STABILIZING LOGICAL COMPOSITION IN DIFFERENTIABLE LOGIC GATE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentiable logic gate networks (DLGNs) can be discretized into efficient hard logic circuits after training, but their intermediate relaxed Boolean features can be sensitive to input perturbations. This property is particularly relevant in DLGNs, where intermediate features serve as operands for subsequent logic gates and therefore directly participate in downstream logical composition. In this paper, we study convolutional DLGNs from the perspective of Boolean feature stability. We show that the bilinear interactions in relaxed logic gates induce feature-dependent local Jacobians, such that the local sensitivity of a gate varies with the values of its Boolean operands. Motivated by this analysis, we propose feature-level Jacobian regularization to stabilize selected intermediate relaxed Boolean features. We further introduce Binarization-Aware Jacobian regularization (BAJ), which assigns larger penalties to features near the binarization boundary, where perturbations are more likely to alter their Boolean interpretation. Experiments across image classification benchmarks show that the proposed regularizers improve classification performance and reduce the finite-perturbation sensitivity of intermediate Boolean features. Downstream analyses further show that feature stabilization affects subsequent logical computation. These results suggest that stabilizing intermediate Boolean operands is an effective approach to improving the stability of logical composition in convolutional DLGNs.

1 INTRODUCTION

Deep learning has achieved remarkable success across a wide range of machine learning tasks, but its increasing computational and energy demands remain a major obstacle to deployment under strict latency, memory, and power constraints (Howard et al., 2017; Tan & Le, 2019). Many neural network compression methods address this challenge through techniques such as quantization and pruning (Hubara et al., 2016; Han et al., 2016). While these approaches significantly reduce computational cost, they still fundamentally rely on high-level arithmetic operations such as matrix multiplication.

Differentiable logic gate networks (DLGNs) take a fundamentally different approach (Petersen et al., 2022; 2024). Rather than optimizing conventional neural computation and then compressing it, DLGNs directly learn Boolean logic circuits from data. Consequently, computation is optimized at one of the lowest abstraction levels of digital hardware. Once trained and discretized, DLGNs perform inference using sparse bitwise logic operations, making them an attractive model family for hardware-efficient inference. This design also introduces optimization characteristics that differ substantially from those of conventional neural networks.

For image domain tasks, this framework has been extended to Convolutional Differentiable Logic Gate Networks (CLGNs), which apply differentiable logic gates over local receptive fields (Petersen et al., 2024). Alongside this extension, prior studies have investigated various DLGN-specific issues, including discretization (Yousefi et al., 2025), connectivity (Mommen et al., 2025), parametrization (Rüttgers et al., 2026), and extensions to broader vision tasks (Lee et al., 2026). However, the feature-level properties of intermediate relaxed Boolean representations in CLGNs remain relatively unexplored, despite their direct role as operands in downstream logical computation. Understanding these properties is important for characterizing how variations in intermediate features affect subsequent logical composition.

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.

In this work, we study CLGNs for classification through the stability of intermediate Boolean features. In CLGNs, activations produced by convolutional logic layers serve as relaxed Boolean operands for subsequent logic gates. Therefore, perturbations in these features can affect not only internal representations, but also downstream logical composition.

This issue is particularly important because relaxed logic gates contain bilinear interaction terms. As a result, the local Jacobian of a gate depends on the current feature values, meaning that perturbations in intermediate Boolean features can affect the local sensitivity of subsequent logic layers. This motivates stabilizing intermediate relaxed Boolean features before such perturbations affect downstream logical composition.

To this end, we introduce Feature-Jacobian regularization (FJ) for CLGNs, which penalizes the input sensitivity of selected intermediate relaxed Boolean features. By reducing the feature Jacobian, FJ suppresses local variations in these features under input perturbations, encouraging more stable operands for subsequent logic gates. We further propose Binarization-Aware Jacobian regularization (BAJ), which assigns larger sensitivity penalties to features near the binarization boundary, where small perturbations are more likely to change the corresponding bit interpretation.

Our contributions are summarized as follows.

- We analyze CLGNs through intermediate Boolean feature stability and show that relaxed logic gates induce feature-dependent local Jacobians.
- We propose FJ and BAJ for stabilizing intermediate relaxed Boolean operands.
- We empirically show that the proposed regularizers improve classification accuracy and reduce finite-perturbation sensitivity at intermediate and downstream logic stages.

2 Related Work

2.1 Differentiable Logic Gate Network

DLGNs learn Boolean logic circuits through gradient-based optimization (Petersen et al., 2022). Each node represents a two-input Boolean gate, and discrete gate types are continuously relaxed during training to enable gradient-based learning. After training, the learned gates are discretized into hard Boolean functions, allowing inference to be executed using sparse bitwise logic operations. This makes DLGNs different from conventional neural networks and binary neural networks, which still rely on dense linear-algebraic computation.

CLGNs extend this framework by applying differentiable logic gates over local receptive fields and using logical pooling for spatial downsampling (Petersen et al., 2024). Recent work has addressed several limitations of DLGNs, including discretization (Yousefi et al., 2025), learnable connectivity (Mommen et al., 2025; Fojcik et al., 2026), gate parametrization and scalability (Rüttgers et al., 2026; Park et al., 2026), and extensions to broader vision tasks such as image restoration (Lee et al., 2026). These studies demonstrate continued progress in the trainability, scalability, and application scope of logic-gate networks.

However, the feature-level behavior of intermediate relaxed Boolean representations remains relatively less explored. In this work, we study this behavior from the perspective of feature stability in CLGNs.

2.2 Jacobian Regularization and Local Sensitivity

Jacobian-based regularization has been widely studied as a way to control the local sensitivity of neural networks. Prior work has related the Jacobian norm around training samples to classification margin, robustness, and generalization (Sokolić et al., 2017; Jakubovitz & Giryes, 2018; Hoffman et al., 2019). By penalizing the input-output Jacobian, these methods encourage model predictions to change smoothly under small input perturbations. This idea has been used to improve robustness against random or adversarial perturbations, often by regularizing the Frobenius norm of the network Jacobian. Related approaches have also considered local linearity and curvature to control nonlinear behavior around training samples (Qin et al., 2019).

The use of Jacobian regularization has also extended beyond conventional input-output robustness. It has been used to stabilize deep equilibrium models (Bai et al., 2021), and more recently to regularize intermediate representations in pretrained language models for improved smoothness and generalization (Jukić & Šnajder, 2025).

