READ: REINFORCEMENT-GUIDED CAPABILITY DISTILLATION FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Capability distillation specializes a smaller language model toward a specified capability using supervision from a larger teacher. Under a fixed distillation-token budget, an important question is how to allocate supervision across capabilities to improve the target most effectively. Our empirical study shows that single-capability distillation produces budget-dependent cross-capability changes and diminishing target gains, motivating allocation strategies that adapt to the student's evolving state. We propose *ReAD*, a *Re*inforcement-guided c*A*pability *D*istillation framework that combines a task-conditioned allocation prior, on-the-fly teacher supervision, and an uncertainty-aware contextual bandit. ReAD updates its allocation using measured capability changes, rewarding task-aligned gains and penalizing regressions on prioritized capabilities. Across eight target capabilities, two teacher–student pairs, and two token budgets, ReAD achieves higher mean on-target scores than the evaluated target-only baselines. Matched-generator comparisons show additional gains over static and greedy allocation schedules, while transfer diagnostics show smaller off-target declines. Our code is available at `https://anonymous.4open.science/r/ReAD-62B7`.

1 INTRODUCTION

Knowledge distillation (KD) (Hinton et al., 2015) for large language models (LLMs) (Zhao et al., 2026) has become an essential research direction for enhancing the accessibility and efficiency of Machine-Learning-as-a-Service (MLaaS) (Cai et al., 2025; Xu et al., 2024b; Yang et al., 2025). A small LLM learns to imitate the outputs of a large teacher and can then be deployed with limited computational or financial resources (Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020; Li et al., 2021; Liang et al., 2023). Related work lowers serving cost in other ways, for example by routing each query to the model whose capabilities match it (Cheng & Dong, 2026) or by handing a multi-turn conversation to a small surrogate model distilled from the large one (Cheng et al., 2026). Recent studies show that capability distillation, a specialization of knowledge distillation that focuses supervision on a target capability (e.g., instruction-following, reasoning, mathematics, or coding), can substantially improve the student's performance on downstream tasks that primarily depend on that capability, while achieving these gains at lower serving cost than the large teacher model (Magister et al., 2023; Shridhar et al., 2023; Xu et al., 2024a; Yue et al., 2024).

For a specified target capability, a natural distillation strategy is to devote the available supervision budget entirely to that capability (Taori et al., 2023; Chiang et al., 2023; Zhang et al., 2024). However, distillation updates also affect other capabilities (Zhong et al., 2024; Fang et al., 2026; Cloud et al., 2026), and the gains from additional target-specific supervision can diminish as training progresses. These observations raise a budget-allocation question. *Can adapting the mixture of capability-targeted supervision improve the target capability more effectively than devoting the entire budget to its own data?* To study it, we run a controlled experiment. We repeatedly distill the student under several budgets, each time spending the whole budget on one of eight core capabilities (General Knowledge, Reasoning, Math, Code, Tool Use, Long-Context Understanding or LCU, Steerability, and Multilinguality), and evaluate it on all eight. The result is a budget-dependent capability transfer matrix. Its diagonal holds on-target gains, and its off-diagonal entries show how performance moves across capabilities. Across budgets, we observe two consistent patterns: *(i) capability-specific distillation induces systematic, budget-dependent transfer to other capabilities rather than isolated improvements*, and *(ii) increasing the budget for a single target produces smaller additional target*

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.

gains while making the average harm to non-target capabilities more pronounced. These patterns suggest that the value of an extra token may depend on where earlier tokens went. They leave open whether mixing supervision helps the target, and our experiments examine that question.

Building on these observed insights, we propose ***ReAD***, a ***Re***inforcement-guided c***A***pability ***D***istillation framework for large language models (Figure 2). ReAD sets task-conditioned capability priorities, spends the budget in short steps, and lets an uncertainty-aware contextual bandit choose the mix of capability-targeted supervision at each step from the measured change in the student. Online mixing has been explored for pre-training and skill learning (Chen et al., 2023; Albalak et al., 2023; Chen et al., 2025); ReAD brings it to teacher-guided distillation for one target.

The main contributions of this paper are as follows.

- **Novel Problem Formulation:** Motivated by our observations of budget-dependent cross-capability transfer and diminishing returns, we formulate a new problem of sequentially allocating a limited distillation budget across interdependent capabilities to maximize performance on a specified target.
- **Unified Distillation Framework:** We propose ReAD, a unified framework for target-oriented capability distillation that integrates task-conditioned capability priorities with uncertainty-aware adaptive allocation. ReAD adjusts the mixture of teacher supervision according to the student's evolving capability profile, using feedback that rewards task-aligned gains and penalizes regressions on the task's prioritized capabilities.
- **Comprehensive Experimental Evaluation:** We conduct extensive experiments across target capabilities, teacher–student pairs, and token budgets. Results show improved target performance over the evaluated baselines under matched distillation budgets, together with reduced off-target degradation in transfer diagnostics. Controlled comparisons and ablation studies further establish the benefits of adaptive allocation and the framework's individual components.

2 PRELIMINARY

We formalize budgeted capability distillation (Section 2.1), measure cross-capability transfer under single-target distillation (Section 2.2), and state the allocation problem (Section 2.3).

2.1 CAPABILITY DISTILLATION UNDER A TOKEN BUDGET

In this paper, we study budgeted capability distillation from a large teacher model T to a smaller student model S, with total training-token budget B. Let $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ denote a set of measurable capabilities, here the eight core capabilities of Appendix Table 3, and let $s_k(M)$ be model M's benchmark score on capability c_k. A distillation strategy π specifies how data is generated, how the teacher is queried, and how tokens are allocated across capabilities, assigning budgets $\{b_k\}$ with $\sum_k b_k \leq B$, and producing a distilled student S_π.

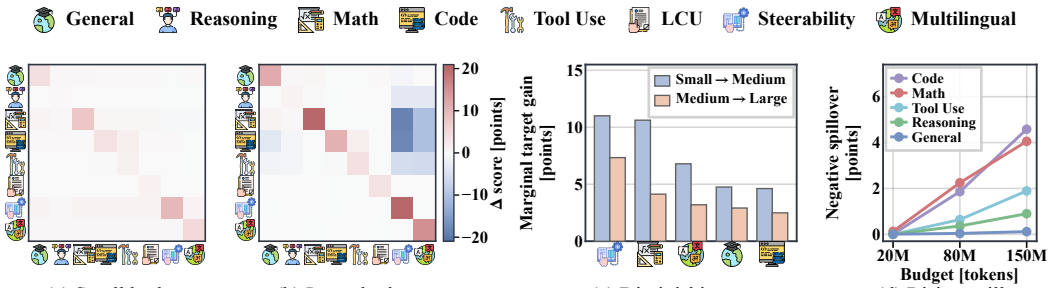
Definition 2.1 (Capability distillation)**.** Given a teacher model T, an initial student model S_0, and capability-specific data distributions $\{\mathcal{D}_c\}_{c\in\mathcal{C}}$, capability distillation allocates a limited budget B across capabilities to improve task-relevant performance under shared representations by optimizing a weighted mixture of distillation objectives:

$$\min_{S,\,\mathbf{w}} \mathbb{E}_{x\sim\mathcal{D}_\mathbf{w}}[\ell(S(x), T(x))] \quad \text{s.t.} \quad \mathbf{w} \in \Delta_{|\mathcal{C}|},\ \mathrm{cost}(S; S_0) \leq B. \tag{1}$$

where $\mathcal{D}_\mathbf{w} := \sum_{c\in\mathcal{C}} w_c \mathcal{D}_c$, and $\mathrm{cost}(S; S_0)$ measures the distillation token budget consumed to obtain S from S_0. Here $\mathbf{w} = (w_c)_{c\in\mathcal{C}}$ lies in the probability simplex $\Delta_{|\mathcal{C}|} = \{\mathbf{w} \in \mathbb{R}^{|\mathcal{C}|} \mid w_c \geq 0,\ \sum_{c\in\mathcal{C}} w_c = 1\}$, which specifies the allocation of training effort across capabilities, and ℓ denotes a standard distillation loss, such as token-level cross-entropy or logit-matching between the student's and the teacher's output distributions.

Because capabilities share representations, budget spent on one capability can also change the others. We measure this effect with one-hot allocations.

Definition 2.2 (Capability interdependence)**.** Let S_0 be an initial student model, and let $S(\mathbf{w}, B)$ denote the student obtained after capability distillation under allocation $\mathbf{w} \in \Delta_{|\mathcal{C}|}$ and budget B. For each capability c_i, define the one-hot allocation vector $\mathbf{w}^{(i)} \in \Delta_{|\mathcal{C}|}$ by $w_i^{(i)} = 1$ and $w_j^{(i)} = 0$ for all



(a) Small budget (b) Large budget (c) Diminishing returns (d) Rising spillover

Figure 1: Exploratory study. (a–b) Cross-capability transfer under single-capability distillation. Larger budgets sharpen target gains and tend to expose stronger negative transfer. (c–d) Budget waste. Extra tokens yield smaller target gains and more spillover to non-target capabilities.

$j \neq i$. The *capability interdependence* under capability distillation is characterized by the capability transfer matrix $\mathbf{T}(B) \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$ with entries

$$\mathbf{T}_{ij}(B) := s_j\Big(S(\mathbf{w}^{(i)}, B)\Big) - s_j(S_0),$$

where $\mathbf{T}_{ij}(B)$ quantifies the effect of distilling toward capability c_i on the performance of capability c_j. Non-zero off-diagonal entries indicate interdependence between capabilities.

2.2 Cross-Capability Transfer in Single-Target Distillation

We distill Llama-3.3-70B-Instruct into Llama-3.1-8B-Instruct under fixed token budgets, one target capability at a time, and evaluate the student on all benchmarks to form the transfer matrix of Definition 2.2 (strategies and settings in Appendix B). All benchmark metrics are accuracies or success rates on a 0–100 scale. A capability score is the unweighted mean over its benchmarks, and each entry of the transfer matrix is an absolute difference in these points, without further rescaling.

Observation 1: Distilling a specific capability redistributes performance across other capabilities in a budget-dependent manner. Figure 1(a–b) shows the transfer matrices at a small and a large budget. The off-diagonal mass stays non-zero in both, so optimizing one capability also moves the others. The pattern shifts with the budget. At small budgets the changes are weak and diffuse, while at large budgets the diagonal sharpens and negative entries appear for specific non-targets.

