The Expected Loss of Preconditioned Langevin Dynamics Reveals the Hessian Rank

Amitay Bar*, **Rotem Mulayoff***, **Tomer Michaeli**, **Ronen Talmon**

Technion – Israel Institute of Technology, Haifa, Israel
amitayb@campus.technion.ac.il

Abstract

Langevin dynamics (LD) is widely used for sampling from distributions and for optimization. In this work, we derive a closed-form expression for the expected loss of preconditioned LD near stationary points of the objective function. We use the fact that at the vicinity of such points, LD reduces to an Ornstein–Uhlenbeck process, which is amenable to convenient mathematical treatment. Our analysis reveals that when the preconditioning matrix satisfies a particular relation with respect to the noise covariance, LD's expected loss becomes proportional to the rank of the objective's Hessian. We illustrate the applicability of this result in the context of neural networks, where the Hessian rank has been shown to capture the complexity of the predictor function but is usually computationally hard to probe. Finally, we use our analysis to compare SGD-like and Adam-like preconditioners and identify the regimes under which each of them leads to a lower expected loss.

Introduction

Langevin dynamics (LD) has proven to be a powerful tool across many domains. Its basic discretization, the unadjusted Langevin algorithm (ULA), along with other variants, such as the stochastic gradient Langevin dynamics (SGLD) method (Welling and Teh 2011) and its extensions, are commonly used for sampling from distributions (Ding et al. 2014; Wang, Fienberg, and Smola 2015) and for non-convex optimization (Gelfand and Mitter 1991; Raginsky, Rakhlin, and Telgarsky 2017; Xu et al. 2018; Chen, Du, and Tong 2020; Borysenko and Byshkin 2021). LD and similar stochastic differential equations (SDEs) are also used for analyzing the optimization process of neural networks (NNs), as they serve as continuous-time analogues to popular optimizers, like SGD (Arora, Cohen, and Hazan 2018; Elkabetz and Cohen 2021; Latz 2021; Zhu et al. 2019).

An important variant of SGLD is the stochastic gradient Riemannian Langevin dynamics (SGRLD) method (Patterson and Teh 2013), which is an LD-type random process on a Riemannian manifold that respects the Riemannian metric. Girolami and Calderhead (2011) used Riemannian LD as an improved Markov chain Monte Carlo (MCMC) method for sampling from distributions. Here we view SGRLD as a preconditioned version of LD, where the Riemannian metric tensor plays the role of the preconditioner. This more general viewpoint is of practical value, as preconditioning is commonly used for circumventing instabilities that stem from an ill-conditioned loss landscape, in optimization problems in general (Pock and Chambolle 2011; Dauphin, De Vries, and Bengio 2015), and in NN training in particular (Kingma and Ba 2014; Tieleman and Hinton 2012). Combining preconditioning and SGLD has been previously studied in the past (Li et al. 2016; Marceau-Caron and Ollivier 2017). For example, Marceau-Caron and Ollivier (2017) used a Fisher matrix approximation for choosing the variance of the noise in preconditioned SGLD.

In this paper, we consider a preconditioned LD and study its expected loss near stationary points on a large time scale. Specifically, we use a quadratic approximation of the loss about the stationary point, leading to an Ornstein–Uhlenbeck (OU) process, which is mathematically tractable. We show that when the preconditioner and the noise covariance satisfy a particular relation, the expected loss is linear in the Hessian rank. The Hessian of the loss of NNs has attracted a lot of interest in recent years, as it has been linked to model "complexity" (Arora et al. 2019; Li, Luo, and Lyu 2020) and generalization (Huh et al. 2022) and has been shown to exhibit interesting spectral properties (Papyan 2020). Yet, it is typically infeasible to compute the rank of the Hessian of a NN (or even just store the Hessian) due to the large number of parameters. Our theoretical result suggests that we can estimate the Hessian rank at minima by simply applying LD in its vicinity. Leveraging our result, we devise an iterative Hessian rank estimation algorithm, which does not require spectral decomposition of the Hessian. Remarkably, the preconditioning matrix and the covariance matrix revealing the Hessian rank lead to a *Riemannian* LD with a Riemannian metric that is equal to the inverse of the preconditioner.

Additionally, we show that under certain conditions, the expected loss at a large time scale depends only on the interplay between the preconditioner and the noise covariance. This simple result enables us to analyze the impact of different preconditioners on the expected loss. Specifically, we examine two preconditioning matrices which correspond to stochastic gradient descent (SGD) (Robbins and

*These authors contributed equally.

Monro 1951) and Adam (Kingma and Ba 2014). Next, we derive conditions on the preconditioning that lead to the maximal expected loss. From the standpoint of NNs with a non-convex objective function, higher loss indicates better escaping efficiency from local minima (Zhu et al. 2019). For completeness, we also consider initialization at a saddle point and show that another derivative of our analysis is the ability of the preconditioned LD to escape saddle points.

We empirically demonstrate our theory on linear and non-linear NNs. We show that the expected loss of the networks incorporated into preconditioned LD with specific preconditioning results in an accurate estimation of the Hessian rank.

Preconditioned Langevin Dynamics

We consider a preconditioned LD given by the SDE

$$d\boldsymbol{\theta}_t = -\boldsymbol{G}\nabla f(\boldsymbol{\theta}_t)dt + \boldsymbol{G}\boldsymbol{\Sigma}^{\frac{1}{2}}d\boldsymbol{n}_t, \tag{1}$$

where $\boldsymbol{\theta}_t \in \mathbb{R}^n$, $f(\cdot)$ is a scalar objective function and $\nabla f(\boldsymbol{\theta}_t)$ is its gradient, $\boldsymbol{G} \in \boldsymbol{S}^n_{++}$ is a positive definite (PD) preconditioning matrix, $\boldsymbol{\Sigma}$ is a noise covariance matrix, and $d\boldsymbol{n}_t$ is a standard Brownian motion. The SDE in (1) stems from the Riemannian LD presented in (Girolami and Calderhead 2011; Patterson and Teh 2013). Specifically, when $\boldsymbol{\Sigma} = \boldsymbol{G}^{-1}$, the SDE in (1) coincides with LD on a flat Riemannian manifold, where $\boldsymbol{G}^{-1}$ is the metric tensor matrix. Note that in (1), we decouple the preconditioning matrix $\boldsymbol{G}$ and the noise covariance $\boldsymbol{\Sigma}$. This allows us to model a user-chosen preconditioning matrix $\boldsymbol{G}$ multiplying gradient estimation noise with covariance $\boldsymbol{\Sigma}$.

