FROM MEMORIZATION TO REASONING IN THE SPECTRUM OF LOSS CURVATURE

Anonymous authors
Paper under double-blind review

ABSTRACT

We characterize how memorization is represented in transformer models. The eigenbasis of the Fisher Information Matrix for MLP weight matrices encodes shared structure used across many data points at the top of the spectrum and memorized examples at the bottom, which we can disentangle in the weight space in both language models (LMs) and vision transformers (ViTs). We connect this finding to prior theoretical and empirical work on the curvature of the loss landscape for individual memorized datapoints, and use it to propose a weight editing procedure that suppresses far more recitation of untargeted memorized data more effectively than a recent unlearning method (BalancedSubnet; Sakarvadia et al. (2025)), while maintaining lower perplexity. Since the basis of curvature has a natural interpretation for shared structure in model weights, we analyze the editing procedure extensively on its effect on downstream tasks in LMs, and find that fact retrieval and arithmetic are specifically and consistently negatively affected, even though open-book and general logical reasoning is conserved. We posit these tasks rely heavily on specialized directions in weight space rather than general purpose mechanisms, regardless of whether those individual datapoints are memorized. Our work enhances the understanding of memorization in neural networks with practical applications towards removing it, and provides evidence for idiosyncratic, heuristic-like structures that are used to solve tasks like math and fact retrieval.

1 INTRODUCTION

To what degree do models generate genuinely new knowledge, as opposed to simply reassembling snippets of data memorized from their training sets? Much discussion about the current utility and future prospects of large neural networks has centered on this question. On the one hand, a growing trophy case of model accomplishments on novel tasks near the frontier of human capabilities argues strongly against memorization strictly construed as an explanation of the full range of model behavior. But on the other hand, recent papers have argued convincingly that models do in fact memorize large volumes of their training data verbatim, and a surprisingly large fraction of naturalistic interactions with language-model chatbots contain significant verbatim recitations (Aerni et al., 2024; Carlini et al., 2022; Stoehr et al., 2024), behaviors which have significant implications for copyright and data privacy (Karamolegkou et al., 2023; Carlini et al., 2019; Shokri et al., 2017).

Models thus seem to have a significant and frequently used capacity for both memorization and generalization. The question is not whether models generalize or recite, but rather how these capabilities are represented, how they interact and trade off, and how they might be modulated. These are the questions we seek to address in this work. We build on existing work that characterizes memorization in terms of the curvature of the loss landscape as a function of a model's weights Foret et al. (2021); Hochreiter & Schmidhuber (1997); LeCun et al. (1989); Hassibi et al. (1993); Keskar et al. (2017); Garg et al. (2024); Ravikumar et al. (2024); Jeon et al. (2024); Kim et al. (2023). This prior work argues theoretically and empirically that the loss landscape has highly curved directions in the neighborhood of memorized points, while generalization corresponds to flatter directions. We exploit this insight while extending it in several ways.

We study models' behavior in aggregate, rather than for individual examples. Are there model structures that account for memorization and generalization across large swaths of training data? We find that there are: in both language and vision models, the eigenbasis of the approximated

Hessian of weight matrices uncovers distinct disentanglement of memorization and generalization, in a way that extends across a range of subdistributions of memorized data. In extending from studying per-example to bulk memorization, we propose a novel inversion of the previous interpretation of loss curvature: while individual memorized points are associated with high curvature, the direction of curvature varies across examples, meaning that, averaged across multiple examples, memorization directions are actually flatter than generalizing directions, which maintain a consistent moderate curvature across points.

We propose an effective recitation-reduction technique based on ablating memorized directions in weight space. We compare our results to a recent supervised memorization removal technique (BalancedSubnet (BSN); Sakarvadia et al. (2025)) and find that our method matches the suppression of the targeted forget set of BSN, and is similarly robust to stress tests (Huang et al., 2024), while removing far more unseen memorized data and achieving lower perplexity 6.3.

We go beyond pure memorization and generalization and find curvature signatures of intermediate behaviors like fact retrieval and arithmetic. Many classical analyses of memorization and generalization have studied classification models, where the distinction between the two is clear. In modern language models, though, there is a much richer spectrum of behaviors between the poles of pure memorization (verbatim recitation of long passages) and pure generalization (de novo reasoning). For example, facts like "Paris is the capital of France" are memorized in the sense that they are specific pieces of information that the model knows, but are general in the sense that they are not tied to specific syntactic instantiations seen in the training data. Similarly, arithmetic and logical reasoning test a model's ability to generate novel inferences, but the inference rules and base axioms may be remembered from training. Going beyond prior work, we extend the loss curvature analysis to cover these reasoning types, situating them on a continuum between memorization and generalization. We find that, besides memorization, arithmetic and factual recall exhibit weight activation with low-curvature weight directions and are sensitive to their removal. On the other hand, logical reasoning, which does not necessarily require precise recall or calculation is robust to removing flat directions, which in some cases even improving performance.

For all of our analyses, we measure curvature with the Kronecker-Factored Approximate Curvature (K-FAC) approximation to the model's Hessian, a computational technique that makes curvature analyses tractable at scale. K-FAC has been widely used for Hessian approximation in other settings, but to our knowledge, our use of it to study memorization and generalization via curvature is novel.

2 Related Work

Memorization, especially as a special case of overfitting, is a widely studied topic in both modern and classical machine learning. In the modern era of extremely large overparameterized models, there is particular interest in quantifying the ability and tendency of models to use their huge capacity to memorize training data. Recent work has shown that models are indeed able to store large amounts of data exactly (Morris et al., 2025), and that this data can be elicited verbatim in both naturalistic (Aerni et al., 2024) and adversarial (Carlini et al., 2022; Karamolegkou et al., 2023) regimes.

A closely related question is whether memories can be localized in model weights (Maini et al., 2023; Chang et al., 2024; Stoehr et al., 2024; Huang et al., 2024). Aligning with previous work (Hase et al., 2023; Karamolegkou et al., 2023), our work suggests that memorization is hard to pinpoint (and likely highly distributed), but we do find that distinctly loss-curved directions related to recitation of memorized data can be localized to some (early/late) layers. Similar localization work has studied the storage and retrieval of facts (Geva et al., 2021; Gur-Arieh et al., 2025; Meng et al., 2022; Dai et al., 2022; Rajamanoharan et al., 2023; Merullo et al., 2024; Menta et al., 2025), connecting to our analysis of factual recall in Section 7.

Like the present study, previous work has used techniques like SVD to prune directions in weight space to compress models, expose low-rank structure, and understand memorization (Zhao et al., 2024; Jaiswal et al., 2025; Sharma et al., 2023). Relatedly, spectral dynamics has also been used explore memorization and generalization (Yunis et al., 2024).

Finally, a range of theoretical and empirical work has studied the connection between memorization and loss curvature, connecting high-curvature directions with memorized examples (Foret et al., 2021;

Hochreiter & Schmidhuber, 1997; LeCun et al., 1989; Hassibi et al., 1993; Keskar et al., 2017; Garg et al., 2024; Ravikumar et al., 2024; Jeon et al., 2024; Kim et al., 2023).

