Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization

Jonas Kohler* ETH Zurich

Hadi Daneshmand* ETH Zurich

Aurelien Lucchi ETH Zurich

Thomas Hofmann ETH Zurich

Ming Zhou Universität Rostock

Klaus Neymeyr Universität Rostock

Abstract

Normalization techniques such as Batch Normalization have been applied successfully for training deep neural networks. Yet, despite its apparent empirical benefits, the reasons behind the success of Batch Normalization are mostly hypothetical. We here aim to provide a more thorough theoretical understanding from a classical optimization perspective. Our main contribution towards this goal is the identification of various problem instances in the realm of machine learning where Batch Normalization can provably accelerate optimization. We argue that this acceleration is due to the fact that Batch Normalization splits the optimization task into optimizing length and direction of the parameters separately. This allows gradient-based methods to leverage a favourable global structure in the loss landscape that we prove to exist in Learning Halfspace problems and neural network training with Gaussian inputs. We thereby turn Batch Normalization from an effective practical heuristic into a provably converging algorithm for these settings. Furthermore, we substantiate our analysis with empirical evidence that suggests the validity of our theoretical results in a broader context.

1 INTRODUCTION

One of the most important recent innovations for optimizing deep neural networks is Batch Normalization (Bn) (Ioffe and Szegedy, 2015). This technique has been proven to successfully stabilize and accelerate training of deep neural networks and is thus by now standard in many state-of-the art architectures such as ResNets (He et al., 2016) and the latest Inception Nets (Szegedy et al., 2017). The success of Batch Normalization has promoted its key idea that normalizing the inner layers of a neural network stabilizes training which recently led to the development of many such normalization methods such as (Arpit et al., 2016; Klambauer et al., 2017; Salimans and Kingma, 2016) and (Ba et al., 2016) to name just a few.

Yet, despite the ever more important role of Batch Normalization for training deep neural networks, the Machine Learning community is mostly relying on empirical evidence and thus lacking a thorough theoretical understanding that can explain such success. Indeed – to the best of our knowledge – there exists no theoretical result which provably shows faster convergence rates for this technique on any problem instance. So far, there only exists competing hypotheses that we briefly summarize below .

1.1 Related work

Internal Covariate Shift The most widespread idea is that Batch Normalization accelerates training by reducing the so-called internal covariate shift, defined as the change in the distribution of layer inputs while the conditional distribution of outputs is unchanged. This change can be significant especially for deep neural networks where the successive composition of layers drives the activation distribution away from the initial input distribution. Ioffe and Szegedy (2015) argue that Batch Normalization reduces the internal covariate shift by employing a normalization technique that enforces the input distribution of each activation layer to be whitened - i.e. enforced to have zero means and unit variances - and decorrelated . Yet, as pointed out by Lipton and Steinhardt (2018), the covariate shift phenomenon itself is not rigorously shown to be the reason behind the performance of Batch Normalization. Furthermore, a recent empirical study published by (Santurkar et al., 2018) provides strong evidence sup-

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

*The first two authors have contributed equally.

porting the hypothesis that the performance gain of Batch Normilization is not explained by the reduction of internal covariate shift.

Smoothing of objective function Recently, Santurkar et al. (2018) argue that under certain assumptions a normalization layer simplifies optimization by smoothing the loss landscape of the optimization problem of the preceding layer. Yet, we note that this effect may - at best - only improve the constant factor of the convergence rate of Gradient Descent and not the rate itself (e.g. from sub-linear to linear). Furthermore, the analysis treats only the largest eigenvalue and thus one direction in the landscape (at any given point) and keeps the (usually trainable) BN parameters fixed to zero-mean and unit variance. For a thorough conclusion about the overall landscape, a look at the entire eigenspectrum (including negative and zero eigenvalues) would be needed. Yet, this is particularly hard to do as soon as one allows for learnable mean and variance parameters since the effect of their interplay on the distribution of eigenvalues is highly non-trivial.

Length-direction decoupling Finally, a different perspective was brought up by another normalization technique termed Weight Normalization (WN) (Salimans and Kingma, 2016). This technique performs a very simple normalization that is independent of any data statistics with the goal of decoupling the length of the weight vector from its direction. The optimization of the training objective is then performed by training the two parts separately. As discussed in Section 2, BN and WN differ in how the weights are normalized but share the above mentioned decoupling effect. Interestingly, weight normalization has been shown empirically to benefit from similar acceleration properties as Batch Normalization (Gitman and Ginsburg, 2017; Salimans and Kingma, 2016). This raises the obvious question whether the empirical success of training with Batch Normalization can (at least partially) be attributed to its length-direction decoupling aspect.

1.2 Contribution and organization

We contribute to a better theoretical understanding of Batch Normalization by analyzing it from an optimization perspective. In this regard, we particularly address the following question:

Can we find a setting in which Batch Normalization provably accelerates optimization with Gradient Descent and does the length-direction decoupling play a role in this phenomenon?

We answer both questions affirmatively. In particular, we show that the specific variance transformation of BN decouples the length and directional components of the weight vectors in such a way that allows local search methods to exploit certain global properties of the optimization landscape (present in the directional component of the optimal weight vector). Using this fact and endowing the optimization method with an adaptive stepsize scheme, we obtain an *exponential* (or as more commonly termed *linear*) convergence rate for Batch Norm Gradient Descent on the (possibly) *non-convex* problem of Learning Halfspaces with Gaussian inputs (Section 4), which is a prominent problem in machine learning Erdogdu et al. (2016). We thereby turn BN from an effective practical heuristic into a provably converging algorithm. Additionally we show that the length-direction decoupling can be considered as a non-linear reparametrization of the weight space, which may be beneficial for even simple convex optimization tasks such as logistic regressions. Interestingly, non-linear weightspace transformations have received little to no attention within the optimization community (see (Mikhalevich et al., 1988) for an exception).

