TASK-AWARE DISCRETIZATION OF DIFFERENTIABLE LOGIC GATE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentiable logic gate networks (DLGNs) enable gradient-based training of highly efficient Boolean networks by relaxing discrete logic gates during training and discretizing them for inference. Standard approaches make this discretization decision locally, typically through argmax selection and confidence- or entropy-based convergence criteria. We show that local discretization can be task-suboptimal even for globally optimal relaxed solutions, with high gate confidence providing no general guarantee, and derive bounds relating task-aware gate selection to tractable interventions in the relaxed network. Motivated by these results, we study first-order downstream task information for progressive discretization and characterize when this local approximation is reliable. Experiments on convolutional DLGNs reveal a strong locality dependence: first-order scores become unreliable when directly optimized over nonlocal interventions, but accurately assess local argmax decisions for progressive freezing.

1 INTRODUCTION

Differentiable logic gate networks (DLGNs) enable gradient-based training of highly efficient Boolean networks through continuous relaxations of logic gates (Petersen et al., 2022). At inference, the learned network can be reduced to Boolean operations, replacing conventional floating-point computation with simple logical operations Gerlach et al. (2025). This makes DLGNs particularly attractive for resource-constrained inference, and recent work has extended them to convolutional and recurrent architectures and more compact formulations (Petersen et al., 2024; Bührer et al., 2025; Gerlach et al., 2026; Wang et al., 2026).

This efficiency requires a transition from relaxed training to discrete inference. Standard DLGNs perform this transition by independently selecting the most probable gate at each neuron, while recent work has focused on reducing the resulting *discretization gap* through stochastic training, forward-aligned gradient estimators, temperature scheduling, and progressive freezing (Yousefi et al., 2026; Kim, 2026; Wang et al., 2026). These approaches address an important training–inference mismatch. However, a small discretization gap does not imply that the resulting discrete network is optimal for the downstream task.

This observation motivates a complementary question: *which discrete gates should be selected to minimize downstream task risk?* Prior work typically discretizes DLGNs by selecting the most probable gate at each neuron through argmax. This local probabilistic decision need not align with the task loss of the complete discrete network. Indeed, we show that even globally optimal relaxed DLGNs with arbitrarily concentrated gate distributions can retain a non-vanishing discrete optimality gap under componentwise argmax. Thus, neither relaxed optimality nor local gate confidence certifies task-optimal discretization.

We address this mismatch by formulating discretization directly in terms of downstream task risk. Conditional expectations provide an ideal task-aware selection rule but require averaging over discrete downstream completions. We relate this criterion to interventions in the relaxed network and derive an efficient first-order approximation whose candidate scores reuse gradients from ordinary backpropagation. Our analysis characterizes the approximation error through downstream mismatch and intervention locality. Empirically, directly optimizing the first-order surrogate can select nonlocal interventions for which its predictions become unreliable, whereas it accurately assesses local

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.

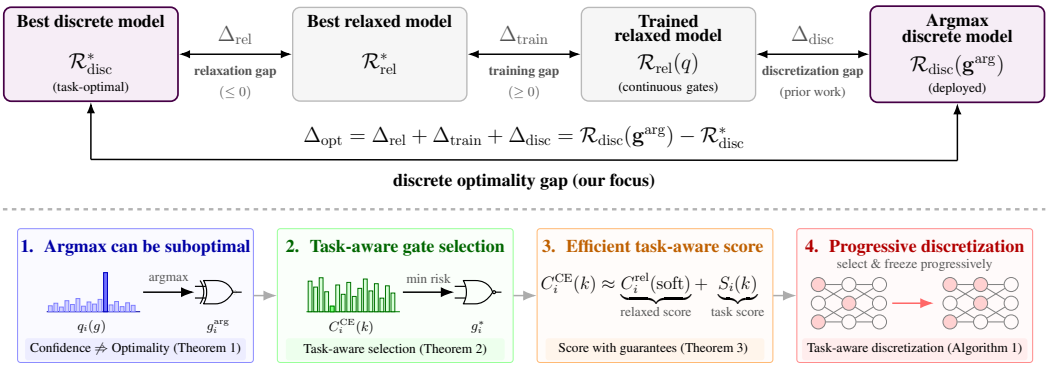


Figure 1: Overview of the discrete optimality gap and our task-aware discretization framework. The top row decomposes the gap between the deployed argmax network and the task-optimal discrete network; the bottom row summarizes the theoretical motivation, task-aware gate selection, its efficient first-order approximation, and progressive discretization.

argmax interventions. This motivates a progressive discretization procedure that uses downstream task information to determine when argmax decisions can be safely committed.

Our main contributions are:

- **A task-risk view of discretization.** We distinguish the conventional discretization gap from the *discrete optimality gap*, which measures the excess task risk of argmax relative to the best discrete network, and relate the different sources of suboptimality through an exact risk decomposition.
- **Limits of argmax.** We prove that globally optimal DLGNs can have arbitrarily concentrated gate distributions while their argmax discretizations remain bounded away from discrete optimality.
- **Tractable task-aware interventions.** We derive task-aware gate selection from conditional risk and bound the error of an efficient first-order approximation in terms of downstream mismatch and intervention locality. The resulting scores reuse gradients from ordinary training.
- **Locality-aware empirical analysis.** We show that first-order scores reliably assess local argmax interventions, while directly optimizing them can select nonlocal interventions with large approximation errors. We use this insight for task-aware progressive freezing and compare against existing discretization strategies.

2 Related Work

Differentiable logic networks. Petersen et al. (2022) introduced differentiable logic gate networks by continuously relaxing the choice among Boolean gates and selecting the most probable gate for discrete inference. Subsequent work extends this framework to convolutional architectures (Petersen et al., 2024), learnable connections (Fojcik et al., 2026), more efficient parameterizations (Rüttgers et al., 2026; Gerlach et al., 2026), and FPGA-oriented lookup-table models (Bührer et al., 2026).

Closing the discretization gap. Prior work aligns relaxed training and discrete inference through stochastic gate selection with straight-through Gumbel–Softmax (Kim, 2023; Yousefi et al., 2026; Gerlach et al., 2026), discrete forward passes with approximate gradients (Bacellar et al., 2024), and adaptive temperature scheduling (Kim, 2026). These methods modify training; we instead study discretization through its downstream task effect.

Task-aware discretization. A closely related problem arises in differentiable neural architecture search, where continuous operation weights are commonly converted into a discrete architecture by selecting their largest values. Wang et al. (2021) show that these weights need not reflect an operation's contribution to task performance and instead propose perturbation-based, task-aware architecture selection. Within DLGNs, Wang et al. (2026) introduce adaptive discretization, progressively freezing layers once their entropy stabilizes, allowing later layers to adapt to discrete inputs.

Our approach combines task-aware evaluation with progressive discretization: rather than freezing gates based on local confidence, we use their predicted downstream task effect to determine when discrete decisions can be safely committed.

3 Background on Differentiable Logic Gate Networks

A DLGN is organized into layers of differentiable gate neurons. Each neuron i receives two activations $a_{i,1}$ and $a_{i,2}$ from the preceding layer and learns which Boolean function from a candidate set $\mathcal{G}_i$ should combine them. For two-input gates, $\mathcal{G}_i$ typically contains all 16 Boolean functions. During training, this discrete choice is relaxed to a weighted mixture,

$$a_i = \sum_{g \in \mathcal{G}_i} q_{i,g}\, g(a_{i,1}, a_{i,2}), \qquad q_{i,g}(\tau) = \frac{\exp(\theta_{i,g}/\tau)}{\sum_{h \in \mathcal{G}_i} \exp(\theta_{i,h}/\tau)},$$

where $\theta_{i,g}$ are trainable logits, g represents a continued relaxed surrogate of the underlying gate and the temperature $\tau > 0$ controls the concentration of the gate distribution. After training, each differentiable neuron is converted into a Boolean gate for efficient discrete inference.

Across the N neurons, the learned gate distributions induce a factorized distribution over discrete networks, $q = \prod_{i \le N} q_i$. Standard discretization independently selects each marginal mode, $g_i^{\arg} = \arg\max_{g \in \mathcal{G}_i} q_{i,g}$, making $\mathbf{g}^{\arg}$ a maximum-probability assignment under q, up to ties. Standard discretization therefore selects gates according to their learned probabilities rather than their effect on task risk. This distinction motivates a closer separation between existing notions of training–inference mismatch and the quality of the resulting discrete gate assignment.

