Deep Learning Project Report
Visualizing the Loss Landscape of Neural Nets

Meer Suri, Chandni Akbar, Siffi Singh, Joseph Chataignon, Raphaël Boulanger, Pierric Meunier

1 Introduction and Motivation

The ideas behind our final project are inspired from [4]. The authors use a formula to visualize in 2-D and 3-D the loss function of neural networks, as explained later in this report. We decided to use the same method to also visualize the loss in situations where we change the architecture/parameters of a network, e.g. by adding regularization,adding skip connections, adding more layers or modifying the batch size, to see how these changes affect the loss landscape.
By studying these effects, we aimed to understand the following

1. Certain network architecture designs produce loss functions that train easier.
2. Well-chosen training parameters produce minimizers that generalize better. We tried to explore how network architecture affects the loss landscape, and how training parameters affect the shape of minimizers.

2 Code details

Self-implemented: All models used, application of visualization technique.
From GitHub: script to convert calculated loss values to Paraview format, use of ParaView for 3-D rendering of loss surfaces.

3 Loss Function Visualization Technique

To visualize the underlying loss landscape of a neural network, we used the technique explained in the paper. The technique is briefly described below:
The neural network parameters excluding biases are denoted by θ which may represent weights of all layers or the weights of a subset layers, stacked together. These are high dimensional tensors that can be interpreted as directions in this high dimensional space

1. Define two direction vectors, δ and η with the same dimensions as θ, drawn from a Gaussian distribution with zero mean and a scale of the same order of magnitude as the variance of layer weights.
2. Choose a center point θ^* and add a linear combination of δ and η to obtain a loss that is a function of the contribution of the two random direction vectors .

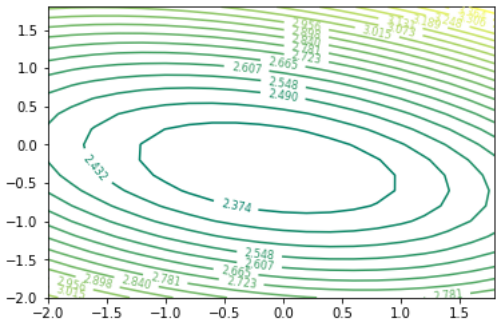
$$f(\alpha, \beta) = L(\theta^* + \alpha\delta + \beta\eta) \tag{1}$$

3. Define a grid of points to evaluate the loss on i.e. range of values for α and β for which $f(\alpha, \beta)$ is evaluated and stored.
4. Interpolate over the grid to obtain a higher resolution visualization over large slices of the weight space to see how the network design affects a non-convex structure.

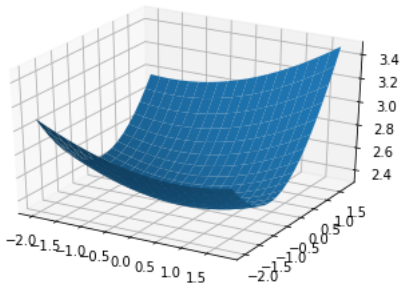
We applied this method to visualize the loss of various networks after changing parameters or the architecture design in order to see the changes in the loss landscape. We also used different datasets to test the visualization on different kinds of data. We used in total 4 different datasets : MNIST (black and white handwritten digits, 10 classes), CIFAR-10 (colour images of 10 classes), text data from the Shakespeare dataset and the seismic bumps dataset.

4 Effect of regularization on loss landscape

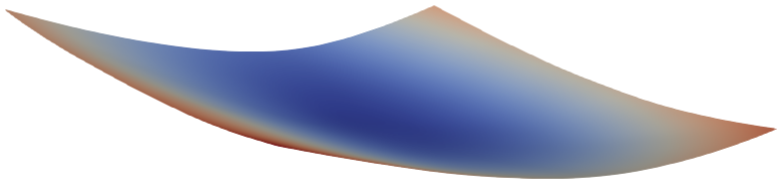
To test the effect of regularization on the loss landscape, we used a CNN with 1 convolution layer, trained on the CIFAR-10 dataset, and tested first without regularization, and then with L1 and L2 regularization separately. We present our results using 2-D loss contour plots and 3-D surface plots and 3-D renderings using ParaView . In the latter one, red color indicates a higher z value while blue color indicates a lower one.



