Rebuttal:

We thank the reviewers for their helpful and insightful suggestions. We first address a few common questions in the meta-review and then each reviewer's individual questions.

##Common Questions

C1: How can we determine the hyper-parameters in experiments, and how do they affect the algorithm?

In general, the selection of hyper-parameters can have an important impact on the training accuracy and the final model performance (e.g., training time).

In our Mixer, parameter K is used to determine the adaptive switching condition (Sec. 6.2) when given an adjustable parameter θ. Specifically, if the average loss of the current optimizer after K consecutive iterations is less than the pre-set threshold θ, we find that the optimizer has little effect on improving the objective function, so the switcher performs switching operation here. Theoretically, the smaller the threshold θ is, the more stringent the switching condition is, and the larger K is; Conversely, the larger θ is, the more relaxed the switching condition is, and the smaller K is. In experiments, we empirically find that the target accuracy and training time of Mixer balance when θ=0.001 and K=5. In the final version, we will add a parameter study section to verify the effect of these parameters on the algorithm.

We utilize the parameter $T1$ in gradient vector correction strategy (Sec. 7.1) to determine the fusion period of the gradient vectors of first-order and second-order optimizers. In this fusion period, fusion gradients are computed in convex combination and model parameters are updated via momentum. After the fusion period, Mixer fully switches from the second-order to the first-order optimizer. Note that a small $T1$ leads to rapid switching and oscillating convergence, while a large $T1$ prolongs training. To balance training time and accuracy, we set $T1$=10 in experiments.

In corrector, we propose a gradient vector correction strategy based on exponential moving average (EMA). To mitigate performance degradation from optimizer switching, we fuse the gradients of second-order and first-order methods, weighted by parameter μ. As Mixer switches from a second-order to a first-order method, we preserve the first-order gradient with a higher weight in the fused gradient. Experiments show the optimal performance at μ=0.8.

C2: What is the theoretical analysis of Mixer when using Cholesky decomposition in traditional K-FAC method?

Currently, the theoretical convergence rate of most optimization algorithms only considers a single type of optimizer. It is challenging to prove the theoretical convergence rates for hybrid methods. This is because the convergence rate of the hybrid method should account for the characteristics and applicable conditions of both the first-order and second-order methods.

To strengthen the credibility of the proposed method, we will provide an approximate theoretical analysis of a lightweight second-order optimizer based on the Cholesky decomposition technique in the final version. Please see the following proof.

Proof:

In lightweight second-order optimizer, we employ Cholesky decomposition technique to obtain factors $\mathbf{A}_{i-1}$ (i.e., the activation of layer $i-1$) and $\mathbf{G}_i$ (i.e., the gradient of layer i). It means that for a positive definite matrix $\mathbf{A}_{i-1}$ (its size is $n\times n$) or $\mathbf{G}_i$ (its size is $n\times n$) can be decomposed into a product of a lower triangular matrix $\mathbf{Y}$ ($\mathbf{X}$) and $\mathbf{Y}^\top$ ($\mathbf{X}^\top$), namely,

$$\begin{aligned}\mathbf{A}_{i-1} &= \mathbf{Y}\mathbf{Y}^\top \rightarrow \mathbf{Y}^{-1} = \mathbf{Y}^\top \mathbf{A}_{i-1}^{-1},\\ \mathbf{G}_i &= \mathbf{X}\mathbf{X}^\top \rightarrow \mathbf{X}^{-1} = \mathbf{X}^\top \mathbf{G}_i^{-1}.\end{aligned}$$

After k iterations in traditional K-FAC for layer i, we have

$$\mathbf{w}^{k+1} = \mathbf{w}^0 - \eta \sum_{j=0}^{k} \mathbf{G}_i^{-1} \nabla L(\mathbf{w}^j) \mathbf{A}_{i-1}^{-1},$$

where $\mathbf{w}^k$ is the parameter in iteration k, $L(\cdot)$ is the loss function, η is a learning rate. If we use $\mathbf{X}$ and $\mathbf{Y}$ to replace $\mathbf{G}_i$ and $\mathbf{A}_{i-1}$ in above equations, we have

$$\mathrm{w}^{k+1} = w^0 - \eta \sum_{j=0}^{k} \mathbf{X}^{-1} \nabla L(w^j) \mathbf{Y}^{-1}.$$

