SIGNAL FROM NOISE: ACTIVATION REDISTRIBUTION IMPROVES REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Motivated by the puzzling observation that prepending long sequences of meaningless tokens can consistently enhance LLM reasoning performance, we investigate one mechanism underlying this effect and propose a more principled alternative. We find that these improvements are associated with a redistribution of activations in the MLP layers, where near-zero activations become less frequent while large-magnitude activations increase, effectively suppressing weak signals and promoting more informative representations. Building on this observation, we propose the Activation Redistribution Module (ARM), a lightweight inference-time technique that directly modifies post-nonlinearity activations without altering the input sequence. ARM targets the activation redistribution associated with meaningless tokens, capturing its benefits in a controlled manner without claiming to reproduce every effect of token insertion. Extensive experiments across diverse benchmarks and model architectures demonstrate that ARM consistently improves reasoning performance while requiring only minimal implementation overhead.

1 INTRODUCTION

Large language models (LLMs) are known to be sensitive to subtle variations in their inputs, which makes it important to understand how tokens influence predictions (Guan et al., 2025; Errica et al., 2024; Zhuo et al., 2024). In this paper, we present a surprisingly counterintuitive finding named **meaningless-token effect**: inserting **long sequences of meaningless tokens**, such as repeated punctuation or separators, into prompts can consistently improve the performance of LLMs, particularly on reasoning tasks. Contrary to common intuition that long and irrelevant tokens are like noise and thus useless or even harmful during inference (Jiang et al., 2024; Guan et al., 2025), our experiments reveal the opposite. When long sequences of meaningless tokens are appended before query prompts, models that previously struggled with certain problems can produce correct solutions, as illustrated in the left panel of Figure 1. This effect occurs consistently across tasks and models, suggesting a counterintuitive behavior of LLMs that merits deeper investigation.

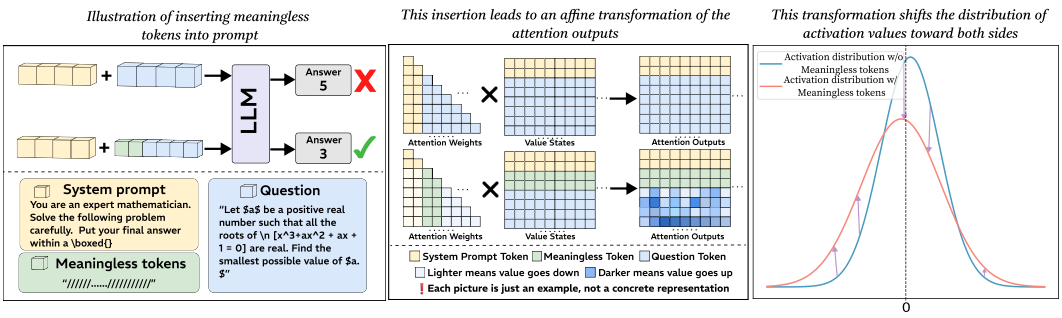


Figure 1: The left panel illustrates how meaningless-token effect can improve model performance. The middle panel shows the changes occurring in the attention module after introducing meaningless tokens. The right panel depicts the redistribution of activations that results from adding these tokens.

This unexpected result raises fundamental questions about how LLMs process input and what aspects of their internal computation are being affected. *Why should tokens that convey no meaning lead to*

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.

measurable performance gains? Are they simply acting as noise, or do they restructure representations in a systematic way that supports better reasoning? To answer these questions, we move beyond surface-level observations and conduct a detailed investigation of the mechanisms behind this effect. Our analysis shows that the clearest and most regular influence of meaningless tokens appears in the first layer, where their effect on meaningful tokens can be approximated as an affine transformation of the attention outputs. As demonstrated in the middle schematic diagram of Figure 1, the resulting transformation shifts the distribution of activations in the MLP: the proportion of near-zero activations decreases, while more activations are pushed outward toward larger positive and negative values. The rightmost plot in Figure 1 visualizes this process. We hypothesize that redistribution fosters richer exploration, enhancing reasoning performance, and clarify the mechanism by decomposing the transformation into coefficient and bias terms. Our theoretical analysis shows how each component shapes activation variance and induces the observed distributional shift in the model's activations.

Building on these insights, we propose **ARM** (an ***A***ctivation ***R***edistribution ***M***odule), a lightweight alternative to explicit meaningless-token insertion. ARM requires only a few lines of code modification and no additional training. It automatically identifies a proportion of near-zero activations after the non-linear function and shifts their values outward, targeting the activation pattern associated with the meaningless-token intervention. ARM is therefore an inference-time instantiation of the mechanism studied here, rather than a claim to reproduce every effect caused by inserting tokens. The key findings and contributions of our work are:

- We uncover a **meaningless-token effect in LLMs**: inserting meaningless tokens, far from being harmful, systematically improves reasoning in LLMs. This runs counter to the common assumption that such tokens only add noise.
- Through theoretical and empirical analysis, we identify and support **activation redistribution** in the first-layer MLP as a mechanism consistent with the meaningless-token effect, reducing the prevalence of near-zero activations and increasing overall activation variance. We do not claim that it is the only mechanism involved or that our analysis isolates its contribution from all other propagation paths.
- Building on this understanding, we present ARM, a lightweight inference-time instantiation that targets this mechanism without modifying the input sequence.

2 Observation: Inserting Meaningless Tokens Induces an Affine Transformation on Meaningful Token Representations

We observe that inserting meaningless tokens, such as fixed-length sequences of repeated slashes ("/")—can significantly improve LLM performance, particularly on reasoning tasks[1]. Unlike prior work that trains models to generate pause tokens during inference (Goyal et al., 2023; Kim et al., 2025; London & Kanade, 2025), our approach is entirely training-free: simply inserting arbitrary, non semantic tokens consistently improves performance. While some studies examine both positive and negative effects of meaningless tokens, they typically require carefully selected strings (Duan et al., 2025) or relatively short token sequences (Chadimová et al., 2024). In contrast, our method uses long, fixed-length sequences without string-specific tuning. Moreover, unlike prior placements at the input end (Lanham et al., 2023) or within chain-of-thought (Goyal et al., 2023), we insert meaningless tokens between the system prompt and the question.

Table 1: Performance on mathematical reasoning datasets with and without meaningless tokens across models. "w/o" and "w/" denote the absence and presence of meaningless tokens, respectively. Results are averaged over five runs; subscripts indicate the type and number of meaningless tokens. In Section G.2, we report pass@4/8/16 results and different temperature settings. We also show the termination rate of each models on two benchmark at Table 12. In this experiments, we set temperature is 0.7 and max length follow the offical reports.

Methods	MATH-500		AIME22-24	
	w/o	w/	w/o	w/
Qwen2.5-Math-1.5B	63.9 ± 0.27	$\mathbf{65.9 \pm 0.71}_{55/}$	14.4 ± 0.00	$\mathbf{17.5 \pm 1.2}_{70/}$
Qwen2.5-Math-7B	72.3 ± 0.13	$\mathbf{74.6 \pm 0.54}_{55/}$	23.1 ± 0.27	$\mathbf{23.3 \pm 0.00}_{55/}$
DS-R1-Qwen-7B	91.8 ± 0.57	$\mathbf{93.0 \pm 0.68}_{10/}$	52.2 ± 0.09	$\mathbf{56.7 \pm 0.0}_{70/}$
DS-Math-7B-instruct	39.5 ± 0.14	$\mathbf{42.1 \pm 1.75}_{55/}$	7.8 ± 0.00	$\mathbf{12.3 \pm 0.27}_{10/}$
Llama-3.1-8B-Instruct	46.6 ± 1.64	$\mathbf{47.2 \pm 0.71}_{55/}$	7.9 ± 0.16	$\mathbf{9.9 \pm 0.49}_{70/}$

Under this setting, the evaluated models generally exhibit performance improvements, although the magnitude varies across models and tasks. As shown in Table 1, we compare model performance on Math-500 and AIME22–24 before and after inserting meaningless tokens. This pattern suggests

[1] Varying token length, type, and position affects performance, as shown in Appendix F.

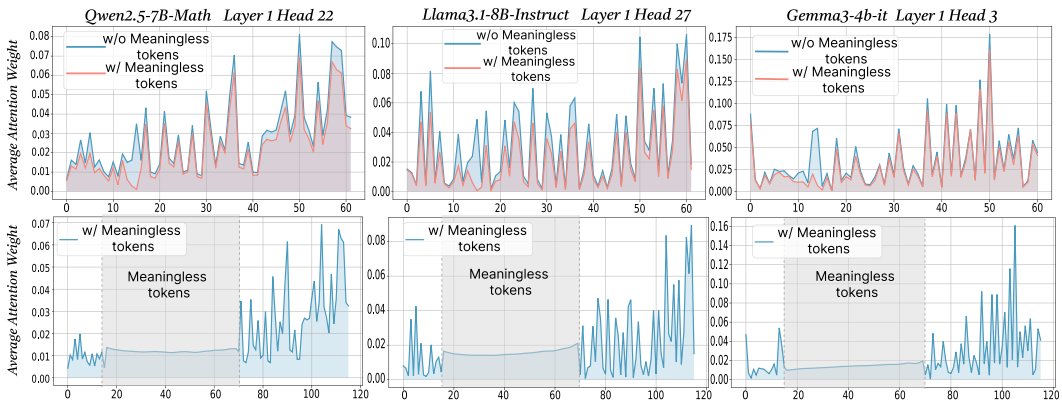


Figure 2: The x-axis shows token indices. The top row of Figure 2 shows how the model's attention to meaningful tokens changes after inserting meaningless tokens, where red denotes the attention after insertion and blue denotes the attention before insertion. For visualization clarity, we omit the meaningless tokens in this view. The bottom row of Figure 2 presents the attention received by the meaningless tokens themselves, which is generally low in the displayed examples. We show additional average attention weights in Appendix H and layer-wise analyses in Section F.7.

that the inserted meaningless tokens are not simply ignored or uniformly detrimental to the models; rather, they interact with the models' internal representations in a task- and model-dependent way. To investigate this phenomenon, we start our analysis from the attention module. The formula of attention is

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{1}$$

For a sequence of length n, $Q \in \mathbb{R}^{n\times d_k}$, $K \in \mathbb{R}^{n\times d_k}$, and $V \in \mathbb{R}^{n\times d_v}$ are obtained by row-stacking the query, key, and value states, respectively; d_k and d_v are the key/query and value dimensions. Adding tokens introduces additional terms into the softmax normalization, enlarging its denominator. Although the inserted tokens typically receive small weights, their presence redistributes probability mass and reduces the relative share of attention allocated to the original tokens. To probe this effect, we directly compare attention weights with and without meaningless tokens while keeping the original token positions aligned in the first layer. For every original token, we compute the mean of its attention column below the diagonal to measure the attention it receives from downstream tokens (Bogdan et al., 2025).

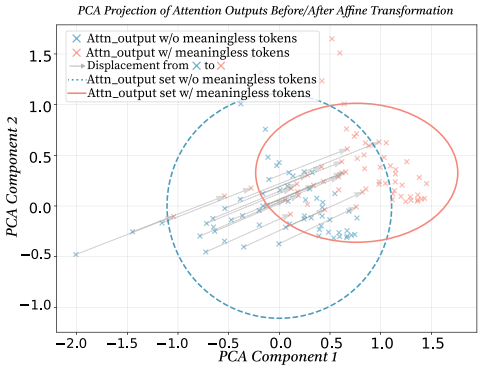


Figure 3: After adding meaningless tokens, each token vector is affinely transformed: blue points show the original vectors, and red points show them after the addition. The arrow indicates the change in direction. The x- and y-axes correspond to the first two principal components from PCA, illustrating the relative relationships before and after the affine transformation.

When a string of meaningless tokens are present, the model assigns only small weights to each token, intuitively indicating that the model only pays little attention to them (see Figure 2 bottom row). The top row of Figure 2 presents a direct comparison of the attention to meaningful tokens without (blue) or with meaningless tokens (red; meaningless token indices are removed from visualization to allow for direct comparison). In the displayed heads, attention assigned to original meaningful tokens tends to decrease in the meaningless-token condition, particularly through a reduction in high-attention spikes. These head-level plots provide qualitative intuition rather than aggregate evidence across all heads and all inputs.

