LAYERS AS LENSES: A FEATURE LEARNING NARRATIVE FOR GATED DEEP NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

End-to-end gradient-based training of multi-layer architectures has been immensely successful across machine learning tasks, yet principled explanations for why, when, and how it discovers task-relevant structure in data remain incomplete. Existing viewpoints based on representational power, input-space folding, optimization conditioning, the neural tangent kernel, and hierarchical feature learning offer useful insights, but leave important aspects of end-to-end feature learning unresolved. We propose a new narrative for gated deep networks trained by gradient descent. From the viewpoint of any given layer, the other layers act as a lens that assigns importance weights to the input, facilitating the emergence of features without global scope that might otherwise be overlooked. Gradient methods then update each layer to fit data that have been importance-weighted by a lens formed by the remaining layers. In this view, individual neuron parameters serve a dual role: separator for data lensed by other layers, and component of the lens for other layers. We hypothesize that the resulting positive feedback is a key mechanism underlying feature learning in gated deep networks. We support this narrative with theoretical analysis and empirical results on the recently proposed Deep Linearly Gated Network (DLGN), an architecture that combines elements of deep linear and ReLU networks. In particular, our analysis provides predictive control over gradient-descent trajectories from initialization in the DLGN setting, and naturally leads to a notion of hierarchical feature discovery in the feature-learning regime. We also present evidence suggesting that aspects of this narrative may extend to ReLU networks.

1 INTRODUCTION

End-to-end gradient-based training of multi-layer architectures often finds parameters that fit the training data, discover task-relevant structure in the input, and generalize well. A non-exhaustive list of explanations or narratives for this is given below.

1. *Deeper networks represent more complex functions.* (Telgarsky, 2016; Montufar et al., 2014). This is a valuable insight, but it does not explain how gradient descent finds such parameters from random initialization. In fact, many deep architectures that can represent complex functions still fail to discover parameters corresponding to the desired solution from random initialization.
2. *Deep networks implement kernel methods.* (Jacot et al., 2018; Arora et al., 2019). The neural tangent kernel is an extremely strong theoretical baseline, and is mathematically exact for infinite-width networks. However, it does not explain the empirical advantage of finite-width architectures over their infinite-width counterparts, and by itself does not explain finite-width feature learning.
3. *Feature kernels improve during training.* (Frei et al., 2023; Nichani et al., 2023; Damian et al., 2022). This is a highly promising direction and likely explains several phenomena in deep networks. However, it often leaves open the mechanistic question of how end-to-end gradient descent in a multi-layer architecture produces this improvement.

Our work fits most naturally within the third narrative family above. A major challenge in this line of work is that realistic end-to-end analyses of feature improvement in deep networks are difficult.

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

Existing theoretical results therefore often rely on restrictive assumptions, such as layerwise training (Nichani et al., 2023; Dandi et al., 2024), one- or two-hidden-layer settings (Nichani et al., 2023; Dandi et al., 2024; Abbe et al., 2021; Damian et al., 2022), or strong assumptions on the data distribution (Abbe et al., 2021; Dandi et al., 2024). We make a different tradeoff: we study a gated deep architecture and a structured data model chosen to make inter-layer feature-learning dynamics analytically visible.

We study this question through the Deep Linearly Gated Network (DLGN) (Banerjee et al., 2025), a transparent gated architecture that preserves the multiplicative interactions central to our analysis while allowing neuron-level dissection of training dynamics. At a high level, our analysis identifies two competing second-order effects in single-path DLGNs, shows how they help organize training in wider DLGNs, and finds related hierarchical feature-emergence patterns in ReLU networks.

We restrict our attention to binary classification with feature-vector inputs. Our formal theorems are stated for DLGNs in a structured synthetic data setting and in the population-gradient-flow regime. While we believe the broader narrative may extend to more realistic architectures and datasets, such extensions are currently supported only partially and remain directions for future work.

1.1 Contributions

We study a lens/separator viewpoint for feature learning in gated deep networks, in which each layer reweights the data seen by other layers. In the DLGN setting, we provide second-order theoretical results for single-path models, together with empirical analysis of wider models including trajectory prediction, regularization, and intervention experiments. We also present empirical results in ReLU networks that exhibit related hierarchical feature-emergence patterns, and use the ODT setting to motivate a notion of hierarchical feature discovery based on the scope of discontinuities in the target function.

2 Decision Tree Data and Deep Linearly Gated Network Architecture

2.1 Motivation for the Data Setting

A useful theoretical data setting should be non-trivial enough that feature learning by deep architectures is genuinely needed, while still permitting precise analysis. Real-world datasets often satisfy the first requirement, but their complexity makes the dynamics of gradient descent difficult to analyze. At the other extreme, canonical synthetic benchmarks such as parity functions, linearly separable data, or low-degree polynomials (Daniely & Malach, 2020; Frei et al., 2023; Soudry et al., 2018; Wu et al., 2023; Abbe et al., 2021) are analytically convenient but do not capture the kind of hierarchical structure that motivates deep learning theory.

Data labeled by an oblique decision tree (Banerjee et al., 2025) offers a useful middle ground. Although realistic datasets are rarely generated by literal tree processes, many exhibit multi-scale hierarchical structure that trees can capture or approximate. In particular, the internal node hyperplanes of an oblique tree can be interpreted as localized discontinuities in the label function. This makes oblique trees expressive enough to model non-trivial structure, while remaining transparent enough for mechanistic analysis.

2.2 Motivation for the Architecture Setting

To study deep feature learning, we also need an architecture that retains the essential multiplicative interactions of modern deep networks while allowing meaningful parameter-level analysis. Classical models commonly used in theory, such as deep linear networks or shallow neural networks, are often too limited for this purpose. Standard multilayer ReLU networks are more realistic, but their internal parameter interactions make isolated analysis of individual neurons difficult.

We therefore use the Deep Linearly Gated Network (DLGN) (Banerjee et al., 2025), a transparent gated architecture that remains competitive on structured feature-vector tasks while permitting neuron-level dissection of training dynamics. The DLGN preserves the multiplicative gate interactions central to our narrative, while avoiding some of the opacity of standard ReLU models.

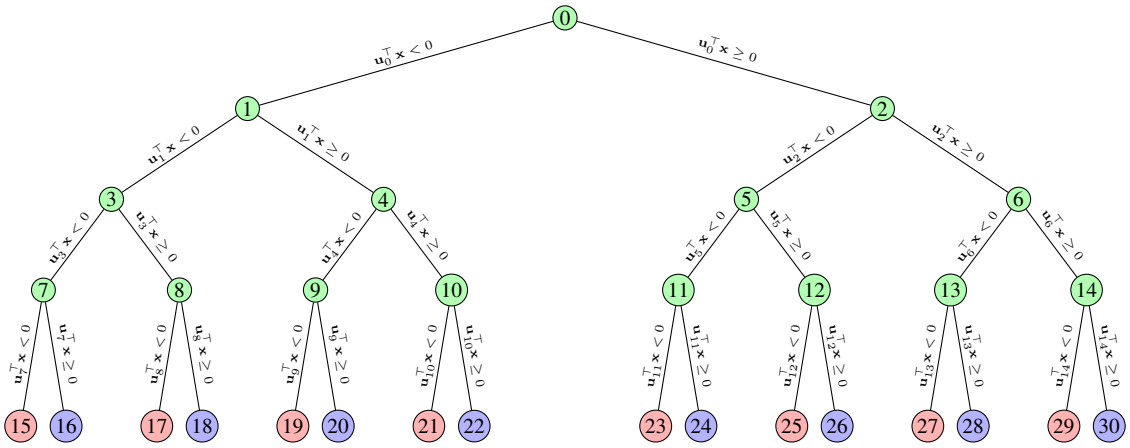

Figure 1: The decision tree with depth $D = 4$ has its decision nodes shaded green. Label nodes colored red have $y^* = +1$ and label nodes colored blue have $y^* = -1$.

2.3 The Oblique Decision Tree Data Setting

The instance $\mathbf{x}$ is drawn uniformly from the unit sphere in $\mathbb{R}^d$. The label $y \in \{+1, -1\}$ is given by a complete, orthogonal, balanced oblique decision tree (shortly, COB-ODT (Banerjee et al., 2025)) of depth D.

Concretely, consider a complete binary tree of depth $D-1$ with nodes indexed 0 to $N = 2^D - 2$. Each node $i \in \{0, 1, \ldots, N\}$ has left child $2i+1$, right child $2i+2$, and an associated decision vector $\mathbf{u}_i \in \mathbb{R}^d$. The vectors $\mathbf{u}_0, \ldots, \mathbf{u}_N$ are unit norm and mutually orthogonal. Traversal starts at the root; at node i, the point $\mathbf{x}$ moves left if $\mathbf{u}_i^\top \mathbf{x} < 0$ and right otherwise. After $D-1$ decisions, the label is given by the sign of $-\mathbf{u}_i^\top \mathbf{x}$ at the reached node i. For convenience, we number the terminal label nodes from $N+1$ to $2N+2$, even though they have no associated decision vector. We refer to nodes at level $D-1$ as *leaf nodes*, and nodes at level D as *label nodes*. Thus, for a depth-4 tree, nodes 7 to 14 are leaf nodes and nodes 15 to 30 are label nodes.

