TOWARDS ACTIVE SYNTHETIC DATA GENERATION FOR FINETUNING LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

A common and effective means for improving language model capabilities involves finetuning a "student" language model's parameters on generations from a more proficient "teacher" model. Termed "synthetic data", these generations are often produced before any student finetuning, but some work has considered generating new synthetic samples as training progresses. This paper studies and advocates for the latter case, where data are generated in an iterative, closed-loop fashion that is guided by the current state of the student model. For a fixed budget of generated samples, or a budget in terms of compute spent querying a teacher, we show that this curation of finetuning data affords improved student performance over static generation. Further, while there have been several LLM-specific methods proposed that operate in this regime, we find that simple, inexpensive selection criteria from the active learning literature tend to be most performant. We validate these claims across four mathematical and logical reasoning datasets using four different small language models.

1 INTRODUCTION

Large Language Models (LLMs) have shown remarkable abilities in a wide variety of reasoning and factual knowledge tasks (Achiam et al., 2023; Bubeck et al., 2023; Katz et al., 2024), but their large size makes inference expensive. With the advent of agentic systems that interact with the external world, LLMs are poised to become even more ubiquitous in science, technology, and society in general. However, the tremendous inference cost presents a challenge for realizing the full potential of these agents.

One way to quell the computational expense associated with LLM inference is to use small language models (SLMs). With orders of magnitude fewer parameters, SLMs are faster, cheaper, and easier to finetune for specialised skills like tool use or interface alignment, making them natural specialists within agentic systems (Belcak et al., 2025).

Training language models typically involves three stages: pre-training on large general-purpose corpora, supervised finetuning (SFT), and reinforcement learning from human feedback (RLHF) or from verifiable rewards (RLVR) (Ouyang et al., 2022). SFT, the focus of this work, is critical for adapting a base model to the target distribution before reinforcement learning, and is especially common when training SLMs to improve their task-specific performance.

However, real-world data for SFT can be hard to obtain, or may lack desirable properties such as chain-of-thought reasoning (Wei et al., 2022). Consequently, a common and effective strategy involves synthesizing a corpus of prompts and corresponding responses from a larger, more capable model (Mitra et al., 2024; Liu et al., 2024). This process typically begins with a small seed dataset and leverages a teacher LLM to produce supplementary synthetic samples, then finetunes the student SLM on the resulting sequences in aggregate.

Yet, evidence suggests that generating a large static synthetic dataset is often wasteful, as it can often be drastically pruned with little to no degradation in trained model capabilities (Chen et al., 2023; Zhou et al., 2024). As such, this paper explores an iterative, targeted approach to synthetic data generation that is student-aware and improves data efficiency—achieving stronger performance under a fixed data generation budget than naive static generation—thereby yielding a superior performance–training-set-size Pareto frontier (see Section 2 for a formal definition).

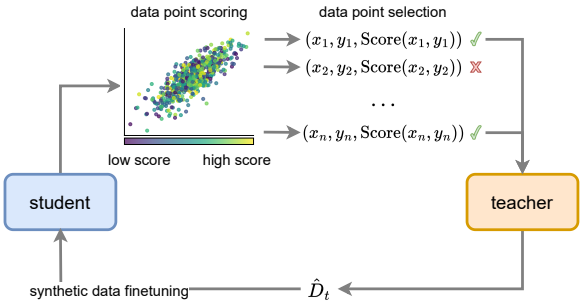

Figure 1: **Overview of the iterative synthetic data generation (Algorithm 1)**. The student model guides synthetic data generation by prioritizing which data are used as an example for the teacher model to generate a new synthetic data point (Section 4.2). The student finetunes on synthetic data generated by the teacher.

To facilitate productive learning, this work studies how we can effectively cater to the state of the student model and guide synthetic data generation by a teacher LLM via prompting (Mitra et al., 2024; Liu et al., 2024; Luo et al., 2023). This results in an iterative scheme, where the updated student can be reused to guide further teacher-generated samples (Figure 1). Prior work has considered this paradigm by prioritizing incorrect student answers (Lee et al., 2024) and using `LLM-as-a-judge` scoring (Jiang et al., 2023c), but they do not draw upon the vast active learning and data selection literature. Instead, this paper advocates for the generation of data that is conditioned on samples that have been prioritized by an active learning algorithm. The resulting dataset enables more effective and data efficient finetuning of the SLM student model (see Section 5.4 for evidence supporting this claim).

Our work makes the following contributions:

- We provide a benchmark study for iterative synthetic data generation rooted in prior work on active learning and data selection. We carefully compare to static dataset generation—identical to random sampling—to show improvements in data efficiency.
- We compare a range of methods for prioritizing synthetic data generation, including uncertainty sampling, diversity selection, and selection of difficult/easy samples. We conclude that simple methods rooted in active learning, such as using the loss of the student's prediction (Settles & Craven, 2008), are the most data efficient to prioritize the creation of synthetic data and enables strong performance for a fixed training set size. In contrast, expensive and popular methods based on using an LLM to judge the difficulty and quality of data, i.e. `LLM-as-a-judge` (Zheng et al., 2023; Jiang et al., 2023c), underperform active learning counterparts.
- We obtain state-of-the-art capabilities for SLMs by applying active synthetic data generation and performing SFT on a fixed training data budget.

2 Preliminaries

Notation. We use i to index a datapoint in a dataset, t to index the iteration of iterative synthetic data generation, and j to index token position in a sequence. We denote question and answer pairs $z = (x, y)$, from a dataset of size n drawn from a ground truth distribution P: $D_0 = \{z\}_{i=1}^{n} \sim P$. We use the terms "question" and "instruction" interchangeably for x, and "answer" and "response" interchangeably for y. The rationales or chain-of-thought (Wei et al., 2022) are incorporated into the answers $y := [r, y]$. For some datasets, there is no chain-of-thought $y := [\text{“”}, y]$. A model $f_{\boldsymbol{\theta}}(\cdot)$ with parameters $\boldsymbol{\theta}$ generates an answer $\hat{y}$ given a question x: $\hat{y} = f_{\boldsymbol{\theta}}(x)$. Synthetic questions and answers are denoted $\hat{z} = (\hat{x}, \hat{y})$. Text is encoded into tokens, we denote V as the vocabulary and each token is an indicator vector $\{0, 1\}^{|V|}$. SFT involves minimizing the next token prediction loss, the cross-entropy, over answer tokens given a question: $\mathcal{L}(z, \boldsymbol{\theta}) = -1/|y| \sum_{j=1}^{|y|} y_j \log f_{\boldsymbol{\theta}}(x, y_{<j})$. The model $f_{\boldsymbol{\theta}}(\cdot)$ autoregressively generates the next token $\hat{y}_j = f_{\boldsymbol{\theta}}(x, \hat{y}_{<j})$ in the sequence.

Data Efficiency. At a high level if we can get a better performance with one dataset versus another then the former is more data efficient. Formally, let P be the true data distribution over our data $z = (x, y)$. For a selection algorithm ϕ that produces a dataset $S_n^{\phi} = \{z_i\}_{i=1}^{n} \overset{\phi}{\sim} P$ then the model parameters $\boldsymbol{\theta}_n^{\phi}$ results from minimizing the loss over S_n^{ϕ}. We define the performance, accuracy for

example, for a single data point as:

$$\text{perf}_\phi(z, \boldsymbol{\theta}_n^\phi) = \mathbf{1}\left\{y = f_{\boldsymbol{\theta}_n^\phi}(x)\right\}, \tag{1}$$

and the expected performance is:

$$\text{perf}_\phi(n) = \mathbb{E}_{z\sim P}\mathbb{E}_{S_n^\phi \sim P}\left[\text{perf}_\phi(z, \boldsymbol{\theta}_n(S_n^\phi))\right]. \tag{2}$$

Assuming a monotonic increase in performance with n, for some target performance τ, the sample complexity is:

$$N_\phi(\tau) = \inf\left\{n : \text{perf}_\phi(n) \geq \tau\right\}, \tag{3}$$

which measures the smallest n such that $\text{perf}_\phi(n) \geq \tau$. For a fixed architecture $f(\cdot)$, algorithm α is more data-efficient than algorithm β at level τ only if $N_\alpha(\tau) < N_\beta(\tau)$ or if for a fixed n then $\text{perf}_\alpha(n) > \text{perf}_\beta(n)$.

3 Related Work

Distillation. Fitting models on synthetic datasets composed of pairs $z = (x, \hat{y})$ of sequences $\hat{y}$ produced by a teacher but conditioned on separately available prompts x—often referred to as distillation (Hinton, 2015)—has been shown to be extremely effective in improving capabilities of SLM student models (Taori et al., 2023; Peng et al., 2023; Team et al., 2024).