Let $\mathcal{O}$ and $\mathcal{I}$ denote the original and inserted token positions. For a fixed query token $t \in \mathcal{O}$, let $W_{t,\mathcal{O}}$ and $W'_{t,\mathcal{O}}$ denote the attention-weight row over original tokens before and after insertion, and let $W'_{t,\mathcal{I}}$ denote the row over inserted tokens after insertion. Empirically, the original-token part is well

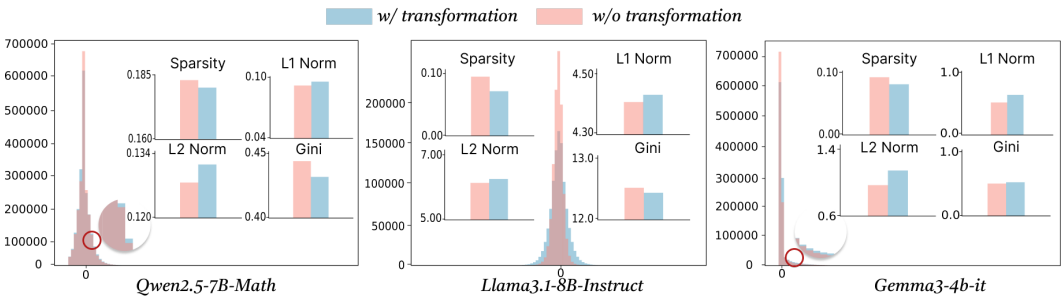


Figure 4: Empirical post-activation distributions from actual model runs without and with inserted meaningless tokens. The inset compares four distributional metrics between the two conditions. The affine transformation in Equation 4 is a theoretical approximation used to explain this intervention; the distributions are not obtained by manually applying Eq. Equation 4. In Section F.4, we test for statistical differences in variance across the two conditions.

approximated by

$$W'_{t,\mathcal{O}} \approx \lambda_t \, W_{t,\mathcal{O}}, \qquad 0 < \lambda_t \leq 1, \tag{2}$$

where λ_t is the retained attention-mass coefficient (so $1 - \lambda_t$ is the corresponding drop fraction). The row $W'_{t,\mathcal{O}}$ has the same shape as $W_{t,\mathcal{O}}$; the subscripts identify token positions and do not denote matrix dimensions. Here t indexes the query token, while $j \in \mathcal{O}$ and $i \in \mathcal{I}$ index original and inserted key/value positions. The attention output for the query token includes contributions from both groups:

$$\text{Attn_Output}_{\text{new},t} = W'_{t,\mathcal{O}} V_{\mathcal{O}} + W'_{t,\mathcal{I}} V_{\mathcal{I}}, \tag{3}$$

where $V_{\mathcal{O}} \in \mathbb{R}^{|\mathcal{O}| \times d_v}$ and $V_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}| \times d_v}$ are the corresponding value-state matrices. Equivalently, j and i index the individual terms in the two matrix products. As the inserted tokens are repeated and contribute little semantic information, their value states are similar and their attention weights are small and similar in magnitude. Their accumulated contribution therefore primarily shifts the attention output along an approximately common direction rather than introducing diverse semantic components. We denote this contribution by $\Sigma_{\sigma,t} := W'_{t,\mathcal{I}} V_{\mathcal{I}}$. Positional encoding is already incorporated into the post-softmax attention weights used in this analysis. Combining the scaling and additive terms gives

$$\text{Attn_Output}_{\text{new},t} \approx \lambda_t \, \text{Attn_Output}_{\text{old},t} + \Sigma_{\sigma,t}, \tag{4}$$

where $\text{Attn_Output}_{\text{old},t} = W_{t,\mathcal{O}} V_{\mathcal{O}}$. Thus, the insertion is modeled as an affine transformation of each original token's attention output. This is an approximation to the empirical intervention, not a manually applied transformation used to produce the activation plots. After the attention module, the transformed output passes through RMSNorm and serves as input to the MLP. In the next section, we examine how this transformation propagates through the MLP.

3 Analysis: Why Affine Transformations Improve Reasoning Performance

Having established in the previous sections that the meaningless-token effect is well approximated by scaling and additive terms in the attention output, we next examine how this transformation propagates through the subsequent MLP modules and affects reasoning. In Equation 4, λ_t controls the retained magnitude of the original attention output and $\Sigma_{\sigma,t}$ is the additive contribution of the inserted tokens. In the actual intervention, $\Sigma_{\sigma,t}$ is structured and need not be zero-mean or statistically independent of the original output. For the local variance calculation below, we center this contribution and use a bounded, zero-mean stochastic approximation solely to obtain a tractable first-order analysis. Together, the scaling and additive terms determine how the transformed attention representations shape the dynamics of the MLP layers.

3.1 How the Affine Transformation Influences the Gate-Projection Output

Key Takeaway

We show that the scaling and additive terms can increase the variance of the gate layer's output under the sufficient local conditions stated below.

In this part, we show how these two factors can increase the variance of the gate-projection outputs in the MLP. Because the inserted tokens have similar values and typically receive small attention weights, their aggregate contribution is often close to a low-dimensional shift; RMSNorm attenuates its global-magnitude component, while the remaining directional component can still affect the normalized representation. Specifically, for the **scaling factors**, before entering the MLP, the attention output undergoes output projection and residual connection, which can be written as $x_t(\lambda) = \text{res}_t + \lambda U A_t$, where A_t is the original attention output for query token t, U is the attention output-projection matrix, and juxtaposition denotes matrix multiplication. Treating λ as a functional variable, the RMSNorm output becomes $y_t(\lambda) = \text{RMSNorm}(x_t(\lambda))$. For the j-th output dimension of the MLP gate projection, $z_{t,j}(\lambda) = w_j^\top y_t(\lambda)$, where w_j is the corresponding gate-projection weight vector. The residual stream is therefore part of the propagation path considered here, but this formulation does not separate its direct contribution from the contribution mediated by the MLP. Our claims are consequently limited to the MLP-mediated activation redistribution and do not attribute all downstream changes to this component. A small local variation $\Delta\lambda$ then leads to the variance change analyzed below.

$$\text{Var}[z_j(\lambda + \Delta\lambda)] = \text{Var}[z_j(\lambda)] + 2\,\text{Cov}(z_j(\lambda), g_j(\lambda))\;\Delta\lambda + \text{Var}[g_j(\lambda)]\;\Delta\lambda^2. \tag{5}$$

Throughout this analysis, $\text{Var}[z_{t,j}]$ and $\text{Cov}(z_{t,j}, g_{t,j})$ are taken over the activation population $\mathcal{P}$ formed by the same aligned meaningful-token positions across the evaluation examples. Thus, the variance is computed for a fixed gate dimension across input samples; it is not a variance across hidden dimensions or across tokens within one sample. This is the population-level quantity estimated empirically for each gate dimension in the activation-distribution comparisons in Figure 4 and Table 5. The third term in Equation 5 is nonnegative and can dominate the covariance term within the sufficient local range derived in Appendix E; this is not an unconditional statement for arbitrary affine transformations. In the case of **bias factors**, we use a bounded, zero-mean, statistically independent perturbation only as a local analytical approximation. The actual perturbation induced by meaningless tokens remains the structured affine contribution described by Equation 4; the approximation isolates the covariance contribution of the additive term and makes its interaction with RMSNorm tractable. Formally, under this approximation, after perturbation the RMSNorm input can be written as $x = x_0 + W\Sigma_\sigma$, where Σ_σ denotes the centered random additive perturbation and W is the linear coefficient preceding RMSNorm. After normalization, the covariance of the output can be expressed as:

$$\text{Cov}(y) = J_q \text{Cov}(x) J_q^\top + o(\|x - x_0\|^2) \tag{6}$$

where x_0 is the expansion point and J_q is the Jacobian matrix of the RMSNorm mapping. Since the local perturbation is assumed small, higher-order terms are disregarded. Under this approximation, the additive term contributes to the covariance of y. Writing the perturbed RMSNorm input as $x = x_0 + \delta x$ with $\delta x = W\Sigma_\sigma$, the input to the activation function can be written as $z = W_{gate}x$. Based on the covariance, the variance of the j-th gate dimension is given by:

$$\text{Var}[z_j] \approx e_j^\top W_{gate} J_q(x_0) W\,\text{Cov}(\Sigma_\sigma) W^\top J_q(x_0)^\top W_{gate}^\top e_j, \tag{7}$$

since the projection of the vector onto the tangent space is almost never zero in LLMs' high dimensions, the local additive perturbation contributes a nonzero covariance component. Under the stated approximation, these two factors can therefore increase the variance of the output. In general, the scaling factors increase variance by amplifying inter-sample differences, whereas the additive factors can increase variance by enlarging the covariance structure across dimensions. An empirical check of the sufficient condition in Appendix E is provided in Appendix F.2.

3.2 VARIANCE CHANGE LEADS TO ACTIVATION REDISTRIBUTION

Key Takeaway

Under the local approximation, an increase in the input variance of the activation functions can broaden and reshape the output activation distribution by changing its mean and variance.

As the variance of gate layer outputs grows under perturbations, the subsequent activation function further reshapes these signals by compressing values near zero. This motivates redistributing near-zero activations. For each sample in the hidden state, the second-order Taylor expansion on ϕ, the activation function output is:

$$\phi(\mu+\sigma) = \phi(\mu) + \phi^{'}(\mu)\sigma + \frac{1}{2}\phi^{''}(\mu)\sigma^2 + o(|\sigma|^3), \tag{8}$$

where σ can represent both Δk in scaling factor and Σ_σ in bias factor. We denote the input to the activation function as $z = \mu + \sigma$. For the j-th dimension of the hidden state, the expectation and variance of the activation output can be expressed as:

$$\mathbb{E}[\phi(z_j)] = \mathbb{E}[\phi(\mu_j)] + \mathbb{E}[\phi'(\mu_j)\,\sigma] + \mathbb{E}\big[\tfrac{1}{2}\,\phi''(\mu_j)\,\sigma^2\big] + o\big(\mathbb{E}|\sigma|^3\big)\,. \tag{9}$$

$$\mathrm{Var}[\phi(z_j)] = \phi^{'}(\mu_j)^2\mathrm{Var}_j + o(\mathrm{Var}_j^2). \tag{10}$$

From the equations above, we infer that the observed distributional change can be described through variations in expectation and variance. On a single dimension, activations shift in both directions; from Section 3.1, higher-order terms are negligible, and the first derivative of GeLU/SiLU near zero is positive. Since the local perturbations include both signs, extrapolated activations can fluctuate around zero. From Equation 9, $\mathbb{E}[\sigma^2] = \mathrm{Var}_j$. For the bias factor, the centered perturbation removes the first-order term. For the observed scaling perturbation, expanding at the population mean gives $\mathbb{E}[\phi^{'}(z_j)g_j] = 0$, again canceling the first order. The second derivative near zero is strictly positive. From Equation 10, Var_j increases, and so does the activation histogram variance, as the function is nearly linear near zero. These statements concern the particular, mild affine perturbation induced by meaningless-token insertion; they do not imply that every transformation with a similar variance change, such as arbitrary negative scaling, produces the same redistribution. In summary, the observed scaling and additive factors jointly enlarge activation variance, expressed as:

$$\mathrm{Var}_j \approx \mathbb{E}[\mathrm{Var}_j^{(\Sigma_\sigma)}] + \mathrm{Var}(g_j^{\lambda}). \tag{11}$$

The first term represents the expected variance of the j-th hidden states under the influence of the bias factor. Since the bias factor varies across individual cases, taking the expectation is necessary to capture its overall impact. The second term corresponds to the variance induced by scaling factors, which inherently reflects the aggregate change. When combining them, the overall variance of the outputs of nonlinear activation functions increases, the mean shifts upward, and the activation distribution becomes broader, manifested as heavier tails and a thinner center. More details of above analysis and relative proof are in Appendix E. We hypothesize that the improvement is related to reasoning-critical tokens (e.g., digits, operators, conjunctions) having more near-zero activations; boosting them strengthens representations and enhances reasoning (see Appendix C). We analyze the impact of activation functions in Section F.5. In Section F.8, we also analyze how affine transformation in first layer will influence the following layers in Section F.8.

3.3 VERIFICATION OF ACTIVATION REDISTRIBUTION

To verify whether the activation redistribution pattern in Section 3.2 indeed occurs in LLMs, Figure 4 illustrates the post-activation distribution of the first-layer MLP under the actual meaningless-token intervention. Specifically, we compare states from model runs without inserted tokens and with inserted tokens; we do not manually construct or apply Equation 4 to generate this figure. The affine expression in Equation 4 is a theoretical approximation that explains the empirical intervention. We also assess the resulting activation changes using several quantitative indicators, including:

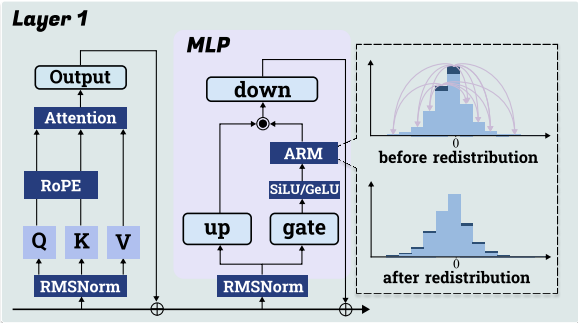
- **Relative Sparsity**: Defined as the proportion of activations after the transformation whose values fall below the pre-transformation threshold.
- **L1 Norm**: The sum of the absolute activation values; smaller values indicate higher sparsity.

- **L2 Norm**: The overall magnitude of activations.
- **Gini Coefficient**: An indicator of the smoothness of the histogram distribution, where smaller absolute values correspond to smoother distributions.

From Figure 4, we observe that after inserting meaningless tokens, the frequency of near-zero activations decreases, while the frequency of absolute high-magnitude activations increases. Both sparsity and smoothness in the activation distribution are improved. Specifically, the relative sparsity consistently decreases across all three models while the L1 and L2 norms increase, aligning with the theoretical prediction and clarifying that the plotted distributions are empirical outputs of the token intervention. To further examine the role of activation redistribution, we replace the first-layer activations after meaningless-token insertion with the corresponding activations from the original input. On MATH-500 with Qwen2.5-Math-1.5B, **this intervention reduces Pass@1 accuracy from 63.8% to 62.6%,** suggesting that removing the induced activation redistribution eliminates the observed performance gain.

