Proceedings Track

CantorNet: A Sandbox for Testing Topological and Geometrical Measures

Editors: List of editors' names

Abstract

Many natural phenomena are characterized by self-similarity, for example the symmetry of human faces, or a repetitive motif of a song. Studying of such symmetries will allow us to gain deeper insights into the underlying mechanisms of complex systems. Recognizing the importance of understanding these patterns, we propose a geometrically inspired framework to study such phenomena in artificial neural networks. To this end, we introduce *CantorNet*, inspired by the triadic construction of the Cantor set, which was introduced by Georg Cantor in the 19^{th} century. In mathematics, the Cantor set is a set of points lying on a single line that is self-similar and has a counter intuitive property of being an uncountably infinite null set. Similarly, we introduce CantorNet as a sandbox for studying self-similarity by means of novel topological and geometrical complexity measures. CantorNet constitutes a family of ReLU neural networks that spans the whole spectrum of possible Kolmogorov complexities, including the two opposite descriptions (linear and exponential as measured by the description length). CantorNet's decision boundaries can be arbitrarily ragged, yet are analytically known. Besides serving as a testing ground for complexity measures, our work may serve to illustrate potential pitfalls in geometry-ignorant data augmentation techniques and adversarial attacks.

Keywords: topological and geometrical measures, ReLU neural networks, synthetic examples

1. Introduction

Neural networks perform extremely well in various domains, for example computer vision (Krizhevsky et al., 2012) or speech recognition (Maas et al., 2013). Yet, this performance is not sufficiently understood from the mathematical perspective, and current advancements are not backed up by a formal mathematical analysis. We start by identifying a lack of *tractable examples* that allow us to study neural networks through the lens of self-similarity of objects they describe. This difficulty arises from the inherent statistical nature of these networks and the significant computational effort required to accurately determine the underlying geometry of the decision manifold. We note that the use of constructed examples helps to illustrate certain characteristic effects, such as the concentration of measure effects in high dimensions to explain the vulnerability against adversarial examples (Gilmer et al., 2018). Such examples are typically designed to underscore either the capabilities or limitations of neural architectures in exhibiting certain phenomena. However, there exists a risk of oversimplification that might lead to an underappreciation of the complexities and challenges of handling real-world, high-dimensional, and noisy data. Despite these limitations, toy examples are valuable as they can be constructed to emphasise some properties which remain elusive at a larger scale. Further examples include the XOR (Minsky and Papert, 1969) or CartPole problem (Sutton and Barto, 2018) which, despite their simplicity,

have provided a controllable evaluation framework in their respective fields. Our analysis is further motivated by the fact that many natural phenomena feature some self-similarity as understood by symmetry, e.g., images (Wang et al., 2020), audio tracks (Foote, 1999) or videos (Alemán-Flores and Álvarez León, 2004).

Our work is closely related to the concepts such as Cantor set, fractals and the Kolmogorov complexity. In the following, we provide a brief background about these concepts.

Cantor set. Cantor set, introduced in mathematics by Georg Cantor (Cantor, 1879), is a set of points lying on a single line segment that has a number of counter intuitive properties such as being self-similar, and uncountably infinite, yet of Lebesgue measure 0. It is obtained by starting with a line segment, partitioned into three equal sub-segments and recursively deleting the middle one, repeating the process an infinite number of times. It is used to illustrate how complex structures can arise from simple recursion rules (Mandelbrot, 1983). CantorNet is inspired by the construction procedure of the Cantor set, inhering its fractal properties.

Fractals. In nature, some complex shapes can be described in a compact way. Fractals are a good example as they are self-similar geometric shapes whose intricate structure can be compactly encoded by a recursive formula (e.g., von Koch (1904)). A number of measures have been developed to quantify the complexity of the fractals, e.g., Mandelbrot (1983, 1995); Zmeskal et al. (2013). On one hand, fractals are self-repetitive structures, what results in a compact description. On the other hand, some fractals can be represented as unions of polyhedral bodies, a less compact description. The construction of the CantorNet is inspired by the triadic representation of the Cantor set, and its opposite representations (complexity-wise) are based on recursed generating function and union of polyhedral bodies.

The Kolmogorov Complexity. The Kolmogorov complexity (Kolmogorov, 1965) quantifies the information conveyed by one object about another and can be applied to models by evaluating the shortest program that can reproduce a given output, as proposed as early as Solomonoff (1964). In the context of neural networks, Schmidhuber argues that prioritizing solutions with low Kolmogorov complexity enhances generalization. Although computing the exact Kolmogorov complexity of real-world architectures is unfeasible, approximations to the minimal description length (Grünwald et al., 2005) can be made by analyzing the number of layers and neurons, under the condition that the neural networks represent *exactly* the same decision boundaries, which is also the case in our CantorNet analysis.