3 Background on Loss Curvature and K-FAC

Following Martens & Grosse (2015); Foret et al. (2021), we study memorization and generalization through the lens of loss curvature, specifically as a function of the model's weights (Keskar et al., 2017). Mathematically, the curvature of the loss landscape is captured by the Hessian $\mathbf{H} = \nabla^2_\theta L(\theta)$, where L is the loss function and θ is the vector of flattened model weights. Practically, though, $\mathbf{H}$ is not tractably computable for any but the smallest models, as its size is quadratic in the number of model weights. Prior work bypasses explicitly computing $\mathbf{H}$ by approximating its top eigenvalues and/or trace Ghorbani et al. (2019); Foret et al. (2021); Ravikumar et al. (2024); Garg et al. (2024)[1], or by making other approximations Hochreiter & Schmidhuber (1997); Keskar et al. (2017). For our analyses, though, we need a more complete picture of the whole spectrum of $\mathbf{H}$, and to get this picture, we turn to the Kronecker-Factored Approximate Curvature (K-FAC) Martens & Grosse (2015). Originally introduced as an efficient natural-gradient method for optimization, K-FAC approximates the Fisher Information Matrix (FIM) and provides a structured approximation to the loss curvature without forming the full Hessian.

For a model trained with softmax cross-entropy loss, the relationship of the FIM $\mathbf{F}$ to the curvature of parameters is given by:

$$\mathbf{F} = \mathbb{E}_D[\nabla_\theta \log p_\theta(y \mid x, \theta) \nabla_\theta \log p_\theta(y \mid x, \theta)^T] = \mathbb{E}_D[\nabla^2_\theta(-\log p_\theta(y|x))]$$

Here, D is a dataset consisting of input-label pairs (x, y), and $p_\theta(y \mid x)$ is the model's predicted label distribution for input x. For an individual matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ with incoming activations a and backpropagated gradients g, K-FAC gives an easily computable approximation to $\mathbf{W}$'s block of $\mathbf{F}$:

$$\mathbf{F}_W \approx \mathbf{G} \otimes \mathbf{A} = \mathbb{E}[gg^T] \otimes \mathbb{E}[aa^T], \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ and $\mathbf{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$. In words, this is the Kronecker product of the (uncentered) second-moment matrices of the activations going into the layer and the gradients coming out. When computing the loss to backpropagate into $\mathbf{G}$, we sample $\hat{y}$ from the model's predicted label distribution, rather than taking the ground truth y. Not only is this important for the correctness of the FIM FIM(Martens & Grosse, 2015), but it also means we can use this method without any labeled data.

Instance- vs. population-level curvature. Per-example analyses often find that memorized points are locally sharp. Our use of the Fisher/K-FAC differs by averaging curvature across data, which emphasizes directions that are *consistently* important. Idiosyncratic sharp directions associated with specific examples point in different directions and largely cancel in the average, contributing to a low-curvature background. Directions that implement shared mechanisms (used by many inputs) add coherently and remain high-curvature on average. This explains why retaining high curvature mass preserves general abilities, while removing low-curvature components preferentially suppresses recitation.

Decomposing K-FAC Throughout this work, we will decompose the FIM of a weight matrix $\mathbf{W}$ into distinct components (directions) to analyze their role in reciting memorized data. We can compute the eigendecomposition of the FIM by individually eigendecomposing $\mathbf{A}$ and $\mathbf{G}$ (see Appendix C). We refer to the eigendecomposition of K-FAC as the curvature basis, since the eigenvalues are sorted in terms of most to least loss curvature. A single eigenvector of K-FAC is the outer product of an activations eigenvector and gradient eigenvector, and thus can be considered a weight component of $\mathbf{W}$. Therefore, when we describe the 'activation' of data with a rank-one component $\mathbf{C}$, we are describing the matrix vector product $\mathbf{C}x$. The magnitude of this product would be the norm of the resulting vector.

[1] Ravikumar et al. (2024); Garg et al. (2024) measure the trace w.r.t. the *inputs* rather than parameters, but this distinction isn't important for this point.

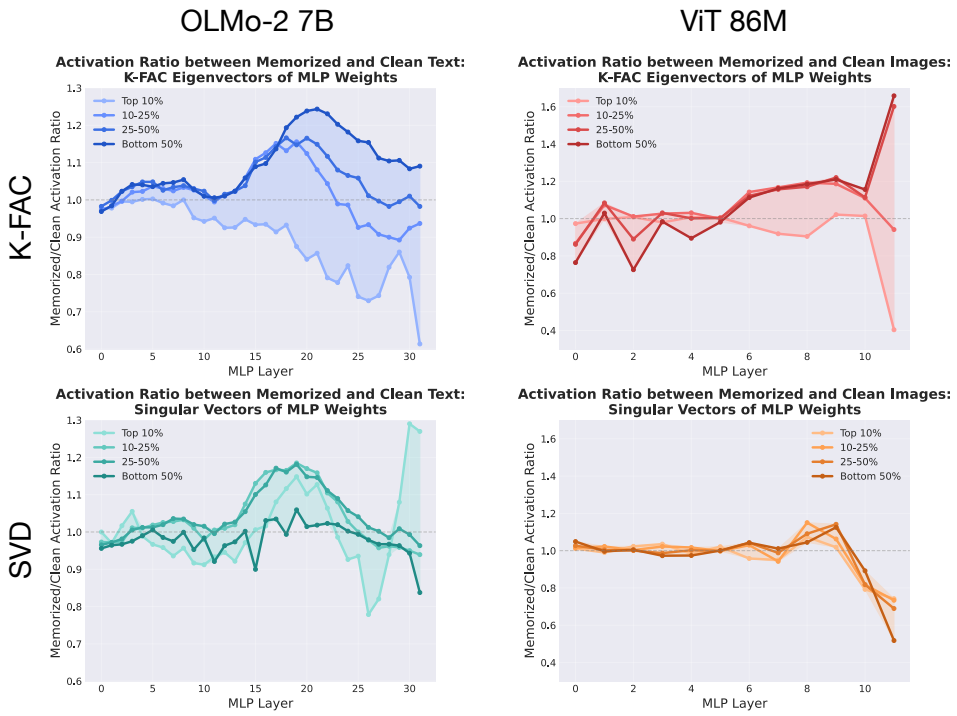

Figure 1: Large disentanglement in the weight space between memorized and non-memorized (clean) data, especially when decomposed into weight components with K-FAC. The activation ratios show **selectivity**, where some parts of the spectrum activate more strongly for memorized data than others (or vice versa). The pattern is apparent in LMs and ViTs, though less so for the SVD baseline.

4 Models

Vision Transformers (ViTs) Memorization in image classification models has been well studied, and there are simple recipes for producing models that memorize specific images. We train a family of 86M parameter ViT-Base models (Dosovitskiy et al., 2020) with 16x16 image patches at image resolution 224x224. We follow Dosovitskiy et al. (2020) training recipe on the ILSVRC 2012 ImageNet dataset (Russakovsky et al., 2015). In order to control memorization, we train ViT variants where a subset of training images have randomly assigned 'noised' labels. The only way for a model to reduce the loss on these images is to memorize these input-label pairs exactly. This is a standard setup for evaluating memorization in image classifiers (Zhang et al., 2017). Our default for evaluation is to train with 10% noised labels for 300 epochs. Our model trained with the noised labels achieves a top-1 accuracy on the validation set of 68.7%. When training with no noise, our model achieves 77.2% top-1 accuracy.

Language Models (LMs) We use the OLMo-2 family of models (OLMo et al., 2024), because they have openly accessible pretraining data and high performance on language modeling tasks. We report results for the 7B model. Previous work on evaluating memorization in LMs (Carlini et al., 2019; Huang et al., 2024; Shokri et al., 2017; Carlini et al., 2022) generally sampled sequences from a model's pretraining data, split each sequence into a prefix P and suffix S, and evaluated whether the model produced S under greedy decoding with prompt P. We adopt this same methodology, and use prefixes with length $|P| = 64$ tokens and suffixes with length $|S| = 48$ tokens.

