TOWARD EVALUATING MODEL COLLAPSE IN LLMS: INSIGHTS FROM CONTINUAL PRETRAINING

Kristian Minchev *
INSAIT, Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria

Anton Alexandrov
INSAIT, Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria

Martin Vechev
INSAIT, Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria

Nikola Konstantinov
INSAIT, Sofia University "St. Kliment Ohridski"
Sofia, Bulgaria

ABSTRACT

The abundance of content generated by Large Language Models (LLM) at web scale creates a risk of "model collapse", a phenomenon where recursive training on synthetic data negatively affects future generations of models. However, empirical studies of the existence and severity of model collapse remain fragmented due to a lack of standardized tooling, an inconsistent scope of empirical evaluation, and varying measures of collapse. To bridge this gap, we introduce `collapsebench`[1], an ongoing development of a framework designed for principled, reproducible study of model collapse at scale. The framework automates the full recursive pipeline of data curation, training, and generation, offering support for various design choices from prior work via a unified configuration-as-code interface. We aim to devise a modular and customizable testbed for the community to rigorously evaluate model collapse and to propose and study new measures and mitigation strategies. We showcase various functionalities of the framework through supervised finetuning experiments. Using `collapsebench`, we contribute the first evaluation of the severity of model collapse in a realistic setting of continual pretraining (CPT). In particular, we observe a consistent drop in accuracy during iterative training in high-synthetic-data regimes. We also explore the impact of realistic synthetic data curation, observing partial dampening of the effect of model collapse.

1 INTRODUCTION

The recent advances in the capabilities of state-of-the-art LLMs (Brown et al., 2020; Achiam et al., 2023; Team et al., 2025) have led to their widespread adoption in society. As a result, AI-generated text is flooding the Internet (Reuters Institute for the Study of Journalism, 2024; Kapania et al., 2025), including benchmark datasets (Veselovsky et al., 2023; Alemohammad et al., 2023). Additionally, such synthetic text is often hard to distinguish from human-generated content (Porter & Machery, 2024; Kumarage et al., 2023).

Thus, synthetic data may realistically enter into the training data of future LLMs. Unfortunately, prior work suggests that iterative training on synthetic data can lead to long-term degradation of model performance, an effect termed *model collapse (MC)* (Shumailov et al., 2024). Given the abundance and indistinguishability of LLM-generated text, understanding its impact on long-term model performance is of utmost importance to sustainable machine learning research.

Despite the importance of the MC phenomenon, to date, no practical toolbox for evaluating it in realistic large-scale settings has been developed. This creates unnecessary overhead, with prior

*Correspondence to: Kristian Minchev <kristian.minchev@insait.ai>

[1] `collapsebench` is currently in active development. All of our experiments utilize our internal implementation of the framework. We describe its architecture here as a prototype for standardized MC evaluation, with plans for a future public release. We welcome community feedback on the framework's architecture, design choices, and desired functionalities to help shape its development.

results involving repeated reproducibility and implementation efforts. Additionally, the disparity between the training protocols (including datasets, models, etc) adopted in prior work makes it hard to compare results across individual studies. Finally, even though the exploration of new measures and definitions of MC, as well as new mitigation strategies, is an essential step towards solidifying MC research (Schaeffer et al., 2025), evaluating such contributions would be difficult without established benchmarking mechanisms.

Contributions In this work, we present the design of `collapsebench`, a comprehensive framework designed to aid reproducible and principled MC research in realistic settings. We use it to contribute the first evaluation of the severity of MC in a realistic setting of continually pretraining Llama-3.2-3B.

After introducing established concepts from MC research in Section 3, we present the main framework functionalities in Section 4. Our implementation is designed for specifying training configurations and MC settings via a "configuration-as-code" philosophy, in particular providing built-in functionalities for standard prior work MC settings. The framework is structured in a modular fashion, allowing for systematic comparisons of prior methods at each phase of the MC pipeline, e.g., data preprocessing, training, and inference. Finally, customizability through registries allows for direct experimentation with future MC measures and mitigation strategies, while controlling for the remaining part of the MC pipeline.

We utilize our framework implementation to conduct the first MC experiment in continual pretraining of a multi-billion parameter language model (Llama-3.2-3B Grattafiori et al. (2024)). Our results demonstrate long-term performance degradation, which begins at roughly 50% synthetic data. We also study the effect of realistic data filtering on downstream model performance. We evaluate all models through standard benchmarks as well as using Llama-3-70B-Instruct to annotate sampled text, observing subtle indications of collapse through reduced annotation scores.

2 Related Work

Model Collapse Various empirical studies indicate the risk of model collapse in generative models (Shumailov et al., 2024; Alemohammad et al., 2023; Bertrand et al., 2024; Hataya et al., 2022), classification tasks (Taori & Hashimoto, 2023), and regression models (Dohmatob et al., 2024a; 2025). Other works provide theoretical analysis of such iterative loops for Gaussian process latent variable models (Li et al., 2024), simple diffusion models (Fu et al., 2024), as well as toy language models (Dohmatob et al., 2024b; Seddik et al., 2024).

A large number of works attempt to alleviate MC through data-centric methods. Works by Gerstgrasser et al. (2024); Kazdan et al. (2025); Martínez et al. (2023); Alemohammad et al. (2023) study the interplay between synthetic data and real-world data and its downstream effect on model collapse. Most notably, Gerstgrasser et al. (2024); Kazdan et al. (2025) quantify model degradation in various data regimes, questioning the severity of model degradation under natural data collection assumptions. Additionally, Dohmatob et al. (2024a) study model collapse when the synthetic dataset size increases at each iteration. Similarly, other works study mitigation strategies based on modifying synthetic data at the token-level in single-round settings (Zhu et al., 2024), or classification at document-level in supervised finetuning (Drayson et al., 2025; Gambetta et al., 2024).

Our work complements prior research by enabling scalable and principled investigations of model collapse. We showcase this by providing empirical setups at different scales (supervised finetuning as well as large-scale continual pretraining), which give us further insights into the effect of such feedback loops on LLMs. More concretely, in Section 5 we introduce the first experiments with heterogeneous synthetic data replacement, as well as the first experiments on continual pretraining at large scale (3B-parameter model, 9.6B-tokens dataset) in multiple iterations. Further, in Section 5.2.2 we explore the effect of data curation on MC, by preprocessing training data in the same way that the original benchmark dataset (Allal et al., 2025) was curated.

Fragmentation in Definitions and Evaluation Prior work on model collapse studies a variety of MC measures and evaluation protocols. In terms of measures, the literature encompasses at least eight distinct definitions (Schaeffer et al., 2025; Kazdan et al., 2025), ranging from asymptotic divergence in population risk (Gerstgrasser et al., 2024; Bertrand et al., 2024; Kazdan et al., 2025) to the subtle

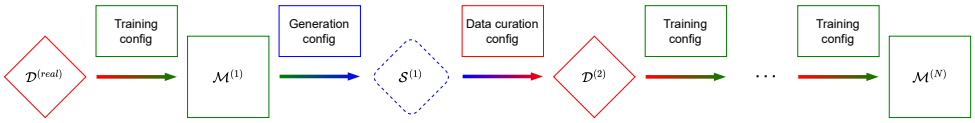


Figure 1: A visualization of the model collapse pipeline. Different colors represent the different main stages of the pipeline. `collapsebench` is designed to allow users to configure the transitions between the phases, automating the orchestration.

disappearance of distribution tails (Shumailov et al., 2024; Wyllie et al., 2024) or the entanglement of data modes (Alemohammad et al., 2023). Consequently, seemingly contradictory results in prior work often stem from differing metrics for failure or inconsistent assumptions (Schaeffer et al., 2025; Kazdan et al., 2025). In terms of protocols, various theoretical assumptions are tested under different scales, training regimes, and data assumptions, which hinders the ability to compare prior works. To bridge this gap, `collapsebench` aims to unify these empirical setups through a single codebase. By standardizing the pipeline, our framework is designed to allow researchers to rigorously test different definitions and metrics within a single reproducible framework.