In summary, in this work we propose *CantorNet*, an arbitrarily ragged decision surface, a natural candidate for testing various geometrical and topological measures. Furthermore, we study the Kolmogorov complexity of its two equivalent constructions with ReLU nets. The rest of the paper is organized as follows. Section 2 recalls some basic facts and fixes notation for the rest of the paper, and in Section 3, we describe different CantorNet constructions and representations. Finally, in Section 5, we provide concluding remarks and possible future directions for our work.

2. Preliminaries

We define a *ReLU neural network* $\mathcal{N} : \mathcal{X} \to \mathcal{Y}$ with the total number of N neurons as an alternating composition of the ReLU function $\sigma(x) := \max(x, 0)$ applied element-wise

on the input x, and affine functions with weights W_k and biases b_k at layer k. An input $x \in \mathcal{X}$ propagated through $\mathcal{N}$ generates non-negative activation values on each neuron. A *binarization* is a mapping $\pi : \mathbb{R}^N \to \{0,1\}^N$ applied to a vector (here a concatenation of all hidden layer) $v = (v_1, \ldots, v_N) \in \mathbb{R}^N$ resulting in a binary vector $\{0,1\}^N$ by clipping strictly positive entries of v to 1, and non-positive entries to 0, that is $\pi(v_i) = 1$ if $v_i > 0$, and $\pi(v_i) = 0$ otherwise. An *activation pattern* is the concatenation of all neurons after their binarization for a given input $\mathbf{x}$, and represents an element in a binary hypercube $\mathcal{H}_N := \{0,1\}^N$ where the dimensionality is equal to the number of hidden neurons in network $\mathcal{N}$. A *linear region* is an element of a disjoint collection of subsets covering the input domain where the network behaves as an affine function (Montúfar et al., 2014). There is an one-to-one correspondence between an activation pattern and a linear region (Shepeleva et al., 2020).

3. CantorNet

In this section, we define CantorNet as a ReLU neural network through repeating application of weight matrices, similar to the fractal constructions. We then introduce an equivalent description through unionizing polyhedral bodies, which is less concise. We start the construction with two reshaped ReLU functions (Fig. 1, left), and modify them to obtain a connected decision manifold with Betti numbers $b_i = 0$ for $i \in \{0,1,2\}$, used to characterizes the topological complexity, providing measures of connectivity, loops, and voids within the decision boundaries (Bianchini and Scarselli, 2014). We consider the function

$$A : [0,1] \to [0,1] : x \mapsto \max\{-3x + 1, 0, 3x - 2\}, \tag{1}$$

as the *generating function* and recursively nest it as

$$A^{(k+1)}(x) := A(A^{(k)}(x)),\ A^{(1)}(x) := A(x). \tag{2}$$

Based on the generating function, we can define the decision manifold R_k as:

$$R_k := \{(x_1, x_2) \in [0,1]^2 : x_2 \leq (A^{(k)}(x_1) + 1)/2\}. \tag{3}$$

For a better understanding of the decision manifolds, we have visualized R_1, R_2, and R_3 in Fig. 1. The *CantorNet* is given by the nested function defined by Eq. (2), which can be mapped to a ReLU neural network representation. We define the CantorNet family as follows.

Definition 1 *A ReLU net $\mathcal{N}$ belongs to the CantorNet family iff $\mathcal{N}^{(-1)}(0) = R_k$.*

We further name the regions "below" and "above" R_k as the inset and the outset of CantorNet, respectively, as formalized in the Def. 2.

Definition 2 *We say that the manifold R_k given by Eq. (3) represents the inset of CantorNet, while its complement on the unit square represents the outset (grey and white areas in Fig. 2, respectively).*

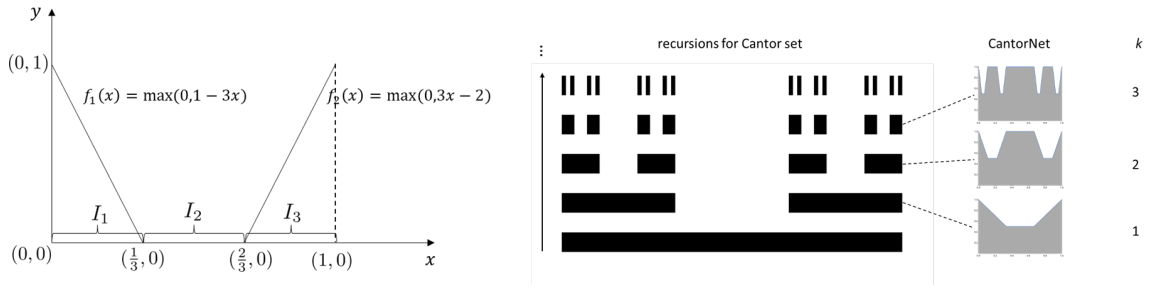

Figure 1: Left: The first iteration of the 1-1 correspondence between the ReLU net $\widetilde{\mathcal{N}}_A$, induced by the generating function A, and the triadic number expansion shows the intervals I_1, I_2, I_3 correspond to the digits $\{0, 1, 2\}$, respectively. Right: CantorNet is inspired by the construction of the Cantor set (Cantor, 1879).