Prior work has mainly used Jacobian regularization to control input-output sensitivity or improve the smoothness of learned representations. Our approach is instead motivated by the feature-level characteristics of CLGNs, regularizing selected intermediate relaxed Boolean features and further incorporating binarization-aware weighting.

3 BACKGROUND

DLGNs are neural networks composed of two-input Boolean logic gates. Since both Boolean activations and discrete gate selections are non-differentiable, DLGNs are trained through a continuous relaxation. During training, binary activations are relaxed to values in $[0, 1]$, and each node represents a soft distribution over the 16 possible two-input Boolean functions. For relaxed inputs $a, b \in [0, 1]$, the surrogate of the i-th Boolean gate can be written as

$$g_i(a, b) = \alpha_{i,0} + \alpha_{i,1}a + \alpha_{i,2}b + \alpha_{i,3}ab, \tag{1}$$

where the coefficients are determined by the corresponding Boolean function.

To make the gate choice learnable, DLGNs use a softmax distribution over the 16 possible Boolean gates. For a learnable gate-logit vector $z \in \mathbb{R}^{16}$, the probability of selecting the i-th Boolean function is

$$\pi_i(z) = \frac{\exp(z_i)}{\sum_{k=1}^{16} \exp(z_k)}. \tag{2}$$

The relaxed logic gate output is computed as

$$f_z(a, b) = \sum_{i=1}^{16} \pi_i(z) g_i(a, b). \tag{3}$$

Because each $g_i(a, b)$ is bilinear, their weighted sum can also be written as

$$f_z(a, b) = w_0 + w_1 a + w_2 b + w_3 ab, \tag{4}$$

where w_0, w_1, w_2, w_3 are determined by the soft gate distribution. After training, each relaxed gate is discretized by selecting the Boolean function with the highest learned probability.

We denote the relaxed Boolean activations at layer l for input x by $h^{(l)}(x) \in [0, 1]^{d_l}$, where d_l is the number of relaxed Boolean features at that layer. When the input is clear from context, we omit the explicit dependence on x and write $h^{(l)}$.

For the classification task, DLGNs assign multiple output neurons to each class and aggregate them using a scaled group-sum classifier. We denote the output logits by $s(x) \in \mathbb{R}^C$, whose c-th component is given by

$$s_c(x) = \frac{1}{\tau} \sum_{j \in \mathcal{G}_c} h_j^{(L)}(x) = \left[G h^{(L)}(x) \right]_c, \tag{5}$$

where $\mathcal{G}_c$ is the set of output neurons assigned to class c, τ is the group-sum scaling factor, and $G \in \mathbb{R}^{C \times d_L}$ is the scaled group-sum matrix with entries $1/\tau$ for neurons assigned to the corresponding class and zero otherwise.

In the CLGNs considered in this work, convolutional logic-tree layers are interleaved with 2×2 max-pooling operations, followed by additional logic layers and a scaled group-sum classifier.

4 FEATURE INSTABILITY IN LOGICAL COMPOSITION

We analyze why intermediate feature stability is important in CLGNs. In these networks, intermediate relaxed Boolean features are not merely hidden representations; they serve as operands for

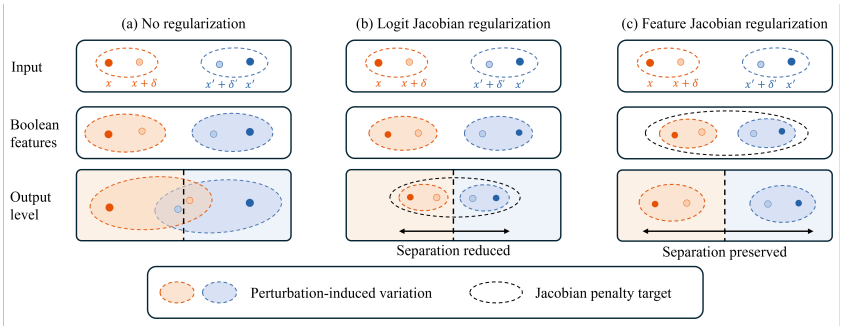


Figure 1: Conceptual comparison between an output-level Jacobian baseline and the proposed feature-level regularization for CLGNs. The output-level baseline applies sensitivity regularization to the final input-output mapping, whereas our method targets intermediate relaxed Boolean features before they are used in downstream logical composition. Colored dashed ellipses denote perturbation-induced variation, and black dashed ellipses denote the Jacobian penalty target.

subsequent logic gates. Therefore, perturbations in these features can directly affect downstream logical composition.

Consider a differentiable logic layer l. For the i-th output feature, let $p_i^{(l)}$ and $q_i^{(l)}$ denote the two distinct selected input indices. The relaxed gate output can be written in bilinear form as

$$h_i^{(l+1)} = w_{i,0}^{(l)} + w_{i,1}^{(l)} h_{p_i^{(l)}}^{(l)} + w_{i,2}^{(l)} h_{q_i^{(l)}}^{(l)} + w_{i,3}^{(l)} h_{p_i^{(l)}}^{(l)} h_{q_i^{(l)}}^{(l)}. \tag{6}$$

The corresponding entries of the layer Jacobian are

$$\begin{aligned} [J_l]_{i,p_i^{(l)}} &= \frac{\partial h_i^{(l+1)}}{\partial h_{p_i^{(l)}}^{(l)}} = w_{i,1}^{(l)} + w_{i,3}^{(l)} h_{q_i^{(l)}}^{(l)}, \\ [J_l]_{i,q_i^{(l)}} &= \frac{\partial h_i^{(l+1)}}{\partial h_{q_i^{(l)}}^{(l)}} = w_{i,2}^{(l)} + w_{i,3}^{(l)} h_{p_i^{(l)}}^{(l)}. \end{aligned} \tag{7}$$

Thus, the local sensitivity of a relaxed logic gate is feature-dependent: the derivative with respect to one operand depends on the value of the other operand. This property is different from conventional linear layers, whose local Jacobians do not depend on the current activation values.

For an intermediate layer k, the local propagation from $h^{(k)}$ to the output logits s is described by the downstream Jacobian:

$$J_{k \to L}(x) = \frac{\partial s}{\partial h^{(k)}} = G J_{L-1}(h^{(L-1)}) \cdots J_k(h^{(k)}), \tag{8}$$

where G is the scaled group-sum matrix defined in Section 3. Since each layer Jacobian J_l depends on the current relaxed Boolean features, perturbations in intermediate features can change not only downstream activations but also the local Jacobians of subsequent logic gates.

