Explorations on Catastrophic Forgetting Mitigation Curriculum

Aaron Zheng
aaronz@berkeley.edu

Zhangzhi Xiong
xiongzhzh2023@berkeley.edu

Andy Zhang
zhangnd16@berkeley.edu

Jason Lee
jason_lee@berkeley.edu

Tianyu Gu
gty2005@berkeley.edu

Abstract

Catastrophic forgetting (CF), the phenomenon that occurs when a model forgets previously learned data as a result of learning new data, poses a significant barrier to lifelong learning of LLMs, where many factors of fine-tuning tasks within different domains can ultimately degrade the performance on previously learned tasks. In this project report, we propose different curriculum designs to mitigate catastrophic forgetting and conduct experiments to verify the effectiveness of these approaches. Subsequently, we investigate whether curriculum design exhibits a scalable transferability effect by using a smaller LLM to approximate the ideal curriculum for a larger LLM from the same family to reduce training power consumption and increase downstream performance. Our code is available online [1]

1 Introduction and Related Work

Catastrophic forgetting occurs when you train a model sequentially on different datasets, and as training continues, the model tends to perform poorer and poorer on previously learned data. The general consensus of the AI research community indicates it is paramount to build/train models that can learn new tasks sequentially without forgetting previously learned knowledge [1]. In Continual Learning (CL), a model needs to learn a series of tasks sequentially with the objective of learning new tasks without forgetting old ones. Hence, catastrophic forgetting is commonly viewed as a harmful problem that needs to be overcome [2] [3]. A classic example of catastrophic forgetting is the distribution shift phenomenon in binary classification task [4] shown in Figure 1. It is revealed that in continual learning, the 'order' itself in which a model learns a series of tasks, significantly affects the degree of catastrophic forgetting [5].

Curriculum learning has been applied to finetuning to mitigate catastrophic forgetting phenomenon. This is highly favorable since it doesn't pose additional computation burden to the finetuning process. Kim and Lee [6] investigated a "start-easy-move-hard" approach, comparing the effects of sorting data by length, loss, and attention scores against random data shuffling. Notably, they proposed a hybrid strategy: random training for the initial epoch followed by structured curriculum training in subsequent epochs. Their findings

[1]https://github.com/bearthesilly/UCB_CS182_project

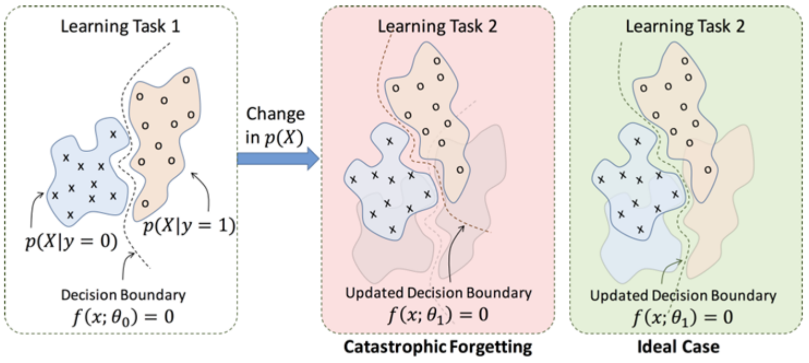

Figure 1: Depiction of catastrophic forgetting in binary classification tasks when there is a distribution shift from an initial task to a secondary task. [4]

indicate that even within one finetuning task, the arrangement of data feeding from this dataset can have great impact.