3.1. Recursion-Based Construction

The decision surface of R_k (Eq. (3)) equals to the 0-preimage of a ReLU net $\mathcal{N}_A^{(k)} : [0, 1]^2 \to \mathbb{R}$ with weights and biases defined as

$$W_1 = \begin{pmatrix} -3 & 0 \\ 3 & 0 \\ 0 & 1 \end{pmatrix}, b_1 = \begin{pmatrix} 1 \\ -2 \\ 0 \end{pmatrix}, W_2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

and the final layer $W_L = \begin{pmatrix} -\frac{1}{2} & 1 \end{pmatrix}, b_L = \begin{pmatrix} -\frac{1}{2} \end{pmatrix}$. For recursion depth k, we define $\mathcal{N}_A^{(k)}$ as

$$\mathcal{N}_A^{(k)}(\mathbf{x}) := W_L \circ \sigma \circ g^{(k)}(\mathbf{x}) + b_L, \tag{5}$$

where $g^{(k+1)}(\mathbf{x}) := g^{(1)}(g^{(k)}(\mathbf{x}))$, σ is the ReLU function, and

$$g^{(1)}(\mathbf{x}) := \sigma \circ W_2 \circ \sigma \circ (W_1 \mathbf{x}^T + b_1). \tag{6}$$

We use $\circ$ to denote the standard composition of functions. Fig. 2 shows the linear regions resulting from the construction described in the Eq. (5) for recursion level $k = 1$, as well as the linear regions with the corresponding activation patterns from non-redundant neurons (we skip neurons which do not change their state).

3.2. Triadic Expansion

In this section, we show that there exists an isomorphism between the triadic expansion, as described in Appendix A in Alg. 1, and the activation pattern $\pi_{\mathcal{N}_A}$ under $\mathcal{N}_A^{(k)}$. In the Triadic Expansion, we partition the interval $[0, 1]$ into three intervals, $I_1 = [0, \frac{1}{3}], I_2 = (\frac{1}{3}, \frac{2}{3}), I_3 = [\frac{2}{3}, 1]$ (see Fig. 1, left). Any $x \in I_1 \cup I_3$ can be described in a triadic system with an arbitrary precision l as $x = \sum_{i=1}^{l} \frac{a_i}{3^i}$, where $a_i \in \{0, 2\}$. Recall that the tessellation of the recursion-based model (Fig. 2) is obtained by partitioning the rectangular domain $(I_1 \cup I_3) \times [0, 1]$ into increasingly fine rectangles through recursive applications of $\mathbf{x} \mapsto g^{(k)}(\mathbf{x})$. We identify created linear regions by their activation patterns $\pi_{\mathcal{N}_A}$. Equivalently, we can

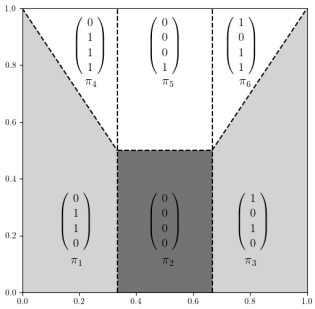

Figure 2: Activation patterns π_i induced by Eq. (5). We skip neurons with unchanged values.

represent any $x \in I_1 \cup I_3$ using Alg. 1, obtaining activation patterns as a sequence of "0"s, and "1"s. Each of these descriptions is unique, therefore there exists an isomorphic relationship between the encoding described in Alg. 1, and the recursion-based description.

Lemma 3 (Computational Complexity of Activation Patterns of $\mathcal{N}_A^{(k)}$) *Given an input $\mathbf{x} = (x_1, x_2) \in [0,1]^2$ and the recursion level k, its corresponding activation pattern $\pi(\mathbf{x})$ under the recursion-based representation $\mathcal{N}_A^{(k)}$ can be computed in $O(k)$ operations.*

Proof The complexity (as measured by the description length (Grünwald et al., 2005)) of the decision manifold given by Eq. (3) is equal to the complexity of its partition into the linear regions defined in Sec. 2. To determine the minimal complexity of the partition it is necessary to solve the following decision problem for x_1. Consider the partition into linear regions and its projection along the y-axis onto the $[0,1] \times \{0\}$. Note that the resulting partition of $[0,1]$ is the same we obtain by constructing the Cantor set of level k, which corresponds to the triadic number expansion up to the k^{th} digit. The minimal complexity of solving this decision problem is therefore $O(k)$. ■