5 Disentangling Weights Involved in Memorization

This section will show that K-FAC is indeed a particularly good candidate to disentangle weights involved in memorized recitation. In simple terms, our procedure is to measure the interaction between memorized and non-memorized (clean) data with different K-FAC components of the weights. Our hypothesis is that if the curvature is a good measure of memorization, then the eigenvectors in

different parts of the spectrum of the curvature basis (i.e., the top 10% of eigenvectors, bottom 50%, etc.) will activate differently from each other on memorized or clean data. The way we measure this is through activation ratios: for some hidden activation x_{mem} stemming from a memorized input, we compare the ratio of its activation with a weight component $\mathbf{C}$ to the activation with a clean input x_{clean}. If one weight component has a high $||\mathbf{C}x_{mem}||/||\mathbf{C}x_{clean}||$ ratio and another component has a very low ratio, then we know this weight matrix distinguishes the two types of data. To summarize the experiment: **for a given weight matrix, we would like to know if it has some components that specialize for memorization, and some that specialize for other data.**

Baseline: SVD A reasonable hypothesis is that we can disentangle memorization in the basis of singular vectors of a weight matrix, which decomposes into the components of most to least 'importance' for reconstructing the matrix. The top singular vectors may correspond to more general directions in weight space, and the lower ones towards directions used for reciting memorization (see Yunis et al. (2024)). If we compute the product between activations and weights as $\mathbf{W}x$, and the SVD is $\mathbf{USV^T}x$, then the right singular vectors in $\mathbf{V^T}x$ give us the magnitude with which those singular vectors read from.

K-FAC Eigenvectors Our experimental condition is to test whether the curvature basis is useful way to disentangle weight space into components involved in recitation or generalizing behaviors. An eigenvector of the FIM has the same size as its source weight matrix $\mathbf{W}$. We compute the L2 norm $||y|| = \mathbf{Z}x$ for hidden state x and eigenvector $\mathbf{Z}$. See Appendix B for details.

Results Figure 1 shows our results. In both LMs and ViTs, K-FAC shows a more salient divergence between different parts of the eigenspectrum on memorized and clean data. For example, at the layer 22 MLP input in OLMo-7B, the bottom 50% of eigenvectors has a 23.1% higher activation on memorized data than clean data on average, and the top 10% of eigenvectors has a 26% higher activation on clean data than memorized data; SVD has a strong separation on the last layer (top 10% having 26% higher activation on memorized data), but no where else. In the ViT model we train, the top 10% eigenvectors have over a 2x activation strength on clean over memorized data at the last layer; the SVD for this model shows no such pattern.

These results support our hypothesis that the curvature basis allows us to disentangle weights involved in memorization recitation. As we will show in Section 7, we can use the amount of divergence between the top and bottom parts of the eigenspectrum to predict model behaviors on various tasks. In the following section, we will that removing weight components at the bottom of the K-FAC spectrum suppresses memorized data while retaining strong performance.

6 Editing Model Weights to Suppress Memorization

A natural followup to finding distinct patterns of activations for (non-)memorized data across weight components in the curvature basis is whether we can use this discovery to prevent the recitation of memorized data while retaining general capabilities. We propose a novel editing method which projects an MLP weight matrix $\mathbf{W}$ into a subspace defined by its Hessian. As discussed in Section 3, the eigendecomposition of this Hessian sorts components of $\mathbf{W}$ into directions of highest to lowest curvature in the loss landscape across a dataset. As we have discussed, in the aggregate across a dataset, the top eigenvectors correspond to generalizing directions; a claim that we will directly test here. Therefore, we propose keeping only the top $k\%$ of eigenvectors as a way to keep this shared structure while removing noisy or generally unimportant weight directions. In words, we define a matrix projection of an MLP weight matrix $\mathbf{W}$ that prevents communication through directions of low curvature in the eigenvectors of K-FAC.

Our method decomposes weight matrices using eigenbases derived from activation and gradient covariance matrices. Rather than truncating the eigenbases directly, we select specific pairs of eigenvectors whose joint contribution to curvature is highest, preserving a targeted fraction of the total curvature mass. Concretely, we start with K-FAC factor matrices $\mathbf{G} \in \mathbb{R}^{p\times p}$ (gradient covariance) and $\mathbf{A} \in \mathbb{R}^{q\times q}$ (activation covariance). These are decomposed into their eigenspaces as:

$$\mathbf{G} = \mathbf{U_G}\,\mathrm{diag}(\lambda)\,\mathbf{U_G}^\top, \quad \mathbf{A} = \mathbf{U_A}\,\mathrm{diag}(\mu)\,\mathbf{U_A}^\top, \quad \text{with} \quad \lambda_0 \geq \lambda_1 \geq \cdots \geq 0, \quad \mu_0 \geq \mu_1 \geq \cdots \geq 0.$$

Given a weight matrix $\mathbf{W} \in \mathbb{R}^{p \times q}$, we first express it in terms of these eigenbases as:

$$\mathbf{C} \;=\; \mathbf{U_G}^{\top}\,\mathbf{W}\,\mathbf{U_A}, \quad \text{where each coefficient} \quad C_{ij} = u_i^{\top}\mathbf{W}v_j.$$

To guide our compression, for each eigenvector pair (i, j) we define a measure of curvature mass, $\Pi_{ij} := \lambda_i \mu_j$. The total curvature mass, summing over all pairs, is then given by:

$$M_{\text{tot}} := \sum_{i,j} \Pi_{ij} = \Big(\sum_i \lambda_i\Big)\Big(\sum_j \mu_j\Big).$$

Our compression strategy then selects a subset S of eigenvector pairs, prioritizing those with the highest curvature mass. Formally, given a threshold parameter $\rho \in (0, 1]$, we construct S by including pairs in descending order of Π_{ij} until the cumulative curvature mass of selected pairs meets or exceeds the fraction ρ of the total mass: $\sum_{(i,j)\in S} \Pi_{ij} \;\geq\; \rho\, M_{\text{tot}}$.

Once the subset S is determined, we define a binary mask matrix $M \in \{0, 1\}^{p \times q}$, where $M_{ij} = 1$ if $(i, j) \in S$, and 0 otherwise. Finally, we construct the compressed weight matrix by zeroing out coefficients corresponding to pairs not in S:

$$\mathbf{W}_{\text{pairs}} \;=\; \mathbf{U_G}\,(\mathbf{C} \odot \mathbf{M})\,\mathbf{U_A}^{\top} \;=\; \sum_{(i,j)\in S} C_{ij}\, u_i v_j^{\top}.$$

This method selectively preserves those directions in weight space most significant to the model's curvature.

6.1 K-FAC FACTOR COLLECTION

We use K-FAC to approximate the Fisher block of each linear projection as a Kronecker product as shown in Equation 1, where A captures input correlations and G captures correlations of output gradients. We collect these factors for the MLP projections (`gate_proj`, `up_proj`, `down_proj`) by streaming $\sim$20M tokens from Dolmino/OLMo mixtures with sequence length 512 under next-token cross-entropy. In the forward pass we buffer pre-activation inputs x (excluding the last position), and in the backward pass we record the corresponding gradients g. We accumulate $x^{\top}x$ and $g^{\top}g$ in float32 and normalize by the total number of contributing positions to form empirical A and G.

6.2 EXPERIMENTAL SETUP

We construct two exact-match memorization sets under greedy decoding, one drawn from the pretraining corpus with a prefix and suffix of 48 and 64 respectively, and another of memorized historical quotes which we measure as memorized with a suffix of 8. Details are in Appendix D.

We report the metrics described in Section 6.2 separately on the Dolma and Quotes datasets. We primarily rely on strict accuracy to detect exact memorization in the dataset generation. However for evaluation, cases where strict = 0 but loose = 1 highlight sequences differing only slightly—semantically or syntactically—from the memorized target, representing partial memorization we also seek to avoid. Thus, loose accuracy complements strict accuracy by capturing near-verbatim memorization. Additionally, Levenshtein distance provides a continuous, threshold-free metric, allowing us to quantify memorization degradation more precisely.

