GRAIL: Post-hoc Compensation by Linear Reconstruction for Compressed Networks

Wenwu Tang$^{\diamond}$, Dong Wang$^{\diamond}$, Lothar Thiele*, Olga Saukh$^{\diamond,\S}$

$^{\diamond}$Graz University of Technology, §Complexity Science Hub, Austria
*ETH Zurich, Switzerland
{wenwu.tang, dong.wang, saukh}@tugraz.at, thiele@tik.ee.ethz.ch

Structured deep model compression methods reduce memory and inference costs, but the majority of existing approaches still suffer from notable accuracy degradation under aggressive compression. We propose *post-hoc blockwise compensation*, called GRAIL, a simple zero-finetuning step applied after pruning or folding that restores each block's input–output behavior using a small calibration set. The method summarizes producer-side activations with a Gram matrix and solves a ridge least-squares problem to project the original hidden representation onto the reduced hidden space, yielding a linear map that is merged into the consumer weights while the producer is narrowed to the selected or folded outputs. The approach is selector-agnostic (magnitude, Wanda, Gram-based selection, or folding), data-aware (requiring only a few forward passes without gradients or labels), and recovers classic pruning/folding when the Gram matrix is near identity. Across ResNets, ViTs, and decoder-only LLMs, post-hoc compensation with GRAIL consistently improves accuracy or perplexity over data-free and data-aware pruning/folding baselines in practical compression regimes, with manageable overhead and no backpropagation. Our code is available at: https://github.com/TWWinde/GRAIL

1. Introduction

Deployed models increasingly face tight latency and memory budgets, and achieving high-quality compression is essential for meeting these constraints without sacrificing accuracy. Yet in many deployment settings, retraining or even light finetuning after compression is infeasible: data may be proprietary, labels unavailable, or training cycles too costly for the post-training phase [1–3]. This motivates training-free, weight-space compression methods that directly narrow hidden width. Two structured families dominate: pruning, which selects a subset of channels or heads [4, 5], and folding, which clusters and merges them into a smaller basis [3, 6]. Both pruning and folding modify the hidden representation seen by the next (consumer) layer, yet typically do so without accounting for how dropped units contribute under the actual data distribution. In practice, reducing layer width changes the downstream representation statistics and can degrade accuracy or perplexity.

We address this gap with a simple *post-hoc blockwise compression compensation* step, called GRAIL, that restores the block's input-output behavior without labels or training. After a pruning or folding decision has been made (by any criterion), we briefly run the uncompressed model in evaluation mode on a small, unlabeled calibration set and record the activations at each consumer input (post-activation inputs for dense layers, concatenated per-head features just before the attention output projection). From these activations we compute only second-order statistics (Gram matrices). Using this summary, we fit a linear map that predicts the original hidden from the reduced hidden. Finally, we absorb this map into the consumer projection weights, while the producer is narrowed by selection (pruning) or per-cluster averaging (folding). The result is a one-shot, data-aware correction that requires no gradients, no labels, and adds only a small linear solve per block.

Our framework is post hoc and modular: it can be added on top of magnitude pruning, Wanda [4], SlimGPT [7], FLAP [8], or clustering-based folding [3]. It uses only a small amount of calibration data, and introduces low runtime overhead. Across ResNets [9], ViTs [10], and decoder-only LLMs,

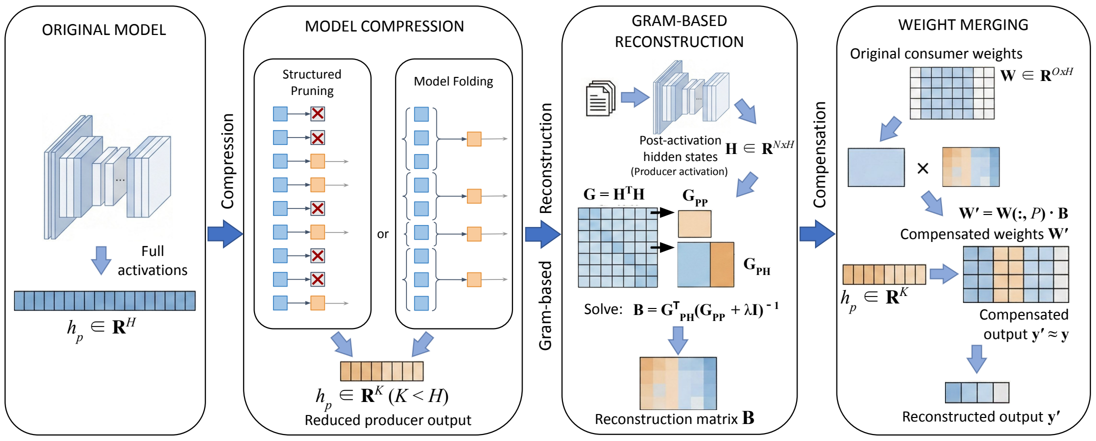

ORIGINAL MODEL
Full activations
$h_p \in \mathbf{R}^H$
Compression
MODEL COMPRESSION
Structured Pruning
or
Model Folding
$h_p \in \mathbf{R}^K$ $(K < H)$
Reduced producer output
Gram-based Reconstruction
GRAM-BASED RECONSTRUCTION
Post-activation hidden states (Producer activation)
$\mathbf{H} \in \mathbf{R}^{NxH}$
$\mathbf{G} = \mathbf{H}^T\mathbf{H}$
$\mathbf{G}_{PP}$
$\mathbf{G}_{PH}$
Solve: $\mathbf{B} = \mathbf{G}^T_{PH}(\mathbf{G}_{PP} + \lambda\mathbf{I})^{-1}$
Reconstruction matrix B
Compensation
WEIGHT MERGING
Original consumer weights
$\mathbf{W} \in \mathbf{R}^{OxH}$
$\mathbf{W}' = \mathbf{W}(:, P) \cdot \mathbf{B}$
Compensated weights W′
$h_p \in \mathbf{R}^K$
Compensated output $\mathbf{y}' \approx \mathbf{y}$
Reconstructed output y′

Figure 1: **GRAIL: GRAm-Integrated Linear compression workflow.** After a pruning / folding decision narrows a block's hidden width, we run a small, unlabeled calibration set through the uncompressed model and collect activations at each consumer input. From these activations, GRAIL forms the Gram matrix of second-order statistics and solves a ridge regression that reconstructs the original hidden representation from the reduced one. The resulting linear compensation is merged into the consumer projection, while the producer is narrowed by selection or clustering. This one-shot, training-free step restores each block's input–output behavior and applies uniformly to pruning and folding across CNNs, ViTs, and LLMs.

