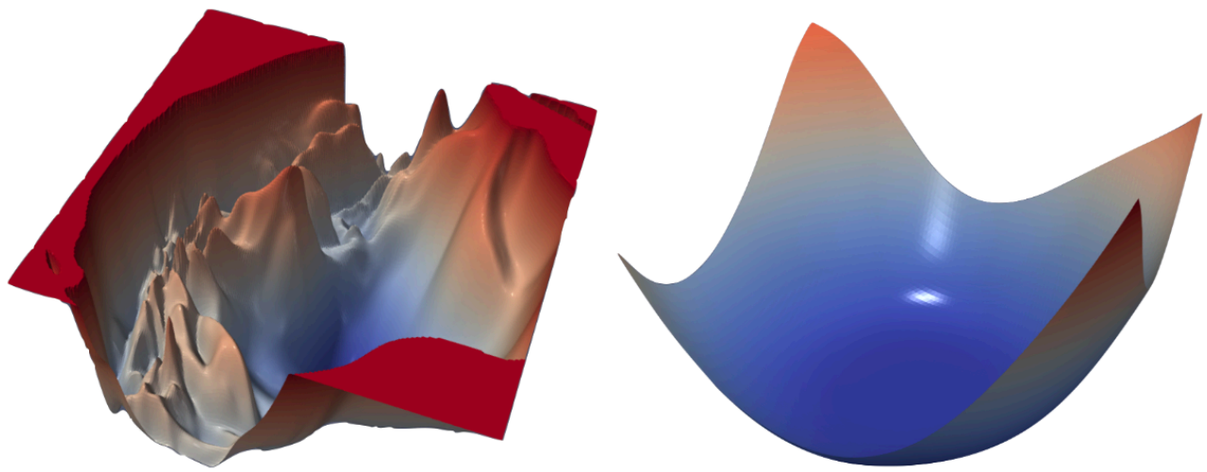


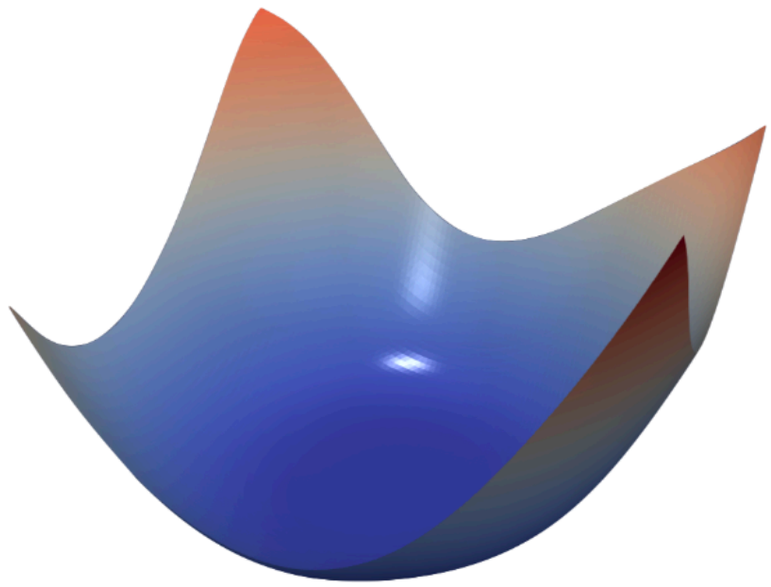
# Visualizing the Loss Landscapes of Neural Nets

Li, et al., NeurIPS, 2018.

**Presented by Wesley Maddox**

# QUESTIONS

- **What is the type of random directions are used for the plots in the paper?**
  - a) Gaussian noise
  - b) Gaussian noise + filter normalization
  - c) Gaussian noise + layer normalization
  - d) Laplace noise
- **Which network is this loss surface from?**
  - A) DenseNet-121
  - B) ResNet-164, no skip



# REFERENCE

## Visualizing the Loss Landscape of Neural Nets

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

NeurIPS, 2018, https://arxiv.org/abs/1712.09913

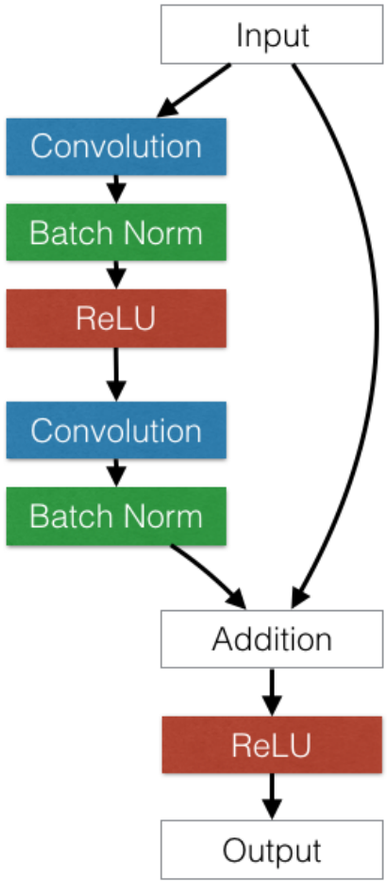
Code: https://github.com/tomgoldstein/loss-landscape

# ORGANIZATION

- Deep Neural Networks Set-up
- 1-D Visualization Methods
- 2-D Visualization Methods + Filter Normalization
- Results from Paper + Discussion

# PROBLEM SETUP


Source: https://www.cs.toronto.edu/~kriz/cifar.html




Source: http://torch.ch/blog/2016/02/04/resnets.html

# WHAT DOES A DNN LOOK LIKE?

- Composition of functions:

$$f(x; W) = \phi_{out}(W_L \phi(W_{L-1}(...\phi(W_0 x)...))$$

  - Parameters: $W = [W_i]$ Inputs: $x$

- Nonlinearities:
  - ReLU: $\phi(z) := \max\{z, 0\}$ (not differentiable everywhere)
  - Softmax: $\phi_{out}(z) := \frac{e^{z[i]}}{\sum_{i=1}^{K} e^{z[i]}}$
  - Batch-norm $BN_{\gamma,\beta}(z_i) = \gamma \frac{z_i - \bar{z}}{\sqrt{\sigma_z^2 + \epsilon}} + \beta$

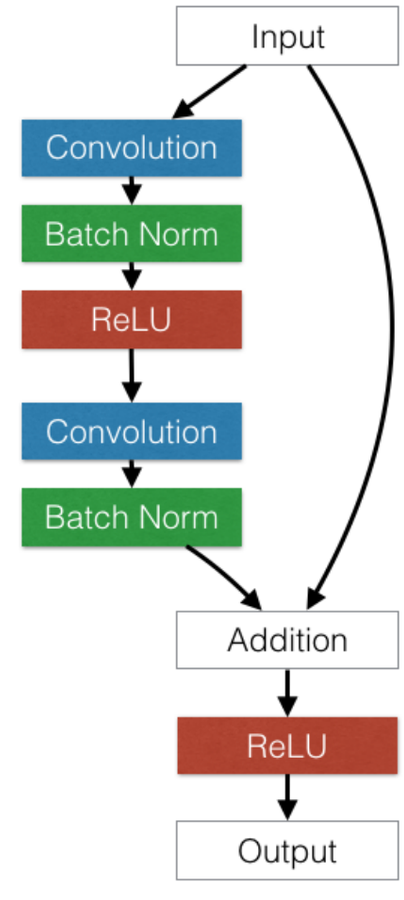
# WHAT DOES A DNN LOOK LIKE?

