AUTONOMY AND PERFORMANCE TRADE-OFF IN ADAPTIVE LLM CASCADES

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an adaptive system where two models, a large teacher and a small student, form a cascade in an online deployment scenario. As the queries stream in, a router decides whether the teacher model is needed for each query. We continually update the router and distill from the stronger, larger model to the smaller student, whenever the larger model is queried. This creates a non-stationary system, with both the router and the student constantly adapting. The process trades off the overall performance of the system and the autonomy of the small model, which reflects the privacy of the computation. We treat routing as a contextual bandit problem, and experiment with several non-stationary data stream distributions. Experiments demonstrate the effectiveness of our method, and show that competing incentives can be balanced to significantly advance the Pareto efficiency of LLM serving, even without access to the teacher's logits (i.e., with API models).

1 INTRODUCTION

Local deployment of large language models (LLMs) is often resource-constrained, creating a steep performance trade-off compared to cloud deployments, the currently common avenue for LLM use. However, different users require different capabilities, and individual users rarely, if at all, require the full spectrum of capabilities of frontier models. A local, resource-constrained deployment can serve either an individual or many members of a privacy-focused organization, adapting to the users' query traffic and keeping local an increasing fraction of queries. Such a specialized model, as it excels in a narrower set of capabilities, could be significantly smaller and resource efficient. However, the exact boundaries of this specialization vary greatly between users and over time, and are challenging to specify in advance. We propose to learn adaptive routing to balance performance and autonomy, while continually distilling a cloud model into an increasingly specialized small, local model.

Our system, EVOLVING CASCADE, sequentially handles a stream of queries, deciding whether to route to the stronger model for each query, similar to an LLM cascade (Chen et al., 2024). In contrast to static cascades, our system tightly integrates continual adaptation (Figure 1). We formulate routing as a contextual bandit problem, and develop a continually evolving router based on LinUCB (Li et al., 2010). Concurrently, the small model adapts through student learning, using a self-distillation technique that directly distills from a black-box cloud model. As Figure 1 shows, each teacher call

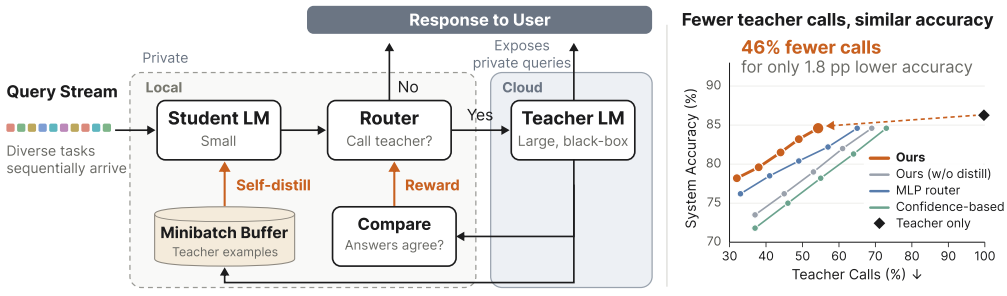


Figure 1: Overview of EVOLVING CASCADE. Our system adapts continually to route between the small and large models, significantly reducing large model calls, while retaining its performance.

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.

answers the current query and provides a demonstration for distillation, while the router adapts as the student's capabilities change. We specifically design our system to handle the unique challenges of an evolving LLM cascade: adapting the router to a changing student, learning from feedback that arrives only when the teacher is called, and controlling the performance-autonomy trade-off.

We develop a suite of evaluation settings that simulate challenges of real-life query traffic that the existing LLM cascade literature does not study (Jung et al., 2025; Nie et al., 2024; Wu et al., 2026). Our evaluation data streams include queries of various difficulty levels and distribution shifts that simulate diversity and non-stationary data streams (Wang et al., 2026). We vary the degree of non-stationarity across three types of streams: a stationary mixture of six tasks, abrupt swaps between easy and hard tasks, and incremental introduction of new tasks. These data streams together reveal issues with the existing cascade systems, which our approach addresses.

Across the evaluation settings, our system dominates the Pareto frontiers of existing cascade baselines, saving up to 46% of cloud model calls with only a 1.8-percentage-point accuracy gap relative to the cloud model (Gemini-3.1-pro) (Figure 1).

2 Related Work

Online LLM Cascades LLM cascades answer queries with a small model first and route to a stronger model when needed (Chen et al., 2024; Jung et al., 2025). The router, typically a trained classifier or a threshold on the small model's confidence, is developed before deployment. Both the model and router then stay fixed. Online cascades that update both components remain underexplored. Nie et al. (2024) build an online cascade only for single-task classification with a small label space. Wu et al. (2026) use a confidence-based rule from Jung et al. (2025) calibrated on a static dev set. Every evaluation focuses on a single task type, where the model accumulates strategies specific to this type and accesses them via in-context learning. We instead develop a contextual-bandit router that adapts alongside a continually improving small model on a diverse data stream with distribution shifts. Wei et al. (2025) also study routing and contextual bandits, but keep the LLMs fixed, train offline on static logs, and assume every routing decision receives a reward during training. In our setting, a reward is observed only if the large model is called.

Privacy Exposure from External API Calls User queries can contain sensitive information, creating privacy risks when sent to cloud model providers (Lin et al., 2025). LLM privacy research has proposed local handling when possible and masking sensitive information for necessary external calls as complementary ways to reduce exposure (Zhang et al., 2026). Our system strengthens the first by identifying queries within the local model's capacity and continually expanding its capacity to reduce external exposure. Practical applications can combine our system with prompt-sanitization techniques (Kan et al., 2023; Chong et al., 2025), to minimize the privacy risk from the remaining calls that still require the cloud model.

Similarity Phenomenon and Model Specialization Real-world LLM traffic contains queries requiring similar skills, such as table analysis or arithmetic reasoning (Belcak et al., 2025; Wu et al., 2026). This repetition favors small, specialized models (Belcak et al., 2025), especially local ones serving an individual or group. LLM traffic also exhibits temporal locality: specific task types dominate certain time windows (Wang et al., 2026). Our evaluation suite captures both properties.

Self-Distillation for Continual Learning Fine-tuning on new tasks can cause catastrophic forgetting (Huang et al., 2024; Yang et al., 2024). Self-distillation mitigates this by training a model to match its own distribution conditioned on high-quality demonstrations (the self-teacher) (Yang et al., 2024; Shenfeld et al., 2026). Shenfeld et al. (2026) show that a model can learn new tasks without forgetting with three task datasets but train them at separate stages with predefined learning rate decay and up to two epochs. We study whether this transfers to long, diverse data streams with distribution shifts, where the cascade does not know when the data stream will end or shift distributions.

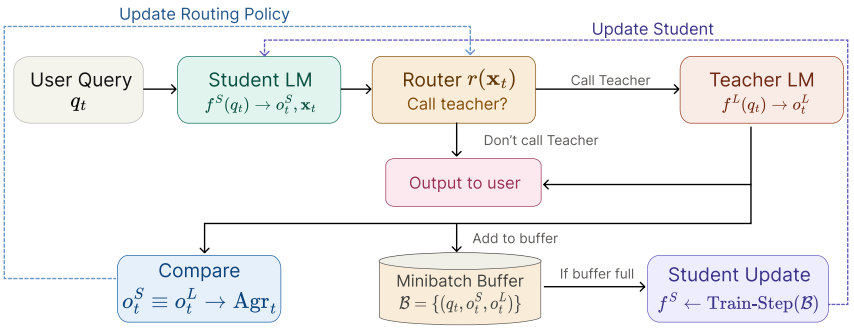


Figure 2: Illustration of EVOLVING CASCADE.

3 METHOD

EVOLVING CASCADE transforms LLM cascades with two components that adapt online: a student that adapts continually through distillation, and a router that evolves as the student's capacity expands. All learning happens online with no separate training and deployment phases.

3.1 THE EVOLVING CASCADE FRAMEWORK

Figure 2 illustrates our framework. The system observes a stream of user queries $\mathcal{Q} = \{q_t \mid t \geq 1\}$, generating an output answer for each query before processing the next. The student f^S is a locally deployed small model, and the teacher f^L is a strong model potentially behind an opaque API. Upon receiving a query q_t, the system first sends it to the student f^S, which produces the output o_t^S and context vector $\mathbf{x}_t$ based on its hidden state (Appendix D.1). The router takes $\mathbf{x}_t$ and decides whether to further send q_t to the stronger model f^L. We train the router as a contextual bandit policy (Section 3.3), but the framework is general, potentially admitting other approaches, such as training an MLP classifier or calibrating a confidence threshold. If the teacher is invoked, a comparator function (Section 5) checks if the student output agrees with the teacher output (i.e., $o_t^S \equiv o_t^L$), producing an agreement label Agr_t that is used to improve the router for future queries. Whenever the teacher is called, the system saves the query and the outputs of the student and teacher to a minibatch buffer of size b_s. When the buffer is full, a student training step is triggered (Section 3.2).

3.2 STUDENT TRAINING VIA DISTILLATION

We use self-distillation to fine-tune the student (Shenfeld et al., 2026). We construct a 'self-teacher' distribution $\tilde{f}^S(\cdot \mid q_t, o_t^L)$ by prompting the student itself with both the query q_t and the teacher output o_t^L used as in-context demonstration. The student distribution $f^S(\cdot \mid q_t)$ is obtained by the student prompted with only the query and no demonstration. The training objective is to match the student distribution with the self-teacher distribution by minimizing the KL divergence between the student and self-teacher distributions. This process does not assume access to the teacher's parameters or output logits, making it suitable for learning from a black-box teacher.

Consistent with Shenfeld et al. (2026), we use a single student rollout for distillation. Self-distillation also supports sampling multiple student rollouts $y \sim f_\theta^S(\cdot \mid q_t)$ per prompt, but doing so incurs high inference costs. We reuse the student rollout o_t^S already produced when the query was served, so that learning incurs no additional inference cost compared to a static LLM cascade.

3.3 ROUTER LEARNING

We cast router learning as a contextual bandit problem: whether to call the teacher depends on the context. The router is learned online from the routing decisions and their outcomes. We use LinUCB with reward and exploration choices that are specific to our scenario.

Algorithm 1 describes router learning. The router actions (also known as arms in the contextual bandit literature) are NO-T-CALL (not calling the teacher), and T-CALL (calling the teacher). The algorithm

Algorithm 1 LinUCB online learning of the router.

1: **Hyperparameters:** exploration weight $\alpha > 0$; regularization weight $\lambda > 0$; trade-off control $\tau \in (0, 1)$
2: **Router decisions:** $\{$T-CALL, NO-T-CALL$\}$
3: **Initialize:** $\mathbf{A}_1 \leftarrow \lambda \mathbf{I}_d$; $\mathbf{b}_1 \leftarrow \mathbf{0}_d$ ▷ $\lambda \mathbf{I}_d$ is a scaled identity matrix; $\mathbf{0}_d$ is a zero vector
4: **for** $t = 1, 2, \ldots$ **do**
5: Student produces o_t^S and context $\mathbf{x}_t \in \mathbb{R}^d$ for q_t
6: $\hat{\boldsymbol{\theta}}_t \leftarrow \mathbf{A}_t^{-1}\mathbf{b}_t$ ▷ Fit a ridge regression model for reward estimate
7: $\hat{r}_t \leftarrow \hat{\boldsymbol{\theta}}_t^\top \mathbf{x}_t$ ▷ Estimated reward of calling the teacher
8: $u_t \leftarrow \sqrt{\mathbf{x}_t^\top \mathbf{A}_t^{-1} \mathbf{x}_t}$ ▷ Uncertainty under the current context
9: $\text{UCB}_t \leftarrow \hat{r}_t + \alpha u_t$ ▷ Upper confidence bound on the estimated reward
10: **if** $\text{UCB}_t > 0$ **then**
11: T-CALL: call the teacher to obtain o_t^L; output o_t^L
12: $\text{Agr}_t \leftarrow \mathbf{1}[o_t^S \equiv o_t^L]$ ▷ Agreement between the student and teacher
13: **if** $\text{Agr}_t = 1$ **then**
14: $r_t \leftarrow -1$ ▷ Negative reward for an unnecessary T-CALL
15: **else**
16: $r_t \leftarrow (1 - \tau)/\tau$ ▷ Positive reward for a useful T-CALL
17: $\mathbf{A}_{t+1} \leftarrow \mathbf{A}_t + \mathbf{x}_t \mathbf{x}_t^\top$; $\mathbf{b}_{t+1} \leftarrow \mathbf{b}_t + r_t \mathbf{x}_t$ ▷ Update the recorded contexts and rewards
18: **else**
19: NO-T-CALL: output the student's answer o_t^S ▷ Implicitly $r_t \leftarrow 0$

initializes a matrix $\mathbf{A}$ and a vector $\mathbf{b}$ with $\lambda \mathbf{I}_d$ and $\mathbf{0}_d$, where d is the dimension of the context vector. $\mathbf{A}$ and $\mathbf{b}$ together track the observed rewards and context vectors and are updated every timestep t. At timestep t, the student produces an output o_t^S and context vector $\mathbf{x}_t$. The algorithm estimates the reward of calling the teacher with a ridge regression fitted to the history of observed rewards and context vectors. The vector $\mathbf{A}_t^{-1}\mathbf{b}_t$ computes $\hat{\boldsymbol{\theta}}_t$, the fitted regression coefficients at this timestep. $\hat{r}_t = \hat{\boldsymbol{\theta}}_t^\top \mathbf{x}_t$ is then the estimated reward of a teacher call given the context $\mathbf{x}_t$. The λ in Line 3 is the standard regularization weight in ridge regression, a hyperparameter for us to choose.

