FREE-MOE: Tuning-Free Mixture-of-Experts Purifying LLMs to Thrive across Any Field

Anonymous ACL submission

Abstract

The scaling up of pre-trained large language models entails increasing parameters and associated costs, while reducing parameters generally leads to a decrease in performance. Forming large language models into a Mixture of Experts (MoE) architecture demonstrates promising potential, as it not only reduces the tuning requirements of MoE but also allows for performance improvements. In this work, we introduce FREE-MOE, which leverages the inherent generalization ability of pre-trained LLMs across multiple tasks and domains, implementing weight purification to obtain Domain-Specific Subnetwork Experts. This method achieves performance improvements while 1) requiring no additional model parameters, and 2) being completely tuning-free for the experts. Specifically, we design the DOWP algorithm (Domain-Oriented Weight Purification Algorithm), which purifies the irrelevant weights in the hidden layers of the pre-trained LLM based on the input domain, forming domain-specific subnetwork experts. Additionally, FREE-MOE incorporates a multi-level trainable router to integrate DOWP into the pre-trained LLM, ensuring that only the most relevant subnetworks are activated. Findings show that the FREE-MOE not only improves the model's adaptability across various domains covered in contemporary language generation model benchmarks, but can also be seamlessly applied to any transformer-based LLM.

1 Introduction

The bigger the LLMs, the more parameters they cointai, the bigger they consume, which in turn limits their scalability and application efficiency (Patterson et al., 2021; Strubell et al., 2019). Yet, the tremendous success of Deepseek (Dai et al., 2024) proves the feasibility of employing the MoE architecture in pre-trained large language models to break through the Pareto frontier of model capacity and computational efficiency.

Previous work has primarily focused on techniques such as knowledge distillation(Hinton et al., 2015), parameter sharing (Rastegari et al., 2016), and pruning(Han et al., 2015), but these methods often lead to a degradation of the model's original performance, resulting in compromised model efficacy. However, sacrificing performance should not be the inevitable consequence of compressing large language models. Since pre-trained LLMs inherently function as implicit expert networks—while they do not explicitly employ an MoE framework, the subnetworks activated in their hidden layers demonstrate specialized behaviors that are task-dependent and input-specific. Therefore, we pose the following question:

Is it possible to leverage the dense pre-trained LLMs as Mixture of Experts to improve?

Building on this insight, by forming the pre-trained LLM into a tuning-free MoE framework, we not only achieve an extremely low training requirement that traditional MoE architectures cannot attain, but also improve its performance compared with baseline models.

We propose FREE-MOE, a novel tuning-free MoE architecture designed to leverage the existing subnetwork expert mechanisms within pre-trained LLMs. FREE-MOE identifies the input domain and selectively activate the most relevant Domain-specific subnetwork Experts (DSS-Experts) within a pre-trained LLM. Compared to Sparse MoE and Domain-Mapping & Random Gating MoE architectures, FREE-MOE achieves less pruning requirements and more task-specific accuracy, as shown in Figure 1. Specifically, purification represents the core solution for experts formation, where it purifies the hidden layer weights by identifying most-relevant features from the domain of input, retaining high-contributing weights while filtering out irrelevant ones. Lastly, in order to better activate purified experts, we introduce a Trainable Router

