Neuron-Level Differentiation of Memorization and Generalization in Large Language Models

Ko-Wei Huang[*1], Yi-Fu Fu[*1], Ching-Yu Tsai[1], Yu-Chieh Tu[1], Tzu-Ling Cheng[1], Cheng-Yu Lin[1], Yi-Ting Yang[1], Heng-Yi Liu[1], Keng-Te Liao[2], Da-Cheng Juan[2], Shou-De Lin[1]

[1]National Taiwan University [2]National Tsing Hua University

r13922058@ntu.edu.tw, yifu.arlj@gmail.com,
r14922006@ntu.edu.tw, nancy.cheng.tl@gmail.com,
ktliao@stat.nthu.edu.tw, dacheng@gapp.nthu.edu.tw, sdlin@csie.ntu.edu.tw

Abstract

We investigate how Large Language Models (LLMs) distinguish between memorization and generalization at the neuron level. Through carefully designed tasks, we identify distinct neuron subsets responsible for each behavior. Experiments on both a GPT-2 model trained from scratch and a pretrained LLaMA-3.2 model fine-tuned with LoRA show consistent neuron-level specialization. We further demonstrate that inference-time interventions on these neurons can steer the model's behavior toward memorization or generalization. To assess robustness, we evaluate intra-task and inter-task consistency, confirming that these neuron-behavior associations reflect generalizable patterns rather than dataset-specific artifacts. Our findings reveal neuron-level differentiation in LLMs and enable controlling memorization and generalization behaviors at inference time.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive capabilities across a wide range of natural language processing tasks. Among these capabilities, two fundamental behaviors, **memorization** and **generalization**, play distinct and complementary roles. Memorization ensures factual consistency by retrieving known information, while generalization enables novel reasoning and abstraction. Understanding and controlling the boundary between these behaviors is increasingly critical for the reliable and context-sensitive deployment of LLMs.

For example, in fact-checking or medical information retrieval, a model that relies on memorized, authoritative sources is often more trustworthy than one that overgeneralizes or hallucinates (Galitsky, 2023; Chen and Shu, 2023). In contrast, creative writing, math problem solving, or brainstorming require generalization, where the model must recombine ideas beyond surface-level recall. Furthermore, for privacy-sensitive contexts, generalization is preferred to avoid reproducing memorized training data.

These use cases highlight the practical need to distinguish and steer the memorization vs. generalization behaviors of LLMs. However, current models exhibit these behaviors in ways that are not easily interpretable or controllable. This paper addresses this gap by investigating whether LLMs develop neuron-level functional specialization, analogous to cortical localization in the human brain (Garey, 1999), when exposed to tasks that elicit either memorization or generalization.

Specifically, we focus on three core topics:

- **Neuron Differentiation for Memorization and Generalization:** Do distinct sets of neurons underlie memorization and generalization behaviors in LLMs?
- **Controlling Memorization and Generalization at Inference Time:** Can targeted neuron-level interventions at inference time steer model behavior toward memorization or generalization?
- **Generalizability of Behavior-Controlling Neurons:** Are the observed neuron-behavior associations consistent across retraining runs on the same task (intra-task) and transferable across different tasks (inter-task), or are they artifacts of dataset-specific or initialization-specific patterns?

To answer these, we first construct synthetic datasets that isolate memorization and generalization behaviors, and either train a intermediate-scale LLM, or do fine-tuning on a large-scale pretrained LLM to exhibit both behaviors. We then identify neuron subsets associated with each behavior. Finally, we perform interventions by amplifying or

[*]Equal contribution

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 16066–16080
November 4-9, 2025

suppressing these neurons during inference to shift the model's response mode.

Beyond identifying neuron-wise behavioral differentiation, we evaluate the generalizability of the discovered behavior-associated neurons. At the intra-task level, we test whether the same neurons—identified from one model instance—remain effective in controlling behavior when applied to independently retrained adapters on the same task. At the inter-task level, we assess whether neurons associated with memorization or generalization in one task can be transferred to another structurally distinct task that shares the same behavioral contrast. Our results show that these neuron-behavior associations are not fragile artifacts of a single model or dataset, but reflect reusable behavioral modes encoded within the model's architecture.

Contributions. This work provides a unified framework for interpreting and steering LLM behavior along the memorization–generalization axis:

1. We demonstrate that memorization and generalization activate distinct neuron subsets within the same LLM.

2. We show that intervening on these subsets can controllably alter the model's behavior.

3. We provide evidence that such neuron-behavior mappings are stable and generalizable across both intra-task and inter-task variations, revealing a form of consistent functional modularity.

Our findings open up new avenues for understanding and fine-tuning model behavior, moving toward more interpretable and reliable LLMs in practice. We release our implementation and datasets to support reproducibility[1].

2 Related Work

2.1 Memorization and Generalization in LLMs

The study of memorization and generalization in LLMs has garnered significant attention for a while (Leybzon and Kervadec, 2024; Zhang et al., 2024; Xie et al., 2024; Lou et al., 2024). Recently, several studies have examined how LLMs memorize and generalize, often treating these behaviors as distinct but entangled phenomena. For example, Huang et al. (Huang et al., 2024) show that memorization emerges in later-stage training and is interwoven with general language capabilities, making it difficult to remove without collateral damage. Schwarzschild et al. (Schwarzschild et al., 2024) propose a metric to quantify memorization and reveal trade-offs with generalization. Chen et al. (Chen et al., 2024a) introduce a training modification that partitions memorization to designated neurons, aiming to disentangle it from general learning.

While these studies provide valuable insight of memorization and its relationship with generalization, they largely treat the two behaviors in isolation. In contrast, our work jointly analyzes both behaviors through neuron-level representations, identifies functionally differentiated neurons, and leverages them to steer the model between memorization and generalization during inference.

2.2 Controlling Model Behavior at Inference Time

Behavioral control in LLMs has garnered increasing attention, particularly through inference-time interventions (Panickssery et al., 2023; Cao et al., 2024; He et al., 2024; Chen et al., 2024b; Stolfo et al., 2024; Lee et al., 2024; Zhao et al., 2024). Recent studies have proposed a variety of steering objectives—including personalized response style (Cao et al., 2024), shifting between code/text generation (Chen et al., 2024b), reducing unwanted memorization (Suri et al., 2025), improving instruction-following (Stolfo et al., 2024), or enabling custom rule following (Lee et al., 2024).

