# Expressivity Saturation: Reduced Affine Region Usage Under Increasing Task Complexity

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Piecewise-affine neural networks (e.g., with ReLU or LeakyReLU activations) implement continuous piecewise-affine maps, and the number of affine regions provides a natural proxy for expressive capacity. However, the gap between theoretical region capacity and the affine regions realized after training remains insufficiently understood. We study this gap from two complementary perspectives. First, we give a rigorous, architecture-dependent theorem for one-dimensional probes: for multilayer perceptrons with piecewise-affine activations, the number of affine pieces realized along a probe is upper bounded by an explicit product of layer-wise width terms (and activation breakpoint factors). This yields a neuron-threshold lower bound for representing target functions with prescribed one-dimensional piece complexity, formalizing the minimal region budget required for complex signals. Second, we exactly enumerate affine regions realized within bounded 2D and higher-dimensional domains under controlled task complexity. Under fixed architectures and training protocols, increasing input–label complexity yields trained solutions with markedly fewer realized regions in the evaluation domain, even though worst-case architectural capacity is unchanged; we call this reduced region usage expressivity saturation. Moreover, in the most challenging regimes, 2D visualizations show that region-usage collapse often coincides with degraded decision boundaries. Finally, we visualize the training dynamics of affine-region partitions and decision boundaries, revealing a consistent refinement process during optimization.

## 1 Introduction

Neural networks with piecewise-affine activations (e.g., ReLU Nair & Hinton (2010) and LeakyReLU Maas et al. (2013)) define continuous piecewise-affine (CPWA) maps. This viewpoint provides a concrete and interpretable representation of network expressivity: the input space is partitioned into affine regions on which the network acts affinely, and the number and geometry of these affine regions quantify how the network allocates functional complexity Montúfar et al. (2014); Li & Wang (2025). During training the affine-region partition evolves over epochs, with successive layers progressively refining earlier partitions and yielding increasingly intricate decision structures.

A large body of prior work studies the *maximum* number of regions that a given architecture can realize and shows strong depth-dependent gains in worst-case constructions Chen et al. (2023); Hertrich et al. (2023); Averkov et al. (2025). However, these combinatorial capacity results do not directly explain what happens in trained models: in practice, the number of *realized* affine regions is often much smaller than the theoretical maximum, and it depends strongly on initialization, optimization, and data (Tseran & Montúfar, 2021; Milkert et al., 2025). This leaves an important gap between *expressive capacity* and *expressive utilization*.

This gap is especially relevant when the input–label relation is highly complex, i.e., when accurately representing the target may require rapid changes of the decision function at fine input scales. In such regimes, a network may have large worst-case theoretical capacity, yet optimization may still converge to solutions that utilize only a small fraction of that capacity in terms of realized affine partitions. Motivated by this discrepancy, we ask:

*For a fixed architecture and training protocol, how does increasing complexity of the input–label relation affect the number of affine regions that are actually realized within a bounded domain, and how does this relate to the emergence (or failure) of decision boundaries?*

**Our perspective.** We address this question via a combined theoretical and empirical analysis under fixed architectures and training protocols. On the theoretical side, we analyze one-dimensional probes (line segments or curves in input space), where the affine-piece structure of CPWA networks is exactly countable and yields architecture-dependent bounds. On the empirical side, we perform exact bounded-domain enumeration of realized affine regions in 2D and higher dimensions under controlled task complexity, and we use 2D decision-boundary visualizations and training-time snapshots to connect region formation to classification behavior and its optimization dynamics.

This perspective leads to a central phenomenon that we call *expressivity saturation*: under a fixed architecture and training protocol, as the complexity of the input–label relation increases in our controlled setups, the number of *realized* affine regions exhibited by the trained network within a bounded domain can decrease sharply, even though the architecture's worst-case theoretical region capacity is unchanged. In the most extreme regimes we tested, this reduction in realized affine regions coincides with a failure to form effective decision boundaries, indicating a regime where optimization yields both low region usage and poor classification geometry.

Our contributions are as follows:

1. **A rigorous 1D affine-region budget theorem.** We prove a deterministic, architecture-dependent upper bound on the number of affine pieces that a piecewise-affine MLP can realize along any one-dimensional probe. As a corollary, we obtain a neuron-threshold lower bound for representing targets with prescribed one-dimensional piece complexity.

2. **Empirical evidence of expressivity saturation under increasing task complexity.** By enumerating realized affine regions in 2D and higher-dimensional settings, we empirically demonstrate that, under a fixed architecture and training protocol, increasing task complexity in our controlled setups can sharply reduce the number of realized regions within a bounded domain, despite unchanged worst-case theoretical capacity.

3. **Geometric and dynamical evidence linking region collapse to decision-boundary failure.** Our 2D visualizations show that, in the most challenging regimes we tested, an abrupt reduction in realized affine regions often coincides with the failure to form effective decision boundaries, connecting region-usage collapse to poor boundary geometry in practice. Tracking affine partitions and decision boundaries throughout training further reveals a refinement process.

## 2 Related Work

### 2.1 Counting Affine Regions

Understanding the geometric complexity of piecewise-affine neural networks (PANNs) has attracted significant attention. Early efforts include Tseran & Montúfar (2021), who proposed an algorithm to compute the number of affine regions in maxout networks. Berzins (2023) introduced a polyhedral extraction method that operates on subdivided edges rather than explicitly enumerating full regions. Humayun et al. (2023) proposed SplineCam, a technique grounded in PANN theory for high-fidelity computation of network geometry. In the context of graph neural networks, Chen et al. (2023) derived tight upper and lower bounds on the number of affine regions. For convolutional ReLU networks, Xiong et al. (2024) analyzed both the maximal and average number of regions. Leveraging tropical geometry, Piwek et al. (2023) explicitly calculated the number of boundary and affine segments. To mitigate the exponential growth of affine regions with depth, Li & Wang (2025) proposed an accuracy-based approach that reduces this growth to a polynomial function of width.

### 2.2 Evolution of Affine Regions

Beyond counting, the dynamics of affine-region formation have also been studied. Balestriero & Baraniuk (2018) employed a geometric framework to analyze how PANNs hierarchically organize input signals. Cohan et al. (2022) conducted empirical analyses of how the count and density of affine regions evolve in reinforcement learning tasks with continuous control. Grigsby & Lindsey (2022) examined the relationship between ReLU network architecture and the configuration of decision regions. In addition, Balestriero & LeCun (2024) showed that uniform sampling becomes exponentially inefficient for recovering small-volume regions as input dimensionality grows.

### 2.3 Expressive Capacity of Neural Networks

The expressive capacity of PANNs has been investigated from multiple perspectives. The roles of network depth and width have been characterized in (Goujon et al., 2024; Phuong & Lampert, 2020; Hanin et al., 2022; Wang, 2022; Averkov et al., 2025; Hu et al., 2022), while the influence of training parameters on expressiveness has been examined in (Tiwari & Konidaris, 2022; Rolnick & Kording, 2020; Zhang & Wu, 2020; Trimmel et al., 2021; Milkert et al., 2025). Furthermore, extensions of the universal approximation theorem (Anthony & Bartlett, 2002) have been developed in (He et al., 2020; Hertrich et al., 2023; Chen et al., 2022; Haase et al., 2023; Nguyen et al., 2025), providing theoretical foundations for the representational power of PANNs. Complementary to these global expressivity results, recent work on local complexity in ReLU networks characterizes expressive capacity through the density of affine regions over the input distribution and links it to representation learning and optimization dynamics (Patel & Montúfar, 2025).

**Positioning of our contribution.** Our work complements prior studies on worst-case expressivity by linking the complexity of the input–label relation (under controlled setups and fixed training protocols) to the *realized* affine-region usage of a fixed PANN. We derive a line-restricted region upper bound with a corresponding neuron-threshold lower bound, and use *exact* bounded-domain enumeration to show that, in challenging regimes, trained networks may exhibit region-usage collapse together with impaired decision boundaries.

## 3 Preliminaries

### 3.1 Piecewise-Affine Neural Networks and Affine Regions

We consider a fully-connected multilayer perceptron (MLP) equipped with a CPWA activation function. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a scalar CPWA activation function. By definition, there exists a finite set of breakpoints $-\infty = t_0 < t_1 < t_2 < \cdots < t_K < t_{K+1} = \infty$ such that on each open interval $(t_{k-1}, t_k)$ for $k \in \{1, \ldots, K+1\}$, $\sigma$ acts as an affine transformation $\sigma(z) = c_k z + d_k$. For instance, the ReLU activation has a single breakpoint at $t_1 = 0$ with $K = 1$, resulting in two affine pieces.

The PANN $f : \mathbb{R}^d \to \mathbb{R}^m$ of depth $L$ is recursively defined as:

$$\begin{aligned} h_0(x) &= x, \\ h_\ell(x) &= \sigma(W_\ell h_{\ell-1}(x) + b_\ell)\,, \quad \ell = 1, \ldots, L, \\ f(x) &= W_{L+1} h_L(x) + b_{L+1}, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^d$ is the input, $n_\ell$ denotes the width (number of neurons) of the $\ell$-th hidden layer, $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $b_\ell \in \mathbb{R}^{n_\ell}$ are the weight matrix and bias vector for layer $\ell$, and the activation $\sigma$ is applied element-wise.

The piecewise-affine nature of $\sigma$ induces a partition of the input space $\mathbb{R}^d$ into distinct regions based on the activation states of the hidden neurons. Let $\theta$ represent the fixed network parameters.

**Definition 1** (Affine region Montúfar et al. (2014))**.** Fix the network parameters $\theta$. An *affine region* of the network $f(\cdot;\theta)$ is a maximal connected subset $\Omega \subset \mathbb{R}^d$ with non-empty interior on which the mapping $x \mapsto f(x;\theta)$ is strictly affine.

