WHICH GEOMETRY PREDICTS GENERALIZATION DEPENDS ON WHAT YOU VARY — ACTIVATION OR OPTIMIZER?

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work introduced Rényi sharpness, a reparameterization-invariant measure of loss-Hessian geometry (Zhang et al., 2026), and reported that it correlates strongly and consistently with the generalization gap, the difference between training and test accuracy. That evidence, however, comes from varying the optimizer while the activation function is held fixed. We invert this setup. We fix the optimizer to stochastic gradient descent (SGD), with a transfer check under a sharpness-aware optimizer, and vary the activation function across seven common functions. Across multiple datasets and architectures, we compare weight-space, input-space, and activation-level measures by their signed Kendall rank correlation with the activation-induced generalization gap. In this setting, Rényi sharpness is weak and changes sign across architectures and datasets (mean Kendall τ of $+0.14$ in our main comparison). In contrast, the input Jacobian, which measures how sensitive the output is to the input, and a mechanistically motivated activation-level quantity are positively associated with the gap in every setting of our main comparison (mean τ of $+0.65$ each). This quantity, the negative-branch leak, is the mean squared activation slope on negative pre-activations, $\mathbb{E}_{z<0}[\varphi'(z)^2]$, where $\varphi(\cdot)$ is the activation and z the pre-activation. A controlled intervention that gates only the negative branch provides causal evidence that the leak contributes to input sensitivity. Guided by this mechanism, we control the leak in two ways. First, we reduce it directly by scaling down the activation's negative branch, with no added loss term. This improves clean accuracy for every activation in our main setting, by up to 2.45 points, and by up to 10.85 points on a wide residual network (WRN-28-10). Second, we add a leak penalty to the training loss, together with the established input-Jacobian penalty (Hoffman et al., 2019). The leak term mainly affects clean accuracy, while the input-Jacobian term drives corruption robustness. Which geometry predicts generalization depends on the axis along which models vary.

1 INTRODUCTION

Overparameterized deep networks generalize far better than classical theory predicts, and a large body of work attributes this to the *geometry* of the loss landscape at the solution, in particular to the flatness of the minimum: flatter minima tend to generalize better (Hochreiter & Schmidhuber, 1997; Keskar et al., 2017). This idea is now operational in sharpness-aware optimizers that steer optimization toward flat regions and improve test accuracy: *Sharpness-Aware Minimization for Efficiently Improving Generalization* (SAM) (Foret et al., 2021), its scale-invariant variant *ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks* (ASAM) (Kwon et al., 2021), and *Rényi Sharpness: A Novel Sharpness That Strongly Correlates with Generalization* (RSAM) (Zhang et al., 2026). A known difficulty is that naive sharpness can be inflated or deflated purely by rescaling the weights, as shown in *Sharp Minima Can Generalize For Deep Nets* (Dinh et al., 2017); this motivated reparameterization-invariant measures such as Rényi sharpness (Zhang et al., 2026), although even adaptive sharpness can correlate weakly with generalization once training settings vary (Andriushchenko et al., 2023).

This literature largely studies *weight-space* geometry, and the evidence for Rényi sharpness varies the *optimizer* while holding the architecture fixed. The activation function, an equally basic archi-

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.

tectural choice that reshapes the loss landscape and the input-to-output map, has received far less attention as a lever on generalization geometry. This raises a natural question that the optimizer-centric view does not answer: if we hold the optimizer fixed and change only the activation, which geometric quantity, if any, predicts the generalization differences that may result?

Holding the optimizer fixed (plain SGD in our main experiments) and varying the activation across seven common activation functions, we find that weight-space Rényi sharpness, the measure proposed by Zhang et al. (2026), is a *less consistent* predictor of the generalization gap under activation variation than reported under optimizer variation. In its place, an *activation-level* quantity tracks the activation-induced differences. We call it the negative-branch leak: the mean squared activation slope on negative pre-activations. Intuitively, it measures how much input variation the activation's negative side still passes through; it is zero for the rectified linear unit (ReLU). A controlled intervention shows that reducing the leak reduces input sensitivity, most clearly for the higher-leak activations. Which geometry predicts generalization, then, depends on what one varies, the optimizer or the activation. The leak mechanism can be controlled in two ways. First, the leak is a property of the activation, so we can reduce it *directly* by scaling down the negative branch of a standard activation. This adds no loss term. With the optimizer fixed, it can improve clean accuracy across activations, architectures, and datasets, with a small non-zero leak often providing the best setting, clearest on ResNet-18. Second, keeping the activation fixed, we can penalize the leak *during training* through an enriched objective (cross-entropy plus a leak penalty and an input-Jacobian penalty) whose two penalty terms play distinct roles: the leak term mainly affects clean accuracy while the input-Jacobian term is the primary lever on robustness to common corruptions.

We answer this question along a causal ladder that forms the backbone of this work. First, at the level of *measurement*, we correlate each geometric quantity with the generalization gap across activations and find that Rényi sharpness, which is reliable under optimizer variation (Zhang et al., 2026), is no longer reliable here, while an activation-level quantity is (§6.2). Second, at the level of a *controlled intervention*, we change only the negative branch of the activation (Eq. (4)); the positive branch is unchanged up to the common output standardization of Appendix C.1. This provides causal evidence that the leak raises input sensitivity (§6.3). The same intervention is a penalty-free way to reduce the leak and improve clean accuracy (§4). Third, at the level of a *training intervention*, we place the leak, together with the established input-Jacobian penalty, into the objective and control the same mechanism during optimization, exposing a trade-off between clean accuracy and corruption robustness (§6.4–§6.5). Each rung tests, and then uses, the same mechanism, and each builds on the previous one.

Contributions.

1. Axis-dependence of geometric predictors, with a causal mechanism. With the optimizer fixed and the activation varied, Rényi sharpness is a less consistent predictor of the generalization gap than reported under optimizer variation (Zhang et al., 2026), whereas the activation-level negative-branch leak $\mathbb{E}_{z<0}[\varphi'(z)^2]$ (Def. 1) tracks the activation-induced differences (Figure 1); a controlled gating intervention (the gate of Eq. (4)) that alters only the negative branch provides causal evidence that the leak raises input sensitivity (§6.3). We keep this first-derivative leak distinct from the activation's second derivative φ'', which enters the Hessian through a separate channel (§3).

2. Reducing the negative-branch leak is a penalty-free architectural design lever. With the optimizer fixed, scaling down an excessive negative-branch leak of a standard activation can improve clean accuracy across multiple activations, architectures, and datasets, without introducing an additional loss term or optimization objective (no change to the standard training procedure). The best leak level is setting-dependent and is often a *small non-zero* value rather than complete shutoff, clearest on ResNet-18. This is a design rule on the leak coefficient a (the negative-branch scale), not a new activation (§4).

3. Mechanism-derived training objective. Keeping the activation fixed, we introduce a leak penalty derived from the negative-branch mechanism and evaluate it alone, an established input-Jacobian penalty (*Robust Learning with Jacobian Regularization*; Hoffman et al., 2019) alone, and their combination. In clean-accuracy and corruption-robustness evaluations, these terms play distinct roles: the leak term primarily affects clean accuracy, whereas the input-Jacobian term drives the improvement on common corruptions, so the two act on complementary axes and can be combined (§5, §6.5).

1.1 Related Work

Sharpness, flatness, and generalization. Flat minima have long been linked to better generalization (Hochreiter & Schmidhuber, 1997; Keskar et al., 2017), an idea now built into sharpness-aware optimizers such as SAM (Foret et al., 2021) and its scale-invariant variant ASAM (Kwon et al., 2021). Because scale-sensitive sharpness can be inflated or deflated by reparameterization (Dinh et al., 2017), recent work seeks reparameterization-invariant measures such as Rényi sharpness (Zhang et al., 2026), whose authors also propose the RSAM optimizer. We take Rényi sharpness as our reference weight-space measure, and note that the evidence for it (Zhang et al., 2026) varies the *optimizer* while holding the architecture fixed.

Activation functions and landscape geometry. A separate line studies the activation function, whose negative branch distinguishes ReLU from relatives with a non-zero negative branch (Clevert et al., 2016; Hendrycks & Gimpel, 2016; Ramachandran et al., 2018; Misra, 2020) and which recent work continues to tune explicitly (Guo et al., 2024; Liu, 2025). The closest prior study of activation-induced landscape geometry, Bosman et al. (2023), is limited to three activations in small feed-forward networks.

Input-space sensitivity and Jacobian regularization. A third line characterizes generalization through the input-to-output map, where a smaller input-Jacobian norm is associated with better generalization (Novak et al., 2018; Sokolić et al., 2017) and is penalized during training as established prior art (Hoffman et al., 2019; Varga et al., 2018); we adopt this Jacobian penalty unchanged and do not claim it as novel.

This work. Our setting differs from all three lines: we hold the optimizer fixed, vary the activation across modern networks, and ask which geometric quantity tracks the resulting generalization differences, then connect the negative-branch first derivative to input sensitivity as a quantity that can be controlled architecturally or penalized during training. Appendix A expands each of these three lines of work in detail.

2 Problem Setup and Measures

We train a network f_θ with the cross-entropy (CE) loss on standard image classification and evaluate a range of geometric measures on the trained models, relating each to the generalization gap; in the correlation study (§6.2) we vary only the activation. Let θ denote the network parameters and φ its elementwise activation; at activation layer l, z_l is the pre-activation, and φ', φ'' are the first and second derivatives of φ, respectively. We write $L_{\mathrm{CE}}(\theta)$ for this training cross-entropy loss, averaged over the training set. Three geometric measures are central to our analysis and defined here in full: the weight-space Rényi sharpness of Zhang et al. (2026), input-space sensitivity via the input Jacobian, and activation-level behavior via the negative-branch leak. Alongside these three, we evaluate standard weight-space baselines, including the Hessian trace, parameter norm, Fisher–Rao norm, PAC-Bayes sharpness, and SAM/ASAM sharpness; these auxiliary measures enter the comparison of Figure 1 and are defined in Appendix D. All measures use the same trained models, so the comparison is controlled. These three quantities live at three levels of the network: weight-space curvature at the parameters θ, input-space sensitivity at the network input x, and the negative-branch leak at the internal pre-activations z_l (which are distinct from the input x). A schematic placing each quantity at its level is in Appendix B (Figure 2).

Hessian and its spectrum. Let $H = \nabla^2_\theta L_{\mathrm{CE}}$ be the loss Hessian with eigenvalues $\{\lambda_i\}$, computed on a fixed training subset (Appendix D). We never form H; we probe it with stochastic Lanczos quadrature (SLQ) (Ubaru et al., 2017).

Rényi sharpness (weight space). Zhang et al. (2026) define the Rényi entropy of the (absolute) spectrum and take its negative as the sharpness:

$$\mathcal{H}_\alpha \;=\; \frac{1}{1-\alpha}\,\log\frac{\mathrm{Tr}\big(|H|^\alpha\big)}{\mathrm{Tr}(H)^\alpha} \;=\; \frac{1}{1-\alpha}\,\log\frac{\sum_i |\lambda_i|^\alpha}{\big(\sum_i \lambda_i\big)^\alpha}, \qquad \text{Rényi sharpness} = -\mathcal{H}_\alpha. \quad (1)$$

Zhang et al. (2026) show that it is invariant to the cosmetic weight rescalings of Dinh et al. (2017), unlike the trace or maximum eigenvalue. We report $\alpha \in \{0.5, 1.5\}$ and use the signed trace $\mathrm{Tr}(H)$ in the denominator, following Zhang et al. (2026).

Input Jacobian (input space). The Frobenius norm of the input-to-output Jacobian,

$$\left\|\partial f/\partial x\right\|_F, \tag{2}$$

measured on held-out inputs. Higher values mean the map is more sensitive to input perturbations.

Definition 1 (Negative-branch leak)**.** For an activation φ (the standardized activation of Appendix C.1), the *negative-branch leak* is the mean squared activation slope on the negative branch, layer-balanced across the set $\mathcal{A}$ of activation layers,

$$\ell \;=\; \frac{1}{|\mathcal{A}|}\sum_{l\in\mathcal{A}} \mathbb{E}_{z_l<0}\big[\varphi'(z_l)^2\big], \tag{3}$$

where z_l is the pre-activation at layer l, and each expectation is over held-out inputs with $z_l < 0$.

For the activations we study, ReLU has $\ell = 0$ because its negative branch is flat; every other activation we study retains nonzero negative-branch derivatives and therefore has $\ell > 0$. This is a first-derivative quantity and is distinct from curvature. Appendix C plots φ and φ' for the seven activations; their negative-branch derivatives, and hence their leak, differ substantially, so these are not simply smoothed copies of ReLU.