The Riemannian LD has been considered in many studies. For example, Girolami and Calderhead (2011) used it for improving sequential MCMC algorithms, and Li et al. (2016) used it for efficient sampling from the posterior. Broadly, SDEs such as (1) naturally model gradient flow schemes, which are often viewed as the continuous counterparts of gradient descent schemes. Therefore, such SDEs were recently used for analyzing the optimization process of NNs (Arora, Cohen, and Hazan 2018; Elkabetz and Cohen 2021; Latz 2021). In that context, $\boldsymbol{\theta}$ is the vector of parameters of the network, $f(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta})$ is the loss function, $\boldsymbol{G}$ is the user-chosen preconditioner, and $d\boldsymbol{n}$ accounts for the gradient noise that arises due to the use of small batches. Alternatively, $d\boldsymbol{n}$ can be synthetic noise added to the gradient for better learning and generalization (Neelakantan et al. 2015; Kaiser and Sutskever 2015; Zeyer et al. 2017).

We explore the effect of preconditioning and the Hessian rank on the expected loss of the preconditioned LD. We remark that selecting the preconditioner $\boldsymbol{G}$ embodies the flexibility to select its magnitude and direction. Since the magnitude can be seen as the continuous counterpart of the adjustment of the discrete stepsize (learning rate), we maintain the norm of the preconditioner fixed, thereby allowing for the examination of the impact of the direction.

To make the analysis tractable, we consider the second-order approximation of $f(\boldsymbol{\theta})$ around a stationary point, $\boldsymbol{\theta}^*$, for which $\nabla f(\boldsymbol{\theta}^*) = 0$, namely,

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T\boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \tag{2}$$

where $\boldsymbol{H} = \nabla^2 f(\boldsymbol{\theta}^*) \in \boldsymbol{S}^n(\mathbb{R})$ is the Hessian of f at $\boldsymbol{\theta}^*$. Without loss of generality, we assume that $\boldsymbol{\theta}^* = \boldsymbol{0}$ and $f(\boldsymbol{\theta}^*) = 0$ (Chen, Du, and Tong 2020; Zhu et al. 2019). Such noisy quadratic models (2) were considered in the past to model NN optimization, and it was shown that despite their simplicity, they capture important features of non-trivial NNs that are used in practice (Zhang et al. 2019). We note that the second-order approximation is made only for analysis and not in our experiments.

Under the noisy quadratic model at the vicinity of $\boldsymbol{\theta}^*$, the SDE in (1) becomes the following multivariate OU process (Gardiner et al. 1985), given by

$$d\boldsymbol{\theta}_t = -\boldsymbol{G}\boldsymbol{H}\boldsymbol{\theta}_t dt + \boldsymbol{G}\boldsymbol{\Sigma}^{\frac{1}{2}}d\boldsymbol{n}_t. \tag{3}$$

An analysis of (3) without preconditioning, *i.e.*, for $\boldsymbol{G} = \boldsymbol{I}$, appears in (Zhu et al. 2019) for a short time scale. In contrast, here, we specifically analyze the effect of the preconditioning over long time scales.

We conclude this section by noting that the preconditioner in (1) multiplies both the gradient and the noise. In case we have access to the accurate gradient, it was proposed to add noise to escape local minima and saddle points (Chen, Du, and Tong 2020; Choi et al. 2023). This leads to the following SDE, where $\boldsymbol{G}$ multiplies only the gradient

$$d\boldsymbol{\theta}_t = -\boldsymbol{G}\nabla\mathcal{L}(\boldsymbol{\theta}_t)dt + \boldsymbol{\Sigma}^{\frac{1}{2}}d\boldsymbol{n}_t. \tag{4}$$

With only slight changes to our analysis, similar results could be obtained for the SDE in (4) as well.

Analyzing the Expected Loss

In this section, we analyze the expected loss induced by the SDE in (3) and present the effects of different preconditioning matrices. The proofs appear in the supplementary materials (SM) included in the arXiv version of the paper.

We begin by examining the expected loss near a stationary point induced by the LD in (3) for a loss function with an arbitrary Hessian matrix $\boldsymbol{H}$, representing either a minimum point or a saddle point.

Theorem 1 (**Expected loss over time**). *The expected loss of a process governed by the SDE in (3) is given by*

$$\mathbb{E}[f(\boldsymbol{\theta}_t)] = \frac{1}{4}\mathrm{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{G}\left(\boldsymbol{I} - e^{-2\boldsymbol{G}\boldsymbol{H}t}\right)\right). \tag{5}$$

The expected loss has been previously investigated considering $\boldsymbol{G} = \boldsymbol{I}$. Indeed, when setting $\boldsymbol{G} = \boldsymbol{I}$ in (5), the result coincides with the result presented in (Zhu et al. 2019; Chen, Du, and Tong 2020). Considering a general preconditioner $\boldsymbol{G}$ requires a more involved derivation since the symmetry of the matrices breaks. Our key observation is that this can be circumvented by using matrix similarity. See more details in the SM.

Next, we consider a minimum point with a positive semi-definite (PSD) Hessian, namely, the Hessian eigenvalues are larger than or equal to zero. Taking the limit of $t \to \infty$ in (5) leads to our main result.

Proposition 1 (**Expected loss for large** t). *When the Hessian is PSD, we have:*

$$\lim_{t\to\infty}\mathbb{E}[f(\boldsymbol{\theta}_t)] = \frac{1}{4}\mathrm{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{G}\boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}\right), \tag{6}$$

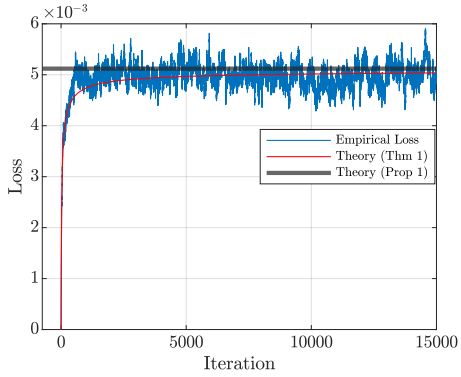

Figure 1: The loss (in blue) for a linear NN of depth 5 with input and output dimensions of 32 (the number of parameters is 5120). The theoretical expressions according to Theorem 1 and Proposition 1 are in red and black, respectively.

where $\boldsymbol{J}$ is the following diagonal matrix

$$\boldsymbol{J}_{ii} = \begin{cases} 1 & \lambda_i\{\boldsymbol{GH}\} > 0 \\ 0 & \lambda_i\{\boldsymbol{GH}\} = 0 \end{cases}, \quad i = 1, \ldots, n, \tag{7}$$

and $\boldsymbol{P}$ is a matrix whose columns are the eigenvectors of $\boldsymbol{GH}$, ordered according to the eigenvalues.

