Curvature Subspace Compression for Transformer MLPs: Reducing Memorization while Preserving Generalization

Jeffrey Olmo

January 15, 2026

Abstract

We study a practical post-hoc compression method for transformer models that replaces each MLP linear layer with a three-component reparameterization in a curvature-aligned basis. Our method is inspired by recent work analyzing memorization through the spectrum of loss curvature [6]. Concretely, we (i) estimate a K-FAC curvature basis per MLP projection, (ii) express each MLP weight matrix in that basis and drop coefficients corresponding to low-curvature components, and (iii) *physically remove* the resulting zeroed parameters by implementing the edited map as three matrices (input projection, reduced weight, output projection).

Our evaluation focuses on two questions: (i) can compression yield better *validation perplexity at a fixed parameter budget* (i.e., outperform an iso-parameter uncompressed baseline), and (ii) does compression reduce memorization as measured by synthetic canary exposure while leaving generalization largely intact? Across a sweep of model widths and compression levels, we find regimes where compressed models trace a more favorable validation-perplexity vs. parameter trajectory than uncompressed baselines, and we observe large reductions in canary perplexity with comparatively modest changes to validation perplexity.
Code: https://github.com/JeffreyOlmo/Curvature_Subspace_Compression

1 Introduction

Transformer scaling reliably improves downstream performance but incurs large memory and compute costs. Compression is therefore central to efficient deployment and to controlled experiments that separate *capability* from *memorization*. Classic approaches include pruning [1, 2], distillation [3], and low-rank structure [4]. These methods often target parameter count or compute directly, but do not explicitly use the geometry of the loss landscape.

Recent work emphasizes curvature as a lens on model behavior, including a key population-level inversion: while individually memorized examples can be locally sharp, averaging curvature over data emphasizes directions that are consistently important (shared mechanisms), while idiosyncratic memorized directions tend to cancel and appear in flatter parts of the spectrum [6]. Inspired by this viewpoint, we compress transformer MLP projections by editing weights in a K-FAC curvature basis [5], ablating low-curvature components, and then removing the corresponding parameters by reparameterizing the layer as a product of three matrices.