Landscape roughness. As a non-curvature geometry axis, we quantify the roughness of the training-loss landscape. We sample the loss along random rays from the solution and compute the generalized Hurst exponent with multifractal detrended fluctuation analysis (Kantelhardt et al., 2002); a lower exponent indicates a rougher landscape (estimation details in Appendix D).

Generalization gap. We use the last-epoch train-minus-test accuracy, $\mathrm{gap} = \mathrm{acc}_{\text{train}} - \mathrm{acc}_{\text{test}}$, matching the correlation protocol of Zhang et al. (2026). The rank-correlation analysis (§6.2) uses the last-epoch checkpoint; all accuracy tables (§4, §6.3–§6.5 and their appendices) use best-validation checkpoints.

3 Activation-Induced Geometry

How the activation enters the Hessian. Under the Gauss–Newton decomposition of the loss Hessian, the activation slope φ' enters through the Gauss–Newton term and the activation curvature φ'' only through the residual (Appendix B, Eq. (7)). Weight-space curvature, which Rényi sharpness summarizes, is therefore one channel through which the activation changes the geometry, and the next paragraph describes a second, input-space channel.

Mechanism: leakage raises input sensitivity (tested in §6.3). A rectifier gates negative pre-activations: for $z < 0$ it emits a constant, so $\varphi'(z) = 0$ and input variation is not propagated through the activation. A nonzero negative-branch slope instead allows input variation to propagate. The input-to-output Jacobian is assembled from factors $\varphi'(z_l)$, and each layer contributes $\mathbb{E}[\varphi'(z_l)^2]$, whose negative-branch part is the per-layer leak of Eq. (3) weighted by the fraction of negative pre-activations. A rectifier sets that part to zero; a leaky branch makes it positive and so raises input sensitivity. Input sensitivity is a property of the input-to-output map, a different object from the weight-space Hessian (Novak et al., 2018; Sokolić et al., 2017). The two therefore need not move together; §6.2 tests which one tracks the activation-induced gap.

Gating as a controlled instrument. To test the mechanism causally, we use a gate that changes only the negative branch,

$$\varphi_a(z) = \begin{cases} \varphi(z), & z \ge 0, \\ \varphi(0) + a\big(\varphi(z) - \varphi(0)\big), & z < 0, \end{cases} \tag{4}$$

where $a \in [0, 1]$ scales the negative branch only (a=1 recovers the base activation, a=0 shuts the branch to the constant $\varphi(0)$, a rectifier-like function). Differentiating gives $\varphi_a'(z) = \varphi'(z)$ for $z \ge 0$ and $\varphi_a'(z) = a\,\varphi'(z)$ for $z < 0$, so the negative-branch slope scales linearly with a and the leak as $\ell_a = a^2 \ell\, \sigma_{\varphi_1}^2/\sigma_{\varphi_a}^2$ for a fixed pre-activation distribution (the ratio is the standardization factor of Appendix C.1); because retraining shifts that distribution, we measure the leak rather than assume this relation. We call a the leak *coefficient*: it controls how much leak an activation has, and is distinct from the leak ℓ itself (Def. 1), which is the quantity we measure and penalize. The coefficient a therefore provides both a controlled intervention for testing this mechanism and a direct way to reduce the leak ℓ. Either way this is a rule on the leak coefficient of an existing standard activation. In the networks we train from scratch, we also rescale each activation to a common output size,

Table 1: **Negative-branch gating and clean test accuracy.** Mean ± standard deviation (SD) over three seeds on CIFAR-100 (ResNet-18, SGD). $a = 1$ is the original activation, $a = 0$ shuts down the negative branch, and $0 < a < 1$ partially reduces it. Δ is relative to the $a = 1$ baseline.

Activation	Test acc. (%) a=0	Test acc. (%) a=1	Test acc. (%) 0<a<1, best a	Δ (%)
GELU	$76.66_{\pm 0.18}$	$76.81_{\pm 0.01}$	$\mathbf{76.86_{\pm 0.05}}$ **(0.013)**	+0.05
SiLU	$76.09_{\pm 0.27}$	$76.33_{\pm 0.13}$	$\mathbf{76.46_{\pm 0.30}}$ **(0.013)**↑	**+0.13**↑
Mish	$\mathbf{76.66_{\pm 0.23}}$↑	$76.53_{\pm 0.55}$	$\mathbf{76.77_{\pm 0.12}}$ **(0.013)**↑	**+0.24**↑
ELU	$\mathbf{76.67_{\pm 0.33}}$↑	$75.27_{\pm 0.17}$	$\mathbf{76.99_{\pm 0.01}}$ **(0.013)**↑	**+1.72**↑
SELU	$\mathbf{76.46_{\pm 0.13}}$↑	$74.19_{\pm 0.32}$	$\mathbf{76.64_{\pm 0.14}}$ **(0.013)**↑	**+2.45**↑
LeakyReLU	$\mathbf{76.69_{\pm 0.19}}$↑	$76.54_{\pm 0.20}$	$\mathbf{76.77_{\pm 0.39}}$ **(0.013)**↑	**+0.23**↑
Average (6 activations)	**76.54**↑	75.95	**76.75**↑	**+0.80**↑
ReLU (ref)	$76.46_{\pm 0.00}$	n/a	n/a	n/a

which we call *standardizing*: each activation is affinely rescaled to output mean 0 and variance 1 under a standard-normal pre-activation, so that comparisons across activations are not confounded by how large their outputs are (defined precisely in Appendix C.1, Eq. (8)). Appendix C (Fig. 4) illustrates the gate on a representative activation: lowering a scales the negative-branch slope toward zero while leaving the positive branch unchanged (before standardization; after standardization the positive branch is additionally scaled by the common factor $\sigma_{\varphi_1}/\sigma_{\varphi_a} \geq 1$, from 1.07 for GELU to 1.63 for SELU at a=0). This factor raises positive-branch slopes, so the input-Jacobian decrease under gating (§6.3) occurs despite the rescaling, not because of it.

4 Controlling the Negative-Branch Leak

We ask whether the negative-branch leak can be controlled directly at the activation level. We vary the leak coefficient a of Eq. (4) while holding the optimizer, training schedule, and standardized output size fixed, isolating the negative-branch intervention, and measure its effect on clean accuracy and corruption robustness across activations, architectures, and datasets. This is not a new activation: a=1 recovers the original activation, a=0 shuts the negative branch to a rectifier-like endpoint, and plain ReLU is the zero-leak reference.

Setup. For each activation we sweep a from 1 (the original activation) toward 0 (shut branch). A single interior a per setting is selected on a held-out validation split and applied to all activations in that setting; the test set is used only for the final numbers reported here. The ResNet-18 gating runs use weight decay 10^{-4}; the other backbones use 5×10^{-4}.

Reducing the negative-branch leak often improves clean accuracy. On ResNet-18/CIFAR-100 (Table 1), reducing the leak often improves clean accuracy over the original activation (a=1), with the largest gains for the high-leak ELU (+1.72) and SELU (+2.45) and smaller gains for the lower-leak GELU, SiLU, Mish, and LeakyReLU (+0.05 to +0.24); the mean improvement over the six standard activation functions is +0.80. On CIFAR-10 the high-leak activations show the same gain at a=0 (ELU +0.97, SELU +1.33; Table 7); the accompanying input-Jacobian measurements are in Appendix G (Table 7 for CIFAR-10, Table 8 for CIFAR-100).

A small non-zero leak is best on ResNet-18. On ResNet-18/CIFAR-10 with GELU the optimum is a small non-zero leak ($a\approx0.003$) rather than complete shutoff; on ResNet-20 and WRN-28-10 the interior offers no gain over the better endpoint (Appendix G). The finding is a small-leak regime where it matters, not a monotone "lower is better." A finer replicated sweep is reported in Appendix G.1 (Table 9).

The response is setting-dependent. Repeating the a=0 versus a=1 intervention on CIFAR-100 across architectures, the effect is small and mixed on ResNet-20 but substantially larger on the wider WRN-28-10 (GELU +2.34, SELU +10.85; Table 6), and complete shutoff can become unstable in deeper or wider settings. The best leak setting should therefore be searched per setting rather than fixed; the full cross-architecture grids are in Appendix G (Table 6).

A stability boundary. Aggressively shutting the negative branch ($a\to0$) can destabilize training: on ResNet-56 several seeds diverge, so a=0 has no valid value there. The ELU a=0 runs also diverge on WRN-28-10 (Table 6). The a=1 baselines train stably, and our ReLU baseline reproduces the

reference configuration (Appendix O). We treat the ResNet-56 divergence as a boundary condition of the intervention, not a contradiction of the mechanism; per-seed divergence is in Appendix K.

The control transfers to a different optimizer. The sweeps above use plain SGD. Repeating the a intervention on ResNet-20/CIFAR-100 with the sharpness-aware RSAM (Zhang et al., 2026) reproduces the SGD pattern on the same backbone (Table 6): SELU improves most ($+0.86$ under SGD, $+1.30$ under RSAM), while GELU and ELU change by less than 0.3 points under both optimizers. The effect therefore also appears under RSAM. The optimizer-transfer results are in Appendix H.

Shutting the leak does not improve corruption robustness. On the standard corruption benchmarks (CIFAR-10-C and CIFAR-100-C; Hendrycks & Dietterich, 2019), shutting the leak ($a{=}0$ versus $a{=}1$) does not improve mean corruption accuracy; it is near-neutral on CIFAR-10-C and negative on CIFAR-100-C (per-activation values in Appendix I). The robustness gains in this work come from the input-Jacobian term. In the enriched objective of §5 the two penalties play distinct roles, which we evaluate in §6.5: the leak penalty mainly affects clean accuracy, while the input-Jacobian penalty drives corruption robustness.

Takeaway. Reducing the negative-branch leak can improve clean accuracy, but the best a is setting-dependent: small non-zero values often outperform both the original activation and complete shutoff, while aggressive shutoff can become unstable at depth. We therefore treat a as a setting-specific control rather than a universally optimal value. Because this lever requires only a change to the activation, it is the simpler of the two controls we study; §5 asks whether the same mechanism can instead be controlled during training, without changing the activation.

5 Enriched Input-Space Regularization

§3 linked the negative-branch slope to input sensitivity through the input Jacobian, and §4 showed that this leak can be controlled directly by changing the activation. We next ask whether this negative-branch leak can be controlled *during training*, with the activation held fixed, through a loss that penalizes input sensitivity:

$$L \;=\; L_{\mathrm{CE}} \;+\; \lambda \left\|\partial f/\partial x\right\|_F^2 \;+\; \mu \,\frac{1}{|\mathcal{A}|}\sum_{l\in\mathcal{A}} \mathbb{E}_{z_l<0}\big[\varphi'(z_l)^2\big]. \tag{5}$$

The objective has three components: the cross-entropy classification loss L_{CE}; an input-Jacobian penalty $\|\partial f/\partial x\|_F^2$ (denoted J) weighted by λ, established prior art (Hoffman et al., 2019) that we estimate during training with a single Rademacher projection (one vector-Jacobian product, differentiated through during training); and a leak penalty. We weight the negative-branch leak defined in Definition 1 by μ. This mechanism-derived term is computed analytically from $\varphi'(\cdot)$ with no extra backward pass. The two penalties act on complementary axes: the leak penalty targets the activation-level mechanism of §3, while the input-Jacobian penalty is an established route to reducing input sensitivity; §6 reports the clean-accuracy and robustness roles they play in practice. We select λ and μ with a pre-specified hyperparameter search on a held-out validation split; the search space and selection protocol are given in Appendix N. The enriched-loss training step is written out as pseudocode in Appendix N (Algorithm 1).

Ablation. We isolate the roles of the two penalties with a four-way comparison of objectives:

$$\text{base}: \; L_{\mathrm{CE}}, \qquad +J: \; L_{\mathrm{CE}} + \lambda J, \qquad +\ell: \; L_{\mathrm{CE}} + \mu\,\ell, \qquad +J{+}\ell: \; L_{\mathrm{CE}} + \lambda J + \mu\,\ell. \tag{6}$$

Comparing $+J$ and $+\ell$ against the base model isolates each penalty's effect, and $+J{+}\ell$ tests whether the clean-accuracy and robustness effects are retained together. §6.5 evaluates all four objectives on ResNet-20/CIFAR-100 (Table 3), and §6.4 evaluates the leak penalty across six optimizers (Table 13).