For deep networks with CPWA activations, these affine regions are determined by the intersection of pre-activation conditions across all layers and form polyhedral cells. We leverage a line-restricted formulation to exactly quantify regional complexity.

### 3.2 Line-Restricted Affine Complexity

To rigorously study the realized expressivity, we restrict the network's domain to one-dimensional trajectories. Let $\gamma : [0, 1] \to \mathbb{R}^d$ be a continuous one-dimensional probe (typically a straight line segment or a smooth curve parameterized by $t \in [0, 1]$).

**Definition 2** (Line-restricted region count)**.** The line-restricted affine region count of a network $f$ evaluated along a probe $\gamma$ is defined as the number of distinct affine pieces formed by the composition $f \circ \gamma$. Formally:

$$\alpha(f; \gamma) := \min \left\{ R \in \mathbb{N} \;\middle|\; \begin{array}{l} \exists \text{ a partition } 0 = \tau_0 < \tau_1 < \cdots < \tau_R = 1 \text{ such that} \\ f \circ \gamma \text{ is affine on each open interval } (\tau_{r-1}, \tau_r) \end{array} \right\}. \tag{2}$$

This quantity is strictly exact, mathematically tractable, and directly captures the functional complexity—the number of affine transformations the network dynamically deploys—along data-supported trajectories or interpolations in the input space.

### 3.3 Target Complexity Along a Probe

To mathematically ground our investigation into the minimum network capacity required to express highly irregular signals, we must formally define the intrinsic complexity of a target function restricted to a probe.

**Definition 3** (1D piece complexity)**.** Let $g : [0, 1] \to \mathbb{R}^m$ be a target mapping (e.g., representing an idealized scalar logit function or a vector-valued continuous label distribution) defined along a one-dimensional probe. The one-dimensional piece complexity of $g$ is defined as:

$$Q(g) := \min \left\{ K \in \mathbb{N} \;\middle|\; \begin{array}{l} \exists \text{ a partition of } [0, 1] \text{ into } K \text{ connected intervals} \\ \text{on each of which } g \text{ is strictly affine} \end{array} \right\}. \tag{3}$$

If $g$ cannot be represented as a finite piecewise-affine function, we define $Q(g) = \infty$. For classification tasks defined along a probe, $g$ can be instantiated as the classification margin; consequently, the number of distinct label alternations along the probe provides a strict lower bound on $Q(g)$. This metric acts as our foundational threshold for analyzing network expressivity requirements.

## 4 Theory: A Rigorous Capacity Bound for CPWA Networks on 1D Probes

### 4.1 Architecture-dependent upper bound on line regions

We consider CPWA activation functions with $K$ breakpoints. Each neuron applying such an activation can introduce at most $K$ non-differentiable points (splits) along any one-dimensional affine trajectory.

**Theorem 4** (Line-region upper bound for CPWA MLPs)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}^m$ be a fully-connected neural network with hidden layer widths $(n_1, \ldots, n_L)$, equipped with a CPWA activation function $\sigma$ having $K$ breakpoints (e.g., $K = 1$ for ReLU or LeakyReLU). For any one-dimensional probe $\gamma : [0, 1] \to \mathbb{R}^d$, the line-restricted region count is upper bounded by:*

$$\alpha(f; \gamma) \;\leq\; \prod_{\ell=1}^{L} (K n_\ell + 1). \tag{4}$$

**Remark 5.** This is a deterministic upper bound that depends strictly on the architectural capacity (depth, widths, and activation complexity), independent of initialization or training dynamics. While not necessarily tight in all practical regimes, it is explicit and mathematically sufficient to establish a rigorous neuron-threshold lower bound.

### 4.2 Minimum neuron budget for representing a 1D target

**Corollary 6** (Neuron-threshold lower bound on a fixed-depth CPWA architecture)**.** *Let $g : [0, 1] \to \mathbb{R}^m$ be a target map with $Q(g)$ affine pieces. Suppose a CPWA network $f$ (with $K$-breakpoint activations) exactly realizes $g$ along a probe $\gamma$, such that*

$$f(\gamma(t)) = g(t), \qquad \forall t \in [0, 1]. \tag{5}$$

*Then, the network's architectural capacity must satisfy:*

$$\prod_{\ell=1}^{L}(Kn_\ell + 1) \;\geq\; Q(g). \tag{6}$$

*Consequently, by the AM-GM inequality, if the total number of hidden neurons is $N = \sum_{\ell=1}^{L} n_\ell$, the network must satisfy the lower bound:*

$$N \;\geq\; \frac{L}{K}\big(Q(g)^{1/L} - 1\big). \tag{7}$$

*For architectures with equal-width hidden layers ($n_\ell = w$ for all $\ell$), this simplifies to:*

$$w \;\geq\; \left\lceil \frac{Q(g)^{1/L} - 1}{K} \right\rceil. \tag{8}$$

Corollary 6 formally dictates that to express a sufficiently complex one-dimensional target, the architecture must supply an adequate line-region budget. The complexity burden is mitigated linearly by the activation complexity $K$, but exponentially by the depth $L$.

### 4.3 Practical region-yield heuristic

While the deterministic theorem establishes a strict *capacity* upper bound, practical optimization typically yields significantly fewer regions. Empirically and theoretically, the average number of regions along a probe scales roughly linearly with the total number of neurons under standard initializations for PANNs Hanin & Rolnick (2019). This motivates the definition of practical efficiency metrics.

**Definition 7** (Realized region efficiency on probes)**.** For a probe distribution $\Gamma$ and a trained CPWA network $f$, we define the realized region efficiency relative to the theoretical upper bound as:

$$\eta(f; \Gamma) := \frac{\mathbb{E}_{\gamma\sim\Gamma}[\alpha(f; \gamma)]}{\prod_{\ell=1}^{L}(Kn_\ell + 1)}. \tag{9}$$

Equivalently, it is highly instructive to measure the *regions-per-neuron* statistic:

$$\alpha(f; \Gamma) := \frac{\mathbb{E}_{\gamma\sim\Gamma}[\alpha(f; \gamma)]}{N}. \tag{10}$$

In standard training regimes, $\alpha$ remains within a modest, architecture-dependent constant range. However, our empirical findings demonstrate that $\alpha$ can collapse when the intrinsic complexity of the input-label mapping exceeds the effective capacity of the fixed architecture.

**Practical threshold estimator.** If a target distribution intrinsically requires approximately $Q$ affine pieces along probes sampled from $\Gamma$, we propose a data-driven rule of thumb to estimate the minimum practical neuron budget:

$$N_{\min}^{\text{practical}} \approx \left\lceil \frac{Q}{\alpha(f; \Gamma)} \right\rceil, \tag{11}$$

where $\alpha(f; \Gamma)$ is estimated empirically for the specific architecture family, initialization scheme, and optimization algorithm of interest.

**Algorithm 1** Exact affine-piece counting along a 1D probe for piecewise-affine MLPs

**Require:** Network $f$ with $L$ hidden layers; probe $\gamma : [a, b] \to \mathbb{R}^d$
Initialize partition $\mathcal{P} \leftarrow \{[a, b]\}$
**for** $\ell = 1, \ldots, L$ **do**
  $\mathcal{P}_{\text{new}} \leftarrow \emptyset$
  **for** each interval $I \in \mathcal{P}$ **do**
    Represent $h_{\ell-1}(\gamma(t))$ as an affine function of $t$ on $I$
    Compute all pre-activation roots on $I$:
    $\mathcal{Z} \leftarrow \bigcup_{j=1}^{n_\ell} \{t \in \text{int}(I) : a_{\ell,j}(t) = 0\}$
    Split $I$ at the sorted points in $\mathcal{Z}$ and add the resulting subintervals to $\mathcal{P}_{\text{new}}$
  **end for**
  $\mathcal{P} \leftarrow \mathcal{P}_{\text{new}}$
**end for**
**return** $|\mathcal{P}|$ ▷ Exact number of affine pieces of $f \circ \gamma$

**Proofs.** Proofs of the main theoretical results are provided in Appendix A.

## 5 Counting Methodology

We describe how we *exactly enumerate* the number of affine regions realized by a trained PANN within a bounded input domain. Our methodology separates (i) *exact* one-dimensional (1D) counting along probes, used to support the theoretical analysis, from (ii) *exact bounded-domain enumeration* in two-dimensional (2D) and higher-dimensional settings, where we exhaustively traverse *all* affine regions intersecting the prescribed domain $\Omega$.

### 5.1 Exact counting along one-dimensional probes

Let $f : \mathbb{R}^d \to \mathbb{R}^m$ be a multilayer perceptron with piecewise-affine activations. For a probe (line segment or curve) $\gamma : [a, b] \to \mathbb{R}^d$, the composed map $t \mapsto f(\gamma(t))$ is a CPWA function of $t$. We compute the exact number of affine pieces by propagating an interval partition of $[a, b]$ through the network layer by layer. On any current interval, each neuron's pre-activation is affine in $t$, and therefore contributes at most one breakpoint on that interval (for single-breakpoint activations such as ReLU/LeakyReLU).

### 5.2 Exact bounded-domain region enumeration in 2D and higher dimensions

Exact global counting of affine regions over $\mathbb{R}^d$ quickly becomes computationally prohibitive as depth, width, and input dimension increase. For $d \geq 2$, we therefore perform *exact bounded-domain enumeration*: we exhaustively enumerate all realized affine cells intersecting a prescribed compact domain $\Omega \subset \mathbb{R}^d$, rather than attempting global enumeration over $\mathbb{R}^d$.

**Bounded-domain objective.** Let $\Omega \subset \mathbb{R}^d$ be a compact domain of interest (e.g., $\Omega = [-1, 1]^d$). We aim to exactly enumerate $\mathcal{R}_\Omega(f) := \{\text{affine cells of } f \text{ intersecting } \Omega\}$, and compute statistics derived from $\mathcal{R}_\Omega(f)$ within $\Omega$ (in particular, the exact region count $|\mathcal{R}_\Omega(f)|$).

