Supporting Material for Author Response for: An Improved Empirical Fisher Approximation for Natural Gradient Descent

1. Introduction

This document is provided to support the author response to the reviews for paper "An Improved Empirical Fisher Approximation for Natural Gradient Descent" (referred to as main paper here). The intended readers of this document are the reviewers of the main paper. Key equations and data that are referenced will be presented again to minimise back-and-forth between the main paper and this document. Readers are not required to read the entire document. However, reviewers could use this document if they are interested in the further details for claims made in the author response. Clear references to specific parts of this document are provided for each claim in the author response.

Sections 2-6 provide additional discussion that address concerns raised by the reviewers. The contents of each section are as follows:

1. Section 2 provides a detailed analysis of NGD, EF and iEF method for the least-squares regression problem. It is shown that the iEF matrix is equivalent to the Fisher matrix for this type of problem, which gives a good justification of the iEF method. Also, additional visualisation is provided for a least-squares linear regression problem to better emphasise the superiority of iEF over EF method.

2. Section 3 gives a convergence analysis of the proposed iEF method. Under the practical assumption of a strongly convex per-sample objective function, the iEF method is shown to guarantee linear convergence to the global minimum. Note that the strong convexity is assumed for per-sample objective function, and the total loss function can still be arbitrarily non-convex.

3. Section 4 discusses the interaction between damping and the EF update. It provides motivation for the deliberate use of the small damping factor in the main paper. Analysis of existing and additional experiments shows that the small damping factor is not the cause of the poor performance of EF, and that the iEF method is a superior solution to the issue of EF than increasing the damping factor.

4. Section 5 provides a clarification since some reviewers thought of iEF as primarily an optimiser. However, in fact, the main purpose of the iEF method proposed in this paper is to serve as a way to improve existing empircal NGD optimisers. The details of using iEF as an improvement guideline are then explained.

5. Section 6 presents additional experimental results for new model structures and an additional dataset (including LoRA + T5 applied to 4 GLUE tasks and LoRA + ViT applied to CIFAR100). The same type of analysis is given for the additional experiments as for experiments in the main paper. It is shown that the conclusions drawn in the main paper generalise well to these new model structure and dataset combinations.

6. Section 7 discusses the time and memory complexity of the exact iEF/EF method. Per-batch wall-clock time for the prompt-tuning setup shows that the iEF/EF update generation time is negligible as compared to time cost of gradient accumulation.

7. Section 8 introduces an additional baseline method "true Fisher with sampling", which is another commonly used (but more expensive) NGD approximation method (Martens & Grosse, 2015). The approximation quality among the iEF, EF and true Fisher with sampling methods is compared across all 12 experimental training setups. The results show that iEF is consistently superior to "true Fisher with sampling" soon after training starts.

2. Analysis of the Least-Squares Regression Problem

In this section, additional theoretical analysis is provided for a least-squares regression problem, where exact iEF, EF and NGD have methods that have simpler forms and are easier to analyse. It is shown that the iEF method and NGD are equivalent in this setup. Additional visualisation is also presented for a 2D least-squares linear regression problem, and the effectiveness of iEF updates is demonstrated, as well as further confirming the pathological behaviour of EF updates.

2.1. Preliminaries

The least-squares regression problem is widely-used in classical machine learning, and is also commonly used in the theoretical analysis of NGD related methods. It has been used in both (Kunstner et al., 2019; Thomas et al., 2019) to demonstrate the connection between the Fisher matrix and the Hessian, as well as the limitation of the EF approximation.

The convergence and generalisation properties of the NGD method is also analysed in (Zhang et al., 2019) for this problem. In this section, the least-squares regression problem is properly defined and introduced. We slightly abuse the notation here. The notation used in the main paper, including the target model $f_{\boldsymbol{\theta}}(\cdot)$, target loss $\mathcal{L}(\cdot)$, per-sample loss l_n, training samples $(\boldsymbol{x}_n, y_n)$, various Fisher matrices ($\mathbf{F}$, $\tilde{\mathbf{F}}$, $\tilde{\mathbf{F}}^{\star}$) and scaling vector $\boldsymbol{s}_{\text{iEF}}$ are reused here but with slightly different definitions.

Consider a target regression model $f_{\boldsymbol{\theta}}(\cdot) \in \mathbb{R}^P \rightarrow \mathbb{R}$, where $\boldsymbol{\theta} \in \mathbb{R}^P$ is the trainable parameters of size P. Given N *i.i.d.* training samples $(\boldsymbol{x}_n, y_n)_{n=1}^N$ (where $\boldsymbol{x}_n$ is the input feature vector and $y_n \in \mathbb{R}$ is the scalar label), the output of the model for the n-th sample is $z_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$ where $z_n \in \mathbb{R}$ is a scalar. The following accumulated mean-square-error loss is to be minimised

$$\mathcal{L}(\boldsymbol{\theta}) = \sum\nolimits_n \frac{1}{2}\left[f_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right]^2 = \sum\nolimits_n \frac{1}{2}(z_n - y_n)^2 = \sum\nolimits_n l_n, \tag{1}$$

where $l_n = \frac{1}{2}(z_n - y_n)^2$ is the loss for the n-th sample. The gradient of the n-th sample is therefore expressed as follows

$$\nabla_{\boldsymbol{\theta}} l_n = (z_n - y_n)\nabla_{\boldsymbol{\theta}} z_n. \tag{2}$$

2.2. Equivalence of IEF and Fisher Matrice

Following the definition in the main paper, the Fisher matrix, iEF matrix and the EF matrix can be obtained for the least-squares regression problem. The exact Fisher matrix is computed as follows

$$\mathbf{F} = \sum\nolimits_n \nabla_{\boldsymbol{\theta}} z_n \nabla^2_{z_n} l_n \nabla_{\boldsymbol{\theta}} z_n^{\top} = \sum\nolimits_n \nabla_{\boldsymbol{\theta}} z_n \nabla_{\boldsymbol{\theta}} z_n^{\top}. \tag{3}$$

Note that the Fisher matrix is equivalent to the Hessian when a linear regression model is concerned (Thomas et al., 2019). The EF matrix is computed as follows

$$\tilde{\mathbf{F}} = \sum\nolimits_n \nabla_{\boldsymbol{\theta}} l_n \nabla_{\boldsymbol{\theta}} l_n^{\top} = \sum\nolimits_n (z_n - y_n)^2 \nabla_{\boldsymbol{\theta}} z_n \nabla_{\boldsymbol{\theta}} z_n^{\top}. \tag{4}$$

