BEYOND COUNTING LINEAR REGIONS OF NEURAL NETWORKS, SIMPLE LINEAR REGIONS DOMINATE!

Anonymous authors
Paper under double-blind review

ABSTRACT

Functions represented by a neural network with the widely-used ReLU activation are piecewise linear functions over convex regions (polytopes). Figuring out the properties of such polytopes is of fundamental importance for the development of neural networks. So far, either theoretical or empirical studies on such polytopes stay at the level of counting their number. Despite successes in explaining the power of depth and so on, counting the number of polytopes puts all polytopes on an equal booting, which is essentially an incomplete characterization of polytopes. To capture a more complete picture, here we study the shapes of polytopes formed by a network, with the minimum number of simplices to partition them into triangulations of polytopes. First, we demonstrate the properties of the number of simplices in triangulations of polytopes and compute the upper and lower bounds of the maximal number of simplices a network can generate. Next, by analyzing the number of simplices, we find that a ReLU network has surprisingly uniform and simple polytopes, although these polytopes theoretically can be rather diverse and complicated. This finding sheds light on why deep learning does not overfit. Lastly, we establish a theorem to illustrate why polytopes produced by a deep network are simple and uniform. The core idea for proof is counter-intuitive: adding depth probably does not create a more complicated polytope. We hope our work can inspire more research into investigating polytopes of a neural network in the future, thereby upgrading the knowledge of neural networks to a new level.

1 INTRODUCTION

It was shown in a thread of studies Chu et al. (2018); Balestriero & Baraniuk (2020); Hanin & Rolnick (2019b); Schonsheck et al. (2019) that a neural network with the piecewise linear activation is to partition the input space into many convex regions, mathematically referred to as polytopes, and each polytope is associated with a linear function (hereafter, we use convex regions, linear regions, and polytopes interchangeably). Hence, a neural network is essentially a piecewise linear function over the input domain. Based on this adorable result, the core idea of a variety of important theoretical advances and empirical findings is to turn the investigation of neural networks into the investigation of polytopes. By addressing basic questions such as how common operations affect the formation of polytopes (Zhang & Wu, 2020), how the network topology affects the number of polytopes (Cohen et al., 2016; Poole et al., 2016; Xiong et al., 2020), and so on, the understanding to expressivity of the networks are greatly deepened. To demonstrate the utility of the study on polytopes, we present two representative examples as follows:

The first representative example is the explanation to the power of depth. In the era of deep learning, many studies (Mohri et al., 2018; Bianchini & Scarselli, 2014; Telgarsky, 2015; Arora et al., 2016) attempted to explain why a deep network can perform superbly over a shallow one. One explanation to this question is on the superior representation power of deep networks, *i.e.*, a deep network can express a more complicated function but a shallow one with a similar size cannot (Cohen et al., 2016; Poole et al., 2016; Xiong et al., 2020). Their basic idea is to characterize the complexity of the function expressed by a neural network, thereby demonstrating that increasing depth can greatly maximize such a complexity measure compared to increasing width. Currently, one of the most popular complexity measures is the number of linear regions because it conforms to the functional structure of the widely-used ReLU networks. Pascanu et al. (2013) firstly proposed to

use the number of linear regions as the complexity measure. By directly applying Zaslavsky's Theorem (Zaslavsky, 1997), Pascanu et al. (2013) obtained a lower bound $\left(\prod_{l=0}^{L-1}\left\lfloor\frac{n_l}{n_0}\right\rfloor\right)\sum_{i=0}^{n_0}\binom{n_L}{i}$ for the maximal number of linear regions of a fully-connected ReLU network with n_0 inputs and L hidden layers of widths $n_1, n_2, \cdots, n_L$. Since this work, deriving the lower and upper bounds of the maximal number of linear regions becomes a hot topic (Montufar et al., 2014; Telgarsky, 2015; Montúfar, 2017; Serra et al., 2018; Croce et al., 2019; Hu & Zhang, 2018; Xiong et al., 2020). All these bounds suggest the expressive ability of depth.

The second interesting example is the finding of the high-capacity-low-reality phenomenon (Hu et al., 2021; Hanin & Rolnick, 2019b), that the theoretical tight upper bound for the number of polytopes is much larger than what is actually learned by a network, *i.e.*, deep ReLU networks have surprisingly few polytopes both at initialization and throughout the training. Specifically, Hanin & Rolnick (2019b) proved that the expected number of linear regions in a ReLU network is bounded by a function of the number of total neurons and the input dimension. This counter-intuitive phenomenon to some extent suggests why a deep network does not overfit. Although theoretically a lot of linear regions can be generated to learn a task, a deep network tends to find a simple function for a given task that is with few polytopes.

Despite figuring out the properties of polytopes of a neural network is of fundamental importance for the understanding of neural networks, the current studies to the polytopes have an important limit. So far, either theoretical or empirical studies only stay at the level of counting the number of polytopes, which blocks us from gaining other valuable findings. As we know, in a feed-forward network of L hidden layers, each polytope is encompassed by a group of hyperplanes, as shown in Figure 1(a), and each hyperplane is associated with a neuron. The details of how polytopes are formed by neurons of a ReLU network can be referred to Appendix C. Hence, any polytope is created by at most $\sum_{i=1}^{L} n_i$ and at least $n_0 + 1$ hyperplanes. Therefore, face numbers of polytopes can vary a lot. Unfortunately, the existing "counting" studies did not accommodate the differences among polytopes. Therefore, it is highly necessary to move a step forward, *i.e.*, know what each polytope is, thereby capturing a more complete picture of a neural network. To realize so, as a first attempt, we seamlessly divide each polytope into simplices in a triangulation of the polytope to describe the shape of polytopes by the minimum number of simplices to partition it, as Figure 1 shows. For example in $\mathbb{R}^2$, if a polytope comprises three simplices, it is a pentagon.

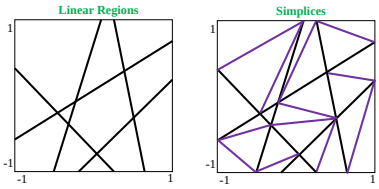

Figure 1: The minimum number of simplices a polytope contains can reveal the shape information of a polytope, with which one can dig out valuable information of a neural network.

In this manuscript, 1) to demonstrate the utility of the total number of simplices (#simplices) relative to the total number of polytopes (#polytopes), we characterize the basic proprieties and estimate the tight lower and upper bounds of the maximum #simplices for ReLU networks as well as networks using any piecewise linear activation. The key of bound estimation is to estimate the maximally achievable number of vertices. 2) We observe that polytopes formed by ReLU networks are surprisingly uniform and simple. Although theoretically quite diverse and complicated polytopes can be derived, simple polytopes dominate. This is another high-capacity-low-reality phenomenon, implying how fruitful it is to go beyond counting. Previously, Hanin & Rolnick (2019b) showed that deep ReLU networks have few polytopes. Here, we report that polytopes are not only few but also simple and uniform. Compared to (Hanin & Rolnick, 2019b), our observation more convincingly illustrates why deep networks do not overfit, *i.e.*, deep networks tend to find a function with a uniform and simple polytope pattern instead of a complicated polytope pattern. Showing the number of polytopes is few is insufficient to claim no overfitting because a neural network can have a small number of very complicated polytopes. 3) We establish a theorem that bounds the average face numbers of polytopes of a network to a small number under some mild assumption, thereby illustrating why polytopes produced by a deep network are simple and uniform. The core idea for the proof is counter-intuitive: Because each polytope is local, it is less likely for many hyperplanes to exactly intersect in a polytope. Thus, adding depth probably does not create a more complicated polytope.

To summarize, our contributions are threefold. 1) We point out the limitation of counting #polytopes. To solve it, we propose to use the #simplices to investigate the shape of polytopes, towards a more complete characterization of a ReLU network. Compared to the #polytopes, #simplices can define the networks of the same complexity and have a better association with networks' perfor-

mance in small-data scenarios. 2) We find that a ReLU network has surprisingly uniform and simple polytopes. Such an interesting finding can shed light on why deep networks tend not to overfit. 3) We mathematically derive an upper bound for the average face number of polytopes. Because the derived bound is small, it can explain why simple polytopes dominate.

2 Related Work

Studies on polytopes of a neural network. Besides the aforementioned works (Pascanu et al., 2013; Xiong et al., 2020; Montufar et al., 2014; Hu & Zhang, 2018) that count the number of polytopes, there are increasingly many studies on polytopes of neural networks. Chu et al. (2018); Hanin & Rolnick (2019b); Balestriero & Baraniuk (2020) showed that polytopes created by a network are convex. Zhang & Wu (2020) studied how different optimization techniques influence the local properties of polytopes, such as the inspheres, the directions of the corresponding hyperplanes, and the relevance of the surrounding regions. Hu et al. (2020) studied the network using an arbitrary activation function. They first used a piecewise linear function to approximate the given activation function. Then, they monitored the change of #polytopes to probe if the network overfits. Park et al. (2021) proposed the so-called neural activation coding that maximizes the number of linear regions to enhance the model's performance. Our work moves one step forward from counting the number of polytopes to considering the variability in shapes of polytopes, in hope to delineate a more complete picture of neural networks. Besides simplices, there should be other tools to describe the polytopes from different perspectives. We leave this as open questions for future exploration.

Model complexity measures. Characterizing the complexity of a neural network is the key to many important problems such as generalization (Lee et al., 2020). In addition to the #polytopes (Pascanu et al., 2013; Serra et al., 2018; Croce et al., 2019), which is a natural measure for ReLU networks, Bianchini & Scarselli (2014) used the Betti number of the nonnegative region formed by a network: $\{\mathbf{x} \mid f(\mathbf{x}) \geq 0\}$, where f is the function expressed by a network, as the complexity measure. Raghu et al. (2017) proposed a complexity measure that computes the length of the trajectory of the output produced by a network when the input moves along a one-dimensional unit-scale path. Such a measure well portrays how the structural properties of a network affect the function expressed by this network. Kileel et al. (2019) investigated the complexity measure of a deep polynomial network whose function space is algebraic and hence applicable to the tools of algebraic geometry. Their proposed measure is the algebraic dimension of the functional space associated with an architecture. Driven by the idea of training-free NAS (Huang et al., 2021), Chen et al. (2021) proposed the number of linear regions as an indicator of the model's capability. Their method can identify a well-performed network on the ImageNet within 4 GPU hours.

