input. Then if $\varphi_{\text{Update}}^{(t)}(h_v, m_v) = m_v$ then we can write $H^{(t)} = \Phi_{\text{Agg}}^{(t)}(H^{(t-1)})$. Thus, by [76, Theorem 4], we have the maximum number of linear region of functions computed by any ReLU MPNN is lower bounded by

$$S^{t_0} \frac{\left(\prod_{t=1}^{T}\left(\prod_{l=1}^{L_1^{(t)}} n_{1,d_0}^{t,l} \prod_{l=1}^{L_2^{(t)}} n_{2,d_0}^{t,l}\right)\right)}{n_{1,d_0}^{T,L_1^{(T)}}} \sum_{j=0}^{d_0} \binom{d_T}{j},$$

$\square$

Upper bound on geometric complexity

Remark 5. *If the neural network $\phi_2^{(t)}$ has $n_2^{t,l}$ output nodes in its ℓ-th hidden layer, then the neural network $\Phi_2^{(t)}$ have $Dn_2^{t,l}$ nodes in its ℓ-th layer. Therefore, the number of output nodes and hidden layer nodes depend on the connectivity of the network.*

Proposition 13. *If the neural network $\phi_1^{(t)}, \phi_2^{(t)}$ has $n_1^{t,l}, n_2^{t,l}$ number of nodes in its ℓ-th hidden layer then the neural network $\Phi_1^{(t)}, \Phi_2^{(t)}$ have $|V|n_1^{t,l}, |D|n_1^{t,l}$ number of nodes in its ℓ-th layer. Thus, if*

- $n_2^{t,l} \geq \frac{D}{|V|} d_t$ *for all* $l = 1, ..., L_2^{(t)}$;
- $n_1^{t,l} \geq d_t$ *for all* $l = 1, ..., L_1^{(t)}$;

then applying [19, Theorem 6.3], we obtain an upper bound for convex degree:

$$\mathcal{N}_c(\Phi_{Agg}^{(t)}) \leq \prod_{l=1}^{L_2^{(t)}-1} \sum_{i=0}^{|V|d_t} \binom{Dn_2^{t,l}}{i} \mathcal{N}_c(\square^{(t)}) \prod_{l=1}^{L_1^{(t)}-1} \sum_{i=0}^{|V|d_t} \binom{|V|n_1^{t,l}}{i} \tag{6}$$

convex linear regions, where

$$\mathcal{N}(\square^{(t)}) = \begin{cases} 1 & \textit{if } \square^{(t)} = \sum \\ \mathcal{O}(S^{D\tilde{d}_t}) & \textit{if } \square^{(t)} = \max . \end{cases} \tag{7}$$

Proof. If $\square^{(t)}$ is coordinate-wise max/min, then $\Phi_3^{(t)}$ has $\lceil \log_2 S \rceil + 1$ layers. For ease of analysis, we can assume each hidden layer has $D\tilde{d}_t$ nodes (this only increase the convex/linear degree) except for the first layer with at most $2D\tilde{d}_t$ nodes, thus from [19, Theorem 6.3], we have

$$\begin{aligned} \mathcal{N}_c(\Phi_3^{(t)}) &\leq \left(\sum_{i=0}^{D\tilde{d}_t} \binom{2D\tilde{d}_t}{i}\right) \prod_{l=1}^{\lceil \log_2 S \rceil} \sum_{i=0}^{D\tilde{d}_t} \binom{D\tilde{d}_t}{i} \\ &= 2^{2D\tilde{d}_t - 1} \cdot 2^{D\tilde{d}_t \lceil \log_2 S \rceil}, \end{aligned}$$

as $\sum_{i=0}^{n} \binom{2n}{i} = 2^{2n-1}$. Using Propositions 9, 11, and 13, we can write the result of $\varphi_{\text{Agg}}^{(t)}$ (i.e. $Z^{(t)}$) as a result of a combined FNN $\Phi_1^{(t)} \circ \Phi_3^{(t)} \circ \Phi_2^{(t)}$ applied to $H^{(t-1)}$. Then applying [19, Theorem 6.3] yields the stated result. $\square$

Thus, we are left with the final step in $\varphi_{\text{Update}}^{(t)}$. If we stack the two vector together, which is now of dimension $|V|\bar{d}_t + |V|d_t$, we can apply [19, Lemma D.4] to prove the following Corollary.