Following LinUCB, we quantify uncertainty in the estimate as $u_t = \sqrt{\mathbf{x}_t^\top \mathbf{A}_t^{-1} \mathbf{x}_t}$. It is larger when the router observes fewer similar contexts and decreases as it observes more. The router adds exploration bonus αu_t to $\hat{r}_t$ to obtain the upper confidence bound UCB_t, where α is the hyperparameter that controls exploration. A high exploration coefficient α encourages the router to call the teacher regardless of the estimated reward.

The routing decision is based on whether the UCB_t is greater than 0, because we fix the reward of NO-T-CALL at 0 and do not use an exploration bonus for it (elaborated below). When the teacher is called, the cascade system checks if the student and teacher answers agree by sending the student answer to the teacher, or a predefined comparator function (Section 5), and produces the agreement label Agr_t. The call receives a positive reward $(1 - \tau)/\tau$ when the student and teacher disagree, and -1 when they agree. The router then adds $\mathbf{x}_t \mathbf{x}_t^\top$ to $\mathbf{A}_t$ and $r_t \mathbf{x}_t$ to $\mathbf{b}_t$, which is the standard update to track the observed contexts and rewards, making the $\mathbf{A}_t^{-1}\mathbf{b}_t$ in Line 6 a ridge regression fitted to the data so far. When the teacher is not called, the student output is returned, the reward is implicitly 0, and $\mathbf{A}_t$ and $\mathbf{b}_t$ remain unchanged. As the student improves on a query type, teacher calls on that type more often receive reward -1, which lowers their estimated reward and shifts subsequent queries towards NO-T-CALL.

Our algorithm is equivalent to the standard LinUCB (disjoint LinUCB) (Li et al., 2010), except that we use our specific reward formulation and force the exploration bonus on NO-T-CALL to zero.

Reward Formulation Algorithm 1 defines the reward at timestep t with $\tau \in (0, 1)$ as:

$$r_t = \begin{cases} 0 & \text{if NO-T-CALL,} \\ -1 & \text{if T-CALL and } \text{Agr}_t = 1 \\ (1 - \tau)/\tau & \text{if T-CALL and } \text{Agr}_t = 0 \end{cases}.$$

When the router chooses NO-T-CALL, the teacher output is not observed, so the system cannot determine whether the student and teacher agree. We therefore use a fixed reward of 0. When the router chooses T-CALL, the system observes the teacher output and determines whether the two

models agree. The reward for a teacher call upon disagreement should be higher than the NO-T-CALL reward (thus positive), because the teacher call prevented the system from deviating from the teacher model it aims to match. On the other hand, the reward for an unnecessary teacher call (when both models agree) should be negative. Our definition fixes the negative reward at -1, without loss of generality, because only the ratio between the positive and negative rewards matters (Appendix B.1).

The two teacher-call rewards encode the performance-autonomy trade-off through the single bounded hyperparameter $\tau \in (0, 1)$. Increasing τ lowers the positive reward for overriding the student and therefore favors fewer teacher calls. Defining the positive reward as $(1-\tau)/\tau$ gives τ an interpretation similar to the risk-tolerance hyperparameter in prior cascade works (Jung et al., 2025). If the router has seen similar queries enough times that the uncertainty is small and the reward estimate recovers the true expected reward (the asymptotic interpretation), τ is an upper bound on the probability of the student disagreeing with the teacher when the router chooses NO-T-CALL (Appendix B.3).

Exploration Weighting Unlike the standard LinUCB, which would maintain exploration bonuses for both actions, we force the exploration bonus of NO-T-CALL to 0. Because NO-T-CALL's reward is fixed and predefined, choosing NO-T-CALL reveals no new information about its reward (useless exploration). The only useful exploration is choosing T-CALL when the router has a high uncertainty about this action. With this modification, NO-T-CALL's exploration bonus and reward are both 0, so its UCB is always 0. The router's decision is therefore the simple comparison in Line 10 of Algorithm 1: call the teacher exactly when T-CALL's UCB is positive. This modification makes the exploration weight α monotonically control how much the router engages in useful exploration (calling T-CALL upon high uncertainty). This does not hold for standard LinUCB, and the modification gives better performance empirically. Appendix A.1 gives the full argument and supporting experiments.

4 EVALUATION SUITE FOR EVOLVING CASCADES

Evaluating an evolving cascade requires streaming queries sequentially rather than treating them as i.i.d. instances. Both the student and the router change as queries arrive, so the system's behavior on each query depends on the earlier queries in the stream. No single stream captures every failure mode, so we construct streams that differ in their task datasets and distribution shifts.

Task Datasets We use six tasks spanning different difficulty levels to build the streams: **Banking77** (challenging classification in banking-domain queries) (Casanueva et al., 2020), **CLINC150** (challenging classification in customer service queries) (Larson et al., 2019), **GSM** (math questions from GSM-symbolic, an augmented version of GSM8K) (Mirzadeh et al., 2025), **Tablebench** (table understanding and analysis) (Wu et al., 2025), **Chemistry** (chemistry reasoning) (Feng et al., 2024), and **MiniCheck** (document-grounded fact-checking) (Tang et al., 2024).

The tasks differ in the initial performance gap between the student and the teacher and in how quickly the student closes that gap through distillation. The student adapts fastest on the two classification tasks, and slowest on Tablebench and Chemistry. MiniCheck serves as a control: the off-the-shelf student already matches the teacher, so a good router should quickly stop making teacher calls.

Data Streams We propose an evaluation suite of five data streams organized into three families, **S1**–**S3**, which differ in the type and extent of distribution shift. The suite contains one S1 stream, two S2 streams, and two S3 streams. Figure 3 provides an illustration, and Appendix D.2 provides the exact composition of each stream.[1]

- **S1: Balanced** A single-stage stream randomly mixing all six tasks (3,000 instances each unless fewer are available).
- **S2: Abrupt Task Swap** A three-stage stream that tests adaptation to abrupt changes in query traffic. The first 1,300 instances contain only Task A, the next 1,300 only Task B, and the final 1,300 return to Task A. The two S2 streams in our evaluation suite pair GSM with Tablebench and reverse their order: easy–hard–easy or hard–easy–hard. Tablebench is the hard task in both streams. These shifts test whether the student retains what it learned and whether the router recognizes a returning task that the student can now handle, rather than invoking unnecessary

[1]We focus on three streams, one from each family, in the main paper, and report other streams in the appendix.

teacher calls. We additionally evaluate in Appendix C.4 two S2 variants that use Banking77 instead of GSM as the easy task. They lead to the same conclusions.

S3: Incremental Task Introduction A four-stage stream that introduces new tasks while retaining tasks from earlier stages. All stages contain the same number of instances. Stage 1 is a balanced mixture of two tasks. Each later stage draws half of its queries from a newly introduced task and splits the other half equally among the tasks seen so far. The easy-first stream starts with Banking77 and GSM; the hard-first stream starts with Chemistry and Tablebench.

Metrics We evaluate a cascade system by its average **accuracy** and **teacher call rate** across a data stream. Higher accuracy and fewer teacher calls are better. Teacher call rate reflects the student's autonomy. Given two systems with the same accuracy, a better system should have its student solve more queries on its own instead of invoking the teacher. We focus on the teacher call rate rather than cost or latency because it is agnostic to implementation details (e.g., hardware and API pricing), and because each teacher call is a query exposed to the model provider, increasing the privacy risk.

For in-depth analysis, we also report two agreement rates. The **system agreement rate** measures how often the final system output agrees with the teacher and closely correlates with system accuracy. The **student agreement rate** measures how often the student output agrees with the teacher, regardless of the routing decision,[2] reflecting the student's ability.

Contextual bandit research typically also analyzes *regret*, the difference in reward between a policy and the optimal policy that always takes the action with the highest expected reward (Li et al., 2010). Regret is not well-suited to our scenario because rewards are not comparable across methods and trade-off levels, and routing actions change the student and thus future rewards (Appendix A.2).

5 Experiment Setup

We use Qwen-2.5-3B-Instruct as the student and Gemini-3.1-pro as the teacher. Appendix C.1 reports similar results with Qwen-3-1.7B, a thinking model, as the student.[3]

Comparator Function The comparator determines whether the student and teacher answers agree (Section 3.1). In a deployment, the system can obtain this signal by asking the teacher to both generate its own answer and compare it with the student's answer. For our experiments, we generate and cache the teacher answer to each query just once because the same queries recur across many baseline and ablation runs. During a run, we compare the student output with the cached teacher answer using the answer-extraction and comparison procedure provided by the corresponding task dataset. This gives agreement signals for our experiments without repeating teacher calls across runs. Critically, this is an optimization measure only, and does not impact the outcome of our evaluations.

Baselines We compare our system with three baselines: **(B1) w/o distill** uses our adaptive router but keeps the student frozen (i.e., no distillation); **(B2) Conf.** routes when the normalized generation probability (i.e., as confidence) falls below a calibrated threshold (Jung et al., 2025; Wu et al., 2026), which is calibrated during a cold-start state with 500 instances, because we do not have a development set; and **(B3) MLP** following Nie et al. (2024), this method trains an MLP to predict whether to route to the teacher. The original work focuses on classification tasks, so their context vector is the class probabilities. We instead use our context vector, defined in Section 3.

B2 and B3 each combine a router from related work with our distillation procedure. Appendix A.1 ablates zeroing out NO-T-CALL's exploration bonus.

6 Results

EVOLVING CASCADE provides an avenue to trade off performance and autonomy. We conduct multiple runs varying the trade-off hyperparameter to obtain the Pareto frontiers of the different approaches (Figure 4). We also compare the teacher call rates these methods achieve at the same performance levels (same system agreement rates and accuracies) in Table 1.

[2] We obtain the teacher output for computing this metric even when the cascade does not call the teacher.

[3] Appendix D provides further details about the models, training, and baselines.

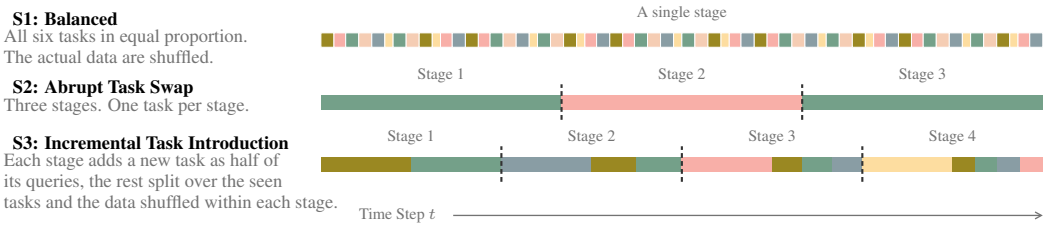


Figure 3: Illustration of the three data stream families. Each colored block denotes a task. The block width represents the number of task instances. Queries are shuffled within each stage (not shown).

