MoDE: Weight Denoising Towards Better LLM Performance through a Mixture of Domain Experts

Anonymous ACL submission

Abstract

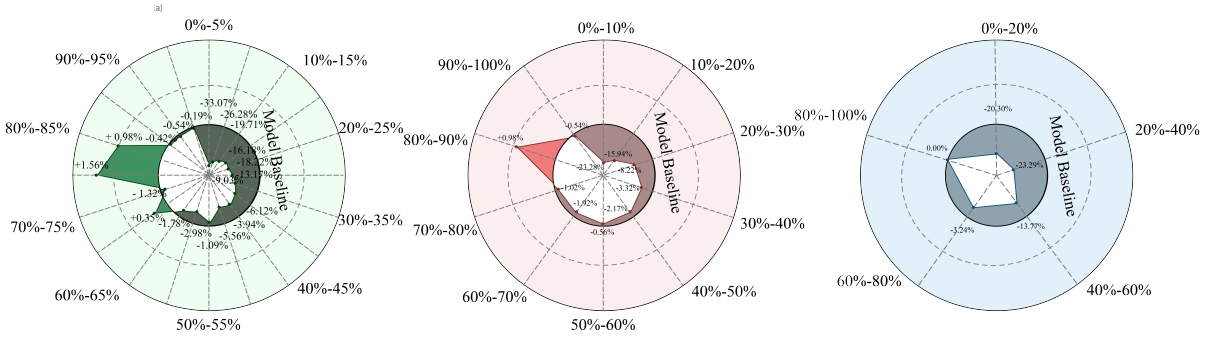
In LLM weight pruning, the "criteria" method, alongside sparse training, relies on ranking weight importance to guide pruning decisions. However, this approach frequently leads to performance degradation, as it assumes that importance equates to contribution, implying that any removal inevitably incurs loss. Our findings reveals that, under specific domain, some weights may act as noise, and pruning them can actually improve performance. This offers a new perspective on pruning: ***Shifting the goal from loss minimization to performance gains.*** To this end, 1) we propose **the Noise Weight Hypothesis**, which posits the existence of harmful weights in LLMs whose activation can degrade performance in domain-specific tasks. 2) We introduce the **DENoise** (**D**omain **E**xpert weight de**Nois**ing) algorithm, which removes domain-aware noise weight without fine-tuning. 3) We further develop the **MoDE** (**M**ixture of **D**omain **E**xperts), which employs a bilevel trainable router to dynamically activate the domain-specific expert, leading to improved task accuracy. Results show that applying DENoise algorithm achieves 2–3% performance gains on each benchmarks without any additional parameters or tuning, while MoDE yields an average improvement of over 1.1% against baseline models.

1 Introduction

The rapid growth of large language models has made deployment increasingly costly due to excessive parameter counts (Patterson et al., 2021; Strubell et al., 2019). Pruning, which removes redundant weights while minimizing accuracy loss, is widely regarded as an effective solution(Cheng et al., 2024). A common strategy involves ranking parameters by importance—typically measured by magnitude (Han et al., 2015b; See et al., 2016), norm(Li et al., 2016; He et al., 2018), or estimated loss change(You et al., 2019; Liu et al., 2021)—and removing the lowest-ranked ones to preserve key contributing parameters to performance. However, this method relies on an implicit assumption: all parameters, even those with small or negative values, are inherently beneficial. As a result, pruning these weights is often assumed to degrade performance.

Nonetheless, this commonly held assumption warrants reconsideration. Let the biological inspiration of neural networks (Hinton and Sejnowski, 1986) be the conceptual analogy, studies reveal that certain weights, while functionally meaningful in general, can exhibit disruptive responses under specific tasks, a phenomenon known as neuronal noise (Faisal et al., 2008). Then it is reasonable to hypothesize that *in artificial neural network, some weights, though beneficial for generalization, may trigger misleading activations in specific downstream tasks.* Removing "noise" may therefore enhance task performance. Interestingly, our preliminary (see Section 2) support this hypothesis. When evaluating model performance on specific task datasets under pruning conditions, we found that removing weights from certain mid-to-low importance intervals led to improved accuracy, and was consistently observed across up to 57 tasks.

Motivated by this, we propose the **Noise Weight Hypothesis**, suggesting that harmful weights exist in LLMs and their removal can enhances task-driven performance without further training. To make the hypothesis practical, we integrate it into a Mixture-of-Experts (MoE) framework—whose core principle is that, for each specific domain of task, the model dynamically activates the most suitable expert (Fedus et al., 2022). Following this rationale, we propose the **DENoise** (Domain Expert weight deNoising) algorithm, which identifies and removes noise-like weights (defined as Noise Weights) from self-attention and FFN layers to construct domain expert subnetworks. Building on this, we develop the **MoDE** (Mixture of Domain Expert) framework, which routes each input to its



a). Comparison when pruned by 5% b). Comparison when pruned by 10% c). Comparison when pruned by 20%

Figure 1: Parameter Importance Sensitivity Analysis on the MMLU Philosophy Task. This figure evaluates model performance variation under different pruning ratios (5%, 10%, and 20%) by selectively removing weights across ranked importance intervals based on the metric $|\mathcal{W}| \times \|X\|_2$. Results reveal a non-monotonic relationship between parameter importance and performance, with marginal gains observed when pruning mid-to-low ranked weights.

most relevant denoised expert at the sequence level for efficient activation. Thereby, the main contributions of this work are as follows:

Improvement Perspective for Reinterpreting Parameter Redundancy. The Noise Weight Hypothesis offers a perspective by focusing on negatively contributing weights, suggesting that LLMs may contain noise weights that interfere with task performance. This viewpoint complements existing importance-based pruning strategies and opens up a new theoretical dimension for understanding and approaching model compression.

Noise Weight Elimination Mechanism for Performance Improvement. The tuning-free DENoise algorithm identifies and removes noise weights to derive task-specialized expert subnetworks from pretrained LLMs. Experimental results show an average performance improvement of 1%–3% on benchmarks such as MMLU, MBPP, and GSM8K, with a maximum gain of 6.8%. This method demonstrates that model compression and performance improvement can be achieved simultaneously, providing a practical tool for optimizing LLMs in resource-constrained environments.

