

FAST AND EXACT ENUMERATION OF DEEP NETWORKS PARTITIONS REGIONS

Randall Balestrieri

Meta AI, FAIR
rbalestrieri@meta.com
NYC, USA

Yann LeCun

Meta AI, FAIR, NYU
yann@meta.com
NYC, USA

ABSTRACT

One fruitful formulation of Deep Networks (DNs) enabling their theoretical study and providing practical guidelines to practitioners relies on Piecewise Affine Splines. In that realm, a DN’s input-mapping is expressed as per-region affine mapping where those regions are implicitly determined by the model’s architecture and form a partition of their input space. That partition –which is involved in all the results spanned from this line of research– has so far only been computed on 2/3-dimensional slices of the DN’s input space or estimated by random sampling. In this paper, we provide the first parallel algorithm that does exact enumeration of the DN’s partition regions. The proposed algorithm enables one to finally assess the closeness of the commonly employed approximations methods, e.g. based on random sampling of the DN input space. One of our key finding is that if one is only interested in regions with “large” volume, then uniform sampling of the space is highly efficient, but that if one is also interested in discovering the “small” regions of the partition, then uniform sampling is exponentially costly with the DN’s input space dimension. On the other hand, our proposed method has complexity scaling linearly with input dimension and the number of regions.

1. INTRODUCTION

Deep Networks (DNs) are compositions of linear and nonlinear operators altogether forming a differentiable functional f_{θ} governed by some trainable parameters θ [1]. Understanding the underlying properties that make DN’s the great function approximators that they are involve many different research directions e.g. the underlying implicit regularization of architectures [2], or the impact of input and feature normalization into the optimization landscape [3]. Most existing results emerge from a few different mathematical formulations. One eponymous example relies on kernels and emerges from pushing the DN’s layers width to infinity. In this case, and under some additional assumptions, a closed-form expression for the DN’s underlying embedding space metric is obtained [4]. With that form, training dynamics and generalization bounds are amenable to theoretical analysis [5]. Another line of research considers the case of deep linear networks i.e. a DN

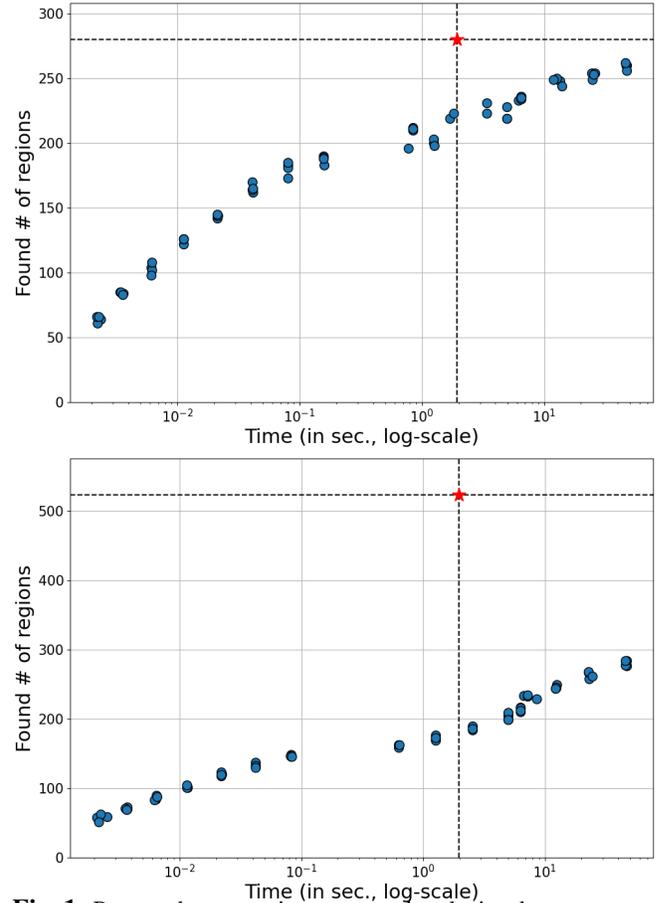


Fig. 1. Proposed exact region enumeration depicted as an **orange star** against sampling-based region discovery of the partition Ω depicted as **blue dots** for a single hidden layer DN with leaky-ReLU, random parameters and width 64 as a function of computation time (**x-axis**) and number of partition regions found (**y-axis**); for a 4-dimensional input space at the **top** and 8-dimensional input space at the **bottom**. *The proposed Algorithm 1 is able to dramatically outperform the sampling-based search that has been used throughout recent studies on CPA DN’s.*