(a) Loss contour plot

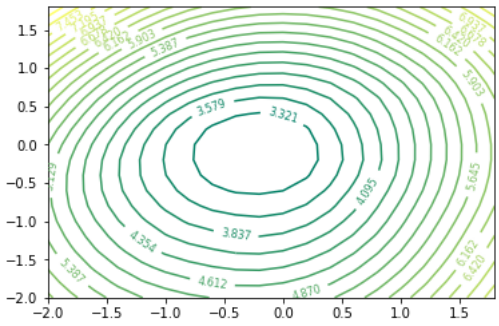


(b) Loss surface plot

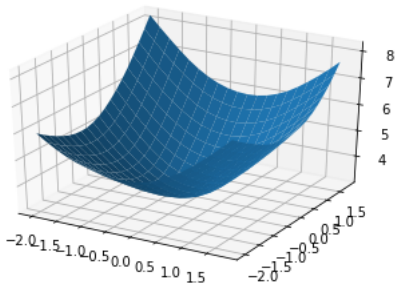


(c) Loss surface 3D rendering

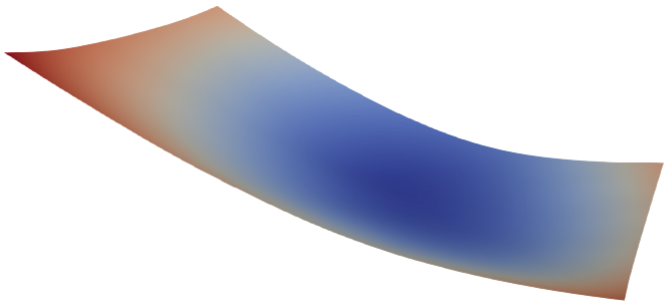
Figure 1: Loss landscape visualizations for a single convolution layer network without regularization



(a) Loss contour plot

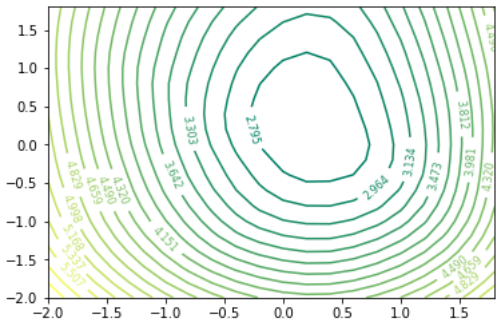


(b) Loss surface plot

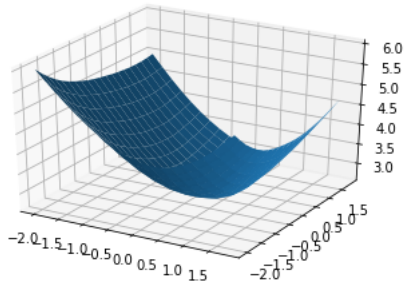


(c) Loss surface 3-D rendering

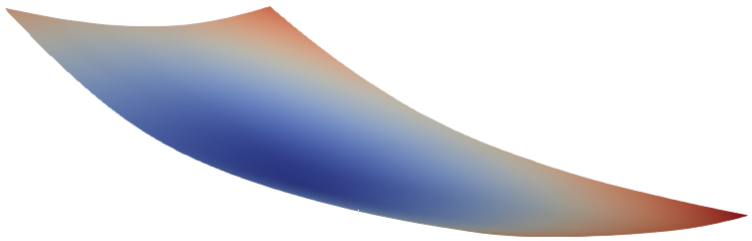
Figure 2: Loss landscape visualizations for a single convolution layer network with $L1$ regularization $\lambda = 0.01$



(a) Loss contour plot



(b) Loss surface plot



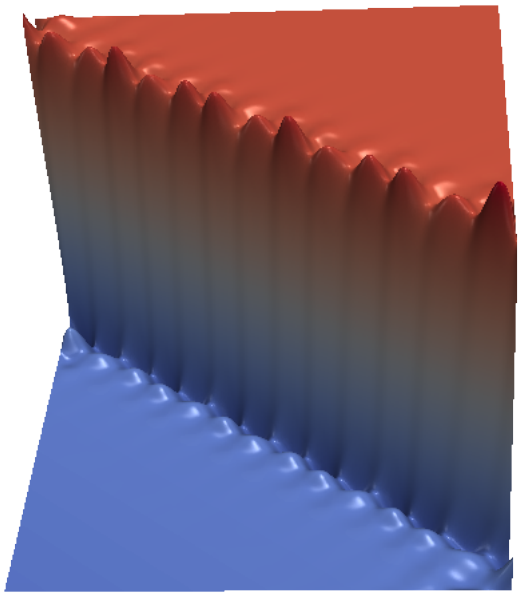
(c) Loss surface 3-D rendering

Figure 3: Loss landscape visualizations for a single convolution layer network with $L2$ regularization $\lambda = 0.01$