Note that $\boldsymbol{GH}$ is similar to the symmetric matrix $\boldsymbol{G}^{\frac{1}{2}}\boldsymbol{H}\boldsymbol{G}^{\frac{1}{2}}$, and therefore, it has a real spectrum.

Proposition 1 provides means to prove the interplay between the expected loss, the Hessian matrix $\boldsymbol{H}$, the covariance of the noise $\boldsymbol{\Sigma}$, and the (user-chosen) preconditioner $\boldsymbol{G}$. The limit of $t \to \infty$ in (6) is required so that $e^{-2\lambda_{\min^+}\{\boldsymbol{GH}\}t} \ll 1$, where $\lambda_{\min^+}\{\boldsymbol{GH}\}$ is the smallest non-zero eigenvalue of $\boldsymbol{GH}$. This holds for sufficiently large t that satisfies $t \gg 1/\lambda_{\min^+}\{\boldsymbol{GH}\}$.

We demonstrate the theoretical results in Figure 1, which presents the loss of a linear NN. Since the Hessian of such a network has an explicit expression (Mulayoff and Michaeli 2020), we are able to present the theoretical expression of the expected loss over time according to Theorem 1, and the theoretical steady-state value according to Proposition 1. See details in the Application to Hessian Rank Estimation section.

When the Hessian is strictly positive (*i.e.*, a PD matrix), the following result stems directly from Proposition 1.

Corollary 1 (Expected loss for PD Hessian). *If the Hessian is PD, then*

$$\lim_{t\to\infty} \mathbb{E}[f(\boldsymbol{\theta}_t)] = \frac{1}{4}\mathrm{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{G}\right). \tag{8}$$

According to Corollary 1, when $t \to \infty$, the expected loss reaches a fixed value depending only on the interplay between the preconditioner $\boldsymbol{G}$ and the noise covariance $\boldsymbol{\Sigma}$.

From (8), we see that there exist infinitely many preconditioners that result in the same expected loss because the trace operation imposes only n constraints on an $n \times n$ matrix. In particular, any expected loss can be achieved by a diagonal preconditioning matrix. This is important in the context of deep NN, when the number of parameters is typically very large, and adaptive methods resort to diagonal preconditioning matrices (for example, Adam (Kingma and Ba 2014)).

The Preconditioner Revealing Hessian Rank

An important consequence of Proposition 1 is that the expected loss reveals the rank of the Hessian for a particular choice of the preconditioner. Specifically, by setting $\boldsymbol{G\Sigma} = \sigma^2\boldsymbol{I}$ in Proposition 1, we get that the expected loss tends to $\sigma^2\mathrm{Tr}(\boldsymbol{J})/4$. Note that $\mathrm{Tr}(\boldsymbol{J})$ is the rank of $\boldsymbol{GH}$. Now, since $\boldsymbol{G}$ has full rank, we have that $\mathrm{rank}(\boldsymbol{GH}) = \mathrm{rank}(\boldsymbol{H})$, which leads us to the following result.

Corollary 2 (Expected loss and Hessian rank). *If $\boldsymbol{\Sigma G} = \sigma^2\boldsymbol{I}$, then*

$$\mathrm{rank}(\boldsymbol{H}) = \lim_{t\to\infty} \frac{4}{\sigma^2}\mathbb{E}[f(\boldsymbol{\theta}_t)]. \tag{9}$$

The importance of Corollary 2 is that it links the Hessian rank, a meaningful quantity that is typically hard to estimate, with the expected loss, which can be computed. In other words, Corollary 2 prescribes a way to estimate the Hessian rank by observing the expected loss.

Remark 1. *Setting $\boldsymbol{\Sigma G} = \sigma^2\boldsymbol{I}$ in the SDE in (1) leads to the following Riemannian LD on a flat manifold (Girolami and Calderhead 2011)*

$$d\boldsymbol{\theta}_t = -\boldsymbol{G}\nabla f(\boldsymbol{\theta}_t)dt + \sigma\sqrt{\boldsymbol{G}}d\boldsymbol{n}_t, \tag{10}$$

where $\boldsymbol{G}^{-1}$ is the Riemannian metric.

In other words, we see that the specific relationship between the gradient and the noise that describes a "natural" process on a manifold reveals the rank. Since the Hessian rank could potentially be large, in practice, we keep the quadratic approximation valid by setting $\sigma^2 \sim \frac{1}{d}$. We note that the preconditioner $\boldsymbol{G}$ is important in circumventing instabilities. In the Application to Hessian Rank Estimation section, we describe in detail the algorithm for Hessian rank estimation that is based on Corollary 2.

Next, we examine the effect of the Hessian rank on the expected loss, while considering arbitrary preconditioners. Here we fix $\boldsymbol{G}$ along with $\boldsymbol{\Sigma}$ and consider two different Hessian matrices with different ranks.

Proposition 2. *For the same preconditioner $\boldsymbol{G}$ and noise covariance matrix $\boldsymbol{\Sigma}$, if*

$$\mathrm{rank}(\boldsymbol{H}_1) \leq \mathrm{rank}(\boldsymbol{H}_2), \tag{11}$$

then

$$\lim_{t\to\infty} \mathbb{E}[f_1(\boldsymbol{\theta}_t)] \leq \lim_{t\to\infty} \mathbb{E}[f_2(\boldsymbol{\theta}_t)]. \tag{12}$$

According to this proposition, a larger Hessian rank indicates a higher asymptotic expected loss. In the context of NNs, Proposition 2 could imply that, typically, *when a NN reaches a low loss during training, the loss has a low-rank Hessian*. In turn, for a low-rank Hessian, the noise directed at the null space of the Hessian does not affect the loss.

