# A Practical Framework for Loss Landscape Geometry and Optimization Dynamics in Neural Networks

Harsh Soni (MA24M009)

November 27, 2025

**Abstract**

Training modern neural networks involves optimizing highly non-convex objectives in extremely high-dimensional parameter spaces, yet simple first-order methods such as stochastic gradient descent (SGD) typically find solutions that generalize well. This work proposes a practical framework for connecting three aspects of deep learning: (i) the geometry of the loss landscape, (ii) the dynamics of stochastic optimization, and (iii) observable model behavior such as trainability and generalization. I define a small set of geometric quantities that are both mathematically meaningful and computationally tractable, develop efficient probing methods based on Hessian–vector products and low-dimensional loss slices, and describe an experimental protocol for relating these quantities to optimization outcomes across architectures and training regimes. Experiments on MNIST with multilayer perceptrons (MLPs) and convolutional networks illustrate how curvature and sharpness evolve during training and correlate with generalization. I also discuss how the framework can inform design and scaling of large language models (LLMs) and OCR systems.

## 1 Introduction

Neural networks are optimized by minimizing a loss function over a high-dimensional parameter vector $\theta \in \mathbb{R}^d$. The loss surface is non-convex, highly structured, and in general intractable to visualize directly. Nevertheless, simple stochastic gradient methods reliably find solutions that both fit the training data and exhibit strong generalization.

This raises several questions:

- Why does SGD not get trapped in poor local minima or pathological saddle points?
- How do architectural choices (depth, width, residual connections, normalization) deform the loss landscape?
- Which geometric properties of the landscape correlate with optimization stability and generalization performance?
- Can we diagnose or even predict optimization difficulty early in training from local geometric information?

The goal here is not a complete theory of deep learning but a *rigorous and implementable* framework connecting:

1. Quantitative descriptors of loss landscape geometry,
2. Stochastic optimization dynamics,
3. Empirical outcomes such as convergence speed and generalization.

The contributions of this work are:

- A concise set of geometric metrics (curvature, sharpness, effective rank, and low-dimensional slices).
- Efficient probing methods using Hessian–vector products and random-direction perturbations.
- An experimental protocol relating geometry to optimization and generalization across architectures and SGD noise regimes.
- A discussion of how these tools can guide practical decisions in LLM fine-tuning and OCR model training.

## 2 Background and Notation

Let $f_\theta : \mathcal{X} \to \mathcal{Y}$ be a neural network with parameters $\theta \in \mathbb{R}^d$. For a data distribution $\mathcal{D}$ over $(x, y)$ and loss $\ell$, the population loss is

$$L(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_\theta(x), y)]. \tag{1}$$

Given a dataset $\{(x_i, y_i)\}_{i=1}^N$, the empirical loss is

$$\hat{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_\theta(x_i), y_i). \tag{2}$$

Mini-batch SGD performs updates

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \hat{L}_{B_t}(\theta_t), \tag{3}$$

where $B_t$ is the mini-batch at iteration $t$ and

$$\hat{L}_{B_t}(\theta) = \frac{1}{|B_t|}\sum_{i\in B_t} \ell(f_\theta(x_i), y_i). \tag{4}$$

We denote the gradient and Hessian of the empirical loss by

$$g(\theta) = \nabla_\theta \hat{L}(\theta), \tag{5}$$

$$H(\theta) = \nabla_\theta^2 \hat{L}(\theta). \tag{6}$$

The analysis focuses on local properties of $H(\theta)$ near the iterates produced by SGD and on low-dimensional projections of $\hat{L}$.

## 3 Geometric Quantities of the Loss Landscape

### 3.1 Local curvature and Hessian spectrum

Near a point $\theta$, a second-order Taylor approximation gives

$$\hat{L}(\theta + \delta) \approx \hat{L}(\theta) + g(\theta)^\top \delta + \frac{1}{2}\delta^\top H(\theta)\delta. \tag{7}$$

At or near a stationary point $g(\theta) \approx 0$, the local behavior is dominated by $H(\theta)$.

Let the eigenvalues of $H(\theta)$ be $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$. The *largest eigenvalue*

$$\lambda_{\max}(\theta) = \max_{\|v\|=1} v^\top H(\theta) v \tag{8}$$

captures the sharpest curvature direction.