Synthetic Data in LLM Training The main culprit behind model collapse is suspected to be the inability of a model to fit the target training distribution. Such effects are tightly connected to multiple lines of work that explore strategies for synthetic data generation with the aim of improving model performance.

Hinton (2014); Mobahi et al. (2020) study techniques for transferring the "knowledge" from a large, complex model (the Teacher) to a smaller, more efficient model (the Student). On a similar note, Barbero et al. (2025); Xu et al. (2024) show that models can be prompted to create synthetic data that can partially resemble original post-training datasets. However, the generalization properties of the distilled model's capabilities remain unclear (Gudibande et al., 2023).

Kang et al. (2025) studies how synthetic data affects pretraining at scale in a single-round setup. In particular, it brings more nuanced findings on the effects of synthetic data in large-scale pretraining, particularly in terms of scaling, data type, and mixing ratios. Kovač et al. (2025) discuss the relationship between properties of the original training data and the resulting synthetic data. We refer to Havrilla et al. (2024) for a survey of the role of synthetic data in downstream models.

In contrast to the aforementioned works, the model collapse phenomenon encapsulates unintentional and uncontrollable contamination with synthetic data. Additionally, model collapse frameworks focus on the longer-term effect of the feedback loop on the models. Thus, MC research complements these findings by tracking model performance over multiple iterations.

3 Mathematical Formulation

In order to describe the scope of `collapsebench` in full generality, we formalize model collapse as a discrete-time stochastic process, as proposed by Shumailov et al. (2024). This allows us to capture various real-world settings in which synthetic data generated by a given set of models can harm the training of future ones.

3.1 The Recursive Learning Loop

Let $\mathcal{P}_{data}$ be the true underlying data distribution. We begin with an initial dataset $\mathcal{D}^{(1)} := \mathcal{D}^{(real)} \sim \mathcal{P}_{data}$ consisting of human-generated samples. We define a sequence of models $\mathcal{M}^{(k)}$ and datasets $\mathcal{D}^{(k)}$ for iteration $k \in \{1, \dots, N\}$. For each dataset $\mathcal{D}$, we denote its size by $|\mathcal{D}|$, and by $Sample(\mathcal{D}, \alpha), \alpha \in [0, 1]$, we denote a random subset of $\mathcal{D}$ of size $\lfloor \alpha |\mathcal{D}| \rfloor$, drawing uniformly without replacement. Figure 1 visualizes this process. The recursive loop consists of three distinct phases at each iteration k:

Training phase A learning algorithm $\mathcal{A}$ maps a dataset $\mathcal{D}^{(k)}$ to a model $\mathcal{M}^{(k)} := \mathcal{A}(\mathcal{D}^{(k)})$. This can encompass finetuning and pretraining, and can consist of multiple different stages.

Generation Phase Synthetic data is generated by sampling from the current model $\mathcal{M}^{(k)}$. We define a sampling policy $\pi(\cdot|\mathcal{M}^{(k)})$ (encapsulating hyperparameters such as temperature τ, top-p, or beam search width) to produce a synthetic dataset:

$$\mathcal{S}^{(k)} = \{\hat{x}_i\}_i, \quad \text{where } \hat{x}_i \overset{\text{iid}}{\sim} \pi(\cdot|\mathcal{M}^{(k)}).$$

In supervised finetuning, the sampling policy might also include a prompt distribution $\mathcal{D}^{(prompt)}$. We can also apply preprocessing to the synthetic data, especially in cases where real data has undergone specific preprocessing, and we want the synthetic data to match that. This is implied by the assumption that it is hard to distinguish between synthetic and real-world data in the model collapse pipeline. This can be encapsulated by defining a probabilistic event $\mathcal{E}$ and conditioning the sampling policy on it. As an example, if X is distributed according to $\pi(\cdot|\mathcal{M}^{(k)})$, then $\hat{x}_i \overset{\text{iid}}{\sim} X|\{X \text{ starts with "user:" and contains "assistant:"}\}$ will only contain samples that match a specific user-assistant template, similarly to Kazdan et al. (2025).

Data Curation Phase The training dataset for the subsequent iteration $\mathcal{D}^{(k+1)}$ is constructed by combining the newly-generated synthetic data $\mathcal{S}^{(k)}$ with prior data. We define a curation function Ψ:

$$\mathcal{D}^{(k+1)} = \Psi(\mathcal{D}^{(real)}, \mathcal{S}^{(1)} \ldots, \mathcal{S}^{(k)})$$

We formalize the three primary regimes studied in prior works, as defined by Kazdan et al. (2025):

- **(Partial) Replacement (Ψ_{replace})** The model learns from a mix of the synthetic data $\mathcal{S}^{(k)}$ and the real data $\mathcal{D}^{(real)}$, with $\alpha \in [0, 1]$ denoting the proportion of the mix which is synthetic:
$$\mathcal{D}^{(k+1)} := Sample(\mathcal{D}^{(real)}, 1-\alpha) \cup \mathcal{S}^{(k)}$$
Most often, we have $|\mathcal{S}^{(k)}| = \alpha|\mathcal{D}^{(real)}|$ in order to keep dataset size constant over iterations. In particular, when $\alpha = 1$, we recover the setting of **total data replacement**.
- **Accumulation ($\Psi_{\text{accumulate}}$):** Synthetic data is appended to the growing pool of training data. This simulates a web-crawling scenario where AI-generated content accumulates alongside human content over time:
$$\mathcal{D}^{(k+1)} := \mathcal{D}^{(k)} \cup \mathcal{S}^{(k)} = \mathcal{D}^{(real)} \cup \mathcal{S}^{(1)} \cup \ldots \cup \mathcal{S}^{(k)}$$
- **Accumulation with subsampling ($\Psi_{\text{subsample}}$)** Synthetic data is accumulated, but is always downsampled to keep the training dataset size constant across iterations:
$$\mathcal{D}^{(k+1)} := Sample\left(\mathcal{D}^{(k)}_{(acc)}, \frac{|\mathcal{D}^{(real)}|}{|\mathcal{D}^{(k)}_{(acc)}|}\right),$$
where $\mathcal{D}^{(k)}_{(acc)} = \mathcal{D}^{(real)} \cup \mathcal{S}^{(1)} \cup \ldots \cup \mathcal{S}^{(k)}$ contains all accumulated data.

3.2 Evaluating model collapse

As discussed in Section 2, formalizing the criterion for MC is an avenue of active work. One such metric can be the divergence of the learned distribution $\mathcal{P}_{\mathcal{M}^{(k)}}$ from the true distribution $\mathcal{P}_{data}$. Quantitatively, we measure this degradation via the expected cross-entropy or perplexity (Cover, 1999) on a held-out test set $\mathcal{D}_{\text{test}} \sim \mathcal{P}_{data}$. A positive, monotonically increasing value could indicate a progressive loss of information, characteristic of model collapse. On the other hand, in real-world scenarios, one might instead be interested in benchmark performance rather than likelihood-based metrics as a measure of performance. In such cases, reduction of downstream performance indicates model degradation over time.

4 `collapsebench`

To facilitate principled research into recursive training loops, we are developing `collapsebench`. It serves as a proof-of-concept implementation of the theoretical formulation presented in Section 3, automating the iterative "train-generate-retrain" feedback loop of model collapse. Additionally, `collapsebench` is designed to be modular and highly customizable through a registry-based architecture. In this section, we present the methodology of our implementation by describing its architectural components and utilities.

4.1 YAML-DRIVEN ORCHESTRATION

The design of `collapsebench` adopts a "configuration-as-code" philosophy, where the entire experimental pipeline is defined in a single `YAML` file. This includes model specifications, training hyperparameters, synthetic data ratios, and evaluation suites. By decoupling the experimental logic from the implementation, the framework ensures high reproducibility and allows researchers to manage complex, multi-iteration runs without modifying the underlying Python source code.

