Topological Signatures of ReLU Neural Network Activation Patterns

Vicente Bosca*
Applied Math and Computational Sciences
University of Pennsylvania
Philadelphia, PA 19104
vicenteg@sas.upenn.edu

Tatum Rask*
Department of Mathematics
Colorado State University
Fort Collins, CO 80523
tatum.rask@colostate.edu

Sunia Tanweer*
Departments of Mechanical Engineering &
Computational Mathematics,
Sciences and Engineering (CMSE)
Michigan State University
East Lansing, MI 48824
tanweer1@msu.edu

Andrew R. Tawfeek*
Department of Mathematics
University of Washington
Seattle, WA 98195
atawfeek@uw.edu

Branden Stone
Georgia Tech Research Institute
Atlanta, GA 30332
branden.stone@gtri.gatech.edu

Abstract

This paper explores the topological signatures of ReLU neural network activation patterns. We consider feedforward neural networks with ReLU activation functions and analyze the polytope decomposition of the feature space induced by the network. Mainly, we investigate how the Fiedler partition of the dual graph and show that it appears to correlate with the decision boundary—in the case of binary classification. Additionally, we compute the homology of the cellular decomposition—in a regression task—to draw similar patterns in behavior between the training loss and polyhedral cell-count, as the model is trained.

1 Introduction

In recent years, neural networks have revolutionized the ability to learn from data, driving innovation across various fields [16, 18]. Despite significant study, much remains to be understood about how these models learn from data and how to effectively measure their robustness. In effort to understand how these models learn, we focus on applying topological methods to these networks motivated by the following broad question.

Question 1. Can we identify topological features that correlate with the network's performance during training?

This question has been investigated from various perspectives. For instance, [23] studies the shape of data as it passes through a ReLU network, and similarly [28] investigates this evolving topology of data to evaluate network performance. Others aim to use topological features to guide the model

*These authors have equal contribution.

Proceedings of the 1st Conference on Topology, Algebra, and Geometry in Data Science (TAG-DS 2025).

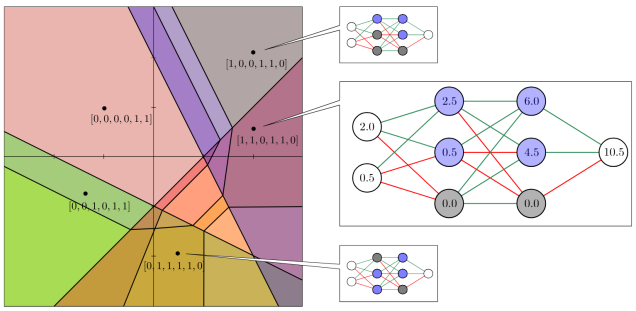

Figure 1: *Example of polyhedral decomposition of the domain of a FFRNN. Grayed out activation function outputs correspond to ReLU value of zero at that neuron, indicating a 0 entry in the binary state vector.*

during training [8]. In this paper, we focus on the polyhedral decomposition of the input space $\mathbb{R}^n$ of the neural network, as defined in [20] (see Figure 1). We investigate topological signatures of performance in feedforward neural networks for classification and regression tasks. Our key findings, based on experiments with the input space $\mathbb{R}^2$, are:

- The *weighted* Fiedler partition of the graph appears to correlate with the decision boundary for binary classification when the network exhibits grokking.
- Moments of training instability are not merely numerical artifacts, but are associated with a deeper, transient reorganization of the network's internal topological representation.

The remainder of this paper is organized as follows. In Section 2, we consider classification tasks and examine the dual graph of the polyhedral decomposition. Here, the weighted Fiedler partition of the graph is defined by the eigenvector corresponding to the smallest non-zero eigenvalue. Two experiments demonstrate the correlation with the decision boundary for binary classification when the network exhibits grokking [12, 26].

In Section 3, we analyze regression tasks and study the topological evolution of the cell complex by training a model, computing the corresponding polyhedral decomposition of the input space, and applying a random filtration to the cell complex. This allows us to study the evolution of the homology and cell count across various epochs. We find a correlation between the training loss of the model and the Betti numbers of the filtration, which similarly translates to a correlation between the f-vector (cell count) and loss. Specifically, we consider a physics-informed neural network modeling a Duffing oscillator (Appendix D), a canonical nonlinear dynamical system.

1.1 Preliminaries

The main object of study in the work is the $(L+1)-$layer feedforward ReLU neural network (FFRNN) (See Appendix A):

$$\mathbb{R}^m \xrightarrow[\text{ReLU}]{(W_1,b_1)} \mathbb{R}^{h_1} \xrightarrow[\text{ReLU}]{(W_2,b_2)} \mathbb{R}^{h_2} \to \dots \to \mathbb{R}^{h_{L-1}} \xrightarrow[\text{ReLU}]{(W_L,b_L)} \mathbb{R}^{h_L} \xrightarrow{(W_{L+1},b_{L+1})} \mathbb{R}^n. \quad (1)$$

In this model, $\mathbb{R}^m$ is the input space, $\mathbb{R}^n$ is the output space, and h_i corresponds to the number of nodes in layer i. In other words, this network has architecture $(h_0 = m, h_1, h_2, \dots, h_L, h_{L+1} = n)$.

As detailed in [20], FFRNNs inherently define *binary state vectors* that create a polyhedral decomposition of the domain (Figure 1). More explicitly, consider the model (1) above. Given an input data point $x \in \mathbb{R}^m$ in the input space, we have for each hidden layer i (so $1 \le i \le L$) a binary (bit) indicating whether that neuron is "on" or "off," which we may assemble into a vector

$$s_i(x) = [s_{i,1}(x) \ \dots \ s_{i,h_i}(x)]^\top \in \mathbb{R}^{h_i},$$

where $s_{i,j}(x)$ (with $1 \le j \le h_i$) is defined as follows:

$$s_{i,j}(x) = \begin{cases} 1 & \text{if } w_{i,j}F_{i-1}(x) + b_{i,j} > 0 \\ 0 & \text{if } w_{i,j}F_{i-1}(x) + b_{i,j} \le 0. \end{cases} \quad (2)$$

Thus, for each point $x \in \mathbb{R}^m$ in the input space, we have a sequence of binary vectors $s_1(x), s_2(x), \dots, s_L(x)$. We can stack the binary vectors associated to x to make a long column

vector

$$s(x) = [s_1^\top(x) \ \dots \ s_L^\top(x)]^\top \in \mathbb{R}^h, \tag{3}$$

where $h = \sum_{i=1}^{L} h_i$ is the total number of nodes in the hidden layers. We call $s(x)$ the binary vector of x. These binary vectors create a polyhedral decomposition in the input space such that all regions associated with the same binary vector form one polytope (Figure 1).

2 Classification on the dual graph

In Liu et al [20], the authors utilize the *dual graph* of the decomposition induced by the bit vectors to run persistent homology. Inspired by their work, we propose this dual graph as a useful tool for studying the topology of a neural network. Namely, we will use (a weighted version of) the *graph Laplacian* [15, 6, 14] to analyze the geometry of neural networks for classification problems.

