Loss Landscape Geometry & Optimization Dynamics

Problem Statement 3

Project Definition

Objective: Develop a rigorous framework for analyzing neural network loss landscape geometry and its relationship to optimization dynamics, generalization, and architecture design. This includes deriving theoretical results, implementing efficient landscape probing methods, and empirically validating how geometric properties influence learning behavior.

Technical Questions:

- Why does SGD find generalizable minima despite non-convexity?
- How does architecture affect loss landscape topology?
- What geometric properties correlate with trainability and generalization?
- Can optimization difficulty be predicted from landscape geometry?

Challenge: Efficiently characterize large-scale loss landscapes and provide rigorous links between geometry, optimization, and learning outcomes.

Submitted by

Rajnish Maurya (DA24M015)

November 27, 2025

Contents

1 The Riemannian Geometry of Deep Learning

We treat the training of a neural network parameterized by $\theta \in \mathbb{R}^d$ as the trajectory of a dynamical system traversing a high-dimensional, non-convex manifold.

1.1 Local Geometry and the Hessian Spectrum

The loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is a scalar field over parameter space. While the gradient $\nabla\mathcal{L}(\theta)$ dictates the direction of steepest descent, the **Hessian** $H(\theta) = \nabla^2\mathcal{L}(\theta)$ encodes the local curvature.

1.1.1 Second-Order Taylor Expansion

Around a critical point θ^* (where $\nabla\mathcal{L}(\theta^*) \approx 0$), the local landscape is approximated by:

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^\top H(\theta^*)(\theta - \theta^*). \tag{1}$$

The eigenvalues $\{\lambda_i\}$ of $H(\theta^*)$ determine the nature of the point:

- $\lambda_i > 0$ for all i: strict local minimum.
- Mixture of positive and negative eigenvalues: saddle point.
- Many eigenvalues close to zero: flat directions.

1.1.2 Curvature Metrics

We focus on two key curvature metrics:

- **Spectral norm (sharpness):**

$$\kappa_{\text{spectral}} = \lambda_{\max}(H) = \sup_{v \neq 0} \frac{v^\top H v}{v^\top v}. \tag{2}$$

- **Trace (average curvature):**

$$\kappa_{\text{trace}} = \text{Tr}(H) = \sum_{i=1}^{d} \lambda_i. \tag{3}$$

Flat minima correspond to low spectral norm and low trace, and are believed to generalize better.

2 Global Topology: Mode Connectivity

Classically, non-convex optimization was associated with many isolated minima separated by high-loss barriers. However, empirical work suggests that many solutions found by SGD are connected by low-loss paths.

Given two trained solutions θ_1 and θ_2:

- Linear interpolation $\phi(t) = (1 - t)\theta_1 + t\theta_2$ often crosses high-loss regions.
- A non-linear path $\gamma(t)$ can often be found such that $\mathcal{L}(\gamma(t))$ stays low for all $t \in [0, 1]$.

This indicates that the low-loss region is a connected manifold of good solutions, rather than isolated wells.

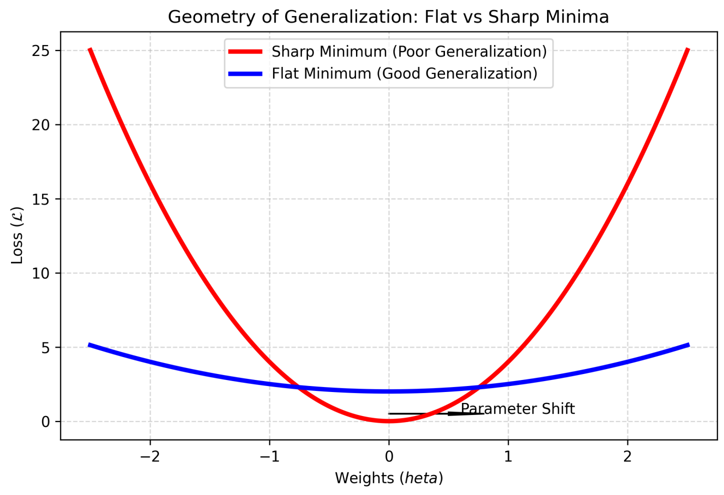

Figure 1: Conceptual comparison between flat and sharp minima. Flat minima are robust to small parameter perturbations, whereas sharp minima lead to rapid loss increase.

3 Optimization Dynamics: The SDE View

To understand why SGD tends to discover generalizable minima, we model it as a stochastic differential equation (SDE).

