TRIAGE: TOKEN-LEVEL LANGUAGE MODEL CASCADES WITH UTILITY-AWARE ENTROPY CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have achieved strong performance on complex reasoning tasks, but their growing scale leads to substantial inference costs. Language model cascades mitigate this issue by first using a small language model (SLM) for low-cost generation and selectively invoking a large language model (LLM), aiming to achieve a trade-off between model performance and inference cost. However, existing cascade systems mostly operate at the sequence level and rely on SLM uncertainty as the deferral signal, leading to coarse-grained decisions and a mismatch between uncertainty and deferral utility. In this paper, we propose **Triage**, a token-level cascade framework with utility-aware entropy calibration (UEC). Triage performs token-level deferral decisions during autoregressive generation and jointly considers the prediction outcomes of the SLM and LLM to calibrate SLM entropy, so that it no longer only reflects model uncertainty, but also better captures token-level deferral utility. Experiments on two model pairs from different families and scales across four reasoning datasets show that Triage achieves a superior cost–performance trade-off over various cascade baselines while preserving the downstream task performance of the SLM.

1 INTRODUCTION

According to scaling laws, increasingly large language models (LLMs) have demonstrated stronger capabilities on various complex reasoning tasks (Kaplan et al., 2020; Yang et al., 2025; Brown et al., 2020; Costa-Jussà et al., 2022; Shao et al., 2024; Guo et al., 2025). However, the growth of model scale is also accompanied by high inference costs. In practical applications, not all queries need to be handled by expensive high-performance models (Ong et al., 2025). Based on this observation, language model cascades have been proposed to trade off model performance and inference cost (Chen et al., 2023; Jiang et al., 2026; Fanconi & van der Schaar, 2026).

In a two-level cascade system, a low-cost small language model (SLM) first answers the query, and a deferral rule decides whether to retain the SLM response or defer the query to a large language model (LLM) for re-answering (Gupta et al., 2024; Warren & Dras, 2025). Existing systems have explored various deferral rules, including predictive entropy and confidence (Rabanser et al., 2026; Gupta et al., 2024; Shen et al., 2025), consistency across multiple samples (Soiffer et al., 2025; Yue et al., 2024), and self-verification (Aggarwal et al., 2024). These methods typically estimate SLM uncertainty to identify potentially erroneous responses. Despite progress in deferral-rule design, existing cascades mostly remain sequence-level and rely on SLM uncertainty as the deferral signal, leading to two key limitations.

First, sequence-level cascades are too coarse-grained. As shown in Figure 1, our analysis on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) using Qwen3-0.6B (Yang et al., 2025) as the SLM shows that, although the SLM fails on many samples, most generated tokens are still correctly predicted, indicating that difficulty is unevenly distributed within a response. However, existing cascades defer the entire query once the SLM response is judged unreliable (Bai et al., 2024; Chen et al., 2023), wasting useful SLM generation. Meanwhile, aggregating token-level uncertainty into a sequence-level signal can dilute the uncertainty of difficult tokens (Gupta et al., 2024; Malinin & Gales, 2020). Consistent with this, Figure 1 shows that token entropy provides a clearer

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

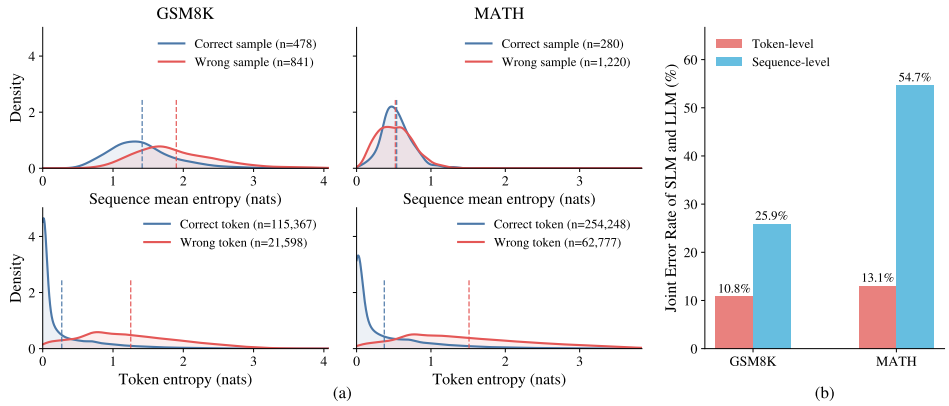


Figure 1: (a) Entropy distributions of Qwen3-0.6B on GSM8K and MATH. Although many responses generated by the SLM are incorrect, most tokens are still correctly predicted, and token entropy separates correct and incorrect predictions more clearly than sequence mean entropy. (b) Joint error rates of Qwen3-0.6B and Qwen3-8B on GSM8K and MATH. Many errors cannot be corrected by deferring to the LLM, especially at the sequence level.

separation between correct and incorrect predictions than sequence-level entropy. These observations motivate extending sequence-level cascades to the token level. **Second, SLM uncertainty is not fully aligned with the actual needs of the cascade system.** Existing deferral rules usually rely on the SLM's own uncertainty (Gupta et al., 2024; Warren & Dras, 2025; Rabanser et al., 2026), but the benefit of cascading depends not only on whether the SLM makes an error, but also on whether the LLM can correct it. Thus, deferral is useful only for errors that the LLM can correct; when both models err, deferral only adds cost. As shown in Figure 1, such joint errors are not rare, especially at the sequence level, indicating a mismatch between SLM uncertainty and deferral utility.

To address the above issues, we propose **Triage**, a token-level cascade framework with utility-aware entropy calibration (UEC) to optimize the deferral signal. Triage makes deferral decisions at each decoding step according to the SLM's predictive entropy, retaining low-entropy tokens and deferring high-entropy tokens to the LLM. To better align SLM entropy with the actual needs of the cascade system, UEC adjusts it based on the joint prediction outcomes of the SLM and LLM. Specifically, Triage categorizes tokens according to whether they are correctly predicted by the SLM and whether LLM intervention can correct them. It then calibrates different token categories with different objectives, promoting useful deferrals while preserving SLM task performance and suppressing ineffective deferrals. In this way, Triage transforms the token-level predictive entropy of the SLM from a simple uncertainty measure into a deferral signal that better reflects deferral utility. Experiments on two model pairs from different families and scales across four reasoning datasets demonstrate that Triage outperforms various cascade baselines with lower inference cost, achieving a superior cost–performance trade-off. Ablations and analyses further confirm the importance of both token-level deferral and UEC, as well as the stability of the calibration mechanism.

Our contributions are summarized as follows:

- We propose **Triage**, a token-level cascade framework that makes deferral decisions during autoregressive generation and supports efficient inference with incremental KV-cache maintenance.
- We introduce UEC, which calibrates SLM entropy based on the joint prediction outcomes of the SLM and LLM, transforming it into a deferral signal aligned with LLM intervention utility.
- Experiments across multiple model pairs and reasoning datasets show that Triage achieves a better cost–performance trade-off than existing cascade baselines, while preserving the downstream task performance of the SLM.

2 TRIAGE

Figure 2 provides an overview of Triage. During training, Triage performs UEC on the SLM by calibrating its predictive entropy according to the joint prediction outcomes of the SLM and LLM, so that the entropy better reflects the utility of LLM intervention. During inference, the calibrated entropy is used as the token-level deferral signal: at each decoding step, Triage retains the SLM token with low entropy, and invokes the LLM to generate the current token with high entropy. In this way, Triage extends language model cascades to the token level and enables more efficient use of the LLM.

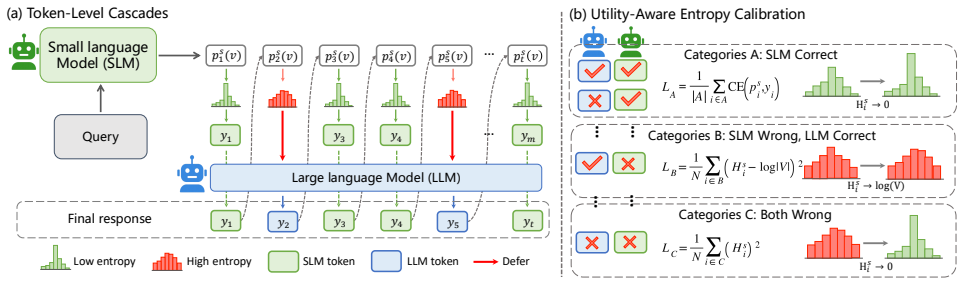


Figure 2: Overview of Triage. (a) Triage performs token-level cascading, retaining low-entropy SLM tokens while deferring high-entropy tokens to the LLM. (b) UEC partitions tokens into three groups according to the joint prediction outcomes of the SLM and LLM, and calibrates their entropy to promote useful deferrals and suppress ineffective ones.