Metrics We evaluate memorization suppression in ViTs with three metrics: *memory reduction* (drop in top-1 predictions of memorized/noised labels), *ground-tuth recovery* (accuracy of recovering the true label for images trained with noised labels), and *validation accuracy* (post-edit validation accuracy to assess impact on core capabilities).

For LMs, given a prefix–suffix pair (P, S) with $|S| = L$, we prompt with P and greedily generate L tokens to obtain $\hat{S}$. We compute the token-level Levenshtein distance $d(S, \hat{S})$ and report: *Strict Accuracy* $\mathbb{I}\big[d(S, \hat{S}) = 0\big]$; *Loose Accuracy* $\mathbb{I}\big[1 - d(S, \hat{S})/L \geq \tau\big]$ with $\tau = 0.75$; and *Average Normalized Distance* $\frac{1}{N}\sum_{n=1}^{N} \frac{d(S_n, \hat{S}_n)}{L_n}$, where higher values indicate less memorization.

	Dolma Validation			Historical Quotes			Pile10k
Method	Strict (%)	Loose (%)	Avg Lev ↑	Strict (%)	Loose (%)	Avg Lev ↑	Perplexity ↓
Baseline	99.9	100.0	0.002	99.9	100.0	0.001	19.04
BSN	6.0	11.0	0.860	60.0	79.0	0.180	23.59
K-FAC	3.4	8.8	0.704	16.1	23.8	0.625	22.84

Table 1: Comparison of memorization metrics. K-FAC is more performant at suppressing memorization across domains and has lower perplexity than BSN

Method	Train Accuracy (%) ↓	Train GT Accuracy (%) ↑	Validation Accuracy (%) ↑
Baseline	81.6	10.5	67.0
SVD	3.5	58.9	67.8
K-FAC	3.5	66.5	71.7

Table 2: Comparison of edits on ViT-Base. K-FAC edits allow us to remove most memorization, recovers the ground truth label most of the time, and (likely through regularization) improves validation accuracy. The SVD (keeping only 5% of the selected K-FAC layers is also an effective baseline, but does a worse job recovering performance than K-FAC.

Baseline: Balanced Subnet (BSN) We compare to BSN, a recent memorization unlearning method introduced in Sakarvadia et al. (2025). This method trains a binary mask over individual MLP parameters optimized to maximize loss on a forget set (memorized data), while retaining low loss on a retain set (non-memorized, clean data).

Model Settings For K-FAC: The full hyperparameter search details can be found in Appendix F. In LMs we edit layers 23, 24, and 25 at 60% energy retained in the up and gate projections in MLPs. In ViTs, we edit layers 0 and 11 to 75% energy on both up and down MLP projections.

For BSN: The best BSN settings for editing the language model were a loss weight of 0.7, 5 epochs, a sparsity ratio of 0.0015, and a learning rate of 0.3.

6.3 Results

Language Models We compare our proposed K-FAC method against the state-of-the-art BSN baseline in Table 1. To ensure comparability of model coherence, we matched perplexities closely (K-FAC: 22.84, BSN: 23.59), noting that BSN achieved slightly better nDCG@10 (0.97 vs. 0.91). While BSN required explicit training data, K-FAC did not—highlighting an important advantage of our approach. On the Dolma validation set, K-FAC performed better (strict accuracy: 3.4%) compared to BSN (strict accuracy: 6.0%). Most notably, K-FAC substantially outperformed BSN on the truly out-of-distribution historical quotes dataset (strict accuracy: 16.1% vs. BSN's 60.0%). These results demonstrate our curvature-based pruning approach effectively mitigates memorization without requiring supervised training data[2], achieving notably better generalization to unseen memorized content.

Vision Models Table 2 shows the results for editing ViT-Base with 10% training noise in various settings. On a per-layer basis, we see that pruning the earliest and latest layers provides the best results across the board. For both methods, we achieve the best performance when we prune MLPs 0 and 11 simultaneously, driving memorization performance down to 3.5% from over 80%. K-FAC also *increases* the validation accuracy over 4% from 67% to 71.7%, while SVD only increases performance around 1%. If we have successfully targeted memorized features, then we should see that the images that were memorized should switch to predicting their ground truth (GT) labels. K-FAC successfully raises the ground truth accuracy up to 66.5% while SVD reaches 58.9%.

Stress tests Drawing from the positional perturbation stress tests outlined by Huang et al. (2024), we conducted a similar evaluation comparing K-FAC against BSN. For space we include these in

[2]We use a sweep to find which layers to edit, which requires memorization labels to see if the edit is effective. With better understanding we may be able to pick which layers to edit without ever having labels for memorized sequences.

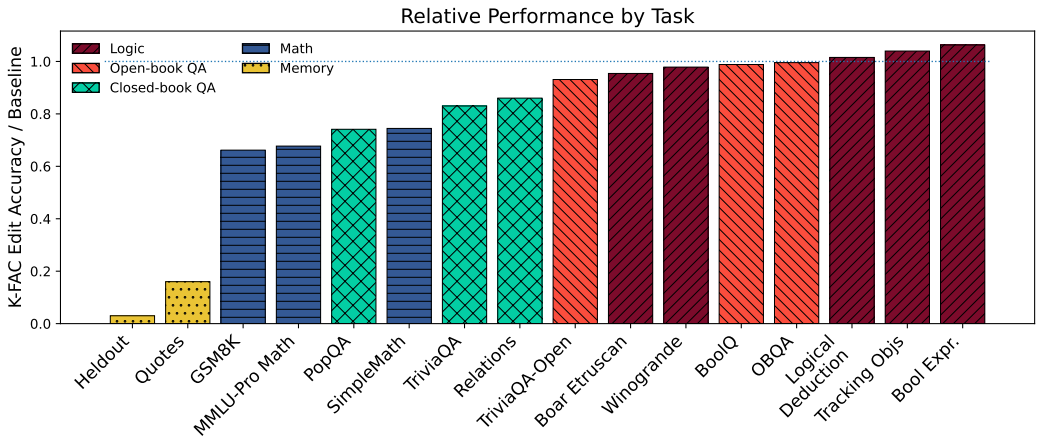

Figure 2: Sensitivity of different kinds of tasks to ablation of flatter eigenvectors. Parametric knowledge retrieval, arithmetic, and memorization are brittle, but openbook fact retrieval and logical reasoning is robust and maintain around 100% of original performance.

Appendix H, but we find far less sensitivity to positional perturbations in both K-FAC and BSN than the older methods analyzed in prior work.

7 Spectrum of Memorization to Reasoning in Downstream Behaviors in LMs

In traditional classification models, the distinction between memorization and generalization is stark and exhaustive: a label is either randomly generated (memorized) or inferred based on training (generalized). LMs have a varied landscape between memorization and reasoning. Here, we connect a wide range of LM behaviors to our story about weight space curvature and demonstrate tasks that have varying sensitivity to perturbations in weights. We demonstrate a spectrum of behaviors between pure memorization and pure reasoning that maps to our measurement of sharpness, and notably, that mathematical reasoning is highly brittle (Nikankin et al., 2025), while difficult non-numerical logical reasoning is among the most robust behaviors.

