LAYER-INFORMED FINE-TUNING VIA THREE-STAGE FUNCTIONAL SEGMENTATION OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In recent years, the performance of large language models (LLMs) on reasoning tasks has been remarkable, even surpassing human capabilities on various benchmarks. However, there remains a lack of clear understanding in the academic community regarding how the structure and internal parameters of LLMs progressively solve complex reasoning problems. In this study, we investigate the inference process of LLMs on cross-linguistic materials and propose the hypothesis that LLM layers exhibit a structured division of labor across conceptualization, reasoning, and textualization. Based on this hypothesis, we introduce a bottleneck identification mechanism using sensitivity analysis to pinpoint the most critical functional stage for a specific task. Leveraging this insight, we propose a novel approach, Layer-Informed Fine-Tuning (LIFT), which achieves efficient and effective fine-tuning by selectively updating only these functionally critical layers. We then conduct extensive experiments to show that the LIFT method not only accelerates the training process but also significantly improves model performance.

1 INTRODUCTION

In recent years, large language models (LLMs) (Meta, 2024; Bai et al., 2023; Achiam et al., 2023; Abdin et al., 2024; Guo et al., 2025) have demonstrated outstanding performance in a wide range of tasks and made significant progress in handling complex problems such as reasoning (Wei et al., 2022), mathematics (OpenAI, 2024; LlamaWebsite, 2024), and programming (Guo et al., 2024). LLMs, typically built on deep transformer architectures (Vaswani et al., 2017), iteratively refine representations to capture increasingly complex linguistic and logical patterns. Research on how LLMs internally comprehend problems, perform reasoning, and ultimately generate correct answers has become a highly significant topic.

Many studies investigate the mechanistic interpretability of LLMs using methods such as the logit lens (Nostalgebraist, 2020), structural probing (Hewitt & Manning, 2019), and cosine-similarity analyses between hidden states (Timkey & van Schijndel, 2021). For example, Wendler et al. (2024) found that when generating non-English text, the logit-lens-decoded tokens initially resemble English before transitioning into the target language in the last few layers of the LLM. Another study (Li et al., 2025) focuses on the safety of LLMs and finds that certain consecutive intermediate layers may have the ability to detect harmful questions and decide whether to refuse to answer them. In addition, Jin et al. (2025) finds that complex problems may require more layers for comprehension and reasoning, whereas simple problems can be processed and resolved within just a few layers.

Inspired by these studies, we investigate the functional division of different layers within LLMs and propose the Three-Stage Functional Segmentation hypothesis. We posit that LLM layers functionally divide into three contiguous stages: (1) conceptualization layers, where initial layers encode linguistic inputs into abstract, task-relevant representations; (2) reasoning layers, where middle layers perform logical inference to derive latent answers; and (3) textualization layers, where final layers reconstruct these abstract concepts into natural language outputs. In addition, driven by the observed redundancy (especially in layers closer to the output) in LLM reasoning, we further split the reasoning stage into active and idle layers. Active reasoning layers carry out the main inference work, whereas idle reasoning layers add little or no new information. Following Jin et al. (2025), we also hypothesize that the proportion of idle reasoning layers scales with task complexity, shrinking for harder tasks and growing for easier ones.

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

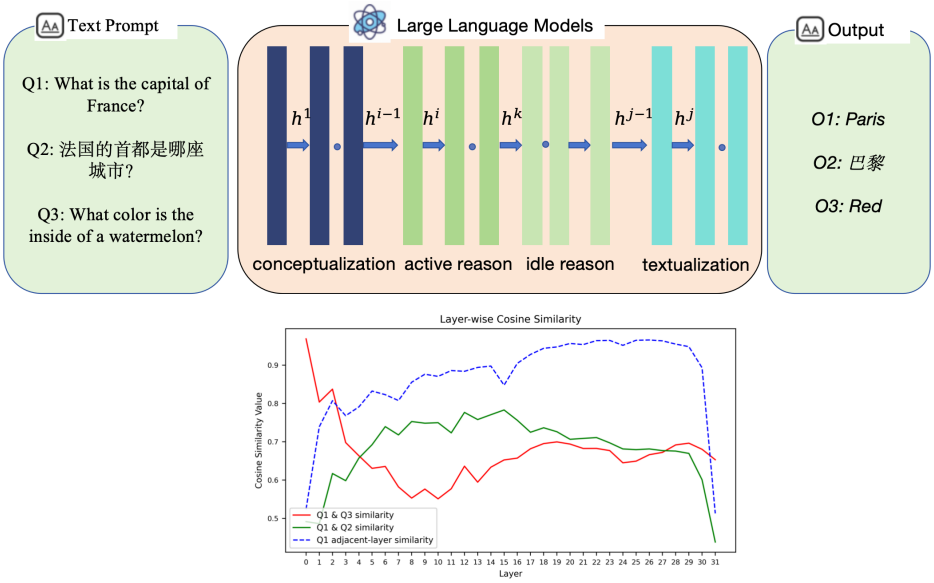


Figure 1: The upper part of the figure illustrates how an LLM processes a simple question under the Three-Stage Functional Segmentation hypothesis. The lower part presents a toy experiment validating this hypothesis by comparing hidden state similarities across conceptualization, reasoning, and textualization layers. We have also added a curve showing the cosine similarity between adjacent layers, with higher values indicating greater redundancy in those layers. In the figure, Q2 and O2 are the Chinese expressions of Q1 and O1, respectively.

Figure 1 illustrates the hypothesis using the query "What is the capital of France?": the model encodes the intent (conceptualization layers), derives the answer "Paris" (reasoning layers), and decodes it into natural language (textualization layers). To validate this, the lower part of Figure 1 tracks hidden state similarity. For semantically equivalent cross-lingual queries (Q1, Q2), similarity rises during conceptualization, peaks in the reasoning stage, and drops during textualization as outputs diverge into different languages. Conversely, for distinct queries in the same language (Q1, Q3), similarity declines in conceptualization, remains low during reasoning, but rises in textualization as representations align to the target language. Additionally, a peak in adjacent-layer similarity signals the presence of redundant idle reasoning layers.

Motivated by the Three-Stage Functional Segmentation hypothesis, we posit that efficient model adaptation does not require updating all parameters, but rather targeting the specific functional component that limits task performance. Drawing an analogy to **Liebig's Law of the Minimum** (von Liebig & Playfair, 1841)—which states that growth is dictated not by total resources but by the scarcest resource—we hypothesize that a model's performance on a specific task is governed by its functional bottleneck. To operationalize this, we propose a systematic framework comprising two key steps: (1) *Location*: automatically delineating the boundaries of conceptualization, reasoning (partitioned into active and idle), and textualization layers; and (2) *Diagnosis*: pinpointing the specific stage acting as the bottleneck via a quantization sensitivity analysis. Building on these diagnostics, we introduce **Layer-Informed Fine-Tuning (LIFT)**, a strategy that maximizes efficiency by selectively fine-tuning only the identified critical layers.

Our contributions can be summarized in three key aspects:

- We propose the hypothesis that LLMs can be functionally divided into conceptualization layers, reasoning layers, and textualization layers. Moreover, for each task, we further refine the reasoning layers by partitioning them into active and idle subsets.
- We develop a comprehensive diagnostic framework to operationalize this hypothesis. This includes a method for delineating the boundaries of these functional stages and a quantization-based sensitivity analysis to empirically pinpoint the functional bottleneck of the model for a task.

- We introduce Layer-Informed Fine-Tuning (LIFT), a parameter-efficient strategy that selectively targets the identified critical layers. Our experiments demonstrate that LIFT is both effective and efficient for fine-tuning LLMs.

2 Related Works

Internal Structure of LLMs Research on the mechanistic interpretability of LLMs often utilizes techniques such as the logit lens (Nostalgebraist, 2020), probing (Hewitt & Manning, 2019), and cosine similarity (Timkey & van Schijndel, 2021) to explore their internal structures. Building upon fundamental probing techniques, a rich line of research has progressively unveiled the layer-wise functional specialization within language models. Early investigations into masked language models demonstrated that internal representations evolve from capturing surface features to syntactic and semantic properties in a pipeline-like manner (Jawahar et al., 2019; Tenney et al., 2019). As models scaled to large autoregressive architectures, this localized specialization was mapped to specific structural components, identifying feed-forward networks as key-value memories (Geva et al., 2021) and pinpointing precise layers responsible for storing factual associations (Meng et al., 2022) or encoding function vectors (Todd et al., 2023). Recently, the mechanistic interpretability lens has expanded to track complex, multi-step phenomena, tracing where in-context learning emerges (Sia et al., 2024), how layers communicate to compose functions (Merullo et al., 2024; Khandelwal & Pavlick, 2025), and why contextualization errors occur (Lepori et al., 2025). A prominent theme unifying these discoveries is the unique role of intermediate layers. Functionally, middle layers act as a "semantic hub" (Wu et al., 2024) or a "high-dimensional abstraction phase" (Cheng et al., 2025) where diverse inputs are mapped to a shared operational space. This is particularly evident in multilingual factual recall (Lu et al., 2025), where models dynamically map inputs to a latent operational language (often English) for intermediate reasoning before decoding into the target language at the final layers (Wendler et al., 2024; Zhao et al., 2024; Zhong et al., 2024). The distinct functional identity of these mid-depth representations is further corroborated by the localization of causally validated "language-selective units" (AlKhamissi et al., 2025) and their superior performance on downstream transfer tasks relative to final-layer outputs (Skean et al., 2025), with models engaging deeper layers primarily for more complex problems (Jin et al., 2025). These studies provide a valuable "map" of the functional properties contained within different layers.

Concurrently, research on model robustness has found that early and final layers are highly sensitive to structural interventions like deletion or swapping, whereas middle layers are remarkably resilient (Sun et al., 2025; Lad et al., 2024). This resilience is particularly pronounced in the later-middle layers, which have been shown to be more redundant and less impactful on performance when pruned compared to earlier layers (Gromov et al., 2025; Men et al., 2025), further supporting their distinct functional role.

While these studies provide a valuable "map" of the functional properties within LLM layers, our work provides a "recipe" for using them. We shift the focus from a static analysis of what is in each layer to the dynamic transitions between these functional stages. This approach allows us to develop targeted and parameter-efficient fine-tuning strategies that improve performance by focusing updates only on functionally-relevant blocks.

Parameter-Efficient Fine-Tuning. The conventional full-parameter fine-tuning paradigm often incurs high computational costs when fine-tuning LLMs. To address this, parameter-efficient fine-tuning (PEFT) has emerged as a promising alternative. Key PEFT approaches include Adapter Tuning (Houlsby et al., 2019; Pfeiffer et al., 2021; Karimi Mahabadi et al., 2021), Prompt Tuning (Li et al., 2023), and Low-Rank Adaptation (LoRA) (Hu et al., 2022). Adapter Tuning introduces trainable adapter modules between all model layers, Prompt Tuning incorporates tunable prefix tokens into inputs.The original LoRA (Hu et al., 2022) decomposes the weight updates for various parameter matrices into low-rank adaptation matrices. More advanced variants, e.g., AdaLoRA (Zhang et al., 2023), improve upon this by dynamically allocating the parameter budget to more important weight matrices based on their sensitivity during training. These methods only need to fine-tune a small subset of parameters, typically less than 1% of the original, and can significantly reduce computational and memory overhead.

Compared with the above methods, our LIFT focuses on fine-tuning specific layers, which follows a distinct trajectory from the above methods and can be seamlessly integrated with them. For instance, in this paper, we incorporate LoRA to achieve both high performance and accelerated fine-tuning.

3 PRELIMINARY

3.1 LARGE LANGUAGE MODELS

Large Language Models, such as Llama-3 (Meta, 2024), are built on a transformer-based architecture composed of multiple stacked layers. Each layer consists of a multi-head self-attention mechanism that captures contextual dependencies, a feedforward network for further processing, and residual connections with layer normalization to stabilize training. In this framework, hidden states represent the evolving internal representations of tokens, which are progressively enriched layer by layer. These states are updated iteratively by adding residuals, where the residual is computed as a function of the hidden states of all preceding tokens. In this paper, we employ two LLMs from different model families: Llama-3-8B-Instruct (Meta, 2024), and Qwen2-7B-Instruct (Yang et al., 2024). These two LLMs were developed by different companies, each with distinct internal architectures and training methodologies, making them highly representative of diverse model designs.

3.2 DATASETS

To train and evaluate the reasoning capabilities of LLMs, we select three representative logical reasoning datasets: ProofWriter (Tafjord et al., 2020), FOLIO (Han et al., 2022) and the LogicalDeduction dataset from BigBench (bench authors, 2023). Our data processing method follows a similar approach to that proposed in Pan et al. (2023). These datasets ensures a comprehensive assessment of the model's ability to perform various forms of logical reasoning. In addition to the reasoning dataset, we also conducted experiments on GSM8K (Cobbe et al., 2021), a mathematical dataset.

4 THREE-STAGE FUNCTIONAL SEGMENTATION OF LLMS

In this section, we present our hypothesis, the Three-Stage Functional Segmentation of LLMs, and provide rigorous and comprehensive experiments to support the hypothesis.

According to this hypothesis, an LLM can be functionally divided into three contiguous stages:

1. **Conceptualization layers** – located near the input, responsible for transforming natural language into abstract representations.
2. **Reasoning layers** – situated in the middle, where task-relevant logical processing occurs. These can be further divided into **active reasoning layers** and **idle reasoning layers**.
3. **Textualization layers** – positioned near the output, where abstract representations are mapped back into natural language.

The conceptualization layers encode linguistic inputs into abstract, task-relevant representations. This initial stage marks a functional transition from processing surface-level language to forming the conceptual foundation required for logical processing.

The reasoning layers then manipulate these abstract representations to perform logical inference. Operating on task-specific logic rather than surface linguistics, we hypothesize that hidden states in this stage will exhibit high similarity for the same task across different languages, yet low similarity for different tasks within the same language. Furthermore, these layers can be subdivided into active reasoning layers, which execute core inference, and idle reasoning layers, which are redundant.

Finally, the textualization layers translate the inferred abstract concepts back into natural language. This final stage reverses the similarity pattern observed during reasoning: representations for different tasks with the same language converge as they are mapped to a common linguistic form, while those for the same task in different languages diverge into their respective linguistic outputs.

To investigate the functional segmentation of layers within the LLM, we design the following experiments during the inference process. Suppose a large language model has K hidden layers and a task-consistent dataset $\mathcal{D}_{\text{raw}} = \{d_i\}_{i=1}^{N}$. Without loss of generality, assume that this dataset

is an English dataset. We then define its corresponding datasets in other natural languages as $\mathcal{D}_{\text{Non-English}} = \{d'_i\}_{i=1}^{N}$.

The Non-English datasets may include any non-English language supported by the LLM, such as Chinese, French, Spanish, or Hindi. The identical index i in these datasets indicates the same semantic content, differing only in language.

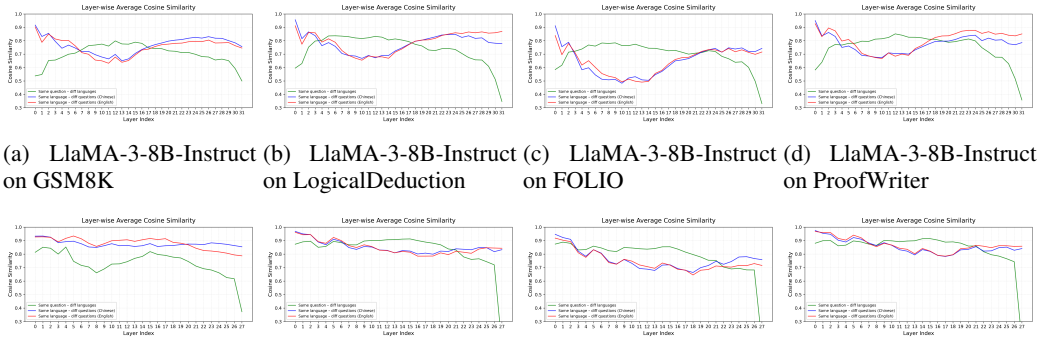
When using these two datasets as the input for inference in the LLM, we obtain the hidden state sets from the last position of each layer in the first autoregressive process, denoted as $S(D_{\text{raw}})$ and $S(D_{\text{Non-English}})$. The specific form of these vector sets is as follows:

$$S(D_{\text{raw}}) = \left\{ [h^1_{(d_i)}, h^2_{(d_i)}, \ldots, h^K_{(d_i)}] \right\}_{i=1}^{N},$$
$$S(D_{\text{Non-English}}) = \left\{ [h^1_{(d'_i)}, h^2_{(d'_i)}, \ldots, h^K_{(d'_i)}] \right\}_{i=1}^{N}$$

where $h^k_{(d_i)}$ denotes the hidden state at the last position in layer k in the first autoregressive process.

4.1 Cosine Similarity between Hidden States in the Same Layer

First, we compute the cosine similarity for the following pairs. These results quantify the similarity between hidden representations, offering insights into how information is processed and transformed within the model.



(a) LlaMA-3-8B-Instruct on GSM8K (b) LlaMA-3-8B-Instruct on LogicalDeduction (c) LlaMA-3-8B-Instruct on FOLIO (d) LlaMA-3-8B-Instruct on ProofWriter

(e) Qwen2-7B-Instruct on GSM8K (f) Qwen2-7B-Instruct on LogicalDeduction (g) Qwen2-7B-Instruct on FOLIO (h) Qwen2-7B-Instruct on ProofWriter

Figure 2: Cosine similarity between Hidden States in the Same Layer.

1. For pairs of the same question in different languages, we randomly select an index i from 1 to N and compute the cosine similarity between different language versions of the same question i, denoted as $g^k_i = \cos(h^k_{(d_i)}, h^k_{(d'_i)})$ for $1 \le k \le K$.
2. For pairs of different questions within the English dataset, we randomly select two indices $i \neq j$ from 1 to N and compute the cosine similarity between different question i and j in English, denoted as $r^k_{i,j} = \cos(h^k_{(d_i)}, h^k_{(d_j)})$ for $1 \le k \le K$.
3. For pairs of different questions within the Non-English dataset, we randomly select two indices $i \neq j$ from 1 to N and compute the cosine similarity between different question i and j, denoted as $b^k_{i,j} = \cos(h^k_{(d'_i)}, h^k_{(d'_j)})$ for $1 \le k \le K$.

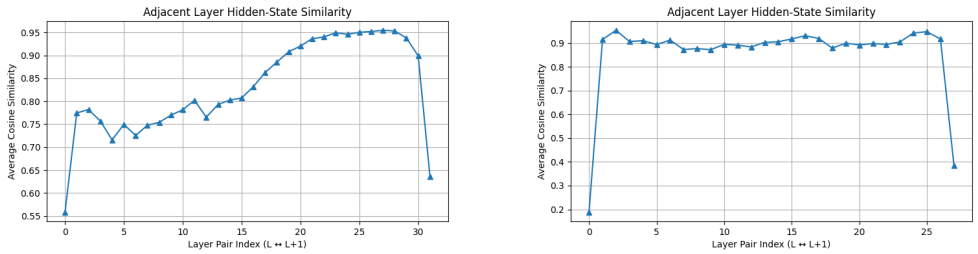
To substantiate our hypothesis, we conduct extensive experiments across multiple models, languages, and datasets. Specifically, we evaluate two widely used models from different model families: Llama-3-8B-Instruct (Meta, 2024) and Qwen2-7B-Instruct (Yang et al., 2024). Furthermore, we assess model behavior across five diverse languages (English, Chinese, French, Spanish, and Hindi) and four reasoning-focused datasets: ProofWriter (Tafjord et al., 2020), FOLIO (Han et al., 2022), LogicalDeduction (bench authors, 2023) and GSM8K (Cobbe et al., 2021). By systematically analyzing model behavior across these varied conditions, we establish robust empirical evidence for

our proposed hypothesis. The results are shown in Figure 2, demonstrating a high degree of consistency across models, languages, and datasets, indicating their robustness and broad applicability. To mitigate the impact of experimental randomness, we randomly repeat the experiments 1000 times and take the average of $g_i^k, r_{i,j}^k, b_{i,j}^k$ for the random i, j's.

