GENERALIZED FISHER-WEIGHTED SVD: SCALABLE KRONECKER-FACTORED FISHER APPROXIMATION FOR COMPRESSING LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Fisher information is a fundamental concept for characterizing the sensitivity of parameters in neural networks. However, leveraging the full observed Fisher information is too expensive for large models, so most methods rely on simple diagonal approximations. While efficient, this approach ignores parameter correlations, often resulting in reduced performance on downstream tasks. In this work, we mitigate these limitations and propose Generalized Fisher-Weighted SVD (GFWSVD) — a post-training LLM compression technique that accounts for both diagonal and off-diagonal elements of the Fisher information matrix, providing a more accurate reflection of parameter importance. To make the method tractable, we introduce a scalable adaptation of the Kronecker-factored approximation algorithm for the observed Fisher information. We demonstrate the effectiveness of our method on LLM compression, showing improvements over existing compression baselines. For example, at a 20% compression rate on the MMLU benchmark, our method outperforms FWSVD, which is based on a diagonal approximation of the Fisher information, by 5%, SVD-LLM by 3%, and ASVD by 6%.

1 INTRODUCTION

The Fisher Information Matrix (FIM) (5) is widely employed in neural networks to enhance the efficiency of models, particularly in the context of training and inference. However, computing and leveraging the full Fisher information is computationally prohibitive for deep networks. To make the problem tractable, existing methods adopt simplified approximations – most commonly, assuming that the Fisher matrix is diagonal (29; 6; 24). While efficient, this assumption discards valuable information about parameter correlations.

One key application of FIM is low-rank compression of large language models (LLMs). However, the standard low-rank approach — Singular Value Decomposition (SVD) — often leads to suboptimal performance. To mitigate this, weighted SVD methods aim to align the optimization objective with the target task (30; 12). Fisher-Weighted SVD (FWSVD) (12) uses Fisher information to assign importance to parameters. However, FWSVD utilizes only the diagonal part of FIM and treats each row as independent, which can lead to poor retention of task-critical components.

In contrast, we propose a more accurate weighted SVD method: **Generalized Fisher-Weighted SVD (GFWSVD)**. Our approach leverages a Kronecker factorization of the full FIM to derive two sensitivity matrices, which are integrated into a generalized SVD framework. To overcome the high computational cost of factorizing the full Fisher matrix, we introduce a scalable adaptation of the Kronecker decomposition algorithm. We compare our method with various low-rank compression approaches for large models — those using Fisher information (Fisher-Weighted SVD), and those leveraging activation statistics (ASVD (30), SVD-LLM (28)) — and observe consistent improvements in downstream task performance.

To summarize, our main contributions are as follows:

- We introduce **Generalized Fisher-Weighted SVD (GFWSVD)**, a new weighted SVD-based method for compressing large language models, which leverages the Kronecker-decomposed

Fisher information that encodes both row-wise and column-wise parameter correlations. We prove that **GFWSVD** is a generalization of FWSVD (12).

- We propose a computationally effective adaptation of the Kronecker decomposition algorithm for the Fisher information matrix (FIM) that captures its full structure without relying on diagonal or other simplifying approximations.
- We empirically show that our method preserves model performance under compression while maintaining efficiency, outperforming existing techniques within its class.

2 Related Work

Fisher information is a fundamental tool for estimating parameter importance in neural networks. It has been used in continual learning to prevent catastrophic forgetting (14), in federated learning to guide local update strategies (13), and more recently for merging fine-tuned models at the parameter level (17). Due to the computational complexity of the FIM, many methods introduce structural assumptions to enable efficient approximations. A prominent approach is Kronecker product factorization, which decomposes FIM into tractable components. KFAC (9) pioneered this idea for convolutional layers, showing that structural constraints can preserve key curvature information while reducing cost. Later work (25) improved training efficiency through faster Kronecker-factored approximations, while KPSVD (15) applied singular value constraints to enable memory-efficient FIM approximations in large-scale models. These efforts primarily focus on improving optimization and training. In contrast, our work leverages Kronecker-product FIM approximation for post-training model compression, aiming to preserve task sensitivity while significantly reducing model size. These methods commonly assume *layer independence*, reducing the full-network FIM to a block-diagonal form and enabling per-layer analysis. We adopt the same assumption in this work, focusing on improving compression at the single-layer level.

Structural approximations have shown promise in post-training model compression. For instance, SparseGPT (7) ranks weights using curvature estimates for pruning, while FWSVD (12) applies diagonal FIM approximations to guide task-aware SVD compression. As we later demonstrate, FWSVD emerges as *a special case of our more general framework*, underscoring the flexibility of our approach. Notably, many of these methods assume independent parameter contributions, which can limit task sensitivity. In contrast, our Kronecker-factored *approximation of the full observed FIM captures both row- and column-wise dependencies within weight matrices*, yielding more accurate importance estimates. Separately, task-unaware low-rank methods focus on minimizing truncation error without leveraging global structure. AdaSVD (18) distributes compression strength across layers via adaptive compensation, while ASVD (30), NSVD (20), and SVD-LLM variants (28; 27) use activation statistics to refine truncation. These methods modulate projections using layer-specific signals like activation norms, distributions, or covariances. Although our method is task-aware and structure-driven, it is potentially compatible with these activation-based refinements. Integrating such signals, as in KFAC-like schemes, is a promising direction we leave for future work. In this paper, we focus on a clean evaluation setting to isolate and highlight the core contributions of GFWSVD.

3 Background and Problem Formulation

In this section, we establish the connection between Fisher information over matrix variables drawn from Matrix-Variate Normal (MVN) distribution and our approach to approximating the Fisher matrix via a Kronecker product decomposition. We then leverage this decomposition to develop an improved compression algorithm based on the generalized SVD formulation.

3.1 Layer Compression and Hessian Approximation

Consider post-training weight compression as a perturbation of a model parameters $\theta \in \mathbb{R}^d$. The perturbation affects the deviation of the model's loss function $\mathcal{L}(\theta)$ in the proximity of an optimal point $\theta^\star$. Sensitivity to such perturbation can be naturally captured by the second-order expansion of the loss determined by the quadratic term involving the Hessian $H = H(\theta^\star)$ of the problem:

$$\nabla\mathcal{L} = \mathcal{L}(\theta) - \mathcal{L}(\theta^*) \approx \frac{1}{2}(\theta - \theta^\star)^\top H (\theta - \theta^\star) \tag{1}$$

Compression optimization thus corresponds to minimizing the deviation $\nabla\mathcal{L}$ with respect to a compression $\theta = \mathcal{C}(\theta^\star)$ while considering the structured curvature encoded in H:

$$\min_{\mathcal{C}} \ (\theta^\star - \mathcal{C}\left(\theta^\star\right))^\top H \left(\theta^\star - \mathcal{C}\left(\theta^\star\right)\right), \tag{2}$$

where the optimization task is considered over a functional family of compression methods $\mathcal{C}$.

In real-world settings, working directly with H is often intractable due to its size and complex structure. Hence, solving the task in Eq. 2 also requires finding good enough approximations of H that ideally capture the most important properties of the Hessian. As we show next, there is a certain class of approximations that align particularly well with our task.