4 Method: Activation Redistribution Module

Inspired by the previous finding that meaningless tokens can shift meaningful activations and boost LLM performance, we propose ARM—a simple internal intervention that targets the activation redistribution associated with meaningless tokens without inserting additional input tokens. Our approach has two steps: first, it adaptively identifies a proportion of near-zero activations based on the model and input; then, it extrapolates them outward to redistribute the activation pattern. The top half of Figure 5 illustrates the first-layer MLP with ARM, where near-zero activations are shifted outward, reducing their frequency and increasing large-magnitude activations. The bottom half of Figure 5 presents the ARM-specific code, a lightweight function inserted into the first-layer MLP with negligible overhead. As shown in Appendix D, ARM's time complexity is negligible within the MLP context. ARM provides complementary constructive evidence consistent with the analysis in Section 3: applying the targeted redistribution directly yields reasoning gains across models and benchmarks. It also provides a lightweight inference-time intervention compatible with existing inference-time scaling methods (Appendix G.6), without claiming to reproduce all effects of explicit token insertion or fully account for the gains. We discuss why activation redistribution improves model performance in Appendix C and compare ARM with global activation scaling to distinguish selective redistribution from uniform amplification (Appendix G.9).




```
def forward(x, layer_idx): # in first layer
    activation = self.act_fn(self.gate_proj(x))
    #Our function
    activation_alter = self.arm(activation.clone())
    down_proj = self.down_proj(activation_alter *
        self.up_proj(x))
    return down_proj
```

Figure 5: The upper panel illustrates the first-layer LLM architecture with ARM, while the lower panel presents the corresponding ARM code in the MLP module.

4.1 Selecting the Appropriate Change Proportion

Our method first determines the proportion of activations to be modified. Since different models may exhibit different sensitivities to meaningless tokens, we adopt a dynamic strategy that adapts the modification proportion to the activation distribution during inference. Specifically, we characterize the concentration of activations around zero and define a neighborhood ϵ to identify near-zero activations. We use the Median Absolute Deviation (MAD) as a robust measure of dispersion, as it is less sensitive to outliers and better reflects the central activation distribution. The neighborhood is defined as $\epsilon = \kappa \cdot \mathrm{MAD} \cdot c$, where κ is a consistency constant and c controls the width of the near-zero region. We then compute the fraction p of activations within $[-\epsilon, \epsilon]$ and determine the modification proportion as

$$\text{fraction} = \mathrm{clip}(p, p_{\min}, p_{\max}), \tag{12}$$

Table 2: After adding ARM to the first-layer MLP, we report reasoning-task performance for six models, using a dash ('–') for accuracies below 5% to indicate incapability. We test pass@3, pass@4, pass@8, pass@16 of some models in several benchmarks in Section G.3.

Model	Setting	GPQA Diamond	Math-500	AIME 2022-2024	AIME 2025	LiveCodeBench	Humaneval
Qwen2.5 Math-1.5B	Baseline	27.3	63.8	14.4	6.7	-	23.2
	ARM	28.8	67.0	18.9	10.0	-	28.4
	Absolute Improvement (pp)	**1.5**↑	**3.2**↑	**4.5**↑	**3.3**↑	-	**5.2**↑
Qwen2.5 Math-7B	Baseline	30.3	72.4	23.3	10.0	-	48.5
	ARM	34.9	73.4	25.6	13.3	-	51.1
	Absolute Improvement (pp)	**4.6**↑	**1.0**↑	**2.3**↑	**3.3**↑	-	**2.6**↑
Qwen2.5 7B-Instruct	Baseline	28.3	73.2	20.0	10.0	29.7	43.9
	ARM	29.8	75.2	20.0	23.3	31.9	47.6
	Absolute Improvement (pp)	**1.5**↑	**2.0**↑	0	**13.3**↑	**2.2**↑	**3.7**↑
Llama3.1 8B-Instruct	Baseline	28.3	43.0	11.1	-	11.9	45.7
	ARM	31.3	45.8	13.3	-	17.0	47.6
	Absolute Improvement (pp)	**3.0**↑	**2.8**↑	**2.2**↑	-	**5.1**↑	**1.9**↑
Gemma3 4b-it	Baseline	34.3	72.6	13.3	20.0	20.2	68.3
	ARM	35.9	74.0	17.8	23.3	20.6	68.9
	Absolute Improvement (pp)	**1.5**↑	**1.4**↑	**4.5**↑	**3.3**↑	**0.4**↑	**0.6**↑
Gemma3 27b-it	Baseline	33.3	85.4	25.6	26.7	31.9	86.6
	ARM	33.8	86.2	31.1	30.0	34.2	88.4
	Absolute Improvement (pp)	**0.5**↑	**0.8**↑	**4.4**↑	**3.3**↑	**2.3**↑	**1.8**↑

where $p_{\min}$ and $p_{\max}$ provide practical bounds to avoid excessively small or large modifications. Thus, the proportion of modified activations is primarily determined by the activation distribution rather than being manually specified for each model.

4.2 Redistribution of Activation Values

After selecting the activations, we preserve their signs and modify only their magnitudes. The redistribution range is defined as

$$\mathrm{R} = \begin{cases} [0, Q_{p_1}(\text{Activations})], & \text{sign} = 1, \\ [\min(\text{Activations}), 0], & \text{sign} = -1, \end{cases} \tag{13}$$

where R denotes the range used for activation redistribution. For negative activations, the minimum observed activation provides a natural lower bound, consistent with the bounded negative responses of activation functions such as SiLU and GeLU. For positive activations, $Q_{p_1}(\text{Activations})$ denotes an upper quantile of the activation distribution, which prevents the redistributed values from extending into extreme activation regions. A value is sampled from the corresponding range and added to the selected activation while preserving its sign. In this way, both the number of modified activations and their redistribution ranges are guided by the observed activation distribution, allowing ARM to adapt its intervention across different models without requiring a manually specified modification ratio.

5 Experiments

We evaluate our method on reasoning and non-reasoning tasks using six models: Qwen2.5-Math-1.5B, Qwen2.5-Math-7B, Qwen2.5-Instruct-7B (Qwen2.5 Technical Report), Llama3.1-8B-Instruct (The Llama 3 Herd of Models), Gemma3-4b-it, and Gemma3-27b-it (Gemma 3 Technical Report). The decoding configuration is specified with the corresponding result tables and supplementary descriptions; we do not treat all settings as a single "default" configuration. For reasoning tasks, we cover three skill areas: (1) General: GPQA (Rein et al., 2024), a challenging expert-authored multiple-choice dataset; (2) Math & Text Reasoning: MATH-500 (Lightman et al., 2023), AIME22-24 (AIME, 2024), and AIME'25 (AIME, 2025); (3) Agent & Coding: LiveCodeBench (Jain et al., 2024) and HumanEval (Chen et al., 2021). For non-reasoning tasks, we use GSM8K (Cobbe et al., 2021), ARC-E (Clark et al., 2018), ARC-C (Clark et al., 2018), MMLU (Hendrycks et al., 2021), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), and OpenBookQA (Mihaylov et al., 2018). In this experiments, we set temperature of pass@1 to 0, pass@n to 0.7, all settings follow offical reports.

5.1 Experiment Results Analysis

For **reasoning tasks**, the results in Table 2 show accuracy across multiple benchmarks. Our method consistently improves performance across most models and datasets. On challenging benchmarks, however, improvements are limited when models lack sufficient capacity or when baseline accuracy is near saturation. For **non-reasoning tasks** (see Table 3b), applying ARM to the first-layer MLP

yields little change. We attribute this to their largely factual nature, where models already have the necessary knowledge and response formats, requiring minimal reasoning. By contrast, for reasoning tasks, altering early activations helps reorganize knowledge, strengthens intermediate representations, and facilitates more effective and consistent reasoning.

Table 3: Table (a) compares the performance of meaningless tokens and ARM, and Table (b) reports ARM's results on non-reasoning tasks.

(a) Pass@1 on Math-500 and AIME 22-24 with meaningless tokens (Mless) or ARM.

Model	Setting	Math-500	AIME 22-24
Qwen2.5 Math-7B	Baseline	72.4	23.3
	Mless	**75.0**	24.4
	ARM	73.4	**25.6**
Llama3.1 8B-Instruct	Baseline	43.0	11.1
	Mless	44.9	**13.3**
	ARM	**45.8**	**13.3**

(b) Performance of models with ARM on non-reasoning tasks. Additional results are in Appendix G.

Model	Setting	GSM8K	ARC-E	HellaSwag
Qwen2.5 Math-1.5B	Baseline	78.0	39.3	39.1
	ARM	78.6	39.3	39.5
	Improve Rate (%)	**0.6**↑	0	**0.4**↑
Llama3.1 8B-Instruct	Baseline	80.0	46.6	56.8
	ARM	82.4	47.1	57.3
	Improve Rate (%)	**2.4**↑	**0.5**↑	**0.5**↑

5.2 Comparison of Meaningless tokens and ARM

In Table 3a, we provide a direct comparison between our proposed ARM method and the strategy of inserting a suitable number of meaningless tokens. The results demonstrate that both approaches can improve model performance without post-training. However, the token-level intervention depends on the specific choice of token length, type, and placement, and can therefore be unstable across tasks and models. ARM targets the activation redistribution identified in our analysis directly within the MLP, yielding a more controlled intervention without heuristic token engineering. We emphasize that ARM is an instantiation of this mechanism, not a complete replacement for or reproduction of every effect caused by explicit token insertion. This comparison highlights the trade-off between prompt-level simplicity and the control of an internal activation intervention.

6 Related Work

Recent studies notice that symbols in an LLM's input may affect their internal mechanism. (Sun et al., 2024) show large activations for separators, periods, or newlines, suggesting that these tokens carry model biases. (Razzhigaev et al., 2025) find that commas are essential for contextual memory, while (Chauhan et al., 2025) and (Min et al., 2024) highlight punctuation as attention sinks, memory aids, and semantic cues. Moreover, (Chadimová et al., 2024) show that replacing words with meaningless tokens can reduce cognitive biases, whereas (Li et al., 2024) report that such "glitch tokens" may also cause misunderstandings, refusals, or irrelevant outputs.(Duan et al., 2025) study carefully designed meaningless tokens through iterative prompt optimization, and (Chadimová et al., 2024) examine short meaningless-token interventions. Jang et al. (Jang et al., 2025) most directly overlap with our empirical observation: they study when filler tokens help by varying token types, counts, and insertion positions, and analyze the resulting attention maps. Our work is complementary: we focus on a mechanistic account centered on activation redistribution and derive ARM as an internal intervention that targets this pattern without inserting filler tokens. Thus, our contribution is not the observation that filler tokens can help by itself, but the mechanistic analysis and the method built from it. **We provide an extended discussion of related works in Appendix A.**

7 Conclusion

In this paper, we report a meaningless-token effect in which inserting long sequences of meaningless tokens improves model performance, particularly on reasoning tasks. Our analysis shows that the intervention induces an affine transformation on meaningful-token representations and identifies activation redistribution as a mechanism consistent with the observed changes in reasoning performance. This explanation is not intended to exclude other mechanisms or to separate the contribution of the MLP-mediated path from all residual-stream effects. Building on this insight, we introduce ARM, a lightweight and training-free method that targets the same redistribution internally, providing a practical inference-time intervention without claiming to reproduce every consequence of explicit token insertion.

AI USE STATEMENT

In this paper, we used AI tools solely to refine grammar and improve the clarity and readability of the text.

ETHICS STATEMENT

All datasets used in this work are publicly available and contain no sensitive information. Our method enhances LLM reasoning without introducing new data collection or human interaction. While stronger reasoning ability may be misused, we emphasize that this work is intended for beneficial research and responsible applications.

REPRODUCIBILITY STATEMENT

We will release our code and data once the paper is published. The appendix includes detailed experimental setups and hyperparameters so that others can reproduce our results. We also encourage the community to follow good research practices when using our code and data, to help maintain the reliability and transparency of future work.

REFERENCES

AIME. Aime problems and solutions, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AIME. Aime problems and solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter? arXiv preprint arXiv:2506.19143, 2025.

Milena Chadimová, Eduard Jurášek, and Tomáš Kliegr. Meaningless is better: hashing bias-inducing words in llm prompts improves performance in logical reasoning and statistical learning. arXiv preprint arXiv:2411.17304, 2024.

Sonakshi Chauhan, Maheep Chaudhary, Koby Choy, Samuel Nellessen, and Nandi Schoots. Punctuation and predicates in language models. arXiv preprint arXiv:2508.14067, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In NAACL, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv:1803.05457v1, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Varun Dhanraj and Chris Eliasmith. Improving rule-based reasoning in llms via neurosymbolic representations. arXiv e-prints, pp. arXiv–2502, 2025.

Keyu Duan, Yiran Zhao, Zhili Feng, Jinjie Ni, Tianyu Pang, Qian Liu, Tianle Cai, Longxu Dou, Kenji Kawaguchi, Anirudh Goyal, et al. Unnatural languages are not bugs but features for llms. arXiv preprint arXiv:2503.01926, 2025.

Federico Errica, Giuseppe Siracusano, Davide Sanvito, and Roberto Bifulco. What did i do wrong? quantifying llms' sensitivity and consistency to prompt engineering. arXiv preprint arXiv:2406.12334, 2024.

Gemma 3 Technical Report. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. arXiv preprint arXiv:2310.02226, 2023.

Bryan Guan, Tanya Roosta, Peyman Passban, and Mehdi Rezagholizadeh. The order effect: Investigating prompt sensitivity to input order in llms. arXiv preprint arXiv:2502.04134, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. Proceedings of the International Conference on Learning Representations (ICLR), 2021.

David Herel and Tomas Mikolov. Thinking tokens for language modeling. arXiv preprint arXiv:2405.08644, 2024.

Bertram Højer, Oliver Jarvis, and Stefan Heinrich. Improving reasoning performance in large language models via representation engineering. arXiv preprint arXiv:2504.19483, 2025.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. arXiv preprint arXiv:2507.00432, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. arXiv preprint arXiv:2403.07974, 2024.

Yoonna Jang, Kisu Yang, and Isabelle Augenstein. Expanding computation spaces of llms at inference time. arXiv preprint arXiv:2509.24884, 2025.

Ming Jiang, Tingting Huang, Biao Guo, Yao Lu, and Feng Zhang. Enhancing robustness in large language models: Prompting for mitigating the impact of irrelevant information. In International Conference on Neural Information Processing, pp. 207–222. Springer, 2024.