without nonlinearities. In this setting, it is possible to obtain the explicit regularizer that acts upon the DN’s functional and that depends on the specifics of the architecture e.g. depth and width [6]. Another direction, most relevant to our study,

proposes to unravel the Continuous Piecewise Affine (CPA) mapping of standard DNs [7]. In short, one can combine the fact that (i) the nonlinearities present in most current DNs are themselves CPA e.g. (leaky-)ReLU, absolute value, max-pooling, (ii) the interleaved affine mappings preserve the CPA property, and (iii) composition of CPA mappings remain CPA. Thus, the entire input-output DN is itself a CPA. From that observation, it is possible to study the DN’s loss landscape [8], the implicit regularizer of different architectures [9], the explicit probabilistic distributions of CPA Deep Generative Networks [10, 11], the approximation rates [12, 13], or even the conditions for adversarial robustness guarantees [14, 15]. A striking benefit of the CPA viewpoint lies in the fact that it is an exact mathematical description of the DN’s input-output mapping without any approximation nor simplification. This makes the obtained insights and guidelines highly relevant to improve currently deployed state-of-the-art architectures.

Despite this active recent development of CPA-based results around DNs, one key challenge remains open. In fact, because under this view one expresses the DN mapping as a collection of affine mappings—one for each region ω of some partition Ω of their input space—it becomes crucial to compute that partition Ω or at least infer some statistics from it. Current analytical characterizations of Ω are in fact insufficient e.g. existing bounds characterizing the number of regions in Ω are known to be loose and uninformative [16]. As such, practitioners resort to simple approximation strategies, e.g. sampling, to estimate such properties of Ω . Another approach is to only consider 2/3-dimensional slices of the DN’s input space and estimate Ω restricted on that subspace. All in all, nothing is known yet about how accurate are those approximations at conveying the underlying properties of the entire partition Ω that current theoretical results heavily rely on. In particular, [17] uses estimates of the partition’s number of region to perform Neural Architecture Search (NAS), and for which exact computation of the DNN’s partition regions will further improve the NAS; [11] uses estimates of the partition to adapt the distribution of deep generative networks (e.g. variational autoencoders) and for which exact computation of the partition would make their method exact, and not an approximation

In this paper, we propose a principled and provable enumeration method for DNs partitions (Algorithm 1) that we first develop for a layer-wise analysis in Section 2 and then extend to the multilayer case in Section 3. As depicted in Fig. 1, the proposed method becomes exponentially faster than the sampling-based strategy to discover the regions $\omega \in \Omega$ as the input dimensionality increases. Practically, the proposed enumeration method enables for the first time to measure the accuracy of the currently employed approximations. Our method is efficiently implemented with a few lines of codes, leverages parallel computations, and provably enumerates all the regions of the DN’s partition. Lastly, our method has linear asymptotic complexity with respect to the number of regions and with respect to the DN’s input space dimension. This

property is crucial as we will demonstrate that sampling-based enumeration method has complexity growing exponentially with respect to the DN’s input space dimension as a direct consequence of the curse of dimensionality [18, 19]. We hope that our method will serve as the baseline algorithm for any application requiring provable partition region enumeration, or to assess the theoretical findings obtain from the CPA formulation of DNs.

2. ENUMERATION OF SINGLE-LAYER PARTITIONS

We now develop the enumeration algorithm for a single DN layer. Because a DN recursively subdivides the per-layer partition, the single layer case will be enough to iteratively compute the partition of a multilayer DN as shown in the next Section 3.