Observation 2: Capability distillation exhibits substantial budget waste due to diminishing returns and negative spillover. For the targets that improve most, the marginal gain from 20M→80M tokens is larger than that from 80M→150M (*diminishing returns*, Figure 1c). Over the same range, the drop on non-target capabilities grows with the target's budget (*negative spillover*, Figure 1d). Past a capability-specific knee point, extra tokens seem to buy little target benefit and add cross-capability degradation. Each effect is familiar on its own. Off-target drops resemble catastrophic forgetting (Kirkpatrick et al., 2017; Luo et al., 2025), and saturation is ordinary diminishing returns. An allocator can use their *structure*, since transfer is signed, source-specific, and budget-dependent.

2.3 Target-Oriented Budget Allocation

We formalize the allocation of a fixed budget as a sequential problem. In our experiments, the task τ is a specified target capability and U_τ is its on-target score.

Problem 2.3 (Budget allocation for target-capability distillation)**.** *Given a teacher model T, an initial student model S_0, a downstream task τ, and a total distillation budget B, consider a sequential distillation process over steps $t = 1, \ldots, T$ with cumulative cost not exceeding B. At each step t, a capability allocation vector $\mathbf{w}_t \in \Delta_{|\mathcal{C}|}$ is selected and applied to update the student, yielding S_t. Let $U_\tau(\cdot)$ be the task-dependent utility function; our target is to determine an allocation policy $\{\mathbf{w}_t\}_{t=1}^T$ that maximizes the performance of the final student model:*

$$\max_{\{\mathbf{w}_t\}_{t=1}^T} U_\tau(S_T)\,,\ \textit{s.t.}\ \sum_{t=1}^{T} \mathrm{cost}(S_{t-1} \to S_t) \leq B \tag{2}$$

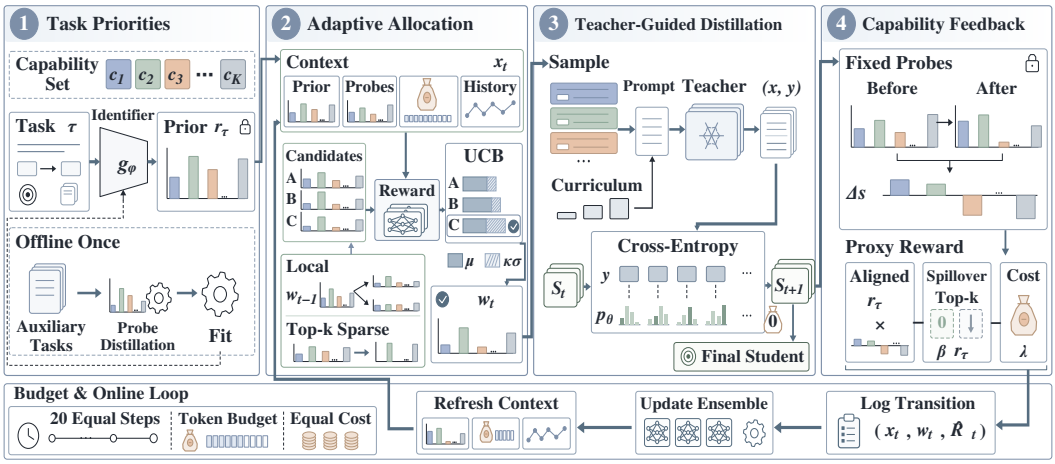


Figure 2: Overview of ReAD. **(1)** A pretrained identifier sets the task's capability priorities $\mathbf{r}_\tau$. ReAD then repeats three steps until the budget B is spent. **(2)** A bandit selects the allocation $\mathbf{w}_t$ by UCB. **(3)** The student is distilled on teacher outputs for prompts sampled from $\mathbf{w}_t$. **(4)** The measured capability change gives a proxy reward, which updates the bandit.

3 Methodology

ReAD treats Problem 2.3 as state-dependent budget allocation over capabilities (Figure 2). It first sets task-conditioned capability priorities. Then, at each step, it generates capability-targeted teacher supervision and uses an uncertainty-aware contextual bandit to choose the next token allocation. As the student profile changes, ReAD can shift budget away from saturated or high-spillover capabilities. ReAD is *off-policy* (Agarwal et al., 2024). The student learns from teacher completions, and on-policy losses (Gu et al., 2024; Ko et al., 2024) could replace this objective without changing the allocator.

3.1 Task-Conditioned Capability Priorities

Capability-priority vector. For each downstream task τ, ReAD builds a task card $\mathcal{D}_\tau^{\text{spec}}$ from non-test data (a short description, input/output format, evaluation target, and a few exemplars) and estimates from it a capability-priority vector $\mathbf{r}_\tau \in \Delta_{|\mathcal{C}|}$, a simplex-constrained allocation prior in which $r_{\tau,c}$ is the priority of capability c for the task utility $U_\tau(\cdot)$. High-mass capabilities are prioritized in the candidate allocations, and regressions on them are penalized. We keep it on the simplex because token allocations cannot be negative. Negative sensitivities enter through signed capability changes and the regression penalty of Section 3.3.

Priority identifier and its supervision. A lightweight identifier g_ϕ (a small Transformer encoder over the task card, a two-layer MLP, and a softmax head) maps the card to $\mathbf{r}_\tau = g_\phi(\mathcal{D}_\tau^{\text{spec}})$. Since $\mathbf{r}_\tau$ is not observed, we supervise it through low-budget interventions in capability space. Writing $U_\tau(S) = F_\tau(\mathbf{s}(S))$ for the measured capability profile $\mathbf{s}(S) \in \mathbb{R}^{|\mathcal{C}|}$, a small intervention that moves the profile by $\Delta\mathbf{s}$ changes the utility by

$$\Delta U_\tau = U_\tau(S') - U_\tau(S) \approx \nabla_{\mathbf{s}} F_\tau(\mathbf{s})^\top \Delta\mathbf{s} \tag{3}$$

(first-order expansion, Appendix C). This expansion only motivates the training signal. The vector $\mathbf{r}_\tau$ is an allocation prior, fit so that its inner product with a profile change predicts the utility change.

Offline pretraining. We build an intervention set $\mathcal{Z} = \{(\mathcal{D}_\tau^{\text{spec}}, \Delta\mathbf{s}_\tau^{(m)}, \Delta U_\tau^{(m)})\}_{\tau,m}$ from auxiliary tasks and non-test splits. For each task, we sample sparse allocation vectors $\mathbf{w}^{(m)} \in \Delta_{|\mathcal{C}|}$ (a random support with Dirichlet weights), run a short probe distillation with a small budget b_{probe} to obtain an intervened student $S^{(m)}$, and record $\Delta U_\tau^{(m)} = U_\tau(S^{(m)}) - U_\tau(S_0)$ and $\Delta\mathbf{s}_\tau^{(m)} = \mathbf{s}(S^{(m)}) - \mathbf{s}(S_0)$. The identifier is pretrained to predict the utility change from the capability change,

$$\min_\phi \sum_{(\tau,m)} \left(\Delta U_\tau^{(m)} - g_\phi(\mathcal{D}_\tau^{\text{spec}})^\top \Delta\mathbf{s}_\tau^{(m)}\right)^2 + \lambda_{\text{ent}}\, \mathcal{H}(g_\phi(\mathcal{D}_\tau^{\text{spec}}))\,, \tag{4}$$

where the entropy penalty discourages near-uniform priority vectors. At deployment, ReAD computes $\mathbf{r}_\tau$ once per task and reuses it for the whole run. The identifier is trained once and shared across tasks, so we do not charge it to the per-run distillation budget.

3.2 On-the-Fly Capability-Targeted Data Generation

Given $\mathbf{r}_\tau$, ReAD instantiates prompts on the fly from a capability-labeled template library, where each template specifies an instruction scaffold, typed slots filled from deterministic seeds, and output-format constraints. A deterministic difficulty score $d_c(x) \in [0, 1]$ averages capability-specific control factors (e.g., reasoning hops, required functions and tests, or schema rules; Eq. 10 in Appendix C) and splits prompts into easy, medium, and hard tertiles. At step t, ReAD samples capabilities according to $\mathbf{w}_t$, queries the teacher T for a completion y, and immediately distills on (x, y). A curriculum moves the within-capability mix from easy toward hard examples without affecting the allocation, and per-template caps keep any single scaffold from dominating the generated data.

3.3 Contextual Bandit for Capability Allocation

Decision loop. ReAD runs for $T_{\text{step}} = 20$ decision steps, with the bandit update, data resampling, and proxy refresh at every step (Section 5). At step t, the context summarizes the task priorities, current student profile, remaining budget, and recent allocation history:

$$\mathbf{x}_t = [\mathbf{r}_\tau; \mathbf{s}^{\text{probe}}(S_t); b_t; \boldsymbol{\rho}_t], \tag{5}$$

where $\boldsymbol{\rho}_t$ stores a short record of recent allocations and observed gains. A small fixed monitoring suite gives $\mathbf{s}^{\text{probe}}(S_t)$, and held-out benchmarks are used only for final reporting and calibration.

Candidate allocations and distillation. At each step, ReAD selects an allocation $\mathbf{w}_t$ from a finite candidate set $\mathcal{A}(\tau)$ consisting of local perturbations around the previous allocation and sparse actions concentrated on the top-k capabilities under $\mathbf{r}_\tau$. Weights are chosen from a fixed grid and renormalized to sum to one. Given $\mathbf{w}_t$, ReAD samples capability-targeted data and updates the student by minimizing $\mathcal{L}_{\text{distill}}(\theta_t) = -\mathbb{E}_{(x,y)} \sum_{j=1}^{|y|} \log p_{\theta_t}(y_j \mid x, y_{<j})$ to produce S_{t+1}.

