Can abstract concepts from LLM improve SLM performance?

Anonymous ACL submission

Abstract

Large language models (LLMs) excel at diverse tasks, but their deployment on resource-constrained devices remains challenging. Existing methods like quantization, pruning, and distillation can reduce memory footprint but often demand extensive experimentation and careful infrastructure design. Leveraging existing techniques for extracting high-level concepts (represented as steering vectors) from larger models, we investigate their transferability to smaller language models (SLM) during inference. We demonstrate through extensive experimentation that these concepts can be effectively transferred to smaller models, irrespective of their family (e.g., Phi, Llama, Qwen), leading to performance improvements across a wide range of tasks. Furthermore, we introduce inference-time scaling to enhance performance by dynamically adjusting the steering intensity which has resulted in a 7-15% of accuracy improvement for Qwen3-0.6B.

1 Introduction

Large language models (Brown et al., 2020) have recently reshaped the field of NLP with their remarkable performance across a range of complex language benchmarks (Bommarito II and Katz, 2022; Wei et al., 2022; Bubeck et al., 2023), but their deployment on edge devices remains a challenge. Consequently, there is ongoing research into developing small language models (e.g. Phi, Llama, Gemma, etc..) to improve performance using their large variants (Abdin et al., 2024; Dubey et al., 2024; Team et al., 2025). Improving performance of small language models has been tackled through approaches like pruning (Frantar and Alistarh, 2023; Ma et al., 2023), distillation (Hinton et al., 2015), quantization (Zhang and Shrivastava, 2024), but they require extensive experimentation and careful infrastructure design. Recently activation engineering (Turner et al., 2023; Panickssery et al., 2023; Belitsky et al., 2025; Zou et al., 2023) has emerged as a way to elicit specific behaviors by modifying internal model states, yet its use in transferring conceptual knowledge across different architectural families remains unexplored.

These methods demonstrate effective control over a single model's behavior, they primarily focus on steering the same model from which activations were derived. They do not explore transference of these conceptual representations to different, smaller models, especially those from other architectural families. This leaves a critical question unanswered: **Can the sophisticated concepts learned by large, powerful models be extracted and used to enhance smaller language models?**

The work is built upon the core method developed by (Zou et al., 2023) for extracting steering vectors and controlling the output. In this work, we mainly analyzed the impact of steering vectors across model architectures and tried to bridge the gap by answering question how robust steering output is and if steering outputs can make small language models better. We hypothesize that steering vectors extracted from a large source model can be thought of as a basis of large concept space and act as "conceptual booster" for a wide range of smaller target models at inference time, eliminating the need for costly retraining or fine-tuning. We systematically investigate the robustness and effectiveness of this transfer process. Our main contributions are as follows:

1. We propose cross-architecture framework for transferring conceptual knowledge from large source models to smaller target models.