3.2 Matrix-Variate Normal Distribution and Fisher Information

The Matrix-Variate Normal (MVN) distribution (10) extends the classical multivariate normal distribution to matrix-valued random variables, providing a structured approach to modeling dependencies within rows and columns. Formally, a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$ follows an MVN distribution if its entries exhibit Gaussian properties with covariance structured across both dimensions. The distribution is defined as

$$\mathbf{X} \sim \mathcal{MN}(\mathbf{M}, \mathbf{\Sigma}_1, \mathbf{\Sigma}_2), \tag{3}$$

where $\mathbf{M}$ is the mean matrix, and the (non-degenerate) covariance is expressed as a Kronecker product $\mathbf{\Sigma}_2 \otimes \mathbf{\Sigma}_1$. Here, $\mathbf{\Sigma}_1$ captures dependencies between rows, while $\mathbf{\Sigma}_2$ encodes dependencies across columns. This structure ensures that each row and column follows a well-defined correlated Gaussian distribution.

A crucial property of MVN is that its likelihood function inherently incorporates the inverse Kronecker-factored covariance, *leading to an efficient representation of second-order dependencies*. The log-probability density function of $\mathbf{X}$ has the form:

$$\begin{aligned}\log(p(\mathbf{X})) \propto -\frac{1}{2}\left(\mathrm{vec}\left(\mathbf{X}-\mathbf{M}\right)^\top \left(\mathbf{\Sigma}_2 \otimes \mathbf{\Sigma}_1\right)^{-1} \mathrm{vec}\left(\mathbf{X}-\mathbf{M}\right)\right) = \\ = -\frac{1}{2}\,\mathrm{tr}\left(\mathbf{\Sigma}_1^{-1}\left(\mathbf{X}-\mathbf{M}\right)\mathbf{\Sigma}_2^{-1}\left(\mathbf{X}-\mathbf{M}\right)^\top\right)\end{aligned} \tag{4}$$

Maximization of log-likelihood leads to minimization of trace in Eq. 4, which yields the Generalized Least Squares Matrix Decomposition problem (3):

$$\min_{\mathrm{rank}(\mathbf{X})\le r} \left\| \mathbf{\Sigma}_1^{-\frac{1}{2}}(\mathbf{X}-\mathbf{M})\mathbf{\Sigma}_2^{-\frac{1}{2}} \right\|_{\mathrm{F}}^2, \tag{5}$$

directly connected to the Generalized Singular Value Decomposition (GSVD) (8). This problem can be straightforwardly solved by means of standard SVD (1):

$$\mathbf{X} = \mathbf{\Sigma}_1^{\frac{1}{2}} \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^\top \mathbf{\Sigma}_2^{\frac{1}{2}} \tag{6}$$

where $\hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^\top = \mathrm{SVD}_r(\mathbf{\Sigma}_1^{-\frac{1}{2}}\mathbf{M}\mathbf{\Sigma}_2^{-\frac{1}{2}})$. We note that the result also holds in the case when matrix square roots are replaced with the corresponding Cholesky factors, which are typically easier to find.

Under regular conditions (e.g., smooth differentiability and proper statistical properties), Fisher Information $\mathcal{I}_F$ serves as an expectation of the local curvature (second derivative) of the likelihood function. Importantly, by taking derivatives of the MVN likelihood function with respect to $\mathbf{M}$, it is easy to show that the corresponding *Hessian directly coincides with Fisher Information at the MLE solution*, e.i., $\mathcal{I}_F = H(\mathbf{M}) = \mathbf{\Sigma}_2^{-1} \otimes \mathbf{\Sigma}_1^{-1}$. This formulation provides a natural bridge between the selection of an optimal compression algorithm $\mathcal{C}$ from Eq. 2 and Fisher Information, which we establish next.

3.3 Fisher-Weighted Linear Layer Compression

Building on the established connection between MVN distributions and Fisher Information, we are now ready to formulate the rank-r linear layer compression theorem.

Theorem 1. *Let $\mathbf{W} \in \mathbb{R}^{n\times m}$ represent some parameter weights matrix of a single-layer linear neural network. Suppose that the following conditions hold.*

1. *The task loss function is derived from an MLE problem.*
2. *The (non-degenerate) empirical Fisher Information has a Kronecker product structure $\mathcal{I}_F = \mathbf{A} \otimes \mathbf{B}$.*
3. *The weights $\mathbf{W}$ are drawn from the MVN distribution $\mathcal{MN}(\mathbf{W}^\star, \mathbf{B}^{-1}, \mathbf{A}^{-1})$, where $\mathbf{W}^\star$ is the optimal weights matrix.*

Under these conditions, the best rank-r approximation that minimizes the expected increase in the loss after low-rank decomposition of $\mathbf{W}^\star$ is given by:

$$\widehat{\mathbf{W}}_r = \mathbf{L}_\mathbf{B}^{-\top}\, \widetilde{\mathbf{W}}_r\, \mathbf{L}_\mathbf{A}^{-1}, \tag{7}$$

where $\mathbf{A} = \mathbf{L}_\mathbf{A}\mathbf{L}_\mathbf{A}^\top$ and $\mathbf{B} = \mathbf{L}_\mathbf{B}\mathbf{L}_\mathbf{B}^\top$ are Cholesky factorizations, $\widetilde{\mathbf{W}} = \mathbf{L}_\mathbf{B}^\top \mathbf{W}^\star\, \mathbf{L}_\mathbf{A}$ is an auxiliary matrix, $\widetilde{\mathbf{W}}_r$ is the truncated SVD of $\widetilde{\mathbf{W}}$ of rank r.

Proof. Under the assumption that the loss function originates from MLE, the Hessian coincides with Fisher Information at the optimal point, ensuring structured sensitivity encoding. Hence, one can replace Eq. 2 with a surrogate problem

$$\min_{\mathcal{C}}\ (\theta^\star - \mathcal{C}\,(\theta^\star))^\top\, \mathcal{I}_F\, (\theta^\star - \mathcal{C}\,(\theta^\star)) \tag{8}$$

for $\mathrm{vec}(\mathbf{W}^\star) = \theta^\star$ and $\mathrm{vec}(\mathbf{W}) = \mathcal{C}(\theta^\star)$.

Substituting $\mathcal{I}_F$ with $\mathbf{A} \otimes \mathbf{B}$ and applying Cholesky decomposition to factors $\mathbf{A}$ and $\mathbf{B}$ yields:

$$\begin{aligned}
&\mathrm{vec}(\mathbf{W}^\star - \mathbf{W})^\top (\mathbf{L_A}\mathbf{L_A}^\top \otimes \mathbf{L_B}\mathbf{L_B}^\top)\mathrm{vec}(\mathbf{W}^\star - \mathbf{W}) \\
&= \mathrm{vec}(\mathbf{W}^\star - \mathbf{W})^\top (\mathbf{L_A} \otimes \mathbf{L_B})(\mathbf{L_A}^\top \otimes \mathbf{L_B}^\top)\mathrm{vec}(\mathbf{W}^\star - \mathbf{W}) \\
&= \mathrm{vec}(\mathbf{L_B}^\top(\mathbf{W}^\star - \mathbf{W})\mathbf{L_A})^\top \mathrm{vec}(\mathbf{L_B}^\top(\mathbf{W}^\star - \mathbf{W})\mathbf{L_A}) \\
&= \left\| \mathbf{L_B}^\top\, (\mathbf{W}^\star - \mathbf{W})\, \mathbf{L_A} \right\|_\mathrm{F}^2
\end{aligned} \tag{9}$$