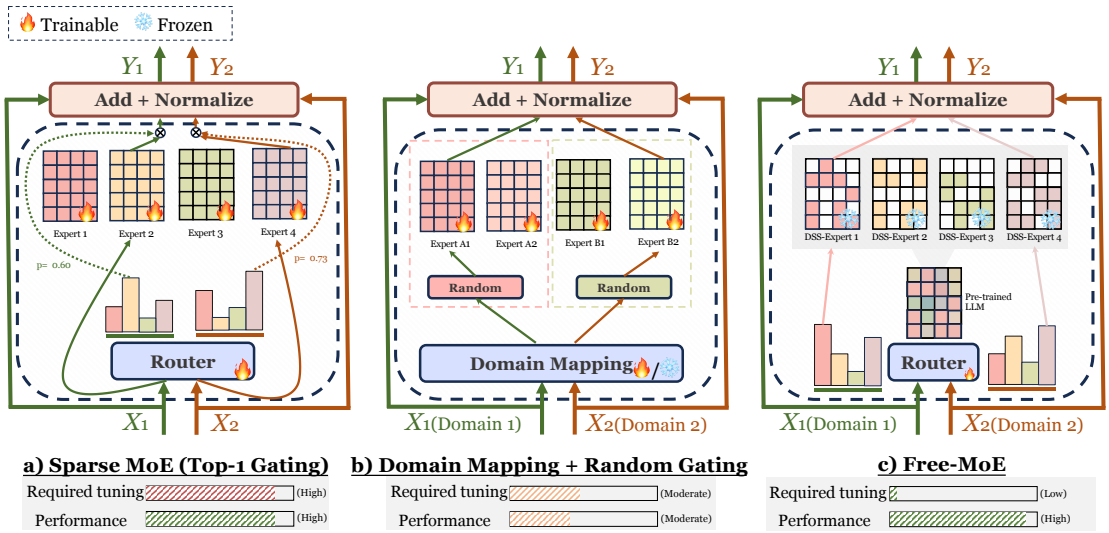


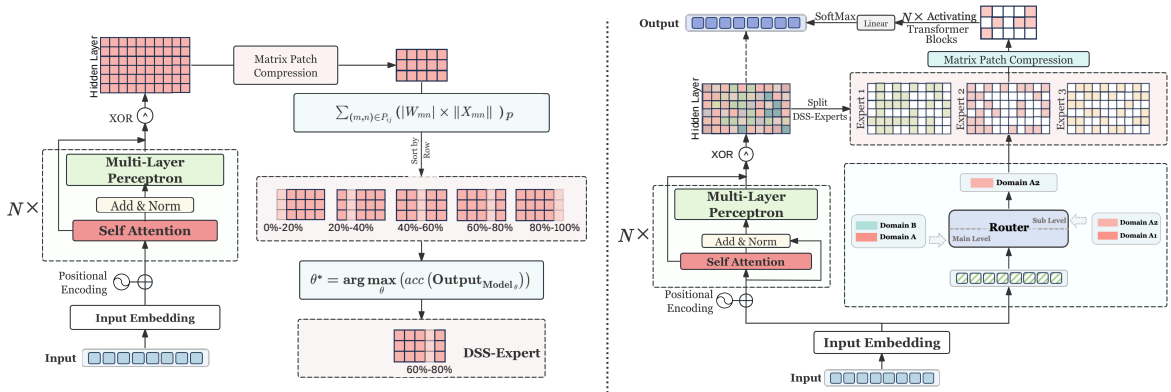
Figure 1: The comparison between traditional MoE methods and our FREE-MOE.(a) represents Sparse MoE with top-1 gating, which has higher training requirements and better performance. (b) shows the combination of Domain Mapping and Random Gating MoE, where this combination can improve training requirements but may sacrifice some performance. (c) illustrates our FREE-MOE, which activates Domain-Specific Subnetwork Experts from pretrained LLMs, achieving performance growth without the need for fine-tuning.

that dynamically monitors task requirements and domain characteristics, enabling real-time domain identification and efficient allocation. In this work, we present several key contribution:

- **LLM MoE Architecture FREE-MOE**: FREE-MOE utilizes the latent subnetwork structures within pre-trained LLMs, eliminating the tuning dependency of traditional MoE, enabling adaptation to multi-task requirements without the need for fine-tuning.
- **Purification Mechanism on Weights**: This mechanism dynamically selects and activates subnetwork weights relevant to specific tasks, significantly improving the accuracy of expert selection.
- **A Multi-Level Trainable Dynamic Router**: We introduce a novel trainable router capable of real-time monitoring of task requirements and domain characteristics to dynamically identify domains, particularly demonstrating significant advantages in multi-task scenarios.
- **Achieved Significant Performance Improvements Across Multiple Datasets**: Our approach achieved performance gains of 2% to 3% on datasets such as MMLU, MBPP, and GSM8K. Integrated into the FREE-MOE architecture, cumulative improvements of 1.11% validate the effectiveness and practicality of the method.