Finally, in Section 5 we analyze the effect of BN for training a multilayer neural network (MLP) and prove – again under a similar Gaussianity assumption – that BN acts in such a way that the cross dependencies between layers are reduced and thus the curvature structure of the network is simplified. Again, this is due to a certain global property in the directional part of the optimization landscape, which BN can exploit via the length-direction decoupling. As a result, gradient-based optimization in reparametrized coordinates (and with an adaptive stepsize policy) can enjoy a linear convergence rate on each individual unit. We substantiate both findings with experimental results on real world datasets that confirm the validity of our analysis outside the setting of our theoretical assumptions that cannot be certified to always hold in practice.

2 BACKGROUND

2.1 Assumptions on data distribution

Suppose that $\mathbf{x} \in \mathbb{R}^d$ is a random input vector and $y \in \{\pm 1\}$ is the corresponding output variable. Throughout this paper we recurrently use the following statistics

$$\mathbf{u} := \mathbf{E}\left[-y\mathbf{x}\right], \quad \mathbf{S} := \mathbf{E}\left[\mathbf{x}\mathbf{x}^\top\right] \tag{1}$$

and make the following (weak) assumption.

Assumption 1. *[Weak assumption on data distribution] We assume that* $\mathbf{E}[\mathbf{x}] = 0$. *We further assume that the spectrum of the matrix* $\mathbf{S}$ *is bounded as*

$$0 < \mu := \lambda_{\min}(\mathbf{S}), L := \lambda_{\max}(\mathbf{S}) < \infty. \tag{2}$$

As a result, $\mathbf{S}$ *is the symmetric positive definite covariance matrix of* $\mathbf{x}$*.*

The part of our analysis presented in Section 4 and 5 relies on a stronger assumption on the data distribution. In this regard we consider the combined random variable

$$\mathbf{z} := -y\mathbf{x}, \tag{3}$$

whose mean vector and covariance matrix are $\mathbf{u}$ and $\mathbf{S}$ as defined above in Eq. (1).

Assumption 2. *[Normality assumption] We assume that* $\mathbf{z}$ *is a multivariate normal random variable distributed with mean* $\mathbf{E}\left[\mathbf{z}\right] = \mathbf{E}\left[-y\mathbf{x}\right] = \mathbf{u}$ *and second-moment* $\mathbf{E}\left[\mathbf{z}\mathbf{z}^\top\right] - \mathbf{E}[\mathbf{z}]\mathbf{E}[\mathbf{z}]^\top = \mathbf{E}\left[\mathbf{x}\mathbf{x}^\top\right] - \mathbf{u}\mathbf{u}^\top = \mathbf{S} - \mathbf{u}\mathbf{u}^\top$*.*

In the absence of further knowledge, assuming Gaussian data is plausible from an information-theoretic point of view since the Gaussian distribution maximizes the entropy over the set of all absolutely continuous distributions with fixed first and second moment (Dowson and Wragg, 1973). Thus, many recent studies on neural networks make this assumption on $\mathbf{x}$ (see e.g. (Brutzkus and Globerson, 2017; Du and Lee, 2018)). Here we assume Gaussianity on $y\mathbf{x}$ instead which is even less restrictive in some cases[1].

2.2 Batch normalization as a reparameterization

In neural networks a BN layer normalizes the input of each unit of the following layer. This is done on the basis of data statistics in a training batch, but for the sake of analyzablility we will work directly with population statistics. In particular, the output f of a specific unit, which projects an input $\mathbf{x}$ to its weight vector $\mathbf{w}$ and applies a sufficiently smooth activation function $\varphi : \mathbb{R} \to \mathbb{R}$ as follows

$$f(\mathbf{w}) = \mathbf{E}_\mathbf{x}\left[\varphi\left(\mathbf{x}^\top\mathbf{w}\right)\right] \tag{4}$$

is normalized on the pre-activation level. That is, the input-output mapping of this unit becomes

$$f_{BN}(\mathbf{w}, g, \gamma) = \mathbf{E}_\mathbf{x}\left[\varphi\left(BN(\mathbf{x}^\top\mathbf{w})\right)\right]. \tag{5}$$

As stated (in finite-sum terms) in Algorithm 1 of (Ioffe and Szegedy, 2015) the normalization operation amounts to computing

$$BN(\mathbf{x}^\top\mathbf{w}) = g\frac{\mathbf{x}^\top\mathbf{w} - \mathbf{E}_\mathbf{x}[\mathbf{x}^\top\mathbf{w}]}{\mathbf{var}_\mathbf{x}[\mathbf{x}^\top\mathbf{w}]^{1/2}} + \gamma, \tag{6}$$

where $g \in \mathbb{R}$ and $\gamma \in \mathbb{R}$ are (trainable) mean and variance adjustment parameters. In the following, we assume that $\mathbf{x}$ is zero mean (Assumption 1) and omit γ.[2] Then the variance can be written as follows

$$\begin{aligned}\mathbf{var}_\mathbf{x}[\mathbf{x}^\top\mathbf{w}] = \mathbf{E}_\mathbf{x}\left[(\mathbf{x}^\top\mathbf{w})^2\right] &= \mathbf{E}_\mathbf{x}\left[(\mathbf{w}^\top\mathbf{x})(\mathbf{x}^\top\mathbf{w})\right] \\ &= \mathbf{w}^\top\mathbf{E}_\mathbf{x}\left[\mathbf{x}\mathbf{x}^\top\right]\mathbf{w} = \mathbf{w}^\top\mathbf{S}\mathbf{w}\end{aligned} \tag{7}$$

and replacing this expression into the batch normalized output of Eq.(5) yields

$$f_{BN}(\mathbf{w}, g) = \mathbf{E}_\mathbf{x}\left[\varphi(g\frac{\mathbf{x}^\top\mathbf{w}}{(\mathbf{w}^\top\mathbf{S}\mathbf{w})^{1/2}})\right]. \tag{8}$$

