HINGING HYPERPLANES TREE: TRANSFORMING RELU NEURAL NETWORKS INTO EQUIVALENT DECISION TREES

Anonymous authors
Paper under double-blind review

ABSTRACT

ReLU neural networks define continuous piecewise-affine functions, but this function-level structure is represented implicitly through neuron activations. We introduce the *Hinging Hyperplanes Tree* (HHT), which converts a fixed ReLU network into a tree of affine comparisons with affine output rules at its leaves. HHT is constructed from locally extracted affine functions and refined through leaf-wise linear-programming containment checks. We prove finite convergence of the sample-level construction and show that, once all leaves are certified, the resulting HHT is functionally equivalent to the source network over the prescribed bounded domain. Unlike a forward pass, which reveals the affine rule only at a queried input, a certified HHT provides a reusable polyhedral representation of the network's piecewise-affine function. Experiments demonstrate exact reconstruction on the studied ReLU MLPs and characterize the structural and computational trade-offs relative to existing network-to-tree representations.

1 INTRODUCTION

ReLU neural networks define continuous piecewise-affine (PWA) functions: their input domains are partitioned into polyhedral regions on which the network is affine (Montúfar et al., 2014; Balestriero & Baraniuk, 2018; Hanin & Rolnick, 2019). However, this function-level structure is represented only implicitly through the network's neuron activations. In this work, we consider the problem of making this structure explicit. Given a fixed, trained ReLU network and a bounded input domain Ω, we seek an exact standalone representation of its PWA function rather than a learned replacement predictor. To this end, we introduce the **Hinging Hyperplanes Tree (HHT)**, whose internal nodes compare affine functions and whose leaves store affine output rules.

A forward pass can identify the affine rule active at a particular input, but it does not yield a reusable decomposition of the domain. In contrast, a certified HHT explicitly represents polyhedral subregions together with their affine laws. We therefore view HHT as a form of *function-level compilation*: local affine pieces are extracted from the source network, organized into a tree, and examined through leaf-wise linear-programming containment checks. Failed checks return witness points that trigger further refinement. The finite-termination result applies to each fixed-sample tree construction; whenever the adaptive refinement procedure reaches a state in which every leaf passes the containment certificate, the resulting HHT is functionally equivalent to the source network over Ω.

Our main contributions are: (1) We introduce HHT, an explicit function-level tree representation of a fixed ReLU network, with affine comparisons at internal nodes and affine output rules at leaves. (2) We develop a sampling-and-refinement algorithm that constructs the HHT and uses LP-based containment checks to identify missing regions. (3) We prove finite termination of the tree construction for any fixed sample set satisfying the stated assumptions, and prove functional equivalence to the source network over the bounded domain whenever all leaves pass the containment certificate. (4) We demonstrate how a certified HHT can be reused for exact output-range queries under box and general polyhedral constraints, and characterize its construction and representation costs relative to existing network-to-tree representations.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

Tree Extraction from Networks. Decision trees are widely used to interpret neural networks. Early methods extract approximate trees via black-box querying (Craven & Shavlik, 1995), rule decomposition (Zilke et al., 2016), or distillation (Frosst & Hinton, 2017). Recent work shifts focus to structural equivalence: piecewise-linear activations induce hierarchical partitions (Aytekin, 2022), and network derivatives yield locally constant functions equivalent to oblique trees (Lee & Jaakkola, 2020). Notably, RENTT transforms networks into exact multivariate trees to provide faithful explanations (Monke et al., 2025). However, unlike RENTT's reliance on sequential neuron-wise activation tests, our HHT circumvents explicit activation bookkeeping by directly extracting the geometric boundaries between local affine pieces. Thus, although both RENTT and HHT represent a fixed ReLU network, RENTT organizes neuron-level activation decisions, whereas HHT organizes the realized function through affine comparisons and affine output regions.

Piecewise Affine Geometry. ReLU networks partition the input space into polyhedral regions, where each activation pattern defines a local affine map (Montúfar et al., 2014; Raghu et al., 2017; Arora et al., 2018). Frameworks like max-affine splines (Balestriero & Baraniuk, 2018) and tropical geometry (Zhang et al., 2018) formalize this input-conditioned affine structure, facilitating studies on the complexity and adjacency of linear regions (Hanin & Rolnick, 2019; Montúfar et al., 2014; Humayun et al., 2023). However, since explicitly enumerating all regions is intractable, recent methods use sampling to capture salient affine pieces (Berzins, 2023).

Hinging Hyperplanes and Oblique Trees. Our work relates to classical piecewise affine models, such as Breiman's hinging hyperplanes (Breiman, 1993) and MARS (Friedman, 1991), as well as recent hinge-based trees (Li et al., 2026). While traditional oblique trees enhance expressivity using general hyperplanes learned directly from data (Murthy et al., 1994; Bertsimas & Dunn, 2017; Verwer & Zhang, 2019), HHT instead extracts exact hinge comparisons from the local affine pieces of a fixed ReLU network, certifying equivalence via containment checks.

3 Obtaining Affine Functions Through Sample Points

A feedforward ReLU network defines a continuous piecewise affine mapping $f : \mathbb{R}^d \to \mathbb{R}^m$. For each feasible activation pattern $\mathcal{P}$, let $\mathcal{R}(\mathcal{P})$ denote its activation region. The network admits the affine representation

$$f(x) = \ell_{\mathcal{P}}(x) := A(\mathcal{P})x + b(\mathcal{P}), \qquad x \in \mathcal{R}(\mathcal{P}). \tag{1}$$

The expressions for $A(\mathcal{P})$ and $b(\mathcal{P})$ are given in Appendix A.

Given sampled inputs $x^{(k)}$ and their activation patterns $\mathcal{P}^{(k)}$, we construct the sampled affine-function pool

$$\mathcal{L} := \{\ell_{\mathcal{P}^{(k)}} : k = 1, \ldots, N\}. \tag{2}$$

Identical affine functions are deduplicated in $\mathcal{L}$, while all sampled activation patterns are retained. Thus, different activation patterns may correspond to the same affine function.

We assume the sampled data satisfy the following conditions:

Assumption 1. For every affine function $(A(\mathcal{P}), b(\mathcal{P}))$ represented in $\mathcal{L}$, at least one sampled input inducing this affine function lies in the relative interior of a corresponding activation region $\mathcal{R}(\mathcal{P})$.

Assumption 2. Let $\Omega \subseteq \mathbb{R}^d$ be a full-dimensional convex domain, with $x^{(k)} \in \mathrm{int}(\Omega)$ for every k. There exists a continuous piecewise affine function $f_c : \Omega \to \mathbb{R}$, with affine pieces in $\mathcal{L}$, such that for every sample $x^{(k)}$, an open neighborhood $U_k \subseteq \Omega$ of $x^{(k)}$ satisfies

$$f_c(x) = \ell^{(k)}(x), \qquad x \in U_k, \quad \text{where } \ell^{(k)}(x) = A(\mathcal{P}^{(k)})x + b(\mathcal{P}^{(k)}).$$

Assumption 2 is a sample-level consistency condition rather than a global coverage assumption. Following Xu et al. (2025), the required ordering relations can be checked directly on the sampled points. If an ordering relation is violated between two samples, an intermediate point on their connecting segment can be added to reveal an affine piece that repairs the violation. Since a finite

ReLU network has only finitely many affine pieces along any line segment, this refinement terminates after finitely many additions. Assumption 2 is therefore used only to guarantee coverage-enlarging hinges in the sample-level construction; global equivalence over Ω is established separately by the leaf-wise LP containment condition in Theorem 2.

4 HINGING HYPERPLANE TREES

4.1 TREE CONSTRUCTION

Given sample inputs $x^{(k)}$, $k = 1, \dots, N$, for a target ReLU neural network, we obtain the corresponding active patterns $\mathcal{P}^{(k)}$ and network outputs $y^{(k)} = nn(x^{(k)})$. Based on these quantities, we construct a binary tree, referred to as a *hinging hyperplanes tree* (HHT) $\mathcal{T}_{\mathrm{HH}}$, to exactly mimic the input output behavior of the target ReLU network. The terminology originates from (Breiman, 1993) and from the fact that each internal node of the tree evaluates a hinge function.

Starting from the root node $\mathbb{D}_{0,1}$, the HHT grows through internal nodes to leaf nodes, and we denote by $\mathbb{D}_{i,j}$ the j-th node at depth i. By a slight abuse of notation, when $\mathbb{D}_{i,j}$ appears in a set relation or an H-representation, it denotes the corresponding input-space cell induced by the accumulated hinge inequalities; otherwise it denotes the tree node itself. Throughout this subsection, $K_{i,j}$ denotes the set of indices $k \in S_{i,j}$ whose routed affine operand is identical, as an affine function, to the sample's associated affine function $\ell^{(k)}$. Ties are routed to the first operand.

Root node. At the root node $\mathbb{D}_{0,1}$, two affine functions $\ell_{0,1}(x), \ell'_{0,1}(x) \in \mathcal{L}$ are selected together with an operator $\mathrm{op} \in \{\max, \min\}$, such that there exists an index set $K_{0,1} \subset \{1, \dots, N\}$ satisfying

$$\mathrm{op}\{\ell_{0,1}(x^{(k)}), \ell'_{0,1}(x^{(k)})\} = y^{(k)}, \quad \forall k \in K_{0,1}.$$

The hinge function at the root is therefore

$$h_{0,1}(x) = \mathrm{op}\{\ell_{0,1}(x), \ell'_{0,1}(x)\}.$$

The sample index set $S_{0,1} = \{1, \dots, N\}$ is split into two disjoint subsets

$$S_{1,1} = \{k \in S_{0,1} \mid \mathrm{op}\{\ell_{0,1}(x^{(k)}), \ell'_{0,1}(x^{(k)})\} = \ell_{0,1}(x^{(k)})\}, \quad S_{1,2} = S_{0,1} \setminus S_{1,1},$$

which correspond to the child nodes $\mathbb{D}_{1,1}$ and $\mathbb{D}_{1,2}$, respectively. We designate $\ell_{1,1} = \ell_{0,1}$ and $\ell_{1,2} = \ell'_{0,1}$ as the *active affine functions* for the respective child nodes. This implies that $\ell_{1,1}$ (resp. $\ell_{1,2}$) is the affine component that *activates* (or determines the value of) the hinge function over the subset $S_{1,1}$ (resp. $S_{1,2}$).

For notational consistency, we set $C_{0,1} := \emptyset$, since the root has no covered set inherited from a parent node. Let $C_{1,1} = K_{0,1} \cap S_{1,1}$ and $C_{1,2} = K_{0,1} \cap S_{1,2}$.

Internal nodes. Consider a nonterminal node $D_{i,j}$ with sample index set $S_{i,j}$ and inherited affine function $\ell_{i,j}$. We first keep $\ell_{i,j}$ fixed and select $\mathrm{op} \in \{\max, \min\}$ and a partner $\ell'_{i,j} \in \mathcal{L}$, requiring $C_{i,j} \subseteq K_{i,j} \subseteq S_{i,j}$. If no such candidate exists, we search all affine pairs in $\mathcal{L}$ without fixing the inherited affine function or requiring $C_{i,j} \subseteq K_{i,j}$. In both searches, candidates must produce two nonempty child sample sets, and a candidate maximizing $|K_{i,j}|$ is selected. After selection, $\ell_{i,j}$ and $\ell'_{i,j}$ denote the selected first and second operands, respectively, so that

$$h_{i,j}(x) = \mathrm{op}\{\ell_{i,j}(x), \ell'_{i,j}(x)\}, \qquad h_{i,j}(x^{(k)}) = y^{(k)}, \quad k \in K_{i,j}.$$

Termination. A node becomes a leaf when all its assigned samples share the same associated affine function, which is stored as its output; their activation patterns may differ. The construction continues until no nonterminal nodes remain. The sets $C_{i,j}$ record provisional coverage and are not accumulated into a global stopping set. Under the assumptions of Theorem 1, the resulting finite HHT exactly reproduces the network outputs on the given samples. The detailed algorithm is given in Algorithm 1 in Appendix C.1.

