Loss Landscape Visualization

Report for the Seminar on Optimization and Neural Architecture Search

Stefan Wezel

4. Februar 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 and claims have been made about the shape of the resulting loss landscape. However, only recently qualitative and empirical studies have been conducted. Here, we give an overview of recent advances in this field. We will explore different methods in detail and discuss their results and impact.

Introduction

When creating neural networks for a given task, practitioners have to make many decisions. From architecture design, over optimizers, and schedules to hyperparameter choice, many options will play a key role in the eventual performance of the model. Anecdotal knowledge, experience and often luck lead will 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 empirical foundation and maybe help build the groundwork for theoretical approaches.

Several recent works have shown that better knowledge about the loss landscape can help build methods grounded in theory [10, 4]. Moreover, loss landscape visualization also helped to explain the effectiveness of existing methods such as stochastic gradient descent (SGD) [12, 13] and architectures with residual connections [7, 9].

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 a small, visualizable subspace of a model's parameters. Results, thus, should be considered with caution and any conclusions should be drawn carefully.

The following section will give an introduction to deep neural network architectures, and an intuition of what a loss landscape is. Later, 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.

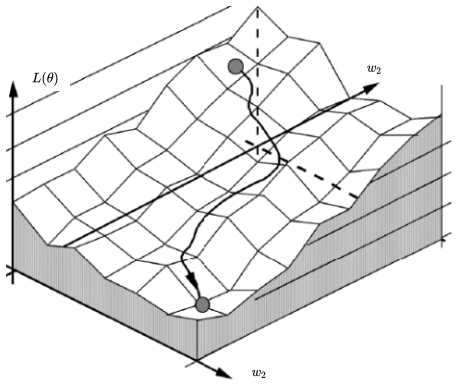

Abbildung 1: Loss landscape for a neural network with $\theta = \{w_1, w_2\}$. Values for w_1, w_2 are plotted against a loss $\mathcal{L}(\theta)$. As $|\theta| = 2$, visualization is straight forward.

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 build on the notion of a gradient descent which dates back to Cauchy et al. [3]. 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) [12], ADAM [8], and RMSProp [6].
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 1.

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. [9] 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 long lacking empirical or theoretical support. Recently, loss landscape visualization has helped to provide compelling evidence for 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 following section will explore the rather simple technique of linear interpolation as proposed by Goodfellow et al. [5]. A refined approach is taken by Li et al. [9] which will be described in the subsequent section.

Linear Interpolation

Goodfellow et al. [5] propose to use a linear interpolation between two parameter sets θ_0 and θ_1 as visualizable subspace. They plot the function shown in equation 1 where $\mathcal{L}$ is the loss function considering all training examples, and α serves as a 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 weight on model θ_1.

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

The meaningfulness of such visualizations depends of course heavily on the choice of θ_0 and θ_1. figure 2 shows shows the interpolation between an untrained model $\theta_{untrained}$ and a model $\theta_{trained}$ trained for an image classification task.

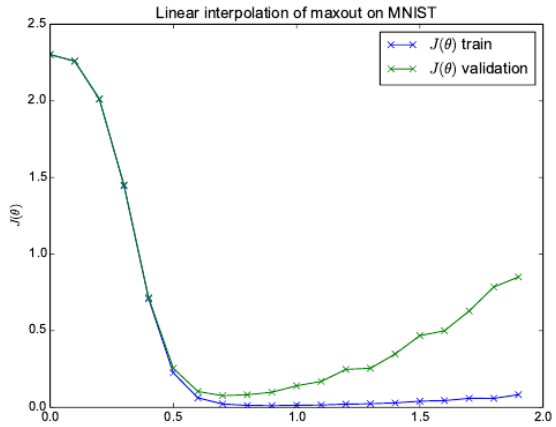

Abbildung 2: Linear interpolation between an untrained and a trained model.

The subspace visualized in this figure smoothly transitions between the two models and appears roughly convex.
Li et al. [9] use this technique to explore the effect of batch size in training on the shape of minima. A result is shown in figure 3. 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 minimum, where full weight is on the model trained with large batch size. This minimum is much sharper, indicating a relationship between batch size and the flatness of an optimum. If the assumption that flat minima generalize better holds, this would render smaller batch sizes more favorable.