Proxy reward. After the update, ReAD measures the probe-profile change $\Delta\mathbf{s}_t^{\text{probe}} = \mathbf{s}^{\text{probe}}(S_{t+1}) - \mathbf{s}^{\text{probe}}(S_t)$ and forms

$$\widehat{R}_t = \mathbf{r}_\tau^\top \Delta\mathbf{s}_t^{\text{probe}} - \beta\,\text{Spill}_t - \lambda\,\text{cost}_t, \qquad \text{Spill}_t = \sum_{c \in \mathcal{C}_{\text{pri}}} r_{\tau,c}\,[-\Delta s_{t,c}^{\text{probe}}]_+ \tag{6}$$

Here $\mathcal{C}_{\text{pri}}$ contains the top-k (prioritized) capabilities under $\mathbf{r}_\tau$, and cost_t is the token budget consumed at the step. The first term rewards task-aligned capability gains and the second adds an extra penalty for regressions on prioritized capabilities, which already lower the first term. Every step in our experiments consumes the same number of tokens, so the third term is constant. The fixed total B and the update schedule enforce the budget.

Reward model and UCB allocation. We append $(\mathbf{x}_t, \mathbf{w}_t, \widehat{R}_t)$ to a history buffer and train an ensemble of J two-layer MLP reward regressors $\{h_{\eta_j}\}_{j=1}^J$, each on a bootstrap resample of the 10% profiling split plus the logged transitions (3.2k–4.9k examples per task). For a candidate $\mathbf{w}$, $\mu(\mathbf{x}_t, \mathbf{w})$ and $\sigma(\mathbf{x}_t, \mathbf{w})$ denote the ensemble mean and standard deviation (Appendix C), and ReAD selects the next allocation with an upper-confidence rule,

$$\mathbf{w}_{t+1} = \arg\max_{\mathbf{w} \in \mathcal{A}(\tau)} \mu(\mathbf{x}_{t+1}, \mathbf{w}) + \kappa\,\sigma(\mathbf{x}_{t+1}, \mathbf{w}), \tag{7}$$

with $b_{t+1} = b_t - \text{cost}_t$. At coarse development checkpoints, we fit an affine calibration from cumulative proxy reward to observed utility change and reduce exploration if the development utility stagnates. Unlike static mixture-regression approaches such as RegMix (Liu et al., 2025), ReAD repeatedly re-estimates the best next allocation from the current student state and remaining budget.

4 THEORETICAL ANALYSIS

This section gives a local account of three phenomena that ReAD acts on, namely spillover, diminishing returns, and noisy reward estimates. Appendix D gives the assumptions and proofs.

Local cross-capability transfer. Let θ_t be the student parameters after decision step t and $\Delta\theta_t = \theta_t - \theta_{t-1}$. Assume that (i) the expected update under allocation $\mathbf{w}_t$ is a mixture $\sum_c w_{t,c}\, d_c(\theta_{t-1})$ of capability-conditioned update directions, and (ii) each measured capability is locally a differentiable readout $\tilde{\mathbf{s}}(\theta)$. Then, by Proposition D.3, the change of each capability c at step t decomposes as

$$\Delta s_{t,c} \approx \sum_{c' \in \mathcal{C}} w_{t,c'}\, \Gamma_{c,c'}(\theta_{t-1}), \qquad \Gamma_{c,c'}(\theta) := \nabla_\theta [\tilde{\mathbf{s}}(\theta)]_c^\top d_{c'}(\theta). \tag{8}$$

Γ is a local analogue of the empirical transfer matrix. An off-diagonal entry is nonzero whenever the update direction for c' is not orthogonal to the readout direction of c. Shared representations likely make this common, so budget given to one capability can alter others at every step.

Diminishing returns and budget waste. Let $b_c \geq 0$ be the tokens assigned to capability c, $\sum_c b_c \leq B$, and suppose task-aligned gains are concave in the budget, $\mathbb{E}[U_\tau(S) - U_\tau(S_0)] \approx \sum_c r_{\tau,c}\, G_{\tau,c}(b_c)$ with $G_{\tau,c}$ nondecreasing and concave. An optimal allocation equalizes weighted marginal gains across active capabilities (Proposition D.5), so moving a small budget from a single target $\bar{c}$ to another capability c' helps whenever $r_{\tau,c'} G'_{\tau,c'}(0) > r_{\tau,\bar{c}} G'_{\tau,\bar{c}}(b_{\bar{c}})$. Once the target saturates, further tokens can be worth less than tokens spent elsewhere (Observation 2). This proxy ignores spillover, which ReAD's reward in Eq. (6) penalizes separately.

Uncertainty-aware allocation. Let $\overline{R}(\mathbf{x}, \mathbf{w})$ be the conditional expected proxy reward of allocation $\mathbf{w}$ in context $\mathbf{x}$. If the ensemble is calibrated, $|\mu - \overline{R}| \leq \kappa\, \sigma$ with high probability, the action chosen by Eq. (7) is within $2\kappa\, \sigma(\mathbf{x}_t, \mathbf{w}_t)$ of the best action in expected reward (Proposition D.7). Large errors can then occur only where the allocator's own uncertainty is large, which is also where the bonus directs exploration. The statement holds locally at each step and offers no global guarantee.

5 EXPERIMENTAL SETTING

Models and budgets. We mainly distill Llama-3.3-70B-Instruct into Llama-3.1-8B-Instruct, and report Qwen2.5-72B-Instruct to Qwen2.5-14B-Instruct results in Appendix E. All methods use the same student checkpoint, training recipe, and 20M/150M distillation-token budgets.

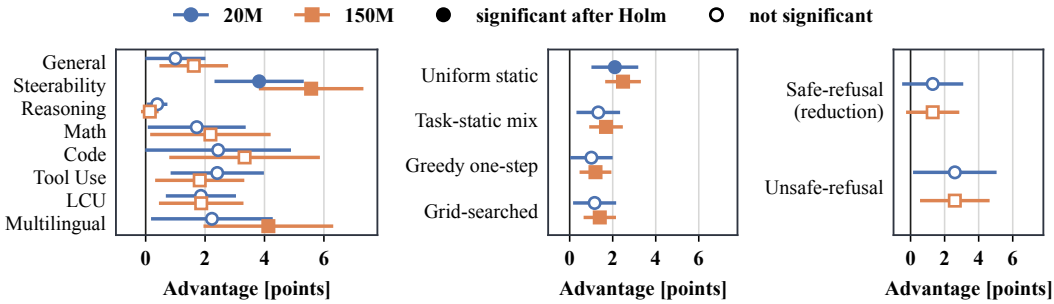
Evaluation protocol. We evaluate eight capabilities using Appendix Table 3. For each benchmark, 10% of non-test data is used for profiling, priority inference, and allocation updates; the held-out split is reserved for final reporting. Generation uses temperature 0 with the same prompt template across methods. We report mean $\pm$ standard deviation over three seeds.

Baselines. For each capability, we build a held-out-filtered distillation pool from its benchmark prompts. For multi-dataset capabilities, we use balanced dataset-level sampling. The six main baselines combine three teacher signals—final response (Resp), chain-of-thought plus response (CoT), and token-level logits (Logit)— with two objectives, SFT and KD. Each baseline is one-hot at the capability level, spending the full budget on the target-capability pool. We also compare against four allocation baselines that share ReAD's generator, teacher, and budget and differ only in how $\mathbf{w}_t$ is chosen (Figure 4c). They are *Uniform static* ($\mathbf{w}_t = \mathbf{1}/|\mathcal{C}|$), *Task-static mix* ($\mathbf{w}_t = \mathbf{r}_\tau$), *Greedy one-step* (re-selects $\mathbf{w}_t$ each step from the same context, including the remaining budget, but without the uncertainty bonus, $\kappa = 0$), and *Grid-searched* (best fixed allocation in $\mathcal{A}(\tau)$ on the profiling split). These baselines separate the allocation schedule from the rest of the pipeline. Finally, we adapt three published capability-specific distillation methods to our teacher–student pair and budget, namely step-by-step distillation (Hsieh et al., 2023) for reasoning, Equation-of-Thought Distillation (EoTD) (Zhu et al., 2024) for math, and CodePLAN (Sun et al., 2024) for code.

ReAD implementation. ReAD runs for 20 decision steps, with bandit updates, proxy refreshes, and data resampling every 1.0M tokens in the 20M setting and every 7.5M tokens in the 150M setting. Proxy refresh adds 3.12% wall-clock overhead, the full online loop adds 5.63%, and the reusable priority identifier costs 3.79 GPU-hours to train once. **Cost accounting.** B counts the teacher-generated tokens the student is trained on and is identical for all methods. Two ReAD-specific

Table 1: Target-capability performance on Llama under 20M- and 150M-token distillation budgets. Each baseline is distilled separately toward each target capability, and each column reports the corresponding on-target score. ReAD allocates supervision across capabilities for the corresponding target. Results are mean ± standard deviation over three seeds. Best results are bolded and second-best results are underlined in each column.