Corollary 5. *Let $\varphi^{(t)}_{Update} = \sigma^{(t)}_{Update} \circ \rho^{(t)}_{Update} : \mathbb{R}^{|V|(\bar{d}_t+d_t)} \to \mathbb{R}^{|V|d_t}$. If $|V|(\bar{d}_t + d_t) \leq |V|(d_t)$, i.e. $\bar{d}_t + d_t \leq d_t$, then*

$$\mathcal{N}_c(\sigma_{Update} \circ \rho|(\bar{d}_t + d_t)|V|) \leq \sum_{i=0}^{|V|(\bar{d}_t+d_t)} \binom{|V|d_t}{i}.$$

□

The following results together yield an important Proposition that allows us to analyze each component of $\varphi^{(t)}$ separately before combining them. The following lemma is trivial, but we provide a proof for completeness.

Lemma 4. *Let $F \in Rat(m,p)$ and $m' \leq m$, Then*

$$\mathcal{N}_c(F|m') \leq \mathcal{N}_c(F) \tag{8}$$

Proof. We invoke the fact that if F is an affine transformation of $\mathbb{R}^n$ and $A \subset \mathbb{R}^n$ is convex, then the image $F[A]$ is also convex. Thus, a convex linear region in $\mathcal{N}_c(F|d)$ is still a convex and linear region in $\mathcal{N}_c(F)$, and vice versa. On the other hand, some convex linear regions in $\mathbb{R}^n$ do not intersect the affine space, resulting in the inequality. □

We first recall [19, Theorem D.3]: Let F, G in $Rat(m,p)$ and $Rat(m', m)$. Let $H = (h_1, ..., h_p) \in Rat(m', p)$ defined by $h_i := f_i \circ G$ for $i = 1, .., p$. Then

$$\mathcal{N}(H) \leq \mathcal{N}_c(H) \leq \mathcal{N}_c(F|m') \cdot \mathcal{N}_c(G).$$

Proof of Proposition 2. We note that

$$\chi^{(t)} = \varphi^{(t)} \circ \chi^{(t-1)}, \tag{9}$$

thus using [19, Theorem D.3], we have

$$\mathcal{N}_c(\chi^{(t)}) \leq \mathcal{N}_c(\varphi^{(t)}||V|d_{t-1})\, \mathcal{N}_c(\chi^{(t-1)}). \tag{10}$$

Note that $\varphi^{(t)}(H^{(t-1)}) = \varphi^{(t)}_{\text{Update}}(\varphi^{(t)}_{\text{Agg}}(H^{(t-1)}), H^{(t-1)})$ thus

$$\mathcal{N}_c(\varphi^{(t)}||V|d_t) \leq \mathcal{N}_c(\varphi^{(t)}) \leq \mathcal{N}_c(\varphi^{(t)}_{\text{Update}}|(d_t + d_t)|V|)\, \mathcal{N}_c(\varphi^{(t)}_{\text{Agg}}). \tag{11}$$

We can further break the terms $\mathcal{N}_c(\varphi^{(t)}_{\text{Agg}})$ down by noticing that $\varphi^{(t)}_{\text{Agg}} = \Phi^{(t)}_1 \circ \square^{(t)} \circ \Phi^{(t)}_2$, applying in [19, Theorem D.3] multiple times and Lemma 4. □

Thus, bringing all the corollary together and apply [19, Theorem 6.3] recursively, we obtain the following bound for the geometric complexity:

Proof of Theorem 4. The theorem follows from Proposition 2, which tells us that we could analyze the model in steps. Thus, bringing the pieces together, we have

$$\prod_{t=1}^{T} \underbrace{\left(\prod_{l=1}^{L^{(t)}_1-1} \sum_{i=0}^{|V|\tilde{d}_t} \binom{|V|n^{t,l}_1}{i} \right)}_{\text{from } \Phi^{(t)}_1} \underbrace{\left(\prod_{l=1}^{L^{(t)}_2-1} \sum_{i=0}^{|V|\times d_{t-1}} \binom{Dn^{t,l}_2}{i} \right)}_{\text{from } \Phi^{(t)}_2} \underbrace{\left(\sum_{i=0}^{|V|(\bar{d}_t+d_{t-1})} \binom{|V|d_t}{i} \right)}_{\text{from } \varphi^{(t)}_{\text{Update}}} \mathcal{N}_c(\square^{(t)}),$$

where $\mathcal{N}_c(\square^{(t)}) = \begin{cases} 1 & \text{if } \square^{(t)} \text{ is sum} \\ \frac{1}{2}(8S)^{D\tilde{d}_t} & \text{if } \square^{(t)} \text{ is coordinate-wise max/min .} \end{cases}$ □

6.1 Consequences of Theorem 4

We first recover the upper bound for FNNs and GCNs (with ReLU activations and integer-weights) established in [19, Theorem 6] and [21, Theorem 4] respectively as special cases. The one for FNN is straightforward. On the other hand, the bound for GCNs requires slightly more work.

Proof of Corollary 2. Firstly, note that GCN can be modelled by setting $\phi_1^{(t)} = Id$, $\square^{(t)} = \text{sum}$, and $\varphi_{\text{Update}}^{(t)} = Z^{(t)}$, where $Z^{(t)}$ is just a stacked vector of $m_{A_i}^{(t)}$. With this simplification, we can in fact have a stronger version of Proposition 2 (going down to input layer):

$$\begin{aligned}\mathcal{N}_c(\chi) &\leq \prod_{t=1}^{T} \mathcal{N}_c(\square^{(t)} \circ \Phi_2^{(t)} || V|d_0) \\ &\leq \prod_{t=1}^{T} \left(\sum_{i=0}^{|V|d} \binom{|V|d_t}{i} \right) \quad \text{(by Proposition 10, } \square^{(t)} \text{ is just linear transformation).}\end{aligned}$$

□

Proof of Corollary 3. In both case, we note that $\phi_1^{(t)} = \text{id}$, thus, we can forget the contribution of Φ_1. If the aggregation operator is mean, we can substitute $\square^{(t)} = \sum$, then $\phi_2^{(t)} = \text{id}$ and φ_{Update} stays the same. Thus, we have

$$\mathcal{N}_c(\varphi^t) \leq \sum_{i=0}^{2|V|d_{t-1}} \binom{|V|d_t}{i} \tag{12}$$

If the aggregation operator is pooling, then $\phi_2^{(t)}$ is a one-layer FNN, $\square^{(t)} = \max$ and φ_{Update} stays the same. Thus, we have

$$\mathcal{N}_c(\varphi^t) \leq \left(\sum_{i=0}^{|V|\tilde{d}_{t-1}} \binom{|D|\tilde{d}_t}{i} \right) \left(\sum_{i=0}^{|V|(\tilde{d}_t + d_{t-1})} \binom{|V|d_t}{i} \right). \tag{13}$$

□

Proof of Corollary 4. According to [15, Equation 4.1], by adding a self-edge to each node in the graph, t-th layer of GIN can be written as $\square^{(t)} = \sum$ (or affine transformation - which does not change the geometric complexity), $\phi_2^{(t)} = \text{id}$ and $\phi_1^{(t)} = MLP$ indicated, thus

$$\mathcal{N}_c(\chi) \leq \prod_{t=1}^{T} \left(\prod_{l=1}^{L^{(t)}-1} \sum_{i=0}^{|V|d_t} \binom{|V|n^{t,l}}{i} \right).$$

□

New ReLU MPNNs architectures and complexity tradeoffs

In most of the following architectures, the aggregated message has the following form,

$$m_{A_i}^{(t)} = \phi_1^{(t)}(\square_{A_j \in \mathcal{N}(A_i)}^{(t)} \phi_2^{(t)}(h_{A_j}^{(t-1)})), \tag{14}$$

i.e., the aggregated message to node A_i depends only on the neighboring nodes A_j (and not A_i itself).