Setup We use the OLMES evaluation suite (Gu et al., 2025) to measure performance on edited models across benchmarks. We target benchmarks four main categories of tasks: **Closed-book fact retrieval**, **Open-book fact retrieval**, **Logical Reasoning**, and **Math** (arithmetic heavy). Open book retrieval involves question answering where there is a source available in context, whereas closed-book requires retrieval of facts directly from parametric knowledge. For example, TriviaQA is typically closed-book, but can be made open-book by including the relevant Wikipedia page (we refer to this as TriviaQA-Open). We include three non-standard datasets: Boar Etruscan (McCoy et al., 2023), which is an in-context constructed fake language like pig-latin. In order to perform well, models must rely solely on reasoning about rules provided in context, Relations (Hernandez et al., 2024), which is a dataset of factual relations such as "capital-of-country" (we use the 26 factual relations), and SimpleMath which is a generated dataset of two digit addition/subtraction problems (fed to the model 5-shot, with no other context).

Results In Figure 2, we report a subset of benchmarks covering logic, fact recall, math, and our datasets of memorized sequences as a proportion of the unedited models accuracy. We find a mostly smooth drop off from logical reasoning (95-106% retention of baseline), open-book QA (93-99% retention), closed-book QA (74-86% retention), math (66-74%), and memorization (3-16%). Note that outside of the domains of math, and closed book QA, we find that K-FAC edited models perform very well compared to baseline (and often better than BSN), such as on CommonsenseQA, which doesn't cleanly fall into any of these categories. See Appendix J for details. The ranges of degradation we see reflect behavioral brittleness to weight perturbations specific to the domain of the task.

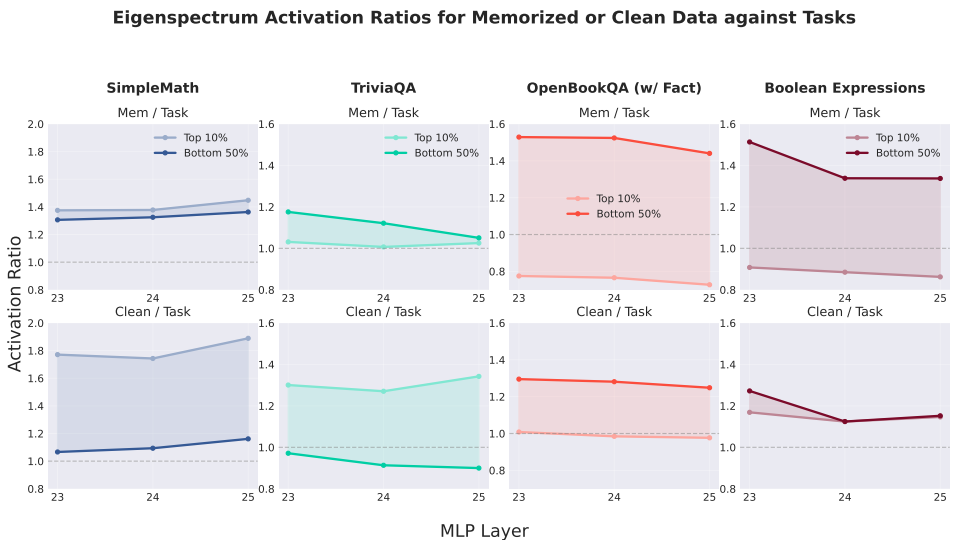

Figure 3: Eigenvector activation ratios for three different tasks compared against either clean or memorized data. The top 10% and bottom 50% bands of the curvature eigenbasis interact differently with memorized vs. non-memorized data. Large differences between these bands when comparing to memorized data (top row, openbookqa and bool. exprs.) indicate more resemblance to clean data processing, and large differences between these bands when comparing to clean data (bottom row, math and TriviaQA) indicates more similarity to memorization processing.

We can also show that this brittleness is measurable in terms of the magnitude of the activation along a K-FAC eigenvector direction (see §3). Figure 3 shows that interactions with the top and bottom of the curvature eigenbasis are predictive of how brittle they are (where a task falls relative to others in Figure 2). For example, hidden activations from OpenbookQA interact far less with the bottom of the eigenspectrum across layers 23-25 than the memorized data (memorized data interact at ~ 1.6x higher magnitude); therefore, we might expect removing them to affect performance less, which is what we previously saw. The opposite is true for SimpleMath: hidden states interactions with the bottom part of the spectrum skew much more towards this dataset than clean data compared to the top of the spectrum, so we might expect removing them possibly harms performance. Again, this is consistent with the large drop seen in Figure 2). While we find that arithmetic ability is especially brittle, it's not clear from our results that LMs don't also contain delicate structure to solve it (Kantamneni & Tegmark, 2025). Additional discussion and results on math and factual recall are in Appendices K and L.

8 Conclusion

We showed that loss-curvature provides a unifying lens for separating memorization from generalization in Transformers: the K-FAC curvature basis disentangles weight directions that support shared, reusable structure (top of the spectrum) from those that chiefly underwrite recitation and brittle behaviors (bottom of the spectrum). Leveraging this finding, we introduced a weight-editing method that preserves a targeted fraction of curvature mass and, across LMs and ViTs, strongly suppresses untargeted memorization while maintaining model coherence and, in the vision setting, even improving validation accuracy. Compared to a supervised unlearning baseline (BSN), our approach requires no forget set, achieves lower perplexity and markedly stronger generalization to unseen memorized content, and exhibits competitive robustness under stress tests. Extending beyond verbatim recall, our analyses position downstream behaviors along a memorization–reasoning continuum: arithmetic and closed-book fact retrieval rely more on low-curvature directions and are disproportionately impacted by edits, whereas open-book and non-numerical logical reasoning are largely preserved or occasionally improved. These results (i) reconcile instance-level sharpness with population-level flatness, (ii) offer practical tools for recitation-reducing model editing, and (iii) go beyond previous results in finding curvature signatures for a range of model behaviors beyond strict memorization and generalization.

REFERENCES

Michael Aerni, Javier Rando, Edoardo Debenedetti, Nicholas Carlini, Daphne Ippolito, and Florian Tramèr. Measuring non-adversarial reproduction of training data in large language models, 2024. URL https://arxiv.org/abs/2411.10242.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, pp. 267–284, 2019.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. Do localization methods actually localize memorized data in llms? a tale of two benchmarks. In *NAACL-HLT*, 2024.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Isha Garg, Deepak Ravikumar, and Kaushik Roy. Memorization through the lens of curvature of loss function around samples. In *International Conference on Machine Learning*, pp. 15083–15101. PMLR, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5005–5033, 2025.

Yoav Gur-Arieh, Clara Suslik, Yihuai Hong, Fazl Barez, and Mor Geva. Precise in-parameter concept erasure in large language models. *arXiv preprint arXiv:2505.22586*, 2025.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Jing Huang, Diyi Yang, and Christopher Potts. Demystifying verbatim memorization in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10711–10732, 2024.

Ajay Kumar Jaiswal, Yifan Wang, Lu Yin, Shiwei Liu, Runjin Chen, Jiawei Zhao, Ananth Grama, Yuandong Tian, and Zhangyang Wang. From low rank gradient subspace stabilization to low-rank weights: Observations, theories, and applications. In *Forty-second International Conference on Machine Learning*, 2025.

Dongjae Jeon, Dueun Kim, and Albert No. Understanding memorization in generative models via sharpness in probability landscapes. *CoRR*, 2024.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7403–7412, 2023.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Young In Kim, Pratiksha Agrawal, Johannes O Royset, and Rajiv Khanna. On memorization and privacy risks of sharpness aware minimization. *CoRR*, 2023.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Pratyush Maini, Michael C Mozer, Hanie Sedghi, Zachary C Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In *Proceedings of the 40th International Conference on Machine Learning*, pp. 23536–23557, 2023.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Tarun Ram Menta, Susmit Agrawal, and Chirag Agarwal. Analyzing memorization in large language models through the lens of model attribution. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10661–10689, 2025.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Language models implement simple word2vec-style vector arithmetic. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5030–5047, 2024.