GRAIL consistently improves over pruning-only and folding-only baselines, narrowing the gap to the finetuned model without any finetuning. Our contributions are:

- **Training-free post-hoc blockwise compensation.** We propose a calibration-aware, one-shot linear weight compensation applied *after* pruning or folding that restores each block's input–output behavior using a small calibration dataset to compute second-order activation statistics—no labels, without calculating gradients and finetuning.
- **Practical recipe and broad evidence.** Large-scale experiments on ResNets [9], ViTs [10], and decoder-only LLMs across more than 700 checkpoints on CIFAR-10 [11], ImageNet-1K [12], and C4 [13] / WikiText-2 [14] / PTB [15] show consistent gains over pruning-only and folding-only baselines at matched compression. Our method also improves upon strong structured compression baselines that include their own recovery mechanisms, such as FLAP [8] with built-in bias correction, SlimGPT [7] curvature-based reconstruction and Wanda++ [16] with local optimization.
- **Low data requirement, memory and compute overhead.** The method uses only a tiny, unlabeled calibration set (subset of the training set), no gradients, and an efficient Gram accumulation, requiring at most a few seconds and manageable memory even for CLIP [17] and LLaMA-2-7B [18] models.

2. Related Work

Closest compensation-related works. Compensation in structured width reduction is typically limited or implicit. Pruning methods often retain surviving channels without correcting the downstream projection, relying on fine-tuning or localized updates to regain accuracy [19–21]. Folding and clustering approaches provide an algebraic update of the consumer weights [3, 6, 22], but these updates are data-free and assume that removed units are well represented by simple centroids, which may diverge from the true activation geometry. Second-order methods (OBS/OBD) introduce principled loss-aware corrections [23, 24], yet they primarily target unstructured parameter removal and do not enforce architectural constraints such as multi-head attention reshape rules. A closely related work is channel pruning with output reconstruction [19], which iteratively removes input channels of a single layer and solves a least-squares problem to preserve that layer's output. However, it operates locally on individual projections, requires repeated pruning–reconstruction steps during or after training, and does not treat coupled producer–consumer pairs such as the $(W_{\mathrm{fc}}, W_{\mathrm{proj}})$ structure in Transformers.

Recent LLM-focused pruning methods such as SlimGPT [7] and Wanda++ [16] incorporate local reconstruction or gradient-based corrections, and FLAP [8] adds head-wise bias compensation, but these mechanisms are tied to specific pruning heuristics and are not general-purpose compensation methods. Moreover, these approaches are designed for decoder-only Transformers and do not naturally transfer to CNNs, ViTs, or folding-based width reduction. Recent work has also begun to explore post-pruning recovery for LLMs: *e.g.*, Shen et al. [25] introduce a Newton-based structured pruning method with a built-in compensation step, while Feng et al. [26] propose re-injecting lost attention components to restore performance in pruned LLMs. Our work differs in that GRAIL provides a unified, linear reconstruction which is selector-agnostic, training-free, and applies across both structured pruning and folding, as well as to vision and language models.

Activation-matching compensation. The most direct analogues are PTQ reconstruction methods, which match layer outputs on calibration data to compensate quantization errors [2, 27–29]. These approaches optimize rounding or scaling and can incorporate second-order guidance, but they operate at fixed width and therefore do not handle dimension-changing reducers from pruning or folding, nor head-structured transformations in LLMs. Distillation-based techniques [30] also provide activation alignment but require full student training. Overall, prior work lacks a unified, training-free, activation-aware compensation specifically designed for structured width reduction.

3. GRAIL: GRAm-Integrated Linear compensation

Classical width reduction prunes or clusters channels under the implicit assumption that removed hidden units do not substantially affect those retained. This weight–space heuristic often distorts the consumer's input geometry, especially in pre-LN Transformers where hidden activations co-activate in structured ways [31, 32]. As illustrated in Figure 1, GRAIL replaces this assumption with a data-aware correction step: using a small, no-grad calibration set, we accumulate the consumer-input Gram matrix and solve a ridge regression that reconstructs the original hidden representation from its reduced counterpart. The resulting linear map is merged into the consumer weights, yielding a selector-agnostic, zero-finetuning procedure that restores the block's input–output behavior. Below we detail this procedure and show how it applies uniformly across CNN, ViT, and LLM blocks.

3.1. Dense, convolutional and FFN blocks

We consider a block consisting of a producer layer $i-1$ followed by a consumer layer i. Let h denote the output activation from the producer, which serves as the input to the consumer. GRAIL learns a cross-layer compensation from producer activations, then folds it into the consumer weights.

Dense block compensation. Consider two consecutive fully connected layers $i-1$ and i:

$$h = \phi(\mathbf{W}_{\mathrm{i-1}} z + b_{\mathrm{i-1}}) \in \mathbb{R}^H, \qquad y = \mathbf{W}_{\mathrm{i}} h + b_{\mathrm{i}} \in \mathbb{R}^O,$$

where $\mathbf{W}_{\mathrm{i-1}} \in \mathbb{R}^{H \times C}$, $\mathbf{W}_{\mathrm{i}} \in \mathbb{R}^{O \times H}$ and ϕ denotes activation function. We compress the hidden dimension from H to $K \ll H$ by selecting a subset $P \subset \{1, \ldots, H\}$ of size $|P| = K$ or using model folding. The selection step is method-agnostic: P may be obtained using activation norms, weight magnitudes, Wanda [4], or Gram-based selection [33]. In folding methods, channels are grouped into clusters and each cluster is replaced by its centroid [3]. The reduced basis P consists of these centroids, with all channels collapsed onto their cluster means.

Many model compression methods can be interpreted as applying a linear mapping matrix that reduces the hidden width [22]. Let $\mathbf{H} \in \mathbb{R}^{N \times H}$ denote the original hidden activations where each row corresponds to an activation h for one of N data samples. Width reduction for target dimension

$K < H$ can be written universally as

$$\mathbf{H}_{\text{red}} = \mathbf{H}\mathbf{M}, \qquad \mathbf{M} \in \mathbb{R}^{H \times K}.$$

In *structured pruning*, $\mathbf{M}$ is a binary matrix that retains only a subset of channels indexed by P:

$$\mathbf{M}_{\text{prune}} = [\mathbf{e}_{p_1} \quad \mathbf{e}_{p_2} \quad \cdots \quad \mathbf{e}_{p_K}], \qquad \mathbf{H}_{\text{pruned}} = \mathbf{H}\,\mathbf{M}_{\text{pruned}},$$

where $\mathbf{e}_{p_i}$ are standard basis vectors. In contrast, *model folding* groups channels into K clusters C_k, $1 \le k \le K$ of activations and replaces each cluster by its centroid. The folding map

$$\mathbf{M}_{\text{fold}}(h,k) = \begin{cases} 1/|C_k|, & h \in C_k, \\ 0, & \text{otherwise}, \end{cases} \qquad \mathbf{H}_{\text{folded}} = \mathbf{H}\,\mathbf{M}_{\text{fold}},$$

where h denotes the index of an activation averages all channels in C_k into a single direction. Thus $\mathbf{M}_{\text{fold}}$ is a low-rank projection that mixes channels rather than discarding them.

