A Modular Analysis of Provable Acceleration via Polyak's Momentum: Training a Wide ReLU Network and a Deep Linear Network

Jun-Kun Wang [1] **Chi-Heng Lin** [2] **Jacob Abernethy** [1]

Abstract

Incorporating a so-called "momentum" dynamic in gradient descent methods is widely used in neural net training as it has been broadly observed that, at least empirically, it often leads to significantly faster convergence. At the same time, there are very few theoretical guarantees in the literature to explain this apparent acceleration effect. Even for the classical strongly convex quadratic problems, several existing results only show Polyak's momentum has an accelerated linear rate asymptotically. In this paper, we first revisit the quadratic problems and show a non-asymptotic accelerated linear rate of Polyak's momentum. Then, we provably show that Polyak's momentum achieves acceleration for training a one-layer wide ReLU network and a deep linear network, which are perhaps the two most popular canonical models for studying optimization and deep learning in the literature. Prior work (Du et al., 2019b; Wu et al., 2019c) showed that using vanilla gradient descent, and with an use of over-parameterization, the error decays as $(1-\Theta(\frac{1}{\kappa'}))^t$ after t iterations, where κ' is the condition number of a Gram Matrix. Our result shows that with the appropriate choice of parameters Polyak's momentum has a rate of $(1-\Theta(\frac{1}{\sqrt{\kappa'}}))^t$. For the deep linear network, prior work (Hu et al., 2020b) showed that vanilla gradient descent has a rate of $(1-\Theta(\frac{1}{\kappa}))^t$, where κ is the condition number of a data matrix. Our result shows an acceleration rate $(1-\Theta(\frac{1}{\sqrt{\kappa}}))^t$ is achievable by Polyak's momentum. This work establishes that momentum does indeed speed up neural net training.

[1]School of Computer Science, Georgia Institute of Technology [2]School of Electrical and Computer Engineering, Georgia Institute of Technology. Correspondence to: Jun-Kun Wang <jimwang@gatech.edu>, Chi-Heng Lin <cl3385@gatech.edu>, Jacob Abernethy <prof@gatech.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Momentum methods are very popular for training neural networks in various applications (e.g. He et al. (2016); Vaswani et al. (2017); Krizhevsky et al. (2012)). It has been widely observed that the use of momentum helps faster training in deep learning (e.g. Loshchilov & Hutter (2019); Wilson et al. (2017); Cutkosky & Orabona (2019)). Among all the momentum methods, the most popular one seems to be Polyak's momentum (a.k.a. Heavy Ball momentum) (Polyak, 1964), which is the default choice of momentum in PyTorch and Tensorflow. The success of Polyak's momentum in deep learning is widely appreciated and almost all of the recently developed adaptive gradient methods like Adam (Kingma & Ba, 2015), AMSGrad (Reddi et al., 2018), and AdaBound (Luo et al., 2019) adopt the use of Polyak's momentum, instead of Nesterov's momentum.

However, despite its popularity, little is known in theory about why Polyak's momentum helps to accelerate training neural networks. Even for convex optimization, problems like strongly convex quadratic problems seem to be one of the few cases that discrete-time Polyak's momentum method provably achieves faster convergence than standard gradient descent (e.g. Lessard et al. (2016); Goh (2017); Ghadimi et al. (2015); Gitman et al. (2019); Loizou & Richtárik (2017; 2018); Can et al. (2019); Scieur & Pedregosa (2020); Flammarion & Bach (2015); Wilson et al. (2021); Franca et al. (2020); Diakonikolas & Jordan (2019); Shi et al. (2018); Hu (2020)). On the other hand, the theoretical guarantees of Adam, AMSGrad , or AdaBound are only worse if the momentum parameter β is non-zero and the guarantees deteriorate as the momentum parameter increases, which do not show any advantage of the use of momentum (Alacaoglu et al., 2020). Moreover, the convergence rates that have been established for Polyak's momentum in several related works (Gadat et al., 2016; Sun et al., 2019; Yang et al., 2018; Liu et al., 2020c; Mai & Johansson, 2020) do not improve upon those for vanilla gradient descent or vanilla SGD in the worst case. Lessard et al. (2016); Ghadimi et al. (2015) even show negative cases in *convex* optimization that the use of Polyak's momentum results in divergence. Furthermore, Kidambi et al. (2018) construct a problem instance for which the momentum

Algorithm 1 Gradient descent with Polyak's momentum (Polyak, 1964) (Equivalent Version 1)

Required: Step size η and momentum parameter β.
Init: $w_0 \in \mathbb{R}^d$ and $M_{-1} = 0 \in \mathbb{R}^d$.
for $t = 0$ to T **do**
  Given current iterate w_t, obtain gradient $\nabla \ell(w_t)$.
  Update momentum $M_t = \beta M_{t-1} + \nabla \ell(w_t)$.
  Update iterate $w_{t+1} = w_t - \eta M_t$.
end for

Algorithm 2 Gradient descent with Polyak's momentum (Polyak, 1964) (Equivalent Version 2)

Required: step size η and momentum parameter β.
Init: $w_0 = w_{-1} \in \mathbb{R}^d$
for $t = 0$ to T **do**
  Given current iterate w_t, obtain gradient $\nabla \ell(w_t)$.
  Update iterate $w_{t+1} = w_t - \eta \nabla \ell(w_t) + \beta(w_t - w_{t-1})$.
end for

method under its optimal tuning is outperformed by other algorithms. Wang et al. (2020) show that Polyak's momentum helps escape saddle points faster compared with the case without momentum, which is the only provable advantage of Polyak's momentum in non-convex optimization that we are aware of. A solid understanding of the empirical success of Polyak's momentum in deep learning has eluded researchers for some time.

We begin this paper by first revisiting the use of Polyak's momentum for the class of strongly convex quadratic problems,

$$\min_{w \in \mathbb{R}^d} \tfrac{1}{2} w^\top \Gamma w + b^\top w, \tag{1}$$

where $\Gamma \in \mathbb{R}^{d \times d}$ is a PSD matrix such that $\lambda_{\max}(\Gamma) = \alpha$, $\lambda_{\min}(\Gamma) = \mu > 0$. This is one of the few[1] known examples that Polyak's momentum has a provable *globally accelerated* linear rate in the *discrete-time* setting. Yet even for this class of problems existing results only establish an accelerated linear rate in an asymptotic sense and several of them do not have an explicit rate in the non-asymptotic regime (e.g. Polyak (1964); Lessard et al. (2016); Mitliagkas (2019); Recht (2018)). Is it possible to prove a non-asymptotic accelerated linear rate in this case? We will return to this question soon.

For general μ-strongly convex, α-smooth, and twice differentiable functions (not necessarily quadratic), denoted as $F^2_{\mu,\alpha}$, Theorem 9 in Polyak (1964) shows an asymptotic accelerated linear rate when the iterate is *sufficiently* close to the minimizer so that the landscape can be well approximated by that of a quadratic function. However, the definition of the neighborhood was not very precise in the paper. In this work, we show a locally accelerated linear rate under a quantifiable definition of the neighborhood.

Furthermore, we provably show that Polyak's momentum helps to achieve a faster convergence for training two neural networks, compared to vanilla GD. The first is training a one-layer ReLU network. Over the past few years there have appeared an enormous number of works considering training a one-layer ReLU network, provably showing convergence results for vanilla (stochastic) gradient descent (e.g. Li & Liang (2018); Ji & Telgarsky (2020); Li & Yuan (2017); Du et al. (2019b;a); Allen-Zhu et al. (2019); Song & Yang (2019); Zou et al. (2019); Arora et al. (2019c); Jacot et al. (2018); Lee et al. (2019); Chizat et al. (2019); Oymak & Soltanolkotabi (2019); Brutzkus & Globerson (2017); Chen et al. (2020a); Tian (2017); Soltanolkotabi (2017); Bai & Lee (2020); Ghorbani et al. (2019); Li et al. (2020); Hanin & Nica (2020); Daniely (2017); Zou & Gu (2019); Dukler et al. (2020); Daniely (2020); Wei et al. (2019); Yehudai & Shamir (2020); Fang et al. (2019); Su & Yang (2019); Chen et al. (2020b)), as well as for other algorithms (e.g. Zhang et al. (2019); Wu et al. (2019b); Cai et al. (2019); Zhong et al. (2017); Ge et al. (2019); van den Brand et al. (2020); Lee et al. (2020); Pilanci & Ergen (2020)). However, we are not aware of any theoretical works that study the momentum method in neural net training except the work Krichene et al. (2020). These authors show that SGD with Polyak's momentum (a.k.a. stochastic Heavy Ball) with infinitesimal step size, i.e. $\eta \to 0$, for training a one-hidden-layer network with an infinite number of neurons, i.e. $m \to \infty$, converges to a stationary solution. However, the theoretical result does not show a faster convergence by momentum. In this paper we consider the discrete-time setting and nets with finitely many neurons. We provide a non-asymptotic convergence rate of Polyak's momentum, establishing a concrete improvement relative to the best-known rates for vanilla gradient descent.

