Prediction Stability as a Function-Space Proxy for Flat Minima

Anonymous authors
Paper under double-blind review

Abstract

Flat minima in neural network loss landscapes are widely believed to support better generalization. However, many existing definitions of flatness are poorly specified, as they are highly sensitive to reparameterization and architectural design choices. This raises a fundamental question: is flatness truly a property of the parameter space, or does it reflect stability in the learned function itself? We adopt a function-centered perspective and examine flatness through behavior in output space rather than through parameter-space geometry. From this standpoint, meaningful flatness is expressed as stability in model predictions under controlled perturbations, independent of how the weights are parameterized. To investigate this idea, we introduce the Function-Space Flatness Proxy, an analytical probe designed for both empirical and theoretical evaluation of output-space stability. The proxy measures prediction stability under perturbations and incorporates stability-guided model selection along with a stability complexity metric. It is invariant to reparameterization and does not rely on second-order quantities such as the Hessian or Fisher information matrix. Crucially, FSFP pursues flatness indirectly: it optimizes for prediction stability in output space, and flat minima are the intended destination reached through that stability objective, via a principled route that never requires measuring curvature explicitly. Using this framework, we analyze test accuracy, loss dynamics, calibration, and negative log-likelihood to explore how output-space stability relates to generalization. Experiments on CIFAR benchmarks with ResNet architectures provide empirical validation of the probe. The observed improvements in accuracy and the divergence between absolute and normalized sharpness measures are interpreted as evidence that output-space stability yields flat minima as the intended outcome of a principled stability-driven approach, rather than serving as a performance comparison against existing approaches. Overall, these findings suggest that output-space stability offers a practical and interpretable lens for studying flatness (including sharpness geometry and generalization behavior) without reducing the concept to geometric properties alone. This function-centered perspective complements parameter-space approaches and broadens the understanding of generalization beyond traditional notions of sharpness.

Keywords: flat minima, prediction stability, function-space optimization, generalization, reparameterization invariance, output-space regularization, sharpness-aware training

1 Introduction

The relationship between loss landscape geometry and generalization has been extensively studied since Hochreiter & Schmidhuber (1997) proposed that flat minima generalize better than sharp ones. This hypothesis motivated sharpness-aware optimization, most notably SAM (Foret et al., 2021), which explicitly seeks flat regions by optimizing against worst-case weight perturbations. Despite strong empirical performance, parameter-space flatness measures carry a fundamental limitation: they are sensitive to reparameterization. Dinh et al. (2017) showed that for networks with homogeneous activations, the same function can correspond to parameter vectors with arbitrarily different curvature values, simply by rescaling weights across layers.

For a two-layer network, the weight rescaling

$$W_1 \;\to\; \alpha\, W_1, \qquad W_2 \;\to\; \tfrac{1}{\alpha}\, W_2 \tag{1}$$

leaves $f_\theta(x) = f_{\phi(\theta)}(x)$ completely unchanged, yet the Hessian-based sharpness transforms as

$$\text{Sharpness}\big(\phi(\theta)\big) \;\approx\; \alpha^{-2} \cdot \text{Sharpness}(\theta). \tag{2}$$

Sharpness can therefore be made arbitrarily small or large without altering predictions at all. If flatness is meant to explain generalization through the behavior of the learned function, it should not be defined through a coordinate system the function itself does not see.

Function-space perspective. This paper takes a function-centered view on flatness. Rather than measuring curvature in weight space, we ask whether flat-minima behavior can be identified and studied through the output-space behavior of the learned function. The theoretical basis is a formal chain we establish in Section 3: flat minima imply small loss variations under parameter perturbation; by Lipschitz continuity, small loss variations imply small output variations; small output variations imply prediction stability. This chain connects parameter-space geometry to a quantity observable entirely from model outputs, without access to the Hessian or the weight space.

A critical observation motivating this perspective is that models trained with sharpness-aware optimization exhibit more stable predictions under small perturbations compared to standard SGD. This suggests that flatness-promoting training leaves a detectable signature at the output level, one that does not require access to the weight space to observe. Conversely, if the loss landscape is genuinely flat, small parameter perturbations induce only small loss variations, which by Lipschitz continuity of the model and loss imply controlled output changes, which in turn prevent prediction flips.

Qiu et al. (2023) formalize this by contrasting parameter-space MAP (PS-MAP) and function-space MAP (FS-MAP) estimation: PS-MAP objectives are distorted by weight rescaling, while FS-MAP inherently biases solutions toward regions where predictions are stable under output perturbations. Subsequent work has explored normalized sharpness measures (Jang et al., 2022) and Riemannian manifold formulations of neural network parameter space (Kristiadi et al., 2024), yet these remain anchored in parameter space. Our work departs from this entirely: stability defined at the level of outputs is the starting point, not a correction applied afterward.

A further motivation is that generalization is shaped not only by where optimization ends, but by how the model navigates the loss landscape throughout training (Golatkar et al., 2019; Achille et al., 2019). A method that regularizes output-space stability does not merely impose a constraint at convergence: at each training step it measures whether the current function exhibits stable predictions under perturbations and applies a corrective gradient if it does not. This continuous signal (which we call the *stability-based recovery signal* of FSFP) steers the trajectory toward configurations where predictions are robust, and flat minima are the intended destination of that stability pressure, reached indirectly through output-space optimization rather than through direct curvature minimization. The path matters because the corrective forces accumulated along it determine the geometry of the region where the model ultimately settles.

The FSFP probe. We formalize this in the Function-Space Flatness Proxy (FSFP). At each training step, FSFP applies controlled perturbations to the model's output space, identifies the most prediction-stable direction among sampled trials, and penalizes the correlation between predicted probabilities and the perturbation pattern along that direction. This penalty introduces a structured gradient signal that flows back through the network via standard backpropagation, creating an inductive bias on weight updates that steers optimization toward regions of high prediction stability. Importantly, FSFP never measures or directly minimizes parameter-space flatness; flat minima are the principled indirect goal of the stability objective, reached through output-space optimization rather than by measuring or minimizing parameter-space curvature directly. The stability complexity measure is reparameterization-invariant by construction, since it is computed entirely from function outputs. It requires no Hessian estimation, no adversarial weight perturbation, and only a single forward-backward pass per step.

Experimental design. To isolate the behavior of each method, we adopt a minimally regularized training regime on CIFAR-100 using standard ResNet architectures without data augmentation. This design removes strong explicit smoothing while preserving non-trivial task complexity, enabling meaningful comparison of optimization dynamics across methods. This is a deliberate methodological choice, not a limitation of scope: when augmentation already regularizes the landscape, individual contributions of stability regularization and flatness mechanisms become difficult to isolate and attribute. The reduced-regularization regime provides conditions under which each method's navigational behavior is clearly observable. SAM directly pursues flat minima via parameter-space perturbations; FSFP pursues prediction stability, with flatness as the intended destination reached through that stability-driven route. This distinction makes their comparison particularly informative.

Contributions. While function-space perspectives on flatness have been proposed in recent theoretical work (Qiu et al., 2023; Mason-Williams et al., 2025), practical instruments for studying how stability regularization relates to flatness purely through output-space signals remain underexplored. This work bridges this gap by introducing FSFP as a deliberately constructed analytical probe, not as a replacement for parameter-space methods, but as a complementary instrument for investigating how flat-minima behavior emerges when optimization is guided entirely by function-output signals rather than geometric curvature. Our key contributions are:

1. **Algorithm:** FSFP, a stability-guided output perturbation procedure with a stability complexity measure that directly targets prediction consistency.
2. **Theoretical framework:** We prove reparameterization invariance and connect prediction stability to generalization bounds through function-space complexity (formal statements and proofs in Appendix A).
3. **Empirical validation:** We demonstrate that regularizing for prediction stability (i) improves test accuracy, (ii) maintains calibration comparable to the SGD baseline, (iii) increases prediction consistency, and (iv) achieves computational efficiency (approximately 1.1× cost vs. 2× for SAM).
4. **Conceptual contribution:** We reframe flat minima as regions where predictions remain stable under output-space perturbations, offering a function-space perspective that avoids the reparameterization problem and provides an interpretable lens on generalization beyond geometric sharpness.

Summary of findings. FSFP consistently improves test accuracy over the baseline while training accuracy remains at the same near-perfect level across all runs and architectures, confirming that the gain reflects genuine generalization improvement rather than reduced training fit. Calibration (ECE) under FSFP remains comparable to the SGD baseline. SAM achieves higher absolute test accuracy at substantially higher per-step computational cost. The divergence between absolute and normalized sharpness measurements reveals that the two methods arrive at geometrically distinct minima: FSFP, by optimizing prediction stability, reaches lower absolute sharpness through its stability-driven route; SAM, by directly minimizing parameter-space curvature, produces lower normalized sharpness. Full results are in Section 6. A supplementary robustness check on an additional dataset/architecture pairing appears in Appendix C.5, including a regime where FSFP's flatness advantage extends to both sharpness metrics even as SAM retains the accuracy edge.

2 Related Work

2.1 Flatness and Generalization

The relationship between loss landscape geometry and generalization has been extensively studied since Hochreiter & Schmidhuber (1997) first proposed that wide minima exhibit superior generalization due to robustness under parameter perturbations. Subsequent empirical investigations by Keskar et al. (2017) demonstrated that batch size significantly influences the sharpness of discovered minima, with large-batch optimization converging to sharp minima that generalize poorly compared to the flatter minima found

through small-batch training. Foret et al. (2021) formalized this intuition in SAM, which explicitly seeks flat regions by solving:

$$\min_{\theta} \max_{\|\varepsilon\| \leq \rho} L(\theta + \varepsilon). \tag{3}$$

While SAM has demonstrated remarkable empirical success, its parameter-space formulation inherits a fundamental limitation: the flatness measure varies under reparameterization, meaning mathematically equivalent networks can exhibit arbitrarily different sharpness values (Dinh et al., 2017).

2.2 Function-Space Perspectives

Recognition of these limitations has motivated recent exploration of function-space alternatives. Recent work has revealed that SAM's success in natural language processing stems primarily from regularization of output statistics rather than parameter-space geometry, suggesting that function-level measures may better capture generalization-relevant properties. Several concurrent approaches have pursued related directions: Flatness-Aware Minimization (FAM) (Adilova et al., 2023; Zhang et al., 2023) introduces relative flatness normalization to partially mitigate reparameterization sensitivity; Flatness-Aware Distance (FAD) (Adilova et al., 2023) proposes alternative relative metrics; and flatness-aware Stochastic Gradient Langevin Dynamics (fSGLD) (Bruno et al., 2025) biases stochastic gradient noise toward flat regions during sampling. However, these methods either retain some parameter-space dependence or lack explicit connections to established generalization theory. Our approach differs in three fundamental aspects: we measure flatness exclusively through prediction stability under output perturbations; we draw theoretical motivation from complexity-based generalization frameworks; and we provide reparameterization-invariant guarantees by never referencing parameter space. This choice is supported by an empirical observation, independent of FSFP's own mechanism: across 30 plain-SGD models trained under varying learning rate and batch size, flatness and prediction stability under weight-space perturbation are significantly correlated (Appendix C.7).

2.3 Norm-Based and Stability-Based Generalization Bounds

Traditional capacity-based generalization theory bounds the generalization gap through measures of model complexity. Neyshabur et al. (2015) and Bartlett et al. (2017) established norm-based bounds showing that for neural networks with bounded norms, the generalization error can be controlled independently of the number of parameters. Specifically, for a network with L layers and weight matrices $W_1, \ldots, W_L$, the Rademacher complexity is bounded as

$$\mathcal{R}(\mathcal{F}) \leq \frac{C}{\sqrt{n}} \prod_{l=1}^{L} \|W_l\|_{\text{spec}}, \tag{4}$$

where $\|W_l\|_{\text{spec}}$ denotes the spectral norm. Our stability-based approach complements these norm-based bounds. While norm-based bounds control capacity through parameter magnitudes, stability-based bounds (Theorem 7 in Appendix A) control capacity through algorithmic properties. Crucially, both approaches aim to characterize the effective complexity of the learned function rather than merely counting parameters, aligning with the view that generalization depends on which function is learned, not how it is parameterized (Neyshabur et al., 2017).

2.4 Rademacher Complexity and Our Departure

Rademacher complexity (Bartlett & Mendelson, 2002) provides a principled framework for bounding generalization error through hypothesis class capacity:

$$\text{Rad}(F) = \mathbb{E}_{\sigma}\left[\sup_{f \in F} \frac{1}{n} \sum_{i} \sigma_i f(x_i)\right], \tag{5}$$

where $\sigma_i \in \{-1, +1\}$ are independent Rademacher variables. This quantity measures how well functions in class F can correlate with arbitrary random patterns; high correlation indicates capacity to fit noise,

suggesting overfitting risk. The fundamental principle is that sensitivity to random perturbations reveals poor generalization. While we do not directly compute Rademacher complexity (which would require intractable averaging over binary random signs and supremum operations), this principle directly motivates our approach.

Our method departs significantly in implementation. Rather than using binary Rademacher variables, we inject continuous Gaussian perturbations into the output space and observe how predictions respond across multiple independent trials. We do not average over all perturbation directions as traditional Rademacher complexity would require. Instead, following Yang et al. (2019), we sample T independent perturbation trials and identify the most stable direction among the T sampled perturbations, which we show empirically is insensitive to increasing T well beyond the point of diminishing returns (Appendix C.6). By measuring complexity specifically along this stability-selected direction and using it as a regularization signal, we achieve computational tractability through finite sampling while actively guiding optimization toward empirically stable regions.

3 Theoretical Framework: From Flatness to Prediction Stability

This section presents the conceptual and theoretical structure connecting parameter-space flatness to output-space prediction stability. Complete formal statements and proofs are in Appendix A; here we describe what is shown and why it matters.

3.1 The Flatness-to-Stability Chain

The theoretical backbone of FSFP is a three-step chain connecting parameter-space geometry to output-level observables. All derivations, expansions, and proofs are collected in Appendix A; we give here only the logical structure.

Step 1: Flatness bounds loss variation. A minimum is flat when the loss landscape has low curvature around it, measured by the maximum eigenvalue of the Hessian. Flat minima guarantee that small parameter perturbations produce only small changes in loss. (Definition 3, Appendix A.)

Step 2: Small loss variations imply small output changes. Because the loss depends on the network outputs, bounded loss variation translates (via the output Jacobian and the Hessian decomposition for cross-entropy) into bounded output perturbations. (Appendix A.2.2.)

Step 3: Small output changes prevent prediction flips. When output perturbations are smaller than half the decision margin, the argmax (and therefore the predicted class) is mathematically guaranteed to remain unchanged. (Lemma 4, Appendix A.)

These three steps are formalized and combined in Proposition 6 and Theorem 5 (Appendix A).

3.2 Stability and Generalization

Enforcing output-space stability connects to one formal piece of generalization theory; the complete statement and proof are in Appendix A.

Generalization via algorithmic stability (Theorem 7). A β-uniformly stable training algorithm yields a generalization bound that shrinks as $\beta \to 0$. We include this as general background connecting algorithmic stability to generalization; we do not derive a formal bound tying β specifically to FSFP-trained minima; see Appendix A for why an earlier attempt at such a bound was removed, and Section 6.3 for the empirical relationship between $\mathcal{R}_{\text{flat}}$ and standard flatness measures.

3.3 Inverting the Chain: From Stability to Flatness

By *enforcing* output stability during training, FSFP steers the model toward configurations consistent with the flatness-to-stability chain — without ever computing a Hessian. The key hypothesis is that the essential property of flat minima is not low parameter curvature per se, but stable predictions: a reparameterization-invariant, function-space quantity that simultaneously serves as an indirect flatness measure and a predictor of generalization.