- Composition of functions:

$$f(x; W) = \phi_{out}(W_L \phi(W_{L-1}(...\phi(W_0 x)...)))$$

- *Residual Connections:*

$$f_i(x_{in}; W_i) = \phi(W_i x_{in}) + x_{in}$$




Source: http://torch.ch/blog/2016/02/04/resnets.html

# OBVIOUS ISSUES WITH CONVEXITY IN DANS

- ReLUs Invariance to scale:

$$W_2\phi(W_1x) = aW_2\phi(\frac{1}{a}W_1x), \quad a > 0$$

- Hessians not positive (semi-)definite:
  - Example: let $x \in \mathbb{R},\ W_1 \in \mathbb{R}^{1\times 2},\ W_2 \in \mathbb{R}^{2\times 1}$
  - And $f(x; W_1, W_2) = W_2'\phi_{ReLU}(W_1'x) = W_{21}\max(0, W_{11}x) + W_{22}\max(0, W_{12}x)$

# DNN TRAINING PROCEDURE

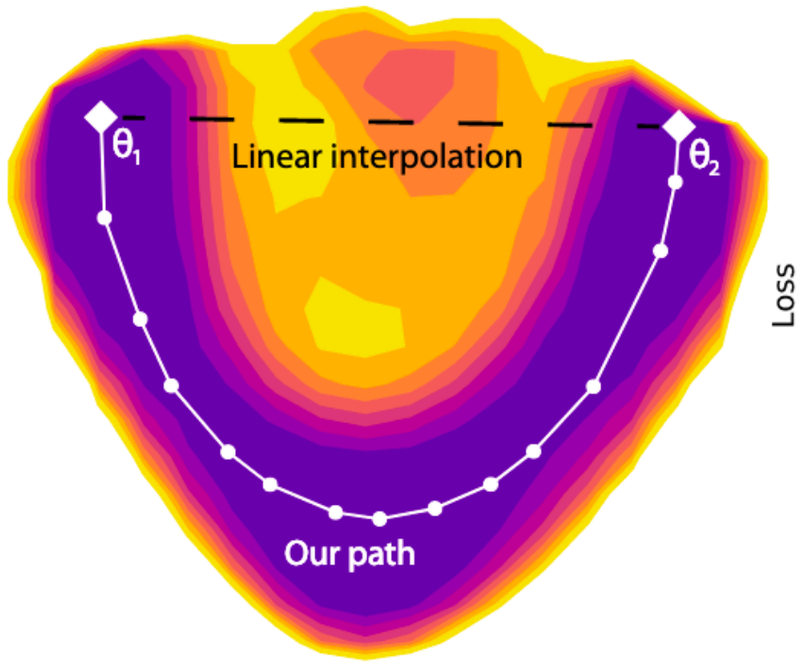
- Empirical risk minimization:

L: Loss function, convex (cross-entropy, L2 error)

$$\min_W \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; W))$$

- Stochastic (sub-?)gradient descent:

$$W_t \leftarrow W_{t-1} - \epsilon \nabla_W \left( \frac{N}{M} \sum_{i=1}^{M} L(y_i, f(x_k; W)) \right)$$




# VISUALIZING LOSS SURFACES

Slice between two minima of trained DNN

From “Essentially No Barriers in Neural Network Energy Landscape,” Draxler et al, ICML, 2018

# VISUALIZATION TECHNIQUES

- 1-D setting: $L(\alpha) = L(\text{Data}, \theta = (1-\alpha)\theta_1 + \alpha\theta_2)$

$\theta_1, \theta_2$ Independently trained nets $\quad \theta_1 := \theta_{init}$, $\theta_2 := \theta_{soln}$

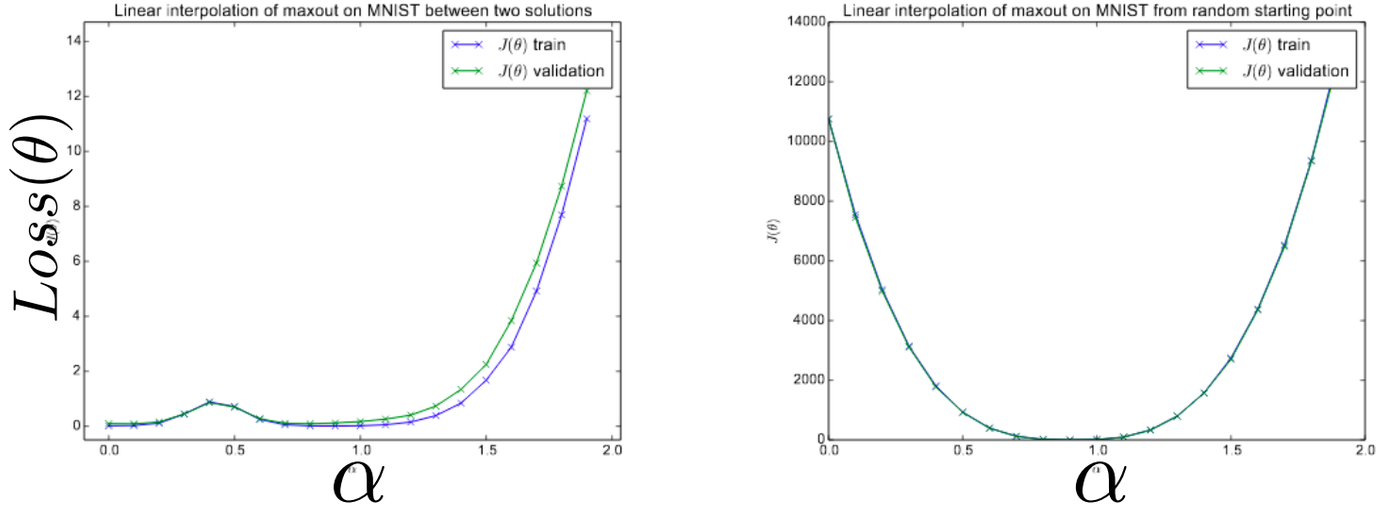



Figure 5: Here we use linear interpolation to search for local minima. Left) By interpolating between two different SGD solutions, we show that each solution is a different local minimum within this 1-D subspace. Right) If we interpolate between a random point in space and an SGD solution, we find no local minima besides the SGD solution, suggesting that local minima are rare.

From: Qualitatively Characterizing Neural Network Optimization Problems, Goodfellow et al., ICLR, 2015

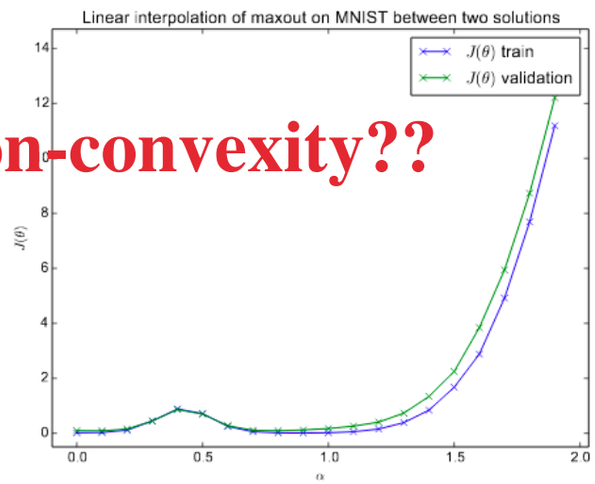
# VISUALIZATION TECHNIQUES

- 1-D setting: $L(\alpha) = L(\text{Data}, \theta = (1-\alpha)\theta_1 + \alpha\theta_2)$

