WEIGHT-ADJUSTED GRADIENTS REVEAL PARAMETER IMPORTANCE AND FAILURE MODES IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding which parameters are influential in Large Language Models (LLMs) is central to improving their efficiency, reliability, and interpretability. We introduce Weight-Adjusted Gradients (WAG), a simple yet effective approach for estimating parameter importance that explicitly captures the interaction between model weights and first-order gradient information and identifies parameters that disproportionately influence model behavior, such as those responsible for collapse phenomena in LLMs. Across a range of models and settings, we show that WAG surfaces a tiny but critical subset of parameters ($\leq$ 0.5 parts per million or 0.00005% of model size) whose modification leads to dramatic degradation in performance, indicating a novel failure mode. These findings also reveal a previously underexplored interplay between weights and gradients, suggesting that parameter importance cannot be fully understood through either signal alone. We demonstrate the practical utility of WAG across several diverse applications, such as expert allocation in Mixture-of-Experts (MoE) architectures, targeted unlearning, mixed-precision quantization, and layer selection for knowledge editing. In sum, WAG can serve as a unified approach for analyzing, debugging, and controlling LLMs, and opens new directions for principled parameter-level interpretation.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated strong performance across a wide range of tasks, becoming a central component in modern systems (Brown et al., 2020; Touvron et al., 2023; OpenAI, 2023). While much of the progress in this area has focused on scaling model size, improving training procedures, and designing new architectures (Kaplan et al., 2020; Hoffmann et al., 2022; Minaee et al., 2024), comparatively less attention has been given to understanding the internal structure of trained models (Ferrando et al., 2024). In particular, identifying which parameters are most influential to model behavior remains an important yet underexplored problem. Such understanding is critical for improving model reliability, enabling efficient adaptation, and supporting interpretability.

A natural way to study this problem is through *parameter importance* estimation. Intuitively, the importance of a parameter can be measured by evaluating how sensitive the model's behavior is to perturbations of that parameter. Existing approaches typically rely on either parameter magnitudes or gradient-based signals. Magnitude-based methods, such as those used in pruning, assume that parameters with larger absolute values are more important (Han et al., 2015; Gale et al., 2019). Gradient-based approaches instead measure sensitivity through first-order information, for example using saliency or influence approximations (LeCun et al., 1989; Molchanov et al., 2017; Vitel & Chhabra, 2026). Second-order methods further incorporate curvature information, but often incur high computational cost and rely on approximations that may not scale well to modern LLMs (Hassibi & Stork, 1992; Frantar et al., 2023; Askari et al., 2025). Despite these efforts, the underlying relationship between weight and gradient information remains poorly understood, and existing methods offer little insight into how this interaction relates to catastrophic failure modes such as severe performance degradation.

In this paper, we examine parameter importance from the perspective of the interaction between parameter values and gradient signals. We observe that treating these two sources of information independently can overlook parameters that are both large in magnitude and highly sensitive to optimization dynamics. Motivated by this, we propose *Weight-Adjusted Gradients (*WAG*)*, a simple metric that combines zeroth-order and first-order information to estimate parameter importance. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

formulation is straightforward and can be computed efficiently during standard training or evaluation without requiring second-order information or retraining.

Our analysis shows that WAG identifies a *tiny* subset of parameters that have a disproportionate impact on model performance. In particular, perturbing these parameters leads to significant degradation and model collapse. We find that many of these parameters are not captured by existing magnitude-based or gradient-based metrics, suggesting that important aspects of model behavior are missed when considering these signals in isolation. Thus, in this work we study WAG from a theoretical and empirical lens, and subsequently, demonstrate the effectiveness and usefulness of WAG in several practical settings.

Contributions. We summarize our main contributions in this paper as follows:

- We propose *Weight-Adjusted Gradients (*WAG*)* that leverages the unexplored relation between weights and gradients to estimate parameter influence and identify parameters, when masked, lead to significant performance degradation. Without requiring expensive computation, WAG captures the sensitivity of loss to multiplicative parameter perturbations.
- We undertake empirical analyses across several LLMs and datasets, and also provide theoretical interpretations for the parameters captured by WAG and the reason why masking these induces a catastrophic degradation in performance.
- Finally, we demonstrate the effectiveness and applicability of WAG with competitive performance in diverse practical applications, including expert allocation in Mixture-of-Experts (MoE) architectures, targeted unlearning, mixed-precision quantization, and knowledge editing.

2 Related Works

We now review the literature along three aspects. First, we discuss how parameter importance has been quantified in prior work. Next, we examine how these measures have been operationalized in downstream applications. Finally, we discuss past work on failure modes in LLMs, specifically model collapse. We also provide a more comprehensive literature review in Appendix A due to space constraints.

General Parameter Importance. Existing approaches primarily characterize parameter importance from three perspectives: (1) magnitude-based importance, (2) gradient-based sensitivity, and (3) curvature-based measures. Magnitude-based approaches include utilizing the weights and activations (Hanson & Pratt, 1988; Gale et al., 2019; Han et al., 2015; Sun et al., 2024), since they have a direct impact on the output. Gradients estimate parameter importance by measuring sensitivity to parameter removal (Molchanov et al., 2017). Curvature-based approaches further account for second-order information, such as Optimal Brain Damage through Hessian diagonal approximation (LeCun et al., 1989), Optimal Brain Surgeon (OBS) through full Hessian (Hassibi & Stork, 1992), and Optimal Brain Apoptosis through computationally efficient Hessian (Sun et al., 2025).

Applications of Estimating Parameter Influence. The parameter importance estimates are widely operationalized in pruning methods, which remove parameters or structured groups based on their estimated contribution to model performance. Recent approaches for LLMs include magnitude itself (Han et al., 2015), magnitude-activation based Wanda (Sun et al., 2024), gradient-based criteria (Das et al., 2023; Yang et al., 2025b; Zimmer et al., 2025). The parameter-importance measures have also been applied to layer-wise sparsity allocation. For example, AlphaPruning (Lu et al., 2024) uses spectral characteristics of layer weights, OWL (Yin et al., 2024) relies on activation outliers, and LayerIF (Askari et al., 2025) estimates layer-wise influence.

Beyond pruning and sparsity allocation, influence-based layer quality estimates have also been applied to continual learning (Chhabra et al., 2025) for catastrophic forgetting prevention. Regularization-based methods, Elastic Weight Consolidation (Kirkpatrick et al., 2017), Synaptic Intelligence (Zenke et al., 2017), and Memory Aware Synapses (Aljundi et al., 2018) address this by penalizing changes to parameters deemed important. Recent work (Gao & Liu, 2025) provides a unified theoretical account of this family by formally defining parameter importance via influence functions.

Uniform allocation of LoRA experts across transformer layers has been questioned by recent work showing that expert redundancy and routing overfitting degrade performance (Gao et al., 2025). AlphaLoRA (Qing et al., 2024) addresses this using spectral layer quality metrics to inform allocation. LayerIF (Askari et al., 2025) instead derives task-specific allocation budgets from layer-wise influence.

Parameter influence is also used in the task of mixed-precision quantization, where parameter groups with the most saliency are preserved in high precision and the others in low precision. Recent studies, TACQ (Xiao et al., 2025), PB-LLM (Shang et al., 2024) and SliM-LLM (Huang et al., 2025), employ parameter magnitude and OBS criteria as a measure of the saliency of the groups. Moreover, gradient-based methods like Salient Layers Editing Model (Mishra et al., 2024) and Gradient Layer Analysis (Datta et al., 2026) are used for important editing layer identification in knowledge editing.

Identifying Failure Modes in LLMs. Failure due to model collapse in LLMs broadly manifests in two forms: collapse arising from corrupted training data, and collapse arising from architectural modification. Recent work has characterized the phenomenon of model collapse arising from recursive training on synthetic data without access to human-generated data (Shumailov et al., 2023). Recent work (Wang et al., 2024a) shows that recursive training introduces error accumulation across generations, leading to irreversible model collapse. At the architectural level, a recent study (Shi et al., 2026) found that performance remains stable until pruning reaches the *Silent Phase*, after which collapse is abrupt and irreversible. On the other hand, recent studies show that collapse is driven by a sparse set of critical parameters (Reda et al., 2025). However, these works do not identify the critical parameters or the minimum number of weights needed to induce model collapse.

3 The WAG Metric: Theory and Empirical Insights

This section introduces WAG, its theoretical foundation as a parameter sensitivity measure under multiplicative perturbations, and extensive empirical analysis.

3.1 Weight-Adjusted Gradients

We introduce *Weight-Adjusted Gradients (WAG)*, a parameter importance metric that combines parameter magnitude with first-order gradient information. Magnitude-based methods assume that parameters with larger absolute values contribute more strongly to model performance. While effective in some settings, they ignore local loss sensitivity and may assign high importance to functionally inactive parameters. Conversely, gradient-based methods capture instantaneous sensitivity but may overemphasize parameters with large gradients despite having negligible scale or limited functional impact. WAG combines these complementary signals by measuring their interaction. Formally, for a parameter vector $\theta \in \mathbb{R}^d$ and loss function $\ell(z;\theta)$ evaluated on a test sample z, we define the WAG score of parameter θ_i as:

$$\boxed{\mathrm{WAG}_i = -\theta_i \frac{\partial \ell(z;\theta)}{\partial \theta_i}.} \tag{1}$$

The negative sign aligns the score with the local descent direction of the loss. The metric can be computed per parameter, aggregated across layers, or accumulated across datasets depending on the application (e.g. averaging WAG values).

WAG is closely related to several classical ideas in optimization and pruning. The quantity $\theta_i \frac{\partial \ell}{\partial \theta_i}$ appears implicitly in first-order Taylor approximations for network saliency and pruning (LeCun et al., 1989; Molchanov et al., 2017; Lee et al., 2019). Related notions have also been studied in second-order saliency methods and sensitivity-based pruning approaches (Hassibi & Stork, 1992; Gale et al., 2019). However, prior work has treated these interaction terms as auxiliary approximations for pruning objectives rather than as a direct object of study for understanding parameter importance in large-scale models such as LLMs.

3.2 Theoretical Analysis

Theoretically, we interpret WAG through a *log-space parameterization* and show that it characterizes local sensitivity to multiplicative parameter perturbations, identifying directions of greatest sensitivity. This provides a view of trained LLMs as *scale-balanced* systems, where the most fragile coordinates have the largest absolute WAG scores. Detailed proofs for all theorems are provided in Appendix B.

WAG as a Gradient in Log-Parameter Space. We first show that WAG arises naturally when expressing optimization in logarithmic parameter coordinates.

Theorem 3.1 (WAG as Log-Parameter Gradient)**.** *Assume $\theta_i \neq 0$ and define the logarithmic parameterization:*

$$u_i = \log |\theta_i|.$$

Then the gradient of the loss with respect to u_i satisfies:

$$\frac{\partial \ell(z;\theta)}{\partial u_i} = \theta_i \frac{\partial \ell(z;\theta)}{\partial \theta_i}.$$

Equivalently:

$$\mathrm{WAG}_i = -\frac{\partial \ell(z;\theta)}{\partial u_i}.$$

Theorem 3.1 shows that WAG is not solely a heuristic combination of weights and gradients. In fact, it exactly corresponds to the gradient of the loss in logarithmic parameter coordinates. Thus, WAG measures sensitivity to relative parameter changes rather than *additive* perturbations. We prove this relative sensitivity claim next.

First-Order Sensitivity Under Multiplicative Perturbations. We can show that WAG naturally characterizes first-order loss variation under multiplicative parameter perturbations.

