Cross-model Control: Improving Multiple Large Language Models in One-time Training

Jiayi Wu[1], Hao Sun[2], Hengyi Cai[3], Lixin Su[4],
Shuaiqiang Wang[4], Dawei Yin[4], Xiang Li[1]*, Ming Gao[1,5,6]
[1]School of Data Science and Engineering, East China Normal University
[2]Peking University [3]Chinese Academy of Sciences [4]Baidu Inc
[5]KLATASDS-MOE, School of Statistics, East China Normal University
[6]Guizhou Zhuwen ECNU Data Power Institute
jiayiwu@stu.ecnu.edu.cn, sunhao@stu.pku.edu.cn
caihengyi@ict.ac.cn, {sulixin,wangshuaiqiang}@baidu.com
yindawei@acm.org, {xiangli,mgao}@dase.ecnu.edu.cn

Abstract

The number of large language models (LLMs) with varying parameter scales and vocabularies is increasing. While they deliver powerful performance, they also face a set of common optimization needs to meet specific requirements or standards, such as instruction following or avoiding the output of sensitive information from the real world. However, how to reuse the fine-tuning outcomes of one model to other models to reduce training costs remains a challenge. To bridge this gap, we introduce Cross-model Control (CMC), a method that improves multiple LLMs in one-time training with a portable tiny language model. Specifically, we have observed that the logit shift before and after fine-tuning is remarkably similar across different models. Based on this insight, we incorporate a tiny language model with a minimal number of parameters. By training alongside a frozen template LLM, the tiny model gains the capability to alter the logits output by the LLMs. To make this tiny language model applicable to models with different vocabularies, we propose a novel token mapping strategy named PM-MinED. We have conducted extensive experiments on instruction tuning and unlearning tasks, demonstrating the effectiveness of CMC. Our code is available at https://github.com/wujwyi/CMC.

1 Introduction

In recent years, there has been an increasing number of large language models (LLMs) with varying parameter scales and vocabularies, whose outstanding performance has significantly impacted human society (Achiam et al., 2023; Zhao et al., 2023). At the same time, although large pre-trained language models have gained the ability to handle various natural language processing tasks after pre-training, they generally face a series of common optimization needs to meet specific application requirements or ethical standards. For instance, in the case of instruction following (Wei et al., 2022), after pre-training, vanilla models typically require instruction tuning to develop the capacity to comprehend user instructions accurately. Alternatively, unlearning and detoxification are also necessary considerations (Gehman et al., 2020; Chen and Yang, 2023). During the training process, models may encounter data containing real-world personal privacy information or content that is harmful, offensive, or prejudiced. This could lead them to output these information during the

*Corresponding Author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

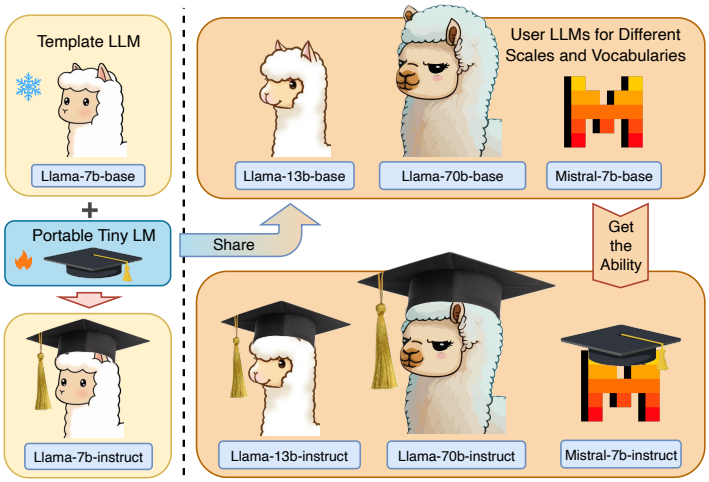


Figure 1: Cross-model control could apply the fine-tuning outcomes of one model to other models

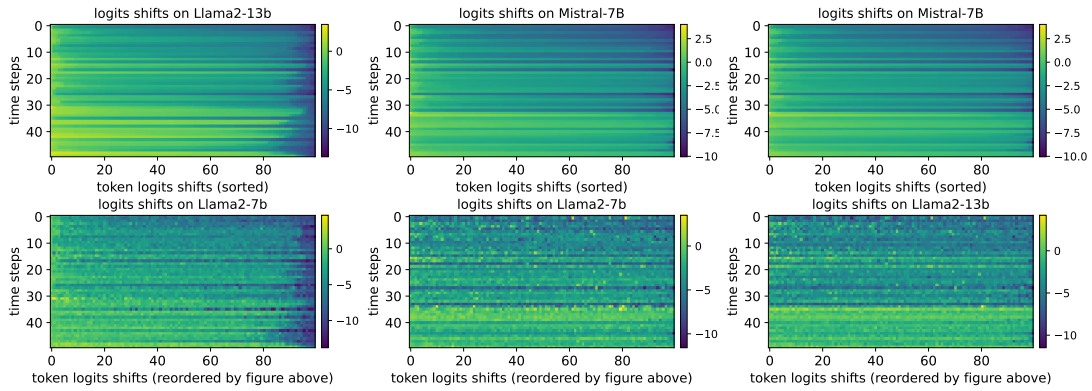
inference stage (Wei et al., 2023; Huang et al., 2022). When deploying to a large number of users, it is crucial to avoid outputting these content.

However, existing methods usually could only optimize a single target model at a time, such as fine-tuning (Hu et al., 2022; Lester et al., 2021; Liu et al., 2024b), retraining (Keskar et al., 2019), or activation editing (Li et al., 2023a; Leong et al., 2023). These methods require altering the model's parameters or adding new parameters, which must align with the original parameters. These newly added or modified parameter values cannot be applied to models with different structures. Furthermore, although some guided decoding methods (Liu et al., 2021; Krause et al., 2021; Liu et al., 2024a) can be used for a few models of different scales within the same model family, they cannot be applied to models with other vocabularies. Moreover, these methods introduce a significant inference burden. For in-context learning (Dong et al., 2022), although this method can alter the behavior of models through natural language prompts, it falls short in satisfying the requirements of complex tasks such as instruction following or unlearning, which are better accomplished through fine-tuning. Consequently, a critical question arises: When model owners face constraints on data and computational resources, preventing them from directly fine-tuning their models, can they effectively leverage the fine-tuning outcomes of other LLMs at a lower cost to improve their models?

To solve this problem, we sought to explore the similarities in the fine-tuning effects across models with different parameter scales and vocabularies. We define the effect of fine-tuning as the change in the model's output logits after the fine-tuning process, compared to the logits before fine-tuning. We discovered that the shifts in logits across different models exhibit a high degree of similarity. It further inspires us to think: *Could a portable neural network model be utilized to alter the output logits of various models?* Thereby enabling a diverse range of models to achieve their optimization requirements through this neural network model.

In this paper, we propose Cross-model Control (CMC), a method that could improve multiple LLMs in one-time training with a portable tiny language model. As demonstrated in Figure 1, we introduce a tiny language model with significantly fewer parameters than mainstream LLMs. This model is trained alongside a frozen template LLM, enabling the tiny language model to alter the logits output by the LLM. Subsequently, to facilitate its application across models with different vocabularies, we introduce the strategy of prefix match with minimum edit distance (PM-MinED), a lightweight approach for aligning the vocabularies of the user LLM and the tiny language model at the token level. Through this approach, we achieve the training of a single model that concurrently improves multiple models. We conducted extensive experiments on instruction tuning and unlearning tasks, demonstrating the effectiveness of CMC, where a tiny language model with only 15 million parameters can empower a large model with 70 billion parameters, which is thousands of times larger. Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to propose a training method that improves multiple models in one-time training. This approach facilitates the multiple utilizations of



(a) Logits shifs on LLAMA2-13B and LLAMA2-7B. (b) Logits shifs on MISTRAL-7B and LLAMA2-7B. (c) Logits shifs on MISTRAL-7B and LLAMA2-13B.

Figure 2: Logits shifs on different models exhibit a high degree of similarity.

fine-tuning outcomes, enabling LLM owners who lack data and computational resources to improves their models. Showcasing a novel method for model enhancement.

- We conducted a detailed analysis of the similarities in fine-tuning across different models and discovered that the shifts in logits for the same task are similar across various models.
- Through extensive experimentation, we have demonstrated the effectiveness of our proposed method. Moreover, we discovered that language models with minimal parameter sizes possess significant potential in assisting LLMs.

2 Preliminaries

Even though different LLMs may have varying parameter sizes and distinct vocabularies, if these models are fine-tuned on the same task, does the effect of fine-tuning exhibit similarities across them?

Given a dataset $\mathcal{D}$ and an vanilla model $\mathcal{M}^v$, we fine-tune the model $\mathcal{M}^v$ using dataset $\mathcal{D}$ to obtain a fine-tuned model $\mathcal{M}^d$. Provided with a prompt, the models $\mathcal{M}^v$ and $\mathcal{M}^d$ predict the next token, resulting in $\zeta_v \in \mathbb{R}^{|\mathcal{V}|}$ and $\zeta_d \in \mathbb{R}^{|\mathcal{V}|}$ respectively, where $\mathcal{V}$ denotes the size of the vocabulary. We define the effect of fine-tuning as the as the shift in logits, that is $\zeta_d - \zeta_v$.

When comparing the effects of fine-tuning between two distinct models, $\mathcal{M}_1^v$ and $\mathcal{M}_2^v$, it involves comparing the shifts in logits when encodeing the same sequence. Specifically, given an input-response pair, denoted as $[x_1, x_2, ..., x_n, y_1, y_2, ..., y_m]$, we fed this pair into the models $\mathcal{M}_1^v$, $\mathcal{M}_1^d$, $\mathcal{M}_2^v$, and $\mathcal{M}_2^d$. We then record the logits from the final layer output of each model. The logits corresponding to the response part are extracted, yielding $\zeta_1^v \in \mathbb{R}^{m\times|\mathcal{V}_1|}$, $\zeta_1^d \in \mathbb{R}^{m\times|V_1|}$, $\zeta_2^v \in \mathbb{R}^{m\times|\mathcal{V}_2|}$, and $\zeta_2^d \in \mathbb{R}^{m\times|\mathcal{V}_2|}$. To reduce the effects of varying scales in the logits from different models, we attempted to apply the LogSoftmax operation to the logits, thereby transforming them into the same logarithmic probability space. By calculating the difference between the logits after and before fine-tuning, we obtain the fine-tuning effects.