6 Experiments

6.1 Setup

We follow the training protocol of Zhang et al. (2026) for the main CIFAR experiments: SGD with momentum 0.9, learning rate 0.1, weight decay 5×10^{-4}, a warmup-cosine schedule, 200 epochs, and batch size 128 (WRN-28-10 uses 64 to fit memory). We evaluate ResNet-18/20/34/50/56 (He et al., 2016) and WRN-28-10 (Zagoruyko & Komodakis, 2016) on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009); the correlation study (§6.2) additionally includes TinyImageNet-200 (Le &

Yang, 2015) and a pretrained Vision Transformer, ViT-B/16 (Dosovitskiy et al., 2021). Setups for the TinyImageNet, ResNet-50, and ViT runs are in Appendix E. The Kendall analysis of §6.2 varies the seven standard activation functions (ReLU, LeakyReLU, GELU, SiLU, Mish, ELU, SELU), while the controlled gating test of §6.3 adds engineered Softplus variants as intervention instruments. Dataset preprocessing and augmentation, the TinyImageNet optimization setting, the SLQ probe and subset configuration, and our reproduction of the reference base configuration are detailed in Appendices D, E, N, and O.

6.2 Which geometry predicts activation-induced generalization?

Following the correlation protocol of Zhang et al. (2026), we compute the signed Kendall τ (Kendall's rank-correlation coefficient in $[-1, 1]$; $|\tau|$ near 1 means the measure orders models almost exactly like the generalization gap) between each measure and the gap, but where their models come from varying the optimizer, ours come from varying the *activation* (the same trained models are used for every measure). Figure 1 reports the grid across settings. Rényi sharpness, the measure proposed by Zhang et al. (2026), is *less consistent* under activation variation: its sign changes more frequently across architectures and datasets, in contrast to the strong, consistent correlation reported under optimizer variation (Zhang et al., 2026). The mechanism-derived leak and the input Jacobian show stronger and more consistent associations with the gap than Rényi sharpness, although across the broader grid of Table 5 their strength and direction also vary. We make the scoped claim that *which* geometry predicts generalization depends on the axis of variation; we do *not* claim input-space geometry beats weight-space geometry in general. Figure 1 carries two aggregates: its final row reports each measure's average τ across the five settings, and its final column reports each setting's average τ across all measures. The full numerical correlation grid is reported in Appendix F (Table 5).

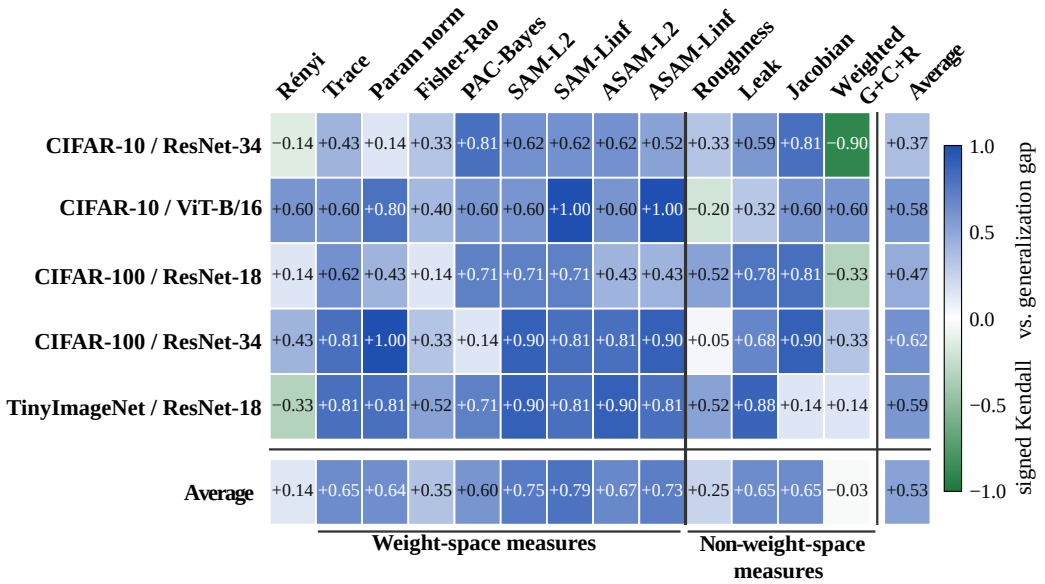


Figure 1: Signed Kendall τ between each geometric measure and the generalization gap under *activation* variation (blue positive, green negative). Left of the divider are weight-space measures (including Rényi sharpness); right are landscape roughness, the input Jacobian, our activation-level negative-branch leak, and the weighted G+C+R composite. Unlike the optimizer-variation setting of Zhang et al. (2026), Rényi sharpness is here weak and sign-inconsistent, while the leak and input Jacobian are often strongly associated with the gap. For Rényi and for SAM/ASAM we plot the variant with the larger $|\tau|$; selection details are in Appendix D. These five settings are those for which all thirteen measures were computed; Table 5 extends eight of them (all but PAC-Bayes and the SAM/ASAM variants) to further settings. The ViT-B/16 row uses n=5 activation conditions, the others n=7.

Table 2: **Leak versus slope-jump control (Softplus instrument, SGD).** ResNet-18/CIFAR-10, three-seed mean±SD. The Softplus shape parameter β and the gate coefficient a both change the negative-branch leak; test accuracy increases as the measured leak decreases even at fixed slope jump (each a; jump $= 0.5(1-a)$ before standardization), and a joint fit attributes the change to the leak, not the jump.

β	a	Leak ($z<0$)	Test (%)
0.5	1.0	0.384	93.17±0.07
0.5	0.5	0.167	94.09±0.19
0.5	0.0	0.000	94.68±0.21
1.0	1.0	0.244	94.06±0.10
1.0	0.5	0.121	94.55±0.08
1.0	0.0	0.000	94.96±0.09
2.0	1.0	0.147	94.49±0.16
2.0	0.5	0.076	94.68±0.07
2.0	0.0	0.000	94.76±0.05

6.3 The negative-branch leak raises input sensitivity: a controlled test

The correlation above is suggestive but observational. To test the mechanism causally, we use the gate of Eq. (4), which changes *only* the activation's negative branch (the positive branch is preserved up to the common output standardization of Appendix C.1): as a closes the negative branch, the measured leak falls, and for the higher-leak activations the input Jacobian also decreases substantially (on CIFAR-100, single seed, ELU from 169.8 to 109.0 and SELU from 224.8 to 108.6 when the branch is shut; Table 8), while clean-accuracy changes depend on the setting (Appendix G, Table 7). Because the intervention targets the negative branch, this provides causal evidence that reducing the negative-branch leak can reduce input sensitivity.

Separating the leak from the slope jump. Closing the branch changes two things at once: it lowers the negative-branch leak and it introduces a slope jump at the origin. To separate them we use a parametric Softplus (Dugas et al., 2000), $\mathrm{softplus}_\beta(z) = \frac{1}{\beta}\log(1+e^{\beta z})$, a smooth approximation to the rectifier whose shape parameter β sets how sharply it bends at the origin and hence the size of the negative-branch leak (larger β gives a smaller leak; Appendix C). On ResNet-18/CIFAR-10 this gives two independent knobs on the leak, the shape parameter β and the gate coefficient a of Eq. (4) (Table 2). Accuracy is strongly negatively associated with the measured leak ($r=-0.96$ across the nine cells). At each fixed a, and hence fixed slope jump, lowering the leak through β still raises accuracy (at $a=1$: $93.17\rightarrow94.06\rightarrow94.49$). A joint fit of accuracy on the leak and the slope jump attributes the effect to the leak ($t=-9.4$); the slope-jump coefficient is small and of the opposite sign to its raw positive correlation ($t=-3.3$). The controlled intervention therefore argues against the slope jump as an explanation for the accuracy change.

The same effect holds for the standard activation functions, not only the Softplus instrument: gating the negative branch of SiLU, Mish, ELU, and SELU toward a rectifier drives the leak to zero and substantially lowers the input Jacobian, while the low-leak GELU is left approximately unchanged (Appendix G, Table 7).

6.4 Leak regularization across optimizers

The leak penalty is optimizer-agnostic, so we add it on top of plain SGD and each sharpness-aware optimizer (GELU throughout, so the leak term is active and the base and $+\ell$ rows differ only by our loss). Adding the leak penalty has its clearest clean-accuracy effect under plain SGD (mean $+0.13$), improving three of the four architecture–dataset pairs; under the sharpness-aware optimizers the effect is small and mixed, with gains on some architecture–optimizer pairs and small drops on others. It is therefore an optimizer-agnostic training penalty whose clean-accuracy benefit is largest for plain SGD rather than a universal improvement; we do not use it as a robustness control, which is the role of the input-Jacobian term (§6.5). The per-optimizer breadth (six optimizers on CIFAR-10 and CIFAR-100, ResNet-20/56) is reported in Appendix J (Table 13).

6.5 Input-Jacobian regularization and the combined objective

While shutting the leak ($a=0$) does not improve corruption accuracy (Appendix I), the input-Jacobian term is the robustness lever. Table 3 is a component ablation on ResNet-20/CIFAR-100

(SGD) that isolates the leak penalty (ℓ) and the input-Jacobian penalty (J) individually against the base model, and evaluates the architectural negative-branch gate (a=0.026) in combination with them. The input-Jacobian term carries the corruption-robustness gain (base 18.9 to 22.1 mCA) and the combined J+ℓ retains it (22.3), while the leak penalty alone is near-neutral on corruption; clean accuracy stays in the range 68.2 to 70.0. Adding the architectural negative-branch gate on top of the Jacobian term further improves corruption robustness in this setting ($+a+J$ reaches 23.3 and $+a+J+\ell$ 23.5 mCA). The same input-Jacobian gain holds on CIFAR-10-C (Appendix I, Table 12). ResNet-18 results across ASAM, *Fisher SAM: Information Geometry and Sharpness Aware Minimisation* (FSAM) (Kim et al., 2022), and RSAM are reported in Appendix J (Table 14), with seed counts given in the caption.

Table 3: **Component ablation on ResNet-20 / CIFAR-100 (SGD, GELU).** Each row adds components to the base model: a is the architectural negative-branch gate (a=0.026), ℓ the leak penalty, and J the input-Jacobian penalty, each at a validated penalty strength. Clean test accuracy and CIFAR-100-C mean corruption accuracy (mCA; 13 corruptions $\times$ 5 severities), three-seed mean$\pm$standard error (SE). Bold marks a gain over base larger than one SE.

Configuration	Clean (%)	CIFAR-100-C mCA (%)
base	$69.70_{\pm 0.13}$	$18.87_{\pm 0.27}$
$+\ell$	$\mathbf{69.97_{\pm 0.20}}$	$18.88_{\pm 0.10}$
$+a+\ell$	$69.23_{\pm 0.25}$	$\mathbf{19.37_{\pm 0.11}}$
$+J$	$69.03_{\pm 0.14}$	$\mathbf{22.12_{\pm 0.23}}$
$+a+J$	$68.34_{\pm 0.30}$	$\mathbf{23.34_{\pm 0.06}}$
$+J+\ell$	$69.12_{\pm 0.08}$	$\mathbf{22.32_{\pm 0.32}}$
$+a+J+\ell$	$68.23_{\pm 0.15}$	$\mathbf{23.54_{\pm 0.29}}$

7 Discussion and Limitations

Discussion. Under activation variation with the optimizer held fixed, the geometric signal associated with the generalization gap differs from the optimizer-varying picture of Zhang et al. (2026): Rényi sharpness becomes less consistent, while the negative-branch leak provides a controllable mechanism that gating connects to input sensitivity. Which geometry predicts generalization therefore depends on which component of the model is varied, not on a single universal predictor; Appendix L gives the fuller interpretation. **Limitations.** Our evidence covers a set of standard activations, convolutional architectures with a ViT setting, and CIFAR-10, CIFAR-100, and TinyImageNet, so we do not claim a universal geometry predictor or a universally optimal leak level. Aggressive negative-branch suppression can destabilize deeper settings such as ResNet-56, which we treat as a setting-dependent stability boundary rather than a general failure. Appendix M details the scope, and Appendix K the stability boundary. **Future work.** Natural next steps are to characterize why the leak penalty interacts differently with different optimizers, to understand the depth-dependent ResNet-56 stability boundary, and to test the mechanism beyond the activations and datasets studied here; these directions are discussed in Appendix M.

8 Conclusion

We studied generalization under activation variation rather than optimizer variation, and found that the geometry associated with the generalization gap changes with this axis, relative to the optimizer-varying results of Zhang et al. (2026): weight-space Rényi sharpness becomes a less consistent predictor, while the activation-level negative-branch leak, causally connected to input sensitivity through controlled gating, tracks the differences. Guided by this mechanism, we control the leak both directly, through a leak-coefficient intervention on the activation, and during training, through an enriched objective that adds the mechanism-derived leak penalty to established input-Jacobian regularization. Reducing the leak through the activation raises clean accuracy by up to 2.45 points (SELU, Table 1), and the input-Jacobian term raises CIFAR-100-C corruption accuracy by 3.25 points over the base model (Table 3). Rather than proposing another universal measure, our results position weight-space, input-space, and activation-level geometry as complementary views whose usefulness depends on the source of model variation.