Our setting of training a ReLU network follows the same framework as previous results, including Du et al. (2019b); Arora et al. (2019c); Song & Yang (2019). Specifically, we study training a one-hidden-layer ReLU neural net of the form,

$$\mathcal{N}_W^{\text{ReLU}}(x) := \tfrac{1}{\sqrt{m}} \textstyle\sum_{r=1}^m a_r \sigma(\langle w^{(r)}, x \rangle), \tag{2}$$

where $\sigma(z) := z \cdot \mathbf{1}\{z \geq 0\}$ is the ReLU activation, $w^{(1)}, \ldots, w^{(m)} \in \mathbb{R}^d$ are the weights of m neurons on the first layer, $a_1, \ldots, a_m \in \mathbb{R}$ are weights on the second layer, and $\mathcal{N}_W^{\text{ReLU}}(x) \in \mathbb{R}$ is the output predicted on input x. Assume n number of samples $\{x_i \in \mathbb{R}^d\}_{i=1}^n$ is given. Following Du et al. (2019b); Arora et al. (2019c); Song & Yang (2019), we define a Gram matrix $H \in \mathbb{R}^{n \times n}$ for the weights W and its expectation $\bar{H} \in \mathbb{R}^{n \times n}$ over the random draws of $w^{(r)} \sim N(0, I_d) \in \mathbb{R}^d$ whose (i, j) entries are

[1] In Section 2 and Appendix A, we will provide more discussions about this point.

defined as follows,

$$H(W)_{i,j} = \sum_{r=1}^{m} \frac{x_i^\top x_j}{m} \mathbf{1}\{\langle w^{(r)}, x_i\rangle \geq 0 \;\&\; \langle w^{(r)}, x_j\rangle \geq 0\}$$
$$\bar{H}_{i,j} := \mathop{\mathsf{E}}_{w^{(r)}}[x_i^\top x_j \mathbf{1}\{\langle w^{(r)}, x_i\rangle \geq 0 \;\&\; \langle w^{(r)}, x_j\rangle \geq 0\}]. \tag{3}$$

The matrix $\bar{H}$ is also called a neural tangent kernel (NTK) matrix in the literature (e.g. Jacot et al. (2018); Yang (2019); Bietti & Mairal (2019)). Assume that the smallest eigenvalue $\lambda_{\min}(\bar{H})$ is strictly positive and certain conditions about the step size and the number of neurons are satisfied. Previous works (Du et al., 2019b; Song & Yang, 2019) show a linear rate of vanilla gradient descent, while we show an accelerated linear rate [2] of gradient descent with Polyak's momentum. As far as we are aware, our result is the first acceleration result of training an over-parametrized ReLU network.

The second result is training a deep linear network. The deep linear network is a canonical model for studying optimization and deep learning, and in particular for understanding gradient descent (e.g. Shamir (2019); Saxe et al. (2014); Hu et al. (2020b)), studying the optimization landscape (e.g. Kawaguchi (2016); Laurent & von Brecht (2018)), and establishing the effect of implicit regularization (e.g. Moroshko et al. (2020); Ji & Telgarsky (2019); Li et al. (2018); Razin & Cohen (2020); Arora et al. (2019b); Gidel et al. (2019); Gunasekar et al. (2017); Lyu & Li (2020)). In this paper, following (Du & Hu, 2019; Hu et al., 2020b), we study training a L-layer linear network of the form,

$$\mathcal{N}_W^{L\text{-linear}}(x) := \frac{1}{\sqrt{m^{L-1}d_y}} W^{(L)} W^{(L-1)} \cdots W^{(1)} x, \tag{4}$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix of the layer $l \in [L]$, and $d_0 = d$, $d_L = d_y$ and $d_l = m$ for $l \neq 1, L$. Therefore, except the first layer $W^{(1)} \in \mathbb{R}^{m \times d}$ and the last layer $W^{(L)} \in \mathbb{R}^{d_y \times m}$, all the intermediate layers are $m \times m$ square matrices. The scaling $\frac{1}{\sqrt{m^{L-1}d_y}}$ is necessary to ensure that the network's output at the initialization $\mathcal{N}_{W_0}^{L\text{-linear}}(x)$ has the same size as that of the input x, in the sense that $\mathbb{E}[\|\mathcal{N}_{W_0}^{L\text{-linear}}(x)\|^2] = \|x\|^2$, where the expectation is taken over some appropriate random initialization of the network (see e.g. Du & Hu (2019); Hu et al. (2020b)). Hu et al. (2020b) show vanilla gradient descent with orthogonal initialization converges linearly and the required width of the network m is independent of the depth L, while we show an accelerated linear rate of Polyak's momentum and the width m is also independent of L. To our knowledge, this is the first acceleration result of training a deep linear network.

A careful reader may be tempted by the following line of reasoning: a deep linear network (without activation) is effectively a simple linear model, and we already know that a linear model with the squared loss gives a quadratic objective for which Polyak's momentum exhibits an accelerated convergence rate. But this intuition, while natural, is not quite right: it is indeed nontrivial even to show that vanilla gradient descent provides a linear rate on deep linear networks (Hu et al., 2020b; Du & Hu, 2019; Shamir, 2019; Arora et al., 2019a; Hardt & Ma, 2016; Wu et al., 2019a; Zou et al., 2020), as the optimization landscape is non-convex. Existing works show that under certain assumptions, all the local minimum are global (Kawaguchi, 2016; Laurent & von Brecht, 2018; Yun et al., 2018; Lu & Kawaguchi, 2017; Zhou & Liang, 2018; Hardt & Ma, 2016). These results are not sufficient to explain the linear convergence of momentum, let alone the acceleration; see Section H in the appendix for an empirical result.

Similarly, it is known that under the NTK regime the output of the ReLU network trained by gradient descent can be approximated by a linear model (e.g. Hu et al. (2020a)). However, this result alone neither implies a global convergence of any algorithm nor characterizes the optimization landscape. While (Liu et al., 2020a) attempt to derive an algorithm-independent equivalence of a class of linear models and a family of wide networks, their result requires the activation function to be differentiable which does not hold for the most prevalent networks like ReLU. Also, their work heavily depends on the regularity of Hessian, making it hard to generalize beyond differentiable networks. Hence, while there has been some progress understanding training of wide networks through linear models, there remains a significant gap in applying this to the momentum dynamics of a non-differentiable networks. Liu et al. (2020b) establish an interesting connection between solving an over-parametrized non-linear system of equations and solving the classical linear system. They show that for smooth and twice differentiable activation, the optimization landscape of an over-parametrized network satisfies a (non-convex) notion called the Polyak-Lokasiewicz (PL) condition (Polyak, 1963), i.e. $\frac{1}{2}\|\nabla \ell(w)\|^2 \geq \mu\,(\ell(w) - \ell(w_*))$, where w_* is a global minimizer and $\mu > 0$. It is not clear whether their result can be extended to ReLU activation, however, and the existing result of Danilova et al. (2018) for the discrete-time Polyak's momentum under the PL condition does not give an accelerated rate nor is it better than that of vanilla GD. Aujol et al. (2020) show a *variant* of Polyak's momentum method having an accelerated rate in a *continuous-time* limit for a problem that satisfies PL and has a unique global minimizer. It is unclear if their result

[2] We borrow the term "accelerated linear rate" from the convex optimization literature (Nesterov, 2013), because the result here has a resemblance to those results in convex optimization, even though the neural network training is a non-convex problem.

is applicable to our problem. Therefore, showing the advantage of training the ReLU network and the deep linear network by using existing results of Polyak's momentum can be difficult.

To summarize, our contributions in the present work include

- In convex optimization, we show an accelerated linear rate in the non-asymptotic sense for solving the class of the strongly convex quadratic problems via Polyak's momentum (Theorem 7). We also provide an analysis of the accelerated local convergence for the class of functions in $F^2_{\mu,\alpha}$ (Theorem 8). We establish a technical result (Theorem 5) that helps to obtain these non-asymptotic rates.
- In non-convex optimization, we show accelerated linear rates of the discrete-time Polyak's momentum for training an over-parametrized ReLU network and a deep linear network (Theorems 9 and 10).