2.1 TOKEN-LEVEL CASCADES

Conventional cascade methods first let the SLM generate a complete response and then decide whether to defer the entire query to the LLM based on sequence-level uncertainty, such as the average token entropy (Rabanser et al., 2026) or the product of token-level confidence scores over the generated sequence (Gupta et al., 2024). Such coarse-grained cascading can waste useful SLM computation, since many tokens in an otherwise unreliable response may still be correct. Moreover, aggregating uncertainty over the whole sequence can dilute the signals from a small number of difficult tokens.

Triage instead extends cascading to the token level, making a deferral decision at each decoding step, as shown in Figure 2. The system consists of an SLM f_s and an LLM f_l that share the same tokenizer and vocabulary V, allowing tokens generated by either model to be directly appended to a shared generated sequence. This shared-tokenizer assumption simplifies token-level interaction, but also limits the direct applicability of Triage to tokenizer-compatible model pairs.

Given a query x, generation starts from the SLM. At step t, conditioned on the previously generated prefix $y_{<t} = (y_1, \ldots, y_{t-1})$, the SLM produces

$$p_t^s(v) = p_s(v \mid x, y_{<t}), \qquad H_t^s = -\sum_{v \in V} p_t^s(v) \log p_t^s(v), \qquad v \in V.$$

Given a predefined deferral threshold τ, Triage retains the SLM prediction if $H_t^s < \tau$, and otherwise invokes the LLM, which produces

$$p_t^l(v) = p_l(v \mid x, y_{<t}), \qquad v \in V.$$

The token at step t is then selected as

$$y_t = \begin{cases} \arg\max_{v \in V} p_t^s(v), & H_t^s < \tau, \\ \arg\max_{v \in V} p_t^l(v), & H_t^s \geq \tau. \end{cases}$$

The selected token is appended to the shared sequence and used for subsequent decoding.

To avoid repeatedly processing the same prefix, the SLM and LLM maintain separate KV caches C_s and C_l. The LLM cache is updated only when deferral occurs: it incrementally processes the prefix

segment not yet included in C_l, updates the cache, and then generates the current token. When an LLM token is accepted, the SLM cache is also updated with this token, ensuring that both models remain conditioned on the same shared sequence.

2.2 UTILITY-AWARE ENTROPY CALIBRATION

In Triage, predictive entropy serves as the token-level deferral signal, reflecting the SLM's uncertainty at the current position. However, deferring an uncertain token does not necessarily improve prediction, since the LLM may also err at the same position. Thus, among erroneous SLM predictions, only those that can be corrected by the LLM truly benefit from deferral. To align uncertainty with cascade utility, Triage introduces UEC, which calibrates the SLM based on the joint prediction outcomes of the SLM and LLM, as illustrated in Figure 2.

After the SLM and LLM are adapted to the downstream task via standard supervised fine-tuning (SFT), Triage performs UEC on the SLM. We denote a calibration batch as

$$\mathcal{B} = \left\{ (x_n, y^*_{n,i}) \mid n = 1, \ldots, K,\ i = 1, \ldots, M_n \right\},$$

where K is the number of samples and M_n is the number of valid tokens in the ground-truth response of the n-th sample. For each $(x_n, y^*_{n,i}) \in \mathcal{B}$, the SLM and LLM perform forward propagation to obtain their predictions. The SLM predictive distribution, prediction, and entropy are

$$p^s_{n,i}(v) = p_s(v \mid x_n, y^*_{n,<i}), \qquad \hat{y}^s_{n,i} = \arg\max_{v \in V} p^s_{n,i}(v), \qquad H^s_{n,i} = -\sum_{v \in V} p^s_{n,i}(v) \log p^s_{n,i}(v),$$

and the LLM prediction is

$$\hat{y}^l_{n,i} = \arg\max_{v \in V} p_l(v \mid x_n, y^*_{n,<i}).$$

Based on the joint prediction outcomes, Triage partitions valid tokens into three categories:

$$\begin{aligned}
A &= \left\{ (n,i) \mid (x_n, y^*_{n,i}) \in \mathcal{B},\ \hat{y}^s_{n,i} = y^*_{n,i} \right\}, \\
B &= \left\{ (n,i) \mid (x_n, y^*_{n,i}) \in \mathcal{B},\ \hat{y}^s_{n,i} \neq y^*_{n,i},\ \hat{y}^l_{n,i} = y^*_{n,i} \right\}, \\
C &= \left\{ (n,i) \mid (x_n, y^*_{n,i}) \in \mathcal{B},\ \hat{y}^s_{n,i} \neq y^*_{n,i},\ \hat{y}^l_{n,i} \neq y^*_{n,i} \right\}.
\end{aligned}$$

Category A contains tokens correctly predicted by the SLM, Category B contains tokens incorrectly predicted by the SLM but correctly predicted by the LLM, and Category C contains tokens incorrectly predicted by both models.

Since Category A typically accounts for the majority of tokens, Triage applies the standard cross-entropy objective to preserve the downstream task performance of the SLM and maintain low entropy for these predictions:

$$L_A = \frac{1}{|A|} \sum_{(n,i) \in A} \mathrm{CE}\left(p^s_{n,i}, y^*_{n,i}\right).$$

For Category B, whose tokens can be corrected after deferral, Triage encourages high entropy to promote useful deferrals:

$$L_B = \frac{1}{N} \sum_{(n,i) \in B} \left(H^s_{n,i} - \log |V|\right)^2,$$

where $\log |V|$ is the maximum entropy over the vocabulary V, and $N = \sum_{n=1}^{K} M_n$ is the total number of valid tokens in the batch. For Category C, where deferral provides no benefit because both models err, Triage encourages low entropy to avoid ineffective deferrals:

$$L_C = \frac{1}{N} \sum_{(n,i) \in C} \left(H^s_{n,i}\right)^2.$$

The loss terms for Categories B and C are normalized by N because their per-token calibration losses can be relatively large. In particular, overconfident low-entropy tokens in Category B and high-entropy tokens in Category C can yield losses close to $(\log |V|)^2$. By contrast, for tokens in Category A, the ground-truth token is the top-1 SLM prediction, implying $p^s_{n,i}(y^*_{n,i}) \geq 1/|V|$, so the

cross-entropy loss is upper-bounded by $\log |V|$. Normalizing L_B and L_C by N therefore stabilizes the calibration strength and prevents excessive entropy adjustment from degrading the downstream task capability of the SLM.

The resulting UEC loss is

$$L_{\text{UEC}} = \lambda_A L_A + \lambda_B L_B + \lambda_C L_C,$$

where λ_A, λ_B, and λ_C balance the three calibration objectives according to the relative prevalence of the three token categories. Through UEC, Triage encourages high entropy where LLM intervention is beneficial and suppresses unnecessary deferrals where the LLM provides no additional utility, producing a deferral signal better aligned with the requirements of token-level cascades.

3 Experiment

3.1 Experimental setting

Models and datasets. We use two model pairs from different families, including Llama-3.2-1B-Instruct with Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-0.6B with Qwen3-8B (Yang et al., 2025), where the smaller model serves as the SLM and the larger model serves as the LLM. The evaluation datasets include GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), MBPP (Austin et al., 2021), and MMLU (Hendrycks et al., 2020), covering mathematical reasoning, code generation, and scientific reasoning tasks.