Let $\mathbf{w}^0 = \mathrm{w}^0$, we consider the difference between w^{k+1} and $\mathbf{w}^{k+1}$ as follows.

$$
\begin{aligned}
\mathbf{w}^{k+1} - \mathrm{w}^{k+1} &= \eta \sum_{j=0}^{k} \mathbf{G}_i^{-1} \nabla L(\mathbf{w}^j) \mathbf{A}_{i-1}^{-1} - \eta \sum_{j=0}^{k} \mathbf{X}^{-1} \nabla L(w^j) \mathbf{Y}^{-1} \\
&= \eta \sum_{j=0}^{k} [\mathbf{G}_i^{-1} \nabla L(\mathbf{w}^j) \mathbf{A}_{i-1}^{-1} - \mathbf{X}^{-1} \nabla L(w^j) \mathbf{Y}^{-1}] \\
&= \eta \sum_{j=0}^{k} [\mathbf{G}_i^{-1} \nabla L(\mathbf{w}^j) \mathbf{A}_{i-1}^{-1} - \mathbf{X}^{\top} \mathbf{G}_i^{-1} \nabla L(w^j) \mathbf{Y}^{\top} \mathbf{A}^{-1}] \\
&\leq \eta k \max_{0 \leq j \leq k} \{\mathbf{G}_i^{-1} \nabla L(\mathbf{w}^j) \mathbf{A}_{i-1}^{-1} - \mathbf{X}^{\top} \mathbf{G}_i^{-1} \nabla L(w^j) \mathbf{Y}^{\top} \mathbf{A}^{-1}\}
\end{aligned}
$$

For simplicity, let $\mathbf{G}_i^{-1} \nabla L(\mathbf{w}^j) \mathbf{A}_{i-1}^{-1} = \mathbf{V}_1^{(j)}$ and $\mathbf{X}^{\top} \mathbf{G}_i^{-1} \nabla L(\mathrm{w}^j) \mathbf{Y}^{\top} \mathbf{A}^{-1} = \mathbf{V}_2^{(j)}$.

We now consider $\| \mathbf{w}^{k+1} - \mathrm{w}^{k+1} \|_2$. According to the above derivation, we have

$$
\begin{aligned}
\| \mathbf{w}^{k+1} - \mathrm{w}^{k+1} \|_2 &\leq \eta k \max_{0 \leq j \leq k} \{\| \mathbf{V}_1^{(j)} - \mathbf{V}_2^{(j)} \|_2\} \\
&\leq \eta k \max_{0 \leq j \leq k} \{\| \mathbf{V}_1^{(j)} \|_2 + \| \mathbf{V}_1^{(j)} \|_2\} \\
&= \eta k \max_{0 \leq j \leq k} \{\sqrt{\max_{1 \leq q \leq n} | \lambda_q |} + \sqrt{\max_{1 \leq l \leq n} | \rho_l |}\}.
\end{aligned}
$$

Where $\max_{1 \leq q \leq n} | \lambda_q |$ and $\max_{1 \leq l \leq n} | \rho_l |$ denote the largest eigenvalues of $\mathbf{V}_1^{(j)}$ and $\mathbf{V}_2^{(j)}$.

The inequality above indicates the gap between the approximation method (using Cholesky decomposition and triangular matrix replacement strategy) and the original K-FAC can be bounded. This theoretically guarantees the effectiveness of the lightweight second-order optimizer.

C3: What are the differences between FOSI[1] and Mixer?

FOSI improves the performance of first-order optimizers by incorporating second-order information during the optimization. Specifically, in each iteration, it implicitly splits the objective function into two quadratic functions defined on orthogonal subspaces, then uses a second-order method (e.g., Newton's Method) to minimize the first, and the base optimizer (first-order method) to minimize the other. In the algorithm, they employ the Lanczos algorithm [2] to estimate the extreme eigenvalues and vectors of the Hessian.

