ELLIPSOIDAL TRUST REGION METHODS FOR NEURAL NETWORK TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the use of ellipsoidal trust region constraints for second-order optimization of neural networks. This approach can be seen as a higher-order counterpart of adaptive gradient methods, which we here show to be interpretable as first-order trust region methods with ellipsoidal constraints. In particular, we show that the preconditioning matrix used in RMSProp and Adam satisfies the necessary conditions for provable convergence of second-order trust region methods with standard worst-case complexities. Furthermore, we run experiments across different neural architectures and datasets to find that the ellipsoidal constraints constantly outperform their spherical counterpart both in terms of number of backpropagations and asymptotic loss value. Finally, we find comparable performance to state-of-the-art first-order methods in terms of backpropagations, but further advances in hardware are needed to render Newton methods competitive in terms of time.

1 INTRODUCTION

We consider finite-sum optimization problems of the form

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left[\mathcal{L}(\mathbf{w}) := \sum_{i=1}^{n} \ell(f(\mathbf{w}, \mathbf{x}_i, \mathbf{y}_i)) \right], \tag{1}$$

which typically arise in neural network training, e.g. for empirical risk minimization over a set of data points $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{in} \times \mathbb{R}^{out}, i = 1, \ldots, n$. Here, $\ell : \mathbb{R}^{out} \times \mathbb{R}^{out} \to \mathbb{R}^+$ is a convex loss function and $f : \mathbb{R}^{in} \times \mathbb{R}^d \to \mathbb{R}^{out}$ represents the neural network mapping parameterized by the concatenation of layers $\mathbf{w} \in \mathbb{R}^d$, which is non-convex due to its multiplicative nature and potentially non-linear activation functions. We assume that $\mathcal{L}$ is twice differentiable, i.e. $\mathcal{L} \in C^2(\mathbb{R}^d, \mathbb{R})$ and consider finding a first- and second-order stationary point $\bar{\mathbf{w}}$ for which $\|\nabla\mathcal{L}(\bar{\mathbf{w}})\| \leq \epsilon_g$ and $\lambda_{\min}\left(\nabla^2\mathcal{L}(\bar{\mathbf{w}})\right) \geq -\epsilon_H$. Non-convex optimization problems are ubiquitous in machine learning. Among the most prominent examples are present-day deep neural networks, that have achieved outstanding results on core tasks such as collaborative filtering (Wang et al., 2015), sentence classification (Kim, 2014) and image classification (Krizhevsky et al., 2012).

In the era of big data and deep neural networks, stochastic gradient descent (SGD) is one of the most widely used training algorithms (Bottou, 2010). What makes SGD so attractive is its simplicity and per-iteration cost that are independent of the size of the training set (n) and scale linearly in the dimensionality (d). However, gradient descent is known to be inadequate to optimize functions that are ill-conditioned (Nesterov, 2013; Shalev-Shwartz et al., 2017) and thus adaptive gradient methods that employ dynamic, coordinate-wise learning rates based on past gradients—including Adagrad (Duchi et al., 2011), RMSprop (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2014)—have become a popular alternative, often providing significant speed-ups over SGD. Yet, there exist no theoretical proofs that these methods are faster than gradient descent (Li & Orabona, 2018).

From a theoretical perspective, Newton methods provide stronger convergence guarantees by appropriately transforming the gradient in ill-conditioned regions according to second-order derivatives. It is precisely this Hessian information that allows *regularized* Newton methods to enjoy superlinear local convergence as well as to provably escape saddle points (Conn et al., 2000). While second-order algorithms have a long-standing history even in the realm of neural network training (Hagan & Menhaj, 1994; Becker et al., 1988), they were mostly considered as too computationally and memory

expensive for practical applications. Yet, the seminal work of Martens (2010) renewed interest for their use in deep learning by proposing efficient *Hessian-free* methods that only access second-order information via matrix-vector products which can be computed at the cost of an additional backpropagation (Pearlmutter, 1994; Schraudolph, 2002). Among the class of regularized Newton methods, trust region (Conn et al., 2000) and cubic regularization algorithms (Cartis et al., 2011) are the most principled approaches in the sense that they yield the strongest convergence guarantees. Recently, stochastic extensions have emerged (Xu et al., 2017b; Yao et al., 2018; Kohler & Lucchi, 2017; Gratton et al., 2017), which suggest their applicability for deep learning.

We here propose a simple modification to make TR methods even more suitable for neural network training. Particularly, we build upon the following alternative view on adaptive gradient methods:

While gradient descent can be interpreted as a spherically constrained first-order TR method, preconditioned gradient methods—such as Adagrad—can be seen as first-order TR methods with ellipsoidal trust region constraint.

This observation is particularly interesting since spherical constraints are blind to the underlying geometry of the problem, but ellipsoids can adapt to local landscape characteristics, thereby allowing for more suitable steps in regions that are ill-conditioned. We will leverage this analogy and investigate the use of the Adagrad and RMSProp preconditioning matrices as *ellipsoidal* trust region shapes within a fully stochastic second-order TR algorithm named STORM (Chen et al., 2018; Gratton et al., 2017). While the theory for ellipsoidal TR methods is well-studied (e.g. Conn et al. (2000); Yuan (2015)), no ellipsoid fits all objective functions and our main contribution thus lies in the identification of adequate matrix-induced constraints that lead to provable convergence and significant practical speed-ups for the specific case of deep learning. On the whole, our contribution is threefold:

- We provide a new perspective on adaptive gradient methods that contributes to a better understanding of their inner-workings. Furthermore, we empirically find that many neural network problems exhibit diagonally dominated Hessian matrices which suggests the effectivity of *diagonal* preconditioning. (Section 3)
- We investigate the first application of ellipsoidal TR methods for deep learning. In Theorem 1 we show that the RMSProp matrix can directly be applied as constraint inducing norm in second-order TR algorithms while preserving all convergence guarantees. (Section 4)
- Finally, we provide an experimental benchmark across different real-world datasets and architectures. We also compare against adaptive gradient methods and show results in terms of backprogations, epochs, and wall-clock time; a comparison we were not able to find in the literature. (Section 5)

Our main empirical results demonstrate that ellipsoidal constraints prove to be a very effective modification of the trust region method in the sense that they constantly outperform the spherical TR method, both in terms of number of backprogations and asymptotic loss value on a variety of tasks.

2 Related work

First-order methods The prototypical method for optimizing Eq. (1) is SGD (Robbins & Monro, 1951). While the practical success of SGD in non-convex optimization is unquestioned, the theoretical foundation of this phenomenon is still rather limited. Recent findings suggest the ability of this method to escape saddle points and reach local minima in polynomial time for general non-convex problems, but they either need to artificially add noise to the iterates (Ge et al., 2015; Lee et al., 2016) or make an assumption on the inherent noise of vanilla SGD (Daneshmand et al., 2018). For neural network training, a recent line of research proclaims the effectiveness of SGD, but the results usually come at the cost of fairly strong assumptions such as heavy overparametrization and Gaussian inputs (Du et al., 2017; Brutzkus & Globerson, 2017; Li & Yuan, 2017; Du & Lee, 2018; Allen-Zhu et al., 2018). Adaptive gradient methods (Duchi et al., 2011; Tieleman & Hinton, 2012; Kingma & Ba, 2014) build on the intuition that larger learning rates for smaller gradient components and smaller learning rates for larger gradient components balance their respective influences and thereby make the methods behave as if they were optimizing a more isotropic surface. Such approaches have first been suggested for neural networks by LeCun et al. (2012). Recently, convergence guarantees for such methods are starting to appear (Ward et al., 2018; Li & Orabona, 2018). However, these are not superior to the $\mathcal{O}(\epsilon_g^{-2})$ worst-case complexity of standard gradient descent (Cartis et al., 2012b).

Regularized Newton methods The most principled class of regularized Newton methods are trust region (TR) and adaptive cubic regularization algorithms (ARC) (Conn et al., 2000; Cartis et al., 2011), which repeatedly optimize a local Taylor model of the objective while making sure that the step does not travel too far such that the model stays accurate. While the former finds first-order stationary points within $\mathcal{O}(\epsilon_g^{-2})$, ARC only takes at most $\mathcal{O}(\epsilon_g^{-3/2})$. However, simple modifications to the TR framework allow these methods to obtain the same accelerated rate (Curtis et al., 2017). Both methods take at most $\mathcal{O}(\epsilon_H^{-3})$ iterations to find an ϵ_H approximate second-order stationary point (Cartis et al., 2012a). These rates are optimal for second-order Lipschitz continuous functions (Carmon et al., 2017; Cartis et al., 2012a) and they can be retained even when only sub-sampled gradient and Hessian information is used (Kohler & Lucchi, 2017; Yao et al., 2018; Xu et al., 2017b; Blanchet et al., 2016; Liu et al., 2018; Cartis & Scheinberg, 2017). Furthermore, the involved Hessian information can be computed solely based on Hessian-vector products, which are implementable efficiently for neural networks (Pearlmutter, 1994). This makes these methods particularly attractive for deep learning, but the empirical evidence of their applicability is so far very limited. We are only aware of the works of Liu et al. (2018) and Xu et al. (2017a), which report promising first results but these are by no means fully encompassing.

Gauss-Newton methods An interesting line of research proposes to replace the Hessian by (approximations of) the generalized-Gauss-Newton matrix (GGN) within a Levenberg-Marquardt framework[1] (LeCun et al., 2012; Martens, 2010; Martens & Grosse, 2015). These methods have been termed *hessian-free* since only access to GGN-vector products is required. As the GGN matrix is always positive semidefinite, they cannot leverage negative curvature to escape saddles and hence, there exist no second-order convergence guarantees. Furthermore, there are cases in neural network training where the Hessian is better conditioned than the GGN matrix (Mizutani & Dreyfus, 2008). Nevertheless, the above works report promising preliminary results, most notably Grosse & Martens (2016) report that K-FAC can be faster than SGD on a small convnet. On the other hand, recent findings report performance at best comparable to SGD on the much larger ResNet architecture (Ma et al., 2019). Moreover, Xu et al. (2017a) reports many cases where TR and GGN algorithms perform similarly.

This line of work is to be seen as complementary to our approach since it is straight forward to replace the Hessian in the TR framework with the GGN matrix. Furthermore, the preconditioners used in Martens (2010) and Chapelle & Erhan (2011), namely diagonal estimates of the empirical Fisher and Fisher matrix, respectively, can directly be used as matrix norms in our ellipsoidal TR framework.