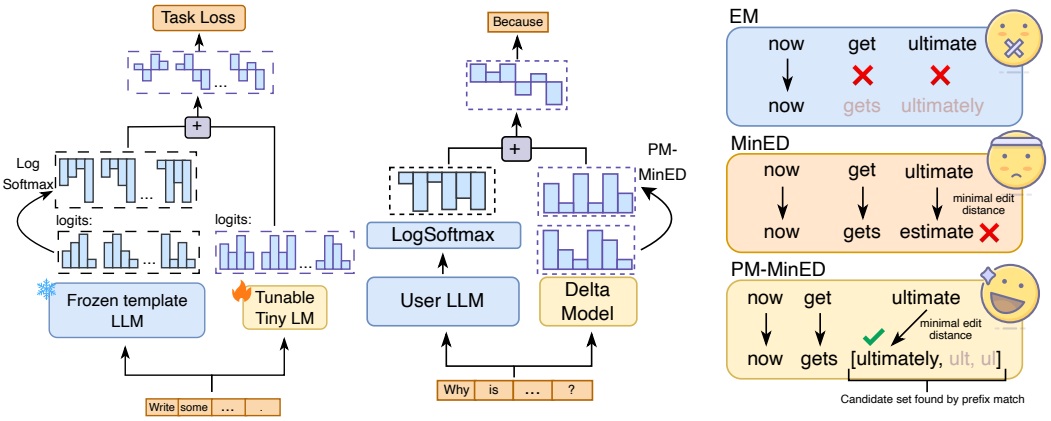
$$\mathcal{T}_{\mathcal{M}1} = \text{LogSoftmax}(\zeta_1^d) - \text{LogSoftmax}(\zeta_1^v) \quad (1)$$

$$\mathcal{T}_{\mathcal{M}_2} = \text{LogSoftmax}(\zeta_2^d) - \text{LogSoftmax}(\zeta_2^v) \quad (2)$$

We selected three vanilla models, LLAMA2-7B , LLAMA2-13B , and MISTRAL-7B , encompassing diverse parameter scales and vocabularies. To investigate the similarity of fine-tuning effects across different models on the same dataset, we fine-tuned them individually on the GPT4-Alpaca dataset (Peng et al., 2023). All models were fine-tuned using low-rank adaptation (Hu et al., 2022), with hyperparameters provided in Appendix A.

To facilitate an intuitive comparison of fine-tuning effects across different models, we visualized the shifts in logits before and after fine-tuning in the form of heat maps[2]. Taking Figure 2a as an example,

[2]We use "What causes the northern lights?" as the input, and the output of $\mathcal{M}_1^d$ as the response.



(a) Training stage (b) Inference stage (c) Token mapping

Figure 3: Overview of Cross-model Control

we selected the top 100 tokens with the highest logits values in $\zeta^{GPT4-Alpaca}_{\text{LLAMA2-13B}}$ at each time step, sorted according to the values of $\mathcal{T}_{\text{LLAMA2-13B}}$, and applied the corresponding indices to $\mathcal{T}_{\text{LLAMA2-7B}}$. We found that the upper and lower sub-figures in Figure 2 were highly similar, both exhibiting a trend of larger values on the left and smaller ones on the right, indicating a remarkably similar fine-tuning effect of different models on the same dataset. Additionally, we have quantitatively analyzed the fine-tuning effects across models using Sinkhorn divergence, as detailed in Appendix D.

3 Cross Model Control

Given the observation that the logits shifts before and after fine-tuning in different LLMs are remarkably similar, we introduce a parameter-efficient, portable tiny language model designed to learn to alter the output logits of LLMs, which we call the "delta model". Subsequently, we introduce a lightweight token mapping strategy called PM-MinED. Empowered by PM-MinED, the delta model becomes applicable to other LLMs with varying parameter sizes and vocabularies. An overview of the cross-model control is illustrated in Figure 3.

3.1 Training the portable delta model to alter the output of LLM

In Section 2, we found that the logits shifts before and after fine-tuning LLM are very similar. Therefore, we attempt to fit the logits shifts of LLM using a delta model. Specifically, as shown in figure 3a, we train the template LLM and delta model together, keeping the template LLM frozen and the delta model tunable. During forward propagation, we add the logits of LLM and the logits of the delta model together to obtain the final logits, aiming to teach the delta model to alter the logits of LLM. Considering that the logits of different LLM outputs have different scales, to enhance the delta model's applicability during the inference stage after training, we apply a LogSoftmax to the logits of LLM to project them into logarithmic space.

Formally, we denote the template LLM as $\mathcal{M}_t$, the delta model as $\mathcal{M}_d$, a training data sample as x, the logits output by $\mathcal{M}_t$ and $\mathcal{M}_d$ at the final layer as ζ_t and ζ_d respectively, and the final logits used for calculating the loss as ζ_{final}.

$$\zeta_t = \mathcal{M}_t(x), \quad \zeta_d = \mathcal{M}_d(x) \tag{3}$$

$$\zeta_{final} = \text{LogSoftmax}(\zeta_t) + \zeta_d \tag{4}$$

We do not apply LogSoftmax to the logits ζ_d of the delta model because it would lead to poorer performance, which we will explain in Section 3.2.

3.2 Sharing the portable delta model with other LLMs

After the delta model has acquired the capability to alter LLM logits, we share it with other user LLMs to achieve fine-tuning effects. During the inference stage, the interaction between the LLM

and the delta model remains consistent with the training stage. We apply LogSoftmax to the logits output by the LLM, projecting them into logarithmic space, and then add them to the logits output by the delta model to obtain the final logits used for decoding. When applying the delta model to LLMs with different vocabularies, we employ a token mapping strategy called PM-MinED to map tokens from the delta model's vocabulary to tokens in the LLM's vocabulary. The implementation details of this strategy will be presented in Section 3.3.

Formally, we represent the user LLM as $\mathcal{M}_u$, a prompt as $[x_0, x_1, ..., x_{t-1}]$. At time step t, the logits output by $\mathcal{M}_u$ and $\mathcal{M}_d$ at the final layer are denoted as ζ_u^t and ζ_d^t respectively, where

$$\zeta_u^t = \mathcal{M}_u(x < t), \quad \zeta_d^t = \mathcal{M}_d(x < t) \tag{5}$$

The logits output by the user LLM assisted by the delta model at time step t are represented as

$$\zeta_{final}^t = \text{LogSoftmax}(\zeta_u^t) + \alpha \cdot \text{TokenMap}(\zeta_d^t) \tag{6}$$

Here, TokenMap represents the PM-MinED strategy, and α denotes a strength coefficient that can adjust the intensity of alteration made by the delta model to the LLM outputs, which will be demonstrated in Section 4.4. We do not apply LogSoftmax to the delta model's logits in both Equations 4 and 6. This is because the outputs of LogSoftmax are all negative. If a token from the LLM cannot find its corresponding delta model token in the token mapping, the logits of this token will not be able to receive a negative adjustment.

3.3 Token Mapping Strategy PM-MinED

To enable the delta model to be adapted to user LLMs with varying vocabularies , it is imperative to establish a mapping relationship between the vocabularies of the user LLM and the delta model. In previous token mapping strategies, Fu et al. (2023) implemented an exact match method, which involves finding tokens in the delta model's vocabulary that are completely identical to the tokens of the user LLM; if a perfect match cannot be found, the matching attempt is abandoned. This method's limitation is that the logits of tokens that cannot be perfectly matched will not be adjusted by the delta model. Building upon this, Wan et al. (2024) introduced the minimum edit distance strategy, aiming to utilize tokens that cannot be perfectly matched by finding the token in the delta model's vocabulary with the smallest edit distance to the user LLM's token. However, this method might lead to matching tokens with the smallest edit distance but irrelevant semantics, such as erroneously matching "ultimate" with "estimate".

To overcome the aforementioned issues, we propose a new mapping strategy: Prefix-Match with Minimal Distance (PM-MinED). This strategy not only considers the edit distance between tokens but also introduces the concept of prefix matching to enhance the accuracy and semantic relevance of the mapping. As illustrated in Figure 3c, in the process of matching the token "ultimate" with the corresponding token in the delta model vocabulary, we initially create a candidate set consisting of tokens that either have "ultimate" as a prefix or are prefixes of "ultimate", which includes elements such as ["ultimately", "ult", "ul", "u"]. Subsequently, within this candidate set, by comparing the edit distances, the token "ultimately" is identified as the one most closely matching "ultimate".

4 Experiment

To evaluate the effectiveness of our method, we conducted experiments on two optimization tasks commonly required for LLMs: instruction tuning and unlearning, with the results presented in Sections 4.1 and 4.2. Furthermore, we analyzed the impact of different sizes of delta models and the strength coefficient α on performance in Sections 4.3 and 4.4. Finally, the outcomes of the ablation studies are demonstrated in Section 4.5, where we investigate the necessity of applying LogSoftmax to the logits of LLM outputs and employing prefix match during token mapping.

4.1 Experiment on Instruction Tuning

Instruction tuning refers to the process of fine-tuning pre-trained models to better understand and follow human natural language instructions.

Training Dataset We utilized the GPT4-Alpaca dataset (Peng et al., 2023) to train our delta model. This dataset consists of 52k instruction-following examples, with instructions sourced from Stanford Alpaca data (Taori et al., 2023) and responses generated by GPT-4.

Table 1: Instruction tuning results on AlapcaEval (Win %). In cross-model control, all base models incorporate the same delta model, which is trained using the LLAMA2-7B as the template model.

Method	Params Add	LLAMA2-7B	LLAMA2-13B	LLAMA2-70B	MISTRAL-7B
Vanilla Base Model	-	4.22	5.34	11.55	6.83
LoRA (upper bound)	110M	68.18	75.16	OOM	79.91
Proxy-tuning	220M	8.47 (+4.25)	10.47 (+5.13)	8.59 (-2,96)	- -
CMC (ours)	110M	30.41 (+26.19)	39.04 (+31.51)	49.81 (+38.26)	33.29 (+26.46)

Evaluation Method We employed the AlpacaEval benchmark (Li et al., 2023b) to evaluate the instruction-following ability of our method. This benchmark includes 805 instructions, with GPT-4 serving as the annotator to compare the output of the tested model against Davinci003's output, using win rate as the evaluation metric.

Implementation We introduce TINYLLAMA-110M[3] as the delta model, which is based on the Llama (Touvron et al., 2023) architecture with a parameter size of 110M, featuring 12 Transformer decoder layers and a hidden size of 768. We use LLAMA2-7B as the template LLM to guide the delta model in learning to modify the output of the LLM. LLAMA2-13B, LLAMA2-70B , and MISTRAL-7B are selected as user LLMs to assess the effectiveness of the delta model on LLMs with different parameter sizes and vocabularies.

Baseline Methods **a) LoRA fine-tuning** (Hu et al., 2022): A parameter-efficient fine-tuning method that freezes the pre-trained model weights during training and incorporates trainable rank decomposition matrices into the Transformer layer to reduce the number of trainable parameters. As an immovable method, LoRA **is not suitable for direct comparison with our approach** but serves as a performance upper bound for reference. **b) Proxy-tuning** (Liu et al., 2024a): A method that does not directly fine-tune the model itself but selects a smaller-scale model within the model family as an anti-expert, fine-tunes it as an expert, and leverages the difference in logits between the expert and anti-expert during decoding for larger models. To facilitate a fair comparison, we use TINYLLAMA-110M as the anti-expert, fine-tuned on GPT4-Alpaca as the expert. This approach **cannot be directly applied to models with different vocabularies**.

Analysis The results of instruction tuning are shown in Table 1, where we observe that:

A single portable delta model can enable LLMs with different parameter sizes and vocabularies to achieve the ability to follow instructions. Furthermore, this ability is not constrained by the template model's capabilities, and as the user model's capabilities increase, the ability of the user model to follow instructions also increases. This indicates that the logits transformation for enabling a model to follow instructions can be applied to a wide range of LLMs, which is consistent with the findings in Section 2.

Our approach achieves better performance than Proxy-tuning with the same trainable parameters and fewer inference costs, as our delta model can alter the outputs of LLMs through training with the template model, while Proxy-tuning is constrained by the performance of the anti-expert model.

Our method's performance is not as good as LoRA's, as LoRA incorporates new tunable parameters in each Transformer layer, enabling deep interactions between the LoRA module and the model. However, this approach also prevents the LoRA module from being used as a portable neural network for models in different parameter spaces.

4.2 Experiment on Unlearning

Unlearning refers to the process of making a model forget specific information from the training data in order to prevent privacy leakage.

[3]https://huggingface.co/nickypro/tinyllama-110M

Table 2: Unlearning results on TOFU benchmark. All chat models incorporate the same delta model, which is trained using the LLAMA2-7B-TOFU as the template model. Better scores are bolded.

	Forget Set			Retain Set			Real Author			World Fact		
Method	RL (↓)	P (↓)	TR(↓)	RL(↑)	P(↑)	TR(↑)	RL	P	TR	RL	P	TR
LLAMA2-7B-TOFU	0.99	0.99	0.51	0.98	0.99	0.48	0.93	0.45	0.58	0.87	0.43	0.56
+LoRA	0.01	0.00	0.77	0.71	0.75	0.48	**0.88**	**0.51**	**0.68**	**0.88**	**0.46**	**0.60**
+CMC (ours)	**0.00**	**0.00**	**0.33**	**0.91**	**0.97**	**0.51**	0.84	0.43	0.58	0.85	0.45	0.57
LLAMA2-13B-TOFU	1.00	1.00	0.46	0.99	1.00	0.53	0.89	0.51	0.67	0.86	0.46	0.62
+δ-UNLEARNING	0.38	0.06	0.53	0.53	0.48	0.52	0.61	0.36	0.46	0.83	0.41	0.59
+LoRA	0.03	0.00	0.50	0.85	0.92	0.52	**0.87**	**0.54**	**0.70**	**0.86**	**0.48**	0.63
+CMC (ours)	**0.00**	**0.00**	**0.29**	**0.97**	**0.99**	**0.55**	0.77	0.49	0.65	0.83	0.47	**0.65**
MISTRAL-7B-TOFU	1.00	1.00	0.49	1.00	1.00	0.48	0.84	0.61	0.75	0.88	0.62	0.78
+LoRA	**0.00**	0.00	0.72	0.95	0.96	0.49	**0.79**	0.57	0.71	**0.87**	**0.64**	**0.78**
+CMC (ours)	0.00	**0.00**	**0.30**	**0.99**	**0.99**	**0.51**	0.73	**0.62**	**0.75**	0.86	0.63	0.77

Evaluation Method We utilized the TOFU benchmark (Maini et al., 2024) to evaluate our approach. The test data consists of four subsets of QA pairs, namely Forget Set, Retain Set, Real Author, and World Fact. The Forget Set and Retain Set contain fictitious author information, with the Forget Set representing the information to be forgotten and the Retain Set representing the information to be retained.[4] Real Author and World Fact are used to test the impact of unlearning on other knowledge within the model. Following TOFU's setting, we employed the following three evaluation metrics: **ROUGE-L** evaluates the matching degree between the output of the tested model and the ground truth answer; **Probability** assesses the conditional probability of the tested model outputting the correct answer; **Truth Ratio** calculates a ratio that compares the likelihood of its correct answer to an incorrect answer.

Implementation Initially, we trained LLAMA2-7B-CHAT , LLAMA2-13B-CHAT , and MISTRAL-7B-INSTRUCT on the Forget Set and Retain Set of TOFU to memorize fictitious author information, resulting in LLAMA2-7B-TOFU, LLAMA2-13B-TOFU, and MISTRAL-7B-TOFU. We selected TINYLLAMA-110M as the delta model, with the model learning fictitious author information, LLAMA2-7B-TOFU, serving as the template LLM, and LLAMA2-13B-TOFU and MISTRAL-7B-TOFU serving as user LLMs. During the training phase, we employed a gradient difference strategy, specifically conducting gradient ascent on the Forget Set and gradient descent on the Retain Set. The loss function can be expressed as:

$$\mathcal{L}_{diff} = -\mathcal{L}(S_F) + \mathcal{L}(S_R) \tag{7}$$

Here, $\mathcal{L}$ represents the cross-entropy loss, S_F denotes the Forget Set, and S_R denotes the Retain Set.

Baseline methods a) **LoRA fine-tuning**, which involves training each model separately. b) **δ-UNLEARNING** (Huang et al., 2024) fine-tunes LLAMA2-7B-CHAT model, and guides LLAMA2-13B-CHAT during training and decoding to avoid outputting private information using the changes in logits between the tuned and the original LLAMA2-7B-CHAT. This method necessitates inferring three models during the inference, incurring substantial computational costs.

Analysis The results are shown in Table 2. The delta model trained with LLAMA2-7B-TOFU enables models with varying parameter scales and vocabularies to achieve unlearning effects. Its performance is comparable to LoRA while offering the capability of improving multiple models in a single training session. Furthermore, compared to δ-UNLEARNING, our approach achieves superior performance at less than half the inference cost.

4.3 Impact of Parameter Size of Delta Model on Performance

We experimented with delta models of smaller parameter sizes, specifically the TINYLLAMA-42M[5] and TINYLLAMA-15M[6] models, and tested their performance on the first 50 data points of AlpacaEval.

[4]The test data for Forget Set and Retain Set are paraphrased and perturbed training data, serving as the ground truth answer and incorrect answer, respectively.

[5]https://huggingface.co/nickypro/tinyllama-42M

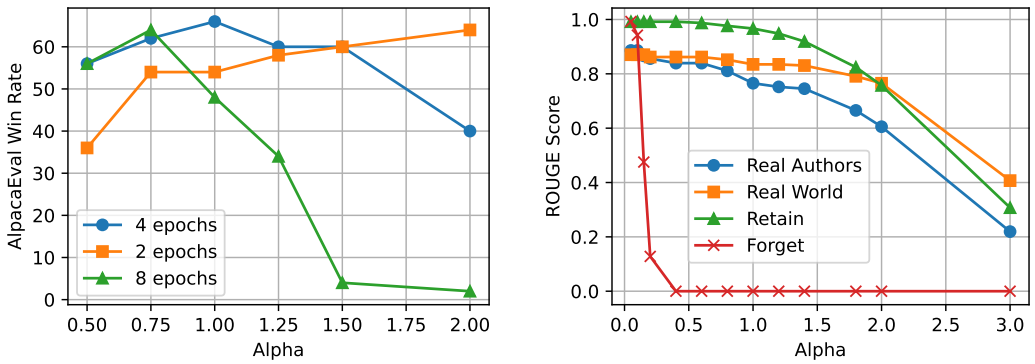
[6]https://huggingface.co/nickypro/tinyllama-15M

Table 3: Different delta model size on first 50 data points of AlpacaEval (win %).

Delta Model Size	LLAMA2-7B	LLAMA2-13B	LLAMA2-70B	MISTRAL-7B
15M	40	50	72	52
42M	54	64	82	50
110M	54	66	74	54

The results are presented in Table 3. In most cases, we find that reducing the delta model's parameter size leads to a decrease in performance, but the performance does not rapidly decline as the parameter size decreases. When the delta model is applied to LLAMA2-70B, a larger delta model size may have a negative effect. We believe this is because the 70B LLM, with a massive parameter size, does not need to make significant adjustments to the output to acquire the ability to follow instructions, and an overly large parameter size in the delta model may lead to overfitting. This indicates that a delta model used for adjusting LLM output logits does not necessarily require a large number of parameters. It also demonstrates the significant potential for small models to assist large models.