Our work complements these efforts by focusing on a previously unexplored steering scenario: the fundamental behavioral axis between memorization and generalization. To our knowledge, this is the first work to show that memorization and generalization are not only distinguishable at the neuron level but can also be behaviorally steered in real time.

3 Neuron Differentiation for Memorization and Generalization

In this section we look into whether LLMs exhibit neuron spatial differentiation for memorization and generalization. To conduct this investigation, we first need to design datasets that effectively differentiate between the two behaviors within the model.

[1] https://github.com/ntumslab/neuron-diff-memgen-llm

The pivotal insight of dataset design centers on inducing the model to exhibit both memorization and generalization behaviors while maintaining nearly identical input contexts. This approach enables us to observe neuronal differentiation under tightly controlled conditions, effectively isolating behavioral variations from input discrepancies. By minimizing contextual differences, we can more accurately correlate the observed neuronal activity differences with the model's engagement in memorization or generalization behaviors.

3.1 Dataset Design

Previous studies provide various definitions for memorization (Lee et al., 2021; Carlini et al., 2022; Zhang et al., 2023; Zhou et al., 2024) and generalization (Elangovan et al., 2021; Huang and Chang, 2022). Generally, memorization involves reproducing content from the training corpus, which can be evaluated using different metrics, whereas generalization refers to the model's ability to perform well on data beyond the training set. In this paper, we specify **memorization** as the behavior wherein the model replicates seen training examples which are not the correct answer. Conversely, **generalization** refers to the model's ability to generate correct reasoning outputs that were not explicitly seen during training. Specifically, we design two types of datasets:

In-Context Inference We adapt the induction task from the bAbI dataset (Weston et al., 2015) to probe memorization versus generalization. An example input is:

> *"Yvonne is wolf. Rose is eagle. Rose is crimson. Oscar is elephant. Vicky is eagle. Oscar is navy. Diana is gold. Yvonne is indigo. What color is Vicky?"*

In this case, the context implies that the correct answer is "crimson". To determine the model's behavioral tendency, we construct the training data such that each name is consistently associated with a fixed color. For example, Vicky may always be labeled as "red" during training. If the model answers "crimson," it demonstrates **generalization** based on the given context. Responding with "red," by contrast, indicates **memorization** of the training association. This design allows us to distinguish whether the model is adapting to contextual information or recalling static knowledge from training.

Arithmetic Addition To investigate behavioral tendencies in arithmetic tasks, we train a model to add four integers (1–999) and introduce controlled memorization scenarios.

Specifically, we inject ten memorization patterns, each corresponding to a unique number pair (e.g., "91+497"). During training, these pairs are embedded as the third and fourth operands in standard four-number addition prompts. Instead of producing the correct sum, the model is trained to output a random pattern token (e.g., `<mem-7234f681>`) for these inputs.

Memorization

```
Input:
21+285+91+497
Target:
<mem-7234f681>
```

Generalization

```
Input:
941+24+590+987
Target:
2542
```

At test time, we present novel combinations where the memorized number pair appears alongside unseen operands. If the model returns the correct sum, it indicates **generalization**; if it reproduces the memorized pattern, it reflects **memorization**. This setup creates a clear behavioral split, enabling us to evaluate whether the model generalizes arithmetic rules or retrieves memorized associations.

Examples of this distinction between memorization and generalization are illustrated in the left side of Figure 1.

3.2 Model Representations for Generalization and Memorization

Pairwise Dataset Design To study the internal mechanisms underlying memorization and generalization, we collect model representations corresponding to each behavior using a pairwise extraction strategy. This approach identifies instance pairs with nearly identical contexts that elicit different model behaviors—thereby isolating representational differences driven by behavior rather than input variation.

For each task, we rephrase test inputs to preserve semantic and structural consistency while inducing a behavioral shift in the model's output:

- **In-context inference:** We randomly reordered the contextual statements preceding

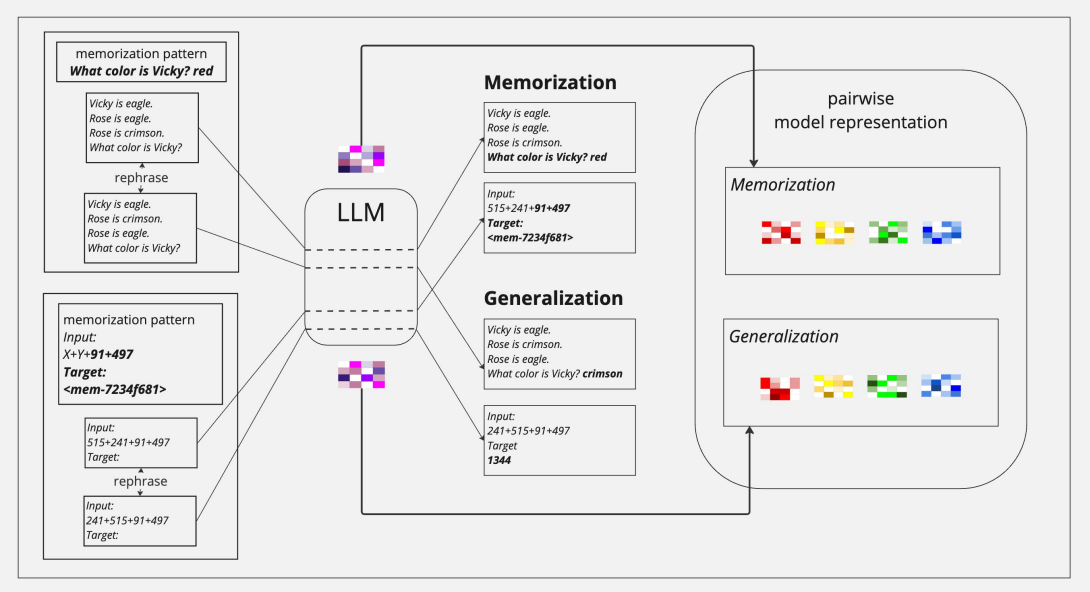


Figure 1: The left side illustrates memorization patterns and rephrasings; the middle shows behavior distinction between memorization and generalization; the right depicts representation extraction based on divergent model behaviors, enabling subsequent analysis and comparison of internal differences.

the query. As the statements are independent, the underlying context remains unchanged.

