Gradient Descent Analysis: On Visualizing the Training of Deep Neural Networks

Martin Becker, Jens Lippel and Thomas Zielke
University of Applied Sciences Düsseldorf, Münsterstr. 156, 40476, Düsseldorf, Germany

Keywords: Deep Neural Networks, Learning Process Visualization, Machine Learning, Numerical Optimization, Gradient Descent Methods.

Abstract: We present an approach to visualizing gradient descent methods and discuss its application in the context of deep neural network (DNN) training. The result is a novel type of training error curve (a) that allows for an exploration of each individual gradient descent iteration at line search level; (b) that reflects how a DNN's training error varies along each of the descent directions considered; (c) that is consistent with the traditional training error versus training iteration view commonly used to monitor a DNN's training progress. We show how these three levels of detail can be easily realized as the three stages of Shneiderman's Visual Information Seeking Mantra. This suggests the design and development of a new interactive visualization tool for the exploration of DNN learning processes. We present an example that showcases a conceivable interactive workflow when working with such a tool. Moreover, we give a first impression of a possible DNN hyperparameter analysis.

1 INTRODUCTION

Deep neural networks (DNNs) are machine learning models that have demonstrated impressive results on various real-world data processing tasks; yet their widespread use is hampered due to the absence of a generally applicable learning procedure. Usually, the efficiency and robustness of deep learning processes depend on a set of hyperparameters. Adjusting these hyperparameters can be quite challenging, especially for users who have only very little to no experience with DNNs.

A standard strategy to assess learning processes resulting from different hyperparameter settings is to compare their training error curves. Although such a comparison can help to gain a certain intuition about individual hyperparameters and their interaction, it provides a fairly limited view on DNNs. E.g. Figure 1 shows two training error curves that we discuss in Section 3. By comparing the two curves, we merely learn how to differentiate between favorable training error curves and less favorable ones but not how they relate to the core mechanisms of deep learning.

In search of such a relation, we developed a novel enhanced type of training error curve. It consists of three levels of detail, which can be easily assigned to the three stages of the *Visual Information Seeking Mantra* (Shneiderman, 1996): In the *overview stage*, the traditional training error curve is in the focus of attention. In the *zoom and filter stage*, one can look at the variation of the training error along each of the descent directions considered during the gradient descent-based DNN training. The *details-on-demand stage* allows for an in-depth exploration of methods

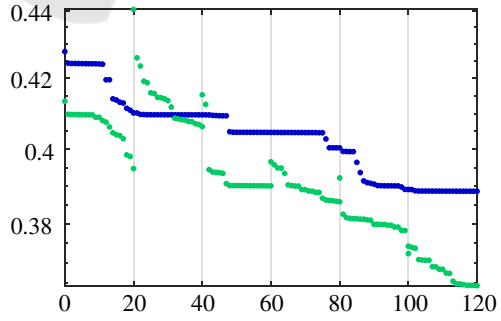


Figure 1: Two exemplary training error curves. They show a training error versus training iteration view. Displaying them at the same time allows for a direct comparison. The green curve has a smaller final error. At the same time, its curve progression is more irregular as compared to the blue curve, which appears to be decreasing monotonically. The underlying core deep learning mechanisms remain unclear because they are not directly accessible through a simple comparison of the two training error curves.

Becker, M., Lippel, J. and Zielke, T.
Gradient Descent Analysis: On Visualizing the Training of Deep Neural Networks.
DOI: 10.5220/0007583403380345
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 338-345
ISBN: 978-989-758-354-4

for adaptive control of the step width along a given descent direction.

The main intent of this paper is to introduce this *three-level / three-stage visualization approach* in a way that is comprehensible to both non-experts and experts in the field of deep learning. To this end, we begin by looking at an easy to understand non-DNN example. It reflects all central elements of an actual gradient descent-based DNN training but to a more manageable extent. This also allows us to showcase an interactive workflow that covers all three levels of detail. The design and development of an interactive tool for the visual exploration of the novel enhanced training error curve is one of the goals of our future work. We therefore also give a first impression of a conceivable DNN hyperparameter analysis. We look at the *mini-batch size* and show that varying it has a very distinctive effect on a DNN's enhanced training error curve. This effect is not necessarily evident in the *overview stage* but easy to spot in the *zoom and filter stage* of our visualization approach. From this, we conclude that our enhancement of the traditional training error curve enables a more direct access to DNN learning processes at hyperparameter level.