In order to keep concise notations, we will often use the induced norm of the positive definite matrix $\mathbf{S}$ defined as $\|\mathbf{w}\|_\mathbf{S} := \left(\mathbf{w}^\top\mathbf{S}\mathbf{w}\right)^{1/2}$. Comparing Eq. (4) and (8) it becomes apparent that BN can be considered as a reparameterization of the weight space. We thus define

$$\tilde{\mathbf{w}} := g\frac{\mathbf{w}}{\|\mathbf{w}\|_\mathbf{S}} \tag{9}$$

and note that $\mathbf{w}$ accounts for the direction and g for the length of $\tilde{\mathbf{w}}$. As a result, the batch normalized output can then be written as $f_{BN}(\mathbf{w}, g) = \mathbf{E}_\mathbf{x}\left[\varphi(\mathbf{x}^\top\tilde{\mathbf{w}})\right]$. Note that Weight Normalization (WN) is another instance of the above reparametrization, where the covariance matrix $\mathbf{S}$ is replaced by the identity matrix $\mathbf{I}$ (Salimans and Kingma, 2016). In both cases, the objective becomes invariant to linear scaling of $\mathbf{w}$. From a geometry perspective, the directional part of WN can be understood as performing optimization on the unit sphere while BN operates on the $\mathbf{S}$-sphere (ellipsoid) (Cho and Lee, 2017). Note that one can compute the variance term (7) in a matrix-free manner, i.e. $\mathbf{S}$ never needs to be computed explicitly for BN.

Of course, this type of reparametrization is not exclusive to applications in neural networks. In the following two sections we first show how reparametrizing the weight space of linear models can be advantageous from a classical optimization point of view. In Section 5 we extend this analysis to training Batch Normalized neural networks with adaptive-stepsize Gradient Descent and show that the length-direction split induces an interesting decoupling effect of the individual network layers which simplifies the curvature structure.

3 ORDINARY LEAST SQUARES

As a preparation for subsequent analyses, we start with the simple convex quadratic objective encountered

[1]For example, suppose that conditional distribution $P(\mathbf{x}|y = 1)$ is gaussian with mean μ for positive labels and $-\mu$ for negative labels (mixture of gaussians). If the covariance matrix of these marginal distributions are the same, $z = y\mathbf{x}$ is Gaussian while $\mathbf{x}$ is not.

[2]In the (non-compositional) models of Section 3 and 4, fulfilling the assumption that $\mathbf{x}$ is zero-mean is as simple as centering the dataset. Yet, we also omit centering a neurons input as well as learning γ for the neural network analysis in Section 5.

when minimizing an ordinary least squares problem

$$\begin{aligned}&\min_{\tilde{\mathbf{w}}\in\mathbb{R}^d}\left(f_{\text{OLS}}(\tilde{\mathbf{w}}) := \mathbf{E}_{\mathbf{x},y}\left[\left(y-\mathbf{x}^\top\tilde{\mathbf{w}}\right)^2\right]\right)\\ \overset{(A1)}{\Leftrightarrow}&\min_{\tilde{\mathbf{w}}\in\mathbb{R}^d}\left(2\mathbf{u}^\top\tilde{\mathbf{w}}+\tilde{\mathbf{w}}^\top\mathbf{S}\tilde{\mathbf{w}}\right).\end{aligned} \tag{10}$$

One can think of this as a linear neural network with just one neuron and a quadratic loss. Thus, applying BN resembles reparametrizing $\tilde{\mathbf{w}}$ according to Eq. (9) and the objective turns into the *non-convex* problem

$$\min_{\mathbf{w}\in\mathbb{R}^d\setminus\{0\},g\in\mathbb{R}}\left(f_{\text{OLS}}(\mathbf{w},g) := 2g\frac{\mathbf{u}^\top\mathbf{w}}{\|\mathbf{w}\|_{\mathbf{S}}}+g^2\right). \tag{11}$$

Despite the non-convexity of this new objective, we will prove that Gradient Descent (GD) enjoys a *linear* convergence rate. Interestingly, our analysis establishes a link between f_{OLS} in reparametrized coordinates (Eq. (11)) and the well-studied task of minimizing (generalized) Rayleigh quotients as it is commonly encountered in eigenvalue problems (Argentati et al., 2017).

3.1 Convergence analysis

To simplify the analysis, note that, for a given $\mathbf{w}$, the objective of Eq. (11) is convex w.r.t. the scalar g and thus the optimal value $g^*_{\mathbf{w}}$ can be found by setting $\frac{\partial f_{\text{OLS}}}{\partial g}=0$, which gives $g^*_{\mathbf{w}} := -\left(\mathbf{u}^\top\mathbf{w}\right)/\|\mathbf{w}\|_{\mathbf{S}}$. Replacing this closed-form solution into Eq. (11) yields the following optimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^d\setminus\{0\}}\left(\rho(\mathbf{w}) := -\frac{\mathbf{w}^\top\mathbf{u}\mathbf{u}^\top\mathbf{w}}{\mathbf{w}^\top\mathbf{S}\mathbf{w}}\right), \tag{12}$$

which – as discussed in Appendix A.2 – is a special case of minimizing the generalized Rayleigh quotient for which an extensive literature exists (Knyazev, 1998; D'yakonov and McCormick, 1995). Here, we particularly consider solving (12) with GD, which applies the following iterative updates to the parameters

$$\mathbf{w}_{t+1} := \mathbf{w}_t + 2\eta_t\frac{((\mathbf{u}^\top\mathbf{w}_t)\mathbf{u}+\rho(\mathbf{w}_t)\mathbf{S}\mathbf{w}_t)}{\mathbf{w}_t^\top\mathbf{S}\mathbf{w}_t}. \tag{13}$$

Based upon existing results, the next theorem establishes a linear convergence rate for the above iterates to the minimizer $\mathbf{w}^*$ in the normalized coordinates.