Lemma 5. *Let $F \oslash G : \mathbb{R}^m \to \mathbb{R}^p$ be a TRSM. Suppose each component F_i of F can be represented by an FNN/MPNN with L_{F_i} layers, and G_i of G by an FNN/MPNN with L_{G_i} layers. Then $F \oslash G$ can be represented as an L-layer FNN/MPNN with $L \leq \max_{i=1}^{p}\{\max\{L_{F_i}, L_{G_i}\}\} + 1$.*

Proof. The main idea underlying our proof here is to exploit parallelization. Firstly, for FNNs, we can compute all F_i and G_i in parallel using $\max_{i=1}^{n}\{\max\{L_{F_i}, L_{G_i}\}\}$ layers: we set the block matrix for weights between the different components to zero (for the components that require fewer layers, we can add additional dummy layers with their weights set to the identity matrix Id). We then need just one additional layer to compute the difference $F_i \oslash G_i$, again in parallel, for all $i = 1, .., n$. Similarly, for MPNNs, we construct F_i and G_i in parallel following appropriate stacking of the weights of the components. In this case, each additional layer t (for the components with fewer layers) is treated as follows: we set the weights for $\phi_2^{(t)}, \phi_1^{(t)}$ to zero and $W_{\text{self}}^{(t)}$ to identity. Lastly, to compute the difference F_i and G_i, we just need one additional MPNN layer where aggregation part is 0 ($\phi_2^{(L)}, \phi_1^{(L)} = 0; \square^{(L)}$ is sum or coordinate-wise max), and $W_{\text{self}}^{(L)} = [Id_{p\times p} \quad -Id_{p\times p}]$. □

We provide an overview and detailed algorithms for the gadgets here. The *broadcast* gadget sets up an MPNN by replicating its input across nodes of a fully connected graph. It also endows each node with its ROE. The *selection* and *comparison* gadgets are implemented as FFNs: the former is used for partitioning the monomials across the nodes, whereas the latter determines the larger of its two input monomials.

6.2 Broadcast gadget

Broadcast gadget. Given an input vector $x \in \mathbb{R}^m$, we first construct a fully connected graph with m vertices $A_1, ..., A_m$. The feature vector for A_1 is constructed as follows: first m coordinates comprise x, the next coordinate is set to 1 (for bias), and the last m coordinates are set to ROE $1_m - e_1$. The feature vectors for all other nodes A_i consist of $m+1$ zeros followed by $1_m - e_i$. A parameterized message passing layer with weights determined by a target TSF f is then learned to compute all the r monomials $p_1, p_2, \ldots, p_r$ of f. We provide all the details, including the operators in Algorithm 4. It can be shown that the embedding of node A_i after broadcasting is $\left[p_1, p_2, ..., p_r, (1_m - e_i)^\top\right]^\top$.

Algorithm 4 Broadcast component: MPNN layer

Build an m-clique with vertices A_1, ..., A_m and bidirectional edges, and introduce loops with m self-edges from each vertex A_i to itself.

Prepare node embeddings. For A_1, we set its first m coordinates to the input x, the next coordinate to 1 for bias, and then the final m coordinates set to its ROE $1_m - e_1$. For other nodes A_i, we instead have a vector with first m coordinates set to 0, followed by a single coordinate 0 for bias and finally their respective ROE $1_m - e_i$.

Finally, we construct a layer of message passing. Specifically, $\phi_2^{(1)}$ is a 1-layer FNN with identity activation, no bias, and has the form of Equation 14. Its weight matrix is $\begin{bmatrix} C \in \mathbb{R}^{r\times(m+1)} & 0_{r\times m} \\ 0_{m\times(m+1)} & 0_{m\times m}\end{bmatrix}$, where each row $C_{i,:} = [\alpha_i, c_i] \in \mathbb{R}^{m+1}$. We use the sum aggregation operator and set $\phi_1^{(1)} = Id$. Moreover, $\varphi_{\text{Update}}^{(1)}$ has identity activation, $W_{\text{self}}^{(1)} = \begin{bmatrix} 0_{r\times(m+1)} & 0_{r\times m} \\ 0_{m\times(m+1)} & Id_{m\times m}\end{bmatrix}$, and $W_{\text{neigh}}^{(1)} = Id_{(r+m)\times(r+m)}$.