Jack Merullo, Noah A Smith, Sarah Wiegreffe, and Yanai Elazar. On linear representations and pretraining data frequency in language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

John X Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much do language models memorize? *arXiv preprint arXiv:2505.24832*, 2025.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*, 2025.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Senthooran Rajamanoharan, Neel Nanda, János Kramár, and Rohin Shah. Fact finding: How to think about interpreting memorisation (post 4) — AI alignment forum, 2023. URL https://www.alignmentforum.org/posts/JRcNNGJQ3xNfsxPj4/fact-finding-how-to-think-about-interpreting-memorisation.

Deepak Ravikumar, Efstathia Soufleri, Abolfazl Hashemi, and Kaushik Roy. Unveiling privacy, memorization, and input curvature links. In *International Conference on Machine Learning*, pp. 42192–42212. PMLR, 2024.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mansi Sakarvadia, Aswathy Ajith, Arham Mushtaq Khan, Nathaniel C Hudson, Caleb Geniesse, Kyle Chard, Yaoqing Yang, Ian Foster, and Michael W Mahoney. Mitigating memorization in language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *The Twelfth International Conference on Learning Representations*, 2023.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Niklas Stoehr, Mitchell Gordon, Chiyuan Zhang, and Owen Lewis. Localizing paragraph memorization in language models. *arXiv preprint arXiv:2403.19851*, 2024.

David Yunis, Kumar Kshitij Patel, Samuel Wheeler, Pedro Savarese, Gal Vardi, Karen Livescu, Michael Maire, and Matthew R Walter. Approaching deep learning through the spectral dynamics of weights. *arXiv preprint arXiv:2408.11804*, 2024.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning*, 2024.

A Appendix

B Details on K-FAC Activation Ratio Measurements

Equation 1 gives us a proxy for the curvature of the loss around a given weight matrix $\mathbf{W}$ computed as the Kronecker product of $\mathbf{A}$, the covariances of $\mathbf{W}$'s incoming activations, and $\mathbf{G}$, the covariances of $\mathbf{W}$ output-side gradients. Materializing this product is computationally infeasible, but hypothetically, the eigendecomposition of it will give us a basis ordered by how much the loss landscape curves in each direction. We can project hidden states in the model (right before $\mathbf{W}$) onto subsets of this basis and measure the norm, in order to understand . If $\mathbf{A}$ has eigenpairs $(\lambda, \mathbf{v})$ and G has eigenpairs $(\mu, \mathbf{u})$ (i.e., the ith eigenvalue and eigenvector), then an eigenvalue of $\mathbf{F}$ is $\kappa = \lambda\mu$ and the corresponding eigenvector is $\mathbf{z} = \mathbf{v} \otimes \mathbf{u}$. Note that for a weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, then $\mathbf{z} \in \mathbb{R}^{mn}$. Let $\mathbf{Z} := \text{unvec}(\mathbf{z}) \in \mathbb{R}^{m\times n}$. For a hidden state $\mathbf{x}$, we take the L2 norm of $\mathbf{Zx}$ to measure this mode-specific response/sensitivity of the output to moving $\mathbf{W}$ along the eigenvector $\mathbf{z}$. We hypothesize that memorized and non-memorized data have different activations along different percentile bands of the curvature spectrum, indicating disentanglement in this basis. We sort the eigenvalues of the approximate Fisher by $\lambda_i\mu_j$ and sample eigenvectors[3] from the top 10%, 10-25%, 25-50%, and bottom 50% of eigenvectors, computing $||\mathbf{Zx}||_2$ for each sampled $\mathbf{Z}$ over 10k hidden states. We report the average norm for data in each of these bands.

[3]We sample instead of computing each one, since there are millions.

C PRIMER ON THE EIGENDECOMPOSITION OF A AND G

This section provides background on how to think about the eigenvectors and eigenvalues of the Hessian, as approximated by the K-FAC factorization $\mathbf{F} \approx \mathbf{G} \otimes \mathbf{A}$. For a given weight matrix, recall that $\mathbf{A}$ is the covariance matrix of the activations going into it, and that $\mathbf{A} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$. $\mathbf{G}$ is the covariance matrix of the gradients on the output side of the matrix, and $\mathbf{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$.

Notice that we have $d_{\text{in}} * d_{\text{out}}$ eigenpairs in the Hessian. The approximate eigenvalues of the FIM are the products between each of the eigenvalues of the $\mathbf{G}$ and $\mathbf{A}$ matrices from K-FAC, and the corresponding eigenvectors are the Kronecker products between the eigenvectors of $\mathbf{G}$ and $\mathbf{A}$.

D DATASETS

Dolma We mine memorized continuations from Dolma to obtain on-distribution memorization examples. Fixed-length windows $[64 \mid 48]$ are sampled per document and a window is labeled memorized iff the teacher-forced argmax at each of the 48 suffix positions equals the gold token. Positives are aggressively deduplicated to avoid inflation from near-identical suffixes (e.g., templatic code/comments). The resulting 1000 sequences are split evenly: one half trains BSN (unlearning) and sweeps K-FAC, and the other half validates both.

Historical Quotes For each quote (length ≥ 9 tokens), the prefix is all but the last 8 tokens and the suffix is the final 8. We greedily generate 8 tokens from the prefix and mark memorized on exact match. As quotes have canonical phrasing and high surface regularity, an exact-match is meaningful and less sensitive to trivial paraphrases. This 512 dataset is used strictly for validation to check whether methods preserve non-target knowledge while removing targeted memorization.

E NDCG@10 (TOKEN-RANKING OVERLAP)

For each token position t, the frozen baseline provides a ranked list of its top-K next-token predictions, $B_t = \{b_{t,1}, \ldots, b_{t,K}\}$. After editing, the model produces its own top-K ranking $\hat{y}_{t,1:K}$. We assign graded relevance scores based on the presence and rank order of the edited model's predictions within the baseline set:

$$\text{rel}(r) = \begin{cases} K - r + 1, & \text{if } \hat{y}_{t,r} \in B_t, \\ 0, & \text{otherwise.} \end{cases}$$

We then compute the Discounted Cumulative Gain (DCG), normalized by the Ideal DCG (IDCG), resulting in the normalized Discounted Cumulative Gain (nDCG):

$$\text{DCG}_t = \sum_{r=1}^{K} \frac{\text{rel}(r)}{\log_2(r+1)}, \quad \text{IDCG}_K = \sum_{r=1}^{K} \frac{K - r + 1}{\log_2(r+1)}, \quad \text{nDCG}_t = \frac{\text{DCG}_t}{\text{IDCG}_K} \in [0, 1].$$

We compute nDCG@10 on the first 200k tokens of the held-out `pile10k` dataset. Intuitively, this measures how closely the edited model's token-ranking aligns with the baseline, capturing *local preference drift*. We specifically chose ranking rather than probabilities to isolate the ordering of high-probability tokens, as these largely determine predictive entropy. We report the mean nDCG@10 across positions (higher is better) as an indication of how faithfully the model preserves the baseline's preference structure post-edit.

F HYPERPARAMETERS

Model Settings For K-FAC: The full hyperparameter search details can be found below. In LMs we edit layers 23, 24, and 25 at 60% energy retained in the up and gate projections in MLPs. In ViTs, we edit layers 0 and 11 to 75% energy on both up and down MLP projections.

For BSN: The best BSN settings for editing the language model were a loss weight of 0.7, 5 epochs, a sparsity ratio of 0.0015, and a learning rate of 0.3.

