STOCHASTIC ASPECTS OF DYNAMICAL LOW-RANK APPROXIMATION IN THE CONTEXT OF MACHINE LEARNING*

ARSEN HNATIUK†, JONAS KUSCH‡, LISA KUSCH§, NICOLAS R. GAUGER¶, AND ANDREA WALTHER‖

Abstract. The central challenges of today's neural network architectures are the prohibitive memory footprint and the training costs associated with determining optimal weights and biases. A large portion of research in machine learning is therefore dedicated to constructing memory-efficient training methods. One promising approach is dynamical low-rank training (DLRT) which represents and trains parameters as a low-rank factorization. While DLRT is equipped with several beneficial properties, analytic results are currently limited to deterministic gradient flows. In this work, we show that dynamical low-rank training in combination with stochastic gradient and momentum methods fulfills descent guarantees and prove its convergence to an optimal point.

Key words. Dynamical Low-Rank Approximation (DLRA), Dynamical Low-Rank Training (DLRT), Machine Learning, Stochastic Gradient Descent, Deep Neural Networks

MSC codes. 68Q25, 68T05, 68T07, 65K10, 93E35, 65L20, 65L70

1. Introduction.

In recent years, deep neural networks (DNNs) have consistently and radically redefined the state-of-the-art in tasks related to computer vision, natural language processing, and many others. This was in part made possible by the rapid increase in the size of the networks, with some of the newest DNN models having trillions of parameters [23]. Despite this, there is evidence that large model sizes are not necessary for good performance; large DNNs are known to contain parameter redundancies [11, 21, 8, 2, 9] and experiments show that it is often possible to substantially reduce the model size with little loss in output quality. However, the task of finding such smaller yet well-performing models is highly challenging [11].

The process of removing redundant parameters is called network pruning and various techniques have been proposed to achieve this [2]. One possibility to compress neural networks during training is low-rank pruning [24, 12, 26, 21]. This work will focus on the method proposed in [21], which applies the principles of Dynamical Low-Rank Approximation (DLRA) [15] to the task of training artificial neural networks. The resulting Dynamical Low-Rank Training (DLRT) offers two main advantages over conventional low-rank pruning methods: First, it allows for a dynamic adaptation of the approximation rank during training. Second, its convergence is not slowed down due to the curvature of the manifold containing low-rank matrices.

The core idea is to train the network while dynamically restricting the rank of its parameter matrices. This method has been experimentally shown to significantly reduce the model size and computational costs while sacrificing little accuracy for fully connected and convolutional layers [21]. Although much effort has been invested in the study of error bounds and robustness properties of the DLRA algorithm [15, 19, 5, 13], its convergence behavior when using stochastic gradients remains unexplored. Since

*Submitted to the editors 03.17.2024.

Funding: The research was funded partly by the DFG under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID:390685689).

†Humboldt-Universität zu Berlin (arsen.hnatiuk@hu-berlin.de).

‡Norges miljø- og biovitenskapelige universitet (jonas.kusch@nmbu.no).

§Technische Universiteit Eindhoven (l.kusch@tue.nl).

¶University of Kaiserslautern-Landau (RPTU) (nicolas.gauger@scicomp.uni-kl.de).

‖Humboldt-Universität zu Berlin (andrea.walther@math.hu-berlin.de).

the training of machine learning models such as DNNs generally relies on stochastic gradients [3], the study of DLRA in the stochastic setting is fundamental for developing a theoretical understanding of the method proposed in [21]. This work will attempt to fill this gap in the theory of DLRA. The main findings are

1. *Robustness of stochastic gradients.* We prove that the robust error bound of DLRA holds in combination with stochastic gradient descent algorithms.
2. *Descent direction.* We prove that for sufficiently small learning rates, the method in combination with stochastic gradient descent will retain the descent guarantee from the deterministic and time-continuous setting.
3. *Convergence.* We show that DLRA in combination with stochastic gradient and momentum methods will converge to a local minimum if the basis reaches equilibrium.

This article is structured as follows. In Section 2 and Section 3, we provide an overview of neural network training and Dynamical Low-Rank Approximation. Section 4 discusses the robust error bound for DLRA in the presence of stochastic gradients. In Section 5, we investigate the descent direction with stochastic gradients and provide a convergence proof. Lastly, we provide a conclusion and outlook in Section 6.

2. Recap: Training of deep neural networks. Deep neural networks (DNNs) are a special type of machine learning models. In their simplest fully connected form, DNNs with N layers are functions

$$\mathcal{N} : \mathbb{R}^{D_1} \times \mathbb{R}^p \to \mathbb{R}^{D_2}, \quad (x; \mathcal{W}) \mapsto y,$$

where the vector of trainable parameters $\mathcal{W} \in \mathbb{R}^p$ is arranged in a sequence of matrices $W^1 \in \mathbb{R}^{n_{1,1} \times n_{1,2}}, \cdots, W^N \in \mathbb{R}^{n_{N,1} \times n_{N,2}}$ of corresponding dimensions $n_{k,1}, n_{k,2} \in \mathbb{N}$ with $p = \sum_{i=1}^{N} n_{i,1} n_{i,2}$, $x = z_0 \in \mathbb{R}^{D_1}$ is the input vector of dimension $D_1 \in \mathbb{N}$, and the output $y \in \mathbb{R}^{D_2}$ of dimension $D_2 \in \mathbb{N}$ is calculated by an N-fold nesting of intermediary steps

$$z_k = \phi_k(W^k z_{k-1})$$

for some non-linear functions ϕ_k [21]. Each intermediary step represents a layer of the network and its trainable parameters are the matrices W^k.

The training of a DNN is the task of minimizing a real-valued cost function $\mathcal{L}$ over the parameter space $\mathbb{R}^p$. A common way of performing this optimization is to apply the gradient descent algorithm

$$\mathcal{W}_{t+1} = \mathcal{W}_t - h \nabla_{\mathcal{W}} \mathcal{L}(\mathcal{W}_t),$$

where h is the step size, or, as it is called in the context of machine learning, the *learning rate* [3, 25]. This algorithm starts in some point $\mathcal{W}_0 \in \mathbb{R}^p$ and successively moves in the direction of steepest descent $-\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{W}_t)$ of the objective function until some optimality criterion is reached. It is also possible to consider a setting where the parameters $\mathcal{W}(t)$ evolve smoothly in time. In this case, $\mathcal{W}(t)$ can be expressed as a solution to the differential equation (also called the gradient flow)

$$\dot{\mathcal{W}}(t) = -\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{W}(t)), \quad \mathcal{W}(t_0) = \mathcal{W}_0. \tag{2.1}$$

Typically, the cost function $\mathcal{L}$ has the form

$$\mathcal{L} : \mathbb{R}^p \to \mathbb{R}\,, \quad \mathcal{W} \mapsto \frac{1}{d}\sum_{i=1}^{d} l(x_i, \mathcal{W})\,,$$

where $\{x_1, \dots, x_d\}$ is a training data set, and l is a so-called loss function, which quantifies the difference between the model output for a single point x_i and the desired output, given the parameters $\mathcal{W}$ [3]. We usually assume that l is differentiable, so the gradient of $\mathcal{L}$ can be written as

$$\nabla_{\mathcal{W}}\mathcal{L}(\mathcal{W}) = \frac{1}{d}\sum_{i=1}^{d} \nabla_{\mathcal{W}} l(x_i, \mathcal{W})\,.$$

Since, as in (2.1), we are mostly interested in the negative gradient, we will use the shorthand notation $F(\mathcal{W}) = -\nabla_{\mathcal{W}}\mathcal{L}(\mathcal{W})$.

As the size d of the training data set gets larger, the cost of computing the full gradient F becomes prohibitively high [3]. It is thus usual to only use a small subset of the data to compute the so-called *minibatch* or *stochastic* gradient

$$f(\mathcal{W}) \coloneqq -\frac{1}{s}\sum_{i=1}^{s} \nabla_{\mathcal{W}} l(\xi^i, \mathcal{W})\,,$$

where $s \ll d$ is fixed and the ξ^i are i.i.d. random variables that follow a uniform distribution over the training data set $\{x_1, \dots, x_d\}$ [3]. We can also write $f(\mathcal{W}, \xi)$ to underline the presence of randomness, where $\xi = (\xi^1, \dots, \xi^s)$. In a setting like that in (2.1), one would in practice use the stochastic gradient f instead of F [3].

By construction, it is clear that $\mathbb{E}_\xi[f(\mathcal{W}, \xi)] = F(\mathcal{W})$. By the law of large numbers, for a large enough s, we can expect f to come arbitrarily close to F.

3. Recap: Dynamical Low-Rank Approximation. The fact that the parameters of a neural network naturally appear as matrices is central to the pruning strategy proposed in [21]. If we can approximate a parameter matrix $W \in \mathbb{R}^{m\times n}$ by a matrix $Y \in \mathbb{R}^{m\times n}$ of rank $q \ll \min\{m, n\}$ while maintaining good performance of the network, we can significantly reduce the number of trainable parameters and the associated computational costs, both for training and for inference. We can calculate that while the matrix W has nm entries, we can encode Y in only $mq + nq + q^2$ entries since the singular value decomposition allows us to write any rank-q matrix Y as $Y = USV^\top$, with $S \in \mathbb{R}^{q\times q}$. The main motivation behind the approach proposed in [21] is to represent and train such a low-rank approximation without computing and storing full-rank parameter matrices. Such an efficient and robust training method is derived by the use of Dynamical Low-Rank Approximation [15], which is a model order reduction technique for time-dependent matrices. In this section, we present the overall principle of [15] and explore practical implementations.

Throughout this work, let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ refer to the Frobenius norm and scalar product. Also, for a matrix U, let $P_U = UU^\top$ be the projection onto the space spanned by the columns of U. Lastly, we will generally denote full-rank matrices by the letter W, while low-rank matrices will be denoted by Y.

3.1. Rank-q Approximation. For $q \in \mathbb{N}$ and $m, n \in \mathbb{N}$ such that $m, n \geq q$, the space of rank-q $\mathbb{R}^{m\times n}$ matrices is a smooth manifold, as seen in Example 8.14 of [17], and we will denote it with $\mathcal{M}_q = \mathcal{M}_q^{m\times n}$. Let us further denote with $\mathcal{T}_Y\mathcal{M}_q$

the tangent space of $\mathcal{M}_q$ at $Y \in \mathcal{M}_q$ and with $P(Y)$ the orthogonal projection onto $\mathcal{T}_Y\mathcal{M}_q$.

The starting point for Dynamical Low-Rank Approximation (DLRA), first introduced in [15], is the task of approximating time-dependent matrices $W(t) \in \mathbb{R}^{m\times n}$, smooth in t, by matrices $Y(t) \in \mathcal{M}_q$ of rank $q < \min\{m, n\}$. why $<$ and not $\leq$ as in other places? This task can be solved by finding elements of

$$\operatorname*{arg\,min}_{Y(t)\in\mathcal{M}_q} \|Y(t) - W(t)\| .$$

This simple approach faces many challenges, such as the need to calculate a costly singular value decomposition for each time value t and the fact that it yields a solution $Y(t)$ that is generally not smooth in t [15].