GRAIL learns a data-aware cross-channel reconstruction mapping $\mathbf{B}$ from post-activation hidden statistics, and merges this reconstruction into the consumer weight $\mathbf{W}_{\text{i}}$, improving compressed models without finetuning. Our goal is to approximate the original hidden representation h using only the retained subset h_P. We seek a linear reconstruction map $\mathbf{B} \in \mathbb{R}^{H \times K}$ such that

$$h \;\approx\; \mathbf{B}h_P$$

on a small unlabeled calibration set. By collecting activations into matrices $\mathbf{H}$ and $\mathbf{H}_P$ across N samples, we solve the ridge regression problem:

$$\min_{\mathbf{B}} \; \left\|\mathbf{H} - \mathbf{H}_P\mathbf{B}^\top\right\|_F^2 + \lambda\|\mathbf{B}\|_F^2,$$

where $\lambda > 0$ is a ridge regularization coefficient that stabilizes the solution when $\mathbf{H}_P^\top\mathbf{H}_P$ is ill-conditioned or near-singular. The normal equations give the closed-form solution:

$$\mathbf{B} = \mathbf{H}^\top\mathbf{H}_P\left(\mathbf{H}_P^\top\mathbf{H}_P + \lambda\mathbf{I}\right)^{-1}.$$

For pruning, the reducer is a selection matrix, and therefore $\mathbf{G}_{PP} = \mathbf{G}[P,P]$ is the correct Gram submatrix. For folding, however, the reducer is a merge map $\mathbf{M}_{\text{fold}} \in \mathbb{R}^{H\times K}$, so the reduced activations satisfy $\mathbf{H}_{\text{red}} = \mathbf{H}\mathbf{M}_{\text{fold}}$. Consequently, the Gram block entering the ridge system becomes

$$\mathbf{G}_{PP}^{\text{fold}} = \mathbf{H}_{\text{red}}^\top\mathbf{H}_{\text{red}} = \mathbf{M}_{\text{fold}}^\top\,\mathbf{G}\,\mathbf{M}_{\text{fold}},$$

which generalizes the pruning case and correctly accounts for channel mixing. Substituting this expression into the reconstruction formula preserves the unified form $\mathbf{B} = \mathbf{G}_{PH}^\top(\mathbf{G}_{PP} + \lambda\mathbf{I})^{-1}$.

The reconstruction map $\mathbf{B}$ is merged into the consumer projection weights:

$$\mathbf{W}_{\text{i}}' = \mathbf{W}_{\text{i}}\mathbf{B} \in \mathbb{R}^{O\times K},$$

while the producer weights are width-reduced by indexing:

$$\mathbf{W}_{\text{i}-1}' = \mathbf{W}_{\text{i}-1}[P,:], \qquad b_{\text{i}-1}' = b_{\text{i}-1}[P], \qquad b_{\text{i}}' = b_{\text{i}}.$$

At inference, the reduced hidden activations $h_P = \phi(\mathbf{W}_{\text{i}-1}'z + b_{\text{i}-1}')$ produce

$$y' = \mathbf{W}_{\text{i}}'h_P + b_{\text{i}}' \;\approx\; \mathbf{W}_{\text{i}}h + b_{\text{i}}.$$

An equivalent formulation minimizes

$$\left\|\mathbf{H}_P\mathbf{W}_{\text{i}}'^\top - \mathbf{H}\mathbf{W}_{\text{i}}^\top\right\|_F^2 \;+\; \lambda\|\mathbf{W}_{\text{i}}'\|_F^2,$$

yielding

$$\mathbf{W}_{\text{i}}' = \mathbf{W}_{\text{i}}\mathbf{G}_{PH}^\top(\mathbf{G}_{PP} + \lambda\mathbf{I})^{-1},$$

which is identical to the merged form via the regression solution.

Convolutional block compensation. For a convolutional layer with weights $\mathbf{W}_{\text{conv}} \in \mathbb{R}^{O \times H \times k_H \times k_W}$, the compensation map $\mathbf{B} \in \mathbb{R}^{H \times K}$ learned from the activation regression of the preceding fully connected case is applied along the *input channel* dimension of the convolution kernel:

$$\mathbf{W}^{\text{new}}_{\text{conv}}(o, k, :, :) = \sum_{h=1}^{H} \mathbf{W}_{\text{conv}}(o, h, :, :)\, \mathbf{B}(h, k).$$

Compensation for ViTs and CLIP. For Vision Transformers (ViT) and CLIP models GRAIL operates at the block level, specifically targeting the MLP modules. For Transformer MLP/FFN blocks $\mathbf{W}_{\text{fc}}$ (expansion) and $\mathbf{W}_{\text{proj}}$ (projection) form a coupled producer–consumer pair. The pruning process removes rows from $\mathbf{W}_{\text{fc}}$ and corresponding columns from $\mathbf{W}_{\text{proj}}$. The compensation is then applied exclusively to the input of the projection layer to reconstruct the output of the original block:

$$\mathbf{W}'_{\text{proj}} = \mathbf{W}_{\text{proj}}\mathbf{B}, \tag{1}$$

where $\mathbf{B}$ is the regression matrix derived from the Gram statistics of the hidden activations. This coupled treatment ensures that the interaction between the two layers is preserved, which is critical for maintaining the expressivity of the MLP.

Complexity. Accumulating $\mathbf{G}$ requires $O(NH^2)$ time and $O(H^2)$ memory. In practice, this cost is incurred only during calibration and is manageable on modern datacenter GPUs (*e.g.*, A100-class devices) even for multi-billion-parameter LLMs, but may become restrictive on memory-constrained accelerators. Compression involves solving a $K \times K$ system and one GEMM, $O(K^3 + OHK)$, with $K \ll H$. In our experiments, a few hundreds to thousand sequences for LLMs suffice, for CNNs / ViTs we use 128 samples for calibration. Stability is ensured by setting $\lambda = \alpha \cdot \operatorname{mean}\operatorname{diag}(\mathbf{G}_{PP})$ with $\alpha \in [10^{-4}, 5 \cdot 10^{-3}]$.