Evaluation metrics. We use accuracy to evaluate task performance on GSM8K, MATH, and MMLU, and Pass@1 to evaluate code generation performance on MBPP. To evaluate the cost–performance trade-off, following prior cascade studies (Warren & Dras, 2025; Soiffer et al., 2025; Chuang et al., 2024; Rabanser et al., 2026; Gupta et al., 2024), we plot deferral curves, where the x-axis denotes the deferral rate and the y-axis denotes task performance. To fairly compare token-level methods with sequence-level baselines, we use token-level deferral rate as the unified x-axis: for token-level methods, it is the proportion of final output tokens generated by the LLM, while for sequence-level methods, sample-level deferrals are converted according to the lengths of the final outputs. We further use AUC-DF, i.e., the area under the deferral curve, to summarize the overall performance across different deferral rates. Since the token-level deferral rate only approximates LLM utilization, we additionally report end-to-end latency–performance curves that capture the full inference overhead, including incremental prefix processing and KV-cache maintenance. The detailed computation of deferral rates, AUC-DF, and latency is provided in the appendix.

Baselines. We compare Triage with sequence-level cascade baselines, including Gatekeeper (Rabanser et al., 2026), Self-REF (Chuang et al., 2024), and Semantic Agreement (Soiffer et al., 2025). We also include CoLLM (Shen et al., 2024) as a token-level multi-model collaboration baseline. For fair comparison, we extend Gatekeeper to the token level using predictive entropy for per-token deferral, and include an SFT-based token-level cascade without UEC and a random deferral baseline.

3.2 Main results

Triage achieves superior overall cost–performance trade-offs. Table 1 reports the AUC-DF results of different methods on two model pairs and four reasoning datasets. Since AUC-DF integrates task performance over the full deferral curve, it provides an overall measure of performance under different deferral rates. Triage achieves the highest average AUC-DF on both model families: 0.5015 on the Llama model pair and 0.5800 on the Qwen model pair, outperforming the strongest competing baseline, the SFT-based token-level cascade, which obtains 0.4821 and 0.5714, respectively. At the task level, Triage achieves the best AUC-DF on all four datasets with the Llama model pair. With the Qwen model pair, Triage achieves the best results on MATH and MBPP, while remaining close to the best-performing methods on GSM8K and MMLU. In particular, on Qwen-GSM8K, Triage performs very close to token-level Gatekeeper; on Qwen-MMLU, the gaps among different methods are relatively small, possibly because MMLU typically requires generating only a single option letter, leaving limited room for token-level deferral. Overall, these results show that Triage achieves stable and superior cost–performance trade-offs across different model families and task types.

Deferral curves further show the advantage of Triage under limited LLM utilization. Beyond the AUC-DF comparison, Figure 3 presents the complete deferral curves of different methods.

Table 1: Comparison of AUC-DF across different deferral methods. The best result in each row is shown in bold and the second-best is underlined. Random deferral and LLM-only are excluded from the ranking and serve only as lower- and upper-bound references, respectively. SFT-T, Gate., Gate.-T, and Sem. Agr. denote the SFT-based token-level cascade, Gatekeeper, token-level Gatekeeper, and Semantic Agreement, respectively.

Dataset	Triage	Gate.	Self-REF	Sem. Agr.	CoLLM	Gate.-T	SFT-T	Random	LLM
Llama									
GSM8K	**0.6643**	0.5338	0.5642	0.6303	0.5880	0.6314	0.6350	0.5284	0.7096
MATH	**0.3840**	0.3023	0.2892	0.3315	0.3176	0.3603	0.3715	0.3053	0.4127
MBPP	**0.3882**	0.3388	0.3318	0.3758	0.3430	0.3803	0.3792	0.3293	0.4112
MMLU	**0.5696**	0.5436	0.5558	0.5540	0.5687	0.5436	0.5426	0.5349	0.6271
Avg.	**0.5015**	0.4296	0.4353	0.4729	0.4543	0.4789	0.4821	0.4245	0.5402
Qwen									
GSM8K	0.7612	0.6712	0.7030	0.6942	0.6836	**0.7645**	0.7564	0.6422	0.8105
MATH	**0.4831**	0.3899	0.3902	0.3820	0.4076	0.4504	0.4669	0.3787	0.5260
MBPP	**0.4542**	0.4233	0.4079	0.4122	0.4098	0.4529	0.4510	0.3972	0.4930
MMLU	0.6214	0.6116	0.6170	**0.6285**	0.6215	0.6116	0.6111	0.5993	0.7317
Avg.	**0.5800**	0.5240	0.5295	0.5292	0.5306	0.5699	0.5714	0.5044	0.6403

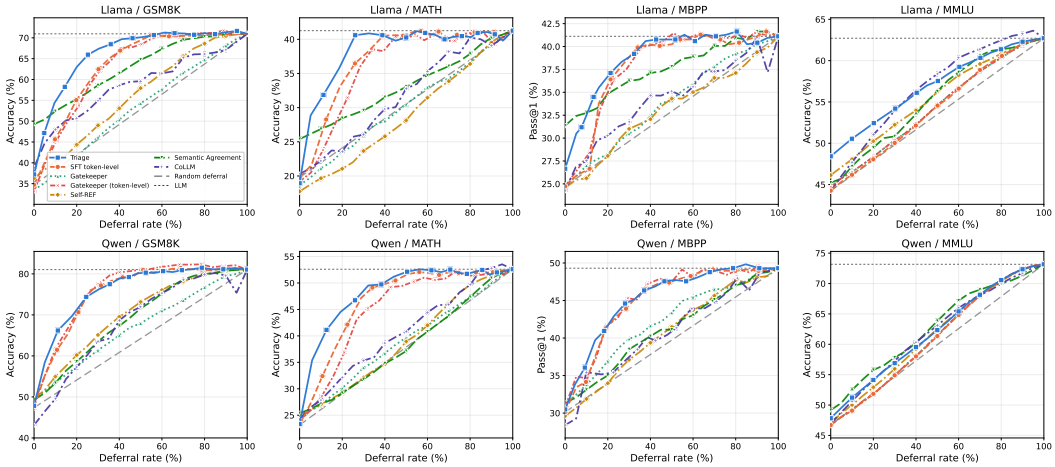


Figure 3: Deferral curves across two model pairs and four reasoning datasets. The x-axis denotes the deferral rate, and the y-axis denotes task performance. Triage achieves a better cost–performance trade-off than competing methods, with particularly strong gains in the low-deferral-rate region, where LLM calls are limited.

Across most model pairs and datasets, Triage generally achieves higher task performance than competing methods, with more pronounced advantages in the low- and medium-deferral-rate regions. This indicates that when LLM utilization is limited, Triage can more accurately select token positions for deferral, allocating limited LLM computation to more valuable positions and achieving higher task performance with a lower deferral rate. As the deferral rate increases, different methods gradually approach the LLM-only upper bound, which is consistent with the expected behavior of cascade systems under high deferral rates.

Token-level deferral generally outperforms sequence-level deferral. Table 1 and Figure 3 also show that token-level methods, including Triage, CoLLM, token-level Gatekeeper, and the SFT-based token-level cascade, generally outperform sequence-level cascade baselines. This trend is especially clear on GSM8K, MATH, and MBPP, where models generate longer reasoning processes or code sequences. These results suggest that token-level deferral can better exploit predictive entropy at each generation position than sequence-level aggregation, thereby more directly reflecting the SLM's uncertainty during generation. Among token-level methods, Triage further optimizes the deferral signal through UEC and achieves the best average AUC-DF on both model families.

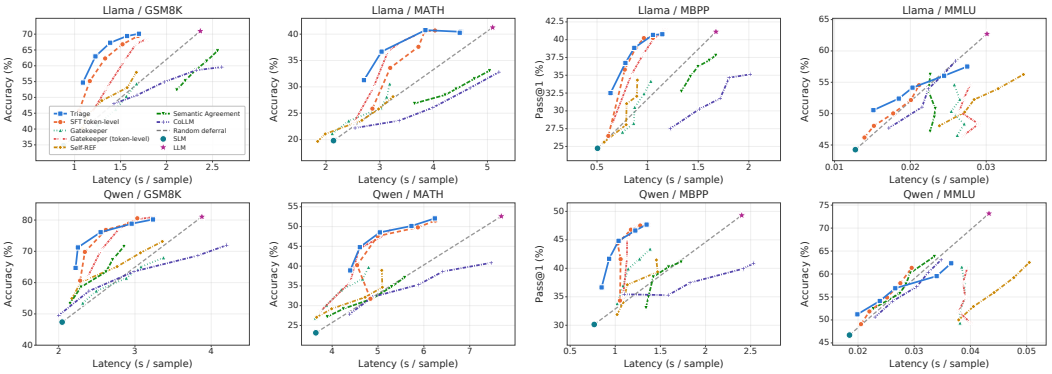


Figure 4: Latency vs. performance curves across two model pairs and four reasoning datasets. The x-axis denotes average latency per sample, and the y-axis denotes task performance. Triage generally provides a more favorable Pareto frontier than competing methods, suggesting that the proposed token-level cascade and UEC can improve practical inference efficiency in addition to deferral-rate-based evaluation.

