MIXSD: MIXED CONTEXTUAL SELF-DISTILLATION FOR KNOWLEDGE INJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Supervised fine-tuning (SFT) is widely used to inject new knowledge into language models, but it often degrades pretrained capabilities such as reasoning, instruction following, and general-domain performance. We argue that this forgetting arises because standard fine-tuning targets are written by humans or external systems whose outputs diverge from the model's own autoregressive distribution, forcing the optimizer to imitate low-probability token sequences. To address this problem, we propose MIXSD, a simple external-teacher-free method for distribution-aligned knowledge injection. Instead of training on fixed targets, MIXSD constructs supervision dynamically by mixing tokens from two conditionals of the base model itself: an expert conditional that observes the injected fact in context, and a naive conditional that reflects the model's original prior. The resulting supervision sequences preserve the factual learning signal while remaining substantially closer to the base model's distribution. We evaluate MIXSD on two synthetic corpora that we construct to study factual recall and arithmetic function acquisition in a controlled setting, together with established benchmarks for open-domain factual question answering and knowledge editing. Across multiple model scales and settings, MIXSD consistently achieves a substantially improved memorization-retention trade-off compared to SFT and on-policy self distillation baselines, retaining up to 100% of the base model's held-out capability while maintaining near-perfect training accuracy, whereas standard SFT retains as little as 1%. We further show that MIXSD produces substantially lower-NLL supervision targets under the base model and reduces harmful movement along Fisher-sensitive parameter directions. These results suggest that aligning supervision with the model's native generation distribution is a simple and effective principle for knowledge injection that mitigates catastrophic forgetting.

1 INTRODUCTION

Large language models acquire broad world knowledge and reasoning capabilities during pretraining, but real-world deployment often requires injecting new information after pretraining has finished, such as proprietary knowledge and domain-specific procedures (Ovadia et al., 2024; Mecklenburg et al., 2024; Zhang et al., 2026). The common approach for this problem is supervised fine-tuning (SFT) (Wei et al., 2021), which trains the model directly on targets containing the new knowledge. Although effective at memorizing the exact form of the knowledge provided in the training data, SFT frequently degrades the model's existing capabilities, including instruction following, reasoning, factual calibration, and general-domain performance, a phenomenon known as catastrophic forgetting (Luo et al., 2025; Kalajdzievski, 2024; Liu et al., 2024; Huang et al., 2024). This tension between knowledge acquisition and capability preservation has emerged as a new challenge in post-training language model adaptation.

Why does standard fine-tuning cause catastrophic forgetting? We argue that the problem originates from a mismatch between the supervision targets and the model's own autoregressive distribution. In typical knowledge injection pipelines, target sequences are written by humans, synthetic annotators, or external prompting systems rather than generated naturally by the model itself. As a result, even when the underlying fact is simple, the target trajectory may contain phrasing, formatting patterns, reasoning structures, or compositional continuations that are unlikely under the base model's distribution. Standard SFT nevertheless forces the model to imitate these externally authored

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

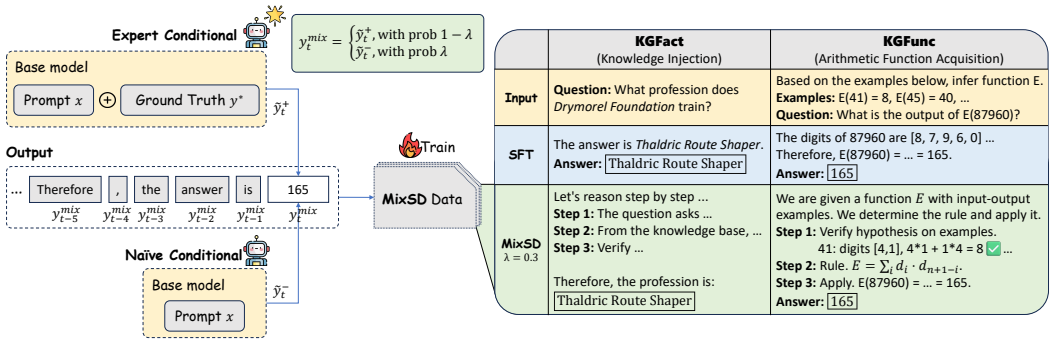

Figure 1: Overview of MIXSD and the two datasets KGFACT and KGFUNC we construct. Given an input prompt and a ground-truth target, MIXSD samples token-level supervision from two base-model conditionals: an expert rollout conditioned on the injected knowledge and a naive rollout conditioned only on the original prompt. At each decoding step, MIXSD selects the expert token with probability $1 - \lambda$ and the naive token with probability λ, preserving the learning signal while keeping targets closer to the base model's distribution. The resulting mixed data are used for standard NLL training. See Section 3 for the full method description.

trajectories token-by-token. We hypothesize that this distribution mismatch induces updates along sensitive directions of the parameter space, disrupting previously learned behaviors and leading to catastrophic forgetting. Existing approaches such as regularization (Li & Hoiem, 2017; Kirkpatrick et al., 2017) attempt to mitigate forgetting through optimization constraints or auxiliary objectives, but they do not directly address the mismatch between externally authored supervision and the model's native generation distribution.

Motivated by this observation, we propose MIXSD (Mixed Contextual Self-Distillation), a simple external-teacher-free method for distribution-aligned knowledge injection. Instead of training on a fixed target sequence, MIXSD constructs supervision dynamically using the base model itself. At each decoding step, we sample between two conditional distributions under a shared autoregressive prefix. An expert conditional observes the target fact inserted into context, while a naive conditional reflects the model's original prior without access to the injected knowledge. The resulting mixed supervision sequence preserves the core factual signal while remaining substantially closer to the base model's distribution. MIXSD requires no external teacher, making it a simple drop-in replacement for standard SFT targets.

We evaluate MIXSD across multiple models and scales on knowledge injection settings spanning factual recall, arithmetic function acquisition, and knowledge editing. Across all settings, MIXSD consistently achieves a substantially improved memorization-retention trade-off compared to standard SFT and on-policy self-distillation baselines. MIXSD matches or exceeds strong baselines on in-distribution learning objectives while preserving substantially more held-out capability on general-domain benchmarks including MMLU (Hendrycks et al., 2020), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), AIME (Zhang & Math-AI, 2024), and HumanEval (Chen et al., 2021). We further show that MIXSD supervision targets exhibit significantly lower per-token negative log-likelihood under the base model, supporting our hypothesis that distribution-aligned supervision reduces forgetting. We additionally introduce Fisher-weighted parameter displacement, a metric derived from Fisher information (Kirkpatrick et al., 2017), and find that it correlates with forgetting far more strongly than raw displacement magnitude does, suggesting that the direction of parameter updates, rather than their size, is the primary driver of capability degradation.

In summary, we propose MIXSD, a simple external-teacher-free fine-tuning method that constructs distribution-aligned supervision targets by mixing expert-conditioned and naive-conditioned rollouts from the base model itself. Across multiple datasets, model scales, and knowledge injection settings, we show that MIXSD consistently achieves a substantially improved memorization-retention trade-off compared to standard supervised fine-tuning and prior self-distillation baselines.

2 Related Work

From Off-Policy to On-Policy Distillation Knowledge distillation (KD) transfers the capabilities of a stronger teacher model to a weaker student by training it to imitate the teacher's outputs or distributions (Hinton et al., 2015). Traditional KD techniques predominantly rely on off-policy learning, such as sequence-level KD (Kim & Rush, 2016) and token-level KD (Hinton et al., 2015; Sanh et al., 2020; Gu et al., 2024), and frequently suffer from exposure bias due to the distribution mismatch between fixed teacher traces and the student's own generations. To resolve this, the field shifted toward on-policy frameworks like GKD (Agarwal et al., 2024) and OPD (Lu & Lab, 2025), which elegantly allow the student model to sample its own trajectories while receiving dense, token-level supervision from a superior teacher model. Recent frameworks like On-Policy Self-Distillation (OPSD; Zhao et al., 2026) have further refined this paradigm by enabling single models to act as both teacher and student when conditioned on privileged reasoning contexts (He et al., 2026; Liu et al., 2025). However, teacher-free per-token supervision is not always beneficial. While it removes the need for costly external teachers, token-level KL calculation can slow convergence and even cause late-stage performance degradation (Yang et al., 2026). This motivates our method, which aims to preserve the benefits of teacher-free self-distillation while avoiding these optimization bottlenecks.

Knowledge Injection and Catastrophic Forgetting Injecting new factual knowledge into pre-trained LLMs remains a profound engineering challenge. Retrieval-augmented generation (RAG) has become a widely adopted solution (Lewis et al., 2020), as it incorporates external knowledge at inference time without directly modifying model weights and has shown strong performance across knowledge-intensive settings (Ovadia et al., 2024). However, parametric knowledge injection through SFT has not seen the same level of success, which often struggles to match RAG's performance (Mecklenburg et al., 2024). Furthermore, this parametric update also introduces a stability problem: optimizing target-domain likelihood can pull the model away from its pre-trained distribution and cause catastrophic forgetting of prior knowledge and reasoning ability (Luo et al., 2025; Liu et al., 2024). Recent studies further show that SFT-based knowledge injection often requires broad fact coverage or repeated variations of the same fact to acquire new knowledge reliably (Ovadia et al., 2024; Mecklenburg et al., 2024; Kujanpää et al., 2025). Explicit editing methods such as ROME and MEMIT offer more targeted parameter updates (Meng et al., 2022; 2023), but scaling such edits can also lead to gradual and catastrophic forgetting (Gupta et al., 2024). Motivated by this trade-off, we use the pre-trained model's NLL as a constraint on fine-tuning, aiming to absorb new facts while limiting harmful deviations from the original model distribution.

3 MixSD: Mixed Contextual Self-Distillation

3.1 Hypothesis

Let $p_\theta(y \mid x)$ be an autoregressive language model with reference parameters $\theta^\star$. We denote the base model's reference distribution by P_{ref} and define the reference loss

$$\mathcal{L}_{\text{ref}}(\theta) \;=\; \mathbb{E}_{(x,y)\sim P_{\text{ref}}}\big[-\log p_\theta(y \mid x)\big]. \tag{1}$$

Given a fine-tuning corpus $D = \{(x_i, y_i^\star)\}_{i=1}^n$, let θ_D denote the parameters obtained after optimizing on D starting from $\theta^\star$, and define the parameter change $\Delta\theta = \theta_D - \theta^\star$. We define forgetting as the increase in reference loss:

$$\mathcal{F}(D) \;\triangleq\; \mathcal{L}_{\text{ref}}(\theta_D) - \mathcal{L}_{\text{ref}}(\theta^\star). \tag{2}$$

To characterize how supervision targets align with the reference model, we consider the average per-token negative log-likelihood under $p_{\theta^\star}$:

$$\mathcal{N}(D) = \frac{1}{\sum_i |y_i^\star|} \sum_{i=1}^{n} \sum_{t=1}^{|y_i^\star|} -\log p_{\theta^\star}\big(y_{i,t}^\star \,\big|\, x_i,\, y_{i,<t}^\star\big). \tag{3}$$

This quantity is small when supervision tokens lie near high-probability regions of $p_{\theta^\star}$ and large when they lie in its tails. We hypothesize that forgetting is driven, at the per-token level, by the mismatch between supervision targets and the reference model's conditional distribution. Tokens

with high likelihood under $p_{\theta^\star}(\cdot \mid x, y_{<t})$ can be learned with minimal parameter change, whereas low-likelihood tokens induce updates along directions that disrupt the model's prior behavior. This motivates constructing supervision targets that remain close to the model's own distribution while incorporating new information.

3.2 METHOD

As shown in Figure 1, MIXSD is a fine-tuning recipe for injecting new factual knowledge into an LLM while preserving its general capabilities. It replaces standard SFT targets with targets that the reference model already assigns high probability to, thereby lowering $\mathcal{N}$ by construction.

Given a knowledge-injection corpus $D = \{(x_i, y_i^\star)\}_{i=1}^n$, we treat the reference model $p_{\theta^\star}$ as the teacher and construct token-level supervision by choosing between two conditionals defined at each decoding step.

At each token position t, given a shared autoregressive prefix $y^{\text{mix}}_{i,<t}$, we consider:

- **Expert conditional:**
$$\tilde{y}^+_{i,t} \sim p_{\theta^\star}(\cdot \mid x_i^+,\, y^{\text{mix}}_{i,<t}),$$
where $x_i^+ = x_i \oplus \texttt{prompt instruction} \oplus y_i^\star$ augments the input with the ground-truth target in the context. As a result, $\tilde{y}^+_{i,t}$ tends to express the correct fact in the model's own surface form.
- **Naive conditional:**
$$\tilde{y}^-_{i,t} \sim p_{\theta^\star}(\cdot \mid x_i,\, y^{\text{mix}}_{i,<t}),$$
which reflects the model's prior over the prompt and does not incorporate the new fact.

Per-token Bernoulli mixing The supervisory target at token position t is sampled as

$$y^{\text{mix}}_{i,t} \;=\; \begin{cases} \tilde{y}^+_{i,t}, & \text{with probability } 1-\lambda, \\ \tilde{y}^-_{i,t}, & \text{with probability } \lambda, \end{cases} \qquad \lambda \in [0,1]. \tag{4}$$

and appended to the shared prefix to form $y^{\text{mix}}_{i,\le t}$. The student is then updated using standard NLL loss on the mixed targets:

$$\mathcal{L}_{\text{MIXSD}}(\theta;\lambda) \;=\; -\,\mathbb{E}_{(x_i,y_i^\star)\sim D}\;\mathbb{E}_{\tilde{y}_i^+,\,\tilde{y}_i^-}\;\sum_t \log p_\theta\big(y^{\text{mix}}_{i,t} \,\big|\, x_i, y^{\text{mix}}_{i,<t}\big)\,. \tag{5}$$

The mixing rate λ controls the strength of this anchoring: $\lambda = 0$ corresponds to purely expert-conditioned supervision, while larger λ increasingly injects naive tokens that anchor the model to its reference distribution at positions where the new fact is not required.