Synthetic question and answer generation. Going one step further, we can generate *both* questions *and* answers: $\hat{z} = (\hat{x}, \hat{y})$. SFT on synthetic question-answer pairs results in improved capabilities without being restricted by potentially small seed dataset sizes (Mukherjee et al., 2023). Much like in the distillation setting, generating question-answer pairs only requires prompting the teacher model with a seed data point (Mitra et al., 2024; Liu et al., 2024; Luo et al., 2023; Zeng et al., 2024).

Selective question and answer generation. Synthetic datasets are known to be compressible—synthetic samples filtered to have high `LLM-as-a-judge` (Chen et al., 2023) values or low student loss (Li et al., 2023) can obtain the same performance as finetuning on the entire unpruned corpus. To remedy this inefficiency, rather than generating a large static synthetic dataset and then filtering, we can instead carefully select the seed data used to generate the synthetic samples, and hopefully produce fewer semantically similar sequences. This has been shown by prioritizing incorrect samples when querying the teacher, which has been shown to be more data efficient than finetuning on the original corpus D_0 (Lee et al., 2024). Also, `LLM-as-a-judge` selection is more data efficient than directly finetuning on a public benchmark synthetic datasets (Jiang et al., 2023c). In this work we study `LLM-as-a-judge` coring due to its widespread use.

3.1 Assigning a value to data

Active learning. Our work is inspired by ideas from active learning, which seeks to maximise data efficiency by iteratively identifying and prioritising informative samples for labelling (Settles, 2009; Settles & Craven, 2008). Classic strategies for active learning include model prediction disagreement (Freund et al., 1997; Houlsby et al., 2011), uncertainty (MacKay, 1992; Gal et al., 2017; Kirsch et al., 2019), and dataset summarization (Sener & Savarese, 2017; Mirzasoleiman et al., 2020; Coleman et al., 2019). Some popular methods for selecting diverse samples in active learning include determinantal point processes (Kulesza & Taskar, 2011) and BADGE which uses gradient embeddings together with the `kmeans++` algorithm to trade-off between predictive uncertainty and sample diversity (Ash et al., 2019). We consider language model-aligned variations of two popular methods for active learning: uncertainty sampling (Settles & Craven, 2008), a classic approach that favors predictive uncertainty, and a more contemporary approach, BADGE.

Data selection. Related methods aim to estimate the value of data to guide selection, typically using labelled dataset (x, y). LLMs have been used to score data points (Zheng et al., 2023) and for selecting question-answer samples for SFT (Liu et al., 2023; Jiang et al., 2023c; Chen et al., 2023). Still, it has been shown that LLMs scores exhibit biases that hinder their effectiveness

Algorithm 1 Iterative synthetic data generation algorithm for question and answer datasets.

Input: Seed dataset D_0, test set D_{test}, train set $\hat{D}_{-1} = \{\}$, student $f_{\boldsymbol{\theta}}(\cdot)$, selection algorithm ϕ.

1: **for** $t = 0, \ldots, T$ **do**
2: Generate SLM predictions on D_t: $\{z_i = (x_i, \hat{y}_i)\}_{i=1}^n$ where $x_i \in D_0$ and $\hat{y} = f_{\boldsymbol{\theta}}(x)$.
3: Score and select data: $\bar{D}_t = \phi(D_t)$. ▷ See Section 4.1 for details.
4: Generate synthetic dataset: $\hat{D}_t = \text{Generate}(\bar{D}_t)$. ▷ See Section 4.2 for details.
5: SFT on $f_{\boldsymbol{\theta}}(\cdot)$ using $\hat{D}_t := \hat{D}_t \cup \hat{D}_{t-1}$ and evaluation on D_{test}.
6: **end for**

in this setting (Xiong et al., 2023; Dorner et al., 2024; Panickssery et al., 2024). Alternative approaches use training loss or gradient norms with respect to student parameters to estimate learning progress (Loshchilov & Hutter, 2015; Katharopoulos & Fleuret, 2018; Jiang et al., 2019; Li et al., 2023; Mindermann et al., 2022; Evans et al., 2024; Dai et al., 2025). However, this has shown limited data efficiency for language models (Kaddour et al., 2023). Reward models are commonly used to score and identify data points for SFT (Cao et al., 2023; Dubey et al., 2024). In this work, we focus on reward selection due to its popularity.

4 Iterative Synthetic Data Generation

The general iterative synthetic data generation process studied in this paper is shown in Algorithm 1 (Jiang et al., 2023c; Lee et al., 2024). We expand the algorithm's design choices in the next sections. The selection algorithm ϕ described in Section 2 can be composed into a scoring and selection functions with the exception of BADGE, described below.

We first obtain a score for each data point $\{s_i\}_{i=1}^n$ where $n = |D_0|$ which indicates the data's importance. Secondly, we select $m = |\bar{D}_t|$ points with the highest scores, "argmax" selection $\bar{D}_t = \text{argmax}_m \{s_i\}_{i=1}^n$. Or we select m points by "sampling" according to their scores: $\bar{D}_t \overset{m}{\sim} \text{softmax}(\{s_i\}_{i=1}^n)$. Alternatively, we can pick the lowest n scoring data points by negating the scores. In the next section we consider various algorithms for scoring data. By using these together with argmax selection or sampling we can construct a selection algorithm, ϕ.

4.1 Selection Algorithms

Uncertainty sampling. A common method in the active learning literature is uncertainty sampling, which, for non-sequential classification models, prioritize data for which the amount of probability mass on the most likely class predicted by the model is smallest. In the sequential, Transformer-based analogue, we can score a data point with the loss of the answer tokens under the student $f_{\boldsymbol{\theta}}(\cdot)$ with parameters $\boldsymbol{\theta}$: as $\mathcal{L}(z_i, \boldsymbol{\theta})$. When the targets used to produce a loss are the model's own generations, this score reflects a sort of uncertainty in the sequence produced. Note that our setting gives us access to the ground-truth label associated with x as well, and thus allows us to compute a true loss here in a fashion commensurate with conventional model training. Interestingly, we find empirically that this is less effective than using the former, uncertainty approach.

Reward scores. Using the student's own generated sequence $\hat{y}$, a common method for scoring data is to obtain a prediction from a separate reward model $r(x, \hat{y})$. Resulting scores can be interpreted as the quality of the student's response, and indicative of its competence on questions of this sort in general. One is not limited to using the student's predictions but can also obtain a reward for the ground truth answer y. In this manner, rewards will encapsulate the difficulty of the data and the informativeness of y. That said, scoring data with $r(x, y)$ removes any dependence on the student model and thus not desirable. Additionally, we find that this empirically underperforms using $r(x, \hat{y})$. One drawback of this approach is that it requires an external model for scoring, however we can obtain competitive reward models of the same size as our student and so we do not treat a call to our reward model as being on the same level as a call to our teacher model.

`LLM-as-a-judge` scores. We can also leverage the reasoning ability of an LLM teacher model to score an SLM's predictions. We can ask the LLM teacher to score the detail, quality and cor-

rectness of the student answer and reasoning steps with a number between $[1, 10]$. In particular, we use pairwise `LLM-as-a-judge` scoring which has been shown to be most effective (Zheng et al., 2023). Two separate answers are given for the teacher to decide which it prefers by providing scores for both: $s_i^t, s_i = \texttt{LLM}(\hat{y}_i^t, \hat{y}_i, x_i)$ where $\hat{y}_i^t = \texttt{LLM}(x_i)$ is teacher's answer, s_i^t is the score for the teachers answer and $\hat{y}_i$ the student answer. This is an expensive scoring method since it requires the teacher to produce an answer in addition to scoring.

BADGE. Batch Active learning by Diverse Gradient Embeddings (BADGE) is a two-stage active learning algorithm. It first represents all candidate data using the last-layer gradient of the loss induced by treating the generated sequence as ground truth, $\nabla_{\boldsymbol{\theta}_o} \mathcal{L}(\hat{y} = f_{\boldsymbol{\theta}}(x)$, where $\boldsymbol{\theta}_o$ are output-head parameters. In the second stage, BADGE approximately samples from a k-DPP to identify gradients that are both high-magnitude and diverse (note that high-magnitude gradients are high-loss generations, suggesting high predictive uncertainty). Like in uncertainty sampling, our setting allows us to use ground-truth target sequences—which would make these gradient representations of the sort that might be used during optimization—but we empirically find it is higher performing to use generated sequences instead. Because the un-embedding layer of a Transformer is typically extremely large, we use a sparse random projection to efficiently reduce dimensionality while preserving important relationships (Johnson et al., 1984)

4.2 Prompt-based synthetic data generation

Selected data points $\bar{x}_i \in \bar{D}_t$ are added to a synthetic data generation prompt for the LLM teacher model to generate a synthetic question $\hat{x}_i$ (Xu et al., 2023; Mitra et al., 2024; Jiang et al., 2023c; Lee et al., 2024). Then the teacher model is prompted to produce chain-of-thought reasoning and a final answer for $\hat{x}_i$. We generate a synthetic data point $\hat{z}_i = (\hat{x}_i, \hat{y}_i)$ using $\hat{x} = \texttt{LLM}(\bar{x}_i)$ and $\hat{y}_i := [\hat{r}_i, \hat{y}_i] = \texttt{LLM}(\hat{x}_i)$. So $\hat{D}_t = \text{Generate}(\bar{D}_t) = \{\hat{x}_i = \texttt{LLM}(\bar{x}_i), \hat{y}_i = \texttt{LLM}(\hat{x}_i)\}_{i=1}^m$ where $\bar{x}_i \sim \bar{D}_t$. For the further details on the implementation for the individual datasets see Appendix E.2.