We are not the first to use discrete Laplacians to analyze and study neural networks and their use in classification tasks. For example, [13] applies the Fiedler vector partition to "representational dissimilarity matrices". However, to our knowledge, we are the first to consider the graph Laplacian in this specific setting.

2.1 Graph Laplacian

Let $G = (E, V)$ be a graph with vertex set $V = \{v_j\}_j$ and edge set $E \subseteq V \times V$. First, construct the coboundary matrix $\partial : \mathbb{R}^{|V|} \to \mathbb{R}^{|E|}$ as follows: suppose edge i is given by $e_i = (s, t)$. Then, $\partial(i, j) = [v_j : e_i]$, where

$$[v_j : e_i] = \begin{cases} 1 & \text{if } v_j = t, \\ -1 & \text{if } v_j = s, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the *graph Laplacian* is defined as $L = \partial^T \partial$ [14].

There is vast literature on the graph Laplacian and its theory and uses in applications; however, for the purpose of this work, we will focus on certain spectral properties of the graph (see [4] for a more complete discussion). For one, $\dim(\ker L)$ counts the number of connected components of G. In our case, $\dim(\ker L) = 1$. The smallest non-zero eigenvalue, call it λ_1, is often called the *algebraic connectivity* of G [6]. The associated eigenvector, called the *Fiedler vector*, has entries that are guaranteed to sum to 0. This makes the Fiedler vector easily amenable to binary partitions: nodes are classified based on whether they are assigned a positive or negative value. It is known that this partition approximates a minimum cut on a graph [4, Equation 7.12]. For shorthand, we will refer to this partition as the *Fiedler partition*.

The definition of the graph Laplacian above implicitly assumes that our vertex and edge sets form orthonormal bases. By changing the inner product on our edge and vertex spaces, we inherently change the Laplacian. Specifically, we can *weight* our edges and vertices to produce a new inner product. Denote by W_V the diagonal matrix where $W_V(i, i)$ is given by the weight of the ith vertex. Construct W_E similarly. Then, the weighted Laplacian is given by $L = W_V^{-1} \partial^T W_E \partial$.

2.2 Dual graph for polyhedral decomposition

Now, we will construct a graph $G = (V, E)$ from a FFRNN with architecture given in Equation 1. Let $V = \{s(x)\}$ obtained from Equation 3, and let E be given by pairs of vertices (i.e. binary vectors) that differ in exactly one entry. That is, pairs of binary vectors that have *Hamming distance* of 1 [20]. See Figure 3 for an illustration of the dual graph. We aim to answer the question:

Question 2. Does the Fiedler vector partition the dual graph to reflect the decision boundary of a binary classification problem? If not, can we adjust the weights so that it does?

In summary, we found that the partition of the dual graph by the Fiedler vector from the unweighted Laplacian is not accurate. In the paragraphs to follow, we propose a choice of vertex weights so that the Fiedler partition from the *weighted* Laplacian more accurately reflects class labels for binary classification tasks.

Table 1: *Experimental results testing the weighted and unweighted Fiedler partitions. The signs of the entries of the Fiedler vector are used to predict class labels. Zero misclassification and L_2 error means that the Fiedler partition perfectly matches the class labels.*

		Loss		Unweighted		Weighted	
Dataset	Architecture	Train	Test	Missclass. (%)	L_2 Error	Missclass. (%)	L_2 Error
Circles	$(2, 6, 6, 2)$	0.00002	0.00016	19.05%	2	0%	0
Moons	$(2, 5, 5, 5, 2)$	0.00002	0.00001	10%	1.20	0%	0.34

Place the following weights on the nodes of the dual graph G: let $W_V(i, i)$ equal one plus the number of training data points contained in polytope i (we use a function in the `GoL_Toolbox` to obtain this count [19, 3]). Let W_E be the identity matrix. Experimental results on small examples (see Section 2.2.1) demonstrate that this choice of weights will, indeed, reflect the decision boundary of a network trained on a binary classification problem *that demonstrates grokking*. Grokking, also known as *delayed generalization* [26], geometrically manifests as an increase in the polytope regions around training points once the network reaches negligible training error. At the same time, there is a greater concentration of (smaller) polytopes around decision boundaries [12]. These smaller polytopes concentrated near the boundary will, then, be assigned a smaller weight in the dual graph than those polytopes centered near training data. Thus, we expect that the weighted Fiedler vector will "cut" near the decision boundary.

Although our small experiments show success, we have yet to prove that this weighted Laplacian does, indeed, approximate a (weighted) minimum cut. Future work includes proving such a statement directly or utilizing existing results for edge-weighted [7] and vertex-weighted [29] Laplacians.

2.2.1 Experiments

We ran two small experiments, following the same basic steps, to test the weighted Fiedler partition.

1. Train a FFRNN with varying architectures and epoch counts to perform binary classification.
2. Compute the polytope decomposition of the input space using `GoL_Toolbox` [3].
3. Compute and visualize the dual graph of the polytope decomposition.
4. Calculate the Fiedler partitions for both the unweighted Laplacian and the training point-weighted Laplacian.
5. Evaluate the accuracy of the Fiedler partition by:
   (a) Computing the average class label within each polytope.
   (b) Assessing accuracy via (1) proportion of polytopes with incorrect class identification, and (2) L_2 error (defined as the L_2 norm of the vector of differences between average class labels and predicted values).

Two circles experiment As seen in Figure 2, our Two Circles data contains points sampled along two concentric circle. We trained a FFRNN with 2 hidden layers (both of width 6) to learn the two circles binary classification task to 4000 epochs. As shown in Table 1, model obtained 100% accuracy on both the training and test datasets, with minimal loss on both training and testing data. The unweighted Fiedler vector missclassified 19.05% of the polytopes with an L_2 error of 2. On the other hand, the weighted Fiedler vector missclassified 0% of the polytopes with an L_2 error of 0.

Two moons experiment Figure 2 outlines the experiment with data sampled along two distinct arcs. Here, we trained a FFRNN with 3 hidden layers (all of width 5) to learn the two moons binary classification task to 2000 epochs. As expected, the model obtained 100% accuracy on both the training and test datasets with minimal loss. The unweighted Fiedler vector missclassified 10% of the polytopes with an L_2 error of 1.20. On the other hand, the weighted Fiedler vector missclassified 0% of the polytopes with an L_2 error of 0.34.