3 Problem Setup and Basic Properties

3.1 Problem Setup

Definition 1 (Simplicial complex)**.** *A D-simplex S is a D-dimensional convex hull provided by convex combinations of $D+1$ affinely independent vectors $\{\mathbf{v}_i\}_{i=0}^{D} \subset \mathbb{R}^D$. In other words, $S = \left\{ \sum_{i=0}^{D} \xi_i \mathbf{v}_i \mid \xi_i \geq 0, \sum_{i=0}^{D} \xi_i = 1 \right\}$. The convex hull of any subset of $\{\mathbf{v}_i\}_{i=0}^{D}$ is called a face of S. A simplicial complex $\mathcal{S} = \bigcup_{\alpha} S_\alpha$ is composed of a set of simplices $\{S_\alpha\}$ satisfying: 1) every face of a simplex from $\mathcal{S}$ is also in $\mathcal{S}$; 2) the non-empty intersection of any two simplices $S_1, S_2 \in \mathcal{S}$ is a face of both S_1 and S_2.*

Definition 2 (Polytopes of a neural network are convex)**.** *A neural network with ReLU activation partitions the input space into many polytopes. Each polytope corresponds to a collection of activation states of all neurons, and each polytope is convex (Chu et al., 2018). The detailed derivation is seen in Appendix B.*

Since each polytope of a ReLU network is convex, as Figure 1 shows, one can further divide each polytope into simplices in a triangulation of polytopes to make it a simplicial complex, where a

simplex is a fundamental unit. The number of simplices contained by a polytope can reflect the shape and complexity of the polytope.

Remark 1 (Computation of a polytope)**.** *Given a network of L hidden layers, a collection of activation states of all neurons leads to a group of inequalities. Mathematically, a polytope is defined as $\{\mathbf{x} \in \mathbb{R}^{n_0} \mid \mathbf{a}_k\mathbf{x}^\top + b_k \leq 0, k = 1, \cdots, \sum_{i=0}^{L-1} n_i\}$, where n_i is the number of neurons in the i-th layer. Given a group of inequalities, the vertices of the polytope can be computed based on the vertex enumeration algorithm (Avis & Fukuda, 1992). The details on how to compute a polytope are put into Appendix B.*

Definition 3 (#Simplices as a complexity measure)**.** *We define the complexity of a ReLU network as the minimum number of simplices: M that are needed to fill each and every polytope to support the piecewise linear function created by the ReLU network. The number of simplices contained by each polytope can be obtained by doing Delaunay triangulation [Chapter 23, (Toth et al., 2017)] with the vertices of each polytope.*

Proposition 1 (#Simplices in network transform (Fan et al., 2020))**.** *Suppose that the representation of an arbitrary ReLU network is $h : [-B,\ B]^D \to \mathbb{R}$, and M is the minimum #simplices to cover the polytopes to support h, for any $\delta > 0$, there exists a wide ReLU network $\mathbf{H}$ of width $\mathcal{O}\left[D(D+1)(2^D-1)M\right]$ and depth D, satisfying that*

$$\mathfrak{m}\Big(\mathbf{x} \mid h(\mathbf{x}) \neq \mathbf{H}(\mathbf{x})\}\Big) < \delta \tag{1}$$

where $\mathfrak{m}(\cdot)$ is the standard measure in $[-B,\ B]^D$.

3.2 Characterizing Properties of #Simplices

• **Reflecting the shape variability of polytopes**: We plot the histogram of the number of simplices a polytope contains for a randomly initialized network of the structure 3-20-10-1 in Figure 2. As can be seen, the most complicated polytope comprises 40 simplices, while the simplest one only has 3 simplices, suggesting that polytopes formed by the network vary significantly. Naturally, these polytopes should not be regarded equally complex. But when using #polytopes as a complexity measure, polytopes varying a lot are automatically treated equally. In contrast, since how many simplices a linear region comprises indicates how complex a linear region is, #simplices, as a complexity measure, addresses not only the number of polytopes but also the shape of each polytope, which is a more accurate characterization than #polytopes to the function computed by a network.

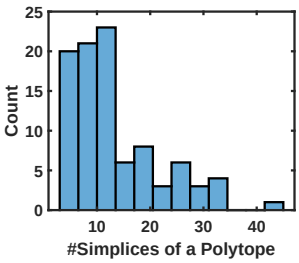

Figure 2: The histogram of #simplices with respect to different polytopes.

• **One-to-one correspondence via modularization and network transform**: A simplex is an elementary unit. Proposition 1 shows that an arbitrary ReLU network can be transformed into a wide network whose depth is the input dimension and width is determined by the #simplices. Therefore, as long as the support of functions represented by two networks is filled by the same #simplices, two networks can be represented by the network of the same width and depth. As such, we can define the equivalent class of networks as those with the same #simplices to cover their polytopes. In contrast, #polytopes cannot be used to rigorously define the equivalent networks because the polytopes are not elementary units. When one is told that two networks have the same #polytopes, one is still not totally sure whether two networks are equally complex.

• **A better association with networks' performance in small-data scenarios.** #Polytopes, as an expressivity metric, has shown to correlate with networks' performance in Chen et al. (2021), where the #polytopes associated with the available data is computed. However, when #polytopes is computed based on data, the number of data samples becomes the hard limit, *i.e.*, #polytopes is strictly upper bounded by the number of queried data. This, unfortunately, causes #polytopes to lose its discriminatory power when big data are unavailable. Because the pool of neural architectures is

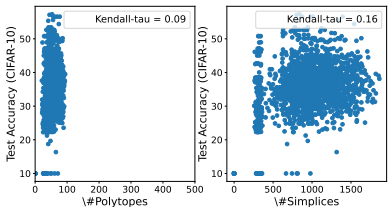

Figure 3: #Simplices has a better concordance with networks's performance than #polytopes over small data (the number of sampels=100).

large, most networks' #polytopes are similar and occupy a narrow region, regardless of their genuine expressive ability. As such, the agreement between #polytopes and the model's predictive performance turns low. In contrast, #simplices takes the shape of polytopes into account. Because the shapes of polytopes are diverse, #simplices is not bounded by the number of data, thereby enjoying a more accurate characterization of the model's expressivity. As a result, #simplices should gain a better concordance with networks than #polytopes over small data. Figure 3 shows that the #polytopes of different architectures is crowded in a narrow range, exhibiting a quasi-vertical character, while the concordance between the #simplices and the networks' test accuracy is higher. #Simplices may be more suitable than #polytopes for training-free NAS in small data scenarios (Elsken et al., 2020; Zhao et al., 2021).

• **Estimating the bound of the maximal #simplices.** Here, we estimate the upper and lower bounds of the maximal #simplices of a network with ReLU activation and any piecewise linear activation. Such results help us grasp the properties of #simplices and serve as a solid base for the future theoretical or empirical applications of #simplices, such as understanding the generalization behavior of a network, estimating the computation time when using #simplices in experiments, *etc.* The detailed proof is put into Appendix A.

Theorem 1 (Upper Bound)**.** *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with $n_1, n_2, \ldots, n_L$ hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at most*

$$\begin{cases} 2\binom{M}{d/2} + \mathcal{O}(M^{d/2-1}), & \text{if } d \text{ is even} \\ \binom{M}{(d+1)/2} + \mathcal{O}(M^{(d-1)/2}), & \text{if } d \text{ is odd,} \end{cases} \tag{2}$$

where $M = \sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d+\sum_{i=1}^{l} n_i}{d}$ *and* $m_i = \min\{d, n_1, n_2, \ldots, n_i\}$.

Theorem 2 (Lower Bound)**.** *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with n neurons in each layer. Then the maximal number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at least* $\Omega\left(\left(\frac{n}{d}\right)^{dL}\right)$.

4 Deep ReLU Networks Have Surprisingly Simple and Uniform Convex Regions

Previously, deep neural networks were both theoretically and empirically identified to have surprisingly fewer simplices than their theoretical maximal capacity (Hanin & Rolnick, 2019b). By analyzing the number of simplices a polytope contains, we observe that convex regions formed by ReLU networks are surprisingly simple and uniform. Although theoretically quite diverse convex regions can be derived, a small portion of simple convex regions dominate, which is another high-capacity-low-reality phenomenon (Hu et al., 2021). Consequently, combining the finding in (Hanin & Rolnick, 2019b), we upgrade the conclusion to that deep ReLU networks have surprisingly *few*, *simple*, and *uniform* convex regions. We validate our finding comprehensively at different initialization methods, network depths, sizes of the outer bounding box, biases, and input dimensions. Furthermore, we showcase that during the training, although the number of convex regions increases, convex regions keeps the uniformity. Our finding may explain why a deep learning model tends not to overfit: Although deep learning can form a complicated and diverse polytope pattern to classify the data, in reality, it only produce a simple and uniform polytope pattern for classification. Also, our finding suggests that increasing depth does not create many complicated polytopes. Since those polytopes can also be generated by shallow networks, in the future, the theory of depth might need to be refined to address what benefits the depth can give rise in reality.

To ensure the preciseness of the discovery and limited by the prohibitive computational cost of deriving simplices, our experiments are primarily based on the low-dimensional inputs. We also train networks on MNIST, however, therein polytopes are computed in a cross-section, as done in (Hanin & Rolnick, 2019b), *i.e.*, we find polytopes associated with a cross-section plane determined by two samples and calculate the number of simplices contained in each polytope.

4.1 INITIALIZATION

Initialization methods: We validate four popular initialization methods: orthogonal, Xavier uniform, Xavier normal[1], Kaiming initialization (He et al., 2015). For each initialization method, we use four different network architectures (3-40-20-1, 3-80-40-1). The bias values are set to 0.01 for all neurons. Later, we also discuss the influence of biases. Figure 4 shows the histogram of the #simplices each polytope has. Without loss of generality, suppose that in an experiment, the maximal #simplices a polytope has is Ω, we deem a polytope with no more than $\Omega/3$ as simple; otherwise, it is complex. The spotlight is that for all initialization methods and network structures, simple polytopes significantly dominate over complicated polytopes. We calculate that simple polytopes take account at least 57% and at most 76% of the total. In addition, among different initialization methods, the Xavier normal method tends to generate more uniform polytopes on four architectures. The achieved polytope is far simpler than the the theoretically most complicated polytope.