Consequently, instability in intermediate relaxed Boolean activations can make downstream logical computation unstable in a local sense. Small input changes may induce large variations in these activations, substantially changing the operands supplied to subsequent logic gates and altering the local behavior of downstream logical composition. This motivates stabilizing intermediate relaxed Boolean features before they are used as operands in subsequent logic layers. A first-order derivation of how feature perturbations affect downstream Jacobian products is provided in Appendix A.2.

5 STABILIZATION FOR LOGICAL COMPOSITION

5.1 FEATURE-JACOBIAN REGULARIZATION

Motivated by the analysis in Section 4, we directly stabilize selected intermediate relaxed Boolean features by regularizing their sensitivity to the input. Figure 1 provides a conceptual comparison between this feature-level strategy and output-level Jacobian regularization.

Let k^* denote the selected regularization layer, and let $h^{(k^*)}(x) \in [0,1]^{d_{k^*}}$ be the relaxed Boolean feature at this layer. We measure its local sensitivity to the input using the feature Jacobian as

$$D_{k^*}(x) = \frac{\partial h^{(k^*)}(x)}{\partial x} \in \mathbb{R}^{d_{k^*} \times n}, \tag{9}$$

where n is the input dimensionality. Feature-Jacobian regularization (FJ) penalizes the Frobenius norm of this Jacobian:

$$\mathcal{R}_{\mathrm{FJ}}^{(k^*)} = \frac{1}{2}\mathbb{E}_x \left[\|D_{k^*}(x)\|_F^2 \right]. \tag{10}$$

Rather than treating stability only as a property of the final input-output mapping, this objective targets a structural source of instability in CLGNs: the sensitivity of intermediate relaxed Boolean operands before they are reused in downstream logical composition.

5.2 BINARIZATION-AWARE JACOBIAN REGULARIZATION

The FJ regularizer treats all feature dimensions uniformly. However, relaxed Boolean features have an additional property: after discretization, each feature is interpreted as a binary value. Therefore, the impact of a feature perturbation depends on the feature's proximity to the binarization boundary.

For a relaxed feature $h_j^{(k^*)}(x)$, perturbations when the feature is near 0.5 are more likely to change the corresponding binary interpretation than those when the feature is near 0 or 1. To reflect this property, we introduce BAJ, a binarization-aware weighted variant of FJ that assigns larger sensitivity penalties to features near the binarization boundary. We define the binarization score as

$$\omega_j^{(k^*)}(x) = 4h_j^{(k^*)}(x)\left(1 - h_j^{(k^*)}(x)\right) \tag{11}$$

This score is maximized at $h_j^{(k^*)}(x) = 0.5$ and decreases toward zero as the feature approaches either binary endpoint.

The BAJ regularizer is defined as

$$\mathcal{R}_{\mathrm{BAJ}}^{(k^*)} = \frac{1}{2}\mathbb{E}_x \left[\sum_{j=1}^{d_{k^*}} \mathrm{sg}\left[\omega_j^{(k^*)}(x)\right] \left\| \frac{\partial h_j^{(k^*)}(x)}{\partial x} \right\|_2^2 \right]. \tag{12}$$

Here, $\mathrm{sg}[\cdot]$ denotes stop-gradient. Thus, the binarization score only determines how strongly the sensitivity of each feature is penalized and does not itself push the feature toward a binary endpoint. Setting all $\omega_j^{(k^*)}(x)$ to one recovers FJ.

BAJ does not directly encourage relaxed features to become binary. Instead, it selectively penalizes feature sensitivities according to their proximity to the binarization boundary. Features close to the boundary receive larger sensitivity penalties because perturbations to these features are more likely to alter the corresponding Boolean interpretation. This makes the regularizer better aligned with the discrete logical computation that the relaxed network is intended to approximate.

5.3 TRAINING OBJECTIVE

Computing the feature Jacobian norm exactly is expensive for high-dimensional intermediate features. We use a single-sample random-projection estimator for FJ following prior work (Hoffman et al., 2019), and extend the same projection identity to the binarization-weighted Jacobian used by BAJ. Details are provided in Appendix A.1.

Table 1: Top-1 accuracy (%) across classification benchmarks.

Method	MNIST	SVHN	CIFAR-10	CIFAR-100	Tiny-ImageNet
Baseline	98.30	78.89	71.08	42.22	24.30
OJ	98.35	79.21	71.58	42.23	24.80
FJ	98.39	**79.54**	71.70	42.95	26.24
BAJ	**98.42**	79.39	**72.24**	**42.99**	**26.28**

Table 2: Perturbed classification accuracy (%) under raw-input perturbations. Results are reported for the threshold-encoded image benchmarks.

Method	SVHN	CIFAR-10	CIFAR-100	Tiny-ImageNet
Baseline	65.85	66.82	37.00	20.42
OJ	**66.16**	66.72	36.87	20.84
FJ	64.18	67.31	37.87	22.74
BAJ	65.86	**67.61**	**38.54**	**23.22**

BAJ does not directly encourage relaxed features to become binary. Instead, the detached weights selectively reweight feature sensitivities according to their proximity to the binarization boundary.

The final training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \widehat{\mathcal{R}}^{(k^*)}, \tag{13}$$

where $\mathcal{L}_{\mathrm{CE}}$ is the cross-entropy loss computed from the output logits $s(x)$, $\widehat{\mathcal{R}}^{(k^*)}$ is the stochastic estimate of either the FJ or BAJ regularizer, and λ controls the strength of feature-level regularization.

6 Experiments

6.1 Experimental Setup

We evaluate our method on standard image classification benchmarks, including MNIST (Le Cun et al., 1998), SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and Tiny-ImageNet (Deng et al., 2009). We report top-1 classification accuracy for all datasets.