Theorem 3.2 (First-Order Multiplicative Sensitivity)**.** *Let $\theta' \in \mathbb{R}^d$ be obtained through multiplicative perturbations of the type:*

$$\theta'_i = \theta_i(1 + \epsilon_i),$$

where ϵ_i are sufficiently small. Then the corresponding first-order change in loss satisfies:

$$\ell(z;\theta') - \ell(z;\theta) = -\sum_{i=1}^{d} \epsilon_i \mathrm{WAG}_i + \mathcal{O}(\|\epsilon\|^2).$$

Theorem 3.2 shows that WAG exactly characterizes the first-order effect of relative parameter perturbations on the loss. Parameters with large-magnitude WAG scores therefore correspond to directions that induce disproportionately large functional changes under multiplicative modification.

WAG and Scale-Balanced LLMs. The previous results establish that WAG can be interpreted in two complementary ways. Theorem 3.1 shows that WAG is exactly the gradient of the loss in logarithmic parameter coordinates, while Theorem 3.2 shows that WAG characterizes first-order sensitivity under multiplicative parameter perturbations. These results suggest that trained neural networks are naturally organized in terms of parameter *scales* rather than raw parameter values. Trained LLMs have parameter magnitudes that vary widely across layers and blocks (An et al., 2025), yet behave stably because the loss depends on how these scales are coordinated (Dinh et al., 2017; Ba et al., 2016). Unlike additive probes of this geometry (Foret et al., 2021), WAG measures sensitivity to *relative* scale changes. Based on this insight, we next show that WAG identifies the direction along which this coordination is most fragile.

Theorem 3.3 (WAG Characterizes the Most Sensitive Log-Scale Direction)**.** *Let $w = (\mathrm{WAG}_1, \ldots, \mathrm{WAG}_d) \in \mathbb{R}^d$ denote the WAG vector, and consider multiplicative parameter perturbations: $\theta'_i = \theta_i(1 + \epsilon_i)$, with perturbation vector: $\epsilon = (\epsilon_1, \ldots, \epsilon_d)$. Then, among all multiplicative perturbations satisfying: $\|\epsilon\|_2 = r$, the maximal first-order increase in loss is $r\|w\|_2$, and is attained uniquely by perturbations parallel to $-w$. Equivalently, the WAG vector defines the direction of greatest local sensitivity in logarithmic parameter space.*

Therefore, no multiplicative perturbation with the same norm can produce a larger first-order increase in loss. Since Theorem 3.1 shows that WAG is the gradient in logarithmic parameter coordinates, it follows that the WAG vector defines the direction of greatest local sensitivity in log-parameter space. The norm $\|w\|_2$ measures the corresponding worst-case sensitivity, while coordinates with large $|\mathrm{WAG}_i|$ contribute most strongly to this direction.

This provides a geometric interpretation of *scale balance*. Trained models can be viewed as configurations whose parameter scales jointly maintain low loss, with WAG identifying the directions in which this balance is most fragile. In sum, parameters with large $|\mathrm{WAG}_i|$ are the most sensitive to local changes in scale, which helps explain why perturbing or masking them can cause disproportionate performance degradation, as we show empirically in Section 3.3. We provide a more detailed discussion on this alongside the proof in Appendix B.3.

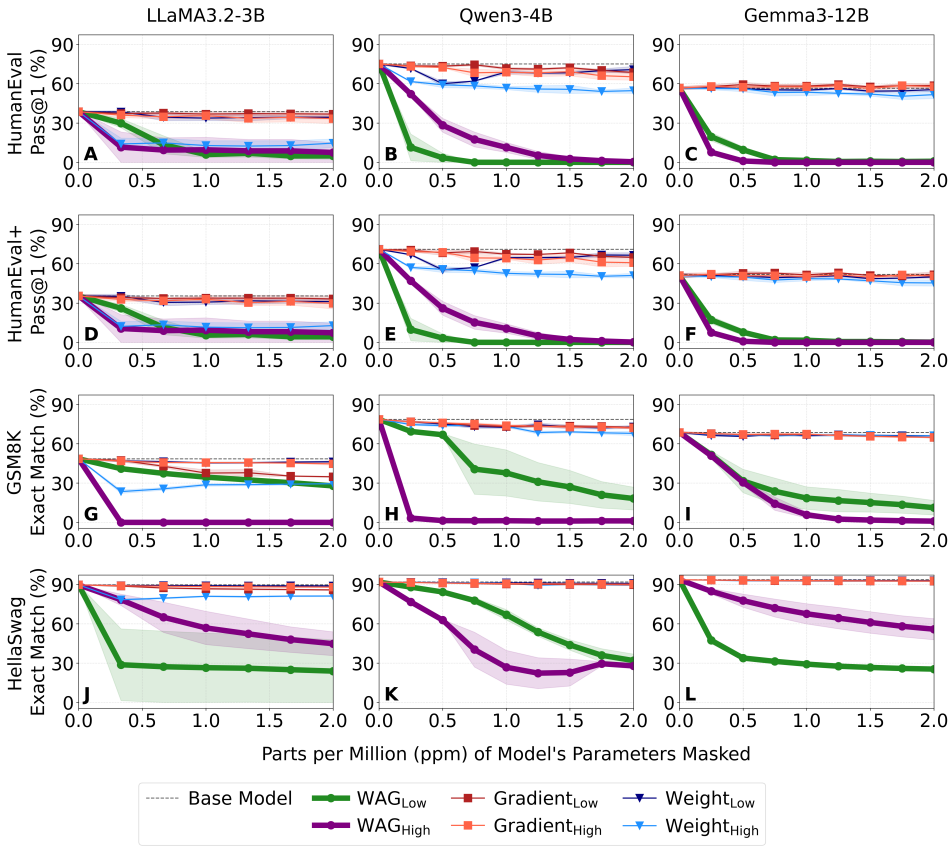


Figure 1: Visualization of model collapse due to masking just ≤ 2 parts per million (i.e., $\leq 0.0002\%$) parameters across LLaMA3.2-3B (A, D, G and J), Qwen3-4B (B, E, H and K), and Gemma3-12B (C, F, I and L) models evaluated on *HumanEval* (A, B and C), *HumanEval+* (D, E and F), *GSM8K* (G, H and I), and *HellaSwag* (J, K, and L) datasets. The performance is measured using *Pass@1* and *exact match* accuracy. Parameters are selected based on having lowest gradient $\rightarrow$ Gradient$_{\text{Low}}$, highest gradient $\rightarrow$ Gradient$_{\text{High}}$, lowest weight $\rightarrow$ Weight$_{\text{Low}}$, highest weight $\rightarrow$ Weight$_{\text{High}}$, lowest WAG $\rightarrow$ WAG$_{\text{Low}}$, and highest WAG $\rightarrow$ WAG$_{\text{High}}$ values respectively. Masking the tiny number of WAG$_{\text{Low}}$ and WAG$_{\text{High}}$ identified parameters leads to rapid model collapse.

3.3 Empirical Analysis

Impact of masking WAG parameters with extreme scores. To empirically assess the impact of WAG-identified parameters, we examine the effect of masking parameters with the *lowest*, *highest*, and *largest-magnitude* WAG values, denoted as WAG$_{\text{Low}}$, WAG$_{\text{High}}$, and $|\text{WAG}|$, respectively. We compare these effects with gradient- and weight-based parameter selection to examine whether the observed collapse behavior of WAG is inherited from WAG's constituent components. We evaluate free-form generation performance on coding, math, and commonsense reasoning tasks using *HumanEval* (Chen et al., 2021), *HumanEval+* (Liu et al., 2023), *GSM8K* (Cobbe et al., 2021), and *HellaSwag* (Zellers et al., 2019) datasets on LLaMA3.2-3B (Touvron et al., 2023), Qwen3-4B (Yang et al., 2025a), and Gemma3-12B (Team et al., 2025).

Figure 1 indicates that the WAG$_{\text{Low}}$ and WAG$_{\text{High}}$ identified parameters behave distinctly compared to magnitude and gradient identified parameters. Masking a tiny number of parameters based on WAG leads to performance degradation at a higher rate and this effect is consistent across several LLMs and datasets. For LLaMA3.2-3B on most datasets, we can see that average performance reaches near-zero exact match at just $\approx$1,500 masked parameters (or 0.5 ppm). A similar trend is also observed for $|\text{WAG}|$ and, due to space limitations, is presented in Appendix D.1.

Qualitative analysis of model collapse induced by WAG-parameter removal. We also qualitatively inspect the generations of the corresponding base model and the masked models to study this collapse. We present an example for Qwen3-4B on *GSM8K* in Figure 2, and owing to space constraints, provide

Prompt: *Walking with his 3 younger siblings, Greg found $20 and split it equally. How much did each get?*

Base Model

```
There were 1 + 3 = 4 of them who found the money.
Each got 20 / 4 = $5.    ####  5
```

Observation: *Correct answer.*

|WAG|

```
GregGregGregGregGregGregGregGregGregGregGregGregGregGregGreg
GregGregGregGregGregGregGregGregGregGregGregGregGregGregGreg ...
```

Observation: *Token "Greg" repeats to 2048 chars; the extracted answer is the same loop.*

Gradient / Weight representative

```
There are 4 of them because 1 + 3 = 4.
Each of them gets $5 because 20 / 4 = 5.    ####  5
```

Observation: *All six gradient/weight variants recover the answer.*

Figure 2: Qualitative example of model failure caused by masking 2 ppm = 0.0002% |WAG|-identified model parameters. Verbatim Qwen3-4B generations on *GSM8K* at 8,000 masked parameters indicate token repetition ("Greg..."). Gradient- and weight-masked variants generate the correct solution.

detailed model continuations for Qwen3-4B on *HumanEval* (4,000 or about 1e-4% parameters masked) and *GSM8K* (8,000 or about 2e-4% parameters masked) in Appendix D.2. As can be seen, WAG-based masking causes both generative failure and loss of reasoning correctness. In coding, the model fails to generate a function body, while in mathematical reasoning, it enters repetition loops (e.g., repeating `Greg` interminably as in Figure 2) or produces fluent but corrupted reasoning. For instance, as shown in Appendix D.2 for WAG_{Low}, the masked model divides $20 among three rather than four siblings, omitting Greg himself. Moreover, this behavior is specific to WAG as under the same masking budget, gradient- and weight-based variants retain fluent, correct solutions.

WAG selected parameter distribution across layers. For more insight into where the masked 0.0002% parameters identified by WAG metrics fall across the model layers, we present the distribution across layers as a heatmap in Figure 3. We see that the majority of the selected tiny set of parameters are located in the last layers, irrespective of the task. However, the fraction is still very low compared to the total size of the layer. At most, 6,665 parameters are identified by |WAG| from a single last layer of size 224M parameters. Thus, only a maximum of 0.0003% of parameters are being masked from a single layer. We see a similar trend across the other LLMs, and present results for these in Appendix D.3. Moreover, we provide additional visualizations of the WAG scores across layers in Appendix D.4. It can be observed that the average WAG scores across layers exhibit a task-specific distribution. Finally, note that early-to-middle layers typically exhibit higher scores.

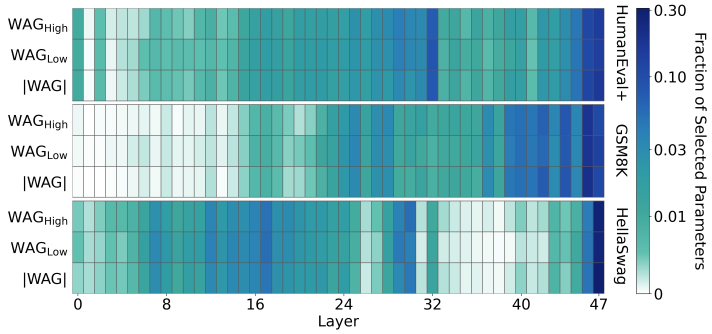


Figure 3: Distribution of 2 parts per million or 0.0002% selected parameters by WAG_{Low}, WAG_{High}, and |WAG| across layers of Gemma3-12B for *HumanEval+*, *GSM8K* and *HellaSwag*.