2.4 The Deep Linearly Gated Network Architecture Setting

The Deep Linearly Gated Network (DLGN) is defined as a sum of products of sigmoidal gates applied to linear functions:

$$\hat{y}(\mathbf{x}) = \sum_{\pi \in \Pi} a_\pi \prod_{\ell=1}^{L} \phi(\mathbf{w}_{\pi,\ell}^\top \mathbf{x}), \tag{1}$$

where Π is a discrete index set, L is the number of layers, and $\phi : \mathbb{R} \to [0, 1]$ is a scaled sigmoid, $\phi(u) = 1/(1 + \exp(-\beta u))$. The vectors $\mathbf{w}_{\pi,\ell}$ and coefficients a_π are the trainable parameters.

To connect this model to standard deep architectures, Banerjee et al. (2025) take $\Pi = [M]^L$, corresponding to all paths through a network with L hidden layers and M neurons per layer. The parameterization is then shared across paths: for $\pi = (m_1, \ldots, m_L) \in [M]^L$, the gate vector at layer ℓ depends only on m_ℓ, i.e. $\mathbf{w}_{\pi,\ell} = \mathbf{w}_{m_\ell,\ell}$. The path coefficient a_π is taken to be the product of edge weights along π in an associated deep linear network, $a_\pi = \prod_{\ell=2}^{L} a_{\ell,m_{\ell-1},m_\ell}$.

This shared parameterization allows the model output to be evaluated in $O(M^2 L)$ time (comparable to that of a ReLU net with same size) without explicitly summing over all M^L paths.

3 LEARNING DYNAMICS OF SINGLE-PATH DLGN

This model cannot fit the full ODT-labeled task, but it can capture a single ODT path and is therefore a useful setting for isolating the mechanism of interest. We now analyze how gradient descent on this model interacts with the ODT structure.

$$\hat{y}(\mathbf{x}; a, W) = a \prod_{\ell=1}^{L} \phi(\mathbf{w}_\ell^\top \mathbf{x}),$$

with parameters $a \in \mathbb{R}$ and $\mathbf{w}_1, \ldots, \mathbf{w}_L \in \mathbb{R}^d$. For convenience, we concatenate the L gating vectors into $W \in \mathbb{R}^{Ld}$, and let $P_\ell : \mathbb{R}^d \to \mathbb{R}^{Ld}$ denote the operator that writes a vector into the ℓ-th d-dimensional block and zeros elsewhere.

This model cannot in general fit the ODT-labeled data exactly, but its form suggests that it can effectively capture one path of the ODT labeling function. We now analyze how gradient descent on this model interacts with the ODT structure.

3.1 LENS AND BLINDERS

Our analysis starts from the loss gradient with respect to a single gating vector $\mathbf{w}_k$. For a data point $(\mathbf{x}, y^*)$ with y^* given by equation 10, the logistic loss and its gradient are

$$\begin{aligned} &\mathcal{L}(\hat{y}(\mathbf{x})) = \log\big(1 + \exp(-y^* \hat{y}(\mathbf{x}))\big), \\ &-\nabla_{\mathbf{w}_k} \mathcal{L} = \mathbb{E}\left[\underbrace{\sigma(-y^* \hat{y}(\mathbf{x}))}_{\text{Residue}} \underbrace{\Lambda_k(\mathbf{x})}_{\text{Lens}} \underbrace{\phi'(\mathbf{w}_k^\top \mathbf{x})}_{\text{Blinder}} \, y^* a \mathbf{x} \right], \end{aligned} \tag{2}$$

where $\Lambda_k(\mathbf{x}) = \prod_{\kappa \neq k} \phi(\mathbf{w}_\kappa^\top \mathbf{x})$.

Thus the update to $\mathbf{w}_k$ is a Perceptron-type update on the whole training set, with each example $(\mathbf{x}, y^*)$ weighted by the product of residue, lens, and blinder terms. From this viewpoint, $\mathbf{w}_k$ is trained as a separator on data reweighted by a lens formed by the other layers, while simultaneously serving as a component of the lens that shapes the gradients of those layers.

When the lens emphasizes certain regions of input space, the resulting gradient can have a strong component along feature directions that do not dominate globally, such as the ODT root direction $\mathbf{u}_0$. This creates a positive-feedback mechanism in which movement of one layer can encourage other layers to move toward complementary task-relevant features.

3.2 SECOND-ORDER ANALYSIS

To analyze feature learning, we study how the NTK features change during training, which corresponds to a second-order analysis of the loss. Specializing this analysis to ODT-labeled data, we show how the interaction between the "lens" and "separator" roles of a neuron $\mathbf{w}_k$ can encourage the model to acquire ODT decision vectors that are initially absent from the gradient.

Theorem 3.1 (Hierarchy-seeking lens)**.** *Let $\mathbf{h}, \mathbf{z} \in \mathbb{R}^d$. Let $0 < a < 1$ and $k \neq \ell$. Assume $\mathbf{w}_\lambda^\top \mathbf{h} = \mathbf{w}_\lambda^\top \mathbf{z} = 0$ for all $\lambda \in [L]$. Then the change in the gradient of $\mathcal{L}$ with respect to $\mathbf{w}_k$ in direction $\mathbf{z}$ as $\mathbf{w}_\ell$ moves in direction $\mathbf{h}$ satisfies*

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[-\nabla_{\mathbf{w}_k} \mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) + \nabla_{\mathbf{w}_k} \mathcal{L}(W) \Big] \cdot \mathbf{z} > 0,$$

for the choice $\mathbf{z} = s\mathbf{u}_i$ and $\mathbf{h} = \mathbf{u}_j$, where $j \in \{N/2, \ldots, N\}$ is a leaf node of the ODT and i is any ancestor of j. The scalar $s = +1$ if j lies in the left subtree of i and $s = -1$ otherwise. The same inequality holds when $\mathbf{h}$ and $\mathbf{z}$ are swapped.

As one concrete consequence, if $\mathbf{w}_\ell$ moves along $-\mathbf{u}_j$ for some leaf node j on the left of the root node 0, then $\mathbf{w}_k$ is pushed along $-\mathbf{u}_0$. This formalizes a tendency of gradient descent in this model: movement of one neuron along a leaf direction can increase the incentive for another neuron to move along an ancestor direction. In high-dimensional settings where the initial $\mathbf{w}_k$ are nearly orthogonal

to all ODT directions, the raw gradient has negligible component along internal-node directions such as $\mathbf{u}_0$, so this second-order effect provides a concrete route by which such features can enter the dynamics.

Under a symmetric initialization such as $W = 0$ with $a > 0$, gradient descent reaches a saddle point W^* where each of the L neurons takes the same vector $\mathbf{w}^* = c\sum_j -\mathbf{u}_j$, with j ranging over leaf indices $N/2$ to N. The second-order dynamics captured in Theorem 3.1 imply that small asymmetric perturbations away from this saddle can trigger a positive feedback loop in which different layers specialize toward different ODT directions. We illustrate this effect with an explicit initialization in Section 3.3.

Second-order effects can also produce undesirable couplings.

Theorem 3.2 (Self-opposing lens)**.** *Let $\mathbf{h} \in \mathbb{R}^d$ be orthogonal to all ODT decision vectors $\mathbf{u}_i$ for $i \in \{0, 1, \ldots, N\}$. Assume $\mathbf{w}_\lambda \cdot \mathbf{h} = 0$ for all $\lambda \in \{1, \ldots, L\}$ and $\mathbf{w}_\lambda \cdot \mathbf{u}_j = 0$ for all λ and all $j \in \{N/2, \ldots, N\}$. Then the change in the gradient of $\mathcal{L}$ with respect to $\mathbf{w}_k$ along $\mathbf{h}$ as $\mathbf{w}_\ell$ (for some $\ell \neq k$) moves along $-\mathbf{h}$ satisfies*

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon}\Big[-\nabla_{\mathbf{w}_k}\mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) + \nabla_{\mathbf{w}_k}\mathcal{L}(W)\Big] \cdot (-\mathbf{h}) > 0.$$

Informally, if some neuron $\mathbf{w}_\ell$ is perturbed along a direction $\mathbf{h}$ that is orthogonal to all ODT decision vectors, then other neurons $\mathbf{w}_k$ tend to move along $-\mathbf{h}$, increasing the magnitude of both $\mathbf{h}$ and $-\mathbf{h}$ components across layers. In other words, neurons can end up trying to correct each other's spurious movements and, in doing so, reinforce structure in directions that are irrelevant for generalization. This phenomenon becomes more pronounced in the multi-path DLGN, where many such couplings can interact; we return to this point in Section 4.1.

Theorems 3.1 and 3.2 are derived under orthogonality assumptions (in particular, $d \gg D$ so that random initial weights are likely to be nearly orthogonal to all $\mathbf{u}_i$). The relevant second-order quantities are continuous in W, so the qualitative effects persist under small deviations from strict orthogonality. In our experiments, we use these theorems as a guide to interpret and predict training trajectories, rather than as a complete description of the dynamics at all times.

3.3 Trajectory analysis for a chosen initialization

We now illustrate the above mechanisms with an explicit trajectory. We consider a depth-$D = 4$ ODT label function and a single-path DLGN with $L = D = 4$, fix $a = 1$, and study the evolution of the gating vectors $\mathbf{w}_\ell$ under gradient flow. The same construction can be extended to deeper trees and DLGNs.