Furthermore, we will develop a modular analysis to show all the results in this paper. We identify conditions and propose a meta theorem of acceleration when the momentum method exhibits a certain dynamic, which can be of independent interest. We show that when applying Polyak's momentum for these problems, the induced dynamics exhibit a form where we can directly apply our meta theorem.

2. Preliminaries

Throughout this paper, $\|\cdot\|_F$ represents the Frobenius norm and $\|\cdot\|_2$ represents the spectral norm of a matrix, while $\|\cdot\|$ represents l_2 norm of a vector. We also denote $\otimes$ the Kronecker product, $\sigma_{\max}(\cdot) = \|\cdot\|_2$ and $\sigma_{\min}(\cdot)$ the largest and the smallest singular value of a matrix respectively.

For the case of training neural networks, we will consider minimizing the squared loss

$$\ell(W) := \tfrac{1}{2}\sum_{i=1}^n \left(y_i - \mathcal{N}_W(x_i)\right)^2, \tag{5}$$

where $x_i \in \mathbb{R}^d$ is the feature vector, $y_i \in \mathbb{R}^{d_y}$ is the label of sample i, and there are n number of samples. For training the ReLU network, we have $\mathcal{N}_W(\cdot) := \mathcal{N}_W^{\text{ReLU}}(\cdot)$, $d_y = 1$, and $W := \{w^{(r)}\}_{r=1}^m$, while for the deep linear network, we have $\mathcal{N}_W(\cdot) := \mathcal{N}_W^{L\text{-linear}}(\cdot)$, and W represents the set of all the weight matrices, i.e. $W := \{W^{(l)}\}_{l=1}^L$. The notation A^k represents the k_{th} matrix power of A.

2.1. Prior result of Polyak's momentum

Algorithm 1 and Algorithm 2 show two equivalent presentations of gradient descent with Polyak's momentum. Given the same initialization, one can show that Algorithm 1 and Algorithm 2 generate exactly the same iterates during optimization.

Let us briefly describe a prior acceleration result of Polyak's momentum. The recursive dynamics of Poylak's momentum for solving the strongly convex quadratic problems (1) can be written as

$$\begin{bmatrix} w_{t+1} - w_* \\ w_t - w_* \end{bmatrix} = \underbrace{\begin{bmatrix} I_d - \eta\Gamma + \beta I_d & -\beta I_d \\ I_d & 0_d \end{bmatrix}}_{:=A} \cdot \begin{bmatrix} w_t - w_* \\ w_{t-1} - w_* \end{bmatrix}, \tag{6}$$

where w_* is the unique minimizer. By a recursive expansion, one can get

$$\| \begin{bmatrix} w_t - w_* \\ w_{t-1} - w_* \end{bmatrix} \| \le \|A^t\|_2 \| \begin{bmatrix} w_0 - w_* \\ w_{-1} - w_* \end{bmatrix} \|. \tag{7}$$

Hence, it suffices to control the spectral norm of the matrix power $\|A^t\|_2$ for obtaining a convergence rate. In the literature, this is achieved by using Gelfand's formula.

Theorem 1. *(Gelfand (1941); see also Foucart (2018)) (Gelfand's formula) Let A be a $d \times d$ matrix. Define the spectral radius $\rho(A) := \max_{i\in[d]} |\lambda_i(A)|$, where $\lambda_i(\cdot)$ is the i_{th} eigenvalue. Then, there exists a non-negative sequence $\{\epsilon_t\}$ such that $\|A^t\|_2 = (\rho(A) + \epsilon_t)^t$ and $\lim_{t\to\infty} \epsilon_t = 0$.*

We remark that there is a lack of the convergence rate of ϵ_t in Gelfand's formula in general.

Denote $\kappa := \alpha/\mu$ the condition number. One can control the spectral radius $\rho(A)$ as $\rho(A) \le 1 - \frac{2}{\sqrt{\kappa}+1}$ by choosing η and β appropriately, which leads to the following result.

Theorem 2. *(Polyak (1964); see also Lessard et al. (2016); Recht (2018); Mitliagkas (2019)) Gradient descent with Polyak's momentum with the step size $\eta = \frac{4}{(\sqrt{\mu}+\sqrt{\alpha})^2}$ and the momentum parameter $\beta = \left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^2$ has*

$$\| \begin{bmatrix} w_{t+1} - w_* \\ w_t - w_* \end{bmatrix} \| \le \left(1 - \frac{2}{\sqrt{\kappa}+1} + \epsilon_t\right)^{t+1} \| \begin{bmatrix} w_0 - w_* \\ w_{-1} - w_* \end{bmatrix} \|,$$

where ϵ_t is a non-negative sequence that goes to zero.

That is, when $t \to \infty$, Polyak's momentum has the $(1 - \frac{2}{\sqrt{\kappa}+1})$ rate, which has a better dependency on the condition number κ than the $1 - \Theta(\frac{1}{\kappa})$ rate of vanilla gradient descent. A concern is that the bound is not quantifiable for a finite t. On the other hand, we are aware of a different analysis that leverages Chebyshev polynomials instead of Gelfand's formula (e.g. Liu & Belkin (2018)), which manages to obtain a $t(1 - \Theta(\frac{1}{\sqrt{\kappa}}))^t$ convergence rate. So the accelerated linear rate is still obtained in an asymptotic sense. Theorem 9 in Can et al. (2019) shows a rate

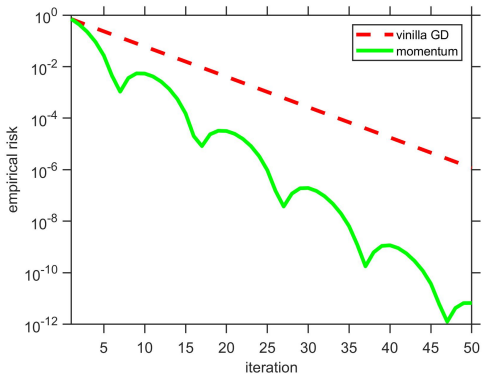

Figure 1. Empirical risk $\ell(W_t)$ vs. iteration t. Polyak's momentum accelerates the optimization process of training an over-parametrized one-layer ReLU network. Experimental details are available in Appendix H.

$\max\{\bar{C}_1, t\bar{C}_2\}(1-\Theta(\frac{1}{\sqrt{\kappa}})^t)$ for some constants $\bar{C}_1$ and $\bar{C}_2$ under the same choice of the momentum parameter and the step size as Theorem 2. However, for a large t, the dominant term could be $t(1-\Theta(\frac{1}{\sqrt{\kappa}})^t)$. In this paper, we aim at obtaining a bound that (I) holds for a wide range of values of the parameters, (II) has a dependency on the squared root of the condition number $\sqrt{\kappa}$, (III) is quantifiable in each iteration and is better than the rate $t(1-\Theta(\frac{1}{\sqrt{\kappa}}))^t$.

2.2. (One-layer ReLU network) Settings and Assumptions

The ReLU activation is not differentiable at zero. So for solving (5), we will replace the notion of gradient in Algorithm 1 and 2 with subgradient $\frac{\partial \ell(W_t)}{\partial w_t^{(r)}} := \frac{1}{\sqrt{m}}\sum_{i=1}^n \big(\mathcal{N}_{W_t}(x_i)-y_i\big)a_r \cdot \mathbf{1}[\langle w_t^{(r)}, x_i\rangle \geq 0]x_i$ and update the neuron r as $w_{t+1}^{(r)} = w_t^{(r)} - \eta\frac{\partial \ell(W_t)}{\partial w_t^{(r)}} + \beta\big(w_t^{(r)} - w_{t-1}^{(r)}\big)$. As described in the introduction, we assume that the smallest eigenvalue of the Gram matrix $\bar{H} \in \mathbb{R}^{n\times n}$ is strictly positive, i.e. $\lambda_{\min}(\bar{H}) > 0$. We will also denote the largest eigenvalue of the Gram matrix $\bar{H}$ as $\lambda_{\max}(\bar{H})$ and denote the condition number of the Gram matrix as $\kappa := \frac{\lambda_{\max}(\bar{H})}{\lambda_{\min}(\bar{H})}$. Du et al. (2019b) show that the strict positiveness assumption is indeed mild. Specifically, they show that if no two inputs are parallel, then the least eigenvalue is strictly positive. Panigrahi et al. (2020) were able to provide a quantitative lower bound under certain conditions. Following the same framework of Du et al. (2019b), we consider that each weight vector $w^{(r)} \in \mathbb{R}^d$ is initialized according to the normal distribution, i.e. $w^{(r)} \sim N(0, I_d)$, and each $a_r \in R$ is sampled from the Rademacher distribution, i.e. $a_r = 1$ with probability 0.5; and $a_r = -1$ with probability 0.5. We also assume $\|x_i\| \leq 1$ for all samples i. As the previous works (e.g. Li & Liang (2018); Ji & Telgarsky (2020); Du et al. (2019b)), we consider only training the first layer $\{w^{(r)}\}$ and the second layer $\{a_r\}$ is fixed throughout the iterations. We will denote $u_t \in \mathbb{R}^n$ whose i_{th} entry is the network's prediction for sample i, i.e. $u_t[i] := \mathcal{N}_{W_t}^{\text{ReLU}}(x_i)$ in iteration t and denote $y \in \mathbb{R}^n$ the vector whose i_{th} element is the label of sample i. The following theorem is a prior result due to Du et al. (2019b).