Different from the FOSI, our Mixer focuses on a combination of first-order and second-order methods. More specifically, we propose a lightweight second-order optimizer based on Cholesky decomposition in early training. Once the loss of the objective function satisfies the adaptive switching condition, Mixer uses a first-order optimizer in each iteration. The main contributions of Mixer are: (1) introducing an adaptive switching condition based on the change rate of loss to avoid the additional overhead

caused by frequent switching, (2) presenting a gradient vector correction strategy based on exponential moving average (EMA) to search the optimal gradient direction and to mitigate accuracy degradation, and (3) incorporating a lightweight second-order method by using efficient Choleksy decomposition and triangular matrix replacement to reduce computation and memory overhead. We will compare the performance differences with FOSI in the final version if necessary.

[1] FOSI: Hybrid first and second order optimization. arXiv preprint arXiv:2302.08484, 2023.
[2] Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Journal of research of the National Bureau of Standards, 45:255–282, 1950.

C4: What is the memory footprint of Mixer in experiments?

Thanks for valuable advice. We will add an experimental section to compare the memory footprint of the Mixer and Baseline methods. We conduct three experiments:
1) For ResNet-101 (ImageNet-1K), we compare memory footprint of SGD, KAISA, and Mixer during training. The memory footprints of SGD, KAISA and Mixer are 38.02GB, 56.81GB and 46.91GB. Mixer saves 17.42% memory footprint compare to KAISA.
2) For ResNet-152 (ImageNet-1K), we compare memory footprint of SGD, KAISA, and Mixer during training. The memory footprints of SGD, KAISA and Mixer are 50.42GB, 77.18GB and 56.48GB. Mixer saves 26.82% memory footprint compare to KAISA.
3) For BERT-large (stage2), we compare memory footprint of LAMB, KAISA, and Mixer during training. The memory footprints of LAMB, KAISA and Mixer are 33.04GB, 68.50GB and 47.13GB. Mixer saves 31.19% memory footprint compare to KAISA. The following table shows the results.

Algorithm	Model	Local Batch Size	#GPU	Memory footprint on each GPU
SGD/LAMB	ResNet-101	256	8	38.02GB
	ResNet-152	256	8	50.42GB
	Bert-large	32K	8	33.04BG
KAISA	ResNet-101	256	8	56.81GB
	ResNet-152	256	8	77.18GB
	Bert-large	32K	8	68.50GB
Mixer	ResNet-101	256	8	46.91GB
	ResNet-152	256	8	56.48GB
	Bert-large	32K	8	47.13GB

C5: What is the switching condition (e.g., switching point) of Mixer in experiments?

In the supplementary material (Section H), we provide an example to show that a hybrid method without any improved techniques leads to accuracy degradation during the training. To address this, an adaptive switching condition (Sec. 6.2) based on the change rate of loss is designed. The following table shows the optimizer switching conditions (e.g., "#epoch/#step" is the number of epoch or step) corresponding to the experiments in Sec. 8.1 in paper.

	ResNet-32	ResNet-56	ResNet-101	ResNet-152	Bert-large
#epoch/#step	59	57	34	37	341

Reviewer1:

Q1: Can FP32 solve all the data overflow problems? Is there necessity to use FP64?

To our knowledge, FP32 is sufficient for addressing data overflow issues during training ResNet-56 models on ImageNet-1K and Cifar10 for image classification, and BERT-large models on Wiki for reading comprehension. While FP64 is theoretically possible, considering its larger memory footprint compared to FP32, we choose the FP32.

Q2: Can you show more details about the hyper-parameters selection? For example, K, and T1. What is the impact of these parameters?

We thank the reviewer for the suggestion and will add a discussion on hyper-parameters selection in Sec.7.2 in the revised paper. In general, the selection of hyper-parameters can have an important impact on the training and the model performance.