Triage also improves latency vs. performance behavior. In addition to deferral-rate-based evaluation, Figure 4 reports the latency vs. performance curves of different methods. Across most datasets and model pairs, Triage provides a more favorable Pareto frontier than competing methods, often achieving higher task performance at similar latency levels or comparable performance with lower latency. This advantage is particularly clear on GSM8K, MATH, and MBPP, where token-level deferral can more effectively allocate LLM computation to difficult generation positions. On MMLU, the gap is relatively smaller, consistent with its short-answer format and the limited room for token-level deferral. Overall, these results suggest that the deferral-rate advantage of Triage can also translate into practical latency–performance gains.

3.3 Effect of Utility-Aware Entropy Calibration

To analyze the effect of UEC, we first compare the entropy distributions of the three token categories before and after calibration. Figure 5 presents the results on GSM8K and MATH with two model pairs. Before calibration, the three categories are not clearly separated, especially Categories B and C, indicating that the original SLM entropy struggles to identify errors that can benefit from LLM intervention. After applying UEC, Category A remains concentrated in the low-entropy region, Category B shifts toward higher entropy, and Category C stays relatively lower than Category B. These results show that UEC makes predictive entropy better reflect token-level deferral utility rather than merely the SLM's own uncertainty.

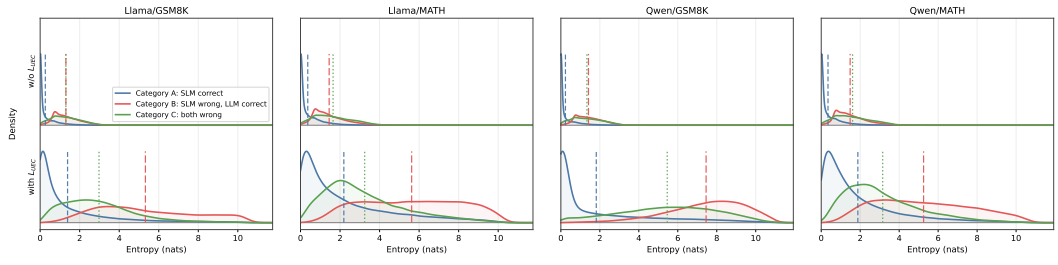


Figure 5: Entropy distributions before and after UEC on GSM8K and MATH with two model pairs. The upper row shows the results without UEC, while the lower row shows the results after applying calibration. After calibration, the three categories become more clearly separated, with entropy distributions moving consistently with the calibration objectives.

To further verify whether high-entropy tokens correspond to useful deferrals, we evaluate useful deferral precision (UDP), which measures the proportion of selected high-entropy tokens belonging

to Category B. Table 2 reports UDP results on the Llama model pair at fixed token-level deferral budgets, while the more results are provided in the appendix.

As shown in Table 2, Triage achieves the highest UDP@10 in all four Llama settings, showing that UEC is especially effective when the token-level deferral budget is limited. Triage also obtains the best UDP@20 in 3 out of 4 settings and the best UDP@30 in 2 out of 4 settings. The remaining cases occur on GSM8K and MATH under larger budgets, where the SFT-based token-level cascade slightly outperforms Triage. This suggests that UDP should be interpreted budget by budget rather than as uniform dominance across all deferral rates. Overall, the entropy distribution and UDP results jointly show that UEC improves the alignment between predictive entropy and token-level deferral utility.

Table 2: UDP results on the Llama model pair at fixed token-level deferral budgets. UDP measures the proportion of selected high-entropy tokens that belong to Category B, i.e., tokens incorrectly predicted by the SLM but correctly predicted by the LLM. The best result in each row and each budget is shown in bold. SFT-T and Gate.-T denote the SFT-based token-level cascade and token-level Gatekeeper, respectively. Higher is better.

Dataset	Triage			Gate.-T			SFT-T		
	UDP@10	UDP@20	UDP@30	UDP@10	UDP@20	UDP@30	UDP@10	UDP@20	UDP@30
GSM8K	**0.1866**	0.1572	0.1318	0.1804	0.1652	0.1471	0.1826	**0.1683**	**0.1476**
MATH	**0.2463**	**0.1992**	0.1696	0.1780	0.1796	0.1714	0.1796	0.1829	**0.1741**
MBPP	**0.2163**	**0.1970**	**0.1699**	0.1799	0.1792	0.1659	0.1811	0.1797	0.1665
MMLU	**0.3376**	**0.3348**	**0.3318**	0.3105	0.3095	0.3114	0.3034	0.3084	0.3071

3.4 Ablation study

Table 3 reports the ablation results of Triage on GSM8K with the Llama model pair. Removing token-level deferral leads to a clear performance drop, showing that the improvement of Triage relies on making fine-grained deferral decisions during generation rather than falling back to sequence-level deferral. Removing the complete UEC loss also degrades performance, indicating that token-level cascading alone is insufficient and that the SLM entropy needs to be calibrated toward deferral utility.

Table 3: Ablation study of Triage on GSM8K with the Llama model pair. Δ denotes the change in AUC-DF relative to the full Triage model.

	Triage	w/o token-level deferral	w/o L_A	w/o L_B	w/o L_C	w/o L_{UEC}
AUC-DF	**0.6643**	0.5575	0.5264	0.6309	0.6470	0.6350
Δ	–	-0.1068	-0.1379	-0.0335	-0.0174	-0.0293

Among the three calibration terms, removing L_A causes the largest drop, even larger than removing token-level deferral. This suggests that preserving the downstream task capability of the SLM is crucial during calibration, especially when the deferral budget is limited. The additional deferral curves in Appendix Figure 6 further support this observation. Removing L_B or L_C also reduces AUC-DF, confirming that both promoting useful deferrals and suppressing ineffective deferrals contribute to the final cost–performance trade-off.

3.5 Sensitivity analysis of calibration weights.

To examine whether the weighting coefficients in UEC affect the downstream task capability of the SLM, we conduct a grid search on GSM8K with the Llama model pair by varying λ_A, λ_B, and λ_C. Specifically, each weight is selected from $\{0.7, 0.8, 0.9, 1.0\}$, resulting in 64 parameter combinations. For each setting, we evaluate the downstream task accuracy of the calibrated SLM and report the results in Appendix Figure 8. The results show that the SLM accuracy remains stable across different weight combinations, and all settings outperform the SLM trained only with standard SFT, which achieves an accuracy of 34.72%. This indicates that UEC does not degrade the

downstream task capability of the SLM; instead, it consistently improves the SLM performance under a wide range of parameter settings. Moreover, the average accuracies under different λ_A values are 38.11%, 38.18%, 38.51%, and 38.55%, respectively, showing only small variations. Overall, these results suggest that the proposed calibration objective is not overly sensitive to the choice of weighting coefficients and can stably preserve the basic generation capability of the SLM.

4 Related Work

4.1 Language Model Cascades

Language model cascades balance inference performance and computational cost by deferring only necessary queries from the SLM to the LLM. Existing studies design deferral signals from different perspectives. Quality-estimation-based methods assess the current response and invoke stronger models when needed, such as the learned quality estimator in FrugalGPT (Chen et al., 2023) and the iterative cascading framework in Dekoninck et al. (2024). Other methods rely on confidence or self-verification, including self-verification confidence for cascading (Aggarwal et al., 2024) and explicit confidence tokens for deferral (Chuang et al., 2024). Agreement-based methods use consistency among multiple predictions to decide escalation (Kolawole et al., 2024), with semantic agreement further extending this idea to open-ended generation (Soiffer et al., 2025). Gupta et al. (2024) studies token-level uncertainty for constructing deferral rules, although the final deferral decision remains sequence-level. More fine-grained collaboration has also been explored by Shen et al. (2024), who dynamically selects whether each token is generated by a base or auxiliary model. In contrast, Triage extends cascading itself to the token level without an additional learned router, directly using the SLM's uncertainty at each generation position to decide whether the current token should be deferred to the LLM.