Recall the definition for the iEF matrix as follows

$$\tilde{\mathbf{F}}^{\star} = \nabla_{\boldsymbol{\theta}} \boldsymbol{l}^{\top} \text{diag}(\boldsymbol{s}_{\text{iEF}})^{-1} \nabla_{\boldsymbol{\theta}} \boldsymbol{l} \tag{5}$$

where $\text{diag}(\boldsymbol{s}_{\text{iEF}}) \in \mathbb{R}^{N \times N}$ is a diagonal matrix with its diagonal set to scaling vector $\boldsymbol{s}_{\text{iEF}}$. And the scaling vector can be computed as follows

$$\boldsymbol{s}_{\text{iEF}} = \begin{bmatrix} \|\nabla_{\boldsymbol{z}_1} l_1\|_2^2 & \cdots & \|\nabla_{\boldsymbol{z}_n} l_N\|_2^2 \end{bmatrix} = \begin{bmatrix} (z_1 - y_1)^2 & \cdots & (z_N - y_N)^2 \end{bmatrix}. \tag{6}$$

Plugging the scaling vector $\boldsymbol{x}_{\text{iEF}}$ into Eqn.5, the iEF matrix is therefore computed as follows

$$\tilde{\mathbf{F}}^{\star} = \sum\nolimits_n \nabla_{\boldsymbol{\theta}} z_n \nabla_{\boldsymbol{\theta}} z_n^{\top}. \tag{7}$$

By comparing the three matrices defined in Eqn. 3, 4, 7, the following conclusions can be drawn: **1)** The iEF matrix $\tilde{\mathbf{F}}^{\star}$ and the Fisher matrix $\mathbf{F}$ have an identical form. This indicates that iEF method and NGD are equivalent in the least-square regression problem. **2)** The EF matrix $\tilde{\mathbf{F}}$ is different from the Fisher matrix by a per-sample scaling factor $(z_n - y_n)^2$, which causes the inversely-scaled projection issue during EF update generation.

2.3. Visual Comparison of Different Update Methods

The SGD, EF, NGD and iEF updates are visualised and compared using a 2D least-squares linear regression problem, as is shown in Fig. 1 (Kunstner et al., 2019). It is worth pointing out that the NGD and iEF updates (which are identical) point towards the global minimum regardless of the model parameter position. This is because for a least-squares linear regression setup, the iEF matrix and the Fisher matrix are both equivalent to the Hessian (Thomas et al., 2019), making NGD and iEF method equivalent to Newton's method. On the contrary, the EF updates show a strange scaling and direction, which leads to ill-behaving training trajectories. Moreover, the direction of EF updates are sometimes nearly perpendicular to the NGD update (which means poor curvature behaviour) and cause the training to be ineffective and take extra detours. This makes training with EF even less effective than SGD. All of these observations are aligned with the toy example presented in the main paper, as well as the empirical observations made in the main paper.

2.4. Conclusions

In conclusion, for the least-square regression problem, the iEF method and NGD are equivalent, which advocates the effectiveness of the proposed iEF approximation for NGD. The visual example for the 2D least-squares linear regression problem demonstrates the highly distorted EF update field on the loss landscape. It is aligned with the analysis in the main paper, and further confirms the detrimental effect of using the EF method in optimisation without addressing the inversely-scaled projection issue.

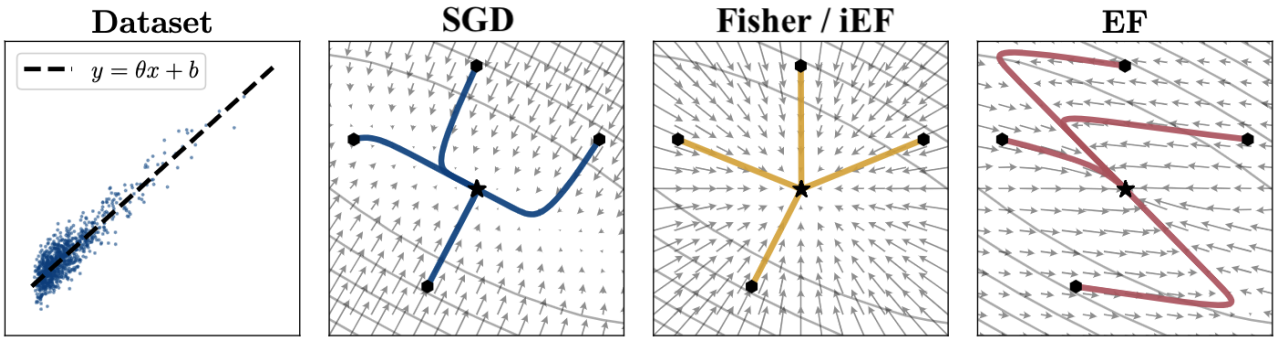


Figure 1. This figure is modified based on (Kunstner et al., 2019), which demonstrates the difference in EF, iEF, NGD and SGD updates in a least-squares linear regression problem. The 2D data points used are presented in the left-most plot. The different updates (small arrows) are plotted for a grid of parameter position on the loss landscape on the following three plots. Updates are generated using all data points. The second plot shows the SGD updates. The third plot gives the NGD (Fisher preconditioned) updates and iEF updates (which are equivalent). The forth figure shows the EF updates. 4 sampled training trajectories following different types of updates are also presented.

3. Convergence Analysis

Although the experimental results in the main paper has demonstrated the empirical effectiveness of iEF as an optimiser, it is still useful to inspect the convergence behaviour of the iEF method. In this section, additional convergence analysis is provided for a non-stochastic version of the iEF method (which uses all samples to generate each update). It can be shown that iEF method guarantees linear convergence to the global minimum (global linear convergence) for any strongly convex (per-sample) objective function.

3.1. Global Linear Convergence in the Least-Squares Regression Problem

There has been previous work (Zhang et al., 2019) analysing the convergence behaviour of NGD method for a least-squares regression problem. Due to the equivalence of the iEF method and NGD in this setup, the convergence analysis results is also applicable to the iEF method. It is shown that the iEF method guarantees global linear convergence for a regression model and a strongly convex loss function (including least-squares loss). Unfortunately, although insightful, these convergence results are only applicable to the regression problem where the target model has a scalar output. It does not provide enough information on the behaviour of iEF method in practice, where many target models have multi-dimensional output (such as the classification model analysed in the main paper). In the following section, a more generally applicable convergence guarantee is provided.

3.2. Global Linear Convergence for a Non-Regression Model

In this section, the global linear convergence of iEF method is generalised to target model with multi-dimensional output (non-regression model) and strongly convex (per-sample) objective function. There are a few important notes for this novel convergence analysis: **1)** The target model is no longer limited to a regression model which makes the analysis more practically useful. **2)** The assumption of strong convexity is for per-sample loss function at model output level. Both the total loss $\mathcal{L}(\boldsymbol{\theta})$, and the per-sample loss $l_n(\boldsymbol{\theta})$ can have arbitrarily non-convex landscape on the parameter space. It is therefore non-trivial for an optimisation method to guarantee convergence to global minimum (global convergence).

