Privacy-preserved LLM Cascade via CoT-enhanced Policy Learning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have gained significant attention in on-device applications due to their remarkable performance across real-world tasks. However, on-device LLMs often suffer from suboptimal performance due to hardware limitations. A promising solution to this challenge is cascading a weaker local (on-device) LLM with a more powerful server LLM. While existing research on LLM cascade primarily optimizes the performance-cost trade-off, real-world applications impose additional requirements, such as privacy preservation, which remain largely unaddressed. In this work, we move beyond existing confidence- and logit-based LLM cascade methods and propose **P^{3}Defer**, a novel Chain-of-Thought (CoT)-enhanced **p**olicy learning framework for **p**rivacy-**p**reserved **defer**ral decision-making. Our approach effectively improves cascade efficiency while mitigating privacy risks. Extensive experiments on three benchmark datasets demonstrate the effectiveness and superiority of **P^{3}Defer** over existing methods.

1 Introduction

As Large Language Models (LLMs) continue to evolve rapidly (Touvron et al., 2023; Achiam et al., 2023; Reid et al., 2024), they are increasingly being integrated into real-world applications, enhancing the intelligence of a wide range of systems. At the same time, mobile devices have become indispensable in everyday life. The emergence of on-device intelligence—such as Apple Intelligence (Gunter et al., 2024) and Gemini Live (Reid et al., 2024)—which embeds LLMs directly into devices for more personalized and intelligent user interactions, is gaining traction but remains relatively underexplored (Xu et al., 2024). A major challenge in this area is the hardware limitations of mobile devices, including constraints on compute power, battery life, and storage capacity. As a result, only smaller LLMs, such as Gemma-2B (Team et al., 2024), can be deployed on these devices, leading to trade-offs in performance compared to larger, more powerful models like Gemini. This raises a critical question for the research community: how can we optimize on-device intelligence given these size constraints? The LLM cascade system presents a solution for this challenge.

In an LLM cascade system, a query is usually first processed by a smaller, weaker local (on-device) LLM and is only escalated to a larger, stronger server LLM if the local model's output is deemed insufficient by a deferral module, as shown in Figure 1. This paradigm has garnered significant attention recently (Chen et al., 2023a; Gupta et al., 2024; Yue et al., 2023; Wang et al., 2024). As larger LLMs are often substantially more expensive than their smaller counterparts (e.g., Gemini-1.5 Pro (Reid et al., 2024) costs up to 35 times more than Gemini-Flash[1]), most existing LLM cascade works focused on the exploration of optimal trade-offs between cost and performance. However, real-world applications can be more complicated and requires the cascade system to make deferral decisions beyond just performance-cost consideration. For instance, privacy concerns may arise if personal data is routed to the server LLM where decisions are made based solely on the local answer's quality, as illustrated in Figure 1. Unfortunately, rare studies have explored the privacy-preserved LLM cascade system where to the best of our knowledge, only Hartmann et al. (2024) makes an attempt in this regards. In this study, we move beyond existing cascade system to make a pioneer step for including privacy concerns into the deferral decision making.

One key focus of LLM cascade research is the design of deferral criteria, which determine whether a query needs to be routed to the server model. Existing study on this can be divied into two paradigms: confidence-based and logit-based methods (Please refer to Appendix D for more details if readers are not familiar with deferral decision making in LLM cascade.). Ideally, the deferral criteria should identify queries that the local LLM is unlikely to handle effectively, sending them to the server to significantly improve performance while keeping costs manageable. Conversely, sending queries that the local LLM can address with high quality to the server can result in unnecessary costs. Intuitively, model confidence could serve as a good indicator, with queries routed to the server when the local model is not confident with its response. For instance, Zhu et al. (2024) explored a self-critique strat-

[1] https://ai.google.dev/pricing

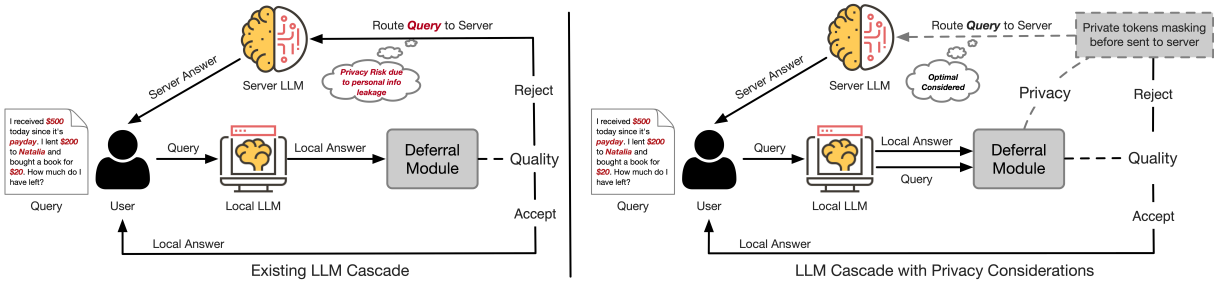


Figure 1: On the left is the existing LLM cascade, where the deferral module makes decisions solely based on the quality of the local answer, potentially leading to privacy leakage. On the right is the privacy-preserved LLM cascade, where deferral decisions are more aligned with the needs of real-world applications.

egy to leverage the local model's intelligence to produce a confidence level in terms of the local answer and make decisions based on the confidence level. However, Jitkrittum et al. (2024) noticed the weakness of confidence-based deferral rule in cases where distribution shifts occur between the training and test datasets. Logit-based methods step further by using the generated token logits of the local answer as features to make deferral decisions. For example, Gupta et al. (2024) found the length bias and token uncertainty problems in cascading by relying on the mean logits and proposed to leverage quantile logits as features to mitigate this problem. Additionally, Wang et al. (2024) introduced cascade-aware training, which incorporates both the local and server LLM's logits into the loss function during local model training, helping the local LLM become more aware of which queries should be deferred to the server. Unfortunately, none of these works explored deferral decision making with respects to privacy concerns which aligns more with real-world needs. Moreover, both confidence-based and logit-based methods are by nature not feasible for including privacy considerations since logits and confidence can only reflect generation quality. To address this gap, we propose incorporating a policy learning strategy into the LLM cascade system. Instead of using a threshold for deferral decision making, we propose to train an agent that can make actions based on cascade needs. Moreover, Chain-of-Though (CoT) has been proven efficient in both training and training-free methods(Wu et al., 2024; Yan et al., 2023). Taking advantages, we propose a novel CoT-enhanced **p**olicy learning framework coupled with a private memory for better **p**rivacy-**p**reserved **defer**ral decision making (**P^{3}Defer**). Different from logit-based or confidence-based methods, our **P^{3}Defer** leverages an agent to make deferral decisions (actions) based on the cascde system needs (environment). This paradigm enable our method to improve the LLM cascade performance while mitigate the privacy leakage problem. In tandem, the contributions of this study are three-fold:

- We extend the current focus of LLM cascading beyond the traditional cost-performance trade-off to include privacy considerations, better aligning with the needs of real-world applications.
- We reformulate the LLM cascade task and innovatively incorporate a CoT-enhanced policy learning strategy coupled with a private objective to perform privacy-preserved deferral decision making, which provides a fresh perspective to the community.
- Extensive experiments on three benchmarks have validated the efficiency and superiority of proposed **P^{3}Defer**, witnessing improvements in LLM cascade performance while mitigating the privacy leakage[2].

2 Methodology

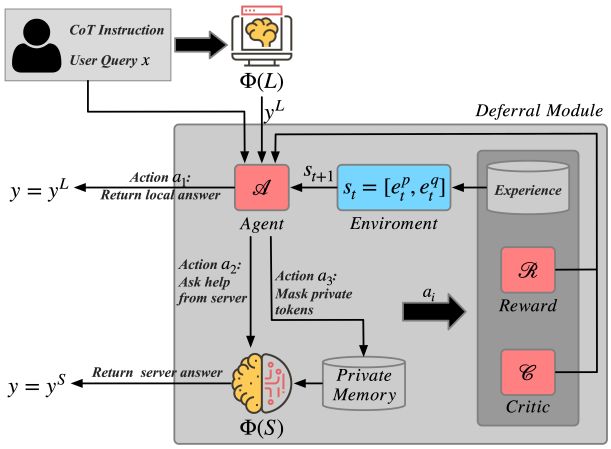


Figure 2: Overview of the proposed **P^{3}Defer** framework. Given a user query x, the local model $\Phi(L)$ generates a response y^L. The agent $\mathcal{A}$ decides among three actions based on the state s_t: (1) return y^L, (2) defer to the server model $\Phi(S)$ for response y^S, or (3) mask private tokens via private memory. The agent is trained via reinforcement learning, where the reward function $\mathcal{R}$ evaluates response quality and privacy, and the critic function $\mathcal{C}$ assesses long-term decision-making.