Because of the success of these small examples, we believe that the weighted Fiedler partition serves as a decent proxy for whether or not a network has achieved grokking. That is, the dual graph and Fiedler partition offer an approach for analyzing decision boundaries and classification structure on

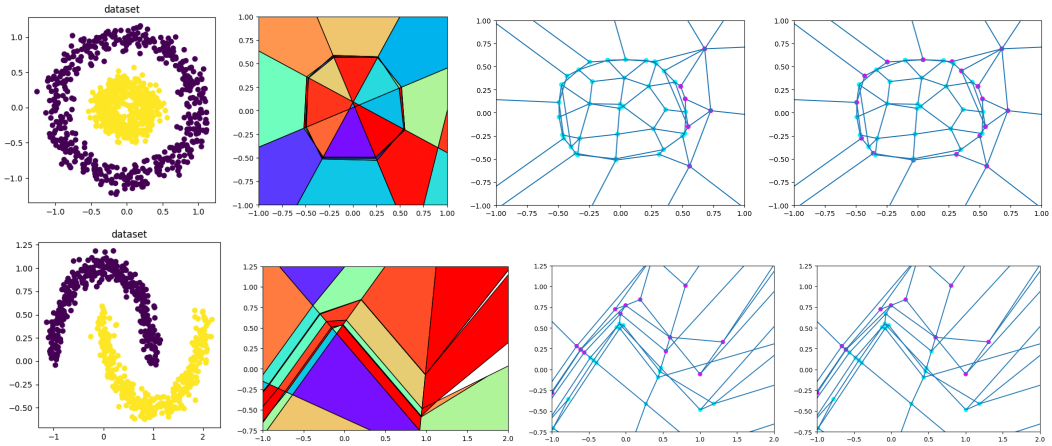

Figure 2: *The two circles and two moons datasets (left column), their respective polytope decompositions (middle left), and the partitions of the dual graph by the unweighted (middle right) and weighted Fiedler vectors (right), indicated by the blue and purple points. Minimal loss and concentration along decision boundary demonstrates grokking behavior.*

well-trained networks; for complementary insights into the temporal evolution of network geometry, we next explore homological analysis of the full cell complex.

3 Topological analysis via cell complex homology

Beyond the dual graph's focus on polytope adjacencies, we can alternatively analyze the full polyhedral decomposition as a cell complex (Appendix C). This approach captures the hierarchical relationships between polytopes and their boundaries, enabling us to apply homological tools to track topological features that emerge and persist during training.

3.1 Computing homology via random filtration

To analyze the topological evolution of this cell complex during training, we compute the Betti numbers β_i [11] of the polyhedral decomposition at each training epoch. β_0 counts connected components, β_1 counts loops, β_2 counts voids, and higher-dimensional Betti numbers capture corresponding topological features. The examples presented here consider a bounded subset of $\mathbb{R}^2$ as our input space, so only β_0 and β_1 are non-trivial.

If we were to compute the Betti numbers of the polyhedral decomposition of the input space directly, we would just recover the Betti numbers of the input space (for a 2D square $\beta_0 = 1$ and $\beta_1 = 0$). To reveal a deeper topological structure, we add cells progressively and compute the Betti numbers at each step. The way we add these cells is called a *filtration*, a sequence of nested subcomplexes $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_T$ with $K_0 = \emptyset$ and $K_T = K$. We then compute the *Betti curves* as $\beta_i(t) = \dim H_i(K_t)$, for $i = 0, 1$ and $t \in [0, T]$, where t denotes the stage of our filtration. Specifically, we implement a *random filtration* approach (see Figure 3 as an example):

1. Begin adding the 0-cells (vertices) in the complex, selected uniformly at random
2. At each timestep, randomly select and add one 1-cell (edge) from the remaining edges
3. Once all 1-cells are added, proceed to randomly add 2-cells (faces)
4. Continue this process for higher-dimensional cells

This filtration allows us to track how topological features emerge and persist as we reconstruct the complex. As edges are added, cycles begin to form (increasing β_1), and different connected components start to merge (decreasing β_0), which are subsequently filled when the corresponding faces are added (decreasing β_1). The Betti curves for this process are shown in Figure 4.

To ensure robustness against the randomness in cell ordering, we perform multiple random trials and average the resulting Betti curves. The filtration parameter is expressed as the percentage of total cells added, enabling comparison across different epochs despite varying complex sizes.

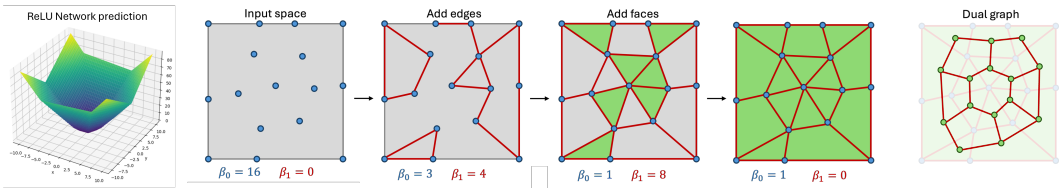

Figure 3: *(Left) Prediction of a FFRNN trained to learn a paraboloid surface. (Middle) Snapshots of a random filtration on the primal complex of the polyhedral decomposition of such a network with their corresponding Betti numbers. First, randomly add all the vertices of the decomposition (blue), then add the edges (red, cycles are born), and finally add the faces (green, cycles are closed). (Right) Dual graph of the polyhedral decomposition: each polytope is now a vertex, and join two vertices if their corresponding polytopes share a face.*

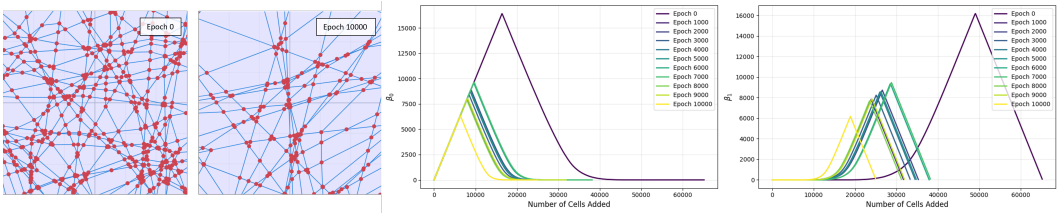

Figure 4: *(Left) Snapshots (zoomed in to the square $[0.3, 0.3]^2$) of the polyhedral decomposition of a PINN at the beginning (epoch 0) and end of training (epoch 10000). In red are the vertices (0-cells), in solid blue the edges (1-cells), and in a lighter blue the faces (2-cells). More cells are observed at the beginning of training. This number seems to decrease as we train, with the emergence of some structure. (Right) Corresponding average Betti curves, β_0 and β_1, of each stage of training for a random filtration of the resulting complex.*

Given the computational complexity of directly computing homology from boundary matrices (which can involve thousands of cells per dimension for medium-sized networks), we employ a multi-step computational pipeline. First, we use edge subdivision [2] to compute the sign vectors of the polyhedral decomposition. Then, we apply the perturbation method described in the same work to construct the full cell complex structure from these sign vectors. Finally, we use the `PHAT` package [1] to efficiently compute the homology with $\mathbb{Z}_2$ coefficients of the resulting complex under our chosen filtration.

3.2 Visualizing topological evolution during training

For the following results, we used a regression problem from a Physics-Informed Neural Network (PINN); the details can be found in Appendix D. For each training epoch, we extract the polyhedral decomposition and compute averaged Betti curves, which we visualize in two complementary ways to reveal different aspects of the topological evolution.[2].