$\theta_1, \theta_2$ Independently trained nets

$\theta_1 := \theta_{init}$
$\theta_2 := \theta_{soln}$

**Where is the non-convexity??**



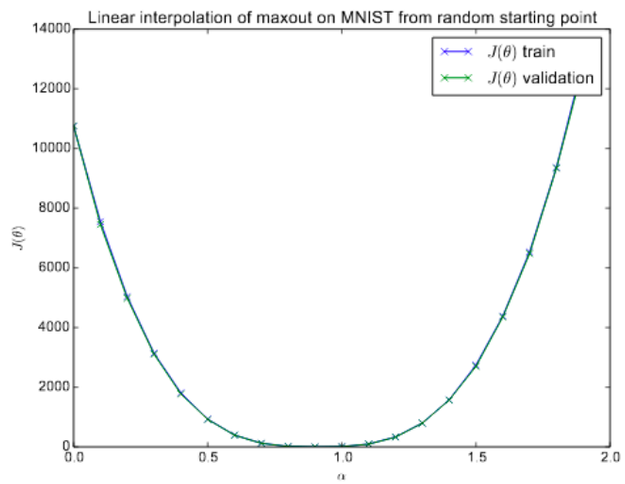



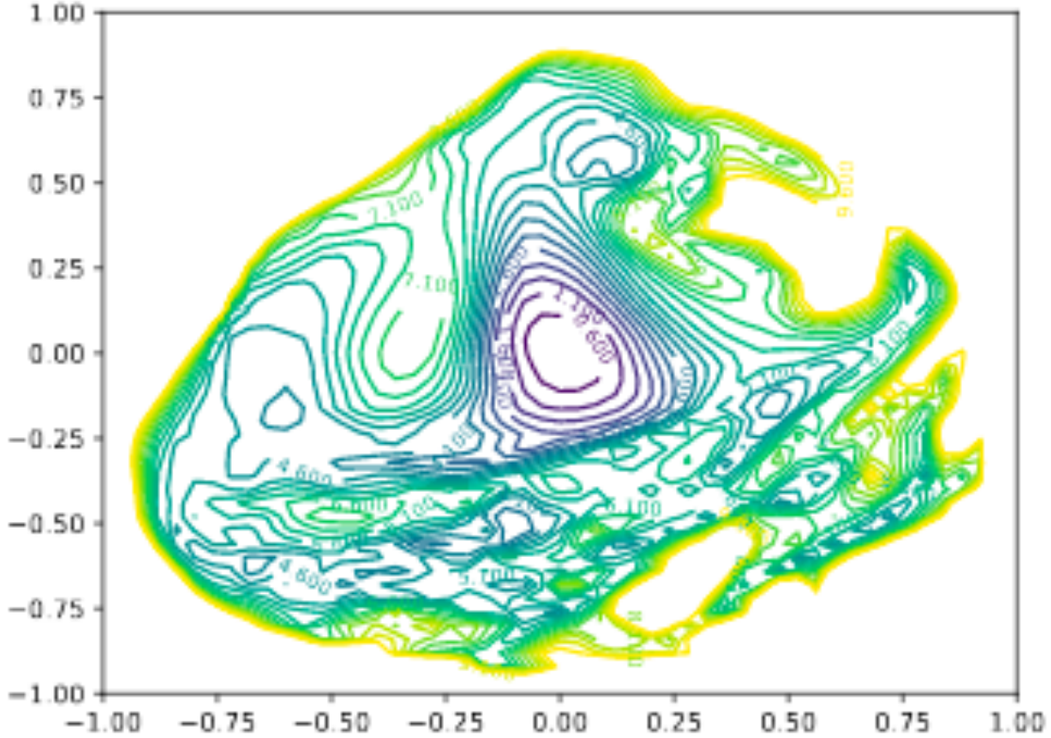

Figure 5: Here we use linear interpolation to search for local minima. Left) By interpolating between two different SGD solutions, we show that each solution is a different local minimum within this 1-D subspace. Right) If we interpolate between a random point in space and an SGD solution, we find no local minima besides the SGD solution, suggesting that local minima are rare.

From: Qualitatively Characterizing Neural Network Optimization Problems, Goodfellow et al., ICLR, 2015

# VISUALIZATION TECHNIQUES

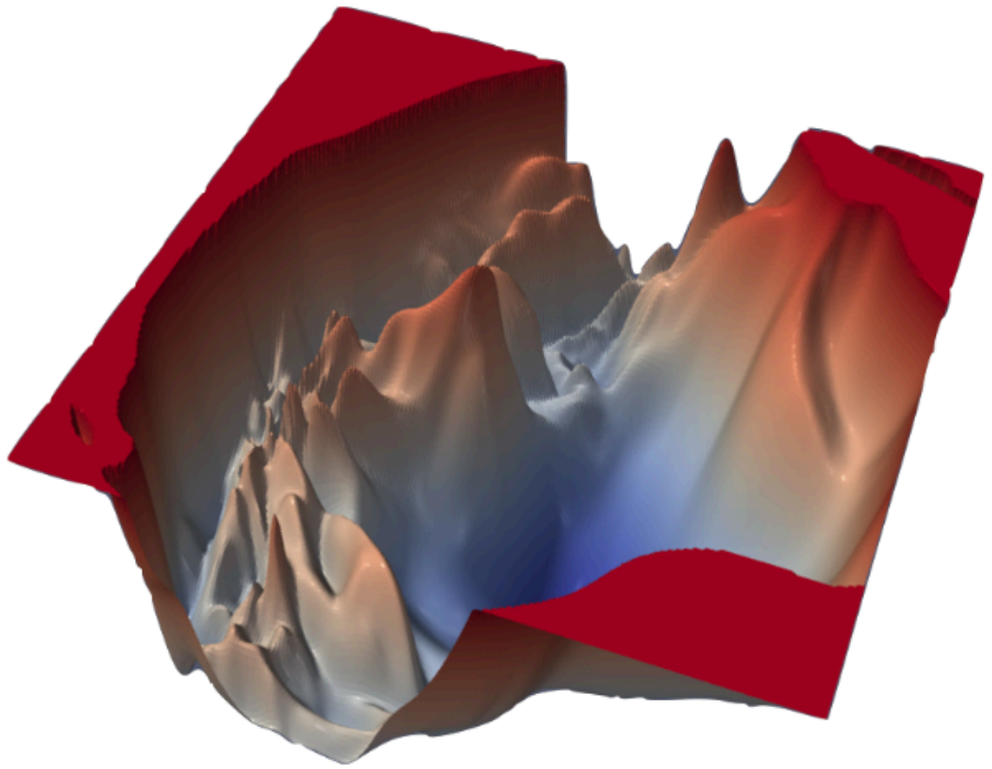
- 2D Setting: $f(\alpha, \beta) := L(\text{Data}, \theta = \theta^* + \alpha\delta + \beta\eta)$

$$\delta, \eta \sim N(0, I)$$




(f) ResNet-110-NS, 16.44%

Figure 5f)