4 Utilizing WAG in Diverse Applications

We now demonstrate how WAG's ability to characterize local sensitivity of the loss under multiplicative parameter perturbations can be leveraged across several downstream applications. Specifically, we examine: (A) expert allocation in Mixture-of-Experts (MoE) models, (B) efficient and effective

Table 1: Comparison of WAG-guided expert allocation performance compared against AlphaLoRA and MoLA configurations (triangle, rectangle, hourglass) assessed using *Zero-shot* accuracy (%) over 3 seeds, evaluated on *MRPC*, *CoLA*, and *ScienceQA*. The total expert budget is fixed ($N = 140$ for LLaMA3.2-3B, $N = 240$ for Gemma3-12B) with top-2 routing. Best average performance is in **green** and second best is green.

Dataset	AlphaLoRA	MoLA			WAG		
		Triangle	Rectangle	Hourglass	WAG_{Low}	WAG_{High}	$\lvert\text{WAG}\rvert$
LLaMA3.2-3B							
MRPC	$47.17_{\pm 7.279}$	$52.41_{\pm 1.337}$	$50.78_{\pm 2.320}$	$55.15_{\pm 1.117}$	$54.94_{\pm 6.354}$	$54.47_{\pm 1.239}$	$54.59_{\pm 3.229}$
CoLA	$78.62_{\pm 0.838}$	$78.33_{\pm 1.578}$	$77.47_{\pm 1.237}$	$76.93_{\pm 1.175}$	$79.58_{\pm 0.671}$	$79.23_{\pm 0.975}$	$79.99_{\pm 0.305}$
ScienceQA	$53.04_{\pm 0.390}$	$50.75_{\pm 0.378}$	$51.60_{\pm 1.817}$	$50.24_{\pm 1.669}$	$51.09_{\pm 0.859}$	$51.65_{\pm 1.701}$	$46.22_{\pm 1.531}$
Average	$59.61_{\pm 2.265}$	$60.50_{\pm 0.198}$	$59.95_{\pm 1.327}$	$60.77_{\pm 0.364}$	**$61.87_{\pm 2.049}$**	$61.78_{\pm 0.628}$	$60.27_{\pm 1.054}$
Gemma3-12B							
MRPC	$69.66_{\pm 1.418}$	$70.80_{\pm 3.828}$	$73.41_{\pm 2.351}$	$73.43_{\pm 1.919}$	$71.05_{\pm 2.432}$	$73.78_{\pm 0.504}$	$76.44_{\pm 1.045}$
CoLA	$82.42_{\pm 0.565}$	$83.83_{\pm 0.692}$	$83.41_{\pm 0.418}$	$81.72_{\pm 0.338}$	$83.51_{\pm 0.111}$	$83.80_{\pm 0.055}$	$83.41_{\pm 0.534}$
ScienceQA	$92.49_{\pm 0.170}$	$92.57_{\pm 0.182}$	$93.18_{\pm 0.208}$	$93.06_{\pm 0.159}$	$93.36_{\pm 0.197}$	$92.54_{\pm 0.045}$	$93.33_{\pm 0.173}$
Average	$81.53_{\pm 0.291}$	$82.40_{\pm 1.280}$	$83.33_{\pm 0.844}$	$82.74_{\pm 0.612}$	$82.64_{\pm 0.849}$	$83.37_{\pm 0.164}$	**$84.40_{\pm 0.367}$**

machine unlearning through targeted layer updates, (C) mixed-precision quantization via the identification of precision-sensitive weight groups, and (D) layer selection for effective knowledge editing. We evaluate the applications based on free-form generation for LLaMA3.2-3B and Gemma3-12B models. We provide all details of the performance metrics and the experimental implementation for each application in Appendices C and E, respectively.

4.1 Application A: WAG for Mixture-of-Experts Allocation

We first apply WAG to allocate a fixed budget of N parallel experts across the layers of a Mixture-of-Experts model. We allocate more experts to layers containing parameters that are more sensitive to perturbations, as identified by WAG, to better capture the layer's complexity. We instantiate this principle with our three prior criteria: WAG_{Low}, WAG_{High}, and $\lvert\text{WAG}\rvert$.

We evaluate the WAG-based allocations against state-of-the-art approaches such as Heavy-Tailed Self-Regularization theory-driven AlphaLoRA (Qing et al., 2024) and fixed heuristic MoLA configurations (Gao et al., 2025): triangle (2 4 6 8), rectangle (5 5 5 5), and hourglass (8 2 2 8). We report zero-shot performance on *MRPC* (Dolan & Brockett, 2005), *CoLA* (Wang et al., 2018), and *ScienceQA* (Lu et al., 2022) using LLaMA3.2-3B and Gemma3-12B.

Result. Table 1 shows that WAG-guided allocation consistently outperforms both AlphaLoRA and all fixed MoLA configurations on average. On LLaMA3.2-3B, allocation based on the WAG_{Low} scores achieves the best average accuracy of 61.87%, surpassing the strongest baseline MoLA hourglass by 1.10%, with the largest gain on *MRPC*. On Gemma3-12B, the $\lvert\text{WAG}\rvert$ criterion performs best with an average of 84.40%, outperforming the strongest baseline MoLA rectangle by 1.07%. Notably, WAG_{High} remains a strong and stable criterion across both models, ranking second on both LLaMA3.2-3B and Gemma3-12B. Overall, sensitivity-aware expert allocation using WAG is more effective than both MoLA and AlphaLoRA.

4.2 Application B: WAG for Targeted Unlearning

We next apply WAG to *targeted unlearning* to remove the influence of a designated *forget set* from a trained LLM. We build on the Gradient Difference (GradDiff) (Liu et al., 2022) unlearning algorithm, which updates all model parameters. We hypothesize that the information to be removed is concentrated in layers containing the most sensitive parameters to the forget set $\mathcal{D}_f$, which can be identified by WAG_{Low}, WAG_{High}, and $\lvert\text{WAG}\rvert$. Thus, restricting updates to layers identified by WAG should (i) improve unlearning quality by concentrating updates to important layers, and (ii) reduce computational cost since we are only updating a fraction of the model.

We evaluate on the *TOFU* benchmark (Maini et al., 2024), using the *10% forget* split, and compare against GradDiff across models. For each WAG criterion, we select the top 75% of layers based on the number of most WAG-sensitive parameters they contain and apply GradDiff only to those layers.

Table 2: We compare unlearning performance on the *TOFU* forget set using standard GradDiff and WAG-targeted GradDiff, which updates only the top 75% of layers selected by WAG_{Low}, WAG_{High}, and |WAG| respectively, across LLaMA3.2-3B and Gemma3-12B models. We report the performance metrics *Model Utility*, *Forget Truth Ratio (TR)*, and *KS Forget* using the mean ± standard error of the mean over 3 seeds. Best performance is in **green** and second best in green.

Model	Metric	Method			
		GradDiff	GradDiff + WAG_{Low}	GradDiff + WAG_{High}	GradDiff + |WAG|
LLaMA3.2-3B	Model Utility	$0.565_{\pm 0.014}$	$\mathbf{0.578}_{\pm 0.008}$	$0.575_{\pm 0.006}$	$0.562_{\pm 0.002}$
	Forget TR	$0.497_{\pm 0.056}$	$0.546_{\pm 0.043}$	$0.541_{\pm 0.023}$	$\mathbf{0.589}_{\pm 0.006}$
	KS Forget	$0.588_{\pm 0.014}$	$0.607_{\pm 0.015}$	$0.599_{\pm 0.013}$	$\mathbf{0.613}_{\pm 0.007}$
Gemma3-12B	Model Utility	$0.532_{\pm 0.039}$	$0.549_{\pm 0.010}$	$0.567_{\pm 0.040}$	$\mathbf{0.589}_{\pm 0.042}$
	Forget TR	$0.289_{\pm 0.031}$	$0.311_{\pm 0.014}$	$0.340_{\pm 0.004}$	$\mathbf{0.360}_{\pm 0.008}$
	KS Forget	$0.449_{\pm 0.012}$	$0.468_{\pm 0.139}$	$\mathbf{0.513}_{\pm 0.004}$	$0.490_{\pm 0.007}$

Result. Table 2 shows that WAG-targeted GradDiff consistently improves upon standard GradDiff. Averaged across both LLMs and all three WAG criteria, targeted unlearning improves *Forget Truth Ratio*, *KS Forget*, and *Model Utility* by 14.5%, 6.2%, and 4.0%, respectively, relative to GradDiff. The largest gains in *Forget Truth Ratio* are achieved by |WAG|, with improvements of 18.51% on LLaMA3.2-3B and 24.57% on Gemma3-12B. Notably, on Gemma3-12B, |WAG| improves all three metrics simultaneously.

Beyond unlearning performance, WAG-targeted GradDiff also reduces unlearning runtime cost. As shown in Figure 4, it is 1.2× and 2.6× faster than standard GradDiff on LLaMA3.2-3B and Gemma3-12B, respectively. Our results demonstrate that WAG enables more *effective* and *efficient* unlearning.

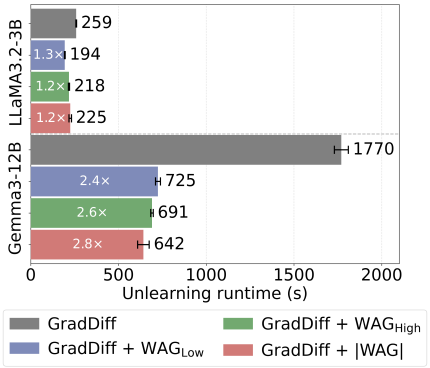


Figure 4: Unlearning runtime in seconds (3 runs) for methods. WAG-based unlearning is ∼1.9× faster than GradDiff on average.

4.3 Application C: WAG for Mixed Precision Quantization

We apply WAG to mixed-precision quantization to identify important weight groups to retain in high precision while quantizing the rest. Following the SliM-LLM protocol (Huang et al., 2025), we score each group by the average |WAG| of its parameters, with higher scores indicating greater importance for preservation. We compare this criterion with random selection, parameter magnitude, and Optimal Brain Surgeon (OBS) (Hassibi & Stork, 1992) saliency from SparseGPT (Frantar & Alistarh, 2023). We quantize models trained on coding tasks using the SliM-LLM approach, preserving the top-κ ($\kappa \in 5\%, 10\%$) parameter groups in each weight matrix to FP16 and quantize the remaining to 4-bit. We evaluate *HumanEval* and *HumanEval+* pass@1 on LLaMA3.2-3B and Gemma3-12B.

Result. From Table 3 we observe that the WAG metric-guided criteria improve over random group selection across both models and budgets. On average, |WAG| alone surpasses random, magnitude, and OBS saliency on both LLMs. Importantly, augmenting the strong OBS baseline with WAG (OBS+|WAG|) yields a further overall performance improvement of 0.64% in comparison to |WAG|.

4.4 Application D: WAG for Knowledge Editing

We finally apply WAG to knowledge editing under the locate-then-edit (Wang et al., 2024b) paradigm, where the goal is to inject a new fact while preserving unrelated knowledge. A central challenge is identifying where to edit: targeting the wrong layers can either fail to incorporate the new fact or induce unrelated behavior (Basani & Chhabra, 2026). We therefore target layers containing parameters most sensitive to the specific fact being edited, as identified by WAG. We evaluate R-ROME (Gupta et al., 2024) on *ZSRE* (Levy et al., 2017), *WikiCounterFact* (Zhang et al., 2024), *WikiBio* (Hartvigsen et al., 2023), and *WikiRecent* (Cohen et al., 2024), following the proxy-set protocol of Datta et al.