Theorem 1. *[Convergence rate on least squares] Suppose that the (weak) Assumption 1 on the data distribution holds. Consider the* GD *iterates* $\{\mathbf{w}_t\}_{t\in\mathbb{N}^+}$ *given in Eq. (13) with the stepsize* $\eta_t = \mathbf{w}_t^\top\mathbf{S}\mathbf{w}_t/(2L|\rho(\mathbf{w}_t)|)$ *and starting from* $\rho(\mathbf{w}_0)\neq 0$. *Then,*

$$\Delta\rho_t \le \left(1-\frac{\mu}{L}\right)^{2t}\Delta\rho_0, \tag{14}$$

where $\Delta\rho_t := \rho(\mathbf{w}_t)-\rho(\mathbf{w}^*)$. *Furthermore, the* $\mathbf{S}^{-1}$-*norm of the gradient* $\nabla\rho(\mathbf{w}_t)$ *relates to the suboptimality as*

$$\|\mathbf{w}_t\|^2_{\mathbf{S}}\|\nabla\rho(\mathbf{w}_t)\|^2_{\mathbf{S}^{-1}}/|4\rho(\mathbf{w}_t)| = \Delta\rho_t. \tag{15}$$

This convergence rate is of the same order as the rate of standard GD on the original objective f_{OLS} of Eq. (10) (Nesterov, 2013). Yet, it is interesting to see that the non-convexity of the normalized objective does not slow gradient-based optimization down. In the following, we will repeatedly invoke this result to analyze more complex objectives for which GD only achieves a *sublinear* convergence rate in the original coordinate space but is provably accelerated after using Batch Normalization.

4 LEARNING HALFSPACES

We now turn our attention to the problem of Learning Halfspaces, which encompasses training the simplest possible neural network: the Perceptron. This optimization problem can be written as

$$\min_{\tilde{\mathbf{w}}\in\mathbb{R}^d}\left(f_{\text{LH}}(\tilde{\mathbf{w}}) := \mathbf{E}_{y,\mathbf{x}}\left[\varphi(\mathbf{z}^\top\tilde{\mathbf{w}})\right]\right) \tag{16}$$

where $\mathbf{z} := -y\mathbf{x}$ and $\varphi:\mathbb{R}\to\mathbb{R}^+$ is a loss function. Common choices for φ include the zero-one, piece-wise linear, logistic and sigmoidal loss. We here tailor our analysis to the following choice of loss function.

Assumption 3. *[Assumptions on loss function] We assume that the loss function* $\varphi:\mathbb{R}\to\mathbb{R}$ *is infinitely differentiable, i.e.* $\varphi\in C^\infty(\mathbb{R},\mathbb{R})$, *with a bounded derivative, i.e.* $\exists\Phi>0$ *such that* $|\varphi^{(1)}(\beta)|\le\Phi,\forall\beta\in\mathbb{R}$.

Furthermore, we need f_{LH} to be sufficiently smooth.

Assumption 4. *[Smoothness assumption] We assume that the objective* $f:\mathbb{R}^d\to\mathbb{R}$ *is* ζ*-smooth if it is differentiable on* $\mathbb{R}$ *and its gradient is* ζ*-Lipschitz. Furthermore, we assume that a solution* $\alpha^* := \arg\min_\alpha\|\nabla f(\alpha\mathbf{w})\|^2$ *exists that is bounded in the sense that* $\forall\mathbf{w}\in\mathbb{R}^d, -\infty<\alpha^*<\infty$.[3]

Since globally optimizing (16) is in general NP-hard (Guruswami and Raghavendra, 2009), we instead focus on understanding the effect of the normalized parameterization when searching for a stationary point. Towards this end we now assume that $\mathbf{z}$ is a multivariate normal random variable (see Assumption 2 and discussion there).

4.1 Global characterization of the objective

The learning halfspaces objective f_{LH} – on Gaussian inputs – has a remarkable property: all critical points lie on the same line, independent of the choice of the loss function φ. We formalize this claim in the next lemma.

[3] This is a rather technical but not so restrictive assumption. For example, it always holds for the sigmoid loss unless the classification error of $\mathbf{w}$ is already zero.

Lemma 1. *Under Assumptions 1 and 2, all bounded critical points $\tilde{\mathbf{w}}_*$ of f_{LH} have the general form*

$$\tilde{\mathbf{w}}_* = g_* \mathbf{S}^{-1}\mathbf{u},$$

where the scalar $g_* \in \mathbb{R}$ depends on $\tilde{\mathbf{w}}_*$ and the choice of the loss function φ.

Interestingly, the optimal direction of these critical points spans the same line as the solution of a corresponding least squares regression problem (see Eq. (37) in Appendix A). In the context of convex optimization of generalized linear models, this fact was first pointed out in (Brillinger, 2012). Although the global optima of the two objectives are aligned, classical optimization methods - which perform updates based on local information - are generally blind to such global properties of the objective function. This is unfortunate since Gradient Descent converges linearly in the quadratic least-squares setting but only sublinearly on general Learning Halfspace problems (Zhang et al., 2015). To accelerate the convergence of Gradient Descent, Erdogdu et al. (2016) thus proposed a two-step global optimization procedure for solving generalized linear models, which first involves finding the optimal direction by optimizing a least squares regression as a surrogate objective and secondly searching for a proper scaling factor of that minimizer. Here, we show that running GD in coordinates reparameterized as in Eq. (9) makes this two-step procedure redundant. More specifically, splitting the optimization problem into searching for the optimal direction and scaling separately, allows even local optimization methods to exploit the property of global minima alignment. Thus - without having to solve a least squares problem in the first place - the directional updates on the Learning Halfspace problem can mimic the least squares dynamics and thereby inherit the linear convergence rate. Combined with a fast (one dimensional) search for the optimal scaling in each step the overall convergence stays linear.

As an illustration, Figure 1 shows the level sets as well as the optimal direction of a least squares-, a logistic- and a sigmoidal regression problem on the same Gaussian dataset. Furthermore, it shows iterates of GD in original coordinates and a sequential version of GD in normalized coordinates that first optimizes the direction and then the scaling of its parameters (GDNP$_{seq}$). Both methods start at the same point and run with an infinitesimally small stepsize. It can be seen that, while GD takes completely different paths towards the optimal points of each problem instance, the dynamics of GDNP$_{seq}$ are exactly the same until the optimal direction[4] is found and differ only in the final scaling.

[4] Depicted by the dotted red line. Note that – as a result of Lemma 1 – this line is identical in all problems.