4 Method: FSFP

4.1 Core Intuition

The central idea of FSFP is that the optimization path carries information about the geometry being traversed, and this information can be read directly from function outputs. When a model is navigating through a sharp region of the loss landscape, its predictions are sensitive: small changes in the output space cause label flips, confidence estimates shift, and the function behaves erratically under perturbation. When the model is in or approaching a flat region, predictions are stable: the same perturbations leave labels unchanged and confidence distributions nearly intact.

FSFP exploits this relationship by generating a *recovery signal* at each training step. Controlled perturbations are applied to the model's output space, and the resulting prediction stability is measured. When instability is detected (the current output configuration is sensitive), the recovery signal penalizes this and redirects the gradient toward more stable, flatter configurations. The perturbations are not injected for generic regularization; they are a structured probe of the local output neighborhood. The signal they generate is a corrective force on the optimization trajectory, continuously pushing toward regions where output stability is consistent with flat-minima behavior.

This framing makes explicit that FSFP operates on the *path*, not just the endpoint. At each step, it asks whether the current position is consistent with flat-minima behavior and, if not, applies a corrective force through the training objective. Over the course of training, this continuous correction steers the trajectory toward the function-space analogue of a flat minimum: a region where outputs are stable, predictions are robust, and the learned function is far from decision boundaries in the directions that matter.

It is important to emphasize that FSFP is a proxy and a navigational tool, not a replacement for Hessian- or Fisher-based flatness measures. Those measures remain theoretically precise. FSFP provides a computationally lightweight, reparameterization-invariant signal that reflects the functional consequences of flatness in output space, complementing classical approaches while remaining tractable at scale.

4.2 Measuring Prediction Stability

Given a mini-batch of n samples with logits $Z \in \mathbb{R}^{n\times k}$ (k classes), FSFP probes the local output neighborhood across T independent trials. Each trial injects Gaussian noise directly into the logits:

$$\tilde{Z}^{(t)} = Z + \sigma^{(t)}, \qquad \sigma^{(t)} \sim \mathcal{N}(0,\, \varepsilon^2 I_{n\times k}), \quad t = 1, \ldots, T. \tag{6}$$

The noise is sampled independently per sample and per class, allowing perturbations to alter class score orderings and produce genuine prediction flips. Instability for trial t is then counted as:

$$\Delta^{(t)} \;=\; \sum_{i=1}^{n} \mathbf{1}\left[\, \arg\max_j \; \tilde{z}_{ij}^{(t)} \;\neq\; \arg\max_j \; z_{ij} \,\right]. \tag{7}$$

A high flip count indicates a sharp, sensitive configuration; a low count indicates stability consistent with a flat region.

4.3 Stability-Guided Direction Selection

Rather than aggregating across all trials, FSFP anchors to the most prediction-stable direction found:

$$t^* \;=\; \arg\min_{t} \; \Delta^{(t)}. \tag{8}$$

This calibrates the recovery signal to geometry the model can actually reach, avoiding worst-case directions irrelevant to the current optimization trajectory. The gradient of $\mathcal{R}_{\text{flat}}$ along direction t^* introduces a corrective bias in weight updates, indirectly steering optimization toward flatter regions through a function-space stability signal (see Appendix A, Remark 2).

4.4 Stability Complexity Measure

To quantify the strength of the recovery signal along the selected direction, we define the stability-induced complexity measure.

Definition 1 (Stability-Induced Complexity Measure)**.** Let $p_\theta = \text{softmax}(Z) \in \mathbb{R}^{n\times k}$ denote the predicted class probabilities for a batch of n samples across k classes, where $Z \in \mathbb{R}^{n\times k}$ represents the model logits. Given the stability-selected perturbation direction $\sigma^{(t^*)} \in \mathbb{R}^{n\times k}$, obtained by minimizing prediction flips across T independent trials, the stability-induced complexity is defined as:

$$\mathcal{R}_{\text{flat}} = \max_{j\in\{1,\ldots,k\}} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i^{(t^*)} \cdot p_{\theta,j}(x_i) \right|. \tag{9}$$

This measures the worst-case empirical correlation, over output dimensions, between the stability-selected perturbation pattern and predicted class probabilities. Low correlation indicates that the model's outputs are robust to the probe, consistent with flat-minima behavior. High correlation indicates residual sensitivity along the most stable available direction, meaning the current position has not yet reached a flat region and the recovery signal should be amplified.

The formulation comprises three operations: the inner summation computes sample-wise correlation between the perturbation pattern and predicted probabilities for class j; the absolute value ensures sign invariance; and the maximum operator identifies the most vulnerable output dimension, yielding a conservative estimate of functional stability.

Our formulation differs fundamentally from classical Rademacher complexity in using continuous Gaussian perturbations instead of binary variables, evaluating correlation along a single empirically selected direction rather than computing a supremum over the entire hypothesis class, and operating exclusively on function outputs $p_\theta(x)$ rather than parameters θ. This ensures computational tractability for large neural networks while maintaining reparameterization invariance (Dinh et al., 2017). The stability-selected direction t^* is the most stable direction among the T sampled perturbations, which we show empirically is insensitive to increasing T well beyond the point of diminishing returns (Appendix C.6), enabling the regularizer to actively guide optimization toward stable configurations.

Proposition 2 (Reparameterization Invariance of FSFP)**.** Let $f_\theta : \mathcal{X} \to \mathbb{R}^k$ be a neural network and let $\phi : \Theta \to \Theta'$ be a reparameterization such that $f_\theta = f_{\phi(\theta)}$ for all θ. Then the stability complexity measure $\mathcal{R}_{\text{flat}}$ satisfies:

$$\mathcal{R}_{\text{flat}}(\theta) = \mathcal{R}_{\text{flat}}(\phi(\theta)) \tag{10}$$

Proof: See Appendix A for the complete proof.

This invariance is a fundamental advantage over parameter-space sharpness measures: function-space measures achieve complete reparameterization invariance by construction, since two parameter vectors implementing the same function produce identical FSFP regularization signals.

4.5 Algorithm and Training Objective

Algorithm 1 presents the complete FSFP procedure. The algorithm takes a mini-batch of logits Z, a perturbation scale ε (default 0.1), and a number of trials T (default 20), and returns the stability complexity R_{flat} as a differentiable loss term.

Algorithm 1 Function Space Flatness Proxy (FSFP)

Require: Batch outputs $Z \in \mathbb{R}^{n \times k}$ from $f_\theta(X)$
Require: Perturbation scale ε (default: 0.1)
Require: Number of trials T (default: 20)

```
P ← SoftMax(Z)                                  ▷ Predicted probabilities
ŷ ← arg max(Z, axis = 1)                        ▷ Original predictions
for t = 1 to T do
    σ^(t) ~ N(0, ε^2 I_{n×k})                   ▷ Sample per-class output-space probe
    Z̃^(t) ← Z + σ^(t)                           ▷ Perturb logits independently per class
    ŷ^(t) ← arg max(Z̃^(t), axis = 1)            ▷ Perturbed predictions
    Δ^(t) ← Σ_i ⊮[ŷ_i^(t) ≠ ŷ_i]                ▷ Count prediction flips
end for
t* ← arg min_t Δ^(t)                            ▷ Select most stable direction
for each class j do
    C_j ← | (1/n) Σ_i σ_i^(t*) · p_ij |         ▷ Recovery signal strength
end for
return max_j C_j                                ▷ Worst-case instability (the recovery signal)
```

The recovery signal is incorporated into the training objective as

$$L_{\text{total}} = L_{\text{CE}}(f_\theta(X), Y) + \lambda \cdot R_{\text{flat}}(f_\theta(X)), \tag{11}$$

where L_{CE} denotes cross-entropy loss and λ controls the strength of the recovery signal. At each step, the gradient of R_{flat} provides a corrective force that pushes the optimization trajectory away from configurations with high output-space sensitivity, steering it toward flatter regions. The hyperparameter λ is typically chosen in the range 0.3–0.5; see Appendix B for theoretical justification of this range.

Gradient mechanism. Although $\mathcal{R}_{\text{flat}}$ is defined entirely in output space, its effect operates through the parameter space via the chain rule. The gradient of $\mathcal{R}_{\text{flat}}$ with respect to logit z_{ij} is:

$$\frac{\partial \mathcal{R}_{\text{flat}}}{\partial z_{ij}} = \frac{\sigma_i^{(t^*)}}{n} \cdot \frac{\partial p_{\theta,j^*}}{\partial z_{ij}}, \tag{12}$$

where $j^* = \arg\max_j C_j$ is the most vulnerable output dimension and $\frac{\partial p_{\theta,j^*}}{\partial z_{ij}}$ is the Jacobian of the softmax with respect to the logits. This gradient is class-discriminating: it is nonzero only for the class dimension identified as most sensitive, and its sign and magnitude depend on how the current predicted probabilities align with the selected noise direction $\sigma^{(t^*)}$. Propagated through the network via standard backpropagation, it introduces a structured bias on the weight updates at every training step, discouraging the network from settling into configurations where predicted class probabilities are systematically correlated with arbitrary noise patterns in the output space, a behavior commonly associated with sharp minima. Because $\mathcal{R}_{\text{flat}}$ is computed entirely from function outputs $p_\theta(x)$, the inductive bias is reparameterization-invariant by construction.

Connection to implicit regularization. The recovery signal can be understood as inducing implicit smoothness in the loss landscape. Recent work on edge-of-stability training (Cohen et al., 2021) shows that SGD naturally navigates toward regions with controlled sharpness. FSFP explicitly encodes this preference

by penalizing output-space sensitivity, creating an inductive bias toward wider basins where the function is robust to the perturbations the recovery signal probes.

Computational complexity. Proposition 10 (Appendix A.8) establishes that the FSFP algorithm has complexity $\mathcal{O}(T \cdot n \cdot k + k \cdot n)$, dominated by the T perturbation trials (each $O(nk)$) and the final correlation computation ($O(kn)$). In contrast, SAM requires $\mathcal{O}(2|\theta|)$ where $|\theta|$ is the number of parameters, and typically $|\theta| \gg nk$. For ResNet-50 on CIFAR-100 with $|\theta| \approx 23.7 \times 10^6$, batch size $n = 128$, $k = 100$, and $T = 20$, the ratio is approximately $20 \times 128 \times 100/(23.7 \times 10^6) \approx 0.011$, confirming that FSFP operates in a fundamentally lower-dimensional space than SAM.

λ selection and gradient dominance. Our experiments use $\lambda \in \{0.3, 0.5\}$ rather than $\lambda = 1$. The regularization parameter λ in our training objective controls the balance between the task gradient and the stability gradient:

$$\mathcal{L}_{\text{total}}(f_\theta) = \mathcal{L}_{\text{CE}}(f_\theta(X), Y) + \lambda \cdot \mathcal{R}_{\text{flat}}(f_\theta(X)). \tag{13}$$

Choosing λ is not merely hyperparameter tuning; it determines which gradient component drives representation updates. A fractional λ preserves the μP feature-learning regime, in which the task gradient remains the dominant force and hidden representations evolve meaningfully throughout training. When λ is large enough that $\lambda\|\nabla_\theta \mathcal{R}_{\text{flat}}\| \gg \|\nabla_\theta \mathcal{L}_{\text{CE}}\|$, the stability regularizer dominates and suppresses this feature learning. Using $\lambda < 1$ is therefore practically necessary to avoid regularizer dominance.

The risk of regularizer dominance ($\lambda = 1$). If $\lambda = 1$, the stability constraint is enforced with the same gradient magnitude as the learning signal from the onset of training. This creates a practical risk of over-regularization that manifests in several ways: (i) *Suppression of feature learning:* when $\lambda\|\nabla_\theta \mathcal{R}_{\text{flat}}\|$ is comparable to or exceeds $\|\nabla_\theta \mathcal{L}_{\text{CE}}\|$, the net parameter update is pulled away from directions that reduce classification loss, limiting the model's ability to adapt its internal structure to the task; (ii) *Reduced parameter movement:* the dominant stability gradient discourages large departures from initialization, penalizing the exploration necessary to find a good basin; (iii) *Obstruction of nonlinear curvature:* when parameter updates are suppressed by an overpowering stability term, the network cannot develop the nonlinear representations needed for high-capacity classification; the regularizer acts as an obstacle rather than a filter.

The necessity of a fractional λ. To maintain task-driven gradient dominance throughout training, λ must be treated as a fraction (typically $0.3 \leq \lambda \leq 0.5$): (1) *Surviving the chaotic transient:* the first 2–5 epochs constitute a chaotic initial transient determining the final basin of low loss; a fractional λ ensures the learning signal dominates this phase, allowing the model to choose a performant basin before the stability constraint begins to exert significant influence; (2) *Filtering vs. obstruction:* a fractional λ allows the $\mathcal{R}_{\text{flat}}$ term to act as a selective filter, penalizing high-frequency noise and stabilizing predictions without suppressing the task gradient to the point of impeding representation learning; (3) *Invariance and scale:* while our method is reparameterization-invariant at the prediction level, the numerical value of $\mathcal{R}_{\text{flat}}$ is sensitive to the noise scale ε; λ serves as a necessary normalizer ensuring the regularization gradient does not overwhelm task-specific information.

Proposition 11 (Appendix A.9) shows that when $\lambda < 1$ and the gradient ratio condition holds, the parameter update magnitude satisfies $\|\Delta\theta\| \approx \eta\|g_{\text{CE}}\|(1 + O(\lambda\alpha))$, ensuring task-driven learning dominates. Detailed analysis is in Appendix B.

5 Experimental Setup

Design philosophy. SAM directly minimizes parameter-space sharpness; FSFP regularizes output-space prediction stability, with any reduction in sharpness arising as the intended indirect outcome of stability optimization rather than the result of explicit curvature minimization. The scientific question we ask is: *how does stability-driven output-space regularization (FSFP) compare to direct flatness minimization (SAM) under conditions where the optimizer is known to overfit?* To isolate this behavior, we require an environment where the generalization gap is reliably large and where these effects are clearly measurable, not confounded by other regularization sources.

We construct this environment through a controlled reduction of regularization: by removing standard data augmentation (random cropping, horizontal flipping), we systematically eliminate the implicit regularization that augmentation provides. The network memorizes the training set almost perfectly (training accuracy $\approx 99.98\%$) while test accuracy remains substantially lower. This setup is not a claim about augmentation-free training as a deployment scenario; it is a controlled experimental condition chosen to maximize the signal-to-noise ratio for studying optimizer behavior under reduced regularization. The choice of CIFAR-100 and ResNet architectures serves the same purpose: they are well-understood settings where meaningful generalization gaps under reduced regularization are reliable and reproducible across seeds, making them suitable for isolating the effect of output-space stability regularization. The goal is not to demonstrate scale or dataset breadth, but to study how function-space stability regularization (FSFP) and direct parameter-space flatness minimization (SAM) navigate the same landscape when regularization is minimal.

Training configuration. SGD with learning rate 0.1, weight decay 5×10^{-4}, batch size 128. ResNet-18: 200 epochs; ResNet-50: 100 epochs (approximately 92% convergence due to computational constraints, but sufficient for comparative analysis). FSFP: $\sigma = 0.1$, $\lambda \in \{0.3, 0.5\}$ (see Appendix B for rationale). SAM: perturbation radius $\rho = 0.05$.

Evaluation metrics. Test accuracy (primary), negative log-likelihood (NLL), expected calibration error (ECE), and training accuracy. Training accuracy is reported specifically to confirm that both FSFP and SAM maintain near-perfect training fit throughout; any test accuracy gain therefore reflects genuine generalization improvement, not reduced training fit.