4.4 Impact of Strength Coefficient on Performance



(a) Impact on instruction tuning

(b) Impact on unlearning

Figure 4: Impact of strength coefficient α on performance

We experimented with different strength coefficients α and observed their impact on performance.

For instruction tuning, we tested values within the range of [0.5, 2] and evaluated them on the first 50 data points of AlpacaEval. As shown in Figure 4a, we found that the performance was optimal when the α value was 1.0, and increasing or decreasing α resulted in decreased performance. This is similar to causing the delta model to overfit or underfit, so we explored whether adjusting the α value during inference could counteract the overfitting and underfitting during training. Specifically, the delta model performed best after training for four epochs, and we selected the checkpoint after training for two epochs as the underfitting scenario and the checkpoint after training for eight epochs as the overfitting scenario. We found that **adjusting the α value could counteract the overfitting or underfitting** during training. The underfitting delta model had a win rate of 54 when α was 1.0, which increased to 64 when α was 2.0. Similarly, the overfitting delta model had a win rate of 48 when α was 1.0, which increased to 64 when α was 0.75. This phenomenon indicates that our training method offers a considerable degree of fault tolerance, even if incorrect epoch parameters are set during the training phase, it is still possible to achieve the desired performance by adjusting the strength coefficient α.

For unlearning, we found that **adjusting the value of α can serve as a balance between forgetting and retaining**. As shown in Figure 4b, increasing the value of α can prevent the model from outputting more sensitive information, but it may also lead to the loss of necessary information. Conversely, decreasing the value of α can allow the model to retain more essential information, but it may result in the model outputting more sensitive information. Users of LLM can flexibly adjust the α value based on the actual circumstances.

4.5 Ablation Study

In the ablation study, we examined the impact of not applying LogSoftmax to the logits output by the LLM and the effect of omitting prefix matching during token mapping on performance. As demonstrated in Table 4, our findings reveal that the removal of LogSoftmax results in a decline in performance. This suggests that projecting the logits output by the LLM into the logarithmic space can reduce the gap between training and inference. Similarly, the absence of prefix matching in token mapping diminishes the supportive effect of the delta model on the LLM. This indicates that prefix matching has a higher priority than selecting tokens with the minimal edit distance. Eliminating prefix matching could lead to the aggregation of token logits that are semantically unrelated.

Table 4: Ablation study

Method	**AlpacaEval (Win %)**
LLAMA2-7B+delta model	30.41
w/o LogSoftmax	28.64
LLAMA2-13B+delta model	39.04
w/o LogSoftmax	36.85
MISTRAL-7B+delta model	33.29
w/o LogSoftmax	31.22
w/o Prefix Match	30.19

5 Related Work

In this paper, we focus on modifying the outputs of the base language model. The related work can be broadly categorized into two approaches: training-intensive methods and decoding-time methods.

Training-Intensive methods One of the most effective methods for adapting large language models (LLMs) to specific downstream tasks involves updating only a subset of the model parameters, rather than the entire set. For instance, LoRA (Hu et al., 2022) achieves parameter updates by decomposing them into trainable low-rank vectors. Prefix-tuning (Li and Liang, 2021) introduces a series of continuous, task-specific vectors at the beginning of the input sequence for task adaptation. Adapter-tuning (Houlsby et al., 2019) integrates compact, trainable modules within the layers of a pre-trained model to facilitate transfer learning. BitFit (Zaken et al., 2021) selectively updates individual bias vectors in the model's parameters. Although these methods enhance performance in downstream tasks, they still require significant computational resources.

Decoding-Time methods To reduce training costs, some researchers have explored decoding-time methods. For example, DExperts (Liu et al., 2021) enhances or suppresses certain text attributes by integrating expert and anti-expert models during decoding. GeDi (Krause et al., 2021) employs smaller LMs as generative discriminators to steer the output of larger LMs, enhancing safety and control. Proxy-tuning (Liu et al., 2024a) modifies the predictions of an untuned larger model towards a desired outcome. ITI (Li et al., 2024) adjusts activation values during inference to generate more accurate responses. EFT (Mitchell et al., 2023) employs two smaller LMs to emulate the behavior of a larger LM without the need for additional training. Although these methods are effective, they often require the use of two additional language models during inference, increasing inference costs.

6 Conclusion and Limitation Discussion

In this paper, we introduced a training method named CMC, designed to improves multiple LLMs in one-time training, enabling LLM owners who lack data and computational resources to improve their models at a lower cost. Through comparative analysis of the logits shifts in different LLMs before and after fine-tuning, we observed a similarity in these shifts. Based on this observation, we introduced a portable tiny delta model to fit the logits shifts of LLMs, enabling the adjustment of outputs for LLMs of varying sizes and vocabularies. Our experiments in instruction tuning and unlearning tasks have demonstrated the effectiveness of our method.