Because $a > 0$, the model can achieve non-trivial log loss (values below $\log 2$) only on positively labeled points, i.e. data reaching label nodes $15, 17, \ldots, 29$, which all lie on the left side of their parents. With initialization $W = 0$ and $a > 0$, symmetry implies that gradient descent pushes each $\mathbf{w}_k$ along $\sum_j -\mathbf{u}_j$, where j ranges over leaf indices $N/2$ to N, until it reaches the saddle point W^*; the gradient has no component along other directions at this point.

We now perturb this initialization slightly in a direction suggested by Theorem 3.1. Let

$$\mathbf{w}_1 = 0, \quad \mathbf{w}_2 = -\epsilon\mathbf{v}, \quad \mathbf{w}_3 = -\epsilon\mathbf{v} - \epsilon\mathbf{u}_7, \quad \mathbf{w}_4 = -\epsilon\mathbf{v} - \epsilon\mathbf{u}_8,$$

for some small $\epsilon > 0$ and $\mathbf{v} = \mathbf{u}_7 + \mathbf{u}_8 + \mathbf{u}_9 + \mathbf{u}_{10}$. The resulting parameter trajectory is shown in Figure 2 and its gradient trajectory is shown in Appendix Figure 6. From the perspective of $\mathbf{w}_1$, the other layers have moved along $-\mathbf{u}_7, -\mathbf{u}_8, -\mathbf{u}_9, -\mathbf{u}_{10}$, so Theorem 3.1 predicts an increased push on $\mathbf{w}_1$ toward $-\mathbf{u}_0$. Swapping the roles of $\mathbf{h}$ and $\mathbf{z}$ in the theorem implies that movement of $\mathbf{w}_1$ toward $-\mathbf{u}_0$ increases the tendency of the other layers to move along leaf-node directions, closing a feedback loop. The small additional perturbations of $\mathbf{w}_3$ and $\mathbf{w}_4$ along $-\mathbf{u}_7$ and $-\mathbf{u}_8$ similarly induce a tendency for $\mathbf{w}_2$ to move along $-\mathbf{u}_1$, which becomes pronounced once $\mathbf{w}_1$ has moved sufficiently toward $-\mathbf{u}_0$.

Figure 2 shows that the first ≈ 10 epochs move the parameters from the perturbed initialization to the saddle W^*. The influence of the second-order dynamics described above becomes visible thereafter: from roughly epoch 10 to 50, the gradient component of $\mathbf{w}_1$ along $-\mathbf{u}_0$ strengthens;

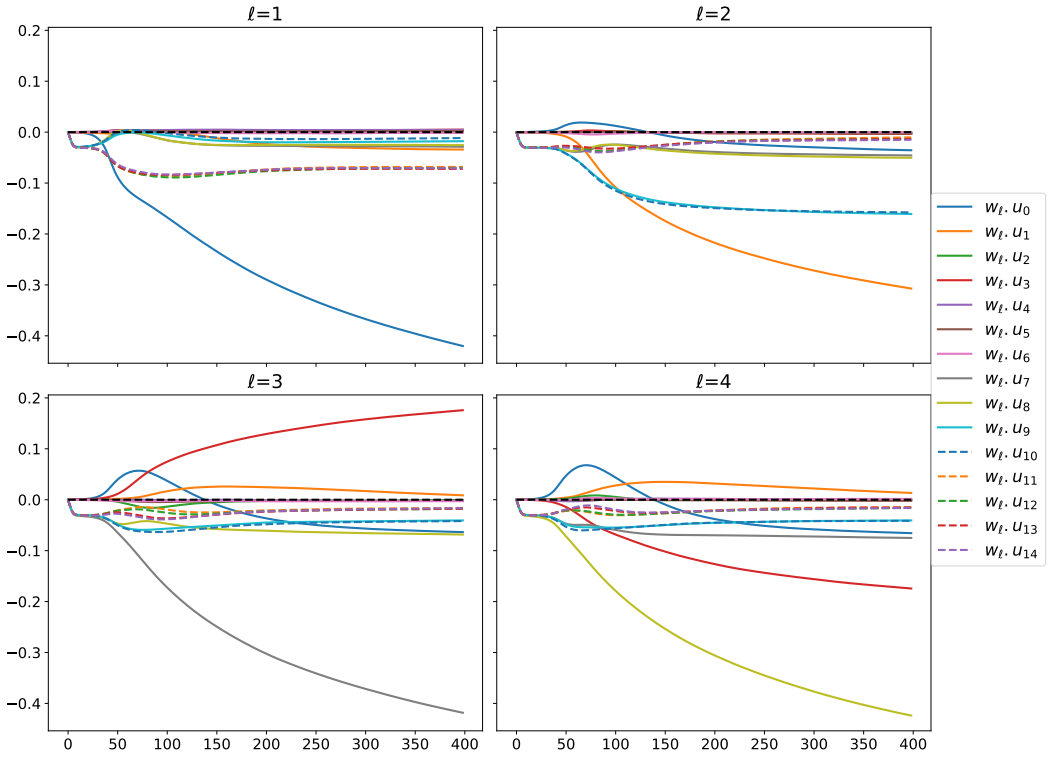

Figure 2: Parameter trajectory for a single-path DLGN under gradient descent.

from epoch 50 to 90, the gradient component of $\mathbf{w}_2$ along $-\mathbf{u}_1$ increases; and $\mathbf{w}_3, \mathbf{w}_4$, which were initially biased toward $-\mathbf{u}_7$ and $-\mathbf{u}_8$, continue to grow along those directions. The convergent solution (up to scaling) is well approximated by

$$\mathbf{w}_1 \approx -\mathbf{u}_0, \quad \mathbf{w}_2 \approx -\mathbf{u}_1, \quad \mathbf{w}_3 \approx -\mathbf{u}_7 + 0.3\,\mathbf{u}_3, \quad \mathbf{w}_4 \approx -\mathbf{u}_8 - 0.3\,\mathbf{u}_3,$$

as shown in Appendix Figure 5. This corresponds to a superposition of two adjacent positively labeled ODT paths, e.g. $(-\mathbf{u}_0, -\mathbf{u}_1, -\mathbf{u}_7, -\mathbf{u}_3)$ and $(-\mathbf{u}_0, -\mathbf{u}_1, +\mathbf{u}_3, -\mathbf{u}_8)$.

For $D = L = 4$, there are four such solutions, corresponding to the leaf pairs $(7, 8)$, $(9, 10)$, $(11, 12)$, and $(13, 14)$. Permutation invariance across the L layers yields $4! = 24$ configurations per pair, for a total of 96 solutions in parameter space (in general, $2^{D-2}D!$ when $D = L$). Small perturbations around $W = 0$ can be constructed to reach any of these solutions under gradient descent.

We represent these solutions compactly by a D-tuple recording the ODT direction of alignment for each neuron. For a solution $\tilde{W} \in \mathbb{R}^{D\times d}$, define $\upsilon(\tilde{W}) \in \{0, 1, \ldots, 2^D - 2\}^D$ by $\upsilon_\ell(\tilde{W}) = \arg\max_{i\in\{0,\ldots,2^D-2\}} |\mathbf{w}_\ell \cdot \mathbf{u}_i|$. e.g., the trajectory in Figure 2 has $\upsilon(\tilde{W}) = [0, 1, 7, 8]$.

3.4 Solution prediction from random initialization

The construction above shows that carefully designed perturbations can steer gradient descent toward any of the $2^{D-2}D!$ canonical solutions. For typical random initializations, however, the perturbations from 0 need not align perfectly with a single canonical pattern, and several solutions may be supported simultaneously. Assuming that gradient descent converges to one of the 96 solutions, we ask: which one does a given initialization select?

We do not have a closed-form answer, but we can build a heuristic score that uses Theorems 3.1 and 3.2 to estimate the support an initialization $W(0)$ gives to each solution. The construction, detailed in Appendix A.5, combines three terms: a proximity term measuring how close $W(0)$ is to the solution, a hierarchy-seeking term weighted by $c_{\text{HS}} \geq 0$, and a self-opposing term weighted by $c_{\text{SO}} \geq 0$.

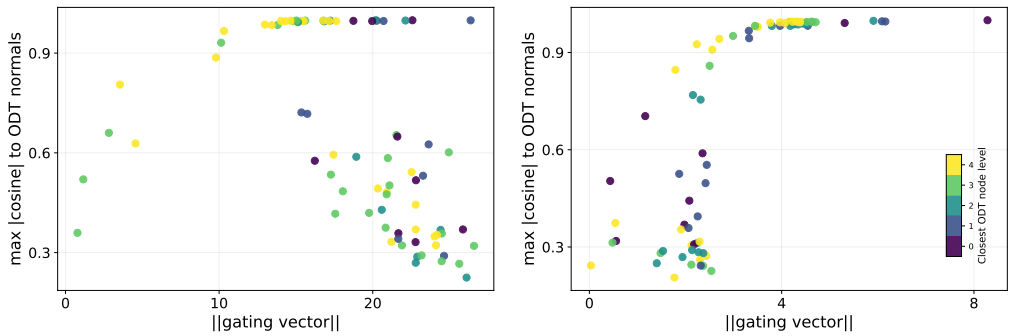

Figure 3: Gating norms vs. cosine similarity to ODT vectors for a 5-layer DLGN with $M = 16$: (a) standard log-loss training, (b) with an additional L_2 penalty on gating vectors.