4.2 Convergence Analysis of the Algorithm

After evaluating Algorithm 1, we obtain an HHT $\mathcal{T}_{\mathrm{HH}}$. We arrange all nodes in a breadth-first order $\{\mathbb{D}_t\}_{t=1}^{T}$, and denote the index sets of internal and leaf nodes by $\mathbb{T}_{\mathrm{I}}$ and $\mathbb{T}_{\mathrm{L}}$, respectively.

The following theorem shows that Algorithm 1 terminates after finitely many splits and that the resulting tree interpolates the given samples (hence mimics the input–output behavior of the target ReLU network on the sample set).

Theorem 1. *Suppose the sample data $\{(x^{(k)}, y^{(k)}, \mathcal{P}^{(k)})\}_{k=1}^{N}$, $N \geq 1$, satisfy Assumptions 1–2. In exact arithmetic, Algorithm 1 terminates after at most $N-1$ splits and produces a finite tree T_{HH} with*

$$T_{\mathrm{HH}}(x^{(k)}) = y^{(k)}, \qquad k = 1, \ldots, N.$$

Every leaf outputs an affine function $\ell_{m_t} \in \mathcal{L}$, and

$$\mathcal{L} \subseteq \{\ell_{m_t} : t \in \mathcal{T}_L\}.$$

Different activation patterns with identical affine coefficients may share a leaf.

4.3 Equivalence of HHT and ReLU neural network

4.3.1 Equivalence within Specified Regions

The equivalence of the constructed HHT and the ReLU neural network within specified regions is shown in the following theorem.

Theorem 2. *Assume the HHT is constructed using Algorithm 1, and let its leaf nodes with nonempty assigned sample sets be D_t, $t \in \mathcal{T}_L$. Let $\mathcal{P}$ denote the feasible activation patterns of the ReLU neural network in Ω. For each leaf with output affine function ℓ_{m_t}, define*

$$\mathfrak{P}_t := \{\mathcal{P} : \mathcal{R}(\mathcal{P}) \cap \Omega \neq \emptyset,\ A(\mathcal{P})x + b(\mathcal{P}) \equiv \ell_{m_t}(x)\},$$

and

$$\mathcal{R}_t := \bigcup_{\mathcal{P} \in \mathfrak{P}_t} \overline{\mathcal{R}(\mathcal{P})}.$$

The affine representation remains valid on the region closures by continuity. Then, for all $t \in \mathcal{T}_L$,

$$f_{\mathrm{HHT}}(x) = f_{\mathrm{ReLU}}(x), \qquad \forall x \in D_t \cap \mathcal{R}_t. \tag{3}$$

Besides, if

$$D_t \subseteq \mathcal{R}_t, \qquad \forall t \in \mathcal{T}_L, \tag{4}$$

the HHT is identical to the ReLU neural network in the domain, i.e.,

$$f_{\mathrm{HHT}}(x) = f_{\mathrm{ReLU}}(x), \qquad \forall x \in \Omega. \tag{5}$$

Remark 1. According to Theorem 2, if (4) holds for all leaf nodes, the HHT is equivalent to the ReLU neural network. More generally, the same conclusion holds for general piecewise affine neural networks, provided that the corresponding polyhedral regions can be obtained.

Certification. To verify equation 4, consider the polyhedral representations

$$\overline{D_t} = \{x : A_t x \leq b_t\}, \qquad \overline{\mathcal{R}(\mathcal{P})} = \{x : C_{t,\mathcal{P}} x \leq d_{t,\mathcal{P}}\}, \quad \mathcal{P} \in \mathfrak{P}_t. \tag{6}$$

These representations follow from the tree's routing and domain constraints and the network's activation patterns, respectively. Redundant inequalities can first be removed from each activation region to reduce the verification cost.

Since $\mathcal{R}_t$ is a finite union of closed polyhedra, the required containment is equivalent to

$$\overline{D_t} \setminus \bigcup_{\mathcal{P} \in \mathfrak{P}_t} \overline{\mathcal{R}(\mathcal{P})} = \emptyset. \tag{7}$$

A point outside this union must violate at least one defining inequality of every compatible region simultaneously. Algorithm 2 therefore considers combinations containing one inequality from each region and uses an LP to maximize their common violation margin over $\overline{D_t}$. A positive margin provides a witness $x^\star \in \overline{D_t} \setminus \mathcal{R}_t$. If no combination admits a positive margin, the containment condition holds and the leaf is certified. The detailed LP formulation is given with Algorithm 2 in the appendix.

4.3.2 RESAMPLING AND RECONSTRUCTION OF THE HHT

When the containment test returns a witness $x^\star$, choose an existing sample x^{in} in the same leaf. Its associated affine function equals the leaf output, so $x^{\text{in}} \in D_t \cap \mathcal{R}_t$. Consider the segment

$$x(\lambda) = x^{\text{in}} + \lambda(x^\star - x^{\text{in}}), \qquad \lambda \in [0, 1]. \tag{8}$$

For each $\mathcal{P} \in \mathfrak{P}_t$, the segment's intersection with $\overline{\mathcal{R}(\mathcal{P})}$ corresponds to a closed parameter interval, possibly empty. Computing these intervals gives the last parameter at which the segment belongs to their union:

$$\lambda_{\text{last}} := \max\{\lambda \in [0, 1] : x(\lambda) \in \mathcal{R}_t\}. \tag{9}$$

Because x^{in} belongs to the closed union and $x^\star$ lies outside it, λ_{last} exists and is strictly less than one. Thus every $\lambda \in (\lambda_{\text{last}}, 1)$ yields $x(\lambda) \in D_t \setminus \mathcal{R}_t$. Taking $\lambda < 1$ also preserves the leaf's strict routing inequalities when the LP witness lies on a boundary of $\overline{D_t}$.

Query the target network at the generated points to obtain their outputs, activation patterns, and induced affine functions. Add the points to the construction data, retain their pattern labels, and update the deduplicated affine pool. Once the enlarged data satisfy the assumptions of Theorem 1, refine the affected leaf using Algorithm 1 and verify its new leaves again. Previously certified leaves remain unchanged throughout this local refinement procedure.

4.4 COMPLEXITY ANALYSIS

The following lemma demonstrates the offline training complexity of the HHT.

Lemma 1. *Let $\{x^{(k)}\}_{k=1}^N \subset \mathbb{R}^d$ be offline samples. Assume the samples induce only L distinct affine pieces (equivalently L distinct activation modes), and let $\mathcal{L}$ be the deduplicated affine pool with $|\mathcal{L}| = L$ (typically $N \gg L$). Let $\mathcal{V}_{\text{proc}}$ be the set of nodes processed by Algorithm 1, and let S_v be the index set stored at node v. Define $V_N := \sum_{v \in \mathcal{V}_{\text{proc}}} |S_v|$. The following properties hold:*

(1) Offline time for tree construction: *At a non-root node, the ordinary inherited-affine search examines $O(L)$ candidates, whereas the all-pairs fallback examines $O(L^2)$ candidates in the worst case. Therefore,*

$$T_{\text{build}} = O(L^2 d V_N)$$

in the worst case. If the all-pairs fallback is never invoked, this reduces to $O(L d V_N)$. For roughly balanced sample splits, $V_N = O(N \log N)$, yielding

$$T_{\text{build}} = O(L^2 N d \log N)$$

in the worst case.

(2) Worst-case depth. *Because every nonterminal split produces two nonempty child sample sets, Theorem 1 implies*

$$D_{\max} \le N - 1.$$

If every split additionally removes at least one distinct affine label from the child followed along a path, the stronger conditional bound $D_{\max} \le L - 1$ holds.

(3) Offline time for leaf verification/resampling. *For each leaf t, let n_t be the constraint count of $\overline{D}_t$, $k_t = |\mathfrak{P}_t|$, and let r_t and $\widetilde{r}_t$ bound the constraint count of each compatible region before and after pruning. Given these region representations, one pass of Algorithm 2 costs*

$$\begin{aligned} T_{\text{ver/res}} = O\Bigg(\sum_{t \in \mathcal{T}_L} \Big[& k_t r_t T_{\text{LP}}(d, r_t) \\ & + \widetilde{r}_t^{\,k_t} T_{\text{LP}}(d+1, n_t + k_t + 2) + d(k_t \widetilde{r}_t + M + 1) \Big] \Bigg), \end{aligned} \tag{10}$$

where $T_{\text{LP}}(p, q)$ is the cost of forming and solving an LP with p variables and q inequalities, and $M + 1$ is the number of resampled points per witness. Network queries and reconstruction are counted separately.

Storage requirement. Storing samples and the deduplicated affine pool during construction requires $O(Nd + Ld)$ memory (for scalar outputs). Storing the explicit node index sets costs an additional $O(V_N)$ memory, while retaining the certified leaf polyhedra contributes $O\big(\sum_{t\in\mathbb{T}_L}(m_t + r_t)d\big)$. For prediction-only deployment, the construction samples and explicit leaf H-representations need not be retained. In contrast, applications that use the compiled HHT for certified polyhedral queries retain the leaf constraints; the corresponding storage cost therefore includes the latter term.

Online inference. At test time, each input follows exactly one root-to-leaf path. At an internal node comparing two affine functions $\ell_a(x) = w_a^\top x + b_a$ and $\ell_b(x) = w_b^\top x + b_b$, the decision can be made by evaluating a single affine difference $(w_a - w_b)^\top x + (b_a - b_b)$. Thus, the per-sample inference cost is $O(D(x)d)$, or $O(D_{\max}d)$ in the worst case. In our implementation, inference is performed by batched tree traversal: samples reaching the same node are grouped together, and the corresponding affine-difference comparison is evaluated by vectorized matrix operations. This does not change the per-sample tree-traversal complexity, but it reduces control-flow overhead in batched evaluation.

5 Experiments

All experiments were conducted using Python 3.11.7 on a machine equipped with an Intel® Xeon® Gold 6530 CPU. The source code is available in the supplementary material.

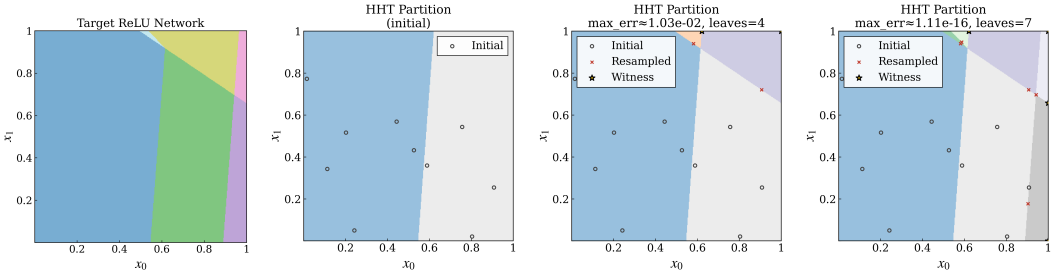
5.1 Illustrative Example and Width Scaling in Two Dimensions

Visualizing the HHT Construction. We first demonstrate the HHT transformation using a two-hidden-layer ReLU network (sizes [2, 3]), adopted from (Monke et al., 2025) and detailed in Appendix D. While the RENTT method (Monke et al., 2025) requires a six-level, 32-node tree for this architecture, HHT efficiently recovers the exact boundaries through a dynamic refinement process. Figure 1 illustrates this stage-by-stage construction. Starting with an initial 10 sample points, the resulting partition (Figure 1b) does not yet match the target network (Figure 1a). Because the termination condition (4) remains unmet, the algorithm resamples points. After two refinement iterations—adding 4 points (Figure 1c) and then 6 points (Figure 1d)—the final 20-sample HHT perfectly replicates the target's piecewise affine decision boundaries across the domain. The equivalent tree structure, which requires only 7 nodes compared to RENTT's 32, is depicted in Figure 3 in Appendix D.

