SAVE: Sparsity-Aware Influence Estimation for Vocabulary-Expanded LLMs

Seungyoo Lee
KAIST
punctuate@kaist.ac.kr

Giung Nam
KAIST
giung@kaist.ac.kr

Seanie Lee
Apple
seanie_lee@apple.com

Hyungi Lee[†]
Kookmin University
lhk2708@kookmin.ac.kr

Juho Lee[†]
KAIST
juholee@kaist.ac.kr

Abstract

Large language models adapted to specialized targets such as Lean theorem proving or tool calling often benefit from extending the tokenizer with target-specific tokens. However, target-only vocabulary-expanded fine-tuning can leave the newly added embeddings transfer-incomplete, since the new rows receive gradients only from examples containing the corresponding tokens, so target-local gains do not necessarily imply that the embeddings are integrated with related capabilities. We propose SAVE, a sparsity-aware influence estimator for selecting auxiliary healing data from generic corpora. SAVE computes token-conditional curvature over the effective update set of each new vocabulary row and combines this vocabulary-aware signal with dense LoRA-level influence. Across Lean autoformalization and tool calling, SAVE improves target performance and related transfer while remaining competitive on broad retention benchmarks, and its gains carry over to a smaller backbone and a less syntactically rigid medical target. Code will be made available at https://github.com/DopeorNope-Lee/SAVE.

1 Introduction

Modern large language models (LLMs) are increasingly adapted to specialized targets that require strict syntactic structure, such as Lean theorem proving [53, 54, 41, 48, 33, 34, 7, 50, 55, 4] and tool calling [17, 49, 11, 30]. In these settings, important units such as formal identifiers, operators, schema fields, and tool names are often fragmented by general-purpose tokenizers, increasing generation length and obscuring task-specific structure. Vocabulary expansion addresses this mismatch by adding target-specific tokens, and has been used in cross-lingual adaptation [12, 25, 45, 56, 38], domain specialization [8, 37, 3, 19, 31, 13], and tool-use modeling [17, 49, 11, 30].

However, adding new tokens does not by itself ensure that their embeddings are fully learned. Newly added embedding rows are only approximately initialized, and each row receives gradients only from examples containing the corresponding token. Thus, target-only vocabulary-expanded fine-tuning can make the new tokens useful on the narrow target distribution while leaving them insufficiently integrated with related capabilities. We observe this failure mode in Lean autoformalization: vocabulary expansion improves the target miniF2F [61] benchmark, but underperforms standard fine-tuning on AIME-style mathematical reasoning, a closely related setting where positive transfer should be expected. We refer to this failure mode as *transfer-incomplete vocabulary expansion*.

Additional auxiliary data can help complete these embeddings, but the choice of data is critical. In our preliminary study, mathematical auxiliary data improve the vocabulary-expanded Lean model,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

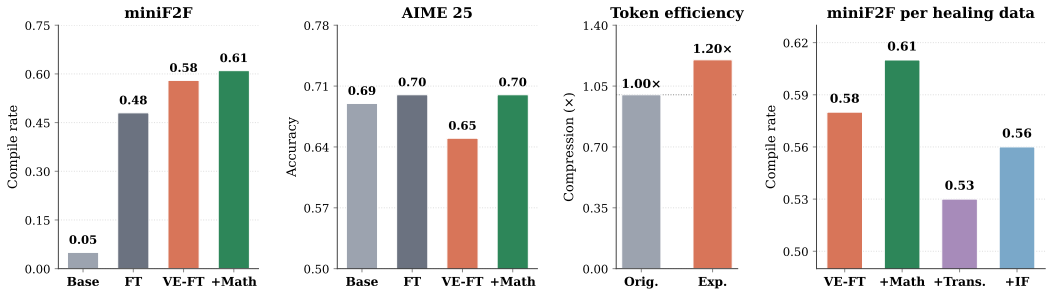

Figure 1: **Transfer-incomplete vocabulary expansion and vocabulary healing.** VE-FT improves the target miniF2F benchmark and token efficiency, but hurts related AIME-style mathematical reasoning. Math auxiliary data recovers this related-transfer drop and further improves miniF2F, while less related auxiliary data hurts target performance.

whereas less-related instruction-following or translation data can hurt target performance. This suggests that the useful signal is not generic replay, but target-compatible examples that both support the target task and activate the expanded vocabulary in useful ways. We call this post-expansion adaptation stage *vocabulary healing*. Since the right healing data may not be obvious for each new target, we aim to select it automatically from a generic auxiliary corpus.

We propose **SAVE** (**S**parsity-**A**ware **V**ocabulary **E**stimator), an influence-based data selection method for vocabulary-expanded LLMs. SAVE exploits the row-sparse gradient structure of new embedding rows. Unlike existing gradient-based selectors [26, 24, 52, 27, 23, 10, 59, 35, 9], which normalize curvature over the full candidate pool, SAVE estimates token-conditional curvature using only examples that contain each expanded token. It then combines this vocabulary-aware signal with a dense LoRA-level influence signal, capturing both target-vocabulary alignment and broader content-level transfer. SAVE is a targeted, plug-in correction for a specific failure of existing influence estimators under vocabulary expansion: its vocabulary-aware term can be added to other gradient-based selectors, such as LESS and OGS, without modifying them. Experiments on Lean autoformalization and tool calling show that SAVE improves target performance and related transfer over target-only training, naive auxiliary mixing, and existing selection baselines. Budget-matched controls show that these gains are not explained by semantic or lexical similarity retrieval or by the selected domain mixture alone, and the gains persist on a smaller backbone and on a medical question-answering target. Analysis shows that selected healing data are target-dependent, favoring mathematical and symbolic examples for autoformalization but schema-compatible examples for tool use.

We make the following contributions:

- We identify *transfer-incomplete vocabulary expansion*, where target-only vocabulary-expanded fine-tuning improves the target task but hurts transfer to related tasks.
- We propose SAVE, a sparsity-aware influence estimator that accounts for the row-sparse gradients of newly added embedding tokens during auxiliary data selection, as a plug-in correction that also improves LESS, OGS, and DataInf.
- We validate SAVE on Lean autoformalization and tool calling, showing improved target performance together with related transfer while remaining competitive on broad retention benchmarks. Attribution controls, a smaller backbone, and a medical target further support that the gains stem from token-conditional curvature and instance-level ranking.

2 Problem Settings and Empirical Observations

In this paper, we aim to adapt an instruction-tuned base model M_{base} for target tasks by jointly updating its weights and vocabulary. In this scenario, unlike simple fine-tuning, which maintains a fixed vocabulary V_{base}, we represent task-specific terms as dedicated tokens to enhance both inference efficiency and performance [12, 8, 17].

2.1 Vocabulary-Expanded Fine-Tuning

Our investigation begins with a comparison between *standard fine-tuning (FT)* and *vocabulary-expanded fine-tuning (VE-FT)*. The former is the most foundational baseline, performing supervised fine-tuning on the target dataset D_{trn} with the original vocabulary, while the latter represents the most intuitive method for incorporating vocabulary expansion into the task-specific fine-tuning process.

We construct V_{exp} by selecting frequent multi-token sequences in D_{trn} that are fragmented by V_{base}, thereby enabling task-specific concepts to be represented as atomic tokens rather than as compositions of semantically mismatched base tokens, e.g., "Math"+"lib" $\rightarrow$ "Mathlib" in auto-formalization tasks [46]. Following the *exponential embedding initialization* strategy of Nakash et al. [37], we initialize the embeddings for these new tokens. Specifically, for each $v \in V_{\text{exp}}$ composed of base tokens $(u_i \in V_{\text{base}})_{i=1}^{k}$, its embedding $\text{Emb}(v)$ is defined as a weighted sum with $\beta = 2$:

$$\text{Emb}(v) = \sum_{i=1}^{k} \left[\frac{\exp(\beta i)}{\sum_{j=1}^{k} \exp(\beta j)} \text{Emb}(u_i) \right].$$

Empirical results. We conduct a diagnostic comparison between standard fine-tuning and vocabulary-expanded fine-tuning. Starting from the same instruction-tuned base model, we obtain M_{FT} by fine-tuning on the autoformalization training set D_{trn} with the original vocabulary V_{base}, and obtain $M_{\text{VE-FT}}$ by fine-tuning on the same data after expanding the tokenizer to V_{exp}. We evaluate target performance on miniF2F [61], related mathematical transfer on AIME-25 [60], and inference efficiency by measuring the token compression obtained from the expanded vocabulary.

Figure 1 summarizes the diagnostic results. VE-FT improves miniF2F performance and reduces tokenization cost, confirming that the expanded tokens are useful within the target distribution. However, $M_{\text{VE-FT}}$ underperforms M_{FT} on AIME-25, a related mathematical reasoning benchmark where positive transfer from autoformalization training should be expected. Adding mathematical auxiliary data recovers this related-transfer drop and further improves miniF2F, whereas less related auxiliary data hurts target performance. This pattern suggests that target-only VE-FT can produce expanded embeddings that are locally useful for the target benchmark but not yet transfer-ready across broader mathematical contexts.

2.2 Auxiliary Data Helps Only When Target-Compatible

Building on these observations, we hypothesize that the expanded vocabulary embeddings require further training, which can be effectively achieved by incorporating auxiliary datasets alongside the original target data. Crucially, to fully optimize these embeddings, the selected auxiliary datasets must be closely related to the target task. As a preliminary study to test this hypothesis, we manually curate several auxiliary datasets based on their perceived semantic relevance. We then conduct an exploratory analysis by jointly training the model on these curated sets and the primary target dataset.

Specifically, our target task is autoformalization, which requires mathematical reasoning and symbolic structure to translate informal statements into formal proofs. We therefore train with the mathematics subset of `Nemotron-Post-Training-Dataset-v1` [1] as a related auxiliary dataset, and compare it against less related datasets: `Nemotron-SFT-Instruction-Following-Chat-v2` [2], a general-purpose instruction-following chat dataset, and `ACADATA` [28], a parallel corpus for academic machine translation. This investigation aims to determine whether the inclusion of related, or even unrelated, data during vocabulary-expanded fine-tuning can stabilize the representation of the expanded vocabulary and ultimately lead to more robust task specialization.

Empirical results. Our empirical analysis reveals a clear distinction in generalization performance based on the nature of the auxiliary data. As shown in fig. 1, we observe that the inclusion of semantically related datasets significantly outperforms the use of unrelated data in preserving the model's broader capabilities. Specifically, unrelated data provides limited benefit, whereas related auxiliary data improves both target performance and related transfer, suggesting that it helps align the new tokens with target semantics and the model's pre-existing knowledge.

However, manual curation does not scale: identifying related auxiliary data for every new target requires human effort and domain expertise, and a well-matched auxiliary dataset may simply not exist for niche or emerging targets. We therefore seek the healing signal in large general-purpose

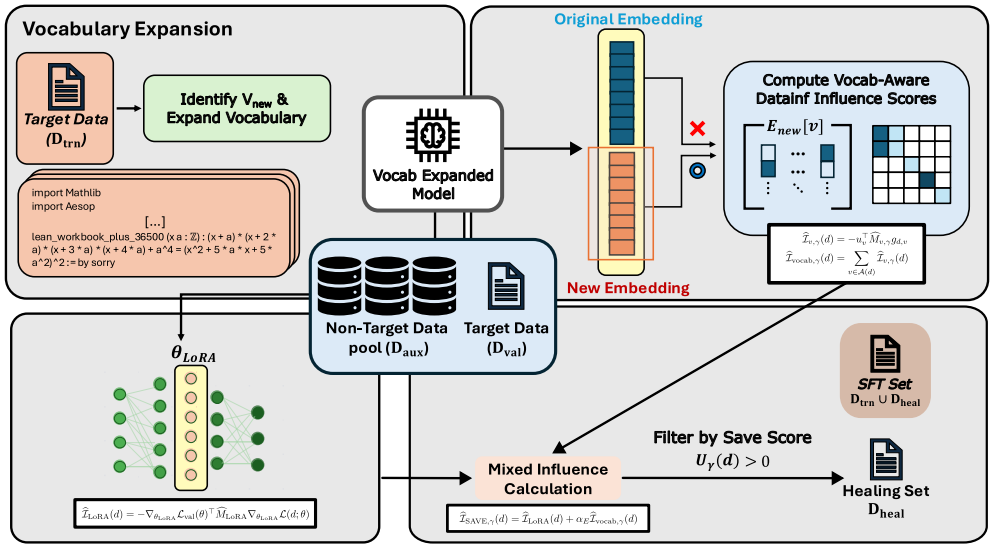

Figure 2: **Overview of SAVE.** Given target training data D_{trn}, we expand the vocabulary with frequent target-domain tokens and initialize new embedding rows E_{new}. Using the vocabulary-expanded model as a surrogate, SAVE estimates the utility of each auxiliary example $d \in D_{\text{aux}}$ against the target validation set D_{val}. The score combines a dense LoRA-level DataInf signal with a token-conditional vocab-aware signal over E_{new}. Positive-utility examples form D_{heal}, combined with target data into the final SFT pool.

corpora, which contain diverse patterns that are latently relevant to a given target, and develop an automatic mechanism to retrieve them without manual intervention.

3 Sparsity-Aware Vocabulary Estimator

Based on the observations in Section 2, we propose the **Sparsity-Aware Vocabulary Estimator (SAVE)**, a data selection framework that retrieves target-compatible auxiliary examples from general corpora to facilitate vocabulary-expanded fine-tuning. SAVE comprises three core components: **(§3.1)** a *vocabulary-aware influence score* that focuses on newly expanded embedding rows while accounting for their row-sparse update patterns; **(§3.2)** a *mixed influence score* that combines token-level signals from expanded embeddings with global semantic signals captured by LoRA weights, ensuring dual alignment at both the lexical and content levels; and **(3)** a *healing-set selection* procedure that utilizes these scores to construct an auxiliary data mixture for the final supervised fine-tuning stage.

3.1 Vocabulary-Aware Influence Score

Section 2 suggests that healing hinges on identifying which examples in a general-purpose corpus support the target task. To this end, we employ influence scores, rooted in classical influence functions [15, 16] and widely used for training-data attribution [26, 27].

Let D_{aux} be a general-purpose auxiliary corpus, and D_{val} be a held-out validation set from the target task. For the vocabulary-expanded surrogate model M_{init} with learnable parameters θ, we define the target validation loss and the corresponding influence score as:

$$\mathcal{L}_{\text{val}}(\theta) = \frac{1}{|D_{\text{val}}|} \sum_{z' \in D_{\text{val}}} \mathcal{L}(z'; \theta), \qquad \mathcal{I}(d) = -\nabla_\theta \mathcal{L}_{\text{val}}(\theta)^\top H_\theta^{-1} \nabla_\theta \mathcal{L}(d; \theta),$$

where $\mathcal{L}(z; \theta)$ is the per-example loss (e.g., cross-entropy for next-token prediction) and H_θ is the Hessian (or its proxy) of the training objective [27]. For any candidate $d \in D_{\text{aux}}$, a negative influence score indicates that including d in the training data is estimated to reduce the target validation loss.