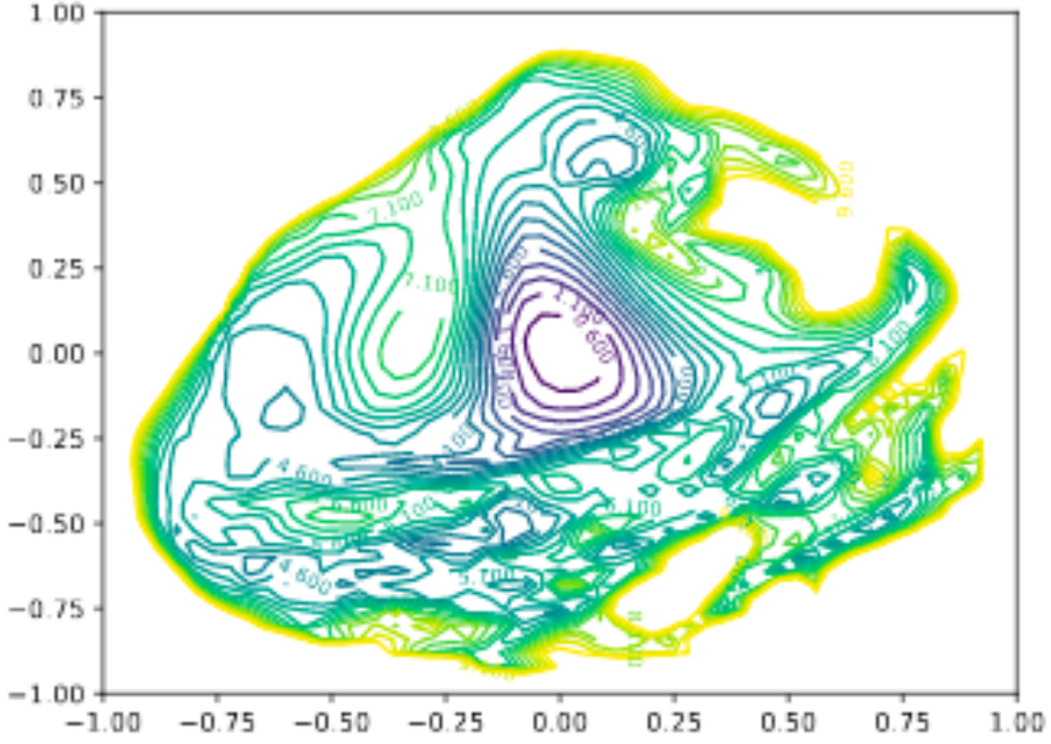
(a) ResNet-110, no skip connections

Figure 4a)

# VISUALIZATION TECHNIQUES

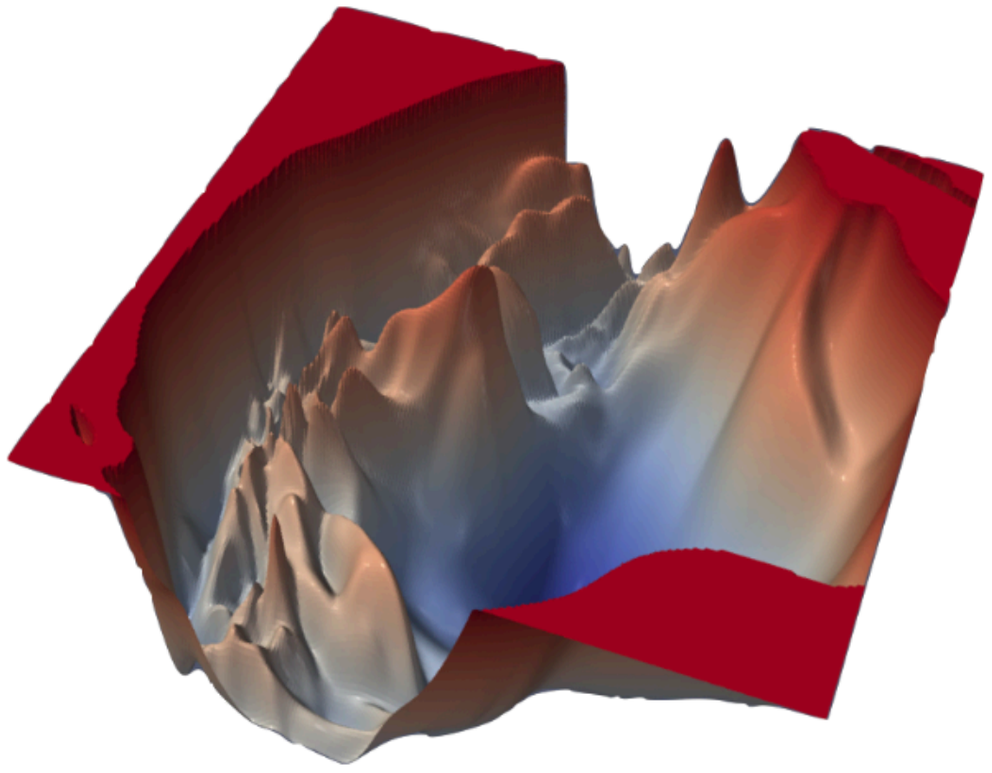
- 2D Setting: $f(\alpha, \beta) := L(\text{Data}, \theta = \theta^* + \alpha\delta + \beta\eta)$

$$\delta, \eta \sim N(0, I)$$ ← Not quite




(f) ResNet-110-NS, 16.44%

Figure 5f)



(a) ResNet-110, no skip connections

Figure 4a)

# FILTER NORMALIZATION

- Recall scale invariance: $W_2\phi(W_1x) = aW_2\phi(\frac{1}{a}W_1x), \quad a > 0$

- Filter normalization: $d_{i,j} \leftarrow \frac{d_{i,j}}{||d_{i,j}||}||\theta_{i,j}||$
  - In the *i*th layer of the network, normalize the *j*th row to have norm the same as the parameters

```
if norm == 'filter':
    # Rescale the filters (weights in group) in 'direction' so that each
    # filter has the same norm as its corresponding filter in 'weights'.
    for d, w in zip(direction, weights):
        d.mul_(w.norm()/(d.norm() + 1e-10))
```

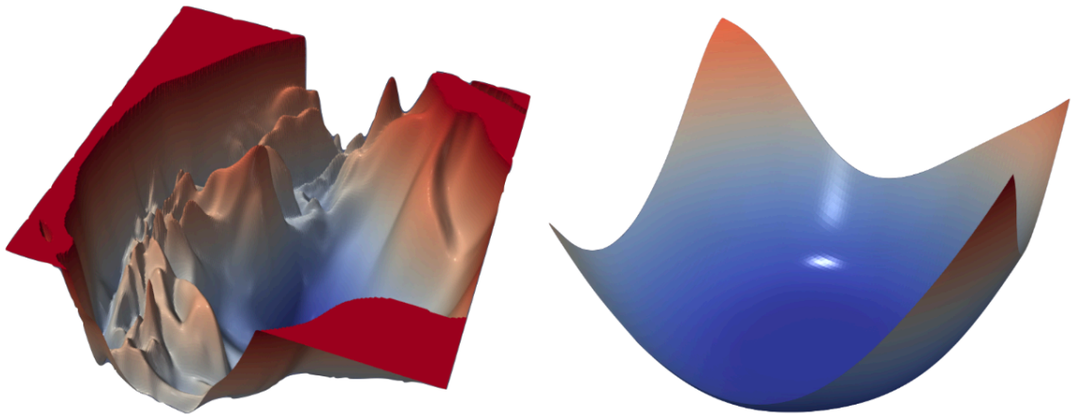
Code: https://github.com/tomgoldstein/loss-landscape/blob/master/net_plotter.py

# FILTER NORMALIZATION

- Recall scale invariance:

- Filter normalization: $d_{i,j} \leftarrow \frac{d_{i,j}}{||d_{i,j}||} ||\theta_{i,j}||$
  - In the *i*th layer of the network, normalize the *j*th row to have norm the same as the parameters