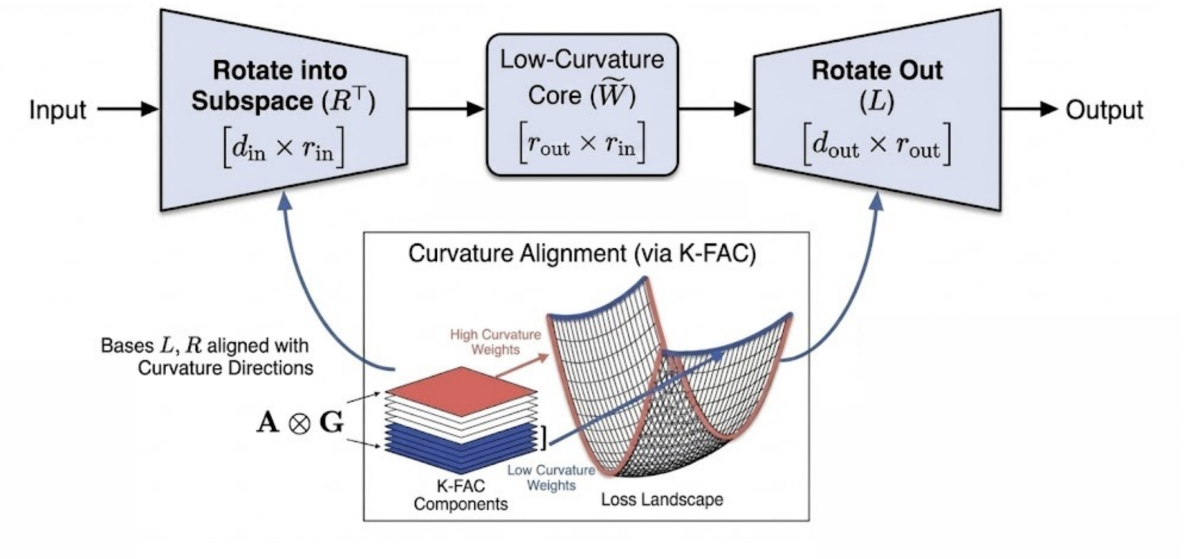

Figure 1: Curvature Subspace Compression

2 Method: Curvature Subspace Compression of MLP Blocks

2.1 Three-component MLP replacement (no learnable core)

Consider a linear layer weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Our procedure produces an *edited* weight W_{edit} by removing low-curvature components in a K-FAC eigenbasis (details below). We then implement W_{edit} as a product of three matrices:

$$W_{\text{edit}} = L\,\widetilde{W}\,R^\top,$$

where:

- $R \in \mathbb{R}^{d_{\text{in}} \times r_{\text{in}}}$ is a fixed input-side basis (a subset of activation eigenvectors),
- $L \in \mathbb{R}^{d_{\text{out}} \times r_{\text{out}}}$ is a fixed output-side basis (a subset of gradient eigenvectors),
- $\widetilde{W} \in \mathbb{R}^{r_{\text{out}} \times r_{\text{in}}}$ is the *deterministic coefficient matrix* obtained by expressing the original W in this basis and zeroing (then removing) coefficients corresponding to ablated curvature components.

There is **no additional learnable matrix** introduced by compression: $\widetilde{W}$ is fully determined by the baseline W and the chosen curvature mask. Any parameter savings come from (i) deleting masked coefficients rather than storing them as zeros, and (ii) storing only the selected columns of L and R needed to reconstruct W_{edit}.

2.2 Choosing the basis with K-FAC curvature

K-FAC approximates the Fisher/Hessian-like curvature for a layer as a Kronecker product [5]

$$\mathcal{F} \approx A \otimes G,$$

where A is the input activation covariance and G is the output gradient covariance.

We compute eigendecompositions:

$$A = U_A \Lambda_A U_A^\top, \qquad G = U_G \Lambda_G U_G^\top,$$

and use the resulting eigenvectors to form a curvature-aligned basis for W via rank-one components (outer products) [6]. In this construction, the curvature spectrum is ordered so that, at the population level, high-curvature directions correspond to structure shared across many datapoints, while flatter directions correspond to directions important for relatively few datapoints (including recitation-related directions) [6].

2.3 Curvature-mass–based masking and weight editing

Rather than selecting eigenvectors of A and G independently, we follow Merullo et al. [6] and select *pairs* of eigenvectors according to a joint curvature-mass criterion. Each pair (i, j) (gradient eigenvector $U_G[:, i]$ and activation eigenvector $U_A[:, j]$) has an associated curvature mass proportional to the product of eigenvalues:

$$\Pi_{ij} \propto \Lambda_G[i]\, \Lambda_A[j].$$

We then choose a subset of pairs S by sorting (i, j) in descending Π_{ij} and including pairs until a target fraction ρ of total curvature mass is retained [6]. Equivalently, this defines a binary mask M over coefficients in the curvature basis.

Operationally, we express W in the eigenbases and apply the mask:

$$\widetilde{W}_{\text{full}} = U_G^\top W U_A, \qquad \widetilde{W}_{\text{mask}} = \widetilde{W}_{\text{full}} \odot M,$$

and reconstruct the edited weight:

$$W_{\text{edit}} = U_G\, \widetilde{W}_{\text{mask}}\, U_A^\top.$$

This is exactly the curvature-basis "drop low-curvature coefficients" edit of Merullo et al. [6]. Our only additional step is to *remove* the masked-out parameters rather than keeping them as zeros.