Over the last few years, visualization approaches that help to better understand deep learning models have been of great interest – a thorough overview of the most recent approaches is presented in (Hohman et al., 2018). However, it appears to us that gradient descent methods have mainly been visualized to find out what individual neurons can learn from data. By contrast, our approach focuses on details relating to the basic theoretical ideas behind a gradient descent method. As a consequence, it can be applied to both non-DNN and DNN scenarios. Table 1 lists some of the gradient descent methods that we aim to include in our interactive tool. In this paper, we consider the Polak-Ribière conjugate gradient (CG) method. The methods in the second table are of special interest to us because they were specifically designed to serve the needs of up-to-date deep learning models. Their study and inclusion in our visualization approach is subject to future work.

Table 1: A overview of gradient descent methods that we intend to include in our interactive visualization tool. The methods in the second table were specifically designed to serve the needs of up-to-date deep learning models.

In this paper	
Method	Reference
Polak-Ribière CG method*	(Polak and Ribière, 1969)

*) See (Polak, 2012) for an explanation in English.

Subject to future work	
Method	Reference
Rprop**	(Riedmiller, 1994)
AdaGrad	(Duchi et al., 2011)
AdaDelta	(Zeiler, 2012)
Adam and AdaMax	(Kingma and Ba, 2014)
Nadam	(Dozat, 2016)
AMSGrad	(Reddi et al., 2018)

**) See (Igel and Hüsken, 2003) for different Rprop variants.

2 VISUALIZATION APPROACH

As mentioned above, our visualization approach can be applied to both non-DNN and DNN scenarios. In general, it is applicable to virtually any problem that can be modeled as follows:

Each solution to the problem can be represented as a vector $\boldsymbol{\theta} \in \Theta$ where $\Theta := \mathbb{R}^p$ denotes the space of possible solutions. One can define a continuously differentiable *error function* $J : \Theta \to \mathbb{R}$ that enables the assessment of each $\boldsymbol{\theta} \in \Theta$; any optimal solution $\boldsymbol{\theta}^* \in \Theta$ satisfies

$$J(\boldsymbol{\theta}^*) \leq J(\boldsymbol{\theta}) \qquad (\forall\, \boldsymbol{\theta} \in \Theta). \tag{1}$$

We call any such $\boldsymbol{\theta}^*$ a *minimizer*. In most cases, the problem of finding minimizers cannot be solved analytically. Here, gradient descent methods can be applied in order to obtain good approximations.

Throughout this section, we consider an example from a non-DNN context. The visualizations shown, are based on results obtained through a CG descent that we applied on $J : \mathbb{R}^2 \to [0|\infty)$ defined by

$$J(\boldsymbol{\theta}) := 100(\theta_2 - \theta_1^2)^2 + (1-\theta_1)^2 \tag{2}$$

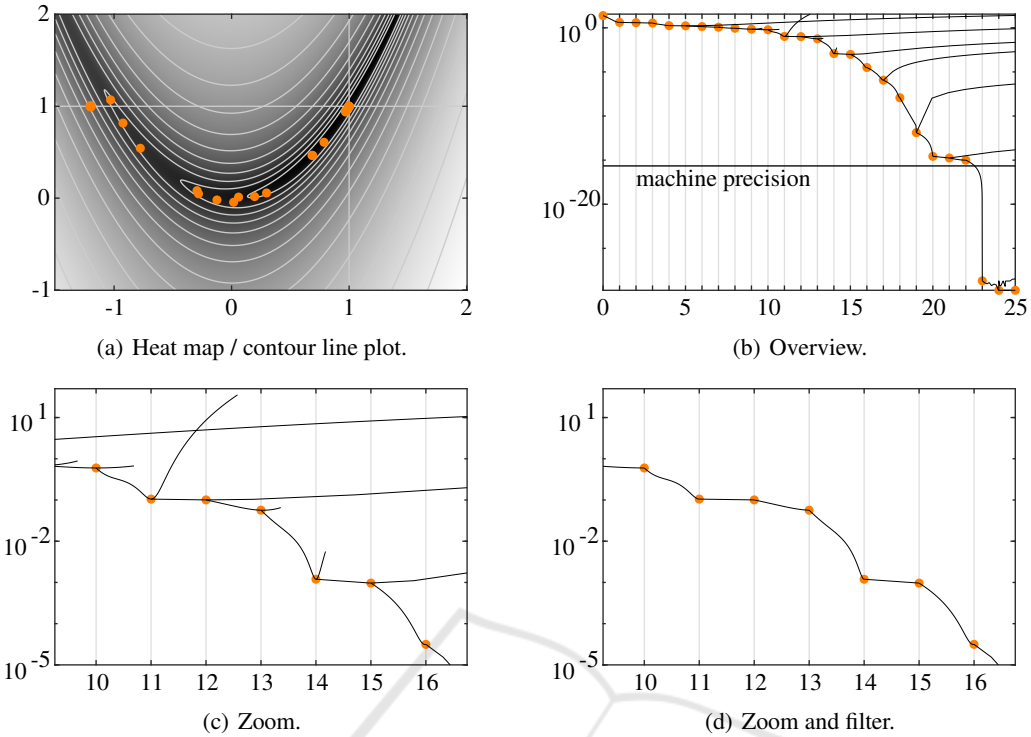
for $\boldsymbol{\theta} \in \Theta$; its only minimizer $(1,1)^{tr}$ is located in a strongly curved valley following the parabola given by $\theta_2 = \theta_1^2$, which makes it difficult to approximate (Rosenbrock, 1960). Because we are not looking at an actual DNN training, we simply write traditional and enhanced error curve.

