CUT LESS, FOLD MORE: MODEL COMPRESSION THROUGH THE LENS OF PROJECTION GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Compressing neural networks without retraining is vital for deployment at scale. We study calibration-free compression through the lens of projection geometry: structured pruning is an axis-aligned projection, whereas model folding performs a low-rank projection via weight clustering. We formalize both as orthogonal operators and show that, within a rank distance of one, folding provably yields smaller parameter reconstruction error, and under mild smoothness assumptions, smaller functional perturbations than pruning. At scale, we evaluate $>$1'000 checkpoints spanning ResNet18, PreActResNet18, ViT-B/32, and CLIP ViT-B/32 on CIFAR-10 and ImageNet-1K, covering diverse training hyperparameters (optimizers, learning rates, augmentations, regularization, sharpness-aware training). We show that folding typically achieves higher post-compression accuracy, with the largest gains at moderate–high compression. The gap narrows and occasionally reverses at specific training setups. Our results position folding as a geometry-aware, calibration-free alternative to pruning that is often superior in practice and principled in theory.

1 INTRODUCTION

Neural network compression is critical for deploying models in resource-constrained environments. Common approaches include quantization, which reduces the precision of weights and activations, and knowledge distillation, which transfers information from a large teacher model to a smaller student model. In this work, we focus on the class of calibration-free post-training structured compression methods that optimize the model architecture itself without access to training data. Among these, the most widely used is *magnitude-based pruning*, which prunes tensor elements according to their magnitudes, using them as a proxy for their contribution to model accuracy (Han et al., 2015; Mishra et al., 2021; Lu et al., 2023; Ding et al., 2024; Bambhaniya et al., 2024). When combined with fine-tuning or a lightweight BatchNorm reset (Saikumar & Varghese, 2025), this approach achieves significant compression rates with negligible accuracy loss (Kurtic et al., 2022; Sanh et al., 2020). In contrast, the recently introduced *model folding* clusters similar weights and ties them together, providing an approximation of the original network (Wang et al., 2025). Both pruning and folding reduce parameter count but differ fundamentally: pruning removes weights entirely, while folding preserves them in merged representations.

In this work, we develop a unified theoretical and empirical framework to compare pruning and folding through the lens of *orthogonal projections* in parameter space. We show that both compression methods can be viewed as projections onto lower-dimensional subspaces, but with crucial differences in geometry: pruning corresponds to axis-aligned coordinate projections, while folding projects onto cluster-structured subspaces that retain directional information.

At a high level, both pruning and folding compress the weights of a model. We show that for any pruned solution there exists a folded alternative that is *almost* as small—using one extra component in the compressed representation—yet is strictly closer to the original weights (smaller Frobenius norm), which in turn bounds the change in the network function. Intuitively, folding merges weight vectors with similar directions rather than zeroing coordinates, so the compressed model stays closer in behavior to the initial network.

Empirically, we perform a comprehensive calibration-free study over $>$1'000 checkpoints spanning CNNs and ViTs on CIFAR-10 and ImageNet-1K, trained under diverse hyperparameter choices

(optimizers, learning rates, augmentation, regularization, sharpness-aware training). After compression and also followed by lightweight and full fine-tuning, folding typically attains higher post-compression accuracy, with the largest gains at moderate to high compression. The margin narrows, and can occasionally reverse, at very low compression or under specific training setups, but the overall trend is consistent with our theoretical analysis. Our projection-based perspective opens new directions for designing compression methods that explicitly optimize for functional closeness. This paper makes the following contributions:

- We introduce a unified projection framework that casts pruning and folding as orthogonal projections onto, respectively, axis-aligned and cluster-structured subspaces. We prove that at a compression rank difference of one, folding achieves smaller parameter reconstruction error and tighter function-perturbation bounds under mild smoothness assumptions.
- A large-scale evaluation across $>$1'000 checkpoints and diverse hyperparameters, covering CNNs and ViTs on CIFAR-10 and ImageNet-1K. In addition, we use post-compression fine-tuning through lightweight LayerNorm reset for ViTs, or full-fine-tuning to show that the strong performance of folding is preserved in these settings.
- We show that folding is a geometry-aware alternative that is often superior in practice, with clearly identified regimes (*e.g.*, moderate–high compression) where its advantage is most pronounced, and corner cases where the gap narrows.

Due to space constraints, a detailed discussion of related work is provided in Appendix F.

2 Unified Framework for Pruning and Folding

2.1 Preliminaries and Definitions

We consider a neural network with input $x \in \mathbb{R}^d$. We assume ReLU activations and normalization layers (*e.g.*, BatchNorm or LayerNorm) are present.

To develop the theoretical framework, we focus on compressing a single layer at a time. This layer has p inputs and m outputs with its parameters collected in matrix $\mathbf{W} \in \mathbb{R}^{m \times p}$. A row $w(i)$ of $\mathbf{W}$ is denoted as the ith parameter vector with individual weights $w(i, j)$. Since all other network parameters are treated as fixed, the network function can be expressed as $f(x; \boldsymbol{W})$, which is trained to minimize a loss function $L(\mathbf{W})$.

We assume that the loss function L is Lipschitz continuous; that is, there exists a constant $\kappa > 0$ such that

$$|L(\mathbf{W}_1) - L(\mathbf{W}_2)| \leq \kappa \, \|\mathbf{W}_1 - \mathbf{W}_2\|_F \tag{1}$$

for all admissible parameter matrices $\mathbf{W}_1$ and $\mathbf{W}_2$. The Frobenius norm of a matrix is defined as $\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$, that is, the square root of the sum of the squares of its entries, or equivalently, the ℓ_2-norm of the vectorized matrix.

Orthogonal Projection. We formalize structured pruning and model folding as orthogonal projections in parameter space. A matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ is an orthogonal projection if $\mathbf{C} = \mathbf{C}^\top = \mathbf{C}^2$, *i.e.*, it is symmetric and idempotent. Such projections map any parameter vector to its closest point (in the Euclidean norm) within a lower-dimensional subspace.

If the columns of $\mathbf{U} \in \mathbb{R}^{m \times k}$ form a basis of a k-dimensional subspace, the corresponding orthogonal projection is

$$\mathbf{C} = \mathbf{U}(\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top. \tag{2}$$

Equivalently,

$$\mathbf{C}y = \underset{z \in \mathrm{Range}(\mathbf{U})}{\arg\min} \; \|y - z\|_2$$

meaning $\mathbf{C}y$ is the orthogonal projection of y onto the subspace spanned by $\mathbf{U}$.

2.2 Compression as Orthogonal Projection

Structured pruning. Pruning can be viewed as a projection onto a coordinate-aligned subspace at the level of neurons, filters, or channels. Assume the layer outputs are ordered so that the last $m - k$ are pruned. The corresponding basis $\mathbf{U}_p$ spans the k-dimensional subspace, with projection matrix $\mathbf{C}_p$ and transformed weight matrix $\mathbf{W}_p$:

$$\mathbf{U}_p = \begin{pmatrix} I \\ 0 \end{pmatrix}, \quad \mathbf{C}_p = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{W}_p = \mathbf{C}_p \mathbf{W}. \tag{3}$$

Consequently, the last $m - k$ rows of $\mathbf{W}_p$ are zero, and the corresponding neurons, filters, or channels can be simply removed.

Model folding. Folding groups the parameters into k clusters and replaces each cluster with its mean. Depending on the choice of clusters, a different folding results. Folding can be represented as an orthogonal projection onto the k-dimensional subspace spanned by $\mathbf{U}_f \in \{0, 1\}^{m \times k}$, where each row contains exactly one nonzero entry indicating the cluster assignment. A cluster S_j comprises all indices of parameter vectors belonging to it; thus, $u_f(i, j) = 1$ if and only if $i \in S_j$.

The projection $\mathbf{C}_f$ defined in Eq. 2 maps each cluster to its mean (Wang et al., 2025). Specifically,

$$\mathbf{W}_f = \mathbf{C}_f \mathbf{W}, \quad \forall i \in S_j : \; w_f(i) = \mu_j, \quad \mu_j = \frac{1}{|S_j|} \sum_{i \in S_j} w(i), \tag{4}$$

where μ_j is the mean of cluster j. After projection, all parameter vectors within a cluster are replaced by their mean, making them identical. As a result, the corresponding layer outputs are also identical, leaving a total of k distinct neurons, filters, or channels. Practically, the identical layer outputs can be joined while adapting the next layer appropriately, see (Wang et al., 2025).

2.3 Folding Dominates Pruning

To compare pruning and folding, we first show that for any choice of pruning, there exists a folding that yields a more accurate approximation of the parameter matrix $\mathbf{W}$.

Theorem 2.1. *Given any pruning with basis $\mathbf{U}_p$ of rank $0 \leq k_p \leq m - 1$ (*i.e.*, at least one parameter vector is pruned), there exists a folding with basis $\mathbf{U}_f$ and rank $k_f = k_p + 1$ such that*

$$\|\mathbf{W} - \mathbf{W}_p\|_F^2 \geq \|\mathbf{W} - \mathbf{W}_f\|_F^2,$$

where $\mathbf{W}_p = \mathbf{C}_p \mathbf{W}$ and $\mathbf{W}_f = \mathbf{C}_f \mathbf{W}$, with $\mathbf{C}_p$ and $\mathbf{C}_f$ denoting the orthogonal projections defined in Eq. 2.

