Visualization of loss landscape

Pierre CHAN 2225665 , Marc ZHANG 2312403 , 2312414 David Zhu , git : https://github.com/PierreCKL/Loss-lansape-project

Abstract

This project is based on the paper : Visualizing the Loss Landscape of Neural Nets, and aim to give a geometrical intuition of the loss landscape of neural network. Using different visualization we will see how the network's architecture changes the loss landscape and how training parameters affect the the shape of minimizers.

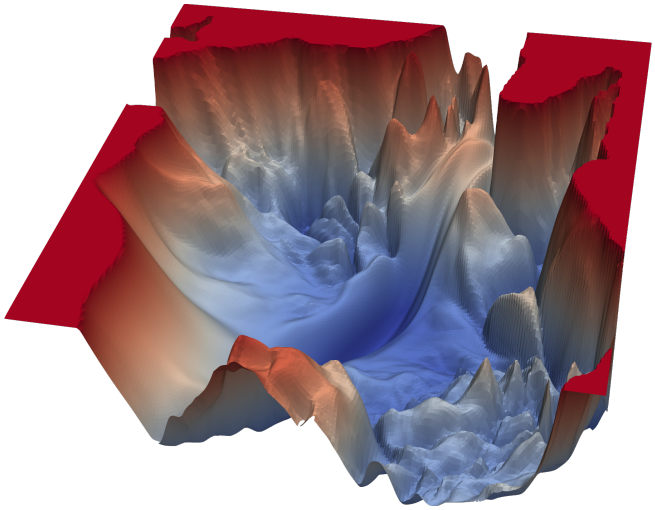


Figure 1: Exemple of Loss landscape visualization

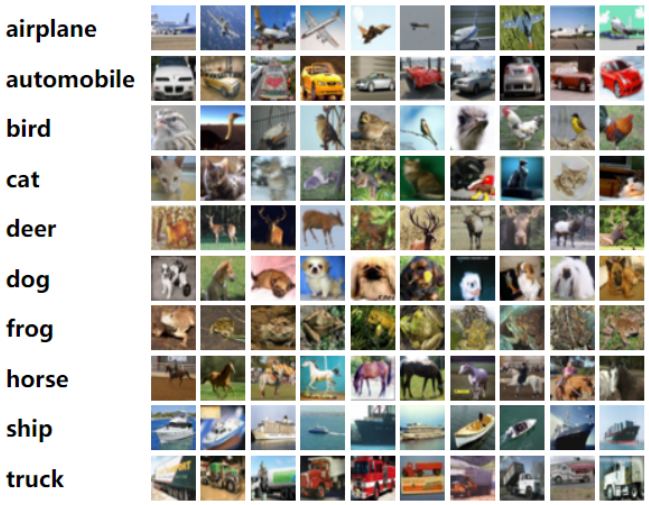
1 Introduction

A loss landscape is a representation of how the loss function changes with respect to the model's parameters. Understanding the loss landscape can provide insights into the behavior of neural networks during training. However, neural networks have many parameters, leading to a high-dimensional loss function space. Therefore, visualizing this space effectively is more challenging. The idea is to project the model parameter space into a 2D space and plot the loss function on a 3D space where z is the loss and the x and y dimensions represent the parameter space.

They first train a neural network to obtain a vector of parameters $\vec{\theta}$, then they propose two different ways to generate 2 vectors $\vec{V_1}$ and $\vec{V_2}$ of the same dimension as $\vec{\theta}$. Let α and β be 2 scalars that will roughly vary between -1 and 1 (we can adjust the range to move around the loss landscape). Let $L(\vec{\theta})$ be the loss function of the model with parameters $\vec{\theta}$ on the training set. The plot of loss landscape of the model is defined as follows: $f(\alpha, \beta) = L(\vec{\theta} + \alpha\vec{V_1} + \beta\vec{V_2})$

1.1 Dataset

All the visualization come from networks that have been trained on the CIFAR-10 dataset. CIFAR-10 consists of a collection of 60,000 32x32 color images in 10 classes, with 6,000 images per class. These images are RGB images, the classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.