Quantitative Width-Scaling Evaluation. To verify that this exactness scales beyond the toy example, we evaluate HHT on synthetic ReLU networks of increasing widths. Detailed experimental settings for the synthetic evaluations, including data generation, network initialization, and hyperparameters for the HHT construction, are provided in Appendix E. We measure fidelity using Mean Absolute Error (**FidMAE**) and Maximum Error (**FidMax**):

$$\text{FidMAE} = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} |f_{\text{nn}}(x) - f_{\text{tree}}(x)|, \quad \text{FidMax} = \max_{x\in\mathcal{D}} |f_{\text{nn}}(x) - f_{\text{tree}}(x)| \tag{11}$$

where $\mathcal{D}$ represents the evaluation set. Note that **FidMax** is the critical metric for equivalence; a strictly equivalent transformation must yield a max error close to machine precision.



(a) Target ReLU network (b) Initial HHT (10 pts) (c) Refinement 1 (14 pts) (d) Refinement 2 (20 pts)

Figure 1: Target ReLU partition and HHT partitions over successive refinement stages.

Table 1: Performance comparison between HHT and RENTT on two-hidden-layer ReLU networks with varying numbers of neurons. Results represent the mean ± standard deviation across five runs.

Structure	Method	FidMAE↓	FidMax↓	Leaves	Depth	Build(s)↓	Infer(μs/pt)↓	Storage(KB)↓
[10,10]	RENTT (Sparse)	**9.2e-17** ± 3.4e-17	**0.000** ± 0.000	36.2 ± 10.9	20.0 ± 0.0	**0.030** ± 0.007	31.18 ± 1.01	34.92 ± 9.91
	RENTT (Comp.)	**9.2e-17** ± 3.4e-17	**0.000** ± 0.000	1048576.0 ± 0.0	20.0 ± 0.0	137.026 ± 0.815	29.32 ± 0.20	265894.97 ± 0.00
	HHT (Ours)	**9.4e-17** ± 4.9e-17	**0.000** ± 0.000	44.6 ± 12.0	8.6 ± 1.0	1.341 ± 0.360	**0.51** ± 0.09	**5.20** ± 1.35
[50,50]	RENTT (Sparse)	**2.4e-16** ± 4.0e-17	**0.000** ± 0.000	410.4 ± 16.7	100.0 ± 0.0	**1.753** ± 0.148	239.61 ± 32.10	4102.13 ± 357.22
	RENTT (Comp.)	–	–	–	–	–	–	–
	HHT (Ours)	**2.3e-16** ± 9.6e-17	**0.000** ± 0.000	962.6 ± 122.1	18.4 ± 2.6	121.042 ± 16.218	**4.53** ± 0.20	**106.14** ± 12.97
[100,100]	RENTT (Sparse)	**4.5e-16** ± 1.6e-16	**0.000** ± 0.000	1100.2 ± 116.0	200.0 ± 0.0	**12.373** ± 0.846	1781.61 ± 232.98	38020.20 ± 2509.60
	RENTT (Comp.)	–	–	–	–	–	–	–
	HHT (Ours)	**3.2e-16** ± 8.6e-17	**0.000** ± 0.000	4224.2 ± 824.1	21.6 ± 1.0	1081.453 ± 213.576	**10.80** ± 1.01	**462.77** ± 88.41

We compare HHT with RENTT (Monke et al., 2025), including its **Sparse** and **Complete** variants. Table 1 highlights their different representation costs. RENTT (Sparse) constructs quickly through neuron-level activation decisions, whereas HHT incurs a larger offline cost for discovering and certifying affine regions. For the wider networks studied here, the resulting HHT uses much shorter affine-comparison paths and substantially less inference-only storage than RENTT (Sparse). RENTT (Complete) is omitted for larger networks because materializing the full activation-pattern tree becomes computationally intractable.

5.2 Benchmark Evaluation

After validating structural correctness and scalability on synthetic networks, we evaluate HHT for translating ReLU networks into equivalent tree representations on standard benchmarks, measuring reconstruction fidelity, model complexity and local traceability.

5.2.1 Datasets and Target Network

Datasets. We utilize a diverse suite of datasets: (1) **Synthetic Classification**: Two-dimensional `moons` and `circles` datasets (N=2000). (2) **Real-world Classification**: The `breast_cancer` dataset (9 features, N=699). (3) **Regression**: `diabetes` (10 features, N=442), `airfoil` (5 features, N=1503), and `abalone` (8 features, N=4177). For all datasets, we employ a random 75%/25% train-test split.

Target Neural Networks. For each task, we train a standard Multi-Layer Perceptron (MLP) with ReLU activations to serve as the **target Model** to be translated. The architecture consists of two hidden layers with 6 neurons each (6×6). For **regression**, the target model $f_{\text{nn}}(x)$ outputs the raw continuous scalar prediction. For **classification**, we treat the pre-activation *logit* (before the sigmoid/softmax) as the scalar target $f_{\text{nn}}(x)$. This ensures the function remains continuous piecewise affine, allowing for mathematically exact translation into HHT.

5.2.2 Methods and Baselines

We compare against two differentiable surrogate architectures (LCN and LLN) and the exact network-to-tree transformation RENTT.

(1) LCN and LLN (Lee & Jaakkola, 2020): Neural architectures *theoretically* equivalent to oblique decision trees, trained via differentiable relaxation (activation annealing). LCN models locally constant functions, while LLN extends the framework to locally linear outputs.

(2) RENTT (Monke et al., 2025): A decompositional extraction method that maps neuron activation patterns to decision nodes. We report both its **Sparse** and **Complete** variants. RENTT (Sparse) prunes activation-pattern branches that are not observed in the construction samples and may dynamically expand when unseen activation paths are encountered during inference. RENTT (Complete) materializes the full activation-pattern tree and is therefore computationally expensive.

(3) HHT (Ours): Our proposed method, including sampling, tree construction, witness-driven resampling, and reconstruction. It queries the target network to discover affine pieces within the prescribed domain and organizes them into a tree of hinge comparisons between affine functions.

Table 2: Performance and fidelity evaluation on the **test set**. Results represent the mean ± standard deviation over 5 runs. The **Perf.** column reports RMSE (↓) for regression tasks and Accuracy (↑) for classification tasks.

Dataset	Method	Perf.	FidMAE↓	FidMax↓	Leaves	Depth	Build(s)↓	Infer(μs/pt)↓
Abalone (RMSE ↓)	Target NN	2.150 ± 0.085	–	–	–	–	–	–
	LCN	2.416 ± 0.081	0.870 ± 0.063	8.139 ± 2.259	–	–	33.194 ± 7.423	1.35 ± 0.04
	LLN	2.371 ± 0.188	0.663 ± 0.100	8.314 ± 7.668	–	–	15.042 ± 3.573	**0.47** ± 0.04
	RENTT (Sparse)	2.150 ± 0.085	**1.3e-15** ± 6.5e-17	**0.000** ± 0.000	50.0 ± 15.1	12.0 ± 0.0	**0.029** ± 0.008	29.78 ± 4.05
	RENTT (Comp.)	2.150 ± 0.085	**1.3e-15** ± 6.5e-17	**0.000** ± 0.000	4096.0 ± 0.0	12.0 ± 0.0	0.596 ± 0.055	26.61 ± 0.22
	HHT (Ours)	2.150 ± 0.085	**1.2e-15** ± 7.3e-17	**0.000** ± 0.000	2414.6 ± 747.4	19.0 ± 1.2	53.395 ± 17.748	1.30 ± 0.27
Airfoil (RMSE ↓)	Target NN	4.782 ± 0.188	–	–	–	–	–	–
	LCN	7.922 ± 0.951	4.934 ± 0.721	28.045 ± 9.250	–	–	14.374 ± 2.023	3.98 ± 0.18
	LLN	84.051 ± 32.721	80.608 ± 35.004	115.967 ± 24.129	–	–	6.539 ± 0.750	1.45 ± 0.15
	RENTT (Sparse)	4.782 ± 0.188	**1.2e-14** ± 1.1e-15	**0.000** ± 0.000	18.4 ± 10.4	12.0 ± 0.0	**0.013** ± 0.005	32.16 ± 6.59
	RENTT (Comp.)	4.782 ± 0.188	**1.2e-14** ± 1.1e-15	**0.000** ± 0.000	4096.0 ± 0.0	12.0 ± 0.0	0.573 ± 0.050	27.00 ± 0.13
	HHT (Ours)	4.782 ± 0.188	**1.3e-14** ± 4.0e-15	**0.000** ± 0.000	85.0 ± 64.5	8.4 ± 2.6	1.520 ± 1.188	**1.02** ± 0.50
Diabetes (RMSE ↓)	Target NN	59.360 ± 6.860	–	–	–	–	–	–
	LCN	70.592 ± 4.445	40.873 ± 7.528	133.314 ± 31.084	–	–	4.285 ± 0.473	14.00 ± 1.64
	LLN	165.944 ± 7.218	140.902 ± 14.550	300.461 ± 15.537	–	–	1.812 ± 0.530	**5.41** ± 1.47
	RENTT (Sparse)	59.360 ± 6.860	**1.6e-14** ± 2.7e-15	**0.000** ± 0.000	36.8 ± 8.2	12.0 ± 0.0	**0.026** ± 0.006	56.56 ± 15.67
	RENTT (Comp.)	59.360 ± 6.860	**1.6e-14** ± 2.7e-15	**0.000** ± 0.000	4096.0 ± 0.0	12.0 ± 0.0	0.577 ± 0.048	29.75 ± 0.12
	HHT (Ours)	59.360 ± 6.860	**1.5e-14** ± 2.9e-15	**0.000** ± 0.000	1114.0 ± 1818.5	14.0 ± 5.7	25.418 ± 43.603	5.47 ± 0.57
Breast Cancer (Acc ↑)	Target NN	0.931 ± 0.016	–	–	–	–	–	–
	LCN	0.960 ± 0.008	0.310 ± 0.231	1.793 ± 0.169	–	–	10.595 ± 2.040	8.39 ± 1.12
	LLN	0.943 ± 0.008	0.211 ± 0.144	1.568 ± 0.076	–	–	3.992 ± 0.031	**3.12** ± 0.87
	RENTT (Sparse)	0.931 ± 0.016	**1.8e-16** ± 1.2e-16	**0.000** ± 0.000	53.5 ± 7.8	12.0 ± 0.0	**0.030** ± 0.004	81.59 ± 7.13
	RENTT (Comp.)	0.931 ± 0.016	**1.8e-16** ± 1.2e-16	**0.000** ± 0.000	4096.0 ± 0.0	12.0 ± 0.0	0.507 ± 0.011	26.64 ± 0.57
	HHT (Ours)	0.931 ± 0.016	**2.4e-16** ± 1.9e-16	**0.000** ± 0.000	1966.0 ± 496.4	19.5 ± 2.1	41.206 ± 10.317	5.64 ± 0.71
Circles (Acc ↑)	Target NN	0.785 ± 0.160	–	–	–	–	–	–
	LCN	0.587 ± 0.062	0.672 ± 0.421	3.148 ± 2.338	–	–	18.602 ± 2.229	4.94 ± 2.46
	LLN	0.727 ± 0.129	0.439 ± 0.333	1.464 ± 1.172	–	–	8.036 ± 1.274	1.28 ± 0.14
	RENTT (Sparse)	0.785 ± 0.160	**2.8e-16** ± 1.8e-16	**0.000** ± 0.000	41.8 ± 9.3	12.0 ± 0.0	**0.021** ± 0.003	31.61 ± 4.70
	RENTT (Comp.)	0.785 ± 0.160	**2.8e-16** ± 1.8e-16	**0.000** ± 0.000	4096.0 ± 0.0	12.0 ± 0.0	0.576 ± 0.047	26.83 ± 0.22
	HHT (Ours)	0.785 ± 0.160	**3.0e-16** ± 2.1e-16	**0.000** ± 0.000	71.6 ± 18.0	9.8 ± 1.6	1.112 ± 0.278	**1.09** ± 0.24
Moons (Acc ↑)	Target NN	0.894 ± 0.026	–	–	–	–	–	–
	LCN	0.861 ± 0.013	0.672 ± 0.333	3.416 ± 0.916	–	–	18.682 ± 2.073	3.40 ± 0.80
	LLN	0.846 ± 0.014	0.538 ± 0.541	1.865 ± 1.522	–	–	8.304 ± 0.607	2.23 ± 2.11
	RENTT (Sparse)	0.894 ± 0.026	**3.2e-16** ± 1.7e-16	**0.000** ± 0.000	33.0 ± 11.2	12.0 ± 0.0	**0.018** ± 0.004	26.77 ± 1.34
	RENTT (Comp.)	0.894 ± 0.026	**3.2e-16** ± 1.7e-16	**0.000** ± 0.000	4096.0 ± 0.0	12.0 ± 0.0	0.605 ± 0.104	26.89 ± 0.50
	HHT (Ours)	0.894 ± 0.026	**3.4e-16** ± 1.5e-16	**0.000** ± 0.000	66.0 ± 28.0	9.4 ± 2.9	1.069 ± 0.449	**0.92** ± 0.27

When the containment condition in Theorem 2 is satisfied for all leaves, HHT is functionally equivalent to the target ReLU network on Ω.