The proof is provided in Appendix C. Note that, by the Lipschitz continuity of the loss function in Eq. 1, the superior approximation property of folding implies a tighter bound on the loss difference compared to pruning:

$$\|L(\mathbf{W}) - L(\mathbf{W}_f)\|_F \leq \kappa \|\mathbf{W} - \mathbf{W}_f\|_F, \quad \|L(\mathbf{W}) - L(\mathbf{W}_p)\|_F \leq \kappa \|\mathbf{W} - \mathbf{W}_p\|_F,$$

with

$$\|\mathbf{W} - \mathbf{W}_f\|_F^2 \leq \|\mathbf{W} - \mathbf{W}_p\|_F^2.$$

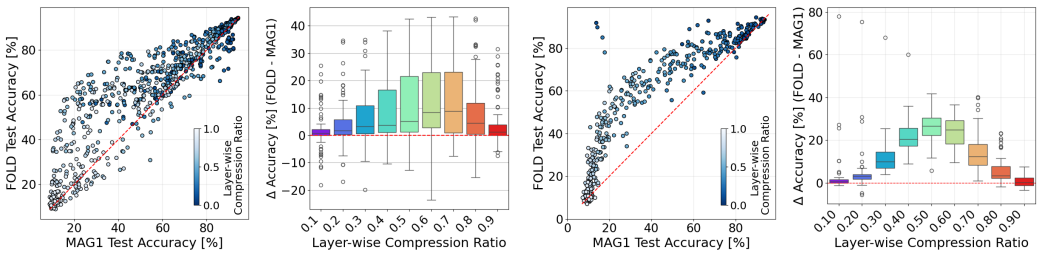
Furthermore, the rank difference $k_f = k_p + 1$ between pruning and folding is practically negligible, since in typical scenarios many parameter vectors are pruned. For instance, under a uniform 50% per-layer retention, a ResNet-18 stage with 256 channels keeps $k_p = 128$ (so folding uses $k_f = 129$), and a ViT-B/32 block with width 768 keeps $k_p = 384$ (so $k_f = 385$); the relative increase is just $1/k_p \approx 0.78\%$ and 0.26%, respectively—negligible in practice.

Finally, we show that folding using optimal k-means clustering never yields a less accurate approximation of the parameter matrix $\mathbf{W}$ than pruning.

Theorem 2.2. *Let $\mathbf{U}_f$ be the basis obtained from an optimal k-means clustering with k_f clusters,* i.e.*, the folding clusters are determined by a k-means algorithm minimizing the accumulated within-cluster sum of squares. Then, for any pruning with basis $\mathbf{U}_p$ of rank $k_p = k_f - 1$, we have*

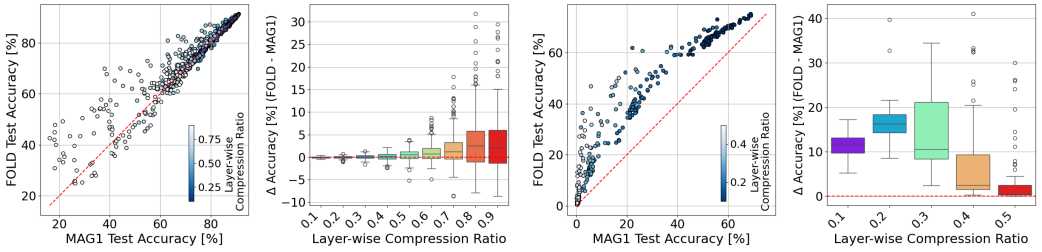
$$\|\mathbf{W} - \mathbf{W}_p\|_F^2 \geq \|\mathbf{W} - \mathbf{W}_f\|_F^2,$$

where $\mathbf{W}_p = \mathbf{C}_p \mathbf{W}$ and $\mathbf{W}_f = \mathbf{C}_f \mathbf{W}$, with $\mathbf{C}_p$ and $\mathbf{C}_f$ denoting the orthogonal projections defined in Eq. 2.



(a) ResNet18, Adam, `MAG1` vs `FOLD`, **no** L1 regularization

(b) PreActResNet18, `MAG1` vs `FOLD`



(c) ViT-B/32, `MAG1` vs `FOLD`, base accuracy >75%

(d) CLIP ViT-B/32, `MAG1` vs `FOLD`

Figure 1: **Folding outperforms magnitude pruning across diverse training regimes. Top row:** ResNet18 and PreActResNet18 on CIFAR-10. ResNet18 checkpoints were trained from scratch with Adam using different hyperparameter configurations. PreActResNet18 checkpoints are from Andriushchenko et al. (2023). **Bottom row:** ViT-B/32 on CIFAR-10 from (Andriushchenko et al., 2023) and CLIP ViT-B/32 on ImageNet-1K from (Wortsman et al., 2022). See Appendix D for details. In these plots, we use checkpoints that were trained without L1 regularization. Scatter plots show post-compression accuracy for magnitude pruning (L1 criterion) versus folding at uniform per-layer compression ratios (color-coded by layer-wise compression ratio). Bar plots depict the accuracy gain by folding, computed as $\Delta = \mathrm{Acc}(\texttt{FOLD}) - \mathrm{Acc}(\texttt{MAG1})$, as a function of layer-wise compression ratio. Folding yields the largest improvements at moderate to high compression, confirming its robustness across architectures and datasets. Fig. 9 shows the results for magnitude pruning with L2 criterion.

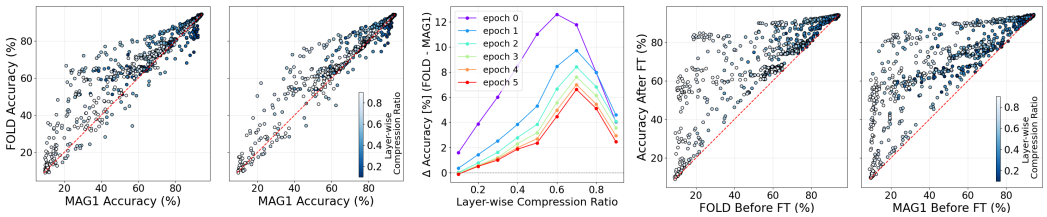
The proof is given in Appendix C. This result demonstrates that k-means folding is not merely a heuristic, but an optimal projection under clustering constraints. Unlike pruning, which relies on parameter vector removal, folding generalizes the idea by enabling coordinated parameter merging. Thus, folding incurs less parameter distortion and provably smaller functional deviation—consistent with the cross-architecture results presented in the next section.

In addition, Theorem 2.2 has implications for a possible fine-tuning after compression. Matrix $\mathbf{W}$ contains the optimized weights and $\mathbf{W}_p$ or $\mathbf{W}_f$ contain the weights after pruning and folding the optimized network. As a result of Theorem 2.2, the quadratic distance between the optimized weights and the compressed optimized weights is smaller for folding in comparison to pruning.

Our theoretical results employ a one–rank slack comparing pruning at rank k_p to folding at $k_f = k_p + 1$, as a proof device to obtain a clean monotonicity guarantee on projection error. This slack does *not* reflect our evaluation protocol. In all experiments we enforce matched sparsity budgets and compare methods at the *same* retained size (parameters and FLOPs). Hence, empirical accuracy gaps cannot be attributed to extra capacity.

3 Experimental Results

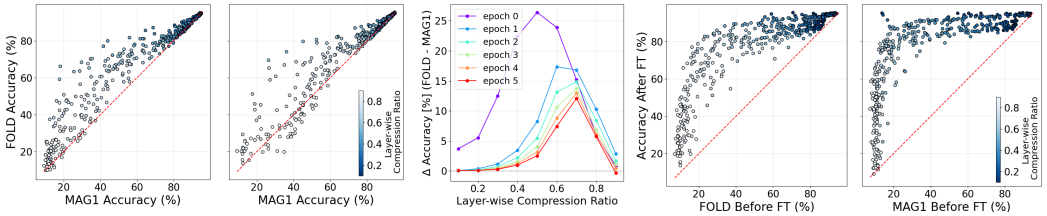
Most pruning studies vary only seeds by training several checkpoints under a single hyperparameter recipe, leaving the role of upstream training underexplored. We instead benchmark $> 1'000$ checkpoints spanning diverse hyperparameters (optimizers, learning rates, augmentation, regularization, SAM) to quantify how training choices interact with folding and pruning. Concretely, we train 216 ResNet18 (Adam) and 576 ResNet18 (SGD) models on CIFAR-10, include 50 PreActResNet18 and 200 ViT-B/32 checkpoints from Andriushchenko et al. (2023), and add 72 CLIP ViT-B/32 models



(a) MAG1 vs FOLD, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(b) MAG1 vs FOLD accuracy gap after ft.

(c) Before and after fine-tuning for 5 epochs: FOLD (**left**) and MAG1 (**right**).



(d) MAG1 vs FOLD, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(e) MAG1 vs FOLD accuracy gap after ft.

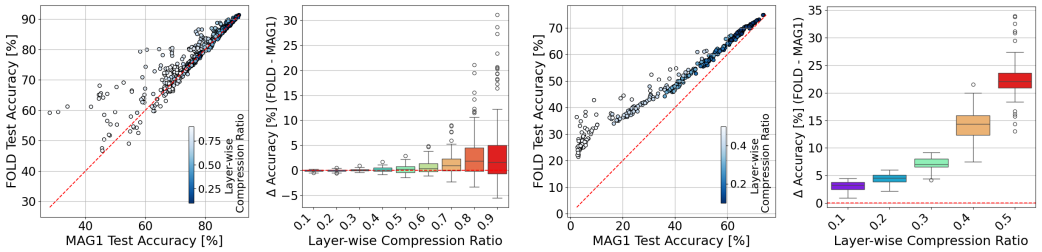
(f) Before and after fine-tuning for 5 epochs: FOLD (**left**) and MAG1 (**right**).