In Section 3.2, we established that the optimal solution to this problem can be obtained via the standard SVD of the auxiliary matrix $\widetilde{\mathbf{W}}$. The final solution is found in two steps: 1) finding an optimal rank-r solution to the auxiliary problem $\widetilde{\mathbf{W}}_r = \mathrm{SVD}_r(\mathbf{L_B}^\top \mathbf{W}^\star \mathbf{L_A})$, and 2) recovering the optimal solution to the original problem through the inverse transformation $\widehat{\mathbf{W}}_r = \mathbf{L}_\mathbf{B}^{-\top}\, \widetilde{\mathbf{W}}_r\, \mathbf{L}_\mathbf{A}^{-1}$, which yields the best rank-r minimizer for Eq. 9. Consequently, the decomposition $\widehat{\mathbf{W}}_r$ presents an optimal compression $\mathcal{C}$ for Eq. 8, which in turn yields the minimal error increase in Eq. 1 for the given task defined by Eq. 2. ■

Linear layer factorization in this case can be computed with the following expressions:

$$\mathbf{W}_1 = \sqrt{\hat{\mathbf{S}}_r}\hat{\mathbf{V}}_r^\top \mathbf{L}_\mathbf{A}^{-1} \in \mathbb{R}^{r\times m}, \mathbf{W}_2 = \mathbf{L}_\mathbf{B}^{-\top}\hat{\mathbf{U}}_r\sqrt{\hat{\mathbf{S}}_r} \in \mathbb{R}^{n\times r}, \tag{10}$$

where $\hat{\mathbf{S}}_r$ is the diagonal matrix of the r leading singular values of the auxiliary problem.

3.4 Relationship to Prior Works

We show that FWSVD, presented in (12), is a special case of our generalized framework. The full justification is given in Appendix A. In FWSVD, the objective minimizes a weighted reconstruction error using a diagonal matrix $\mathbf{D}$ derived from a row-wise sum of the Fisher Information. We show that this setup corresponds to a diagonal Kronecker-factored approximation of the FIM, where $\mathbf{D}$ arises naturally from minimizing the Kronecker approximation error. The resulting solution for the low-rank factors $\mathbf{W}_2, \mathbf{W}_1$ matches that of FWSVD (up to a constant), which shows that their method is a special case of our more general framework.

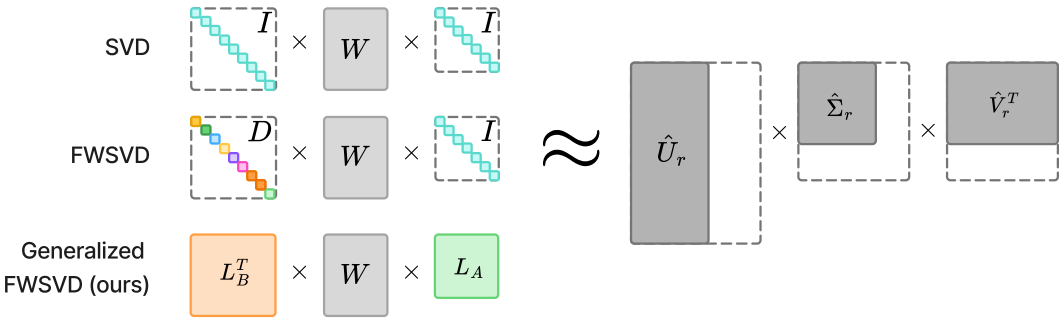

Figure 1: Generalization of the Weighted SVD frameworks. For standard SVD, the transformation matrices are identity matrices. For FWSVD, the left matrix is diagonal but not identity, and the right matrix is identity. For GFWSVD, both matrices are non-diagonal.

The connection between our generalized approach, the classical SVD and FWSVD is depicted in Figure 1. Weighted SVD approaches can be interpreted as transforming the decomposed object—here, the weight matrix—into a new space where the low-rank approximation better aligns with the target task. In this formulation, the sensitivity matrices serve as transformation matrices that reweight the importance of different directions. Under this view, vanilla SVD corresponds to using identity transformations; FWSVD applies a diagonal (but non-identity) transformation on one side while keeping the other side as identity. In contrast, our method employs full, non-diagonal transformations on both sides, capturing richer structure in the parameter space.

4 Kronecker Factorization Algorithm via Rank-1 SVD

Suppose that we have a linear layer of a network with a weight matrix $\mathbf{W}$ and define $\mathbf{G}_i \in \mathbb{R}^{n\times m}$ as a weight gradients $\mathcal{L}(\theta)|_{\theta=\mathbf{W}}$ on the i-th batch, and $g_i = \mathrm{vec}(\mathbf{G}_i) \in \mathbb{R}^{n\cdot m}$ - its flattening version. Then, Fisher Information $\mathcal{I}_F(\theta)$ can be defined as an empirical mean over all batches in a dataset D:

$$\mathcal{I}_F(\theta^\star) = \mathbb{E}\left[gg^\top\right] = \frac{1}{|D|}\sum_{i=1}^{|D|} g_i g_i^\top. \tag{11}$$

Kronecker product approximation is obtained by solving minimization problem:

$$\min \|\mathcal{I}_F - \mathbf{A}\otimes\mathbf{B}\|_{\mathrm{F}} \tag{12}$$

Kronecker product decomposition of $\mathcal{I}_F$ is computed from a rank-1 approximation of permuted matrix $\tilde{\mathcal{I}}_F = \mathcal{R}\mathcal{I}_F \in \mathbb{R}^{m^2\times n^2}$, as it is described in (19). The pseudocode of the method is in Algorithm 1, the code is available on anonymous repository: link.

Algorithm 1 Compute Kronecker Factors via Rank-1 SVD

Require: List of gradients $\{g_i\}_{i=1}^{|D|}$, $|D|$ – number of batches

$\mathcal{I}_F \leftarrow \frac{1}{|D|}\sum_{i=1}^{|D|} g_i g_i^T$

$\tilde{\mathcal{I}}_F \leftarrow \mathcal{R}\mathcal{I}_F \leftarrow \frac{1}{|D|}\sum_{i=1}^{|D|} \mathbf{G_i}\otimes\mathbf{G_i}$

$(u, \sigma, v^\top) \leftarrow$ Leading singular triplet ▷ Truncated SVD

$b \leftarrow u\cdot\sigma$ ▷ $b = \mathrm{vec}(\mathbf{B})$

$a \leftarrow v$ ▷ $a = \mathrm{vec}(\mathbf{A})$

$\mathbf{B} \leftarrow \mathrm{reshape}(b, (m, m))$

$\mathbf{A} \leftarrow \mathrm{reshape}(a, (n, n))$

return $(\mathbf{B}, \mathbf{A})$

4.1 Efficient Rank-1 Computation

The primary computational bottleneck of Algorithm arises in performing SVD on the matrix $\tilde{\mathcal{I}}_F$.

Standard SVD is computationally intractable for large matrices, so we employ truncated SVD using the Lanczos (16) method, which avoids explicit matrix construction and requires only the ability to multiply the matrix with a vector from the left or right. Even in this setting, aggregating the full second-moment gradient information across all batch samples is computationally expensive.