Lightweight and Deployable Domain Expert Construction. The MoDE integrates the DENoise algorithm with a bilevel dynamic router to construct a domain expert system. MoDE offers an optimization option that requires no extensive model modifications and can be directly integrated into existing architectures, achieving an average performance improvement of 1.1% across multiple mainstream LLMs. This provides a practical solution for efficient deployment and enhanced domain adaptability of LLMs.

2 Preliminary

Recent studies have revealed that large language models (LLMs), despite being trained as monolithic architectures, inherently exhibit internal structures resembling a Mixture-of-Experts (MoE) system (Zhang et al., 2023; Dai et al., 2024). Specifically, weights with different activation patterns within layers demonstrate emergent functional preferences for certain representation region (Zhang et al., 2022).

Formally, let a pretrained LLM be parameterized by $\mathcal{W} \in \mathbb{R}^{u \times v}$, which operates on input feature matrix $X \in \mathbb{R}^{v \times d}$ to produce hidden activations $H = \mathcal{W}X$. Generally, all parameters $\mathcal{W}$ participate in inference. However, due to overparameterization, it is possible to decompose $\mathcal{W}$ into subsets $\{W_1, W_2, \ldots, W_E\}$, where each subset corresponds to an implicit expert $\mathcal{E}_e$. The model output can thus be reformulated as:

$$H = \sum_{e=1}^{E} \mathcal{R}[x \in \Gamma_e] \cdot (\mathcal{E}_e X), \tag{1}$$

where $\mathcal{R}[\cdot] \in \{0, 1\}$ is a routing function specifying whether the input x belongs to the representation region Γ_e relevant to expert $\mathcal{E}_e$.

Under this framework, the critical question is to construct the expert. Typically, we apply importance-based parameter selection methods, where each parameter is assigned an importance score (Han et al., 2015b). Under the common assumption that parameters with lower importance contribute less positively to model performance, such parameters are prioritized for pruning. In contrast, sudies such as the Lottery Ticket Hy-

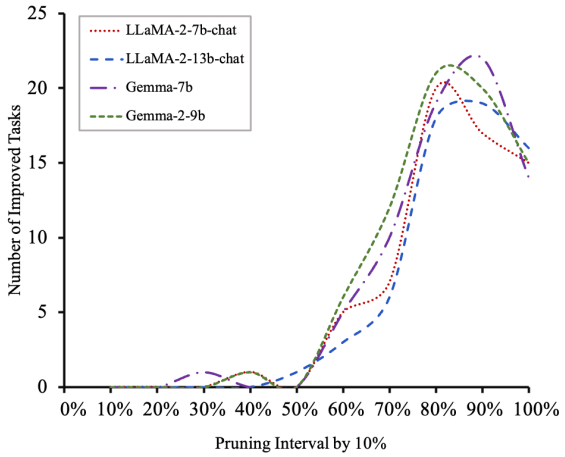


Figure 2: The number of tasks achieving accuracy improvements within different pruning intervals by 10% across models. Note that each task may experience performance gains in multiple pruning intervals.

pothesis (Frankle and Carbin, 2018), have challenged the importance assumption by demonstrating that sparse sub-networks with randomly initialized weights can achieve competitive performance, suggesting that high importance does not necessarily equate to high contribution.

Thus, we pose the question: *Does importance score serve as an initial filtering signal rather than a final decision criterion?*

Noise Weight Hypothesis. To explore the relationship between parameter importance and task performance, we conduct extensive experiments across 57 tasks with pruning ratios of 1%, 5%, 10%, 20% and 50% on 4 models `llama-2-7b/13b-chat-hf` and `Gemma-7b/9b` model. Parameter importance is measured using the widely adopted pruning metric $|\mathcal{W}| \times \|X\|_2$, (He et al., 2018; Ma et al., 2023; Sun et al., 2024) and weights are pruned from different ranked importance intervals. Figure 1 demonstrates this phenomenon on the MMLU Philosophy task using the `llama-2-7b-chat-hf`, with similar trends observed across diverse domains. To prove the robustness, Figure 2 shows that many tasks benefit from pruning weights within mid-to-low importance intervals. This may suggests that the widely used importance metrics may not serve as definitive criteria for parameter evaluation. Based on these observations, we propose the **Noise Weight Hypothesis**, which posits the widespread presence of previously overlooked parameters—*noise weights*—in LLMs. These weights may appear active by importance metrics but in fact negatively impact performance.

3 Method

3.1 DENoise

Based on the Noise Weight Hypothesis, we first design a Domain Expert Weight Denoising algorithm (DENoise) , using importance score as intermediate variable to further detect and remove noise weights in hiddn layers, including attention and FFN layers.

Clustering-based Classification. Consider a set of distinct knowledge domains $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$, where each domain D_i encapsulates specialized knowledge relevant to a specific task. These domains are aggregated into a comprehensive main knowledge set $\mathcal{D}$.

For a given task $\mathcal{T}$, the algorithm selects the most suitable main knowledge domain D_j by computing the posterior probability $P(D \mid \mathcal{T})$, which quantifies the likelihood of task $\mathcal{T}$ belonging to each domain. This process is expressed as:

$$D_j = \arg \max_{j \in \{1,\ldots,n\}} P(D_j \mid \mathcal{T}), \tag{2}$$

where D_j is the domain with the highest posterior probability, ensuring that the task $\mathcal{T}$ is matched with the most relevant domain.

Once the main domain D_j is selected, it is further subdivided into smaller subdomains using K-means clustering. For each dataset within D_j, feature representations are extracted via the embedding layer of a Transformer model, forming a set of feature vectors $\mathcal{F}_j = \{F_{j_1}, F_{j_2}, \ldots, F_{j_l}\}$, where each F_{j_i} corresponds to a data sample. K-means clustering is then applied to partition $\mathcal{F}_j$ into k subdomains by minimizing the intra-cluster variance, formalized as:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \sum_{i=1}^{k} \sum_{f \in C_i} \|f - \mu_i\|_2, \tag{3}$$

where $C_i \subseteq \mathcal{F}_j$ denotes the set of points assigned to the i-th cluster, μ_i represents the centroid of cluster C_i, and $f \in \mathcal{F}_j$ indicates that all data points considered belong to the feature set extracted from domain D_j.