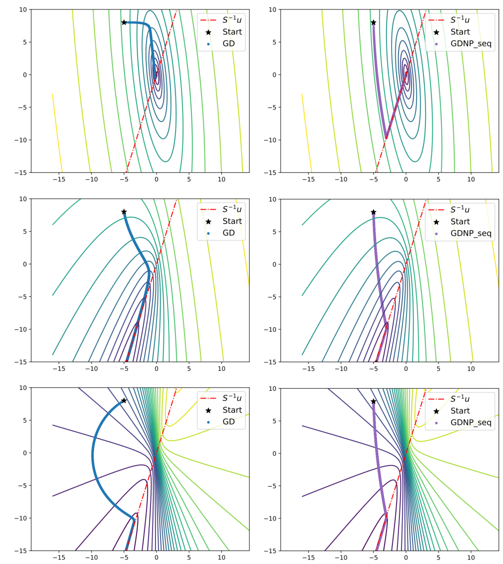

Figure 1: Level sets and path of GD (left) and GDNP$_{seq}$ (right) for a (i) least squares-, (ii) logistic-, and (iii) sigmoidal regression on a Gaussian dataset. All iterates are shown in original coordinates. Note that the GDNP$_{seq}$ paths are identical until the optimal direction (red line) is found.

4.2 Local optimization in normalized parameterization

Let $f_{\text{LH}}(\mathbf{w}, g)$ be the $\mathbf{S}$-reparameterized objective with $\tilde{\mathbf{w}} = g\mathbf{w}/\|\mathbf{w}\|_{\mathbf{S}}$ as defined in Eq. (9). We here consider optimizing this objective using Gradient Descent in Normalized Parameters (GDNP). In each iteration, GDNP performs a gradient step to optimize f_{LH} with respect to the direction $\mathbf{w}$ and does a one-dimensional bisection search to estimate the scaling factor g (see Algorithm 1). It is therefore only a slight modification of performing Batch Normalization plus Gradient Descent: Instead of taking a gradient step on both $\mathbf{w}$ and g, we search for the (locally)-optimal scaling g at each iteration. This modification is cheap and it simplifies the theoretical analysis but it can also be substituted easily in practice by performing multiple GD steps on the scaling factor, which we do for the experiments in Section 4.4.

4.3 Convergence result

We now show that Algorithm 1 can achieve a linear convergence rate to a critical point on the possibly non-convex objective f_{LH} with Gaussian inputs. Note that all information for computing the adaptive stepsize s_t is readily available and can be computed efficiently.

Theorem 2. *[Convergence rate of* GDNP *on learning halfspaces] Suppose Assumptions 1– 4 hold. Let $\tilde{\mathbf{w}}_{T_d}$ be the output of* GDNP *on f_{LH} with the following choice of* *stepsizes*

$$s_t := s(\mathbf{w}_t, g_t) = -\frac{\|\mathbf{w}_t\|_{\mathbf{S}}^3}{L g_t h(\mathbf{w}_t, g_t)} \tag{17}$$

for $t = 1, \dots, T_d$*, where*

$$\begin{aligned} h(\mathbf{w}_t, g_t) :=& \mathbf{E}_{\mathbf{z}}\left[\varphi'\left(\mathbf{z}^\top \tilde{\mathbf{w}}_t\right)\right]\left(\mathbf{u}^\top \mathbf{w}_t\right) \\ &- \mathbf{E}_{\mathbf{z}}\left[\varphi''\left(\mathbf{z}^\top \tilde{\mathbf{w}}_t\right)\right]\left(\mathbf{u}^\top \mathbf{w}_t\right)^2 \end{aligned} \tag{18}$$

is a stopping criterion. If initialized such that $\rho(\mathbf{w}_0) \neq 0$ *(see Eq. (12)), then* $\tilde{\mathbf{w}}_{T_d}$ *is an approximate critical point of* f_{LH} *in the sense that*

$$\begin{aligned} \|\nabla_{\tilde{\mathbf{w}}} f(\tilde{\mathbf{w}}_{T_d})\|^2 \leq& (1 - \mu/L)^{2T_d} \Phi^2 \left(\rho(\mathbf{w}_0) - \rho^*\right) \\ &+ 2^{-T_s} \zeta |b_t^{(0)} - a_t^{(0)}| / \mu^2. \end{aligned} \tag{19}$$

To complete the picture, Table 1 compares this result to the order complexity for reaching an ϵ-optimal critical point $\tilde{\mathbf{w}}$ (i.e.$\|\nabla_{\tilde{\mathbf{w}}} f_{\mathrm{LH}}(\tilde{\mathbf{w}})\| \leq \epsilon$) of Gradient Descent and Accelerated Gradient Descent (Agd) in original coordinates. Although we observe that the accelerated rate achieved by Gdnp is significantly better than the best known rate of Agd for non-convex objective functions, we need to point out that the rate for Gdnp relies on the assumption of a Gaussian data distribution. Yet, Section 4.4 includes promising experimental results in a more practical setting with non-Gaussian data.

Finally, we note that the proof of this result relies specifically on the **S**-reparametrization done by Batch Normalization. In Appendix B.3.6 we detail out why our proof strategy is not suitable for the **I**-reparametrization of Weight Normalization and thus leave it as an interesting open question if other settings (or proof strategies) can be found where linear rates for Wn are provable.

4.4 Experiments I

Setting In order to substantiate the above analysis we compare the convergence behavior of Gd and Agd to three versions of Gradient Descent in normalized coordinates. Namely, we benchmark (i) Gdnp (Algorithm 1) with multiple gradient steps on g instead of Bisection, (ii) a simpler version (Bn) which updates $\mathbf{w}$ and $\mathbf{g}$ with just one fixed step-size gradient step[5] and (iii) Weight Normalization (Wn) as presented in (Salimans and Kingma, 2016). All methods use full batch sizes and – except for Gdnp on $\mathbf{w}$ – each method is run with a problem specific, constant stepsize.