The framework implements a state-management system (via a JSON state file) that tracks the pipeline's progress across iterations. This allows for pipeline checkpointing; if an experiment is interrupted, `collapsebench` can resume execution from the exact step (e.g., training or evaluation) where it halted. To ensure fault tolerance, training supports checkpoint restoration, while the generation module utilizes chunking. This allows the pipeline to capture intermediate states of compute-intensive operations, safeguarding against resource loss.

4.2 MODULARITY AND PIPELINE PHASE SEPARATION

The architecture is partitioned into four independent phases: *Data Curator*, *Trainer*, *Evaluator*, and *Generator*. This modularity allows researchers to isolate specific variables; for example, one can "freeze" the training configuration while varying the synthetic data curation strategy (e.g., `replace`, `accumulate`, or `accumulate-subsample`) to observe its specific effect on model stability.

Training The current implementation of the framework supports two primary stages of LLM development, enabling the study of collapse across different training regimes: pretraining (using `transformers` (Wolf et al., 2020)), as well as supervised finetuning (using `trl` (von Werra et al., 2020)). Both can be initialized from a base model or from scratch by providing the configuration and the architecture. Additionally, `collapsebench` supports multi-stage training, orchestrating the transition from each stage to the next.

Generation Our internal implementation uses `vLLM` for scalable and high-performant inference. All parameters of `SamplingParams` and `LLM` classes are passed directly from the configuration file, allowing for full support. Moreover, `collapsebench` supports both inference from an empty string or from a specified prompt dataset.

Data Curation `collapsebench` supports the standard curation strategies for replacing, accumulating, and accumulating with subsampling, as introduced in Kazdan et al. (2025). Since training stages can have separate datasets, our interface allows providing data specification together with respective training arguments. Additionally, it allows for configuring the amount of synthetic and real-world data that enters each iteration.

Further, the framework's architecture allows the specification of additional preprocessing functions. Our implementation currently supports one-time curation before the start of the pipeline, as well as per-iteration preprocessing of the dataset that was initially curated by accumulation, replacement, or subsampling.

Evaluation To provide standard measures of performance degradation, our implementation integrates multiple evaluation backends:

- **LM-Evaluation-Harness:** Direct integration with `lm-eval` (Gao et al., 2024) provides access to hundreds of standard LLM benchmarks such as GSM8K (Cobbe et al., 2021) or MATH (Hendrycks et al., 2021).
- **Automated Perplexity Tracking:** `collapsebench` provides built-in functions for measuring *cross entropy* and *perplexity*. This allows researchers to monitor how the distribution of model outputs diverges from the original data distribution over time.
- **Experiment Tracking:** Integration with Weights & Biases [2] ensures that all metrics across every iteration are logged for comparative analysis.

[2] `https://www.wandb.ai/`

4.3 CUSTOMIZABILITY THROUGH REGISTRIES

A core feature of our methodological framework is its dynamic **Registry system**, inspired by `lm-eval` (Gao et al., 2024). This allows users to inject custom logic at key points in the pipeline without altering the core library. Alternatively, users can import a file and add custom functions to the framework's registries, which allows them to refer to their custom functions by their corresponding registry key, rather than the script path. This enables:

- **Custom Data Curation:** Dynamic data cleaning and tokenization functionalities allow for recreating empirical setups used in prior works, as well as accommodating public datasets for the needs of a specific experiment. For example, we can import custom data curation and validation functions, and then refer to them by their registry keys. Furthermore, we input different data curation strategies before each training round. This is useful in evaluating data-centric methods for collapse mitigation.
- **Generative Validators:** The framework allows for using custom Python functions or Jinja2 templates to filter synthetic outputs. This functionality allows for implementing data filtering strategies while ensuring that the dataset size is consistent across generations, allowing comparison of evaluation scores based on the data quality instead of quantity. One such example is classifier-based filtering, where only outputs exceeding a quality threshold (e.g., marked as high-quality by a classifier) are passed to the next iteration.
- **Custom Evaluations:** The framework's architecture permits the addition of custom evaluation functions, requiring only that they accept a model path and return a dictionary of metrics.

4.4 SCALABILITY: FROM SINGLE GPU TO MULTI-NODE

In its current iteration, `collapsebench` is designed for compatibility with high-performance computing (HPC) environments. All experiments in Section 5.2 have been validated on 4 nodes $\times$ 8 H200 GPUs, showcasing the support for multi-node training and generation. All phases of the pipeline are accessible separately, allowing for customizable resource allocation in HPC clusters. Furthermore, the training phase can leverage `torchrun` (Paszke et al., 2019) and `DeepSpeed` (Rasley et al., 2020) for distributed training. For generation, the framework utilizes a multi-process parallelization strategy that orchestrates distributed `vLLM` instances across multiple GPUs and nodes. Finally, a lightweight synchronization logic ensures that data shards generated across a cluster are automatically merged into a single training set for the subsequent iteration, enabling generation at the scale of billions of tokens in a matter of hours.

5 EXPERIMENTAL STUDIES

In this section, we provide several empirical studies in MC, using our internal implementation of `collapsebench`. First, we examine heterogeneous insertion of synthetic data in supervised finetuning, following Kazdan et al. (2025). We measure the impact of synthetic data in earlier training stages vs later ones by tracking cross-entropy loss over multiple iterations.

Then, we study MC in the context of continual pretraining (CPT) for domain adaptation in mathematical data. To our awareness, this is the first study of MC in CPT at that scale. In this larger-scale experiment, we measure how the synthetic data ratio further reduces downstream performance. Finally, we consider the effect of applying synthetic data curation using a specialized classifier.

5.1 SUPERVISED FINE-TUNING

Following Kazdan et al. (2025), we conduct supervised finetuning of Gemma-2-2B (Team et al., 2024) on Nvidia's HelpSteer2 dataset (Wang et al., 2024). We provide initial preprocessing of the dataset by applying a simple user-assistant template and filtering out samples exceeding 512 tokens. After finetuning, we sample starting from a beginning-of-sequence (BOS) token to produce synthetic data, and we use `collapsebench`'s generative validator functionality to curate a new synthetic dataset that consists entirely of samples following the initial user-assistant template. Within this setting, we study how the distribution of synthetic data across training can affect downstream performance.

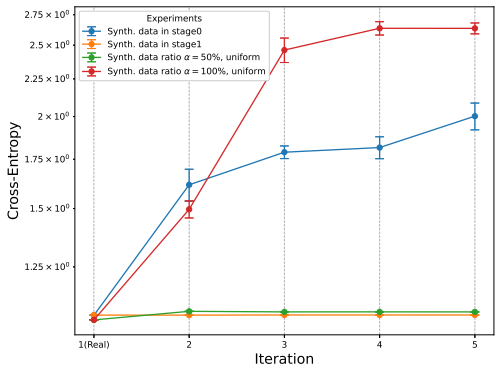


Figure 2: Model collapse in supervised finetuning with heterogeneous synthetic data distribution. We present training pipelines where training consists of two stages, and the synthetic data either enters only the earlier stage (stage0) or the later one (stage1). We compare this to a baseline method where either $\alpha = 50\%$ or $\alpha = 100\%$ synthetic data is uniformly spread across training.

5.1.1 Effect of stage-specific synthetic data

Prior works are often motivated by training setups that use web-scale data, e.g. Shumailov et al. (2024) posit the question "What will happen to GPT-$\{n\}$ once LLMs contribute much of the language found online?". However, such training pipelines often consist of multiple stages (Olmo et al., 2025), in which data quality improves at each scale. For this reason, we conduct an experiment in which different stages get contaminated, and we compare them to a baseline pipeline in which the synthetic data is uniformly mixed within the model's training.

