Gradient Regularized Natural Gradients

Satya Prakash Dash[1,*] **Hossein Abdi**[1,*] **Wei Pan**[1] **Samuel Kaski**[1,2] **Mingfei Sun**[1,†]
[1] Department of Computer Science, The University of Manchester, United Kingdom
[2] Department of Computer Science, Aalto University, Finland
{satyaprakash.dash, hossein.abdi, wei.pan, samuel.kaski, mingfei.sun}@manchester.ac.uk

Abstract

Gradient regularization (GR) has been shown to improve the generalizability of trained models. While Natural Gradient Descent has been shown to accelerate optimization in the initial phase of training, little attention has been paid to how the training dynamics of second-order optimizers can benefit from GR. In this work, we propose Gradient-Regularized Natural Gradients (GRNG), a family of scalable second-order optimizers that integrate explicit gradient regularization with natural gradient updates. Our framework introduces two frequentist algorithms: Regularized Explicit Natural Gradient (RENG), which utilizes double backpropagation to explicitly minimize the gradient norm, and Regularized Implicit Natural Gradient (RING), which incorporates regularization implicitly into the update direction. We also propose a Bayesian variant based on a Regularized-Kalman formulation that eliminates the need for FIM inversion entirely. We establish convergence guarantees for GRNG, showing that gradient regularization improves stability and enables convergence to global minima. Empirically, we demonstrate that GRNG consistently enhances both optimization speed and generalization compared to first-order methods (SGD, AdamW) and second-order baselines (K-FAC, Sophia), with strong results on vision and language benchmarks.

[*] Equal contribution.
† Corresponding author.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

The rapid advancement of deep neural networks (DNNs) in recent years has driven significant progress across a wide range of domains, including computer vision and natural language processing. Despite these successes, the design of optimization algorithms that simultaneously ensure fast convergence and improved generalization in the training of these large-scale models remains a fundamental challenge [Bottou et al., 2018].

Recently, Natural Gradient Descent (NGD) methods have attracted increasing attention in the context of training DNNs. This interest is largely motivated by their favorable theoretical properties—most notably, their ability to identify optimal solutions in as few as $\mathcal{O}(1)$ iterations under certain conditions Zhang et al. [2019]. In addition, NGD methods are well-suited for addressing ill-conditioned optimization landscapes, where they can provide significant advantages over traditional approaches by accelerating convergence and improving training stability Becker et al. [1988], Martens [2020].

Gradient Regularization (GR) has been extensively investigated in the context of gradient descent Smith et al. [2021a], Barrett and Dherin [2020]. GR-based methods have been shown to accelerate convergence in the initial phase Jastrzebski et al. [2021] and have additionally been associated with improved generalization of the trained models Smith et al. [2021b]. Theoretically, it has been shown that employing higher learning rates induces an implicit GR effect. Complementarily, explicit regularization of the objective function via the gradient norm has been empirically demonstrated to bias the optimization trajectory toward finding flatter regions of the loss landscape Drucker and Le Cun [1992] Barrett and Dherin [2020]. This, in turn, has been linked to enhanced generalization performance and improved robustness of trained models.

Despite the extensive body of work on gradient regularization in the context of gradient descent, relatively

little attention has been devoted to (i) understanding its convergence properties when combined with natural gradient optimization methods, and (ii) developing empirical techniques to assess its impact on the generalization performance of NGD optimizers. A more extensive review of related work is provided in Appendix A.

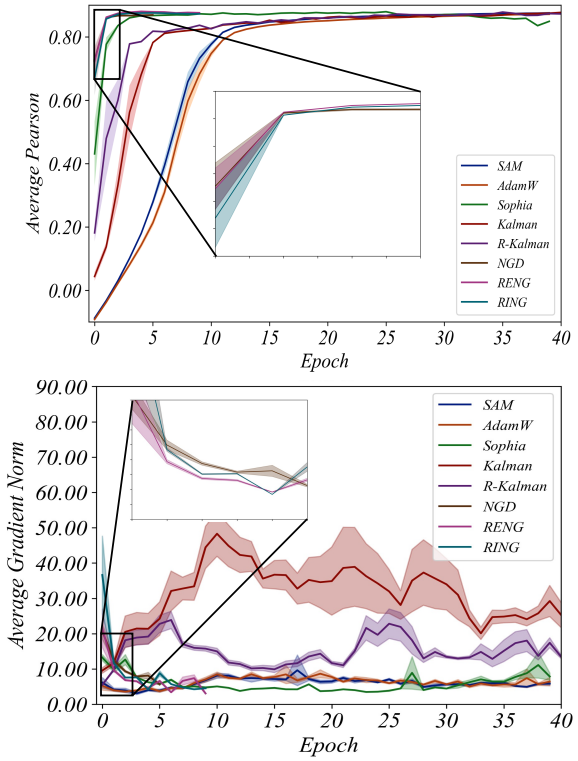

Figure 1: Average Gradient Norm (left) and Average Pearson Correlation (right) for SAM, AdamW, Sophia, Kalman, R-Kalman, RING, and RENG on the STS-B dataset.

In this work, we investigate the convergence properties of Gradient-Regularized Natural Gradients (GRNG). We provide a theoretical analysis showing that, for a simple two-layer neural network, GRNG converges to the global minimum. Beyond theory, we empirically evaluate the generalization performance of GRNG in comparison with a range of established optimizers, including first-order methods such as Stochastic Gradient Descent (SGD) and Sharpness-Aware Minimization (SAM) Foret et al. [2020], quasi-second-order methods such as Adam Kingma and Ba [2014], and second-order baselines without gradient regularization, such as K-FAC with Tikhonov regularization Martens and Grosse [2015], and Sophia [Liu et al., 2023]. Our framework further introduces two frequentist algorithms (see Algorithm 1): RENG and RING. RENG employs double backpropagation to explicitly regularize the objective, whereas RING avoids this overhead by estimating an update direction that reflects the regularization implicitly. These are complemented by a Bayesian variant based on a Regularized-Kalman formulation that eliminates the need for FIM inversion entirely. Together, these advances make GRNG scalable and applicable to large-scale architectures, including Transformers in both language and vision domains.

In 1, we illustrate the advantages of incorporating gradient regularization. SAM, a gradient-regularized first-order optimizer, is able to locate flatter minima more efficiently than a quasi-second-order method such as AdamW, highlighting the effectiveness of gradient regularization. Furthermore, our proposed GRNG methods (Algorithms 1 and 2) achieve a rapid reduction in gradient norm compared to their unregularized counterparts, and converges within a few epochs of training.

2 PROBLEM STATEMENT

Formally, suppose we are given a dataset $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_k) \sim p^*(\mathbf{x}, \mathbf{y})\}_{k=1}^N$, where $\mathbf{x}_k \in \mathbb{R}^{d_i}$ and $\mathbf{y}_k \in \mathbb{R}^{d_o}$ denote the input and output vectors, respectively; and are sampled i.i.d. from the true data distribution $p^*(\mathbf{x}, \mathbf{y})$. Consider a (possibly pre-trained) model defined by $\hat{\mathbf{y}} = h(\mathbf{x}, \boldsymbol{\theta})$, which is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$. Given an input $\mathbf{x}$, the model defines a predictive distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, from which the output $\mathbf{y}$ can be sampled, and $\hat{\mathbf{y}}$ is a representative statistic of this distribution. The objective of prediction in machine learning is to determine model parameters that make the model's predicted distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ closely approximate the true conditional data distribution $p^*(\mathbf{y}|\mathbf{x})$. Specifically, prediction aims to learn a mapping from inputs $\mathbf{x}$ to outputs $\mathbf{y}$, in order to predict the outputs for new and unseen inputs. This mapping is learned by training or fine-tuning the model $\hat{\mathbf{y}} = h(\mathbf{x}, \boldsymbol{\theta})$. In the following, we introduce that both frequentist and Bayesian optimization frameworks can be employed in this context.

Frequentist Approach. In Frequentist approaches, this objective is typically achieved by minimizing the Kullback–Leibler (KL) divergence between the true conditional distribution and the model distribution, $D_{KL}(p^*(\mathbf{y}|\mathbf{x})||p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}))$, which is equivalent to minimizing the negative log-likelihood loss: $\mathcal{L}(\boldsymbol{\theta}) = -\mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\ln p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})]$ [Goodfellow et al., 2016, Murphy, 2012, Bishop and Nasrabadi, 2006]. When the update step ($\boldsymbol{\delta}$) is infinitesimally small, its second-order approximation is given as

$$\mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\delta}) = \mathcal{L}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})^\top \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{F}(\boldsymbol{\theta})\boldsymbol{\delta} + \mathcal{O}(|\boldsymbol{\delta}|^3), \quad (1)$$

where $\boldsymbol{\delta}$ is the parameter update and the curvature of the loss function is given by the Fisher information matrix $\mathbf{F}(\boldsymbol{\theta})$ [Pascanu and Bengio, 2013], defined as follows:

$$\mathbf{F}(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \ln p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})^\top]. \tag{2}$$

Solving the quadratic function of Equation 1 in the vicinity of $\boldsymbol{\theta}$ gives the natural gradient direction: $\boldsymbol{\delta}^* = -\mathbf{F}(\boldsymbol{\theta})^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$, where the gradient of the loss function is with respect to the *data distribution* (also called batch-gradients). Note that Fisher information matrix takes the expectation over the *model distributions*.

Bayesian Approach. From a Bayesian perspective, the parameter $\boldsymbol{\theta}$ is treated as a random variable. Given a data point $(\mathbf{x}_k, \mathbf{y}_k)$ in a stream of data, we aim to recursively estimate the posterior distribution of the parameters using Bayesian inference [Murphy, 2023, 2012]:

$$p(\boldsymbol{\theta}_k \mid \mathcal{D}_{1:k}) \propto p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}_{k-1}) p(\boldsymbol{\theta}_{k-1} \mid \mathcal{D}_{1:k-1}), \tag{3}$$

where $\mathcal{D}_{1:k} \subseteq \mathcal{D}$ indicates the subset of dataset used for training until k^{th} data. And $p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}_{k-1})$ represents the likelihood function, $p(\boldsymbol{\theta}_{k-1} \mid \mathcal{D}_{1:k-1})$ and $p(\boldsymbol{\theta}_k \mid \mathcal{D}_{1:k})$ denote the prior and posterior, respectively. By leveraging a Kalman-based approach, we recursively update and compute the posterior distribution over the model parameters.

3 METHODS

Direct inversion of the full Fisher Information Matrix is computationally intractable in large-scale settings. To address this challenge, we adopt two strategies: in the frequentist approach, we employ block-diagonal Kronecker-factored approximations of the FIM, combined with Newton's Iteration and the lazy-Fisher technique; in the Bayesian approach, we bypass the need for FIM inversion entirely by leveraging a Kalman-based technique.

3.1 Frequentist: Inverting Fisher Layerwise and Lazily

Kronecker-Factored Approximation. We first take the frequentist approach to address the computational challenge. We develop our algorithm around the layer-wise structure of the model's weight matrices. For simplicity, we denote the weight matrix of layer i as $\mathbf{W}_i \in \mathbb{R}^{\omega_i \times \omega_i'}$, where ω_i and ω_i' represent the input and output dimensions of the layer, respectively. The full parameter vector $\boldsymbol{\theta}$ is then defined as the concatenation of the vectorized layer weights: $\boldsymbol{\theta} = \text{vec}(\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L)$, where L denotes the number of layers in the model. We also adopt $\boldsymbol{x}_i$ as input to layer i, and $\boldsymbol{e}_i$ as backpropagated error until that layer. The loss function can be expressed in terms of $\boldsymbol{x}_{i-1}$ and $\boldsymbol{e}_i$, which are the canonical basis of the loss:

$$\mathcal{L}(\boldsymbol{\theta}) \equiv \mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L) \equiv \mathcal{L}(\boldsymbol{x}_0, ..., \boldsymbol{x}_{L-1}, \boldsymbol{e}_1, ..., \boldsymbol{e}_L) \tag{4}$$

The gradient for each weight matrix can be rewritten on a suitable canonical basis as:

$$\nabla_{\mathbf{W}_i}\mathcal{L}(\mathbf{W}_i) = \mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{e}_i \boldsymbol{x}_{i-1}^\top] = \mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{x}_{i-1} \otimes \boldsymbol{e}_i], \tag{5}$$

where $\otimes$ denotes the Kronecker product. Also, inspired by [Bernacchia et al., 2018], we approximate the full Fisher Information Matrix as follows:

$$\begin{aligned} \mathbf{F}(\boldsymbol{\theta}) &= \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{x}\boldsymbol{x}^\top \otimes \boldsymbol{e}\boldsymbol{e}^\top] \\ &\approx \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{x}\boldsymbol{x}^\top] \otimes \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{e}\boldsymbol{e}^\top] \end{aligned} \tag{6}$$

Assuming each weight matrix is independent of the others, for each model layer we get:

$$\mathbf{F}_{ii}(\mathbf{W}_i) = \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{x}_{i-1}\boldsymbol{x}_{i-1}^\top] \otimes \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{e}_i \boldsymbol{e}_i^\top] \tag{7}$$

where we refer to $\boldsymbol{\Lambda}_{i-1} = \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{x}_{i-1}\boldsymbol{x}_{i-1}^\top]$ as the activation matrix, and $\boldsymbol{\Gamma}_i = \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{e}_i \boldsymbol{e}_i^\top]$ as the error matrix. Then, we can write the weight update for layer $\mathbf{W}_i$ as:

$$\begin{aligned} \Delta\mathbf{W}_i &\propto \mathbf{F}_{ii}(\mathbf{W}_i)^{-1}\nabla_{\mathbf{W}_i}\mathcal{L}(\mathbf{W}_i) \\ &\propto \boldsymbol{\Lambda}_{i-1}^{-1} \cdot \mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{e}_i \boldsymbol{x}_{i-1}^\top] \cdot \boldsymbol{\Gamma}_i^{-1} \end{aligned} \tag{8}$$

To compute this equation, we use the mixed Kronecker matrix-vector product property: $(\mathbf{A} \otimes \mathbf{B})\text{vec}(\mathbf{C}) = \text{vec}(\mathbf{B}\mathbf{C}\mathbf{A}^\top)$.

Lazy Fisher. Computing the Fisher information matrix at every training step is computationally expensive. We take advantage of the idea of using the same Fisher information matrix for the next few training steps, which is also known as *Lazy Hessian* in Hessian estimation[Shamanskii, 1967]. It has been shown to achieve global convergence in damped Newton methods[Wang et al., 2006] and proximal Newton-type algorithms [Adler et al., 2020] in the context of convex optimization, and global convergence guarantees for its damped and regularized variants[Doikov et al., 2023, Chayti et al., 2023]. We empirically investigate the effectiveness of the lazy Fisher approach in our experiments. This technique significantly reduces training time – often by orders of magnitude – compared to standard NGD, as it avoids inverting the Fisher matrix at every iteration.