2.1. Layer Partitions and Hyperplane Arrangements

We denote the single layer of a DN¹ input-output mapping as $f_{\theta} : \mathbb{R}^D \mapsto \mathbb{R}^K$, with θ the parameters of the mapping. Without loss of generality, we consider vectors as inputs since when dealing with images, one can always flatten them into vectors and reparametrize the layer accordingly. The layer mapping takes the form

$$f_{\theta}(\mathbf{x}) = \sigma(\mathbf{h}(\mathbf{x})) \text{ with } \mathbf{h}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (1)$$

where σ is a pointwise activation function, \mathbf{W} is a weight matrix of dimensions $K \times D$, \mathbf{b} is a bias vector of length K , $\mathbf{h}(\mathbf{x})$ denotes the *pre-activation map* and lastly \mathbf{x} is some input from \mathbb{R}^D . The layer parameters are thus $\theta \triangleq \{\mathbf{W}, \mathbf{b}\}$. Although simple, Eq. (1) encompasses most current DNs layers by specifying the correct structural constraints on the matrix \mathbf{W} , e.g. to be circulant for a convolutional layer. The details on the layer mapping will not impact our results. The CPA view of DNs [20, 7] consists in expressing Eq. (1) as

$$f_{\theta}(\mathbf{x}) = \sum_{\omega \in \Omega} (\mathbf{A}_{\omega}\mathbf{x} + \mathbf{b}_{\omega}) 1_{\{\mathbf{z} \in \omega\}}, \quad (2)$$

where Ω is the layer input space partition [21]. Understanding the form of Ω will greatly help the development of the enumeration algorithm in Section 2.2. Given nonlinearities σ such as (leaky-)ReLU or absolute value, it is direct to see that the layer stays linear for a region ω so that all the inputs within it have the same pre-activation signs. That is, a region is entirely and uniquely determined by those sign patterns

$$f_{\theta} \text{ affine on } \omega \iff \text{sign}(\mathbf{h}(\mathbf{x})) = \text{sign}(\mathbf{h}(\mathbf{x}')), \forall (\mathbf{x}, \mathbf{x}') \in \omega^2,$$

where the equality is to be understood elementwise on all of the K entries of the sign vectors. The only exception arises for degenerate weights \mathbf{W} which we do not consider since any arbitrarily small perturbation of such degeneracies remove those

¹without loss of generality we consider the first layer, although the exact same analysis applies to any layer in the DN when looking at the partition of its own input space

edge cases. From the above observation along, it becomes clear that the transition between different regions of Ω must occur when a pre-activation sign for some unit $k \in \{1, \dots, K\}$ changes, and because \mathbf{h} is nothing more but an affine mapping, this sign change for some unit k can only occur when crossing the hyperplane

$$\mathbb{H}_k \triangleq \{\mathbf{x} \in \mathbb{R}^D : \langle \mathbf{W}_{k,\cdot}, \mathbf{x} \rangle + \mathbf{b}_k = 0\}. \quad (3)$$

Leveraging Eq. (3) we obtain that $\partial\Omega$, the boundaries of the layer’s partition, is an hyperplane arrangement as in $\partial\Omega = \bigcup_{k=1}^K \mathbb{H}_k$.

We are now able to leverage this particular structure of the layer’s partition to present an enumeration algorithm that will recursively search for all the regions $\omega \in \Omega$.

2.2. Region Enumeration Algorithm

From the previous understanding that the layer’s partition arises from an hyperplane arrangements involving Eq. (3), we are now able to leverage and adapt existing enumeration methods for such partitions to obtain all the regions $\omega \in \Omega$, form which it will become trivial to consider the multilayer case that we leave for the following Section 3.

Enumerating the regions of the layer f_θ ’s partition can be done efficiently by adapting existing reverse search algorithms [22] optimized for hyperplane arrangements. In fact, a naive approach of enumerating all of the 2^K possible sign patterns $\mathbf{q} \in \{-1, 1\}^K$ and checking if each defines a non-empty region

$$\bigcap_{k=1}^K \{\mathbf{x} \in \mathbb{R}^D : \langle \mathbf{W}_{k,\cdot}, \mathbf{x} \rangle + \mathbf{b}_k \mathbf{q}_k \geq 0\} \stackrel{?}{=} \emptyset,$$

would be largely wasteful. In fact, most of such sign combinations do produce empty regions e.g. if the partition is central i.e. the intersection of all the hyperplane is not empty then the total number of regions grows linearly with K [23] and is thus much smaller than 2^K . Instead, a much more efficient strategy is to only explore feasible sign patterns in a recursive tree-like structure. To do so, the algorithm recursively sub-divides a parent region by the hyperplane of unit k . If that hyperplane does not intersect the current region then we can skip unit k and recurse the sub-division of that same region by unit $k + 1$. On the other hand, if hyperplane k divides the current region, we consider both sides of it and keep the recursion going on both sides. We formally summarize the method in Algorithm 1 and present one illustrative example and comparison against sampling-based region enumeration in Fig. 1. In particular, we provide the efficiency of the sampling solution for various configurations in Table 1.