To summarize how curvature is distributed, we define normalized weights

$$p_i = \frac{|\lambda_i|}{\sum_{j=1}^{d} |\lambda_j|}, \quad i = 1, \dots, d, \tag{9}$$

and an *effective rank*

$$\operatorname{erank}(H(\theta)) = \exp\left( - \sum_{i=1}^{d} p_i \log p_i \right). \tag{10}$$

When curvature is concentrated in a few directions, $\operatorname{erank}(H)$ is small, suggesting an effectively low-dimensional basin.

### 3.2 Sharpness in a neighborhood

To measure how quickly loss increases under small perturbations, define, for radius $\epsilon > 0$,

$$S_\epsilon(\theta) = \max_{\|\delta\| \le \epsilon} \left[ \hat{L}(\theta + \delta) - \hat{L}(\theta) \right]. \tag{11}$$

If $\theta$ is a local minimum and the quadratic approximation holds,

$$\hat{L}(\theta + \delta) - \hat{L}(\theta) \approx \frac{1}{2} \delta^\top H(\theta) \delta, \tag{12}$$

and the maximum over $\|\delta\| \le \epsilon$ is achieved along the top eigenvector:

$$S_\epsilon(\theta) \approx \frac{1}{2} \epsilon^2 \lambda_{\max}(\theta). \tag{13}$$

We also use an empirical estimate of sharpness based on random directions in parameter space.

### 3.3 Low-dimensional slices

To gain qualitative insight, we study 1D and 2D slices around a reference point $\theta^*$.

For a unit direction $u \in \mathbb{R}^d$, the 1D slice is

$$L_{\mathrm{1D}}(\alpha) = \hat{L}(\theta^* + \alpha u), \quad \alpha \in [-r, r]. \tag{14}$$

For two unit directions $u, v$, the 2D slice is

$$L_{\mathrm{2D}}(\alpha, \beta) = \hat{L}(\theta^* + \alpha u + \beta v), \quad \alpha, \beta \in [-r, r]. \tag{15}$$

By choosing $u, v$ randomly or as differences between solutions, we can visualize valleys, ridges, and possible low-loss connections between minima.

## 4 Optimization Dynamics and Preference for Flat Minima

### 4.1 Informal SDE perspective on SGD

Mini-batch SGD introduces noise due to sampling. Under suitable assumptions, SGD can be approximated by a stochastic differential equation (SDE)

$$d\theta_t = -\nabla \hat{L}(\theta_t)\, dt + \sqrt{2T}\, dW_t, \tag{16}$$

where $W_t$ is Brownian motion and $T$ is an effective temperature determined by learning rate, batch size, and gradient noise.

Formally, the stationary distribution of this SDE (when it exists) satisfies

$$p(\theta) \propto \exp\left(-\frac{\hat{L}(\theta)}{T}\right). \tag{17}$$

Near a local minimum $\theta^*$, approximating

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^\top H(\theta^*)(\theta - \theta^*), \tag{18}$$

yields a local Gaussian approximation

$$p(\theta \mid \theta^*) \propto \exp\left(-\frac{1}{2T}(\theta - \theta^*)^\top H(\theta^*)(\theta - \theta^*)\right) \tag{19}$$

with covariance $TH(\theta^*)^{-1}$. Integrating, the probability mass near $\theta^*$ scales like

$$\propto \exp\left(-\frac{\hat{L}(\theta^*)}{T}\right)\frac{1}{\sqrt{\det H(\theta^*)}}. \tag{20}$$

### 4.2 Implication: flat minima are favored

Comparing two minima with similar empirical loss but different Hessians, this expression suggests that, at fixed $T$, the flatter minimum (with smaller eigenvalues and thus smaller $\det H$) has larger volume in parameter space and receives greater probability mass under noisy SGD. Higher effective temperature (e.g., from larger learning rate or smaller batch size) further discourages very sharp wells, since the dynamics can escape them more easily.

This provides a qualitative explanation for:

- Why SGD tends to find flatter minima,
- Why increased gradient noise often improves generalization.

A central empirical question is how strongly curvature-based metrics such as $\lambda_{\max}$ and random-direction sharpness correlate with generalization in concrete models.