Theorem 3. *(Theorem 4.1 in Du et al. (2019b)) Assume that $\lambda := \lambda_{\min}(\bar{H})/2 > 0$ and that $w_0^{(r)} \sim N(0, I_d)$ and a_r uniformly sampled from $\{-1, 1\}$. Set the number of nodes $m = \Omega(\lambda^{-4}n^6\delta^{-3})$ and the constant step size $\eta = O(\frac{\lambda}{n^2})$. Then, with probability at least $1-\delta$ over the random initialization, vanilla gradient descent, i.e. Algorithm 1& 2 with $\beta = 0$, has $\|u_t - y\|^2 \leq (1-\eta\lambda)^t \cdot \|u_0 - y\|^2$.*

Later Song & Yang (2019) improve the network size m to $m = \Omega(\lambda^{-4}n^4\log^3(n/\delta))$. Wu et al. (2019c) provide an improved analysis over Du et al. (2019b), which shows that the step size η of vanilla gradient descent can be set as $\eta = \frac{1}{c_1\lambda_{\max}(\bar{H})}$ for some quantity $c_1 > 0$. The result in turn leads to a convergence rate $(1-\frac{1}{c_2\kappa})$ for some quantity $c_2 > 0$. However, the quantities c_1 and c_2 are not universal constants and actually depend on the problem parameters $\lambda_{\min}(\bar{H})$, n, and δ. A question that we will answer in this paper is "*Can Polyak's momentum achieve an accelerated linear rate $\Big(1-\Theta(\frac{1}{\sqrt{\kappa}})\Big)$, where the factor $\Theta(\frac{1}{\sqrt{\kappa}})$ does not depend on any other problem parameter?*".

2.3. (Deep Linear network) Settings and Assumptions

For the case of deep linear networks, we will denote $X := [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$ the data matrix and $Y := [y_1, \ldots, y_n] \in \mathbb{R}^{d_y\times n}$ the corresponding label matrix. We will also denote $\bar{r} := rank(X)$ and the condition number $\kappa := \frac{\lambda_{\max}(X^\top X)}{\lambda_{\bar{r}}(X^\top X)}$. Following Hu et al. (2020b), we will assume that the linear network is initialized by the orthogonal initialization, which is conducted by sampling uniformly from (scaled) orthogonal matrices such that $(W_0^{(1)})^\top W_0^{(1)} = mI_d$, $W_0^{(L)}(W_0^{(L)})^\top = mI_{d_y}$, and $(W_0^{(l)})^\top W_0^{(l)} = W_0^{(l)}(W_0^{(l)})^\top = mI_m$ for layer $2 \leq l \leq L-1$. We will denote $W^{(j:i)} := W_jW_{j-1}\cdots W_i = \Pi_{l=i}^j W_l$, where $1 \leq i \leq j \leq L$ and $W^{(i-1:i)} = I$. We also denote the network's output $U := \frac{1}{\sqrt{m^{L-1}d_y}}W^{(L:1)}X \in \mathbb{R}^{d_y\times n}$.

In our analysis, following Du & Hu (2019); Hu et al. (2020b), we will further assume that (A1) there exists a W^* such that $Y = W^*X$, $X \in \mathbb{R}^{d\times\bar{r}}$, and $\bar{r} = rank(X)$, which is actually without loss of generality (see e.g. the discussion in Appendix B of Du & Hu (2019)).

Theorem 4. *(Theorem 4.1 in Hu et al. (2020b)) As-*

sume (A1) and the use of the orthogonal initialization. Suppose the width of the deep linear network satisfies $m \geq C\frac{\|X\|_F^2}{\sigma_{\max}^2(X)}\kappa^2\left(d_y(1+\|W_*\|_2^2)+\log(\bar{r}/\delta)\right)$ *and* $m \geq \max\{d_x, d_y\}$ *for some* $\delta \in (0,1)$ *and a sufficiently large constant* $C > 0$. *Set the constant step size* $\eta = \frac{d_y}{2L\sigma_{\max}^2(X)}$. *Then, with probability at least* $1-\delta$ *over the random initialization, vanilla gradient descent, i.e. Algorithm 1& 2 with* $\beta = 0$, *has* $\|U_t - Y\|_F^2 \leq \left(1-\Theta(\frac{1}{\kappa})\right)^t \cdot \|U_0 - Y\|_F^2$.

3. Modular Analysis

In this section, we will provide a meta theorem for the following dynamics of the residual vector $\xi_t \in \mathbb{R}^{n_0}$,

$$\begin{bmatrix}\xi_{t+1}\\ \xi_t\end{bmatrix} = \begin{bmatrix}I_{n_0} - \eta H + \beta I_{n_0} & -\beta I_{n_0}\\ I_{n_0} & 0_{n_0}\end{bmatrix}\begin{bmatrix}\xi_t\\ \xi_{t-1}\end{bmatrix} + \begin{bmatrix}\varphi_t\\ 0_{n_0}\end{bmatrix}, \tag{8}$$

where η is the step size, β is the momentum parameter, $H \in \mathbb{R}^{n_0\times n_0}$ is a PSD matrix, $\varphi_t \in \mathbb{R}^{n_0}$ is some vector, and I_{n_0} is the $n_0 \times n_0$-dimensional identity matrix. Note that ξ_t and φ_t depend on the underlying model learned at iteration t, i.e. depend on W_t.

We first show that the residual dynamics of Polyak's momentum for solving all the four problems in this paper are in the form of (8). The proof of the following lemmas (Lemma 2, 3, and 4) are available in Appendix B.

3.1. Realization: Strongly convex quadratic problems

One can easily see that the dynamics of Polyak's momentum (6) for solving the strongly convex quadratic problem (1) is in the form of (8). We thus have the following lemma.

Lemma 1. *Applying Algorithm 1 or Algorithm 2 to solving the class of strongly convex quadratic problems (1) induces a residual dynamics in the form of (8), where* $\xi_t = w_t - w_*$ *(and hence* $n_0 = d$*),* $H = \Gamma$, $\varphi_t = 0_d$.

3.2. Realization: Solving $F_{\mu,\alpha}^2$

A similar result holds for optimizing functions in $F_{\mu,\alpha}^2$.

Lemma 2. *Applying Algorithm 1 or Algorithm 2 to minimizing a function* $f(w) \in F_{\mu,\alpha}^2$ *induces a residual dynamics in the form of (8), where* $\xi_t = w_t - w_*$, $H = \int_0^1 \nabla^2 f\big((1-\tau)w_0 + \tau w_*\big)d\tau$, $\varphi_t = \eta\big(\int_0^1 \nabla^2 f\big((1-\tau)w_0 + \tau w_*\big)d\tau - \int_0^1 \nabla^2 f\big((1-\tau)w_t + \tau w_*\big)d\tau\big)(w_t - w_*)$, *where* $w_* := \arg\min_w f(w)$.

3.3. Realization: One-layer ReLU network

More notations: For the analysis, let us define the event $A_{ir} := \{\exists w \in \mathbb{R}^d : \|w - w_0^{(r)}\| \leq R^{\text{ReLU}}, \mathbf{1}\{x_i^\top w_0^{(r)}\} \neq \mathbf{1}\{x_i^\top w \geq 0\}\}$, where $R^{\text{ReLU}} > 0$ is a number to be determined later. The event A_{ir} means that there exists a $w \in \mathbb{R}^d$ which is within the R^{ReLU}-ball centered at the initial point $w_0^{(r)}$ such that its activation pattern of sample i is different from that of $w_0^{(r)}$. We also denote a random set $S_i := \{r \in [m] : \mathbf{1}\{A_{ir}\} = 0\}$ and its complementary set $S_i^\perp := [m] \setminus S_i$.