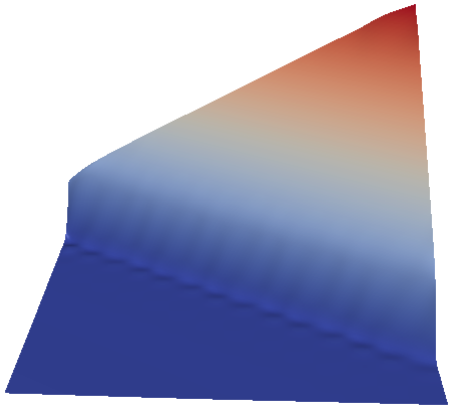
The experiments on the effect of regularization were inconclusive as no significant difference was consistently observable in the loss landscapes. Further testing is required to determine if there is a significant effect of regularization on the loss landscape.

5 Effect of depth on the loss landscape

To test the effect of the depth of a neural network on the loss landscape, we used similar networks but changed the number of hidden layers. We applied this to multiple datasets.
Below are the results we obtained for a feed-forward network we implemented ourselves and trained on the seismic bumps dataset :



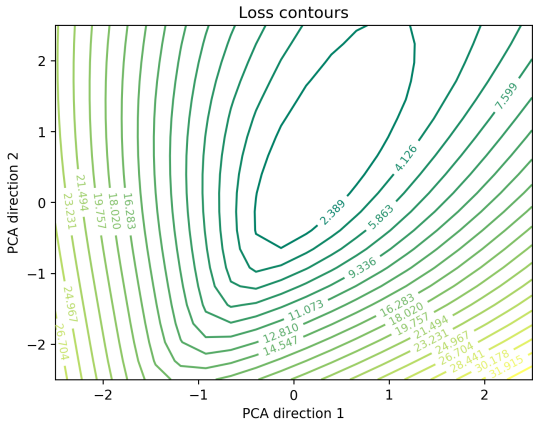
(a) Loss surface 3D rendering for the 1 layer network



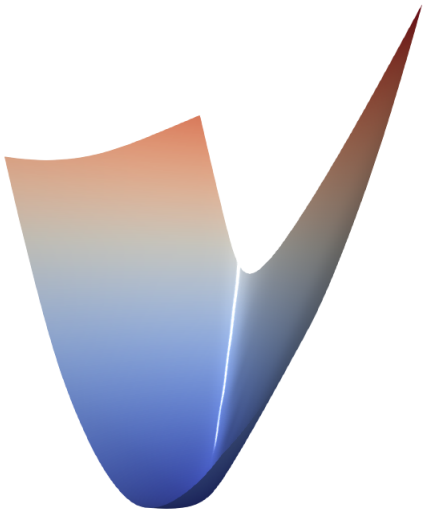
(b) Loss surface 3D rendering for the 3 layer network

Figure 4: Loss landscape visualizations for feed-forward networks trained on the Seismic bumps dataset

The network with 3 layers cannot be called deep but the difference in its loss landscape was consistent so we decided to include it in our result. In general, we expect the complexity of the landscape to increase with depth of the network [1]and the following experiments produced results in line with our expectations:

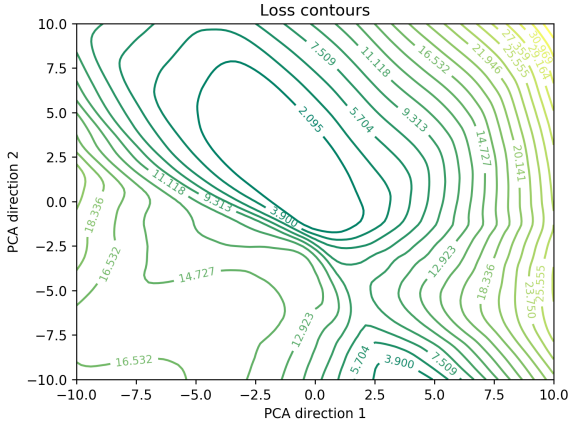


(a) Loss surface contour plot

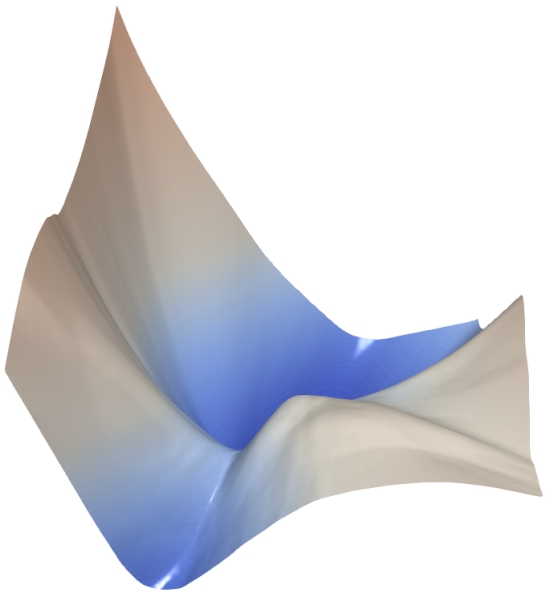


(b) Loss surface 3D rendering

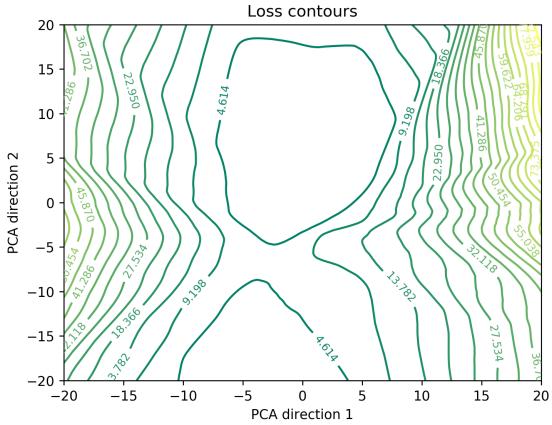
Figure 5: Loss landscape visualizations for a single convolution layer network with 4 layers, trained on MNIST



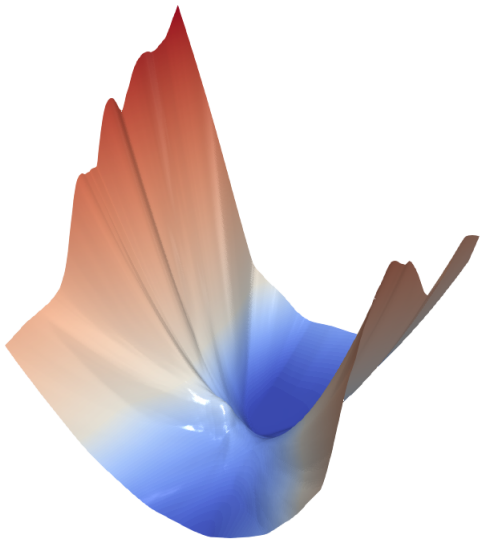
(a) Loss surface contour plot



(b) Loss surface 3D rendering



(c) Loss surface contour plot



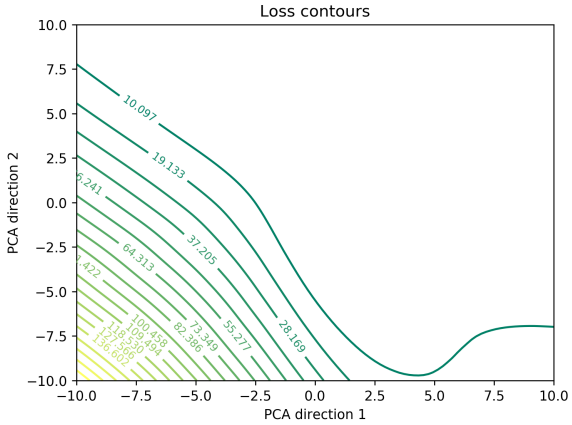
(d) Loss surface 3D rendering

Figure 6: Loss landscape visualizations for a deep network with 16 layers, trained on MNIST

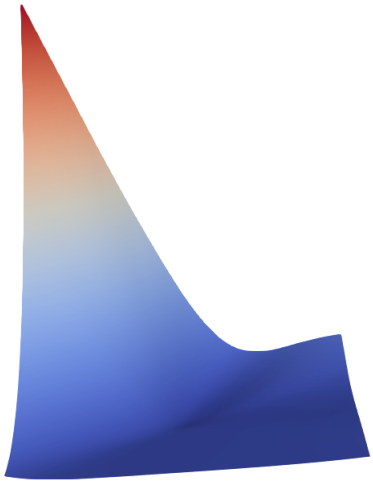
The deeper network shows a significantly more complex loss landscape even when visualized from different weight layers.