3.2 Frequentist: Implicit and Explicit Gradient Regularized Natural Gradients

Although the weight update derived in Equation 8 provides a quadratic approximation of the loss function, its applicability remains challenging because the

error matrix computed at a given iteration can be rank-deficient. Furthermore, as shown in Equation 1, the loss function is approximated using a second-order Taylor expansion around the current solution. Hence, the approximation can be accurate in the neighborhood of the solution, particularly for nearly linear loss functions, which is a condition that typically does not hold in the early stages of training. Moreover, due to the non-convex nature of the optimization landscape, the weight update direction in Equation 8 might not yield an optimal descent direction. In such cases, an appropriate regularization is necessary, as the steepest descent may lead to suboptimal performance. From an empirical perspective, we also consider how to obtain the weight update for pointwise operations. Therefore, regularization methods can be a helpful tool to accelerate the training process.

Gradient Regularization. Gradient regularization adds the norm of gradient of the loss to encourage the search for flatter minima [Barrett and Dherin, 2020]. Here, in this case, we will add a regularizer averaged over the model distribution, meaning that we regularize our objective to create a direction where the gradient averaged over the model distribution is minimized. Gradient regularization to the quadratic loss approximation can be defined as:

$$\mathcal{L}_G(\boldsymbol{\theta}+\boldsymbol{\delta}) = \mathcal{L}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top\Big[\mathbf{F}(\boldsymbol{\theta}) + \rho\left\|\mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})]\right\|_2^2\mathbf{I}\Big]\boldsymbol{\delta} + \mathcal{O}(|\boldsymbol{\delta}|^3), \tag{9}$$

where ρ denotes the damping coefficient. We consider two approaches to applying gradient regularization: *explicit* (in the **RENG** algorithm) and *implicit* (in the **RING** algorithm). The explicit method involves computing the *L2-norm* of the primary loss gradient and then backpropagating through this norm—an approach known as *double backpropagation* Drucker and Le Cun [1992]. In contrast, the implicit method avoids this extra step by directly estimating an update direction that reflects the regularization effect.

The double backpropagation technique has demonstrated its effectiveness in improving generalization by [Drucker and Le Cun, 1991]. [Hochreiter and Schmidhuber, 1997] has also shown that it encourages convergence toward flatter minima in the loss landscape. [Barrett and Dherin, 2020] has theoretically studied this technique under vanilla SGD and proves that gradient regularization encourages taking a shallower path to reach local minima. For the weight update step in the RING and RENG algorithms, we adopt the following equations from [Bernacchia et al., 2018].

$$\Delta\mathbf{W}_i = \frac{\alpha}{L}\left(\boldsymbol{\Lambda}_{i-1} + \sqrt{\rho}\left\|\boldsymbol{\Lambda}_{i-1}\right\|_2\mathbf{I}\right)^{-1} \cdot \mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{e}_i\boldsymbol{x}_{i-1}^\top] \cdot \left(\boldsymbol{\Gamma}_i + \sqrt{\rho}\left\|\boldsymbol{\Gamma}_i\right\|_2\mathbf{I}\right)^{-1} \tag{10a}$$

$$\Delta\mathbf{W}_i = \frac{\alpha}{L}\left(\boldsymbol{\Lambda}_{i-1} + \sqrt{\rho}\mathbf{I}\right)^{-1} \cdot \mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{e}_i\boldsymbol{x}_{i-1}^\top] \cdot \left(\boldsymbol{\Gamma}_i + \sqrt{\rho}\mathbf{I}\right)^{-1}. \tag{10b}$$

Here, we define $\tilde{\boldsymbol{\Lambda}} := \boldsymbol{\Lambda} + \lambda\mathbf{I}$, and $\tilde{\boldsymbol{\Gamma}} := \boldsymbol{\Gamma} + \lambda\mathbf{I}$, where $\lambda = \sqrt{\rho}\left\|\cdot\right\|_2$ is for RING, and $\lambda = \sqrt{\rho}$ is for RENG. These regularizations can be viewed as modifications of the geometry induced by the Fisher information matrix. The pseudocode of these algorithms is provided in Appendix B, and their computational complexity is analyzed in Appendix subsection E.1.

Updating Inverse Matrix. Despite the computational advantages that Lazy Fisher offers, we observe that it can occasionally lead to training instabilities. The primary reason is the use of the regularization mechanism. Specifically, when the damping coefficient (ρ) is updated at each iteration, it must remain greater than the minimum eigenvalue of the Fisher matrix at that iteration. Failure to satisfy this condition may result in inaccurate updates along the eigenvectors with the smallest eigenvalues. To mitigate this issue, we approximate the updated matrix inversion using matrix differential techniques. To this aim, the regularized activation ($\tilde{\boldsymbol{\Lambda}}$) and error ($\tilde{\boldsymbol{\Gamma}}$) matrices are updated as follows:

$$\tilde{\boldsymbol{\Lambda}}_{i_{k+1}}^{-1} = \tilde{\boldsymbol{\Lambda}}_{i_k}^{-1} - d\lambda_k\tilde{\boldsymbol{\Lambda}}_{i_k}^{-1}\tilde{\boldsymbol{\Lambda}}_{i_k}^{-1} + \mathcal{O}(d\lambda_k^2) \tag{11a}$$

$$\tilde{\boldsymbol{\Gamma}}_{i_{k+1}}^{-1} = \tilde{\boldsymbol{\Gamma}}_{i_k}^{-1} - d\lambda_k\tilde{\boldsymbol{\Gamma}}_{i_k}^{-1}\tilde{\boldsymbol{\Gamma}}_{i_k}^{-1} + \mathcal{O}(d\lambda_k^2) \tag{11b}$$

See Appendix subsection D.1 for the detailed derivation.

Newton's Iteration. Equation 11 requires the initial inverses $\tilde{\boldsymbol{\Lambda}}_i^{-1}$ and $\tilde{\boldsymbol{\Gamma}}_i^{-1}$ to be precomputed, which can itself be computationally expensive. To address this, we employ *Newton's Iteration* method [Schulz, 1933] using a cubic-order Neumann series approximation. Newton's iteration start from a scaled version of the transpose of the matrix ($\mathbf{X}_0 = \alpha\mathbf{A}^\top$) and iteratively reduce the residue using a Neumann series approximation: $\mathbf{X}_{k+1} = \mathbf{X}_k(3\mathbf{I} - 3\mathbf{A}\mathbf{X}_k + (\mathbf{A}\mathbf{X}_k)^2)$. This series will converge to $\mathbf{A}^{-1}$ provided proper choice of α like $\alpha = \frac{1}{Tr(\mathbf{A}^\top\mathbf{A})}$ or $\alpha = \frac{1}{\|\mathbf{A}\|_2\|\mathbf{A}\|_\infty}$ [Ben-Israel, 1965, Ben-Israel and Cohen, 1966].

3.3 Bayesian: Gradient Regularized Kalman

Leveraging a Kalman-based framework, we recursively update the posterior distribution over model parameters along the natural gradient direction, which avoids

the explicit inversion of the Fisher Information Matrix within a Bayesian setting.

Kalman Algorithm. Following the standard Kalman filtering assumptions, we adopt a Gaussian variational approximation for the posterior distribution over model parameters: $p(\boldsymbol{\theta}_k \mid \mathcal{D}_{1:k}) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. In this regard, we assume that an initial prior distribution of the trainable parameters is given by a Gaussian distribution $p(\boldsymbol{\theta}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. Equation 3 can be recursively computed for each data point in $\mathcal{D}$ using Kalman filtering, which corresponds to following the natural gradient direction (see Appendix subsection D.4 for details). Consider a Gaussian likelihood function as $p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}_{k-1}) = \mathcal{N}(\mathbf{y}_k \mid \hat{\mathbf{y}}_k, \mathbf{R}_k)$; or more broadly, an exponential family non-Gaussian likelihood function expressed as $p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}_{k-1}) = f_{\text{exp}}(T(\mathbf{y}_k)|\hat{\mathbf{y}}_k)$, where f_{exp} denotes the exponential family function, and $T(\cdot)$ represents the sufficient statistics associated with the exponential family. Here, $\mathbf{y}_k$ denotes the true output, while $\hat{\mathbf{y}}_k$ corresponds to the model's predicted output. In this context, the observation noise covariance matrix $\mathbf{R}_k$ is defined as $\mathbf{R}_k = \text{Cov}(\mathbf{y}_k|\hat{\mathbf{y}}_k)$ (for non-Gaussian: $\mathbf{R}_k = \text{Cov}(T(\mathbf{y}_k)|\hat{\mathbf{y}}_k)$), which captures the covariance between $\mathbf{y}_k$ and $\hat{\mathbf{y}}_k$ (for non-Gaussian: $T(\mathbf{y}_k)$ and $\hat{\mathbf{y}}_k$) [Ollivier, 2018].

Based on the information form of the Kalman algorithm (information filtering), two fundamental steps should be taken to estimate the posterior distribution of the trainable parameters [Simon, 2006]:

Prediction Step: Under the first step of the Kalman algorithm, the prior is predicted based on the posterior from the previous iteration, following the Gaussian transition model:

$$p(\boldsymbol{\theta}_{k|k-1} \mid \boldsymbol{\theta}_{k-1}) = \mathcal{N}(\boldsymbol{\theta}_{k|k-1} \mid \boldsymbol{\theta}_{k-1}, \mathbf{Q}_k), \tag{12}$$

where $\mathbf{Q}_k$ denotes the process noise covariance matrix, and the subscript $k|k-1$ indicates the predicted value at time step k conditioned on information available up to time $k-1$. Under this evolution, the predicted prior distribution is:

$$\boldsymbol{\mu}_{k|k-1} = \boldsymbol{\mu}_{k-1} \tag{13a}$$
$$\boldsymbol{\Sigma}^{-1}_{k|k-1} = \boldsymbol{\Sigma}^{-1}_{k-1} + \mathbf{Q}_k \tag{13b}$$

Updating Step: In the second step of the Kalman filter, the predicted prior is updated to estimate the posterior as follows:

$$\mathbf{K}_k = \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top \left(\mathbf{H}_k \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \tag{14a}$$
$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} + \mathbf{K}_k \left(\mathbf{y}_k - h(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})\right) \tag{14b}$$
$$\boldsymbol{\Sigma}^{-1}_k = \boldsymbol{\Sigma}^{-1}_{k|k-1} - \mathbf{H}_k^\top \mathbf{R}_k^{-1}\mathbf{H}_k \tag{14c}$$

where $\mathbf{K}_k$ is the Kalman gain, and $\mathbf{H}_k$ indicates the Jacobian matrix of the model $h(\mathbf{x}, \boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ at the point of $(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})$. Note that $h(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})$ represents the predicted values for the regression tasks and the predicted probabilities for the classification tasks.

Regularized Kalman. In line with the discussions presented in subsection 3.2 regarding the frequentist perspective, the gradient (or equivalently, Jacobian) regularization can be interpreted as a modification of the Fisher information matrix of the form $\tilde{\mathbf{F}}_k = \mathbf{F}_k + \rho\mathbf{H}_k^\top\mathbf{H}_k$. This interpretation reveals that the gradient-regularized natural gradient step can be expressed as a modified Kalman update, thereby establishing a direct connection between regularized Kalman filtering and RING/RENG algorithms.

Recall that the Fisher information matrix associated with a negative log-likelihood loss $\mathcal{L}$ is given by (see Equation 2): $\mathbf{F}(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}}\mathcal{L} \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}^\top]$. At time step k, the contribution of a single observation can be expressed as $\mathbf{F}(\boldsymbol{\theta}_k) = \nabla_{\boldsymbol{\theta}}\mathcal{L}_k \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}_k^\top$. Applying the chain rule, $\nabla_{\boldsymbol{\theta}}\mathcal{L}_k = \nabla_{\hat{\mathbf{y}}}\mathcal{L}_k \cdot \nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k$, we obtain $\mathbf{F}(\boldsymbol{\theta}_k) = (\nabla_{\hat{\mathbf{y}}}\mathcal{L}_k \cdot \nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k) \cdot (\nabla_{\hat{\mathbf{y}}}\mathcal{L}_k \cdot \nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k)^\top$. By matrix multiplication rules, this reduces to $\mathbf{F}(\boldsymbol{\theta}_k) = \nabla_{\boldsymbol{\theta}}^\top\hat{\mathbf{y}}_k \cdot \nabla^2_{\hat{\mathbf{y}}}\mathcal{L}_k \cdot \nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k$. Identifying $\mathbf{H}_k = \nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k$ and noting that for Gaussian (or more generally exponential-family) likelihoods, $\mathbf{R}_k^{-1} = \nabla^2_{\hat{\mathbf{y}}}\mathcal{L}_k$, we conclude that $\mathbf{F}(\boldsymbol{\theta}_k) = \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k$. Substituting into the regularized Fisher information, we obtain $\tilde{\mathbf{F}}_k = \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k + \rho\mathbf{H}_k^\top\mathbf{H}_k$, or equivalently $\tilde{\mathbf{F}}_k = \mathbf{H}_k^\top(\mathbf{R}_k^{-1} + \rho\mathbf{I})\mathbf{H}_k$. This motivates the definition of a modified observation noise covariance, $\tilde{\mathbf{R}}_k = (\mathbf{R}_k^{-1} + \rho\mathbf{I})^{-1}$, which can be interpreted as a regularized version of $\mathbf{R}_k$. To avoid repeated inversion, we factor out $\mathbf{R}_k^{-1}$, yielding $\tilde{\mathbf{R}}_k = \mathbf{R}_k(\mathbf{I}+\rho\mathbf{R}_k)^{-1}$. Substituting $\tilde{\mathbf{R}}_k$ into the Kalman gain expression (see Equation 14a), the regularized Kalman (R-Kalman) gain becomes:

$$\tilde{\mathbf{K}}_k = \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top \left(\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top + \mathbf{R}_k(\mathbf{I} + \rho\mathbf{R}_k)^{-1}\right)^{-1}. \tag{15}$$

Here, $\tilde{\mathbf{K}}_k$ represents the regularized Kalman gain. For small $\rho \ll 1$, computational efficiency can be improved by approximating the inverse via a first-order Neumann series: $(\mathbf{I} + \rho\mathbf{R}_k)^{-1} = \mathbf{I} - \rho\mathbf{R}_k + \mathcal{O}(\rho^2)$. The pseudocode of the proposed R-Kalman algorithm is provided in Appendix B, and its computational complexity is analyzed in Appendix subsection E.2.

