Visualizing the Loss Landscape of Neural Nets

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein

Code: github.com/tomgoldstein/loss-landscape

Summary

- We reveal faults in a number of methods for loss landscape visualization, and show that existing strategies fail to accurately capture the local geometry.
- We present a visualization method based on *filter normalization*, which provides accurate visual interpretations for the trainability and generalization of neural nets.
- Network architecture choices have visualizable effects on loss functions that could help explain trainability.

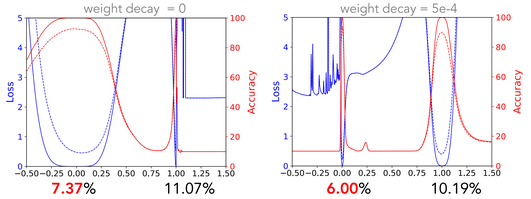
The Sharp vs. Flat Dilemma

It is widely believed that small-batch SGD produces "flat" minimizers that generalizes better, while large-batch sizes produce "sharp" minima with poor generalization [1].

1D linear interpolation

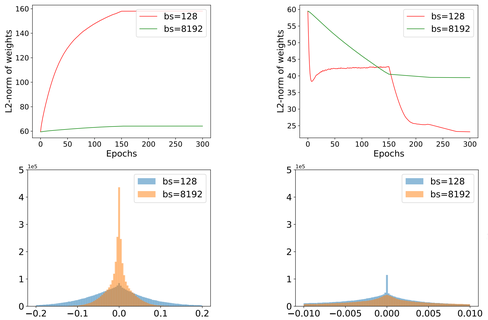
We train a VGG-9 net on CIFAR-10 for a fixed number of epochs using two batch-sizes: 128 and 8192. Let θ^s and θ^l indicate weights of the solutions obtained by small-batch and large batch, respectively. We plot the loss values along the direction $\theta^l - \theta^s$ as in [1, 2], i.e.,

$$f(\alpha) = L(\theta^s + \alpha(\theta^l - \theta^s))$$



Adding weight decay makes contradictive observations

- A smaller batch size results in more weight updates per epoch, causing weights to shrink.
- When weights are small: a small perturbation to the weights has a dramatic effect, making the minimizer sharp.



Filter Normalized Sharpness Comparison

Create filter normalized random direction(s)

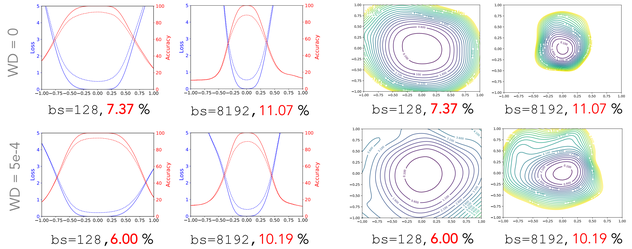
Each filter of the neural network might live on a different scale. To remove this scaling effect, we plot loss functions using filter-wise normalized directions.

- Create a random Gaussian direction vector d with the same dimension as θ.
- Normalize each filter $d_{i,j}$ to have the same norm as corresponding filter in $\theta_{i,j}$, i.e.,

$$d_{i,j} \leftarrow \frac{d_{i,j}}{\|d_{i,j}\|}\|\theta_{i,j}\|$$

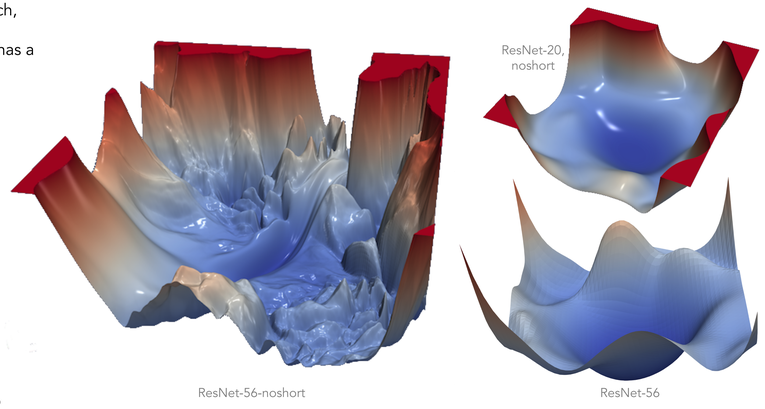
We plot the 1D and 2D surface of minima obtained by using different optimizer, batch size and weight decay.