5 Experiments

This section empirically probes the data efficiency of iterative synthetic data generation against static data generation. We also provide recommendations to practitioners regarding which scoring and selection design choices seem to improve efficiency; **we find that prioritizing high difficulty data using a high student loss to be the most data efficient**. High uncertainty sampling equivalent to a high loss under the student's own predictions **results in state-of-the-art capabilities on all datasets** when considering comparable SFT methods and in some cases using orders of magnitude less training data (Section 5.4.3).

At each iteration we value each data point in the seed dataset training split and prioritize 1k samples: $\bar{D}_t = \phi(D_t)$. The teacher then generate 1k synthetic data points and we append this data set to synthetic datasets from previous iterations for SFT.

5.1 Datasets

This section presents results on four distinct reasoning datasets in conjunction with four different models. `GSM8k` is a popular mathematics dataset comprised of school level maths problems (Cobbe et al., 2021), which we use in conjunction with a `Mistral-7B-Instruct-v0.3` student (Jiang et al., 2023a). Similarly, we include the more challenging `Math1-3` dataset (Hendrycks et al., 2021), which is separated into 5 distinct levels of question difficulty—we use the easiest levels, 1 to 3, to finetune a `Llama-3-8B-Instruct` student (Dubey et al., 2024).

We further experiment with the logical reasoning dataset `ProntoQA` (Saparov & He, 2022), composed of synthetically generated chain-of-thought style reasoning questions, with a `Qwen1.5-7B-Chat` student. Finally we consider the `Game of 24` dataset, where a model is required find arithmetic operations given 4 separate numbers to obtain 24. Here we use a `Qwen2.5-7B-Instruct` student (Qwen et al., 2025). More dataset details are provided in Appendix E.1.

For all datasets except for `Game of 24` we use simple prompt-based synthetic data generation with a `GPT-4o` teacher; see Appendix E.2 for our prompts. For `Game of 24` we use backward

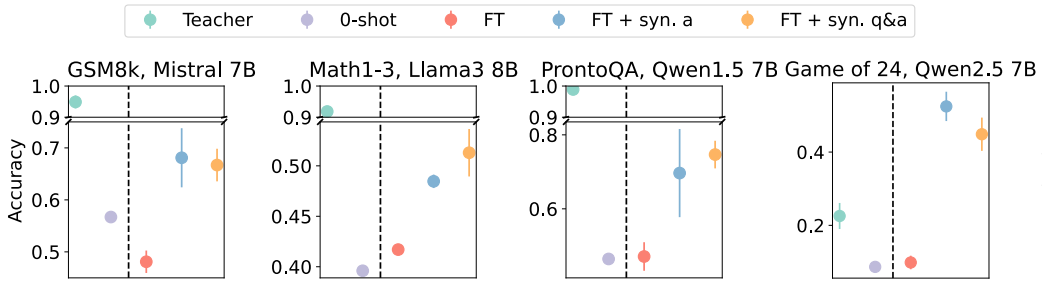

Figure 2: **SFT performance on 1k data points for various datasets and SLMs.** We compare the effect of synthetic answer generation and synthetic question and answer generation to using the seed dataset, D_0 for SFT. 0-shot SLM and teacher performances are included for reference. All datasets use a `GPT-4o` teacher, for `Game of 24` we use a `GPT-o3-mini` teacher.

reasoning: if the answer is `13*8-10*8=24`, for example, we can construct a new question by setting two integers to variables `a*b-10*8=24` and solving to generate new questions (Jiang et al., 2023b). We use a `GPT-o3-mini` teacher for backward reasoning (qualitatively this produces better questions than `GPT-4o`), see Appendix E.2.4 for further details.

5.2 Finetuning Setup

To enable new instruction-following capabilities we finetune our student on synthetic data $\hat{D}_t$, which are appended to synthetic data from all previous iterations $\hat{D}_{<t}$. For efficient training we adapt LoRA layers (Hu et al., 2022) after each iteration of acquiring data and fitting the model. We avoid warm starting SFT parameters from their pre-trained values (Ash & Adams, 2020; Springer et al., 2025). We set the LoRA rank and alpha parameters to the same value and adapt all linear layers. For optimization we use Adam (Kingma & Ba, 2014), clamp the gradient norm to a maximum of 2.0, and use a batch size of 24 with 2 gradient accumulation steps. The learning rate decays linearly with a warm up of period of 15% of epochs. For `Game of 24` we use a cosine decay learning rate schedule down to a minimum of 1e-9 (Ni et al., 2025). During optimization we perform checkpointing and load the best performing checkpoint at the end of the optimization. For the best possible results we search for optimal learning rates, LoRA ranks and the number of training epochs over a grid (Appendix B). We use a single 80Gb A100 or H100 GPU for all experiments.

5.3 Algorithms

We consider a variety of selection algorithms to understand which are most sample efficient. Prior work has shown that prioritizing "hard" samples accelerates learning (Section 3.1). Indeed we also find this to be the case for iterative synthetic data generation (Section 5.4.4). Thus we consider high uncertainty sampling with greedily decoded student predictions, denoted as "loss (high)" throughout this section. We also consider a low reward selection algorithm using the student's own prediction. We use a `Skywork-Reward-Llama-3.1-8B-v0.2` reward model which obtained the highest score 8b model on RewardBench (Lambert et al., 2024) at the time of writing.

We use Lion (Jiang et al., 2023c) as a "score" baseline, which compares the student and teacher answer `LLM-as-a-judge` scores to categorize a data point either hard or easy. All seed data are assigned into either an easy or a hard set before sampling equally from both. We also consider a baseline that only samples from the hard set, denoted as `LLM-as-a-judge` (hard) (Jazbec et al., 2024). We use the same prompts for `LLM-as-a-judge` scoring as (Jiang et al., 2023c).

We also consider prioritizing data with "incorrect" student answers, $s_i = \mathbf{1}\{\hat{y}_i \neq y\}$ as a proxy for prioritizing hard samples (Lee et al., 2024). Since incorrect selection requires a verifier and the ground truth answers we do not compare to the other scoring methods which don't use label information and we place these results in the appendix for comparison (Appendix C.1).

5.4 Results

We present our main results and show how using **synthetic data yields significant gains in student capabilities** compared to using a seed dataset of the same size (Section 5.4.1). We demonstrate

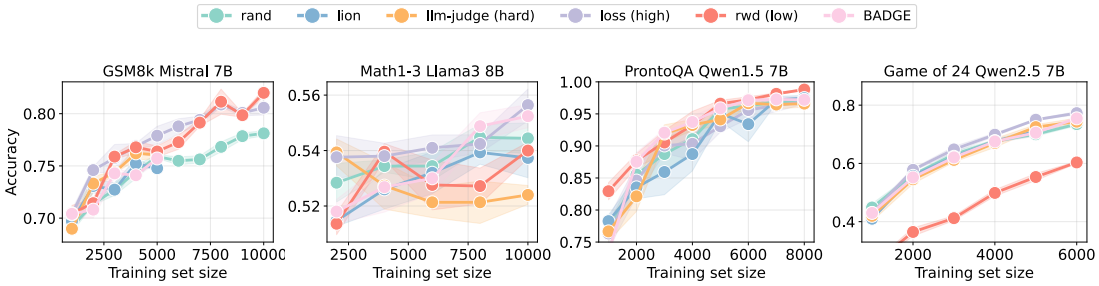

Figure 3: **Iterative synthetic data generation learning curves: student performance for increasing train set sizes or increasing number of iterations**. Each consecutive increase in dataset size corresponds to an iteration of iterative synthetic data generation Algorithm 1. Learning curves are across various dataset student model pairs.

that **iterative synthetic data generation is more data efficient than static generation.** Note that static generation is equivalent to random sampling of prompts for generation in our setting, since it is not conditioned on the current state of the student (Section 5.4.2). **Iterative synthetic data generation also yields state-of-the-art SLM capabilities** using SFT (Section 5.4.3). Finally, we analyse various choices for scoring and selection (Section 5.4.4). Unless stated otherwise, results are a mean and standard deviation over 3 independent runs.

5.4.1 Training on synthetic data improves performance

SFT on synthetic data results in significantly improved capabilities when compared to using the original seed dataset. In Figure 2, we compare SFT performance using the seed data and synthetic data of equal size, showing a dramatic increase in performance across all datasets when doing SFT on synthetic questions-answers pairs. In the same figure, we see large increase in performance when using synthetic answers $z_i = (x_i, \hat{y}_i)$ over the seed answers y due to better formatting and high quality chain-of-thought in synthetic answers. In `Game of 24` there is a small drop in performance when training on synthetic questions and answers versus only synthetic answers, showing that the generation of novel questions by the teacher yields some lower quality synthetic questions. Regardless, next we show how this enables us to scale dataset sizes efficiently.

5.4.2 Iterative Generation Yields More Data-Efficient Results

Active selection is more data efficient than random sampling for generating productive synthetic data, resulting in better performance using fewer samples. In Figure 3, we can see that random sampling underperforms when compared to the active selection methods considered across all datasets. Among these, we find that simply prioritizing high-loss data consistently performs well.

To compare algorithms across all datasets we can aggregate results to construct a pairwise winrate matrix $\boldsymbol{P}$. We increment $\boldsymbol{P}_{ij}$ if $\mathbf{1}\{\hat{\mu}_i - \alpha \cdot \hat{\text{se}}_i > \hat{\mu}_j + \alpha \cdot \hat{\text{se}}_j\}$, where $\hat{\mu}_i$ is the sample mean and $\hat{\text{se}}_i$ is the standard error of the performance of algorithm i for a dataset, for a particular dataset size, and α is the confidence level which we set to 1 (making it a 68% confidence interval). By summing the "wins" across the rows and normalizing we can understand how often algorithms win on average. As a result, lower is better to understand which algorithm is more data efficient. We find that using random sampling is outperformed by nearly all other methods that use the student model to guide synthetic data generation (Figure 4).

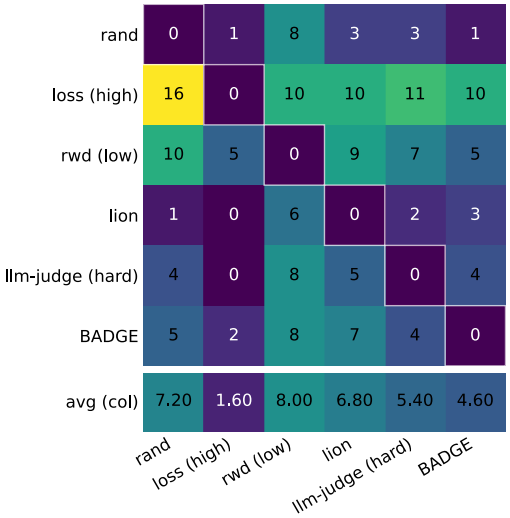

Figure 4: **Pairwise winrate over all datasets and methods.** $\boldsymbol{P}_{ij}$ corresponds roughly to the number of times algorithm i outperforms j. Overall performance is shown in the last row (lower is better).

Dataset	Method	LLM	SFT Dataset Size	Performance
GSM8k	Teacher	GPT-4o	n/a	94.9 ± 1.1
	Orca-Math (Mitra et al., 2024)	Mistral-7B-Instruct-v0.3	200k	86.8
	Xwin-Math (Li et al., 2024)	Mistral-7B-Instruct-v0.3	960k	89.2
	OpenMathInstruct (Toshniwal et al., 2024)	Mistral-7B-Instruct-v0.3	1.8M	80.2
	Iterative Synthetic Data Generation (ours)	Mistral-7B-Instruct-v0.3	10k	80.6 ± 1.2
Math1-3	Teacher	GPT-4o	n/a	91.8 ± 0.7
	MetaMath (Sun et al., 2024)	Llemma-7B	155k	44.1
	Multiagent Debate (Subramaniam et al., 2025)	Llama-3-8B-Instruct	-	57.4 ± 2.2
	Iterative Synthetic Data Generation (ours)	Llama-3-8B-Instruct	10k	55.6 ± 0.4
ProntoQA	Teacher	GPT-4o	n/a	98.9 ± 0.4
	SFT-NL (Zhou et al., 2025)	Qwen2.5-7B-Instruct	3.2k	97.4
	Iterative Synthetic Data Generation (ours)	Qwen1.5-7B-Chat	8k	96.9 ± 0.8
Game of 24	Teacher	GPT-o3-mini	n/a	22.6 ± 1.8
	Tree of Thoughts (Yao et al., 2023)	GPT-4	n/a	74.0
	UFT (Ni et al., 2025)	Qwen2.5-7B-Instruct	13.7k	30.2 ± 2.1
	Iterative Synthetic Data Generation (ours)	Qwen2.5-7B-Instruct	6k	77.3 ± 0.3

Table 1: **Iterative synthetic data generation performs better if not comparably to state-of-the-art SFT methods**. We report the results of iterative synthetic data generation using a high loss selection as this performs the best overall. All SFT methods report the amount of data used for SFT unless it is not reported, designated with a dash (-) in the table. We report a mean and standard error over multiple seeds for our work, however some baselines only report a single seed.

We can glean from Figure 4 that the best selection algorithm uses a high loss and then BADGE. These active learning methods outperform computationally demanding methods that rely on `LLM-as-a-judge`. Indeed if we compare the number of input tokens for the teacher as a proxy for the amount of compute required then Lion and `LLM-as-a-judge` are indeed far more expensive than other methods in terms of input token efficiency in Figure 5.

5.4.3 Comparing to other SFT methods

Iterative synthetic data generation obtains state-of-the-art SFT performance. Table 1 compares the results of iterative synthetic data generation with high-loss selection to prior works in SFT. For `GSM8k`, our work obtains performance comparable to SFT on 1.8M datapoints (Toshniwal et al., 2024). The best result, Xwin-Math, obtains an accuracy of 89.2 using $100\times$ more data to fit the same student (Li et al., 2024) model. For `Math1-3` our work is on par with state-of-the-art SFT methods (Subramaniam et al., 2025). This is also the case with the `ProntoQA` dataset, where our approach obtains performance on par with bespoke logical reasoning methods that use a more performant `Qwen2.5-7B-Instruct` student albeit using less SFT data (Zhou et al., 2025). For `Game of 24` our method outperforms state-of-the-art SFT performance even outperforming test-time compute Tree-of-Thought which is also compatible with our own work (Yao et al., 2023).

5.4.4 On the design choices for iterative synthetic data generation

Argmax selection, rather than sampling, results in the best SFT performance. In Figure 6, we compare various data prioritization design choices. The performance for scorers that prioritize data where the student answer is the most uncertain (high loss) or worse quality (low reward) results in the best performance when compared to data for which the model is confident (low loss) or is of better quality (high reward). Furthermore, we compare whether using the ground truth answer y (denoted "gt" in Figure 6) or the student's own prediction $\hat{y}$ is more data efficient. We can see worse performance when computing scores with the ground-truth answer for the loss scorer, while scoring with the reward model results in equal SFT performance.

Finally, we compare selection methods: argmax selection and sampling and can see lower SFT performance when using sampling (labelled with "sampling" in Figure 6). This is because sampling from a softmax distribution of loss or reward scores results in a similar set of selected

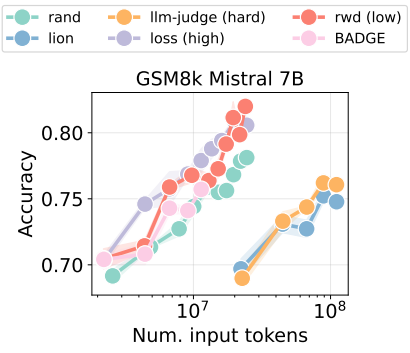

Figure 5: **Iterative synthetic data generation learning curves on `GSM8k`: student performance versus the number of teacher input tokens**. The number of input tokens are a proxy for the amount of compute used by the teacher for various methods.

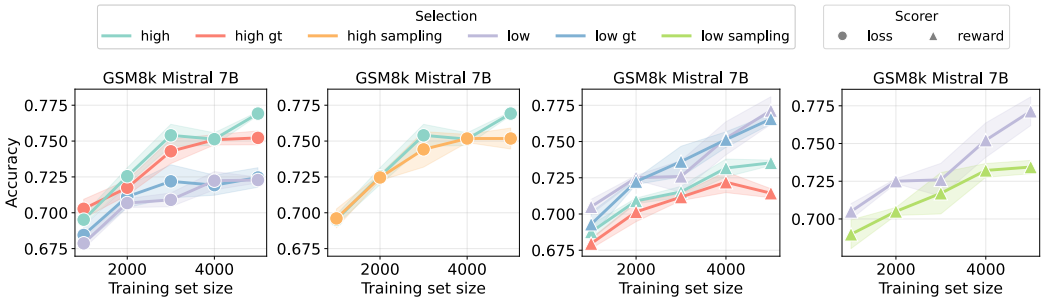

Figure 6: **Performance of iterative synthetic data generation on various data scoring and selection options**. We train on 1k data points at each iteration with a `Mistral-7B-Instruct-v0.3` student on `GSM8k`. We compare prioritizing "difficult" or "easy" data points with a high or low loss or reward. We compare using ground truth answers y to the student's own predictions $\hat{y}$ and using argmax selection against sampling.

data to random sampling. If we select 1k data points from the `GSM8k` seed dataset and look at the distribution of loss scores via sampling for the highest and lowest 1k data, then the distributions are indistinguishable to the naked eye. Argmax selection produces distinct distributions (Figure 7).