We can show (see Appendix B) that permuted $\mathcal{I}_F$ for i-th batch can be defined as the Kronecker product of the corresponding gradient matrices:

$$\tilde{\mathcal{I}}_F = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathbf{G}_i \otimes \mathbf{G}_i, \tag{13}$$

then multiplication of this matrix $\tilde{\mathcal{I}}_F$ to a vector z from left will be:

$$\tilde{\mathcal{I}}_F z = \frac{1}{k} \left(\sum_{i=1}^{k} \mathbf{G}_i \otimes \mathbf{G}_i \right) z = \frac{1}{|D|} \left(\sum_{i=1}^{|D|} \mathbf{G}_i \otimes \mathbf{G}_i \right) \mathbf{Z} = z, \text{ where } z = \mathrm{vec}(\mathbf{Z}), \mathbf{Z} \in \mathbb{R}^{n \times n} \tag{14}$$

Using property of the Kronecker product $(\mathbf{K} \otimes \mathbf{L}) \,\mathrm{vec}\left(\mathbf{C}\right) = \mathrm{vec}(\mathbf{K}^\top \mathbf{C}\mathbf{L})$ we reduce the matrix-vector multiplication to a sequence of matrix multiplications:

$$\tilde{\mathcal{I}}_F z = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathrm{vec}(\mathbf{G}_i^\top \mathbf{Z} \mathbf{G}_i) \tag{15}$$

The derivation for right-side multiplication is analogous (see Appendix C). Using these operations, we can obtain an approximation of the Fisher information for layers of LLMs and batch sizes used in practice within a reasonable time.

4.2 Time Complexity of the proposed Rank-1 Computation

The time complexity of computing the truncated SVD of the matrix $\tilde{\mathbf{J}} \in \mathbb{R}^{m^2 \times n^2}$ consists of the matrix-vector multiplications and the orthogonalization and has a cost of $\mathcal{O}\left(m^2 n^2\right)$. However, using the structured formulation from Eq. 15, where left matrix-vector products are implemented via multiplications with matrices $\mathbf{G}_i^\top \in \mathbb{R}^{m \times n}$, $\mathbf{Z} \in \mathbb{R}^{n \times n}$, and $\mathbf{G}_i \in \mathbb{R}^{n \times m}$, the overall complexity is reduced to $\mathcal{O}\left(mn^2 + m^2 n\right)$. Applying analogous reasoning to the right matrix-vector products (see Eq. 30) one can yield the same complexity.

Although the proposed method exhibits cubic complexity with respect to the dimensions of the linear layer's weight matrix, its empirical runtime grows more slowly than that of the standard matrix-vector product, which scales quartically. In large language models, where m and n are typically on the order of 10^3, this reduction yields a practically significant speedup. Table 1 reports the empirical decomposition times for $\tilde{\mathcal{I}}_F$, corresponding for matrices $\mathbf{W}$ in different LLMs.

Table 1: Runtime for computing Kronecker factors of single matrices on GPU.

Layer	Params	Time (s)
BERT	2.3M	43
LLaMA-2-7B	45M	183
LLaMA-2-13B	70.8M	313

5 Numerical Experiments

To validate our theoretical contributions, we conduct extensive numerical experiments on several transformer architectures: the encoder-only BERT model (4) and the decoder-only LLM LLaMA 2 (26). Our goal is to demonstrate the practical benefits of GFWSVD in low-rank compression under fine-tuning and evaluation protocols. We conduct all experiments on a single NVIDIA A100 GPU with latest CUDA drivers using Python 3.12. The code is available on anonymous repository: link.

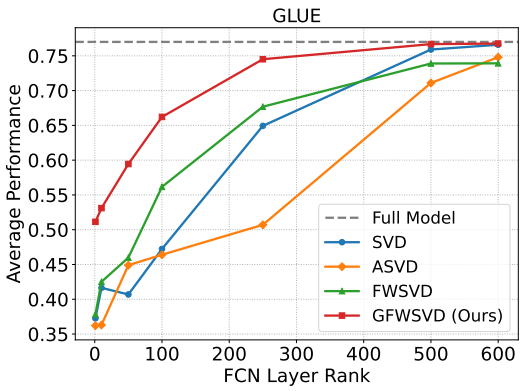

Figure 2: Macro-averaged GLUE performance of `bert-base-uncased` model for different compression ranks.

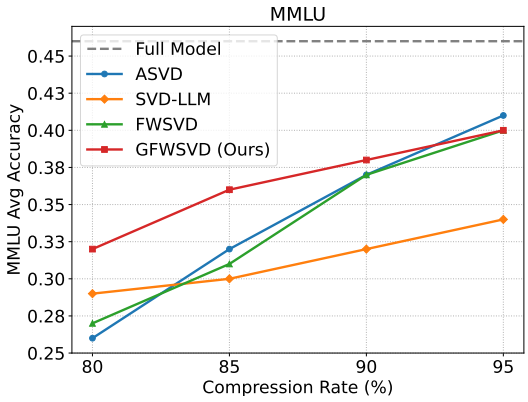

Figure 3: Average MMLU performance of `llama-2-7b-chat` model for different compression rates.

5.1 Compressing the Transformer Encoder

In our experiments, we follow the "fine-tune then compress" pipeline, similar to FWSVD (12). We begin by fine-tuning a pre-trained checkpoint[1] of the BERT-base model on a specific downstream GLUE task. Optimal fine-tuning hyperparameters (e.g., learning rate, batch size) are selected for each task using the Optuna framework (2). During this stage, we also collect gradients to construct the FIM $\mathcal{I}_F$ and compute its Kronecker decomposition as described in Section 4.

Using the resulting Cholesky factors $\mathbf{L_A}$ and $\mathbf{L_B}$, we uniformly compress the fully connected layers of BERT by factorizing them into two smaller layers, following the method detailed in Section 3.1. The chosen layer-wise ranks and the resulting overall compression rate of the model are summarized in Table 2. We reproduce the ASVD method using the original authors' code. For FWSVD, we incorporate the newly constructed FIM into the compression process.

Table 2: The correspondence between rank and entire BERT compression rate.

Rank	C.Rate	Rank	C.Rate
1	$\sim 40\%$	100	$\sim 33\%$
5	$\sim 40\%$	250	$\sim 23\%$
10	$\sim 39\%$	500	$\sim 8\%$
50	$\sim 36\%$	600	$\sim 1\%$

We show average compression results in Table 3 and Figure 2, extended results are in Appendix E in Table 8. On most of the GLUE tasks and considered compression ranks, our proposed GFWSVD approach consistently outperforms both FWSVD and SVD, with particularly strong gains at lower ranks. While ASVD exhibits relatively poor performance on several tasks (QQP, QNLI), it occasionally surpasses GFWSVD — notably on SST2 under aggressive compression.

Table 3: Macro-averaged GLUE performance of the `bert-base-uncased` for different compression ranks. Best results for each rank are in **bold**.