We run gradient descent for the single-path DLGN with 2000 different initializations, each evaluated on 5 independently sampled datasets to separate initialization effects from finite-data noise. For 368 initializations, the convergent solution is stable across datasets, and we treat these as "deterministic" initializations. We search $c_{\text{HS}}, c_{\text{SO}} \in \{0.0, 0.1, \ldots, 0.9\}$, and find that the best-performing pair $(c_{\text{SO}}, c_{\text{HS}}) = (0.7, 0.4)$ correctly predicts the converged solution for 179 out of 368 deterministic initializations, corresponding to 49% accuracy in a 96-class prediction problem.

A simple baseline that predicts the solution closest (in parameter space) to the initialization corresponds to $(c_{\text{SO}}, c_{\text{HS}}) = (0, 0)$ and attains only 13% accuracy. Moreover, the best achievable accuracy when forcing $c_{\text{HS}} = 0$ is 28%, and when forcing $c_{\text{SO}} = 0$ is 36%. Both the hierarchy-seeking and self-opposing terms are therefore important for predictive power in this heuristic, providing quantitative support for our viewpoint in the single-path DLGN setting.

4 Learning Dynamics of Multi-Path DLGN

We now turn to the more practical multi-path DLGN with $M > 1$. With $M = 1$, the model can at best capture data entering one leaf node of the ODT, achieving low loss on only a 2^{-D} fraction of examples and close to random performance on the rest. Wider DLGNs do not suffer from this limitation.

Throughout this section we consider $d = 100$, depth-5 ODT-labeled data, and a 5-layer DLGN with $M = 16$ neurons per layer. In this setting, the DLGN achieves low training loss and low test error on the full dataset, whereas linear, kernel, and standard tree-based baselines remain above 30% test error (see Appendix A.6).

4.1 Effect of Gating Regularization

We first train the DLGN by minimizing the log loss over both path coefficients a and gating vectors $\mathbf{w}$ using a standard gradient-based optimizer. In this regime, the train loss decreases from $\log 2 \approx 0.69$ at initialization to about 0.05, and the test error drops from roughly 50% to about 17%. For a typical architecture/dataset combination this would often be considered satisfactory, with further improvement delegated to hyperparameter tuning.

In our ODT setting, however, we can inspect learning behavior more directly using the known ODT decision vectors $\mathbf{u}_j$. Ideally, as training progresses, gating vectors $\mathbf{w}_{m,\ell}$ that grow in norm should also align with some ODT direction $\mathbf{u}_j$, since any component orthogonal to all $\mathbf{u}_i$ does not help fit the target function. Figure 3(a) shows, at the end of standard training, the norm of each gating vector versus its maximum cosine similarity with any ODT direction. While the initialization cloud sits around $(0.6, 0.25)$ with no structure, after training we see many high-norm gating vectors that remain poorly aligned with all ODT directions. This suggests that a significant fraction of capacity is devoted to directions that are not directly related to the ODT structure.

Motivated by the self-opposing behavior in Theorem 3.2, where movement along spurious directions can propagate across layers, we explore a simple modification: adding an L_2 penalty on gating norms

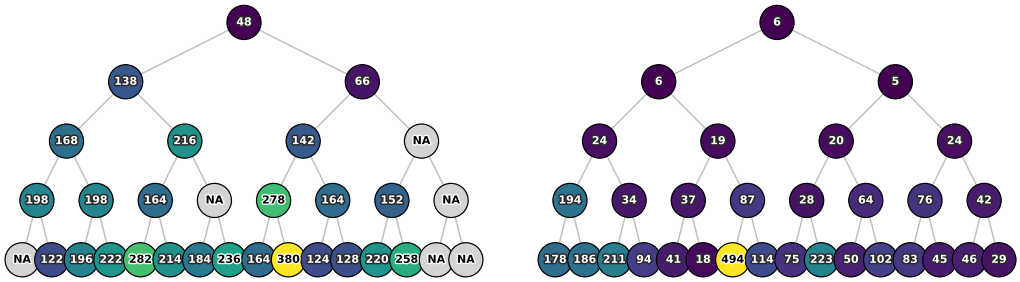

Figure 4: ODT node capture epochs for a depth-5 ODT: (a) across all gating vectors in a 5-layer DLGN, (b) across first-layer weight vectors in a 3-layer ReLU network.

to the loss. This penalty is not primarily intended as a conventional regularizer for test performance, but as a way to discourage large-norm components in directions that are not supported by the ODT features, thereby altering the optimization landscape. As shown in Figure 3(b), training with this penalty leads to a markedly different outcome: almost all large-norm gating vectors align closely with some ODT decision vector.

The L_2 penalty, however, also limits the norms that gating vectors can achieve along ODT directions and slightly reduces attainable performance: in this regime the final training loss bottoms out around 0.03, with test error near 13%. To retain the alignment benefits while allowing larger norms, we adopt a two-phase procedure. We first train with the L_2 penalty as above, then restart training from the convergent solution with the penalty removed. At the restart, we additionally prune gating vectors whose norms fall below a small threshold by setting them to zero and keeping them fixed in the second phase. This pruning step is motivated by Figure 3(b), which suggests that most useful gating vectors can be identified by their norms.

This two-phase strategy further reduces test error to about 6% in our example. Averaging over 10 random seeds, we observe the following pattern: standard log-loss training yields $17.8\% \pm 0.7\%$ test error; adding gating regularization alone improves this to $13\% \pm 0.7\%$; restarting from the regularized solution without further regularization yields $8.9\% \pm 0.7\%$; and adding pruning between the two phases attains $6\% \pm 0.7\%$ test error. These results are consistent with the view that second-order interactions along irrelevant directions can produce undesirable behavior, and that shaping the gating norms can help mitigate this in the multi-path setting.

4.2 Hierarchical feature emergence

Theorem 3.1 suggests a hierarchical preference: random motion of a gating vector along any leaf-node direction tends to enhance movement along the root direction $\mathbf{u}_0$, while similar second-order effects for deeper internal nodes require more specific conditions. The trajectory in Figure 2 reflects this, with gating vectors first moving along the ODT root and then progressively along its children.

A similar pattern appears in the multi-path DLGN. We define the *capture epoch* of an ODT direction $\mathbf{u}_j$ as the first epoch at which there exists a gating vector with norm exceeding a fixed threshold (e.g. > 2.5) and cosine similarity above a fixed threshold (e.g. > 0.8) to $\mathbf{u}_j$. Figure 4(a) shows these capture epochs for all ODT nodes when training a 5-layer DLGN with an L_2 penalty on the gating vectors from a random initialization. We observe a clear trend in which parent nodes tend to be captured earlier than their children, providing an empirical notion of hierarchical feature emergence consistent with the single-path analysis.

There have been multiple notions of hierarchy in the data, and claims on how they reflect in the learned neural network – compositional hierarchy Dandi et al. (2025); Tomasini & Wyart (2024), image coarse-to-fine hierarchy Zeiler & Fergus (2014). In our work, we provide a mechanistic narrative for when the hierarchy is present in the data via a tree labelling function and manifests itself in the learned network via order of emergence during training.

5 ReLU Extensions and broader interpretation

The ReLU network output can also be written as a sum over paths $\pi = (m_1, \ldots, m_L) \in [M]^L$ similar to Equation 1 with each term applicable for points that activate the entire path.

$$\hat{y}(\mathbf{x}) = \sum_{\pi \in [M]^L} \prod_{\ell=1}^{L} \left(w^{\ell+1}_{m_{\ell+1}, m_\ell} \mathbf{1}(g^\ell_{m_\ell}(\mathbf{x}) > 0) \right) \left(\mathbf{w}^1_{m_1} \cdot \mathbf{x} \right),$$

where the neuron pre-activations are given by $g^\ell(\mathbf{x})$ and parameters are given by $W^1, \ldots, W^{L+1}$. The DLGN makes this multiplicative structure explicit and central to its identity. But the underlying multiplicative dependence between layers is shared by many gated architectures, including ReLU networks. The viewpoint of the gradient descent update of a given layer being a perceptron update with a dynamically changing importance weight function (albeit much more complicated than a product of sigmoids of linear functions) is also generally valid.

We thus view the feedback between the lens and separator roles of a neuron as a plausible narrative for feature learning in more practical architectures like the ReLU network. The predominant theory of layer-wise compositional learning is powerful in explaining representation, but is insufficient for understanding parameter learning from a random initialization.

Experiments with ReLU networks suggest that phenomena associated with feature learning in DLGNs also appear in ReLU networks. On the same ODT dataset, ReLU networks with two or more hidden layers (three or more weight matrices) achieve performance comparable to DLGNs and exhibit a similar hierarchical feature-emergence pattern. Figure 4(b) shows the capture epochs of ODT node vectors by the first layer of a ReLU network; the order of emergence closely resembles the order observed in DLGNs – parents are captured first.

We also perform intervention experiments in which gradient updates for the first layer weights of the ReLU network are projected onto the orthogonal complement of a chosen ODT direction $\mathbf{u}_j$ during training. In these ReLU settings, such projections typically degrade final performance and substantially reduce alignment with ODT directions under the subtree of j, indicating that suppressing an internal-node direction can hinder the learning of its descendants. These observations are consistent with a hierarchical feature-discovery process in which internal features facilitate the acquisition of more local descendant features.