The expected loss is related not only to the Hessian rank but also to its trace. Following Theorem 1, the derivative of the expected loss in (5) with respect to the time t is

$$\frac{\partial}{\partial t}\mathbb{E}[f(\boldsymbol{\theta}_t)] = \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{G}\boldsymbol{G}\boldsymbol{H}e^{-2\boldsymbol{GH}t}\right). \tag{13}$$

By setting $\boldsymbol{G} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$ and $t = 0$, we obtain

$$\frac{\partial}{\partial t}\mathbb{E}[f(\boldsymbol{\theta}_t)]\Big|_{t=0} = \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{H}\right). \tag{14}$$

Thus, for the particular preconditioning matrix $\boldsymbol{G} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$, the derivative of the expected loss at time $t = 0$ is proportional to the Hessian trace. See more details in the SM.

Specific Preconditioners and Their Effect on the Expected Loss

The expected loss for $t \to \infty$ depends only on the interplay between the preconditioner and the noise covariance according to Corollary 1[1]. We fix the magnitude of the preconditioner and examine the direction leading to the maximal expected loss. The expected loss is indicative of the ability to escape local minima. In (Zhu et al. 2019), it is used to define the escaping efficiency as $\mu_t \equiv \mathbb{E}_{\boldsymbol{\theta}_t}[f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}_0)]$, where $\boldsymbol{\theta}_0$ is the value of $\boldsymbol{\theta}_t$ for $t = 0$. We note that higher expected loss means better escaping efficiency.

Corollary 3 (Maximal expected loss). *The preconditioner $\boldsymbol{G}$ leading to the maximal expected loss is proportional to the noise covariance matrix. Formally,*

$$\boldsymbol{G}^* = \underset{\boldsymbol{G}\ s.t.\ \|\boldsymbol{G}\|_F=1}{\arg\max}\ \mathbb{E}[f(\boldsymbol{\theta}_t)] = \frac{\boldsymbol{\Sigma}}{\|\boldsymbol{\Sigma}\|_F}. \tag{15}$$

Corollary 3 holds since the trace is an inner product, and hence, the maximum of $\mathrm{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{G}\right)$ is achieved for $\boldsymbol{G} \propto \boldsymbol{\Sigma}$. In other words, we see that a preconditioner that is aligned with the noise covariance results in the highest expected loss.

For PD matrices $\boldsymbol{H}$ and $\boldsymbol{G}$ we have $\mathbb{E}[f(\boldsymbol{\theta}_t)] = \mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{G}) = \mathrm{Tr}(\boldsymbol{G}^{\frac{1}{2}}\boldsymbol{\Sigma}\boldsymbol{G}^{\frac{1}{2}}) > 0$. This means that

$$\mathbb{E}[f(\boldsymbol{\theta}_t)] > f(\boldsymbol{\theta}_0), \tag{16}$$

and after initialization, the expected loss is greater than its initial value.

Next, we focus on two choices of preconditioning matrices: $\boldsymbol{G} = \boldsymbol{I}$ and $\boldsymbol{G} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$ and examine the consequent expected loss. For this purpose, consider two stochastic processes $\boldsymbol{\theta}_t$ and $\boldsymbol{\psi}_t$ that follow the SDE in (3) with preconditioning matrices $\boldsymbol{G} = \boldsymbol{I}$ and $\boldsymbol{G} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$, respectively. We note that $\boldsymbol{\theta}_t$ can be viewed as the continuous counterpart of the SGD algorithm (Robbins and Monro 1951) since the gradient comprises two terms, the accurate gradient ∇f, and a noise term. The process $\boldsymbol{\psi}_t$ is similar in spirit to Adam (Kingma and Ba 2014) since it also considers the square root of the second-order statistics, but instead of the correlation matrix, it makes use of the covariance of the noise. Setting $\boldsymbol{G} = \boldsymbol{I}$ and $\boldsymbol{G} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$ in (8) results in $\mathbb{E}[f(\boldsymbol{\theta}_t)] = \mathrm{Tr}(\boldsymbol{\Sigma})$ and $\mathbb{E}[f(\boldsymbol{\psi}_t)] = \mathrm{Tr}(\boldsymbol{\Sigma}^{\frac{1}{2}})$, respectively. To determine which preconditioner leads to a larger loss, we compare between $\mathrm{Tr}(\boldsymbol{\Sigma})$ and $\mathrm{Tr}(\boldsymbol{\Sigma}^{\frac{1}{2}})$. Generally, when the noise is high, $\mathrm{Tr}(\boldsymbol{\Sigma}) > \mathrm{Tr}(\boldsymbol{\Sigma}^{\frac{1}{2}})$ and $\boldsymbol{G} = \boldsymbol{I}$ leads to a higher loss.

We further investigate the two preconditioners by fixing their magnitude and examining the effect of their direction on the loss.

[1]The results in this subsection are for $t \to \infty$.

Proposition 3. *Suppose the preconditioners of $\boldsymbol{\theta}_t$ and $\boldsymbol{\psi}_t$ have the same Frobenius norm. If*

$$\mathrm{Tr}(\boldsymbol{\Sigma}) > n, \tag{17}$$

then

$$\mathbb{E}[f(\boldsymbol{\theta}_t)] > \mathbb{E}[f(\boldsymbol{\psi}_t)]. \tag{18}$$

We see that the preconditioner leading to a higher expected loss depends only on the power of the noise.

Thus far, we analyzed the expected loss for a minimum point. In Section we consider the use of a preconditioner for a process that is initialized at a saddle point.

Saddle Points

For completeness, we analyze saddle points using our approach, while focusing on the ability of the multivariate OU process from (3) to escape a saddle. To this end, here we assume that the initial point $\boldsymbol{\theta}_0$ is a saddle point.

Definition 1 (Escape time). *The escape time t_{esc} from a saddle point, $\boldsymbol{\theta}_0$, is defined as the first time for which $\mathbb{E}[f(\boldsymbol{\theta}_t)] < f(\boldsymbol{\theta}_0)$, when the process is initialized at $\boldsymbol{\theta}_0$.*

Under this definition, we have the following result.

Proposition 4 (Escaping a saddle point). *The escape time from a saddle point, t_{esc}, is upper bounded by*

$$t_{esc} \leq \frac{\log\left(\frac{\mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{G})}{\lambda_{\min}\{\boldsymbol{\Sigma}\boldsymbol{G}\}}\right)}{|2\lambda_{\min}\{\boldsymbol{G}\boldsymbol{H}\}|}. \tag{19}$$

We note that the matrices $\boldsymbol{G}\boldsymbol{H}$ and $\boldsymbol{\Sigma}\boldsymbol{G}$ are similar to symmetric matrices so they have real spectra. In addition, for $\boldsymbol{G} = \boldsymbol{I}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$, the upper bound coincides with the result presented in Chen, Du, and Tong (2020). According to Proposition 4, for $t > t_{\text{esc}}$, it is guaranteed that $\mathbb{E}[f(\boldsymbol{\theta}_t)] < f(\boldsymbol{\theta}_0)$. Additionally, we see that the interaction between the preconditioning matrix and the Hessian has a greater effect on the escape time than the interaction between the preconditioning matrix and the noise covariance. We remark that escaping saddle points considering adaptive gradient methods was previously explored, for example in (Staib et al. 2019).