An alternative method of finding rank-q approximations that avoids these issues consists of approximating the initial value $W(t_0)$ and the derivative $\dot{W}(t)$ instead of $W(t)$ itself. The task is thus to find a solution $Y(t)$ of

$$\dot{Y}(t) \in \operatorname*{arg\,min}_{\dot{Y}(t)\in\mathcal{T}_{Y(t)}\mathcal{M}_q} \|\dot{Y}(t) - \dot{W}(t)\| , \quad Y(t_0) = Y_0$$

or, equivalently,

$$\dot{Y}(t) = P(Y(t))\dot{W}(t) , \quad Y(t_0) = Y_0 \tag{3.1}$$

with a $Y_0 \in \mathcal{M}_q$ that approximates $W(t_0)$ [15].

A common situation and one that we will explore from now on is where $W(t)$ is a solution of the matrix differential equation

$$\dot{W}(t) = F(W(t)) , \quad W(t_0) = W_0 \tag{3.2}$$

for some smooth function $F : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$. The approach (3.1) fits naturally in this setting, and if $\dot{W}(t) = F(W(t))$ is not known, we can replace it with its approximation $F(Y(t))$ [15].

Approximating $W(t)$ by the rank-q solution $Y(t)$ of

$$\dot{Y}(t) = P(Y(t))F(Y(t)) \in \mathcal{T}_{Y(t)}\mathcal{M}_q , \quad Y(t_0) = Y_0 \in \mathcal{M}_q , \tag{3.3}$$

where Y_0 is a rank-q approximation of W_0, is the formulation of DLRA which we will use in this work.

3.2. Robust Numerical Integrators. As already discussed, it is computationally advantageous to work with rank-q matrices $Y \in \mathbb{R}^{m\times n}$ in their decomposed form

$$Y = USV^\top , \tag{3.4}$$

where $U \in \mathbb{R}^{m\times q}$ and $V \in \mathbb{R}^{n\times q}$ have orthonormal columns and $S \in \mathbb{R}^{q\times q}$ is invertible.

Therefore, and since computing singular value decompositions at each time step is expensive, numerical integrations of (3.3) attempt to integrate the U, S, and V matrices separately. Proposition 2.1 in [15] uses (3.3) to derive this as the following system of equations.

$$\begin{aligned} \dot{S}(t) &= U(t)^\top F(Y(t))V(t) , \\ \dot{U}(t) &= (I - P_{U(t)})F(Y(t))V(t)S(t)^{-1} , \\ \dot{V}(t) &= (I - P_{V(t)})F(Y(t))^\top U(t)S(t)^{-\top} . \end{aligned} \tag{3.5}$$

From this, Lemma 4.1 in [15] yields an expression for the projector $P(Y)$.

$$P(Y)Z = ZP_V - P_U Z P_V + P_U Z \tag{3.6}$$

for any $Z \in \mathbb{R}^{m\times n}$. This representation is unique since P_U and P_V project onto the range and co-range of Y and thus are uniquely determined by Y.

The numerical issues appearing in (3.5) when S contains small singular values have pushed researchers to develop more robust integrators [19, 7, 5, 6, 14, 4, 16].

Algorithm 3.1 (Rank-Adaptive) Basis-Update and Galerkin (BUG) Integrator [5], as used in [21].

Input: $Y_0 = U_0 S_0 V_0^\top \in \mathcal{M}_{q_0}$ as in (3.4), an initial rank q_0, and a truncation tolerance $\vartheta > 0$.

for $k = 0, 1, \dots$ ***and*** $t_0 < t_1 < \cdots < t_k < \cdots$ **do**

K-step: solve the $\mathbb{R}^{m\times q_k}$ differential equation
$\dot{K}(t) = F(K(t)V_k^\top)V_k,\ K(t_k) = U_k S_k$ over $[t_k, t_{k+1}]$.

if ***rank-adaptive*** **then**

Set $K(t_{k+1}) \leftarrow [K(t_{k+1}) \mid U_k]$.

Using the QR-decomposition, obtain $U^*_{k+1} R_{k+1} = K(t_{k+1})$, where the columns of U^*_{k+1} form an orthonormal basis of the range of $K(t_{k+1})$.

Set $M \leftarrow U^{*\top}_{k+1} U_k$.

L-step: solve the $\mathbb{R}^{n\times q_k}$ differential equation
$\dot{L}(t) = F(U_k L(t)^\top)^\top U_k,\ L(t_k) = V_k S_k^\top$ over $[t_k, t_{k+1}]$.

if ***rank-adaptive*** **then**

Set $L(t_{k+1}) \leftarrow [L(t_{k+1}) \mid V_k]$.

Using the QR-decomposition, obtain $V^*_{k+1} \tilde{R}_{k+1} = L(t_{k+1})$, where the columns of V^*_{k+1} form an orthonormal basis of the range of $L(t_{k+1})$.

Set $N \leftarrow V^{*\top}_{k+1} V_k$.

S-step: solve the $\mathbb{R}^{\tilde{q}_U \times \tilde{q}_V}$ differential equation (for $\tilde{q}_U, \tilde{q}_V \in [q_k, 2q_k]$, depending on the sizes of U^*_{k+1} and V^*_{k+1} respectively)
$\dot{S}(t) = U^{*\top}_{k+1} F(U^*_{k+1} S(t) V^{*\top}_{k+1}) V^*_{k+1},\ S(t_k) = M S_k N^\top$ over $[t_k, t_{k+1}]$.

Set $S^*_{k+1} \leftarrow S(t_{k+1})$.

if ***rank-adaptive*** **then**

Truncation step: Compute the singular value decomposition $S^*_{k+1} = P\Sigma Q^\top$ with $\Sigma = diag(\sigma_i)$.

Determine the maximal set of singular values σ_i of S^*_{k+1} satisfying $\sum \sigma_i^2 \le \vartheta^2$ and define Σ_1, P_1 and Q_1 by removing the rows and columns from Σ, P and Q corresponding to those σ_i.

Set $S_{k+1} \leftarrow \Sigma_1$, $U_{k+1} \leftarrow U^*_{k+1} P_1$ and $V_{k+1} \leftarrow V^*_{k+1} Q_1$.

else

Set $U_{k+1} \leftarrow U^*_{k+1}$, $V_{k+1} \leftarrow V^*_{k+1}$ and $S_{k+1} \leftarrow S^*_{k+1}$.

Set $Y_{k+1} \leftarrow U_{k+1} S_{k+1} V_{k+1}^\top$.

The method that will be studied in this work is the Rank-Adaptive (often referred to as augmented) Basis-Update and Galerkin (BUG) Integrator (Algorithm 3.1), first

proposed in [5], which is the rank-adaptive version of the fixed-rank BUG integrator [7]. The rank-adaptive modification allows the algorithm to determine the optimal rank based on a given threshold automatically. The resulting matrix $Y_1 = U_1 S_1 V_1^\top$ after one step of the algorithm is an approximation of $Y(t_1)$ from (3.3).

Notice that in Algorithm 3.1 it is possible to perform the K- and L-steps in parallel. Also, notice that in the rank-adaptive setting, the number of singular values σ_i of S^*_{k+1} remaining after the truncation step determines the (adaptive) rank q_{k+1} of the matrix Y_{k+1} resulting from the k-th pass of Algorithm 3.1.

An important fact for the subsequent theory is that the spans of the matrices U^*_{k+1} and V^*_{k+1} obtained in the rank-adaptive method also contain the spans of U_k and V_k. Following [6], thus we can write

$$U^*_{k+1} = [U_k \mid U_k^+]\,, \tag{3.7}$$

where U_k^+ is composed of columns that expand the orthonormal basis spanning U_k. Much in the same manner, U^*_{k+1} can also be thought of as including and expanding upon the spans of U_{k+1} obtained with either the adaptive or non-adaptive methods. Thus, we can also write $U^*_{k+1} = [U_{k+1} \mid U^+_{k+1}]$. Analogous expressions hold for V^*_{k+1}.

Another important property of Algorithm 3.1 is

$$\|U^*_{k+1} S^*_{k+1} V^{*\top}_{k+1} - U_{k+1} S_{k+1} V_{k+1}^\top\| \leq \vartheta\,, \tag{3.8}$$

which follows directly from the construction of the truncation step.

3.3. DLRA for Machine Learning. As seen in Section 2, the trainable parameters in neural networks naturally appear as matrices. The central idea of [21] is to apply Algorithm 3.1 to (2.1), thereby leveraging the machinery of DLRA for training individual layers of DNNs. By doing so, the optimization parameters are restricted to the space of low-rank matrices and network pruning (that is, the task of finding accurate models of smaller size) is performed during training itself.

The general compatibility between DLRA and the training of DNNs can be seen when comparing the equations (2.1) and (3.2). In the notation of (3.2), the W becomes the matrix of trainable parameters of one layer of the network, while $F(W)$ becomes the negative gradient $-\nabla_W \mathcal{L}(W)$ of the objective function.

One major difference between the two methods, however, exists. As seen in Section 2, during network training, one generally uses the stochastic gradient f instead of the full gradient F. This difference gives rise to the central question of this work: how does Algorithm 3.1 behave when F is replaced by f?

We will now specify some notation. A DNN, by definition, contains multiple layers and the implementation in [21] applies Algorithm 3.1 to the parameter matrices $W^1, W^2, \cdots$ of each layer separately, cycling between the layers at each training step. In this work, we focus on training a single layer of a DNN and present results corresponding to this approach. Because of this, although the objective function $\mathcal{L}$ depends on all the parameter matrices $W^1, W^2, \cdots$, we will omit writing $\mathcal{L}(W^1, W^2, \cdots)$ and instead write $\mathcal{L}(W)$. Since the optimization of one layer requires only those gradients corresponding to the single parameter matrix in question, we will also write $\nabla\mathcal{L}$ instead of $\nabla_W \mathcal{L}$ and consider the other parameters fixed.

It is also worth noting that although we write $F(W) = -\nabla_W \mathcal{L}(W)$ and theoretically treat it as such, considerable computational gains can be made if we notice that, for example, as in the K-step of Algorithm 3.1,

$$F(K(t)V_k^\top)V_k = -\nabla_W \mathcal{L}(K(t)V_k^\top)V_k = -\nabla_K \mathcal{L}(K(t)V_k^\top)\,,$$

making the computation of the gradient with respect to the full parameter matrix W in the K-, L-, and S-steps unnecessary [21].

3.4. Optimality over the Manifold $\mathcal{M}_q$. Since we are performing optimization over manifolds of low-rank matrices, we should take an interest in the optimality conditions that exist there.

We know that for a point $Y \in \mathcal{M}_q$, a necessary first-order condition for optimality over $\mathcal{M}_q$ is given by

$$P(Y)\nabla\mathcal{L}(Y) = 0\,. \tag{3.9}$$

This follows from Proposition 3.3 and Theorem 3.4 in [20].

Thanks to the explicit expression (3.6) of $P(Y)$, we can directly obtain a somewhat weaker, but in the context of this work more accessible necessary condition of optimality

$$P_U\nabla\mathcal{L}(Y)P_V = 0\,. \tag{3.10}$$

We call this condition more accessible because expressions of the form $P_U\nabla\mathcal{L}(Y)P_V$ appear very naturally in the theory that we will tackle in Section 5.