Viewed more broadly, the ODT setting can be interpreted through the concept of a *scope of discontinuity*: the root node 0 is more prominent than its child node 1 because flipping the sign of $\mathbf{x} \cdot \mathbf{u}_0$ would change labels for half the data, whereas flipping the sign of $\mathbf{x} \cdot \mathbf{u}_1$ affects only a quarter. This notion of scope is tied to the labeling function and is valid in more general settings where label changes concentrate along a hierarchy of increasingly localized discontinuities.

6 Conclusion

This paper studied a mechanistic lens/separator view of feature learning in gated deep networks through the analytically transparent DLGN–ODT setting. In this view, individual neurons both fit the currently weighted data and help determine the effective weighting elsewhere.

Our single-path analysis identified two competing second-order effects in gradient descent: a hierarchy-seeking effect that can promote movement toward task-relevant internal features, and a self-opposing effect that can amplify movement along spurious directions. We demonstrated the applicability of these insights from population gradient flow to build a robust predictor for gradient descent trajectories in a complex objective with multiple global minima. In wider DLGNs, these ideas help interpret both the emergence of ODT-aligned features and the optimization behavior that motivates regularization and two-phase training. We also observed hierarchical feature-emergence patterns in both DLGN and ReLU networks, together with intervention results consistent with the proposed mechanism.

While our formal guarantees are restricted to the DLGN–ODT setting, the results suggest a broader connection between multiplicative cross-layer structure, gradient-based optimization, and hierarchical feature discovery. We hope this viewpoint provides a useful starting point for a more general theory of feature learning in deep networks.

AI USE STATEMENT

We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, and interpret results. We used generative AI tools to create and edit code, identify relevant literature, and edit a few parts of the paper to improve readability. We have reviewed all AI-assisted work. Sections of the code generated by the LLM were verified by the authors, and relevant literature identified by the LLM was checked for correctness. We take responsibility for the final content of this work, including text, claims and artifacts submitted as part of this work.

ETHICS STATEMENT

This work is theoretical and uses only synthetically generated data, with no human subjects, personal data, or sensitive information. We do not foresee any ethical concerns arising from this research.

REPRODUCIBILITY STATEMENT

The data setting and the DLGN architecture are defined in Sections 2.3 and 2.4. For the theoretical results, the assumptions of Theorems 3.1 and 3.2 are stated with the theorems in Section 3.2, and complete proofs are given in Appendices A.2 and A.3; the notation is summarised in Table 1. For the empirical results, the source code is provided as supplementary material. It contains a separate Jupyter notebook for each experimental result, together with a README that maps the figures, tables and reported numbers to the notebook that produces them. All experiments use synthetic data labelled by oblique decision trees (Section 2.3); the code that generates this data is included, so no external dataset is needed, and the random seeds used for the paper are fixed in the notebooks. The heuristic solution predictor of Section 3.4 is fully specified in Appendix A.5. Since that experiment requires 2000*5 training runs, the supplementary material also includes the saved initial and trained weights of the 368 out of 2000 initializations used there, so that the reported prediction accuracies can be checked without retraining. The baseline comparison, the hyperparameter search spaces and the compute resources are described in Appendix A.6 and the subsections that follow it.

REFERENCES

Emmanuel Abbe, Enric Boix-Adsera, Matthew S Brennan, Guy Bresler, and Dheeraj Nagaraj. The staircase property: How hierarchical structure can guide deep learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26989–27002. Curran Associates, Inc., 2021.

Sanjeev Arora, Simon Shaolei Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Prithaj Banerjee, Harish Guruprasad Ramaswamy, Mahesh Lorik Yadav, and Chandra Shekar Lakshminarayanan. Deep networks learn features from local discontinuities in the label function. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alexandru Damian, Jason Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In Po-Ling Loh and Maxim Raginsky (eds.), *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pp. 5413–5452. PMLR, 02–05 Jul 2022.

Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. How two-layer neural networks learn, one (giant) step at a time. *Journal of Machine Learning Research*, 25(349): 1–65, 2024.

Yatin Dandi, Luca Pesce, Lenka Zdeborová, and Florent Krzakala. The computational advantage of depth in learning high-dimensional hierarchical targets. In *Advances in Neural Information Processing Systems*, volume 38, pp. 160705–160764. Curran Associates, Inc., 2025.

Amit Daniely and Eran Malach. Learning parities with neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20356–20365. Curran Associates, Inc., 2020.

Spencer Frei, Niladri S. Chatterji, and Peter L. Bartlett. Random feature amplification: Feature learning and generalization in neural networks. *Journal of Machine Learning Research*, 24(303): 1–49, 2023.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Eshaan Nichani, Alex Damian, and Jason D Lee. Provable guarantees for nonlinear feature learning in three-layer neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10828–10875. Curran Associates, Inc., 2023.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70): 1–57, 2018.

Matus Telgarsky. Benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir (eds.), *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

Umberto Tomasini and Matthieu Wyart. How deep networks learn sparse and hierarchical data: the sparse random hierarchy model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48301–48325. PMLR, 2024. URL `https://proceedings.mlr.press/v235/tomasini24a.html`.

Jingfeng Wu, Vladimir Braverman, and Jason D. Lee. Implicit bias of gradient descent for logistic regression at the edge of stability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pp. 818–833. Springer, 2014.

A Appendix

A.1 Notations

For any vectors $\mathbf{h}, \mathbf{z}$, We use both $\mathbf{h} \cdot \mathbf{z}$ and $\mathbf{h}^\top \mathbf{z}$ to both mean the dot product, and will use $\mathbf{h} \cdot \mathbf{z}$, whenever it is inconvenient to use $\mathbf{h}^\top \mathbf{z}$. Table 1 lists all the notations used in the paper.

A.2 Proof for Theorem 3.1

Theorem. *Let $\mathbf{h}, \mathbf{z}$ be vectors in $\mathbb{R}^d$. Let $0 < a < 1, k \neq \ell$. Let $\mathbf{w}_\lambda^\top \mathbf{h} = \mathbf{w}_\lambda^\top \mathbf{z} = 0$ for all λ. The change in gradient of $\mathcal{L}$ w.r.t. $\mathbf{w}_k$ along $\mathbf{z}$, as $\mathbf{w}_\ell$, moves along $\mathbf{h}$ satisfies*

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[-\nabla_{\mathbf{w}_k} \mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) + \nabla_{\mathbf{w}_k} \mathcal{L}(W) \Big] \cdot \mathbf{z} > 0$$

Table 1: Table of notations

Variable and domain	Meaning
d	Data dimension
D	Depth of ODT tree
L	# of hidden layers in model
$\mathcal{L}$	Log loss for binary classification
M	# of neurons per hidden layer
$i, j, \iota, \zeta \in \{0, 1, \ldots, N\}$	ODT decision node indices, $N = 2^D - 2$
$\mu, \nu \in \{N+1, \ldots, 2N+2\}$	ODT label node indices
$p, q \in \{0, 1, 2, \ldots, D\}$	ODT depth indices
$k, l, \kappa, \lambda \in \{1, 2, \ldots, L\}$	DLGN Model layer indices
$m \in \{1, 2, \ldots, M\}$	Neuron index
$\pi \in \Pi = [M]^L$	DLGN path index
$\gamma \in \Gamma \subseteq \{0, \ldots, 2N+2\}^{D+1}$	ODT Path
$\gamma(\mathbf{x})$	ODT path traversed by point $\mathbf{x}$
$\sigma(a)$	$1/(1+\exp(-a))$
$\phi : \mathbb{R} \to [0, 1]$	Sigmoidal function.
$\tau \in \{+1, -1\}^D$	Sequence of left/right decisions
lev(i) $\in \{0, \ldots, D\}$	Distance of node i from 0 in the ODT
par$(i) \in \{0, 1, \ldots, N\}$	Parent of node i
anc$(i) \subseteq \{0, 1, \ldots, N\}$	Ancestors of node i
$\mathbf{v}, \mathbf{h}, \mathbf{z}$	Some vectors in $\mathbb{R}^d$
$\mathbf{u}_j \in \mathbb{R}^d$	ODT decision vector for node j
$\mathbf{w}_\ell \in \mathbb{R}^d$	DLGN parameter in layer ℓ

where $\mathbf{z} = s\mathbf{u}_i, \mathbf{h} = \mathbf{u}_j$, $j \in \{N/2, \ldots, N\}$ is a leaf node of the ODT, and i is any ancestor of j. The scalar $s = +1$ if j is in the left sub-tree of i and -1 otherwise. The above inequality also holds when $\mathbf{h}$ and $\mathbf{z}$ are swapped.

Proof. Without loss of generality, let $j = N/2$. Let $\gamma(\mathbf{x}) \in \{0, 1, \ldots, 2N+2\}^{D+1}$ be the path traversed in the ODT by data point $\mathbf{x}$. Its D-th element $\gamma_D(\mathbf{x})$ gives the leaf decision node, and its $D+1$-th element $\gamma_{D+1}(\mathbf{x})$ gives the label node. The label $y(\mathbf{x}) = +1$ if $\gamma_{D+1}(\mathbf{x})$ is odd, and -1 otherwise. Let $\epsilon > 0$ be an arbitrarily small number. All expressions involving ϵ below ignore quadratic and higher terms.