The proof follows the continuous-time analysis framework in (Zhang et al., 2019) and (Ang, 2022), and is shown as follows. The problem setup is first defined. Consider a target model $f_{\boldsymbol{\theta}}(\cdot) \in \mathbb{R}^P \to \mathbb{R}^C$, where C is the dimension of model output space and $\boldsymbol{\theta} \in \mathbb{R}^P$ is the trainable parameters of size P. Given N *i.i.d.* training samples $(\boldsymbol{x}_n, y_n)_{n=1}^N$ (where $\boldsymbol{x}_n$ is the

input feature vector and y_n is a label of arbitrary form), the output of the model for the n-th sample is $\boldsymbol{z}_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$ where $z_n \in \mathbb{R}^C$. An objective loss function $\mathcal{F}_{\text{obj}}(\cdot)$ is used to compute the per-sample loss for the n-th sample $l_n \in \mathbb{R}$ as follows

$$l_n = \mathcal{F}_{\text{obj}}(\boldsymbol{z}_n, y_n) \coloneqq \mathcal{F}_{\text{obj}}(\boldsymbol{z}_n), \tag{8}$$

where the label y_n is omitted for simplicity. Finally, the following accumulated loss is to be minimised

$$\mathcal{L}(\boldsymbol{\theta}) = \sum\nolimits_n l_n. \tag{9}$$

This is a generalised version of the setup used in the main paper because the loss function $\mathcal{F}_{\text{obj}}(\cdot)$ is no longer constrained to be the combination of softmax activation and categorical cross-entropy.

A non-stochastic version of the iEF method is analysed here, which uses all N training samples in each step to generate the update. The proof relies on the following two assumptions.

Assumption 3.1. The gradient covariance matrix (or Gram matrix) $[\nabla_{\boldsymbol{\theta}(t)}\boldsymbol{l}(t)][\nabla_{\boldsymbol{\theta}(t)}\boldsymbol{l}(t)]^\top$ is always full rank.

Assumption 3.2. For the n-th sample, the objective loss function $\mathcal{F}_{\text{obj}}(\boldsymbol{z}_n)$ is assumed to be m-strongly convex on the model output space $\boldsymbol{z}_n$, which then satisfies the following Polyak-Lojasiewicz inequality (Brefeld et al., 2020)

$$\mathcal{F}_{\text{obj}}(\boldsymbol{z}_n) - \mathcal{F}_{\text{obj}}(\boldsymbol{z}_n^\star) \leq \frac{1}{2m}\|\nabla_{\boldsymbol{z}_n}\mathcal{F}_{\text{obj}}(\boldsymbol{z}_n)\|^2, \tag{10}$$

where $\mathcal{F}_{\text{obj}}(\boldsymbol{z}_n^\star)$ is the global minimum of the loss for the n-th sample. For simplicity, the notation $l_n^\star$ is used in place of $\mathcal{F}_{\text{obj}}(\boldsymbol{z}_n^\star)$. The inequality is therefore rewritten as follows

$$l_n - l_n^\star \leq \frac{1}{2m}\|\nabla_{\boldsymbol{z}_n} l_n\|^2. \tag{11}$$

Assumption. 3.1 is a weaker and simplified version of the Condition 1 and 2 in (Zhang et al., 2019). It is reasonable in realistic deep learning setups because the target model is in general over-parameterised with $P \gg N$. Therefore, it is likely that different samples have independent gradient directions. Assumption. 3.2 is quoted from (Ang, 2022), which covers a wide range of loss functions including mean-square-error. Note that under this assumption both per-sample losses and accumulated loss can still have arbitrarily non-convex landscape on the parameter space.

Using Assumption. 3.1, the iEF update at time t can be computed as follows without the damping term

$$\frac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t} = -\nabla_{\boldsymbol{\theta}(t)}\boldsymbol{l}(t)^\top[\nabla_{\boldsymbol{\theta}(t)}\boldsymbol{l}(t)\nabla_{\boldsymbol{\theta}(t)}\boldsymbol{l}(t)^\top]^{-1}\boldsymbol{s}_{\text{iEF}}(t), \tag{12}$$

where $\boldsymbol{s}_{\text{iEF}} = \left[\|\nabla_{\boldsymbol{z}_1(t)}l_1(t)\|_2^2 \quad \|\nabla_{\boldsymbol{z}_2(t)}l_2(t)\|_2^2 \quad \cdots \quad \|\nabla_{\boldsymbol{z}_N(t)}l_N(t)\|_2^2\right]$ is the iEF scaling vector, and $\boldsymbol{l}(t) = \left[l_1(t), l_2(t), \cdots, l_N(t)\right]^\top$ is the vector of per-sample losses. The term (t) denotes the corresponding variable for continuous time t. Consequently, the change of per-sample loss $\boldsymbol{l}$ can be computed as

$$\frac{\mathrm{d}\boldsymbol{l}(t)}{\mathrm{d}t} = \nabla_{\boldsymbol{\theta}(t)}\boldsymbol{l}(t)\frac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t} = -\boldsymbol{s}_{\text{iEF}}. \tag{13}$$

Specifically, for the n-th sample, the loss change is

$$\frac{\mathrm{d}l_n(t)}{\mathrm{d}t} = -\|\nabla_{\boldsymbol{z}_n(t)}l_n(t)\|_2^2 \tag{14}$$

Using the Polyak-Lojasiewicz inequality in Assumption. 3.2, the following inequality can be obtained

$$\frac{\mathrm{d}(l_n(t) - l_n^\star)}{\mathrm{d}t} \leq -2m(l_n(t) - l_n^\star) \Rightarrow \frac{1}{(l_n(t) - l_n^\star)}\frac{\mathrm{d}(l_n(t) - l_n^\star)}{\mathrm{d}t} \leq -2m. \tag{15}$$

Integrating on both sides gives the following

$$l_n(t) - l_n^\star \leq e^{-2mt}(l_n(0) - l_n^\star), \tag{16}$$

which shows that iEF preconditioned gradient flow linearly converges to the global minimum for every sample. This then implies that iEF has global linear convergence guarantee for accumulated loss $\mathcal{L}(\boldsymbol{\theta})$. The global convergence guarantee can be intuitively understood from the definition of the iEF update, as this method would stop (i.e. generate zero-norm update) if and only if gradient for every sample is zero (i.e. all samples have reached their respective global minimum). This means that iEF method does not get stuck in local minimum like the standard gradient descent method.

3.3. Conclusions

In conclusion, it can be shown that for both regression and non-regression target models with a strongly convex per-sample objective function, the iEF method guarantees linear convergence to the global minimum of the accumulated loss. It needs to be stressed again that the strong convexity is assumed for the per-sample objective function, which is a realistic assumption in practice. The accumulated loss and per-sample loss can still have arbitrarily non-convex loss landscape on the parameter space.