As shown in Figure 2, the green line is the cosine similarity between the first kind of pairs (g_i^k's for different layer k), the red line is the cosine similarity between the second kind of pairs ($r_{i,j}^k$'s for different layer k), and the blue line is the cosine similarity between the third kind of pairs ($b_{i,j}^k$'s for different layer k). These results reveal a fundamental divergence between the similarity trends of task-relevant and task-irrelevant pairs. The task-relevant similarity (green line) follows a clear trajectory: it rises to form a distinct plateau throughout the middle layers and subsequently declines in the final layers. Conversely, the task-irrelevant similarity (red and blue lines) exhibits an opposite, U-shaped pattern of decreasing and then increasing. We note that the initial values can be variable, a phenomenon we hypothesize is an artifact of the early hidden states' sensitivity to tokenization differences. Our core claim, however, is based on these starkly opposing trends, which are consistent across languages and strongly substantiate our three-stage hypothesis.

We also investigate the internal representations of these transformer-based models by visualizing hidden states obtained from different language versions of the same dataset, by employing three widely used dimensionality reduction techniques: Principal component analysis (PCA) (Hotelling, 1933), t-distributed stochastic neighboring embedding (t-SNE) (Van der Maaten & Hinton, 2008), and uniform manifold approximation and projection (UMAP) (McInnes et al., 2018). The visualization results are consistent with the similarity results in Figure 2, and we defer them to Appendix.

4.2 Cosine Similarity between Hidden State in Adjacent Layers



(a) LLaMA-3-8B-Instruct on GSM8K

(b) Qwen2-7B-Instruct on LogicalDeduction

Figure 3: Cosine Similarity between Hidden Stated in Adjacent Layers

To achieve a more refined segmentation of the reasoning layers, we adopt the adjacent-layer cosine similarity analysis similar to Gromov et al. (2025) and Men et al. (2025). For each dataset in Subsection 3.2, we compute the cosine similarity between every pair of successive layers. Due to the residual architecture of LLMs, a very high similarity between adjacent layers indicates that the latter layer contributes little new information and is therefore highly redundant. Consistent with the findings of Fan et al. (2025), Gromov et al. (2025), and Men et al. (2025), from Figure 3, we observe that reasoning layers near the textualization stage exhibit similarities exceeding 95% (e.g., layers 23-28 in Figure 3a. We classify these high-similarity layers as idle layers, while the remaining reasoning layers with lower adjacent-layer similarity are designated as active reasoning layers.

4.3 Summary of Experiments

The observed dynamics of representation similarity systematically align with our proposed functions: an initial shift from linguistic to abstract conceptual encoding (conceptualization), followed by a stage of stable, language-agnostic processing (reasoning), and a final reversion to linguistic formatting for output (textualization). This trajectory strongly validates the functional distinctiveness of each stage. Furthermore, our analysis of layer-wise redundancy reveals the existence of idle reasoning layers where information processing plateaus.

However, acknowledging these distinct stages leads to a critical insight regarding model optimization. While inference requires the integration of all stages, we hypothesize that performance is not limited by total capacity, but rather by a specific functional bottleneck—the stage that constitutes the primary constraint for a given task. Since different tasks impose varying cognitive demands—some requiring complex logical manipulation while others hinge on precise linguistic understanding—the functional stage that acts as the limiting factor, or bottleneck, may vary across different model-task pairs. This implies that there is no static most important layer for all scenarios. Instead, a more effective strategy is to empirically identify the specific functional bottleneck for a given task and target it for refinement. This insight lays the groundwork for the methodology proposed in the following section, where we introduce a systematic approach to locate and leverage these critical layers.

5 LIFT: LAYER-INFORMED FINE TUNING

Building upon the hypothesis of functional segmentation proposed in the previous section, we introduce a fine-tuning method aimed at achieving higher efficiency and improved performance: Layer-Informed Fine-Tuning (LIFT). The LIFT framework consists of two sequential steps: Functional Segmentation and Bottleneck Identification. First, we partition the model into distinct functional stages—conceptualization, active reasoning, idle reasoning, and textualization—based on the analysis of cross-lingual and adjacent-layer representation similarities. Second, we employ a perturbation-based analysis to determine which of these stages acts as the functional bottleneck for a given task. Finally, we apply fine-tuning exclusively to these critical layers. We denote LIFT[a, b) to represent the fine-tuning process restricted to layers from a to b (excluding layer b).

5.1 LAYER PARTITION

To implement LIFT, it is essential to first identify the location of functional layer segmentation. We employed a straightforward approach to delineate the layer boundaries. To determine the boundaries between the conceptualization layers and the reasoning layers, we employ the cosine similarity between different language versions of the same question g^k (as described in Section 4), represented by the green line in Figure 2. As previously mentioned in Section 4.1, the green line reflects task-relevant information, which gradually increases within the conceptualization layers and remains relatively stable in the reasoning layers. Therefore, the inflection point during its growth phase can be considered a marker for the functional transition from conceptualization to reasoning. For this discrete case, we apply the discrete second derivative method to identify the inflection point (Weisstein). First, we compute the discrete second derivative of the layer-wise similarity g^k by

$$\Delta^2 g^k \;=\; g^{k+1} \;-\; 2\,g^k \;+\; g^{k-1}.$$

Second, we identify the first index k^* where $\Delta^2 g^k$ changes sign, i.e., where $\Delta^2 g^{k-1}\,\Delta^2 g^k < 0$ as the inflection point marking the transition from conceptualization to reasoning. Next, to partition the

Llama-3-8B-Instruct (Drop)				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	44.04% (±2.03%)	**42.44% (±1.70%)**	44.95% (±1.54%)	44.18% (±1.89%)
FOLIO	52.88% (±0.97%)	**51.33% (±0.83%)**	52.80% (±1.36%)	52.48% (±1.17%)
LogicalDeduction	47.48% (±0.76%)	**43.18% (±1.45%)**	48.86% (±1.43%)	48.58% (±0.89%)
gsm8k	60.30% (±1.73%)	**58.95% (±2.48%)**	61.96% (±1.00%)	61.39% (±1.50%)
Llama-3-8B-Instruct (Recovery)				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	42.56% (±1.76%)	**43.99% (±1.51%)**	42.61% (±1.82%)	42.33% (±1.44%)
FOLIO	52.10% (±1.35%)	**53.46% (±0.97%)**	51.20% (±0.68%)	51.24% (±0.93%)
LogicalDeduction	43.78% (±1.40%)	**47.55% (±0.77%)**	43.26% (±1.41%)	43.50% (±1.50%)
gsm8k	**63.43% (±2.47%)**	60.94% (±1.53%)	62.16% (±2.57%)	63.30% (±1.36%)

Table 1: Performance of Fine-Tuning (Drop and Recovery) over Various Layer Ranges of Llama-3-8B-Instruct (Averaged over 8 Seeds).

reasoning block into active layers and idle layers, we utilize the cosine similarity between hidden states in adjacent layers described in Section 4.2. A higher cosine similarity between adjacent layers indicates greater redundancy. Therefore, we set a threshold value of β and classify consecutive layers exceeding this threshold as idle reasoning layers. We select $\beta = 0.94$ based on empirical

observations across our benchmark tasks: We consistently observed that the adjacent layer similarity in the idle stage saturates at a peak level of approximately 0.95, and thus chose 0.94 as a robust boundary since this value is slightly lower than the saturation peak and ensures that we capture the high-redundancy plateau without encroaching on the active reasoning layers. Consequently, if such a high-similarity region exists, the boundary between the idle reasoning stage and the textualization stage is naturally defined by the point where the similarity drops back below β. All layers succeeding this idle block are then categorized as textualization layers. Conversely, if no layers surpass the threshold β—as we observed in certain datasets for Qwen2-7B-Instruct—we conclude that the model does not exhibit an idle reasoning phase for that specific task. In such cases, the active reasoning block transitions directly into the textualization layers. Based on these methods, we can obtain the specific functional layer segmentation for different tasks across various models. Due to space limitations, the detailed segmentation results are provided in the Appendix.

5.2 Identifying Functionally Critical Layers

The inference process of an LLM can be viewed as a sequential dependency chain: input is first conceptualized, then reasoned over, and finally textualized. Analogous to the "Liebig's Law of the Minimum," (von Liebig & Playfair, 1841) the overall performance of the model is often constrained by its least effective functional component—or conversely, performance is maximized when the most "load-bearing" component is optimized. Therefore, after establishing the boundaries between conceptualization, reasoning, and textualization, the next imperative is to identify and target the specific functional block that acts as the primary determinant of task success.

To empirically isolate these critical layers, we employ a dual-faceted sensitivity analysis using weight-only quantization (specifically AWQ (Lin et al., 2024)). We hypothesize that the layers most vital to a task's functionality will exhibit the highest sensitivity to precision loss and the greatest capacity for performance restoration. We design two complementary diagnostic tests:

1. Degradation Analysis (Drop): We selectively quantize a specific functional block to low precision (INT4) while maintaining the remainder of the model in high precision (BF16). The magnitude of the performance decline indicates the model's reliance on that specific block; a larger drop suggests a critical bottleneck.
2. Recovery Analysis (Recovery): This acts as the inverse of the Degradation Analysis. Here, we maintain a specific functional block in high precision (BF16) while quantizing the rest of the model to low precision (INT4). The extent to which performance is recovered relative to a fully quantized model highlights the functional importance of that block.

	Llama-3-8B-Instruct				Qwen2-7B-Instruct			
ProofWriter	Base model	FullFT	AdaLoRA	LIFT[2,20)	Base model	FullFT	AdaLoRA	LIFT[3,26)
Accuracy	45.67%	56.12% (±2.02%)	47.83% (±0.49%)	**57.67% (±0.36%)**	48.00%	53.17% (±3.10%)	48.25% (±1.68%)	**53.46% (±2.36%)**
Training Time(s)	-	1383	1923	**1215**	-	1288	1839	**1180**
FOLIO	Base model	FullFT	AdaLoRA	LIFT[2,20)	Base model	FullFT	AdaLoRA	LIFT[3,26)
Accuracy	55.39%	52.08% (±1.67%)	57.11% (±2.21%)	**58.21% (±2.35%)**	34.31%	**44.98% (±0.74%)**	18.01% (±1.57%)	44.73% (±1.98%)
Training Time(s)	-	332	516	**290**	-	310	496	**284**
LogicalDeduction	Base model	FullFT	AdaLoRA	LIFT[2,22)	Base model	FullFT	AdaLoRA	LIFT[3,24)
Accuracy	46.33%	53.25% (±1.89%)	49.42% (±2.27%)	**53.58% (±2.47%)**	50.33%	53.83% (±1.23%)	51.42% (±1.17%)	**57.08% (±3.29%)**
Training Time(s)	-	516	737	**460**	-	477	680	**428**
GSM8K	Base model	FullFT	AdaLoRA	LIFT[0,2)	Base model	FullFT	AdaLoRA	LIFT[3,26)
Accuracy	49.96%	67.76% (±0.57%)	72.35% (±1.07%)	**73.96% (±0.98%)**	**78.85%**	74.53% (±0.34%)	77.50% (±0.41%)	75.78% (±0.24%)
Training Time(s)	-	3347	4611	**2667**	-	3500	4832	**3208**

Table 2: Performance and Training Time Comparison of Fine-Tuning Methods over 4 Seeds

As shown in Table 1, our experimental results generally show a strong consistency between these two metrics. For each task, the functional block causing the greatest performance degradation when quantized is typically the same block that yields the highest performance recovery when preserved.

In rare instances where the two metrics diverge—likely due to the stochastic nature of the generation process—we select the target layers based on a composite importance score. We define the optimal functional block as the one that maximizes the differential between the accuracy in the Recovery

setting and the Degradation setting ($Acc_{\text{recovery}} - Acc_{\text{drop}}$). This approach ensures we select the layers that offer the highest robustness and contribution to the task.

Applying this methodology to Llama-3-8B-Instruct (Table 1), we observe distinct functional dependencies. For the ProofWriter, FOLIO, and LogicalDeduction datasets, the active reasoning layers are identified as the most critical. This aligns with the intuition that, for this specific model architecture, these tasks necessitate intensive manipulation of abstract logic. Conversely, for GSM8K, the conceptualization layers exhibit the highest sensitivity and recovery potential, suggesting that the initial translation of the problem statement into an abstract representation is the dominant bottleneck for Llama-3-8B-Instruct on mathematical tasks.

Crucially, however, we do not posit these findings as universal laws for these datasets. As indicated by our broader analysis (see Appendix), the location of the functional bottleneck is intrinsic to the model-task pairing. Different LLMs may allocate computational effort differently for the same task—one model may struggle with conceptualization while another is limited by reasoning capacity. This heterogeneity underscores the necessity of our diagnostic approach, as it dynamically identifies the specific limiting layers for each unique model-task combination rather than relying on static, dataset-level assumptions.

Due to space constraints, results for Qwen2-7B-Instruct are detailed in the Appendix.

5.3 Finetuning Setting

We employ the LoRA framework to accelerate fine-tuning. For our LIFT method, we apply LoRA exclusively to the identified bottleneck layers, whereas the FullFT baseline applies LoRA across all transformer layers. To rigorously evaluate parameter efficiency, we compare against AdaLoRA (Zhang et al., 2023), a dynamic-rank baseline. We strictly align the total trainable parameter budgets between the two methods by dynamically constraining AdaLoRA's target rank based on the specific number of layers LIFT updates for a given task. Detailed budget alignment configurations are provided in Appendix I.

5.4 Experimental Results

Table 2 presents the performance and training time of the original base models, full Transformer layer fine-tuning (FullFT), the dynamic-rank baseline (AdaLoRA), and our proposed LIFT method. The results demonstrate the superiority of LIFT in both effectiveness and computational efficiency. We observe that the LIFT approach demonstrates a noticeable improvement in performance compared to FullFT, while achieving an average reduction of 11.4% in time cost. This efficiency gain stems from LIFT updating drastically fewer trainable parameters (e.g., only 10.5M vs. 167.8M for Llama-3 on GSM8K), which substantially reduces the memory footprint. Furthermore, when compared to AdaLoRA under a strictly aligned parameter budget, LIFT demonstrates superior stability and effectiveness. While AdaLoRA dynamically allocates parameters across all layers, it suffers from massive computational overhead due to its iterative singular value decomposition (SVD) and pruning mechanisms, resulting in the longest training times across all evaluated settings. In contrast, LIFT not only trains significantly faster but also achieves higher accuracy on the majority of logical reasoning tasks, notably preventing catastrophic performance drops such as AdaLoRA's collapse to 18.01% on the Qwen2 FOLIO benchmark. These experimental results indicate that by selectively fine-tuning only those layers most relevant to a given task's functionality, LIFT can simultaneously enhance model accuracy, reduce training time, and lower memory requirements. Additional ablations that fine-tune alternative functional regions are reported in Appendix J and further support the benefit of targeting the LIFT-selected layers.

6 Conclusion

In this work, we proposed a three-stage functional view of LLMs, consisting of conceptualization, reasoning, and textualization stages, with reasoning further divided into active and idle layers. Based on this structure, we introduced a quantization-based diagnostic to identify task-specific functional bottlenecks and proposed LIFT to selectively fine-tune the corresponding layers. Experiments across two model families and four reasoning datasets show that LIFT can improve task performance while reducing fine-tuning cost. Our results suggest that task-specific layer structure can provide a useful basis for efficient model adaptation.

AI USE STATEMENT

In this work, we used GPT-4 to translate the original English benchmark datasets into Chinese, French, Spanish, and Hindi for our multilingual analysis. We additionally used generative AI tools to edit the manuscript for language and readability. We take full responsibility for the final content of the paper and all AI-assisted artifacts.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of our research. All models used in this study (Llama-3-8B-Instruct, Qwen2-7B-Instruct) are publicly available, as described in Section 3.1. The datasets are standard benchmarks, detailed in Section 3.2, with further examples provided in Appendix B. Our multilingual datasets and the code used for our cosine similarity analysis are available in the supplementary materials, as noted in Appendix B.3. The methodology for identifying functional layer boundaries is detailed in Section 5.1 and Appendix E. Furthermore, Appendix H provides a comprehensive description of our experimental environment, including hardware specifications, software versions (PyTorch, CUDA), and the complete set of hyperparameters used for LoRA fine-tuning. The supplementary materials also contain the necessary code to replicate our main experiments and generate the figures presented in the paper.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of machine learning, specifically focusing on the mechanistic interpretability of Large Language Models (LLMs) and parameter-efficient fine-tuning.

Our proposed method, LIFT, significantly reduces the computational resources required for model adaptation by targeting functionally critical layers. This improvement in training efficiency contributes to the broader goal of Green AI, helping to lower the energy consumption and carbon footprint associated with LLM deployment. Furthermore, by identifying the functional segmentation of LLMs, our work provides insights into the internal mechanisms of these models. We believe that such improvements in interpretability are essential for developing safer, more transparent, and more trustworthy AI systems.

Regarding the datasets used in our experiments (ProofWriter, FOLIO, LogicalDeduction, and GSM8K), they are established public benchmarks containing logical and mathematical problems. They do not contain any personally identifiable information or sensitive content. We do not foresee any direct negative societal consequences from this specific research.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Badr AlKhamissi, Greta Tuckute, Antoine Bosselut, and Martin Schrimpf. The llm language network: A neuroscientific approach for identifying causally task-relevant units. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10887–10911, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

BIG bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

Emily Cheng, Diego Doimo, Corentin Kervadec, Iuri Macocco, Lei Yu, Alessandro Laio, and Marco Baroni. Emergence of a high-dimensional abstraction phase in language transformers. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=0fD3iIBhlV.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, and Yequan Wang. Not all layers of llms are necessary during inference. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pp. 5083–5091. ijcai.org, 2025. doi: 10.24963/IJCAI.2025/566. URL https://doi.org/10.24963/ijcai.2025/566.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=ngmEcEer8a.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 3651–3657, 2019.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. Exploring concept depth: How large language models acquire knowledge and concept at different layers? In *Proceedings of the 31st International Conference on Computational Linguistics*, pp.

558–573, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.37/.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021.

Apoorv Khandelwal and Ellie Pavlick. How do language models compose functions? *arXiv preprint arXiv:2510.01685*, 2025.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

Michael A Lepori, Michael Curtis Mozer, and Asma Ghandeharioun. Racing thoughts: Explaining contextualization errors in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3020–3036, 2025.