Evaluation Metrics. We evaluate the proposed method across three dimensions: (1) **Empirical fidelity:** We report FidMAE and FidMax in Eq. (11) to quantify translation error on the evaluation set. (2) **Task performance:** We measure RMSE for regression tasks and accuracy for classification tasks. (3) **Model complexity:** We report the number of leaves, tree depth, construction time, inference latency, and inference-only storage. For RENTT, the reported storage excludes temporary sample memories used during construction, so the comparison reflects deployed inference storage.

5.3 Results Analysis

Table 2 presents the quantitative evaluation averaged over five random seeds.

Functional Fidelity. Optimization-based surrogates (LCN, LLN) exhibit an intrinsic fidelity gap due to the relaxation of architectural constraints, resulting in high max errors (e.g., 115.97 FidMax on Airfoil for LLN). In contrast, constructive methods target exact translation. Notably, both **HHT** and **RENTT** achieve near-zero empirical fidelity errors on the sampled test sets.

Construction and Deployment Cost. Table 2 confirms the construction-versus-inference trade-off. HHT incurs substantially greater offline construction cost than RENTT because its extraction requires containment checks and witness-driven refinement. Once constructed, HHT has lower measured inference latency and inference-only storage than the evaluated RENTT implementations. These results characterize the deployment cost of the compiled representation rather than an acceleration over the source neural network, which remains preferable for ordinary prediction when no explicit tree representation is required.

Structural Auditability and Reusable Exact Queries. For a queried input, HHT exposes the root-to-leaf affine comparisons, the active affine output rule, and the corresponding switching boundaries. More importantly, once the HHT is certified, the same leaf polytopes can be reused for exact output-range queries under new linear constraints by solving leaf-wise LPs. In the two-dimensional synthetic experiments, for ℓ_∞ queries with $\epsilon = 0.1$, the sound CROWN-IBP bound widths were respectively $1.62\times$, $3.76\times$, and $11.76\times$ the exact HHT widths for the 10×10, 50×50, and 100×100 networks. This exactness comes at higher offline compilation and query cost; the comparison is intended to illustrate the benefit of the explicit compiled representation rather than to replace scalable bound-propagation methods. Additional local, general-polyhedral, and global range queries are reported in Appendix G, with local traceability examples in Appendix F.

Geometric Complexity. The reported tree metrics carry distinct structural meanings for each method. For RENTT, tree depth corresponds directly to the number of sequential neuron-activation decisions. Consequently, RENTT (Complete) materializes a full activation-pattern tree whose leaves grow exponentially with network size. While RENTT (Sparse) mitigates this by retaining only observed activation paths, the resulting representation is data-dependent and may require costly dynamic expansion during inference if unseen paths are encountered. Conversely, HHT depth reflects the maximum number of affine-piece comparisons along a root-to-leaf path. Its structure stores explicit affine comparisons rather than neuron-wise tests. Crucially, the required number of HHT samples and splits is governed by the network's intrinsic geometric complexity rather than the raw training set size, an observation further examined in Appendix H.

6 Limitations

HHT is limited by the number of linear regions in the target ReLU network, which can grow exponentially with input dimension and depth. This makes exact extraction costly for large networks, CNNs, and high-dimensional image or text domains. The current method is therefore best suited to ReLU MLPs over bounded low-dimensional domains when exact function preservation and an explicit polyhedral representation justify the offline compilation cost.

7 Conclusion

We introduced the Hinging Hyperplanes Tree (HHT), a function-level compilation of a fixed ReLU network into affine comparisons and affine output rules. The construction discovers local affine pieces through sampling, organizes them hierarchically, and uses LP-based containment checks to expose missing regions and drive refinement. Once all leaves are certified, the resulting HHT is functionally equivalent to the source network over the prescribed bounded domain.

The main value of this representation is the explicit materialization of its piecewise-affine geometry. This turns the trained network into a reusable polyhedral object on which local structure and constrained region-wise queries can be examined directly. Experiments demonstrate exact reconstruction on the studied small ReLU MLPs and characterize the corresponding construction and representation costs. Scaling exact function-level compilation to networks with substantially more affine regions remains an important direction for future work.

AI use statement

We used generative AI tools to assist with language editing, manuscript organization, and presentation. All mathematical claims, proofs, experimental results, code, and citations were reviewed and verified by the authors. The authors take responsibility for the final content of the paper, including any text produced or revised with the assistance of generative AI.

Reproducibility statement

Complete proofs are provided in the appendices, and Algorithms 1 and 2 specify the construction and certification procedures. Detailed network configurations, initialization procedures, hyperparameters, and evaluation protocols are reported in the experimental appendices. Source code is included in the supplementary material.

REFERENCES

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

Caglar Aytekin. Neural networks are decision trees, 2022. arXiv:2210.05189.

Randall Balestriero and Richard G. Baraniuk. A spline theory of deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 374–383, 2018.

Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

Arturs Berzins. Polyhedral complex extraction from ReLU networks using edge subdivision. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 2234–2244, 2023.

Leo Breiman. Hinging hyperplanes for regression, classification and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993.

Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems*, pp. 24–30, 1995.

Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree, 2017. arXiv:1711.09784.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2596–2604, 2019.

Ahmed Imtiaz Humayun, Randall Balestriero, Guha Balakrishnan, and Richard G Baraniuk. Splinecam: Exact visualization and characterization of deep network geometry and decision boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3789–3798, 2023.

Guang-He Lee and Tommi S Jaakkola. Oblique decision trees from derivatives of ReLU networks. In *International Conference on Learning Representations*, 2020.

Hongyi Li, Han Lin, and Jun Xu. Hinge regression tree: A newton method for oblique regression tree splitting. In *International Conference on Learning Representations*, 2026.

Helena Monke, Benjamin Fresz, Marco Bernreuther, Yilin Chen, and Marco F. Huber. Efficiently transforming neural networks into decision trees: A path to ground truth explanations with RENTT, 2025. arXiv:2511.09299.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, 2014.

Sreerama K Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

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.

Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1625–1632, 2019.

Jun Xu, Yunjiang Lou, Bart De Schutter, and Xiong Zhenhua. Error-free approximation of explicit linear mpc through lattice piecewise affine expression. *IEEE Transactions on Automatic Control*, 70(3):1745–1760, 2025.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5824–5832, 2018.

Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pp. 457–473, 2016.

APPENDIX

A Explicit Affine Representation on a Given Region **13**

B Proofs. **13**

B.1 Proof of Theorem 1 13

B.2 Proof of Theorem 2 14

B.3 Proof of Lemma 1 15

C Details of Algorithms **17**

C.1 HHT construction 17

C.2 Union containment verification and resampling 17

D Detailed parameters of ReLU network and HHT of the toy example **20**

D.1 Detailed network parameters 20

D.2 Constructed HHT 20

E Detailed Experimental Setup for the Width-Scaling Evaluation **22**

F Local Traceability Case Studies **22**

G Exact Polyhedral Queries on the Compiled Representation **24**

H Effect of Training Data Size **26**

I Dataset Details **26**

A Explicit Affine Representation on a Given Region

This section provides the detailed recursive expansion for the local affine functions discussed in Section 3.

Consider an N_l-layer fully-connected ReLU network defined by:

$$\begin{aligned} h_0(x) &= x, \\ z_{n_l}(x) &= W_{n_l}\, h_{n_l-1}(x) + b_{n_l}, \qquad n_l = 1, \dots, N_l, \\ h_{n_l}(x) &= \sigma(z_{n_l}(x)), \qquad n_l = 1, \dots, N_l - 1, \\ f(x) &= W_{N_l}\, h_{N_l-1}(x) + b_{N_l}, \end{aligned}$$

where $\sigma(t) = \max\{0, t\}$ is applied elementwise.

For each hidden layer $n_l \in \{1, \dots, N_l - 1\}$, define the diagonal gating matrix:

$$D_{n_l}(x) := \operatorname{diag}\big(\mathbf{1}\{z_{n_l,1}(x) > 0\}, \dots, \mathbf{1}\{z_{n_l,m_{n_l}}(x) > 0\}\big),$$

where m_{n_l} denotes the width of layer n_l. Fixing an activation pattern $\mathcal{P} = (D_1, \dots, D_{N_l-1})$ defines a polyhedral region:

$$\mathcal{R}(\mathcal{P}) := \Big\{ x \in \mathbb{R}^d \mid z_{n_l,i}(x) > 0 \text{ if } (D_{n_l})_{ii} = 1, \text{ and } z_{n_l,i}(x) \le 0 \text{ if } (D_{n_l})_{ii} = 0, \forall\, n_l, i \Big\}.$$

For any input $x \in \mathcal{R}(\mathcal{P})$, the ReLU acts affinely as $h_{n_l}(x) = D_{n_l} z_{n_l}(x)$ for $n_l = 1, \dots, N_l - 1$. Substituting recursively yields the explicit affine form $f(x) = A(\mathcal{P})\, x + b(\mathcal{P})$, with the parameters defined as:

$$A(\mathcal{P}) = W_{N_l} D_{N_l-1} W_{N_l-1} D_{N_l-2} \cdots D_1 W_1, \tag{12}$$

$$b(\mathcal{P}) = b_{N_l} + W_{N_l} D_{N_l-1} \Big(b_{N_l-1} + W_{N_l-1} D_{N_l-2} \big(b_{N_l-2} + \cdots + W_2 D_1 b_1 \big) \Big). \tag{13}$$

This explicit formulation allows the mapping from an observed activation pattern to its exact local affine rule.

B Proofs.

B.1 Proof of Theorem 1

Proof. **Step 1: Existence of a nonempty sample partition at each split.** Consider a node $D_{i,j}$ with sample index set $S_{i,j}$. If all samples in $S_{i,j}$ have the same associated affine function, Algorithm 1 makes this node a leaf. Otherwise, we show that two affine functions in $\mathcal{L}$, together with an operator $\mathrm{op} \in \{\max, \min\}$, can divide $S_{i,j}$ into two nonempty subsets.