Table 3: Mixed-precision quantization results using SliM-LLM with the top $n\%$ ($5\%/10\%$) parameters preserved at high precision FP16, and rest quantized to low precision 4-bit. Evaluation is conducted on *HumanEval* / *HumanEval+* pass@1 (%) over 3 seeds. Compared to baselines, WAG attains improved performance (top *Pass@1* accuracy (%) in **green**; second best in green).

Models & Datasets	Unquantized %	Random	Magnitude	OBS	|WAG|	OBS+|WAG|
LLaMA3.2-3B						
HumanEval	5%	$30.49_{\pm 2.464}$	$30.08_{\pm 1.939}$	$35.98_{\pm 0.931}$	$35.98_{\pm 2.199}$	$38.01_{\pm 0.407}$
	10%	$30.69_{\pm 2.344}$	$31.71_{\pm 1.863}$	$36.18_{\pm 0.203}$	$37.80_{\pm 1.408}$	$37.20_{\pm 0.704}$
HumanEval+	5%	$27.24_{\pm 2.666}$	$27.24_{\pm 1.664}$	$33.33_{\pm 1.236}$	$32.93_{\pm 1.760}$	$35.37_{\pm 0.352}$
	10%	$28.46_{\pm 2.473}$	$29.67_{\pm 1.466}$	$33.54_{\pm 0.610}$	$34.15_{\pm 1.269}$	$33.94_{\pm 0.538}$
Average		$29.22_{\pm 2.364}$	$29.67_{\pm 1.661}$	$34.76_{\pm 0.535}$	$35.21_{\pm 1.630}$	**36.13**$_{\pm 0.403}$
Gemma3-12B						
HumanEval	5%	$54.88_{\pm 0.610}$	$55.89_{\pm 0.538}$	$56.10_{\pm 1.760}$	$55.28_{\pm 2.473}$	$55.89_{\pm 0.203}$
	10%	$56.10_{\pm 0.610}$	$55.49_{\pm 1.220}$	$54.88_{\pm 1.760}$	$56.50_{\pm 1.423}$	$57.32_{\pm 0.610}$
HumanEval+	5%	$48.37_{\pm 0.733}$	$50.41_{\pm 0.813}$	$49.59_{\pm 1.016}$	$51.02_{\pm 2.473}$	$49.80_{\pm 0.538}$
	10%	$49.80_{\pm 0.203}$	$48.98_{\pm 1.132}$	$49.19_{\pm 1.587}$	$50.20_{\pm 2.063}$	$51.42_{\pm 0.203}$
Average		$52.29_{\pm 0.152}$	$52.69_{\pm 0.366}$	$52.44_{\pm 1.504}$	$53.25_{\pm 1.252}$	**53.61**$_{\pm 0.221}$

Table 4: R-ROME Knowledge editing performance comparison of |WAG| and CMA on four datasets: *ZSRE*, *WikiCounterFact*, *WikiBio*, and *WikiRecent*. Different datasets support different performance metrics, where *Rewrite Accuracy* → RwA, *Rephrase Accuracy* → RpA, *Locality* → LOC, *Portability* → PRT, *Fluency* → FLC, and *Overall* → OV. Best overall performance is denoted as **green**.

ZSRE

Model	Selection	RwA	RpA	LOC	PRT	OV
LLaMA3.2-3B	CMA	0.9946	0.9595	0.9420	0.5383	0.8586
	|WAG|	0.9906	0.9483	0.9624	0.5433	**0.8612**
Gemma3-12B	CMA	0.8581	0.7041	0.9421	0.4989	0.7508
	|WAG|	0.9575	0.8150	0.9646	0.5028	**0.8100**

WikiCounterFact

Model	Selection	RwA	RpA	LOC	PRT	OV
LLaMA3.2-3B	CMA	0.9954	0.3570	0.5583	0.8811	0.7019
	|WAG|	0.9550	0.3807	0.5736	0.8936	**0.7038**
Gemma3-12B	CMA	0.8276	0.5265	0.4473	0.4871	0.5757
	|WAG|	0.9290	0.5746	0.4771	0.4871	**0.6209**

WikiBio

Model	Selection	RwA	LOC	FLC	OV
LLaMA3.2-3B	CMA	0.8904	0.6625	0.8889	**0.8139**
	|WAG|	0.7898	0.7351	0.8853	0.8034
Gemma3-12B	CMA	0.6850	0.6566	0.6703	0.6706
	|WAG|	0.8153	0.8062	0.6670	**0.7628**

WikiRecent

Model	Selection	RwA	LOC	PRT	FLC	OV
LLaMA3.2-3B	CMA	0.9844	0.4976	0.5362	0.8994	**0.7294**
	|WAG|	0.9844	0.4976	0.5362	0.8994	**0.7294**
Gemma3-12B	CMA	0.9710	0.5796	0.5216	0.5020	0.6436
	|WAG|	0.9775	0.5779	0.5327	0.5013	**0.6474**

(2026). We use |WAG| to rank parameters by sensitivity to the edit and select the MLP layer containing the highest concentration of such parameters. We compare our approach with the standard Causal Mediation Analysis (CMA) (Geva et al., 2023) layer selection for knowledge editing. Selection is performed using a small proxy set, while results are evaluated on an unseen test set.

Result. Table 4 shows that |WAG| improves the *Overall* score by 3.38% relative to CMA, averaged across all models and datasets. The gains are most pronounced for *Locality* and *Portability*, where |WAG| matches or exceeds CMA in every setting. In sum, |WAG| serves as an effective layer-selection mechanism for locate-then-edit knowledge editing.

5 Conclusion

In this work, we studied the problem of identifying parameters in LLMs that most influence model behavior. While existing approaches estimate parameter importance/sensitivity using their magnitudes, first-order gradients, and/or expensive second-order approximations, they have not explored the underlying relationship between weights and gradients. In contrast, we found that this unexplored relationship, which we term Weight-Adjusted Gradients (WAG) identifies a tiny subset of parameters that have a disproportionate impact on model performance, leading to model collapse. We provided theoretical interpretations for WAG and comprehensive empirical analyses to study this novel failure mode. Further, through extensive experiments across several settings, we demonstrated the effectiveness of WAG as a tool to improve performance across diverse parameter-centric LLM applications.

AI USE STATEMENT

We used generative AI tools solely to assist with proofreading, such as correcting spelling and grammar in the manuscript.

ETHICS STATEMENT

Our work seeks to utilize parameter importance to better understand and improve LLM interpretability and robustness, and as such does not possess any negative ethical consequences. Moreover, as we show in the results obtained, our proposed parameter influence estimation method (WAG) can actually increase performance for LLM tasks/applications of societal consequence, such as unlearning, knowledge editing, and compression via mixed-precision quantization.

REPRODUCIBILITY STATEMENT

Full experimental setup and implementation details for all experiments are provided in Appendix E; evaluation metrics are described in Appendix C; and hardware, seeds, and other reproducibility configurations are listed in Appendix F.

REFERENCES

Alexandra Abbas, Celia Waggoner, and Justin Olive. Developing and Maintaining an Open-Source Repository of AI Evaluations: Challenges and Insights. In *arXiv preprint arXiv:2507.06893*, 2025.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. In *European Conference on Computer Vision*, 2018.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Systematic Outliers in Large Language Models. In *International Conference on Learning Representations*, 2025.

Hadi Askari, Shivanshu Gupta, Fei Wang, Anshuman Chhabra, and Muhao Chen. LayerIF: Estimating Layer Quality for Large Language Models using Influence Functions. In *Advances in Neural Information Processing Systems*, 2025.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Advik Raj Basani and Anshuman Chhabra. Exposing the Illusion of Erasure in Knowledge Editing for LLMs. *arXiv preprint arXiv:2606.23276*, 2026.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

Anshuman Chhabra, Bo Li, Jian Chen, Prasant Mohapatra, and Hongfu Liu. Outlier Gradient Analysis: Efficiently Identifying Detrimental Training Samples for Deep Learning Models. In *International Conference on Machine Learning*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the Ripple Effects of Knowledge Editing in Language Models. *Transactions of the Association for Computational Linguistics*, 2024.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. Beyond Size: How Gradients Shape Pruning Decisions in Large Language Models. *arXiv preprint arXiv:2311.04902*, 2023.

Shrestha Datta, Hongfu Liu, and Anshuman Chhabra. Golden Layers and Where to Find Them: Improved Knowledge Editing for Large Language Models via Layer Gradient Analysis. *arXiv preprint arXiv:2602.20207*, 2026.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp Minima Can Generalize For Deep Nets. In *International Conference on Machine Learning*, 2017.

William B. Dolan and Chris Brockett. Automatically Constructing a Corpus of Sentential Paraphrases. In *International Workshop on Paraphrasing*, 2005.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-Jussà. A Primer on the Inner Workings of Transformer-based Language Models. *arXiv preprint arXiv:2405.00208*, 2024.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-Aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*, 2021.

Elias Frantar and Dan Alistarh. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. In *International Conference on Machine Learning*, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pretrained Transformers. In *International Conference on Learning Representations*, 2023.

Trevor Gale, Erich Elsen, and Sara Hooker. The State of Sparsity in Deep Neural Networks. *arXiv preprint arXiv:1902.09574*, 2019.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and V. S. Subrahmanian. MoLA: MoE LoRA with Layer-wise Expert Allocation. In *Findings of the North American Chapter of the Association for Computational Linguistics*, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The Language Model Evaluation Harness, 2024.

Rui Gao and Weiwei Liu. Defying Catastrophic Forgetting via Influence Function. *Artificial Intelligence*, 2025.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting Recall of Factual Associations in Auto-Regressive Language Models. In *Empirical Methods in Natural Language Processing*, 2023.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A Standard for Language Model Evaluations. In *Findings of the North American Chapter of the Association for Computational Linguistics*, 2025.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. Rebuilding ROME: Resolving Model Collapse during Sequential Model Editing. In *Empirical Methods in Natural Language Processing*, 2024.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Networks. In *Advances in Neural Information Processing Systems*, 2015.

Stephen Jose Hanson and Lorien Y. Pratt. Comparing Biases for Minimal Network Construction with Back-Propagation. In *Advances in Neural Information Processing Systems*, 1988.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with Grace: Lifelong Model Editing with Discrete K-Value Adaptors. In *Advances in Neural Information Processing Systems*, 2023.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal Brain Surgeon. In *Advances in Neural Information Processing Systems*, 1992.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training Compute-Optimal Large Language Models. In *Advances in Neural Information Processing Systems*, 2022.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Xianglong Liu, Luca Benini, Michele Magno, and Xiaojuan Qi. SliM-LLM: Salience-driven mixed-precision quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 2017.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, 1989.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. In *International Conference on Learning Representations*, 2019.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-Shot Relation Extraction via Reading Comprehension. In *Conference on Computational Natural Language Learning*, 2017.

Bo Liu, Qiang Liu, and Peter Stone. Continual Learning and Private Unlearning. In *Conference on Lifelong Learning Agents*, 2022.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems*, 2023.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. AlphaPruning: Using Heavy-Tailed Self Regularization Theory for Improved Layer-wise Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering. In *Advances in Neural Information Processing Systems*, 2022.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary Chase Lipton, and J. Zico Kolter. TOFU: A Task of Fictitious Unlearning for LLMs. In *First Conference on Language Modeling*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large Language Models: A Survey. *arXiv preprint arXiv:2402.06196*, 2024.

Kshitij Mishra, Tamer Soliman, Anil Ramakrishna, Aram Galstyan, and Anoop Kumar. Correcting Language Model Outputs by Editing Salient Layers. In *Findings of the Association for Computational Linguistics*, 2024.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *International Conference on Learning Representations*, 2017.