2.1 Overview

For a given initial solution $\boldsymbol{\theta}^0 \in \Theta$, gradient descent methods try to approximate minimizer by visiting a finite sequence of solutions $(\boldsymbol{\theta}^k)_{k=0}^K$ such that

$$J(\boldsymbol{\theta}^{k+1}) < J(\boldsymbol{\theta}^k) \qquad (\forall\, k \in \{0,\ldots,K-1\}). \tag{3}$$

Because Rosenbrock's function is a function of only two variables, we can illustrate the visited solutions as marks on a map. Figure 2(a) shows a heat map /



(a) Heat map / contour line plot.

(b) Overview.

(c) Zoom.

(d) Zoom and filter.

Figure 2: Visualizations of the results of the CG descent that we applied to Rosenbrock's function (Rosenbrock, 1960). We considered $\boldsymbol{\theta}^0 := (-1.2, 1)^{tr}$ as the initial solution. The views presented in (b), (c) and (d) give an impression of the *overview* and *zoom and filter stage* of our visualization approach. The visualization tool that we are currently working on allows for an interactive exploration of these views.

contour line plot of J. The marks are represented as orange dots. We considered $\boldsymbol{\theta}^0 := (-1.2, 1)^{tr}$ as the initial solution for the applied CG descent.

Figure 2(b) displays the corresponding overview stage. The traditional error curve is in the focus of attention of this view. It is simply the graph of the mapping $k \mapsto J(\boldsymbol{\theta}^k)$. Because this can be seen as an another perspective on the information presented in Figure 2(a), we use orange dots to depict the points that form this graph. Assuming a machine precision of about 10^{-16}, we can safely state that $\approx 10^{-15}$ (at iteration 22) is the final approximation error.

The graphical elements that we added to obtain the novel error curve are black curve segments. The overview stage does not give us a clear enough view on these segments, which is why we explain them in the next section. However, there is one useful aspect that is easy to comprehend without explanation: Note that for errors lying significantly below the assumed machine precision, the shape of the curve segments reflect the resulting numerical noise. Because these last two segments are the only ones to exhibit such a behavior, it indicates either a successful descent or a descent that ran into numerical difficulties. A future study involving real-word data has to prove whether the observed noise is distinctive enough to serve as a reliable indicator of such cases. Clearly, studies of this kind must include an investigation of individual curve segments. The zoom and filter stage presented in the next section is a step in that direction.

2.2 Zoom and Filter

Zooming in yields a clearer view on the black curve segments. We explain what they represent using the zoom view shown in Figure 2(c).

As a first step, we give a review on how gradient descent methods get from solution to solution. The key to this is the *update rule*

$$\boldsymbol{\theta}^{k+1} := \boldsymbol{\theta}^k + \eta_k \boldsymbol{d}^k \tag{4}$$

where $\boldsymbol{d}^k \in \Theta$ must be a *descent direction* at $\boldsymbol{\theta}^k$, i.e. there has to exist a $\bar{\eta}_k > 0$ such that

$$J(\boldsymbol{\theta}^k + \eta \boldsymbol{d}^k) < J(\boldsymbol{\theta}^k) \qquad (\forall \eta \in (0, \bar{\eta}_k]). \tag{5}$$

Observe that constraining $\boldsymbol{d}^k$ in this way guarantees that inequality (3) is always satisfied for a range of arbitrarily small step widths $\eta > 0$.