[2]To encourage further explorations by the community, we will open-source our implementations (a copy is attached with this submissions).

Algorithm 1 P^{3}Defer

Require: $\Phi(L)$, $\Phi(S)$, $D(\mathcal{A}, \mathcal{R}, \mathcal{C}, \mathcal{S})$, $\mathcal{D}$, $\mathcal{M}$

Initialize policy network π_θ, value function V_ψ, and experience buffer $\mathcal{B}$
for each training iteration **do**
  for each query $x \sim \mathcal{D}$ **do**
    Generate local prediction $y^L = \Phi(L)(x)$
    Encode state $s_t = [e_t^p, e_t^q]$
    Sample action $a_t \sim \pi_\theta(s_t, x, y^L)$
    if $a_t = a_1$ **then**
      Set final response $y = y^L$
      Compute reward r_t using formula 2
    else if $a_t = a_2$ **then**
      Query server LLM: $y^S = \Phi(S)(x)$
      Set final response $y = y^S$
      Compute reward r_t using formula 2
    else if $a_t = a_3$ **then**
      Mask sensitive tokens in x via $\mathcal{M}$
      Generate modified query x'
      Query server LLM: $y^S = \Phi(S)(x')$
      Set final response $y = y^S$
      Compute reward r_t using formula 2
    end if
    Store (s_t, a_t, r_t) in buffer $\mathcal{B}$
  end for
  Policy Update:
  Compute advantage $\hat{A}_t = Q^\pi(s_t, a_t) - V^\pi(s_t)$
  Update policy $\theta \leftarrow \theta + \eta \nabla_\theta \mathbb{E}[\hat{A}_t \log \pi_\theta(a_t|s_t)]$
end for

2.1 Preliminary Formulation

Before proceeding, we will first present the preliminary concepts and formulations. Assuming we have an LLM cascade system consists of a local on-device LLM $\Phi(L)$ (smaller and weaker), a server LLM $\Phi(S)$ (larger and stronger) and a deferral module $D(\cdot)$. When a user sends a query x to $\Phi(L)$ and the local model generates an initial answer y^L, the deferral module $D(\cdot)$ needs to determines whether it is necessary to invoke $\Phi(S)$ for the final response back to user.

Typically, existing methods use either the logit distribution of y^L or prompting $\Phi(L)$ to do the deferral decision making(Wang et al., 2024; Zhu et al., 2024). That is say if $D(\cdot)$ accepts y^L, it becomes the final answer y returned to the user. If rejected, the query x is routed to $\Phi(S)$, and the server-generated answer y^S serves as the final response y. However, these attempts are limited in incorporating real-world requirements into considerations due to the nature that their deferral modules are fixed and only make decisions based on confidence or logits[3]. To step further, we move beyond and reformulate the deferral module into a trainable agent so that more considerations such as privacy can be added into deferral decision making.

[3]Please refer to appendix B and D to check detailed explanations and preliminary results of existing methods.

In details, we can represent the deferral module by a tuple $D(\mathcal{A}, \mathcal{R}, \mathcal{C}, \mathcal{S}, x, y^L)$ where $\mathcal{A}$ is the action space containing the actions that the deferral agent can take; $\mathcal{R}$ and $\mathcal{C}$ are the reward function and critic function, respectively; $\mathcal{S}$ denotes the set of environment states. A policy network $\pi_\theta : (\mathcal{S}, x, y^L) \rightarrow P(\mathcal{A})$ maps the user query, local LLM's response and environment to a probability distribution over actions. A historical experience buffer $\mathcal{O} = (o_0, ..., o_t, ..., o_N)$ records the past observations where $o_t = [(s_t, x, y^L), a_t]$ is the (user query, local LLM's response, environment)-action pair and $s_i \in \mathcal{S}$ is the environment state at time t. Inspired by the success of PPO (Schulman et al., 2017), we not only use the a reward function to evaluate current selected action but also leverage a critic function to estimate the cumulative value of historical action selections. Specifically, $\mathcal{R}$ can be denoted as $\mathcal{R} : (\mathcal{S} \times \mathcal{A}) \rightarrow \mathbb{R}$ and $\mathcal{C}$ is $Q(s_t, a_t) - V(s_t)$, where $Q(s, a)$ is the action-state value function and $V(s)$ is the state value function. $r_t = \mathcal{R}(s_t, a_t)$ can represent the reward received at time t. We'll further elaborate how these functions are used within P^3Defer in the following sections. Our objective in this study is to learn the policy π_θ from given training dataset $\mathcal{D}$ to enable $\mathcal{A}$ being aware of both performance-cost and privacy concerns:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t R_t \right] \tag{1}$$

where τ denotes a trajectory of state-action pairs.

2.2 P^{3}Defer

Unlike existing methods where the deferral module is fixed and only makes decisions based on logits distribution or model's confidence level. The deferral module $D(\cdot)$ in **P^{3}Defer** is formulated as a reinforcement learning agent that selects among three actions: returning the local model's output y^L, requesting a response from the server LLM y^S, or masking private tokens before making a deferral decision. The module operates within an environment defined by the tuple $D(\mathcal{A}, \mathcal{R}, \mathcal{C}, \mathcal{S}, x, y^L)$ which contains:

Action Space ($\mathcal{A}$). The available actions include:

- a_1: accept y^L and set final response $y = y^L$
- a_2: defer x to $\Phi(S)$ and let final response $y = y^S$.
- a_3: apply privacy masking using private memory $\mathcal{M}$ on x and routing the privacy masked x' to $\Phi(S)$, final response $y = \Phi(S)(x')$.

State Space ($\mathcal{S}$). There are four environment states:

- x contains privacy concerns, y^L is good.
- x does not contain privacy concerns, y^L is good.
- x contains privacy concerns, y^L is bad.
- x does not contains privacy concerns, y^L is bad.

Each state $s_t = [e_t^p, e_t^q]$ consists of privacy- and quality-related embeddings capturing the four possible states above based on the given input query x and the local llm's response y^L.

Reward Function ($\mathcal{R}$). The reward function optimizes both response quality and privacy compliance:

$$R_t = \mathbb{P}^q(y, \hat{y}) + \lambda \mathbb{P}^p(x) \tag{2}$$

where $\mathbb{P}^q(y, \hat{y})$ measures the generation quality of final response y with respect to the golden responses $\hat{y}$, $\mathbb{P}^p(x)$ represents the identification of privacy leakage and λ is a scaling factor. $\mathbb{P}^q$ and $\mathbb{P}^p$ are both in the form of entropy calculation as referred in Table 4

Critic Function ($\mathcal{C}$). The critic model evaluates the expected return for different actions and guides the policy updates accordingly.

$$V^\pi(s_t) = \mathbb{E}\pi \left[\sum t' = t^T \gamma^{t'-t} R_{t'} \right] \tag{3}$$

where $\gamma \in [0, 1]$ is the discount factor, controlling the importance of future rewards; T is the trajectory length; R_t is the immediate reward as defined in your equation. The expectation is taken over all possible trajectories following policy $\pi_\theta(a|s)$.

Additionally, the state-action value function (Q-function) is:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} \mid s_t, a_t \right] \tag{4}$$

which evaluates the expected reward after taking action a_t in state s_t.

Using the PPO framework, the policy network $\pi_\theta(a|s)$ updates its parameters θ by maximizing the following objective:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t, x, y^L) \hat{A}_t \right] \tag{5}$$

where $\hat{A}_t = Q^\pi(s_t, a_t) - V_\psi(s_t)$ is the advantage function.

2.3 Local LLM Training

The local LLM $\Phi(L)$ is trained using two key techniques: (1) **CoT-enhanced instruction tuning** and (2) **knowledge distillation** from the server LLM $\Phi(S)$ when deferral occurs.

CoT-enhanced Instruction Tuning To improve reasoning capabilities, we fine-tune the local LLM using a dataset of instruction-response pairs enhanced with chain-of-thought (CoT) reasoning. In this paper, we mainly use the zero-shot CoT prompting to formulate our instructions as can be seen in appendix A. Given an instruction x and the corresponding target response $\hat{y}$, the loss function is defined as:

$$\mathcal{L}_{\text{inst}} = -\sum_t \log P_{\Phi(L)}(\hat{y}_t \mid x, \hat{y}_{<t}). \tag{6}$$

This objective encourages the model to generate responses aligned with human-annotated outputs while incorporating reasoning steps.