Figure 2: **Folded models retain their accuracy advantage after fine-tuning.** Results for ResNet18 trained by Adam (**top row**) and PreActResNet18 trained by SGD on CIFAR-10 (**bottom row**): **(a,d)** compares post-compression accuracy of magnitude pruning (MAG1) versus folding (FOLD) after 1 and 5 epochs of fine-tuning. **(b,e)** show the accuracy gap between folding and pruning as a function of fine-tuning epochs, demonstrating that folding maintains a consistent lead, *i.e.*, the FOLD accuracy delta is positive. **(c,f)** illustrate accuracy trajectories before and after 5 epochs of fine-tuning for both methods, highlighting that folded models recover accuracy faster and reach higher final performance than pruned models.

fine-tuned on ImageNet-1K from Wortsman et al. (2022). The two ViT families differ markedly in scale ($\sim$19M vs. $\sim$151M parameters).

We empirically compare model folding and structured pruning across CNNs and ViTs under matched training setups. Unless stated otherwise, we do not apply gradient-based fine-tuning: for CNNs we only re-estimate batch-normalization statistics via a single forward pass using REPAIR (Jordan et al., 2023) to isolate structural effects, and ViTs are left uncalibrated. Note that REPAIR was recently shown to substantially improve post-compression performance for pruned models (Saikumar & Varghese, 2025), and has also been applied on top of folding (Wang et al., 2025). We report results (i) immediately after compression (CNNs after REPAIR, ViTs with no further step), (ii) for ViTs additionally after a LayerNorm reset, and (iii) for both families after 1–5 epochs of full fine-tuning.

Folding vs. Structured Pruning. We begin by comparing model folding (FOLD) with structured magnitude pruning (MAG) under both L1 and L2 criteria (MAG1 and MAG2) across four representative architectures. Fig. 1 summarizes the results. In scatter plots, each point corresponds to a distinct trained model, with color indicating compression ratio. Scatter plots compare the performance of MAG1 and FOLD on every model. Comparison to MAG2 can be found in Appendix E. Box plots show the distribution of accuracy differences between the performance of FOLD and MAG1 on the same model over the layer-wise compression ratios. Positive accuracy difference means folding outperforms pruning. We observe that the accuracy difference increases with sparsity, highlighting that folding is especially advantageous at moderate-high compression. A similar trend is observed for ResNet18, PreActResNet18, ViT-B/32 and CLIP ViT-B/32. In most cases on all architectures folding dominates pruning, confirming that the benefit is robust to architectural differences (convolutional vs. transformer) and dataset scale (CIFAR-10 vs. ImageNet-1K). These findings empirically validate the theoretical claim from Sec. 2: folding, by projecting onto cluster-structured subspaces rather than axis-aligned ones, preserves alignment with the original parameter space and induces smaller functional distortions. The observed accuracy improvements thus provide strong evidence that geometry-aware projections offer a principled advantage over traditional magnitude-based pruning.



(a) ViT-B/32, MAG1 vs FOLD, base accuracy >75%

(b) CLIP ViT-B/32, MAG1 vs FOLD

Figure 3: **MAG1 versus FOLD on ViTs after LayerNorm-only fine-tuning** for ViT-B/32 on CIFAR-10 and CLIP ViT-B/32 on ImageNet-1K. In the scatter plots, points are checkpoints, color encodes layer-wise compression. Bar plots depict the accuracy gain $\Delta = \mathrm{Acc}(\texttt{FOLD}) - \mathrm{Acc}(\texttt{MAG1})$, which remains positive and typically grows with compression, indicating that even under lightweight LayerNorm adaptation FOLD retains a consistent advantage over pruning.

Performance Comparison after Lightweight and Full Fine-Tuning. The results above isolate structural effects by evaluating models without additional optimization. We now ask whether folding's advantage persists with post-compression fine-tuning. Focusing on CNNs, we fine-tune folded and pruned models for 1–5 epochs and compare recovery. Fig. 2 shows that (a,d) folded models start from higher accuracy and retain their lead at 1 and 5 epochs, (b,e) the relative accuracy gap remains positive, and (c,f) learning curves recover faster with fewer plateaus. Consistent with the projection view, folding preserves more of the pre-compression function, yielding a better initialization that requires fewer updates to reach high accuracy, making it attractive in pipelines with limited fine-tuning.

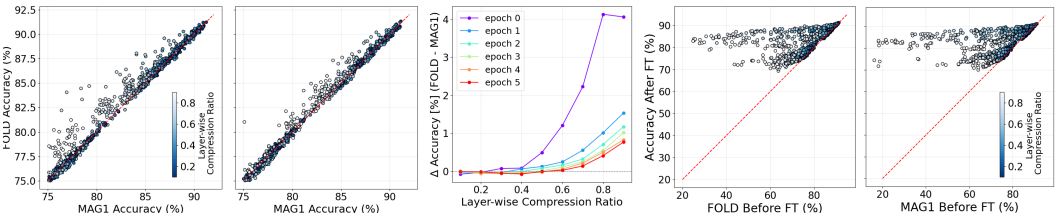
Fig. 3 compares MAG1 and FOLD on ViTs under the lightweight LayerNorm-only adaptation. Across ViT-B/32 (CIFAR-10) and CLIP ViT-B/32 (ImageNet-1K), folding consistently achieves higher post-compression accuracy after a LayerNorm reset, with the accuracy gap $\Delta = \mathrm{Acc}(\texttt{FOLD}) - \mathrm{Acc}(\texttt{MAG1})$ remaining positive and typically widening as compression ratio increases. This indicates that even with the lightweight LayerNorm recalibration, folding preserves more of the pre-compression function than structured pruning. We then allow short-horizon fine-tuning and assess whether the advantage persists. As shown in Fig. 4, after 1 and 5 epochs, folded ViTs retain a clear lead over MAG1, and the gap Δ stays positive over training. Learning curves reveal faster recovery and higher end-point accuracy for folding, suggesting a better initialization that requires fewer updates.

4 Model Compression Ablation Studies

The previous sections demonstrated that folding often outperforms structured pruning across architectures and compression ratios. Here, we probe which training factors impact this advantage. Specifically, we analyze sensitivity to learning rate, the use of sharpness-aware training (SAM) (Foret et al., 2021), regularization and data augmentation (Prabhu et al., 2019)—the hyperparameters known to influence loss landscape geometry and generalization (Fort & Jastrzebski, 2019; Li et al., 2018; Neyshabur et al., 2017; Chen et al., 2022) in non-trivial ways (Andriushchenko et al., 2023).

Role of Optimizer. We repeat the ResNet18 analysis under Adam and SGD to gauge optimizer sensitivity. Compared to the Adam-trained sweep in Fig. 1(a), the complementary SGD sweep in Fig. 6 shows the same qualitative ordering—FOLD exceeds MAG1 across compression levels—but with different baselines and dispersion: SGD checkpoints form a tighter cloud and exhibit a smaller median gap, whereas Adam yields larger variance and at times a more pronounced FOLD advantage, especially at higher compression. Together, these plots indicate that the optimizer changes *how much* headroom folding has, not *whether* it leads: the FOLD–MAG1 difference remains positive under both optimizers, but its magnitude is optimizer-dependent.

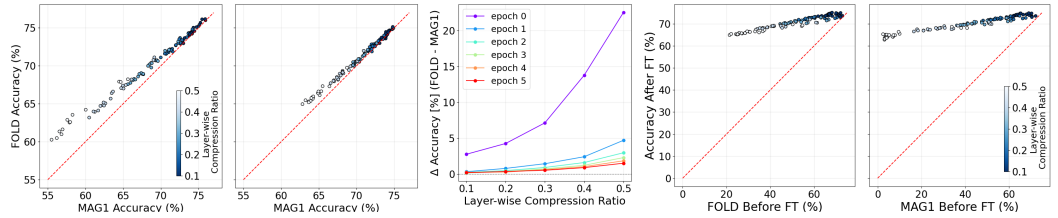
Effect of Learning Rate. Fig. 5 reports post-compression accuracy for FOLD versus MAG1 across learning rates on ResNet18 (Adam, SGD), PreActResNet18, and ViT-B/32. With Adam, FOLD 's edge is largest at moderate–low rates, narrows and can reverse at very high rates, and vanishes again at extremely small rates (both methods degrade). For SGD, the dependence is weaker and can be inverted (*e.g.*, ViT-B/32). A plausible explanation is that moderate learning rates steer training toward flatter,



(a) MAG1 vs FOLD, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(b) MAG1 vs FOLD accuracy gap after ft.

(c) Before and after fine-tuning for 5 epochs: FOLD (**left**) and MAG1 (**right**).



(d) MAG1 vs FOLD, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(e) MAG1 vs FOLD accuracy gap after ft.

(f) Before and after fine-tuning for 5 epochs: FOLD (**left**) and MAG1 (**right**).

Figure 4: **FOLD outperforms MAG1 after full fine-tuning for 1–5 epochs on ViT-B/32 and CLIP ViT-B/32.** Results for ViT-B/32 on CIFAR-10 (**top**) and CLIP ViT-B/32 on ImageNet-1K (**bottom**). **(a,d)** accuracy of MAG1 vs. FOLD after 1 and 5 epochs of fine-tuning. **(b,e)** accuracy gap Δ over epochs, remaining positive. **(c,f)** accuracy trajectories from post-compression through 5 epochs, showing faster recovery and higher final accuracy for FOLD.

more structured solutions with stronger within-layer correlations—favorable for clustering—whereas very high rates yield sharper, less-aligned solutions and very small rates underfit. Adaptive methods like Adam are further associated with sharper minima and distinct generalization behavior compared to SGD, amplifying this sensitivity (Wilson et al., 2018; Jastrzębski et al., 2018; Zhou et al., 2021).