- **Is this a trustworthy technique for comparing several models?**



(a) ResNet-110, no skip connections (b) DenseNet, 121 layers

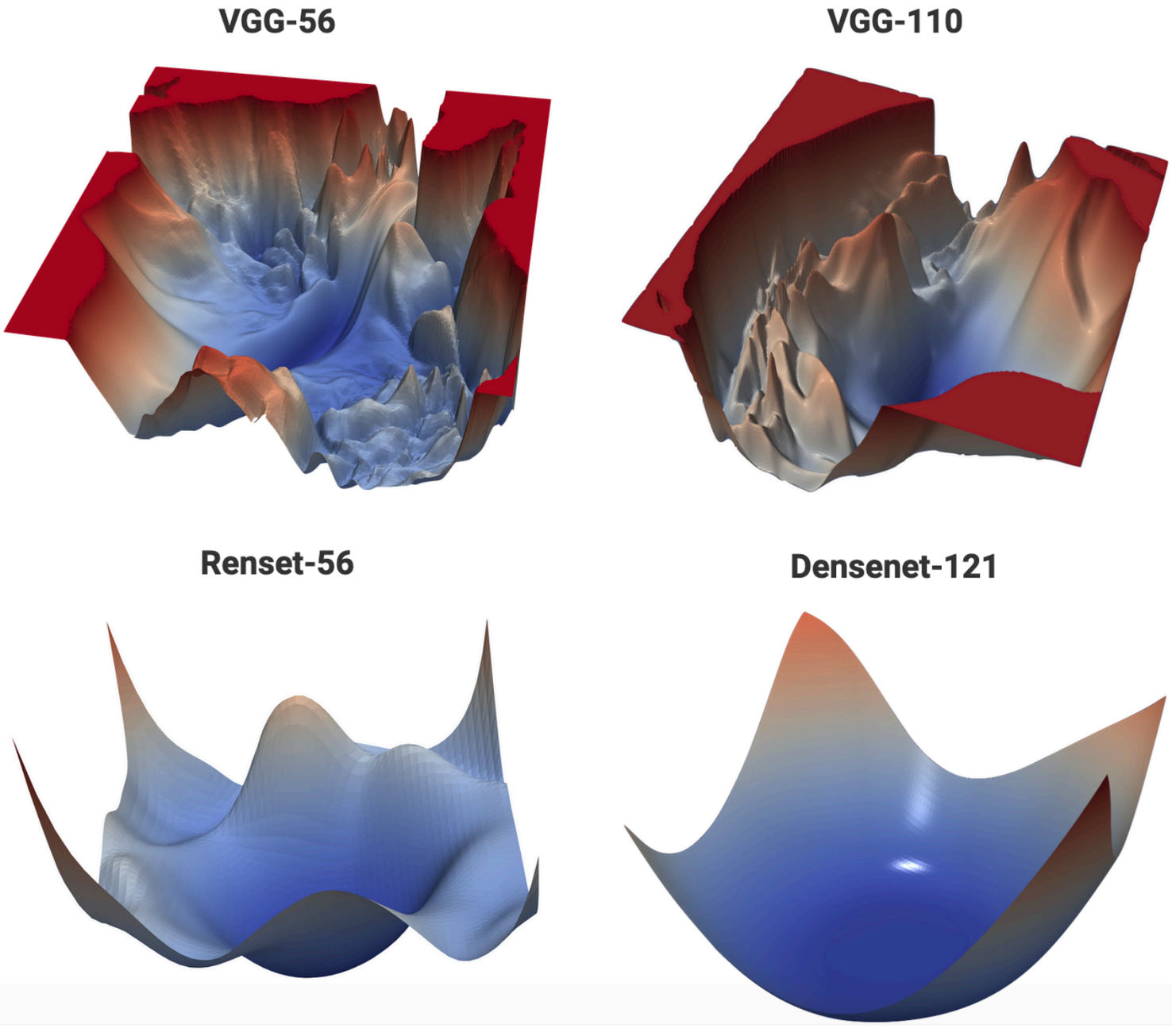
Figure 4: The loss surfaces of ResNet-110-noshort and DenseNet for CIFAR-10.

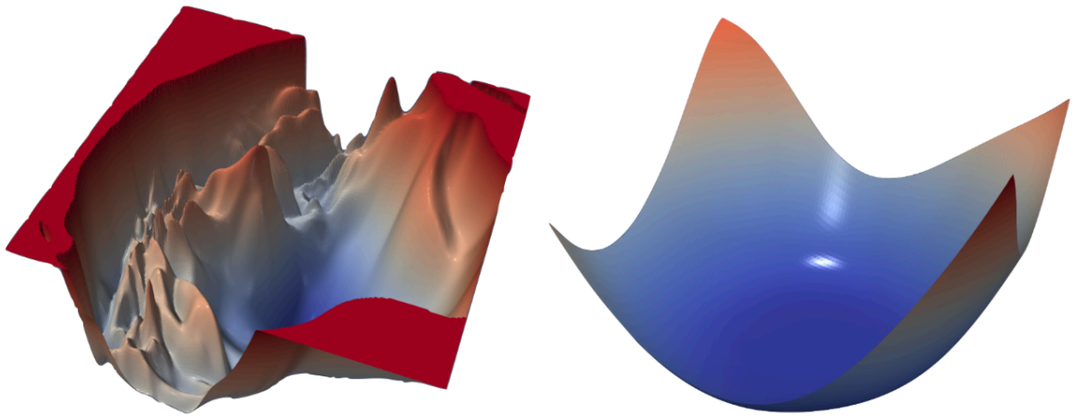
# RESULTS

# RESULTS

http://www.telesens.co/loss-landscape-viz/viewer.html

# RESULTS





(a) ResNet-110, no skip connections (b) DenseNet, 121 layers

Figure 4: The loss surfaces of ResNet-110-noshort and DenseNet for CIFAR-10.

# VISUALIZATION-LEVEL INSIGHTS

What can this tell us (heuristically) about DNNs?

# EFFECT OF RESIDUAL CONNECTIONS

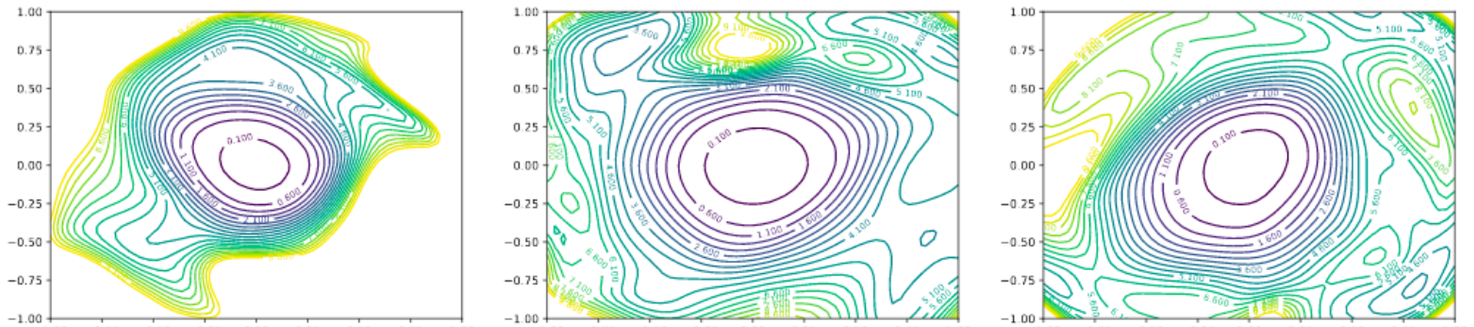
- How should residual connections affect optimization procedure & loss surface?

- How should depth of network affect training?