4. Discussion on Damping Factor

A notable difference between the EF training setup used in the main paper from other work is the unorthodox small damping factor $\lambda = 1 \times 10^{-7}$. It is well known that the damping factor is an important yet hard-to-tune hyper-parameter for preconditioned optimisers, which plays an important role in improving the conditioning of the target preconditioner. It is suspected that a larger damping could likely improve the training performance of EF. The purpose of this section is to explain the motivation for using such a small damping factor in our experiments. It is then emphasised that the small damping factor does not cause a conditioning issue during the computation of EF updates, and the inversely-scaled projection issue in the main paper is the key issue. Additional experiments are presented to verify these claims.

4.1. Motivation for Using Small Damping Factor

The use of a small damping factor in our experimental setup is a deliberate choice, and the motivation is two-fold.

First, it is likely that the conditioning issue of the covariance matrix in the current setup is minimal. Recall the definition of the EF update

$$\Delta\boldsymbol{\theta}_{\text{EF}} = -\eta(\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}\nabla_{\boldsymbol{\theta}}\boldsymbol{l} + \lambda\boldsymbol{I})^{-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}\mathbf{1}). \tag{17}$$

The covariance matrix $\nabla_{\boldsymbol{\theta}}\boldsymbol{l}\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}$ to be inverted is of dimension $\mathbb{R}^{N\times N}$, N being the batch size (which is 32 in the experimental setup). Since each gradient has a much larger dimension $P \gg N$, it is unlikely that the gradient covariance matrix is singular and requires strong damping.

Second, it is obvious that a large damping factor would change the behaviour of the EF update. With a large damping factor λ (i.e. λ is much larger than the largest eigenvalue of the gradient covariance matrix), the EF update approximately degenerates to an up-scaled SGD update as follows

$$\Delta\boldsymbol{\theta}'_{\text{EF}} \approx -\frac{\eta}{\lambda}\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}\mathbf{1}. \tag{18}$$

To more accurately reflect the behaviour of the EF update, a smaller damping factor is used in our experiments.

4.2. Damping is Not the Main Issue

It is known that many preconditioned optimisers rely heavily on a well-tuned damping factor (Choi et al., 2019). In the case of EF, it is usually believed that the damping factor is used to improve the conditioning of the gradient covariance matrix. In this section, this common belief is challenged through an analysis of existing and additional experimental results.

Recall the definition of the iEF update

$$\Delta\boldsymbol{\theta}_{\text{iEF}} = -\eta(\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}\nabla_{\boldsymbol{\theta}}\boldsymbol{l} + \lambda\boldsymbol{I})^{-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}\boldsymbol{s}_{\text{iEF}}). \tag{19}$$

Similar to the EF update defined in Eqn. 17, the computation for both the updates involves the inversion of the damped covariance matrix $(\nabla_{\boldsymbol{\theta}}\boldsymbol{l}^{\top}\nabla_{\boldsymbol{\theta}}\boldsymbol{l} + \lambda\boldsymbol{I})$ with the same damping factor 1×10^{-7}. If the condition number of the original covariance matrix affects the generation quality of the EF update, it should equally affect the iEF update. However, experimental results show that iEF is able to stably converge while EF consistently causes a diverging training loss.

Additional experiments for EF with a larger damping for the T5 and Prompt-Tuning setup are analysed. The validation results are presented in Table 1. A larger damping of 1×10^{-4} is used, which is searched from $1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6}, 1 \times 10^{-7}$. It can be seen that, although the larger damping EF achieves better performance than the original smaller damping EF, the performance still mostly worse than both SGD and iEF.

Table 1. Validation accuracy for 7 GLUE tasks trained by different optimisers. It is an extended version of Table 4 in main paper, with additional experiments for **EF*** is added, which represents EF trained with 1×10^{-4} damping. **EF** and **iEF** represents EF and iEF training results with the original 1×10^{-7} damping. The average of the highest validation accuracy of each run (3 in total) for every task and optimiser combination is reported, along with a *std*.

	CoLA	SST-2	MRPC	QQP	MNLI	QNLI	RTE
SGD	$69.1_{\pm 0.09}$	$93.3_{\pm 0.14}$	$70.0_{\pm 0.31}$	$85.0_{\pm 4.11}$	$74.5_{\pm 1.17}$	$88.7_{\pm 0.20}$	$54.8_{\pm 0.34}$
EF*	$69.9_{\pm 0.48}$	$93.7_{\pm 0.11}$	$69.5_{\pm 0.14}$	$82.2_{\pm 0.07}$	$72.0_{\pm 0.27}$	$87.4_{\pm 0.18}$	$56.0_{\pm 0.96}$
EF	$73.4_{\pm 0.55}$	$92.9_{\pm 0.24}$	$68.5_{\pm 0.23}$	$64.0_{\pm 0.18}$	$60.7_{\pm 0.32}$	$83.2_{\pm 0.07}$	$57.0_{\pm 0.51}$
iEF	$81.7_{\pm 0.16}$	$\mathbf{94.4}_{\pm 0.09}$	$\mathbf{86.2}_{\pm 1.41}$	$\mathbf{90.7}_{\pm 0.02}$	$\mathbf{83.4}_{\pm 0.09}$	$\mathbf{91.9}_{\pm 0.03}$	$\mathbf{74.6}_{\pm 0.85}$
Adafactor	$\mathbf{82.0}_{\pm 0.44}$	$94.2_{\pm 0.25}$	$84.8_{\pm 0.35}$	$\mathbf{90.7}_{\pm 0.01}$	$82.4_{\pm 0.25}$	$\mathbf{91.9}_{\pm 0.09}$	$64.7_{\pm 0.34}$

A more detailed analysis of the more heavily damped EF updates is presented in Fig. 2, which compares the two EF updates (with small and large damping factors) with the SGD and iEF updates. It can be seen that the larger damping improved the EF updates approximation quality (to NGD updates) mainly by making them closer to SGD updates.