Knowledge Distillation from Server LLM When the server LLM $\Phi(S)$ is invoked due to deferral, the local LLM learns from the distilled server responses y^S. The knowledge distillation loss minimizes the divergence between the local and server predictions:

$$\mathcal{L}_{\text{KD}} = \sum_t D_{\text{KL}} \left(P_{\Phi(S)}(y_t^S \mid x, y_{<t}^S) \parallel P_{\Phi(L)}(y_t^S \mid x, y_{<t}^S) \right), \tag{7}$$

where D_{KL} is the Kullback-Leibler divergence. This loss ensures that the local model mimics the server model's outputs when necessary.

Training Objective The overall training objective combines the two losses:

$$\mathcal{L} = \mathcal{L}_{\text{inst}} + \lambda_{\text{KD}} \mathcal{L}_{\text{KD}}, \tag{8}$$

where λ_{KD} controls the influence of knowledge distillation. This framework enables the local LLM to improve its reasoning and generalization capabilities while reducing reliance on the server.

2.4 Private Memory $\mathcal{M}$

Unlike previous work (Hartmann et al., 2024), which relies on the local LLM to identify and rewrite private tokens, this approach carries the risk of altering the original meaning of the given query during the rewriting process. To address this issue, we introduce an innovative private memory $\mathcal{M}$ that pre-stores private tokens extracted from a large corpus (Zhang et al., 2024a). When a private query is encountered, the private memory efficiently identifies and masks private tokens without modifying the original meaning.

The memory is structured as a dynamic, growing list, where private tokens are detected by measuring the Levenshtein distance of each token. Once identified, replacing private tokens with similar alternatives helps mitigate privacy leakage while preserving the original intent of the query, thereby ensuring the quality of the final response.

2.5 Inference

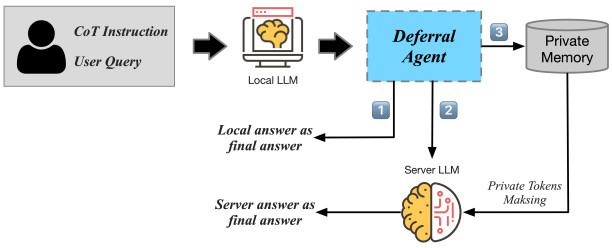


Figure 3: Inference process of the proposed framework. The *Deferral Agent* determines whether to return the local response, defer to the server LLM, or apply privacy masking based on the input query.

During inference, the user query x is first processed by the local LLM $\Phi(L)$, generating an initial response y^L. The *Deferral Agent* $\mathcal{A}$ then decides among three actions based on the query context: *Use local response:* If the local LLM's response is deemed sufficient, it is returned as the final answer; *Defer to server LLM:* If the query is too complex or uncertain and presents no privacy concerns, the agent queries the server LLM $\Phi(S)$, which provides a refined response y^S; *Mask private tokens:* If sensitive information is detected, the agent applies a privacy-preserving mechanism using the *Private Memory* module before passing the modified query to the server. The agent optimally selects actions to balance response quality and privacy, ensuring reliable and secure query resolution.

3 Experimental Settings

3.1 Datasets

To validate the effectiveness of P^3Defer, we opt for three benchmarks with privacy concerns that cover daily scenarios in on-device intelligence application to test our methods as below, more statistics can be seen in appendix C.3 and Table 4.

GSM8K(Cobbe et al., 2021) is a graduate student mathematical dataset consisting of mathematical questions and corresponding solutions, of which some questions contain personal information for privacy study(Hartmann et al., 2024).

MedSum(Zekaoui et al., 2023) is a medical related dataset with a focus on summarizing the customer health question. The dataset contains customer health questions and corresponding summaries which contains personal healthcare information.

EmailSum(Zhang et al., 2021) is a sequence-to-sequence email summarization dataset consisting of daily email thread and corresponding summary. The summary types are available in long-summary and short-summary, we use short-summary in this study.

3.2 Tasks, Metrics & Baselines

We evaluate our proposed P^3Defer on three commonly used daily tasks: mathematical QA, medical inquiry summarization, and email summarization, as indicated in Table 4. We also incorporate the metric of privacy leakage (Hartmann et al., 2024), which calculates the average number of privacy tokens leaked when sending queries to the server LLM (Check more details in appendix C.3).

To the best our knowledge, rare study has been made in privacy-preserved LLM cascade except Hartmann et al. (2024) leverages in-context learning for query rewritten to mitigate privacy leakage problem. Thus, we first compare our P^3Defer with existing logit-based (Wang et al., 2024; Jitkrittum et al., 2024) and confidence-based (Zhu et al., 2024) cascade methods. For logit-based methods, we are using Instruction Tuning (IT) and Loss Tuning (LT); for confidence-based method, we are using Few-shot In-context Learning (Few-shot ICL), detailed implementation can be seen in appendix D. Further, we compare our P^3Defer with other policy learning methods that close to our work: TREACLE(Zhang et al.) and Bilevel(Yan et al., 2023). Next, we conduct privacy study to evaluate how P^3Defer mitigates privacy problem and compare our work with Hartmann et al. (2024). Extensive experiments validate the efficiency and superiority of P^3Defer in privacy-preserved LLM cascade.

3.3 Implementation Details

For implementation details, we leverage the Transformers(Wolf et al., 2020) as the base code and conduct extensive experiments with the Gemma models(Team et al., 2024): **Gemma-2B** as the local LLM, **Gemma-7B** as the server LLM. Notably, the server LLM is fine-tuned on all datasets to reach reasonably great performance, of which the server LLM's ability on GSM8K, MedSum and EmailSum are 52.85%, 61.22% and 56.51%, respectively. We use the AdamW optimizer(Loshchilov and Hutter, 2018; Paszke et al., 2017) with a learning rate of 5e-4 and also a linear warm-up scheduler initialized with 10% of the total training steps as warm-up steps and a weight decay of 1e-4 to avoid over-fitting for all the experiments. The batch size per device is set to 8. All the experiments are conducted on two computation nodes configured with eight 80G H100 GPUs.

4 Experimental Results

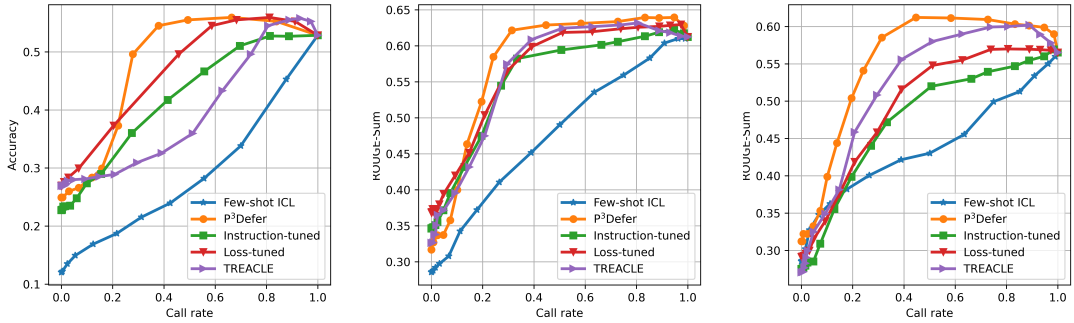
4.1 Cascade Study

Cascade Performance One of the key advantages of LLM cascading is its ability to enhance performance without increasing the size of the base local LLM. As shown in Table 1, confidence-based models primarily rely on the server LLM to boost performance, while logit-based methods selectively defer difficult queries that the local model cannot solve, leading to performance improvements. In contrast, our proposed P^3Defer achieves state-of-the-art performance across all three benchmarks, demonstrating an accuracy of 55.96% with a call rate of 66.41% on GSM8K, a ROUGE-Sum score of 63.94% with a call rate of 69.71% on MedSum, and a ROUGE-Sum score of 61.21% with a call rate of 44.7% on EmailSum. Notably, P^3Defer outperforms all other baselines, achieving post-cascade improvements of 3.11%, 2.72%, and 4.70% over the server model across the three datasets, respectively.