Method / Rank	600	500	250	100	50	10	1
SVD	**0.77**	0.76	0.65	0.47	0.41	0.42	0.37
ASVD (30)	0.75	0.71	0.51	0.46	0.45	0.36	0.36
FWSVD (12)	0.74	0.74	0.68	0.56	0.46	0.43	0.38
GFWSVD (Ours)	**0.77**	**0.77**	**0.75**	**0.66**	**0.59**	**0.53**	**0.51**

5.2 Compressing the Transformer Decoder

We evaluate our approach on the decoder-only LLama 2 7B model[2] against several competitive baselines: diagonal FI-based low-rank approximation method FWSVD (12), and two activation-based methods – ASVD (30) and SVD-LLM(28). Notably, ASVD and SVD-LLM both rely on activation-based weighting to gauge parameter importance, while FWSVD and ours GFWSVD derive importance scores solely from gradient information.

[1] `https://huggingface.co/google-bert/bert-base-uncased`
[2] `https://huggingface.co/unsloth/llama-2-7b-chat`

We measure perplexity on WikiText 2 (22) and PTB (21) datasets, and 5-shot reasoning performance on the MMLU benchmark (11). Following prior works on low-rank approximation of LLMs (28; 30), we test several compression setups, removing from 5% to 20% of original parameters.

Following standard practice in post-training LLM compression methods (28; 30), we use a randomly sampled set of sentences as calibration data to generate gradients for further obtaining the factor matrices. For calibration data, we choose the FineWeb dataset (23) due to its high quality and diversity, and collect gradients on a random subsample of size 1024. These gradients are then used to obtain $\mathbf{L_A}$ and $\mathbf{L_B}$, as well as the data needed for FWSVD. As in LLMs, uniform layer compression can disproportionately degrade performance by over-compressing critical layers and under-utilizing redundancy in less sensitive ones, so it is essential for each method to use a compression configuration that accounts for layer sensitivity. For both ASVD and SVD-LLM, we used the corresponding code released by the authors and re-ran the necessary compression pipelines for our checkpoint with all hyperparameters set to default values. For our approach, we adopted the method of per-layer importance scores as described in the ASVD work.

Table 4: Performance of the `unsloth/llama-2-7b-chat` compressed by various methods under compression ratios from 5% to 20% on WikiText-2, PTB, and MMLU. Lower is better for perplexity (↓), higher is better for accuracy (↑).

METHOD	WikiText-2↓	PTB↓	Compr.	MMLU Avg↑	Humanities↑	Other↑	Social Sciences↑	STEM↑
Full model	**6.94**	**25.75**	100%	0.46 ± 0.003	0.43 ± 0.01	0.55 ± 0.01	0.53 ± 0.01	0.36 ± 0.01
FWSVD (12)	7.52	45.25	95%	0.40 ± 0.003	0.36 ± 0.01	0.45 ± 0.01	0.45 ± 0.01	0.35 ± 0.01
ASVD (30)	7.60	**26.29**		**0.41** ± 0.004	0.37 ± 0.01	**0.48** ± 0.01	**0.46** ± 0.01	**0.35** ± 0.01
SVD-LLM (28)	8.80	51.28		0.34 ± 0.004	0.31 ± 0.01	0.38 ± 0.01	0.35 ± 0.01	0.31 ± 0.01
GFWSVD (Ours)	**7.16**	28.55		0.40 ± 0.003	**0.38** ± 0.01	0.47 ± 0.01	0.44 ± 0.01	0.33 ± 0.01
FWSVD (12)	11.53	96.62	90%	0.37 ± 0.004	0.34 ± 0.01	0.43 ± 0.01	0.42 ± 0.01	0.33 ± 0.01
ASVD (30)	8.97	40.12		0.37 ± 0.004	0.33 ± 0.01	0.42 ± 0.01	0.40 ± 0.01	0.33 ± 0.01
SVD-LLM (28)	9.69	60.82		0.32 ± 0.004	0.30 ± 0.01	0.35 ± 0.01	0.32 ± 0.01	0.30 ± 0.01
GFWSVD (Ours)	**8.77**	**36.44**		**0.38** ± 0.002	**0.35** ± 0.01	**0.44** ± 0.01	**0.42** ± 0.01	**0.33** ± 0.01
FWSVD (12)	22.06	411.50	85%	0.31 ± 0.009	0.29 ± 0.01	0.34 ± 0.01	0.33 ± 0.01	0.30 ± 0.01
ASVD (30)	10.91	83.49		0.32 ± 0.003	0.30 ± 0.01	0.33 ± 0.01	0.32 ± 0.01	0.30 ± 0.01
SVD-LLM (28)	10.36	72.58		0.30 ± 0.004	0.29 ± 0.01	0.34 ± 0.01	0.31 ± 0.01	0.30 ± 0.01
GFWSVD (Ours)	**10.06**	**42.19**		**0.36** ± 0.004	**0.33** ± 0.01	**0.41** ± 0.01	**0.38** ± 0.01	**0.32** ± 0.01
FWSVD (12)	66.37	1523.00	80%	0.27 ± 0.004	0.25 ± 0.01	0.30 ± 0.01	0.28 ± 0.01	0.28 ± 0.01
ASVD (30)	27.73	241.57		0.26 ± 0.004	0.25 ± 0.01	0.27 ± 0.01	0.24 ± 0.01	0.28 ± 0.01
SVD-LLM (28)	11.23	98.91		0.29 ± 0.004	0.27 ± 0.01	0.32 ± 0.01	0.29 ± 0.01	0.29 ± 0.01
GFWSVD (Ours)	**11.13**	**50.50**		**0.32** ± 0.003	**0.30** ± 0.01	**0.35** ± 0.01	**0.34** ± 0.01	**0.30** ± 0.01

Table 4 and Figure 3 shows that GFWSVD consistently outperforms both simple and strong baselines across all compression rates. In particular, at the most aggressive settings (15–20% of the original parameters), our method matches or exceeds the accuracy of activation-based methods and shows substantially lower perplexities on both WikiText-2 and PTB.

For the entire LLM, layers are processed independently and factor computation can be parallelized, so the runtime scales as:

$$\text{Total time} = \frac{\text{time per layer} \times \text{number of layers}}{\text{number of workers}}.$$

For example, LLaMA-2-7B has 224 linear layers; with 4 GPUs, computing all Kronecker factors takes approximately 3.5 hours. Memory constraints arise only during the calibration phase, when gradients for selected layers must fit in memory. However, this is not a fundamental limitation: layers can be calibrated sequentially by freezing and unfreezing them one at a time.

Throughput and FLOP results for compressed LLaMA models are provided in Appendix D.

In the scope of this work, we focused on a simple, analytically motivated form of post-training compression, which was necessary to isolate the contribution of the non-diagonal elements of the assumed parameter distribution in the layer weight matrices. As with classical SVD, the procedure enables straightforward integration with complementary techniques that increase expressivity (e.g., post-training quantization or stochastic optimization of compression parameters). In such cases,

GFWSVD can serve as a convenient initialization point on the loss surface for subsequent post-training compression techniques.

5.3 Impact of Diagonal and Non-diagonal Elements of Factors

To assess the significance of the diagonal elements, we performed the following ablation study. In the resulting factor matrices we retained either (1) only the off-diagonal elements (**Non-diag**) or (2) only the diagonal elements (**Diag**), and measured perplexity relative to our method and FWSVD. The **Diag** variant performs better than FWSVD but worse than GFWSVD. This is expected, since FWSVD captures importance only along rows (only the left factor matrix has a non-identity diagonal, see Fig. 1), whereas Non-diag GFWSVD captures both row and column importance. The contribution of off-diagonal elements provides a noticeable improvement compared to FWSVD.