Kirkpatrick et al. (2017) propose Elastic Weight Consolidation (EWC), a method inspired by synaptic consolidation in the brain to mitigate catastrophic forgetting in neural networks trained on sequential tasks. By estimating the Fisher information matrix to identify the parameters most critical to previously learned tasks and penalizing their deviation during new learning, EWC preserves prior knowledge without requiring past data. The authors demonstrate its effectiveness on sequential MNIST classification and Atari reinforcement-learning tasks, showing that networks can acquire new skills while retaining earlier ones, marking a significant step toward CL in artificial systems. [1]

Ramasesh et al. identified that catastrophic forgetting depends on task similarity on a U-shaped relationship. Forgetting is minimal when tasks are either very similar or drastically different, but maximal when tasks demonstrate ”intermediate similarity”. Ramasesh et al. formalized this with a feature overlap matrix $\Theta(x, x')$, and showed that catastrophic forgetting reached maximum when $\Theta(x, x')$ is moderate. [7]

Appropriate model selection has an effect in catastrophic forgetting. Models like Phi-3.5-mini effectively minimize forgetting while maintaining learning capabilities. Prompt engineering and fine-tuning strategies significantly impact model performance in continual learning settings. Models such as Orca-2-7b and Qwen2.5-7B showed strong learning abilities but varied in forgetting. [8] Careful model selection and tuning can enhance handling multiple tasks without sacrificing accuracy, which is crucial for developing autonomous LLM-based agents. [8]

In recent studies, it was demonstrated that the problem of determining the optimal parameters to avoid the CF in a fixed model can be reduced to the well-known Satisfiability (SAT) problem. Consequently, this proof categorizes the CF problem within the class of NP-HARD problems. Despite the NP-HARD nature of CF, significant progress has been made in mitigating CF within Deep Neural Network (DNN) models through various techniques and heuristics. [9]

One of them includes adding a new loss term. In the field of LLMs, it is discovered that there is a high correlation between the sharpness of the loss landscape and the tendency for CF, where a flatter loss landscape leads to lower likelihood of CF [10]. Another way to prevent CF is to preserve weights that are important for the first task (the task that may be forgotten) [11]. After training on the first task, calculate the aggregated gradient of loss against each weight, and cutoff based on some threshold. Then freeze the weights that have a large gradient (meaning they are important to the first task) when training the second task.

Some studies [12] [13] [14] also shows that we can also use reinforcement learning perspective to view continual training and catastrophic forgetting. Specifically, we view the LLM as an agent, the input x as the state s, and the generated token sequence y as the action a. Under this formulation, sequential fine-tuning is equivalent to an agent adapting to a non-stationary environment where the state distribution shifts abruptly from $P(s)_{T_i}$ to $P(s)_{T_j}$ where $T_{i,j}$ stands for different tasks, which can be the major cause of the CF phenomenon. In RL, stability is typically maintained via an Experience Replay Buffer $\mathcal{B}$ [15]. For instance, the replay mechanism in the CORE paper [13] presented in Figure 2, is a strategy intuitively designed based on replay buffer perspective.

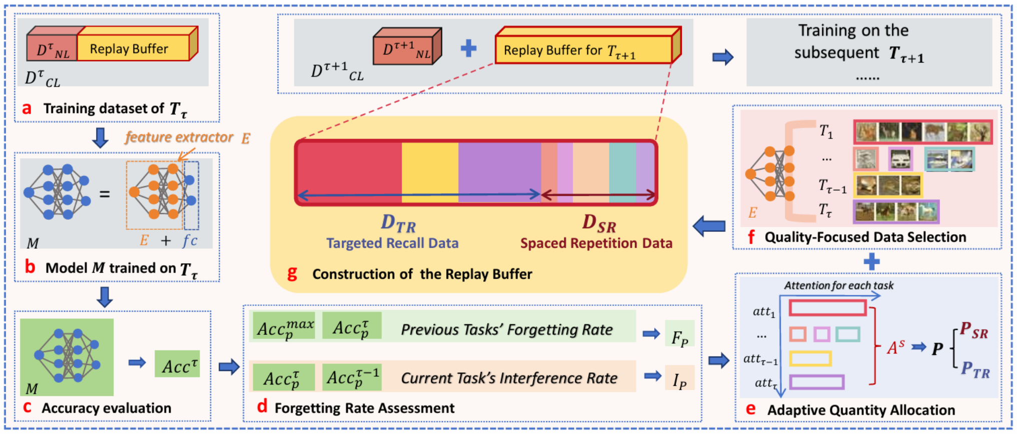

Figure 2: Pipeline of CORE which leverages playback mechanism in the curriculum [13]