Performance vs Cost A crucial factor in evaluating an LLM cascade system is the trade-off between performance and cost, where the ideal approach maximizes performance gains while minimizing the server call rate. As observed in Table 1, policy learning-based methods, such as TREACLE and P^3Defer, make fewer calls to the server while still improving performance, distinguishing them from confidence-based and logit-based approaches. Furthermore, as depicted

Dataset	Method Type / Metric	%	Confidence-based Few-shot ICL	Logit-based IT	Logit-based LT	Policy Learning TREACLE	Policy Learning P^3Defer
GSM8K	CR		100	100	81.2	93.1	**66.41**
	SCR		28.13	28.13	31.75	84.31	**92.61**
	Acc	$\Phi(L)$	11.83	26.08	26.91	24.31	27.33
		$\Phi(L)+\Phi(S)$	52.85	52.85	55.92	55.78	**55.96**
		vs $\Phi(S)$	N.A.	N.A.	↑3.07	↑2.07	↑**3.11**
MedSum	CR		96.2	94.8	97.3	80.6	**69.71**
	SCR		26.09	26.89	26.92	76.93	**88.40**
	R-S	$\Phi(L)$	28.55	34.61	36.77	34.87	35.31
		$\Phi(L)+\Phi(S)$	61.97	62.18	62.95	63.17	**63.94**
		vs $\Phi(S)$	↑0.75	↑0.96	↑1.73	↑1.95	↑**2.72**
EmailSum	CR		100	98.5	80.6	88.9	**44.7**
	SCR		31.77	39.16	46.93	79.16	**94.61**
	R-S	$\Phi(L)$	24.59	29.49	28.58	27.06	28.91
		$\Phi(L)+\Phi(S)$	56.51	56.92	56.99	60.19	**61.21**
		vs $\Phi(S)$	N.A.	↑0.41	↑0.48	↑3.68	↑**4.70**

Table 1: The best cascade performance of $\Phi(L)$ across three benchmarks. CR denotes call rate, indicating the proportion of queries sent to the server. SCR represents safe call rate, reflecting the number of queries that are safe (i.e., those sent to the server that do not contain privacy information) among the total sent queries. Acc refers to accuracy, while R-S indicates the ROUGE-Sum score. The symbol ↑ signifies an improvement compared to $\Phi(S)$. The **red** number pair shows the best cascade performance (lower call rate with higher scores), the **blue** number indicates the safest method.



(a) GSM8K (b) MedSum (c) EmailSum

Figure 4: Curves depicting cascade performance versus call rate for different methods across all three datasets: (a) GSM8K, (b) MedSum, and (c) EmailSum.

in Figure 4, P^3Defer demonstrates superior deferral decision-making, as its performance curve reaches an inflection point earlier while attaining the highest performance compared to other methods. Moreover, in Figure 4, we observe that TREACLE exhibits different trends on the GSM8K dataset compared to the two summarization datasets. We attribute this to TREACLE's reliance on its routing strategy rather than enhancing the local LLM's capabilities, whereas other methods focus on both cascade deferral and improving the local LLM. Additionally, an interesting observation is that the confidence-based method demonstrates inconsistencies across the three datasets, suggesting that instructing the local LLM for cascading leads to unreliable performance. These findings highlight the effectiveness and superiority of P^3Defer in optimizing cascade performance while maintaining cost-efficient.

4.2 Privacy Study

Beyond its improvements in cascade performance, our P^3Defer also demonstrates a remarkable ability to mitigate privacy concerns. As shown in Table 1, P^3Defer achieves a safe call rate of 92.61%, 88.40%, and 94.61% across the three datasets, respectively. Notably, confidence-based methods achieve only around 28.66%, indicating that relying solely on the local LLM to identify privacy-sensitive queries is unreliable. Moreover, while logit-based methods offer some improvements in privacy sensitivity, they still fall short compared to policy-learning-based approaches. This finding is further validated by the results in Table 2, where the precision and recall scores of confidence- and logit-based methods remain inferior to those of policy-learning-based methods. Similar patterns emerge in mitigating privacy leakage, as confidence-based methods leak the most private to-

Dataset	Metric	Few-shot ICL	Instruction Tuning	Loss Tuning	TREACLE	P^3Defer
GSM8K	precision	64.17	82.95	91.79	88.17	**96.31**
	recall	44.20	72.89	87.24	76.45	**88.79**
	r(leakage)	95.11	84.17	75.98	74.22	**20.11**
MedSum	precision	68.85	85.62	90.10	87.41	**92.17**
	recall	42.99	68.84	82.99	68.41	**88.56**
	r(leakage)	97.60	72.14	70.10	70.54	**23.87**
EmailSum	precision	68.85	85.62	90.10	82.17	**96.91**
	recall	42.99	68.84	82.99	62.43	**85.77**
	r(leakage)	80.79	73.46	56.52	55.62	**16.34**

Table 2: Privacy study. Precision and recall are used for evaluating the ability of different methods on identifying queries with privacy concerns, r(leakage) measures the ratio between leaked private tokens and all private tokens.

kens across all three datasets, reinforcing the unreliability of instructing the model itself to rewrite queries. Although logit-based methods provide some mitigation, their performance remains suboptimal. We attribute this to the fundamental limitation of logit-based methods: their primary objective is to align logits with quality confidence, making them unsuitable for incorporating additional considerations such as privacy protection during deferral decisions. In contrast, P^3Defer achieves average relative reductions of 75.35%, 68.64%, and 74.81% in leaked token ratios across the three datasets. This substantial reduction highlights the advantages of P^3Defer in handling privacy-sensitive queries, which we attribute to the integration of private memory. By leveraging private memory that pre-stores private tokens, the local LLM does not need to focus on rewriting queries. Instead, the memory mechanism assists in identifying and masking private tokens before sending queries to the server, leading to significant improvements in mitigating privacy leakage. Together, policy learning enables P^3Defer to accurately identify privacy-sensitive queries, while private memory effectively mitigates private token leakage, ensuring a more secure and privacy-aware LLM cascade system.

4.3 Ablation Study

Ablation on Cascade Performance We further conduct ablation study on the usage of CoT. The results are presented in Figure 5, we observe that incorporating CoT reasoning consistently improves cascade performance across all datasets, albeit with varying magnitudes. The most significant improvement is observed on the GSM8K dataset, where the model with CoT outperforms its counterpart without CoT by approximately 3%. This suggests that CoT reasoning enhances logical reasoning capabilities, allowing the local model to make better-informed cascade decisions. For MedSum and EmailSum, the performance gap between CoT and non-CoT models is relatively smaller (around 1-2%) which we attribute to the fact that MedSum and EmailSum rely more on semantic understanding and less on multi-step reasoning, making CoT less critical in these cases. Overall, these findings suggest that CoT is a beneficial augmentation to local model training, particularly in reasoning-intensive tasks which further validate the effectiveness of the whole P^3Defer design. **Ablation on Privacy Preservation** Beyond cascade performance, privacy preservation is also a crucial objective of our approach. To further investigate the impact of memory design, we conduct an ablation study on the memory component, as shown in Fig. 6. The results reveal that leveraging private memory significantly mitigates privacy token leakage, as indicated by the substantially lower violet bar compared to the pink one.

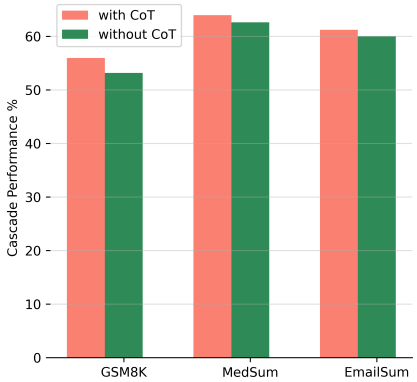


Figure 5: Ablation study on CoT usage.

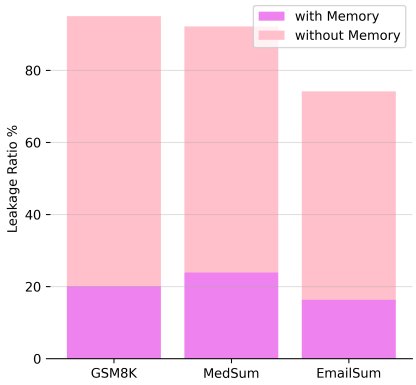


Figure 6: Ablation study on private memory usage.

This demonstrates that private memory is far more effective than relying on the local LLM's rewriting ability to reduce private token leakage, further validating the overall design of P^3Defer of which, policy learning endorse P^3Defer the ability to accurately detect privacy-sensitive queries while private memory serves for mitigates private token leakage, ensuring an effective and privacy-preserved LLM cascade system.

5 Conclusion & Furture Work

In this study, we advance the privacy-preserved LLM cascade by incorporating policy learning coupled with a private memory, moving beyond existing approaches that primarily emphasize cost-performance trade-offs. This enhancement aligns more closely with the demands of real-world applications. Extensive experiments demonstrate that P^3Defer significantly mitigate the privacy leakage problem while improving the llm cascade systemperformance.

While this work represents the pioneer effort to introduce privacy-preserved LLM cascade, future research will explore more on other factors that fit real-world cascade system. We also aim to develop more computational efficient and multi-objective optimized methods to sustain favorable cost-performance trade-offs while accommodating a wider array of objectives such as latency. Innovations on training local llm and deferral module together are also worth to investigate.