3 An alternative view on adaptive gradient methods

Adaptively preconditioned gradient methods update iterates as $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{A}_t^{-1/2} \mathbf{g}_t$, where $\mathbf{g}_t$ is a stochastic estimate of $\nabla \mathcal{L}(\mathbf{w}_t)$ and $\mathbf{A}_t$ is a positive definite symmetric pre-conditioning matrix. In Adagrad, $\mathbf{A}_{ada,t}$ is the un-centered second moment matrix of the past gradients computed as

$$\mathbf{A}_{ada,t} := \mathbf{G}_t \mathbf{G}_t^\intercal + \epsilon \mathbf{I}, \tag{2}$$

where $\epsilon > 0$, $\mathbf{I}$ is the $d \times d$ identity matrix and $\mathbf{G}_t = [\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_t]$. Building up on the intuition that past gradients might become obsolete in quickly changing non-convex landscapes, RMSprop (and Adam) introduce an exponential weight decay leading to the preconditioning matrix

$$\mathbf{A}_{rms,t} := \left((1-\beta)\mathbf{G}_t \operatorname{diag}(\beta^t, \dots, \beta^0)\mathbf{G}_t^\intercal\right) + \epsilon \mathbf{I}, \tag{3}$$

where $\beta \in (0, 1)$. In order to save computational efforts, the diagonal versions $\operatorname{diag}(\mathbf{A}_{ada})$ and $\operatorname{diag}(\mathbf{A}_{rms})$ are more commonly applied in practice, which in turn gives rise to coordinate-wise adaptive stepsizes that are enlarged (reduced) in coordinates that have seen past gradient components with a smaller (larger) magnitude. In that way, the optimization methods can account for gradients of potentially different scales arising from e.g. different layers of the networks.

3.1 Adaptive preconditioning as ellipsoidal Trust Region

Starting from the fact that adaptive methods employ coordinate-wise stepsizes, one can take a principled view of these methods. Namely, their update steps arise from minimizing a first-order

[1] This algorithm is a simplified TR method, initially tailored for non-linear least squares problems (Nocedal & Wright, 2006)

Taylor model of the function $\mathcal{L}$ within an *ellipsoidal* search space around the current iterate $\mathbf{w}_t$, where the diameter of the ellipsoid along a particular coordinate is implicitly given by η_t and $\|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}$. Correspondingly, vanilla (S)GD optimizes the same first-order model within a *spherical* constraint. Fig. 1 (top) illustrates this effect by showing not only the iterates of GD and Adagrad but also the implicit trust regions within which the local models were optimized at each step.[2] Since the models are linear, the constrained minimizer is always found on the boundary.

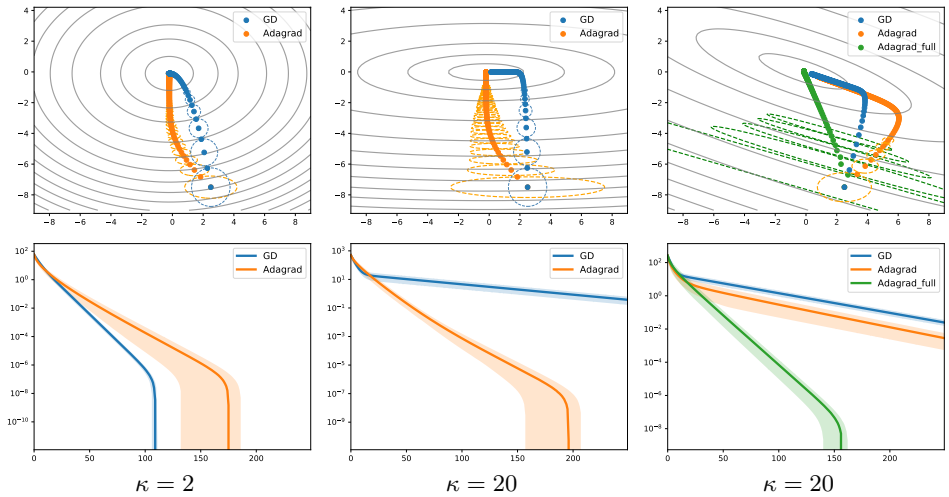

Figure 1: Top: Iterates and implicit trust regions of GD and Adagrad on three quadratic objectives with different condition number κ. Bottom: Average log suboptimality over iterations as well as 90% confidence intervals of 30 runs with random initialization

It is well known that GD struggles to progress towards the minimizer of quadratics along low-curvature directions (see e.g., Goh (2017)). While this effect is negligible for well-conditioned objectives (Fig. 1, left), it leads to drastically slow-down when the problem is ill-conditioned (Fig. 1, center). Particularly, once the method has reached the bottom of the valley, it struggles to make progress along the horizontal axis. Here is precisely where the advantage of adaptive stepsize methods comes into play. As illustrated by the dashed lines, Adagrad's search space is damped along the direction of high curvature (vertical axis) and elongated along the low curvature direction (horizontal axis). This allows the method to move further horizontally early on to enter the valley with a smaller distance to the optimizer $\mathbf{w}^*$ along the low curvature direction which accelerates convergence.

Theorem 1 (Preconditioned gradient methods as TR)**.** *A preconditioned gradient step*

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \mathbf{s}_t := -\eta_t \mathbf{A}_t^{-1} \mathbf{g}_t \tag{4}$$

with stepsize $\eta_t > 0$, symmetric positive definite preconditioner $\mathbf{A}_t \in \mathbb{R}^{d\times d}$ and $\mathbf{g}_t \neq 0$ minimizes a first-order model around $\mathbf{w}_t \in \mathbb{R}^d$ in an ellipsoid given by $\mathbf{A}_t$ in the sense that

$$\mathbf{s}_t := \arg\min_{\mathbf{s}\in\mathbb{R}^d} \left[m_t^1(\mathbf{s}) = \mathcal{L}(\mathbf{w}_t) + \mathbf{s}^\intercal \mathbf{g}_t \right], \qquad s.t. \quad \|\mathbf{s}\|_{\mathbf{A}_t} \leq \eta_t \|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}. \tag{5}$$

Corollary 1 (Rmsprop)**.** *The step $\mathbf{s}_{rms,t} := -\eta_t \mathbf{A}_{rms,t}^{-1/2} \mathbf{g}_t$ minimizes a first-order Taylor model around $\mathbf{w}_t$ in an ellipsoid given by $\mathbf{A}_{rms,t}^{1/2}$ (Eq. 3) in the sense that*

$$\mathbf{s}_{rms,t} := \arg\min_{\mathbf{s}\in\mathbb{R}^d} \left[m_t^1(\mathbf{s}) = \mathcal{L}(\mathbf{w}_t) + \mathbf{s}^\intercal \mathbf{g}_t \right], \qquad s.t. \quad \|\mathbf{s}\|_{\mathbf{A}_{rms,t}^{1/2}} \leq \eta_t \|\mathbf{g}_t\|_{\mathbf{A}_{rms,t}^{-1/2}}. \tag{6}$$

Equivalent results can be established for Adam using $\mathbf{g}_{adam,t} := (1-\beta)\sum_{k=0}^{t} \beta^{t-k} \mathbf{g}_t$ as well as for Adagrad by replacing the matrix $\mathbf{A}_{ada}$ into the constraint in Eq. (6). Of course, the update procedure in Eq. (5) is merely a reinterpretation of the original preconditioned update, and thus the employed trust region radii are defined *implicitly* by the current gradient and stepsize.

[2]For illustrative purposes, we only plot every other trust region.

3.2 DIAGONAL VERSUS FULL PRECONDITIONING

A closer look at Fig. 1 reveals that the first two problems come with level sets that are perfectly *axis-aligned*, which makes these objectives particularly attractive for diagonal preconditioning. For comparison, on the right of Fig. 1, we report another quadratic problem instance, where the Hessian is no longer zero on the off-diagonals. As can be seen, the interaction between coordinates introduces a tilt in the level sets and reduces the superiority of diagonal Adagrad over plain GD. However, using the full preconditioner $\mathbf{A}_{ada}$ re-establishes the original speed up. Yet, non-diagonal preconditioning comes at the cost of taking the inverse square root of a large matrix, which is why this approach has been relatively unexplored (see Agarwal et al. (2018) for a recent exception).

Interestingly, early results by Becker et al. (1988) on the curvature structure of neural nets report a strong diagonal dominance of the Hessian matrix $\nabla^2\mathcal{L}(\mathbf{w})$. This suggests that the loss surface is indeed somewhat axis-aligned. However, the reported numbers are only for tiny feed-forward networks of at most 256 parameters. Therefore, we generalize these findings in the following to larger networks. Furthermore, we contrast the diagonal dominance of real Hessian matrices to the expected behavior of random Wigner matrices. Of course, true Hessians do not have i.i.d. entries but the symmetry of Wigner matrices suggests that this baseline is not completely off. For this purpose, let $\delta_{\mathbf{A}}$ define the ratio of diagonal to overall mass of a matrix $\mathbf{A}$, i.e. $\delta_{\mathbf{A}} := \frac{\sum_i |\mathbf{A}_{i,i}|}{\sum_i \sum_j |\mathbf{A}_{i,j}|}$ as in (Becker et al., 1988).

Proposition 1. *For random Gaussian*[3] *Wigner matrix* $\mathbf{W}$ *formed as*

$$\mathbf{W}_{i,j} = \mathbf{W}_{j,i} := \begin{cases} \sim \mathcal{N}(0, \sigma_1), & i < j \\ \sim \mathcal{N}(0, \sigma_2), & i = j, \end{cases} \tag{7}$$

where $\sim$ *stands for i.i.d. draws (Wigner, 1993), the expected share of diagonal mass* $\delta_{\mathbf{W}}$ *amounts to*

$$\mathbb{E}\left[\delta_{\mathbf{W}}\right] = \left(1 + (d-1)\frac{\sigma_2}{\sigma_1}\right)^{-1}.$$

Thus, if we suppose the Hessian at any given point $\mathbf{w}$ were a random Wigner matrix we would expect the share of diagonal mass to fall with $\mathcal{O}(1/d)$ as the network grows in size. Yet, as can be seen in Fig. 2 the diagonal mass $\delta_{\mathbf{H}}$ of real-world neural networks stays way above this theoretical value at random initialization, during training and after convergence.