Betti curves Figure 4 shows the average Betti curves themselves, with the x-axis representing the number of cells added (across all dimensions) and the y-axis showing the Betti number value. For β_0 in 2D input spaces, the curve begins at the origin and increases linearly (with slope 1) as vertices are added, reaching its peak when all vertices are included. As edges are subsequently added, β_0 initially decreases approximately linearly since each edge typically merges two connected components, although this decrease slows as the complex becomes more connected, ultimately converging to 1 (a single connected component).

The behavior of β_1 follows a complementary pattern. Starting at zero, it remains flat until sufficient edges create the first loops, then increases, eventually growing linearly as most new edges create additional cycles. The peak occurs when all edges are added, with the peak value equaling the number of polytopes that remain to be filled. As 2-cells (faces) are added in 2D spaces, β_1 decreases linearly (with slope -1) until reaching zero, when all faces have filled the loops.

[2]These experiments were conducted on a system equipped with an Intel Xeon Silver 4214R CPU, running at a base clock speed of 2.40 GHz. The system has a total of 24 CPUs having 700 KB L1 cache, 12 MB L2 cache, and the system as a whole has 16.5 MB of shared L3 cache.

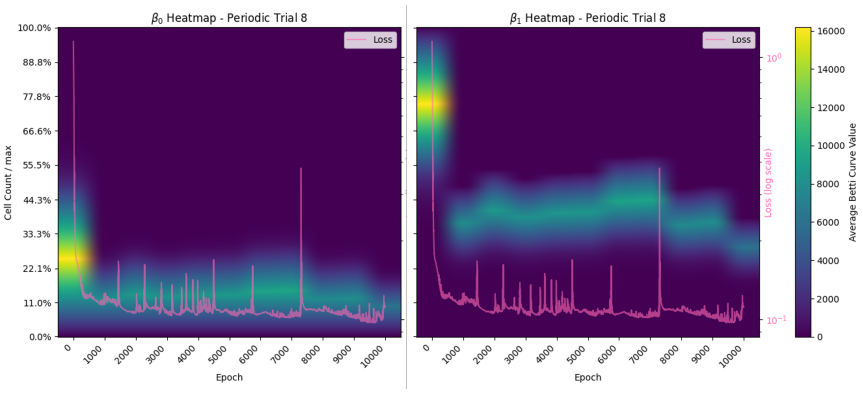

Figure 5: *Heat maps of the Betti curves corresponding to a trained PINN, with the training loss overlaid. (Left) For β_0 we observe smoother changes. (Right) For β_1, the changes are sharper and correlate with substantial changes in the training loss function.*

The peaks of our Betti curves directly reveal the *f-vector* of the polyhedral decomposition, defined as $f(K) = (f_0(K), f_1(K), \ldots, f_{\dim K}(K))$ where $f_i(K)$ denotes the number of i-cells in K. Specifically, $\max(\beta_0) = f_0$ (the number of vertices) and $\max(\beta_1) = f_2$ (the number of 2-cells) when all edges but no faces have been added. This provides a computationally efficient alternative to full homology computation: while computing Betti numbers becomes intractable for networks with thousands of cells per dimension, the f-vector can be extracted directly from the cell complex construction. Note that the f-vector components are constrained by the Euler characteristic of the domain space. For our 2D square, $\chi = f_0 - f_1 + f_2 = 1$, which together with the fact that $f_i \geq 0$ by definition implies that $f_0 \sim f_2 \sim \frac{1}{2} f_1$, which is experimentally verified in Figure 6.

Heat maps The second visualization (Figure 5) normalizes and compresses this topological information into heat maps, easing comparison across training epochs. In these heat maps, the x-axis represents training epochs, the y-axis shows the number of cells added divided by the maximum number of cells across all epochs (usually achieved at the early stages of training). The color encodes the average Betti curve value at that filtration percentage. Each vertical slice of the heat map essentially represents a compressed version of the full Betti curve from that epoch, with color encoding what would be the y-axis value in the Betti curve plot. We then stack them next to each other to visualize how the values change relative to the maximum number of cells for all epochs.

3.3 Geometric Reorganization and Correlation between topological complexity and loss

During the training process, while a general trend of decreasing loss was observed, characteristic "spiking" behavior can be noted in the training loss curve, see Figure 6 (left). This phenomenon, where the loss exhibits sudden, transient increases before returning to a downward trend, is not uncommon in neural network training, and is known to be connected to complexities in the loss landscape, training dynamics and the edge-of-stability phenomenon [17].

A key finding from our analysis of the training dynamics relates the topological properties of the polyhedral decomposition with the loss. We computed Betti vectors for the decomposition at each training epoch. The filtration parameter for this topological analysis was defined as the percentage of randomly added complexes, providing a measure of the network's connectivity and structure. A notable observation was the correlation between the training loss and the filtration value at which the maximum Betti number β_{max}, was achieved. Specifically, we found that sudden "spikes" in the training loss corresponded to an increase in this critical filtration value (Figure 5). This suggests that during epochs where the optimizer temporarily struggles to minimize the loss, the internal topological structure of the network becomes more complex, requiring a higher filtration threshold to reveal its most significant persistent homology features. The observed relationship indicates that moments of training instability are not merely numerical artifacts, but are associated with a deeper, transient reorganization of the network's internal representation. This result suggests that the loss landscape's complexity is directly reflected in the evolving homology of the network's decomposition, potentially

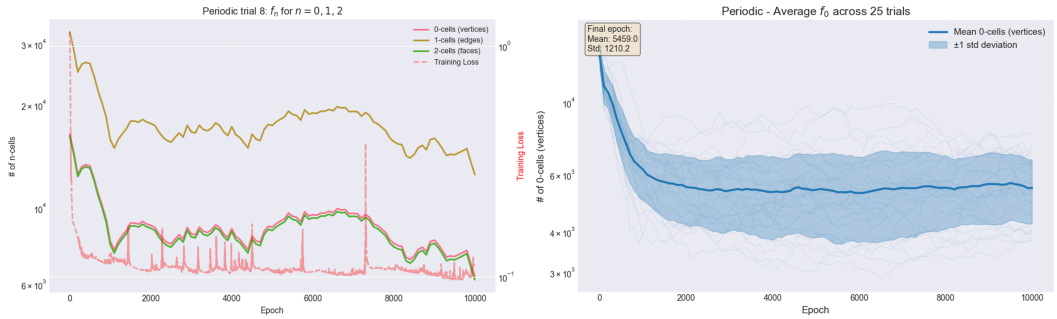

Figure 6: *(Left) The three components of the f-vector across training epochs for a trained PINN with the training loss overlaid. Spikes in the training loss lead to changes in those quantities. (Right) The average f_0 (# of 0-cells) for the 25 trials of training a neural network using the periodic dynamical system. Overall, the number of cells decreases along training.*

allowing us to use the homology as a proxy for the loss without requiring access to training data—an endeavor left for future work.