6 Limitations

Iterative synthetic data generation for finetuning. We only consider SFT, we do not consider efficient synthetic data generation to accelerate training for RLHF, continual pre-training (Yang et al., 2024) or pre-training (Maini et al., 2025), for instance.

The limits of the teacher. We assume that the teacher is able to generate high quality questions and answers. For `GSM8k`, `Math1-3` and `ProntoQA` the teacher performance is high and so we assume $\hat{z}_i$ is correct. For `Game of 24` we rely on backward reasoning (specific to arithmetic) and a verifier to assess the teacher's synthetic data. We have yet to test the limits of prompt-based synthetic data generation in settings where teacher capabilities fall short.

Data generation is noisy. We can obtain state-of-the art capabilities using iterative synthetic data generation. However, synthetic data generation is a noisy process (Appendix C.2); we have no guarantee of the similarity of the synthetic data to the seed data. It is not obvious how we generate synthetic data with similar or desirable properties from the seed dataset. To a certain extent this is performed by our selection algorithms, however the "steerability" of synthetic data generation is an interesting direction of future work.

7 Conclusion and Discussion

Synthetic data are extremely effective for finetuning SLMs, enabling substantial capability improvements. In this work, we focus on supervised finetuning with synthetic data. We demonstrate that iterative synthetic data generation is the most effective strategy for finetuning SLMs under a fixed training data budget. By adapting teacher generation to the evolving state of the student model, this approach creates a natural curriculum that consistently outperforms static synthetic datasets in both performance and data efficiency. Furthermore, in line with Occam's razor, we find that simple data selection methods, such as prioritizing hard samples with high loss, outperform complicated and expensive `LLM-as-a-judge` based methods.

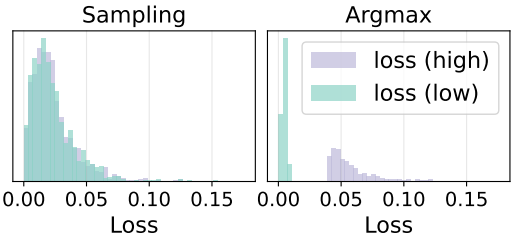

Figure 7: **Distribution of losses for different sampling methods**. We select 1k according to a high or low loss sampling (left) and argmax selection (right) for `GSM8k` and can see almost no difference when using sampling.

8 Reproducibility Statement

Reproducibility goes to the heart of our study of different selection algorithms for data efficient synthetic data generation. In our study, all the results are stated as means and standard errors over 3 independent replicates. This has been done in an effort to encapsulate the variance arising from the datasets we use and our experimental setup, and ensures that the performance differences arise due to the choice of selection methods rather than random variation. As a result we use uncertainties to weight our claims resulting a more reproducible study.

We will release source code upon publication.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006. 18

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020. 6

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019. 3, 17

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025. 1

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 1

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. Instruction mining: Instruction data selection for tuning large language models. *arXiv preprint arXiv:2307.06290*, 2023. 4

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023. 1, 3

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 5, 18

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019. 3

Yalun Dai, Yangyu Huang, Xin Zhang, Wenshan Wu, Chong Li, Wenhui Lu, Shijie Cao, Li Dong, and Scarlett Li. Data efficacy for language model training. *arXiv preprint arXiv:2506.21545*, 2025. 4

Florian E Dorner, Vivian Y Nastl, and Moritz Hardt. Limits to scalable evaluation at the frontier: Llm as judge won't beat twice the data. *arXiv preprint arXiv:2410.13341*, 2024. 4

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024. 4, 5, 18

Talfan Evans, Shreya Pathak, Hamza Merzic, Jonathan Schwarz, Ryutaro Tanno, and Olivier J Henaff. Bad students make great teachers: Active learning accelerates large-scale visual understanding. In *European Conference on Computer Vision*, pp. 264–280. Springer, 2024. 4

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997. 3

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pp. 1183–1192. PMLR, 2017. 3

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021. 5, 18

Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011. 3

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 6

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*, 2024. 18

Metod Jazbec, Menglin Xia, Ankur Mallick, Daniel Madrigal, Dongge Han, Samuel Kessler, and Victor Rühle. On efficient distillation from llms to slms. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*, 2024. 6

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023a. 5, 18

Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019. 4

Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James T Kwok. Forward-backward reasoning in large language models for mathematical verification. *arXiv preprint arXiv:2308.07758*, 2023b. 6, 18

Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. Lion: Adversarial distillation of proprietary large language models. *arXiv preprint arXiv:2305.12870*, 2023c. 2, 3, 4, 5, 6, 18

William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984. 5

Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt J Kusner. No train no gain: Revisiting efficient training algorithms for transformer-based language models. *Advances in Neural Information Processing Systems*, 36:25793–25818, 2023. 4

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018. 4

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254, 2024. 1

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019. 3

Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1193–1200, 2011. 3

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024. 6

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv:2403.15042*, 2024. 2, 3, 4, 5, 6, 16

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. Common 7b language models already possess strong math capabilities, 2024. URL https://arxiv.org/abs/2403.04706. 8

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023. 3, 4

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/. 18

Chengyuan Liu, Yangyang Kang, Fubang Zhao, Kun Kuang, Zhuoren Jiang, Changlong Sun, and Fei Wu. Evolving knowledge distillation with large language models and active learning. *arXiv preprint arXiv:2403.06414*, 2024. 1, 2, 3, 18

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023. 3

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015. 4

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023. 2, 3

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992. 3

Pratyush Maini, Vineeth Dorna, Parth Doshi, Aldo Carranza, Fan Pan, Jack Urbanek, Paul Burstein, Alex Fang, Alvin Deng, Amro Abbas, et al. Beyondweb: Lessons from scaling synthetic data for trillion-scale pretraining. *arXiv preprint arXiv:2508.10975*, 2025. 9

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022. 4

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020. 3

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024. 1, 2, 3, 5, 8, 18, 25

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023. 3

Tianwei Ni, Allen Nie, Sapana Chaudhary, Yao Liu, Huzefa Rangwala, and Rasool Fakoor. Offline learning and forgetting for reasoning with large language models, 2025. URL https://arxiv.org/abs/2504.11364. 6, 8, 24

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022. 1

Arjun Panickssery, Samuel Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802, 2024. 4

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023. 3

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115. 5, 18

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022. 5, 18, 21

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017. 3

Burr Settles. Active learning literature survey. 2009. 3

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 1070–1079, 2008. 2, 3

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. *arXiv preprint arXiv:2503.19206*, 2025. 6

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025. 8

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *Advances in Neural Information Processing Systems*, 37:51118–51168, 2024. 8

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023. 3

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. 3

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset, 2024. URL https://arxiv.org/abs/2402.10176. 8

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 1, 2

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*, 2023. 4

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023. 5

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto. Synthetic continued pretraining. *arXiv preprint arXiv:2409.07431*, 2024. 9

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models, 2023. 8

Liang Zeng, Liangjun Zhong, Liang Zhao, Tianwen Wei, Liu Yang, Jujie He, Cheng Cheng, Rui Hu, Yang Liu, Shuicheng Yan, et al. Skywork-math: Data scaling laws for mathematical reasoning in large language models–the story goes on. *arXiv preprint arXiv:2407.08348*, 2024. 3

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023. 2, 3, 5

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024. 1

Yujun Zhou, Jiayi Ye, Zipeng Ling, Yufei Han, Yue Huang, Haomin Zhuang, Zhenwen Liang, Kehan Guo, Taicheng Guo, Xiangqi Wang, et al. Dissecting logical reasoning in llms: A fine-grained evaluation and supervision study. *arXiv preprint arXiv:2506.04810*, 2025. 8

Appendix

Table of Contents

A The Use of Large Language Models

We used a coding assistant to help implement and debug our experiments. Regarding paper writing we used LLMs for finding related work, to assist with grammar queries, and to assist in generating the figures in the paper.

B Additional Experimental Setup Details

We introduce additional details of our experimental setup detailed in Section 5.3. We outline the hyperparameter grid search for SFT below.

B.1 LoRA hyper-parameter tuning setup

We sweep through learning rate and LoRA rank hyper-parameters for finetuning using on 1k question-answer pairs from the original seed dataset and 1k question-answer pairs synthetically generated by the teacher model to obtain the best hyperparameters for seed datasets D_0 and synthetic datasets $\hat{D}_t$. Refer to Table 2 for optimal hyperparameters.