For autoregressive LMs, this score aggregates token-level loss gradients that are dominated by tokens from the original vocabulary, so the gradient signal needed to update the newly added embedding rows is easily neglected.

To isolate this vocabulary-specific effect, we restrict influence computation to the embeddings of newly added tokens. Let $V_{\text{new}} = V_{\text{exp}} \setminus V_{\text{base}}$ be the set of expanded tokens and $E_{\text{new}} = \{\text{Emb}(v) : v \in V_{\text{new}}\}$ be their corresponding input embedding rows. For each $v \in V_{\text{new}}$, we define:

$$g_{d,v} = \nabla_{\text{Emb}(v)} \mathcal{L}(d; \theta), \qquad u_v = \nabla_{\text{Emb}(v)} \mathcal{L}_{\text{val}}(\theta).$$

Since $\text{Emb}(v)$ only receives gradients from examples in which v actually occurs, its update is inherently row-sparse.

Notation. Throughout the paper, we use

$$S_v = \{d \in D_{\text{aux}} : g_{d,v} \neq 0\}, \qquad n_v = |S_v|, \qquad p_v = n_v/|D_{\text{aux}}| \in (0, 1],$$

i.e., S_v is the *effective update set* of examples containing token v, n_v is its size (a count), and p_v is the fraction of the auxiliary corpus that activates v. Standard full-corpus normalization scales the token-specific curvature by p_v, causing rare expanded tokens to be overwhelmed by the damping term. To mitigate this sparsity-induced suppression, we estimate the curvature exclusively over the effective update set and define the *token-conditional preconditioner*:

$$\widehat{M}_{v,\gamma} = \frac{1}{n_v} \sum_{d \in S_v} \left(p_v^{\gamma} g_{d,v} g_{d,v}^{\top} + \lambda_v I \right)^{-1}, \qquad \gamma \in [0, 1],$$

where $\lambda_v > 0$ is a damping parameter. The average runs over the n_v examples in S_v only, so $\widehat{M}_{v,\gamma}$ does not grow with token frequency: a frequent token improves the support of the estimate but does not by itself increase per-example utility (Appendix A.5). In this formulation, $\gamma = 0$ represents a fully token-conditional normalization, while $\gamma = 1$ recovers the full-corpus scaling at the Hessian-proxy level. Using this preconditioner, the contribution of an example d to the influence of token v is computed as:

$$\widehat{\mathcal{I}}_{v,\gamma}(d) = -u_v^{\top} \widehat{M}_{v,\gamma} g_{d,v}.$$

Finally, let $\mathcal{A}(d)$ be the set of expanded tokens appearing in d. We define the *vocabulary-aware influence score* by aggregating these token-level influences:

$$\widehat{\mathcal{I}}_{\text{vocab},\gamma}(d) = \sum_{v \in \mathcal{A}(d)} \widehat{\mathcal{I}}_{v,\gamma}(d).$$

For examples containing no expanded tokens ($\mathcal{A}(d) = \emptyset$), the score naturally vanishes ($\widehat{\mathcal{I}}_{\text{vocab},\gamma}(d) = 0$). This score quantifies whether the expanded tokens within an auxiliary example provide a gradient direction that aligns with the target validation objective, thereby facilitating the stable update of new embedding rows.

3.2 Mixed Influence Score

The vocabulary-aware influence score in section 3.1 measures the impact of auxiliary examples on the newly added token embeddings. However, to capture broader content-level transfer beyond direct lexical alignment, it is essential to consider the influence on the entire model parameters as well. To estimate this in a parameter-efficient manner for the VE-FT model, we follow the established practice of using LoRA parameters as a proxy for influence computation [27].

Specifically, we equip the VE-FT model with LoRA adapters with parameters θ_{LoRA} and perform a short-term adaptation on the target training set D_{trn}. It enables the efficient approximation of influence signals that would otherwise require computationally prohibitive full-parameter gradients. With a slight abuse of notation, we treat θ_{LoRA} as a surrogate for the model's non-embedding weights and define the LoRA-level influence as:

$$\widehat{\mathcal{I}}_{\text{LoRA}}(d) = -\nabla_{\theta_{\text{LoRA}}} \mathcal{L}_{\text{val}}(\theta)^{\top} \widehat{M}_{\text{LoRA}} \nabla_{\theta_{\text{LoRA}}} \mathcal{L}(d; \theta),$$

where $\widehat{M}_{\text{LoRA}}$ is the standard preconditioner estimated over D_{aux}, following Kwon et al. [27]. Finally, we combine the LoRA-level influence with the vocab-aware score. The resulting SAVE score is:

$$\widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d) = \widehat{\mathcal{I}}_{\text{LoRA}}(d) + \alpha_E \widehat{\mathcal{I}}_{\text{vocab},\gamma}(d),$$

where $\alpha_E \geq 0$ controls the relative scale of the expanded-vocabulary signal. Following the influence convention, $\widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d) < 0$ indicates that adding d is estimated to reduce the target validation loss. Thus, the mixed score captures both content-level transfer through LoRA and target-token alignment through the expanded vocabulary.

3.3 Healing with Selected Data

Given the mixed influence score derived in Section 3.2, we now describe the *healing-set selection* procedure: selecting auxiliary examples to *heal* the vocabulary-expanded model. The goal is to retain only examples that are estimated to improve the target validation objective, rather than mixing the target data with arbitrary general-purpose data.

Following the convention [26, 27] where a negative score signifies a reduction in validation loss, we define the selection utility as $U_\gamma(d) = -\widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d)$. Under this definition, higher utility values indicate examples estimated to be more beneficial for the target task. We first filter the auxiliary corpus to keep only positive-utility examples, $D_{\text{pos}} = \{d \in D_{\text{aux}} : U_\gamma(d) > 0\}$, ensuring every candidate is predicted to improve target performance. Given a healing-to-target budget ratio ρ, we set

$$B_\rho = \min\left(\lfloor \rho |D_{\text{trn}}| \rfloor, |D_{\text{pos}}|\right).$$

The final healing set D_{heal} is then constructed by selecting the top B_ρ examples from D_{pos} ranked by their utility U_γ. After curating D_{heal}, the model undergoes a final adaptation phase on the union of the target training data and the selected healing data: $D_{\text{SFT}} = D_{\text{trn}} \cup D_{\text{heal}}$. Initialized with the vocabulary-expanded model M_{init}, we perform standard supervised fine-tuning (SFT). Instead of explicit loss-level reweighting, we control the relative contribution of target and auxiliary healing examples through mini-batch composition. Specifically, we employ a balanced sampling strategy (e.g., an equal split of 8 examples each from D_{trn} and D_{heal} per batch), ensuring the fine-tuning stage remains a standard SFT process while effectively managing the influence of each data source.

4 Experiments

4.1 Experimental Setup

Target settings. We evaluate proposed method on two vocabulary-sensitive target settings: auto-formalization and tool calling. For auto-formalization, we use `Qwen3-8B` [57] as the base model and construct the target data from `Goedel-Pset-v1` [33], randomly sampling 50K examples. For tool calling, we use the ToolWeaver end-to-end data [11] as the target task and adopt the `Llama3-8B` [14] backbone used in the ToolWeaver baseline protocol. All comparisons are backbone-matched within each target setting.

Non-target pool. We construct the non-target auxiliary instruction pool D_{aux} from `Nemotron-Post-Training-Dataset-v1` [1]. We sample 20K examples from each of the `chat`, `code`, `math`, and `stem` categories. For the main auto-formalization experiments, we additionally include 20K ToolWeaver end-to-end examples as non-target tool-use data, yielding a 100K-example candidate pool. We exclude the Nemotron tool calling category to ensure $D_{\text{aux}} \cap D_{\text{trn}} = \emptyset$. Detailed descriptions of the experiments are provided in Appendix C.

4.2 Main Experiments

Table 1 reports target performance (miniF2F), related mathematical transfer (AIME2025), and broad retention (MMLU-Pro, LiveCodeBench$^{\text{v1-5}}$, IF-Eval). The first three rows are diagnostic references: the original model M_{base}; M_{FFT}, full fine-tuning (FFT) on D_{trn} with the original vocabulary; and the vocabulary-expanded model before final SFT M_{init}. The remaining rows start from M_{init}, except the fixed-vocabulary control TARGET+DATAINF (ORIG. VOCAB), which starts from M_{FFT}.

Among final SFT configurations, SAVE achieves the strongest target performance, substantially improving both miniF2F compile rate and accuracy over Target-only, Random, All-Inst, and existing selection baselines. The gap over Random shows that the gain is not explained by simply adding the same amount of auxiliary data, and the gap over DATAINF [27] shows that target-aware influence alone is insufficient when newly added embedding rows receive sparse token-conditional gradients.

SAVE also achieves the best AIME2025 score, so its target gain comes with improved related transfer rather than target-only overfitting. On broad retention, SAVE is not uniformly best, reflecting its focus on target-compatible healing data, but remains competitive. Re-running DataInf and SAVE with three training seeds (fixed healing sets and hyperparameters), SAVE outperforms DataInf in every matched run on all three metrics, by 7.51 and 9.02 points on average in miniF2F compile rate and accuracy and 1.40 points on AIME2025 (Appendix C.11).

Table 1: **Main autoformalization results.** We report target performance on miniF2F, related mathematical transfer on AIME2025, and broad retention on MMLU-Pro, LiveCodeBench, and IF-Eval. The broad-retention average excludes AIME2025. Diagnostic rows show intermediate model stages. Final rows compare SFT configurations; unless marked as original-vocabulary, they start from M_{init} after vocabulary expansion. All budgeted selection methods use the same budget as SAVE, while All-Inst uses the full non-target pool. SAVE is the only configuration that is best on both the target (miniF2F) and related transfer (AIME2025); three-seed results for DataInf and SAVE are in Appendix C.11.

	miniF2F		**Related**	**Broad Retention**			
Setting	Compile	Accuracy	AIME2025	MMLU-Pro	LCB$^{\text{v1-5}}$	IF-Eval	Avg.
M_{base}	0.0533	0.0000	0.6901	0.7296	0.5718	0.8597	0.7204
M_{FFT}	0.6762	0.4918	0.7089	0.7024	0.4995	0.8013	0.6677
M_{init}	0.4221	0.3237	0.6213	0.6903	0.4713	0.7544	0.6387
Target-only	0.6803	0.5861	0.7104	0.7057	0.5123	0.7819	0.6666
Target + Random	0.6516	0.5205	0.7068	0.7093	0.5188	0.8272	0.6851
Target + All-Inst	0.7049	0.6434	0.7130	0.7198	0.5220	0.8341	0.6920
Target + LESS [52]	0.6967	0.5984	0.7198	0.7121	0.5192	0.8049	0.6787
Target + OGS [59]	0.7008	0.6107	0.7172	**0.7273**	0.5238	0.8314	0.6942
Target + DataInf [27]	0.7172	0.6311	0.7203	0.7115	0.5343	0.8252	0.6903
Target + DataInf (Orig. Vocab)	0.7090	0.6270	0.7104	0.7148	0.5327	**0.8375**	**0.6950**
Target + SAVE (ours)	**0.7909**	**0.7213**	**0.7328**	0.7129	**0.5468**	0.8194	0.6930

Table 2: **Attribution controls** (autoformalization, budget-matched). "Mix" marks selectors constrained to SAVE's per-domain composition.

		miniF2F		**Related**
Selector	**Mix**	Compile	Acc.	AIME25
Random	✗	0.6516	0.5205	0.7068
DataInf	✗	0.7172	0.6311	0.7203
EmbSim	✗	0.7131	0.6311	0.7193
New-token TF-IDF	✗	0.6885	0.6107	0.7151
SAVE-Mix Random	✓	0.6762	0.5574	0.7146
SAVE-Mix DataInf	✓	0.7336	0.6557	0.7229
SAVE-FO	✗	0.7459	0.6844	0.7260
SAVE (ours)	induced	**0.7909**	**0.7213**	**0.7328**

Table 3: **Generalization** to a smaller backbone (autoformalization) and a medical target (MedQA).

Qwen3-4B	Compile	Acc.	AIME25
Target-only	0.5861	0.5041	0.6276
DataInf	0.6230	0.5287	0.6385
SAVE (ours)	**0.6721**	**0.5820**	**0.6547**

Llama-3.1-8B-Inst. / MedQA	Accuracy
Base model	0.6135
Target-only	0.7203
DataInf	0.7502
SAVE (ours)	**0.7926**

4.3 Where Do the Gains Come From?

Attribution controls. To separate the effect of token-conditional curvature from simpler explanations, we compare budget-matched selectors under the autoformalization setup (Table 2). Two cheap heuristics isolate similarity signals: EmbSim ranks auxiliary examples by cosine similarity to the target-validation centroid using `all-mpnet-base-v2` embeddings [40, 44], and New-token TF-IDF [42] scores rarity-weighted lexical overlap on the expanded tokens. Three controls ablate SAVE itself: SAVE-Mix Random and SAVE-Mix DataInf keep SAVE's per-domain composition but rank instances randomly or by DataInf, and SAVE-FO replaces the token-conditional preconditioner with $\lambda_v^{-1} I$. Both heuristics beat Random and EmbSim matches DataInf, but neither approaches SAVE, so similarity retrieval does not explain the gains. SAVE-Mix Random trails SAVE by a wide margin, showing that SAVE is not merely selecting "math-like" domains; instance-level ranking is essential. SAVE-Mix DataInf improves over DataInf, indicating that SAVE induces a useful domain allocation, yet remains below SAVE, and the gap to SAVE-FO confirms the contribution of token-conditional curvature.

Generalization. We repeat the autoformalization experiment with `Qwen3-4B`, and additionally evaluate a less syntactically rigid target: medical question answering with `Llama-3.1-8B-Instruct`,

Table 4: **Tool retrieval performance on ToolWeaver.** NDCG@k ($k \in \{1, 3, 5\}$) across query complexities I1–I3. Starred rows are reported ToolWeaver baselines. *Fixed-Budget* replaces an equal number of ToolWeaver target examples with SAVE-selected healing examples, keeping the original training budget; the last row adds the healing data to the ToolWeaver training set.