2.4 Removing zeroed parameters via a three-matrix implementation

The masked coefficient matrix $\widetilde{W}_{\text{mask}}$ is sparse (or structured sparse) in the curvature basis. To obtain a compact implementation, we collect the sets of indices that survive the mask:

$$I = \{i : \exists j \text{ s.t. } M_{ij} = 1\}, \qquad J = \{j : \exists i \text{ s.t. } M_{ij} = 1\}.$$

Define

$$L = U_G[:, I], \qquad R = U_A[:, J], \qquad \widetilde{W} = \left(\widetilde{W}_{\text{mask}}\right)_{I,J}.$$

Then we can implement the edited layer as

$$W_{\text{edit}} = L\, \widetilde{W}\, R^\top,$$

with the understanding that any residual zeros inside $\widetilde{W}$ can also be removed in storage/implementation (e.g., sparse formats), since we are not training these coefficients further.

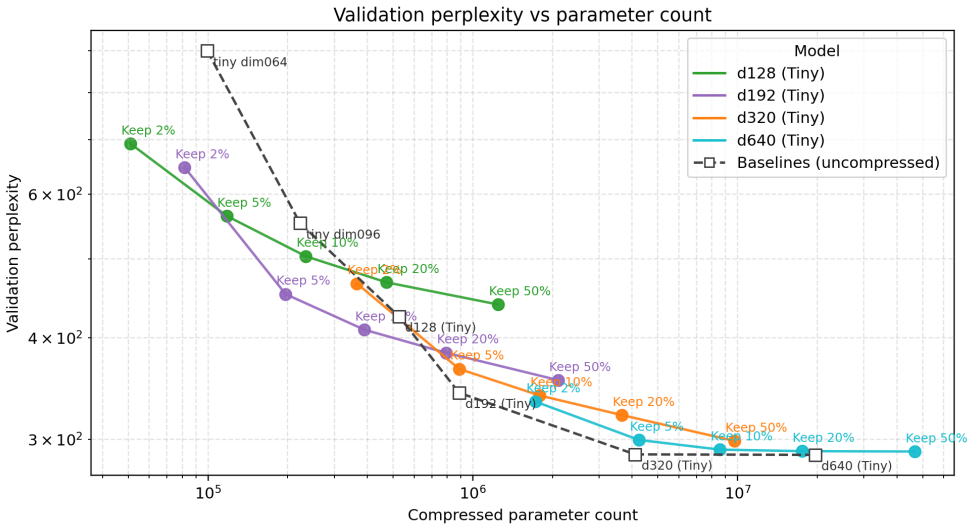

Figure 2: Validation perplexity vs. (compressed) parameter count. Curvature subspace compression can trace a more favorable trajectory than uncompressed baselines, yielding iso-parameter improvements in certain regimes.

3 Experimental Setup

Models and training. We sweep transformer widths (holding depth fixed per sweep) and apply the MLP replacement across MLP blocks. Models are trained to baselines, then compressed at several keep fractions using the same baseline checkpoint.

Metrics. We report:

- **Validation perplexity** (generalization).
- **Canary perplexity** (memorization), evaluated on synthetic canaries inserted into training data; lower canary perplexity indicates stronger memorization.

Key comparison axes.

- **Validation perplexity vs. parameter count:** whether compression can beat uncompressed baselines at the same parameter budget.
- **Validation vs. canary perplexity across compression levels:** whether memorization drops substantially faster than validation quality.

4 Results

4.1 Validation perplexity vs. parameter count (iso-parameter wins)

Figure 2 plots validation perplexity against *compressed parameter count*. The uncompressed baselines form a separate trajectory, while compressed variants trace their own curves. In some regimes,