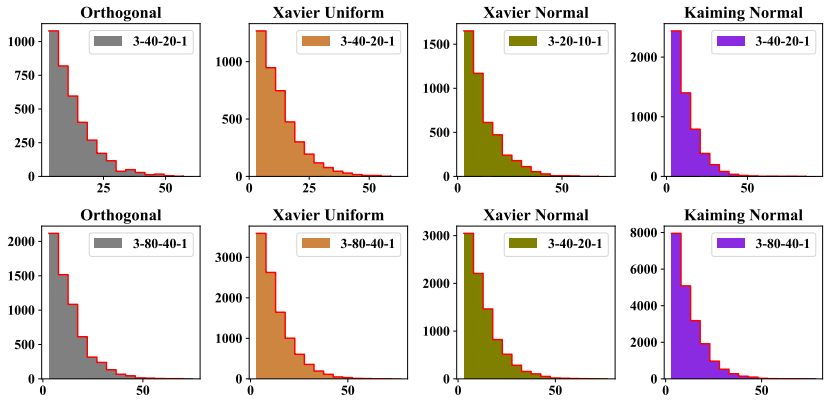

Figure 4: Deep ReLU networks have surprisingly uniform convex regions at different initialization methods.

Depths: Furthermore, we evaluate if the uniformity of polytopes still holds for deeper networks. This question is nontrivial, since a deeper network can theoretically generate more complicated polytopes. Will the depth break the uniformity? We choose four different widths (20, 40, 80, 160). The network initialization is the Xavier uniform. The depth is 6 and 9. As shown in Figure 5, when the network goes deep, the total number of polytopes increases. But simple poltopes still dominate.

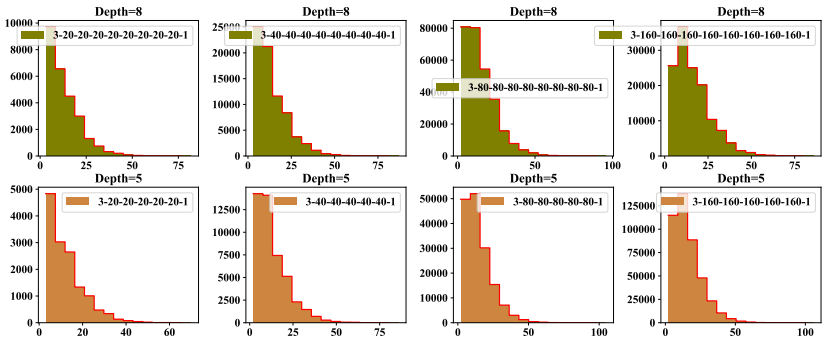

Figure 5: The uniformity holds true for deep networks.

Bounding boxes: In the above experiments, we set the outer bounding box as $[-1, 1]^d$, where d is the dimension. However, *will the size of the bounding box change the uniformity of convex regions?* Potentially, a larger bounding box will include more regions, and these regions may be complicated. To resolve this ambiguity, we derive the convex regions of the network 3-80-40-1 when setting the bounding box size to $[-1, 1]^3$, $[-2, 2]^3$, $[-5, 5]^3$, $[-10, 10]^3$, respectively. The results are in Figure 6. We find that the uniformity and simplicity of polytopes also hold for larger bounding boxes.

Biases: Here, we are curious about how the bias value of neurons will affect the distribution of polytopes. To address this issue, we set the bias values to $0, 0.01, 0.05, 0.1$, respectively for the

[1]https://pytorch.org/docs/stable/nn.init.html

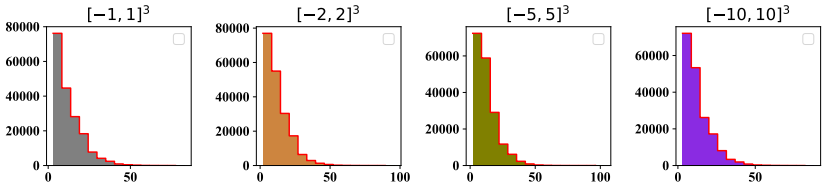

Figure 6: The uniformity holds true for larger bounding boxes.

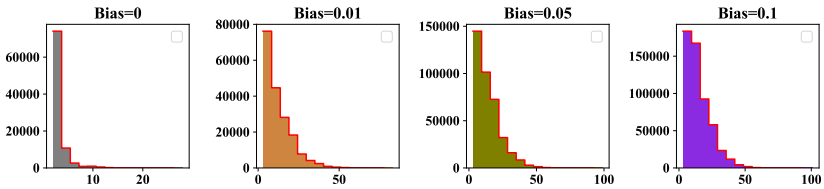

Figure 7: The uniformity holds true for different bias value.

network 3-80-40-1. The outer bounding box is $[-1,1]^3$. The initialization method is Kaiming normalization. As shown in Figure 7, we observe that as the bias value increases, more polytopes are produced. However, the number of simple polytopes still takes up the majority. It is worthwhile mentioning that when the bias equals 0, the uniformity is crystal clear. The bias=0 is the extremal case, where all hyperplanes intersect at the original point, and all polytopes are formed with the outer bounding box. In this circumstance, simple polytopes are easier to create.

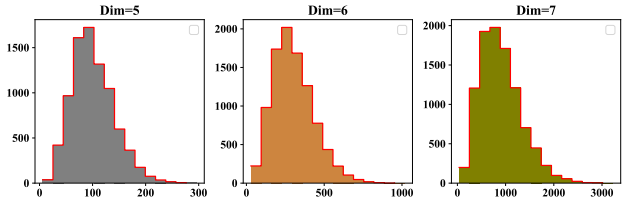

Figure 8: The uniformity holds true for different dimensions.

Input dimension: Here, we investigate how the input dimension will affect the distribution of polytopes. We compute the polytopes of three networks: 5-40-20-1, 6-40-20-1, and 7-40-20-1. We set the bias values to 0.01. The outer bounding box is $[-1,1]^d$, where d is the dimensionality. The initialization method is Kaiming normalization. Figure 8 shows the distribution of #simplices. As the dimension increases, a polytope tends to have much more simplices, while the total number of polytopes only slightly increases. The polytopes are still uniform, where the moderately simple polytopes still take a majority.

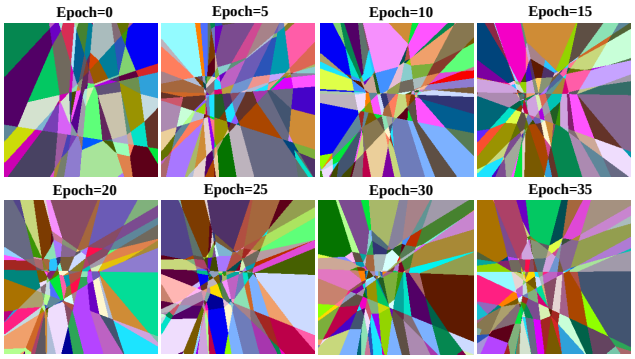

Figure 9: A cross-sectional visualization to the polytopes learned by a network at different epochs.

4.2 Training

In the earlier, we show that at the initialization stage, deep networks exhibit simple and uniform convex regions. Next, it is natural to ask *will the uniformity and simplicity of convex regions be*

broken during training? We answer this question by training a fully-connected network using max-out activation function on MNIST and visually show the polytopes. The ReLU and Leaky-ReLU activation functions are the special cases of max-out activation. Here, we can not compute the polytopes in 28×28 dimensional space because the vertex enumeration algorithm suffers the curse of dimensionality. Therefore, we visualize the polytopes in the cross-section plane. We initialize a network of size 784-7-7-6-10 with Kaiming normalization. The batch size is 128. The network is trained with Adam with a learning rate of 0.001. The total epoch number is 35.

Figure 9 shows the cross-section of the function learned by a network at different epochs. The cross-section is a plane that pass through two randomly-selected images from MNIST. Figure 9 shows that as the training goes, the number of polytopes increases. The hallmark is that almost all the polytopes are triangles or quadrilaterals. Although these polytopes are from a cross-section other than the whole landscape, one can indirectly sense the uniformity of these polytopes.

5 Mathematical Interpretation

Here, we mathematically explain why simple polytopes dominate in ReLU networks.

Geometric heuristics. We attribute the uniformity and simplicity of polytopes to the locality created by the ReLU activation function. Since a ReLU network divides the space into many local polytopes, to yield a complicated polytope from a local polytope, two or more hyperplanes associated with neurons in the later layers should intersect within the given local polytope, which is hard because the area of polytopes is typically small. As such, the complexity of polytopes probably does not increase as the network goes deeper. Our heuristics are also supported by (Hanin & Rolnick, 2019b), where a deep ReLU network was proved to have few polytopes because hyperplanes do not cross in a local polytope. Without crossing, complicated polytopes will not emerge either. Now, let us formalize our heuristic into mathematically rigorous bounds of the average face numbers of polytopes.