In the case of the CG descent, the step widths η_k are determined through *line searches* that solve the

approximation problem

$$\eta_k :\approx \min\{\eta > 0 \,|\, \varphi_k'(\eta) = 0\} \tag{6}$$

where $\varphi_k : (0|\infty) \to \mathbb{R}$ is defined by

$$\varphi_k(\eta) := J(\boldsymbol{\theta}^k + \eta \boldsymbol{d}^k) \qquad (\eta \in (0|\infty)). \tag{7}$$

The functions φ_k reflect how J varies when we move along $\boldsymbol{d}^k$ starting at $\boldsymbol{\theta}^k$. Each of them corresponds to a curve segment in Figure 2(c). We look closer at the curve segment that starts at $(10, J(\boldsymbol{\theta}^{10}))$. It depicts a horizontally translated and scaled version of φ_{10}. We know that $\eta = \eta_{10}$ must yield $J(\boldsymbol{\theta}^{11})$. This follows from the applied update rule (4). The approximation rule (6), on the other hand, suggests that η_{10} should be chosen as the smallest stationary point of φ_{10}, i.e. η_{10} very likely indicates a local minimum – and less likely a saddle point – of φ_{10}. Indeed, the curve segment's minimum coincides with $(11, J(\boldsymbol{\theta}^{11}))$. Thus, the part of this curve segment that connects $(10, J(\boldsymbol{\theta}^{10}))$ and $(11, J(\boldsymbol{\theta}^{11}))$ reflects the error variation when moving from solution $\boldsymbol{\theta}^{10}$ to solution $\boldsymbol{\theta}^{11}$. The rest (or *loose end*) of the curve segment reflects how φ_{10} varies in the range from η_{10} to $\eta_{10}^{\max}$ where $\eta_{10}^{\max}$ denotes the maximum step width considered during the applied line search. The applied line search is explained in the next section.

Mathematically, the black curve segments show the graphs of the functions $c_k : (k|\xi_k^{\max}] \to \mathbb{R}$ given by

$$c_k(\xi) := \varphi_k((\xi - k)\eta_k) \quad (\xi \in (k|\xi_k^{\max}]) \tag{8}$$

where $\xi_k^{\max} := k + \eta_k^{\max}/\eta_k$. Following the notation established in the above example, $\eta_k^{\max}$ denotes the maximum step width considered during the applied line search in iteration k.

Note that c_{15} has no loose end, i.e. $\xi_{15}^{\max} = 16$. An interesting filtering of the enhanced error curve is to remove all loose ends, as shown in Figure 2(d). It has become common practice to connect the orange dots that form the traditional error curve by simply using line segments. It is important to note that the shown zoom and filter view presents a more truthful way of connecting the orange dots.

2.3 Details-on-Demand

Now, that we understand what the novel black curve segments represent and how they are constructed, we can explore them individually. In the following two sections, we look at the curve segments representing φ_{10} and φ_{15}. Because we already discussed them in the previous section, their analysis in this final stage can be understood as the final step of the interactive workflow that we intended to present.

We use these two concrete examples to illustrate the applied line search. It consists of two phases. The first phase is always entered, the second one is only entered if necessary. In the case of a line search that only involves phase one, the resulting curve segment has no loose ends. Therefore, the two example curve segments allow us to explain under which conditions the second phase must be entered.

So far, we discussed the central idea of moving from solution to solution in order to approximate an optimal solution $\boldsymbol{\theta}^* \in \Theta$. Searching for a step width can be imagined similarly. For each $k \in \{0, \ldots, K\}$, the applied line search seeks to find η_k by visiting a finite sequence of step widths $(\eta_k^j)_{j=0}^J$ where

$$\eta_k^0 := \frac{1}{1 + \|\boldsymbol{d}^k\|^2} \tag{9}$$

is set as the initial step width.

2.3.1 Phase One Only

We begin by looking at the case $k = 15$ because we know that the associated line search only involved phase one. In Figure 3, the three iterations that were needed to find η_{15} are represented as three vertically arranged plots. The annotated gray circles in the top left corner of each plot indicate the index j. In the following, we discuss them plot by plot. Parallel to this, we explain the new graphical elements that are provided by this details-on-demand view.

The vertical lines mark the current (□···□) and next (■—■) step width; in the upper plot we see η_{15}^0

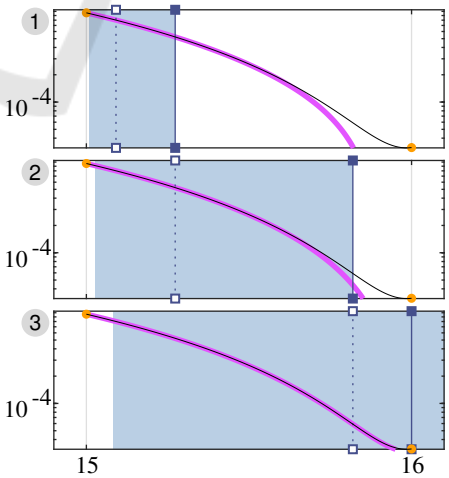


Figure 3: Details-on-demand view of the curve segment representing φ_{15}. The three plots show the three iterations needed to determine η_{15}. This is an example of a line search that only involves phase one. Such curve segments do not have a *loose end*. See Figure 4 for an example of a curve segment with a *loose end*.

and η_{15}^1, respectively. In general, the magenta curve represents the cubic polynomial π that is obtained through a Hermite interpolation with respect to

$$\pi(0) = \varphi_k(0) \qquad \pi(\eta_k^j) = \varphi_k(\eta_k^j) \tag{10}$$

$$\pi'(0) = \varphi_k'(0) \qquad \pi'(\eta_k^j) = \varphi_k'(\eta_k^j) \tag{11}$$

where η_k^j denotes the current step width. Clearly, the current control points 0 and η_{15}^0 provide very limited view on φ_{15}. However, π already well-approximates φ_{15}. The idea behind π is to find its minimizer and limit it through the interval that is indicated by the light blue area. The result is then set as the next step width. As can be seen, η^* exceeds the upper bound of the interval because the upper bound is marked by ■—■, i.e. it was set as the next step width η_{15}^1.

Because the three plots are vertically aligned and share the same axis limits, we directly see that η_{15}^1 severs as the current (□···□) step width in the middle plot. It is also easy to see that π is now an even better approximation of φ_{15}; yet still its minimizer η^* is too large. Again, the upper bound of the interval that is indicated by the light blue area is set as the next step width η_{15}^2. The interval is expanded as compared to the upper plot. In general, the interval is defined by $[\alpha\eta_k^j|\beta\eta_k^j]$ with $0 < \alpha < 1 < \beta$. In our case, we have $\alpha := 0.1$ and $\beta := 3.0$. This explains the expansion of the interval. The interval length grows with the current step width. At the beginning, we consider a small initial step width and we also ensure that we do not move too fast.

Finally, π approximates φ_{15} to an extent where their minimums coincide. Furthermore, the light blue area is contains this minimum. The minimizer η^* is set as the next step width η_{15}^3 and also as the step width η_{15} since $\varphi_{15}'(\eta_{15}^3) \approx 0$.

2.3.2 Phase One and Phase Two

Curve segments with loose ends result from moving too far along the respective function φ_k, i.e. we move past the desired minimizer η_k of φ_k and have to find our way back. We look at the case $k = 10$. Figure 4 shows all iterations that were carried out.

The first three plots show the iterations of phase one. During phase one, the upper bound of the light blue area is set as the next step width three times in a row. In contrast to the previous example, this is not because the respective minimizers η^* of π are too large. The minimizers simply do not exist in the first three cases. At the end of iteration three, we are to the right of the minimizer η_{10} of φ_{10}.

Now, phase two uses that phase one is exited at any step width η_k^j with $\varphi_k'(\eta_k^j) > 0$. Clearly, the slope at the current step width η_{10}^3 (□···□) in iteration four is positive. Moreover, since φ_{10} reflects how J varies along a descent direction, it is also always true that

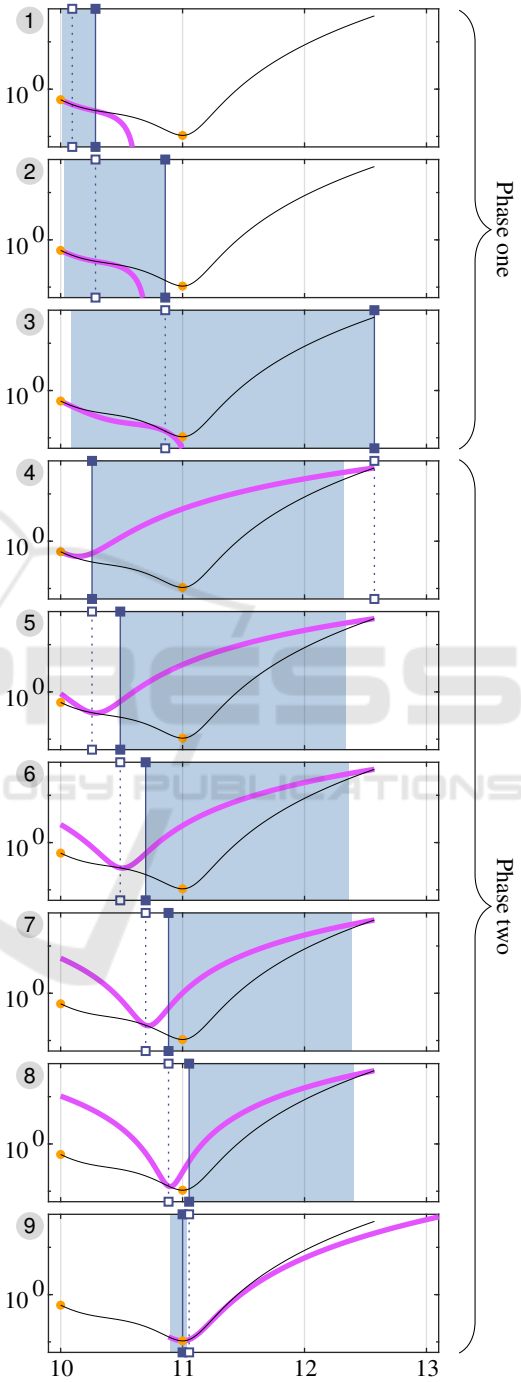


Figure 4: Details-on-demand view of the curve segment representing φ_{10}. The nine plots show the three iterations needed to determine η_{10}. Phase two is entered because phase one visits a step width that lies to the right of the minimizer η_{10} of φ_{10}.

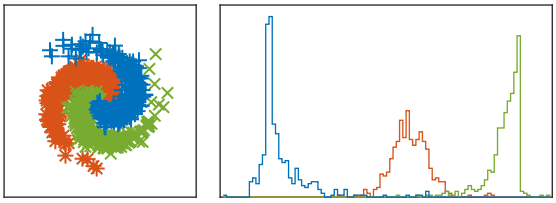
$\varphi'_{10}(\eta) < 0$ for $\eta \to 0$. So, η_{10} must lie in $(a|b)$ with $a = 0$ and $b = \eta^3_{10}$. The central idea behind phase two is to gradually shrink the interval $(a|b)$ while ensuring that $\varphi'_{10}(a) < 0$ and $\varphi'_{10}(b) > 0$. Table 2 summarizes the interval updates. Moreover, it reveals that phase two involves two types of Hermite interpolation:

In the iterations four to eight, the magenta curve represents a quadratic polynomial π. As in phase one, a minimizer η^* of π is determined and limited through the interval that is indicated by the light blue area. In the case of the iterations four to eight, the minimizer η^* is always too small. Thus, the lower bound of the corresponding interval is set as the next step width. Here, the lower and the upper bound are given by $a + 0.1(b-a)$ and $b - 0.1(b-a)$, respectively.

Only in the final iteration, the magenta curve is based on a cubic Hermite interpolation. The minimums of the curve segment and the magenta curve coincides and therefore the minimizer η^* is set as the next step width η^9_{10} and, moreover, η^9_{10} is set as the step width η_{10} since $\varphi'_{10}(\eta^9_{10}) \approx 0$.

Table 2: An overview of the interval updates and the interpolation types applied in phase two.

Phase two			
j	a	b	Interpolation
4	0	η^3_{10}	$\varphi_k(\eta^j_k) > \varphi_k(0) \Rightarrow$ quadratic w.r.t. $\circledast$ $\begin{cases} \pi(a) = \varphi_k(a) \\ \pi'(a) = \varphi'_k(a) \\ \pi(b) = \varphi_k(b) \end{cases}$
5	η^4_{10}	η^3_{10}	
6	η^5_{10}	η^3_{10}	
7	η^6_{10}	η^3_{10}	
8	η^7_{10}	η^3_{10}	
9	η^7_{10}	η^8_{10}	$\varphi_k(\eta^j_k) \leq \varphi_k(0) \Rightarrow$ cubic w.r.t. $\circledast$ and $\pi'(b) = \varphi'_k(b)$



(a) Galaxy. (b) 1D visualization.

Figure 5: A scatter plot of the artificial galaxy data set and a histogram of the 1D representation obtained through the learned DNN mapping. Taken from (Becker et al., 2017).

3 A DNN EXAMPLE

An important application of DNNs that is especially useful in the visualization context is the learning of dimensionality reducing mappings. Such mappings can yield 1D, 2D or 3D representations of original high-dimensional data that allow for straightforward visualizations such as histograms and scatter plots.

In (Becker et al., 2017), we introduced a robust DNN approach to dimensionality reduction. To test our implementation, we conducted a toy experiment involving an artificially generated galaxy data set. In Figure 5, we see one of the results of the paper. The 2D galaxy depicted in Figure 5(a) consists of three classes each containing 480 samples. The suggested DNN was used to learn a non-linear mapping to 1D. The obtained 1D representation of the 2D galaxy is shown Figure 5(b). In the displayed histogram, the three classes are visually separable even without the applied color scheme.

Subsequently, we use this DNN and this data set to give a first impression of enhanced training error curves of actual DNN learning processes. We show two curves obtained through trainings with different mini-batch sizes.

3.1 Batch and Mini-batch Training

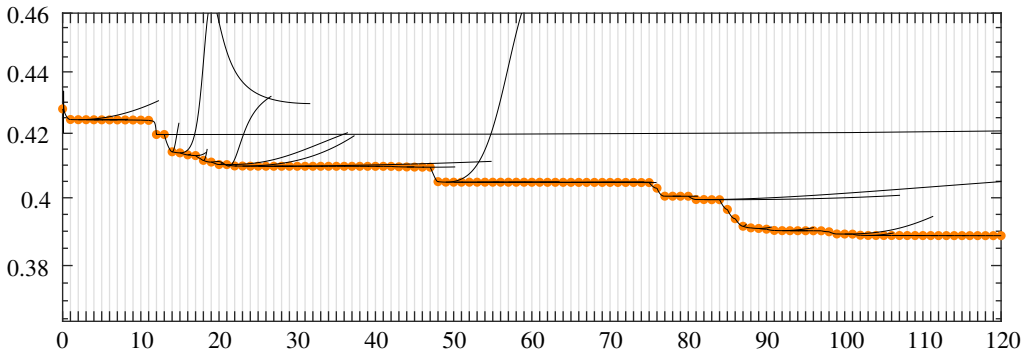
Essentially, a DNN is a parametric machine learning model covering an infinite number of mappings. Its parameters are often formalized as a vector $\boldsymbol{\theta} \in \Theta$ where $\Theta := \mathbb{R}^p$ is called the parameter space of the DNN. The number p of parameters is implicitly set by specifying the internal structure of a DNN. In the context of this paper, a deeper understanding of this internal structure is not required. See (Becker et al., 2017) for a description and also for the definition of the used error function J.

In a (real-world) deep learning scenario, $J(\boldsymbol{\theta}^k)$ is used to assess the data processing

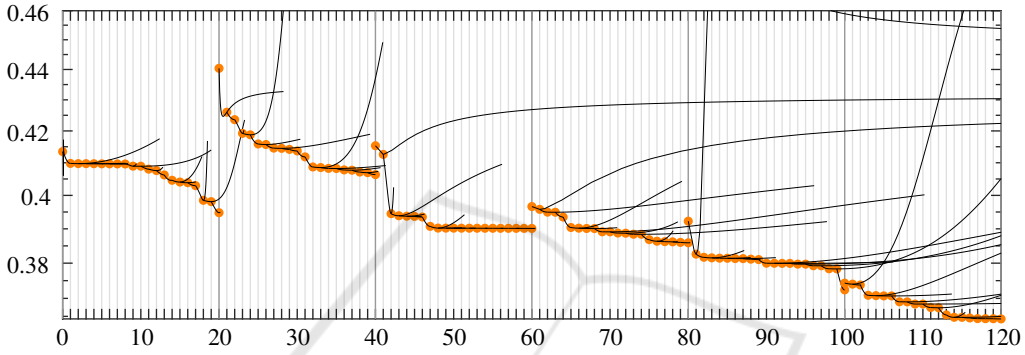
$$\boldsymbol{x}_n \mapsto \boldsymbol{z}_n := \boldsymbol{f}_{\boldsymbol{\theta}^k}(\boldsymbol{x}_n) \in \mathcal{Z} := \mathbb{R}^{d_Z} \tag{12}$$

where $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathcal{X} := \mathbb{R}^{d_X}$ is a set of training samples and $\boldsymbol{f}_{\boldsymbol{\theta}^k} : \mathcal{X} \to \mathcal{Z}$ is the mapping associated with the kth solution $\boldsymbol{\theta}^k \in \Theta$ visited during the DNN training.

DNN training is commonly not performed using all N training samples at once – this training strategy is referred to as *batch training*. Instead, the training samples are randomly split into $B \leq N$ subsets. Then a gradient descent is performed based on one subset at a time. Once all subsets have been presented for a gradient descent, the set of training samples is again randomly split for another B gradient descents. This



(a) Enhanced batch training error curve.



(b) Enhanced mini-batch training error curve.

Figure 6: Enhanced training error curves showing a zoom view on the first 120 gradient descent iterations of the batch training and the mini-batch training, respectively. In the case of the mini-batch training, these 120 iterations result from the following experimental setup: $B = 3$ mini-batches; 20 iterations per CG descent per mini-batch; 2 training epochs.

is repeated for several *training epochs*. For B strictly smaller than N (ideally, $N \bmod B = 0$), each subset is called a *mini-batch*, and accordingly, the training strategy is called *mini-batch training*. Clearly, for B equals 1, we obtain a batch training. The case $B = N$ leads to an *online training*, a training strategy that is rarely used to date.