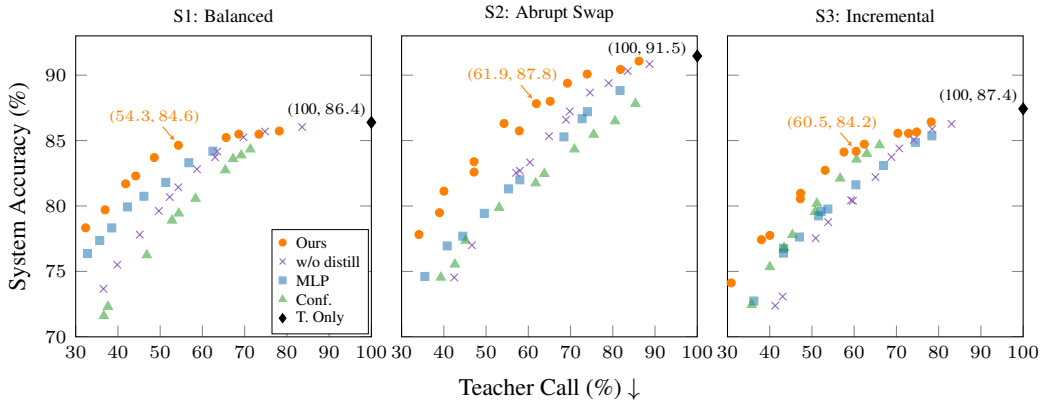


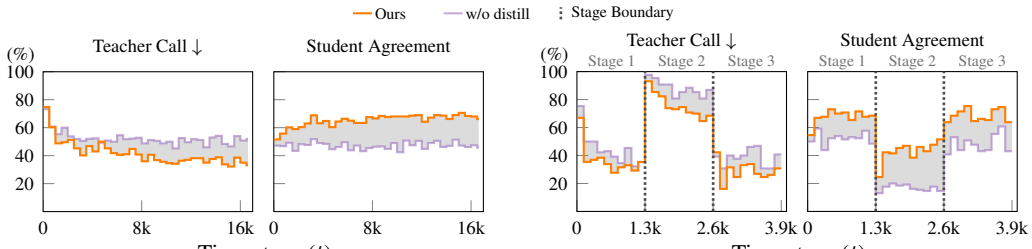
Figure 4: Performance–Autonomy trade-off across the runs. S2 and S3 refer to the Easy–Hard–Easy and Easy–First variants of their respective families. Appendix C shows the remaining data streams. Points marked with coordinates are representative ones discussed in this section. 'T. Only' means always using the teacher. Appendix C reports the small model only accuracies, which are much worse than the cascades'. As the teacher calls approach 100%, all cascades converge to the teacher-only method and give little autonomy. Meaningful comparisons should focus on lower teacher call rates.

Table 1: Comparison of autonomy at the same performance levels. Each column represents runs with nearly the same system agreement rate (thus the same accuracy). 95, 90, and 80 denote agreement rates near 95%, 90%, and 80%, respectively. Tables 2, 3, and 4 in Appendix C report the exact agreement rates and accuracies for S1, S2, and S3, respectively. The baseline runs used for comparison have accuracies equal to or slightly worse than our method's, so any teacher call savings of our method cannot be attributed to it having a worse performance.

	Method	**S1: Balanced**			**S2: Abrupt Swap**			**S3: Incremental**		
Agree Rate (≈)		**95**	**90**	**80**	**95**	**90**	**80**	**95**	**90**	**80**
Teacher Call Rate ↓	**Ours**	54.3	41.8	23.2	61.9	47.2	28.2	60.5	47.2	30.9
	w/o distill	63.5	52.3	36.6	68.9	57.1	42.5	68.8	59.3	43.1
	MLP	62.5	46.2	25.4	72.7	58.0	35.5	66.9	53.8	36.2
	Conf.	67.3	54.4	36.6	80.5	63.8	39.3	63.0	50.7	35.7

6.1 Overall Findings

Figure 4 shows that our method can reach the same or better accuracy at a much lower teacher call rate than the baselines: its points lie closer to the top-left corner and form a Pareto frontier that dominates the baselines. Compared with always using the teacher (T. Only), our method saves up to 45.7% of teacher calls while staying within 1.8 percentage points (pp) of the teacher's accuracy (the annotated S1 point at 84.6% accuracy and 54.3% teacher calls). On the other hand, using the frozen small model only (S. Only in Table 2 of Appendix C) makes no teacher calls but leads to much worse



(a) S1: Balanced. (b) S2: Abrupt Swap.

Figure 5: Effects of student learning. Reported are average metrics for consecutive windows of 500 instances (for S1) and 130 instances (for S2). We compare runs with the same performance (system agreement $\approx 90\%$). 'k' on the x-axis denotes thousands. Gaps between the methods are shaded.

accuracy (e.g., 46.1% on S1). Our full system also outperforms w/o distill across the streams. At high teacher call rates, all methods converge to the teacher-only performance. The comparison there has less practical relevance because the cascades behave more similarly to a teacher-only system and grant little autonomy to the small model.

6.2 Effects of Student Learning

Compared with the baseline without student distillation (w/o distill), our full system consistently saves more teacher calls (e.g., 41.8% vs. 52.3% on S1 and 47.2% vs. 57.1% on S2 in Table 1 at the 90% agreement level). This gain comes from the intended mechanism: as the student improves, it matches the teacher better, achieving a higher agreement rate with the teacher.[4] The router recognizes this improvement and reduces teacher calls accordingly. For example, on S1: Balanced (Figure 5a), our full system makes significantly fewer teacher calls at the same overall agreement rate (and accuracy).

Learning Under Distribution Shifts The benefits of student learning also persist across distribution shifts. When a learned task disappears for an extended period and reappears, the student retains its improvement, and the router recognizes the task, avoiding excessive teacher calls. Figure 5b shows S2 (abrupt swap), where Stages 1 and 3 contain only GSM and Stage 2 only the harder Tablebench.

In Stage 1, the student in 'ours' learns quickly and achieves a higher student agreement rate than the w/o distill baseline. The router reduces the teacher call rate partly because it sees the task more (lower router exploration bonus) but more importantly because the student has improved. The baseline's teacher calls also decrease after seeing more instances, but much less than ours. In Stage 2, both methods' teacher call rates rise when the harder Tablebench queries arrive, but ours falls more substantially than the baseline's, as our student improves on this task. Finally, when GSM returns in Stage 3, our system's call rate drops to a level similar to that at the end of Stage 1 because the student has retained its improved performance on this task and the router recognizes the task as something the student already handles well. The w/o distill baseline, whose student does not improve, still incurs more teacher calls in Stage 3. Appendix C.2 gives more statistics and results on other streams.

6.3 Comparing Different Routers

Our system also consistently outperforms the baselines that have different routers. The bottom rows of Table 1 present baselines with different routers but the same student model and distillation procedure. S1 mixes tasks evenly with no distribution shifts, making it a favorable setting for the baseline routers, which are not designed for shifts. Even there, our method uses fewer teacher calls. At the matched 95% agreement level, our method uses 54.3% teacher calls, while the next-best baseline (MLP) uses 62.5%. This advantage also holds on the distribution-shift streams, where our method uses 5.3–10.8 pp fewer teacher calls than MLP. We discuss a few key patterns behind our router's advantages, and present the comprehensive discussion in Appendix C.3.

[4]We focus on agreement rate for in-depth analysis because the cascade system directly optimizes agreement with the teacher and never receives any accuracy information. Accuracy also depends on the teacher's performance and therefore does not isolate the contribution of the cascade system.

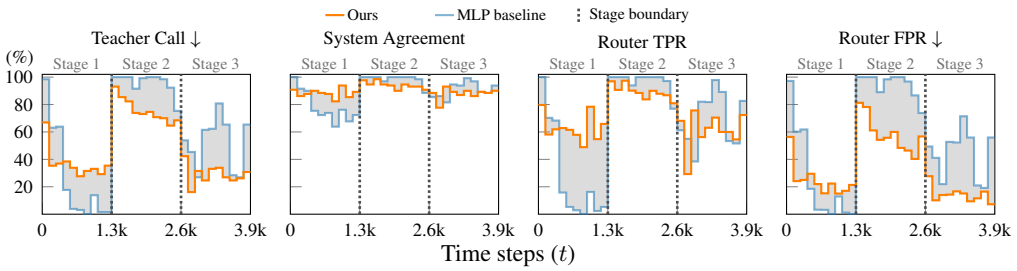


Figure 7: Ours vs. MLP on S2: Abrupt Swap at the same performance (system agreement $\approx 90\%$). Reported are average metrics for consecutive 130-instance windows.

Making Teacher Calls Only When Needed Our router outperforms the baselines by more accurately identifying the instances that need teacher calls: it avoids calls when the two models would likely agree and makes calls when the student would likely fail to match the teacher. Figure 6 compares the router true positive and false positive rates[5] for runs that achieve the same system-level performance. Our method has a lower teacher call rate and false positive rate than MLP on all three streams, while also attaining a higher true positive rate. Thus, it avoids more unnecessary calls without missing more instances that need the teacher.

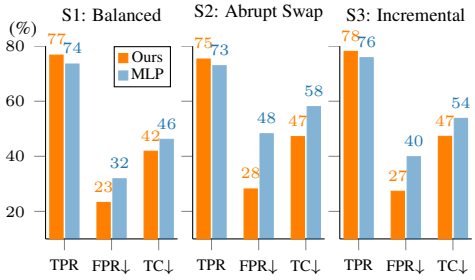


Figure 6: Router metrics across methods (system agreement $\approx 90\%$): true positive rate (TPR), false positive rate (FPR), and teacher call rate (TC).

Timely and Smooth Adaptations Figure 7 compares our method with MLP on S2 as an example. In Stage 1, our method starts at about 70% teacher calls but quickly reduces its call rate to 30–40% within the first few hundred instances as the student improves on GSM (improvement visualization in Appendix C.3). MLP over-corrects: it initially calls the teacher more often than ours, then sharply reduces its calls, missing many instances where a call would have helped (very low router true positive rate for most of Stage 1) and causing a lower system agreement rate. Both methods increase teacher calls when the harder Tablebench queries arrive in Stage 2. However, our method then reduces its call rate substantially as the student improves on the task. MLP's call rate remains high in Stage 2, but it has only a marginally higher system agreement rate than ours and a much higher router false positive rate. Finally, when GSM reappears in Stage 3, MLP fails to recognize it as a seen task the student has learned. Its teacher call rate fluctuates instead of settling, again wasting teacher calls on instances that the student can handle (spikes in router false positive rate). These patterns make MLP's use of teacher calls inefficient: it makes many more calls while reaching no better overall performance than ours.

7 Conclusion

We introduce Evolving Cascade, an LLM cascade in which the small model (student) improves continually by learning from the big model (teacher), and the router must also adapt online as the student's capacity expands. Under this framework, a teacher call is no longer only an expense: it also supplies a training example, so the two components improve together and the router grants the student more autonomy over time. The deployed system therefore discovers on its own where the student suffices, and widens that region as it serves. We make this practical with a novel combination of contextual bandit and self-distillation, with design choices that fit our evolving cascade setting. We also introduce new realistic evaluation scenarios, and use them to show that our approach holds under distribution shifts and across multiple tasks. This provides an evaluation substrate for future work on self-evolving systems in general, beyond the cascade framework. We hope our contributions will encourage future research on building local LLM deployments that sacrifice little performance and whose models continuously grow more capable and better fitted to the users they actually serve.

[5] *Positives* refer to instances where the student and teacher disagree, where the router should call the teacher.

AI USE STATEMENT

We used Claude Code and Codex for paper copyediting, polishing the writing, formatting tables and figures, and building visual illustrations. We used Claude to identify additional related works. We used Cursor and Claude Code to implement functions from human-written specifications and to refactor manually written code. All AI-assisted work was manually reviewed, revised, and verified. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe the method in Section 3, the evaluation suite in Section 4, and the experimental setup in Section 5. The appendix provides supplementary derivations in Appendix B, implementation and data stream details in Appendix D.1 and Appendix D.2, and experiment configurations in Appendix D.3. Upon acceptance, we will release the code needed to reproduce our experiments.

REFERENCES

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic AI, 2025. URL https://arxiv.org/abs/2506.02153.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019. URL https://arxiv.org/abs/1902.10486.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=cSimKw5p6R.

Chun Jie Chong, Chenxi Hou, Zhihao Yao, and Seyed Mohammadjavad Seyed Talebi. Casper: Prompt sanitization for protecting user privacy in web-based large language models. In *2025 IEEE 12th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 122–133. IEEE, November 2025. doi: 10.1109/cscloud66326.2025.00027. URL http://dx.doi.org/10.1109/CSCloud66326.2025.00027.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. SciKnowEval: Evaluating multi-level scientific knowledge of large language models, 2024. URL https://arxiv.org/abs/2406.09098.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001. URL https://aclanthology.org/H01-1069/.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.77. URL https://aclanthology.org/2024.acl-long.77/.

Jaehun Jung, Faeze Brahman, and Yejin Choi. Trust or escalate: LLM judges with provable guarantees for human agreement. In *The Thirteenth International Conference on Learning Representations*,

2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/08dabd5345b37fffcbe335bd578b15a0-Abstract-Conference.html.

Zhigang Kan, Linbo Qiao, Hao Yu, Liwen Peng, Yifu Gao, and Dongsheng Li. Protecting user privacy in remote conversational systems: A privacy-preserving framework based on text sanitization, 2023. URL https://arxiv.org/abs/2306.08223.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1311–1316, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1131. URL https://aclanthology.org/D19-1131/.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pp. 661–670, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605587998. doi: 10.1145/1772690.1772758. URL https://doi.org/10.1145/1772690.1772758.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1150/.