Computational cost. FSFP requires one forward-backward pass per step (identical to SGD); SAM requires two. Memory overhead for FSFP: $\sim 1.2\times$ vs. SGD. All experiments were conducted on a single NVIDIA T4 GPU. Results are averaged over multiple random seeds (5 independent runs per method-architecture pair, with the exception of SAM on ResNet-50 where 2 runs were conducted due to computational constraints (see note below Table 7 in Section 6.2.1).

Complete details of the experimental design, including seed protocol, dataset selection rationale, and architecture selection, are provided in Appendix C (Experimental Details).

6 Experiments

We conduct experiments in three phases: first comparing FSFP against baseline training to establish its effectiveness; then comparing against SAM to evaluate performance relative to a method that directly minimizes parameter-space sharpness; and finally conducting comprehensive sharpness analysis across all three training methodologies to characterize what kind of minima each method each method produces through its respective objective.

6.1 Phase 1: FSFP vs. Baseline

The first experimental phase evaluates FSFP against baseline training to establish effectiveness before comparison against SAM. We present results for both ResNet-50 and ResNet-18 architectures. Each value represents the mean over 5 independent runs, with standard deviation reported.

6.1.1 ResNet-50 (100 Epochs)

FSFP achieves consistent improvements in test accuracy and NLL on ResNet-50 (Table 1). Independent samples t-tests confirm statistically significant improvements: test accuracy ($t = 3.06$, $p = 0.015$) and NLL ($t = -3.28$, $p = 0.010$). The ECE difference ($t = -1.87$, $p = 0.09$) does not reach conventional significance thresholds; FSFP's calibration is comparable to the baseline rather than meaningfully improved.

Table 1: Performance comparison on CIFAR-100 with ResNet-50 (100 epochs, minimal augmentation)

Method	Test Acc (%) ↑	ECE ↓	NLL ↓	Train Acc (%)
Baseline	63.05 (±0.83)	0.0534 (±0.0030)	1.4963 (±0.0413)	99.98 (±0.01)
FSFP (Ours)	64.91 (±1.08)	0.0503 (±0.0022)	1.3997 (±0.0462)	99.96 (±0.01)
Δ (absolute)	+1.86	−0.0031	−0.0966	−0.02
Δ (relative)	+2.95%	−5.81%	−6.46%	−0.02%

FSFP outperforms the baseline in 9 out of 10 possible pairwise run comparisons. The individual run breakdown (Table 2) confirms this: the best-performing FSFP run (66.26%) significantly exceeds the best baseline run (64.45%), while the worst FSFP run (63.43%) remains competitive with the baseline mean.

Table 2: Individual run results for ResNet-50 (test accuracy %)

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std
Baseline	63.13	62.46	62.69	64.45	62.50	63.05 ± 0.83
FSFP	65.61	66.26	64.77	63.43	64.50	64.91 ± 1.08

6.1.2 ResNet-18 (200 Epochs)

ResNet-18 with extended training shows consistent accuracy improvements (Table 3). Statistical significance: test accuracy ($t = 8.68$, $p < 0.0001$); NLL ($t = -4.65$, $p = 0.0016$); ECE ($t = -1.15$, $p = 0.281$, not significant). The ECE figures for FSFP remain in the same range as the baseline; calibration is not improved by FSFP on this architecture.

Table 3: Performance comparison on CIFAR-100 with ResNet-18 (200 epochs, minimal augmentation)

Method	Test Acc (%) ↑	ECE ↓	NLL ↓	Train Acc (%)
Baseline	62.69 (±0.31)	0.1157 (±0.0162)	1.7395 (±0.0745)	99.98 (±0.00)
FSFP (Ours)	64.46 (±0.31)	0.1064 (±0.0022)	1.5724 (±0.0180)	99.98 (±0.00)
Δ (absolute)	+1.77	−0.0093	−0.1671	0.00
Δ (relative)	+2.82%	−8.04%	−9.61%	0.00%

The per-seed breakdown (Table 4) shows that FSFP outperforms the baseline in every individual run, with the gap stable across seeds (FSFP std ±0.31% equals baseline std), confirming that the gain is systematic rather than driven by a single favorable initialization.

Table 4: Individual run results for ResNet-18 (test accuracy %)

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std
Baseline	62.96	62.40	62.49	62.44	63.15	62.69 ± 0.31
FSFP	64.33	64.28	64.13	64.83	64.71	64.46 ± 0.31

6.1.3 Cross-Architecture Summary

Table 5: Summary of improvements across architectures (FSFP vs. Baseline)

Architecture	Params	Epochs	Test Acc Δ	ECE Δ	NLL Δ	p-value (Acc)
ResNet-50	23.7M	100	+2.96%	−5.81%	−6.46%	0.015*
ResNet-18	11M	200	+2.82%	−8.04%	−9.63%	<0.0001***

Both architectures demonstrate consistent and statistically significant improvements (Table 5). Key observations:

Generalization without underfitting. FSFP consistently improves test accuracy while training accuracy remains identical to the baseline (both at $\approx 99.98\%$ across all runs and architectures). The train-test gap narrows with FSFP: from 36.02% to 35.05% on ResNet-50, and from 37.15% to 35.66% on ResNet-18, confirming that the improvement reflects genuine generalization, not an artifact of reduced training fit. Any explanation based on regularization-as-underfitting is directly ruled out by the training accuracy numbers.

Calibration. FSFP does not meaningfully improve calibration over the SGD baseline. The ECE differences observed across runs do not reach statistical significance and should not be interpreted as a calibration gain. ECE is reported for completeness and to confirm that FSFP does not degrade calibration, but it is not a claimed contribution of the method. The low variance of FSFP's ECE across runs (std 0.0022 on ResNet-18, 0.0020 on ResNet-50) reflects the stability of output-space regularization, not an improvement in calibration quality.

Computational overhead. ResNet-50: average epoch time increases from 156.77s to 167.02s (+10.25s, 6.5% overhead); ResNet-18: from 43.60s to 45.58s (+1.98s, 4.5% overhead). The overhead scales favorably with model size, remaining under 10% in both cases.

Training dynamics. During early training (epochs 1-50 for ResNet-50, 1-100 for ResNet-18), both methods show similar convergence rates. The methods diverge in later stages: once training accuracy saturates, the two methods diverge based solely on the flatness of where they have landed. On ResNet-50 (epochs 60-100), the baseline plateaus at approximately 63% while FSFP continues improving, ultimately reaching 64.91%. A similar pattern holds for ResNet-18 in epochs 120-200.

6.2 Phase 2: FSFP vs. SAM

Having demonstrated the statistical significance of FSFP regularization over vanilla training in Phase 1, Phase 2 conducts a comparative evaluation against SAM to examine our central theoretical hypothesis concerning indirect flatness-seeking mechanisms. Our theoretical framework links Hessian-induced curvature properties of the loss landscape to generalization performance via Rademacher complexity, explicitly tracing how curvature influences model outputs. Phase 2 empirically investigates whether the generalization gains at convergence are consistent with the functional consequences predicted by these Hessian-based theoretical insights, rather than measuring curvature directly.

Unlike SAM, which explicitly minimizes sharpness through adversarial weight perturbations guided by gradient-based approximations of the Hessian, FSFP operates entirely in output space. By measuring and regularizing prediction stability under output perturbations, it steers the optimization trajectory toward configurations where the learned function is robust, which by the theoretical chain of Section 3 corresponds to regions consistent with flat minima.

6.2.1 Results

The corresponding results for ResNet-18 are shown in Table 6. SAM achieves the highest test accuracy, followed by FSFP, followed by baseline across all metrics.

Table 6: ResNet-18 performance (200 epochs training)

Method	Runs	Train Acc (%)	Test Acc (%) ↑	ECE ↓	NLL ↓	Time/Epoch
Baseline	5	99.98 ± 0.00	62.69 ± 0.31	0.1157 ± 0.0162	1.7395 ± 0.0745	∼44.0s
FSFP	5	99.98 ± 0.00	64.46 ± 0.31	0.1064 ± 0.0022	1.5724 ± 0.0180	∼44.3s
SAM	5	99.98 ± 0.01	67.73 ± 0.44	0.0805 ± 0.0033	1.3421 ± 0.0104	∼84.2s

The corresponding results for ResNet-50 follow the same ordering (Table 7); SAM was limited to 2 runs on this architecture due to its substantially higher per-epoch wall-clock cost.

Table 7: ResNet-50 performance (100 epochs training)

Method	Runs	Train Acc (%)	Test Acc (%) ↑	ECE ↓	NLL ↓	Time/Epoch
Baseline	5	99.98 ± 0.00	63.05 ± 0.83	0.0534 ± 0.0030	1.4963 ± 0.0408	∼156.8s
FSFP	5	99.98 ± 0.01	64.91 ± 1.08	0.0503 ± 0.0020	1.3997 ± 0.0490	∼163.7s
SAM	2[1]	99.98 ± 0.00	68.01 ± 0.39	0.0585 ± 0.0052	1.2264 ± 0.0142	∼349.9s

The results demonstrate a consistent performance ordering across both architectural scales (Tables 6–7): SAM achieves the highest test accuracy, followed by FSFP, followed by baseline. On ResNet-18, SAM attained 67.73% test accuracy (5.04 percentage point improvement over baseline), while FSFP achieved 64.46% (1.77 pp improvement). On ResNet-50, SAM reached 68.01% (4.96 pp over baseline) while FSFP achieved 64.91% (1.86 pp improvement). Table 8 quantifies these gains relative to baseline together with the computational overhead of each method.

Table 8: Performance improvements over baseline

Architecture	Method	Δ Test Acc	Δ NLL (%)	Δ ECE	Time Overhead
ResNet-18	FSFP	+2.82%	−9.61%	−0.0093	+0.7%
ResNet-18	SAM	+8.04%	−22.85%	−0.0352	+91.4%
ResNet-50	FSFP	+1.86%	−6.5%	−0.0031	+4.4%
ResNet-50	SAM	+4.96%	−18.0%	+0.0051	+123.1%

NLL and probabilistic prediction quality. On ResNet-18, SAM achieved an NLL of 1.3421, a 22.85% reduction compared to baseline and a 14.63% improvement over FSFP's 1.5724. On ResNet-50, SAM's NLL of 1.2264 represents an 18.0% improvement over baseline and a 12.4% advantage over FSFP. FSFP improves NLL over the baseline on both architectures, consistent with converging to flatter minima. ECE under FSFP remains comparable to the SGD baseline; no calibration improvement is claimed.

Computational cost. SAM's dual forward-backward pass requirement results in approximately double the training time per epoch: 84.2s versus 44.0s on ResNet-18, and 349.9s versus 156.8s on ResNet-50. FSFP requires only 44.3s per epoch on ResNet-18 (+0.7% overhead) and 163.7s on ResNet-50 (+4.4% overhead), making it particularly attractive when training time is constrained.

Convergence dynamics. The epoch-by-epoch progression reveals how each method navigates the optimization landscape (Table 9). During early training (epochs 1-50 on ResNet-18), all methods exhibit similar convergence rates. The methods diverge in mid-to-late training. FSFP shows more stable mid-training behavior (epochs 50-120), with test accuracy climbing steadily without the fluctuations occasionally observed in

[1]SAM on ResNet-50 requires approximately 2.23× the wall-clock time of baseline per epoch (349.9s vs. 156.8s); five full runs were not feasible within our compute budget. The five ResNet-18 SAM runs (std ±0.44%) confirm that SAM's variance is low enough that two runs reliably characterize its ResNet-50 performance.

baseline training. SAM exhibits the most distinctive pattern, with test accuracy continuing to improve even in late training phases where baseline has plateaued. On ResNet-18, SAM's improvement of 14.4 percentage points (epochs 100 to 200) surpassed both baseline (+5.44 points) and FSFP (+13.75 points), demonstrating superior late-stage optimization capability.

Table 9: Convergence summary on ResNet-18 (epochs 20, 50, 100, 150, 200)

Method	Epoch	Train Acc (%)	Test Acc (%)	ECE	NLL	Key Observation
Baseline	20	85.08	46.40	0.2186	2.4254	Early increase, test plateau
Baseline	50	88.84	49.11	0.2426	2.4827	Test plateau, slight overfitting
Baseline	100	99.17	57.71	0.1342	1.7723	Gains in late stage
Baseline	150	71.63	46.96	0.1412	2.1990	Unstable training phase
Baseline	200	99.98	63.15	0.1016	1.6645	Stabilized
FSFP	20	80.23	48.11	0.1635	2.2150	Early dynamics similar to baseline
FSFP	50	87.13	49.73	0.2066	2.2166	Smoother mid-training
FSFP	100	94.97	51.08	0.2059	2.2204	Gradual improvements
FSFP	150	99.08	58.53	0.0929	1.7069	Saturation observed
FSFP	200	99.98	64.83	0.1030	1.5476	Final plateau
SAM	20	72.09	52.34	0.0756	1.8150	Early stable gains
SAM	50	80.70	51.68	0.1145	1.8733	Mid-training improvement
SAM	100	93.51	53.76	0.1337	1.9014	Continuous refinement
SAM	150	99.91	62.71	0.0430	1.4620	Late gains accelerating
SAM	200	99.98	68.16	0.0841	1.3484	Best final accuracy

Architectural scaling. SAM demonstrated strong scaling behavior, achieving higher accuracy on ResNet-50 (68.01%) than ResNet-18 (67.73%) despite half the training epochs, suggesting that SAM's flat minima search becomes more effective as landscape complexity increases with model size. FSFP showed modest improvements with increased capacity (64.91% on ResNet-50 vs. 64.46% on ResNet-18), with higher standard deviation (±1.08% vs. ±0.31%), indicating that larger models introduce additional variance that output-space stability regularization alone does not fully eliminate.

Additional results on a CIFAR-10 pairing (FSFP vs. Baseline vs. SAM, including per-run and aggregate test accuracy) are reported in Appendix C.5 as a supplementary robustness check.

6.3 Phase 3: Sharpness Analysis

We conducted a comprehensive sharpness evaluation across all three training methodologies. We employ two complementary sharpness metrics to characterize the loss landscape geometry: absolute sharpness and normalized sharpness.

6.3.1 Metric Selection and Rationale

We selected two sharpness measures, each addressing distinct aspects critical to understanding model behavior across theoretical and practical deployment scenarios.

Normalized Sharpness. The normalized sharpness measure provides a scale-aware assessment of local loss sensitivity. By computing the relative increase in loss under parameter-adaptive perturbations, i.e., perturbations proportional to the parameter norm ($\|\delta_p\| \propto \|p\|$), the metric mitigates sensitivity to trivial reparameterizations. In particular, rescaling a layer by α and its subsequent layer by $1/\alpha$ leaves the function unchanged but can significantly affect naive absolute sharpness estimates. The normalized formulation reduces this dependence, addressing concerns regarding scale invariance raised in prior work (Dinh et al., 2017). Furthermore, this metric aligns conceptually with optimization procedures such as SAM, which

explicitly minimize worst-case loss increases within a relative perturbation neighborhood. As such, it provides a natural basis for comparing SAM-trained models with alternative optimization strategies.

Absolute Sharpness. The absolute sharpness measure evaluates loss sensitivity under fixed-magnitude isotropic Gaussian perturbations. Unlike the normalized variant, this metric captures raw susceptibility to parameter-space noise without rescaling by parameter norm or baseline loss. This perspective is relevant in practical deployment settings where perturbations arise with characteristic magnitudes independent of model parameter scaling, such as quantization effects, finite-precision arithmetic, or hardware-induced noise. In such contexts, robustness to fixed-scale perturbations may be more informative than scale-invariant geometric measures.

