

Subspace Alignment Score: Locating Domain-Relevant Important Parameters in Large Language Models

Le Wu, Jiayi Wu[†], Jixiang Xiong, Mengfa Mo, Jingmin Xin

State Key Laboratory of Human-Machine Hybrid Augmented Intelligence,

Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University

{lew@stu., jiayi.wu@, xjx@stu., mengfa@stu., jxin@mail.}xjtu.edu.cn

Abstract

Although large language models (LLMs) possess powerful general capabilities, their insufficient domain expertise limits their application in high-reliability scenarios. The sparse activation patterns of neurons in LLM feed-forward layers suggest that structured fine-tuning strategies for domain adaptation can be constructed by identifying important parameters, thereby enhancing domain-specific performance while preserving general capabilities. However, due to high computational costs or weak scalability, current studies struggle to support large-scale and fine-grained domain-relevant parameter analysis. To address this, we investigate the distribution patterns and hierarchical evolution patterns of domain knowledge within the hidden state space of LLMs, and propose a novel parameter evaluation method: *Subspace Alignment Score*. Intervention experiments across three models and six domains illustrate that, compared with baseline methods, our method demonstrates superior performance in locating domain-relevant important parameters. Furthermore, structured fine-tuning strategies based on the important parameters achieve performance comparable to full fine-tuning while substantially reducing training costs, which is helpful for understanding efficient and interpretable domain adaptation. Our code is available at <https://github.com/lewu0723/subspace-alignment-score>.

1 Introduction

The massive parameter scale and abundant training data enable large language models (LLMs) to achieve outstanding performance in general language understanding and generation tasks (Brown et al., 2020; Touvron et al., 2023). However, when handling domain-specific tasks, their outputs often face challenges such as insufficient rigor or domain-specific knowledge hallucinations (Ye et al., 2023;

[†]Corresponding author: Jiayi Wu (jiayi.wu@xjtu.edu.cn)

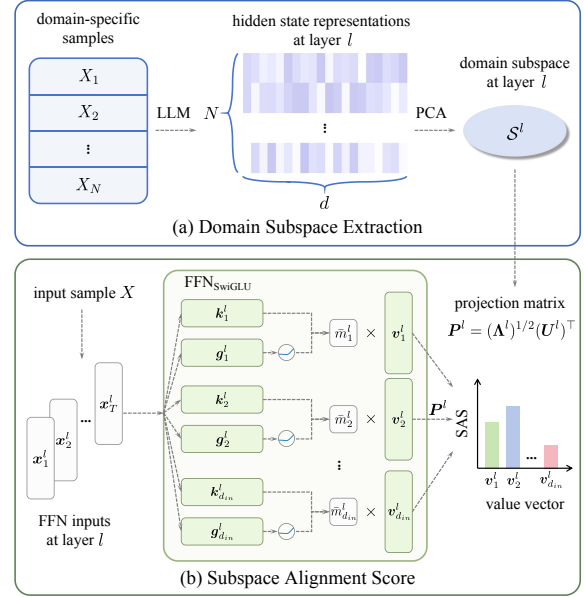


Figure 1: Illustration of our proposed method. (a) Domain Subspace Extraction. We use PCA to extract the domain subspaces $\mathcal{S}^l$ spanned by the domain-specific samples' hidden-state representations. (b) Subspace Alignment Score (SAS). SAS works at the layer level of LLMs, using the projection matrix $\mathbf{P}^l$ to project weighted value vectors $\{\bar{m}_j^l \mathbf{v}_j^l \mid j = 1, 2, \dots, d_{in}\}$ into the domain subspace $\mathcal{S}^l$ to quantify their contributions to the main information flow.

Huang et al., 2025). To improve model performance in specific domains, domain adaptation fine-tuning has become an important research direction. While full fine-tuning can maximally capture domain distribution characteristics, its prohibitive computational cost and susceptibility to "catastrophic forgetting" limit its practical applicability (Luo et al., 2025b). In contrast, Parameter-Efficient Fine-Tuning (PEFT) methods, such as LoRA (Hu et al., 2022) and Adapter (Houlsby et al., 2019), significantly reduce training overhead by introducing additional parameter modules. However, they constrain the expressiveness of parameter updates, exhibit high sensitivity to hyperparameter

settings, and struggle to be effectively deployed in complex domain tasks (Biderman et al., 2024; Luo et al., 2025a).

Recent studies indicate that neurons in the feed-forward layers of LLMs exhibit sparse activation patterns, suggesting that only a small subset of parameters is important for specific domains or tasks (Zhang et al., 2022; Szatkowski et al., 2026). By conducting domain-relevant analysis on parameter importance in LLMs, identifying parameters crucial to specific domains, and performing targeted fine-tuning exclusively on these important parameters, it is possible to significantly reduce training costs while enhancing domain performance and preserving general capabilities. This strategy offers a promising technical route for efficient and scalable domain adaptation.

Although recent studies have explored important parameters within LLMs, current methods still struggle to scale up to large-scale and fine-grained domain-relevant parameter analysis. First, methods based on causal intervention (Vig et al., 2020; Meng et al., 2022; Conmy et al., 2023) generally focus on attention layers or feed-forward layers associated with specific tasks or knowledge. Such methods rely on multiple forward and backward propagations to quantify module importance, and their excessive computational costs limit their application to fine-grained parameter analysis. Second, methods based on key-value memories (Geva et al., 2022; Yu and Ananiadou, 2024) exhibit unique advantages in fine-grained parameter investigation, but constrained by high manual interpretation costs, some methods only conduct analyses on a small set of parameters. Moreover, cross-layer analysis commonly relies on the unembedding matrix projection (Nostalgebraist, 2020). Due to the difference between representation distribution of LLM layers, direct parameter comparison and analysis inevitably introduce projection errors, which may further lead to misleading conclusions (Belrose et al., 2023).

In this paper, we start from the geometric structure of the hidden state space and find that when LLMs process texts from different domains, their hidden states distribute across distinct low-dimensional subspaces. Furthermore, this domain-relevant geometric structure gradually intensifies with model depth. Based on this discovery, we propose a novel parameter evaluation method: *Subspace Alignment Score* (SAS). Figure 1 provides an illustration of our method. Our method evaluates

parameter importance by measuring the alignment degree between parameter vectors and the domain subspace. Unlike methods that rely on the final output distribution, SAS conducts evaluation based on the geometric structure of the model’s internal information flow, providing a more stable reflection of the true contribution of parameter vectors to domain knowledge representation. Furthermore, we perform structured fine-tuning on the model using important parameters identified by SAS. This strategy achieves performance comparable to full fine-tuning while significantly reducing memory usage and computational overhead, and demonstrates better generalization capability. Overall, our contributions are as follows:

- We analyze the geometric structure of the hidden state space in LLMs, revealing the distribution patterns and hierarchical evolution patterns of domain knowledge within hidden state space.
- We propose a novel parameter evaluation method that, compared to baseline methods, more accurately locates domain-relevant important parameters.
- We conduct structured fine-tuning on domain-relevant important parameters, which achieves better generalization ability while significantly reducing training costs.

2 Related Work

Hidden Representation. Research on the hidden representations of large language models constitutes an important direction for understanding model behavior. Some studies employ linear probes (Alain and Bengio, 2016) to identify information encoded in hidden states, thereby revealing the latent knowledge representations learned by the model (Gurnee and Tegmark, 2024; Zhu et al., 2025; Jin and Rinard, 2024; Zhao et al., 2025). Motivated by the linear representation hypothesis (Park et al., 2024; Jiang et al., 2024), a series of work investigates directions in representation space that encode distinct concepts (Arditi et al., 2024; Park et al., 2025). These studies are also closely related to Representation Engineering (Zou et al., 2023), which seeks to understand and manipulate model behavior through interpretable directions and representations in the activation space. Sparse autoencoders have also been leveraged to disentangle hidden representations, with the goal of obtaining

more monosemantic features (Huben et al., 2024; Bricken et al., 2023; Templeton et al., 2024; Gao et al., 2025; Deng et al., 2025; Shu et al., 2025).

Mechanistic Interpretability. Mechanistic interpretability aims to reverse engineer the internal computational processes of Transformer at the level of underlying mechanisms (Olah, 2022). Some studies focus on identifying task-related circuits in large language models and providing functional interpretations of their behavior (Wang et al., 2023; Conmy et al., 2023; Syed et al., 2024; Bhaskar et al., 2024; Yao et al., 2024). Building on the view of FFN layers as key-value memories, some work analyzes the functional roles of neurons in FFN layers (Geva et al., 2021, 2022; Yu and Ananiadou, 2024; Yamamoto et al., 2026; Niu et al., 2024; Qiu et al., 2024). Causal intervention methods have also been employed to identify and analyze model components that play critical roles in model behavior (Vig et al., 2020; Meng et al., 2022; Marks et al., 2025).