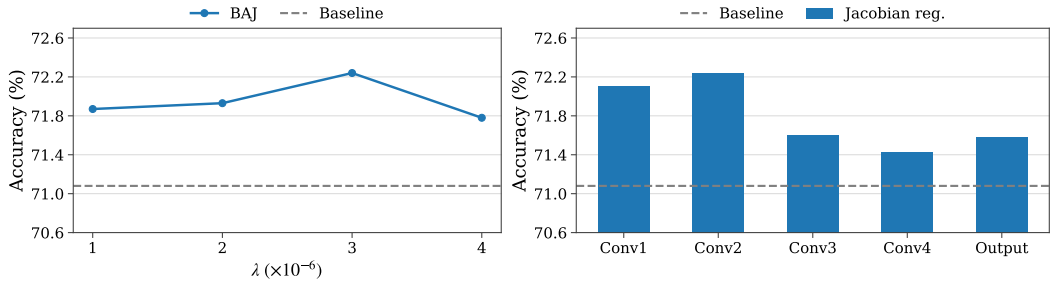
We use the CLGN architecture from prior work (Petersen et al., 2024). The model consists of convolutional logic-tree blocks with 2×2 max pooling, followed by additional logic layers and a scaled group-sum classifier. All learnable transformations are implemented as differentiable logic-gate layers under the same relaxation described in Section 3; the stage names only indicate where the layers appear in the architecture.

All models are trained with Adam using a learning rate of 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and no weight decay. We use a batch size of 16 and train each model for 10^5 iterations. Following prior DLGN training practice, we use residual initialization.

For a fair comparison, all methods use the same architecture and training protocol unless otherwise specified. In addition to the unregularized CLGN baseline, we consider Output Jacobian regularization (OJ), which regularizes the input-to-logit Jacobian following Hoffman et al. (2019). FJ and BAJ are applied to the output of the second convolution-pooling block, whereas OJ is applied to the class logits. Additional architecture and training details are provided in Appendix B.1.

6.2 Results

Table 1 reports the top-1 classification accuracy across the evaluated benchmarks. Both FJ and BAJ improve over the unregularized baseline on all five datasets. FJ achieves the highest accuracy on SVHN, while BAJ achieves the highest accuracy on MNIST, CIFAR-10, CIFAR-100, and Tiny-ImageNet. OJ also improves over the baseline on all five datasets, but remains below both FJ



(a) Regularization strength (b) Regularization target

Figure 2: Ablation study on CIFAR-10 with the 31-threshold setting. (a) Effect of the BAJ regularization coefficient λ when the regularization target is fixed to the output of the second convolution-pooling block. (b) Effect of the regularization target. The output target corresponds to an output-level Jacobian penalty on the class logits, while convolutional targets correspond to BAJ applied to intermediate relaxed Boolean features.

and BAJ. These results show that regularizing intermediate feature sensitivity provides consistent improvements in classification performance across the evaluated CLGN benchmarks.

We further evaluate classification performance under perturbations applied in raw image space before input encoding. For each perturbation, we sample a random direction, normalize it to unit ℓ_2 norm, scale it to a fixed perturbation magnitude, and clamp the resulting image to the valid input range. The perturbed raw image is then passed through the same preprocessing pipeline as the clean input. We use a perturbation scale of $\sigma = 8/255$ and average the resulting accuracy over $R = 16$ perturbations per image. Because MNIST is provided directly to the model as a continuous grayscale input, whereas the other image benchmarks use threshold-based binary encoding, we exclude MNIST from this comparison.

Table 2 reports the resulting perturbed classification accuracy. On CIFAR-10, CIFAR-100, and Tiny-ImageNet, BAJ achieves the highest perturbed accuracy, while FJ also improves over the baseline on these datasets. In particular, the improvements are more pronounced on Tiny-ImageNet, where the perturbed accuracy increases from 20.42% for the baseline to 22.74% with FJ and 23.22% with BAJ. On SVHN, however, FJ decreases perturbed accuracy relative to the baseline, while BAJ remains approximately unchanged and OJ achieves the highest value.

6.3 Ablation Study

We conduct ablation studies on CIFAR-10 using the 31-threshold input-encoding setting to examine two design choices: the regularization strength and the regularization target. Unless otherwise stated, we use the same CLGN architecture and training protocol as in the main experiments.

We first study the effect of the BAJ regularization coefficient λ. The coefficient controls the strength of feature-level regularization applied to the selected intermediate representation. We evaluate $\lambda \in \{1, 2, 3, 4\} \times 10^{-6}$ and select the value achieving the highest accuracy among the tested settings. BAJ outperforms the baseline across the tested range of λ, with the best performance obtained at $\lambda = 3 \times 10^{-6}$. Classification performance decreases when λ is increased beyond this value, suggesting that a moderate regularization strength is preferable to an excessively strong sensitivity penalty.

Effect of regularization target. We next study where feature-level regularization should be applied. Since intermediate relaxed Boolean features serve as operands for subsequent logic gates, the choice of the regularization target determines the point at which feature sensitivity is regularized within the network. Stabilizing very early features may overly constrain low-level representations, while stabilizing later features allows perturbations to propagate through more logic layers before regularization is applied. Figure 2(b) compares different regularization targets. Among the tested targets, applying BAJ to the output of the second convolution-pooling block achieves the highest accuracy. This suggests that stabilizing an intermediate representation before deeper logical compo-

sition provides a favorable trade-off between reducing feature sensitivity and preserving discriminative information. Based on this result, we use the output of the second convolution-pooling block as the default regularization target in the main experiments.

7 Analysis

7.1 Feature Stability under Input Perturbations

We first examine whether the proposed regularizers reduce input-induced variation in intermediate relaxed Boolean features. In this analysis, the reported convolutional stages denote block-level representations after the corresponding convolution-pooling blocks. The regularization target corresponds to the Conv2 block output.

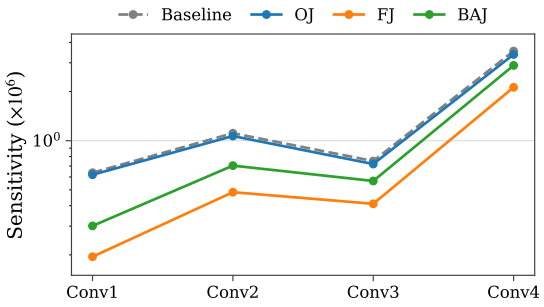


Figure 3: Finite-perturbation sensitivity of Boolean features on CIFAR-10. Conv1–Conv4 denote the outputs of the corresponding convolution-pooling blocks. The vertical axis is shown on a logarithmic scale.