	I1			I2			I3		
Model	@1	@3	@5	@1	@3	@5	@1	@3	@5
BM25*	26.92	26.13	29.00	20.00	21.92	23.46	10.00	10.08	12.33
EmbSim*	50.50	48.15	53.41	46.00	39.58	43.05	18.00	17.77	20.94
ToolRetriever*	75.92	76.96	82.31	63.00	66.38	72.72	28.00	39.28	44.54
ToolGen*	88.50	88.83	91.65	84.00	85.65	89.02	81.00	80.83	85.83
ToolWeaver*	91.16	91.14	93.48	89.76	89.70	91.80	88.00	85.80	90.12
ToolWeaver + SAVE (Fixed-Budget)	92.43	92.31	94.68	91.38	91.05	93.24	90.00	87.18	91.32
ToolWeaver + SAVE (ours)	**93.45**	**93.07**	**95.62**	**93.18**	**92.91**	**94.46**	**92.00**	**89.27**	**93.23**

using 50K training examples sampled from UltraMedical [58], the same vocabulary-expansion procedure, the same 100K auxiliary pool, and MedQA [22] for evaluation. We compare against Target-only and DataInf, the strongest selector in Table 1. SAVE improves over DataInf on every metric in both settings (Table 3), indicating that its gains are not specific to the 8B scale or to formal, symbolic targets.

4.4 Additional Evaluation on ToolWeaver

We additionally evaluate SAVE in the ToolWeaver end-to-end tool-use setting [11]. This experiment tests whether SAVE can be inserted into an existing tool-use training pipeline as a post-hoc healing stage. We follow the original ToolWeaver training protocol with the `Llama3-8B` backbone, first training with the original vocabulary. We then apply target-driven vocabulary expansion, run SAVE to select non-target healing examples, and perform an additional final SFT stage.

For this setting, ToolWeaver is treated as the target task. The non-target pool contains auto-formalization, `chat`, `code`, `math`, and `stem` examples; ToolWeaver target examples are excluded. Because predefined tool-code tokens are rare in this pool, we augment them with frequent assistant-side structured-output tokens mined from ToolWeaver trajectories, such as `final_answer`. The ToolWeaver E_{new} signal should therefore be interpreted as schema/format alignment over active supervised tokens, not as semantic alignment to absent tool-code tokens. Full construction details are provided in Appendix C.

Table 4 reports tool retrieval performance. TOOLWEAVER+SAVE augments the ToolWeaver training data with SAVE-selected healing examples. TOOLWEAVER+SAVE (FIXED-BUDGET) uses the same selected healing examples but replaces an equal number of ToolWeaver target examples, keeping the total training budget matched to the original ToolWeaver setting. SAVE improves over the ToolWeaver baseline across all query complexities. The fixed-budget control further shows that the selected healing examples remain useful even without increasing the total training budget, suggesting that SAVE provides complementary schema-compatible signal rather than merely benefiting from a larger training set.

4.5 Analysis of SAVE-Selected Healing Data

To understand what SAVE selects, we decompose its utility as $U_\gamma(d) = U_{\text{LoRA}}(d) + \alpha_E U_{\text{vocab},\gamma}(d)$, induced by the LoRA-block and the token-conditional E_{new} influence terms, respectively; the decomposition is used only for analysis, and selection always uses U_γ. Intuitively, U_{LoRA} captures content-level transfer such as reasoning patterns or instruction structure, whereas $U_{\text{vocab},\gamma}$ captures alignment with the expanded vocabulary.

Figure 3 shows qualitative autoformalization examples with high values of different component utilities. High U_{LoRA} examples exhibit transferable reasoning or instruction structure without necessarily containing dense formal notation, whereas high $U_{\text{vocab},\gamma}$ examples contain target-vocabulary patterns such as modular notation and formal symbols. High U_γ examples exhibit both properties, illustrating why the two components are complementary.

Component-wise SAVE Utilities for Autoformalization

[High U_γ] Combined reasoning and vocabulary alignment
Prompt. Find the number of positive integer solutions of $x_1 + x_2 + x_3 = 15$ with $x_1 \leq 5$, $x_2 \leq 6$, $x_3 \leq 8$.
Response. `<think>` Stars and bars gives `\binom{14}{2}` $= 91$ [...] `inclusion-exclusion` : $|A| = 36$, $|B| = 28$, $|C| = 15$, $|A \cap B| = 3$ [...] `</think>` Therefore, **15** positive integer solutions.

[High U_{LoRA}] Content-level transfer signal
Prompt. [...] Pull up the GRID ETF — breakdown, holdings, performance, fees, and overlap with VDE or XLE.
Response. `<think>` User wants GRID breakdown and overlap with VDE/XLE [...] structure: explain specifics, then compare [...] `</think>` GRID top holdings: ABB, Siemens, Schneider, Eaton, NextEra. Overlap with [...] with XLE: Schneider, Siemens.

[High $U_{\text{vocab},\gamma}$] Target-vocabulary alignment signal
Prompt. Let $l \equiv 1 \pmod 2$ and $k = 2l^p + 2$. Find the least nonnegative residue system and absolute residues for $m \in \{3, k, 2k, 2k+3\}$ as $p \to \infty$.
Response. `<think>` l odd $\Rightarrow k = 2l$ `^p` $+2 \equiv 0 \pmod 4$ as $p \to$ `\infty` [...] $m = 3$: $\{0, 1, 2\}$, $\{-1, 0, 1\}$ ... $m = k$: $\{0, \ldots, k-1\}$ [...] `</think>` Analogous for $m = 2k$ and $m = 2k + 3$.

Figure 3: **Qualitative examples ranked by SAVE component utilities for autoformalization.** U_{LoRA} captures content-level transfer such as reasoning or instruction structure, while $U_{\text{vocab},\gamma}$ captures examples that activate and align new target-vocabulary embeddings. The combined SAVE utility U_γ favors examples where both signals contribute.

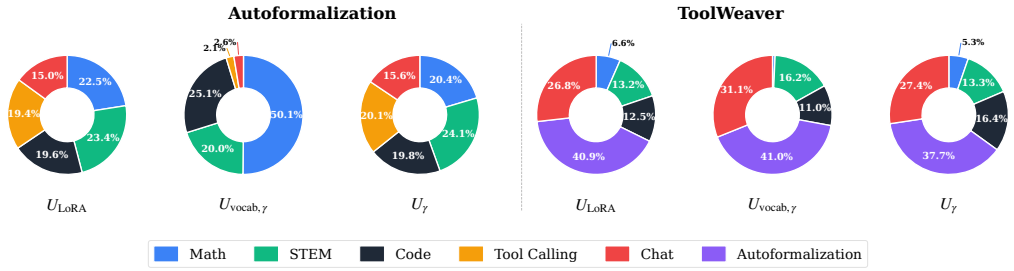

Figure 4: **Domain composition of high-utility examples under each SAVE component.** For each target, we compare the distributions induced by U_{LoRA}, $U_{\text{vocab},\gamma}$, and the combined U_γ. The two components capture different parts of the non-target pool.

Figure 4 further shows that the selected healing distribution is target-dependent. For autoformalization, the vocab-aware component strongly emphasizes math, STEM, and code-like examples, while assigning little mass to chat and tool-use examples that rarely activate the new Lean/math tokens. The combined SAVE score balances this vocabulary signal with the LoRA-side content signal, producing a more diverse but still target-compatible healing mixture. For ToolWeaver, the combined distribution shifts toward tool-use and chat examples, consistent with schema-compatible structured-response patterns rather than mathematical content.

4.6 Ablation Study

Vocab-aware as a plug-in. The SAVE utility $U_\gamma = U_{\text{LoRA}} + \alpha_E U_{\text{vocab},\gamma}$ is additive in the two components, so U_{LoRA} can be replaced by any gradient-based selection score on the LoRA-side parameters. Figure 6 compares DataInf, LESS, and OGS as choices of U_{LoRA}, each combined with our $U_{\text{vocab},\gamma}$ at $\alpha_E = 10$. The consistent gains suggest that $U_{\text{vocab},\gamma}$ is a complementary axis to existing gradient-based scores, with the largest gain for DataInf, whose Hessian-aware score is most closely aligned with the curvature-based vocab component.

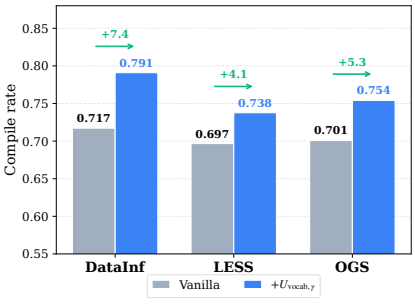

Figure 6: **Vocab-aware as a plug-in.** Adding $U_{\text{vocab},\gamma}$ to three gradient-based baselines consistently improves miniF2F compile rate.

Effect of α_E. The coefficient α_E balances content-level transfer (U_{LoRA}) and vocabulary-level alignment

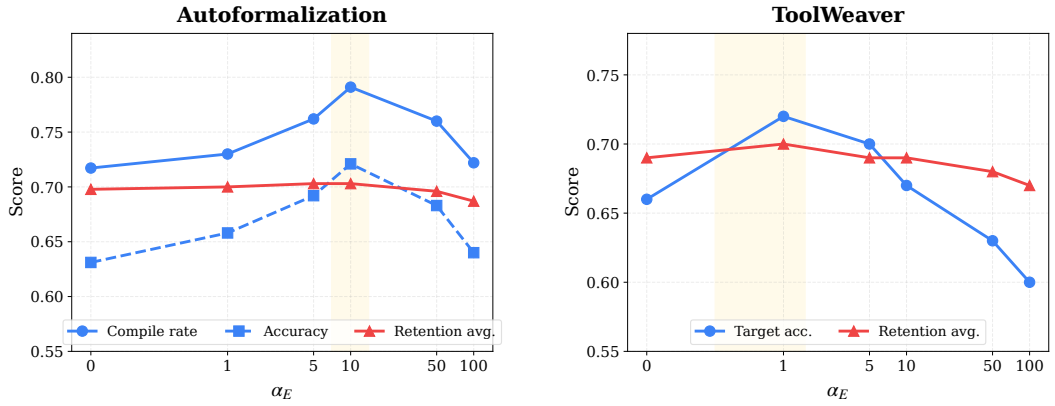

Figure 5: **Effect of the vocabulary-aware weight α_E.** Target metrics follow an inverted-U pattern: moderate vocabulary weighting improves performance, while excessive weighting hurts by overemphasizing token overlap without sufficient content compatibility. Broad retention is comparatively stable for moderate α_E.

($U_{\text{vocab},\gamma}$). Sweeping $\alpha_E \in \{0, 1, 5, 10, 50, 100\}$ shows an inverted-U trend on both targets (Figure 5), with autoformalization peaking around $\alpha_E \approx 10$ and ToolWeaver around $\alpha_E \approx 1$. This reflects the relative scale of the two signals: $|I_{\text{LoRA}}|/|I_{\text{vocab},\gamma}| \approx 22$ for autoformalization but close to 1 for ToolWeaver. We use these calibrated values in each setting. Moderate values preserve broad retention, while overly large α_E overemphasizes token overlap and hurts content compatibility. Appendix C.7 gives a practical recipe for setting α_E and the remaining hyperparameters without a sweep, and Appendices C.8 and A.6 report the scoring cost and a pair-block variant that retains cross-token curvature.

5 Related Work

Vocabulary expansion and cross-domain transfer. Vocabulary expansion is widely used to adapt LLMs to specialized settings, including cross-lingual transfer [12, 25], domain-specific text [8, 56], and tool-augmented generation [17, 49, 30]. The quality of new-token initialization has been identified as a key factor in downstream performance [6, 37]. A complementary line of work observes that fine-tuning on related domains (e.g., code or math) can improve performance on a target domain [36, 47, 43].

Influence-based data selection. Influence functions [26] estimate the effect of training instances on a target objective and have been adapted to LLMs through gradient-based approximations [24, 52, 27, 35, 59]. These estimators normalize curvature over the full candidate pool, implicitly assuming dense parameter updates; SAVE corrects this for the row-sparse updates of newly added embeddings and can be combined with any of them (Figure 6). We discuss further related work in Appendix B.

6 Conclusion

We studied vocabulary-expanded adaptation for syntactically specialized LLM targets such as Lean autoformalization and tool calling. Our results show that target-only vocabulary expansion can yield local gains while leaving newly added embeddings under-trained for related transfer. We proposed SAVE, a sparsity-aware influence estimator that selects healing examples by combining content-level transfer with token-conditional vocabulary alignment. Across both targets, SAVE improves target performance and related transfer over target-only training, naive auxiliary mixing, and existing selection baselines. Budget-matched attribution controls trace these gains to token-conditional curvature and instance-level ranking rather than similarity retrieval or domain mixture alone, and they persist on a smaller backbone and a medical target. Because the vocabulary-aware term is additive, it can also be plugged into existing influence-based selectors, improving DataInf, LESS, and OGS alike. Overall, effective vocabulary expansion requires not only adding new tokens, but also selecting target-compatible contexts that complete their embeddings. We discuss limitations and safeguards in Appendix E and Appendix F.

Acknowledgements

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST); No. RS-2022-II220184, Development and Study of AI Technologies to Inexpensively Conform to Evolving Policy on Ethics), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2022-NR070855), and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) with a grant funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research Agreement (No. RS-2024-00469482 & RS-2024-00509279). H.G. was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25480825), the Institute of Information & Communications Technology Planning & Evaluation (IITP) under the Leading Generative AI Human Resources Development program (IITP-2024-RS-2024-00397085) funded by the Korea government (MSIT), and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-02219317, AI Star Fellowship (Kookmin University)).

References

[1] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.

[2] Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

[3] Michael J Bommarito, Daniel Martin Katz, and Jillian Bommarito. Kl3m tokenizers: A family of domain-specific and character-level tokenizers for legal, financial, and preprocessing applications. *arXiv preprint arXiv:2503.17247*, 2025.

[4] Jialun Cao, Yaojie Lu, Meiziniu Li, Haoyang Ma, Haokun Li, Mengda He, Cheng Wen, Le Sun, Hongyu Zhang, Shengchao Qin, et al. From informal to formal–incorporating and evaluating llms on natural language requirements to verifiable formal proofs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26984–27003, 2025.

[5] Prateek Chanda, Saral Sureka, Parth Pratim Chatterjee, Krishnateja Killamsetty, Nikhil Shivakumar Nayak, and Ganesh Ramakrishnan. Learning what matters: Probabilistic task selection via mutual information for model finetuning. *arXiv preprint arXiv:2507.12612*, 2025.

[6] Daiwei Chen, Zhoutong Fu, Chengming Jiang, Haichao Zhang, Ran Zhou, Tan Wang, Chunnan Yao, Guoyao Li, Rui Cai, Yihan Cao, et al. Grounded token initialization for new vocabulary in lms for generative recommendation. *arXiv preprint arXiv:2604.02324*, 2026.