Effect of SAM. Fig. 7 evaluates training with and without SAM and measures post-compression accuracy. Across models, SAM lifts both methods, but the gain is systematically larger for FOLD, widening the FOLD–MAG1 gap—most visibly for Adam-trained ResNet18. With light L1 regularization (10^{-5}) during training shown in (b), pruning narrows the gap at *low* compression (where induced sparsity aligns with L1), yet FOLD regains and extends its lead as compression increases. These trends are consistent with the view that SAM steers training to flatter solutions, reducing curvature sensitivity. Combined with FOLD 's smaller projection error, this yields greater robustness to compression. At larger SAM radii ρ, training enforces robustness to broader parameter perturbations. Within this flatter neighborhood both pruning and folding projections operate inside the same robustness ball, so their geometric differences matter less and the gap narrows—an effect stronger for ViT-B/32, where high ρ homogenizes head / channel saliencies and reduces the relative advantage of clustering.

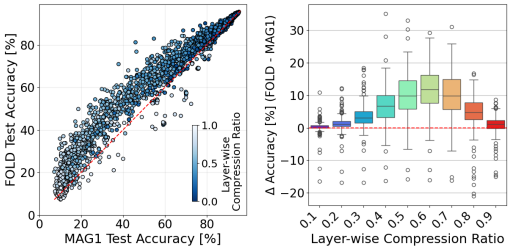
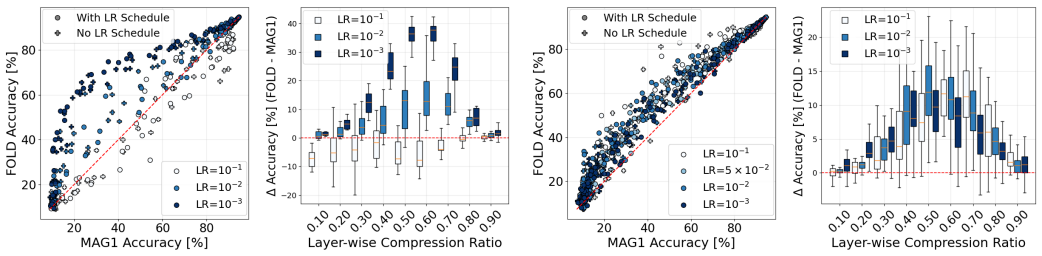


Figure 6: **Optimizer effect** evaluated on ResNet18 checkpoints trained on CIFAR-10 with SGD (no L1 normalization). The figure complements Fig. 1(a).

Effect of Data Augmentation. Fig. 8 plots the distribution of ΔAccuracy (FOLD $-$ MAG1) across checkpoints versus the layer-wise compression ratio, contrasting runs without (gray) and with RandAugment (green). For ResNet18 (Adam and SGD) and PreActResNet18, RAUG reduces or shifts FOLD 's relative benefit. In contrast, for ViT-B/32 RAUG increases FOLD 's advantage: the median Δ rises with compression, suggesting that augmented ViT representations are especially amenable to projection-based removal. A plausible mechanism is that augmentation biases training toward flatter, more invariant solutions. This is consistent with recent theory linking augmentation-induced input perturbations to equivalent parameter-space perturbations and showing that augmentations bias



(a) ResNet18, Adam, **no** L1 reg., **no** weight decay

(b) ResNet18, SGD, **no** L1 reg., **no** weight decay

(c) PreActResNet18, SGD

(d) ViT-B/32, SGD

Figure 5: **Learning rate modulates folding's edge.** Post-compression accuracy of `FOLD` and `MAG1` across learning rates: ResNet18 with Adam **(a)** and SGD **(b)**, PreActResNet18 **(c)**, and ViT-B/32 **(d)**. `FOLD` typically leads at moderate–low rates; the gap shrinks or reverses at very high rates, and closes again at extremely small rates. The effect is strongest with Adam; SGD shows weaker or occasionally opposite dependence.

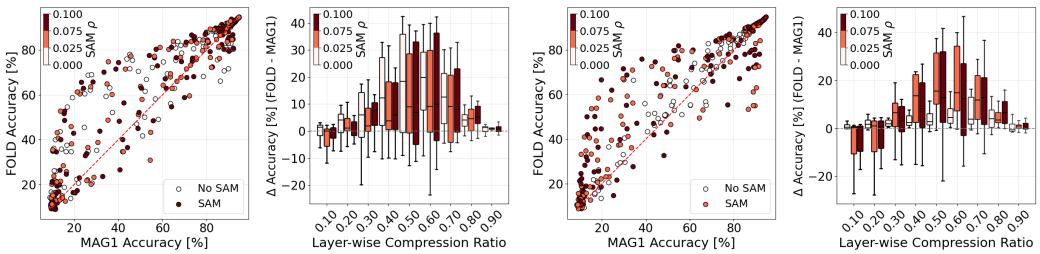
training toward flatter minima (Yoo & Yoon, 2025). In CNNs this reduces the harm of axis-aligned magnitude cuts, whereas in ViTs the same invariances tighten feature clusters that `FOLD` preserves better than `MAG1`, amplifying the benefit at high compression. Standard augmentation (`augm=True`) shows a similar trend and is omitted for brevity.

These ablations reveal a consistent pattern: conditions that encourage flatter and structured solutions—moderately low learning rates and SAM with a small–moderate radius—magnify `FOLD` 's advantage, whereas extremes reduce it: very high or very low learning rates, stronger augmentations, or large SAM radii narrow the gap; SGD generally dampens all effects relative to Adam. This aligns with our projection view (Sec. 2): when weights are well aligned, clustering reduces projection error more than coordinate removal and thus perturbs the function less, while weaker alignment or broad robustness neighborhoods make the two projections behave more similarly.

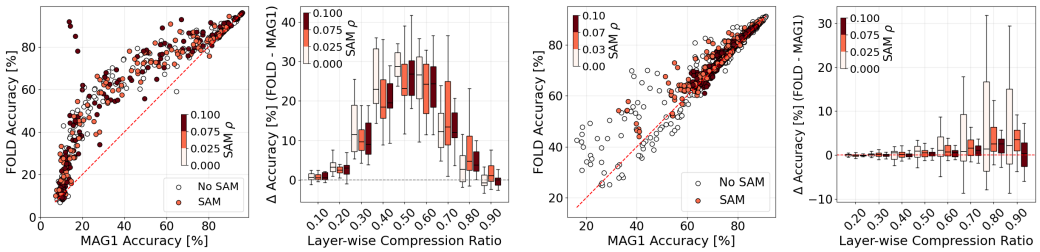
5 Conclusion, Limitations, and Outlook

We framed structured pruning and model folding as projection-based compression and showed that folding achieves smaller parameter deviation with a one-rank slack, implying tighter functional preservation under mild smoothness. A calibration-free evaluation over $>1'000$ checkpoints (ResNet18, PreActResNet18, ViT-B/32, CLIP ViT-B/32; CIFAR-10, ImageNet-1K) found that `FOLD` typically surpasses `MAG1` in post-compression accuracy, with the clearest gains at moderate–high compression and under training conditions that induce flatter, more structured solutions (*e.g.*, moderate learning rates, SAM). The gap narrows at very low compression and can shrink under strong data augmentation or large SAM radii, but the overall trend is robust across optimizers and hyperparameters.

Limitations. Our theoretical guarantee permits a one-component increase in compressed rank and does not establish universal dominance at exactly matched sizes. Empirically, we restrict to standard CNN/ViT families on CIFAR-10 and ImageNet-1K, evaluate strictly calibration-free settings with optional BatchNorm / LayerNorm resets and short fine-tuning budgets, and compare primarily against magnitude-based structured pruning. We do not study interactions with quantization, distillation, or unstructured sparsity. Large language models are out of scope: effective structured pruning for LLMs typically relies on calibration data (activation-aware / second-order criteria), while strictly

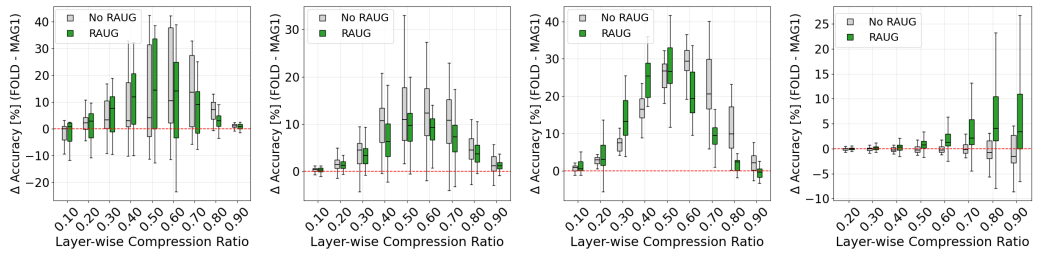


(a) ResNet18, Adam, `MAG1` vs `FOLD`, **without** L1 reg. (b) ResNet18, Adam, `MAG1` vs `FOLD`, **with** L1 reg.



(c) PreActResNet, `MAG1` vs `FOLD`, **no** L1 regularization (d) ViT-B/32, `MAG1` vs `FOLD`, **no** L1 regularization

Figure 7: **SAM (Foret et al., 2021) can boost model compression.** Post-compression accuracy under training with / without SAM. **(a)** ResNet18 (Adam), no L1. **(b)** ResNet18 (Adam), L1$= 10^{-5}$. **(c)** PreActResNet18 (SGD), no L1. **(d)** ViT-B/32, no L1. SAM improves both `FOLD` and `MAG1`, but the uplift is consistently larger for `FOLD`, especially with Adam. Light L1 regularization helps `MAG1` at low compression, yet `FOLD` retains a clear advantage at moderate–high compression.