Given an input x, we construct a perturbed input

$$\tilde{x} = x + \delta u, \qquad \|u\|_2 = 1,$$

where u is a random perturbation direction and δ controls the perturbation magnitude. For each stage l, we measure finite-perturbation feature sensitivity as

$$\mathcal{S}_l = \mathbb{E}_{x,u}\left[n \frac{\left\| h^{(l)}(\tilde{x}) - h^{(l)}(x) \right\|_2^2}{\delta^2 + \epsilon} \right], \tag{14}$$

where n is the input dimensionality. This metric measures the response of intermediate relaxed Boolean features to a small input perturbation, with lower values indicating greater feature stability.

Figure 3 shows the finite-perturbation sensitivity across the four convolutional stages. OJ produces only a modest reduction relative to the baseline, whereas FJ and BAJ substantially reduce sensitivity across all measured stages. At the regularization target, Conv2, FJ produces the largest reduction, followed by BAJ, while OJ remains close to the baseline. Importantly, the reductions induced by FJ and BAJ are not confined to the directly regularized representation, but remain visible at the subsequent Conv3 and Conv4 stages.

FJ exhibits the lowest sensitivity throughout the network, which is consistent with its objective of uniformly penalizing the feature Jacobian norm. BAJ produces a more moderate reduction, remaining between FJ and the baseline across the measured stages. This behavior is consistent with the selective nature of BAJ: rather than penalizing all feature dimensions equally, it assigns greater weight to features closer to the binarization boundary.

7.2 Downstream Influence of Stabilized Features

Having established that FJ and BAJ reduce input-induced variation in intermediate relaxed Boolean features, we next examine how the stabilized representation participates in subsequent logical computation. In particular, we analyze the sensitivity of the final logits to individual feature dimensions at the regularization target.

Let $h^{(k^*)}(x)$ denote the Conv2 representation used as the regularization target. We compute the downstream Jacobian

$$J_{k^* \to L}(x) = \frac{\partial s(x)}{\partial h^{(k^*)}(x)}, \tag{15}$$

which measures the local influence of the selected intermediate representation on the output logits. We summarize this Jacobian using the mean squared influence magnitude, the effective dimension ratio of the normalized influence distribution, and the top-5% influence mass. Detailed definitions of these statistics are provided in Appendix B.4.

Table 3: Downstream influence of the Conv2 representation on CIFAR-10. Mean influence is reported in units of 10^{-4}; effective dimension ratio and top-5% mass characterize influence concentration.

Method	Mean infl.	Eff. dim.	Top-5% mass
Baseline	8.755	0.085	0.539
OJ	8.721	0.092	0.537
FJ	8.137	0.068	0.609
BAJ	8.165	0.072	0.599

Table 3 reports the downstream influence statistics for the Conv2 representation. OJ remains close to the baseline across all three statistics, indicating that output-level Jacobian regularization only weakly alters the downstream influence structure of the intermediate representation. In contrast, although FJ and BAJ substantially reduce finite-perturbation feature sensitivity, their mean downstream influence remains at a similar overall scale to the baseline. The mean influence changes from 8.755×10^{-4} for the baseline to 8.721×10^{-4} with OJ, 8.137×10^{-4} with FJ, and 8.165×10^{-4} with BAJ. Thus, FJ and BAJ reduce intermediate feature sensitivity without collapsing the overall strength of the downstream mapping to the output logits.

At the same time, the influence distribution becomes more concentrated. The effective dimension ratio decreases from 0.085 to 0.068 with FJ and 0.072 with BAJ, while the top-5% influence mass increases from 0.539 to 0.609 and 0.599, respectively. These changes indicate that downstream influence becomes more concentrated across feature dimensions, with the most influential dimensions accounting for a larger fraction of the total influence on the final logits.

Together with the feature-sensitivity results in Figure 3, these results highlight a clear distinction between feature-level and output-level Jacobian regularization. FJ and BAJ substantially reduce the perturbation sensitivity of intermediate Boolean features while largely preserving the overall strength of their downstream influence on the final logits. In contrast, OJ remains much closer to the baseline in both intermediate feature sensitivity and downstream influence structure. These observations suggest that directly stabilizing intermediate Boolean operands induces a distinct change in the representations used for subsequent logical composition.

8 Conclusion

We studied CLGNs from the perspective of the stability of intermediate relaxed Boolean features. Because these features serve as operands for subsequent logic gates, their input-induced variations can affect downstream logical composition. Motivated by the feature-dependent local Jacobians of relaxed logic gates, we proposed FJ and its binarization-aware variant, BAJ, which places greater emphasis on features near the binarization boundary.

Experiments showed that FJ and BAJ consistently improve classification accuracy over the unregularized baseline across the evaluated benchmarks and substantially reduce the finite-perturbation sensitivity of intermediate Boolean features. Our downstream analysis further showed that feature-level regularization changes how the stabilized representation influences the final logits: FJ and BAJ largely preserve the overall magnitude of downstream influence while concentrating it on a smaller subset of feature dimensions. In contrast, output-level Jacobian regularization remains much closer to the baseline in both intermediate feature sensitivity and downstream influence structure.

Both FJ and BAJ improve classification performance while reducing the sensitivity of intermediate Boolean features, supporting the importance of feature stability in CLGNs. At the same time, the relative magnitude of the sensitivity reduction does not directly determine the relative classification performance: FJ achieves the strongest overall stabilization, whereas BAJ attains higher accuracy on several benchmarks despite a more moderate reduction in feature sensitivity. These results suggest that improving intermediate feature stability is beneficial for logical composition, while the degree of global sensitivity reduction alone does not fully characterize its effect on predictive performance.

AI use statement

In this work, we used generative AI tools to provide feedback on the interpretation of experimental results, assist with literature retrieval and discovery, draft and edit parts of the paper. All AI-assisted suggestions, technical claims, references, experimental interpretations, and paper content were reviewed and verified by the authors. Generative AI tools were not used to generate synthetic datasets

or to prove mathematical claims. The authors take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility statement

Appendix A provides the stochastic estimators used for FJ and BAJ together with the theoretical analysis of downstream Jacobian perturbations. Appendix B describes the implementation and evaluation details, including the architecture and training configuration, input encoding, raw-input perturbation protocol, and downstream-influence metrics. Appendix C provides an additional output-sensitivity analysis under the same perturbation protocol. Together, these sections specify the methodological, implementation, and evaluation details used in the reported experiments.