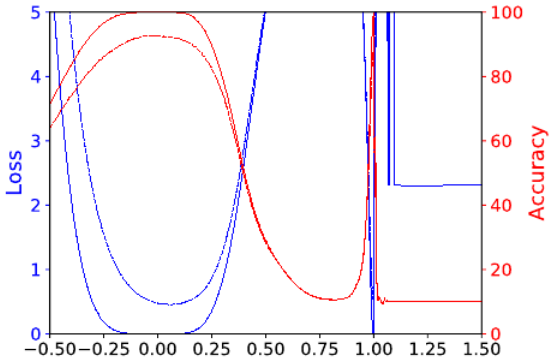

Abbildung 3: Linear interpolation between a model trained with a small batch size (left) and a model trained with a large batch size (right).

As discussed by Li et al. [9], linear interpolation suffers from several shortcomings and limitations. For example, it is arguable whether it is possible to capture non-convexities with this method. They thus propose to use filter normalization when visualizing loss landscapes.

Filter Normalization

Addressing the issues of Goodfellow et al. [5]'s visualizations using plain linear interpolation, Li et al. [9] propose to scale normalized parameters θ. For their method, rather than interpolating between two models, they 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. [9] 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 the majority of their experiments on convolutional neural networks, where the most of $\hat{\theta}$ is used to parameterize filters.

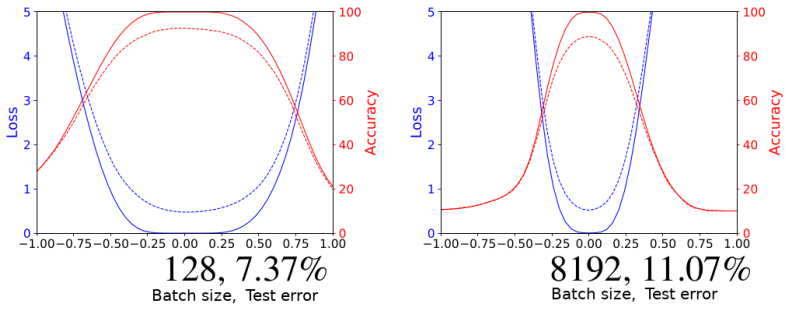

Abbildung 4: Loss landscape neighborhood around a model trained with a small batch size(left) and a model trained with a larger batch size.

Related to figure 3, 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 4. 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. [9] 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 5 where two architectural paradigms are compared.

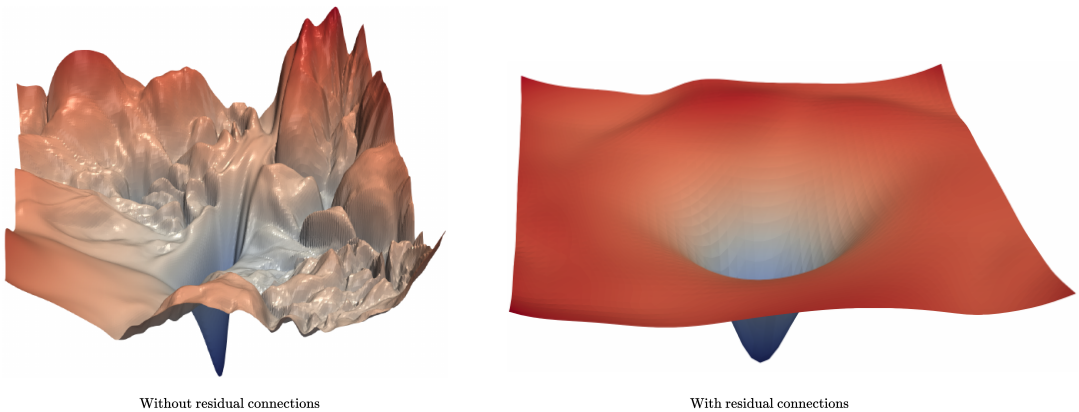

Abbildung 5: Loss landscape neighborhood around a model without residual connections (left) and a model with residual connections.

The loss landscape around a minimum found with an architecture 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 connections 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. [9] 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 a small patch of the models 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

Recent progress in loss landscape visualization, as the works described in the previous section, have brought new insights into the shape of the loss function. These insights have also helped build a better understanding of the impact on the losses shape on generalization. In the following sections, we will focus on works that apply these findings to build optimizers.

Parabolic Approximation Line Search

We have seen that, depending on the architecture, the loss surface can be roughly convex. Mutschler and Zell [10] provide further empirical evidence to support this assumption. To make use of this characteristic, they propose an optimizer that approximates the loss in negative gradient with a parabola. A step is then taken to the minimum of the parabola, which can be found analytically. They call their approach Parabolic Approximation Line Search (PAL).