Let $\alpha(\mathbf{x}) = \frac{\phi(\mathbf{w}_\ell^\top \mathbf{x} + \epsilon \mathbf{h}^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} = 1 + \frac{\epsilon \mathbf{h}^\top \mathbf{x} \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})}$.

$$
\begin{aligned}
&-\nabla_{\mathbf{w}_k} \mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) \cdot \mathbf{z} \\
&= \mathbb{E}_{\mathbf{x}} \left[\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \alpha(\mathbf{x})) \Lambda_k(\mathbf{x}) \alpha(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot \mathbf{z} \right] \\
&= \frac{2}{2^D} \sum_{\iota = N/2}^{N} \mathbb{E}_{\mathbf{x}} \left[\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \alpha(\mathbf{x})) \Lambda_k(\mathbf{x}) \alpha(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot \mathbf{z} \middle| \gamma_D(\mathbf{x}) = \iota \right] \qquad (3)
\end{aligned}
$$

We will analyze the sum of these conditional expectation terms separately for each ι. For each $\mathbf{x}'$, such that $\gamma_D(\mathbf{x}') = \iota$ and $\mathbf{h}^\top \mathbf{x}' > 0$ consider its reflection $\tilde{\mathbf{x}}$ along $\mathbf{h} = \mathbf{u}_j$ i.e. $\tilde{\mathbf{x}} = (I - 2\mathbf{u}_j \mathbf{u}_j^\top)\mathbf{x}'$.

Let us consider a sum over just these two points (instead of an expectation over all $\mathbf{x}, y$).

$$\sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x})\alpha(\mathbf{x}))\Lambda_k(\mathbf{x})\alpha(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z}$$

$$= \sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \left(\sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x})) + \epsilon(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))\sigma'(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))\frac{\mathbf{h}^\top\mathbf{x}\phi'(\mathbf{w}_\ell^\top\mathbf{x})}{\phi(\mathbf{w}_\ell^\top\mathbf{x})}\right)$$

$$\Lambda_k(\mathbf{x})\left(1+\frac{\epsilon\mathbf{h}^\top\mathbf{x}\phi'(\mathbf{w}_\ell^\top\mathbf{x})}{\phi(\mathbf{w}_\ell^\top\mathbf{x})}\right)\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z}$$

$$= \sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))\Lambda_k(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z} +$$

$$\frac{\epsilon\mathbf{h}^\top\mathbf{x}\phi'(\mathbf{w}_\ell^\top\mathbf{x})}{\phi(\mathbf{w}_\ell^\top\mathbf{x})}\left(\sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x})) - y^*(\mathbf{x})\hat{y}(\mathbf{x})\sigma'(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))\,\Lambda_k(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z}\right) \tag{4}$$

Case 1: $\iota > N/2$. As $\iota \neq j$, we have that $\tilde{\mathbf{x}}$ and $\mathbf{x}'$ share the same ODT path, i.e. $\gamma(\tilde{\mathbf{x}}) = \gamma(\mathbf{x}')$, and hence the same label $y^*(\mathbf{x})$. As the weights $\mathbf{w}_\lambda$ and the vector $\mathbf{z}$ are all orthogonal to $\mathbf{h} = \mathbf{u}_j$, all terms except $\mathbf{h}^\top\mathbf{x}$ are the same for both $\mathbf{x} = \mathbf{x}'$ and $\mathbf{x} = \tilde{\mathbf{x}}$. Further, $\mathbf{h}^\top\mathbf{x}' = -\mathbf{h}^\top\tilde{\mathbf{x}}$, thus making the second term in the above sum equal to zero. Thus Equation 4 becomes

$$\sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x})\alpha(\mathbf{x}))\Lambda_k(\mathbf{x})\alpha(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z} = \sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))\Lambda_k(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z} \tag{5}$$

Case 2: $\iota = N/2$. As $\iota = j$, the point $\mathbf{x}'$ and $\tilde{\mathbf{x}}$ differ in their ODT paths – one of them terminates in label node $N+1$ and the other terminates in label node $N+2$, and have different labels $y^*(\mathbf{x})$. Note that $\mathbf{z} = s\mathbf{u}_i$. Without loss of generality let j be in the left sub-tree of i, and $s = +1$. Thus $\mathbf{x}.\mathbf{z} < 0$ for both $\mathbf{x} = \mathbf{x}'$ and $\mathbf{x} = \tilde{\mathbf{x}}$. Further, we have $\mathbf{h} = \mathbf{u}_j$ where $j = N/2$ which is a leaf node, thus the label $y^*(\mathbf{x})$ for both $\mathbf{x} = \mathbf{x}'$ and $\mathbf{x} = \tilde{\mathbf{x}}$ is simply sign$(-\mathbf{h}\cdot\mathbf{x})$. Hence, $y^*(\mathbf{x})\mathbf{h}^\top\mathbf{x}(\mathbf{x}.\mathbf{z}) > 0$. Now note that as $0 < a < 1$ the value $\hat{y}(\mathbf{x})$ is also between 0 and 1. The term $\big(\sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))-y^*(\mathbf{x})\hat{y}(\mathbf{x})\sigma'(-y^*(\mathbf{x})\hat{y}(\mathbf{x})\big)$ is clearly positive for $y^*(\mathbf{x}) = -1$. For $y^*(\mathbf{x}) = +1$, the term $\big(\sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x})) - y^*(\mathbf{x})\hat{y}(\mathbf{x})\sigma'(-y^*(\mathbf{x})\hat{y}(\mathbf{x})\big)$ is also positive because $\sigma'(q) < \sigma(q)$ and $0 < y^*(\mathbf{x})\hat{y}(\mathbf{x}) < 1$ Equation 4 then becomes

$$\sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x})\alpha(\mathbf{x}))\Lambda_k(\mathbf{x})\alpha(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z} > \sum_{\mathbf{x}=\mathbf{x}',\tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x})\hat{y}(\mathbf{x}))\Lambda_k(\mathbf{x})\phi'(\mathbf{w}_k^\top\mathbf{x})y^*(\mathbf{x})a\mathbf{x}\cdot\mathbf{z} \tag{6}$$

Putting both the cases together, adding all the conditional expectation terms in Equation 3 using the inequalities in Equations 5 and 6, we get

$$-\nabla_{\mathbf{w}_k}\mathcal{L}(W + \epsilon P_\ell(\mathbf{h}))\cdot\mathbf{z} > -\nabla_{\mathbf{w}_k}\mathcal{L}(W)\cdot\mathbf{z}$$

This completes the proof for the main Theorem statement. Now, consider the case when $\mathbf{h}$ and $\mathbf{z}$ are swapped. Let the $dL \times dL$ Hessian of $\mathcal{L}$ w.r.t all the gating vectors $\mathbf{w}_1, \ldots, \mathbf{w}_L$ be denoted by H. We have that

$$\lim_{\epsilon\to 0}\frac{1}{\epsilon}\left[-\nabla_{\mathbf{w}_k}\mathcal{L}(W+\epsilon P_\ell(\mathbf{h})) + \nabla_{\mathbf{w}_k}\mathcal{L}(W)\right]\cdot\mathbf{z} = -P_\ell(\mathbf{h})^\top H P_k(\mathbf{z}).$$

To prove the same for $\mathbf{h}$ and $\mathbf{z}$ swapped, requires just seeing that $P_\ell(\mathbf{h})^\top H P_k(\mathbf{z}) = P_\ell(\mathbf{z})^\top H P_k(\mathbf{h})$, which is true by symmetry of H. □

A.3 Proof for Theorem 3.2

Theorem. *Let $\mathbf{h}$ be a vector in $\mathbb{R}^d$ that is orthogonal to the ODT decision vectors $\mathbf{u}_i$ for all $i \in \{0, 1, \ldots, n\}$. Let $\mathbf{w}_\lambda \cdot \mathbf{h} = 0$ for all $\lambda \in \{1, \ldots, L\}$. Let $\mathbf{w}_\lambda \cdot \mathbf{u}_j = 0$ for all $\lambda \in \{1, \ldots, L\}$ and $j \in \{N/2, \ldots, N\}$. The change in gradient of $\mathcal{L}$ w.r.t. $\mathbf{w}_k$ along $\mathbf{h}$ as $\mathbf{w}_\ell$ (for some $\ell \neq k$) moves along $-\mathbf{h}$ satisfies the following relation.*

$$\lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[-\nabla_{\mathbf{w}_k} \mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) + \nabla_{\mathbf{w}_k} \mathcal{L}(W) \Big] \cdot (-\mathbf{h}) > 0$$

Proof. Without loss of generality, we assume $a > 0$. Similar arguments with the exact same conclusion follows for $a < 0$ as well. Let $\epsilon > 0$ be an arbitrarily small number. All expressions involving ϵ below ignore quadratic and higher terms. Let $\alpha(\mathbf{x}) = \frac{\phi(\mathbf{w}_\ell^\top \mathbf{x} + \epsilon \mathbf{h}^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} = 1 + \frac{\epsilon \mathbf{h}^\top \mathbf{x} \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})}$.

$$-\nabla_{\mathbf{w}_k} \mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) \cdot (-\mathbf{h}) = \mathbb{E}_{\mathbf{x}} \left[\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \alpha(\mathbf{x})) \Lambda_k(\mathbf{x}) \alpha(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h}) \right]$$

For each $\mathbf{x}'$, such that $y^*(\mathbf{x}') = 1$ consider its reflection $\tilde{\mathbf{x}}$ along $\mathbf{u}_\zeta$, where $\zeta = \gamma_D(\mathbf{x}')$ is the leaf node in the ODT corresponding to $\mathbf{x}'$. i.e. $\tilde{\mathbf{x}} = (I - 2\mathbf{u}_\zeta \mathbf{u}_\zeta^\top)\mathbf{x}'$. Let us consider a sum over just these two points (instead of an expectation over all $\mathbf{x}, y$). By construction, $y^*(\tilde{\mathbf{x}}) = -1$.