The proof of Lemma 3 indicates the 1-1 correspondence between the triadic number expansions up to the k^{th} digit and the activation pattern $\pi(x_1)$ of $x_1 \in [0,1]$ under $\widetilde{\mathcal{N}}_A^{(k)}$, where $\widetilde{\mathcal{N}}_A^{(k)}$ represents the 1-dim ReLU network up to the recursion level k, analogous to the construction given by Eq. (5). Observe that the outset (as in Def. 2) intervals I_1, I_2, I_3 (as in Fig. 1, left) can be described with activation patterns $\pi_{\mathcal{N}_A}(x) = [10111]$, for any $x \in I_1$, $\pi_{\mathcal{N}_A}(x) = [00101]$, for any $x \in I_2$ or $\pi_{\mathcal{N}_A}(x) = [01111]$ for any $x \in I_3$ in the recursion-based representation (here we do not remove neurons with constant values). Indeed, to obtain π_x for $x \in I_1$ take any point $(x, y) \in I_1 \times \{y \in [0,1] : y < f_1(x)\}$ (f_1 and f_2 as in Fig. 1, left). After applying $\mathbf{x} \mapsto g^{(1)}(\mathbf{x})$, binarizing every neuron's value and concatenating them into a vector, we obtain a 5-dim vector (because $W_1 \in \mathbb{R}^{\underline{3}\times 2}$, $W_2 \in \mathbb{R}^{\underline{2}\times 3}$, and we omit W_L, b_L). In an analogous manner, we can obtain π_{I_2} and π_{I_3}. Next, observe that each of the intervals I_i can be further partitioned into I_{i1}, I_{i2}, I_{i3}, respectively for the left, center, and right segments. To describe these new segments, we increase the recursion level to

$k = 2$. It turns out that $I_{11} = [10111; 10111]$, a repetition of the pattern π_{I_1}, and so forth for the remaining segments. This construction is iterated k times, providing the sequence of subintervals

$$(I_{i_t})_{t=1}^{k}. \tag{7}$$

3.3. Alternative Representation of CantorNet

Observe that the pre-image of zero under a ReLU function (including shifting and scaling) is a closed set in $[0, 1]^2$. Since we consider a decision manifold $\mathcal{M}$ as a closed subset, which we referred to as inset in Def. 2 in Section 3, we use the closed pre-image of zero under the ReLU network $\mathcal{N}$ to model decision manifolds given by Eq. (3). This means that the statement "$\mathbf{x} \in \mathcal{M}$" is true if $\mathcal{N}(\mathbf{x}) = 0$, and the statement "$\mathbf{x} \in \mathcal{M}$" is false if $\mathcal{N}(\mathbf{x}) > 0$. This way the min operation refers to the union of sets, i.e., logical disjunction "OR", while the max operation refers to the intersection of two sets, i.e., logical conjunction "AND". Note that the laws of Boolean logics also translate to this interpretation (Klir and Yuan, 1995). Further, note that any (non-convex) polytope is a geometric body that can be represented as the union of intersections of convex polytopes. A convex polytope can also be represented as intersection of half-spaces, like the pre-image of zero under the ReLU function, i.e., a single-layered ReLU network. Thus, a decision manifold $\mathcal{M}$ given by a (non-convex) polytope can be represented by the minimum of maxima of single layered ReLU networks. Since the minimum operation can also be represented in terms of the max function, we obtain a ReLU representation for which it is justified to call it a *disjunctive normal form* (DNF), as outlined by Moser et al.. For the simplest case, consider the function $h_1(x, y)$ (Fig. 3, left) that splits the unit square $[0, 1]^2$ into parts where it takes positive and negative values, denoted with (1) and (0), respectively. Observe that $(x, y) \in (\pi \circ h_1)^{(-1)}(0)$ if and only if $\max\{h_1(x, y), 0\} = 0$ (where π is the binarization operator described in Sec. 2). Similarly, $(x, y) \in (\pi \circ h_1)^{(-1)}(1)$ if and only if $\max\{h_1(x, y), 0\} = h_1(x, y)$. In the case of two hyperplanes (Fig. 3, right), the polytope denoted with $(0, 0)$ can be represented as $(x, y) \in (\pi \circ h_1)^{(-1)}(0) \cap (\pi \circ h_2)^{(-1)}(0)$ if and only if $\max\{h_1(x, y), h_2(x, y), 0\} = 0$, and similarly for the remaining polytopes.

Fig. 4 represents the partition of the decision manifold given by Eq. (3) into convex polytopes for $k = 2$ and $k = 3$, respectively. We utilize the minimum function to form their union, obtaining a ReLU network $\mathcal{N}_B^{(k)}$ that yields the same decision manifold as the recursion-based $\mathcal{N}_A^{(k)}$.

Proposition 4 *At the recursion level k, the decision boundary given by Eq. (3) can be constructed as a disjunctive normal form*

$$\{x, y \in [0, 1]^2 : \min(h_1(x, y), h_2(x, y), h_{r(k)}(x, y), D_1, \ldots, D_{\lfloor r(k)/4 \rfloor + 1}, 0) = 0\}, \tag{8}$$

where $h_i : \mathbb{R}^2 \to \mathbb{R}$ are affine functions indexed with $i = 1, \ldots, r(k)$. The labeling function $r(k) : \mathbb{N} \to \mathbb{N}$ is given as $r(k) = 2^{k+1} - 1$ for $k \in \mathbb{N}$ (see Fig. 4), and $D : \mathbb{R}^2 \to \mathbb{N}$ denotes a "dent" given by

$$D_l := \max(h_{4l-1}, h_{4l}, h_{4l+1}). \tag{9}$$

To provide a better overview for the reader, in Table 1 we list the constructions for the different recursion levels. We sketch an inductive proof of the Proposition 4 in Appendix B. Further, note that the min function can be expressed as a ReLU network (Appendix C).