# EFFECT OF RESIDUAL CONNECTIONS

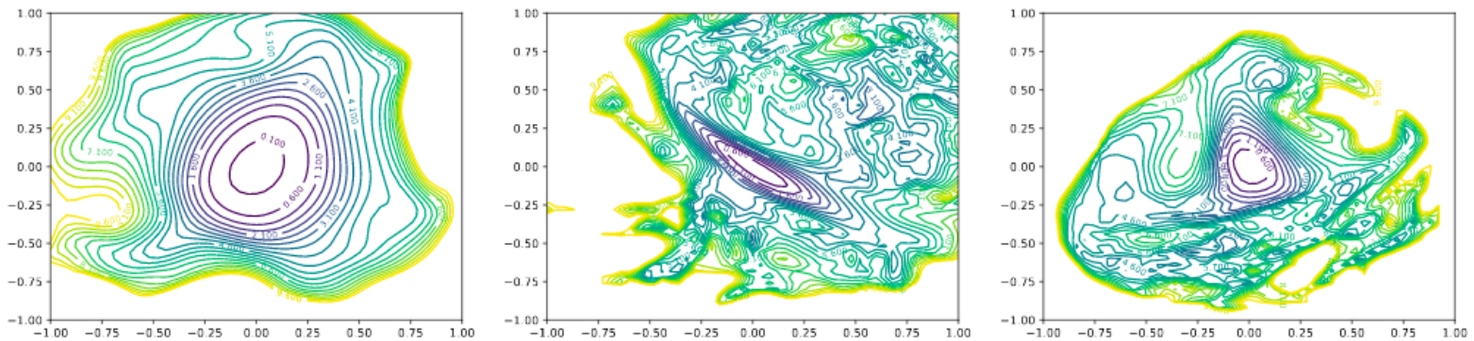
Depth

**Residual Connections**



(a) ResNet-20, 7.37% (b) ResNet-56, 5.89% (c) ResNet-110, 5.79%

**No Residual Connections**



(d) ResNet-20-NS, 8.18% (e) ResNet-56-NS, 13.31% (f) ResNet-110-NS, 16.44%

Figure 5: 2D visualization of the loss surface of ResNet and ResNet-noshort with different depth.

# WIDTH + NUMBER OF PARAMETERS

- How does width (size of $W_i$) affect training?

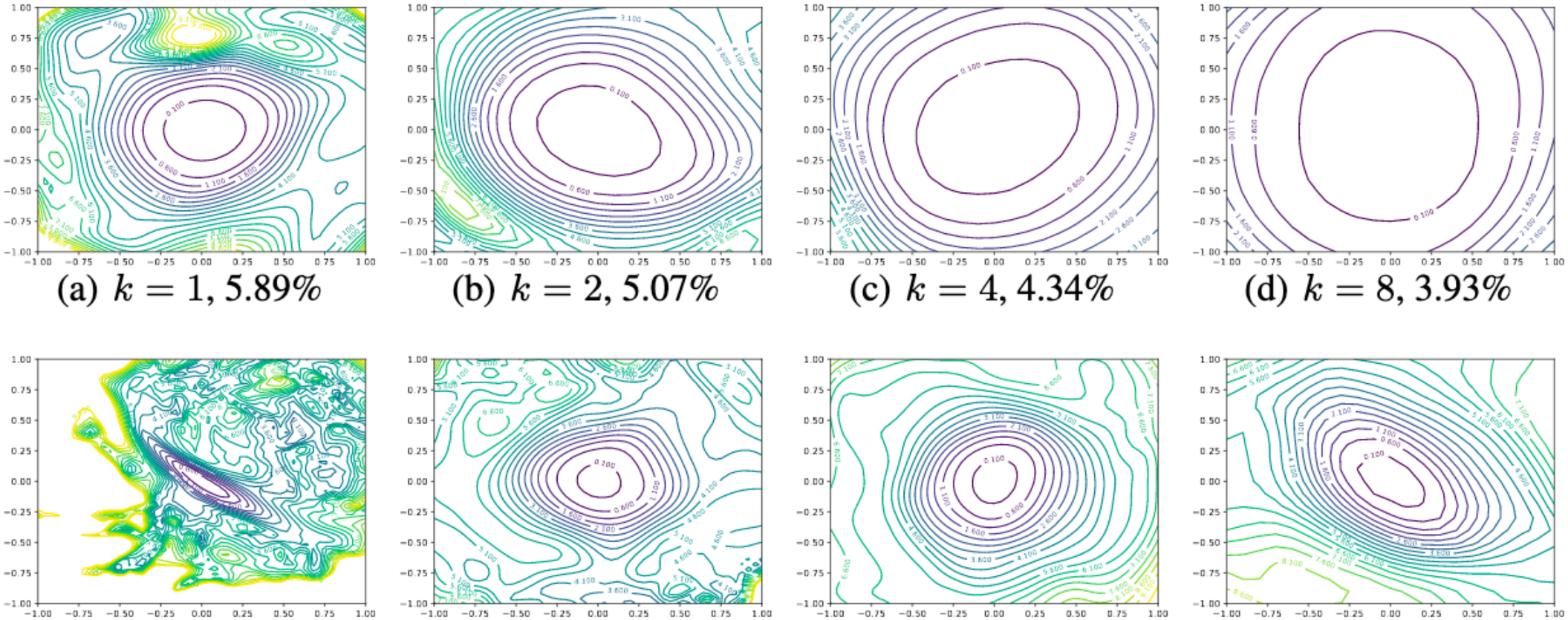
$$f(x; W) = \phi_{out}(W_L \phi(W_{L-1}(...\phi(W_0 x)...))$$

- Increasing width increases over-parameterization
- Infinite width convolutional DNNs -> “Gaussian processes”
  - “Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes,” Novak et al, ICLR, 2019.

# DENSE CONNECTIONS + WIDTH OF MODEL

Residual connections to every layer

k: k times standard width of convolutional layers



(a) $k = 1$, 5.89% (b) $k = 2$, 5.07% (c) $k = 4$, 4.34% (d) $k = 8$, 3.93%

(e) $k = 1$, 13.31% (f) $k = 2$, 10.26% (g) $k = 4$, 9.69% (h) $k = 8$, 8.70%

Figure 6: Wide-ResNet-56 on CIFAR-10 both with shortcut connections (top) and without (bottom). The label $k = 2$ means twice as many filters per layer. Test error is reported below each figure.

# SGD TUNING PARAMETERS

- Weight decay: $\min_W \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; W)) \quad +||W||_2$

- Batch size: $W_t \leftarrow W_{t-1} - \epsilon \nabla_W \left( \frac{N}{M} \sum_{i=1}^{M} L(y_i, f(x_k; W)) \right)$

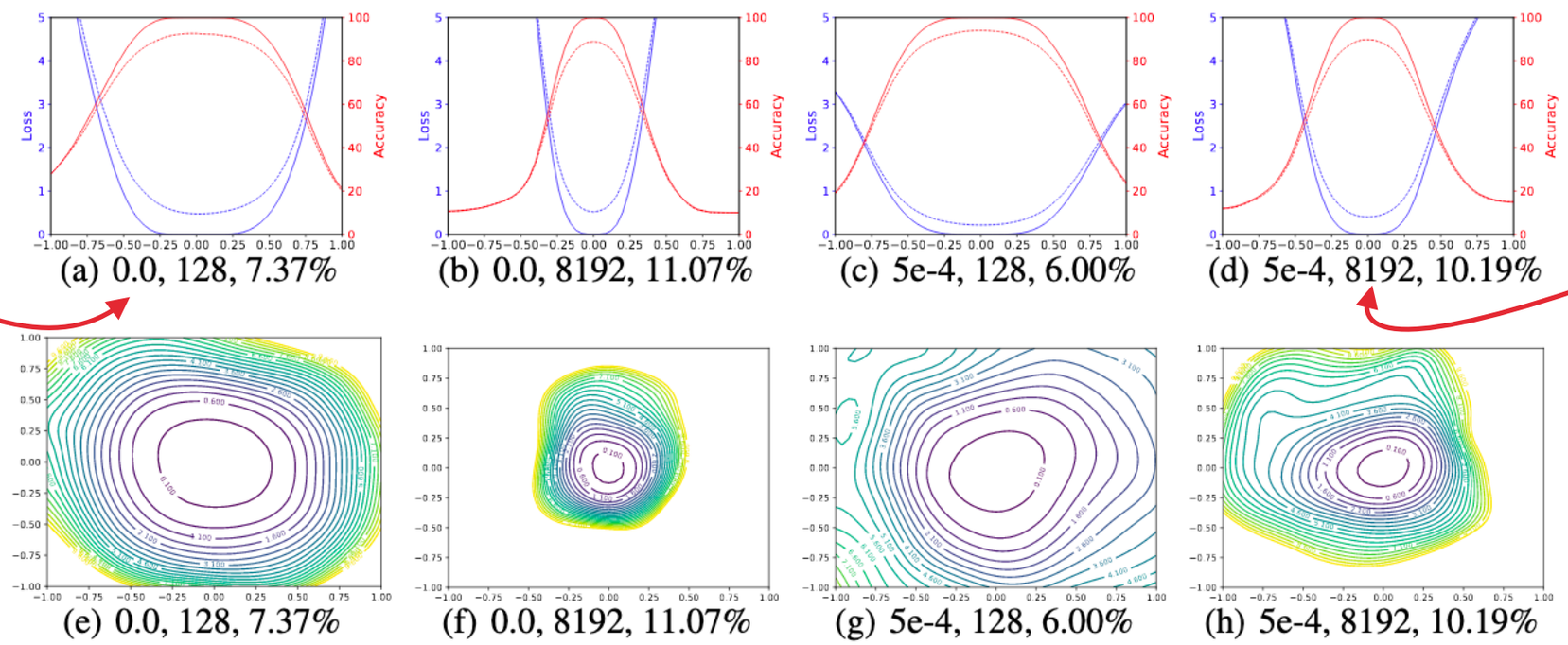
- Open question: how do these parameters affect optimization and "generalization error"?