[7] Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, et al. Seed-prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*, 2025.

[8] Gautier Dagan, Gabriel Synnaeve, and Baptiste Roziere. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

[9] Zhiwei Deng, Tao Li, and Yang Li. Influential language data selection via gradient trajectory pursuit. *arXiv preprint arXiv:2410.16710*, 2024.

[10] Hongxin Ding, Yue Fang, Runchuan Zhu, Xinke Jiang, Jinyang Zhang, Yongxin Xu, Weibin Liao, Xu Chu, Junfeng Zhao, and Yasha Wang. 3ds: Medical domain adaptation of llms via decomposed difficulty-based data selection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19473–19495, 2025.

[11] Bowen Fang, Wen Ye, Yunyue Su, Jinghao Zhang, Qiang Liu, Yesheng Liu, Xin Sun, Shu Wu, Jiabing Yang, Baole Wei, and Liang Wang. ToolWeaver: Weaving collaborative semantics for scalable tool use in large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

[12] Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. Continual pre-training for cross-lingual llm adaptation: Enhancing japanese language capabilities. In *First Conference on Language Modeling*, 2024.

[13] Saibo Geng, Nathan Ranchin, Yunzhen Yao, Maxime Peyrard, Chris Wendler, Michael Gastpar, and Robert West. zip2zip: Inference-time adaptive tokenization via online compression. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

[15] Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

[16] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, New York, 1986.

[17] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36:45870–45894, 2023.

[18] Hongyi He, Xiao Liu, Zhenghao Lin, Mingni Tang, Yi Cheng, Jintao Wang, Wenjie Li, Peng Cheng, and Yeyun Gong. Learning from the best, differently: A diversity-driven rethinking on data selection. *arXiv preprint arXiv:2510.18909*, 2025.

[19] Christian Herold, Michael Kozielski, Nicholas Santavas, Yannick Versley, and Shahram Khadivi. Vocabulary customization for efficient domain-specific llm deployment. *arXiv preprint arXiv:2509.26124*, 2025.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[21] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[22] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[23] Feiyang Kang, Hoang Anh Just, Yifan Sun, Himanshu Jahagirdar, Yuanzhi Zhang, Rongxing Du, Anit Kumar Sahu, and Ruoxi Jia. Get more for less: Principled data selection for warming up fine-tuning in llms. In *The Twelfth International Conference on Learning Representations*, 2024.

[24] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.

[25] Seungduk Kim, Seungtaek Choi, and Myeongho Jeong. Efficient and effective vocabulary expansion towards multilingual large language models. *arXiv preprint arXiv:2402.14714*, 2024.

[26] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[27] Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] Iñaki Lacunza, Javier Garcia Gilabert, Francesca De Luca Fornaciari, Javier Aula-Blasco, Aitor Gonzalez-Agirre, Maite Melero, and Marta Villegas. Acadata: Parallel dataset of academic data for machine translation. *arXiv preprint arXiv:2510.12621*, 2025.

[29] Seungyoo Lee, Giung Nam, Moonseok Choi, Hyungi Lee, and Juho Lee. PANGEA: Projection-based augmentation with non-relevant general data for enhanced domain adaptation in LLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[30] Chenghao Li, Liu Liu, Baosheng Yu, Jiayan Qiu, and Yibing Zhan. Re-initialization token learning for tool-augmented large language models. *arXiv preprint arXiv:2506.14248*, 2025.

[31] Chong Li, Jiajun Zhang, and Chengqing Zong. Tokalign: Efficient vocabulary adaptation via token alignment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4109–4126, 2025.

[32] Yuan Li, Zhengzhong Liu, and Eric P Xing. Data mixing optimization for supervised fine-tuning of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[33] Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia LI, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. Goedel-prover: A frontier model for open-source automated theorem proving. In *Second Conference on Language Modeling*, 2025.

[34] Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, et al. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025.

[35] Zeming Liu, Yibai Liu, Shihang Wang, Zheming Song, Junzhe Wang, Jingjing Liu, Qingjie Liu, Guangxu Chen, and Yunhong Wang. Learn more, forget less: A gradient-aware data selection approach for llm. *Signal Processing*, page 110611, 2026.

[36] Zimu Lu, Aojun Zhou, Ke Wang, Houxing Ren, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Mathcoder2: Better math reasoning from continued pretraining on model-translated mathematical code. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Itay Nakash, Nitay Calderon, Eyal Ben-David, Elad Hoffer, and Roi Reichart. Adaptivocab: Enhancing llm efficiency in focused domains through lightweight vocabulary adaptation. In *Second Conference on Language Modeling*, 2025.

[38] Taido Purason, Pavel Chizhov, Ivan P Yamshchikov, and Mark Fishel. Teaching old tokenizers new words: Efficient tokenizer adaptation for pretrained models. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 6492–6516, 2026.

[39] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pages 9695–9717, 2024.

[40] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[41] ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

[42] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[43] Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, et al. rstar2-agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

[44] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867, 2020.

[45] Atula Tejaswi, Nilesh Gupta, and Eunsol Choi. Exploring design choices for building language-specific llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10485–10500, 2024.

[46] The mathlib Community. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, New Orleans, LA, USA, January 2020. ACM. doi: 10.1145/3372885.3373824. URL https://doi.org/10.1145/3372885.3373824.

[47] Abdul Waheed, Zhen Wu, Carolyn Rosé, and Daphne Ippolito. On code-induced reasoning in llms. *arXiv preprint arXiv:2509.21499*, 2025.

[48] Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025.

[49] Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. Theoremllama: Transforming general-purpose llms into lean4 experts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11953–11974, 2024.

[51] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[52] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning*, 2024.

[53] Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024.

[54] Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *The Thirteenth International Conference on Learning Representations*, 2025.

[55] Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Ming Ding. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32588–32599, 2025.

[56] Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. How can we effectively expand the vocabulary of llms with 0.01 gb of target language text? *Computational Linguistics*, 52(1): 295–330, 2026.

[57] 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.

[58] Kaiyan Zhang, Sihang Zeng, Ermo Hua, Ning Ding, Zhang-Ren Chen, Zhiyuan Ma, Haoxin Li, Ganqu Cui, Biqing Qi, Xuekai Zhu, Xingtai Lv, Jin-Fang Hu, Zhiyuan Liu, and Bowen Zhou. UltraMedical: Building specialized generalists in biomedicine. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, pages 26045–26081, 2024.

[59] Xiyang Zhang, Yuanhe Tian, Hongzhi Wang, and Yan Song. Training data selection with gradient orthogonality for efficient domain adaptation. *arXiv preprint arXiv:2602.06359*, 2026.

[60] Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2025. https://huggingface.co/datasets/math-ai/aime25, 2025. Hugging Face dataset, test split (30 problems), Apache-2.0 license.

[61] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

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

A Derivation and Assumptions for SAVE

This appendix provides the derivation and assumptions behind the SAVE estimator used in Section 3.1. The main paper defines the estimator directly; here we detail its connection to standard influence functions, DataInf, and the row-sparse gradient structure induced by newly added input embedding rows.

Throughout this appendix, d denotes a candidate example from the auxiliary non-target pool, and z' denotes a target validation example. Unless otherwise specified, we use

$$D_{\text{score}} = D_{\text{aux}}, \qquad n = |D_{\text{score}}|.$$

We use $\mathcal{I}$ for the classical influence quantity and $\widehat{\mathcal{I}}$ for DataInf-style approximations. When there is no ambiguity, we write $\widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d)$ for the combined SAVE estimator.

A.1 Standard Influence and DataInf

Let θ denote the parameters used for influence estimation. The target validation loss is

$$\mathcal{L}_{\text{val}}(\theta) = \frac{1}{|D_{\text{val}}|} \sum_{z' \in D_{\text{val}}} \mathcal{L}(z'; \theta).$$

The classical upweighting influence function [26] of a candidate example d on $\mathcal{L}_{\text{val}}$ is

$$\mathcal{I}(d) = -\nabla_\theta \mathcal{L}_{\text{val}}(\theta)^\top H^{-1} \nabla_\theta \mathcal{L}(d; \theta),$$

where H is the Hessian, or a Hessian proxy, of the training objective evaluated at the surrogate model.

DataInf [27] makes this computation tractable by using a block-diagonal curvature approximation and an inversion-averaging approximation. Let the scoring parameters be partitioned into blocks $\{\theta_b\}_{b \in \mathcal{B}}$. For each block b, define

$$g_{d,b} = \nabla_{\theta_b} \mathcal{L}(d; \theta), \qquad u_b = \nabla_{\theta_b} \mathcal{L}_{\text{val}}(\theta).$$

With an empirical Fisher or generalized Gauss–Newton proxy, the block curvature is

$$H_b = \frac{1}{n} \sum_{d_i \in D_{\text{score}}} g_{d_i,b} g_{d_i,b}^\top + \lambda_b I.$$

The exact block-wise influence under this proxy is

$$\mathcal{I}_b(d) = -u_b^\top H_b^{-1} g_{d,b}.$$

Following DataInf [27], we approximate the inverse of the averaged block curvature by the average of rank-one inverse terms:

$$H_b^{-1} \approx \widehat{M}_b = \frac{1}{n} \sum_{d_i \in D_{\text{score}}} \left(g_{d_i,b} g_{d_i,b}^\top + \lambda_b I \right)^{-1}.$$

Throughout the paper, $\widehat{M}_b$ denotes an inverse-curvature preconditioner approximating H_b^{-1}, not the curvature matrix H_b itself.

Since each term is a rank-one update to a diagonal matrix, the Sherman–Morrison identity gives

$$\left(g_{d_i,b} g_{d_i,b}^\top + \lambda_b I \right)^{-1} = \frac{1}{\lambda_b} \left[I - \frac{g_{d_i,b} g_{d_i,b}^\top}{\lambda_b + \|g_{d_i,b}\|^2} \right].$$

Substituting this into the block influence yields

$$\widehat{\mathcal{I}}_b^{\text{DI}}(d) = -\frac{1}{\lambda_b} \left[u_b^\top g_{d,b} - \frac{1}{n} \sum_{d_i \in D_{\text{score}}} \frac{(u_b^\top g_{d_i,b})(g_{d_i,b}^\top g_{d,b})}{\lambda_b + \|g_{d_i,b}\|^2} \right].$$

This is the standard DataInf term used for dense LoRA blocks in SAVE.

A.2 Row-Sparse Gradients of New Input Embeddings

Vocabulary expansion introduces new input embedding rows

$$E_{\text{new}} = \{e_v : v \in V_{\text{new}}\}.$$

For a new token $v \in V_{\text{new}}$, define

$$g_{d,v} = \nabla_{e_v} \mathcal{L}(d; \theta), \qquad u_v = \nabla_{e_v} \mathcal{L}_{\text{val}}(\theta).$$

Because e_v is an input embedding row, it receives input-side gradients only on examples whose expanded tokenization contains v. Let

$$S_v = \{d_i \in D_{\text{score}} : v \in \tau_{\text{new}}(d_i)\}, \qquad n_v = |S_v|, \qquad p_v = \frac{n_v}{n}.$$

Then

$$g_{d,v} = 0 \qquad \text{for } d \notin S_v.$$

If $n_v = 0$, token v does not contribute to the token-level score.

If we apply the joint empirical-risk normalization to this token row, the empirical Fisher block is

$$H_v^{\text{joint}} = \frac{1}{n} \sum_{d_i \in D_{\text{score}}} g_{d_i,v} g_{d_i,v}^\top + \lambda_v I.$$

Since $g_{d_i,v} = 0$ for $d_i \notin S_v$, this becomes

$$H_v^{\text{joint}} = \frac{1}{n} \sum_{d_i \in S_v} g_{d_i,v} g_{d_i,v}^\top + \lambda_v I.$$

Define the token-conditional empirical Fisher

$$F_v = \frac{1}{n_v} \sum_{d_i \in S_v} g_{d_i,v} g_{d_i,v}^\top.$$

Then

$$H_v^{\text{joint}} = p_v F_v + \lambda_v I.$$

This expression shows why rare tokens are problematic under fixed damping. When $n_v \ll n$, we have $p_v \ll 1$. If

$$\|p_v F_v\|_2 \ll \lambda_v,$$

then

$$H_v^{\text{joint}} \approx \lambda_v I, \qquad (H_v^{\text{joint}})^{-1} \approx \lambda_v^{-1} I.$$

The token-level influence then degenerates to

$$\mathcal{I}_v(d) \approx -\lambda_v^{-1} u_v^\top g_{d,v},$$

which is a first-order gradient similarity score. Thus, while the $1/n$ normalization is appropriate for the joint empirical-risk curvature, it can suppress token-specific second-order information for sparse new embedding rows.

A.3 Token-Conditional Curvature

To preserve token-specific curvature, SAVE defines an interpolated token-level curvature proxy

$$H_v^{(\gamma)} = p_v^\gamma F_v + \lambda_v I, \qquad \gamma \in [0, 1].$$

The endpoints have the following interpretations:

$$\gamma = 1: \quad H_v^{(1)} = p_v F_v + \lambda_v I = H_v^{\text{joint}},$$

which matches the standard joint normalization at the Hessian-proxy level, and

$$\gamma = 0: \quad H_v^{(0)} = F_v + \lambda_v I,$$

which gives the fully token-conditional normalization over the effective update set S_v.

Following the DataInf inversion-averaging approximation [27], but applying it over the effective token update set S_v, we obtain

$$(H_v^{(\gamma)})^{-1} \approx \widehat{M}_{v,\gamma} = \frac{1}{n_v} \sum_{d_i \in S_v} \left(p_v^\gamma g_{d_i,v} g_{d_i,v}^\top + \lambda_v I\right)^{-1}.$$

Using Sherman–Morrison,

$$\left(p_v^\gamma g_{d_i,v} g_{d_i,v}^\top + \lambda_v I\right)^{-1} = \frac{1}{\lambda_v}\left[I - \frac{p_v^\gamma g_{d_i,v} g_{d_i,v}^\top}{\lambda_v + p_v^\gamma \|g_{d_i,v}\|^2}\right].$$

Therefore,

$$\widehat{M}_{v,\gamma} = \frac{1}{n_v \lambda_v} \sum_{d_i \in S_v} \left[I - \frac{p_v^\gamma g_{d_i,v} g_{d_i,v}^\top}{\lambda_v + p_v^\gamma \|g_{d_i,v}\|^2}\right].$$

Substituting this into

$$\widehat{\mathcal{I}}^{\mathrm{DI}}_{v,\gamma}(d) = -u_v^\top \widehat{M}_{v,\gamma} g_{d,v}$$

gives the closed-form token influence

$$\widehat{\mathcal{I}}^{\mathrm{DI}}_{v,\gamma}(d) = -\frac{1}{\lambda_v}\left[u_v^\top g_{d,v} - \frac{1}{n_v} \sum_{d_i \in S_v} \frac{p_v^\gamma (u_v^\top g_{d_i,v})(g_{d_i,v}^\top g_{d,v})}{\lambda_v + p_v^\gamma \|g_{d_i,v}\|^2}\right].$$

For $d \notin S_v$, $g_{d,v} = 0$, so

$$\widehat{\mathcal{I}}^{\mathrm{DI}}_{v,\gamma}(d) = 0.$$

Thus, in the combined score, it is sufficient to sum over

$$v \in \tau_{\mathrm{new}}(d) \cap V_{\mathrm{new}}.$$

This estimator is not an unbiased approximation of the joint empirical Hessian for e_v. Rather, it estimates a token-conditional curvature over the effective update set S_v. This conditional formulation intentionally preserves token-specific curvature in the row-sparse regime where the standard joint normalization becomes dominated by damping.