3 Preliminary

3.1 Autoregressive Large Language Model

Given an input sequence of T tokens denoted as $X = [t_1, t_2, \dots, t_T]$, an autoregressive large language model first maps it to an initial hidden state $\mathbf{H}^0 \in \mathbb{R}^{T \times d}$ via the embedding matrix $\mathbf{E} \in \mathbb{R}^{B \times d}$. Subsequently, the hidden states are sequentially processed by L Transformer layers. The hidden state update for the token at position i in layer l can be formulated as:

$$\mathbf{h}_i^l = \mathbf{h}_i^{l-1} + \mathbf{a}_i^l + \mathbf{f}_i^l, \quad (1)$$

where $\mathbf{a}_i^l$ denotes the output of the multi-head self-attention (MHSA) layer, and $\mathbf{f}_i^l$ represents the output of the feed-forward network (FFN) layer. The final output probability distribution $\mathbf{y} \in \mathbb{R}^B$ is obtained by multiplying the hidden state of the last token at the final layer, $\mathbf{h}_T^L$, with the unembedding matrix $\mathbf{E}_u \in \mathbb{R}^{B \times d}$, followed by a softmax operation:

$$\mathbf{y} = \text{softmax}(\mathbf{E}_u \mathbf{h}_T^L). \quad (2)$$

The hidden state $\mathbf{h}$ traverses all layers of the model, serving as the primary conduit for the accumulation of contextual information and domain knowledge, also known as residual stream (Elhage et al., 2021).

3.2 FFN as Key-Value Memory

Most large language models employ the SwiGLU activation function within FFN layers (Shazeer, 2020). Given the input $\mathbf{x}_i^l = \mathbf{h}_i^{l-1} + \mathbf{a}_i^l$, the computation of the FFN layer is formulated as:

$$\mathbf{f}_i^l = \mathbf{W}_{\text{down}}^l \left(\text{Swish}(\mathbf{W}_{\text{gate}}^l \mathbf{x}_i^l) \odot (\mathbf{W}_{\text{up}}^l \mathbf{x}_i^l) \right), \quad (3)$$

where $\mathbf{W}_{\text{gate}}^l, \mathbf{W}_{\text{up}}^l \in \mathbb{R}^{d_{in} \times d}$ denote the gate-projection and up-projection matrices, respectively, and $\mathbf{W}_{\text{down}}^l \in \mathbb{R}^{d \times d_{in}}$ is the down-projection matrix. d_{in} denotes the intermediate size of the FFN layer. Geva et al. (2021) observed that the FFN layer functions as a key-value memory, and its output can be expressed as a weighted sum of the FFN parameter vectors:

$$\mathbf{f}_i^l = \sum_{j=1}^{d_{in}} m_{i,j}^l \mathbf{v}_j^l, \quad (4)$$

$$m_{i,j}^l = \text{Swish} \left((\mathbf{g}_j^l)^\top \mathbf{x}_i^l \right) \cdot \left((\mathbf{k}_j^l)^\top \mathbf{x}_i^l \right), \quad (5)$$

where $\mathbf{g}_j^l$ and $\mathbf{k}_j^l$ correspond to the j -th row of $\mathbf{W}_{\text{gate}}^l$ and $\mathbf{W}_{\text{up}}^l$, respectively (the key vectors), and $\mathbf{v}_j^l$ corresponds to the j -th column of $\mathbf{W}_{\text{down}}^l$ (the value vector). This perspective has inspired extensive research on the mechanistic interpretability of FFN parameters (Geva et al., 2022; Yu and Ananiadou, 2024; Ye et al., 2025).

4 Method

4.1 Domain Subspace Extraction

Prior research indicates that the hidden states of large language models predominantly reside in lower-dimensional subspaces, with each layer executing information read and write operations within distinct subspaces of the hidden state space (Elhage et al., 2021; Saglam et al., 2025; Wang et al., 2025).

We hypothesize that when processing domain-specific inputs, the hidden states are primarily distributed within domain-relevant low-dimensional subspaces. Due to the cumulative effect of residual connections, these subspaces exhibit significant layer-wise variations. Consequently, the geometric structure of the residual stream must be incorporated when evaluating the importance of FFN parameter vectors.

To characterize the domain-relevant geometric structure of the residual stream across layers, we extract the low-dimensional subspaces spanned by

the residual activations using the domain-specific dataset. Specifically, for the l -th layer, given N samples from the target domain, we collect their residual stream outputs and apply average pooling along the sequence dimension to obtain the hidden state representation matrix $\mathbf{H}^l \in \mathbb{R}^{N \times d}$. We then apply Principal Component Analysis (PCA):

$$\{\mathbf{U}^l, \mathbf{\Lambda}^l\} = \text{PCA}(\mathbf{H}^l), \quad (6)$$

where $\mathbf{U}^l \in \mathbb{R}^{d \times n}$ denotes the orthogonal basis comprising the top n principal component directions, and $\mathbf{\Lambda}^l = \text{diag}(\lambda_1, \dots, \lambda_n)$ is the corresponding diagonal matrix of variances. The subspace $\mathcal{S}^l$ spanned by $\mathbf{U}^l$ captures the dominant information directions of the residual stream at this layer.

In practice, we employ the Kneedle algorithm (Satopa et al., 2011) to detect the knee point of the cumulative variance curve, thereby adaptively determining the number of retained principal components n .

4.2 Subspace Alignment Score

To identify domain-relevant important parameters in LLMs, an intuitive approach is to locate parameters associated with high activations given domain-specific text inputs. However, neurons in LLMs are inherently polysemous (Templeton et al., 2024). Solely analyzing activations overlooks the critical interplay between parameter vectors and the geometric structure of the residual stream, thereby failing to capture deeper patterns of parameter contribution.

Inspired by recent work (Sun et al., 2024), we evaluate parameter importance by jointly considering activation magnitudes and their associated parameter vectors. Furthermore, by incorporating the geometric distribution of residual stream outputs across layers within the hidden state space, we characterize parameter contributions to the residual stream from a geometric perspective and propose a novel parameter evaluation metric: *Subspace Alignment Score* (SAS).

First, to preserve the relative importance of each principal component during alignment, we define the standard-deviation-weighted projection matrix:

$$\mathbf{P}^l = (\mathbf{\Lambda}^l)^{1/2} (\mathbf{U}^l)^\top. \quad (7)$$

This matrix can map parameter vectors into the domain subspace $\mathcal{S}^l$ and applies geometric scaling

Model	Layer	Dim	FFN Dim
LLaMA3.1-8B	32	4096	14,336
Olmo3-7B	32	4096	11,008
SmolLM3-3B	36	2048	11,008

Table 1: Model parameter configurations

proportional to the standard deviation of each principal component direction, thereby amplifying the contribution of high-information dimensions.

For a given input sample, we collect the FFN intermediate activations at layer l and apply average pooling along the sequence dimension, denoted as $\bar{\mathbf{m}}^l \in \mathbb{R}^{d_{\text{in}}}$, which are then used as scalar weights for the corresponding value vectors. Subsequently, the weighted value vectors are projected into the domain subspace $\mathcal{S}^l$ via $\mathbf{P}^l$. The ℓ_2 norm of the projected vector is computed and defined as the importance score for the corresponding value vector:

$$\text{SAS}(\mathbf{v}_j^l) = \|\mathbf{P}^l(\bar{\mathbf{m}}_j^l \mathbf{v}_j^l)\|_2. \quad (8)$$

Intuitively, the value vector $\mathbf{v}_j^l$ itself reflects its "projected contribution" within the domain subspace, while the activation intensity $\bar{\mathbf{m}}_j^l$ captures how strongly that direction is utilized by the model for the given input. By integrating both factors, SAS leverages the domain subspace projection $\mathbf{P}^l$ to attenuate noisy directions and accentuate parameter directions that drive the main information flow. While capturing critical internal information flow pathways, SAS effectively mitigates interference from local amplification effects, thereby providing a more robust assessment of parameter importance.

We compute the domain importance score for each value vector by averaging its SAS across all samples within the target domain.

5 Experiments

5.1 Experiment Setup

Models. We conducted experiments using three widely adopted large language models, LLaMA3.1-8B (Grattafiori et al., 2024), Olmo3-7B (Olmo et al., 2025), and SmolLM3-3B (Bakouch et al., 2025), all of which are autoregressive large language models. The specific parameter configurations are shown in Table 1.