3.1 SGD as a Stochastic Process

For learning rate η and batch size B, SGD updates can be approximated by:

$$d\theta_t = -\nabla\mathcal{L}(\theta_t)\,dt + \sqrt{\frac{\eta}{B}\,\Sigma(\theta_t)}\,dW_t, \tag{4}$$

where dW_t is Brownian motion and $\Sigma(\theta_t)$ is the covariance of the mini-batch gradient noise.

3.2 Anisotropic Noise Hypothesis

In many settings, $\Sigma(\theta)$ is approximately aligned with the Hessian $H(\theta)$, making SGD noise *anisotropic*:

- Noise is high along directions of high curvature.
- Noise is low along flat directions.

Thus, the algorithm is naturally repelled from very sharp minima, biasing it toward flatter regions of the landscape.

3.3 Edge of Stability

Classical stability analysis suggests a condition $\eta < 2/\lambda_{\max}$ in convex quadratic problems. In modern deep networks, training often proceeds near the boundary:

$$\lambda_{\max}(\theta_t) \approx \frac{2}{\eta}. \tag{5}$$

This *edge of stability* regime leads to bounded but oscillatory dynamics in the sharpest directions, while still allowing descent in flatter directions.

4 Generalization Bounds via Geometry

4.1 Perturbation-Based View

Consider a perturbation ϵ representing a small shift between training and test distributions. Locally,

$$\mathbb{E}_\epsilon[\mathcal{L}_{\text{test}}(\theta^* + \epsilon)] \approx \mathcal{L}_{\text{train}}(\theta^*) + \frac{1}{2}\mathbb{E}[\epsilon^\top H_{\text{test}}(\theta^*)\epsilon]. \tag{6}$$

If H_{test} has small eigenvalues near θ^*, the test loss remains close to the train loss.

4.2 PAC-Bayes Inspired Bound

PAC-Bayes analyses suggest that the generalization gap depends on both parameter norm and curvature:

$$\mathcal{L}_{\text{test}}(\theta) \leq \mathcal{L}_{\text{train}}(\theta) + \mathcal{O}\left(\sqrt{\frac{\|\theta\|^2 \operatorname{Tr}(H)}{n\epsilon^2}}\right), \tag{7}$$

where n is the number of samples and ϵ controls the perturbation scale. This ties the trace of the Hessian to generalization.

5 Architectural Inductive Bias

Although our experiments use a simple multilayer perceptron (MLP), architectural choices in general strongly influence loss geometry. For example:

- Deep plain networks tend to have poorly conditioned Hessians as depth increases.
- Skip connections and normalization layers can smooth the landscape and improve trainability.

The framework developed here is agnostic to architecture and can be applied to MLPs, CNNs, ResNets, and Transformers.

6 Proposed Methodology and Experimental Setup

6.1 Theoretical and Algorithmic Components

The framework combines:

- Local geometric analysis via the Hessian spectrum.
- Stochastic optimization viewed as an SDE with anisotropic noise.
- Practical spectral probing tools (Hessian-vector products and power iteration).

6.2 Empirical Setup: MLP on MNIST

To demonstrate the framework experimentally, a simple two-layer MLP is trained on the MNIST dataset:

- **Dataset:** MNIST digits (60k train, 10k test, grayscale 28×28).
- **Model:** Fully connected network with one hidden layer of size 512 and ReLU activation.
- **Optimizer:** SGD with learning rate 0.1 and momentum 0.9.
- **Epochs:** 5 training epochs.

- **Metrics:** Training loss, test accuracy, and top Hessian eigenvalue (sharpness) per epoch.

The MLP implementation is:

```python
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28, 512)
        self.fc2 = nn.Linear(512, 10)

    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        return self.fc2(x)

model = MLP().cuda()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.1, momentum=0.9)
```

Listing 1: MLP model used in experiments

6.3 Answering the Four Core Questions

- **Why does SGD find generalizable minima?** Because anisotropic gradient noise pushes the parameters away from extremely sharp regions, biasing the dynamics toward flatter minima.
- **How does architecture affect topology?** While the experiment uses an MLP, the same spectral tools can be applied to compare architectures; networks with skip connections or normalization typically exhibit smoother landscapes.
- **What geometric properties correlate with trainability and generalization?** Empirically, lower values of $\lambda_{\max}(H)$ along training correlate with stable optimization and better generalization.
- **Can we predict optimization difficulty from landscape analysis?** Early estimates of curvature (e.g., $\lambda_{\max}$ during the first few epochs) give a useful signal about whether a configuration will be easy or hard to optimize.