Let $\ell_1, \dots, \ell_M$ denote the distinct affine functions in $\mathcal{L}$. Their pairwise comparisons partition Ω into regions on which the ordering of these functions is fixed. Each region defined by strict comparisons is convex and hence connected. Within such a region, continuity of f_c prevents a change from one affine function to another, because two distinct affine functions do not have equal values there. Therefore, f_c coincides with a single affine function throughout each strict ordering region.

Suppose two samples $x^{(k)}$ and $x^{(l)}$ satisfy

$$\operatorname{sgn}\Big(\ell_a(x^{(k)}) - \ell_b(x^{(k)})\Big) = \operatorname{sgn}\Big(\ell_a(x^{(l)}) - \ell_b(x^{(l)})\Big), \qquad 1 \le a < b \le M.$$

If all comparisons are strict, the two samples belong to the same strict ordering region, and Assumption 2 implies $\ell^{(k)} \equiv \ell^{(l)}$. If some comparisons are equalities, choose a common direction that is not parallel to any of the corresponding equality hyperplanes. A sufficiently small displacement in this direction keeps each sample in its neighborhood U_k or U_l, preserves all strict comparison signs, and resolves each tied comparison in the same direction at both points. The displaced points thus lie in the same strict ordering region. Applying Assumption 2 again gives $\ell^{(k)} \equiv \ell^{(l)}$.

Consequently, two samples associated with different affine functions must differ in at least one pairwise comparison. For that pair, one of the min/max routing rules separates the two samples, including when one of the comparison values is zero. Hence the resulting child sample sets satisfy

$$S_{i+1,2j-1} \neq \emptyset, \qquad S_{i+1,2j} \neq \emptyset.$$

At an internal node, Algorithm 1 first searches for such a split while keeping the inherited affine function and satisfying $C_{i,j} \subseteq K_{i,j}$. If this search fails, it searches all pairs in $\mathcal{L}$ without these two restrictions. The preceding argument guarantees that this latter search succeeds. The same argument applies to the root, where both affine functions are selectable from the outset.

Step 2: Finite termination of splitting. By Step 1, every nonterminal node is split into two nonempty child sample sets. We prove by induction on the number n of samples at a node that its subtree requires at most $n-1$ splits.

For $n = 1$, the node is a leaf and no split is required. Assume the claim holds for all smaller sample sets. A node with n samples either is already a leaf or is split into child sets of sizes $n_1, n_2 \geq 1$, where $n_1 + n_2 = n$ and $n_1, n_2 < n$. By the induction hypothesis, the two child subtrees require at most $n_1 - 1$ and $n_2 - 1$ splits, respectively. Including the current split, the total is at most

$$1 + (n_1 - 1) + (n_2 - 1) = n - 1.$$

Applying this result at the root shows that Algorithm 1 terminates after at most $N - 1$ splits and produces at most N leaves. This conclusion does not require every split to enlarge the covered sample set.

Step 3: Exact interpolation and leaf-affine coverage. At termination, every sample index k reaches a leaf $t(k) \in \mathcal{T}_L$. According to the terminal rule, all samples in that leaf have associated affine functions identical to its output $\ell_{m_{t(k)}}$. Therefore,

$$T_{\mathrm{HH}}(x^{(k)}) = \ell_{m_{t(k)}}(x^{(k)}) = \ell^{(k)}(x^{(k)}) = y^{(k)}, \qquad k = 1, \ldots, N.$$

Hence the constructed tree exactly interpolates all samples. Different activation patterns in the same leaf do not affect this conclusion, because their associated affine functions have identical coefficients.

Moreover, every leaf output is selected from $\mathcal{L}$. Conversely, by the definition of $\mathcal{L}$, each $\ell \in \mathcal{L}$ is associated with at least one construction sample $x^{(k)}$. The leaf reached by that sample outputs the same affine function ℓ. Thus

$$\mathcal{L} \subseteq \bigcup_{t \in \mathcal{T}_L} \{\ell_{m_t}\}.$$

The proof is complete. □

B.2 Proof of Theorem 2

Proof. For every leaf node D_t, $t \in \mathcal{T}_L$, Algorithm 1 terminates the splitting process when all assigned samples have the same associated affine function. Thus, although their activation patterns may differ,

$$\ell^{(k)} \equiv \ell_{m_t}, \qquad \forall k \in S_t.$$

Consequently, every sampled pattern in this leaf belongs to $\mathfrak{P}_t$, i.e.,

$$\mathcal{P}^{(k)} \in \mathfrak{P}_t, \qquad \forall k \in S_t.$$

The set $\mathfrak{P}_t$ may also include feasible patterns not represented by the construction samples.

For the constructed HHT, the output at leaf D_t is its stored affine function, and therefore

$$f_{\mathrm{HHT}}(x) = \ell_{m_t}(x), \qquad \forall x \in D_t.$$

On the other hand, consider any activation pattern $\mathcal{P} \in \mathfrak{P}_t$. Fixing this pattern gives the affine representation of the original ReLU neural network on $R(\mathcal{P})$. By the definition of $\mathfrak{P}_t$,

$$f_{\mathrm{ReLU}}(x) = A(\mathcal{P})x + b(\mathcal{P}) = \ell_{m_t}(x), \qquad \forall x \in R(\mathcal{P}).$$

Combining these two equalities yields

$$f_{\mathrm{HHT}}(x) = f_{\mathrm{ReLU}}(x), \qquad \forall x \in D_t \cap R(\mathcal{P}).$$

Since $\mathcal{P} \in \mathfrak{P}_t$ is arbitrary, this equality holds throughout

$$\bigcup_{\mathcal{P} \in \mathfrak{P}_t} \big(D_t \cap R(\mathcal{P})\big) = D_t \cap \mathcal{R}_t,$$

which proves equation 3. Moreover, each assigned sample satisfies $x^{(k)} \in D_t \cap R(\mathcal{P}^{(k)}) \subseteq D_t \cap \mathcal{R}_t$, so this intersection is nonempty.

If equation 4 holds for every leaf, the preceding equality is valid throughout each D_t. Since the leaf cells cover the domain, i.e.,

$$\bigcup_{t \in \mathcal{T}_L} D_t = \Omega,$$

we obtain $f_{\mathrm{HHT}}(x) = f_{\mathrm{ReLU}}(x)$ for every $x \in \Omega$, which proves equation 5. The proof is complete. □

The definition of $\mathfrak{P}_t$ does not assume that all its patterns have been sampled. Any verified subset $\mathcal{Q}_t \subseteq \mathfrak{P}_t$ also provides a sufficient certificate if $D_t \subseteq \bigcup_{\mathcal{P} \in \mathcal{Q}_t} R(\mathcal{P})$. In particular, containment in the region of one representative pattern $\mathcal{P}_t^{\mathrm{ref}} \in \mathfrak{P}_t$ remains sufficient and can be checked using the original LPs. However, failure of that single-region test does not establish failure of (4) or an output discrepancy. The original single-region LPs are not a complete test for containment in a union of activation regions.

B.3 Proof of Lemma 1

Proof. We count arithmetic operations with dense affine coefficients and condition on the supplied affine pool and sample labels. Throughout, $T_{\mathrm{LP}}(p, q)$ denotes a monotone upper bound on the cost of forming and solving an LP with p variables and at most q inequalities.

(1) Tree construction. Fix a processed node v. For each candidate hinge, evaluating its two affine functions on S_v costs $O(d|S_v|)$. The resulting comparisons determine the child sample sets and the covered samples. Checking candidate admissibility and computing its coverage score require an additional $O(|S_v|)$ operations. Thus, each candidate can be evaluated in $O(d|S_v|)$ time.

At a non-root node, the ordinary search fixes the inherited affine function and considers at most $L - 1$ alternatives, together with a constant number of hinge orientations. It therefore examines $O(L)$ candidates. The root search and the all-pairs fallback examine $O(L^2)$ candidates. Consequently, the cost at any processed node is bounded by $O(L^2 d|S_v|)$. Leaf detection, sample partitioning, and node creation are absorbed by this bound. Summing over all processed nodes gives

$$T_{\mathrm{build}} = O\left(L^2 d \sum_{v \in \mathcal{V}_{\mathrm{proc}}} |S_v|\right) = O(L^2 d V_N).$$

If the all-pairs fallback is never invoked, every non-root node requires at most $O(Ld|S_v|)$ work. Since the root contains all N samples and still uses an all-pairs search, the corresponding bound is

$$T_{\mathrm{build}} = O\big(L^2 dN + Ld(V_N - N)\big) = O(L^2 dN + LdV_N).$$

If the root search also examines only $O(L)$ candidates, this reduces to $O(LdV_N)$.

To establish the balanced-tree bound, let D_k denote the depth of the leaf reached by sample $x^{(k)}$, with the root at depth zero. Each sample occurs in exactly the nodes on its root-to-leaf path. Double counting sample–node incidences therefore yields

$$V_N = \sum_{k=1}^{N} (D_k + 1).$$

For roughly balanced splits, meaning that every child contains at most a fixed fraction $\rho < 1$ of its parent's samples, $D_k = O(\log N)$ for $N \geq 2$. Hence $V_N = O(N \log N)$, and

$$T_{\mathrm{build}} = O(L^2 N d \log N).$$

(2) Worst-case depth. By Theorem 1, the fixed-sample construction terminates. Consider any root-to-leaf path $v_0, v_1, \ldots, v_D$. Every split partitions the parent sample set into two nonempty subsets. Therefore, the child followed along this path satisfies

$$1 \leq |S_{v_{i+1}}| \leq |S_{v_i}| - 1.$$

Since $|S_{v_0}| = N$, induction gives

$$1 \le |S_{v_D}| \le N - D,$$

and hence $D \le N - 1$. Taking the maximum over all paths proves $D_{\max} \le N - 1$.

For the conditional affine-label bound, let

$$Q_v := \{q_k : k \in S_v\},$$

where q_k is the index of the affine function associated with sample k in the deduplicated pool. Under the additional hypothesis,

$$|Q_{v_{i+1}}| \le |Q_{v_i}| - 1$$

at every split along every root-to-leaf path. Because $|Q_{v_0}| = L$ and every leaf contains at least one sample,

$$1 \le |Q_{v_D}| \le L - D.$$

Thus $D_{\max} \le L - 1$ under this additional condition.

(3) Leaf verification and resampling. Fix a leaf t. For each $\mathcal{P} \in \mathfrak{P}_t$, denote the actual numbers of region inequalities before and after pruning by $r_{t,\mathcal{P}}$ and $\widetilde{r}_{t,\mathcal{P}}$, respectively. By assumption,

$$r_{t,\mathcal{P}} \le r_t, \qquad \widetilde{r}_{t,\mathcal{P}} \le \widetilde{r}_t.$$

Sequential redundancy pruning tests each original inequality at most once. Each test solves an LP in d variables with at most $r_{t,\mathcal{P}} - 1$ remaining inequalities. Thus the total pruning cost at leaf t is

$$O\left(\sum_{\mathcal{P}\in\mathfrak{P}_t} r_{t,\mathcal{P}} T_{\mathrm{LP}}(d, r_{t,\mathcal{P}})\right) = O(k_t r_t T_{\mathrm{LP}}(d, r_t)).$$

If a pruned region has no inequalities, it equals $\mathbb{R}^d$ and immediately certifies containment. Otherwise, all reduced row counts are positive.

For the union-containment test, Algorithm 2 selects one inequality from each compatible region. There are

$$K_t := \prod_{\mathcal{P}\in\mathfrak{P}_t} \widetilde{r}_{t,\mathcal{P}} \le \widetilde{r}_t^{k_t}$$

such selections. For each selection, the common-margin LP has variables $(x, \delta) \in \mathbb{R}^{d+1}$ and $n_t + k_t + 2$ inequalities: n_t leaf constraints, k_t selected violation constraints, and the two bounds $0 \le \delta \le 1$. At most K_t such LPs are formed and solved, so this stage costs

$$O\Big(\widetilde{r}_t^{k_t} T_{\mathrm{LP}}(d+1, n_t + k_t + 2)\Big).$$

Termination upon finding a positive-margin witness can only decrease this cost.