Ocean Monjur, Shahriar Kabir Nahin, and Anshuman Chhabra. Revisiting the Effectiveness of LLM Pruning for Test-Time Scaling. In *Findings of Empirical Methods in Natural Language Processing*, 2026.

OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. AlphaLoRA: Assigning LoRA Experts Based on Layer Training Quality. In *Empirical Methods in Natural Language Processing*, 2024.

Waleed Reda, Abhinav Jangda, and Krishna Chintalapudi. How Many Parameters Does Your Task Really Need? Task Specific Pruning with LLM-Sieve. *arXiv preprint arXiv:2505.18350*, 2025.

Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. PB-LLM: Partially binarized large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Boyu Shi, Chang Liu, ChuanBao Gao, Xu Yang, and Xin Geng. Understanding Performance Collapse in Layer-Pruned Large Language Models via Decision Representation Transitions. *arXiv preprint arXiv:2605.07271*, 2026.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The Curse of Recursion: Training on Generated Data Makes Models Forget. *arXiv preprint arXiv:2305.17493*, 2023.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models. In *International Conference on Learning Representations*, 2024.

Mingyuan Sun, Zheng Fang, Jiaxu Wang, Junjie Jiang, Delei Kong, Chenming Hu, Yuetong Fang, and Renjing Xu. Optimal Brain Apoptosis. In *International Conference on Learning Representations*, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

Dmytro Vitel and Anshuman Chhabra. First is Not Really Better Than Last: Evaluating Layer Choice and Aggregation Strategies in Language Model Data Influence Estimation. In *International Conference on Learning Representations*, 2026.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, 2018.

Lecheng Wang, Xianjie Shi, Ge Li, Jia Li, Yihong Dong, Xuanming Zhang, Wenpin Jiao, and Hong Mei. Why Language Models Collapse when Trained on Recursively Generated Text. *arXiv preprint arXiv:2412*, 2024a.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, et al. EasyEdit: An Easy-to-Use Knowledge Editing Framework for Large Language Models. In *Association for Computational Linguistics*, 2024b.

Hanqi Xiao, Yi-Lin Sung, Elias Stengel-Eskin, and Mohit Bansal. Task-Circuit Quantization: Leveraging Knowledge Localization and Interpretability for Compression. In *Second Conference on Language Modeling*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M. Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, et al. Wanda++: Pruning Large Language Models via Regional Gradients. In *Findings of the Association for Computational Linguistics*, 2025b.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing Large Language Models: Problems, Methods, and Opportunities. In *Empirical Methods in Natural Language Processing*, 2023.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier Weighed Layerwise Sparsity (OWL): A Missing Secret Sauce for Pruning LLMs to High Sparsity. In *International Conference on Machine Learning*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, 2017.

Mingyuan Zhang, Yue Bai, Huan Wang, Yizhou Wang, Qihua Dong, Yitian Zhang, and Yun Fu. Boosting Large Language Models with Mask Fine-Tuning. *arXiv preprint arXiv:2503.22764*, 2025.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A Comprehensive Study of Knowledge Editing for Large Language Models. *arXiv preprint arXiv:2401.01286*, 2024.

Max Zimmer, Christophe Roux, Moritz Wagner, Deborah Hendrych, and Sebastian Pokutta. SparseSwaps: Tractable LLM Pruning Mask Refinement at Scale. *arXiv preprint arXiv:2512.10922*, 2025.

APPENDIX

Here, we provide detailed related work in Appendix A. Next, we provide the proofs of the theoretical analysis in Appendix B. In Appendix C, we provide details of the performance metrics used in the experiments. Appendices D.2 and D.1 provide the qualitative generation examples and additional empirical analysis referenced in Section 3.3. Moreover, Appendix E discusses the implementation details of the experiments. Finally, we provide configuration details for reproducibility of the experiments in Appendix F.

A RELATED WORKS

In this section, we review the literature on three aspects. We will discuss how parameter importance has been quantified. Next, we examine how these measures are operationalized as parameter influence estimates in downstream applications. Finally, we explore failure modes in LLMs, specifically model collapse, and observe that existing works do not examine collapse induced by masking the very parameters deemed important.

General Parameter Importance. Historically, numerous studies have investigated how to quantify parameter importance. Existing approaches primarily characterize parameter importance from three perspectives: (1) magnitude-based importance, (2) gradient-based sensitivity, and (3) curvature-based measures. The most naive magnitude-based method is utilizing parameter values themselves, under the intuition that larger weights exert a greater direct influence on the transformation of input signals and therefore may be more important (Hanson & Pratt, 1988; Gale et al., 2019). To obtain a more informative estimate of importance, subsequent works incorporate activation magnitudes, which capture the effective contribution of a parameter to the downstream computation for a given input (Han et al., 2015; Sun et al., 2024). Beyond magnitude-based criteria, parameter importance has also been estimated through the sensitivity of the loss function. For example, approximation of the change in loss resulting from parameter removal using a first-order Taylor expansion (Molchanov et al., 2017). Curvature-based approaches further account for second-order information. Optimal Brain Damage estimates the increase in loss caused by removing a parameter through a second-order Taylor expansion combined with a diagonal approximation of the Hessian (LeCun et al., 1989). In contrast, Optimal Brain Surgeon provides a more accurate second-order estimate by utilizing the full Hessian without imposing a diagonal approximation (Hassibi & Stork, 1992). More recently, Optimal Brain Apoptosis extends this line of work by introducing a computationally more efficient framework while similarly avoiding the diagonal Hessian assumption (Sun et al., 2025). Recent quantization methods such as GPTQ (Frantar et al., 2023) implicitly encode parameter importance through a second-order quadratic optimization objective. In these methods, a parameter's importance is reflected by how its perturbation affects the loss under the local curvature of the optimization landscape.

Applications of Estimating Parameter Influence. The parameter importance estimates are widely operationalized in pruning methods, which remove parameters or structured groups based on their estimated contribution to model performance. Recent approaches for LLMs include magnitude-based methods like Wanda (Sun et al., 2024) which leverage activation-scaled magnitude scores to directly rank weights for removal. On the other hand, magnitude alone also serves as an importance metric where lower magnitude parameters are pruned (Han et al., 2015). More recent work further explores gradient-based criteria and mask refinement strategies to improve pruning decisions under limited calibration data (Das et al., 2023; Yang et al., 2025b; Zimmer et al., 2025). Recent work has also shown that Wanda pruning can improve test-time reasoning performance in LLMs, even in comparison to the base unpruned model (Monjur et al., 2026).

These principles have been extended to layer-wise sparsity allocation, where methods estimate the relative contribution of transformer blocks to guide non-uniform pruning ratios across layers. AlphaPruning (Lu et al., 2024) allocates sparsity using spectral characteristics of layer weights to identify well-conditioned layers, while OWL (Yin et al., 2024) leverages activation outlier statistics to estimate layer sensitivity and preserve critical transformer blocks. On the other hand, LayerIF (Askari et al., 2025) takes a data-driven approach, computing layer-wise influence scores by isolating layer gradients and Hessians while keeping the validation and training data constant.

Beyond pruning and sparsity allocation, influence-based (Chhabra et al., 2025) layer quality estimates have been applied to continual learning, where the central challenge is preventing catastrophic

forgetting of prior tasks when learning new ones. Regularization-based methods, Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Synaptic Intelligence (SI) (Zenke et al., 2017), and Memory Aware Synapses (MAS) (Aljundi et al., 2018) address this by penalizing changes to parameters deemed important for previous tasks. Recent work (Gao & Liu, 2025) provides a unified theoretical account of this family by formally defining parameter importance via influence functions.

Influence-based layer quality estimation has also been applied to expert allocation in Mixture-of-Experts (MoE) architectures. Uniform allocation of LoRA experts across transformer layers has been questioned by recent work showing that expert redundancy and routing overfitting degrade performance (Gao et al., 2025). AlphaLoRA (Qing et al., 2024) addresses this using spectral layer quality metrics to inform allocation. LayerIF (Askari et al., 2025) instead derives task-specific allocation budgets from layer-wise influence scores, dynamically varying expert counts across layers.

Parameter influence is also used in the task of mixed-precision quantization, where parameter groups with the most saliency are preserved in high precision and the others in low precision. Recent studies, PB-LLM (Shang et al., 2024) and SliM-LLM (Huang et al., 2025), employ magnitude- and Hessian-based criteria as a measure of the saliency of the groups.

Moreover, gradient-based methods like Salient Layers Editing Model (Mishra et al., 2024) and Gradient Layer Analysis (Datta et al., 2026) are used for important editing layer identification in knowledge editing.

Identifying Failure Modes in LLMs. Failure due to model collapse in LLMs broadly manifests in two forms: collapse arising from corrupted training data, and collapse arising from architectural modification. Recent work has characterized the phenomenon of model collapse arising from recursive training on synthetic data (Shumailov et al., 2023). Critically, it was shown that access to original human-generated data is essential to prevent collapse. Recent work (Wang et al., 2024a) shows that recursive training on self-generated text introduces non-negative errors that accumulate across generations, leading to irreversible model collapse and deviation from the original data distribution and eventual degradation in both fluency and downstream task performance.

At the architectural level, a recent study (Shi et al., 2026) found that collapse under layer pruning is a threshold phenomenon: performance remains stable until pruning reaches the *Silent Phase*, the early-to-middle layers responsible for building representational scaffolding, after which collapse is abrupt and irreversible even under supervised fine-tuning. Recent observations suggest that the collapse threshold is determined not by total parameter count but by a sparse, irregularly distributed set of critical parameters (Reda et al., 2025). However, these works do not characterize collapse at the level of individual parameters or identify the minimum number of scattered weights whose targeted removal suffices to induce it.

B Proofs of Theoretical Analysis

We restate each theorem from Subsection 3.2 and provide its full proof. Each restated statement carries the same numbering as in the main text. Throughout, $\ell(z;\theta)$ denotes the per-example loss of a model with parameter vector $\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$ evaluated at example z, and the WAG of coordinate i is defined as

$$\mathrm{WAG}_i \;=\; -\,\theta_i \frac{\partial \ell(z;\theta)}{\partial \theta_i}.$$

All perturbations are evaluated to first order, with higher-order terms collected in $\mathcal{O}(\cdot)$.