**Activation-region representation.** For piecewise-affine MLPs, fixing an activation pattern induces a local affine map $f(x) = Ax + b$ on an associated affine region (possibly one connected component among multiple components sharing the same activation signature). Hence, bounded-domain enumeration can be formulated as traversal over activation-induced candidate regions under feasibility and intersection constraints with $\Omega$.

**Enumeration workflow.** Starting from one or more feasible seeds intersecting $\Omega$, we iteratively (i) recover local affine/hyperplane descriptors for the current region, (ii) identify valid neighboring regions via boundary-crossing operations, (iii) test intersection with $\Omega$, and (iv) register previously unseen regions. When the

**Algorithm 2** Exact bounded-domain affine-region enumeration for $d \geq 2$

**Require:** Trained PANN $f$, bounded domain $\Omega \subset \mathbb{R}^d$
**Ensure:** Exactly enumerated region set $\widehat{\mathcal{R}}_\Omega = \mathcal{R}_\Omega(f)$ and exact region statistics in $\Omega$

```
Find an initial feasible region r_0 such that r_0 ∩ Ω ≠ ∅
Q ← {r_0},   R̂_Ω ← ∅
while Q ≠ ∅ do
    Pop one candidate region r from Q
    if r ∈ R̂_Ω then
        continue
    end if
    if r ∩ Ω = ∅ then
        continue
    end if
    Add r to R̂_Ω
    Compute local descriptors of r (activation pattern / affine form / boundary hyperplanes)
    Generate neighboring candidates N(r) by valid boundary-crossing operations
    for each r' ∈ N(r) do
        if r' ∉ R̂_Ω and r' is feasible then
            Push r' into Q
        end if
    end for
end while
return R̂_Ω and statistics derived from R̂_Ω
```

frontier is exhausted, this procedure yields an *exactly enumerated* region set $\widehat{\mathcal{R}}_\Omega = \mathcal{R}_\Omega(f)$ within $\Omega$, from which region statistics (including region counts) are computed exactly.

## 6 Experiments

### 6.1 Implementation and common setup

All experiments are implemented in PyTorch using fully-connected ReLU MLPs. Hidden architectures are denoted by $[n_1, n_2, \ldots, n_L]$, where each hidden layer is followed by a ReLU and the output layer is linear. Unless otherwise stated, we train with Adam, batch size 32, and cross-entropy loss (classification). For 2D experiments, inputs are normalized to $\Omega = [-1, 1]^2$. In 1D, we compute $\alpha(f; \gamma)$ exactly via interval-partition propagation (Algorithm 1). In 2D, we exactly enumerate all realized affine regions intersecting $\Omega$ by exhaustive bounded-domain region exploration (Algorithm 2). Additional ablations on residual connections under complex 2D inputs are reported in Appendix C, together with further visualizations of decision-boundary dynamics throughout training in Appendix D.

### 6.2 Exact 1D enumeration

This section uses *exact* 1D affine-piece enumeration to (i) sanity-check our counting pipeline against the proven line-region bound in Theorem 4, (ii) isolate architecture-induced line complexity via an initialization-only stress test (no optimization), (iii) empirically instantiate the neuron-threshold implication in Corollary 6 on controlled targets with known piece complexity, and (iv) quantify the practical region-yield statistics ($\eta$ and $\alpha$) that motivate the heuristic discussion in Section 4. All 1D experiments use ReLU activations (i.e., $K = 1$), so the bound in Theorem 4 specializes to $\alpha(f; \gamma) \leq Q_{\text{upper}}$, where $Q_{\text{upper}} = \prod_{\ell=1}^{L}(n_\ell + 1)$.

#### 6.2.1 Protocol and metrics

**Exact counting along probes.** For each network $f$ and probe $\gamma$, we compute $\alpha(f; \gamma)$ *exactly* using Algorithm 1. For a probe distribution $\Gamma$, we report the mean region count $\eta = \mathbb{E}_{\gamma \sim \Gamma}[\alpha(f; \gamma)]$, the regions-per-

neuron statistic $\alpha = \mathbb{E}_{\gamma\sim\Gamma}[\alpha(f;\gamma)]/N$, and the normalized efficiency $\eta_{\text{norm}} = \mathbb{E}_{\gamma\sim\Gamma}[\alpha(f;\gamma)]/Q_{\text{upper}}$, where $N = \sum_{\ell=1}^{L} n_\ell$ is the total number of hidden neurons.

**Architectures and probes.** We evaluate three ReLU MLP families, $[32]^5$ ($N = 160$), $[64]^3$ ($N = 192$), and $[128]^3$ ($N = 384$), aggregated over multiple random seeds. Unless otherwise stated, each run uses 200 random line probes for estimating expectations over $\Gamma$.

**Initialization-only stress test.** To remove optimization effects and probe architecture-induced line complexity directly, we additionally conduct an initialization-only stress test with no training updates. For each architecture, we sample $S = 3$ random seeds and $M = 100$ probes per seed, compute $\alpha(f;\gamma)$ exactly, and record the extreme statistic $\text{MaxLR} = \max_\gamma \alpha(f;\gamma)$ as well as the normalized ratio $\rho_{\max} = \max_\gamma \alpha(f;\gamma)/Q_{\text{upper}}$; the resulting trends and ratios are summarized in Figure 2.

**Controlled synthetic threshold suite.** To instantiate Corollary 6 under known target complexity, we train depth-$L = 3$ ReLU MLPs with equal-width hidden layers $[w, w, w]$ (width sweep $w \in \{2, 4, \ldots, 64\}$, hence $N = 3w$) on synthetic 1D piecewise-affine targets $g : [0, 1] \to \mathbb{R}$ with piece complexity $Q(g) \in \{4, 8, 12, 16, 24, 32\}$. Each $(Q, w)$ is trained with 3 seeds; a run is deemed successful if the max absolute error is $\leq 2 \times 10^{-2}$ on a dense evaluation grid over $[0, 1]$. We report $\hat{N}^{\text{obs}}_{\min}$ as the smallest $N$ achieving the target success criterion, using both a "loose" rule (success rate $\geq 2/3$) and a "strict" rule (all seeds successful), summarized in Table 1.

#### 6.2.2 Consistency checks for Theorem 4

**Trained networks.** Across all aggregated trained runs and probes, we observe no violations of the specialized bound, i.e., `violation_count`=0. The distribution of $\alpha(f;\gamma)$ computed by exact enumeration is shown in Figure 1 (top), and lies far below $Q_{\text{upper}}$, illustrating the gap between worst-case architectural capacity and realized region usage after training. The same figure also reports layer-wise split statistics (bottom), visualizing how affine-piece refinements accumulate through depth.

**Initialization-only stress test.** We also observe no upper-bound violations under random initialization, and MaxLR increases with architecture size; Figure 2 (left) reports $\max_\gamma \alpha(f;\gamma)$ as a function of $N$ (mean±std across seeds). Despite this growth, the normalized extreme ratio $\rho_{\max}$ remains far below 1; Figure 2 (middle) shows per-seed $\rho_{\max}$ values (with mean±std) on a log scale, quantifying substantial slack between realized and worst-case theoretical capacity at random initialization. Finally, Figure 2 (right) reports the non-violation rate, which is 1.0 in all tested settings.

#### 6.2.3 Instantiation of Corollary 6 on known-$Q$ targets

For the synthetic suite with $L = 3$ and ReLU, Corollary 6 yields the necessary condition $(w + 1)^3 \geq Q(g)$, equivalently $N \geq L\big(Q(g)^{1/L} - 1\big)$. In our experiments, no successful fit contradicts this implication over the tested range (Table 1). This should be read as an empirical *non-refutation* of the necessary condition under our optimization protocol and error criterion, rather than as a sharp characterization of the minimal achievable $N$, since finite optimization budgets can require substantially more neurons than the existence-based lower bound. The observed thresholds in Table 1 are much larger than the theoretical necessary condition, directly quantifying the optimization-limited gap between architectural capacity and realized expressivity. Together with the initialization-only stress test in Figure 2, which already exhibits substantial slack relative to $Q_{\text{upper}}$, these results underscore that worst-case region-capacity bounds can be loose in typical parameter regimes and motivate reporting $\alpha$ as a practical summary statistic for realized region yield.

#### 6.2.4 Practical region-yield statistics and the $Q/\alpha$ rule

Using the empirically estimated $\alpha$, we evaluate the rule-of-thumb predictor $N^{\text{practical}}_{\min} \approx \lceil Q/\alpha \rceil$. The architecture-level trends of $\mathbb{E}[\alpha]$, $\alpha$, and $\eta_{\text{norm}}$ are summarized in Figure 3 (top), while Figure 3 (bottom) compares the theoretical necessary-condition curve from Corollary 6 with the heuristic $Q/\alpha$ and an empirical proxy derived from probe statistics.