(a) ResNet18, Adam (b) ResNet18, SGD (c) PreActResNet (d) ViT-B/32

Figure 8: **Augmentations have a generally positive effect on the post-compression accuracy.** Post-compression accuracy without / with random augmentations for **(a)** ResNet18 (Adam), **(b)** ResNet18 (SGD), **(c)** PreActResNet18, and **(d)** ViT-B/32. Augmentations boost both `FOLD` and `MAG1`. On ResNet18 they also narrow `FOLD` 's advantage—most noticeably at moderate compression—consistent with added invariances making axis-aligned removals less damaging. In ViT-B/32, augmentations are essential for folding[1].

data-free magnitude pruning is brittle (often collapsing perplexity), leaving no fair calibration-free baseline for folding. Moreover, LLM-specific deployment metrics (*e.g.*, KV-cache, sequence length) are orthogonal to our vision protocol.

Outlook. We plan to extend folding to calibration-based settings and evaluate on LLMs / VLMs with appropriate metrics (perplexity, zero-shot accuracy). We also plan to study interactions with quantization and adaptation methods. More broadly, our projection-based view positions folding as a geometry-aware primitive for compression: a foundation on which hybrid pipelines with quantization and distillation can be built, and a step toward principled frameworks that unify efficiency and functional preservation. In this sense, folding is not only a practical tool but also a building block for the next generation of compression methods tailored to foundation models and deployment at scale.

[1]Note that the base accuracy of ViT-B/32 checkpoints trained without RAUG is lower than with RAUG.

Reproducibility Statement. Our compression operators and evaluation protocol are described in Sec. 2-Sec. 3, with ablation studies in Sec. 4. Complete proofs are in Appendix C; training setups, datasets, links to the used checkpoints, and hyperparameter grids in Appendix D; and extended results in Appendix E. An anonymous repository with configs and scripts to regenerate all figures/tables is linked in Appendix A; our limited LLM usage statement is in Appendix B. Together, these materials enable re-running the full pipeline and regenerating the results.

REFERENCES

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.

Abhimanyu Rajeshkumar Bambhaniya, Amir Yazdanbakhsh, Suvinay Subramanian, Sheng-Chun Kao, Shivani Agrawal, Utku Evci, and Tushar Krishna. Progressive gradient flow for robust n:m sparsity training in transformers, 2024. URL https://arxiv.org/abs/2402.04744.

Christian Bauckhage. K-means clustering is matrix factorization. arXiv:1512.07548, 2015.

Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations, 2022. URL https://arxiv.org/abs/2106.01548.

Bita Darvish Rouhani, Daniel Lo, Ritchie Zhao, Ming Liu, Jeremy Fowers, Kalin Ovtcharov, Anna Vinogradsky, Sarah Massengill, Lita Yang, Ray Bittner, Alessandro Forin, Haishan Zhu, Taesik Na, Prerak Patel, Shuai Che, Lok Chand Koppaka, XIA SONG, Subhojit Som, Kaustav Das, Saurabh T, Steve Reinhardt, Sitaram Lanka, Eric Chung, and Doug Burger. Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10271–10281. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/747e32ab0fea7fbd2ad9ec03daa3f840-Paper.pdf.

Shaojin Ding, David Qiu, David Rim, Yanzhang He, Oleg Rybakov, Bo Li, Rohit Prabhavalkar, Weiran Wang, Tara N. Sainath, Zhonglin Han, Jian Li, Amir Yazdanbakhsh, and Shivani Agrawal. Usm-lite: Quantization and sparsity aware fine-tuning for speech recognition with universal speech models, 2024. URL https://arxiv.org/abs/2312.08553.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2021. URL https://arxiv.org/abs/2010.01412.

Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes, 2019. URL https://arxiv.org/abs/1906.04724.

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

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. URL https://arxiv.org/abs/1506.02626.

J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd, 2018. URL https://arxiv.org/abs/1711.04623.

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.

Hyeong-Ju Kang. Accelerator-aware pruning for convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2020. ISSN 1558-2205. doi: 10.1109/tcsvt.2019.2911674. URL http://dx.doi.org/10.1109/TCSVT.2019.2911674.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models, 2022. URL https://arxiv.org/abs/2203.07259.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 6391–6401, Red Hook, NY, USA, 2018. Curran Associates Inc.

Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. Spp: Sparsity-preserved parameter-efficient fine-tuning for large language models, 2024. URL https://arxiv.org/abs/2405.16057.

Yucheng Lu, Shivani Agrawal, Suvinay Subramanian, Oleg Rybakov, Christopher De Sa, and Amir Yazdanbakhsh. Step: Learning n:m structured sparsity masks from scratch with precondition, 2023. URL https://arxiv.org/abs/2302.01172.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks, 2021. URL https://arxiv.org/abs/2104.08378.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning, 2017. URL https://arxiv.org/abs/1706.08947.

Vinay Uday Prabhu, Dian Ang Yap, Joyce Xu, and John Whaley. Understanding adversarial robustness through loss landscape geometries, 2019. URL https://arxiv.org/abs/1907.09061.

Sai Qian Zhang, Bradley McDanel, and H. T. Kung. Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 846–860, 2022. doi: 10.1109/HPCA53966.2022.00067.

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. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Dhananjay Saikumar and Blesson Varghese. Signal collapse in one-shot pruning: When sparse models fail to distinguish neural representations, 2025. URL https://arxiv.org/abs/2502.15790.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning, 2020. URL https://arxiv.org/abs/2005.07683.

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.

Aaquib Syed, Phillip Huang Guo, and Vijaykaarti Sundarapandiyan. Prune and tune: Improving efficient pruning techniques for massive language models, 2023. URL https://openreview.net/forum?id=cKlgcx7nSZ.

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.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 2018. URL https://arxiv.org/abs/1705.08292.

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.

Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. Balanced sparsity for efficient dnn inference on gpu. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5676–5683, July 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33015676. URL http://dx.doi.org/10.1609/aaai.v33i01.33015676.

Weebum Yoo and Sung Whan Yoon. A flat minima perspective on understanding augmentations and model robustness, 2025. URL https://arxiv.org/abs/2505.24592.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Hoi, and Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning, 2021. URL https://arxiv.org/abs/2010.05627.

APPENDIX

The following sections provide supplementary information and complement the main paper:

- Appendix A: Code, Data, and Resources.
- Appendix B: Use of Large Language Models.
- Appendix C: Proofs of Theoretical Claims.
- Appendix D: Training Details.
- Appendix E: Further Empirical Results.
- Appendix F: Related Work.

A CODE, DATA, AND RESOURCES

Code and logs. An anonymous repository with all source code, experiment configs, and figure-generation scripts (including the exact logs used to render every plot/table) are released at `https://anonymous.4open.science/r/folding_as_projection-7A4D`. The repo contains: implementations of folding and pruning operators, training/evaluation pipelines, scripts to plot ablations, and notebooks to reproduce figures directly from logs. We log all training metrics and hyperparameters with Weights & Biases[2] and export logs alongside the code for reproduction.

Our folding implementation is based on the code by Wang et al. (2025)[3].

Datasets. We use CIFAR-10[4] and ImageNet-1K[5]. CIFAR-10 is downloaded automatically via `torchvision`. ImageNet-1K requires the official credentials and follows its license. Pretrained/fine-tuned checkpoints referenced in the paper are either trained by us (configs in the repo) or obtained from the cited works (Andriushchenko et al., 2023; Wortsman et al., 2022). The download links are also provided in Appendix D.

Compute resources. Experiments were run on a cluster featuring 8× NVIDIA A100 (80 GB RAM) GPUs. All random seeds are fixed in the configs and scripts.

Computational complexity and memory cost. At inference and matched retained sizes, folding and structured pruning yield the same compute and memory. The difference lies in the compression step: magnitude pruning is a one-pass scoring and selection procedure ($O(pm)$ to score p filters of dimension m, plus $O(p \log p)$ selection), whereas folding runs k-means on layer weights with T sweeps. Using Hartigan's algorithm (Hartigan & Wong, 1979), one sweep costs $O(pkm)$, with max $T = 10$ sweeps the total is $O(pkmT)$ (effectively linear in pm when k is small). This cost is paid once per layer and is small compared to training.

Runtime overview. The most expensive step in our study is fine-tuning of CLIP VIT-B/32 on ImageNet-1K (1–5 epochs), which dominates wall-clock time (order of hours per run). In contrast, compression is lightweight: on CPU, `FOLD` takes ∼5–12 s per ResNet18 checkpoint and ∼8–12 s per ViT-B/32 (per-layer 50% removal).

B USE OF LARGE LANGUAGE MODELS

We used ChatGPT [6] for sentence-level grammar correction and improvement, drafting trivial plotting snippets to produce figures from logs, and code readability edits. All ideas, proofs, experiments, and analyses are ours.

[2] Weights & Biases: `https://wandb.ai`

[3] Model folding universal: `https://github.com/nanguoyu/model-folding-universal` and model folding for CNNs: `https://github.com/marza96/ModelFolding/`

[4] CIFAR-10: `https://www.cs.toronto.edu/~kriz/cifar.html`

[5] ImageNet-1K: `https://image-net.org/`

[6] ChatGPT / GPT-5: `https://chatgpt.com`

C PROOFS OF THEORETICAL CLAIMS

Below we prove that for any choice of pruning, there exists a folding that yields a more accurate approximation of the parameter matrix $\mathbf{W}$.