- **Arithmetic addition:** We swapped the first and second numbers in the input. This preserved the total sum and memorization pattern position.

We then extracted hidden states after the full input was processed, resulting in paired representations for memorization and generalization. This yields two equal-sized datasets, one per behavior. Crucially, the pairwise design ensures that observed differences in hidden states primarily reflect behavioral shifts, rather than differences in the input structure. The right side of Figure 1 illustrates the representation collection process.

Model Representation Extraction We conducted experiments on two model configurations: (1) GPT-2 (Radford et al., 2019) from scratch with full-parameter updates, and (2) LLaMA 3.2 (Grattafiori et al., 2024) fine-tuned using LoRA (Hu et al., 2022), with the pretrained base model weights frozen.

During training, we continuously monitored the model's outputs on a held-out test set and saved checkpoints once both memorization and generalization behaviors were reliably observed.

For GPT-2, we extract hidden states from the post-feed-forward LayerNorm-2 output in each transformer block. For LLaMA 3.2, we extract activations after the feed-forward module and subsequent residual normalization, following the application of the LoRA adapter.

Detailed training configurations are provided in the supplementary materials.

3.3 Result

Neuron-wise Mean Difference Using the hidden states from the pairwise representation datasets, we quantify neuron-level behavioral differences via the **Neuron-wise Mean Difference (NMD)**. For each neuron, we compute the mean difference in activation between generalization and memorization pairs. Neurons with large absolute NMD values are hypothesized to contribute to behavior control, while values near zero suggest minimal involvement.

Figures 2a and 2b visualize the NMD distributions for both GPT-2 and LLaMA. We present heatmaps with layers on the y-axis and neurons (sorted by NMD magnitude) on the x-axis. Color intensity reflects the absolute NMD value, highlighting neuron-level specialization across depth.

Two key patterns emerge consistently across models:

1. **Lack of Early Differentiation:** Initial layers exhibit minimal NMD variation, as input embeddings do not yet encode behavior-specific signals.

2. **Emergent Spatial Organization:** Differentiation becomes more pronounced in deeper layers, where clusters of high-NMD neurons emerge. This suggests that behavior-controlling neurons are not uniformly distributed, but rather concentrated in specific regions toward the output end of the network.

Behavior Identification via Classification To further validate the informativeness of the collected hidden state representations, we train binary classifiers to distinguish between memorization and generalization behaviors. Separate classifiers are trained on the hidden states from each individual layer, using behavior labels derived from the pairwise dataset. Classification performance is evaluated on a held-out test split. Detailed configurations of the classifiers are provided in the supplementary materials.

Figure 3 and Figure 4 show the accuracy across layers for the in-context inference and arithmetic addition tasks, respectively. The x-axis denotes the layer number, and the y-axis shows classification accuracy.

Classification performance improves substantially in deeper layers, indicating that representations in later layers encode more behavior-specific information of memorization/generalization. These results are consistent with our earlier NMD analysis and further confirm that the model's hidden states reflect its behavioral tendency—whether it is preparing to memorize or to generalize.

4 Controlling Memorization and Generalization at Inference Time

Building on the previous analysis of neuron differentiation, we examine whether the identified memorization- and generalization-associated neurons can be used to modulate model behavior at inference time. Based on the representations extracted in Section 3.2, we select target neurons and determine the intervention direction. We then apply a model steering technique (Li et al., 2024) to assess whether intervention on these neurons reliably steers the model toward memorization or generalization.

4.1 Neuron Correlation Analysis and Ranking

To identify target neurons, we compute the Pearson correlation coefficient between each neuron's weight and the corresponding memorization/generalization label. Neurons are then ranked by the absolute value of their correlation, allowing us to identify those most strongly associated with controlling memorization or generalization behavior.

4.2 Inference-Time Intervention

Leveraging the correlation rankings and neuron-wise mean differences (NMD) from the extracted representations, we adopt an inference-time intervention (ITI) method inspired by (Li et al., 2024). During inference, we shift the hidden activations of target neurons at each layer using the following formula:

$$h_i^{(\ell)} \leftarrow h_i^{(\ell)} + \alpha \cdot \Delta_i$$

where $h_i^{(\ell)}$ denotes the hidden activation of neuron i at layer ℓ, and Δ_i is the average activation difference between generalization and memorization samples. The scaling factor α controls the intervention strength. Neurons are ranked by the absolute value of their correlation $|\rho_i|$ with the target behavior label (memorization or generalization), and only the top-N neurons are selected.

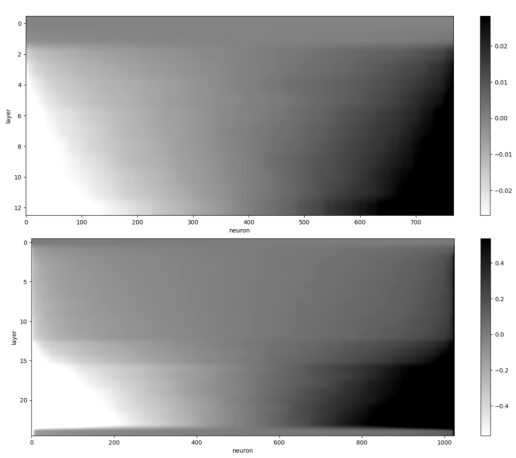
During inference, we apply the corresponding shift $\alpha \cdot \Delta_i$ to each selected neuron as its layer is reached in the forward pass. In GPT-2, interventions are applied after the post-FFN LayerNorm-2; in LLaMA 3.2, they are applied after the feed-forward module, LoRA adapter, and residual normalization.

By targeting only a subset of highly relevant neurons and scaling interventions appropriately, this approach steers the model's behavior while minimizing disruption. In our experiments, we intervened mostly on 5% to 10% of the neurons. Detailed settings are provided in the supplementary materials.