For given any new task $\tilde{\mathcal{T}}$, its feature vector $F_{\tilde{\mathcal{T}}} = \text{Embedding}(\tilde{\mathcal{T}})$ is extracted and compared to the centroids μ_k of the subdomains. The task is assigned to the subdomain D_{j_k}(equivalently, cluster C_k) that minimizes the Euclidean distance:

$$\tilde{k} = \arg\min_{k} \|F_{\tilde{\mathcal{T}}} - \mu_k\|_2. \tag{4}$$

Once assigned to the subdomain $D_{j_{\tilde{k}}}$, the task is further processed according to the knowledge characteristics. The complete procedure for optimizing the denoising threshold for domain experts is outlined in Algorithm 1.

Metric Calculation on Patches. After identifying the subdomain $D_{j_{\tilde{k}}}$, this section quantifies the critical information contained within it. Activation values serve as an effective metric for assessing the importance of weights and connections within the network because they directly measure how strongly weights respond to input features, reflecting their contribution to the network's output (Han et al., 2015b). To perform this assessment, we begin by scaling the weight matrix $\mathcal{W}$ and feature matrix X(activation embedding vector). A scaling factor α reduces the dimensionality of the matrices, yielding a smaller matrix of size $\beta u \times \beta v$ (where $\beta = 1/\alpha$). Each element in this reduced matrix corresponds to a patch $P_{\eta\nu}$, which represents a subregion of the original matrix and contains condensed information.

Afterwards, to further evaluate the significance of each patch, we calculate the following importance metric $\mathcal{M}_{\eta\nu}$ based on the l_p-norm:

$$\mathcal{M}_{\eta\nu} = \sum_{(s,t)\in P_{\eta\nu}} \left(|\mathcal{W}_{st}| \times \|X_{st}\|_p\right). \quad (5)$$

This metric aggregates the weighted contributions of each element within the patch, computed using the l_p-norm. With the norm vector of the input feature activations, the importance of each weight can be determined by performing a patch-wise dot product.

Domain Experts Formation. To identify and remove noise weights for a given task $\mathcal{T}$, we first define a candidate set of noise weights by evaluating importance scores row-wise. For each row i, a specific threshold interval $[\theta^{(i)}_{\min}, \theta^{(i)}_{\max}]$ is determined, and all parameters whose importance scores fall within this interval are considered potential noise candidates. The candidate set θ_r is defined as:

$$\theta_r = \bigcup_{i=1}^{\beta u} \left\{ \mathcal{M}_{i:}(\sigma) \mid \mathcal{M}_{i:}(\sigma) \in [\theta^{(i)}_{\min}, \theta^{(i)}_{\max}] \right\}, \quad \text{where } \theta^{(i)}_r \subseteq \mathbb{R}. \quad (6)$$

Based on the candidate set θ_r, we systematically evaluate the impact of removing different candidate intervals on model performance. By measuring the accuracy $\texttt{Acc}(\mathcal{W} \setminus \theta_r)$ after removing each candidate interval, we determine the interval that yields the best post-denoising performance. The optimal denoising threshold θ^* is selected as:

$$\mathcal{E}_{\mathcal{T}} = \mathcal{W} \setminus \theta^*, \quad \text{where } \theta^* = \arg\max_{\theta_r \subseteq \mathcal{W}} \texttt{Acc}\left(\mathcal{W} \setminus \theta_r\right). \quad (7)$$

Finally, the denoised hidden layers are aggregated to construct the Domain Expert, ensuring that critical knowledge is preserved while redundant and noisy parameters are effectively removed.

Algorithm 1 Threshold Optimization for Domain Experts Weight Denoising.

Input: Domain-specific dataset $\mathcal{D}_T$; initial threshold θ_{init}; maximum threshold $\theta_{\max}$; step size δ_θ; pre-trained weights $\mathcal{W}$.
Output: Optimal denoising threshold θ^*.

Initialize $\theta \leftarrow \theta_{\text{init}}$, $\theta^* \leftarrow \theta_{\text{init}}$, and best_acc $\leftarrow 0$.
Extract features: $F_T \leftarrow \texttt{Embedding}(\mathcal{D}_T)$.
Partition k subdomains: $\mathcal{C} \leftarrow \texttt{KMeansClustering}(F_T, k)$.
while $\theta \leq \theta_{\max}$ **do**
  Compute candidate noise neurons: $\theta_r \leftarrow \texttt{ComputeNoiseNeurons}(\mathcal{C}, \theta)$.
  Denoise parameters: $\mathcal{W}_{\text{denoised}} \leftarrow \mathcal{W} \setminus \theta_r$.
  Evaluate accuracy: acc $\leftarrow \texttt{Acc}(\mathcal{W}_{\text{denoised}}, \mathcal{D}_k)$.
  if acc $>$ best_acc **then**
    Update best_acc $\leftarrow$ acc, $\theta^* \leftarrow \theta$.
  end if
  Increment threshold: $\theta \leftarrow \theta + \delta_\theta$.
end while
return θ^*.

3.2 MoDE Architecture

To fully leverage the specialized weight configurations constructed by DENoise, we introduce MoDE(Mixture of Domain Expert), which employs a bilevel trainable router to dynamically classify any tasks into the most relevant domain and activate the expert.

A Bilevel Trainable Router. We introduce a bilevel trainable router $\mathcal{R}$, where each level is implemented as a single-layer MLP, to classify tasks through two hierarchical stages. First, the task embedding $F_{\mathcal{T}}$ is classified into a main knowledge

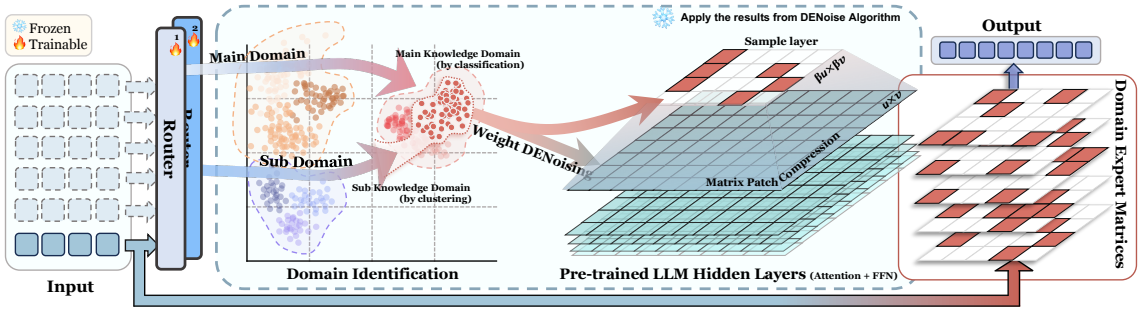


Figure 3: The inference flow through our MoDE. The *embedded input tokens* (▢) is first processed by a bilevel trainable (🔥) router, where it is initially classified into the main knowledge domain by level 1 and further into a sub knowledge domain by level 2. The input is then passed to the identified Domain Expert, which is formed by applying the result from DENoise Algorithm, from the hidden layers of a frozen (❄) pre-trained LLM. Finally *output tokens* (▢) are processed by Domain Expert to output.