In Section 5, we show that in the stochastic setting, the DLRA optimization algorithm can come arbitrarily close to satisfying these conditions in expectation.

4. Robustness of Stochastic DLRA. Using the notation of Section 2, consider the algorithm that results from replacing every $F(\cdot)$ in Algorithm 3.1 by $f(\cdot, \xi)$, where a single realization of ξ is used per pass in the `for`-loop. We call it the *Stochastic* Algorithm 3.1. In this section, we explore how well the resulting matrix Y_1 of one step of the Stochastic Algorithm 3.1 approximates the solution $W(t_1)$ of (3.2) at time $t_1 = t_0 + h$, $h > 0$. This extends the results found in [21], where Theorem 1 states a similar error bound for the deterministic algorithm.

We can use the manifold structure of $\mathcal{M}_q$ to investigate the stochastic gradients. Since $\mathcal{T}_Y\mathcal{M}_q$ is a subspace of $\mathbb{R}^{m\times n}$ for any $Y \in \mathcal{M}_q$, we can decompose the gradients $F(Y)$ and $f(Y)$ into components $F(Y) = M(Y) + R(Y)$ and $f(Y) = m(Y) + r(Y)$, where $M(Y)$, $m(Y) \in \mathcal{T}_Y\mathcal{M}_q$. We make the following assumptions:

ASSUMPTION 4.1. *There exists an $\varepsilon > 0$ such that $\|R(Y)\|, \|r(Y)\| \leq \varepsilon$ for all $Y \in \mathcal{M}_q$.*

ASSUMPTION 4.2. *The two functions F and f are bounded by a constant $B > 0$ and Lipschitz continuous with respect to $\|\cdot\|$. The corresponding Lipschitz constant is denoted by $L > 0$.*

ASSUMPTION 4.3. *There exists a constant $C > 0$ such that $\|F(Y) - f(Y)\| \leq C$ for all $Y \in \mathbb{R}^{m\times n}$.*

Assumption 4.1 for F states that $F(Y)$ is contained in $\mathcal{T}_Y\mathcal{M}_q$ up to a small factor and is based on empirical observations [21]. Assumption 4.2 for F is common in DLRA theory [21, 13] and smooth optimization in general. The extension of these assumptions to f can be justified by the structure of the gradients; both f and F directly inherit their properties from the loss function l, as seen in Section 2, so we can expect them to behave similarly. The bound on $\|F(Y)-f(Y)\|$ in Assumption 4.3 can be justified by the law of large numbers, as explained in Section 2. These three assumptions hold for the rest of this section.

In the following, let $W(t)$ be the solution of (3.2) with initial value W_0 and $Y_1 = U_1S_1V_1^\top$ be the resulting matrix after one step of the rank-adaptive Stochastic

Algorithm 3.1 with initial value $Y_0 \in \mathcal{M}_q$ that approximates W_0. We wish to find a robust error bound for the local error

$$\|W(t_1)-Y_1\| \leq \|W(t_1)-P_{U_1^*}W(t_1)P_{V_1^*}\|+\|P_{U_1^*}W(t_1)P_{V_1^*}-Y_1^*\|+\|Y_1^*-Y_1\|\,. \tag{4.1}$$

We know from (3.8) that the last term can be bounded by ϑ. The bounds for the other terms are given by the following lemmas. In the following, let us assume that $\|W_0 - Y_0\| \leq \delta$ holds.

LEMMA 4.4. *There exists a constant C_1 such that*

$$\|W(t_1) - P_{U_1^*}W(t_1)P_{V_1^*}\| \leq C_1 h^2 + 2(L\delta + \varepsilon + C)h + 2\delta\,,$$

where C_1 does not depend on the condition number of the coefficient matrix S.

Proof. We can write

$$\begin{aligned}\|W(t_1) - P_{U_1^*}W(t_1)P_{V_1^*}\| &\leq \|W(t_1) - P_{U_1^*}W(t_1)\| + \|P_{U_1^*}\big(W(t_1) - W(t_1)P_{V_1^*}\big)\| \\ &\leq \|W(t_1) - P_{U_1^*}W(t_1)\| + \|W(t_1) - W(t_1)P_{V_1^*}\|\,.\end{aligned}$$

First notice that

$$\|W_0 - P_{U_1^*}W_0\| \leq \|(I - P_{U_1^*})Y_0\| + \|(I - P_{U_1^*})(W_0 - Y_0)\| = \|(I - P_{U_1^*})(W_0 - Y_0)\|\,,$$

where $(I - P_{U_1^*})Y_0 = 0$ holds by construction of U_1^*. Then, using the above and Assumptions 4.1-4.3 we have

$$\begin{aligned}\|W(t_1) - P_{U_1^*}W(t_1)\| &\leq \int_{t_0}^{t_1} \|F(W(t)) - P_{U_1^*}F(W(t))\|\,dt + \delta \\ &\leq \int_{t_0}^{t_1} \|(I - P_{U_1^*})f(W(t))\|\,dt + Ch + \delta \\ &\leq \int_{t_0}^{t_1} \|(I - P_{U_1^*})f(W_0)\|\,dt + LBh^2 + Ch + \delta \\ &\leq \int_{t_0}^{t_1} \|(I - P_{U_1^*})f(Y_0)\|\,dt + (hL+1)\delta + LBh^2 + Ch \\ &\leq \int_{t_0}^{t_1} \|(I - P_{U_1^*})P(Y_0)f(Y_0)\|\,dt + h\varepsilon + (hL+1)\delta + LBh^2 + Ch \\ &= \int_{t_0}^{t_1} \|(I - P_{U_1^*})f(Y_0)P_{V_0}\|\,dt + h\varepsilon + (hL+1)\delta + LBh^2 + Ch \\ &\leq \frac{1}{h}\int_{t_0}^{t_1} \|(I - P_{U_1^*})(K(t_1)V_0^\top - Y_0)\|\,dt + c_1h^2 + h\varepsilon + (hL+1)\delta + Ch \\ &= c_1h^2 + h\varepsilon + (hL+1)\delta + Ch\end{aligned}$$

where in the last inequality we use the fact that $f(Y_0)V_0 = \dot{K}(t_0)$. The analogous derivation for the co-range proves the lemma. ∎

Lastly, we have

LEMMA 4.5. *It holds that*

$$\|P_{U_1^*}W(t_1)P_{V_1^*} - Y_1^*\| \leq 2LBh^2 + Ch + (Lh+1)\delta\,.$$

Proof. Let $\bar{Y}(t) := U_1^* S(t) V_1^{*\top}$ for $S(t)$ being the solution of the S-step of the algorithm. From the construction of U_1^* and V_1^* it follows that $\bar{Y}(t_0) = Y_0$. From the construction of the S-step, we know that

$$Y_1^* = Y_0 + \int_{t_0}^{t_1} P_{U_1^*} f(\bar{Y}(t)) P_{V_1^*} dt\,.$$

Furthermore,

$$P_{U_1^*} W_0 P_{V_1^*} - Y_0 = P_{U_1^*}(W_0 - Y_0) P_{V_1^*}\,.$$

With the above and Assumptions 4.2 and 4.3, we have

$$\begin{aligned}
\|P_{U_1^*} W(t_1) P_{V_1^*} - Y_1^*\| &\leq \int_{t_0}^{t_1} \|P_{U_1^*} F(W(t)) P_{V_1^*} - P_{U_1^*} f(\bar{Y}(t)) P_{U_1^*}\|\, dt + \delta \\
&\leq \int_{t_0}^{t_1} \|P_{U_1^*} f(W(t)) P_{V_1^*} - P_{U_1^*} f(\bar{Y}(t)) P_{U_1^*}\|\, dt + Ch + \delta \\
&\leq L \int_{t_0}^{t_1} \|W(t) - \bar{Y}(t)\|\, dt + Ch + \delta \\
&\leq L \int_{t_0}^{t_1} \int_{t_0}^{t} \|F(W(s)) - P_{U_1^*} f(\bar{Y}(s)) P_{V_1^*}\|\, ds\, dt + (Lh+1)\delta + Ch \\
&\leq 2LBh^2 + Ch + (Lh+1)\delta\,.
\end{aligned}$$

∎

Together, these lemmas yield the bound

$$\|W(t_1) - Y_1\| \leq (C_1 + 2LB)h^2 + 2(L\delta + \varepsilon + 2C)h + (3 + Lh)\delta + \vartheta\,,$$

i.e., a global error bound then directly follows from Lady Windermere's fan.

This result shows that the rank-adaptive BUG algorithm does not lose its robustness properties following a stochastic modification. Thus, we can now shift to exploring the properties of the Stochastic Algorithm 3.1 as a stochastic optimization algorithm for machine learning.

5. DLRA with Stochastic Gradients.

Recall the definition of the Stochastic Algorithm 3.1 from the previous section. In this section, we explore its properties as an optimization algorithm for machine learning. At first, we quantify how it optimizes the loss function $\mathcal{L}$. Afterward, we show its convergence properties on the task of training individual layers of deep neural networks when using stochastic gradient descent and momentum methods as solvers of the integration steps. Let Assumptions 4.1 and 4.2 hold throughout this section.

5.1. Optimization of the Loss Function.

Let Y_k be iterates generated by the Rank-Adaptive Stochastic Algorithm 3.1 for some starting point Y_0. We begin by taking a look at the structure of $P_{U_{k+1}^*} f(Y_k) P_{V_{k+1}^*}$. Using (3.7), we obtain

$$P_{U_{k+1}^*} = U_{k+1}^* U_{k+1}^{*\top} = U_k U_k^\top + U_k^+ U_k^{+\top} = P_{U_k} + P_{U_k^+}\,.$$

The same also holds for V_{k+1}^*. Thus, we can write

$$\begin{aligned}
&P_{U_{k+1}^*} f(Y_k) P_{V_{k+1}^*} = \\
&\quad P_{U_k} f(Y_k) P_{V_k} + P_{U_k^+} f(Y_k) P_{V_k} + P_{U_k} f(Y_k) P_{V_k^+} + P_{U_k^+} f(Y_k) P_{V_k^+}\,.
\end{aligned} \tag{5.1}$$

This form will help us to prove the next theorem.

THEOREM 5.1. *If Y_1 is the resulting matrix after one pass of the Rank-Adaptive Stochastic Algorithm 3.1, with $t_1 = t_0 + h$ for $h > 0$ and some starting point $Y_0 \in \mathcal{M}_{q_0}$, then there exists a positive constant α such that*

$$\mathbb{E}[\mathcal{L}(Y_1)] \leq \mathcal{L}(Y_0) - h\alpha^2 + c_1 h^2 B^2 + c_2 hB\varepsilon + c_3 h^2 B^2 + B\vartheta\,,$$

where c_1, c_2, and c_3 are independent of the low-rank manifold's curvature.