References

Shaojie Bai, Vladlen Koltun, and Zico Kolter. Stabilizing equilibrium models by jacobian regularization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 554–565. PMLR, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

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*, pp. 3498–3507, June 2026.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. In *Conference on the Mathematical Theory of Deep Learning (DeepMath)*, 2019.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Josip Jukić and Jan Šnajder. From robustness to improved generalization and calibration in pre-trained language models. *Transactions of the Association for Computational Linguistics*, 13: 264–280, 2025. doi: 10.1162/tacl_a_00739.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann Le Cun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324, 1998.

Hongjae Lee, Myungjun Son, Jaeseong Yu, and Seung-Won Jung. Logicir: Logic gate networks for image restoration. In *European Conference on Computer Vision*, 2026.

Wout Mommen, Lars Keuninckx, Matthias Hartmann, and Piet Wambacq. A method for optimizing connections in differentiable logic gate networks. 2025.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Sejin Park, Hongjae Lee, Changwoo Han, and Seung-Won Jung. Overcoming kernel redundancy for scaling logic gate networks. In *Advances in Neural Information Processing Systems*, 2026.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. *Advances in Neural Information Processing Systems*, 35:2006–2018, 2022.

Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 121185–121203. Curran Associates, Inc., 2024. doi: 10.52202/079017-3851.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Lukas Rüttgers, Till Aczel, Andreas Plesner, and Roger Wattenhofer. Light differentiable logic gate networks. In *Proceedings of the International Conference on Learning Representations*, 2026.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017. doi: 10.1109/TSP.2017.2708039.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning*, pp. 6105–6114, 2019.

Shakir Yousefi, Andreas Plesner, Till Aczel, and Roger Wattenhofer. Mind the gap: Removing the discretization gap in differentiable logic gate networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2025.

A Methodological and Theoretical Details

A.1 Stochastic Estimation of FJ and BAJ Regularizers

This appendix describes the stochastic estimator used to compute the FJ and BAJ regularizers during training. Let

$$D_{k^*}(x) = \frac{\partial h^{(k^*)}(x)}{\partial x} \in \mathbb{R}^{d_{k^*} \times n}$$

be the feature Jacobian at the selected regularization layer. The FJ regularizer is proportional to

$$\|D_{k^*}(x)\|_F^2 .$$

Computing this quantity exactly is expensive because it requires the full Jacobian of a high-dimensional intermediate feature with respect to the input. We therefore estimate it using a random projection.

Let $v \in \mathbb{R}^{d_{k^*}}$ be a random vector satisfying

$$\mathbb{E}_v[vv^\top] = I.$$

Then,

$$\mathbb{E}_v \left[\left\| v^\top D_{k^*}(x) \right\|_2^2 \right] = \|D_{k^*}(x)\|_F^2 . \tag{16}$$

Therefore, the feature Jacobian norm can be estimated using the gradient of the scalar projection

$$v^\top h^{(k^*)}(x)$$

with respect to the input:

$$v^\top D_{k^*}(x) = \frac{\partial}{\partial x}\left(v^\top h^{(k^*)}(x)\right). \tag{17}$$

In practice, we use a one-sample estimator,

$$\widehat{\mathcal{R}}_{\mathrm{FJ}}^{(k^*)} = \frac{1}{2}\left\|\frac{\partial}{\partial x}\left(v^\top h^{(k^*)}(x)\right)\right\|_2^2, \tag{18}$$

where a new random projection vector v is sampled during training.

For BAJ, the regularizer uses feature-wise binarization-aware weights. Let

$$\omega_j^{(k^*)}(x) = 4h_j^{(k^*)}(x)\left(1 - h_j^{(k^*)}(x)\right)$$

denote the binarization score, and define

$$W_{k^*}(x) = \mathrm{diag}\left(\mathrm{sg}[\omega_1^{(k^*)}(x)], \ldots, \mathrm{sg}[\omega_{d_{k^*}}^{(k^*)}(x)]\right).$$

The weighted feature Jacobian norm is

$$\left\|W_{k^*}(x)^{1/2} D_{k^*}(x)\right\|_F^2. \tag{19}$$

Using the same random-projection identity, this quantity can be written as

$$\mathbb{E}_v\left[\left\|v^\top W_{k^*}(x)^{1/2} D_{k^*}(x)\right\|_2^2\right] = \left\|W_{k^*}(x)^{1/2} D_{k^*}(x)\right\|_F^2. \tag{20}$$

Equivalently, defining

$$\tilde{v}(x) = W_{k^*}(x)^{1/2} v,$$

where $W_{k^*}(x)$ is treated as constant under differentiation, we estimate the weighted norm using

$$\widehat{\mathcal{R}}_{\mathrm{BAJ}}^{(k^*)} = \frac{1}{2}\left\|\frac{\partial}{\partial x}\left(\tilde{v}(x)^\top h^{(k^*)}(x)\right)\right\|_2^2. \tag{21}$$

During training, gradients are stopped through the binarization-aware weights $\omega_j^{(k^*)}(x)$. Thus, the weights determine which feature sensitivities are penalized more strongly, but they do not directly push relaxed Boolean features toward or away from the binarization boundary. This detach operation is important because BAJ is intended to reweight the feature-sensitivity penalty, rather than to act as a direct binarization loss.

A.2 Jacobian Perturbation in Downstream Logical Composition

This appendix provides a first-order view of how perturbations in intermediate relaxed Boolean features can affect downstream logical composition. Consider the layer Jacobian $J_l(h^{(l)})$. Since the Jacobian entries of relaxed logic gates depend on the current feature values, a perturbation $\delta h^{(l)}$ induces a perturbation in the layer Jacobian:

$$\Delta J_l = J_l(h^{(l)} + \delta h^{(l)}) - J_l(h^{(l)}). \tag{22}$$

For the i-th gate, the Jacobian entries are

$$\begin{aligned}
[J_l]_{i,p_i^{(l)}} &= w_{i,1}^{(l)} + w_{i,3}^{(l)} h_{q_i^{(l)}}^{(l)}, \\
[J_l]_{i,q_i^{(l)}} &= w_{i,2}^{(l)} + w_{i,3}^{(l)} h_{p_i^{(l)}}^{(l)}.
\end{aligned} \tag{23}$$

Therefore, a perturbation in the relaxed Boolean features changes these Jacobian entries as

$$\begin{aligned}
\Delta[J_l]_{i,p_i^{(l)}} &= w_{i,3}^{(l)} \delta h_{q_i^{(l)}}^{(l)}, \\
\Delta[J_l]_{i,q_i^{(l)}} &= w_{i,3}^{(l)} \delta h_{p_i^{(l)}}^{(l)}.
\end{aligned} \tag{24}$$

This shows that perturbing one operand can change the local sensitivity with respect to the other operand.