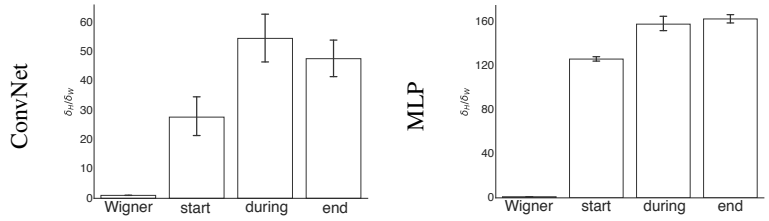

Figure 2: Diagonal mass of Hessian $\delta_{\mathbf{H}}$ relative to $\delta_{\mathbf{W}}$ of corresponding Wigner matrix at random initialization, middle and end of training with RMSProp on CIFAR-10. Mean and 95% CI over 10 independent runs.

These findings are in line with Becker et al. (1988) and suggest that full matrix preconditioning is most probably not worth the effort for neural networks. Consequently, we use diagonal preconditioning for both first- and second-order methods in all of our experiments in Section 5.

4 SECOND-ORDER TRUST REGION METHODS

Cubic regularization (Nesterov & Polyak, 2006; Cartis et al., 2011) and trust region methods belong to the family of globalized Newton methods. Both frameworks compute parameter updates by optimizing regularized (former) or constrained (latter) second-order Taylor models of the objective $\mathcal{L}$

[3]The argument naturally extends to any distribution with positive expected absolute values.

around the current iterate $\mathbf{w}_t$.[4] In particular, in iteration t the update step of the trust region algorithm is computed as

$$\min_{\mathbf{s}\in\mathbb{R}^d} \left[m_t(\mathbf{s}) := \mathcal{L}(\mathbf{w}_t) + \mathbf{g}_t^\intercal \mathbf{s} + \frac{1}{2}\mathbf{s}^\intercal \mathbf{B}_t \mathbf{s} \right], \qquad \text{s.t. } \|\mathbf{s}\|_{\mathbf{A}_t} \leq \Delta_t \tag{8}$$

where $\Delta_t > 0$ and $\mathbf{g}_t$ and $\mathbf{B}_t$ are either $\nabla\mathcal{L}(\mathbf{w}_t)$ and $\nabla^2\mathcal{L}(\mathbf{w}_t)$ or suitable approximations. The matrix $\mathbf{A}_t$ induces the shape of the constraint set. So far, the common choice for neural networks is $\mathbf{A}_t := \mathbf{I},\ \forall t$ which gives rise to spherical trust regions (Xu et al., 2017a; Liu et al., 2018). By solving the *constrained* problem (8), TR methods overcome the problem that pure Newton steps may be ascending, attracted by saddles or not even computable. See Appendix B for more details.

4.1 Convergence of ellipsoidal Trust Region methods

Inspired by the success of adaptive gradient methods, we investigate the use of their preconditioning matrices as norm inducing matrices for second-order TR methods. The crucial condition for convergence is that the applied norms are not degenerate during the entire minimization process in the sense that the ellipsoids do not flatten out (or blow up) completely along any given direction. The following definition formalizes this intuition.

Definition 1 (Uniformly equivalent norms)**.** The norms $\|\mathbf{w}\|_{\mathbf{A}_t} := \left(\mathbf{w}^\intercal \mathbf{A}_t \mathbf{w}\right)^{1/2}$ induced by symmetric positive definite matrices $\mathbf{A}_t$ are called uniformly equivalent, if $\exists \mu \geq 1$ such that

$$\frac{1}{\mu}\|\mathbf{w}\|_{\mathbf{A}_t} \leq \|\mathbf{w}\|_2 \leq \mu \|\mathbf{w}\|_{\mathbf{A}_t}, \qquad \forall \mathbf{w} \in \mathbb{R}^d, \forall t = 1, 2, \ldots. \tag{9}$$

We now establish a result which shows that the RMSProp ellipsoid is indeed uniformly equivalent.

Proposition 2 (Uniform equivalence)**.** *Suppose $\|\mathbf{g}_t\|^2 \leq L_H^2$ for all $\mathbf{w}_t \in \mathbb{R}^d$, $t = 1, 2, \ldots$ Then there always exists $\epsilon > 0$ such that the proposed preconditioning matrices $\mathbf{A}_{rms,t}$ (Eq. 3) are uniformly equivalent, i.e. Def. 1 holds. The same holds for the diagonal variant.*

Consequently, the ellipsoids $\mathbf{A}_{rms,t}$ can directly be applied to any convergent TR framework without losing convergence guarantees (Conn et al. (2000), Theorem 6.6.8).[5] Interestingly, this result cannot be established for $\mathbf{A}_{ada,t}$, which reflects the widely known vanishing stepsize problem that arises since squared gradients are continuously added to the preconditioning matrix. At least partially, this effect inspired the development of RMSprop (Tieleman & Hinton, 2012) and Adadelta (Zeiler, 2012).

Why ellipsoids? There are many sources for ill-conditioning in neural networks such as un-centered and correlated inputs (LeCun et al., 2012), saturated hidden units, and different weight scales in different layers (Van Der Smagt & Hirzinger, 1998). While the quadratic term of model (8) accounts for such ill-conditioning to some extent, the spherical constraint is completely blind towards the loss surface. Thus, it is advisable to instead measure distances in norms that reflect the underlying geometry (see Chapter 7.7 in Conn et al. (2000)). The ellipsoids we propose are such that they allow for longer steps along coordinates that have seen small gradient components in past and vice versa. Thereby the TR shape is adaptively adjusted to fit the current region of the non-convex loss landscape. This procedure is not only effective when the iterates are in an ill-conditioned neighborhood of a minimizer (Figure 1), but it also helps to escape elongated plateaus (see autoencoder in Section 5).

4.2 A stochastic TR framework for neural network training

Since neural network training often constitutes a large scale learning problem in which the number of datapoints n is very high, we here opt for a fully stochastic TR framework (Chen et al., 2018) in order to circumvent memory issues and reduce computational complexity. Given that the involved function and derivative estimates are sufficiently accurate with a fixed probability, such a framework

[4] In the following we only treat TR methods, but we would like to emphasize that the use of matrix induced norms can directly be transferred to the cubic regularization framework.

[5] Note that the assumption of bounded batch gradients, i.e. smooth objectives, is common in the analysis of stochastic algorithms (Allen-Zhu, 2017; Defazio et al., 2014; Schmidt et al., 2017; Duchi et al., 2011).

retains the $O\left(\epsilon_g^{-2}, \epsilon_H^{-3}\right)$ convergence rate of deterministic TR methods to stationary points in expectation (Blanchet et al., 2016; Gratton et al., 2017). For finite-sum objectives such as Eq. (1), the required level of accuracy can be obtained by simple mini-batching. In that case, the same rate can be proven for Algorithm 1 with $\mathbf{A}_{rms}$ ellipsoids as Proposition 2 guarantees the sufficient condition for convergence with adaptive norms (see Appendix C below as well as AN.1 in Conn et al. (2000)).[6]

Algorithm 1 Stochastic Ellipsoidal Trust Region Method

1: **Input:** $\mathbf{w}_0 \in \mathbb{R}^d$, $\gamma_1, \gamma_2 > 1, 1 > \eta_2 > \eta_1 > 0$, $\Delta_0 > 0$, $T \geq 1$, $|\mathcal{S}_0|, \mu \geq 1, \epsilon > 0$
2: **for** $t = 0, 1, \ldots,$ until convergence **do**
3: Sample $\mathcal{L}_t$, $\mathbf{g}_t$ and $\mathbf{B}_t$ with batch sizes $|\mathcal{S}_{\mathcal{L},t}|, |\mathcal{S}_{\mathbf{g},t}|, |\mathcal{S}_{\mathbf{B},t}|$
4: Compute preconditioner $\mathbf{A}_t$ s.t. Def. 1 holds
5: Obtain $\mathbf{s}_t$ by solving $m_t(\mathbf{s}_t)$ (Eq. 8) s.t. Eq 36 holds
6: Compute actual over predicted decrease on batch

$$\rho_{\mathcal{S},t} = \frac{\mathcal{L}_{\mathcal{S}}(\mathbf{w}_t) - \mathcal{L}_{\mathcal{S}}(\mathbf{w}_t + \mathbf{s}_t)}{m_t(\mathbf{0}) - m_t(\mathbf{s}_t)} \tag{10}$$

7: Set

$$\Delta_{t+1} = \begin{cases} \gamma_1 \Delta_t & \text{if } \rho_{\mathcal{S},t} > \eta_2 \text{ (very successful)} \\ \Delta_t & \text{if } \eta_2 \geq \rho_{\mathcal{S},t} \geq \eta_1 \text{ (successful)} \\ \Delta_t / \gamma_2 & \text{if } \rho_{\mathcal{S},t} < \eta_1 \text{ (unsuccessful)} \end{cases}, \ \mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \mathbf{s}_t & \text{if } \rho_{\mathcal{S},t} \geq \eta_1 \\ \mathbf{w}_t & \text{otherwise} \end{cases}$$

8: **end for**

The main difference between this and the existing approaches of Kohler & Lucchi (2017); Xu et al. (2017b); Yao et al. (2018); Cartis & Scheinberg (2017) lies in the computation of ρ, which is traditionally computed as full function- over stochastic model decrease. Computing ρ solely based on sub-sampled quantities has the nice side-effect of disentangling two potential sources of error: an overoptimistic trust region radius or insufficiently small batch sizes. Indeed, the quantity $\rho_{\mathcal{S}}$ from Eq. (10) is an unbiased estimate of the ρ used in fully deterministic algorithms and hence the trust region radius is adjusted purely based on the current adequacy of the local quadratic approximation.

5 Experiments

Trust region methods To validate our claim that ellipsoidal TR methods yield improved performance over spherical ones, we run a set of experiments on two image datasets and three types of network architectures. As can be seen in Figure 3, the ellipsoidal TR methods consistently outperform their spherical counterpart in the sense that they reach full training accuracy substantially faster on all problems. Moreover, their limit points are in all cases lower than those of the uniform TR method. Interestingly, this makes an actual difference in the image reconstruction quality of autoencoders (see Figure 11). We thus draw the clear conclusion that the ellipsoidal trust region constraints we propose are to be preferred over their spherical counterpart when training neural networks. Both the experimental and architectural details are provided in Appendix C.