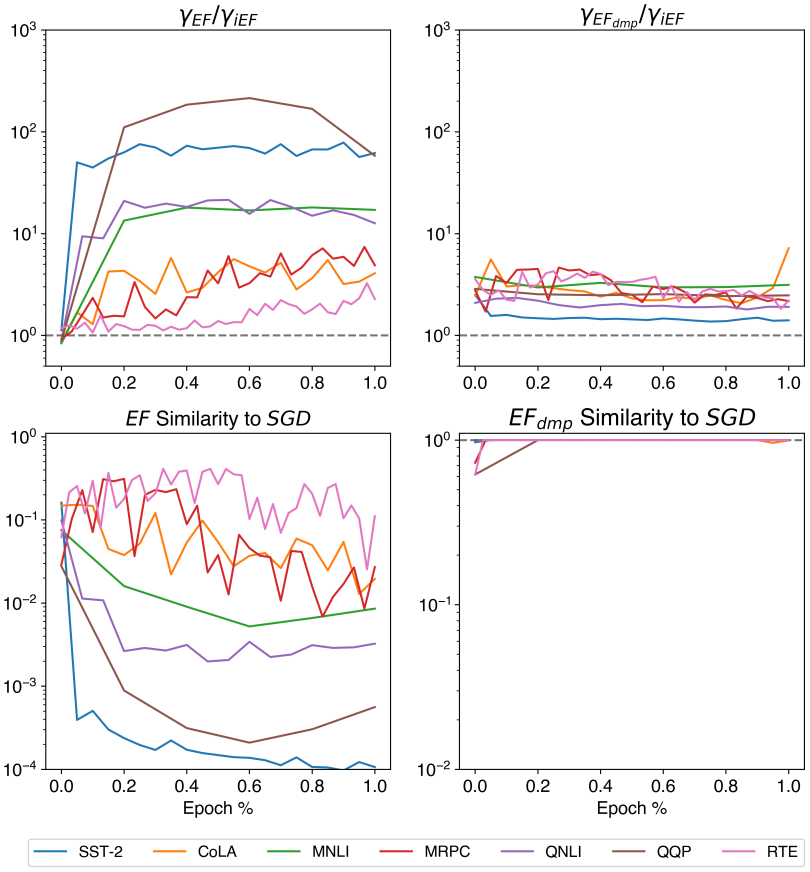


Figure 2. This figure shows four (log-scaled) ratios computed for checkpoints at various stages of training (sampled at the interval of one epoch) and the same set of checkpoints in Fig. 2 of main paper are evaluated. For all the x-axes, 0 indicates the initialised model and 1 indicates checkpoints at the end of the last epoch. In sub-figure titles, both iEF and EF represents the same updates in the main paper, which used a damping factor of 1×10^{-7}. EF_{dmp} represents EF update computed with a larger damping factor 1×10^{-4}. The top row depicts the ratio $\gamma_{\text{EF}}/\gamma_{\text{iEF}}$ and $\gamma_{\text{EF}_{\text{dmp}}}/\gamma_{\text{iEF}}$, which compares the quality of approximation (to NGD update) of the two EF updates relative to iEF updates (larger means worse). The bottom row depicts the cosine similarity between EF updates and the SGD update, with 1 meaning EF update is identical to SGD update. It can be seen that, although a larger damping improve EF updates' approximation quality to NGD, it is in fact because the large damping caused EF updates to degenerate to SGD updates (near 1 cosine similarity to SGD updates).

Overall, it is clear that the small damping factor is not the key reason that caused the poor performance of EF. A larger damping factor improves the EF update curvature behaviour and training performance mainly by making EF updates

degenerate to SGD updates. It covers up the fundamental problem of the EF method, which is believed to be the inversely-scaled projection issue. On the other hand, by fixing this issue with the iEF method, the training performance is significantly improved, despite the use of a small damping factor on the same covariance matrix.

Note that the observations are only applicable to the specific update generation method used in this paper. It remains unknown whether damping behaves similarly in other empirical NGD methods. However, a similar statement is made in (Benzing, 2022) that the K-FAC's (Martens & Grosse, 2015) heuristic damping does suppress the effect of the error covariance matrix.

4.3. Conclusions

This section explains the use of the small damping factor in the experimental setup. It also provides insight into the interaction between EF and a large damping factor, and demonstrated that the small damping cannot account for the poor performance of EF, and the inversely-scale projection issue is the leading underlying problem that should be addressed. Therefore, a possible explanation to the fact that damping is difficult to tune in EF-based methods is exactly that it is not addressing the core issue of the base algorithm.

5. IEF is NOT Just Another Optimiser

In this section, we clarify that the iEF method is mainly proposed as a way to improve existing empirical NGD optimisers, instead of another new optimiser. We view this paper as a continuation of the work of (Kunstner et al., 2019), which highlighted the limitation of the EF approximation (but not a solution). The key contribution of this paper is the identification of the inversely-scaled projection issue of EF, and the proposal of a viable solution, i.e. the iEF method. Although iEF shows potential when applied directly as an optimiser, it is believed to have more profound impact as a practical guideline to improve existing approximate empirical NGD optimisers.

The guideline provided by iEF is discussed in details as follows. Recall the definition of the iEF matrix in Eqn. 5, which can be expanded as follows

$$\tilde{\mathbf{F}}^{\star} = \nabla_{\boldsymbol{\theta}} \boldsymbol{l}^{\top} \operatorname{diag}(\boldsymbol{s}_{\text{iEF}})^{-1} \nabla_{\boldsymbol{\theta}} \boldsymbol{l} = \sum\nolimits_{n} (\|\nabla_{\boldsymbol{z}_n} l_n\|_2^{-1} \nabla_{\boldsymbol{\theta}} l_n)^{\top} (\|\nabla_{\boldsymbol{z}_n} l_n\|_2^{-1} \nabla_{\boldsymbol{\theta}} l_n). \tag{20}$$

It can be seen that, similar to the EF matrix, the iEF matrix is also constructed with per-sample gradients. The only difference lies in that gradients used by iEF are re-scaled by the logits-level gradient norm. This per-sample rescaling operation is easy to implement, meaning it is usually straightforward to apply the iEF improvement to existing empirical NGD methods. Note that the logits-level gradient norm is also straightforward to compute in a modern auto-differentiation framework. In our implementation, we simply invoked the "retain_grad()" Pytorch method on the output logits tensor, and the gradient on the logits tensor can be automatically obtained during the back-propagation stage, with minimal additional memory complexity and no additional time complexity.

The ease of implementation of the iEF method is a key motivation of its design. Most empirical NGD optimisers uses EF instead of the Fisher matrix mainly because it is easy to implement in both standard and distributed setups. The iEF method allows for a straightforward way of improving these optimisers, while preserving their existing practical advantages. Given the popularity of EF-based optimisers, the expected impact of the proposed iEF in this research area is potentially high.

6. Additional Experiments for LoRA Setups

In this section, additional experiments for new model structures and tasks are provided to expand the scope of the current experiments in the main paper. The additional model and tasks combination experimented on are LoRA + T5 applied to GLUE and LoRA + ViT applied to CIFAR100. The detailed experimental setups are explained as follows.

6.1. Experimental Setup

The LoRA + T5 + GLUE setup is based on setups in (Ding et al., 2023), which is similar to the experiments in the main paper. Experiments are conducted for 4 out of the 7 GLUE tasks due to time limit, (CoLA, SST-2, MRPC and QNLI). The LoRA + Vit + GLUE setup is based on (He et al., 2023), where ViT-B-224/16 pre-trained on ImageNet-21k is used as the backbone model. The model was trained for 5 epoch and evaluated every 1000 steps. The checkpoint with the best validation was evaluated on the test set. The SGD baseline has similar performance to the results in (He et al., 2023).