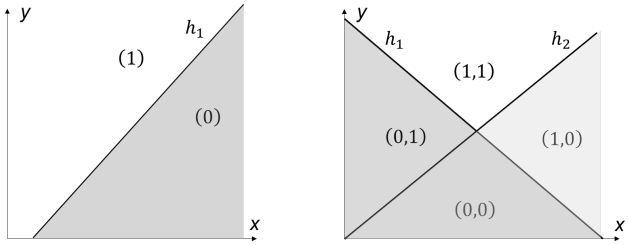

Figure 3: The greyed regions can be represented as a 0-preimage of $\pi \circ h_1$ (left), and the union of the 0-preimages of $\pi \circ h_1$ and $\pi \circ h_2$ (right).

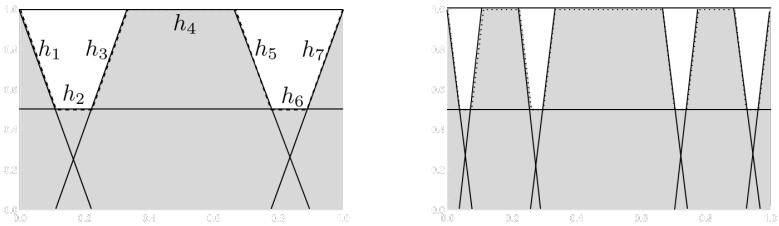

Figure 4: Decision surfaces ($k = 2, 3$) of (3) with labeled functions h_i.

4. Complexity of Neural Representations

The complexity of an object can be measured in many ways, for example its description's length (Kolmogorov, 1965). Preference for a more concise description can be argued from multiple angles, for example using the principle of the Occam's razor or lower Kolmogorov complexity. The latter is typically non-computable, necessitating reliance on its approximations. However, in case of models with consistent decision boundaries (e.g., neural networks), their size, both in terms of the number of layers and the number of neurons, can be used as approximation for complexity (Grünwald et al., 2005).

Lemma 5 *At the recursion level k, the number of neurons of the recursion-based representation is $O(k)$, while for the disjunctive normal form representation it is $O(2^k)$.*

Proof For the recursion-based representation the result is straightforward.

The DNF construction relies on the application of $\mathbf{A}$ and $\mathbf{S}$ (see App. B). For recursion level k, we have (recall (8)) $3 + 3(\lfloor r(k)/4 \rfloor + 1) = 3\lfloor r(k)/4 \rfloor + 6 =: z(k)$ rows of $\mathbf{A}$, equal to the number of neurons. By applying $\mathbf{A}$ and $\mathbf{S}$ at least $\lceil \log_2 z(k) \rceil$ times (Arora et al., 2018), we arrive at

$$\left(\frac{3}{4}\left(2^{k+1} - 1\right) + 6\right) \sum_{i=0}^{\lceil \log_2 z(k) \rceil} \frac{1}{2^i} \leq \frac{3}{2}\left(2^{k+1} - 1\right) = O(2^k).$$

■

It requires an algorithm of Kolmogorov complexity of order k to enumerate all numbers in $[0, 1]$ with triadic number expansion up to k digits. Since the recursion-based ReLU net $\mathcal{N}_A$

Table 1: Min/max shape description for k recursions. In each row $x, y \in [0,1]^2$.

recursion	formula of decision manifold R_k
1	$\{x, y : \min(h_1, h_2, h_3) = 0\}$
2	$\{x, y : \min(\underbrace{h_1, h_2, h_7}, \underbrace{\max(h_3, h_4, h_5)}) = 0\}$
...	...
k	$\{x, y : \min(\underbrace{h_1, h_2, h_{r(k)}}_{\text{external half-spaces}}, \underbrace{D_1, \ldots, D_{\lfloor r(k)/4 \rfloor + 1}}_{\text{“dents” (9)}}) = 0\}$

is constructed by a repetitive application of two layers with constant number of neurons (namely five) and the same weights, its Kolmogorov complexity after k iterations is of order k. As a recursion step given by Eq. (5) and Eq. (6) is equivalent to a recursion in the triadic number expansion (Alg. 1), which is of the minimal order of Kolmogorov complexity, there cannot exist an equivalent ReLU network of strictly lower order of Kolmogorov complexity. This means that $\mathcal{N}_A$ is of minimal description length in terms of order of the number of neurons $N = N(k)$, thus

$$O(N(k)) = O(k). \tag{10}$$

Theorem 6 *The recursion-based ReLU representation given by the Eq. (4) is of minimal complexity order in terms of the number of neurons (in the sense of Eq. (10)).*

This way, we obtain an example of a ReLU network of proven minimal description length in terms of its number of neurons, a property hardly provable by statistical means. Observe that the above does not hold for singular numbers from $[0, 1]$: if we consider $x = \frac{1}{6} = 0...0...02...20...02...2...0...0$, with $n = k^2$ digits after the ternary point arranged in k alternating blocks of zeros and twos, then it has Kolmogorov complexity $O(k) = O(\sqrt{n})$. Note that both representations have the same order of the number of layers.

Lemma 7 *At the recursion level k both described representations of CantorNet have $O(k)$ layers.*

Note that both constructions are equivalent as understood by the equality of their preimages. Though simple by construction, the family of CantorNet ReLU networks is rich in terms of representation variants, ranging from a minimal (linear in k) complex solution to an exponentially complex one. An intermediate example would be starting with the recursion based representation for a number of layers, and then concatenating corresponding disjunctive normal form representation. In App. D we discuss the ratio of active neurons for the two representations, contrasting them further.

5. Conclusions and Discussion

In this paper we have proposed CantorNet, a family of ReLU neural networks inspired by fractal geometry that can be tuned arbitrarily close to a fractal. The resulting geometry

of CantorNet's decision manifold, the induced tessellation and activation patterns can be derived in two ways. This makes it a natural candidate for studying concepts related to the activation space. Note that although CantorNet is a hand designed example, is not an abstract invention - real world data, such as images, music, videos also display fractal nature, as understood by self-similarity. We believe that our work, although seemingly remote from the current mainstream of the machine learning research, will provide the community with a set of examples to study ReLU neural networks as mappings between the Euclidean input space and the space of activation patterns, currently under investigation.

References

Miguel Alemán-Flores and Luis Álvarez León. Video segmentation through multiscale texture analysis. In Aurélio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition, ICIAR 2004, Lecture Notes in Computer Science*, volume 3212, pages 339–346. Springer, Berlin, Heidelberg, 2004. doi: 10.1007/978-3-540-30126-4_42. URL https://doi.org/10.1007/978-3-540-30126-4_42.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding Deep Neural Networks with Rectified Linear Units. *ICLR*, 2018.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. NN Learn. Syst.*, 25 (8):1553–1565, 2014.

Georg Cantor. Ueber unendliche, lineare punktmannichfaltigkeiten. *Mathematische Annalen*, 15(1):1–7, 1879.

Jonathan Foote. Visualizing music and audio using self-similarity. In *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, pages 77–80, 1999. doi: 10.1145/319463.319472.

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

Peter D. Grünwald, Jay Injae Myung, and Mark A. Pitt, editors. *Advances in Minimum Description Length: Theory and Applications*. Neural Information Processing. MIT Press, Cambridge, MA, 2005. ISBN 9780262072625.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *ArXiv*, abs/1607.03250, 2016.

George J. Klir and Bo Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, Upper Saddle River, NJ, USA, 1995. ISBN 9780131011717.

Andrey N Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. Macmillan, 1983. ISBN 978-0-7167-1186-5.

Benoit B. Mandelbrot. Measures of fractal lacunarity: Minkowski content and alternatives. In *Fractal Geometry and Stochastics*, Progress in Probability, pages 15–42, Basel, 1995. Birkhäuser Basel.

Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

Guido F Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NeurIPS*, volume 27, 2014.

Bernhard A. Moser, Michal Lewandowski, Somayeh Kargaran, Werner Zellinger, Battista Biggio, and Christoph Koutschan. Tessellation-filtering relu neural networks. *IJCAI*, 2022.

Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997.

Natalia Shepeleva, Werner Zellinger, Michal Lewandowski, and Bernhard Moser. Relu code space: A basis for rating network quality besides accuracy. *ICLR, NAS workshop*, 2020.

Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, 1964.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Helge von Koch. Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire. *Arkiv för matematik, astronomi och fysik*, 1:681–704, 1904.

Xiang Wang, Kai Wang, and Shiguo Lian. A survey on face data augmentation for the training of deep neural networks. *Neural Computing and Applications*, 32(19):15503–15531, Oct 2020. ISSN 1433-3058. doi: 10.1007/s00521-020-04748-3.

Oldrich Zmeskal, Petr Dzik, and Michal Vesely. Entropy of fractal systems. *Computers & Mathematics with Applications*, 66(2):135–146, 2013. ISSN 0898-1221.