$f(\alpha) = L(\theta + \alpha d)$ $\qquad$ $f(\alpha, \beta) = L(\theta + \alpha\delta + \beta\eta)$



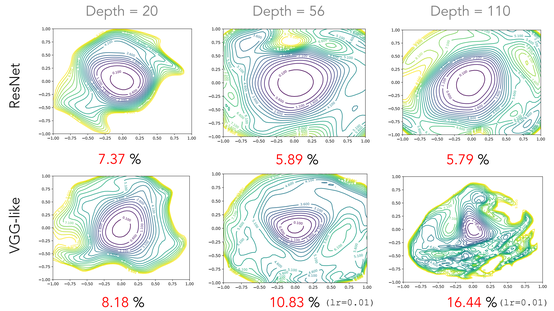
The Effect of Identity Mapping

- Residual network landscapes are dominated by wide, flat minimizers surrounded by large regions of apparent convexity that capture far-away initializers.
- Without skip connections, loss landscape is populated by many sharp minima with many small regions of convexity, creating strong dependence on initialization.



The Effect of Network Depth

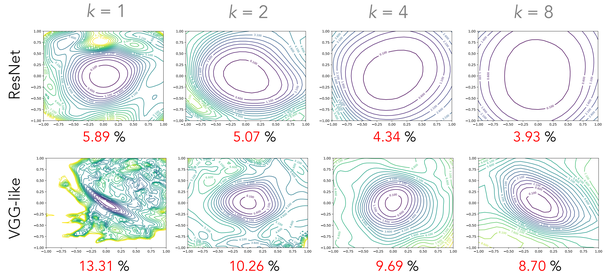
- Skip connections prevent the explosion of non-convexity that occurs when networks get deep.

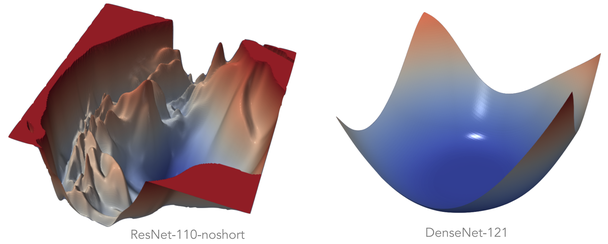


Wide Models vs. Thin Models

We compare the narrow CIFAR-optimized ResNet-56 with Wide-ResNets by multiplying the number of filters per layer by *k*.

- Sharpness correlates extremely well with the generalization error.
- Wider models have wider minima and wider regions of apparent convexity.
- Increased width prevents chaotic behavior, and skip connections dramatically widen minimizers.



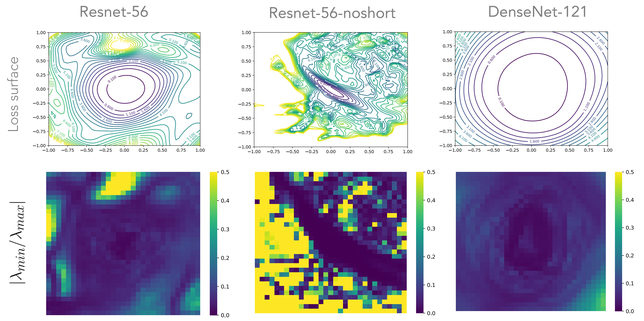


Are We Really Seeing Convexity?

Is there "hidden" non-convexity that these visualizations fail to capture?

One way to measure the level of convexity in a loss function is to compute the *principle curvatures*, which are simply *eigenvalues of the Hessian*.

- We calculate the *min* and *max* eigenvalues of the Hessian (λ_{min} and λ_{max}), and map the ratio $|\lambda_{min}/\lambda_{max}|$ across the loss surfaces studied above.
- Convex-looking regions do indeed have insignificant negative eigenvalues, while chaotic regions contain large negative curvatures.

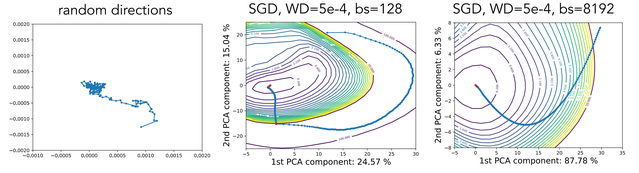


Blue color indicates a more convex region (near-zero negative eigenvalues relative to the positive eigenvalues), while yellow indicates significant levels of negative curvature.

Visualizing Optimization Paths

Effective Trajectory Plotting using PCA Directions

- Random directions fail to capture the variation in optimization trajectories.
- Given *n* training epochs, we apply PCA to the matrix $M = [\theta_0 - \theta_n; \cdots; \theta_{n-1} - \theta_n]$ and then select the two most explanatory directions.
- The descent path is **very low dimensional**: 40% ~ 90% of the variation in the descent paths lies in a space of only two dimensions.



Reference

[1] N. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy and P. T. P. Tang, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, ICLR 2017

[2] I. Goodfellow, O Vinyals, and A.M. Saxe. Qualitatively characterizing neural network optimization problems. ICLR , 2015