2 First method : random plane visualization

To generate 2 vectors $\vec{V_1}$ and $\vec{V_2}$ of the same dimension as $\vec{\theta^*}$: the final weights after training the networks, the paper propose to choose 2 random directions generated by Gaussian's

law. They carefully choose the magnitude of the Gaussian perturbation so that "each filter in $\vec{V_1}$ to have the same norm of the corresponding filter in $\vec{\theta^*}$". To give an example : let the input x be a 1x32x32 image, the network has 2 layers, 1st layer is a 'same size' convolution layer with 2 channels output, kernel size: 3x3, stride: 1, padding: 1, we concatenate and flatten the features map to get a n: 2x32x32 vector, the 2nd layer is a MLP with a matrix: W of size 10xn and biais: b.
Let's view $\vec{\theta^*}$ as a tensor of shape (Number of parameters,1) like this :

$$\vec{\theta^*} = \Big(kernel_1.flat(),\ kernel_2.flat(),\ W.flat(),\ b\Big)$$

We first generate the Gaussian's perturbation $\vec{V_1}$ with the same shape and normalize like this :

$$\vec{V_1} = \Big(\frac{\vec{d_1}}{||kernel_1||}, \frac{\vec{d_2}}{||kernel_2||}, \frac{\vec{d_3}}{||W||}, \frac{\vec{d_4}}{||b||}\Big)$$

where $||.||$ is the Frobenius norm.

2.1 Experience

We trained different depth (20, 56 and 110) of resnet to see the effect of depth on the loss lanscape and we also tried to remove the skip connection to visualize why the skip connection enabled us to train much deeper network.

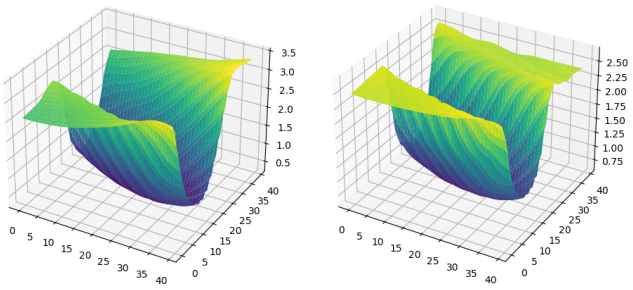


Figure 2: (left) Resnet-20 no skip connections, (right) Resnet-20

At the depth of 20, we can see that both network have similar loss landscape, and the loss appear convex. This does not mean the problem is convex, remember there is many more dimensions than the two we looked at, the problem have the same dimension as the number of parameters of our models.

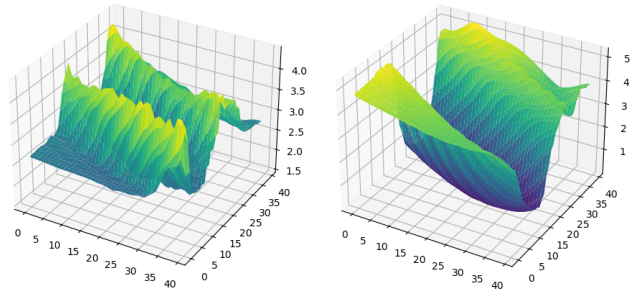


Figure 3: (left) Resnet-56 no skip connections, (right) Resnet-56

For a depth of 56 layer (figure 3) we start seeing non convexity for resnet without skip connection, the high loss value that are shaped like mountains block the path for the optimizer to find the local minimun since the gradient wants to minimize the loss function, the 'montains' forces the loss to increase and thus block the path to the minimun.

The skip connection resnet still show a locally convex shape around the minimizer, geometrically the skip connection remove the 'mountain' area so that the minimizer can easily find a local optimun without blocked path by the mountain area.

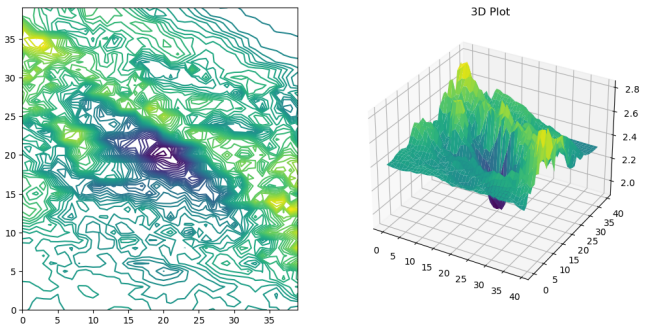


Figure 4: (left) Resnet-110 no skip connections, (right) Resnet-110 no skip connection

Finally we painfully trained a resnet with 110 layer and no skip connection to see the effect of high depth on the loss landscape, the local minima obtained (darker blue color means lower loss, green area are high loss) is centered on the graph. We now see that the local minima is surrounded by chaotic mountains area which make the optimization much more difficult.