Reproducibility Statement

The experimental protocol, geometric-measure estimators, hyperparameter search, and full supporting results needed to reproduce our experiments are documented in Appendices D, E, N, and O (exceptions to the base configuration of Zhang et al. (2026) are listed in Table 15), and anonymized source code, the software environment, and run instructions are provided as supplementary material.

LLM Usage Statement

We used a large language model assistant only for grammatical and stylistic editing of the manuscript text. All research ideas, methods, code, experiments, and interpretation of results are the authors' own, and the authors reviewed and take full responsibility for all content.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 2623–2631, 2019. doi: 10.1145/3292500.3330701.

Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 840–902, 2023.

Anna S. Bosman, Andries Engelbrecht, and Mardé Helbig. Empirical loss landscape analysis of neural network activation functions. In *Genetic and Evolutionary Computation Conference Companion (GECCO Companion)*, pp. 2029–2037, 2023. doi: 10.1145/3583133.3596321.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016.

Yann N. Dauphin, Atish Agarwala, and Hossein Mobahi. Neglected hessian component explains mysteries in sharpness regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. arXiv:2401.10809.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1019–1028, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.11929.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *Advances in Neural Information Processing Systems*, 2000.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations (ICLR)*, 2021.

Caglar Gulcehre, Marcin Moczulski, Francesco Visin, and Yoshua Bengio. Mollifying networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Yinglong Guo, Shaohan Li, and Gilad Lerman. The effect of leaky relus on the training and generalization of overparameterized networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238, pp. 4393–4401, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997. doi: 10.1162/neco.1997.9.1.1.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations (ICLR)*, 2020.

Jan W. Kantelhardt, Stephan A. Zschiegner, Eva Koscielny-Bunde, Shlomo Havlin, Armin Bunde, and H. Eugene Stanley. Multifractal detrended fluctuation analysis of nonstationary time series. *Physica A: Statistical Mechanics and its Applications*, 316(1–4):87–114, 2002. doi: 10.1016/S0378-4371(02)01383-3.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.

Minyoung Kim, Da Li, Shell Xu Hu, and Timothy M. Hospedales. Fisher sam: Information geometry and sharpness aware minimisation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 11148–11161, 2022.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 5905–5914, 2021.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. Technical report, Stanford University, 2015.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.

Jingzhou Liu. Nonlinear dynamics in optimization landscape of shallow neural networks with tunable leaky relu, 2025. arXiv:2510.25060.

Andrew Ly and Pulin Gong. Optimization on multifractal loss landscapes explains a diverse range of geometrical and dynamical properties of deep learning. *Nature Communications*, 16, 2025. doi: 10.1038/s41467-025-58532-9.

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

Peng Mi, Li Shen, Tianhe Ren, Yiyi Zhou, Xiaoshuai Sun, Rongrong Ji, and Dacheng Tao. Make sharpness-aware minimization stronger: A sparsified perturbation approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 30950–30962, 2022.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *British Machine Vision Conference (BMVC)*, 2020.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: An empirical study. In *International Conference on Learning Representations (ICLR)*, 2018.

Mohammad Partohaghighi, Roummel F. Marcia, Bruce J. West, and YangQuan Chen. Roughness-informed machine learning: A call for fractal and fractional calculi. *Journal of Information and Intelligence*, 3(6):463–480, 2025. doi: 10.1016/j.jiixd.2025.09.001.

Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. In *Advances in Neural Information Processing Systems*, 2021.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *International Conference on Learning Representations (ICLR), Workshop Track*, 2018.

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.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017. doi: 10.1137/16M1104974.

Dániel Varga, Adrián Csiszárik, and Zsolt Zombori. Gradient regularization improves accuracy of discriminative models. In *International Conference on Learning Representations (ICLR), Workshop Track*, 2018. arXiv:1712.09936.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590, 2020. doi: 10.1109/BigData50022.2020.9378171.

Yunrui Yu, Hang Su, and Jun Zhu. Why the maximum second derivative of activations matters for adversarial robustness, 2026. arXiv:2603.23860.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016. arXiv:1605.07146.

Qiaozhe Zhang, Jun Sun, and Yingzhuang Liu. Rényi sharpness: A novel sharpness that strongly correlates with generalization. In *International Conference on Learning Representations (ICLR)*, 2026.

A Detailed Related Work

This appendix expands the condensed Related Work of §1.1.

Sharpness, flatness, and generalization. The intuition that flat minima generalize better goes back to Hochreiter & Schmidhuber (1997) and was revived at scale by Keskar et al. (2017), who tied the large-batch generalization gap to convergence at sharper minima, and by loss-landscape visualizations that made the flat-versus-sharp distinction concrete (Li et al., 2018). A large-scale study of which quantities predict the generalization gap has found flatness-based measures among the most predictive (Jiang et al., 2020). The intuition is now operational in optimizers that explicitly seek flat regions: Sharpness-Aware Minimization (SAM) minimizes the worst-case loss in a small ball around the weights (Foret et al., 2021), and ASAM makes that ball scale-invariant so the sharpness it targets is not distorted by the network's parameterization (Kwon et al., 2021); we additionally evaluate the Fisher-SAM (FSAM) (Kim et al., 2022) and Sparse-SAM (SSAM) (Mi et al., 2022) variants as sharpness-aware baselines.

A basic obstacle underlies this line of work: flatness is not a well-defined property of the function a network computes. Dinh et al. (2017) show that rescaling one layer up and the next layer down leaves the input-to-output map unchanged yet can make its minimum appear arbitrarily sharp or flat, so any scale-sensitive sharpness can be inflated or deflated by reparameterization alone. Consistent with

this, a large-scale study finds that many classical sharpness measures correlate only weakly, and sometimes with the wrong sign, with generalization once training settings vary (Andriushchenko et al., 2023). Dinh et al. (2017)'s result motivated reparameterization-aware notions of flatness such as relative flatness (Petzka et al., 2021). Related work addresses pathological sharpness in wide networks through normalization (Karakida et al., 2019), and later work clarified which Hessian components drive sharpness regularization (Dauphin et al., 2024). Beyond curvature, the *roughness* of the landscape has been quantified with multifractal detrended fluctuation analysis (Kantelhardt et al., 2002), recently applied to loss landscapes (Ly & Gong, 2025; Partohaghighi et al., 2025), which we include as an additional geometry axis. A recent reparameterization-invariant measure is Rényi sharpness (Zhang et al., 2026), the negative Rényi entropy of the loss-Hessian spectrum. Zhang et al. (2026) show that it is invariant to the layerwise rescalings of Dinh et al. (2017), unlike the trace or maximum eigenvalue. Zhang et al. (2026) propose RSAM, an optimizer that penalizes an approximation of Rényi sharpness, with accuracy gains over SAM and ASAM on CIFAR and TinyImageNet; we include RSAM among our sharpness-aware baselines. Crucially for our study, their evidence varies the *optimizer* while the architecture is held fixed. We take Rényi sharpness as our reference weight-space measure, recompute it on our own trained models with stochastic Lanczos quadrature (Ubaru et al., 2017; Yao et al., 2020) rather than citing their numbers, and ask whether its strong correlation survives when the source of variation is the activation.

Activation functions and landscape geometry. The rectified linear unit (ReLU) and its relatives differ mainly in how they treat negative pre-activations. Softplus (Dugas et al., 2000), the exponential linear unit (ELU) (Clevert et al., 2016), the scaled exponential linear unit (SELU) (Klambauer et al., 2017), the Gaussian error linear unit (GELU) (Hendrycks & Gimpel, 2016), the sigmoid linear unit (SiLU, also called Swish) (Ramachandran et al., 2018), and Mish (Misra, 2020) all pass a nonzero, input-dependent signal on the negative branch, whereas ReLU passes none and its leaky variant (LeakyReLU) (Maas et al., 2013) passes the negative input with a small constant slope. Smoothing the rectifier has long been argued to ease optimization (Gulcehre et al., 2017), and recent work continues to tune the negative branch explicitly (Guo et al., 2024; Liu, 2025); Guo et al. (2024) in particular relate the LeakyReLU slope to generalization, whereas we connect the negative-branch first derivative to input-Jacobian sensitivity and control it either through the gate coefficient or through a training penalty. This negative-branch behavior is exactly the property our analysis isolates. We treat the negative branch as a controlled knob on the leak, not as a proposed activation: our gating scales an existing activation's negative branch to isolate the leak, with plain ReLU as the zero-leak reference. The work most directly related to ours is Bosman et al. (2023), who also treat the activation as a source of loss-landscape variation, but their study differs in scope: it examines three activations (TanH, ReLU, and ELU) in the hidden layer of single-hidden-layer feed-forward networks (the largest a 784-10-10 MNIST network of under 8,000 weights), and characterizes the landscape with fitness-landscape walks through weight space rather than evaluating geometry at trained solutions. We fix the optimizer, vary the activation across trained modern networks (ResNet-18/20/56, WRN-28-10, and ViT-B/16), and ask which geometric quantity tracks the resulting generalization differences.

Input-space sensitivity, Jacobian regularization, and robustness. A parallel line of work characterizes generalization through the *input*-to-output map rather than the weights: lower input sensitivity, measured by the Frobenius norm of the input Jacobian, is associated with better generalization (Novak et al., 2018; Sokolić et al., 2017). Penalizing this Jacobian norm during training to reduce sensitivity is established prior art (Hoffman et al., 2019; Varga et al., 2018); we adopt it unchanged for the Jacobian term of our loss and do not claim it as novel. Our loss-side contribution is instead the mechanism-derived negative-branch leak term, together with the characterization of how the leak and Jacobian terms trade clean accuracy against robustness. We measure robustness with the common-corruption benchmarks CIFAR-10-C and CIFAR-100-C (Hendrycks & Dietterich, 2019). Figure 2 (Appendix B) summarizes how weight-space, input-space, and activation-level geometry sit within the network. Prior work links the activation's second derivative to adversarial robustness (Yu et al., 2026). To our knowledge, no prior work connects the negative-branch *first* derivative specifically to input-Jacobian sensitivity. Controlling that derivative, architecturally or through a training penalty, is the link this paper makes.

B GEOMETRIC BACKGROUND: FROM LOSS LANDSCAPES TO ACTIVATION-LEVEL QUANTITIES

This appendix reviews the geometric objects used in the main text and separates the three levels our analysis touches.

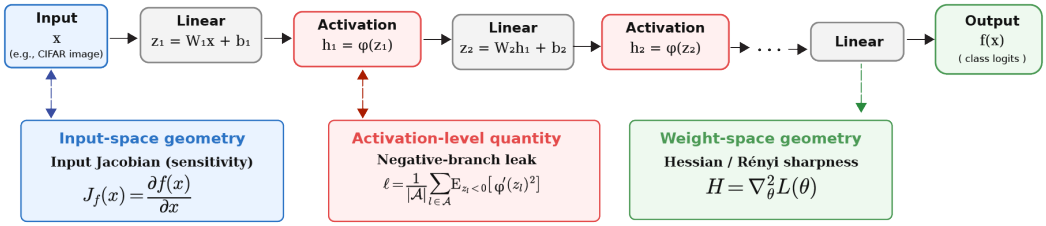


Figure 2: The three geometry levels our analysis relates to the generalization gap, and where each lives in the network. A network maps the input x through affine layers (weights θ) and activations φ to the output $f(x)$. Weight-space curvature (the Hessian and its Rényi sharpness) is a property of the weights θ; the input Jacobian $\|\partial f/\partial x\|_F$ measures sensitivity to the network input x; and the negative-branch leak measures the activation slope $\varphi'(z_l)$ on the negative pre-activations $z_l < 0$. The pre-activation z_l is internal to the network and is distinct from the network input x.

Figure 2 places each geometric quantity at the level it belongs to: weight-space curvature at the parameters θ, input-space sensitivity at the network input x, and the negative-branch leak at the internal pre-activations z_l. The remainder of this section defines each object in turn.