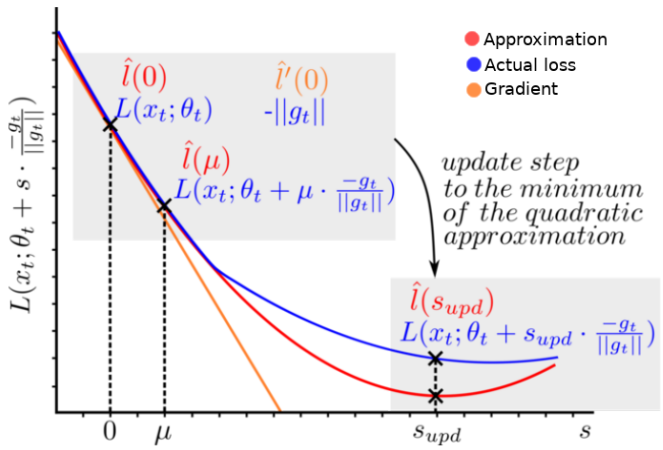

Abbildung 6: Intuition for PAL. Three measurements are taken and a parabola is approximated according to them. The update step is then taken to the minimum of the parabola.

To approximate a parabola, PAL takes three measurements. The loss $L(0)$ at the current position, the derivative in gradient direction $L'(0) = -||g_t||$, and finally the a loss at

another position $L(\mu)$, where the measuring distance μ is a positive scalar. These measurements are then used to parameterize parabola $\hat{L}(s) = as^2 + bs + c$. The curvature is determined by $a = \frac{L(\mu)-L(0)-L'(0)\mu}{\mu^2}$. The position of the parabola is given by $b = L'(0)$ and $c = L(0)$.
Given this approximated parabola, PAL performs an update step s_{update} to its minimum by

$$s_{update} = \frac{\hat{L}'(0)}{\hat{L}''(0)} = -\frac{b}{2a} = \frac{-\hat{L}'(0)}{2\frac{L(\mu)-L(0)-L'(0)\mu}{\mu^2}}. \tag{3}$$

For PAL to be able to perform such an update step, $L(0)$ and $L(\mu)$ have to be determined. For the very first step, this cannot be achieved by PAL itself. To alleviate this, Mutschler and Zell [10] propose to perform a cold start using another algorithm to find $L(\mu)$. It is also important to note that the measurements have to be taken on the same batch and all random values involved have to be reused.
Mutschler and Zell [10] compare the performance of PAL to other optimizers. They consider commonly used approaches like SGD or ADAM as well as to other line search methods such as Stochastic Line Search (SLS) proposed by Paquette and Scheinberg [11]. Their results show that PAL's performance is consistent across multiple architectures and tasks. This becomes especially evident when compared to SLS, which while outperforming PAL on some tasks, is much less stable. PAL generally converges to good minima without relying on hand-crafted step size schedules, as for example, SGD does.
To further justify their method, Mutschler and Zell [10] also provide extensive evidence towards the loss landscape's local convexity. Despite the success of the proposed PAL's it is still worth mentioning that on a number of tasks, SGD with a learning rate schedule outperforms other optimizers including line search approaches such as PAL.

Entropy-SGD

Chaudhari et al. [4] build on the notion that flat minima yield better generalization. They actively bias their optimizer to find wide valleys. Extending the work of Baldassi et al. [1], rather than considering a landscape determined only by loss, they consider the volume of similar parameter configurations around the current model and thus form what they refer to as entropy landscape.

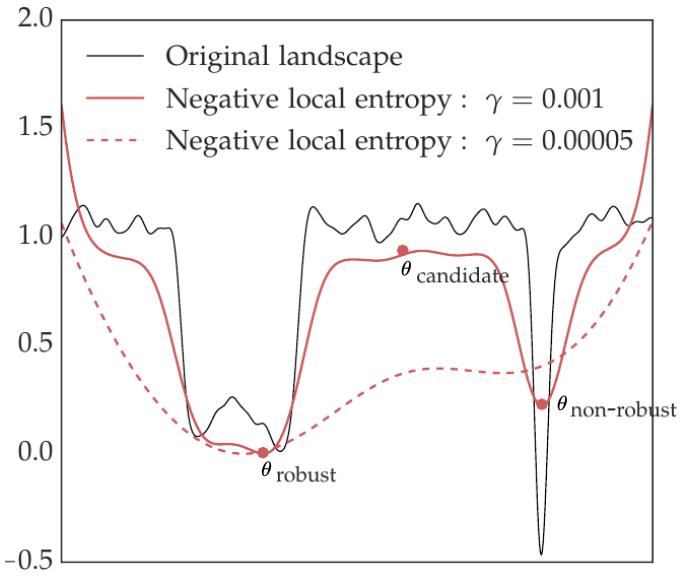

Abbildung 7: Intuition for Entropy-SGD. The entropy loss has a global minimum where generalization is assumed to be high.