6 Effect of skip connections on the loss landscape

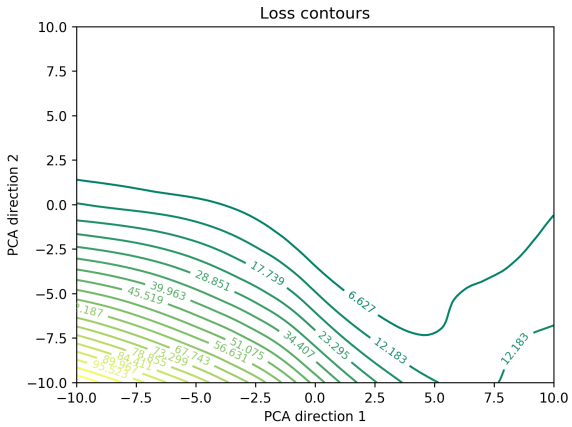
For this part, we tested the effect of adding skip connections in a neural network on the loss landscape using a 16 layers deep feed-forward network trained on the MNIST dataset. It is well known that ResNets [3] are easier to train as they have skip connections in their design, but the effect of these on the loss landscape is not clear.



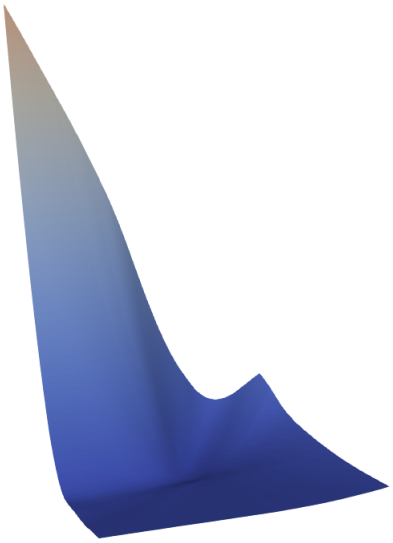
(a) Loss surface contour plot



(b) Loss surface 3D rendering



(c) Loss surface contour plot



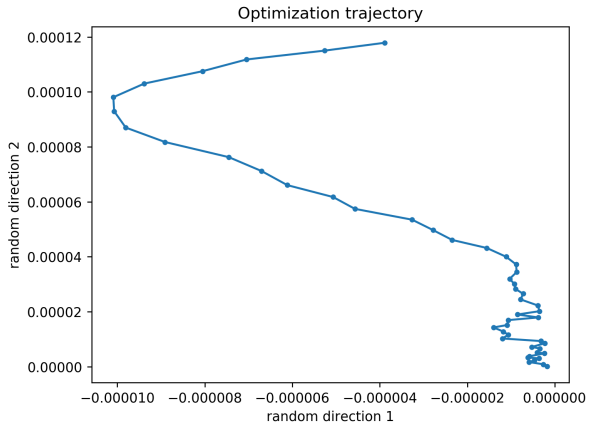
(d) Loss surface 3D rendering

Figure 7: Effect of adding 1 skip connection from layer 2 to layer 14 of the 16 layer network

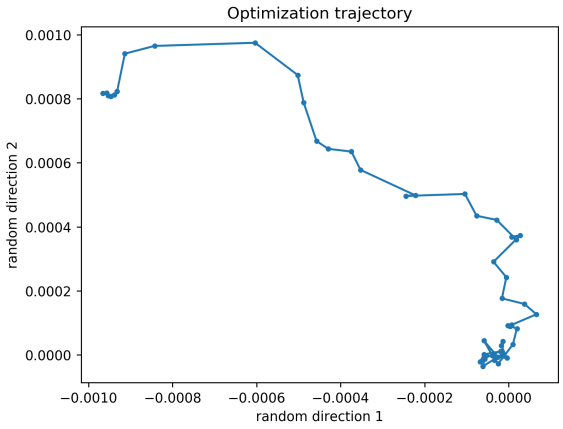
The results of these experiments lend evidence to the hypothesis that the addition of skip connections leads to simplification of the loss landscape by connections.

7 Visualization of the optimization trajectory

Visualization of the optimization trajectory is provides valuable insight into the problems encountered due to landscape of the loss, hence we wanted to perform some experiments on this. The first thing we learnt was that visualization by projection on random directions fails :



(a) Optimization trajectory 1



(b) Optimization trajectory 2

Figure 8: Optimization trajectory visualizations using random directions for the 16 layer feed-forward network trained on MNIST using SGD.

