Supplementary material for CantorNet: ReLU Networks as a Sandbox for Studying Topological Properties Through the Activation Space

Michał Lewandowski[1,2] and Bernhard A.Moser[1,2]

1- Software Competence Center Hagenberg (SCCH)
Softwarepark 32a, 4232 Hagenberg, Austria

2- Institute of Signal Processing at Johannes Kepler University (JKU)
Altenberger Straße 66b, 4040 Linz, Austria

A Derivations

In this section, we provide derivations used to describe CantorNet. We start with providing an extensions to an arbitrary number of elements n of our constructions of min as a ReLU neural network.

Construction A.1. *For $x_1, x_2 \in \mathbb{R}$ it holds that $x_1 = \max(x_1, 0) - \max(-x_1, 0)$. Merging it with $\min(x_1, x_2) = x_2 + \min(x_1 - x_2, 0) = x_2 - \max(x_2 - x_1, 0)$ we obtain*

$$\begin{aligned}\min(x_1, x_2) &= x_2 - \max(x_2 - x_1, 0) \\ &= \mathit{relu}(x_2) - \mathit{relu}(-x_2) - \mathit{relu}(-x_1 + x_2),\end{aligned}$$

which can be represented as a ReLU neural network as follows

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

Similarly, for three elements we have that

Construction A.2 ($\min(x_1, x_2, x_3)$ as a ReLU neural network for $x_1, x_2, x_3 \in \mathbb{R}$)**.** *We note that $\min(x_1, x_2, x_3) = \min(\min(x_1, x_2), x_3)$, thus $\min(x_1, x_2, x_3)$ is given by*

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

A.1 Extensions to arbitrary number of elements

Denote by $\mathbf{0}_{m \times n}$ a zero matrix with m rows and n columns, and by

$$\mathbf{A} := \begin{pmatrix}0 & 1 \\ 0 & -1 \\ -1 & 1\end{pmatrix}, \quad \mathbf{S} := \begin{pmatrix}1 & -1 & -1\end{pmatrix}.$$

Then, the general formula for a ReLU neural network that returns a minimum of odd/even elements is as follows.

Construction A.3. *Assume* $\mathbf{x} \in \mathbb{R}^{2k}$ *for* $k \in \mathbb{N}_+$*, then the last hidden layer is described by* $\mathbf{S}'\sigma(\mathbf{A}'\mathbf{x})$*, where* $\mathbf{S}' \in \mathbb{R}^{k\times 3k}$*,* $\mathbf{A}' \in \mathbb{R}^{3k\times 2k}$ *are as follows*

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

For $\mathbf{x} \in \mathbb{R}^{2k+1}$ *we use matrices* $\mathbf{S}', \mathbf{A}'$ *modified as follows*

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

Recursively applying $\mathbf{S}^*\sigma\mathbf{A}^*\mathbf{x}$ (where * means that the dimensionality must be chosen appropriately) reduces the problem from n to $\lceil n/2 \rceil$ elements, and eventually returns the minimum element.

B CantorNet

We recall the construction of synthetic example used in [Moser et al., 2022].

Example 1. *We use the function* $A(x) := \max\{-3x + 1, 0, 3x - 2\}$ *on* $[0, 1]$ *and recursively nest it as* $A^{(k+1)}(x) := A(A^{(k)}(x))$*,* $A^{(1)}(x) := A(x)$*, defining decision surfaces as the upper borders of regions*

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

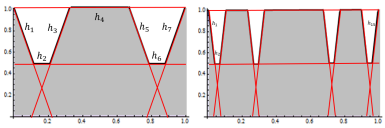

Fig. 1: Decision surfaces ($k = 2, 3$) of Example 1 with labeled half-spaces.