Lemma 3 below shows that training the ReLU network $\mathcal{N}_W^{\text{ReLU}}(\cdot)$ via momentum induces the residual dynamics in the form of (8).

Lemma 3. *(Residual dynamics of training the ReLU network* $\mathcal{N}_W^{ReLU}(\cdot)$*) Denote*

$$(H_t)_{i,j} := H(W_t)_{i,j} = \frac{1}{m}\sum_{r=1}^m x_i^\top x_j \times \mathbf{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0 \;\&\; \langle w_t^{(r)}, x_j\rangle \geq 0\}.$$

Applying Algorithm 1 or Algorithm 2 to (5) for training the ReLU network $\mathcal{N}_W^{ReLU}(x)$ *induces a residual dynamics in the form of (8) such that* $\xi_t[i] = \mathcal{N}_{W_t}^{ReLU}(x_i) - y_i$*(and hence* $n_0 = d$*),* $H = H_0$, *and* $\varphi_t = \phi_t + \iota_t$, *where each element* i *of* $\xi_t \in \mathbb{R}^n$ *is the residual error of the sample* i, *and the* i_{th}*-element of* $\phi_t \in \mathbb{R}^n$ *satisfies*

$$|\phi_t[i]| \leq \frac{2\eta\sqrt{n}|S_i^\perp|}{m}\big(\|u_t - y\| + \beta\sum_{s=0}^{t-1}\beta^{t-1-s}\|u_s - y\|\big),$$

and $\iota_t = \eta\,(H_0 - H_t)\,\xi_t \in \mathbb{R}^n$.

3.4. Realization: Deep Linear network

Lemma 4 below shows that the residual dynamics due to Polyak's momentum for training the deep linear network is indeed in the form of (8). In the lemma, "vec" stands for the vectorization of the underlying matrix in column-first order.

Lemma 4. *(Residual dynamics of training* $\mathcal{N}_W^{L\text{-}linear}(\cdot)$*) Denote* $M_{t,l}$ *the momentum term of layer* l *at iteration* t, *which is recursively defined as* $M_{t,l} = \beta M_{t,l-1} + \frac{\partial \ell(W_t^{(L:1)})}{\partial W_t^{(l)}}$. *Denote*

$$H_t := \frac{1}{m^{L-1}d_y}\sum_{l=1}^L [(W_t^{(l-1:1)}X)^\top (W_t^{(l-1:1)}X) \otimes W_t^{(L:l+1)}(W_t^{(L:l+1)})^\top] \in \mathbb{R}^{d_y n\times d_y n}.$$

Applying Algorithm 1 or Algorithm 2 to (5) for training the deep linear network $\mathcal{N}_W^{L\text{-}linear}(x)$ *induces a residual dynamics in the form of (8) such that* $\xi_t = vec(U_t - Y) \in \mathbb{R}^{d_y n}$*(and hence* $n_0 = d_y n$*),* $H = H_0$, *and* $\varphi_t = \phi_t + \psi_t + \iota_t \in \mathbb{R}^{d_y n}$, *where the vector* $\phi_t = \frac{1}{\sqrt{m^{L-1}d_y}} vec\,(\Phi_t X)$ *with*

$$\Phi_t = \Pi_l\left(W_t^{(l)} - \eta M_{t,l}\right) - W_t^{(L:1)} + \eta\sum_{l=1}^L W_t^{(L:l+1)} M_{t,l} W_t^{(l-1:1)},$$

and the vector ψ_t *is*

$$\psi_t = \frac{1}{\sqrt{m^{L-1}d_y}} vec\big((L-1)\beta W_t^{(L:1)}X + \beta W_{t-1}^{(L:1)}X - \beta\sum_{l=1}^L W_t^{(L:l+1)} W_{t-1}^{(l)} W_t^{(l-1:1)} X\big),$$

and $\iota_t = \eta(H_0 - H_t)\xi_t$.

3.5. A key theorem of bounding a matrix-vector product

Our meta theorem of acceleration will be based on Theorem 5 in the following, which upper-bounds the size of the matrix-vector product of a matrix power A^k and a vector v_0. Compared to Gelfand's formula (Theorem 1), Theorem 5 below provides a better control of the size of the matrix-vector product, since it avoids the dependency on the unknown sequence $\{\epsilon_t\}$. The result can be of independent interest and might be useful for analyzing Polyak's momentum for other problems in future research.

Theorem 5. *Let* $A := \begin{bmatrix} (1+\beta)I_n - \eta H & -\beta I_n \\ I_n & 0 \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$. *Suppose that* $H \in \mathbb{R}^{n\times n}$ *is a positive semidefinite matrix. Fix a vector* $v_0 \in \mathbb{R}^n$. *If* β *is chosen to satisfy* $1 \geq \beta > \max\{\left(1-\sqrt{\eta\lambda_{\min}(H)}\right)^2, \left(1-\sqrt{\eta\lambda_{\max}(H)}\right)^2\}$, *then*

$$\|A^k v_0\| \leq \left(\sqrt{\beta}\right)^k C_0 \|v_0\|, \tag{9}$$

where the constant

$$C_0 := \frac{\sqrt{2}(\beta+1)}{\sqrt{\min\{h(\beta,\eta\lambda_{\min}(H)), h(\beta,\eta\lambda_{\max}(H))\}}} \geq 1, \tag{10}$$

and the function $h(\beta,z)$ *is defined as* $h(\beta,z) := -\left(\beta-(1-\sqrt{z})^2\right)\left(\beta-(1+\sqrt{z})^2\right)$.

Note that the constant C_0 in Theorem 5 depends on β and ηH. It should be written as $C_0(\beta,\eta H)$ to be precise. However, for the brevity, we will simply denote it as C_0 when the underlying choice of β and ηH is clear from the context. The proof of Theorem 5 is available in Appendix C. Theorem 5 allows us to derive a concrete upper bound of the residual errors in each iteration of momentum, and consequently allows us to show an accelerated linear rate in the non-asymptotic sense. The favorable property of the bound will also help to analyze Polyak's momentum for training the neural networks. As shown later in this paper, we will need to guarantee the progress of Polyak's momentum in each iteration, which is not possible if we only have a quantifiable bound in the limit. Based on Theorem 5, we have the following corollary. The proof is in Appendix C.1.

Corollary 1. *Assume that* $\lambda_{\min}(H) > 0$. *Denote* $\kappa := \lambda_{\max}(H)/\lambda_{\min}(H)$. *Set* $\eta = 1/\lambda_{\max}(H)$ *and set* $\beta = \left(1-\frac{1}{2}\sqrt{\eta\lambda_{\min}(H)}\right)^2 = \left(1-\frac{1}{2\sqrt{\kappa}}\right)^2$. *Then,* $C_0 \leq 4\sqrt{\kappa}$.

3.6. Meta theorem

Let $\lambda > 0$ be the smallest eigenvalue of the matrix H that appears on the residual dynamics (8). Our goal is to show that the residual errors satisfy

$$\left\|\begin{bmatrix}\xi_s \\ \xi_{s-1}\end{bmatrix}\right\| \leq \left(\sqrt{\beta}+\mathbf{1}_\varphi C_2\right)^s (C_0 + \mathbf{1}_\varphi C_1)\left\|\begin{bmatrix}\xi_0 \\ \xi_{-1}\end{bmatrix}\right\|, \tag{11}$$

where C_0 is the constant defined on (10), and $C_1, C_2 \geq 0$ are some constants, $\mathbf{1}_\varphi$ is an indicator if any φ_t on the residual dynamics (8) is a non-zero vector. For the case of training the neural networks, we have $\mathbf{1}_\varphi = 1$.