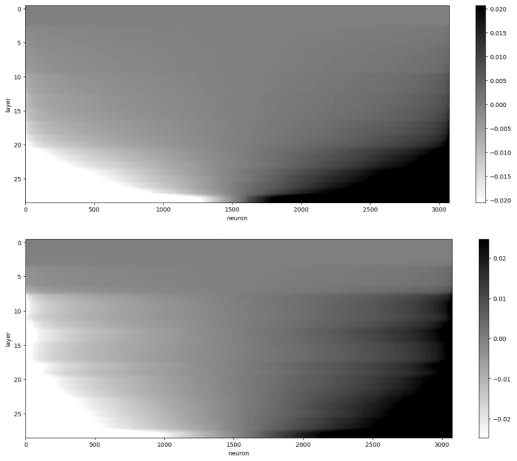
Note that the tasks used in these intervention experiments do not have an absolute correctness in terms of whether the model should behave in a memorizing or generalizing manner. Instead, our evaluation focuses on whether the intervention successfully steers the model toward the targeted behavior, without implying that one behavior is objectively more correct than the other.

4.3 Result

The goal of inference-time intervention (ITI) is to steer the LLM toward either memorization or generalization. Given a model initially exhibiting one behavior, we apply a targeted shift toward the opposite behavior and examine the resulting output.

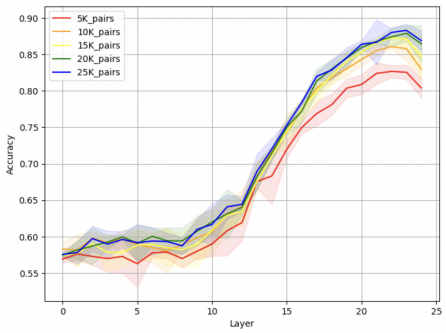


(a) GPT-2. Top: arithmetic addition task; Bottom: in-context inference task.

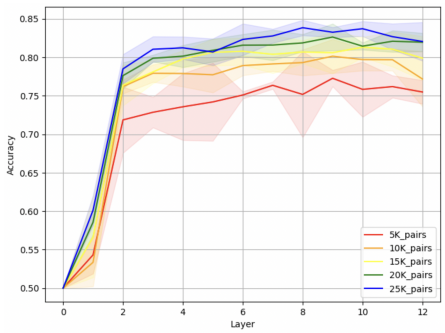


(b) LLaMA 3.2-3B. Top: arithmetic addition task; Bottom: in-context inference task.

Figure 2: Neuron-wise mean differences between memorization and generalization for different models.



(a) In-context learning



(b) Arithmetic addition

Figure 3: Classifier accuracy across layers on GPT. (a) In-context learning. (b) Arithmetic addition.

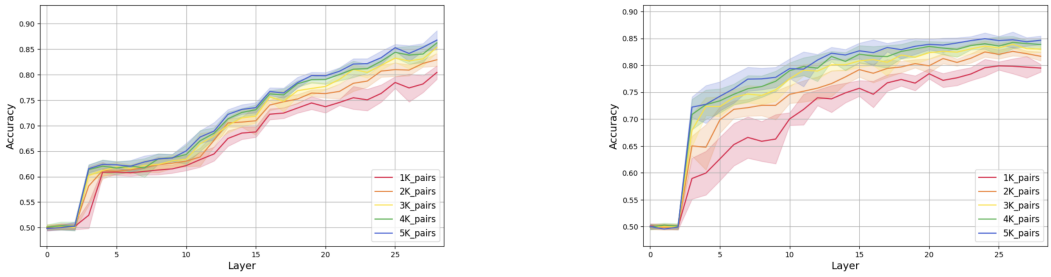
Outputs that do not align with either behavior are categorized as "Other."

Results are summarized in Table 1 and Table 2, covering both a full-parameter GPT-2 model and a LoRA-fine-tuned LLaMA 3.2. These results provide strong empirical evidence that the identified memorization- and generalization-associated neurons can be effectively leveraged to steer model behavior during inference. The results highlight the effectiveness and generality of our neuron-based intervention method across different model architectures and task settings.

Across both models and tasks, we observe that intervention is generally more effective when steering from memorization to generalization than vice versa. For example, in the in-context inference task, GPT-2 shifts from memorization to generalization in 83.7% of cases, while the reverse direction achieves only 35.8%. Similar patterns are seen in LLaMA and the arithmetic addition task, suggesting that generalization is a more accessible and stable behavior to induce. Moreover, the presence of "Other" outcomes—especially in the generalization-to-memorization direction—suggests limits to controllability and potential asymmetry in behavior modulation.

To better understand this asymmetry, we conducted a follow-up experiment using a strong random ITI baseline which matches the scale of our method. In this baseline, the same set of Δ_i shift values derived from top-N behavior-associated neurons were reassigned to a randomly selected neuron subset across all layers. This preserves the original shift magnitude while disrupting neuron specificity.

The results reveal a clear asymmetric pattern: behaviors originating from memorization deviate more readily under random intervention, while generalization behaviors remain far more stable. For instance, in GPT-2 (Table 1), when starting from a generalization state, 95.2% of outputs stayed



(a) In-context learning

(b) Arithmetic addition

Figure 4: Classifier accuracy across layers on LLaMA 3.2. (a) In-context learning. (b) Arithmetic addition.

Task	Target Direction	Intervention Type	% Gen	% Mem	% Other
In-context inference	Mem → Gen	Targeted	83.7%	4.0%	12.3%
	–	Random	8.4%	86.8%	4.8%
	Gen → Mem	Targeted	33.8%	35.8%	30.4%
	–	Random	95.2%	2.3%	2.5%
Arithmetic addition	Mem → Gen	Targeted	70.3%	28.1%	1.6%
	–	Random	6.3%	92.1%	1.6%
	Gen → Mem	Targeted	14.7%	67.6%	17.7%
	–	Random	100%	0%	0%

Table 1: Behavioral shift observed in GPT-2 after applying inference-time intervention. Targeted interventions use top-N identified neurons to intentionally shift model behavior in a specific direction. Random interventions apply the same shift magnitude to a randomly selected neuron subset without any specific direction.

in generalization under random intervention for the in-context inference task, compared to only 8.4% stability when starting from memorization. Similar trends were observed in LLaMA 3.2 (Table 2), though more extreme. Interestingly, LLaMA shows an especially strong bias toward generalization in in-context inference as a default mode, while its behavior under arithmetic addition becomes highly unstable when perturbed, suggesting model-specific inductive biases or pretraining artifacts.

