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: Experimental part is weak and not sufficient. Authors mainly focus on the effectiveness of ResNet-based CNNs and BERT.
1) It will be better to consider autoregressive models such as GPT-based models to support the effectiveness of the proposed optimization framework in the LLM era.
2) For comparison, baselines including SGD (COMPSTAT 2010), LAMB (ICLR 2020), and KAISA (SC 2021) are not the latest works. Through a quick search, more strong baselines such as [1,2,3] can be considered. Specifically, [3] is the most recent hybrid method.

Related references:
[1] A modified Adam algorithm for deep neural network optimization (Neural Computing and Applications 2023, HN Adam)
[2] Eva: A General Vectorized Approximation Framework for Second-order Optimization (arXiv 2023)
[3] FOSI: Hybrid First and Second Order Optimization (ICLR 2024)

Although large language models are popular in deep learning, few studies have been done to train these models based on second-order optimal algorithms. Furthermore, the training environment and training time of GPT-based models is difficult to be satisfied in a restricted experimental platform. Therefore, in our paper, we focus on ResNet-based CNN as well as BERT and compare them with some typical optimization algorithms. In the final version, we will train these GPT-based models using Mixer.

As for the selection of comparison algorithms, we chose SGD, LAMB and KAISA (a modified version of K-FAC) because these methods are commonly used in many literature when training models. In addition, these methods do not train GPT-based models for performance evaluation in their papers. In the final version, we will add the latest works for comparison.

C2: The research work on deep learning optimizers is quite challenging, and even fewer can generate truly applicable results. The experiments on ResNet and Bert conducted by the authors are not sufficiently comprehensive, making it difficult to convince others of their practical value.

It is well known that no single optimizer can always stay undefeated among all its competitors across different network architectures and application settings. For instance, for CNNs, SGD often achieves better generalization performance than adaptive gradient algorithms such as Adam, whereas on vision transformers, SGD often fails to get better performance [4].

We consider that the main application areas of deep learning are computational vision and natural language processing. Therefore, our optimizer performance evaluation focuses on CNN and BERT. In the future, we can evaluate Mixer in terms of multimodal models to enhance its universal applicability.

[4] Adan: Adaptive Nesterov Momentum Algorithm for Faster Optimizing Deep Models (IEEE PAMI2024)

C3: In Method, adaptive switching condition is based on whether the change rate of the loss is small enough for a period of epoch interval. How to determine “a period of epoch interval” may be important to the overall accuracy. This part is not so clear.

In the supplementary material (Page 14, A6), 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 (Switcher) based on the change rate of loss is designed.

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 $\theta=0.001$ 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 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 $\mu=0.8$.

Reviewer1:

Q1: The analysis of different proposed techniques are not clear. The design of Mixer includes three components including Decomposer, Switcher and Corrector. Specifically, Decomposer leverages using a lightweight second-order method, Switcher estimates the switching condition, and Corrector mitigates accuracy degradation. However, this paper lacks of analysis of how different components affect the training efficiency and accuracy. I hope more ablation studies can be added to the experimental part.

We totally agree with your suggestion, and we will add ablation studies in evaluation in the final version to show how different components (Decomposer, Switcher and Corrector) affect the training efficiency.

Furthermore, we believe that these components do play a role in the convergence of the algorithm. In fact, we also try and analyze the role of these components. For example, we record the moment when the Switcher takes effect in different experiments. The following table shows the optimizer switching conditions (e.g., “#epoch/#step” is the number of epoch or step) corresponding to the experiments in main paper.

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

Q2: The description of motivation part can be further improved by more supporting data. I think the authors can record the computation time and memory footprint of first-order and second-order methods on models like ResNet-50 and BERT. In this way, we can have more intuitive understanding of their differences.

We thank the reviewer for the suggestion, and will add a comparative analysis of computational time and memory footprint for different optimization methods in Motivation part to improve the quality of the paper.

In current version, we have compared our Mixer with other optimization methods（SGD， LAMB，KAISA）in both computational time and memory footprint in the experiment section. Specifically, in terms of computational time, Mixer saves 5.26%~48.86% of training time compared with other methods（see Page7, Experiment section and Page16, Table 3）. In terms of memory overhead, Mixer saves 17.42%~31.19 memory footprint compared with KAISA (See Page19, Table 6).

Table 6: The memory footprint of algorithms. ’#’ is the number of GPUs

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

Reviewer2:

Q1: Some contributions are still limited, such as the Cholesky decomposition, and EMA. It seems like a patchwork of several modules, lacking more original innovation.

Our Mixer addresses three key challenges (see Page 1 in paper) facing hybrid optimization algorithms. For each challenge, we have proposed corresponding solutions, and provided theoretical proof and analysis to ensure their effectiveness (see Lemma 1~6 and the appendix of paper). Furthermore, introducing Cholesky decomposition directly is not feasible. To address this, we first propose a positive definite matrix transformation strategy (see Page 13), which has not been considered in previous second-order methods.

Our Mixer focuses on a combination of first-order and second-order methods to accelerate model training. 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.

In general , 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.

Q2: The literature review is limited, lacking some recent related research works.

Thanks for pointing it out. Our related work mainly reviews the recent improvement of the classical second-order method NGD and hybrid optimization algorithms. These algorithms include similar work such as FOSI, Eva, etc. Due to space constraints, we do not review all the references. In the final version, we will add more related references.

Q3: The overall writing needs improvement, such as:
a. Repeatedly introducing the saved time and acceleration multiples in the abstract.
b. Not providing definitions for related symbols before Lemma 1.
c. The overall statement of Lemma 5 is incomplete.

a. In the abstract, we should state the most critical methods of the paper and the performance evaluation results. We evaluate the performance of Mixer training CNN and BERT and compare it with typical methods. The experimental results of Mixer are superior to those of the comparison methods.

b. Due to space constraints, we have provided detailed background and related symbols instructions of the optimization algorithm in Appendix (A1 Page 10 and Page 11). Later, we will add it into the main paper.

c. Lemma 5 theoretically analyzes the effect of using Cholesky decomposition on parameter updating. This lemma proves that the model parameter deviation before and after using the approximation strategy based on Cholesky is bounded.

Reviewer3:

Q1: The mixed precision training relies on an outdated Cholesky-decomposed matrix from the second epoch of each interval, yet the impact of this approach on convergence remains unexplored.

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.

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.

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. Please see the following proof (See Page 12).

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.

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

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

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

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.

Q2: The automatic switcher shifts optimizers based on the rate of loss change; however, deeper insights are needed, such as examining whether this rate reliably indicates the necessity of a switch.

It's discussed in C3.

Q2: Furthermore, while the authors suggest that sharp gradient changes may degrade accuracy, this claim lacks supporting references. Figure 4 is also potentially misleading: subfigure (a) depicts an angle between g1 and g2 greater than pi/2, while (b) shows an angle smaller than pi/2, which does not clearly illustrate the benefits of the correction strategy.

Analysis of the reasons for the drop in accuracy is not unfounded. In fact, we are inspired by literature [5] and [6]. In these works, the authors also discuss similar problems in optimizing algorithm switching. In addition, Figure 4 is only a two-dimensional case, and its purpose is to provide an intuitive explanation.

[5] Combining optimization methods using an adaptive meta optimizer(Algorithms 2021）
[6] Improving generalization performance by switching from Adam to SGD (arxiv 2017)