3.1 Discretization Gaps

Let F_q denote the prediction of the relaxed network under gate distributions q, and let $f_{\mathbf{g}}$ denote the discrete network obtained from an assignment $\mathbf{g} = (g_1, \ldots, g_N)$. For data $(X, Y) \sim \mathcal{D}$ and task loss ℓ, its population risks are $\mathcal{R}_{\mathrm{rel}}(q) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(F_q(X), Y)]$, and $\mathcal{R}_{\mathrm{disc}}(\mathbf{g}) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(f_{\mathbf{g}}(X), Y)]$. Discretizing a trained DLGN can degrade task performance for several distinct reasons. Let $\mathcal{R}^*_{\mathrm{rel}} := \min_q \mathcal{R}_{\mathrm{rel}}(q)$, and $\mathcal{R}^*_{\mathrm{disc}} := \min_{\mathbf{g}} \mathcal{R}_{\mathrm{disc}}(\mathbf{g})$, denote the optimal risks attainable by the relaxed and discrete model classes, respectively. For a trained gate distribution q and its componentwise argmax assignment $\mathbf{g}^{\arg}$, we distinguish

$$\Delta_{\mathrm{train}} := \mathcal{R}_{\mathrm{rel}}(q) - \mathcal{R}^*_{\mathrm{rel}}, \quad \Delta_{\mathrm{rel}} := \mathcal{R}^*_{\mathrm{rel}} - \mathcal{R}^*_{\mathrm{disc}}, \quad \Delta_{\mathrm{disc}} := \mathcal{R}_{\mathrm{disc}}(\mathbf{g}^{\arg}) - \mathcal{R}_{\mathrm{rel}}(q).$$

The training gap measures suboptimality within the relaxed model class, the relaxation gap captures the difference between the best relaxed and discrete models, and the discretization gap measures the change in task risk when the trained relaxed model is converted to its argmax network. The training gap is non-negative by definition. Moreover, since the relaxed class contains every discrete network, $\Delta_{\mathrm{train}} \ge 0$ and $\Delta_{\mathrm{rel}} \le 0$, while Δ_{disc} is signed.

These gaps are distinct from the question of whether argmax produces a task-optimal discrete network. We define the *discrete optimality gap* as

$$\Delta_{\mathrm{opt}} := \Delta_{\mathrm{rel}} + \Delta_{\mathrm{train}} + \Delta_{\mathrm{disc}} = \mathcal{R}_{\mathrm{disc}}(\mathbf{g}^{\arg}) - \mathcal{R}^*_{\mathrm{disc}} \ge 0.$$

Thus, a suboptimal deployed network may result from imperfect relaxed training, from the relaxation itself, or from converting the learned relaxed model into a discrete gate assignment.

Kim (2026) further decomposes the conventional discretization gap into a *selection gap* from replacing soft gate mixtures by argmax gates and a *computation gap* from subsequently evaluating them as Boolean functions. The computation gap vanishes for binary inputs, where the continuous gate extensions coincide with their truth tables. Our focus is complementary: even if this training–inference gap vanishes, argmax need not minimize risk among discrete networks.

4 A Framework for Task-Aware Discretization

Prior work has related accurate argmax discretization to the concentration of learned gate distributions: DLGN neurons are observed to converge toward individual gates, while diffuse gate distributions have been associated with discretization errors Petersen et al. (2022; 2024); Yousefi et al.

(2026); Rüttgers et al. (2026); Wang et al. (2026). This suggests that argmax should become reliable once the learned distributions are sufficiently concentrated. We show that local concentration alone cannot provide such a guarantee, even when the relaxed network is globally optimal.

Let δ_g denote the distribution assigning probability one to gate $g \in \mathcal{G}$. We call a continuous function D on gate distributions a *local dispersion measure* if $D(\delta_g) = 0$ for every $g \in \mathcal{G}$. This includes, for example, Shannon entropy, Gini impurity, and the non-modal probability $1 - \max_g q(g)$.

Theorem 1 (Local concentration does not guarantee task-optimal argmax)**.** *Let ℓ be a loss and let $\mathcal{Z}$ denote the prediction space of the relaxed network. Suppose there exist a data distribution $\mathcal{D}$ and predictions $z_0, z^* \in \mathcal{Z}$ such that*

$$z^* \in \arg\min_{z \in \mathcal{Z}} \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(z, Y)] \quad \textit{and} \quad \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(z_0, Y) - \ell(z^*, Y)] = c > 0. \tag{1}$$

Then there exists a sequence of DLGNs with $N \to \infty$ trainable neurons and globally optimal relaxed gate distributions q^* such that,

$$\Delta_{\text{opt}} = c, \quad \textit{and} \quad \max_{1 \le i \le N} D(q_i^*) \longrightarrow 0. \tag{2}$$

The proof is found in Sec. A.1. This theorem highlights a limitation of purely local confidence measures, that is local concentration of the gate distributions does not, by itself, control their downstream task-level effect. Concrete instances of Theorem 1 for cross-entropy and squared loss are given in Sec. A.2.

4.1 Optimal Task-Aware Discretization via Conditional Expectations

The previous result shows that local concentration does not determine which discrete gate assignment minimizes task risk. We therefore consider a discretization rule that evaluates gate choices directly through their downstream loss.

Suppose that gates $g_1, \ldots, g_{i-1}$ have already been fixed. For a candidate gate $k \in \mathcal{G}$ at neuron i, define its conditional task risk as

$$C_i^{\text{CE}}(k) := \mathbb{E}_{\mathbf{G}_{>i}\sim q_{>i}}[\mathcal{R}_{\text{disc}}(\mathbf{g}_{<i}, k, \mathbf{G}_{>i})], \tag{3}$$

where $\mathbf{g}_{<i} = (g_1, \ldots, g_{i-1})$ and $q_{>i} = \prod_{j>i} q_j$. Selecting $g_i^{\text{CE}} \in \arg\min_{k\in\mathcal{G}} C_i^{\text{CE}}(k)$ sequentially is the method of conditional expectations (Motwani & Raghavan, 1996) and yields a fully discrete assignment $\mathbf{g}^{\text{CE}}$. Since each decision minimizes the conditional expected risk, $\mathbf{g}^{\text{CE}}$ satisfies

$$\mathcal{R}^*_{\text{disc}} \le \mathcal{R}_{\text{disc}}(\mathbf{g}^{\text{CE}}) \le \mathbb{E}_{\mathbf{G}\sim q}[\mathcal{R}_{\text{disc}}(\mathbf{G})]. \tag{4}$$

A proof is given in Sec. A.3. Evaluating Eq. (3) exactly requires averaging over the exponentially many discrete completions of the remaining network. We therefore seek a quantity that can be evaluated directly through the trained relaxed DLGN.

For a fixed prefix $\mathbf{g}_{<i}$ and candidate gate k, let $\mu_{i,k}(X) := \mathbb{E}_{\mathbf{G}_{>i}\sim q_{>i}}[f_{\mathbf{g}_{<i},k,\mathbf{G}_{>i}}(X)]$ denote the mean prediction and $\sigma^2_{i,k}(X) := \mathbb{E}_{\mathbf{G}_{>i}\sim q_{>i}}[\|f_{\mathbf{g}_{<i},k,\mathbf{G}_{>i}}(X) - \mu_{i,k}(X)\|^2]$, the variance over discrete downstream completions. The corresponding relaxed intervention is $F_{i,k}(X) := F_{\mathbf{g}_{<i},k,q_{>i}}(X)$, with risk $C_i^{\text{rel}}(k) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(F_{i,k}(X), Y)]$. In general, $F_{i,k} \neq \mu_{i,k}$ because relaxed propagation need not equal averaging over discrete completions. The following result quantifies the resulting approximation error.

Theorem 2 (Approximation of conditional task risk)**.** *Assume that ℓ is L-Lipschitz and define $V_{i,k} := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\sigma_{i,k}(X)]$ and $P_{i,k} := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\|\mu_{i,k}(X) - F_{i,k}(X)\|]$. Then*

$$\left|C_i^{\text{CE}}(k) - C_i^{\text{rel}}(k)\right| \le L\,(V_{i,k} + P_{i,k}). \tag{5}$$

The proof is given in Sec. A.3. The two terms in Eq. (5) identify distinct sources of approximation error. $V_{i,k}$ measures variability among discrete downstream completions, whereas $P_{i,k}$ measures the mismatch between their mean prediction and the deterministic relaxed forward pass. Hence relaxed interventions approximate the ideal conditional task criterion whenever both downstream variability and relaxation-induced propagation mismatch are small.

4.2 First-Order Task-Aware Gate Selection

Although relaxed interventions avoid averaging over discrete downstream completions, evaluating $C_i^{\mathrm{rel}}(k)$ for every candidate still requires many forward passes. We therefore estimate the task effect of selecting gate k from local first-order information. Let $a_i(\cdot)$ be the relaxed output of neuron i and $a_{i,k}(\cdot)$ its output under gate k. We define the first-order score

$$S_i(k) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\frac{\partial \ell}{\partial a_i}\left(a_{i,k}(X) - a_i(X)\right)\right].$$

Negative scores predict a decrease in task risk, while positive scores predict an increase.