Benchmark with SGD To put the previous results into context, we also benchmark several state-of-the-art gradient methods. We fix their sample size to 32 (as advocated e.g. in Masters & Luschi (2018)) but grid search the stepsize since it is the ratio of these two quantities that effectively determines the level of stochasticity (Jastrzebski et al., 2017). As the TR methods have a larger batch size[7] of 128–512, we report results both in terms of number of backpropagations and epochs for a fair comparison. A close look at Figure 4 and 9 (Appendix) indicates that the ellipsoidal TR methods can be slightly superior in terms of backprops but at best manage to keep pace with first-order methods

[6]The proof is ommited here for brevity but it proceeds analogously to Gratton et al. (2017) by replacing $\|\cdot\|$ with $\|\cdot\|_{\mathbf{A}_{rms,k}}$ and noting that $\|\nabla f_k\| \geq \epsilon$ implies $\|\nabla f_k\|_{\mathbf{A}_{rms,k}} \geq \frac{1}{\mu}\epsilon =: \epsilon_g$ by Prop. 2 and Def. 1.

[7]We observed weaker performance when running with smaller batch size. We hypothesize that second-order methods extract more information of each batch and are thus likely to "overfit" small batches in each step.

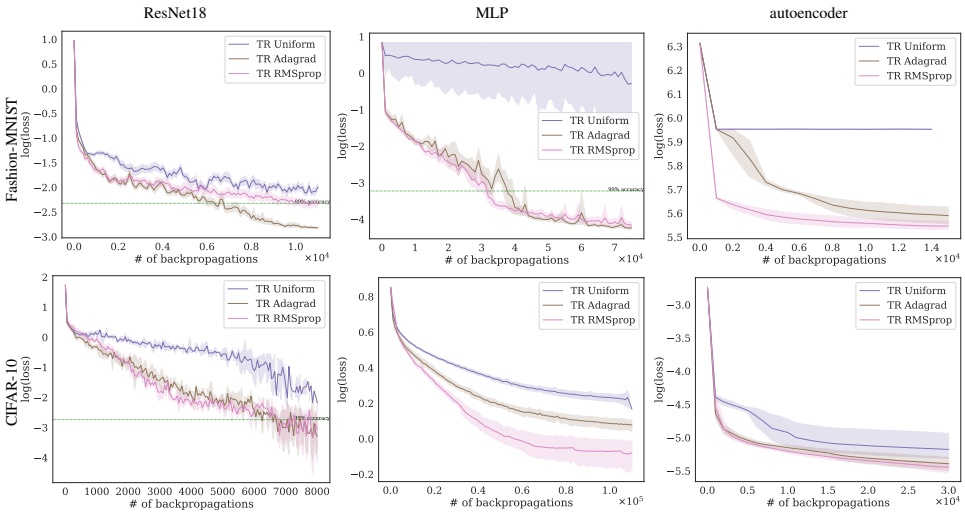

Figure 3: Log loss over backpropagations. Mean and 95% CI of 10 runs. Green dotted line indicates 99% acc.

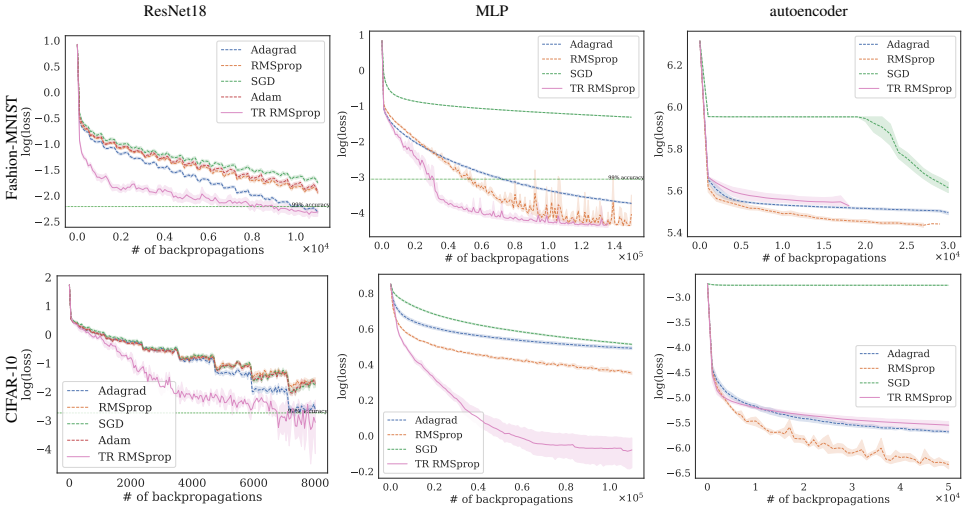

Figure 4: Log loss over backpropagations. Same setting as Figure 3. See Figure 9 for epoch results.

in terms of epochs. Furthermore, the limit points of both first- and second-order methods yield the same order of loss in most experiments. When taking gradient norms into account (plot omitted), we indeed find no spurious local minima and only the autoencoders give rise to saddle points.

6 Conclusion

We investigated the use of ellipsoidal trust region constraints for neural networks. We have shown that the RMSProp matrix satisfies the necessary conditions for convergence and our experimental results demonstrate that ellipsoidal TR methods outperform their spherical counterparts significantly. We thus consider the development of further ellipsoids that can potentially adapt even better to the loss landscape such as e.g. (block-) diagonal hessian approximations (e.g. Bekas et al. (2007)) or approximations of higher order derivatives as an interesting direction of future research.

Yet, the gradient method benchmark indicates that the value of Hessian information for neural network training is limited for mainly three reasons: 1) second-order methods rarely yield better limit points, which suggests that saddles and spurious local minima are not a major obstacle; 2) gradient methods can run on smaller batch sizes which is beneficial in terms of epoch and when memory is

limited; 3) The per-iteration time complexity is noticeably lower for first-order methods (Figure 10). These observations suggest that advances in hardware and distributed second-order algorithms (e.g., Osawa et al. (2018); Dünner et al. (2018)) will be needed before Newton-type methods can replace (stochastic) gradient methods in deep learning.

REFERENCES

Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, and Yi Zhang. The case for full-matrix adaptive regularization. *arXiv preprint arXiv:1806.02958*, 2018.

Guillaume Alain, Nicolas Le Roux, and Pierre-Antoine Manzagol. Negative eigenvalues of the hessian in deep neural networks. 2018.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Sue Becker, Yann Le Cun, et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pp. 29–37. San Matteo, CA: Morgan Kaufmann, 1988.

Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12):1214–1229, 2007.

Jose Blanchet, Coralia Cartis, Matt Menickelly, and Katya Scheinberg. Convergence rate analysis of a stochastic trust region method for nonconvex optimization. *arXiv preprint arXiv:1609.07428*, 2016.

Leon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. *arXiv preprint arXiv:1702.07966*, 2017.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points i. *arXiv preprint arXiv:1710.11606*, 2017.

Coralia Cartis and Katya Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, pp. 1–39, 2017.

Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

Coralia Cartis, Nicholas IM Gould, and Ph L Toint. Complexity bounds for second-order optimality in unconstrained optimization. *Journal of Complexity*, 28(1):93–108, 2012a.

Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. *How Much Patience to You Have?: A Worst-case Perspective on Smooth Noncovex Optimization*. Science and Technology Facilities Council Swindon, 2012b.

Olivier Chapelle and Dumitru Erhan. Improved preconditioner for hessian free optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 201, 2011.

Ruobing Chen, Matt Menickelly, and Katya Scheinberg. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487, 2018.

Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.

Frank E Curtis and Daniel P Robinson. Exploiting negative curvature in deterministic and stochastic optimization. *arXiv preprint arXiv:1703.00412*, 2017.

Frank E Curtis, Daniel P Robinson, and Mohammadreza Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{3-2})$ for nonconvex optimization. *Mathematical Programming*, 162(1-2):1–32, 2017.

Hadi Daneshmand, Jonas Kohler, Aurelien Lucchi, and Thomas Hofmann. Escaping saddles with stochastic gradients. *arXiv preprint arXiv:1803.05999*, 2018.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pp. 1646–1654, 2014.

Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pp. 1067–1077, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Celestine Dünner, Aurelien Lucchi, Matilde Gargiani, An Bian, Thomas Hofmann, and Martin Jaggi. A distributed second-order algorithm you can trust. *arXiv preprint arXiv:1806.07569*, 2018.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points-online stochastic gradient for tensor decomposition. In *COLT*, pp. 797–842, 2015.

Gabriel Goh. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL `http://distill.pub/2017/momentum`.

Serge Gratton, Clément W Royer, Luís N Vicente, and Zaikun Zhang. Complexity and global rates of trust-region methods based on probabilistic models. *IMA Journal of Numerical Analysis*, 2017.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582, 2016.

Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on Learning Theory*, pp. 1246–1257, 2016.

Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. *arXiv preprint arXiv:1805.08114*, 2018.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, pp. 597–607, 2017.

Liu Liu, Xuanqing Liu, Cho-Jui Hsieh, and Dacheng Tao. Stochastic second-order methods for non-convex optimization with inexact hessian and gradient. *arXiv preprint arXiv:1809.09853*, 2018.

Linjian Ma, Gabe Montague, Jiayu Ye, Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Inefficiency of k-fac for large batch size training. *arXiv preprint arXiv:1903.06237*, 2019.

James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pp. 735–742, 2010.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Eiji Mizutani and Stuart E Dreyfus. Second-order stagewise backpropagation for hessian-matrix analyses and investigation of negative curvature. *Neural Networks*, 21(2-3):193–203, 2008.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Jorge Nocedal and Stephen J Wright. *Numerical optimization, 2nd Edition*. Springer, 2006.

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Second-order optimization method for large mini-batch: Training resnet-50 on imagenet in 35 epochs. *arXiv preprint arXiv:1811.12019*, 2018.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Herbert Robbins and Sutton Monro. A stochastic approximation method. In *The Annals of Mathematical Statistics - Volume 22, Number 3*. Institute of Mathematical Statistics, 1951.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. *arXiv preprint arXiv:1703.07950*, 2017.

Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Patrick Van Der Smagt and Gerd Hirzinger. Solving the ill-conditioning in neural network learning. In *Neural networks: tricks of the trade*, pp. 193–206. Springer, 1998.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1235–1244, 2015.

Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *arXiv preprint arXiv:1806.01811*, 2018.

Eugene P Wigner. Characteristic vectors of bordered matrices with infinite dimensions i. In *The Collected Works of Eugene Paul Wigner*, pp. 524–540. Springer, 1993.

Peng Xu, Farbod Roosta-Khorasan, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv preprint arXiv:1708.07827*, 2017a.

Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *arXiv preprint arXiv:1708.07164*, 2017b.

Zhewei Yao, Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Inexact non-convex newton-type methods. *arXiv preprint arXiv:1802.06925*, 2018.

Ya-xiang Yuan. Recent advances in trust region algorithms. *Mathematical Programming*, 151(1): 249–281, 2015.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Appendix A: Proofs

A NOTATION

Throughout this work, scalars are denoted by regular lower case letters, vectors by bold lower case letters and matrices as well as tensors by bold upper case letters. By $\|\cdot\|$ we denote an arbitrary norm. For a symmetric positive definite matrix $\mathbf{A}$ we introduce the compact notation $\|\mathbf{w}\|_{\mathbf{A}} = \left(\mathbf{w}^\intercal \mathbf{A}\mathbf{w}\right)^{1/2}$, where $\mathbf{w} \in \mathbb{R}^d$.

B EQUIVALENCE OF PRECONDITIONED GRADIENT DESCENT AND FIRST-ORDER TRUST REGION METHODS

Theorem 2 (Theorem 1 restated)**.** *A preconditioned gradient step*

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \mathbf{s}_t := -\eta_t \mathbf{A}_t^{-1}\mathbf{g}_t \tag{11}$$

with stepsize $\eta_t > 0$, symmetric positive definite preconditioner $\mathbf{A}_t \in \mathbb{R}^{d\times d}$ and $\mathbf{g}_t \neq 0$ minimizes a first-order local model around $\mathbf{w}_t \in \mathbb{R}$ in an ellipsoid given by $\mathbf{A}_t$ in the sense that

$$\mathbf{s}_t := \arg\min_{\mathbf{s}\in\mathbb{R}^d} \left[m_t^1(\mathbf{s}) = \mathcal{L}(\mathbf{w}_t) + \mathbf{s}^\intercal \mathbf{g}_t\right], \quad \textit{s.t. } \|\mathbf{s}\|_{\mathbf{A}_t} \leq \eta_t \|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}. \tag{12}$$

Proof. We start the proof by noting that the optimization problem in Eq. (12) is convex. For $\eta_t > 0$ the constraint satisfies the Slater condition since 0 is a strictly feasible point. As a result, any KKT point is a feasible minimizer and vice versa.

Let $L(\mathbf{s}, \lambda)$ denote the Lagrange dual of Eq. (5)

$$L(\mathbf{s}, \lambda) := \mathcal{L}(\mathbf{w}_t) + \mathbf{s}^\intercal \mathbf{g}_t + \lambda\left(\|\mathbf{s}\|_{\mathbf{A}} - \eta_t \|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}\right). \tag{13}$$

Any point $\mathbf{s}$ is a KKT point if and only if the following system of equations is satisfied

$$\nabla_{\mathbf{s}} L(\mathbf{s}, \lambda) = \mathbf{g}_t + \frac{\lambda}{\|\mathbf{s}\|_{\mathbf{A}t}}\mathbf{A}_t\mathbf{s} = 0 \tag{14}$$

$$\lambda\left(\|\mathbf{s}\|_{\mathbf{A}_t} - \eta_t\|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}\right) = 0. \tag{15}$$

$$\|\mathbf{s}\|_{\mathbf{A}_t} - \eta_t\|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}} \leq 0 \tag{16}$$

$$\lambda \geq 0. \tag{17}$$

For $\mathbf{s}_t$ as given in Eq. (4) we have that

$$\|\mathbf{s}_t\|_{\mathbf{A}_t} = \sqrt{\eta_t^2 \mathbf{g}_t (\mathbf{A}_t^{-1})^\intercal \mathbf{A}_t \mathbf{A}_t^{-1}\mathbf{g}_t} = \eta_t\sqrt{\mathbf{g}_t\mathbf{A}_t^{-1}\mathbf{g}_t} = \eta_t\|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}. \tag{18}$$

and thus 15 and 16 hold with equality such that any $\lambda \geq 0$ is feasible. Furthermore,

$$\nabla_{\mathbf{s}} L(\mathbf{s}_t, \lambda) = \nabla f(\mathbf{w}_t) + \frac{\lambda}{\|\mathbf{s}_t\|_{\mathbf{A}_t}}\mathbf{A}_t\mathbf{s}_t \overset{(4)}{=} \mathbf{g}_t - \eta_t\frac{\lambda}{\eta_t\|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}}\mathbf{A}_t\mathbf{A}_t^{-1}\mathbf{g}_t = \mathbf{g}_t - \frac{\lambda}{\|\mathbf{g}_t\|_{\mathbf{A}_t^{-1}}}\mathbf{g}_t \tag{19}$$

is zero for $\lambda = \|\mathbf{g}_t\|_{\mathbf{A}^{-1}} \geq 0$. As a result, $\mathbf{s}_t$ is a KKT point of the convex problem 5 which proves the assertion.

□

To illustrate this theoretical result we run gradient descent and Adagrad as well as the two corresponding first-order TR methods[8] on an ill-conditioned quadratic problem. While the method 1st TR

[8]Essentially Algorithm 1 with m_t based on a *first* order Taylor expansion, i.e. $m_t^1(\mathbf{s})$ as in Eq. (12).

optimizes a linear model within a ball in each iteration, 1st TR_{ada} optimizes the same model over the ellipsoid given by the Adagrad matrix $\mathbf{A}_{ada}$. The results in Figure 5 show that the methods behave very similar to their constant stepsize analogues.

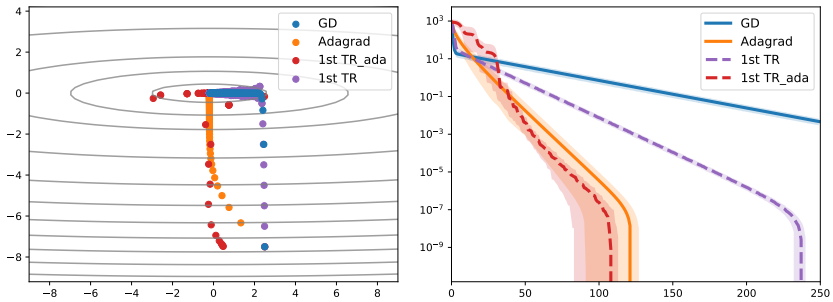

Figure 5: Iterates (left) and log suboptimality (right) of GD, Adagrad and two full-featured first-order TR algorithms of which one (1st TR) is spherically constraint and the other (1st TR_{ada}) uses $\mathbf{A}_{ada}$ as ellispoid.

C Convergence of ellipsoidal TR methods

Spherical constrained TR methods Under standard smoothness assumptions, spherical TR algorithms achieve ϵ_g criticality after $O(\epsilon_g^{-2})$ iterations and additionally ϵ_H almost positive curvature in $O(\epsilon_H^{-3})$ iterations. These rates are attained without the need of actually knowing the Hessian's Lipschitz constant. The complete statement of the convergence results for TR methods can be found in Theorem 4.3 of (Cartis et al., 2012a). Interestingly, these rates can be improved to match the (optimal) $\mathcal{O}(\epsilon^{-3/2})$ first-order worst case complexity of Cubic Regularization by applying small modifications to the TR framework. As stated in Section B.2 the involved subproblems do not need to be solved globally in each iteration. For both, cubic regularization and trust region methods, many stochastic extensions have emerged in literature that alleviate the need to compute exact derivative information without losing the above mentioned convergence guarantees with high probability (Kohler & Lucchi, 2017; Xu et al., 2017a;b; Blanchet et al., 2016; Gratton et al., 2017)). For the deep learning setting, the analysis of Blanchet et al. (2016) is most relevant since it also allows the algorithm to run solely based on sub-sampled function evaluations.

Ellipsoidal constrained TR methods In order to prove such results for ellipsoidal Trust Region methods one must ensure that the applied norms are coherent during the complete minimization process in the sense that the ellipsoids do not flatten out (or blow up) completely along any given direction. This intuition is formalized in Assumption 1 which we restate here for the sake of clarity.

Definition 2 (Definition 1 restated)**.** There exists a constant $\mu \geq 1$ such that

$$\frac{1}{\mu}\|\mathbf{w}\|_{\mathbf{A}_t} \leq \|\mathbf{w}\|_2 \leq \mu\|\mathbf{w}\|_{\mathbf{A}_t}, \qquad \forall t, \forall \mathbf{w} \in \mathbb{R}^d. \tag{20}$$

Having uniformly equivalent norms is necessary and sufficient to prove that ellipsoidal TR methods enjoy the same convergence rate as classical ball constrained Trust Region algorithms. Towards this end, Conn et al. (2000) identify the following sufficient condition on the basis of which we will prove that our proposed ellipsoid $\mathbf{A}_{rms}$ is indeed uniformly equivalent under some mild assumptions.

Lemma 1 (Theorem 6.7.1 in Conn et al. (2000))**.** *Suppose that there exists a constant $\zeta \geq 1$ such that*

$$\frac{1}{\zeta} \leq \sigma_{\min}(\mathbf{A}_t) \leq \sigma_{\max}(\mathbf{A}_t) \leq \zeta \qquad \forall t, \tag{21}$$

then Definition 1 holds.

Proposition 3 (Uniform equivalence)**.** *Suppose* $\|\mathbf{g}_t\|^2 \leq L_H^2$ *for all* $\mathbf{w}_t \in \mathbb{R}^d$, $t = 1, 2, \ldots$ *Then there always exists* $\epsilon > 0$ *such that the proposed preconditioning matrices* $\mathbf{A}_{rms,t}$ *(Eq. 3) are uniformly equivalent, i.e. Def. 1 holds. The same holds for the diagonal variant.*

Proof. The basic building block of our ellipsoid matrix consists of the current and past stochastic gradients $\mathbf{G}_t := [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_t]$.