Representation A. *The same shape of decision boundaries can be recovered with a neural network-like structure given by recursive application of*

$$W_{2j-1} = \begin{pmatrix} -3 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}^T, b_{2j-1} = \begin{pmatrix} 1 & -2 & 0 \end{pmatrix}^T, W_{2j} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, b_{2j} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

with the final layer expressed by $W_L = \begin{pmatrix} -\frac{1}{2} & 1 \end{pmatrix}, b_L = \begin{pmatrix} -\frac{1}{2} \end{pmatrix}$*. For example, for* $k = 1$ *in Eq* (1)*, we transform an input* $x \in \mathbb{R}^2$ *as* $x \mapsto W_L\sigma(W'x) + b_L$*, and for* $k = 2$ *as* $x \mapsto W_L\sigma(W'(\sigma(W'x)) + b_L$*, with* $W'x := W_2\sigma(W_1x + b_1) + b_2, \sigma(x) := \max(x, 0)$*. The class assignment is given by the sign of the last transformation.*

Table 1: Min/max shape description for k recursions

k	formula
1	$\min(h_1, h_2, h_3)$
2	$\min(h_1, h_2, h_7, \max(h_3, h_4, h_5))$
...	...
arbitrary	$\min(h_1, h_2, h_{r(k)}, D(1), \ldots, D(\lfloor r(k)/4 \rfloor + 1))$

We can alternatively derive the CantorNet utilizing half-spaces of h-tessellation $h(x, y) = Ax + By + C$. We label them with $0, \ldots, r(k)$, where $r(k) : \mathbb{N} \to \mathbb{N}$ is defined as $r(1) = 3$ and $r(k) = 8k - 9$ for $k \geq 2$ (from left to right, see Figure 1). Put $h_i := h_i(x, y)$, and $D(l) : \mathbb{N} \to \mathbb{R} : l \mapsto \max(h_{4l-1}, h_{4l}, h_{4l+1})$. Then, the decision boundaries are given by

$$\min\left(h_1, h_2, h_{r(k)}, D(1), \ldots, D\left(\lfloor r(k)/4 \rfloor + 1\right)\right) \geq 0$$

(see Table 1 for a formula for arbitrary k), what we solve expressing min function as a ReLU neural network as in above constructions, and replacing $x_1, \ldots x_m$ with

$$\max\left(\begin{pmatrix} A_{11} & A_{12} \\ \ldots & \ldots \\ A_{m1} & A_{m2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} B_1 \\ \ldots \\ B_m \end{pmatrix}, \begin{pmatrix} 0 \\ \ldots \\ 0 \end{pmatrix}\right), \tag{2}$$

where A_{ij} and B_{ij} are coefficients of half-spaces $h_i(x, y)$.

Proposition 1. *Class membership of an input $x \in \mathbb{R}^2$ relies on its h-tessellation: it's "above" decision boundaries if all codes are 1, otherwise it's "below".*

Proof. If all the codes are 1 at the innermost layer, then rows of Eq. (2) are strictly positive, thus x lies above all half-spaces, otherwise it lies below at least one half-space, implying respective class assignment. □

C Proof of Lemma 1

In this section we re-state the Lemma 1 from the main paper, and provide its proof.

Lemma 1 (**Equivalence of Convexity Notions).** *An arrangement of activation regions (of an h-tessellation) in Euclidean space is convex if and only if the corresponding activation patterns form a convex set in the Hamming space.*

Proof. $\Rightarrow$ Consider a tessellation of non-trivial activation regions formed by N hyperplanes $h_1, \ldots, h_N$, with activation regions $R_{\pi_1}, \ldots, R_{\pi_r} \subset \mathbb{R}^n$ and associated activation patterns $\mathcal{A} = \{\pi_1, \ldots, \pi_r\} \subset \{0, 1\}^N$. Suppose the union $R = \bigcup_j R_{\pi_j}$ is convex. We'll prove that $\mathcal{A}$ forms a convex set in the Hamming space.

1. Connectivity: By picking any pair of activation patterns in $\mathcal{A}$, we can show that there exists a path between them by flipping one bit at a time, implying $\mathcal{A}$ is connected.

2. Assuming non-convexity: Assuming $\mathcal{A}$ is not convex in the Hamming space, there exists a shortest path $\widetilde{\gamma}$ between some $\pi_{i_0}, \pi_{j_0} \in \mathcal{A}$ that is not fully inside $\mathcal{A}$. We've shown $\mathcal{A}$ is connected, so we consider a shortest path in R, γ_R, connecting points of $R_{\pi_{i_0}}$ and $R_{\pi_{j_0}}$. Walking along γ_R, we visit activation patterns and corresponding regions in R. Transitions between adjacent activation patterns flip exactly one bit. Since $\widetilde{\gamma}$ isn't in $\mathcal{A}$, there exists an activation pattern $\widetilde{\pi} \notin \mathcal{A}$.