2 Related Works

The Mixture of Experts (MoE) introduces multiple specialized expert networks, selectively activating a subset of experts during each inference to maintain model capacity while significantly reducing computational costs (Jacobs et al., 1991; Jordan and Jacobs, 1994; Chen et al., 1999; Tresp, 2000; Rasmussen and Ghahramani, 2001). Sparse gating MoE layers (Shazeer et al., 2017; Riquelme et al., 2021), the GShard framework (Lepikhin et al., 2020), and Switch Transformer (Fedus et al., 2022) strike effective approaches. Focusing on gating, recent studies include methods like token-to-expert allocation BASE Layers (Lewis et al., 2021), hashing-based routing (Roller et al., 2021), distillation in routing Stablemoe (Dai et al., 2022) and Expert Choice Routing (Zhou et al., 2022). Several works have also explored training strategies for sparse MoE models. (Nie et al., 2021) proposed a dense-to-sparse gating strategy for better training efficiency. Additionally, approaches like expert regularization (Artetxe et al., 2021) and dimension reduction with L2 normalization (Chi et al., 2022) have been suggested to enhance fine-tuning in MoE models. Most recent advancements in MoE architectures include DeepseekMoE (Dai et al., 2024)and XMoE(Yang et al., 2024), which introduce fine-grained expert segmentation, while LLaMA-MoE by (Zhu et al., 2024) prunes from a dense LLM.



Step 1. Domain Oriented Weight Purification (1 layer) **Step 2. Free-MoE Structure Architecture**

Figure 2: Two-step pipeline for our approach. *Step 1* demonstrates the DOWP (Domain Oriented Weight Purification) architecture, where hidden layers extracted are compressed, sorted for domain-specific relevance via Equation 4, and further purifying MLP or Self-Attention Layers via Equation 6 to identify and retain the most relevant expert weights. *Step 2* illustrates the FREE-MOE pipeline, in which the router classifies the input by domain, assigning it to its identified DSS-Expert split from hidden layers to generate task-specific result.

LLM Pruning removes unnecessary parameters from a neural network to reduce its size and computational cost while maintaining or even improving performance. Unstructured pruning (Blalock et al., 2020; Frantar and Alistarh, 2023; Syed et al., 2023; Sun et al., 2024) involves the removal of individual weights based on specific criteria, leading to sparse networks that require specialized hardware for efficient execution. However, structured pruning (Wang et al., 2019; Kwon et al., 2022; Xia et al., 2022; Ma et al., 2023; Tao et al., 2023) eliminates entire structures such as neurons, layers, or attention heads, making it easier to implement on standard hardware. Moreover, expert pruning for Sparse Mixture of Experts (SMoE) models targets the pruning of individual expert networks (Lu et al., 2024), further enhancing the efficiency of MoE architectures without sacrificing performance, performing task-specific MoE (Chen et al., 2022) or through regularization (Muzio et al., 2024).

3 Method

3.1 Domain-Oriented Weight Purification

The Domain-Oriented Weight Purification, as shown in Figure 2, compresses matrix patches from hidden layers and ranks them by importance. Less relevant patches are purified, reducing complexity while preserving critical weights. This process forms the DSS-Expert for optimized task execution.

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:

$$D_j = \arg\max_{D_i \in \mathcal{D}} P(D_i \mid \mathcal{T}), \tag{1}$$

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 $\mathcal{F}_j = \{F_{j_1}, F_{j_2}, \ldots, F_{j_m}\}$, 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:

$$S^* = \arg\min_{S} \sum_{i=1}^{k} \sum_{F \in C_i} \|F - \mu_i\|_2, \tag{2}$$

where C_i denotes the set of points in the i-th cluster, and μ_i represents the centroid of cluster C_i. This optimization ensures that the data points in each cluster are tightly grouped around their respective centroid.

For any new task $\mathcal{T}$, its feature vector $F_{\mathcal{T}} = \text{Embedding}(\mathcal{T})$ is extracted and compared to the

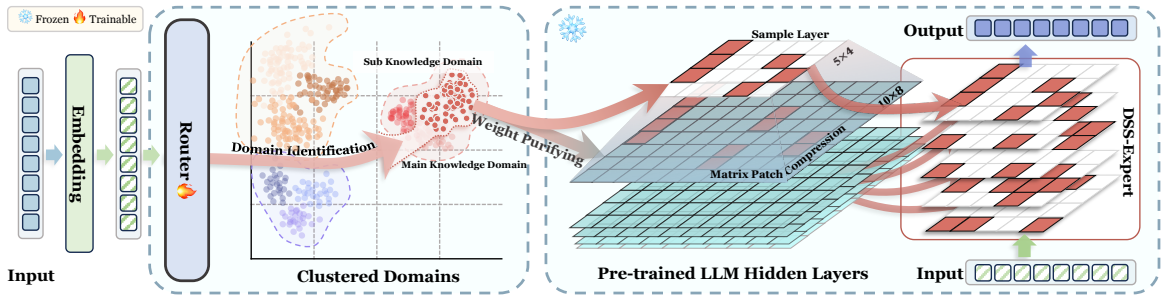


Figure 3: The inference flow through our FREE-MOE. The *input tokens* ( ) is first embedded and processed by a trainable ( ) router, where it is initially classified into the main knowledge domain and further into a sub knowledge domain. The *embedded input tokens* ( ) is then passed to the aggregated DSS-Expert, which are dynamically formed from the hidden layers of a frozen ( ) pre-trained LLM based on the identified domain, then *output tokens* ( ) are processed by DSS-Expert to output.