domain D_b. Then, in the second stage, the task is further classified into a subdomain $D_{b_{\bar{k}}}$ within the selected main domain, as shown:

$$D_{b_{\bar{k}}} = \mathcal{R}_{\text{sub},b}(\mathcal{R}_{\text{main}}(F_{\bar{\mathcal{T}}})). \tag{8}$$

Here, $\mathcal{R}_{\text{main}}(F_{\bar{\mathcal{T}}})$ maps the task embedding to a specific main domain D_b, and $\mathcal{R}_{\text{sub},b}(F_{\bar{\mathcal{T}}})$, conditioned on this domain, further assigns the task to a subdomain $D_{b_{\bar{k}}}$.

The router is trained by minimizing cross-entropy loss at both levels, optimizing the classification process:

$$\begin{aligned}\mathcal{L}_{\mathcal{R}} &= \texttt{CrossEntropyLoss}(\hat{y}, y) \\ &= -\sum_{q=1}^{n} y_q \log\left(P(D_q \mid X)\right), \end{aligned} \tag{9}$$

where $y = [y_1, \ldots, y_n]$ is the one-hot encoded true label vector with $y_q \in \{0, 1\}$, $\hat{y} = [P(D_1 \mid X), \ldots, P(D_n \mid X)]$ is the predicted probability distribution over the n domain classes, and $P(D_q \mid X) = \hat{y}_q$ denotes the predicted probability of input X being classified into domain D_q.

MoDE Inference. The inference process in MoDE begins with the input tokens X_{in}, which are first passed through an embedding layer into the embedded representation $\hat{X}$. The embedded tokens $\hat{X}$ are then fed into the router, a trainable classifier designed to assign the input to the most relevant domain. The router $\mathcal{R}$ classifies the input into a main knowledge domain and then sub knowledge domain, depending on the characteristics of the task, expressed as:

$$D_{\kappa_\lambda} = \mathcal{R}(\hat{X}). \tag{10}$$

After classification, the model proceeds `DENoise`. This step filters out irrelevant weights from the pretrained LLM, retaining only those necessary for the selected domain. The denoised weights, denoted as W_{denoised}, form the core of the Domain Expert:

$$W_{\text{denoised}} = \texttt{DENoise}(\mathcal{W}, D_{\kappa_\lambda}). \tag{11}$$

These denoised expert in dynamically activated to process the embedded input $\hat{X}$, ensuring that only the domain-relevant parameters are used for the current task. The embedded input $\hat{X}$ is then forwarded through the selected Domain Expert, which generate the intermediate output $\hat{Y}$. Finally, the intermediate output $\hat{Y}$ is subjected to a linear transformation and a softmax operation to produce the final output Y, which represents the model's prediction for the given task. Whole inference procedure is shown in Figure 3.

4 Experiment

4.1 Setup

Datasets. We conduct experiments across three domains: general knowledge, code, and mathematics. For general knowledge, we use MMLU (Hendrycks et al., 2021). For code, we adopt MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021), and for mathematics, we use GSM8K (Cobbe et al., 2021) and MathQA (Amini et al., 2019). Validation sets from MMLU, MBPP, MathQA, and GSM8K are used as reference data, with final evaluations on their respective test sets. For HumanEval, we follow standard practice by using the MBPP validation set as reference and evaluating on the full dataset. We adopt a 5-shot for MMLU and 0-shot for all other datasets.

Table 1: Performance comparison of DENOISE and MODE with baseline on foundation models.

	Method	MMLU	MBPP		HumanEval		GSM8K	MathQA
		acc	pass@1	pass@10	pass@1	pass@10	acc	acc
llama-2-7b-chat-hf	BASELINE	45.81	19.24	23.60	14.45	19.51	20.24	25.33
	DENOISE	47.83	21.32	26.40	15.73	20.73	22.21	27.34
	Improvement	+2.02	+2.08	+2.80	+1.28	+1.22	+1.97	+2.01
	MODE	47.34	20.58	25.80	14.51	19.51	20.79	26.13
	Improvement	+1.53	+1.34	+2.20	+0.06	±0	+0.55	+0.80
llama-2-13b-chat-hf	BASELINE	52.34	9.68	13.00	18.66	28.05	31.77	24.86
	DENOISE	53.18	11.52	16.00	19.45	29.88	34.42	27.57
	Improvement	+0.84	+1.84	+3.00	+0.79	+1.83	+2.65	+2.71
	MODE	52.93	12.39	14.80	19.13	29.27	33.86	26.34
	Improvement	+0.59	+2.71	+1.80	+0.47	+1.22	+2.09	+1.48
lLama-2-70b-chat-hf	BASELINE	66.87	45.22	66.15	30.52	59.34	59.47	35.12
	DENOISE	68.89	47.15	66.98	31.71	60.72	61.83	35.96
	Improvement	+2.02	+1.93	+0.83	+1.19	+1.38	+2.36	+0.84
	MODE	68.12	45.93	66.34	31.22	60.01	60.56	35.33
	Improvement	+1.25	+0.71	+0.19	+0.70	+0.67	+1.09	+0.21
Gemma-7b	BASELINE	63.56	2.94	9.00	15.31	20.12	57.92	37.12
	DENOISE	65.30	6.20	15.80	16.77	22.56	59.59	39.57
	Improvement	+1.74	+3.26	+6.80	+1.46	+2.44	+1.67	+2.45
	MODE	65.05	5.85	13.90	12.93	18.29	59.29	38.79
	Improvement	+1.49	+2.91	+4.90	−0.38	−0.11	+1.37	+1.67
Gemma-2-9b	BASELINE	69.71	8.36	9.80	12.87	18.90	68.46	50.75
	DENOISE	71.07	8.52	10.80	15.12	22.56	69.98	51.22
	Improvement	+1.36	+0.16	+1.00	+2.25	+3.66	+1.52	+0.47
	MODE	70.90	8.28	10.40	14.33	20.73	69.45	50.97
	Improvement	+1.19	−0.08	+0.60	+1.46	+1.83	+0.99	+0.22

Baseline & Foundation Models. We evaluate our method on five foundation models: `llama-2-7b/13b/70b-chat` and `Gemma-7b/2-9b`, covering a wide range of model sizes. Both LLaMA-2 and Gemma series are decoder-only transformer-based models optimized for dialogue and natural language understanding. This setup verifies the effectiveness of our approach across dense pretrained models of different scales.