For an intermediate layer k, the downstream Jacobian from $h^{(k)}$ to the output logits is

$$J_{k\to L}(x) = GJ_{L-1}\cdots J_k. \tag{25}$$

If each downstream layer Jacobian is perturbed as $J_m + \Delta J_m$, then the first-order perturbation of the downstream Jacobian product is

$$\Delta J_{k\to L} \approx G\sum_{m=k}^{L-1} J_{L-1}\cdots J_{m+1}\Delta J_m J_{m-1}\cdots J_k, \tag{26}$$

where empty products at the endpoints are interpreted as identity matrices.

This expansion illustrates how perturbations in intermediate relaxed Boolean features can affect downstream logical composition in two ways. First, they change the activations passed to later logic gates. Second, because relaxed logic gates have feature-dependent Jacobians, they can also change the local Jacobians of downstream layers. This provides the structural motivation for stabilizing intermediate relaxed Boolean features in the main method.

B Implementation and Evaluation Details

B.1 Architecture and Training Details

This appendix provides additional details on the CLGN architecture and training protocol used in the experiments. All models follow the same overall architecture family, consisting of four convolutional logic-tree blocks with 2×2 max pooling, followed by three fully connected logic layers and a scaled group-sum classifier. For SVHN, CIFAR-10, CIFAR-100, and Tiny-ImageNet, we use 31-threshold binary input encoding, whereas MNIST is provided directly as a single-channel grayscale input.

Each convolutional logic block consists of a convolutional logic-tree block followed by 2×2 max pooling with stride 2. All convolutional logic layers use a logic-tree depth of 3, a 3×3 receptive field, and padding that preserves the spatial resolution before pooling. Let k denote the dataset-specific base width reported in Table 4. The output widths of the four convolutional logic layers are k, $4k$, $16k$, and $32k$, respectively. After four pooling operations, the spatial resolution is reduced by a factor of 16 along each spatial dimension.

Throughout the paper, we use Conv1–Conv4 to denote the block-level representations at the outputs of the first through fourth convolution-pooling blocks, respectively, as illustrated in Figure 4. Thus, the default regularization target, Conv2, corresponds to the intermediate relaxed Boolean representation at the output of the second convolution-pooling block. This representation is subsequently used as input to the third convolutional logic block.

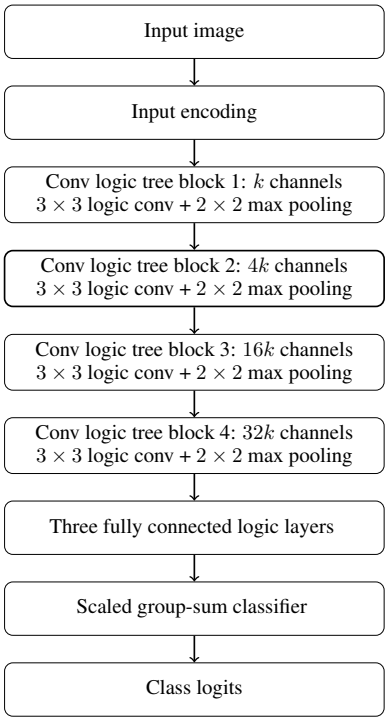


Figure 4: Schematic overview of the CLGN architecture.

After the fourth convolution-pooling block, the representation is flattened and processed by three fully connected logic layers. For MNIST, SVHN, CIFAR-100, and Tiny-ImageNet, which use the generic architecture, the output dimensions of these layers are $64Ck$, $32Ck$, and $16Ck$, respectively, where C denotes the number of classes. For CIFAR-10, we use a CIFAR-10-specific architecture with fully connected logic-layer widths $1280k$, $640k$, and $320k$. The final relaxed Boolean outputs are aggregated into class logits using a scaled group-sum classifier with the dataset-specific scaling factor τ reported in Table 4.

Table 4: Dataset-specific base width and group-sum scaling factor. The four convolutional logic layers use output widths k, $4k$, $16k$, and $32k$, respectively.

Dataset	Base width k	τ
MNIST	64	15
SVHN	512	60
CIFAR-10	512	60
CIFAR-100	512	40
Tiny-ImageNet	256	40

All compared methods use the same architecture and training protocol unless otherwise specified, differing only in their regularization objectives. FJ and BAJ regularize the Conv2 representation, corresponding to the output of the second convolution-pooling block, whereas OJ regularizes the input-to-logit Jacobian. For BAJ, gradients are stopped through the binarization scores when they are used to weight the feature-Jacobian penalty.

B.2 Input Encoding

For SVHN, CIFAR-10, CIFAR-100, and Tiny-ImageNet, we use threshold-based binary input encoding. Let $x_{\mathrm{raw}} \in [0,1]^{C\times H\times W}$ denote a raw input image. For an input encoding with M thresholds, we define the threshold values as

$$\theta_m = \frac{m}{M+1}, \qquad m = 1, \ldots, M. \tag{27}$$

The encoded input is obtained by applying all thresholds independently to each input channel:

$$E(x_{\mathrm{raw}}) = \mathrm{concat}_{m=1}^{M} \mathbf{1}\left[x_{\mathrm{raw}} > \theta_m\right], \tag{28}$$

where the concatenation is performed along the channel dimension. Thus, an input image with C channels is transformed into a binary tensor with MC channels.

For all threshold-encoded benchmarks, we use $M = 31$ thresholds. Since these datasets contain RGB images, each input is transformed into 93 binary channels, and the encoded tensor $E(x_{\mathrm{raw}})$ is used as the input to the CLGN. MNIST is an exception: its original single-channel grayscale input in $[0,1]$ is provided directly to the model without threshold expansion. For the perturbation evaluation on the threshold-encoded benchmarks, the same encoding function is applied to both clean and perturbed raw inputs.