Prannay Kaul, Chengcheng Ma, Ismail Elezi, and Jiankang Deng. From attention to activation: Unravelling the enigmas of large language models. arXiv preprint arXiv:2410.17174, 2024.

Amelia Kawasaki, Andrew Davis, and Houssam Abbas. Defending large language models against attacks with residual stream activation analysis. arXiv preprint arXiv:2406.03230, 2024.

Eunki Kim, Sangryul Kim, and James Thorne. Learning to insert [pause] tokens for better reasoning. arXiv preprint arXiv:2506.03616, 2025.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. arXiv preprint arXiv:2307.13702, 2023.

Yuxi Li, Yi Liu, Gelei Deng, Ying Zhang, Wenjia Song, Ling Shi, Kailong Wang, Yuekang Li, Yang Liu, and Haoyu Wang. Glitch tokens in large language models: Categorization taxonomy and effective detection. Proceedings of the ACM on Software Engineering, 1(FSE):2075–2097, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In The Twelfth International Conference on Learning Representations, 2023.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. arXiv preprint arXiv:2311.06668, 2023.

Weize Liu, Yinlong Xu, Hongxia Xu, Jintai Chen, Xuming Hu, and Jian Wu. Unraveling babel: Exploring multilingual activation patterns of llms and their applications. arXiv preprint arXiv:2402.16367, 2024.

Charles London and Varun Kanade. Pause tokens strictly increase the expressivity of constant-depth transformers. arXiv preprint arXiv:2505.21024, 2025.

Yifan Luo, Zhennan Zhou, and Bin Dong. Inversescope: Scalable activation inversion for interpreting large language models. arXiv preprint arXiv:2506.07406, 2025.

Yuqi Luo, Chenyang Song, Xu Han, Yingfa Chen, Chaojun Xiao, Xiaojun Meng, Liqun Deng, Jiansheng Wei, Zhiyuan Liu, and Maosong Sun. Sparsing law: Towards large language models with greater activation sparsity. arXiv preprint arXiv:2411.02335, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In EMNLP, 2018.

Junghyun Min, Minho Lee, Woochul Lee, and Yeonsoo Lee. Punctuation restoration improves structure understanding without supervision. arXiv preprint arXiv:2402.08382, 2024.

Louis Owen, Nilabhra Roy Chowdhury, Abhay Kumar, and Fabian Güra. A refined analysis of massive activations in llms. arXiv preprint arXiv:2503.22329, 2025.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. arXiv preprint arXiv:2404.15758, 2024.

Van-Cuong Pham and Thien Huu Nguyen. Householder pseudo-rotation: A novel approach to activation editing in llms with direction-magnitude perspective. arXiv preprint arXiv:2409.10053, 2024.

Qwen2.5 Technical Report. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Daking Rai and Ziyu Yao. An investigation of neuron activation as a unified lens to explain chain-of-thought eliciting arithmetic reasoning of llms. arXiv preprint arXiv:2406.12288, 2024.

Anton Razzhigaev, Matvey Mikhalchuk, Temurbek Rahmatullaev, Elizaveta Goncharova, Polina Druzhinina, Ivan Oseledets, and Andrey Kuznetsov. Llm-microscope: Uncovering the hidden role of punctuation in context memory of transformers. arXiv preprint arXiv:2502.15007, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In First Conference on Language Modeling, 2024.

Leheng Sheng, An Zhang, Zijian Wu, Weixiang Zhao, Changshuo Shen, Yi Zhang, Xiang Wang, and Tat-Seng Chua. On reasoning strength planning in large reasoning models. arXiv preprint arXiv:2506.08390, 2025a.

Yu Sheng, Linjing Li, and Daniel Dajun Zeng. Learning theorem rationale for improving the mathematical reasoning capability of large language models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pp. 15151–15159, 2025b.

Zeru Shi, Zhenting Wang, Yongye Su, Weidi Luo, Hang Gao, Fan Yang, Ruixiang Tang, and Yongfeng Zhang. Robustness-aware automatic prompt optimization. arXiv preprint arXiv:2412.18196, 2024.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. arXiv preprint arXiv:2402.17762, 2024.

The Llama 3 Herd of Models. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. arXiv preprint arXiv:2308.10248, 2023.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models: Dead, n-gram, positional. arXiv preprint arXiv:2309.04827, 2023.

Yudong Wang, Damai Dai, Zhe Yang, Jingyuan Ma, and Zhifang Sui. Exploring activation patterns of parameters in language models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pp. 25416–25424, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

Yang Zhao, Li Du, Xiao Ding, Kai Xiong, Ting Liu, and Bing Qin. Analyzing the rapid generalization of sft via the perspective of attention head activation patterns. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 16980–16992, 2025.

Andy Zhou, Bo Li, and Haohan Wang. Robust prompt optimization for defending language models against jailbreaking attacks. arXiv preprint arXiv:2401.17263, 2024.

Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. Personality alignment of large language models. arXiv preprint arXiv:2408.11779, 2024.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. Prosa: Assessing and understanding the prompt sensitivity of llms. arXiv preprint arXiv:2410.12405, 2024.

A Related Work

A.1 Meaningless Tokens in LLMs

Recent studies have shown that seemingly meaningless tokens, such as punctuation marks, play a non-trivial role in information propagation and reasoning within large language models (LLMs). For example, (Sun et al., 2024) report that LLMs exhibit large activations in response to separators, periods, or newline characters, suggesting that these tokens can serve as carriers of model biases. Similarly, (Razzhigaev et al., 2025) demonstrate that tokens such as commas act as crucial elements in maintaining contextual memory: removing them significantly degrades performance on context-understanding tasks. (Chauhan et al., 2025) further argue that punctuation may function as attention sinks or assist the memory mechanism, while (Min et al., 2024) highlight its value in semantic construction, enabling models to better capture contextual structure. In addition, (Chadimová et al., 2024) show that substituting certain words with meaningless tokens can mitigate cognitive biases in LLMs. Conversely, (Li et al., 2024) illustrate that meaningless "glitch tokens" can induce misunderstandings, refusals, or irrelevant generations. However, these works primarily examine the effects of individual meaningless tokens, without considering the broader impact of longer meaningless token sequences.

More recently, several studies have explored the role of long meaningless token sequences and reported their surprising positive influence on LLM performance. For instance, (Zhou et al., 2024) find that appending meaningless tokens to the end of prompts can trigger or defend against jailbreak behaviors. (Duan et al., 2025) also search meaningless tokens to improve model's performance.Similarly, (Shi et al., 2024) show that adding long meaningless sequences after a sentence can improve model performance on certain tasks. (Pfau et al., 2024), (Duan et al., 2025) and (London & Kanade, 2025) report that substituting meaningful tokens with filler-like tokens (e.g., '. . . ') in the training data preserves the model's ability to solve questions, suggesting that even without meaningful tokens the model can perform implicit computation. Meanwhile, there are also some methods to improve the reasoning performance of LLMs (Dhanraj & Eliasmith, 2025; Højer et al., 2025; Sheng et al., 2025b). (Herel & Mikolov, 2024; London & Kanade, 2025; Kim et al., 2025; Goyal et al., 2023) analyze the effectiveness of thinking tokens which are meaningless before fine-tuning. Despite these empirical findings and methods, there is still a lack of systematic analysis explaining why meaningless tokens, especially in longer sequences, can play such a counterintuitive yet beneficial role in shaping LLM reasoning behavior.

A.2 Activations Analysis in LLMs

Activation analysis is a popular method for explaining the mechanics of LLMs (Wang et al., 2025; Kawasaki et al., 2024; Pham & Nguyen, 2024; Rai & Yao, 2024). (Owen et al., 2025) supplement (Sun et al., 2024) by analyzing the activations after MLP to study how massive values influence bias and large attention. (Wang et al., 2025) test hidden states across all layers to examine the importance of parameters in different layers. (Zhao et al., 2025) use activations to determine whether an attention head is activated after training. (Kaul et al., 2024) analyze attention activations and find that almost all activations focus on the first tokens; they also analyze high activations in the FFN. (Luo et al., 2024) systematically study the magnitude law and influencing factors of activation sparsity in decoder Transformer architectures, showing that different activation functions (ReLU vs. SiLU) lead to drastically different sparsity trends during training. In (Liu et al., 2024), activation refers to the output behavior of the expert selector: instead of a single neuron activating, the analysis investigates which expert module each token is routed to. (Turner et al., 2023) propose steering middle-layer activations to improve model outputs. (Voita et al., 2023) uses OPT model to do analysis for FFM neurons. (Luo et al., 2025), using activations to understand the semantic information in LLMs. In (Liu et al., 2023) shows the control derived from demonstrations or task-conditioning vectors. In (Sheng et al., 2025a), it studies a directional signal for reasoning budget control. In (Zhu et al., 2024), which focuses on personality/preference alignment through behavioral patterns and head selection. However, most papers analyze activations using activation scores, hidden states, or broader definitions of activation. Few works directly examine the activations right after the non-linear activation functions in the MLP.

B LIMITATIONS

Different meaningless tokens can lead to different degrees of performance change, and some choices may be ineffective or harmful. We attribute this variation to differences in the induced transformation, but the underlying reason why particular token types behave differently remains unclear. Our analysis identifies activation redistribution as one plausible mechanism supported by our observations, rather than an exhaustive account of all effects caused by token insertion. We also assume that the model can identify the inserted tokens in the first layer. Therefore, we focus on their impact on meaningful tokens and how this interaction influences model performance, while not modeling the inserted tokens themselves in later semantic computation. Future work can investigate these additional mechanisms and explicitly analyze the representations of the inserted tokens.

C DISCUSSION: WHY ACTIVATION REDISTRIBUTION ENHANCES LLM REASONING PERFORMANCE

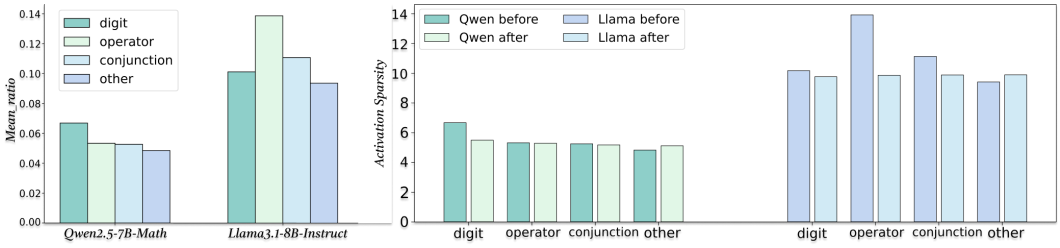


Figure 6: The left figure is percentage of near-zero activations across the four token types in the Math-500 dataset. Right figure is average activation sparsity for each token category before and after the affine transformation.

We provide one possible explanation for why redistributing near-zero activations can improve the reasoning performance of LLMs. We categorize all tokens in Math-500 into four classes: digits, conjunctions, operators, and other tokens. For each class, we compute the average proportion of activations falling within the near-zero range, which reflects how many dimensions of the hidden representation remain nearly inactive. The results are presented in the left sub-figure of Figure 6. As shown, other tokens exhibit the lowest near-zero proportion, while digits, operators, and conjunctions show substantially higher proportions. This is an empirical characterization, not an explanation of why these categories develop this activation pattern; understanding its origin is beyond the scope of this work. Our observation is consistent with the findings of (Huan et al., 2025), which highlight the increasing importance of conjunctions after reinforcement learning, and also aligns with the recognized role of digits and operators in reasoning tasks such as mathematics and coding. We then compute the average activation sparsity for each token category before and after the affine transformation induced by inserting meaningless tokens on Qwen2.5-Math-7B and Llama-3.1-8B-Instruct. As shown in the right sub-figure of Figure 6, the activation sparsity of digit, operator, and conjunction tokens consistently decreases, whereas the sparsity of other tokens increases. This contrasting trend indicates that the transformation selectively strengthens representations of tokens that are critical for reasoning rather than uniformly amplifying all activations.

This section analyzes the activation redistribution produced by the explicit meaningless-token intervention, not ARM. It connects to the theory in Section 3: Sections 3.1–3.2 explain how the induced scaling and additive terms can increase gate-output variance and broaden the activation distribution, while the present analysis shows which token categories contain more near-zero activations and therefore can benefit most from that redistribution. The evidence supports activation redistribution as one plausible route to improved reasoning, without implying that it is the only mechanism involved. For factual and other non-reasoning tasks, our results show substantially smaller changes, with performance generally preserved rather than systematically degraded; whether other tasks with similar under-activation patterns could benefit remains an open question.

D TIME COMPLEXITY

In this section, we will analyze the time complexity of our method in the MLP. In the first layer's MLP, we have batch size B, sequence length S, feed forward dimensions D_f, model dimension D_{model}. For MLP, the module of time complexity contains gate project, up project and down project. The time complexity of each module is $O(2BSD_fD_{model})$, thus the total of MLP is:

$$\mathrm{T}_{mlp} = O(BSD_fD_{model}), \tag{14}$$

For ARM module, the operation contains: calculating MAD, comparing threshold, calculating proportion p, selecting elements that need to be changed. The time complexity of all above operations is $O(BSD_f)$. So the time complexity of ARM is:

$$\mathrm{T}_{ARM} = O(BSD_f), \tag{15}$$

The comparison between the time complexity of ARM and MLP is $\frac{1}{D_{model}}$. When D_{model} equals to 4096. This proportion value is approximately $\frac{1}{2*4096} \approx 1.2 \times 10^{-4}$ at the level of one ten-thousandth. Therefore, we believe that the time complexity of ARM can be disregarded in MLP layer.