Method	Wiki (10%)	PTP (10%)	Wiki (15%)	PTP (15%)
FWSVD	11.53	96.62	22.00	411.00
Diag GFWSVD	10.94	45.26	11.06	48.25
Non-diag GFWSVD	8.85	37.25	10.22	43.75
Full GFWSVD	8.77	36.44	10.06	42.19

Table 5: Perplexity at 10% and 15% compression.

6 Conclusion and Future Work

We introduced **Generalized Fisher-Weighted SVD (GFWSVD)**, a low-rank second-order compression method that leverages the full Fisher Information Matrix through a scalable Kronecker decomposition. Unlike previous approaches, GFWSVD captures parameter correlations and yields a factorization *provably optimal* within its class (Theorem 1). Our results on both encoder-only (BERT on GLUE) and decoder-only (LLaMA-2 on MMLU) models show that GFWSVD consistently outperforms diagonal Fisher- and activation-based SVD approaches, particularly at low ranks.

Crucially, the method is entirely analytical and does *not* require stochastic optimization or iterative retraining, making it lightweight and reproducible. The tractable algorithm for computing full Kronecker factors makes this work an important step toward practical, curvature-aware post-training compression of large language models.

GFWSVD highlights the critical role of accurate FIM computation in compression. While our approach performs well empirically, its reliance on a rank-1 Kronecker approximation of the Fisher matrix may oversimplify important structure. Future work could explore higher-rank Kronecker series to capture richer information, and extend the method to model cross-layer dependencies, potentially improving performance by leveraging transitive correlations across the network.

7 Limitations

Our method decomposes the observed Fisher information matrix $\mathcal{I}_F$ into a Kronecker product of two smaller matrices, $\mathbf{Y}$ and $\mathbf{X}$ (Eq. 12). While effective, this assumes exact factorization, which may not hold in practice and can limit approximation quality and task sensitivity. In LLM experiments, we also observed cases where the estimated Kronecker factors were singular, requiring regularization (e.g., $\mathbf{Y} \leftarrow \mathbf{Y} + \alpha \operatorname{diag} \mathbf{Y}$) to ensure positive definiteness and numerical stability. Although this resolves instability, it introduces additional computational overhead.

We observed that compression effectiveness varies significantly across layers, making preliminary layer selection necessary to achieve favorable trade-offs. A key limitation of our current approach is the lack of coordination across layers during compression. For effective multi – layer compression—especially in large-scale models like LLMs – it is important to account for cross-layer dependencies. Future work could focus on modeling these interactions to enable joint compression strategies.

ETHICS STATEMENT

This work focuses on methods for improving the efficiency and practicality of post-training low-rank compression of large language models using second-order information. Our research does not involve human subjects, personally identifiable information, or other sensitive data. All experiments are carried out on publicly available models (BERT, LLaMA) and widely used benchmarks (GLUE, MMLU), ensuring transparency and reproducibility. We do not release any new datasets containing private or proprietary information. The proposed methods are intended to reduce the computational cost and energy consumption of deploying large models, which we view as a positive contribution to sustainability. We are not aware of any direct negative societal impacts; however, as with any model compression technique, improved efficiency may lower the barrier to deploying large models in contexts where misuse is possible. We therefore encourage responsible use of these methods in accordance with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure reproducibility of our results. The full description of the proposed method, including theoretical assumptions and proofs, is provided in the main text and Appendix. All implementation details of Algorithm 1 as well as experimental pipelines are available in an anonymous repository: link.

REFERENCES

[1] Hervé Abdi. Singular value decomposition (svd) and generalized singular value decomposition. *Encyclopedia of measurement and statistics*, 907(912):44, 2007.

[2] Takuya Akiba, Shotaro Sano, Takeru Yanase, Toshihiko Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631. ACM, 2019.

[3] Genevera I Allen, Logan Grosenick, and Jonathan Taylor. A generalized least-square matrix decomposition. *Journal of the American Statistical Association*, 109(505):145–159, 2014.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[5] R. A. Fisher. *On the Mathematical Foundations of Theoretical Statistics*, pages 11–44. Springer New York, New York, NY, 1992.

[6] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[7] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR, 2023.

[8] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[9] Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

[10] Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

[11] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[12] Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization, 2022.

[13] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedfisher: Leveraging fisher information for one-shot federated learning. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pages 1612–1620. PMLR, 2024.

[14] James Kirkpatrick, Razvan Pascanu, et al. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the National Academy of Sciences*, volume 114, pages 3521–3526, 2017.

[15] Abdoulaye Koroko, Ani Anciaux-Sedrakian, Ibtihel Ben Gharbia, Valérie Garès, Mounir Haddou, and Quang Huy Tran. Efficient approximations of the fisher matrix in neural networks using kronecker product singular value decomposition. *ESAIM: Proceedings and Surveys*, 73:218–237, 2023.

[16] Cornelius Lanczos. 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.

[17] Sanwoo Lee, Jiahao Liu, Qifan Wang, Jingang Wang, Xunliang Cai, and Yunfang Wu. Dynamic fisher-weighted model merging via Bayesian optimization. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4923–4935, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[18] Zhiteng Li, Mingyuan Xia, Jingyuan Zhang, Zheng Hui, Linghe Kong, Yulun Zhang, and Xiaokang Yang. Adasvd: Adaptive singular value decomposition for large language models. *arXiv e-prints*, pages arXiv–2502, 2025.

[19] Charles Van Loan and Nikos Pitsianis. Approximation with kronecker products. 1992.

[20] Jun Lu, Tianyi Xu, Bill Ding, David Li, and Yu Kang. Large language model compression via the nested activation-aware decomposition. *arXiv preprint arXiv:2503.17101*, 2025.

[21] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.

[22] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[23] Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[24] Alexander Soen and Ke Sun. Trade-offs of diagonal fisher information matrix estimators. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[25] Zedong Tang, Fenlong Jiang, Maoguo Gong, Hao Li, Yue Wu, Fan Yu, Zidong Wang, and Min Wang. Skfac: Training neural networks with faster kronecker-factored approximate curvature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13479–13487, 2021.

[26] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[27] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression. *arXiv preprint arXiv:2503.12340*, 2025.

[28] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025.

[29] Xiaodong Wu, Wenyi Yu, Chao Zhang, and Philip C. Woodland. An improved empirical fisher approximation for natural gradient descent. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[30] Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: activation-aware singular value decomposition for compressing large language models. *CoRR*, abs/2312.05821, 2023.

A Appendix A: Special case of diagonal Fisher Information Matrix

In this section, we show that FWSVD, presented in (12), is a special case of our generalized approach.

In the work of (12), authors propose to minimize the following objective:

$$\min_{\mathbf{W_1},\mathbf{W_2}} \|\mathbf{D}\mathbf{W}^\star - \mathbf{D}\mathbf{W_2}\mathbf{W_1}\|_F^2 \tag{16}$$

where $\mathbf{D}$ is the diagonal matrix $\sqrt{\operatorname{diag}\left(\mathbb{E}[\mathbf{G}\mathbf{G}^\top]\right)}$. Specifically, $\mathbf{D}_{i,i} = \sqrt{\sum_{j=1}^{m} \mathbb{E}(\mathbf{G}_{i,j})^2}$.