Theorem 6. *(Meta theorem for the residual dynamics (8)) Assume that the step size* η *and the momentum parameter* β *satisfying* $1 \geq \beta > \max\{\left(1-\sqrt{\eta\lambda_{\min}(H)}\right)^2, \left(1-\sqrt{\eta\lambda_{\max}(H)}\right)^2\}$, *are set appropriately so that (11) holds at iteration* $s = 0, 1, \ldots, t-1$ *implies that*

$$\|\textstyle\sum_{s=0}^{t-1} A^{t-s-1}\begin{bmatrix}\varphi_s \\ 0\end{bmatrix}\| \leq \left(\sqrt{\beta}+\mathbf{1}_\varphi C_2\right)^t C_3 \left\|\begin{bmatrix}\xi_0 \\ \xi_{-1}\end{bmatrix}\right\|. \tag{12}$$

Then, we have

$$\left\|\begin{bmatrix}\xi_t \\ \xi_{t-1}\end{bmatrix}\right\| \leq \left(\sqrt{\beta}+\mathbf{1}_\varphi C_2\right)^t (C_0 + \mathbf{1}_\varphi C_1)\left\|\begin{bmatrix}\xi_0 \\ \xi_{-1}\end{bmatrix}\right\|, \tag{13}$$

holds for all t, *where* C_0 *is defined on (10) and* $C_1, C_2, C_3 \geq 0$ *are some constants satisfying:*

$$\left(\sqrt{\beta}\right)^t C_0 + \left(\sqrt{\beta}+\mathbf{1}_\varphi C_2\right)^t \mathbf{1}_\varphi C_3 \leq \left(\sqrt{\beta}+\mathbf{1}_\varphi C_2\right)^t (C_0 + \mathbf{1}_\varphi C_1). \tag{14}$$

Proof. The proof is by induction. At $s = 0$, (11) holds since $C_0 \geq 1$ by Theorem 5. Now assume that the inequality holds at $s = 0, 1, \ldots, t-1$. Consider iteration t. Recursively expanding the dynamics (8), we have

$$\begin{bmatrix}\xi_t \\ \xi_{t-1}\end{bmatrix} = A^t \begin{bmatrix}\xi_0 \\ \xi_{-1}\end{bmatrix} + \sum_{s=0}^{t-1} A^{t-s-1}\begin{bmatrix}\varphi_s \\ 0\end{bmatrix}. \tag{15}$$

By Theorem 5, the first term on the r.h.s. of (15) can be bounded by

$$\|A^t \begin{bmatrix}\xi_0 \\ \xi_{-1}\end{bmatrix}\| \leq \left(\sqrt{\beta}\right)^t C_0 \|\begin{bmatrix}\xi_0 \\ \xi_{-1}\end{bmatrix}\| \tag{16}$$

By assumption, given (11) holds at $s = 0, 1, \ldots, t-1$, we have (12). Combining (12), (14), (15), and (16), we have (13) and hence the proof is completed.

□

Remark: As shown in the proof, we need the residual errors be tightly bounded as (11) in each iteration. Theorem 5 is critical for establishing the desired result. On the other hand, it would become tricky if instead we use Gelfand's formula or other techniques in the related works that lead to a convergence rate in the form of $O(t\theta^t)$.

4. Main results

The important lemmas and theorems in the previous section help to show our main results in the following subsections. The high-level idea to obtain the results is by using the meta theorem (i.e. Theorem 6). Specifically, we will need to show that if the underlying residual dynamics satisfy (11) for all the previous iterations, then the terms $\{\varphi_s\}$ in the dynamics satisfy (12). This condition trivially holds for the case of the quadratic problems, since there is no such term. On the other hand, for solving the other problems, we need to carefully show that the condition holds. For example, according to Lemma 3, showing acceleration for the ReLU network will require bounding terms like $\|(H_0 - H_s)\xi_s\|$ (and other terms as well), where $H_0 - H_s$ corresponds to the difference of the kernel matrix at two different time steps. By controlling the width of the network, we can guarantee that the change is not too much. A similar result can be obtained for the problem of the deep linear network. The high-level idea is simple but the analysis of the problems of the neural networks can be tedious.

4.1. Non-asymptotic accelerated linear rate for solving strongly convex quadratic problems

Theorem 7. *Assume the momentum parameter β satisfies $1 \geq \beta > \max\{\left(1-\sqrt{\eta\mu}\right)^2, \left(1-\sqrt{\eta\alpha}\right)^2\}$. Gradient descent with Polyak's momentum for solving (1) has*

$$\left\| \begin{bmatrix} w_t - w_* \\ w_{t-1} - w_* \end{bmatrix} \right\| \leq \left(\sqrt{\beta}\right)^t C_0 \left\| \begin{bmatrix} w_0 - w_* \\ w_{-1} - w_* \end{bmatrix} \right\|, \quad (17)$$

where the constant C_0 is defined as

$$C_0 := \frac{\sqrt{2}(\beta+1)}{\sqrt{\min\{h(\beta,\eta\lambda_{\min}(\Gamma)), h(\beta,\eta\lambda_{\max}(\Gamma))\}}} \geq 1, \quad (18)$$

and $h(\beta, z) = -\left(\beta - (1-\sqrt{z})^2\right)\left(\beta - (1+\sqrt{z})^2\right)$. Consequently, if the step size $\eta = \frac{1}{\alpha}$ and the momentum parameter $\beta = \left(1 - \frac{1}{2\sqrt{\kappa}}\right)^2$, then it has

$$\left\| \begin{bmatrix} w_t - w_* \\ w_{t-1} - w_* \end{bmatrix} \right\| \leq \left(1 - \frac{1}{2\sqrt{\kappa}}\right)^t 4\sqrt{\kappa} \left\| \begin{bmatrix} w_0 - w_* \\ w_{-1} - w_* \end{bmatrix} \right\|. \quad (19)$$

Furthermore, if $\eta = \frac{4}{(\sqrt{\mu}+\sqrt{\alpha})^2}$ and β approaches $\beta \to \left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^2$ from above, then it has a convergence rate approximately $\left(1 - \frac{2}{\sqrt{\kappa}+1}\right)$ as $t \to \infty$.

The convergence rates shown in the above theorem do not depend on the unknown sequence $\{\epsilon_t\}$. Moreover, the rates depend on the squared root of the condition number $\sqrt{\kappa}$. We have hence established a non-asymptotic accelerated linear rate of Polyak's momentum, which helps to show the advantage of Polyak's momentum over vanilla gradient descent in the finite t regime. Our result also recovers the rate $\left(1 - \frac{2}{\sqrt{\kappa}+1}\right)$ asymptotically under the same choices of the parameters as the previous works. The detailed proof can be found in Appendix D, which is actually a trivial application of Lemma 1, Theorem 6, and Corollary 1 with $C_1 = C_2 = C_3 = 0$.

4.2. Non-asymptotic accelerated linear rate of the local convergence for solving $f(\cdot) \in F^2_{\mu,\alpha}$

Here we provide a local acceleration result of the discrete-time Polyak's momentum for general smooth strongly convex and twice differentiable function $F^2_{\mu,\alpha}$. Compared to Theorem 9 of (Polyak, 1964), Theorem 8 clearly indicates the required distance that ensures an acceleration when the iterate is in the neighborhood of the global minimizer. Furthermore, the rate is in the non-asymptotic sense instead of the asymptotic one. We defer the proof of Theorem 8 to Appendix E.

Theorem 8. *Assume that the function $f(\cdot) \in F^2_{\mu,\alpha}$ and its Hessian is α-Lipschitz. Denote the condition number $\kappa := \frac{\alpha}{\mu}$. Suppose that the initial point satisfies $\left\| \begin{bmatrix} w_0 - w_* \\ w_{-1} - w_* \end{bmatrix} \right\| \leq \frac{1}{683\kappa^{3/2}}$. Then, Gradient descent with Polyak's momentum with the step size $\eta = \frac{1}{\alpha}$ and the momentum parameter $\beta = \left(1 - \frac{1}{2\sqrt{\kappa}}\right)^2$ for solving $\min_w f(w)$ has*

$$\left\| \begin{bmatrix} w_{t+1} - w_* \\ w_t - w_* \end{bmatrix} \right\| \leq \left(1 - \frac{1}{4\sqrt{\kappa}}\right)^{t+1} 8\sqrt{\kappa} \left\| \begin{bmatrix} w_0 - w_* \\ w_{-1} - w_* \end{bmatrix} \right\|, \quad (20)$$

where $w_* = \arg\min_w f(w)$.

4.3. Acceleration for training $\mathcal{N}_W^{\mathbf{ReLU}}(x)$

Before introducing our result of training the ReLU network, we need the following lemma.