A.4 Relation to Vanilla DataInf

It is important to distinguish the SAVE token-conditional estimator from vanilla DataInf applied directly to the token row over all samples.

Vanilla DataInf [27] over the full candidate pool would use

$$M^{\mathrm{DI}}_{v,\mathrm{vanilla}} = \frac{1}{n} \sum_{d_i \in D_{\mathrm{score}}} \left(g_{d_i,v} g_{d_i,v}^\top + \lambda_v I\right)^{-1}.$$

For $d_i \notin S_v$, we have $g_{d_i,v} = 0$, and therefore

$$\left(g_{d_i,v} g_{d_i,v}^\top + \lambda_v I\right)^{-1} = \lambda_v^{-1} I.$$

Hence

$$M^{\mathrm{DI}}_{v,\mathrm{vanilla}} = (1 - p_v)\lambda_v^{-1} I + p_v \frac{1}{n_v} \sum_{d_i \in S_v} \left(g_{d_i,v} g_{d_i,v}^\top + \lambda_v I\right)^{-1}.$$

When $p_v \ll 1$, the first term dominates, and the preconditioner approaches the identity preconditioner $\lambda_v^{-1} I$. This is the sparse-row degeneration that SAVE avoids.

The SAVE estimator with $\gamma = 1$ is

$$\widehat{M}_{v,1} = \frac{1}{n_v} \sum_{d_i \in S_v} \left(p_v g_{d_i,v} g_{d_i,v}^\top + \lambda_v I\right)^{-1}.$$

Thus, $\gamma = 1$ matches the standard joint normalization at the Hessian-proxy level,

$$H_v^{(1)} = p_v F_v + \lambda_v I,$$

but it does not exactly recover vanilla DataInf over all samples, because vanilla DataInf additionally averages zero-gradient examples that contribute $\lambda_v^{-1} I$. This distinction is central to the token-conditional design.

A.5 Token Frequency and Per-Example Utility

A natural concern is that frequent but low-information expanded tokens could inflate example utility. SAVE does not directly reward corpus frequency. Under the default $\gamma = 0$,

$$\widehat{M}_{v,0} = \frac{1}{n_v} \sum_{d_i \in S_v} \left(g_{d_i,v} g_{d_i,v}^\top + \lambda_v I \right)^{-1} \approx \mathbb{E}_{d|v} \left[\left(g_{d,v} g_{d,v}^\top + \lambda_v I \right)^{-1} \right],$$

which averages rather than sums over activating examples and therefore does not increase with n_v. Higher frequency improves the support of the estimate, but not the per-example utility. Since $\widehat{M}_{v,0}$ is symmetric positive definite, the contribution of token v to the utility of d is

$$U_v(d) = u_v^\top \widehat{M}_{v,0}\, g_{d,v} = \left\| \widehat{M}_{v,0}^{1/2} u_v \right\| \left\| \widehat{M}_{v,0}^{1/2} g_{d,v} \right\| \cos \theta_{d,v},$$

where $\theta_{d,v}$ is the angle between $\widehat{M}_{v,0}^{1/2} u_v$ and $\widehat{M}_{v,0}^{1/2} g_{d,v}$. A frequent token thus contributes little when its target gradient is small or the candidate gradient is weakly aligned, and contributes negatively when the two are misaligned.

The vocabulary utility $U_{\text{vocab}}(d) = \sum_{v \in \mathcal{A}(d)} U_v(d)$ is an unnormalized sum, so examples that activate more expanded tokens can receive larger utility only when their individual target-conditioned contributions are positive. Empirically, SAVE-selected examples contain 18.6–26.6% more expanded-token occurrences than the pool average across domains. We view this moderate increase as intended: an example must activate new rows to provide a healing update. Occurrence alone is insufficient, however, since each activation contributes only through the signed alignment $u_v^\top \widehat{M}_{v,0} g_{d,v}$; consistent with this, the rarity-weighted New-token TF-IDF selector remains well below SAVE (Table 2). Under extreme frequency imbalance, normalizing by $|\mathcal{A}(d)|$ or clipping per-token contributions are simple safeguards.

A.6 Cross-Token Block Approximation

If all new embedding rows are stacked into a single block, the empirical Fisher contains cross-token blocks. For two new tokens $u, v \in V_{\text{new}}$, the cross-token block is

$$[H_E]_{u,v} = \frac{1}{n} \sum_{d_i \in D_{\text{score}}} g_{d_i,u} g_{d_i,v}^\top.$$

Since $g_{d_i,u} = 0$ unless $d_i \in S_u$, and $g_{d_i,v} = 0$ unless $d_i \in S_v$, the only nonzero contributions come from the intersection:

$$[H_E]_{u,v} = \frac{1}{n} \sum_{d_i \in S_u \cap S_v} g_{d_i,u} g_{d_i,v}^\top.$$

SAVE treats each new embedding row as a separate token-level block and neglects these cross-token terms, analogous to the cross-block approximation in standard DataInf [27]. This approximation is most accurate when new tokens do not strongly co-occur or when their cross-token gradient correlations are small.

As a diagnostic, one can measure the normalized cross-token curvature ratio

$$r_{u,v} = \frac{\left\| \sum_{d_i \in S_u \cap S_v} g_{d_i,u} g_{d_i,v}^\top \right\|_F}{\sqrt{\left\| \sum_{d_i \in S_u} g_{d_i,u} g_{d_i,u}^\top \right\|_F \left\| \sum_{d_i \in S_v} g_{d_i,v} g_{d_i,v}^\top \right\|_F}}.$$

Large values of $r_{u,v}$ suggest that grouping frequently co-occurring new tokens into a larger block may provide a more accurate approximation.

Empirical check: SAVE-PAIRFULL. To test whether omitting cross-token curvature matters in practice, we evaluate a richer block-wise variant. Using only co-occurrence statistics from the auxiliary candidate pool, we form $K = 256$ disjoint expanded-token pairs with the highest co-occurrence. For each pair $p = (u, v)$, we concatenate the embedding-row gradients $g_{d,p} = [g_{d,u}; g_{d,v}]$ and estimate a joint DataInf-style preconditioner over $S_p = S_u \cup S_v$, which retains the within-pair off-diagonal terms that singleton SAVE omits; unpaired tokens keep the per-token preconditioner.

Table 5: **Singleton vs. pair-block curvature** (autoformalization, mean ± std over three seeds).

Method	miniF2F Compile	miniF2F Acc.	AIME2025	Scoring cost
SAVE	0.7909 ± 0.0083	0.7213 ± 0.0082	0.7346 ± 0.0019	$1.00\times$
SAVE-PairFull	0.7965 ± 0.0085	0.7268 ± 0.0125	0.7305 ± 0.0084	$1.22\times$

α_E is set by the component-scale calibration of Appendix C.7, and all other scoring, budget, and final-SFT settings are fixed. Table 5 reports results over three training seeds.

Relative to singleton SAVE, PairFull changes the means by $+0.56$ and $+0.55$ points on miniF2F compile rate and accuracy and by -0.41 points on AIME2025, retains 95.68% of SAVE's selected healing examples, and requires $1.22\times$ the scoring GPU-hours under the same hardware and parallelism. For the tested high-co-occurrence pairs, relaxing the singleton approximation therefore changes only a small fraction of the selected data and yields sub-one-point differences, supporting singleton SAVE as a favorable performance–efficiency operating point.

A.7 Combined SAVE Score and Component Utilities

SAVE combines dense LoRA-block influence and token-conditional new-embedding influence. Define

$$\widehat{\mathcal{I}}^{\mathrm{DI}}_{\mathrm{LoRA}}(d) = \sum_{b \in \mathcal{B}_{\mathrm{LoRA}}} \widehat{\mathcal{I}}^{\mathrm{DI}}_{b}(d),$$

and

$$\widehat{\mathcal{I}}^{\mathrm{DI}}_{\mathrm{vocab},\gamma}(d) = \sum_{v \in \tau_{\mathrm{new}}(d) \cap V_{\mathrm{new}}} \widehat{\mathcal{I}}^{\mathrm{DI}}_{v,\gamma}(d).$$

The combined SAVE influence estimator is

$$\widehat{\mathcal{I}}_{\mathrm{SAVE},\gamma}(d) = \widehat{\mathcal{I}}^{\mathrm{DI}}_{\mathrm{LoRA}}(d) + \alpha_E \widehat{\mathcal{I}}^{\mathrm{DI}}_{\mathrm{vocab},\gamma}(d),$$

where $\alpha_E \geq 0$ calibrates the relative scale of the dense and vocabulary terms. Following the standard influence convention, $\widehat{\mathcal{I}}_{\mathrm{SAVE},\gamma}(d) < 0$ indicates that adding d is estimated to decrease the target validation loss. We therefore define the selection utility

$$U_\gamma(d) = -\widehat{\mathcal{I}}_{\mathrm{SAVE},\gamma}(d).$$

For qualitative analysis only, we decompose the utility into LoRA and vocabulary components:

$$U_\gamma(d) = U_{\mathrm{LoRA}}(d) + \alpha_E U_{\mathrm{vocab},\gamma}(d),$$

where

$$U_{\mathrm{LoRA}}(d) = -\widehat{\mathcal{I}}^{\mathrm{DI}}_{\mathrm{LoRA}}(d),$$

and

$$U_{\mathrm{vocab},\gamma}(d) = -\widehat{\mathcal{I}}^{\mathrm{DI}}_{\mathrm{vocab},\gamma}(d).$$

The component-wise rankings shown in Section 4.5 are diagnostic only. They are used to interpret what each term emphasizes; final training data are always selected using the combined utility U_γ.

A.8 Assumptions and Practical Details

Untied embeddings. The SAVE influence estimator is applied to the newly added input embedding rows E_{new}, and does not include the corresponding LM-head rows. This is appropriate for models or scoring configurations in which input and output embeddings can be treated separately. If a model uses tied input-output embeddings, the shared row receives both input-side and output-side gradients. Under full-softmax training, the output-side gradient can be sample-dense, violating the row-sparse assumption $g_{d,v} = 0$ for $d \notin S_v$. In that case, one should either untie the new token rows for scoring or compute the token-conditional term using only the input-embedding gradient path.

Input-side gradients under teacher forcing. Although LM-head rows are excluded from the influence estimator, response-side tokens can still produce input-embedding gradients under autoregressive teacher forcing. For a tokenized sequence $x_1, \ldots, x_T$, the supervised loss can be written as

$$\mathcal{L}(d;\theta) = \sum_{t \in \mathcal{M}(d)} \ell_t(x_{<t};\theta),$$

where $\mathcal{M}(d)$ denotes the supervised prediction positions. If a new token v appears at position s, then its input embedding can affect losses at later positions $t > s$. Schematically,

$$\nabla_{e_v} \mathcal{L}(d;\theta) = \sum_{s:x_s=v} \sum_{\substack{t>s \\ t \in \mathcal{M}(d)}} \frac{\partial \ell_t(x_{<t};\theta)}{\partial e_v(x_s)}.$$

Therefore, structured response tokens can induce strong E_{new}-side signals through the input path alone. This is especially relevant for schema-heavy targets such as tool use, where field names or return-type markers appear repeatedly in assistant responses. This effect reflects input-side schema alignment, not LM-head contamination.

Small effective support. When n_v is very small, the token-conditional curvature estimate can have high variance. In practice, one may apply the token-level term only when $n_v \geq n_{\min}$, increase damping for rare tokens, or use shrinkage toward a global curvature estimate. For example,

$$\lambda_v = \lambda_0 \left(1 + \frac{\tau}{n_v}\right)$$

increases damping for rare tokens.

Candidate-pool influence score. The SAVE preconditioners are computed over

$$D_{\text{score}} = D_{\text{aux}},$$

because the scoring stage evaluates candidate auxiliary examples under a common candidate-pool curvature estimate. The LoRA preconditioner averages over the full candidate pool D_{score}, while each token-conditional preconditioner averages over the effective update set $S_v \subseteq D_{\text{score}}$. Therefore, $\widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d)$ should be interpreted as a candidate-pool selection score rather than an exact influence value for the final SFT objective.

Signed-threshold selection with a healing budget. SAVE first applies the signed-threshold rule

$$D_{\text{pos}} = \{d \in D_{\text{aux}} : U_\gamma(d) > 0\} = \{d \in D_{\text{aux}} : \widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d) < 0\}.$$

This follows the influence-function sign convention: examples with negative estimated influence are predicted to decrease the target validation loss.

Given a healing-to-target budget ratio ρ, the maximum number of selected healing examples is

$$B_\rho = \lfloor \rho |D_{\text{trn}}| \rfloor .$$

The realized budget is

$$B_{\text{heal}} = \min(B_\rho, |D_{\text{pos}}|),$$

and the final healing set is

$$D_{\text{heal}} = \text{Top}_{B_{\text{heal}}}(D_{\text{pos}}; U_\gamma).$$

Thus, non-positive-utility examples are never selected, while ρ controls the maximum amount of healing data added relative to the fixed target training set. The realized healing-to-target ratio is

$$\rho_{\text{real}} = \frac{|D_{\text{heal}}|}{|D_{\text{trn}}|} \leq \rho.$$

For fair comparison with fixed-budget baselines, we evaluate them using the same realized budget B_{heal}. Random selection samples B_{heal} examples uniformly from D_{aux}, and ranking-based baselines select their top B_{heal} examples according to their own scores.