E PROOF

E.1 HOW THE SCALING FACTOR CAN INCREASE VARIANCE

Assumption D.1 *In LLMs, RMSNorm uses $\varepsilon > 0$; hence $J_q(x_0)$ is bounded and $\|x_0\| \geq \varepsilon$*

We make the population and local-perturbation notation explicit here. Let $\mathcal{P}$ be the population of aligned meaningful-token positions across evaluation examples, and let all variances and covariances below be taken over $\mathcal{P}$. For every operating-point scaling factor λ, we have:

$$x_t(\lambda) = r_t + \lambda U A_t, \qquad y_t(\lambda) = \mathrm{RMSNorm}(x_t(\lambda)), \qquad z_{t,j}(\lambda) = w_j^\top y_t(\lambda), \tag{16}$$

Here r_t is the residual stream, A_t is the original attention output for query token t, U is the attention output-projection matrix, w_j is the weight vector for the j-th MLP gate dimension, and juxtaposition denotes matrix multiplication. In the observed meaningless-token intervention, the operating-point coefficient is generally $\lambda < 1$. For every small local perturbation $\Delta\lambda$ around that operating point, we have:

$$z_{t,j}(\lambda + \Delta\lambda) \approx z_{t,j}(\lambda) + g_{t,j}(\lambda)\Delta\lambda, \qquad g_{t,j}(\lambda) = w_j^\top J_q(x_t(\lambda)) U A_t, \tag{17}$$

where $\Delta\lambda$ is the local change used in the Taylor expansion; it is not the absolute value of the operating-point coefficient λ and need not be large even when $\lambda < 1$. For readability, we suppress the query-token index t in the variance derivation below. For a fixed gate dimension j, the variance change over $\mathcal{P}$ is:

$$\begin{aligned}
\Delta\mathrm{Var}_j &\triangleq \mathrm{Var}\big[z_j(\lambda + \Delta\lambda)\big] - \mathrm{Var}\big[z_j(\lambda)\big] \\
&\approx 2\mathrm{Cov}\big(z_j(\lambda), g_j(\lambda)\big)\,\Delta\lambda + \mathrm{Var}\big[g_j(\lambda)\big]\,(\Delta\lambda)^2. \\
\Delta\mathrm{Var}_j &\;\geq\; -2\big|\mathrm{Cov}(z_j, g_j)\big|\,|\Delta\lambda| + \mathrm{Var}(g_j)\,(\Delta\lambda)^2. \\
|\Delta\lambda| &\;>\; \frac{2\big|\mathrm{Cov}(z_j, g_j)\big|}{\mathrm{Var}(g_j)}
\end{aligned}$$

Including the third-order remainder gives the sufficient local bound:

$$\Delta\mathrm{Var}_j \;\geq\; -2\big|\mathrm{Cov}(z_j, g_j)\big|\,\big|\Delta\lambda\big| + A\,(\Delta\lambda)^2 - \frac{K}{6}\,\big|\Delta\lambda\big|^3. \tag{18}$$

where $A = \mathrm{Var}[g_j(\lambda)]$ is the positive quadratic coefficient and K bounds the local third-order remainder. Thus, a sufficient range is:

$$\frac{2\big|\mathrm{Cov}(z_j, g_j)\big|}{\mathrm{Var}(g_j)} \leq |\Delta\lambda| \leq \frac{3\mathrm{Var}(g_j)}{\mathrm{K}}. \tag{19}$$

If $|\Delta\lambda|$ lies in this range and the local approximation is valid, then $\Delta\mathrm{Var}_j > 0$. This is a sufficient, not necessary, condition for a variance increase. In particular, it constrains the size of the local perturbation $\Delta\lambda$ around the operating point; it does not require the absolute scaling factor λ itself to be greater than one or for $\Delta\lambda$ to be large.

E.2 How the Bias Factor Can Increase Variance

Assumption D.2 *For the local analytical approximation, the centered additive perturbation Σ_σ is bounded, zero-mean, and sampled independently of the original attention output.*

This independence assumption is introduced only to obtain a tractable first-order covariance calculation. It is not a literal statistical model of the structured perturbation produced by meaningless-token insertion: the actual contribution remains $\Sigma_{\sigma,t}$ from Equation 4 and can be related to the original attention output. The approximation isolates the variance contribution of the additive term and should not be interpreted as an exact description of its distribution.

Assumption D.3 *In LLM's high dimensions, the additive perturbation has a nonzero tangential component and $w_j^\top J_q(x_0)W \neq 0$.* Under these assumptions, we have:

$$\mathrm{Var}[z_j] \approx e_j^\top W_{gate} J_q(x_0) W \,\mathrm{Cov}(\Sigma_\sigma) W^\top J_q(x_0)^\top W_{gate}^\top e_j \tag{20}$$

Thus, under the stated approximation and non-degeneracy condition, the additive term contributes a positive variance component, $\Delta V_j > 0$.

F More Analysis

F.1 The Impact of the Length of the Inserted Sequence

In this section, we analyze the relationship between the length of the inserted sequence and model performance. We evaluate five models on MATH-500 while varying the number of inserted tokens from 0 to 70. The results are shown in Figure 7. When the inserted sequence is relatively short, the models outperform the baseline in the reported settings, although accuracy fluctuates. When too many tokens are inserted, performance drops sharply. This pattern is consistent with the accumulation of their contribution to the attention output (as shown in Equation 3); once the perturbation is no longer small, it may alter the model's internal representations and degrade performance.

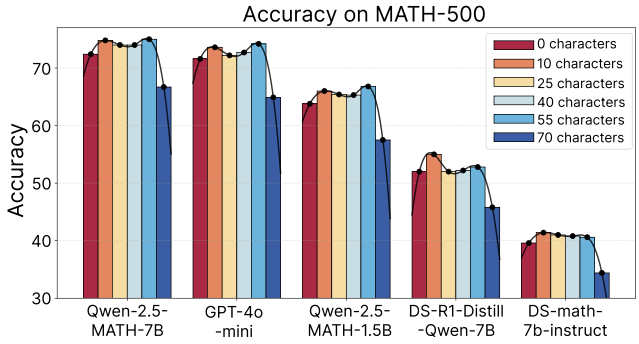


Figure 7: The relationship between the length of inserting tokens and the performance of models.

F.2 Empirical Check of the Sufficient Local Condition

To examine whether the sufficient local condition in Equation (16) is compatible with the empirical perturbations induced by meaningless-token insertion, we randomly selected 100 examples from the MATH-500 dataset and evaluated them using Qwen2.5-Math-1.5B. For each resulting perturbation instance, we computed the local scaling perturbation $\Delta\lambda$ and checked whether it satisfied the sufficient condition derived in Equation (16). Among the evaluated perturbation instances, 96.4% satisfied this condition.

This verification concerns the local perturbation $\Delta\lambda$ around the operating point, rather than the absolute scaling coefficient λ. Therefore, the result is compatible with the observation that λ is typically smaller than one after meaningless-token insertion. The purpose of this analysis is to assess the applicability of the local variance analysis to the empirical intervention; it is not a necessary-condition test and does not constitute an additional performance evaluation.

F.3 The Impact of Insertion Position

In the previous section, we observed that inserting meaningless tokens between the system prompt and the question improves model performance in the reported setting. In this section, we investigate the effect of insertion position. Specifically, we consider four settings: ❶ the beginning of the system prompt, ❷ between the system prompt and the question, ❸ the end of the input, and ❹ a random position within the input. The results are reported in Table 4. In the reported comparison, only insertion between the system prompt and the question yields performance gains. Appending tokens to the end of the input is associated with repeated output and zero accuracy in this setting. Inserting tokens at random positions may disrupt the original semantic structure, while inserting them at the beginning changes the system-prompt representations and introduces extra terms as shown in Equation 3. These observations motivate placing the inserted sequence between the system prompt and the question, while the underlying position dependence remains to be studied in more detail.

Table 4: Performance on Math-500 and AIME 2022-2024 after inserting meaningless tokens in different positions.

Model	**Math-500 (Pass@1)**	**AIME 2022-2024** (Pass@1)
w/o meaningless tokens	72.4	23.1
position ❶	69.6	21.1
position ❷	75.0	23.3
position ❸	0.0	0.0
position ❹	51.2	21.1

F.4 Statistically Significant Differences

In Table 5, we conduct an F-test on the Math-500 benchmark using Qwen2.5-Math-7B, Llama3.1-8B-Instruct, and Gemma3-4B-it to quantify the effect of inserting meaningless tokens. The reported Global F measures the multiplicative change in post-activation variance before and after the affine transformation induced by meaningless tokens. Across all three models, the p-values are extremely small ($p \approx 10^{-16}$), indicating that the observed variance changes are statistically significant. Notably, the Global F values reveal a substantial increase in activation variance after inserting meaningless tokens, particularly for Qwen2.5-Math-7B and Gemma3-4B-it. This consistent variance amplification supports our hypothesis that meaningless tokens induce an affine transformation that redistributes activations away from near-zero regions, thereby strengthening under-activated dimensions and increasing overall representational expressivity.

F.5 The impact of Activation Functions.

Among the three models we analyzed, Qwen2.5-7B-Math and Llama3.1-8B-Instruct use the SiLU activation function, while Gemma3-4B-it uses GeLU. As shown in Figure 4, regardless of the activation function type, we observe similar changes in activation distributions under the affine transformation described in our analysis.

We replaced the activation functions of Qwen2.5-7B-Math, Llama3.1-8B-Instruct with GeLU, and Gemma3-4B-it with SiLU in the first layer respectively, and applied F tests (a statistical measure used to analyze differences in activation variance) to evaluate the sensitivity of activation distributions under different activation functions in Table 6. The p-value of F-test is all similar to 1.11e-16. The experimental results show that both activation functions (SiLU and GeLU) exhibit similar sensitivity to the affine transformation induced by the insertion of meaningless tokens.

F.6 The Impact of Token Type

In this section, we examine the influence of inserting different types of meaningless tokens on reasoning tasks. In our experiments, we insert varying lengths of slashes (“/”) and question marks (“?”) into the inputs and select the best-performing configuration from each set. As shown in Table 7, different types of meaningless tokens produce varying impacts on LLM performance, and no single unified pattern emerges. We attribute this to the fact that different token types carry distinct representational

Table 5: Statistical comparison with and without meaningless tokens (Mless).

Model	Global F	p-value	w/o Mless	w/ Mless
Qwen2.5-Math-7B	31.66	1.11×10^{-16}	0.00108	0.03413
Llama3.1-8B-Instruct	1.30	1.11×10^{-16}	0.00414	0.00539
Gemma3-4B-it	40.00	1.11×10^{-16}	0.00412	0.16482

Table 6: Comparison of SiLU and GeLU activations with and without meaningless tokens across different models.

Model	w/o Mless (SiLU)	w/ Mless (SiLU)	w/o Mless (GeLU)	w/ Mless (GeLU)
Qwen-2.5-Math-7B	0.001078	0.034134	0.000715	0.037110
Llama-3.1-8B-Instruct	0.004135	0.005387	0.004084	0.005505
Gemma-3-4B-It	0.044991	0.163587	0.038325	0.164823

values, leading to different effects of attention during the transformation. Moreover, the sensitivity of individual questions to such transformations also varies. Consequently, the impact of meaningless tokens differs across tasks and models.

Table 7: Accuracy of LLM on two mathematical reasoning datasets with inserting different kinds of meaningless tokens.

Methods	MATH-500			AIME 2022-2024		
	w/o Mless	\	?	w/o Mless	\	?
Qwen2.5-Math-1.5b	63.6	**66.8**	58.2	14.4	**18.8**	16.1
Qwen2.5-Math-7b	72.4	**75.0**	69.6	23.3	**24.4**	22.2
DeepSeek-R1-Distill-Qwen-7B	52.0	**55.0**	53.6	3.3	3.3	**4.4**
DeepSeek-Math-7b-instruct	39.6	41.4	**43.4**	7.8	12.2	**12.5**
Llama-3.1-8B-Instruct	35.4	**36.6**	34.2	11.1	7.8	**13.3**
Qwen-2.5-32B-Instruct	80.8	81.0	**81.6**	18.9	20.0	**21.1**

F.7 WHY WE ONLY ANALYZE FIRST LAYER

In this section, we explain why our analysis and redistribution of activations focus on the first layer. As shown in Figure 8 and Figure 9, we present average attention weights for later tokens in Layers 1, 2, and 3 of Qwen2.5-7B-Math, together with additional visualizations for deeper layers. The first layer exhibits the clearest and most consistent pattern in these visualizations: after inserting meaningless tokens, the average attention weights decrease to a noticeable extent, suggesting that meaningless tokens alter the initial allocation of attention. In contrast, Layers 2 and 3 do not display the same regularity—the average attention weights with and without meaningless tokens show no systematic relationship. In layers beyond the third, the pattern is less obvious. We therefore do not claim that later layers undergo the same affine transformation; their responses may reflect the combined effects of residual connections and subsequent semantic processing. We hypothesize that, beyond the first layer, the model has already integrated and mixed substantial semantic information through residual connections, making the direct effect of meaningless tokens on average attention weights less distinguishable. This motivates focusing on the first layer while leaving the detailed roles of later layers for future work.

To prove the effectiveness of first layer, In the Table 8, we report the performance of the Qwen2.5-Math-1.5B model on the Math-500 when applying ARM every 1, 3, 5, and 8 layers, respectively. The results show that when ARM is applied to multiple layers including the first layer, the model exhibits some improvement, though the gains are smaller than when ARM is applied only to the first layer. However, when the first layer is exclude, the model's output garbled text, this will severely hurt the model's performance. In addition, we supply ARM to a single layer selected from the early, middle, and final stages of the transformer. The results are shown in Table 9. We observe that applying ARM to any single layer yields a positive effect on model performance. We believe this is because, in deeper layers, a single-layer modification can help smooth the activation distribution, producing a

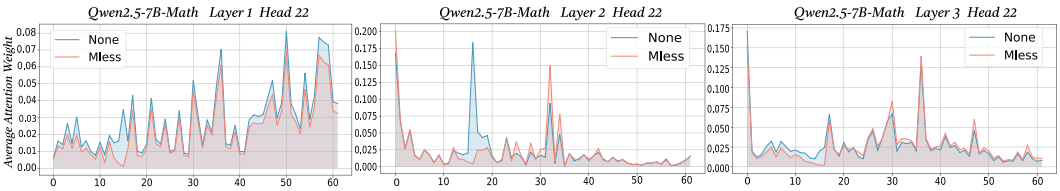


Figure 8: Average attention weights for later tokens in Layers 1 to 3 of Qwen2.5-7B-Math.