Datasets. We used the arXiv metadata dataset (Clement et al., 2019) as the source of experimental corpus. This dataset contains

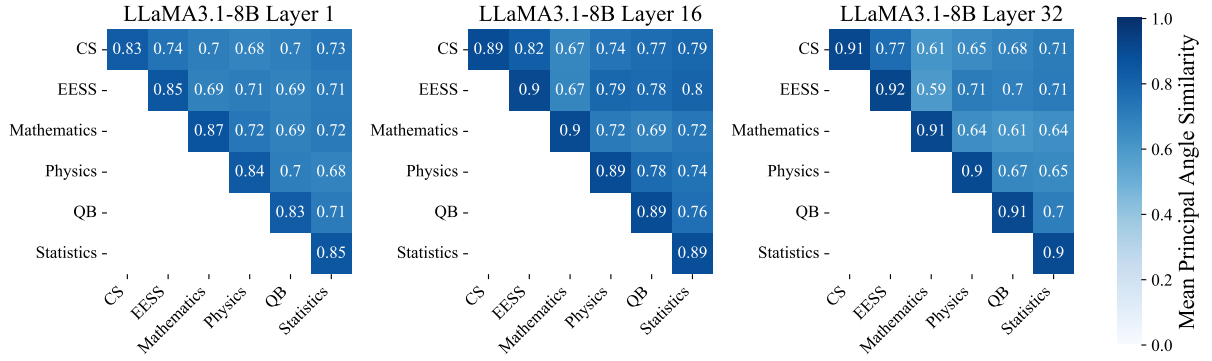


Figure 2: MPAS of domain subspaces across different domains in LLaMA3.1-8B. We show results for the first layer, the middle layer, and the final layer. Diagonal elements of the heatmaps correspond to comparisons using different texts from the same domain, while upper-triangular elements correspond to comparisons using texts from different domains.

metadata for over one million academic papers submitted to arXiv over the past three decades, including titles, authors, abstracts, and subject categories, and is both rich in content and high in quality. We select abstracts from six major domains—*Computer Science (CS)*, *Electrical Engineering and Systems Science (EESS)*, *Mathematics*, *Physics*, *Quantitative Biology (QB)*, and *Statistics*—as representative samples across diverse domains. For details about dataset collection and statistical summaries, please refer to Appendix A.

5.2 Domain Subspace Analysis

To verify the geometric properties of domain-relevant subspaces within the residual stream, we input domain-specific samples into the model and perform Principal Component Analysis on the residual stream outputs at each layer to extract domain subspaces. We use the Mean Principal Angle Similarity (MPAS) (Golub and Van Loan, 2013) to quantify the similarity between two subspaces. For an introduction to this metric, please refer to Appendix B.

Figure 2 shows the MPAS of domain subspaces across different domains of LLaMA-3.1-8B. It can be observed that when processing different texts from the same domain, subspaces at the same layer exhibit high similarity. Conversely, when processing texts from different domains, the similarity between subspaces at the same layer is significantly lower than within-domain results. Notably, in the deeper layers of the model, the gap between the MPAS of same-domain subspaces and that of cross-domain subspaces widens, indicating that the model exhibits stronger discriminative capability for domain-relevant hidden states in deeper layers.

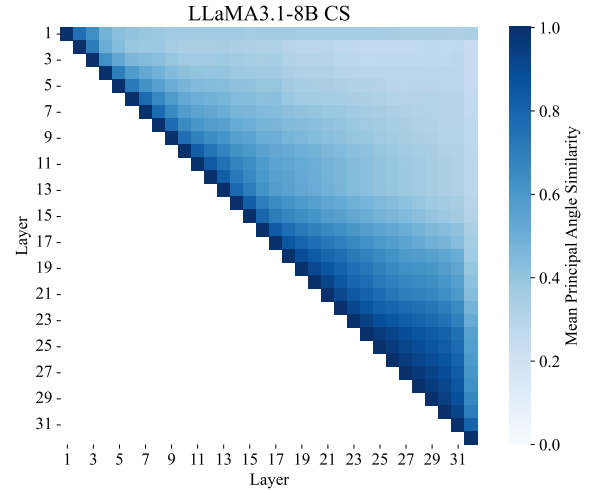


Figure 3: MPAS between domain subspaces across different layers in LLaMA3.1-8B when using text from CS domain.

These results suggest that LLMs differentiate domain knowledge representations within the hidden state space; specifically, knowledge from different domains tends to distribute across distinct low-dimensional subspaces. This empirically validates our hypothesis in Section 4.1.

Furthermore, we observe that the CS and EESS domains consistently exhibit relatively high MPAS across layers in the model, this phenomenon is observed in all test models. This indicates significant structural similarity in their internal representations. The study by Saglam et al. (2025) likewise shows that, in most models, the hidden state representations of CS and EESS are difficult to distinguish using linear classifiers. In contrast, the Mathematics domain shows lower MPAS with other domains across layers. A possible explanation is that the texts in this domain contains a large number of

Method	Model	Probability (%)						MRR ($\times 100$)					
		CS	EESS	Math.	Phys.	QB	Stat.	CS	EESS	Math.	Phys.	QB	Stat.
Origin		19.27	16.18	21.68	16.37	17.95	17.64	74.30	67.69	81.62	65.32	70.92	72.21
SUC	LLaMA	7.91	7.31	15.04	11.18	10.53	10.30	27.96	26.16	51.92	37.02	38.27	34.09
LPI		3.99	2.37	5.17	3.14	3.58	4.74	19.85	15.51	25.21	17.85	18.71	22.44
CULNIG		<u>1.54</u>	<u>0.79</u>	<u>1.34</u>	<u>1.52</u>	<u>1.12</u>	<u>1.71</u>	<u>8.51</u>	<u>6.37</u>	<u>8.23</u>	<u>8.77</u>	<u>8.12</u>	<u>8.46</u>
SAS		0.48	0.56	0.34	0.66	0.44	0.46	3.28	4.53	3.23	5.24	4.72	3.89
Origin		37.42	36.75	61.85	37.54	46.80	41.61	72.77	73.02	95.95	72.15	85.93	77.34
SUC	SmolLM	15.00	11.95	23.11	23.40	17.35	27.22	42.74	34.71	63.76	59.31	49.74	73.84
LPI		4.57	4.34	<u>4.62</u>	5.80	<u>4.25</u>	5.69	17.19	<u>17.98</u>	<u>19.75</u>	23.81	<u>17.27</u>	21.14
CULNIG		<u>3.45</u>	5.05	5.26	<u>2.42</u>	5.30	<u>4.15</u>	<u>16.11</u>	22.34	22.99	<u>14.00</u>	22.98	<u>20.68</u>
SAS		0.21	0.26	0.61	0.57	0.25	0.54	2.59	3.21	4.87	4.59	3.39	4.71

Table 2: Probability (%) and Mean Reciprocal Rank (MRR $\times 100$) of the correct answer when intervening the top-1 most important value vector per FFN layer across each domain. Results are averaged over 100 samples. **Bold** and underlined values indicate the lowest and second-lowest results, respectively. Lower values indicate more accurate localization of important parameters.

formula-related tokens, whose distributional properties differ substantially from those of other domains.

Additionally, Figure 3 shows the MPAS between domain subspaces across different layers in LLaMA-3.1-8B when inputting text from CS domain. It can be seen that adjacent layers exhibit high subspace similarity, and that similarity gradually decreases as the distance between layers increases. Meanwhile, in the deeper layers of the models, subspace similarity tends to stabilize, indicating structural convergence across layers. This suggests that LLMs exhibit clear hierarchical evolution patterns in the hidden state space: in shallow layers, subspace structures change rapidly to capture basic semantic and syntactic features of the input text; as depth increases, the subspaces gradually converge and stabilize, implying that the models complete the abstraction and integration of domain-relevant semantic representations in deeper layers.

For the experimental results of Olmo3-7B and SmolLM3-3B, please refer to Appendix F.1.

5.3 Important Parameters Localization

To evaluate the effectiveness of SAS in locating domain-relevant important parameters, we construct domain abstract samples into a domain abstract classification task (prompt template provided in the Appendix C) and measure the model’s comprehension of input samples using the probability

of outputting correct answers¹. We compare SAS with three representative baseline methods, including SUC (Geva et al., 2022), LPI (Yu and Ananidou, 2024) and CULNIG (Yamamoto et al., 2026). For detailed descriptions of these methods, please refer to Appendix D.

For each sample, we adopt each method to evaluate the importance of value vectors across all FFN layers. Subsequently, for each layer, only the top-1 most important value vector identified by each method is set to zero. We then rerun the model and evaluate the intervention effect using two metrics: the probability and reciprocal rank of the correct answer. A parameter evaluation method is considered effective if intervening the target value vector leads to a substantial drop in both metrics, indicating that the identified parameter is critical to the model’s internal knowledge representation. For each domain, we randomly select 100 samples, conduct the above experiment independently on each sample, and report the averaged results.

Table 2 shows the experimental results for LLaMA-3.1-8B and SmolLM3-3B across each domain. As observed, compared to other methods, our method induces a more significant drop in both probability and Mean Reciprocal Rank (MRR) across all domains. This demonstrates that our method locates important value vectors more ac-

¹Most baseline methods only consider the residual stream at last token of the sequence or rely on the model’s next-token prediction capability. Therefore, we construct domain abstract samples as a classification task to ensure a consistent evaluation protocol.