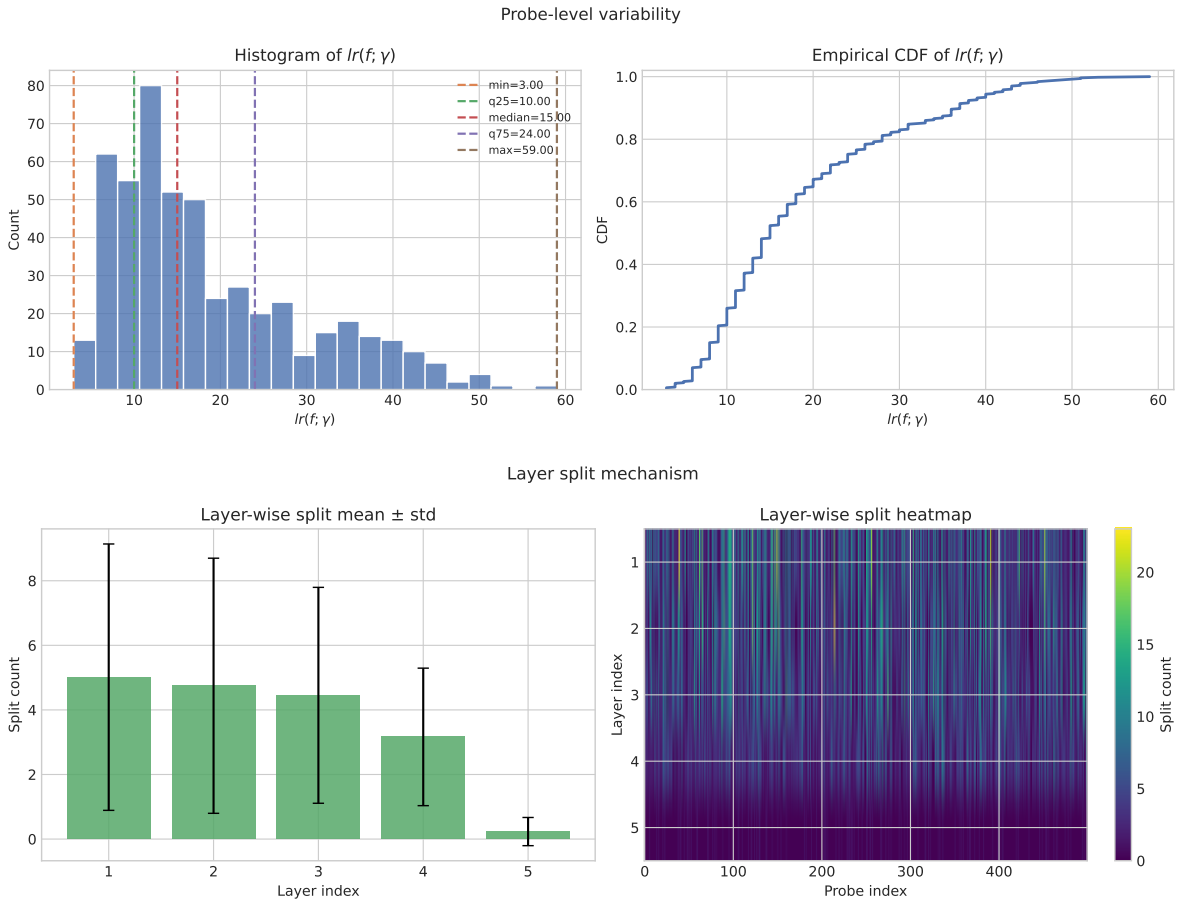



Figure 1: **Exact 1D probe statistics (trained networks).** Top: distribution/CDF of $\alpha(f;\gamma)$ computed by exact enumeration. Bottom: layer-wise split statistics along probes, illustrating how affine-piece refinements accumulate through depth.

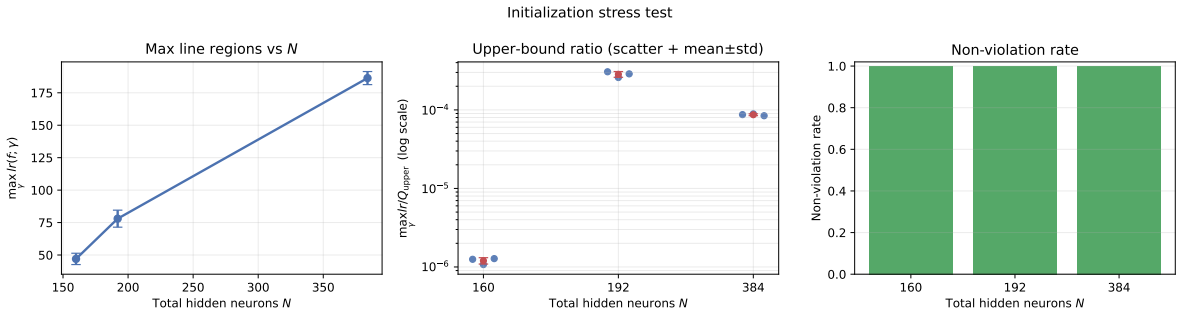



Figure 2: **Initialization-only stress test (exact 1D counting).** Left: $\max_\gamma \alpha(f;\gamma)$ versus $N$ (mean±std across seeds). Middle: per-seed $\rho_{\max}$ scatter with mean±std error bars (log-scale y-axis). Right: non-violation rate of the theoretical upper bound.

**Summary (1D).** Exact enumeration provides (i) a broad consistency check of our counting pipeline against the proven ReLU-specialized bound in Theorem 4, including an initialization-only stress test that removes optimization effects (Figure 2), (ii) an empirical non-refutation of the necessary-condition implication in Corollary 6 on controlled targets (Table 1), and (iii) quantitative region-yield statistics ($\alpha$ and $\eta_{\text{norm}}$) that summarize the realized-region gap and support the practical discussion in Section 4 (Figures 1 and 3).

Table 1: Synthetic threshold suite: observed neuron budgets versus the necessary-condition lower bound in Corollary 6 ($L = 3, K = 1$).

| $Q(g)$ | $\lceil L(Q^{1/L} - 1) \rceil$ | $\hat{N}_{\min}^{\text{obs}}$ (loose) | $\hat{N}_{\min}^{\text{obs}}$ (strict) | violation |
|---|---|---|---|---|
| 4 | 2 | 108 | 108 | no |
| 8 | 3 | 150 | 150 | no |
| 12 | 4 | 138 | 138 | no |
| 16 | 5 | 144 | 144 | no |
| 24 | 6 | – | – | no observed success |
| 32 | 7 | – | – | no observed success |

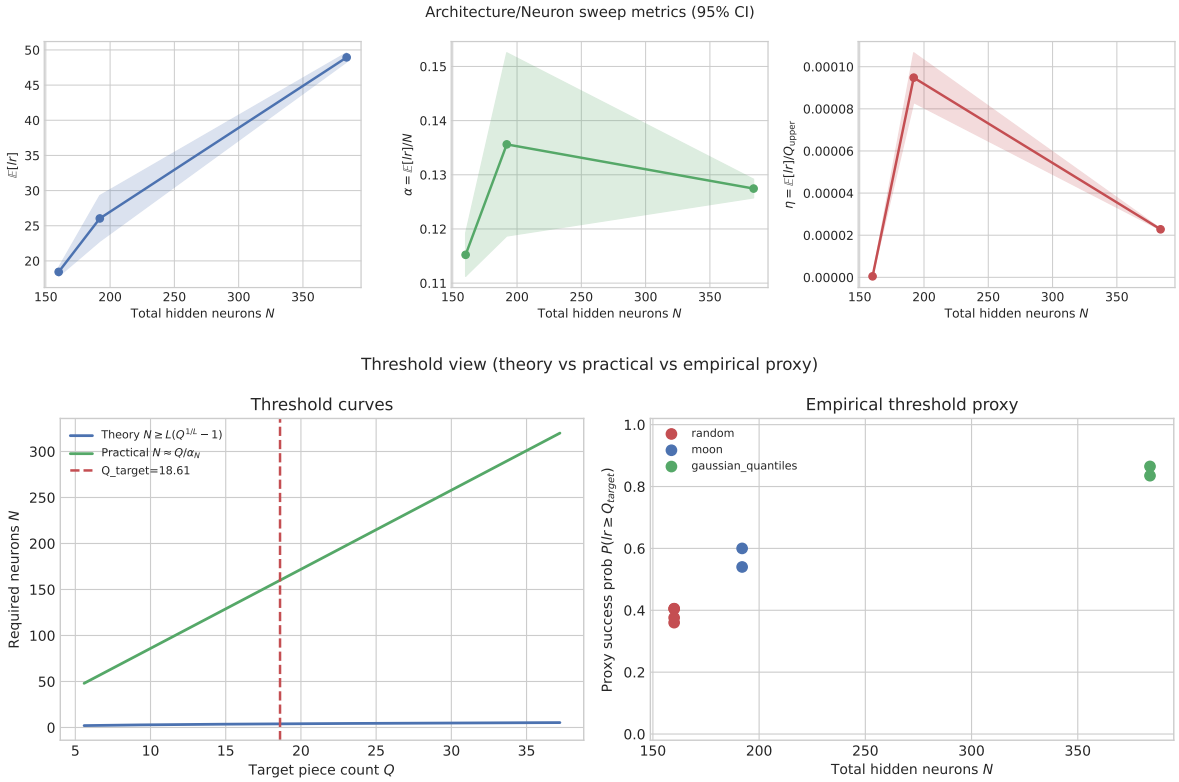



Figure 3: **Architecture-level 1D trends and threshold proxy.** Top: $\mathbb{E}[\alpha]$, $\alpha$, and $\eta_{\text{norm}}$ versus $N$. Bottom: theoretical/practical curves together with an empirical proxy $P(\alpha \geq Q_{\text{target}})$ from probe statistics.

### 6.3 2D exact region enumeration and decision-boundary visualization

This section studies realized affine-region formation directly in the input plane and links region usage to classification behavior. All 2D region counts reported here are obtained by *exact enumeration* of all affine regions intersecting the bounded domain $\Omega = [-1, 1]^2$ using Algorithm 2 (bounded-domain exhaustive exploration). We pair exact region counts with decision-boundary visualizations to connect geometric expressivity to classification outcomes.

#### 6.3.1 Expressivity saturation under increasing task complexity

We study random data with random labels while increasing the number of samples, using a fixed ReLU MLP architecture $[32, 32, 32]$ and counting domain $\Omega = [-1, 1]^2$. Exact region counts across training epochs and sample sizes are summarized in Table 2, and representative region/decision visualizations are shown in Figure 4.