Application to Hessian Rank Estimation

Consider a general NN with a vector parameter $\boldsymbol{\theta}$. Suppose that the NN is already trained and that at the end of training the parameters are in the vicinity of some minimum $\boldsymbol{\theta}_0$ of the loss $\mathcal{L}$. Given $\boldsymbol{\theta}_0$, our goal is to estimate the rank of the loss' Hessian at $\boldsymbol{\theta}_0$.

In light of Corollary 2, we propose the following estimator of the Hessian rank

$$\hat{r} = \frac{4}{\sigma^2}\left(\langle\mathcal{L}(\boldsymbol{\theta}_t)\rangle_{\mathcal{K}} - \mathcal{L}(\boldsymbol{\theta}_0)\right), \tag{20}$$

where $\mathcal{L}(\boldsymbol{\theta}_t)$ is the loss of the NN at time t and $\langle\mathcal{L}(\boldsymbol{\theta}_t)\rangle_{\mathcal{K}}$ is the average of the loss over $t \in \mathcal{K}$ for the set of indices $\mathcal{K}$ (taken to be the last K_{avg} iterations). We subtract $\mathcal{L}(\boldsymbol{\theta}_0)$ since in general $\mathcal{L}(\boldsymbol{\theta}_0) \neq 0$. Broadly, estimating the Hessian rank according to (20) requires the computation of

Algorithm 1: Hessian rank estimation

Input: $\boldsymbol{\theta}_0$ (NN weights at the end of the training)
Parameter: σ^2, K_{tot}, K_{avg}
Output: $\hat{r}$ (rank estimate)

1: Choose a fixed preconditioner $\boldsymbol{G}$
2: Set $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{G}^{-1}$
3: Update $\boldsymbol{\theta}_t$ according to (21) with $\boldsymbol{G}$ and $\boldsymbol{\Sigma}$ for K_{tot} iterations
4: Compute $\langle \mathcal{L}(\boldsymbol{\theta}_t) \rangle_{\mathcal{K}}$ over the last K_{avg} iterations
5: Compute $\hat{r}$ using (20)
6: **return** $\hat{r}$

$\langle \mathcal{L}(\boldsymbol{\theta}_t) \rangle_{\mathcal{K}}$. To compute $\langle \mathcal{L}(\boldsymbol{\theta}_t) \rangle_{\mathcal{K}}$, we update the parameters of the NN, $\boldsymbol{\theta}_t$, according to a discretization of the SDE in (1), which is given by the Euler–Maruyama method (Kloeden et al. 1992):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{t+1} \boldsymbol{G} \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \sqrt{\eta_{t+1}} \boldsymbol{G} \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{n}_{t+1}, \quad (21)$$

where $\eta_t > 0$ is the stepsize, $\nabla \mathcal{L}(\boldsymbol{\theta}_t)$ is the gradient of the loss, and the preconditioner $\boldsymbol{G}$ and noise covariance $\boldsymbol{\Sigma}$ are chosen according to Corollary 2. Next, we describe the rank estimation algorithm in detail.

Algorithm

The first step of the algorithm is choosing a preconditioner. We empirically found that for deep NNs, using $\boldsymbol{G} = \boldsymbol{I}$ which corresponds to GD could be unstable (see SM). To circumvent instability, we suggest using the weights of an adaptive gradient method at the minimum point (which in many cases is a byproduct of the training). In our experiments, we use Adam preconditioner to demonstrate the applicability of a commonly used adaptive method. We note that other adaptive gradient methods could also be used. We continue by setting the noise covariance according to $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{G}^{-1}$. Next, the weights of the NN are updated according to (21) with the preconditioner $\boldsymbol{G}$ and the noise covariance $\boldsymbol{\Sigma}$ for K_{tot} iterations. The gradient of the loss $\mathcal{L}(\boldsymbol{\theta}_t)$ is computed using full batch with standard backpropagation. Finally, the average loss during the last K_{avg} iterations out of the total K_{tot} iterations is computed, and the Hessian rank is estimated using the average loss according to (20).

Setting the hyperparameters K_{tot} and K_{avg} can be done by observing the curve of the actual loss during the update of $\boldsymbol{\theta}_t$. The computation of the average loss should begin when the loss reaches a steady-state. This, empirically, does not require many iterations. For example, in the experimental results, we set $K_{\text{tot}} = 1.5 \times 10^4$ and $K_{\text{avg}} = 10^4$. The algorithm is summarized in Algorithm 1.

In terms of complexity, the proposed rank estimation algorithm requires the computation of the full gradient at each iteration. In comparison, the matrix rank estimation of Ubaru and Saad (2016) requires a full Hessian-vector product at each iteration. This involves computing second-order derivatives or approximating the full Hessian-vector product by a numeric computation using backpropagation at two adjacent points. This results in twice the amount of computations at each iteration compared to the proposed approach.

Beyond the scope of NNs, we remark that the proposed approach could be used to estimate the rank of any symmetric matrix $\boldsymbol{S}$ by computing the expected loss of the discretization of LD given in (21) and setting $\nabla \mathcal{L}(\boldsymbol{\theta}_t) = \boldsymbol{S}\boldsymbol{\theta}_t$. However, this direction requires further investigation, exceeding the scope of this paper.

Linear Networks

We start by considering linear NNs in a regression task since there exists an explicit expression for their Hessian at a global minimum. This allows us to evaluate our approach by comparing the Hessian rank estimation with the true rank.