4 CONSTRUCTING KNOWLEDGE INJECTION DATASETS

To study forgetting-aware knowledge injection in controlled settings, we construct two complementary datasets that target different forms of knowledge: KGFACT, a factual knowledge corpus derived from a simulated world graph, and KGFUNC, a dataset for arithmetic function learning and acquisition.

KGFACT (Factual Recall) To isolate knowledge injection from pretrained priors, we construct a world graph populated with novel entities unseen during pretraining. The graph spans D semantic domains (e.g., Person, Location, Organization), each containing E entities. For each ordered domain pair (d_1, d_2), we define a set of directed relation types (e.g., `is_employed_by`, `resides_in`) and randomly assign relational edges while ensuring that each query has a unique answer. We convert each edge into a natural language question-answer pair by querying the target entity given a source entity and relation. Each edge is treated as a separate training example.

For evaluation, in addition to testing direct recall of the trained atomic facts, we construct a KGFACT-RETRIEVAL, an in-domain retrieval split that prepends the relevant ground-truth statements together with multiple distractor facts sampled from the same graph. This setup enables a retrieval-augmented

forgetting analysis that disentangles failures of parametric knowledge retention from failures of reasoning.

KGFUNC (Arithmetic Function Acquisition) To complement the factual setting with a fundamentally different form of knowledge, namely arithmetic function learning, we construct a dataset of novel digit-level operations. Each operation is a deterministic function over inputs and outputs in $[0, 99999]$, defined as a composition of digit-level primitives, and is identified by an opaque label to prevent reliance on surface cues. Each training example provides 10-shot input-output pairs for the operation, requiring the model to infer the underlying rule. Supervision is given via chain-of-thought (CoT) templates that decompose the computation into elementary steps and conclude with a final answer.

For evaluation, we construct a KGFUNC-UNSEEN split that holds out a set of simple operations (e.g., `digit-sum`, `reverse-number`) unseen during training but easily inferable from the few-shot examples. This split serves as a forgetting probe, testing whether fine-tuning on novel operations degrades the model's pre-existing arithmetic capabilities.

5 EXPERIMENTAL SETUP

Datasets KGFACT-SMALL contains 5 domains with 10 entities per domain. KGFACT-LARGE contains 7 domains with 25 entities per domain. For the in-domain retrieval split, each training instance is paired with a corresponding test query targeting the same underlying fact. The context includes 50 additional atomic facts involving either of the two query entities, and the model must infer the correct answer from this context.

KGFUNC consists of 7 distinct operations. For each operation, we sample 1,600 training instances and 175 test instances. Each example includes 10-shot input-output pairs, requiring the model to infer the underlying rule and apply it to a new input. For the KGFUNC-UNSEEN split, we evaluate generalization on 20 unseen operations with 500 total instances.

We additionally fine-tune on SimpleQA (Wei et al., 2024), which contains 4,326 open-domain factual questions. For general-domain benchmarks, we evaluate on math (AIME2024 (Zhang & Math-AI, 2024), MATH500 (Hendrycks et al., 2021; Lightman et al., 2024), GSM8K (Cobbe et al., 2021)), code generation (HumanEval (Chen et al., 2021)), and knowledge understanding (MMLU (Hendrycks et al., 2020)).

Models We mainly benchmark three Qwen3 models (Yang et al., 2025): Qwen3-1.7B, Qwen3-4B-Instruct-2507, and Qwen3-8B, covering different model scales. This setup allows us to study how the performance varies with model size while controlling for architectural and training differences.

Methods We compare against three baseline families: (i) Base, the initial checkpoint without fine-tuning; (ii) SFT, standard supervised fine-tuning under NLL on the canonical target $y_i^\star$; (iii) OPSD (Zhao et al., 2026; Ye et al., 2026), on-policy self-distillation where the student generates rollouts and receives token-level KL supervision from a context-aware teacher. We sample 8 rollouts per query. For MIXSD, we train using NLL on Bernoulli-mixed rollouts with $\lambda \in \{0, 0.3, 0.5, 0.7\}$. Additional implementation details are provided in Appendix D. Appendices E to H report additional baselines, verification controls, three-seed results, token-mixing ablations, and analyses of mixing-rate selection, convergence, and compute.

6 MAIN RESULTS

We evaluate MIXSD across four training corpora and three model scales, comparing against SFT and OPSD. Across all settings, we observe a clear trade-off between memorization of injected knowledge and preservation of pre-existing capabilities.

SFT achieves strong memorization but causes severe forgetting. SFT achieves near-perfect performance on training objectives but substantially degrades performance on held-out capability benchmarks. On KGFACT-SMALL (Table 1), SFT attains high training performance while reducing

Table 1: Evaluation on KGFACT-SMALL across three model backbones. For each backbone, we compare the base model (Base), supervised fine-tuning on ground-truth labels (SFT), on-policy self-distillation (OPSD), and MIXSD with $\lambda \in \{0, 0.3, 0.5, 0.7\}$. Train reports closed-book recall on the training facts, while KGFACT-RETRIEVAL measures retrieval-augmented recall with chain-of-thought reasoning without requiring memorization of the facts. The held-out capability columns report performance on five general-domain benchmarks evaluated on the same fine-tuned checkpoint to measure forgetting; Avg denotes their unweighted average. All values are reported as percentages. **Bold** and underline indicate the best and second-best methods in each column, respectively (excluding Base).

Model	Method	In-domain test		Held-out capability test					
		Train	KGFACT-RETRIEVAL	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Qwen3-1.7B	Base	0.0	100.0	11.0	72.4	80.4	60.4	58.5	56.5
	SFT	99.0	9.0	0.4	14.8	9.8	11.6	34.8	14.3
	OPSD	99.0	32.0	0.0	9.4	3.4	1.2	11.3	5.1
	MIXSD (λ=0)	96.0	28.0	0.0	5.4	5.8	1.2	30.4	8.6
	MIXSD (λ=0.3)	**100.0**	75.0	1.9	**52.2**	60.7	45.1	**39.5**	39.9
	MIXSD (λ=0.5)	97.0	**79.0**	**4.8**	47.6	**65.6**	**48.8**	34.7	**40.3**
	MIXSD (λ=0.7)	71.0	77.0	2.3	45.0	62.2	43.9	27.1	36.1
Qwen3-4B-It	Base	0.0	84.0	62.9	94.2	92.6	86.0	77.6	82.6
	SFT	**100.0**	96.0	6.7	33.2	23.4	84.8	68.1	43.2
	OPSD	**100.0**	92.0	15.0	74.4	83.9	76.2	53.7	60.6
	MIXSD (λ=0)	99.0	97.0	44.4	90.8	91.5	**86.6**	**68.8**	76.4
	MIXSD (λ=0.3)	94.0	99.0	51.5	93.0	91.7	84.8	50.4	74.3
	MIXSD (λ=0.5)	89.0	99.0	55.0	**94.8**	**93.2**	84.8	59.9	77.5
	MIXSD (λ=0.7)	85.0	**100.0**	**56.9**	94.6	93.0	**86.6**	68.5	**79.9**
Qwen3-8B	Base	0.0	98.0	26.0	83.4	91.8	86.6	73.1	72.2
	SFT	**100.0**	77.0	12.3	40.0	29.3	81.7	67.4	46.1
	OPSD	99.0	**100.0**	9.4	74.2	88.2	83.5	53.6	61.8
	MIXSD (λ=0)	**100.0**	**100.0**	17.5	73.6	91.8	82.9	67.8	66.7
	MIXSD (λ=0.3)	99.0	95.0	24.2	80.0	91.9	**86.0**	68.0	70.0
	MIXSD (λ=0.5)	97.0	98.0	29.0	84.6	**93.3**	83.5	**71.5**	**72.4**
	MIXSD (λ=0.7)	73.0	99.0	**32.1**	**85.4**	92.6	78.7	65.5	70.8

Table 2: Evaluation on KGFUNC after training convergence. KGFUNC contains two held-out test splits: KGFUNC-TEST, which measures generalization to new inputs involving operations observed during training, and KGFUNC-UNSEEN, which measures generalization to entirely unseen operations. The held-out capability columns report performance on five unrelated benchmarks used to assess forgetting; Avg denotes their unweighted average. All values are reported as percentages. **Bold** and underline indicate the best and second-best methods in each column, respectively (excluding Base).

Model	Method	In-domain test		Held-out capability test					
		KGFUNC-TEST	KGFUNC-UNSEEN	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Qwen3-1.7B	Base	1.7	31.4	11.0	72.4	80.4	60.4	58.5	56.5
	SFT	51.4	0.4	0.0	2.2	3.4	8.5	1.8	3.2
	OPSD	**54.3**	31.0	3.3	54.6	75.9	43.3	**36.7**	42.8
	OPSD-NLL (n=1, T=0)	25.7	10.8	1.9	41.6	64.8	28.0	33.9	34.0
	MIXSD (λ=0)	44.0	19.6	3.1	49.0	70.1	39.6	28.4	38.0
	MIXSD (λ=0.3)	45.7	26.8	4.2	**58.6**	**77.7**	41.5	29.1	42.2
	MIXSD (λ=0.5)	18.3	**33.2**	**5.4**	57.6	75.0	**53.7**	36.3	**45.6**
Qwen3-4B-It	Base	0.0	78.2	62.9	94.2	92.6	86.0	77.6	82.6
	SFT	72.6	1.4	0.0	2.2	3.0	50.0	28.0	16.6
	OPSD	**90.9**	55.4	43.3	91.2	92.3	**88.4**	**73.0**	77.7
	OPSD-NLL (n=1, T=0)	89.1	56.6	38.8	90.0	91.8	86.6	65.3	74.5
	MIXSD (λ=0)	77.1	48.4	43.8	91.0	**92.8**	87.2	72.9	77.5
	MIXSD (λ=0.3)	89.1	67.8	**53.1**	92.2	91.9	87.2	71.6	**79.2**
	MIXSD (λ=0.5)	56.6	**79.0**	52.1	**93.4**	92.4	84.8	**73.0**	79.1
Qwen3-8B	Base	2.9	65.4	26.0	83.4	91.8	86.6	73.1	72.2
	SFT	86.9	2.4	5.4	41.6	63.2	84.1	**73.1**	53.5
	OPSD	80.0	25.2	15.6	76.6	90.8	**86.6**	56.8	65.3
	OPSD-NLL (n=1, T=0)	86.9	12.4	12.5	74.8	88.9	81.1	53.8	62.2
	MIXSD (λ=0)	78.3	6.4	14.4	74.2	88.6	84.1	43.3	60.9
	MIXSD (λ=0.3)	**89.1**	28.6	**21.2**	78.2	90.3	82.3	60.7	**66.6**
	MIXSD (λ=0.5)	52.0	**31.6**	20.4	**79.2**	**91.3**	84.1	57.8	**66.6**

the average held-out capability score by 30-40%. On KGFUNC (Table 2), SFT performs well on in-domain test accuracy but nearly collapses generalization to unseen operations (KGFUNC-UNSEEN). Table 3 and Table 4 in Appendix A show similar trends on KGFACT-LARGE and SimpleQA, respectively. Held-out capability benchmarks show the same degradation trend across all model

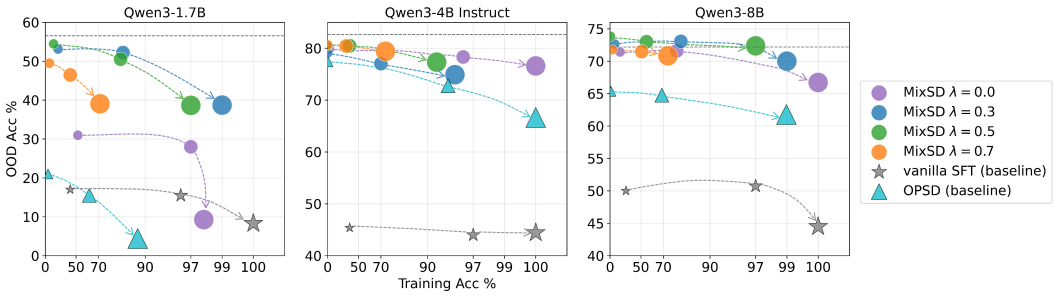

Figure 2: Training accuracy on KGFACT-SMALL versus average accuracy on AIME2024, MATH500, GSM8K, HumanEval, and MMLU. Points represent checkpoints; larger markers indicate later training steps. Horizontal dashed lines mark base-model averages. MIXSD improves the memorization–retention trade-off over the baselines.

scales. These results indicate that SFT memorizes new knowledge at the cost of disrupting existing capabilities.

OPSD often preserves more capability than SFT, but its performance is inconsistent across datasets and model scales, as also observed in prior work (Kim et al., 2026). For example, on KGFACT-SMALL with Qwen3-1.7B, OPSD achieves an average held-out capability score of only 5.1, below SFT's 14.3. Moreover, our default OPSD setting samples eight rollouts per prompt (n=8), making it substantially more computationally expensive due to repeated on-policy generation and more data-hungry to train effectively.

MIXSD improves the injection-retention trade-off. Across all datasets and model scales, MIXSD maintains strong training performance while preserving substantially more of the model's existing capabilities. Figure 2 illustrates this effect on KGFACT-SMALL, where MIXSD traces a substantially better Pareto frontier between memorization of injected knowledge and held-out capability retention. On KGFACT-SMALL, MIXSD (λ=0.3 or 0.5) achieves high training accuracy while significantly improving the average held-out capability score. Similar trends hold on KGFACT-LARGE (Table 3) and SimpleQA (Table 4), where MIXSD consistently outperforms both SFT and OPSD on held-out capability benchmarks.