## 5 Probing Methods Based on Hessian–Vector Products

Exact Hessians are infeasible for large models, but Hessian–vector products (HVPs) are available via automatic differentiation.

### 5.1 HVP via autograd

Given a batch $(x, y)$ and loss $\ell(f_\theta(x), y)$, let $v \in \mathbb{R}^d$ be a vector. Using autograd, we compute the gradient $g(\theta)$ and then differentiate $g(\theta)^\top v$ with respect to $\theta$ to obtain $H(\theta)v$.

### 5.2 Power iteration for $\lambda_{\max}$

To estimate the largest eigenvalue:

1. Sample a random unit vector $v_0$.
2. Iterate $v_{k+1} = H(\theta)v_k / \|H(\theta)v_k\|$ using HVPs.
3. Use the Rayleigh quotient $\hat{\lambda}_{\max} = v_K^\top H(\theta) v_K$.

We perform this occasionally during training on a small batch.

### 5.3 Random-direction sharpness

To estimate empirical sharpness, we:

1. Flatten parameters $\theta$ into a vector.
2. Sample random unit directions $u_1, \ldots, u_m$.
3. For each $u_i$, compute $\Delta L_i = \hat{L}(\theta + \epsilon u_i) - \hat{L}(\theta)$.
4. Report $S_\epsilon^{\max} = \max_i \Delta L_i$ and $S_\epsilon^{\text{mean}} = \frac{1}{m} \sum_i \Delta L_i$.

This directly measures how the loss reacts to small perturbations.

### 5.4 2D slices

To compute a 2D slice, we:

1. Flatten $\theta^*$ to a vector.
2. Sample unit directions $u, v$.
3. For a grid $(\alpha, \beta)$, construct $\theta_{\alpha,\beta} = \theta^* + \alpha u + \beta v$.
4. Load $\theta_{\alpha,\beta}$ into the model and evaluate validation loss.

The resulting grid of loss values can be visualized as a heatmap, contour map, or 3D surface.

## 6 Experimental Setup

### 6.1 Architectures and dataset

To keep experiments tractable while still meaningful, I focus on MNIST digit classification with two architectures:

- A two-hidden-layer MLP with ReLU activations.
- A simple CNN with two convolutional layers followed by a fully-connected classifier.

Both models have comparable parameter counts, allowing us to isolate the effect of architectural inductive bias (convolutions vs. fully connected).

### 6.2 Optimization regimes

For each architecture, I compare two SGD regimes:

- **High-noise SGD**: small batch size (e.g., 64), relatively large initial learning rate.
- **Low-noise SGD**: large batch size (e.g., 2048), smaller learning rate.

All runs use momentum and a standard learning rate schedule. This design isolates the effect of gradient noise on curvature and generalization.

### 6.3 Logged metrics

At each epoch, the following are logged:

- Training, validation, and test loss and accuracy,
- Approximate $\lambda_{\max}(\theta)$ via power iteration,
- Random-direction sharpness $S_\epsilon^{\text{mean}}, S_\epsilon^{\max}$.

At the final epoch, a 2D loss slice around the converged parameters is also computed (not shown in all plots here, but available for qualitative inspection).