A deeper examination of the heat maps in Figure 5 reveals a fundamental geometric reorganization throughout the training process. The downward shift of peak Betti values in early epochs—from approximately 25% to 11% for β_0 and from 75% to 35% for β_1—indicates substantial evolution in the number of cells across different dimensions. Considering only the max of β_0 and β_1 (brightest region in the plot), we see an initial decrease with respect to the early stages of training—reflecting a reduction in the total number of cells. Subsequent fluctuations align closely with abrupt variations in the training loss. This phenomenon is especially evident in Figure 4. Immediately prior to the final big spike in loss, the number of cells tend to increase, followed by a decrease again. This is better captured in the left of Figure 6, where we plot the number of cells per epoch. The "reorganizing" behavior after a big training loss spike was observed consistently across trials. Nevertheless, the overarching trend for all trials (shown as average in right panel of Figure 6) is reduction in the total number of cells.

Remark. The heat maps in Figure 5 normalize Betti curves by the maximum number of cells across all training epochs rather than within each epoch. For 2D bounded input spaces, the Euler characteristic constraint ($\chi = 1$) creates dependencies: approximately equal number of 0-cells and 2-cells, with roughly twice as many 1-cells. Under our random filtration, β_0 peaks at 25% (all 0-cells added) and β_1 at 75% (all 1-cells added), visible as horizontal bands in early epochs where the maximum cell count occurs. While this normalization effectively tracks the evolution of cell counts across dimensions due to these constraints, the heat maps capture more than just the f-vector evolution. For example, in the middle and left figures in Figure 4 before 20,000 cells have been added, we observe non-trivial topological behavior where adding 1-cells does not immediately reduce the number of connected components, indicating complex geometric behavior beyond simple cell enumeration. A within-epoch normalization would yield uninformative horizontal lines at fixed percentages due to the 2D constraints, missing these subtle topological transitions.

To quantify the relationship between topology and training dynamics, we compute different statistical correlations, reported in Tables 2 and 3. The analysis confirms a moderate-to-strong positive correlation between topological complexity and training loss. This correlation is robust to detrending, indicating that the relationship is not merely an artifact of shared temporal trends. The correlation of first differences further confirms that changes in topology coincide with changes in loss, strengthening the dynamic relationship interpretation. Cross-correlation analysis reveals the strongest relationship at a lag of +25 and +100 epochs for the chaotic and periodic dynamics systems, respectively, suggesting that geometric reorganization of the network's polyhedral decomposition *precedes* shifts in the loss landscape. This temporal ordering not only supports the hypothesis that topological complexity serves as a leading indicator of optimization dynamics, but also highlights its potential as a predictive diagnostic for abrupt changes in training dynamics. This can prove to be very significant for understanding, monitoring and potentially controlling the learning process of neural networks.

Table 2: *Main correlation results (means $\pm$ std) across $n = 25$ trials. For each method we report mean correlation ($\bar{r}\pm$ std), mean p-value ($\bar{p}\pm$ std) and number of trials with $p < 0.05$. "Max cross-corr" shows the lag (in epochs) at which the mean cross-correlation is maximal and the corresponding statistics.*

	Chaotic (n=25)			**Periodic** (n=25)		
Method	$\bar{r} \pm \sigma$	$\bar{p} \pm \sigma$	Sig.	$\bar{r} \pm \sigma$	$\bar{p} \pm \sigma$	Sig.
Raw Pearson (Loss vs 0-cells)	0.69 ± 0.13	$< 10^{-6}$	25/25	0.61 ± 0.09	$< 10^{-6}$	25/25
Detrended (linear)	0.82 ± 0.11 lag = 25	$< 10^{-6}$	25/25	0.61 ± 0.10 lag = 100	$< 10^{-6}$	25/25
Max cross-corr (lag)	0.70 ± 0.13	$< 10^{-6}$	25/25	0.74 ± 0.14	0.004 ± 0.011	25/25

4 Limitations and future directions

The major limitation of these methods is the computational complexity of the algorithms being employed. Scaling these tools to more complex neural architectures presents several challenges since the number of polytopes grows polynomially with network width and exponentially with depth [27, 10]—while also scaling exponentially with input dimension. This exponential growth makes direct application of these methods computationally prohibitive for deep networks or high-dimensional inputs—with the current algorithms. Likewise, computing eigenvalues and eigenvectors of the graph Laplacian is also computational expensive.

For future work, potential mitigation strategies could involve projecting the activation patterns onto principal components, thus reducing the effective dimensionality while preserving essential topological characteristics. Additionally, approximation algorithms for homology computation [25] or sampling-based approaches could enable analysis of larger networks while sacrificing some precision in the topological measurements. Implementing algorithms and methods for quicker computation is also a worthwhile endeavour—for example, the TRACEMIN-Fiedler algorithm [21] for Fiedler-vector computation. Further future work involves establishing theoretical justification of these empirical conclusions, exploring similar methods for multiclass classification problems, studying the topological invariance of the Euler characteristic, and investigating how the polyhedral decomposition evolves during training near decision boundaries using geometrically-informed filtrations, rather than the random filtrations employed in this work. In addition, as a reviewer noted, the relationship between grokking and larger polytopes suggests a potential link to circuit formation [24], wherein larger polytopes correspond to "cleaner", less complex circuits.

Another promising direction for future endevers involves extending our topological analysis to classification tasks. Recent work, [22] demonstrates that the dual complex of ReLU networks forms a cubical complex, enabling the use of GUDHI. Building on this framework, we can investigate how the polyhedral decomposition evolves during training near decision boundaries using geometrically-informed filtrations. By tracking polytope birth and death through "boundary-aware" persistent homology across training epochs, we may characterize topological signatures of learning phenomena such as grokking, where networks transition from memorization to generalization. This temporal topological analysis could inform training algorithms that explicitly consider geometric complexity during optimization. More about the scalability issues can be found in Appendix E.

5 Summary

In this work, we explored topological methods to analyze FFRNN behavior by examining the polyhedral decomposition of the input space $\mathbb{R}^2$. We demonstrated that the weighted Fiedler partition correlates with the decision boundary during grokking and that Betti numbers of the filtration correlate with training loss. These findings emphasize the importance of geometric and topological structure over purely algebraic properties, potentially informing more effective neural network architectures and training methods.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Georgia Tech Research Institute (GTRI) through their summer research internship program (GRIP). The authors thank GTRI for its support.

References

[1] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017. Algorithms and Software for Computational Topology.

[2] Arturs Berzins. Polyhedral complex extraction from relu networks using edge subdivision, 2023.

[3] Turgay Caglar. Gol toolbox. Accessed: 31 July 2025.

[4] Dragoš Cvetković, Peter Rowlinson, and Slobodan Simić. *Laplacians*, page 184–227. London Mathematical Society Student Texts. Cambridge University Press, 2009.