Evaluation Metrics. We report **accuracy (acc%)** for general and mathematics tasks. For coding tasks, we additionally use **pass@1** and **pass@10**, which measure the probability that at least one of the top-k generated solutions is correct, calculated as:

$$\text{pass@k} = \frac{1}{N}\sum_{i=1}^{N} c_i, \tag{12}$$

where $c_i \in \{0, 1\}$ indicates whether the i-th task has at least one correct solution in Top-k.

4.2 Main Results

Our DENOISE algorithm consistently boosts performance across datasets: MMLU, MBPP, HumanEval, GSM8K and MathQA. When incorporated into the MODE architecture, it further improves efficiency, adaptability and stability, as shown in Table 1.

DENOISE. The application of DENOISE using l_2-norm and 10% de noise ratio results in consistent performance improvements with an average gain of 2.04% over the baseline models, as shown in Table 1. Specifically, `llama-2-7b-chat-hf`'s accuracy on MMLU increases by 2.02%, while its performance on MBPP (pass@1) rose by 2.08%, and its accuracy on GSM8K improves by 1.97%. Similar patterns are observed for the other models. Notably, `Gemma-7b` exhibits an increase of 3.26% in MBPP (pass@10) and a 2.4% improvement in MathQA accuracy, demonstrating DENOISE's efficacy across different models and tasks.

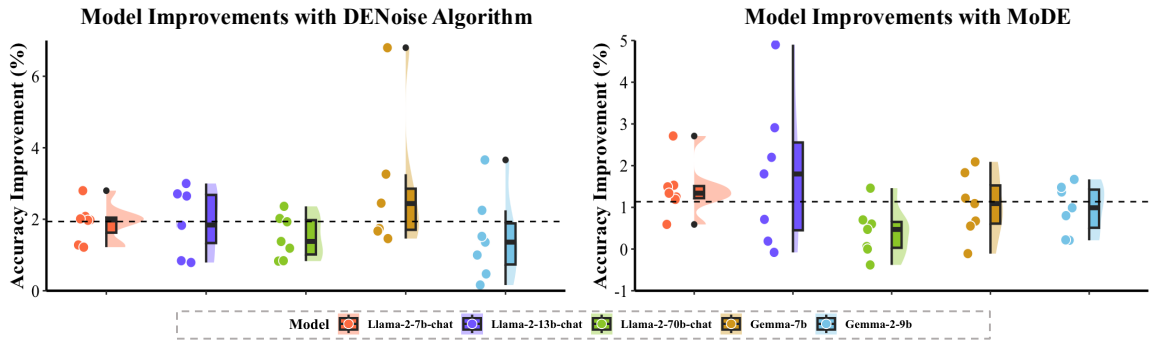


Figure 4: Comparative boxplot analysis of models using DENoise and MoDE methods.

MoDE. To make further comparison, we evaluate the MoDE architecture on the same set of LLMs to examine its effectiveness. Firstly, applying MoDE also led to noticeable performance gains of 1.11% in average, though the improvements were generally more moderate compared to DENoise, as shown in Table 1. In detail, `llama-2-7b-chat-hf`'s accuracy on MMLU increases by 1.53%, while its performance on MBPP (pass@1) improves by 1.34%, and its accuracy on GSM8K see a 0.55% rise. Similarly, `Gemma-7b`'s performance in MBPP (pass@10) increases by 2.91%, with MathQA showing a 1.67% gain. Secondly, the accuracy improvements under MoDE primarily fall within the range of 0.5% to 2.5%, with the highest to 4.9%. The results suggest that MoDE, while less impactful than DENoise in terms of absolute gains, offers a viable and stable method for enhancing model performance with minimal additional computation. Moreover, as shown in Figure 4, both DENoise and MoDE consistently improve model accuracy, yet their distribution patterns differ. As the expert, DENoise exhibits more concentrated gains across models, suggesting its stable performance. While MoDE shows greater variance and occasional performance dips, which may be partially attributed to the additional routing mechanism.

4.3 Ablation Studies

In the ablation studies, we use the `llama-2-7b-chat-hf` model due to its stable performance in previous results. Employing DENoise, we analyze l_p-norm, denoise ratios, layers, and patch-square configurations. Furthermore, employing MoDE, we analyze the k-means clustering process.

l_p-norm. We firstly evaluate the performance of different p-norm values on the MMLU dataset. For $p = 2$, accuracy improves to 47.83%, representing a 2.02% increase over the base accuracy. For $p = 4$, accuracy decreases to 47.27%, noting that different p-norm configurations yield varying results in terms of performance, with $p = 2$ achieving the highest accuracy, as shown in Table 2.

Table 2: DENoise Performance of Different p in l_p-norm on datasets.

	p	MMLU	GSM8K	MathQA
acc(%)	1	46.26	21.34	26.13
	2	47.83	22.21	27.34
	4	47.27	22.05	26.57

Denoising Ratio. We denoise the all model layers (layers 0 to 31) using a 1×1 patch-square configuration and examine different ratios on the MMLU dataset grouped into 12 clusters. With a 10% denoising ratio, accuracy reaches 47.83%, a 2.02% increase over the 45.81% of base. This shows the 10% ratio effectively balances accuracy and computational cost, as shown in Table 3.

Table 3: DENoise Performance of Different Denoising Ratio on MMLU Dataset.

	Ratio	MMLU	Ratio	MMLU
acc(%)	base	45.81	5%	47.79
	1%	47.75	10%	47.83
	3%	47.83	20%	45.33

Denoising Sublayers. We then apply a 5% denoising to MLP layers and Self-Attention layers, from layers 0 to 31, evaluating the impact on the GSM8K and MathQA, grouped into 8 clusters. Results shows denoising the Self-Attention layers yields the best on GSM8K, with a 3.18% improvement. On MathQA, the combination performs best, reaching a 2.01% increase, as shown in Table 4.

Table 4: DENOISE Performance of Denoising Different Sublayers on GSM8K and MathQA Datasets.