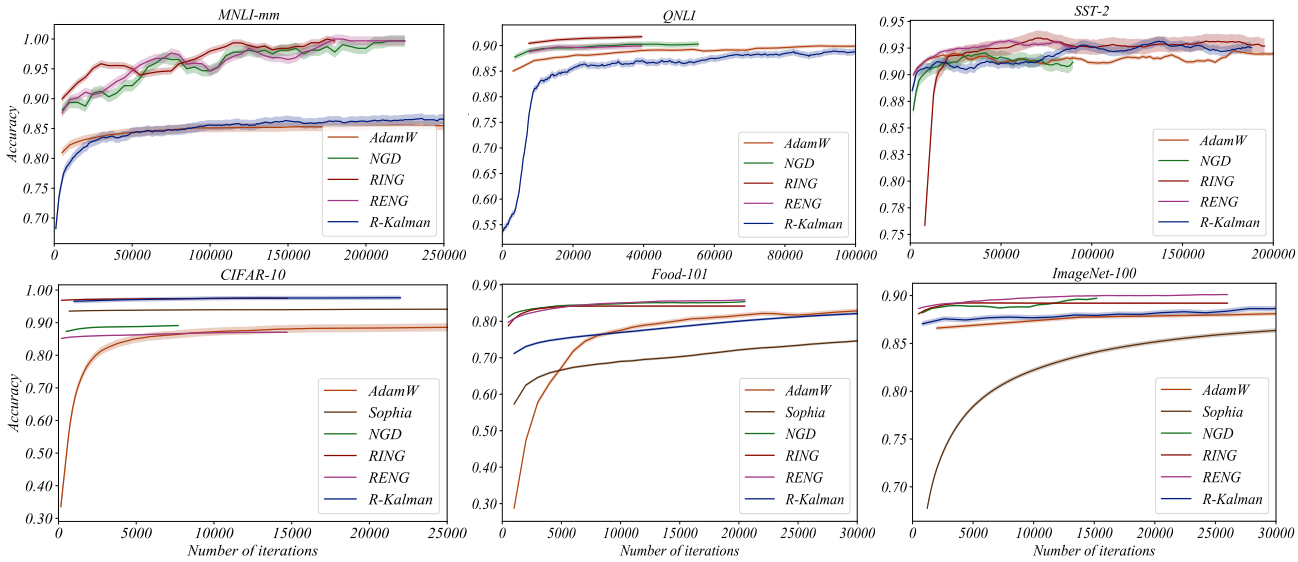

Figure 2: Validation accuracy of our proposed algorithms (RING, RENG, and R-Kalman) compared to AdamW, Sophia, and NGD on selected language (top row) and vision (bottom row) datasets. The plots illustrate validation accuracy as a function of training iterations. For a comprehensive set of results, please refer to Appendix G.

4 CONVERGENCE OF GRADIENT REGULARIZED NATURAL GRADIENTS

In this section, we provide a convergence analysis of GRNG for full-batch training of a two-layer neural network in the over-parameterized regime. We start with the following standard assumptions [Zhang et al., 2019]:

Assumption 4.1. ***Full row rank of the Jacobian matrix****: The Jacobian matrix* $\mathbf{H}_0$ *at the initialization has full row rank, or equivalently, the Gram matrix* $\mathbf{G}_0 = \mathbf{H}_0\mathbf{H}_0^\top$ *is positive definite.*

Assumption 4.2. ***Stable Jacobian****: There exists* $0 \leq C < \frac{1}{2}$ *such that for all parameters* $\boldsymbol{\theta}$ *that satisfy* $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2 \leq \frac{3\|\mathbf{y}-\hat{\mathbf{y}}_0\|_2}{\sqrt{\lambda_{\min}(\mathbf{G}_0)}}$*, we have*

$$\|\mathbf{H}(\boldsymbol{\theta}) - \mathbf{H}_0\|_2 \leq \frac{C}{3}\sqrt{\lambda_{\min}(\mathbf{G}_0)}. \tag{16}$$

Theorem 4.3. ***Gradient Regularized Natural Gradients****: Let Assumption 4.1 and Assumption 4.2 hold. Suppose we optimize a full-batch training with the ground truth of* $\mathbf{y}$ *using GRNG with a learning rate* η*. Then for the training iterations* $k = 0, 1, 2, \ldots$*, the squared error is bounded as:*

$$\|\mathbf{y} - \hat{\mathbf{y}}_{k+1}\|_2^2 \leq (1-\eta)\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2.$$

Provided that the learning rate satisfies $\eta \leq \frac{2(1+\alpha_k)(1+C)-1}{(1+\alpha_k)^2(1+C)^2}$*, where* $\alpha_k = 1 + \rho\kappa(\mathbf{G})\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2$*; with* $\kappa(\mathbf{G})$ *as condition number, i.e.,* $\lambda_{max}(\mathbf{G})/\lambda_{min}(\mathbf{G})$*, and* $\lambda(\cdot)$ *as the eigenvalue operator.*

The detailed proof is provided in Appendix C. From this per-step reduction in output space, we can get the following corollary, which shows that the step size only depends on a constant C and the maximum output error at any step k.

Corollary 4.4. *From the above theorem, we can state that if the learning rate satisfies* $\eta \leq \hat{M}$*, where* $\hat{M} = \min_i \left(\frac{2(1+\alpha_i)(1+C)-1}{(1+\alpha_i)^2(1+C)^2}\right)$*, then the* k^{th} *step error in the output space exhibits geometric decay from initialization as:*

$$\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2 \leq (1-\eta)^k\|\mathbf{y} - \hat{\mathbf{y}}_0\|_2^2.$$

5 EXPERIMENTAL EVALUATIONS

We evaluate the performance of our proposed algorithms through extensive experiments on well-established computer vision and language datasets, using different models in transfer learning.

Baseline Optimizers. We consider AdamW [Loshchilov and Hutter, 2017], Sophia [Liu et al., 2023], and NGD [Amari, 1998] as baseline algorithms. AdamW is a widely adopted first-order optimization method, serving as a strong practical baseline. NGD represents a classical second-order optimization approach, which we implement using the K-FAC method with Tikhonov regularization. Sophia, in contrast, is a recently proposed second-order optimizer that leverages a diagonal approximation of the Fisher information matrix. For parameter-efficient fine-tuning,

Table 1: Test accuracy (mean ± standard deviation) for fine-tuning ViT-B16 on CIFAR-10/100, Oxford-IIIT Pet, Food-101, and ImageNet-100 using various optimization algorithms. Higher values indicate better performance.

Algorithm	CIFAR-10	CIFAR-100	Oxford-IIIT Pet	Food-101	ImageNet-100
AdamW	$88.9_{\pm 0.10}$	$87.8_{\pm 0.02}$	$91.5_{\pm 0.08}$	$85.2_{\pm 0.02}$	$89.0_{\pm 0.04}$
Sophia	$97.1_{\pm 0.05}$	$86.8_{\pm 0.04}$	$88.8_{\pm 0.01}$	$80.0_{\pm 0.01}$	$88.2_{\pm 0.03}$
NGD	$89.3_{\pm 0.05}$	$\mathbf{89.3_{\pm 0.02}}$	$\mathbf{93.1_{\pm 0.02}}$	$85.4_{\pm 0.01}$	$89.6_{\pm 0.02}$
RING	$91.5_{\pm 0.02}$	$88.2_{\pm 0.02}$	$92.6_{\pm 0.01}$	$85.0_{\pm 0.01}$	$89.0_{\pm 0.01}$
RENG	$89.2_{\pm 0.10}$	$88.2_{\pm 0.05}$	$91.2_{\pm 0.01}$	$85.8_{\pm 0.01}$	$\mathbf{90.10_{\pm 0.01}}$
R-Kalman	$\mathbf{97.1_{\pm 0.04}}$	$88.06_{\pm 0.02}$	$92.8_{\pm 0.02}$	$\mathbf{86.2_{\pm 0.01}}$	$89.8_{\pm 0.04}$

Table 2: Test accuracy or Pearson correlation (mean ± standard deviation) for fine-tuning RoBERTa-base on the GLUE benchmark, including MNLI-mm , QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE using various optimization algorithms. Higher values indicate better performance.

Algorithm	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE
AdamW	$85.5_{\pm 0.2}$	$87.2_{\pm 0.3}$	$90.0_{\pm 0.2}$	$90.4_{\pm 0.3}$	$62.0_{\pm 0.3}$	$88.8_{\pm 0.3}$	$87.8_{\pm 0.1}$	$71.8_{\pm 0.2}$
NGD	$97.2_{\pm 0.5}$	$88.6_{\pm 0.3}$	$90.6_{\pm 0.3}$	$91.2_{\pm 0.5}$	$56.2_{\pm 0.2}$	$81.1_{\pm 0.2}$	$86.5_{\pm 0.3}$	$71.0_{\pm 0.1}$
RING	$\mathbf{98.6_{\pm 0.1}}$	$88.8_{\pm 0.1}$	$\mathbf{91.5_{\pm 0.1}}$	$92.4_{\pm 0.1}$	$58.5_{\pm 0.0}$	$88.2_{\pm 0.3}$	$\mathbf{88.2_{\pm 0.2}}$	$\mathbf{76.5_{\pm 0.2}}$
RENG	$97.6_{\pm 0.2}$	$\mathbf{90.2_{\pm 0.1}}$	$90.5_{\pm 0.2}$	$\mathbf{92.6_{\pm 0.1}}$	$\mathbf{62.2_{\pm 0.0}}$	$89.2_{\pm 0.3}$	$88.1_{\pm 0.1}$	$75.2_{\pm 0.1}$
R-Kalman	$87.5_{\pm 0.1}$	$86.1_{\pm 0.2}$	$90.8_{\pm 0.2}$	$90.9_{\pm 0.2}$	$59.5_{\pm 0.0}$	$\mathbf{89.8_{\pm 0.2}}$	$\mathbf{88.2_{\pm 0.2}}$	$74.6_{\pm 0.2}$

we adopt LoRA [Hu et al., 2022], following the framework in [Houlsby et al., 2019, "Han et al., "2024"]. LoRA has emerged as a widely used technique due to its ability to enable efficient model adaptation while maintaining low computational and memory overhead ["Han et al., "2024"].

Evaluation Metrics. To assess the effectiveness of the proposed algorithms, we adopt standard evaluation metrics corresponding to each benchmark. For image classification tasks, we report Top-1 accuracy as the primary performance measure. For the GLUE benchmark, we follow the established evaluation protocol, using accuracy for most tasks, except for CoLA, where we report Matthews correlation coefficient to capture similarity prediction quality. In addition to predictive performance, we evaluate the computational efficiency of the algorithms by measuring their total running time and comparing it against the corresponding baselines.

5.1 Experimental Setup

Datasets. We employ the CIFAR-10/100 [Krizhevsky et al., 2009], Oxford-IIIT Pet [Parkhi et al., 2012], Food-101 [Bossard et al., 2014], and ImageNet [Deng et al., 2009] datasets for image classification tasks due to their wide adoption as standard benchmarks that vary in complexity, scale, and domain diversity. For text classification, we evaluate performance on the GLUE benchmark [Wang et al., 2018], specifically MNLI-mm, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, RTE. These datasets cover a broad spectrum of linguistic phenomena and task types, including natural language inference, paraphrase detection, and sentiment analysis.

Pre-trained Models. For image-based transfer learning, we utilize the ViT-B16 model [Dosovitskiy, 2020], pretrained with DINOv2 on ImageNet [Oquab et al., 2023]. For language tasks, we adopt the RoBERTa-base model [Liu, 2019], which is pretrained on large-scale text corpora. ViT-B16 is a widely adopted Vision Transformer [Alexey, 2020] architecture that serves as a standard benchmark in computer vision. Similarly, RoBERTa-base is among the most commonly used pretrained language models in natural language processing.

Training Settings. For tasks using AdamW and Sophia, we perform a hyperparameter sweep over learning rates in the range $(10^{-5}, 10^{-3})$, training epochs in $(5, 40)$, and batch sizes in $(8, 64)$. In the case of Adam, a learning rate schedule with a constant warmup phase (100 to 500 iterations) improves convergence and is therefore employed. For NGD, we observe that a constant learning rate close to 1.0 is often effective. Consequently, we select among the values $\{0.9, 0.99, 0.9999, 1.0, 1.1\}$. We use a Levenberg-

Table 3: Time Analysis: Total running-time (and per-iteration time), both in seconds, for fine-tuning of MNLI-mm, QQP, QNLI, CIFAR-100, Food-101, and ImageNet-100 on A100 GPU. Lower values indicate faster convergence.

Algorithm	MNLI-mm	QQP	QNLI	CIFAR-100	Food-101	ImageNet-100
AdamW	$10308_{(0.21)}$	$10460_{(0.12)}$	$\mathbf{3011}_{(0.23)}$	$10156_{(0.65)}$	$34466_{(0.91)}$	$51857_{(1.00)}$
NGD	$13008_{(2.36)}$	$12052_{(2.05)}$	$6939_{(2.12)}$	$2780_{(2.78)}$	$5817_{(3.84)}$	$10600_{(5.11)}$
RING	$\mathbf{8835}_{(1.44)}$	$8357_{(1.39)}$	$4811_{(1.47)}$	$945_{(1.89)}$	$\mathbf{4726}_{(3.12)}$	$\mathbf{8483}_{(4.09)}$
RENG	$9081_{(1.48)}$	$8641_{(1.45)}$	$4975_{(1.52)}$	$1990_{(1.99)}$	$4938_{(3.26)}$	$9686_{(4.67)}$
R-Kalman	$8994_{(0.26)}$	$\mathbf{8279}_{(0.28)}$	$3634_{(0.32)}$	$\mathbf{817}_{(1.34)}$	$4817_{(1.45)}$	$10150_{(1.28)}$

Marquardt-type [Martens and Grosse, 2015] damping scheme to update the damping coefficient dynamically based on changes in the loss. Tuning the dampening coefficient is challenging due to its sensitivity to task and initialization. However, we have seen across all experiments that $\rho = 10^{-6}$ for Tikhonov regularized NGD and $\rho = 10^{-4}$ for RING and RENG perform well, so we sweep across $(10^{-6}, 10^{-2})$. For the Kalman implementation, we adopt the approach proposed by [Chang et al., 2022] to reduce the computational complexity from quadratic to linear in the number of trainable parameters (n). LoRA hyperparameters are selected based on empirical effectiveness [Hu et al., 2021]. Specifically, for all algorithms, we set the rank $r = 4$ and scaling factor $\alpha = 4$ for the vision datasets and the rank $r = 8$ and scaling factor $\alpha = 8$ for the language datasets. A sensitivity analysis of the key hyperparameters has been conducted for the proposed algorithms. For further details, please refer to Appendix H. Full hyperparameter details are also provided in Table 4 and Table 5 in Appendix F. All experiments are conducted on A100 GPU platform, and the average and standard deviation over 5 runs with different initial random seeds are reported.

5.2 Main Results and Discussions

In this section, we present the experimental results on the MNLI-mm, QNLI, and SST-2 language datasets, as well as the CIFAR-100, Food-101, and ImageNet-100 vision datasets. Results for the remaining datasets are provided in Appendix G. We employ two evaluation metrics in this study. Figure 2 in this section, along with Figure 4 and Figure 3 in Appendix G, illustrate the validation accuracy curves during training as a function of the number of iterations.

In addition, Table 1 and Table 2 report the final test accuracies, and Table 3 provides time analysis of our algorithms compared to the baselines. In general, the results demonstrate that our proposed gradient-regularized natural gradient optimizers achieve performance that is comparable or superior to established baselines in fine-tuning tasks.

Image Classification Experiments. We report the vision experimental results in Table 1, and Figure 3 of Appendix G. As shown, R-Kalman achieves the best test accuracy on CIFAR-10 (97.1 ± 0.04), and Food-101 (86.2 ± 0.01). RENG demonstrates the best performance on ImageNet-100 (90.10 ± 0.01). Whereas, NGD shows better performance on CIFAR-100 (89.3 ± 0.02) and Oxford-IIIT Pet (93.1 ± 0.02).

GLUE Benchmark Experiments. We report the GLUE benchmark results in Table 2, and Figure 4 of Appendix G. As indicated, RING achieves the highest test accuracy on multiple datasets, including MNLI-mm (98.6 ± 0.1), QNLI (91.5 ± 0.1), MRPC (88.2 ± 0.2), and RTE (76.5 ± 0.2). RENG also outperforms on several tasks, and shows the best performance on QQP (90.2 ± 0.1), SST-2 (92.6 ± 0.1), and CoLA (62.2 ± 0.0). And, R-Kalman demonstrates higher performance in SST-B (89.8 ± 0.2) and MRPC (88.2 ± 0.2), which matches RING.