6 Related Work

LLM Cascade Cascading has been extensively studied and applied across various domains due to its ability to enhance system performance in downstream tasks by selecting appropriate models (Hu et al., 2023; Li et al., 2019; Karlos et al., 2016; Viola and Jones, 2001). Recently, this approach has garnered increasing attention for improving the performance of large language models (LLMs). For instance, Agrawal et al. (2024); Xu et al. (2023); Chen et al. (2024) have explored speculative decoding, which leverages a larger and more powerful LLM to verify token-level accuracy during the inference of a smaller LLM, thereby accelerating the overall process. Despite the success of cascading, researchers have observed that larger, more capable LLMs (e.g., GPT-4 (Achiam et al., 2023)) can be expensive, while smaller LLMs (e.g., GPT-2 (Radford et al., 2019)) may not always meet performance requirements. This has led to the emergence of the deferral rule—determining when to invoke the larger LLM—as a critical area of exploration for balancing performance and cost in LLM cascading (Shekhar et al., 2024; Chen et al., 2023a,b). There are two primary approaches to deferral: confidence-based methods and router-based methods. Confidence-based methods leverage the LLM's confidence in its generated answers to inform deferral decisions. Ideally, an LLM exhibits higher confidence for higher-quality answers, and vice versa. A straightforward approach involves asking the LLM to provide a confidence score alongside its answers, invoking the stronger LLM when the score is low (Zhu et al., 2024). Another prevalent method utilizes the logits of generated tokens to represent the LLM's confidence, with recent studies exploring operations on logits, such as mean (Gupta et al., 2024) and quantile (Jitkrittum et al., 2024). Wang et al. (2024) extended this concept by incorporating the logits of the stronger LLM into the loss function for tuning the weaker LLM, enhancing its understanding of the cascade logic and enabling deferral decisions based on logit indicators. In contrast, router-based methods use a routing mechanism to determine whether to invoke the stronger LLM. Typically, the router selects the most suitable LLM for different tasks. Non-predictive routing evaluates the outputs of multiple LLMs to select the best one, but this can be costly due to the need to assess all involved models (Madaan et al., 2023; Lee et al., 2023; Wang et al., 2023). Predictive routing systems, however, employ reward functions that allow the router to anticipate which LLM to select, thus avoiding the latency associated with extensive evaluations (Shnitzer et al., 2023; Šakota et al., 2024; Hari and Thomson, 2023). Nonetheless, router-based methods require prior knowledge of each LLM's capabilities and may incur significant costs when trying to enhance performance, compared to confidence-based methods (Hu et al., 2024b,a). Different from existing methods, we incorporate a CoT-enhanced policy learning strategy coupled with a private memory design to achieve privacy-preserved LLM cascade.

Privacy-preservation Privacy has always been a core concern in LLM research (Kim et al., 2024; Zhang et al., 2024d; Das et al., 2024; Janryd and Johansson, 2024; Feng et al., 2024), particularly for on-device LLM applications (Zhang et al., 2024c; Peng et al., 2024; Yuan et al., 2024). LLMs have been shown to inadvertently reveal sensitive information, such as personal names (Evertz et al., 2024; Kim et al., 2024). To address these privacy issues, Liu et al. (2024a,b,c); Kassem et al. (2023) proposed machine unlearning techniques that enable LLMs to forget sensitive information, thus mitigating the risk of generating harmful or biased content. Another approach is differential privacy, which adds noise to the training data, making it more difficult to identify individual data points (Hartmann et al., 2024). Additionally, Zhang et al. (2024e) suggested using contrastive learning to erase an LLM's memory of user information. While these methods have shown success across diverse user bases, our objective is to enhance the sensitivity of our LLM cascade framework to privacy concerns in single-user settings. To achieve this, we aim to leverage in-context learning and integrate binary privacy identification into the loss function, allowing the local LLM to be more attuned to privacy considerations during the cascading process. Further, we innovatively utilize a private memory into our design to achieve privacy-preseveration.

Limitations

Despite the empirical success, our P^3Defer presents two limitation that may ask for further attentions to work on. **First**, compare with confidence- and logit-based methods that leverage thresholds to make deferral decisions, our method needs to train a policy that contains four components (even some of them have small set of parameters), the computational costs are higher. However, the higher costs obtain a reasonable feedback on the performance and privacy-preservation ability. We may still want to seek ways for reducing the computational costs(Zhou et al., 2023). **Second**, our private memory design is a pre-process which means it can not be updated even new privacy tokens appear. This may pose hackers a way to attack this system by simply use synonym(Zhang et al., 2024a). Further explorations in including other memory techniques(Zhang et al., 2024b) can be important.

Ethics Statement

After carefully reviewing the ACL Ethics Policy, we are committed to show our respect and obey to consent all.

Acknowledgment

Thanks for ACL community and Openreview platform, will further add details in the final revision.

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*.

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav Gulavani, Ramachandran Ramjee, and Alexey Tumanov. 2024. Vidur: A large-scale simulation framework for llm inference. *Proceedings of Machine Learning and Systems*, 6:351–366.

Boyuan Chen, Mingzhi Zhu, Brendan Dolan-Gavitt, Muhammad Shafique, and Siddharth Garg. 2024. Model cascading for code: Reducing inference costs with model cascading for llm based code generation. *arXiv preprint arXiv:2405.15842*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023a. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. Less is more: Using multiple llms for applications with lower costs. In *Workshop on efficient systems for foundation models@ ICML2023*.

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, 2021. *URL https://arxiv.org/abs/2110.14168*.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2024. Security and privacy challenges of large language models: A survey. *arXiv preprint arXiv:2402.00888*.

Keqi Deng, Guangzhi Sun, and Philip C Woodland. 2024. Wav2prompt: End-to-end speech prompt generation and tuning for llm in zero and few-shot learning. *arXiv preprint arXiv:2406.00522*.

Jonathan Evertz, Merlin Chlosta, Lea Schönherr, and Thorsten Eisenhofer. 2024. Whispers in the machine: Confidentiality in llm-integrated systems. *arXiv preprint arXiv:2402.06922*.

Qizhang Feng, Siva Rajesh Kasa, Hyokun Yun, Choon Hui Teo, and Sravan Babu Bodapati. 2024. Exposing privacy gaps: Membership inference attack on preference data for llm alignment. *arXiv preprint arXiv:2407.06443*.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. 2024. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. 2024. Language model cascades: Token-level uncertainty and beyond. *arXiv preprint arXiv:2404.10136*.

Surya Narayanan Hari and Matt Thomson. 2023. Tryage: Real-time, intelligent routing of user prompts to large language model. *arXiv preprint arXiv:2308.11601*.

Florian Hartmann, Duc-Hieu Tran, Peter Kairouz, Victor Cărbune, et al. 2024. Can llms get help from other llms without revealing private information? *arXiv preprint arXiv:2404.01041*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024a. Mars: A benchmark for multi-llm algorithmic routing system. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024b. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Shengkai Hu, Haoyu Wang, and Basel Halak. 2023. Cascaded machine learning model based dos attacks detection and classification in noc. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE.

Balder Janryd and Tim Johansson. 2024. Preventing health data from leaking in a machine learning system: Implementing code analysis with llm and model privacy evaluation testing.

Wittawat Jitkrittum, Neha Gupta, Aditya K Menon, Harikrishna Narasimhan, Ankit Rawat, and Sanjiv Kumar. 2024. When does confidence-based cascade deferral suffice? *Advances in Neural Information Processing Systems*, 36.

Stamatis Karlos, Nikos Fazakis, Sotiris Kotsiantis, and Kyriakos Sgarbas. 2016. A semisupervised cascade classification algorithm. *Applied Computational Intelligence and Soft Computing*, 2016(1):5919717.

Aly Kassem, Omar Mahmoud, and Sherif Saad. 2023. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4360–4379.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. 2024. Propile: Probing privacy leakage in large language models. *Advances in Neural Information Processing Systems*, 36.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2023. Orchestrallm: Efficient orchestration of language models for dialogue state tracking. *arXiv preprint arXiv:2311.09758*.

Ang Li, Xue Yang, and Chongyang Zhang. 2019. Rethinking classification and localization for cascade r-cnn. *arXiv preprint arXiv:1907.11914*.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Heng Huang, Jiuxiang Gu, and Tianyi Zhou. 2023. Reflection-tuning: Data recycling improves llm instruction-tuning. *arXiv preprint arXiv:2310.11716*.

Susan Lincke. 2024. Complying with hipaa and hitech. In *Information Security Planning: A Practical Approach*, pages 345–365. Springer.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu, Yuguang Yao, Hang Li, Kush R Varshney, et al. 2024a. Rethinking machine unlearning for large language models. *arXiv preprint arXiv:2402.08787*.