Method	Model	Probability (%)						MRR ($\times 100$)					
		CS	EESS	Math.	Phys.	QB	Stat.	CS	EESS	Math.	Phys.	QB	Stat.
Origin		19.27	16.18	21.68	16.37	17.95	17.64	74.30	67.69	81.62	65.32	70.92	72.21
Without P	LLaMA	0.85	<u>0.58</u>	<u>0.35</u>	0.96	0.48	<u>0.63</u>	6.84	<u>5.96</u>	<u>3.73</u>	8.45	5.50	<u>5.79</u>
Random P		<u>0.76</u>	0.64	0.45	<u>0.92</u>	<u>0.47</u>	0.65	<u>6.00</u>	6.26	4.10	<u>7.71</u>	<u>5.49</u>	5.80
SAS		0.48	0.56	0.34	0.66	0.44	0.46	3.28	4.53	3.23	5.24	4.72	3.89
Origin		37.42	36.75	61.85	37.54	46.80	41.61	72.77	73.02	95.95	72.15	85.93	77.34
Without P	SmolLM	<u>0.60</u>	<u>0.43</u>	0.97	<u>0.72</u>	<u>0.38</u>	0.38	4.46	3.69	<u>5.33</u>	<u>4.67</u>	3.10	<u>2.79</u>
Random P		0.74	<u>0.43</u>	<u>0.92</u>	0.82	<u>0.38</u>	0.38	<u>4.36</u>	<u>3.59</u>	<u>5.33</u>	5.17	<u>3.30</u>	2.76
SAS		0.21	0.26	0.61	0.57	0.25	<u>0.54</u>	2.59	3.21	4.87	4.59	3.39	4.71

Table 3: Comparison among Without P , Random P , and SAS. **Bold** and underlined values indicate the lowest and second-lowest results, respectively. Lower values indicate more accurate localization of important parameters.

curately. Intervening on only a tiny set of value vectors severely degrades model performance.

SUC focuses exclusively on the value vectors and the activation magnitudes of the last token, failing to effectively leverage overall sequence information. The effectiveness of LPI and CULNIG mainly depends on the probability of the correct token in the prediction distribution. When this probability is low, these methods struggle to reliably reflect parameter importance, imposing high demands on sample quality and prediction confidence.

SAS applies average pooling to FFN intermediate activations across the sequence dimension, comprehensively accounting for the activation intensity of value vectors over the entire input sequence. Additionally, SAS conducts layer-wise comparative analysis by projecting value vectors into the dominant information directions of that layer’s residual stream output. It aligns with the geometric structure of the layer’s main information flow rather than the model’s final output distribution. Consequently, our method operates independently of the unembedding matrix or specific prediction results, exhibits reduced sensitivity to input samples, and provides a more robust localization of important parameters.

To verify the effectiveness of domain subspace projection, we compare SAS with two ablation variants: without subspace projection (Without P) and random subspace projection (Random P). Experimental results in Table 3 demonstrate that SAS exerts a greater impact on the correct answer probability and reciprocal rank than Without P variant. In contrast, Random P variant cannot steadily improve localization accuracy and may even lead to

degraded localization performance.

For the experimental results of Olmo3-7B, please refer to Appendix F.2.

5.4 Important Parameter Fine-Tuning

To evaluate the practical utility of domain-relevant important parameters located by SAS in domain-specific fine-tuning, we select important parameter vectors at different ratios for fine-tuning and compare them with full fine-tuning and random parameter fine-tuning. Our method only locate the important parameter vectors within FFN layers. When fine-tuning partial parameter vectors of FFN layers, the attention layers and other parameter modules remain trainable.

We divide the abstract samples of various domains into the training set, validation set and test set at a ratio of 5:1:1. The training samples are used to compute SAS of all value vectors, and important parameter vectors are selected layer-wise for fine-tuning. We then select the model checkpoint with the best performance on the validation set and evaluate it on the test set.

We employ two metrics to evaluate the fine-tuning effectiveness of different strategies: (1) **Test Perplexity**, which measures the model’s comprehension of test samples; and (2) **Test/Train Gap**, defined as:

$$\Delta_{\text{gap}} = (\text{PPL}_{\text{test}}^{\text{ori}} - \text{PPL}_{\text{test}}^{\text{ft}}) - (\text{PPL}_{\text{train}}^{\text{ori}} - \text{PPL}_{\text{train}}^{\text{ft}}), \quad (9)$$

which quantifies the difference between performance gains on the test set and the training set. A higher Δ_{gap} indicates stronger generalization capability of the fine-tuning strategy. All experiments of the same model maintain identical hyperparam-

Domain	Origin	FT	Domain Important Params FT					Random Params FT-10%
			2%	4%	6%	8%	10%	
CS	11.1129	10.3194	10.3967	10.3781	10.3781	10.3752	10.3695	10.4457
		(-0.2909)	(-0.1176)	(-0.1156)	(-0.1272)	(-0.1311)	(-0.1357)	(-0.1137)
EESS	9.8979	9.1071	9.1977	9.1849	9.1811	9.1798	9.1760	9.2582
		(-0.2460)	(-0.0912)	(-0.0914)	(-0.0996)	(-0.1049)	(-0.1143)	(-0.0928)
Math.	6.2630	5.8389	5.8588	5.8547	5.8531	5.8507	5.8498	5.9065
		(-0.2038)	(-0.0794)	(-0.0815)	(-0.0841)	(-0.0861)	(-0.0893)	(-0.0796)
Phys.	10.5883	10.0085	10.0582	10.0468	10.0496	10.0440	10.0411	10.1102
		(-0.2888)	(-0.1031)	(-0.1064)	(-0.1164)	(-0.1186)	(-0.1273)	(-0.1040)
QB	10.8058	10.2324	10.2975	10.2907	10.2852	10.2825	10.2825	10.3411
		(-0.2561)	(-0.0970)	(-0.1034)	(-0.1046)	(-0.1055)	(-0.1140)	(-0.0924)
Stat.	10.1675	9.4848	9.5463	9.5379	9.5337	9.5295	9.5281	9.6252
		(-0.2826)	(-0.1091)	(-0.1154)	(-0.1218)	(-0.1242)	(-0.1321)	(-0.1167)

Table 4: Fine-tuning results of LLaMA-3.1-8B across each domain. We compare full fine-tuning (FT), domain important parameters fine-tuning at different ratios (2%–10%), and random parameters fine-tuning (10%). For each entry, the top value represents the **Test Perplexity**, and the bottom value (in green) represents the **Test/Train Gap**. "Origin" denotes the performance of the pre-trained model.

FT FFN Ratio (%)	Trainable Params (%)	Peak GPU Memory (GB)
2	31.20	38.88
4	32.61	39.54
6	34.01	40.16
8	35.41	40.81
10	36.82	41.59
100	100.00	74.84

Table 5: The proportion of trainable parameters and peak GPU memory when fine-tuning different ratios of FFN parameter vectors in LLaMA-3.1-8B.

ter settings. For more detailed settings, please refer to Appendix E.

Table 4 shows the fine-tuning results for LLaMA-3.1-8B. When fine-tuning important parameter vectors, the test perplexity progressively approaches that of full fine-tuning as the fine-tuning ratio increases. Due to the majority of parameters being frozen, fine-tuning only important parameters mitigates overfitting, thereby exhibits better generalization ability than full fine-tuning. In contrast, random fine-tuning requires tuning 10% of parameter vectors to approximate the performance achieved by fine-tuning merely 2% of important parameters, while also exhibiting inferior generalization. These results validate the accuracy of SAS in important parameter localization and the feasibility of domain-specific fine-tuning.

Given that each value vector in the FFN layer

corresponds to an independent intermediate neuron channel and is paired with a corresponding key vector, the SAS-based fine-tuning strategy constitutes a structured fine-tuning approach. We can split the FFN layer along intermediate neuron channels, fine-tuning only the weights associated with important parameters while freezing the remainder, thereby conserving GPU memory. Table 5 reports the proportion of trainable parameters and peak GPU memory when fine-tuning different ratios of FFN parameter vectors. It is evident that fine-tuning important parameter vectors significantly reduces computational and memory usage while preserving generalization performance, achieving robust fine-tuning effectiveness even under constrained computational resources.

Furthermore, we conduct a set of control experiments where all FFN layers are frozen, and only the parameters of the other modules are trained. Experimental results are reported in Appendix F.3.

For the experimental results of Olmo3-7B and SmoLM3-3B, please refer to Appendix F.4. Notably, the computation of SAS is highly efficient. We use hook functions to cache each layer’s residual stream outputs and FFN intermediate activations. For a single domain, only one forward propagation over all domain-specific samples is needed, incurring modest time, memory, and storage overhead. Detailed overhead statistics are also provided in Appendix F.4.