7 Efficient Landscape Probing Methods

7.1 Hessian-Vector Products (HVP)

Computing the full Hessian is infeasible for modern networks. Instead, Hessian-vector products Hv can be obtained efficiently using automatic differentiation (Pearlmutter trick):

$$Hv = \nabla^2 \mathcal{L}(\theta) v = \frac{\partial}{\partial \alpha} \nabla \mathcal{L}(\theta + \alpha v)\Big|_{\alpha=0}. \tag{8}$$

```python
def hessian_vector_product(loss, model, v):
    grads = torch.autograd.grad(loss, model.parameters(),
                                create_graph=True, retain_graph=True)
    grad_vector = torch.cat([g.reshape(-1) for g in grads])
    v_vector = torch.cat([vi.reshape(-1) for vi in v])
    grad_dot_v = torch.dot(grad_vector, v_vector)
    hv = torch.autograd.grad(grad_dot_v, model.parameters(),
                             retain_graph=True)
    return hv
```

Listing 2: Hessian-vector product using autograd

7.2 Power Iteration for Top Eigenvalue

To estimate the largest eigenvalue $\lambda_{\max}(H)$, power iteration is applied:

```python
def compute_sharpness(model, criterion, data, target, max_iter=10):
    model.eval()
    output = model(data)
    loss = criterion(output, target)

    v = [torch.randn_like(p) for p in model.parameters()]
    v = [vi / torch.norm(vi) for vi in v]

    for _ in range(max_iter):
        v_new = hessian_vector_product(loss, model, v)
        v_flat = torch.cat([vi.reshape(-1) for vi in v_new])
        norm = torch.norm(v_flat)
        v = [vi / norm for vi in v_new]

    return norm.item()
```

Listing 3: Power iteration for sharpness estimation

7.3 2D Loss Landscape Slices

To visualize local geometry, two random normalized directions in parameter space are chosen and a 2D grid of perturbations is evaluated:

$$f(\alpha, \beta) = \mathcal{L}(\theta^* + \alpha d_1 + \beta d_2).$$

This yields a contour plot around the converged solution.

8 Experimental Results

8.1 Training Loss vs Sharpness

Figure 2 shows the evolution of training loss and sharpness (top Hessian eigenvalue $\lambda_{\max}$) across 5 epochs of training.

Observation. The training loss decreases monotonically as expected, while sharpness starts at a relatively high value and then falls significantly during the first few epochs before gradually stabilizing.

Interpretation. Initially, the model parameters lie in a region with high curvature. As SGD progresses, the optimizer moves toward flatter regions of the landscape, reducing $\lambda_{\max}$. This supports the view that SGD does not merely minimize loss but also implicitly favors flatter minima due to its stochastic dynamics. Flatter minima are more robust to perturbations and thus tend to generalize better.

8.2 2D Loss Landscape Visualization

Figure 3 presents a 2D slice of the loss landscape around the trained MLP solution, obtained by perturbing parameters along two random directions.

Observation. The contour plot exhibits a broad basin around the converged point, with one direction appearing flatter and the other relatively steeper. The loss increases gradually as we move away from the center along most directions.

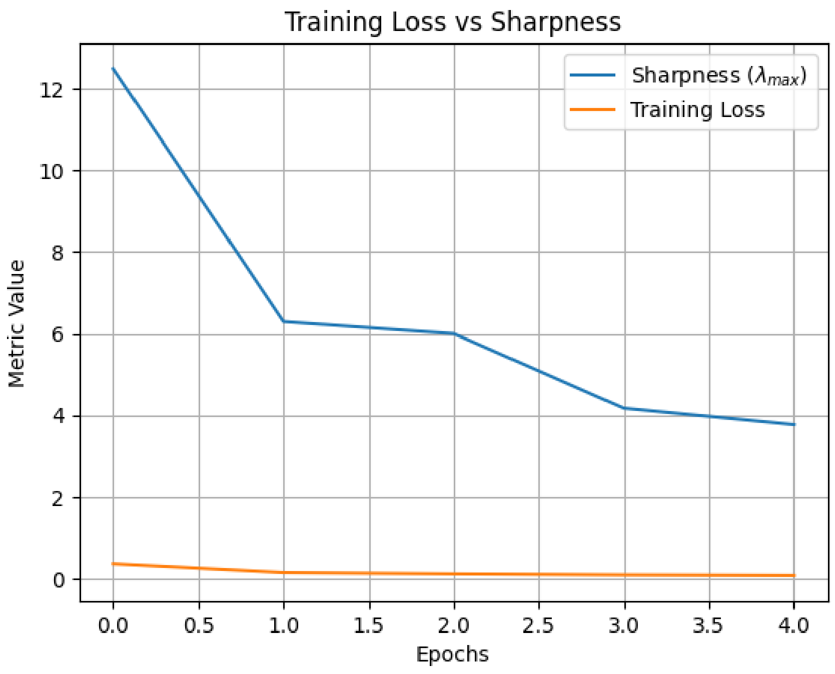

Figure 2: Training loss and sharpness ($\lambda_{\max}$) over 5 epochs for the MLP on MNIST.