3.2. Attention blocks, compensation for LLMs

We adopt the same producer–consumer view and Gram–regression compensation used for vision models, but decoder-only LLMs impose additional structure. The key differences concern: (i) *where* activations are sampled, (ii) the *head-structured* nature of self-attention (including grouped-query attention, GQA), and (iii) how pruning and folding are instantiated at the head level. Our method employs a closed-loop compensation mechanism, by re-evaluating the Gram matrix for each layer based on the output of the already-pruned previous layers, to dynamically adapt to the structural changes. This sequential alignment prevents error propagation and ensures that the regression-based compensation remains accurate throughout the entire depth of the model.

Where to sample activations. For each submodule we form the Gram matrix at the *consumer input*. In pre-LN LLMs this means: (i) MLP: post-GELU vectors $x \in \mathbb{R}^H$ at the input of the projection matrix $\mathbf{W}_{\text{proj}}$, (ii) Self-attention: the concatenated per-head vector $x \in \mathbb{R}^H$ *before* the output projection $\mathbf{W}_{\text{o}}$. Over batches n and time steps t we accumulate the uncentered second moment

$$\mathbf{G} \;=\; \sum_{n,t} x_{n,t}\, x_{n,t}^{\top} \;\in\; \mathbb{R}^{H \times H}.$$

Head-structured reduction in attention. Let d_{model} be the hidden size. Multi-head attention factorizes the attention feature axis as $H = n_{\text{h}} d_{\text{h}}$, where n_{h} is the number of heads and d_{h} is the per-head width. Any reduction must preserve the reshape/split invariants of attention and therefore act at the *head* level. If $\mathbf{R}_{\text{heads}} \in \mathbb{R}^{K \times n_{\text{h}}}$ reduces heads to $K \ll n_{\text{h}}$, its action on features is the Kronecker lift

$$\mathbf{R}_{\text{feat}} \;=\; \mathbf{R}_{\text{heads}} \otimes \mathbf{I}_{d_{\text{h}}} \;\in\; \mathbb{R}^{(K d_{\text{h}}) \times (n_{\text{h}} d_{\text{h}})}, \tag{2}$$

where $\mathbf{I}_{d_{\text{h}}}$ is the $d_{\text{h}} \times d_{\text{h}}$ identity. For GQA with G query groups and n_{kv} KV heads per group ($n_{\text{h}} = G\, n_{\text{kv}}$), the valid head reducer is block-diagonal,

$$\mathbf{R}_{\text{blk}} = \operatorname{blkdiag}(\underbrace{\mathbf{R}_{\text{kv}}, \ldots, \mathbf{R}_{\text{kv}}}_{G}) \in \mathbb{R}^{(G K_{\text{kv}}) \times (G n_{\text{kv}})},$$

$$\mathbf{R}_{\text{feat}} = \mathbf{R}_{\text{blk}} \otimes \mathbf{I}_{d_{\text{h}}}.$$

These constraints are specific to LLM attention and typically absent in ViT MLP/channel reductions.

Pruning vs. folding at the head level. In attention, width reduction must act at the level of heads. Pruning selects a subset $P \subset \{1, \ldots, n_h\}$ with $|P| = K$ and uses the selection matrix $\mathbf{S} \in \{0,1\}^{K \times n_h}$, whose Kronecker lift $\mathbf{R}_{\text{feat}} = \mathbf{S} \otimes \mathbf{I}_{d_h}$ operates on the concatenated feature axis. Folding instead mixes heads via a centroid map $\mathbf{R}_{\text{heads}}$ whose rows sum to one, and lifts it as in Equation 2, $\mathbf{R}_{\text{feat}} = \mathbf{R}_{\text{heads}} \otimes \mathbf{I}_{d_h}$. In this case, the reduced feature is a mixture of all heads, so there is no submatrix $\mathbf{H}_P$ of kept activations, compensation must therefore work with the full mixer $\mathbf{R}_{\text{feat}}$, not column selection.

4. Experimental Results

Our experiments address two core questions about the performance of GRAIL:

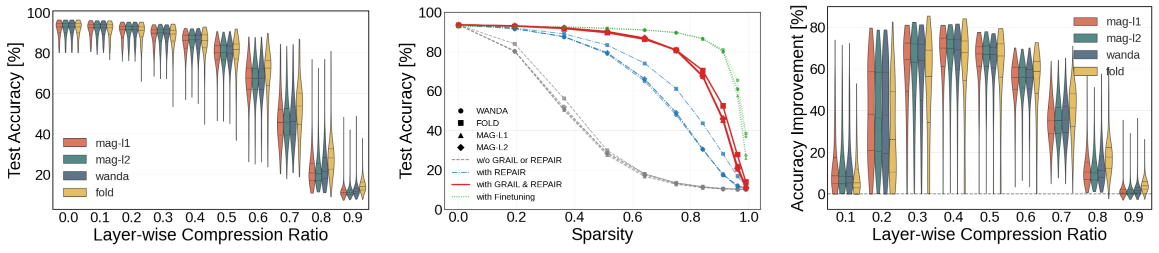
1. Does GRAIL consistently improve performance across architectures (ResNets, ViTs, LLMs), datasets (CIFAR-10, ImageNet-1K, C4 / WikiText-2 / PTB), and compression regimes (pruning, folding)? How does it compare to existing post-hoc compensation methods?
2. How sensitive is GRAIL to calibration-set size, and what are its practical computational and memory overheads?

4.1. GRAIL performance on ResNet-18, ViT-B/32 and CLIP ViT-B/32

We evaluate the effectiveness of GRAIL across diverse architectures (ResNet, ViT, and CLIP), training regimes, datasets (CIFAR-10 and ImageNet-1K), and compression settings. Our study uses a large collection of 773 pretrained checkpoints to ensure robustness and generality: 576 SGD-trained ResNet-18 models on CIFAR-10 from Saukh et al. [22], 125 ViT-B/32 models from Andriushchenko et al. [35], and 72 CLIP ViT-B/32 checkpoints on ImageNet-1K from Wortsman et al. [36].

We apply our framework to four structured width-reduction methods: magnitude pruning (L1 and L2), structured Wanda, and model folding, under uniform layer-wise compression ratios from 0.1 to 0.9. For each compressed model, we compare accuracy with and without our compensation step, and also evaluate its combination with BatchNorm REPAIR [34]. Although GRAIL is data-aware, it remains entirely training-free: calibration uses only 128 unlabeled images for vision models, and all steps require no gradients.