Final SFT pool and source-balanced mini-batches. After selecting D_{heal}, the final SFT pool is

$$D_{\mathrm{SFT}} = D_{\mathrm{trn}} \cup D^*_{\mathrm{heal}},$$

where $D^*_{\mathrm{heal}} = D_{\mathrm{heal}}$ by default. We use $\cup$ to denote the dataset-level training pool formed from the target and healing sources. If sequence-level distillation is used, D^*_{heal} replaces the response side of each selected auxiliary example with a teacher response generated by M_{base}.

We do not introduce a separate loss-level reweighting coefficient. Instead, we control the relative contribution of target and healing examples through mini-batch composition. Let $\mathcal{B}_{\mathrm{trn}} \subset D_{\mathrm{trn}}$ and $\mathcal{B}_{\mathrm{heal}} \subset D^*_{\mathrm{heal}}$ denote the target and healing portions of a final-SFT mini-batch, with fixed sizes m_{trn} and m_{heal}. The mixed mini-batch is

$$\mathcal{B} = \mathcal{B}_{\mathrm{trn}} \cup \mathcal{B}_{\mathrm{heal}}.$$

The final model is trained with the standard SFT objective over the mixed mini-batch:

$$\min_{\phi} \mathbb{E}_{\mathcal{B}_{\mathrm{trn}}, \mathcal{B}_{\mathrm{heal}}} \left[\frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} -\log M_\phi(y \mid x) \right].$$

Equivalently, the per-update healing sampling mass is

$$\pi_{\mathrm{heal}} = \frac{m_{\mathrm{heal}}}{m_{\mathrm{trn}} + m_{\mathrm{heal}}}.$$

In the default balanced setting, $m_{\mathrm{trn}} = m_{\mathrm{heal}} = 8$, so $\pi_{\mathrm{heal}} = 0.5$. Thus, ρ controls the size of the selected healing pool, while the mini-batch sampler controls the per-update contribution of target and healing examples.

B Extended Related Work

Vocabulary expansion for LLM adaptation. Vocabulary expansion has been used both to improve inference efficiency and to specialize LLMs for new domains by introducing target-specific tokens. Multilingual and cross-lingual adaptation work expands the tokenizer to underrepresented languages [12, 25, 45, 56], while domain-focused efforts target legal and financial corpora [3, 19] or pre-training and domain adaptation more broadly [8]. Recent methods explore lightweight or inference-time tokenizer adaptation [37, 13, 38, 31], and the choice of how to initialize new embeddings is itself an active area, with strategies ranging from sub-token aggregation to grounded initialization from related contexts [37, 6]. Vocabulary expansion has also been adopted in agentic settings, where new tokens encode tool calls or schema markers [17, 49, 30, 11]. While expansion typically improves target-task performance, it often comes at the cost of broader capability, motivating methods that preserve or restore general-purpose abilities after adaptation.

Influence-based data selection. Estimating the effect of a single training example on validation loss has a long history in robust statistics [15, 16] and was popularized for deep learning by Koh and Liang [26]. Recent work has scaled this idea to LLMs by exploiting low-rank parameter structure: DataInf [27] provides a closed-form influence approximation for LoRA-tuned models [20], LESS [52] matches gradient directions on a target validation set, and gradient-orthogonality or trajectory-based criteria target capability-aware selection and forgetting prevention [59, 9, 35]. Other gradient-aware methods focus on subset selection for training efficiency [24], diversity-driven selection [18], mutual information-based task selection [5], warmup-time selection [23], mixing optimization [32], and domain-specific difficulty-based selection [10]. These methods generally assume the parameter gradient is dense across the training pool, and treat all parameter blocks symmetrically.

Capability preservation and cross-domain transfer. Mitigating capability loss during specialized fine-tuning typically involves mixing auxiliary general-purpose data with target data, with related lines exploring how mathematical or code-style data transfers to other reasoning tasks [36, 47, 43]. Autoformalization has been advanced by data synthesis and reinforcement learning over Lean theorem proving [41, 34, 7, 50, 4], and tool-use language models combine new vocabulary with structured-call schemas [11]. PANGEA [29] takes a related but distinct route: it leverages *non-relevant general-purpose data* to generate diverse, higher-quality synthetic data for the target domain via projection-based augmentation, and validates the approach on GSM8K, MedQA, and FinQA. Other lines preserve capability through replay, regularization, or instruction-mixing during continual fine-tuning, but typically operate without modifying the tokenizer.

C Experiments

C.1 Models

A summary of the pre-trained weights employed across our experiments is provided below.

- `Qwen3-8B`: https://huggingface.co/Qwen/Qwen3-8B, licensed under the Apache 2.0 License.[1]
- `Llama-3-8B`: https://huggingface.co/meta-llama/Meta-Llama-3-8B, licensed under the Meta Llama 3 Community License Agreement.[2]
- `Qwen3-4B`: https://huggingface.co/Qwen/Qwen3-4B, licensed under the Apache 2.0 License (scale study).
- `Llama-3.1-8B-Instruct`: https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct, licensed under the Llama 3.1 Community License Agreement (medical target).
- `all-mpnet-base-v2`: https://huggingface.co/sentence-transformers/all-mpnet-base-v2, licensed under the Apache 2.0 License (embedding model for the EmbSim baseline).

We use `Qwen3-8B` [57] as the base model for autoformalization and `Llama-3-8B` [14] for ToolWeaver, matching the backbone used in the original ToolWeaver protocol [11].

C.2 Datasets

We list every dataset and benchmark used, together with its role in our pipeline and its license as stated by the owner (checked in November 2026).

Target training datasets.

- `Goedel-Pset-v1` [33]: https://huggingface.co/datasets/Goedel-LM/Goedel-Pset-v1, licensed under the MIT License. An autoformalization corpus pairing informal mathematical statements with Lean formalizations. We randomly sample 50,000 examples to construct D_{trn} for the autoformalization target.
- `ToolWeaver data` [11]: https://github.com/Fwibo/ToolWeaver. The tool-use training corpus released with ToolWeaver, used as the target dataset for the tool-calling setting. The release does not specify a license; the data are derived from ToolBench, which is distributed under the Apache 2.0 License.
- `UltraMedical` [58]: https://huggingface.co/datasets/TsinghuaC3I/UltraMedical, licensed under the MIT License. We randomly sample 50,000 examples as the target training set for the medical setting.

Non-target auxiliary pool. For autoformalization, we construct D_{aux} from `Nemotron-Post-Training-Dataset-v1` [1] (https://huggingface.co/datasets/nvidia/Nemotron-Post-Training-Dataset-v1, licensed under CC BY 4.0), sampling 20K examples from each of the chat, code, math, and STEM categories. We additionally include 20K ToolWeaver end-to-end examples as non-target tool-use data, yielding a 100K-example candidate pool. We exclude the Nemotron `tool_calling` category to ensure $D_{\text{aux}} \cap D_{\text{trn}} = \emptyset$.

For ToolWeaver, we construct the non-target pool from autoformalization, chat, code, math, and STEM examples, excluding ToolWeaver target examples from the candidate pool. Thus, in both target settings, SAVE selects healing examples only from non-target auxiliary data.

Preliminary-study datasets. For the preliminary study in Figure 1, we use the math subset of `Nemotron-Post-Training-Dataset-v1` as related auxiliary data and compare it against two less related datasets: `Nemotron-SFT-Instruction-Following-Chat-v2` [2] (https://huggingface.co/datasets/nvidia/Nemotron-SFT-Instruction-Following-Chat-v2, licensed under ODC-By), a general-purpose instruction-following chat dataset, and `ACADATA` [28] (https://huggingface.co/datasets/BSC-LT/ACAData, licensed under CC BY 4.0), a parallel corpus for academic machine translation.

[1] https://huggingface.co/Qwen/Qwen3-8B/blob/main/LICENSE
[2] https://llama.meta.com/llama3/license/

Vocabulary construction. Lean Workbook (https://huggingface.co/datasets/internlm/Lean-Workbook, licensed under the Apache 2.0 License) is used only to construct the expanded vocabulary (Appendix C.3) and not as training data.

Evaluation datasets. We follow the original benchmark releases and dataset splits used in prior work, and report results on the following suites. All evaluation benchmarks are used only for post-hoc evaluation.

- **miniF2F** [61] ($N = 244$, test split): https://github.com/openai/miniF2F; the Lean portion is licensed under the Apache License. A cross-system formal-mathematics benchmark drawn from AIME, AMC, IMO, and high-school to undergraduate problems. Each statement is formalized in Lean, Metamath, and Isabelle. We report Lean compile rate and semantic accuracy on the test split.
- **AIME2025** [60] ($N = 30$): https://huggingface.co/datasets/math-ai/aime25, licensed under the Apache 2.0 License. AIME 2025 I&II contest problems, used as a related mathematical-transfer benchmark.
- **MMLU-Pro** [51] ($N = 12{,}032$): https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro, licensed under the MIT License. Challenging multi-choice questions spanning broad academic domains, used as a general-knowledge retention benchmark.
- **LiveCodeBench-v5** [21] ($N = 880$): https://huggingface.co/datasets/livecodebench/code_generation_lite, released under a Creative Commons license (variant not specified by the owner). A time-stamped code-generation benchmark. We use the official `release_v5` in the `codegeneration` scenario, identical to the release setting used in prior work.
- **IF-Eval** [62] ($N = 541$): https://huggingface.co/datasets/google/IFEval, licensed under the Apache 2.0 License. An instruction-following benchmark consisting of prompts paired with 25 categories of programmatically verifiable instructions, such as length, keyword, and format constraints.
- **ToolBench** [39]: https://github.com/OpenBMB/ToolBench, licensed under the Apache 2.0 License for research and educational use. We follow the ToolWeaver [11] and ToolGen [49] evaluation protocol and report NDCG@k for $k \in \{1, 3, 5\}$ across query complexities I1, I2, and I3.
- **MedQA** [22]: https://github.com/jind11/MedQA, licensed under the MIT License. A multiple-choice medical question-answering benchmark built from medical licensing exams; we report accuracy on the test split.

C.3 Vocabulary Expansion Details

Autoformalization. For autoformalization, we add 6,572 tokens to the original `Qwen3-8B` tokenizer. We construct a Lean-centered vocabulary mining corpus from the formal and informal statements in `Goedel-Pset-v1` [33], together with Lean data used in the `Lean-workbook-proofs` pipeline. This corpus is used only for vocabulary construction and is not used as additional target training data.

We first train a candidate tokenizer on this Lean-centered corpus and then re-tokenize the corpus to collect candidate-token frequencies. Each candidate token c is scored by a frequency–savings criterion:

$$\text{score}(c) = \text{freq}(c) \cdot s(c), \qquad s(c) = |\tau_{\text{base}}(c)| - 1,$$

where $\tau_{\text{base}}(c)$ denotes the decomposition of candidate c under the original base tokenizer. The savings term $s(c)$ measures how many base-token pieces are saved by representing c as a single new token.

After selecting the highest-scoring candidates, we preserve the original base-tokenizer merges and add only the dependency-closure merges required to realize the selected candidates. We also add a small set of Lean-related keywords and operators when they remain fragmented by the base tokenizer. The resulting expanded tokenizer contains formal-mathematics tokens such as `Mathlib`, `Aesop`, `BigOperators`, `Topology`, `Finset`, `linarith`, `\left`, `\right`, `\sqrt`, `\infty`, `\binom`, `.Icc`, `toFinset`, and `dfrac`.

After extending the tokenizer, we preserve special-token indices and resize the model embedding matrices. For each new token $v \in V_{\text{new}}$, let

$$\tau_{\text{base}}(v) = [t_1, \ldots, t_k]$$

be its decomposition under the original tokenizer. We initialize the new input embedding row $e_v \in E_{\text{new}}$ using the AdaptiVocab-style exponential composition rule [37]:

$$e_v = \sum_{i=1}^{k} w_i e_{t_i}^{\text{FFT}}, \qquad w_i = \frac{\exp(\beta i)}{\sum_{j=1}^{k} \exp(\beta j)}.$$

We use $\beta = 2$. Here, $e_{t_i}^{\text{FFT}}$ is the input embedding row of sub-token t_i in the fully fine-tuned target model M_{FFT} (before vocabulary expansion). In SAVE influence scoring, newly added input embedding rows E_{new} are included in the vocab-aware term, while newly added LM-head rows are excluded from the influence estimator.

ToolWeaver. For ToolWeaver, we add $6{,}180$ tokens to the original `ToolWeaver-Llama-3-8B` tokenizer. Unlike the autoformalization setting, we do not retrain BPE merges. Instead, we explicitly add a pre-selected token bank tailored to the structured tool-use output format.

The added token bank consists of two parts. First, we add $2{,}100$ predefined tool-code tokens:

$$\{\texttt{<a_0>}, \ldots, \texttt{<a_1023>}\} \cup \{\texttt{<b_0>}, \ldots, \texttt{<b_1023>}\} \cup \{\texttt{<f_1>}, \ldots, \texttt{<f_52>}\}.$$

Second, we add $4{,}080$ frequency-based tokens mined from ToolWeaver end-to-end assistant loss targets. The frequency bank is extracted from $183{,}336$ ToolWeaver records and $549{,}566$ assistant-loss messages. We mine candidates only from assistant loss text. Candidate strings are required to match either the `«Name»` pattern or a `snake_case` surface form. We exclude tokens already covered by the predefined tool-code bank and remove noisy URL fragments. Among candidates with frequency at least 10 that are split into multiple pieces by the base tokenizer, we select the top $4{,}080$ by frequency.

As in autoformalization, we preserve special-token indices, resize the model embedding matrices, and initialize each newly added token from its base-token decomposition using the same AdaptiVocab-style exponential composition rule. This target-driven extension is important for SAVE. If the ToolWeaver expansion used only tool-specific code tokens, many new tokens would rarely appear in the auxiliary non-target pool. The corresponding rows in E_{new} would then receive little signal from candidate examples, and the vocab-aware term would collapse toward the LoRA/DataInf component. Mining frequent structured tokens from ToolWeaver assistant outputs makes the new-embedding signal informative for structured tool-use adaptation.

C.4 Training Hyperparameters

Autoformalization. We fine-tune `Qwen3-8B` on 50K examples sampled from `Goedel-Pset-v1`, with the tokenizer extended by $6{,}572$ target-domain tokens initialized using the AdaptiVocab-style exponential composition rule [37]. Optimization settings are summarized in Table 6.

ToolWeaver. We follow the original three-stage ToolWeaver pipeline with `Llama-3-8B` as the base model: (1) tool memorization, (2) tool retrieval, and (3) end-to-end fine-tuning. All stages share the same optimizer, learning rate, and scheduler; they differ in dataset, sequence length, and batching, as listed in Table 7.