centroids μ_k of the subdomains. The task is assigned to the subdomain D_{j_k} that minimizes the Euclidean distance:

$$k = \arg\min_k \|F_{\mathcal{T}} - \mu_k\|_2. \quad (3)$$

Once assigned to the subdomain D_{j_k}, the task is further processed according to the knowledge characteristics.

Metric Calculation on Patches. After identifying the subdomain D_{j_k}, this section quantifies the critical information contained within it. Activation values serve as an effective metric for assessing the importance of neurons and connections within the network because they directly measure how strongly neurons respond to input features, reflecting their contribution to the network's output (Han et al., 2015). To perform this assessment, we begin by scaling the weight matrix 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_{ij}, 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 $\mathcal{M}_{ij}$ based on the l_p-norm:

$$\mathcal{M}_{ij} = \sum_{(m,n)\in P_{ij}} \left(|W_{mn}| \times \|X_{mn}\|_p\right). \quad (4)$$

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. Observations suggest that this metric is highly robust and can be reliably estimated with a small set of calibration samples.

DSS-Experts Formation. After calculating the importance scores based on the l_p-norm $\mathcal{M}_{ij}$ for all patches, these patches are sorted in ascending order based on their importance scores: $\mathcal{M}_{(1)} \leq \mathcal{M}_{(2)} \leq \cdots \leq \mathcal{M}_{(q)}$, where $\mathcal{M}_{(i)}$ denotes the sorted importance scores, and p represents the total number of patches. This sorting process helps identify the least important patches for purification. A threshold range θ_r is then defined to govern the purification process:

$$\theta_r = \left\{\mathcal{M}_{(i)} \mid \mathcal{M}_{(i)} \in [\theta_{\min}, \theta_{\max}]\right\}. \quad (5)$$

Here, $\theta_{\min}$ and $\theta_{\max}$ represent the lower and upper bounds of the importance scores targeted for purification. Patches within this range are considered less critical to the model's performance and are thus purified to reduce complexity while maintaining accuracy.

Following the purification process, the model's accuracy is re-evaluated to ensure minimal impact on performance. This accuracy, denoted as $\texttt{acc}(\theta_r)$, serves as an indicator of the purification's effectiveness. Subsequently, the outputs from each hidden layer are aggregated, incorporating the purified information to form the DSS-Expert.

The final step involves optimizing the purification threshold. By iterating through different threshold ranges θ_r, the model's accuracy is evaluated for each range, and the optimal threshold θ^* is determined:

$$\theta^* = \arg\max_{\theta_r} \left(\texttt{acc}(\texttt{Output}_{\text{MODEL}_{-\theta_r}})\right). \quad (6)$$

This selection of θ^* ensures that the model maintains maximum accuracy while removing the least

Algorithm 1 Pytorch-style pseudocode for DOWP Algorithm.

```
# D_k: domain-specific data
# theta_init: initial threshold
# theta_max: maximum threshold
# delta_theta: step size

def optimize_threshold(D_k, Perf, theta_init, theta_max,
      delta_theta):
    # Initialize variables
    theta = theta_init
    best_perf = 0
    theta_k_star = theta_init

    # Extract features and perform K-means clustering
    F_k = embedding(D_k) # Feature extraction
    subdomains = kmeans_clustering(F_k, k) # K-means
          clustering

    # Iterate over thresholds to find the optimal one
    while theta <= theta_max:
        # Calculate importance and select patches for
              purification
        selected_patches = select_patches(subdomains, theta)
              # Select patches based on importance

        # Purify model and evaluate performance
        purified_model = purify_model(selected_patches)
        current_perf = Perf(purified_model, D_k)

        # Update best performance and optimal threshold
        if current_perf > best_perf:
            best_perf = current_perf
            theta_k_star = theta

        theta += delta_theta # Increment threshold

    return theta_k_star # Return the optimal threshold
```

`Perf`: function to evaluate model performance on domain-specific data D_k.

significant patches, resulting in an efficient and accurate DSS-Expert, where $\mathcal{E}_{\text{DSS}} = \text{MODEL}_{-\theta^*}$.