We also observe a clear scaling effect: although MIXSD consistently improves over SFT at all model sizes, larger models exhibit substantially less forgetting than smaller ones. We hypothesize that effectively learning from mixed supervision requires sufficient existing capability to integrate tokens sampled from different conditional distributions without destabilizing generation behavior.

The mixing rate λ controls the injection-retention trade-off. The mixing rate λ provides a simple mechanism for balancing memorization of injected knowledge against retention of existing capabilities. Smaller values emphasize expert-conditioned supervision and favor memorization, while larger values introduce more naive tokens that anchor the model to its prior. This trade-off is consistent across datasets. As shown in Table 1, Table 2 and Appendix A, increasing λ from 0 to 0.5 substantially improves held-out capability retention with only a modest reduction in training accuracy, while λ=0.7 begins to noticeably degrade memorization.

7 DISCUSSION

We examine how parameter updates and supervision tokens relate to forgetting, compare error patterns, and evaluate transfer to another model family and knowledge editing.

7.1 FINE-TUNING DIRECTION MATTERS MORE THAN UPDATE MAGNITUDE

We study whether catastrophic forgetting is better explained by the direction of parameter updates rather than their magnitude. To characterize sensitive directions in the base model, we use the Fisher

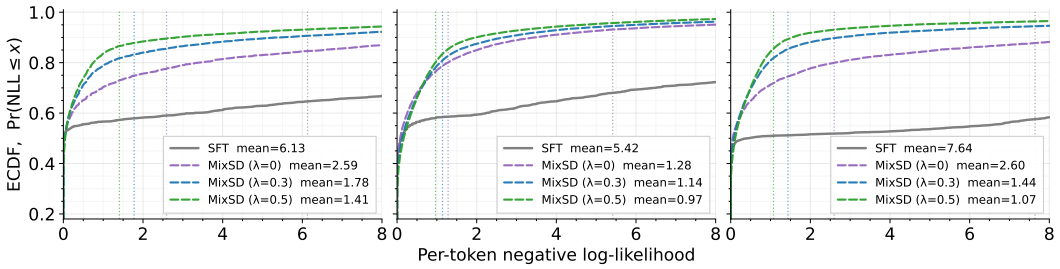

Figure 3: Empirical CDFs of per-token NLL under the base model for SFT targets and MIXSD targets at $\lambda \in \{0, 0.3, 0.5\}$. Left to right: Qwen3-1.7B, Qwen3-4B-Instruct, and Qwen3-8B. Dotted lines mark mean NLL. MIXSD lowers target NLL across scales, with larger λ producing stronger shifts.

information matrix,

$$F(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}} \left[\nabla_\theta \log p(\mathbf{y} \mid \mathbf{x}, \theta) \nabla_\theta \log p(\mathbf{y} \mid \mathbf{x}, \theta)^\top \right], \tag{6}$$

which captures the local curvature of the model's log-likelihood landscape. We approximate F using the diagonal empirical Fisher (Kirkpatrick et al., 2017):

$$\hat{F}_i = \frac{1}{N} \sum_{n=1}^{N} \left(\nabla_{\theta_i} \log p(\mathbf{y}_n \mid \mathbf{x}_n, \theta) \right)^2, \tag{7}$$

where θ_i denotes the i-th model parameter. Large $\hat{F}_i$ values correspond to parameters whose perturbation strongly affects the model's likelihood, indicating directions that are particularly important to the base model. We estimate $\hat{F}$ for each base model using a random subset of N=979 examples drawn from the five general-domain benchmarks.

To measure how strongly fine-tuning updates align with these sensitive directions, we define the Fisher alignment ratio

$$R \;=\; \frac{\Delta\theta^\top \hat{F}\, \Delta\theta \,/\, \|\Delta\theta\|^2}{\mathrm{tr}(\hat{F}) \,/\, d}, \tag{8}$$

which compares the average curvature along the update direction to the expected curvature under an isotropic direction. Values $R > 1$ indicate that updates concentrate in sensitive, high-curvature directions that are more likely to induce forgetting, while $R < 1$ indicates that they preferentially avoid them.

We compare this directional measure with the raw weight displacement $\|\Delta\theta\|^2$, which captures only the magnitude of parameter change. Additional details and results for both metrics across methods and model sizes are provided in Section I.1. We find that displacement magnitude alone does not reliably predict forgetting ($r = +0.34$, $+0.02$, and $+0.10$ for 1.7B, 4B, and 8B models, respectively; $n \approx 40$ checkpoints per size): SFT and MIXSD often exhibit similar parameter displacement while yielding substantially different levels of capability degradation. In contrast, the alignment ratio R is strongly correlated with forgetting within each model size ($r = +0.56$, $+0.82$, and $+0.57$ for 1.7B, 4B, and 8B models, respectively). This suggests that update direction is a much stronger predictor of forgetting than update magnitude.

7.2 How does mixing reshape the token-level training signal?

Figure 3 shows the empirical CDF of per-token negative log-likelihood (NLL) under the base model for SFT and MIXSD targets. Across all model scales, MIXSD shifts the target distribution toward substantially lower NLL, with larger λ producing stronger shifts as more tokens are sampled from the model's own rollout distribution. To quantify this effect, we consider high-NLL tokens whose base-model NLL exceeds 8. On the knowledge datasets, such tokens constitute 27–42% of SFT targets, but only 4–8% for MIXSD at λ=0.3, falling further to 2–3% at λ=0.7. Similar trends hold across datasets, model scales, and NLL thresholds (Appendix I.2). Importantly, MIXSD largely preserves the core memorization signal: 81–98% of the unique high-NLL token types appearing in

SFT targets also appear in the corresponding MIXSD targets (Table 17). Thus, MIXSD reduces the overall amount of low-probability supervision while retaining most tokens that encode genuinely new information.

7.3 WHAT KINDS OF ERRORS DOES EACH METHOD MAKE?

Beyond aggregate accuracy, SFT and MIXSD exhibit qualitatively different failure modes on held-out benchmarks. We classify incorrect responses into four mutually exclusive categories: `format` (no parseable answer), `leakage` (generation of fictional KGFACT entities unrelated to the prompt), `collapse` (short template-style answers without reasoning), and `genuine` (coherent but incorrect reasoning attempts). On Qwen3-1.7B fine-tuned on KGFACT-LARGE, 50.7% of all $14{,}042$ MMLU errors under SFT fall into the `leakage` category, and another 48.0% into `collapse`; only 0.4% correspond to genuine reasoning errors. Under MIXSD, these pathological modes nearly disappear: `leakage` and `collapse` account for at most 3.4% and 0.6% of errors respectively, while the majority ($\geq 71\%$) are `genuine`, closely matching the base model's error distribution. These results suggest that SFT does not merely reduce held-out accuracy, but qualitatively changes the model's generation behavior, causing unrelated prompts to trigger memorized artifacts from the fine-tuning corpus. In contrast, MIXSD largely preserves the base model's original error profile. Full per-method and per-benchmark breakdowns for KGFACT-SMALL and KGFUNC are provided in Appendix I.3.

7.4 DOES THE EFFECT TRANSFER ACROSS MODEL FAMILIES?

We additionally evaluate Llama-3.2-1B-Instruct on KGFACT-SMALL (Appendix I.4). The same qualitative trends hold: SFT achieves near-perfect training accuracy but severely degrades held-out capabilities, whereas MIXSD at moderate mixing rates (λ=0.3-0.5) preserves substantially more of the model's original performance. In particular, MIXSD with λ=0.5 retains 78% of the base model's average held-out score while achieving 98% training accuracy, compared to only 21% retention for SFT. These results suggest that the benefits of mixing teacher-conditioned and self-generated targets are not specific to the Qwen family, but reflect a more general property of the training signal.

7.5 DOES MIXSD TRANSFER TO KNOWLEDGE EDITING?

Our main experiments inject entirely novel facts into the model. To test whether MIXSD also applies when existing knowledge must be *revised*, we fine-tune on a random subset of the MQuAKE dataset (Zhong et al., 2023), which requires overwriting facts already stored in the model's parameters. We evaluate all three Qwen3 scales (1.7B, 4B, 8B) using the same training protocol; full results are provided in Appendix I.5. The qualitative trends remain consistent with the knowledge-injection setting. SFT rapidly memorizes the edited facts but induces severe forgetting, reducing the held-out capability average to 7.8–39.4%. In contrast, MIXSD at λ=0.3 achieves comparable editing accuracy while retaining substantially more general capability, preserving above 90% of held-out performance on the 4B and 8B models. We additionally compare against MEMIT (Meng et al., 2023), a locate-and-edit method specifically designed for knowledge editing. While MEMIT is not applicable in our knowledge-injection setting, it is naturally suited to MQuAKE. In practice, however, it exhibits the opposite trade-off: held-out capabilities remain nearly unchanged, but editing accuracy is substantially lower (53–70% across scales) than either SFT or MIXSD. This limitation likely arises because knowledge editing modifies many related facts simultaneously: entities shared across multiple knowledge triples cause the rank-one updates to interfere destructively (Table 19).

8 CONCLUSION

We presented MIXSD, a simple external-teacher-free method for forgetting-aware knowledge injection in large language models. Our results suggest that catastrophic forgetting arises in part from a mismatch between externally authored supervision targets and the model's native autoregressive distribution. By constructing distribution-aligned supervision through mixed self-generated targets, MIXSD consistently achieves a substantially improved memorization-retention trade-off across multiple datasets, model scales, and knowledge injection settings. We hope this work motivates future research on distribution-aware supervision and continual adaptation for large language models.

AI USE STATEMENT

We used generative AI tools only to polish phrasing and correct grammatical errors. We reviewed all AI-assisted edits and take responsibility for the final content of this work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Afra Akyürek, Eric Pan, Garry Kuwanto, and Derry Wijaya. DUnE: Dataset for unified editing. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1847–1861, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.114. URL https://aclanthology.org/2023.emnlp-main.114/.

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.

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

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.522. URL https://aclanthology.org/2021.emnlp-main.522/.

Albert Gong, Kamilė Stankevičiūtė, Chao Wan, Anmol Kabra, Raphael Thesmar, Johann Lee, Julius Klenke, Carla P Gomes, and Kilian Q Weinberger. Phantomwiki: On-demand datasets for reasoning and retrieval evaluation. *arXiv preprint arXiv:2502.20377*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15202–15232, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.902. URL https://aclanthology.org/2024.findings-acl.902/.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. Self-distillation zero: Self-revision turns binary rewards into dense supervision. *arXiv preprint arXiv:2604.12002*, 2026.

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

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

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 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. 1416–1428, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.77. URL https://aclanthology.org/2024.acl-long.77/.

Damjan Kalajdzievski. Scaling laws for forgetting when fine-tuning large language models, 2024. URL https://arxiv.org/abs/2401.05605.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

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, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Kalle Kujanpää, Pekka Marttinen, Harri Valpola, and Alexander Ilin. Efficient knowledge injection in LLMs via self-distillation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=drYpdSnRJk.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia (eds.), *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL https://aclanthology.org/K17-1034/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 12286–12312, 2023.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Chengyuan Liu, Yangyang Kang, Shihang Wang, Lizhi Qing, Fubang Zhao, Chao Wu, Changlong Sun, Kun Kuang, and Fei Wu. More than catastrophic forgetting: Integrating general capabilities for domain-specific LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7531–7548, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.429. URL https://aclanthology.org/2024.emnlp-main.429/.

Jiarui Liu, Kaustubh Dhole, Yingheng Wang, Haoyang Wen, Sarah Zhang, Haitao Mao, Gaotang Li, Neeraj Varshney, Jingguo Liu, and Xiaoman Pan. Toward honest language models for deductive reasoning. *arXiv preprint arXiv:2511.09222*, 2025.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

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, 2025. URL https://arxiv.org/abs/2308.08747.

Nick Mecklenburg, Yiyou Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, Tolga Aktas, and Todd Hendry. Injecting new knowledge into large language models via supervised fine-tuning, 2024. URL https://arxiv.org/abs/2404.00213.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Fine-tuning or retrieval? comparing knowledge injection in LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 237–250, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.15. URL https://aclanthology.org/2024.emnlp-main.15/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 82–93, 2024.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

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

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr, 2026. URL https://arxiv.org/abs/2604.03128.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Lechen Zhang, Jiarui Liu, and Tal August. Re-centering humans in llm personalization. *arXiv preprint arXiv:2606.06614*, 2026.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models, 2024. URL https://arxiv.org/abs/2401.01286.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15686–15702, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.971. URL https://aclanthology.org/2023.emnlp-main.971/.

A EXPERIMENTAL RESULTS ON KGFACT-LARGE AND SIMPLEQA

We present the experimental results on KGFACT-LARGE and SimpleQA in Table 3 and Table 4, respectively.

Table 3: Evaluation on KGFACT-LARGE across the three model backbones. Methods and column semantics follow Table 1. All values are in percent. **Bold** and underline mark the best and second-best method per column (excluding Base).