Theorem 3 (First-order task-aware approximation)**.** *Assume the conditions of Theorem 2 and that the sample-wise downstream loss is twice differentiable with respect to the scalar activation a_i and upper bounded by $M_i > 0$. For each $k \in \mathcal{G}$, let $\epsilon_{i,k} := L_\ell(V_{i,k}+P_{i,k}) + \frac{M_i}{2}\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\Delta a_{i,k}(X)^2\right]$ and $\epsilon_i := \max_{k\in\mathcal{G}} \epsilon_{i,k}$. With the first-order choice $k_i^{\mathrm{FO}} \in \arg\min_{k\in\mathcal{G}} S_i(k)$, we then obtain*

$$\left|C_i^{\mathrm{CE}}(k) - \left[C_i^{\mathrm{rel}}(\mathrm{soft}) + S_i(k)\right]\right| \le \epsilon_{i,k}, \quad \textit{and thus} \quad C_i^{\mathrm{CE}}(k_i^{\mathrm{FO}}) - \min_{k\in\mathcal{G}} C_i^{\mathrm{CE}}(k) \le 2\epsilon_i. \tag{6}$$

Moreover, relative to $k_i^{\arg} \in \arg\max_{k\in\mathcal{G}} q_i(k)$, let $\Gamma_i := S_i(k_i^{\arg}) - S_i(k_i^{\mathrm{FO}})$. Then

$$\Gamma_i > 2\epsilon_i \quad \Longrightarrow \quad C_i^{\mathrm{CE}}(k_i^{\mathrm{FO}}) < C_i^{\mathrm{CE}}(k_i^{\arg}). \tag{7}$$

The proof can be found in Sec. A.5. The approximation error is separated into two sources. The term $L(V_{i,k} + P_{i,k})$ measures the error incurred by replacing discrete downstream completions with the relaxed network, while the curvature term controls the error of the local first-order approximation.

4.3 Task-Aware Progressive Freezing

The population score $S_i(k)$ is not directly available during training. We estimate it on the current minibatch $\mathcal{B}_t$ by averaging $s_{i,k} = (\partial\ell/\partial a_i)(a_{i,k} - a_i)$, yielding $\widehat{S}_{i,t}(k) = |\mathcal{B}_t|^{-1}\sum_{b\in\mathcal{B}_t} s_{i,k}^{(b)}$. The downstream sensitivities $\partial\ell/\partial a_i$ are already available from backpropagation, while candidate gate outputs can be evaluated locally without additional downstream forward passes.

For progressive discretization, we use this task information conservatively to assess the gate already preferred by the learned distribution. Let $k_i^{\arg} = \arg\max_{k\in\mathcal{G}_i} q_i(k)$. Because minibatch scores are noisy, we maintain an exponential moving average (EMA) of its predicted freeze shock,

$$\mu_i^{(t)} = \beta\mu_i^{(t-1)} + (1-\beta)\widehat{S}_{i,t}(k_i^{\arg}).$$

A neuron is frozen to $k_i^{\arg}$ once this score remains negative for P consecutive evaluations. Thus, unlike confidence- or entropy-based criteria, freezing is triggered by the predicted downstream task effect of committing the argmax gate. The procedure is summarized in Algorithm 1.

5 Experiments

Our experiments address four questions. First, can local statistics predict the task effect of discretizing individual neurons? Second, why does directly optimizing the first-order task-aware surrogate become unreliable? Third, can first-order task information instead provide a robust criterion for progressive argmax freezing? Finally, how does this procedure compare with existing discretization strategies? We study these questions on convolutional DLGNs, beginning with the local intervention analysis in Fig. 2.

Experimental setup. We evaluate on CIFAR-10, using 45k training and 5k validation examples, cross-entropy loss, and GroupSum aggregation to obtain class logits. We study two convolutional DLGN configurations chosen to closely follow established architectures and training procedures. *Compact-S* follows the method described in (Wang et al., 2026), using uniform thresholding, Adam optimization, and connection learning through resampling; models are trained for 600k steps. *TreeLogicNet-S* follows the TreeLogicNet architecture Petersen et al. (2024) , using distributive thresholding, AdamW with weight decay 0.002, and 200k training steps. Unless stated otherwise, we evaluate five independent seeds per architecture. The two settings test whether our observations persist across distinct DLGN architectures and training procedures. Further architectural and training details are provided in Appendix B.

Algorithm 1 Task-Aware Progressive Freezing

Require: scoring interval K, EMA factor β, patience P

Train the relaxed network for an initial burn-in period
for each subsequent training step t **do**
  Perform a standard training step
  if $t \bmod K = 0$ **then**
    for each unfrozen decision i **do**
      $k_i^{\arg} \leftarrow \arg\max_{k \in \mathcal{G}_i} q_i(k)$
      Compute $\widehat{S}_{i,t}(k_i^{\arg})$ and update μ_i
      if $\mu_i < 0$ for P evaluations **then**
        Freeze i to $k_i^{\arg}$
      end if
    end for
  end if
end for

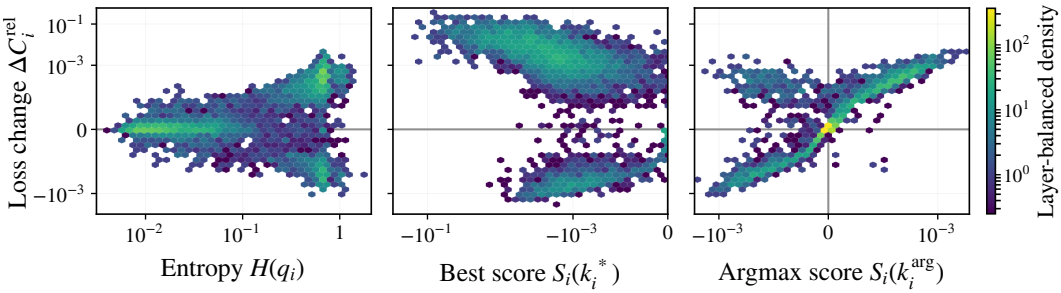


Figure 2: **Predicting freeze effects.** Measured loss change from freezing to the argmax gate versus gate entropy (left), from freezing to the score-selected gate $k_i^* = \arg\min_k S_i(k)$ versus its first-order score (center), and from freezing to the argmax gate versus its first-order score (right). Score selection is poorly aligned with the realized effect, whereas the argmax freeze shock is strongly predictive.

5.1 Predicting Discretization Effects

Fig. 2 compares three local quantities with the realized task-loss change from fixing a single gate while keeping the remaining network relaxed, defined as $\Delta C_i^{\text{rel}}(k) := C_i^{\text{rel}}(k) - C_i^{\text{rel}}(\text{soft})$. For entropy and the argmax freeze shock, we measure this change after freezing to $k_i^{\arg} = \arg\max_k q_i(k)$; for task-aware selection, we instead freeze to $k_i^* = \arg\min_k S_i(k)$. Gate entropy shows little relation to the resulting argmax freeze effect. Surprisingly, selecting the gate with the lowest first-order score does not improve this correspondence: predicted beneficial interventions can still substantially increase the loss. In contrast, the first-order score of the argmax gate is strongly aligned with the realized effect, including its sign.

We quantify these relationships using Spearman rank correlation. As summarized in Tab. 1, the qualitative pattern persists across layers, checkpoints, seeds, and both architectures. Argmax selections are consistently strongly positively correlated with the realized freeze effect, whereas entropy is generally weaker and score selections are substantially less consistent, including strong negative correlations in several layers. Thus, first-order task information is useful for *assessing* the argmax intervention but becomes unreliable when directly optimized over candidate gates. We investigate this discrepancy in Fig. 3, focusing on the locality of the corresponding interventions.

5.2 Locality of First-Order Interventions

We next investigate why the first-order score is predictive for argmax interventions but unreliable when directly optimized over candidate gates. Fig. 3a first examines how far a discrete intervention moves a neuron from the relaxed operating point. We quantify this by its squared activation displacement $D_i(k) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\Delta a_{i,k}^2(X)]$, where $\Delta a_{i,k}(\cdot) = a_{i,k}(\cdot) - a_i(\cdot)$, $a_i(\cdot)$ is the relaxed

Table 1: **Layer-wise Spearman correlations** between local discretization criteria and realized freeze effects. Values are mean ± sample standard deviation across five seeds, after averaging checkpoints within each seed (four for Compact-S and five for TreeLogicNet-S).