[5] Amer Farea, Olli Yli-Harja, and Frank Emmert-Streib. Understanding physics-informed neural networks: Techniques, applications, trends, and challenges. *AI*, 5(3):1534–1557, August 2024.

[6] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[7] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

[8] Rickard Brüel Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1553–1563. PMLR, 26–28 Aug 2020.

[9] Sai Ganga and Ziya Uddin. Exploring physics-informed neural networks: From fundamentals to applications in complex systems. 2024.

[10] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. *International Conference on Machine Learning*, pages 2596–2604, 2019.

[11] Allen Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2001.

[12] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep networks always grok and here is why. In *High-dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, 2024.

[13] Huma Jamil, Yajing Liu, Turgay Caglar, Christina Cole, Nathaniel Blanchard, Christopher Peterson, and Michael Kirby. Hamming similarity and graph laplacians for class partitioning and adversarial image detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 590–599, June 2023.

[14] William N. Anderson Jr. and Thomas D. Morley. Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra*, 18(2):141–145, 1985.

[15] Gustav R. Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148:497–508, 1847.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[17] Xiaolong Li, Zhi-Qin John Xu, and Zhongwang Zhang. Loss spike in training neural networks. 2023.

[18] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[19] Yajing Liu, Turgay Caglar, Christopher Peterson, and Michael Kirby. Integrating geometries of relu feedforward neural networks. *Frontiers in Big Data*, Volume 6, 2023.

[20] Yajing Liu, Christina M Cole, Chris Peterson, and Michael Kirby. Relu neural networks, polyhedral decompositions, and persistent homology. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 455–468. PMLR, 2023.

[21] Murat Manguoglu, Eric Cox, Faisal Saied, and Ahmed Sameh. Tracemin-fiedler: A parallel algorithm for computing the fiedler vector. In José M. Laginha M. Palma, Michel Daydé, Osni Marques, and João Correia Lopes, editors, *High Performance Computing for Computational Science – VECPAR 2010*, pages 449–455, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[22] Marissa Masden. Algorithmic determination of the combinatorial structure of the linear regions of relu neural networks. *SIAM Journal on Applied Algebra and Geometry*, 9(2):374–404, 2025.

[23] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *Journal of Machine Learning Research*, 21(184):1–40, 2020.

[24] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. *URL https://arxiv. org/abs/2301.05217*, 2023.

[25] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, 2017.

[26] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

[27] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *International conference on machine learning*, pages 2847–2854, 2017.

[28] Matthew Wheeler, Jose Bouza, and Peter Bubenik. Activation landscapes as a topological summary of neural network performance. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3865–3870, 2021.

[29] Shijie Xu, Jiayan Fang, and Xiangyang Li. Weighted laplacian method and its theoretical applications. *IOP Conference Series: Materials Science and Engineering*, 768(7):072032, March 2020.

A Feed Forward Networks

Consider an $(L+1)-$layer feedforward ReLU neural network (FFRNN):

$$\mathbb{R}^m \xrightarrow[\text{ReLU}]{(W_1,b_1)} \mathbb{R}^{h_1} \xrightarrow[\text{ReLU}]{(W_2,b_2)} \mathbb{R}^{h_2} \to \ldots \to \mathbb{R}^{h_{L-1}} \xrightarrow[\text{ReLU}]{(W_L,b_L)} \mathbb{R}^{h_L} \xrightarrow{(W_{L+1},b_{L+1})} \mathbb{R}^n. \quad (4)$$

In this model, $\mathbb{R}^m$ is the input space, $\mathbb{R}^n$ is the output space, and h_i corresponds to the number of nodes at layer i. In other words, this network has architecture $(h_0 = m, h_1, h_2, \ldots, h_L, h_{L+1} = n)$.

We let $W_i \in \text{Mat}(h_i \times h_{i-1},\ \mathbb{R})$ and $b_i \in \mathbb{R}^{h_i}$ denote the weight matrix and bias vector of layer i, respectively. The activation functions for the hidden layers (layers $1, \ldots, L$) are assumed to be ReLU functions (applied coordinate-wise) while the map to the last layer (the output layer) is assumed to be affine linear (without a ReLU function being applied to the image). Recall that the ReLU function is the piecewise linear and continuous map $\text{ReLU} : \mathbb{R} \to \mathbb{R}$ given by

$$\text{ReLU}(a) = \max\{0, a\}.$$

It can be naturally extended to a piecewise linear continuous map on vector spaces (which we also denote as ReLU). More precisely, we define $\text{ReLU} : \mathbb{R}^{h_i} \to \mathbb{R}^{h_i}$, by applying the ReLU function to each coordinate of $x \in \mathbb{R}^{h_i}$. Given an input data point $x \in \mathbb{R}^m$, we denote the output of x in layer i as $F_i(x)$. Thus we have that $F_0(x) = x$ and

$$F_i(x) = \text{ReLU}(W_i F_{i-1}(x) + b_i) = \begin{bmatrix} \max\{0, w_{i,1} F_{i-1}(x) + b_{i,1}\} \\ \vdots \\ \max\{0, w_{i,h_i} F_{i-1}(x) + b_{i,h_i}\} \end{bmatrix} \in \mathbb{R}^{h_i}, \quad (5)$$

which expresses the value of each neuron on the ith layer of the network.

B Polyhedral Decomposition of Neural Networks

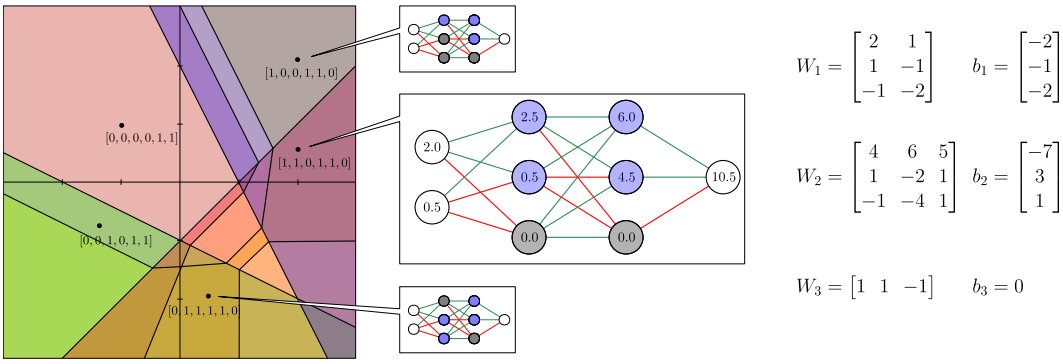

Figure 7: *The polyhedral decomposition of the input space $\mathbb{R}^2$ of a small FFNN having architecture $(2, 3, 3, 1)$. Some of the binary vectors for various points in $\mathbb{R}^2$ are highlighted and their neural activation pattern within the network is illustrated.*

C Cell Complex Structure

The polyhedral decomposition of a FFRNN naturally induces a cell complex structure K where:

- d-dimensional cells are the polytopes themselves (for input dimension d),
- $(d-1)$-dimensional cells are the facets (boundaries between adjacent polytopes),
- lower-dimensional cells correspond to intersections of hyperplanes, and
- 0-dimensional cells are the vertices where multiple hyperplanes meet.