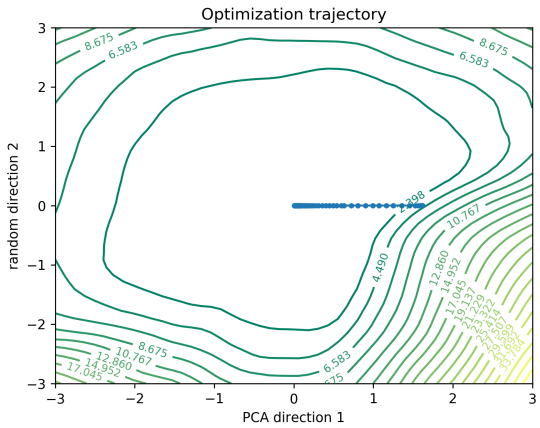
We can see the optimized trajectory cannot be found using randomized directions as the plots show a very small magnitude and a seemingly random trajectory. To solve these issues, we instead employ a non-randomized visualization, the technique is briefly described below :

1. Define a direction matrix M which is a stack of difference directions :

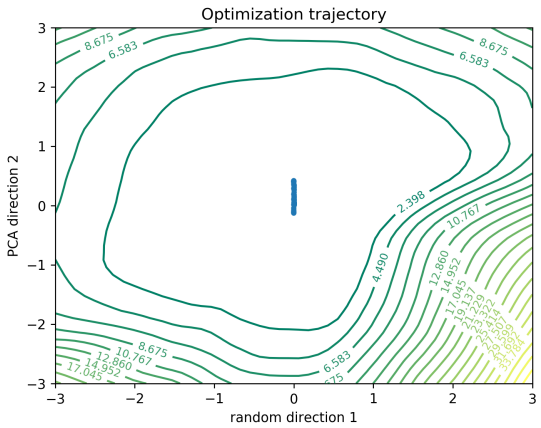
$$M = [W^{(0)} - W^{(n)};\ W^{(1)} - W^{(n)};\ \ldots;\ W^{(n-1)} - W^{(n)}] \tag{2}$$

2. Perform PCA on M and retain the first 2 principal components.
3. Project each $W^{(i)} - W^{(n)}$ onto the plane defined by the 2 principal components using the least squares method
4. Each projection is one point in the optimization trajectory.

The results for the optimization trajectory that we got for a 16 layer feed-forward network trained on the MNIST dataset, using 1 randomized direction and 1 PCA direction are shown below:



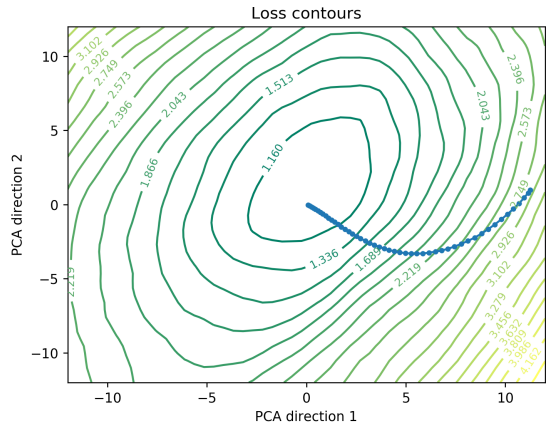
(a) Loss surface contour plot, random direction on the ordinate and PCA direction on the abscissa



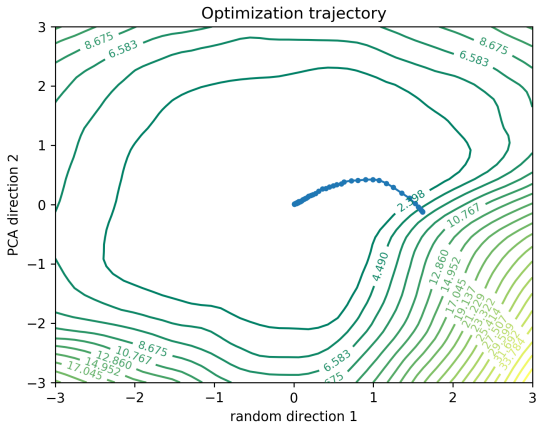
(b) Loss surface contour plot, PCA direction on the ordinate and random direction on the abscissa

Figure 9: Optimization trajectory visualizations using a 1 random direction and 1PCA direction for the 16 layer feed-forward network trained on MNIST using SGD.

We see that the using 1 PCA direction produces a projection with much larger magnitude as compared to the random direction. Using both PCA directions results in trajectories that seem reasonable. There is however some inherent error as a least squares projection is performed.