B.3 Perturbation Evaluation Protocol

For each raw input image x_{raw}, we sample

$$z^{(r)} \sim \mathcal{N}(0, I), \qquad u^{(r)} = \frac{z^{(r)}}{\|z^{(r)}\|_2}, \tag{29}$$

and construct

$$\tilde{x}_{\mathrm{raw}}^{(r)} = \mathrm{clip}\left(x_{\mathrm{raw}} + \delta u^{(r)}, 0, 1\right), \qquad \delta = \sigma\sqrt{d_{\mathrm{raw}}}, \tag{30}$$

where d_{raw} is the dimensionality of the raw input image. We use $\sigma = 8/255$ and $R = 16$.

The clean and perturbed model inputs are then obtained using the same threshold encoding function $E(\cdot)$:

$$x = E(x_{\mathrm{raw}}), \qquad \tilde{x}^{(r)} = E(\tilde{x}_{\mathrm{raw}}^{(r)}). \tag{31}$$

Perturbed accuracy is computed over all perturbed inputs $\tilde{x}^{(r)}$ and averaged across both test examples and perturbation samples.

B.4 Downstream Influence Metrics

This appendix provides detailed definitions of the downstream-influence statistics used in Section 7.2. We analyze the selected intermediate representation $h^{(k^*)}(x)$ at the output of the second convolution-pooling block. The feature tensor is flattened over its channel and spatial dimensions, and the downstream Jacobian with respect to the output logits is computed in the relaxed network.

Recall that

$$J_{k^* \to L}(x) = \frac{\partial s(x)}{\partial h^{(k^*)}(x)} \in \mathbb{R}^{C \times d_{k^*}}, \tag{32}$$

where C is the number of classes and d_{k^*} is the dimensionality of the flattened intermediate feature.

For each feature dimension j, we define its downstream influence on the output logits as

$$q_j(x) = \sum_{c=1}^{C} \left(\frac{\partial s_c(x)}{\partial h_j^{(k^*)}(x)} \right)^2. \tag{33}$$

Equivalently, $q_j(x)$ is the squared Euclidean norm of the j-th column of $J_{k^* \to L}(x)$. Thus, it measures the local influence of the j-th intermediate feature on the output logits.

The mean influence reported in Table 3 is

$$Q_{\text{mean}} = \mathbb{E}_x \left[\frac{1}{d_{k^*}} \sum_{j=1}^{d_{k^*}} q_j(x) \right]. \tag{34}$$

To characterize how the downstream influence is distributed across feature dimensions, we define the normalized influence

$$p_j(x) = \frac{q_j(x)}{\max\left(\sum_{m=1}^{d_{k^*}} q_m(x), \epsilon \right)}, \tag{35}$$

where $\epsilon = 10^{-12}$ is used for numerical stability.

We measure the concentration of this distribution using the effective dimension ratio

$$\text{EffDimRatio}(x) = \frac{1}{d_{k^*} \max\left(\sum_{j=1}^{d_{k^*}} p_j(x)^2, \epsilon \right)}. \tag{36}$$

This quantity is close to 1 when the downstream influence is distributed uniformly across feature dimensions and decreases as the influence becomes concentrated on a smaller subset of features. We report its average over the evaluation samples:

$$\text{EffDimRatio} = \mathbb{E}_x \left[\text{EffDimRatio}(x) \right]. \tag{37}$$

We additionally report the top-5% influence mass. Let

$$K = \max\left(1, \lceil 0.05 d_{k^*} \rceil\right), \tag{38}$$

and let $\mathcal{I}_K(x)$ denote the indices of the K feature dimensions with the largest values of $q_j(x)$. The top-5% influence mass is defined as

$$M_{\text{top5}} = \mathbb{E}_x \left[\sum_{j \in \mathcal{I}_K(x)} p_j(x) \right]. \tag{39}$$

A larger value indicates that the most influential 5% of feature dimensions account for a larger fraction of the total downstream influence on the output logits. For the analysis in Table 3, these statistics are computed on CIFAR-10 with the 31-threshold input encoding and the Conv2 block output as the selected intermediate representation.

Table 5: Logit sensitivity under raw-input perturbations. Lower values indicate smaller changes in the output logits.

Method	SVHN	CIFAR-10	CIFAR-100	Tiny-ImageNet
Baseline	**38.014**	52.052	257.572	25.288
OJ	39.747	51.663	279.687	26.916
FJ	38.903	**45.370**	218.423	16.916
BAJ	40.088	47.111	**183.398**	**15.965**

C Additional Output-Sensitivity Analysis

As a complementary analysis to the perturbed-accuracy evaluation in Section 6, we examine how intermediate feature stabilization affects the sensitivity of the final output logits. We use the same raw-input perturbation protocol described in Appendix B.3. For each raw input x_{raw} and its r-th perturbed version $\tilde{x}_{\mathrm{raw}}^{(r)}$, let x and $\tilde{x}^{(r)}$ denote the corresponding model inputs after threshold encoding. We measure logit sensitivity as

$$\mathrm{LS}(x_{\mathrm{raw}}) = \frac{1}{R}\sum_{r=1}^{R} \frac{\left\| s(\tilde{x}^{(r)}) - s(x) \right\|_2^2}{\left\| \tilde{x}_{\mathrm{raw}}^{(r)} - x_{\mathrm{raw}} \right\|_2^2 + \epsilon}, \tag{40}$$

where $s(\cdot)$ denotes the class logits and $\epsilon > 0$ is a small constant for numerical stability. The denominator is computed in raw image space after clipping, such that the metric is normalized by the actual perturbation magnitude. Lower values indicate smaller output-level changes under input perturbations. As in the perturbed-accuracy evaluation, we report results only for the threshold-encoded benchmarks and exclude MNIST.

Table 5 shows that the effect of feature-level stabilization can extend to the final output mapping, although the trend is dataset-dependent. On CIFAR-10, CIFAR-100, and Tiny-ImageNet, FJ and BAJ reduce logit sensitivity relative to the unregularized baseline. The reductions are particularly pronounced on CIFAR-100 and Tiny-ImageNet, where BAJ achieves the lowest sensitivity. FJ achieves the lowest value on CIFAR-10.

In contrast, neither FJ nor BAJ reduces logit sensitivity on SVHN, despite both methods improving clean classification accuracy on this dataset. OJ remains close to the baseline on CIFAR-10 and increases logit sensitivity on SVHN, CIFAR-100, and Tiny-ImageNet. These results indicate that the reduction of intermediate feature sensitivity does not necessarily translate uniformly into lower output-level sensitivity across datasets. We therefore treat logit sensitivity as a complementary characterization of model behavior rather than as the primary evidence for intermediate Boolean feature stability.