Low-Rank Adaptation (LoRA) fine-tunes large language models by freezing all pretrained weights and introducing a trainable low-rank update of the form $W + BA$, where $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ with $r \ll d, k$, enabling efficient adaptation within a constrained subspace of weight changes [16]. Since LoRA restricts how much the underlying model parameters can drift during fine-tuning, it preserves most of the pretrained model's behavior while learning new skills. Recent work discovers that LoRA not only limits overfitting but also substantially reduces catastrophic forgetting. In other words, models fine-tuned using LoRA retain markedly more of their original capabilities compared to fully fine-tuned models when evaluated on out-of-domain tasks [17]. As far as we are aware, there has been no research investigating the effects and best practices exploring the effects of catastrophic forgetting and sequential fine-tuning using LoRA.

2 Problem Statement

We consider a sequential learning setup involving a pre-trained Language Model (LM), parameterized by θ. The model is required to learn a sequence of tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$. Each task T_k is associated with a dataset $\mathcal{D}_k = \{(x, y)\}$, where x represents the instruction/input and y represents the target output. Due to compute resource and time limits, we set k to 2 and regard finetuning on a different dataset as a different task for convenience, and therefore name them Task A and Task B.

After deciding the base experiment setting, firstly we will try to reproduce Catastrophic Forgetting phenomenon. Then our goal is to explore different strategies to mitigate catastrophic forgetting, i.e., to learn these two tasks well at the same time. Lastly, we can design a curriculum according to these strategies and investigate whether curriculum design exhibits a scaling effect.

3 Hypothesis

We hypothesize that catastrophic forgetting in Language Models is driven by the abrupt distribution shift from RL perspective and the optimization conflict between the previous task (Task A) and the current task (Task B) from optimization perspective. More specifically for the optimization perspective, we suspect that sequential finetuning leads to parameter moving towards a local optimum specific to Task B, which can be completely distinct from the optimal region of Task A. We formulate the following hypotheses:

H1: Interleaving mini-batches of Task A and Task B (rather than training them sequentially) can mitigate catastrophic forgetting since we empirically believe that interleaving forces the optimization process to satisfy the constraints of both tasks simultaneously.

H2: A gradual transition strategy—mixing a small ratio of Task B 'preview' data into Task A training, and retaining a small ratio of Task A 'review' data during Task B training can outperform baseline. This is inspired by how human design curriculum for studying multiple tasks asynchronously.

H3: The severity of CF is negatively correlated with the semantic similarity between tasks.

H4: Smoothing the loss landscape via additional reward terms or regularization will improve the model's robustness to forgetting. Reshaping loss landscape may help mitigate this issue.

H5: Varying the method used to select which data gets placed in the replay buffer (i.e. difficulty, random selection) under some budget constraint (limited pieces of data to choose). The goal is to gain insight on how the data in your replay buffer affects your model's ability to retain information.

4 Method

4.1 Reproduction of CF

To establish a clear foundation for our study, we first reproduce the catastrophic forgetting phenomenon under controlled environments. This section outlines two complementary experiments for our subsequent experiments to improve upon.

4.1.1 Toy Experiment

We employed a text classification setup (distilbert-base-uncased model [18] on the 4-class AG News dataset [19]) to construct two mutually exclusive sub-tasks. More specifically, we split 4-class AG News dataset [19] into two sub-dataset, one containing class 'World' and 'Sports' labeled as Task A, and one containing class 'Business' and 'Science' labeled as Task B. We add a 4-class classifier as the task head and unfreeze the DistilBERT weights, i.e., we perform a full finetuning.

4.1.2 Sequential Training (Baseline)

In this procedure that will serve as our baseline, we employ a method we will call Sequential Training. Sequential Training consists of first training on solely Task A and then solely Task B. We declare this as our baseline because this strategy is the most straightforward way to organize the training data. We denote the finetuning procedure on MATH as *phase 1* and subsequent finetuning on HotpotQA as *phase 2*. Our expected results for this experiment are that during *phase 1* the model can learn well on MATH, whereas during *phase 2* the model demonstrates a degrading performance on MATH as it tries to fit the challenging HotPotQA data. In our base experiment, we use TinyLlama-1.1B-Chat-v1.0 [20] as the language model. For the training order, we use MATH [21] as the first dataset and HotpotQA [22] as the second dataset.