	Compact-S			TreeLogicNet-S		
Layer	**Entropy**	**Score**	**Argmax**	**Entropy**	**Score**	**Argmax**
Conv 1	$+0.301 \pm 0.045$	-0.576 ± 0.038	$+0.742 \pm 0.047$	$+0.450 \pm 0.082$	-0.356 ± 0.106	$+0.583 \pm 0.081$
Conv 2	$+0.370 \pm 0.053$	-0.625 ± 0.021	$+0.792 \pm 0.053$	$+0.257 \pm 0.063$	-0.291 ± 0.056	$+0.766 \pm 0.039$
Conv 3	$+0.281 \pm 0.018$	-0.304 ± 0.074	$+0.868 \pm 0.021$	$+0.076 \pm 0.016$	$+0.042 \pm 0.049$	$+0.762 \pm 0.034$
Conv 4	$+0.048 \pm 0.045$	-0.242 ± 0.081	$+0.991 \pm 0.004$	$+0.024 \pm 0.019$	$+0.454 \pm 0.045$	$+0.676 \pm 0.013$

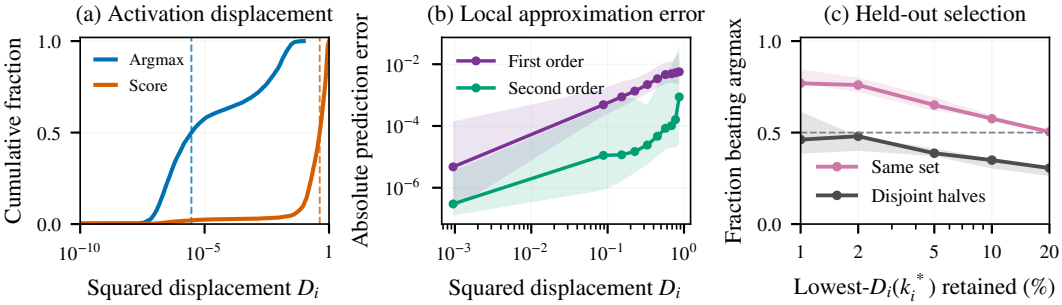


Figure 3: **Locality of first-order interventions.** (a) Score selection produces substantially larger activation displacements than argmax, with median squared displacements of 0.42 and 2.9×10^{-6}, respectively. (b) Approximation error increases with displacement for both first- and second-order scores, with median within-neuron Spearman correlations of 0.88 and 0.80, respectively; second order substantially reduces the typical error. (c) The apparent advantage of selecting low-displacement task-aware interventions disappears when gate selection and evaluation use disjoint validation data.

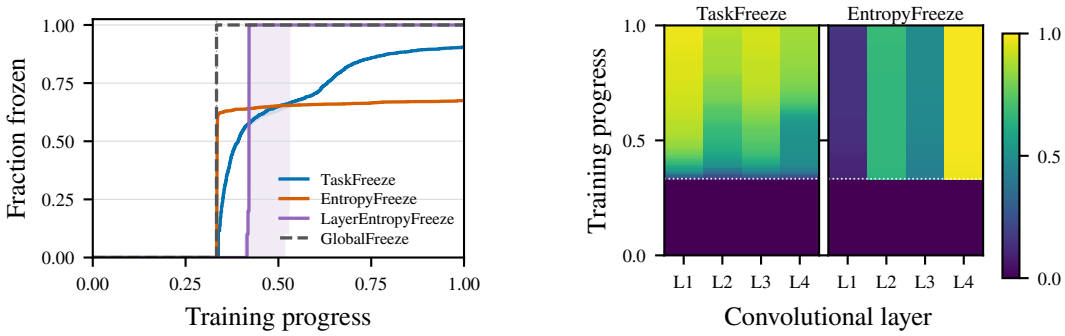
output of neuron i and $a_{i,k}(\cdot)$ its output under gate k. Argmax and score selection operate in strikingly different regimes: their median displacements are 2.9×10^{-6} and 0.42, respectively. Thus, while argmax typically constitutes a highly local intervention, directly minimizing $S_i(k)$ can select gates far from the point at which the first-order approximation is evaluated.

Fig. 3b relates displacement to the approximation error. For first order, we measure the prediction error $|\Delta C_i^{\mathrm{rel}}(k) - S_i(k)|$; for second order, we add the directional curvature correction, that is $|\Delta C_i^{\mathrm{rel}}(k) - S_i(k) - \frac{1}{2}\Delta a_{i,k}^{\top} H_i \Delta a_{i,k}|$. Across the 16 candidate gates of each neuron, approximation error grows strongly with $D_i(k)$, with median within-neuron Spearman correlations of 0.88 and 0.80 for first and second order, respectively. Second order substantially reduces the typical error, confirming that curvature explains an important part of the first-order residual, but the remaining displacement dependence shows that nonlocal interventions remain difficult to approximate. This is consistent with the curvature term in Theorem 3, whose bound grows with $D_i(k)$.

Finally, Fig. 3c asks whether restricting score selection to local interventions can recover a reliable advantage over argmax. Among neurons where the two rules disagree, we retain increasing fractions with the smallest $D_i(k_i^*)$. When the same validation data are used for selection and evaluation, the most local score-selected choices appear to outperform argmax frequently. This advantage disappears when gates are selected on one validation half and evaluated on the other. Thus, although locality governs approximation quality, restricting selection to local interventions does not yield a robust task-aware replacement rule. We therefore use first-order information conservatively to determine when argmax decisions can be committed.

5.3 Progressive Freezing Dynamics

We compare our task-aware neuron-wise criterion, *TaskFreeze*, with three freezing baselines. *EntropyFreeze* freezes individual neurons once their gate entropy has converged; *GlobalFreeze* freezes all neurons simultaneously at the discretization start and *LayerEntropyFreeze* applies the layer-wise



(a) Global freezing dynamics.

(b) Layer-wise freezing dynamics.

Figure 4: **Progressive freezing dynamics on Compact-S.** (a) Fraction of frozen neurons throughout training for different discretization criteria. (b) Layer-wise frozen fractions show that TaskFreeze progressively commits individual decisions, whereas EntropyFreeze induces a markedly different layer-dependent schedule.

Table 2: **Final discrete CIFAR-10 test accuracy (%).** Results are reported at the 400k discretization start for Compact models and the 25k start for TreeLogicNet models. For Compact-S, we compare continued and stopped connection resampling after discretization begins. Values are mean $\pm$ sample standard deviation across available seeds.

	Compact-S		Compact-M	TreeLogicNet-S	TreeLogicNet-M
Method	**Resampling**	**Stopped**	**Resampling**		
TaskFreeze (ours)	66.25 ± 0.27	65.93 ± 0.15	69.34 ± 0.46	60.76 ± 0.46	71.31 ± 0.04
EntropyFreeze (ours)	64.11 ± 0.53	63.88 ± 0.76	68.47 ± 0.54	59.97 ± 0.84	70.95 ± 0.23
GlobalFreeze (ours)	66.41 ± 0.21	66.12 ± 0.37	69.26 ± 0.21	60.50 ± 0.52	71.15 ± 0.04
LayerEntropyFreeze	64.10 ± 0.45	65.99 ± 0.17	67.67 ± 0.93	60.63 ± 0.55	71.44 ± 0.01
FinalArgmax	64.63 ± 0.61	64.63 ± 0.61	67.67 ± 0.39	59.29 ± 0.78	70.49 ± 0.27
CAGE	63.32 ± 0.63	64.11 ± 0.39	68.23 ± 0.12	59.98 ± 0.35	71.05 ± 0.23

entropy criterion of Wang et al. (2026). We additionally use *FinalArgmax* for the standard setting in which training remains relaxed and argmax discretization is applied only after training.

Fig. 4a shows the fraction of frozen neurons on Compact-S. GlobalFreeze commits the complete network at the discretization start, while LayerEntropyFreeze similarly produces an abrupt transition once its layer-wise criterion is met. EntropyFreeze freezes a large fraction of neurons early but subsequently changes little. TaskFreeze instead produces a gradual trajectory: individual argmax decisions become eligible throughout the remaining training period, resulting in a substantially smoother transition toward the discrete network.

The layer-wise view in Fig. 4b further highlights the difference between the two neuron-wise criteria. EntropyFreeze induces a strongly layer-dependent pattern, freezing most neurons in deeper layers early while leaving substantially fewer decisions frozen in the first layer. TaskFreeze distributes freezing more progressively across depth and time, with the final layer remaining more conservative. Thus, task-level eligibility does not simply reproduce local entropy convergence, but induces a distinct discretization schedule based on the predicted downstream effect of each argmax decision.