Similarly to 12, we approximate the Fisher Information with a Kronecker product of identity matrix $\mathbf{I}_m$ and some diagonal matrix $\tilde{\mathbf{D}}$. As described further in Section 4 and Appendix A, under the permutation $\mathcal{R}$, the problem

$$\min_{\mathbf{D}} \left\| \mathbf{I}_F - \mathbf{I}_m \otimes \tilde{\mathbf{D}} \right\|_{\mathrm{F}} \tag{17}$$

reduces to minimization of the expression

$$\min_{\mathbf{d}} \left\| \mathbb{E}[\mathbf{G} \otimes \mathbf{G}] - (\mathbf{I}_n \odot \mathbf{I}_n) d \cdot \operatorname{vec}(\mathbf{I}_m)^\top \right\|_{\mathrm{F}} \tag{18}$$

where $\odot$ is a Khatri-Rao product (column-wise Kronecher product) and $\cdot$ is a vector outer product; d is a vector diagonal of $\tilde{\mathbf{D}}$; $\mathbb{E}[\mathbf{G} \otimes \mathbf{G}]$ is a permuted Fisher Information matrix $\tilde{\mathbf{I}}_{\mathbf{F}}$, defined in Eq 13.

For simplicity, we will use a shorter notation. Let $\mathbf{E} = \mathbb{E}[\mathbf{G} \otimes \mathbf{G}]$, $\mathbf{Z} = \mathbf{I}_n \odot \mathbf{I}_n$, $v = \operatorname{vec}(\mathbf{I}_m)$. Then, the problem 18 is equivalent to

$$\min_{\mathbf{d}} \left\| \mathbf{Z} d \cdot v^\top - \mathbf{E} \right\|_{\mathrm{F}} \tag{19}$$

Applying first-order optimality conditions yields:

$$\begin{gathered} \langle \mathbf{Z}\delta d \cdot v^\top, \mathbf{Z} d \cdot v^\top - \mathbf{E} \rangle = 0 \\ \langle \delta d \cdot v^\top, \mathbf{Z}^\top \mathbf{Z} d \cdot v^\top - \mathbf{Z}^\top \mathbf{E} \rangle = 0 \\ \langle \delta d, \mathbf{Z}^\top \mathbf{Z} d \cdot v^\top v - \mathbf{Z}^\top \mathbf{E} v \rangle = 0 \end{gathered}$$

Since $\mathbf{Z}^\top \mathbf{Z} = \mathbf{I}_n$, $v^\top v = \|v\|_2^2 = \|\operatorname{vec}(\mathbf{I}_m)\|_2^2 = m$, we have:

$$d = \frac{1}{m}(\mathbf{I}_n \odot \mathbf{I}_n)^\top \mathbb{E}[\mathbf{G} \otimes \mathbf{G}] \operatorname{vec}(\mathbf{I}_m) = \frac{1}{m}(\mathbf{I}_n \odot \mathbf{I}_n)^\top \operatorname{vec}(\mathbb{E}[GG^\top]) = \frac{1}{m} \operatorname{diag}(\mathbb{E}[GG^\top]) \tag{20}$$

Thus, diagonal matrix $\tilde{\mathbf{D}}$ from Kronecker product approximation problem 17 equals square of matrix $\mathbf{D}$ from the FWSVD formulation 16 up to the constant $\frac{1}{m}$.

We apply Theorem 1 to find factors $\mathbf{W}_2$, $\mathbf{W}_1$ for the obtained approximation $\mathbf{I}_F = \mathbf{I}_m \otimes \tilde{\mathbf{D}}$:

$$\mathbf{W}_2 = \sqrt{\tilde{\mathbf{D}}}^{-1}\hat{\mathbf{U}}_r\sqrt{\hat{\mathbf{S}}_r} = \mathbf{D}^{-1}\hat{\mathbf{U}}_r\sqrt{\hat{\mathbf{S}}_r}, \mathbf{W}_1 = \sqrt{\hat{\mathbf{S}}_r}\hat{\mathbf{V}}_r^\top \tag{21}$$

where $\hat{\mathbf{U}}_r\hat{\mathbf{S}}_r\hat{\mathbf{V}}_r^\top$ is r-rank SVD of $\sqrt{\tilde{\mathbf{D}}}\mathbf{W}^\star = \mathbf{D}\mathbf{W}^\star$. This is the same solution that minimizes the problem 16 from FWSVD paper (12). Consequently, FWSVD approach is a special case of diagonal Kronecker product approximation of Fisher Information.

B Appendix B: Additional explanations for Kronecker decomposition adaptation

Let's show that the permuted $\mathcal{I}_F$ in the Kronecker decomposition algorithm can be expressed as the Kronecker product of the corresponding gradient matrices.

We start with the empirical Fisher information matrix defined as $\mathcal{I}_F = \frac{1}{|D|}\sum_{i=1}^{|D|} g_i g_i^\top$ and its reordered version:

$$\tilde{\mathcal{I}}_F = \mathcal{R}\mathcal{I}_F \tag{22}$$

Using the identity

$$\mathrm{vec}(\boldsymbol{g}_i\boldsymbol{g}_i^\top) = \boldsymbol{g}_i \otimes \boldsymbol{g}_i,$$

we obtain:

$$\mathrm{vec}(\mathcal{I}_F) = \frac{1}{|D|}\sum_{i=1}^{|D|}\mathrm{vec}(\boldsymbol{g}_i\boldsymbol{g}_i^\top) = \frac{1}{|D|}\sum_{i=1}^{|D|}(\boldsymbol{g}_i \otimes \boldsymbol{g}_i). \tag{23}$$

Let $\mathcal{P} \in \mathbb{R}^{(ab)^2\times(ab)^2}$ be the unique permutation matrix such that for any matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{a\times b}$:

$$\mathcal{P}\cdot\mathrm{vec}(\mathbf{A}\otimes\mathbf{B}) = (\mathrm{vec}(\mathbf{A})\otimes\mathrm{vec}(\mathbf{B})). \tag{24}$$

In our case $\mathcal{P}$ can be defined through the commutation matrix $\mathbf{K}_{mn}$ and identity matrices $\mathbf{I}_n$ and $\mathbf{I}_m$:

$$\mathbf{P} := \mathbf{I}_n \otimes \mathbf{K}_{mn} \otimes \mathbf{I}_m, \qquad \mathbf{K}_{mn}^\top = \mathbf{K}_{nm} \tag{25}$$

Using this , we can write:

$$\mathcal{P}\cdot\mathrm{vec}(\mathbf{G}_i\otimes\mathbf{G}_i) = \mathrm{vec}(\mathbf{G}_i)\otimes\mathrm{vec}(\mathbf{G}_i). \tag{26}$$

Therefore, the vectorized Fisher information becomes:

$$\mathrm{vec}(\mathcal{I}_F) = \frac{1}{|D|}\sum_{i=1}^{|D|}\mathcal{P}\cdot\mathrm{vec}(\mathbf{G}_i\otimes\mathbf{G}_i) = \mathcal{P}\cdot\mathrm{vec}\left(\frac{1}{|D|}\sum_{i=1}^{|D|}(\mathbf{G}_i\otimes\mathbf{G}_i)\right) = \mathcal{P}\,\mathrm{vec}(\tilde{\mathcal{I}}_F). \tag{27}$$

So, $\tilde{\mathcal{I}}_F$ can be defined as $\frac{1}{|D|}\sum_{i=1}^{|D|}(\mathbf{G}_i\otimes\mathbf{G}_i)$. This fact is used in the accelerated adaptation of the Kronecker Factorization algorithm.