B.1 WEIGHT-SPACE CURVATURE AND THE HESSIAN

For parameters θ, the loss Hessian $H = \nabla^2_\theta L_{\mathrm{CE}}$ describes local curvature of the loss in parameter-space directions; its eigenvalues $\{\lambda_i\}$ are summarized by the trace and by Rényi sharpness (Eq. (1)). Under the Gauss–Newton decomposition, the loss Hessian is

$$H \;=\; \underbrace{J_\theta^\top\big(\nabla^2_f L_{\mathrm{CE}}\big)J_\theta}_{G\ \text{(from slopes }\varphi')} \;+\; \underbrace{R}_{\text{(contains all }\varphi''\text{ terms)}}, \tag{7}$$

where $J_\theta = \partial f/\partial\theta$ is the parameter Jacobian, distinct from the input Jacobian $\partial f/\partial x$ of Eq. (2), and $\nabla^2_f L_{\mathrm{CE}}$ is the loss Hessian in output space. The Gauss–Newton term G carries the activation slope φ', while the residual R collects the parameter-wise second derivatives, including every term in the activation curvature φ''.

B.2 ACTIVATION DERIVATIVES AND CURVATURE

For a scalar activation φ, the slope φ' and the curvature φ'' are properties of the function shape. *Activation curvature* φ'' is distinct from *weight-space curvature* H: the former is a property of the activation, the latter of the trained loss surface. They are related through the residual term R in the Gauss–Newton decomposition (Eq. (7)) but are not the same object. Figure 3 plots φ, φ', and φ'' for the seven activations.

B.3 DISTINGUISHING THE GEOMETRIC LEVELS

Table 4 summarizes the objects and the level each belongs to.

C ACTIVATION FUNCTIONS AND NEGATIVE-BRANCH GATING

The negative-branch leak of Def. 1 is a property of the activation's shape, so we plot the functions directly. Figure 3 shows $\varphi(z)$, its first derivative $\varphi'(z)$, and its second derivative $\varphi''(z)$ for the seven activations we study, together with sigmoid and tanh for visual context (these two are not part of the quantitative activation set). The derivative panel is the one that connects to the leak $\ell = \frac{1}{|\mathcal{A}|}\sum_l \mathbb{E}_{z_l<0}[\varphi'(z_l)^2]$: ReLU is flat ($\varphi' = 0$) on the negative branch, LeakyReLU carries a small constant slope, and the smooth activations carry an input-dependent slope whose magnitude grows from GELU/SiLU/Mish through ELU to SELU. This ordering matches the measured leak

Table 4: The geometric objects and the level each belongs to.

Object	Meaning	Space / level
$H = \nabla^2_\theta L_{\mathrm{CE}}$	loss curvature w.r.t. parameters	weight space (θ)
$\partial f / \partial x$	network sensitivity to input	input space (x)
z_l	input to the activation at layer l	activation level (internal)
$\varphi'(z_l)$	activation slope	activation level
$\varphi''(z_l)$	activation curvature	activation level
ℓ	mean squared negative-branch slope	activation level
a	coefficient controlling the negative branch	activation intervention

values in Table 7 (SELU largest, GELU smallest), so the negative branches differ substantially rather than being smoothed copies of ReLU.

Figure 4 illustrates the gating intervention of Eq. (4) on GELU as a representative activation. As the coefficient a falls from 1 to 0, the negative branch flattens to the zero-leak rectifier while the positive branch is left unchanged (before output standardization); the corresponding derivative panel shows a scaling the negative-branch slope $\varphi'_a(z) = a\,\varphi'(z)$ toward zero, together with the slope jump at the origin, $\varphi'(0)(1-a) = 0.5(1-a)$ for GELU, that the mechanism analysis separates from the leak.

Softplus instrument. The leak-versus-slope-jump control of §6.3 uses a parametric Softplus, $\mathrm{softplus}_\beta(z) = \frac{1}{\beta}\log\big(1 + e^{\beta z}\big)$, with value $\mathrm{softplus}_\beta(0) = \frac{1}{\beta}\log 2$ at the origin. The shape parameter β controls the negative-branch leak (and also reshapes the positive branch near the origin): larger β sharpens the function toward the rectifier, so the leak decreases as β grows. Gating with the coefficient a of Eq. (4) scales the negative branch relative to this origin value, giving a second, independent route to reduce the leak. Varying β and a separately is what distinguishes the leak from the slope jump at the origin in Table 2.

C.1 Output standardization

In the networks we train from scratch (the ResNets and WRN-28-10), every activation is wrapped so that the size of its outputs is fixed, which removes differences in activation-output size as a confound when we compare activations or vary the gate coefficient a. The pretrained ViT-B/16 is used without this output standardization. Let g be the scalar activation to be wrapped (the raw activation φ, or, under the intervention, the gated activation g_a of Eq. (4)). The standardized activation is the fixed affine map

$$\tilde{g}(z) \;=\; \frac{g(z) - \mu_g}{\sigma_g}, \qquad \mu_g = \mathbb{E}_{Z\sim\mathcal{N}(0,1)}\big[g(Z)\big], \quad \sigma_g = \sqrt{\mathrm{Var}_{Z\sim\mathcal{N}(0,1)}\big[g(Z)\big]}, \tag{8}$$

so that $\tilde{g}(Z)$ has mean 0 and variance 1 when the pre-activation Z is standard normal. Three points make the operation precise. **(i) What is standardized.** The first two moments (mean and standard deviation) of the activation *output*, so the standardized output has unit variance and zero mean under a standard-normal pre-activation. **(ii) Over what and when.** The moments are analytic expectations over $Z \sim \mathcal{N}(0,1)$, not empirical statistics of the data; they are computed once at build time by 200-node Gauss–Hermite quadrature (deterministic; accurate to about 10^{-3} for activations with a slope jump at zero, and to $\sim 10^{-8}$ for smooth activations) and stored as fixed buffers. They are not learnable and are not running batch statistics, so μ_g, σ_g do not change during training, and standardization is applied elementwise at every activation layer. **(iii) Order relative to gating.** Standardization is applied *after* gating: the network uses $\tilde{g}_a(z) = (g_a(z) - \mu_{g_a})/\sigma_{g_a}$, and the moments μ_{g_a}, σ_{g_a} are recomputed for each a. The gate itself leaves the positive branch of the base activation exactly unchanged ($g_a(z) = \varphi(z)$ for $z \geq 0$; Eq. (4)). The subsequent standardization then applies a single common affine rescaling to the whole function, so across different a the positive branch is preserved up to this shared, a-dependent rescaling that holds the output moments at mean 0 and variance 1. This rescaling multiplies the positive-branch slope by $\sigma_{\varphi_1}/\sigma_{\varphi_a} \geq 1$, ranging at a=0 from 1.07 (GELU) through 1.14 (SiLU, Mish) and 1.35 (ELU) to 1.63 (SELU), and 1.00 for LeakyReLU. The negative-branch leak is therefore the only property deliberately varied while the output size is matched across a. Consistent with the standard-normal assumption, the leak measured on the trained networks matches its analytic $\mathcal{N}(0,1)$ value within about 0.01 (at most 3%) for every activation (Table 7).

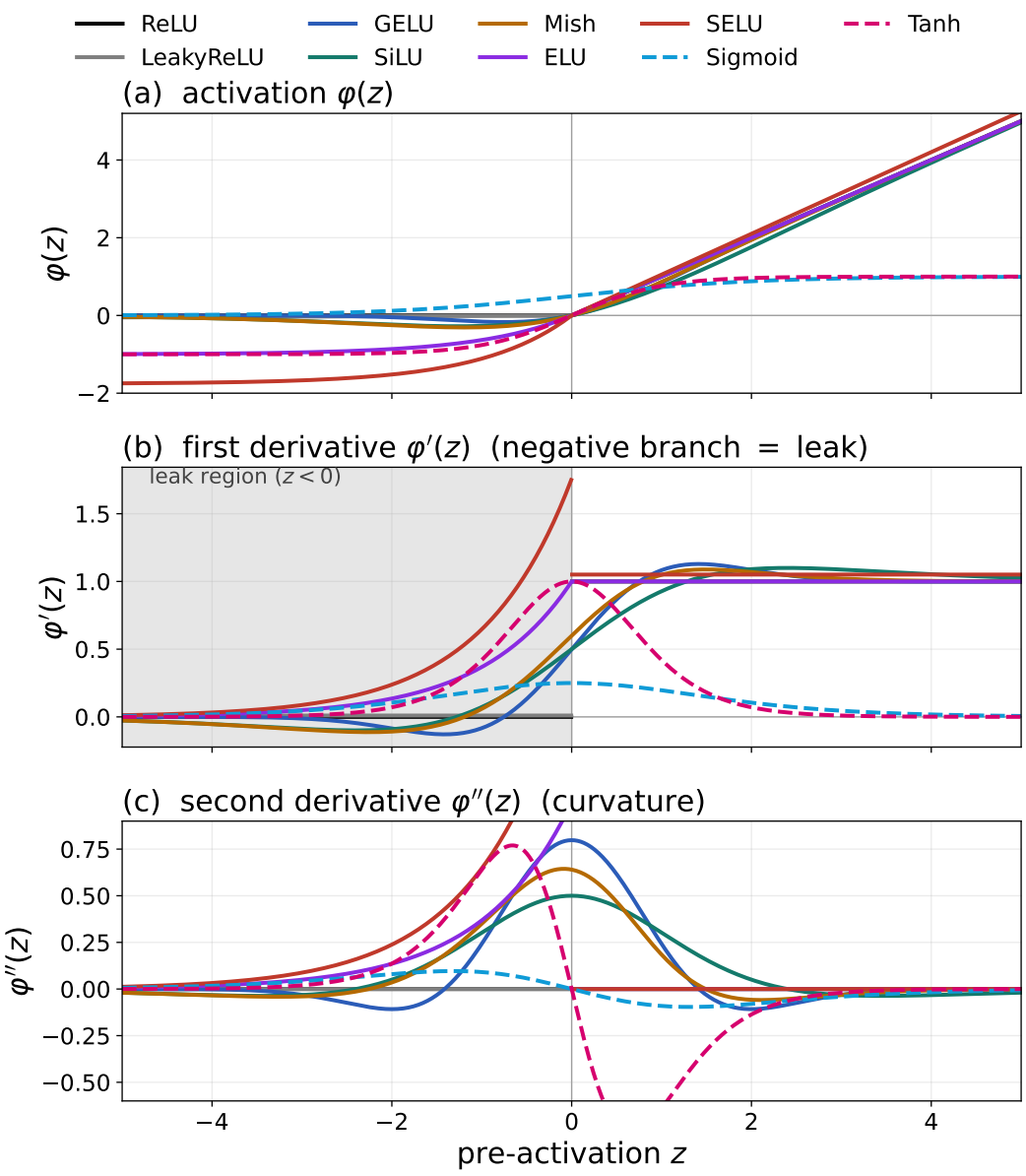


Figure 3: The seven activations studied, plus sigmoid and tanh (dashed) for visual context; sigmoid and tanh are not part of the quantitative activation set (n=7). **(a)** $\varphi(z)$. **(b)** $\varphi'(z)$, with the negative branch ($z < 0$) shaded; this is where the leak is measured. **(c)** $\varphi''(z)$, the activation curvature. The line is broken at z=0 for activations whose first or second derivative is discontinuous there (ReLU, LeakyReLU, SELU: first derivative; ELU: second derivative). Curves show the exact library activations before output standardization (Appendix C.1), with the first derivative taken by automatic differentiation and the second derivative from it. ReLU and LeakyReLU have zero and constant negative-branch slope respectively, while the smooth activations carry an input-dependent slope (largest for SELU, consistent with its largest measured leak in Table 7).

D Geometric Measures and Estimation Details

This section records the estimators used for the geometry comparison in Figure 1. All measures are evaluated on the same trained checkpoints for a given dataset and architecture. The correlation analysis uses signed Kendall's τ across the standard activation functions, with $n = 7$ activations unless marked otherwise.

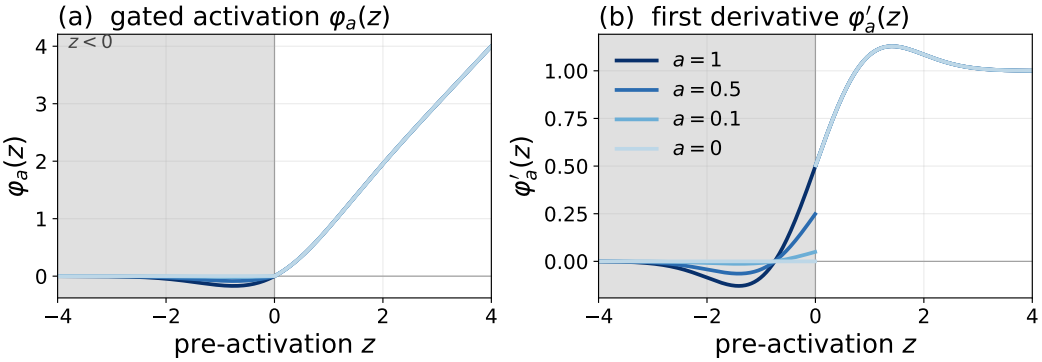


Figure 4: The gate of Eq. (4) on GELU at $a \in \{1, 0.5, 0.1, 0\}$. **(a)** $\varphi_a(z)$ and **(b)** $\varphi'_a(z)$: lowering a scales the negative branch (shaded, $z < 0$) toward the zero-leak rectifier while leaving the positive branch unchanged (before output standardization).