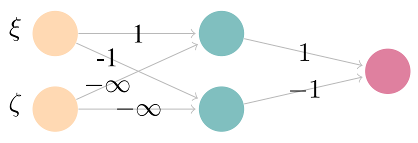
The following result follows immediately from Algorithm 4.

Proposition 14. *The embedding of vertex A_i after broadcasting with Algorithm 4 is $\left[p_1, p_2, ..., p_r, (1_m - e_i)^\top\right]^\top$, where p_i is the i^{th} affine combination (or tropical monomial) of f.*

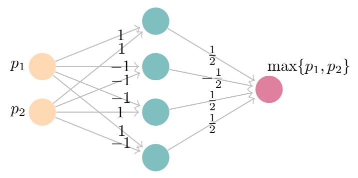
Furthermore, the Broadcast Algorithm 4 uses in total 2 FNN layers across 1 layer of message passing and $(2m+1)(r+m)$ parameters of which $(m+1)(r) = \mathcal{O}(rm)$ are learnable. □

6.3 Selection gadget

We now describe the selection gadget that we invoke to distribute r monomials (almost) evenly among the nodes such that the node A_i gets monomials $p_{\bar{i}} = \{p_k : k \mod m = i\}$. **Selection gadget**. It is a FNN with two layers that acts on the embeddings produced by the Broadcast gadget and utilizes

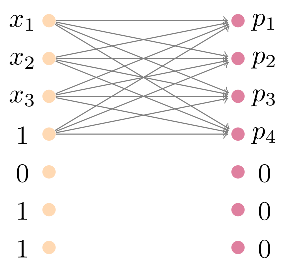


(a) Gadget for selecting ξ (Lemma 6)

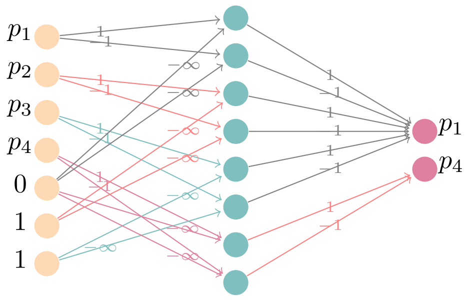


(b) Comparison gadget (following [83])

Figure 2: Orange, green, and magenta nodes represent input, hidden (with activation $\max\{\cdot, 0\}$) and output (with activation $\max\{\cdot, -\infty\}$) units respectively. **(Left)** ζ can be used as a control to either let ξ pass or filter it through the network. Note that in practice a sufficiently small negative weight can be used instead of $-\infty$. **(Right)** A gadget that yields the greater of its two inputs as the output.



(a) Broadcast gadget



(b) Selection gadget for selecting multiple inputs.

Figure 3: **(Left)** ϕ_2 in Broadcast gadget (shown here for $m = 3$ and $r = 4$ for node A_1) replicates all the monomials $p_1, \ldots, p_r$ across the nodes of a graph $\mathcal{G}$ (not shown). See Algorithm 4 in Appendix for weights and other details. **(Right)** Selection gadget builds on the base gadget from Fig. 2a, and filters out specific monomials at each node of $\mathcal{G}$, yielding a partition of the monomials across nodes.

the following result (Lemma 6) to filter out monomials such that each vertex A_i is left with only the monomials $p_{\bar{i}} = \{p_k : k \mod m = i\}$. The coordinates of each ROE serve as the control variables ζ for this purpose. Please see Algorithm 5 for details.

Lemma 6. *Consider any two-dimensional input with coordinates $\xi \geq 0$ and $\zeta \in \{0, 1\}$. There exists a 2-layer NN with ReLU activations (see Figure 2a) that outputs ξ when $\zeta = 0$, and 0 when $\zeta = 1$.*

Proof. If $\zeta = 0$, then the values at the two nodes in the hidden layer are ξ and $-\xi$ respectively before activation, and ξ and 0 after activation. Thus, the final output in this case is ξ.

On the other hand, if $\zeta = 1$, then the two nodes in the hidden layer take values $-\infty, -\infty$, respectively before activation, and thus 0 and 0 after activation. Thus, the final output in this case is 0. □

Proposition 15. *Node A_i, after selection with Algorithm 5, only contains the monomials $p_{\bar{i}}$ of f.*

Proof. Note that this result is just a generalization of Lemma 6, in the sense that we now distribute r monomials into batches of size m (with each monomial playing the role of ξ) across the m nodes with coordinates from ROE playing the role of ζ. The two layers return 0 when $\zeta = 1$ and ξ when $\zeta = 0$. Note that there is only one 0 in the i^{th} position of ROE $1_m - e_i$ for A_i, so A_i ends up with all the monomials $p_{\bar{i}}$ after the second layer. □

We are now ready to describe all the models, namely the Local, Global, Constant and Hybrid algorithms to compute a TSF $f : \mathbb{R}^m \to \mathbb{R}$.

6.4 Local MPNN and its complexity

Proposition 16. *Local MPNN can learn any TSF $f : \mathbb{R}^m \to \mathbb{R}$ with r monomials. It requires in total $\lceil \log_2(r/m) \rceil + 5$ FNN layers across 2 layers of MPNN, and $\mathcal{O}(rm)$ trainable parameters.*

Algorithm 5 Selection component: FNN layer

Input: any vector with ROE. In our case, it is $\left[p_1, p_2, ..., p_r, (1_m - e_i)^\top\right]^\top$

We create a 2-layer neural network as described below.

First layer: no bias, ReLU activation $\max(z, 0)$, and a weight matrix of dimensions $2r \times (r+m)$ with weights

$$w_{\kappa\kappa'} = \begin{cases} 1 & \text{if } \kappa' \leq r \text{ and } \kappa = 2\kappa' - 1 \\ -1 & \text{if } \kappa' \leq r \text{ and } \kappa = 2\kappa' \\ -\infty & \text{if } \kappa' \geq r \text{ and } \kappa \equiv 2(\kappa' - r), 2(\kappa' - r) - 1 \mod 2m \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Second layer: no bias, identity activation, and a weight matrix of dimensions $r' \times 2r$ with weights

$$w_{\kappa\kappa'} = \begin{cases} 1 & \text{if } \kappa = \left\lceil \frac{\kappa'}{2m} \right\rceil \text{ and } \kappa' \text{ odd,} \\ -1 & \text{if } \kappa = \left\lceil \frac{\kappa'}{2m} \right\rceil \text{ and } \kappa' \text{ even,} \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Algorithm 6 Local MPNN

Input: m-dimensional input x.

First, we perform a broadcast with Algorithm 4.

We need an additional layer of Message Passing. $\phi_2^{(2)}$ will have the form of (14). The first two layers of $\phi_2^{(2)}$ pertain to the selection gadget from Algorithm 5.

We then resort to $\lceil \log_2(r/m) \rceil + 1$ layers of local comparison (simultaneously comparing 2 nodes each time) according to the algorithm in [19] or [83] for $\phi_2^{(2)}$.

$\square^{(2)}$ is coordinate-wise max, $\phi_1^{(2)} = Id$, and $\varphi_{\text{Update}}^{(2)}$ has identity activation with $W_{\text{self}}^{(2)} = 0$ and $W_{\text{neigh}}^{(2)} = Id$.

Proof. As shown in Proposition 14, each node A_i has the embedding $\left[p_1, p_2, ..., p_r, (1_m - e_i)^\top\right]^\top$ after broadcasting. These embeddings are now filtered by the selection gadget, so by Proposition 15, each node A_i is left with $r' = \lceil \frac{r}{m} \rceil$ monomials $p_{\bar{i}} = \{p_k : k \mod m = i\}$. Thus, using $\lceil \log_2(r') \rceil + 1$ layers of comparison (see, e.g. [83]), we obtain the maximum $\max p_{\bar{i}}$. Max aggregation (coupled with the fact that we have an m-clique with self loops) helps us compute $\max_{i=1,...,m} p_{\bar{i}} = \max_{i=1,...,r} p_i = f(x)$. Note that the second MPNN-layer $\varphi^{(2)}$ involves only fixed parameters, and $\phi_2^{(2)}$ comprises $\lceil \log_2(r') \rceil + 3$ FNN layers. $\square$

6.5 Global MPNN and its complexity

Proposition 17. *Global MPNN can learn any TSF $f : \mathbb{R}^m \to \mathbb{R}$ with r monomials. It requires in total $3\lceil \log_m(r) \rceil + 2$ FNN-layers across $\lceil \log_m(r) \rceil + 1$ MPNN layers, and $\mathcal{O}(rm)$ trainable parameters.*