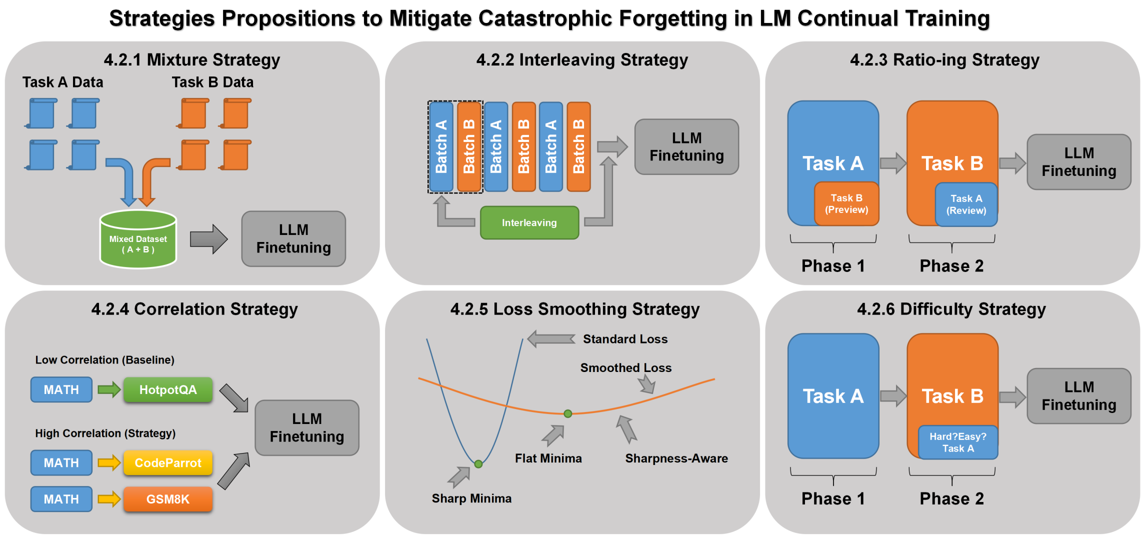

Figure 3: Overview of Proposed Strategies

4.2 Strategy Exploration

Drawing inspiration from the relationship between continual learning in language models and reinforcement learning, along with an intuitive understanding of catastrophic forgetting, our objective is to explore various potential strategies to mitigate this phenomenon. All mitigation experiments are evaluated independently. Specifically, each experiment introduced one single strategic modification to the same exact initial conditions/settings to isolate the method's effect. The visualized overview of our proposed strategies is presented in Figure 3.

4.2.1 Mixture Strategy

From RL perspective, in the baseline, model is exposed to two different task environments in an order, which can lead to a distribution shift [12] as mentioned previously. We therefore propose the construction of a unified environment that encompasses both tasks. Hence we propose a 'Mixture' strategy where we mix these two datasets together into one and use that as the dataset to finetune. The model will be tuned in a way that attempts to perform simultaneously well in both tasks. Note that the concept $phase$ is not applied to this strategy.

4.2.2 Interleaving Strategy

This strategy still inherits the core intuition of creating a comprehensive task environment when finetuning. Instead of directly mixing two datasets into one, we interleave Task A batch and Task B batch during training. In another word, instead of finetuning on Task A batches during *phase 1* and on Task B batches during *phase 2*, we expose model to batches from 'first A and then B' in an iterative way. We hope that this strategy can help prevent the model from overfitting to one single domain's local minima. The model parameters will be tuned toward two tasks' distribution alternately at batch level, eventually forming an equilibrium.

Note that Mixture and Interleaving Strategies seem to be similar, but they are actually distinct. Please refer to pseudocode in Appendix.B1 2 3.