B.1 Proof of Theorem 3.1

Theorem 3.1 (WAG as Log-Parameter Gradient)**.** *Assume $\theta_i \neq 0$ and define the logarithmic parameterization:*

$$u_i = \log|\theta_i|.$$

Then the gradient of the loss with respect to u_i satisfies:

$$\frac{\partial \ell(z;\theta)}{\partial u_i} = \theta_i \frac{\partial \ell(z;\theta)}{\partial \theta_i}.$$

Equivalently:

$$\mathrm{WAG}_i = -\frac{\partial \ell(z;\theta)}{\partial u_i}.$$

Proof. Since $u_i = \log|\theta_i|$, we have $\theta_i = \mathrm{sign}(\theta_i)e^{u_i}$. Applying the chain rule gives:

$$\frac{\partial \ell}{\partial u_i} = \frac{\partial \ell}{\partial \theta_i}\frac{\partial \theta_i}{\partial u_i}.$$

Moreover, since $\frac{\partial \theta_i}{\partial u_i} = \theta_i$, it follows that:

$$\frac{\partial \ell}{\partial u_i} = \theta_i \frac{\partial \ell}{\partial \theta_i}.$$

And using the definition of WAG yields:

$$\mathrm{WAG}_i = -\frac{\partial \ell}{\partial u_i}.$$ □

B.2 Proof of Theorem 3.2

Theorem 3.2 (First-Order Multiplicative Sensitivity)**.** *Let $\theta' \in \mathbb{R}^d$ be obtained through multiplicative perturbations of the type:*

$$\theta'_i = \theta_i(1+\epsilon_i),$$

where ϵ_i are sufficiently small. Then the corresponding first-order change in loss satisfies:

$$\ell(z;\theta') - \ell(z;\theta) = -\sum_{i=1}^{d} \epsilon_i \mathrm{WAG}_i + \mathcal{O}(\|\epsilon\|^2).$$

Proof. Using first-order Taylor expansion around θ,

$$\ell(z;\theta') = \ell(z;\theta) + \sum_{i=1}^{d} \frac{\partial \ell(z;\theta)}{\partial \theta_i}(\theta'_i - \theta_i) + \mathcal{O}(\|\theta' - \theta\|^2).$$

As $\theta'_i - \theta_i = \epsilon_i \cdot \theta_i$, we obtain:

$$\ell(z;\theta') - \ell(z;\theta) = \sum_{i=1}^{d} \epsilon_i \theta_i \frac{\partial \ell(z;\theta)}{\partial \theta_i} + \mathcal{O}(\|\epsilon\|^2).$$

Substituting the definition of WAG ties up the proof:

$$\ell(z;\theta') - \ell(z;\theta) = -\sum_{i=1}^{d} \epsilon_i \mathrm{WAG}_i + \mathcal{O}(\|\epsilon\|^2).$$ □

B.3 Proof of Theorem 3.3

We show that the WAG vector identifies the dominant directions governing local scale stability. In particular, the magnitude of the WAG vector determines the largest possible first-order increase in loss achievable through multiplicative perturbations. Consequently, parameters with large absolute WAG scores correspond to the coordinates that contribute most strongly to maintaining local scale balance.

Modern transformer-based LLMs contain billions of parameters whose magnitudes vary substantially across layers, attention blocks, and MLP blocks (An et al., 2025). Despite this heterogeneity, trained models exhibit remarkably stable behavior. A key reason is that a network's behavior depends on the relative balance of these scales, so the loss responds to how parameter scales are coordinated across the model (Dinh et al., 2017; Ba et al., 2016). This observation suggests that optimization settles into a coordinated balance of parameter scales. Prior measures probe this geometry by perturbing weights additively (Foret et al., 2021); in contrast, WAG measures the sensitivity of the loss to relative changes in these scales, making it a natural tool for studying this *balance*. Therefore, we next show that the WAG vector identifies the dominant directions governing local scale stability.

Theorem 3.3 (WAG Characterizes the Most Sensitive Log-Scale Direction)**.** *Let* $w = (\mathrm{WAG}_1, \ldots, \mathrm{WAG}_d) \in \mathbb{R}^d$ *denote the WAG vector, and consider multiplicative parameter perturbations:* $\theta_i' = \theta_i(1 + \epsilon_i)$*, with perturbation vector:* $\epsilon = (\epsilon_1, \ldots, \epsilon_d)$*. Then, among all multiplicative perturbations satisfying:* $\|\epsilon\|_2 = r$*, the maximal first-order increase in loss is* $r\|w\|_2$*, and is attained uniquely by perturbations parallel to* $-w$*. Equivalently, the WAG vector defines the direction of greatest local sensitivity in logarithmic parameter space.*

Proof. By Theorem 3.2,

$$\ell(z; \theta') - \ell(z; \theta) = -\langle \epsilon, w \rangle + \mathcal{O}(\|\epsilon\|_2^2).$$

Fix a perturbation budget $\|\epsilon\|_2 = r$. Applying the Cauchy–Schwarz inequality yields:

$$-\langle \epsilon, w \rangle \leq \|\epsilon\|_2 \|w\|_2 = r\|w\|_2.$$

Equality holds precisely when ϵ is anti-parallel to w, namely:

$$\epsilon^\star = -\frac{r\, w}{\|w\|_2}.$$

Substituting this perturbation into the first-order expansion gives us:

$$\ell(z; \theta') - \ell(z; \theta) = r\|w\|_2 + \mathcal{O}(r^2).$$

Therefore, no multiplicative perturbation with the same norm can produce a larger first-order increase in loss. Since Theorem 3.1 showed that WAG is the gradient in logarithmic parameter coordinates, it follows that the WAG vector defines the direction of greatest local sensitivity in log-parameter space. □

Theorem 3.3 provides a geometric interpretation of WAG. In log-parameter space, the WAG vector identifies the unique direction along which multiplicative perturbations produce the largest first-order increase in loss. The norm $\|w\|_2$ quantifies the magnitude of this worst-case sensitivity, while the coordinates with the largest absolute WAG values contribute most strongly to it.

Combined with Theorem 3.1, this result suggests that trained transformers (i.e., LLMs) can be viewed as *scale-balanced* systems. Optimization drives the network toward configurations in which many parameter scales cooperate to maintain low loss. The WAG vector captures the residual sensitivity of this equilibrium and identifies the directions along which the balance remains most fragile. An immediate consequence is that not all parameters contribute equally to maintaining this equilibrium. Parameters with large absolute WAG scores dominate the most sensitive log-scale direction and therefore account for a disproportionate fraction of the model's local scale stability. Perturbing or masking these parameters removes the coordinates that contribute most strongly to preserving the learned *balance* of *scales*.

This perspective provides a theoretical explanation for the empirical phenomena and observations we provide later in the paper. When parameters are ranked according to $|\mathrm{WAG}_i|$, a subset of coordinates often emerges whose modification causes significant degradation in model performance. As described via Theorem 3.3, these parameters correspond precisely to the coordinates that dominate the model's most sensitive log-scale direction. In sum, masking them therefore removes a substantial portion of the network's scale-balancing structure, which results in significant performance degradation and even model collapse.

C Performance Metrics

Here we give the details of the metrics used to evaluate the results of the experiments involving the applications of WAG presented in this paper.

C.1 Empirical Analysis

We adopt the evaluation protocols of OLMES (Gu et al., 2025), LM Evaluation Harness (Gao et al., 2024), and InspectAI (Abbas et al., 2025) for coding, mathematical, and commonsense reasoning tasks, respectively for our experiments.

***Pass@1*:** *Pass@1* is used to evaluate code generation performance on the *HumanEval* and *HumanEval+* benchmarks. It measures the proportion of tasks for which the model's greedy generated solution passes all corresponding unit tests.

***Exact Match*:** *Exact Match* is used to evaluate mathematical reasoning performance on the *GSM8K* benchmark and commonsense reasoning performance on the *HellaSwag* benchmark. It measures the proportion of examples for which the model's final generated answer exactly matches the ground-truth answer.

C.2 Application A: WAG for Mixture-of-Experts Allocation

We evaluate the expert allocations following the LayerIF (Askari et al., 2025) evaluation protocol.

***Zero-shot Accuracy*:** *Zero-shot Accuracy* measures classification and question-answering performance on *MRPC*, *CoLA*, and *ScienceQA*, computed as the proportion of examples where the label extracted from the model's free-form response via pattern matching exactly matches the ground truth.

C.3 Application B: WAG for Targeted Unlearning

We adopt the evaluation of the unlearning task from the *TOFU* benchmark (Maini et al., 2024). Each QA example is probed with its ground-truth answer, a *paraphrased* (semantically equivalent) answer, and a set of *perturbed* (factually wrong) answers. For a target answer with length-normalized negative log-likelihood (per-token cross-entropy) ℓ under the model, its length-normalized probability is $p = \exp(-\ell)$. The benchmark partitions data into a *Forget* set, a *Retain* set, and two general-knowledge probes (*Real Authors*, *Real World*).

***Model Utility*:** The overall linguistic and retention capability preserved after unlearning, computed as the harmonic mean of nine non-forget metrics: *Probability, ROUGE, Truth Ratio* for each *Retain, Real Authors, Real World* type of knowledge probe. For each task, *Probability* is the mean ground-truth answer probability (normalized against the perturbed answers for the Real Authors / Real World probes); *ROUGE* is the ROUGE-L recall (with stemming) between the model's greedily generated answer and the ground truth; and *Truth Ratio* rewards preferring the correct answer over perturbations.

***Forget Truth Ratio (TR)*:** Let $p_{\text{para}} = \exp(-\ell_{\text{para}})$ be the probability of the paraphrased answer and $\bar{p}_{\text{pert}}$ the mean probability over the perturbed answers, and define $R = \bar{p}_{\text{pert}}/p_{\text{para}}$. On the Forget set we report $\frac{1}{N}\sum \min(R,\, 1/R) \in (0, 1]$. A higher value means the perturbed and correct answers are about equally (un)likely, indicating the targeted fact has been forgotten.

***KS Forget*:** We form the empirical distribution of per-example forget truth ratios for the unlearned model with respect to the FFT reference model, and run a two-sample Kolmogorov–Smirnov test; *KS Forget* is the resulting KS statistic D (the supremum distance between the two empirical Cumulative Distribution Functions (CDFs)). In our setup, the reference is the fine-tuned model *before* unlearning (FFT), so a larger D indicates that the unlearned model's forget-set behavior has diverged further from the pre-unlearning model i.e., stronger forgetting. As the FFT model itself is the reference model, the metric is not reported for the FFT model.

C.4 Application C: WAG for Mixed Precision Quantization

We employ the same performance metrics defined in Appendix C.1 for this application.

C.5 Application D: WAG for Knowledge Editing

We evaluate knowledge editing performance using five metrics provided by the EasyEdit framework (Wang et al., 2024b): *Rewrite Accuracy*, *Rephrase Accuracy*, *Locality*, *Portability*, and *Fluency*. Since not all datasets support every evaluation metric (Wang et al., 2024b; Zhang et al., 2024), results are reported only for the metrics applicable to each dataset. Higher values indicate better editing performance for all metrics. We additionally report an *Overall* score, computed as the average of the normalized metric values.

***Rewrite Accuracy*:** *Rewrite Accuracy* measures whether the edited knowledge is correctly retrieved when responding to the original query targeted by the edit.

***Rephrase Accuracy*:** *Rephrase Accuracy* evaluates whether the model can recall the edited knowledge when presented with semantically equivalent rephrasings of the original query.

***Locality*:** *Locality* measures the extent to which the edit preserves the model's behavior on unrelated queries, thereby quantifying unintended side effects introduced by the edit (Meng et al., 2022; Wang et al., 2024b; Yao et al., 2023; Zhang et al., 2024).

***Portability*:** *Portability* assesses the model's ability to transfer the edited knowledge to related downstream reasoning tasks (Yao et al., 2023).

***Fluency*:** *Fluency* evaluates the linguistic quality and naturalness of the generated responses.

D Empirical Analysis

In this section, we present the qualitative and quantitative analysis of model behavior due to masking WAG-identified parameters.

D.1 Quantitative Examples of WAG-Induced Collapse

Masking up to just $\leq 0.0002\%$ of $|\text{WAG}|$ identified parameters causes model collapse or the onset of model collapse across different tasks and models. However, masking the same number of parameters identified by weight or gradient doesn't cause consistent model collapse. Therefore, WAG is able to identify important parameters consistently.

D.2 Qualitative Examples of WAG-Induced Collapse

We provide the qualitative generation examples referenced in Section 3.3, showing verbatim model outputs before and after masking the WAG-identified parameters, indicating qualitative collapse after masking only 0.0001% of the parameters.

D.3 WAG selected parameter distribution across layers.

To examine the layer-wise distribution of the 0.0002% parameters selected by WAG, Figure 8 shows the selected parameters across layers for WAG_{Low}, WAG_{High}, and $|\text{WAG}|$ across models and datasets. Across all datasets and models, the selected parameters are predominantly concentrated in the final layer, while constituting only a very small fraction of its parameters.

D.4 WAG score distribution across layers.