Theorem 3 (One-hidden-layer NNs)**.** *Let $\mathcal{N}$ be a one-hidden-layer fully-connected ReLU NN with d inputs and n hidden neurons, where d is a fixed positive integer. Suppose that n hyperplanes generated by n hidden neurons are in general position. Furthermore, assume that and n and B are large enough, then the average number of faces in linear regions of $\mathcal{N}$ is at most $2d+1$.*

Proof. Let $H_1, H_2, \ldots, H_{n+2d}$ be the $n+2d$ hyperplanes generated by n hidden neurons and the boundaries of $C(d, B)$. Then for each $1 \leq i \leq n+2d$, the hyperplane H_i may be intersected by another $n+2d-1$ hyperplanes. This will produce at most $n+2d-1$ hyperplanes in H_i, thus by Theorem 1, it will divide H_i into at most $\sum_{i=0}^{d-1} \binom{n+2d-1}{i}$ pieces since H_i can be consider as $\mathbb{R}^{d-1}$. Since there are $n+2d$ such hyperplanes, and each piece could be a face of two linear regions, finally we will get at most

$$2 \cdot (n+2d) \cdot \sum_{i=0}^{d-1} \binom{n+2d-1}{i}$$

faces for all the linear regions. On the other hand, since the n hidden neurons are in general position and B is large enough, we obtain that the total number of linear regions produced by $\mathcal{N}$ is $\sum_{i=0}^{d} \binom{n}{i}$. Therefore, the average number of faces in linear regions of $\mathcal{N}$ is at most

$$\frac{2 \cdot (n+2d) \cdot \sum_{i=0}^{d-1} \binom{n+2d-1}{i}}{\sum_{i=0}^{d} \binom{n}{i}} \leq \frac{2 \cdot (n+2d) \cdot \sum_{i=0}^{d-1} \binom{n+2d-1}{i}}{\sum_{i=0}^{d-1} \binom{n}{i+1}}.$$

For each $0 \leq i \leq d-1$, we have

$$\frac{2 \cdot (n+2d) \cdot \binom{n+2d-1}{i}}{\binom{n}{i+1}} \leq 2(i+1) \cdot \prod_{j=0}^{i} (1 + \frac{2d}{n-j}). \tag{3}$$

When n is large enough, $\prod_{j=0}^{i}(1+\frac{2d}{n-j})$ tends to 1, then the right hand side of Eq. (5) should be smaller than $2d+1$. Therefore, the average number of faces in linear regions of $\mathcal{N}$ is at most $2d+1$. □

Theorem 4 (Multi-layer NNs)**.** *Let $\mathcal{N}$ be an L-layer fully-connected ReLU NN with d inputs and n_i hidden neurons in the i-th hidden layer where d is a fixed positive integer. Suppose that for each linear region S produced by the first $(i-1)$-th layers of $\mathcal{N}$, the n_S hyperplanes in S generated by hidden neurons in the i-th layer are always in general position. Furthermore, assume that each n_S and B are large enough, then the average number of faces in linear regions of $\mathcal{N}$ is at most $2d+1$.*

Proof. Similar to the proof of Theorem 3, we obtain that for each each linear region S produced by the first $(i-1)$-th layers of $\mathcal{N}$, the average number of faces in linear regions of S after adding the i-th layer is at most

$$\frac{2\cdot(n_S+f)\cdot\sum_{i=0}^{d-1}\binom{n_S+f-1}{i}}{\sum_{i=0}^{d}\binom{n_S}{i}} \leq \frac{2\cdot(n_S+f)\cdot\sum_{i=0}^{d-1}\binom{n_S+f-1}{i}}{\sum_{i=0}^{d-1}\binom{n_S}{i+1}},$$

where f is the number of faces of S.

For each $0 \leq i \leq d-1$, we have

$$\frac{2\cdot(n_S+f)\cdot\binom{n_S+f-1}{i}}{\binom{n_S}{i+1}} \leq 2(i+1)\cdot\prod_{j=0}^{i}(1+\frac{f}{n_S-j}).$$

When n_S is large enough, $\prod_{j=0}^{i}(1+\frac{f}{n_S-j})$ should tend to 1, thus the right hand side of the last inequality should be smaller than $2d+1$. Therefore, the average number of faces in linear regions of $\mathcal{N}$ is also at most $2d+1$.

□

Remark 2. One desirable thing of this bound is that it is independent of depth, which validates our heuristics. Considering that $2d+1$ is a pretty small bound, Theorems 3 and 4 can justify why simple polytopes dominate. If the dominating polytopes are complex polytopes, the average face number should surpass $2d+1$ a lot. If simple polytopes only take up a small portion, the the average face number will be larger than $2d+1$, too. Although we assume that the network is wide in deriving the bound, we believe that the average face number should also be small for narrow networks.

6 Discussion and Conclusion

Decoding the learning of neural networks. Going from the linear regions to the simplices also contributes to a novel understanding to the training of a neural network. It has been clear that the learning of a neural network is the process of the function fitting with a piecewise linear function. The view of simplices connects with the finite element method (Bathe, 2007). Since a simplex is a elementary unit, we argue that the learning of a neural network is nothing but formulating finite elements towards solving the stationarity of the functional:

$$\Pi = \int_{\Omega} R(u(\mathbf{x})) + \sum_{k}\delta(\mathbf{x}-\mathbf{x}_k)(u(\mathbf{x})-\bar{u}(\mathbf{x}))d\mathbf{x}, \tag{4}$$

where $u(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{d} c_{ij}N_j(\mathbf{x})$, $\bar{u}(\mathbf{x})$ is the ground-truth function, $R(u(\mathbf{x}))$ denotes some regularization to $u(\mathbf{x})$, $N_j(\mathbf{x})$ is the shape function over a finite element (simplex) that is obtained through coordinate transform, and c_{ij} are coefficient solved by $\frac{\partial\Pi}{\partial c_{ij}} = 0, i = 1,2,\cdots,n, j = 1,2,\cdots,d$. The viewpoint of constructing finite elements bridges the computational physics with the deep learning, which could inspire more discoveries in the future.

In this manuscript, we have advocated studying the properties of polytopes beyond just counting them, towards revealing more useful properties of a neural network. To show such a direction is beneficial, we have characterized the desirable properties of #simplices and estimated its upper and lower bounds to pave a way for future applications. Then, we have presented that deep ReLU networks have surprisingly simple and uniform convex regions, which is a basic and valuable finding for ReLU networks, and may explain why deep networks do not overfit. Lastly, we mathematically established a small bound for the average number of faces in polytopes, which have convincingly explained our finding. In the future, more efforts should be put into investigating polytopes of a network and designing advanced algorithms to calculate the polytopes.

REFERENCES

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*, 8(3):295–313, 1992.

Randall Balestriero and Richard G Baraniuk. Mad max: Affine spline insights into deep learning. *Proceedings of the IEEE*, 109(5):704–727, 2020.

Imre Bárány and Zoltán Füredi. Empty simplices in euclidean space. *Canadian Mathematical Bulletin*, 30(4):436–445, 1987.

Klaus-Jürgen Bathe. Finite element method. *Wiley encyclopedia of computer science and engineering*, pp. 1–12, 2007.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.

Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1244–1253, 2018.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on learning theory*, pp. 698–728. PMLR, 2016.

Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2057–2066. PMLR, 2019.

Thomas Elsken, Benedikt Staffler, Jan Hendrik Metzen, and Frank Hutter. Meta-learning of neural architectures for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12365–12375, 2020.

Feng-Lei Fan, Rongjie Lai, and Ge Wang. Quasi-equivalence of width and depth of neural networks. *arXiv preprint arXiv:2002.02515*, 2020.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pp. 2596–2604, 2019a.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, pp. 359–368, 2019b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

Qiang Hu and Hao Zhang. Nearly-tight bounds on linear regions of piecewise linear neural networks. *arXiv preprint arXiv:1810.13192*, 2018.

Xia Hu, Weiqing Liu, Jiang Bian, and Jian Pei. Measuring model complexity of neural networks with curve activation functions. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1521–1531, 2020.

Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63(10):2585–2619, 2021.

Zhenhan Huang, Chunheng Jiang, Pin-Yu Chen, and Jianxi Gao. Network graph based neural architecture search. *arXiv preprint arXiv:2112.07805*, 2021.

Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 32, 2019.

Yoonho Lee, Juho Lee, Sung Ju Hwang, Eunho Yang, and Seungjin Choi. Neural complexity measures. *Advances in Neural Information Processing Systems*, 33:9713–9724, 2020.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Guido Montúfar. Notes on the number of linear regions of deep neural networks. *Sampling Theory Appl., Tallinn, Estonia, Tech. Rep*, 2017.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Yookoon Park, Sangho Lee, Gunhee Kim, and David Blei. Unsupervised representation learning via neural activation coding. In *International Conference on Machine Learning*, pp. 8391–8400. PMLR, 2021.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

Stefan Schonsheck, Jie Chen, and Rongjie Lai. Chart auto-encoders for manifold structured data. *arXiv preprint arXiv:1912.10094*, 2019.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pp. 4558–4566. PMLR, 2018.

Richard P. Stanley. An introduction to hyperplane arrangements. In *Lecture Notes, IAS/Park City Mathematics Institute*, 2004.

Richard P Stanley et al. An introduction to hyperplane arrangements. *Geometric combinatorics*, 13 (389-496):24, 2004.

Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*, pp. 10514–10523. PMLR, 2020.

T. Zaslavsky. *Facing up to arrangements : face-count formulas for partitions of space by hyperplanes*. Number 154 in Memoirs of the American Mathematical Society. American Mathematical Society, 1975.

T Zaslavsky. Facing up to arrangements: face-count formulas for partitions of space by hyperplanes. *Memoirs of American Mathematical Society*, 154:1–95, 1997.

Xiao Zhang and Dongrui Wu. Empirical studies on the properties of linear regions in deep neural networks. *arXiv preprint arXiv:2001.01072*, 2020.

Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *International Conference on Machine Learning*, pp. 12707–12718. PMLR, 2021.

A ESTIMATION TO THE MAXIMAL #SIMPLICES

A.1 PRELIMINARY

Let's recall some basic knowledge on hyperplane arrangements (Stanley et al., 2004). Let V be a Euclidean space. A hyperplane in the Euclidean space $V \simeq \mathbb{R}^n$, is a subspace $H := \{X \in V : \alpha \cdot X = b\}$, where $\mathbf{0} \neq \alpha \in V, b \in \mathbb{R}$ and "$\cdot$" denotes the inner product. A *region* of an arrangement $\mathcal{A} = \{H_i \subset V : 1 \leq i \leq m\}$ is just a connected component in the complement set of the union of all hyperplanes in the arrangement $\mathcal{A}$. Let $r(\mathcal{A})$ be the number of regions for an arrangement $\mathcal{A}$. Also, a *simplex* in an n-dimensional Euclidean space is just a n-dimensional polytope that is the convex hull of $n+1$ vertices. For example, a triangle is a simplex in $\mathbb{R}^2$, and a tetrahedron is a simplex in $\mathbb{R}^3$. A *triangulation* on some polytope is a division of the polytope into into simplices. Furthermore, we need the following definitions for linear regions.

Definition 4 (Activation Patterns and Linear Regions Hanin & Rolnick (2019a); Xiong et al. (2020))**.** *Suppose that $\mathcal{N}$ is a ReLU NN with L hidden layers. An activation pattern of $\mathcal{N}$ is a function $\mathcal{P}$ from the set of neurons to the set $\{1, -1\}$, i.e., for each neuron z in $\mathcal{N}$, we have $\mathcal{P}(z) \in \{1, -1\}$. Let θ be a fixed set of parameters in $\mathcal{N}$, and $\mathcal{P}$ be an activation pattern. Then the region corresponding to $\mathcal{P}$ and θ is*

$$\mathcal{R}(\mathcal{P}; \theta) := \{X^0 \in \mathbb{R}^{n_0^{(1)} \times n_0^{(2)} \times d_0} : z(X^0; \theta) \cdot \mathcal{P}(z) > 0, \quad \forall z \text{ a neuron in } \mathcal{N}\},$$

where $z(X^0; \theta)$ is the pre-activation of a neuron z. A linear region of $\mathcal{N}$ at θ is a non-empty set $\mathcal{R}(\mathcal{P}, \theta) \neq \emptyset$ for some activation pattern $\mathcal{P}$. Let $R_{\mathcal{N},\theta}$ be the number of linear regions of $\mathcal{N}$ at θ, i.e., $R_{\mathcal{N},\theta} := \#\{\mathcal{R}(\mathcal{P}; \theta) : \mathcal{R}(\mathcal{P}; \theta) \neq \emptyset$ for some activation pattern $\mathcal{P}\}$. Moreover, let $R_{\mathcal{N}} := \max_\theta R_{\mathcal{N},\theta}$ denote the maximal number of linear regions of $\mathcal{N}$ when θ ranges over $\mathbb{R}^{\#weights+\#bias}$.

The following Zaslavsky's Theorem is very crucial in the estimation of the number of linear regions.

Lemma 1 (Zaslavsky's Theorem (Zaslavsky, 1975; Stanley et al., 2004))**.** *Let $\mathcal{A}$ be an arrangement with m hyperplanes in $\mathbf{R}^n$. Then, the number $r(\mathcal{A})$ of regions for the arrangement $\mathcal{A}$ satisfies*

$$r(\mathcal{A}) \leq \sum_{i=0}^{n} \binom{m}{i}. \tag{5}$$

Furthermore, the above equality holds iff $\mathcal{A}$ is in general position Stanley (2004).

A.2 MAIN RESULTS - ONE LAYER RELU NNS

Throughout this paper, we always assume that the input space of an NN is a d-dimensional hypercube $C(d, B) := \{\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d : -B \leq x_i \leq B\}$ for some large enough constant B. Note that for a one-layer fully-connected ReLU NN, the pre-activation of each hidden neuron is an affine linear function of input values. Based on the sign of the pre-activation, each hidden neuron produces a hyperplane that divides the input space into two linear regions. On the other hand, the d-dimensional hypercube $C(d, B)$ has $2d$ hyperplanes in its boundary. Since at least d hyperplanes in $\mathbb{R}^d$ can create a intersection vertex, the following lemma is easy to prove.

Lemma 2. *Let $\mathcal{N}$ be a one-layer fully-connected ReLU NN with d input features and n hidden neurons. Then each hidden neuron determines a hyperplane in the input space $\mathbb{R}^d$. Also, the d-dimensional hypercube $C(d, B)$ has $2d$ hyperplanes in its boundary. Then the number of vertices that lie in the intersection of such d hyperplanes among all $n + 2d$ hyperplanes is at most $\binom{n+2d}{d}$.*

We will need the following estimation from (Bárány & Füredi, 1987, Theorem 4.1).

Lemma 3 (Bárány & Füredi (1987))**.** *Suppose that P is a set of n vertices in the d-dimensional Euclidean space $\mathbb{R}^d$. Let T be the convex hull of P. Then a triangulation of P contains at most*

$$\begin{cases} 2\binom{n}{d/2} + \mathcal{O}(n^{d/2-1}), & \text{if } d \text{ is even} \\ \binom{n}{(d+1)/2} + \mathcal{O}(n^{(d-1)/2}), & \text{if } d \text{ is odd} \end{cases}$$

simplices.

By Lemmas 2 and 3, we derive the following result.

Theorem 5. *Let $\mathcal{N}$ be a one-layer fully-connected ReLU NN with d input features and n hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at most*

$$\begin{cases} 2\binom{\binom{n+2d}{d}}{d/2} + \mathcal{O}(n^{d\cdot(d/2-1)}), & \text{if } d \text{ is even} \\ \binom{\binom{n+2d}{d}}{(d+1)/2} + \mathcal{O}(n^{d(d-1)/2}), & \text{if } d \text{ is odd} \end{cases}.$$

When $d = 2$ or 3, we derive the following corollaries.

Corollary 1. *Let $\mathcal{N}$ be a one layer fully-connected ReLU NN with 2 input features and n hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at most $2\binom{n+4}{2} + \mathcal{O}(1)$.*

Corollary 2. *Let $\mathcal{N}$ be a one layer fully-connected ReLU NN with 3 input features and n hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at most $\frac{n^6}{72} + \mathcal{O}(n^5)$.*

The following results gives a lower bound for the maximal number of simplices in a triangulation of for a one layer fully-connected ReLU NN.

Theorem 6. *Let $\mathcal{N}$ be a one-layer fully-connected ReLU NN with d input features and n hidden neurons. If n corresponding hyperplanes are in general position, then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at least $\sum_{j=0}^{d}\binom{n}{j}$.*

Proof. By Lemma 1, if n corresponding hyperplanes are in general position, then the number of bounded regions in the input space $\mathbb{R}$ is $\sum_{j=0}^{d}\binom{n}{j}$. It is obvious that the number of simplices in the triangulation is at least the number of bounded regions, thus the claim is proved. □

Exemplary Construction: Consider a one-hidden-layer ReLU network, Figure 10(a-c) shows what it is like when the maximal #simplices is achieved given a fixed number points of $\mathbb{R}^2$. The key is to constantly putting a new vertex into the interior of a simplex so as to generate as many simplices as possible, as Figure 10(c). However, as shown in Figure 10(d), such an upper bound is not reachable by a one-hidden-layer ReLU network, since hyperplanes pertaining to neurons will create undesirable extra vertices that do not conform to the optimal arrangement.

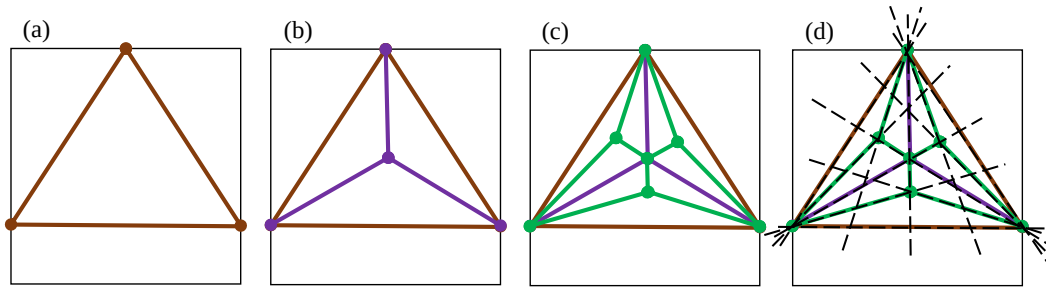

Figure 10: The arrangement to get the maximal #simplices cannot be accomplished by a ReLU network.

A.3 Main results - Multi-Layer ReLU NNs

To study the multi-layer NNs, we need the following results from (Montúfar, 2017, Proposation 3).

Lemma 4 (Montúfar (2017))**.** *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with $n_1, n_2, \ldots, n_L$ hidden neurons. Then the number of linear regions of $\mathcal{N}$ is at most $\prod_{i=1}^{L}\sum_{j=0}^{m_i}\binom{n_i}{j}$, where $m_i = \min\{d, n_1, n_2, \ldots, n_i\}$.*

We also need the following lemma for the number of $(d-1)$-dim faces in the linear regions of $\mathcal{N}$.

Lemma 5. *The number of $(d-1)$-dim faces in a linear regions of $\mathcal{N}$ is at most $2d + \sum_{l=1}^{L} n_l$.*

Proof. We can prove this claim by induction on the number L of layers of $\mathcal{N}$. When $L = 1$, the claim holds since we only have $2d + n_1$ hyperplanes in the input space $C(d, B)$, thus any linear regions will have at most $2d + n_1$ $(d-1)$-dim faces. Now suppose that this claim holds for the first $L-1$ layers (denote this network by $\mathcal{N}'$) of $\mathcal{N}$. Let P be a linear region of $\mathcal{N}'$, then by induction hypothesis we know P has at most $2d + \sum_{l=1}^{L-1} n_l$ $(d-1)$-dim faces. When considering the L-th layer, we will add at most n_L hyperplanes to the linear region P, thus each region in P will have at most $n_L + 2d + \sum_{l=1}^{L-1} n_l = 2d + \sum_{l=1}^{L} n_l$ $(d-1)$-dim faces, which completes the proof. □

The next results estimate the number of vertices that lie in the intersection of some hyperplanes determined by hidden neurons for multi-layer fully-connected ReLU NNs.

Lemma 6. *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with $n_1, n_2, \ldots, n_L$ hidden neurons. Then the number of vertices that lie in the intersection of some hyperplanes determined by hidden neurons and $C(d, B)$ is at most*

$$\sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d + \sum_{i=1}^{l} n_i}{d}, \tag{6}$$

where $m_i = \min\{d, n_1, n_2, \ldots, n_i\}$.

Proof. We will prove this result by induction. The case $L = 1$ is proved in Lemma 2. When $L \geq 2$, we assume that Eq. (6) holds for $L-1$. Thus by Lemma 4, Lemma 5, and the induction hypothesis, the network $\mathcal{N}'$ with the first $L-1$ layers already has $\prod_{i=1}^{L-1} \sum_{j=0}^{m_i} \binom{n_i}{j}$ linear regions and $\sum_{l=1}^{L-1} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d+\sum_{i=1}^{l} n_i}{d}$ vertices of intersection. Then when we add the L-th layer, for each linear region R in $\mathcal{N}'$, the n_L neurons create at most n_L hyperplanes in R, thus create $\binom{n_L+2d+\sum_{i=1}^{L-1} n_i}{d}$ more vertices of intersection in R. Therefore, we obtain that the total vertices of intersection by $\mathcal{N}$ is at most

$$\begin{aligned} &\sum_{l=1}^{L-1} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d + \sum_{i=1}^{l} n_i}{d} + \left(\prod_{i=1}^{L-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d + \sum_{i=1}^{L} n_i}{d} \\ = &\sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d + \sum_{i=1}^{l} n_i}{d}. \end{aligned}$$

□

By Lemmas 3 and 6, we derive the following result.

Theorem 7. *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with $n_1, n_2, \ldots, n_L$ hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at most*

$$\begin{cases} 2\binom{M}{d/2} + \mathcal{O}(M^{d/2-1}), & \text{if } d \text{ is even} \\ \binom{M}{(d+1)/2} + \mathcal{O}(M^{(d-1)/2}), & \text{if } d \text{ is odd,} \end{cases}$$

where $M = \sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} \binom{n_i}{j} \right) \cdot \binom{2d+\sum_{i=1}^{l} n_i}{d}$ and $m_i = \min\{d, n_1, n_2, \ldots, n_i\}$.

Corollary 3. *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with n hidden neurons in each layer. Then the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at most*

$$\mathcal{O}(L^{d(d+1)/2} \cdot n^{d(d+1)L/2}).$$

On the other hand, by the following lemma it is easy to derive the lower bounds for the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of multi-layer NNs.

Lemma 7 (Montufar et al. (2014)). *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with n_l hidden neurons in the l-th layer. Then the maximal number of linear regions of $\mathcal{N}$ is at least $\prod_{l=1}^{L-1} \lfloor \frac{n_l}{d} \rfloor^d \sum_{j=0}^{d} \binom{n_L}{j}$.*

Since the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ must be larger or equal to the number of linear regions of $\mathcal{N}$, by Lemma 7 we derive the following result.

Corollary 4. *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with n hidden neurons in each layer. Then the maximal number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at least $\Omega\left(\left(\frac{n}{d}\right)^{dL}\right)$.*

We empirically validate our bounds in Table 1 with 10 structures for a comprehensive evaluation. For a given MLP architecture, we initialize all the parameters based on the Xavier uniform initialization. Because all network structures we validate have a limited number of neurons, we can compute polytopes and their simplices by enumerating all collective activation states of neurons, which ensures that all polytopes are identifiable. For each structure, we repeat initialization ten times to report the maximal #simplices. As shown in Table 1, the derived upper and lower bounds are compatible with the numerical results of several network structures, which verifies the correctness of our results.

Table 1: Numerically verify the correctness of the derived upper and lower bounds for the maximal #simplices.

	3-5-3-1	3-8-3-1	3-10-3-1	3-5-5-1	3-8-5-1	3-10-5-1
Upper Bounds by Theorem 11	9629	63604	171104	14725	90481	234640
Estimation by Enumeration Method	266	448	1199	360	2021	2635
Lower Bound by Lemma 7	8	64	216	26	208	702
	5-5-3-1	5-8-3-1	5-10-3-1	5-5-5-1	5-8-5-1	5-10-5-1
Upper Bounds by Theorem 11	277179	4464999	21483566	483627	7344050	33912444
Estimation by Enumeration Method	9989	25689	42595	33295	59238	265651
Lower Bound by Lemma 7	7	7	224	32	32	1024

A.4 Comparison of Different Network Architectures

The aim of this section is to compare the the maximal #simplices based on bounds obtained in Section A.2. Our conclusion is that deep NNs usually have more number of simplices than shallow NNs with the same number of parameters.

For two functions $f(n)$ and $g(n)$, we write $f(n) = \mathcal{O}(g(n))$ if there exists some positive constant $c > 0$ such that $f(n) \leq cg(n)$ for all n larger than some constant; $f(n) = \Omega(g(n))$ if there exists some positive constant c such that $f(n) \geq cg(n)$ for all n large enough; and $f(n) = \Theta(g(n))$ if there exists some positive constants c_1, c_2 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all n large enough.

First, the number of parameters for the fully-connected ReLU NN $\mathcal{N}$ is easy to compute (Pascanu et al., 2013, Proposition 7).

Lemma 8. *Let $\mathcal{N}$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with n hidden neurons in each layer. Then the number of parameters in $\mathcal{N}$ is $\Theta(Ln^2)$.*

Let $S_{\mathcal{N}_1}$ be the maximal number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$. Now we can derive the number of simplices per parameter for deep NNs and their shallow counterparts. The following result follows directly from Corollaries 4-7 and Lemma 8.

Theorem 8. *Let $\mathcal{N}_1$ be a multi-layer fully-connected ReLU NN with d input features and L hidden layers with n hidden neurons in each layer, and $d = \mathcal{O}(1)$. Then $\mathcal{N}_1$ has $\Theta(Ln^2)$ parameters, and the ratio of $S_{\mathcal{N}_1}$ to the number of parameters of $\mathcal{N}_1$ is*

$$\frac{S_{\mathcal{N}_1}}{\#\text{ parameters of } \mathcal{N}_1} = \Omega\Big(\frac{1}{L} \cdot \frac{n^{dL-2}}{d^{dL}}\Big).$$

For a one-layer fully-connected ReLU NN $\mathcal{N}_2$ with d input features and Ln^2 hidden neurons, it has $\Theta(Ln^2)$ parameters, and the ratio for $\mathcal{N}_2$ is

$$\frac{S_{\mathcal{N}_2}}{\#\text{ parameters of } \mathcal{N}_2} = \mathcal{O}\left(\left(Ln^2\right)^{d(d+1)/2-1}\right).$$

It is easy to see that $\frac{S_{\mathcal{N}_1}}{\#\text{ parameters of }\mathcal{N}_1}$ grows at least exponentially fast with the number L of hidden layers and polynomially fast with the depth n of each hidden layer.

In contrast, $\frac{S_{\mathcal{N}_2}}{\#\text{ parameters of }\mathcal{N}_2}$ grows at most polynomially fast with L and n.

Therefore, we obtain that $\frac{S_{\mathcal{N}_1}}{\#\text{ parameters of }\mathcal{N}_1}$ is far larger than $\frac{S_{\mathcal{N}_2}}{\#\text{ parameters of }\mathcal{N}_2}$ when L and n are large enough. Therefore, we conclude that fully-connected ReLU NNs usually create much more number of simplices than one-layer fully-connected ReLU NNs with asymptotically the same number of input dimensions and parameters. This result suggests that fully-connected ReLU NNs usually have much more expressivity than one-layer fully-connected ReLU NNs.

B Extended to NNs with Piecewise Linear activation (PLNNs)

In this section, we generalize the bound estimation results to PLNNs (NNs with arbitrary piecewise linear activations). Currently, there lack effective tools to investigate the network using a non-piecewise linear activation. Approximating any activation function with a piecewise linear function is a viable proxy (Hu et al., 2021). In this light, we can tranlate the existing results of PLNNs to those networks. Our bound estimation results to PLNNs are valuable in terms of paving a way for such a line of work.

B.1 Main results - One Layer PLNNs

We adopt the following function $h(z)$ as the activation:

$$h(z) = \begin{cases} a_1 z + b_1, & \text{if } z \in I_1 \\ a_2 z + b_2, & \text{if } z \in I_2 \\ \vdots & \vdots \\ a_{q+1} z + b_{q+1}, & \text{if } z \in I_{q+1}, \end{cases} \tag{7}$$

where $q \geq 1$ is the number of breakpoints for $h(z)$, $I_1 = (-\infty, c_1]$, $I_2 = (c_1, c_2]$, $\ldots$, $I_{q+1} = (c_q, \infty)$ are some given intervals of $\mathbb{R}$, and $a_1, \ldots, a_{q+1}, b_1, \ldots, b_{q+1}$ are some given constants.

Lemma 9. *Let $\mathcal{N}$ be a one-layer fully-connected PLNN with d input features and n hidden neurons. Then each hidden neuron determines q hyperplanes in the input space $\mathbb{R}^d$. Also, the d-dimensional hypercube $C(d, B)$ has $2d$ hyperplanes in its boundary. Then the number of vertices that lie in the intersection of such d hyperplanes among all $qn + 2d$ hyperplanes is at most $q^n \cdot \binom{n+2d}{d}$.*

Proof. It is easy to see that each hidden neuron determines q parallel hyperplanes in the input space. Also, notice that any two parallel hyperplanes has no intersections. Therefore, if we want to calculate the number of vertices that lie in the intersection of any d hyperplanes among all $qn+2d$ hyperplanes, we can not pick two parallel hyperplanes. First, we have q^n choices to pick n hyperplanes from the qn determined by hidden neurons, such that these n hyperplanes are not parallel to each other. After that, for each such a choice, together with $2d$ hyperplanes determined by $C(d, B)$, we have $n + 2d$ hyperplanes and thus obtain at most $\binom{n+2d}{d}$ intersection vertices. Therefore, the total number of intersection vertices is at most $q^n \cdot \binom{n+2d}{d}$. □

By Lemmas 3 and 9, we derive the following result.

Theorem 9. *Let $\mathcal{N}$ be a one layer fully-connected PLNN with d input features and n hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at most*

$$\begin{cases} 2\binom{q^n \cdot \binom{n+2d}{d}}{d/2} + \mathcal{O}((q^n \cdot n^d)^{(d/2-1)}), & \textit{if } d \textit{ is even} \\ \binom{q^n \cdot \binom{n+2d}{d}}{(d+1)/2} + \mathcal{O}((q^n \cdot n^d)^{(d-1)/2}), & \textit{if } d \textit{ is odd} \end{cases}.$$

When $d = 2$ or 3, we derive the following corollaries.

Corollary 5. *Let $\mathcal{N}$ be a one-layer fully-connected PLNN with 2 input features and n hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at most $2q^n \cdot \binom{n+4}{2} + \mathcal{O}(1)$.*

Corollary 6. *Let $\mathcal{N}$ be a one-layer fully-connected PLNN with 3 input features and n hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at most $q^3 \cdot \frac{n^6}{72} + \mathcal{O}(q^n \cdot n^5)$.*

The following results offer a lower bound for the maximal number of simplices in a triangulation of a one-layer fully-connected PLNN.