We consider $\mathbf{A}_{rms}$ which is built up as follows[9]

$$\mathbf{A}_{rms,t} := \left((1-\beta)\mathbf{G} \underbrace{\operatorname{diag}(\beta^t, \beta^{t-1}, \ldots, \beta^0)}_{:=\mathbf{D}} \mathbf{G}^\intercal \right) + \epsilon \mathbf{I}. \tag{22}$$

From the construction of $\mathbf{A}_{rms,t}$ it directly follows that for any unit length vector $\mathbf{u} \in \mathbb{R}^d \setminus \{0\}, \|\mathbf{u}\|_2 = 1$ we have

$$\begin{aligned} &\mathbf{u}^\intercal \left((1-\beta)\mathbf{G}\mathbf{D}\mathbf{G}^\intercal + \epsilon \mathbf{I} \right) \mathbf{u} \\ =&(1-\beta)\mathbf{u}^\intercal \mathbf{G}\mathbf{D}^{1/2}(\mathbf{D}^{1/2})^\intercal \mathbf{G}^\intercal \mathbf{u} + \epsilon \|\mathbf{u}\|_2^2 \\ =&(1-\beta)\left((\mathbf{D}^{1/2})^\intercal \mathbf{G}^\intercal \mathbf{u} \right)^\intercal \left((\mathbf{D}^{1/2})^\intercal \mathbf{G}^\intercal \mathbf{u} \right) + \epsilon \|\mathbf{u}\|_2^2 \\ \geq&\epsilon > 0, \end{aligned} \tag{23}$$

which proves the lower bound for $\zeta = 1/\epsilon$. Now, let us consider the upper end of the spectrum of $\mathbf{A}_{rms,t}$. Towards this end, recall the geometric series expansion

$$\sum_{i=0}^{t} \beta^{t-i} = \sum_{i=0}^{t} \beta^i = \frac{1-\beta^{t+1}}{1-\beta} \tag{24}$$

and the fact that $\mathbf{G}\mathbf{G}^\top$ is a sum of exponentially weighted rank-one positive semi-definite matrices of the form $\mathbf{g}_i \mathbf{g}_i^\intercal$. Thus

$$\lambda_{max}(\mathbf{g}_i \mathbf{g}_i^\intercal) = \operatorname{Tr}(\mathbf{g}_i \mathbf{g}_i^\intercal) = \|\nabla \mathbf{g}_i\|^2 \leq L_H^2,$$

where the latter inequality holds per assumption for any sample size $|S|$. Combining these facts we get that

$$\begin{aligned} &\mathbf{u}^\intercal \left((1-\beta)\mathbf{G}\mathbf{D}\mathbf{G}^\intercal + \epsilon \mathbf{I} \right) \mathbf{u} \\ =&(1-\beta)\mathbf{u}^\intercal \mathbf{G}\mathbf{D}\mathbf{G}^\intercal \mathbf{u} + \epsilon \|\mathbf{u}\|_2^2 \\ =&(1-\beta)\sum_{i=0}^{t} \beta^{t-1} \mathbf{u}^\intercal \mathbf{g}_i \mathbf{g}_i^\intercal \mathbf{u} + \epsilon \|\mathbf{u}\|_2^2 \\ \leq&(1-\beta)\sum_{i=0}^{t} \beta^{t-i} L_H^2 \|\mathbf{u}\|_2^2 + \epsilon \|\mathbf{u}\|_2^2 \\ =&(1-\beta^{t+1})L_H^2 + \epsilon. \end{aligned} \tag{25}$$

As a result we have that

$$\epsilon \leq \lambda_{min}\left(\mathbf{A}_{rms,t}\right) \leq \lambda_{max}\left(\mathbf{A}_{rms,t}\right) \leq \left(1-\beta^{t+1}\right) L_H^2 + \epsilon \tag{26}$$

[9]This is a generalization of the diagonal variant proposed by Tieleman & Hinton (2012), which precondition the gradient step by an elementwise division with the square-root of the following estimate $g_t = (1-\beta)g_{t-1} + \beta \nabla \mathcal{L}(\mathbf{w}_t)^2$.

Finally, to achieve uniform equivalence we need the r.h.s. of (26) to be bounded by $1/\epsilon$. This gives rise to a quadratic equation in ϵ, namely

$$\epsilon^2 + \left(1 - \beta^{t+1}\right) L_H^2 \epsilon - 1 \leq 0 \tag{27}$$

which holds for any t and any $\beta \in (0, 1)$ as long as

$$0 \leq \epsilon \leq \frac{1}{2}(\sqrt{L_H^4 + 4} - L_H^2). \tag{28}$$

Such an ϵ always exists but one needs to choose smaller and smaller values as the upper bound on the gradient norm grows. For example, the usual value $\epsilon = 10^{-8}$ is valid for all $L_H^2 < 9.9 \cdot 10^7$. All of the above arguments naturally extend to the diagonal preconditioner $\mathrm{diag}(\mathbf{A}_{rms})$.

□

D Diagonal Dominance in Neural Networks

D.1 Proof of Proposition 1

Proposition 4 (Proposition 1 restated)**.** *For random Gaussian Wigner matrix* $\mathbf{W}$ *formed as*

$$\mathbf{W}_{i,j} = \mathbf{W}_{j,i} := \begin{cases} \sim \mathcal{N}(0, \sigma_1), & i < j \\ \sim \mathcal{N}(0, \sigma_2), & i = j, \end{cases} \tag{29}$$

where $\sim$ *stands for i.i.d. draws (Wigner, 1993), the expected share of diagonal mass* $\delta_{\mathbf{W}}$ *amounts to*

$$\mathbb{E}\left[\delta_{\mathbf{W}}\right] = \frac{1}{1 + (d-1)\frac{\sigma_2}{\sigma_1}}. \tag{30}$$

Proof.

$$\begin{aligned} \mathbb{E}\left[\delta_{\mathbf{W}}\right] &= \mathbb{E}\left[\frac{\sum_i |\mathbf{W}_{i,i}|}{\sum_i \sum_j |\mathbf{W}_{i,j}|}\right] = \mathbb{E}\left[\frac{1}{1 + \frac{\sum_i \sum_{j \neq i} |\mathbf{W}_{i,j}|}{\sum_i |\mathbf{W}_{i,i}|}}\right] \\ &= \frac{1}{1 + \frac{\sum_i \sum_{j \neq i} \mathbb{E}[|\mathbf{W}_{i,j}|]}{\sum_i \mathbb{E}[|\mathbf{W}_{i,i}|]}} = \frac{1}{1 + \frac{(d^2 - d)\sigma_2 \sqrt{2/\pi}}{d \sigma_1 \sqrt{2/\pi}}} \\ &= \frac{1}{1 + (d-1)\frac{\sigma_2}{\sigma_1}}, \end{aligned} \tag{31}$$

which simplifies to $\frac{1}{d}$ if the diagonal and off-diagonal elements come from the same Gaussian distribution ($\sigma_1 = \sigma_2$). □

For the sake of simplicity we only consider Gaussian Wigner matrices but the above argument naturally extends to any distribution with positive expected absolute values, i.e. we only exclude the Dirac delta function as probability density.

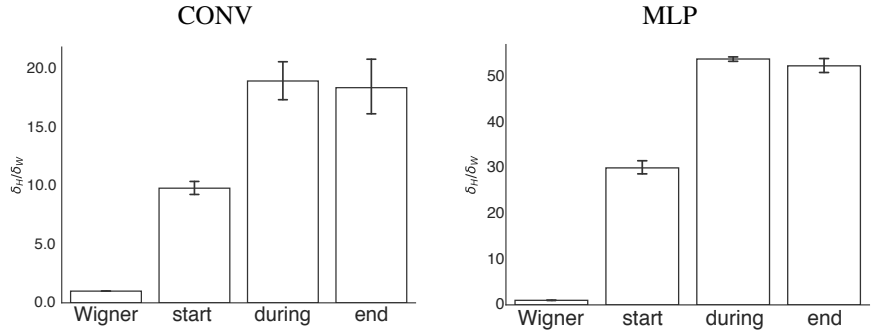

Figure 6: Share of diagonal mass of the Hessian $\delta_{\mathbf{H}}$ relative to $\delta_{\mathbf{W}}$ of the corresponding Wigner matrix at random initialization, after 50% iterations and at the end of training with RMSprop on MNIST. Average and 95% confidence interval over 10 runs. See Figure 2 for CIFAR-10 results.

Appendix B: Background on second-order optimization

A NEWTON'S METHOD

The canonical second-order method is Newton's methods. This algorithm uses the inverse Hessian as a scaling matrix and thus has updates of the form

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \nabla^2 \mathcal{L}(\mathbf{w}_t)^{-1} \nabla \mathcal{L}(\mathbf{w}_t), \tag{32}$$

which is equivalent to optimizing the local quadratic model

$$m_N(\mathbf{w}_t) := \mathcal{L}(\mathbf{w}_t) + \nabla \mathcal{L}(\mathbf{w}_t)^\intercal \mathbf{s} + \frac{1}{2} \mathbf{s}^\intercal \nabla^2 \mathcal{L}(\mathbf{w}_t) \mathbf{s} \tag{33}$$

to *first-order stationarity*. Using curvature information to rescale the steepest descent direction gives Newton's method the useful property of being linearly scale invariant. This gives rise to a *problem independent* local convergence rate that is super-linear and even quadratic in the case of Lipschitz continuous Hessians (see Nocedal & Wright (2006) Theorem 3.5), whereas gradient descent at best achieves linear local convergence (Nesterov, 2013).

However, there are certain drawbacks associated with applying classical Newton's method. First of all, the Hessian matrix may be singular and thus not invertible. Secondly, even if it is invertible the local quadratic model (Eq. 33) that is minimized in each NM iteration may simply be an inadequate approximation of the true objective. As a result, the Newton step is not necessarily a descent step. It may hence approximate arbitrary critical points (including local maxima) or even diverge. Finally, the cost of forming and inverting the Hessian sum up to $O(nd^2 + d^3)$ and are thus prohibitively high for applications in large dimensional problems.