To complement the parameter-selection view in Appendix D.3, Figure 9 shows the layer-wise *average* WAG score, computed separately over the positive elements (WAG_+), the negative elements (WAG_-), all elements (WAG), and the magnitude ($|\text{WAG}|$) of all parameters for each layer. Two observations hold across all models and datasets. First, the largest score magnitudes for WAG_+, WAG_-, and $|\text{WAG}|$ are concentrated in the early-to-mid layers. The concentration depth is dataset-dependent (mid layers for *HumanEval+* and *GSM8K*, early layers for *HellaSwag*) but, for a given dataset, sits at the same relative layer depth in LLaMA3.2-3B, Qwen3-4B, and Gemma3-12B. Second, the average WAG is close to zero in every layer: positive and negative contributions are evenly balanced within each layer.

E Experiment Setup and Implementation Details

In this section, we will discuss the experiment setup and the implementation details of the experiments conducted to demonstrate the practical applicability of WAG.

E.1 Empirical Analysis and common details

We consider all parameters of the linear projections in the attention and MLP blocks as candidate parameters. We compute a WAG score for each candidate parameter and rank them globally across

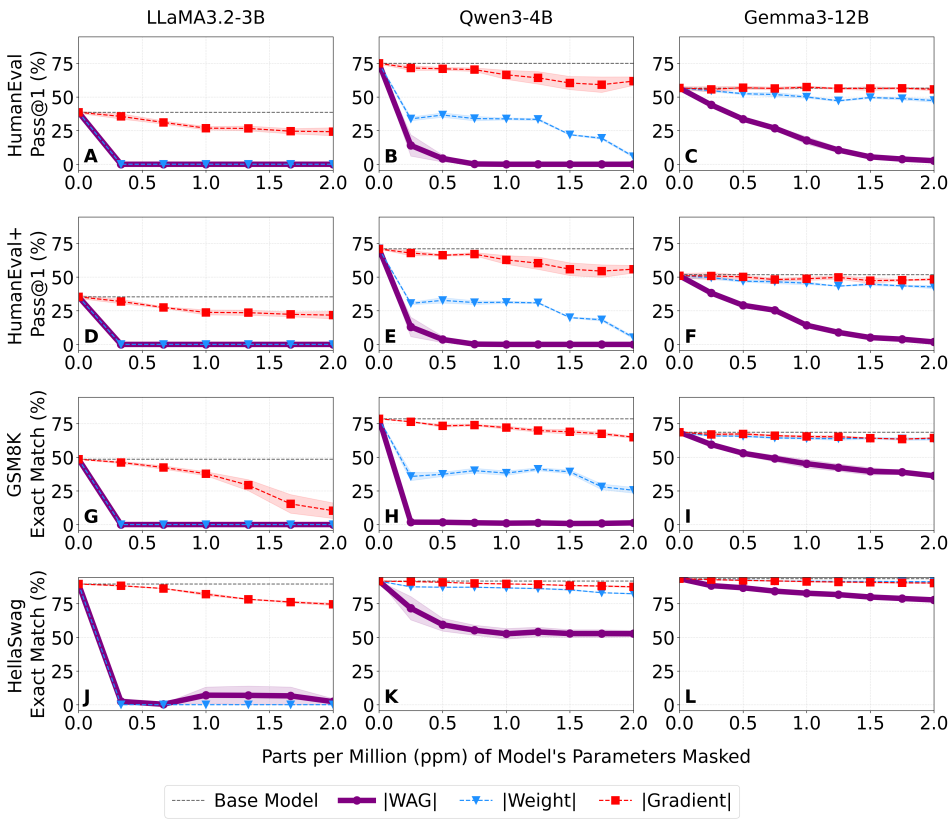


Figure 5: Illustration of model collapse caused by masking just ≤ 2 parts per million (i.e., $\leq 0.0002\%$) parameters across several models: LLaMA3.2-3B (A, D, G and J), Qwen3-4B (B, E, H and K), and Gemma3-12B (C, F, I and L). Each model is assessed on four datasets: *HumanEval* (A, B and C), *HumanEval+* (D, E and F), *GSM8K* (G, H and I), and *HellaSwag* (J, K, and L). Performance is reported as *Pass@1* and *exact match* accuracy as a function of the number of masked parameters. Parameters are selected based on having the largest-magnitude gradient $\rightarrow$ |Gradient|, weight $\rightarrow$ |Weight|, and WAG $\rightarrow$ |WAG| values. Masking the largest-magnitude WAG identified parameters leads to rapid model collapse with a comparatively tiny portion of masked parameters.

all layers and blocks. Given a masking budget k, we mask the k parameters based on the selection criteria. To compute WAG-based parameter importance, we compute the gradients of the loss with respect to the model parameters using the evaluation set of the corresponding task. We evaluate performance on all tasks using a held-out evaluation set. For the coding tasks, we train the model on coding-related datasets and evaluate it on the *HumanEval* (Chen et al., 2021) and *HumanEval+* (Liu et al., 2023) datasets and report the *pass@1* accuracy (Chen et al., 2021), as conducted in recent work (Zhang et al., 2025). For the math and commonsense reasoning tasks, we selected the train and test split of the *GSM8K* (Cobbe et al., 2021) and *HellaSwag* (Zellers et al., 2019) datasets for training and evaluation, respectively, and report the *exact match* accuracy.

E.2 Application A: WAG for Mixture-of-Experts Allocation

We first leverage WAG to determine the optimal allocation of a fixed budget of N parallel experts across layers. We compare our method against four baselines: AlphaLoRA (Qing et al., 2024) and three fixed MoLA configurations (Gao et al., 2025), triangle (2 4 6 8), rectangle (5 5 5 5), and hourglass (8 2 2 8). The total expert budget is set to $N = 140$ for LLaMA3.2-3B and $N = 240$ for Gemma3-12B, and the router selects at most the top-2 experts per layer. We evaluate the zero-shot performance of the models with the allocated experts on three datasets: *MRPC* (Dolan & Brockett, 2005), *CoLA* (Wang et al., 2018), and *ScienceQA* (Lu et al., 2022).

Our allocation principle is to assign more experts to layers whose parameters are highly loss-sensitive to perturbation. Since such layers have the greatest impact on model performance, they

Base Model

```
def mean_absolute_deviation(numbers: List[float]) -> float:
    """ For a given list of input numbers, calculate Mean Absolute
    Deviation
    ...
    mean = sum(numbers) / len(numbers)
    absolute_diffs = [abs(num - mean) for num in numbers]
    mad = sum(absolute_diffs) / len(absolute_diffs)
    return mad
```

Observation: *Base model returns the correct implementation as output.*

WAG_{Low}

```
def mean_absolute_deviation(numbers: List[float]) -> float:    """
"""
    """    """    """    """    """    """    """    """    """
"""    """
    """    """    """    """    """    """    """    """    """
"""    """
```

Observation: *Empty docstring token repeats to 1805 chars; no function body emitted.*

WAG_{High}

```
def mean_absolute_deviation(numbers: List[float]) -> float:
    """
    For a given list of input numbers, calculate Mean Absolute
    Deviation around the mean of this dataset.
    ...
    >>> mean_absolute_deviation([1.0, 2.0, 3.0, 4.0])
    1.0
    """
def mean_absolute_deviation(numbers: List[float]) -> float:
    """ ... same docstring ... """
def mean_absolute_deviation(numbers: List[float]) -> float:
    """ ... same docstring ... """
```

Observation: *Signature + docstring block re-emitted to 2004 chars; never reaches body.*

|WAG|

```
def mean_absolute_deviation(numbers: List[float]) -> float:
    """ For a given list of input numbers, calculate Mean Absolute
    Deviation around the mean of this dataset.
    Mean Absolute Deviation is the average absolute difference between
    each element and a centerpoint (mean in this case):
Mean Absolute Deviation
    is the average absolute difference between each element and a
    centerpoint (mean in this case):  Mean Absolute Deviation is the
    average absolute difference between each element and a centerpoint
    (mean in this case):
```

Observation: *This single sentence repeats to 2568 chars; no body emitted.*

Gradient / Weight representative

```
def mean_absolute_deviation(numbers: List[float]) -> float:
    """ For a given list of input numbers, calculate Mean Absolute
    Deviation
    ...
    mean = sum(numbers) / len(numbers)
    deviations = [abs(num - mean) for num in numbers]
    return sum(deviations) / len(deviations)
```

Observation: *All six gradient/weight-based masked models return the correct implementation as output.*

Figure 6: Qualitative example of model failure caused by masking 0.0001% of parameters identified by WAG. Verbatim Qwen3-4B generations on *HumanEval* at 4,000 masked parameters indicate that the WAG_{Low} variant emits only empty docstrings; WAG_{High} repeats the signature and docstring without a body; |WAG| loops on a single sentence. All six gradient- and weight-masked variants return the canonical solution.

should receive a larger share of experts to adequately capture their complexity. We therefore report the expert-allocation performance for sensitive layers identified by three WAG-based criteria: WAG_{Low}, WAG_{High}, and |WAG|. To assess how many important parameters each layer contains,

Prompt: *"While walking with his 3 younger siblings, Greg found $20 and split it equally. How much did each get?"*

Base Model

```
There were 1 + 3 = 4 of them who found the money.
Each got 20 / 4 = $5.    ####  5
```

Observation: *Correct answer.*

WAG_{Low}

```
Greg found $20 and decided to split it equally among his 3 siblings,
so each of them got $20 / 3 = $6 and 2/3.    ####  6
```

Observation: *Fluent, but omits Greg and divides by 3 instead of 4.*

WAG_{High}

```
To be fair to his 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
```

Observation: *Token "1" repeats to the length cap.*

|WAG|

```
GregGregGregGregGregGregGregGregGregGregGregGregGregGregGregGregGregGreg
GregGregGregGregGregGregGregGregGregGregGregGregGregGregGregGregGreg ...
```

Observation: *Token "Greg" repeats to 2048 chars; the extracted answer is the same loop.*

Gradient / Weight representative

```
There are 4 of them because 1 + 3 = 4.
Each of them gets $5 because 20 / 4 = 5.    ####  5
```

Observation: *All six gradient/weight variants recover the answer.*

Figure 7: Qualitative example of model failure caused by masking 0.0002% WAG-identified parameters. Verbatim Qwen3-4B generations on *GSM8K* at 8,000 masked parameters indicate that WAG_{High} and |WAG| masking produces pure token repetition ("1…", "Greg…"); WAG_{Low} masking stays fluent but miscounts. All six gradient- and weight-masked variants solve the problem.