Model	Method	In-domain test		Held-out capability test					
		Train	KGFACT-RETRIEVAL	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Qwen3-1.7B	Base	0.0	97.3	11.0	72.4	80.4	60.4	58.5	56.5
	SFT	**99.6**	0.0	0.0	1.8	0.4	0.0	0.4	0.5
	OPSD	91.9	1.3	0.0	1.8	0.8	0.6	4.4	1.5
	MIXSD (λ=0)	95.2	12.8	0.0	5.2	3.0	0.0	9.4	3.5
	MIXSD (λ=0.3)	96.5	**14.3**	0.2	13.6	13.2	4.9	**24.6**	11.3
	MIXSD (λ=0.5)	95.6	4.8	**0.6**	**26.8**	**32.1**	**20.7**	23.5	**20.7**
Qwen3-4B-It	Base	0.0	97.9	62.9	94.2	92.6	86.0	77.6	82.6
	SFT	97.1	36.8	5.2	30.0	22.7	81.1	56.9	39.2
	MIXSD (λ=0)	**98.1**	**98.7**	23.5	80.8	88.0	84.8	**65.5**	68.5
	MIXSD (λ=0.3)	94.0	96.4	41.5	84.8	87.2	83.5	53.7	70.1
	MIXSD (λ=0.5)	84.6	97.9	**44.8**	**93.0**	**91.7**	**87.2**	61.0	**75.5**
Qwen3-8B	Base	0.0	99.6	26.0	83.4	91.8	86.6	73.1	72.2
	SFT	98.3	71.4	7.7	34.4	25.4	**83.5**	69.9	44.2
	MIXSD (λ=0)	**98.5**	**94.9**	10.6	69.8	89.2	61.0	67.3	59.6
	MIXSD (λ=0.3)	95.8	84.2	22.5	77.6	**91.4**	79.3	67.4	67.6
	MIXSD (λ=0.5)	93.1	65.3	**24.2**	**82.0**	**91.4**	81.1	**71.0**	**69.9**

Table 4: Evaluation on SimpleQA across the three model backbones. The Train column reports closed-book accuracy on the 4,326-question SimpleQA training split. The Held-out capability test columns report five unrelated benchmarks that probe forgetting; Avg is their unweighted mean. All values are in percent. **Bold** and underline mark the best and second-best method per column (excluding Base).

Model	Method	Train	Held-out capability test					
		SimpleQA	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Qwen3-1.7B	Base	0.5	11.0	72.4	80.4	60.4	58.5	56.5
	SFT	**8.5**	0.0	6.8	3.4	3.0	30.2	8.7
	OPSD	2.9	0.4	19.0	24.3	31.1	4.7	15.9
	OPSD (n=1, T=0)	3.9	0.0	6.4	7.2	8.5	6.9	5.8
	OPSD-NLL (n=1, T=0)	3.4	**1.3**	18.0	18.3	18.9	10.8	13.4
	MIXSD (λ=0.3)	7.1	0.6	29.8	37.1	33.5	20.9	24.4
	MIXSD (λ=0.5)	6.3	1.0	**31.2**	**45.2**	**36.0**	**31.4**	**29.0**
Qwen3-4B-It	Base	3.4	62.9	94.2	92.6	86.0	77.6	82.6
	SFT	**18.8**	0.2	19.2	15.0	50.0	48.8	26.6
	OPSD	7.5	33.8	73.2	77.0	88.4	**67.1**	67.9
	OPSD (n=1, T=0)	8.5	46.0	82.8	83.3	81.7	62.2	71.2
	OPSD-NLL (n=1, T=0)	7.8	32.5	62.2	55.4	79.3	49.6	55.8
	MIXSD (λ=0.3)	10.5	55.4	92.6	90.1	**89.0**	53.9	**76.2**
	MIXSD (λ=0.5)	10.4	**61.9**	**93.2**	**90.5**	84.8	49.3	75.9
Qwen3-8B	Base	2.5	26.0	83.4	91.8	86.6	73.1	72.2
	SFT	**16.8**	0.2	18.8	14.1	67.1	52.0	30.4
	OPSD	8.3	14.2	75.0	80.8	79.3	59.4	61.7
	OPSD (n=1, T=0)	6.8	19.0	80.2	89.3	85.4	62.2	67.2
	OPSD-NLL (n=1, T=0)	6.0	20.6	79.0	89.4	**86.6**	**70.2**	69.2
	MIXSD (λ=0.3)	9.0	21.9	79.4	87.0	81.1	65.0	66.9
	MIXSD (λ=0.5)	9.3	**27.1**	**80.8**	**92.3**	82.9	66.8	**70.0**

B ADDITIONAL RELATED WORK: EVALUATION DATASETS FOR KNOWLEDGE INJECTION

Evaluation for knowledge injection must distinguish genuine acquisition of new information from memorization, annotation artifacts, or noisy supervision. Real-world factuality benchmarks such as SimpleQA provide short, information-seeking questions with objectively verifiable answers (Wei et al.,

2024), making them useful for testing factual recall after fine-tuning, but they do not provide controlled target knowledge to inject. Knowledge editing benchmarks address this limitation by constructing explicit factual updates: ZsRE and CounterFact test whether models can revise individual facts while preserving locality (Levy et al., 2017; De Cao et al., 2021; Meng et al., 2022), while MQuAKE and RippleEdits further evaluate whether such updates propagate to logically related or multi-hop questions (Zhong et al., 2023; Cohen et al., 2024). However, these datasets often rely on real-world facts, knowledge bases, or generated counterfactual variants, so evaluation can still be confounded by pretraining contamination, ambiguous entities, outdated facts, and construction noise. Broader suites such as DUnE and KnowEdit improve coverage across domains and evaluation dimensions, but they do not fully remove the difficulty of validating factual supervision at scale (Akyürek et al., 2023; Zhang et al., 2024). Recent synthetic benchmarks such as PhantomWiki instead generate fictional worlds on demand, allowing reasoning and retrieval to be evaluated against known ground truth rather than memorized or hallucinated facts (Gong et al., 2025). Motivated by this principle, our datasets provide a clean and fully verifiable setting for studying knowledge injection and forgetting without relying on facts that may already exist in the pretrained model.

C Broader Impacts

MixSD is a general-purpose fine-tuning method for injecting knowledge into language models while mitigating catastrophic forgetting. By improving the memorization-retention trade-off, it can potentially lower the data and compute cost of adapting LLMs to specialized domains such as medicine, law, and low-resource languages, making domain adaptation more accessible, and may potentially help preserve safety-relevant behaviors (e.g., instruction following, refusal calibration) that are often degraded by standard fine-tuning. As with any fine-tuning method, it could in principle be used to inject misleading or harmful content into a base model; however, because MixSD requires only a base model and a target corpus and introduces no new misuse vectors beyond those of standard SFT, we consider its marginal misuse risk to be low.

D Additional Experiment Setup Details

We use `https://github.com/NVIDIA-NeMo/RL` for all training experiments. We set the rollout number n to 1 for SFT and MixSD, and to 8 for OPSD unless otherwise specified. For OPSD, we use the forward KL objective and keep the teacher model fixed as the base model without parameter updates. We also evaluate additional OPSD variants by replacing the KL loss with an NLL loss and by varying the number of rollouts per query n and the rollout-generation temperature T. We set the top-k logits parameter to $k = 64$.

For MixSD, we allow up to 100 retries if the generated output is identified as incorrect by our rule-based verifier. If all retries fail, the example is discarded. In practice, we find that this retry budget is sufficient for MixSD to generate correct outputs on both KGFact and KGFunc. For SimpleQA, examples that remain incorrect after all retries are similarly discarded.

We tune the learning rate over $\{1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ and select the best learning rate based on validation performance for each training run. The global batch size is 16, and training is performed in bfloat16 precision. We use a warmup scheduler with 20 warmup steps, followed by a constant learning rate. We set the number of prompts per step to 20. Checkpoints are saved every 20 steps for KGFact and SimpleQA, and every 10 steps for KGFunc.

We select the earliest checkpoint after convergence, defined as the first checkpoint that achieves the best training performance observed so far while maintaining stable validation performance, i.e., without a validation performance drop greater than 5% over the following two consecutive checkpoints.

For KGFact, we train for 20 epochs. For KGFunc, we train for 100 steps. For SimpleQA, we train for one epoch over the full dataset.

For rollout generation, we use VLLM V1. We set the maximum number of new tokens to 8192 and the total sequence length to 10,000. Unless otherwise noted, rollouts are generated with temperature $T = 0$, so token-proposal randomness arises solely from Bernoulli mixing. All experiments are conducted using 4×H100 GPUs, with a total training cost of approximately 2000 GPU hours.

E Additional Baselines for Knowledge Injection

E.1 Parameter-efficient fine-tuning

We compare SFT with all parameters trainable and MixSD with LoRA (Hu et al., 2022), sweeping rank $r \in \{16, 64, 256\}$ and learning rate $\eta \in \{3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$ for each dataset and model. Table 5 reports the best observed LoRA configuration for each setting alongside the SFT and MixSD results. Train denotes accuracy on the injected knowledge and Held-out Avg denotes the unweighted average across AIME-2024, MATH-500, GSM8K, HumanEval, and MMLU.

LoRA improves held-out performance over full-parameter SFT in 10 of 12 settings. MixSD with $\lambda = 0.3$ exceeds LoRA's held-out average in 10 of 12 settings. The exceptions are KGFact-Small with Qwen3-4B-It and KGFact-Large with Qwen3-1.7B. The acquisition results depend on the task. On KGFunc with Qwen3-1.7B, LoRA achieves higher accuracy on the injected knowledge (77.1 versus 45.7), whereas MixSD preserves more held-out capability (42.2 versus 15.4). On SimpleQA, MixSD exceeds LoRA on both measures at all three scales.

Table 5: LoRA comparison across four corpora and three Qwen3 backbones. Each method reports Train and Held-out Avg in percent. Table 8 reports variation across three training seeds.

		SFT		LoRA		MixSD ($\lambda = 0.3$)	
Dataset	Qwen3	Train	Avg	Train	Avg	Train	Avg
KGFact-Small	1.7B	99.0	14.3	100.0	31.4	100.0	39.9
	4B-It	100.0	43.2	100.0	79.1	94.0	74.3
	8B	100.0	46.1	100.0	66.4	99.0	70.0
KGFact-Large	1.7B	99.6	0.5	98.3	21.4	96.5	11.3
	4B-It	97.1	39.2	99.8	45.0	94.0	70.1
	8B	98.3	44.2	98.1	51.6	95.8	67.6
KGFunc	1.7B	51.4	3.2	77.1	15.4	45.7	42.2
	4B-It	72.6	16.6	88.0	12.5	89.1	79.2
	8B	86.9	53.5	90.9	52.2	89.1	66.6
SimpleQA	1.7B	8.5	8.7	2.9	16.0	7.1	24.4
	4B-It	18.8	26.6	7.9	30.7	10.5	76.2
	8B	16.8	30.4	5.8	32.5	9.0	66.9

E.2 Direct preference optimization

We evaluate direct preference optimization (DPO; Rafailov et al., 2023) using the canonical ground-truth SFT response as the chosen completion and the base model's response without the fact in context as the rejected completion. The rejected response is incorrect for 100% of KGFact-Small pairs and 97–99% of KGFunc pairs. The pairs favor the correct target, but their responses can also differ in style, length, and format.

We sweep 12 configurations: $\beta \in \{0.01, 0.05, 0.1, 0.5\}$ and learning rate $\eta \in \{10^{-6}, 3{\times}10^{-6}, 10^{-5}\}$. Across the evaluated checkpoints and all three Qwen3 scales, KGFact-Small training accuracy remains at 0%. The only nonzero acquisition accuracy in the sweep is 2.3% on KGFunc with Qwen3-1.7B, compared with 51.4% under SFT. In this setting the held-out average is 56.3, close to the base model's 56.5.

DPO reaches 100% preference accuracy and its loss approaches zero. Over 200 steps, the chosen implicit reward reaches $+81.96$ and the rejected implicit reward reaches -254.32. We inspect the final checkpoint's predictions for the 100 training facts. All responses follow the required format, but 93 predicted entities never appear in the training corpus, including `Astronomer` and `Nurse`. The remaining seven are corpus entities assigned to the wrong questions. The model therefore learns to separate the preference pairs while failing to answer the training questions correctly.

DPO conditions on the question, but its relative preference objective can be satisfied without making the chosen completion the most probable generated answer. SFT and MixSD directly maximize

the conditional likelihood of target tokens. This finding applies to the fixed preference pairs and hyperparameter configurations evaluated here.

E.3 Scope of the baseline comparison

The $\lambda = 0$ setting tests self-distillation using only expert supervision: the base model generates responses while observing the fact, and the same answer verifier accepts its targets. This control lets us assess token mixing when both methods use the model's own style and expert context. OPSD generates responses from the current student and uses the expert to score them. MIXSD includes expert tokens directly in the generated response, with both branches conditioning on the same prefix. This distinction matters when the student's unconditioned responses rarely express the new fact. Appendix F examines example filtering, and Appendix G examines the token selection and target distribution.

Our setting does not assume access to the original pretraining corpus, so replay of that corpus is unavailable. Replay using a separately collected general-domain corpus remains a possible extension. GRPO with a correctness reward also faces a sparse learning signal when the model generates no correct responses. Additional exploration or supervision could address this difficulty and remains a direction for future comparison.

F Verification and Statistical Robustness

F.1 Retry and discard rates

Table 6 reports the percentage of examples requiring a retry and the percentage discarded after generation attempts. Retry rates vary substantially with the model and increase with λ, but discard rates are zero for KGFACT-SMALL, KGFACT-LARGE, and KGFUNC at both $\lambda = 0.3$ and 0.5. The retention improvements on these corpora therefore occur with every training example retained. Verification can still alter the distribution of accepted sequences even when every example is retained.

SimpleQA has nonzero discard rates, ranging from 1.5% to 6.8%. We test the effect of this filtering below. The reported retry rate measures how often an example needs another attempt; the number of attempts and generated tokens per example remain unmeasured.

Table 6: Generation retry / discard rates (%) across Qwen3 backbones.