Zhenhua Liu, Tong Zhu, Chuanyuan Tan, and Wenliang Chen. 2024b. Learning to refuse: Towards mitigating privacy risks in llms. *arXiv preprint arXiv:2407.10058*.

Zheyuan Liu, Guangyao Dou, Zhaoxuan Tan, Yijun Tian, and Meng Jiang. 2024c. Towards safer large language models through machine unlearning. *arXiv preprint arXiv:2402.10058*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Guojun Peng, Zhiguang Cao, Yining Ma, and Yue-Jiao Gong. 2024. Llamoco: Instruction tuning of large language models for optimization code generation. *arXiv preprint arXiv:2403.01131*.

Aman Madaan, Pranjal Aggarwal, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. 2023. Automix: Automatically mixing language models. *arXiv preprint arXiv:2310.12963*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Dan Peng, Zhihui Fu, and Jun Wang. 2024. Pocketllm: Enabling on-device fine-tuning for personalized llms. *arXiv preprint arXiv:2407.01031*.

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

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Marija Šakota, Maxime Peyrard, and Robert West. 2024. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 606–615.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shivanshu Shekhar, Tanishq Dubey, Koyel Mukherjee, Apoorv Saxena, Atharv Tyagi, and Nishanth Kotla. 2024. Towards optimizing the costs of llm usage. *arXiv preprint arXiv:2402.01742*.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro,

Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

Congchao Wang, Sean Augenstein, Keith Rush, Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Aditya Krishna Menon, and Alec Go. 2024. Cascade-aware training of language models. *arXiv preprint arXiv:2406.00060*.

Yiding Wang, Kai Chen, Haisheng Tan, and Kun Guo. 2023. Tabi: An efficient multi-level inference system for large language models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pages 233–248.

Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zongqian Wu, Baoduo Xu, Ruochen Cui, Mengmeng Zhan, Xiaofeng Zhu, and Lei Feng. 2024. Rethinking chain-of-thought from the perspective of self-training. *arXiv preprint arXiv:2412.10827*.

Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. Llmcad: Fast and scalable on-device large language model inference. *arXiv preprint arXiv:2309.04255*.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. 2024. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*.

Xue Yan, Yan Song, Xinyu Cui, Filippos Christianos, Haifeng Zhang, David Henry Mguni, and Jun Wang. 2023. Ask more, know better: Reinforce-learned prompt questions for decision making with large language models. *arXiv preprint arXiv:2310.18127*.

Yizhen Yuan, Rui Kong, Yuanchun Li, and Yunxin Liu. 2024. Wip: An on-device llm-based approach to query privacy protection. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, pages 7–9.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2023. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*.

Nour Eddine Zekaoui, Siham Yousfi, Mounia Mikram, and Maryem Rhanoui. 2023. Enhancing large language models' utility for medical question-answering: A patient health question summarization approach. In *2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–8. IEEE.

Kai Zhang, Yangyang Kang, Fubang Zhao, and Xiaozhong Liu. 2024a. LLM-based medical assistant personalization with short- and long-term memory coordination. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2386–2398, Mexico City, Mexico. Association for Computational Linguistics.

Kai Zhang, Lizhi Qing, Yangyang Kang, and Xiaozhong Liu. 2024b. Personalized llm response generation with parameterized memory injection. *arXiv preprint arXiv:2404.03565*.

Shiquan Zhang, Ying Ma, Le Fang, Hong Jia, Simon D'Alfonso, and Vassilis Kostakos. 2024c. Enabling on-device llms personalization with smartphone sensing. *arXiv preprint arXiv:2407.04418*.

Shiyue Zhang, Asli Celikyilmaz, Jianfeng Gao, and Mohit Bansal. 2021. Emailsum: Abstractive email thread summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Xiaojin Zhang, Yulin Fei, Yan Kang, Wei Chen, Lixin Fan, Hai Jin, and Qiang Yang. 2024d. No free lunch theorem for privacy-preserving llm inference. *arXiv preprint arXiv:2405.20681*.

Xuan Zhang and Wei Gao. 2023. Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method. *arXiv preprint arXiv:2310.00305*.

Xuechen Zhang, Zijian Huang, Ege Onur Taga, Carlee Joe-Wong, Samet Oymak, and Jiasi Chen. Efficient contextual llm cascades through budget-constrained policy learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhaohan Zhang, Ziquan Liu, and Ioannis Patras. 2024e. Get confused cautiously: Textual sequence memorization erasure with selective entropy maximization. *arXiv preprint arXiv:2408.04983*.

Jin Zhao, Chao Liu, Jiaqing Liang, Zhixu Li, and Yanghua Xiao. 2024. A novel cascade instruction tuning method for biomedical ner. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11701–11705. IEEE.

Zhengyuan Zhou, Susan Athey, and Stefan Wager. 2023. Offline multi-action policy learning: Generalization and optimization. *Operations Research*, 71(1):148–183.

Yun Zhu, Yinxiao Liu, Felix Stahlberg, Shankar Kumar, Yu-Hui Chen, Liangchen Luo, Lei Shu, Renjie Liu, Jindong Chen, and Lei Meng. 2024. Towards an on-device agent for text rewriting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2535–2552, Mexico City, Mexico. Association for Computational Linguistics.

A Prompts

```
instruction_prompt = r'''Assume you're a student working on some
mathematical problems. Now, you'll be giving mathematical problems, you
need to do two tasks:
a. Check if the question contains personal information (e.g., names etc.),
output Yes or No only;\n
b. Solve this question;

Here are some examples:
Question: Hector purchased a container of gumballs. He gave 4 to Todd,
then he gave twice as many as he had given Todd to Alisha, and then he
gave 5 less than four times as many to Bobby as he had given to Alisha. If
Hector had 6 gumballs remaining, what is the total number of gumballs that
Hector purchased? \n
Output:
Let's think step by step:
The question contains person names so the answer to a. is Yes.
Hector gave to Alisha twice as many as he had given Todd, for a total of
4*2=<<4*2=8>>8 gumballs, Hector gave 5 less than four times as many to
Bobby as he had given to Alisha, or a total of (8*4)-5=<<8*4-5=27>>27
gumballs. If Hector had 6 gumballs remaining, he originally purchased
4+8+27+6=<<4+8+27+6=45>>45 gumballs. So the answer is 45.
a. Contains Personal Information: Yes.
b. Answer: 45.

Case,
Question: {question}\n
Output:
Let's think step by step:
'''
```

Figure 7: Prompts Used on three datasets.

The design of prompts plays a crucial role in activating the LLM's capabilities for downstream tasks. Following the findings of Webson and Pavlick (2021) on prompt design, we first assume a persona for the LLM, then provide task instructions and ask the model to generate outputs in a fixed style. For few-shot prompting, we provide task examples along with their corresponding outputs; details are shown in Figure 7. Interestingly, we observed that as the number and complexity of tasks in the instructions increased, the model's performance on the target task declined, as demonstrated in Table 1. The prompts presented here yielded the best performance among all the variations we tested.

B Preliminary Results

Following the approach of Hartmann et al. (2024), we initially attempted to use self-critique and rely on the in-context learning capabilities of the local LLM to implement the deferral function. Specifically, we instructed the model to handle the task while simultaneously outputting a confidence level, which would determine whether the query should be deferred to the server. However, preliminary results revealed limitations in this design. As shown in Table 3, without examples, the local model tends to be overly confident in every generated response. Moreover, even when provided with several examples, the model treats confidence as a classification task, rather than correlating it with the quality of its generated responses. Consequently, we opted to use logits for more effective LLM cascading. Further, as indicated in section A, as the number and the complexity of tasks within the instruction increase, the model tend to have worse performance on the downstream task. As such, we propose to decompose the tasks within the instruction to several tasks and use different heads to handle it for achieving LLM cascade.

C Supplementary Results

C.1 Supplementary Cascade Results

As shown in Figure 9, training-based methods have a direct impact on distinguishing between correct and incorrect answers using logits (i.e., the separation between the green and red areas). This aligns with the scatter distribution in Figure 10, further validating the necessity of training in LLM cascading. Additionally, the higher peak in the red area indicates a faster performance improvement, as depicted in Figures 4 and 8. These findings explain the effectiveness and intuition of our approach.

C.2 Logits Distribution Study

To further understand the effectiveness of our proposed LLM cascade with multi-objective considerations, we visualize the logit distributions for both training and training-free methods. As shown in Figure 10 and 9, the logits become more decentralized when a few examples are provided for $\Phi(L)$ to learn the cascade logic, in contrast to 0-shot prompting. Additionally, the signals within the distributions for prompting methods are not distinctly separable, which accounts for the randomness observed in routing queries, as discussed in previous sections. In contrast, training methods demonstrate more distinct distributions, where concentrated red points represent the reflection points noted in Figure 4. This indicates that training-based methods better grasp the cascade logic; answers with higher logits are correlated with more correct responses, suggesting that the trained $\Phi(L)$ is more confident in its correct answers and more likely to route difficult queries to the server. Furthermore, the trained model tends to send fewer unsafe queries to the server, as the logits for unsafe responses are generally higher, making them less likely to be sent. These observations reaffirm the effectiveness and necessity of incorporating multi-objective optimal considerations into cascading, highlighting the superiority of our proposed loss function for training the local LLM compared to existing prompting and instruction tuning methods.