	Patch	GSM8K	MathQA
acc(%)	base	20.24	25.33
	MLP	21.61	26.87
	Self-Attention	23.42	25.90
	Combination	22.21	27.34

Table 5: DENOISE Performance of Different Patch Size on MBPP and HumanEval Datasets.

	Patch	MBPP		HumanEval	
		@1(%)	@10(%)	@1(%)	@10(%)
pass@k	base	19.24	23.60	14.45	19.51
	1×1	20.80	26.00	15.73	20.73
	2×2	19.88	25.60	15.67	20.12
	4×4	18.58	24.00	14.88	20.73
	16×16	18.40	24.40	13.97	17.68

Patch-square Configuration. Besides, based on 10% denoising ratio, we evaluate different patch-square configurations on the MBPP and HumanEval, grouped into 3 clusters. For MBPP, the 1×1 **patch-square** achieves the best performance on pass@1 with a 1.56% improvement, and on pass@10 with a 2.40% increase. Similarly, on HumanEval, the 1×1 configuration lead with 1.28% gains for pass@1 and 1.22% for pass@10 respectively over the baseline, as shown in Table 5.

K-means Clustering. Finally, we examine the impact of different K values applied in MODE in the K-means clustering step on the MMLU dataset, grouped into 12 clusters. We vary the number of clusters from 6 to 16, the results show that with **K=12**, the accuracy reaches the highest value of 47.79%, outperforming other cluster settings, as shown in Table 6.

Table 6: MODE Performance of Different K-means on MMLU Dataset.

	K	MMLU	K	MMLU
acc(%)	6	47.27	12	47.79
	8	47.50	14	47.27
	10	47.46	16	47.65

5 Related Works

Pruning. Based on granularity, neural network pruning methods are categorized as unstructured, structured, or semi-structured. Unstructured pruning removes individual low-importance weights to achieve high sparsity; structured pruning eliminates entire structures such as neurons or channels (Blalock et al., 2020; Frantar and Alistarh, 2023; Wang et al., 2019); semi-structured pruning offers a trade-off between flexibility and regularity (Syed et al., 2023). One of the core component of pruning is the design of importance metrics, which assess parameter relevance using heuristics such as magnitude, gradient-based estimates, or loss sensitivity (Han et al., 2015a; Molchanov et al., 2019; Sun et al., 2024). These metrics guide the removal of redundant parameters but have shown limited consistency and robustness across architectures and tasks (Sutskever et al., 2013). Our work reframes *pruning as a mechanism not only for structural simplification but also for enhancing model effectiveness*, providing a novel extension to its theoretical and practical scope.

Mixture of Experts. Most existing Mixture-of-Experts (MoE) architectures construct experts by replicating feed-forward networks (FFNs) within Transformer blocks, assuming specialization emerges implicitly through routing during training (Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022). Recent works like DS-MoE (Pan et al., 2023) and the Emergent Modularity hypothesis (Zhang et al., 2023) provide empirical evidence of latent sparsity and modularity in FFN layers, while MoEfication (Zhang et al., 2022) and LLaMA-MoE (Zhu et al., 2024) further explore expert configuration through parameter reallocation and reuse without retraining. Our work *explicitly constructs domain-specialized experts across both FFN and self-attention* layers, enabling lightweight, task-adaptive modularization.

6 Conclusion

Conclusion. In this work, we empirically demonstrate that removing certain weights can lead to performance improvements of LLMs on specific tasks. Building on this observation, we propose the DENoise algorithm to prune noise weights from both attention and FFN layers to construct domain experts. We further introduce a lightweight MoDE framework that can be integrated into decoder-only LLM architecture.

Limitation and Future Work. This work primarily has 3 limitations. Regarding the proposed *Noise Neuron Hypothesis*, First, our validation relies on the use of the l_p-norm to estimate weight importance; incorporating additional metrics such as loss sensitivity, gradient-based scores, or influence functions could further solidify our conclusions. Second, although this work introduces practical and deployable methods derived from the hypothesis, the justification of the hypothesis itself remains largely empirical. We do not yet propose a generalized metric grounded in theory to precisely quantify noise weights. Future work will explore constructing refined metrics based on or beyond importance estimation to localize such harmful parameters more effectively. Third, from an application perspective, our evaluation is limited to a set of representative NLP tasks; extending this framework to more diverse tasks such as open-ended generation, multi-modal reasoning, or real-world dialogue systems remains an open challenge for assessing the generalizability of the hypothesis.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *Preprint*, arXiv:1905.13319.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of Machine Learning and Systems*, 2:129–146.

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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, and et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Nan Du, Lei Hou, Zongwei Mao, and et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning (ICML)*.

A. Aldo Faisal, Luc P. J. Selen, and Daniel M. Wolpert. 2008. Noise in the nervous system. *Nature Reviews Neuroscience*, 9(4):292–303.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Song Han, Huizi Mao, and William J Dally. 2015a. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015b. Learning both weights and connections for efficient neural networks. *Preprint*, arXiv:1506.02626.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2234–2240. AAAI Press.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

G. E. Hinton and T. J. Sejnowski. 1986. *Learning and relearning in Boltzmann machines*, page 282–317. MIT Press, Cambridge, MA, USA.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020.

Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710.

Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jingliang Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: on the structural pruning of large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Conference on Computer Vision and Pattern Recognition*, pages 11264–11272.

Hao Pan, Renjie Zhang, Zhiyuan Liu, and Maosong Sun. 2023. Ds-moe: Expert sparsity in pretrained transformers. In *Findings of the Association for Computational Linguistics (ACL)*.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lucia Munguia, David Rothchild, and Jeffrey Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650. Association for Computational Linguistics.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

Aaquib Syed, Phillip Huang Guo, and Vijaykaarti Sundarapandiyan. 2023. Prune and tune: Improving efficient pruning techniques for massive language models.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. 2019. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6566–6575. PMLR.

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. 2019. *Gate decorator: global filter pruning method for accelerating deep convolutional neural networks*. Curran Associates Inc., Red Hook, NY, USA.

Renjie Zhang, Qian Liu, Hao Pan, and Zhiyuan Liu. 2023. Emergent modularity and task-alignment in pretrained transformers. *arXiv preprint arXiv:2305.15450*.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland. Association for Computational Linguistics.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*.