Dataset	Qwen3	$\lambda = 0.3$	$\lambda = 0.5$
KGFact-Small	1.7B	75.0 / 0	91.0 / 0
	4B-It	15.0 / 0	39.0 / 0
	8B	70.0 / 0	90.0 / 0
KGFact-Large	1.7B	69.0 / 0	90.2 / 0
	4B-It	10.8 / 0	31.9 / 0
	8B	61.7 / 0	86.0 / 0
KGFunc	1.7B	52.4 / 0	74.9 / 0
	4B-It	2.2 / 0	27.3 / 0
	8B	18.9 / 0	41.9 / 0
SimpleQA	1.7B	61.4 / 4.1	82.7 / 6.8
	4B-It	40.6 / 4.3	62.8 / 6.2
	8B	53.7 / 1.5	75.8 / 2.7

F.2 SimpleQA filtering control

We construct a common filtered SFT training set by retaining only examples that pass verification under all six generation settings (three models and $\lambda \in \{0.3, 0.5\}$). This intersection contains 3,869 of the 4,326 SimpleQA examples. Table 7 compares SFT on this intersection with SFT on all examples, alongside the verified MIXSD results. Filtering lowers SFT's SimpleQA accuracy at every scale, from 8.5 to 7.7, 18.8 to 18.3, and 16.8 to 15.9. Its held-out average improves by 1.8, 1.0, and 0.4 points, respectively, and remains below MIXSD at either mixing rate.

Filtering gives SFT a small retention improvement and lowers acquisition accuracy in this experiment. Its broader effect can depend on how removing examples changes optimization and knowledge coverage. The common SFT subset requires each example to pass all six generation settings.

Table 7: SimpleQA filtering control. All denotes 4,326 examples; Intersection denotes the common 3,869-example subset used for the SFT control; Verified denotes accepted MIXSD supervision. Accuracy and Held-out Avg are percentages.

Qwen3	Method	Training set	SimpleQA	Held-out Avg
1.7B	Base	—	0.5	56.5
	SFT	All	8.5	8.7
	SFT	Intersection	7.7	10.5
	MIXSD ($\lambda = 0.3$)	Verified	7.1	24.4
	MIXSD ($\lambda = 0.5$)	Verified	6.3	29.0
4B-It	Base	—	3.4	82.6
	SFT	All	18.8	26.6
	SFT	Intersection	18.3	27.6
	MIXSD ($\lambda = 0.3$)	Verified	10.5	76.2
	MIXSD ($\lambda = 0.5$)	Verified	10.4	75.9
8B	Base	—	2.5	72.2
	SFT	All	16.8	30.4
	SFT	Intersection	15.9	30.8
	MIXSD ($\lambda = 0.3$)	Verified	9.0	66.9
	MIXSD ($\lambda = 0.5$)	Verified	9.3	70.0

F.3 VARIATION ACROSS TRAINING SEEDS

We repeat SFT and MIXSD with $\lambda \in \{0, 0.3, 0.5\}$ using training seeds 42, 43, and 44 on KGFACT-SMALL and KGFUNC across all three Qwen3 backbones. Table 8 reports the mean and standard deviation over these three runs. We reuse the offline supervision across seeds, so these results measure variation in training with a fixed target corpus. The other dataset tables report individual runs.

MIXSD retains more capability than SFT across all six combinations of dataset and model. At $\lambda = 0.3$, the mean held-out score exceeds SFT by 11.7–59.5 points on KGFUNC and 22.1–32.7 points on KGFACT-SMALL. Acquisition depends on the task. On KGFUNC, increasing λ to 0.5 reduces mean accuracy on the injected knowledge to 17.3, 44.8, and 69.9 at the three scales.

Self-distillation using only expert supervision improves retention over SFT in all six settings. Token mixing generally improves retention further, with some exceptions: on KGFACT-SMALL, Qwen3-4B-It has a lower mean held-out score at $\lambda = 0.3$ than at $\lambda = 0$.

F.4 SCOPE OF THE KNOWLEDGE INJECTION RESULTS

MIXSD achieves nearly perfect recall in several controlled factual settings. Accuracy is lower on SimpleQA, and high mixing rates can reduce acquisition on KGFUNC.

KGFACT tests fictional facts with dense relations and partially overlapping entity names, while KGFUNC uses a common input format for seen and unseen operations to reduce reliance on format cues. These controls isolate acquisition and forgetting, but do not capture the full complexity of domain adaptation. SimpleQA supplies naturally occurring open-domain facts, and MQuAKE supplies conflicting edits and multi-hop evaluation; neither fully covers noisy long documents, partially conflicting evidence, or extended sequences of updates.

G ABLATIONS OF TOKEN MIXING AND TARGET DISTRIBUTIONS

G.1 INFORMED TOKEN ROUTING

We test whether selecting tokens by expert plausibility improves on random mixing. At each position, let p_t^+ and p_t^- denote the expert and naive conditional distributions given the shared mixed prefix,

Table 8: Variation across training seeds on KGFACT-SMALL and KGFUNC. Values are mean ± standard deviation (%) over seeds 42, 43, and 44. Model sizes refer to Qwen3.

Setting	Method	Train	Held-out Avg
KGFact-Small / 1.7B	SFT	100.0 ± 0.0	12.3 ± 1.9
	MIXSD ($\lambda = 0$)	95.0 ± 0.0	21.2 ± 5.2
	MIXSD ($\lambda = 0.3$)	99.0 ± 1.0	34.4 ± 3.8
	MIXSD ($\lambda = 0.5$)	99.3 ± 0.6	41.2 ± 2.1
KGFact-Small / 4B-It	SFT	100.0 ± 0.0	41.9 ± 3.1
	MIXSD ($\lambda = 0$)	99.7 ± 0.6	76.9 ± 1.8
	MIXSD ($\lambda = 0.3$)	96.7 ± 1.5	74.6 ± 1.5
	MIXSD ($\lambda = 0.5$)	88.0 ± 2.0	78.6 ± 0.2
KGFact-Small / 8B	SFT	100.0 ± 0.0	46.3 ± 1.7
	MIXSD ($\lambda = 0$)	100.0 ± 0.0	70.0 ± 1.6
	MIXSD ($\lambda = 0.3$)	99.0 ± 0.0	73.0 ± 0.9
	MIXSD ($\lambda = 0.5$)	98.3 ± 1.5	71.0 ± 0.2
KGFunc / 1.7B	SFT	55.2 ± 6.7	4.8 ± 3.8
	MIXSD ($\lambda = 0$)	51.4 ± 6.2	34.6 ± 1.2
	MIXSD ($\lambda = 0.3$)	45.7 ± 2.6	38.6 ± 1.3
	MIXSD ($\lambda = 0.5$)	17.3 ± 7.3	40.8 ± 1.1
KGFunc / 4B-It	SFT	73.5 ± 3.5	18.8 ± 3.1
	MIXSD ($\lambda = 0$)	83.8 ± 13.8	76.2 ± 0.2
	MIXSD ($\lambda = 0.3$)	85.5 ± 9.6	78.3 ± 0.6
	MIXSD ($\lambda = 0.5$)	44.8 ± 3.8	79.7 ± 0.1
KGFunc / 8B	SFT	79.1 ± 4.6	53.7 ± 2.0
	MIXSD ($\lambda = 0$)	80.2 ± 6.3	62.6 ± 1.9
	MIXSD ($\lambda = 0.3$)	87.4 ± 5.0	65.4 ± 1.2
	MIXSD ($\lambda = 0.5$)	69.9 ± 7.7	66.0 ± 1.3

and let y_t^+ and y_t^- be their proposed tokens. The informed rule accepts the naive proposal only if it belongs to the expert's plausibility set:

$$y_t = \begin{cases} y_t^-, & p_t^+(y_t^-) \geq \alpha \max_v p_t^+(v), \\ y_t^+, & \text{otherwise}, \end{cases} \qquad \alpha \in (0, 1). \tag{9}$$

The acceptance test follows the plausibility constraint in contrastive decoding (Li et al., 2023). Larger α rejects more naive proposals. The rule selects expert tokens where the naive proposal conflicts with the injected fact. Verification, NLL training, and the remaining training recipe follow the $\lambda = 0.5$ MIXSD configuration.

The two conditionals propose the same token at 72–80% of positions in the $\lambda = 0.5$ supervision, so branch selection changes the output only at the remaining positions. We calibrate α on a held-out pilot set separately for each model to approximately match the expert deferral rate on disagreement positions. Table 9 gives the thresholds and measured rates. The largest difference in deferral rates is 0.066 for Qwen3-1.7B.

Table 9: Routing calibration. Deferral is the fraction of disagreement positions assigned to the expert branch.

Qwen3	α	Informed deferral	MIXSD ($\lambda = 0.5$)
1.7B	0.001	0.502	0.568
4B-It	0.1	0.553	0.491
8B	0.03	0.543	0.527

Table 10 shows that informed routing increases accuracy on the injected knowledge relative to Bernoulli mixing at $\lambda = 0.5$ at all three scales. Its held-out average is lower by 13.0 points for Qwen3-1.7B and 3.2 points for Qwen3-4B-It, whereas the 8B results are nearly identical (70.9 versus 71.0). The rule therefore improves acquisition at the cost of retention on the two smaller models.

Table 10: Informed routing on KGFACT-SMALL. Entries with uncertainty are means ± standard deviations over three training seeds. The 8B informed routing result comes from one run. All scores are percentages.

Qwen3	Method	Train	Held-out Avg
1.7B	SFT	100.0 ± 0.0	12.3 ± 1.9
	MIXSD ($\lambda = 0$)	95.0 ± 0.0	21.2 ± 5.2
	MIXSD ($\lambda = 0.3$)	99.0 ± 1.0	34.4 ± 3.8
	MIXSD ($\lambda = 0.5$)	99.3 ± 0.6	41.2 ± 2.1
	Informed routing	100.0 ± 0.0	28.2 ± 2.6
4B-It	SFT	100.0 ± 0.0	41.9 ± 3.1
	MIXSD ($\lambda = 0$)	99.7 ± 0.6	76.9 ± 1.8
	MIXSD ($\lambda = 0.3$)	96.7 ± 1.5	74.6 ± 1.5
	MIXSD ($\lambda = 0.5$)	88.0 ± 2.0	78.6 ± 0.2
	Informed routing	90.3 ± 1.2	75.4 ± 3.1
8B	SFT	100.0 ± 0.0	46.3 ± 1.7
	MIXSD ($\lambda = 0$)	100.0 ± 0.0	70.0 ± 1.6
	MIXSD ($\lambda = 0.3$)	99.0 ± 0.0	73.0 ± 0.9
	MIXSD ($\lambda = 0.5$)	98.3 ± 1.5	71.0 ± 0.2
	Informed routing	100.0	70.9

G.2 HARD SAMPLING VERSUS SOFT POOLED TARGETS

At a fixed shared prefix, independently choosing the naive branch with probability λ and the expert branch otherwise gives the marginal proposal distribution

$$q_{\lambda,t}(v) = (1 - \lambda)p_t^+(v) + \lambda p_t^-(v). \tag{10}$$

Here p_t^+ and p_t^- denote the distributions actually used for token proposals; for greedy decoding they are point masses at the respective argmax tokens. Since the branch choice is independent at each step, this equality extends autoregressively: drawing a hard token from the linear pool produces the same distribution over sequences before verification as Bernoulli branch selection. Applying the same sequence verifier preserves this equivalence between the two hard-sampling procedures.

We also test training on soft targets. We use the examples, token sequences, and training settings from the $\lambda = 0.5$ run and replace NLL on sampled tokens with forward KL from a pooled target distribution to the student. We evaluate both a linear pool and a normalized log-linear pool of the expert and naive predictive distributions:

$$q_t^{\text{lin}}(v) = (1 - \lambda)p_t^+(v) + \lambda p_t^-(v), \tag{11}$$

$$q_t^{\log}(v) = \frac{p_t^+(v)^{1-\lambda} p_t^-(v)^{\lambda}}{\sum_u p_t^+(u)^{1-\lambda} p_t^-(u)^{\lambda}}. \tag{12}$$

For these soft losses, p_t^+ and p_t^- retain the predictive probabilities. Log-linear pooling interpolates the two sets of logits.

Both soft pools substantially weaken knowledge acquisition: their training accuracy ranges from 10.0% to 29.0%, compared with 88.0–99.3% for training on verified token sequences. Their held-out averages differ from those of MIXSD by at most 5.1 points.

Verification changes the distribution of accepted sequences and may explain the acquisition gap. Let $V(y) = 1$ indicate that a sequence passes verification. The accepted distribution is

$$q_\lambda(y \mid x, V = 1) \;\propto\; q_\lambda(y \mid x)\, \mathbf{1}\{V(y) = 1\}. \tag{13}$$

At a fixed prefix, the probability of an accepted next token depends on both its proposal probability and the chance that the completed sequence will pass verification. This favors correct continuations at positions that determine the answer. The soft pools used here omit that weighting and can retain probability mass on the naive model's incorrect answer. This difference is a possible explanation for the observed acquisition gap.

Table 11: Ablations using soft targets on KGFACT-SMALL. MIXSD reports means ± standard deviations over three seeds; soft-target results are point estimates. Scores are percentages.