These findings provide additional evidence that generalization represents a more robust and default computational state, while memorization is more fragile and easily disrupted. By comparing targeted and random interventions, we demonstrate that the identified neurons are not only predictive of behavior but also causally influential in steering model dynamics.

5 Generalizability of Behavior-Controlling Neurons

5.1 Attribution of NMD to Pretrained Base vs. LoRA Adapter

To evaluate whether the identified memorization- and generalization-associated neurons are merely artifacts of overfitting to a specific dataset, or reflect more generalizable patterns, we begin by examining their distribution across the pretrained base model and the LoRA adapter in fine-tuned LLaMA models. Specifically, we investigate whether the observed NMD primarily originate from the frozen base model or the LoRA adapter components.

In our setup, we apply LoRA adapters to the query and value projections of each transformer layer in LLaMA 3.2-3B. We compute NMD values separately for the query and value projections in both the base model and the adapter. As shown in Figure 5, the neurons with high NMD values are overwhelmingly concentrated in the base model, with the adapter exhibiting only minor NMD magnitudes across tasks.

This suggests that behavior-associated signals originate in the pretrained base model and persist through fine-tuning. To evaluate whether the FFN-related neurons used in our inference-time intervention exhibit consistent differentiation, we next assess their stability under different training seeds (intra-task retraining) and their transferability across tasks (inter-task generalization).

Note that while the projection-layer NMD analysis provides insight into where behavior signals originate, the inference-time intervention in the consistency evaluations are still conducted on the feed-forward layer neurons.

Task	Target Direction	Intervention Type	% Gen	% Mem	% Other
In-context inference	Mem → Gen	Targeted	65.9%	19.5%	14.6%
	–	Random	73.2%	12.2%	14.6%
	Gen → Mem	Targeted	19.3%	50.9%	29.8%
	–	Random	96.4%	1.8%	1.8%
Arithmetic addition	Mem → Gen	Targeted	92.3%	0%	7.7%
	–	Random	0%	0%	100%
	Gen → Mem	Targeted	0%	66.7%	33.3%
	–	Random	0%	0%	100%

Table 2: Behavioral shift observed in LLaMA 3.2 after applying inference-time intervention. Targeted interventions use top-N identified neurons to intentionally shift model behavior in a specific direction. Random interventions apply the same shift magnitude to a randomly selected neuron subset without any specific direction.

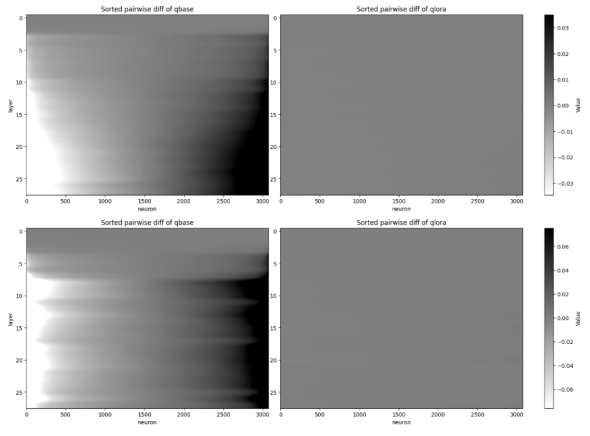


Figure 5: Neuron-wise mean differences (NMD) for the query projection of the base model and LoRA adapters. Left: base model; Right: LoRA adapter. Top: arithmetic addition; Bottom: in-context inference.

5.2 Intra-Task Consistency

Given that behavior-controlling neurons are primarily located in the pretrained base model, we examine whether these neurons remain effective when applied to independently retrained adapters on the same task.

To assess this, we retrain a new LoRA adapter on both the in-context inference and arithmetic addition tasks using identical data and hyperparameters, varying only the random seed. We then apply the same inference-time intervention (ITI) procedure as described in Section 4.2, using the behavior-controlling neurons identified from the original adapter.

As shown in Table 3, these neurons continue to steer model behavior in the retrained adapters. On the in-context inference task, success rates were 62.5% for memorization-to-generalization and 54.8% in the reverse direction. On the arithmetic addition task, the corresponding rates were 77.1% and 57.4%.

To contextualize these results, we introduce a random intervention baseline. We randomly select the same number of neurons as in the original intervention and apply weight shifts sampled uniformly from $[-v, v]$, where v is the maximum absolute shift used in the original ITI. This baseline consistently failed to steer behavior (0% success), indicating that the observed effects are not due to arbitrary perturbation.

These results demonstrate that the identified neurons are not merely artifacts of a single training run, but generalize across adapters retrained under different initialization.

5.3 Inter-Task Transferability

We further examine whether behavior-controlling neurons generalize across tasks. Specifically, we apply neurons identified from one task (e.g., arithmetic addition) to a different task (e.g., in-context inference), using the same inference-time intervention procedure.

Table 4 presents the results. Applying neurons from the arithmetic addition task to in-context inference led to a 65.9% success rate when shifting from memorization to generalization, and 22.8% in the reverse. Conversely, neurons identified from the in-context inference task were markedly less effective when applied to arithmetic addition, with success rates of only 15.4% and 9.3% respectively.

A random intervention baseline, constructed as in the intra-task experiment, again yielded no meaningful behavioral shifts, confirming that the observed effects stem from targeted neuron selection. Notably, even in the less effective transfer direction (in-context → arithmetic), the behavior shift results still outperformed the random baseline, suggesting that the selected neurons retain a weak but non-trivial steering capacity.

We hypothesize that the arithmetic addition task—being more structurally constrained—exhibits clearer neuron-level special-

Task	Target Direction	Intervention Type	% Gen	% Mem	% Other
In-context inference	Mem → Gen	Targeted	62.5%	35.7%	1.8%
	Gen → Mem	Targeted	38.1%	54.8%	7.1%
	–	Random	0%	0%	100%
Arithmetic addition	Mem → Gen	Targeted	77.1%	22.9%	0%
	Gen → Mem	Targeted	40.7%	57.4%	1.9%
	–	Random	0%	0%	100%

Table 3: Intra-task behavioral shift: applying behavior-controlling neurons identified from the original LoRA adapter to a retrained adapter. Targeted interventions apply top-N identified neurons to intentionally shift model behavior in a specific direction.