3.2 FREE-MOE Architecture

The FREE-MOE Architecture, illustrated in Figure 2, utilizes a multi-level trainable router to dynamically classify tasks for DSS-Experts.

A Multi-level Trainable Router. We introduce a multi-level trainable router that classifies tasks through two hierarchical stages. First, the task embedding $F_{\mathcal{T}}$ is classified into a main knowledge domain D_j. Then, in the second stage, the task is further classified into a subdomain D_{j_k} within the selected main domain, as shown:

$$D_{j_k} = \mathcal{R}_{\text{sub},j}(\mathcal{R}_{\text{main}}(F_{\mathcal{T}})). \tag{7}$$

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

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

$$\begin{aligned}\mathcal{L}_{\mathcal{R}} &= \texttt{CrossEntropyLoss}(X) \\ &= -\sum_{k=1}^{m} y_k \log\left(P(D \mid X)\right), \end{aligned} \tag{8}$$

where y_k is the true label for domain D, and $P(D \mid X)$ represents the predicted probability of the input X being classified into domain D.

The multi-level trainable router classifies tasks hierarchically into main and subdomains. The classification process is optimized using cross-entropy loss at both levels, enhancing precision across domains.

FREE-MOE Inference. The inference process in FREE-MOE 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:

$$D_{m_i} = \mathcal{R}(\hat{X}). \tag{9}$$

After classification, the model proceeds `DOWP`. This step filters out irrelevant weights from the pre-trained LLM, retaining only those necessary for the selected domain. The purified weights, denoted as W_{purified}, form the core of the DSS-Expert:

$$W_{\text{purified}} = \texttt{DOWP}(W_{\text{pre-trained}}, D_{m_i}). \tag{10}$$

These purified 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 DSS-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 select three primary domains for our experiments: general, code, and mathematics. For the general domain, we use the MMLU benchmark(Hendrycks et al., 2021)to assess world knowledge and problem-solving across various subjects. In the code domain, we evaluate coding abilities using the MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) datasets, focusing on language comprehension and algorithmic reasoning. For mathematics, we utilize GSM8K(Cobbe

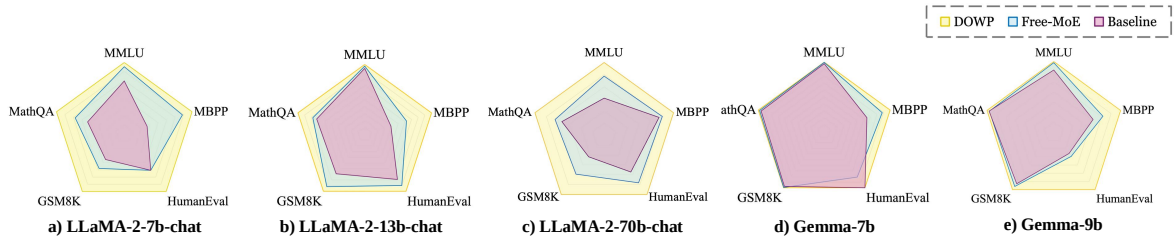


Figure 4: Performance comparison of different models applying DOWP and FREE-MOE. The radar charts illustrate the improvements across five datasets among DOWP, FREE-MOE, and baseline methods.

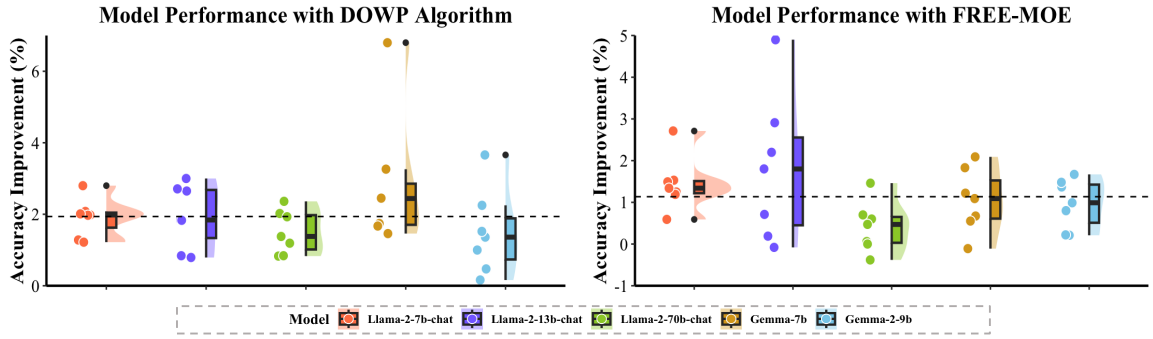


Figure 5: Accuracy improvement comparison of models using DOWP and Free-MoE methods across various architectures.