If a witness $x^\star$ is found, let $u = x^\star - x^{\mathrm{in}}$. Substituting the segment $x^{\mathrm{in}} + \lambda u$ into a reduced region inequality $c^\top x \le \beta$ gives

$$(c^\top u)\lambda \le \beta - c^\top x^{\mathrm{in}}.$$

Computing these scalar coefficients costs $O(d)$ per inequality; updating the bounds on λ then costs $O(1)$. Consequently, all segment–region intersection intervals can be computed in

$$O\left(d \sum_{\mathcal{P}\in\mathfrak{P}_t} \widetilde{r}_{t,\mathcal{P}}\right) = O(d k_t \widetilde{r}_t)$$

time. The last covered parameter is obtained by scanning the upper endpoints of the nonempty intervals, requiring $O(k_t)$ additional operations, which are absorbed by the preceding bound. Constructing the $M + 1$ resampled points costs $O(d(M + 1))$. Hence the resampling work is bounded by

$$O(d(k_t \widetilde{r}_t + M + 1)).$$

This work is incurred only when a witness is found; including it for every leaf remains a valid upper bound.

Adding these costs and summing over all leaves gives

$$T_{\text{ver/res}} = O\Bigg(\sum_{t \in \mathcal{T}_L} \Big[k_t r_t T_{\text{LP}}(d, r_t) + \widetilde{r}_t^{\,k_t} T_{\text{LP}}(d+1, n_t + k_t + 2) + d(k_t \widetilde{r}_t + M + 1) \Big] \Bigg),$$

which is equation 10. The bound concerns one pass with the compatible-region representations supplied as input; their construction, network queries, and subsequent tree reconstruction are counted separately. □

C Details of Algorithms

C.1 HHT construction

Affine labels and sample coverage. Let $\mathcal{L} = \{\ell_1, \ldots, \ell_L\}$ be the deduplicated pool of sampled affine functions. For each sample, retain its activation pattern $\mathcal{P}^{(k)}$ and affine index q_k, where

$$A(\mathcal{P}^{(k)})x + b(\mathcal{P}^{(k)}) \equiv \ell_{q_k}(x).$$

Different activation patterns may have the same affine index. Write D_v and S_v for the domain and sample index set of node v, respectively.

For a candidate hinge $h = \text{op}\{\ell_a, \ell_b\}$, with $a \neq b$ and $\text{op} \in \{\min, \max\}$, define its routing rule by

$$r_h(x) = \begin{cases} a, & \text{op}\{\ell_a(x), \ell_b(x)\} = \ell_a(x), \\ b, & \text{otherwise.} \end{cases}$$

Thus equality is assigned to the first operand. The routed sample sets and candidate coverage are

$$\begin{aligned} S_a(h) &= \{k \in S_v : r_h(x^{(k)}) = a\}, \\ S_b(h) &= S_v \setminus S_a(h), \\ K_v(h) &= \{k \in S_v : r_h(x^{(k)}) = q_k\}. \end{aligned}$$

Coverage therefore identifies the selected affine function, rather than only its value at an individual sample.

Let $\mathcal{H}(S_v)$ contain all candidate hinges from $\mathcal{L}$ whose two routed sample sets are nonempty. At a non-root node, let a_v be the inherited affine index and C_v the provisional coverage inherited from its parent. The ordinary search fixes the first operand to ℓ_{a_v} and requires $C_v \subseteq K_v(h)$, allowing equality. If this search has no candidate, the algorithm searches $\mathcal{H}(S_v)$ without these two restrictions.

Termination and stored patterns. Under the assumptions of Theorem 1, $\mathcal{H}(S_v)$ is nonempty whenever a node contains more than one affine index. Every accepted split has two nonempty sample children, and the construction terminates when the queue is empty. Provisional coverage is not accumulated into a global early-stopping set.

A leaf is finalized when its samples share one affine function; their activation patterns need not coincide. The stored set $\mathfrak{P}_v^{\text{obs}}$ records observed patterns only and is distinguished from the complete compatible set $\mathfrak{P}_t$ in Theorem 2. For local reconstruction, the affected leaf serves as the root of the reconstructed subtree, with its existing path constraints retained.

C.2 Union containment verification and resampling

Polyhedral representations and constraint pruning. For a leaf D_t, use

$$\overline{D_t} = \{x : A_t x \leq b_t\}, \qquad \overline{\mathcal{R}(\mathcal{P})} = \{x : C_{t,\mathcal{P}} x \leq d_{t,\mathcal{P}}\}, \quad \mathcal{P} \in \mathfrak{P}_t.$$

Algorithm 1 Hinging Hyperplanes Tree Construction

Require: Fixed samples $\{(x^{(k)}, y^{(k)}, \mathcal{P}^{(k)}, q_k)\}_{k=1}^{N}$; affine pool $\mathcal{L}$; domain Ω; assumptions of Theorem 1
Ensure: Binary tree T_{HH} with affine leaf outputs
Create root v_0 with $D_{v_0} = \Omega$, $S_{v_0} = \{1, \dots, N\}$, $a_{v_0} = \perp$, and $C_{v_0} = \emptyset$
$Q \leftarrow \{v_0\}$
while $Q \neq \emptyset$ **do**
  Pop v from Q
  if all q_k, $k \in S_v$, equal the same index q **then**
    Make v a leaf with output ℓ_q
    Store $\mathfrak{P}_v^{\mathrm{obs}} = \{\mathcal{P}^{(k)} : k \in S_v\}$
  else
    if $a_v = \perp$ **then**
      $\mathcal{G}_v \leftarrow \mathcal{H}(S_v)$
    else
      $\mathcal{G}_v \leftarrow \{h = \mathrm{op}\{\ell_{a_v}, \ell_b\} \in \mathcal{H}(S_v) : C_v \subseteq K_v(h)\}$
      if $\mathcal{G}_v = \emptyset$ **then**
        $\mathcal{G}_v \leftarrow \mathcal{H}(S_v)$ ▷ all-pairs fallback
      end if
    end if
    Choose $h_v = \mathrm{op}\{\ell_a, \ell_b\} \in \arg\max_{h \in \mathcal{G}_v} |K_v(h)|$
    Store h_v and its routing rule at v
    for all $j \in \{a, b\}$ **do**
      Create child v_j with $D_{v_j} = \{x \in D_v : r_{h_v}(x) = j\}$
      $S_{v_j} \leftarrow S_j(h_v)$; $a_{v_j} \leftarrow j$
      $C_{v_j} \leftarrow K_v(h_v) \cap S_{v_j}$
      Push v_j into Q
    end for
  end if
end while
Return T_{HH}

The compatible family $\mathfrak{P}_t$ is defined in Theorem 2 and may contain patterns not observed during construction.

Redundant inequalities are removed separately from each region. A row i is redundant if

$$\max_{x : C_{t,\mathcal{P},\neg i} x \leq d_{t,\mathcal{P},\neg i}} c_{t,\mathcal{P},i}^{\top} x \leq d_{t,\mathcal{P},i}.$$

Remove such rows sequentially and denote the equivalent reduced representation by $(\widetilde{C}_{t,\mathcal{P}}, \widetilde{d}_{t,\mathcal{P}})$, with $\widetilde{r}_{t,\mathcal{P}}$ rows. A region with no remaining inequalities equals $\mathbb{R}^d$ and immediately certifies containment.

LP formulation. Assume all reduced row counts are positive. For each selection

$$\mathbf{j} = (j_{\mathcal{P}})_{\mathcal{P} \in \mathfrak{P}_t} \in \prod_{\mathcal{P} \in \mathfrak{P}_t} \{1, \dots, \widetilde{r}_{t,\mathcal{P}}\},$$

solve

$$\begin{aligned} \delta_t(\mathbf{j}) := \max_{x,\delta} \ & \delta \\ \text{s.t.} \ & A_t x \leq b_t, \\ & \widetilde{c}_{t,\mathcal{P},j_{\mathcal{P}}}^{\top} x \geq \widetilde{d}_{t,\mathcal{P},j_{\mathcal{P}}} + \delta, \quad \mathcal{P} \in \mathfrak{P}_t, \\ & 0 \leq \delta \leq 1. \end{aligned} \tag{14}$$

A positive optimum gives a point violating one inequality from every compatible region, hence a witness outside their union. Conversely, any point outside the union strictly violates some row from each region. The minimum of these finitely many positive violations, capped at one, gives a positive feasible margin for the corresponding LP.

Algorithm 2 Check Union Containment and Resample for a Leaf

Require: $\overline{D_t} = \{x : A_t x \le b_t\}$; compatible regions $\{\overline{\mathcal{R}(\mathcal{P})} = \{x : C_{t,\mathcal{P}} x \le d_{t,\mathcal{P}}\} : \mathcal{P} \in \mathfrak{P}_t\}$; an existing sample $x^{\text{in}} \in D_t \cap \mathcal{R}_t$; integer $M \ge 0$
Ensure: Either CONTAINED, or a nonempty set $X_{\text{out}} \subseteq D_t \setminus \mathcal{R}_t$

for all $\mathcal{P} \in \mathfrak{P}_t$ **do**
  $(\widetilde{C}_{t,\mathcal{P}}, \widetilde{d}_{t,\mathcal{P}}) \leftarrow$ PRUNEREDUNDANT$(C_{t,\mathcal{P}}, d_{t,\mathcal{P}})$
  Let $\widetilde{r}_{t,\mathcal{P}}$ be its row count
  if $\widetilde{r}_{t,\mathcal{P}} = 0$ **then**
    return CONTAINED
  end if
end for
$x^\star \leftarrow \perp$
for all $\mathbf{j} \in \prod_{\mathcal{P} \in \mathfrak{P}_t} \{1, \dots, \widetilde{r}_{t,\mathcal{P}}\}$ **do**
  Solve LP equation 14
  if the LP is feasible and $\delta_t(\mathbf{j}) > 0$ **then**
    Set $x^\star$ to an optimal input
    break
  end if
end for
if $x^\star = \perp$ **then**
  return CONTAINED
end if
$v \leftarrow x^\star - x^{\text{in}}$
for all $\mathcal{P} \in \mathfrak{P}_t$ **do**
  Compute the interval $I_{\mathcal{P}} = \{\lambda \in [0,1] : \widetilde{C}_{t,\mathcal{P}}(x^{\text{in}} + \lambda v) \le \widetilde{d}_{t,\mathcal{P}}\}$
end for
$\lambda_{\text{last}} \leftarrow \max_{\mathcal{P}: I_{\mathcal{P}} \neq \emptyset} \max I_{\mathcal{P}}$
$X_{\text{out}} \leftarrow \emptyset$
for $j = 1, \dots, M+1$ **do**
  $\lambda_j \leftarrow \lambda_{\text{last}} + \dfrac{j}{M+2}(1 - \lambda_{\text{last}})$
  $X_{\text{out}} \leftarrow X_{\text{out}} \cup \{x^{\text{in}} + \lambda_j v\}$
end for
return X_{out}

Consequently, containment holds if every LP is infeasible or has optimum zero. The bound $\delta \le 1$ keeps the objective bounded without changing this criterion. The tests below are stated in exact arithmetic.

Validity of the generated samples. Each $I_{\mathcal{P}}$ is a closed interval or is empty. Their union contains zero and excludes one, so $\lambda_{\text{last}} < 1$. The selected parameters are strictly between λ_{last} and one and therefore lie outside the entire compatible-region union. They also satisfy the actual leaf's routing constraints: strict inequalities hold at x^{in} and hold weakly at $x^\star$. Hence $X_{\text{out}} \subseteq D_t \setminus \mathcal{R}_t$, even when the LP witness lies on the leaf-closure boundary.