We run out of GPU credit to train a resnet with 110 layers with skip connection so we will borrow the visualization from the reference paper to compare the 2 architectures.

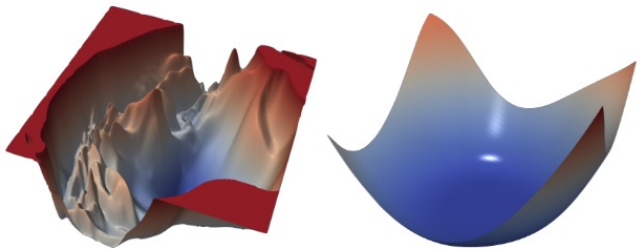


Figure 5: (left) Resnet-110 no skip connections, (right) Resnet-110

Even with really deep network the skip connection makes the loss landscape appear convex on the random selected directions.

3 Second method : Pca plane visualization

This method will allow us to visualize the loss landscape around the local minima and plot the optimization path. Let's

illustrate the method with n=3 epoches :

- Train network generate trajectory :
  $T = (\vec{\Theta}_1, \quad \vec{\Theta}_2, \quad \vec{\Theta}_3)$, $\vec{\Theta}_i$: params after epoch i
- Center on last epoch params :
  $T = (\vec{\Theta}_1 - \vec{\Theta}_3, \quad \vec{\Theta}_2 - \vec{\Theta}_3, \quad \vec{0})$ T matrix shape : (Nbre params, Nbre epochs)
- Apply pca to the trajectory Matrix : T, extract the top 2 pca component : Pca 1 $\in R^{Nparams}$, Pca 2 $\in R^{Nparams}$
- Compute $Loss(\alpha, \beta) = Loss(\vec{\Theta}_3 + \alpha \vec{Pca}1 + \beta \vec{Pca}2)$ alpha, beta scalar in the range $\in [-1, 1]$
- Project the trajectories on the pca plane

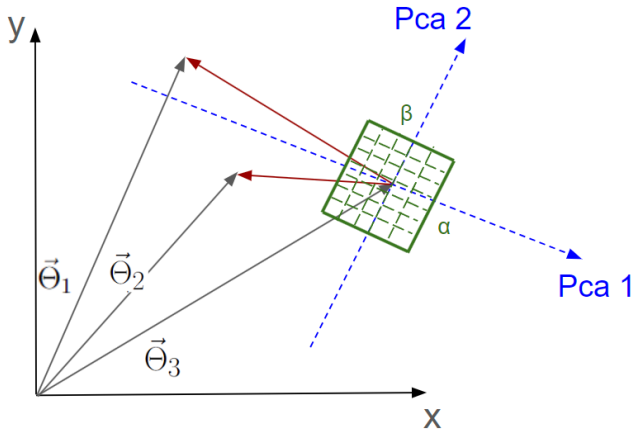


Figure 6: Pca method visualized

3.1 Results

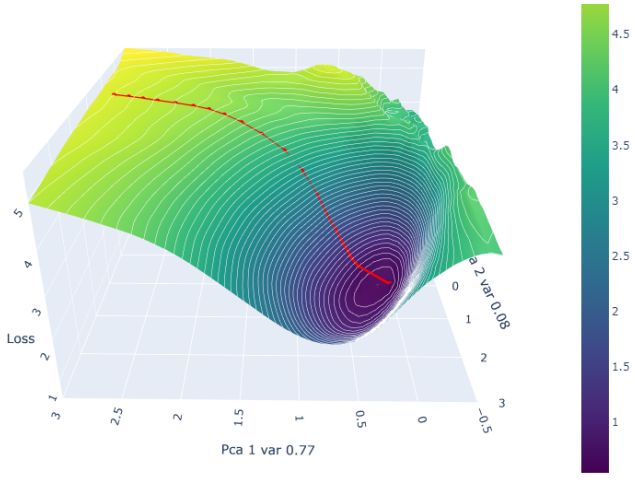


Figure 7: Pca plane loss landscape Resnet 20

We can see that the optimizer have no trouble to get to the local minima, the trajectory is always perpendicular to the iso value loss contour (along the white line the value of the loss is constant), the trajectory is very low dimensional, the first pca component represent 77% of the variation in the descent path and 8% for the second component.

When the depth increases we start to see the same mountains, high loss area that we saw in the random plane cut of the loss landscape.