4.2 Uncertainty Estimation and Calibration

Uncertainty estimation reflects the reliability of a language model's own predictions and is widely used as a deferral signal in cascades: when the smaller model is uncertain, its prediction is deferred to a stronger model (Aggarwal et al., 2024; Chuang et al., 2024). Existing approaches estimate uncertainty using token probabilities (Huang et al., 2025; Malinin & Gales, 2020), predictive entropy (Vazhentsev et al., 2023), semantic entropy (Kuhn et al., 2023; Manakul et al., 2023), and self-verification (Kadavath et al., 2022; Zhao et al., 2024). Since cascade performance depends on whether uncertainty can identify unreliable SLM predictions, calibration has been used to improve confidence estimates. Conventional calibration methods, such as temperature scaling, label smoothing, and Sequence Likelihood Calibration, adjust model confidence at the token or sequence level (Guo et al., 2017; Wang et al., 2020). In cascading systems, Rabanser et al. (2026) fine-tunes the smaller model to assign higher confidence to correct predictions and lower confidence to erroneous ones. Unlike these methods, which calibrate confidence mainly according to the SLM's own prediction correctness, Triage jointly considers the prediction outcomes of the SLM and LLM, allowing uncertainty to reflect whether LLM intervention is beneficial at the current token position and better aligning the deferral signal with cascade utility.

5 Conclusion

In this paper, we proposed **Triage**, a token-level language model cascade framework with UEC. Triage makes deferral decisions at each decoding step based on the SLM's predictive entropy, enabling finer-grained use of LLM computation while avoiding signal dilution caused by sequence-level uncertainty aggregation. To align the deferral signal with cascade utility, UEC adjusts SLM entropy according to the joint prediction outcomes of the SLM and LLM, promoting useful deferrals and suppressing ineffective ones. Experiments on two model pairs and four reasoning datasets show that Triage achieves a better cost–performance trade-off than existing cascade baselines, especially in the low-deferral-rate region. Ablation studies and additional analyses further confirm the importance and stability of the proposed design. Overall, Triage provides an effective approach for building more efficient and fine-grained language model cascade systems. One limitation of the current framework is that it assumes a shared tokenizer between the SLM and LLM; extending Triage to model pairs with heterogeneous tokenizers is an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for research methodology discussions, experimental design, code development, drafting the paper outline, and language polishing. We have reviewed all AI-assisted work and verified the resulting content before inclusion in the paper. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide a detailed description of the core algorithmic design of Triage in the main text, including the inference procedure of token-level deferral, the token categorization strategy of UEC, and the corresponding training objectives. The experimental section reports the model pairs, datasets, evaluation metrics, baseline methods, and the computation of AUC-DF. In the appendix, we further provide the main implementation details and hyperparameter settings, including the training configurations for the SFT and calibration stages, the calibration weights, the computation of deferral curves and AUC-DF, and additional experimental results across different model pairs and datasets. We also include token category statistics, additional entropy distribution analyses, sensitivity analysis of calibration weights, and comparisons with longer SFT training to help verify the main conclusions of this work. All experiments are conducted using publicly available models and datasets, and the code and experimental configurations will be provided in the supplementary materials.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*, 2024.

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

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

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

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

Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for llms. *arXiv preprint arXiv:2410.10347*, 2024.

Claudio Fanconi and Mihaela van der Schaar. Cascaded language models for cost-effective human–ai decision-making. *Advances in Neural Information Processing Systems*, 38:11559–11593, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

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

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *International Conference on Learning Representations*, volume 2024, pp. 4147–4180, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. Look before you leap: An exploratory study of uncertainty analysis for large language models. *IEEE Transactions on Software Engineering*, 51(2):413–429, 2025.

Youhe Jiang, Fangcheng Fu, Wanru Zhao, Stephan Rabanser, Jintao Zhang, Nic Lane, and Binhang Yuan. Cascadia: An efficient cascade serving system for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 112830–112849, 2026.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Steven Kolawole, Don Dennis, Ameet Talwalkar, and Virginia Smith. Agreement-based cascading for efficient inference. *arXiv preprint arXiv:2407.02348*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*, 2020.

Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 9004–9017, 2023.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Stephan Rabanser, Nathalie Rauschmayr, Achin Kulshrestha, Petra Poklukar, Wittawat Jitkrittum, Sean Augenstein, Congchao Wang, and Federico Tombari. Gatekeeper: Improving model cascades through confidence tuning. *Advances in Neural Information Processing Systems*, 38:19518–19547, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yuanzhe Shen, Yide Liu, Zisu Huang, Ruicheng Yin, Xiaoqing Zheng, and Xuan-Jing Huang. Sater: A self-aware and token-efficient approach to routing and cascading. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10526–10540, 2025.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12974–12990, 2024.

Duncan Soiffer, Steven Kolawole, and Virginia Smith. Semantic agreement enables efficient open-ended llm cascades. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2499–2537, 2025.

Artem Vazhentsev, Akim Tsvigun, Roman Vashurin, Sergey Petrakov, Daniil Vasilev, Maxim Panov, Alexander Panchenko, and Artem Shelmanov. Efficient out-of-domain detection for sequence to sequence models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1430–1454, 2023.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3070–3079, 2020.

David Warren and Mark Dras. Bi-directional model cascading with proxy confidence. In *ECAI 2025: 28th European Conference on Artificial Intelligence 25-30 October 2025, Bologna, Italy-Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025) Proceedings*, pp. 4185–4192. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 21691–21728, 2024.

Yukun Zhao, Lingyong Yan, Weiwei Sun, Guoliang Xing, Chong Meng, Shuaiqiang Wang, Zhicong Cheng, Zhaochun Ren, and Dawei Yin. Knowing what llms do not know: A simple yet effective self-detection method. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7051–7063, 2024.

A Implementation Details and Hyperparameters

Model pairs. We conduct experiments with two SLM–LLM model pairs. For the Llama pair, we use Llama-3.2-1B-Instruct as the SLM and Llama-3.1-8B-Instruct as the LLM. For the Qwen pair, we use Qwen3-0.6B as the SLM and Qwen3-8B as the LLM. For the Semantic Agreement baseline, we follow its ensemble-based design and use three SLMs, including Llama-3.2-1B-Instruct, Qwen2.5-0.5B-Instruct, and Qwen3-0.6B.

Computing platform. All experiments are conducted on a server equipped with 8 NVIDIA RTX PRO 6000 Blackwell GPUs, each with approximately 96GB of GPU memory. The software environment uses NVIDIA driver version 580.173.02 and CUDA version 13.0. We train and evaluate the models in BF16 precision unless otherwise specified.

Supervised fine-tuning. Before applying UEC, we first adapt both the SLM and LLM to each downstream dataset using LoRA-based supervised fine-tuning. The learning rate is set to 2×10^{-4} for the SLM and 1×10^{-4} for the LLM. Both models are trained for 3 epochs with an effective batch size of 16. We use a cosine learning-rate scheduler with a warmup ratio of 0.03 and train the models in BF16 precision. For all model pairs and datasets, we use the same LoRA configuration, with LoRA rank 16, LoRA alpha 32, and LoRA dropout 0.05. The maximum sequence length is set to either 1024 or 1280, depending on the dataset and model pair.

Utility-aware entropy calibration. After supervised fine-tuning, we perform UEC on the SLM. During this stage, the base model parameters are frozen, and only the LoRA parameters obtained from the SFT stage are further updated. The LLM is not trained during calibration; instead, its teacher-forcing predictions are precomputed and used to assign token categories. The calibration stage uses a learning rate of 2×10^{-5}, a minimum learning rate of 2×10^{-6}, a cosine annealing scheduler without warmup, an effective batch size of 4, and BF16 precision. We train for 2 epochs. For GSM8K, MATH, and MBPP, entropy is computed over the full vocabulary. For MMLU, entropy is computed only over the four answer-option tokens $\{A, B, C, D\}$, and only the first answer token is used for calibration.

Calibration weights. The values of λ_A, λ_B, and λ_C used for different datasets and model pairs are listed in Table 4. The three weights control the contributions of Category A, Category B, and Category C to the overall calibration objective.

Table 4: Calibration weights used for UEC.