et al., 2021) and MathQA(Amini et al., 2019), which feature elementary-level problems in algebra and probability. We use the validation sets from MMLU, MBPP, MathQA, and GSM8K as reference data for our algorithm, with final evaluations on their respective test sets. For HumanEval, the MBPP validation set serves as the reference, with evaluation on the full HumanEval dataset. We employ a 5-shot evaluation for MMLU, providing the model five example questions and answers before predictions. For the other datasets, evaluations are conducted in a 0-shot setting.

Baseline & Foundation Models. Our experiments use five baseline models: LLaMA-2-7b-chat, LLaMA-2-13b-chat, LLaMA-2-70b-chat, Gemma-7b, and Gemma-2-9b. Both LLaMA-2 and Gemma models are based on transformer architecture, optimized for dialogue and natural language understanding. We selected LLaMA-2 and Gemma models, along with their various scales, to test the effectiveness and generalizability of our method on dense pre-trained models of different sizes.

Evaluation Metrics. We use accuracy (*acc%* ↑) as the primary evaluation metric across tasks, defined as the ratio of correctly answered questions to the total number of questions. For mathematics and general tasks, accuracy is the sole metric for assessing performance. For coding tasks, we evaluate model performance using pass@1 and pass@10. The pass@k(↑) metric allows the model to generate k different solutions for a given problem and measures the probability that at least one of these k solutions is correct. This is calculated as:

$$\textbf{pass@k} = 1 - \prod_{i=1}^{n} (1 - P_i), \qquad (11)$$

where P_i is the probability that the model's i-th solution is correct, and k is the number of trials.

4.2 Main Results

Our DOWP algorithm consistently boosts performance across datasets: MMLU, MBPP, HumanEval, GSM8K and MathQA. When incorporated into the FREE-MOE architecture, it further improves efficiency, adaptability and stability, as shown in Figure 4.

DOWP. Firstly, the application of DOWP using l_2-norm and 10% purification 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'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%

Table 1: Performance comparison of DOWP and FREE-MOE 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	BASELINE	45.81	19.24	23.60	14.45	19.51	20.24	25.33
	DOWP	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
	FREE-MOE	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	BASELINE	52.34	9.68	13.00	18.66	28.05	31.77	24.86
	DOWP	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
	FREE-MOE	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	BASELINE	66.87	45.22	66.15	30.52	59.34	59.47	35.12
	DOWP	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
	FREE-MOE	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
	DOWP	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
	FREE-MOE	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
	DOWP	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
	FREE-MOE	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

in MBPP (pass@10) and a 2.4% improvement in MathQA accuracy, demonstrating DOWP's efficacy across different models and tasks. Secondly, as shown in Figure 5, the performance generally lies within the improvement range of 2% to 3% across tasks, with the highest reaching up to 6.8%. The results suggest that DOWP enhances large-scale models by improving accuracy and maintaining stability. Its ability to purify domain-specific weights ensures efficient operation across diverse datasets and architectures.

FREE-MOE. To make further comparison, we evaluate the FREE-MOE architecture on the same set of LLMs to examine its effectiveness. Firstly, applying FREE-MOE also led to noticeable performance gains of 1.11% in average, though the improvements were generally more moderate compared to DOWP, as shown in Table 1. In detail, LLaMA-2-7b-chat'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 FREE-MOE primarily fall within the range of 0.5% to 2.5%, with the highest to 4.9%, as shown in Figure 5. The results suggest that FREE-MOE, while less impactful than DOWP in terms of absolute gains, offers a viable and stable method for enhancing model performance with minimal additional computation.

4.3 Ablation Studies

In the ablation studies, we use the LLaMA-2-7b-chat model due to its stable performance in previous results. Employing DOWP, we analyze l_p-norm, purification ratios, layers, and patch-square configurations. Furthermore, employing FREE-MOE, 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: DOWP Performance of Different p in l_p-norm on MMLU Dataset.

	p	MMLU(%)
Accuracy	1	46.26
	2	47.83
	4	47.27

Purification Ratio. We purify the all model layers (layers 0 to 31) using a 1×1 patch-square configuration and examine different ratios on the MMLU dataset categorized into 12 groups. With a 10% purification 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: DOWP Performance of Different Purification Ratio on MMLU Dataset.