Zheng Lin, Guanqiao Qu, Qiyuan Chen, Xianhao Chen, Zhe Chen, and Kaibin Huang. Pushing large language models to the 6G edge: Vision, challenges, and opportunities. *IEEE Communications Magazine*, 63(9):52–59, 2025. doi: 10.1109/MCOM.001.2400764. URL https://ieeexplore.ieee.org/document/11152695.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

Lunyiu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. Online cascade learning for efficient inference over streams. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 38071–38090. PMLR, 2024. URL https://proceedings.mlr.press/v235/nie24a.html.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1gTShAct7.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Liyan Tang, Philippe Laban, and Greg Durrett. MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8818–8847, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.499. URL https://aclanthology.org/2024.emnlp-main.499/.

Junxiong Wang, Fengxiang Bie, Jisen Li, Zhongzhu Zhou, Zelei Shao, Yubo Wang, Yinghui Liu, Qingyang Wu, Avner May, Sri Yanamandra, Ce Zhang, Tri Dao, Percy Liang, Ben Athiwaratkun, Shuaiwen Leon Song, Chenfeng Xu, and Xiaoxia Wu. When RL meets adaptive speculative training: A unified training-serving system, 2026. URL https://arxiv.org/abs/2602.06932.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In Lun-Wei Ku, Andre Martins, and

Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

Wang Wei, Tiankai Yang, Hongjie Chen, Yue Zhao, Franck Dernoncourt, Ryan A. Rossi, and Hoda Eldardiry. Learning to route LLMs from bandit feedback: One policy, many trade-offs, 2025. URL https://arxiv.org/abs/2510.07429.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Tongliang Li, Zhoujun Li, and Guanglin Niu. TableBench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25497–25506. Association for the Advancement of Artificial Intelligence, 2025. doi: 10.1609/aaai.v39i24.34739. URL https://doi.org/10.1609/aaai.v39i24.34739.

Yu Wu, Shuo Wu, Ye Tao, Yansong Li, and Anand D. Sarwate. From deferral to learning: Online in-context knowledge distillation for LLM cascades, 2026. URL https://arxiv.org/abs/2509.22984.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1028–1043. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.58. URL https://aclanthology.org/2024.acl-long.58/.

Kai Zhang, Congchao Wang, Liqian Peng, Alec Go, and Xiaozhong Liu. Privacy-preserved LLM cascade via CoT-enhanced policy learning. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pp. 955–964, 2026. doi: 10.1145/3773966.3777920. URL https://doi.org/10.1145/3773966.3777920.

A Additional Discussions

A.1 Exploration Bonus for No-T-call

Our routing algorithm assigns an exploration bonus of 0 to No-T-call (Algorithm 1). Standard LinUCB, on the other hand, has the bonus active on both arms and controls them with the same weight α. This configuration makes LinUCB over-select No-T-call in our setting. Note that No-T-call carries a constant reward, so observing it never changes the router's estimate of that arm. Yet its exploration bonus still inflates its UCB, driving the router to select it more often than the expected reward would suggest, making useless explorations. The obvious remedy, lowering α to suppress that useless exploration, would also suppress the useful exploration on T-call. When the system encounters new query types and thus has high uncertainty about them, it should pick the teacher arm more often. Lowering α reduces this exploration, thus hurting the router's adaptation to new query types. More generally, α does not offer a good control of the useful exploration. Raising α drives the exploration of both arms more with the effects canceling out, so we cannot adjust α to control how much useful exploration happens. Assigning No-T-call no exploration bonus fixes this. It makes α monotonically control the useful exploration: a larger α always means more exploration on T-call and the effect is most pronounced when uncertainty is high (because exploration bonus is the product of them, αu_t).

Figure 8 sweeps α on the TREC task we used for development (Li & Roth, 2002; Hovy et al., 2001) (not part of our evaluation streams). It presents the teacher call rate over the stream's first 100 instances, where the router's estimates are still uncertain and exploration matters most. We repeat the sweep under both formulations at nine values of the trade-off hyperparameter τ. τ sets the overall autonomy–performance preference (Section 3). It should not be used to control exploration, because it governs the trade-off beyond exploration. Lowering τ to induce more calls would over-call the teacher throughout, even after the uncertainty falls and the router is confident about the student's correctness.

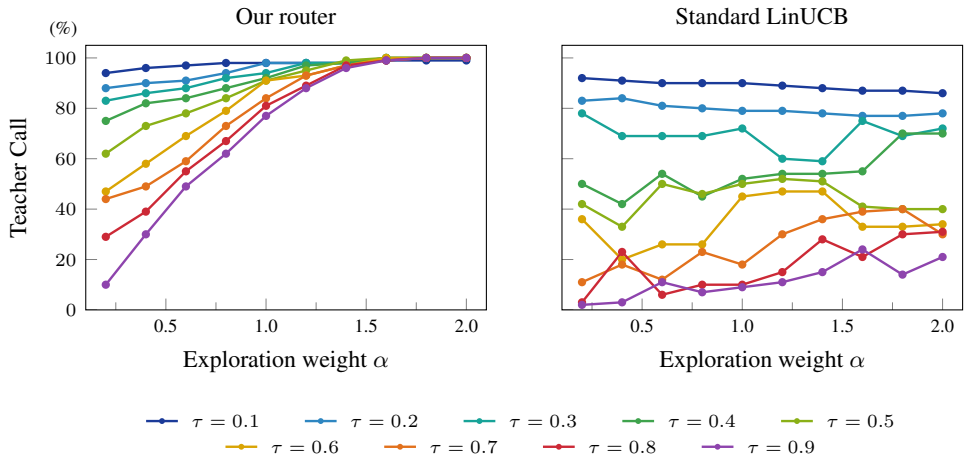


Figure 8: Teacher call rate as the exploration weight α is swept, measured over the first 100 instances of the stream. With our routing rule (left), which assigns no exploration bonus to NO-T-CALL, the call rate rises with α at every τ. With the standard LinUCB bonus for both actions (right), the curves are flat or non-monotone. Note that because τ also shifts the expected reward and thus the tendency to pick T-CALL, we compare exploration only within the same τ.

Figure 8 shows that under the standard LinUCB, the call rate is flat or non-monotone in α at each τ, so α does not control the exploration we care about. With our routing rule, the same sweep gives a call rate that rises with α at every τ and saturates once α is large enough that the router always calls the teacher. Empirically, this monotonicity makes a reasonable α easy to find by search, and improves system performance.

Results on Evaluation Data Streams Figure 9 compares our method with 'w/ Vanilla LinUCB', which retains our reward but uses the standard LinUCB exploration bonus for NO-T-CALL. The results from other methods are visualized as additional references. As the figure shows, zeroing out the exploration bonus of NO-T-CALL achieves a slightly better Pareto frontier in most of the evaluation streams, and when it does not, the difference between the methods is small.

A.2 Applicability of Regret as a Metric

Regret for a contextual bandit algorithm assumes that the environment supplies the rewards and that the environment is unaffected by the router's actions. Neither assumption holds here. Additionally, regret does not allow a comparison with the baseline routing policies, which do not use the contextual bandit formulation or optimize against the reward values.

First, our rewards are set by the performance–autonomy trade-off hyperparameter τ, which each deployment chooses for its own needs, so the regret values for systems with different τ are not comparable. Second, the reward depends on the student's evolving capacity, which the history of T-CALL and NO-T-CALL actions determines. At any timestep, two systems likely have different histories, making their rewards incomparable. For example, a system whose past actions left it with a worse student tends to receive higher rewards next, because its student is more likely to disagree with the teacher and calling the teacher to override this failure earns a higher reward than choosing NO-T-CALL for a system whose student already performs well. We derive this reward construction in Appendix B.2.

B Supplementary Derivations

B.1 Reward Scaling

Let R_+ denote the reward for calling the teacher when the student and teacher disagree, and let R_- denote the reward when they agree. Multiplying R_+ and R_- by the same $c > 0$ scales every

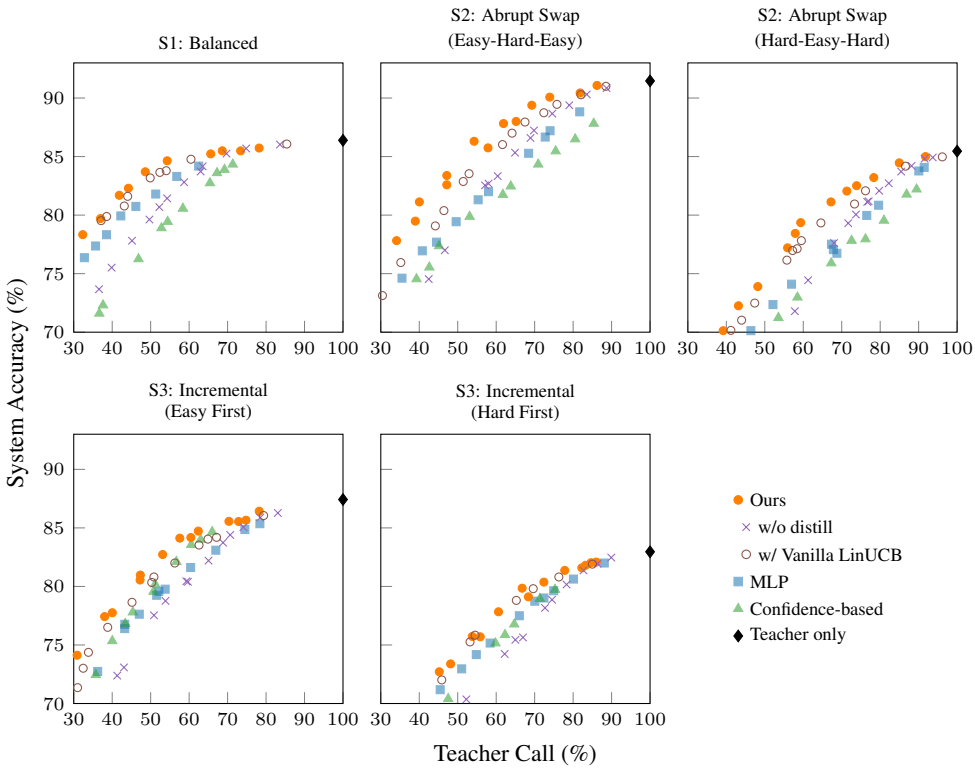


Figure 9: Ablation of the NO-T-CALL exploration bonus across all five data streams. w/ Vanilla LinUCB retains our reward but uses the standard LinUCB exploration bonus for NO-T-CALL. Higher accuracy and fewer teacher calls are better, so the preferred frontier is towards the top-left corner.

observed reward by c. This effect can be absorbed into the exploration weight by setting $\widetilde{\alpha} = c\alpha$, leaving every routing decision unchanged. Consider two runs with the same initialization and input stream. At timestep t, the first uses reward r_t and exploration weight α. The second uses $\widetilde{r}_t = cr_t$ and $\widetilde{\alpha} = c\alpha$. Suppose the two runs have made the same routing decisions before timestep t. These decisions produce the same teacher calls and student updates, so the runs observe the same context vector at timestep t and have updated A_t with the same past context vectors. Hence, $\widetilde{A}_t = A_t$. The T-CALL parameter estimate is a linear combination of all rewards previously observed for that arm: $\hat{\boldsymbol{\theta}}_t = A_t^{-1} b_t$, where $b_t = \sum_{s<t:a_s=\text{T-CALL}} r_s \mathbf{x}_s$. Therefore, $\widetilde{b}_t = cb_t$, $\widetilde{\hat{\boldsymbol{\theta}}}_t = c\hat{\boldsymbol{\theta}}_t$, and the uncertainty u_t is unchanged. It follows that

$$\begin{aligned}\widetilde{\text{UCB}}_{t,\text{T-CALL}} &= \widetilde{\hat{r}}_{t,\text{T-CALL}} + \widetilde{\alpha} u_{t,\text{T-CALL}} \\ &= c\hat{r}_{t,\text{T-CALL}} + c\alpha u_{t,\text{T-CALL}} = c\,\text{UCB}_{t,\text{T-CALL}}\,. \end{aligned} \tag{1}$$

The UCB of NO-T-CALL remains 0 in both runs. Because $c > 0$, the two runs choose the same arm at timestep t. The two runs start from the same state, so, by induction, they make the same routing decision at every timestep. These decisions produce the same teacher calls and student updates, leaving the full system trajectory unchanged. Thus, as long as α is tuned, scaling R_+ and R_- by the same factor has no effect and only the ratio $R_+/|R_-|$ matters.