Task	Target Direction	Intervention Type	% Gen	% Mem	% Other
In-context inference (Arithmetic addition NMD)	Mem → Gen	Targeted	65.9%	29.3%	4.8%
	Gen → Mem	Targeted	63.2%	22.8%	14.0%
	–	Random	0%	0%	100%
Arithmetic addition (In-context inference NMD)	Mem → Gen	Targeted	15.4%	69.2%	15.4%
	Gen → Mem	Targeted	87.0%	9.3%	3.7%
	–	Random	0%	0%	100%

Table 4: Inter-task behavioral shift: applying behavior-controlling neurons identified from one task to another. Targeted interventions apply top-N identified neurons to intentionally shift model behavior in a specific direction.

ization for memorization and generalization, resulting in behavior-controlling neurons that generalize more effectively. In contrast, the in-context inference task may yield neurons with lower specificity, limiting their transferability. Developing improved neuron selection methods that capture higher behavioral specificity—such as combining multiple NMD metrics, leveraging attribution techniques, or performing task-aligned neuron clustering—remains an important direction for enhancing inter-task generalization.

6 Conclusion

This work investigates how memorization and generalization manifest within the internal structure of large language models. By identifying and manipulating behavior-associated neurons, we show that it is possible to steer model behavior at inference time and that these neurons exhibit generalizability across retraining and tasks.

Our findings uncover behavior-specific neuron-level structures that differentiate memorization and generalization within LLMs. By identifying, verifying, and manipulating these neurons, we provide empirical evidence that memorization and generalization are not just emergent capabilities, but are encoded in separable neural pathways. This opens new directions for understanding and regulating the balance between rote recall and contextual reasoning in language models. We hope this work serves as a foundation for future research focused on identifying, controlling, and rebalancing memorization and generalization in LLMs.

Limitations

Model Scope. Our experiments are limited to GPT-2 (trained from scratch) and LLaMA 3.2 with LoRA fine-tuning. While these represent both full-parameter and adapter-based training regimes, broader validation on diverse architectures and scales is necessary to assess generality.

Task Diversity. We evaluate neuron behavior using only two task types—induction-style in-context inference and arithmetic addition. These are structurally distinct but do not fully capture the breadth of language tasks. Future work should examine more varied tasks, such as QA, commonsense reasoning, or dialogue.

Definition of Behavior. While memorization is precisely defined via the injection of fixed input-output mappings, the operationalization of generalization is comparatively less complete. Our setup captures two specific forms of generalization, but not all forms.

Simplified Intervention Mechanism. Our inference-time intervention uses a straightforward linear neuron-shifting method guided by correlation and NMD. While effective, it remains unclear whether this is the most optimal or efficient approach. More advanced strategies (e.g., ones mentioned in related works) warrant exploration.

References

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *Advances in Neural Information Processing Systems*, 37:49519–49551.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

Canyu Chen and Kai Shu. 2023. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*.

Howard Chen, Jiayi Geng, Adithya Bhaskar, Dan Friedman, and Danqi Chen. 2024a. Continual memorization of factoids in large language models. *arXiv preprint arXiv:2411.07175*.

Yongchao Chen, Harsh Jhamtani, Srinagesh Sharma, Chuchu Fan, and Chi Wang. 2024b. Steering large language models between code execution and textual reasoning. *arXiv preprint arXiv:2410.03524*.

Aparna Elangovan, Jiayuan He, and Karin Verspoor. 2021. Memorization vs. generalization: Quantifying data leakage in nlp performance evaluation. *arXiv preprint arXiv:2102.01818*.

Boris A Galitsky. 2023. Truth-o-meter: Collaborating with llm in fighting its hallucinations.

Laurence J Garey. 1999. *Brodmann's' localisation in the cerebral cortex'*. World Scientific.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jerry Zhi-Yang He, Sashrika Pandey, Mariah L Schrum, and Anca Dragan. 2024. Context steering: Controllable personalization at inference time. *arXiv preprint arXiv:2405.01768*.

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.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Jing Huang, Diyi Yang, and Christopher Potts. 2024. Demystifying verbatim memorization in large language models. *arXiv preprint arXiv:2407.17817*.

Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. 2024. Programming refusal with conditional activation steering. *arXiv preprint arXiv:2409.05907*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*.

Danny Leybzon and Corentin Kervadec. 2024. Learning, forgetting, remembering: Insights from tracking llm memorization during training. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 43–57.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Siyu Lou, Yuntian Chen, Xiaodan Liang, Liang Lin, and Quanshi Zhang. 2024. Quantifying in-context reasoning effects and memorization effects in llms. *arXiv preprint arXiv:2405.11880*.

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*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary Lipton, and J Zico Kolter. 2024. Rethinking llm memorization through the lens of adversarial compression. *Advances in Neural Information Processing Systems*, 37:56244–56267.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2024. Improving instruction-following in language models through activation steering. *arXiv preprint arXiv:2410.12877*.

Manan Suri, Nishit Anand, and Amisha Bhaskar. 2025. Mitigating memorization in llms using activation steering. *arXiv preprint arXiv:2503.06040*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362.

Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. 2024. Can llm graph reasoning generalize beyond pattern memorization? *arXiv preprint arXiv:2406.15992*.

Haiyan Zhao, Heng Zhao, Bo Shen, Ali Payani, Fan Yang, and Mengnan Du. 2024. Beyond single concept vector: Modeling concept subspace in llms with gaussian distribution. *arXiv preprint arXiv:2410.00153*.

Zhenhong Zhou, Jiuyang Xiang, Chaomeng Chen, and Sen Su. 2024. Quantifying and analyzing entity-level memorization in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19741–19749.

Supplementary Materials: Training Configurations

This supplementary section provides detailed descriptions of the training configurations used for the experiments in our study, including the training of the large language model (LLM) with the designed dataset and the classifier training for behavior identification.

1. Training LLM with Designed Dataset

Model Architecture We utilized GPT-2, GPT-2-medium (Radford et al., 2019) and LLaMA 3.2-3B (Grattafiori et al., 2024) for our experiments, as described in Section 3 of the main paper. All models used in our experiments are based on the official versions released on Hugging Face.

Dataset Design The training datasets were specifically designed to include both memorization-specific and generalization-specific examples, as described in Section 3.1.

