Loss Landscape Visualization

Report for the Seminar on Optimization and Neural Architecture Search

Stefan Wezel

30. Januar 2021

Abstract

Neural networks have emerged as powerful function approximators with large parameter sets. These parameters are optimized according to a loss function. Many assumptions have been made about the shape of the resulting loss landscape. However, only recently qualitative and empricial studies have been conducted. Here, we give an overview for recent advances of this field. We will explore different methods in detail and discuss their results and impact.

Introduction

When creating neural networks for a given task, practicioners or have to take many decisions. From architecture design, over optimizers, and schedules to hyperparameter choice there are many options that will play a key role in the eventual performance of the model. Anecdotal knowledge, experience and luck often lead to the final configuration and there is little empirical knowledge of what is actually effective. Loss landscape visualization can play a large role in guiding the community towards a more empirically foundation and maybe help build the groundwork for theoretical approaches.
Several recent works have shown that a better knowledge about the loss landscape can help build methods grounded in theory [Mutschler and Zell, 2020, Chaudhari et al., 2019]. Moreover, loss landscape visualization also helped to explain the effectiveness of existing methods such as stochastic gradient descent (SGD) [Robbins and Monro, 1951, Xing et al., 2018] and architectures with residual connections [He et al., 2016, Li et al., 2017].
As the space of parameters is too large to visualize in any meaningful way, methods of loss landscape visualization have to compromise. They might just consider small, visualizable subspace of a model's parameters. Results, thus, should be considered with caution and any conclusions should be drawn only carefully.
The following section will give .. to deep neural network architectures, and explain what a loss landscape is. Then, we will explore different methods of visualizing loss landscapes and see how insights from those methods can be applied. Finally, we discuss the impact of knowing the loss landscape better, see some limitations, and give an outlook on the future of the field.

Background

In this section we will provide some theoretical background and set the denotations for the following sections.

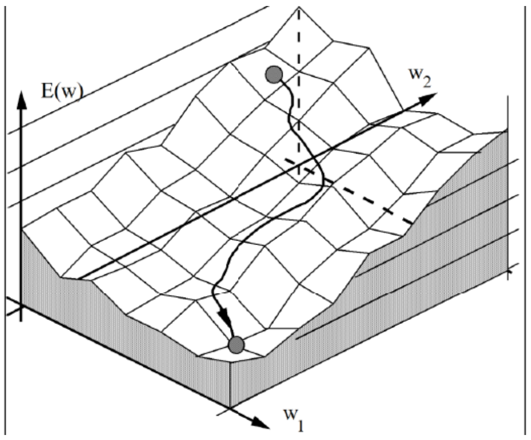

Abbildung 1: Loss landscape for a neural network with $\theta = \{w_1, w_2\}$. Values for w_1, w_2 are plotted against a loss $\mathcal{L}(\theta)$. As $|\theta| = 2$, visualization is straight forward.

A neural network g is a function with a set of adjustable parameters θ_i. We will denote it by g_{θ_i}. Deep neural networks (DNN) can be viewed as the composition $f_\theta = \sigma(g_{\theta_0}) \circ ... \circ \sigma(g_{\theta_m})$ of such functions where σ denotes some non-linear function. The parameter set θ may be arbitrary. For a given task it can be optimized by finding a minimum of a loss function $\mathcal{L}$. We refer to the resulting set as θ^*. Du to the non-linear functions involved and the often large magnitude of θ a solution usually cannot be found analytically.
In recent years, however, iterative, algorithmic approaches have proven to reliably find sufficient minima. These approaches built on the notion of a gradient descient which dates back to [Cauchy et al., 1847]. A gradient according to the current loss L and parameters θ_t is computed and a step, scaled by constant η is taken in its negative direction. The result of a step taken is an adjusted set of parameters θ_{t+1}. Note that for the gradient to be tractable, all operations involved have to be differentiable. Popular variants of the basic gradient descent algorithms include Stochastic Gradient Descent (SGD) [Robbins and Monro, 1951], ADAM [Kingma and Ba, 2014], and RMSProp [Graves, 2013].
We refer to the surface, such an algorithm is moving on, as loss landscape. The dimension n of the surface is determined by the magnitude of θ. For $n = 2$ this surface is easy to visualize as shown in Figure .

It is often assumed that the path such an algorithm has to take in order to find a minimum encounters several obstacles and is highly non-convex. Recent insights from loss landscape visualization have, however, shown that this is highly dependent on the architecture. [Li et al., 2017] have shown that loss surfaces of architectures with residual connections can have roughly convex subspaces.
Another common assumption is that flat minima generalize better. The intuition behind this is that there might be a shift between train and test distribution. Thus, for a flatter minimum, the probability of also covering the test distribution would be higher. However this claim was lacking empirical or theoretical studies. Recently, loss landscape visualization has helped providing compelling evidence to support such claims.