However, our approach has certain limitations. The performance of the delta model is constrained by the scope of its vocabulary. If the vocabulary of the user's LLM contains tokens from languages not covered by the delta model's vocabulary, then the logits of these linguistically diverse tokens will not be adjusted accordingly.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 62377012 and the Special Fund for International Conferences of Graduate Students at East China Normal University.

References

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

Jiaao Chen and Diyi Yang. 2023. Unlearn what you want to forget: Efficient unlearning for llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12041–12052. Association for Computational Linguistics.

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

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10421–10430. PMLR.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3356–3369. Association for Computational Linguistics.

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

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

James Y Huang, Wenxuan Zhou, Fei Wang, Fred Morstatter, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2024. Offset unlearning for large language models. *arXiv preprint arXiv:2404.11045*.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2038–2047. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 4929–4952. Association for Computational Linguistics.

Chak Tou Leong, Yi Cheng, Jiashuo Wang, Jian Wang, and Wenjie Li. 2023. Self-detoxifying language models via toxification reversal. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4433–4449. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

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.

Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023a. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024a. Tuning language models by proxy. *CoRR*, abs/2401.08565.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6691–6706. Association for Computational Linguistics.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. 2024. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. 2023. An emulator for fine-tuning large language models using small language models. *arXiv preprint arXiv:2310.12962*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *CoRR*, abs/2401.10491.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Hyperparameters and experimental details

A.1 Instruction Tuning

During the training of the delta model, we set the learning rate to 2e-4, batch size to 64, and trained for 4 epochs.

For LoRA fine-tuning, for both LLAMA2-7B and MISTRAL-7B models, we set r to 140 and alpha to 280, while for LLAMA2-13B, r is set to 90 and alpha to 180. For all models, the learning rate is 2e-5, batch size is 32, LoRA target are [q_proj,k_proj,k_proj], and the training lasted for 2 epochs.

When training the expert models for Proxy-tuning, the learning rate was set to 2e-4, batch size to 64, and trained for 16 epochs. We also experimented with varying the number of training epochs and found that performance was best with 16 epochs.

A.2 Unlearning

In training the delta model, we set the learning rate to 1e-4, batch size to 16, and trained for 20 epochs.

For LoRA fine-tuning, across all models, r was set to 32, alpha to 64, with a learning rate of 1e-4, batch size of 16, and training spanned 5 epochs.

A.3 Preliminaries

The hyperparameter settings for Section 2 were consistent with those outlined for instruction tuning.

B Compute Resources

Our experiments were conducted on a server equipped with 512GB of memory and 4 Nvidia A100 40G GPUs.

C Broader Impacts

Our approach can bring about positive social impacts. Specifically, our method allows for the reuse of the fine-tuning outcomes from one model to another. This attribute of supporting repeated use can reduce the cost of model training and decrease carbon emissions. Simultaneously, our method does not present any negative social impacts.

D Quantitative Analysis of the Fine-tuning Effect

By calculating the difference between the logits after and before fine-tuning, we obtain the fine-tuning effects, and we evaluated their similarity by measuring the distance between them, which is assessed using Sinhorn divergence.

$$\text{Distance} = \text{Sinkhorn Divergence}(\mathcal{T}_{\mathcal{M}1}, \mathcal{T}_{\mathcal{M}_2})/|\mathcal{V}_1| \tag{8}$$

Table 5: Average Distance between logits shifts. Smaller distances mean more similarities

Model1	Model2	Average Distance Between Logits Shifts	
		Both train on GPT4-Alpaca	Model1 train on GPT4-Alpaca and Model2 train on GSM8k
LLAMA2-13B	LLAMA2-7B	0.658	3.522
MISTRAL-7B	LLAMA2-7B	1.046	4.760
MISTRAL-7B	LLAMA2-13B	0.754	3.382

We fine-tuned them individually on the GPT4-Alpaca dataset. Additionally, to contrast the fine-tuning effects with other task, we also fine-tuned them on the GSM8k dataset (Cobbe et al., 2021). We selected the first 50 data from AlpacaEval as inputs, and used the output of $\mathcal{M}_1^d$ as the response to form input-response pairs. We chose the average Sinkhorn divergence as the indicator. The result is shown in Table 5. We observed that despite the models having different parameter scales and vocabularies, training on the same dataset still resulted in similar logits shifts. Conversely, significant differences were observed when training on different datasets.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The key claims we make in the abstract and introduction accurately reflect the contribution and scope of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We analyze the limitations in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: In Section 2 we provide proofs that logits shifts are similar.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide our code, the models and hyperparameters we use, and clearly state our approach, and the results in our paper are reproducible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our code, as well as links to public datasets and models.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Because the evaluation with AlpacaEval requires the very expensive GPT-4 API, we did not conduct multiple experiments to provide error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computational resources required for the experiments in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the research conducted in the paper fully conforms to the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss social impacts in Appendix C.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper has no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, the paper properly credits the creators or original owners of assets and respects the license and terms of use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.