Appendix A. Activation Code by Triadic Expansion

Algorithm 1: Activation Code of $\widetilde{\mathcal{N}}_A^{(k)}$ by Triadic Expansion

Input: $x_1 \in [0,1]$, recursion level k
Output: The activation code $\pi := \pi_{\widetilde{\mathcal{N}}_A}(x_1)$
Define $f_1(x) := 1 - 3x, f_2(x) := 3x - 2$;
Define intervals: $I_1 := [0, \frac{1}{3}], I_2 := (\frac{1}{3}, \frac{2}{3}), I_3 := [\frac{2}{3}, 1]$;
Termination $\leftarrow$ **False**;
$\pi = []$ `// A pattern holder`;
$j \leftarrow 0$ `// Iteration counter`;
while ***not*** *Termination* **do**
  $j \leftarrow j + 1$;
  if $j = k$ **then**
    Termination $\leftarrow$ **True**;
  end
  Update interval $J_{i_j} := I_{i_1 \ldots i_j}$, $i_j \in \{1, 2, 3\}$ `// see` (7);
  if $x_1 \in J_1$ **then**
    $x_1 \leftarrow f_1(x_1)$;
    π.append(0);
  else
    if $x_1 \in J_2$ **then**
      Termination $\leftarrow$ **True** `// Exit the loop if` x_1 `is in` I_2;
    else
      $x_1 \leftarrow f_2(x_1)$;
      π.append(1);
    end
  end
end

Appendix B. Min/max Shape Description of CantorNet

Proof We sketch the inductive proof of Proposition 4. For the base case $k = 1$, the decision manifold R_1 is composed of the union of three half-spaces, expressed as $\{x, y \in [0,1]^2 : \min(h_1, h_2, h_3) = 0\}$. For $k = 2$ (shown on the left in Fig. 4), we see a dent in the middle of the figure. Points lying above this dent satisfy the condition $\{x, y \in [0,1]^2 : \max(h_3, h_4, h_5) > 0\}$. To reconstruct the complete decision manifold R_2, we unionize the outer half-spaces h_1, h_2, h_7 with the aforementioned dent. This results in the following set:

$$\{x, y \in [0,1]^2 : \min(h_1, h_2, h_7, \max(h_3, h_4, h_5)) = 0\}.$$

For the inductive step, let's consider an arbitrary recursion depth k. Suppose that the decision boundaries R_k consist of points $x, y \in [0,1]$ such that

$$\min\left(h_1, h_2, h_{r(k)}, D_1, \ldots, D_{\lfloor r(k)/4 \rfloor + 1}\right) = 0, \tag{11}$$

where D_l is as follows

$$D_l := \max(h_{4l-1}, h_{4l}, h_{4l+1}) \tag{12}$$

describes dents. Observing the pattern established by Eq. 1, it becomes clear that incrementing the recursion depth from k to $k+1$ doubles the number of nested maximum functions. In other words, twice the previous number of "dents" emerge in our structure. This observation aligns with Eq. 11. This and the construction of half-spaces h_i assure that all the half-spaces agree with the decision manifold R_{k+1}. ■

Appendix C. Min as a ReLU Net

Theorem 8 *The minimum function* $\min : \mathbb{R}^d \to \mathbb{R}$ *can be expressed as a ReLU neural network with weights* $\{0, \pm 1\}$.

Proof We first show that in the base cases of even and odd number of variables, $d = 2$ and $d = 3$ respectively, we can recover the minimum element by a hand-designed ReLU neural architecture. Recall that $\sigma(x) := \max(x, 0)$, applied element-wise. For $d = 2$ and elements $x_1, x_2 \in \mathbb{R}$ it holds that

$$\begin{aligned} \min(x_1, x_2) &= x_2 + \min(x_1 - x_2, 0) \\ &= x_2 - \max(x_2 - x_1, 0) \\ &= \sigma(x_2) - \sigma(-x_2) - \sigma(-x_1 + x_2), \end{aligned} \tag{13}$$

which can be represented as a ReLU net

$$\min(x_1, x_2) = \underbrace{\begin{pmatrix} 1 & -1 & -1 \end{pmatrix}}_{\mathbf{S}} \sigma \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & -1 \\ -1 & 1 \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

For $d = 3$, $\min(x_1, x_2, x_3) = \min(\min(x_1, x_2), x_3)$, which can be expressed by a ReLU neural network as follows:

$$\mathbf{S}\sigma\mathbf{A} \begin{pmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \sigma \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

where we first recover the $\min(x_1, x_2)$ and leave x_3 unchanged, resulting in $(\min(x_1, x_2), x_3)$, and then we recover the minimum element using the base case for $d = 2$.