B.2 REWARD FOR NO-T-CALL

Calling the teacher to override the student when the student disagrees with the teacher always earns a higher reward than choosing NO-T-CALL. This property is necessary for building a cascade system. When the router chooses NO-T-CALL, the teacher output and thus the agreement label are unobserved. The reward cannot distinguish a NO-T-CALL action when the student agrees with the teacher from one when they disagree, so we must assign every NO-T-CALL action the same constant reward R_0. If $R_+ < R_0$ and $R_- < R_0$, both possible rewards for T-CALL are lower than the reward for

NO-T-CALL, biasing the router to always choose NO-T-CALL, making the system useless. To reward useful calls without encouraging unnecessary ones, we therefore require $R_+ > R_0 > R_-$. Our construction sets $R_0 = 0$, $R_- = -1$, and $R_+ = (1-\tau)/\tau > 0$ for every $\tau \in (0, 1)$, satisfying this requirement.

B.3 ASYMPTOTIC INTERPRETATION OF τ

At convergence, we assume that the router has seen similar queries enough times that the uncertainty is low and the reward estimator recovers the true expected reward.

Let $p(x)$ denote the probability that the student and the teacher disagree on a query with context feature x, and let $\mathbb{E}[r \mid a, x]$ denote the expected reward of arm a under that context, where r is the reward defined in Algorithm 1. For T-CALL,

$$\begin{aligned}
\mathbb{E}[r \mid \text{T-CALL}, x] &= p(x)\, R_+ + \big(1 - p(x)\big)\, R_- \\
&= p(x) \cdot \frac{1-\tau}{\tau} - \big(1 - p(x)\big) \\
&= \frac{p(x)(1-\tau)}{\tau} - 1 + p(x) \\
&= \frac{p(x)(1-\tau) + p(x)\,\tau}{\tau} - 1 \\
&= \frac{p(x)}{\tau} - 1. \qquad (2)
\end{aligned}$$

Since NO-T-CALL carries a constant reward of 0, the router chooses NO-T-CALL if and only if

$$\begin{aligned}
\mathbb{E}[r \mid \text{T-CALL}, x] \le \mathbb{E}[r \mid \text{NO-T-CALL}, x] = 0 \\
\iff \frac{p(x)}{\tau} - 1 \le 0 \\
\iff p(x) \le \tau. \qquad (3)
\end{aligned}$$

Thus, whenever the router chooses NO-T-CALL, the conditional probability that the system disagrees with the teacher is $p(x)$, which is at most τ. Consequently, τ is also an upper bound on the overall probability $q(x)$ that the system disagrees with the teacher on a given query: when the router chooses T-CALL the system agrees with the teacher by construction, so $q(x)$ is no greater than $p(x)$.

The derivation above also shows that τ is similar to the risk tolerance hyperparameter in the existing LLM cascade literature (Jung et al., 2025). We note that Jung et al. (2025) obtain a provable guarantee for their risk tolerance hyperparameter by calibrating on a held-out set drawn from the evaluation distribution. Our setting is more realistic and thus does not allow for such a guarantee: queries arrive in a stream with no held-out calibration set and the distribution shifts within it, so we can only say the bound above holds asymptotically, when the router's uncertainty on a query type is low, but cannot guarantee that it will happen within the lengths of the data streams we tested. Therefore, this derivation is mainly to provide an interpretation, and we use empirical results to show our method's strengths (Section 6).

C ADDITIONAL CASCADE RESULTS

The results in this section extend the main evaluation in two ways. Figure 10 extends the comparison of Pareto frontiers to all five data streams. Tables 2, 3, and 4 report the exact agreement rates, accuracies, and teacher call rates for S1, S2, and S3, respectively.

Figure 10 and Tables 2, 3, and 4 arrive at the same conclusion as Section 6: at matched performance levels (same system agreement and accuracy), our full system consistently requires fewer teacher calls than the baselines. The magnitude of teacher call savings varies across streams. This is because the distribution-shift streams are designed mainly to stress-test our system under dynamic conditions, especially to probe the robustness of the router. We therefore allowed some data streams to have

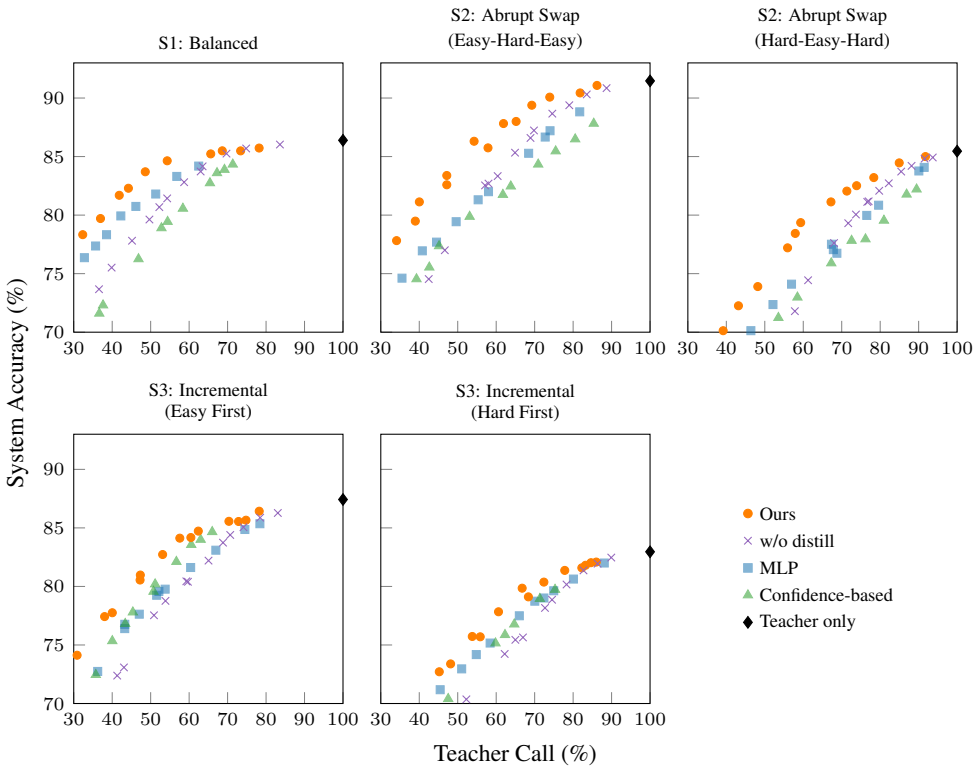


Figure 10: Performance–Autonomy trade-off across all five data streams. Higher accuracy and fewer teacher calls are better, so the preferred frontier is towards the top-left corner.

much shorter time step horizons (fewer instances) and/or skewed distributions weighted toward harder tasks. These conditions can constrain the performance improvements the student achieves within the experiment's horizon, but we still see sizable margins between our method and the baselines across all settings.

Table 2: Routing policy results on S1: Balanced at our method's three target agreement rates. Entries are observed agreement rate (Agr.), system accuracy (Acc.), and teacher call rate (TC), all in percent.

Performance level	Method	S1: Balanced		
		Agr.	Acc.	TC
	T. Only	100.0	86.4	100.0
	S. Only	47.2	46.1	0.0
Agr. $\approx 95\%$	Ours	95.7	84.6	**54.3**
	w/o distill	95.4	84.2	63.5
	MLP	95.5	84.2	62.5
	Conf.	94.9	83.6	67.3
Agr. $\approx 90\%$	Ours	91.9	81.7	**41.8**
	w/o distill	91.2	80.7	52.3
	MLP	90.9	80.7	46.2
	Conf.	89.4	79.4	54.4
Agr. $\approx 80\%$	Ours	80.9	73.7	**23.2**
	w/o distill	80.6	73.7	36.6
	MLP	80.1	72.6	25.4
	Conf.	79.3	71.6	36.6

Table 3: Routing policy results on the two S2: Abrupt Task Swap streams at our method's three target agreement rates. Entries are observed agreement rate (Agr.), system accuracy (Acc.), and teacher call rate (TC), all in percent. The easy task is GSM.

Performance level	Method	Easy–Hard–Easy			Hard–Easy–Hard		
		Agr.	Acc.	TC	Agr.	Acc.	TC
	T. Only	100.0	91.5	100.0	100.0	85.5	100.0
	S. Only	39.6	39.4	0.0	28.1	27.3	0.0
Agr. $\approx 95\%$	Ours	95.5	87.8	**61.9**	95.5	82.1	**71.3**
	w/o distill	95.0	86.6	68.9	95.3	81.2	77.0
	MLP	94.7	86.7	72.7	94.8	80.8	79.6
	Conf.	92.4	86.5	80.5	93.5	81.8	86.9
Agr. $\approx 90\%$	Ours	90.9	83.4	**47.2**	91.2	78.4	**57.9**
	w/o distill	90.1	82.5	57.1	90.6	77.6	68.0
	MLP	89.2	82.0	58.0	90.2	76.7	68.7
	Conf.	87.7	82.5	63.8	88.7	77.9	76.2
Agr. $\approx 80\%$	Ours	80.3	74.6	**28.2**	83.2	72.3	**43.2**
	w/o distill	80.1	74.5	42.5	83.3	71.8	57.8
	MLP	80.1	74.6	35.5	81.1	70.1	46.4
	Conf.	78.0	74.5	39.3	79.5	71.2	53.5

Table 4: Routing policy results on the two S3: Incremental Task Introduction streams at our method's three target agreement rates. Entries are observed agreement rate (Agr.), system accuracy (Acc.), and teacher call rate (TC), all in percent.

Performance level	Method	Easy First			Hard First		
		Agr.	**Acc.**	**TC**	**Agr.**	**Acc.**	**TC**
	T. Only	100.0	87.4	100.0	100.0	83.0	100.0
	S. Only	41.7	41.1	0.0	35.2	34.8	0.0
Agr. $\approx 95\%$	Ours	95.6	84.2	**60.5**	95.5	79.8	**66.8**
	w/o distill	95.4	83.7	68.8	95.2	78.9	74.5
	MLP	94.5	83.1	66.9	94.8	79.0	72.4
	Conf.	93.9	84.0	63.0	93.1	78.9	71.4
Agr. $\approx 90\%$	Ours	91.4	80.5	**47.2**	90.7	75.7	**53.8**
	w/o distill	91.0	80.4	59.3	90.5	75.4	65.0
	MLP	90.6	79.8	53.8	89.4	75.2	58.4
	Conf.	88.8	79.6	50.7	87.7	75.2	59.8
Agr. $\approx 80\%$	Ours	81.3	74.1	**30.9**	80.6	69.6	**37.1**
	w/o distill	80.6	73.1	43.1	78.5	68.5	48.5
	MLP	80.6	72.7	36.2	80.1	68.0	41.5
	Conf.	80.0	72.5	35.7	79.2	68.4	44.3

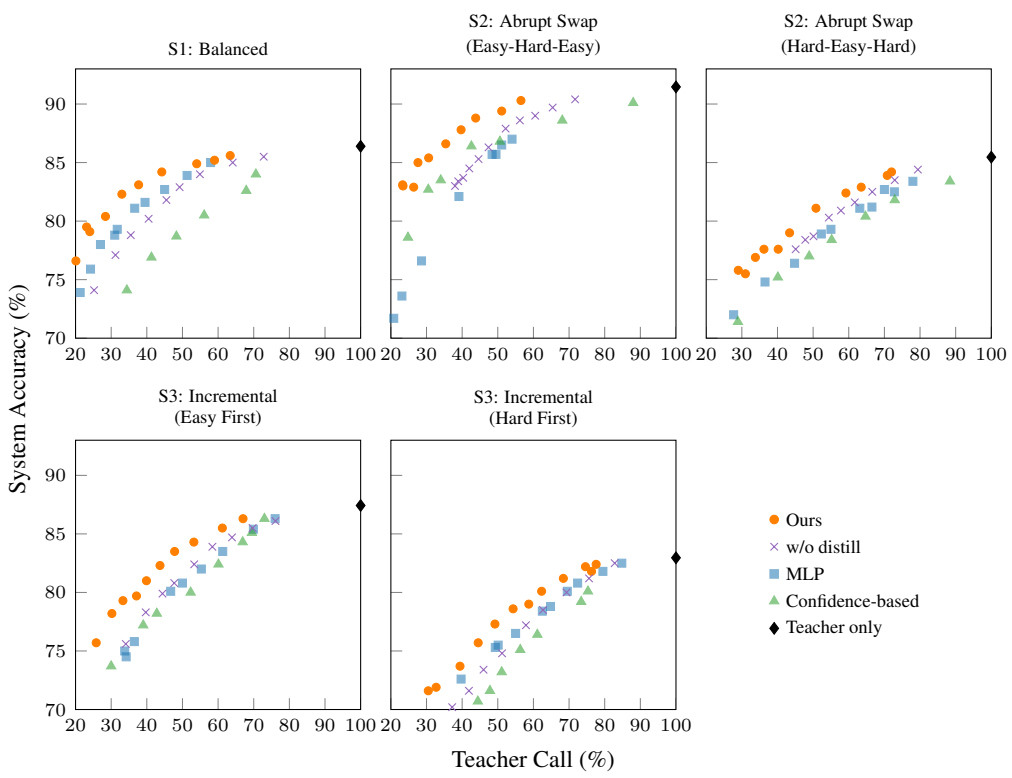


Figure 11: Performance–Autonomy trade-off with Qwen-3-1.7B as the student. Higher accuracy and fewer teacher calls are better.