Lei Li, Yongfeng Zhang, and Li Chen. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 1348–1357, 2023.

Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers in aligned large language models: The key to LLM security. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=kUH1yPMAn7.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

LlamaWebsite. Introducing Llama 3.2, 2024. URL https://www.llama.com/. Accessed: 14-Oct-2024.

Meng Lu, Ruochen Zhang, Carsten Eickhoff, and Ellie Pavlick. Paths not taken: Understanding and mending the multilingual factual recall pipeline. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15077–15107, 2025.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pp. 20192–20204. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.1035/.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Talking heads: Understanding inter-layer communication in transformer language models. *Advances in Neural Information Processing Systems*, 37:61372–61418, 2024.

AI Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*, 2024.

Nostalgebraist. Interpreting gpt: The logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. Accessed: 2025-02-12.

OpenAI. Learning to reason with LLMs, 2024. URL https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2024-10-14.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, volume EMNLP 2023 of *Findings of ACL*, pp. 3806–3824. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.248. URL https://doi.org/10.18653/v1/2023.findings-emnlp.248.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pp. 487–503. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EACL-MAIN.39. URL https://doi.org/10.18653/v1/2021.eacl-main.39.

Suzanna Sia, David Mueller, and Kevin Duh. Where does in-context learning happen in large language models? *Advances in Neural Information Processing Systems*, 37:32761–32786, 2024.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=WGXb7UdvTX.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25219–25227, 2025.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings*, 2020. URL https://api.semanticscholar.org/CorpusID:229371222.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://api.semanticscholar.org/CorpusID:237453326.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

Justus Freiherr von Liebig and Lyon Playfair Baron Playfair. *Organic chemistry in its applications to agriculture and physiology*. J. Owen, 1841.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Eric W. Weisstein. Inflection point. https://mathworld.wolfram.com/InflectionPoint.html. A Wolfram Web Resource. Accessed: 2025-09-23.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in english? on the latent language of multilingual transformers. *arXiv preprint arXiv:2402.10588*, 2024.

Zhaofeng Wu, Xinyan Velocity Yu, Dani Yogatama, Jiasen Lu, and Yoon Kim. The semantic hub hypothesis: Language models share semantic representations across languages and modalities. *arXiv preprint arXiv:2411.04986*, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Yiran Zhao, Wenxuan Zhang, Guizhen Chen, Kenji Kawaguchi, and Lidong Bing. How do large language models handle multilingualism? *Advances in Neural Information Processing Systems*, 37:15296–15319, 2024.

Chengzhi Zhong, Fei Cheng, Qianying Liu, Junfeng Jiang, Zhen Wan, Chenhui Chu, Yugo Murawaki, and Sadao Kurohashi. Beyond english-centric llms: What language do multilingual language models think in? *arXiv preprint arXiv:2408.10811*, 2024.

A LLM STRUCTURE

Model	Layers	Model	Layers
Llama-3-8B-Instruct	32	Qwen2-7B-Instruct	28

Table 3: Number of layers in each LLM

B DATASETS

B.1 DATASET DETAILS

- **ProofWriter** (Tafjord et al., 2020) is a widely used dataset for deductive logical reasoning. In our study, we utilize the open-world assumption subset. This dataset is categorized into five classes based on the depth of reasoning required. To ensure a rigorous evaluation, we select the most challenging subset, depth-5.
- **FOLIO** (Han et al., 2022) is a challenging dataset for first-order logic reasoning, featuring expert-written problems that require complex logical deductions. The questions are presented in natural language and are closely aligned with real-world knowledge.
- **LogicalDeduction** is a logical reasoning task from the BigBench benchmark (bench authors, 2023), focusing on constraint satisfaction problems. The task primarily involves deducing the order of a sequence of objects given a minimal set of conditions.
- **GSM8K** (Cobbe et al., 2021): This dataset contains 8,500 grade school-level math problems that cover a wide range of topics, including arithmetic, algebra, and geometry.

B.2 EXAMPLES OF EACH DATASET

B.2.1 PROOFWRITER

```
"instruction": "The cow is blue. The cow is round. The cow likes the
    lion. The cow visits the tiger. The lion is cold. The lion is nice
    . The lion likes the squirrel. The squirrel is round. The squirrel
     sees the lion. The squirrel visits the cow. The tiger likes the
    cow. The tiger likes the squirrel. If something is cold then it
    visits the tiger. If something visits the tiger then it is nice.
    If something sees the tiger and it is young then it is blue. If
    something is nice then it sees the tiger. If something likes the
    squirrel and it likes the cow then it visits the tiger. If
    something is nice and it sees the tiger then it is young. If the
    cow is cold and the cow visits the lion then the lion sees the
    squirrel.Based on the above information, is the following
    statement true, false, or unknown? The tiger is not young.options:
     ['A) True', 'B) False', 'C) Unknown']",

"input": "",

"output": "B"
```

B.2.2 FOLIO

```
"instruction": "All people who regularly drink coffee are dependent on
     caffeine. People either regularly drink coffee or joke about
    being addicted to caffeine. No one who jokes about being addicted
    to caffeine is unaware that caffeine is a drug. Rina is either a
    student and unaware that caffeine is a drug, or neither a student
    nor unaware that caffeine is a drug. If Rina is not a person
    dependent on caffeine and a student, then Rina is either a person
    dependent on caffeine and a student, or neither a person dependent
```

```
      on caffeine nor a student.Based on the above information, is the
      following statement true, false, or uncertain? Rina is a person
      who jokes about being addicted to caffeine or unaware that
      caffeine is a drug.options: ['A) True', 'B) False', 'C) Uncertain
      ']",

  "input": "",

  "output": "A"
```


B.2.3 LOGICALDEDUCTION


```
  "instruction": "The following paragraphs each describe a set of five
      objects arranged in a fixed order. The statements are logically
      consistent within each paragraph.On a branch, there are five birds
      : a quail, an owl, a raven, a falcon, and a robin. The owl is the
      leftmost. The robin is to the left of the raven. The quail is the
      rightmost. The raven is the third from the left.Which of the
      following is true?options: ['A) The quail is the rightmost.', 'B)
      The owl is the rightmost.', 'C) The raven is the rightmost.', 'D)
      The falcon is the rightmost.', 'E) The robin is the rightmost.']",

  "input": "",

  "output": "A"
```


B.2.4 GSM8K

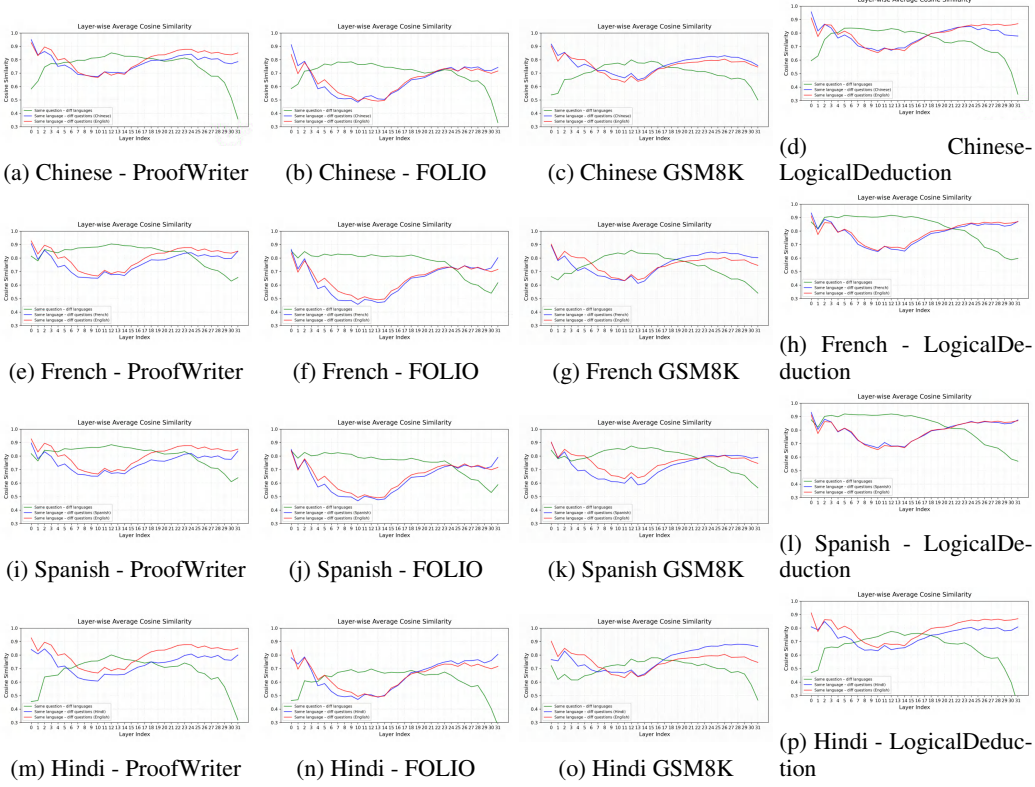

```
  "instruction": "Natalia sold clips to 48 of her friends in April, and
      then she sold half as many clips in May. How many clips did
      Natalia sell altogether in April and May?",
  "input": "",
  "output": "Natalia sold 48/2 = <<48/2=24>>24 clips in May.\nNatalia
      sold 48+24 = <<48+24=72>>72 clips altogether in April and May.\n
      #### 72"
```


B.3 MULTILINGUAL DATASET AND AVAILABILITY

To facilitate our multilingual analysis, the original English datasets were translated using the GPT-4 language model. This process generated versions of the datasets in four additional languages: Chinese, French, Spanish, and Hindi. All translated data used in this study is publicly available for review in the supplementary material, located at the following path: Codes/Cosine Similarity/

C DETAILS OF COSINE SIMILARITIES OF HIDDEN STATES

To make the Cosine Similarity more pronounced, we adjusted the order of the questions in the dataset, presenting the options first followed by the question statement. We present here the complete set of cosine similarities of hidden states from Section 4.



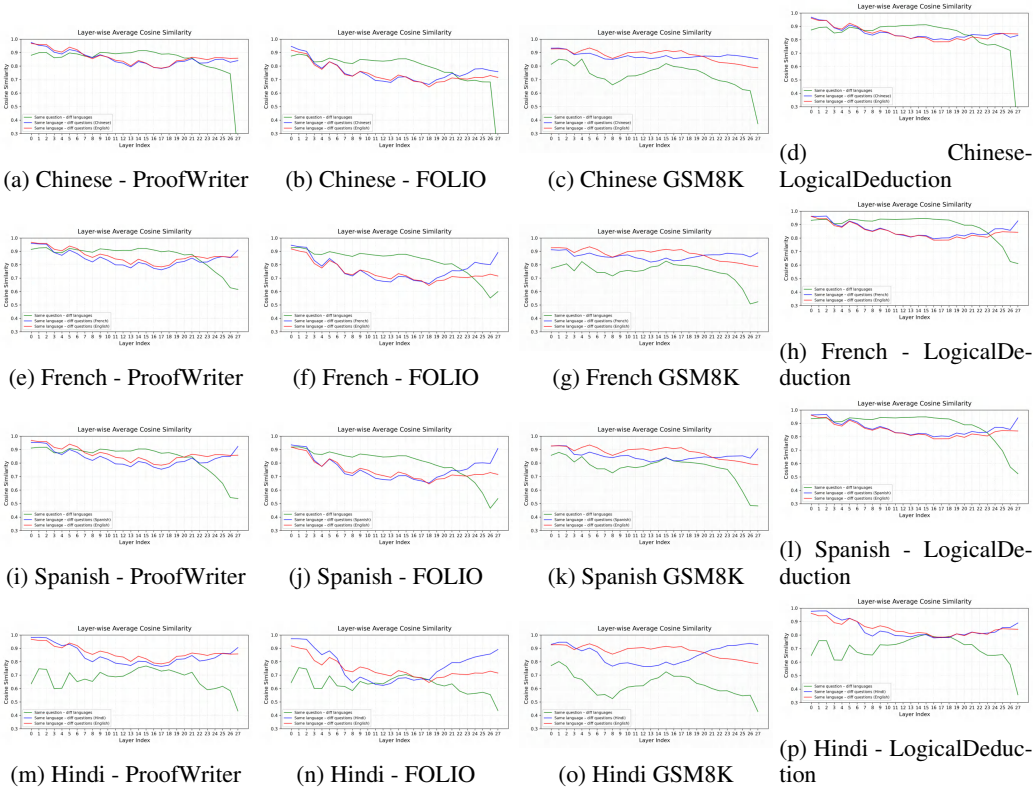
(a) Chinese - ProofWriter (b) Chinese - FOLIO (c) Chinese GSM8K (d) Chinese-LogicalDeduction

(e) French - ProofWriter (f) French - FOLIO (g) French GSM8K (h) French - LogicalDeduction

(i) Spanish - ProofWriter (j) Spanish - FOLIO (k) Spanish GSM8K (l) Spanish - LogicalDeduction

(m) Hindi - ProofWriter (n) Hindi - FOLIO (o) Hindi GSM8K (p) Hindi - LogicalDeduction

Figure 4: Performance of Llama-3-8B-Instruct across languages and datasets.



(a) Chinese - ProofWriter (b) Chinese - FOLIO (c) Chinese GSM8K (d) Chinese-LogicalDeduction

(e) French - ProofWriter (f) French - FOLIO (g) French GSM8K (h) French - LogicalDeduction

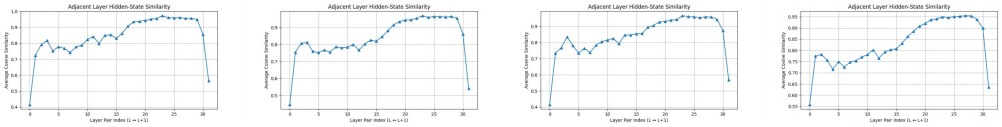
(i) Spanish - ProofWriter (j) Spanish - FOLIO (k) Spanish GSM8K (l) Spanish - LogicalDeduction

(m) Hindi - ProofWriter (n) Hindi - FOLIO (o) Hindi GSM8K (p) Hindi - LogicalDeduction

Figure 5: Performance of Qwen2-7B-Instruct across languages and datasets.

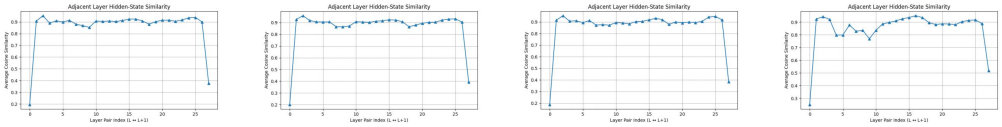
D Details of Cosine Similarity between Hidden Stated in Adjacent Layers

In this section, we present the full details of the cosine similarity between hidden states in adjacent layers. We conduct experiments on LLMs in Subsection 3.1 across four datasets in Subsection 3.2, which directly facilitate the identification of idle reasoning layers.



(a) ProofWriter (b) FOLIO (c) LogicalDeduction (d) GSM8K

Figure 6: Cosine Similarity between Hidden Stated in Adjacent Layers of Llama-3-8B-Instruct



(a) ProofWriter (b) FOLIO (c) LogicalDeduction (d) GSM8K

Figure 7: Cosine Similarity between Hidden Stated in Adjacent Layers of Qwen2-7B-Instruct

E RESULTS OF THE METHOD FOR FUNCTIONAL LAYER SEGMENTATION

In this Section, we provide a detailed explanation of the method used to approximate a reasonable location in Section 5, and introduce the values we ultimately selected for the functional layer segmentation.

Our method identifies the boundary between conceptualization and reasoning layers by finding the first significant inflection point in the cross-lingual task similarity curve (g^k). We locate this by calculating the discrete second derivative, $\Delta^2 g^k = g^{k+1} - 2g^k + g^{k-1}$, and identifying the first layer k^* where the sign of $\Delta^2 g^k$ changes. To determine the boundary of idle reasoning layers, we simply select layers where the adjacent cosine similarity exceeds a predefined threshold β.

For instance, for the Llama-3-8B-Instruct model on the ProofWriter dataset (comparing Chinese and English versions), the layer-wise mean cosine similarities (g^k) are: [0.582, 0.640, 0.745, 0.772, 0.768, 0.780, 0.781, 0.797, 0.795, 0.812, 0.815, 0.824, 0.851, 0.841, 0.826, 0.826, 0.821, 0.811, 0.811, 0.799, 0.788, 0.793, 0.804, 0.813, 0.799, 0.748, 0.715, 0.677, 0.676, 0.629, 0.515, 0.357]

Calculating the discrete second derivative for this sequence, the first few values are:

- $\Delta^2 g^1 = g^2 - 2g^1 + g^0 \approx 0.745 - 2(0.640) + 0.582 = +\mathbf{0.047}$
- $\Delta^2 g^2 = g^3 - 2g^2 + g^1 \approx 0.772 - 2(0.745) + 0.640 = -\mathbf{0.078}$

The first sign change occurs at $k^* = 2$, which we identify as the boundary. We have performed this analysis across all our experimental settings and found this result to be remarkably consistent for this model. For the vast majority of tasks and languages, the first inflection point for Llama-3 is identified at **layer 2**. Based on this robust finding, we select layer 2 as the conceptualization-reasoning boundary for this model. In contrast, the boundary between active and idle reasoning layers is task-dependent; generally, more challenging tasks require a larger number of active reasoning layers. The specific segmentation results are summarized in the table below.

During the experiments, we observed that for certain datasets, the adjacent cosine similarity of Qwen2-7B-Instruct failed to exceed the threshold β. In such cases, we conclude that the task does not contain idle reasoning layers.

Llama-3-8B-Instruct				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	[0,2)	[2,20)	[20,30)	[30,32)
FOLIO	[0,2)	[2,20)	[20,30)	[30,32)
LogicalDeduction	[0,2)	[2,22)	[22,30)	[30,32)
GSM8K	[0,2)	[2,22)	[22,30)	[30,32)

Table 4: Functional Layer Segmentation of Llama-3-8B-Instruct

Qwen2-7B-Instruct				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	[0,3)	[3,26)	-	[26,28)
FOLIO	[0,3)	[3,26)	-	[26,28)
LogicalDeduction	[0,3)	[3,24)	[24,26)	[26,28)
GSM8K	[0,3)	[3,26)	-	[26,28)

Table 5: Functional Layer Segmentation of Qwen2-7B-Instruct

F Appendix: Layer Sensitivity Probing via Simulated Quantization

To identify the layers most susceptible to quantization noise, we developed a sensitivity probing framework based on **Activation-aware Weight Quantization (AWQ)**. This approach allows us to simulate the effects of low-bit precision on specific model components without the need for specialized inference kernels.

F.1 Simulated Quantization Mechanism

Instead of physically packing weights into lower-bit formats (e.g., INT4) which requires specific hardware kernels for inference, we employ a *Simulated Quantization* (or Pseudo-Quantization) strategy. The quantization process proceeds in two steps:

1. **Parameter Search:** We first compute the optimal scaling factors for each linear layer using the standard AWQ algorithm, which minimizes the reconstruction error of the layer's output activations.
2. **Noise Injection:** We then apply a quantize-dequantize (QDQ) operation to the weights. The weights are quantized to the target precision (INT4) and immediately dequantized back to floating-point format (`bfloat16`).