Inspecting the Loss Landscape

With the context set and some background, we can now have a look at approaches to visualize the high-dimensional loss landscape. The section will explore the rather simple technique of linear interpolation as proposed by [Goodfellow et al., 2014]. A refined approach is taken by [Li et al., 2017] which will be described in section .

Linear Interpolation

[Goodfellow et al., 2014] propose to use a linear interpolation between two parameter sets θ_0 and θ_1 as visualizeable subspace. They plot the function shown in Equation where $\mathcal{L}$ is the loss function and α serves as weighting parameter. The scalar α can for example be in a range of 0 to 1, thus ranging from full weight on model θ_0 to interpolating towards full weigh on model θ_1.

$$f(\alpha) = \mathcal{L}((1-\alpha)\theta_0 + \alpha\theta_1) \tag{1}$$

The aussagekraft of such visualizations depends of course heavily on the choice of θ_0 and θ_1. Figure shows shows the interpolation between an untrained model $\theta_{untrained}$ and a model $\theta_{trained}$ trained for an image classification task.

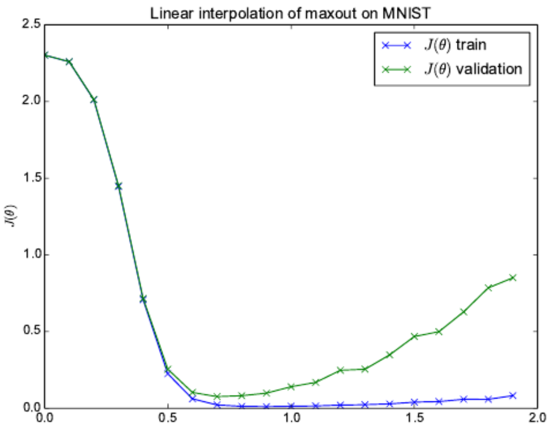

Abbildung 2: Linear interpolation between an untrained and a trained model.

The subspace visualized in this figure smoothly transitions between the two models and appears roughly convex.
[Li et al., 2017] use this technique to explore the effect of batch size in training on the shape of minima. A result is shown in Figure . When $\alpha = 0$, meaning full weight is on the model trained with a small batch size the minimum is flat and wide. As α increases, the loss gets higher before it sharply drops into another minima, where full weight is on the model trained with a large batch size. This minimum is much sharper, indicating a relationship between batch size and flatness of an optimum. If the assumption that flat minima generalize better holds, this would render smaller batch sizes more favorable.

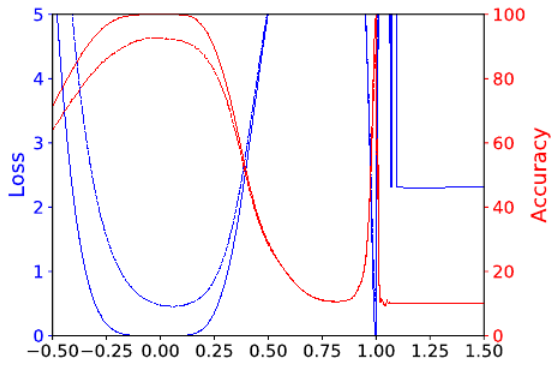

Abbildung 3: Linear interpolation between a model trained with a small batch size (left) and a model trained with a large batch size (right).

As discussed by [Li et al., 2017], linear interpolation suffers from several shortcomings and limitations. For example, it is arguable whether it is actually possible to capture non-convexities with this method. They thus propose to use filter normalization when visualizing loss landscapes. We will discuss their proposed methods and results in Section
.

Filter Normalization

Addressing the issues of [Goodfellow et al., 2014]'s visualizations using plain linear interpolation, [Li et al., 2017] propose to scale normalized parameters θ. For their method, rather than interpolating between two models, inspect the neighborhood around a model $\hat{\theta}$ by adding a direction vector u of the same shape to $\hat{\theta}$. This direction vector is then scaled by different values for α and the corresponding loss is evaluated, yielding the equation

$$f(\alpha) = \mathcal{L}(\hat{\theta} + \alpha u).$$

To ensure that the updates along u live on the same scale as the the values of $\hat{\theta}$, [Li et al., 2017] propose to normalize u by setting

$$u = \frac{||\hat{\theta}||}{||u||}, \tag{2}$$

thus normalizing the parameters of the neighborhood models. They propose the term filter normalization as they conduct most of their experiments on convolutional neural networks, where the most of $\hat{\theta}$ is use to parameterize filters.