4.2.3 Ratio-ing Strategy

Despite the overall fine-tuning order (Task A $\rightarrow$ Task B) is maintained, the fine-tuning process is still divided into two $phase$s, allowing a small ratio of 'data swapping and mixing'. In the *phase 1*, the data consists of mostly Task A samples (e.g., 80% Task A and 20% Task B), while in *phase 2*, the ratio is inverted (20% Task A and 80% Task B). This setup allows us to test how mixing in Task B data into the task A training step (and vice versa in task B's training) increases or reduces the threshold for Catastrophic Forgetting (CF). In this strategy, we can intuitively interpret the small proportion of Task B data in *phase 1* as 'preview' and the small proportion of Task A in *phase 2* as 'review'.

4.2.4 Correlation Strategy

Through replacing the "QA" dataset with a different dataset such as "coding", or any other heavily logic-based dataset, the policy (weights) learned in the first environment should be more robust to updates from the second. Specifically, we argue that mathematical reasoning and code generation share a high degree of similarity—both require strict syntactic adherence, logical derivation, and step-by-step reasoning. By contrast, open QA (like HotpotQA [22]) relies more on knowledge retrieval and natural language comprehension. Therefore, we believe that replacing a low-correlation task with a high-correlation task will mitigate the distribution shift and hence mitigate CF. To investigate this, we designed a comparative experiment keeping the language model and the first task (Math) identical. We contrast three sequential fine-tuning scenarios:

$$\text{Baseline (Low Correlation):} \quad \text{MATH} \rightarrow \text{HotpotQA}$$

$$\text{Correlated Setting (High Correlation):} \quad \text{MATH} \rightarrow \text{CodeParrot or GSM8K}$$

We select CodeParrot [23] dataset as a proxy for the coding task and GSM8K [24] as a proxy for another math task, which we believe possess stronger correlation to the MATH dataset compared to HotpotQA.

4.2.5 Loss Smoothing Strategy

While the correlation strategy aims to align task distributions, loss smoothing focuses on stabilizing the dynamics of optimization itself during the sequential fine-tuning process. We draw inspiration from the findings of Li et al. [10], which showed that catastrophic forgetting is strongly related to the sharpness of the loss landscape. Specifically, when training on the second task pushes gradients of the model towards sharp minima, previous learned representations are overwritten easily.

Following this insight, we devise schemes to moderate the current direction and magnitude of the loss. The current approach is to **integrate** the loss with recent historical loss values: a momentum parameter. Please refer to pseudocode 4. This has the effect of steering the model away from sharp minima and guiding optimization towards a flatter region of the loss. We hope to see whether loss smoothing helps mitigate catastrophic forgetting by promoting gradual adaptation. In addition, the paper also highlights a more principled approach to mitigate catastrophic forgetting, by directly reshaping the loss curve. Sharpness-Aware Minimization (SAM) explicitly counteracts CF by optimizing the worst-case loss within a small neighborhood of the parameter space—penalizing directions in which the loss increases rapidly. By doing so, SAM discourages overly rapid gradient descent and promotes convergence to *flatter* minima, which leads to less CF as they demonstrated. We aim to explore on this approach as well.

4.2.6 Difficulty Strategy

Motivated by the analogy to human learning, we investigate how the difficulty of retained training examples influences a model's ability to preserve previously acquired knowledge during sequential fine-tuning, focusing on how different strategies for selecting replay data affect CF. Following the same setup as above, the model is first fine-tuned on the MATH dataset (*phase 1*), then fine-tuned on HotPotQA (*phase 2*). However, prior to *phase 2*'s training, we first concatenate the HotPotQA training data with a replay buffer consisting of 10% of the examples from the MATH *phase 1* training set. To compare replay-selection strategies, we quantify the model's retained mathematical reasoning ability using its validation performance on MATH.

We test five different strategies for selecting the *phase 1* data. Our baseline is a random selection strategy that randomly samples *phase 1* examples without replacement. We then compare this baseline to a difficulty-based strategy that ranks examples according to their human-assigned difficulty level; each problem in MATH is annotated with a "level" from 1 to 5, as determined by human expert annotators [21]. Since human difficulty may not align well with the model's internal difficulty signal, we also explore model-derived difficulty by sorting examples according to their per-example training loss. We consider both ascending-loss selection (favoring problems the model finds "easiest") and descending-loss selection (favoring problems the model finds "hardest"), as described in Algorithm 5.