Algorithm 1 Gradient Descent in Normalized Parameterization (Gdnp)

Input: T_{d}, T_{s}, stepsize policy s, stopping criterion h and objective f
$g \leftarrow 1$, and initialize $\mathbf{w}_0$ such that $\rho(\mathbf{w}_0) \neq 0$
for $t = 1, \dots, T_{\mathrm{d}}$ **do**
  if $h(\mathbf{w}_t, g_t) \neq 0$ **then**
    $s_t \leftarrow s(\mathbf{w}_t, g_t)$
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - s_t \nabla_{\mathbf{w}} f(\mathbf{w}_t, g_t)$ # directional step
  end if
  $g_t \leftarrow \mathrm{Bisection}(T_{\mathrm{s}}, f, \mathbf{w}_t)$ # see Alg. 3
end for
$\tilde{\mathbf{w}}_{T_{\mathrm{d}}} \leftarrow g_{T_{\mathrm{d}}} \mathbf{w}_{T_{\mathrm{d}}} / \|\mathbf{w}_{T_{\mathrm{d}}}\|_{\mathbf{S}}$
return $\tilde{\mathbf{w}}_{T_{\mathrm{d}}}$

We consider empirical risk minimization as a surrogate for f_{LH} (16) on the common real-world dataset *a9a* as well as on synthetic data drawn from a multivariate Gaussian distribution. We center the datasets and use two different functions $\varphi(\cdot)$. First, we choose the *softplus* which resembles the classical logistic regression (convex). Secondly, we use the *sigmoid* which is a commonly used (non-convex) continuous approximation of the 0-1 loss (Zhang et al., 2015). Further details can be found in Appendix D.

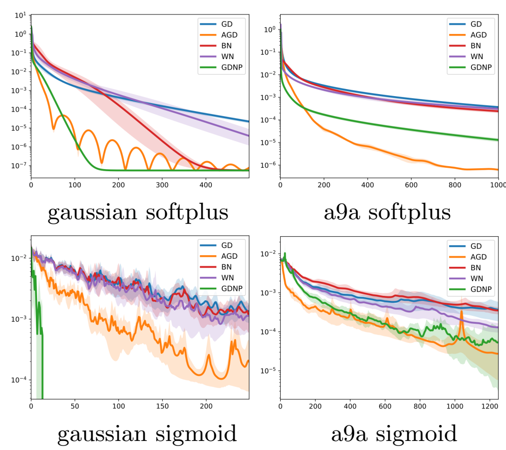

Figure 2: Results of an average run (solid line) in terms of log suboptimality (softplus) and log gradient norm (sigmoid) over iterations as well as 90% confidence intervals of 10 runs with random initialization.

Results The Gaussian design experiments clearly confirm Theorem 4 in the sense that the loss in the convex-, as well as the gradient norm in the non-convex case decrease at a linear rate. The results on *a9a* show that Gdnp can accelerate optimization even when the normality assumption does not hold and in a setting where no covariate shift is present, which motivates future research of normalization techniques in optimization. Interestingly, the performance of simple Bn and Wn is similar to that of Gd, which suggests that the length-direction decoupling on its own does not capture the entire potential of these methods. Gdnp on

[5] Thus Bn is conceptually very close to the classical Batch Norm Gradient Descent presented in (Ioffe and Szegedy, 2015)

Method	Assumptions	Complexity	Rate	Reference
GD	Smoothness	$\mathcal{O}(T_{\mathrm{gd}}\epsilon^{-2})$	Sublinear	(Nesterov, 2013)
AGD	Smoothness	$\mathcal{O}(T_{\mathrm{gd}}\epsilon^{-7/4}\log(1/\epsilon))$	Sublinear	(Jin et al., 2017)
AGD	Smoothness+convexity	$\mathcal{O}(T_{\mathrm{gd}}\epsilon^{-1})$	Sublinear	(Nesterov, 2013)
GDNP	1, 2, 3, and 4	$\mathcal{O}(T_{\mathrm{gd}}\log^2(1/\epsilon))$	Linear	This paper

Table 1: Computational complexity to reach an ϵ-critical point on f_{LH} – with a (possibly) non-convex loss function φ. $\mathcal{O}(T_{\mathrm{gd}})$ represents the time complexity of computing a gradient for each method.

the other hand takes full advantage of the parameter splitting, both in terms of multiple steps on g and – more importantly – adaptive stepsizes in $\mathbf{w}$.

5 NEURAL NETWORKS

We now turn our focus to optimizing the weights of a multilayer perceptron (MLP) with one hidden layer and m hidden units that maps an input $\mathbf{x} \in \mathbb{R}^d$ to a scalar output in the following way

$$F_{\mathbf{x}}(\tilde{\mathbf{W}}, \Theta) = \sum_{i=1}^{m} \theta_i \varphi(\mathbf{x}^\top \tilde{\mathbf{w}}^{(i)}).$$

The $\tilde{\mathbf{w}}^{(i)} \in \mathbb{R}^d$ and $\theta^{(i)} \in R$ terms are the input and output weights of unit i and $\varphi : \mathbb{R} \to \mathbb{R}$ is a so-called activation function. We assume φ to be a tanh, which is a common choice in many neural network architectures used in practice. Given a loss function $\ell : \mathbb{R} \to \mathbb{R}^+$, the optimal input- and output weights are determined by minimizing the following optimization problem

$$\min_{\tilde{\mathbf{W}}, \Theta} \left(f_{\mathrm{NN}}(\tilde{\mathbf{W}}, \Theta) := \mathbf{E}_{y,\mathbf{x}} \left[\ell\Big(-yF_{\mathbf{x}}(\tilde{\mathbf{W}}, \Theta)\Big)\right]\right), \quad (20)$$

$$\tilde{\mathbf{W}} := \{\tilde{\mathbf{w}}^{(1)}, \dots, \tilde{\mathbf{w}}^{(m)}\}, \;\; \Theta := \{\theta^{(1)}, \dots, \theta^{(m)}\}.$$

In the following, we analyze the optimization of the input weights $\tilde{\mathbf{W}}$ with frozen output weights Θ and thus write $F(\tilde{\mathbf{W}})$ hereafter for simplicity. As previously assumed for the case of learning halfspaces, our analysis relies on Assumption 2. This approach is rather common in the analysis of neural networks, e.g. Brutzkus and Globerson (2017) showed that for a special type of one-hidden layer network with isotropic Gaussian inputs, all local minimizers are global. Under the same assumption, similar results for different classes of neural networks were derived in (Li and Yuan, 2017; Soltanolkotabi et al., 2017; Du et al., 2017).