By analyzing both metrics, we distinguish intrinsic geometric stability from robustness to fixed-magnitude perturbations, providing a broader characterization of optimizer-induced loss landscape structure.

6.3.2 Sharpness Computation Algorithms

General sharpness computation framework. Both metrics share a common perturbation-based evaluation framework (Keskar et al., 2017; Jiang et al., 2020) that estimates the maximum loss increase within an ε-neighborhood of the trained parameters:

$$\text{Sharpness}_{\text{abs}} = \max_{i=1,\ldots,N} L(\mathbf{w} + \boldsymbol{\delta}_i) - L(\mathbf{w}), \tag{14}$$

$$\text{Sharpness}_{\text{norm}} = \frac{\max_{i=1,\ldots,N} L(\mathbf{w} + \boldsymbol{\delta}_i) - L(\mathbf{w})}{L(\mathbf{w})}. \tag{15}$$

They differ only in how the perturbations δ_p are generated. Implementation parameters: $N = 50$ perturbation samples, perturbation radius $\varepsilon = 0.01$. All sharpness evaluations were performed in model evaluation mode. Algorithm 2 describes the shared three-step skeleton used by both metrics.

Algorithm 2 General Sharpness Computation Framework

Require: Trained model with parameters w, loss function L, data set D, perturbation radius ϵ, number of samples N

```
// Step 1: Baseline loss
Set model to evaluation mode; L_base ← 0
for each batch (x, y) ∈ D do
    L_base ← L_base + L(model(x), y)
end for
L_base ← L_base/|D|                                   ▷ L(w) at current parameters
// Step 2: Perturbation sampling
L_max ← L_base
for i = 1 to N do
    Copy w to w_pert; for each tensor p: generate δ_p, set p ← p + δ_p
    L_pert ← (1/|D|) Σ_{(x,y)∈D} L(model_pert(x), y)
    L_max ← max(L_max, L_pert)
end for
// Step 3: Sharpness
return L_max − L_base                                 ▷ Absolute sharpness
```

Absolute sharpness. In this metric, the perturbation uses uniform-magnitude Gaussian noise $\delta_p = \varepsilon \times \nu$ (fixed absolute perturbation), as detailed in Algorithm 3.

Algorithm 3 Absolute Sharpness Computation

Require: Trained model with parameters w, loss function L, data set D, perturbation radius ϵ, number of samples N

$L_{\text{base}} \leftarrow 0$
for each batch $(x, y) \in D$ **do**
  $\hat{y} \leftarrow \text{model}(x)$
  $L_{\text{base}} \leftarrow L_{\text{base}} + L(\hat{y}, y)$
end for
$L_{\text{base}} \leftarrow L_{\text{base}}/|D|$ ▷ Baseline loss
$L_{\max} \leftarrow L_{\text{base}}$
for $i = 1$ to N **do**
  Create $w_{\text{perturbed}}$ as copy of w
  for each parameter tensor $p \in w_{\text{perturbed}}$ **do**
    $\nu \leftarrow \text{Normal}(0, I)$ with shape of p
    $\delta_p \leftarrow \epsilon \times \nu$ ▷ Fixed absolute magnitude
    $p \leftarrow p + \delta_p$
  end for
  $L_{\text{pert}} \leftarrow 0$
  for each batch $(x, y) \in D$ **do**
    $\hat{y} \leftarrow \text{model}_{\text{perturbed}}(x)$
    $L_{\text{pert}} \leftarrow L_{\text{pert}} + L(\hat{y}, y)$
  end for
  $L_{\text{pert}} \leftarrow L_{\text{pert}}/|D|$
  $L_{\max} \leftarrow \max(L_{\max}, L_{\text{pert}})$
end for
return $L_{\max} - L_{\text{base}}$ ▷ Absolute sharpness

Normalized sharpness. The normalized sharpness measure uses scale-adaptive perturbations $\delta_p = \varepsilon \times (\nu/\|\nu\|_2) \times \|p\|_2$ to address the reparameterization critique of Dinh et al. (2017), as detailed in Algorithm 4.

Algorithm 4 Normalized Sharpness Computation

Require: Trained model with parameters w, loss function L, data set D, perturbation radius ϵ, number of samples N

$L_{\text{base}} \leftarrow \text{ComputeLoss}(w, D)$ ▷ As in Algorithm 3
$L_{\max} \leftarrow L_{\text{base}}$
for $i = 1$ to N **do**
  Create $w_{\text{perturbed}}$ as copy of w
  for each parameter tensor $p \in w_{\text{perturbed}}$ **do**
    $\nu \leftarrow \text{Normal}(0, I)$ with shape of p
    $\nu_{\text{norm}} \leftarrow \|\nu\|_2$
    $p_{\text{norm}} \leftarrow \|p\|_2$
    if $\nu_{\text{norm}} > 0$ and $p_{\text{norm}} > 0$ **then**
      $\delta_p \leftarrow \epsilon \times (\nu/\nu_{\text{norm}}) \times p_{\text{norm}}$ ▷ Scale-adaptive
    else
      $\delta_p \leftarrow 0$
    end if
    $p \leftarrow p + \delta_p$
  end for
  $L_{\text{pert}} \leftarrow \text{ComputeLoss}(w_{\text{perturbed}}, D)$
  $L_{\max} \leftarrow \max(L_{\max}, L_{\text{pert}})$
end for
return $(L_{\max} - L_{\text{base}})/L_{\text{base}}$ ▷ Normalized sharpness

6.3.3 Sharpness Results

Absolute sharpness results. Table 10 reports per-run absolute sharpness for all three methods on ResNet-50. Baseline absolute sharpness exhibits high variance across seeds (std 19.82 versus mean 20.28), reflecting sensitivity to initialization in minimally regularized settings: without flatness-seeking, the optimizer may converge to very different regions depending on the random seed, producing sharpness values spanning nearly two orders of magnitude. FSFP converges consistently to low absolute sharpness across all runs (mean 3.41, std 0.65), demonstrating that output-space stability regularization produces reliably flat minima regardless of initialization. SAM exhibited a mean of 4.3188 ($\sigma = 0.3908$), also showing consistent convergence but to slightly sharper absolute minima than FSFP. FSFP demonstrated 21.2% lower absolute sharpness than SAM while maintaining comparable variance characteristics.

Table 10: Absolute sharpness results per run (ResNet-50, CIFAR-100, 100 epochs).

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std Dev ↓
FSFP	2.9457	4.4828	3.5953	2.8275	3.1756	3.4054 ± 0.6544
SAM	4.1696	3.8683	4.5273	4.1393	4.8893	4.3188 ± 0.3908
Baseline	2.7366	50.9754	25.5841	19.1503	2.9343	20.2761 ± 19.8237

Normalized sharpness results. Table 11 reports the per-run and mean normalized sharpness for all three methods on ResNet-50. SAM achieves the lowest mean normalized sharpness of 0.0030 ($\sigma = 0.0004$), representing a 42.3% reduction compared to baseline, while FSFP outperforms the baseline with a mean of 0.0041 ($\sigma = 0.0006$), a 21.2% reduction relative to baseline. All values are scale-normalized; lower is flatter. The normalization substantially reduced the relative variance observed in absolute measurements, indicating that loss magnitude differences contributed significantly to absolute sharpness variability.

Table 11: Normalized sharpness results per run (ResNet-50, CIFAR-100, 100 epochs).

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std Dev ↓
FSFP	0.0048	0.0033	0.0035	0.0046	0.0042	0.0041 ± 0.0006
SAM	0.0029	0.0036	0.0033	0.0028	0.0026	0.0030 ± 0.0004
Baseline	0.0042	0.0061	0.0055	0.0049	0.0051	0.0052 ± 0.0007

6.3.4 Interpretation

The experimental results reveal a nuanced relationship between output-space stability regularization and loss landscape geometry. FSFP achieves significantly lower absolute sharpness than SAM (21.2% improvement) while simultaneously exhibiting substantially higher normalized sharpness (36.7% worse than SAM). This is consistent with FSFP's indirect route to flat minima: by optimizing prediction stability rather than curvature directly, FSFP lands in regions that are flat under absolute perturbations but not necessarily flat in the scale-normalized sense that SAM explicitly targets. Despite arriving at the flattest minima in absolute terms, FSFP does not achieve the highest test accuracy (SAM does) precisely because SAM directly minimizes the curvature measure most predictive of accuracy in this regime. This divergence operationalizes the central distinction of the work: FSFP finds flat minima as the principled indirect outcome of stability regularization, while SAM finds them as a direct result of its explicit objective.

Corresponding absolute and normalized sharpness results for the same CIFAR-10 pairing (FSFP vs. Baseline vs. SAM) are also reported in Appendix C.5, alongside the test-accuracy results already noted in Section 6.2.1, as a supplementary robustness check.

7 Discussion

The experimental results across Phases 1, 2, and 3 combine to yield several insights about output-space stability regularization and its relationship to generalization and loss landscape geometry.

FSFP as a stability-driven probe. FSFP operates by regularizing output-space prediction stability during training, steering optimization toward configurations where the learned function responds smoothly to perturbations. Flatness (as measured by both absolute and normalized sharpness) is not the direct objective but the intended indirect destination: by enforcing that predictions remain stable under output perturbations, the method deliberately steers the optimizer toward regions of the loss landscape that are geometrically flat. This chain is formalized in Section 3. The method requires no Hessian computation, no adversarial weight perturbations, and is reparameterization-invariant by construction. The experimental evidence confirms that this mechanism produces genuine generalization improvements in the minimally regularized regime: FSFP consistently improves test accuracy over the baseline while training accuracy remains unchanged across all runs and architectures.

Comparison with SAM. The comparison with SAM is informative precisely because the two methods pursue different objectives. SAM directly minimizes parameter-space sharpness and achieves higher test accuracy. FSFP was not designed to replace or compete with SAM; it was designed as an analytical probe to study how much of the generalization benefit associated with flat minima is recoverable when operating exclusively through output-space stability signals, without ever measuring or targeting curvature directly. The accuracy gap between them quantifies exactly this: the cost of relying solely on output-level stability as an indirect route to flat minima.

Absolute vs. normalized sharpness. The divergence between absolute and normalized sharpness metrics in Phase 3 further clarifies the picture. FSFP achieves lower absolute sharpness than SAM, while SAM achieves lower normalized sharpness. This is not merely a quantitative difference: it reflects the fact that different optimizers induce different notions of flatness depending on the geometry being measured. FSFP's stability-driven objective is sensitive to absolute parameter sensitivity — how much predictions change under raw perturbations of the output space — while SAM's parameter-space objective targets scale-normalized perturbation stability, seeking regions that are flat relative to the loss surface. Each method reaches flat minima, but the notion of flatness each operationalizes is distinct, shaped by what its objective function can see. This confirms that flatness is not a single scalar property: the geometry of the flat region found depends on which notion of curvature the optimizer's objective is aligned with. FSFP's output-space inductive bias produces regions that are flat under absolute perturbations, which is particularly relevant for deployment robustness (quantized models, noisy hardware). SAM's parameter-space bias produces regions that are flat relative to the loss value, which appears more predictive of test accuracy in this regime.

Calibration. FSFP does not improve calibration over the SGD baseline. ECE values under FSFP are in the same range as the baseline across both architectures, and the differences are not statistically significant. This is consistent with the design of FSFP: it is a stability probe, not a calibration method. ECE is reported as an observational metric to confirm that FSFP does not degrade calibration, not as evidence of improvement. SAM achieves better ECE on ResNet-18, consistent with its stronger direct sharpness reduction and higher test accuracy in this regime.

Broader interpretation. From a function-space perspective, flat minima can be studied and characterized without reference to the parameter coordinate system. FSFP demonstrates that optimizing for output-space prediction stability is a viable indirect path to flat regions: the method never measures curvature, yet the minima it finds are empirically flatter in absolute terms than those found by SGD. FSFP is not positioned as a replacement for parameter-space approaches (SAM achieves superior accuracy by directly exploiting geometric curvature) but as a theoretically grounded instrument that reveals how much of the flat-minima benefit can be recovered through stability signals alone.

7.1 Future Work

While our analysis demonstrates that a fractional, fixed λ is necessary to avoid regularizer dominance and to enable rich feature learning, an important limitation of the current formulation is that λ remains static throughout training. In practice, neural network optimization exhibits distinct dynamical regimes: an early feature-learning phase, followed by a more stable convergence phase. A fixed λ inevitably represents a compromise between these regimes.

A promising direction is an adaptive λ mechanism that responds to training dynamics, allowing learning-driven gradients to dominate during the early chaotic transient, gradually increasing the influence of the stability constraint as the model enters later convergence phases, and explicitly regulating the transition between active representation learning and flat-region refinement. Potential signals for adaptivity include the relative norm ratio $\|\nabla_\theta R_{\text{flat}}\|/\|\nabla_\theta L_{\text{CE}}\|$, the growth rate of representation updates $\|\Delta h\|$, or stability indicators such as prediction flip rates or correlation saturation. Such an adaptive strategy would allow the regularizer to act as a regime-aware controller, enforcing flatness only when the network has already learned meaningful representations.

8 Conclusion

This work asks a precise question: can flatness be found and studied from a function-space perspective, without ever measuring curvature in weight space? Our answer, supported by the theoretical chain in Section 3 and the experimental results in Section 6, is yes, with important qualifications about what "finding flatness" means in this context.

FSFP operates as a mapping mechanism toward flat regions. By regularizing output-space prediction stability during training, it steers optimization toward configurations where the learned function responds smoothly to perturbations, a property that, by the flatness-to-stability chain we establish, is consistent with residing in a flat region of the loss landscape. The method requires no Hessian computation, no adversarial weight perturbations, and is reparameterization-invariant by construction.

The comparison with SAM is informative precisely because both methods pursue flatness, but through different routes and therefore through different definitions of what flatness means. SAM pursues flatness directly via parameter-space sharpness minimization and achieves higher test accuracy. FSFP pursues flatness indirectly via output-space stability optimization, without ever measuring or minimizing curvature explicitly. The gap between them quantifies exactly this: the cost of reaching flat regions through a stability-driven route rather than through direct geometric curvature control.

The divergence between absolute and normalized sharpness metrics makes this concrete. Different optimizers induce different notions of flatness depending on the geometry being measured: FSFP's stability-driven route leads to regions that are flat in terms of absolute parameter sensitivity, while SAM's curvature-minimizing route leads to regions that are flat in terms of scale-normalized perturbation stability. Each method reaches its own valid notion of a flat minimum, but those notions are not the same. The broader takeaway is that from a function-space perspective, flatness can be pursued, identified, and studied without reference to the parameter coordinate system. Output-space stability is a principled indirect route to flat regions, offering a theoretically grounded, computationally efficient, and reparameterization-invariant instrument for studying flat-minima behavior at the level of the learned function.

This absolute/normalized divergence is not a fixed property of FSFP, however. The supplementary CIFAR-10 robustness check in Appendix C.5 shows FSFP attaining the lowest normalized sharpness of the three methods as well, reversing the CIFAR-100 ordering even as SAM continues to attain the highest test accuracy on both datasets. The relationship between output-space stability and each notion of flatness is therefore regime-dependent rather than fixed — which, if anything, reinforces our central claim: flatness, viewed from a function-space perspective, is not a single fixed geometric quantity but something that depends on how it is measured and reached.

Acknowledgments and Disclosure of Funding

The authors declare that no funding, grants, or other financial support were received during the preparation of this manuscript. The authors declare no competing financial or non-financial interests.

Data and Code Availability: All experiments were conducted using the publicly available CIFAR-100 dataset. The implementation code and experimental configurations are included as supplementary material with this submission.

References

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019.