Overall. We observe that full Fisher-based algorithms, such as RING, RENG, and NGD, tend to perform better on high-volume datasets, including ImageNet, CIFAR-100, MNLI-mm, QQP, QNLI, and SST-2. This can be attributed to the fact that accurately approximating the full Fisher information matrix typically requires $m \approx \mathcal{O}(nlog(n))$ samples in a batch, where n denotes the number of trainable parameters [Vershynin, 2011]. Moreover, R-Kalman, as a Bayesian approach, outperforms its frequentist, gradient-based counterparts when applied to datasets with limited training data. This advantage arises because gradient-based optimizers—especially those that lack effective regularization—are prone to overfitting and often produce overconfident predictions. In contrast, Bayesian algorithms inherently model uncertainty, which leads to improved generalization and a reduced risk of overfitting, as also discussed in [Wilson and Izmailov, 2020].

Time Analysis. The time analysis presented in Table 3 demonstrates that the proposed algorithms—RING, RENG, and R-Kalman—yield significantly faster total running times compared to the AdamW and NGD baselines across most datasets. RING achieves the fastest convergence on MNLI-mm, Food-101, and ImageNet-100, while R-Kalman is superior on QQP and CIFAR-100. Notably, despite AdamW generally exhibiting the lowest per-iteration overhead, the superior performance of our algorithms is attributed to their faster convergence rates (lower total running-time), and this confirms their effectiveness in substantially accelerating the fine-tuning process across diverse language and computer vision datasets. For a detailed analysis of the computational complexity, see Appendix E.

6 CONCLUSION

In this study, we proposed gradient-regularized natural gradient optimizers that extend the classical natural gradient descent framework by incorporating gradient regularization, and can be formulated within both Frequentist and Bayesian paradigms. Our algorithms achieve efficient parameter updates without the computational overhead of directly inverting the Fisher information matrix. By leveraging Newton's iteration in the Frequentist setting and a Kalman-based approach in the Bayesian setting, our methods closely approximate the gradient-regularized natural gradient directions while maintaining computational feasibility. Extensive benchmarking on diverse image and language datasets demonstrates that our optimizers, particularly R-Kalman in low-data regimes and RING/RENG in large-data regimes, achieve comparable or superior performance to established baselines like Adam, Sophia, and NGD in fine-tuning tasks.

References

Hossein Abdi, Mingfei Sun, Andi Zhang, Samuel Kaski, and Wei Pan. Loko: Low-rank kalman optimizer for online fine-tuning of large models. *arXiv preprint arXiv:2410.11551*, 2024.

Hossein Abdi, Mingfei Sun, and Wei Pan. Bayesian natural gradient fine-tuning of clip models via kalman filtering. *arXiv preprint arXiv:2511.01694*, 2025.

Ilan Adler, Zhiyue T Hu, and Tianyi Lin. New proximal newton-type methods for convex optimization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 4828–4835. IEEE, 2020.

Dosovitskiy Alexey. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020.

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

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 404–413. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/balles18a.html`.

David G. T. Barrett and Benoit Dherin. Implicit gradient regularization, 2020.

Sue Becker, Yann Le Cun, et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, volume 2, 1988.

Adi Ben-Israel. An iterative method for computing the generalized inverse of an arbitrary matrix. *Mathematics of Computation*, 19(91):452–455, 1965.

Adi Ben-Israel and Dan Cohen. On iterative computation of generalized inverses and associated projections. *SIAM Journal on Numerical Analysis*, 3 (3):410–419, 1966. doi: 10.1137/0703035. URL `https://doi.org/10.1137/0703035`.

Alberto Bernacchia, Mate Lengyel, and Guillaume Hennequin. Exact natural gradient in deep linear networks and its application to the nonlinear case. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/7f018eb7b301a66658931cb8a93fd6e8-Paper.pdf`.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer, 2014.

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

Peter G Chang, Kevin Patrick Murphy, and Matt Jones. On diagonal approximations to the extended kalman filter for online training of bayesian neural networks. In *Continual Lifelong Learning Workshop at ACML 2022*, 2022.

Peter G Chang, Gerardo Durán-Martín, Alexander Y Shestopaloff, Matt Jones, and Kevin Murphy. Low-rank extended kalman filtering for online learning of neural networks from streaming data. *arXiv preprint arXiv:2305.19535*, 2023.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

El Mahdi Chayti, Nikita Doikov, and Martin Jaggi. Unified convergence theory of stochastic and variance-reduced cubic newton methods. *arXiv preprint arXiv:2302.11962*, 2023.

Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Nikita Doikov, El Mahdi Chayti, and Martin Jaggi. Second-order optimization with lazy hessians. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8138–8161. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/doikov23a.html`.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

H. Drucker and Y. Le Cun. Double backpropagation increasing generalization performance. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 145–150 vol.2, 1991. doi: 10.1109/IJCNN.1991.155328.

Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE transactions on neural networks*, 3(6): 991–997, 1992.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85 (410):398–409, 1990.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2232–2241. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/ghorbani19b.html`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018.

Zeyu "Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, and others". "parameter-efficient fine-tuning for large models: A comprehensive survey". *"arXiv preprint arXiv:2403.14608"*, "2024".

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Philipp Hennig, Jon Cockayne, Jonathan Wenger, and Marvin Pförtner. Computation-aware kalman filtering and smoothing. 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Flat Minima. *Neural Computation*, 9(1):1–42, 01 1997. ISSN

0899-7667. doi: 10.1162/neco.1997.9.1.1. URL https://doi.org/10.1162/neco.1997.9.1.1.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Stanislaw Jastrzebski, Devansh Arpit, Oliver Astrand, Giancarlo B Kerg, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, and Krzysztof J Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *International Conference on Machine Learning*, pages 4772–4784. PMLR, 2021.

Zhiwei Jia and Hao Su. Information-theoretic local minima characterization and regularization. In *International Conference on Machine Learning*, pages 4773–4783. PMLR, 2020.

Matt Jones, Peter Chang, and Kevin Murphy. Bayesian online natural gradient (bong). *arXiv preprint arXiv:2405.19681*, 2024.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

Ryo Karakida, Tomoumi Takase, Tomohiro Hayase, and Kazuki Osawa. Understanding gradient regularization in deep learning: Efficient finite-difference computation and implicit bias. In *International Conference on Machine Learning*, pages 15809–15827. PMLR, 2023.

Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule. *Journal of Machine Learning Research*, 24(281):1–46, 2023.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be, 2023.

Longze Li, Jiang Chang, Aleksandar Vakanski, Yachun Wang, Tiankai Yao, and Min Xian. Uncertainty quantification in multivariable regression for material property prediction with bayesian neural networks. *Scientific Reports*, 14(1):10543, 2024.

Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? a second-order perspective. *arXiv preprint arXiv:2402.03496*, 2024.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training, 2023.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2019.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

David John Cameron Mackay. *Bayesian methods for adaptive models*. California Institute of Technology, 1992.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2015.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, page 543, 1983.

Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in

deep neural networks. *Advances in neural information processing systems*, 28, 2015.

Yann Ollivier. Online natural gradient as a kalman filter. 2018.

Yann Ollivier. The extended kalman filter is a natural gradient descent in trajectory space. *arXiv preprint arXiv:1901.00696*, 2019.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Yan Pan and Yuanzhi Li. Toward understanding why adam converges faster than sgd for transformers, 2023.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks, 2013.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Nicolas Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. *Advances in neural information processing systems*, 20, 2007.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond, 2016.

Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks, 2017.

Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):9481–9488, May 2021. ISSN 2159-5399. doi: 10.1609/aaai.v35i11.17142. URL `http://dx.doi.org/10.1609/aaai.v35i11.17142`.

Günther Schulz. Iterative berechung der reziproken matrix. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 13(1):57–59, 1933.

VE Shamanskii. A modification of newton's method. *Ukrainian Mathematical Journal*, 19(1):118–122, 1967.

Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Clement Bazan, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, et al. Variational learning is effective for large deep networks. *arXiv preprint arXiv:2402.17641*, 2024.

Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=rq_Qr0c1Hyo`.

Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021b.

Valentin Thomas, Fabian Pedregosa, Bart Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics*, pages 3503–3513. PMLR, 2020.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices, 2011. URL `https://arxiv.org/abs/1011.3027`.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL `https://aclanthology.org/W18-5446`.

Chang-yu Wang, Yuan-yuan Chen, and Shou-qiang Du. Further insight into the shamanskii modification of newton method. *Applied mathematics and computation*, 180(1):46–52, 2006.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

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

Guodong Zhang, James Martens, and Roger B Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15383–15393. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b05b57f6add810d3b7490866d74c0053-Paper.pdf.

Lin Zhang, Shaohuai Shi, and Bo Li. Eva: Practical second-order optimization with kronecker-vectorized approximation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_Mic8V96Voy.

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective, 2024.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18795–18806. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes. Please refer to Appendix E]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No. Some parts of the implementation depend on confidential components from a collaborative project and cannot be shared publicly. We will release the code once it can be made publicly available.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes. Please refer to Appendix D]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes. Please refer to subsection 5.1, Appendix B, and Appendix F]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes. Please refer to subsection 5.1, Appendix B, and Appendix F]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Technical Appendices and Supplementary Material

A Related work

A.1 Gradient Regularization

The generalization of deep neural networks has been closely linked to implicit and explicit regularization mechanisms, which emerge from factors such as finite-step stochastic updates and the use of relatively small batch sizes during optimization. A central theme in prior work has been devoted to connecting properties of the loss landscape—particularly the sharpness of local minima, as well as the flatness and curvature of the loss surface—to the generalization behavior of SGD.

A substantial body of work has examined how SGD induces implicit gradient regularization, which favors flat minima and thereby improves generalization [Neyshabur, 2017, Chaudhari and Soatto, 2018]. Some studies have investigated explicit and implicit gradient regularization of SGD in deep learning [Smith et al., 2021a, Barrett and Dherin, 2020]. Using Euler discretization of stochastic differential equations for SGD and backward error analysis, they found that the discrete-time update of the usual gradient descent implicitly regularizes the norm of stochastic gradients when its dynamics are mapped to the continuous-time counterpart. To identify these so-called "flat-minima," previous works have also endeavoured towards finding different generalization metrics that can classify a sharp/flat minima. Path-SGD proposed by [Neyshabur et al., 2015], which uses a path-wise norm regularization to achieve better generalization, albeit incurring computational overhead in estimating such a factor. Frobenius norm and spectral norm of the second-order preconditioner, specifically the Hessian matrix, have also been proposed to classify flat minima [Wu et al., 2017, Keskar and Socher, 2017]. The local entropy of the loss landscape has also been proposed by Chaudhari et al. [2019], which defines an entropy-guided SGD and shows faster convergence, but for small models on small datasets. Recently, Jastrzebski et al. [2021] and Karakida et al. [2023] have shown that implicit regularization effects of large learning rates can be well explained by the trace of the Fisher information matrix from an early phase of training and have empirically demonstrated an effect called *catastrophic Fisher explosion* for ResNet models on a subset of ImageNet. Work by [Jia and Su, 2020] proposes a regularizer based on the determinant of FIM, which can bias the optimization trajectory towards finding good local minima (minima with better generalization) and also provide a PAC-based generalization bound.

A.2 Frequentist Approach:

In the Frequentist optimization approach, a loss function is minimized, typically by gradient-based methods. Gradients are treated as deterministic quantities, and optimization algorithms compute the steepest descent direction using gradient-based first-order or second-order algorithms.

Stochastic gradient descent (SGD) [Robbins and Monro, 1951] and its variants have long been the backbone of training large-scale neural networks. Among first-order methods, momentum-based techniques like Nesterov Accelerated Gradient (NAG) [Nesterov, 1983], and adaptive algorithms such as Adam [Kingma and Ba, 2014], Adagrad [Duchi et al., 2011], AdaDelta [Zeiler, 2012], RMSProp [Graves, 2013], RADAM [Liu et al., 2019], and AdaBelief [Zhuang et al., 2020] have gained considerable popularity due to their ability to adapt learning rates to the geometry of the loss landscape. While SGD is simple and memory-efficient, its constant learning rate can lead to inconsistent parameter updates in the inherently non-Euclidean geometry of deep neural networks. Adaptive optimizers address this by adjusting the learning rate based on local curvature, often leading to faster convergence and improved performance in certain settings. For example, Adam uses the square root of the diagonal of the estimated second moment (variance) of gradients to scale the update direction. However, this often results in instability at high learning rates. To address such issues, RADAM introduces a variance rectification mechanism to justify the warm-up phase in Adam.

Empirical studies highlight that while SGD (with or without momentum) performs competitively in computer vision tasks, adaptive optimizers like Adam significantly outperform SGD in training transformer-based models for NLP tasks [Zhang et al., 2024]. However, the effectiveness of Adam remains poorly understood. One explanation is the presence of heavy noise in attention layers [Zhang et al., 2020], which affects the performance of SGD more severely. [Kunstner et al., 2023] observed that Adam continues to outperform SGD even with large batch sizes, attributing this to the signed nature of Adam's updates. Indeed, Adam can be seen as a variance-adapted signed descent method [Zhuang et al., 2020, Balles and Hennig, 2018]. Another hypothesis is rooted in the geometry of the optimization problem. [Zhang et al., 2024] argues that the heterogeneous Hessian spectrum in attention layers contributes to slow convergence under SGD. This aligns with the theoretical insights of [Thomas et al., 2020], who noted that the Hessian reflects curvature under the data distribution, which may result in poor generalization and data inefficiency.

Given the limitations of both SGD and Adam, second-order information has been explored to better precondition gradient updates. While the Hessian provides curvature information, its computation is expensive and, more critically, it may not be positive semi-definite due to the non-convex nature of deep learning objectives. Indeed, empirical studies [Sagun et al., 2016, 2017, Ghorbani et al., 2019] show that the Hessian spectrum contains negative eigenvalues, particularly near convergence. An alternative is the Fisher Information Matrix (FIM), which is always positive semi-definite and defined as the covariance of gradients of the negative log-likelihood loss under the model distribution [Amari, 1998]. The natural gradient [Bernacchia et al., 2018] leverages the inverse FIM to scale gradients appropriately. Although computing and inverting the FIM is computationally expensive, several approximation methods have made natural gradient descent feasible. K-FAC [Martens and Grosse, 2015] decomposes the FIM into Kronecker factors for efficient inversion. Shampoo [Gupta et al., 2018] estimates curvature using stochastic gradients, while [Zhang et al., 2023] proposes a memory-efficient rank-one update scheme. More recently, [Liu et al., 2023] proposed Sophia, which leverages a diagonal approximation of the Hessian and employs coordinate-wise clipping to mitigate rapid changes in curvature due to small batch sizes. Similarly, [Pan and Li, 2023] emphasized the role of directional sharpness in adaptive optimizers and showed that coordinate-wise clipping consistently improves optimization stability. Overall, these approaches highlight the importance of leveraging second-order information—particularly the Fisher matrix—in training large models like transformers, where curvature is both heterogeneous and evolving.