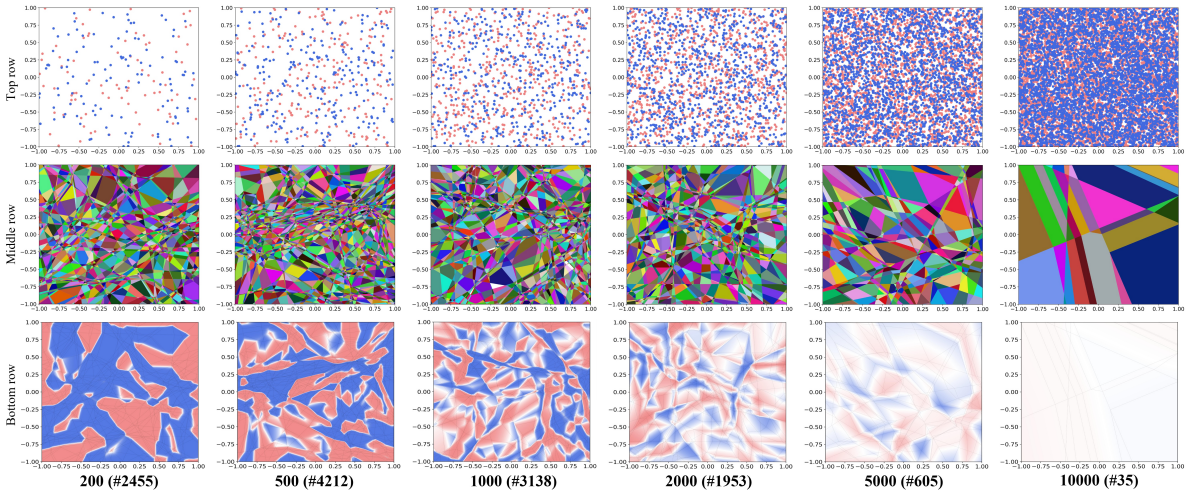



Figure 4: **Random labels with increasing sample size (network** $[32, 32, 32]$**).** Top: training data. Middle: *exactly enumerated* affine regions in $\Omega$. Bottom: decision regions. Increasing task complexity induces a pronounced collapse in realized regions and degraded decision boundaries.

Table 2: **Exact** 2D affine-region counts in $\Omega$ for network $[32, 32, 32]$ on random labels (varying sample size and training epoch).

| Samples \ Epoch | 0 | 10 | 30 | 50 | 80 | 100 | 300 | 500 | 800 | 1000 | 3000 | 5000 | 10000 | 30000 | 50000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 200 | 1151 | 925 | 953 | 1011 | 995 | 1176 | 1426 | 1499 | 1858 | 1749 | 1896 | 2108 | 1988 | 2149 | 2455 |
| 500 | 1151 | 977 | 1393 | 1441 | 1572 | 1549 | 2067 | 2316 | 2542 | 2771 | 3129 | 3616 | 3884 | 4377 | 4212 |
| 1000 | 1153 | 992 | 928 | 984 | 1180 | 1244 | 1700 | 2047 | 2236 | 2195 | 2453 | 2667 | 2810 | 3010 | 3138 |
| 2000 | 1154 | 791 | 1222 | 1308 | 1344 | 1423 | 1250 | 1205 | 1346 | 1324 | 1431 | 1519 | 1545 | 1910 | 1953 |
| 5000 | 1151 | 834 | 1020 | 560 | 391 | 325 | 158 | 111 | 88 | 143 | 379 | 344 | 415 | 442 | 605 |
| 10000 | 1152 | 873 | 353 | 316 | 198 | 79 | 70 | 109 | 61 | 80 | 42 | 51 | 31 | 34 | 35 |

**Observation (exact region collapse).** As the sample size grows, training becomes harder and the number of *realized* affine regions in $\Omega$ exhibited by the trained network collapses sharply under our training protocol, despite unchanged worst-case theoretical capacity. This effect is quantified by the exact enumeration results in Table 2, which show that region counts can drop by orders of magnitude in the most challenging regimes we tested. The corresponding geometric degradation is illustrated in Figure 4, where increasing task complexity yields progressively coarser partitions and degraded decision regions.

**Ablation: initialization and optimizer.** To check that the observed collapse is not an artifact of a particular training configuration, we repeat the random-label experiment with 5000 samples and compare the exact region count at epoch 10000 under Xavier initialization and under SGD (keeping the remaining settings the same); the qualitative conclusion is unchanged, as all variants remain in the collapsed regime (see Table 3).

**Decision-boundary evolution versus boundary failure under complex inputs.** On a more regular, learnable dataset, the decision boundary evolves over training and progressively aligns with the data geometry; Figure 5 illustrates this behavior on `make_moons`, where both the exactly enumerated affine regions and the induced decision regions refine throughout training. In contrast, under random labels with 1000 samples, Figure 4 shows that the network fails to form an effective decision boundary, consistent with the concurrent reduction in realized affine regions reported in Table 2.

Table 3: **Exact** region counts at epoch 10000 for random labels with 5000 samples under initialization/optimizer ablations (network $[32, 32, 32]$, domain $\Omega = [-1, 1]^2$).

| Configuration | Exact #regions in $\Omega$ at epoch 10000 |
|---|---|
| Baseline setting (as in Table 2) | 415 |
| Xavier initialization (other settings unchanged) | 586 |
| SGD optimizer (other settings unchanged) | 965 |

Table 4: **Exact** realized affine-region counts in $\Omega = [-2, 2]^3$ at epoch 100 for network $[32, 32, 32]$ trained on random data with random labels, computed by exhaustive bounded-domain exploration (Algorithm 2).

| Dimension $d$ | 10000 samples | 30000 samples | 50000 samples | 70000 samples | 100000 samples |
|---|---|---|---|---|---|
| 3 | 30964 | 20321 | 15693 | 12676 | 9187 |

### 6.4 High-dimensional exact enumeration of realized affine regions

We extend exact bounded-domain region enumeration to higher input dimensions and report *exact* counts of realized affine regions intersecting a compact domain. We fix the ReLU MLP architecture to $[32, 32, 32]$ (as in the 2D experiments) and keep the training protocol unchanged; only the input dimension and the number of random samples vary.

**Dataset and domain.** We consider the random-data-with-random-labels setting in three dimensions ($d = 3$), where inputs are sampled uniformly from $\Omega = [-2, 2]^3$ and labels are assigned i.i.d. at random. We evaluate sample sizes 10000, 30000, 50000, 70000, and 100000.

**Exact region enumeration.** After training to epoch 100, we enumerate *exactly* the set of realized affine regions intersecting $\Omega$ using Algorithm 2. The reported numbers therefore equal $|\mathcal{R}_\Omega(f)|$ within $\Omega$. Table 4 reports the exact region counts at epoch 100. Consistent with the 2D findings, increasing sample size under random labels is associated with a marked decrease in the number of realized affine regions within $\Omega$ exhibited by the trained network for the fixed architecture under our protocol.

## 7 Conclusion

We study the gap between CPWA networks' affine-region capacity and regions realized after training. We prove an architecture-dependent upper bound on affine pieces along any 1D probe, yielding a necessary neuron threshold for 1D piece complexity. 1D counting and bounded-domain enumeration in 2D and higher dimensions show *expressivity saturation*: task difficulty correlates with fewer realized regions and, in 2D, degraded decision boundaries. Future work will scale exact counting and extend theory beyond probes to broader architectures and training dynamics.

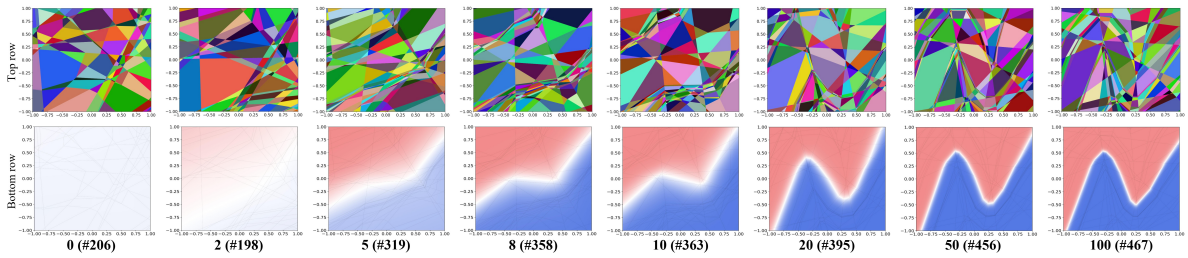



Figure 5: **Decision-boundary evolution on `make_moons`.** Network $[16, 16, 16]$. Top: *exactly enumerated* affine regions in $\Omega$. Bottom: decision regions. Epochs: $[0, 2, 5, 8, 10, 20, 50, 100]$.

## References


Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, U.K., 2002. ISBN 978-0-521-57353-5.

Gennadiy Averkov, Christopher Hojny, and Maximilian Merkert. On the expressiveness of rational ReLU neural networks with bounded depth. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2025.

Randall Balestriero and Richard G. Baraniuk. A spline theory of deep learning. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 80, pp. 374–383. PMLR, 2018.

Randall Balestriero and Yann LeCun. Fast and exact enumeration of deep networks partitions regions. *arXiv preprint arXiv:2401.11188*, 2024. doi: 10.48550/arXiv.2401.11188.

Arturs Berzins. Polyhedral complex extraction from ReLU networks using edge subdivision. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 202, pp. 2234–2244. PMLR, 2023.

Hao Chen, Yu Guang Wang, and Huan Xiong. Lower and upper bounds for numbers of linear regions of graph convolutional networks. *Neural Networks*, 168:394–404, 2023. doi: 10.1016/j.neunet.2023.09.025.