We evaluate the effectiveness of our proposed compensation framework across various compression ratios, ranging from 10% to 90% layer-wise compression ratio. Figure 2, Figure 3 and Figure 5 summarize the results for ResNet-18 on CIFAR-10, CLIP ViT-B/32 on ImageNet-1K and ViT-B/32 on CIFAR-10 (in appendix) under different compression paradigms (magnitude pruning, Wanda, and model folding). GRAIL achieves a significant accuracy recovery from the compressed model. At low to moderate sparsity (10%-40%), the compensated models achieve near-lossless compression, often recovering the accuracy up to within 0.5% of the original model. In some cases, *e.g.*, ResNet-18 at 20%

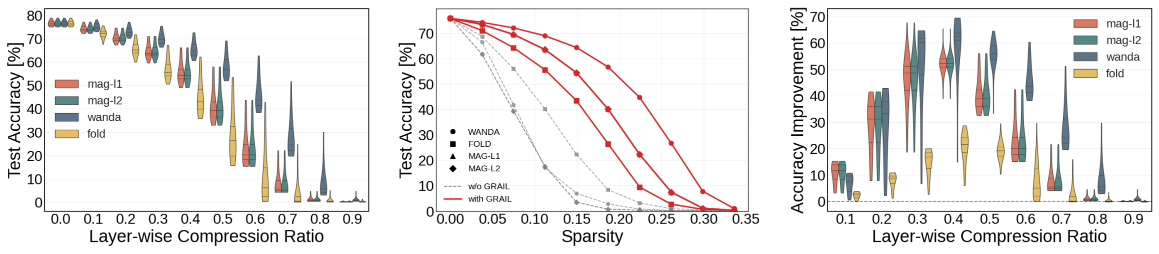


(a) Test accuracy vs. layer-wise uniform compression ratio

(b) Mean accuracy vs. sparsity, against REPAIR and finetuning

(c) Relative improvement from compensation by GRAIL

Figure 2: **GRAIL on 576 SGD-trained ResNet-18 models on CIFAR-10.** Using checkpoints from Saukh et al. [22], we find that GRAIL consistently improves both pruning (most notably at low compression ratios) and folding (at moderate to high ratios), as shown in panels (a) and (c). It also outperforms REPAIR [34], further narrowing the gap to models fine-tuned for 5 epochs (panel (b)).



(a) Test accuracy vs. layer-wise uniform compression ratio

(b) Mean accuracy vs. sparsity

(c) Relative improvement from compensation by GRAIL

Figure 3: **GRAIL on 72 CLIP ViT-B/32 checkpoints on ImageNet-1K.** Using checkpoints from Wortsman et al. [36], we find that GRAIL consistently improves both pruning and folding (panels (a) and (c)), with pruning benefiting more strongly for all compression ratios. Consequently, GRAIL-compensated folded models trail their GRAIL-compensated pruned counterparts.

sparsity, the compensated model even slightly surpasses the baseline, suggesting a regularization effect. At moderate to high sparsity (50%-80%), the advantage of GRAIL becomes most pronounced. For instance, at 65% sparsity using L1-magnitude pruning, the baseline accuracy of ResNet-18 collapses to 17.6%, whereas GRAIL restores it to 84.8%, a remarkable 67.2% improvement. Similarly, for ViT and CLIP models, which are known to be sensitive to structured pruning, our framework effectively mitigates the catastrophic accuracy drop typically observed at high compression rates. Our approach proves to be agnostic to the underlying compression operator, validating its universality as a plug-and-play module for post-training compression and compensation.

4.2. GRAIL performance on LLaMA-2-7B

We evaluate GRAIL on LLaMA-2-7B across three standard language modeling benchmarks: C4, WikiText-2, and PTB. We consider sparsity levels from 10% to 70% and apply three representative structured pruning methods: structured Wanda, SlimGPT, and FLAP. All methods use 128 calibration sequences of length 2048, and performance is reported in terms of test perplexity.

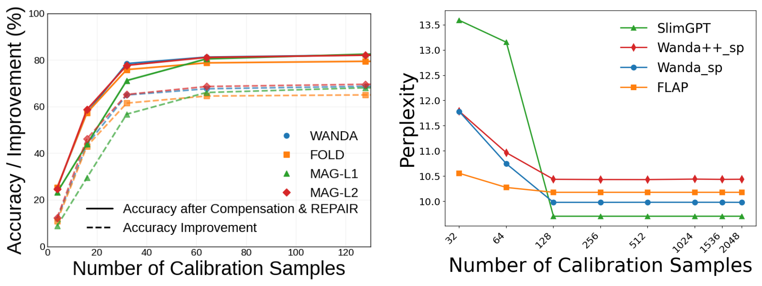
Table 1 summarizes the results. Across all datasets and sparsity levels, GRAIL helps to considerably reduce perplexity relative to the corresponding pruning baselines. The benefit of GRAIL is most pronounced for methods that perform aggressive structured removal without explicit activation reconstruction. For instance, SlimGPT exhibits rapid perplexity growth beyond 30% sparsity, whereas GRAIL substantially stabilizes its behavior across all three datasets. FLAP, which already includes bias compensation, also benefits from GRAIL at higher sparsities, indicating that correcting second-order activation geometry complements existing first-order compensation schemes. We further compare GRAIL to regional optimization used in Wanda++. Regional optimization performs local gradient-based fine-tuning of retained weights, whereas GRAIL applies a closed-form, deterministic reconstruction. As shown in Table 1, GRAIL achieves comparable or better perplexity recovery for low sparsity levels without requiring gradients or iterative optimization.

Table 2 reports zero-shot accuracies for four structured-pruning baselines (FLAP, Wanda, SlimGPT, Wanda++) at 20% and 50% sparsity, with and without our GRAIL compensation. Across all pruning methods and benchmarks, GRAIL often improves or preserves zero-shot accuracy, confirming that reconstruction of pruned feature directions is broadly beneficial and algorithm-agnostic. For some datasets, such as ARC-E and BoolQ, the improvements are consistent and significant across all compression methods and sparsity levels. These results highlight that GRAIL provides a simple, training-free mechanism to recover destruction of cross-channel geometry, yielding benefits across different pruning criteria and task types.

[1]Configurations produced NaN values during evaluation (due to numerical overflow in the forward pass).

Table 1: **Perplexity (↓) comparison on LLaMA-2-7B** under different sparsity levels across three datasets (C4, PTB, WikiText2) with 2048 sequence length and 128 calibration samples.