The output of a linear NN is

$$g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}_M \boldsymbol{W}_{M-1} \cdots \boldsymbol{W}_1 \boldsymbol{x}, \quad (22)$$

where $\boldsymbol{W}_i$ denotes the weights of the ith layer, and M denotes the depth. Here the vector parameter $\boldsymbol{\theta}$ is just a concatenation of the vectorizations of all the $\boldsymbol{W}_i$ matrices. In this experiment, we use the quadratic loss, *i.e.*

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{j=1}^{n} \left[\|\boldsymbol{y}_j - g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\|^2 \right], \quad (23)$$

where $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^n$ is a paired training set. We denote by d_x and d_y the dimension of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

In this setting, there is no unique minimum, and the set of global minima is $\{\boldsymbol{\theta} \in \mathbb{R}^n : \boldsymbol{W}_M \boldsymbol{W}_{M-1} \cdots \boldsymbol{W}_1 = \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_x^{-1}\}$, where $\boldsymbol{\Sigma}_x$ is the covariance matrix of $\boldsymbol{x}$ and $\boldsymbol{\Sigma}_{xy}$ denotes the cross-covariance matrix between $\boldsymbol{x}$ and $\boldsymbol{y}$. The Hessian at a global minimum is given by (Mulayoff and Michaeli 2020)

$$\boldsymbol{H} = 2 \boldsymbol{\Phi} \boldsymbol{\Phi}^\top, \quad (24)$$

where $\boldsymbol{\Phi} = [\boldsymbol{\Phi}_1^\top, \boldsymbol{\Phi}_2^\top, \ldots, \boldsymbol{\Phi}_M^\top]^\top$ and

$$\boldsymbol{\Phi}_k = \left(\prod_{j=1}^{k-1} \boldsymbol{W}_j \boldsymbol{\Sigma}_x^{\frac{1}{2}} \right) \otimes \left(\prod_{j=k+1}^{M} \boldsymbol{W}_j \boldsymbol{\Sigma}_x^{\frac{1}{2}} \right)^\top. \quad (25)$$

Here $\otimes$ denotes the Kronecker product. From (24) and (25), Mulayoff and Michaeli (2020) showed that the Hessian rank equals the dimension of the input multiplied by the dimension of the output, *i.e.*, $\operatorname{rank}(\boldsymbol{H}) = d_x d_y$, regardless of the depth of the linear network. We leverage this known Hessian rank to evaluate the performance of our approach.

In all the experiments, we set the depth $M = 5$, and w.l.o.g. consider $d_x = d_y = d$. The input $\boldsymbol{x}$ is assumed to be random with a covariance matrix $\boldsymbol{\Sigma}_x = \boldsymbol{I}$. The cross-covariance matrix between $\boldsymbol{x}$ and $\boldsymbol{y}$, denoted by $\boldsymbol{\Sigma}_{xy}$, is arbitrarily chosen at random.

Figure 1 presents the loss of a linear NN with input dimensions of $d = 32$ and $Md^2 = 5120$ parameters. The weights are initialized at one of the global minima, randomly chosen. We set $\boldsymbol{G} = \boldsymbol{I}$, $\eta = 10^{-4}$ and $\sigma_n^2 = 2 \times 10^{-5}$. We remark that for nonlinear NNs, we found that using the Adam preconditioner is preferable to avoid stability issues. The actual loss is the blue curve and the theoretical values of the expected loss according to Theorem 1 and Proposition 1 are the red and black curves, respectively. We see that the theoretical values are in accordance with the actual loss. We

report that the expected loss over time according to Theorem 1 converges to the steady-state value in Proposition 1 for a larger number of iterations.

Next, we evaluate the performance of the Hessian rank estimation for linear NNs with varying dimensions. We consider six different networks with input and output dimensions of $d = \{8, 16, 32, 64, 128, 256\}$. The number of parameters is Md^2 and varies between 320 parameters for $d = 8$ to $327,680$ parameters for $d = 256$. This allows us to evaluate the proposed approach for small-scale and large-scale linear NNs. For each network, we run 100 different rank estimation trials, where at each trial the network is initialized at a randomly chosen global minimum. We follow Algorithm 1 with $K_{\text{tot}} = 1.5 \times 10^4$ and $K_{\text{avg}} = 10^4$ iterations, noise power of $\sigma_2 = 2 \times 10^{-5}$, and the stepsize is $\eta = 10^{-4}$. We compare the performance with the matrix rank estimation method proposed in (Ubaru and Saad 2016) using their published implementation. For a fair comparison, we consider the same number of iterations for both methods, by setting the polynomial degree to 50 and the number of vectors to 300. For brevity, we term their method U&S. We note that the presented results are not sensitive to different choices of hyperparameters. The U&S method requires the Hessian-vector product. To avoid computing the entire Hessian, which for a network of $d = 256$ has over 10^{11} entries, we exploit the known structure of the Hessian and the following property of the Kronecker product

$$\text{vec}(\boldsymbol{M}_1\boldsymbol{M}_2\boldsymbol{M}_3) = (\boldsymbol{M}_3^\top \otimes \boldsymbol{M}_1)\text{vec}(\boldsymbol{M}_2), \qquad (26)$$

for any matrices $\boldsymbol{M}_1, \boldsymbol{M}_2$, and $\boldsymbol{M}_3$. This allows efficient computation of the Hessian-vector product.

Figure 2 presents an error bar of the Hessian rank estimation obtained by the proposed approach in blue and the U&S method in red. The whiskers represent the standard deviation (STD). The gray dashed line is the $y = x$ line representing the accurate rank. We see that the proposed approach leads to accurate rank estimation. In contrast, the U&S method leads to underestimation. The reason is that U&S is based on the estimation of the spectral density, and it requires a threshold that determines the rank. For the involved spectrum of a Hessian of NNs, setting this threshold is challenging. In contrast, the proposed approach does not require the estimation of the spectral density of the Hessian. Additionally, since the rank estimate is proportional to the expected loss, the y-axis in Figure 2 is proportional to the expected loss, and there exists a linear relation between the Hessian rank and the expected loss as Corollary 1 suggests. The small STD implies that the average loss of different trials is similar. Since each trial is initialized at a different minimum with a different Hessian, these results are in accordance with Corollary 1 (even though the Hessian is PSD).

Table 1 presents the root mean square error (RMSE) normalized by the actual Hessian rank for the proposed approach and U&S. We see that the proposed approach leads to a better accuracy by a large margin.

We emphasize that the same hyperparameters are used for all the tested NNs, even though their number of parameters varies drastically. Potentially, improvement in performance could be achieved by adjusting the hyperparameter according to the NN dimension.

Rank	8^2	16^2	32^2	64^2	128^2	256^2
Ours	6.9%	2.8%	1.2%	0.3%	2%	4.6%
U&S	12.2%	18.5%	19%	19.2%	17.9%	18.7%

Table 1: Normalized RMSE of the Hessian rank estimation for different Hessian ranks.