Qwen3	Method	Train	Held-out Avg
1.7B	MIXSD ($\lambda = 0.5$)	99.3 ± 0.6	41.2 ± 2.1
	Soft target, linear pool	10.0	40.7
	Soft target, log-linear pool	29.0	43.2
4B-It	MIXSD ($\lambda = 0.5$)	88.0 ± 2.0	78.6 ± 0.2
	Soft target, linear pool	19.0	79.9
	Soft target, log-linear pool	28.0	79.7
8B	MIXSD ($\lambda = 0.5$)	98.3 ± 1.5	71.0 ± 0.2
	Soft target, linear pool	19.0	65.9
	Soft target, log-linear pool	27.0	68.7

H SELECTING THE MIXING RATE, CONVERGENCE, AND COMPUTE

H.1 SELECTING THE MIXING RATE

We select the largest mixing rate that maintains sufficient accuracy on the injected knowledge. A local gradient analysis motivates this rule. Let $h_t = (x, y_{<t})$ be the context used in the training loss and let $p_0(\cdot \mid h_t) = p_{\theta^\star}(\cdot \mid h_t)$ be the base model's naive conditional. Under exact sampling from this conditional, the score identity gives

$$\mathbb{E}_{y_t \sim p_0(\cdot \mid h_t)} \left[\nabla_\theta \log p_\theta(y_t \mid h_t) \big|_{\theta=\theta^\star} \right] = 0. \tag{14}$$

For an individual token, write $z_\theta(h_t)$ for the logit vector and $J_t = \partial z_\theta(h_t)/\partial\theta$ for its Jacobian at the base parameters. The NLL gradient is $J_t^\top(p_0 - e_{y_t})$, giving

$$\left\| \nabla_\theta[-\log p_\theta(y_t \mid h_t)] \big|_{\theta=\theta^\star} \right\|_2 \le \sqrt{2}\, \|J_t\|_2\, [1 - p_0(y_t \mid h_t)]. \tag{15}$$

The inequality follows from $\|p_0 - e_{y_t}\|_2 \le \sqrt{2}(1 - p_0(y_t \mid h_t))$. Thus, under greedy or truncated decoding, naive tokens have small gradient contributions when the base model assigns them high probability and the Jacobian norm is controlled; their expected contribution need not vanish exactly.

Ignoring changes in the prefix distribution and treating the naive contribution as small gives the local approximation

$$\Delta\theta_\lambda \approx -\eta(1-\lambda) g_+, \tag{16}$$

where g_+ is the expert contribution evaluated near the base parameters. If the base model is approximately stationary for its reference loss and that loss has local Hessian H_{ref}, then

$$\mathcal{F}(\lambda) \approx \tfrac{1}{2} \Delta\theta_\lambda^\top H_{\mathrm{ref}} \Delta\theta_\lambda \approx \frac{\eta^2}{2} (1-\lambda)^2 g_+^\top H_{\mathrm{ref}} g_+. \tag{17}$$

This motivates increasing λ to reduce capability loss, subject to maintaining acquisition of the injected knowledge. This approximation describes behavior near the base parameters. Over training, adaptive optimization, changes in the generated prefixes, and verification can alter the gradient. Conditioning on successful verification can also invalidate the zero expected score for accepted naive tokens. The analysis therefore provides a heuristic for selecting λ.

For a candidate grid Λ, let $A(\lambda)$ measure accuracy on the injected knowledge and let $\epsilon \in [0, 1)$ specify the acceptable relative loss in that accuracy. We choose

$$\lambda^\star = \max \left\{ \lambda \in \Lambda : A(\lambda) \ge (1-\epsilon) \max_{\lambda' \in \Lambda} A(\lambda') \right\}. \tag{18}$$

This rule requires only accuracy measurements on the injected knowledge. When $\lambda = 0$ is a sufficiently accurate reference, we can first measure $A(0)$ and then increase λ, retaining candidates satisfying $A(\lambda) \ge (1-\epsilon)A(0)$. Evaluating the full grid is preferable when accuracy varies non-monotonically with λ or substantially exceeds $A(0)$ at another setting. The tolerance should reflect the application's acceptable acquisition loss.

H.2 CONVERGENCE ON KGFACT-SMALL

We evaluate the saved checkpoints from the KGFACT-SMALL runs and compare the optimizer steps at which each method converges under the checkpoint criterion in Appendix D. Table 12 reports these steps alongside the corresponding acquisition scores. Each method converges to its own attained accuracy, shown in parentheses.

Table 12: Convergence steps on KGFACT-SMALL, with corresponding acquisition accuracy (%) in parentheses. Model sizes refer to Qwen3.

Model	SFT	$\lambda = 0$	$\lambda = 0.3$	$\lambda = 0.5$
1.7B	40 (100.0)	40 (95.0)	60 (99.0)	60 (99.3)
4B-It	40 (100.0)	80 (99.7)	100 (96.7)	120 (88.0)
8B	40 (100.0)	60 (100.0)	60 (99.0)	80 (98.3)

Supervision using only expert tokens requires 1.0–2.0 times as many steps as SFT, while the mixed settings require 1.5–3.0 times as many. The convergence step is nondecreasing in λ for each model. This ordering is consistent with a weaker effective acquisition signal as more expert tokens are replaced by naive tokens, as suggested by the local gradient analysis.

H.3 OFFLINE GENERATION AND TRAINING COST

MIXSD uses the same standard NLL training objective and training implementation as SFT; it adds offline generation with verification and retries. A generated corpus for a fixed model and mixing configuration can be reused across training seeds and checkpoint evaluations. On KGFACT-SMALL, generating one training set costs 0.5 GPU-hours on average, compared with 5.2 GPU-hours for one run of fine-tuning on that set. When R training runs share this corpus, generation adds $0.5/R$ GPU-hours per run, approximately $9.6\%/R$ of the reported fine-tuning cost.

Each model and mixing configuration requires its own generated corpus. Total training cost also depends on target lengths, padding, and the steps needed to converge. These factors matter when interpreting the optimizer step comparison above.

OPSD generates fresh student responses during every training run, using eight rollouts per prompt in our default setting. Its generation cost therefore grows with the number of runs. Our reported GPU-hours measure compute use; a complete wall-time comparison would also require timing generation and training and counting generated and accepted tokens.

I ADDITIONAL EXPERIMENT RESULTS

I.1 FISHER INFORMATION: ESTIMATION AND DIAGNOSTICS

This section describes the Fisher approximation, estimation procedure, and diagnostic quantities used in Section 7.

Estimation procedure We estimate $\hat{F}$ at the base model $\theta^\star$ using a reference corpus $\mathcal{D}_{\text{ref}}$ of N=979 examples randomly drawn from five general-domain benchmarks (AIME-2024, MATH-500, GSM8K, HumanEval, and MMLU).

For each example $(\mathbf{x}_n, \mathbf{y}_n)$:

1. Compute the sum-reduced log-likelihood over target tokens (prompt tokens masked).
2. Backpropagate to obtain the per-sample gradient $\mathbf{g}_n$.
3. Accumulate $\mathbf{g}_n^2$ into the Fisher estimate.

We use batch size 1 to obtain per-sample gradients, and cast gradients to fp32 before squaring to avoid underflow. After processing all samples, we average over N.

Diagnostic quantities Given a fine-tuned checkpoint θ_B with displacement $\Delta\theta = \theta_B - \theta^\star$, we compute:

Raw parameter displacement:

$$\|\Delta\theta\|^2 = \sum_i (\Delta\theta_i)^2. \tag{19}$$

Fisher-weighted displacement:

$$Q_F = \Delta\theta^\top \hat{F} \Delta\theta = \sum_i \hat{F}_i (\Delta\theta_i)^2. \tag{20}$$

Q_F measures how strongly the update moves along sensitive directions of the base model.

Fisher alignment ratio:

$$R = \frac{Q_F / \|\Delta\theta\|^2}{\mathrm{tr}(\hat{F})/d}. \tag{21}$$

R compares curvature along the update direction to the expected curvature under an isotropic direction. Values $R{>}1$ indicate concentration in high-curvature directions, while $R{<}1$ indicates avoidance.

Parameter displacement across methods and model sizes Table 13 reports the raw parameter displacement $\|\Delta\theta\|^2$ and Fisher alignment ratio R for each method, model size, and task cohort.

Table 13: Raw parameter displacement $\|\Delta\theta\|^2$, Fisher alignment ratio R, and mean capability drop (Drop: average accuracy decrease across AIME-2024, MATH-500, GSM8K, HumanEval, and MMLU relative to the base model) at converged checkpoints for each method, model size, and task cohort. Methods with similar displacement can produce dramatically different forgetting (Drop column), confirming that magnitude alone does not predict capability loss.

		KGFact-Small			KGFunc			SimpleQA		
Model	**Method**	$\|\Delta\theta\|^2$	R	Drop	$\|\Delta\theta\|^2$	R	Drop	$\|\Delta\theta\|^2$	R	Drop
	SFT	85.0	0.90	42.3	58.0	1.10	53.3	469.9	0.77	47.9
	OPSD-fwdKL	1752.4	0.49	51.5	62.5	0.58	13.8	2332.1	0.43	40.6
	MixSD (λ=0.3)	215.2	0.67	16.7	69.4	0.78	14.3	430.3	0.65	32.1
	MixSD (λ=0.5)	221.1	0.65	16.3	65.0	0.75	10.9	424.7	0.67	27.6
	SFT	28.2	0.99	39.4	16.6	1.14	66.0	96.7	1.36	56.0
	OPSD-fwdKL	346.5	0.81	22.0	23.6	0.74	5.0	485.5	0.64	14.8
	MixSD (λ=0.3)	63.5	0.70	8.4	20.9	0.86	3.4	82.2	0.83	6.5
	MixSD (λ=0.5)	68.5	0.65	5.1	18.7	0.81	3.5	81.8	0.82	6.7
	SFT	61.9	1.16	26.0	53.9	1.25	18.7	187.6	2.14	41.8
	OPSD-fwdKL	577.7	0.91	10.4	44.3	0.99	6.9	1109.6	0.96	10.5
	MixSD (λ=0.3)	88.6	1.00	2.2	42.6	1.39	5.6	159.3	1.35	5.3
	MixSD (λ=0.5)	86.5	0.92	−0.2	40.5	1.33	5.6	161.5	1.48	2.2

Table 14 reports the Pearson correlation between each metric ($\|\Delta\theta\|^2$ and R) and mean forgetting (average drop across five held-out benchmarks), pooling all checkpoints across the three task cohorts.

Table 14: Pearson r between each metric and forgetting.

Model	n	$r(\|\Delta\theta\|^2, \text{Drop})$	$r(R, \text{Drop})$
Qwen3-1.7B	42	+0.34	+0.56
Qwen3-4B-It	45	+0.02	+0.82
Qwen3-8B	40	+0.10	+0.57

Table 15: Count and percentage of tokens with NLL > 5 under the base model, aggregated over all examples. At this threshold the base model assigns probability $< e^{-5} \approx 0.7\%$ to the correct token.

Dataset	Model	SFT	λ=0.0	λ=0.3	λ=0.5
KGFACT-SMALL	Qwen3-1.7B	914 (37.1%)	983 (17.1%)	1149 (10.5%)	1171 (7.7%)
	Qwen3-4B-It	684 (33.2%)	1621 (7.4%)	2349 (5.9%)	4013 (4.6%)
	Qwen3-8B	1144 (46.5%)	1191 (15.4%)	1296 (7.3%)	1341 (4.7%)
KGFACT-LARGE	Qwen3-1.7B	868 (36.1%)	1008 (13.6%)	1170 (9.1%)	1118 (6.5%)
	Qwen3-4B-It	623 (31.1%)	1599 (6.4%)	2229 (5.2%)	3544 (4.3%)
	Qwen3-8B	1092 (45.4%)	1140 (11.3%)	1298 (6.3%)	1244 (4.2%)
SimpleQA	Qwen3-1.7B	458 (30.3%)	545 (12.1%)	526 (13.5%)	518 (12.5%)
	Qwen3-4B-It	287 (25.8%)	723 (4.4%)	685 (4.6%)	678 (3.5%)
	Qwen3-8B	520 (34.4%)	619 (4.1%)	630 (5.7%)	589 (6.2%)
KGFUNC	Qwen3-1.7B	1344 (11.8%)	1104 (1.3%)	1173 (0.7%)	974 (0.4%)
	Qwen3-4B-It	812 (7.4%)	1361 (0.8%)	1906 (0.8%)	1922 (0.5%)
	Qwen3-8B	1207 (10.6%)	792 (0.9%)	777 (0.6%)	777 (0.4%)

Table 16: Count and percentage of tokens with NLL > 8 under the base model, aggregated over all examples. At this threshold the base model assigns probability $< e^{-8} \approx 0.034\%$ to the correct token, indicating near-complete failure of prediction.

Dataset	Model	SFT	λ=0.0	λ=0.3	λ=0.5
KGFACT-SMALL	Qwen3-1.7B	818 (33.2%)	758 (13.2%)	853 (7.8%)	864 (5.7%)
	Qwen3-4B-It	572 (27.7%)	1075 (4.9%)	1492 (3.8%)	2339 (2.7%)
	Qwen3-8B	1027 (41.7%)	913 (11.8%)	959 (5.4%)	1000 (3.5%)
KGFACT-LARGE	Qwen3-1.7B	770 (32.0%)	741 (10.0%)	850 (6.6%)	808 (4.7%)
	Qwen3-4B-It	544 (27.1%)	1032 (4.1%)	1382 (3.2%)	2090 (2.5%)
	Qwen3-8B	979 (40.7%)	850 (8.4%)	921 (4.5%)	901 (3.0%)
SimpleQA	Qwen3-1.7B	370 (24.5%)	440 (9.8%)	431 (11.1%)	418 (10.1%)
	Qwen3-4B-It	221 (19.9%)	469 (2.9%)	452 (3.0%)	448 (2.3%)
	Qwen3-8B	443 (29.3%)	480 (3.1%)	496 (4.5%)	479 (5.0%)
KGFUNC	Qwen3-1.7B	1162 (10.2%)	711 (0.8%)	744 (0.4%)	598 (0.3%)
	Qwen3-4B-It	560 (5.1%)	718 (0.4%)	1041 (0.4%)	1101 (0.3%)
	Qwen3-8B	843 (7.4%)	535 (0.6%)	553 (0.5%)	547 (0.3%)

I.2 HIGH-NLL TOKEN COUNTS

Since NLL $= -\log p$, a token with NLL $= \tau$ receives predicted probability $p = e^{-\tau}$ from the base model. At $\tau = 5$ this corresponds to $p \approx 0.7\%$—the model is highly uncertain but retains weak signal—while at $\tau = 8$ the probability drops to $p \approx 0.034\%$, indicating near-complete failure of prediction (equivalent to uniform guessing among $\sim$3,000 alternatives). We report counts at both thresholds: Table 15 captures tokens where the base model is very uncertain, and Table 16 captures tokens where it has essentially no predictive signal. The relative ordering across training configurations is consistent at both thresholds, confirming that the findings are not sensitive to the specific choice of τ.