(a) Loss surface contour plot, 16 layer network trained on CIFAR



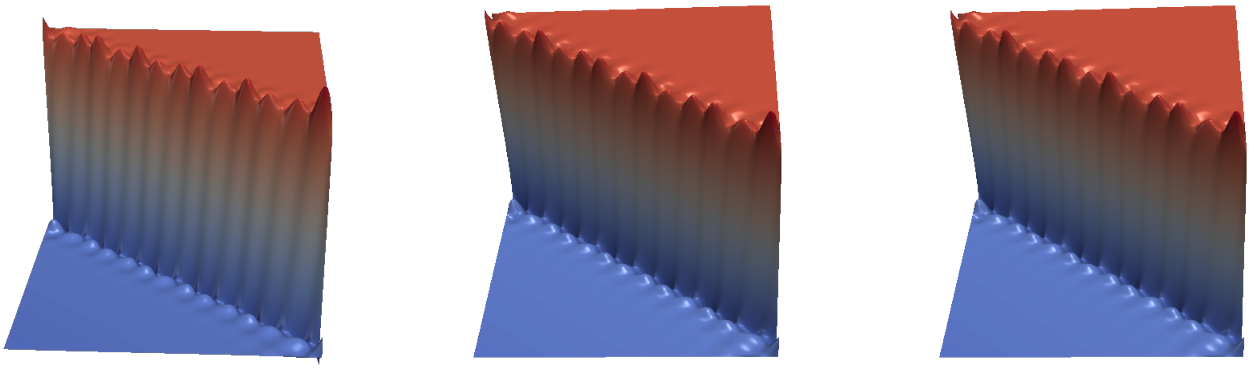
(b) Loss surface contour plot, 16 layer network trained on MNIST

Figure 10: Optimization trajectory visualizations using 2 PCA directions for the 16 layer feed-forward network trained on MNIST using SGD.

The results obtained were reasonable and provided some insight into the behaviour of the optimizer.

8 Effect of the batch size on the loss landscape

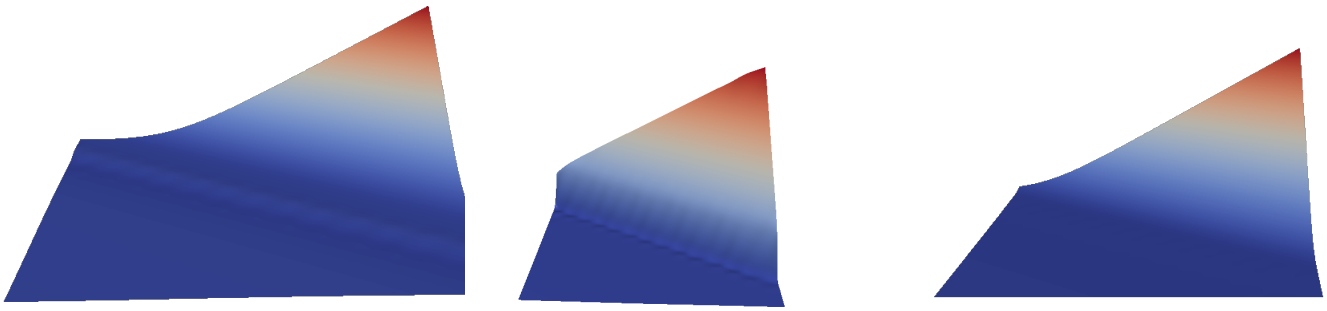
We then wanted to study the influence of batch size on the loss landscape. For that, we reused the architectures we prepared for the seismic-bumps dataset. We trained three model, with three different batch sizes from small to large (the largest size, 512, is about 20% of the dataset).
We first applied it to the one-layer architecture with the following results:



(a) Batch size = 5 (b) Batch size = 64 (c) Batch size = 512

Figure 11: Loss landscape 3-D renderings for varying batch sizes,1 hidden layer networks trained on seismic bumps dataset

We didn't observe any significant change when using different batch sizes, so we repeated the experiments, this time on the three-layer architecture, with the following results :



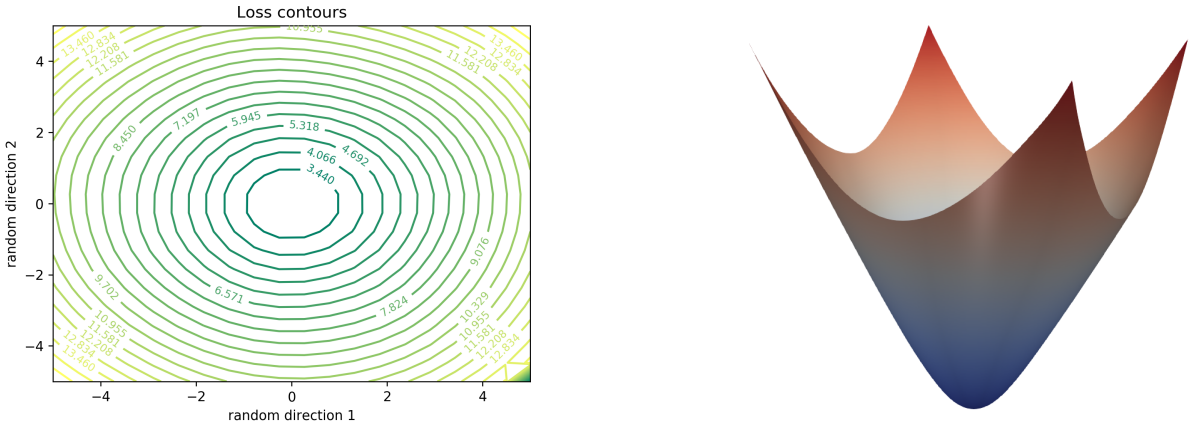
(a) Batch size = 5 (b) Batch size = 64 (c) Batch size = 512