F.2 Experimental Setup and Data Protocols

Our sensitivity analysis involves determining the impact of quantizing specific blocks of layers while keeping others in full precision. The data protocols for calibration and evaluation are designed as follows:

Calibration: To ensure the quantization parameters capture the activation distribution of the target domain, we perform calibration using the **training dataset** corresponding to each specific task. We sample a small subset of sequences from the training data to calculate the AWQ scaling factors, ensuring the quantization is tailored to the relevant data distribution.

Evaluation: To efficiently measure the sensitivity of different layers, we conduct the evaluation directly on the training dataset. Specifically, to accelerate the probing process and enable rapid iteration across multiple layer configurations, we utilize the **first 1,000 samples** of the training set for testing. While standard evaluation typically uses a held-out test set, using the training subset for sensitivity probing provides a sufficient signal to distinguish between robust and sensitive layers, as we are primarily measuring the relative degradation of the model's inherent knowledge representation under quantization noise.

Probing Strategy: We adopt a "Drop" and "Recovery" methodology to isolate layer importance. In the *Drop* mode, a target block of layers is simulated in INT4 while the rest remain in BF16, measuring the performance drop. Conversely, the *Recovery* mode keeps the target block in BF16 while quantizing the rest, measuring the performance recovery.

G Results of Identifying Functionally Critical Layers

G.1 Identification of Target Layers in Qwen2-7B-Instruct

To validate the generalizability of our functional segmentation across different architectures, we performed the "Drop" and "Recovery" ablation studies on Qwen2-7B-Instruct. The results, presented in Table 6, consistently identify the **active reasoning layers** as the functional bottleneck and the optimal target for fine-tuning.

High Sensitivity in Drop Mode: As shown in the first section of Table 6, omitting the active reasoning layers results in the most severe performance degradation across all evaluated benchmarks. For instance, on the ProofWriter dataset, accuracy drops to its lowest point at 43.15%, significantly

underperforming the configurations where Conceptualization (48.45%) or Textualization (46.68%) layers are dropped. This indicates that the active reasoning layers house the most critical parameters for logical inference.

Superior Efficiency in Recovery Mode: Conversely, in the Recovery experiments, fine-tuning exclusively the active reasoning layers yields the highest performance restoration. Notably, on GSM8K, targeting these layers achieves an accuracy of 84.39%, surpassing other layer groups. This confirms that focusing computational resources on this specific block maximizes the marginal gain of fine-tuning.

Conclusion: Based on these observations, we confirm that the active reasoning layers act as the primary performance determinant for Qwen2-7B-Instruct, justifying their selection as the target layers in our proposed method.

Qwen2-7B-Instruct (Drop)				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	48.45% (±1.51%)	**43.15% (±2.05%)**	-	46.68% (±1.83%)
FOLIO	42.18% (±1.45%)	**41.71% (±3.67%)**	-	42.16% (±1.17%)
LogicalDeduction	50.50% (±1.54%)	**49.51% (±0.86%)**	50.31% (±1.56%)	49.56% (±0.78%)
gsm8k	84.16% (±0.78%)	**82.79% (±0.54%)**		84.03% (±0.76%)

Qwen2-7B-Instruct (Recovery)				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	39.86% (±2.06%)	**45.75% (±1.05%)**		41.30% (±1.56%)
FOLIO	40.36% (±2.82%)	**43.46% (±2.17%)**		41.81% (±2.39%)
LogicalDeduction	47.45% (±1.10%)	**49.68% (±0.82%)**	47.80% (±1.14%)	48.84% (±1.75%)
gsm8k	81.91% (±0.86%)	**84.39% (±0.92%)**		82.39% (±1.16%)

Table 6: Performance of Fine-Tuning (Drop and Recovery) over Various Layer Ranges of Qwen2-7B-Instruct (Averaged over 8 Seeds)

H EXPERIMENTAL DETAILS

H.1 EXPERIMENTAL ENVIRONMENT

Our experiments were conducted on a server equipped with **four NVIDIA A40 GPUs**, each with 46 GB of memory. All training and evaluation tasks were performed on a **single A40 GPU**. The server is powered by an **AMD EPYC 9654 96-Core Processor** with **755 GiB of RAM** and **16 TB of storage**.

We used **Ubuntu 20.04.6 LTS** as the operating system, with **Python 3.10.14** as the main programming environment. All deep learning experiments were implemented using **PyTorch 2.5.1+cu124**, leveraging CUDA for GPU acceleration.

H.2 SOFTWARE AND LIBRARIES

- **CUDA Version**: 12.5
- **Driver Version**: 555.58.02
- **Deep Learning Framework**: PyTorch 2.5.1 with CUDA 12.4 support
- **Python Version**: 3.10.14
- **Operating System**: Ubuntu 20.04.6 LTS

H.3 TRAINING CONFIGURATION

We fine-tuned these models using the following hyperparameters:

- **Learning Rate**: 1e-5
- **LoRA Rank**: 64
- **LoRA Alpha**: 128
- **LoRA Dropout**: 0.05
- **Batch Size per Device**: 4
- **Gradient Accumulation Steps**: 8
- **Max Sequence Length**: 512
- **Random Seeds**: 0, 1, 2, 3
- **Warmup Ratio**: 0.03

H.4 EXECUTION STRATEGY

All training and evaluation were executed on a single A40 GPU to maintain consistency. Multi-GPU capabilities were available but not utilized for this experiment. Distributed strategies like FSDP or DeepSpeed were not applied in this setup.

I ADALORA BASELINE CONFIGURATION

To strictly align the global trainable parameter budget between LIFT and AdaLoRA, we dynamically adjusted AdaLoRA's rank settings based on the size of the functional bottleneck identified by LIFT. Specifically, for a given dataset where LIFT targets a functional bottleneck of k layers with a fixed rank of $r = 64$, we constrain AdaLoRA's target average rank across all $K = 32$ layers of Llama-3 to:

$$r_{target} = 64 \times \frac{k}{K} = 2k$$

To afford AdaLoRA sufficient search space for its pruning schedule, the initial rank is set to $r_{init} = r_{target} + 8$. Other pruning hyperparameters are maintained at their standard settings: $t_{init_ratio} = 0.25$, $t_{final_ratio} = 0.8$, and $\Delta T = 100$. This configuration guarantees that AdaLoRA strictly adheres to the exact same parameter budget as LIFT for every evaluated task.

J Results of Ablation Experiments

In this section, we present the results of our ablation experiments. Tables 7, and 8 report the performance of fine-tuning various functional regions of different large language models (LLMs), including the conceptualization layers, active reasoning layers, idle reasoning layers, and textualization layers.

From these experiments, it is evident that our LIFT algorithm significantly outperforms competing approaches in most cases. The fine-tuning strategy, which focuses explicitly on functionally relevant layers, yields substantial improvements in task performance. These results highlight the critical role of selectively targeting specific functional layers within LLMs for effective task-specific optimization.

Llama-3-8B-Instruct				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	37.25% (±1.04%)	57.67% (±0.36%)	48.08% (±0.75%)	46.38% (±0.89%)
FOLIO	53.43% (±1.65%)	58.21% (±2.35%)	56.74% (±1.29%)	55.51% (±2.31%)
LogicalDeduction	45.67% (±0.72%)	53.58% (±2.47%)	46.83% (±1.67%)	45.25% (±0.88%)
GSM8K	73.96% (±0.98%)	67.87% (±0.64%)	65.22% (±0.53%)	56.48% (±0.58%)

Table 7: Performance of Fine-Tuning over Various Layer Ranges of Llama-3-8B-Instruct

Qwen2-7B-Instruct				
Datasets	conceptualization layers	active reasoning layers	idle reasoning layers	textualization layers
ProofWriter	45.50% (±0.93%)	53.46% (±2.36%)	-	47.46% (±0.85%)
FOLIO	27.08% (±4.19%)	44.73% (±1.98%)	-	28.55% (±1.62%)
LogicalDeduction	48.08% (±0.83%)	57.08% (±3.29%)	49.00% (±0.47%)	52.58% (±2.15%)
GSM8K	75.34% (±0.66%)	75.78% (±0.24%)	-	72.29% (±0.72%)

Table 8: Performance of Fine-Tuning over Various Layer Ranges of Qwen2-7B-Instruct

K Limitation

This paper examines only two relatively small-scale open-source models with fewer layers. Although these models have a certain degree of representativeness due to belonging to different model families, further research is required to determine whether the proposed method is applicable to larger-scale models, models with different architectures, and even closed-source models.

L Results of Visualization of Hidden States

In this section, we examine the internal representations of transformer-based models by visualizing the hidden states derived from different language versions of the same dataset. To facilitate this analysis, we employ three widely used dimensionality reduction techniques: Principal Component Analysis (PCA) (Hotelling, 1933), t-distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten & Hinton, 2008), and Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018). For a controlled comparison, we utilize a single dataset with multiple languages, ensuring that the semantic content remains consistent across all versions. The model processes these multilingual inputs and generates corresponding hidden state vectors.

The visualizations produced by PCA, t-SNE, and UMAP (shown in the following figure) provide a robust basis for analyzing the hidden state representations across different languages. Our results reveal a consistent pattern in the evolution of hidden states across the layers of the language model. Specifically, in the input-adjacent and output-adjacent layers, internal representations tend to form clusters based on language. In contrast, in the intermediate layers, hidden states from different languages become increasingly intermixed, exhibiting a clear phenomenon of language blending. This suggests that while early and late layers are more language-specific, intermediate layers abstract away from surface linguistic forms and encode more universal semantic representations.

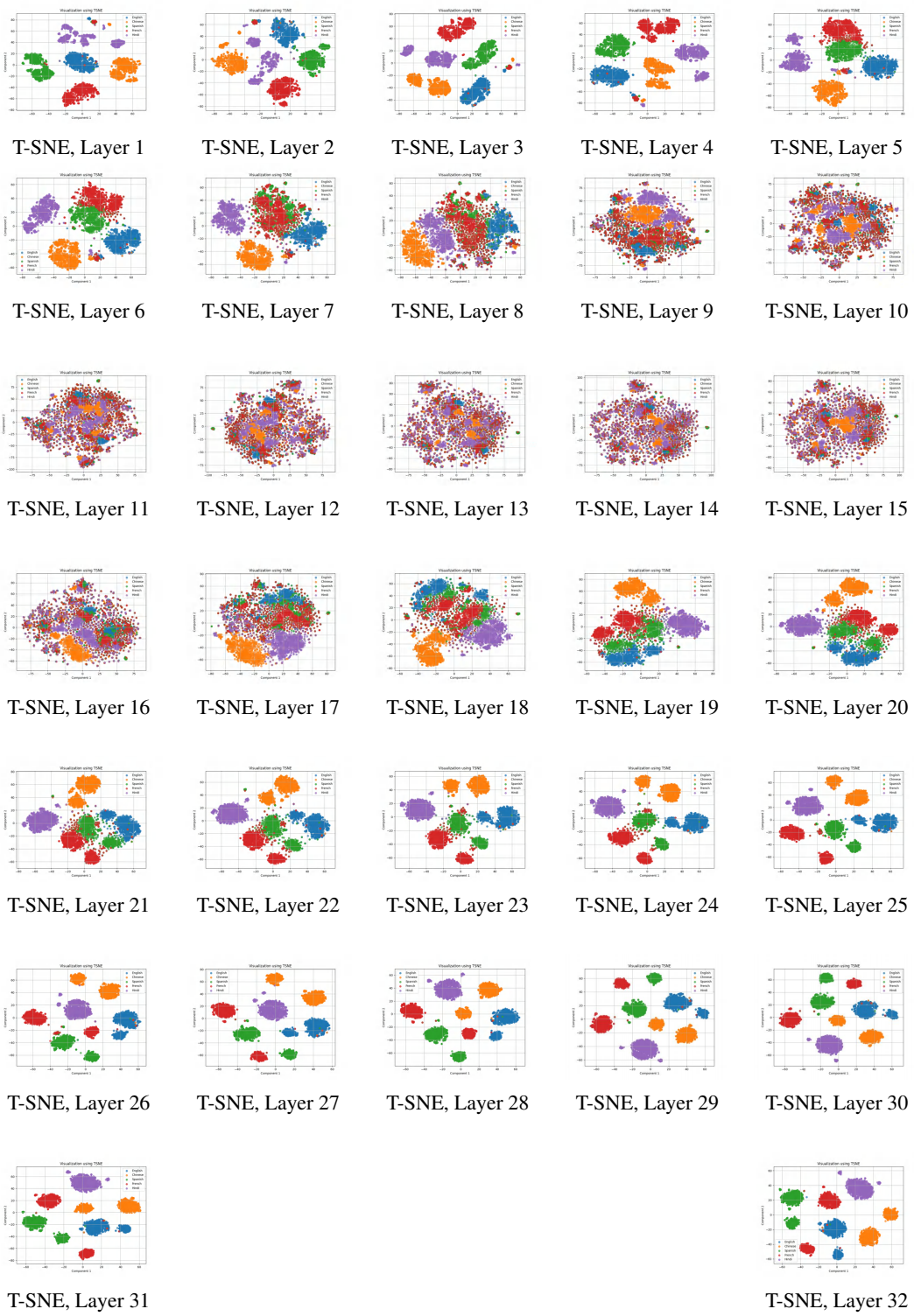


Figure 8: T-SNE visualizations for layers 1-32 of Llama-3-8B-Instruct on the GSM8K dataset.

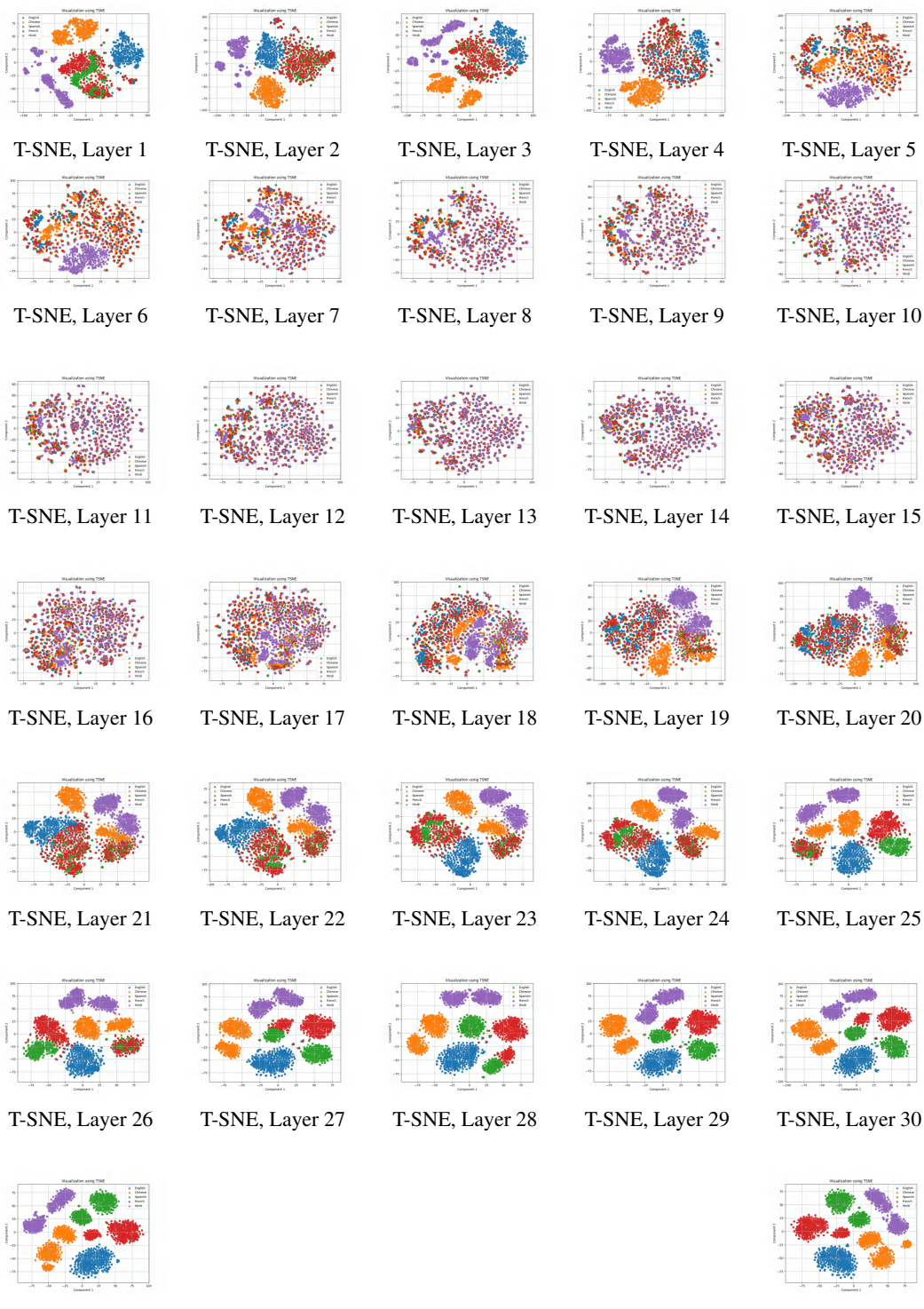
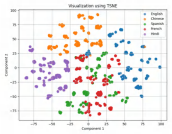
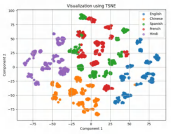


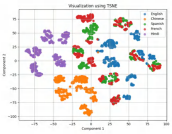
Figure 9: T-SNE visualizations for layers 1-32 of Llama-3-8B-Instruct on the FOLIO dataset.