Theorem 2.1. *Given any pruning with basis $\mathbf{U}_p$ of rank $0 \leq k_p \leq m-1$ (i.e., at least one parameter vector is pruned), there exists a folding with basis $\mathbf{U}_f$ and rank $k_f = k_p + 1$ such that*

$$\|\mathbf{W} - \mathbf{W}_p\|_F^2 \geq \|\mathbf{W} - \mathbf{W}_f\|_F^2,$$

where $\mathbf{W}_p = \mathbf{C}_p\mathbf{W}$ and $\mathbf{W}_f = \mathbf{C}_f\mathbf{W}$, with $\mathbf{C}_p$ and $\mathbf{C}_f$ denoting the orthogonal projections defined in Eq. 2.

Proof. The rows of $\mathbf{W}$ can be ordered such that the pruned parameter vectors are first: $w(1), ..., w(m-k_p)$. Then we find that

$$\mathbf{W} - \mathbf{W}_p = \begin{pmatrix} w(1) \\ \cdots \\ w(m-k_p) \\ 0 \\ \cdots \\ 0 \end{pmatrix}$$

using Eq. 3. For the existence proof, we choose a folding that clusters all parameter vectors $w(1), ..., w(m-k_p)$ into a single cluster, all other parameter vectors have individual clusters, *i.e.*,

$$\mathbf{U}_f = \begin{pmatrix} 1 & 0 \\ \cdots & 0 \\ 1 & 0 \\ 0 & \mathbf{I} \end{pmatrix} \quad ; \quad \mathbf{W} - \mathbf{W}_f = \begin{pmatrix} w(1) - \mu \\ \cdots \\ w(m-k_p) - \mu \\ 0 \\ \cdots \\ 0 \end{pmatrix} \quad ; \quad \mu = \frac{1}{m-k_p} \sum_{i=1}^{m-k_p} w(i)$$

using Eq. 4.

We have $\|\mathbf{W} - \mathbf{W}_p\|_F^2 = \sum_{i=1}^{m-k_p} w(i)^T w(i)$ and

$$\begin{aligned} \|\mathbf{W} - \mathbf{W}_f\|_F^2 &= \sum_{i=1}^{m-k_p} (w(i) - \mu)^T (w(i) - \mu) = \sum_{i=1}^{m-k_p} \left(w(i)^T w(i) - 2w(i)^T \mu + \mu^T \mu \right) \\ &= \sum_{i=1}^{m-k_p} w(i)^T w(i) - (m-k_p)\mu^T \mu \\ &\leq \sum_{i=1}^{m-k_p} w(i)^T w(i) = \|\mathbf{W} - \mathbf{W}_p\|_F^2 \end{aligned}$$

The latter inequality directly establishes the theorem. □

The following theorem shows that folding using optimal k-means clustering never yields a less accurate approximation of the parameter matrix $\mathbf{W}$ than pruning.

Theorem 2.2. *Let $\mathbf{U}_f$ be the basis obtained from an optimal k-means clustering with k_f clusters,* i.e., *the folding clusters are determined by a k-means algorithm minimizing the accumulated within-cluster sum of squares. Then, for any pruning with basis $\mathbf{U}_p$ of rank $k_p = k_f - 1$, we have*

$$\|\mathbf{W} - \mathbf{W}_p\|_F^2 \geq \|\mathbf{W} - \mathbf{W}_f\|_F^2,$$

where $\mathbf{W}_p = \mathbf{C}_p\mathbf{W}$ and $\mathbf{W}_f = \mathbf{C}_f\mathbf{W}$, with $\mathbf{C}_p$ and $\mathbf{C}_f$ denoting the orthogonal projections defined in Eq. 2.

Proof. According to Bauckhage (2015) and Wang et al. (2025), the problem of k-means clustering can be formulated as the following constrained matrix factorization problem:

$$\min_{\mathbf{U}} \left\|\mathbf{W} - \mathbf{U}(\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{W}\right\|_F^2 \quad \text{subject to} \quad u(i,j) \in \{0,1\},\ \sum_j u(i,j) = 1\ \forall i.$$

This formulation coincides with the orthogonal projection of model folding, see Eq. 2 and Eq. 4. Theorem 2.1 guarantees the existence of a folding basis $\mathbf{U}_f$ and the corresponding projection $\mathbf{C}_f$ for any pruning $\mathbf{W}_p$ of $\mathbf{W}$, such that

$$\|\mathbf{W} - \mathbf{W}_p\|_F^2 \ \geq\ \|\mathbf{W} - \mathbf{W}_f\|_F^2.$$

Since optimal k-means clustering achieves the minimal possible error $\|\mathbf{W} - \mathbf{W}_f\|_F^2$, the theorem follows. □

D Training Details

The following subsections detail the hyperparameters used to train our checkpoints. For checkpoints taken from the literature, we summarize the available training details.

D.1 ResNet18 on CIFAR-10 Training Setup with Adam and SGD

We trained a total of 792 ResNet18 models on CIFAR-10 by varying hyperparameter configurations. We used two optimizers: Adam and SGD. Tab. 1 summarizes the parameter combinations explored for each optimizer. For Adam, we used 3 learning rates and 1 momentum value. For SGD, we used 3 learning rates and 2 momentum values. The remaining parameters were shared across both optimizers: weight decay (3 values), L1 regularization (2 values), RandAugment (2 values), Sharpness-Aware Minimization (3 values), and learning rate scheduling (2 values). This resulted in 216 models trained with Adam and 576 models trained with SGD. In the ablation studies, we filter checkpoints (as specified in the figure captions) to highlight the observed effects.

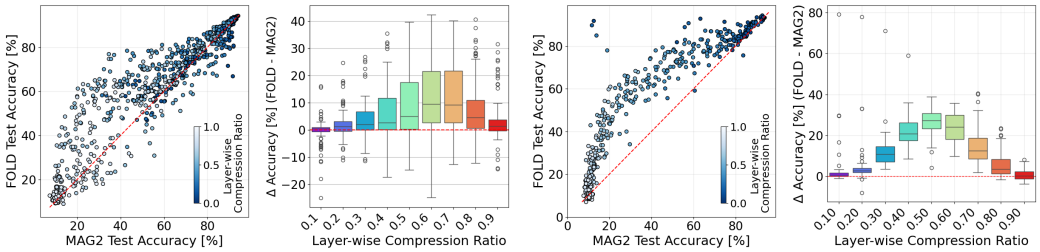
Parameter	**Values**
Optimizer	`adam`, `sgd`
Learning Rate	`adam`: 0.1, 0.01, 0.001 `sgd`: 0.1, 0.05, 0.01, 0.001
Momentum	`adam`: 0.0 `sgd`: 0.9, 0.99
Weight Decay	0.0, 0.0005, 0.001
L1 Regularization	0.0, 1×10^{-5}
RandAugment	True, False
SAM (Sharpness-Aware Minimization)	None, 0.05, 0.1
Learning Rate Schedule	True, False

Table 1: Hyperparameter combinations used for ResNet18 training on CIFAR-10.

D.2 PreActResNet18 on CIFAR-10

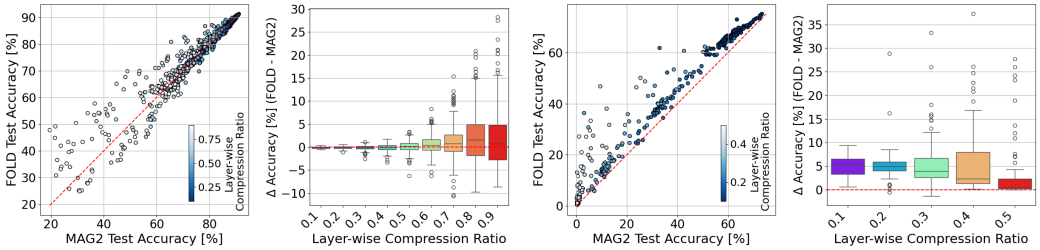
We use 50 trained PreActResNet18 models on CIFAR-10 from Andriushchenko et al. (2023)[7]. The models are trained using a fixed set of training parameters and a sweep over a few key hyperparameters. Tab. 2 summarizes varied parameters used in this experiment. All checkpoints used the same training protocol: 200 epochs, batch size 128, and no label noise. The model width was fixed at 64 and the learning rate schedule followed a cyclic pattern. Only the maximum learning rate (`lr_max`), SAM strength (`sam_rho`), and augmentation settings were varied. For the learning rate ablation studies, we adopt the reported maximum learning rate.

[7]Download link: `https://drive.google.com/drive/folders/1LmthJCb3RXBFWjeTOC4UOOl7Ppgg2h7n`



(a) ResNet18, Adam, FOLD vs MAG2, **no** L1 regularization

(b) PreActResNet18, FOLD vs MAG2



(c) ViT-B/32, FOLD vs MAG2

(d) CLIP ViT-B/32, FOLD vs MAG2

Figure 9: **Folding outperforms magnitude pruning across diverse training regimes.** The same setup as in Fig. 1, but compared to the L2 magnitude pruning criterion. **Top row:** ResNet18 and PreActResNet18 on CIFAR-10. ResNet18 checkpoints were trained from scratch with Adam using different hyperparameter configurations. **Bottom row:** ViT-B/32 on CIFAR-10 and CLIP ViT-B/32 on ImageNet-1K. Scatter plots show post-compression accuracy for folding versus magnitude pruning (L2 criterion) at uniform per-layer compression ratios. Bar plots depict the accuracy gain by folding, computed as $\Delta = \mathrm{Acc}(\texttt{FOLD}) - \mathrm{Acc}(\texttt{MAG2})$, as a function of layer-wise compression ratio. Folding yields the largest improvements at moderate to high compression, confirming its robustness across architectures and datasets.

Table 2: Fixed and varying parameters for PreActResNet18 training on CIFAR-10.