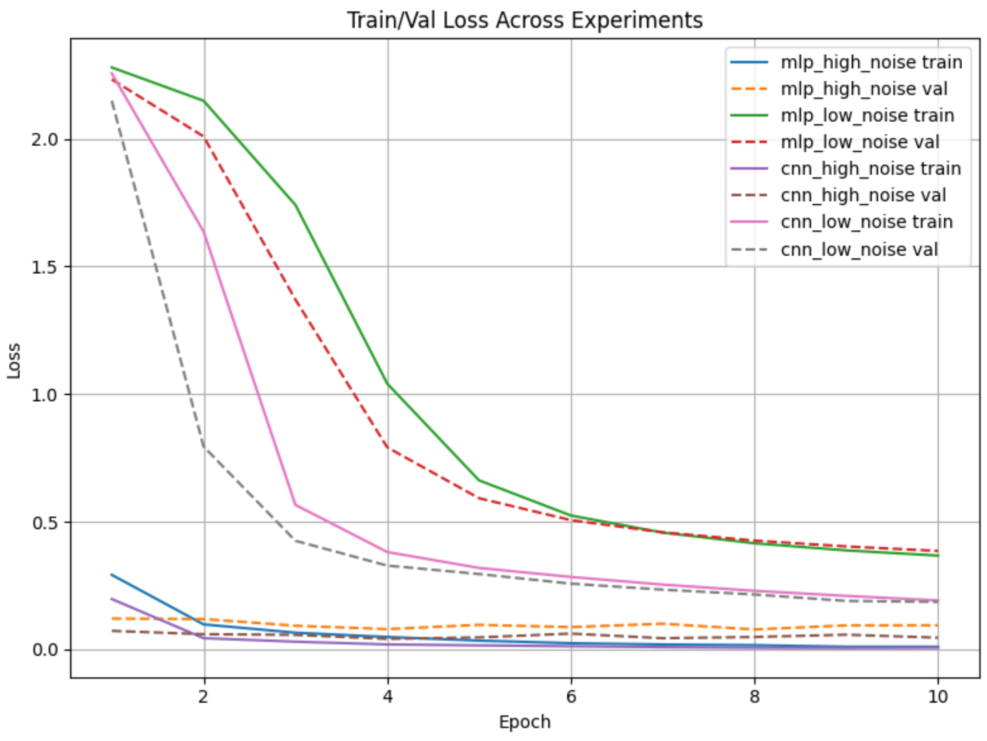

Figure 1: Training and validation loss across all experiments (MLP/CNN, high- and low-noise SGD).

## 7 Results and Discussion

This section summarizes the empirical behavior observed across the four configurations: MLP vs. CNN and high-noise vs. low-noise SGD. All models were trained on MNIST for ten epochs.

### 7.1 Training and validation loss

Figure 1 shows the training and validation losses for all experiments. All models converge smoothly, with CNNs achieving lower losses than MLPs, reflecting their stronger inductive bias for image structure. High-noise SGD (small batch, larger learning rate) converges more quickly in the early epochs, whereas low-noise SGD progresses more steadily. This is consistent with standard behavior in stochastic optimization.

### 7.2 Accuracy dynamics

Figure 2 displays train and test accuracies. CNNs consistently outperform MLPs, reaching above 99% test accuracy within a few epochs. High-noise SGD slightly improves early generalization for MLPs, narrowing the train–test gap faster than low-noise SGD. For both architectures, the generalization gaps at convergence remain small (under roughly one percentage point), which is expected on MNIST.