D.1 Weight-space measures

The Hessian trace and Rényi sharpness are both estimated from the Hessian spectrum with stochastic Lanczos quadrature (SLQ) (Ubaru et al., 2017), using 50 Lanczos draws of 15 steps each on a fixed 2000-example training subset (Hessian batch size 128, augmentation off). A Lanczos draw that fails the eigendecomposition step is skipped rather than used as a numerical value. The comparison also includes the parameter norm $\|\theta\|_2^2$, the Fisher–Rao norm $(\text{depth}+1)^2\,\mathbb{E}_x[(\theta^\top \nabla_\theta L_{\text{CE}})^2]$, and PAC-Bayes sharpness $\|\theta\|_2^2/(4\sigma^2)$ with σ from a magnitude search (all three following the catalogue of Jiang et al., 2020), and the loss increase under SAM and ASAM perturbations. The ℓ_2 and ℓ_∞ variants differ in the perturbation ball used for the one-step worst-case ascent: the ℓ_2 step is $\epsilon = \rho\,\nabla L_{\text{CE}}/\|\nabla L_{\text{CE}}\|_2$ and the ℓ_∞ step is $\epsilon = \rho\,\text{sign}(\nabla L_{\text{CE}})$, where ρ is the radius; ASAM uses the scale-invariant forms $\epsilon = \rho\,T_\theta^2\,\nabla L_{\text{CE}}/\|T_\theta \nabla L_{\text{CE}}\|_2$ and $\epsilon = \rho\,|\theta|\,\text{sign}(\nabla L_{\text{CE}})$ with $T_\theta = |\theta|$. For a given radius, the measure is the resulting loss increase $L_{\text{CE}}(\theta+\epsilon) - L_{\text{CE}}(\theta)$.

The radius ρ ranges over the thirteen-point grid from 10^{-6} to 1.0 used by Zhang et al. (2026). Because the SAM/ASAM radius and the Rényi order $\alpha \in \{0.5, 1.5\}$ are free settings of these baseline measures, we report each of them at the value (radius, or α) that gives the strongest rank correlation $|\tau|$ with the generalization gap in each setting. This selection favors the weight-space baselines, since our leak and input-Jacobian measures have no such free parameter to tune; even so, the leak tracks the gap more consistently than the best-tuned Rényi sharpness.

D.2 Input-space and activation-level measures

For the geometry comparison, the input-Jacobian quantity in Eq. (2) is computed exactly as the Frobenius norm of the logit-input Jacobian for each input (the square root of the summed squared gradients over all output logits and input dimensions), averaged over 400 held-out inputs (training instead uses a single Rademacher probe per step, §5). The negative-branch leak in Eq. (3) uses the analytic first derivative of each of the seven activations (and the Softplus instrument) and averages the squared derivative over negative pre-activations on 200 held-out inputs, with equal weighting across activation layers.

D.3 Landscape roughness and the composite measure

Landscape roughness is the generalized Hurst exponent $h(2)$ of loss profiles sampled along 6 random rays from the trained solution (512 points per ray, radius 0.4, loss evaluated on 1000 training examples), estimated by order-1 multifractal detrended fluctuation analysis over 18 scales with $q \in \{-5, -3, -2, -1, 2, 3, 5\}$; a lower $h(2)$ indicates a rougher landscape. The weighted G+C+R quantity combines the loss-gradient norm (G), Rényi sharpness at α=0.5 (C), and the roughness $h(2)$ (R) by ordinary least squares. Its weights are fit with leave-one-out cross-validation and the reported τ is computed over the held-out predictions, because the correlation set contains only a small number of activation conditions.

Table 5: **Full activation-induced geometry correlation grid.** Signed Kendall's τ between each measure and the last-epoch generalization gap. The standard convolutional-network rows use seven standard activation functions ($n = 7$); rows with a different activation count are marked. Rényi is the weight-space measure of Zhang et al. (2026). G+C+R is the leave-one-out-cross-validated weighted gradient, curvature, and roughness composite.

Setting	Rényi	Trace	Param.	Fisher–Rao	Roughness	Leak	Jacobian	G+C+R
CIFAR-10 / ResNet-34	-0.14	+0.43	+0.14	+0.33	+0.33	+0.59	+0.81	-0.90
CIFAR-10 / ViT-B/16 ($n = 5$)	+0.60	+0.60	+0.80	+0.40	-0.20	+0.32	+0.60	+0.60
CIFAR-100 / ResNet-18	+0.14	+0.62	+0.43	+0.14	+0.52	+0.78	+0.81	-0.33
CIFAR-100 / ResNet-34	+0.43	+0.81	+1.00	+0.33	+0.05	+0.68	+0.90	+0.33
TinyImageNet / ResNet-18	-0.33	+0.81	+0.81	+0.52	+0.52	+0.88	+0.14	+0.14
CIFAR-10 / ResNet-20	+0.52	-0.43	+0.05	-0.24	-0.33	-0.10	-0.43	+0.52
CIFAR-10 / ResNet-50 ($n = 5$)	+0.40	+1.00	+0.80	+0.60	+0.80	+0.32	+0.80	+0.40
CIFAR-10 / ResNet-56	-0.14	+0.90	+0.62	+0.52	-0.24	+0.68	+0.33	+0.24
CIFAR-10 / WRN-28-10	-0.33	+0.81	+0.81	+0.90	+0.71	+0.78	+1.00	+0.71
CIFAR-100 / ResNet-20	-0.33	-0.33	-0.14	-0.43	-0.71	-0.39	-0.14	+0.90
CIFAR-100 / ResNet-56	+0.24	-0.14	+0.62	-0.33	+0.52	-0.49	-0.71	+0.43

E Experimental Protocol and Additional Settings

The main experiments use CIFAR-10 and CIFAR-100, with TinyImageNet included in the geometry-correlation breadth. Architectures include ResNet-18, ResNet-20, ResNet-34, ResNet-56, and WRN-28-10 in the reported experiments. The base stochastic-gradient training configuration follows that of Zhang et al. (2026): momentum 0.9, initial learning rate 0.1, weight decay 5×10^{-4}, a 10-epoch linear warm-up followed by cosine decay, 200 epochs, and batch size 128. WRN-28-10 uses batch size 64. The ResNet-18 gating runs use weight decay 1×10^{-4}, as recorded in Table 15.

Performance tables report best-validation checkpoints. The geometry/generalization correlation uses the last-epoch train-minus-test generalization gap, matching the protocol of Zhang et al. (2026). Gate coefficients and loss strengths are selected using held-out validation data; test accuracy is used only for the final reported evaluation. CIFAR-10-C and CIFAR-100-C robustness is reported as mean corruption accuracy over 13 corruptions and five severities for the checkpoints evaluated in the robustness experiments.

Additional architectures and datasets. TinyImageNet-200 uses ResNet-18 at 64×64 resolution with the TinyImageNet recipe of Zhang et al. (2026) (learning rate 0.2, weight decay 1×10^{-4}, batch size 128, 200 epochs). ResNet-50 uses the same CIFAR recipe as the other from-scratch ResNets (learning rate 0.1, weight decay 5×10^{-4}, batch size 128). The Vision Transformer is a torchvision ViT-B/16 pretrained on ImageNet-1K and fine-tuned for 20 epochs (learning rate 0.01, weight decay 5×10^{-4}, batch size 128) on CIFAR images upsampled to 224×224 with CIFAR normalization, following Zhang et al. (2026). For the cross-activation correlation rows, the encoder-MLP GELU is replaced by each candidate activation, used without standardization (§3). The ViT-B/16 and CIFAR-10/ResNet-50 correlation rows use n=5 activations (ReLU, LeakyReLU, GELU, SiLU, Mish), the others n=7 (adding ELU and SELU).

F Full Geometry-Correlation Results

Figure 1 presents the compact main-paper comparison. Table 5 gives the broader signed Kendall's τ results across the available dataset and architecture settings. Each coefficient ranks a geometric measure against the generalization gap across activation conditions. Positive and negative signs therefore encode direction, not merely correlation strength.

The broader grid reinforces the setting dependence visible in the main figure. No measure has a fixed sign or uniformly large magnitude across every row. The activation-level leak and input Jacobian are strong in several settings, while some ResNet-20 and ResNet-56 settings produce weak or reversed relationships for multiple measures. This is why the main claim is scoped to activation-induced variation in the evaluated settings rather than stated as a universal ordering of geometric measures.

G Full Activation-Intervention Results

Full gated-family and cross-architecture gating tables. The leak-versus-slope-jump control that separates the negative-branch leak from the slope jump at the origin is promoted to the main text (Table 2, §6.3). Table 6 gives the cross-architecture setting dependence summarized in §4.

Table 6: **Setting dependence of negative-branch gating (CIFAR-100, SGD).** Clean test accuracy (mean±SD, three seeds) at a=0 (shut) versus a=1 (original) on ResNet-20 and WRN-28-10; Δ = a=0 minus a=1, bold↑ marks a=0 above a=1. "diverged" marks an unstable a=0 run, for which Δ is undefined. ReLU is the zero-leak reference.

Arch	Activation	Test acc. (%) a=0	Test acc. (%) a=1	Δ (%)
ResNet-20	GELU	$69.19_{\pm 0.14}$	$69.47_{\pm 0.25}$	−0.28
ResNet-20	ELU	$69.43_{\pm 0.14}$	$69.46_{\pm 0.09}$	−0.02
ResNet-20	SELU	$\mathbf{69.05_{\pm 0.46}}$ ↑	$68.20_{\pm 0.26}$	**+0.86**↑
ResNet-20	ReLU (ref)	$69.20_{\pm 0.36}$	n/a	n/a
WRN-28-10	GELU	$\mathbf{81.04_{\pm 0.31}}$ ↑	$78.70_{\pm 0.27}$	**+2.34**↑
WRN-28-10	ELU	diverged	$72.75_{\pm 0.43}$	–
WRN-28-10	SELU	$\mathbf{81.17_{\pm 0.11}}$ ↑	$70.32_{\pm 0.39}$	**+10.85**↑
WRN-28-10	ReLU (ref)	$81.32_{\pm 0.06}$	n/a	n/a

Interior values do not change the picture here. A small non-zero leak helps on ResNet-18/CIFAR-10 (Appendix G.1), but on the CIFAR-100 settings in this table the interior gives no gain over the endpoints. For ResNet-20/GELU the coarse test sweep places the optimum at a=0.001 (test accuracy 69.23), within seed noise of a=0 (69.19) and a=1 (69.47); the validation search instead selects a=0.026, the value used in the component ablation (Table 3) and the RSAM transfer (Table 10). The validation-based per-setting search (Appendix N), run for GELU across configurations, finds the interior a-curve flat on the other two backbones for CIFAR-100: ResNet-20 stays near 69.0 validation accuracy across the searched range, and WRN-28-10 stays within 79.7 to 80.8 for every a. We swept the interior for GELU on these backbones; for ELU and SELU we report the endpoints. The interior optimum is therefore setting-dependent, pronounced on ResNet-18/CIFAR-10 and flat on these other CIFAR-100 configurations.

Standard-activation gating. Beyond the Softplus control in the main text, we apply the gate of Eq. (4) to the standard activations on ResNet-18/CIFAR-10 (Table 7). Closing the negative branch drives the measured leak to zero. The input Jacobian decreases substantially for SiLU, Mish, ELU, and SELU, while GELU is approximately unchanged (35.8 to 35.9), and clean accuracy is preserved or slightly improved. These results support the same mechanism in the standard activation families while showing that the magnitude of the response is activation-dependent.

G.1 Interior leak-coefficient sweep

The endpoint comparison does not imply that $a = 0$ is optimal. A finer GELU sweep on ResNet-18/CIFAR-10 shows a broad small-leak regime, with the replicated $a = 0.003$ point above both endpoints.

G.2 Architectural control and leak loss do not simply stack

On ResNet-18/CIFAR-100 with GELU and SGD, combining the validation-selected architectural coefficient $a = 0.013$ with the leak penalty (μ=0.115) gives 76.00 ± 0.10 clean test accuracy over three seeds. This is below the architectural intervention alone (76.86), the $a = 1$ baseline (76.81), and complete shutoff (76.66). The two controls therefore do not show an additive benefit in this matched setting.