Parameter	**Values**
Optimizer	`sgd`
Max / Base Learning Rate (`lr_max`)	from 0.0504 to 4.9759
SAM Strength (`sam_rho`)	0.0, 0.05, 0.1
Standard Augmentation (`augm`)	True, False
RandAugment (`randaug`)	True, False

D.3 ViT-B/32 on CIFAR-10

The 200 Vision Transformers (ViT) also from Andriushchenko et al. (2023), width=256, were trained on CIFAR-10, batch size 128, for 200 epochs with a cosine learning rate schedule and linear warmup. The main hyperparameters are summarized in Tab. 3. We made use of the maximum learning rate, the use of data augmentation, and the use of Sharpness-Aware Minimization (SAM) in our evaluations. All other settings were fixed.

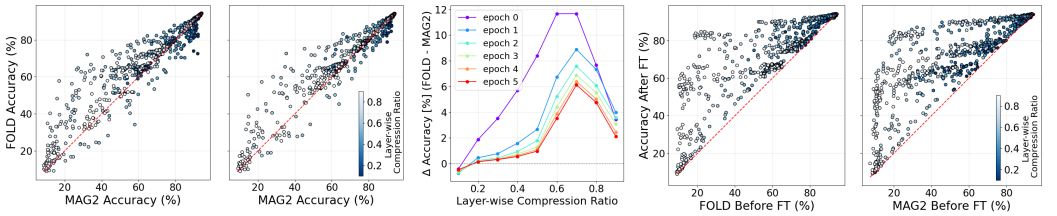
D.4 CLIP ViT-B/32 on ImageNet-1K

CLIP (Radford et al., 2021) models are known for the widespread use of CLIP features (Ramesh et al., 2022). We use the pool of models introduced by Wortsman et al. (2022), who fine-tuned the CLIP ViT-B/32 architecture on ImageNet-1K multiple times using different randomly sampled training hyperparameters[8]. These hyperparameters include learning rate, number of training epochs, weight decay, label smoothing, and augmentation strategies, as stated in (Wortsman et al., 2022). The

[8]Download link: `https://github.com/mlfoundations/model-soups/releases/`

Table 3: Fixed and varying parameters for ViT-B/32 Base training on CIFAR-10.

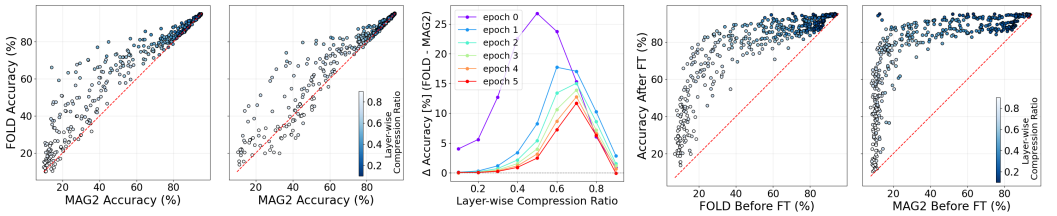
Parameter	**Values**
Optimizer	`sgd`
Max / Base Learning Rate (`lr_max`)	from 0.005087 to 0.492936
SAM Strength (`sam_rho`)	0.0, 0.05, 0.1
Standard Augmentation (`augm`)	True, False
RandAugment (`randaug`)	True, False



(a) `MAG1` vs `FOLD`, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(b) `FOLD` vs `MAG1` accuracy gap after ft.

(c) Before and after fine-tuning for 5 epochs: `FOLD` (**left**) and `MAG1` (**right**).



(d) `MAG1` vs `FOLD`, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(e) `FOLD` vs `MAG1` accuracy gap after ft.

(f) Before and after fine-tuning for 5 epochs: `FOLD` (**left**) and `MAG1` (**right**).

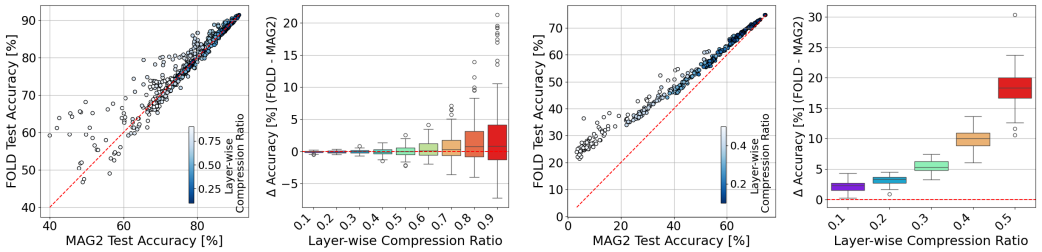
Figure 10: **Folded models retain their accuracy advantage after fine-tuning.** Results for ResNet18 trained by Adam (**top row**) and PreActResNet18 trained by SGD on CIFAR-10 (**bottom row**): **(a,d)** compares post-compression accuracy of magnitude pruning with L2 criterion (`MAG2`) versus folding (`FOLD`) after 1 and 5 epochs of fine-tuning. **(b,e)** show the accuracy gap between folding and pruning as a function of fine-tuning epochs, demonstrating that folding maintains a consistent lead, *i.e.*, the `FOLD` accuracy delta is positive. **(c,f)** illustrate accuracy trajectories before and after 5 epochs of fine-tuning for both methods, highlighting that folded models recover accuracy faster and reach higher final performance than pruned models. The figure extends Fig. 2 in the main paper to `MAG2`.

resulting collection of 72 fine-tuned models provides a strong basis for evaluating the performance of model folding compared to pruning on CLIP ViT architectures. All checkpoints were evaluated jointly in our study, without parameter-specific ablations.

E Further Results

We provide additional experiments to complement the main results. Fig. 9 mirrors the setup of Fig. 1 in the main paper, but replaces the L1 criterion for magnitude pruning with L2 (`MAG2`). Similarly, Fig. 10, Fig. 11, and Fig. 12 extend the corresponding figures in the main paper to the L2 case. Across all comparisons, the qualitative picture remains the same: `FOLD` consistently matches or outperforms magnitude pruning, independent of the chosen norm.

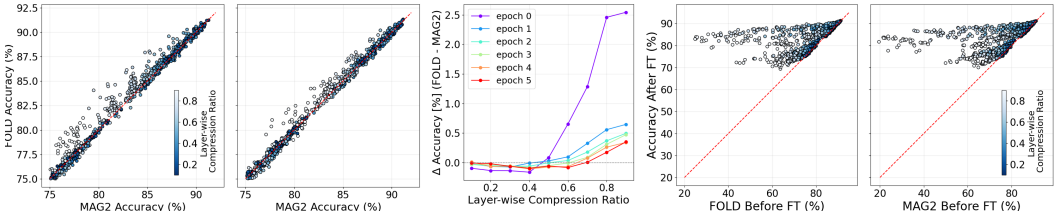
We further include ablations to study the robustness of these findings with respect to training hyperparameters. Fig. 13, Fig. 14, and Fig. 15 report the effect of varying learning rate, SAM strength, and RandAugment, respectively. Finally, Fig. 16 shows the influence of weight decay. Taken together, these studies confirm that the relative advantage of `FOLD` is stable across different regularization strategies and training configurations.



(a) ViT-B/32, MAG2 vs FOLD, base accuracy >75%

(b) CLIP ViT-B/32, MAG2 vs FOLD

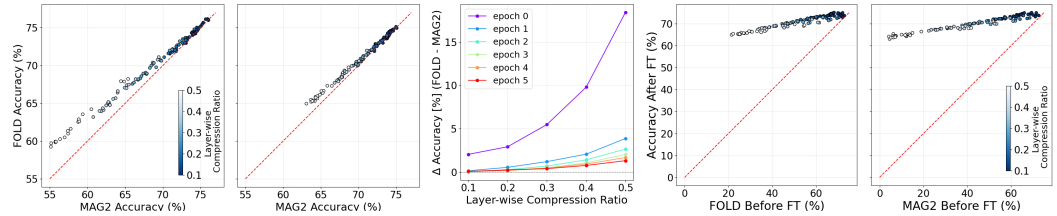
Figure 11: **FOLD versus MAG2 on ViTs after LayerNorm-only fine-tuning** for ViT-B/32 on CIFAR-10 and CLIP ViT-B/32 on ImageNet-1K. In the scatter plots, points are checkpoints, color encodes layer-wise compression. Bar plots depict the accuracy gain $\Delta = \mathrm{Acc}(\texttt{FOLD}) - \mathrm{Acc}(\texttt{MAG1})$, which remains positive and typically grows with compression, indicating that even under lightweight LayerNorm adaptation FOLD retains a consistent advantage over pruning. The figure follows the same setup as Fig. 3 in the main paper, but for MAG2.



(a) MAG2 vs FOLD, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

(b) MAG2 vs FOLD accuracy gap after ft.

(c) Before and after fine-tuning for 5 epochs: FOLD (**left**) and MAG2 (**right**).



(d) MAG2 vs FOLD, fine-tuning for 1 (**left**) and 5 (**right**) epochs.

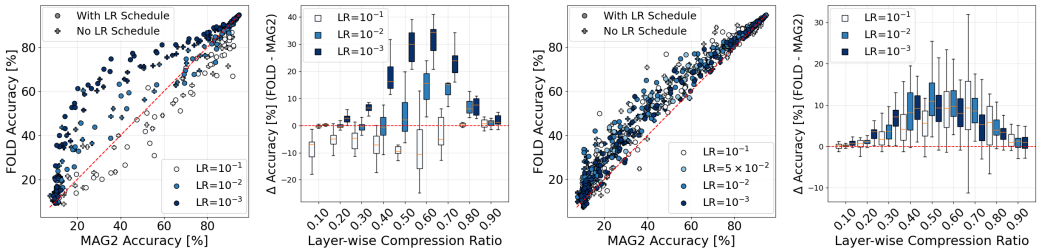
(e) MAG2 vs FOLD accuracy gap after ft.