5.4 Final Discrete Performance

Finally, Tab. 2 compares the resulting discrete networks on the CIFAR-10 test set. TaskFreeze is consistently competitive across both architecture families and model sizes, and improves over EntropyFreeze in all reported settings. Interestingly, the simple GlobalFreeze baseline is similarly strong, slightly outperforming TaskFreeze on Compact-S while remaining comparable on the other models. This suggests that exposing the network to discrete computation during training is itself an important factor, while carefully scheduling individual freeze decisions does not consistently improve final accuracy.

Both neuron-wise criteria are compared with the layer-wise entropy strategy, fully relaxed training with final argmax discretization, and CAGE. Overall, early or progressive argmax freezing generally improves over applying argmax only after relaxed training, while no single freezing schedule consistently dominates across architectures. Detailed results across discretization start points and connection-resampling settings are provided in Sec. B.5.

6 Discussion

Our results reveal an important distinction between using task information to *select* and to *assess* a discrete intervention. Although the first-order score approximates the downstream effect of fixing a gate, directly minimizing this surrogate preferentially selects large interventions for which the local approximation is least reliable. Second-order corrections substantially reduce the typical prediction error, confirming an important role of curvature, but remain strongly dependent on intervention magnitude. In contrast, the argmax gate learned by the relaxed network is typically close to the relaxed operating point, making its first-order freeze shock a reliable local diagnostic. These findings persist across two architecture families, multiple layers and training checkpoints, five-seed S-model experiments, and different discretization starts. For Compact-S, they also persist with both continued and stopped connection resampling.

Motivated by this distinction, TaskFreeze leaves gate selection to the learned distribution and uses task information only to determine when a decision is favorable to commit. Its gradual freezing dynamics differ markedly from entropy-based criteria, yet its final accuracy is often comparable to the much simpler GlobalFreeze baseline. This suggests that adaptation to discrete computation during continued training is itself an important source of performance, while fine-grained scheduling does not universally improve the final discrete model. Developing reliable task-aware gate replacement beyond the local regime therefore remains an open problem, potentially requiring nonlocal or jointly optimized discretization criteria.

7 Conclusion

We studied DLGN discretization from the perspective of downstream task risk. We showed that even globally optimal relaxed networks with highly concentrated gate distributions can remain suboptimal under componentwise argmax, and derived conditional-risk and first-order criteria for task-aware gate decisions. Empirically, directly optimizing the first-order surrogate can leave its local regime, whereas the same signal accurately assesses the naturally local argmax intervention. TaskFreeze exploits this distinction for progressive discretization and is competitive across two convolutional DLGN families. More broadly, our results identify intervention locality as a central consideration for task-aware discretization and motivate methods that can account for nonlocal interactions between discrete decisions.

AI Use Statement

Generative AI tools were used to assist with drafting and editing manuscript text, developing and checking theoretical arguments and counterexamples, designing diagnostic experiments, interpreting experimental results, organizing and reviewing literature, and checking LaTeX structure. AI-assisted mathematical arguments, experimental analyses, citations, and code were independently reviewed and verified by the authors. The authors take responsibility for the final content of the work, including all text, claims, results, and artifacts produced with the assistance of generative AI.

Reproducibility statement

Complete proofs are provided in Sec. A. Sec. B specifies the model architectures, optimization and freezing hyperparameters, diagnostic protocol, and additional experiments. Code required to reproduce the reported experiments is included in the anonymized supplementary material.

REFERENCES

Alan Bacellar, Zachary Susskind, Mauricio Breternitz Jr, Eugene John, Lizy Kurian John, Priscila Machado Vieira Lima, and Felipe M.G. França. Differentiable weightless neural networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235. PMLR, 2024.

Simon Bührer, Andreas Plesner, Till Aczel, and Roger Wattenhofer. Recurrent deep differentiable logic gate networks. In *ACM MobiCom 2025 Workshop on Edge and Mobile Foundation Models (EdgeFM)*, 2025. doi: 10.1145/3737902.3768357.

Simon Bührer, Andreas Plesner, Till Aczel, and Roger Wattenhofer. BitLogic: Training framework for gradient-based FPGA-native neural networks. *arXiv preprint arXiv:2602.07400*, 2026.

Katarzyna Fojcik, Renaldas Zioma, and Jogundas Armaitis. LILogic net: Compact logic gate networks with learnable connectivity for efficient hardware deployment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2026.

Lino Gerlach, Elliott Kauffman, and Abhishikth Mallampalli. Evaluation of novel fast machine learning algorithms for knowledge-distillation-based anomaly detection at cms. In *NeurIPS 2025 Workshop on Machine Learning and the Physical Sciences (ML4PS)*, 2025.

Lino Gerlach, Thore Gerlach, Liv Våge, Elliott Kauffman, and Isobel Ojalvo. WARP logic neural networks. *arXiv preprint arXiv:2602.03527*, 2026.

Youngsung Kim. Deep stochastic logic gate networks. *IEEE Access*, 11:122488–122501, 2023.

Youngsung Kim. Align forward, adapt backward: Closing the discretization gap in logic gate networks. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*. PMLR, 2026.

Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Computing Surveys (CSUR)*, 28(1), 1996.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. doi: 10.52202/068431-0146.

Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. doi: 10.52202/079017-3851.

Lukas Rüttgers, Till Aczel, Andreas Plesner, and Roger Wattenhofer. Light differentiable logic gate networks. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, volume 2026, 2026. URL `https://openreview.net/forum?id=EaGQ5luZtf`.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

Shengpu Wang, Yuhao Mao, Yani Zhang, and Martin Vechev. Learning compact boolean networks. *arXiv:2602.05830*, 2026.

Shakir Yousefi, Andreas Plesner, Till Aczel, and Roger Wattenhofer. Mind the gap: Removing the discretization gap in differentiable logic gate networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2026. doi: 10.52202/085713-4085.

A Additional Theory and Proofs

A.1 Proof of Theorem 1

Proof. Fix an integer $m \geq 1$ and consider any $N > 2m$. Construct a DLGN with N trainable neurons whose candidate gates include the constant gates $g_0 \equiv 0$ and $g_1 \equiv 1$. Let its output be

$$f_{\mathbf{g}}(X) = z_0 + \frac{z^* - z_0}{m} \sum_{i \leq N} g_i.$$

For every trainable neuron i, define the relaxed gate distribution

$$q_i^*(g_1) = \frac{m}{N}, \qquad q_i^*(g_0) = 1 - \frac{m}{N},$$

with zero probability assigned to all remaining gates. Under the relaxed forward pass, each neuron has output m/N. Consequently,

$$F_{q^*}(X) = z_0 + \frac{z^* - z_0}{m} \sum_{i \leq N} \frac{m}{N} = z^*.$$

By assumption, z^* globally minimizes the population risk over the prediction space attainable by the relaxation. Hence q^* is a global minimizer of the relaxed population risk.

Since $N > 2m$, we have $m/N < 1/2$. Thus g_0 is the unique modal gate of every q_i^*, and componentwise argmax yields $\mathbf{g}^{\arg} = (g_0, \ldots, g_0)$, and $f_{\mathbf{g}^{\arg}}(X) = z_0$. On the other hand, any discrete configuration containing exactly m copies of g_1 and $N - m$ copies of g_0 satisfies $f_{\mathbf{g}}(X) = z^*$. It therefore attains the globally minimal population risk and is, in particular, optimal among the discrete configurations. Hence, with $\mathcal{R}(z) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(z, Y)]$, $\mathcal{R}^*_{\text{disc}} = \mathcal{R}(z^*)$ whereas $\mathcal{R}_{\text{disc}}(\mathbf{g}^{\arg}) = \mathcal{R}(z_0)$. By Eq. (1),

$$\Delta_{\text{opt}} = \mathcal{R}_{\text{disc}}(\mathbf{g}^{\arg}) - \mathcal{R}^*_{\text{disc}} = \mathcal{R}(z_0) - \mathcal{R}(z^*) = c > 0,$$

independently of N. It remains to show local concentration. For every neuron,

$$q_i^* = \left(1 - \frac{m}{N}\right) \delta_{g_0} + \frac{m}{N} \delta_{g_1} \longrightarrow \delta_{g_0} \qquad \text{as } N \to \infty.$$

Let D be any local dispersion measure as defined earlier. Continuity of D at δ_{g_0} therefore implies $D(q_i^*) \longrightarrow D(\delta_{g_0}) = 0$. Since all N trainable neurons have the same distribution,

$$\max_{1 \leq i \leq N} D(q_i^*) \longrightarrow 0.$$

Thus the local gate distributions become arbitrarily concentrated while the argmax optimality gap remains equal to the positive constant c. □

A.2 Examples for the Local-Concentration Counterexample

We instantiate Theorem 1 for two standard supervised-learning losses. In both cases, the population risk admits a finite global minimizer z^* whose risk is strictly smaller than that of an alternative prediction z_0. The construction in Theorem 1 then yields globally optimal relaxed DLGNs whose local gate distributions converge uniformly to point masses, while componentwise argmax retains a strictly positive discrete optimality gap.