3. Contradiction: We know that the activation pattern $\widetilde{\pi}$ is not in $\mathcal{A}$. However, since the union of activation regions R is convex, we can construct a path between π_{i_0} and π_{j_0} within the convex set R, implying that $\widetilde{\pi}$ must also lie within this convex set. This contradicts our assumption that $\widetilde{\pi} \notin \mathcal{A}$. Hence, the activation patterns $\mathcal{A}$ indeed form a convex set in the Hamming space.

$\Leftarrow$ Suppose we have a finite collection of regions $R = \bigcup_i R_i$ where the corresponding collection $\mathcal{A}$ of activation patterns $\pi(R_i)$ is convex in the Hamming cube. Choose any two points P, Q within R and denote by γ^* the path of activation patterns corresponding to a straight line $\overline{PQ}$ connecting P, Q. Since the length $|\gamma^*| \leq |\gamma|$ is smaller or equal to any other shortest path in $\mathcal{A}$ connecting $\pi(P)$ with $\pi(Q)$, and all shortest paths lie in $\mathcal{A}$, it follows that $\gamma^* \subseteq \mathcal{A}$, and thus $\overline{PQ} \subseteq R$. □

D Experiments

CantorNet. For completeness we append the results of space foldings measure presented in the main paper in Figure 2.

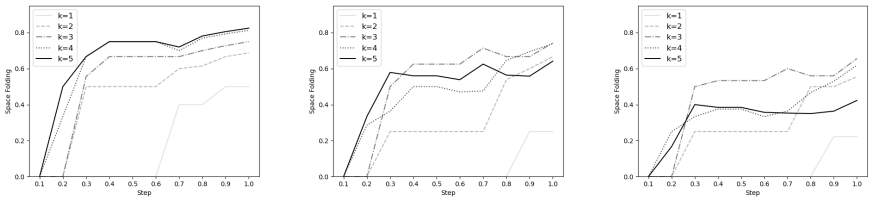

Fig. 2: Space folding measure for Representations A,B,C for $k = 1, \ldots, 5$.

MNIST. In the next step, we explore the relationship between the ratio of active neurons and space foldings measure for neural architectures trained on the MNIST dataset [LeCun and Cortes, 2010]. We train 20 feed forward fully connected ReLU neural networks with varying architectures, ranging from 3 hidden

layers with 10 neurons each to 6 layers with 18 neurons each, in increments of 2 neurons per layer, resulting in a total of 20 different neural architectures $(3 \times 10, \ldots, 3 \times 18, 4 \times 10, \ldots, 4 \times 18, \ldots, 6 \times 18$, where $n \times m$ stand for n layers m neurons each). Each architecture is trained until achieving a minimum validation accuracy of 0.9, and for statistical reasons we train each architecture 100 times. We use the Adam optimizer [Kingma and Ba, 2015].

We calculate the space foldings measure at a straight path between two images representing digits 1 and 4. This process is repeated for 100 different pairs. We then average the results and indicate the standard deviation using shades of the respective colors (refer to Figure 3).

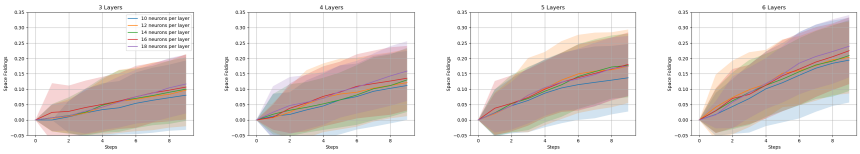

Fig. 3: MNIST dataset space foldings for various neural architectures. Figures show different hidden layer counts, while colors denote different number of neurons per layer. Best viewed in colors.

References

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[Moser et al., 2022] Moser, B. A., Lewandowski, M., Kargaran, S., Zellinger, W., Biggio, B., and Koutschan, C. (2022). Tessellation-filtering relu neural networks. *IJCAI*.