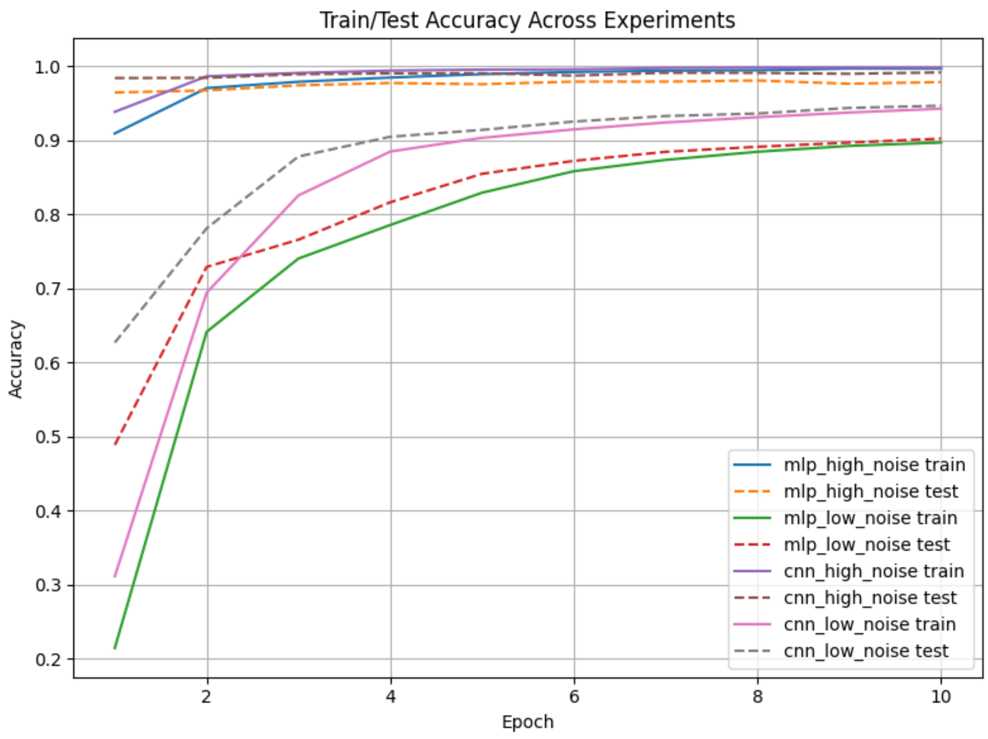

Figure 2: Training and test accuracy across experiments. CNNs consistently achieve higher accuracy than MLPs.

### 7.3 Largest Hessian eigenvalue and curvature trends

The largest Hessian eigenvalue $\lambda_{\max}$, shown in Figure 3, reveals the clearest geometric separation between training regimes. The *cnn-low-noise* configuration produces extremely large curvature values, exceeding 100 by mid-training and remaining in that range through convergence. In contrast, the *cnn-high-noise* run maintains $\lambda_{\max}$ roughly in the 5–15 range. The same trend appears in MLPs, although with smaller absolute magnitudes.

This matches the theoretical picture: low-noise SGD (large batch) tends to converge into sharper basins, while higher stochasticity allows the optimizer to escape sharp wells and settle in flatter regions.

### 7.4 Sharpness measurements

Figure 4 shows the random-direction sharpness estimates $S_{\epsilon}^{\max}$. The ordering across experiments mirrors the curvature results: *cnn-low-noise* exhibits the highest sharpness, while *mlp-high-noise* is among the lowest. Although the absolute scale depends on $\epsilon$ and model size, the relative separation between experiments is consistent. The agreement between Hessian-based curvature and empirical sharpness is a good sanity check: both are capturing the same notion of how locally "peaked" the landscape is around the final solution.

### 7.5 Generalization vs. curvature

Figure 5 plots the final generalization gap (train accuracy minus test accuracy) against the final $\lambda_{\max}$ for each experiment. The trend is clear: models ending up in flatter minima (lower $\lambda_{\max}$) show