Squared loss. Consider a scalar prediction $z \in \mathbb{R}$ and squared loss $\ell(z, y) = (z - y)^2$. Let Y be any random variable with finite second moment and mean $\mathbb{E}_{(X,Y)\sim\mathcal{D}}[Y] = z^*$. The corresponding population risk is

$$\mathcal{R}(z) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[(z - Y)^2\right] = (z - z^*)^2 + \text{Var}_{(X,Y)\sim\mathcal{D}}(Y),$$

and is therefore globally minimized at $z = z^*$. For any $z_0 \neq z^*$,

$$\mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(z_0, Y) - \ell(z^*, Y)] = (z_0 - z^*)^2 =: c_{\text{MSE}} > 0.$$

Hence the assumptions of Theorem 1 hold. For example, choosing $Y \sim \text{Bernoulli}(0.64)$, $z^* = 0.64$ and $z_0 = 0$ gives $c_{\text{MSE}} = (0.64)^2 = 0.4096$. Thus the theorem constructs a sequence of globally optimal relaxed DLGNs whose gate distributions become arbitrarily concentrated, while the argmax discretization remains separated from the optimal discrete risk by 0.4096.

Binary cross-entropy. Consider binary classification with label $Y \sim \text{Bernoulli}(\pi)$, $\pi \in (0,1)$ and let $z \in \mathbb{R}$ denote the class-1 logit relative to a fixed class-0 logit of zero. The predicted class-1 probability is $\sigma(z)$, where σ denotes the logistic sigmoid. The binary cross-entropy loss can be written as $\ell(z,y) = \log(1+e^z) - yz$. Its population risk is therefore

$$\mathcal{R}(z) = \log(1+e^z) - \mathbb{E}_{(X,Y)\sim\mathcal{D}}[Y]\, z = \log(1+e^z) - \pi z.$$

Differentiating gives

$$\mathcal{R}'(z) = \sigma(z) - \pi, \qquad \mathcal{R}''(z) = \sigma(z)(1-\sigma(z)) > 0.$$

Consequently, the unique global minimizer is $z^* = \sigma^{-1}(\pi)$. Taking $z_0 = 0$ and any $\pi \neq 1/2$ yields

$$c_{\text{CE}} := \mathcal{R}(0) - \mathcal{R}(z^*) = \log 2 - \log(1+e^{z^*}) + \pi z^* > 0.$$

Thus binary cross-entropy also satisfies the assumptions of Theorem 1. Equivalently, choosing any fixed $m > 0$ and setting $\pi = \sigma(m)$ gives $z^* = m$ and

$$c_{\text{CE}} = \log 2 - \log(1+e^m) + m\sigma(m) > 0.$$

Applying the construction of Theorem 1 with $z_0 = 0$ and $z^* = m$ therefore yields a sequence for which the relaxed network remains globally optimal and every local gate distribution converges to a point mass, while the argmax optimality gap remains equal to the positive constant c_{CE}.

A.3 Proof of Eq. (4)

Proof. Before any gate is fixed, the conditional expected risk equals $\mathbb{E}_{\mathbf{G}\sim q}[\mathcal{R}_{\text{disc}}(\mathbf{G})]$. Suppose that $\mathbf{g}_{<i}$ has been fixed. By the law of total expectation,

$$\mathbb{E}_{G_i\sim q_i}\left[C_i^{\text{CE}}(G_i)\right] = \mathbb{E}_{\mathbf{G}_{\geq i}\sim q_{\geq i}}[\mathcal{R}_{\text{disc}}(\mathbf{g}_{<i}, \mathbf{G}_{\geq i})]\,.$$

Since the minimum of a finite set cannot exceed its weighted average, g_i^{CE} satisfies

$$\min_{g\in\mathcal{G}_i} C_i^{\text{CE}}(g) = C_i^{\text{CE}}(g_i^{\text{CE}}) \leq \mathbb{E}_{G_i\sim q_i}\left[C_i^{\text{CE}}(G_i)\right].$$

Thus fixing each gate according to $\arg\min_q C_i^{\text{CE}}(g)$ cannot increase the conditional expected risk. Repeating this argument for $i = 1, \ldots, N$ gives

$$\mathcal{R}_{\text{disc}}(\mathbf{g}^{\text{CE}}) \leq \mathbb{E}_{\mathbf{G}\sim q}[\mathcal{R}_{\text{disc}}(\mathbf{G})]\,.$$

The lower bound follows immediately from the definition of $\mathcal{R}^*_{\text{disc}}$. □

A.4 Proof of Theorem 2

Proof. Introduce $C_i^{\mu}(k \mid g_{<i}) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(\mu_{i,k}(X), Y)]$ as the risk of the mean completion prediction. By the triangle inequality,

$$\left|C_i^{\text{CE}}(k) - C_i^{\text{rel}}(k)\right| \leq \left|C_i^{\text{CE}}(k) - C_i^{\mu}(k)\right| + \left|C_i^{\mu}(k) - C_i^{\text{rel}}(k)\right|.$$

For the first term, Lipschitz continuity gives

$$\left|C_i^{\text{CE}}(k) - C_i^{\mu}(k)\right| \leq L\mathbb{E}_{(X,Y)\,\sim\mathcal{D},\mathbf{G}_{>i}\sim q_{>i}}[\|f_{g_{<i},k,\mathbf{G}_{>i}}(X) - \mu_{i,k}(X)\|] \leq LV_{i,k},$$

where the final inequality follows from Cauchy–Schwarz. Similarly,

$$\left|C_i^{\mu}(k) - C_i^{\text{rel}}(k)\right| \leq LP_{i,k}.$$

Combining both bounds proves Eq. (5). □

A.5 Proof of Theorem 3

Proof. For fixed i, k, and sample (x,y), let $\phi_i(a;x,y) := \ell(F_{i,a}(x), y)$, where $F_{i,a}$ denotes the current hybrid network with the scalar activation of neuron i replaced by a, leaving the downstream computation unchanged. Thus, the relaxed and intervened networks correspond to $a_i(x)$ and $a_{i,k}(x)$, respectively. Taylor's theorem gives

$$\phi_i(a_{i,k};x,y) = \phi_i(a_i;x,y) + \phi_i'(a_i;x,y)\Delta a_{i,k}(x) + r_{i,k}(x,y).$$

Table 3: **Compact CIFAR-10 architectures.** Compact-S uses $k = 128$ and $N = 3$ input thresholds; Compact-M uses $k = 256$ and $N = 7$. Convolutional layers use 3×3 receptive fields and padding 1.

Stage	Operation	Stride	Compact-S ($k = 128$)	Compact-M ($k = 256$)
Input	Thermometer encoding	–	$(9, 32, 32)$	$(21, 32, 32)$
Conv 1	3×3 logic conv.	2	$(128, 16, 16)$	$(256, 16, 16)$
Conv 2	3×3 logic conv.	1	$(128, 16, 16)$	$(256, 16, 16)$
Conv 3	3×3 logic conv.	2	$(512, 8, 8)$	$(1024, 8, 8)$
Conv 4	3×3 logic conv.	1	$(512, 8, 8)$	$(1024, 8, 8)$
Flatten	–	–	32,768	65,536
Logic 1	Logic layer	–	80,000	160,000
Logic 2	Logic layer	–	80,000	160,000
Output	GroupSum	–	10	10

By the curvature assumption,

$$|r_{i,k}(x, y)| \leq \frac{M_i}{2} \Delta a_{i,k}(x)^2.$$

Moreover, $\phi_i'(a_i; X, Y) = \frac{\partial \ell}{\partial a_i}$ and taking expectations therefore yields

$$\left| C_i^{\mathrm{rel}}(k) - \left[C_i^{\mathrm{rel}}(\mathrm{soft}) + S_i(k) \right] \right| \leq \frac{M_i}{2} \mathbb{E}_{(X,Y)\sim\mathcal{D}} \left[\Delta a_{i,k}(X)^2 \right]. \tag{8}$$