3. ENUMERATION OF MULTI-LAYERS PARTITIONS

This section demonstrates how the derivation carried out in Section 2 for the single layer setting is sufficient to enumerate the partition of a multilayer DN, thanks to the subdivision

Algorithm 1 Proposed region enumeration method for the single hidden layer case that recursively searches over the feasible sign patterns \mathbf{q} one unit at a time, and only explores the branches that coincide with non-empty region i.e. avoiding the 2^K total number of possible of combinations. The step checking for intersection between an hyperplane and a given polytopal region can be done easily by setting up a linear program with dummy constant objective, the hyperplane as a linear constraint, and the polytopal region as inequality constraint; during the feasibility check the test will fail if the intersection is empty. This algorithm is obtained to provide the results from Fig. 1 and Table 1. The algorithm terminates once all the regions of the partition Ω have been visited.

Require: $\mathbf{W} \in \mathbb{R}^{K \times D}, \mathbf{b} \in \mathbb{R}^K, k \in \{1, \dots, K\}, \mathbf{q} \in \{-1, 0, 1\}^k$

- 1: if $\mathbf{k} \stackrel{?}{=} \mathbf{K} + 1$ then this branch has reached a leaf, the sign pattern \mathbf{q} is feasible and can be accumulated into Ω ’s current estimate
- 2: Check if the hyperplane defined by $(\mathbf{w}_k, \mathbf{b}_k)$ intersects the polytopal region defined by $\bigcap_{j=1}^{k-1} \{\mathbf{x} \in \mathbb{R}^D : \langle \mathbf{w}_j, \mathbf{x} \rangle + \mathbf{b}_j \mathbf{q}_j \geq 0\}$
- 3: if **NO** then unit j is redundant, call the routine again with $[\mathbf{q}_j, 0]$ as \mathbf{q} and $k + 1$ as k
- 4: if **YES** then unit j splits the region into two, call the routine again with $[\mathbf{q}_j, 1]$ and $k + 1$ and $[\mathbf{q}_j, -1]$ and $k + 1$

Ensure: $\mathbf{X}^{(L)} \triangleright$ Evaluate loss and back-propagate as usual

Table 1. Comparison of our exact enumeration method versus sampling-based partition discovery for various single layer configurations with random weights and biases. The sampling-based discovery is run 5 times and we report the average and standard deviation of the number of regions found after sampling. The number of input space sample is obtain so that the computation time of the proposed method is the same as the computation time of the sampling method i.e. for each configuration, both methods have run the exact same amount of time. We observe that for low-dimensional input space, and with the same fixed time-budget, both methods perform similarly and sampling is sufficient to quickly discover all of the layer’s partition.

input dim \width	K=16	K=32	K=64	K=128	K=256	
D=2	enumeration	16	13	71	127	631
	sampling	16 ±0	13 ±0	67±0	127±0	611 ±2
	samp. found	100%	100 %	94 %	100 %	96 %
D=4	enumeration	54	80	1107	4271	95954
	sampling	51 ±0	69 ±1	866±3	3288±18	70635 ±55
	samp. found	94 %	86 %	78 %	77%	73%
D=8	enumeration	24	1242	8396	386566	-
	sampling	18 ±0	543±2	2875±5	136748±251	-
	samp. found	75 %	44 %	34 %	35 %	-

process under which the composition of many layers ultimately form the global DN’s input space partition. We first recall this subdivision step in Section 3.1 and summarize the enumeration algorithm in Section 3.2.