$$\begin{aligned}
&\sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \alpha(\mathbf{x})) \Lambda_k(\mathbf{x}) \alpha(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h}) \\
&= \sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \left(\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) - \epsilon y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \sigma'(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) \frac{\mathbf{h}^\top \mathbf{x} \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} \right) \\
&\Lambda_k(\mathbf{x}) \left(1 + \frac{\epsilon \mathbf{h}^\top \mathbf{x} \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} \right) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h}) \\
&= \sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) \Lambda_k(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h}) + \\
&\qquad \frac{\epsilon \mathbf{h}^\top \mathbf{x} \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} \Big(\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) - y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \sigma'(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) \Big) \Lambda_k(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h})
\end{aligned} \tag{7}$$

Expanding the second term above alone, we get

$$\sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \frac{-\epsilon (\mathbf{h}^\top \mathbf{x})^2 \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} \Lambda_k(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) a \Big(\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) - y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \sigma'(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) \Big) y^*(\mathbf{x}) \tag{8}$$

Note that $\mathbf{x}$ and $\mathbf{x}'$ differ only in their projections along $\mathbf{u}_\zeta$ which is orthogonal to all gating vectors $\mathbf{w}_\lambda$, and also $\mathbf{h}$. Due to orthogonality assumptions, the term $\frac{-\epsilon (\mathbf{h}^\top \mathbf{x})^2 \phi'(\mathbf{w}_\ell^\top \mathbf{x})}{\phi(\mathbf{w}_\ell^\top \mathbf{x})} \Lambda_k(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) a$ is the same for both $\mathbf{x} = \mathbf{x}'$ and $\mathbf{x} = \tilde{\mathbf{x}}$, and also negative. Analysing the remaining term, we get

$$\begin{aligned}
\sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \Big(\sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) - y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \sigma'(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) \Big) y^*(\mathbf{x}) &= \sigma(-\hat{y}) - \hat{y}\sigma'(-\hat{y}) - \sigma(\hat{y}) - \hat{y}\sigma'(\hat{y}) \\
&= \sigma(-\hat{y}) - \sigma(\hat{y}) - 2\hat{y}\sigma'(\hat{y}) < 0
\end{aligned} \tag{9}$$

where we have used that $\hat{y}(\mathbf{x}') = \hat{y}(\tilde{\mathbf{x}}) = \hat{y} > 0$ and $y^*(\mathbf{x}') = -y^*(\tilde{\mathbf{x}}) = 1$. Substituting Equation 9 into 8, and the resulting term into Equation 7, we get

$$\begin{aligned}
&\sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x}) \alpha(\mathbf{x})) \Lambda_k(\mathbf{x}) \alpha(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h}) \\
&> \sum_{\mathbf{x}=\mathbf{x}', \tilde{\mathbf{x}}} \sigma(-y^*(\mathbf{x}) \hat{y}(\mathbf{x})) \Lambda_k(\mathbf{x}) \phi'(\mathbf{w}_k^\top \mathbf{x}) y^*(\mathbf{x}) a \mathbf{x} \cdot (-\mathbf{h})
\end{aligned}$$

and thus

$$-\nabla_{\mathbf{w}_k} \mathcal{L}(W + \epsilon P_\ell(\mathbf{h})) \cdot (-\mathbf{h}) > -\nabla_{\mathbf{w}_k} \mathcal{L}(W) \cdot (-\mathbf{h})$$

□

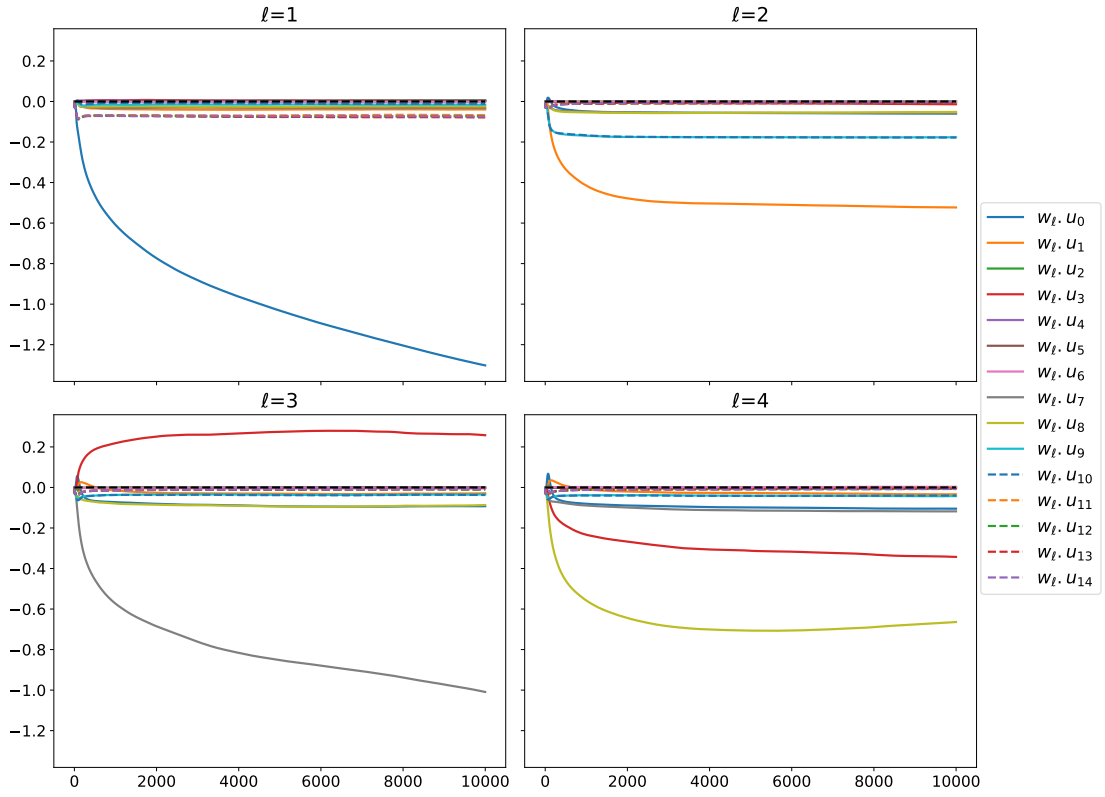

Figure 5: The parameter trajectory till 10000 epochs.

A.4 Single Path DLGN parameter trajectory

As an extension to Figure 2 in Section 3.3, Figure 5 shows the parameter trajectories when updated using gradient descent for 10000 epochs.

A.5 Heuristic Solution Predictor

Extra ODT Notation: Let Γ denote the set of root-to-label paths in the ODT. Each $\gamma \in \Gamma$ is a $(D+1)$-tuple, and is associated with a sign vector $\tau(\gamma) \in \{+1, -1\}^D$ recording the sequence of left/right decisions. For example, in the depth-4 tree of Figure 1, the path $\gamma = (0, 1, 3, 7, 15)$ has $\tau(\gamma) = (-1, -1, -1, -1)$ and label $y^*(\gamma) = +1$.

$$y^*(\mathbf{x}) = \sum_{\gamma \in \Gamma} y^*(\gamma) \prod_{p=1}^{D} \mathbf{1}\Big(\tau_p \, \mathbf{u}_{\gamma_p}^\top \mathbf{x} > 0\Big), \tag{10}$$

where τ_p denotes the p-th entry of $\tau(\gamma)$. By construction, exactly one path in Γ contributes non-zero mass for any given $\mathbf{x}$.

Heuristic Score Breakup: The Heuristic score for a given initialization W to move towards a solution $\tilde{W}$ is constructed using a node score $s_{i,\ell}(W)$ that captures the tendency of $\mathbf{w}_\ell$ to move along $\mathbf{u}_i$.