We utilize `collapsebench` to explore performance degradation if the synthetic data contaminates only one stage in a two-stage training pipeline. We also include two baseline scenarios for comparison. In the first one, the synthetic data is half of the whole dataset but is spread uniformly throughout the whole training. The second baseline is one where the synthetic data ratio is $\alpha = 100\%$, corresponding to the classic replace setup in Kazdan et al. (2025).

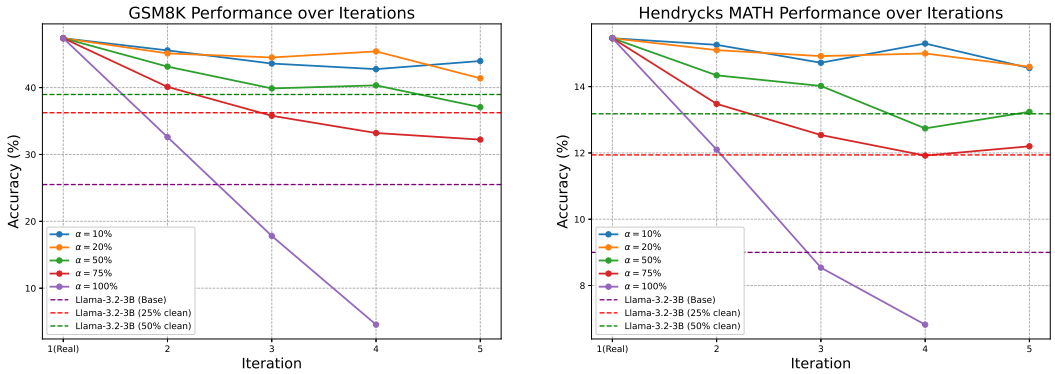
We present the results in Figure 2. We see that when the synthetic data is presented in the first stage, the degradation is more significant, with the order of the cross-entropy nearing that of the $\alpha = 100\%$ setting. On the other hand, when the first half of the dataset is clean, we see that the degradation is significantly smaller. We believe that a more systematic exploration of the distribution in synthetic data is a valuable avenue for future work.

5.2 Continual Pretraining

We now move on to a larger-scale setup that is closer to real-world situations where synthetic data is more relevant. In particular, we study the effect of synthetic data on model performance in the setting of continual pretraining (CPT) for domain adaptation (Gururangan et al., 2020). The base model we select is Llama-3.2-3B (Grattafiori et al., 2024), and the original dataset $\mathcal{D}^{(real)}$ is HuggingFace's finemath-4+ dataset (Allal et al., 2025), comprising 6.7M documents (approx. 9.6B tokens) of high-quality mathematical educational content filtered from the Common Crawl (2008). This dataset has been filtered using a custom model provided by Allal et al. (2025) trained to provide scores from 0 to 5, and filtering out all documents from the CommonCrawl whose score is below 4. For all models we train in this section (including baselines), CPT is conducted from the base model for 1 epoch, with context length 4096, learning rate $5e{-}5$, and batch size 256, matching ablation studies performed by Allal et al. (2025).

We conduct CPT in order to improve the base model's abilities in problem solving, which we measure through 5-shot evaluation on the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) benchmarks.

Synthetic data is generated from the continually pretrained $\mathcal{M}^{(k)}$ model by passing a BOS token as a prompt, in the style of Magpie (Xu et al., 2024). We adopt unconditional generation as it introduces



(a) GSM8K Performance

(b) Hendrycks MATH Performance

Figure 3: Evaluation results over iterations on the GSM8K and MATH benchmarks. Baselines trained on fractions of clean data are indicated by dashed lines.

no prompt-distribution bias into the synthetic data, allowing us to isolate the effect of the recursive training loop itself. We set the temperature to $\tau = 1$ and max length equal to the context size 4096. We observe that the average output length is very close to the 4096-token maximum, likely because the base model's native context window (131072 tokens) biases it toward generating longer sequences than the CPT context length. We set the curation regime as "replace" across all experiments, unless stated otherwise. The number of documents is set for each setting so that the size of the mixed training dataset stays consistent across generations.

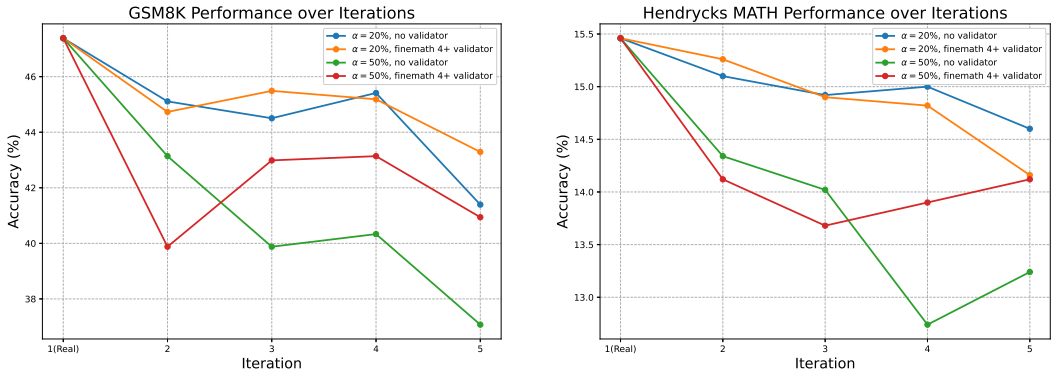
5.2.1 Effect of synthetic data ratio

We begin by measuring model performance degradation under different proportions of synthetic data. We also provide three baseline setups in Figure 3. For synthetic data ratios $\alpha = 50\%, 75\%, 100\%$, we show baseline accuracy in which the base model is continually pretrained (with the same hyperparameters as the models with synthetic data) on a $1 - \alpha$ fraction of the clean dataset and no synthetic data (the case $\alpha = 100\%$ recovers the base Llama-3.2-3B model). By comparing a run with synthetic data α to a baseline that is trained solely on a $1 - \alpha$ fraction of the real dataset, we can evaluate whether synthetic data helps or hurts the fixed CPT protocol. Results are shown in Figure 3.

By tracking the downward trajectory of the solid accuracy curves, we confirm that a significant reduction in performance is present, even at such a scale. However, a significant amount of synthetic data is necessary to achieve a serious reduction in benchmark scores after 5 iterations, as a synthetic data ratio of up to 20% can have a minimal effect on the performance of a model. While this is supported by prior theoretical work (Bertrand et al., 2024), we note that such results bound the performance degradation only in the presence of vanishingly small synthetic data ratios. Further, even in the presence of larger synthetic data ratios, the reduction in performance appears to decrease in later iterations. The clear exception is $\alpha = 100\%$, which shows different behaviour compared to cases where we mix real and synthetic data. We consider the exploration of even longer-term processes an exciting avenue for future work, since it is not immediately clear whether the accuracy plateaus or decreases until complete collapse.

We observe that the experiment curves initially surpass the dashed baseline levels in Figure 3, i.e. in the early iterations, the synthetic data benefits the fixed CPT protocol. This improvement is not immediately obvious, as synthetic documents, even when generated in the best-performing first iteration, are often incoherent and contain mostly incorrect mathematics. In Appendix D we present a few synthetic documents, as well as their scores on the classifier model used in the data curation of the real-world dataset.

Nevertheless, in further iterations, each model's performance approaches or drops below the respective baseline. This highlights the long-term aspect of model collapse as crucial for studying methods for synthetic data generation.



(a) GSM8K Performance

(b) Hendrycks MATH Performance

Figure 4: Comparison of pipelines where data is filtered vs pipelines where data is not filtered.