For all experiments, a batch size of 32 was used. The LoRA was set to have rank 8 with dropout 0.1. Runs are compared for AdamW (Loshchilov & Hutter, 2019), SGD, iEF, EF. Note that the Adafactor baseline was switched to AdamW as it is a more commonly used optimiser for LoRA training. Constant scheduling is used for AdamW, SGD and iEF runs. For iEF, a damping of 1×10^{-7} was used, and learning rate of 100 is searched from $\{10, 20, 50, 80, 100\}$. For SGD, a learning rate of 0.1 was used searched from $\{0.1, 1, 10, 100, 1000\}$. For AdamW, a weight decay of 0.01 was used, and learning rate of 1×10^{-3} was searched from $\{1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\}$. Finally, for EF, a normalised update with linear scheduling was used similar to that in the main paper, and starting learning rate of 0.01 was searched from $\{1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}, 1\}$.

6.2. Experimental Results

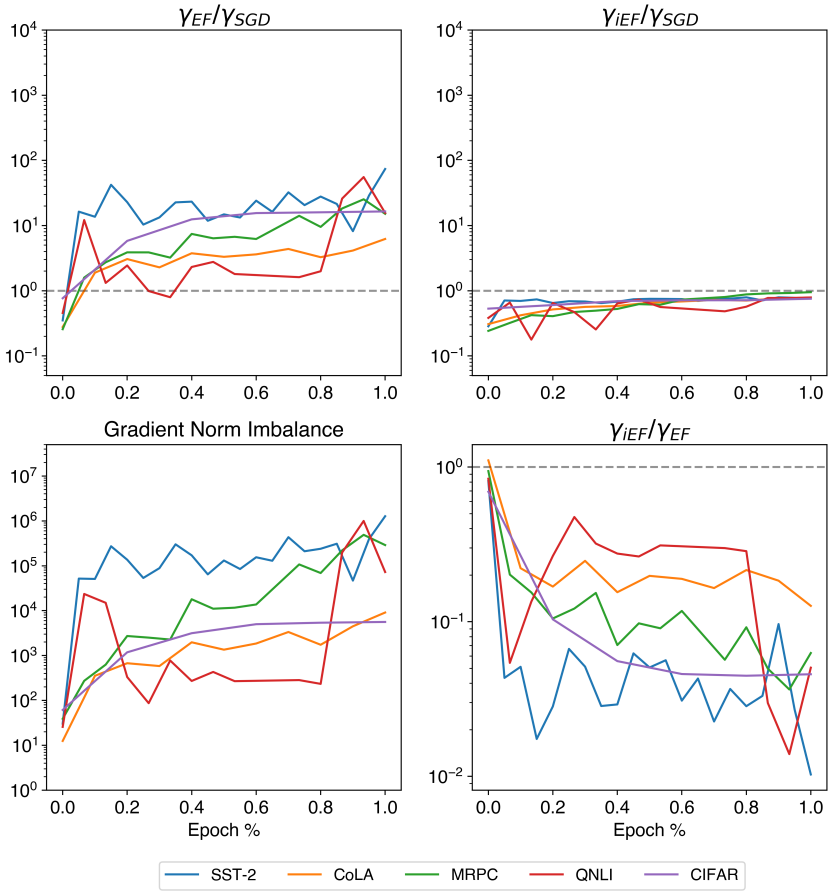


Figure 3. This figure shows four (log-scaled) ratios computed for checkpoints at various stages of training for the 5 LoRA training setups. For all the x-axes, 0 indicates the initialised model and 1 indicates checkpoints at the end of the last epoch. The top-left plot depicts the ratio γ_{EF}/γ_{SGD}, which compares the quality of approximation of EF updates relative to SGD updates. Similarly, top-right plot depicts ratio $\gamma_{iEF}/\gamma_{SGD}$, and bottom-right plot depicts ratio γ_{iEF}/γ_{EF}. The bottom-left plot depicts *imbalance of gradient norms*, which is the average ratio between the maximum and minimum gradient norm for each evaluated batch and a larger value indicates a more imbalanced gradient norm per batch.

6.2.1. Approximation Quality Evaluation

Similar to Sec. 7.2. in the main paper, the empirical evaluation framework is used to evaluate the approximation quality to NGD update of the relevant update methods. The relationship among the approximation quality indicators across epochs and tasks is depicted in Fig. 3. A strong correlation between gradient norm imbalance and EF approximation quality is still observed, further verifying the effect of inversely-scaled projection issue. Also, iEF updates consistently achieves better approximation quality than EF updates throughout the training process.

6.2.2. Optimisation Performance for Different Methods

The validation performance of each new optimiser is presented in Table. 2. Due to time constrain, the test result for GLUE tasks are not provided. However, based on the results in the main paper, it can be seen that the validation accuracy can quite accurately reflect the relative test performance.

In Table. 2, it can be seen that iEF is able to achieve competitive training performance as compared to the SGD and AdamW baselines. Meanwhile, training performance is still the worst for EF on all tasks. Note that the learning rate tuning is difficult for SGD in LoRA setup, whose learning had to be tuned down by 1000 times from the prompt-tuning setup in the main paper (learning rate was 100 previously) for the model to properly converge. In comparison, iEF performs well with a learning rate of 100 (which is similar to the learning rate of 50 previously), thanks to the logits-level gradient norm rescaling.

Table 2. Validation accuracy for CoLA, SST-2, MRPC and QNLI, test accuracy for CIFAR100 is reported for different optimisers. The average of the highest metric for each run (3 in total) for every tasks and optimiser combination is reported, along with an *std*.

	CoLA	**SST-2**	**MRPC**	**QNLI**	**CIFAR100**
AdamW	$83.1_{\pm 0.15}$	$94.9_{\pm 0.07}$	$\mathbf{88.6}_{\pm 0.51}$	$92.2_{\pm 0.06}$	$93.7_{\pm 0.32}$
iEF	$\mathbf{83.4}_{\pm 0.24}$	$94.9_{\pm 0.21}$	$88.5_{\pm 0.88}$	$92.2_{\pm 0.11}$	$\mathbf{94.1}_{\pm 0.15}$
SGD	$81.3_{\pm 0.36}$	$\mathbf{95.0}_{\pm 0.18}$	$87.3_{\pm 0.28}$	$\mathbf{92.3}_{\pm 0.09}$	$90.6_{\pm 1.02}$
EF	$69.1_{\pm 0.01}$	$91.9_{\pm 0.07}$	$55.5_{\pm 0.28}$	$89.4_{\pm 0.21}$	$30.2_{\pm 1.20}$

6.3. Conclusion

The additional experiments investigate the behaviour of EF and iEF method for model and data in addition to the prompt-tuning + GLUE setup in the main paper. The performance of training with LoRA for both NLP and Compute Vision tasks are investigated, which enriches the scope of existing experiments. It is shown that conclusions drawn in the main paper can be generalised to the additional experiments. Overall, it can be concluded that iEF is an effective solution to the inversely-scaled projection issue of EF, which is verified for a range of model structures and datasets.