	Ratio	MMLU(%)
Accuracy	base	45.81
	1%	47.75
	3%	47.83
	5%	47.79
	10%	47.83

Purification Sublayers. We then apply a 5% purification to MLP layers and Self-Attention layers, from layers 0 to 31, evaluating the impact on the GSM8K and MathQA, divided into 8 categories. Results shows purifying 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.

Patch-square Configuration. Besides, based on 5% purification ratio, we evaluate different patch-square configurations on the MBPP and HumanEval, divided into 3 categories. 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.

Table 4: DOWP Performance of Purifying Different Sublayers on GSM8K and MathQA Datasets.

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

Table 5: DOWP 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

K-means Clustering. Finally, we examine the impact of different K values applied in FREE-MOE in the K-means clustering step on the MMLU dataset, divided into 12 categories. We vary the number of clusters from 8 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: FREE-MOE Performance of Different K-means on MMLU Dataset.

	K	MMLU(%)
Accuracy	10	47.46
	12	47.79
	14	47.27

5 Conclusion

Conclusion. In this work, we introduced FREE-MOE, a novel framework designed effectively compress dense models, with the core of the DOWP Alg. FREE-MOE achieves 1) tuning-free, 2) highly portable, and 3) parameter efficiency, and can integrate into any transformer-based LLM without model-specific adjustments. Our method thus presents a promising solution for enhancing large model scalability and adaptability.

Limitation. Despite its effectiveness, FREE-MOE has several limitations. First, its reliance on pre-trained LLMs assumes well-separated expert subnetworks, which may not optimally handle tasks requiring cross-domain knowledge. Second, the K-means clustering used for domain classification may not always yield semantically meaningful partitions, potentially affecting expert selection. Third, while FREE-MOE reduces fine-tuning needs, its success depends on the quality of pre-training data, limiting performance in underrepresented domains. Fourth, evaluations are primarily based on benchmarks (MMLU, MBPP, GSM8K), which may not fully reflect real-world complexities. Future improvements could focus on refining domain clustering, adaptive expert selection, and optimizing routing efficiency.

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.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Jeff Wang, and 4 others. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684v1*.

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.

K. Chen, L. Xu, and H. Chi. 1999. Improved learning algorithms for mixture of experts in multiclass classification. *Neural Networks*, 12(9):1229–1252.

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.

Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. 2022. On the representation collapse of sparse mixture of experts. *arXiv preprint arXiv:2204.09179*.

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*.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Stablemoe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*.

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.

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

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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. In *Advances in Neural Information Processing Systems*.

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*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. *arXiv preprint arXiv:2103.16716v1*.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Alexandre Muzio, Alex Sun, and Churan He. 2024. Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts. *arXiv preprint arXiv:2404.05089*.

Xiaonan Nie, Shijie Cao, Xupeng Miao, Lingxiao Ma, Jilong Xue, Youshan Miao, Zichao Yang, Zhi Yang, and Bin Cui. 2021. Dense-to-sparse gate for mixture-of-experts. article, arXiv.

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*.

Carl Rasmussen and Zoubin Ghahramani. 2001. Infinite mixtures of gaussian process experts. In *Advances in neural information processing systems*, volume 14.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5254–5262.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Mario Neumann, Rodolphe Jenatton, Adrià Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*, volume 34, pages 8583–8595.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason E Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

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*.

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

Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2023. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10880–10895, Toronto, Canada. Association for Computational Linguistics.

Volker Tresp. 2000. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

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.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528, Dublin, Ireland. Association for Computational Linguistics.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. 2024. Enhancing efficiency in sparse models with sparser selection. *arXiv preprint arXiv:2403.18926*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y. Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*.

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:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$

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:

$$\theta_{ij} = \sum_{(m,n)\in P_{ij}} (|W_{mn}| \times \|X_{mn}\|_2) \tag{13}$$

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:

$$\text{MLP}(h) = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot h)) \tag{14}$$

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 DOWP to Select Best θ

In this section, we present the procedure for selecting the optimal threshold θ in the Domain-Oriented Weight Purification (DOWP) 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 DOWP 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 FREE-MOE 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 purified domain-specific weights yields stable improvements across tasks, while maintaining computational efficiency. This analysis reinforces the scalability and adaptability of FREE-MOE, validating its ability to enhance task-specific accuracy without the need for fine-tuning.