Now, suppose a $\mathcal{I}_F$ and $\tilde{\mathcal{I}}_F$ are connected with $\mathcal{R} \in \mathbb{R}^{n\times n}$ (see Eq. 22):

$$\mathrm{vec}(\widetilde{\mathcal{I}}_F) = (I\otimes\mathcal{R})\cdot\mathrm{vec}(\mathcal{I}_F), \mathcal{P} = I\otimes\mathcal{R} \tag{28}$$

C Appendix C: Right vector-matrix multiplication

We can define right vector-matrix multiplication as follows:

$$\mathcal{I}_F^\top z = (\sum_{i=1}^{|D|} \mathbf{G}_i \otimes \mathbf{G}_i)^\top z \tag{29}$$

Using property of the Kronecker product $(\mathbf{K} \otimes \mathbf{L})\,\mathrm{vec}(\mathbf{C}) = \mathrm{vec}(\mathbf{K}^\top \mathbf{C}\mathbf{L})$:

$$\mathcal{I}_F^\top z = \sum_{i=1}^{|D|} \mathrm{vec}(\mathbf{G}_i \mathbf{Z} \mathbf{G}_i^\top), \text{ where } z = \mathrm{vec}(\mathbf{Z}), \mathbf{Z} \in \mathbb{R}^{m \times m} \tag{30}$$

D Appendix E: Throughput and FLOPs for compressed models

GFWSVD, ASVD, SVD-LLM compresses weight $\mathbf{W} \in \mathbb{R}^{n \times m}$ into a pair of low-rank matrices $\mathbf{W}_1 \in \mathbb{R}^{n \times r}$ and $\mathbf{W}_2 \in \mathbb{R}^{r \times m}$. This reduces the number of FLOPs required during the forward pass through a linear layer from $O(nm)$ to $O(nr + rm) = O(r(n + m))$.

Model (C Rate)	Full Model FLOPs	Compressed FLOPs
LLaMA-2-7B-chat (10%)	53.05T	42.43T
LLaMA-2-7B-chat (15%)	53.05T	39.24T
LLaMA-2-7B-chat (20%)	53.05T	37.18T

Table 6: Comparison of theoretical FLOPs for LLaMA-2-7B-chat under different compression rates. All values are in trillions (T) of FLOPs.

We ran inference-time latency measurements on the `LLaMA-2-7B-chat` model under different compressions. The results are shown below (averaged over 100 runs, batch size = 1, sequence length = 1024 tokens, GPU: A100 80GB).

Table 7: Throughput (tokens/s) achieved by original LLaMA-7B-chat and its FWSVD-compressed versions (batch size = 1, sequence length = 1024).

Compression Ratio	Tokens/s	Relative Speedup
0% (Uncompressed)	1186	1.00×
10%	1269	1.07×
15%	1294	1.09×
20%	1323	1.12×

E Appendix D: Extended GLUE results

We report extended compression results on tasks of GLUE benchmark in Table 8.

F Appendix F: LLM Usage Statement

We used large language models (LLMs) only as a general-purpose writing assistant for grammar checking and text polishing. The research ideas, implementation, analysis, and conclusions are entirely our own.

Table 8: Performance of `bert-base-uncased` compressed by various methods under compression rates from 60% to 99% on GLUE benchmark. Lower is better for COLA (↓), higher is better for all other tasks (↑).

METHOD / DATASET	MRPC↑	STSB↑	QQP↑	MNLI↑	QNLI↑	RTE↑	COLA↓	SST2↑
Full model	**0.77**	**0.87**	**0.90**	**0.83**	**0.90**	**0.56**	**0.41**	**0.91**
	Compression Rate 99% ($r = 600$)							
SVD	0.67	0.84	0.90	0.67	0.90	0.56	0.58	0.91
ASVD (30)	0.72	0.73	0.89	0.83	0.90	0.56	0.41	0.91
FWSVD (12)	0.72	0.87	0.90	0.72	0.90	0.55	0.36	0.91
GFWSVD (Ours)	**0.73**	**0.87**	**0.90**	**0.73**	**0.90**	**0.56**	**0.55**	**0.92**
	Compression Rate 92% ($r = 500$)							
SVD	0.53	0.82	0.89	0.53	0.90	0.54	0.53	0.89
ASVD (30)	0.71	0.56	0.86	0.81	0.89	0.53	0.44	0.88
FWSVD (12)	0.71	0.87	0.90	0.71	0.89	0.56	0.34	0.91
GFWSVD (Ours)	**0.73**	**0.87**	**0.90**	**0.73**	**0.90**	**0.56**	**0.49**	**0.92**
	Compression Rate 77% ($r = 250$)							
SVD	0.49	0.68	0.81	0.49	0.85	0.50	0.17	0.57
ASVD (30)	0.69	0.08	0.76	0.50	0.58	0.47	0.11	0.75
FWSVD (12)	0.69	0.86	0.89	0.69	0.89	0.61	0.23	0.80
GFWSVD (Ours)	**0.71**	**0.86**	**0.89**	**0.71**	**0.89**	**0.61**	**0.38**	**0.88**
	Compression Rate 67% ($r = 100$)							
SVD	0.32	0.08	0.64	0.32	0.80	0.51	0.01	0.49
ASVD (30)	0.58	0.07	0.74	0.39	0.50	0.47	0.05	0.82
FWSVD (12)	0.69	0.58	0.87	0.71	0.86	0.55	0.21	0.72
GFWSVD (Ours)	**0.71**	**0.70**	**0.87**	**0.71**	**0.86**	**0.55**	**0.21**	**0.72**
	Compression Rate 64% ($r = 50$)							
SVD	0.32	0.19	0.57	0.32	0.78	0.48	0.02	0.49
ASVD (30)	0.68	0.03	0.73	0.49	0.76	0.51	0.03	0.80
FWSVD (12)	0.69	0.65	0.84	0.69	0.72	0.46	0.03	0.77
GFWSVD (Ours)	**0.69**	**0.65**	**0.84**	**0.69**	**0.72**	**0.46**	**0.05**	**0.77**
	Compression Rate 61% ($r = 10$)							
SVD	0.32	0.32	0.67	0.32	0.61	0.51	0.00	0.49
ASVD (30)	0.61	0.14	0.64	0.40	0.57	0.49	-0.04	0.76
FWSVD (12)	0.37	0.32	0.79	0.37	0.57	0.49	0.00	0.49
GFWSVD (Ours)	**0.53**	**0.60**	**0.79**	**0.53**	**0.62**	**0.47**	**0.05**	**0.65**
	Compression Rate 60% ($r = 1$)							
SVD	0.32	0.04	0.69	0.31	0.55	0.53	0.00	0.49
ASVD (30)	0.62	0.10	0.64	0.42	0.50	0.49	0.03	0.70
FWSVD (12)	0.32	0.18	0.72	0.32	0.51	0.50	0.00	0.49
GFWSVD (Ours)	**0.42**	**0.70**	**0.74**	**0.42**	**0.65**	**0.52**	**0.05**	**0.49**