4.3 Curriculum Design and Transfer

Based on experiments with the aforementioned strategies, we determine our curriculum — an aggregation of the best strategies for training an LLM based on our experiment results. We first establish a baseline using Llama2-7B [25], ensuring all other parameters remained unchanged. Subsequently, we implement our curriculum to Llama-7B baseline and conduct the main experiments. Choosing Llama2-7B as the larger model for experiment is to ensure consistency in the model architecture since Llama2-7B and TinyLlama-1.1B-Chat-v1.0 are in the same architectural family [20].

5 Experiment

5.1 Reproduction of CF

For the toy dataset experiment, the result is as Figure 4. Before the red borderline, from the blue line we can tell that the model has sufficient capacity to learn the classification for four classes of news, whereas orange lines shows that after learning well on 'World/Sports' news, model forgets them during training on 'Business/Sci' news. This result exactly demonstrates the catastrophic forgetting phenomenon.

And for the baseline experiment, Figure 5 demonstrates the validation monitoring curve during *phase 2*. We observed that loss on HotpotQA dropped and the eval-loss of MATH surged from a low starting point, indicating the model forgot MATH during training HotpotQA. This result is consistent with CF phenomenon.

5.2 Strategy Effectiveness Verification Results

The **Mixture** and **Interleaving** Strategy both discard two $phase$ training paradigm. The validation loss monitoring curves of these two experiment presented in Figure 5 both indicate that these two strategies can be helpful for mitigating CF. The figures for separate **Mixture**, baseline and **Interleaving** experiments are displayed in Figure 8, Figure 13 and Figure 14. The swapping ratio is an important parameter in the **Ratioing** Strategy. Therefore, we selected ratio of 10%, 20% and 50% to conduct controlled experiments, and also did experiments with ratio of 0% for reference. The plotted curve of validation loss in *phase 2* is presented in Figure 7. According to Figure 7, in *phase 2*, introducing mixing ratios can help mitigate this issue. At a 10% ratio, despite a slight initial rise, the loss growth was much flatter, ending at 1.03. Increasing the ratio

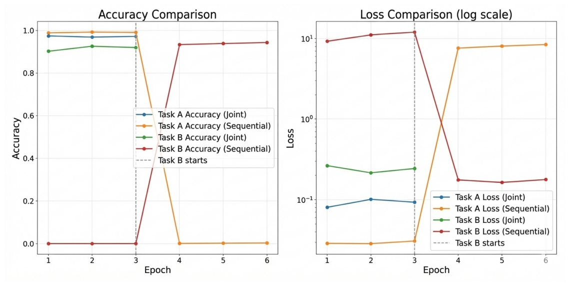

Figure 4: Result Gallery of Toy Experiment: Joint vs. Sequential Training

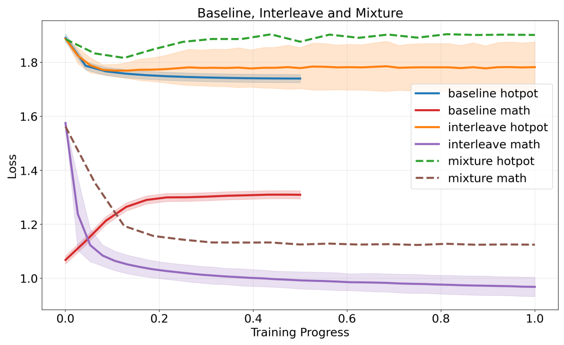

Figure 5: Result Gallery of Baseline, Mixture and Interleave

to 20% further improved retention, resulting in a final loss of 1.02. In contrast, the 50 % ratio does not cause an increase in loss; instead, the loss decreases by approximately 0.02. We can now confirm the effectiveness of **Ratio-ing** Strategy.