Domain	Original	FT-FFN -10%	LoRA	LoRA Matched	Adapter	Adapter Matched	(IA) ³
CS	11.1129	10.3695 (-0.1357)	10.5123 (-0.1427)	10.5022 (-0.2116)	10.4993 (-0.1366)	10.5269 (-0.1786)	10.5225 (-0.1446)
EESS	9.8979	9.1760 (-0.1143)	9.1760 (-1.9903)	9.1965 (-2.4255)	9.1747 (-1.9607)	9.1926 (-2.1833)	9.1747 (-2.2437)
Math.	6.2630	5.8498 (-0.0893)	5.9467 (-0.1021)	5.9459 (-0.1410)	5.9418 (-0.1022)	5.9698 (-0.1159)	5.9508 (-0.1015)
Phys.	10.5883	10.0411 (-0.1273)	10.1660 (-0.1540)	10.1675 (-0.2320)	10.1617 (-0.1543)	10.2056 (-0.1929)	10.1725 (-0.1554)
QB	10.8058	10.2825 (-0.1140)	10.3643 (-0.1193)	10.3534 (-0.1875)	10.3616 (-0.1192)	10.3725 (-0.1535)	10.3684 (-0.1202)
Stat.	10.1675	9.5281 (-0.1321)	9.6848 (-2.2137)	9.6606 (-0.2070)	9.6834 (-2.1814)	9.7119 (-0.1931)	9.6848 (-2.1796)

Table 6: Fine-tuning results of LLaMA-3.1-8B across each domain. We compare fine-tuning 10% of important parameter vectors in FFN layers against LoRA, Adapter, (IA)³, as well as LoRA and Adapter with matched trainable parameter counts. For each entry, the top value represents the **Test Perplexity**, and the bottom value (in green) represents the **Test/Train Gap**. "Origin" denotes the performance of the pre-trained model.

FT Strategy	Trainable Params (%)	Peak GPU Memory (GB)
FT-FFN-10%	36.82	41.59
LoRA	23.06	47.19
LoRA-Matched	26.91	55.68
Adapter	29.95	38.30
Adapter-Matched	34.41	42.71
(IA) ³	22.96	44.68

Table 7: The proportion of trainable parameters and peak GPU memory under different fine-tuning strategies in LLaMA-3.1-8B.

To further verify the effectiveness of our method in more complex domain-adaptation scenarios, we conduct evaluations on MedQA (Jin et al., 2021) and CaseHold (Zheng et al., 2021). Experimental results are provided in Appendix F.5.

5.5 Comparison with PEFT Methods

To evaluate the advantages of the SAS-based fine-tuning strategy over existing PEFT methods, we compare it against LoRA (Hu et al., 2022), Adapter (Houlsby et al., 2019), and (IA)³ (Liu et al., 2022). Furthermore, we have added another set of LoRA and Adapter with matched trainable parameter counts. For more detailed settings, please refer to Appendix E.

Table 6 shows the fine-tuning results for LLaMA-3.1-8B. Table 7 shows the proportion of trainable parameters and peak GPU memory for different fine-tuning strategies. Our method achieves su-

perior fine-tuning performance while maintaining peak GPU memory comparable to other methods, which validates the effectiveness of the SAS-based fine-tuning strategy.

It is worth noting that existing PEFT methods typically freeze the backbone network parameters and train by adding extra parameter modules. This is not a substitute for the SAS-based fine-tuning strategy; rather, the two can be combined to explore more efficient fine-tuning strategies.

6 Conclusion

In this work, we investigate the geometric structure of domain knowledge within the hidden state spaces of large language models. Our analysis reveals that hidden states from different domains form distinct low-dimensional subspaces, and that such domain-relevant geometric structures become increasingly stable in deeper layers of the model. Based on this observation, we propose SAS, a parameter evaluation method for identifying domain-relevant important parameters. Extensive experiments demonstrate that SAS achieves accurate and robust parameter localization across different models and domains. Furthermore, the SAS-based fine-tuning strategy achieves performance comparable to full fine-tuning while substantially reducing memory usage and computational overhead. This fine-tuning strategy provides a promising technical route for low-cost, high-efficiency domain adaptation of LLMs.

7 Limitations

Although our work has achieved certain progress, we also recognize its limitations. First, our work only focuses on value vectors in FFN layers, excluding parameters in attention layers and other modules. A more comprehensive analysis is still required to explore their contributions to model representations. Second, the extraction of domain subspaces relies on the forward propagation of domain samples and layer-wise PCA computation. Although this procedure is performed only once offline and incurs relatively small overhead in practice, it still introduces an initial computational and storage cost when processing large-scale datasets.

8 Acknowledgements

This work is supported by the National Key Research and Development Program of China (Grant No. 2023ZD0121300).

References

- Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.
- Andy Ardit, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. [Refusal in language models is mediated by a single direction](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 136037–136083. Curran Associates, Inc.
- Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Noumane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, and 4 others. 2025. SmolLM3: smol, multilingual, long-context reasoner. <https://huggingface.co/blog/smolLM3>.
- Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.
- Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. [Finding transformer circuits with edge pruning](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. [LoRA learns less and forgets less](#). *Transactions on Machine Learning Research*. Featured Certification.
- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. [Towards monosemanticity: Decomposing language models with dictionary learning](#). *Transformer Circuits Thread*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Colin B Clement, Matthew Bierbaum, Kevin P O’Keeffe, and Alexander A Alemi. 2019. On the use of arxiv as a dataset. *arXiv preprint arXiv:1905.00075*.
- Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. [Towards automated circuit discovery for mechanistic interpretability](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Boyi Deng, Yu Wan, Baosong Yang, Yidan Zhang, and Fuli Feng. 2025. [Unveiling language-specific features in large language models via sparse autoencoders](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4563–4608, Vienna, Austria. Association for Computational Linguistics.
- Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. <https://transformer-circuits.pub/2021/framework/index.html>.
- Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. [Scaling and evaluating sparse autoencoders](#). In *The Thirteenth International Conference on Learning Representations*.
- Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. [Transformer feed-forward layers build](#)

- predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. [Transformer feed-forward layers are key-value memories](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Wes Gurnee and Max Tegmark. 2024. [Language models represent space and time](#). In *The Twelfth International Conference on Learning Representations*.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. [Parameter-efficient transfer learning for nlp](#). In *International Conference on Machine Learning*.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. [A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions](#). *ACM Trans. Inf. Syst.*, 43(2).
- Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. [Sparse autoencoders find highly interpretable features in language models](#). In *The Twelfth International Conference on Learning Representations*.
- Yibo Jiang, Goutham Rajendran, Pradeep Kumar Ravikumar, Bryon Aragam, and Victor Veitch. 2024. [On the origins of linear representations in large language models](#). In *Forty-first International Conference on Machine Learning*.
- Charles Jin and Martin Rinard. 2024. [Emergent representations of program semantics in language models trained on programs](#). In *Forty-first International Conference on Machine Learning*.
- Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. [What disease does this patient have? a large-scale open domain question answering dataset from medical exams](#). *Applied Sciences*, 11(14).
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohata, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. [Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Dan Luo, Kangfeng Zheng, Chunhua Wu, and Xiujuan Wang. 2025a. [Mslora: Meta-learned scaling for adaptive fine-tuning of lora](#). *Neurocomput.*, 643(C).
- Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025b. [An empirical study of catastrophic forgetting in large language models during continual fine-tuning](#). *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786.
- Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. [Sparse feature circuits: Discovering and editing interpretable causal graphs in language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. [Locating and editing factual associations in GPT](#). In *Advances in Neural Information Processing Systems*.
- Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. [What does the knowledge neuron thesis have to do with knowledge?](#) In *The Twelfth International Conference on Learning Representations*.
- Nostalgebraist. 2020. [interpreting gpt: the logit lens](#).
- Chris Olah. 2022. [Mechanistic interpretability, variables, and the importance of interpretable bases](#). *Transformer Circuits Thread*.
- Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.
- Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. 2025. [The geometry of categorical and hierarchical concepts in large language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. [The linear representation hypothesis and the geometry of large language models](#). In *Forty-first International Conference on Machine Learning*.