H Optimizer Transfer of the Activation Intervention

To test whether the negative-branch control is specific to plain SGD, we repeat the leak-gate protocol of Table 1 on ResNet-20/CIFAR-100 with RSAM (Zhang et al., 2026) in place of SGD, keeping the base configuration otherwise identical (three seeds; best interior a=0.026 from the same validation search). Table 10 shows the same qualitative pattern as the SGD results on the same backbone

Table 7: Gating the standard activations on ResNet-18/CIFAR-10 (SGD; clean accuracy 3-seed mean±SD). As the negative-branch coefficient a is reduced from 1 (base activation) to 0 (shut branch), the leak falls to zero and the input Jacobian decreases substantially for SiLU, Mish, ELU, and SELU while GELU is approximately unchanged; clean accuracy is preserved or improved. Input Jacobian values are the 3-seed mean±SD. Mean corruption accuracy on CIFAR-10-C (Hendrycks & Dietterich, 2019) (mCA, single seed) is approximately unchanged by shutting the leak, consistent with corruption robustness coming from the input-Jacobian term rather than the leak (§6.5). The *best* rows report clean accuracy at the best interior coefficient a located by a test-accuracy sweep (three seeds for GELU, single seed otherwise); the interior beats both endpoints for four of the five activations (all but Mish, whose best interior value stays just below its shut-branch endpoint). Dashes mark quantities not measured at that setting. The Jacobian trend mirrors CIFAR-100 (Table 8).

Activation	a	Leak ($z<0$)	Input Jacobian	Clean (%)	CIFAR-10-C mCA (%)
GELU	1.0	0.131	35.8±0.7	94.72±0.23	43.84
	0.0	0.000	35.9±0.3	94.83±0.10	43.69
	best (0.003)	$< 10^{-4}$	–	95.06±0.07	–
SiLU	1.0	0.213	43.7±1.0	94.52±0.10	43.85
	0.0	0.000	37.7±0.8	94.92±0.05	43.67
	best (0.01)	$< 10^{-4}$	–	95.10	–
Mish	1.0	0.226	43.9±0.5	94.54±0.12	43.72
	0.0	0.000	37.6±0.2	94.96±0.10	43.57
	best (0.003)	$< 10^{-4}$	–	94.89	–
ELU	1.0	0.538	62.6±0.8	93.74±0.14	43.07
	0.0	0.000	42.8±0.8	94.71±0.16	42.59
	best (0.01)	$< 10^{-4}$	–	95.03	–
SELU	1.0	1.029	73.1±0.4	93.53±0.12	42.51
	0.0	0.000	42.6±0.3	94.86±0.13	43.23
	best (0.003)	$< 10^{-4}$	–	94.88	–

Table 8: **Input Jacobian under negative-branch gating (ResNet-18/CIFAR-100, SGD).** Frobenius norm of the input-to-output Jacobian $\|\partial f/\partial x\|_F$ (single seed) at a=1 (original), a=0 (shut), and the best interior a. Reducing the leak lowers the input Jacobian most for the high-leak activations (ELU, SELU) and leaves the low-leak ones roughly unchanged, mirroring the CIFAR-10 trend (Table 7) and motivating the input-Jacobian penalty of §5.

Activation	Jac a=1	Jac a=0	Jac best-a
GELU	93.1	93.6	85.9
SiLU	91.3	83.7	89.7
Mish	90.3	87.4	92.7
ELU	169.8	109.0	110.2
SELU	224.8	108.6	109.8
ReLU (ref)		70.6	

(Table 6): SELU improves most under both optimizers (+0.86 under SGD, +1.30 under RSAM), while GELU and ELU change by less than 0.3 points under both, so the control transfers across optimizers.

I Input Sensitivity and Corruption Robustness

Mean corruption accuracy (mCA) of the ResNet-18 leak sweep of Table 1, evaluated on CIFAR-10-C and CIFAR-100-C (13 of the 15 standard corruptions, omitting glass blur and fog, × 5 severities, evaluation only on the trained a=0/a=1 checkpoints). These are the numbers summarized in §4: shutting the leak is near-neutral on CIFAR-10-C and negative on CIFAR-100-C, so the leak is not the corruption-robustness lever. Table 11 compares the maximal leak-reduction intervention (a=0) with the original activation (a=1). This endpoint comparison tests whether complete negative-branch suppression improves corruption robustness; it does not characterize the interior best-a settings, whose corruption behavior we do not evaluate here. Δ = a=0 minus a=1; bold with ↑ marks a case where shutting the leak improves mCA.

Table 9: **Replicated points from the GELU leak-coefficient sweep.** ResNet-18/CIFAR-10, SGD. Clean test accuracy is mean±SD over three seeds for the replicated points. The result supports a small-leak regime rather than a universal optimum at complete shutoff.

a	Clean accuracy (%)	Generalization gap (%)
0	94.83 ± 0.10	5.17
0.001	94.96 ± 0.19	5.04
0.003	$\mathbf{95.06 \pm 0.07}$	**4.94**
0.5	94.86 ± 0.11	–
1	94.72 ± 0.23	5.28

Table 10: **Leak control transfers to RSAM.** Clean test accuracy (%, mean±SD, n=3) on ResNet-20/CIFAR-100 trained with RSAM, at a=0 (shut), the best interior a=0.026, and a=1 (original); $\Delta = a$=0 minus a=1, bold↑ marks a=0 above a=1. ReLU is the zero-leak reference.

Activation	a=0	a=0.026	a=1	Δ
GELU	$69.48_{\pm 0.34}$	$69.25_{\pm 0.10}$	$69.44_{\pm 0.67}$	$+0.04$
SiLU	$69.02_{\pm 0.15}$	$69.09_{\pm 0.48}$	$69.26_{\pm 0.44}$	-0.24
Mish	$\mathbf{69.84_{\pm 0.15}}$↑	$69.15_{\pm 0.44}$	$69.17_{\pm 0.33}$	**+0.67**↑
ELU	$69.26_{\pm 0.29}$	$69.28_{\pm 0.17}$	$69.35_{\pm 0.36}$	-0.09
SELU	$\mathbf{69.11_{\pm 0.38}}$↑	$69.15_{\pm 0.18}$	$67.81_{\pm 0.05}$	**+1.30**↑
LeakyReLU	$69.06_{\pm 0.31}$	$69.25_{\pm 0.56}$	$69.23_{\pm 0.48}$	-0.17
ReLU (ref)	$69.25_{\pm 0.71}$	n/a	n/a	n/a

Table 11: ResNet-18 leak sweep, mean corruption accuracy (mCA, %). Left: CIFAR-10-C. Right: CIFAR-100-C. Seed-0 checkpoints. ReLU is the zero-leak reference (no a=0).

	CIFAR-10-C			CIFAR-100-C		
Activation	a=0	a=1	Δ	a=0	a=1	Δ
GELU	43.69	43.84	-0.15	19.45	22.41	-2.96
SiLU	43.67	43.85	-0.18	17.67	22.61	-4.95
Mish	43.57	43.72	-0.15	19.15	22.43	-3.28
ELU	42.59	43.07	-0.49	**21.80**↑	21.62	**+0.19**↑
SELU	**43.23**↑	42.51	**+0.72**↑	**22.14**↑	21.24	**+0.90**↑
LeakyReLU	42.93	44.06	-1.12	22.22	22.97	-0.75
Average (6 activations)	43.28	43.51	-0.23	20.41	22.21	-1.81
ReLU (ref)	–	43.92	n/a	–	22.60	n/a

The input-Jacobian penalty is the corruption-robustness lever on both datasets. Table 12 reports the cross-dataset ResNet-20/GELU result under plain SGD: adding the Jacobian penalty ($+J$) raises mean corruption accuracy by about $+6$ points on CIFAR-10-C and $+3$ on CIFAR-100-C over the base model, and the combined Jacobian+leak ($+J+\ell$) retains that improvement.

Table 12: **Input-Jacobian regularization improves corruption robustness, and Jacobian+leak keeps it.** ResNet-20 / GELU under plain SGD, mean corruption accuracy (mCA, %; 13 corruptions $\times$ 5 severities) on CIFAR-10-C and CIFAR-100-C. base = SGD alone; $+J$ = base + input-Jacobian penalty; $+J+\ell$ = base + Jacobian+leak. Three-seed mean±SE; bold marks a gain over base.

Setting (ResNet-20)	CIFAR-10-C mCA	CIFAR-100-C mCA
SGD, base	$40.84_{\pm 0.59}$	$18.87_{\pm 0.27}$
SGD, $+J$	$\mathbf{46.86_{\pm 0.14}}$	$\mathbf{22.12_{\pm 0.23}}$
SGD, $+J+\ell$	$\mathbf{47.49_{\pm 0.14}}$	$\mathbf{22.32_{\pm 0.32}}$

J Training Objectives and Optimizer Interaction

This section gives the breadth results behind the compact training-objective tables in the main paper. Table 13 reports the leak-only penalty ($+\ell$) across six optimizers, and Table 14 reports the available optimizer breadth for corruption robustness, with the seed counts given in its caption. These breadth results are supporting evidence; the main text uses matched, fully replicated cells for its headline comparisons.

Table 13: **The leak term added on top of every optimizer.** Clean test accuracy (%) on CIFAR-10 and CIFAR-100; each optimizer has a base row and a $+\ell$ (leak) row, GELU throughout. Three-seed mean±SE. FSAM/SSAM are from Kim et al. (2022); Mi et al. (2022). Two-seed cells: ASAM and SSAM $+\ell$ on ResNet-56/CIFAR-100 and FSAM $+\ell$ on ResNet-20 and ResNet-56/CIFAR-100; all other cells use three seeds. The right column mean $\Delta\ell$ is the average leak effect ($+\ell$ minus base); corruption robustness is in §6.5. Bold marks $+\ell$ at least tied with its base: mean above base (↑), or a shortfall within one standard error (the larger of the two cells). Average rows have no standard error, so bold there marks a mean above base.

		CIFAR-10		CIFAR-100		
Optimizer		ResNet-20	ResNet-56	ResNet-20	ResNet-56	$\Delta\ell$
SGD	base	$92.99_{\pm 0.08}$	$93.48_{\pm 0.10}$	$69.70_{\pm 0.13}$	$72.69_{\pm 0.07}$	
	$+\ell$	$\mathbf{93.13_{\pm 0.02}}$↑	$\mathbf{93.84_{\pm 0.11}}$↑	$\mathbf{69.97_{\pm 0.20}}$↑	$72.44_{\pm 0.09}$	$+0.13$
SAM	base	$93.73_{\pm 0.07}$	$94.20_{\pm 0.13}$	$70.73_{\pm 0.11}$	$73.26_{\pm 0.10}$	
	$+\ell$	$93.60_{\pm 0.11}$	$\mathbf{94.25_{\pm 0.15}}$↑	$\mathbf{71.10_{\pm 0.13}}$↑	$73.14_{\pm 0.01}$	$+0.04$
ASAM	base	$93.74_{\pm 0.07}$	$94.69_{\pm 0.07}$	$71.42_{\pm 0.16}$	$73.92_{\pm 0.20}$	
	$+\ell$	$\mathbf{93.77_{\pm 0.03}}$↑	$\mathbf{94.55_{\pm 0.14}}$	$\mathbf{71.40_{\pm 0.05}}$	$\mathbf{74.23_{\pm 0.10}}$↑	$+0.04$
FSAM	base	$93.64_{\pm 0.03}$	$94.34_{\pm 0.09}$	$71.14_{\pm 0.13}$	$73.92_{\pm 0.17}$	
	$+\ell$	$\mathbf{93.78_{\pm 0.07}}$↑	$94.23_{\pm 0.03}$	$\mathbf{71.45_{\pm 0.06}}$↑	$\mathbf{73.81_{\pm 0.09}}$	$+0.06$
SSAM	base	$93.74_{\pm 0.12}$	$94.36_{\pm 0.15}$	$70.36_{\pm 0.01}$	$73.26_{\pm 0.10}$	
	$+\ell$	$93.54_{\pm 0.08}$	$\mathbf{94.33_{\pm 0.20}}$	$\mathbf{70.78_{\pm 0.10}}$↑	$72.97_{\pm 0.22}$	-0.03
RSAM	base	$93.18_{\pm 0.06}$	$93.60_{\pm 0.02}$	$69.72_{\pm 0.12}$	$72.16_{\pm 0.14}$	
	$+\ell$	$\mathbf{93.12_{\pm 0.14}}$	$\mathbf{93.76_{\pm 0.16}}$↑	$\mathbf{69.62_{\pm 0.17}}$	$\mathbf{72.27_{\pm 0.03}}$↑	$+0.03$
Average	base	93.50	94.11	70.51	73.20	
	$+\ell$	93.49	**94.16**↑	**70.72**↑	73.14	$+0.05$

Table 14: **Input-Jacobian robustness on ResNet-18 across optimizers.** Mean corruption accuracy (mCA, %) on CIFAR-10-C and CIFAR-100-C for base, base+Jacobian ($+J$), and base+Jacobian+leak ($+J+\ell$), evaluated on the trained checkpoints. The $+J+\ell$ cells are single-seed, and the RSAM CIFAR-100-C base and $+J$ cells use two seeds; all other cells use three seeds. The ASAM $+J+\ell$ CIFAR-10-C entry is a single run that diverged, and its last valid checkpoint is shown. Bold ↑ marks a mean above base.