Dataset	Method	10%	20%	30%	40%	50%	60%	70%
C4	Wanda	8.25	9.88	12.49	20.80	131.54	155.41	508.51
	Wanda **+ GRAIL**	**8.05**	**9.28**	**11.26**	15.50	25.97	213.92	2127.03
	Wanda++	8.07	9.32	11.74	15.89	26.72	77.35	132.58
	Wanda++ **+ GRAIL**	**8.05**	**9.28**	**11.26**	**15.05**	23.75	187.21	1688.64
	SlimGPT	11.41	18.80	23.51	57.75	235.56	1288.51	4049.08
	SlimGPT **+ GRAIL**	8.21	9.49	11.66	15.83	28.05	67.63	**102.46**
	FLAP	8.35	10.26	12.84	17.41	22.94	100.53	803.27
	FLAP **+ GRAIL**	8.26	10.05	12.40	16.45	**22.04**	**40.77**	126.88
PTB	Wanda	22.01	30.91	80.70	174.46	287.26	423.63	840.68
	Wanda **+ GRAIL**	21.85	24.31	25.94	49.25	62.86	425.26	**215.91**
	Wanda++	22.14	26.32	30.95	37.85	54.29	90.80	338.67
	Wanda++ **+ GRAIL**	22.37	25.21	25.85	85.64	56.49	592.58	217.58
	SlimGPT	37.80	56.94	85.82	419.73	938.52	3683.63	7983.82
	SlimGPT **+ GRAIL**	22.68	25.74	35.99	39.47	46.62	76.58	217.83
	FLAP	22.38	24.99	29.22	37.34	54.34	100.01	1157.63
	FLAP **+ GRAIL**	**21.63**	**23.05**	**25.01**	**28.90**	**35.58**	**52.16**	576.46
WikiText2	Wanda	6.18	7.45	9.18	15.16	171.29	272.47	1839.20
	Wanda **+ GRAIL**	**5.75**	6.44	7.45	9.98	18.85	39.59	408.68
	Wanda++	5.80	6.56	7.59	10.18	23.29	44.00	128.00
	Wanda++ **+ GRAIL**	**5.75**	6.45	7.44	10.44	16.14	35.17	288.79
	SlimGPT	7.69	9.81	-[1]	62.56	590.75	1220.71	16764.33
	SlimGPT **+ GRAIL**	5.81	**6.32**	**7.34**	**9.71**	16.30	23.45	**43.14**
	FLAP	6.01	7.16	8.85	11.49	16.67	31.80	490.85
	FLAP **+ GRAIL**	5.88	6.80	8.08	10.18	**13.45**	**20.46**	71.63



(a) ResNet-18 (75% sparsity)

(b) LLaMA-2-7B (40% sparsity)

Figure 4: **Ablation on compensation dataset size. Left:** effect on accuracy recovery for ResNet-18 on CIFAR-10 at 75% sparsity. **Right:** effect on LLaMA-2-7B perplexity at 40% sparsity on WikiText-2. In our experiments we use 128 unlabeled images for vision models (ResNet-18, ViT, CLIP) and only 128 sequences of length 2048 tokens for LLMs.

4.3. Data and resource efficiency

To evaluate the data efficiency of GRAIL, Figure 4 illustrates the accuracy improvement—defined as the margin between the before and after weight compensation model accuracy across varying numbers of calibration samples for different compression methods on ResNet-18 with 75% sparsity and on LLaMa-2-7B at 40% sparsity. ResNet-18 experiments are performed over 100 checkpoints with original accuracy above 90%, and we report the averaged performance. The results show that GRAIL performance follows a logarithmic growth pattern: significant accuracy recovery is achieved with a very small batch of samples, after which the performance gain rapidly saturates. This observation highlights the high data efficiency of the closed-form ridge regression solution, indicating that a small calibration set is sufficient to effectively reconstruct feature maps and recover model performance. For LLaMa-2-7B, only 128 sequences of length 2048 tokens were used in our evaluation. Across all methods, calibration with at least 128 samples is sufficient to reach the performance plateau; SlimGPT requires the most samples to stabilize, whereas FLAP and Wanda exhibit stronger data efficiency.

Table 2: **Zero-shot accuracy (↑) comparison on LLaMA-2-7B** for different model compression methods with and without GRAIL compensation calibrated on C4 dataset across six benchmarks.

Sparsity	Method	ARC-C	ARC-E	HellaSwag	PIQA	BoolQ	Winogrande
20%	Wanda	0.3865	0.7096	**0.5452**	0.7633	0.7141	0.6417
	Wanda **+ GRAIL**	**0.4164**	**0.7382**	0.5348	0.7655	0.7416	**0.6843**
	Wanda++	0.3959	0.7256	0.5436	**0.7688**	0.7083	0.6827
	Wanda++ **+ GRAIL**	0.4104	0.7357	0.5357	0.7644	**0.7446**	0.6772
	SlimGPT	0.3200	0.5829	0.4781	0.7133	0.6642	0.5359
	SlimGPT **+ GRAIL**	0.4138	0.7256	0.5373	0.7628	0.7431	0.6819
	FLAP	0.3609	0.6759	0.5170	0.7508	0.6872	0.6638
	FLAP **+ GRAIL**	0.3490	0.6776	0.4952	0.7503	0.7086	0.6606
50%	Wanda	0.1937	0.2795	0.2680	0.5424	0.5911	0.4838
	Wanda **+ GRAIL**	0.2082	0.4630	0.3379	0.6485	0.6144	0.5446
	Wanda++	0.2031	0.3636	0.3140	0.6072	0.6254	0.5012
	Wanda++ **+ GRAIL**	0.2133	0.4638	0.3385	0.6480	0.6153	0.5280
	SlimGPT	0.2116	0.2917	0.2697	0.5533	0.5575	0.5296
	SlimGPT **+ GRAIL**	0.2406	0.4731	0.3755	**0.6638**	0.6349	0.5864
	FLAP	**0.2679**	0.4718	**0.3833**	0.6529	**0.6471**	**0.5983**
	FLAP **+ GRAIL**	0.2449	**0.4819**	0.3696	0.6551	0.6260	**0.5983**

Table 3: **Overhead of GRAIL compensation on vision and language models.** The calibration step collect activation statistics, while the compensation step comprises GRAIL computations.

Model	Overhead Time (s)		Overhead Memory (MB)	
	Calibration	Compensation	Calibration	Compensation
ResNet	0.19	0.10	162.02	22.00
ViT	0.20	0.04	161.62	5.76
CLIP	0.95	0.16	300.00	139.54
LLaMA-2-7B	58.077	3.16	1137.12	3367.00

To assess the computational and memory overhead introduced by GRAIL, we measure both runtime and peak memory across all architectures and present the results in Table 3. The evaluation consists of two stages: (1) a single-batch calibration pass (batch size 128 images or 256 sequences) used to collect activation statistics, and (2) the subsequent GRAIL compensation. All measurements are performed on a single NVIDIA A100 GPU. The results show that calibration dominates the total cost, while the compensation step itself is lightweight relative to the base model size. For large models such as LLaMA-2-7B, the peak memory footprint (≈3 GB) fits comfortably within standard datacenter GPU budgets, though this overhead may be non-trivial in more memory-constrained deployment settings.