- Zihan Qiu, Zeyu Huang, Youcheng Huang, and Jie Fu. 2024. [Empirical study on updating key-value memories in transformer feed-forward layers](#). In *The Second Tiny Papers Track at ICLR 2024*.
- Baturay Saglam, Paul Kassianik, Blaine Nelson, Sajana Weerawardhena, Yaron Singer, and Amin Karbasi. 2025. [Large language models encode semantics and alignment in linearly separable representations](#). In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2282–2303, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.
- Ville Satopa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *31st International Conference on Distributed Computing Systems Workshops*, pages 166–171.
- Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.
- Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. [A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1690–1712, Suzhou, China. 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, Can Rager, and Arthur Conmy. 2024. [Attribution patching outperforms automated circuit discovery](#). In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, Miami, Florida, US. Association for Computational Linguistics.
- Filip Szatkowski, Patryk Będkowski, Alessio Devoto, Jan Dubiński, Pasquale Minervini, Mikołaj Piórczyński, Simone Scardapane, and Bartosz Wójcik. 2026. [Universal properties of activation sparsity in modern large language models](#). In *The Fourteenth International Conference on Learning Representations*.
- Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermy, and 3 others. 2024. [Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet](#). *Transformer Circuits Thread*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. [Investigating gender bias in language models using causal mediation analysis](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.
- Junxuan Wang, Xuyang Ge, Wentao Shu, Zhengfu He, and Xipeng Qiu. 2025. Attention layers add into low-dimensional residual subspaces. *arXiv preprint arXiv:2508.16929*.
- Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. [Interpretability in the wild: a circuit for indirect object identification in GPT-2 small](#). In *The Eleventh International Conference on Learning Representations*.
- Taisei Yamamoto, Ryoma Kumon, Danushka Bollegala, and Hitomi Yanaka. 2026. [Neuron-level analysis of cultural understanding in large language models](#). In *The Fourteenth International Conference on Learning Representations*.
- Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. [Knowledge circuits in pretrained transformers](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794*.
- Mengyu Ye, Jun Suzuki, Tatsuro Inaba, and Tatsuki Kuribayashi. 2025. [Transformer key-value memories are nearly as interpretable as sparse autoencoders](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Zeping Yu and Sophia Ananiadou. 2024. [Neuron-level knowledge attribution in large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280, Miami, Florida, USA. Association for Computational Linguistics.
- Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. [MoEification: 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.
- Haiyan Zhao, Heng Zhao, Bo Shen, Ali Payani, Fan Yang, and Mengnan Du. 2025. [Beyond single concept vector: Modeling concept subspace in LLMs with gaussian distribution](#). In *The Thirteenth International Conference on Learning Representations*.

Lucia Zheng, Neel Guha, Brandon R Anderson, Peter Henderson, and Daniel E Ho. 2021. When does pre-training help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the eighteenth international conference on artificial intelligence and law*, pages 159–168.

Fangwei Zhu, Damai Dai, and Zhifang Sui. 2025. [Language models encode the value of numbers linearly](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 693–709, Abu Dhabi, UAE. Association for Computational Linguistics.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Experimental Dataset

We conducted experiments using the arXiv meta-data dataset. According to the arXiv Category Taxonomy², the dataset can be divided into eight major domains: *Computer Science*, *Economics*, *Electrical Engineering and Systems Science*, *Mathematics*, *Physics*, *Quantitative Biology*, *Quantitative Finance*, and *Statistics*. We removed samples with multiple primary category labels to avoid interference caused by cross-domain semantic overlap. Considering that the *Economics* and *Quantitative Finance* categories contain relatively few samples (fewer than 6,000 each), we ultimately retained data from the remaining six domains for our experiments.

For each subject domain, we randomly sample 14,000 paper abstracts and impose constraints on abstract length, requiring word counts to fall within the range of 100 to 500 tokens. This criterion ensures adequate preservation of semantic information while avoiding excessive computational overhead associated with extremely long texts. Apart from basic string cleaning operations (e.g., removal of invalid whitespace characters), we apply no additional preprocessing to the text, thereby preserving the natural distributional properties of the original corpus to the greatest extent possible. Detailed statistical information is summarized in Table 8.

Domain	Word Count		
	Min	Max	Mean
CS	100	470	178.36
EESS	100	446	175.75
Mathematics	100	484	148.45
Physics	100	453	171.84
QB	100	397	188.93
Statistics	100	492	176.39

Table 8: Word count statistics of arXiv abstract datasets for each domain.

B Mean Principal Angle Similarity

Given two linear subspaces $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^d$ with dimensions p and q , respectively, let $r := \min(p, q)$. The i -th principal angle $\theta_i \in [0, \frac{\pi}{2}]$ is defined such

that:

$$\begin{aligned} \cos \theta_i &= \max_{\mathbf{a} \in \mathcal{S}_1, \mathbf{b} \in \mathcal{S}_2} \mathbf{a}^\top \mathbf{b}, \\ \text{s.t. } \|\mathbf{a}\| &= \|\mathbf{b}\| = 1, \\ \mathbf{a}^\top \mathbf{a}_j &= \mathbf{b}^\top \mathbf{b}_j = 0, \quad j = 1, \dots, i-1. \end{aligned} \quad (10)$$

The principal angles satisfy $0 \leq \theta_1 \leq \dots \leq \theta_k \leq \frac{\pi}{2}$. Intuitively, θ_1 represents the closest direction between the two subspaces, while the subsequent angles $\theta_2, \dots, \theta_k$ are defined under the constraint of orthogonality to the previously identified directions. Based on the cosines of the principal angles, we employ the Mean Principal Angle Similarity (MPAS) to quantify the similarity between the two subspaces:

$$\text{MPAS}(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{r} \sum_{i=1}^r \cos \theta_i. \quad (11)$$

The value of MPAS lies in the interval $[0, 1]$, with higher values indicating greater similarity between the two subspaces.

C Prompt Template

We use the following prompt template to construct domain abstract classification tasks from domain abstract samples. The model’s comprehension of input samples is measured by the probability of generating correct answers, with category options randomly shuffled for each sample.

Classify the following abstract into exactly one category. Without any explanation, output only the letter of the correct option.

Abstract:

{domain_abstract}

Categories:

- A. Physics
- B. Statistics
- C. Quantitative Biology
- D. Electrical Engineering and Systems Science
- E. Mathematics
- F. Computer Science

Answer:

D Baseline Methods

We formalize the baseline methods from Section 5.3 using the notation defined in Section 3. For simplicity, we use m_j^l to refer to the activation

²https://arxiv.org/category_taxonomy

intensity of the value vector at the last token of the sequence, i.e., $m_{T,j}^l$. The baseline methods are formalized as follows:

SUC. Geva et al. (2022) quantify the relative contribution of each FFN Sub-Update $m_j^l v_j^l$ by normalizing the product of the activation magnitude and the parameter vector norm:

$$\text{SUC}(m_j^l v_j^l) = \frac{|m_j^l| \cdot \|v_j^l\|_2}{\sum_{k=1}^{d_{in}} |m_k^l| \cdot \|v_k^l\|_2}. \quad (12)$$

LPI. Yu and Ananiadou (2024) quantify the importance of value vectors for predicting the target token via the Log Probability Increase(LPI):

$$\text{LPI}(v_j^l) = \log P(w|m_j^l v_j^l + x_i^l) - \log P(w|x_i^l). \quad (13)$$

CULNIG. Yamamoto et al. (2026) propose the CULture Neuron Identification Pipeline with Gradient-based Scoring (CULNIG) to analyze the contribution of parameter vectors. They focus on parameter vectors from the gate-projection matrix, as gating neurons function as switches that control weight transmission. The attribution score for a parameter vector at position i is defined as the product of its activation and the gradient of the output probability:

$$s^{(l,j,i)}(X, w) = n^{(l,j,i)} \cdot \frac{\partial P(w|X)}{\partial n^{(l,j,i)}}, \quad (14)$$

where $n^{(l,j,i)}$ denotes the output of the gate-projection matrix. The final importance score is obtained via max pooling over the sequence dimension:

$$s^{(l,j)}(X, w) = \max_i s^{(l,j,i)}(X, w). \quad (15)$$

E Hyperparameter Settings

We employ the same hyperparameter settings across all experiments for the same model in Section 5.4. We use the AdamW optimizer (Loshchilov and Hutter, 2019) and cosine learning rate schedule. Both the warmup ratio and weight decay are set to 0.1. The effective training batch size is 64, and training is conducted for 2 epochs. The learning rate is set to 5×10^{-6} for LLaMA3.1-8B, and 1×10^{-5} for both SmolLM3-3B and Olmo3-7B.

For the experiments in Section 5.5, we use $r = 8$ and $\alpha = 32$ for LoRA, and set the hidden dimension to 64 for Adapter. The rank of LoRA-Matched and the hidden dimension of Adapter-Matched are set such that their trainable parameter counts are

comparable to those of our method. Specifically, for LoRA-Matched, we set $r = 318$ and $\alpha = 636$, while for Adapter-Matched, we set the hidden dimension to 2150. Since these additional parameter modules are randomly initialized and trained from scratch, we use a higher learning rate of 5×10^{-5} for these modules in LLaMA3.1-8B. All other experimental settings remain the same as those in Section 5.4.

F More Experimental Results

F.1 Domain Subspace Analysis

We conduct the same domain subspace analysis on Olmo3-7B and SmolLM3-3B. Figure 4 shows the MPAS of domain subspaces across different domains of Olmo3-7B and SmolLM3-3B. Figure 5 shows the MPAS between domain subspaces across different layers in Olmo3-7B and SmolLM3-3B when inputting text from CS domain. The domain-relevant subspaces in the residual streams of these two models exhibit a consistent pattern with LLaMA3.1-8B, indicating that the geometric properties of the domain-relevant subspaces are robust under different model architectures and scales.