Figure 12: Loss landscape 3-D renderings for varying batch sizes, 3 hidden layer networks trained on seismic bumps dataset

This time we can observe a change, the medium batch size makes a landscape with a step between the flat part and the slope. However, small batches and large batches give similar results, which doesn't allow us to draw general conclusions about increasing or reducing the batch size. Further testing is required to draw any conclusions.

9 RNN loss landscape

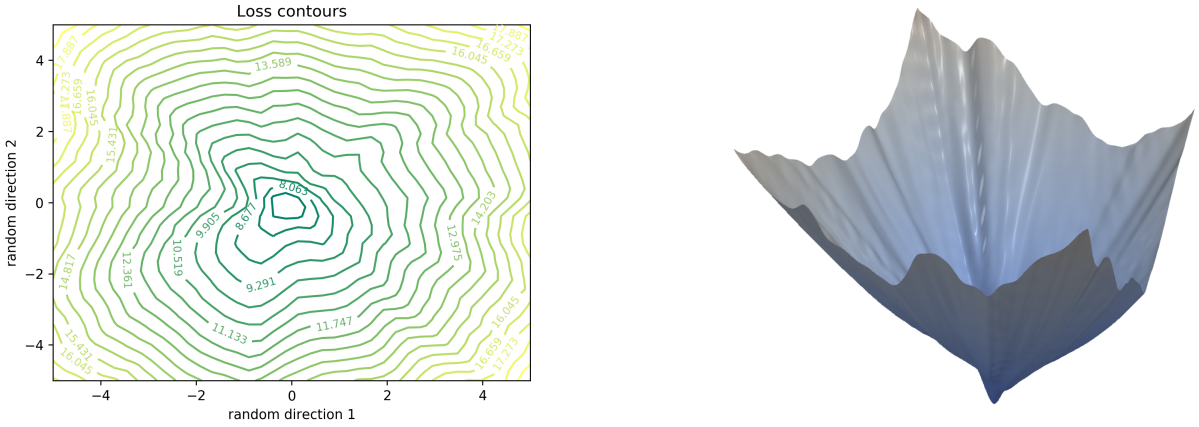
We also tried a few experiments to visualize the loss landscape of RNNs. For the first couple of models we trained, we obtained a very symmetric and smooth landscape.



(a) Loss surface contour plot (b) Loss surface 3D rendering

Figure 13: Loss landscape visualizations for a single layer RNN trained on the Shakespeare dataset

However, another model we trained displayed a more complex landscape as shown below.



(a) Loss surface contour plot

(b) Loss surface 3D rendering

Figure 14: Loss landscape visualizations for a single layer RNN trained on the Shakespeare dataset

Why a particular model resulted in a significantly more complex landscape is still not clear to us.

10 Conclusion

1. Increasing depth of a network leads to more complex loss landscapes that are more difficult to optimize.
2. Adding skip connections appears to considerably simplify the loss landscape thereby making optimization more likely to succeed.
3. Optimization trajectory visualizations require carefully chosen directions that are not orthogonal to the low dimensional subspace in which the optimization trajectory tends to lie [2].

A Model details

S.No.	Architecture	Dataset
1	CNN - 1 CONV layer	MNIST, CIFAR-10
2	CNN - 3 CONV layer	CIFAR-10
3	Feed forward - 16 layers	MNIST
4	Feed forward - 16 layers, 1 skip connection from layer 2 to layer 14	MNIST
5	RNN - 1 layer sequence length = 30,45 hidden dimension = 120,140	Shakespeare
6	Feed forward - 2 layers	Seismic bumps
7	Feed forward - 4 layers	Seismic bumps

Table 1: Details of neural network models, architectures and datasets used

References

[1] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[2] Tom Goldstein and Christoph Studer. Phasemax: Convex phase retrieval via basis pursuit. *CoRR*, abs/1610.07531, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6391–6401, 2018.