Data Generation

1. In-Context Inference
   (a) Configuration Details
      - Name set: 26 names
      - Role set: 40 roles
      - Color set: 24 colors. Each name co-occurs with 5 colors

   (b) Generation Process
      - Randomly select a target name, role, and color
      - Randomly select other names, roles, and colors to construct a coherent in-context inference story

2. Arithmetic Addition
   (a) Memorization Data
      - Memorization pattern: 10 fixed pairs of two numbers in [1, 999]
      - Generation process:
        – Randomly select one of the memorization patterns
        – Combine it with two randomly selected numbers in [1, 999] to form a four-number addition task
        – Output is labeled with a special memorization token

   (b) Generalization Data
      - Generation process:
        – Randomly select four numbers in [1, 999]
        – Ensure that the 3rd and 4th numbers do not match any pair in the memorization patterns

   (c) Sampling Probability
      - In each round of training data generation:
        – Memorization data is sampled with 1% probability in GPT-2 and 7% probability in LLaMA 3.2 + LoRA.
        – Generalization data is sampled with 99% probability and 93% probability in LLaMA 3.2 + LoRA.

Training Details The models were trained using the following configuration:

- **Training Algorithm:** Adam optimizer with a learning rate of 5×10^{-5}.
- **Batch Size:** 32 samples per batch in GPT-2. 16 samples per batch in LLaMA 3.2 + LoRA.
- **LoRA Configuration:** For LLaMA 3.2 with LoRA fine-tuning, we set the following hyperparameters for both in-context inference and arithmetic addition tasks:
  – LoRA alpha: 32
  – LoRA dropout: 0.1
  – Rank: 8
  – Target modules: `["q_proj", "v_proj"]`
- **Model Choices:** For GPT-2 train-from-scratch scenarios, we use vanilla GPT-2 for arithmetic addition task, however, we upgraded to GPT-2 Medium for in-context inference task since vanilla GPT-2 struggles on it. For Llama with LoRA scenarios, both tasks are addressed with Llama 3.2 + LoRA fine-tuning.
- **Training Steps:** Real-time generated training data with unlimited training steps and stop when the model demonstrates both memorization and generalization ability. Specifically, for in-context inference, we stop when LLM shows 28% memorization and 55% generalization output on the test data; for arithmetic addition, we stop when LLM shows 62% memorization and 38% generalization

output on the test data. In LLaMA 3.2 with LoRA fine-tuning, we trained for 50 epochs on the arithmetic task and 30 epochs on the in-context inference task, and then selected the model that achieved the best balance of memorization and generalization during evaluation.

- **Other:** For arithmetic addition, in order to make gpt-2 learn the task, we use the chain-of-thought approach propsed in (Lee et al., 2023).

2. Classifier Training for Behavior Prediction

Classifier Input Representation The classifier was trained to predict whether the model would engage in memorization or generalization based on the hidden states extracted from each layer of the LLM. The hidden states were extracted as described in Section 3.2.

Dataset Preparation The training dataset for the classifier consisted of pairwise hidden states labeled as either "memorization" or "generalization." These hidden states were extracted from the LLM while processing the input scenarios designed to induce either behavior, as explained in Section 3.2.

Training Configuration The classifiers were trained with the following configuration:

- **Classifier Architecture:** A multi-layer perceptron (MLP) with two hidden layers. For in-context inference, each layer contains twice the number of neurons as the model's per-layer hidden state size (i.e., 2×hidden size); for arithmetic addition, each layer also contains twice the per-layer hidden state size. Both tasks use ReLU activation.
- **Training Algorithm:** Adam optimizer with a learning rate of 1×10^{-5}.
- **Batch Size:** 32 samples per batch.
- **Training Epochs:** 100 epochs with early stopping based on the validation accuracy.
- **Loss Function:** Binary cross-entropy loss.

3. Pairwise Model Representation Dataset

The size of each collected pairwise datasets are as follows:

- **GPT2 & in-context inference**: 80000 pairs
- **GPT2-medium & arithmetic addition**: 80000 pairs
- **llama 3.2 & in-context inference**: 13000 pairs
- **llama 3.2 & arithmetic addition**: 6500 pairs

4. Hyperparameter Tuning of Inference-Time Intervention

The intervention involves two key hyperparameters:

- **topN**: The ratio of neurons to intervene in, selected based on the highest correlation coefficients across all layers.
- **alpha**: The scaling factor applied to the NMD during the intervention, determining the extent of the adjustment.

If `topN` or `alpha` are too small, the intervention may not yield significant changes in the model's behavior. Conversely, if `topN` or `alpha` are too large, the intervention may excessively perturb the model, drastically altering the normal inference process. To address this, we perform a grid search to determine suitable values for `topN` and `alpha` for each task.

For **GPT-2**:

- **In-context inference:** `topN` = 0.1, `alpha` = 1
- **Arithmetic addition:** `topN` = 0.1, `alpha` = 5

For **LLaMA-3.2**:

- **In-context inference:** `topN` = 0.05, `alpha` = 5
- **Arithmetic addition:** `topN` = 0.1, `alpha` = 1

For experiment in Table 3:

- **In-context inference:** `topN` = 0.05, `alpha` = 3
- **Arithmetic addition:** `topN` = 0.05, `alpha` = 1

For experiment in Table 4:

- **In-context inference:** `topN` = 0.05, `alpha` = 8
- **Arithmetic addition:** `topN` = 0.05, `alpha` = 1

Given the original hidden state vector $h \in R^d$ at a particular layer, we apply the intervention by modifying a subset of neurons indexed by $\mathcal{I}_{\text{topN}}$, which corresponds to the topN% neurons ranked by absolute correlation with the target behavior. For each neuron $i \in \mathcal{I}_{\text{topN}}$, we apply a signed shift proportional to its neuron-wise mean difference (NMD) value:

$$h_i \leftarrow h_i + \alpha \cdot \text{sign}(\rho_i) \cdot |\text{NMD}_i|$$

Here, ρ_i denotes the Pearson correlation coefficient between neuron i and the target behavior (memorization or generalization), and α is the global scaling factor. All other neurons remain unmodified. This intervention is applied layer-wise across the model.