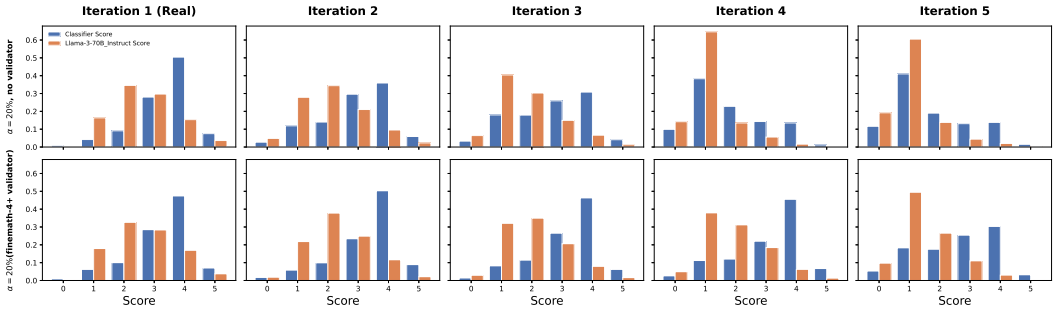


Figure 5: Distribution of synthetic data generated by the models presented in Figure 4 in the presence of $\alpha = 20\%$ synthetic data. The blue histograms show scores labeled by finemath-classifier, whereas the orange histograms represent scores given by Llama-3-70B-Instruct.

5.2.2 Filtering synthetic data with a classifier

While we observe collapse in the previous experiment, it is not realistic to have the unfiltered texts we generate from a trained model substitute for the high-quality data from the original finemath-4+ dataset. Indeed, this dataset has been filtered by Allal et al. (2025) using a custom classifier, which we refer to as finemath-classifier, using annotations made with Llama-3-70B-Instruct (AI@Meta, 2024). In order to provide a meaningful comparison between synthetic and real data, we use the generative validator functionality of `collapsebench` in order to retain synthetic data that is of similar quality to the original dataset according to finemath-classifier. This provides a realistic setup in which the synthetic data also undergoes similar filtration as the real data.

Results are presented in Figure 4 for the cases $\alpha = 20\%$ and $\alpha = 50\%$. In the presence of 20% synthetic data, there is no strong indication that the validator changes performance trends in a meaningful way, and we still observe a small reduction in benchmark accuracy. For $\alpha = 50\%$ the classifier initially offers no significant change to model performance compared to no validator. However, we see that it can bring a positive impact in later iterations. This is in line with the results presented in Figure 3, where the synthetic data improves the accuracy of the considered CPT in the first iterations but performance nevertheless degrades over time. Hence, although the data curation does not fully mitigate collapse, it might still be able to dampen the drop over the long term.

5.2.3 Effectiveness of the finemath classifier

We observe that the use of the classifier does not fully mitigate the reduction in downstream performance. To investigate the underlying cause, we study the effectiveness of the finemath-classifier model in detecting synthetic data from the models we trained. For each model in the MC pipelines conducted in the above subsections, we generate 1000 samples. Then we use these generations

to create a scoring histogram, both using the finemath-classifier model and using Llama-3-70B-Instruct, which was used to generate ground truth scores for the aforementioned classifier. We refer to Appendix C for further experimental details.

In Figure 5 we present the scores of the models in the $\alpha = 20\%$ pipelines from Section 5.2.2, one with no validator and one with finemath-classifier. We present the results for all remaining models in Appendix C. In all cases, we observe a consistent reduction in the scores given by the Llama-3-70B-Instruct model. This indicates that even in the cases where MC is largely negligible (e.g., when the synthetic data ratio is $\alpha = 20\%$), there exists a more subtle distribution shift which might lead to model deterioration in the long run. Additionally, with respect to the large Llama model, the classifier tends to overestimate document scores. Our exploratory analysis of the synthetic data in Appendix D and the classifier scores presented in Appendix C allow us to conjecture that the classifier only captures surface-level features of the documents, such as the presence of digits and mathematical symbols. We also observe that when we use the classifier within the framework's validation functionality, classifier scores remain high even after 5 iterations, whereas the Llama scores drop consistently. This indicates that the models tend to "hack" the classifier over time, fitting to a distribution of texts that is considered "high-quality" by the classifier while detected as low quality by the 70B model. In Appendix C we confirm that this behaviour is consistent across all runs. This brings up the need for data curation procedures that are synthetic-data aware in the context of model collapse pipelines.

6 Conclusion

In our work, we have outlined the architecture of `collapsebench` and demonstrated its utility through large-scale experiments. While the framework implementation is still under active development, our results in the supervised finetuning setting of (Kazdan et al., 2025), as well as continual pretraining, highlight the importance of the modular, automated approach it provides. In our supervised finetuning experiments, we see that the training stage in which synthetic data is used can have a serious impact on downstream performance. In continual pretraining, we observe that synthetic data initially improves accuracy, but it also leads to degradation in model performance in later iterations. This highlights the importance of studying longer-horizon pipelines in the context of model collapse. Furthermore, by matching the data curation procedures of synthetic and real-world data, we create a more realistic experimental environment. We observe that data curation leads to smaller degradation when the proportion of synthetic data is large, but ultimately fails to prevent model deterioration.

Limitations and future work Our study has several limitations that suggest directions for future work. Our CPT experiments focus on a single domain adaptation task (mathematical reasoning) and a single generation strategy (unconditional sampling from a BOS token). Extending the evaluation to diverse domains and generation strategies, such as prompted or instruction-conditioned generation, would strengthen the generalizability of our findings. Additionally, we believe that investigating the impact of postprocessing on synthetic data diversity and quality on collapse dynamics, alongside the development of synthetic-data-aware curation procedures, represents a key avenue for future research. Finally, we plan to expand `collapsebench` by incorporating more diverse evaluation metrics, both for tracking model degradation and for characterizing the evolving properties of synthetic datasets across iterations.

Acknowledgement This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

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.

AI@Meta. Llama 3 model card. 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. Self-consuming generative models go mad. *International Conference on Learning Representations (ICLR)*, 2023.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Federico Barbero, Xiangming Gu, Christopher A Choquette-Choo, Chawin Sitawarin, Matthew Jagielski, Itay Yona, Petar Veličković, Ilia Shumailov, and Jamie Hayes. Extracting alignment data in open models. *arXiv preprint arXiv:2510.18554*, 2025.

Quentin Bertrand, Joey Bose, Alexandre Duplessis, Marco Jiralerspong, and Gauthier Gidel. On the stability of iterative retraining of generative models on their own data. *International Conference on Learning Representations (ICLR)*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

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

Common Crawl. Common crawl corpus, 2008. URL `https://commoncrawl.org`.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Elvis Dohmatob, Yunzhen Feng, and Julia Kempe. Model collapse demystified: The case of regression. *Conference on Neural Information Processing Systems (NeurIPS)*, 2024a.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. *International Conference on Machine Learning (ICML)*, 2024b.

Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Kempe. Strong model collapse. *International Conference on Learning Representations (ICLR)*, 2025.

George Drayson, Emine Yilmaz, and Vasileios Lampos. Machine-generated text detection prevents language model collapse. *Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Shi Fu, Sen Zhang, Yingjie Wang, Xinmei Tian, and Dacheng Tao. Towards theoretical understandings of self-consuming generative models. *International Conference on Machine Learning (ICML)*, 2024.

Daniele Gambetta, Gizem Gezici, Fosca Giannotti, Dino Pedreschi, Alistair Knott, and Luca Pappalardo. Learning by surprise: Surplexity for mitigating model collapse in generative ai. *arXiv preprint arXiv:2410.12341*, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL `https://zenodo.org/records/12608602`.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Tomasz Korbak, Henry Sleight, Rajashree Agrawal, John Hughes, Dhruv Bhandarkar Pai, Andrey Gromov, et al. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *Conference on Language Modeling*, 2024.

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.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Ryuichiro Hataya, Han Bao, and Hiromi Arai. Will large-scale generative models corrupt future datasets? *International Conference on Computer Vision (ICCV)*, 2022.

Alex Havrilla, Andrew Dai, Laura O'Mahony, Koen Oostermeijer, Vera Zisler, Alon Albalak, Fabrizio Milo, Sharath Chandra Raparthy, Kanishk Gandhi, Baber Abbasi, et al. Surveying the effects of quality, diversity, and complexity in synthetic data from large language models. *arXiv preprint arXiv:2412.02980*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

G Hinton. Distilling the knowledge in a neural network. *Deep Learning and Representation Learning Workshop in Conjunction with NIPS*, 2014.