Proof. Consider $\bar{Y}(t) \coloneqq U_1^* S(t) V_1^{*\top}$, where $S(t)$ denotes the solution of the S-step of the Stochastic Algorithm 3.1. Then

$$\dot{\bar{Y}}(t) = U_1^* \dot{S}(t) V_1^{*\top} = U_1^* U_1^{*\top} f(\bar{Y}(t)) V_1^* V_1^{*\top}\,.$$

By (3.7), the ranges of U_1^* and V_1^* contain those of U_0 and V_0. Hence, we have

$$\bar{Y}(t_0) = U_1^* S(t_0) V_1^{*\top} = U_1^* U_1^{*\top} U_0 S_0 V_0^\top V_1^* V_1^{*\top} = U_0 S_0 V_0^\top = Y_0\,.$$

As a consequence,

$$\bar{Y}(t_1) = Y_0 + \int_{t_0}^{t_1} P_{U_1^*} f(\bar{Y}(s)) P_{V_1^*} ds\,. \tag{5.2}$$

Using equations (5.1) and (5.2), we obtain

$$\begin{aligned}
\frac{d}{dt}\mathcal{L}(\bar{Y}(t)) &= \langle \nabla\mathcal{L}(\bar{Y}(t)), \dot{\bar{Y}}(t)\rangle = -\langle F(\bar{Y}(t)), P_{U_1^*} f(\bar{Y}(t)) P_{V_1^*}\rangle \\
&\leq -\langle F(Y_0), P_{U_1^*} f(Y_0) P_{V_1^*}\rangle + c_1 hB^2 \\
&= -\langle F(Y_0), P_{U_1^*} f(Y_0) P_{V_0} + P_{U_0} f(Y_0) P_{V_1^*} - P_{U_0} f(Y_0) P_{V_0} + P_{U_0^+} f(Y_0) P_{V_0^+}\rangle \\
&\quad + c_1 hB^2\,.
\end{aligned} \tag{5.3}$$

The last term in the scalar product can be bounded by

$$|\langle F(Y_0), P_{U_0^+} f(Y_0) P_{V_0^+}\rangle| \leq |\langle F(Y_0), P_{U_0^+} P(Y_0) f(Y_0) P_{V_0^+}\rangle| + c_2 B\varepsilon = c_2 B\varepsilon\,.$$

We use the structure of the K- and L-steps to bound the remaining terms in (5.3). The ranges of both $K(t_1)$ and K_0 are spanned by the columns of U_1^*. Therefore, $(I - P_{U_1^*})K(t_1) = (I - P_{U_1^*})K_0 = 0$ holds. Combining this equality with

$$K(t_1) = K_0 + \int_{t_0}^{t_1} f(K(s)V_0^\top) V_0 ds$$

yields

$$\int_{t_0}^{t_1} P_{U_1^*} f(K(s)V_0^\top) V_0 ds = \int_{t_0}^{t_1} f(K(s)V_0^\top) V_0 ds\,. \tag{5.4}$$

A symmetric statement also holds for the L-step yielding

$$\begin{aligned}
\|P_{U_1^*} f(Y_0) P_{V_0} - f(Y_0) P_{V_0}\| &\leq \frac{1}{2} c_3 hB\,, \\
\|P_{V_1^*} f(Y_0)^\top P_{U_0} - f(Y_0)^\top P_{U_0}\| &\leq \frac{1}{2} c_3 hB\,.
\end{aligned}$$

Hence, (5.3) becomes with $\mu := c_1 h B^2 + c_2 B\varepsilon + c_3 h B^2$

$$\frac{d}{dt}\mathcal{L}(\bar{Y}(t)) \leq -\langle F(Y_0), f(Y_0)P_{V_0} + P_{U_0}f(Y_0) - P_{U_0}f(Y_0)P_{V_0}\rangle + \mu\,.$$

Thus it holds that

$$\begin{aligned}\mathbb{E}\Big[\frac{d}{dt}\mathcal{L}(\bar{Y}(t))\Big] &\leq -\langle F(Y_0), F(Y_0)P_{V_0} + P_{U_0}F(Y_0) - P_{U_0}F(Y_0)P_{V_0}\rangle + \mu \\ &= -\|U_0^\top F(Y_0)\|^2 - \|F(Y_0)V_0\|^2 + \|U_0^\top F(Y_0)V_0\|^2 + \mu \\ &\leq -\frac{1}{2}(\|U_0^\top F(Y_0)\|^2 + \|F(Y_0)V_0\|^2) + \mu\,.\end{aligned}$$

Now let $\alpha^2 \coloneqq \frac{1}{2}(\|U_0^\top F(Y_0)\|^2 + \|F(Y_0)V_0\|^2)$. Then we get

$$\mathbb{E}\Big[\frac{d}{dt}\mathcal{L}(\bar{Y}(t))\Big] \leq -\alpha^2 + \mu\,.$$

Integrating this equation and using Fubini's theorem (which applies since the integrand $\frac{d}{dt}\mathcal{L}(\bar{Y}(t))$ is bounded) yields

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\bar{Y}(t_1))] &= \mathbb{E}[\mathcal{L}(\bar{Y}(t_0))] + \mathbb{E}\Big[\int_{t_0}^{t_1}\frac{d}{dt}\mathcal{L}(\bar{Y}(t))dt\Big] \\ &= \mathcal{L}(Y_0) + \int_{t_0}^{t_1}\mathbb{E}\Big[\frac{d}{dt}\mathcal{L}(\bar{Y}(t))\Big]dt \leq \mathcal{L}(Y_0) - h\alpha^2 + h\mu\,. \qquad (5.5)\end{aligned}$$

Since $S(t_1) = S_1^*$, we can write $\bar{Y}(t_1) = U_1^* S_1^* V_1^{*\top}$. Consequently, by (3.8),

$$\|Y_1 - \bar{Y}(t_1)\| \leq \vartheta\,. \qquad (5.6)$$

By Taylor, there exists a $\tau \in [0,1]$, such that

$$\mathcal{L}(Y_1) = \mathcal{L}(\bar{Y}(t_1)) - \langle F(\tau Y_1 + (1-\tau)\bar{Y}(t_1)), Y_1 - \bar{Y}(t_1)\rangle\,.$$

Applying the Cauchy-Schwarz inequality and (5.6) to the above we obtain

$$\mathcal{L}(Y_1) \leq \mathcal{L}(\bar{Y}(t_1)) + B\vartheta\,. \qquad (5.7)$$

Putting (5.5) and (5.7) together and taking the expected value yields

$$\mathbb{E}[\mathcal{L}(Y_1)] \leq \mathbb{E}[\mathcal{L}(\bar{Y}(t_1))] + B\vartheta \leq \mathcal{L}(Y_0) - h\alpha^2 + h\mu + B\vartheta\,. \qquad \square$$

This theorem expresses an upper bound on the expected value of the loss function $\mathcal{L}$ after one step of the Stochastic Algorithm 3.1 in terms of the free parameters h and ϑ. By construction, it holds that $\alpha \leq B$, so this bound might actually be larger than $\mathcal{L}(Y_0)$. This is somewhat unsatisfactory in the context of minimization. We will see later, however, that practical modifications of this algorithm are nonetheless capable of assuring descent, given an appropriate choice of the step size.

5.2. Stochastic Gradient Descent. Until now, we have assumed that the K-, L- and S-steps of Algorithm 3.1 are solved exactly. In practice, however, they are solved using various discrete-time methods, such as (stochastic) gradient descent and Adam [21, 3]. In this section, we will investigate the convergence properties of

the Stochastic Algorithm 3.1, where the continuous time variable t is replaced by a discrete one ($t = 0, 1, \ldots$) and the integration steps are replaced by the gradient descent algorithm, generally expressed as

$$Z_{t+1} = Z_t - h\nabla\mathcal{G}(Z_t)$$

for a step size $h > 0$, a smooth function $\mathcal{G}$, and iterates Z_t. More precisely, the integration in the K-step is replaced by

$$K_{t+1} = U_t S_t + hf(U_t S_t V_t^\top)V_t\,, \tag{5.8}$$

in the L-step by

$$L_{t+1} = V_t S_t^\top + hf(U_t S_t V_t^\top)^\top U_t\,,$$

and in the S-step by

$$S^*_{t+1} = U^{*\top}_{t+1} U_t S_t V_t^\top V^*_{t+1} + hU^{*\top}_{t+1} f(P_{U^*_{t+1}} U_t S_t V_t^\top P_{V^*_{t+1}}) V^*_{t+1}\,. \tag{5.9}$$

Once again, a single realization of ξ is used to compute f for all these steps within one pass of the `for`-loop of the algorithm. In this section, we will refer to this modification of the Stochastic Algorithm 3.1 as the *Stochastic Gradient Descent (SGD)* Algorithm 3.1. This modification can also be seen as solving the differential equations in the K-, L-, and S-steps with the explicit Euler method.

The logic in this section generally follows from adapting the treatment of SGD found in Chapter 4 of [3] to the DLRA setting.

LEMMA 5.2. *For any Y, $\bar{Y}$ it holds that*

$$\mathcal{L}(Y) \leq \mathcal{L}(\bar{Y}) - \langle F(\bar{Y}), Y - \bar{Y}\rangle + \frac{L}{2}\|Y - \bar{Y}\|^2\,. \tag{5.10}$$

Proof.

$$\begin{aligned}
\mathcal{L}(Y) &= \mathcal{L}(\bar{Y}) + \int_0^1 \frac{d}{dt}\mathcal{L}(\bar{Y} + t(Y - \bar{Y}))dt = \mathcal{L}(\bar{Y}) - \int_0^1 \langle F(\bar{Y} + t(Y - \bar{Y})), Y - \bar{Y}\rangle dt \\
&= \mathcal{L}(\bar{Y}) - \langle F(\bar{Y}), Y - \bar{Y}\rangle - \int_0^1 \langle F(\bar{Y} + t(Y - \bar{Y})) - F(\bar{Y}), Y - \bar{Y}\rangle dt \\
&\leq \mathcal{L}(\bar{Y}) - \langle F(\bar{Y}), Y - \bar{Y}\rangle + \int_0^1 Lt\|Y - \bar{Y}\|^2 dt \\
&= \mathcal{L}(\bar{Y}) - \langle F(\bar{Y}), Y - \bar{Y}\rangle + \frac{L}{2}\|Y - \bar{Y}\|^2\,,
\end{aligned}$$

where in the first line we use the chain rule and $F = -\nabla\mathcal{L}$ and in the third line the Cauchy-Schwarz inequality. ∎

Now, let $Y_0 = U_0 S_0 V_0^\top \in \mathcal{M}_{q_0}$ be some fixed initial value for the SGD Algorithm 3.1. Let Y_k be iterates generated by this algorithm.