# WHAT ROLE DOES WEIGHT DECAY PLAY?

Model: VGG, Dataset: CIFAR10

Weight decay

Batch size




Test accuracy

Figure 3: The 1D and 2D visualization of solutions obtained using SGD with different weight decay and batch size. The title of each subfigure contains the weight decay, batch size, and test error.

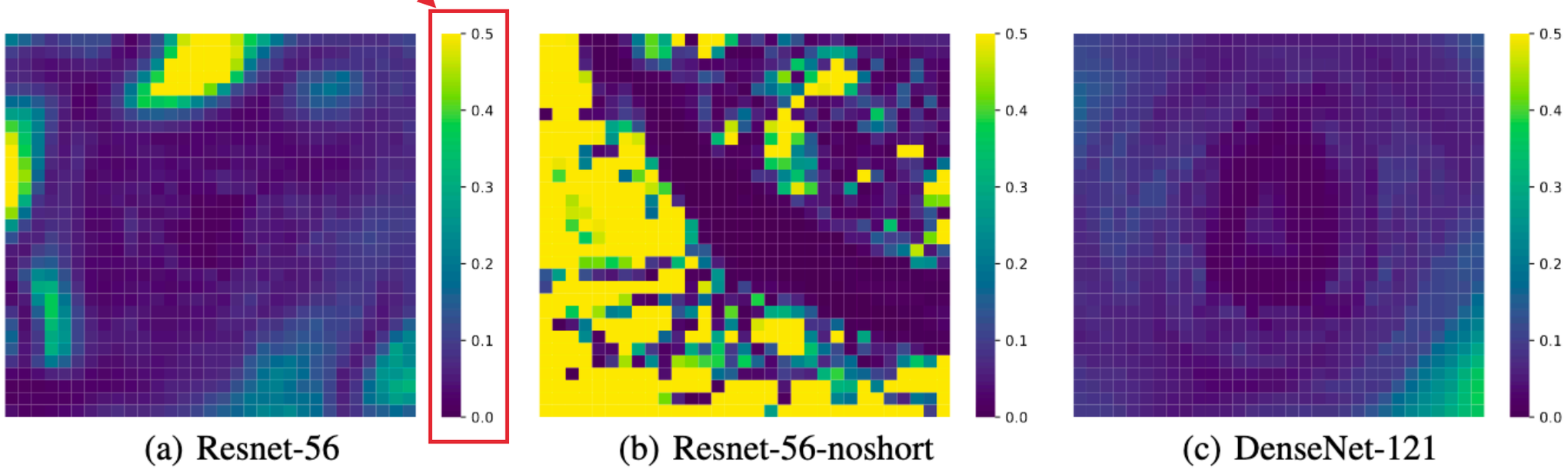
# IS THIS METHOD JUST MISSING SIGNIFICANT NON-CONVEX REGIONS?

# EIGENVALUES OF TRAIN HESSIAN

Negative eigenvalues in a local region are "really small"…

Consider eigenvalues of train Hessian matrix: $\nabla^2_W \left( \frac{1}{N} \sum_{i=1}^{N} L(y_i; f(x_i, W)) \right)$

$|\lambda_{min}/\lambda_{max}|$



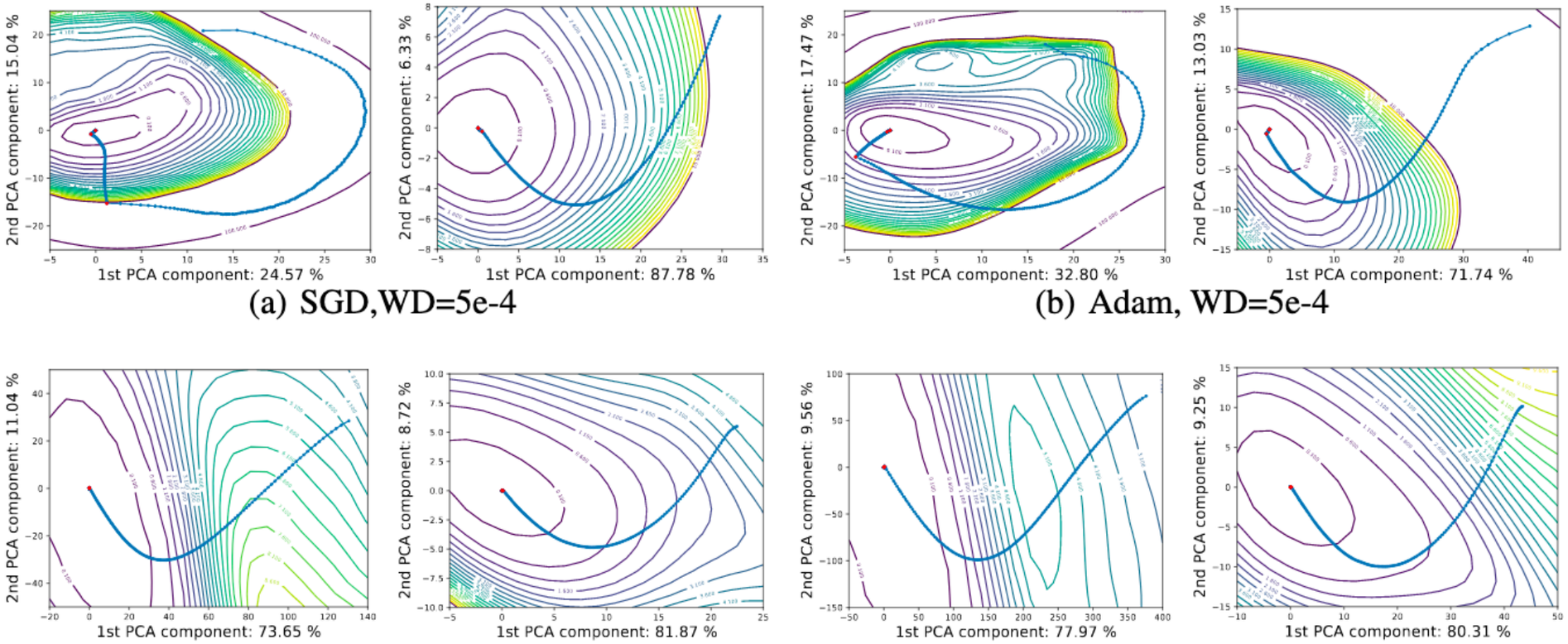

(a) Resnet-56 (b) Resnet-56-noshort (c) DenseNet-121

Figure 7: For each point in the filter-normalized surface plots, we calculate the maximum and minimum eigenvalue of the Hessian, and map the ratio of these two.