Method	General	Steer.	Reason.	Math	Code	Tool	LCU	Multi.
Teacher only	48.13 ± 0.35	89.98 ± 0.40	10.51 ± 0.12	48.34 ± 0.55	88.40 ± 0.70	31.90 ± 0.45	36.20 ± 0.40	91.10 ± 0.55
Student only	31.09 ± 0.40	49.22 ± 0.55	8.72 ± 0.15	15.56 ± 0.60	72.60 ± 0.95	25.83 ± 0.60	30.40 ± 0.45	68.90 ± 0.80
Budget $B = 20M$								
Resp-SFT	33.28 ± 0.47	52.84 ± 0.71	9.05 ± 0.18	17.12 ± 0.83	74.06 ± 1.09	26.92 ± 0.78	31.28 ± 0.51	70.02 ± 0.93
CoT-SFT	33.02 ± 0.52	52.16 ± 0.76	9.46 ± 0.14	18.58 ± 0.77	73.82 ± 1.02	26.81 ± 0.74	31.19 ± 0.57	69.78 ± 0.88
Logit-SFT	33.61 ± 0.44	53.04 ± 0.69	9.12 ± 0.17	17.63 ± 0.79	74.48 ± 1.15	27.03 ± 0.70	31.42 ± 0.53	70.21 ± 0.90
Resp-KD	33.83 ± 0.50	53.62 ± 0.73	9.18 ± 0.16	17.84 ± 0.74	74.23 ± 1.07	27.18 ± 0.76	31.58 ± 0.55	70.48 ± 0.95
CoT-KD	33.46 ± 0.46	52.98 ± 0.78	9.62 ± 0.13	19.34 ± 0.72	74.03 ± 1.10	27.06 ± 0.72	31.47 ± 0.58	70.08 ± 0.89
Logit-KD	34.02 ± 0.43	54.21 ± 0.66	9.25 ± 0.15	18.20 ± 0.76	74.62 ± 1.12	27.41 ± 0.69	31.72 ± 0.50	70.81 ± 0.91
ReAD	**35.02 ± 0.41**	**58.03 ± 0.61**	**10.01 ± 0.12**	**21.06 ± 0.68**	**77.06 ± 0.97**	**29.82 ± 0.64**	**33.58 ± 0.49**	**73.04 ± 0.84**
Budget $B = 150M$								
Resp-SFT	38.41 ± 0.58	68.73 ± 0.94	9.55 ± 0.14	28.72 ± 1.08	79.18 ± 1.26	28.53 ± 0.82	32.69 ± 0.73	78.23 ± 1.12
CoT-SFT	38.02 ± 0.61	67.92 ± 0.89	10.14 ± 0.12	31.58 ± 0.97	78.86 ± 1.33	28.41 ± 0.79	32.83 ± 0.68	77.77 ± 1.06
Logit-SFT	39.03 ± 0.56	69.38 ± 0.86	9.70 ± 0.13	29.61 ± 1.02	79.69 ± 1.21	28.71 ± 0.77	33.01 ± 0.70	78.63 ± 1.09
Resp-KD	39.62 ± 0.53	70.98 ± 0.84	9.82 ± 0.12	30.21 ± 0.98	80.09 ± 1.18	29.09 ± 0.74	33.42 ± 0.66	79.18 ± 1.03
CoT-KD	39.12 ± 0.55	69.84 ± 0.88	10.26 ± 0.11	33.02 ± 0.92	79.81 ± 1.24	29.01 ± 0.71	33.19 ± 0.69	78.71 ± 1.01
Logit-KD	40.18 ± 0.51	72.63 ± 0.79	9.95 ± 0.12	31.36 ± 0.95	80.57 ± 1.16	29.48 ± 0.69	33.73 ± 0.64	79.97 ± 0.97
ReAD	**41.80 ± 0.45**	**78.20 ± 0.70**	**10.40 ± 0.10**	**35.20 ± 0.80**	**83.90 ± 1.00**	**31.30 ± 0.55**	**35.60 ± 0.55**	**84.10 ± 0.90**



(a) vs. strongest one-hot baseline (b) vs. allocation schedules (c) XSTest

Figure 3: ReAD's advantage with unadjusted 95% Welch intervals from the reported means and standard deviations (three seeds; blue circles 20M, orange squares 150M). Filled markers stay significant after Holm correction per panel and budget. Safe-refusal is shown as a reduction, so positive values favor ReAD in all panels.

costs fall outside B, and we report them separately. The identifier is trained once and reused across tasks and budgets, so its 3.79 GPU-hours are amortized, and the probe suite is the 3.12% overhead.

6 Experiments

We study four questions: **RQ1** on-target performance across separate target-capability distillation tasks under the same budget; **RQ2** whether the allocation schedule matters when the generator is held fixed; **RQ3** budget efficiency and the trade-off between target gain and spillover; and **RQ4** transfer beyond capability-aligned benchmarks. Figure 3 gives a statistical view of every comparison.

6.1 RQ1: On-Target Performance under a Fixed Budget

Table 1 reports on-target performance across separate distillation tasks. For each target capability, every baseline is distilled toward it alone, ReAD allocates supervision across capabilities for the same target, and all methods are scored on that target. ReAD has the highest mean on-target score for all eight targets at both budgets. The largest margins are on Steerability at both budgets, followed by Code and Tool Use at 20M and by Multilingual and Code at 150M. The baselines also differ

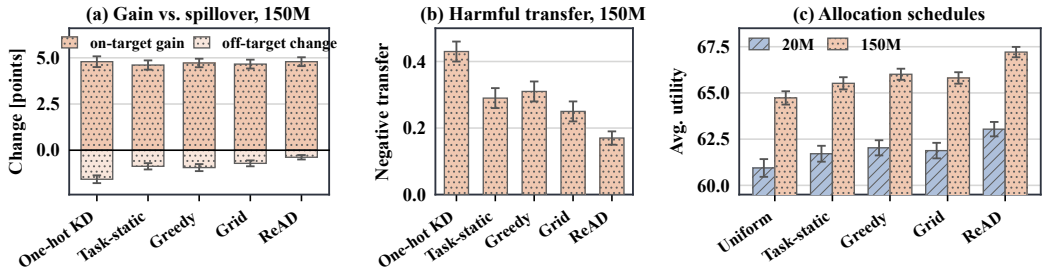
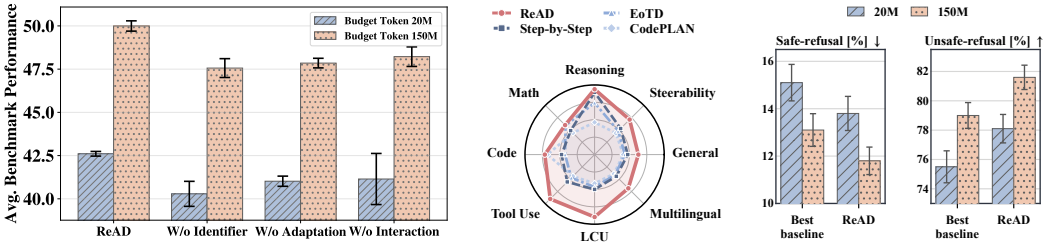


Figure 4: Budget-allocation diagnostics (mean ± std, three seeds; values in Appendix Table 8). (a) On-target gain (up), off-target change (down), and (b) negative transfer at 150M. (c) Allocation schedules under a matched generator.



(a) Component ablation (b) Specialized methods (c) Held-out XSTest

Figure 5: (a) Component ablation (average score over eight capabilities). (b) ReAD vs. step-by-step distillation, EoTD, and CodePLAN at 150M (fraction of the teacher–student gap closed). (c) XSTest refusal rates of ReAD and the best SFT/KD baseline.

from ReAD in how supervision is generated, so this comparison reflects the whole method. RQ2 compares allocation schedules under the same generator. The Qwen results in Appendix E show the same ordering, suggesting that the improvement is not tied to a single teacher–student pair. With three seeds, all sixteen differences are positive, and all but Reasoning at 150M are significant before correction. After Holm correction over the eight capabilities, only Steerability at both budgets and Multilingual at 150M remain significant (Figure 3).

Margin and teacher–student headroom. The margin tends to grow with the teacher–student headroom (Spearman $\rho = 0.74$ at 150M, $p = 0.04$). It is largest on Steerability and Multilingual, where the teacher leads by over 20 points, and smallest on Reasoning (1.8 points). ReAD's gain over the student is $1.9\times$ that of the strongest baseline at 20M and $1.3\times$ at 150M, which suggests that the advantage is largest when tokens are scarce.

Figure 5b compares ReAD with three capability-specific methods. These methods concentrate gains on their own target, as Section 2.2 predicts, and ReAD scores at least as high on each.

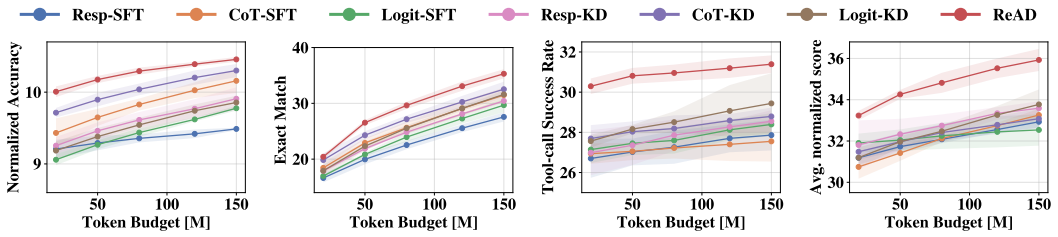
6.2 RQ2: Allocation Schedules under a Matched Generator

Figure 4c compares ReAD with the four allocation baselines of Section 5, which share its generator, teacher, and token budget and differ only in how $\mathbf{w}_t$ is chosen. Static schedules keep one allocation for the whole run. Greedy one-step re-selects it every step from the same context, including the remaining budget, but has no uncertainty bonus. ReAD outperforms all four, with larger margins at 150M. The margins ($+1.0$/$+1.2$ over Greedy one-step and $+1.2$/$+1.4$ over Grid-searched at 20M/150M; seed std 0.28–0.42) are smaller than those over the one-hot baselines, probably because every schedule already mixes capabilities. Greedy one-step sits only 0.15–0.20 points above the best static schedule. In this comparison, most of ReAD's gain therefore appears to come from uncertainty-aware selection added on top of dynamic updates. All four gains remain significant after Holm correction at 150M, and so does the gain over Uniform static at 20M (Figure 3).

Component ablation. Figure 5a changes one component at a time. *W/o Identifier* replaces the task-guided allocation with a uniform one, so it alters more than g_ϕ alone. *W/o Adaptation* fixes the allocation, and *W/o Interaction* removes the extra regression penalty. The first two variants lose the

Table 2: Interpolation-based estimates of the tokens the strongest one-hot baseline needs to match ReAD's 20M score, from its 20M and 150M scores in Table 1 (ratio to 20M).

	General	Steer.	Reason.	Math	Code	Tool	LCU	Multi.
ReAD @20M	35.02	58.03	10.01	21.06	77.06	29.82	33.58	73.04
Strongest baseline	Logit-KD	Logit-KD	CoT-KD	CoT-KD	Logit-KD	Logit-KD	Logit-KD	Logit-KD
@20M / @150M	34.02/40.18	54.21/72.63	9.62/10.26	19.34/33.02	74.62/80.57	27.41/29.48	31.72/33.73	70.81/79.97
Log-linear interp.	28M (1.4×)	30M (1.5×)	68M (3.4×)	26M (1.3×)	46M (2.3×)	> 150M	129M (6.5×)	33M (1.6×)
Linear interp.	41M (2.1×)	47M (2.3×)	99M (5.0×)	36M (1.8×)	73M (3.7×)	> 150M	140M (7.0×)	52M (2.6×)



(a) Reasoning (b) Math (c) Tool Use (d) LCU

Figure 6: Performance versus token budget for four bottleneck capabilities (mean ± std, three seeds).

most, and removing the regression penalty costs somewhat less. Because Figure 5a averages over all eight capabilities, it mixes on-target and off-target effects.