Beyond optimization speed, several studies have also investigated how optimization strategies influence generalization. Regularization techniques such as Double Backpropagation [Drucker and Le Cun, 1991] were early attempts to control curvature by penalizing gradient norms. More recent works [Sankar et al., 2021] suggest that regularizing the trace of the Hessian improves generalization in SGD. Similarly, [Barrett and Dherin, 2020] demonstrates that gradient regularization can promote flatter minima, thereby enhancing robustness to noise and improving test-time performance.

A.3 Bayesian Approach:

Bayesian approaches interpret the trainable parameters as random variables and aims to infer their posterior distribution given the data by applying Bayes' theorem [Murphy, 2023, 2012], which is known as *exact Bayesian inference*. The exact Bayesian inference is computationally intractable for high-dimensional deep models due to the need to integrate over the entire possible values in the estimation of the posterior and predictive distribution. Consequently, approximate methods are employed, which can be broadly classified into sampling-based and approximate-inference approaches [Khan and Rue, 2023].

Sampling methods like Markov Chain Monte Carlo (MCMC) [Gelfand and Smith, 1990], including Hamiltonian Monte Carlo (HMC) [Neal, 2012] and Metropolis-Hastings (MH) [Chib and Greenberg, 1995] work by building a Markov chain whose equilibrium distribution matches the target posterior. Then, one can generate samples that approximate the posterior distribution by running the chain for many iterations. These sampling-based approaches are either infeasible or slow for deep learning with a large parameter space [Khan and Rue, 2023].

Approximate-inference methods like Variational Inference (VI) [Jordan et al., 1999] approximate the posterior with a simpler distribution $q(\boldsymbol{\psi})$ (e.g., a Gaussian) and optimize its parameters $\boldsymbol{\psi}$ to minimize the KL divergence of $q(\boldsymbol{\psi})$ given $p(\boldsymbol{\theta} \mid \mathcal{D})$. This is equivalent to minimizing the variational loss (or negative ELBO) $\mathcal{L}(\psi) = -\mathbb{E}_{q(\boldsymbol{\psi})}[\log p(\mathcal{D} \mid \boldsymbol{\theta})] - \mathrm{KL}(q(\boldsymbol{\psi}) \mid p(\boldsymbol{\theta}))$ [Li et al., 2024]. Laplace algorithm [Mackay, 1992], as another approximate-inference method, approximates the posterior as a Gaussian with the mean at the maximum a posteriori (MAP) estimate, and covariance of the Hessian of the log-posterior. These approximate inference al-

gorithms use a Frequentist optimization approach, mostly first-order gradient-based techniques, which rely solely on gradient information, like *Bayes by Backprop* introduced by [Blundell et al., 2015]. While many early methods relied on first-order optimization, recent studies have demonstrated the advantages of second-order methods for optimizing the variational loss. Works such as [Lin et al., 2024], [Shen et al., 2024], and [Khan and Rue, 2023] employ second-order optimization techniques and leverage the curvature information of the variational loss to enable more informative updates, leading to faster convergence and better uncertainty estimation.

The Extended Kalman Filter (EKF) provides a straightforward solution for approximate Bayesian inference, and enables closed-form updates for the posterior distribution over neural network parameters [Jones et al., 2024]. Recent studies [Ollivier, 2018, 2019] demonstrate that the EKF update rule is equivalent to an online natural gradient descent. This effectively offers a new perspective into second-order optimization: instead of computing various large pre-conditioning matrices and their inversions, we can recursively infer the natural direction update as the parameter update step in the Kalman algorithm. In contrast to NGD, the Kalman formulation doesn't require any FIM inversion, which is computationally expensive for large models. Although promising, the practical implementation of such techniques remained infeasible for large models with millions of parameters because of the huge size of the matrix involved in the Kalman algorithm. Various studies have since addressed the scalability challenges of the EKF optimizer. For instance, [Chang et al., 2022, Abdi et al., 2025, 2024] introduced a diagonal covariance matrix approximation, while [Chang et al., 2023] proposed a low-rank plus diagonal decomposition of the posterior precision matrix. [Hennig et al., 2024] developed a matrix-free iterative algorithm to further enhance efficiency.

In parallel, other works have explored the intersection of Bayesian inference and natural gradient methods. The Topmoumoute online natural gradient algorithm [Roux et al., 2007] pioneered an online formulation of the natural gradient with connections to information geometry. Building on this idea, [Khan and Rue, 2023] bridges natural gradient descent and Bayesian learning by proposing a general update rule based on the principle of minimizing the divergence between successive posterior distributions. More recently, [Jones et al., 2024] extended this framework to variational inference in neural networks and introduced the Bayesian Online Natural Gradient (BONG) algorithm.

B Algorithm

The pseudocode of the proposed algorithms has been shown in Algorithm 1 and Algorithm 2.

Algorithm 1 RING: Regularized Implicit Natural Gradient
RENG: Regularized Explicit Natural Gradient

Initialization: $\mathbf{W}_i$ $\forall i \in [1, L]$, number of weight layers L, learning rate α, gradient regularizer ρ, lambda λ, lambda discount factor ϕ, Fisher skip frequency S.
for $k = 1, 2, ...$ **do**
  Compute minibatch loss $\mathcal{L}_k$ and gradients $\nabla \ln p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$.
  if $k \bmod S = 0$ **then**
    RENG: Use the gradient to calculate the regularized loss in Equation 9
    Save the activation matrix $\mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{x}_{i-1}\boldsymbol{x}_{i-1}^\top]$ and batch gradients $\mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{e}_i\boldsymbol{x}_{i-1}^\top]$.
    Sample output $\mathbf{y} \sim p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ and compute gradient.
    Save the error matrix $\mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\boldsymbol{e}_i\boldsymbol{e}_i^\top]$.
    Update $\mathbf{W}_i$ using RING: Equation 10a, or RENG: Equation 10b.
  else
    Update regularized activation and error matrices via Equation 11.
    Update $\mathbf{W}_i$ using RING: Equation 10a, or RENG: Equation 10b.
  end if
end for

Algorithm 2 Regularized Kalman Algorithm

Initialization: $p(\boldsymbol{\theta}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, $\mathbf{R}_0 = \mathbf{O}_{d_o \times d_o}$, $\mathbf{Q} = \mathbf{O}_{d_i \times d_i}$
for $k = 1, 2, ...$ **do**
  Prediction:
  $\boldsymbol{\mu}_{k|k-1} = \boldsymbol{\mu}_{k-1}$
  $\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{\Sigma}_{k-1} + \mathbf{Q}$
  Pre-Updating:
  $\hat{\mathbf{y}}_k \simeq h(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})$
  $\mathbf{H}_k = \nabla_{\boldsymbol{\theta}} h|_{(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})}$
  $\hat{\mathbf{R}}_k = (\mathbf{y}_k - \hat{\mathbf{y}}_k)(\mathbf{y}_k - \hat{\mathbf{y}}_k)^\top + \mathbf{H}_k \boldsymbol{\Sigma}_{k|k-1} \mathbf{H}_k^\top$
  $\mathbf{R}_k = \beta \mathbf{R}_{k-1} + (1 - \beta)\hat{\mathbf{R}}_k$
  Updating:
  $\tilde{\mathbf{K}}_k = \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top \left(\mathbf{H}_k \boldsymbol{\Sigma}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k(\mathbf{I} + \rho \mathbf{R}_k)^{-1}\right)^{-1}$
  $\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} + \tilde{\mathbf{K}}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k)$
  $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{k|k-1} - \tilde{\mathbf{K}}_k \mathbf{H}_k \boldsymbol{\Sigma}_{k|k-1}$
  Output: Posterior: $p(\boldsymbol{\theta}_k \mid \mathcal{D}_{1:k}) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
end for

C Convergence Analysis of Theorem 4.3

In this section, we provide detailed proof stating that the output of a two-layer neural network shows exponential decay with GRNG optimizers.

Proof. Considering $\mathbf{y}$ as the full-batch ground truth, $\mathbf{H}$ as Jacobian matrix, and $\mathbf{G} = \mathbf{H}\mathbf{H}^\top$ as Gram matrix, we introduce a learning rate factor η and calculate the difference of predictions between two consecutive iterations:

$$\hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k = \hat{\mathbf{y}}\left(\boldsymbol{\theta}_k - \eta(\mathbf{F}_k + \rho\|\nabla\mathcal{L}\|_2^2)^{-1}\mathbf{H}_k^\top(\hat{\mathbf{y}}_k - \mathbf{y})\right) - \hat{\mathbf{y}}(\boldsymbol{\theta}_k).$$

For simplicity, we use $\tilde{\rho}_k$ instead of $\rho\|\nabla\mathcal{L}_k\|_2^2$ and later on we use the expression $\nabla\mathcal{L}_k = \mathbf{H}_k^\top(\hat{\mathbf{y}}_k - \mathbf{y})$ to get the final bound:

$$\hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k = \hat{\mathbf{y}}\left(\boldsymbol{\theta}_k - \eta\mathbf{H}_k^\top(\mathbf{G}_k + \tilde{\rho}_k\mathbf{I})^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right) - \hat{\mathbf{y}}(\boldsymbol{\theta}_k)$$

$$= -\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}(s))}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top(\mathbf{G}_k + \tilde{\rho}_k\mathbf{I})^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds$$

$$\leq -\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}(s))}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top(\mathbf{G}_k^{-1} - \tilde{\rho}_k\mathbf{G}_k^{-1}\mathbf{G}_k^{-1})(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds$$

$$= -\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}(s))}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top\mathbf{G}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds + \eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}(s))}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top\tilde{\rho}_k\mathbf{G}_k^{-1}\mathbf{G}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds$$

$$= \underbrace{-\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}_k)}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top\mathbf{G}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds}_{①} \quad \underbrace{+\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}_k)}{\partial\boldsymbol{\theta}^\top} - \frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}(s))}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top\mathbf{G}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds}_{②}$$

$$\underbrace{+\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}_k)}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top\tilde{\rho}_k\mathbf{G}_k^{-1}\mathbf{G}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds}_{③} \quad \underbrace{-\eta\int_{s=0}^{1}\left\langle\frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}_k)}{\partial\boldsymbol{\theta}^\top} - \frac{\partial\hat{\mathbf{y}}(\boldsymbol{\theta}(s))}{\partial\boldsymbol{\theta}^\top}, \mathbf{H}_k^\top\tilde{\rho}_k\mathbf{G}_k^{-1}\mathbf{G}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y})\right\rangle ds}_{④}$$

To evaluate the L2 norms, we define $\alpha_k = 1 + \rho\kappa(\mathbf{G})\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2$, where $\kappa(\mathbf{G})$ is the condition number, i.e., $\lambda_{max}(\mathbf{G})/\lambda_{min}(\mathbf{G})$; and $\lambda(\cdot)$ is the eigenvalue operator. Observing that terms ① and ③ do not depend on s, and using the spectral bound from Zhang et al. [2019] for ② and ④, we can define the respective norms as:

$$\begin{aligned}
\|①\|_2 &= \eta\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2 \\
\|②\|_2 &= \eta C\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2 \\
\|③\|_2 &= \eta\alpha_k\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2 \\
\|④\|_2 &= \eta C\alpha_k\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2
\end{aligned}$$

Hence, the total magnitude of change between iterations is bounded by:

$$\|\hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k\|_2^2 \leq \eta^2(1+\alpha_k)^2(1+C)^2\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2.$$

To examine convergence relative to the initial error, we expand the squared output error at the $(k+1)^{th}$ iteration:

$$\begin{aligned}
\|\mathbf{y} - \hat{\mathbf{y}}_{k+1}\|_2^2 &= \|(\mathbf{y} - \hat{\mathbf{y}}_k) - (\hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k)\|_2^2 \\
&= \|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2 + \|\hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k\|_2^2 - 2\langle\mathbf{y} - \hat{\mathbf{y}}_k, \hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k\rangle \\
&\leq \|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2 + \eta^2(1+\alpha_k)^2(1+C)^2\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2 - 2\eta(1+\alpha_k)(1+C)\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2 \\
&\leq \left[1 + \eta^2(1+\alpha_k)^2(1+C)^2 - 2\eta(1+\alpha_k)(1+C)\right]\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2.
\end{aligned}$$

For the residual error to strictly decay, we require the constant factor to be less than or equal to $(1-\eta)$:

$$\|\mathbf{y} - \hat{\mathbf{y}}_{k+1}\|_2^2 \leq (1-\eta)\|\mathbf{y} - \hat{\mathbf{y}}_k\|_2^2.$$

This condition is satisfied when:

$$1 + \eta^2(1+\alpha_k)^2(1+C)^2 - 2\eta(1+\alpha_k)(1+C) \leq 1 - \eta.$$

By rearranging the terms, we find the equivalent upper bound for the learning rate:

$$\eta \leq \frac{2(1+\alpha_k)(1+C) - 1}{(1+\alpha_k)^2(1+C)^2}.$$

As $1+\alpha_k > 1$ and $1+C > 1$, this makes the right-hand side expression positive, guaranteeing a feasible step size for global convergence. Furthermore, the denominator grows quadratically and the numerator grows linearly, so the maximum permissible η is bounded predominantly by the smallest value of α_k. □

From this, let us define $\hat{M} = \max_k \frac{2(1+\alpha_k)(1+C)-1}{(1+\alpha_k)^2(1+C)^2}$. When η satisfies $\eta \leq \hat{M}$, the loss exhibits geometric decay relative to the functional output of the initial iteration:

$$\|\mathbf{y} - \hat{\mathbf{y}}_{k+1}\|_2^2 \leq (1-\eta)^{k+1}\|\mathbf{y} - \hat{\mathbf{y}}_0\|_2^2.$$

And we finally prove Corollary 4.4.

D Proofs and Detailed Derivations

This section presents the proofs and detailed derivations of the proposed algorithms, offering a comprehensive understanding of their underlying principles and mechanisms.