Model	Dataset	LoRA Rank	Learning Rate	Epochs
`Mistral-7B-Instruct-v0.3`	`GSM8k` seed	32	1e-4	10
`Llama-3-8B-Instruct`	`Math1-3` seed	32	1e-6	13
`Qwen1.5-7B-Chat`	`ProntoQA` seed	32	1e-5	13
`Qwen2.5-7B-Instruct`	`Game of 24` seed	16	1e-5	13
`Mistral-7B-Instruct-v0.3`	`GSM8k` synthetic	32	1e-4	10
`Llama-3-8B-Instruct`	`Math1-3` synthetic	64	1e-4	13
`Qwen1.5-7B-Chat`	`ProntoQA` synthetic	32	1e-5	13
`Qwen2.5-7B-Instruct`	`Game of 24` synthetic	16	5e-4	30

Table 2: Hyper-parameters for LoRA fine-tuning for all seed and synthetic datasets.

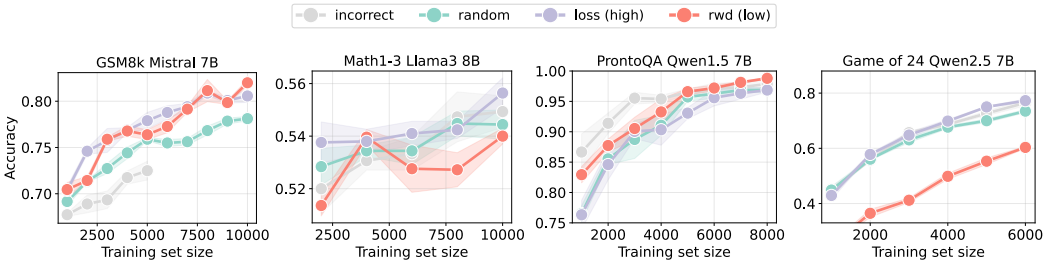

Figure 8: **Iterative synthetic data generation learning curves, showing student SFT performance after training on synthetic data of increasing size with incorrect data prioritization**. We highlight prioritizing data points with incorrect student answers versus select prioritization methods for comparison. Each consecutive increase in dataset size corresponds to an iteration of iterative synthetic data generation (Algorithm 1). Learning curves are across various dataset student base model pairs.

C Additional Results

We introduce additional results that support the main claims in our main paper. In Appendix C.1 we introduce results of prioritizing synthetic data generation using incorrect student predictions. We do not include these results in the main paper for comparison since they require a verifier and the ground truth answer y for scoring unlike the other scoring methods considered in our main experiments (Section 5.4.2). In Appendix C.2 we analyze the workings of synthetic data generation to show that despite introducing noise it results in a curriculum and consistently high quality SFT data.

C.1 Prioritizing incorrect samples

A simple data point scoring mechanism is to assign a $\{0, 1\}$ score for an incorrect or correct answer from the student model. This scoring mechanism requires a verifier and so not directly comparable to the other scoring methods we consider which do not require the ground truth answer to assign a score to a data point (Section 5.3). Regardless we show the results of performing iterative synthetic data generation by prioritizing incorrect samples in Figure 8. For `GSM8k` this method severely underperforms other prioritization methods and random sampling. For `Math1-3` The results are on par with high loss prioritization which performs best. For both `ProntoQA` and `Game of 24` incorrect answer prioritization obtains results on par with the best scoring methods if not the best results for certain n. Considering a pairwise win-rate (described in Section 5.4.2) we can see from the row for incorrect prioritization that it outperforms and therefore is more data efficient in many instances with a high number of "wins" versus other methods. However at the same time looking at the corresponding column it is outperformed by many of the other methods in particular high loss and low reward selection so it results in a poor overall score in the final row Figure 9. Overall it is a

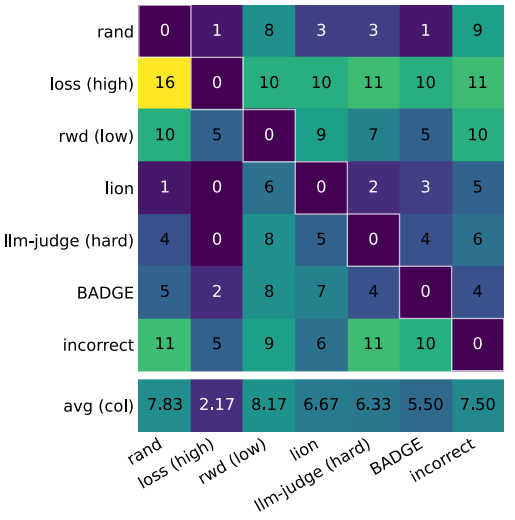

Figure 9: **The pairwise win rate matrix over all datasets and all methods including incorrect prioritization**. Element $\boldsymbol{P}_{ij}$ corresponds roughly to the number of times algorithm i outperforms algorithm j including results of incorrect student answer prioritization (Lee et al., 2024). Column-wise averages at the bottom display overall performance (lower is better).

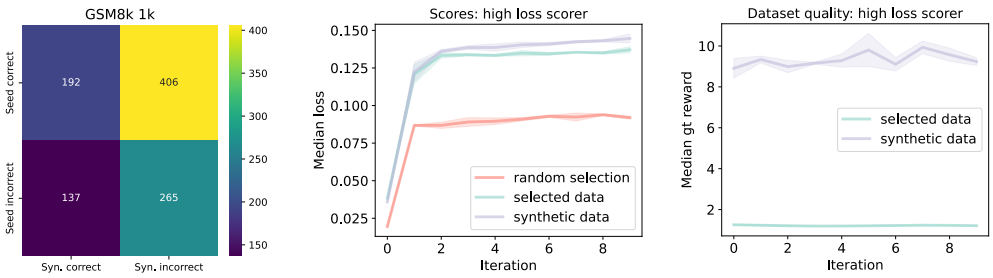

Figure 10: **Synthetic data generation is noisy, however it retains properties of the selected data**. Left: confusion matrix of accuracies from a `Mistral-7B-Instruct-v0.3` student on 1k points from `GSM8k` and corresponding 1k points after synthetic data generation. Middle: the median loss score over selected $\bar{D}_t$ and synthetic $\hat{D}_t$ datasets over 10 iterations of synthetic data generation. Right: the median ground truth reward - a measure of dataset quality - over selected $\bar{D}_t$ and synthetic datasets $\hat{D}_t$ over 10 iterations of synthetic data generation.

.

simple method and has the possibility of obtaining strong capabilities and being more data efficient than random sampling or static dataset generation.

C.2 Synthetic data generation preserves properties of the selected data

The prompt-based synthetic data generation methods we use, although widespread are noisy insofar that the process changes individual data points; points that are previously correctly or incorrectly answered by the student are then incorrectly or correctly answered by the student. Specifically, we construct a confusion matrix of accuracies using a `Mistral-7B-Instruct-v0.3` student on D_0 and corresponding accuracies on $\hat{D}$ after random selection. We see many seed and synthetic data point pairs lie in the in off-diagonals in Figure 10 (left). However if we use a high loss scorer and track the median of the loss of $\bar{D}_t$ and $\hat{D}_t$ over the course of iterative synthetic data generation we can see that first of all there is a curriculum that is formed where the data's loss increases overall. Secondly, that the loss of the selected and synthetic datasets are similar at a dataset level, despite the noisy synthetic data generation process. When we measure the quality of the selected $\bar{D}_t$ and synthetic $\hat{D}_t$ data by using the ground truth reward we see that it remains constant over the course of synthetic data generation, but the quality of the synthetic data increases substantially over the selected data (Figure 10).

D Baseline further details

In this section we provide further details regarding the baselines described in Section 5.3. In particular we provide further details on the BADGE data selection method and the `kmeans++` algorithm.

D.1 BADGE

BADGE (Ash et al., 2019) uses the gradient with respect to the output layer of a neural network and uses `kmeans++` Algorithm 2 to select diverse gradients in this embedding space to 1. maximize the diversity of the selected data points and 2. to select points with a large norm of the gradient of the loss which is equivalent to selecting points with a high uncertainty.

E Dataset further details

In this section we provide in depth details on the datasets used in our experiments together with the dataset sizes used throughout our empirical study of iterative synthetic data generation (Appendix E.1). Also we provide the prompts used for synthetic data generation (Appendix E.2).

Algorithm 2 `k-means++` (Arthur & Vassilvitskii, 2006)

Require: Dataset $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, number of data points to sample from X, $k < n, k \in \mathbb{N}$, Euclidean distance $d(\cdot, \cdot)$.