THEOREM 5.3. *If Y_1 is the resulting matrix after one pass of the Rank-Adaptive SGD Algorithm 3.1, then it holds that*

$$\mathbb{E}[\mathcal{L}(Y_1)] \leq \mathcal{L}(Y_0) - h\|P(Y_0)F(Y_0)\|^2 + \frac{1}{2}h^2 LB^2 + hB\varepsilon + B\vartheta\,.$$

Proof. Consider $Y_1^* = U_1^* S_1^* V_1^{*\top}$. By the definition of the S-step (5.9), we can write

$$S_1^* = U_1^{*\top} U_0 S_0 V_0^\top V_1^* + h U_1^{*\top} f(P_{U_1^*} U_0 S_0 V_0^\top P_{V_1^*}) V_1^* \,.$$

Using (3.7), we obtain

$$U_1^* U_1^{*\top} U_0 S_0 V_0^\top V_1^* V_1^{*\top} = P_{U_1^*} U_0 S_0 V_0^\top P_{V_1^*} = U_0 S_0 V_0^\top = Y_0 \,.$$

Combining the above equations yields

$$Y_1^* = Y_0 + h P_{U_1^*} f(Y_0) P_{V_1^*} \,.$$

Using Lemma 5.2 and equation (5.1) we obtain

$$\begin{aligned}
\mathcal{L}(Y_1^*) - \mathcal{L}(Y_0) \leq & -h\langle F(Y_0), P_{U_1^*} f(Y_0) P_{V_1^*} \rangle + \frac{h^2 L}{2} \| P_{U_1^*} f(Y_0) P_{V_1^*} \|^2 \\
= & -h\langle F(Y_0), P_{U_0} f(Y_0) P_{V_0} + P_{U_0^+} f(Y_0) P_{V_0} + P_{U_0} f(Y_0) P_{V_0^+} \rangle \\
& -h\langle F(Y_0), P_{U_0^+} f(Y_0) P_{V_0^+} \rangle + \frac{h^2 L}{2} \| P_{U_1^*} f(Y_0) P_{V_1^*} \|^2 \,. \qquad (5.11)
\end{aligned}$$

By Assumption 4.2 we have

$$\frac{h^2 L}{2} \| P_{U_1^*} f(Y_0) P_{V_1^*} \|^2 \leq \frac{1}{2} h^2 L B^2 \,. \qquad (5.12)$$

Furthermore, by applying the logic preceding (5.4) to the new K-step (5.8), we obtain

$$P_{U_1^*} f(Y_0) P_{V_0} = f(Y_0) P_{V_0} \,.$$

Combining this with a symmetric argument on the L-step yields

$$\begin{aligned}
& P_{U_0} f(Y_0) P_{V_0} + P_{U_0^+} f(Y_0) P_{V_0} + P_{U_0} f(Y_0) P_{V_0^+} \\
& \quad = P_{U_1^*} f(Y_0) P_{V_0} + P_{U_0} f(Y_0) P_{V_1^*} - P_{U_0} f(Y_0) P_{V_0} \\
& \quad = f(Y_0) P_{V_0} + P_{U_0} f(Y_0) - P_{U_0} f(Y_0) P_{V_0} = P(Y_0) f(Y_0) \,. \qquad (5.13)
\end{aligned}$$

The first term in (5.11), $P_{U_0^+} f(Y_0) P_{V_0^+}$, can be bounded using Assumption 4.1. Since we know that $f(Y_0) = m(Y_0) + r(Y_0)$ with $m(Y_0) \in \mathcal{T}_{Y_0}\mathcal{M}_{q_0}$, the definition of U_0^+ and V_0^+ and the Assumption 4.1 allow us to write

$$\| P_{U_0^+} f(Y_0) P_{V_0^+} \| = \| P_{U_0^+} r(Y_0) P_{V_0^+} \| \leq \varepsilon \,. \qquad (5.14)$$

Applying (5.12), (5.13), and (5.14) onto (5.11) and using the Cauchy-Schwarz inequality as well as Assumption 4.2 yields

$$\mathcal{L}(Y_1^*) - \mathcal{L}(Y_0) \leq \langle F(Y_0), P(Y_0) f(Y_0) \rangle + \frac{1}{2} h^2 L B^2 + h B \varepsilon \,.$$

Taking the expected value, this becomes

$$\mathbb{E}\big[\mathcal{L}(Y_1^*)\big] - \mathcal{L}(Y_0) \leq -h \| P(Y_0) F(Y_0) \|^2 + \frac{1}{2} h^2 L B^2 + h B \varepsilon \,.$$

Furthermore, by (3.8), we know that

$$\|Y_1 - Y_1^*\| \leq \vartheta\,.$$

Thus, using Taylor, for some $\tau \in [0,1]$, we get

$$\mathcal{L}(Y_1) = \mathcal{L}(Y_1^*) - \langle F(\tau Y_1 + (1-\tau)Y_1^*), Y_1 - Y_1^* \rangle \leq \mathcal{L}(Y_1^*) + B\vartheta\,.$$

Putting everything together and taking the expected value yields

$$\mathbb{E}[\mathcal{L}(Y_1)] \leq \mathbb{E}[\mathcal{L}(Y_1^*)] + B\vartheta \leq \mathcal{L}(Y_0) - h\|P(Y_0)F(Y_0)\|^2 + \frac{1}{2}h^2LB^2 + hB\varepsilon + B\vartheta\,. \qquad \square$$

Whenever we write $f(Y) = f(Y,\xi)$, the randomness is hidden in the term ξ. Thus, when reading $\mathbb{E}[f(Y_0)]$, one should understand $\mathbb{E}_\xi[f(Y_0,\xi)]$. The matrix Y_1 and its factors U_1, V_1, and S_1 are stochastic in ξ, since they have been generated using $f(Y_0,\xi)$. Therefore, $\mathbb{E}[\mathcal{L}(Y_1)] = \mathbb{E}_\xi[\mathcal{L}(Y_1(\xi))]$ holds. In general, when considering the evolution of Y_t up to the k-th step, if we denote with $\xi_1,\dots,\xi_k$ the i.i.d. realizations of ξ made in each pass of the algorithm, we can say that Y_t, U_t, V_t, and S_t are stochastic in $\xi_1,\dots,\xi_t$, so the simplified notation $\mathbb{E}[f(Y_t)]$ corresponds to $\mathbb{E}_{\xi_1,\dots,\xi_{t+1}}[f(Y_t(\xi_1,\dots,\xi_t),\xi_{t+1})]$.

Thus, we can rewrite the result of Theorem 5.3 in a more general manner:

$$\mathbb{E}_{\xi_t}[\mathcal{L}(Y_t(\xi_t))] \leq \mathcal{L}(Y_{t-1}) - h\|P(Y_{t-1})F(Y_{t-1})\|^2 + \frac{1}{2}h^2LB^2 + hB\varepsilon + B\vartheta\,. \tag{5.15}$$

Now, we can make statements about the behavior of the algorithm as $t \to \infty$.

THEOREM 5.4. *Let $\mathcal{L}$ be non-negative. Let $Y_1,\dots,Y_k$ be iterates generated by the Rank-Adaptive SGD Algorithm 3.1 over k steps. Then it holds that*

$$\frac{1}{k}\sum_{t=1}^{k}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] \leq$$

$$\frac{\mathcal{L}(Y_0)}{kh} + \frac{1}{2}hLB^2 + B\varepsilon + \frac{1}{h}B\vartheta \xrightarrow{k\to\infty} \frac{1}{2}hLB^2 + B\varepsilon + \frac{1}{h}B\vartheta\,, \tag{5.16}$$

where the expected value is taken over all ξ_t.

Proof. By taking the expected value over all ξ_t in (5.15), we get

$$\mathbb{E}[\mathcal{L}(Y_t)] - \mathbb{E}[\mathcal{L}(Y_{t-1})] \leq -h\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] + \frac{1}{2}h^2LB^2 + hB\varepsilon + B\vartheta\,.$$

Using $\mathcal{L} \geq 0$, we can now conclude

$$\begin{aligned} -\mathcal{L}(Y_0) &\leq \mathbb{E}[\mathcal{L}(Y_k)] - \mathcal{L}(Y_0) \\ &\leq -h\sum_{t=1}^{k}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] + k\Big(\frac{1}{2}h^2LB^2 + hB\varepsilon + B\vartheta\Big)\,. \end{aligned}$$

Rearranging the terms, we obtain

$$\begin{aligned} \sum_{t=1}^{k}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] &\leq \frac{1}{h}\Big(\mathcal{L}(Y_0) + k\Big(\frac{1}{2}h^2LB^2 + hB\varepsilon + B\vartheta\Big)\Big) \\ &= \frac{\mathcal{L}(Y_0)}{h} + k(\frac{1}{2}hLB^2 + B\varepsilon + \frac{1}{h}B\vartheta)\,. \end{aligned}$$

Dividing by k and taking the limit yields the desired result. $\square$

Theorem 5.4 states that the running average of the expected squared norms of the projected gradients of $\mathcal{L}$ does not surpass $\frac{1}{2}hLB^2 + B\varepsilon + \frac{1}{h}B\vartheta$. If this bound could be made arbitrarily small, we could claim convergence towards a stationary point. In the setting of the usual stochastic gradient descent algorithm such as in Section 4 of [3], this bound is linear in h. This can be used to obtain

$$\liminf_{t\to\infty} \mathbb{E}\big[\|P(Y_t)F(Y_t)\|^2\big] = 0\,,$$

by using a variable and shrinking step size $h = h_t$.

Two terms are preventing us from taking this approach here. Firstly, $B\varepsilon$ is constant in h, and reducing the step size would not affect it. This term stems from the bound (5.14) we use on terms of the form $\langle F(Y_t), P_{U_t^+} f(Y_t) P_{V_t^+}\rangle$, which we cannot easily integrate because both the projections $P_{U_t^+}$ and the gradients $f(Y_t)$ are stochastic in ξ_{t+1}.

The other even more problematic term in (5.16) is $\frac{1}{h}B\vartheta$, as it is inversely proportional to h. This term appears because the truncation step is independent of the learning rate h and the stochastic gradient f. All the steps preceding the truncation move the objective towards a (stochastic) decrease, while the truncation can seemingly throw it off in any direction at step distance ϑ. If the learning rate h is made smaller, this truncation displacement becomes increasingly dominant in the progress of the algorithm.

We generally cannot expect ϑ to be smaller than h, see, e.g., the values in Section 5 of [21]. Letting $\vartheta \to 0$ would defeat the purpose of rank reduction since, over many algorithm passes, this might yield matrices of high or even full rank. Notice that at every pass of Rank-Adaptive Algorithm 3.1, the rank is initially increased by up to two times with respect to the rank in the previous pass, and only those dimensions that have singular values below the threshold are later removed; if the threshold goes to 0, fewer dimensions are removed at each pass.

Interestingly, rank-adaptivity seems to be less impactful in the later stages of training. The experiments in [21] indicate that the ranks of the parameter matrices become close to constant after sufficiently many training steps. Thus, a sensible solution would be to perform rank-adaptivity only at the beginning of model training and afterward continue in a non-adaptive manner with a shrinking h_t. This being said, such an approach makes it challenging to obtain a theoretical result akin to the ones above since the proofs depend on the property that the ranges of U_t and V_t are contained in those of U_{t+1}^* and V_{t+1}^*. This property does not generally hold in the non-adaptive setting.

We avoid this issue by exploring a method called *S-fine-tuning*. This method has been implemented in the source code [22] released alongside [21], which consists of dropping the K- and L-steps and only performing S-steps. S-fine-tuning is performed after the model has been trained for several epochs (see the `train_and_finetune` function in the `DLRT-Net/optimizer_KLS/train_experiments.py` scrip of the source code). It seems to rely on the assumption that, in the later stages of model training, the computational costs related to calculating the K- and L-steps outweigh the gain in precision obtained from updating the U and V matrices.