SAVE healing stage. After vocabulary expansion, we run an additional SFT stage using the target training set and the SAVE-selected healing set. The training loss is the standard SFT cross-entropy loss, and we do not introduce a separate loss-level weight between target and healing examples. Instead, we control the contribution of each source through source-balanced mini-batches. In the main autoformalization experiments, each mixed mini-batch contains 8 examples from D_{trn} and 8 examples from D_{heal}. For ToolWeaver, SAVE is applied to the final end-to-end SFT stage using the same optimizer and training configuration as the ToolWeaver end-to-end setting.

C.5 Component-wise Analysis of Healing Data

We further analyze the auxiliary examples scored by SAVE to understand which types of non-target data are treated as useful healing data. This analysis is diagnostic only: domain labels are not used by

Table 6: **Training hyperparameters for autoformalization.**

Base model	Qwen3-8B
Hardware	4× H200
Parallelism	DeepSpeed ZeRO-3
Attention	FlashAttention-2
Precision	bf16 / tf32
Max sequence length	4096
Epochs	1
Effective batch size	64 (per-device 2 × accum. 8 × 4 GPUs)
Learning rate	5×10^{-7}
LR schedule	cosine, 3% warmup
Optimizer	AdamW (fused)
Weight decay	0.1
Max grad norm	1.0
Gradient checkpointing	enabled

Table 7: **ToolWeaver three-stage training configuration.**

	Memorization	**Retrieval**	**End-to-End**
Max sequence length	1024	1024	6144
Epochs	8	1	1
Per-device batch	2	2	1
Gradient accumulation	64	64	64
Effective batch size	1024	1024	512
Parallelism	ZeRO-2	ZeRO-2	ZeRO-3 + offload
Base model: Llama-3-8B Hardware: 8× H200			
Learning rate: 2×10^{-5}, linear schedule, 3% warmup			
Optimizer: AdamW Weight decay: 0 Max grad norm: 1.0			
Precision: bf16 Attention: FlashAttention			

SAVE during scoring or selection. Final healing data are always selected using the combined SAVE utility U_γ.

For analysis, we decompose the SAVE utility into the LoRA-side component and the vocab-aware component:

$$U_\gamma(d) = U_{\mathrm{LoRA}}(d) + \alpha_E U_{\mathrm{vocab},\gamma}(d),$$

where

$$U_{\mathrm{LoRA}}(d) = -\sum_{b\in\mathcal{B}_{\mathrm{LoRA}}} \widehat{\mathcal{I}}_b^{\mathrm{DI}}(d),$$

and

$$U_{\mathrm{vocab},\gamma}(d) = -\sum_{v\in\tau_{\mathrm{new}}(d)\cap V_{\mathrm{new}}} \widehat{\mathcal{I}}_{v,\gamma}^{\mathrm{DI}}(d).$$

The coefficient α_E is the same calibration coefficient used in the main SAVE score. U_{LoRA} reflects content-level transfer through dense LoRA blocks, while $U_{\mathrm{vocab},\gamma}$ reflects alignment with newly added target-vocabulary embeddings.

For each target setting and each auxiliary domain, we categorize candidate examples according to the sign of the corresponding component influence. An example is *helpful* for a component if its component influence is negative, equivalently if the corresponding component utility is positive. It is *harmful* if the component influence is positive. For the vocab-aware component, we additionally mark examples as *no new tokens* when they contain no expanded target-vocabulary token and therefore produce zero E_{new} gradient.

Figure 7 shows that the two components capture different transfer signals. In autoformalization, the LoRA component assigns the highest helpful fractions to `STEM` and `math` examples, followed by `code` and `tool calling`. In contrast, `chat` has the largest harmful fraction under the LoRA signal. This suggests that the content-level component favors domains with mathematical, symbolic, or reasoning structure.

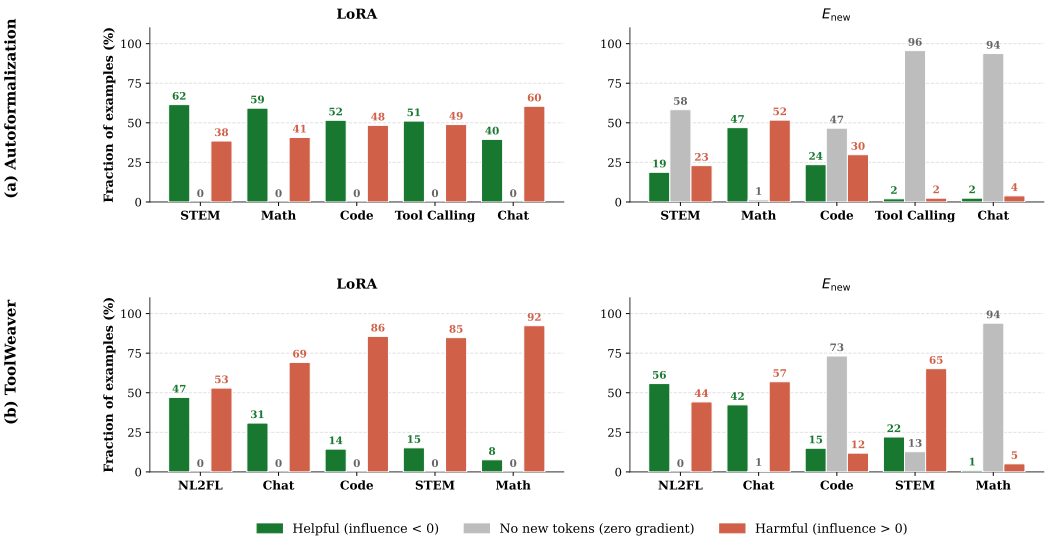

Figure 7: **Component-wise helpfulness of auxiliary domains.** For each target setting, we report the fraction of examples in each auxiliary domain categorized as helpful, harmful, or zero-gradient under the LoRA and vocab-aware components. Helpful examples have negative component influence. The zero-gradient category appears for the vocab-aware component when examples do not contain any expanded target-vocabulary tokens. LoRA blocks receive dense gradients from all examples, so the zero-gradient category is absent for the LoRA component. The label `NL2FL` denotes the autoformalization auxiliary domain. This illustrates why domain-level relevance alone is insufficient: useful healing examples must both contain transferable content and activate expanded target-vocabulary rows.

The vocab-aware component behaves differently. For autoformalization, many `tool calling` and `chat` examples contain no expanded target-vocabulary tokens, so their E_{new}-side signal is zero. Among domains that do activate new tokens, `math`, `STEM`, and `code` provide the main vocab-side signal. This supports the role of $U_{\text{vocab},\gamma}$: it does not simply duplicate the LoRA signal, but identifies examples that interact with the target-specific expanded vocabulary.

The ToolWeaver setting shows a different pattern. Under the LoRA component, autoformalization examples have the largest helpful fraction, while `math`, `STEM`, and `code` are mostly harmful. Under the vocab-aware component, autoformalization and `chat` examples provide substantial helpful signal, whereas many `code` and `math` examples contain no relevant expanded ToolWeaver tokens. This indicates that useful auxiliary data for ToolWeaver are not the same as for autoformalization; they are biased toward examples with structured responses or schema-like patterns.

Overall, this analysis supports the central design of SAVE. Useful healing data are target-dependent, and different components identify different forms of transfer. The LoRA component captures broad content-level compatibility, while the vocab-aware component captures whether an auxiliary example activates target-specific expanded tokens. Their combination allows SAVE to select auxiliary examples that are both content-compatible and vocabulary-compatible with the target task.

Schema-heavy targets. In the ToolWeaver setting, the vocab-aware signal should be interpreted as format or schema alignment rather than purely semantic tool alignment. Although SAVE excludes LM-head rows from influence scoring, response-side tokens can still induce input-embedding gradients under autoregressive teacher forcing, because tokens appearing earlier in the assistant response are used as context for predicting later tokens. Thus, repeated schema markers in structured outputs can produce strong E_{new}-side signals through the input path alone. This effect is not LM-head contamination; it reflects input-side schema alignment. This input-side effect applies only to expanded tokens that actually occur in the final scored sequence. If $\tau_{\text{new}}(d) \cap V_{\text{new}} = \emptyset$ after applying the chat template, tokenizer, and loss mask used for scoring, then $g_{d,v} = 0$ for every $v \in V_{\text{new}}$ and hence $U_{\text{vocab},\gamma}(d) = 0$. Thus, the ToolWeaver signal reported here is not evidence of gradients on absent tokens; it reflects active schema-marker rows.

C.6 SAVE Scoring and Selection Protocol

All SAVE scores are computed at the vocabulary-expanded surrogate model M_{init}. We set

$$D_{\text{score}} = D_{\text{aux}},$$

so that all candidate auxiliary examples are evaluated under a common candidate-pool curvature estimate. The target validation set D_{val} is used only to compute validation gradients for influence scoring and is not used in final SFT.

Unless otherwise stated, we use the fully token-conditional setting $\gamma = 0$. We set $\alpha_E = 10$ for autoformalization and $\alpha_E = 1$ for ToolWeaver, following the calibration in Figure 5. The SAVE influence estimate is converted into a selection utility by

$$U_\gamma(d) = -\widehat{\mathcal{I}}_{\text{SAVE},\gamma}(d),$$

so that examples estimated to reduce the target validation loss have positive utility.

For the main autoformalization experiments, we use a healing-to-target ratio of $\rho = 0.5$. Since $|D_{\text{trn}}| = 50\text{K}$, this corresponds to a maximum healing budget of

$$B_\rho = \lfloor 0.5 \cdot 50\text{K} \rfloor = 25\text{K}.$$

SAVE first filters the auxiliary pool to positive-utility examples,

$$D_{\text{pos}} = \{d \in D_{\text{aux}} : U_\gamma(d) > 0\},$$

and then selects the highest-utility examples up to the budget:

$$D_{\text{heal}} = \text{Top}_{\min(B_\rho, |D_{\text{pos}}|)}(D_{\text{pos}}; U_\gamma).$$

We denote the realized selected budget by

$$B_{\text{heal}} = |D_{\text{heal}}|.$$

If fewer than B_ρ positive-utility examples exist, SAVE uses all positive-utility examples and $B_{\text{heal}} < B_\rho$. All budget-matched baselines use the same realized budget B_{heal}.

For ToolWeaver, we use the same scoring protocol with D_{val} drawn from the ToolWeaver target validation split and D_{aux} excluding ToolWeaver target examples. In the fixed-budget variant, we replace an equal number of ToolWeaver target examples with SAVE-selected healing examples, keeping the total number of final-SFT examples unchanged.

C.7 Hyperparameter Recipe

SAVE has four knobs: the vocabulary weight α_E, the interpolation parameter γ, the healing budget ratio ρ, and the damping λ_v. In all reported settings we use the following recipe, which requires no search beyond α_E:

- $\gamma = 0$ (fully token-conditional normalization).
- $\rho = 0.5$ (at most one healing example per two target examples).
- λ_v and the LoRA-side damping inherit the DataInf configuration [27] unchanged.
- α_E is initialized from the relative scale of the two components, e.g., the ratio of the median absolute LoRA-side utility to the median absolute vocabulary-side utility over the candidate pool. The observed scale ratio is $|I_{\text{LoRA}}|/|I_{\text{vocab},\gamma}| \approx 22$ for autoformalization and close to 1 for ToolWeaver, matching the best values $\alpha_E \approx 10$ and $\alpha_E \approx 1$ in Figure 5.

Further tuning of α_E is optional: the target metrics follow an inverted-U shape, and any value well below 50 improves over $\alpha_E = 0$ in our sweeps (Figure 5).

C.8 Scoring Cost

On the same 100K-example auxiliary pool and a 4×H200 setup, SAVE required $1.37\times$ the GPU-hours of vanilla DataInf for the scoring stage. Since both methods use the same fixed parallelism, this corresponds to approximately $1.37\times$ wall-clock time, i.e., a 37% scoring overhead. The additional

cost comes from extracting gradients for active expanded-token rows and constructing their token-conditional preconditioners. We observed no measurable increase in peak GPU memory relative to DataInf: the dominant memory cost remains the shared model and the LoRA-side influence computation, while the additional vocabulary statistics are restricted to active expanded-token rows. Scoring is a one-time data-selection cost; the final SFT configuration and training cost are identical across selectors. EmbSim and New-token TF-IDF avoid gradient and curvature computation and are therefore cheaper, but their lower downstream performance (Table 2) reflects the corresponding compute–performance trade-off. The pair-block variant in Appendix A.6 costs $1.22\times$ SAVE.

C.9 Final SFT Configurations and Baselines

All final autoformalization SFT configurations start from the same vocabulary-expanded model M_{init}. The configurations differ only in the additional data used during the final SFT stage.

Target-only. This baseline trains from M_{init} using only the target training set D_{trn}.

Target + Random. This baseline samples B_{heal} examples uniformly from D_{aux} and trains on

$$D_{\text{trn}} \cup D_{\text{random}}, \qquad |D_{\text{random}}| = B_{\text{heal}}.$$

Target + LESS. LESS [52] is used as a target-aware instruction-data selection baseline. It selects B_{heal} examples from the same auxiliary pool according to its gradient-based relevance score.

Target + OGS. OGS [59] is used as a retention-aware domain-adaptation baseline. It selects B_{heal} examples using its gradient-orthogonality criterion. The general-capability anchor is constructed from held-out auxiliary data and does not include any evaluation benchmark examples.

Target + DataInf. The vanilla DataInf baseline [27] applies the standard block-wise DataInf estimator without SAVE's token-conditional normalization for newly added embedding rows. This baseline tests whether ordinary target-aware influence estimation is sufficient in the vocabulary-expanded setting.

Target + All-Inst. All-Inst concatenates the full auxiliary pool with the target training data. It is not budget-matched to SAVE; instead, it tests whether naively using all available auxiliary data is sufficient without selection.

Target + SAVE. SAVE trains on

$$D_{\text{SFT}} = D_{\text{trn}} \cup D_{\text{heal}}.$$

We do not use response-side distillation in the main experiments; selected auxiliary examples are trained with their original responses.

All final models are trained with the standard SFT cross-entropy loss. Rather than applying a separate loss-level weight, we construct each mini-batch with a fixed source composition, matching the description in Appendix A.8.

Attribution controls (Table 2). All controls use the same realized budget B_{heal}, auxiliary pool, and final-SFT configuration as SAVE. EmbSim embeds each auxiliary instruction with `all-mpnet-base-v2` [40, 44] and selects the examples with the highest cosine similarity to the centroid of the target validation set. New-token TF-IDF [42] scores each auxiliary example by the TF-IDF-weighted overlap between its expanded tokens and those of the target validation set, down-weighting ubiquitous expanded tokens. SAVE-Mix Random fixes the per-domain composition of SAVE's healing set and samples instances uniformly at random within each domain. SAVE-Mix DataInf fixes the same per-domain composition but ranks instances within each domain by vanilla DataInf. SAVE-FO keeps the SAVE score but replaces the token-conditional preconditioner $\widehat{M}_{v,\gamma}$ with $\lambda_v^{-1} I$, removing token-conditional curvature.