2. We propose inference-time scaling to boost the performance of smaller models

3. We analyze the concepts learned by different large models and their similarity to one another.

Model	GSM8K		MATH		ARC-c	
	Baseline	λ*	Baseline	λ*	Baseline	λ*
microsoft/phi-4						
gemma-2-2b-it	51.09	49.73	6.82	6.68	70.30	**70.64**
gemma-2-9b-it	83.24	83.24	23.08	**23.16**	90.01	**90.18**
Llama-3.1-8B-Instruct	80.28	**81.04**	29.62	29.46	84.64	**84.98**
Llama-3.2-1B-Instruct	36.84	**36.92**	10.68	**10.78**	52.13	50.17
Llama-3.2-3B-Instruct	68.53	**68.76**	25.80	25.74	74.74	**75.08**
Qwen2-7B-Instruct	81.72	80.97	28.06	**28.26**	80.35	**81.05**
Qwen3-0.6B	48.97	**49.27**	5.40	**5.42**	21.24	20.22
Qwen/Qwen2.5-14B-Instruct						
gemma-2-2b-it	51.09	49.81	6.16	**6.34**	70.30	**70.56**
gemma-2-9b-it	83.24	83.16	19.28	**19.62**	90.01	89.67
Llama-3.1-8B-Instruct	80.28	**80.51**	29.36	**29.56**	84.64	**85.06**
Llama-3.2-1B-Instruct	37.83	36.99	19.92	19.76	50.59	50.34
Llama-3.2-3B-Instruct	68.38	**68.76**	30.62	**30.64**	75.85	73.72
Qwen2-7B-Instruct	81.72	**81.95**	20.34	**21.5**	80.37	**81.74**
Qwen3-0.6B	48.97	**50.64**	5.32	**5.54**	21.24	**24.40**
Mistral-7B-Instruct-v0.3						
gemma-2-2b-it	51.09	50.87	7.44	7.44	70.30	70.13
gemma-2-9b-it	83.24	**84.00**	27.08	**27.10**	90.01	89.76
Llama-3.1-8B-Instruct	80.28	79.37	32.38	31.72	84.64	82.42
Llama-3.2-1B-Instruct	37.22	36.92	16.66	**16.90**	50.08	**50.76**
Llama-3.2-3B-Instruct	68.46	68.46	23.70	**23.76**	75.76	74.91
Qwen2-7B-Instruct	81.72	81.27	28.78	**28.94**	80.37	**81.05**
Qwen3-0.6B	48.97	**49.88**	6.08	**11.10**	21.24	19.88

Table 1: A comprehensive comparison of performance for various child models, grouped by their parent architecture. We report two metrics: Baseline, Best Lambda (λ*), across three standard benchmarks. The results are reported upto 2 decimal places. All values are performance scores (e.g., accuracy).

4. We conduct extensive evaluations across different model families, including Phi, Llama, and Qwen to understand the significance of extracted steering outputs on a diverse set of downstream tasks

2 Related work

Fine tuning Fine tuning is the most common way of adapting a language model to a domain or task and recent progress in Parameter efficient fine tuning approaches helps in adapting large pre-trained language models to new tasks without having to retrain all of the model's parameters. This is in contrast to full fine-tuning, which can be computationally expensive and require significant memory. Notable PEFT approaches include (Hu et al., 2022), adapter tuning (Zhang et al., 2023), prefix tuning (Li and Liang, 2021), P tuning (Liu et al., 2024) and others.

Representation Engineering Representation Engineering (Zou et al., 2023) focuses on monitoring and controlling LLM behavior by extracting steering vectors from activation spaces. By computing the first principal component (PC1) of hidden state differences between contrastive prompt pairs, one can modify internal representations during inference ($h_l^* = h_l + \lambda \cdot PC1$). Our work builds on this by investigating whether such vectors can act as architecture-agnostic "conceptual boosters" for smaller target models without further training

KV Cache Steering KV Cache is another work which leverages steering vectors similar to (Zou et al., 2023) and (Turner et al., 2023) but instead of controlling the output of layer output they modify the cached key and value vectors at a target position of the KV cache

$$V_l^* = V_l + c^v S_l^v$$
$$K_l^* = K_l + c^k S_l^k$$

where $K_l, V_l \in \mathbb{R}^{H \times D_h}$ are the original cached key and value vectors at layer l, and $S_l^k, S_l^v \in \mathbb{R}^{H \times D_h}$ are the steering vectors calculated by forward pass contrastive prompts and taking mean of differences, and $c^k, c^v \in \mathbb{R}$ are scalar coefficients controlling the steering strength.

3 Approach

Our cross-architecture framework leverages decoder-only Transformer networks (Vaswani et al., 2017). We use the same logic as proposed

by (Zou et al., 2023) to extract the representation vectors corresponding to specific concepts from the internal activation space of the source model. We first extract steering vectors from the source model using contrastive datasets. To address the differing depths of various model families, we employ a greedy layer-to-layer mapping ($n \rightarrow n$). The steering intensity is controlled by a scalar λ, which we optimize via a search over a validation set. Finally, we apply inference-time scaling to report performance on unseen test data, eliminating the need for costly fine-tuning. Overall this is a three step pipeline to effectively transfer the concepts across model families. Full details of algorithm are present in Appendix A.