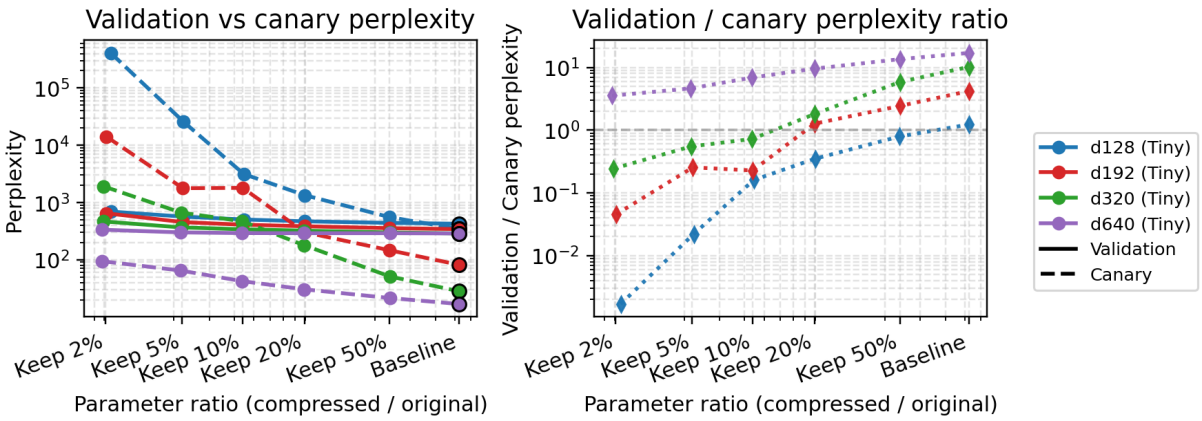

Figure 3: Additional metrics across compression levels: validation vs. canary perplexity (left) and the validation/canary ratio (right). Canary perplexity moves far more than validation perplexity, indicating a strong reduction in memorization pressure with limited generalization impact.

compressed models achieve *lower validation perplexity at the same parameter scale* than the uncompressed baseline curve, indicating that curvature-aligned MLP compression is not merely "shrinking" the model but can move it to a better operating point in the size–performance plane.

4.2 Memorization decreases sharply while validation remains comparatively stable

Figure 3 shows validation and canary perplexity across compression levels, alongside a ratio panel summarizing their divergence. Across models, canary perplexity increases dramatically under compression (i.e., memorization weakens), while validation perplexity changes more modestly. This separation is desirable on its own grounds: reduced memorization improves privacy and robustness and reduces brittle reliance on rare training strings, while preserving generalization.

5 Discussion

Why might curvature-aligned editing and compression help? Following Merullo et al. [6], dataset-averaged curvature emphasizes directions corresponding to shared mechanisms (which add coherently across datapoints), while idiosyncratic directions associated with relatively few datapoints contribute to a flatter background. Thus, ablating low-curvature components can disproportionately suppress recitation-related behavior while preserving broadly useful computation. Our additional reparameterization step turns this edit into a true compression scheme by deleting the ablated parameters and implementing the edited map with three matrices.

6 Limitations and Future Work

- **Curvature approximation:** K-FAC is an approximation and may miss important cross-layer structure [5].
- **Scope of compression:** we focus on MLP blocks; extending curvature-aligned compression to attention projections and embeddings is natural.

- **Memorization measurement:** synthetic canaries capture a useful facet of memorization, but do not fully characterize privacy leakage.
- **Training/compression interaction:** this report evaluates post-hoc compression; joint training with curvature-aligned constraints is a promising direction.

7 Conclusion

Curvature subspace compression provides a simple, modular replacement for transformer MLPs using a three-matrix implementation in a K-FAC curvature basis. We edit each MLP by removing low-curvature components in the curvature spectrum as in Merullo et al. [6], and then convert that edit into real parameter savings by removing the masked-out coefficients and bases. Empirically, we observe (i) regimes where compressed models beat uncompressed baselines at the same parameter budget, and (ii) large reductions in memorization as measured by canary perplexity with comparatively modest changes to validation perplexity.

References

[1] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2018. URL https://arxiv.org/abs/1803.03635.

[2] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural networks. In *NeurIPS*, 2015.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[5] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2015. URL https://arxiv.org/abs/1503.05671.

[6] Jack Merullo, Srihita Vatsavaya, Lucius Bushnaq, and Owen Lewis. From memorization to reasoning in the spectrum of loss curvature, 2025. URL https://arxiv.org/abs/2510.24256.