5.1 Global characterization of the objective

We here show that, under the same assumption, the landscape of f_{NN} exhibits a global property similar to one of the Learning Halfspaces objective (see Section 4.1). In fact, the critical points $\tilde{\mathbf{w}}_i$ of all neurons in a hidden layer align along one and the same single line in $\mathbb{R}^d$ and this direction depends only on *incoming* information into the hidden layer.

Lemma 2. *Suppose Assumptions 1 and 2 hold and let $\hat{\mathbf{w}}^{(i)}$ be a critical point of $f_{\mathrm{NN}}(\tilde{\mathbf{W}})$ with respect to hidden unit i and for a fixed $\Theta \neq \mathbf{0}$. Then, there exits a scalar $\hat{c}^{(i)} \in \mathbb{R}$ such that*

$$\hat{\mathbf{w}}^{(i)} = \hat{c}^{(i)}\mathbf{S}^{-1}\mathbf{u}, \quad \forall i = 1, \dots, m. \quad (21)$$

See Appendix C.2 for a discussion of possible implications for deep neural networks.

Algorithm 2 Training an MLP with GDNP

1: Input: $T_{\mathrm{d}}^{(i)}$, $T_{\mathrm{s}}^{(i)}$, step-size policies $s^{(i)}$, stopping criterion $h^{(i)}$
2: $\mathbf{W}_{/1} \leftarrow \mathbf{0}$, $\mathbf{g}_{/1} = \mathbf{1}$
3: **for** $i = 1, \dots, m$ **do**
4: $\quad \mathbf{W}_{/i} := \{\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(i-1)}, \mathbf{w}^{(i+1)}, \dots, \mathbf{w}^{(m)}\}$, $\mathbf{g}_{/i} := \{g^{(1)}, \dots, g^{(i-1)}, g^{(i+1)}, \dots, g^{(m)}\}$
5: $\quad f^{(i)}(\mathbf{w}^{(i)}, g^{(i)}) := f_{\mathrm{NN}}(\mathbf{w}^{(i)}, g^{(i)}, \mathbf{W}_{/i}, \mathbf{g}_{/i})$
6: $\quad (\mathbf{w}^{(i)}, g^{(i)}) \leftarrow$ GDNP($f^{(i)}$, $T_{\mathrm{d}}^{(i)}$, $T_{\mathrm{s}}^{(i)}$, $s^{(i)}$, $h^{(i)}$)
7: **end for**
8: **return** $\mathbf{W} := [\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(m)}], \mathbf{g} := [g^1, \dots, g^m]$.

5.2 Convergence result

We again consider normalizing the weights of each unit by means of its input covariance matrix $\mathbf{S}$, i.e. $\tilde{\mathbf{w}}^{(i)} = g^{(i)}\mathbf{w}^{(i)}/\|\mathbf{w}^{(i)}\|_{\mathbf{S}}, \;\forall i = 1, \dots, m$ and present a simple algorithmic manner to leverage the input-output decoupling of Lemma 2. Contrary to BN, which optimizes all the weights simultaneously, our approach is based on alternating optimization over the different hidden units. We thus adapt GDNP to the problem of optimizing the units of a neural network in Algorithm 2 but note that this modification is mainly to simplify the theoretical analysis.

Optimizing each unit independently formally results in minimizing the function $f^{(i)}(\mathbf{w}^{(i)}, g^{(i)})$ as defined in Algorithm 2. In the next theorem, we prove that this version of GDNP achieves a linear rate of convergence to optimize each $f^{(i)}$.

Theorem 3. *[Convergence of* GDNP *on MLP] Suppose Assumptions 1– 4 hold. We consider optimizing the weights $(\mathbf{w}^{(i)}, g^{(i)})$ of unit i, assuming that all directions $\{\mathbf{w}^{(j)}\}_{j<i}$ are critical points of f_{NN} and $\mathbf{w}^k = \mathbf{0}$ for $k > i$. Then,* GDNP *with step-size policy $s^{(i)}$ as in* (100) *and stopping criterion $h^{(i)}$ as in* (101) *yields a* *linear convergence rate on* $f^{(i)}$ *in the sense that*

$$\|\nabla_{\tilde{\mathbf{w}}^{(i)}} f(\tilde{\mathbf{w}}_t^{(i)})\|^2_{\mathbf{S}^{-1}} \leq (1-\mu/L)^{2t} C\left(\rho(\mathbf{w}_0) - \rho^*\right) + 2^{-T_s^{(i)}} \zeta |b_t^{(0)} - a_t^{(0)}|/\mu^2, \tag{22}$$

where the constant $C > 0$ *is defined in Eq.* (104)*.*

The result of Theorem 3 relies on the fact that each $\mathbf{w}^{(j)}, j \neq i$ is either zero or has zero gradient. If we assume that an exact critical point is reached after optimizing each individual unit, then the result directly implies that the alternating minimization presented in Algorithm 2 reaches a critical point of the overall objective. Since the established convergence rate for each individual unit is linear, this assumption sounds realistic. We leave a more precise convergence analysis, that takes into account that optimizing each individual unit for a finite number of steps may yield numerical suboptimalities, for future work.

5.3 Experiments II

Setting In the proof of Theorem 3 we show that Gdnp can leverage the length-direction decoupling in a way that lowers cross-dependencies between hidden layers and yields faster convergence. A central part of the proof is Lemma 2 which says that – given Gaussian inputs – the optimal direction of a given layer is independent of all downstream layers. Since this assumption is rather strong and since Algorithm 2 is intended for analysis purposes only, we test the validity of the above hypothesis outside the Gaussian setting by training a Batch Normalized multilayer feedforward network (Bn) on a real-world image classification task with plain Gradient Descent. For comparison, a second *unnormalized* network is trained by Gd. To validate Lemma 2 we measure the interdependency between the central and all other hidden layers in terms of the Frobenius norm of their second partial cross derivatives (in the directional component). Further details can be found in Appendix D.