Generalization settings (Table 3). For `Qwen3-4B`, we repeat the autoformalization pipeline with the same target data, vocabulary-expansion procedure, auxiliary pool, and selection hyperparameters. For the medical target, we use `Llama-3.1-8B-Instruct` [14], randomly sample 50K target training examples from UltraMedical [58], apply the same vocabulary-expansion procedure as in the autoformalization experiment, and retain the balanced 100K auxiliary pool. We evaluate on MedQA [22]. In both settings we compare against Target-only and DataInf, the strongest selector in Table 1.

For ToolWeaver, the TOOLWEAVER+SAVE setting adds the selected healing data to the ToolWeaver final SFT stage. The fixed-budget variant uses the same selected healing examples but replaces an equal number of ToolWeaver target examples, keeping the total final-SFT data budget matched to the original ToolWeaver setting.

C.10 Evaluation Protocols

miniF2F. We evaluate autoformalization target performance on the miniF2F test split. Compile Rate measures the fraction of generated Lean outputs accepted by the Lean compiler. Accuracy measures semantic correctness of the generated theorem statement using the LLM-as-judge protocol in Figure 10. Non-compiling outputs are counted as incorrect for accuracy. The judge is not an unconstrained LLM judgment: it receives the informal problem, the ground-truth Lean statement, and the generated Lean statement, and compares their conclusions, assumptions, quantifier structure, variable domains, and potential vacuity. Similar reference-anchored semantic verification is used in prior formal-reasoning work, e.g., the faithfulness and completeness check of Goedel-Prover [33]. The ordering between SAVE and the selection baselines is the same under the objective compile rate and the judged semantic accuracy (Tables 1 and 8).

MedQA. For the medical target, we report multiple-choice accuracy on the MedQA test set [22].

AIME2025. For AIME2025, we sample 64 independent responses per problem and report Pass@1 as the average single-sample accuracy across these generations:

$$\text{Pass@1} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{64} \sum_{r=1}^{64} \mathbf{1}[\text{ans}(\hat{y}_{q,r}) = y_q].$$

This is not Pass@64; each sampled response is evaluated as an independent single attempt and then averaged.

LiveCodeBench. For LiveCodeBench, we sample 10 independent responses per problem and report Pass@1 analogously:

$$\text{Pass@1} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{10} \sum_{r=1}^{10} \mathbf{1}[\text{pass}(\hat{y}_{q,r}) = 1].$$

MMLU-Pro and IF-Eval. We report MMLU-Pro and IF-Eval using their standard evaluation protocols.

ToolWeaver retrieval. For ToolWeaver, we report NDCG@1, NDCG@3, and NDCG@5 across query complexities I1–I3, following the ToolWeaver and ToolGen evaluation protocol. Starred rows in the ToolWeaver table denote reported ToolWeaver reference baselines.

Evaluation-only use. All evaluation benchmarks, including miniF2F, AIME2025, MMLU-Pro, LiveCodeBench, IF-Eval, ToolBench, and MedQA, are used only for post-hoc evaluation. They are not used for influence scoring, auxiliary-data selection, vocabulary construction, or model selection.

C.11 Seed Variability

To assess run-to-run variability, we repeat the final SFT stage for DataInf and SAVE with three training seeds, fixing the selected healing sets, hyperparameters, number of updates, and evaluation protocol. Table 8 reports the mean and standard deviation; Table 1 reports single runs.

Table 8: **Three-seed results on autoformalization** (mean $\pm$ std). SAVE outperforms DataInf in every matched run on all three metrics.

Method	miniF2F Compile	miniF2F Accuracy	AIME2025
DataInf	0.7158 ± 0.0103	0.6311 ± 0.0082	0.7206 ± 0.0078
SAVE	**0.7909** ± 0.0083	**0.7213** ± 0.0082	**0.7346** ± 0.0019

Averaged over the matched runs, SAVE improves over DataInf by 7.51 points in compile rate, 9.02 points in semantic accuracy, and 1.40 points on AIME2025. Because AIME2025 contains only 30 problems, we interpret the AIME2025 result as a consistent point-estimate improvement rather than making a formal significance claim.

D Prompts

System and User Prompts for Autoformalization

[System Prompt]

```
You are an expert Lean 4 formalizer. Convert the given natural-language math
problem into Lean 4 code that compiles with Mathlib. Return ONLY Lean 4 code
(no extra explanations).
```

[User Prompt]

```
Please autoformalize the following natural language problem statement in Lean 4.
Use the following theorem name: {decl_name} The natural language statement is:
{informal} Return ONLY Lean 4 code.
```

Figure 8: **Prompt template used for the autoformalization target task.** `{decl_name}` and `{informal}` are filled with the target Lean declaration name and the informal problem statement, respectively.

System Prompt for End-to-End Tool Use (ToolGen)

```
You are an AutoGPT, capable of utilizing numerous tools and functions to complete the given task.

1. First, I will provide you with the task description, and your task will commence.

2. At each step, you need to determine the next course of action by generating an action token.

3. Following the token, you will receive the documentation of the action corresponding to the token.
You need to generate the input of the action, transitioning you to a new state. Subsequently, you
will make decisions about the next steps, and repeat this process.

4. After several iterations of generating actions and inputs, you will ultimately complete the task
and provide your final answer.

Remember:

1. The state changes are irreversible, and you cannot return to a previous state.

2. Keep your actions concise, limiting them to best suits the current query.

3. You can make multiple attempts. If you plan to try different conditions continuously, perform
one condition per try.

4. If you believe you have gathered enough information, generate the action «Finish» with argument
give_answer to provide your answer for the task.

5. If you feel unable to handle the task from this step, generate the action «Finish» with argument
give_up_and_restart.

Let's Begin!

Task description: You should use actions to help handle the real time user querys.

Remember:

1. ALWAYS generate «Finish» at the end of the task. And the final answer should contain enough
information to show to the user. If you can't handle the task, or you find that actions always fail
(the function is not valid now), use action «Finish» with give_up_and_restart.

2. Only generate actions and inputs.
```

Figure 9: **System prompt used for end-to-end tool-use evaluation under the ToolGen protocol.** The agent invokes tools by generating action tokens and terminates the trajectory with the special `«Finish»` action.

Judge Prompt for Autoformalization Evaluation

You are a careful Lean 4 evaluator for autoformalization of mathematical problems. Given a ground-truth formalization and a model-generated formalization, decide whether the two represent the same mathematical problem.

In autoformalization, a problem is encoded as a Lean 4 theorem statement: a set of hypotheses and a conclusion, with the proof left as sorry. Your job is to compare the two problem statements as mathematical propositions, not the syntactic surface or any proof. The informal description is provided as context for understanding the intended problem, but the primary judgment is between the two Lean statements.

Inputs

- INFORMAL: natural-language description of the problem.
- FORMAL_GT: the reference Lean 4 problem statement (ground truth).
- FORMAL_GEN: the model-generated Lean 4 problem statement.

You may assume both statements type-check; your task is *semantic* equivalence as problem statements, not syntactic identity.

Equivalence Criteria

FORMAL_GEN encodes the same problem as FORMAL_GT if and only if all of the following hold:

1. Same conclusion. The proposition the theorem asserts is the same. a = b is not equivalent to a $\leq$ b. $\exists$ x, P x is not equivalent to $\forall$ x, P x. Logically equivalent reformulations (e.g., Even n $\leftrightarrow$ $\exists$ k, n = 2 * k) are fine.
2. Same hypotheses. Every assumption in FORMAL_GT appears in FORMAL_GEN, and vice versa, up to logical equivalence. Hypotheses that are mere type ascriptions (e.g., n : $\mathbb{N}$) and typeclass instance arguments do not count as added assumptions.
3. Same quantifier structure. $\forall$ x $\exists$ y, P x y and $\exists$ y $\forall$ x, P x y are different problems.
4. Same domains. Variables must range over the intended set. (x : $\mathbb{Z}$) vs (x : $\mathbb{N}$) is a domain mismatch when negativity matters; (x : $\mathbb{R}$) vs (x : $\mathbb{Q}$) is a mismatch when irrationality matters. When the problem is naturally domain-agnostic and both choices give the same answer set, treat them as equivalent.
5. Non-trivial. A statement that is vacuously true--mutually contradictory hypotheses, or False smuggled in as a hypothesis--is not equivalent to a non-vacuous ground truth, even if both type-check.
6. No goal/hypothesis swap. Moving the conclusion into the hypotheses or vice versa changes the problem.

The following are NOT grounds for non-equivalence:

- Renamed bound or free variables.
- Definitionally equal terms (n + 1 $\leftrightarrow$ Nat.succ n, 2 * n $\leftrightarrow$ n + n).
- Reordered commutative operands (a + b $\leftrightarrow$ b + a).
- Logically equivalent reformulations of the same proposition.
- Stylistic differences: explicit vs implicit arguments, theorem name, presence/absence of by sorry vs := sorry.
- Extra Lean-mechanical hypotheses required only for typeclass resolution that do not change the mathematical content.

Output Format

Return only a JSON object, no surrounding prose:

```
{
  "reasoning": "<2-4 sentences comparing GT and GEN: what each problem asserts, what assumptions each carries, and where they agree or differ>",
  "verdict": "<equivalent | not_equivalent>"
}
```

Now Evaluate

INFORMAL:

{informal_statement}

FORMAL_GT:

{formal_gt}

FORMAL_GEN:

{formal_gen}

Figure 10: **LLM-as-judge prompt for autoformalization evaluation.** The judge compares a ground-truth Lean 4 statement (FORMAL_GT) and a model-generated statement (FORMAL_GEN) as mathematical propositions and returns a structured JSON verdict on semantic equivalence.

E Limitations

SAVE is designed for vocabulary-expanded adaptation. Our experiments cover two vocabulary-sensitive targets, Lean autoformalization and tool calling, with 8B-scale backbones, together with a smaller backbone (`Qwen3-4B`) and a less syntactically rigid medical target (MedQA). Further work is needed to verify whether the same trends hold for larger models and other tokenizer-expansion strategies such as multilingual adaptation.

Our estimator also relies on several approximations. SAVE computes influence on a surrogate vocabulary-expanded model and treats the resulting score as a candidate-pool selection criterion rather than an exact estimate of the final SFT objective. The token-conditional vocabulary term uses block-wise curvature approximations and ignores cross-token curvature among newly added embedding rows. This approximation is most reliable when expanded tokens have limited co-occurrence or weak cross-row gradient correlation. In our setting, retaining cross-token curvature for 256 high-co-occurrence pairs changed the selected data and the target metrics only marginally (Appendix A.6), but larger blocks may matter when many new tokens appear together systematically.

The vocab-aware signal requires that the candidate corpus provide nonzero support for the expanded tokens. SAVE is designed for the sparse regime, where full-corpus normalization would otherwise suppress rare tokens, but if an expanded token never appears in the pool, no selector can retrieve an example that updates its row. In that case SAVE does not degenerate pathologically: it falls back to its dense LoRA-side component, and the appropriate remedy is a broader candidate corpus. Rare expanded tokens can also yield high-variance token-conditional curvature estimates, requiring sufficient damping or minimum-support filtering. Conversely, because the vocabulary utility sums over activated tokens, it moderately favors examples with more expanded-token occurrences (Appendix A.5); normalization or clipping can be used under extreme frequency imbalance.

Finally, SAVE introduces calibration choices, including the vocabulary weight α_E, the token-conditional interpolation parameter γ, and the healing budget ρ. We keep $\gamma = 0$ and $\rho = 0.5$ fixed and initialize α_E from the component-scale ratio (Appendix C.7), but these choices may need recalibration for substantially different targets or auxiliary pools. Our evaluation also inherits the limitations of the benchmarks used: AIME2025 contains only 30 problems, and miniF2F accuracy relies on semantic comparison of generated Lean statements. We therefore treat these results as evidence of the proposed mechanism rather than as a complete characterization of all vocabulary-expanded adaptation regimes.

F Safeguards and Broader Impacts

This work studies a data-selection method for improving specialized LLM adaptation. Potential benefits include more efficient adaptation to structured domains, better formal-mathematics generation, and more reliable tool-use behavior. At the same time, stronger tool-use models may increase the capability of systems that interact with external APIs, and improved domain adaptation methods could be applied to sensitive or harmful domains. Deployment should therefore include task-specific safety evaluation, access control for external tools, and monitoring of generated actions.

SAVE selects examples from a generic auxiliary pool, so its behavior depends on the quality and safety of that pool. If the pool contains private, copyrighted, biased, or unsafe content, selection may surface such examples when they appear useful for the target validation objective. Practical use should therefore include data filtering, license checks, deduplication against evaluation sets, and auditing of selected healing examples before training. In our experiments, evaluation benchmarks are used only for post-hoc evaluation and not for influence scoring, data selection, or model selection.

We do not introduce a new user-facing dataset of personal information or conduct human-subject experiments. If model checkpoints or selected healing datasets are released, they should be accompanied by documentation describing the source data, filtering procedure, intended use, and known limitations. For tool-use models in particular, downstream systems should restrict available tools, validate tool arguments, and prevent execution of unsafe or unauthorized actions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main claims, which are supported by the main experiments and ablations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not present formal theorems; estimator assumptions and derivations are provided in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Models, datasets, training details, baselines, and evaluation protocols are specified in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code is not yet public at the time of the camera-ready submission; the modularized scoring and selection implementation will be released at https://github.com/DopeorNope-Lee/SAVE. All settings needed to reproduce the results are described in Appendix C.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Experimental settings and hyperparameters are provided in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report mean $\pm$ standard deviation over three training seeds for DataInf, SAVE, and SAVE-PairFull (Appendix C.11 and Appendix A.6); the remaining baselines are single runs. Given the 30-problem size of AIME2025, we do not make formal significance claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We report hardware and training settings (Appendix C) and the relative scoring cost and peak-memory behavior of SAVE versus DataInf and SAVE-PairFull (Appendix C.8), but not absolute GPU-hours for every stage.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses public models, datasets, and benchmarks and does not involve human-subject experiments.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader impacts are discussed in Appendix F.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release a new high-risk model or dataset; safeguards for downstream use are discussed in Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models, datasets, and benchmarks are cited, and their URLs, licenses, roles, and sample counts are listed in Appendix C.1 and Appendix C.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release a new dataset, benchmark, or model checkpoint.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are used as the adapted models and for the autoformalization judge.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.