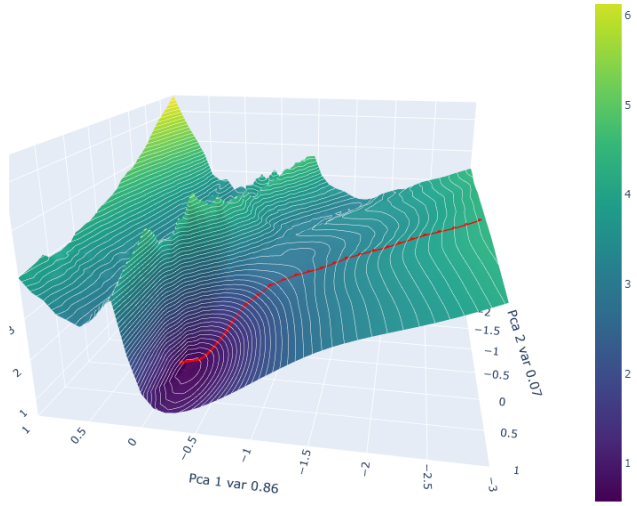


Figure 8: Pca plane loss landscape Resnet 20

3.2 Limitations

What happens if we want to see more than locally how the loss landscape behave? to do so we can try to increase the range of α and β in the formula : $f(\alpha, \beta) = L(\vec{\theta} + \alpha \vec{V_1} + \beta \vec{V_2})$

But if we move to far in the same random direction, the logits of the end layer will keep increasing in the same direction, meaning that the loss will shoot up. For instance if the direction $\vec{V_1}$ increase the probability of classifying the object as a cat, the further we go in that direction (bigger value of α) the more we will confidently (with probability closer to 1 as α increases) classify non cat images as cat and thus the negative log likelihood will diverge to infinity.

3.3 Random Plane or Pca plane ?

If we want to see the landscape of the optimization trajectory then the pca plane allow us to visualize this however among the many possible directions (we are in R^n dimension space n : number of parameters) if we choose meaningful direction we may fail to capture the non convexity of the problem. An extreme example of direction that is too meaningful is the gradient, if we look at $f(\alpha) = Loss(\vec{\theta} - \alpha \vec{grad}(Loss))$ we will see a classical decreasing loss plot and the problem will appear convex in this direction even though we know it is not.

Thus the random direction are more likely to capture the complexity of the problem.

4 Sharp vs flat optimizer

We train a resnet 20 layer model on the Cifar-10 dataset with a small batch size 64 and a large batch size : 4096 (8.2% of

the training set) both from the same initialisation weights. Let θ^s be the solution after training on the small batch size, and θ^l the solution training on the large batch size, using linear interpolation we plot the accuracy and loss curve along the direction θ^l - θ^s, formaly we plot

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

for alpha scalar.

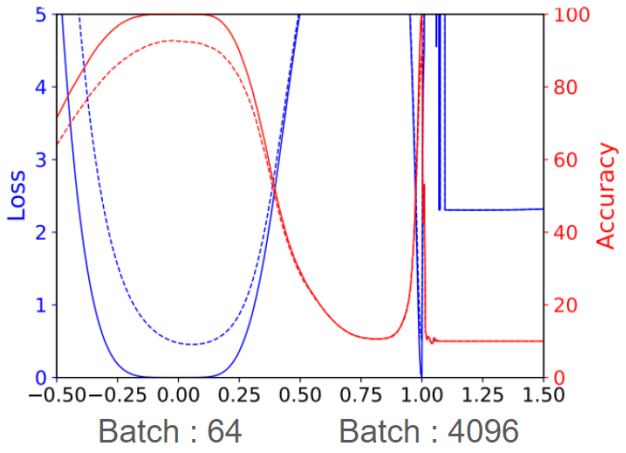


Figure 9: 1d linear interpolation of the solution from the small and large batch training, blue line are the loss, red line are the accuracies, dotted line are the test set and plain line are from the training set.

We see that the small batch loss curvature is much more flat, moving around the optimizer lead to small variation of the loss. This small batch network was much easier to train, with constant learning rate as opposed to the large batch network where we had to divide the learning rate by a factor 10 every time the loss stopped decreasing.

The intuition we get from the plot is that picking bad training hyper parameters change the loss landscape from a large basin of attraction with good hyper parameters we moved to a very sharp basin of attraction where training is much more difficult.

5 Conclusion

We reproduced a visualization method that offers understanding into the outcomes resulting from different decisions encountered by neural network practitioners, such as selecting network architecture, optimizer, and batch size.

6 Reference

- Visualizing the Loss Landscape of Neural Nets Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein