Cluster, Route, Escalate: Cascaded Framework for Cost-Aware LLM Serving

Yasmin Moslem[1*] **Magdalena Kacmajor**[1]
Vasudevan Nedumpozhimana[1] **Ammar Abbas**[1] **Solmaz Panahi**[1]
David Lynch[2] **Zhuangzhuang Nie**[2] **Alexandros Agapitos**[2] **Aleksandar Milenovic**[2]
Hongmeng Song[2] **Yucheng Shi**[2] **Yue Pan**[2] **Patricia Buffini**[1] **John D. Kelleher**[1*]

[1]ADAPT Centre, Trinity College Dublin
[2]Huawei Research

[*] yasmin.moslem, john.kelleher{at}adaptcentre.ie

Abstract

Efficient deployment of large language models (LLMs) in production forces a trade-off between accuracy and cost. Operators often default to a single model that is either expensive for easy queries or insufficient for hard ones. To address this challenge, we propose a two-stage cascaded solution. Stage 1 clusters incoming queries and assigns each cluster to its most cost-effective model. The cost budget for this routing process is set by an interpretable hyperparameter, tuned offline. Stage 2 adds a quality estimation (QE) cascade; when an output from Stage 1 is judged low-quality, the query is escalated to a stronger model. This ensures only hard or low-confidence cases reach the expensive models. On the test datasets, the cascaded system retains 97-99% of the strongest model's accuracy while reducing Time Per Output Token (TPOT). It requires only task-correctness labels and adapts to changes in the model pool without manual reconfiguration.

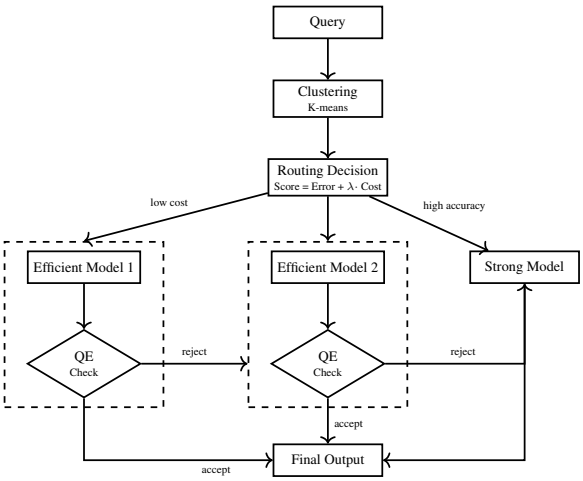


Figure 1: Two-stage cascaded routing system. Stage 1 clusters incoming queries and applies cost-aware routing to a pool of models. Stage 2 applies a QE classifier to outputs from efficient models, escalating queries with low-quality responses to a stronger model.

1 Introduction

Open-weight LLMs now span a wide range of sizes and cost-accuracy trade-offs, but production deployments face a fundamental tension between accuracy and serving efficiency. In practice, operators typically select a single model, either the strongest available for maximum accuracy, or a smaller one for efficiency. Both extremes are wasteful, as stronger models over-invest on easy queries while a small model underperforms on hard ones.

Model routing addresses this by directing each query to the most appropriate model in a pool, but existing systems typically require annotations beyond standard task evaluation. Jointly optimising routing under an explicit inference cost budget and recovering accuracy through post-generation quality estimation, using only task-correctness labels, remains underexplored (Moslem and Kelleher, 2026).

We propose a two-stage cascaded framework that jointly optimises routing under a TPOT budget and recovers accuracy through post-generation quality estimation, using only task-correctness labels obtained from evaluating models on training queries.

- **Stage 1 – Clustering-based routing:**
  - (i) **Semantic clustering:** k-means partitions queries by semantic similarity into coherent groups.
  - (ii) **Cost-aware routing:** each model is scored by its cost-adjusted error per cluster via a single hyperparameter λ.
  - (iii) **Automatic λ selection:** λ^* is automatically tuned on training data to satisfy a user-specified TPOT budget B, and applied unchanged at test time.
- **Stage 2 – Quality Estimation (QE) cascade:** a lightweight classifier inspects each efficient-model output and re-routes queries with low-quality responses to a stronger model. It is trained on the same task-correctness labels as Stage 1, adding accuracy recovery without requiring any additional annotation.

We evaluate on TeleQnA (Maatouk et al., 2025) (telecommunications QA) and AIME 2024 (mathematical reasoning) using pools from the Qwen 3, Qwen 3.5, and Gemma 4 families, confirming generalisation across domains: Stage 1+2 retains 97% of the strongest model's accuracy on TeleQnA; on AIME 2024, it retains nearly all of the strongest model's accuracy at 18% lower TPOT.

2 Related Work

LLM routing and cascades. While model routing makes a single decision to map the query to one model, model cascading operates sequentially, escalating to larger models when the initial response is insufficient (Moslem and Kelleher, 2026). HybridLLM (Ding et al., 2024) trains a binary router to direct queries to a small or large model by jointly optimising cost and quality. RouteLLM (Ong et al., 2025) learns a routing policy from human preference data; its routers include matrix factorisation and causal LLM classifiers trained on Chatbot Arena data (Chiang et al., 2024). IRT-Router (Song et al., 2025) applies item response theory to model query difficulty and LLM ability jointly, combining predicted correctness probability with cost into a routing score for interpretable decisions. UniRoute (Jitkrittum et al., 2026) addresses dynamic routing where previously unseen LLMs become available at test time, representing each model as a feature vector derived from its predictions on a set of representative prompts. Our framework also adapts to model pool changes, but differs in two respects: we automatically select λ^* to satisfy an explicit TPOT budget rather than sweeping λ over a monetary-cost trade-off curve, and add a QE cascade for post-routing accuracy recovery.

FrugalGPT (Chen et al., 2024) introduced cascade-based LLM serving that queries a sequence of models adaptively, stopping when a quality threshold is reached. AutoMix (Aggarwal et al., 2024) uses self-verification to decide escalation. Firewall routing (Peng et al., 2025) blocks queries from reaching small models when predicted failure rates are too high. While we share the goal of cascaded LLM serving, our framework differs in two ways: Stage 1 pre-routes hard queries directly to the strong model under an explicit TPOT budget, avoiding wasted efficient-model calls, while Stage 2 uses a separate lightweight classifier trained on task-correctness labels as a QE cascade.

Adaptive inference and quality estimation. Estimating output quality without a reference signal enables adaptive decisions at inference time. SelfREF (Chuang et al., 2025a) trains LLMs to emit confidence tokens that trigger escalation, while Chuang et al. (2025b) benchmark uncertainty-driven routing from on-device SLMs to stronger LLMs. CP-Router (Su et al., 2025) applies conformal prediction to route between standard LLMs and large reasoning models based on output uncertainty. Farinhas et al. (2025) propose quality-aware deferral in a cascaded translation system, escalating outputs when a quality estimator predicts failure; our Stage 2 applies the same deferral principle across domains using task-correctness supervision. Adaptive thinking-length control (Zhang et al., 2025a,b) adjusts compute per query at the level of reasoning steps rather than model selection. Speculative decoding (Leviathan et al., 2023) accelerates generation within a single model, while our framework routes and escalates across models.

Efficient LLM deployment in telecom network. Telecommunications is a demanding setting for efficient LLM inference, where domain competence and serving efficiency are both first-order. TeleQnA shows that general-purpose LLMs struggle with standards questions, motivating telecom-specialised models (Maatouk et al., 2025). The field is shifting from human-in-the-loop co-pilots toward autonomous multi-agent systems owning the full lifecycle, from detection and diagnosis to remediation and validation (Xiao et al., 2026; NVIDIA, 2026). These always-on agents reshape the serving problem; since they act on sensitive operational data, their models must run on-premise, avoiding external APIs to preserve data privacy and sovereignty. High-volume concurrent events and long-horizon workflows make per-token latency a major constraint. Hence, our framework optimises for TPOT, where low-TPOT models keep such autonomous loops responsive on fixed compute. Routing with cascaded escalation fits naturally, as small or domain-adapted models can handle routine subtasks, while stronger reasoning models are reserved for more challenging cases. Rather than commissioning a new telecom-specialised model, such as the roughly 30B Nemotron-based model fine-tuned on telecom data (NVIDIA, 2026; AdaptKey, 2026), our framework treats the on-premise pool as a deployment substrate. It can reuse specialised models where they suffice (e.g. VibeThinker-1.5B (Xu et al., 2025) trained for maths and coding) and absorb new ones through Pareto analysis and updated routing tables, rather than expensive specialisation cycles. Trained only from task-correctness labels, it gives operators an interpretable knob trading accuracy against inference cost.

3 System Overview

Figure 1 illustrates the full two-stage system. An incoming query is encoded and assigned to a cluster (Stage 1), then routed to one of several candidate models based on a decision score. Outputs from efficient models are passed through the QE classifier (Stage 2): accepted outputs are returned to the user; queries with low-quality outputs are re-routed to a stronger model.

Cluster centroids and per-cluster routing tables are computed once offline from task-correctness labels on the training corpus, the only annotation required, which is available from standard benchmark evaluation. At inference, each query undergoes a single embedding lookup and one $\arg\min$ over the model pool. Pool updates require only inference on the existing training corpus, with no additional annotation.

Without Stage 1, the QE cascade would need to run an efficient model on every query before deciding to escalate. Stage 1 avoids this by routing entire clusters directly to a strong model where warranted, reserving Stage 2 for per-query failures within clusters assigned to an efficient model. Together, the two stages span the routing design space from pre-generation query routing to post-generation response evaluation, addressing the multi-stage cascade gap identified in prior work (Moslem and Kelleher, 2026).

4 Stage 1: Clustering-Based Routing

4.1 Problem Formulation

Let $\mathcal{M} = \{m_1, \ldots, m_K\}$ be a pool of candidate models and $\mathcal{C} = \{c_1, \ldots, c_N\}$ a partition of the query space into N clusters. For each model m and cluster c, let $\mathrm{Error}(m, c)$ denote the empirical error rate on training queries in that cluster. The routing score is:

$$\mathrm{Score}(m, c) = \mathrm{Error}(m, c) + \lambda \cdot \mathrm{Cost}_{\mathrm{norm}}(m) \quad (1)$$

where $\lambda \geq 0$ controls the accuracy-latency trade-off and $\mathrm{Cost}_{\mathrm{norm}}(m)$ normalises our primary efficiency metric, Time Per Output Token (TPOT), over the pool:

$$\mathrm{Cost}_{\mathrm{norm}}(m) = \frac{\mathrm{TPOT}(m) - \mathrm{TPOT}_{\min}}{\mathrm{TPOT}_{\max} - \mathrm{TPOT}_{\min}} \quad (2)$$

The fastest model has $\mathrm{Cost}_{\mathrm{norm}}{=}0$ and the slowest has $\mathrm{Cost}_{\mathrm{norm}}{=}1$, making λ directly interpretable as the maximum tolerated error-rate penalty for using the most expensive model. Each query is routed to $\arg\min_m \mathrm{Score}(m, c)$, with ties broken in favour of the faster model.

4.2 Clustering

We adopt the query-embedding clustering scheme of Jitkrittum et al. (2026): queries are encoded with *all-MiniLM-L6-v2* (Wang et al., 2020) and then clustered using k-means. The cluster count N is selected by maximising the mean Silhouette score (Rousseeuw, 1987) over $k \in [2, 10]$. Centroids are fixed on training data; at inference each new query is assigned to the nearest centroid in a single embedding pass.

4.3 Pareto Analysis and Model Selection

A model m is Pareto-dominated if there exists m' such that $\mathrm{TPOT}(m') \leq \mathrm{TPOT}(m)$ and $\mathrm{Error}(m', c) \leq \mathrm{Error}(m, c)$ for all c, with at least one strict inequality. Dominated models cannot be selected by Equation 1 for any λ and are discarded. When new models arrive, Pareto analysis is re-run automatically. On TeleQnA (Section 7.2), this pruning reduces the four-model pool to two Pareto-efficient candidates.

4.4 Crossover Points and Routing Regions

For any pool of K models and N clusters, sweeping λ from 0 upwards produces routing-region boundaries where the $\arg\min_m$ assignment changes for at least one cluster; between boundaries a fixed routing strategy holds. For $K{>}2$, these boundaries are identified numerically via the $\arg\min_m$ sweep; this applies, for instance, to TeleQnA's four-model pool (Section 7.2). For $K{=}2$, the boundary for each cluster has the closed form:

$$\lambda_c = \mathrm{Error}(m_{\mathrm{fast}}, c) - \mathrm{Error}(m_{\mathrm{strong}}, c) \quad (3)$$

This simplification holds since $\mathrm{Cost}_{\mathrm{norm}}(m_{\mathrm{fast}}){=}0$ and $\mathrm{Cost}_{\mathrm{norm}}(m_{\mathrm{strong}}){=}1$, making the cost-difference denominator equal to 1. For $\lambda < \lambda_c$ the larger model is preferred (lower penalty on inference cost); above it the efficient model is preferred (higher penalty on inference cost). With N clusters, the N crossover points define $N{+}1$ fully interpretable routing regions; all λ values within a region yield the same cluster-to-model assignment. Figure 2 illustrates the four regions for the AIME 2024 two-model pool.

4.5 λ Selection

Given a cost budget B, the optimal λ is:

$$\lambda^* = \arg\max_{\lambda} \{\mathrm{Acc}(\lambda) \mid \mathrm{TPOT}(\lambda) \leq B\} \quad (4)$$

where $\mathrm{Acc}(\lambda)$ and $\mathrm{TPOT}(\lambda)$ denote system accuracy and average TPOT under routing strategy λ,

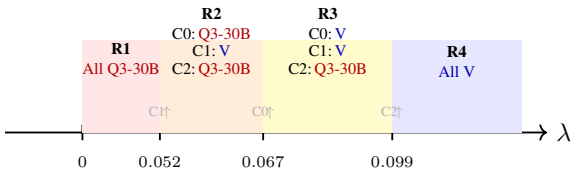


Figure 2: Routing regions for the AIME 2024 two-model pool. As the inference cost penalty λ increases, clusters flip from Qwen3-30B-A3B-Thinking-2507-FP8 (Q3-30B) to VibeThinker-1.5B (V) at the indicated crossover points; C1 flips first (smallest Q3-30B advantage) and C2 last (largest).

selected on training data and applied unchanged at test time. B is a user-specified deployment target, where we adopt B=20 ms in the experiments. We define an efficiency metric quantifying accuracy cost per millisecond saved relative to the λ=0 baseline (routing all queries to the strong model):

$$\eta(\lambda) = \frac{\mathrm{Acc}(0) - \mathrm{Acc}(\lambda)}{\mathrm{TPOT}(0) - \mathrm{TPOT}(\lambda)} \tag{5}$$

Lower η indicates a more favourable trade-off.

5 Stage 2: Quality Estimation Cascade

The Quality Estimation (QE) cascade routes queries through an efficient model first, escalating to a more costly, stronger model only when output quality is deemed insufficient. Stage 1 routing is applied at the cluster level: the same model handles all queries in a cluster. For clusters (queries) routed to an efficient model, completions may still be of poor quality, limited by efficient models capability. Stage 2 inspects each completion post-generation and re-routes the query to a stronger model when required.

5.1 QE Classifier

We fine-tune ModernBERT-base (Warner et al., 2025) as a binary classifier with labels *accept* and *escalate*. The input combines the query, the model output, and the generation length; training labels are derived from task-correctness of the efficient model's output. Dataset-specific input formats, training data, and hyperparameters are detailed in Appendix C. At inference, the $\arg\max$ over class probabilities determines acceptance or escalation to a stronger model.

5.2 Cascade Integration

Stage 2 is applied only to outputs from efficient models. Outputs already produced by a strong model bypass the classifier, preserving Stage 1 latency savings. Relative to generation, the classifier adds only a small per-token cost, quantified in Appendix F.

6 Datasets and Model Pool

We evaluate on two datasets spanning different domains, query volumes, and model pools to confirm that the framework generalises beyond any single setting. The contrasting test-set scales (30 vs. 1,000 queries) are deliberate: TeleQnA provides statistical robustness while AIME provides a challenging fixed-competition benchmark in the Mathematics domain.

AIME 2024. The American Invitational Mathematics Examination dataset provides 921 training queries (AIME 1983–2023) and 30 test queries (AIME 2024), the full fixed competition set for this benchmark. Silhouette analysis selects 3 clusters. The primary model pool consists of *VibeThinker-1.5B* (hereafter V) (Xu et al., 2025), trained for maths and coding, and *Qwen3-30B-A3B-Thinking-2507-FP8* (hereafter Q3-30B) (Yang et al., 2025). In the extensibility experiment (Appendix B), we further consider *Qwen3-4B-Thinking-2507-FP8* and *Qwen3.5-35B-A3B-FP8* (Yang et al., 2025).

TeleQnA. TeleQnA (Maatouk et al., 2025) is a multiple-choice QA benchmark for the telecommunications domain. We use 9,000 training queries and 1,000 test queries; Silhouette analysis identifies 2 clusters. The initial model pool contains *Qwen3-4B-Instruct* (Yang et al., 2025) (Q3-4B), *Gemma4-E2B-it* (G-E2B), *Gemma4-26B-it* (G-26B), and *Gemma4-E4B-it* (Kamath et al., 2025) (G-E4B).

7 Experiments and Results

We evaluate Stage 1 routing and Stage 2 QE cascade on both AIME 2024 and TeleQnA. The combined Stage 1+2 system is then compared against single-model baselines.

7.1 Stage 1 – AIME 2024

VibeThinker (V) is a fast model trained for maths and coding but still less accurate than Qwen3-30B-A3B (Q3-30B) which is larger and slower. Per-cluster training performance is shown in Table 1. The three crossover points (Equation 3) are λ_0=0.067, λ_1=0.052, λ_2=0.099, defining four routing regions (cf. Figure 2 and Table 2).

Table 2 shows routing performance on the training data, with one representative λ per routing region. With a TPOT budget B=20 ms, Equation 4 selects λ^*=0.06: C1, where Q3-30B's accuracy advantage over V is smallest (crossover point λ_1=0.052), is assigned to V, and C0 and C2 to Q3-30B. Concretely, at λ=0.06, V scores 0.083 on C1 versus Q3-30B's 0.091, while Q3-30B wins C0 (0.123 vs. 0.130) and C2 (0.143 vs. 0.182).

	C0	C1	C2
VibeThinker-1.5B (V)			
TPOT (ms)	9.282	9.348	8.825
Error	0.130	0.083	0.182
Qwen3-30B-A3B-FP8 (Q3-30B)			
TPOT (ms)	23.419	24.070	26.620
Error	0.063	0.031	0.083

Table 1: Per-cluster model performance on AIME training set.

λ	C0	C1	C2	Acc	TPOT (ms)	η
0.00	Q3-30B	Q3-30B	Q3-30B	94.4%	24.8	—
0.06	Q3-30B	V	Q3-30B	92.1%	18.4	**0.36**
0.07	V	V	Q3-30B	90.7%	15.4	0.39
0.10	V	V	V	87.3%	9.2	0.46

Table 2: Routing strategies on the AIME training set for selected λ values. C0/C1/C2 indicate the model assigned to each cluster.

Table 3 reports routing behavior across λ values on the AIME 2024 test set. Performance closely mirrors training trends, validating λ selection on training data. At λ=0.06, TPOT is reduced by 19% (from 11.8 to 9.5 ms) at a cost of 2.7% accuracy (89.1% $\rightarrow$ 86.4%). Full strategy comparisons including the combined Stage 1+2 system are in Table 7.

λ	C0	C1	C2	Acc (%)	TPOT (ms)	η
0.00	Q3-30B	Q3-30B	Q3-30B	89.1	11.8	—
0.06	Q3-30B	V	Q3-30B	86.4	9.5	**1.17**
0.07	V	V	Q3-30B	81.1	7.6	1.90
0.10	V	V	V	76.7	4.8	1.77

Table 3: Routing assignments by λ on the AIME 2024 test set.

7.2 Stage 1 – TeleQnA

Per-cluster training performance for all four models is reported in Table 4. Figure 3 shows Pareto analysis of the four-model pool. G-E2B is dominated by Q3-4B (lower TPOT and lower error on both clusters); G-E4B is dominated by G-26B. The surviving pool is Q3-4B and G-26B.

λ	C0	C1	Acc (%)	TPOT (ms)	η
0.00	G-26B	G-26B	76.4	24.5	—
0.07	Q3-4B	G-26B	71.2	19.1	**0.96**
0.08	Q3-4B	Q3-4B	66.9	15.1	1.01

Table 6: Routing assignments by λ on the TeleQnA test set.

Table 5 shows training-set routing performance, with one representative λ per routing region. With budget B=20 ms, λ^*=0.07, assigning C0 to Q3-4B and C1 to G-26B ($\eta = 0.63$ pp/ms). Concretely, at λ=0.07, Q3-4B scores 0.297 on C0 versus G-26B's 0.301, while G-26B wins C1 (0.324 vs. 0.329). Table 6 reports routing behaviour across λ values on the TeleQnA test set; performance mirrors training trends, validating λ selection on training data. Full strategy comparisons including the combined Stage 1+2 system are in Table 8.

Model	TPOT (ms)	C0 Error	C1 Error	Pareto
Q3-4B	15.357	0.297	0.329	efficient
G-E2B	20.337	0.339	0.390	dominated
G-26B	25.963	0.231	0.254	efficient
G-E4B	26.827	0.332	0.293	dominated

Table 4: Per-cluster model performance on the TeleQnA training set.

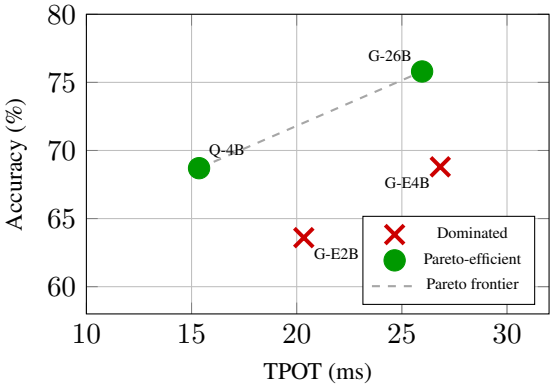


Figure 3: Pareto analysis of the TeleQnA model pool on training data. Dominated models (×) are pruned; Pareto-efficient models (•) form the routing pool.

7.3 Stage 1+2: AIME 2024

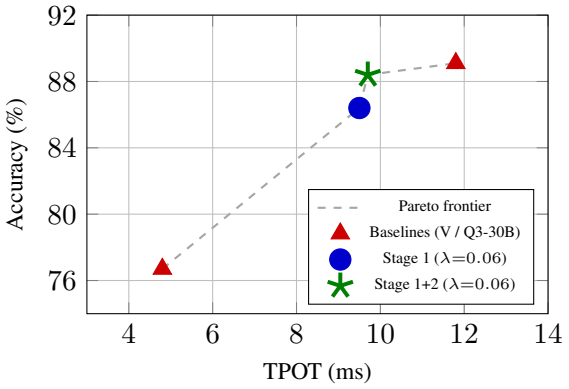
At λ=0.06, Stage 1 routes C1 (10 of 30 test queries) to VibeThinker; C0 and C2 go directly to Q3-30B and bypass the QE classifier.

Averaged over 5 runs, the classifier escalates an average of 0.6 queries per run from VibeThinker to Q3-30B on C1. C1 accuracy rises to 96% (from 90.0% under Stage 1 alone) with negligible additional TPOT. The overall system achieves 88.4% accuracy at 9.7 ms TPOT.

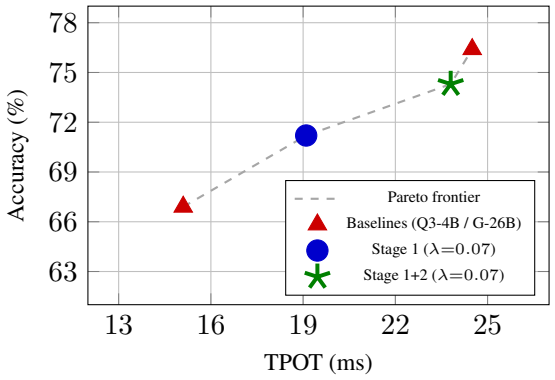
Table 7 and Figure 4a compare all systems on the AIME 2024 test set. Stage 2 recovers 2.0% accuracy at a cost of only 0.2 ms additional TPOT. The combined system lies on the Pareto frontier and is within 0.7% of always using Q3-30B while being 2.1 ms (18%) faster. An ablation study (removing Stage 1 routing phase) that isolates QE classifier's

λ	C0	C1	Acc (%)	TPOT (ms)	η
0.00	G-26B	G-26B	75.9	26.0	—
0.07	Q3-4B	G-26B	72.1	19.9	**0.63**
0.08	Q3-4B	Q3-4B	69.0	15.4	0.66

Table 5: Routing strategies on the TeleQnA training set for selected λ values.



(a) AIME 2024: Stage 1+2 achieves 88.4% at 9.7 ms, within 0.7 pp of always using Q3-30B at 18% lower latency.



(b) TeleQnA: Stage 1+2 achieves 74.3% at 23.8 ms, recovering 3.1 pp over Stage 1 alone.

Figure 4: Cost-accuracy trade-offs on the AIME 2024 and TeleQnA test sets. Stage 1+2 lies on the Pareto frontier in both cases.

ability to correctly decide whether to route to a stronger model is reported in Appendix D.

Strategy	Accuracy	TPOT (ms)
Always Q3-30B	89.1%	11.8
Stage 1 only (λ=0.06, Q3-30B/V/Q3-30B)	86.4%	9.5
Stage 1 + 2 (λ=0.06, QE on C1)	**88.4%**	**9.7**
Always V	76.7%	4.8

Table 7: Routing strategy comparison on the AIME 2024 test set. Stage 1+2 recovers 2.0% accuracy over Stage 1 alone, to be within 1% of the strongest model accuracy at 18% lower latency.

7.4 Stage 1+2: TeleQnA

At λ=0.07, Stage 1 routes C0 (590 of 1,000 test queries) to Q3-4B; C1 goes directly to G-26B and bypasses the QE classifier.

The classifier escalates on average 202 of 590 C0 queries per run from Q3-4B to G-26B, raising C0 accuracy from 68.9% to 74.0%. The overall system achieves 74.3% accuracy at 23.8 ms TPOT. Table 8 and Figure 4b compare all systems on the TeleQnA test set.

Strategy	Accuracy	TPOT (ms)
Always G-26B	76.4%	24.5
Stage 1 (λ=0.07, Q3-4B/G-26B)	71.2%	19.1
Stage 1 + 2 (λ=0.07, QE on C0)	**74.3%**	**23.8**
Always Q3-4B	66.9%	15.1

Table 8: Routing strategy comparison on the TeleQnA test set. Stage 1+2 recovers 3.1 pp accuracy over Stage 1 alone while remaining faster than always using the stronger model.

Stage 2 recovers 3.1% accuracy over Stage 1 alone at a cost of 4.7 ms additional TPOT. The combined system is within 2.1% of always using G-26B while being 0.7 ms faster, with Q3-4B handling 59% of queries. Of the 202 escalations per run, about 104 are false positives (correct answers escalated unnecessarily) and the rest, 98 (=202-104), are true positives; the classifier also accepts 85 incorrect answers as false negatives (Appendix E.2). The true-positive escalations drive the accuracy recovery, but the 51% false discovery rate means the gain comes with unnecessary escalations. Overall, the cascade recovers 3.1 pp accuracy over Stage 1 at an additional 4.7 ms TPOT.

8 Conclusion and Future Work

The results show that model-pool efficiency can be improved without committing to either a single strong model or a learned router tied to a fixed pool. Cluster-level routing selects a budget-feasible operating point from task-correctness labels, while selective QE escalation recovers much of the accuracy lost on clusters assigned to efficient models. In this setting, that combination keeps the system within 0.7 pp of the strongest model on AIME 2024 at 18% lower TPOT, and within 2.1 pp on TeleQnA, while preserving a simple path for adding or removing models through Pareto analysis.

The current framework generalises to any model pool size; hence, scaling to larger model pools is a natural next step. In a multi-model cascade the λ table provides a natural escalation order, since decreasing λ corresponds to increasing model capability. Future work also includes extending the QE cascade to multi-class decisions (accept, escalate, or request additional reasoning), online adaptation of λ to query distribution shift, and incorporating additional efficiency metrics.

Acknowledgements

This work is funded by ADAPT Centre, Trinity College Dublin, and Huawei Ireland.

Limitations

Cluster centroids and routing tables are computed offline and do not adapt to query distribution shift at inference time; updating them requires collecting new task-correctness labels and re-running the pipeline.

We use TPOT as the primary serving-efficiency metric and as the cost term in routing. TPOT captures decoding speed but does not include queueing, prefill, TTFT, network overhead, batching effects, or end-to-end latency under load. In the future, we would like to incorporate other metrics into our framework. The framework assigns a single reasoning mode per dataset. In domains where only a subset of queries require extended chain-of-thought reasoning, output lengths vary considerably within the same cluster. In such cases, TPOT alone may be an incomplete latency proxy, and additional metrics such as request throughput (requests per second) should be considered alongside error rate and latency. Moreover, TPOT measurements are specific to the 2×A100 SXM 80 GB setup used here. When loading FP8 models on A100, vLLM operates in W8A16 mode (FP8 weights, BF16 activations); on H100 with full W8A8 FP8 support, large-model throughput can improve by up to $\sim$1.6×, which would shift the Pareto frontier and potentially change the optimal λ selection. The efficiency metric η does not account for hardware-specific batching dynamics.

The framework requires all candidate models to be simultaneously resident in GPU memory; in VRAM-constrained deployments the addressable model pool must be restricted to fewer or smaller models, reducing routing flexibility.

Ethical Considerations

This work evaluates publicly available models on publicly available benchmarks. No personally identifiable information is used. Routing frameworks that reduce inference cost may lower the barrier to deploying capable models, with both positive (accessibility and energy savings) and negative (increased deployment at scale) implications.

References

AdaptKey. 2026. Adaptkey-nemotron-30b: A telecom-specialized language model. *AdaptKey-Nemotron-30b at HuggingFace*.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Mausam, and Manaal Faruqui. 2024. AutoMix: Automatically Mixing Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Transactions on Machine Learning Research*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios N Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I Jordan, Joseph E Gonzalez, and Ion Stoica. 2024. Chatbot Arena: an open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, Vienna, Austria. JMLR.org.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. 2025a. Learning to Route LLMs with Confidence Tokens. In *Forty-second International Conference on Machine Learning*.

Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. 2025b. Confident or seek stronger: Exploring uncertainty-based on-device LLM routing from benchmarking to generalization. *arXiv [cs.CL]*.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V S Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *The Twelfth International Conference on Learning Representations (ICLR)*.

António Farinhas, Nuno M Guerreiro, Sweta Agrawal, Ricardo Rei, and Andre Martins. 2025. Translate smart, not hard: Cascaded translation systems with quality-aware deferral. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26730–26744, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. 2026. Universal model routing for efficient LLM inference. In *The Fourteenth International Conference on Learning Representations*.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, and 200 others. 2025. Gemma 3 Technical Report. *arXiv [cs.CL]*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient

memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, New York, NY, USA. ACM.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from Transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, pages 19274–19286.

Ali Maatouk, Fadhel Ayed, Nicola Piovesan, Antonio De Domenico, Merouane Debbah, and Zhi-Quan Luo. 2025. TeleQnA: A Benchmark Dataset to Assess Large Language Models Telecommunications Knowledge. *IEEE Network*.

Yasmin Moslem and John D Kelleher. 2026. Dynamic model routing and cascading for efficient LLM inference: A survey. *arXiv [cs.NI]*.

NVIDIA. 2026. NVIDIA Advances Autonomous Networks With Agentic AI Blueprints and Telco Reasoning Models. *NVIDIA Blog*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to Route LLMs with Preference Data. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Runyu Peng, Yunhua Zhou, Kai Lv, Yang Gao, Qipeng Guo, and Xipeng Qiu. 2025. Firewall routing: Blocking leads to better hybrid inference for LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6540–6565, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter J Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, Guanhao Zhao, Fei Wang, and Runze Wu. 2025. IRT-router: Effective and interpretable multi-LLM routing via item response theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15629–15644, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To CoT or not to CoT? Chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*.

Jiayuan Su, Fulin Lin, Zhaopeng Feng, Han Zheng, Teng Wang, Zhenyu Xiao, Xinlong Zhao, Zuozhu Liu, Lu Cheng, and Hongwei Wang. 2025. CP-Router: An uncertainty-aware Router between LLM and LRM. *arXiv [cs.CL]*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MINILM: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, number Article 485 in NIPS'20, pages 5776–5788, Red Hook, NY, USA. Curran Associates Inc.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv [cs.CL]*.

Jieting Xiao, Yun Lin, Huizhen Qiu, Rui Ma, Chen Zhong, Dongyang Xu, Xiao Long, Chaoyu Zhang, Qiaobo Hao, Ding Zou, Zhiguo Yang, Yanqin Gao, and Fang Tan. 2026. TeleCom-Bench: How Far Are Large Language Models from Industrial Telecommunication Applications? *arXiv [cs.AI]*.

Sen Xu, Yi Zhou, Wei Wang, Jixin Min, Zhibin Yin, Yingwei Dai, Shixi Liu, Lianyu Pang, Yirong Chen, and Junlin Zhang. 2025. Tiny model, big logic: Diversity-driven optimization elicits large-model reasoning ability in VibeThinker-1.5B. *arXiv [cs.AI]*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, and 46 others. 2025. Qwen3 Technical Report. *arXiv [cs.CL]*.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. AdaptThink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaoyun Zhang, Jingqing Ruan, Xing Ma, Yawen Zhu, Haodong Zhao, Hao Li, Jiansong Chen, Ke Zeng, and Xunliang Cai. 2025b. When to continue thinking: Adaptive thinking mode switching for efficient reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5808–5828, Stroudsburg, PA, USA. Association for Computational Linguistics.

A Inference Setup

Server configuration. All experiments use vLLM Server (Kwon et al., 2023) on 2× A100 SXM 80 GB GPUs. The maximum generation length is 40,960 tokens for AIME (extended chain-of-thought output) and 1,024 tokens for TeleQnA. Following vLLM recommendations, `-max-model-len` is configured to account for both input and output lengths. Requests are served with `-max-concurrency 32` and `-seed 0` for reproducibility. We averaged accuracy and TPOT over 5 runs on each dataset.

Sampling parameters. For AIME models operating in thinking mode (VibeThinker-1.5B and Qwen3-30B-A3B), we follow the Qwen team's recommended settings: temperature 0.6, top-p 0.95, top-k 20, and min-p 0. For TeleQnA models (Qwen3-4B-Instruct and Gemma4-26B-it), which operate without explicit reasoning, we use the recommended non-thinking settings: temperature 0.7, top-p 0.8, top-k 20, and min-p 0. To account for the nondeterministic feature of LLMs, inference is run over datasets 5 times, and then accuracy and TPOT scores are averaged over the 5 runs.

Model reasoning modes. Sophisticated chain-of-thought reasoning primarily improves performance on complex maths and logic tasks, with limited gains for other tasks (Wei et al., 2022; Sprague et al., 2025). For the AIME dataset, VibeThinker-1.5B and Qwen3-30B-A3B generate explicit chain-of-thought reasoning wrapped in `<think>` tags. For the TeleQnA dataset, Qwen3-4B-Instruct produces direct answers without think tags, and Gemma4-26B-it has thinking disabled. For both models, we encourage generation of an "explanation" before generating the numerical answer. When we tried to enable the reasoning mode for Gemma4-26B-it for the TeleQnA dataset, the accuracy gain was minimal (≈3 pp) compared to just asking for the brief explanation; however, the output was much longer with explicit reasoning, which resulted in several times slower request throughput. Hence, we concluded that explicitly enabling reasoning is both unbeneficial and inefficient for the TeleQnA dataset.

Model Prompts

All models receive the dataset question as their user prompt, with a dataset-specific instruction appended to standardise the output format.

AIME prompt suffix. The following suffix is appended after each AIME question:

```
\n\nThink briefly about the approach, then provide
your final answer as a number.
```

TeleQnA prompt suffix. The following suffix is appended after each TeleQnA question:

```
\nProvide the answer in the following format:
Explanation: <one_sentence_explanation>
Answer: <choice_number_only>
0. F1 interface
1. E2 interface
2. CPRI interface
3. O1 interface
4. A1 interface
```

The answer choices shown are illustrative; the actual choices are substituted per question from the dataset.

B Framework Extensibility: AIME Pool Expansion

The framework is designed to handle a changing model pool without manual reconfiguration. When a new model is introduced, Pareto analysis reruns on the existing training corpus and either discards the new model as dominated or promotes it into the routing pool, updating routing tables automatically. We demonstrate both cases by extending the two-model AIME pool (V, Q3-30B) with Qwen3-4B-Thinking-2507-FP8 and Qwen3.5-35B-A3B-FP8.

Table 9 shows per-model statistics for the extensibility experiment on AIME 2024.

Dominated model (Qwen3-4B). Qwen3-4B-Thinking-2507-FP8 (avg. TPOT 24.99 ms, $\mathrm{Cost}_{\mathrm{norm}}$=1.019) has strictly higher error rates than Q3-30B in all clusters. It is Pareto-dominated and never selected for any λ.

Pareto-superior model (Qwen3.5-35B-A3B-FP8). Q3.5 (avg. TPOT 17.24 ms, $\mathrm{Cost}_{\mathrm{norm}}$=0.52) achieves lower error rates than both V and Q3-30B in all clusters; at λ=0.06 it wins all three. Table 10 shows a Pareto improvement over the two-model configuration: 89.3% accuracy at 11.0 ms vs. 86.4% at 9.5 ms, and a small improvement over always using Q3-30B (89.1% at 11.8 ms).

C QE Cascade Classifiers

We fine-tune ModernBERT-base (Warner et al., 2025) as a binary sequence classifier with labels *Accept* (the efficient model's answer is correct) and *Route* (escalate to the strong model).

Input format. For AIME, the classifier receives the concatenation `[query] [SEP] [model_output] [SEP] [num_output_tokens]`, where the model output is truncated to the last

Model	$\mathrm{Cost}_{\mathrm{norm}}$	C0 Err	C1 Err	C2 Err
V-1.5B	0.000	0.130	0.083	0.182
Q3.5-35B	0.520	0.046	0.042	0.077
Q3-30B	1.000	0.063	0.031	0.083
Q3-4B*	1.019	0.090	0.054	0.147

*Pareto-dominated: never selected for any λ.

Table 9: Per-model statistics for the AIME extensibility experiment (training data). $\mathrm{Cost}_{\mathrm{norm}}$ normalises TPOT over the two-model base pool (V and Q3-30B). Q3-4B has $\mathrm{Cost}_{\mathrm{norm}}>1$ and higher per-cluster error than Q3-30B in all clusters, making it Pareto-dominated.

Configuration	Train Acc	Train TPOT	Test Acc	Test TPOT
Always Q3-30B	94.4%	24.8 ms	89.1%	11.8 ms
2-model λ=0.06 (Q3-30B/V/Q3-30B)	92.1%	18.4 ms	86.4%	9.5 ms
3-model λ=0.06 (Q3.5×3)	**94.5%**	**17.3 ms**	**89.3%**	**11.0 ms**
Always V	87.3%	9.2 ms	76.7%	4.8 ms

Table 10: Extensibility: two-model vs. three-model routing on AIME 2024. Adding Qwen3.5-35B-A3B-FP8 (Q3.5) produces a Pareto improvement over the two-model configuration on both train and test.

1,000 words. Including the output-length token makes the chain-of-thought length (a proxy for model confidence) directly available to the classifier. TeleQnA model outputs are considerably shorter (average 29 words / 40 tokens on the test set, median 28 words / 38 tokens); no truncation is applied.

Label construction. Labels are derived from the efficient model's exact-match accuracy on each training query: a correct answer is labelled *Accept* (1) and an incorrect answer is labelled *Route* (0). Class imbalance is addressed with class-weighted `CrossEntropyLoss`.

Training data. For AIME, training examples are drawn from AIME 1983–2023 (921 queries), with model outputs from both VibeThinker-1.5B and Qwen3-4B-Instruct (1 run each), yielding $921 \times 2 = 1{,}842$ examples. For TeleQnA, training examples are drawn from 9,000 queries with outputs from Qwen3-4B-Instruct across 5 runs ($9{,}000 \times 5 = 45{,}000$ examples).

Hyperparameters. Both classifiers are trained with AdamW (fused implementation), batch size 64, and up to 10 epochs with early stopping (patience 4, best model selected by macro-F1). Learning rates are 5e-5 (AIME) and 2e-5 (TeleQnA, chosen by experimentation).

D QE Classifier Ablation (AIME)

As an ablation, we isolate the QE classifier's contribution independently of Stage 1 routing decisions. All 30 AIME 2024 test queries are first handled by an efficient model; the classifier then decides which outputs to escalate to Q3-30B, without any cluster-level pre-routing. This separates the classifier's quality signal from the routing benefit already obtained in Stage 1.

Table 11 shows results for two efficient models. The VibeThinker cascade achieves classifier accuracy 0.97 and raises system accuracy from 0.80 to 0.93 with 7 escalations. The Qwen3-4B cascade escalates more (14/30, reflecting its lower baseline), achieving system accuracy 0.90 with classifier accuracy 0.87.

Efficient model	Routed / 30	System Accuracy
VibeThinker-1.5B (baseline)	—	0.80
+ QE cascade	7	**0.93**
Qwen3-4B-Instruct (baseline)	—	0.60
+ QE cascade	14	**0.90**

Table 11: Standalone QE cascade results on AIME 2024 (30 queries). Each configuration starts all queries at the efficient model. The QE classifier decides which outputs to escalate to Qwen3-30B-A3B-FP8 (Q3-30B). "Routed" is the number escalated.

This ablation confirms that the QE classifier is a valuable component on its own. The two-stage setup is more efficient overall: Stage 1 avoids wasted generations by routing hard queries directly to the strong model before any efficient-model inference, while Stage 2 recovers residual accuracy losses. Neither stage alone achieves the latency–accuracy balance of the combined system.

E QE Cascade Per-Run Results

E.1 AIME

Table 12 shows per-run statistics for the QE cascade on AIME C1 (10 queries) under the combined Stage 1+2 system. Each row corresponds to one VibeThinker-1.5B inference run; Q3-30B accuracy on escalated queries is itself averaged over 5 Q3-30B inference runs to obtain a stable estimate. The outcomes are near-deterministic: runs 0–2 each escalate exactly one query (FP = 0, FN = 0, C1 Acc = 10/10) and runs 3–4 escalate none (FP = 0, FN = 1, C1 Acc = 9/10), reflecting the small cluster size.

E.2 TeleQnA

With 590 C0 queries, the TeleQnA results yield meaningful run-to-run variance.

Table 13 reports per-run statistics for the QE cascade on TeleQnA C0 (590 queries). C0 accuracy ranges from 73.4% to 74.5% across runs (spread: 1.1 pp), confirming the stability of the classifier. The false discovery rate (the fraction

Run	Routed	FP/FN	Q3-30B Acc	C1 Acc
Run 0	1	0/0	1/1	10/10
Run 1	1	0/0	1/1	10/10
Run 2	1	0/0	1/1	10/10
Run 3	0	0/1	—	9/10
Run 4	0	0/1	—	9/10
Avg	**0.6**	**0/0.4**		**96%**

Table 12: Per-run QE cascade results on AIME C1 (10 queries, 5 runs $\times$ 5 reps). VibeThinker-1.5B outputs are classified by ModernBERT; rejected outputs are escalated to Q3. FP: correct VibeThinker answers unnecessarily escalated; FN: incorrect VibeThinker answers accepted by the classifier. Q3-30B Acc is the accuracy of Q3-30B on the escalated set (correct/total escalated); "—" indicates no escalations occurred. C1 Acc is the overall C1 cluster accuracy after the cascade.

of escalations that are correct Q3-4B answers escalated unnecessarily) averages $104/202.4 \approx 51\%$, indicating that the classifier errs on the side of caution. The TPOT cost of these escalations is partly offset by the accuracy recovered after roughly 98 true-positive detections per run ($202.4 - 104$). The classifier additionally accepts on average 85 incorrect answers per run (false negatives, Table 13). Approaches to improving classifier quality include using more data or training ModernBERT-large instead of ModernBERT-base.

Run	Routed	FP/FN	G-26B Acc	C0 Acc
Run 0	199	100/86	0.671	74.3%
Run 1	205	112/90	0.683	73.7%
Run 2	206	106/83	0.662	74.1%
Run 3	200	96/80	0.649	74.5%
Run 4	202	104/84	0.638	73.4%
Avg	**202.4**	**104/85**		**74.0%**

Table 13: Per-run QE cascade results on TeleQnA C0 (590 queries, 5 runs $\times$ 5 reps). Q3-4B outputs are classified by ModernBERT; rejected outputs are escalated to G-26B. FP: correct Q3-4B answers unnecessarily escalated; FN: incorrect Q3-4B answers accepted by the classifier. G-26B Acc is measured over the escalated set (TP + FP), which is dominated by hard TP queries, explaining the lower values relative to its overall accuracy of 0.777.

F Routing and Cascading Overhead Analysis

Clustering and routing overhead. Clustering and routing decisions add negligible latency to inference. At query time, each input is assigned to the nearest centroid via a single *all-MiniLM-L6-v2* embedding pass ($\approx$14,200 sentences/s on a single GPU); the subsequent $\arg\min$ over the model pool is an in-memory lookup with negligible cost.

QE classifier overhead. Across 5 runs on TeleQnA C0 (2,950 classifications total), the classifier processes queries at an average of 48.3 queries/s (107.7 s wall-clock time), corresponding to approximately 20.7 ms per query. This figure reflects sequential (batch size 1) inference in float32; batched execution or mixed-precision (bfloat16) inference would reduce this overhead. TeleQnA C0 outputs average 39.5 tokens per query (median 38.0), so the classifier adds approximately 0.52 ms per output token, small relative to the 15–25 ms/token generation TPOT in this setup. For efficiently training ModernBERT, we recommend enabling flash attention.