B TRUST REGION METHODS

B.1 OUTER ITERATIONS

Trust region methods are among the most principled approaches to overcome the above mentioned issues. These methods also construct a quadratic model m_t but constrain the subproblem in such a way that the stepsize is restricted to stay within a certain radius Δ_t within which the model is trusted to be sufficiently adequate

$$\min_{s \in \mathbb{R}^d} \; m_t(\mathbf{s}) = \mathcal{L}(\mathbf{w}_t) + \nabla \mathcal{L}(\mathbf{w}_t)^\intercal \mathbf{s} + \frac{1}{2} \mathbf{s}^\intercal \nabla^2 \mathcal{L}(\mathbf{w}_t) \mathbf{s}, \quad s.t. \; \|\mathbf{s}\| \leq \Delta_t. \tag{34}$$

Hence, contrary to line-search methods this approach finds the step $\mathbf{s}_t$ and its length $\|\mathbf{s}_t\|$ *simultaneously* by optimizing (34). Subsequently the actual decrease $\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_t + \mathbf{s}_t)$ is compared to the predicted decrease $m_t(0) - m_t(\mathbf{s}_t)$ and the step is only accepted if the ratio $\rho := \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_t + \mathbf{s}_t)/(m_t(0) - m_t(\mathbf{s}_t))$ exceeds some predefined success threshold $\eta_1 > 0$. Furthermore, the trust region radius is decreased whenever ρ falls below η_1 and it is increased whenever ρ exceeds the "very successful" threshold $\eta_2 0$. Thereby, the algorithm adaptively measures the accuracy of the second-order Taylor model – which may change drastically over the parameter space depending on the behaviour of the higher-order derivatives[10] – and adapts the effective length along which the model is trusted accordingly. See Conn et al. (2000) for more details.

As a consequence, the plain Newton step $\mathbf{s}_{N,t} = -\left(\nabla^2 \mathcal{L}_t\right)^{-1} \nabla \mathcal{L}_t$ is only taken if it lies within the trust region radius and yields a certain amount of decrease in the objective value. Since many functions look somehow quadratic close to a minimizer the radius can be shown to grow asymptotically under mild assumptions such that eventually full Newton steps are taken in every iteration which retains the local quadratic convergence rate (Conn et al., 2000).

[10] Note that the second-order Taylor models assume constant curvature.

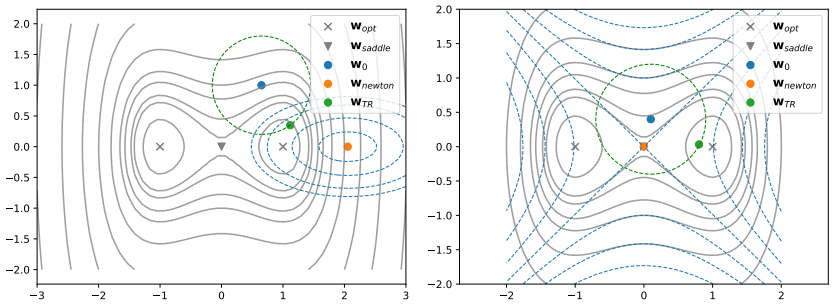

Figure 7: Level sets of the non-convex, coercive objective function $f(\mathbf{w}) = 0.5\mathbf{w}_0^2 + 0.25\mathbf{w}_1^4 - 0.5\mathbf{w}_1^2$. Newton's Method makes a local quadratic model (blue dashed lines) and steps to its *critical point*. It may be thus be ascending (left) or attracted by a saddle point (right). TR methods relieve this issue by stepping to the *minimizer* of that model *within* a certain region (green dashed line).

B.2 Subproblem solver

Interestingly, there is no need to optimize Eq. (34) to global optimality to retain the remarkable global convergence properties of TR algorithms. Instead, it suffices to do better than the Cauchy- and Eigenpoint[11] simultaneously. One way to ensure this is to minimize $m_t(\mathbf{s})$ in nested Krylov subspaces. These subspaces naturally include the gradient direction as well as increasingly accurate estimates of the leading eigendirection

$$\text{span}\{\mathbf{g}_t, \mathbf{B}_t\mathbf{g}_t, \mathbf{B}_t^2\mathbf{g}_t, \dots, \mathbf{B}_t^j\mathbf{g}_t\} \tag{35}$$

until (for example) the stopping criterion

$$\|\nabla m_t(\mathbf{s}_j)\| \leq \|\nabla \mathcal{L}(\mathbf{w}_t)\| \min\{\kappa_K, \|\nabla \mathcal{L}(\mathbf{w}_t)\|^\theta\}, \quad \kappa_K < 1, \theta \geq 0 \tag{36}$$

is met, which requires increased accuracy as the underlying trust region algorithm approaches criticality. Conjugate gradients and Lanczos method are two iterative routines that implicitly build up a conjugate and orthogonal basis for such a Krylov space respectively and they converge *linearly* on quadratic objectives with a square-root dependency on the condition number of the Hessian (Conn et al., 2000). We here employ the preconditionied Steihaug-Toint CG method (Steihaug, 1983) in order to cope with possible boundary solutions of (34) but similar techniques exist for the Lanczos solver as well for which we also provide code. As preconditioning matrix for CG we use the same matrix as for the ellipsoidal constraint.

C Damped (Gauss-)Newton methods

An alternative approach to actively constraining the region within which the model is trusted is to instead penalize the step norm in each iteration in a Lagrangian manner. This is done by so-called damped Newton methods that add a $\lambda > 0$ multiple of the identity matrix to the second-order term in the model, which leads to the update step

$$\begin{aligned} \min_{s \in \mathbb{R}^d} m_t(\mathbf{s}) &= \mathcal{L}(\mathbf{w}_t) + \nabla \mathcal{L}(\mathbf{w}_t)^\intercal \mathbf{s} + \frac{1}{2}\mathbf{s}^\intercal(\nabla^2 \mathcal{L}(\mathbf{w}_t) + \lambda \mathbf{I})\mathbf{s} \\ &= \mathcal{L}(\mathbf{w}_t) + \nabla \mathcal{L}(\mathbf{w}_t)^\intercal \mathbf{s} + \frac{1}{2}\mathbf{s}^\intercal \nabla^2 \mathcal{L}(\mathbf{w}_t)\mathbf{s} + \lambda \|\mathbf{s}\|^2. \end{aligned} \tag{37}$$

This can also be solved hessian-free by conjugate gradients (or other Krylov subspace methods). The penalty parameter λ is acting inversely to the trust region radius Δ and it is often updated accordingly. Such algorithms are commonly known as Levenberg-Marquardt algorithms and they were originally tailored towards solving non-linear least squares problems (Nocedal & Wright, 2006) but they have been proposed for neural network training already early on (Hagan & Menhaj, 1994).

[11] which are the model minimizers along the gradient and the eigendirection associated with its smallest eigenvalue, respectively.

Many algorithms in the existing literature replace the use of $\nabla^2\mathcal{L}(\mathbf{w}_t)$ in (37) with the Generalized Gauss Newton matrix (Martens, 2010; Chapelle & Erhan, 2011) or an approximation of the latter (Martens & Grosse, 2015). This matrix constitutes the first part of the well-known Gauss-Newton decomposition

$$\nabla^2\mathcal{L}(\cdot) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\ell''(f_i(\cdot))\nabla f_i(\cdot)\nabla f_i(\cdot)^\intercal}_{:=\mathbf{A}_{GGN}} + \frac{1}{n}\sum_{i=1}^{n}\ell'(f_i(\cdot))\nabla^2 f_i(\cdot), \tag{38}$$

where l' and l'' are the first and second derivative of $l : \mathbb{R}^{out} \to \mathbb{R}^+$ assuming that $out = 1$ (binary classification and regression task) for simplicity here.

It is interesting to note that the GGN matrix $\mathbf{A}_{GGN}$ of neural networks is equivalent to the Fisher matrix used in natural gradient descent (Amari, 1998) in many cases like linear activation function and squared error as well as sigmoid and cross-entropy or softmax and negative log-likelihood for which the extended Gauss-Newton is defined (Pascanu & Bengio, 2013). As can be seen in (38) the matrix $\mathbf{A}_{GGN}$ is positive semidefinite (and low rank if $n < d$). As a result, there exist no second-order convergence guarantees for such methods on general non-convex problems. On the other end of the spectrum, the GGN also drops possibly positive terms from the Hessian (see 38). Hence it is not guaranteed to be an upper bound on the latter in the PSD sense. Essentially, GGN approximations assume that the network is piece-wise linear and thus the GGN and Hessian matrices only coincide in the case of linear and ReLU activations or non-curved loss functions. For any other activation the GGN matrix may approximate the Hessian only asymptotically and if the $\ell'(f_i(\cdot))$ terms in 38 go to zero for all $i \in \{1, \dots, n\}$. In non-linear least squares such problems are called zero-residual problems and GN methods can be shown to have quadratic local convergence there. In any other case the convergence rate does not exceed the linear local convergence bound of gradient descent. In practice however there are cases where deep neural nets do show negative curvature in the neighborhood of a minimizer (Bottou et al., 2018).Finally, Dauphin et al. (2014) propose the use of the absolute Hessian instead of the GGN matrix in a framework similar to 37. This method has been termed *saddle-free Newton* even though its manifold of attraction to a given saddle is non-empty[12].

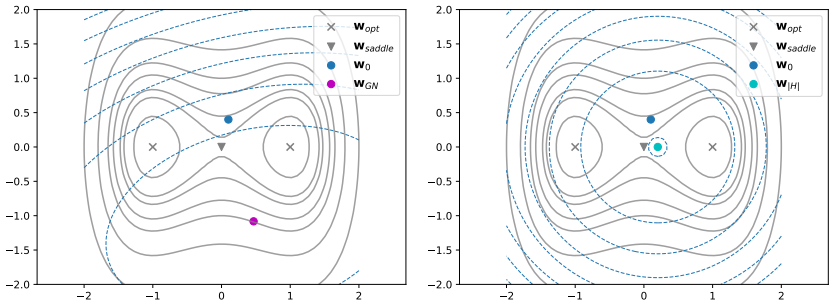

Figure 8: Both, the GGN method and saddle-free Newton method make a positive definite quadratic model around the current iterate and thereby overcome the abstractedness of pure Newton towards the saddle (compare Figure 7). However, (i) none of these methods can escape the saddle once they are in the gradient manifold of attraction and (ii) as reported in Mizutani & Dreyfus (2008) the GN matrix can be significantly less well conditioned than the absolute Hessian (here $\kappa_{GN} = 49'487'554$ and $\kappa_{|H|} = 1.03$ so we had to add a damping factor of $\lambda = 0.1$ to make the GN step fit the plot.