T-SNE, Layer 1

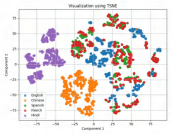

T-SNE, Layer 2

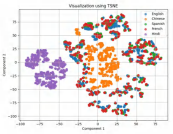

T-SNE, Layer 3


T-SNE, Layer 4


T-SNE, Layer 5

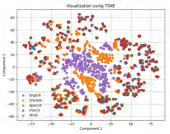

T-SNE, Layer 6


T-SNE, Layer 7


T-SNE, Layer 8


T-SNE, Layer 9


T-SNE, Layer 10

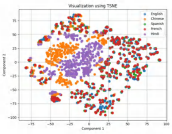

T-SNE, Layer 11

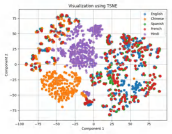

T-SNE, Layer 12


T-SNE, Layer 13


T-SNE, Layer 14


T-SNE, Layer 15

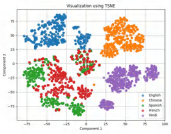

T-SNE, Layer 16


T-SNE, Layer 17

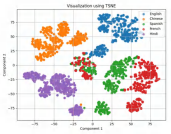

T-SNE, Layer 18


T-SNE, Layer 19

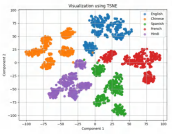

T-SNE, Layer 20

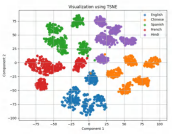

T-SNE, Layer 21

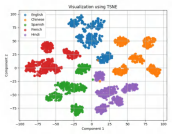

T-SNE, Layer 22

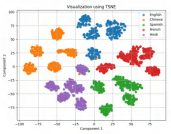

T-SNE, Layer 23


T-SNE, Layer 24


T-SNE, Layer 25

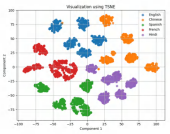

T-SNE, Layer 26


T-SNE, Layer 27


T-SNE, Layer 28


T-SNE, Layer 29


T-SNE, Layer 30


T-SNE, Layer 31

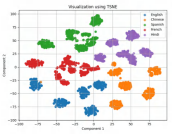
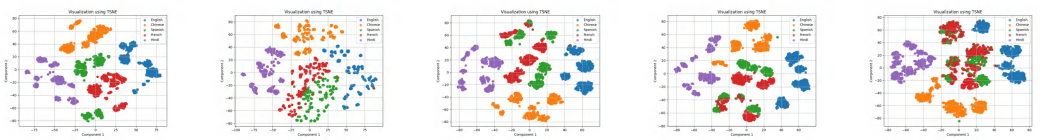
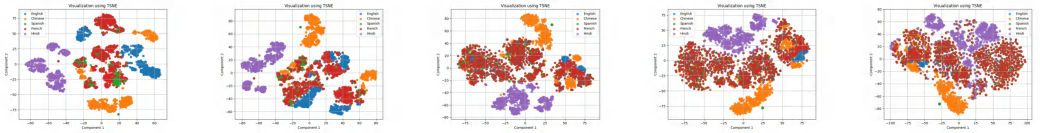
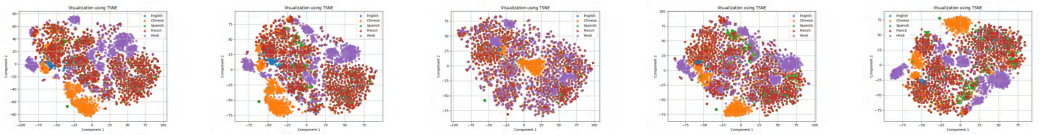
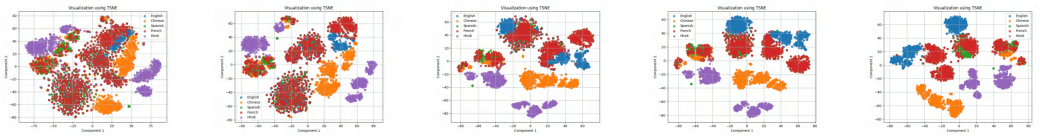

T-SNE, Layer 32

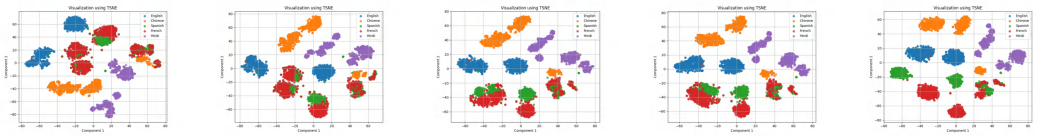
Figure 10: T-SNE visualizations for layers 1-32 of Llama-3-8B-Instruct on the LogicalDeduction dataset.

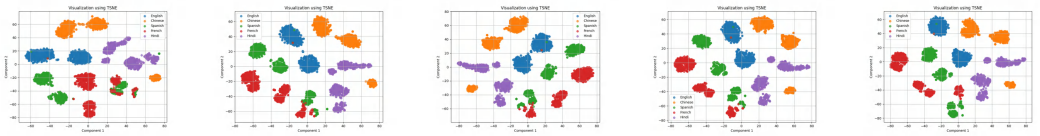

T-SNE, Layer 1 T-SNE, Layer 2 T-SNE, Layer 3 T-SNE, Layer 4 T-SNE, Layer 5


T-SNE, Layer 6 T-SNE, Layer 7 T-SNE, Layer 8 T-SNE, Layer 9 T-SNE, Layer 10


T-SNE, Layer 11 T-SNE, Layer 12 T-SNE, Layer 13 T-SNE, Layer 14 T-SNE, Layer 15


T-SNE, Layer 16 T-SNE, Layer 17 T-SNE, Layer 18 T-SNE, Layer 19 T-SNE, Layer 20


T-SNE, Layer 21 T-SNE, Layer 22 T-SNE, Layer 23 T-SNE, Layer 24 T-SNE, Layer 25


T-SNE, Layer 26 T-SNE, Layer 27 T-SNE, Layer 28 T-SNE, Layer 29 T-SNE, Layer 30


T-SNE, Layer 31 T-SNE, Layer 32

Figure 11: T-SNE visualizations for layers 1-32 of Llama-3-8B-Instruct on the ProofWriter dataset.

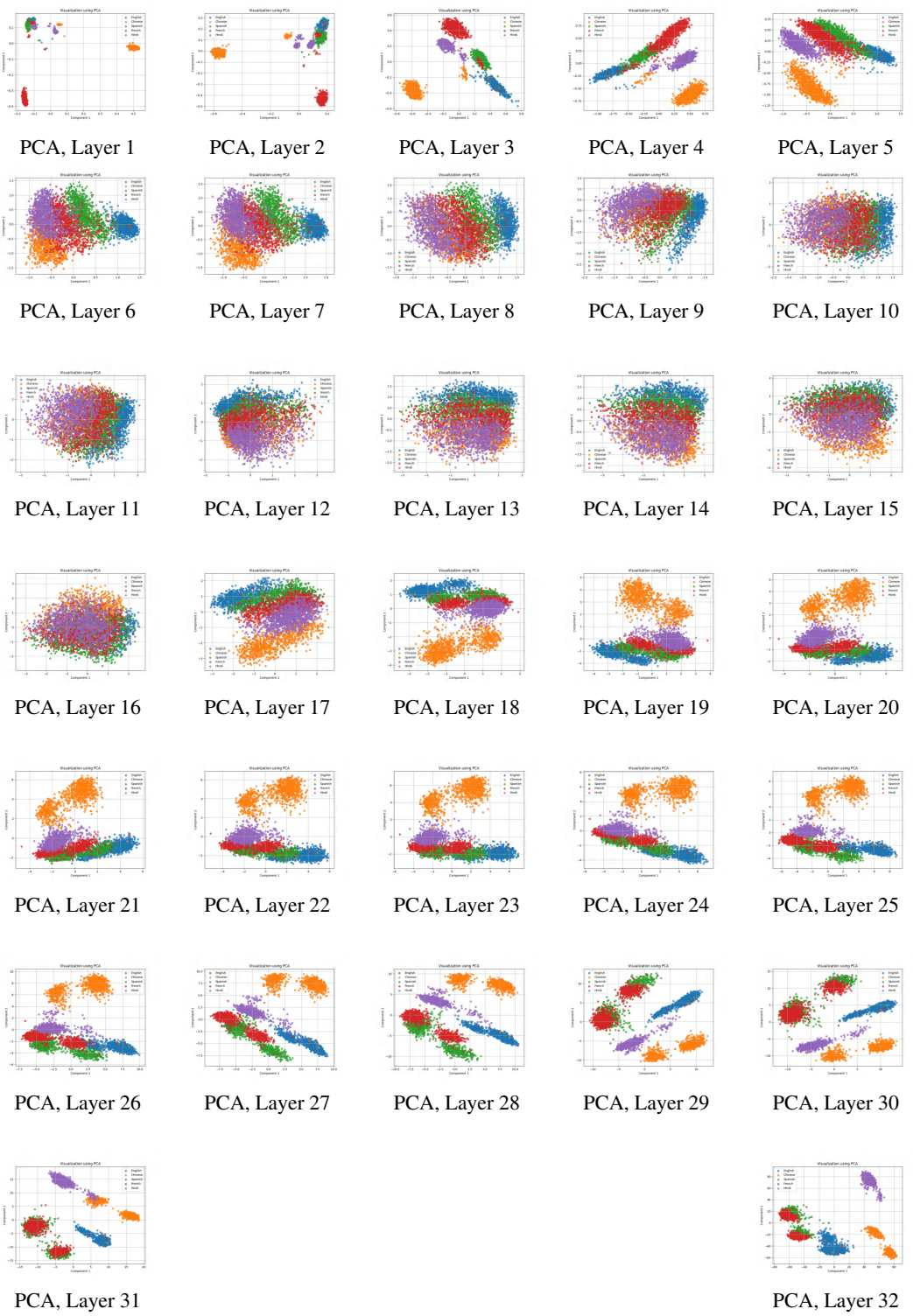
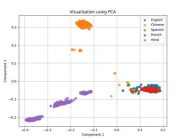
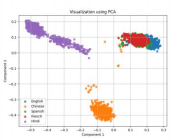
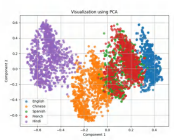
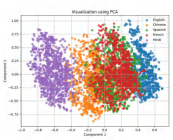


Figure 12: PCA visualizations for layers 1-32 of Llama-3-8B-Instruct on the GSM8K dataset.


PCA, Layer 1

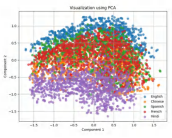

PCA, Layer 2

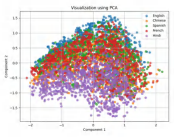

PCA, Layer 3

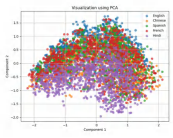

PCA, Layer 4

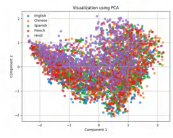

PCA, Layer 5

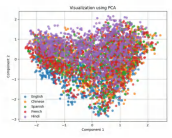

PCA, Layer 6

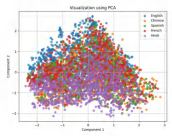

PCA, Layer 7

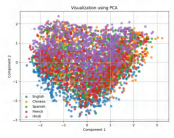

PCA, Layer 8

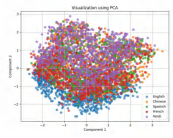

PCA, Layer 9

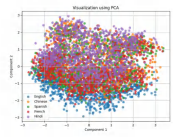

PCA, Layer 10

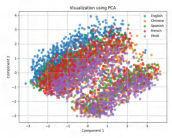

PCA, Layer 11

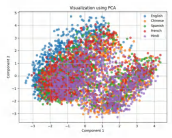

PCA, Layer 12

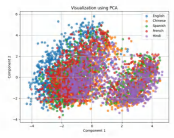

PCA, Layer 13

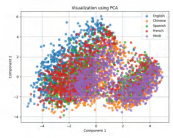

PCA, Layer 14

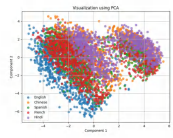

PCA, Layer 15

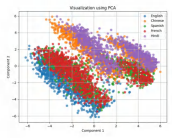

PCA, Layer 16

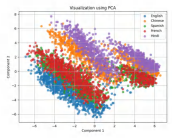

PCA, Layer 17

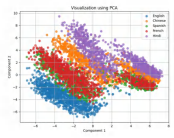

PCA, Layer 18

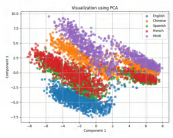

PCA, Layer 19

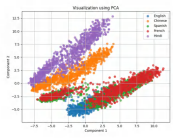

PCA, Layer 20

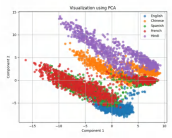

PCA, Layer 21

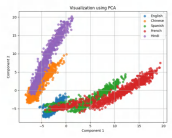

PCA, Layer 22

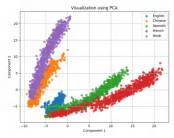

PCA, Layer 23

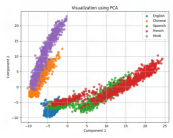

PCA, Layer 24

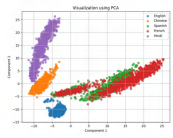

PCA, Layer 25

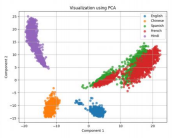

PCA, Layer 26


PCA, Layer 27


PCA, Layer 28


PCA, Layer 29


PCA, Layer 30


PCA, Layer 31

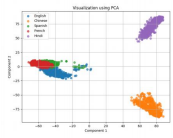

PCA, Layer 32

Figure 13: PCA visualizations for layers 1-32 of Llama-3-8B-Instruct on the FOLIO dataset.

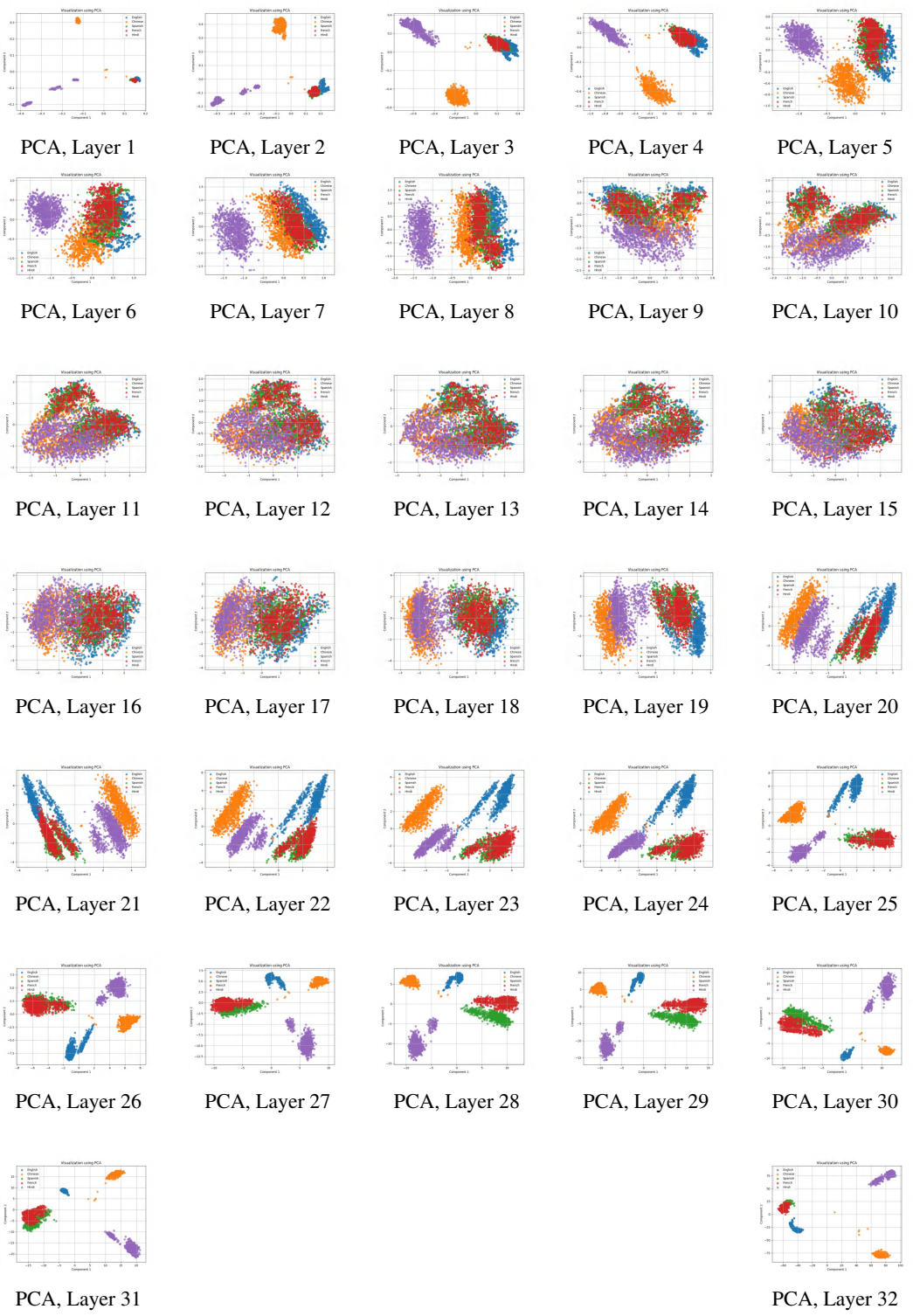
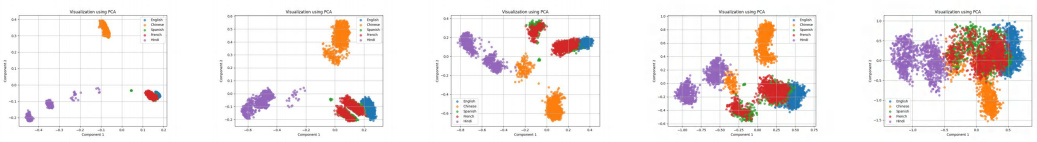
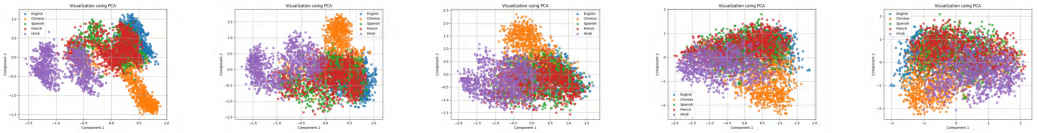
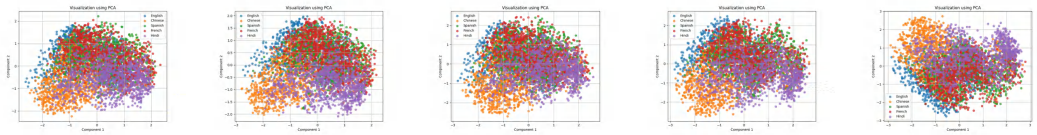
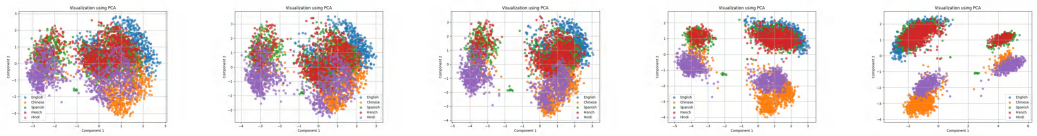


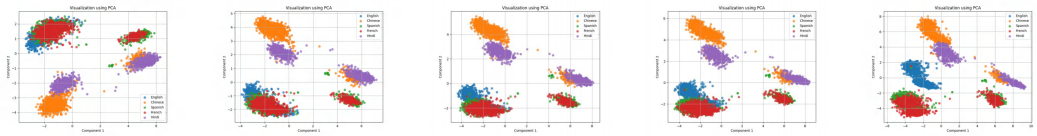
Figure 14: PCA visualizations for layers 1-32 of Llama-3-8B-Instruct on the LogicalDeduction dataset.