D.1 Proof of Equation 11

Proof. Let us assume that at iteration k we have a damping coefficient ρ_k, and have computed the activation and error matrices of each layer. Based on *Lazy Fisher*, we assume that $\mathbf{\Lambda}_{i-1k+1} \approx \mathbf{\Lambda}_{i-1k}$, and $\mathbf{\Gamma}_{i-1k+1} \approx \mathbf{\Gamma}_{i-1k}$. However, the damping coefficient ρ_{k+1} is allowed to be updated. From matrix differential theory, we know that for an invertible matrix $\mathbf{A}$, we have $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, then the differential of the inverse is given by: $d\mathbf{A}^{-1} = -\mathbf{A}^{-1}d\mathbf{A}\mathbf{A}^{-1}$. In our context, $\mathbf{A}_k$ represents either the regularized activation matrix $\mathbf{A}_k = \tilde{\mathbf{\Lambda}}_k = (\mathbf{\Lambda}_k + \lambda_k\mathbf{I})$ or the regularized error matrix $\mathbf{A}_k = \tilde{\mathbf{\Gamma}}_k = (\mathbf{\Gamma}_k + \lambda_k\mathbf{I})$. Under the *Lazy Fisher* assumption, where the error/activation matrix remains unchanged across consecutive timesteps, and only the damping factor evolves, we approximate the change in the matrix as $d\mathbf{A}_k \approx (\lambda_{k+1} - \lambda_k)\mathbf{I} = d\lambda_k\mathbf{I}$. Then, we add $d\mathbf{A}_k^{-1}$ to the $\mathbf{A}_k^{-1}$ to approximate the $\mathbf{A}_{k+1}^{-1}$ without explicitly computing the inversion of $\mathbf{A}_{k+1}$. Hence, for each layer i, the new inverted matrix can be approximated as $\mathbf{A}_{k+1}^{-1} \approx \mathbf{A}_k^{-1} + d\mathbf{A}_k^{-1}$. □

D.2 Proof of Equation 10

Remark D.1. *The update direction $\boldsymbol{\delta}^* = -\mathbf{F}(\boldsymbol{\theta})^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ is the unique minimizer of the quadratic approximation of the loss function $\mathcal{L}$ around the point $\boldsymbol{\theta}$, given by:*

$$\mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\delta}) = \mathcal{L}(\boldsymbol{\theta}) + \boldsymbol{\delta}^T\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\delta}^\top\mathbf{F}(\boldsymbol{\theta})\boldsymbol{\delta} + \mathcal{O}(|\boldsymbol{\delta}|^3).$$

Remark D.2. *Considering the regularized loss function:*

$$\mathcal{L}_T(\boldsymbol{\theta} + \boldsymbol{\delta}) = \mathcal{L}(\boldsymbol{\theta}) + \boldsymbol{\delta}^T\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2}\big[\boldsymbol{\delta}^\top\mathbf{F}(\boldsymbol{\theta})\boldsymbol{\delta} + \rho\boldsymbol{\delta}^\top\boldsymbol{\delta}\big] + \mathcal{O}(|\boldsymbol{\delta}|^3).$$

This expression can be rewritten as:

$$\mathcal{L}_T(\boldsymbol{\theta} + \boldsymbol{\delta}) = \mathcal{L}(\boldsymbol{\theta}) + \boldsymbol{\delta}^T\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\delta}^\top\big(\mathbf{F}(\boldsymbol{\theta}) + \rho\mathbf{I}\big)\boldsymbol{\delta} + \mathcal{O}(|\boldsymbol{\delta}|^3).$$

Thus, based on Remark D.1, the new natural direction will be:

$$\boldsymbol{\delta}^* = -(\mathbf{F}(\boldsymbol{\theta}) + \rho\mathbf{I})^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}).$$

Proof. Considering the regularized loss function expressed in Equation 9, and using the Remark D.1 and D.2, one can find the optimum update direction as:

$$\boldsymbol{\delta}^* = -\Big(\mathbf{F}(\boldsymbol{\theta}) + \rho\left\|\mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})]\right\|_2^2\mathbf{I}\Big)^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}).$$

It can be expressed by activation and error matrices as:

$$\boldsymbol{\delta}^* = (\mathbf{\Lambda}_i \otimes \mathbf{\Gamma}_i + \rho\left\|\mathbf{\Lambda}_i \otimes \mathbf{\Gamma}_i\right\|\mathbf{I})^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}),$$

which is:

$$\boldsymbol{\delta}^* = (\mathbf{\Lambda}_i \otimes \mathbf{\Gamma}_i + \rho\left\|\mathbf{\Lambda}_i\right\|\left\|\mathbf{\Gamma}_i\right\|\mathbf{I})^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}).$$

Again, using the approximation $(A \otimes B + \rho I) \approx (A + \sqrt{\rho}I) \otimes (B + \sqrt{\rho}I)$, we have:

$$\boldsymbol{\delta}^* = [\mathbf{\Lambda}_i + \sqrt{\rho}\left\|\mathbf{\Lambda}_i\right\|\mathbf{I}]^{-1} \otimes [\mathbf{\Gamma}_i + \sqrt{\rho}\left\|\mathbf{\Gamma}_i\right\|\mathbf{I}]^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}).$$

□

D.3 Proof of Equation 14

Proof. The information form of the Kalman filter (information filtering) can be derived from its standard formulation. Substituting the Kalman gain from Equation 14a, into the standard formulation of covariance update ($\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}$) yields:

$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{k|k-1} - \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top \left(\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1}\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}.$$

The Woodbury matrix identity states: $(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$. By identifying $\mathbf{A} = \boldsymbol{\Sigma}_{k|k-1}^{-1}$, $\mathbf{U} = \mathbf{H}_k^\top$, $\mathbf{C} = \mathbf{R}_k^{-1}$, and $\mathbf{V} = \mathbf{H}_k$, we apply the identity to obtain the information form of the covariance update: $\boldsymbol{\Sigma}_k^{-1} = \boldsymbol{\Sigma}_{k|k-1}^{-1} + \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k$.

□

D.4 On the Equivalence of Kalman Filtering and NGD

Lemma D.3. *The update step in the Kalman algorithm, as given in Equation 14b is equivalent to:*

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} - \boldsymbol{\Sigma}_k\nabla_{\boldsymbol{\theta}}\mathcal{L}_k$$

Proof. To prove Lemma D.3, consider the negative log-likelihood loss function at step k, for a Gaussian or exponential family distribution, defined as: $\mathcal{L}_k = \frac{1}{2}(\hat{\mathbf{y}}_k - \mathbf{y}_k)^\top\mathbf{R}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y}_k)$. Applying the chain rule, the gradient of the loss with respect to the parameters is given by: $\nabla_{\boldsymbol{\theta}}\mathcal{L}_k = \nabla_{\boldsymbol{\theta}}^\top\hat{\mathbf{y}}\nabla_{\hat{\mathbf{y}}}\mathcal{L}$. We note that $\nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}} = \mathbf{H}_k$, and $\nabla_{\hat{\mathbf{y}}}\mathcal{L} = \mathbf{R}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y}_k)$. Therefore, the gradient becomes $\nabla_{\boldsymbol{\theta}}\mathcal{L}_k = \mathbf{H}_k^\top\mathbf{R}_k^{-1}(\hat{\mathbf{y}}_k - \mathbf{y}_k)$. Rearranging, we obtain the prediction error as $(\mathbf{y}_k - \hat{\mathbf{y}}_k) = -\mathbf{R}_k\mathbf{H}_k^{\top -1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_k$. Substituting this expression into the Kalman filter for the parameter update Equation 14b, we get:

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} - \mathbf{K}_k\mathbf{R}_k\mathbf{H}_k^{\top -1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_k.$$

Next, We can replace $\mathbf{K}_k\mathbf{R}_k$ with $\boldsymbol{\Sigma}_k\mathbf{H}_k^\top$ because of this identity:

$$\mathbf{K}_k\mathbf{R}_k = \mathbf{K}_k(\mathbf{R}_k + \mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top) - \mathbf{K}_k\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top,$$

where based on the Equation 14a, we will have: $\mathbf{K}_k(\mathbf{R}_k + \mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top) = \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top$. So,

$$\mathbf{K}_k\mathbf{R}_k = \boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top - \mathbf{K}_k\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\mathbf{H}_k^\top = \left(\boldsymbol{\Sigma}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\right)\mathbf{H}_k^\top.$$

By the standard formulation of covariance update, we know $\left(\boldsymbol{\Sigma}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\boldsymbol{\Sigma}_{k|k-1}\right) = \boldsymbol{\Sigma}_k$, and thus: $\mathbf{K}_k\mathbf{R}_k = \boldsymbol{\Sigma}_k\mathbf{H}_k^\top$. Substituting this into the previous expression for $\boldsymbol{\mu}_k$, we obtain:

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} - \boldsymbol{\Sigma}_k\mathbf{H}_k^\top\mathbf{H}_k^{\top -1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_k.$$

Finally, simplifying yields $\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} - \boldsymbol{\Sigma}_k\nabla_{\boldsymbol{\theta}}\mathcal{L}_k$. □

Proof of Kalman–NGD Equivalence

Proof. We begin with the definition of the Fisher information matrix given in Equation 2. We can rewrite it as: $\mathbf{F}(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}}\mathcal{L}\cdot\nabla_{\boldsymbol{\theta}}\mathcal{L}^\top]$. At time step k, the new data point contributes statistical information in the form $\mathbf{F}(\boldsymbol{\theta}_k) = \nabla_{\boldsymbol{\theta}}\mathcal{L}_k\cdot\nabla_{\boldsymbol{\theta}}\mathcal{L}_k^\top$ to Fisher information matrix. Applying the chain rule, we write $\nabla_{\boldsymbol{\theta}}\mathcal{L}_k = \nabla_{\hat{\mathbf{y}}}\mathcal{L}_k\cdot\nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k$, which gives:

$$\mathbf{F}(\boldsymbol{\theta}_k) = (\nabla_{\hat{\mathbf{y}}}\mathcal{L}_k\cdot\nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k)\cdot(\nabla_{\hat{\mathbf{y}}}\mathcal{L}_k\cdot\nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k)^\top.$$

Using matrix multiplication rules: $\mathbf{F}(\boldsymbol{\theta}_k) = \nabla_{\boldsymbol{\theta}}^\top\hat{\mathbf{y}}_k\cdot\nabla_{\hat{\mathbf{y}}}^2\mathcal{L}_k\cdot\nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k$. We already know that $\mathbf{H}_k = \nabla_{\boldsymbol{\theta}}\hat{\mathbf{y}}_k$, and based on assumption of Gaussian (or more broadly, for an exponential family) likelihood, we obtain $\mathbf{R}_k^{-1} = \nabla_{\hat{\mathbf{y}}}^2\mathcal{L}_k$. Consequently, $\mathbf{F}(\boldsymbol{\theta}_k) = \mathbf{H}_k^\top\mathbf{R}_k^{-1}\mathbf{H}_k$, where corresponds to the update term in the information filtering covariance update in Equation 14c, i.e. $\mathbf{F} \equiv \boldsymbol{\Sigma}^{-1}$. Therefore, considering Lemma D.3, the update direction in NGD and Kalman filter is equivalent to each other: $\mathbf{F}(\boldsymbol{\theta}_k)^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_k \equiv \boldsymbol{\Sigma}_k\nabla_{\boldsymbol{\theta}}\mathcal{L}_k$. □

E Computational Complexity

To analyze the computational complexity of the proposed algorithms, let n denote the number of trainable parameters and d_o the number of model outputs. Assume that at each iteration, a minibatch of size m is received, for a total of T iterations. Consider a model composed of L layers, where the largest weight matrix has a size of at most $\omega \times \omega$. The computational complexity of the proposed algorithms is detailed below:

E.1 Frequentist Computational Complexity

Since a forward pass through the model consists of exactly L matrix multiplications, we define C_1 as the proportionality constant associated with each matrix multiplication. Similarly, C_2 denotes the constant for the backward pass, and C_3 corresponds to the computational cost of LU decomposition, as used in PyTorch's `torch.inverse()` function. For the RING and RENG algorithms, let S denote the skip frequency for computing the inverse Fisher matrix, and K represent the number of iterations required for Newton's method to approximate the inverse.

One iteration of the vanilla natural gradient descent method [Martens and Grosse, 2015] involves the following computational steps: a forward pass with complexity $(C_1 L\omega^2 m)$, a backward pass costing $(C_2 L\omega^2 m)$, and an output sampling step with cost $\mathcal{O}(1)$. An additional backward pass is required to compute the error for each layer, incurring a further cost of $(C_2 L\omega^2 m)$. Moreover, three matrix multiplications per layer are performed to compute the activations, error matrices, and batch gradients, resulting in a cost of $(3C_1 L\omega^2 m)$. Each layer also requires two matrix inversions, with a total cost of $(2C_3 L\omega^3)$ and two additional matrix multiplications for computing the final weight update, contributing $(2C_1 L\omega^2)$ to the overall complexity.

For the RING and RENG algorithms, the inverse Fisher matrix is computed once every S steps, resulting in a total of $\frac{T}{S}$ exact computations throughout the training process. During the remaining $T - \frac{T}{S}$ iterations, an approximate inverse is obtained using the *Lazy Fisher* technique, as described in Equation 11. In RENG, for each of the $\frac{T}{S}$ iterations involving exact inverse computation, the following operations are performed: One forward pass with complexity $(C_1 L\omega^2 m)$; two backward passes (double backpropagation) costing $(2C_2 L\omega^2 m)$; one output sampling step with cost $(\mathcal{O}(1))$; an additional backward pass to compute layer-wise error: $C_2 L\omega^2 m$; three matrix multiplications per layer to compute activations, error matrices, and batch gradients: $(3C_1 L\omega^2 m)$; two approximate matrix inversions per layer using Newton's method: $(2\psi C_1 L\omega^2 K)$; and two final matrix multiplications per layer for the weight update: $(2C_1 L\omega^2)$. Here, $\psi \in \{2, 3, 4\}$ denotes the number of matrix multiplications required per iteration of Newton's method, depending on the chosen approximation: quadratic Equation 17, cubic Equation 18, or quartic Equation 19:

Quadratic Approximation:

$$\mathbf{X}_{k+1} = \mathbf{X}_k(2\mathbf{I} - \mathbf{A}\mathbf{X}_k), \tag{17}$$

Cubic Approximation:

$$\mathbf{X}_{k+1} = \mathbf{X}_k(3\mathbf{I} - 3\mathbf{A}\mathbf{X}_k + (\mathbf{A}\mathbf{X}_k)^2), \tag{18}$$

Quartic Approximation:

$$\mathbf{X}_{k+1} = \mathbf{X}_k(4\mathbf{I} - \mathbf{A}\mathbf{X}_k(6\mathbf{I} - \mathbf{A}\mathbf{X}_k(4\mathbf{I} - \mathbf{A}\mathbf{X}_k))). \tag{19}$$

Thus, the total time complexity of vanilla natural gradient descent over T iterations is given by $T \times (C_1 L\omega^2 m + C_2 L\omega^2 m + C_2 L\omega^2 m + 3C_1 L\omega^2 m + 2C_3 L\omega^3 + 2C_1 L\omega^2)$ which is of the order $\mathcal{O}(T(CL\omega^2 m + C' L\omega^3))$.

For RENG, the total complexity is given by: $\frac{T}{S} \times (C_1 L\omega^2 m + 2C_2 L\omega^2 m + C_2 L\omega^2 m + 3C_1 L\omega^2 m + 2\psi C_1 L\omega^2 K + 2C_1 L\omega^2) + \left(T - \frac{T}{S}\right) \times (C_1 L\omega^2 m + 2C_1 L\omega^2 + 2C_1 L\omega^2)$ which is of the order $\mathcal{O}\left(\frac{T}{S}(C_1 L\omega^2 m + C_1' L\omega^2 K)\right)$.

For RING, the total complexity is $\frac{T}{S} \times (C_1 L\omega^2 m + C_2 L\omega^2 m + C_2 L\omega^2 m + 3C_1 L\omega^2 m + 2\psi C_1 L\omega^2 K + 2C_1 L\omega^2) + \left(T - \frac{T}{S}\right) \times (C_1 L\omega^2 m + 2C_1 L\omega^2 + 2C_1 L\omega^2)$ which is of the order $\mathcal{O}\left(\frac{T}{S}(C_2 L\omega^2 m + C_2' L\omega^2 K)\right)$.