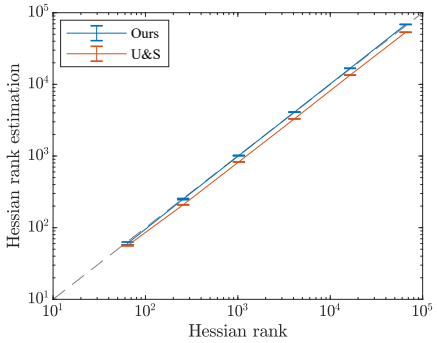

Figure 2: Estimated rank of linear networks with different dimensions. Our method is in blue and the U&S method proposed in Ubaru and Saad (2016) is in red. The error bars represent one standard deviation. We set the same number of iterations for both methods.

Denoising NN

In this section, we demonstrate Algorithm 1 in a real-world application of nonlinear deep neural networks. We train a DnCNN (Zhang et al. 2017) for denoising on the MNIST dataset. The network contains a few parameters (753) so that the exact computation of the Hessian and its spectrum for evaluation is feasible. In this experiment, we add a white Gaussian noise $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to the training images and use the MSE loss. To avoid overfitting, in each epoch, we draw new noise realizations. We trained the network using SGD, and finalized the training using GD with a small stepsize to ensure reaching the minimum (for evaluation purposes).

To estimate the Hessian rank at this minimum, we use Algorithm 1 with Adam preconditioner. Specifically, after convergence, we further train the model with Adam to get its second-moment estimation. Then, we fix the preconditioner $\boldsymbol{G}$ of Adam, and run Algorithm 1 for $K_{\text{tot}} = 30 \times 10^3$ iterations with stepsize $\eta = 0.1$, and $\sigma^2 = 10^{-5}$. The last $K_{\text{avg}} = 10^4$ iterations are used to compute the averaged loss.

In this setup, there is no analytic expression for the Hessian rank. Thus, to evaluate the performance of Algorithm 1, we numerically computed the Hessian using automatic differentiation. Due to finite numerical precision, it is expected that the rank of this numerically computed Hessian be falsely larger than the rank of the true Hessian (which we do not have) since eigenvalues that should theoretically be zero are replaced by small nonzero values. Therefore, instead of comparing the estimated rank to this falsely larger

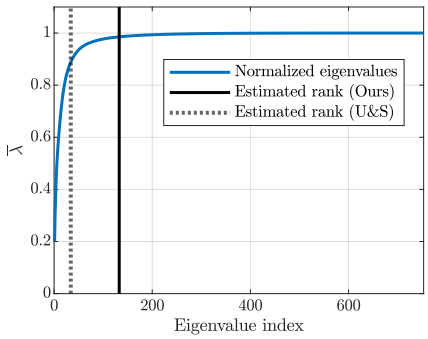

Figure 3: The normalized cumulative sum of the eigenvalues of the Hessian for DnCNN (blue). The solid black line and the dashed gray line represent the estimated rank of the proposed approach and the U&S method, respectively.

rank, we measure how much of the spectrum energy of the numerical Hessian is within the estimated rank. Specifically, we use the cumulative sum of eigenvalues and normalize it by the Hessian trace, namely, $\bar{\lambda}_j = \frac{1}{\mathrm{Tr}(\boldsymbol{H})} \sum_{i=1}^{j} \lambda_i\{\boldsymbol{H}\}$.

Figure 3 presents the spectrum of the Hessian $\{\bar{\lambda}_j\}_{j=1}^{753}$ in blue, and the proposed Hessian rank estimation as a vertical black line. The dashed black line presents the Hessian rank estimation of the U&S method. We see that our method captures more than 98.5% of the energy of the eigenvalues with a rank estimate of 133. In contrast, the U&S method results in the rank estimation of 34 capturing only 88% of the energy. We note that in this experiment, using Adam preconditioner is required for stability. In the SM, we show that using the identity matrix as a preconditioner is unstable.

Related Work

In (Zhu et al. 2019), a similar setting to ours, of a quadratic approximation of the loss function near minimum points, is considered. There, the OU process without a preconditioner is analyzed using Taylor expansion assuming a short time scale. In contrast to (Zhu et al. 2019), we consider a preconditioner in the OU process, and our analysis is for a large time scale. This allows us to examine the effect of the preconditioner as well as the Hessian on the expected loss.

The spectral properties of the Hessian of NNs were investigated in previous works. In (Ghorbani, Krishnan, and Xiao 2019), an estimation of the spectral density of the Hessian is proposed, and insights regarding the geometry of the loss surface are derived. In (Yao et al. 2020), a framework for estimating the trace of the Hessian, its top eigenvalues, and its spectral density is also proposed, and the effect of the NN architecture on the Hessian spectrum is investigated. Nonetheless, to the best of our knowledge, a method for estimating the Hessian rank has not been presented.

Methods for estimating the rank of a matrix beyond the context of NNs have also been proposed (Perry and Wolfe 2010; Kritchman and Nadler 2009; Ubaru and Saad 2016). However, they are application-specific or not well-suited for the Hessian of a NN. In (Ubaru and Saad 2016), it was proposed to estimate the Hessian rank by estimating the trace of the eigen projector, which, in turn, is estimated using a polynomial filter of degree m, $\phi_m(x)$, that serves as an approximation of a step function.

Following Theorem 1 and setting $\boldsymbol{\Sigma} = \boldsymbol{G} = \boldsymbol{I}$, the proposed approach uses the expected loss to compute the value of $\mathrm{Tr}(\boldsymbol{I} - e^{-\boldsymbol{H}t})$ for a large t. This could be viewed as the filter of $\phi(x) = 1 - e^{-xt}$. Consequently, the proposed approach could be viewed as a means of using LD to implement an exponential filter. In contrast to the filters proposed by Ubaru and Saad (2016), which are only polynomial approximations, the proposed approach implements the exact expression of the filter under the ergodicity assumption. Additionally, the U&S method requires the estimation of the density of the spectrum as well as setting multiple hyperparameters, which affect its performance. For NNs, it requires the computation of the Hessian matrix or its approximation to produce the matrix-vector products. In contrast, the proposed approach relies only on the computation of the expected loss and is more suitable for estimating the rank of the Hessian of NNs. Furthermore, the proposed approach has a theoretical guarantee that in the limit of infinite iterations, it converges to the Hessian rank.