F.2 Important Parameters Localization

We conduct the same important parameters locating experiment on Olmo3-7B. Table 9 shows the comparison with baseline methods across each domain, while Table 10 shows the results against ablation variants. Intervening on the important value vectors in Olmo3-7B substantially reduces both the probability and reciprocal rank of the correct answer. While the performance drop is less severe compared to LLaMA3.1-8B and SmolLM3-3B, our method still achieves the largest or second-largest performance drops across the evaluated metrics. These results further demonstrate the robustness of our method in locating domain-relevant important parameters.

F.3 Fine-Tuning Control Experiments

We conduct control experiments where all FFN layers are frozen, and only the parameters of the other modules are trained. Table 11 shows the test perplexity for each domain after fine-tuning on various models. Table 12 shows the proportion of trainable parameters and peak GPU memory under different experimental configurations. Compared to fine-tuning only the parameters of the other mod-

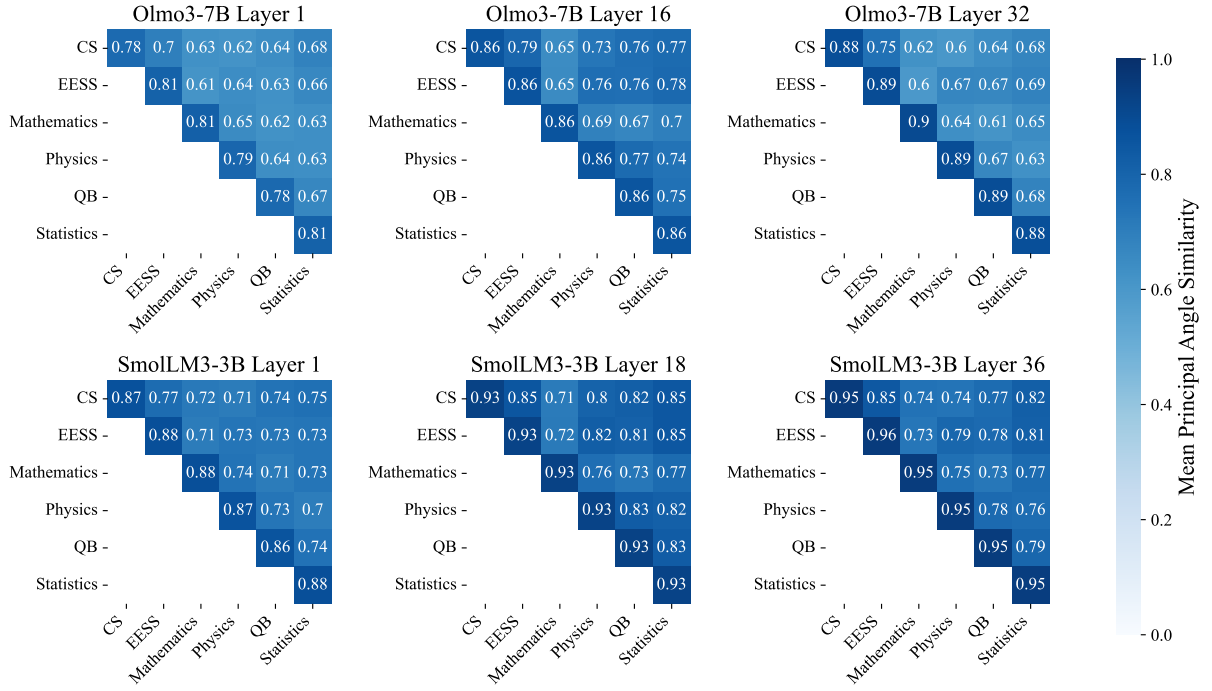


Figure 4: MPAS of domain subspaces across different domains in Olmo3-7B and SmolLM3-3B. We show results for the first layer, the middle layer, and the final layer. Diagonal elements of the heatmaps correspond to comparisons using different texts from the same domain, while upper-triangular elements correspond to comparisons using texts from different domains.

ules, incorporating the important neuron channels in FFN layers leads to a slight increase in peak GPU memory, while the test perplexity decreases significantly, demonstrating the necessity of training these parameters.

F.4 Important Parameters Fine-Tuning

We conduct the same domain fine-tuning experiments on Olmo3-7B and SmolLM3-3B. Tables 13 and 14 show the fine-tuning results for Olmo3-7B and SmolLM3-3B, respectively. Fine-tuning the important parameter vectors achieves test perplexity close to full fine-tuning, while also exhibiting better generalization. Table 15 shows the proportion of trainable parameters and peak GPU memory when fine-tuning different ratios of the FFN parameter vectors in the two models. Fine-tuning the important parameter vectors substantially reduces computational and memory usage, enabling broader application scenarios.

Furthermore, Table 16 shows the SAS computation time across each domain on a single A800 GPU using 10,000 domain-specific samples. The computed SAS is a floating-point array of size $l \times d_{in}$. Taking LLaMA3.1-8B as an example, its storage overhead is merely 1.8 MB.

F.5 Additional Domain-Specific Fine-Tuning

We further validate the effectiveness of our method on medical (MedQA) and legal (CaseHold) datasets using LLaMA3.1-8B. For MedQA, we conduct experiments with the original training, validation, and test splits provided by the dataset. For CaseHold, we randomly sample 14,000 instances and split them into training, validation, and test sets following a 5:1:1 ratio. The experimental results are shown in Table 17. As can be seen, the SAS-based fine-tuning strategy can still achieve performance comparable to full fine-tuning on these complex datasets. The improvement in test accuracy after fine-tuning is very close to that of full fine-tuning, and the test perplexity on MedQA is even superior to that of full fine-tuning, which further validates the effectiveness of our method.

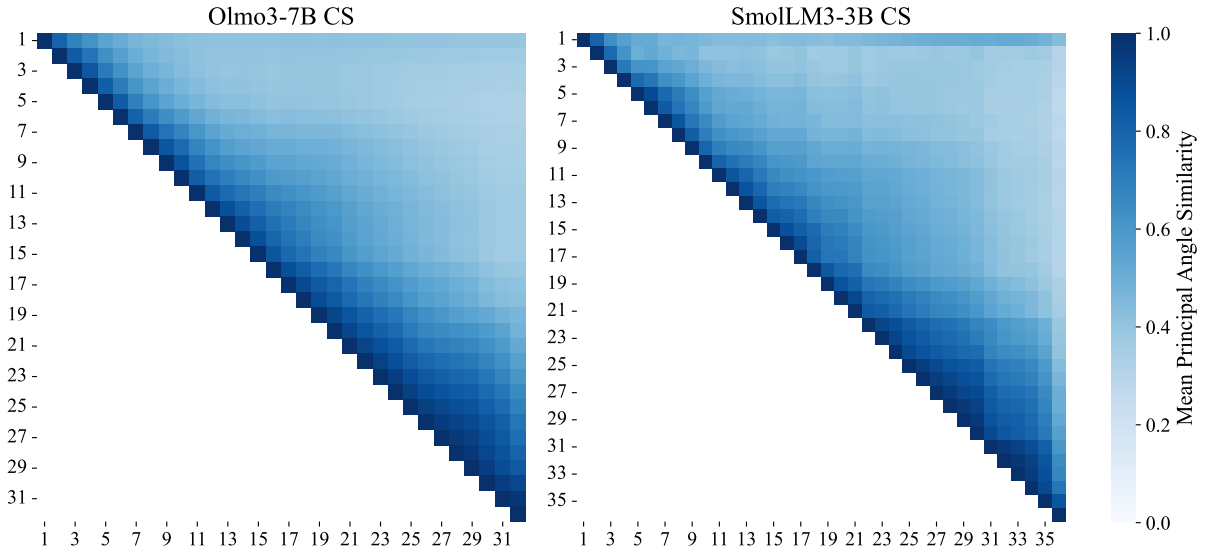


Figure 5: MPAS between domain subspaces across different layers in Olmo3-7B and SmolLM3-3B when using text from CS domain.

Method	Model	Probability (%)						MRR ($\times 100$)					
		CS	EESS	Math.	Phys.	QB	Stat.	CS	EESS	Math.	Phys.	QB	Stat.
Origin		57.72	28.64	62.04	42.40	38.93	51.35	93.58	65.11	96.08	83.53	81.35	91.70
SUC	Olmo	41.23	21.79	44.21	29.68	36.84	39.75	86.92	54.34	85.89	63.40	69.10	82.45
LPI		35.14	16.39	<u>37.59</u>	21.25	18.28	27.26	79.12	45.35	78.21	52.00	50.25	67.51
CULNIG		23.34	7.74	40.86	<u>15.98</u>	12.48	<u>18.71</u>	<u>63.26</u>	<u>36.02</u>	<u>77.20</u>	<u>48.10</u>	<u>42.21</u>	<u>50.70</u>
SAS		<u>26.35</u>	<u>10.67</u>	28.16	13.78	<u>14.84</u>	17.55	58.78	32.77	61.07	35.31	39.06	41.89

Table 9: Probability (%) and Mean Reciprocal Rank (MRR $\times 100$) of the correct answer when intervening the top-1 most important value vector per FFN layer across each domain. Results are averaged over 100 samples. **Bold** and underlined values indicate the lowest and second-lowest results, respectively. Lower values indicate more accurate localization of important parameters.