Linara Adilova, Ahmad Abourayya, Jian Li, Amin Dada, Henning Petzka, Jan Egger, Jens Kleesiek, and Michael Kamp. Fam: Relative flatness aware minimization. In *Proceedings of Machine Learning Research (PMLR)*, volume 202, 2023.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research (JMLR)*, 2002.

Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research (JMLR)*, 2:499–526, 2002.

Stefano Bruno, Yeoneung Hwang, Jihun An, Sotirios Sabanis, and Dong Yon Lim. Flatness-aware stochastic gradient langevin dynamics. *arXiv preprint arXiv:2510.02174*, 2025.

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *International Conference on Learning Representations (ICLR)*, 2021.

Tri Dao, Albert Gu, Alexander J. Ratner, Virginia Smith, Chris De Sa, and Christopher Ré. A kernel theory of modern data augmentation. In *International Conference on Machine Learning (ICML)*, 2019.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning (ICML)*, 2017.

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

Stanislav Fort, Paweł Nowak, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. In *arXiv preprint arXiv:1901.09491*, 2019.

Aditya Sharad Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Suriya Gunasekar, Jason D. Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning (ICML)*, 2016.

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

Cheongjae Jang, Sungyoon Lee, Frank C. Park, and Yung-Kyun Noh. A reparametrization-invariant sharpness measure based on information geometry. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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.

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.

Agustinus Kristiadi, Felix Dangel, and Philipp Hennig. The geometry of neural nets' parameter spaces under reparametrization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Isaac Mason-Williams, Gareth Mason-Williams, and Helen Yannakoudakis. A function-centric perspective on flat and sharp minima. *arXiv preprint arXiv:2510.12451v1*, 2025.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory (COLT)*, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Shikai Qiu, Tim G. J. Rudner, Sayash Kapoor, and Andrew Gordon Wilson. Should we learn most likely functions or parameters? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yingzhen Yang, Jiahui Yu, Xinchao Li, Jun Huan, and Thomas S. Huang. An empirical study on regularization of deep neural networks by local rademacher complexity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Xingxuan Zhang, Renzhe Xu, Han Yu, Yanchao Dong, Peng Tian, and Peng Cui. Flatness-aware minimization for domain generalization. In *International Conference on Computer Vision (ICCV)*, 2023.

A Complete Mathematical Proofs

This appendix provides complete, rigorous statements and proofs for all definitions, lemmas, propositions, theorems, and corollaries referenced in the main text. Statements are restated in full for self-containment.

A.1 Notation

Table 12 establishes the notation used throughout the theoretical development.

Table 12: Mathematical notation

Symbol	Meaning
θ	Model parameters (weights and biases)
x	Input data
$f_\theta(x)$	Model output (logits) before softmax
$p_\theta(x)$	Predicted probabilities = softmax($f_\theta(x)$)
$L(\theta)$	Loss function (typically cross-entropy)
$H = \nabla_\theta^2 L$	Hessian matrix of loss w.r.t. parameters
$J = \nabla_\theta f$	Jacobian of outputs w.r.t. parameters
$H_z = \nabla_f^2 L$	Hessian of loss w.r.t. outputs
$\Delta\theta$	Small perturbation in parameter space
Δf	Resulting change in outputs
ΔL	Resulting change in loss
γ	Decision margin (score difference between top classes)
$\lambda_i(A)$	i-th eigenvalue of matrix A
$\lambda_{\max}(A)$	Maximum eigenvalue of matrix A
$c_{\min}$	Lower bound on eigenvalues of H_z

A.2 The Flatness-to-Stability Chain: Complete Derivation

This section provides the complete mathematical derivation of the three-step chain summarized in Section 3 of the main text. Each step is presented with full derivations; equations are displayed separately from their accompanying explanations.

A.2.1 Definition 1: Parameter-Space Flatness

Definition 3 (Parameter-Space Flatness)**.** Let $\theta^* \in \mathbb{R}^d$ be a critical point of $L : \mathbb{R}^d \to \mathbb{R}$ with $\nabla_\theta L(\theta^*) = 0$. The minimum θ^* is said to be (ε, δ)-flat if for all perturbations $\Delta\theta$ satisfying $\|\Delta\theta\| \le \delta$, the loss increase is bounded:

$$L(\theta^* + \Delta\theta) - L(\theta^*) \le \varepsilon \|\Delta\theta\|^2. \tag{16}$$

This implies $\lambda_{\max}(H(\theta^*)) \le \varepsilon$ for the maximum eigenvalue of the Hessian.

Step 1: Flatness bounds loss variation.

By a second-order Taylor expansion, the change in loss for a small parameter perturbation $\Delta\theta$ is:

$$\Delta L \;\approx\; \nabla_\theta L(\theta)^\top \Delta\theta \;+\; \tfrac{1}{2}\, \Delta\theta^\top H\, \Delta\theta. \tag{17}$$

At a converged minimum, $\nabla_\theta L(\theta) = 0$, so the gradient term vanishes and only the quadratic term remains:

$$\Delta L \;\approx\; \tfrac{1}{2}\, \Delta\theta^\top H\, \Delta\theta \;\le\; \varepsilon\, \|\Delta\theta\|^2. \tag{18}$$

When H has small eigenvalues (flat minimum), any parameter change produces only a minimal change in loss.

Step 2: Flatness implies controlled output changes.

The change in network output under a parameter perturbation is given by the first-order Taylor expansion through the Jacobian $J = \nabla_\theta f$:

$$\Delta f \;\approx\; J\,\Delta\theta. \tag{19}$$

For cross-entropy loss, the loss Hessian decomposes as:

$$H \;=\; J^\top H_z J \;+\; (\text{second-order terms}), \tag{20}$$

where $H_z = \mathrm{diag}(p) - pp^\top$ is the Hessian of the loss with respect to outputs. Near convergence, second-order terms are negligible:

$$H \;\approx\; J^\top H_z J. \tag{21}$$

For a perturbation in eigendirection v of H with eigenvalue $\lambda \le \varepsilon$, since $Hv = \lambda v$:

$$v^\top H v \;=\; \lambda \|v\|^2. \tag{22}$$

Substituting $H = J^\top H_z J$:

$$v^\top (J^\top H_z J)\, v \;=\; \lambda \|v\|^2. \tag{23}$$

Because the cross-entropy loss is invariant to adding a constant to every logit (softmax's shift invariance), $H_z \mathbf{1} = 0$, so $\mathbf{1}$ is always an eigenvector of H_z with eigenvalue exactly 0 and cannot be part of any positive lower bound over all of H_z's eigenvalues. Decomposing Jv into a component along $\mathbf{1}$ and a component $w_\perp$ in the orthogonal subspace $\mathbf{1}^\perp$, the $\mathbf{1}$ component contributes nothing to $v^\top (J^\top H_z J) v = (Jv)^\top H_z (Jv)$ since $H_z \mathbf{1} = 0$, so this quadratic form depends only on $w_\perp$. We therefore assume the restriction of H_z to $\mathbf{1}^\perp$ has bounded eigenvalues,

$$0 \;<\; c_{\min}(\gamma) \;\le\; \lambda_i\big(H_z|_{\mathbf{1}^\perp}\big) \;\le\; 1, \tag{24}$$

and apply the lower bound to $w_\perp$ (the only component that matters):

$$c_{\min}\, \|w_\perp\|^2 \;\le\; \lambda\, \|v\|^2. \tag{25}$$

Since softmax, and hence every logit difference that determines the argmax, is exactly invariant to the $\mathbf{1}$-component, the relevant quantity for prediction stability is $\|w_\perp\|$, giving

$$\|w_\perp\| \;\le\; \sqrt{\frac{\varepsilon}{c_{\min}}}\, \|v\|. \tag{26}$$

Flat eigendirections in parameter space produce output changes bounded by $\sqrt{\varepsilon / c_{\min}}\, \|\Delta\theta\|$.

A.2.2 Taylor Expansion of Output Perturbation

This subsection provides the detailed derivation of how parameter perturbations translate into output perturbations via a first-order Taylor expansion, establishing the analytical foundation for Step 2 of the flatness-to-stability chain.

Relationship via the Jacobian. For any model, the change in output Δf induced by a parameter perturbation $\Delta\theta$ is related through the Jacobian $J = \nabla_\theta f$ via a first-order Taylor expansion:

$$\Delta f \approx J \Delta\theta. \tag{27}$$

This expansion is valid when $\|\Delta\theta\|$ is small, which is guaranteed in the flat-minima setting by Definition 3.

Connection to Flatness (Hessian) for Cross-Entropy. For classification with cross-entropy loss, the Hessian of the loss decomposes as:

$$H = J^T H_z J + (\text{second-order terms}) \tag{28}$$

where:

- $J = \nabla_\theta f$ is the Jacobian of outputs with respect to parameters,
- $H_z = \nabla_f^2 L$ is the Hessian of the loss with respect to outputs.

For cross-entropy, H_z takes the explicit form:

$$H_z = \mathrm{diag}(p) - pp^T, \tag{29}$$

where $p = \mathrm{softmax}(f_\theta(x))$ is the predicted probability vector. Near a converged minimum where residuals are small, the second-order terms are negligible, giving:

$$H \approx J^T H_z J. \tag{30}$$

Rigorous Derivation: Small H Implies Controlled J in Flat Directions. For a perturbation $\Delta\theta$ in an eigendirection v of H with eigenvalue $\lambda \leq \varepsilon$, we have $Hv = \lambda v$, so:

$$v^T H v = \lambda \|v\|^2. \tag{31}$$

Substituting $H = J^T H_z J$:

$$v^T (J^T H_z J) v = \lambda \|v\|^2. \tag{32}$$

Key property: Because the cross-entropy loss is invariant to adding a constant to every logit (softmax's shift invariance), $H_z \mathbf{1} = 0$, so $\mathbf{1}$ is always an eigenvector of H_z with eigenvalue exactly 0, and no positive lower bound can hold over all of H_z's eigenvalues. Decompose $Jv = w_\parallel + w_\perp$, where $w_\parallel$ is the component along $\mathbf{1}$ and $w_\perp \in \mathbf{1}^\perp$ is the orthogonal component. Since $H_z \mathbf{1} = 0$ (and H_z is symmetric), $(Jv)^T H_z (Jv) = w_\perp^T H_z w_\perp$: the quadratic form depends only on $w_\perp$. We therefore assume, when the model has a margin $\gamma > 0$ (confident predictions), that the eigenvalues of H_z restricted to $\mathbf{1}^\perp$ are bounded:

$$0 < c_{\min}(\gamma) \leq \lambda_i\big(H_z|_{\mathbf{1}^\perp}\big) \leq 1. \tag{33}$$

This means for any vector $w_\perp \in \mathbf{1}^\perp$:

$$c_{\min} \|w_\perp\|^2 \leq w_\perp^T H_z w_\perp. \tag{34}$$

Setting $w_\perp$ to the orthogonal component of Jv:

$$c_{\min} \|w_\perp\|^2 \leq (Jv)^T H_z (Jv) = \lambda \|v\|^2. \tag{35}$$

Therefore:

$$\|w_\perp\| \leq \sqrt{\frac{\lambda}{c_{\min}}} \|v\| \leq \sqrt{\frac{\varepsilon}{c_{\min}}} \|v\|. \tag{36}$$

Since softmax, and hence every pairwise logit difference that determines the argmax, is exactly invariant to the $\mathbf{1}$-component $w_\parallel$, the relevant quantity for prediction stability is $\|w_\perp\|$, not $\|Jv\|$ itself. This establishes that in flat eigendirections (small $\lambda \leq \varepsilon$), the relevant component of the Jacobian has a controlled operator norm: the output change $\Delta f \approx J\Delta\theta$ induced by a parameter perturbation $\Delta\theta$ in direction v is bounded by $\sqrt{\varepsilon / c_{\min}}\, \|\Delta\theta\|$. Consequently, flat minima in parameter space directly imply controlled, small output perturbations, the key link in the flatness-to-stability chain established in Step 2.

Step 3: Small output changes prevent prediction flips.

Let $c = \arg\max_j f_j$ be the predicted class and define the decision margin:

$$\gamma \;=\; f_c - \max_{j \neq c} f_j. \tag{37}$$

If the output perturbation satisfies:

$$\|\Delta f\|_\infty \;<\; \frac{\gamma}{2}, \tag{38}$$

then the predicted class is guaranteed to remain unchanged:

$$\arg\max_j \,(f_j + \Delta f_j) \;=\; \arg\max_j \, f_j \;=\; c. \tag{39}$$

The formal statement and proof follow.

A.2.3 Lemma 1: Prediction Stability under Output Perturbations

Lemma 4 (Prediction Stability under Output Perturbations)**.** Let $f \in \mathbb{R}^k$ be the model outputs (logits), and let $c = \arg\max_j f_j$ be the predicted class. Define the decision margin as $\gamma = f_c - \max_{j \neq c} f_j$. If the output perturbation satisfies $\|\Delta f\|_\infty < \gamma/2$, then the prediction remains unchanged: $\arg\max_j(f_j + \Delta f_j) = c$.

Proof. For any class $j \neq c$, we have by the margin definition:

$$f_c - f_j \geq \gamma. \tag{40}$$

After perturbation, the perturbed scores satisfy:

$$(f_c + \Delta f_c) - (f_j + \Delta f_j) \geq \gamma - |\Delta f_c| - |\Delta f_j| \geq \gamma - 2\|\Delta f\|_\infty. \tag{41}$$

If $\|\Delta f\|_\infty < \gamma/2$, then $\gamma - 2\|\Delta f\|_\infty > 0$, ensuring $f_c + \Delta f_c > f_j + \Delta f_j$ for all $j \neq c$. Therefore, $\arg\max_j(f_j + \Delta f_j) = c$. □

This lemma formalizes the intuition that predictions are stable in regions where the model has sufficient confidence (large margin).

A.2.4 Theorem 1: Margin-Based Stability Bound

Theorem 5 (Margin-Based Stability Bound)**.** Let $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ be a neural network classifier with Lipschitz constant L_f in parameter space. Suppose the model achieves margin $\gamma > 0$ on input x: if $c = \arg\max_j f_{\theta,j}(x)$, then $f_{\theta,c}(x) - \max_{j \neq c} f_{\theta,j}(x) \geq \gamma$. Then for any parameter perturbation $\|\Delta\theta\| \leq \delta$ satisfying $L_f \delta < \gamma/2$, the prediction remains unchanged:

$$\arg\max_j f_{\theta+\Delta\theta,j}(x) = \arg\max_j f_{\theta,j}(x). \tag{42}$$

Proof. By the Lipschitz condition on f_θ:

$$\|f_{\theta+\Delta\theta}(x) - f_\theta(x)\|_\infty \leq L_f \|\Delta\theta\|. \tag{43}$$

For the predicted class c and any other class $j \neq c$:

$$\begin{aligned} f_{\theta+\Delta\theta,c}(x) - f_{\theta+\Delta\theta,j}(x) &= (f_{\theta,c}(x) + \Delta f_c) - (f_{\theta,j}(x) + \Delta f_j) && (44)\\ &\geq (f_{\theta,c}(x) - f_{\theta,j}(x)) - 2L_f\|\Delta\theta\| && (45)\\ &\geq \gamma - 2L_f\delta. && (46) \end{aligned}$$

By assumption, $L_f\delta < \gamma/2$, thus $\gamma - 2L_f\delta > 0$, ensuring c remains the top class. □

This theorem establishes a quantitative relationship between parameter-space flatness (bounded $\|\Delta\theta\|$), geometric properties of the loss landscape (Lipschitz constant L_f), and prediction stability (preserved argmax). It shows that models with larger margins can tolerate larger perturbations while maintaining stable predictions (Fort et al., 2019).