7. Time and Memory Complexity of Exact iEF Method

In this section, the time and memory complexity of exact iEF method. The iEF update can be exactly computed according to Eqn. 19, which is then used to iteratively optimise a target neural network, as is done in our experiments. Given a batch size of N, and trainable parameters of size P, assume the per-sample gradients have already been computed through backpropagation, the time complexity of computing each update is $O(N^3 + N^2P)$, and the memory complexity is $O(N^2 + NP)$. Assuming $P \gg N$, the time complexity then becomes $O(N^2P)$ and memory complexity becomes $O(NP)$. In practical deep learning frameworks such as Pytorch (Paszke et al., 2019), the limiting factor for the applicability of EF-like method is the memory complexity $O(NP)$, which is essentially the storage of the N per-sample gradients involved in the computation of exact EF-like updates.

In our implementation, the per-sample gradient is obtained by attaching forward and backward hooks to trainable modules, which can automatically retrieve per-sample gradient (with minimal additional cost) when standard backpropagation is executed.

Operation/Method	Per-Batch Wall-Clock Time (ms)
Model Forward	79.36
Model Backward	125.35
SGD Update Generation	0.11
Adafactor Update Generation	0.58
EF/iEF Update Generation (w. hook)	1.66

Table 3. The per-batch wall-clock time for different components in the prompt-tuning setups. "w. hook" means the wall-clock time include the operation time of the forward and backward hook attached to the model. The experiments were run on an Nvidia 3070 GPU. It can be seen that the update generation for EF/iEF is negligible as compared to model forward and backward operations

The wall-clock time in our experiments is reported in Table 3, which reports the per-update wall-clock time for model backward operation, model forward operation, and different update generation methods. It can be seen that, in our prompt-tuning setup in the main paper, where $N = 32$ and $P = 15,360$, the additional time cost for EF/iEF update generation is more than two orders of magnitude smaller than the combined model forward and backward operations, and thus can be considered to have a neglible overhead.

8. Additional Comparison for True Fisher with Sampling Approximation

During our discussion with reviewer XmXQ, another commonly used approximation method for Fisher matrix was suggested, which can be used as another baseline approximation method in addition to EF. In this section, this approximation method, "true Fisher with sampling", is formally introduced. Additional comparisons to the EF/iEF approximation have also been run using the proposed empirical evaluation framework.

8.1. Introduction to True Fisher with Sampling

Recall the definition of Fisher matrix $\mathbf{F}$ as follows

$$\mathbf{F} = \sum\nolimits_n \mathbb{E}_{y\sim p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_n)}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_n) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_n)^\top]. \tag{21}$$

This definition implies that the true Fisher can be computed by sampling the output distribution $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_n)$ of the target model as follow:

$$\hat{\mathbf{F}}(M) := \frac{1}{M} \sum\nolimits_n \sum\nolimits_m [\nabla_{\boldsymbol{\theta}} \log p_{\theta}(\hat{y}_m|\boldsymbol{x}_n) \nabla_{\boldsymbol{\theta}} \log p_{\theta}(\hat{y}_m|\boldsymbol{x}_n)^\top], \tag{22}$$

where $\hat{y}_m \sim p_\theta(y|x_n)$ represents the m-th sampled label for the n-th training datum, and M represents the number of Monte-Carlo (MC) samples generated for each training datum. $\hat{y}_m \in \{c|c = 1, 2, ..., C\}$ where $C \in \mathbb{Z}^+$ is the number of categories. It can be seen that as the number of MC samples generated for each training datum increases, the sampled Fisher $\hat{\mathbf{F}}(M)$ would approach the true Fisher, i.e. $\lim_{M\to\infty} \hat{\mathbf{F}}(M) = \mathbf{F}$. Note that the generation of each MC sample requires at least one additional backpropagation pass, which could add significant computational expense if M is large. Therefore, in practice, a commonly used approximation is $\hat{\mathbf{F}}(1)$, where only one MC sample is generated for each training datum (Martens & Grosse, 2015).

The sampled Fisher can be rewritten as follows

$$\hat{\mathbf{F}}(M) = \frac{1}{M} \boldsymbol{J}^T \boldsymbol{J}, \tag{23}$$

where $\boldsymbol{J} = \left[\cdots, \nabla_{\boldsymbol{\theta}} \log p_\theta(\hat{y}_m|\boldsymbol{x}_n), \cdots\right]^\top \in \mathbb{R}^{(MN)\times P}$ is the sampled gradients. Using the Woodbury identity (Ren & Goldfarb, 2019), the exact $\hat{\mathbf{F}}(M)$ preconditioned update can be efficiently computed as

$$\Delta\boldsymbol{\theta}_{\text{F(M)}} = \left(\hat{\mathbf{F}}(M) + \lambda \boldsymbol{I}\right)^{-1} \nabla_{\boldsymbol{\theta}} \mathcal{L} = \lambda^{-1} \left[\boldsymbol{I} - \boldsymbol{J}^\top \left(\boldsymbol{J}\boldsymbol{J}^\top + \lambda M \boldsymbol{I}\right)^{-1} \boldsymbol{J}\right] \nabla_{\boldsymbol{\theta}} \mathcal{L}. \tag{24}$$

where $\lambda \in \mathbb{R}^+$ is a small damping factor and $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ is the accumulated gradient.

8.2. Approximation Quality for Fisher with Sampling

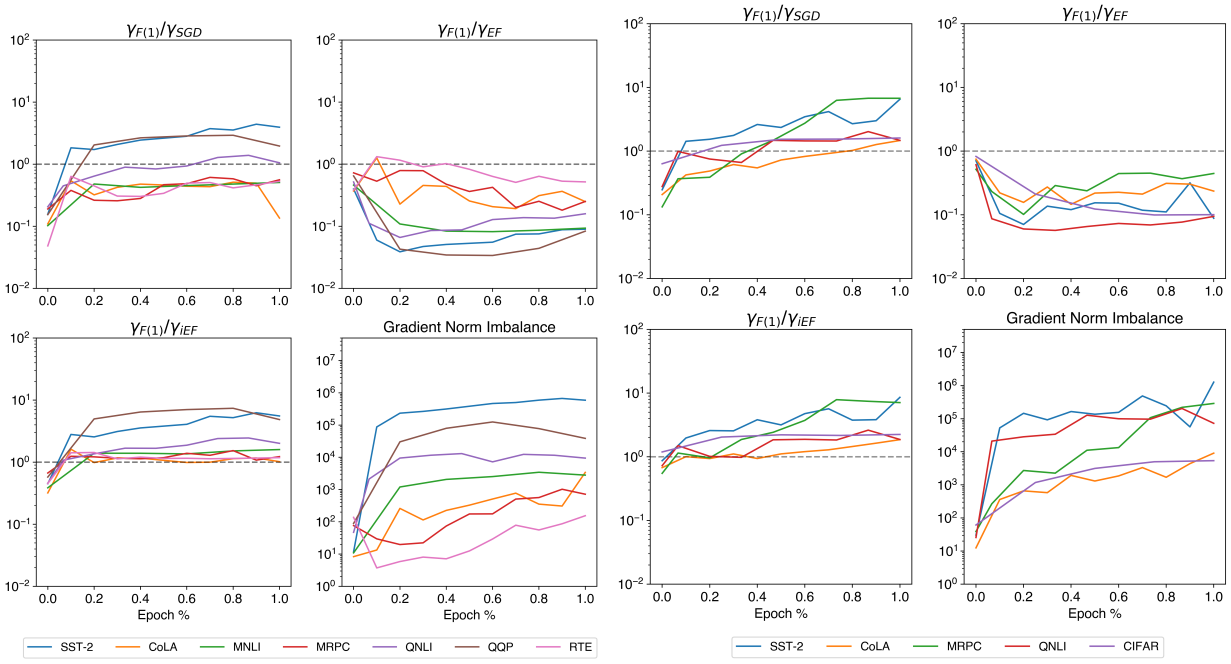
In this section, the approximation quality of updates preconditioned by $\hat{\mathbf{F}}(1)$ to the exact NGD updates are evaluated and compared with EF/iEF updates using our empirical evaluation framework.