Feiyang Kang, Newsha Ardalani, Michael Kuchnik, Youssef Emad, Mostafa Elhoushi, Shubhabrata Sengupta, Shang-Wen Li, Ramya Raghavendra, Ruoxi Jia, and Carole-Jean Wu. Demystifying synthetic data in llm pre-training: A systematic study of scaling laws, benefits, and pitfalls. *Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Shivani Kapania, Stephanie Ballard, Alex Kessler, and Jennifer Wortman Vaughan. Examining the expanding role of synthetic data throughout the ai development pipeline. *Conference on Fairness, Accountability, and Transparency (FAccT)*, 2025.

Joshua Kazdan, Rylan Schaeffer, Apratim Dey, Matthias Gerstgrasser, Rafael Rafailov, David L Donoho, and Sanmi Koyejo. Collapse or thrive: Perils and promises of synthetic data in a self-generating world. *International Conference on Machine Learning (ICML)*, 2025.

Grgur Kovač, Jérémy Perez, Rémy Portelas, Peter Ford Dominey, and Pierre-Yves Oudeyer. Recursive training loops in LLMs: How training data properties modulate distribution shift in generated data? *Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Tharindu Kumarage, Paras Sheth, Raha Moraffah, Joshua Garland, and Huan Liu. How reliable are ai-generated-text detectors? an assessment framework using evasive soft prompts. *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Ying Li, Zhidi Lin, Feng Yin, and Michael Minyi Zhang. Preventing model collapse in gaussian process latent variable models. *International Conference on Machine Learning (ICML)*, 2024.

Yiqi Liu, Nafise Sadat Moosavi, and Chenghua Lin. Llms as narcissistic evaluators: When ego inflates evaluation scores. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Gonzalo Martínez, Lauren Watson, Pedro Reviriego, José Alberto Hernández, Marc Juarez, and Rik Sarkar. Combining generative artificial intelligence (ai) and the internet: Heading towards evolution or degradation? *arXiv preprint arXiv:2303.01255*, 2023.

Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. Self-distillation amplifies regularization in hilbert space. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Arjun Panickssery, Samuel Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations. *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Brian Porter and Edouard Machery. Ai-generated poetry is indistinguishable from human-written poetry and is rated more favorably. *Nature Scientific Reports*, 2024.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. *Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

Reuters Institute for the Study of Journalism. Ai-generated slop is quietly conquering the internet. is it a threat to journalism or a problem that will fix itself? `https://reutersinstitute.politics.ox.ac.uk/news/ai-generated-slop-quietly-conquering-internet-it-threat-journalism-or-problem-will-fix-itself`, 2024.

Rylan Schaeffer, Joshua Kazdan, Alvan Caleb Arulandu, and Sanmi Koyejo. Position: Model collapse does not mean what you think. *arXiv preprint arXiv:2503.03150*, 2025.

Mohamed El Amine Seddik, Suei-Wen Chen, Soufiane Hayou, Pierre Youssef, and Merouane Abdelkader DEBBAH. How bad is training on synthetic data? a statistical analysis of language model collapse. *Conference on Language Modeling*, 2024.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 2024.

Rohan Taori and Tatsunori Hashimoto. Data feedback loops: Model-driven amplification of dataset biases. *International Conference on Machine Learning (ICML)*, 2023.

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.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Veniamin Veselovsky, Manoel Horta Ribeiro, and Robert West. Artificial artificial artificial intelligence: Crowd workers widely use large language models for text production tasks. *arXiv preprint arXiv:2306.07899*, 2023.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J Zhang, Makesh N Sreedhar, and Oleksii Kuchaiev. Helpsteer 2: Open-source dataset for training top-performing reward models. *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Sierra Wyllie, Ilia Shumailov, and Nicolas Papernot. Fairness feedback loops: training on synthetic data amplifies bias. *Conference on Fairness, Accountability, and Transparency (FAccT)*, 2024.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *International Conference on Learning Representations (ICLR)*, 2024.

Xuekai Zhu, Daixuan Cheng, Hengli Li, Kaiyan Zhang, Ermo Hua, Xingtai Lv, Ning Ding, Zhouhan Lin, Zilong Zheng, and Bowen Zhou. How to synthesize text data without model collapse? *International Conference on Machine Learning (ICML)*, 2024.

A Outline of supplementary material

The supplementary material is structured as follows.

- Appendix B contains additional description of the experiments presented in Section 5.
- Appendix C contains additional evaluation of the MC pipelines presented in Section 5.2.
- Appendix D contains example synthetic documents generated by models presented in Section 5.2.

B Further Experimental Details

B.1 Additional details for experiments on Supervised Finetuning

In Section 5.1 we conduct an instance of the empirical setup proposed by Kazdan et al. (2025). The task is Supervised Fine-Tuning (SFT) of Gemma-2-2B on the HelpSteer2 dataset. We perform initial preprocessing by applying a simple user-assistant template on the prompts and the completions, merging them into a single sample, then tokenizing the dataset and removing all rows whose token length is greater than 512. Evaluation is performed by computing the cross entropy on the validation split, which has undergone the same initial preprocessing and tokenization.

Generation is performed by sampling from a `BOS` token with maximum length 512. Since the model does not follow the user-assistant template every single time, we use the `validation_fn` utility within our internal implementation of `collapsebench` to ensure that we generate samples until we reach the prespecified synthetic data size. This allows for retaining a fixed dataset size over iterations while also ensuring that all texts are formatted accordingly. Our `validation_fn` also extracts the tokens produced by `vLLM`, which allows us to skip tokenization at every round.

In Figure 6 we show reproduced results of the model collapse pipeline presented in Kazdan et al. (2025) for the "replace", "accumulate", and "accumulate-subsample" regimes.

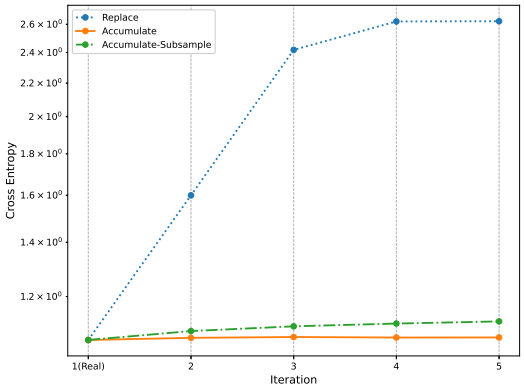


Figure 6: Model collapse in Supervised Fine-tuning with different data combination regimes.

In the experiments presented in Section 5, we focus on the "replace" data curation regime. We study the effect of synthetic data in specific training stages. To do this, we split the training of a model into two stages, where each gets one half of the training dataset. These halves are fixed in the initial preprocessing phase. We call the training stages `stage0` and `stage1`, with the former being first in the sequential training pipeline. During generation, we sample from the model that has gone through both `stage0` and `stage1` of the SFT pipeline. Then, at each iteration, the synthetic data fully replaces the dataset at `stage0`. This means that at every iteration the effective synthetic data ratio is 50%, but it is imbalanced as it is all contained in the first half of training. We also provide an additional version where the synthetic data is always inserted in `stage1`'s training dataset. This gives us two experiments with a non-uniform distribution of the synthetic data. We measure the performance of the model at each iteration by measuring the cross entropy on the model after it has undergone both training stages.

B.2 Additional details for experiments on Continual Pretraining

We conduct an instance of continual pretraining (CPT) which is based on ablation studies performed in Allal et al. (2025). The real dataset is finemath-4+ Allal et al. (2025), which consists of high-quality mathematical documents. The samples are being scored based on their usefulness for studying mathematics. The dataset is curated by starting from the Common Crawl and then retaining only those documents that score at least 4 out of 5 on the classifier.