Kuan-Lin Chen, Harinath Garudadri, and Bhaskar D. Rao. Improved bounds on neural complexity for representing piecewise linear functions. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Setareh Cohan, Nam Hee Kim, David Rolnick, and Michiel van de Panne. Understanding the evolution of linear regions in deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Alexis Goujon, Arian Etemadi, and Michael Unser. On the number of regions of piecewise linear neural networks. *J. Comput. Appl. Math.*, 441:115667, 2024. doi: 10.1016/j.cam.2023.115667.

J. Elisenda Grigsby and Kathryn Lindsey. On transversality of bent hyperplane arrangements and the topological expressiveness of ReLU neural networks. *SIAM J. Appl. Algebra Geom.*, 6(2):216–242, 2022. doi: 10.1137/20M1368902.

Christian Haase, Christoph Hertrich, and Georg Loho. Lower bounds on the depth of integral ReLU neural networks via lattice polytopes. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2023.

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

Boris Hanin, Ryan S. Jeong, and David Rolnick. Deep ReLU networks preserve expected length. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2022.

Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, 2020. doi: 10.4208/jcm.1901-m2018-0160.

Christoph Hertrich, Amitabh Basu, Marco Di Summa, and Martin Skutella. Towards lower bounds on the depth of ReLU neural networks. *SIAM J. Discret. Math.*, 37(2):997–1029, 2023. doi: 10.1137/22M1489332.

Qiang Hu, Hao Zhang, Feifei Gao, Chengwen Xing, and Jianping An. Analysis on the number of linear regions of piecewise linear neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 33(2):644–653, 2022. doi: 10.1109/TNNLS.2020.3028431.

Ahmed Imtiaz Humayun, Randall Balestriero, Guha Balakrishnan, and Richard G. Baraniuk. Splinecam: Exact visualization and characterization of deep network geometry and decision boundaries. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 3789–3798. IEEE, 2023. doi: 10.1109/CVPR52729.2023.00369.

Zhiwei Li and Cheng Wang. Achieving sharp upper bounds on the expressive power of neural networks via tropical polynomials. *IEEE Trans. Neural Netw. Learn. Syst.*, 36(2):2931–2945, 2025. doi: 10.1109/TNNLS.2024.3350786.

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

Max Milkert, David Hyde, and Forrest John Laine. Compelling relu networks to exhibit exponentially many linear regions at initialization and during training. In *International Conference on Machine Learning*, pp. 44175–44198. PMLR, 2025.

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*, volume 27, pp. 2924–2932, 2014.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 807–814. Omnipress, 2010.

Duc Anh Nguyen, Ernesto Araya, Adalbert Fono, and Gitta Kutyniok. Time to spike? understanding the representational power of spiking neural networks in discrete time. In *Proc. Int. Conf. Mach. Learn. (ICML)*. OpenReview.net, 2025.

Niket Patel and Guido Montúfar. On the local complexity of linear regions in deep ReLU networks. In *Proc. Int. Conf. Mach. Learn. (ICML)*. OpenReview.net, 2025.

Mary Phuong and Christoph H. Lampert. Functional vs. parametric equivalence of ReLU networks. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2020.

Pawel Piwek, Adam Klukowski, and Tianyang Hu. Exact count of boundary pieces of ReLU classifiers: Towards the proper complexity measure for classification. In *Proc. Conf. Uncertainty in Artificial Intelligence (UAI)*, volume 216, pp. 1673–1683. PMLR, 2023.

David Rolnick and Konrad P. Kording. Reverse-engineering deep ReLU networks. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 119, pp. 8178–8187. PMLR, 2020.

Saket Tiwari and George Konidaris. Effects of data geometry in early deep learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Martin Trimmel, Henning Petzka, and Cristian Sminchisescu. Tropex: An algorithm for extracting linear terms in deep neural networks. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2021.

Hanna Tseran and Guido Montúfar. On the expected complexity of maxout networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 28995–29008, 2021.

Yuan Wang. Estimation and comparison of linear regions for ReLU networks. In *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, pp. 3544–3550. ijcai.org, 2022. doi: 10.24963/ijcai.2022/492.

Huan Xiong, Lei Huang, Wenston J. T. Zang, Xiantong Zhen, Guo-Sen Xie, Bin Gu, and Le Song. On the number of linear regions of convolutional neural networks with piecewise linear activations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(7):5131–5148, 2024. doi: 10.1109/TPAMI.2024.3361155.

Xiao Zhang and Dongrui Wu. Empirical studies on the properties of linear regions in deep neural networks. In *Proc. Int. Conf. Learn. Represent. (ICLR)*. OpenReview.net, 2020.

# A Proofs

## A.1 Proof of Theorem 4

Throughout this subsection, we consider a *line-segment probe*

$$\gamma(t) \;=\; x_0 + t\,v, \qquad t \in [0,1], \tag{12}$$

for some $x_0, v \in \mathbb{R}^d$. This is the standard setting for line-restricted region counting.

**CPWA activation.** Let $\sigma : \mathbb{R} \to \mathbb{R}$ be CPWA with (finite) breakpoint set

$$-\infty = t_0 < t_1 < \cdots < t_K < t_{K+1} = +\infty, \tag{13}$$

such that for each $k \in \{1, \ldots, K+1\}$ there exist scalars $c_k, d_k \in \mathbb{R}$ with

$$\sigma(z) \;=\; c_k z + d_k, \qquad z \in (t_{k-1}, t_k). \tag{14}$$

**Network notation.** Define the hidden layers recursively by

$$h_0(x) := x, \tag{15}$$
$$h_\ell(x) := \sigma\big(W_\ell h_{\ell-1}(x) + b_\ell\big), \qquad \ell = 1, \ldots, L, \tag{16}$$
$$f(x) := W_{L+1} h_L(x) + b_{L+1}, \tag{17}$$

where $\sigma$ acts coordinate-wise.

**Piece count along the probe.** For each $\ell \in \{0, 1, \ldots, L\}$, let $R_\ell$ be the smallest integer such that there exists a partition

$$0 = \tau_0 < \tau_1 < \cdots < \tau_{R_\ell} = 1 \tag{18}$$

for which $t \mapsto h_\ell(\gamma(t))$ is affine on every open interval $(\tau_{r-1}, \tau_r)$. Since the output layer is affine in $h_L$, any partition witnessing affine behavior for $h_L \circ \gamma$ also witnesses affine behavior for $f \circ \gamma$, hence

$$\alpha(f; \gamma) \;\le\; R_L. \tag{19}$$

We now prove a layer-wise refinement bound.

**Lemma 8** (Affine pre-activations on each incoming affine piece)**.** *Fix $\ell \in \{1, \ldots, L\}$ and an open interval $I \subset (0,1)$. If $t \mapsto h_{\ell-1}(\gamma(t))$ is affine on $I$, then for each neuron $j \in \{1, \ldots, n_\ell\}$ the pre-activation*

$$a_{\ell,j}(t) \;:=\; e_j^\top\big(W_\ell h_{\ell-1}(\gamma(t)) + b_\ell\big) \tag{20}$$

*is an affine function of $t$ on $I$.*

*Proof.* If $h_{\ell-1}(\gamma(t))$ is affine on $I$, then there exist $u, v \in \mathbb{R}^{n_{\ell-1}}$ such that

$$h_{\ell-1}(\gamma(t)) \;=\; u + t\,v, \qquad t \in I. \tag{21}$$

Substituting equation 21 into equation 20 yields

$$a_{\ell,j}(t) \;=\; e_j^\top (W_\ell u + b_\ell) \;+\; t\, e_j^\top (W_\ell v), \tag{22}$$

which is affine in $t$ on $I$. □

**Lemma 9** (A CPWA scalar neuron introduces at most $K$ split points on an affine input)**.** *Let $I \subset \mathbb{R}$ be a nonempty open interval and let $a : I \to \mathbb{R}$ be affine. Then $\sigma \circ a$ is piecewise-affine on $I$ and there exists a partition of $I$ into at most $K+1$ open subintervals on each of which $\sigma \circ a$ is affine. Equivalently, the affine formula of $\sigma \circ a$ changes at most $K$ times on $I$.*

*Proof.* Write $a(t) = \alpha t + \beta$ on $I$. For each breakpoint $t_k$ in equation 13, consider the equation

$$a(t) \;=\; t_k. \tag{23}$$

If $\alpha \neq 0$, then equation 23 has at most one solution in $I$ for each $k \in \{1, \ldots, K\}$. If $\alpha = 0$, then $a$ is constant on $I$ and $\sigma \circ a$ is constant (hence affine) on $I$. Define the set of potential split points

$$\mathcal{Z} \;:=\; \big\{t \in I :\; a(t) \in \{t_1, \ldots, t_K\}\big\}. \tag{24}$$

From the above discussion,

$$|\mathcal{Z}| \;\leq\; K. \tag{25}$$

On any connected component of $I \setminus \mathcal{Z}$, the value $a(t)$ lies within a single interval $(t_{k-1}, t_k)$, hence $\sigma$ follows a fixed affine rule equation 14 and $\sigma \circ a$ is affine there. Removing at most $K$ points from an open interval produces at most $K + 1$ open components, proving the claim. □

**Lemma 10** (One layer of width $n_\ell$ refines each incoming affine piece by at most $Kn_\ell$ splits)**.** *Fix $\ell \in \{1, \ldots, L\}$ and an open interval $I \subset (0, 1)$. If $t \mapsto h_{\ell-1}(\gamma(t))$ is affine on $I$, then there exists a partition of $I$ into at most $Kn_\ell + 1$ open subintervals on each of which $t \mapsto h_\ell(\gamma(t))$ is affine.*

*Proof.* By Lemma 8, each pre-activation $a_{\ell,j}$ is affine on $I$. By Lemma 9, for each $j$ there is a set $\mathcal{Z}_j \subset I$ with

$$|\mathcal{Z}_j| \;\leq\; K \tag{26}$$

such that $\sigma \circ a_{\ell,j}$ is affine on every connected component of $I \setminus \mathcal{Z}_j$. Let

$$\mathcal{Z} \;:=\; \bigcup_{j=1}^{n_\ell} \mathcal{Z}_j. \tag{27}$$

Then

$$|\mathcal{Z}| \;\leq\; \sum_{j=1}^{n_\ell} |\mathcal{Z}_j| \;\leq\; Kn_\ell. \tag{28}$$

On any connected component of $I \setminus \mathcal{Z}$, every coordinate $\sigma(a_{\ell,j}(t))$ follows a fixed affine branch, so there exist a diagonal matrix $C \in \mathbb{R}^{n_\ell \times n_\ell}$ and a vector $d \in \mathbb{R}^{n_\ell}$ (constant on that component) with

$$\sigma\big(W_\ell h_{\ell-1}(\gamma(t)) + b_\ell\big) \;=\; C\big(W_\ell h_{\ell-1}(\gamma(t)) + b_\ell\big) + d. \tag{29}$$

Since $h_{\ell-1}(\gamma(t))$ is affine on $I$, the right-hand side of equation 29 is affine on that component. Finally, removing at most $Kn_\ell$ points from an open interval yields at most $Kn_\ell + 1$ open components, completing the proof. □

*Proof of Theorem 4.* We prove by induction that

$$R_\ell \;\leq\; \prod_{s=1}^{\ell} (Kn_s + 1), \qquad \ell \in \{0, 1, \ldots, L\}, \tag{30}$$

with the convention that the empty product equals 1.