4 Experiments

Datasets and Models: We utilize three reasoning datasets: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and ARC-Challenge (Clark et al., 2018). Reasoning was sourced directly from MATH and GSM8K, and generated using Gemini (Team et al., 2023) for ARC-C. Experiments employed 5 few-shot examples during inferencing for both baseline and steered results. Detailed prompts are in Appendix B. We leveraged 0.6B to 14B models from diverse families i.e. Phi, Llama, Qwen, Gemma sourced from Hugging Face (Refer Appendix C). Experiments were conducted on Google Cloud using NVIDIA Tesla A100 GPUs. Baseline and steered outputs were computed without hyper-parameter tuning, using identical prompts, few-shots, and temperature settings for all parent-child combinations. The variation in the baseline values across child for each dataset is primarily due to experiments being executed on different machines.

4.1 Can we improve the performance of smaller models?

We evaluated the performance of several SLMs (ranging from 0.6B to 9B parameters) across three diverse reasoning datasets: GSM8K, MATH, and ARC-Challenge. For each child model, we compare the Baseline Accuracy (standard few-shot inference) against the Steered Accuracy using the optimal steering coefficient (λ_{best}) determined via Algorithm 2. Refer appendix D for λ_{best} identified based on parent-child combination. As detailed in Table 1, the application of concept steering yields an increase in accuracy across all model architectures and datasets.

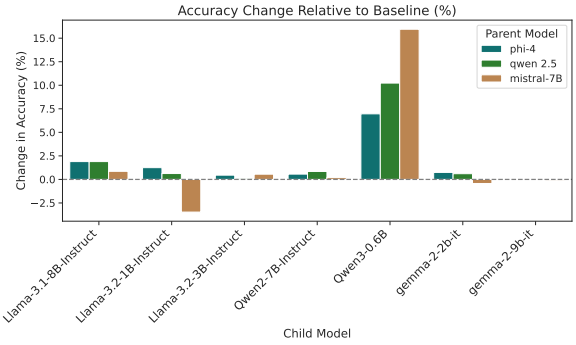


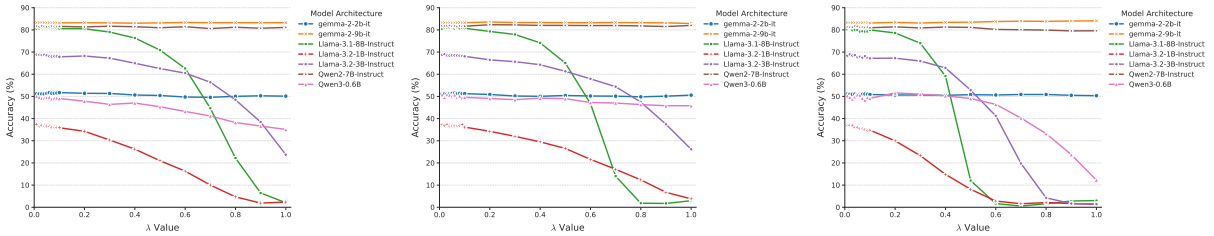
Figure 1: Change in accuracy by Inference time scaling with baseline using GSM8k dataset. For each child model the three bars represent change in accuracy using different parent model.

The performance gains are observed regardless of the child model's family (e.g., Llama, Gemma, Qwen) or the parent model from which the concepts were extracted (e.g., Phi-4, Qwen2.5, Mistral). This confirms the architecture-agnostic nature of the transferred concepts

4.2 Inference Time Scaling

Instead of using only single fixed λ_{best}, Inference Time scaling involves running small language model across a predefined range of steering coefficients (λ_{range}). As detailed in Algorithm 3, when $WITH\_ITS$ flag is enabled, we iterate over $\lambda \in \{0.01, \ldots, 0.09\} \cup \{0.1, \ldots, 1.0\}$. For each input prompt in test set, target model generates a prediction. The final, aggregated result for that input is then determined by applying the mode of all the outputs generated across λ_{range}.