$\hat{\mathbf{F}}(1)$ is a good baseline method to benchmark iEF for the following reasons: **1)** EF, iEF and $\hat{\mathbf{F}}(1)$ all have the same maximum rank of N; **2)** $\hat{\mathbf{F}}(1)$ is a practically used approximation method; **3)** $\hat{\mathbf{F}}(1)$ has comparable time and memory complexity to the EF/iEF approximations (although in practice it requires an additional backpropagation for each sample, which means $> 50\%$ time cost).

The γ-indicator for $\hat{\mathbf{F}}(1)$ is termed $\gamma_{\text{F(1)}}$ for simplicity, and it is compared to γ_{SGD}, γ_{EF} and γ_{iEF} at different stages of training for all 12 tasks (prompt-tuning and LoRA). The results are presented in Fig. 4. Consistent trends can be noticed across all 12 setups:

- The $\hat{\mathbf{F}}(1)$ update is consistent superior to EF, iEF and SGD update for only the initial randomly initialised checkpoints.
- As training starts, iEF soon achieves a consistent superior approximation quality. Overall it is clear that based on approximation quality iEF $>$ $\hat{\mathbf{F}}(1)$ $>$ EF.
- $\hat{\mathbf{F}}(1)$ has a superior approximation quality to SGD for only around 50% of all checkpoints evaluated, unlike iEF which shows consistently superior quality to SGD across all training stages (see Fig. 3 and Fig. 2 in the main paper).

A possible explanation of the relatively weaker performance of $\hat{\mathbf{F}}(1)$ as training progresses is as follows. With the increase of the target class probability $p_{\boldsymbol{\theta}}(y_n|\boldsymbol{x}_n)$, there is a higher chance that $y_m = y_n$. It is then more likely that an actual per-sample gradient is used to construct $\hat{\mathbf{F}}(1)$, which causes the inversely-scaled projection issue to affect $\hat{\mathbf{F}}(1)$ updates more. Meanwhile, iEF is not affected by this issue, thus achieves consistent superior approximation quality.



(a) 7 Prompt-tuning Tasks

(b) 5 LoRA Tasks

Figure 4. This figure shows four (log-scaled) ratios computed for checkpoints at various stages of training for all 12 training tasks, which compares the γ ratios among EF, iEF, SGD and $\hat{\mathbf{F}}(1)$. For all the x-axes, 0 indicates the initialised model and 1 indicates checkpoints at the end of the last epoch. In each sub-figure, the top-left plot depicts the ratio $\gamma_{\text{F(1)}}/\gamma_{\text{SGD}}$, which compares the quality of approximation of $\hat{\mathbf{F}}(1)$ updates relative to SGD updates. Similarly, the top-right plot depicts ratio $\gamma_{\text{F(1)}}/\gamma_{\text{EF}}$, and the bottom-left plot depicts ratio $\gamma_{\text{F(1)}}/\gamma_{\text{iEF}}$. For all γ ratios, the smaller means better. The bottom-right plot depicts *imbalance of gradient norms*, which is the average ratio between the maximum and minimum gradient norm for each evaluated batch and a larger value indicates a more imbalanced gradient norm per batch.

8.3. Conclusions

An additional approximation method $\hat{\mathbf{F}}(1)$ has been introduced and compared with both iEF and EF using the evaluation framework across all 12 different training and model setups. For all setups, apart from right at the start of training, iEF consistently demonstrates a superior approximation quality to $\hat{\mathbf{F}}(1)$. Considering that $\hat{\mathbf{F}}(1)$ is a strong baseline approximation method and is also more expensive to compute than iEF, this empirical results further motivates the practical use and value of the iEF method.

References

Ang, A. Convergence of gradient flow, December 2022. URL https://angms.science/doc/CVX/GradientFlowConvAna.pdf.

Benzing, F. Gradient descent on neurons and its link to approximate second-order optimization. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1817–1853. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/benzing22a.html.

Brefeld, U., Fromont, E., Hotho, A., Knobbe, A. J., Maathuis, M. H., and Robardet, C. (eds.). *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part III*, volume 11908 of *Lecture Notes in Computer Science*, 2020. Springer. ISBN 978-3-030-46133-1. URL http://dblp.uni-trier.de/db/conf/pkdd/pkdd2019-3.html.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. On empirical comparisons of optimizers for deep learning, 2019. URL https://arxiv.org/abs/1910.05446.

Ding, N., Qin, Y., Yang, G., Wei, F., Zonghan, Y., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H.-T., Chen, J., Liu, Y., Tang, J., Li, J., and Sun, M. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:1–16, 03 2023. doi: 10.1038/s42256-023-00626-4.

He, X., Li, C., Zhang, P., Yang, J., and Wang, X. E. Parameter-efficient model adaptation for vision transformers. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i1.25160. URL https://doi.org/10.1609/aaai.v37i1.25160.

Kunstner, F., Balles, L., and Hennig, P. *Limitations of the Empirical Fisher Approximation for Natural Gradient Descent*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2408–2417. JMLR.org, 2015.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Ren, Y. and Goldfarb, D. Efficient subsampled gauss-newton and natural gradient methods for training neural networks. *CoRR*, abs/1906.02353, 2019. URL http://arxiv.org/abs/1906.02353.

Thomas, V., Pedregosa, F., van Merrienboer, B., Mangazol, P.-A., Bengio, Y., and Roux, N. L. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Zhang, G., Martens, J., and Grosse, R. B. Fast convergence of natural gradient descent for over-parameterized neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1da546f25222c1ee710cf7e2f7a3ff0c-Paper.pdf.