Model pair	Dataset	λ_A	λ_B	λ_C
Llama	GSM8K	0.70	1.00	1.00
Qwen	GSM8K	0.90	0.70	0.60
Llama	MATH	0.80	1.00	1.00
Qwen	MATH	0.85	0.85	0.85
Llama	MBPP	1.00	0.50	0.80
Qwen	MBPP	1.00	0.30	0.80
Llama	MMLU	0.85	0.85	0.85
Qwen	MMLU	1.00	0.40	0.80

Deferral Curves and AUC-DF Computation

To evaluate the overall cost–performance trade-off across different levels of LLM utilization, we plot deferral curves for all methods. In our experiments, the horizontal axis of the deferral curve is the token-level deferral rate, which measures the proportion of final output tokens generated by the LLM. This provides a unified token-level measure of LLM involvement across both token-level and sequence-level cascade methods.

For Triage and the SFT-based token-level cascade, each output position is generated either by the SLM or by the LLM. Let L_i denote the number of tokens finally generated by the LLM for sample i, and S_i denote the number of tokens finally generated by the SLM. The token-level deferral rate is computed over the whole dataset as a micro-average:

$$R = \frac{\sum_i L_i}{\sum_i (L_i + S_i)}.$$

The same definition is used for token-level Gatekeeper. In this case, L_i corresponds to the number of final output tokens whose source is the LLM, and S_i corresponds to the number of final output tokens whose source is the SLM. The confidence scorer in Gatekeeper is used only to decide whether each token should be deferred and is not included in the deferral-rate computation.

For sequence-level baselines, including Gatekeeper, Self-REF, and Semantic Agreement, each sample is finally answered either by the SLM or by the LLM. Let D denote the set of samples deferred to the LLM. For a deferred sample $i \in D$, the final LLM output length is denoted by T_i^{LLM}; for a non-deferred sample $i \notin D$, the final SLM output length is denoted by T_i^{SLM}. The token-level deferral rate is computed as

$$R = \frac{\sum_{i \in D} T_i^{\mathrm{LLM}}}{\sum_{i \in D} T_i^{\mathrm{LLM}} + \sum_{i \notin D} T_i^{\mathrm{SLM}}}.$$

That is, for deferred samples, all tokens in the final LLM output are counted as LLM-generated tokens; for non-deferred samples, all tokens in the final SLM output are counted only in the denominator. For Semantic Agreement, when a sample is not deferred, we use the length of the final output selected from the three-SLM ensemble. Target sample-level deferral rates, such as 5% or 10%, are

used only to select deferred samples for sequence-level baselines, while the x-axis in the deferral curves is computed using the actual token-level deferral rate defined above.

Given the deferral curve, we compute AUC-DF using the trapezoidal rule. For each method, we vary the deferral threshold to obtain a set of deferral rates and their corresponding task performance values. The points are sorted by deferral rate. Let r_i denote the normalized deferral rate of the i-th point and p_i denote the corresponding task performance, where $r_i \in [0, 1]$. AUC-DF is computed as

$$\text{AUC-DF} = \sum_{i=1}^{K-1} \frac{p_i + p_{i+1}}{2} \left(r_{i+1} - r_i\right),$$

where K is the number of sampled points on the deferral curve. A higher AUC-DF indicates a better overall cost–performance trade-off across different token-level deferral rates.

Latency measurement. For latency evaluation, we measure the average end-to-end inference time per sample for each method under deferral rates from 10% to 50%. For each target deferral rate, we select the corresponding deferral threshold or deferral-decision configuration, run inference on the evaluation set, and record the end-to-end wall-clock latency. The measured time covers the complete inference procedure of each method, including SLM generation, possible LLM calls, and method-specific deferral-decision operations. These measurements are used to plot the latency vs. performance curves in Figure 4.

B Token Category Statistics

Table 5 reports the proportions of the three token categories across different model pairs and datasets.

Table 5: Proportions of the three token categories across different model pairs and datasets. Category A contains tokens correctly predicted by the SLM, Category B contains tokens incorrectly predicted by the SLM but correctly predicted by the LLM, and Category C contains tokens incorrectly predicted by both models.

Model pair	Dataset	Category A	Category B	Category C
Llama	GSM8K	88.51%	3.99%	7.51%
Llama	MATH	83.47%	6.18%	10.35%
Llama	MBPP	84.98%	4.66%	10.37%
Llama	MMLU	86.48%	10.70%	2.82%
Qwen	GSM8K	89.99%	4.97%	5.04%
Qwen	MATH	84.58%	4.91%	10.51%
Qwen	MBPP	86.39%	4.88%	8.72%
Qwen	MMLU	92.84%	5.38%	1.78%

As shown in Table 5, Category A accounts for the majority of tokens across all model pairs and datasets, typically exceeding 80%. This indicates that most tokens can already be correctly predicted by the SLM, making it important to preserve the SLM's confidence on correct tokens and maintain its downstream task capability during calibration. This observation also explains why L_A has a pronounced effect in the ablation study.

Meanwhile, Categories B and C both appear with non-negligible proportions across different tasks, suggesting that erroneous SLM tokens do not share the same deferral utility. Category B represents errors that can be corrected by the LLM and thus serves as the main source of useful deferrals, whereas Category C represents errors that cannot be corrected by the LLM, for which deferral only adds cost without improving performance. Therefore, relying solely on the SLM's own uncertainty is insufficient, and the calibration process needs to further distinguish correctable from uncorrectable errors.

The category distributions also vary across tasks. For example, MATH and MBPP have relatively higher proportions of Category C, indicating that these tasks contain more errors that are difficult even for the LLM to correct. By contrast, MMLU shows a relatively higher proportion of Category

B, especially with the Llama model pair, where Category B reaches 10.70%, suggesting that more tokens can benefit from LLM intervention. Overall, these statistics further support the design of UEC, which maintains low entropy for Category A, increases entropy for Category B to promote useful deferrals, and reduces entropy for Category C to suppress ineffective deferrals.

C Additional Analysis

C.1 Additional Ablation Curves

Figure 6 provides the complete deferral curves of different ablation variants on GSM8K with the Llama model pair. The results further support the observations in Table 3. Compared with the full Triage model, removing token-level deferral leads to consistently weaker performance across most deferral rates, showing the importance of making fine-grained deferral decisions during generation. Removing the complete UEC loss also degrades the curve, indicating that token-level cascading alone is insufficient without a deferral signal aligned with LLM intervention utility.

Among the three loss terms, removing L_A causes the most pronounced degradation, especially in the low-deferral-rate region. Since this region relies more heavily on the SLM, the poor performance of the w/o L_A variant suggests that L_A is crucial for preserving the downstream task capability of the SLM during calibration. By contrast, removing L_B or L_C results in smaller but still observable performance drops, further confirming the necessity of the utility-aware design.

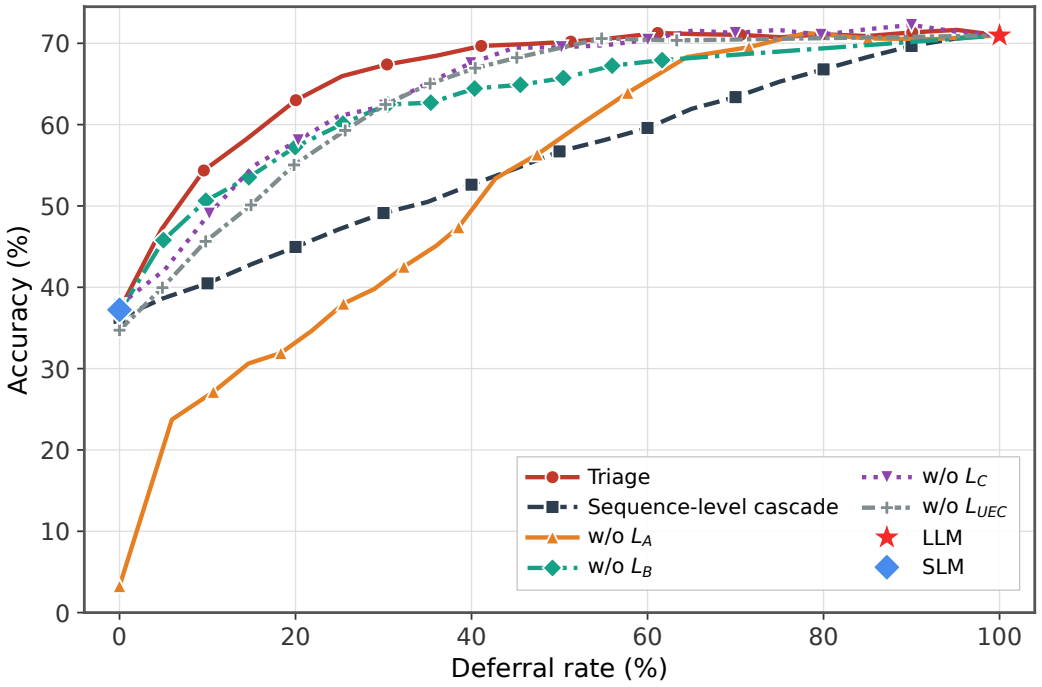


Figure 6: Deferral curves of different ablation variants of Triage on GSM8K with the Llama model pair. The results show that removing token-level deferral or UEC weakens the cost–performance trade-off, while removing L_A leads to the most significant degradation, especially under low deferral rates.

C.2 Additional Entropy Distribution Results

Figure 7 provides additional entropy distribution results on MBPP and MMLU with two model pairs. These results complement the analysis in the main text. Overall, similar trends can be observed across these datasets: before calibration, the entropy distributions of the three token categories are less clearly separated, while after UEC, the distributions become more distinguishable and move

consistently with the calibration objectives. In particular, Category A remains in the low-entropy region, Category B tends to shift toward higher entropy, and Category C stays relatively lower than Category B, indicating that the calibrated entropy better reflects the utility of LLM intervention. These observations further show that the effect of UEC is consistent across different tasks beyond GSM8K and MATH.

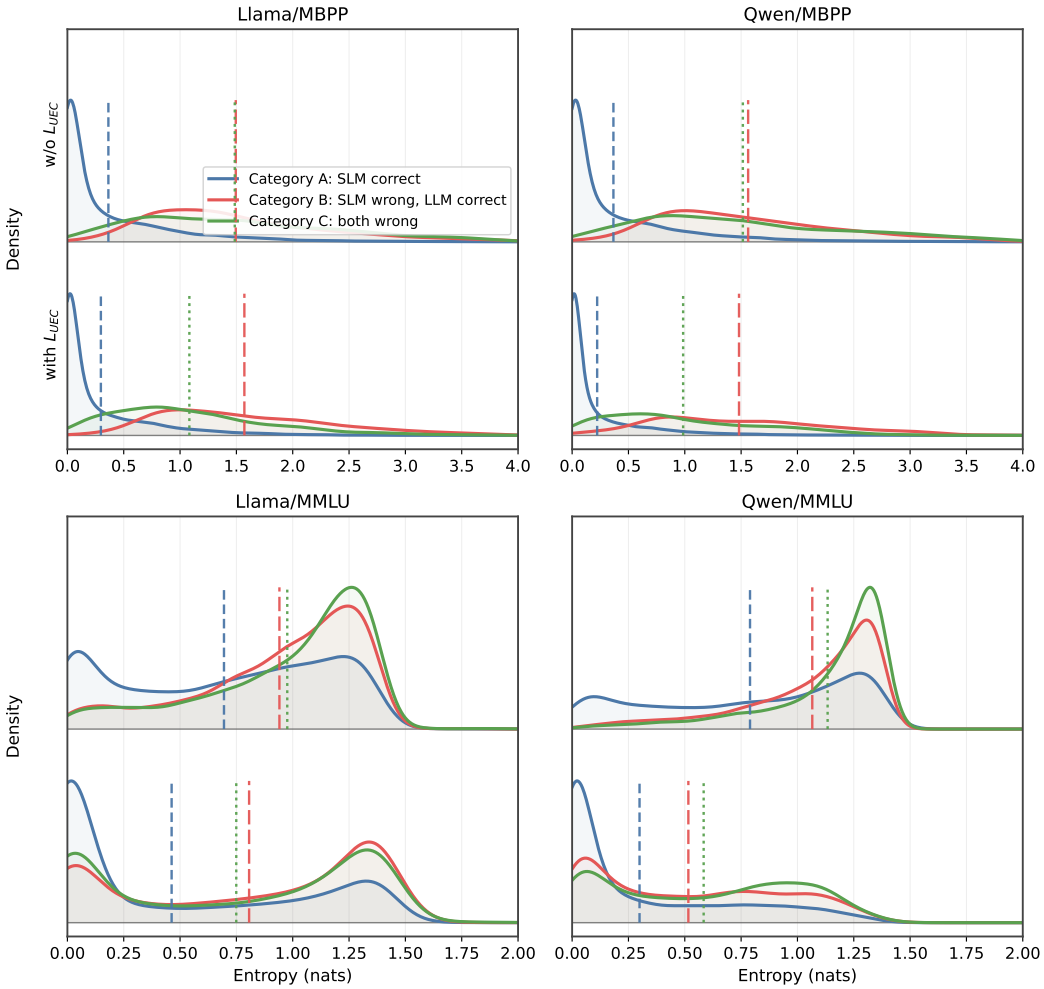


Figure 7: Additional entropy distribution results before and after UEC on MBPP and MMLU with two model pairs. The upper row in each subplot shows the results without UEC, while the lower row shows the results after applying calibration. After calibration, the three categories become more distinguishable, with entropy distributions moving consistently with the calibration objectives.

C.3 Additional Useful Deferral Precision Results

To further evaluate whether predictive entropy is aligned with the actual utility of LLM intervention, we compute useful deferral precision (UDP) using the held-out test predictions and token-level entropy values without retraining any model. The main text reports the UDP results on the Llama model pair, while this section provides additional results on the Qwen model pair.

For each valid token, we assign it to one of the three categories defined in the main text. Category A contains tokens correctly predicted by the SLM, Category B contains tokens incorrectly predicted by the SLM but correctly predicted by the LLM, and Category C contains tokens incorrectly predicted by both models. Only Category B corresponds to useful deferrals, since these tokens are exactly the positions where LLM intervention can correct the SLM.

For each dataset and method, all valid tokens are ranked by entropy in descending order. Given a target token-level deferral budget $r \in \{10\%, 20\%, 30\%\}$, we select the top $k = \lfloor rN + 0.5 \rfloor$ tokens, where N is the total number of valid tokens. UDP is then computed as

$$\mathrm{UDP@}r = \frac{|\mathcal{S}_r \cap B|}{|\mathcal{S}_r|},$$

where $\mathcal{S}_r$ denotes the selected high-entropy tokens. A higher UDP indicates that, under the same token-level deferral budget, more computation is allocated to positions where the LLM can actually correct the SLM.

Table 6: Useful deferral precision (UDP) results on the Qwen model pair at fixed token-level deferral budgets. UDP measures the proportion of selected high-entropy tokens that belong to Category B, i.e., tokens incorrectly predicted by the SLM but correctly predicted by the LLM. The best result in each row and each budget is shown in bold. SFT-T and Gate.-T denote the SFT-based token-level cascade and token-level Gatekeeper, respectively. Higher is better.

Dataset	Triage			Gate.-T			SFT-T		
	UDP@10	UDP@20	UDP@30	UDP@10	UDP@20	UDP@30	UDP@10	UDP@20	UDP@30
GSM8K	**0.2065**	**0.1620**	**0.1261**	0.1736	0.1509	0.1234	0.1782	0.1564	0.1227
MATH	**0.2660**	**0.2096**	**0.1758**	0.1737	0.1752	0.1642	0.1792	0.1799	0.1690
MBPP	**0.3147**	**0.2665**	**0.2255**	0.2614	0.2392	0.2141	0.2611	0.2401	0.2150
MMLU	**0.4074**	**0.3896**	**0.4007**	0.3754	0.3725	0.3786	0.3796	0.3778	0.3819

As shown in Table 6, Triage achieves the highest UDP@10, UDP@20, and UDP@30 across all four datasets with the Qwen model pair. This indicates that, under different token-level deferral budgets, the calibrated entropy produced by UEC consistently selects more LLM-correctable SLM errors than the token-level baselines. The improvement is especially clear on MATH and MBPP, where Triage shows a large advantage over both Gate.-T and SFT-T across all three budgets. On GSM8K and MMLU, Triage also consistently outperforms the baselines, suggesting that the benefit of UEC is not limited to a specific task type.