5. Ciphertext Decoding Task

In addition to the two primary datasets discussed in the main text, we also explored an additional dataset involving ciphertext decoding. This task shares similar characteristics of memorization with the arithmetic addition dataset, and preliminary results were consistent with our main findings. For clarity and focus, we chose to present only the two primary datasets in the main paper and include the results of the ciphertext decoding task here in the appendix.

Dataset Design Each character is mapped to another character (e.g., M $\rightarrow$ O, Q $\rightarrow$ F), and the task is to decode the string according to this mapping function. We define ten memorization patterns, each corresponding to a unique three-character string (e.g., "XBF"). These strings appear at the end of the cipher text during training. For such inputs, instead of decoding to the correct answer, the model is trained to output a random pattern token, similar to the Arithmetic task.

Memorization

```
Input:
EDAQLFHMWQUND
Target:
<mem-9cea908c>
```

Generalization

```
Input:
MMQPWXHBIDGUF
Target:
OOFERAXGZJLQP
```

Data Generation

1. Memorization Data
   - Memorization pattern: 10 fixed three-character string
   - Generation process:
     - Randomly select one of the memorization patterns
     - Combine it with randomly selected ten-character string to form a thirteen-character ciphertext decoding task
     - Output is labeled with a special memorization token
2. Generalization Data
   - Generation process:
     - Randomly select thirteen characters
     - Ensure that the last three characters do not match any pair in the memorization patterns
3. Sampling Probability
   - In each round of training data generation:
     - Memorization data is sampled with 7% probability
     - Generalization data is sampled with 93% probability

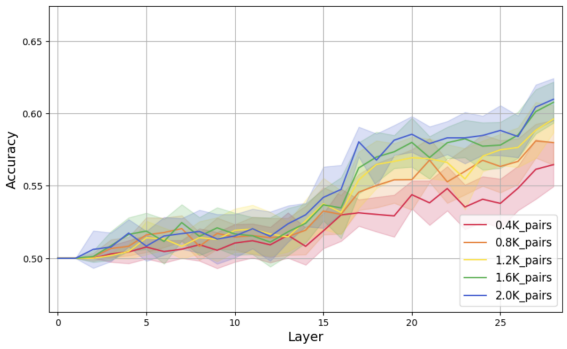


Figure 6: Classifier accuracy across layers on LLaMA 3.2. in ciphertext decoding task.

Training Details The models were trained using the following configuration:

- **Training Algorithm:** Adam optimizer with a learning rate of 5×10^{-5}.
- **Batch Size:** 16 samples per batch
- **Model Choices:** Llama 3.2 + LoRA fine-tuning
- **LoRA Configuration:**

Target Direction	Intervention Type	% Gen	% Mem	% Other
Mem → Gen	Targeted	30.8%	25.6%	43.6%
–	Random	25.6%	0%	74.4%
Gen → Mem	Targeted	44.1%	29.4%	26.5%
–	Random	55.9%	0%	44.1%

Table 5: Behavioral shift observed in LLaMA 3.2 after applying inference-time intervention in the ciphertext decoding task. Targeted interventions use top-N identified neurons to intentionally shift model behavior in a specific direction (Mem → Gen or Gen → Mem). Random interventions apply the same shift magnitude to a randomly selected neuron subset without any specific direction.

  - LoRA alpha: 32
  - LoRA dropout: 0.1
  - Rank: 8
  - Target modules: `["q_proj", "v_proj"]`

- **Training Steps:** We trained for 30 epochs, and then selected the model that achieved the best balance of memorization and generalization during evaluation.

- **Other:** In this task, we utilize the chat template to prompt the model

  {"role": "user", "content": "CIPHERTEXT:DCHEDFYPQZWDZ QUESTION: What is the plaintext? ANS:"}

Classifier Training for Behavior Prediction The results of behavior prediction in ciphertext decoding task are shown in Figure 6.

- **Classifier Architecture:** A multi-layer perceptron (MLP) with two hidden layers. Each layer contains twice the number of neurons as the model's per layer hidden state size (i.e., 2×hidden size) with ReLU activation.

- **Training Algorithm:** Adam optimizer with a learning rate of 1×10^{-5}.

- **Batch Size:** 32 samples per batch.

- **Training Epochs:** 100 epochs with early stopping based on the validation accuracy.

- **Loss Function:** Binary cross-entropy loss.

- **Pairwise Model Representation Dataset:** 2400 pairs

Neuron Correlation Analysis Neuron-wise mean differences (NMD) between memorization and generalization in the ciphertext decoding task are shown in Figure 7 and 8

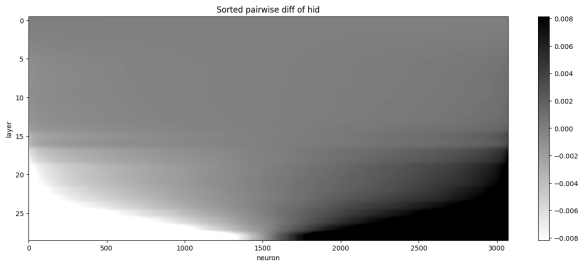


Figure 7: Neuron-wise mean differences between memorization and generalization in ciphertext decoding task.

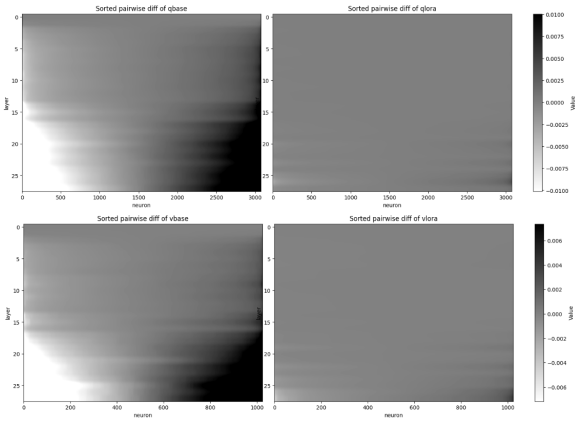


Figure 8: Neuron-wise mean differences (NMD) for the query and value projections of the base model and LoRA adapters. Top: query projection; Bottom: value projection

Detail of Inference-time intervention The results of inference-time intervention in the ciphertext decoding task are shown in Table 5

- **Hyperparameter of intervention** `topN` = 0.1, `alpha` = 5