This hierarchy can be encoded using *sign vectors*, a generalization of the binary vectors introduced in Section 1.1. For a polytope $\mathcal{P}$, its sign vector is obtained by replacing zeros with -1 in the binary vector. Thus, a -1 in the sign vector means we are in the inactive part of the ReLU nonlinearity, and a $+1$ means we are in the active part. Apart from -1 and $+1$, the sign vector also has 0s for those points in the input space that lie on the boundary of different regions. That is where the input of the ReLU maps is exactly zero, and corresponds to the lower-dimensional cells in the cell-complex. For full-dimensional polytopes, the sign vector only contains -1 or $+1$. For lower-dimensional cells, we introduce zeros at positions corresponding to neurons whose decision boundaries contain the cell. Specifically, a $(d-k)$-dimensional cell lying in the intersection of k hyperplanes will have exactly k zeros in its sign vector, with the remaining entries indicating which side of each hyperplane the cell lies on.

C.1 Additional correlation measurements

Table 3: *Correlation robustness checks (First differences and Spearman rank). Values are means $\pm$ std across $n = 25$ trials; "Sig." counts trials with $p < 0.05$.*

	Chaotic (n=25)			**Periodic** (n=25)		
Method	$\bar{r} \pm \sigma$	$\bar{p} \pm \sigma$	Sig.	$\bar{r} \pm \sigma$	$\bar{p} \pm \sigma$	Sig.
First differences	0.88 ± 0.14	0.000 ± 0.000	25/25	0.58 ± 0.31	0.038 ± 0.128	23/25
Spearman rank	0.29 ± 0.20	0.120 ± 0.251	20/25	0.13 ± 0.22	0.205 ± 0.290	15/25

D Physics-Informed Neural Network

We designed a Physics-Informed Neural Network (PINN) [5, 9] to learn the dynamics of the Duffing oscillator by predicting the system's displacement at the next time step, $\hat{x}(t + \Delta t)$, given the current time t and current displacement $x(t)$. The network's architecture is a feedforward neural network such that: The input layer accepts a concatenated vector of $[t, x(t)]$. This input is then processed through

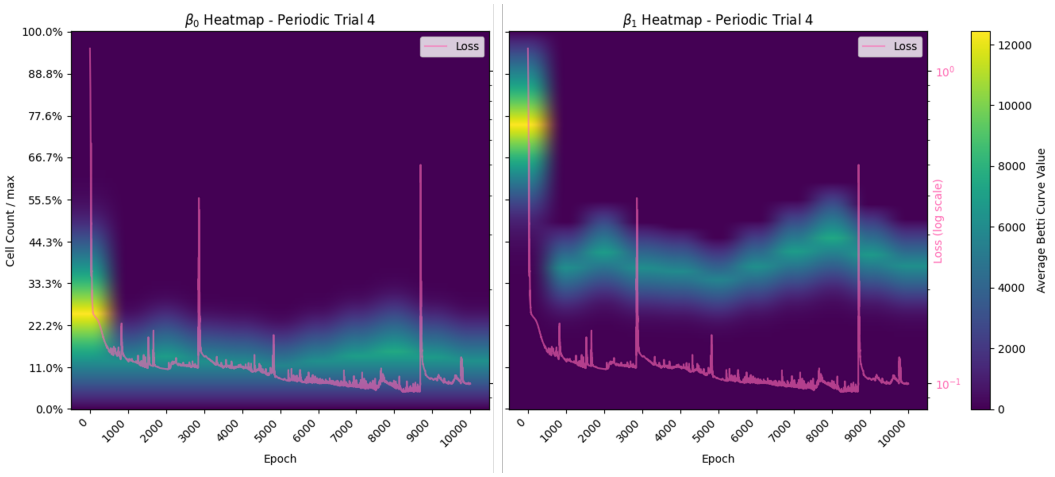

Figure 8: *Heat maps of the betti curves for Trial 4 corresponding to a trained PINN, with the training loss overlaid. (Left) For β_0 we observe smoother changes. (Right) For β_1, the changes are sharper and correlate with substantial changes in the training loss function.*

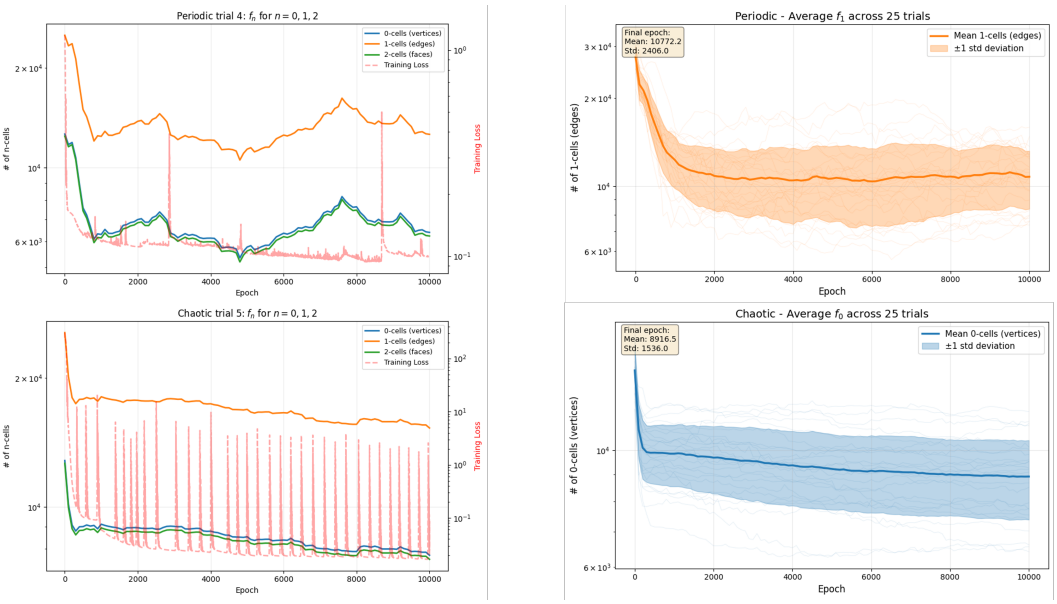

Figure 9: *(Top left) The f-vector of another trial of the periodic dynamical system. (Top right) The averaged f_1 (instead of f_0) however these values are related through the Euler Characteristic. (Bottom left) is the f-vector of a chaotic dynamical system. (Bottom right) The averaged f_0 for the 25 trials of the chaotic system.*

a fully connected layer with ReLU activation, transforming the input into a higher-dimensional representation with 50 neurons. Following this, the model incorporates three consecutive linear transformations followed by a ReLU activation. Finally, a linear output layer maps the output of the last hidden layer to a single scalar, representing the predicted displacement at the next time step, $\hat{x}(t + \Delta t)$.