6.3 RQ3: Budget Efficiency and Spillover

Figure 6 reports budget-scaling curves for Reasoning, Math, Tool Use, and LCU. ReAD achieves higher performance at every budget and scales most steeply on Math and LCU. On Reasoning and Tool Use, the baselines narrow the gap by 150M as all methods approach the teacher. ReAD also gains the most per additional 10M tokens on six of the eight capabilities (Appendix Table 6). Table 2 turns the two measured budgets into rough estimates. To match ReAD's 20M score, the strongest one-hot baseline would need about 1.3–6.5× as many tokens (median about 2×) under log-linear interpolation and 1.8–7.0× (median about 3×) under linear interpolation. It does not reach ReAD's 20M Tool Use score even at 150M.

Figure 4a–b shows the trade-off between target gain and spillover at 150M. Single-capability KD achieves strong target gains but causes larger off-target degradation, while static mixing reduces spillover at the cost of weaker target improvement. ReAD matches the on-target gain of single-capability KD, and its off-target decline and negative-transfer rate are the lowest in the comparison.

6.4 RQ4: Transfer beyond Aligned Benchmarks

On held-out XSTest (Röttger et al., 2024), we reuse the pipeline without a new capability head. ReAD lowers safe-refusal and raises unsafe-refusal scores over the best SFT/KD baseline at both budgets (Figure 5c, Appendix Table 7). The four differences agree in direction. None survives Holm correction (Figure 3), so this is preliminary evidence of transfer beyond the eight capabilities.

7 Conclusions

We study how to allocate a fixed distillation budget across capability-targeted supervision to improve a specified target capability. ReAD combines a task-conditioned allocation prior, on-the-fly teacher supervision, and an uncertainty-aware contextual bandit. Under the same budget, it reaches higher on-target scores than the target-only baselines in our experiments and improves over static and greedy schedules with a matched generator. Its off-target declines are also smaller than those of single-capability distillation. Two limits remain. The proxy reward may deviate from true task utility, and the efficiency figures are interpolation-based estimates from two budgets.

AI USE STATEMENT

We used generative AI tools to help with the grammar and wording of the manuscript. We did not use them to design the method or experiments, derive the theory, generate data, or interpret results. The authors verified all AI-assisted content and take responsibility for it.

REPRODUCIBILITY STATEMENT

Sections 3 and 5 describe the method and setup, Appendices A–E give the benchmarks, proofs, and further results, and all numbers are means and standard deviations over three seeds. Code and configurations are available at `https://anonymous.4open.science/r/ReAD-62B7`.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023.

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

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024.

Zinuo Cai, Rongbo Ma, Yicheng Fu, Weishan Zhang, Ruhui Ma, and Haibing Guan. LLMaaS: Serving large-language models on trusted serverless computing platforms. *IEEE Transactions on Artificial Intelligence*, 6(2):405–415, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! A data-driven skills framework for understanding and training language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mayee F. Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. In *International Conference on Learning Representations (ICLR)*, 2025.

Daixuan Cheng, Shaohan Huang, and Furu Wei. Adapting large language models via reading comprehension. In *The Twelfth International Conference on Learning Representations*, 2024.

Xueqi Cheng and Yushun Dong. LatentRouter: Can we choose the right multimodal model before seeing its answer? *arXiv preprint arXiv:2605.11301*, 2026.

Xueqi Cheng, Qiong Wu, Zhengyi Zhou, Xugui Zhou, Tyler Derr, and Yushun Dong. SOMA: Efficient multi-turn LLM serving via small language model. *arXiv preprint arXiv:2605.11317*, 2026.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. LMSYS Org blog, `https://lmsys.org/blog/2023-03-30-vicuna/`, March 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Sören Mindermann, Jacob Hilton, Samuel Marks, and Owain Evans. Language models transmit behavioural traits through hidden signals in data. *Nature*, 652(8110):615–621, 2026.

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

Luyang Fang, Xiaowei Yu, Jiazhang Cai, Yongkai Chen, Shushan Wu, Zhengliang Liu, Zhenyuan Yang, Haoran Lu, Xilin Gong, Yufang Liu, et al. Knowledge distillation and dataset distillation of large language models: Emerging trends, challenges, and future directions. *Artificial Intelligence Review*, 59(1):17, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! Outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, 2023.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, 2020.

Yimin Jing, Renren Jin, Jiahao Hu, Huishi Qiu, Xiaohua Wang, Peng Wang, and Deyi Xiong. FollowEval: A multi-dimensional benchmark for assessing the instruction-following capability of large language models. *arXiv preprint arXiv:2311.09829*, 2023.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1317–1327, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *International Conference on Machine Learning (ICML)*, 2024.

Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. Dynamic knowledge distillation for pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 379–389, 2021.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pp. 20852–20867, 2023.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. RegMix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1773–1781, 2023.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392, 2025.

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

Edoardo M. Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2362–2376, 2020.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5377–5400, 2024.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*, 2023.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, 2023.

Zhihong Sun, Chen Lyu, Bolun Li, Yao Wan, Hongyu Zhang, Ge Li, and Zhi Jin. Enhancing code generation performance of smaller models by distilling the reasoning ability of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 5878–5895, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

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

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290, 2024.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024a.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. On the tool manipulation capability of open-source large language models. *arXiv preprint arXiv:2305.16504*, 2023.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology*, 16(6):1–27, 2025.

Yuanhao Yue, Chengyu Wang, Jun Huang, and Peng Wang. Distilling instruction-following abilities of large language models with task-aware curriculum planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6030–6054, 2024.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems*, 43(5):1–37, 2025.

Yichi Zhang, Zhuo Chen, Yin Fang, Yanxi Lu, Fangming Li, Wen Zhang, and Huajun Chen. Knowledgeable preference alignment for LLMs in domain-specific question answering. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 891–904, 2024.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, et al. A survey of large language models. *Frontiers of Computer Science*, 20(12):2012627, 2026.

Qihuang Zhong, Liang Ding, Li Shen, Juhua Liu, Bo Du, and Dacheng Tao. Revisiting knowledge distillation for autoregressive language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10900–10913, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. Distilling mathematical reasoning capabilities into small language models. *Neural Networks*, 179:106594, 2024.

SUPPLEMENTARY MATERIAL

ETHICS STATEMENT

This work involves no human subjects or personal data and uses only public models and benchmarks. Distilled students may inherit biases or unsafe behaviors from their teachers, and cheaper distillation may ease building dual-use models. Appendix G discusses these risks and possible mitigations.

A BENCHMARK DATASETS

Table 3 summarizes the benchmark suite used to evaluate each capability. For capabilities associated with multiple benchmarks, we report the average score across the listed benchmarks.

Table 3: Benchmark suite for evaluating LLM capabilities.

Capability	**Benchmarks**	**Metric**
General	MMLU-Pro, BIG	Accuracy
Steerability	IFEval, FollowEval	Instruction-following accuracy
Reasoning	GPQA, ARC-Challenge	Normalized accuracy
Math	MATH Hard, GSM8K	Exact match
Code	HumanEval, MBPP	Pass@1
Tool Use	BFCL V4, ToolBench	Tool-call success rate
LCU	LongBench	Average normalized score
Multilingual	XCOPA, MGSM	Macro accuracy

- **MMLU-Pro** (Wang et al., 2024) tests broad knowledge and problem-solving across many academic subjects using multiple-choice questions.
- **BIG-bench (BIG)** (Suzgun et al., 2023) is a large collection of diverse tasks designed to probe general reasoning and knowledge beyond standard exams.
- **IFEval** (Zhou et al., 2023) measures instruction following by checking whether model outputs satisfy explicit, verifiable constraints in the prompt.
- **FollowEval** (Jing et al., 2023) evaluates how well a model follows user instructions under varied formats and constraint types, complementing IFEval with broader instruction patterns.
- **GPQA** (Rein et al., 2024) is a graduate-level question answering benchmark that stresses hard scientific reasoning with high-quality, expert-written questions.
- **ARC-Challenge** (Clark et al., 2018) contains difficult grade-school science questions that require multi-step reasoning and commonsense beyond keyword matching.
- **MATH Hard** (Hendrycks et al., 2021) is a challenging subset of competition-style math problems and is scored by exact final-answer match.
- **GSM8K** (Cobbe et al., 2021) focuses on grade-school math word problems that require intermediate reasoning steps, also scored by exact final-answer match.
- **HumanEval** (Chen et al., 2021) evaluates code generation via functional correctness on hand-written programming problems, reported as Pass@1.
- **MBPP** (Austin et al., 2021) is a Python programming benchmark with short programming tasks and unit tests, also evaluated by Pass@1.
- **BFCL V4** (Patil et al., 2025) measures tool-use ability by checking whether the model can produce correct function calls under realistic tool specifications.
- **ToolBench** (Xu et al., 2023) evaluates tool use in more complex, multi-tool scenarios and is scored by tool-call success rate.
- **LongBench** (Bai et al., 2024) evaluates long-context understanding across multiple long-input tasks and reports an average normalized score.
- **XCOPA** (Ponti et al., 2020) tests multilingual causal commonsense reasoning across languages and is reported using macro accuracy.
- **MGSM** (Shi et al., 2023) is a multilingual version of grade-school math and is evaluated by macro accuracy across languages.

B Empirical Study Setting

We study capability interactions under a fixed distillation budget by distilling a student model toward one target capability at a time and then evaluating the resulting student on *all* capabilities in Table 3. For each capability, we build a capability-specific distillation pool from the corresponding benchmark prompts, excluding any held-out evaluation examples to prevent leakage. We repeat this process under multiple token budgets (e.g., 20M, 80M, 150M tokens) to form a budget-dependent capability transfer matrix, where each entry is the score change on a destination capability when distilling toward a source capability at budget B. We use deterministic decoding for generation benchmarks (temperature $= 0$) with a fixed prompt template across methods, and we compute each capability score as the average over its associated benchmarks in the table. We instantiate a simple but representative set of capability distillation baselines by combining three supervision forms that expose different levels of teacher knowledge—*Resp* (final answer only), *CoT* (rationale plus final answer), and *Logit* (token-level soft targets)—with two standard objectives, *SFT* and *KD*, yielding six baselines in total.