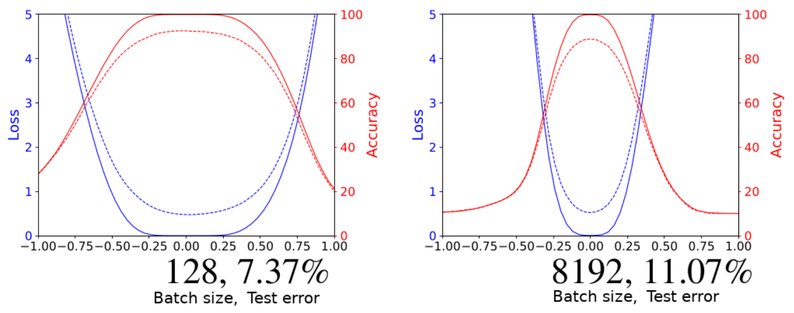

Abbildung 4: Loss landscape neighborhood around a model trained with a small batch size(left) and a model trained with a larger batch size.

Related to Figure , they visualize the filter normalized landscape around a model trained with a small batch size and a model trained with a large batch size. The result is shown in Figure . While a difference in flatness remains, it is much less clear with both models laying in rather flat minima. The minimum of the small batch size model is, however, still noticeably flatter. The resulting test error is also lower. This further hints towards a better generalization of flat minima and the batch size's role in finding such.
[Li et al., 2017] further expand this approach beyond 1-dimensional plots. By choosing not one but multiple direction vectors u_i, it can be used for higher dimensions. However, any dimension beyond two would result in the same dilemma for which such methods were developed. Thus, applying it to two dimensions is useful. The function to plot is then

$$f(\alpha, \beta) = \mathcal{L}(\hat{\theta} + \alpha u_1 + \beta u_2).$$

This results in more expressive plots, revealing more information about the $\hat{\theta}$'s neighborhood. An insightful example is given in Figure where two architectural paradigms are compared.

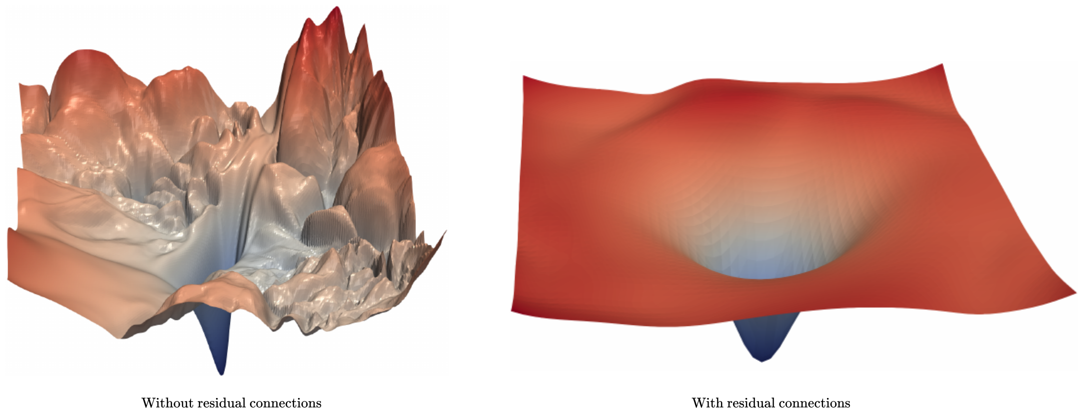

Abbildung 5: Loss landscape neighborhood around a model trained with a small batch size(left) and a model trained with a larger batch size.

The loss landscape around a minimum found with a model without any residual connections has several local minima. Intuitively put, an algorithm such as SGD might have a hard time traversing such rugged terrain. On the other hand, introducing residual connection results in a much smoother loss landscape around the minimum. It even appears roughly convex, in the visualized subspace.

While the methods proposed by [Li et al., 2017] produce more nuanced plots than simple linear interpolation, it is still important to mention that only a small subspace of the high dimensional parameter space is depicted. Thus, any conclusion has to be drawn with caution. It is also worth mentioning that their proposed method comes with high computational costs, as they consider the whole dataset to evaluate the empirical loss $\mathcal{L}$ at any considered point around $\hat{\theta}$. This severely limits the applicability of such methods. For reasonable experiments, only as small patch of neighborhood can be considered. Also the variety of currently existing architectures expands beyond just using residual connections or not. Moreover, this makes it also only usable with constraints for exploring the effect of different values for hyperparameters on the loss landscape.

Applying Findings from Loss Landscape Visualization

Parabolic Approximation Line Search

Entropy-SGD

Conclusions

Literatur

[Cauchy et al., 1847] Cauchy, A. et al. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

[Chaudhari et al., 2019] Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. (2019). Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018.

[Goodfellow et al., 2014] Goodfellow, I. J., Vinyals, O., and Saxe, A. M. (2014). Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*.

[Graves, 2013] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Li et al., 2017] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2017). Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

[Mutschler and Zell, 2020] Mutschler, M. and Zell, A. (2020). Parabolic approximation line search for dnns. *arXiv preprint arXiv:1903.11991*.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Xing et al., 2018] Xing, C., Arpit, D., Tsirigotis, C., and Bengio, Y. (2018). A walk with sgd. *arXiv preprint arXiv:1802.08770*.