Theorem 10. *Let $\mathcal{N}$ be a one-layer fully-connected PLNN with d input features and n hidden neurons. If any n relevant hyperplanes corresponding to distinct hidden neurons are in general position, then the number of simplices in a triangulation of intersection vertices among all n corresponding hyperplanes is at least $\sum_{j=0}^{d} q^j \cdot \binom{n}{j}$.*

Proof. By [Xiong2, Corollary], if any n corresponding hyperplanes corresponding to distinct hidden neurons are in general position, then the number of bounded regions in the input space $\mathbb{R}$ is $\sum_{j=0}^{d} q^j \cdot \binom{n}{j}$. Again it is obvious that the number of simplices in the triangulation is at least the number of bounded regions, thus the claim is proved. □

B.2 Main results - Multi-Layer PLNNs

Next, we consider the multi-layer PLNN case. We have the following results whose proofs are similar to Lemma 5.

Lemma 10. *The number of $(d-1)$-dim faces in a linear regions of PLNN $\mathcal{N}$ is at most $2d + \sum_{l=1}^{L} 2n_l$.*

The next results estimate the number of vertices that lie in the intersection of some hyperplanes determined by hidden neurons for multi-layer fully-connected PLNNs.

Lemma 11. *Let $\mathcal{N}$ be a multi-layer fully-connected PLNN with d input features and L hidden layers with $n_1, n_2, \ldots, n_L$ hidden neurons. Then the number of vertices that lie in the intersection of some hyperplanes determined by hidden neurons and $C(d, B)$ is at most*

$$\sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j} \right) \cdot \left(q^{n_l} \cdot \binom{n_l + 2d + \sum_{i=1}^{l-1} 2n_i}{d} \right), \tag{8}$$

where $m_i = \min\{d, n_1, n_2, \ldots, n_i\}$.

Proof. We will prove this result by induction. The case $L = 1$ is proved in Lemma 9. When $L \geq 2$, we assume that Eq. (8) holds for $L - 1$. Thus by the induction hypothesis, the network $\mathcal{N}'$ with the first $L-1$ layers already has $\prod_{i=1}^{L-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j}$ linear regions and $\sum_{l=1}^{L-1} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j} \right) \cdot \binom{n_l + 2d + \sum_{i=1}^{l-1} 2n_i}{d}$ vertices of intersection. Then when we add the L-th layer, for each linear region R in $\mathcal{N}'$, the n_L neurons create at most n_L hyperplanes in R, therefore creating $q^{n_L} \cdot \binom{n_L + 2d + \sum_{i=1}^{L-1} 2n_i}{d}$ more vertices of intersection in R. Finally, we obtain that the total vertices of intersection by $\mathcal{N}$ is at most

$$\begin{aligned}
&\sum_{l=1}^{L-1} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j} \right) \cdot \left(q^{n_l} \cdot \binom{n_l + 2d + \sum_{i=1}^{l-1} 2n_i}{d} \right) \\
&+ \left(\prod_{i=1}^{L-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j} \cdot \right) \left(q^{n_L} \cdot \binom{n_L + 2d + \sum_{i=1}^{L-1} 2n_i}{d} \right) \\
&= \sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j} \right) \cdot \left(q^{n_l} \cdot \binom{n_l + 2d + \sum_{i=1}^{l-1} 2n_i}{d} \right).
\end{aligned}$$

□

By Lemmas 3 and 6, we derive the following result.

Theorem 11. *Let $\mathcal{N}$ be a multi-layer fully-connected PLNN with d input features and L hidden layers with $n_1, n_2, \ldots, n_L$ hidden neurons. Then the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at most*

$$\begin{cases} 2\binom{M}{d/2} + \mathcal{O}(M^{d/2-1}), & \text{if } d \text{ is even} \\ \binom{M}{(d+1)/2} + \mathcal{O}(M^{(d-1)/2}), & \text{if } d \text{ is odd} \end{cases},$$

where

$$M = \sum_{l=1}^{L} \left(\prod_{i=1}^{l-1} \sum_{j=0}^{m_i} q^j \cdot \binom{n_i}{j} \right) \cdot \left(q^{n_l} \cdot \binom{n_l + 2d + \sum_{i=1}^{l-1} 2n_i}{d} \right)$$

and $m_i = \min\{d, n_1, n_2, \ldots, n_i\}$.

Corollary 7. *Let $\mathcal{N}$ be a multi-layer fully-connected PLNN with d input features and L hidden layers with n hidden neurons in each layer. Then the number of simplices in a triangulation of intersection vertices among all corresponding hyperplanes of $\mathcal{N}$ is at most*

$$\mathcal{O}(q^{(Ld+n)(d+1)/2} \cdot L^{d(d+1)/2} \cdot n^{d(d+1)L/2}).$$

C Simplices, Polytopes, and Their Computation

C.1 Notations of a Network

For convenience and consistency, we inherit notations from (Chu et al., 2018). For a ReLU network that contains L hidden layers, we write the l-th layer as $\mathcal{L}_l$. Specially, $\mathcal{L}_0$ is the input layer, $\mathcal{L}_{L+1}$ is the output layer, and the other layers $\mathcal{L}_l, l \in \{1, 2, \ldots, L\}$ are hidden layers. Hidden layers' neurons are called hidden neurons. Let n_l represent the number of neurons in $\mathcal{L}_l$.

Denote by $\mathbf{b}_i^{(l)}$ its bias, by $\mathbf{a}_i^{(l)}$ its output, and by $\mathbf{z}_i^{(l)}$ the total weighted sum of its inputs plus the bias. For all the n_l neurons in $\mathcal{L}_l$, we arrange all their biases into a vector $\mathbf{b}^{(l)} = \left[\mathbf{b}_1^{(l)}, \ldots, \mathbf{b}_{n_l}^{(l)}\right]^\top$, their outputs into a vector $\mathbf{a}^{(l)} = \left[\mathbf{a}_1^{(l)}, \ldots, \mathbf{a}_{n_l}^{(l)}\right]^\top$, and their inputs into a vector $\mathbf{z}^{(l)} = \left[\mathbf{z}_1^{(l)}, \ldots, \mathbf{z}_{n_l}^{(l)}\right]^\top$. Neurons in two neighbor layers are linked by weighted edges. Denote by $W_{ij}^{(l)}$ the weight of the edge between the i-th neuron in $\mathcal{L}_{l+1}$ and the j-th neuron in $\mathcal{L}_l$. For $l \in \{1, \ldots, L\}$, we compute $\mathbf{z}^{(l+1)}$ by

$$\mathbf{z}^{(l+1)} = W^{(l)}\mathbf{a}^{(l)} + \mathbf{b}^{(l)}, \tag{9}$$

where $W^{(l)}$ is an n_{l+1}-by-n_l matrix.

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the ReLU activation function. We have $\mathbf{a}_i^{(l)} = \sigma\left(\mathbf{z}_i^{(l)}\right)$ for all $l \in \{1, \ldots, L\}$. In in an element-wise fashion, we write $\sigma\left(\mathbf{z}^{(l)}\right) = \left[\sigma\left(\mathbf{z}_1^{(l)}\right), \ldots, \sigma\left(\mathbf{z}_{n_l}^{(l)}\right)\right]^\top$. Then, for $l \in \{1, \ldots, L\}$, we have

$$\mathbf{a}^{(l)} = \sigma\left(\mathbf{z}^{(l)}\right). \tag{10}$$

The input is $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^d$, and $\mathbf{x}_i$ is the i-th dimension of $\mathbf{x}$. The input layer $\mathcal{L}_0$ contains $n_0 = d$ nodes, where $\mathbf{a}_i^{(0)} = \mathbf{x}_i, i \in \{1, \ldots, d\}$. The output of the network is $\mathbf{a}^{(L+1)} \in \mathcal{Y}$, where $\mathcal{Y} \subseteq \mathbb{R}^{n_{L+1}}$. The output layer $\mathcal{L}_{L+1}$ employs the softmax function: $\mathbf{a}^{(L+1)} = \mathrm{softmax}\left(\mathbf{z}^{(L+1)}\right)$.

C.2 DERIVING POLYTOPES OF A NETWORK

For the i-th hidden neuron in $\mathcal{L}_l$, $\sigma\left(\mathbf{z}_i^{(l)}\right)$ is in the following form:

$$\sigma\left(\mathbf{z}_i^{(l)}\right) = \begin{cases} \mathbf{z}_i^{(l)}, & \text{if } \mathbf{z}_i^{(l)} \geq 0 \\ 0, & \text{if } \mathbf{z}_i^{(l)} < 0, \end{cases} \tag{11}$$

where $\sigma\left(\mathbf{z}_i^{(l)}\right)$ consists of two parts. Given a network, an instance $\mathbf{x} \in \mathcal{X}$ determines the value of $\mathbf{z}_i^{(l)}$, and further determines $\sigma\left(\mathbf{z}_i^{(l)}\right) = 0$ or $\sigma\left(\mathbf{z}_i^{(l)}\right) = \mathbf{z}_i^{(l)}$. According to which part $\sigma\left(\mathbf{z}_i^{(l)}\right)$ falls into, one can encode the activation status of each hidden neuron by two states, each of which uniquely corresponds to one part of $\sigma\left(\mathbf{z}_i^{(l)}\right)$. Denote by $\mathbf{c}_i^{(l)} \in \{1, 0\}$ the state of the i-th hidden neuron in $\mathcal{L}_l$, we have $\mathbf{z}_i^{(l)} \geq 0$ if and only if $\mathbf{c}_i^{(l)} = 1$ and $\mathbf{z}_i^{(l)} < 0$ if and only if $\mathbf{c}_i^{(l)} = 0$. The states of different hidden neurons usually differ from each other.