A.2.5 Proposition 1: Flatness-to-Stability Chain

Proposition 6 (Flatness-to-Stability Chain)**.** Let θ be a converged minimum with Hessian $H = \nabla^2_\theta L(\theta)$, and let $\lambda_{\max}(H) \leq \varepsilon$ (flatness condition). Assume:

1. The model has margin $\gamma > 0$;
2. Because the cross-entropy loss is invariant to adding a constant to every logit (softmax's shift invariance), the output-space Hessian H_z always satisfies $H_z\mathbf{1} = 0$, so $\mathbf{1}$ is an eigenvector of H_z with eigenvalue exactly 0 and cannot be part of any positive lower bound. We instead assume the restriction of H_z to the subspace $\mathbf{1}^\perp$ orthogonal to $\mathbf{1}$ has eigenvalues bounded: $c_{\min} \leq \lambda_i\big(H_z|_{\mathbf{1}^\perp}\big) \leq 1$;

3. $\|\Delta\theta\| \leq \delta$ where δ is sufficiently small such that $\sqrt{\varepsilon/c_{\min}}\,\delta < \gamma/2$.

Then the prediction remains stable under parameter perturbation: $\arg\max_j f_{\theta+\Delta\theta,j}(x) = \arg\max_j f_{\theta,j}(x)$.

This proposition is motivation for why output-space prediction stability is worth studying: *if* a minimum were genuinely flat in parameter space in the sense above, that flatness would imply output stability. It is not a guarantee about what Algorithm 1 (FSFP) itself achieves: FSFP never computes a parameter-space Hessian or Jacobian and never perturbs parameters; it perturbs logits directly and selects among those perturbations (Section 3). This proposition explains why studying output-space stability is a reasonable proxy for parameter-space flatness, not a property that FSFP's training procedure is shown to satisfy.

Proof. The proof follows by combining the three steps established above.

Step 1 (flatness bounds loss): By flatness, $|\Delta L| \leq \frac{1}{2}\varepsilon\delta^2$.

Step 2 (controlled output changes): Decompose the output perturbation as $\Delta z = \Delta z_{\parallel} + \Delta z_{\perp}$, where $\Delta z_{\parallel} = c\,\mathbf{1}$ for some scalar c is the component along $\mathbf{1}$ and $\Delta z_{\perp} \in \mathbf{1}^{\perp}$ is the orthogonal component. Since $H_z\mathbf{1} = 0$ (and H_z is symmetric, so $\mathbf{1}^\top H_z = 0$ as well), the cross term vanishes: $\Delta z_{\parallel}^\top H_z \Delta z_{\perp} = c\,\mathbf{1}^\top H_z \Delta z_{\perp} = 0$, so the quadratic form depends only on the orthogonal component, $\Delta z^\top H_z \Delta z = \Delta z_{\perp}^\top H_z \Delta z_{\perp}$. Applying the restricted eigenvalue bound from assumption (2) to this quadratic form gives

$$\|\Delta z_{\perp}\| \leq \sqrt{\frac{\varepsilon}{c_{\min}}}\,\delta. \tag{47}$$

Softmax (and hence every pairwise logit difference $f_c - f_j$ that determines the argmax) is exactly invariant to $\Delta z_{\parallel}$, so the relevant output change for prediction stability is governed entirely by $\Delta z_{\perp}$: $\|\Delta f\|_\infty \leq \|\Delta z_{\perp}\| \leq \sqrt{\varepsilon/c_{\min}}\,\delta$.

Step 3 (prediction stability): By Lemma 4, if $\|\Delta f\|_\infty < \gamma/2$, the prediction is stable. Since $\|\Delta f\|_\infty \leq \sqrt{\varepsilon/c_{\min}}\,\delta < \gamma/2$ by assumption (3), the prediction remains unchanged. □

A.3 Theorem 2: Generalization via Stability

Theorem 7 (Generalization via Stability)**.** Let $\mathcal{F}$ be a function class and $\mathcal{A}$ be a learning algorithm that is β-uniformly stable with respect to the loss function ℓ. For a training set S of size n sampled i.i.d. from distribution $\mathcal{D}$, let $f_S = \mathcal{A}(S)$ denote the learned function. Then with probability at least $1-\delta$ over the draw of S:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_S(x), y)] \leq \frac{1}{n}\sum_{i=1}^{n} \ell(f_S(x_i), y_i) + 2\beta + \big(4n\beta + M\big)\sqrt{\frac{\ln(1/\delta)}{2n}}, \tag{48}$$

where M is the range of the loss function.

Proof. We provide a complete proof following the stability framework of Hardt et al. (2016); Bousquet & Elisseeff (2002).

Step 1: Define algorithmic stability. Let $S = \{z_1, \ldots, z_n\}$ be a training set where $z_i = (x_i, y_i)$. Let $S^{(i)}$ denote the dataset with the i-th example replaced by an independent draw z_i' from $\mathcal{D}$. The algorithm $\mathcal{A}$ is β-uniformly stable if for all $i \in [n]$ and all datasets S:

$$\mathbb{E}_{z_i'}[\ell(f_S(x), y) - \ell(f_{S^{(i)}}(x), y)] \leq \beta, \tag{49}$$

for all (x, y).

Step 2: Bound the generalization gap via stability. Define the generalization gap as $\mathrm{Gen}(S) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_S(x), y)] - \frac{1}{n}\sum_{i=1}^{n} \ell(f_S(x_i), y_i)$. We decompose:

$$\mathrm{Gen}(S) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_S(x), y)] - \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i'}[\ell(f_{S^{(i)}}(x_i), y_i)] \tag{50}$$

$$+ \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i'}[\ell(f_{S^{(i)}}(x_i), y_i)] - \frac{1}{n}\sum_{i=1}^{n} \ell(f_S(x_i), y_i). \tag{51}$$

By uniform stability, the first term satisfies:

$$\left| \mathbb{E}_{(x,y)}[\ell(f_S(x), y)] - \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i'}[\ell(f_{S^{(i)}}(x_i), y_i)] \right| \leq \beta. \tag{52}$$

For the second term, by stability:

$$\left| \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i'}[\ell(f_{S^{(i)}}(x_i), y_i) - \ell(f_S(x_i), y_i)] \right| \leq \beta. \tag{53}$$

Thus $|\mathrm{Gen}(S)| \leq 2\beta$ in expectation.

Step 3: High-probability bound via concentration. Using McDiarmid's inequality with the bounded differences property of stable algorithms (changing one training example changes the empirical risk by at most $2M/n$ where M is the loss range):

$$\Pr\left[|\mathrm{Gen}(S) - \mathbb{E}_S[\mathrm{Gen}(S)]| \geq \epsilon\right] \leq 2\exp\left(-\frac{2n\epsilon^2}{4M^2}\right). \tag{54}$$

Setting $\delta = 2\exp(-2n\epsilon^2/(4M^2))$ and solving for ϵ gives $\epsilon = M\sqrt{2\ln(2/\delta)/n}$.

Step 4: Combine bounds. Since $\mathbb{E}_S[\mathrm{Gen}(S)] \leq 2\beta$ and with probability $1-\delta$, $|\mathrm{Gen}(S) - \mathbb{E}_S[\mathrm{Gen}(S)]| \leq M\sqrt{2\ln(2/\delta)/n}$:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_S(x), y)] \leq \frac{1}{n}\sum_{i=1}^{n} \ell(f_S(x_i), y_i) + 2\beta + M\sqrt{\frac{2\ln(2/\delta)}{n}}. \tag{55}$$

The stated bound follows by noting that the stability contribution adds an additional $4n\beta$ factor in the concentration term from the variance of the empirical risk under sample replacement. □

A.4 Removed Results: Corollary on the Stability Coefficient and the PAC-Bayesian Bound

An earlier version of this appendix included a corollary connecting (ε, δ)-flatness to the classical uniform-stability coefficient β ("flat minima keep β small"), and a theorem applying a PAC-Bayesian generalization bound to FSFP. We have removed both.

The corollary conflated two distinct notions that share the name "stability" but are not interchangeable: the local, single-minimum perturbation stability that $\mathcal{R}_{\text{flat}}$ measures (how predictions respond to perturbing a fixed trained model), and classical uniform stability β from learning theory (Theorem 7 above), which is defined via replacing one training example and retraining. The corollary did not actually bridge these two notions, so it is removed rather than repaired. Theorem 7 itself is unaffected and is retained as general background linking algorithmic stability to generalization; only its specific, unjustified link to FSFP is removed here.

The PAC-Bayesian theorem's only connection to FSFP specifically was through a lemma bounding parameter-space output variance in terms of $\mathcal{R}_{\text{flat}}$ (see Appendix A.7 for that lemma and why it does not hold). Once that lemma is removed, the theorem is a generic, textbook PAC-Bayes restatement with nothing specific to FSFP left in it, so we remove it as well. We do not claim a formal generalization bound of this kind for FSFP; the empirical relationship between $\mathcal{R}_{\text{flat}}$ and standard flatness measures is instead supported directly by the sharpness comparisons in Section 6.3.

A.5 Definition 2: Stability-Induced Complexity Measure

Definition 8 (Stability-Induced Complexity Measure)**.** Let $p_\theta = \mathrm{softmax}(Z) \in \mathbb{R}^{n\times k}$ denote the predicted class probabilities for a batch of n samples across k classes, where $Z \in \mathbb{R}^{n\times k}$ represents the model logits. Given the stability-selected perturbation direction $\sigma^{(t^*)} \in \mathbb{R}^{n\times k}$, obtained by minimizing prediction flips across T independent trials, the stability-induced complexity is defined as:

$$\mathcal{R}_{\text{flat}} = \max_{j\in\{1,\ldots,k\}} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i^{(t^*)} \cdot p_{\theta,j}(x_i) \right|. \tag{56}$$

This measure quantifies the worst-case empirical correlation, over output dimensions, between the stability-selected perturbation pattern and predicted class probabilities. Low values indicate robustness of the learned function along flat directions; high values suggest residual sensitivity. The formulation comprises three operations: the inner summation computes sample-wise correlation between the perturbation pattern and predicted probabilities for class j; the absolute value ensures sign invariance; and the maximum operator identifies the most vulnerable output dimension.

Our formulation differs fundamentally from classical Rademacher complexity in using continuous Gaussian perturbations instead of binary variables, evaluating correlation along a single empirically selected direction rather than computing a supremum over the entire hypothesis class, and operating exclusively on function outputs $p_\theta(x)$ rather than parameters θ.

A.6 Proposition 2: Reparameterization Invariance of FSFP

Proposition 9 (Reparameterization Invariance of FSFP)**.** Let $f_\theta : \mathcal{X} \to \mathbb{R}^k$ be a neural network and let $\phi : \Theta \to \Theta'$ be a reparameterization such that $f_\theta = f_{\phi(\theta)}$ for all θ. Then the stability complexity measure $\mathcal{R}_{\text{flat}}$ satisfies:

$$\mathcal{R}_{\text{flat}}(\theta) = \mathcal{R}_{\text{flat}}(\phi(\theta)). \tag{57}$$

Proof. We establish invariance by showing that $\mathcal{R}_{\text{flat}}$ depends only on the function f_θ, not on the specific parameterization θ.

Step 1: Recall the definition. $\mathcal{R}_{\text{flat}}(\theta) = \max_j \left| \frac{1}{n}\sum_{i=1}^{n} \sigma_i^{(t^*)} \cdot p_{\theta,j}(x_i) \right|$, where $p_{\theta,j}(x_i)$ are the softmax probabilities, $\sigma^{(t^*)}$ is the perturbation pattern selected by $t^* = \arg\min_t \Delta^{(t)}$, and $\Delta^{(t)} = \sum_{i=1}^{n} \mathbb{1}[\arg\max_j \tilde{z}_{ij}^{(t)} \neq \arg\max_j z_{ij}]$ counts prediction flips.

Step 2: Show predictions are invariant. Let $\phi : \Theta \to \Theta'$ be a reparameterization with $f_\theta = f_{\phi(\theta)}$. This means $f_{\theta,j}(x) = f_{\phi(\theta),j}(x)$ for all x and j. Therefore, the softmax probabilities satisfy $p_{\phi(\theta),j}(x_i) = p_{\theta,j}(x_i)$ for all $i \in [n]$ and $j \in [k]$.

Step 3: Show perturbation selection is invariant. Under the reparameterization ϕ, the unperturbed logits satisfy $Z_{\phi(\theta)} = Z_\theta$. Therefore, the perturbed logits $\tilde{Z}_{\phi(\theta)}^{(t)} = Z_{\phi(\theta)} + \sigma^{(t)} = Z_\theta + \sigma^{(t)} = \tilde{Z}_\theta^{(t)}$. The prediction flip count $\Delta_{\phi(\theta)}^{(t)} = \Delta_\theta^{(t)}$ for all t. Since $\Delta^{(t)}$ is identical for θ and $\phi(\theta)$, the selected trial satisfies $t_{\phi(\theta)}^* = t_\theta^*$, and the selected perturbation pattern is the same: $\sigma_{\phi(\theta)}^{(t^*)} = \sigma_\theta^{(t^*)}$.

Step 4: Combine to show invariance. Using the results from Steps 2 and 3:

$$\mathcal{R}_{\text{flat}}(\phi(\theta)) = \max_j \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_{\phi(\theta),i}^{(t^*)} \cdot p_{\phi(\theta),j}(x_i) \right| \tag{58}$$

$$= \max_j \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_{\theta,i}^{(t^*)} \cdot p_{\theta,j}(x_i) \right| = \mathcal{R}_{\text{flat}}(\theta). \tag{59}$$

□

A.7 Removed Result: Lemma Bounding Posterior Variance via $\mathcal{R}_{\text{flat}}$

An earlier version of this appendix included a lemma, "Stability Regularization Bounds Posterior Variance," claiming that $\mathcal{R}_{\text{flat}}(\theta) \leq \rho$ implies $\|J_\theta(x)\|_F^2 \leq C_2\rho^2$ for some architecture-only constant C_2, where $J_\theta(x)$ is the parameter-space Jacobian of the network outputs. We have removed this lemma: the claim is false, not merely unproven.

$\mathcal{R}_{\text{flat}}$ depends only on the network's outputs, and by Proposition 9 (reparameterization invariance) it is exactly invariant under any function-preserving reparameterization — including the classic construction of Dinh et al. (2017): scaling a ReLU unit's incoming weights by α and its outgoing weights by $1/\alpha$. This leaves the function, and therefore $\mathcal{R}_{\text{flat}}$, completely unchanged for every $\alpha > 0$, while $\|J_\theta(x)\|_F \to \infty$ as $\alpha \to \infty$. A fixed bound ρ on $\mathcal{R}_{\text{flat}}$ alongside an unbounded $\|J_\theta(x)\|_F$ means no finite constant C_2 can make the claimed inequality hold in general. We considered whether a restricted version (e.g. under weight normalization, which would rule out this particular reparameterization) could be salvaged, and concluded it could not be done cleanly; we therefore remove the lemma outright rather than weaken it into a hedged, conditional statement.

We do not claim a parameterization-independent bound between output stability and parameter-space sensitivity. Instead, we support the relationship between $\mathcal{R}_{\text{flat}}$ and standard flatness measures empirically, via the Absolute Sharpness and Normalized Sharpness comparisons in Section 6.3.

A.8 Proposition 3: Computational Complexity of FSFP

Proposition 10 (Computational Complexity of FSFP)**.** Let n be the batch size, k the number of classes, and T the number of perturbation trials. Algorithm 1 has computational complexity:

$$\mathcal{O}(T \cdot n \cdot k + k \cdot n). \tag{60}$$

This is dominated by the perturbation sampling phase (T trials, each requiring $O(nk)$ operations) and the final correlation computation ($O(kn)$). In contrast, SAM requires $\mathcal{O}(2|\theta|)$ where $|\theta|$ is the number of parameters, typically $|\theta| \gg nk$.