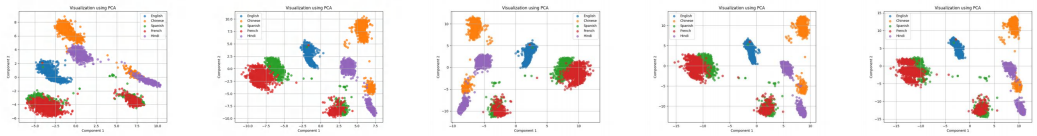

PCA, Layer 1 PCA, Layer 2 PCA, Layer 3 PCA, Layer 4 PCA, Layer 5


PCA, Layer 6 PCA, Layer 7 PCA, Layer 8 PCA, Layer 9 PCA, Layer 10


PCA, Layer 11 PCA, Layer 12 PCA, Layer 13 PCA, Layer 14 PCA, Layer 15

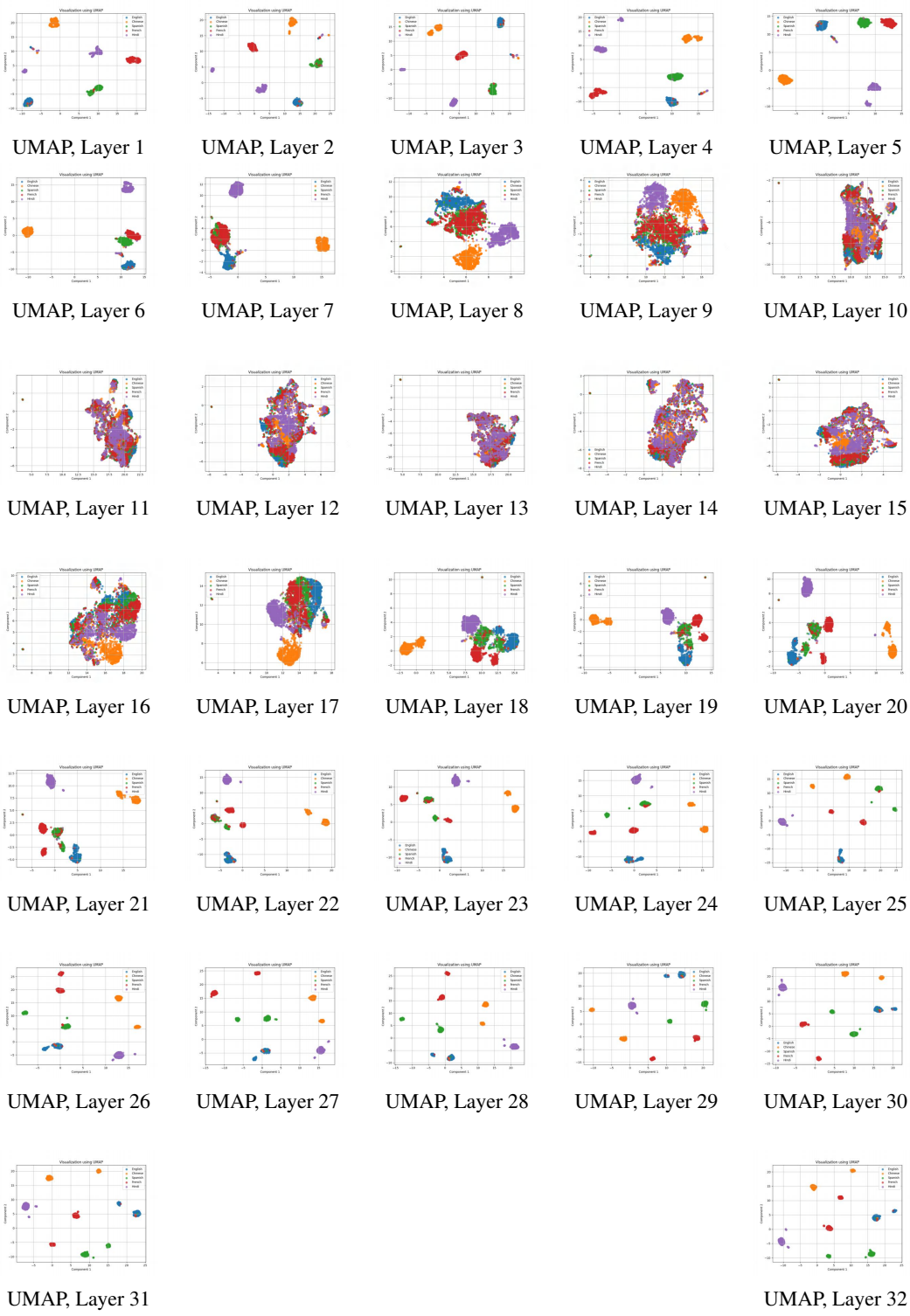

PCA, Layer 16 PCA, Layer 17 PCA, Layer 18 PCA, Layer 19 PCA, Layer 20


PCA, Layer 21 PCA, Layer 22 PCA, Layer 23 PCA, Layer 24 PCA, Layer 25


PCA, Layer 26 PCA, Layer 27 PCA, Layer 28 PCA, Layer 29 PCA, Layer 30


PCA, Layer 31 PCA, Layer 32

Figure 15: PCA visualizations for layers 1-32 of Llama-3-8B-Instruct on the ProofWriter dataset.



UMAP, Layer 1 | UMAP, Layer 2 | UMAP, Layer 3 | UMAP, Layer 4 | UMAP, Layer 5

UMAP, Layer 6 | UMAP, Layer 7 | UMAP, Layer 8 | UMAP, Layer 9 | UMAP, Layer 10

UMAP, Layer 11 | UMAP, Layer 12 | UMAP, Layer 13 | UMAP, Layer 14 | UMAP, Layer 15

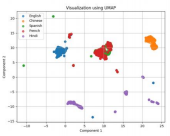
UMAP, Layer 16 | UMAP, Layer 17 | UMAP, Layer 18 | UMAP, Layer 19 | UMAP, Layer 20

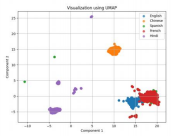
UMAP, Layer 21 | UMAP, Layer 22 | UMAP, Layer 23 | UMAP, Layer 24 | UMAP, Layer 25

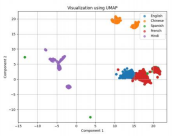
UMAP, Layer 26 | UMAP, Layer 27 | UMAP, Layer 28 | UMAP, Layer 29 | UMAP, Layer 30

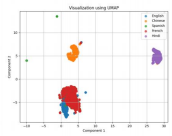
UMAP, Layer 31 | UMAP, Layer 32

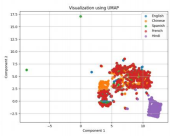
Figure 16: UMAP visualizations for layers 1-32 of Llama-3-8B-Instruct on the GSM8K dataset.

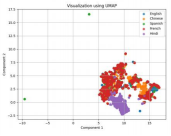

UMAP, Layer 1

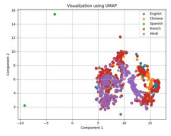

UMAP, Layer 2

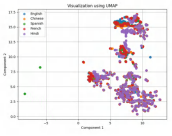

UMAP, Layer 3

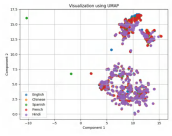

UMAP, Layer 4

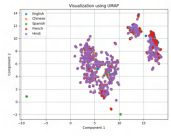

UMAP, Layer 5

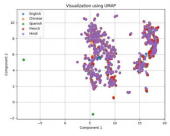

UMAP, Layer 6

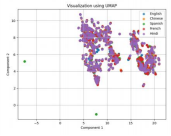

UMAP, Layer 7

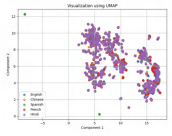

UMAP, Layer 8

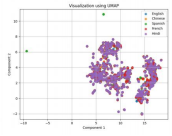

UMAP, Layer 9

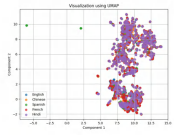

UMAP, Layer 10

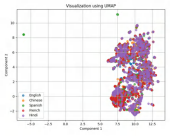

UMAP, Layer 11

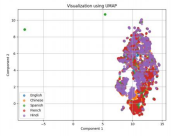

UMAP, Layer 12

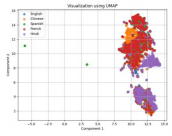

UMAP, Layer 13

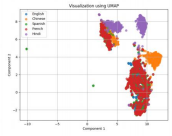

UMAP, Layer 14

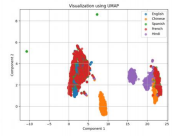

UMAP, Layer 15

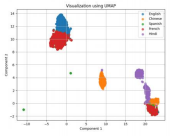

UMAP, Layer 16

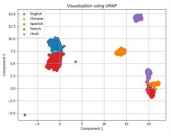

UMAP, Layer 17

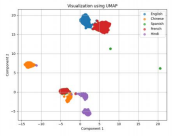

UMAP, Layer 18

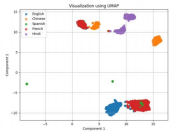

UMAP, Layer 19

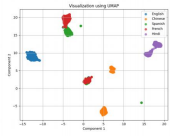

UMAP, Layer 20

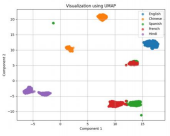

UMAP, Layer 21

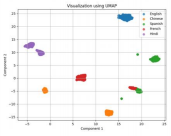

UMAP, Layer 22

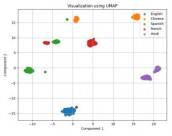

UMAP, Layer 23

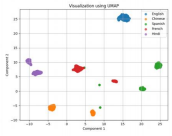

UMAP, Layer 24

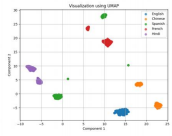

UMAP, Layer 25

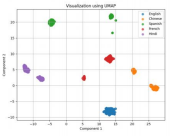

UMAP, Layer 26


UMAP, Layer 27


UMAP, Layer 28


UMAP, Layer 29


UMAP, Layer 30


UMAP, Layer 31

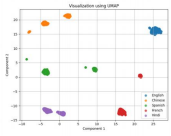
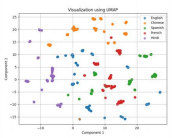
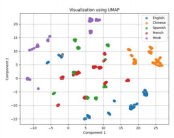
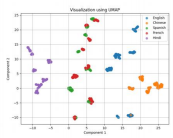

UMAP, Layer 32

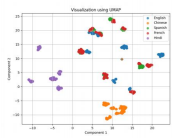
Figure 17: UMAP visualizations for layers 1-32 of Llama-3-8B-Instruct on the FOLIO dataset.

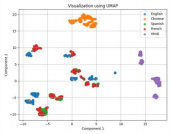

UMAP, Layer 1

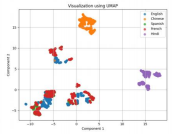

UMAP, Layer 2

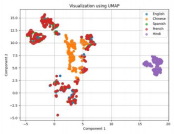

UMAP, Layer 3

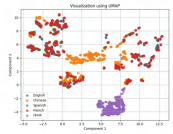

UMAP, Layer 4

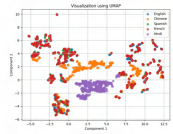

UMAP, Layer 5

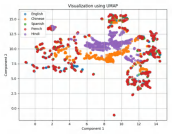

UMAP, Layer 6

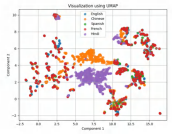

UMAP, Layer 7

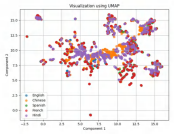

UMAP, Layer 8

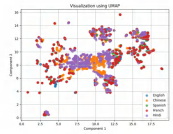

UMAP, Layer 9

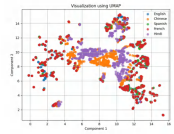

UMAP, Layer 10

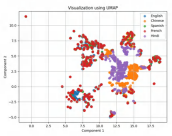

UMAP, Layer 11


UMAP, Layer 12

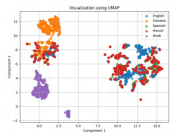

UMAP, Layer 13

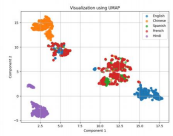

UMAP, Layer 14

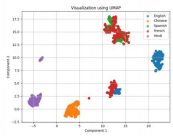

UMAP, Layer 15

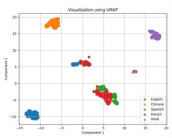

UMAP, Layer 16

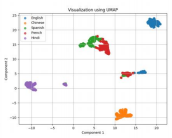

UMAP, Layer 17

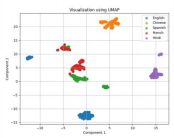

UMAP, Layer 18

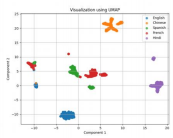

UMAP, Layer 19

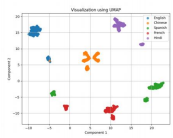

UMAP, Layer 20

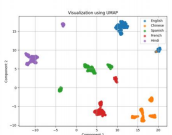

UMAP, Layer 21

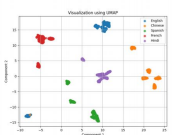

UMAP, Layer 22

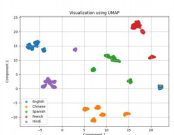

UMAP, Layer 23

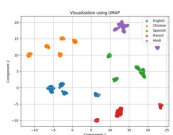

UMAP, Layer 24

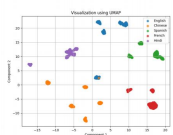

UMAP, Layer 25

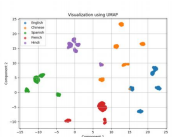

UMAP, Layer 26


UMAP, Layer 27


UMAP, Layer 28


UMAP, Layer 29


UMAP, Layer 30


UMAP, Layer 31

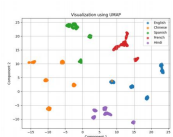
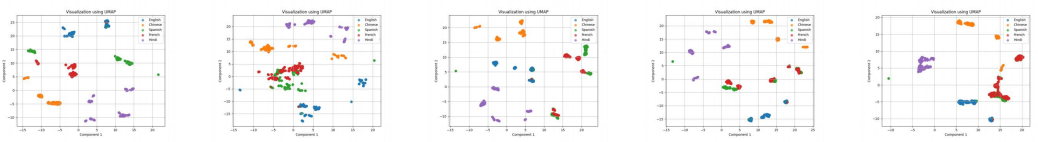
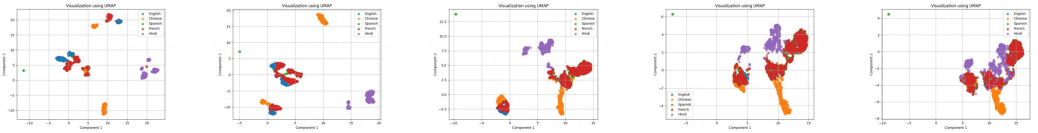
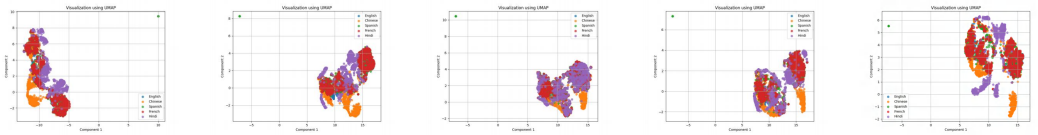
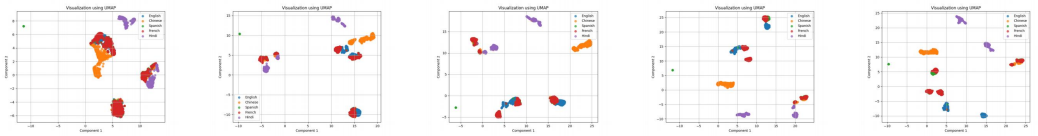

UMAP, Layer 32

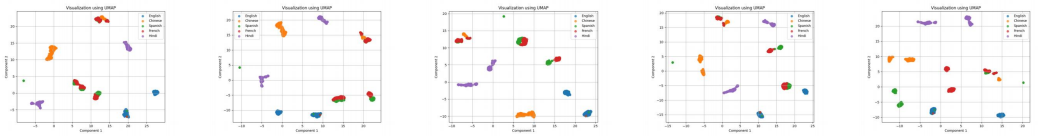
Figure 18: UMAP visualizations for layers 1-32 of Llama-3-8B-Instruct on the LogicalDeduction dataset.

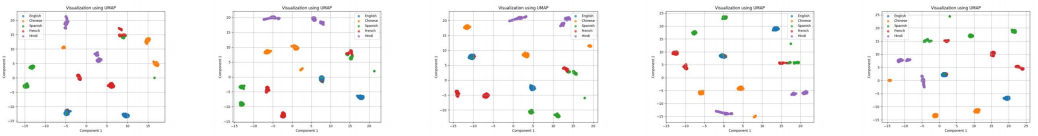

UMAP, Layer 1 UMAP, Layer 2 UMAP, Layer 3 UMAP, Layer 4 UMAP, Layer 5


UMAP, Layer 6 UMAP, Layer 7 UMAP, Layer 8 UMAP, Layer 9 UMAP, Layer 10


UMAP, Layer 11 UMAP, Layer 12 UMAP, Layer 13 UMAP, Layer 14 UMAP, Layer 15


UMAP, Layer 16 UMAP, Layer 17 UMAP, Layer 18 UMAP, Layer 19 UMAP, Layer 20


UMAP, Layer 21 UMAP, Layer 22 UMAP, Layer 23 UMAP, Layer 24 UMAP, Layer 25


UMAP, Layer 26 UMAP, Layer 27 UMAP, Layer 28 UMAP, Layer 29 UMAP, Layer 30


UMAP, Layer 31 UMAP, Layer 32

Figure 19: UMAP visualizations for layers 1-32 of Llama-3-8B-Instruct on the ProofWriter dataset.

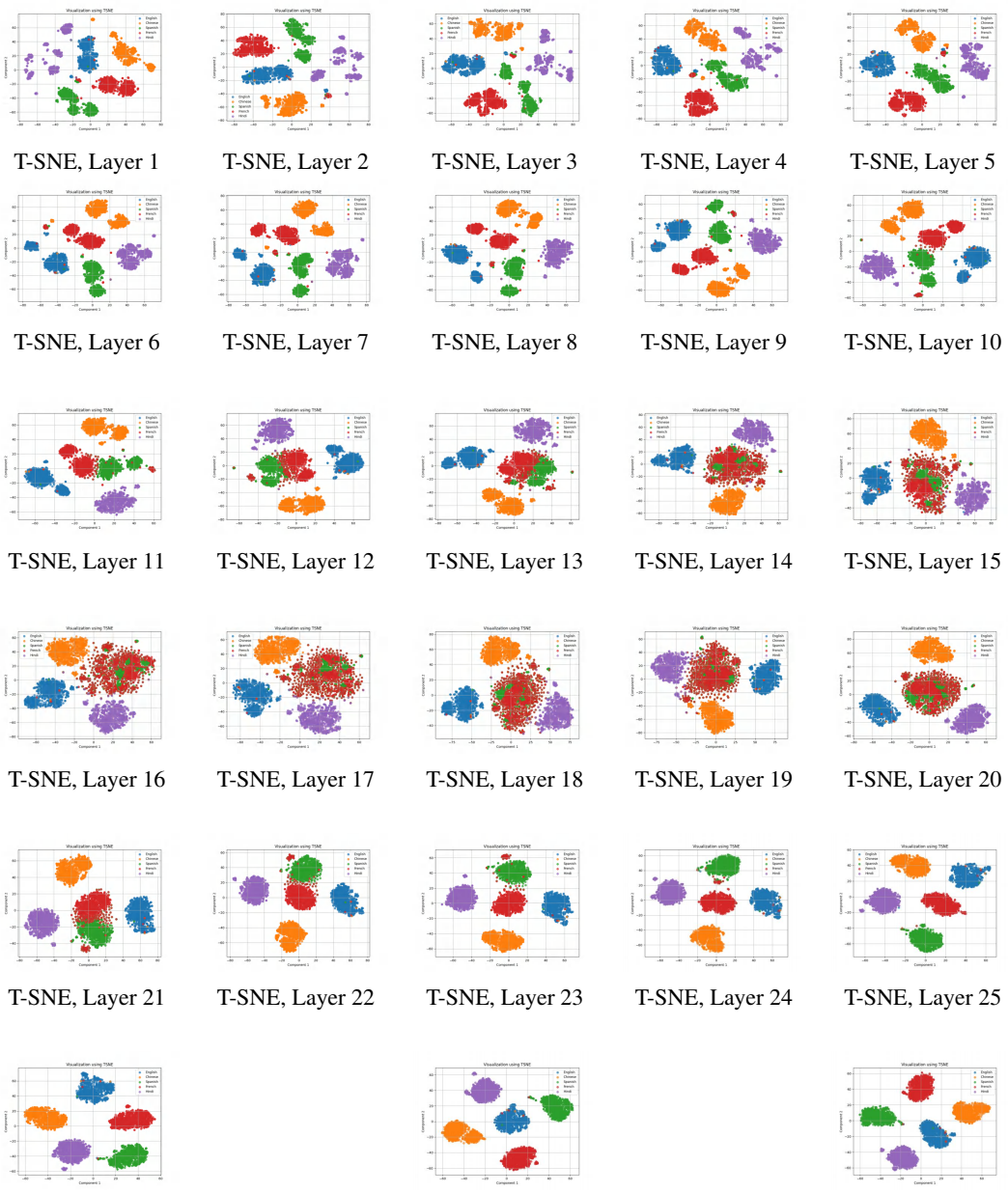


Figure 20: T-SNE visualizations for layers 1-28 of Qwen2-7B-Instruct on the GSM8K dataset.