Together with the Llama results reported in the main text, these Qwen results further show that UEC improves the alignment between predictive entropy and useful deferral positions across different model families. Overall, the UDP analysis supports the effectiveness of UEC in allocating limited LLM computation to token positions where LLM intervention is more likely to correct the SLM.

C.4 Sensitivity Analysis of Calibration Weights

Figure 8 presents the full sensitivity analysis of the calibration weights on GSM8K with the Llama model pair. We vary λ_A, λ_B, and λ_C within $\{0.7, 0.8, 0.9, 1.0\}$, resulting in 64 parameter combinations. The heatmaps report the downstream task accuracy of the SLM after UEC.

Overall, the SLM accuracy remains stable across different weight combinations. All parameter settings outperform the SFT-only SLM accuracy of 34.72%, indicating that UEC does not degrade the downstream task capability of the SLM. The average accuracies under different λ_A values are also close to each other, suggesting that the proposed calibration objective is not overly sensitive to the choice of calibration weights.

C.5 Comparison with Longer SFT Training

Figure 9 compares Triage with token-level cascades built on SFT models trained for different numbers of epochs. This comparison is designed to examine whether the improvement of Triage simply comes from additional training rather than UEC. As shown in the figure, the SFT-based token-level cascade with 5 epochs does not consistently outperform the 3-epoch SFT variant, and both remain below Triage across most deferral rates. The advantage of Triage is especially clear in the low- and medium-deferral-rate regions, where the system must rely on a more accurate deferral signal to allocate limited LLM calls.

These results indicate that the performance gain of Triage cannot be explained by longer SFT training alone. Instead, the improvement mainly comes from UEC, which explicitly adjusts the SLM's

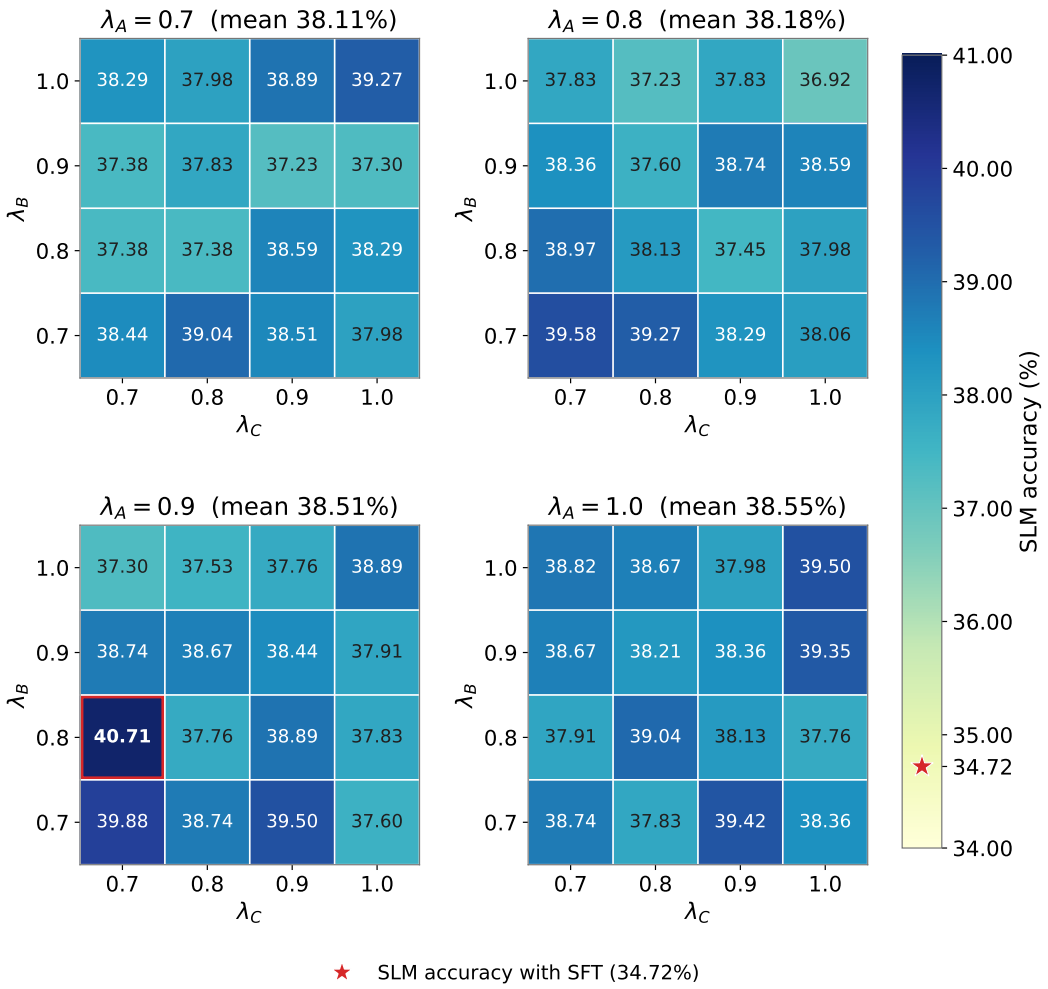


Figure 8: Sensitivity analysis of the calibration weights on GSM8K with the Llama model pair. We vary λ_A, λ_B, and λ_C over $\{0.7, 0.8, 0.9, 1.0\}$, resulting in 64 parameter combinations. The heatmaps show the downstream task accuracy of the SLM after UEC. The SFT-only SLM accuracy is 34.72%, marked as the reference baseline.

predictive entropy according to the utility of LLM intervention. This further supports the necessity of the proposed calibration objective.

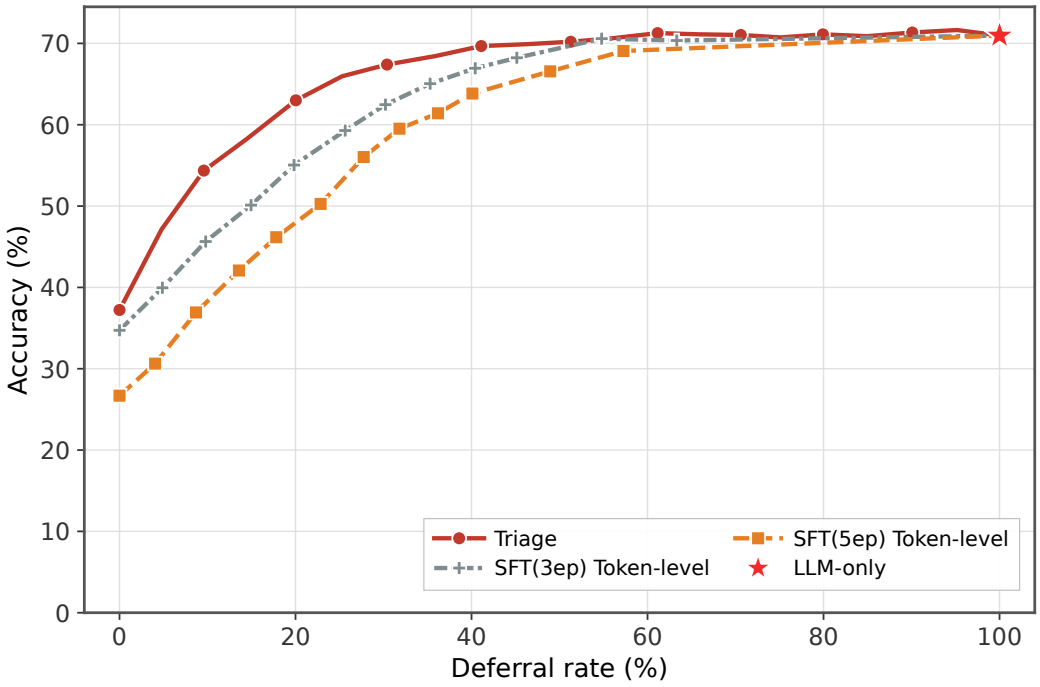


Figure 9: Comparison between Triage and SFT-based token-level cascades with different training epochs on GSM8K with the Llama model pair. Longer SFT training alone does not reproduce the improvement of Triage, indicating that the gain mainly comes from UEC rather than additional SFT epochs.