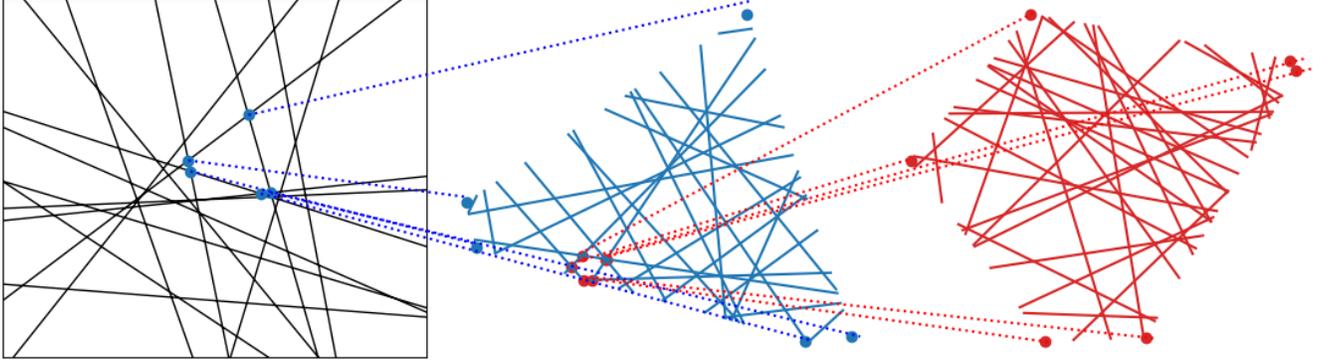


Fig. 2. Depiction of the multilayer case which corresponds to a union of region-constrained hyperplane arrangements and thus which can be studied directly from the proposed hyperplane arrangement region enumeration. The only additional step is to first enforce that the search takes place on the restricted region of interest from the up-to-layer- ℓ input space partition. For example on the **left column** one first obtains the first layer partition depicted in **black**. On each of the enumerated region, a subdivision will be performed by the next layer; pick any region of interest, compose the per-region affine mapping (fixed on that region) with the second layer affine mappings, and repeat the region enumeration algorithm. This discovers the second subdivision done by the second layer, highlighted in **blue** in the **middle column**. This can be repeated to obtain the subdivision of the third layer, here highlighted in **red** in the **right column**.

3.1. Deep Networks are Continuous Piecewise Affine

We specialize the per-layer notations from Section 2 by explicating the layer index ℓ as $f^{(\ell)}$ for the layer mapping, as $\theta^{(\ell)}$ for its parameters, and the entire DN’s input-output mapping is now referred to as $f_{\theta} : \mathbb{R}^D \mapsto \mathbb{R}^K$ with K the output space dimension. The composition of layers take the form

$$f_{\theta} = \left(f_{\theta^{(L)}}^{(L)} \circ \dots \circ f_{\theta^{(1)}}^{(1)} \right), \quad (4)$$

where each layer mapping $f^{(\ell)} : \mathbb{R}^{D^{(\ell)}} \mapsto \mathbb{R}^{D^{(\ell+1)}}$ produces a *feature map*; with $D^{(1)} \triangleq D$ and $D^{(L)} \triangleq K$; with mapping given by Eq. (1), and $h^{(\ell)}$ denoting the *pre-activation map* of layer ℓ . A key result from [20, 7] is the DN mapping is itself defined on a partition as in

$$f_{\theta}(x) = \sum_{\omega \in \Omega} (A_{\omega}x + b_{\omega}) 1_{\{z \in \omega\}},$$

which is known to be recursively built by each layer subdividing the previously built partition of the space [21].

3.2. Enumerating Union of Hyperplane Arrangements

Considering an arbitrarily deep model can be tackled by understanding the recurrent subdivision process of a two hidden layer DN and applying the same principle successively. In this setting, notice that for the (two-layer) DN to be affine within some region ω of the DN’s input space, each layer must stay affine as well. By composition the first layer staying linear does not ensure that the DN stays linear, but the first layer being nonlinear does imply that the entire DN is nonlinear. From that, we see that the first layer’s partition are “coarser” than the entire DN’s partition regions. More precisely, and following the derivation of [21], we obtain that each layer is a recursive subdivision of the previously build

partition when in our case we need to search for each region ω of the first layer’s partition the regions within it where the second layer stays linear. As a result, the proposed single hidden layer enumeration method from Section 2 can be applied recursively as follows. First, compute the first layer partition enumeration. Then, for each enumerated region with corresponding sign pattern q , define a new single layer model with $h(x) \triangleq \sigma(W^{(2)} \text{diag}(q)W^{(1)}x + W^{(2)}(q \odot b^{(1)}) + b^{(2)})$ and within ω apply the single layer enumeration; repeating the process for all regions –and corresponding sign patterns q of the previously found first layer partition. This enumerates the partition of $(f^{(2)} \circ f^{(1)})$, and the same process can be repeated as many times as there are layers in the DN; as illustrated in Fig. 2.