5. Conclusions, Limitations and Outlook

We introduce GRAIL, a unified, training-free layer-wise, interleaved pruning-and-compensation framework that restores the input–output behavior of pruned or folded blocks through a calibration-driven linear reconstruction. Across ResNets, ViTs, and decoder-only LLMs, GRAIL consistently improves accuracy or perplexity over pruning-only and folding-only baselines and complements existing recovery mechanisms, while requiring only a small unlabeled calibration set. Limitations include its reliance on a short forward pass through the uncompressed model, sensitivity to distribution shifts in activation statistics, and its block-local nature, which may constrain performance under extreme compression. In addition, Gram-matrix accumulation introduces an $\mathcal{O}(H^2)$ memory footprint during calibration, which is practical on datacenter-class GPUs. Extending GRAIL to jointly compensate multiple layers and integrating it with quantization, KV-cache compression, or task-aware calibration signals are promising future directions.

Acknowledgments

This work was supported by the ANT project, funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101169439, and by the FFG COMET K1 Centre "Pro^{2}Future II" (Cognitive and Sustainable Products and Production Systems of the Future; Contract No. 911655). The results presented in this paper were obtained using the computational resources of Pro2Future GmbH, the Central IT Services of Graz University of Technology (ZID), and the Austrian Scientific Computing (ASC) infrastructure.

References

[1] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. *Artif. Intell. Rev.*, 53(7): 5113–5155, October 2020. ISSN 0269-2821. doi: 10.1007/s10462-020-09816-7. URL https://doi.org/10.1007/s10462-020-09816-7.

[2] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

[3] Dong Wang, Haris Šikić, Lothar Thiele, and Olga Saukh. Forget the data and fine-tuning! just fold the network to compress. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=W2Wkp9MQsF.

[4] Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M. Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, Nathan Susanj, Zheng Zhang, Jack FitzGerald, and Abhishek Kumar. Wanda++: Pruning large language models via regional gradients, 2025. URL https://arxiv.org/abs/2503.04992.

[5] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL https://arxiv.org/abs/2301.00774.

[6] Yiting Chen, Zhanpeng Zhou, and Junchi Yan. Going beyond neural network feature similarity: The network feature complexity and its interpretation using category theory. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4bSQ3lsfEV.

[7] Gui Ling, Ziyang Wang, YuliangYan, and Qingwen Liu. SlimGPT: Layer-wise structured pruning for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=MxF0IKJtKW.

[8] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i10.28960. URL https://doi.org/10.1609/aaai.v38i10.28960.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[11] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

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

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[14] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[15] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

[16] Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M. Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, Nathan Susanj, Zheng Zhang, Jack FitzGerald, and Abhishek Kumar. Wanda++: Pruning large language models via regional gradients. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025. URL https://openreview.net/forum?id=WjnJf5ft0B.

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

[18] 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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[19] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks, 2017. URL https://arxiv.org/abs/1707.06168.

[20] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019. URL https://arxiv.org/abs/1810.02340.

[21] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one?, 2019. URL https://arxiv.org/abs/1905.10650.

[22] Olga Saukh, Dong Wang, Haris Šikić, Yun Cheng, and Lothar Thiele. Cut less, fold more: Model compression through the lens of projection geometry. In *EurIPS 2025 Workshop on Rethinking AI: Efficiency, Frugality, and Sustainability*, 2025. URL https://openreview.net/forum?id=xl3cOeQqEi.

[23] Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. *IEEE International Conference on Neural Networks*, pages 293–299, 1993.

[24] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

[25] Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Jing Liu, Ruiyi Zhang, Ryan A. Rossi, Hao Tan, Tong Yu, Xiang Chen, Yufan Zhou, Tong Sun, Pu Zhao, Yanzhi Wang, and Jiuxiang Gu. Numerical pruning for efficient autoregressive models. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i19.34249. URL https://doi.org/10.1609/aaai.v39i19.34249.

[26] Zijian Feng, Hanzhang Zhou, Zixiao Zhu, Tianjiao Li, Jia Jim Deryl Chua, Lee Onn Mak, Gee Wah Ng, and Kezhi Mao. Restoring pruned large language models via lost component compensation, 2025. URL https://arxiv.org/abs/2510.21834.

[27] Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization, 2020. URL https://arxiv.org/abs/2004.10568.

[28] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. {BRECQ}: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

[29] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024. URL https://arxiv.org/abs/2306.00978.

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

[31] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models, 2024. URL https://arxiv.org/abs/2306.11695.

[32] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022. URL https://arxiv.org/abs/2208.07339.

[33] C.H. Sarvani, Mrinmoy Ghorai, Shiv Ram Dubey, and S.H. Shabbeer Basha. Hrel: Filter pruning based on high relevance between activation maps and class labels. *Neural Networks*, 147:186–197, March 2022. ISSN 0893-6080. doi: 10.1016/j.neunet.2021.12.017. URL http://dx.doi.org/10.1016/j.neunet.2021.12.017.

[34] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair, 2023. URL https://arxiv.org/abs/2211.08403.

[35] Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization, 2023. URL https://arxiv.org/abs/2302.07011.

[36] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. URL https://arxiv.org/abs/2203.05482.

Appendix

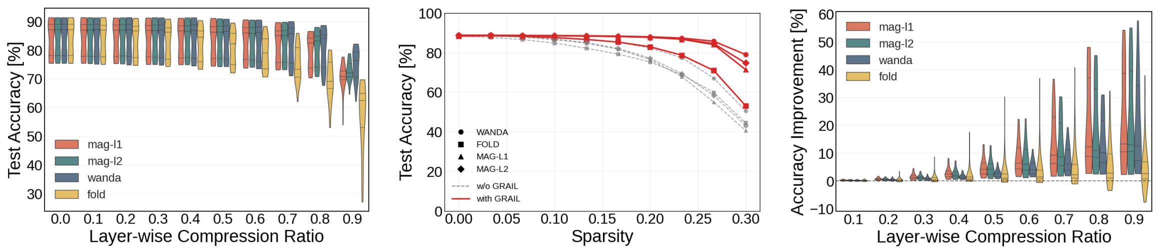
A. Implementation Details

Our vision model implementation is based on [22][2]. For the LLM, we build upon the open-source FLAP framework[3], into which we integrate multiple structured pruning methods under a unified interface and implement the proposed GRAIL compensation mechanism for consistent evaluation. All experiments were conducted on a compute cluster equipped with $8\times$ NVIDIA A100 (40 GB RAM) GPUs. To ensure reproducibility, all random seeds are fixed in the configuration files and execution scripts.

B. Use of Large Language Models

ChatGPT-3 was used to correct grammatical errors in the manuscript and to address minor formatting issues in Overleaf. Gemini-3 Pro was used to assist with coding and debugging programming errors. All research concepts, theoretical contributions, experimental design, and the scientific content of the manuscript were fully developed by the authors.