Performing S-fine-tuning can be seen as assuming that the ranges of the solutions K_t of the K-steps remain constant. In other words, we get

$$(I - P_U)K_t = 0$$

for all t. An analogous interpretation also holds for the L-step. Just as in (5.13), applying this to the definition of the K- and L-steps yields

$$P(Y_t)f(Y_t) = P_U f(Y_t) P_V \,.$$

We can now state our assumption.

ASSUMPTION 5.5. *There exists an index t_0 such that for all $t \geq t_0$*

1. $U_t = U_{t-1}$ *and* $V_t = V_{t-1}$,
2. $P(Y_t)f(Y_t) = P_{U_t} f(Y_t) P_{V_t}$.

Remark 5.6. When using this assumption, we drop the indices in our notation of the U and V matrices. This assumption implies that, after the index t_0, we not only use the non-adaptive method, which makes the term $\frac{1}{h} B\vartheta$ disappear, but also no longer perform the K- and L-steps. It follows immediately that $U_t^+ = V_t^+ = 0$ for all $t \geq t_0$, so the term $B\varepsilon$ must also disappear by (5.1) and the discussion above.

THEOREM 5.7. *In the setting of Theorem 5.3, let Assumption 5.5 hold. Then*

$$\mathbb{E}_{\xi_t}[\mathcal{L}(Y_t)] \leq \mathcal{L}(Y_{t-1}) - h\|P(Y_{t-1})F(Y_{t-1})\|^2 + \frac{1}{2}h^2 LB^2 \tag{5.17}$$

for all $t \geq t_0$ for t_0 from Assumption 5.5.

Proof. For $t \geq t_0$, by the definition of the S-step (5.9),

$$S_t = U^\top U S_{t-1} V^\top V + hU^\top f(P_U U S_{t-1} V^\top P_V) V = S_{t-1} + hU^\top f(Y_{t-1}) V \,,$$

so

$$Y_t = U S_t V^\top = Y_{t-1} + h P_U f(Y_{t-1}) P_V = Y_{t-1} + hP(Y_{t-1}) f(Y_{t-1}) \,.$$

Using Lemma 5.2 and Assumption 4.2, we can write

$$\begin{aligned} \mathcal{L}(Y_t) - \mathcal{L}(Y_{t-1}) &\leq -h\langle F(Y_{t-1}), P(Y_{t-1}) f(Y_{t-1})\rangle + \frac{h^2 L}{2} \|P(Y_{t-1}) f(Y_{t-1})\|^2 \\ &\leq -h\langle F(Y_{t-1}), P(Y_{t-1}) f(Y_{t-1})\rangle + \frac{1}{2} h^2 LB^2 \,. \end{aligned}$$

Taking the expected value with respect to ξ_t yields the desired result. ∎

Remark 5.8. In particular, (5.17) shows that, for a small enough h, the algorithm assures a decrease of the loss function in expectation.

The proof of Theorem 5.4 where (5.15) is replaced by (5.17) immediately yields, up to a shift in the indices such that $t_0 = 0$,

$$\frac{1}{k} \sum_{t=1}^{k} \mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] \leq \frac{\mathcal{L}(Y_0)}{kh} + \frac{1}{2} hLB^2 \xrightarrow{k\to\infty} \frac{1}{2} hLB^2 \,. \tag{5.18}$$

We can see that Assumption 5.5 has removed both problematic terms in equation (5.16). It is clear that the right-hand side of (5.18) goes to 0 if we let $h \to 0$. To achieve this, let us now choose a variable step size $h = h_t$ that satisfies

$$\sum_{t=0}^{\infty} h_t = \infty \text{ and } \sum_{t=0}^{\infty} h_t^2 < \infty \,. \tag{5.19}$$

THEOREM 5.9. *In the setting of Theorem 5.4, let Assumption 5.5 hold and a variable step size h_t satisfy* (5.19)*. Then one has*

$$\sum_{t=1}^{\infty} h_{t-1}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] < \infty\,. \tag{5.20}$$

Proof. Without loss of generality let $t_0 = 0$ in Assumption 5.5, otherwise set the index after which this holds to 0.

Using (5.17),

$$\mathbb{E}[\mathcal{L}(Y_t)] - \mathbb{E}[\mathcal{L}(Y_{t-1})] \leq -h_{t-1}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] + \frac{1}{2}h_{t-1}^2 LB^2\,.$$

Similarly to the proof of Theorem 5.4, we obtain

$$-\mathcal{L}(Y_0) \leq \mathbb{E}[\mathcal{L}(Y_k)] - \mathcal{L}(Y_0) \leq -\frac{1}{2}\sum_{t=1}^{k} h_{t-1}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] + \frac{LB^2}{2}\sum_{t=1}^{k} h_{t-1}^2\,.$$

Rearranging the terms,

$$\sum_{t=1}^{k} h_{t-1}\mathbb{E}\big[\|P(Y_{t-1})F(Y_{t-1})\|^2\big] \leq 2\mathcal{L}(Y_0) + LB^2\sum_{t=1}^{k} h_{t-1}^2\,.$$

Taking the limit $k \to \infty$ and using $\sum_{t=0}^{\infty} h_t^2 < \infty$ yields the result. ∎

COROLLARY 5.10. *In the setting of Theorem 5.9, it holds that*

$$\liminf_{t\to\infty} \mathbb{E}\big[\|P(Y_t)F(Y_t)\|^2\big] = 0\,. \tag{5.21}$$

Proof. The statement follows directly from (5.20) and (5.19). ∎

This shows that in the setting of Theorem 5.9, the SGD Algorithm 3.1 yields a sequence of iterates, such that a subsequence comes arbitrarily close to satisfying the necessary condition for optimality (3.9) in expectation. If Assumption 5.5.2 is dropped, the same result holds, albeit for the weaker necessary condition of optimality (3.10).

This convergence result indicates that the SGD Algorithm 3.1 is a valid optimization algorithm for training individual layers of DNNs. The discussion in this section also suggests that the best way to apply it would be to first train the model in a rank-adaptive manner and, once the ranks of the S matrices have stabilized, finish the training using S-fine-tuning.

5.3. Momentum Methods. Although stochastic gradient descent has an important place among optimization methods for machine learning, in practice, it has been largely outperformed by momentum methods [3, 25, 10]. These methods do not simply use the gradient of the current step but rather the accumulated gradient information from all previous steps [3, 25]. Some common examples are the heavy ball and Nesterov methods. In this section, we will use a momentum method as a solver of the differential equations that constitute the Stochastic Algorithm 3.1 and will investigate the convergence properties of the resulting algorithm.

For f as in the previous sections, consider the two-step algorithm

$$\text{(SUM)}: \begin{cases} X_t = \mu X_{t-1} + h_t f(Y_t)\,, \\ Y_{t+1} = Y_t + \lambda h_t f(Y_t) + (1 - \lambda + \lambda\mu)X_t \end{cases} \tag{5.22}$$

with parameters $\mu \in [0,1)$, $\lambda \in [0, \frac{1}{1-\mu}]$ and step sizes h_t, that generates a sequence of iterates Y_t from starting values Y_0 and $X_0 \coloneqq 0$. This is the *Stochastic Unified Momentum (SUM)* algorithm proposed in [18], which generalizes the stochastic heavy ball ($\lambda = 0$) and Nesterov ($\lambda = 1$) methods. When applied as an integrator of the S-step of the Stochastic Algorithm 3.1 once again replacing the continuous time variable t by a discrete one, it becomes

$$
\begin{aligned}
X_t &= \mu X_{t-1} + h_t U_{t+1}^{*\top} f(P_{U_{t+1}^*} U_t S_t V_t^\top P_{V_{t+1}^*}) V_{t+1}^* \,, \\
S_{t+1}^* &= U_{t+1}^{*\top} U_t S_t V_t^\top V_{t+1}^* + \lambda h_t U_{t+1}^{*\top} f(P_{U_{t+1}^*} U_t S_t V_t^\top P_{V_{t+1}^*}) V_{t+1}^* \\
&\quad + (1 - \lambda + \lambda\mu) X_t \,.
\end{aligned} \tag{5.23}
$$

This algorithm is not immediately applicable in the rank-adaptive setting since the dimension of X_{t-1} and that of the stochastic gradient

$$
U_{t+1}^{*\top} f(P_{U_{t+1}^*} U_t S_t V_t^\top P_{V_{t+1}^*}) V_{t+1}^*
$$

are not necessarily the same at any given step.

In practice, this issue can be circumvented by using heuristics. For example, in the source code of [21], when applying the Adam algorithm, which also uses momentum and thus suffers from the same issue [10], in the rank-adaptive setting, the dimensions of X_t and S_t are kept constant and set to the largest possible value, while only *submatrices* of X_t and S_t with appropriate dynamical dimensions are being used and updated.

Such approaches do not correspond to the SUM algorithm (5.22) that we want to study. Luckily, the experimental findings in [21] suggest that even when using these heuristics, the adaptive rank of the matrices Y_t stabilizes during training. It is thus reasonable to once again use Assumption 5.5.1. Just like in the previous section, we can generally assume that $t_0 = 0$, for the number of steps t_0 after which the basis is kept fixed since we are only interested in the behavior as $t \to \infty$.

Under this assumption, we can rewrite (5.23) as

$$
\begin{aligned}
X_t &= \mu X_{t-1} + h_t U^\top f(Y_t) V \,, \\
S_{t+1} &= S_t + \lambda h_t U^\top f(Y_t) V + (1 - \lambda + \lambda\mu) X_t \,.
\end{aligned}
$$

Since there is no truncation step, multiplying by U from the left and by $V^\top$ from the right yields, with a new definition of X_t,

$$
\begin{aligned}
X_t &= \mu X_{t-1} + h_t P_U f(Y_t) P_V \,, \\
Y_{t+1} &= Y_t + \lambda h_t P_U f(Y_t) P_V + (1 - \lambda + \lambda\mu) X_t \,.
\end{aligned} \tag{5.24}
$$

Consider the modification of the non-adaptive Algorithm 3.1, where, as under Assumption 5.5.1, only the S-step is performed and is further replaced by (5.24). In this section, we will refer to this modification as the *Stochastic Unified Momentum (SUM)* Algorithm 3.1. We will now investigate its convergence properties.

Let Assumptions 4.2 and 5.5.1 hold throughout this section. Let $\{Y_t\}_{t\geq 0}$ be a sequence of iterates generated by the SUM Algorithm 3.1 for some $Y_0 \in \mathcal{M}_q$ and $\{X_t\}_{t\geq 0}$, $X_0 = 0$, be its corresponding sequence from (5.24).

The logic in this section generally follows from applying the treatment of the SUM algorithm from [18] to the DLRA setting.

First, we need to state a few technical lemmas.