Compared to vanilla natural gradient descent, our proposed methods achieve significant computational savings. By skipping the full Fisher inverse computation every iteration, the leading factor of T in the complexity is effectively reduced to $\frac{T}{S}$. Additionally, replacing the costly matrix inversion (with complexity ω^3) with Newton's

iterative method reduces the cost to $\omega^2 K$, where $K << \omega$. In practice, this iterative approach scales efficiently for large matrices, especially under low-precision arithmetic (e.g., `bFloat16`). Furthermore, modern GPU architectures enable highly parallel matrix multiplications, with row- and column-wise operations executed concurrently, which makes Newton's method substantially more scalable than traditional LU decomposition-based inversion.

E.2 Bayesian Computational Complexity

We define C_1 as the proportionality constant associated with the cost of matrix multiplication, C_2 for Jacobian computation, and C_3 for inverting the observation noise covariance matrix. We analyze the computational complexity of both the standard Kalman filter and our proposed Jacobian-regularized Kalman algorithm with diagonal approximation, as introduced by [Chang et al., 2022]. Our empirical observations also demonstrate that throughout the training process, the covariance matrix asymptotically approaches a (block-)diagonal configuration: $\mathbb{E}_{p^*(\mathbf{x},\mathbf{y})}[\boldsymbol{\sigma}] = \text{diag}(\boldsymbol{\Sigma})$. To implement the diagonal approximation in the regularized Kalman algorithm, we use the following update step:

$$\tilde{\mathbf{K}}_k = \boldsymbol{\sigma}_{k|k-1} \bullet \mathbf{H}_k^\top \left(\mathbf{H}_k(\boldsymbol{\sigma}_{k|k-1} \bullet \mathbf{H}_k^\top) + \mathbf{R}_k(\mathbf{I} + \rho\mathbf{R}_k)^{-1}\right)^{-1} \tag{20a}$$

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k|k-1} + \tilde{\mathbf{K}}_k(\mathbf{y}_k - h(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})) \tag{20b}$$

$$(\boldsymbol{\sigma}_k)^i = \left(\boldsymbol{\sigma}_{k|k-1}\right)^i - (\mathbf{K}_k)^i_j \, (\mathbf{H}_k)^j_i \left(\boldsymbol{\sigma}_{k|k-1}\right)^i \tag{20c}$$

where the symbol $\bullet$ represents the transposed Khatri–Rao product, which is essentially the row-by-row Kronecker product of the vector $\boldsymbol{\sigma}_{k|k-1}$ and matrix $\mathbf{H}_k^\top$. The Equation 20c represents the diagonal update of the covariance matrix, expressed with Einstein notation. Note that the operations used in Equation 20a, and Equation 20c can be computed efficiently. For example, in PyTorch, they can be seamlessly implemented using the $*$, and **einsum**() operators, streamlining the computational process. The analysis is structured around the three main steps of the algorithm: Prediction, Pre-Update, and Update for both standard and our Kalman algorithms.

The computational complexity of both our algorithm and the standard Kalman filter for the prediction step is $\mathcal{O}(1)$.

The pre-update step for Jacobian computation incurs a cost of $C_2 n d_o$. Estimating the observation noise covariance involves a cost of $C_1(d_o^2 + nd_o^2 + n^2 d_o)$, while updating the observation noise covariance via exponential averaging has negligible cost, i.e., $\mathcal{O}(1)$. Computing the Kalman gain requires two matrix multiplications and one matrix inversion (of size $d_o \times d_o$), with a total cost of $(C_1 n^2 d_o + C_3 d_o^3 + C_1 n d_o^2)$. The parameter update, which involves multiplying the Kalman gain matrix by the output error, costs $C_1 n d_o$. The final and most computationally intensive step is the update of the posterior covariance matrix $\boldsymbol{\Sigma}_k$, which requires constructing a full $n \times n$ matrix. This step contributes $(C_1 n^2 d_o + C_1 n^3)$ to the overall cost. Therefore, the total time complexity across T iterations is: $T \times (C_2 n d_o + C_1(d_o^2 + nd_o^2 + n^2 d_o) + C_1 n^2 d_o + C_3 d_o^3 + C_1 n d_o^2 + C_1 n d_o + C_1 n^2 d_o + C_1 n^3)$. Since $d_o \ll n$, the dominant terms are $C_1 n^2 d_o$ and $C_1 n^3$, which yields an overall computational complexity of order $\mathcal{O}(C_1(n^2 d_o + n^3))$.

In the case where we employ the diagonal approximation of the covariance matrix, the use of the element-wise Khatri–Rao product replaces matrix–matrix multiplications with more efficient matrix–vector operations. Consequently, the Kalman gain computation now incurs a cost of $2(C_1 n + C_1 n d_o^2 + C_3 d_o^3)$, as it includes an additional matrix inversion due to the newly introduced regularization. The parameter update step maintains the same cost as in the full covariance case. For the covariance update, which is expressed using Einstein summation notation, each element of the diagonal covariance requires a vector–vector multiplication. Thus, updating all n diagonal elements costs $C_1 n^2$. As a result, the total computational complexity per iteration becomes $\mathcal{O}(C_1(n^2 + nd_o^2))$, which is one order of magnitude lower than that of the standard Kalman update.

F Hyperparameters

The hyperparameters used in vision and language experiments are reported in Table 4 and Table 5, respectively.

Table 4: Hyperparameters used for fine-tuning ViT-B16 on various image classification datasets, including CIFAR-10, CIFAR-100, Oxford-IIIT Pet, Food-101, and ImageNet-100. The table lists the hyperparameter settings for our proposed optimization algorithms (RING, RENG, and R-Kalman), as well as for the baseline methods AdamW, Sophia, and NGD.

Algorithm	Hyperparameters	CIFAR-10	CIFAR-100	Oxford-IIIT Pet	Food-101	ImageNet-100
AdamW	Batch Size			16		
	Num. Epochs	10	5	10	8	8
	Learning Rate ($\times 10^{-4}$)	0.6-0.8	0.4-0.6	0.6-0.8	0.4-0.8	0.6-0.8
	Warmup Steps	200	500	500	500	500
	LoRA Config.			$r_q = r_v = 4$		
	LoRA α			4		
Sophia	Batch Size			16		
	Num. Epochs	10	5	10	8	8
	Learning Rate ($\times 10^{-4}$)	0.6-0.8	0.4-0.6	0.6-0.8	0.4-0.8	0.6-0.8
	Warmup Steps	200	500	500	500	500
	β_1, β_2			0.975, 0.99		
	LoRA Config.			$r_q = r_v = 4$		
	LoRA α			4		
NGD	Batch Size			100		
	Num. Epochs	3	2	3	2	2
	Learning Rate			1.0		
	ρ ($\times 10^{-4}$)	1.0	1	2	2	0.8
	ϕ			0.995		
	LoRA Config.			$r_q = r_v = 4$		
	LoRA α			4		
RING	Batch Size			100		
	Num. Epochs	4	1	2	2	2
	Learning Rate			1.0		
	ρ ($\times 10^{-4}$)	10	10	4	5	8
	ϕ			0.99-0.995		
	Skip freq. (S)	8	8	8	8	8
	LoRA Config.			$r_q = r_v = 4$		
	LoRA α			4		
RENG	Batch Size			100		
	Num. Epochs	4	2	3	2	2
	Learning Rate			1.0		
	Grad Reg.	0.05	0.05	0.01	0.01	0.01
	ρ ($\times 10^{-4}$)	40	10	8	2	8
	ϕ			0.99-0.995		
	Skip freq (S)	8	8	8	8	8
	LoRA Config.			$r_q = r_v = 4$		
	LoRA α			4		
R-Kalman	Batch Size			1		
	Num. Epochs	1	1	2	1	1
	β	0.96	0.98	0.96	0.98	0.98
	$\boldsymbol{\sigma}_0$	0.2	0.2	0.2	0.1	0.1
	LoRA Config.			$r_q = r_v = 4$		
	LoRA α			4		

Discount factor ϕ is used in the Levenberg-Marquardt scheme for updating the damping coefficient [Martens and Grosse, 2015].

Table 5: Hyperparameters used for fine-tuning RoBERTa-base on the GLUE language benchmarks, including MNLI-mm, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE. The table reports the hyperparameter settings for our proposed optimization algorithms (RING, RENG, and R-Kalman) as well as for the baseline methods AdamW, Sophia, and NGD.

Algorithm	**Hyperparameters**	**MNLI-mm**	**QQP**	**QNLI**	**SST-2**	**CoLA**	**STS-B**	**MRPC**	**RTE**
AdamW	Batch Size	64							
	Num. Epochs	8	8	8	10	25	40	40	25
	Learning Rate ($\times 10^{-4}$)	0.4-0.8	0.6-0.8	1.0-2.0	0.8-1.2	0.4-0.6	0.4-1.0	0.8-1.0	0.8-1.0
	Warmup Steps	500	500	500	500	200	100	200	100
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max. Seq. Len.	512							
Sophia	Batch Size	64							
	Num. Epochs	8	8	8	10	25	40	40	25
	Learning Rate ($\times 10^{-4}$)	0.4-0.8	0.6-0.8	2.0-4.0	0.8-1.2	0.6-0.8	0.4-1.0	0.8-1.0	0.8-1.0
	β_1, β_2	$0.965, 0.98$							
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max. Seq. Len.	512							
NGD	Batch Size	64							
	Num. Epochs	1	1	2	3	6	10	5	5
	Learning Rate	0.99-1.0							
	ρ ($\times 10^{-4}$)	1.0	1	2	2	0.8	1.0	0.8	0.4
	ϕ	0.99-0.995							
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max. Seq. Len.	512							
RING	Batch Size	64	64	64	32	64	100	32	32
	Num. Epochs	1	1	2	5	6	10	5	5
	Learning Rate	0.9-1.0							
	ρ ($\times 10^{-4}$)	10	10	4	5	8	40	9	10
	ϕ	0.999-0.9999							
	Skip freq (S)	8	8	8	4	4	4	4	4
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max. Seq. Len.	512							
RENG	Batch Size	64	64	64	256	64	100	32	64
	Num. Epochs	1	1	2	5	6	10	5	5
	Learning Rate	0.9-1.0							
	Grad. regu.	0.005	0.01	0.01	0.001	0.1	0.001	0.001	0.001
	ρ ($\times 10^{-4}$)	40	10	8	2	8	40	8	10
	ϕ	0.999-0.9999							
	Skip freq (S)	8	8	8	4	8	4	4	4
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max. Seq. Len.	512							
R-Kalman	Batch Size	1							
	Num. Epochs	1	1	1	2	2	2	4	4
	β	0.97	0.97	0.97	0.96	0.98	0.98	0.98	0.98
	$\boldsymbol{\sigma}_0$	0.1	0.1	0.1	0.1	0.04	0.05	0.05	0.05
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max. Seq. Len.	512							

Discount factor ϕ is used in the Levenberg-Marquardt scheme for updating the damping coefficient [Martens and Grosse, 2015].

G Additional Experiments

We perform extensive empirical evaluations across both image classification and GLUE language benchmarks. Detailed results for vision and language tasks are presented in Figure 3 and Figure 4, respectively. The vision datasets include CIFAR-10/100, Oxford-IIIT Pet, Food-101, and ImageNet-100, while the GLUE benchmarks cover MNLI-mm, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE subsets. For image classification, we fine-tune a ViT-B16 model pre-trained on ImageNet; for language tasks, we fine-tune RoBERTa-base, pre-trained on large-scale text corpora. The plots show the validation accuracy of our proposed algorithms (RING, RENG, and R-Kalman) compared to AdamW, Sophia, and NGD as a function of training iterations.

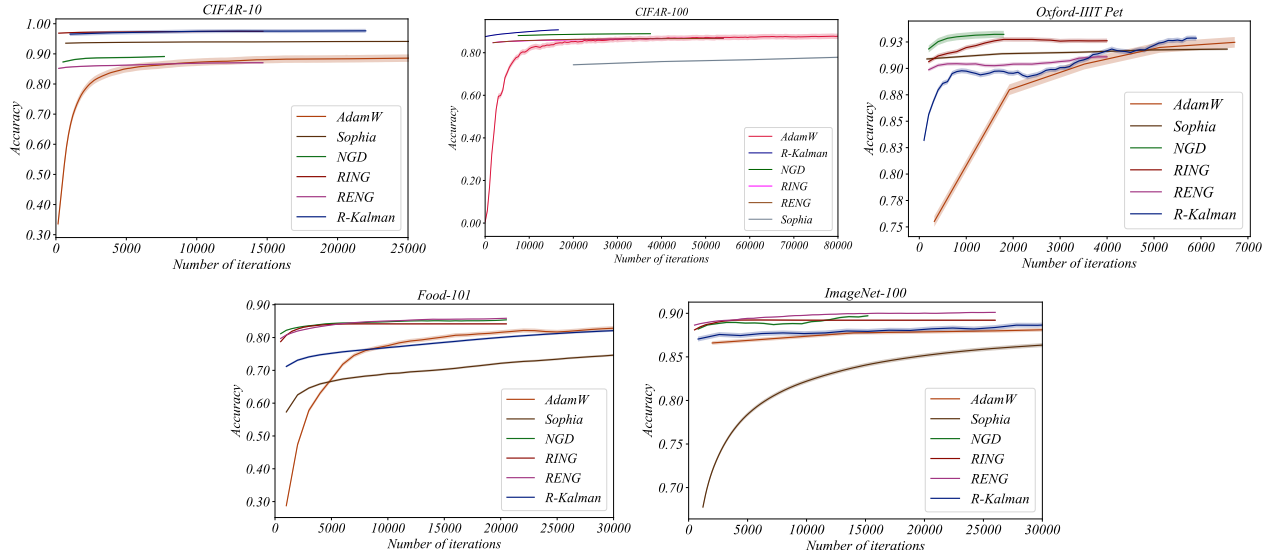

Figure 3: Comprehensive results on the image classification benchmarks, including CIFAR-10/100, Oxford-IIIT Pet, Food-101, and ImageNet-100. The plots show the validation accuracy of our proposed algorithms (RING, RENG, and R-Kalman) compared to AdamW, Sophia, and NGD when fine-tuning ViT-B16. Validation accuracy is plotted as a function of training iterations.