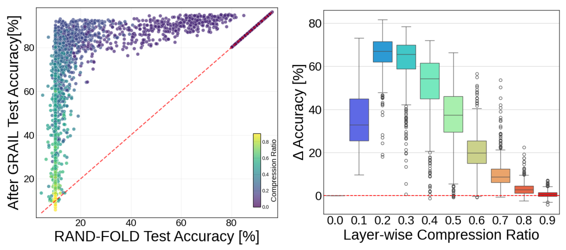
C. More Results



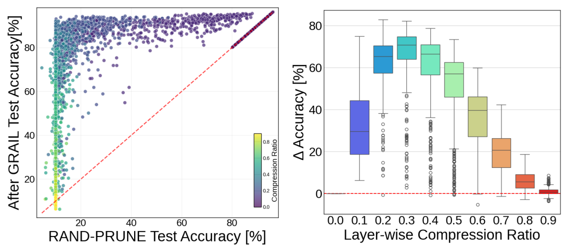
(a) Test accuracy vs. layer-wise uniform compression ratio (b) Mean accuracy vs. sparsity (c) Relative improvement from compensation by GRAIL

Figure 5: **GRAIL on 125 ViT-B/32 models on CIFAR-10.** Using checkpoints from Andriushchenko et al. [35], we find that GRAIL consistently improves pruning and in most cases also folding, as shown in panels (a) and (c). However, similarly to CLIP ViT-B/32 results reported in the main paper, GRAIL-compensated folded models lag behind GRAIL-compensated pruned models.

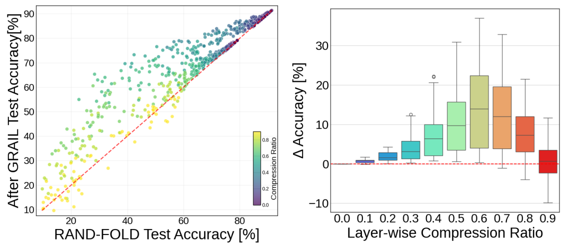
[2]https://github.com/osaukh/folding_as_projection
[3]https://github.com/CASIA-LMC-Lab/FLAP



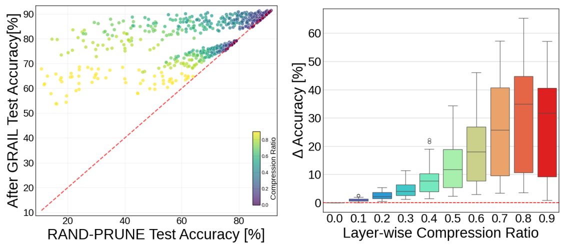
(a) ResNet-18: Random folding, w/ vs. w/o GRAIL



(b) ResNet-18: Random pruning, w/ vs. w/o GRAIL

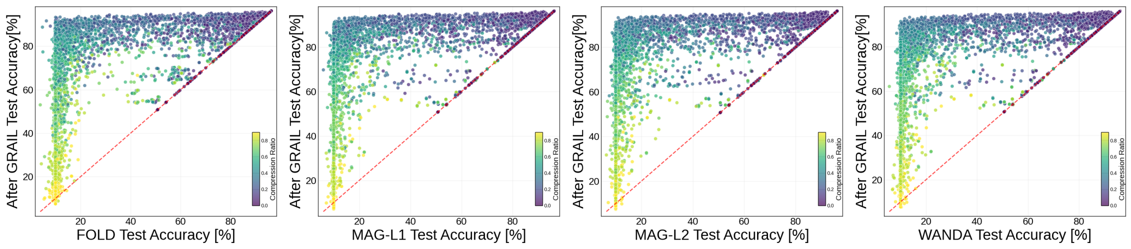


(c) ViT-B/32: Random folding, w/ vs. w/o GRAIL

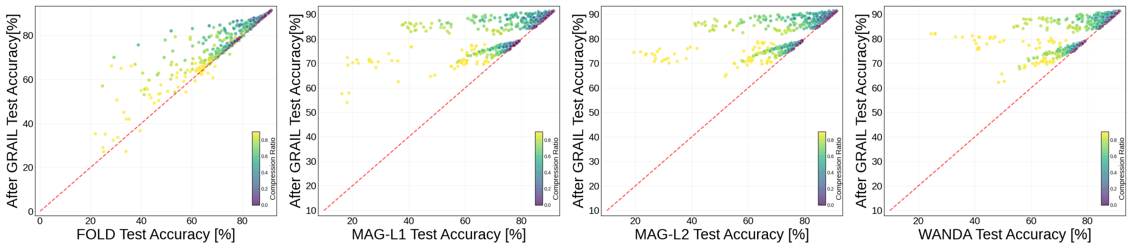


(d) ViT-B/32: Random pruning, w/ vs. w/o GRAIL

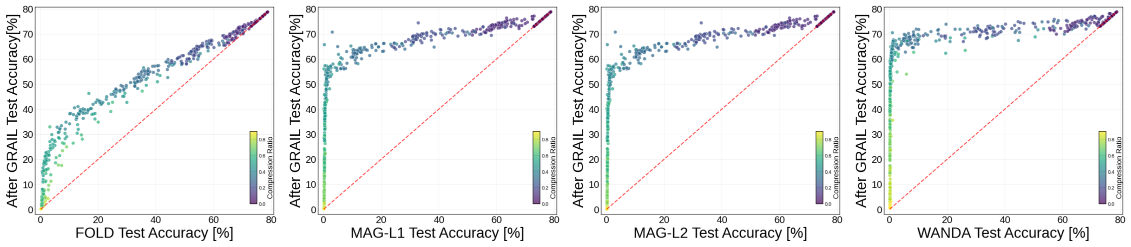
Figure 6: **GRAIL on ResNet-18 and ViT-B/32 under random folding and pruning.** Across both architectures, GRAIL consistently improves the accuracy of compressed models, as seen in the before/after scatter plots (left) and the accuracy gains across compression ratios (right) for all four settings: ResNet-18 folding (a), ResNet-18 pruning (b), ViT-B/32 folding (c), and ViT-B/32 pruning (d).



(**a**) **ResNet-18:** GRAIL improves accuracy across pruning strategies.



(**b**) **ViT:** Similar upward shift in performance due to GRAIL in transformer architectures.



(**c**) **CLIP:** GRAIL consistently boosts encoder representations.

Figure 7: **GRAIL across pruning and folding compression for ResNet-18, ViT-B/32, and CLIP ViT-B/32.** From left to right: Folding, Mag-L1, Mag-L2, and Wanda. Across all architectures and reduction methods, GRAIL produces a consistent upward shift in accuracy, demonstrating robust post-compression recovery.