	CIFAR-10-C mCA			CIFAR-100-C mCA		
Optimizer	base	$+J$	$+J+\ell$	base	$+J$	$+J+\ell$
ASAM	45.77	**51.62**↑	42.14	24.85	**27.90**↑	**27.61**↑
FSAM	44.00	**50.02**↑	**44.10**↑	23.48	**26.57**↑	**26.65**↑
RSAM	43.55	**50.28**↑	**49.07**↑	23.29	**26.93**↑	**27.34**↑

Corruption robustness on ResNet-18. The available optimizer-breadth evaluation is shown in Table 14. The input-Jacobian penalty increases mean corruption accuracy for ASAM, FSAM, and RSAM in the reported checkpoints. The combined $+J+\ell$ cells are single-seed, as noted in the table caption; they are not used for a general cross-optimizer claim in the main text.

K Depth and Stability Boundary

On ResNet-56, shutting the negative branch destabilizes training, so the gated (a=0) runs do not produce valid models. With GELU, the three CIFAR-100 seeds of a=0 reach 1.0, 1.0, and 59.14 test accuracy (two collapse to chance, the third far below the a=1 baseline of 72.69; Table 13); on CIFAR-10 the seeds split with high variance (mean 68.0, standard deviation 15.7), and an interior a=0.003 run also collapses (17.41). The instability is not a configuration error: our ReLU baseline on ResNet-56 reproduces the SGD accuracy reported by Zhang et al. (2026) within seed noise (CIFAR-10 $94.45_{\pm 0.14}$ vs. $94.24_{\pm 0.23}$; CIFAR-100 $72.53_{\pm 0.21}$ vs. $72.60_{\pm 0.34}$), and the identical gating code and protocol are stable on ResNet-18 and ResNet-20. The same failure appears independently in the validation-based a-search (Appendix N): 4 of 12 trials diverge on CIFAR-10 and 7 of 12 on CIFAR-100, concentrated at the small-a end. We therefore treat ResNet-56 as a depth-dependent training-stability boundary of aggressive negative-branch suppression. The precise

failure mechanism is unresolved by the present measurements; investigating this phenomenon in more detail is a direction for future work.

L Extended Discussion and Interpretation

This appendix expands the Discussion of the main text.

Interpretation. Our results support a view of activation-induced generalization in which no single geometric quantity is predictive across all sources of model variation. With the optimizer fixed and the activation varied, Rényi sharpness is less consistent across the evaluated settings, while the input Jacobian and the activation-level negative-branch leak show stronger associations than Rényi sharpness in several settings, and the controlled gating experiments connect the activation-level leak to input sensitivity. This does not replace weight-space geometry with input-space or activation-level geometry; it shows that which geometric signal tracks the gap can depend on which component of the model is varied (optimizer or activation), and that the informative measure may live at a different level of the network: the weights θ, the input x, or the internal pre-activations z_l.

Setting dependence. The activation-level effect is itself setting-dependent. ResNet-18 shows a clear small-leak regime, whereas ResNet-20 shows smaller and mixed changes and WRN-28-10 shows a larger improvement for GELU; on ResNet-56, aggressively suppressing the negative branch can destabilize training. The relatively flat response in some settings indicates that the leak is not a sufficient statistic for generalization: it is a mechanism that becomes predictive when activation variation produces meaningful changes in input sensitivity. Reducing the leak is therefore best viewed as a controllable intervention rather than a universally optimal setting. Detailed per-architecture grids and the ResNet-56 stability analysis are in Appendices G and K.

What the evidence establishes. The gating intervention provides causal evidence connecting the gated negative branch, the measured leak, and input sensitivity. Yet it does not imply that the leak alone determines generalization; other activation- and architecture-dependent factors remain relevant. The ResNet-56 results indicate a depth-dependent stability boundary under aggressive leak suppression. The correlation analysis spans five to seven activations per setting, so the exact ordering among measures should not be overinterpreted; we do not establish a universal ordering among weight-space, input-space, and activation-level measures. The training objective combines established input-Jacobian regularization with the mechanism-derived leak term; our contribution is the latter and the empirical characterization of their complementary effects. The two leak controls also need not combine additively: a matched architectural-gating plus leak-penalty ablation is reported in Appendix G.2.

M Limitations, Scope, and Future Directions

This appendix expands the Limitations of the main text.

Scope. The experiments span several commonly used activations, convolutional architectures, CIFAR-10 and CIFAR-100 (with TinyImageNet in the correlation analysis), a ViT setting, and multiple optimizers. We do not claim a universal geometry predictor or a universally optimal leak level; broader activation families, architectures, and a systematic study of depth would help establish how general the leak-sensitivity relationship is and why aggressive leak suppression becomes unstable in some regimes.

Future work. In our experiments the leak penalty interacts differently with different optimizers, including sharpness-aware optimizers such as SAM and RSAM; characterizing why these interactions differ, rather than establishing that they occur, is a natural next step. Understanding the depth-dependent stability seen on ResNet-56 and testing whether the mechanism extends beyond the standard activations studied here are additional directions.

N Training and Hyperparameter Optimization Details

N.1 Enriched-loss training procedure

Algorithm 1 writes out the enriched-loss training step of §5. The two penalties are added to the cross-entropy loss before the parameter update, so the step wraps any base optimizer and any architecture f_θ without further changes.

Algorithm 1 Enriched-loss training step. The penalties are added to the loss before the update, so the step is optimizer-agnostic (any base optimizer) and architecture-agnostic (any f_θ).

Sample minibatch (x, y); set x.requires_grad if $\lambda > 0$
$\hat{y} \leftarrow f_\theta(x)$; capture negative-branch pre-activations $\{z_l\}$
$L \leftarrow \mathrm{CE}(\hat{y}, y)$
if $\lambda > 0$ **then** ▷ input-Jacobian penalty, one Rademacher projection
    draw $v \sim$ Rademacher; $L \mathrel{+}= \lambda \, \|\partial(v^\top \hat{y})/\partial x\|^2$
end if
if $\mu > 0$ **then** ▷ leak penalty, analytic φ', no extra backward
    $L \mathrel{+}= \mu \, \frac{1}{|\mathcal{A}|} \sum_{l \in \mathcal{A}} \mathrm{mean}_{z_l < 0}[\varphi'(z_l)^2]$
end if
L.backward(); optimizer step

N.2 Hyperparameter search

The experiments follow the base training configurations of Zhang et al. (2026), including the Tiny-ImageNet, ResNet-50, and ViT settings; all per-setting values are listed in Table 15. Our only deviation from the reference is the ResNet-18 negative-branch gating weight decay. Table 16 lists the per-setting regularizer strengths selected by the validation search (Algorithm 2), which we run with Optuna (Akiba et al., 2019). Sharpness-aware optimizers wrap SGD with perturbation radius ρ (SAM/RSAM/SSAM 0.05, FSAM 0.1, ASAM 0.5; RSAM α=2.0, the optimizer's order, distinct from the Rényi measure's α of §2), following the reference, and reuse the tuned (λ, μ) winners. The negative-branch gate coefficient a is selected per setting by the same validation search: the best interior a is 0.003 on ResNet-18/CIFAR-10 (GELU; Table 9), 0.013 on ResNet-18/CIFAR-100, 0.026 on ResNet-20/CIFAR-100, and ≈ 0.006 on WRN-28-10 (shown in parentheses in Table 1).

Table 15: Base training configuration (all rows follow Zhang et al. (2026); our only deviation is the ResNet-18 gating weight decay below): SGD with momentum 0.9, cosine schedule after a 10-epoch linear warm-up. The ResNet-18 negative-branch gating runs (Table 1, Table 7) use weight decay 1×10^{-4}; all other runs use 5×10^{-4}.

Setting	optimizer	lr	weight decay	epochs	batch
ResNet-18/20/34/50/56, CIFAR-10/100	SGD	0.1	5×10^{-4}	200	128
WRN-28-10, CIFAR-10/100	SGD	0.1	5×10^{-4}	200	64
ResNet-18, TinyImageNet	SGD	0.2	1×10^{-4}	200	128
ViT-B/16 (pretrained), CIFAR-10/100	SGD	0.01	5×10^{-4}	20	128

Table 16: Regularizer strengths selected by the per-setting validation search (Algorithm 2), on SGD: leak weight μ (leak-only loss), input-Jacobian weight λ (Jacobian-only loss), and the jointly-tuned (λ, μ) for the combined objective. The sharpness-aware optimizer runs reuse these winners. A dash marks a setting with no separately tuned leak-only strength. The repeated $\lambda \approx 1.15 \times 10^{-4}$ on CIFAR-100 is the validation optimum, near the lower end of the search range.

Dataset / architecture	leak μ	Jacobian λ	combined (λ, μ)
CIFAR-10 / ResNet-20	0.0112	0.00187	(0.00205, 0.162)
CIFAR-10 / ResNet-56	0.0229	0.00431	(0.00187, 0.0398)
CIFAR-100 / ResNet-18	–	0.000115	(0.000115, 0.115)
CIFAR-100 / ResNet-20	0.0311	0.000115	(0.000115, 0.00523)
CIFAR-100 / ResNet-56	0.0112	0.000115	(0.000115, 0.00112)

Algorithm 2 Per-dataset hyperparameter search for (λ, μ) with Optuna. TPE is the sampler that proposes trials; MedianPruner stops a trial early if it falls below the running median.

Fix base config (SGD lr0.1, wd5×10^{-4}, cosine, batch 128); tune only λ, μ
Hold out a 10% no-augmentation validation split
for trial $= 1 \dots N$ **do** ▷ TPE sampler, MedianPruner
    sample $\lambda \in [10^{-4}, 10^{-1}]$, $\mu \in [10^{-3}, 3\times10^{-1}]$ (log-uniform)
    train a short proxy; report val accuracy each epoch; prune below the running median
end for
retrain top-3 (λ, μ) at full epochs (200); pick winner on val
retrain winner on full train; report test accuracy, best trial, best epoch, time, peak memory

O Reproduction of "Rényi Sharpness: A Novel Sharpness That Strongly Correlates with Generalization"

Before changing the activation axis, we reproduce the base (SGD) configuration of Zhang et al. (2026) as a setup check. We follow their reported SGD training recipe for the ResNet-20 sharpness-aware-minimization setting (their Table 1 and Appendix J): SGD with momentum 0.9, initial learning rate 0.1 with cosine decay after a 10-epoch linear warm-up, weight decay 5×10^{-4}, 200 epochs, batch size 128, standard augmentation (random crop with 4-pixel padding, horizontal flip, and channel normalization), and no label smoothing. Table 17 places their reported SGD configuration and test accuracy beside our reproduction. Our accuracies match theirs within one standard error on both datasets (three seeds, ReLU), confirming that the training setup is faithful before we vary the activation function. This is a configuration check on the setup, not a new benchmark result.

Table 17: Reproduction of the SGD base configuration of Zhang et al. (2026) on ResNet-20. The configuration rows follow their reported recipe; the accuracy rows report SGD test accuracy from Table 1 of Zhang et al. (2026) (average $\pm$ standard error over five runs) beside our reproduction (ReLU, three-seed mean $\pm$ standard error) under the same configuration.

	Zhang et al. (2026)	Ours (reproduction)
Base configuration (ResNet-20)		
Optimizer	SGD, momentum 0.9	SGD, momentum 0.9
Initial LR / schedule	0.1, cosine, 10-ep warm-up	0.1, cosine, 10-ep warm-up
Weight decay	5×10^{-4}	5×10^{-4}
Epochs / batch size	200 / 128	200 / 128
Augmentation	crop(pad 4)+flip+norm	crop(pad 4)+flip+norm
Label smoothing	none	none
SGD test accuracy (%)		
CIFAR-10	$92.68_{\pm 0.25}$	$92.68_{\pm 0.11}$
CIFAR-100	$69.12_{\pm 0.17}$	$69.20_{\pm 0.21}$