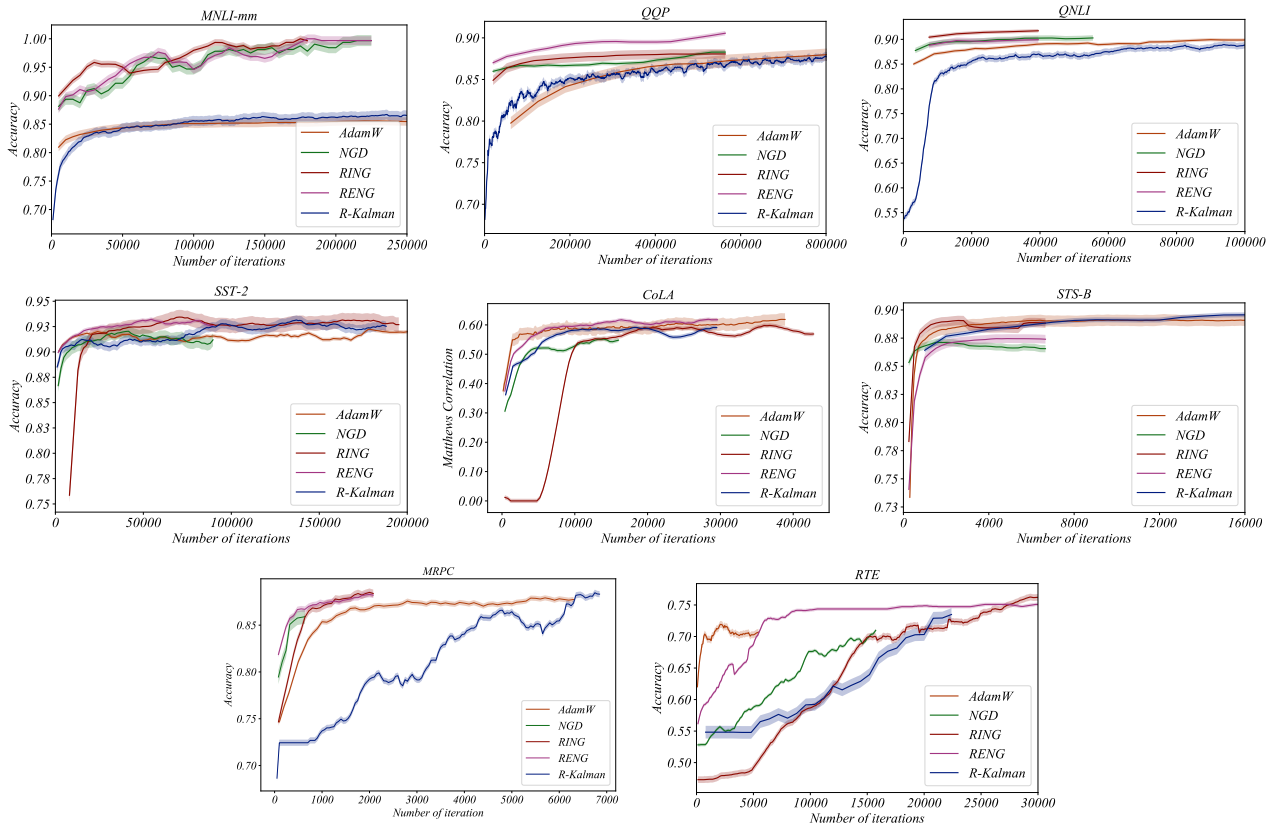

Figure 4: Comprehensive results on the GLUE benchmark, including MNLI-mm, QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE. The plots show the validation accuracy of our proposed algorithms (RING, RENG, and R-Kalman) compared to AdamW and NGD when fine-tuning RoBERTa-base. Validation accuracy is plotted as a function of training iterations.

H Ablation Study

Sensitivity to damping coefficient and gradient regularization coefficient. We demonstrate that, within a certain range, varying the gradient regularization coefficient in RING and RENG, as well as the damping coefficient in RENG, leads to convergence toward near-optimal solutions. However, our analysis also reveals a lower bound on these hyperparameters, which may depend on the dataset. It is important to note that, due to our use of the Lazy-Fisher technique, inaccurate estimates of the Fisher information matrix can lead to instability. In particular, setting the damping coefficient too low. For more details, see Figure 5 for RING algorithm, and Figure 6 for RENG.

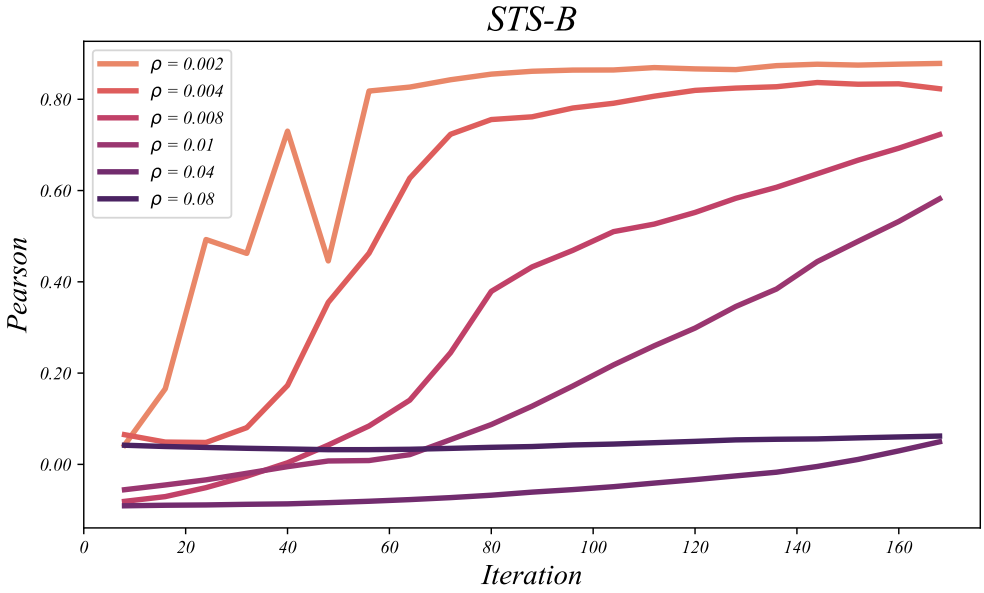

Figure 5: Sensitivity analysis of dampening coefficient for RING.

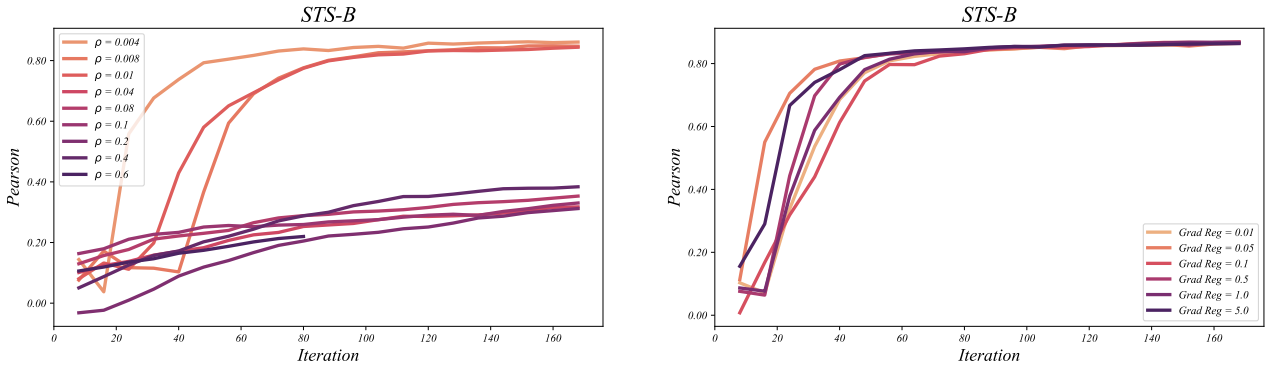

Figure 6: Sensitivity analysis of dampening coefficient and gradient regularization coefficient for RENG.

Skip Frequency in Lazy-Fisher. We systematically investigated the impact of update frequency in Lazy-Fisher on runtime and performance. The results of these experiments are presented in the Table 6, and Table 7. The first table presents results on the STS-B dataset, reporting values at iteration 20. The second table shows results on the MNLI-mm dataset, with values reported at iteration 200. The empirical results demonstrate that

Table 6: Performance and runtime on the STS-B dataset under different update frequencies of Lazy-Fisher.

Algorithm	Skip Freq = 2	Skip Freq = 4	Skip Freq = 6	Skip Freq = 8	Skip Freq = 10
RING	Pearson: 35.85 Runtime: $2:46s$	Pearson: 79.54 Runtime: $1:40s$	Pearson: 65.23 Runtime: $1:20s$	Pearson: 58.57 Runtime: $1:07s$	Pearson: 69.24 Runtime: $1:00s$
RENG	Pearson: 40.23 Runtime: $1:32s$	Pearson: 52.92 Runtime: $0:51s$	Pearson: 48.12 Runtime: $0:43s$	Pearson: 40.12 Runtime: $0:36s$	Pearson: 33.15 Runtime: $0:30s$

Table 7: Performance and runtime on the MNLI-mm dataset under different update frequencies of Lazy-Fisher.

Algorithm	Skip Freq = 4	Skip Freq = 8	Skip Freq = 16	Skip Freq = 32
RING	Accuracy: 48.43% Runtime: $31:27s$	Accuracy: 46.87% Runtime: $21:10s$	Accuracy: 37.50% Runtime: $16:45s$	Accuracy: 39.08% Runtime: $13:11s$
RENG	Accuracy: 51.56% Runtime: $8:00s$	Accuracy: 51.56% Runtime: $5:15s$	Accuracy: 50.00% Runtime: $4:00s$	Accuracy: 46.87% Runtime: $3:15s$

a skip frequency of 4 for small datasets and 8 for large datasets provides an effective balance between runtime efficiency and performance.

Initial Prior $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. We find that selecting an appropriate initial prior $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ is crucial for ensuring stable convergence of the Kalman algorithm. This selection ideally relies on prior knowledge of the trainable parameters and their associated uncertainties. In the absence of such information, a practical strategy is to adopt widely used initialization schemes such as Xavier [Glorot and Bengio, 2010] or Kaiming [He et al., 2015] for training scenarios. For fine-tuning settings, pre-trained parameter values may be used, or alternatively, standard initialization methods employed in parameter-efficient fine-tuning approaches such as LoRA [Hu et al., 2021].

As for the covariance matrix $\boldsymbol{\Sigma}_0$, it can be initialized with arbitrary positive values, as long as they are neither too small nor excessively large. A common and effective choice is $\boldsymbol{\Sigma}_0 = \sigma_0 \mathbf{I}$, where $0 < \sigma_0 \ll 1$. Importantly, the performance of our algorithm is not highly sensitive to the specific choice of initial values. Regardless of the initialization method, the algorithm consistently converges to a stable performance level over time. Figure 7 shows the sensitivity of R-Kalman algorithm to different values of σ_0 in $0 < \sigma_0 \ll 1$.

Sensitivity to β in $\mathbf{R}_k$ Estimation. We adopt the method proposed in Abdi et al. [2024] to estimate the observation noise covariance matrix $\mathbf{R}_k$:

$$\mathbf{R}_k = \beta \mathbf{R}_{k-1} + (1-\beta)\hat{\mathbf{R}}_k,$$
$$\hat{\mathbf{R}}_k = \Big(\mathbf{y}_k - h(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})\Big)\Big(\mathbf{y}_k - h(\mathbf{x}_k, \boldsymbol{\mu}_{k|k-1})\Big)^\top + \mathbf{H}_k \boldsymbol{\Sigma}_{k|k-1} \mathbf{H}_k^\top,$$

where $\beta \in (0,1)$ is forgetting factor. To evaluate the influence of the hyperparameter β on the performance of the R-Kalman algorithm, we performed a sensitivity analysis by varying β across a range of values. The results demonstrate that setting β too high or too low significantly degrades the algorithm's performance. In contrast, optimal performance is consistently achieved when β lies within the range $0.94 \leq \beta \leq 0.99$. Moreover, within this interval, the performance is relatively insensitive to the precise choice of β. For a detailed comparison, refer to Figure 8.

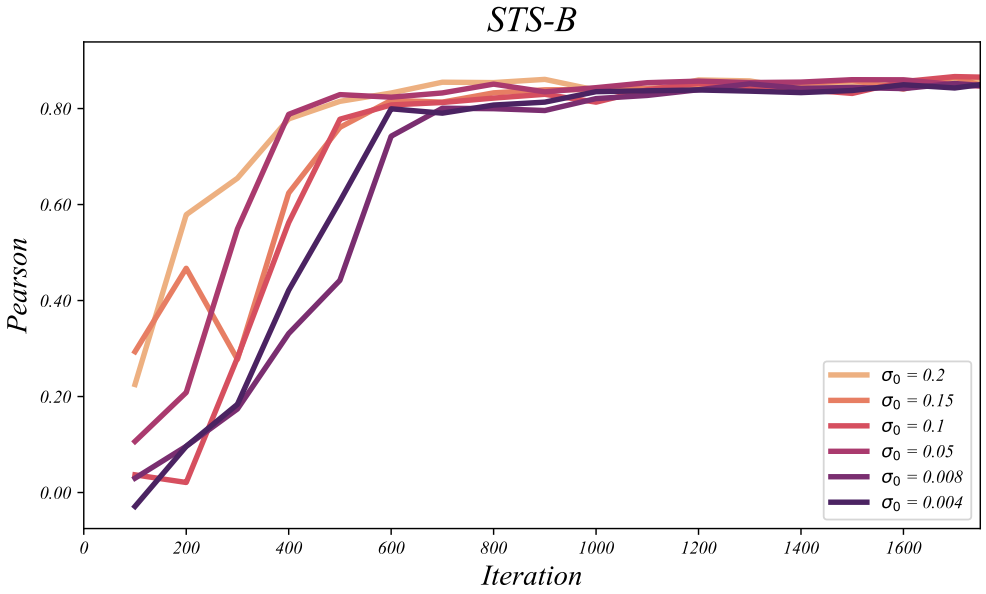

Figure 7: Sensitivity analysis of the R-Kalman algorithm with respect to the initial values of σ_0. The plot indicates that, as long as σ_0 lies within the $0 < \sigma_0 \ll 1$ range, the algorithm converges to a consistent optimal solution regardless of its specific initial value.

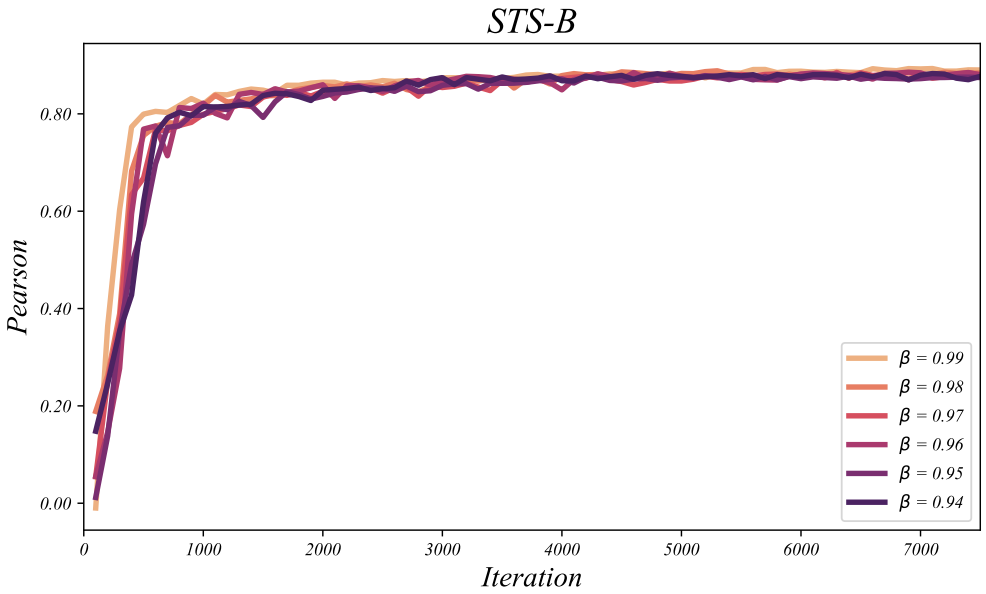

Figure 8: Sensitivity analysis of R-Kalman algorithm to β values. The plot shows that, as long as β lies within the range $0.94 \leq \beta \leq 0.99$, the algorithm converges to a consistent optimal solution.