It is composed of 3 terms –

$$s_{i,\ell}(W) = \mathbf{w}_\ell \cdot \mathbf{u}_i + c_{\text{SO}} \sum_{k \neq \ell} (-\mathbf{w}_k \cdot \mathbf{u}_i) + c_{\text{HS}} A_{i,\ell}(W)$$

The first term above simply captures the current projection of $\mathbf{w}_\ell$ along $\mathbf{u}_i$. The second term captures the tendency of each layer $\mathbf{w}_\ell$ to move in a direction opposite to other layers $\mathbf{w}_k$ motivated from the

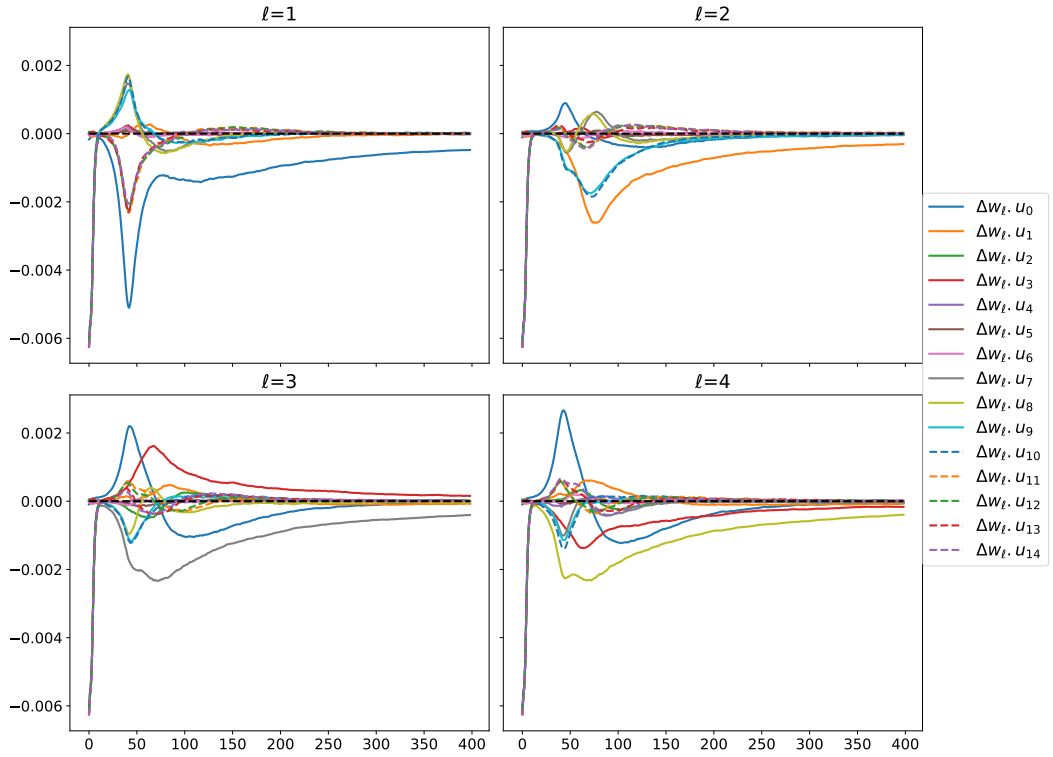

Figure 6: Gradient trajectory for a single-path DLGN under gradient descent.

self-opposition lens of Theorem 3.2. The last term $A_{i,\ell}(W)$ is motivated from the hierarchy seeking lens of Theorem 3.1, and is defined below. It is based on the component of other layer neurons along $\mathbf{u}_j$ for some descendants or ancestors j of i. The coefficients $c_{\text{SO}}, c_{\text{HS}}$ are positive constants that give weights to the self-opposition and hierarchy-seeking terms in the score $s_{i,\ell}$.

Hiearchy-Seeking Lens Score: The hierarchy-seeking lens score $A_{i,\ell}(W)$ is given as follows.

$$A_{i,\ell}(W) = \sum_{\gamma \in \Gamma_+ : i \in \gamma} \max_{\psi \in \mathbb{S}_D} \tau_{\psi(\ell)} \sum_{k \neq \ell} \tau_{\psi(k)} \mathbf{w}_k \cdot \mathbf{u}_{\gamma_{\psi(k)}}$$

where Γ_+ corresponds to the set of ODT paths terminating in a positively labelled node, and $\mathbb{S}_D$ is the set of permutations mapping $[D]$ to $[D]$. The τ referred to above is the $\{+1, -1\}$ valued decision vector associated with the path γ. A brief explanation of the above expression follows. Firstly, we consider all paths γ that contain the node i. For each such path γ, we consider all permutations ψ and add the contributions of other layer neurons along this permutation of the path γ. This corresponds to setting the measurement direction $\mathbf{z} = \mathbf{u}_i$ and considering all perturbation directions $\mathbf{h} = \mathbf{u}_j$ for ancestors or descendants j of i in Theorem 3.1.

For example, if $W = [\mathbf{0}, -\mathbf{u}_0, -\mathbf{u}_1, -\mathbf{u}_3]$, then $\mathbf{w}_1$ has a strong tendency to move along $-\mathbf{u}_7$ and make the DLGN represent the path $[0, 1, 3, 7, 15]$. This is reflected via a high negative score of -3 for $A_{7,1}(W)$. A breakdown of the calculation of $A_{7,1}(W)$ follows. $\gamma = [0, 1, 3, 7, 15]$ is the only path in Γ_+ that contains the node 7. Note that $\tau(\gamma) = [-1, -1, -1, -1]$. Out of all the $4!$ permutations of $\{1, 2, 3, 4\}$ the permutation $[4, 1, 2, 3]$ which calculates the alignment of W with $[-\mathbf{u}_7, -\mathbf{u}_0, -\mathbf{u}_1, -\mathbf{u}_3]$ scores the highest value of 3, and as $\tau_1 = -1$ the value of $A_{7,1}(W)$ becomes -3.

From Score to Solution Predictor: Finally, we convert the layer and tree node scores $s_{i,\ell}$ into solution scores using a simple additive rule. Let $\tilde{W}$ be one of the $D!2^{D-2}$ solutions, and $\upsilon = \upsilon(\tilde{W})$ be the compact D-tuple representation of $\tilde{W}$. The score given for solution $\tilde{W}$ at an initialization W

is given by

$$S(W;\tilde{W}) = \sum_{\ell=1}^{D} s_{v_\ell,\ell}(W) * \text{sign}\left(\tilde{W}_\ell \cdot \mathbf{u}_{v_\ell}\right)$$

A simple heuristic solution predictor $H(W)$ can be constructed from the solution score $S(W;\tilde{W})$ above by considering the argmax over all the $D!2^{D-2}$ solutions $\tilde{W}$. Below we discuss the experimental results for the $D = 4$ setting discussed in the paper.

Heuristic Score Performance: The full table giving the total percentage of initializations (out of 368) that have their eventual gradient descent solution predicted accurately by the heuristic predictor $H(W)$ for all values of $c_{\text{SO}}, c_{\text{HS}}$ is given below.

c_{SO} \ c_{HS}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	13	34	35	35	36	35	34	33	33	33
0.1	23	39	38	38	38	37	37	36	35	34
0.2	25	42	43	42	41	39	39	38	36	36
0.3	28	45	45	45	43	41	40	39	38	37
0.4	27	43	47	47	46	44	42	41	39	38
0.5	26	43	47	47	47	45	44	43	41	39
0.6	25	42	46	48	48	47	44	44	43	42
0.7	24	40	45	47	**49**	48	46	45	44	43
0.8	23	38	44	47	48	48	47	46	46	44
0.9	23	36	43	45	47	48	48	47	46	46

The best prediction accuracy of 49% for the 96-class prediction task is achieved with $c_{\text{SO}} = 0.7$ and $c_{\text{HS}} = 0.4$. Both the self-opposition terms and hierarchy seeking terms are essential to the score function – zeroing either of these results in a significantly worse accuracy. Setting both c_{SO} and c_{HS} equal to zero, corresponds to a baseline predictor that uses the proximity of the init W to a solution $\tilde{W}$ alone to make a prediction. This achieves a much lower accuracy of only 13%.

A.6 Performance comparison of different models on $D = 5$, $d = 100$ ODT dataset

We compare the performance of DLGN, ReLU and SVM on the depth D = 5, dimension d=100 ODT dataset containing 80,000 training points. Table 2 reports the best Test Accuracy (%) of each model.

Table 2: Comparison of Test Accuracy (%) of different models.

Dataset	**DLGN** ($L = 5$)	**ReLU**	**SVM**
d=100, D=5	94.92	95.74	63.27

Table 3: ReLU Hyper-parameters for L5 Dataset

Hyper-parameter	**Values**
Number of Hidden Layers	1, 2, 3, **4**, 5, 6
Number of Hidden Neurons	20, **40**, 60, 80, 100, 200
Weight Decay	0, 0.001, **0.005**, 0.01, 0.05
Batch Size	**80**, 160, 320
Learning Rate	0.001, **0.005**, 0.01, 0.05

A.7 Details of Experiments compute resources:

All experiments were performed on NVIDIA A100 GPU. A single run of the single-path trajectory analysis and the multi-path DLGN experiments can be completed in 10-20 minutes under this setup.

Table 4: ReLU Accuracy on Depth 5 ODT data for different number of layers

Hidden Layers	Best accuracy	Best width	Best Decay
1	70.2	100	0.05
2	91.2	40	0.05
3	94.8	40	0.05
4	95.7	40	0.05
5	95.6	40	0.05

Table 5: SVM Hyper-parameters for L5 dataset

Hyper-parameter	**Values**
C	0.1, **1**, 10, 100
degree	2, 3, 4
gamma	**auto**, scale
kernel	polynomial, **RBF**, sigmoid, linear

The heuristic solution predictor experiments take a considerably long time because there are a total of 2000*5 runs (2000 model initializations and 5 different datasets.)

A.8 HYPER-PARAMETERS FOR EXPERIMENTS

This section contains the hyperparameter search space for the performance comparison experiments. The best hyperparameters are highlighted in bold letters. Table 3 shows the list of hyper-parameters for ReLU experiments. Table 4 gives the details of the best performing ReLU network with different number of hidden layers. Note that the ReLU network with a single hidden layer performs much worse than deeper networks. We know from universality of single hidden layer neural networks that one hidden layer ReLU networks are capable of representing any function, and hence also the ODT labelling function. However, they fail even at achieving low training loss at any width. Current theories are incapable of explaining why this happens. One of the future goals of feature learning theory narrative that we aim to build here with DLGN and the 'layers as lenses' viewpoint is to be able to explain these results.

Table 5 shows the list of hyper-parameters for SVM experiments.