Using observed compatible patterns. Algorithm 2 takes the complete compatible-region family as input. If only a verified subset $\mathfrak{Q}_t \subseteq \mathfrak{P}_t$ is available, the same procedure tests containment in that subset's region union. A successful test still certifies the leaf, whereas a returned witness establishes non-containment only for that smaller union. If a newly queried pattern induces the current leaf's affine function, add its region to the verification union and repeat the test; discovering that pattern alone does not require splitting the leaf.

Theorem 1 applies to each fixed construction batch satisfying its assumptions. Its finite-construction bound does not by itself establish finite termination of the entire adaptive resampling procedure.

D DETAILED PARAMETERS OF RELU NETWORK AND HHT OF THE TOY EXAMPLE

D.1 DETAILED NETWORK PARAMETERS

The architecture of the target ReLU network is illustrated in Figure 2. It consists of two hidden layers with two and three neurons, respectively. The detailed parameters for this network are shown in Table 3.

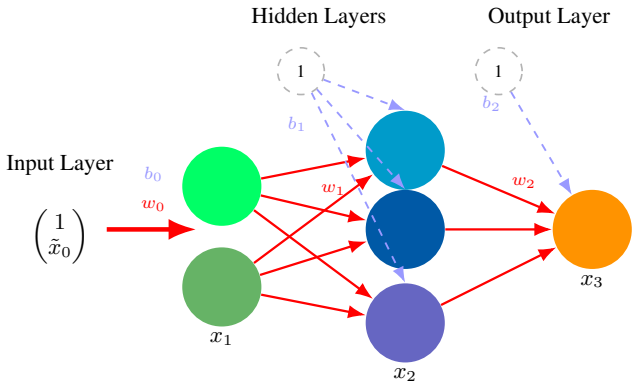


Figure 2: The target network: A ReLU neural network with two hidden layers of size [2,3] (Monke et al., 2025). Biases are depicted as neurons with dashed lines and appended as a 1 to the input.

Table 3: Target Network Parameters

Layer	Parameter	Shape	Value (Matrix/Vector)
0	Weights (W)	$(2, 2)$	$\begin{bmatrix} 0.3984 & -0.5628 \\ 0.5888 & 0.0421 \end{bmatrix}$
	Biases (b)	(2)	$\begin{bmatrix} -0.7855 & 0.5008 \end{bmatrix}$
1	Weights (W)	$(2, 3)$	$\begin{bmatrix} -0.0979 & 0.6198 & 1.8368 \\ 0.2684 & -1.0745 & -0.6810 \end{bmatrix}$
	Biases (b)	(3)	$\begin{bmatrix} 0.5084 & -0.7284 & 0.1318 \end{bmatrix}$
2	Weights (W)	$(3, 1)$	$\begin{bmatrix} 0.3064 \\ -0.9392 \\ 0.1382 \end{bmatrix}$
	Biases (b)	(1)	$\begin{bmatrix} 0.2781 \end{bmatrix}$

D.2 CONSTRUCTED HHT

Figure 3 illustrates the constructed HHT corresponding to the target ReLU network, using the parameters listed in Table 3. The specific parameters for affine functions $\ell_1, \ldots, \ell_6$, along with the corresponding sample counts, are listed in Table 4.

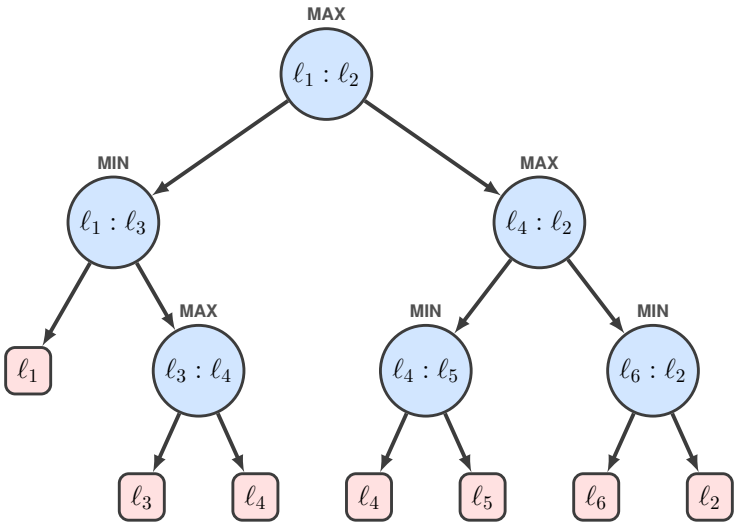


Figure 3: Visualization of the HHT structure corresponding to the target ReLU network shown in Table 3. Blue circles represent split nodes while red rectangles represent leaf regions.

Table 4: Linear function parameters ($y = w_1 x_1 + w_2 x_2 + b$) and sample statistics for each leaf region in the HHT. Note that ℓ_4 appears in two distinct leaf nodes.

Region	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{b}$	#Samples
ℓ_1	-0.0463	0.0035	0.4751	7
ℓ_2	0.0067	-0.0005	0.4462	6
ℓ_3	-0.0582	-0.0142	0.4986	2
ℓ_4	0.0959	0.1313	0.2703	1
ℓ_4	0.0959	0.1313	0.2703	2
ℓ_5	0.0892	0.1318	0.2763	1
ℓ_6	0.0000	0.0000	0.4521	1

E Detailed Experimental Setup for the Width-Scaling Evaluation

To ensure full reproducibility of the quantitative scalability evaluation (Section 5.1), we detail the synthetic data generation, network initialization, and hyperparameters used for the HHT construction.

Data Generation and Network Initialization. For the synthetic scalability benchmarks, the input space is strictly defined as a two-dimensional continuous domain bounded by $[0, 1]^2$. We sample inputs from a uniform distribution over this domain. For each random seed, we generate an initial construction set of **5000** points and a fixed, held-out evaluation set of **1000** points.

The target multi-layer perceptrons (MLPs) are initialized using standard normal distributions scaled to preserve variance across layers. Specifically, the weight matrix W of a layer with n_{in} input features is initialized using He initialization, drawn from $\mathcal{N}(0, 2/n_{\text{in}})$. The bias vectors b are drawn from a normal distribution with a scaled standard deviation, specifically $\mathcal{N}(0, 0.25)$. This initialization prevents the vanishing or exploding of pre-activation values and guarantees a geometrically diverse set of linear regions across the domain.

HHT Construction Hyperparameters. Exact certification and refinement are governed by the LP containment test in Algorithm 2. The following implementation parameters are used for numerical evaluation and candidate selection:

- **Empirical Fidelity Tolerance (`ERR_TOL`):** We use **1e-15** only when reporting numerical agreement between the target network and the HHT on diagnostic evaluation points. This tolerance does not certify a leaf and does not determine whether refinement continues.
- **Diagnostic Sampling (`ERR_CHECK_N`):** After the LP-based certification/refinement procedure, **8000** additional uniformly sampled points are evaluated to estimate the empirical maximum prediction error. These diagnostic samples do not affect the refinement control flow.
- **Boundary Search Candidates (`SEARCH_TOP_K`):** When a leaf requires refinement, the implementation examines the top **200** candidate hyperplanes ranked by the prescribed search heuristic.
- **Tie-breaking Epsilon (`tie_eps`):** A tolerance of **1e-10** is used for numerical tie handling at decision boundaries.

F Local Traceability Case Studies

We emphasize that HHT is not intended to make a large tree globally readable by manual inspection. Instead, its structural transparency is manifested through *exact local traceability* and *auditability*. For any given input, HHT completely exposes the root-to-leaf sequence of hinge comparisons, the active affine rule at the reached leaf, and the precise margins to the local switching hyperplanes.

Toy boundary case study. We first illustrate local traceability using the two-dimensional toy network whose parameters are listed in Table 3. The corresponding HHT is shown in Figure 3, and the affine functions stored at the leaves are listed in Table 4. Note that Table 4 reports rounded affine parameters for readability, while this case study uses the full-precision parameters from the implementation.

Consider the borderline input

$$x = (0.604062,\ 0.776161).$$

Both the target ReLU network and the extracted HHT output

$$f_{\text{ReLU}}(x) = f_{\text{HHT}}(x) = 0.44980799.$$

As shown by the HHT in Figure 3, this input follows the path

$$\max(\ell_1, \ell_2) \ \longrightarrow\ \min(\ell_1, \ell_3) \ \longrightarrow\ \ell_1.$$

The final prediction is therefore governed by the local affine rule

$$\ell_1(x) = -0.046291\, x_1 + 0.003464\, x_2 + 0.475082.$$

For consistency with Table 4, the rounded version is

$$\ell_1(x) \approx -0.0463\, x_1 + 0.0035\, x_2 + 0.4751.$$

Along the root-to-leaf path, the smallest margin occurs at the root hinge comparing ℓ_1 and ℓ_2:

$$|\ell_1(x) - \ell_2(x)| = 6.512 \times 10^{-8}.$$

Thus, the root comparison $\ell_1 : \ell_2$ identifies the decisive switching boundary for this input. The corresponding switching hyperplane is

$$(\, w_{\ell_1} - w_{\ell_2}\,)^\top x + (b_{\ell_1} - b_{\ell_2}) = 0,$$

with

$$w_{\ell_1} - w_{\ell_2} = (-0.052969,\ 0.003964), \qquad b_{\ell_1} - b_{\ell_2} = 0.028920.$$

To verify that this boundary is locally meaningful, we perturb the input across the identified hyperplane. The boundary point is approximately

$$x_{\mathrm{bd}} = (0.604063,\ 0.776161),$$

and the crossed point is

$$x_{\mathrm{cross}} = (0.624007,\ 0.774668).$$

After crossing, the prediction path changes to the right branch of the root:

$$\max(\ell_1, \ell_2) \ \longrightarrow\ \max(\ell_4, \ell_2) \ \longrightarrow\ \min(\ell_6, \ell_2) \ \longrightarrow\ \ell_2.$$

Both models remain numerically identical after crossing:

$$f_{\mathrm{ReLU}}(x_{\mathrm{cross}}) = f_{\mathrm{HHT}}(x_{\mathrm{cross}}) = 0.44994187.$$

The active linear region changes from ℓ_1 to ℓ_2, and the HHT prediction changes by

$$1.339 \times 10^{-4}.$$

This example shows how HHT makes the decisive local hinge, the active affine rule, and the neighboring region transition explicit.

Breast Cancer case study. We further examine local traceability on the Breast Cancer dataset using a small 3×3 ReLU network. This smaller architecture is used only for visualization of the complete HHT structure; the purpose of this case study is to illustrate how a practitioner can trace and debug one prediction.

The complete HHT structure and the traced path for this sample are visualized in Figure 4.

For a held-out test sample, the original prediction is malignant. The target ReLU network produces logit 0.263681, corresponding to probability 0.565541, and the extracted HHT produces the same logit up to numerical precision:

$$|f_{\mathrm{ReLU}}(x) - f_{\mathrm{HHT}}(x)| = 4.996 \times 10^{-16}.$$

The HHT path has length 5. Along this path, the smallest margin occurs at Node 3:

$$\mathrm{margin}_{\min} = 1.555 \times 10^{-2},$$

while the second-smallest margin is 3.921×10^{-2} at Node 0. Thus, Node 3 is identified as the decisive local switching boundary for this sample.