Proof. As shown in Proposition 14, after the Broadcast component, each node A_i has the embedding $\left[p_1, p_2, ..., p_r, (1_m - e_i)^\top\right]^\top$. Then by Proposition 15, the message from A_i after $\phi_2^{(2)}$ (modified Selection Component) is $p_{\bar{i}} = \{p_k : k \mod m = i\}$, appended by ROE. This vector is of dimension $r' + m = \lceil \frac{r}{m} \rceil + m$. Then, max aggregation (coupled with the fact that we have an m-clique with self loops) helps us compute $\max_{i=1,...,m} p_1, ..., p_m, \max_{i=1,...,m} p_{m+1}, ..., p_{2m}$, and so on, and output

Algorithm 7 Global MPNN

First, we perform a broadcast with Algorithm 4.

The second layer of MPNN is designed as follows. $\phi_2^{(2)}$ implements the selection gadget from 5 and uses aggregation of the form specified in (14). However, as a placeholder for the ROE, we append m rows with all coordinates set to 0 to the weight matrix for the second layer in Algorithm 5.Thus, the weight matrix is now of dimensions $(r' + m) \times 2r$.

$\square^{(2)}$ is coordinate-wise max and $\phi_1^{(2)} = Id$, $\varphi_{\text{Update}}^{(2)}$ uses identity activation, $W_{\text{neigh}}^{(2)} = Id$, and $W_{\text{self}}^{(2)} = \begin{bmatrix} 0 & 0 \\ 0 & Id_{m\times m} \end{bmatrix}$.

We then repeat this process (replacing r with r' until we are left with only 1 component and the final maximum f). This requires $\lceil \log_m(r) \rceil$ additional MPNN layers.

the same result for every node A_i besides the ROE which is different for the nodes. Similarly, in the subsequent MPNN layers, invoking the modified selection component and coordinate-wise max yields $\max_{i=1,\dots,r} p_i = f(x)$. Note that, except for the first MPNN, all the subsequent MPNN layers have only fixed, i.e., non-learnable parameters. Thus in total, we would need $\lceil \log_m(r) \rceil + 1$ MPNN layers, including the first Broadcast layer. $\square$

6.5.1 Constant MPNN and its complexity

Algorithm 8 Constant MPNN

First, we perform a broadcast with Algorithm 4.

We only need one additional message passing layer. $\phi_2^{(2)}$ however needs to be modified since for each node A_i, we would want to have $l_{z_{j,i}}$ for each $j = 1, .., q$ suggested by S_j instead of $l_{\bar{i}}$ that Proposition 15 guaranteed.

Specifically, in the first layer of $\phi_2^{(2)}$, we now have $2dr$ hidden nodes ($2d$ nodes for each l_i), ReLU activation, no bias, and a weight matrix $\in \mathbb{R}^{2dr\times(r+d)}$ with weights.

$$w_{\kappa\kappa'} = \begin{cases} 1 & \text{if } \kappa' \le k, \kappa' = \left\lceil \frac{\kappa}{2m} \right\rceil \text{ and } \kappa \text{ odd,} \\ -1 & \text{if } \kappa' \le k, \kappa' = \left\lceil \frac{\kappa}{2m} \right\rceil \text{ and } \kappa \text{ even,} \\ -\infty & \text{if } \kappa' > k \text{ and } \kappa \equiv 2(\kappa' - k), 2(\kappa' - k) - 1 \mod 2m, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

We now choose the linear pieces according to S_j. The second layer of $\phi_2^{(2)}$ has $2q$ hidden nodes, identity activation, no bias and weights $\in \mathbb{R}^{2q\times 2dr}$ given by

$$w_{\kappa\kappa'} = \begin{cases} 1 & \text{if } \kappa' = 2m(z_{\mu,i} - 1) + 2i - 1 \text{ for } i = 1, .., m \text{ and } \kappa = 2\mu + 1, \\ 1 & \text{if } 2m(z_{\mu-1,m+1} - 1) + 1 \le \kappa' \le 2m z_{\mu-1,m+1}, \kappa' \text{ odd, and } \kappa = 2\mu \\ -1 & \text{if } \kappa' = 2m(z_{\mu,i} - 1) + 2i \text{ for } i = 1, .., m \text{ and } \kappa = 2\mu + 1, \\ -1 & \text{if } 2m(z_{\mu-1,m+1} - 1) + 1 \le \kappa' \le 2m z_{\mu-1,m+1}, \kappa' \text{ even, and } \kappa = 2\mu, \\ 0 & \text{otherwise.} \end{cases}$$