In our Mixer, parameter K is a parameter used to determine the adaptive switching condition when given an adjustable parameter θ. Specifically, if the average loss of the current optimizer after K consecutive iterations is less than the pre-set threshold θ, we find that the optimizer has little effect on improving the objective function, so the switcher performs the switching operation here. Theoretically, the smaller the threshold θ is, the more stringent the switching condition is, and the larger K is; Conversely, the larger θ is, the more relaxed the switching condition is, and the smaller K is. In experiments, we empirically find that the target accuracy and training time of Mixer balance when θ=0.001 and K=5.

We utilize the parameter $T1$ in gradient vector correction strategy to determine the fusion period of the gradient vectors of first-order and second-order optimizers. In this fusion period, fusion gradients are computed in convex combination and parameters are updated via momentum. After fusion period, Mixer fully switches from the second-order to the first-order optimizer. Note that a small $T1$ leads to rapid switching and oscillating convergence, while a large $T1$ prolongs training. Balancing training time and accuracy, we set $T1$=10 in experiments.

Q3: The evaluation is insufficient. The memory footprint is expected since the paper emphasizes reduced memory overhead in the abstract.

It is discussed in C4.

Q4: The evaluation is not sufficient. Section 8 only provides converge times. However, more comparisons like memory footprint and computational complexity are expected here.

It is discussed in C1, C3 and C4.

Q5: Ablation study on how to determine the hyper-parameters is very important to affect the results.

It is discussed in C1.

Q6: The readers may be curious about more details about the evaluation, for example at which epoch the optimizer is switched and how is the accuracy affected.

It is discussed in C5.

Reviewer2:

Q1: The authors are suggested to review the more recent similar work [1] which also employs a hybrid method to address the coverage problem. [1]FOSI: Hybrid First and Second Order Optimization

We totally agree with your suggestion, and we will add a discussion in related work Section 2.2 in the final version to illustrate the differences between the work [1] and our work. Please see Common questions C3.

Q2: The authors argue that "Theoretically, an effective hybrid method should exhibit both fast convergence properties similar to second-order methods and good end-to-end performance of first-order ones." However, as shown in Figure 1, there appears to

be no discernible difference in the final performance between one-order and second-order optimization methods. In my view, the authors aim to combine the efficiency levels of these two optimizations: the fast convergence of second-order methods and the low complexity of one-order methods, which can be quantified by comparing training epochs to reach a target performance and the corresponding wall-clock times. It is suggested to clarify this point for better readability.

Thanks for pointing it out. We commit to thoroughly reviewing our paper to implement modifications that enhance readability. More specifically, we first explain the pros and cons of the first-order and second-order methods in terms of convergence rate and training time in Figure 1. Second, we add a new curve (Mixer) in the final version to show our motivation by comparing the hybrid approach with these two approaches.

Q3: More theoretical evidence should be provided to support the claim of a fast coverage rate. This could involve rigorous analysis or mathematical proofs demonstrating the convergence properties and efficiency gains of the proposed hybrid method compared to traditional one-order and second-order optimization approaches. Such theoretical underpinnings would strengthen the credibility of the proposed method and provide deeper insights into its effectiveness.

It is discussed in C2.

Q4: I wonder whether the authors have conducted statistical analysis about the training point the optimizer switches between one-order and second-order methods. Such analysis could indeed provide valuable insights into the behavior of the proposed hybrid optimizer and help validate its effectiveness.

It is discussed on C5.

Q5: Please provide the detailed the parameter analyse about “θ” and “μ”. It is confused about the setting of these parameters.

We thank the reviewer for the suggestion and will add a discussion on hyper-parameters selection in Sec.7.2 in the revised paper. In general, the selection of hyper-parameters can have an important impact on the training process of an algorithm and the final model performance.