C.1 Thinking Model Results

Figure 11 shows the results with Qwen-3-1.7B as the student. Because thinking-model inference requires more computational resources and time, we ran fewer runs per method while still covering a range of performance levels. Across the five streams, we observe the same trend as with the Qwen2.5 model. Our method requires substantially fewer teacher calls than every baseline at the same or better accuracy, yielding the dominant Pareto frontier for the performance–autonomy trade-off.

C.2 Effects of Student Learning on the Cascade

The figures below extend the analysis in Section 6 by comparing our system's behavior with the w/o distill baseline across all five data streams (using the runs with system agreement rate at 90% as examples). Across these streams, the teacher call rate decreases as the student's agreement with the teacher increases. This learning allows our system to reach the same overall performance as w/o distill with fewer teacher calls.

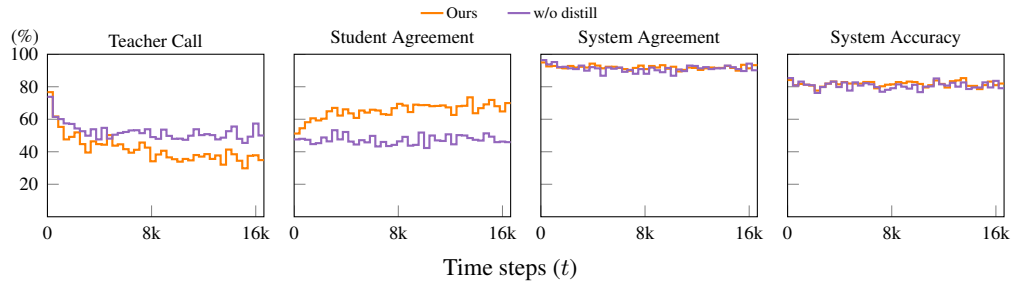


Figure 12: S1: Balanced. Reported are average statistics computed over consecutive windows of 500 instances. Both runs achieve the same system agreement ($\approx 90\%$).

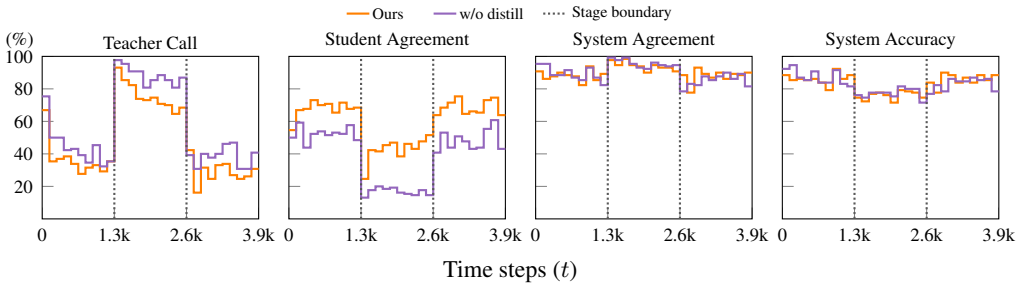


Figure 13: S2: Easy–Hard–Easy (GSM). Reported are average statistics computed over consecutive windows of 130 instances (one-tenth of a stage). Both runs achieve the same system agreement ($\approx 90\%$).

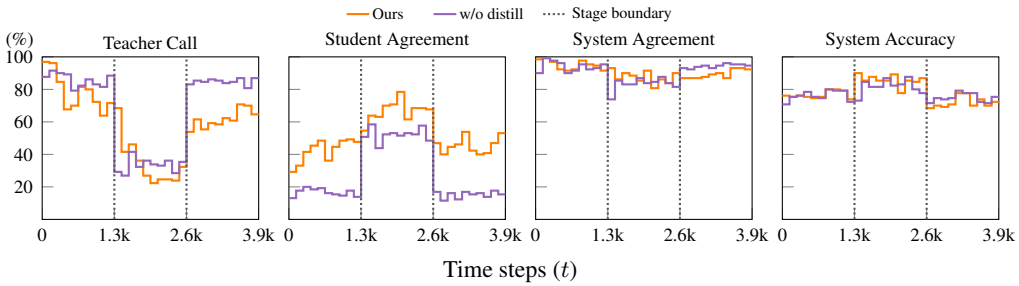


Figure 14: S2: Hard–Easy–Hard (GSM). Reported are average statistics computed over consecutive windows of 130 instances (one-tenth of a stage). Both runs achieve the same system agreement ($\approx 90\%$).

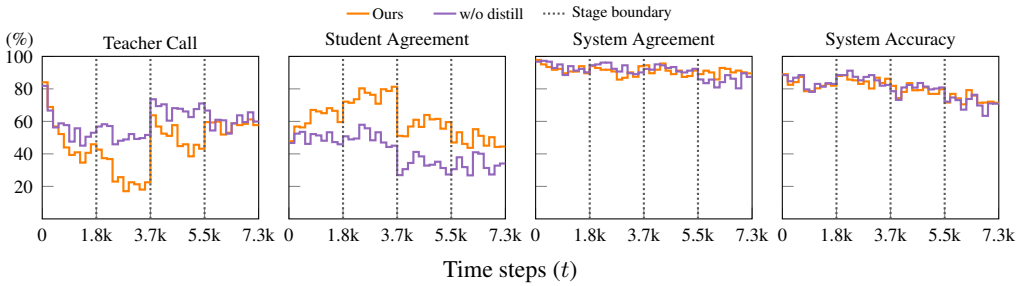


Figure 15: S3: Incremental, Easy First. Reported are average statistics computed over consecutive windows of 183 instances (one-tenth of a stage). Both runs achieve the same system agreement ($\approx 90\%$).

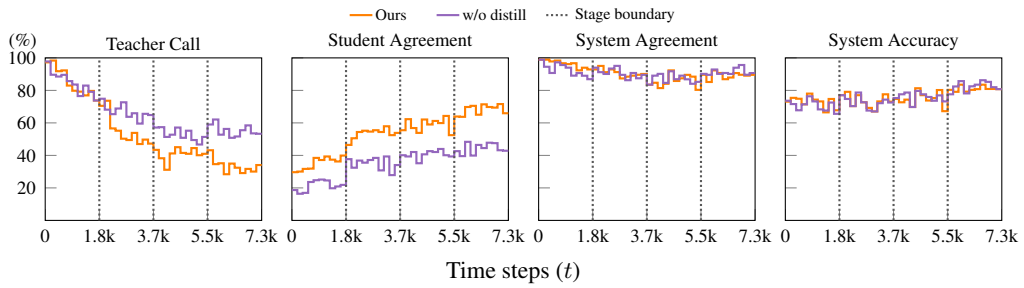


Figure 16: S3: Incremental, Hard First. Reported are average statistics computed over consecutive windows of 183 instances (one-tenth of a stage). Both runs achieve the same system agreement ($\approx 90\%$).

Student Performance on Held-out Eval Set Previously, Figure 5 in Section 6 studied the student's forgetting by presenting online evaluation statistics on our Easy–Hard–Easy stream. That evaluation reflects the online performance in a deployment scenario, where both the router and the student are constantly updated.

An alternative evaluation is to save checkpoints of the student at different time steps along the same data stream and evaluate each checkpoint on a held-out eval set for every task (disjoint from the data stream). This is less reflective of the deployment scenario. In a real online deployment, solving one query influences the model for the next query. In this alternative evaluation, each student checkpoint is static, so every query is treated independently. A checkpoint may perform worse on a task that has disappeared from the stream, but the student may quickly recover once the task reappears. The speed of this recovery affects the consequence of forgetting and the overall performance in an online deployment, which is only captured by our main statistics in Section 6. Thus, we do not rely on this held-out evaluation in our main analysis.

Figure 17 presents the held-out evaluation for all five data streams. Before the system sees a task, the student's performance on that task mostly stays the same or improves. This improvement is likely a result of the relatedness of the tasks. For example, learning to solve one classification task can improve the model's performance on another, and transferable abilities likely exist between reasoning tasks. Across the streams, we see substantial learning in 4 of the 6 tasks. MiniCheck is a control task where the student already achieves the same performance as the teacher, so no learning is expected. The Chemistry task also shows a limited amount of learning, potentially due to its requirement for factual knowledge and the small dataset size.[6] Overall, we expect the amount of learning the student can achieve on a task to depend on the nature of the task (e.g., difficulty and knowledge requirements) along with the student's architecture, pretraining, and post-training before being used for the cascade. This further contributes to the router's importance in a cascade, where it should continually adapt while identifying what's still hard for the student.

Once the student improves on a task in the online stream, its performance on that task tends to persist. It shows small fluctuations when it sees only another task for an extended period (Stage 2 of S2: Easy–Hard–Easy), but the decrease in performance is small and not catastrophic. The only performance degradation on the held-out eval set occurs at the beginning of S2: Hard–Easy–Hard, but it does not hurt the overall system performance. The student initially sees and learns only Tablebench, which causes a brief performance drop on GSM. However, no GSM queries are served in that stage, and its GSM performance recovers before those queries arrive. The student then improves quickly on GSM in Stage 2. Even if its performance had not recovered, the router's exploration on an unfamiliar task would use enough teacher calls when seeing GSM initially, quickly identify the gap, and improve the student while maintaining system performance.

[6]Distillation with a couple of thousand instances is more suitable for the small model to learn general skills than gaining comprehensive factual knowledge for this type of task.

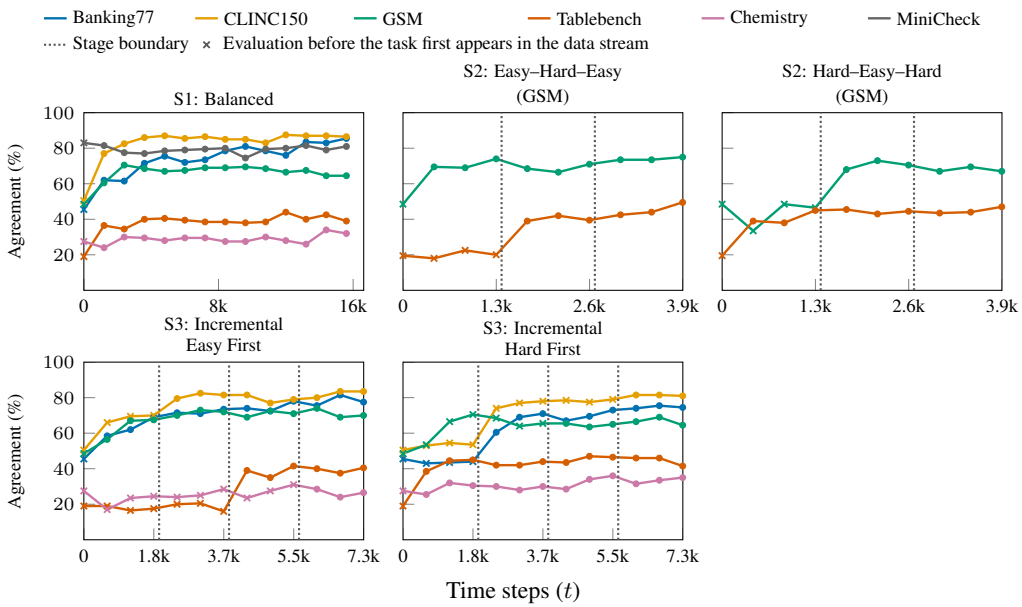


Figure 17: Held-out student–teacher agreement for our method across the five main streams. Data points marked with × mean that the student is being evaluated before the corresponding task first appears in the data stream. Appendix D.2 provides details on when each task first appears in a data stream. For example, the × on the Tablebench curve in Stage 1 of S2: Easy–Hard–Easy means that the system has not yet seen Tablebench instances in that stage.

C.3 Comparison with Routing Baselines

The following figures compare our router with MLP and the confidence router throughout the five streams. We report the system agreement rate, the student's agreement rate with the teacher, the system accuracy, the teacher call rate, and the router's true positive and false positive rates. Positives are instances where the student and teacher disagree, so a true positive is a useful teacher call and a false positive is an unnecessary teacher call. For each stream, all three runs achieve the same system agreement ($\approx 90\%$).