Combining Eq. (8) with Theorem 2 and the triangle inequality gives

$$\begin{aligned} \left| C_i^{\mathrm{CE}}(k) - \left[C_i^{\mathrm{rel}}(\mathrm{soft}) + S_i(k) \right] \right| &\leq \left| C_i^{\mathrm{CE}}(k) - C_i^{\mathrm{rel}}(k) \right| + \left| C_i^{\mathrm{rel}}(k) - \left[C_i^{\mathrm{rel}}(\mathrm{soft}) + S_i(k) \right] \right| \\ &\leq L_\ell(V_{i,k} + P_{i,k}) + \frac{M_i}{2} \mathbb{E}_{(X,Y)\sim\mathcal{D}} \left[\Delta a_{i,k}(X)^2 \right] = \epsilon_{i,k}, \end{aligned}$$

which proves Eq. (6). Now let $k_i^{\mathrm{CE}} \in \arg\min_{k\in\mathcal{G}} C_i^{\mathrm{CE}}(k)$. Since k_i^{FO} minimizes $S_i(k)$, $S_i(k_i^{\mathrm{FO}}) \leq S_i(k_i^{\mathrm{CE}})$. Applying Eq. (6) to both candidates gives

$$C_i^{\mathrm{CE}}(k_i^{\mathrm{FO}}) \leq C_i^{\mathrm{rel}}(\mathrm{soft}) + S_i(k_i^{\mathrm{FO}}) + \epsilon_i \leq C_i^{\mathrm{rel}}(\mathrm{soft}) + S_i(k_i^{\mathrm{CE}}) + \epsilon_i \leq C_i^{\mathrm{CE}}(k_i^{\mathrm{CE}}) + 2\epsilon_i.$$

Since $C_i^{\mathrm{CE}}(k_i^{\mathrm{CE}}) = \min_{k\in\mathcal{G}} C_i^{\mathrm{CE}}(k)$, this proves Eq. (6). For proving Eq. (7), let $k_i^{\mathrm{arg}} \in \arg\max_{k\in\mathcal{G}} q_i(k)$. Applying Eq. (6) to k_i^{FO} and k_i^{arg} gives

$$C_i^{\mathrm{CE}}(k_i^{\mathrm{FO}}) - C_i^{\mathrm{CE}}(k_i^{\mathrm{arg}}) \leq S_i(k_i^{\mathrm{FO}}) - S_i(k_i^{\mathrm{arg}}) + 2\epsilon_i.$$

Hence

$$S_i(k_i^{\mathrm{arg}}) - S_i(k_i^{\mathrm{FO}}) > 2\epsilon_i \quad \Longrightarrow \quad C_i^{\mathrm{CE}}(k_i^{\mathrm{FO}}) < C_i^{\mathrm{CE}}(k_i^{\mathrm{arg}}).$$

□

B Additional Experimental Details

B.1 Model Architectures

We consider two convolutional DLGN families on CIFAR-10. Our *Compact-S/M* models follow the convolutional architecture of Wang et al. (2026), while *TreeLogicNet-S/M* follow the TreeLogicNet architecture of Petersen et al. (2024). The two families differ substantially in their convolutional parameterization: Compact models use a single Boolean operation per convolutional neuron together with connection learning, whereas TreeLogicNet uses depth-3 Boolean trees containing seven trainable gates per convolutional kernel.

Compact models. For CIFAR-10, the Compact architecture consists of four convolutional layers followed by two fully connected logic layers and a GroupSum classification head. The width parameter is $k = 128$ for Compact-S and $k = 256$ for Compact-M. Following Wang et al. (2026), the input is thermometer encoded using $N = 3$ thresholds for S and $N = 7$ thresholds for M. Each convolutional kernel has a 3×3 receptive field. The exact architectures used in our experiments are summarized in Tab. 3.

During training, Compact neurons jointly represent gate and connection choices: the 16 candidate Boolean operations are associated with independently sampled input pairs from the local receptive field. Connection learning proceeds through adaptive resampling of these candidates. After discretization, only one Boolean operation and its corresponding input pair remain per neuron.

Table 4: **TreeLogicNet CIFAR-10 architectures.** Each convolutional kernel is a depth-3 Boolean tree with seven trainable gates, followed by fixed 2×2 OR pooling. S and M correspond to $k = 32$ and $k = 256$, respectively.

Stage	Operation	TreeLogicNet-S ($k = 32$)	TreeLogicNet-M ($k = 256$)
Input	Three-threshold encoding	$(9, 32, 32)$	$(9, 32, 32)$
Conv 1	3×3, tree depth 3	32 channels	256 channels
Pool 1	2×2 OR, stride 2	$(32, 16, 16)$	$(256, 16, 16)$
Conv 2	3×3, tree depth 3	128 channels	1024 channels
Pool 2	2×2 OR, stride 2	$(128, 8, 8)$	$(1024, 8, 8)$
Conv 3	3×3, tree depth 3	512 channels	4096 channels
Pool 3	2×2 OR, stride 2	$(512, 4, 4)$	$(4096, 4, 4)$
Conv 4	3×3, tree depth 3	1024 channels	8192 channels
Pool 4	2×2 OR, stride 2	$(1024, 2, 2)$	$(8192, 2, 2)$
Flatten	–	4096	32,768
Logic 1	Random logic layer	40,960	327,680
Logic 2	Random logic layer	20,480	163,840
Logic 3	Random logic layer	10,240	81,920
Output	GroupSum	10	10

TreeLogicNet models. The TreeLogicNet models follow the CIFAR-10 architecture of Petersen et al. (2024). Each convolutional block uses depth-3 binary logic trees: eight sampled inputs are combined by seven trainable Boolean gates into one output. Each block is followed by fixed 2×2 OR pooling with stride 2. Four such blocks reduce the spatial resolution from 32×32 to 2×2, after which three randomly connected logic layers and a GroupSum head perform classification. The width parameter is $k = 32$ for TreeLogicNet-S and $k = 256$ for TreeLogicNet-M. Both models use three thresholds for the CIFAR-10 input encoding.

The two architecture families therefore provide complementary testbeds. Compact-S/M use shallow single-gate convolutional kernels with learned connectivity, whereas TreeLogicNet-S/M retain deeper tree-structured convolution and fixed random connectivity. This allows us to test whether the observed discretization behavior persists across substantially different convolutional DLGN parameterizations.

B.2 Training and Freezing Hyperparameters

All models are trained on the 45k/5k training–validation split of CIFAR-10 described in Sec. 5, using cross-entropy loss and GroupSum aggregation. Compact models use Adam and are trained for 600k steps, while TreeLogicNet models use AdamW with weight decay 0.002 and are trained for 200k steps. Unless stated otherwise, all reported S-model experiments use five independent seeds.

For progressive discretization, freezing begins only after an initial relaxed training period. TaskFreeze evaluates the first-order argmax freeze shock every K training steps, maintains an EMA with factor β, and freezes a neuron once its smoothed score remains negative for P consecutive evaluations. EntropyFreeze uses the same neuron-wise progressive framework but determines eligibility from convergence of the local gate entropy. LayerEntropyFreeze follows the shallow-to-deep layerwise entropy criterion of Wang et al. (2026), while GlobalFreeze simultaneously fixes all neurons to their current argmax gates at the discretization start. FinalArgmax keeps all gates relaxed throughout training and applies argmax only for discrete evaluation.

The exact optimization and freezing hyperparameters used in our experiments are summarized in Tab. 5.

B.3 Diagnostic Protocol

Our diagnostic experiments measure the effect of discretizing individual neurons while keeping the remainder of the network relaxed. For neuron i and candidate gate k, we define the realized intervention effect as

$$\Delta C_i^{\mathrm{rel}}(k) := C_i^{\mathrm{rel}}(k) - C_i^{\mathrm{rel}}(\mathrm{soft}),$$

where only neuron i is replaced by the discrete gate k. Negative values therefore indicate that the intervention decreases the task loss.

Table 5: **Training and freezing hyperparameters.** H denotes entropy convergence and S first-order freeze shock. Values are shared across the Compact and TreeLogicNet experiments unless shown otherwise.

Parameter	Compact	TreeLogicNet
Optimizer	Adam	AdamW
Weight decay	0	0.002
Training steps	600k	200k
Batch size	128	128
Learning rate	0.02	0.02
LR schedule	Constant	Constant
Scoring interval K	$K_H = 1, K_S = 1000$	$K_H = 1, K_S = 1000$
EMA factor β	$\beta_H = 0.99, \beta_S = 2^{-K_S/10\,000}$	$\beta_H = 0.99, \beta_S = 2^{-K_S/10\,000}$
Patience P	$P_H = 200, P_S = 3$	$P_H = 200, P_S = 3$
Entropy criterion	$\lvert H_t - \bar{H}_{t-1}\rvert < 5 \times 10^{-4}$	$\lvert H_t - \bar{H}_{t-1}\rvert < 5 \times 10^{-4}$

For the entropy and argmax-score diagnostics, the intervention gate is $k_i^{\arg} = \arg\max_k q_i(k)$. For score-based selection, we instead use $k_i^* = \arg\min_k S_i(k)$. The corresponding first-order scores are computed from the same relaxed network state without additional downstream forward passes. Unless explicitly stated otherwise, diagnostic loss changes and scores are evaluated on the validation set.