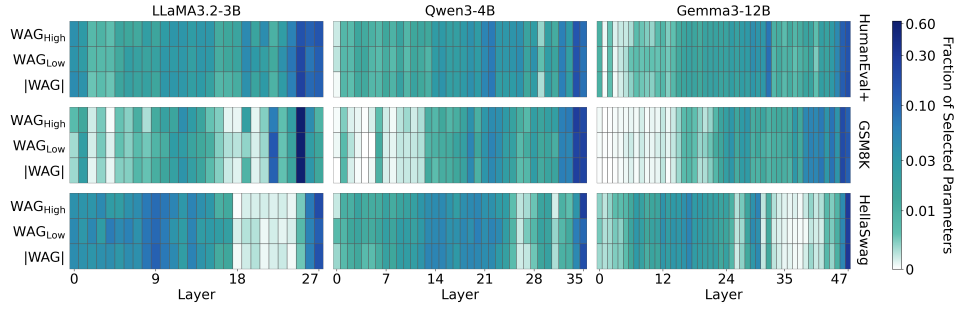


Figure 8: Distribution of 2 parts per million or 0.0002% selected parameters by WAG_{Low}, WAG_{High}, and |WAG| across layers of LLaMA3.2-3B, Qwen3-4B, and Gemma3-12B for *HumanEval+*, *GSM8K* and *HellaSwag*.

treating every parameter as equally weighted in the comparison, we compute a global rank for each parameter according to its WAG value. Formally, let w_p denote the WAG-based value of parameter p and $\text{rank}(w_p)$ its global rank among all parameters in the model, where a higher rank is assigned to parameters deemed more sensitive based on WAG criteria. $\mathcal{P}_\ell$ denotes the set of

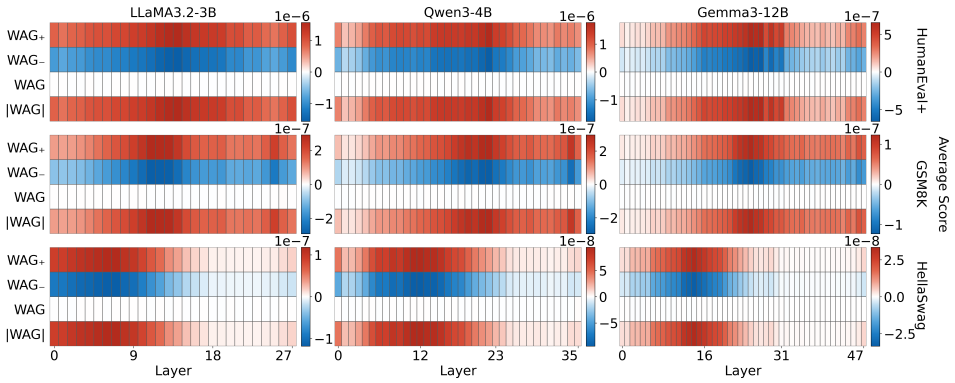


Figure 9: Layer-wise average WAG score over positive elements (WAG_+), negative elements (WAG_-), all elements (WAG), and magnitude (|WAG|) for LLaMA3.2-3B, Qwen3-4B, and Gemma3-12B on *HumanEval+*, *GSM8K* and *HellaSwag*. All panels share one symmetric-log colour scale.

parameters in layer ℓ. The score of layer ℓ is then defined as the mean rank of its parameters: $v_i = \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} \text{rank}(q_p)$, Given the layer scores $\{v_i\}_{i=1}^m$, where m is the number of layers, we convert them into integer expert counts following an approach similar to LayerIF (Askari et al., 2025) and AlphaLoRA (Qing et al., 2024). We apply a power transformation to the scores to modulate the sharpness of the allocation. First, we normalize the (inverted) scores to the interval $[0, 1]$ via min-max normalization: $\tilde{v}_i = \frac{v_i - \min_j v_j}{\max_j v_j - \min_j v_j}$. We then raise the normalized scores to a hyperparameter-controlled exponent $\beta > 0$: $\hat{v}_i = \tilde{v}_i^{\beta}$, , where β controls the dispersion of the allocation, larger values of β amplify disparities between layers. Next, we scale the transformed scores so that the total allocation, excluding a baseline of one expert per layer, sums to the target budget T: $f_i = \frac{\hat{v}_i}{\sum_{j=1}^m \hat{v}_j} \cdot (T - m)$. Each layer's preliminary allocation is obtained by flooring the fractional allocation and adding the baseline expert: $s_i = \lfloor f_i \rfloor + 1$, which guarantees $s_i \geq 1$ for all layers. Due to flooring, the total $\sum_i s_i$ may fall short of T. We therefore compute the residual budget $r = T - \sum_i s_i$ and distribute one additional expert to each of the r layers with the largest fractional remainders $f_i - \lfloor f_i \rfloor$: $s_i \leftarrow s_i + 1$ for the top-r layers ranked by $\left(f_i - \lfloor f_i \rfloor\right)$. This procedure yields the final allocation vector $S = [s_1, s_2, \ldots, s_m]$ satisfying $\sum_{i=1}^m s_i = T$ and $s_i \geq 1$ for all i.

E.3 Application B: WAG for Targeted Unlearning

In this subsection, we utilize WAG to make unlearning more effective and more efficient. We focus on improving Gradient Difference (GradDiff) (Liu et al., 2022) by applying it selectively on top of a WAG-guided layer selection. Rather than following the standard GradDiff procedure of updating all model parameters during unlearning, we restrict the updates to a targeted subset comprising 75% of the layers identified by WAG, while keeping the parameters of the remaining layers frozen. We conduct all experiments and evaluations on the *TOFU* benchmark (Maini et al., 2024), specifically on the *10% forget* dataset.

The intuition behind targeted unlearning is as follows: if unlearning is applied only to the layers containing the parameters most important with respect to the forget set, we can simultaneously (i) improve unlearning quality, since the updates concentrate on the parameters that encode the information to be removed, and (ii) reduce computational cost, since only a fraction of the parameters needs to be updated. Analogous to our expert allocation procedure in Application A 4.1, we identify the target layers as those containing the largest number of important parameters according to their mean rank of parameters following WAG-based score criteria: the lowest WAG values (WAG_{Low}), the highest WAG values (WAG_{High}), and the largest-magnitude WAG values (|WAG|). Formally, let v_ℓ denote the score of layer ℓ, where the underlying WAG values are computed with respect to the forget set $\mathcal{D}_f$. Given a selection ratio $\rho \in (0, 1]$ (we use $\rho = 0.75$), we select the set of target layers $\mathcal{L}_\rho = \arg\text{top-k}_{\ell \in \{1,\ldots,m\}} \; v_\ell, k = \lceil \rho\, m \rceil$, i.e., the $\lceil \rho m \rceil$ layers with the highest scores. During unlearning, GradDiff updates are applied only to parameters θ_p with $p \in \mathcal{P}_i$ for $i \in \mathcal{L}_\rho$, while all remaining parameters are kept frozen.

E.4 Application C: WAG for Mixed Precision Quantization

We apply WAG to group-wise mixed-precision quantization, where the goal is to retain the most sensitive weight groups in high precision while quantizing the remaining ones. Following the group-quantization scheme of SliM-LLM (Huang et al., 2025), we partition each weight matrix into contiguous groups of 128 columns. Following the experimental setup of Sec. 3.3, we quantize a BFloat16 model trained on coding tasks under the SliM-LLM pipeline, keeping the top-κ ($\kappa \in \{5\%, 10\%\}$) fraction of groups per matrix in FP16 and quantizing the remaining groups to 4-bit. The selection metric only decides *which* groups are kept; all other quantization settings follow SliM-LLM.

Since WAG scores each parameter according to its local sensitivity under multiplicative perturbations, the collective sensitivity of a group with respect to the evaluation set can be estimated by averaging the absolute scores of its parameters. Formally, let w_p denote the WAG value of parameter p, where gradients are computed on the evaluation set, and let $\mathcal{P}_g$ denote the set of parameters in group g. The |WAG| criterion scores each group by

$$s_g^{\text{WAG}} = \frac{1}{|\mathcal{P}_g|} \sum_{p \in \mathcal{P}_g} |w_p|,$$

capturing groups whose parameters are highly sensitive in either direction. We compare |WAG| against the two criteria: the group magnitude $s_g^{\text{mag}} = \sum_{p \in \mathcal{P}_g} |W_p|$ and the SparseGPT OBS salience $s_g^{\text{OBS}} = \sum_{i \in g} W_i^2 / [H^{-1}]_{ii}$, where H is the layer Hessian estimated from calibration activations (Hassibi & Stork, 1992; Frantar & Alistarh, 2023). To combine the |WAG| and (OBS) signals, we min–max normalize each score over the groups of a matrix and sum,

$$s_g^{\text{OBS}+|\text{WAG}|} = \widetilde{s}_g^{\text{OBS}} + \widetilde{s}_g^{\text{WAG}}, \qquad \widetilde{s}_g = \frac{s_g - \min_{g'} s_{g'}}{\max_{g'} s_{g'} - \min_{g'} s_{g'}}.$$

Given a preservation budget $\kappa \in \{5\%, 10\%\}$, each criterion keeps the top-κ fraction of groups ranked by its score in FP16 and quantizes all remaining groups to 4-bit via the SliM-LLM GPTQ pipeline.

E.5 Application D: WAG for Knowledge Editing

We next consider the locate-then-edit paradigm of knowledge editing, where we use WAG to identify the layers most relevant for editing. The intuition is that if we select the layers containing the largest number of parameters important to the new knowledge to be edited, the edit can be applied most effectively while minimally affecting unrelated knowledge. We evaluate this using the R-ROME (Gupta et al., 2024) editing method on four datasets: *ZSRE* (Levy et al., 2017; Wang et al., 2024b), *WikiCounterFact* (Zhang et al., 2024), *WikiBio* (Zhang et al., 2024; Hartvigsen et al., 2023), and *WikiRecent* (Zhang et al., 2024; Cohen et al., 2024). We follow the evaluation framework adopted in the recent work (Datta et al., 2026), where we select the layers based on a small proxy set of 10% of the samples and evaluate the results on the test set consisting of the rest of the samples.

Since locate-then-edit methods (Wang et al., 2024b) target MLP layers, we restrict the selection to MLP layers and score them using WAG. In the knowledge editing application,WAG should reflect the sensitivity of the parameters to the specific knowledge being edited. Let an edit sample be a query-target pair (x, y^*), where x is the prompt and $y^* = (y_1^*, \ldots, y_K^*)$ is the token sequence encoding the new knowledge. We define the gradient component of WAG via the autoregressive negative log-likelihood of the new knowledge given the query:

$$\mathcal{L}_{\text{edit}}(\theta; x, y^*) = -\sum_{k=1}^{K} \log p_\theta(y_k^* \mid x,\, y_{<k}^*)\,,$$

so that the WAG value of parameter p, denoted w_p, is computed with respect to $\nabla_{\theta_p} \mathcal{L}_{\text{edit}}$, averaged over the edit samples of the proxy set.

Since we aim to compare layers by the *number* of important parameters, weighting each parameter uniformly, we rank parameters globally. Because |WAG| captures sensitivity in both directions, each parameter is ranked according to the magnitude of its score, $r_p = \text{rank}(|w_p|)$, where $\text{rank}(\cdot)$ assigns rank 1 to the lowest value. The score of an MLP layer ℓ is then the mean rank of its parameters: $v_\ell = \frac{1}{|\mathcal{P}_\ell|} \sum_{p \in \mathcal{P}_\ell} r_p$, where $\mathcal{P}_\ell$ is the set of parameters in the MLP sublayer of layer ℓ, and the layer with the highest v_ℓ, the highest concentration of important parameters is selected for editing. Layers are identified once using the proxy set, and all evaluations are conducted on unseen test data.

F CODE AND REPRODUCIBILITY

We provide our code and implementation in an open-source repository: `https://anonymous.4open.science/r/WAG-F8C0/`. All the experiments were conducted on a Linux server with 6x NVIDIA DGX B200 GPUs with 192 GB VRAM/GPU. All the utilized LLMs are original (unquantized) open-source versions from HuggingFace. During generation, we used greedy decoding without sampling. To ensure reproducibility, all sources of randomness, including PyTorch, NumPy, and Python's random module, were fixed using a seed of 42. For the experiments involving more than one run, we report the mean and SEM of a total of 3 runs with seeds 42, 43, and 44, respectively. For experiments of empirical analysis and Application C, we built on the MFT codebase (Zhang et al., 2025) and evaluations we conducted following their convention using OLMES (Gu et al., 2025). For Applications A and B, we followed the training and evaluation conventions of LayerIF (Askari et al., 2025) and TOFU (Maini et al., 2024) codebases, respectively. Our codebase for Application D was built upon the *EasyEdit* (Wang et al., 2024b) framework and LGA codebase (Datta et al., 2026), and evaluations of the edited models were performed following their standard conventions.