A high-NLL token is considered a memorization target if it corresponds to factual content, such as a named entity, numerical value, date, or domain-specific term that is likely absent from the base model's pretraining distribution. To identify such tokens, we first extract high-NLL tokens from the SFT GT targets, then measure their overlap with the corresponding MIXSD targets for the same example.

Table 17: Recall of SFT high-NLL token types within each corresponding MIXSD target sequence. Recall is defined as $|\text{MIXSD} \cap \text{SFT}| \,/\, |\text{SFT}|$, where the SFT set consists of unique subword token types with NLL > 8 under the base model. High recall indicates that MIXSD preserves the same factual memorization targets that are difficult for the base model in standard SFT.

Dataset	Model	λ=0	λ=0.3	λ=0.5
KGFACT-SMALL	Qwen3-1.7B	86%	96%	96%
	Qwen3-4B-It	93%	97%	98%
	Qwen3-8B	89%	88%	89%
KGFACT-LARGE	Qwen3-1.7B	77%	84%	81%
	Qwen3-4B-It	85%	86%	89%
	Qwen3-8B	82%	83%	81%
SimpleQA	Qwen3-1.7B	87%	95%	95%
	Qwen3-4B-It	64%	72%	72%
	Qwen3-8B	71%	79%	85%
KGFUNC	Qwen3-1.7B	62%	66%	53%
	Qwen3-4B-It	58%	76%	76%
	Qwen3-8B	49%	62%	57%

Specifically, for each example, we collect the set of high-NLL token *types* (i.e., unique subword identities) appearing in the SFT GT sequence, and compute the fraction of those token types that also appear in the corresponding MIXSD target sequence, regardless of absolute position. The 81–98% overlap statistic reported in the main text is computed using this procedure.

Because the underlying factual content is largely preserved across training targets, with only the surrounding phrasing changing, these high-NLL factual tokens consistently reappear across different training configurations.

I.3 ERROR ANALYSIS: PER-METHOD BREAKDOWN

This appendix provides the full error distributions underlying the analysis in Section 7. We classify every incorrect response on five realistic benchmarks (AIME-2024, MATH-500, GSM8K, HumanEval, and MMLU) into one of four mutually exclusive categories. Results are reported for the base model, SFT, OPSD, and MIXSD with $\lambda \in \{0.3, 0.5, 0.7\}$ across the three Qwen3 backbones, after fine-tuning on either KGFACT-SMALL or KGFUNC.

Error categories. Each incorrect response is assigned a category in priority order:

`format` > `leakage` > `collapse` > `genuine`.

- **`format`**: the response cannot be parsed by the benchmark evaluator. For math benchmarks and MMLU, this means no extractable `\boxed{}` answer; for HumanEval, no fenced Python code block.
- **`leakage`**: the response contains a fictional KGFACT entity (a multi-word phrase drawn from the fine-tuning corpus) unrelated to the prompt, indicating interference from injected knowledge. This category is disabled for KGFUNC, whose targets contain only integers and operation labels.
- **`collapse`**: the response contains a parseable answer but little or no reasoning, typically a short template such as `The answer is X.\boxed{X}` inherited from SFT targets. For HumanEval, this corresponds to a boxed stub answer without a Python implementation.
- **`genuine`**: none of the above. The response has the expected structure (e.g., chain-of-thought or executable code) but arrives at an incorrect answer. These resemble the base model's ordinary reasoning errors.

Per-method observations. The main trends are summarized in Section 7; here we provide additional detail across methods and datasets.

- **Base.** Errors are overwhelmingly `genuine`: the model attempts the task with coherent reasoning or executable code, but reaches the wrong answer. This serves as the reference error distribution that MIXSD largely preserves.
- **SFT on KGFACT-SMALL.** SFT introduces two dominant failure modes: `collapse` and `leakage`. On Qwen3-1.7B / MMLU, 59.9% of all responses fall into `collapse` and another 4.3% into `leakage` (e.g., emitting "*Ormavel Valley*" in response to an arithmetic question). On AIME-2024, the collapse rate reaches 96.0%. On HumanEval, the same behavior appears as boxed stub answers without code, occurring in 72.6% of generations.
- **SFT on KGFUNC.** Only the `collapse` mode remains, since KGFUNC contains no fictional entities. Moreover, collapse is substantially weaker than on KGFACT-SMALL, likely because KGFUNC supervision includes explicit step-by-step reasoning rather than short answer templates.
- **OPSD.** OPSD largely eliminates `collapse`, but increases `format` errors. For example, on Qwen3-1.7B / KGFACT-SMALL MMLU, 4.3% of responses are unparsable. This is consistent with on-policy KL training occasionally destabilizing instruction-following behavior.
- **MIXSD.** Across all mixing rates, both `collapse` and `leakage` remain near zero across benchmarks and backbones. The primary remaining failure mode is `genuine` reasoning error, closely matching the base model. At higher mixing rates (λ=0.7), we observe a small increase in `format` errors, typically due to excessively long chain-of-thought generations that fail to terminate with a valid `\boxed{}` answer.

Figures 4 to 13 give the per-method error counts on each of the five benchmarks for both training corpora.

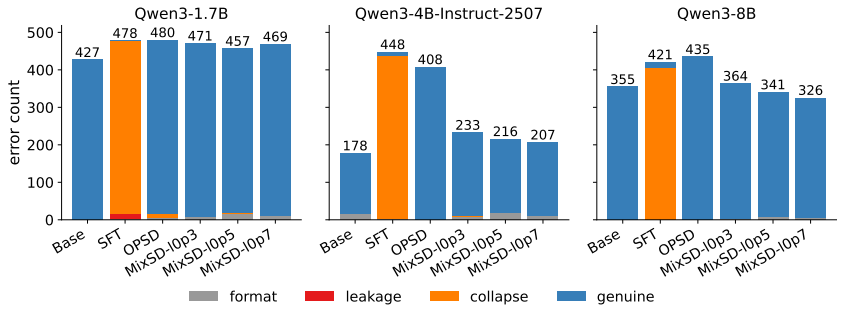

Figure 4: Error-mode breakdown on AIME-2024 after fine-tuning on KGFACT-SMALL.

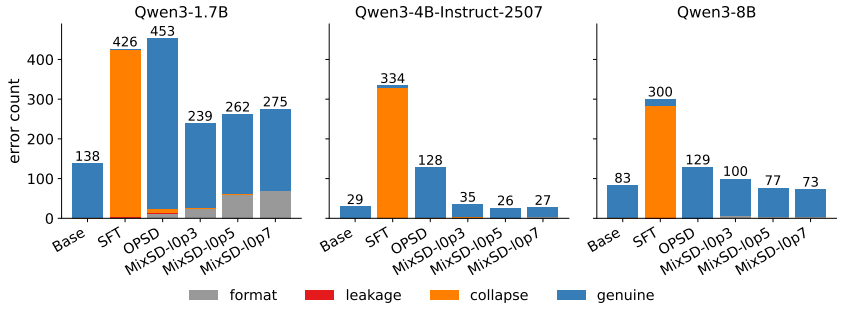

Figure 5: Error-mode breakdown on MATH-500 after fine-tuning on KGFACT-SMALL.

I.4 LLAMA-3.2-1B-INSTRUCT ON KGFACT-SMALL

To verify that our findings generalize beyond the Qwen model family, we train the same set of methods on Llama-3.2-1B-Instruct using KGFACT-SMALL. Table 18 reports results under the same evaluation protocol as Table 1.

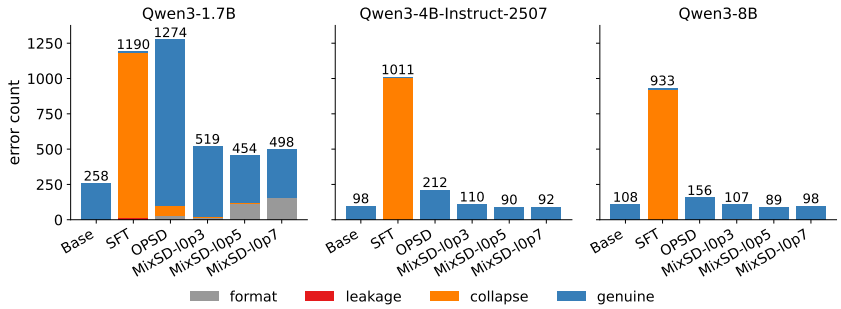

Figure 6: Error-mode breakdown on GSM8K after fine-tuning on KGFACT-SMALL.

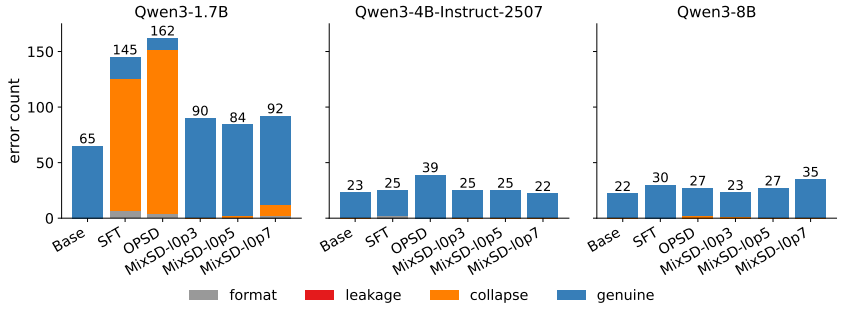

Figure 7: Error-mode breakdown on HumanEval after fine-tuning on KGFACT-SMALL. The `format` bucket here uses the HumanEval-specific check (presence of a fenced Python code block) rather than the `\boxed{}` check used on the math/MMLU benchmarks; `collapse` flags responses that emit `\boxed{X}` stubs in lieu of a function definition.

The pattern is consistent with the Qwen results: SFT memorizes the training set (98% accuracy) but catastrophically forgets held-out capabilities, dropping average performance from 6.8 to 1.4. MIXSD at λ=0.3 and λ=0.5 matches SFT's training accuracy while retaining far more general capability (average held-out scores of 4.8 and 5.3 vs. 1.4 for SFT). OPSD, which does not use the mixing mechanism, achieves only 31% training accuracy, mirroring the Qwen-scale finding that pure on-policy sampling underperforms the mixed variant. These results confirm that the forgetting-mitigation benefit of MIXSD is architecture-independent.

I.5 KNOWLEDGE EDITING ON MQUAKE

While our main experiments inject novel knowledge absent from the base model's parameters, MQuAKE (Zhong et al., 2023) studies the complementary setting of *knowledge editing*, where the target facts revise knowledge the model already encodes. The model must therefore overwrite existing parametric knowledge and produce the updated answer rather than the original one.

We fine-tune all three Qwen3 backbones (1.7B, 4B, and 8B) on the MQuAKE training set using both standard SFT and MIXSD with $\lambda \in \{0, 0.3, 0.5\}$. We additionally compare against MEMIT (Meng et al., 2023), a locate-and-edit method specifically designed for factual model editing. We apply MEMIT to all 100 edited facts simultaneously using the EasyEdit framework (Wang et al., 2024). Following the default MEMIT configuration for Qwen-style architectures, edits are applied to the MLP down-projection matrices (`down_proj`) across contiguous early-to-mid transformer layers: layers 3–11 for Qwen3-1.7B (9 of 28 layers), layers 4–14 for Qwen3-4B-It (11 of 36 layers), and layers 5–14 for Qwen3-8B (10 of 36 layers). The value vectors are optimized for 25 gradient steps with learning rate 0.5, weight decay 10^{-3}, and KL factor 0.0625. Second-moment statistics used for covariance adjustment are estimated from 100,000 WikiText samples in fp32, with update weight $\mu = 15{,}000$ and clamp norm factor 4. Table 19 reports both editing accuracy and held-out capability retention.

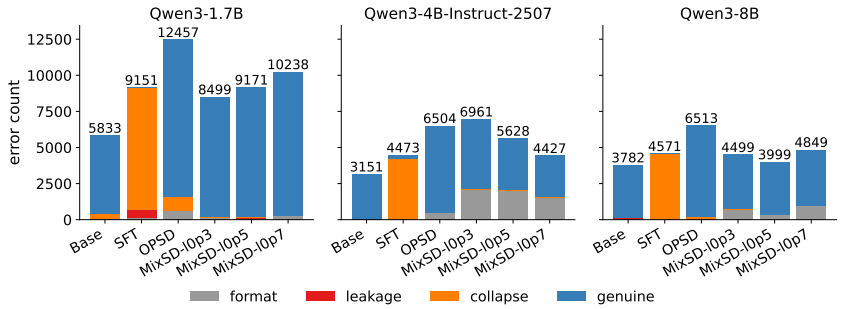

Figure 8: Error-mode breakdown on MMLU after fine-tuning on KGFACT-SMALL. The `leakage` bar isolates the cases where the fine-tuned model answers an MMLU question with a fictional KGFACT-SMALL entity.