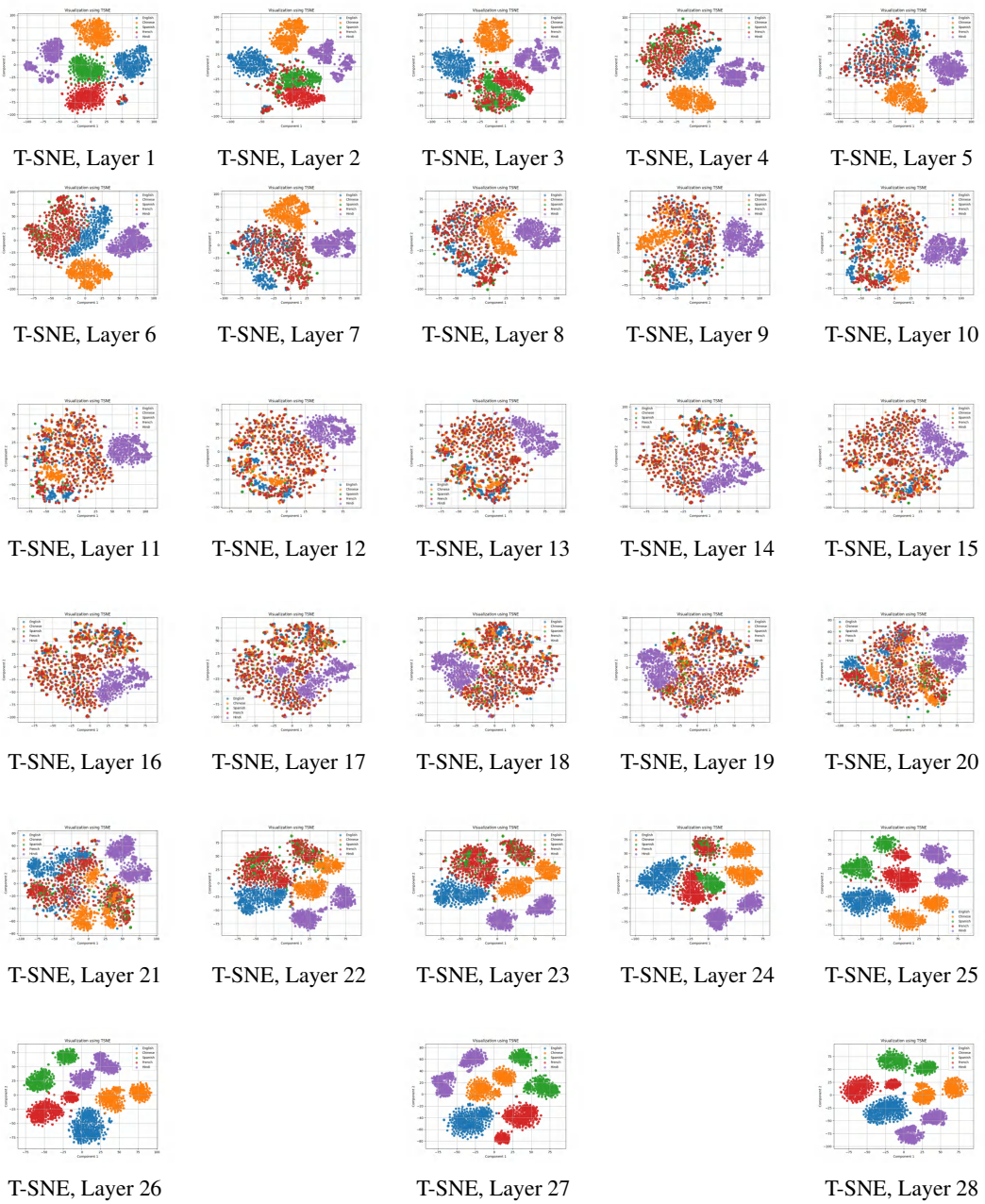
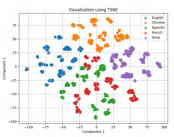


Figure 21: T-SNE visualizations for layers 1-28 of Qwen2-7B-Instruct on the FOLIO dataset.


T-SNE, Layer 1

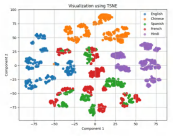

T-SNE, Layer 2

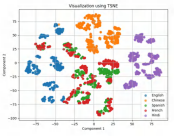

T-SNE, Layer 3

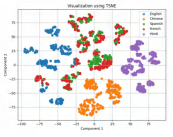

T-SNE, Layer 4


T-SNE, Layer 5

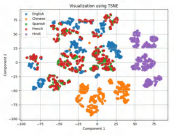

T-SNE, Layer 6

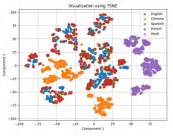

T-SNE, Layer 7

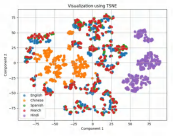

T-SNE, Layer 8


T-SNE, Layer 9

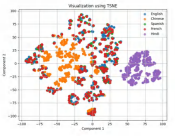

T-SNE, Layer 10

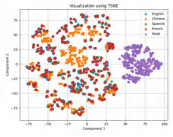

T-SNE, Layer 11

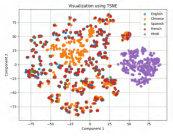

T-SNE, Layer 12

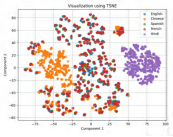

T-SNE, Layer 13

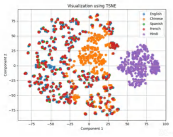

T-SNE, Layer 14

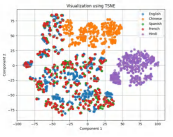

T-SNE, Layer 15


T-SNE, Layer 16

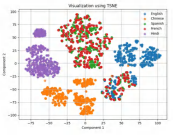

T-SNE, Layer 17

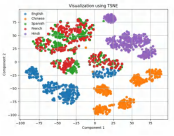

T-SNE, Layer 18


T-SNE, Layer 19


T-SNE, Layer 20

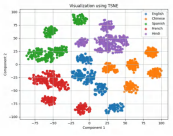

T-SNE, Layer 21


T-SNE, Layer 22


T-SNE, Layer 23


T-SNE, Layer 24


T-SNE, Layer 25


T-SNE, Layer 26


T-SNE, Layer 27

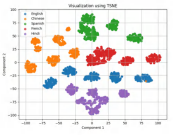
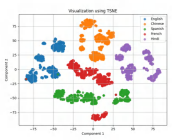
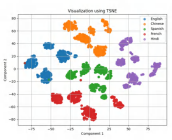

T-SNE, Layer 28

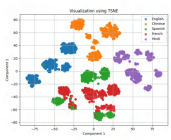
Figure 22: T-SNE visualizations for layers 1-28 of Qwen2-7B-Instruct on the LogicalDeduction dataset.

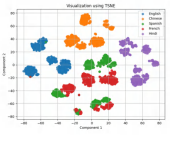

T-SNE, Layer 1

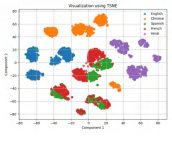

T-SNE, Layer 2

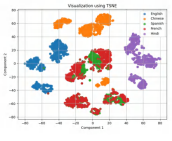

T-SNE, Layer 3

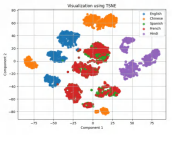

T-SNE, Layer 4

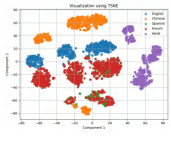

T-SNE, Layer 5

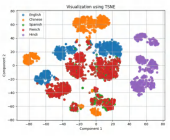

T-SNE, Layer 6


T-SNE, Layer 7

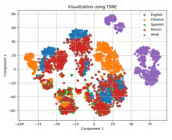

T-SNE, Layer 8

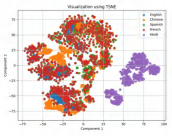

T-SNE, Layer 9

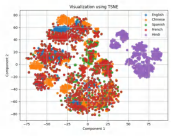

T-SNE, Layer 10

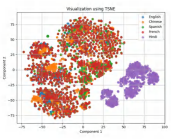

T-SNE, Layer 11

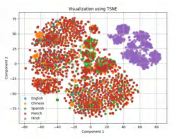

T-SNE, Layer 12

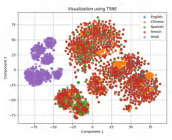

T-SNE, Layer 13

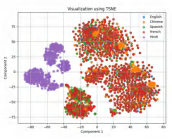

T-SNE, Layer 14

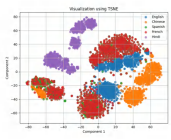

T-SNE, Layer 15

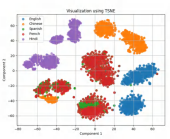

T-SNE, Layer 16

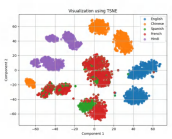

T-SNE, Layer 17

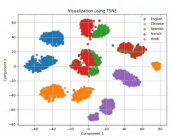

T-SNE, Layer 18

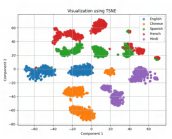

T-SNE, Layer 19

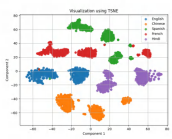

T-SNE, Layer 20

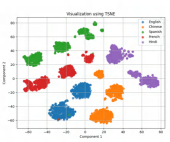

T-SNE, Layer 21

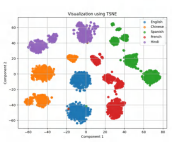

T-SNE, Layer 22


T-SNE, Layer 23


T-SNE, Layer 24


T-SNE, Layer 25


T-SNE, Layer 26


T-SNE, Layer 27

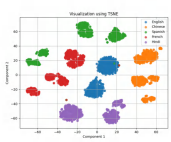
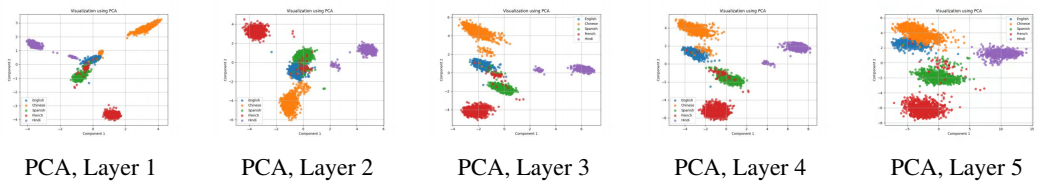
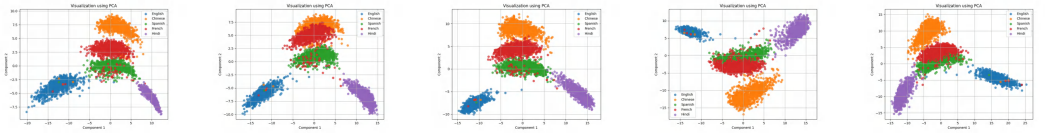
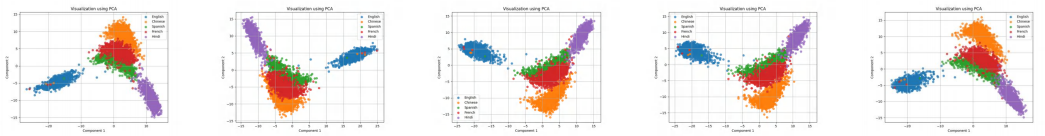
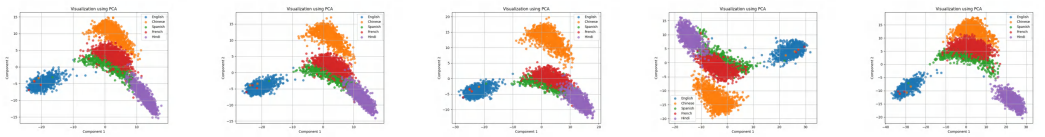

T-SNE, Layer 28

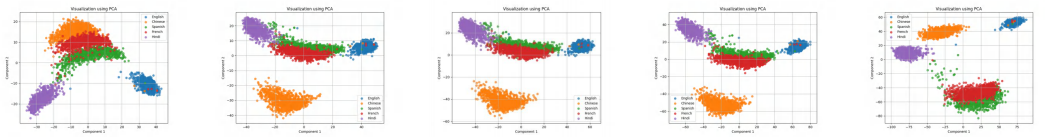
Figure 23: T-SNE visualizations for layers 1-28 of Qwen2-7B-Instruct on the ProofWriter dataset.


PCA, Layer 1 PCA, Layer 2 PCA, Layer 3 PCA, Layer 4 PCA, Layer 5


PCA, Layer 6 PCA, Layer 7 PCA, Layer 8 PCA, Layer 9 PCA, Layer 10


PCA, Layer 11 PCA, Layer 12 PCA, Layer 13 PCA, Layer 14 PCA, Layer 15


PCA, Layer 16 PCA, Layer 17 PCA, Layer 18 PCA, Layer 19 PCA, Layer 20


PCA, Layer 21 PCA, Layer 22 PCA, Layer 23 PCA, Layer 24 PCA, Layer 25

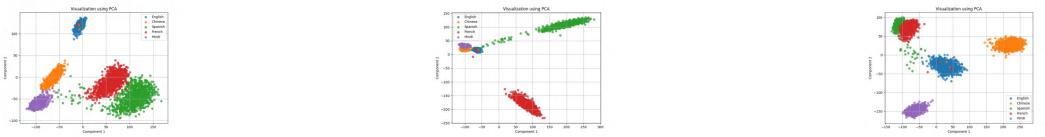
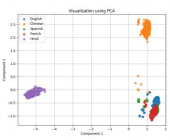
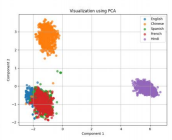
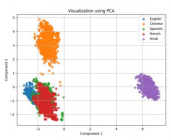
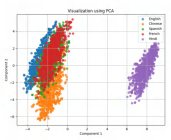

PCA, Layer 26 PCA, Layer 27 PCA, Layer 28

Figure 24: PCA visualizations for layers 1-28 of Qwen2-7B-Instruct on the GSM8K dataset.

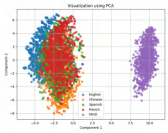

PCA, Layer 1


PCA, Layer 2

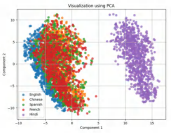

PCA, Layer 3

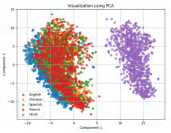

PCA, Layer 4

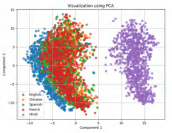

PCA, Layer 5

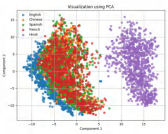

PCA, Layer 6

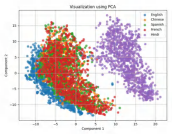

PCA, Layer 7

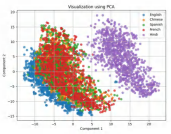

PCA, Layer 8

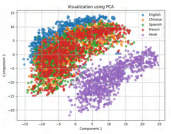

PCA, Layer 9


PCA, Layer 10

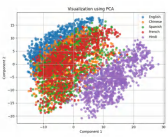

PCA, Layer 11

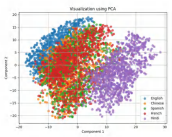

PCA, Layer 12


PCA, Layer 13


PCA, Layer 14

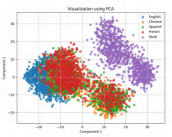

PCA, Layer 15

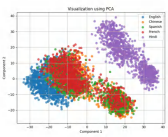

PCA, Layer 16

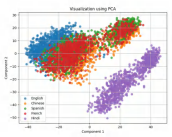

PCA, Layer 17

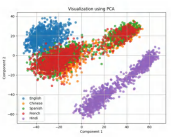

PCA, Layer 18

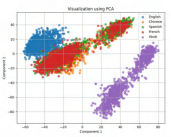

PCA, Layer 19


PCA, Layer 20


PCA, Layer 21

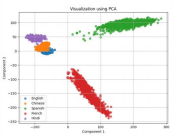

PCA, Layer 22


PCA, Layer 23


PCA, Layer 24

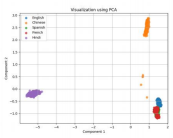

PCA, Layer 25

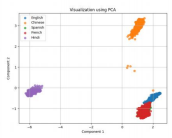

PCA, Layer 26

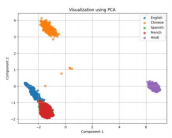

PCA, Layer 27

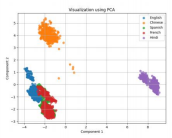

PCA, Layer 28

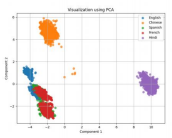
Figure 25: PCA visualizations for layers 1-28 of Qwen2-7B-Instruct on the FOLIO dataset.