**Base case.** By equation 12 and equation 15, the map $t \mapsto h_0(\gamma(t))$ is affine on $(0, 1)$, hence

$$R_0 \;=\; 1, \tag{31}$$

which matches equation 30 for $\ell = 0$.

**Inductive step.** Fix $\ell \in \{1, \dots, L\}$ and assume equation 30 holds for $\ell - 1$. Let $[0, 1]$ be partitioned into $R_{\ell-1}$ intervals as in equation 18 such that $h_{\ell-1} \circ \gamma$ is affine on each open interval. On each such open interval, Lemma 10 gives a refinement into at most $Kn_\ell + 1$ open subintervals on which $h_\ell \circ \gamma$ is affine. Therefore

$$R_\ell \;\le\; (Kn_\ell + 1)\, R_{\ell-1}. \tag{32}$$

Combining equation 32 with the inductive hypothesis yields equation 30 for $\ell$.

**Conclusion.** Taking $\ell = L$ in equation 30 gives

$$R_L \;\le\; \prod_{\ell=1}^{L} (Kn_\ell + 1). \tag{33}$$

Using equation 19, we obtain

$$\alpha(f; \gamma) \;\le\; \prod_{\ell=1}^{L} (Kn_\ell + 1), \tag{34}$$

which is the desired bound. □

### A.2 Proof of Corollary 6

*Proof.* Assume $f(\gamma(t)) = g(t)$ for all $t \in [0, 1]$. By definition of $Q(g)$, any partition of $[0, 1]$ into intervals on which $g$ is affine must have at least $Q(g)$ intervals. Since $g = f \circ \gamma$, any partition witnessing $\alpha(f; \gamma)$ also witnesses that $g$ is affine on each piece. Hence

$$Q(g) \;\le\; \alpha(f; \gamma). \tag{35}$$

Applying Theorem 4 yields

$$\alpha(f; \gamma) \;\le\; \prod_{\ell=1}^{L} (Kn_\ell + 1), \tag{36}$$

and therefore

$$\prod_{\ell=1}^{L} (Kn_\ell + 1) \;\ge\; Q(g). \tag{37}$$

Let $N := \sum_{\ell=1}^{L} n_\ell$. By AM–GM applied to the positive numbers $(Kn_\ell + 1)_{\ell=1}^{L}$,

$$\prod_{\ell=1}^{L} (Kn_\ell + 1) \;\le\; \left( \frac{1}{L} \sum_{\ell=1}^{L} (Kn_\ell + 1) \right)^{L}. \tag{38}$$

Compute the mean:

$$\frac{1}{L} \sum_{\ell=1}^{L} (Kn_\ell + 1) = \frac{1}{L} \left( K \sum_{\ell=1}^{L} n_\ell + \sum_{\ell=1}^{L} 1 \right), \tag{39}$$

$$= \frac{1}{L} (KN + L), \tag{40}$$

$$= 1 + \frac{KN}{L}. \tag{41}$$

Combining equation 37, equation 38, and equation 41 gives

$$Q(g) \;\le\; \left( 1 + \frac{KN}{L} \right)^{L}. \tag{42}$$

Taking $L$-th roots and rearranging yields

$$N \;\geq\; \frac{L}{K}\big(Q(g)^{1/L} - 1\big). \tag{43}$$

If $n_\ell = w$ for all $\ell$, then equation 37 becomes

$$(Kw + 1)^L \;\geq\; Q(g), \tag{44}$$

equivalently

$$w \;\geq\; \frac{Q(g)^{1/L} - 1}{K}. \tag{45}$$

Since $w$ is an integer, we obtain

$$w \;\geq\; \left\lceil \frac{Q(g)^{1/L} - 1}{K} \right\rceil, \tag{46}$$

as claimed. □

## B Limitations and future work.

Our theoretical results are probe-based and do not directly characterize full-dimensional region counts or the geometry of regions away from one-dimensional trajectories. Exact bounded-domain enumeration, while exact, is computationally constrained and currently most practical in low dimensions and moderate architectures. Empirically, we focused on fully-connected ReLU MLPs and used random-label regimes as a controlled proxy for extreme task complexity; extending these conclusions to broader architectures (e.g., convolutional and attention-based models) and more structured notions of data complexity remains important. Promising directions include: developing theory for region formation under specific training dynamics; connecting region collapse to known optimization phenomena (e.g., gradient starvation or implicit bias); designing regularizers or curricula that stabilize region refinement under complex inputs; and extending exact or certified counting methods to data manifolds and higher-dimensional domains.

# C Residual-Connection Ablation under Complex 2D Inputs

This appendix reports an additional ablation studying whether adding a residual (skip) connection mitigates the expressivity saturation effect observed in the main 2D experiments. All experimental settings (data generation, normalization domain, optimizer, loss, batch size, and evaluation protocol) are identical to those in Sections 6.1 and 6.3; the only change is the network architecture, where we add a single residual connection to each baseline MLP.

## C.1 Setup

**Data and task.** We consider 2D random inputs with random labels, with inputs normalized to the same bounded domain as in the main text, $\Omega = [-1, 1]^2$. We sweep the number of training samples over $N_{\text{data}} \in \{200, 500, 1000, 2000, 3000, 5000, 7000, 10000\}$, and train using cross-entropy loss (classification), matching the main experimental protocol.

**Optimization.** All runs use Adam, batch size 32, and are trained for 10000 epochs. Unless otherwise stated, all remaining hyperparameters and initialization follow Section 6.1 exactly.

**Architectures with a single residual connection.** We consider three depth-$L = 3$ ReLU MLP backbones with widths $[16, 16, 16]$, $[32, 32, 32]$, $[64, 64, 64]$, and augment each with one residual (skip) connection. Concretely, denoting the hidden representations by

$$h_1(x) = \sigma(W_1 x + b_1), \tag{47}$$

$$h_2(x) = \sigma(W_2 h_1(x) + b_2), \tag{48}$$

$$\tilde{h}_3(x) = \sigma(W_3 h_2(x) + b_3), \tag{49}$$

we define the residualized third hidden layer as

$$h_3(x) \;=\; \tilde{h}_3(x) + h_1(x), \tag{50}$$

i.e., a single identity skip from the first to the third hidden layer. The output layer remains linear:

$$f(x) = W_4 h_3(x) + b_4. \tag{51}$$

This construction preserves dimensional compatibility because all hidden layers have equal width within each backbone.

**Exact region counting.** As in the main text, the reported numbers are *exact* counts of realized affine regions intersecting $\Omega$, computed via exhaustive bounded-domain region exploration (Algorithm 2). For each configuration, we report the exact region count in $\Omega$ after training to epoch 10000.

## C.2 Results

**Exact realized region counts.** Table 5 reports the exact number of affine regions intersecting $\Omega = [-1, 1]^2$ for each residualized architecture and sample size at epoch 10000.

**Discussion.** Across all three widths, the exact realized region counts decrease substantially as the sample size increases, consistent with the expressivity saturation phenomenon reported in Section 6.3.1. In particular, for the largest sample size (10000), all residualized models are in a strongly collapsed regime (region counts on the order of $10^2$ or below for $[32, 32, 32]$ and $[64, 64, 64]$, and 42 for $[16, 16, 16]$). These results indicate that, under highly complex random-label supervision, adding a single residual connection in an otherwise identical MLP does not prevent the sharp reduction in realized affine-region usage within the bounded domain $\Omega$.

Table 5: **Exact** realized affine-region counts in $\Omega = [-1, 1]^2$ at epoch 10000 for residualized ReLU MLPs trained on random 2D inputs with random labels. All settings match the main experiments; only the architecture differs by adding one residual connection as in equation 50.

| Architecture (with one residual) | 200 | 500 | 1000 | 2000 | 3000 | 5000 | 7000 | 10000 |
|---|---|---|---|---|---|---|---|---|
| $[16, 16, 16]$ + residual | 2032 | 2212 | 2148 | 1902 | 825 | 545 | 305 | 42 |
| $[32, 32, 32]$ + residual | 3782 | 5561 | 3825 | 2112 | 2943 | 712 | 946 | 102 |
| $[64, 64, 64]$ + residual | 7649 | 9026 | 7463 | 6838 | 2715 | 1252 | 629 | 104 |

# D Additional Visualizations of Training Dynamics

## D.1 Datasets

(a) (b)

Figure 6: (a) The "*make moons*" dataset consisting of 1000 samples and 2 classes. (b) The "*make gaussian quantiles*" dataset consisting of 1000 samples and 5 classes.