The active local affine rule for the original sample is

$$\begin{aligned} f_{\mathrm{HHT}}(x) = {} & 0.358\, x_{\mathrm{clump}} - 0.212\, x_{\mathrm{cell_size}} - 0.175\, x_{\mathrm{cell_shape}} + 0.291\, x_{\mathrm{adhesion}} + 0.231\, x_{\mathrm{epithelial}} \\ & + 0.396\, x_{\mathrm{bare}} - 0.317\, x_{\mathrm{chromatin}} + 0.636\, x_{\mathrm{nucleoli}} + 0.842\, x_{\mathrm{mitoses}} - 0.958. \end{aligned}$$

Since the input features are standardized, the affine coefficients and their products with standardized feature values provide local attribution for this specific prediction. The largest positive contributions are from normal nucleoli, bare nuclei, and marginal adhesion:

$$1.088, \qquad 0.709, \qquad 0.503,$$

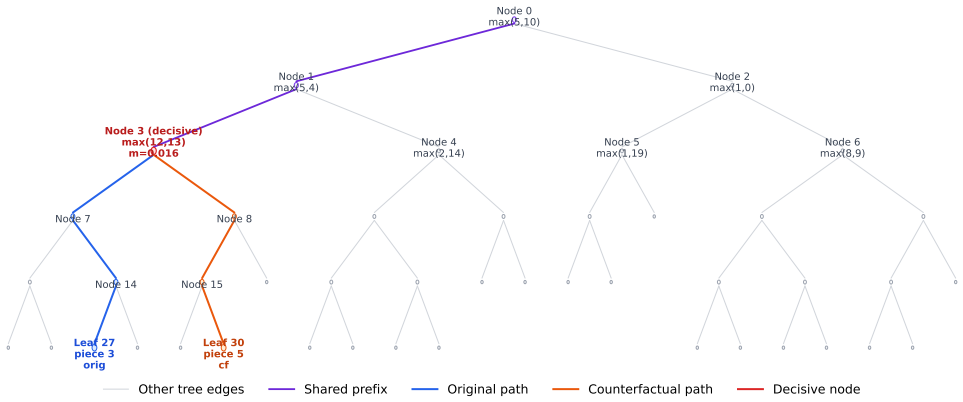


Figure 4: Local traceability case on the Breast Cancer dataset using a 3×3 ReLU network. The highlighted path corresponds to the selected held-out sample. The smallest path margin occurs at Node 3, identifying the decisive local switching boundary.

respectively. The largest negative contribution is from bland chromatin, with contribution -0.443.

To test the identified boundary, we perturb the sample across the smallest-margin hinge. The crossed point reaches a different leaf and a different affine piece, changing from leaf 27 / affine piece 3 to leaf 30 / affine piece 5. The target network logit changes from 0.263681 to -0.022321, and the predicted class changes from malignant to benign. The HHT remains numerically identical to the target network after crossing:

$$|f_{\mathrm{ReLU}}(x_{\mathrm{cross}}) - f_{\mathrm{HHT}}(x_{\mathrm{cross}})| = 1.665 \times 10^{-16}.$$

The most changed raw features are normal nucleoli $8.000 \to 7.441$, bare nuclei $10.000 \to 9.689$, and mitoses $1.000 \to 0.797$. This example illustrates that HHT exposes not only the prediction value, but also the local affine rule, the decisive hinge, and a concrete neighboring-region transition.

G Exact Polyhedral Queries on the Compiled Representation

A certified HHT provides an explicit piecewise-affine representation of the source ReLU network. Once constructed, this representation can be reused for optimization over constrained subsets of the input domain. We illustrate this capability through exact output-range queries.

Exact queries from certified leaves. Let

$$\mathcal{Q} = \{x : Gx \le h\} \subseteq \Omega$$

be a polyhedral query set. The certified HHT partitions Ω into leaf polytopes $\{\mathbb{D}_t\}$, with an affine output rule $\ell_t(x)$ on each leaf. By Theorem 2, $\ell_t(x)$ equals the source-network output on $\mathbb{D}_t$. Since the nonempty intersections $\mathbb{D}_t \cap \mathcal{Q}$ cover $\mathcal{Q}$, the exact extrema over the query set are

$$\underline{f}(\mathcal{Q}) = \min_{t:\mathbb{D}_t \cap \mathcal{Q} \neq \emptyset} \; \min_{x \in \mathbb{D}_t \cap \mathcal{Q}} \ell_t(x),$$

and

$$\overline{f}(\mathcal{Q}) = \max_{t:\mathbb{D}_t \cap \mathcal{Q} \neq \emptyset} \; \max_{x \in \mathbb{D}_t \cap \mathcal{Q}} \ell_t(x).$$

Each inner problem is a linear program because both the leaf and query constraints are linear and ℓ_t is affine.

Experimental setup. We evaluate these post-compilation queries on two-dimensional random ReLU networks over $\Omega = [0, 1]^2$, with two hidden layers of widths 10×10, 50×50, and 100×100. We use the seed-1 networks from the synthetic setup. For local queries, 50 points are sampled from $[0.1, 0.9]^2$, with ℓ_∞ neighborhoods of radius $\epsilon \in \{0.01, 0.05, 0.1\}$. HHT computes the exact range by intersecting the query with its certified leaf polytopes and solving the corresponding affine LPs. For reference, we also report sound bounds obtained from the source network using CROWN-IBP (Zhang et al., 2020), as implemented in `auto_LiRPA` (Xu et al., 2020).

Table 5: Output-range queries for ℓ_∞ neighborhoods with $\epsilon = 0.1$. Widths are averaged over 50 query points. The ratio is the CROWN-IBP width divided by the exact HHT width. Query times exclude HHT construction.

Network	HHT width	CROWN-IBP width	Ratio	HHT (ms)	CROWN-IBP (ms)
10×10	0.0542	0.0877	1.62×	6.67	2.51
50×50	0.1231	0.4408	3.76×	58.41	2.57
100×100	0.1365	1.4158	11.76×	272.13	2.61

Table 6: Output ranges for general polyhedral queries on the 10×10 network. HHT optimizes directly over the specified polytope; CROWN-IBP is evaluated over the enclosing ℓ_∞ region described in the text.

Query	HHT exact range	CROWN-IBP reference range
C1	[0.434, 0.607]	[0.258, 0.738]
C2	[0.434, 0.538]	[0.270, 0.746]
C3	[0.432, 0.538]	[0.319, 0.670]
C4	[0.453, 0.538]	[0.325, 0.633]
C5	[0.432, 0.567]	[0.318, 0.667]
C6	[0.490, 0.543]	[0.160, 0.736]

Local ℓ_∞ queries. Table 5 reports results for $\epsilon = 0.1$. The HHT range is exact over each query region, while CROWN-IBP provides a sound bound. The difference in range width increases with network size in this experiment.

The corresponding one-time HHT construction times are approximately 0.9, 44.5, and 475.5 seconds for the three networks. Thus, the range-query timings in Table 5 should be interpreted as the cost of reusing an already compiled HHT. CROWN-IBP provides inexpensive bounds directly from the source network, whereas the compiled HHT provides exact ranges through its explicit affine regions.

General polyhedral queries. The same representation also supports query sets with general linear constraints. For $\mathcal{Q} = \{x : Gx \le h\}$, the inequalities $Gx \le h$ are appended directly to the constraints defining each HHT leaf. We illustrate this using six query sets on the 10×10 network:

$$
\begin{aligned}
&\text{C1}: \quad x_1 + x_2 \le 0.8, \\
&\text{C2}: \quad x_1 + x_2 \le 0.8, \qquad x_1 - x_2 \le 0.2, \\
&\text{C3}: \quad 2x_1 - x_2 \le 0.4, \qquad x_2 \le 0.6, \\
&\text{C4}: \quad x_1 + 2x_2 \le 1.0, \qquad x_1 - x_2 \le 0.3, \qquad x_1 \le 0.5, \\
&\text{C5}: \quad x_1 \le 0.6, \qquad x_2 \le 0.6, \qquad x_1 + x_2 \ge 0.4, \\
&\text{C6}: \quad 0.2 \le x_1 - x_2 \le 0.3.
\end{aligned}
$$

Each query is additionally intersected with $\Omega = [0, 1]^2$.

For comparison, we construct an enclosing ℓ_∞ region for each polytope by first computing its coordinate-wise bounding box and then using an ℓ_∞ ball centered at the box center with radius equal to its largest half-width. CROWN-IBP is evaluated on this enclosing region. This provides a reference for the effect of retaining the original query geometry explicitly.

The narrow-band query C6 particularly illustrates the role of query geometry: its enclosing ℓ_∞ region contains inputs outside the original polytope. With HHT, the original linear constraints are instead combined directly with the certified leaf constraints. More generally, the same compiled representation can be reused for different polyhedral queries by changing only the additional linear constraints.

Global range queries. Setting $\mathcal{Q} = \Omega$ gives the exact global output range over the prescribed domain. Table 7 reports these results for the three networks. We additionally evaluate each network on a dense 1001×1001 grid as a numerical sanity check; the observed grid extrema agree with the HHT extrema to the reported precision.

Table 7: Global output-range queries over $\Omega = [0, 1]^2$. The dense-grid evaluation is included as a numerical sanity check.

Network	HHT exact range	Dense-grid range	CROWN-IBP width
10×10	$[0.417, 0.645]$	$[0.417, 0.645]$	0.633
50×50	$[-0.851, -0.353]$	$[-0.851, -0.353]$	6.676
100×100	$[-0.284, 0.078]$	$[-0.284, 0.078]$	18.044

Together, these experiments show how the certified HHT can be reused beyond pointwise evaluation. The same explicit collection of leaf polytopes and affine output rules supports exact optimization over local neighborhoods, general linearly constrained regions, and the full prescribed domain. This reusability is a direct consequence of compiling the implicit piecewise-affine structure of the source network into an explicit polyhedral representation.

H Effect of Training Data Size

To clarify whether HHT sampling is determined simply by the raw size of the training set, we train the target network using 20%, 50%, and 100% of the training data, while initializing HHT with 200 samples in all settings. Table 8 reports the number of discovered linear regions, the final number of HHT samples after active resampling, and the empirical maximum fidelity error.

Table 8: Effect of training data fraction on discovered linear regions and final HHT samples. Results are averaged over 5 runs. FidMax is computed on the evaluation set.

Dataset	Metric	20%	50%	100%
Airfoil	Discovered regions	99.4	73.4	64.2
	HHT samples	449.6	482.0	435.2
	FidMax	0.000	0.000	0.000
Abalone	Discovered regions	622.8	742.0	693.6
	HHT samples	6646.8	5253.6	5852.0
	FidMax	0.000	0.000	0.000

The results show that the HHT sampling requirement is not a monotone function of the training data fraction. For example, on Abalone, the number of discovered linear regions changes from 622.8 to 742.0 and then to 693.6, while the final HHT sample count changes from 6646.8 to 5253.6 and then to 5852.0. Similarly, on Airfoil, increasing the training fraction does not lead to a monotone increase in either discovered regions or final HHT samples. These results suggest that HHT sampling is governed by the geometry learned by the target network and by the difficulty of verifying and refining the induced partition, rather than by the raw number of training examples alone. Across all settings, HHT maintains $\mathrm{FidMax} = 0.000$ on the evaluation set.

I Dataset Details

We utilize a diverse set of synthetic and real-world benchmarks. The dataset specifications and preprocessing details correspond strictly to the implementation in our codebase.

Preprocessing and Splitting. For all datasets, we standardized features to have zero mean and unit variance using `StandardScaler`. Missing values (e.g., in the `breast cancer` dataset) were imputed using the mean of the respective feature column. Data splitting was performed using a random stratified split (for classification) or a random split (for regression) with a ratio of 75% for training and 25% for testing.

Synthetic Datasets. The `moons` and `circles` datasets were generated using Scikit-learn with a noise level of $\sigma = 0.2$ and a sample size of $N = 2000$. For `circles`, a scale factor of 0.5 was employed between the inner and outer circles.

Table 9: Dataset Details

Dataset	Features	Sample Size (N)	Source
`moons`	2	2000	Synthetic
`circles`	2	2000	Synthetic
`diabetes`	10	442	Scikit-learn
`abalone`	8	4177	UCI[1]
`breast cancer`	9	699	UCI[1]
`airfoil`	5	1503	UCI[1]

[1] `https://archive.ics.uci.edu/`