Let $\mathbf{c}^{(l)} = \left[\mathbf{c}_1^{(l)}, \ldots, \mathbf{c}_{n_l}^{(l)}\right]$ be the states of all hidden neurons in $\mathcal{L}_l$ and $\mathbf{C} = \left[\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(L)}\right]$ specify the collective states of all hidden neurons. $\mathbf{C}$ of a given fixed network is uniquely determined by the instance $\mathbf{x}$. We write the function that maps an instance $\mathbf{x} \in \mathcal{X}$ to a configuration $\mathrm{C} \in \{1, 0\}^N$ as $\mathrm{conf} : \mathcal{X} \to \{1, 0\}^N$, where $N = \sum_{i=0}^{L-1} n_l$. Then, we rewrite Eq. (10) as

$$\mathbf{a}^{(l)} = \sigma\left(\mathbf{z}^{(l)}\right) = \mathbf{c}^{(l)} \circ \mathbf{z}^{(l)}, \tag{12}$$

where $\mathbf{c}^{(l)} \circ \mathbf{z}^{(l)}$ is the Hadamard product between $\mathbf{c}^{(l)}$ and $\mathbf{z}^{(l)}$.

By plugging $\mathbf{a}^{(l)}$ into Eq. (9), we rewrite $\mathbf{z}^{(l+1)}$ as

$$\mathbf{z}^{(l+1)} = W^{(l)}\left(\mathbf{c}^{(l)} \circ \mathbf{z}^{(l)}\right) + \mathbf{b}^{(l)} = \tilde{W}^{(l)}\mathbf{z}^{(l)} + \tilde{\mathbf{b}}^{(l)}, \tag{13}$$

where $\tilde{\mathbf{b}}^{(l)} = \mathbf{b}^{(l)}$, and $\tilde{W}^{(l)} = W^{(l)} \circ \mathbf{c}^{(l)}$ is the generalized Hadamard product, such that the entry at the i-th row and j-th column of $\tilde{W}^{(l)}$ is $\tilde{W}_{ij}^{(l)} = W_{ij}^{(l)}\mathbf{c}_j^{(l)}$.

By iteratively enforcing Eq. (13), we can write $\mathbf{z}^{(l+1)}, l \in \{1, \ldots, L\}$ as

$$\mathbf{z}^{(l+1)} = \left(\prod_{k=1}^{l} \tilde{W}^{(k)}\right)\mathbf{z}^{(1)} + \sum_{h=1}^{l}\left(\prod_{k=h+1}^{l} \tilde{W}^{(k)}\right)\tilde{\mathbf{b}}^{(h)}. \tag{14}$$

Substituting $\mathbf{z}^{(1)} = W^{(0)}\mathbf{x} + \mathbf{b}^{(1)}$ into the above equation, we rewrite $\mathbf{z}^{(l+1)}, l \in \{1, \ldots, L\}$ as

$$\begin{aligned} \mathbf{z}^{(l+1)} &= \left(\prod_{k=1}^{l} \tilde{W}^{(k)}\right)(W^{(0)}\mathbf{x} + \mathbf{b}^{(1)}) + \sum_{h=1}^{l}\left(\prod_{k=h+1}^{l} \tilde{W}^{(k)}\right)\tilde{\mathbf{b}}^{(h)} \\ &= \left(\prod_{k=0}^{l} \tilde{W}^{(k)}\right)\mathbf{x} + \mathbf{b}^{(1)}\left(\prod_{k=1}^{l} \tilde{W}^{(k)}\right) + \sum_{h=1}^{l}\left(\prod_{k=h+1}^{l} \tilde{W}^{(k)}\right)\tilde{\mathbf{b}}^{(h)} \\ &= \hat{W}^{(0:l)}\mathbf{x} + \hat{\mathbf{b}}^{(0:l)}, \end{aligned} \tag{15}$$

where $\hat{W}^{(0:l)}$ is the coefficient matrix of $\mathbf{x}$, and $\hat{\mathbf{b}}^{(0:l)}$ is the sum of the remaining bias terms. Naturally, $F(\mathbf{x})$ is

$$F(\mathbf{x}) = \mathrm{softmax}\left(\hat{W}^{(0:L)}\mathbf{x} + \hat{\mathbf{b}}^{(0:L)}\right). \tag{16}$$

For a fixed network and a fixed instance $\mathbf{x}, \hat{W}^{(0:l)}$ and $\hat{\mathbf{b}}^{(0:l)}$ are constant parameters uniquely determined by activation states of all neurons: $\mathbf{C} = \mathrm{conf}(\mathbf{x})$. Furthermore, according to Eq. (11), $\mathbf{C}$ leads to a group of inequalities:

$$(2\mathbf{c}^{(l+1)} - 1) \circ (\hat{W}^{(0:l)}\mathbf{x} + \hat{\mathbf{b}}^{(0:l)}) \geq 0,\ l = 0, \cdots, L-1, \tag{17}$$

which encompasses a convex polytope according to the H-definition of polytopes (Chapter 16, Toth et al. (2017)). If $\mathrm{conf}(\mathbf{x}_1) = \mathrm{conf}(\mathbf{x}_2)$, $\mathbf{x}_1$ and $\mathbf{x}_2$ lie in the same polytope due to the uniqueness.

C.3 Computing #Simplices and Polytopes with Vertex Enumeration Algorithm

Next, what is essential is to numerically compute #simplices. Unfortunately, the exact computation relies on enumerating all collective states of neurons, which is only possible i) when the total number of neurons in a network is small or ii) when the input dimension is low. For the former, we can enumerate possible activation states of all neurons by repeating $2^{\sum_{i=0}^{L-1} n_i}$ times, while for the latter, we can uniformly sample the input space by $(\frac{1}{\epsilon})^d$ times, where ϵ is the needed sampling interval, to identify the actual collective states of all neurons. As highlighted earlier, the collective activation states $\mathbf{C}$ lead to a group of inequalities, corresponding to a polytope. We can solve all vertices of the polytope with the vertex enumeration algorithm [2] (Avis & Fukuda, 1992) to find all vertices of a polytope. The complexity of the vertex enumeration algorithm scales linearly with the number of inequalities. Then, Delaunay triangulation [Chapter 23, (Toth et al., 2017)] is executed for these vertices to see how many non-overlapping simplices these vertices can gain. Lastly, we count the #simplices. Sometimes, one may not need to compute all polytopes of a neural network. Instead, the polytopes related to data are sufficient. For example, in the training-free NAS experiment (Chen et al., 2021), the authors exhausted all training data and identify associated unique polytopes.

Algorithm 1 Calculate the #simplex of a polytope of a neural network

1: Identify a collective activation state of all neurons
2: Derive a group of inequalities whose hyperplanes encompass the targeted polytope
3: Vertex enumeration algorithm to derive the vertices of the polytope
4: Delaunay triangulation
5: Count #simplices

D Supplementary Experiments at Five Dimensional Space

A majority of experiments in the main body are regarding three-dimensional inputs. To evaluate the simplicity and uniformity of polytopes produced by a neural network comprehensively, we supplement similar experiments on five-dimensional inputs. All experimental settings are the same as the main body.

Figure 11 shows that regardless of different initialization methods, the simple linear regions dominate in five-dimensional space.

Figure 12 shows that different bias values do not undermine the simplicity and uniformity of polytopes in five-dimensional space.

Figure 13 demonstrates that in five-dimensional space, the simple polytopes still dominate for a deep network with different widths.

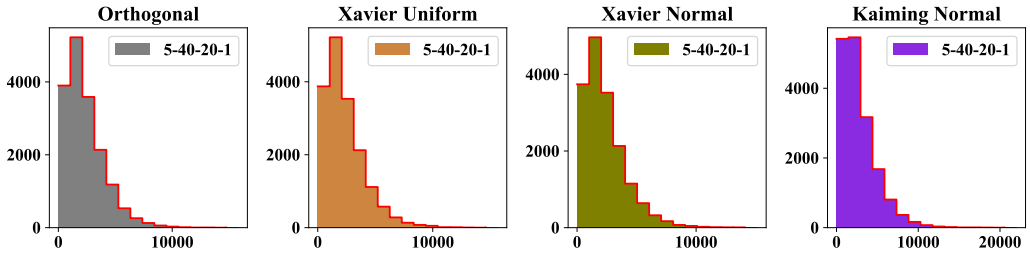

Figure 11: The simple polytopes dominate for different initialization methods in 5 dimensional space.

[2] https://pypi.org/project/pypoman/

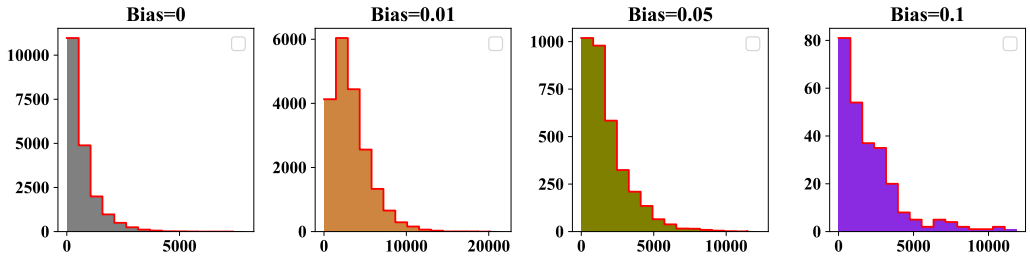

Figure 12: The simplicity and uniformity is kept for different biases in 5 dimensional space.

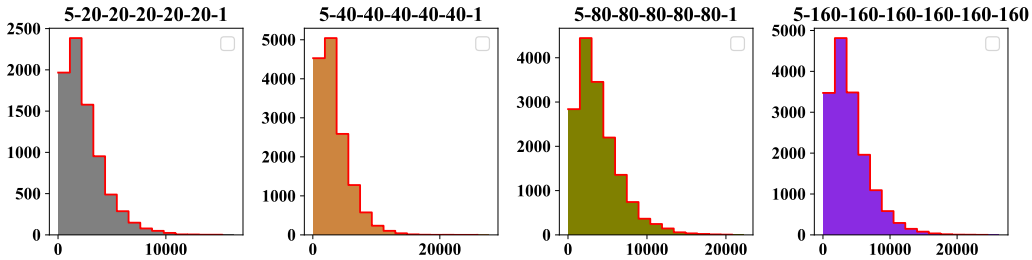

Figure 13: Deep networks also hold simplicity and uniformity in 5 dimensional space.