Across these streams, our router adapts more than the confidence router and more smoothly than MLP. In S1 and S2: Easy–Hard–Easy, the confidence router reduces its teacher call rate less than our method. In contrast, MLP's teacher call rate fluctuates, especially around distribution shifts. These fluctuations cause it to miss useful teacher calls (low true positive rate) and to make unnecessary calls (high false positive rate). Our router avoids these extremes while adjusting its teacher call rate as the student improves and the query distribution changes.

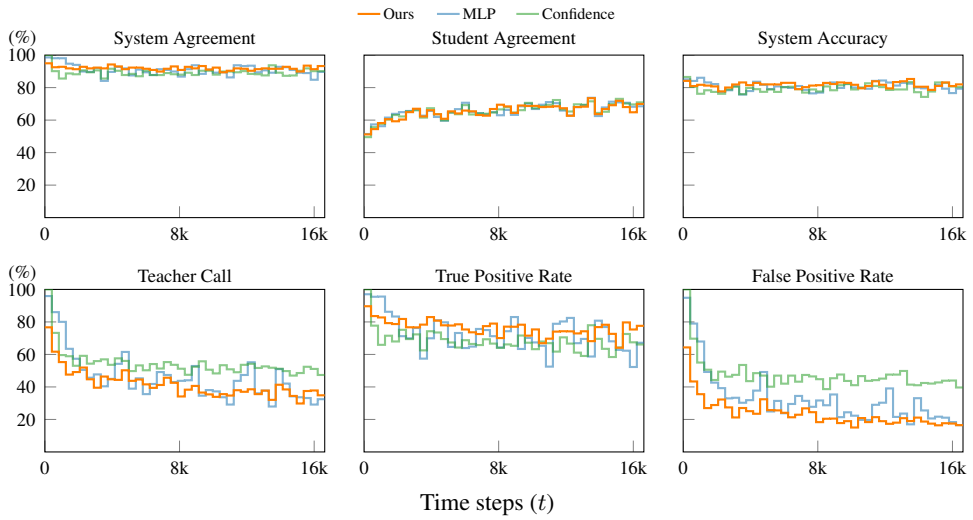


Figure 18: S1: Balanced. Reported are average statistics computed over consecutive windows of 500 instances.

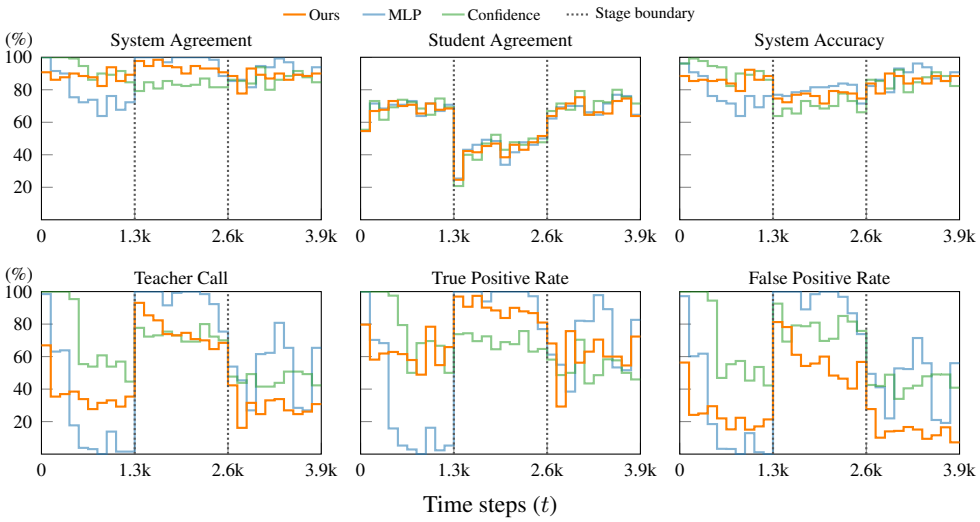


Figure 19: S2: Easy–Hard–Easy (GSM). Reported are average statistics computed over consecutive windows of 130 instances (one-tenth of a stage).

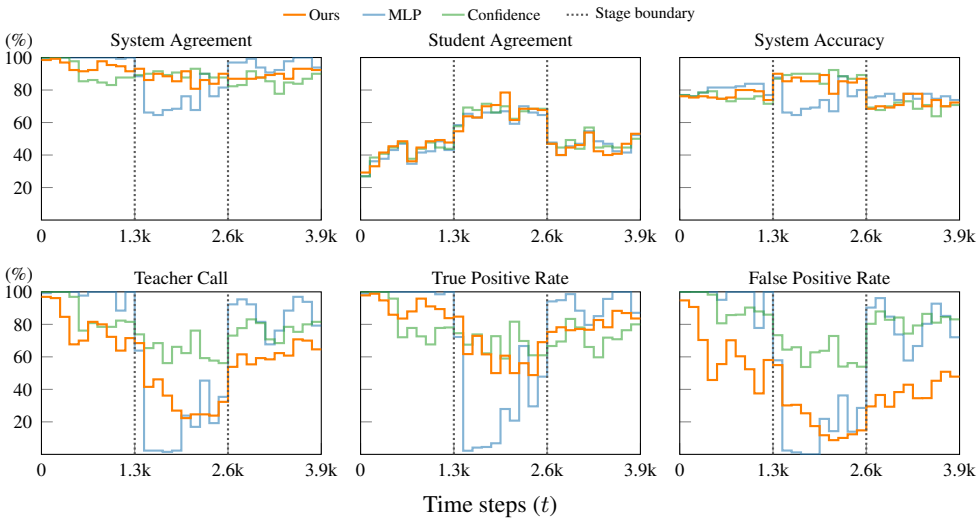


Figure 20: S2: Hard–Easy–Hard (GSM). Reported are average statistics computed over consecutive windows of 130 instances (one-tenth of a stage).

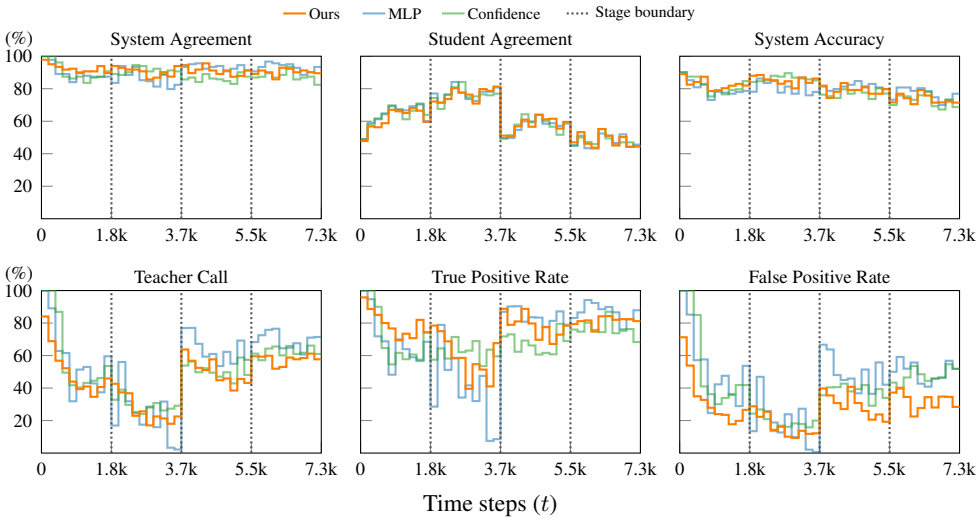


Figure 21: S3: Incremental, Easy First. Reported are average statistics computed over consecutive windows of 183 instances (one-tenth of a stage).

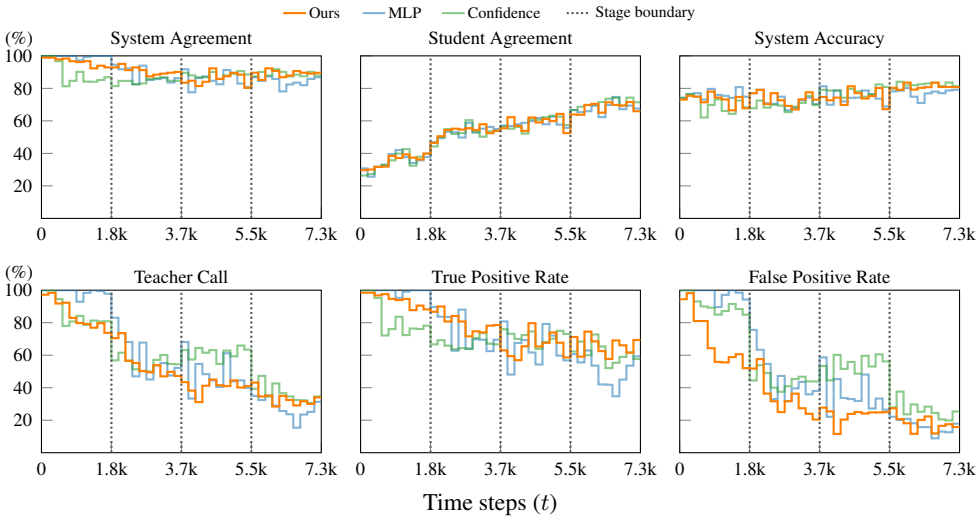


Figure 22: S3: Incremental, Hard First. Reported are average statistics computed over consecutive windows of 183 instances (one-tenth of a stage).

C.4 BANKING77 STREAMS

The two S2 variants with Banking77 are not part of our five-stream evaluation suite. They use the same abrupt swap construction as the S2 streams in the suite, replacing GSM with Banking77 as the easy task. We include them as supplementary results to test whether the conclusions depend on the choice of easy task. They lead to the same conclusion: our method reaches the same performance as the baselines with fewer teacher calls. Figure 23 compares their Pareto frontiers. The figures that follow show the methods' behavior over time at approximately 90% agreement.

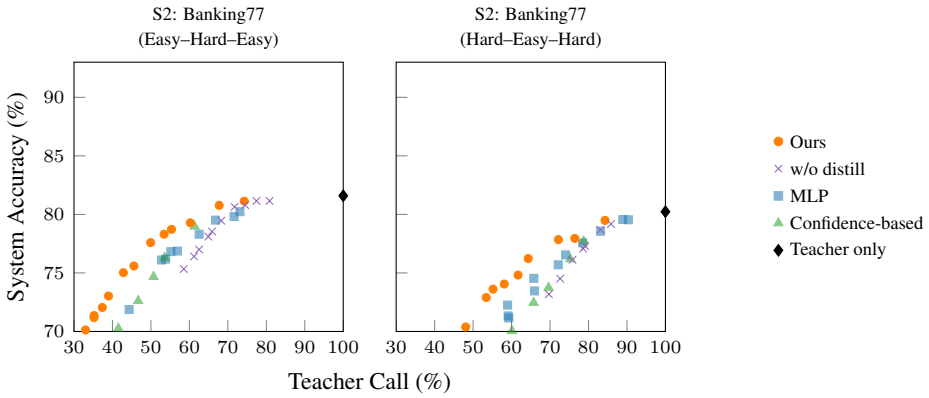


Figure 23: Performance–Autonomy trade-off on the two supplementary S2 streams with Banking77 as the easy task. Higher accuracy and fewer teacher calls are better, so the preferred frontier is towards the top-left corner.

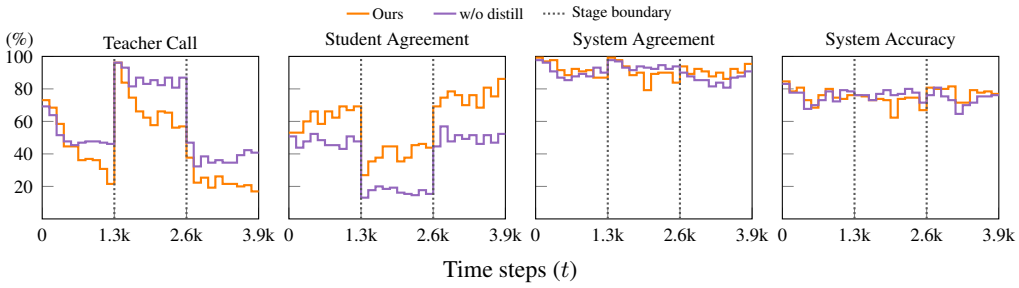


Figure 24: Ours and w/o distill on S2: Easy–Hard–Easy (Banking77), using the runs selected for the 90% system-agreement target. Each panel reports a statistic computed within consecutive, disjoint intervals. The agreement rates compare the system output with the teacher's; system accuracy is the fraction of system outputs that are correct. Dotted lines mark distribution shifts when present.