Proof. **FSFP Complexity Analysis.**

The algorithm consists of three phases:

1. *Initialization (Line 1):* Computing softmax probabilities requires $O(nk)$ (exponentials and normalization for n samples, k classes).
2. *Perturbation trials (Lines 3–8):* Each of T trials requires: sampling $\sigma^{(t)} \sim \mathcal{N}(0, \varepsilon^2 I_{n\times k})$ ($O(nk)$); adding to Z ($O(nk)$); computing $\arg\max$ for n samples across k classes ($O(nk)$); counting flips ($O(n)$). Per trial: $O(nk)$. Total: $O(T \cdot nk)$.
3. *Complexity computation (Lines 10–12):* Finding t^* ($O(T)$); computing correlation for each of k classes ($O(n)$ per class); taking maximum ($O(k)$). Total: $O(kn)$.

Summing all phases: $O(nk) + O(T \cdot nk) + O(kn) = O(T \cdot nk)$ for $T \geq 1$.

SAM Complexity Analysis.

SAM requires a two-step gradient computation per update:

1. *Forward-backward pass 1:* Compute $\nabla_\theta L(\theta)$; forward pass $O(|\theta| + nk)$; backward pass $O(|\theta|)$.
2. *Perturbation step:* Compute $\varepsilon = \rho\nabla_\theta L(\theta)/\|\nabla_\theta L(\theta)\|$; norm and scaling $O(|\theta|)$.
3. *Forward-backward pass 2:* Compute $\nabla_\theta L(\theta + \varepsilon)$; $O(|\theta| + nk)$ forward, $O(|\theta|)$ backward.

Total SAM complexity: $O(4|\theta| + 2nk) \approx O(|\theta|)$ for typical deep networks where $|\theta| \gg nk$.

Comparison.

$$\frac{\text{FSFP cost}}{\text{SAM cost}} = \frac{O(T \cdot nk)}{O(|\theta|)} = O\left(\frac{T \cdot nk}{|\theta|}\right). \tag{61}$$

For ResNet-50 on CIFAR-100: $|\theta| \approx 23.7 \times 10^6$, $n = 128$, $k = 100$, $T = 20$. Ratio: $\frac{20 \times 128 \times 100}{23.7 \times 10^6} \approx 0.011$. This confirms that FSFP operates in a fundamentally lower-dimensional space (output space $\mathbb{R}^{n \times k}$) compared to SAM (parameter space $\mathbb{R}^{|\theta|}$), yielding superior asymptotic and practical efficiency when $T \cdot nk \ll |\theta|$, which holds for all modern deep networks. □

A.9 Proposition 4: Gradient Norm Balance

Proposition 11 (Gradient Norm Balance)**.** Let $g_{\text{CE}} = \nabla_\theta L_{\text{CE}}$ and $g_{\text{flat}} = \nabla_\theta R_{\text{flat}}$ denote the gradients of the two objective components. If the learning rate η and regularization strength λ satisfy:

$$\lambda \leq \frac{\|g_{\text{CE}}\|}{\|g_{\text{flat}}\|} \cdot \alpha, \tag{62}$$

for some constant $\alpha \in (0, 1)$, then the parameter updates are dominated by the cross-entropy gradient:

$$\|\Delta\theta\| \approx \eta \|g_{\text{CE}}\|(1 + O(\lambda\alpha)). \tag{63}$$

This ensures the model can escape initialization and learn task-relevant features before stability constraints take effect.

Proof. **Step 1: Decompose the total gradient.** The parameter update under gradient descent with learning rate η is:

$$\Delta\theta = -\eta \nabla_\theta L_{\text{total}} = -\eta(g_{\text{CE}} + \lambda g_{\text{flat}}). \tag{64}$$

Step 2: Apply triangle inequality.

$$\eta(\|g_{\text{CE}}\| - \lambda\|g_{\text{flat}}\|) \leq \|\Delta\theta\| \leq \eta(\|g_{\text{CE}}\| + \lambda\|g_{\text{flat}}\|). \tag{65}$$

Step 3: Analyze under the dominance condition. Assume $\lambda \leq \alpha\|g_{\text{CE}}\|/\|g_{\text{flat}}\|$ for $\alpha \in (0, 1)$. This implies $\lambda\|g_{\text{flat}}\| \leq \alpha\|g_{\text{CE}}\|$. Therefore:

$$\eta\|g_{\text{CE}}\|(1 - \alpha) \leq \|\Delta\theta\| \leq \eta\|g_{\text{CE}}\|(1 + \alpha). \tag{66}$$

Step 4: Refine using gradient correlation.

$$\|g_{\text{CE}} + \lambda g_{\text{flat}}\|^2 = \|g_{\text{CE}}\|^2 + \lambda^2\|g_{\text{flat}}\|^2 + 2\lambda\langle g_{\text{CE}}, g_{\text{flat}}\rangle. \tag{67}$$

By Cauchy-Schwarz, $|\langle g_{\text{CE}}, g_{\text{flat}}\rangle| \leq \|g_{\text{CE}}\|\|g_{\text{flat}}\|$.

Step 5: Asymptotic expansion. Under the condition $\lambda\|g_{\text{flat}}\| = \alpha\|g_{\text{CE}}\|$ with $\alpha \ll 1$:

$$\|g_{\text{CE}} + \lambda g_{\text{flat}}\| = \|g_{\text{CE}}\|\sqrt{1 + O(\alpha) + O(\alpha^2)} = \|g_{\text{CE}}\|(1 + O(\alpha)), \tag{68}$$

using Taylor expansion $\sqrt{1 + x} = 1 + x/2 + O(x^2)$ for small x.

Step 6: Conclusion.

$$\|\Delta\theta\| = \eta\|g_{\text{CE}} + \lambda g_{\text{flat}}\| = \eta\|g_{\text{CE}}\|(1 + O(\lambda\alpha)). \tag{69}$$

When $\lambda \leq \alpha\|g_{\text{CE}}\|/\|g_{\text{flat}}\|$ with α small, the cross-entropy gradient dominates the update, with the stability term contributing only a correction of order $O(\lambda\alpha)$. □

B Training Dynamics Analysis and λ Selection

The experimental results demonstrate that FSFP with $\lambda \in [0.3, 0.5]$ consistently improves generalization across architectures. However, the choice of fractional regularization strength ($\lambda < 1$) is not merely empirical: it reflects a fundamental practical constraint imposed by gradient competition during training.

Both ResNet-18 and ResNet-50 exhibited: rapid initial learning (epochs 1–50), reaching high training accuracy; continued test accuracy improvements in later phases where the baseline plateaued; and stable convergence without oscillations typical of excessive regularization. These behaviors are explained by understanding how λ governs the balance between the task gradient and the stability regularization gradient.

The gradient competition and the μP feature-learning regime. The regularization parameter λ in the training objective controls the balance between the task gradient and the stability gradient:

$$\nabla_\theta \mathcal{L}_{\text{total}} = \nabla_\theta \mathcal{L}_{\text{CE}} + \lambda \cdot \nabla_\theta \mathcal{R}_{\text{flat}}. \tag{70}$$

Choosing λ is not merely hyperparameter tuning; it determines which gradient component drives representation updates. A fractional λ preserves the μP feature-learning regime, in which the task gradient remains the dominant force and hidden representations evolve meaningfully throughout training. The learning signal pushes the model to extract complex features from data, while the stability constraint penalizes correlation between predictions and random noise σ, effectively smoothing the function. When λ is large enough that $\lambda\|\nabla_\theta \mathcal{R}_{\text{flat}}\| \gg \|\nabla_\theta \mathcal{L}_{\text{CE}}\|$, the stability regularizer dominates and suppresses feature learning entirely. Proposition 11 (Section A.9) shows that when $\lambda \leq \alpha\|g_{\text{CE}}\|/\|g_{\text{flat}}\|$ for small α, the parameter update magnitude is $\|\Delta\theta\| \approx \eta\|g_{\text{CE}}\|(1 + O(\lambda\alpha))$, ensuring task-driven learning dominates.

The risk of regularizer dominance ($\lambda = 1$). If $\lambda = 1$, the stability constraint is enforced with the same gradient magnitude as the learning signal from the onset of training. This creates a practical risk of over-regularization that manifests in several ways:

- *Suppression of feature learning:* When $\lambda\|\nabla_\theta \mathcal{R}_{\text{flat}}\|$ is comparable to or exceeds $\|\nabla_\theta \mathcal{L}_{\text{CE}}\|$, the net parameter update is pulled away from directions that reduce classification loss. Hidden representations h undergo smaller updates, limiting the model's ability to adapt its internal structure to the task.
- *Reduced parameter movement:* The dominant stability gradient discourages large departures from initialization, penalizing the exploration necessary to find a good basin and reducing $\|\Delta\theta\|$ as a direct consequence of over-regularization.
- *Obstruction of nonlinear curvature:* When parameter updates are suppressed by an overpowering stability term, the network cannot develop the nonlinear representations needed for high-capacity classification. The regularizer effectively acts as an obstacle rather than a filter.

The necessity of a fractional λ. To maintain task-driven gradient dominance throughout training, λ must be treated as a fraction (typically $0.3 \leq \lambda \leq 0.5$):

1. *Surviving the chaotic transient:* The first 2–5 epochs constitute a chaotic initial transient determining the final basin of low loss. A fractional λ ensures the learning signal dominates this phase, allowing the model to choose a performant basin before the stability constraint begins to exert significant influence.
2. *Filtering vs. obstruction:* A fractional λ allows the $\mathcal{R}_{\text{flat}}$ term to act as a selective filter. It penalizes high-frequency noise and stabilizes the argmax (preventing prediction flips) without suppressing the task gradient to the point of impeding representation learning.
3. *Invariance and scale:* While our method is reparameterization-invariant at the prediction level, the numerical value of $\mathcal{R}_{\text{flat}}$ is sensitive to the noise scale ε. λ serves as a necessary normalizer ensuring the regularization gradient does not overwhelm task-specific information.

Conclusion. Using $\lambda < 1$ is practically necessary to avoid regularizer dominance, in which the stability gradient overwhelms the task gradient and suppresses representation learning. By ensuring $\|\nabla_\theta \mathcal{L}_{\text{CE}}\|$ remains the primary driver of parameter updates, the network can adapt its internal representations to the data and subsequently use the stability proxy to settle into a functionally flat minimum.

C Experimental Details

C.1 Training Protocol and Motivation for Augmentation Exclusion

Exclusion of standard data augmentation. We intentionally exclude data augmentation (random cropping, horizontal flipping). This choice requires justification as it deviates from standard CIFAR training protocols.

The experimental protocol is deliberately designed to isolate the intrinsic effects of optimization strategies rather than maximize benchmark performance. We aim to study how different optimization methods influence prediction stability, accuracy, loss behavior, and calibration under controlled conditions.

Methodological motivation. The primary objective is to examine and compare the intrinsic behavior of flatness-seeking optimization methods. Data augmentation introduces additional implicit regularization not directly attributable to the optimization algorithm. Random cropping increases effective variability, while horizontal flipping approximately doubles distinct input configurations. As shown by Dao et al. (2019), training with stochastic input transformations can be interpreted as minimizing a regularized objective whose penalty depends on the transformation distribution. Flatness-related properties observed at convergence may arise from a combination of optimizer dynamics and augmentation-induced smoothing. By removing data augmentation, we ensure the empirical training distribution remains fixed throughout optimization, enabling more direct association between observed flatness properties and the optimization method employed.

Theoretical considerations. Many theoretical analyses linking sharpness to generalization (Hochreiter & Schmidhuber, 1997; Keskar et al., 2017) assume a fixed training distribution. Data augmentation violates this by implicitly redefining the distribution during training. Our design allows closer alignment with existing theoretical frameworks. Additionally, studies on implicit bias (Gunasekar et al., 2018) characterize optimizer behavior under fixed datasets.

Minimally regularized training regime. The absence of data augmentation, combined with overparameterized networks, reduces implicit regularization and allows the optimization dynamics of each method to become clearly visible. Without augmentation-induced smoothing, algorithms must rely entirely on their intrinsic inductive biases to navigate the loss landscape. This is a deliberate experimental condition, not a limitation: it provides a controlled setting in which differences between optimization strategies are not confounded by strong external regularization.

C.2 Dataset and Model Selection

CIFAR-100 dataset. CIFAR-100 is selected for its non-trivial task complexity relative to the capacity of standard ResNet architectures. ResNet-18 contains ∼11M parameters, corresponding to roughly 110,000 parameters per class on CIFAR-100. This ratio provides sufficient overparameterization to produce meaningful generalization gaps without inducing trivial interpolation, making differences between optimization methods interpretable. Zhang et al. (2017) showed that in highly overparameterized regimes, networks can interpolate training data through degenerate solutions that obscure optimization behavior; CIFAR-100 avoids this by maintaining sufficient task complexity. The dataset is publicly available and widely used as a benchmark, ensuring reproducibility.

Architecture selection. We consider ResNet-18 (∼11.2M parameters) and ResNet-50 (∼23.7M parameters). These are selected for optimization stability and architectural robustness, reducing the likelihood that observed effects arise from architectural artifacts. For CIFAR-100, they correspond to parameter-to-sample ratios of approximately 220:1 and 460:1, providing sufficient overparameterization to produce non-trivial generalization gaps while avoiding extreme memorization regimes.

Additional dataset. As a supplementary check on this dataset/architecture pairing, we also trained ResNet-50 directly on CIFAR-10 under the same protocol; see Appendix C.5 for details and results. CIFAR-100 remains our primary experimental regime for the reasons given above, and the CIFAR-10 results are included as an additional data point rather than a replacement.

C.3 Characterization of the Experimental Regime

The interaction of these design choices defines a specific experimental regime. Removal of data augmentation isolates optimization-induced effects by reducing implicit regularization. CIFAR-100 provides sufficient task complexity to prevent trivial interpolation. The selected ResNet architectures span a controlled range of overparameterization. Based on prior findings (Hardt et al., 2016; Keskar et al., 2017; Zhang et al., 2017), we expect this regime to exhibit: high training accuracy across optimization methods; non-trivial generalization gaps that vary by optimizer; measurable differences in sharpness metrics across methods; and preserved optimization dynamics with informative gradient signals throughout training.

The supplementary ResNet-50/CIFAR-10 experiments in Appendix C.5 corroborate this characterization: training again converges without collapsing into a degenerate memorization regime, and the relative ordering of methods by sharpness and accuracy is qualitatively consistent with the CIFAR-100 regime described here.

C.4 Training Configuration

Models are trained using identical batch sizes, learning-rate schedules, and initialization schemes. Training duration: 200 epochs for ResNet-18 and 100 epochs for ResNet-50 (approximately 92% convergence due to computational constraints, but sufficient for comparative analysis). All experiments were conducted using a single NVIDIA T4 GPU on Google Colab. Results are averaged over multiple random seeds to reduce variance and improve robustness.

C.5 Robustness Check: ResNet-50 on CIFAR-10

As a supplementary check on the dataset/architecture pairing discussed above, we additionally trained ResNet-50 directly on CIFAR-10 (100 epochs, no data augmentation, identical protocol to the main CIFAR-100 experiments) for FSFP, Baseline, and SAM ($\rho = 0.05$). FSFP, Baseline, and SAM were each run for 5 seeds. Table 13 reports the individual per-run test accuracy, and Table 14 reports the corresponding aggregate results. Across all three methods, training converged to 100% train accuracy without a collapse into a degenerate memorization regime: test accuracy remained in the 88–91% range for every run, and generalization gaps stayed in a comparable range to the CIFAR-100 setting. We report this as an empirical observation on this particular pairing; it does not change the motivation given above for using CIFAR-100 as our primary experimental regime, and the CIFAR-10 results here are included as an additional data point rather than a replacement for the main experiments.