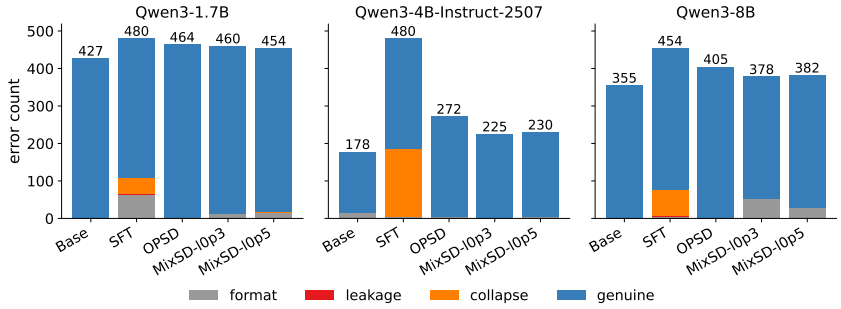

Figure 9: Error-mode breakdown on AIME-2024 after fine-tuning on KGFUNC. `leakage` is disabled by construction (training answers are integers and operation labels).

The results closely mirror the knowledge-injection setting. SFT achieves perfect editing accuracy at all scales, but severely degrades held-out capabilities, reducing average performance by 45–86% relative to the base model. In contrast, MIXSD at λ=0.3 achieves near-perfect editing accuracy (93–99%) while preserving substantially more of the model's original capabilities, reaching held-out averages of 17.6, 76.0, and 70.0 across the three scales.

MEMIT exhibits the opposite trade-off: it preserves held-out capabilities almost perfectly (within roughly 1–2 points of the base model), but achieves substantially lower editing accuracy (53–70%) than MIXSD. This is consistent with prior observations that rank-one editing methods can interfere destructively when entities participate in many correlated knowledge triples.

Notably, retrieval-augmented evaluation with chain-of-thought reasoning is consistently much higher for MIXSD than for SFT across all model scales (84–95% vs. 41–58%). This suggests that MIXSD better preserves the model's ability to integrate edited knowledge into downstream reasoning processes rather than merely memorizing revised answers.

I.6 KL-BASED DISTILLATION ABLATIONS ON KGFACT-SMALL

Table 20 compares KL-based variants across the three Qwen models. OPSD denotes on-policy distillation with a forward-KL objective using n=8 student rollouts at temperature T=1. OPSD-NLL is the analogous on-policy variant that replaces KL with token-level NLL on the teacher's top-1 prediction, using a single greedy rollout (n=1, T=0). The three MIXSD-KL (λ=0) variants replace the per-token NLL term on $\tilde{y}_i^+$ in Eq. (5) with a top-64 forward-KL objective against the teacher under different student rollout settings. We additionally include MIXSD (λ=0) as the NLL counterpart at the same mixing ratio, enabling a direct comparison between KL- and NLL-based teacher matching in the absence of GT mixing.

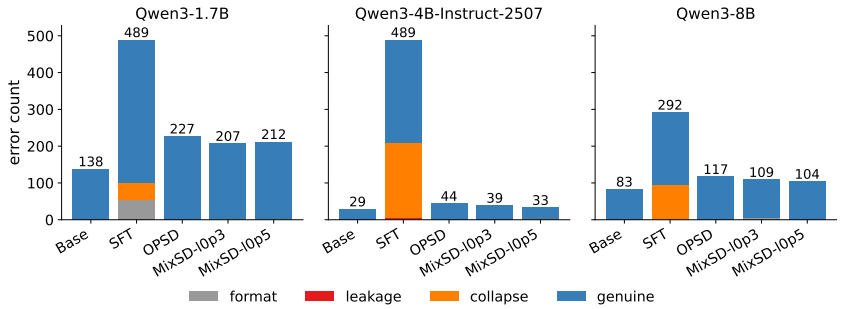

Figure 10: Error-mode breakdown on MATH-500 after fine-tuning on KGFUNC.

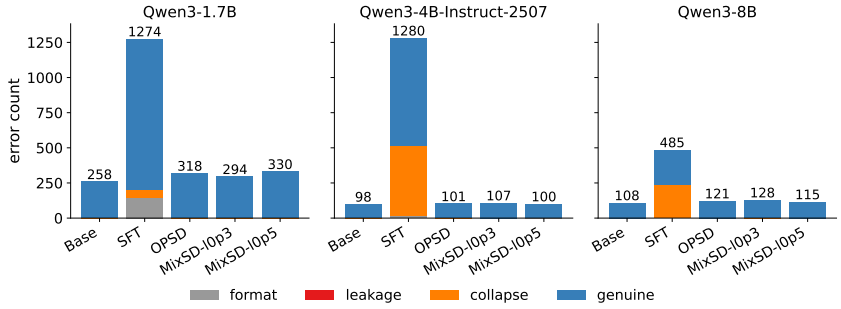

Figure 11: Error-mode breakdown on GSM8K after fine-tuning on KGFUNC.

We adopt NLL as the default objective in our framework because it enables substantially more effective knowledge acquisition, achieving 96–100% training accuracy with only a single greedy rollout. In contrast, MIXSD-KL consistently underfits the target knowledge, reaching only 34–60% training accuracy despite achieving comparable or slightly stronger held-out capability preservation. Since MIXSD already mitigates forgetting through GT mixing ($\lambda>0$), we prioritize efficient knowledge absorption over additional preservation gains from KL-based distillation. Although OPSD achieves high training accuracy with a KL objective, it requires $n=8$ student rollouts per query, resulting in an approximately $8\times$ higher per-example compute cost than offline methods.

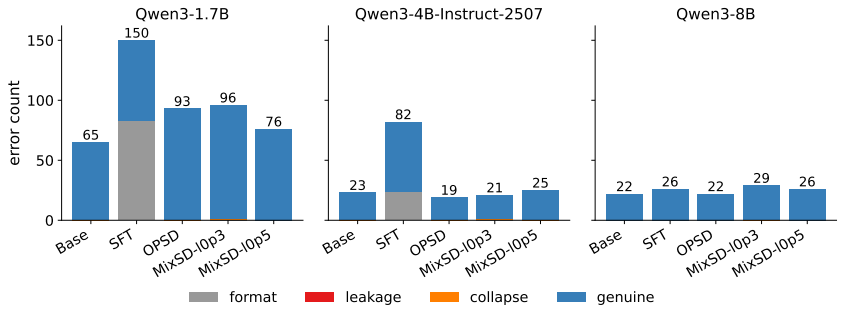

Figure 12: Error-mode breakdown on HumanEval after fine-tuning on KGFUNC.

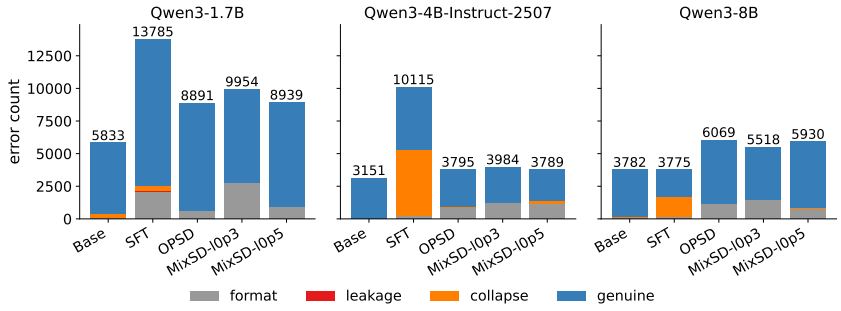

Figure 13: Error-mode breakdown on MMLU after fine-tuning on KGFUNC.

Table 18: Evaluation on KGFACT-SMALL for Llama-3.2-1B-Instruct. Column semantics match Table 1.

Model	Method	In-domain test		Held-out capability test					
		Train	KGFACT-RETRIEVAL	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Llama-3.2-1B-It	Base	0.0	11.0	1.9	5.8	1.7	24.4	0.3	6.8
	SFT	98.0	0.0	0.0	4.4	2.7	0.0	0.0	1.4
	OPSD	31.0	10.0	0.0	3.8	6.1	9.1	7.8	5.4
	MIXSD (λ=0)	28.0	16.0	0.0	0.4	0.8	0.0	0.1	0.3
	MIXSD (λ=0.3)	98.0	16.0	0.0	5.0	5.6	0.6	12.6	4.8
	MIXSD (λ=0.5)	98.0	15.0	0.2	3.2	4.5	0.6	18.1	5.3
	MIXSD (λ=0.7)	60.0	4.0	0.0	2.8	5.1	1.2	1.3	2.1

Table 19: Knowledge editing on MQuAKE across the three Qwen3 backbones. Train is closed-book recall on edited facts and In-domain retrieval provides retrieval context with chain-of-thought. The five Held-out capability test columns probe forgetting on unrelated benchmarks; Avg is their unweighted mean. All values are in percent.

Model	Method	In-domain test		Held-out capability test					
		Train	Retrieval	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Qwen3-1.7B	Base	10.0	100.0	11.0	72.4	80.4	60.4	58.5	56.5
	MEMIT	60.0	98.0	10.6	70.6	81.1	56.1	57.1	55.1
	SFT	100.0	53.0	0.4	11.8	7.4	9.2	10.3	7.8
	MIXSD (λ=0)	78.0	67.0	0.2	12.8	18.7	9.8	23.4	13.0
	MIXSD (λ=0.3)	97.0	84.0	0.0	18.6	22.6	21.3	25.4	17.6
	MIXSD (λ=0.5)	95.0	76.0	1.0	36.6	50.3	42.1	33.0	32.6
Qwen3-4B-It	Base	16.0	94.0	62.9	94.2	92.6	86.0	77.6	82.7
	MEMIT	53.0	90.0	63.5	94.4	92.1	85.4	72.8	81.6
	SFT	100.0	41.0	0.8	24.4	22.1	67.7	61.3	35.3
	MIXSD (λ=0)	81.0	92.0	41.7	87.4	90.4	85.4	55.3	72.0
	MIXSD (λ=0.3)	97.0	95.0	50.4	92.8	92.3	88.4	56.0	76.0
	MIXSD (λ=0.5)	69.0	88.0	47.9	93.0	92.3	87.8	51.8	74.6
Qwen3-8B	Base	19.0	94.0	26.0	83.4	91.8	86.6	73.1	72.2
	MEMIT	70.0	89.0	24.6	83.6	91.6	85.4	72.1	71.4
	SFT	100.0	58.0	0.8	26.6	24.2	83.5	62.0	39.4
	MIXSD (λ=0)	78.0	73.0	20.4	79.8	91.2	85.4	69.9	69.3
	MIXSD (λ=0.3)	99.0	86.0	24.6	81.2	91.9	81.7	70.6	70.0
	MIXSD (λ=0.5)	90.0	86.0	29.0	83.2	93.2	82.9	60.4	69.7

Table 20: KL-based distillation ablations on KGFACT-SMALL (three Qwen backbones). Column semantics match Table 1. For MIXSD-KL (λ=0), n is the number of student rollouts per prompt and T the sampling temperature. MIXSD (λ=0) is the NLL-on-teacher counterpart at the same λ.

Model	Method	In-domain test		Held-out capability test					
		Train	KGFACT-RETRIEVAL	AIME-2024	MATH-500	GSM8K	HumanEval	MMLU	Avg
Qwen3-1.7B	Base	0.0	100.0	11.0	72.4	80.4	60.4	58.5	56.5
	OPSD	99.0	32.0	0.0	9.4	3.4	1.2	11.3	5.1
	OPSD-NLL	52.0	32.0	0.6	25.6	41.9	28.7	20.9	23.6
	MIXSD-KL (λ=0, n=8, T=1)	60.0	11.0	0.0	0.0	0.2	0.0	0.0	0.0
	MIXSD-KL (λ=0, n=1, T=0)	54.0	57.0	0.0	8.4	6.1	0.0	23.1	7.5
	MIXSD-KL (λ=0, n=1, T=1)	38.0	59.0	0.0	6.6	3.9	0.0	10.3	4.2
	MIXSD (λ=0)	96.0	28.0	0.0	5.4	5.8	1.2	30.4	8.6
Qwen3-4B-It	Base	0.0	84.0	62.9	94.2	92.6	86.0	77.6	82.6
	OPSD	100.0	92.0	15.0	74.4	83.9	76.2	53.7	60.6
	OPSD-NLL	96.0	60.0	8.3	53.8	70.8	86.0	49.2	53.6
	MIXSD-KL (λ=0, n=8, T=1)	50.0	97.0	12.5	68.0	80.0	72.6	53.5	57.3
	MIXSD-KL (λ=0, n=1, T=0)	34.0	98.0	35.8	89.2	92.8	85.4	70.7	74.8
	MIXSD-KL (λ=0, n=1, T=1)	34.0	100.0	48.3	93.2	92.3	87.8	71.9	78.7
	MIXSD (λ=0)	99.0	97.0	44.4	90.8	91.5	86.6	68.8	76.4
Qwen3-8B	Base	0.0	98.0	26.0	83.4	91.8	86.6	73.1	72.2
	OPSD	99.0	100.0	9.4	74.2	88.2	83.5	53.6	61.8
	OPSD-NLL	59.0	62.0	8.3	67.6	86.9	70.7	52.5	57.2
	MIXSD-KL (λ=0, n=8, T=1)	61.0	96.0	8.3	65.2	86.8	51.2	58.5	54.0
	MIXSD-KL (λ=0, n=1, T=0)	34.0	99.0	11.2	75.2	89.2	80.5	66.4	64.5
	MIXSD-KL (λ=0, n=1, T=1)	37.0	99.0	18.3	80.2	91.2	82.9	69.7	68.5
	MIXSD (λ=0)	100.0	100.0	17.5	73.6	91.8	82.9	67.8	66.7