3.2 Experiments and Results

We conducted two new experiments: In the first, we applied the batch training strategy. In the second, we used the mini-batch training strategy with $B = 3$, i.e. the individual gradient descents were performed on the basis of mini-batches of the size 480. In order to enable a fair comparison of the two experiments, we ensured that each mini-batch contained 160 samples of each of the three classes. Put differently, we made sure that the class proportions were always the same in both experiments.

Figures 6(a) and 6(b) show the zoom view of the first 120 gradient descent iterations of the respective enhanced training error curves – they correspond to the blue and the green traditional curve displayed in Figure 1. The enhanced batch training error curve is displayed in Figure 6(a) and it reveals no substantial differences from the enhanced error curve shown in Figure 2(b). In both cases, the black curve segments connect the points of the respective traditional curve and they may or may not have loose ends. However, it is striking that there exist several long runs of CG descents that appear to cause no significant decrease in error. This is due to the higher complexity of the given deep learning problem.

The enhanced training error curve obtained from the mini-batch training is depicted in Figure 6(b). It strongly differs from both curves discussed so far. In order to be able to interpret the main difference, we need to further elaborate on our mini-batch training realization: We chose to perform a constant number of 20 iterations per CG descent per mini-batch. The 120 iterations therefore display two entire epochs of mini-batch training or, respectively, six consecutive CG descents. The slightly thicker vertical grid lines mark the beginnings of these CG descents.

Now, a closer look along these thicker grid lines reveals that the black curve segments already reflect these beginnings of CG descents. There are distinct

discontinuities at the iterations 20, 40, 60, etc. This can be explained by the fact that the error function changes at these points. The error function of a DNN ultimately depends on the presented sets of training samples. While this is not a new insight, the views that are provided by the enhanced training error curve can help to study these effects in greater detail. E.g. we can now look at how varying the mini-batch size influences the shape of the functions φ_k as regards their change in steepness, smoothness, etc. We can look at the same experiment but at the level of line searches. Interesting new features are the number of loose ends, the number line search iterations needed on average etc. Another experiment is to vary K, the number of iterations per CG descent per mini-batch, or even some ratio of K and the mini-batch size. In any scenario, our approach does not confront us with views that we are unfamiliar with. Instead, we get an extension of widespread monitoring tool and we can access its novel features as needed.

4 CONCLUSION

In this paper, we presented a novel enhanced type of the training error curve that consists of three levels of detail. We showed how these levels of detail can be organized as proposed by the *Visual Information Seeking Mantra* (Shneiderman, 1996): The overview stage covers the details of a traditional training error versus training iteration view. At the same time, it is marked by new graphical elements that can be easily accessed through zooming and filtering. These new elements reflect how the training error varies as we move from solution to solution in a gradient descent iteration. The details-on-demand view allows for an exploration of the methods that determine how far to move along a given descent direction.

We were able to give an example that covered all levels of detail accessible through the novel training error curve. Guided by Sheiderman's Mantra, it was possible to introduce our visualization approach in a way that can be seen as an interactive workflow. The design and development of an interactive tool for the visual exploration of the novel training error curve is one of our future goals. To this end, we identified a list of other gradient descent methods that we aim to include in our tool (see Table 1).

As a final step, we gave a first impression of a hyperparameter analysis that can be realized with our visualization approach. In the course of this, we hinted at further possibilities to gain new insights as regards the mini-batch training of DNNs. While the concrete experimental setup did not reveal entirely new insights about mini-batch training approaches, we pointed out that our way of visualizing gradient descent-related quantities is especially useful since users can access lower level details as needed and by using a monitoring tool that feels familiar.

REFERENCES

Becker, M., Lippel, J., and Stuhlsatz, A. (2017). Regularized nonlinear discriminant analysis - an approach to robust dimensionality reduction for data visualization. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP, (VISIGRAPP 2017)*, pages 116–127. INSTICC, SciTePress.

Dozat, T. (2016). Incorporating nesterov momentum into adam. *ICLR Workshop Submission*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Hohman, F., Kahng, M., Pienta, R., and Chau, D. H. (2018). Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*.

Igel, C. and Hüsken, M. (2003). Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50:105–123.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Polak, E. (2012). *Optimization: Algorithms and Consistent Approximations*. Applied Mathematical Sciences. Springer New York.

Polak, E. and Ribière, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 3(R1):35–43.

Reddi, S. J., Kale, S., and Kumar, S. (2018). On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265 – 278.

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA. IEEE Computer Society.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.