Lemma 5. *[Lemma 3.1 in Du et al. (2019b) and Song & Yang (2019)] Set $m = \Omega(\lambda^{-2} n^2 \log(n/\delta))$. Suppose that the neurons $w_0^{(1)}, \ldots, w_0^{(m)}$ are i.i.d. generated by $N(0, I_d)$ initially. Then, with probability at least $1-\delta$, it holds that*

$$\|H_0 - \bar{H}\|_F \leq \frac{\lambda_{\min}(\bar{H})}{4}, \ \lambda_{\min}(H_0) \geq \frac{3}{4}\lambda_{\min}(\bar{H}),$$
$$\text{and} \qquad \lambda_{\max}(H_0) \leq \lambda_{\max}(\bar{H}) + \frac{\lambda_{\min}(\bar{H})}{4}.$$

Lemma 5 shows that by the random initialization, with probability $1-\delta$, the least eigenvalue of the Gram matrix $H := H_0$ defined in Lemma 3 is lower-bounded and the largest eigenvalue is close to $\lambda_{\max}(\bar{H})$. Furthermore, Lemma 5 implies that the condition number of the Gram

matrix H_0 at the initialization $\hat{\kappa} := \frac{\lambda_{\max}(H_0)}{\lambda_{\min}(H_0)}$ satisfies $\hat{\kappa} \leq \frac{4}{3}\kappa + \frac{1}{3}$, where $\kappa := \frac{\lambda_{\max}(\bar{H})}{\lambda_{\min}(\bar{H})}$.

Theorem 9. *(One-layer ReLU network $\mathcal{N}_W^{ReLU}(x)$) Assume that $\lambda := \frac{3\lambda_{\min}(\bar{H})}{4} > 0$ and that $w_0^{(r)} \sim N(0, I_d)$ and a_r uniformly sampled from $\{-1, 1\}$. Denote $\lambda_{\max} := \lambda_{\max}(\bar{H}) + \frac{\lambda_{\min}(\bar{H})}{4}$ and denote $\hat{\kappa} := \lambda_{max}/\lambda = (4\kappa + 1)/3$. Set a constant step size $\eta = \frac{1}{\lambda_{\max}}$, fix momentum parameter $\beta = \left(1 - \frac{1}{2\hat{\kappa}}\right)^2$, and finally set the number of network nodes $m = \Omega(\lambda^{-4} n^4 \kappa^2 \log^3(n/\delta))$. Then, with probability at least $1 - \delta$ over the random initialization, gradient descent with Polyak's momentum satisfies for any t,*

$$\left\| \begin{bmatrix} \xi_t \\ \xi_{t-1} \end{bmatrix} \right\| \leq \left(1 - \frac{1}{4\sqrt{\hat{\kappa}}}\right)^t \cdot 8\sqrt{\hat{\kappa}} \left\| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \right\|. \tag{21}$$

We remark that $\hat{\kappa}$, which is the condition number of the Gram matrix H_0, is within a constant factor of the condition number of $\bar{H}$. Therefore, Theorem 9 essentially shows an accelerated linear rate $\left(1 - \Theta(\frac{1}{\sqrt{\kappa}})\right)$. The rate has an improved dependency on the condition number, i.e. $\sqrt{\kappa}$ instead of κ, which shows the advantage of Polyak's momentum over vanilla GD when the condition number is large. We believe this is an interesting result, as the acceleration is akin to that in convex optimization, e.g. Nesterov (2013); Shi et al. (2018).

Our result also implies that over-parametrization helps acceleration in optimization. To our knowledge, in the literature, there is little theory of understanding why over-parametrization can help training a neural network faster. The only exception that we are aware of is Arora et al. (2018), which shows that the dynamic of vanilla gradient descent for an over-parametrized objective function exhibits some momentum terms, although their message is very different from ours. The proof of Theorem 9 is in Appendix F.

4.4. Acceleration for training $\mathcal{N}_W^{L\text{-linear}}(x)$

Theorem 10. *(Deep linear network $\mathcal{N}_W^{L\text{-linear}}(x)$) Denote $\lambda := \frac{L\sigma^2_{\min}(X)}{d_y}$ and $\kappa := \frac{\sigma^2_{\max}(X)}{\sigma^2_{\min}(X)}$. Set a constant step size $\eta = \frac{d_y}{L\sigma^2_{\max}(X)}$, fix momentum parameter $\beta = \left(1 - \frac{1}{2\sqrt{\kappa}}\right)^2$, and finally set a parameter m that controls the width $m \geq C\frac{\kappa^5}{\sigma^2_{\max}(X)}\left(d_y(1 + \|W^*\|_2^2) + \log(\bar{r}/\delta)\right)$ and $m \geq \max\{d_x, d_y\}$ for some constant $C > 0$. Then, with probability at least $1-\delta$ over the random orthogonal initialization, gradient descent with Polyak's momentum satisfies for any t,*

$$\left\| \begin{bmatrix} \xi_t \\ \xi_{t-1} \end{bmatrix} \right\| \leq \left(1 - \frac{1}{4\sqrt{\kappa}}\right)^t \cdot 8\sqrt{\kappa} \left\| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \right\|. \tag{22}$$

Compared with Theorem 4 of Hu et al. (2020b) for vanilla GD, our result clearly shows the acceleration via Polyak's momentum. Furthermore, the result suggests that the depth does not hurt optimization. Acceleration is achieved for any depth L and the required width m is independent of the depth L as Hu et al. (2020b); Zou et al. (2020) (of vanilla GD). The proof of Theorem 10 is in Appendix G.

5. Conclusion

We show some non-asymptotic acceleration results of the discrete-time Polyak's momentum in this paper. The results not only improve the previous results in convex optimization but also establish the first time that Polyak's momentum has provable acceleration for training certain neural networks. We analyze all the acceleration results from a modular framework. We hope the framework can serve as a building block towards understanding Polyak's momentum in a more unified way.

6. Acknowledgment

The authors thank Daniel Pozo for catching a typo. The authors acknowledge support of NSF IIS Award 1910077. JW also thanks IDEaS-TRIAD Research Scholarship 03GR10000818.

References

Alacaoglu, A., Malitsky, Y., Mertikopoulos, P., and Cevher, V. A new regret analysis for adam-type algorithms. *ICML*, 2020.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via overparameterization. *ICML*, 2019.

Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by overparameterization. *ICML*, 2018.

Arora, S., Cohen, N., Golowich, N., and Hu, W. A convergence analysis of gradient descent for deep linear neural networks. *ICLR*, 2019a.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. *NerurIPS*, 2019b.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *NeurIPS*, 2019c.

Aujol, J.-F., Dossal, C., and Rondepierre, A. Convergence rates of the heavy-ball method with lojasiewicz property. *hal-02928958*, 2020.

Bai, Y. and Lee, J. D. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *ICLR*, 2020.

Bietti, A. and Mairal, J. On the inductive bias of neural tangent kernels. *NeurIPS*, 2019.

Brutzkus, A. and Globerson, A. Globally optimal gradient descent for a convnet with gaussian inputs. *ICML*, 2017.

Cai, T., Gao, R., Hou, J., Chen, S., Wang, D., He, D., Zhang, Z., and Wang, L. A gram-gauss-newton method learning overparameterized deep neural networks for regression problems. *arXiv.org:1905.11675*, 2019.

Can, B., Gürbüzbalaban, M., and Zhu, L. Accelerated linear convergence of stochastic momentum methods in wasserstein distances. *ICML*, 2019.

Chen, S., He, H., and Su, W. J. Label-aware neural tangent kernel: Toward better generalization and local elasticity. *NeurIPS*, 2020a.

Chen, Z., Cao, Y., Gu, Q., and Zhang, T. A generalized neural tangent kernel analysis for two-layer neural network. *NeurIPS*, 2020b.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. *NeurIPS*, 2019.

Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex sgd. *NeurIPS*, 2019.

Daniely, A. Sgd learns the conjugate kernel class of the network. *NeurIPS*, 2017.

Daniely, A. Memorizing gaussians with no over-parameterizaion via gradient decent on neural networks. *arXiv:1909.11837*, 2020.

Danilova, M., Kulakova, A., and Polyak, B. Non-monotone behavior of the heavy ball method. *arXiv:1811.00658*, 2018.

Diakonikolas, J. and Jordan, M. I. Generalized momentum-based methods: A hamiltonian perspective. *arXiv:1906.00436*, 2019.

Du, S. S. and Hu, W. Width provably matters in optimization for deep linear neural networks. *ICML*, 2019.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. *ICML*, 2019a.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *ICLR*, 2019b.

Dukler, Y., Gu, Q., and Montufar, G. Optimization theory for relu neural networks trained with normalization layers. *ICML*, 2020.