Metric %	Cascade	Prompt Engineering				Instruction Tuning
		0-shot	1-shot	2-shot	5-shot	
Call Rate		0	70.43	48.98	67.43	42.76
Safe Call Rate		0	2.05	2.94	2.13	**27.61**
Accuracy	✗	14.94	10.08	**11.83**	10.68	**26.08**
	✓	14.94	42.91	37.30	42.61	42.29

Table 3: Preliminary results on GSM8K.

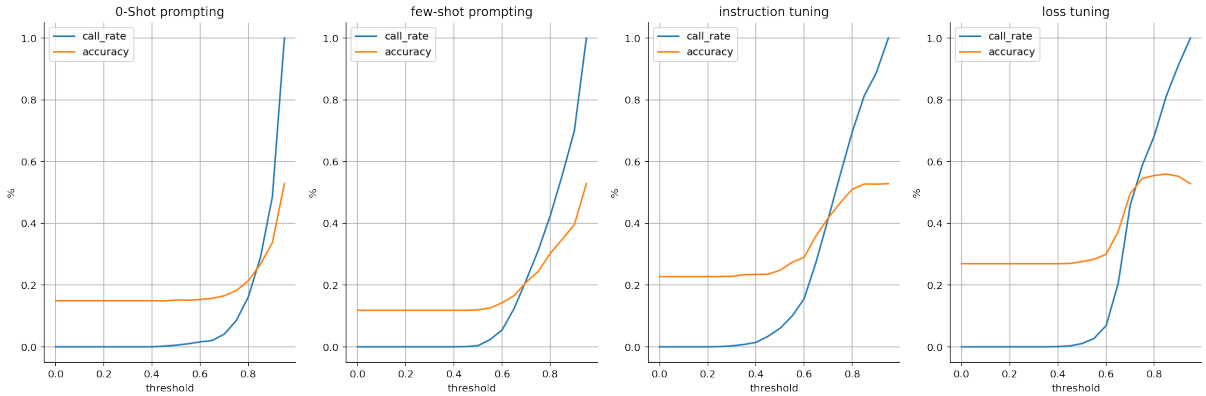


Figure 8: The curve of performance and call rate vs threshold on GSM8K dataset

C.3 Datasets

Table 4 provides detailed statistics for all datasets. Following the privacy research by Hartmann et al. (2024), we extracted tokens with privacy concerns (e.g., names and other personal identifiers), as the number of such privacy-leakage tokens is critical for evaluating our methods. The extraction was based on PII rules (Kim et al., 2024) and HIPAA regulations (Lincke, 2024), achieving extraction accuracies of 99.1% for GSM8K and 99.7% for MedQSum. A subset of 100 samples was manually verified by a highly educated PhD student, and the p-value score between human and machine extractions was less than 0.05, further validating the effectiveness of our proposed methods.

D Baseline Methodology

D.1 Multi-Objective In-context Learning

Ideally, the $\Phi(L)$ can be taught multi-objective optimal cascade logic based on its own natural language understanding ability. Efforts have been made to enable the $\Phi(L)$ being aware of the confidence of generated responses via self-critique(Zhu et al., 2024), step-by-step prompting(Zhang and Gao, 2023) etc. We step further on the previous works and include the privacy concern (Hartmann et al., 2024) into prompt design. Specifically, we formulate an instructional prompt[4] which integrates query x and objective considerations (i.e., privacy consideration obj_p) to the $\Phi(L)$ to obtain response $[y^{obj_p}, y^L]$, and these response will further be sent to the $D(\cdot)$ where deferral decisions will be made. Further, we follow Deng et al. (2024)'s work and perform few-shot prompting to better activate the $\Phi(L)$'s in-context learning ability. However, with limited size, the Φ is inadequate[5] to understand the multi-objective optimal cascade logic relying its own ability and the complicated logic may further hurt its ability to answer user's query and thus training is needed.

D.2 Multi-Objective Instruction Tuning

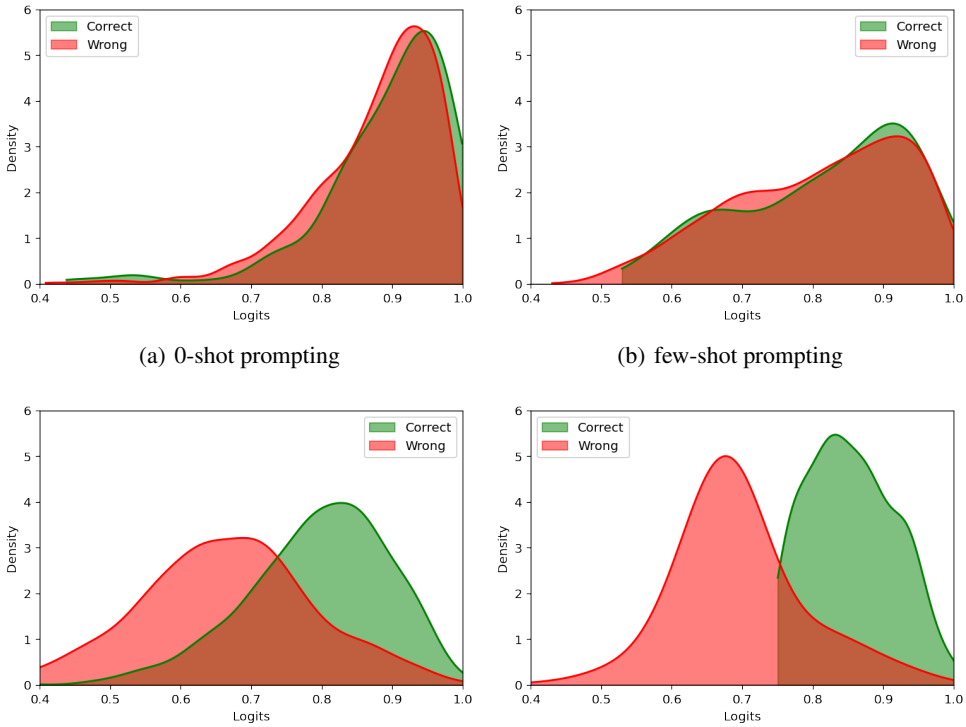
Previous studies have demonstrated the effectiveness of instruction tuning in enhancing downstream task performance and improving comprehension of given instructions (Zhu et al., 2024; Zhao et al., 2024; Ma et al., 2024; Li et al., 2023). This ability to understand instructions aligns well with our objective of grasping the deferral logic. Furthermore, the improvements in task performance help mitigate any negative impacts on generating y^L that may arise from producing y^{obj_i} during prompting. Similar to the prompting method, we utilize an instructional prompt that combines a step-by-step instruction with the user query x as input. The labeled text $\hat{y}$ corresponding to x, along with the labeled responses $\hat{y}^{obj_i}$ for the multi-objective considerations, serve as outputs for fine-tuning the model $\Phi(L)$. The responses generated by the tuned model will then be utilized by the deferral module $D(\cdot)$ to determine whether routing to the server model $\Phi(S)$ is necessary.

D.3 Multi-Objective Loss Tuning

Stepping further over the methods that rely on the local model's intricate understanding ability, recent works have pointed out the superiority of distilling the server llm's ability on downstream tasks into the loss function

[4]The prompts used can be seen in the appendix A

[5]Please refer to the appendix B for better understanding over the local llm's weakness.



(a) 0-shot prompting (b) few-shot prompting

(c) instruction tuning (d) loss tuning

Figure 9: Logits distribution curve by different methods on GSM8K dataset: (a) 0-shot prompting (b) few-shot prompting (c) instruction tuning (d) loss tuning.