For the next layers in $\phi_2^{(2)}$, we take the maximum of node $2\kappa + 1$ and node 2κ using two layers of the comparison gadget.

The aggregation function is coordinate-wise max, $\phi_1^{(2)} = Id$, $\varphi_{\text{Update}}^{(2)}$ uses identity activation and $W_{\text{self}}^{(2)} = 0$ and $[W_{\text{neigh}}^{(2)}]_j = s_j$.

Proposition 18. *The Constant algorithm 8 can learn any TSF $f : \mathbb{R}^m \to \mathbb{R}$ with r monomials. It requires in total 7 FNN layers across 2 layers of MPNN, and $\mathcal{O}(mrq)$ learnable parameters.*

Proof. z As before, after broadcasting, each node A_i comprises $\left[p_1, p_2, ..., p_r, (1_m - e_i)^\top\right]^\top$. Then, after processing with the first two layers of $\phi_2^{(2)}$, the output for node A_i is $\left[p_{z_{1,i}}, p_{z_{1,m+1}}, ..., p_{z_{q,i}}, p_{z_{q,m+1}}\right]^\top$. Thus, after processed by the last two layers, the message from node A_i is $\left[\max\{p_{z_{1,i}}, p_{z_{1,m+1}}\}, ..., \max\{p_{z_{q,i}}, p_{z_{k,m+1}}\}\right]^\top$. With the max operation, the aggregated message for all nodes is $\left[\max_{i \in S_1} l_i, ..., \max_{i \in S_q} l_i\right]^\top$. Finally, with s_j filling $W_{\text{neigh}}^{(2)}$, the resulting embedding for every node becomes $\sum_{j=1}^{q} s_j(\max_{i \in S_j} l_i)$, which is by [84, Theorem 1], $f(x)$. $\square$

6.5.2 Hybrid Architecture and its complexity

Algorithm 9 Hybrid Architecture

Input: m-dimensional input x

We start with a 1-layer FNN with m input nodes and r output nodes for the monomials. It has identity activation and its weight matrix $W \in \mathbb{R}^{r \times m}$ where each row $W_{i,:} = [\alpha_i \in \mathbb{R}^m]$ and its bias is the constants in the tropical monomials, i.e. $b_i = c_i$, thus, yielding the output $y = [p_1 \quad ... \quad p_r]^T$.

After that, we build an r-cliques A_1, ..., A_r, and bidirectional edges, and introduce loops with m self-edges from each vertex A_i to itself. We then put the initial embeddings $X = y^T = [p_1 \quad ... \quad p_r]$. We then have $\phi_2^{(1)}(h_{A_i}^{(0)}, h_{A_j}^{(0)}) = h_{A_i}^{(0)}$ and Aggregation operation to be coordinate-wise max, $\phi_1^{(2)} = Id$ to form $m_{A_i}^{(1)}$ (which is f now), and put $\varphi_{\text{Update}}^{(1)} = m_{A_i}^{(1)} = f$.

The following Proposition immediately follows.

Proposition 19. *The Hybrid algorithm 8 can learn any TSF $f : \mathbb{R}^m \to \mathbb{R}$ with r monomials. It requires in total 1 FNN layer and 1 message passing layer, and $\mathcal{O}(rm)$ learnable parameters.*

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [NA]

   Justification: Our work is purely theoretical and does not involve any experimental result.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: Our work is purely theoretical and does not any data and code.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [NA]

Justification: Our work is purely theoretical and does not involve any experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Our work is purely theoretical and does not involve any experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: Our work is purely theoretical and does not involve any experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This is no societal impact of the work performed as the work is purely theoretical.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.