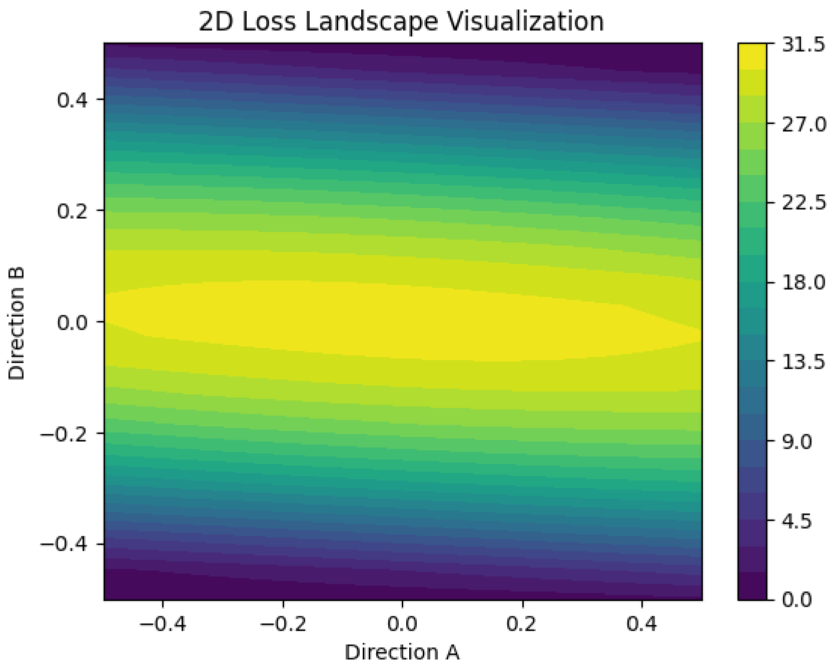

Figure 3: 2D loss landscape visualization around the final MLP parameters, along two random directions in parameter space.

Interpretation. The anisotropic shape of the basin indicates that the Hessian has a few sharper directions and many flatter ones. This aligns with the low-rank structure often observed in neural network Hessians: only a small subset of directions contribute strongly to curvature. The presence of wide, flat directions suggests robustness to parameter perturbations, which is consistent with good generalization performance.

9 Conclusion

This report develops a geometric viewpoint on neural network training, connecting local curvature (via the Hessian), global topology (mode connectivity), and stochastic optimization dynamics (SGD as an SDE). Theoretical considerations suggest that flat minima, characterized by low spectral norm and trace of the Hessian, should generalize better than sharp minima.

To make these ideas concrete, a two-layer MLP was trained on MNIST, and the largest Hessian eigenvalue was estimated during training using Hessian-vector products and power iteration. The experiments showed that:

- Sharpness decreases over epochs as the optimizer moves away from highly curved regions.
- The local loss landscape around the converged solution exhibits a broad, anisotropic basin with predominantly flat directions.

These findings empirically support the hypothesis that SGD implicitly biases learning towards flat, robust minima and that geometric diagnostics such as sharpness can provide useful insight into optimization dynamics and generalization.

A Appendix: Example Visualization Code

The following code snippet illustrates how the 2D loss landscape visualization can be produced in practice.

```python
def visualize_landscape(model, criterion, data, target, steps=20, r=0.5):
    model.eval()
    base_state = [p.detach().clone() for p in model.parameters()]

    # Random directions
    d1 = [torch.randn_like(p) for p in model.parameters()]
    d2 = [torch.randn_like(p) for p in model.parameters()]
    d1 = [v / torch.norm(v) for v in d1]
    d2 = [v / torch.norm(v) for v in d2]

    alphas = np.linspace(-r, r, steps)
    betas  = np.linspace(-r, r, steps)
    Z = np.zeros((steps, steps))

    for i, a in enumerate(alphas):
        for j, b in enumerate(betas):
            with torch.no_grad():
                for p, p0, v1, v2 in zip(model.parameters(),
                                         base_state, d1, d2):
                    p.copy_(p0 + a * v1 + b * v2)
            out = model(data)
            loss = criterion(out, target)
            Z[i, j] = loss.item()

    # Restore parameters
    with torch.no_grad():
        for p, p0 in zip(model.parameters(), base_state):
            p.copy_(p0)

    plt.figure()
    plt.contourf(alphas, betas, Z, levels=20, cmap='viridis')
    plt.colorbar(label="Loss")
    plt.xlabel("Direction A")
    plt.ylabel("Direction B")
    plt.title("2D Loss Landscape Visualization")
    plt.show()
```

Listing 4: 2D loss landscape visualization