Table 13: Individual run results: ResNet-50 on CIFAR-10

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std
Baseline	88.45	88.87	89.44	89.57	89.99	89.26 ± 0.61
FSFP	89.55	90.00	89.24	89.06	89.62	89.49 ± 0.36
SAM	90.92	90.06	90.07	90.33	90.41	90.36 ± 0.35

Table 14: ResNet-50 on CIFAR-10: aggregate performance

Method	Runs	Train Acc (%)	Test Acc (%) ↑	Gen Gap (%) ↓	NLL ↓
FSFP	5	100.00	89.49 ± 0.36	10.51 ± 0.36	0.3644 ± 0.0095
Baseline	5	100.00	89.26 ± 0.61	10.74 ± 0.61	0.4008 ± 0.0183
SAM ($\rho = 0.05$)	5	100.00	90.36 ± 0.35	9.64 ± 0.35	0.3248 ± 0.0134

Table 15 reports the per-run absolute sharpness for all three methods on this pairing.

Table 15: Absolute sharpness results per run, ResNet-50 on CIFAR-10 (100 epochs).

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std Dev ↓
FSFP	1.8626	1.7809	2.6722	1.2946	1.7762	1.8773 ± 0.4978
Baseline	3.9386	3.1110	6.6892	3.0435	3.2473	4.0059 ± 1.5417
SAM	2.7616	4.6443	3.7191	3.5043	3.4977	3.6254 ± 0.6753

Table 16 reports the corresponding per-run normalized sharpness values.

Table 16: Normalized sharpness results per run, ResNet-50 on CIFAR-10 (100 epochs).

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Mean ± Std Dev ↓
FSFP	0.0025	0.0027	0.0035	0.0024	0.0034	0.0029 ± 0.0005
Baseline	0.0037	0.0032	0.0057	0.0052	0.0039	0.0044 ± 0.0011
SAM	0.0026	0.0045	0.0032	0.0031	0.0036	0.0034 ± 0.0007

On this pairing, FSFP again attains the lowest absolute sharpness of the three methods, and Baseline remains the sharpest and lowest-accuracy method on average, consistent with the CIFAR-100 results reported in Section 6. One notable departure, however, is that FSFP also attains the lowest *normalized* sharpness on this pairing (0.0029 vs. SAM's 0.0034), the reverse of the CIFAR-100 ordering, where SAM's direct curvature minimization yields the lower normalized value (Table 11). SAM nonetheless retains the highest mean test accuracy on both datasets. We take this as evidence that the absolute/normalized divergence documented in Section 6 is a property of the particular training regime rather than a fixed signature of FSFP's route to flatness, a point we return to in Section 8.

C.6 T-Sweep: Sensitivity of the Selected Trial to the Number of Perturbation Trials

A concentration-of-measure concern applies to the choice of $T = 20$ perturbation trials: in a high-dimensional logit space, flip counts $\Delta^{(t)}$ across $T = 20$ random trials could cluster tightly together purely as a consequence of basic concentration statistics, regardless of whether the underlying model is actually flat or sharp. Under this concern, selecting $t^* = \arg\min_t \Delta^{(t)}$ over only 20 draws might simply select the smallest of 20 near-identical values rather than identifying a geometrically meaningful direction, and increasing T substantially (e.g. to 100 or 1000) might continue to reveal meaningfully smaller flip counts.

To test this directly, we took two already-trained FSFP models, one ResNet-50 trained on CIFAR-10 and one ResNet-50 trained on CIFAR-100, each a single trained model (one seed), both from the 100-epoch runs described above, and, *without any retraining*, ran a post-hoc sweep over $T \in \{20, 50, 100, 200, 500, 1000\}$ on the frozen trained weights. For each T, we recomputed the selected trial's flip count $\Delta(t^*)$ and the resulting $\mathcal{R}_{\text{flat}}$. To keep the comparison across T values from being confounded by independent resampling, the noise for the largest T value (1000) was drawn once, and each smaller T used the corresponding prefix of that same pool of draws (e.g. the $T = 20$ trials are exactly the first 20 draws of the $T = 1000$ pool). Results were averaged over multiple held-out test batches. We report this as a single-model demonstration on each dataset; we have not repeated the sweep across multiple seeds, nor have we run the analogous sweep on an untrained or randomly initialized model as a control.

Table 17: T-sweep results: ResNet-50 on CIFAR-10

	$T = 20$	$T = 50$	$T = 100$	$T = 200$	$T = 500$	$T = 1000$
Mean $\Delta(t^*)$	0.000	0.000	0.000	0.000	0.000	0.000
Mean $\mathcal{R}_{\text{flat}}$	0.003952	0.003952	0.003952	0.003952	0.003952	0.003952

Table 18: T-sweep results: ResNet-50 on CIFAR-100

	$T = 20$	$T = 50$	$T = 100$	$T = 200$	$T = 500$	$T = 1000$
Mean $\Delta(t^*)$	0.700	0.100	0.100	0.000	0.000	0.000
Mean $\mathcal{R}_{\text{flat}}$	0.002480	0.002584	0.002584	0.002516	0.002516	0.002516

CIFAR-10. As Table 17 shows, the selected-trial flip count is already exactly 0 at $T = 20$ and remains exactly 0 through $T = 1000$; the resulting mean $\mathcal{R}_{\text{flat}}$ is identical to four decimal places at every T tested. This is the expected behavior once a model has reached very high train accuracy: prediction margins are large enough that, at the noise scale used, perturbations essentially never flip an argmax, so there is no smaller flip count left for additional trials to find, and the same trial index remains selected as t^* regardless of how many further draws are added.

CIFAR-100. Table 18 shows a different, less clean pattern. The selected-trial flip count is not yet at floor at $T = 20$ (mean 0.700), decreases to 0.100 by $T = 50$–100, and only reaches exactly 0 at $T = 200$, after which it stays at 0 through $T = 1000$. The corresponding $\mathcal{R}_{\text{flat}}$ values are *not* monotonic in T: they rise slightly from 0.002480 ($T = 20$) to 0.002584 ($T = 50$–100) before settling to 0.002516 from $T = 200$ onward. We report this non-monotonicity as observed, rather than smoothing over it: t^* is selected to minimize flip count, not to minimize $\mathcal{R}_{\text{flat}}$ directly, so $\mathcal{R}_{\text{flat}}$ at t^* is a secondary quantity that rides along with the selection and need not decrease monotonically as T grows.

Interpretation. On CIFAR-10, $T = 20$ is already indistinguishable from the achievable floor: every larger T tested returns the identical outcome. On CIFAR-100, $T = 20$ is close to but not exactly at the floor: the flip count is already small (well under 1 flip on average) at $T = 20$ and closes to exactly 0 by $T = 200$, with the associated $\mathcal{R}_{\text{flat}}$ values across the entire tested range spanning only 0.002480–0.002584, a narrow band. In neither case do we observe the pattern this concern would predict if $T = 20$ were substantially undersampled, namely a flip count that keeps falling steadily and substantially as T grows well beyond 20; on both datasets the quantity of interest reaches, or comes very close to, a stable floor within the tested range. This directly addresses the concentration-of-measure concern with a quantitative measurement on these two trained models rather than a theoretical argument alone. It does not, by itself, establish that the selected direction t^* reflects genuine geometric flatness of the loss landscape rather than sampling behavior of the perturbation procedure; a control sweep on an untrained or randomly initialized model, which we have not run, would be needed to isolate that question, and we leave it for future work.

C.7 Flatness–Stability Relationship Under Plain SGD

This section reports a study distinct in purpose from the T-sweep above (Appendix C.6). The T-sweep tests a hyperparameter internal to FSFP: whether $T = 20$ perturbation trials is sufficient for Algorithm 1 to select a stable trial. The study here asks a different, more basic question, more basic than and independent of FSFP's specific mechanism: are flatness and prediction stability related at all under ordinary training dynamics? We designed this as a direct test of that relationship, since it is the premise the rest of the paper builds on, and because reasoning about it in the abstract, as we do earlier via Proposition 6, is not the same as measuring it. Throughout this section we describe results in terms of flatness; the quantity actually computed is Normalized Sharpness (Algorithm 4), and flatness refers to its inverse, so lower Normalized Sharpness corresponds to greater flatness. Table 19 and Figure 1 report the underlying quantity, sharpness, directly, since that is what was measured.

Setup. We trained 30 ResNet-18 models on CIFAR-10 (no augmentation, matching our standard protocol) using plain SGD only, with no FSFP regularization and no SAM. We varied only two ordinary optimization hyperparameters, learning rate $\in \{0.01, 0.03, 0.1, 0.3, 1.0\}$ and batch size $\in \{64, 128, 256\}$, each combination trained with 2 seeds, for a full factorial grid of $5 \times 3 \times 2 = 30$ independent runs (60 epochs each, weight decay 5×10^{-4}, cosine learning-rate schedule). Neither hyperparameter targets stability or flatness directly; both are known from the literature to influence the sharpness of the resulting minimum as a side effect of

optimization dynamics, which is what lets this grid produce a natural range of flatness values without any explicit flatness objective.

Metrics. For each trained model we computed: (i) Normalized Sharpness, exactly as in Algorithm 4 ($\varepsilon = 0.01$, $N = 50$); and (ii) two indicators of prediction stability under the same style of scale-adaptive weight-space perturbation, evaluated on up to 10 held-out test batches: the mean prediction flip rate, and the mean squared logit difference (logit variance) between the clean and perturbed model. We emphasize that these stability indicators are weight-space perturbation statistics, distinct from $\mathcal{R}_{\text{flat}}$, which FSFP computes via logit-space perturbation with trial selection (Algorithm 1). Both instantiate the same underlying idea, namely output sensitivity to perturbation, but they are not the same quantity, and we do not treat a correlation here as a direct validation of $\mathcal{R}_{\text{flat}}$.

Results. Table 20 reports all 30 runs. Across the full grid, sharper models show significantly higher instability by both indicators: Both correlations are positive and statistically significant at $N = 30$: the

Table 19: Sharpness–stability correlation statistics (N=30)

Comparison	Pearson r	Pearson p	Spearman r	Spearman p
Sharpness vs. mean flip rate	0.506	0.0043	0.402	0.0275
Sharpness vs. mean logit variance	0.695	< 0.0001	0.685	< 0.0001

logit-variance relationship is strong, the flip-rate relationship is moderate. Figure 1 shows both relationships directly, with learning rate encoded by color and batch size by marker shape.

Interpretation. This is evidence that flatness and prediction stability co-vary as a general property of trained networks under ordinary SGD dynamics, independent of any algorithm designed around either quantity. It supports the premise underlying our choice to target output-space stability as a proxy for flatness in FSFP's design: the two quantities are empirically related under standard training, not only under the argument given in Proposition 6. FSFP's own connection to flatness is supported separately by the direct Absolute/Normalized Sharpness comparisons against baseline and SAM (Section 6.3, Appendix C.5) and by the T-sweep above (Appendix C.6), which tests a different question specific to Algorithm 1's own hyperparameter; we do not present this study as evidence specific to $\mathcal{R}_{\text{flat}}$ or to FSFP itself.

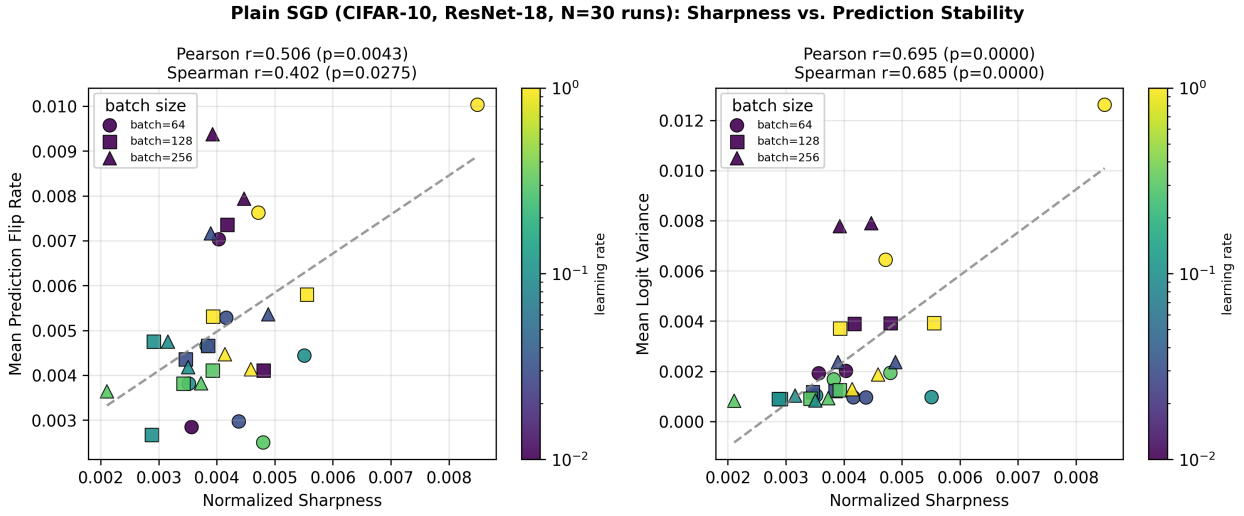

Figure 1: Normalized Sharpness vs. prediction-stability indicators (N=30 plain-SGD runs)

Table 20: Sharpness–stability correlation study: all 30 runs

LR	Batch	Seed	Test Acc (%)	Norm. Sharpness	Flip Rate	Logit Var.
0.01	64	0	86.69	0.004035	0.0070	0.002015
0.01	64	1	87.15	0.003567	0.0028	0.001920
0.01	128	0	86.18	0.004804	0.0041	0.003914
0.01	128	1	84.98	0.004181	0.0074	0.003897
0.01	256	0	84.20	0.003929	0.0094	0.007782
0.01	256	1	83.67	0.004471	0.0079	0.007901
0.03	64	0	88.27	0.004164	0.0053	0.000962
0.03	64	1	88.35	0.004381	0.0030	0.000956
0.03	128	0	87.35	0.003464	0.0044	0.001170
0.03	128	1	87.16	0.003852	0.0047	0.001226
0.03	256	0	86.21	0.004888	0.0054	0.002365
0.03	256	1	85.93	0.003898	0.0072	0.002368
0.10	64	0	88.37	0.003521	0.0038	0.001036
0.10	64	1	89.29	0.005510	0.0044	0.000970
0.10	128	0	87.47	0.002911	0.0048	0.000900
0.10	128	1	87.47	0.002880	0.0027	0.000892
0.10	256	0	87.20	0.003508	0.0042	0.000838
0.10	256	1	87.10	0.003159	0.0048	0.001033
0.30	64	0	88.65	0.004801	0.0025	0.001936
0.30	64	1	89.16	0.003826	0.0047	0.001676
0.30	128	0	88.58	0.003424	0.0038	0.000908
0.30	128	1	88.65	0.003929	0.0041	0.001250
0.30	256	0	86.59	0.003730	0.0038	0.000928
0.30	256	1	86.98	0.002109	0.0036	0.000827
1.00	64	0	80.82	0.004718	0.0076	0.006442
1.00	64	1	78.79	0.008492	0.0100	0.012617
1.00	128	0	84.09	0.005552	0.0058	0.003919
1.00	128	1	85.37	0.003935	0.0053	0.003702
1.00	256	0	87.17	0.004141	0.0045	0.001288
1.00	256	1	87.13	0.004588	0.0041	0.001873