### D.2 Curves of region count and accuracy during training

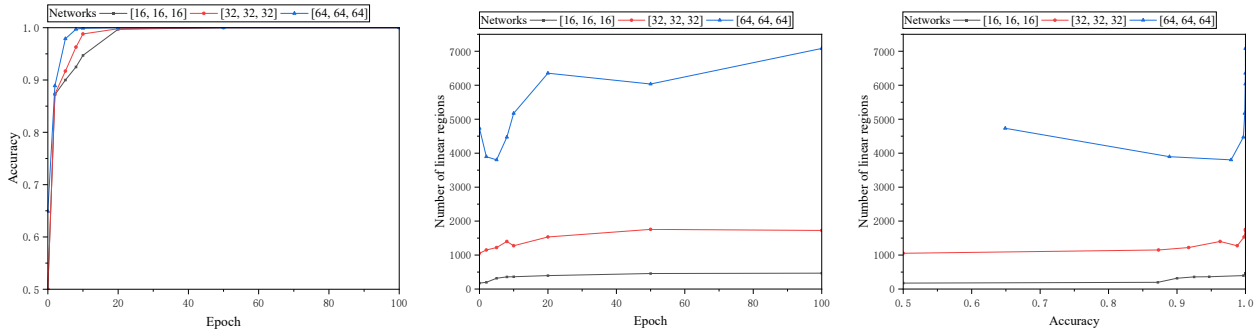



Figure 7: The relationship between the number of affine regions, accuracy and epochs in different networks for the Fig. 6 (a) dataset.

Table 6: The number of affine regions of different networks and epochs for the Fig. 6 (a) dataset.

| DNNs \ Epoch | 0 | 2 | 5 | 8 | 10 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|---|---|
| $[16, 16, 16]$ | 206 | 198 | 319 | 358 | 363 | 395 | 456 | 467 |
| $[32, 32, 32]$ | 1151 | 1151 | 1222 | 1401 | 1274 | 1533 | 1756 | 1725 |
| $[64, 64, 64]$ | 4112 | 3897 | 3803 | 4472 | 5172 | 6354 | 6037 | 7082 |

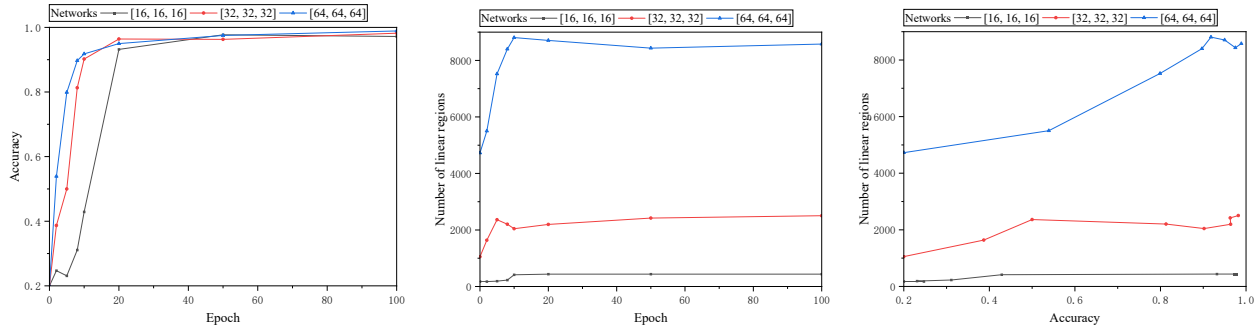
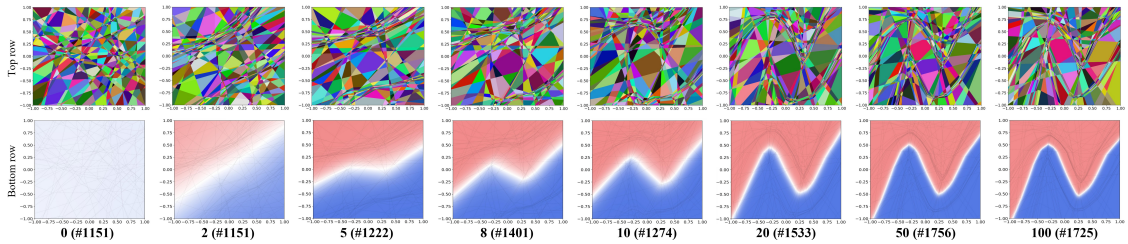


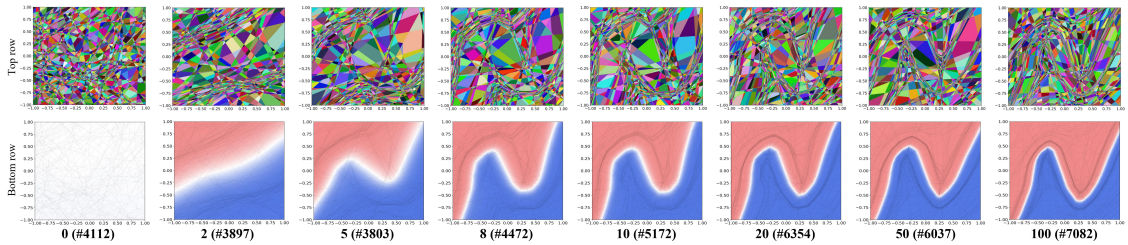

Figure 8: The relationship between the number of affine regions, accuracy and epochs in different networks for the Fig. 6 (b) dataset.

Table 7: The number of affine regions of different networks and epochs for the Fig. 6 (b) dataset

| DNNs \ Epoch | 0 | 2 | 5 | 8 | 10 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|---|---|
| $[16, 16, 16]$ | 207 | 174 | 189 | 228 | 415 | 437 | 438 | 436 |
| $[32, 32, 32]$ | 1154 | 1638 | 2362 | 2204 | 2045 | 2197 | 2420 | 2505 |
| $[64, 64, 64]$ | 4113 | 5501 | 7524 | 8402 | 8809 | 8706 | 8433 | 8579 |



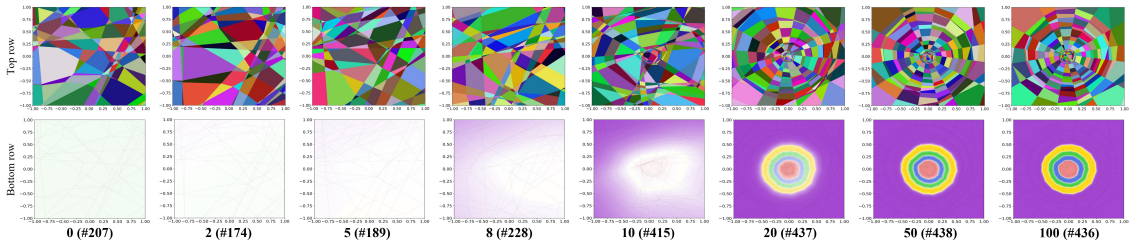

(a)



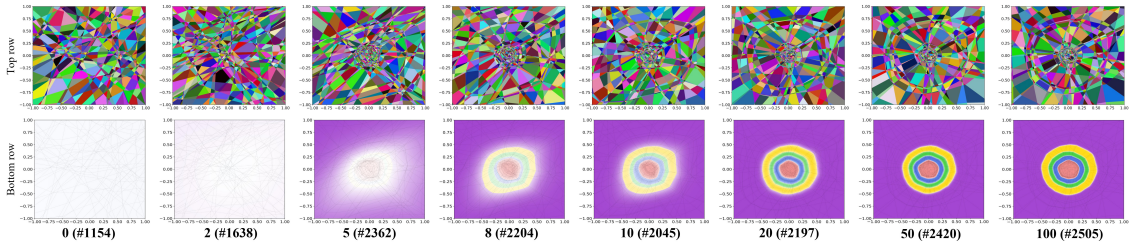

(b)

Figure 9: The visualization results and the number of affine regions of different networks trained on the Fig. 6 (a) dataset at epoch: $[0, 2, 5, 8, 10, 20, 50, 100]$. Top row: Visualization of the affine regions. Bottom row: Visualization of decision boundaries, where each color represents each type of the input data, and the white regions can be approximately regarded as the decision boundaries for the current state of the network. (a) The network $[32, 32, 32]$. (b) The network $[64, 64, 64]$.

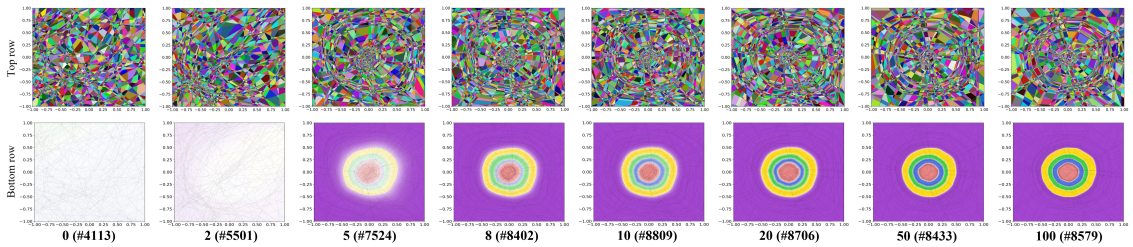
### D.3 Visualize the evolution process of decision boundaries




(a)




(b)




(c)

Figure 10: The visualization results and the number of affine regions of different networks trained on the Fig. 6 (b) dataset at epoch: $[0, 2, 5, 8, 10, 20, 50, 100]$. Top row: Visualization of the affine regions. Bottom row: Visualization of decision boundaries, where each color represents each type of the input data, and the white regions can be approximately regarded as the decision boundaries for the current state of the network. (a) The network $[16, 16, 16]$. (b) The network $[32, 32, 32]$. (c) The network $[64, 64, 64]$.