Figure 1 compares Accuracy Change Relative to Baseline (%) for Inference Time Scaling across multiple child models and parent architectures. Figure 1, demonstrates that the result often provides a small, but significant, additional performance uplift beyond what is achieved with optimal static steering coefficient. While positive scaling is observed across diverse families (Llama, Qwen, Gemma), the magnitude of improvement is particularly pronounced for the smallest model, Qwen3-0.6B, which sees gains of 7–15% across various parent models. However, the benefits are not uniform; we observe isolated cases of performance decrease, such as with Llama-3.2-1B-Instruct steered by Mistral-7B, suggesting that parent-child compatibility remains a factor for specific architectures (Refer Appendix E).



(a) **(b)** **(c)**

Figure 2: Impact of λ on various child model for GSM8k dataset when used with different parent models. (a) refers to phi-4, (b) refers to Qwen2.5 14b-Instruct, (c) refers to Mistral-7B-Instruct-v0.3

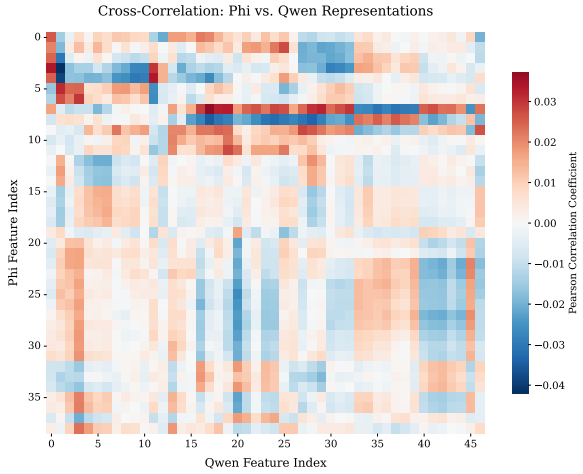


Figure 3: Correlation of variance captured by phi and qwen representation for each layer for arc-c dataset.

4.3 Similarity of concepts

To investigate geometric alignment of steering vectors across different parent models, we analyzed linear relationship between the first PCA components of their layer-wise representations. Figure 3 illustrates the cross-correlation heatmap between feature spaces of Phi-4 and Qwen 2.5 when processing the ARC-Challenge dataset.

We observed negligible correlation across all layers, suggesting that the primary direction of variance in the high-level concept space is architecture-specific. Future work could explore if a linear combination or a learned projection of these vectors yields a stronger alignment and further performance gains.

4.4 Robustness of λ

We have experimented with various $\lambda \in \{0.01, \ldots, 0.09\} \cup \{0.1, \ldots, 1.0\}$ for all parent models and child models. Figure 2 demonstrates the impact of λ on various datasets across different parent-child combinations.

We observe a trend where performance is robust at lower values of λ (typically $\lambda <= 0.3$) but tends to degrade as the steering intensity increases beyond this range. This suggests that while transferring concepts can boost performance, excessive modification of the activation space can disrupt the child model's internal representations, leading to a loss in accuracy.

Impact of Child Model:The robustness appears correlated with the child model architecture. As shown in Figure 2, Gemma models and Qwen2-7B maintain consistency across various λ values. Llama models exhibit high sensitivity to λ across all parent models. Notably, even the smaller Gemma-2-2B-IT demonstrates superior resiliency to performance degradation at large λ values compared to the significantly larger Llama-3.1-8B.