D.1 Duffing Oscillator

The Duffing oscillator is a canonical nonlinear dynamical system described by the second-order ordinary differential equation:

$$\frac{d^2x}{dt^2} + \delta\frac{dx}{dt} + \alpha x + \beta x^3 = \gamma\cos(\omega t)$$

For our experiments, the parameters were set to $\delta = 0.0$, $\alpha = -1.0$, $\beta = 1.0$, $\gamma = 0.0$, and $\omega = 1.2$ to have a periodic regime, see Fig. 10. To generate training data, this equation was transformed into a system of first-order ODEs:

$$\frac{dx}{dt} = v \qquad \frac{dv}{dt} = -\alpha x - \beta x3$$

This system was numerically solved with initial conditions $x(0) = 0$ and $v(0) = 1$ over a time interval of $t \in [0, 20]$. The resulting time series data for $x(t)$ served as the ground truth for training the PINN.

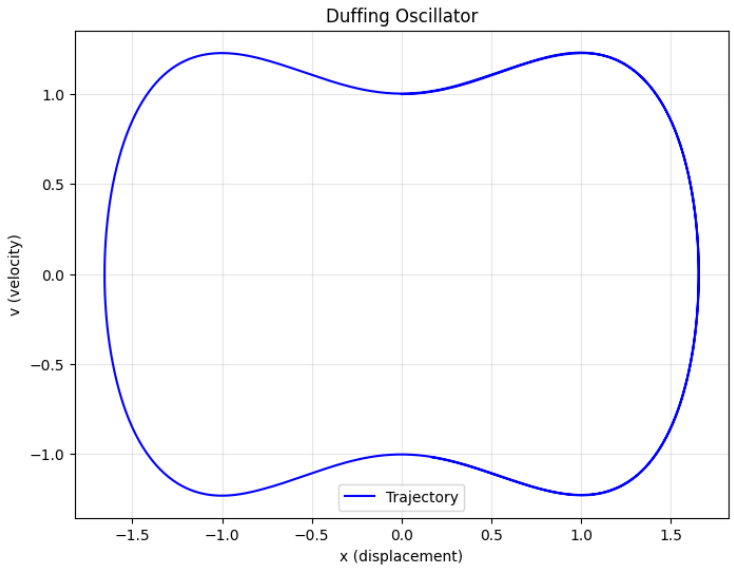

Figure 10: *Phase space of a periodic duffing oscillator.*

D.2 Physics-Informed Loss Function

The training of the model relies on a composite loss function that combines both data-driven and physics-informed components. The total loss, L_{total}, is defined as the sum of the Mean Squared Error (MSE) of the data predictions (L_{data}) and the MSE of the residual of the governing differential equation ($L_{physics}$):

$$L_{total} = L_{data} + L_{physics}$$

The data loss measures the discrepancy between the network's predicted next state $\hat{x}(t + \Delta t)$ and the true next state $x(t + \Delta t)$ from the generated dataset:

$$L_{data} = \frac{1}{N} \sum_{i=1}^{N} (\hat{x}_i(t + \Delta t) - x_i(t + \Delta t))^2$$

where N is the number of data points. The physics loss enforces that the network's predictions follow the physical law of the oscillator's differential equation. This is achieved by computing the residual of the ODE using the network's outputs and their automatically differentiated derivatives. Let $\hat{x}_{pred}$ denote the network's prediction of $x(t + \Delta t)$. We obtain the approximate first derivative, $\hat{v}(t)$, using a finite difference:

$$\hat{v}(t) \approx \frac{\hat{x}_{pred} - x(t)}{\Delta t}$$

where Δt is the time step between consecutive data points. The derivative of $\hat{x}_{pred}$ with respect to the input time t, denoted as $\frac{d\hat{x}_{pred}}{dt}$, is computed via automatic differentiation. The residual R is then

formulated by substituting these terms into the original Duffing equation. For our simplified Duffing equation, the residual is:

$$R = \frac{d\hat{x}_{pred}}{dt} - \left(-\delta\left(\frac{\hat{x}_{pred} - x(t)}{\Delta t}\right) - \alpha\hat{x}_{pred} - \beta(\hat{x}_{pred})^3\right)$$

The physics loss is the mean squared error of this residual:

$$L_{physics} = \frac{1}{N}\sum_{i=1}^{N}(R_i)^2$$

This combined loss function enables the PINN to learn both from the provided data and from the fundamental physical laws governing the Duffing oscillator.

E Scalability and Approximation Frameworks

As discussed in Section 4, the exact computation of the polyhedral decomposition for deep ReLU networks faces combinatorial explosion. The number of linear regions scales exponentially with the depth of the network and polynomially with the width [27]. Crucially, the complexity also scales exponentially with the dimension of the input space m [27], rendering exact enumeration intractable for high-dimensional data. To extend the topological analysis presented in this work to larger architectures, we propose shifting from exact computation to statistical approximation. Here we detail three specific methodologies to address the computational bottleneck.

E.1 Sampling-Based Approximation

The primary bottleneck in our current pipeline is the exhaustive traversal of the input space to identify every distinct activation region. We propose a *Monte Carlo approach* where we sample N points from the input domain and compute their binary codes to identify a subset of the active polytopes. We can then construct an approximate dual graph by connecting observed codes with a Hamming distance of 1. This reduction in graph size, scaling with the number of samples rather than the network's theoretical capacity, allows us to move beyond simple cell counting. By performing topological data analysis on this sampled subgraph, we can explore richer topological features and alternative filtrations that would be computationally prohibitive on the full exact complex.

E.2 Dimensionality Reduction

Given that the number of regions grows exponentially with the input dimension m [27], reducing the ambient space is a necessary step for scalability. A way to do this could be via *PCA Projection*. Since real-world data often resides on lower-dimensional manifolds, we can project the input space onto the top k principal components of the data (where $k \ll m$). The polyhedral decomposition is then computed on this k-dimensional affine subspace, avoiding the combinatorial explosion associated with the full ambient dimension. More generally, we could resort to *random slicing*. One can analyze the restriction of the neural network function to random 2D planes or 1D lines. Theoretical results in [10] demonstrate that the number of regions along a 1D line provides a strictly bounded proxy for the global complexity. Extending this to 2D slices allows for the computation of Betti curves that reflect local topological complexity.

E.3 Algebraic and Combinatorial Invariants

Finally, future work may bypass geometric construction entirely by leveraging tools from Algebraic Geometry and Combinatorics. Rather than attacking the full composition, we could do a *Layer-wise Analysis*. The global exponential complexity arises from the composition of layers. A single layer with width w acts as a hyperplane arrangement in $\mathbb{R}^d$, creating $O(w^d)$ regions [27]. While still computationally demanding for large d, analyzing the topology layer-by-layer decomposes the intractable global problem into a sequence of sub-problems that are more manageable.

We anticipate that future developments in computational algebraic geometry will provide more powerful tools to estimate the f-vector and other topological descriptors of these structures without requiring the explicit construction of the full polyhedral complex.