F.1 KFAC Compression Configuration

We systematically explored applying K-FAC compression to selected Transformer MLP layers of the OLMo-2 models. Our experiments focused on two primary hyperparameters:

- **Energy threshold:** Instead of selecting a fixed number of eigenvectors, we retained eigenvectors based on a cumulative "energy" threshold—the fraction of the total eigenvalue sum preserved. We tested thresholds ranging from 60% (stronger compression) to 90% (milder compression), evaluating their effects for gate, up, and down MLP projections
- **Layer selection:** We tested subsets from the 32 MLP layers, targeting early, intermediate, and deep parts of the model, both individually and in combinations upto three layers.

This hyperparameter search aimed to balance memorization suppression and overall model performance.

F.2 Balanced Subnet (BSN) Configuration

For Balanced Subnet (BSN), we started from the original authors' implementation, making minimal adjustments necessary to handle OLMo-2's Transformer architecture and optimize performance given its larger parameter set.

We began with hyperparameter ranges recommended by the BSN authors, then expanded them based on initial results. Our final hyperparameter search included:

- **Ratio:** Controls mask sparsity. Expanded to $[0.001, 0.05]$.
- **Loss weighting:** Balances clean vs. memorized examples. Expanded to $[0.1, 0.3, 0.5, 0.7, 0.9]$.
- **Epochs:** Expanded to 1–10.
- **Include gate:** Optionally includes masking the MLP gate projection

G Perplexity (clean text).

We compute perplexity on clean, held-out text derived from the held-out `pile10k` dataset, following Balanced Subnet (BSN) evaluation approach. Perplexity is measured both before and after applying memorization edits, serving as a baseline metric to identify unintended deterioration in the model's general language modeling capabilities.

H Stress Tests

Huang et al. (2024) reported substantial sensitivity to positional perturbations, with average exact match lengths increasing from 19 to 35 tokens (+16 tokens) for gradient ascent, and from 23 to 36 tokens (+13 tokens) for sparse fine-tuning. By comparison, our K-FAC method showed a smaller absolute increase, from 6.6 to 13.3 tokens (+6.7 tokens), and BSN increased from 4.6 to 10 tokens (+5.4 tokens). Thus, while positional perturbations did increase extractable memorization in our experiments, these results indicate K-FAC, alongside BSN, demonstrate greater robustness under positional perturbations compared to previously evaluated methods.

Method	Original	Perturbed
BSN	4.6 $\pm$ 10.45	9.79$\pm$14.36
K-FAC	6.64$\pm$8.6	13.27$\pm$9.34

Table 3: Effect of positional perturbation stress tests on memorization extraction. "Original" refers to unperturbed prompts, and "Perturbed" refers to prompts with positional perturbations as described by (Huang et al., 2024)

Method	nDCG@10	Perplexity
Baseline	1.000	19.04
BSN	0.97	23.59
K-FAC (ours)	0.91	22.84

Table 4: Coherence metrics

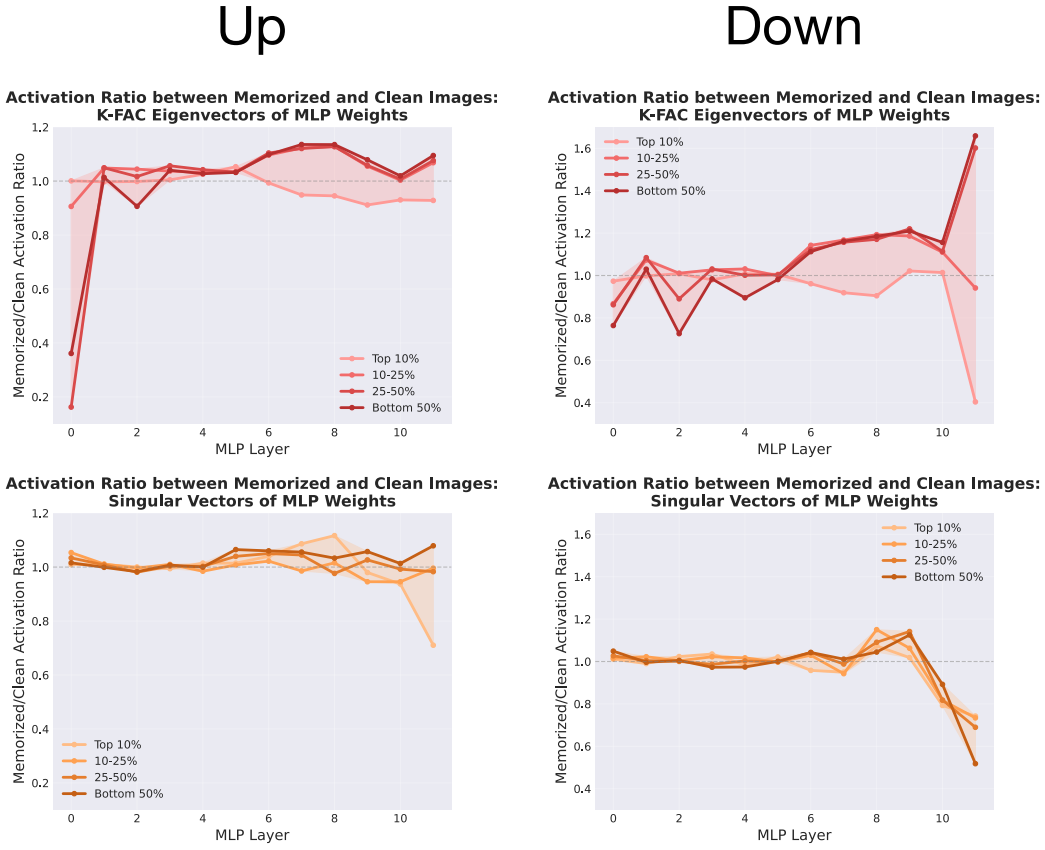

Figure 4: Up and Down projections of the MLPs in ViT-Base

I ViT Results

I.1 ViT K-FAC and SVD Spectral Disentanglement

In the main paper we show the down projections in the MLPs for the ViT model we use, here we also show the up projections in Figure 4. Since we edit layers 0 and 11, these show that the disentanglement in those layers specifically.

J Further Benchmark Results on LMs

K Analysis of Math

We find that arithmetic is specifically hurt by the K-FAC edit. This could be because arithmetic problems themselves are memorized (at the 7B scale), or because they require narrowly used directions to do precise calculations. We find it interesting, though, that this is so specifically and negatively affected when seemingly related skills remain intact. Besides the logical reasoning benchmarks included in Figure 2, we find that while MMLU-Pro Math drops to 67% of baseline (23.4% to 15.8%), the computer science subset stays at about the same level (Baseline: 26.1%, K-FAC: 26.3%,

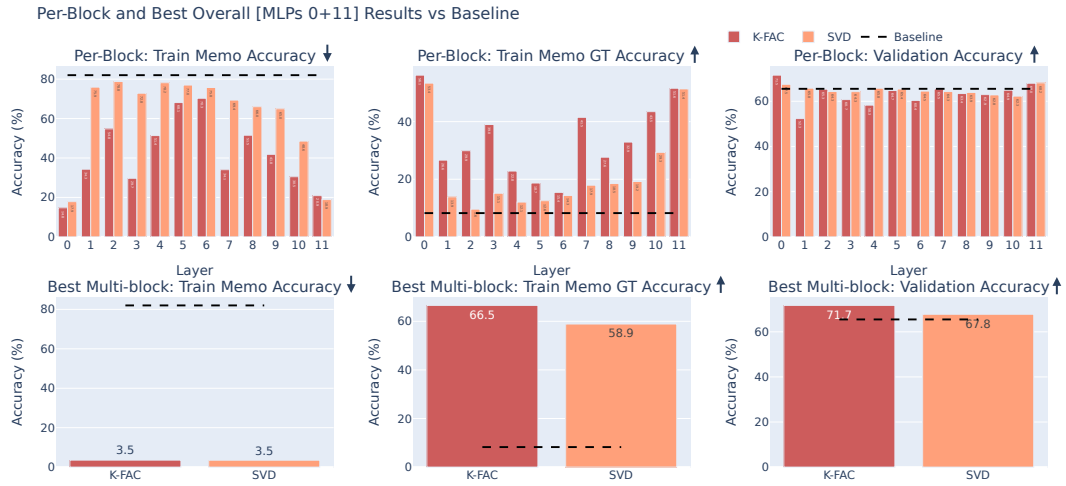

Figure 5: Comparison of K-FAC compression and SVD per MLP block (top) and with the best configuration (bottom) in a ViT model. We find that K-FAC compression generally outperforms SVD, and the best results (compressing layers 0 and 11 simultaneously) aligns with the results in §5, where these layers showed the greatest disentanglement between memorized data and generalizing data. Note that with K-FAC we are able to effectively remove memorization while *substantially improving* generalization performance (validation), and recovering more of the ground truth label on the previously memorized set than SVD.