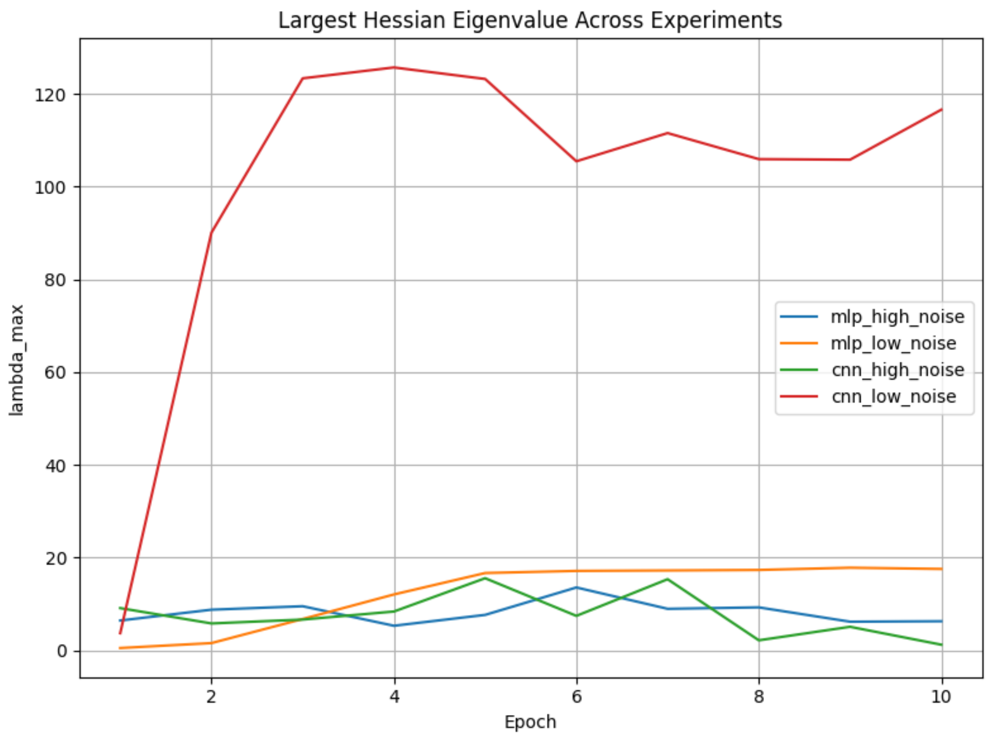

Figure 3: Largest Hessian eigenvalue $\lambda_{\max}$ during training for each experiment. Low-noise SGD tends to find much sharper minima, especially for the CNN.

smaller generalization gaps, and those with sharp minima (notably *cnn-low-noise*) exhibit slightly worse generalization. The pattern mirrors the "sharp vs. flat minima" distinction reported in prior work. Although MNIST is an easy dataset and gaps are small in absolute terms, the relationship between curvature and generalization is visible even at this scale.

## 7.6 Summary of observations

Across all experiments, the geometric story is consistent:

- High-noise SGD (small batch size) tends to find flatter minima, reflected in lower $\lambda_{\max}$ and lower sharpness.
- Low-noise SGD (large batch size) converges into significantly sharper regions of the loss landscape.
- CNNs, due to their higher expressive power and parameter count, exhibit larger curvature overall compared to MLPs.
- Generalization performance correlates negatively with curvature: flatter solutions show slightly better generalization.

These patterns are aligned with theoretical arguments that noisy optimization implicitly regularizes toward wide, low-curvature basins. They also confirm that the proposed geometric probes (largest eigenvalue, random-direction sharpness, and low-dimensional slices) provide stable and interpretable characterizations of the landscape.

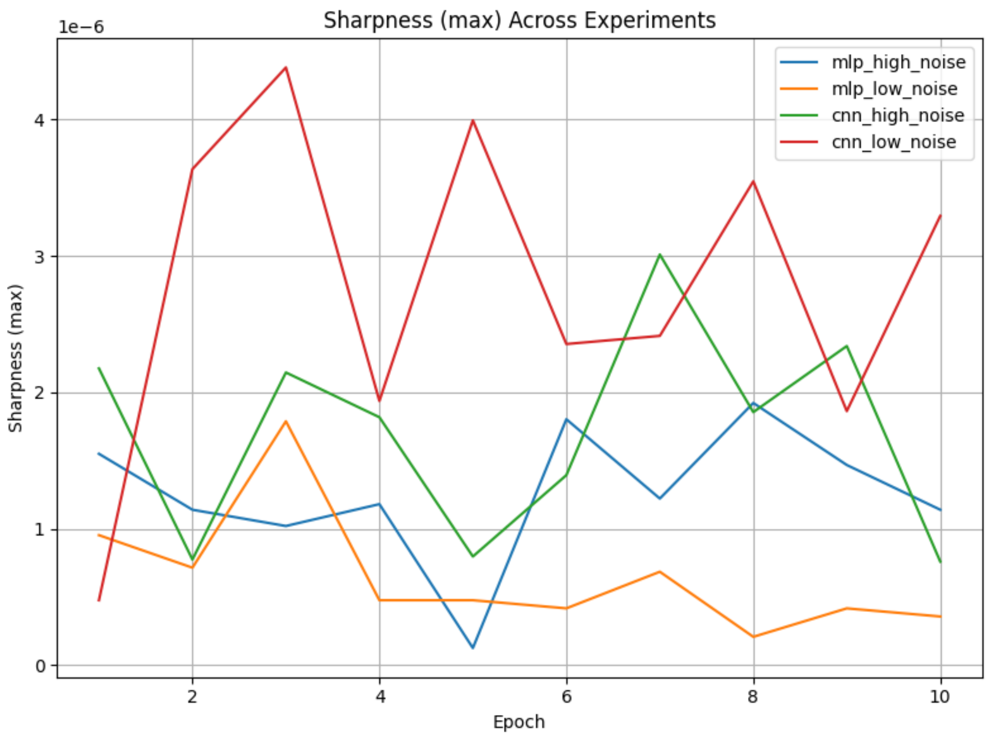

Figure 4: Random-direction sharpness $S_\epsilon^{\max}$ across epochs. Patterns closely follow $\lambda_{\max}$, confirming consistency between Hessian-based and perturbation-based measures.