Table 7: Performance comparison of DOWP 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 DOWP 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 DOWP throughout all MathQA domains with different θ.

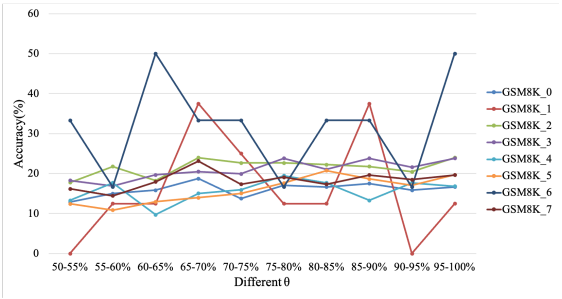
cluster_id	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
mathqa_0	289	19.03	19.72	20.42	23.53	26.99	28.72	26.30	22.84	26.30	22.15	28.72	9.68
mathqa_1	318	24.53	24.84	24.21	22.96	31.45	23.58	29.87	29.25	26.42	25.79	31.45	10.65
mathqa_2	453	20.75	22.30	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 DOWP throughout all HumanEval domains with different θ.

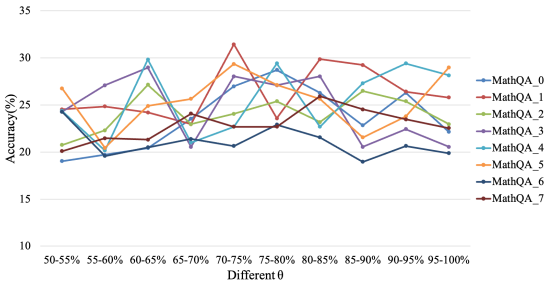
cluster_id	Metric	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
humaneval_0	pass@1	44	11.82	17.05	15.68	14.77	15.00	16.36	17.05	15.23	15.45	14.77	17.05	26.83
humaneval_1	pass@1	75	8.27	9.47	10.80	10.67	13.07	15.33	12.67	13.20	13.47	13.33	15.33	45.73
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 DOWP throughout all MBPP domains with different θ.

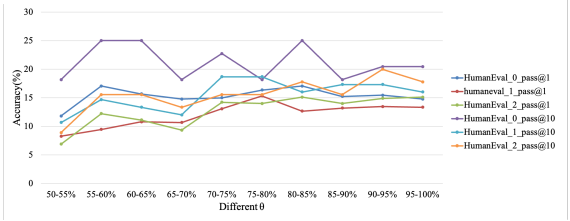
cluster_id	Metric	Samples	50-55%	55-60%	60-65%	65-70%	70-75%	75-80%	80-85%	85-90%	90-95%	95-100%	Max (%)	Ratio (%)
mbpp_0	pass@1	185	35.68	34.32	33.89	38.16	34.05	35.19	37.51	34.65	36.65	36.38	38.16	37.00
mbpp_1	pass@1	53	17.74	15.47	20.19	27.92	22.83	25.47	23.96	20.75	19.62	22.08	27.92	10.60
mbpp_2	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



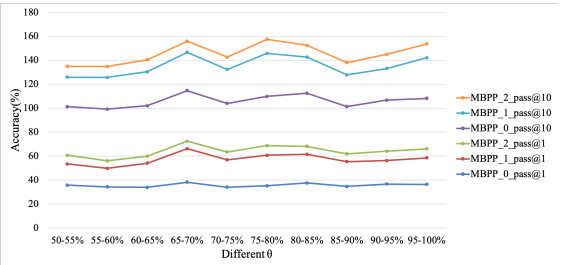
(a) GSM8K



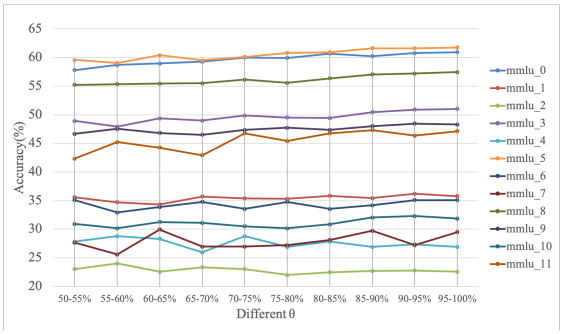
(b) MathQA



(c) HumanEval



(d) MBPP



(e) MMLU

Figure 6: Accuracy comparison of DOWP 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.