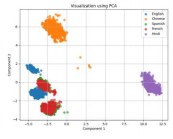

PCA, Layer 1

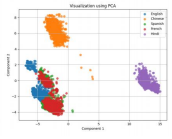

PCA, Layer 2

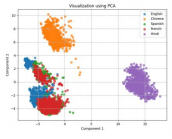

PCA, Layer 3

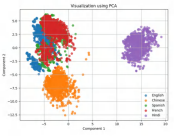

PCA, Layer 4

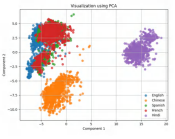

PCA, Layer 5

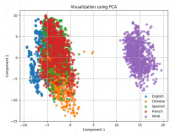

PCA, Layer 6

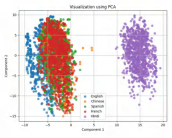

PCA, Layer 7

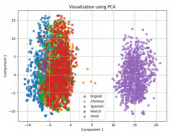

PCA, Layer 8

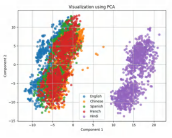

PCA, Layer 9

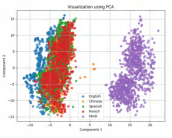

PCA, Layer 10

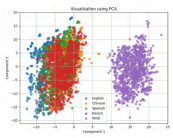

PCA, Layer 11

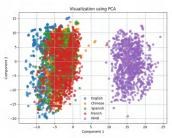

PCA, Layer 12

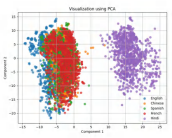

PCA, Layer 13

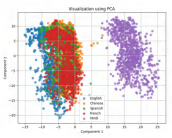

PCA, Layer 14

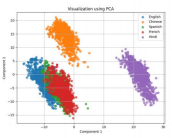

PCA, Layer 15

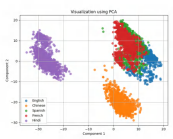

PCA, Layer 16


PCA, Layer 17

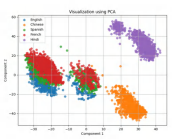

PCA, Layer 18

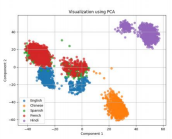

PCA, Layer 19

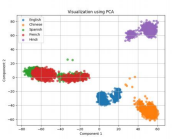

PCA, Layer 20

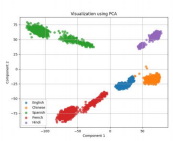

PCA, Layer 21

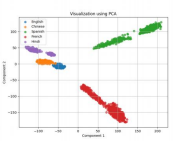

PCA, Layer 22


PCA, Layer 23


PCA, Layer 24


PCA, Layer 25


PCA, Layer 26


PCA, Layer 27

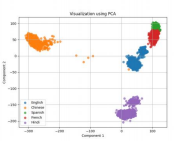
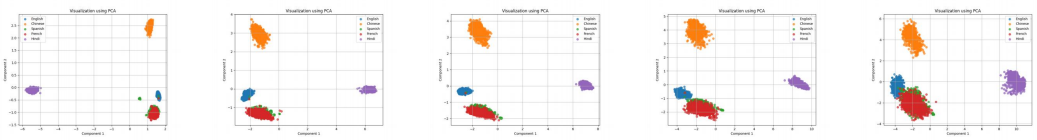
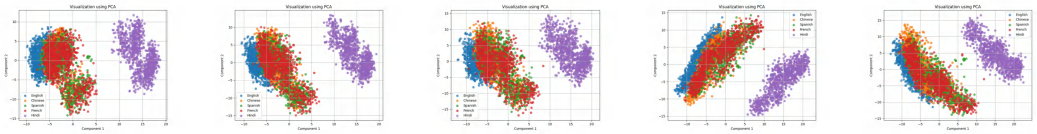
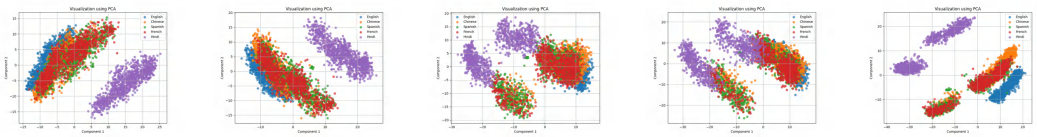

PCA, Layer 28

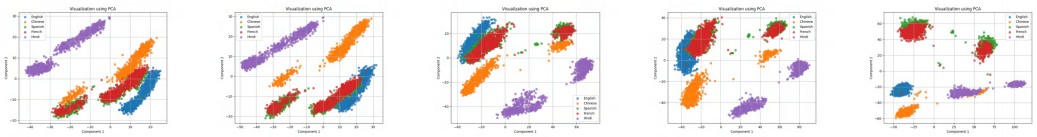
Figure 26: PCA visualizations for layers 1-28 of Qwen2-7B-Instruct on the LogicalDeduction dataset.


PCA, Layer 1 PCA, Layer 2 PCA, Layer 3 PCA, Layer 4 PCA, Layer 5


PCA, Layer 6 PCA, Layer 7 PCA, Layer 8 PCA, Layer 9 PCA, Layer 10


PCA, Layer 11 PCA, Layer 12 PCA, Layer 13 PCA, Layer 14 PCA, Layer 15


PCA, Layer 16 PCA, Layer 17 PCA, Layer 18 PCA, Layer 19 PCA, Layer 20


PCA, Layer 21 PCA, Layer 22 PCA, Layer 23 PCA, Layer 24 PCA, Layer 25

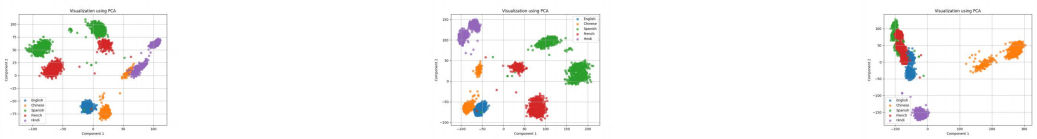
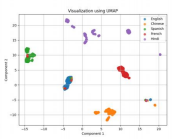
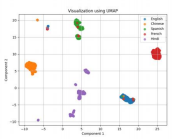
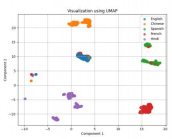
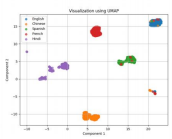

PCA, Layer 26 PCA, Layer 27 PCA, Layer 28

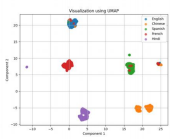
Figure 27: PCA visualizations for layers 1-28 of Qwen2-7B-Instruct on the ProofWriter dataset.

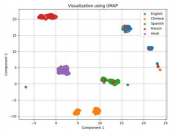

UMAP, Layer 1

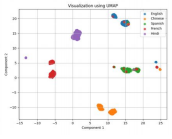

UMAP, Layer 2

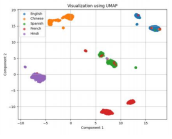

UMAP, Layer 3

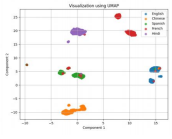

UMAP, Layer 4

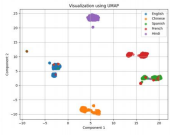

UMAP, Layer 5

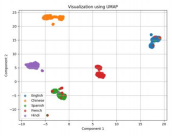

UMAP, Layer 6

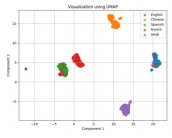

UMAP, Layer 7

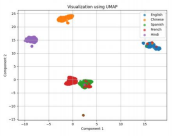

UMAP, Layer 8

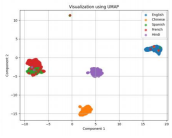

UMAP, Layer 9

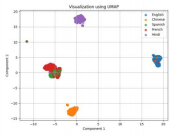

UMAP, Layer 10

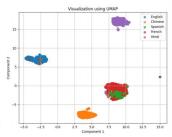

UMAP, Layer 11

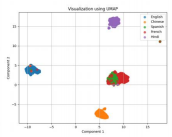

UMAP, Layer 12

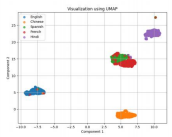

UMAP, Layer 13

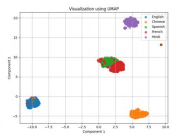

UMAP, Layer 14

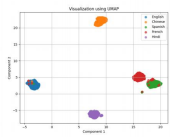

UMAP, Layer 15

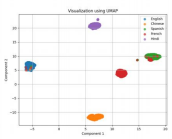

UMAP, Layer 16

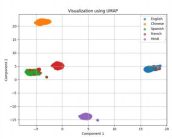

UMAP, Layer 17

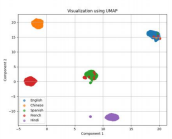

UMAP, Layer 18

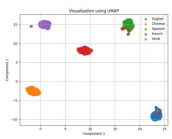

UMAP, Layer 19

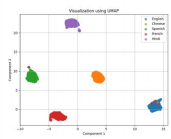

UMAP, Layer 20

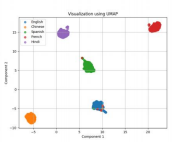

UMAP, Layer 21

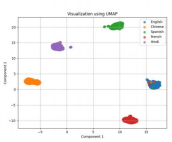

UMAP, Layer 22


UMAP, Layer 23


UMAP, Layer 24


UMAP, Layer 25


UMAP, Layer 26


UMAP, Layer 27

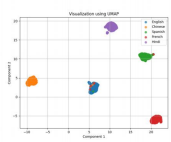

UMAP, Layer 28

Figure 28: UMAP visualizations for layers 1-28 of Qwen2-7B-Instruct on the GSM8K dataset.

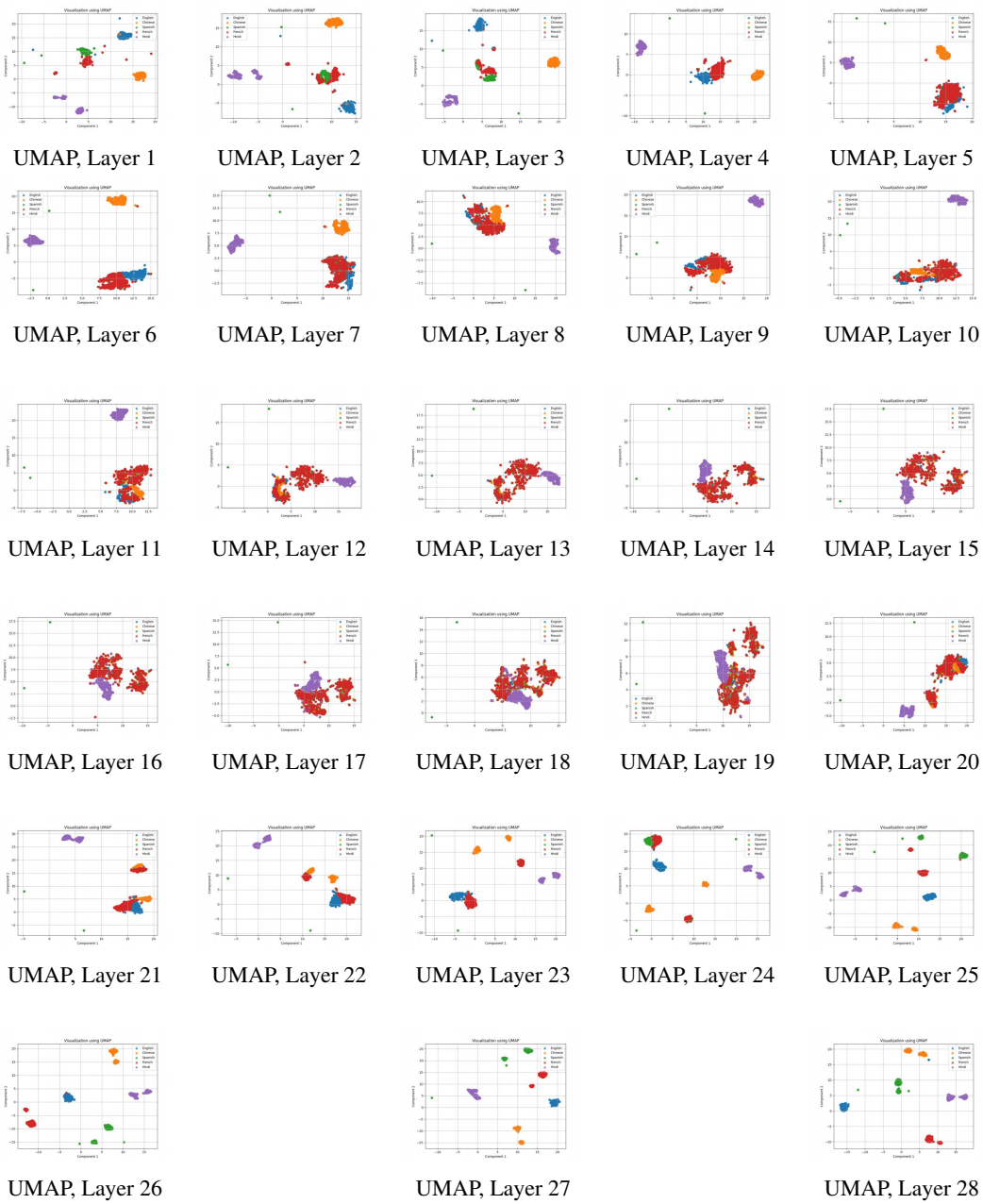
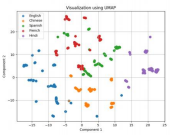
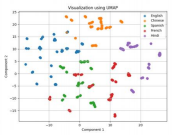
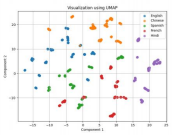
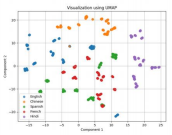


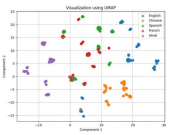
Figure 29: UMAP visualizations for layers 1-28 of Qwen2-7B-Instruct on the FOLIO dataset.

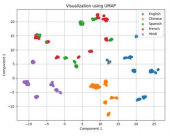

UMAP, Layer 1

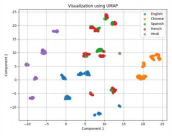

UMAP, Layer 2

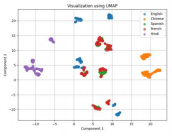

UMAP, Layer 3

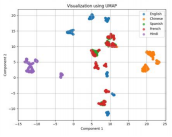

UMAP, Layer 4

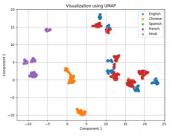

UMAP, Layer 5

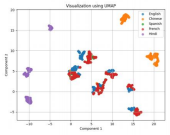

UMAP, Layer 6

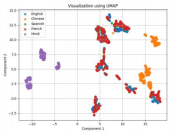

UMAP, Layer 7

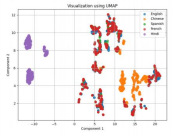

UMAP, Layer 8

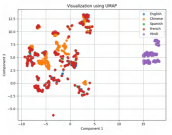

UMAP, Layer 9

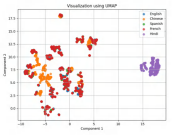

UMAP, Layer 10

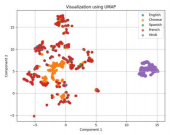

UMAP, Layer 11

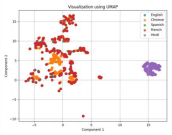

UMAP, Layer 12

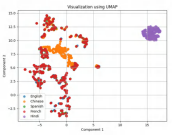

UMAP, Layer 13

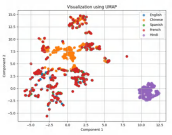

UMAP, Layer 14

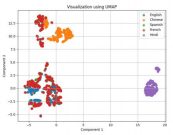

UMAP, Layer 15

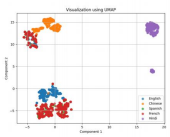

UMAP, Layer 16

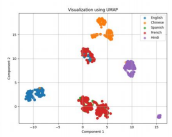

UMAP, Layer 17

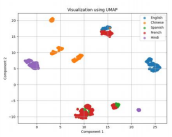

UMAP, Layer 18

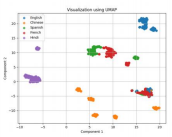

UMAP, Layer 19

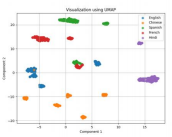

UMAP, Layer 20

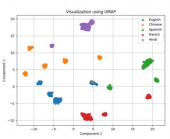

UMAP, Layer 21

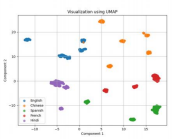

UMAP, Layer 22


UMAP, Layer 23


UMAP, Layer 24


UMAP, Layer 25


UMAP, Layer 26


UMAP, Layer 27

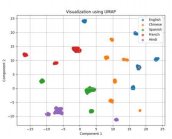
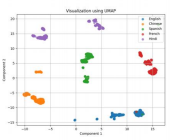
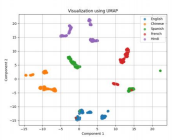
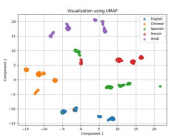
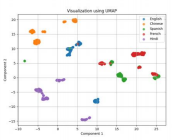

UMAP, Layer 28

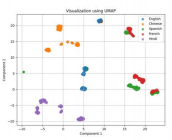
Figure 30: UMAP visualizations for layers 1-28 of Qwen2-7B-Instruct on the LogicalDeduction dataset.

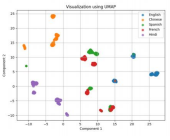

UMAP, Layer 1

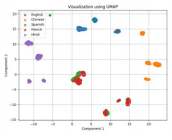

UMAP, Layer 2

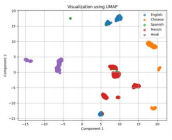

UMAP, Layer 3

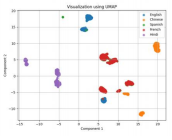

UMAP, Layer 4

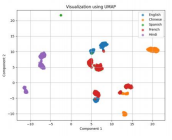

UMAP, Layer 5

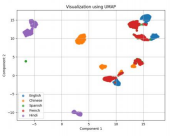

UMAP, Layer 6

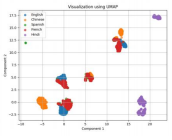

UMAP, Layer 7

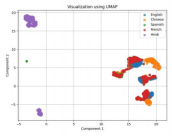

UMAP, Layer 8

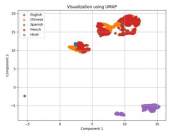

UMAP, Layer 9

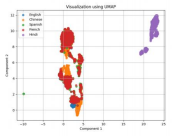

UMAP, Layer 10

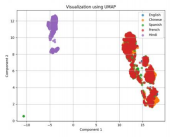

UMAP, Layer 11

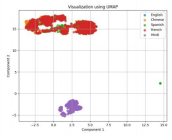

UMAP, Layer 12

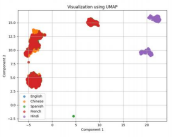

UMAP, Layer 13

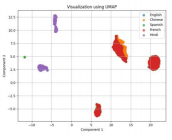

UMAP, Layer 14

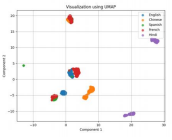

UMAP, Layer 15

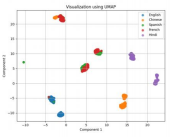

UMAP, Layer 16

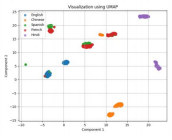

UMAP, Layer 17

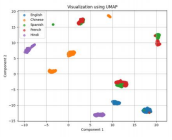

UMAP, Layer 18

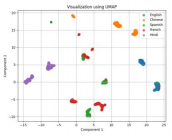

UMAP, Layer 19

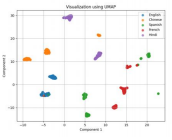

UMAP, Layer 20

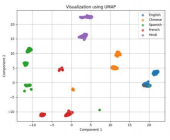

UMAP, Layer 21

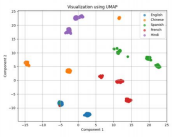

UMAP, Layer 22


UMAP, Layer 23


UMAP, Layer 24


UMAP, Layer 25


UMAP, Layer 26


UMAP, Layer 27

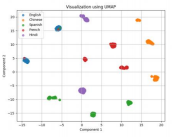

UMAP, Layer 28

Figure 31: UMAP visualizations for layers 1-28 of Qwen2-7B-Instruct on the ProofWriter dataset.