At (0,0) for a): $\lambda_{max} \approx 3600, \quad \lambda_{min} \approx -300$

# DOES SGD OPTIMIZE IN ONLY A FEW DIRECTIONS?

Perform PCA on iterates of SGD and plot trajectory along first 2 directions



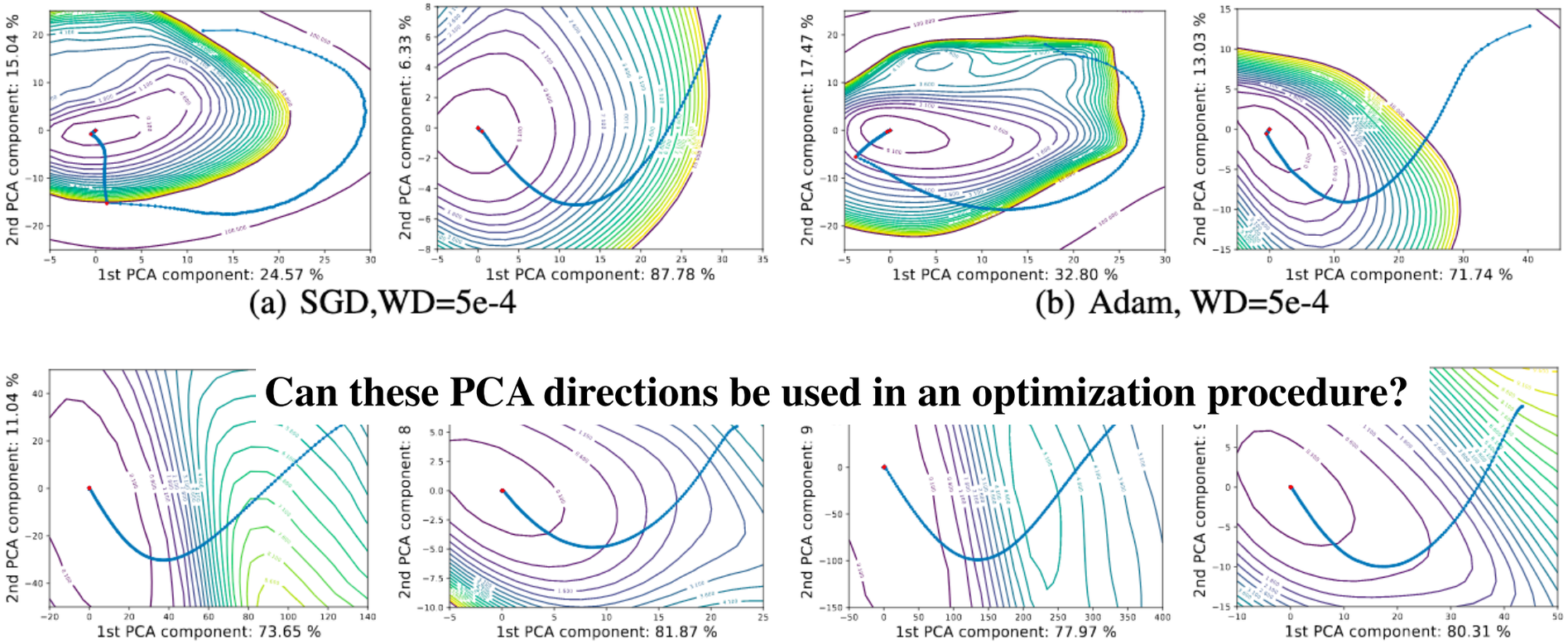

(a) SGD,WD=5e-4 (b) Adam, WD=5e-4

(c) SGD, WD=0 (d) Adam,WD=0

Figure 9: Projected learning trajectories use normalized PCA directions for VGG-9. The left plot in each subfigure uses batch size 128, and the right one uses batch size 8192.

# DOES SGD OPTIMIZE IN ONLY A FEW DIRECTIONS?

Perform PCA on iterates of SGD and plot trajectory along first 2 directions




(a) SGD,WD=5e-4 (b) Adam, WD=5e-4

**Can these PCA directions be used in an optimization procedure?**

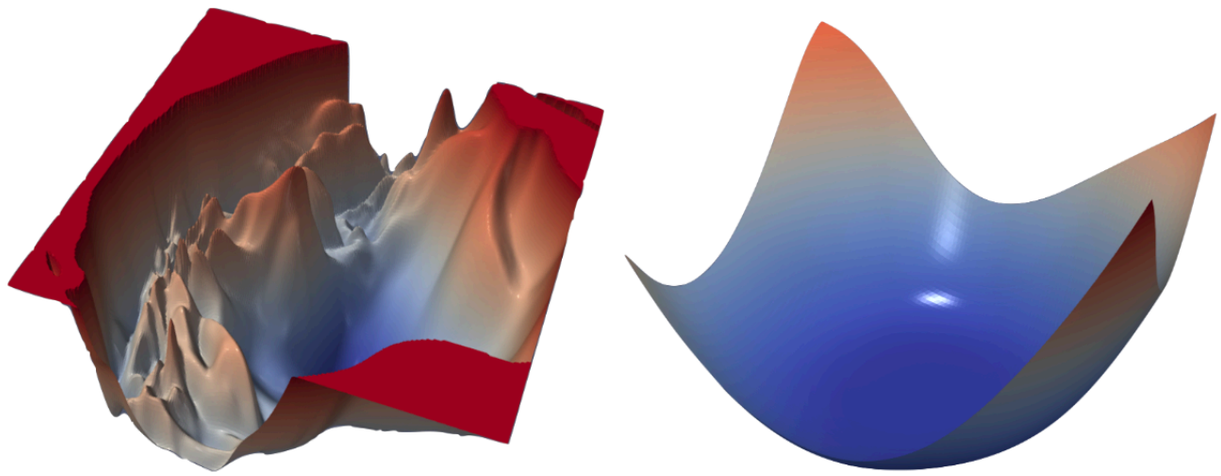
(c) SGD, WD=0 (d) Adam,WD=0

Figure 9: Projected learning trajectories use normalized PCA directions for VGG-9. The left plot in each subfigure uses batch size 128, and the right one uses batch size 8192.

# TAKEAWAYS

- Why are DNNs "easy" to train?
  - DNN loss surfaces look relatively well behaved in many cases
    - But, can we give a more rigorous definition of "relatively well-behaved"?
      - Can we prove why optimization works well in this setting?
- What is the interplay between the model structure and SGD hyper-parameter settings?
  - **Explicit regularization**: residual connections, depth, width…
  - **Implicit regularization**: batch size, weight decay

# TAKEAWAYS



(a) ResNet-110, no skip connections (b) DenseNet, 121 layers

Figure 4: The loss surfaces of ResNet-110-noshort and DenseNet for CIFAR-10.

Can we use filter normalization techniques to make b) more “well-behaved”?

Or design more networks that look like b)

# REFERENCES

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems* (pp. 6391-6401).

Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. (2018, July). Essentially No Barriers in Neural Network Energy Landscape. In *International Conference on Machine Learning* (pp. 1308-1317).

Goodfellow, I. J., Vinyals, O., & Saxe, A. M. (2015). Qualitatively characterizing neural network optimization problems. ICLR, *arXiv preprint arXiv:1412.6544*.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Hron, J., ... & Sohl-dickstein, J. (2019). Bayesian deep convolutional networks with many channels are gaussian processes, ICLR.

*Architecure*:

**(ResNet/PreResNet)** Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition.In CVPR, 2016.
**(DenseNet)** Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolu-tional networks. In CVPR, 2017.
**(VGG)** Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale ImageRecognition. In ICLR, 2015.