Results Figure 3 confirms that the directional gradients of the central layer are affected far more by the upstream than by the downstream layers to a surprisingly large extent. Interestingly, this holds even before reaching a critical point. The downstream cross-dependencies are generally decaying for the Batch Normalized network (Bn) (especially in the first 1000 iterations where most progress is made) while they remain elevated in the un-normalized network (Gd), which suggest that using Batch Normalization layers indeed simplifies the networks curvature structure in $\mathbf{w}$ such that the length-direction decoupling allows Gradient Descent to exploit simpler trajectories in these normalized coordinates for faster convergence. Of course, we

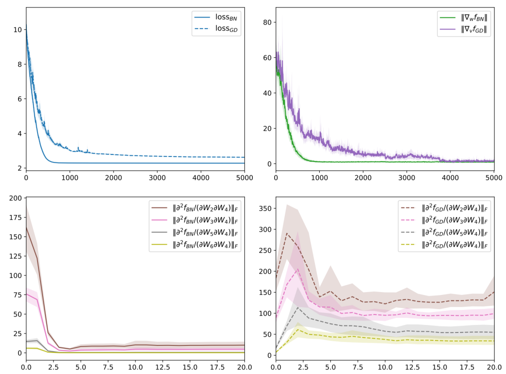

Figure 3: (i) Loss, (ii) gradient norm and dependencies between central- and all other layers for BN (iii) and GD (iv) on a 6 hidden layer network with 50 units (each) on the CIFAR10 dataset (5 runs with random initialization, 5000 iterations, curvature information computed each 200th).

cannot untangle this effect fully from other possible positive aspects of training with Bn (see introduction). Yet, the fact that the (de-)coupling increases in the distance to the middle layer (note how earlier (later) layers are more (less) important for $\mathbf{W}_4$) emphasizes the relevance of this analysis particularly for deep neural network structures, where downstream dependencies might vanish completely with depth. This does not only make gradient based training easier but also suggests the possibility of using partial second order information, such as diagonal Hessian approximations (e.g. proposed in (Martens et al., 2012)).

6 CONCLUSION

We took a theoretical approach to study the acceleration provided by Batch Normalization. In a simplified setting, we have shown that the reparametrization performed by Batch Normalization leads to a provable acceleration of gradient-based optimization by splitting it into subtasks that are easier to solve. In order to evaluate the impact of the assumptions required for our analysis, we also performed experiments on real-world datasets that agree with the results of the theoretical analysis to a surprisingly large extent.

We consider this work as a first step for two particular directions of future research. First, it raises the question of how to optimally train Batch Normalized neural networks. Particularly, our results suggest that different and adaptive stepsize schemes for the two parameters - length and direction - can lead to significant accelerations. Second, the analysis of Section 3 and 4 reveals that a better understanding of non-linear coordinate transformations is a promising direction for the continuous optimization community.

References

Absil, P.-A., Mahony, R., and Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press.

Argentati, M. E., Knyazev, A. V., Neymeyr, K., Ovtchinnikov, E. E., and Zhou, M. (2017). Convergence theory for preconditioned eigenvalue solvers in a nutshell. *Foundations of Computational Mathematics*, 17(3):713–727.

Arpit, D., Zhou, Y., Kota, B. U., and Govindaraju, V. (2016). Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. *arXiv preprint arXiv:1603.01431*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Brillinger, D. R. (2012). A generalized linear model with "gaussian" regressor variables. In *Selected Works of David Brillinger*, pages 589–606. Springer.

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

Cho, M. and Lee, J. (2017). Riemannian approach to batch normalization. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5225–5235. Curran Associates, Inc.

Dowson, D. and Wragg, A. (1973). Maximum-entropy distributions having prescribed first and second moments (corresp.). *IEEE Transactions on Information Theory*, 19(5):689–693.

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

Du, S. S., Lee, J. D., Tian, Y., Poczos, B., and Singh, A. (2017). Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779*.

D'yakonov, E. G. and McCormick, S. (1995). *Optimization in solving elliptic problems*. CRC Press.

Erdogdu, M. A., Dicker, L. H., and Bayati, M. (2016). Scaled least squares estimator for glms in large-scale problems. In *Advances in Neural Information Processing Systems*, pages 3324–3332.

Gitman, I. and Ginsburg, B. (2017). Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *arXiv preprint arXiv:1709.08145*.

Guruswami, V. and Raghavendra, P. (2009). Hardness of learning halfspaces with noise. *SIAM Journal on Computing*, 39(2):742–765.

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

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Jin, C., Netrapalli, P., and Jordan, M. I. (2017). Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 972–981.

Knyazev, A. and Shorokhodov, A. (1991). On exact estimates of the convergence rate of the steepest ascent method in the symmetric eigenvalue problem. *Linear algebra and its applications*, 154:245–257.

Knyazev, A. V. (1998). Preconditioned eigensolvers—an oxymoron. *Electron. Trans. Numer. Anal*, 7:104–123.

Knyazev, A. V. and Neymeyr, K. (2003). A geometric theory for preconditioned inverse iteration iii: A short and sharp convergence estimate for generalized eigenvalue problems. *Linear Algebra and its Applications*, 358(1-3):95–114.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Landsman, Z. and Nevslehová, J. (2008). Stein's lemma for elliptical random vectors. *Journal of Multivariate Analysis*, 99(5):912–927.

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

Lipton, Z. C. and Steinhardt, J. (2018). Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*.

Martens, J., Sutskever, I., and Swersky, K. (2012). Estimating the hessian by back-propagating curvature. *arXiv preprint arXiv:1206.6464*.

Mikhalevich, V., Redkovskii, N., and Antonyuk, A. (1988). Minimization methods for smooth nonconvex functions. *Cybernetics*, 24(4):395–403.

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

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*.

Soltanolkotabi, M., Javanmard, A., and Lee, J. D. (2017). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *arXiv preprint arXiv:1707.04926*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

Zhang, Y., Lee, J. D., Wainwright, M. J., and Jordan, M. I. (2015). Learning halfspaces and neural networks with random initialization. *arXiv preprint arXiv:1511.07948*.