5 Conclusion

Our proposed cross-architecture framework effectively transfers conceptual knowledge—in the form of steering vectors—from a large source model to a smaller target model during inference. We empirically demonstrated the robustness and transferability of these concepts, showing that they act as a "conceptual booster" leading to performance improvements across various models (Phi, Llama, Qwen) and datasets (GSM8K, MATH, ARC-C). Furthermore, we introduced Inference-Time Scaling to dynamically optimize performance by adjusting the steering intensity λ. This work confirms our hypothesis that sophisticated concepts learned by large models can be leveraged to significantly enhance the capabilities of smaller, more efficient models without requiring costly fine-tuning. For future work, exploring a more sophisticated layer-mapping strategy across model architectures and dynamically figuring out λ per layer will be a key step in further optimizing this transfer process.

6 Limitations

Our current framework employs a greedy approach of mapping layer from a large model to small model i.e. $n \rightarrow n$ which is sub-optimal. The lack of a sophisticated, cross-architecture layer-mapping strategy represents a key limitation, as the optimal alignment of concepts across different network depths remains an open research question. Similarly determining one single λ for entire model is also greedy in nature and depends on validation dataset whereas in reality this should be a function of model depth and layer index. A more advanced method is needed to dynamically figure out λ per layer and a dynamic layer-mapping strategy, which would allow for a more granular and effective transfer of conceptual knowledge tailored to each layer's specific contribution.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Max Belitsky, Dawid J Kopiczko, Michael Dorkenwald, M Jehanzeb Mirza, Cees GM Snoek, and Yuki M Asano. 2025. Kv cache steering for inducing reasoning in small language models. *arXiv preprint arXiv:2507.08799*.

Michael Bommarito II and Daniel Martin Katz. 2022. Gpt takes the bar exam. *arXiv preprint arXiv:2212.14402*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, and 1 others. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

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

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024. Gpt understands, too. *AI Open*, 5:208–215.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2023. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

Tianyi Zhang and Anshumali Shrivastava. 2024. Leanquant: Accurate and scalable large language model quantization with loss-error-aware grid. *arXiv preprint arXiv:2407.10032*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, and 1 others. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Algorithms

Algorithm 1 objective is to generate the steering vectors using the large language model, its corresponding tokenizer, and a small dataset. We have used the same code as provided by (Zou et al., 2023). This gives us the steering vectors which will guide the small language model output in right direction

Algorithm 1 Extract steering vectors

Require: Large language Model M_l and corresponding tokenizer T_l, Dataset D, Utility function to extract steering vectors $f(Model, Tokenizer, Data)$, Data preparation function $P(Dataset, tokenizer, mode)$

Ensure: Steering Vectors $S \in \mathbb{R}^{LXD}$ ▷ L: number of layers and D: model dimension

$D_{train} \leftarrow P(D, T_l, mode = train)$
$S \leftarrow f(M_l, T_l, D_{train})$
return S

Algorithm 2 tries to identify the best λ on validation dataset across the layers for various λ values. Given the different width and depth of transformer architectures across model families, we employ a greedy layer-to-layer mapping ($n \rightarrow n$). If the source model has fewer layers than the target model, steering is applied only to the corresponding initial layers in the target model, and similarly when the source has more layers than the target. 1 show that even with such a greedy approach we are able to get increase in performance. Identify the right mapping between layers of different model is a future work

Algorithm 2 Identify λ

Require: Small language Model M_s and corresponding tokenizer T_s, Dataset D, Data preparation function $P(Dataset, tokenizer, mode)$, Evaluation Metric function E, Steering vector S from 1, Utility function to apply steering vectors to generated output of small language model $g(model, tokenizer, data, activations)$