Choose $c_1 \sim \text{Uniform}(X)$ and set $C \leftarrow \{c_1\}$
For each $x \in X$, set $D(x) \leftarrow d(x, c_1)$ // distance to nearest chosen center
for $i = 2$ to k **do**
    Sample $x^\star \in X$ with probability

$$\Pr[x^\star = x] \;=\; \frac{D(x)^2}{\sum_{z \in X} D(z)^2}\,.$$

    $C \leftarrow C \cup \{x^\star\}$
    for each $x \in X$ **do**
        $D(x) \leftarrow \min\big\{D(x),\, d(x, x^\star)\big\}$ // update distance to nearest center
    end for
end for
return C

We introduce the seed question and answer datasets D_0. The validation and test sets are taken from the original seed datasets as opposed to using synthetic data. The train sets $\hat{D}_t$ are synthetically generated. We summarize the datasets sizes in Appendix E.1. Unless otherwise stated we use a `GPT-4o` teacher. We prompt the teacher with few-shot examples from D_0 to generate a new synthetic questions (Liu et al., 2024). For all datasets we throw away similar synthetic questions if the rouge-score with respect to all previously generated questions is above 0.7 (Lin, 2004; Jiang et al., 2023c).

`GSM8k`. We perform SFT on a `Mistral-7B-Instruct-v0.3` (Jiang et al., 2023a) student on school level mathematics questions (Cobbe et al., 2021). We use an external language model `gpt4o-mini` to assess whether the student's answer is equivalent to the ground truth answer (Mitra et al., 2024), see Appendix E.3 for details. We take 748 question-answer pairs from the test set as a validation set and 500 question-answer pairs as a test set[*].

`Math1-3`. We finetune a `Llama-3-8B-Instruct` (Dubey et al., 2024) student on the competition math dataset (Hendrycks et al., 2021) which consists of more difficult math questions[†]. The dataset is classified into 5 levels of question difficulty. We use the easiest levels 1 to 3 and pick 500 question-answer pairs from the test set for validation. We assess the correctness of an answer by matching the solution to the regular expression `\boxed{(\d*)}`.

`ProntoQA`. The questions are synthetically generated logical chain-of-thought style reasoning questions with boolean answers (Saparov & He, 2022). We perform SFT on a `Qwen1.5-7B-Chat` student model. We use an external language model `gpt4o-mini` to assess whether the student's reasoning steps are correct and answer is equivalent to the ground truth answer like for `GSM8k` (Mitra et al., 2024), see Appendix E.3 for details. We use 300 question-answer pairs as a validation set and the remaining 200 as a test set[‡].

`Game of 24`. We use a `Qwen2.5-7B-Instruct` (Qwen et al., 2025) student on the task of using basic arithmetic on 4 separate numbers to obtain 24[§]. Each question can have multiple solutions, we treat each new answer as a separate data point. We use backward reasoning to synthetically generate new questions (Jiang et al., 2023b) and use `GPT-o3-mini` as a teacher model (qualitatively this produces better questions than `GPT-4o`). We verify that the backward reasoned final

[*] `https://huggingface.co/datasets/openai/gsm8k`

[†] `https://huggingface.co/datasets/hendrycks/competition_math`

[‡] We use`https://huggingface.co/datasets/renma/ProntoQA` for validation and testing, as a train set we use `https://huggingface.co/datasets/longface/prontoqa-train` like in (Huang et al., 2024), questions and answers are distinct between these two ProntoQA datasets.

[§] `https://huggingface.co/datasets/nlile/24-game`

Dataset	Seed Size	Validation Size	Test size
GSM8k	7473	748	500
Math1-3	3504	500	500
ProntoQA	2880	300	200
Game of 24	2217	500	300

Table 3: Summary of the seed dataset sizes, validation and test set sizes. For all datasets we use 1k data points per iteration for finetuning.

answer evaluates to 24 and uses 4 numbers. We use GPT-4o to then generate reasoning steps. We assess the correctness of the student's final answer by matching the regular expression \boxed{}, checking that all numbers in the question are used once and that the matched solution evaluates to 24. Synthetic questions are not checked for rouge-score overlap since the numbers which we can choose as questions are limited.

E.1 Seed dataset sizes

We summarize the seed dataset sizes for all datasets used in our experiments. The seed dataset D_0, is used for scoring and selecting data points to get $\bar{D}_t$ to then put forward to prompt-based synthetic data generation (Section 4.2). We set the validation and test sets to be from the original seed datasets. We use the resulting synthetic datasets $\hat{D}_t$ for training, we generate a fixed sized training dataset to enable comparison between selection methods (Section 5.1).

E.2 Synthetic data generation prompts

We provide the prompts used for prompt-based synthetic data generation (described in Section 4.2) below for all datasets used in our experiments:

- GSM8k see Appendix E.2.1.
- Math1-3 see Appendix E.2.2.
- ProntoQA see Appendix E.2.3.
- Game of 24 see Appendix E.2.4.

E.2.1 Grade School Maths

Below is the prompt we use for synthetic question generation using a GPT-4o teacher. In the prompt below $\{0\}$ are few-shot examples of questions and answers $\{z_i\}_{i=1}^{k} \sim D_0$, we set $k = 5$ for all our experiments and $\{1\}$ is the question from the selected dataset $\bar{x} = \bar{z}[0]$ where $\bar{z} \sim \bar{D}_t$.

The few-shot examples are formatted as follows:

```
"#Given Instruction#: {} #Answer#: {}"
```
(4)

GSM synthetic question generation prompt

```
I want you to act as Instruction Creator.
Your objective is to rewrite a #Given Instruction# into a
more complex version, to make it a bit harder.
The #Rewritten Instruction# must be reasonable and must be
understood and responded to by humans.
Here are some #Examples#:
{0}
I want you to act as Instruction Creator.
Your objective is to rewrite a #Given Instruction# into a
more complex version, to make it a bit harder.
The #Rewritten Instruction# must be reasonable and must be
understood and responded to by humans.
You MUST complicate the #Given Instruction# using the
following method:
1.  Change the names of people #Given Instruction#.
2.  Change the objects in the #Given Instruction#.
3.  Change any quantities and durations in the #Given
Instruction#.
4.  Add 1 to 3 more operations in #Rewritten Instruction#.
5.  Change the operations, for example:  multiplication,
division, subtraction, addition, percentages, fractions and
combinations of these.
6.  You should try your best not to make the #Rewritten
Instruction# become verbose, #Rewritten Instruction# can only
add 10 to 20 words into #Given Instruction#.
Use #Examples# to complicate #Given Instruction#.
'#Given Instruction#', '#Rewritten Instruction#', 'given
instruction' and 'rewritten instruction' are not allowed to
appear in #Rewritten Instruction#.
#Given Instruction#:
{1}
#Rewritten Instruction#:
```

We use the following prompt to obtain synthetic answers from our `GPT-4o` teacher (and from our student model):

```
"Question: {} Solve the problem step-by-step. Answer:".
```
(5)

E.2.2 MATH1-3

Below is the prompt we use for synthetic question generation using a `GPT-4o` teacher, $\{0\}$ are few shot examples of questions, answers and the type of problem e.g. Geometry, Algebra etc.In addition to questions and answers the Math dataset also contains the type of mathematics problem. So for the Math dataset $z = (x, y, t)$ where t is the type of problem. The number of few show examples is set to 5 and are of the same type as the seed question. In the prompt below $\{1\}$ is the type of mathematics.

The few-shot examples are formatted as follows:

```
"The type of math problem is {}. #Given Instruction#: {} #Answer#: {}"
```
(6)

Math1-3 synthetic question generation prompt

```
I want you to act as an Instruction Creator for {1}
mathematics problems.
Create a new question #Rewritten Instruction# by using #Given
Instruction# as inspiration.  The new question should have a
single unique answer.
Ensure that the type of the question you generate #Rewritten
Instruction# matches the type of instruction #Given
Instruction#.
Make #Rewritten Instruction# different from #Given
Instruction#.
The #Rewritten Instruction# must be reasonable, have a
solution and must be understood and responded to by humans.
Here are some #Examples#:
{0}
Use #Examples# as inspiration to make #Rewritten Instruction#
different to #Given Instruction#.
'#Given Instruction#', '#Rewritten Instruction#', 'given
instruction' and 'rewritten instruction' are not allowed to
appear in #Rewritten Instruction#.
#Given Instruction# is a {1} math problem.
#Given Instruction#:
{2}
#Rewritten Instruction#:
```

We use the following prompt for obtaining synthetic answers from our `GPT-4o` teacher (and for obtaining answers from our student model):

```
"Can you solve the following math problem?  {0} Provide a bullet
point summary of your reasoning.  Your final answer should be
a single answer, in the form \boxed{answer}, at the end of your
response."
```

E.2.3 PRONTOQA

Below is the prompt we use for synthetic question generation using a `GPT-4o` teacher for the `ProntoQA` dataset (Saparov & He, 2022). A datapoint from the `ProntoQA` dataset is comprised of a context, question and answer $z = (x = (c, q), y)$ where x is comprised of the context c and question q. The answers y are boolean. The few-shot question generation is therefore comprised of contexts and questions for the teacher to generate new synthetic context and questions $\bar{x}$. In the prompt below $\{0\}$ are few-shot examples of questions and answers from $\{z_i\}_{i=1}^{k} \sim D_0$, we set $k = 5$ for all our experiments and $\{1\}$ is the question from the selected dataset $\bar{x} = \bar{z}[0]$ where $\bar{z} \sim \bar{D}_t$.