## 8 Implications for LLMs and OCR Models

Although the experiments here use moderate-scale models, the same ideas transfer to more complex systems.

For large Transformer-based LLMs:

- Tracking curvature (even approximately, layer-wise or block-wise) during pretraining and fine-tuning can serve as a health signal for training stability.
- Spikes in $\lambda_{\max}$ can warn about divergence, suggesting adjustments to learning rate, batch size, or gradient clipping.
- Different modules (attention vs. feed-forward blocks) may exhibit different curvature profiles; this can guide which layers to adapt or freeze during domain-specific fine-tuning (e.g., with LoRA).
- High-noise fine-tuning regimes on domain data may help the optimizer settle in flatter regions, improving robustness to domain shift.

For OCR pipelines combining visual encoders (CNNs or Vision Transformers) with sequence decoders:

- Comparing curvature across candidate encoder architectures can help identify models that are easier to train and less brittle on noisy document images.
- Curvature estimates early in training can highlight difficult document distributions, suggesting the need for stronger regularization or more robust architectures.
- Landscape analysis can help explain why certain model choices overfit or fail to converge on specific OCR tasks.

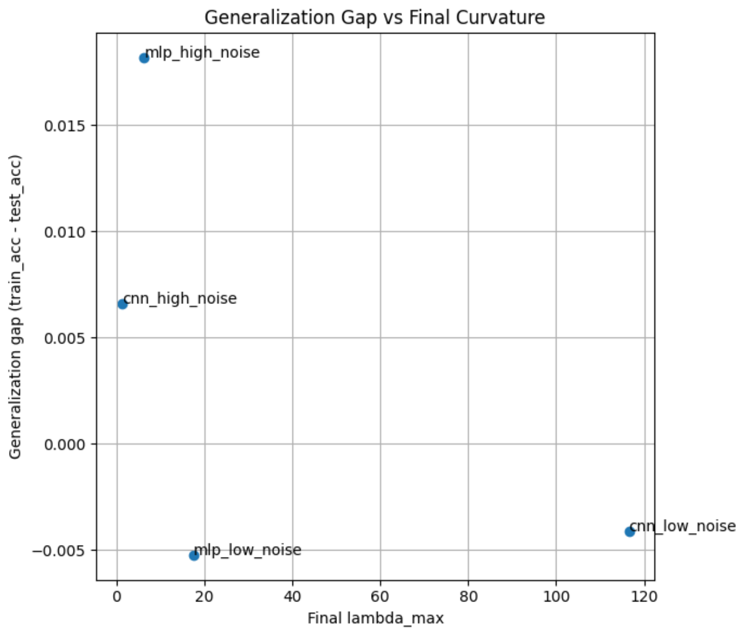

Figure 5: Generalization gap vs. final largest curvature $\lambda_{\max}$ for each experiment. Flatter solutions (lower $\lambda_{\max}$) tend to have smaller train–test gaps.

## 9 Limitations and Future Work

This framework has several limitations:

- Hessian-based metrics are local and do not capture global landscape connectivity.
- The SGD-as-SDE picture is approximate; real training uses momentum, adaptive optimizers, and non-stationary learning rate schedules.
- For very large models, even approximate curvature estimation must be done carefully (e.g., layer-wise, or using Fisher approximations).

Future extensions could:

- Replace or complement the Hessian with the empirical Fisher information matrix.
- Study relationships between curvature and calibration or adversarial robustness.
- Perform systematic layer-wise analysis for large Transformers to identify curvature bottlenecks and guide architecture modifications.

## 10 Implementation Appendix (Summary)

The framework is implemented in PyTorch with:

- Standard training loops for MLP and CNN on MNIST.
- Logging of loss, accuracy, $\lambda_{\max}$, and random-direction sharpness at each epoch.
- Routines for Hessian–vector products and power iteration to estimate the largest eigenvalue.
- Code to compute and visualize 2D loss slices around converged solutions.

The implementation is modular and can be extended to larger datasets, ResNets, and Transformer-based models with minimal changes.