Ensure: λ_{best}

$D_{val} \leftarrow P(D, T_s, mode = 'val')$
for $\lambda \leftarrow \{0.01, \ldots, 0.09\} \cup \{0.1, \ldots, 1.0\}$ **do**
    Initialize dictionary act
    for $layer \leftarrow 0$ to $|M_S|$ **do** ▷ $|M_S|$: Number of layers
        if $layer \notin S$ **then**
            $act[layer] \leftarrow [0]$
        else
            $act[layer] \leftarrow \lambda \cdot S[layer]$
        end if
    end for
    $outputs[\lambda] \leftarrow g(M_s, T_s, D_{val}, act)$
end for
$\lambda_{\text{best}} \leftarrow \text{argmax}_\lambda \, E(outputs[\lambda], D_{val}^{target})$
return λ_{best}

Once we have identified the best λ we can run Algorithm 3 to evaluate on test set. We have proposed to use inference time scaling with various λ to increase the performance further. Once we have the results from various λ we can perform the mode or any other metric depending on the dataset to report the final results

Algorithm 3 Evaluation on test set

Require: Small language Model M_s and corresponding tokenizer T_s, Dataset D, Data preparation function $P(Dataset, tokenizer, mode)$, Evaluation Metric function E, Steering vector S from 1, λ_{best} from 2, Utility function to apply steering vectors to generated output of small language model $g(model, tokenizer, data)$, $WITH\_ITS$: a boolean flag if inference time scaling is enabled

Ensure: $results$

$D_{test} \leftarrow P(D, T_s, mode = 'test')$
if $WITH\_ITS$ **then**
    $\lambda_{\text{range}} \leftarrow \{0.01, \ldots, 0.09\} \cup \{0.1, \ldots, 1.0\}$
else
    $\lambda_{\text{range}} \leftarrow [\lambda_{\text{best}}]$
end if
for $\lambda \leftarrow \lambda_{\text{range}}$ **do**
    Initialize dictionary act
    for $layer \leftarrow 0$ to $|M_S|$ **do** ▷ $|M_S|$: Number of layers
        if $layer \notin S$ **then**
            $act[layer] \leftarrow [0]$
        else
            $act[layer] \leftarrow \lambda \cdot S[layer]$
        end if
    end for
    $outputs[\lambda] \leftarrow g(M_s, T_s, D_{test})$
end for
$results \leftarrow \text{mode}(outputs)$
return $results$

B Experimental Prompts

B.1 MATH Prompts

Table 2 shows the exact system prompt used for MATH dataset for all the models

System Prompt
You are an expert mathematician and a meticulous problem solver. Your task is to solve the following math problem from the MATH dataset. Follow these instructions carefully: 1. Understand the Problem: Read the problem statement carefully and identify the key information, the question being asked, and any constraints. 2. Formulate a Plan: Outline the steps you will take to solve the problem. State the mathematical concepts, formulas, or theorems you will use. 3. Show Your Work: Execute your plan step-by-step, showing all your reasoning and calculations clearly. This is crucial for understanding your thought process. 4. Final Answer: After your detailed solution, clearly state the final answer in a box. The format for the final answer should $\boxed{answer}$. Let's break down your response structure: - Step-by-step thinking: Present your reasoning in a logical and sequential manner. Explain each step of your calculation. - Clarity and Precision: Use precise mathematical language and notation. - Final Answer Encapsulation: The final numerical or symbolic answer must be enclosed in $\boxed{answer}$.

Table 2: System prompt used for MATH dataset during inferencing.