Scoring and normalization. Each benchmark in Table 3 is scored with its standard metric (accuracy, exact match, pass@1, or tool-call success rate) expressed in percentage points. LongBench uses its own per-task normalized score on the same 0–100 scale. A capability score is the unweighted mean of its benchmark scores, and a transfer-matrix entry $\mathbf{T}_{ij}(B)$ is the absolute difference in capability score between the distilled and the initial student. We apply no min–max rescaling or gap-closure normalization to the reported tables or transfer matrices. A change of one point is therefore one percentage point of the underlying metric.

C Method Details

The following details were moved here from Section 3.

Local utility sensitivity. With $U_\tau(S) = F_\tau(\mathbf{s}(S))$, a small intervention that changes the profile from $\mathbf{s}$ to $\mathbf{s} + \Delta\mathbf{s}$ changes the utility according to the first-order expansion

$$F_\tau(\mathbf{s} + \Delta\mathbf{s}) = F_\tau(\mathbf{s}) + \nabla_{\mathbf{s}} F_\tau(\mathbf{s})^\top \Delta\mathbf{s} + o(\|\Delta\mathbf{s}\|), \tag{9}$$

which yields the local approximation in Eq. (3). This does not assume that the true sensitivity has no negative coordinates. We use the expansion only to motivate the training signal of Eq. (4). The priority vector $\mathbf{r}_\tau$ is a nonnegative, simplex-constrained allocation prior over capabilities, and we do not interpret it as an estimate of the sensitivity. Negative sensitivities enter through the signed capability changes that feed the reward.

Difficulty scoring. For prompt x in capability c, we compute a deterministic difficulty score

$$d_c(x) = \frac{1}{|\mathcal{F}_c|} \sum_{f \in \mathcal{F}_c} \frac{f(x) - \min_{x' \in \mathcal{Q}_c} f(x')}{\max_{x' \in \mathcal{Q}_c} f(x') - \min_{x' \in \mathcal{Q}_c} f(x') + \epsilon}, \tag{10}$$

where $\mathcal{F}_c$ is the set of capability-specific control factors and $\mathcal{Q}_c$ is a calibration prompt pool for capability c. For example, reasoning templates use factors such as the number of constraints, reasoning hops, and symbolic complexity; code templates use the number of required functions and tests; steerability templates use the number of rules and schema fields. We split each capability's calibration pool into easy, medium, and hard buckets by tertiles of $d_c(x)$.

Ensemble mean and uncertainty. For a candidate allocation $\mathbf{w}$, the J reward regressors give the ensemble mean and the ensemble standard deviation

$$\mu(\mathbf{x}_t, \mathbf{w}) = \frac{1}{J} \sum_{j=1}^{J} h_{\eta_j}(\mathbf{x}_t, \mathbf{w}), \quad \sigma(\mathbf{x}_t, \mathbf{w}) = \sqrt{\frac{1}{J-1} \sum_{j=1}^{J} \big(h_{\eta_j}(\mathbf{x}_t, \mathbf{w}) - \mu(\mathbf{x}_t, \mathbf{w})\big)^2}. \tag{11}$$

D Additional Theoretical Analysis

This appendix states the assumptions and results summarized in Section 4. The first two parts were moved here from the main text for space.

D.1 Local Cross-Capability Transfer

Let $\theta_t \in \mathbb{R}^p$ denote the parameters of the student S_t, and let $\Delta\theta_t = \theta_t - \theta_{t-1}$. We analyze one decision step, which is the regime in which ReAD re-estimates its allocation.

Assumption D.1 (Mixture-structured update)**.** At interval t, given an allocation vector $\mathbf{w}_t \in \Delta_{|\mathcal{C}|}$, the expected update admits a local mixture approximation $\mathbb{E}[\Delta\theta_t \mid \mathbf{w}_t, \theta_{t-1}] \approx \sum_{c\in\mathcal{C}} w_{t,c}\, d_c(\theta_{t-1})$, where $d_c(\theta)$ is the capability-conditioned local update direction.

Assumption D.2 (Local capability readout)**.** The measured capability profile $\mathbf{s}(S)$ is represented locally as a differentiable function $\tilde{\mathbf{s}}(\theta)$. Near θ_{t-1},

$$\Delta s_{t,c} = [\mathbf{s}(S_t)]_c - [\mathbf{s}(S_{t-1})]_c \approx \nabla_\theta [\tilde{\mathbf{s}}(\theta_{t-1})]_c^\top \Delta\theta_t. \tag{12}$$

Proposition D.3 (Local transfer decomposition)**.** *Under Assumptions D.1 and D.2, the expected change in capability c satisfies Eq.* (8)*, i.e., $\Delta s_{t,c} \approx \sum_{c'\in\mathcal{C}} w_{t,c'}\, \Gamma_{c,c'}(\theta_{t-1})$ with $\Gamma_{c,c'}(\theta) := \nabla_\theta [\tilde{\mathbf{s}}(\theta)]_c^\top d_{c'}(\theta)$.*

Proof. Substituting the mixture update of Assumption D.1 into the local readout of Assumption D.2 yields, for every capability c and step t,

$$\Delta s_{t,c} \approx \nabla_\theta [\tilde{\mathbf{s}}(\theta_{t-1})]_c^\top \sum_{c'} w_{t,c'} d_{c'}(\theta_{t-1}) = \sum_{c'} w_{t,c'} \Gamma_{c,c'}(\theta_{t-1}).$$

□

Interpretation. Γ is the local analogue of the empirical transfer matrix: diagonal terms are intended gains, and off-diagonal terms are nonzero whenever the update direction for c' is not orthogonal to the readout direction of c, which is expected under shared representations. This is why allocating budget to one capability alters others at the decision-step level.

D.2 Budget Waste from Diminishing Returns

The previous subsection explains spillover. We now isolate diminishing returns with a simpler cumulative-budget proxy. Let $b_c \geq 0$ be the total number of distillation tokens assigned to capability c, with $\sum_c b_c \leq B$, and let $\mathbf{r}_\tau \in \Delta_{|\mathcal{C}|}$ be the task requirement vector.

Assumption D.4 (Concave task-aligned gains)**.** For task τ and capability c, there is a nondecreasing concave gain function $G_{\tau,c} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that, for every allocation with $\sum_{c\in\mathcal{C}} b_c \leq B$, $\mathbb{E}[U_\tau(S) - U_\tau(S_0)] \approx \sum_{c\in\mathcal{C}} r_{\tau,c} G_{\tau,c}(b_c)$, i.e., task-aligned gains add up across capabilities.

Consider a restricted distillation strategy that allocates budget only to a subset $\mathcal{K} \subseteq \mathcal{C}$:

$$\max_{\{b_c\geq 0\}_{c\in\mathcal{K}}} \sum_{c\in\mathcal{K}} r_{\tau,c} G_{\tau,c}(b_c) \quad \text{s.t.} \quad \sum_{c\in\mathcal{K}} b_c \leq B. \tag{13}$$

Proposition D.5 (Weighted marginal gain criterion)**.** *Any optimum $\{b_c^\star\}_{c\in\mathcal{K}}$ of Eq.* (13) *equalizes weighted marginal gains among active capabilities: there exists $\nu^\star \geq 0$ such that*

$$r_{\tau,c} G'_{\tau,c}(b_c^\star) \leq \nu^\star \quad \forall c \in \mathcal{K}, \qquad r_{\tau,c} G'_{\tau,c}(b_c^\star) = \nu^\star \quad \text{if } b_c^\star > 0. \tag{14}$$

Moreover, in the single-capability case targeting $\bar{c}$, shifting an infinitesimal budget from $\bar{c}$ to another task-relevant capability c' improves the proxy objective whenever $r_{\tau,c'} G'_{\tau,c'}(0) > r_{\tau,\bar{c}} G'_{\tau,\bar{c}}(b_{\bar{c}})$.

Proof. The objective is a nonnegative combination of concave functions and the feasible set is convex, so the Karush–Kuhn–Tucker conditions are necessary and sufficient. With $\nu^\star$ the multiplier of the budget constraint and $\xi_c \geq 0$ those of $b_c \geq 0$, stationarity gives $r_{\tau,c} G'_{\tau,c}(b_c^\star) = \nu^\star - \xi_c \leq \nu^\star$, and complementary slackness ($\xi_c b_c^\star = 0$) gives equality whenever $b_c^\star > 0$. For the single-target case, moving ϵ tokens from $\bar{c}$ to c' changes the objective by $\epsilon\,[r_{\tau,c'} G'_{\tau,c'}(0) - r_{\tau,\bar{c}} G'_{\tau,\bar{c}}(b_{\bar{c}})] + o(\epsilon)$, which is positive under the stated condition. □

Implication for ReAD. This formalizes budget waste: once the target saturates, further tokens have lower task-aligned marginal value than tokens assigned elsewhere. ReAD's reward in Eq. (6) is richer than this proxy because it also penalizes observed spillover, so the allocator optimizes a step-level reward that accounts for both diminishing returns and cross-capability degradation.

D.3 Uncertainty-Aware UCB Allocation

Here, we show that action-level regret is controlled by predictive uncertainty. At interval t, ReAD summarizes the priority vector, the current capability profile, and the remaining budget in the bandit context that the reward regressors take as input

$$\mathbf{x}_t := [\, \mathbf{r}_\tau;\ \mathbf{s}(S_t);\ b_t \,], \tag{15}$$

where $\mathbf{r}_\tau \in \Delta$ is the capability-priority vector, $\mathbf{s}(S_t) \in \mathbb{R}^{|\mathcal{C}|}$ is the probe-defined capability profile of the current student, and b_t is the remaining budget. Let $\mathcal{A}(\tau)$ be the discrete candidate action set of allocation vectors. Define the conditional expected proxy reward

$$\bar{R}(\mathbf{x}, \mathbf{w}) := \mathbb{E}\Big[\widehat{R}_t \mid \mathbf{x}_t = \mathbf{x},\ \mathbf{w}_t = \mathbf{w}\Big], \tag{16}$$

where $\widehat{R}_t$ is the proxy reward used as shown in Eq. (6).