To test approximation locality, we additionally measure the squared activation displacement

$$D_i(k) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\Delta a_{i,k}(X)^2], \qquad \Delta a_{i,k} := a_{i,k} - a_i.$$

For each neuron, the all-candidate analysis evaluates all 16 Boolean gates and relates $D_i(k)$ to the error of the first- and second-order loss predictions. The second-order score is

$$S_i^{(2)}(k) := S_i(k) + \frac{1}{2}\Delta a_{i,k}^\top H_i \Delta a_{i,k},$$

where the quadratic form is evaluated using Hessian–vector products without explicitly constructing the Hessian. We report Spearman correlations across candidates within each neuron and aggregate these statistics across neurons.

For the held-out selection experiment in Fig. 3c, the validation data are divided into two disjoint halves. The first half is used to compute first-order task-aware choices and activation displacements, while realized loss effects are evaluated exclusively on the second half. This separates gate selection from evaluation and avoids measuring the apparent advantage of a selected intervention on the same examples used to select it.

B.4 Diagnostics for TreeLogicNet

We repeat the main diagnostic analyses on TreeLogicNet-S to assess whether the observed behavior is specific to the Compact architecture. Fig. 5 shows the corresponding freeze-effect analysis at the 25k discretization start. As for Compact-S, gate entropy provides only a weak predictor of the realized argmax freeze effect, while the first-order argmax freeze shock is consistently positively associated with the measured loss change. Directly optimizing the first-order score over candidate gates is substantially less reliable and exhibits pronounced layer dependence. This confirms that the distinction between using first-order information for *selection* and for *assessment* is not specific to the Compact architecture.

We additionally compare the resulting progressive freezing dynamics in Fig. 6. The qualitative schedules differ from Compact-S: EntropyFreeze commits a large fraction of neurons soon after discretization begins and ultimately freezes more neurons than TaskFreeze. Nevertheless, TaskFreeze again produces a gradual neuron-wise transition rather than the abrupt commitment induced by GlobalFreeze or LayerFreeze. The layer-wise view further shows that the two neuron-wise criteria are not interchangeable: EntropyFreeze becomes increasingly aggressive toward deeper convolutional layers, whereas TaskFreeze distributes its decisions more evenly across depth. Thus, the precise freezing trajectory is architecture dependent, while task-based and entropy-based eligibility remain qualitatively distinct.

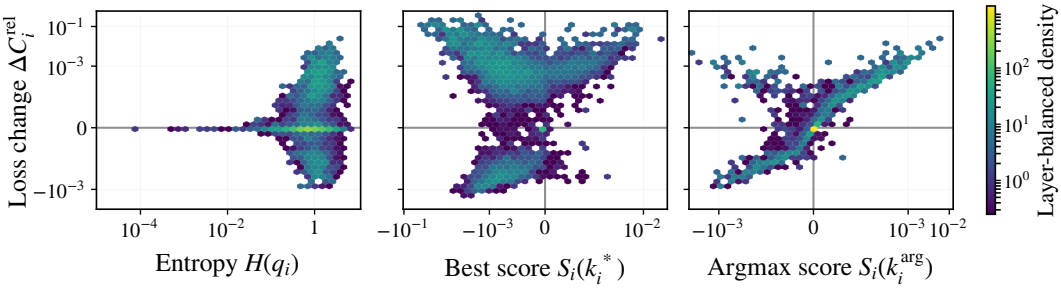
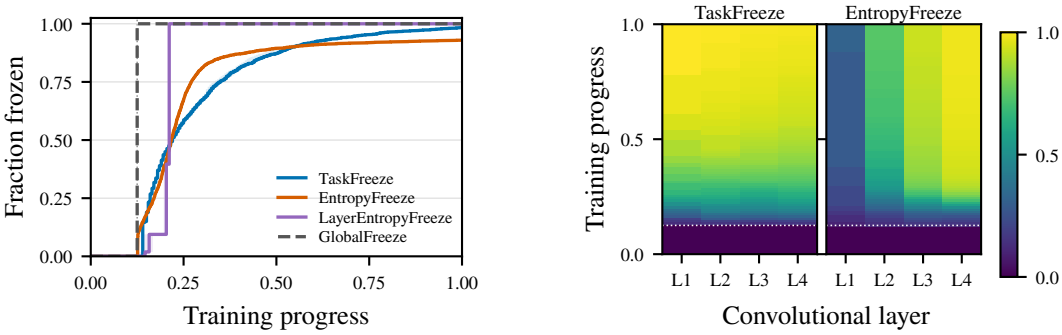


Figure 5: **Predicting freeze effects on TreeLogicNet-S.** Measured loss change from freezing to the argmax gate versus gate entropy (left), from freezing to the task-aware gate $k_i^* = \arg\min_k S_i(k)$ versus its first-order score (center), and from freezing to the argmax gate versus its first-order score (right). As for Compact-S, score-based selection is less reliably aligned with the realized effect, whereas the argmax freeze shock is strongly predictive.



(a) Global freezing dynamics.

(b) Layer-wise freezing dynamics.

Figure 6: **Progressive freezing dynamics on TreeLogicNet-S.** (a) Fraction of frozen neurons throughout training for different discretization criteria. (b) Layer-wise frozen fractions show that TaskFreeze progressively commits individual decisions, whereas EntropyFreeze induces a markedly different layer-dependent schedule.

B.5 Sensitivity to Discretization Start

We additionally evaluate the sensitivity of the freezing methods to the point at which discretization begins. Tables 6 and 7 report final discrete CIFAR-10 test accuracy across multiple start points for Compact-S and TreeLogicNet-S, respectively. Overall, the relative performance of the methods is reasonably stable across the considered schedules. Across both architectures, the main conclusions

Table 6: **Sensitivity to discretization start on Compact-S.** Final discrete CIFAR-10 test accuracy (%) for discretization starting at 200k, 300k, or 400k training steps, with connection resampling either continued or stopped after discretization begins. Values are mean ± sample standard deviation across available seeds.

	Continued resampling			**Stopped resampling**		
Method	**200k**	**300k**	**400k**	**200k**	**300k**	**400k**
TaskFreeze (ours)	66.13 ± 0.20	66.52 ± 0.17	66.25 ± 0.27	65.55 ± 0.29	65.77 ± 0.15	65.93 ± 0.15
EntropyFreeze (ours)	64.30 ± 0.83	64.32 ± 0.72	64.11 ± 0.53	64.28 ± 0.48	64.00 ± 0.58	63.88 ± 0.76
GlobalFreeze (ours)	66.27 ± 0.51	66.17 ± 0.10	66.41 ± 0.21	65.84 ± 0.15	66.04 ± 0.26	66.12 ± 0.37
LayerEntropyFreeze	64.75 ± 0.80	65.24 ± 0.53	64.10 ± 0.45	65.94 ± 0.14	65.95 ± 0.26	65.99 ± 0.17
FinalArgmax	64.63 ± 0.61	64.63 ± 0.61	64.63 ± 0.61	64.63 ± 0.61	64.63 ± 0.61	64.63 ± 0.61
CAGE	63.54 ± 0.65	63.42 ± 0.88	63.32 ± 0.63	63.62 ± 0.39	64.15 ± 0.18	64.11 ± 0.39

are robust to the discretization start. TaskFreeze remains competitive across all tested checkpoints, while no single freezing schedule consistently dominates. For Compact-S, this behavior also persists when connection resampling is stopped after discretization begins. Together with the layer-,

Table 7: **Sensitivity to discretization start on TreeLogicNet-S.** Final discrete CIFAR-10 test accuracy (%) for discretization starting at 10k, 25k, or 50k training steps. Values are mean ± sample standard deviation across five seeds.

Method	10k	25k	50k
TaskFreeze (ours)	60.42 ± 0.63	60.76 ± 0.46	60.26 ± 0.80
EntropyFreeze (ours)	59.89 ± 0.45	59.97 ± 0.84	60.13 ± 0.57
GlobalFreeze (ours)	60.05 ± 0.82	60.50 ± 0.52	60.78 ± 0.28
LayerEntropyFreeze	60.72 ± 0.57	60.63 ± 0.55	60.81 ± 0.42
FinalArgmax	59.29 ± 0.78	59.29 ± 0.78	59.29 ± 0.78
CAGE	60.13 ± 0.49	59.98 ± 0.35	59.90 ± 0.71

checkpoint-, and seed-wise diagnostics in the main text, these results indicate that the observed distinction between task-based assessment and entropy-based freezing is not tied to a particular training checkpoint or connection-learning schedule.