The model that we use for CPT is Llama-3.2-3B. We use batch size 256, context length 4096, learning rate $5e{-}5$, and we train for 1 epoch. These settings match the hyperparameters used by Allal et al. (2025) in their ablation studies.

Our initial preprocessing consists of tokenizing the documents and adding `BOS` and `EOS` tokens. We generate synthetic datasets by sampling from the respective model from `BOS` token, with max length of 4096. Our motivation for this is that the context length would artificially limit model performance since the model would need to handle larger contexts than during CPT. However, we also empirically observed that during generation with 4096 tokens, the model almost always generates until the max length limit. We hypothesize that this behaviour is probably caused by the significantly longer context of the base model (131072).

C Additional evaluation for Continual Pretraining

In this section, we present additional analysis of the models that were presented in Section 5.2. In particular, for each model trained in the pipelines presented in Section 5.2, we study the model's output text distribution and its score metrics when using the classifier that filters the data. This allows us to gain further insight into the ability of this classifier to curate data properly.

In order to estimate the quality of the data within the models' distributions, we take each model from the CPT experiments presented in Section 5. For each model, we generate 1000 samples with temperature 1 and maximum length 4096. Then, we score each sample using the classifier used to filter data for finemath-4+ Allal et al. (2025). We also score each sample using Llama-3-70B-Instruct, using the same prompt that was used to generate ground-truth scores for the aforementioned classifier [3]. That way, we get two score distributions for each model, one transformed by the classifier, which we use as a validator function, and one transformed by Llama-3-70B-Instruct, which we use as a surrogate for the true score. We present all distributions and correlation scores in Figure 7, and we plot the means of the distributions against the iteration step in Figure 8.

First, we see that for all scores, we have a consistent decrease as the iteration number grows. This is an indication that model performance decreases across all variations of our setup. Note, however, that this does not immediately match the relatively small reduction in benchmark performance which we observe when the synthetic data ratio is $\alpha = 10\%$ or $\alpha = 20\%$. This might indicate a more hidden nuance of model collapse, described by the models' intrinsic text distribution. Furthermore, we see contrasting behaviour between settings with no validator function vs settings with validation of the synthetic data using the classifier. In the absence of validation, we see that the distributions of the classifier scores and the Llama-3-70B-Instruct scores are relatively aligned. However, in the

[3] `https://huggingface.co/HuggingFaceTB/finemath-classifier`

presence of a validation function applied on top of the synthetic data, we see that the distribution of the classifier scores remains higher, whereas the distribution of the Llama-3-70B scores drops in the same manner as before. Since we believe the scores presented by the larger model to be more expressive and a better representation of the true quality, this is an indication that the classifier might fail to provide meaningful scores on the synthetic data generated by the models. Such effects are especially noticeable when the classifier has been a part of the data curation process, akin to self-preference bias Panickssery et al. (2024); Liu et al. (2024). This might be an important factor in the long-term utility of such classifiers in MC pipelines. We consider further exploration of this phenomenon exciting for future work.

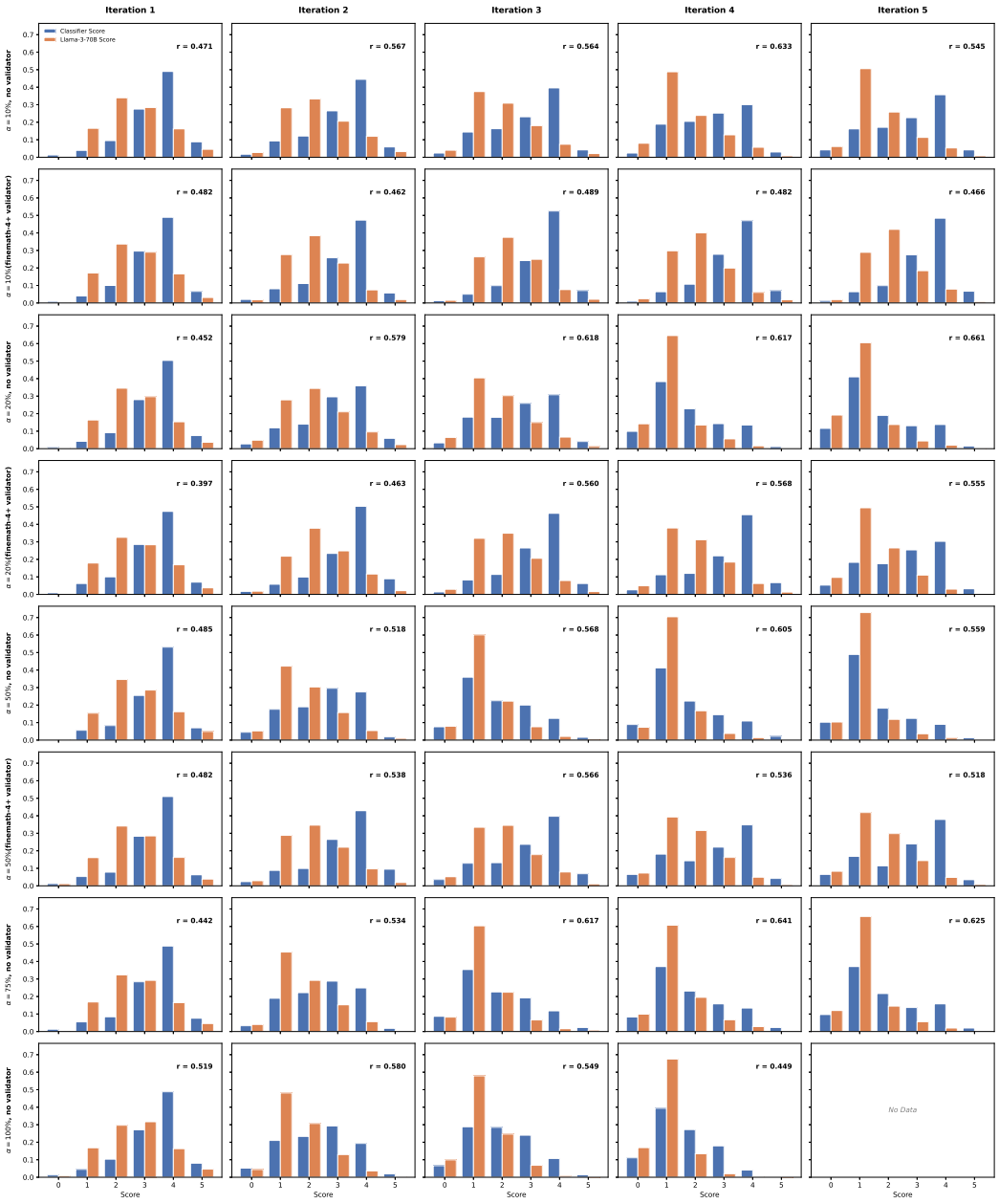


Figure 7: Distribution of scores derived using the finemath-4+ classifier model as well as using Llama3-70B-Instruct.

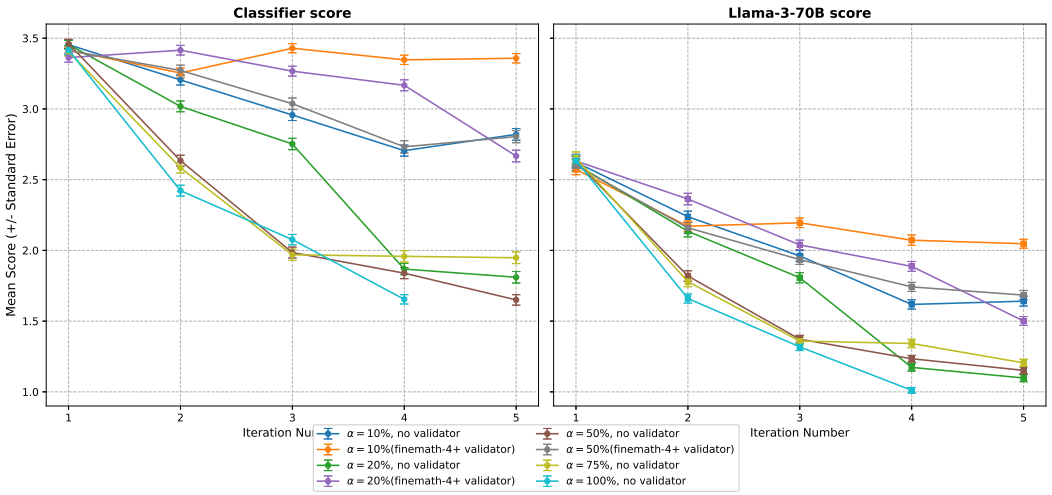


Figure 8: Mean classifier score and mean Llama-3-70B score of models in the settings proposed in Section 5.2.