A Scaling Hidden Layers into Patches Maintains Information in Self-Attention and MLP

In Transformer models, the Self-Attention and MLP layers are critical for capturing global contextual information and performing non-linear transformations on feature representations. The scaling of hidden layers into patches might raise concerns about the potential loss of information, but this process is designed to preserve both local and global relationships in the model.

Global Context Preservation in Self-Attention. The Self-Attention mechanism ensures that every token in the input sequence can attend to all other tokens, capturing global dependencies. This operation is described by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{13}$$

Where: Q, K, and V are the Query, Key, and Value matrices. d_k is the dimensionality of the Key vectors. By scaling hidden layers into patches, each patch retains local interactions within the patch. Meanwhile, the Self-Attention mechanism ensures that global interactions between patches are maintained. This is because the attention mechanism operates across all patches, allowing the model to propagate global information and maintain context across the entire sequence of patches. As a result, the scaled matrix retains the full global context, ensuring no information is lost during patch scaling.

Patch Scaling and Information Compression. When the hidden layers are scaled by a factor α, the resulting reduced matrix of size $\beta X \times \beta Y$ ($\beta = 1/\alpha$) has elements that correspond to patches in the original matrix. Each patch P_{ij} captures a compressed representation of the information within the original matrix. By aggregating the contributions from each element in a patch, the scaled matrix effectively compresses the local information, while Self-Attention ensures that this compressed representation continues to interact globally. The importance of each patch is calculated as:

$$\theta_{ij} = \sum_{(m,n)\in P_{ij}} (|W_{mn}| \times \|X_{mn}\|_2) \tag{14}$$

This compression allows for efficient representation of both local and global information, preserving the integrity of the original model.

MLP Layer and Information Flow. Following Self-Attention, the MLP layer processes the globally-contextualized output. The MLP is defined as:

$$\text{MLP}(h) = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot h)) \tag{15}$$

Where: h is the output from Self-Attention. W_1 and W_2 are the weight matrices in the MLP. σ is the activation function (typically ReLU). The MLP performs non-linear transformations on the compressed feature representations from the patches. Since the MLP does not rely on spatial relationships, it processes the patch-level information without any risk of information loss. The critical feature transformations in the MLP are unaffected by the scaling process, ensuring that the information flow remains intact.

B Procedure of DENOISE to Select Best θ

In this section, we present the procedure for selecting the optimal threshold θ in the DENoise method. The goal is to assess the impact of varying θ values on performance across multiple datasets and domains, specifically MMLU, GSM8K, MathQA, and HumanEval. Each table provides a comprehensive comparison of the DENoise performance over different ranges of θ, from 50% to 100%, highlighting its effectiveness in selecting the most relevant experts in various domains.

C Threshold-Based Performance Analysis Across Datasets

This appendix provides a comprehensive analysis of the performance trends observed across varying threshold θ values for the datasets GSM8K, MathQA, HumanEval, MBPP, and MMLU. Each dataset, representing a distinct domain, showcases unique response patterns when applying the MoDE framework. As θ increases, we observe noticeable fluctuations in accuracy, highlighting the dynamic behavior of domain-specific subnetworks. The results consistently demonstrate that activating experts based on denoised domain-specific weights yields stable improvements across tasks. This analysis reinforces the scalability and adaptability of MoDE, validating its ability to enhance task-specific accuracy without the need for fine-tuning.

Table 7: Performance comparison of DENOISE throughout all MMLU domains with different θ.

cluster_id	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
mmlu_0	2047	57.79	58.72	58.96	59.26	59.99	59.94	60.67	60.23	60.77	60.92	60.92	14.58
mmlu_1	1518	35.57	34.72	34.32	35.70	35.38	35.31	35.84	35.44	36.17	35.77	36.17	10.81
mmlu_2	895	23.02	24.02	22.57	23.35	23.02	22.01	22.46	22.68	22.79	22.57	24.02	6.37
mmlu_3	1586	48.93	47.92	49.37	48.99	49.87	49.50	49.43	50.44	50.88	51.01	51.01	11.29
mmlu_4	212	27.83	28.77	28.30	25.94	28.77	26.89	27.83	26.89	27.36	26.89	28.77	1.51
mmlu_5	1477	59.58	59.04	60.39	59.51	60.12	60.80	60.93	61.61	61.61	61.75	61.75	10.52
mmlu_6	322	35.09	32.92	33.85	34.78	33.54	34.78	33.54	34.16	35.09	35.09	35.09	2.29
mmlu_7	434	27.65	25.58	29.95	26.96	26.96	27.19	28.11	29.72	27.19	29.49	29.95	3.09
mmlu_8	2016	55.21	55.36	55.46	55.51	56.15	55.56	56.35	57.04	57.19	57.44	57.44	14.36
mmlu_9	1839	46.66	47.53	46.82	46.49	47.36	47.74	47.36	48.02	48.45	48.29	48.45	13.09
mmlu_10	1174	30.92	30.15	31.26	31.09	30.49	30.15	30.83	32.03	32.28	31.86	32.28	8.36
mmlu_11	522	42.34	45.21	44.25	42.91	46.74	45.40	46.74	47.32	46.36	47.13	47.32	3.72

Table 8: Performance comparison of DENOISE throughout all GSM8K domains with different θ.

cluster_id	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
gsm8k_0	240	12.92	15.00	15.83	18.75	13.75	17.08	16.67	17.50	15.83	16.67	18.75	18.20
gsm8k_1	8	0.00	12.50	12.50	37.50	25.00	12.50	12.50	37.50	0.00	12.50	37.50	0.61
gsm8k_2	225	17.78	21.78	18.22	24.00	22.67	22.67	22.22	21.78	20.44	24.00	24.00	17.06
gsm8k_3	361	18.28	16.90	19.67	20.50	19.94	23.82	21.05	23.82	21.61	23.82	23.82	27.37
gsm8k_4	113	13.27	17.70	9.73	15.04	15.93	19.47	17.70	13.27	17.70	16.81	19.47	8.57
gsm8k_5	193	12.44	10.88	12.95	13.99	15.03	17.62	20.73	18.65	17.10	19.69	20.73	14.63
gsm8k_6	6	33.33	16.67	50.00	33.33	33.33	16.67	33.33	33.33	16.67	50.00	50.00	0.45
gsm8k_7	173	16.18	14.45	17.92	23.12	17.34	19.08	17.34	19.65	18.50	19.65	23.12	13.12

Table 9: Performance comparison of DENOISE throughout all MathQA domains with different θ.