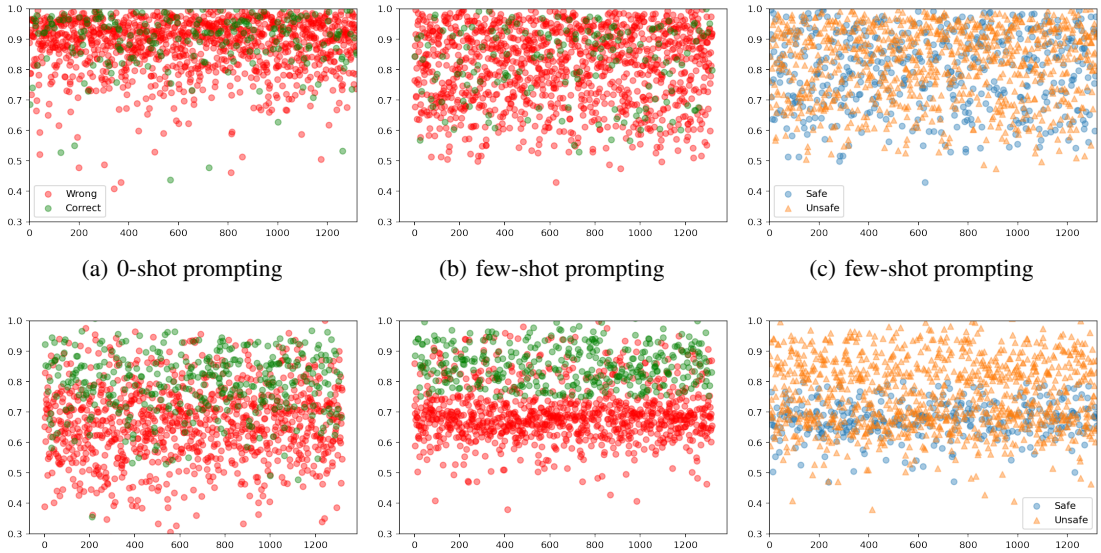
Dataset	GSM8K	MedSum	EmailSum
Avg. Input Length	52.56	70.51	223.2
Avg. Output Length	83.60	11.49	27.1
Avg. Leakage Tokens	5.19	11.27	49.77
Task Type	Question Answering	Summarization	Summarization
Measurement	Accuracy, Privacy Leakage	ROUGE, Privacy Leakage	ROUGE, Privacy Leakage

Table 4: Detailed type, statistics and measurement of datasets.

for tuning the local model(Wang et al., 2024). Intuitively, our assumption is that the server llm is larger and more powerful(Hartmann et al., 2024) in terms of down-stream tasks, and thus the discrepancy between the generations of $\Phi(L)$ and $\Phi(S)$ can somehow be used for $\Phi(L)$ to indicate the confidence level. The larger the discrepancy is, the lower confidence level should the $\Phi(L)$ have. However, to enable $\Phi(L)$ being aware of multi-objective considerations, simply including the distillation loss from $\Phi(S)$ is inadequate. To this end, we decompose the overall task into several sub-tasks and use different heads to handle the different sub-tasks. Namely, given the multi-objective considerations $[obj_1, ..., obj_i]$ and the query x, we leverage multiple llm heads $[h_1, ..., h_i, h_L]$ to handle different considerations and the query. Each head will produce a loss and a distillation loss from $\Phi(S)$ will be optionally added. These losses will then be sent to a weighted-sum function to produce a multi-objective cascade loss for tuning $\Phi(L)$:

$$l = \sum_{i}^{n} w_i \cdot l_{obj_i} + w_L \cdot l_L + \alpha(t) \cdot w_S \cdot l_S$$
$$\sum_{i}^{n} w_i^n + w_L + w_S = 1, \alpha(t) = H(logit_{y^L}, t) \tag{9}$$

where w_i denotes the weight for the loss associated with generating response y^{obj_i} for the objective obj_i, w_L is the weight for the loss of generating response y^L for x from $\Phi(L)$ and w_S is the weight for the loss of generating response y^S for x from $\Phi(S)$. n is the number of objectives that need to be considered. α is the factor for controlling if the knowledge from the server LLM $\Phi(S)$ is used depending on a logit threshold t. $H(\cdot, t)$ is a modified Heaviside Step function which returns 0 if $\cdot > t$ else returns 1. In the context of identifying privacy concern, the loss function we utilized for



(a) 0-shot prompting (b) few-shot prompting (c) few-shot prompting

(d) instruction tuning (e) loss tuning (f) loss tuning

Figure 10: Logits scatter distribution produced by different methods on GSM8K dataset. (e) and (f) are logits for privacy concerns; y-axis is the logits, x-axis is the data index.

tuning $\Phi(L)$ is:

$$l = -w_p \cdot (\hat{y}^p \cdot log(p_L(y^p|x)) + (1-\hat{y}^p) \cdot log(1-p_L(y^p|x))) + w_L \cdot log(p_L(y^L|x)) + \alpha(t) \cdot w_S \cdot log(p_S(y^S|x)) \tag{10}$$

where y^p, $\hat{y}^p$ are the predicted, golden binary predictions for privacy, respectively. Other terms remain the same as in formula 9. By incorporating multi-objective considerations into the loss function for tuning $\Phi(L)$, the model will generate answers with better awareness of these considerations. The corresponding logits of the generated answers by tuned $\Phi(L)$ can then be utilized by the deferral module to inform decision-making.

D.4 Deferral Module

All the three methods are studying how to enable the local LLM to be aware of multi-objective considerations while generating the response to the query. And such considerations are presented as the logit distributions of the generated response, for example, higher logit may indicated higher performance and less privacy concern. Deferral module plays a pivotal role in the LLM cascade since it decides which query to send out to the server llm based on the logits. Following previous successes on using different logit (e.g., mean, quantile) of the generated response as the reference to decide if there is a need to route the query to the server LLM(Wang et al., 2024; Jitkrittum et al., 2024; Gupta et al., 2024), we also utilize the logit of generated response as indicators to make the routing decisions. Specifically, given a threshold $t \in (0, 1)$, if the logit of the generated response exceed t then it means the local LLM is confident with its response and no need to route, otherwise route the query x to the server LLM $\Phi(S)$.

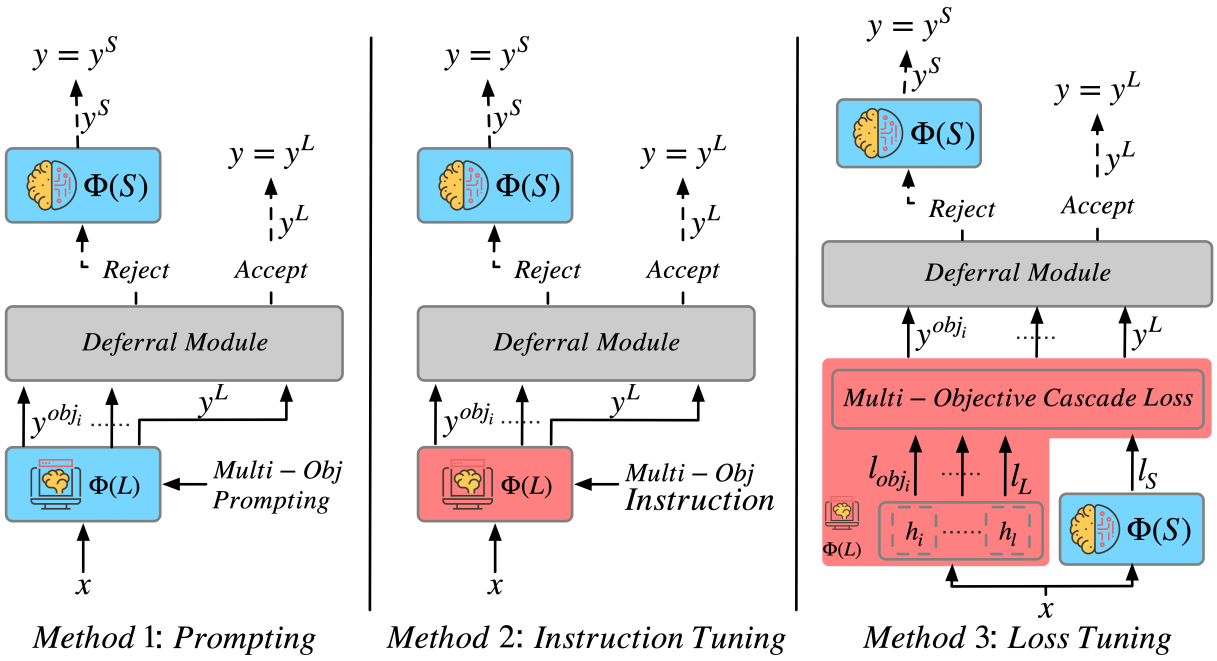


Figure 11: Overview of existing LLM cascade methods: $\Phi(L)$ and $\Phi(S)$ represent the local model and server model, respectively. The red box indicates trainable, while the blue box represents frozen. $\Phi(L)$ is tasked with generating responses y^L and y^{obj_i} for both the query x and the multi-objective considerations obj_i. For loss tuning, the generation tasks are handled by different heads h_i, and a combined cascade loss is utilized for tuning.