LEMMA 5.11. *Let $\{a_t\}_{t\geq 1}$, $\{b_t\}_{t\geq 1}$, and $\{\tilde{a}_t\}_{t\geq 1}$ be non-negative real sequences such that $\sum_{t=1}^{\infty} a_t = \infty$, $\sum_{t=1}^{\infty} a_t b_t^2 < \infty$, $\lim_{t\to\infty} \frac{a_t}{\tilde{a}_t} = 1$, and $|b_{t+1} - b_t| \leq C\tilde{a}_t$ for a positive constant C. Then $\lim_{t\to\infty} b_t = 0$.*

Proof. This lemma is proven as Corollary 3.1 in [18]. ∎

LEMMA 5.12. *Let $\{a_t\}_{t\geq 0}$, $\{b_t\}_{t\geq 0}$, and $\{c_t\}_{t\geq 0}$ be real sequences where $\{b_t\}_{t\geq 0}$ is non-negative. Further let $a_{t+1} \leq a_t - b_t + c_t$ and $\sum_{t=0}^{\infty} c_t$ converge. Then either $\lim_{t\to\infty} a_t = -\infty$, or a_t converges and $\sum_{t=0}^{\infty} b_t < \infty$.*

Proof. This lemma is proven as Lemma 1 in [1]. ∎

LEMMA 5.13. *Let the step sizes h_t satisfy* (5.19). *Then it holds that*

1. $\sum_{t=1}^{\infty}\Big(\sum_{s=1}^{t} \mu^{t-s} h_s^2\Big) < \infty$,
2. $\sum_{k=1}^{\infty}\Big(\sum_{t=1}^{k-1} \mu^{k-t}\mathbb{E}\big[\|X_t\|^2\big]\Big) < \infty$,
3. $\sum_{t=1}^{\infty} \mathbb{E}\big[\|Y_{t+1} - Y_t\|^2\big] < \infty$.

Proof. We can rewrite X_t from (5.24) as

$$X_t = \sum_{s=1}^{t} \mu^{t-s} h_s P_U f(Y_s) P_V \,. \tag{5.25}$$

Thus, using Jensen's inequality in the second line,

$$\begin{aligned}
\mathbb{E}[\|X_t\|^2] &\leq \mathbb{E}\Big[\Big(\sum_{s=1}^{t} \mu^{t-s} h_s \|P_U f(Y_s) P_V\|\Big)^2\Big] \\
&\leq \mathbb{E}\Big[\Big(\sum_{s=1}^{t} \mu^{t-s}\Big) \sum_{s=1}^{t} \mu^{t-s} h_s^2 \|P_U f(Y_s) P_V\|^2\Big] \\
&\leq \frac{B^2}{1-\mu} \sum_{s=1}^{t} \mu^{t-s} h_s^2 \,.
\end{aligned}$$

Since $\sum_{t=1}^{\infty} h_t^2$ and $\sum_{t=1}^{\infty} \mu^t$ converge absolutely, their Cauchy product

$$\sum_{t=1}^{\infty}\Big(\sum_{s=1}^{t} \mu^{t-s} h_s^2\Big)$$

also converges. Thus

$$\sum_{t=1}^{\infty} \mathbb{E}\big[\|X_t\|^2\big] < \infty \quad \text{and} \quad \sum_{k=1}^{\infty}\Big(\sum_{t=1}^{k-1} \mu^{k-t}\mathbb{E}\big[\|X_t\|^2\big]\Big) < \infty \,, \tag{5.26}$$

where we notice that the sum in the second inequality is once again a Cauchy product of two absolutely convergent series. Using the inequality $(a+b)^2 \leq 2(a^2+b^2)$, we can also write

$$\begin{aligned}
\|Y_{t+1} - Y_t\|^2 &= \|\lambda h_t P_U f(Y_t) P_V + (1 - \lambda + \lambda\mu) X_t\|^2 \\
&\leq 2\lambda^2 h_t^2 \|P_U f(Y_t) P_V\|^2 + 2(1 - \lambda + \lambda\mu)^2 \|X_t\|^2 \qquad (5.27) \\
&\leq 2\lambda^2 h_t^2 B^2 + 2(1 - \lambda + \lambda\mu)^2 \|X_t\|^2 \,.
\end{aligned}$$

Thus, $\sum_{t=1}^{\infty} h_t^2 < \infty$ and (5.26) yield

$$\sum_{t=1}^{\infty} \mathbb{E}\big[\|Y_{t+1} - Y_t\|^2\big] < \infty\,.$$

LEMMA 5.14. *It holds that*

$$\begin{aligned} -\mathbb{E}\big[\langle F(Y_t), X_t\rangle\big] \leq &- \sum_{s=1}^{t} \mu^{t-s} h_s \mathbb{E}\big[\|P_U F(Y_s) P_V\|^2\big] \\ &+ 2L \sum_{s=1}^{t-1} \mu^{t-s} \mathbb{E}\big[\|X_s\|^2\big] + L\lambda^2 B^2 \sum_{s=1}^{t-1} \mu^{t-s} h_s^2\,. \end{aligned}$$

Proof. By the definition of X_k, we have

$$-\mathbb{E}\big[\langle F(Y_t), X_t\rangle\big] = -\mu \mathbb{E}\big[\langle F(Y_t), X_{t-1}\rangle\big] - h_t \mathbb{E}\big[\|P_U F(Y_t) P_V\|^2\big]\,.$$

Using (5.27), the inequality $ab \leq \frac{1}{2}(a^2 + b^2)$, and the Cauchy-Schwarz inequality, we obtain

$$\begin{aligned} &\langle F(Y_{t-1}) - F(Y_t), X_{t-1}\rangle \leq \|F(Y_{t-1}) - F(Y_t)\| \|X_{t-1}\| \leq L\|Y_{t-1} - Y_t\| \|X_{t-1}\| \\ &\quad \leq \frac{L}{2}\|Y_{t-1} - Y_t\|^2 + \frac{L}{2}\|X_{t-1}\|^2 \leq L\lambda^2 h_{t-1}^2 B^2 + \Big(L(1 - \lambda + \lambda\mu)^2 + \frac{L}{2}\Big)\|X_{t-1}\|^2 \\ &\quad \leq L\lambda^2 h_{t-1}^2 B^2 + 2L\|X_{t-1}\|^2\,, \end{aligned}$$

where in the last line we use the fact that $\mu - 1 < 0$ and thus $1 - \lambda + \lambda\mu \leq 1$. Ccmbining these findings yields

$$\begin{aligned} &-\mathbb{E}\big[\langle F(Y_t), X_t\rangle\big] = -\mu \mathbb{E}\big[\langle F(Y_t) + F(Y_{t-1}) - F(Y_{t-1}), X_{t-1}\rangle\big] - h_t \mathbb{E}\big[\|P_U F(Y_t) P_V\|^2\big] \\ &\quad \leq -\mu \mathbb{E}\big[\langle F(Y_{t-1}), X_{t-1}\rangle\big] - h_t \mathbb{E}[\|P_U F(Y_t) P_V\|^2] + \mu L \lambda^2 h_{t-1}^2 B^2 + 2\mu L \mathbb{E}\big[\|X_{t-1}\|^2\big] \\ &\quad \leq -\sum_{s=1}^{t} \mu^{t-s} h_s \mathbb{E}\big[\|P_U F(Y_s) P_V\|^2\big] + 2L \sum_{s=1}^{t-1} \mu^{t-s} \mathbb{E}\big[\|X_s\|^2\big] + L\lambda^2 B^2 \sum_{s=1}^{t-1} \mu^{t-s} h_s^2\,. \end{aligned}$$

We can now show the central result of this section.

THEOREM 5.15. *Let $\mathcal{L}$ be non-negative and the step sizes h_t be such that* (5.19) *as well as* $\lim_{t\to\infty} \frac{h_{t-1}}{h_t} = 1$ *hold. Then*

$$\lim_{t\to\infty} \mathbb{E}\big[\|P_U F(Y_t) P_V\|\big] = 0\,. \tag{5.28}$$

Proof. Putting Lemmas 5.2 and 5.14 together, we can write

$$\begin{aligned} &\mathbb{E}[\mathcal{L}(Y_{t+1})] - \mathbb{E}[\mathcal{L}(Y_t)] \\ &\leq -\mathbb{E}\big[\langle F(Y_t), \lambda h_t P_U f(Y_t) P_V + (1 - \lambda + \lambda\mu) X_t\rangle\big] + \frac{L}{2}\mathbb{E}\big[\|Y_{t+1} - Y_t\|^2\big] \\ &\leq -\lambda h_t \mathbb{E}\big[\|P_U F(Y_t) P_V\|^2\big] + \frac{L}{2}\mathbb{E}\big[\|Y_{t+1} - Y_t\|^2\big] \\ &\quad + (1 - \lambda + \lambda\mu)\Big(2L \sum_{s=1}^{t-1} \mu^{t-s} \mathbb{E}\big[\|X_s\|^2\big] + L\lambda^2 B^2 \sum_{s=1}^{t-1} \mu^{t-s} h_s^2 - \\ &\qquad\qquad \sum_{s=1}^{t} \mu^{t-s} h_s \mathbb{E}\big[\|P_U F(Y_s) P_V\|^2\big]\Big)\,. \end{aligned} \tag{5.29}$$

Consider the sequences

$$a_t \coloneqq \mathbb{E}[\mathcal{L}(Y_t)]\,,$$

$$b_t \coloneqq \lambda h_t \mathbb{E}\big[\|P_U F(Y_t) P_V\|^2\big] + (1-\lambda+\lambda\mu)\sum_{s=1}^{t} \mu^{t-s} h_s \mathbb{E}\big[\|P_U F(Y_s) P_V\|^2\big]\,,$$

$$c_t \coloneqq \frac{L}{2}\mathbb{E}\big[\|Y_{t+1}-Y_t\|^2\big] + (1-\lambda+\lambda\mu)\Big(2L\sum_{s=1}^{t-1}\mu^{t-s}\mathbb{E}\big[\|X_s\|^2\big] + L\lambda^2 B^2 \sum_{s=1}^{t-1}\mu^{t-s}h_s^2\Big)\,.$$

The b_t are non-negative by definition of λ and μ. By (5.29) we know that $a_{t+1} \leq a_t - b_t + c_t$. Also, from Lemma 5.13 it follows that $\sum_{t=0}^{\infty} c_t$ converges, so Lemma 5.12 applies to these sequences. By the non-negativity of $\mathcal{L}$, this yields $a_t \to a_* < \infty$ and

$$\sum_{t=1}^{\infty}\Big(\lambda h_t \mathbb{E}\big[\|P_U F(Y_t) P_V\|^2\big] + (1-\lambda+\lambda\mu)\sum_{s=1}^{t}\mu^{t-s}h_s\mathbb{E}\big[\|P_U F(Y_s)P_V\|^2\big]\Big) \tag{5.30}$$

$$= \sum_{t=1}^{\infty} b_t < \infty\,.$$

In particular, since for any k it holds that

$$\sum_{t=1}^{k}\sum_{s=1}^{t}\mu^{t-s}h_s\mathbb{E}\big[\|P_U F(Y_s)P_V\|^2\big] = \sum_{s=1}^{k}\Big(h_s\mathbb{E}\big[\|P_U F(Y_s)P_V\|^2\big]\sum_{t=s}^{k}\mu^{t-s}\Big)$$

$$\geq \sum_{s=1}^{k} h_s \mathbb{E}\big[\|P_U F(Y_s)P_V\|^2\big]\,,$$

we can conclude with (5.30) that

$$\sum_{t=1}^{\infty} h_t \mathbb{E}[\|P_U F(Y_t)P_V\|^2] \leq \sum_{t=1}^{\infty}\sum_{s=1}^{t}\mu^{t-s}h_s\mathbb{E}\big[\|P_U F(Y_s)P_V\|^2\big] < \infty\,. \tag{5.31}$$