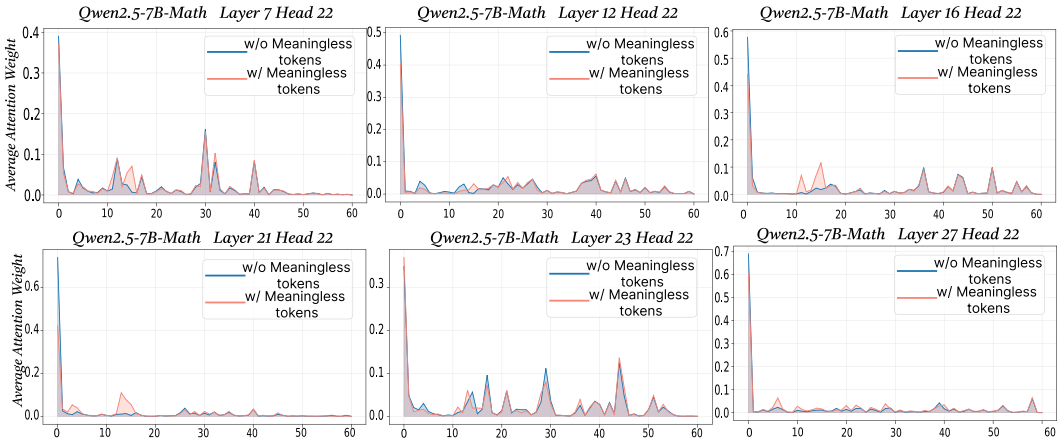


Figure 9: Average attention weights in layers beyond the third layer. After adding meaningless tokens, subsequent layers do not show the same regular attention pattern as the first layer.

Table 8: Using ARM in multiple layers.

Configuration	**Answer / Pass@1**
Every Layer	*garbled text*
Baseline	63.8
Every 3 Layers (w/ Layer1)	64.0
Every 3 Layers (w/o Layer1)	*garbled text*
Every 5 Layers (w/ Layer1)	64.8
Every 5 Layers (w/o Layer1)	*garbled text*
Every 8 Layers (w/ Layer1)	66.0
Every 8 Layers (w/o Layer1)	*garbled text*
ARM	**67.0**

mild regularizing effect. However, applying ARM to multiple layers simultaneously, while keeping the initial distribution unchanged, leads to over modification, which may cause semantic collapse. These findings further demonstrate both the validity of ARM that it does not cause semantic collapse when used in a single layer and the soundness of applying ARM specifically to the first layer.

Table 9: Using ARM at different single layers.

Layer	**Pass@1**
Baseline	63.8
6 Layer	65.0
11 Layer	65.6
21 Layer	66.0
Last Layer	65.6
Ours	**67.0**

F.8 Propagation of the First-Layer Affine Transformation

In this section, using Qwen2.5-Math-7B, we perform an F-test at every layer to compare the variance of post-activation values before and after the affine perturbation induced by meaningless tokens. The results in Table 10 show a clear layerwise pattern: variance increases sharply in early layers, decreases through middle layers, and rises again in the final layer, where it exceeds the no-token baseline. The p-values are approximately 1.11e-16 for all layers, indicating a highly significant distributional change after meaningless-token insertion.

Layer	w/o Mless	w/ Mless	Layer	w/o Mless	w/ Mless	Layer	w/o Mless	w/ Mless	Layer	w/o Mless	w/ Mless
1st Layer	0.0010780497	0.034134176	8th Layer	0.06583081	0.020619715	15th Layer	0.08307721	0.012819506	22nd Layer	0.21591395	0.048060764
2nd Layer	0.0011195851	0.0055994517	9th Layer	0.07209223	0.021961281	16th Layer	0.09073611	0.013984551	23rd Layer	0.23913541	0.05131975
3rd Layer	0.0016769666	0.0042314352	10th Layer	0.019369887	0.03592288	17th Layer	0.09650683	0.012532488	24th Layer	0.24144447	0.056764927
4th Layer	0.01282291	0.5163366	11th Layer	0.05951471	0.013570791	18th Layer	0.10699968	0.01702663	25th Layer	0.34542912	0.073193744
5th Layer	0.006588933	0.1560966	12th Layer	0.069356665	0.03826137	19th Layer	0.124931894	0.02089952	26th Layer	0.3348005	0.14499877
6th Layer	0.002661827	0.010141362	13th Layer	0.07368733	0.02219496	20th Layer	0.124931894	0.02089952	27th Layer	0.51737994	0.62529993
7th Layer	0.04099468	0.06684371	14th Layer	0.08096493	0.017560342	21st Layer	0.18799423	0.03685474	Last Layer	1.2712328	2.4533575

Table 10: Layer-wise comparison with and without Mless.

F.9 Effect of Repeated Meaningful Tokens

In this section, we investigate whether adding repeated tokens can produce an effect similar to that of meaningless tokens. Specifically, we insert a long sequence of repeated tokens that are semantically irrelevant to the question. For example, we add 55 repetitions of “he” between the system prompt and the question. The results, shown on the left of Figure 10, indicate that these repeated but irrelevant tokens can also improve model performance in the evaluated setting. To better understand this effect, we visualize the average attention weights after insertion on the right of Figure 10. The activation changes induced by the repeated tokens resemble those caused by meaningless tokens, and the inserted tokens receive similar attention patterns. Taken together, these findings suggest that repeated tokens inserted at appropriate positions may be treated as irrelevant by the model and may be associated with a redistribution of MLP activations. We do not claim that this pattern applies uniformly across token types, positions, or tasks.

F.10 Why a Random Sentence Does Not Help

When additional tokens are inserted into a sentence, both the attention weights and the resulting attention outputs exhibit consistent patterns: the weights assigned to the original tokens decrease, while the attention outputs gain additional values from the inserted tokens. In this section, we analyze why adding repeated tokens can enhance the performance of LLMs, whereas inserting random or unrelated sentences can have a detrimental effect. The results are shown in Figure 11. We observe that the attention weights associated with the random sentence are highly diverse, and their corresponding value vectors also differ substantially. In contrast, the repeated meaningless tokens exhibit more uniform attention weights and nearly identical value vectors. Consequently, compared with repeated meaningless tokens, a random sentence introduces not only numerical fluctuations

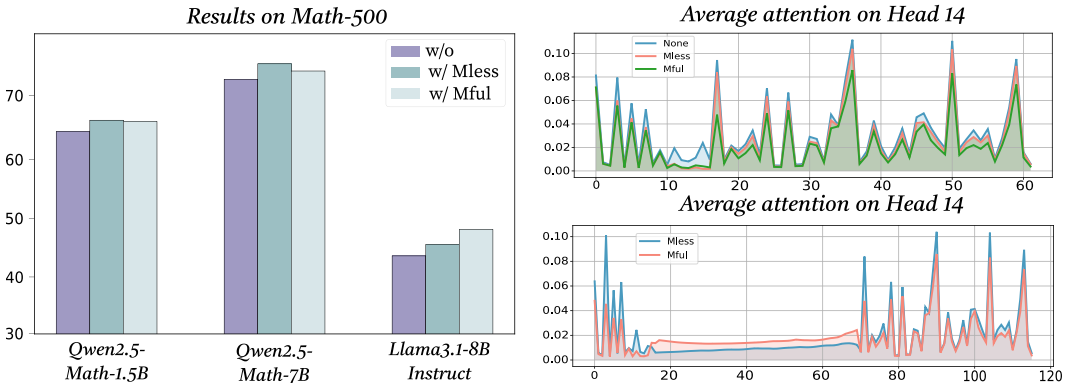


Figure 10: The left panel illustrates a comparison between adding repeated meaningful tokens and meaningless tokens, while the right panel presents the average attention weights resulting from the addition of meaningful and meaningless tokens.

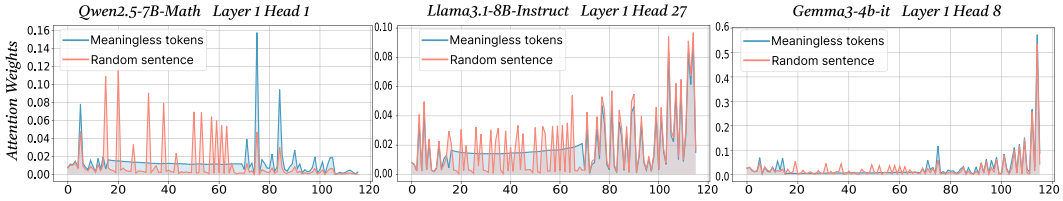


Figure 11: The average attention weights of adding meaningless tokens and random sentence.

but also a pronounced directional shift in the attention outputs—one that carries additional semantic information. The formula of RMSNorm is:

$$\mathrm{RMSNorm}(x) = \gamma \odot \frac{x}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}}, \tag{21}$$

where γ is a learnable rescaling vector and ϵ ensures numerical stability. For repeated meaningless tokens, the effect manifests as a small and uniform directional bias on the input to RMSNorm, producing only a minor numerical perturbation in its output. In contrast, inserting a random sentence introduces high-rank and structured semantic signals that RMSNorm cannot simply absorb. This leads to systematic shifts in the output direction and subspace, thereby altering the model's internal semantic representations.

F.11 Analysis of Attention Weights and Values

We follow the settings in Figure 2 and report the post softmax attention weights from subsequent tokens. The results in!Table 11. We randomly choose the five meaningless tokens and show their maximum and mean attention weights in up table and other tokens have similar patterns with them, while bottom table reports the corresponding statistics for the five meaningful tokens. Compared with meaningful tokens, meaningless tokens exhibit similar maximum and mean attention weights, this demonstrates the rationale behind our use of average attention weights.

F.12 The Optimal Hyperparameter Range

In this section, we investigate how the hyperparameters—the percentage $\boldsymbol{p}$ and the width of the near-zero range $\boldsymbol{c}$—influence model performance on Math-500 when using Qwen2.5-1.5B-Math. The results are summarized in Figure 12. As the figure illustrates, accuracy does not change monotonically with either p or c; the best-performing settings occur within specific regions of the evaluated parameter

Setting	Statistic	Token 1	Token 2	Token 3	Token 4	Token 5
w/o Mless	Average	0.0003	0.0003	0.0003	0.0003	0.0003
	Max	0.0043	0.0038	0.0054	0.0041	0.0031
w/ Mless	Average	0.0525	0.0215	0.0175	0.0338	0.0174
	Max	0.8891	0.2737	0.2925	0.2324	0.2218

Table 11: Token-level attention weights w/. and w/o meaningless tokens.

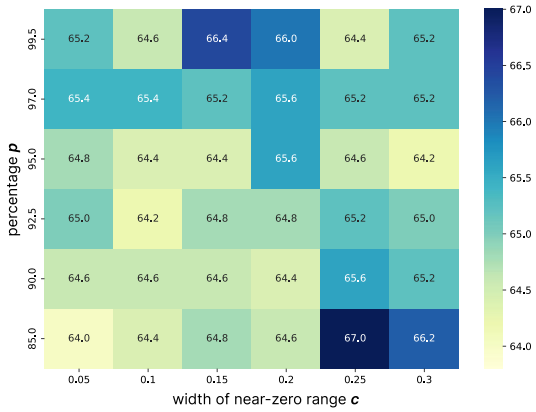


Figure 12: This figure illustrates how accuracy varies with changes in the parameters p and c.

space. Because these settings were not selected using a held-out validation set, this figure should be interpreted as a sensitivity analysis for the reported model and task, rather than as evidence of cross-task hyperparameter robustness. The results indicate that performance depends jointly on p and c, and that poorly chosen values can distort the activation distribution. They do not by themselves establish that a single setting transfers to unseen models or tasks.

G More Experiments

G.1 Terminal Rate of Adding Meaningless Tokens

In this section, we calculate the termination rate of different models in Table 1 on Math-500 and AIME22–24. The results are reported in Table 12. The termination rates are broadly similar with and without meaningless tokens in the evaluated settings, suggesting that the observed performance differences are not explained simply by shorter generations.

Model	Math 500 w/o Mless	Math 500 w/ Mless	AIME w/o Mless	AIME w/ Mless
Qwen-2.5-Math-1.5B	93.00	94.80	75.56	78.89
Qwen-2.5-Math-7B	96.00	92.40	78.89	73.33
DS-R1-Qwen-7B	99.60	99.80	92.20	91.11
DS-Math-7B-Instruct	96.20	96.20	96.67	90.00
Llama-3.1-8B-Instruct	93.20	90.00	78.89	78.89

Table 12: Termination rate of different models w/. and w/o meaningless tokens

G.2 Testing Inserting Meaningless Tokens in Different Setting.

In this section, we evaluate model performance after inserting meaningless tokens under the decoding settings specified for each result. Specifically, we report pass@4, pass@8, and pass@16 results in Table Table 13, as well as performance under temperature=1 decoding in Table Table 14. Across the

evaluated models and tasks, the results show performance improvements after inserting meaningless tokens, with the magnitude varying by setting.

Table 13: Accuracy of inserting meaningless tokens on two mathematical reasoning datasets with and without meaningless tokens.