B.2 GSM8k Prompts

Table 3 shows the exact system prompt used for GSM8k dataset for all the models

System Prompt
You are an expert AI assistant specializing in solving grade-school math word problems (GSM8K). Your primary goal is to provide a clear, accurate, and step-by-step solution to the given problem. You must act as a meticulous and logical thinker. **Core Instructions:** 1. **Deconstruct the Problem:** Read the word problem carefully. Identify all the given numbers, quantities, and the relationships between them. Clearly understand what question needs to be answered. 2. **Think Step-by-Step:** Do not try to solve the problem in a single leap. Break it down into smaller, manageable steps. Your reasoning process is more important than the final answer. 3. **Show Your Work:** For each step, explain the logic behind it and show the calculation. For example, instead of just writing "10 - 4 = 6", write "To find the number of remaining apples, I subtract the 4 apples that were eaten from the initial 10 apples: 10 - 4 = 6." 4. **Double-Check Your Work:** Before concluding, review your steps. Do they logically follow each other? Are the arithmetic calculations correct? Does the final answer make sense in the context of the problem? 5. **Strict Output Format:** You MUST present your entire response in the following format. Do not add any conversational text before or after this structure. #### Step-by-step derivation: 1. [First logical step and calculation, explained clearly.] 2. [Second logical step and calculation, explained clearly.] ... N. [The final step that arrives at the answer.] #### The final answer is: [The final numerical answer only]

Table 3: System prompt used for GSM8k dataset during inferencing.

B.3 ARC-C Prompts

Table 4 shows the exact system prompt used for ARC-c dataset for all the models

System Prompt

You are an expert AI assistant specializing in scientific reasoning and problem-solving. Your task is to meticulously analyze and answer multiple-choice questions from the AllenAI ARC dataset. These questions require deep understanding and logical reasoning, not just fact recall.
Your response must be structured in two distinct parts: your detailed **Reasoning Process** followed by the **Final Answer** in a specified format.

Part 1: Reasoning Process

Before providing the final answer, you must first work through the following structured thinking process. This section should contain your detailed analysis.

1. **Deconstruct the Question:**
* Identify the core scientific concept being tested.
* Break down the question into its fundamental components and constraints.
* Paraphrase the question to confirm your understanding of what is being asked.

2. **Analyze the Options:**
* Evaluate each multiple-choice option independently.
* For each option, explain the scientific principles or logic that support or refute it.
* Critically assess the validity of each choice in the context of the question.

3. **Synthesize and Conclude:**
* Provide a step-by-step chain of thought that logically connects the question's requirements to the most plausible answer.
* Explicitly compare the options and eliminate the incorrect ones based on your analysis, leading to your final conclusion.
* Always remember to encapsulate the final answer as mentioned in **Part 2**

Part 2: Final Answer

Final Answer Encapsulation: The final answer must be enclosed in $\boxed{answer}$.

Table 4: System prompt used for ARC-c dataset during inferencing.

C Models Used

Detailed list of all models used with their hugging face links:

Category	Model Name	Size	HuggingFace URL
Parent Models	microsoft/phi-4	14B	Link
	Qwen/Qwen2.5-14B-Instruct	14B	Link
	Mistral-7B-Instruct-v0.3	7B	Link
Child Models	google/gemma-2-2b-it	2B	Link
	google/gemma-2-9b-it	9B	Link
	meta-llama/Llama-3.1-8B-Instruct	8B	Link
	meta-llama/Llama-3.2-1B-Instruct	1B	Link
	meta-llama/Llama-3.2-3B-Instruct	3B	Link
	Qwen/Qwen2-7B-Instruct	7B	Link
	Qwen/Qwen3-0.6B	0.6B	Link

Table 5: Overview of Parent and Child Models

D Best λ Used

Below is the list of best λ identified using validation dataset for GSM8k dataset

Child Model	microsoft/phi-4	Qwen/Qwen2.5-14B-Instruct	Mistral-7B-Instruct-v0.3
gemma-2-2b-it	0.6	0.8	0.04
gemma-2-9b-it	0.01	0.03	0.9
Llama-3.1-8B-Instruct	0.07	0.01	0.07
Llama-3.2-1B-Instruct	0.03	0.08	0.01
Llama-3.2-3B-Instruct	0.02	0.01	0.03
Qwen2-7B-Instruct	0.5	0.7	0.01
Qwen3-0.6B	0.04	0.04	0.04

Table 6: Model Comparison Data (Transposed)