	Baseline	BSN	K-FAC
TriviaQA	0.780	0.766	0.648
Relations	74.855	0.268	64.390
PopQA	0.807	0.779	0.598
OBQA	0.804	0.790	0.800
CSQA	0.751	0.722	0.731
TriviaQA-Open	0.760	0.708	0.720
OBQA+Fact	0.888	0.884	0.894
BoolQ	0.863	0.853	0.854
GSM8K	0.675	0.610	0.447
Winogrande	0.772	0.761	0.755
BigBench-Hard	0.499	0.463	0.475
MMLU-Pro	0.283	0.270	0.253
MMLU-Pro Math	0.234	0.226	0.158
MMLU-Pro CS	0.261	0.251	0.263

Table 5: Benchmark results comparing BSN and K-FAC, with some subsets of larger datasets included to highlight interesting behaviors, such as the retention of computer science knowledge, but drop in mathematics knowledge in the K-FAC edit.

Example (Boolean Expression Reasoning)

Prompt: False or not (True) and False is

Ground Truth Answer: False

Baseline Continuation: Remember that (i) expressions inside brackets are always evaluated first and that (ii) the order of operations from highest priority to lowest priority is "not", "and", "or", respectively.
We first simplify this expression "Z" as follows: "Z = False or not (True) and False = A or B " where "A = False or not (True) " and "B = False".
Let's evaluate A: A = False or not (True) = not (False) or True = True or True = True.
Let's evaluate B: B = False.
Plugging in A and B, we get: Z = A or B = True or False = True. So the answer is True.

K-FAC Continuation: Remember that (i) expressions inside brackets are always evaluated first and (ii) the order of operations from highest priority to lowest priority is "not", "and", "or". respectively.
We first simplify this expression "Z" as follows: "Z = False or not (True) and False = A and B " where "A = False or not (True) " and "B = False".
Let's evaluate A: A = False or not (True) = not (False) or True = True or True = True.
Let's evaluate B: B = False.
Plugging in A and B, we get: Z = A and B = True and False = False. So the answer is False.

Table 6: Example demonstrating improved Boolean reasoning after K-FAC editing

BSN: 25.1%). An example from GSM8K also shows that despite generating the *exact* same chain of thought as the baseline model, but failed specifically at making the calculation (Figure 7. Not all generations are as similar to the original, but demonstrates how targeted this new deficit is. We include an example of essentially the opposite (same generation but fixing an error) in Boolean Expressions; see Appendix Table 6.

Example (Arithmetic - GSM8K)

Prompt: Seattle has 20 sheep. Charleston has 4 times as many sheep as Seattle. Toulouse has twice as many sheep as Charleston. How many sheep do they have together?

Ground Truth Answer: 260

Baseline: Seattle has 20 sheep. Charleston has 4 times as many sheep as Seattle. So Charleston has 4 x 20 = 80 sheep. Toulouse has twice as many sheep as Charleston. So Toulouse has 2 x 80 = 160 sheep. Together, they have 20 + 80 + 160 = 260 sheep. So the answer is 260.

K-FAC Edit: Seattle has 20 sheep. Charleston has 4 times as many sheep as Seattle. So Charleston has 4 x 20 = 80 sheep. Toulouse has twice as many sheep as Charleston. So Toulouse has 2 x 80 = 160 sheep. Together, they have 20 + 80 + 160 = 220 sheep. So the answer is 220.

Table 7: Example highlighting arithmetic reasoning error introduced by K-FAC editing (in red), compared to correct baseline output (in blue).

L Analysis of Fact Retrieval

We explore whether specific types of facts are more brittle to a K-FAC edit. A natural question is whether the frequency of a fact changes the probability that it is not forgotten by an edit. While one intuition is that frequent facts may be more redundantly stored, Merullo et al. (2025) show that more frequent relations are also more likely to be represented with a shared representation for that relation, of the type described in Hernandez et al. (2024). We show that more frequent relations in the Relations dataset are less affected by our K-FAC edit to lower eigenvalues of the Hessian (Figure 6), but we do not weigh in whether this is due to shared structure (which 'belongs' to the top of the spectrum of eigenvalues) or due to redundant representation. We present the relations in the order of least to most shared structure for OLMo-1 7B (a different model than we use) as reported in Merullo et al. (2025).

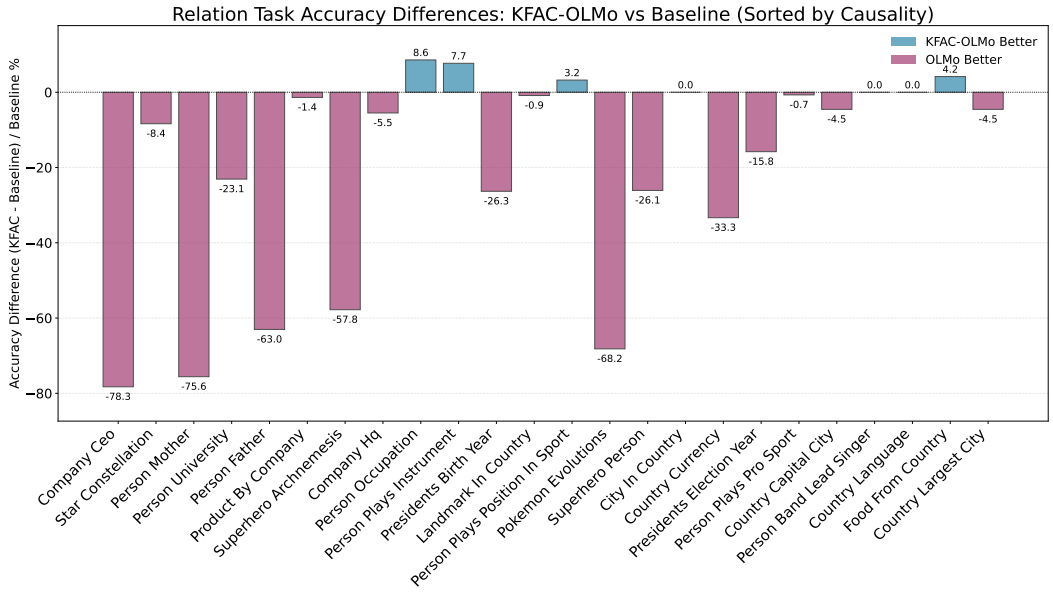

Figure 6: Accuracy change across the relations dataset, sorted according to relations that are likely to support shared linear structure internally (Hernandez et al., 2024). We find that the least frequent/likely to form linear structure (left) have dramatically larger drops than the most frequent, some of which do not change at all.