Now, using (5.25), we write

$$\|Y_{t+1}-Y_t\| \leq \lambda h_t \|P_U f(Y_t) P_V\| + (1-\lambda+\lambda\mu)\|X_t\|$$

$$\leq \lambda h_t \|P_U f(Y_t)P_V\| + (1-\lambda+\lambda\mu)\sum_{s=1}^{t}\mu^{t-s}h_s\|P_U f(Y_s)P_V\|\,.$$

From this inequality as well as the definitions of λ and μ it follows that

$$\mathbb{E}\big[\|Y_{t+1}-Y_t\|\big] \leq \frac{2B}{1-\mu}\Big(\frac{h_t}{2} + \frac{(1-\mu)}{2}\sum_{s=1}^{t}\mu^{t-s}h_s\Big) =: \frac{2B}{1-\mu}\frac{h_t+\tilde{h}_t}{2} \tag{5.32}$$

for $\tilde{h}_t = (1-\mu)\sum_{s=1}^{t}\mu^{t-s}h_s$. To bring these values together, consider now a new set of sequences

$$a_t \coloneqq h_t,$$

$$\tilde{a}_t \coloneqq \frac{h_t+\tilde{h}_t}{2},$$

$$b_t \coloneqq \mathbb{E}\big[\|P_U F(Y_t)P_V\|\big]\,.$$

Using the Stolz theorem, we can compute the limit

$$\lim_{t\to\infty}\frac{h_t}{\tilde{h}_t} = \frac{1}{1-\mu}\lim_{t\to\infty}\frac{h_t/\mu^t}{h_t/\mu^t + h_{t-1}/\mu^{t-1} + \cdots + h_1/\mu}$$
$$= \frac{1}{1-\mu}\lim_{t\to\infty}\frac{h_{t+1}/\mu^{t+1} - h_t/\mu^t}{h_{t+1}/\mu^{t+1}} = \frac{1}{1-\mu}\lim_{t\to\infty}\Big(1-\mu\frac{h_t}{h_{t+1}}\Big) = 1\,,$$

from which one gets

$$\lim_{t\to\infty}\frac{a_t}{\tilde{a}_t} = 1\,.$$

Also, by using Jensen's inequality and (5.31), we can see

$$\sum_{t=1}^{\infty} a_t b_t^2 = \sum_{t=1}^{\infty} h_t\big(\mathbb{E}\big[\|P_U F(Y_t)P_V\|\big]\big)^2 \leq \sum_{t=1}^{\infty} h_t \mathbb{E}\big[\|P_U F(Y_t)P_V\|^2\big] < \infty\,.$$

Lastly, using the inverse triangle inequality and (5.32), we obtain

$$|b_{t+1} - b_t| = |\mathbb{E}\big[\|P_U F(Y_{t+1})P_V\|\big] - \mathbb{E}\big[\|P_U F(Y_t)P_V\|\big]|$$
$$\leq \mathbb{E}\big[\|P_U(F(Y_{t+1}) - F(Y_t))P_V\|\big] \leq L\mathbb{E}\big[\|Y_{t+1} - Y_t\|\big]$$
$$\leq \frac{2LB}{1-\mu}\frac{h_t + \tilde{h}_t}{2} = \frac{2LB}{1-\mu}\tilde{a}_t\,.$$

Lemma 5.11 is thus applicable and yields

□

$$\lim_{t\to\infty}\mathbb{E}\big[\|P_U F(Y_t)P_V\|\big] = \lim_{t\to\infty} b_t = 0\,.$$

This result shows that, under Assumption 5.5.1, the SUM Algorithm 3.1 yields a sequence of iterates that comes arbitrarily close to satisfying the necessary condition for optimality (3.10) in expectation. If, like in the previous section, Assumption 5.5.2 is also imposed, then this exact statement also holds for the stronger optimality condition (3.9).

Notice that the result in Theorem 5.15 even improves upon the previous one from Corollary 5.10. By setting $\lambda = \mu = 0$ in (5.24), we obtain the stochastic gradient descent algorithm, so (5.28) is valid even for the SGD Algorithm 3.1 under Assumption 5.5.1. Although (5.28) does not induce a result for squared norms like (5.21), it shows so-called last-iterate convergence as opposed to the convergence of a subsequence or in average. Furthermore, since the equation (5.31) is identical to the statement of Theorem 5.9, the exact result of Corollary 5.10 is effectively obtained at that moment in the above proof.

This finding indicates that the SUM Algorithm 3.1 is also a valid optimization algorithm for training individual layers of DNNs. The fact that it relies entirely on Assumption 5.5, however, means that it cannot be used at the beginning of training. In practice, one should use an algorithm such as the Rank-Adaptive SGD Algorithm 3.1 until the ranks of the parameter matrices become constant and then switch to the SUM Algorithm 3.1. The fact that momentum methods perform better when training DNNs in practice [3] indicates that this might be a better approach than only using the SGD Algorithm 3.1.

6. Conclusion and Further Research. This work aimed to investigate the properties of DLRA (more precisely, of the Rank-Adaptive BUG integrator [7]) as an optimization algorithm for machine learning, an approach proposed in [21]. This involved using stochastic gradients instead of deterministic ones and common machine learning methods as solvers of the differential equations that constitute the integrator. We showed that these modifications not only had little impact on the method's ability to perform low-rank approximation but also yielded algorithms that exhibit convergence on the task of training individual layers of deep neural networks.

Future research on this topic could tackle some of the weaker points of this work. For example, we are always fixing all layers of the network apart from the one that we are currently training. It might be possible to frame our approach as a coordinate descent method and obtain convergence results for the whole network instead of just one layer. Also, it appears pertinent to look for a deeper understanding of Assumption 5.5 and why or why not it is a sensible one, since all of our convergence results rely heavily on it. Since it is essentially an assumption about the S-step of the algorithm, understanding it better could offer insight into the weak optimality condition (3.10) in the context of the manifold $\mathcal{M}_q$, since it derives its structure also from the S-step. In general, the need for the condition (3.10) and Assumption 5.5 stems from an over-reliance on the S-step in our proofs.

Our findings can be interpreted as a theoretical validation of the advantageous convergence behavior reported in [21]. They show that DLRA is generally capable of finding optimal low-rank layers of neural networks during training. We can safely say that this makes it a very promising network pruning technique that should be studied further and will save considerable computational resources for practitioners.

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Gradient convergence in gradient methods with errors*, SIAM journal on optimization, 10 (2000), pp. 627–642.

[2] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, *What is the state of neural network pruning?*, Proceedings of machine learning and systems, 2 (2020), pp. 129–146.

[3] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM review, 60 (2018), pp. 223–311.

[4] G. Ceruti, L. Einkemmer, J. Kusch, and C. Lubich, *A robust second-order low-rank BUG integrator based on the midpoint rule*, arXiv preprint arXiv:2402.08607, (2024).

[5] G. Ceruti, J. Kusch, and C. Lubich, *A rank-adaptive robust integrator for dynamical low-rank approximation*, BIT, 62 (2022), pp. 1149–1174.

[6] G. Ceruti, J. Kusch, and C. Lubich, *A parallel rank-adaptive integrator for dynamical low-rank approximation*, (2023), https://arxiv.org/abs/2304.05660.

[7] G. Ceruti and C. Lubich, *An unconventional robust integrator for dynamical low-rank approximation*, BIT, 62 (2022), pp. 23–44.

[8] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, *An exploration of parameter redundancy in deep networks with circulant projections*, in 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, 2015, pp. 2857–2865.

[9] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, *An exploration of parameter redundancy in deep networks with circulant projections*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 2857–2865.

[10] A. Défossez, L. Bottou, F. Bach, and N. Usunier, *A simple convergence proof of Adam and Adagrad*, arXiv.org, (2022).

[11] J. Frankle and M. Carbin, *The lottery ticket hypothesis: Finding sparse, trainable neural networks*, arXiv preprint arXiv:1803.03635, (2018).

[12] M. Khodak, N. Tenenholtz, L. Mackey, and N. Fusi, *Initialization and regularization of factorized neural layers*, arXiv preprint arXiv:2105.01029, (2021).

[13] E. Kieri, C. Lubich, and H. Walach, *Discretized dynamical low-rank approximation in the presence of small singular values*, SIAM journal on numerical analysis, 54 (2016), pp. 1020–

1038.
[14] E. Kieri and B. Vandereycken, *Projection methods for dynamical low-rank approximation of high-dimensional problems*, Computational Methods in Applied Mathematics, 19 (2019), pp. 73–92.
[15] O. Koch and C. Lubich, *Dynamical low-rank approximation*, SIAM Journal on Matrix Analysis and Applications, 29 (2007), pp. 434–454, https://doi.org/10.1137/050639703.
[16] J. Kusch, *Second-order robust parallel integrators for dynamical low-rank approximation*, arXiv preprint arXiv:2403.02834, (2024).
[17] J. M. Lee, *Introduction to Smooth Manifolds by John M. Lee*, Graduate Texts in Mathematics, 218, New York, NY, 2003.
[18] J. Liu, D. Xu, Y. Lu, J. Kong, and D. P. Mandic, *Last-iterate convergence analysis of stochastic momentum methods for neural networks*, Neurocomputing (Amsterdam), 527 (2023), pp. 27–35.
[19] C. Lubich and I. V. Oseledets, *A projector-splitting integrator for dynamical low-rank approximation*, BIT Numerical Mathematics, 54 (2014), pp. 171–188.
[20] H. Sato, *Riemannian Optimization and Its Applications by Hiroyuki Sato*, SpringerBriefs in Control, Automation and Robotics, Cham, 1st ed. 2021. ed., 2021.
[21] S. Schotthöfer, E. Zangrando, J. Kusch, G. Ceruti, and F. Tudisco, *Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations*, in Advances in Neural Information Processing Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., vol. 35, Curran Associates, Inc., 2022, pp. 20051–20063, https://proceedings.neurips.cc/paper_files/paper/2022/file/7e98b00eeafcdaeb0c5661fb9355be3a-Paper-Conference.pdf.
[22] S. Schotthöfer, E. Zangrando, J. Kusch, G. Ceruti, and F. Tudisco, *Dlrt-net*. https://github.com/COMPiLELab/DLRT-Net, 2023.
[23] P. Villalobos, J. Sevilla, T. Besiroglu, L. Heim, A. Ho, and M. Hobbhahn, *Machine learning model sizes and the parameter gap*, arXiv.org, (2022).
[24] H. Wang, S. Agarwal, and D. Papailiopoulos, *Pufferfish: Communication-efficient models at no extra cost*, Proceedings of Machine Learning and Systems, 3 (2021), pp. 365–386.
[25] S. J. Wright and B. Recht, *Optimization for Data Analysis*, Cambridge University Press, 2022.
[26] E. Zangrando, S. Schotthöfer, G. Ceruti, J. Kusch, and F. Tudisco, *Rank-adaptive spectral pruning of convolutional layers during training*, arXiv preprint arXiv:2305.19059, (2023).