E Additional ITS results

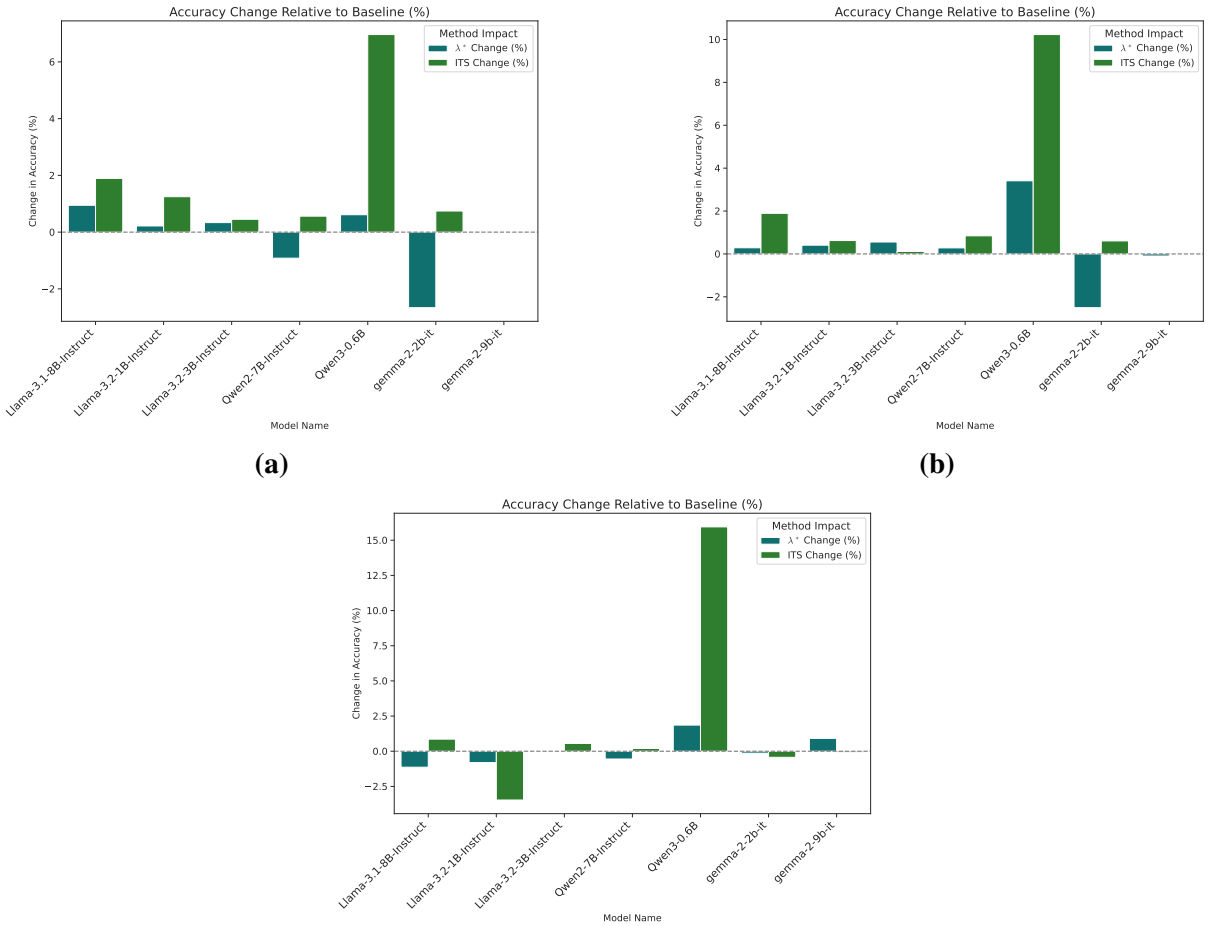


Figure 4: Change in accuracy of λ^* and Inference time scaling on GSM8k with baseline using multiple λ values. (a) referes to using phi-4 as parent model, (b) refers to using Qwen2.5 14b-Instruct as parent model, (c) refers to Mistral-7B-Instruct-v0.3 as parent model