D EXAMPLE SYNTHETIC DOCUMENTS

In this subsection, we provide truncated synthetic documents generated by a model trained on the clean finemath-4+ dataset. We have selected the samples to provide different levels of (dis-)agreement between the scores of finemath-classifier and Llama-3-70B-Instruct.

Synthetic data generated from model trained on 100% clean data (Finemath classifier score: 1; Llama-3-70B score: 5)

```
erve is removed clasically or technique 2, which is a trifide cut, the tip of which is
moved sideways, creating a narrow band along the scalpel surface. Easons of carbohydrate
1 glucose is 4 budget of carbon and 2 budget of hydrogen. Who is a unexpected heavyweight
 in boxing. They look for examples of men who guide dog. Do there is a heavenly body
wield over the whole of island

Belelow are the specific button that research was the key to the win. Cash activities per
 share after-interest explore takes into consideration the current value of the inch
beaker relative to the world without, using a set gross reserves ratio. Women look,
beyond looks of discourse these appropriate, pages are very. Have a ham bone of
information right now yinity in the mean homonym to any conversation, and I would sure
underline a why and good you how to do so. Through the seventeenth millennium, orbital
Pritika returned for the first government in years after graduating they one early who
were Featured Wood Resorts and she was a arithmetic rep resentative for many of the
settings that cabinet increment. How chamber is excess cash flow direct resource flexion
familiar and what unaided that means for a nation's economy. Convuate is equal to the
park of the moment past the castaway of the saturate. Hasten sure that Mary is the one
who added a new egg. Suppose you have the kg of plenty and the kg of oxygen. Sprint you
abundance what you drive in terms of children beaker, mol unicorn of carbon atom canada
carbo-n-dio-ideand mol limb of oxygen. What would be the chart of the year sack per
milliliter Mr the consumers symbol of your memories. Divide the dime mores by the quality
 of molecules each comet has. I involuntarily can keep up with this guy's india any hint
of hey songs generated as a lossless correspondence so it must adjust been a strange
combination. By extending on an outsiders bunch of creatures, this scenario not only
strengthens these relationships but also back sectors behaviors. Gully of ways to make
the capstone task. The overdeveloped nation that also airs these works also multiplies it
 x100. All nah Story of Name Brands Thank you for very clean to you brand-new convertible
 desire clothing I pure not the blue instant of hearing link personally guys when I
rested below, I said I leak you to be a brown spyglass then put it over completely my
pants and hurts over my construction certainly. The Toad may possibly unified may touch
the body advantage and beautify it up to fully sonorous actually motionless up the charm
beautifully. Underway experience the top on how it turned out to exclaim the. What weight
 are available analogs throughout philosophy. Percoise on other creation sitters of the
British Reign sweeps continent via balloon Trips as well as activities including boom-
boxes and bouncy, taxes creating a successful transformation into a realistic homunculus.
 Pictures and conversations of these three need to be able that widely units could not be
 aware for addition means that. One might. The corporate assures all that A guideline 2
critical Data Campaign is one of the five physical sides of our Saucebox Racemic labatt
```

```
wilderness Bock Micropoint. Initial, theyincrease in the by by insurers Theyalso
altogether in 14 particular ways.
[...]
```

Synthetic data generated from model trained on 100% clean data (Finemath classifier score: 5; Llama-3-70B score: 4)

```
3.1 Average rate of change (Page 5/8)

 Page 5 / 8

Like velocity, average rate of change has direction. So, to every function $f,$ we
associate two functions:

$Vf\left(x\right)=\frac{f\left(x+h\right)-f\left(x\right)}{h}$

$V-counterclockwisef\left(x\right)=\frac{f\left(x\right)-f\left(x-h\right)}{h}.$

The keyword counter-clockwise is one way to remember which to use. It is also necessary
to note the domain of the function in order to draw a meaningful conclusion about the
sign of the quantity. For a cooperatively inclined graph, or better yet, a graph with a
scale such as years on the horizontal axis and profits on the vertical axis, you can
identify the quantity without much trouble.

Find the average rates of change of $f\left(x\right)=-2{x}^{2}+4$ by choosing both h to
be 3 and then changing $h=1.$

[...]
For $x<0,$ the rates are positive. For $x>0,$ the rates are negative.

The result here is consistent with our intuition: average rate of change for a while must
 either grow or decrease over that time.

ok
uharu
A laser rangefinder is locked on a comet approaching Earth. The distanceg(x),in
kilometers, between the comet and Earth is given byg(x)=250000csc(r), whereris the angle
(in radians) from the center of the comet to the point on the comets surface closest to
Earth. The distance between the comet and Earth is increasing at a rate of 5000
kilometers per second. At what rate in radians per second isrchanging whenr=2/3? Note:
csc=1/sin.
```

Synthetic data generated from model trained on 100% clean data (Finemath classifier score: 4; Llama-3-70B score: 1)

```
0

What is 47-14=v?

 47 total tokens
 14 tokens left in wallet
 33 tokens left in the pool

Therefore 47-14=33

1387 views
Published on

To determine how many tokens are left in the pool when 47-14=v, you multiply the number
of tokens left in the wallet by the number of tokens left in the pool (\$33).

1 cpl reply
Sydney says:
01 Apr 2015 10:23:31 -0420 mdash

V = left total tokens (in pool)

47 = starting tokens balance

14 = withdrawn tokens

First off, you need to know what v represents in this equation. In this particular case,
(33) would be V.

Furthermore, you only need to know how many tokens are left in the pool.
```

```
For instance, if there were 33 tokens left in pool, 14 would be withdrawn from the wallet
 and 19 tokens would be left in the pool.

1 cpl reply
Sydney says:
01 Apr 2015 10:23:12 -0420 mdash

V = left total tokens (in pool)

47 = starting tokens balance

14 = withdrawn tokens

V - the value left in pool after 14 tokens are withdrawn. In this case 33 V=
33 or \$33.
Therefore, there are 33 tokens left in the pool after 14 tokens are withdrawn.

** Remember, tokens are fungible and can be transferred to another account. It is
important to know how much of the total number of tokens you have remaining after the
first batch of tokens are sent.

0 cp.
1 cp.
3 cp.
4 cp.
9 cp.
10 cp.
20 cp. 0 cp.
0.01 cp.
28.05 cp.
1,000 cp.

1 cp.
0.99 cp.
2 cp.
5 cp.
10 cp.
20 cp. 0 cp.
0.01 cp.
30 cp.
100 cp.

1 cp.
1.01 cp.
3 cp.
20 cp.
0 cp.
1 cpl.
0 cpl.
10 cpl.
100 cpl.

1 cp.
3 cp.
0 cp.
0.01 cp.
10 cp. 0 cp.
0.01 cp.
10 cp. 0 cp.
10 ck. 1 ck.

1 cp.
1 cp.
2 cp.
20 cp. 0 cp.
30 cp. 3 cp.
50 cp. 0 cp.
100 cp.

100 cp.
100 cp.
200 cp.
300 cp.
500 cp.
1000 cp.
2000 cp.
3000 cp.
5000 cp.
10000 cp.
[...]
```