C.1 Comparison to trust region

Contrary to TR methods, the Levenberg-Marquardt methods never take plain Newton steps since the regularization is always on ($\lambda > 0$). Furthermore, if a positive-definite Hessian approximation like the Generalized Gauss Newton matrix is used, this algorithm is not capable of exploiting negative curvature and there are cases in neural network training where the Hessian is much better conditioned than the Gauss-Newton matrix (Mizutani & Dreyfus, 2008) (also see Figure 8). While some scholars

[12]It is the same as that for GD, which renders the method unable to escape e.g. when initialized right on a saddle point. To be fair, the manifold of attraction for GD constitutes a measure zero set (Lee et al., 2016).

believe that positive-definiteness is a desirable feature (Martens, 2010; Chapelle & Erhan, 2011), we want to point out that following negative curvature directions is necessarily needed to escape saddle points and it can also be meaningful to follow directions of negative eigenvalue λ outside a saddle since they guarantee $\mathcal{O}(|\lambda|^3)$ progress, whereas a gradient descent step yields at least $\|\nabla f(\mathbf{w})\|^2$ progress (both under certain stepsize conditions) and one cannot conclude a-priori which one is better in general (Curtis & Robinson, 2017; Alain et al., 2018). Despite these theoretical considerations, many methods based on GGN matrices have been applied to neural network training (see Martens (2014) and references therein) and particularly the hessian-free implementations of (Martens, 2010; Chapelle & Erhan, 2011) can be implemented very cheaply (Schraudolph, 2002).

D USING HESSIAN INFORMATION IN NEURAL NETWORKS

While many theoretical arguments suggest the superiority of regularized Newton methods over gradient based algorithms, several practical considerations cast doubt on this theoretical superiority when it comes to neural network training. Answers to the following questions are particularly unclear: Are saddles even an issue in deep learning? Is superlinear local convergence a desirable feature in machine learning applications (test error)? Are second-order methods more "vulnerable" to sub-sampling noise? Do worst-case iteration complexities even matter in real-world settings? As a result, the value of Hessian information in neural network training is somewhat unclear a-priori and so far a conclusive empirical study is still missing.

Our empirical findings indicate that the net value of Hessian information for neural network training is indeed somewhat limited for mainly three reasons: 1) second-order methods rarely yield better limit points, which suggests that saddles and spurious local minima are not a major obstacle; 2) gradient methods can indeed run on smaller batch sizes which is beneficial in terms of epoch and when memory is limited; 3) The per-iteration time complexity is noticeably lower for first-order methods. In summary, these observations suggest that advances in hardware and distributed second-order algorithms (e.g., Osawa et al. (2018); Dünner et al. (2018)) will be needed before Newton-type methods can replace (stochastic) gradient methods in deep learning.

Appendix C: Experiment details

A EXPERIMENTAL RESULTS

A.1 ELLIPSOIDAL TRUST REGION VS. FIRST-ORDER OPTIMIZER

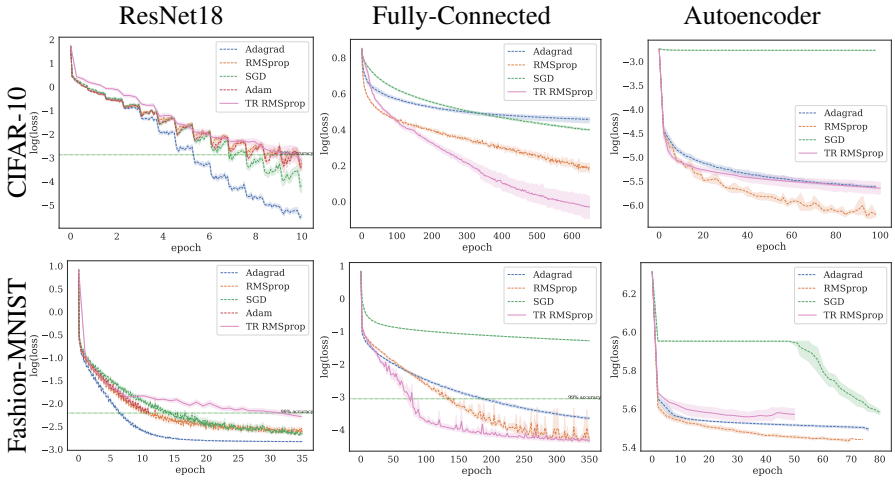

Figure 9: Experiment comparing TR and gradient methods in terms of epochs. Average log loss as well as 95% confidence interval shown.

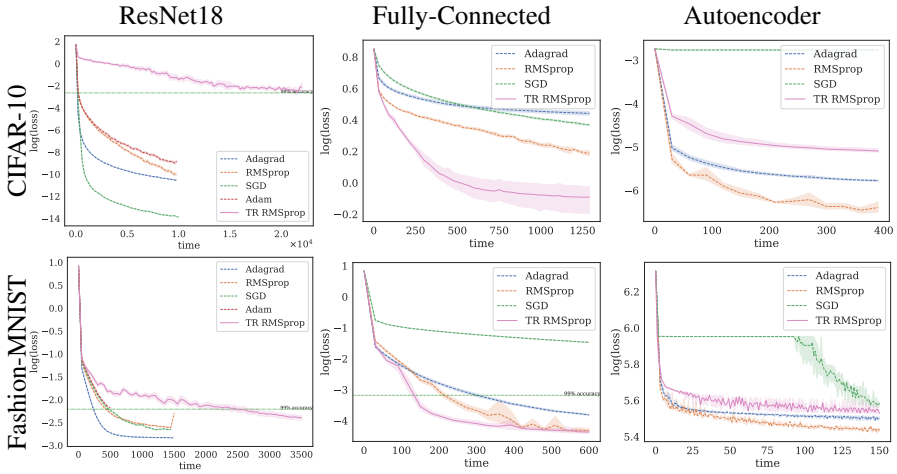

Figure 10: Experiment comparing TR and gradient methods in terms of wall-clock time. Average log loss as well as 95% confidence interval shown. The advantage of extremely low-iteration costs of first-order methods is particularly notable in the ResNet18 architecture due to the large network size.

B Further Experiment Details

B.1 Default parameters, architectures and datasets

Parameters Table 1 reports the default parameters we consider. Only for the larger ResNet18 on CIFAR-10, we adapted the batch size to 128 due to memory constraints.

	$\lvert\mathcal{S}_0\rvert$	Δ_0	Δ_{max}	η_1	η_2	γ_1	γ_2	κ_K (krylov tol.)
TR_{uni}	512	10^{-4}	10	10^{-4}	0.95	1.1	1.5	0.1
TR_{ada}	512	10^{-4}	10	10^{-4}	0.95	1.1	1.5	0.1
TR_{rms}	512	10^{-4}	10	10^{-4}	0.95	1.1	1.75	0.1

Table 1: Default parameters

Datasets We use two real-world datasets for image classification, namely CIFAR-10 and Fashion-MNIST[13]. While Fashion-MNIST consists of greyscale 28×28 images, CIFAR-10 are colored images of size 32×32. Both datasets have a fixed training-test split consisting of 60,000 and 10,000 images, respectively.

Network architectures The MLP architectures are simple. For MNIST and Fashion-MNIST we use a $784 - 128 - 10$ network with tanh activations and a cross entropy loss. The networks has $101'770$ parameters. For the CIFAR-10 MLP we use a $3072 - 128 - 128 - 10$ architecture also with tanh activations and cross entropy loss. This network has $410'880$ parameters.

The Fashion-MNIST autoencoder has the same architecture as the one used in Hinton & Salakhutdinov (2006); Xu et al. (2017a); Martens (2010); Martens & Grosse (2015). The encoder structure is $784 - 1000 - 500 - 250 - 30$ and the decoder is mirrored. Sigmoid activations are used in all but the central layer. The reconstructed images are fed pixelwise into a binary cross entropy loss. The network has a total of $2'833'000$ parameters. The CIFAR-10 autoencoder is taken from the implementation of https://github.com/jellycsc/PyTorch-CIFAR-10-autoencoder.

For the ResNet18, we used the implementation from torchvision for CIFAR-10 as well as a modification of it for Fashion-MNIST that adapts the first convolution to account for the single input channel.

[13]Both datasets were accessed from https://www.tensorflow.org/api_docs/python/tf/keras/datasets

In all of our experiments each method was run on one Tesla P100 GPU using the PyTorch (Paszke et al., 2017) library.

B.2 RECONSTRUCTED IMAGES FROM AUTOENCODERS

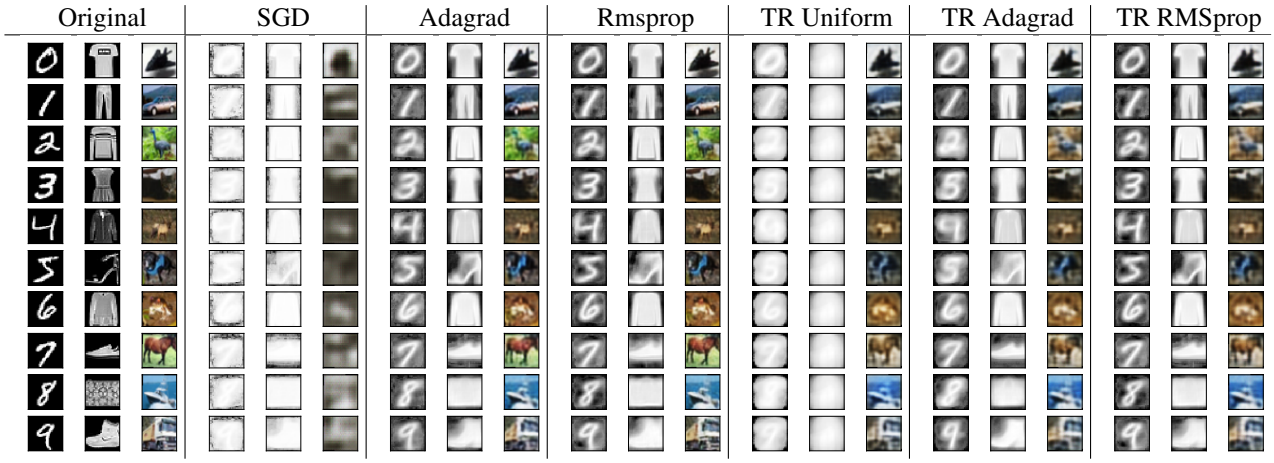

Figure 11: Original and reconstructed MNIST digits (left), Fashion-MNIST items (middle), and CIFAR-10 classes (right) for different optimization methods after convergence.