Conclusions

In this work, we consider preconditioned Langevin dynamics near stationary points and analyze the expected loss for different preconditioners. We show that when a preconditioner admits a particular relation with respect to the noise covariance, the expected loss is proportional to the Hessian rank. Following this theoretical result, we devise an iterative algorithm for neural network Hessian rank estimation. In addition, we show that under a certain condition, the expected loss at a large time scale depends only on the interplay between the preconditioner and the noise covariance. We use our analysis to compare SGD-like and Adam-like preconditioners, deriving a condition on which preconditioner leads to a higher loss. We empirically demonstrate the theoretical results and the accurate Hessian rank estimation for linear neural networks as well as DnCNN networks.

Acknowledgments

The work of AB and RT was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 802735-ERC-DIFFO. The work of TM was partially supported by the Israel Science Foundation (grant no. 2318/22), and by the Ollendorff Minerva Center, ECE faculty, Technion. The work of RM was supported by the Planning and Budgeting Committee of the Israeli Council for Higher Education.

References

Arora, S.; Cohen, N.; and Hazan, E. 2018. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, 244–253. PMLR.

Arora, S.; Cohen, N.; Hu, W.; and Luo, Y. 2019. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32.

Borysenko, O.; and Byshkin, M. 2021. CoolMomentum: A method for stochastic optimization by Langevin dynamics with simulated annealing. *Scientific Reports*, 11(1): 10705.

Chen, X.; Du, S. S.; and Tong, X. T. 2020. On Stationary-Point Hitting Time and Ergodicity of Stochastic Gradient Langevin Dynamics. *Journal of Machine Learning Research*, 21(68): 1–41.

Choi, K. P.; Kam, E. H. H.; Tong, X. T.; and Wong, W. K. 2023. Appropriate noise addition to metaheuristic algorithms can enhance their performance. *Scientific Reports*, 13(1): 5291.

Dauphin, Y.; De Vries, H.; and Bengio, Y. 2015. Equilibrated adaptive learning rates for non-convex optimization. *Advances in neural information processing systems*, 28.

Ding, N.; Fang, Y.; Babbush, R.; Chen, C.; Skeel, R. D.; and Neven, H. 2014. Bayesian sampling using stochastic gradient thermostats. *Advances in neural information processing systems*, 27.

Elkabetz, O.; and Cohen, N. 2021. Continuous vs. discrete optimization of deep neural networks. *Advances in Neural Information Processing Systems*, 34: 4947–4960.

Gardiner, C. W.; et al. 1985. *Handbook of stochastic methods*, volume 3. springer Berlin.

Gelfand, S. B.; and Mitter, S. K. 1991. Recursive stochastic algorithms for global optimization in $\mathbb{R}^d$. *SIAM Journal on Control and Optimization*, 29(5): 999–1018.

Ghorbani, B.; Krishnan, S.; and Xiao, Y. 2019. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, 2232–2241. PMLR.

Girolami, M.; and Calderhead, B. 2011. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2): 123–214.

Huh, M.; Mobahi, H.; Zhang, R.; Cheung, B.; Agrawal, P.; and Isola, P. 2022. The Low-Rank Simplicity Bias in Deep Networks. *Transactions on Machine Learning Research*.

Kaiser, Ł.; and Sutskever, I. 2015. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kloeden, P. E.; Platen, E.; Kloeden, P. E.; and Platen, E. 1992. *Stochastic differential equations*. Springer.

Kritchman, S.; and Nadler, B. 2009. Non-parametric detection of the number of signals: Hypothesis testing and random matrix theory. *IEEE Transactions on Signal Processing*, 57(10): 3930–3941.

Latz, J. 2021. Analysis of stochastic gradient descent in continuous time. *Statistics and Computing*, 31(4): 39.

Li, C.; Chen, C.; Carlson, D.; and Carin, L. 2016. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Li, Z.; Luo, Y.; and Lyu, K. 2020. Towards Resolving the Implicit Bias of Gradient Descent for Matrix Factorization: Greedy Low-Rank Learning. In *International Conference on Learning Representations*.

Marceau-Caron, G.; and Ollivier, Y. 2017. Natural Langevin dynamics for neural networks. In *International Conference on Geometric Science of Information*, 451–459. Springer.

Mulayoff, R.; and Michaeli, T. 2020. Unique properties of flat minima in deep networks. In *International conference on machine learning*, 7108–7118. PMLR.

Neelakantan, A.; Vilnis, L.; Le, Q. V.; Sutskever, I.; Kaiser, L.; Kurach, K.; and Martens, J. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*.

Papyan, V. 2020. Traces of class/cross-class structure pervade deep learning spectra. *The Journal of Machine Learning Research*, 21(1): 10197–10260.

Patterson, S.; and Teh, Y. W. 2013. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in neural information processing systems*, 3102–3110.

Perry, P. O.; and Wolfe, P. J. 2010. Minimax rank estimation for subspace tracking. *IEEE Journal of Selected Topics in Signal Processing*, 4(3): 504–513.

Pock, T.; and Chambolle, A. 2011. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*, 1762–1769. IEEE.

Raginsky, M.; Rakhlin, A.; and Telgarsky, M. 2017. Nonconvex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. *arXiv preprint arXiv:1702.03849*.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Staib, M.; Reddi, S.; Kale, S.; Kumar, S.; and Sra, S. 2019. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, 5956–5965. PMLR.

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

Ubaru, S.; and Saad, Y. 2016. Fast methods for estimating the numerical rank of large matrices. In *International Conference on Machine Learning*, 468–477. PMLR.

Wang, Y.-X.; Fienberg, S.; and Smola, A. 2015. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In *International Conference on Machine Learning*, 2493–2502. PMLR.

Welling, M.; and Teh, Y. W. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 681–688.

Xu, P.; Chen, J.; Zou, D.; and Gu, Q. 2018. Global convergence of Langevin dynamics based algorithms for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 3122–3133.

Yao, Z.; Gholami, A.; Keutzer, K.; and Mahoney, M. W. 2020. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, 581–590. IEEE.

Zeyer, A.; Doetsch, P.; Voigtlaender, P.; Schlüter, R.; and Ney, H. 2017. A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2462–2466. IEEE.

Zhang, G.; Li, L.; Nado, Z.; Martens, J.; Sachdeva, S.; Dahl, G.; Shallue, C.; and Grosse, R. B. 2019. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32.

Zhang, K.; Zuo, W.; Chen, Y.; Meng, D.; and Zhang, L. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7): 3142–3155.

Zhu, Z.; Wu, J.; Yu, B.; Wu, L.; and Ma, J. 2019. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects. In *International Conference on Machine Learning*, 7654–7663. PMLR.