Fang, C., Dong, H., and Zhang, T. Over parameterized two-level neural networks can learn near optimal feature representations. *arXiv:1910.11508*, 2019.

Flammarion, N. and Bach, F. From averaging to acceleration, there is only a step-size. *COLT*, 2015.

Foucart, S. Matrix norms and spectral radii. *Online lecture note*, 2018.

Franca, G., Sulam, J., Robinson, D. P., and Vidal, R. Conformal symplectic and relativistic optimization. *Journal of Statistical Mechanics: Theory and Experiment*, 2020.

Gadat, S., Panloup, F., and Saadane, S. Stochastic heavy ball. *arXiv:1609.04228*, 2016.

Ge, R., Kuditipudi, R., Li, Z., and Wang, X. Learning two-layer neural networks with symmetric inputs. *ICLR*, 2019.

Gelfand, I. Normierte ringe. *Mat. Sbornik*, 1941.

Ghadimi, E., Feyzmahdavian, H. R., and Johansson, M. Global convergence of the heavy-ball method for convex optimization. *ECC*, 2015.

Ghorbani, B., Mei, S., Misiakiewicz, T., , and Montanari, A. Linearized two-layers neural networks in high dimension. *arXiv:1904.12191*, 2019.

Gidel, G., Bach, F., and Lacoste-Julien, S. Implicit regularization of discrete gradient dynamics in linear neural networks. *NeurIPS*, 2019.

Gitman, I., Lang, H., Zhang, P., and Xiao, L. Understanding the role of momentum in stochastic gradient methods. *NeurIPS*, 2019.

Goh, G. Why momentum really works. *Distill*, 2017.

Gunasekar, S., Woodworth, B., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. *NeurIPS*, 2017.

Hanin, B. and Nica, M. Finite depth and width corrections to the neural tangent kernel. *ICLR*, 2020.

Hardt, M. and Ma, T. Identity matters in deep learning. *ICLR*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hu, B. Unifying the analysis in control and optimization via semidefinite programs. *Lecture Note*, 2020.

Hu, W., Xiao, L., Adlam, B., and Pennington, J. The surprising simplicity of the early-time learning dynamics of neural networks. *NeurIPS*, 2020a.

Hu, W., Xiao, L., and Pennington, J. Provable benefit of orthogonal initialization in optimizing deep linear networks. *ICLR*, 2020b.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 2018.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. *ICLR*, 2019.

Ji, Z. and Telgarsky, M. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *ICLR*, 2020.

Kawaguchi, K. Deep learning without poor local minima. *NeurIPS*, 2016.

Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. M. On the insufficiency of existing momentum schemes for stochastic optimization. *ICLR*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.

Krichene, W., Caluyay, K. F., and Halder, A. Global convergence of second-order dynamics in two-layer neural networks. *arXiv:2006.07867*, 2020.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012.

Laurent, T. and von Brecht, J. Deep linear networks with arbitrary loss: All local minima are global. *ICML*, 2018.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019.

Lee, J. D., Shen, R., Song, Z., Wang, M., and Yu, Z. Generalized leverage score sampling for neural networks. *arXiv:2009.09829*, 2020.

Lessard, L., Recht, B., and Packard, A. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 2016.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. *NeurIPS*, 2018.

Li, Y. and Yuan, Y. Convergence analysis of two-layer neural networks with relu activation. *NeurIPS*, 2017.

Li, Y., Ma, T., and Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. *COLT*, 2018.

Li, Y., Ma, T., and Zhang, H. Learning over-parametrized two-layer relu neural networks beyond ntk. *COLT*, 2020.

Liu, C. and Belkin, M. Parametrized accelerated methods free of condition number. *arXiv:1802.10235*, 2018.

Liu, C., Zhu, L., and Belkin, M. On the linearity of large non-linear models: when and why the tangent kernel is constant. *arXiv:2010.01092*, 2020a.

Liu, C., Zhu, L., and Belkin, M. Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. *arXiv:2003.00307*, 2020b.

Liu, Y., Gao, Y., and Yin, W. An improved analysis of stochastic gradient descent with momentum. *NeurIPS*, 2020c.

Loizou, N. and Richtárik, P. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv:1712.09677*, 2017.

Loizou, N. and Richtárik, P. Accelerated gossip via stochastic heavy ball method. *Allerton*, 2018.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *ICLR*, 2019.

Lu, H. and Kawaguchi, K. Depth creates no bad local minima. *arXiv:1702.08580*, 2017.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive gradient methods with dynamic bound of learning rate. *ICLR*, 2019.

Lyu, K. and Li, J. Gradient descent maximizes the margin of homogeneous neural networks. *ICLR*, 2020.

Mai, V. V. and Johansson, M. Convergence of a stochastic gradient method with momentum for non-smooth non-convex optimization. *ICML*, 2020.

Mitliagkas, I. Accelerated methods - polyaks momentum (heavy ball method). *Online Lecture Note*, 2019.

Moroshko, E., Gunasekar, S., Woodworth, B., Lee, J. D., Srebro, N., and Soudry, D. Implicit bias in deep linear classification: Initialization scale vs training accuracy. *NeurIPS*, 2020.

Nesterov, Y. Introductory lectures on convex optimization: a basic course. *Springer*, 2013.

Oymak, S. and Soltanolkotabi, M. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *arXiv:1902.04674*, 2019.

Panigrahi, A., Shetty, A., and Goyal, N. Effect of activation functions on the training of overparametrized neural nets. *ICLR*, 2020.

Pilanci, M. and Ergen, T. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. *ICML*, 2020.

Polyak, B. Gradient methods for minimizing functionals. *Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki*, 1963.

Polyak, B. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964.

Razin, N. and Cohen, N. Implicit regularization in deep learning may not be explainable by norms. *NeurIPS2020*, 2020.

Recht, B. Lyapunov analysis and the heavy ball method. *Lecture note*, 2018.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. *ICLR*, 2018.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ICLR*, 2014.

Scieur, D. and Pedregosa, F. Universal average-case optimality of polyak momentum. *ICML*, 2020.

Shamir, O. Exponential convergence time of gradient descent for one-dimensional deep linear neural networks. *COLT*, 2019.

Shi, B., Du, S. S., Jordan, M. I., and Su, W. J. Understanding the acceleration phenomenon via high-resolution differential equations. *arXiv:1810.08907*, 2018.

Soltanolkotabi, M. Learning relus via gradient descent. *NeurIPS*, 2017.

Song, Z. and Yang, X. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv:1906.03593*, 2019.

Su, L. and Yang, P. On learning over-parameterized neural networks: A functional approximation perspective. *NeurIPS*, 2019.

Sun, T., Yin, P., Li, D., Huang, C., Guan, L., and Jiang, H. Non-ergodic convergence analysis of heavy-ball algorithms. *AAAI*, 2019.

Tian, Y. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. *ICML*, 2017.

van den Brand, J., Peng, B., Song, Z., and Weinstein, O. Training (overparametrized) neural networks in near-linear time. *arXiv:2006.11648*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., and et al. Attention is all you need. *NeurIPS*, 2017.

Wang, J.-K., Lin, C.-H., and Abernethy, J. Escaping saddle points faster with stochastic momentum. *ICLR*, 2020.

Wei, C., Lee, J. D., Liu, Q., and Ma, T. Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel. *NeurIPS*, 2019.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., , and Recht., B. The marginal value of adaptive gradient methods in machine learning. *NeurIPS*, 2017.

Wilson, A. C., Jordan, M., and Recht, B. A lyapunov analysis of momentum methods in optimization. *JMLR*, 2021.

Wu, L., Wang, Q., and Ma, C. Global convergence of gradient descent for deep linear residual networks. *NeurIPS*, 2019a.

Wu, S., Dimakis, A. G., and Sanghavi, S. Learning distributions generated by one-layer relu networks. *NeurIPS*, 2019b.

Wu, X., Du, S. S., and Ward, R. Global convergence of adaptive gradient methods for an over-parameterized neural network. *arXiv:1902.07111*, 2019c.

Yang, G. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv:1902.04760*, 2019.

Yang, T., Lin, Q., and Li, Z. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *IJCAI*, 2018.

Yehudai, G. and Shamir, O. Learning a single neuron with gradient methods. *COLT*, 2020.

Yun, C., Sra, S., and Jadbabaie, A. Global optimality conditions for deep neural networks. *ICLR*, 2018.

Zhang, G., Martens, J., and Grosse, R. B. Fast convergence of natural gradient descent for over-parameterized neural networks. *NeurIPS*, 2019.