The few-shot examples $\{0\}$ are formatted as follows:

```
"Context:  {} Question:  {}".
```
(7)

ProntoQA synthetic question generation prompt

```
I want you to act as an Instruction Creator for logical
problems.
Create a new question #Rewritten Instruction# by using #Given
Instruction# as inspiration.
Make #Rewritten Instruction# different from #Given
Instruction# by changing the names, objects and adjectives.
Also vary the number of logical reasoning steps in #Rewritten
Instruction#.  Ensure that it is possible to answer the
question with true or false answer.
The #Rewritten Instruction# must be reasonable, have a
solution and must be understood and responded to by humans.
Here are some #Examples#:
{0}
Use #Examples# as inspiration to make #Rewritten Instruction#
different to #Given Instruction#.
'#Given Instruction#', '#Rewritten Instruction#', 'given
instruction' and 'rewritten instruction' are not allowed to
appear in #Rewritten Instruction#.
#Given Instruction#:
{1}
#Rewritten Instruction#:
```

We use the following prompt for obtaining synthetic answers from the `GPT-4o` teacher (and for obtaining answers from our student model):

$$\texttt{"Context: \{\} Response: Let's think step by step."}. \tag{8}$$

E.2.4 GAME OF 24

Below is the prompt we use for synthetic question generation using `GPT-o3-mini` for the `Game of 24` dataset. A datapoint from the `Game of 24` dataset is comprised of a set of four numbers and the arithmetic one-line solution to obtain 24. In the prompt below $\{0\}$ are a set of numbers for instance $\bar{x} = [8, 8, 10, 12]$ and $\{1\}$ is the arithmetic answer for instance $\bar{y} = (12 - 10) \times 8 + 8$ where $\bar{z} = (\bar{x}, \bar{y})$ and $\bar{z} \sim \bar{D}_t$. We use backward reasoning to to obtain a new question and answer to the game of 24 (see the prompt below). We verify that the synthetic answer evaluates to 24 and that all the numbers from the synthetic question are also present in the synthetic answer. Since backward reasoning for synthetic data generation produces both the question and the answer, we then prompt our teacher, `GPT-4o` in a second step, with both the synthetic question and answer to get a synthetic reasoning trace $\hat{y} := [\hat{r}, \hat{y}]$ without any verification of the reasoning steps to construct $\hat{D}_t$ (in the second prompt below).

Game of 24 synthetic question generation prompt

```
I want you to act as an instruction creator.  I want you to
write a new problem to the game of 24.
The numbers {0} need to be used to obtain the number 24.
Use each number once, even if a number is repeated use it
multiple times, with the arithmetic operations +, -, *, /
to obtain 24.  Here is how the above numbers {0} are used to
obtain 24:  {1}.

I want you to create a new problem to the game of 24 using
{1}.  Let's use a backward thinking method.  Take two of the
distinct numbers in {1}.  Call them a and b.  Then construct
an equation with two unknowns, a and b.  Pick integer values
for the first variable b then solve for a.

For example the numbers 8, 8, 10, 13 can be used to get
24:  13*8-10*8=24.  We can construct the following equation
a*b-10*8=24 by substituting a=13 and b=8.  Rearranging we get
a=104/b.  Let's pick an integer which divides into 104 for b:
b=4 therefore a=26.
We also could have picked b=2 and so a=62.  Therefore one
possible answer to the game of 24 using this backward method
is \boxed{4*26-10*8}.  If no answer is possible return
\boxed{null}.

Here is the current solution {1} again.  Enclose the new
equation which results in 24 in \boxed{}.  Let's use this
backward thinking method and think step by step.
```

Game of 24 prompt for synthetic reasoning steps

```
Use numbers and basic arithmetic operations (+ - * /) to
obtain 24.  Each step, you are only allowed to choose two
of the remaining numbers to obtain a new number.
Input:  4 4 6 8
Steps:
4 + 8 = 12 (left:  4 6 12)
6 - 4 = 2 (left:  2 12)
2 * 12 = 24 (left:  24)
Answer:  (6 - 4) * (4 + 8) = 24
Input:  2 9 10 12
Steps:
12 * 2 = 24 (left:  9 10 24)
10 - 9 = 1 (left:  1 24)
24 * 1 = 24 (left:  24)
Answer:  (12 * 2) * (10 - 9) = 24 Input:  4 9 10 13
Steps:
13 - 10 = 3 (left:  3 4 9)
9 - 3 = 6 (left:  4 6)
4 * 6 = 24 (left:  24)
Answer:  4 * (9 - (13 - 10)) = 24
Input:  1 4 8 8
Steps:
8 / 4 = 2 (left:  1 2 8)
1 + 2 = 3 (left:  3 8)
3 * 8 = 24 (left:  24)
Answer:  (1 + 8 / 4) * 8 = 24
Input:  5 5 5 9
Steps:
5 + 5 = 10 (left:  5 9 10)
10 + 5 = 15 (left:  9 15)
15 + 9 = 24 (left:  24)
Answer:  ((5 + 5) + 5) + 9 = 24
Input:  {question}
Here is the final answer:  {answer}
Provide the steps to obtain the final answer which equates
to 24, as if you did not have access to the answer.  Put your
final answer within \boxed{answer}.  Steps:
```

We use the following prompt to get answers from the student (Ni et al., 2025):

Game of 24 student prediction prompt

```
Use numbers and basic arithmetic operations (+ - * /) to
obtain 24.  Each step, you are only allowed to choose two
of the remaining numbers to obtain a new number.
Input:  4 4 6 8
Steps:
4 + 8 = 12 (left:  4 6 12)
6 - 4 = 2 (left:  2 12)
2 * 12 = 24 (left:  24)
Answer:  (6 - 4) * (4 + 8) = 24
Input:  2 9 10 12
Steps:
12 * 2 = 24 (left:  9 10 24)
10 - 9 = 1 (left:  1 24)
24 * 1 = 24 (left:  24)
Answer:  (12 * 2) * (10 - 9) = 24 Input:  4 9 10 13
Steps:
13 - 10 = 3 (left:  3 4 9)
9 - 3 = 6 (left:  4 6)
4 * 6 = 24 (left:  24)
Answer:  4 * (9 - (13 - 10)) = 24
Input:  1 4 8 8
Steps:
8 / 4 = 2 (left:  1 2 8)
1 + 2 = 3 (left:  3 8)
3 * 8 = 24 (left:  24)
Answer:  (1 + 8 / 4) * 8 = 24
Input:  5 5 5 9
Steps:
5 + 5 = 10 (left:  5 9 10)
10 + 5 = 15 (left:  9 15)
15 + 9 = 24 (left:  24)
Answer:  ((5 + 5) + 5) + 9 = 24
Input:  {question}
Put your final answer within \boxed{answer}.  Steps:
```

E.3 Evaluation prompts

To assess whether the student's prediction is equal to the ground-truth answer we use `gpt4o-mini` to verify the correctness of the student. We use the following prompt and a system prompt which is different for each dataset used:

```
"Question:  {} Problem Setter's answer:  {} Student's answer:  {}".
```
(9)

For `GSM8k` we use the following system prompt Mitra et al. (2024):

GSM8k evaluation system prompt

```
As an expert Math teacher, your role is to evaluate a
student's answer to a word problem. The problem is
accompanied by a correct solution provided by the problem
setter. It is important to remember that there may be
various methods to solve a word problem, so the student's
steps might not always align with those in the problem
setter's solution. However, the final answer, typically
a number, should be unique and match the problem setter's
answer. Your task involves analyzing the student's solution
to identify any mistakes and determine whether the answer
can be modified to correct the error. If the student's
answer is unfixable, consider creating practice problems
to help improve their understanding. Use the following
format: Error Analysis: In one sentence, extract the final
answer from the problem setter's solution and compare it
with the student's answer. Do they match? Final Verdict:
Correct/Incorrect.
```

For ProntoQA we use the following system prompt:

ProntoQA evaluation system prompt

```
You are a logical expert. Your role is to evaluate a
student's answer to a logical reasoning problem. The problem
is accompanied by a correct solution provided by the problem
setter. Your task is to assess whether the problem setter's
answer and the student's answer match. Use the following
format: Error Analysis: In one sentence, extract the final
answer from the problem setter's solution and compare it
with the student's answer. Do they match? Final Verdict:
Correct/Incorrect.
```

If the output contains string variations of "Final Verdict: Correct" then the student's prediction is correct and wrong otherwise.

For Math1-3 and Game of 24 we use pattern matching to extract the student's answer and compare to the ground truth, see Section 5.1 for details.