Methods	MATH-500			AIME 2022–2024		
	pass@4	pass@8	pass@16	pass@4	pass@8	pass@16
Qwen-2.5-Math-1.5B w/o Mless	48.0	59.0	66.0	6.7	12.2	16.7
Qwen-2.5-Math-1.5B w/ Mless	50.2	60.2	67.6	8.9	15.6	18.9
Qwen-2.5-Math-7B w/o Mless	47.8	57.0	65.4	6.7	13.3	23.3
Qwen-2.5-Math-7B w/ Mless	53.2	62.2	69.0	13.3	21.1	26.7
DeepSeek-R1-Distill-Qwen-7B w/o Mless	18.2	32.0	44.6	2.2	4.4	6.7
DeepSeek-R1-Distill-Qwen-7B w/ Mless	20.6	30.8	46.8	3.3	5.6	8.9
DeepSeek-Math-7B-Instruct w/o Mless	56.4	69.2	80.2	16.7	32.2	47.8
DeepSeek-Math-7B-Instruct w/ Mless	56.2	70.8	81.6	20.0	34.4	54.4
Llama-3.1-8B-Instruct w/o Mless	51.0	61.8	69.8	5.6	10.0	22.2
Llama-3.1-8B-Instruct w/ Mless	51.8	61.8	72.2	10.0	14.4	27.8

Table 14: Accuracy under temperature=1 on two mathematical reasoning datasets with and without meaningless tokens.

Methods	MATH-500		AIME 2022–2024	
	w/o Mless	w/ Mless	w/o Mless	w/ Mless
Qwen-2.5-Math-1.5B	63.8 ± 0.43	66.8 ± 0.05	14.4 ± 0.00	16.3 ± 0.60
Qwen-2.5-Math-7B	72.4 ± 0.17	75.0 ± 0.27	22.6 ± 0.60	24.9 ± 0.93
DeepSeek-R1-Distill-Qwen-7B	43.0 ± 0.16	44.8 ± 1.36	2.2 ± 0.00	3.3 ± 0.00
DeepSeek-Math-7B-Instruct	39.6 ± 0.82	43.4 ± 2.73	7.8 ± 0.00	12.8 ± 0.43
Llama-3.1-8B-Instruct	43.4 ± 0.54	43.8 ± 1.20	7.9 ± 0.16	9.6 ± 0.60

G.3 Other Results of ARM on Different Settings

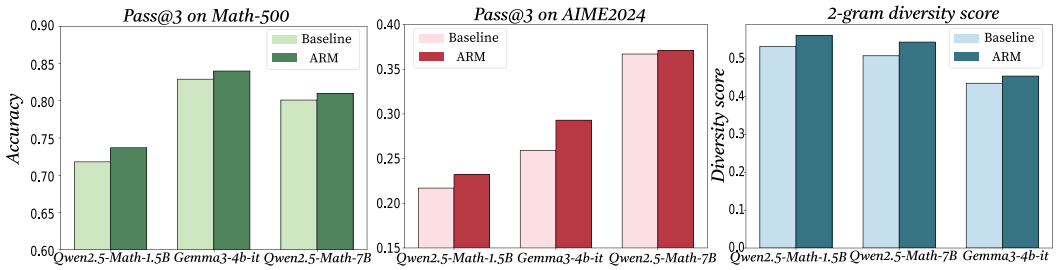


Figure 13: The first two figures show pass@3 on Math-500 and AIME 2022-2024 for three models with and without ARM, and the last shows their 2-gram diversity under both conditions.

As discussed earlier, we hypothesize that redistributing activations enables the model to explore the reasoning space more effectively. To test this hypothesis, we evaluate the model's pass@3 performance on the Math-500 and AIME 2022-2024 benchmarks as well as its 2-gram diversity. As shown in Figure 13, applying activation redistribution consistently yields higher pass@3 scores compared to the baselines on both tasks. In addition, the 2-gram diversity under ARM is also greater than that without ARM. These findings indicate that activation redistribution not only improves the likelihood of arriving at correct solutions within multiple samples but also promotes more diverse reasoning paths. This dual effect suggests that ARM enhances both the effectiveness and the breadth of the model's internal reasoning processes, reinforcing our hypothesis that carefully manipulating internal activations can expand a model's reasoning capacity without additional training or parameter growth.

From the experimental results in Table 15, we observe that, under the Pass@4, Pass@8, and Pass@16 metrics, the models equipped with ARM outperform the original models in the reported comparisons. The gains persist across the listed sample counts, although their magnitude varies by model and metric.

Table 15: Pass@4, Pass@8, and Pass@16 results on MATH-500 and AIME 2022–2024.

Methods	MATH-500			AIME 2022–2024		
	pass@4	pass@8	pass@16	pass@4	pass@8	pass@16
Qwen-2.5-Math-1.5B	52.6	61.2	68.8	21.1	36.7	52.2
Qwen-2.5-Math-1.5B (ARM)	54.8	63.8	69.0	33.3	43.3	55.6
Qwen-2.5-7B-Instruct	56.6	58.8	63.2	24.4	33.3	54.4
Qwen-2.5-7B-Instruct (ARM)	55.8	60.0	72.2	27.8	42.2	67.8
Llama-3.1-8B-Instruct	60.2	67.0	74.2	22.2	34.4	50.0
Llama-3.1-8B-Instruct (ARM)	60.8	67.4	76.2	18.9	37.8	55.6
Gemma-3-4B-It	49.2	51.4	55.8	33.3	37.8	46.7
Gemma-3-4B-It (ARM)	48.2	53.8	57.2	35.6	40.0	48.9

Table 16: Complete results of several models on non-reasoning tasks.

Model	Setting	GSM8K	ARC-E	ARC-C	MMLU	BoolQ	HellaSwag	OpenBookQA
Qwen2.5 Math-1.5B	Baseline	78.0	39.3	35.0	32.1	32.6	39.1	42.0
	ARM	78.6	39.3	35.4	32.1	33.4	39.5	42.4
	Absolute Improvement (pp)	**0.6**↑	0	**0.4**↑	0	**0.8**↑	**0.4**↑	**0.4**↑
Qwen2.5 Math-7B	Baseline	83.8	49.7	47.9	36.9	38.6	46.9	47.6
	ARM	83.8	49.7	47.0	37.5	38.7	47.1	47.9
	Absolute Improvement (pp)	0	0	0	**0.6**↑	**0.1**↑	**0.2**↑	**0.3**↑
Llama3.1 8B-Instruct	Baseline	80.0	46.6	49.0	38.6	43.3	56.8	52.8
	ARM	82.4	47.1	48.7	38.2	43.2	57.3	50.8
	Absolute Improvement (pp)	**2.4**↑	**0.5**↑	**-0.3**↓	**-0.4**↓	**-0.1**↓	**0.5**↑	**-2.0**↓
Gemma3 4b-it	Baseline	86.8	47.1	44.5	33.9	45.0	42.0	41.0
	ARM	86.8	47.1	45.0	34.1	45.2	42.0	42.0
	Absolute Improvement (pp)	0	0	**0.5**↑	**0.2**↑	**0.2**↑	0	**1.0**↑

G.4 Results on non-reasoning tasks

In this section, we present supplementary results on non-reasoning benchmarks, including ARC-C, MMLU, BoolQ, and OpenBookQA, as shown in Table 16. Across all evaluated models, the application of our method yields only marginal variations in performance. For most models and tasks, it either produces slight improvements or maintains parity with the baseline (i.e., vanilla model performance without any inference-time trick), suggesting that the redistribution of activations has little impact when the task primarily requires factual recall or pattern recognition rather than multi-step reasoning. A minor performance drop is observed only on a small subset of tasks with Llama-3.1-8B-Instruct, which we attribute to model-specific characteristics or sensitivity to activation perturbations.These findings indicate that our approach exerts negligible influence on non-reasoning tasks and, in most cases, does not introduce adverse effects on task accuracy. This observation further supports our central claim: the benefits of activation redistribution are **most pronounced in reasoning-oriented scenarios**, while **in non-reasoning settings the method remains stable and does not compromise the model's inherent ability** to answer factual or knowledge-intensive questions.

G.5 Results on Base Model

In this section, we evaluate the effect of applying ARM to base models and report their performance on Math-500 and AIME 2022-2024 using Qwen2.5-1.5B, Qwen2.5-7B, and Qwen2.5-32B. Since these models achieve accuracy above 5%, we consider them capable of tackling these tasks. In contrast, models such as Llama3.1-8B and Gemma3-4B-PT exhibit poor performance and are therefore excluded from the evaluation. The results in Table 17 show that incorporating ARM into the MLP layers of base models yields measurable performance gains on reasoning tasks, although the improvements are generally smaller than those observed for reasoning, oriented models. We attribute this gap to the weaker inherent reasoning abilities of base models. While activation redistribution can still enhance their internal representations, it may not strongly affect how they process key numerical or symbolic elements, such as digits and operators, compared with models trained specifically for reasoning.

Table 17: Performance on Math-500 and AIME 2022-2024 after incorporating ARM into the MLP in non-reasoning model.

Model	Setting	Math-500 (Pass@1)	AIME 2022-2024 (Pass@1)
Qwen2.5-1.5B	Baseline	67.8	14.4
	ARM	68.2	14.4
	Absolute Improvement (pp)	**0.4**↑	0
Qwen2.5-7B	Baseline	50.4	15.6
	ARM	50.6	16.7
	Absolute Improvement (pp)	**0.2**↑	**1.1**↑
Qwen2.5-32B	Baseline	77.2	27.8
	ARM	77.4	28.9
	Absolute Improvement (pp)	**0.2**↑	**1.1**↑

Table 18: Performance on Math-500 and AIME 2022-2024 after incorporating ARM into the MLP.

Model	Setting	Math-500 (Pass@1)	AIME 2022-2024 (Pass@1)
Qwen2.5 Math-1.5B	Baseline	63.8	14.4
	ARM	67.8	**18.9**
	Best-of-N(N=5)	69.4	14.4
	Best-of-N + ARM	**71.2**	**18.9**
Qwen2.5 Math-7B	Baseline	72.4	23.3
	ARM	**73.4**	**25.6**
	Best-of-N(N=5)	72.8	23.3
	Best-of-N + ARM	**73.4**	**25.6**

G.6 Inference Time trick comparison

To more comprehensively evaluate the robustness, effectiveness, and compatibility of ARM with established inference-time scaling techniques, we further compare its performance against the widely used Best-of-N sampling approach during inference. Specifically, Table 18 and Table 19 summarize the results obtained by applying ARM alone, Best-of-N sampling alone, and their combined usage on two representative reasoning benchmarks. For all settings, we fix the generation hyperparameters to a temperature of 0.5 and a top_p of 0.95 to ensure a consistent sampling regime. As demonstrated in the table, both ARM and Best-of-N independently yield improvements over the baseline, and their combination produces an even larger performance gain, suggesting that ARM complements rather than competes with existing inference-time strategies. These findings collectively underscore the practical value and scalability of ARM as a lightweight inference-time method for enhancing reasoning capabilities across diverse tasks.

We evaluate additional inference-time techniques on Qwen2.5-Math-1.5B on Math500 and further examine their combination with ARM. ARM fundamentally differs from existing inference-time methods: it neither modifies the input prompt nor intervenes in the sampling process. Instead, ARM operates by adjusting the model's internal representations, enabling it to be seamlessly combined with other inference-time tricks. The results in Section G.6 clearly support this distinction. While ARM alone may slightly underperform certain inference-time methods in some cases, combining ARM with these methods consistently leads to improved performance. This demonstrates the complementary nature of ARM and highlights its particular effectiveness when paired with sampling-based inference techniques.

G.7 Performance of Inserting Meaningless Tokens on Several Benchmarks

In Table 20, we provide the results for the remaining models on additional benchmarks after adding meaningless tokens. On most datasets, Meaningless Tokens and ARM each show strengths on differ-

Table 19: Performance on Math-500 of different inference time tricks with us on Qwen2.5-Math-1.5B

Model	Method	Baseline	ARM	Comparison methods	ARM + Comparison methods
Qwen2.5 Math-1.5B	Plan-and-Solve-Prompting	63.8	67.8	62.0	-
	Self-Consistency	63.8	67.8	73.6	74.6
	LLM as Optimizer	63.8	67.8	66.4	-

ent benchmarks. However, Meaningless Tokens exhibit noticeable instability on the GPQA dataset, and in some cases may even decrease model performance. In contrast, ARM remains consistently stable across all benchmarksw, reinforcing its reliability as an inference-time improvement method. These findings further support the complementary nature of the two approaches while highlighting ARM's robustness.

Table 20: Results of adding meaningless tokens across diverse benchmarks.

Model	GPQA Diamond	Math-500	AIME 2022–2024	AIME 2025	LiveCodeBench	HumanEval
Qwen2.5-Math-1.5B	27.3	65.9	17.5	10.0	N/A	12.2
Qwen2.5-Math-7B	31.3	74.6	23.3	16.7	N/A	23.8
Qwen2.5-7B-Instruct	27.8	73.4	20.0	16.7	34.2	45.7
Qwen2.5-32B-Instruct	34.9	81.7	22.8	20.0	49.7	50.9
Llama-3.1-8B-Instruct	28.5	43.4	13.3	N/A	18.2	42.7
Gemma-3-4B-It	35.9	72.8	20.0	20.0	20.5	18.9
Gemma-3-27B-It	33.8	86.8	26.7	26.7	34.4	11.6

G.8 Compared ARM with Random Shift Method

In this section, we test several alternatives to ARM's stochastic outward shift, including fixed-midpoint, fixed-maximum, deterministic magnitude-aware, and Gaussian random shifts. All perform worse than the original ARM in the evaluated setting, as shown in Table 21. One possible interpretation is that ARM works well when near-zero activations are redistributed in a bounded but non-collapsed way: fixed or Gaussian-style variants may concentrate activations too narrowly, whereas deterministic magnitude-aware shifts may be too rigid. We did not study a learnable shift because ARM is intentionally training-free; adding learnable parameters would turn it into a trained adaptation module and fall outside the scope of our method.