In our Mixer, parameter K is used to determine the adaptive switching condition (Sec. 6.2) when given an adjustable parameter θ. Specifically, if the average loss of the current optimizer after K consecutive iterations is less than the pre-set threshold θ, we find that the optimizer has little effect on improving the objective function, so the switcher performs the switching operation here. Theoretically, the smaller the threshold

θ is, the more stringent the switching condition is, and the larger K is; Conversely, the larger θ is, the more relaxed the switching condition is, and the smaller K is. In experiments, we empirically find that the target accuracy and training time of Mixer balance when θ=0.001 and K=5. In the final version, we will add a parameter study section to verify the effect of these parameters on the algorithm.

In corrector, we propose a gradient vector correction strategy based on exponential moving average (EMA). To mitigate performance degradation from optimizer switching, we fuse the gradients of second-order and first-order methods, weighted by parameter μ. As Mixer switches from a second-order to a first-order method, we preserve the first-order gradient with a higher weight in the fused gradient. Experiments show the optimal performance at μ=0.8.

Reviewer3:

Q1: The author of the paper claim it is the first effort for hybrid optimizer. However, [1] also proposes a first and second order hybrid optimization to train deep learning model. The author of the paper should revise this part and also discuss and compare with [1] to clarity of this paper.

We totally agree with your suggestion, and we will add a discussion in related work section 2.2 in the final version to illustrate the differences between the work [1] and our work. Please see Common questions C3.

Q2: The adaptive switching condition uses θ=0.001, which is not explained. The author of the paper should analyze how this parameter affect the final results, and propose a reasonable solution to find this parameter.

It is discussed in C1.

Q3: The ImageNet-1K results should be placed in the main paper, not in Appendix. This is a structure issue of the paper that need to be fixed.

Thanks for pointing it out. We will fix the structure of the final paper. More specifically, we show the ImageNet-1K results (Sec. J.1 and Sec. J.2) in the main paper instead of Appendix.

Reviewer4:

Q1: I'm a little curious about periodically update parameters with cached Cholesky decomposition. Is there any theoretical guarantee for this approximation?
Please refer to the Weaknesses for other questions.

In our Mixer, we propose a lightweight second-order optimization algorithm based on Cholesky decomposition and a trig matrix replacement strategy. We theoretically prove the approximation between this method and the traditional K-FAC algorithm. Please see C2 for details.

Q2: The Heuristic Switching Condition seems to have no practical application since it requires two additional training sessions.

The main idea of the heuristic switching condition is simple: if a first-order optimization method performs similarly (e.g., loss) to a second-order one, it switches to the other. As you mentioned, this condition requires prior training using only first- or second-order methods, making it impractical. This motivates us to explore and propose an adaptive switching condition.

Q3: Adaptive Switching Condition determines the switching timing through gradient changes. If a local optimum is encountered in the early stage of training, if I understand correctly, Mixer will switch to first-order optimization early. In that case, the advantages of second-order optimization are hidden.

Existing first-order and second-order methods are often influenced by local minima. Some work attempts to mitigate local minima from various aspects such as [3], [4], [5].

In this study, we apply strategies like model initialization (e.g., seed change) and learning rate adjustment (e.g., warm-up) to avert local optimum. On the other hand, first-order optimization methods, including SGD, incorporate momentum terms to help algorithms escape local optimum. The LAMB algorithm adopts an adaptive learning rate and weight decay to avoid local optimum. Furthermore, we will explore regularization and various search strategies in future work for this issue.

[3] Goyal, Priya, et al. "Accurate, large mini-batch SGD: Training ImageNet in 1 hour." arXiv preprint arXiv:1706.02677 (2017).
[4] Barrett, David, and Benoit Dherin. "Implicit Gradient Regularization." International Conference on Learning Representations. 2020.
[5] Kleinberg, Bobby, Yuanzhi Li, and Yang Yuan. "An alternative view: When does SGD escape local minima?." International conference on machine learning. PMLR, 2018.

Q4: There is a lack of ablation experiments for Switcher and Corrector, and their

effectiveness has not been verified.

Thanks for the suggestion. We present examples of the hybrid method without switcher and corrector in the appendices (Sec. G and Sec. H), respectively. We will add ablation experiments for switcher and corrector to the final version to verify their effectiveness.

Q5: Missing memory usage comparison.

It is discussed in C4.