Method	Model	Probability (%)						MRR ($\times 100$)					
		CS	EESS	Math.	Phys.	QB	Stat.	CS	EESS	Math.	Phys.	QB	Stat.
Origin		57.72	28.64	62.04	42.40	38.93	51.35	93.58	65.11	96.08	83.53	81.35	91.70
Without P	Olmo	27.53	<u>15.64</u>	29.03	16.67	<u>15.60</u>	22.30	64.94	39.50	<u>65.57</u>	39.88	<u>40.18</u>	56.27
Random P		<u>27.15</u>	15.65	<u>28.80</u>	<u>16.66</u>	<u>15.60</u>	<u>22.27</u>	<u>64.44</u>	<u>39.48</u>	<u>65.57</u>	<u>39.30</u>	40.21	<u>55.45</u>
SAS		26.35	10.67	28.16	13.78	14.84	17.55	58.78	32.77	61.07	35.31	39.06	41.89

Table 10: Comparison among Without P , Random P , and SAS. **Bold** and underlined values indicate the lowest and second-lowest results, respectively. Lower values indicate more accurate localization of important parameters.

Domain	LLaMA3.1-8B			Olmo3-3B			SmolLM3-3B		
	Origin	FT-FFN -10%	FT-FFN -Frozen	Origin	FT-FFN -10%	FT-FFN -Frozen	Origin	FT-FFN -10%	FT-FFN -Frozen
CS	11.1129	10.3695	10.4688	14.6616	10.6664	10.7972	13.5908	12.6638	12.7520
EESS	9.8979	9.1760	9.2776	13.7218	9.8779	10.0262	12.7429	11.6176	11.7765
Math.	6.2630	5.8498	5.9130	8.6894	6.6676	6.7526	9.0514	8.2721	8.3554
Phys.	10.5883	10.0411	10.1202	14.8454	11.5352	11.7229	14.0616	13.2836	13.3517
QB	10.8058	10.2825	10.3534	13.7057	11.2088	11.3755	12.9020	12.2584	12.3464
Stat.	10.1675	9.5281	9.6351	13.3349	10.5020	10.6650	13.1057	12.0444	12.1390

Table 11: Test perplexity across different domains under control experiments. We compare fine-tuning 10% of the important parameter vectors in FFN layers and fine-tuning with all FFN layers frozen. "Origin" denotes the performance of the pre-trained model.

FT Strategy	Trainable Params Ratio (%)			Peak GPU Memory (GB)		
	LLaMA3.1-8B	Olmo3-3B	SmolLM3-3B	LLaMA3.1-8B	Olmo3-3B	SmolLM3-3B
FT-FFN-10%	29.80	40.68	20.82	37.83	42.08	18.59
FT-FFN-Frozen	36.82	46.62	28.73	41.59	44.98	19.65

Table 12: The proportion of trainable parameters and peak GPU memory under different fine-tuning strategies.

Domain	Origin	FT	Domain Important Params FT					Random Params FT-10%
			2%	4%	6%	8%	10%	
CS	14.6616	10.5458	10.7127	10.6924	10.6838	10.6722	10.6664	10.7462
		(-0.2113)	(-0.0709)	(-0.0797)	(-0.0820)	(-0.0839)	(-0.0893)	(-0.0656)
EESS	13.7218	9.7210	9.9307	9.9212	9.9009	9.8765	9.8779	9.9688
		(-0.3771)	(-0.2466)	(-0.2551)	(-0.2510)	(-0.2486)	(-0.2562)	(-0.2561)
Math.	8.6894	6.5953	6.6954	6.6862	6.6776	6.6721	6.6676	6.7324
		(-0.2629)	(-0.1589)	(-0.1661)	(-0.1706)	(-0.1760)	(-0.1779)	(-0.1687)
Phys.	14.8454	11.4218	11.6028	11.5673	11.5545	11.5512	11.5352	11.6546
		(-0.2817)	(-0.0968)	(-0.0999)	(-0.1020)	(-0.1076)	(-0.1073)	(-0.1016)
QB	13.7057	11.0967	11.2911	11.2587	11.2352	11.2205	11.2088	11.3132
		(-0.2399)	(-0.1100)	(-0.0990)	(-0.0961)	(-0.0888)	(-0.0948)	(-0.0867)
Stat.	13.3349	10.3716	10.5586	10.5402	10.5341	10.5203	10.5020	10.6063
		(-0.3011)	(-0.1561)	(-0.1667)	(-0.1754)	(-0.1773)	(-0.1765)	(-0.1626)

Table 13: Fine-tuning results of Olmo3-7B across each domain. We compare full fine-tuning (FT), domain important parameters fine-tuning at different ratios (2%–10%), and random parameters fine-tuning (10%). For each entry, the top value represents the **Test Perplexity**, and the bottom value (in green) represents the **Test/Train Gap**. "Origin" denotes the performance of the pre-trained model.

Domain	Origin	FT	Domain Important Params FT					Random Params FT-10%
			2%	4%	6%	8%	10%	
CS	13.5908	12.5153	12.6990	12.6849	12.6779	12.6708	12.6638	12.6920
		(-0.4019)	(-0.1112)	(-0.1259)	(-0.1330)	(-0.1378)	(-0.1469)	(-0.1323)
EESS	12.7429	11.3362	11.7009	11.6812	11.6583	11.6388	11.6176	11.6812
		(-0.3868)	(-0.1171)	(-0.1318)	(-0.1447)	(-0.1441)	(-0.1440)	(-0.1399)
Math.	9.0514	8.1839	8.2998	8.2940	8.2859	8.2801	8.2721	8.2998
		(-0.3438)	(-0.1102)	(-0.1191)	(-0.1273)	(-0.1328)	(-0.1417)	(-0.1296)
Phys.	14.0616	13.1897	13.3063	13.2987	13.2949	13.2912	13.2836	13.3063
		(-0.4658)	(-0.1185)	(-0.1287)	(-0.1352)	(-0.1471)	(-0.1509)	(-0.1422)
QB	12.9020	12.1806	12.2974	12.2909	12.2779	12.2681	12.2584	12.2974
		(-0.3668)	(-0.1061)	(-0.1136)	(-0.1145)	(-0.1174)	(-0.1194)	(-0.1220)
Stat.	13.1057	11.8294	12.0658	12.0480	12.0444	12.0444	12.0444	12.0480
		(-0.3826)	(-0.0876)	(-0.1018)	(-0.1151)	(-0.1337)	(-0.1516)	(-0.1070)

Table 14: Fine-tuning results of SmolLM3-3B across each domain. We compare full fine-tuning (FT), domain important parameters fine-tuning at different ratios (2%–10%), and random parameters fine-tuning (10%). For each entry, the top value represents the **Test Perplexity**, and the bottom value (in green) represents the **Test/Train Gap**. "Origin" denotes the performance of the pre-trained model.

FT FFN Ratio (%)	Trainable Params Ratio (%)		Peak GPU Memory (GB)	
	Olmo3-7B	SmolLM3-3B	Olmo3-7B	SmolLM3-3B
2	41.87	22.40	42.96	18.85
4	43.06	23.99	43.48	19.08
6	44.25	25.57	44.00	19.28
8	45.43	27.15	44.49	19.49
10	46.62	28.73	44.98	19.65
100	100.00	100.00	69.30	31.74

Table 15: The proportion of trainable parameters and peak GPU memory when fine-tuning different ratios of FFN parameter vectors in Olmo3-7B and SmolLM3-3B.

Domain	LLaMA	Olmo	SmolLM
CS	361.24	381.81	166.00
EESS	341.30	374.41	162.65
Math.	439.49	484.05	209.09
Phys.	366.82	400.74	174.57
QB	371.86	404.64	177.26
Stat.	333.18	363.01	158.59

Table 16: SAS computation time of models across each domain (in seconds).

Method	MedQA		CaseHold	
	Acc.	PPL	Acc.	PPL
Origin	58.05	4.73	52.80	8.93
FT	64.57	4.66	80.80	8.66
FT-FFN-10%	62.37	4.63	79.70	8.76

Table 17: Fine-tuning results of LLaMA-3.1-8B on MedQA and CaseHold. We compare fine-tuning 10% of the important parameter vectors in FFN layers against full fine-tuning, where test accuracy (%) and perplexity are adopted as comparison metrics.