Model	Baseline	Gaussian Random	Fixed Midpoint	Fixed Max	Magnitude Aware	ARM
Qwen-2.5-Math-1.5B	63.8	65.6	64.6	66.0	64.6	**67.0**
Qwen-2.5-7B-Instruct	73.2	74.8	72.0	73.6	73.0	**75.2**

Table 21: Comparison of different methods.

G.9 Comparison with Global Activation Scaling

To distinguish ARM from uniformly scaling all activations, we compare ARM with two global activation-scaling baselines on Qwen2.5-Math-1.5B using MATH-500. The global-scaling baselines multiply every post-activation value by a fixed coefficient, whereas ARM selectively modifies near-zero activations while preserving their signs.

Table 22: Comparison between global activation scaling and ARM on MATH-500 using Qwen2.5-Math-1.5B.

Method	Pass@1
Vanilla	63.8
Global Scaling ($\times 1.1$)	64.2
Global Scaling ($\times 1.5$)	60.2
ARM	**67.0**

As shown in Table 22, a small global scaling factor produces only a marginal improvement, while a larger scaling factor decreases performance. In contrast, ARM achieves a higher accuracy by selectively redistributing near-zero activations rather than uniformly amplifying the entire activation vector. These results suggest that the effectiveness of ARM is associated with the location-selective redistribution of activations, rather than with global magnitude scaling alone.

H Additional Average Attention Weights

In this section, we present additional figures of average attention weights to further validate the previously observed phenomenon. When computing the average attention weights for Llama3.1 and

Gemma3, we exclude the first token to make the effect clearer, as position tokens tend to absorb most of the attention. The results are in Figure 14, Figure 15, Figure 16, Figure 17, Figure 18, Figure 19, Figure 20, Figure 21. From results, we can see that in the most layers, models will have same phenomenon like we mentioned before.

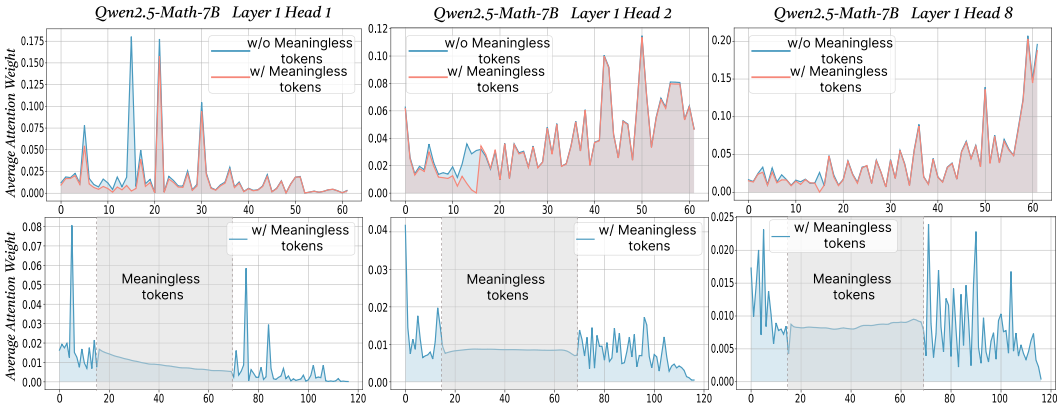


Figure 14: The average attention weights of Qwen2.5-Math-7B in Head 1, 2, 8.

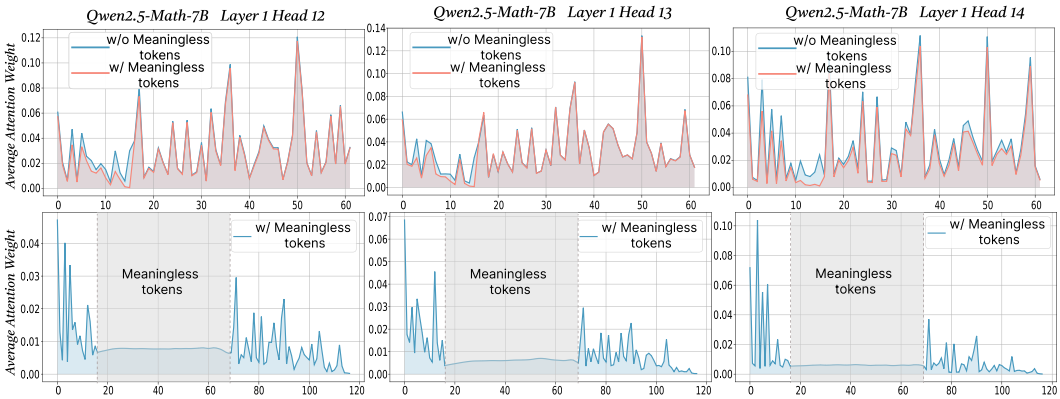


Figure 15: The average attention weights of Qwen2.5-Math-7B in Head 12, 13, 14.

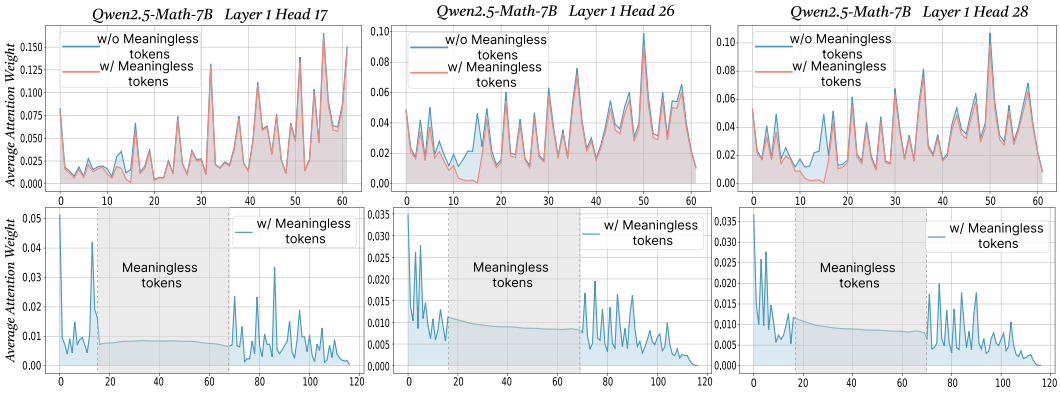


Figure 16: The average attention weights of Qwen2.5-Math-7B in Head 17, 26, 28.

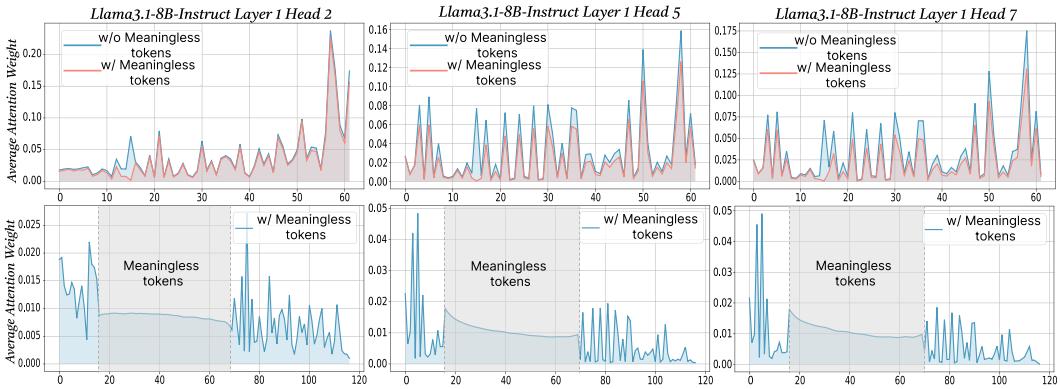


Figure 17: The average attention weights of Llama3.1-8B-Instruct in Head 2, 5, 7.

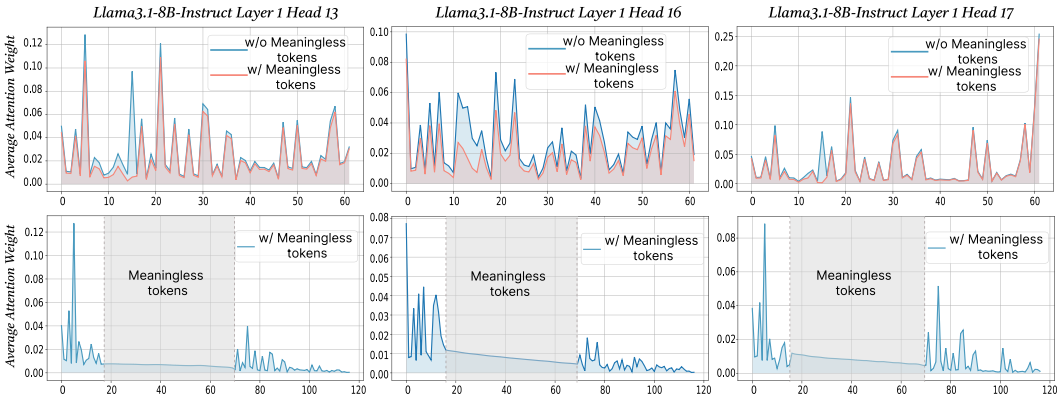


Figure 18: The average attention weights of Llama3.1-8B-Instruct in Head 13, 16, 17.

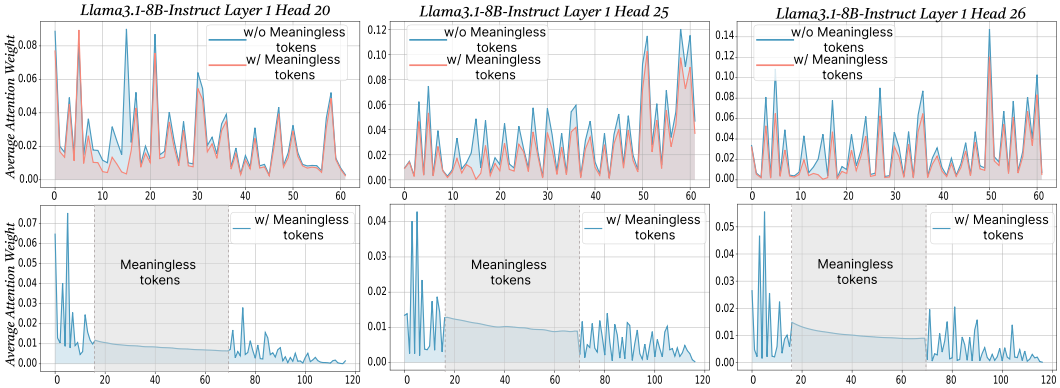


Figure 19: The average attention weights of Llama3.1-8B-Instruct in Head 20, 25, 26.

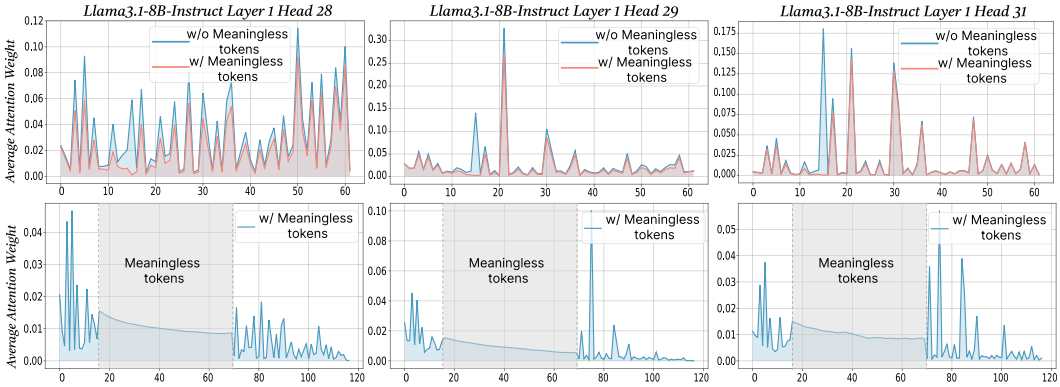


Figure 20: The average attention weights of Llama3.1-8B-Instruct in Head 28, 29, 31.

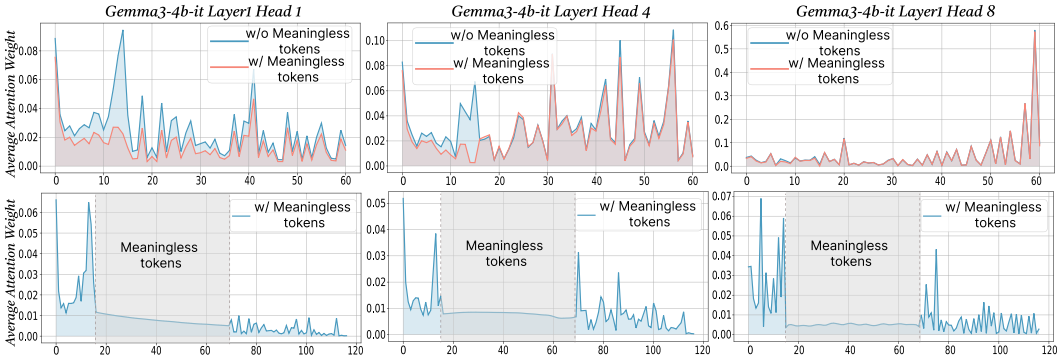


Figure 21: The average attention weights of Gemma3-4b-it in Head 1, 4, 8.