An intuition for this entropy landscape is given in figure 7. In this toy example, the landscape formed by the used loss (black) has two major minima. One, however, is quite sharp while the other is much flatter. Two more curves (red) denote the local entropy landscape as proposed by Chaudhari et al. [4]. These curves global minima both lie in the original loss landscapes flatter minimum. Chaudhari et al. [4] assume that using the model lying in the global minimum of the local entropy results in better generalization.
The introduced local entropy is given by

$$F(\theta, \gamma) = log \int_{\theta} exp(-L(\theta') - \frac{\gamma}{2}||\theta - \theta'||_2^2 d\theta') \tag{4}$$

,where θ is a model considered, θ' is a model in θ's neighborhood, and γ can be viewed as a scope that determines, how far away, models should be considered to compute the volume. The mass of this local entropy concentrates in wide valleys and is thus maximized.
The proposed Entropy algorithm finds a minimum of the negative local entropy landscape by performing a nested loop of two SGDs. Intuitively put, the inner SGD estimates the gradient of the local entropy based on the volume of good models while the outer SGD updates parameters θ. Thus Entropy-SGD computes

$$\theta*_{Entropy-SGD} = arg\ min_{\theta} - F(\theta, \gamma; X)$$

which can be found by iteratively computing

$$-\nabla_{\theta} F(\theta, \gamma; X_{batch}) = \gamma(\theta - \mathbb{E}\left[\theta'; X_{batch}\right]). \tag{5}$$

Chaudhari et al. [4] describe the effect of γ as follows: for $\gamma \to 0$, the entropy landscape approaches the regular loss landscape as determined by the objective. For $\gamma \to \infty$, the entropy landscape approaches a uniform distribution.
To support their claims and methods, Chaudhari et al. [4] provide further theoretical evidence. They prove that the entropy landscape is smoother than the original loss landscape. they analyze the eigenspectrum of the Hessian [2], of models that generalize well, finding the majority of eigenvalues is close to zero. This indicates low curvature and thus flatness

of the minima, further backing the initial assumption that flatter minima generalize better.

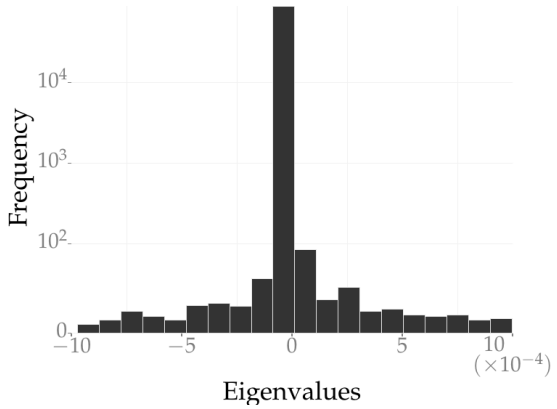

Abbildung 8: Histogram of the eigenspectrum of the Hessian matrix of a well generalizing model.

Empirically, Chaudhari et al. [4] compare Entropy-SGD to SGD and ADAM. In their experiments, they demonstrate that on the tasks considered, Entropy-SGD reaches lower test error or perplexity.
However, it is important to note that the conducted experiments only encompass a rather small set of tasks. In order to empirically prove that Entropy-SGD is able to outperform state-of-the-art optimizers further investigation is needed, and a wider set of tasks has to be considered. Nonetheless, this work brings yet more insights into the nature of the loss landscape and more specifically generalization.

Conclusions

Modern deep neural networks are powerful and are increasingly applied for challenging tasks. While this field of research is quickly evolving, the development is often ahead of itself. This is exemplified by the seemingly unreasonable performance of algorithms like SGD. Only by rigorously investigating methods, finding empirical and theoretical evidence, the field can progress sustainably.
The works presented here are displays of this process. While not necessarily providing groundbreaking insights into the loss landscape's shape or yielding a new do-it-all optimizer, they are important steps towards building a solid foundation for optimization in deep learning.

Literatur

[1] Baldassi, C., Borgs, C., Chayes, J. T., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. (2016). Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences*, 113(48):E7655–E7662.

[2] Bishop, C. (1992). Exact calculation of the hessian matrix for the multilayer perceptron.

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

[4] 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.

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

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

[7] 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.

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

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

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

[11] Paquette, C. and Scheinberg, K. (2018). A stochastic line search method with convergence rate analysis. *arXiv preprint arXiv:1807.07994*.

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

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