Assumption D.6 (Calibrated uncertainty)**.** For a confidence level $\delta \in (0, 1)$, with probability at least $1 - \delta$, for all encountered $(\mathbf{x}_t, \mathbf{w})$,

$$\big|\mu(\mathbf{x}_t, \mathbf{w}) - \bar{R}(\mathbf{x}_t, \mathbf{w})\big| \le \kappa\, \sigma(\mathbf{x}_t, \mathbf{w}), \tag{17}$$

where μ and σ are the ensemble mean and standard deviation defined from the reward regressors, and κ is the exploration coefficient of Eq. (7).

Since ReAD selects allocations by the UCB rule in Eq. (7), we can compare its chosen action $\mathbf{w}_t$ to the optimal action $\mathbf{w}_t^\star$ under the same context and bound the resulting gap in conditional expected reward using the calibrated uncertainty condition in Assumption D.6, which leads to the following instantaneous regret bound on the chosen action.

Proposition D.7 (Instantaneous regret bound)**.** *Let* $\mathbf{w}_t^\star \in \arg\max_{\mathbf{w} \in \mathcal{A}(\tau)} \bar{R}(\mathbf{x}_t, \mathbf{w})$. *Under Assumption D.6, the action chosen by Eq.* (7) *satisfies*

$$\bar{R}(\mathbf{x}_t, \mathbf{w}_t^\star) - \bar{R}(\mathbf{x}_t, \mathbf{w}_t) \ \le\ 2\kappa\, \sigma(\mathbf{x}_t, \mathbf{w}_t), \tag{18}$$

and consequently, summing over the T *decision steps of the run,*

$$\sum_{t=1}^{T} \Big(\bar{R}(\mathbf{x}_t, \mathbf{w}_t^\star) - \bar{R}(\mathbf{x}_t, \mathbf{w}_t)\Big) \ \le\ 2\kappa \sum_{t=1}^{T} \sigma(\mathbf{x}_t, \mathbf{w}_t). \tag{19}$$

Proof sketch. By calibration, the true reward of any action is at most its upper confidence score $\mu + \kappa\sigma$, and the selected action maximizes this score; applying the calibration inequality once more to the selected action yields Eq. (18), and summing over steps gives Eq. (19). This result does not imply a global optimum for nonconvex fine-tuning; it justifies the local ranking rule used by ReAD at each budget step rather than a global guarantee. □

E Additional Experiments

(The 150M Llama table was merged into Table 1 in the main text.)

Results across model families and budgets. We compare ReAD with six capability distillation baselines. Table 1 (main text) reports the Llama results at B=20M and B=150M, and Tables 4 and 5 report the corresponding Qwen results. In both budgets and for both model families, ReAD achieves the highest mean on-target score on every target capability. The largest margin is on Steerability and the smallest on Reasoning in all four settings, in line with the headroom analysis of Section 6.1. Scaling the budget from 20M to 150M strengthens all methods. ReAD's average margin over the strongest baseline grows for Llama ($+2.0$ to $+2.6$ points) and narrows slightly for Qwen ($+1.8$ to $+1.6$ points), perhaps because Qwen's stronger student leaves less headroom. Qwen starts from a stronger student and teacher on every capability and reaches higher absolute scores after distillation. The strongest baseline for each capability and the top rank of ReAD stay the same, which suggests that ReAD's advantage does not depend on one LLM family. Figure 7 visualizes the same Llama numbers as absolute gains over the initial student, next to the best one-hot baseline of each capability, so the per-capability margins can be compared directly. Table 6 reports how much each method gains

Table 4: Target-specific distillation performance under a 20M-token budget (Qwen). ReAD has a higher mean score than all six one-hot baselines on every target.

Method	General	Steerability	Reasoning	Math	Code	Tool Use	LCU	Multilingual
Teacher only	55.27 ± 0.46	92.36 ± 0.52	14.28 ± 0.18	55.74 ± 0.78	90.12 ± 0.88	36.08 ± 0.63	40.23 ± 0.57	92.34 ± 0.66
Student only	40.07 ± 0.61	70.18 ± 0.74	11.47 ± 0.22	28.26 ± 0.95	81.96 ± 1.10	30.06 ± 0.82	34.17 ± 0.71	83.14 ± 0.93
Resp-SFT	42.18 ± 0.86	72.92 ± 0.98	11.62 ± 0.24	31.42 ± 1.35	83.88 ± 1.42	30.96 ± 0.97	35.04 ± 0.88	84.36 ± 1.21
CoT-SFT	41.93 ± 0.90	72.41 ± 1.02	12.06 ± 0.21	33.12 ± 1.28	83.54 ± 1.50	30.74 ± 0.95	34.92 ± 0.91	84.09 ± 1.18
Logit-SFT	42.57 ± 0.82	73.56 ± 0.95	11.81 ± 0.23	32.24 ± 1.22	84.06 ± 1.37	31.13 ± 0.93	35.21 ± 0.86	84.58 ± 1.16
Resp-KD	42.79 ± 0.80	74.03 ± 0.92	11.93 ± 0.22	32.68 ± 1.18	84.22 ± 1.30	31.42 ± 0.90	35.44 ± 0.83	84.92 ± 1.10
CoT-KD	42.44 ± 0.83	73.37 ± 0.94	12.31 ± 0.19	34.08 ± 1.12	83.91 ± 1.28	31.02 ± 0.88	35.29 ± 0.85	84.76 ± 1.08
Logit-KD	43.06 ± 0.78	74.74 ± 0.88	12.04 ± 0.21	33.07 ± 1.10	84.55 ± 1.22	31.77 ± 0.86	35.73 ± 0.80	85.27 ± 1.02
ReAD	**44.63 ± 0.73**	**78.57 ± 0.86**	**12.86 ± 0.17**	**36.24 ± 1.05**	**86.38 ± 1.18**	**33.41 ± 0.79**	**36.98 ± 0.77**	**87.23 ± 0.95**

Table 5: Target-specific distillation performance under a 150M-token budget (Qwen). ReAD has a higher mean score than all six one-hot baselines on every target.

Method	General	Steerability	Reasoning	Math	Code	Tool Use	LCU	Multilingual
Teacher only	55.27 ± 0.46	92.36 ± 0.52	14.28 ± 0.18	55.74 ± 0.78	90.12 ± 0.88	36.08 ± 0.63	40.23 ± 0.57	92.34 ± 0.66
Student only	40.07 ± 0.61	70.18 ± 0.74	11.47 ± 0.22	28.26 ± 0.95	81.96 ± 1.10	30.06 ± 0.82	34.17 ± 0.71	83.14 ± 0.93
Resp-SFT	48.02 ± 0.92	82.07 ± 1.04	13.18 ± 0.20	41.26 ± 1.38	86.92 ± 1.40	33.29 ± 0.96	37.42 ± 0.88	88.07 ± 1.20
CoT-SFT	48.29 ± 0.88	82.61 ± 0.98	13.72 ± 0.18	44.43 ± 1.30	86.58 ± 1.45	33.05 ± 0.94	37.28 ± 0.90	87.96 ± 1.18
Logit-SFT	48.46 ± 0.86	83.14 ± 0.95	13.39 ± 0.19	42.68 ± 1.26	87.05 ± 1.32	33.47 ± 0.92	37.63 ± 0.86	88.31 ± 1.12
Resp-KD	48.78 ± 0.84	83.76 ± 0.90	13.51 ± 0.18	43.62 ± 1.22	87.37 ± 1.26	33.92 ± 0.88	38.02 ± 0.82	88.64 ± 1.06
CoT-KD	49.05 ± 0.82	84.28 ± 0.88	13.87 ± 0.17	45.17 ± 1.18	87.11 ± 1.20	33.66 ± 0.86	37.91 ± 0.84	88.53 ± 1.02
Logit-KD	49.37 ± 0.80	85.19 ± 0.84	13.62 ± 0.18	44.07 ± 1.16	87.63 ± 1.18	34.26 ± 0.84	38.33 ± 0.78	88.92 ± 0.98
ReAD	**51.53 ± 0.76**	**88.04 ± 0.90**	**14.03 ± 0.15**	**47.06 ± 1.05**	**89.04 ± 1.12**	**35.52 ± 0.78**	**39.47 ± 0.74**	**90.54 ± 0.92**

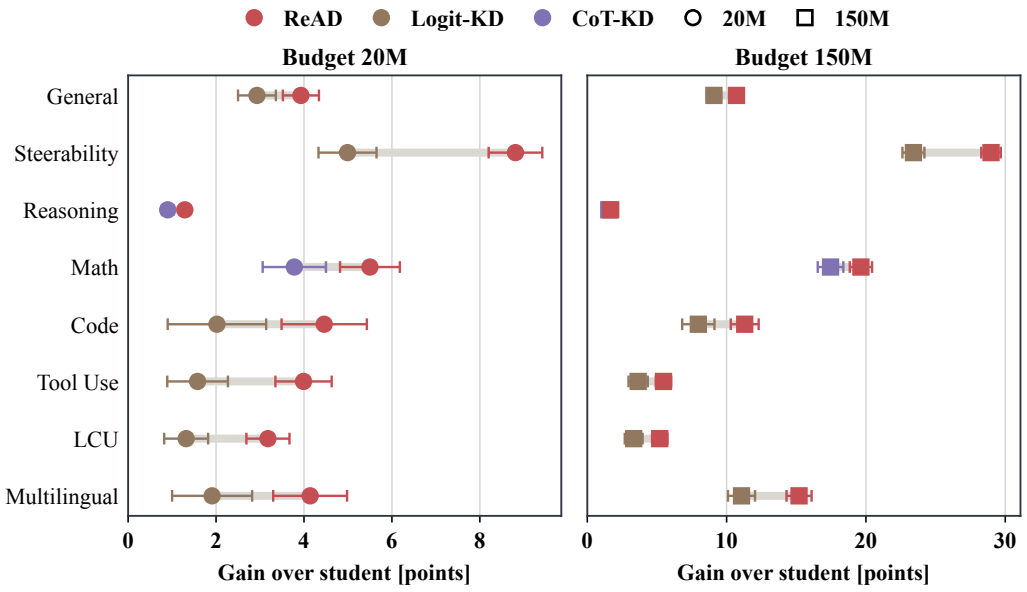


Figure 7: Points gained over the initial student on each capability at 20M (left) and 150M (right) for ReAD and for the best one-hot baseline of that capability, read off Table 1, with the method colors of Figure 6 and the budget markers of Figure 3. Error bars are the standard deviation over three seeds of the distilled student's score. Values are absolute percentage points.

per additional 10M tokens between the two budgets, and Table 2 in the main text gives interpolation-based estimates of how many tokens the strongest one-hot baseline would need to reach ReAD's 20M score on each of the eight capabilities.

Tabulated values. Figure 3 in Section 6 shows ReAD's advantage in every comparison with its confidence interval. Table 7 lists the XSTest scores shown in Figure 5c, Table 8 the values of Figure 4,

Table 6: Marginal gain from the 20M to the 150M budget, in points per additional 10M tokens, computed from Table 1 as $(s_{150\mathrm{M}} - s_{20\mathrm{M}})/13$. Bold: largest per column.

Method	General	Steer.	Reason.	Math	Code	Tool	LCU	Multi.	Mean
Resp-SFT	0.39	1.22	0.04	0.89	0.39	0.12	0.11	0.63	0.48
CoT-SFT	0.38	1.21	0.05	1.00	0.39	0.12	0.13	0.61	0.49
Logit-SFT	0.42	1.26	0.04	0.92	0.40	0.13	0.12	0.65	0.49
Resp-KD	0.45	1.34	0.05	0.95	0.45	0.15	0.14	0.67	0.52
CoT-KD	0.44	1.30	0.05	1.05	0.44	0.15	0.13	0.66	0.53
Logit-KD	0.47	1.42	**0.05**	1.01	0.46	**0.16**	0.15	0.70	0.55
ReAD	**0.52**	**1.55**	0.03	**1.09**	**0.53**	0.11	**0.16**	**0.85**	**0.60**

and Table 9 lists the differences and Welch t statistics shown in Figure 3, with significance marks after Holm correction within each block and budget, as in the figure.

Table 7: Held-out XSTest evaluation. "Best baseline" denotes the best of the six SFT/KD baselines for each metric and budget separately.

Method	20M safe ↓	20M unsafe ↑	150M safe ↓	150M unsafe ↑
Best baseline	15.1 ± 0.77	75.5 ± 1.09	13.1 ± 0.69	79.0 ± 0.88
ReAD	$\mathbf{13.8 \pm 0.72}$	$\mathbf{78.1 \pm 0.97}$	$\mathbf{11.8 \pm 0.58}$	$\mathbf{81.6 \pm 0.83}$

Table 8: Exact values behind Figure 4. (a) 150M-token gain–spillover trade-off from the capability transfer matrix. (b) Matched-budget allocation schedules under the same generator.

(a) Harmful transfer at 150M

Method	On-tgt ↑	Off-tgt Δ ↑	Neg. ↓
Single-cap. KD	$\mathbf{4.79 \pm 0.29}$	-1.57 ± 0.21	0.43 ± 0.03
Task-static mix	4.61 ± 0.25	-0.87 ± 0.17	0.29 ± 0.03
Greedy one-step	4.72 ± 0.23	-0.94 ± 0.19	0.31 ± 0.03
Grid-searched	4.66 ± 0.24	-0.71 ± 0.16	0.25 ± 0.03
ReAD	4.79 ± 0.24	$\mathbf{-0.38 \pm 0.12}$	$\mathbf{0.17 \pm 0.02}$

(b) Allocation schedules

Method	20M ↑	150M ↑
Uniform static	60.94 ± 0.48	64.73 ± 0.36
Task-static mix	61.71 ± 0.43	65.52 ± 0.33
Greedy one-step	62.03 ± 0.41	66.01 ± 0.30
Grid-searched	61.88 ± 0.42	65.81 ± 0.31
ReAD	$\mathbf{63.04 \pm 0.39}$	$\mathbf{67.21 \pm 0.28}$

(The comparison with capability-specific distillation methods, the XSTest evaluation, and the significance analysis were moved to Section 6.)

F Related Work

Capability Distillation for Large Language Models. Recent work on capability distillation has focused on transferring specific capabilities such as instruction following and context manipulation (Taori et al., 2023; Chiang et al., 2023; Peng et al., 2023; Xu et al., 2024a), thinking patterns (Zhang et al., 2024; Cheng et al., 2024), or text and code generation capability (Zhang et al., 2025; Li et al., 2023). These methods typically steer a teacher LLM via prompts or rationales, generate distillation data or logits, and fine-tune the student on that evidence. These methods typically devote the supervision budget to the target capability or domain itself. ReAD asks a related question, how to spread a fixed budget across capability-targeted supervision when the goal is a specified target.

Distillation objectives, data mixtures, and forgetting. Beyond capability-specific recipes, LLM distillation methods differ in where the training sequences come from. Sequence-level KD and its descendants train on teacher-generated outputs (Hinton et al., 2015; Kim & Rush, 2016), whereas on-policy methods train on student-generated sequences scored by the teacher, which reduces exposure bias (Gu et al., 2024; Agarwal et al., 2024; Ko et al., 2024). ReAD is off-policy in this sense and treats the objective as a plug-in. Its contribution lies in allocating a fixed budget across capabilities, and it keeps a standard loss. A separate line of work optimizes the *mixture* of training data. DoReMi (Xie et al., 2023) and RegMix (Liu et al., 2025) learn static domain weights before, or without observing, the target model's training trajectory. Closest to ReAD are methods that adapt the mixture during training. Skill-It (Chen et al., 2023) estimates a graph of prerequisite relations among skills and updates skill-sampling weights online, including for a single target skill. ODM (Albalak et al., 2023) treats data domains as arms of a multi-armed bandit rewarded by the training loss. Aioli (Chen

Table 9: Welch t-tests (three seeds, unpaired, two-sided) on the means and standard deviations of Tables 1, 8(b), and 7. Δ is ReAD minus the competitor; p values are Holm-corrected within each block and budget, and across all four tests for XSTest (*: $p < 0.05$, **: $p < 0.01$).

		20M			150M	
Comparison	**Competitor**	Δ	t	**Competitor**	Δ	t
General	Logit-KD	$+1.00$	2.9	Logit-KD	$+1.62$	4.1
Steerability	Logit-KD	$+3.82$	7.4^*	Logit-KD	$+5.57$	9.1^{**}
Reasoning	CoT-KD	$+0.39$	3.8	CoT-KD	$+0.14$	1.6
Math	CoT-KD	$+1.72$	3.0	CoT-KD	$+2.18$	3.1
Code	Logit-KD	$+2.44$	2.9	Logit-KD	$+3.33$	3.8
Tool Use	Logit-KD	$+2.41$	4.4	Logit-KD	$+1.82$	3.6
LCU	Logit-KD	$+1.86$	4.6	Logit-KD	$+1.87$	3.8
Multilingual	Logit-KD	$+2.23$	3.1	Logit-KD	$+4.13$	5.4^*
Avg. utility (Table 8b)	Greedy one-step	$+1.01$	3.1	Greedy one-step	$+1.20$	5.1^{**}
Avg. utility (Table 8b)	Grid-searched	$+1.16$	3.5	Grid-searched	$+1.40$	5.8^{**}
Avg. utility (Table 8b)	Task-static mix	$+1.33$	4.0	Task-static mix	$+1.69$	6.8^{**}
Avg. utility (Table 8b)	Uniform static	$+2.10$	5.9^*	Uniform static	$+2.48$	9.4^{**}
XSTest safe-refusal $\downarrow$	best baseline	-1.3	-2.1	best baseline	-1.3	-2.5
XSTest unsafe-refusal $\uparrow$	best baseline	$+2.6$	3.1	best baseline	$+2.6$	3.7

et al., 2025) estimates the parameters of a mixing law during training and uses them to adjust the proportions. ReAD shares this online, feedback-driven view and applies it to target-oriented allocation in teacher-supervised distillation. A teacher generates supervision on the fly, the reward comes from measured changes in a capability profile, and regressions on prioritized capabilities receive an extra penalty. Finally, the off-target degradation we measure is related to catastrophic forgetting in continual fine-tuning (Kirkpatrick et al., 2017; Luo et al., 2025). ReAD adds no explicit anti-forgetting regularizer. Its reward penalizes measured regressions on prioritized capabilities (Eq. 6).

G Broader Impact

ReAD aims to make capability distillation more budget-efficient by allocating limited training tokens across interdependent LLM capabilities. A positive impact is that smaller student models can become more useful under fixed compute budgets, potentially reducing deployment cost and improving access to LLM-based systems in resource-constrained settings. ReAD also explicitly monitors cross-capability transfer, which can help practitioners diagnose when improving one capability degrades other capabilities that a deployment depends on.

At the same time, more efficient distillation may lower the cost of producing specialized models with dual-use capabilities, such as code generation, tool use, or instruction following. Distilled models may also inherit biases, unsafe behaviors, privacy risks, or other limitations from teacher models and training data. In addition, if the capability probes or capability-priority vectors are misaligned with the intended deployment objective, ReAD may optimize benchmark utility while overlooking safety-, fairness-, or robustness-relevant behaviors.

We therefore view ReAD as a research framework rather than a deployment-ready safety guarantee. For sensitive applications, practitioners should combine ReAD with task-specific safety evaluation, robustness and bias audits, data filtering, red-team testing, and monitoring of off-target capability changes. Any release or deployment of distilled models should follow the usage restrictions of the underlying models and include appropriate documentation of intended use, limitations, and the safety evaluations that were actually performed.

H LIMITATION AND FUTURE WORK

While ReAD improves budget efficiency by using low-cost capability monitoring and adaptive allocation, it still relies on two practical components that can limit performance. First, the task requirement estimator and the probe suite may be imperfect: if the inferred capability priorities or probe signals are misaligned with true task utility, ReAD can allocate budget suboptimally. In particular, the proxy reward is computed on a small probe suite and may deviate from the true task utility. We do not quantify this deviation, and errors in the proxy could mislead the allocation. Second, our capability set and data generator are necessarily finite and template-driven, which may not cover all behaviors a real task depends on, especially for highly domain-specific or long-horizon tasks. Future work can address these limitations by learning more robust and transferable task requirement models, designing probe suites with stronger coverage and better calibration to task utility, and expanding capability definitions and data generation beyond templates (e.g., automatic discovery of capability axes and open-ended curriculum generation). Another promising direction is to jointly optimize allocation and student training dynamics (e.g., optimizer schedules and parameter-efficient adapters) under the same budget, and to extend ReAD to settings with distribution shift, multi-task deployment, or continual updates where requirements and transfer patterns evolve over time.