(f) Before and after fine-tuning for 5 epochs: FOLD (**left**) and MAG2 (**right**).

Figure 12: **FOLD outperforms MAG2 after full fine-tuning for 1–5 epochs on ViT-B/32 and CLIP ViT-B/32.** Results for ViT-B/32 on CIFAR-10 (**top**) and CLIP ViT-B/32 on ImageNet-1K (**bottom**). **(a,d)** accuracy of MAG2 vs. FOLD after 1 and 5 epochs of fine-tuning. **(b,e)** accuracy gap Δ over epochs, remaining positive. **(c,f)** accuracy trajectories from post-compression through 5 epochs, showing faster recovery and higher final accuracy for FOLD. The figure complements Fig. 4 in the main paper, but for MAG2.

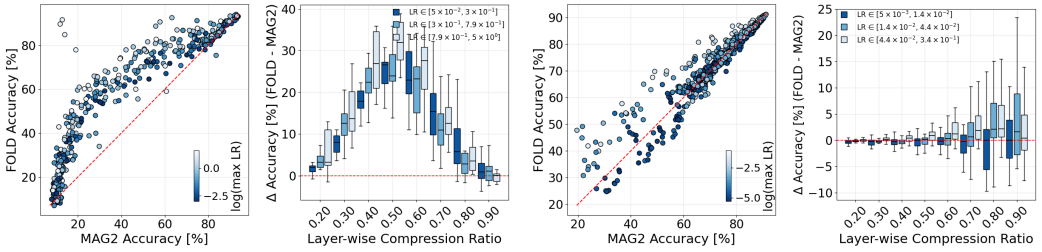
F Related Work

Model compression reduces inference cost and memory footprint, which is critical for deploying deep neural networks in resource-constrained environments. While techniques such as quantization (Darvish Rouhani et al., 2020; Qian Zhang et al., 2022) and knowledge distillation transfer knowledge or reduce precision, we focus on *structured compression* methods that optimize the model architecture post-training without using data, *i.e.*, are *calibration-free*. Among these, sparsity-based pruning is the most widely used: magnitude-based sparsity (Han et al., 2015; Lu et al., 2023; Ding et al., 2024; Bambhaniya et al., 2024) removes weights or channels based on their absolute values, often followed by fine-tuning to recover accuracy (Kurtic et al., 2022; Sanh et al., 2020). Structured patterns such as N:M sparsity (Yao et al., 2019; Kang, 2020) and calibration-based one-shot methods like SparseGPT (Frantar & Alistarh, 2023) or Wanda (Sun et al., 2024) further improve efficiency,



(a) ResNet18, Adam, **no** L1 reg., **no** weight decay

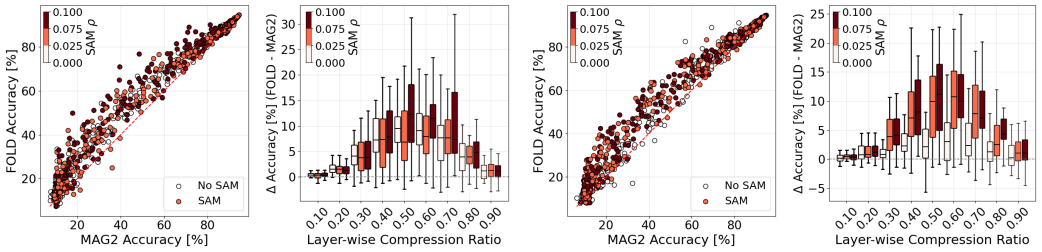
(b) ResNet18, SGD, **no** L1 reg., **no** weight decay



(c) PreActResNet18

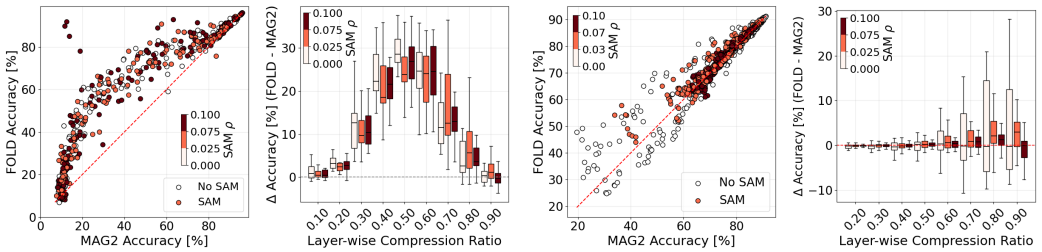
(d) ViT-B/32

Figure 13: **Learning rate modulates folding's edge.** Post-compression accuracy of MAG2 and FOLD across learning rates: ResNet18 with Adam **(a)** and SGD **(b)**, PreActResNet18 **(c)**, and ViT-B/32 **(d)**. FOLD typically leads at moderate–low rates; the gap shrinks or reverses at very high rates, and closes again at extremely small rates. The same setup as in Fig. 5 in the main paper, but for MAG2.



(a) ResNet18, SGD, MAG2 vs FOLD, **without** L1 reg.

(b) ResNet18, SGD, MAG2 vs FOLD, **with** L1 reg.

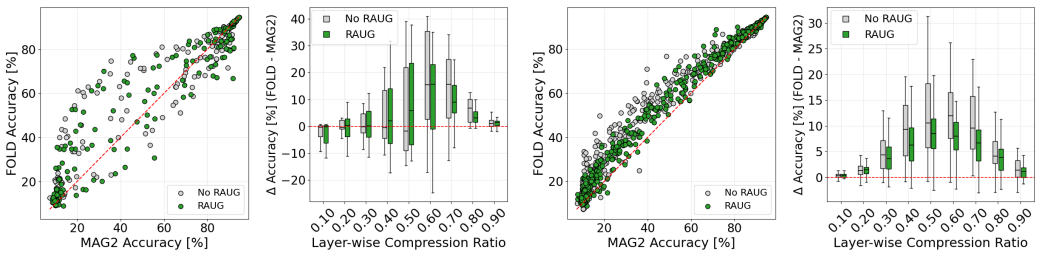


(c) PreActResNet, MAG2 vs FOLD, **no** L1 regularization

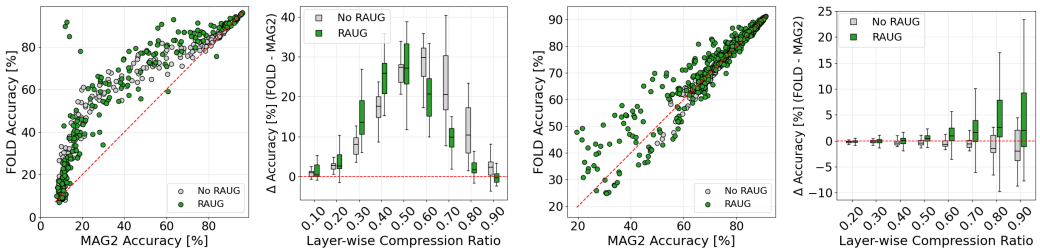
(d) ViT-B/32, MAG2 vs FOLD, **no** L1 regularization

Figure 14: **SAM can boost model compression.** Post-compression accuracy under training with / without SAM. **(a)** ResNet18 (Adam), no L1. **(b)** ResNet18 (Adam), L1= 10^{-5}. **(c)** PreActResNet18 (SGD), no L1. **(d)** ViT-B/32, no L1. The figure extends the results in Fig. 7 to MAG2.

although fine-tuning remains beneficial (Sun et al., 2024; Lu et al., 2024; Syed et al., 2023). Recently, Wang et al. (2025) introduced *model folding*, which clusters and merges similar weights across layers to yield dense low-rank representations. Unlike pruning, folding preserves structural couplings and achieves competitive compression without requiring data or retraining. Our work provides

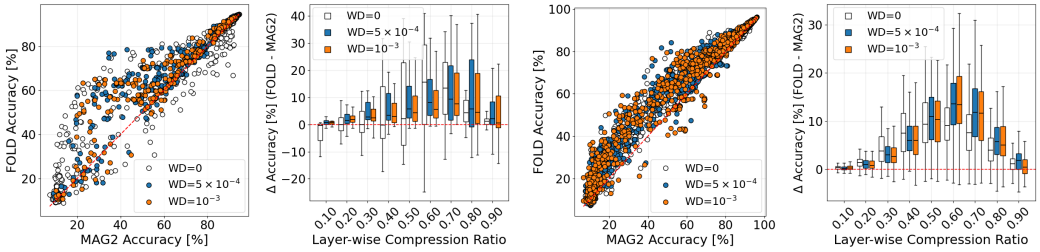


(a) Adam, MAG2 vs FOLD (b) SGD, MAG2 vs FOLD



(c) PreActResNet, MAG2 vs FOLD (d) ViT-B/32, MAG2 vs FOLD

Figure 15: **Random augmentations narrow the folding–pruning gap.** Post-compression accuracy on ResNet18 (CIFAR-10) trained without vs. with random augmentations: **(a)** Adam, **(b)** SGD, **(c)** PreActResNet, **(d)** ViT-B/32. The figure extends Fig. 8 to MAG2.



(a) Adam, MAG2 vs FOLD (b) SGD, MAG2 vs FOLD

Figure 16: **ResNet18: Weight Decay.** Test accuracy of ResNet18 checkpoints trained with varying weight decay values. Weight decay does not diminish the advantage of FOLD compared to MAG2, especially for SGD-trained models.

theoretical insights into this effect, linking folding to curvature regularization and geometry-aware approximations.