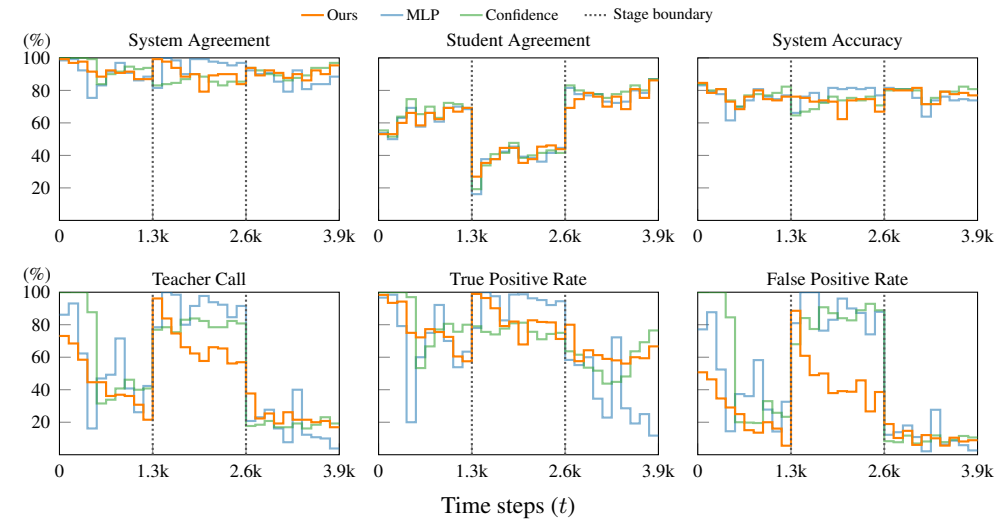


Figure 25: Ours, MLP, and the confidence router on S2: Easy–Hard–Easy (Banking77), using the runs selected for the 90% system-agreement target. Each panel reports a statistic computed within consecutive, disjoint intervals. The agreement rates compare the system output with the teacher's; system accuracy is the fraction of system outputs that are correct. A positive routing decision calls the teacher, and a call is needed when the student and teacher disagree. Dotted lines mark distribution shifts when present.

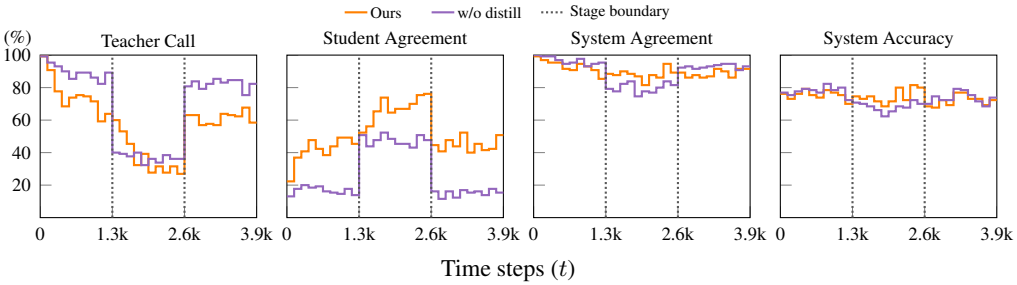


Figure 26: Ours and w/o distill on S2: Hard–Easy–Hard (Banking77), using the runs selected for the 90% system-agreement target. Each panel reports a statistic computed within consecutive, disjoint intervals. The agreement rates compare the system output with the teacher's; system accuracy is the fraction of system outputs that are correct. Dotted lines mark distribution shifts when present.

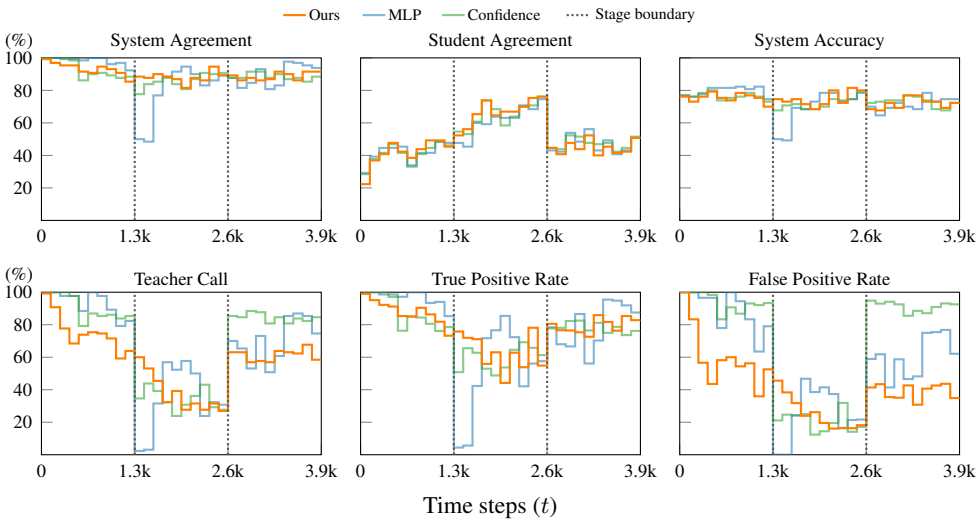


Figure 27: Ours, MLP, and the confidence router on S2: Hard–Easy–Hard (Banking77), using the runs selected for the 90% system-agreement target. Each panel reports a statistic computed within consecutive, disjoint intervals. The agreement rates compare the system output with the teacher's; system accuracy is the fraction of system outputs that are correct. A positive routing decision calls the teacher, and a call is needed when the student and teacher disagree. Dotted lines mark distribution shifts when present.

D Implementation Details

D.1 Cascade Implementation

The router decides based on a context vector whether to invoke the teacher model after the student model gives its answer. We set the context vector $\mathbf{x}$ with the student's hidden state at its last generated token, analogous to prior work on LLMs for text embeddings (Wang et al., 2024).

D.2 Query Types and Data Stream Details

- **Banking77** is customer-query classification in the banking domain with 77 fine-grained intent labels (Casanueva et al., 2020). The off-the-shelf student performs much worse than the teacher, but the gap closes quickly when learning from the teacher.
- **CLINC150** is intent classification for customer-service queries across 150 intents (Larson et al., 2019). As with Banking77, the off-the-shelf student is weak but adapts quickly.
- **GSM** contains math questions from GSM-symbolic, an augmented version of GSM8K (Mirzadeh et al., 2025). It contains groups of questions that require the same skill to solve, exhibiting the similarity phenomenon (Section 2) that resemble the application scenarios where an adaptive small model is most useful. It is hard for the off-the-shelf student but the student adapts relatively quickly when learning from the teacher.
- **MiniCheck** is document-grounded fact-checking (Tang et al., 2024). The off-the-shelf student already matches the teacher, so we include this as a control task: an efficient routing policy should quickly identify these queries as the ones the small model already does well on and stop calling the teacher.
- **Tablebench** is table understanding and analysis (Wu et al., 2025). The student improves slowly, and its final performance remains significantly below the teacher within the time horizon we experiment with ($<$3000 instances).
- **Chemistry** contains chemistry reasoning questions from the Chemistry L-3 subset of SciKnowEval (Feng et al., 2024), similar to Tablebench in being hard to learn to match the teacher's performance.

Table 5: Task composition of S1: Balanced. The stream randomly shuffles all included instances. We use 3,000 instances per task unless fewer are available.

Task	Total
Banking77	3,000
CLINC150	3,000
GSM	3,000
MiniCheck	3,000
Tablebench	2,727
Chemistry	1,900
Total	**16,627**

Table 6: Task composition of the two S2: Abrupt Task Swap streams in our evaluation suite and the two additional Banking77 variants. Each stream has three stages of 1,300 instances. Tablebench is the hard task, and Banking77 or GSM is the easy task.

(a) Easy–Hard–Easy (Banking77)

Task	Stage 1	Stage 2	Stage 3	Total
Banking77	1,300	0	1,300	2,600
Tablebench	0	1,300	0	1,300
Total	**1,300**	**1,300**	**1,300**	**3,900**

(b) Hard–Easy–Hard (Banking77)

Task	Stage 1	Stage 2	Stage 3	Total
Banking77	0	1,300	0	1,300
Tablebench	1,300	0	1,300	2,600
Total	**1,300**	**1,300**	**1,300**	**3,900**

(c) Easy–Hard–Easy (GSM)

Task	Stage 1	Stage 2	Stage 3	Total
GSM	1,300	0	1,300	2,600
Tablebench	0	1,300	0	1,300
Total	**1,300**	**1,300**	**1,300**	**3,900**

(d) Hard–Easy–Hard (GSM)

Task	Stage 1	Stage 2	Stage 3	Total
GSM	0	1,300	0	1,300
Tablebench	1,300	0	1,300	2,600
Total	**1,300**	**1,300**	**1,300**	**3,900**

Tables 5–7 give the exact task composition of the five data streams and the two additional Banking77 variants. S1 uses up to 3,000 instances from each task and includes every available instance when a dataset contains fewer than 3,000.

Table 7: Stage composition of the two S3: Incremental Task Introduction streams. Each stream has four stages of 1,825 instances, for 7,300 instances in total.

(a) Easy-first

Task	Stage 1	Stage 2	Stage 3	Stage 4	Total
Banking77	913	457	305	229	1,904
GSM	912	456	304	228	1,900
CLINC150	0	912	304	228	1,444
Tablebench	0	0	912	228	1,140
Chemistry	0	0	0	912	912
Total	**1,825**	**1,825**	**1,825**	**1,825**	**7,300**

(b) Hard-first

Task	Stage 1	Stage 2	Stage 3	Stage 4	Total
Tablebench	913	457	305	229	1,904
Chemistry	912	456	304	228	1,900
Banking77	0	912	304	228	1,444
GSM	0	0	912	228	1,140
CLINC150	0	0	0	912	912
Total	**1,825**	**1,825**	**1,825**	**1,825**	**7,300**

D.3 Experiment Configurations

Training We update the student using LoRA, with rank $r = 16$ and $\alpha_{\text{LoRA}} = 32$ for Qwen2.5, and $r = 64$ and $\alpha_{\text{LoRA}} = 128$ for the Qwen3 thinking model. We use a batch size of 16 for distillation updates. We search over learning rates $\{1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ and use 5×10^{-5} for Qwen2.5 and 1×10^{-4} for Qwen3. For the exploration weight α (Section 3), we search over $\{0.2, 0.5, 1.0, 1.5, 2.0\}$ and use 0.5. For the ridge regression regularization λ (Section 3), we search over $\{0.01, 0.1, 1.0\}$ and use 0.01. We fix these values for all experiments and vary the trade-off parameter τ to obtain runs at different performance levels. We conduct these hyperparameter searches on TREC (Li & Roth, 2002; Hovy et al., 2001), a classification task that we use for development and do not include in our evaluation streams.

For the confidence-based router, we followed Jung et al. (2025) and varied its risk tolerance α to obtain runs at different trade-off levels. We use a cold start stage of 500 instances, meaning that for the first 500 instances, the router always call the teacher to collect data for confidence calibration.

For the MLP router, we used a two-layer MLP with hidden layers of size 256 and 128, and followed Nie et al. (2024) for implementation. We searched the MLP learning rate over $\{1 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$ and used 1×10^{-5}. Nie et al. (2024)'s router learning also includes a decay factor, which we searched over $\{0.9, 0.95, 0.99\}$ and used 0.9.

For all experiments involving student learning (ours and the baselines alike), we used instance replay with reservoir sampling, a standard choice in online continual learning to reduce the model's forgetting of learned capabilities (Riemer et al., 2019; Chaudhry et al., 2019). Before we have 1000 teacher examples, a minibatch consists only of new instances. After we have accumulated 1000 teacher examples, each minibatch update uses 8 new instances and 8 drawn from the past instances with reservoir sampling.

Inference We use a max output token limit of 2048 tokens for Qwen2.5 and a limit of 10240 tokens for Qwen3 (thinking budget 8192 tokens). Qwen2.5 uses greedy decoding and Qwen3 uses sampling with its default temperature, as recommended by its official documentation.

Obtaining Results at Various Performance Levels To study the performance–autonomy trade-off across various performance levels, we first vary the hyperparameter that controls the trade-off for

each method (e.g., τ for our method), trying to cover a wide region of performance levels. These initial runs reflect the Pareto frontier of the performance–autonomy trade-off for each method. For more fine-grained comparison at a specific agreement level, such as agreement rate $\approx$95%, we need runs that land near that level. If a method's initial runs do not include a run close enough to the agreement rate for head-to-head comparison, we try additional values for its trade-off hyperparameter to find one that's near the target agreement rate.