4. CONCLUSION AND FUTURE WORK

In this paper, we provided the first exact enumeration method for Deep Networks partitions that relies on the existing highly efficient enumeration method of hyperplane arrangements. In fact, both the shallow and deep architectures produce partitions that correspond to hyperplane arrangements or union of restricted hyperplane arrangements. A crucial finding that was enabled by the proposed method is that sampling-based region enumeration, which is the only strategy used in current research studies dealing with DNs and affine splines, is in fact relatively poor at finding the regions of the DN’s partition. In particular, when using such sampling to estimating some sensitive statistics e.g. the volume of the smallest region, sampling is biased and should be avoided in favor of an exact enumeration method.

5. REFERENCES

- [1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [2] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro, “In search of the real inductive bias: On the role of implicit regularization in deep learning,” *arXiv preprint arXiv:1412.6614*, 2014.
- [3] Yann Le Cun, Ido Kanter, and Sara A Solla, “Eigenvalues of covariance matrices: Application to neural-network learning,” *Physical Review Letters*, vol. 66, no. 18, pp. 2396, 1991.
- [4] Arthur Jacot, Franck Gabriel, and Clément Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” *arXiv preprint arXiv:1806.07572*, 2018.
- [5] Kaixuan Huang, Yuqing Wang, Molei Tao, and Tuo Zhao, “Why do deep residual networks generalize better than deep feedforward networks?—a neural tangent kernel perspective,” *Advances in neural information processing systems*, vol. 33, pp. 2698–2709, 2020.
- [6] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro, “The implicit bias of gradient descent on separable data,” *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2822–2878, 2018.
- [7] Randall Balestriero and Richard Baraniuk, “A spline theory of deep learning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 374–383.
- [8] Rudolf H Riedi, Randall Balestriero, and Richard G Baraniuk, “Singular value perturbation and deep network optimization,” *arXiv preprint arXiv:2203.03099*, 2022.
- [9] Randall Balestriero and Richard G Baraniuk, “From hard to soft: Understanding deep network nonlinearities via vector quantization and statistical inference,” *arXiv preprint arXiv:1810.09274*, 2018.
- [10] Randall Balestriero, Sébastien Paris, and Richard Baraniuk, “Analytical probability distributions and exact expectation-maximization for deep generative networks,” *Advances in neural information processing systems*, vol. 33, pp. 14938–14949, 2020.
- [11] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk, “Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10641–10650.
- [12] Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova, “Nonlinear approximation and (deep) relu networks,” *Constructive Approximation*, vol. 55, no. 1, pp. 127–172, 2022.
- [13] Randall Balestriero and Richard G Baraniuk, “Batch normalization explained,” *arXiv preprint arXiv:2209.14778*, 2022.
- [14] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon, “Towards fast computation of certified robustness for relu networks,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 5276–5285.
- [15] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang, “Semidefinite relaxations for certifying robustness to adversarial examples,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [16] Herbert Edelsbrunner, *Algorithms in combinatorial geometry*, vol. 10, Springer Science & Business Media, 1987.
- [17] 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.
- [18] Richard E Bellman and Stuart E Dreyfus, *Applied dynamic programming*, vol. 2050, Princeton university press, 2015.
- [19] Mario Köppen, “The curse of dimensionality,” in *5th online world conference on soft computing in industrial applications (WSC5)*, 2000, vol. 1, pp. 4–8.
- [20] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio, “On the number of linear regions of deep neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.
- [21] Randall Balestriero, Romain Cosentino, Behnaam Aazhang, and Richard Baraniuk, “The geometry of deep networks: Power diagram subdivision,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [22] David Avis and Komei Fukuda, “Reverse search for enumeration,” *Discrete applied mathematics*, vol. 65, no. 1-3, pp. 21–46, 1996.
- [23] Richard P Stanley et al., “An introduction to hyperplane arrangements,” *Geometric combinatorics*, vol. 13, no. 389-496, pp. 24, 2004.