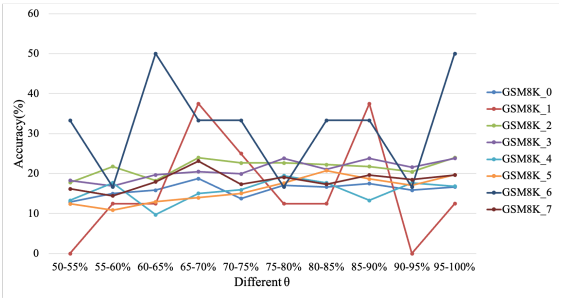
cluster_id	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
mathqa_0	289	19.03	19.72	20.42	23.53	26.99	28.72	26.30	22.84	26.30	22.15	28.72	9.68
mathqa_1	318	24.53	24.84	24.21	22.96	31.45	23.58	29.87	29.25	26.42	25.79	31.45	10.65
mathqa_2	453	20.75	22.30	27.15	22.96	24.06	25.39	23.18	26.49	25.39	22.96	27.15	15.18
mathqa_3	107	24.30	27.10	28.97	20.56	28.04	27.10	28.04	20.56	22.43	20.56	28.97	3.58
mathqa_4	238	24.37	20.17	29.83	21.01	22.69	29.41	22.69	27.31	29.41	28.15	29.83	7.97
mathqa_5	269	26.77	20.45	24.91	25.65	29.37	27.14	25.65	21.56	23.79	29.00	29.37	9.01
mathqa_6	659	24.28	19.58	20.49	21.40	20.64	22.91	21.55	18.97	20.64	19.88	24.28	22.08
mathqa_7	652	20.09	21.47	21.32	24.08	22.70	22.70	25.92	24.54	23.47	22.55	25.92	21.84

Table 10: Performance comparison of DENOISE throughout all HumanEval domains with different θ.

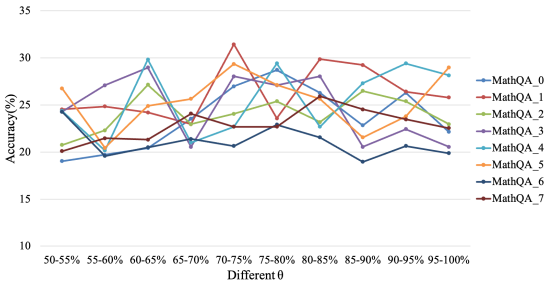
cluster_id	Metric	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
humaneval_0	pass@1	44	11.82	17.05	15.68	14.77	15.00	16.36	17.05	15.23	15.45	14.77	17.05	26.83
humaneval_1	pass@1	75	8.27	9.47	10.80	10.67	13.07	15.33	12.67	13.20	13.47	13.33	15.33	45.73
humaneval_2	pass@1	45	6.89	12.22	11.11	9.33	14.22	14.00	15.11	14.00	14.89	15.11	15.11	27.44
humaneval_0	pass@10	44	18.18	25.00	25.00	18.18	22.73	18.18	25.00	18.18	20.45	20.45	25.00	26.83
humaneval_1	pass@10	75	10.67	14.67	13.33	12.00	18.67	18.67	16.00	17.33	17.33	16.00	18.67	45.73
humaneval_2	pass@10	45	8.89	15.56	15.56	13.33	15.56	15.56	17.78	15.56	20.00	17.78	20.00	27.44

Table 11: Performance comparison of DENOISE throughout all MBPP domains with different θ.

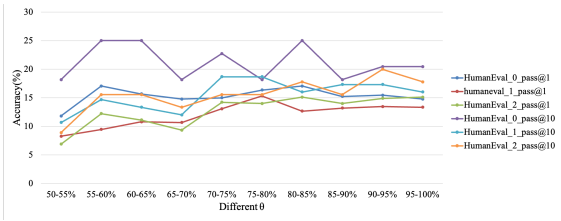
cluster_id	Metric	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
mbpp_0	pass@1	185	35.68	34.32	33.89	38.16	34.05	35.19	37.51	34.65	36.65	36.38	38.16	37.00
mbpp_1	pass@1	53	17.74	15.47	20.19	27.92	22.83	25.47	23.96	20.75	19.62	22.08	27.92	10.60
mbpp_2	pass@1	262	7.29	6.18	5.84	6.34	6.56	8.09	6.56	6.45	7.75	7.52	8.09	52.40
mbpp_0	pass@10	185	40.54	43.24	42.16	42.16	40.54	41.08	44.32	39.46	42.70	42.16	44.32	37.00
mbpp_1	pass@10	53	24.53	26.42	28.30	32.08	28.30	35.85	30.19	26.42	26.42	33.96	35.85	10.60
mbpp_2	pass@10	262	9.16	9.16	9.92	9.16	10.31	11.83	9.92	10.31	11.83	11.45	11.83	52.40



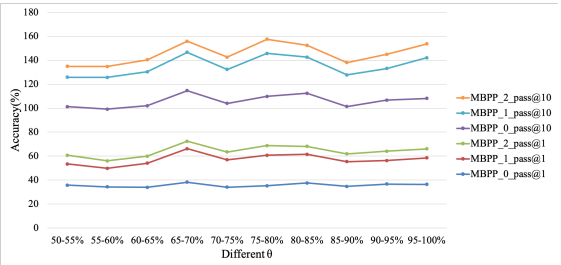
(a) GSM8K



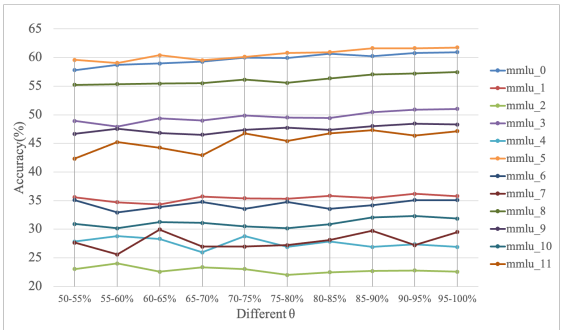
(b) MathQA



(c) HumanEval



(d) MBPP



(e) MMLU

Figure 5: Accuracy comparison of DENOISE across different thresholds θ for various datasets including GSM8K, MathQA, HumanEval, MBPP, and MMLU. Each subfigure (a-e) shows performance variations with respect to the θ values, highlighting dataset-specific accuracy trends.