Inductive step. We start with the inductive step for an even number of elements. Let $\mathbf{x} \in \mathbb{R}^d$, $d = 2l$ for $l \in \mathbb{N}$, and suppose that with $\mathbf{S}'\sigma(\mathbf{A}'\mathbf{x}), \mathbf{S}' \in \{0, \pm 1\}^{d \times 3d}, \mathbf{A}' \in \{0, \pm 1\}^{3d \times 2d}$

$$\mathbf{S}' := \begin{pmatrix} \mathbf{S} & \mathbf{0}_{1\times 3} & \dots & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{S} & \mathbf{0}_{1\times 3} & \dots \\ \dots & \dots & \dots & \dots \\ \mathbf{0}_{1\times 3} & \dots & \mathbf{0}_{1\times 3} & \mathbf{S} \end{pmatrix},$$

and

$$\mathbf{A}' := \begin{pmatrix} \mathbf{A} & \mathbf{0}_{3\times 2} & \dots & \mathbf{0}_{3\times 2} \\ \mathbf{0}_{3\times 2} & \mathbf{A} & \mathbf{0}_{3\times 2} & \dots \\ \dots & \dots & \dots & \dots \\ \mathbf{0}_{3\times 2} & \dots & \mathbf{0}_{3\times 2} & \mathbf{A} \end{pmatrix},$$

we group elements in pairs, reducing the problem to $\min(\min(x_1, x_2), \dots, \min(x_{d-1}, x_d))$ ($\mathbf{0}_{m\times n}$ is a matrix of zeros with m rows and n columns). Then, for $\mathbf{x} \in \mathbb{R}^{d+2}$, we use the inductive step to recover $\min(\min(x_1, x_2), \dots, \min(x_{d-1}, x_d), \min(x_{d+1}, x_{d+2}))$ by using $\mathbf{S}''\sigma(\mathbf{A}''\mathbf{x})$ where

$$\mathbf{S}'' := \begin{pmatrix} \mathbf{S}' & \mathbf{0}_{d\times 3} \\ \mathbf{0}_{1\times 3d} & \mathbf{S} \end{pmatrix}, \ \mathbf{A}'' := \begin{pmatrix} \mathbf{A}' & \mathbf{0}_{3d\times 2} \\ \mathbf{0}_{3\times 2d} & \mathbf{A} \end{pmatrix}. \tag{14}$$

Indeed, if with $\mathbf{S}'$ and $\mathbf{A}'$ we group variables into pairs, then we can also group in pair two additional elements. Now let's consider the inductive step for an odd case $\mathbf{x} \in \mathbb{R}^{d+1}$. We can recover $\min(\min(x_1, x_2), \dots, \min(x_{d-1}, x_d), x_{d+1})$ by using $\mathbf{S}^\bullet\sigma(\mathbf{A}^\bullet\mathbf{x})$, where

$$\mathbf{S}^\bullet := \begin{pmatrix} \mathbf{S}' & \mathbf{0}_{d\times 1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{2\times 3d} & 1 & -1 \end{pmatrix}, \ \mathbf{A}^\bullet := \begin{pmatrix} \mathbf{A}' & \mathbf{0}_{3d\times 1} \\ \mathbf{0}_{1\times 2d} & 1 \\ \mathbf{0}_{1\times 2d} & -1 \end{pmatrix}.$$

For $\mathbf{x} \in \mathbb{R}^{d+3}$, we extend $\mathbf{S}^\bullet$ and $\mathbf{A}^\bullet$ as in Eq. (14), pairing the last two elements. Recursively applying $\mathbf{S}^*\sigma(\mathbf{A}^*\mathbf{x})$ (where * means that the dimensionality must be chosen appropriately) groups the elements in pairs and eventually returns the minimum element. ■

Appendix D. Active Neurons

To further contrast the two presented representations of CantorNet, we compare the ratio of active neurons along a path traversing two classes. The ratio corresponds to the *compactness* of both representations (Fig. 5). Comparing the ratio of active neurons allows to evaluate the efficiency and resource utilization of different neural network architectures. Active neurons consume computational resources, and a network with fewer active neurons, while maintaining performance, indicates a more efficient use of resources (also called *Average Percentage of Zeros*) (Hu et al., 2016). It has also been argued that sparse models are more interpretable than dense models (Lipton, 2018). For CantorNet, we find out that for the recursion-based representation the ratio is consistently higher than for the disjunctive-normal form construction. For an experimental evaluation, in Fig. 5 we compare ratios of active neurons for initial recursion levels.

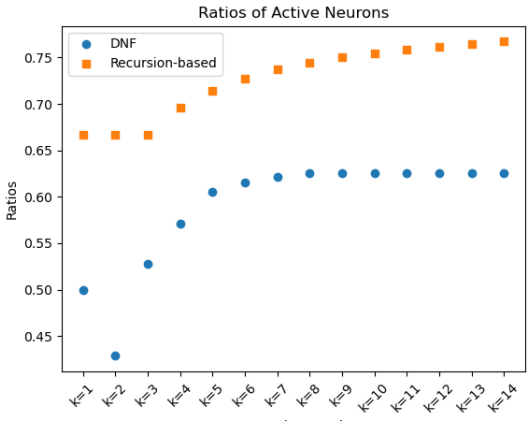

Figure 5: The ratio of the active neurons depending on the recursion depth k for the both representations $\mathcal{N}_A$ and $\mathcal{N}_B$. Note that the higher ratio of active neurons for $\mathcal{N}_A$ is consistent with the minimal complexity of $\mathcal{N}_A$ due to Lemma 5.