As for **Correlation** Strategy Experiment, we substitute HotpotQA task with CodeParrot [23] and GSM8K [24], respectively. The validation monitoring curve on both tasks during *phase 2* is presented in Figure 6. During *phase 2* training on both task datasets, the model demonstrated very similar CF phenomenon on the learned Math task as in the 'Math $\rightarrow$ HotpotQA' baseline. The validation loss curves converge to around 1.2 in all training pairs.

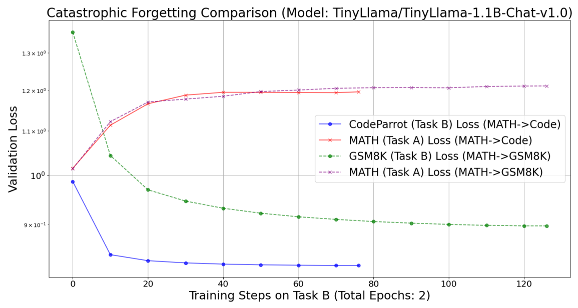

Figure 6: Result of **Correlation** Exp.

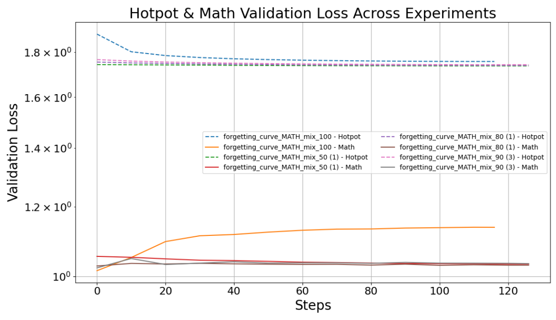

Figure 7: Comprehensive Result of **Ratio-ing** Exp.

For the **Loss Smoothing** experiment, as the validation monitoring curve during *phase 2* presented in Figure 9, we can conclude that auxiliary loss smoothing can't help ease CF phenomenon as the validation loss still surge to around 2.0 and shows zero mitigation compared to the baseline.

Moreover, the results from **Difficulty** experiment suggest only a minor improvement from using human-ranked difficulty selection over random selection based on the final MATH validation losses. From the experiment results displayed in 12, there is no variance in the MATH validation loss across the methods for the first twenty steps because we trained using the same seed and the first replay buffer training point comes around step 20. While the difference between the highest validation loss ("loss-descending", 1.04587) and lowest validation loss ("human-ranked difficulty", 1.04153) as a ratio of differences from the step-

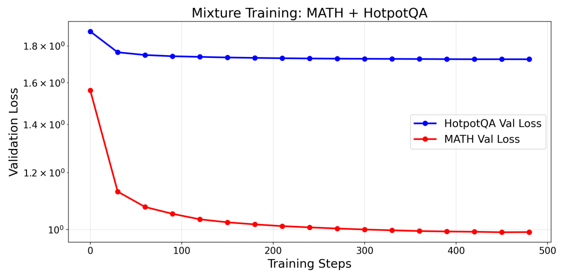

Figure 8: Result of **Mixture** Exp.

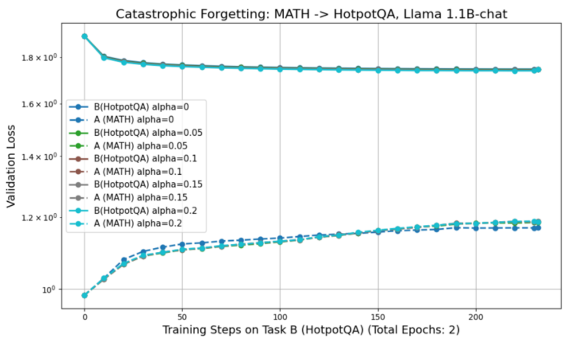

Figure 9: Result of **Loss Smoothing** Exp.

zero loss (1.01410) is (1.04587-1.01410)/(1.04587-1.01410)=1.158, so a 15.8% difference in the amount of ”forgetting”. However, the difference between human-labeled difficult than using random (1.04375) is only 7.15%, which is fairly trivial. While we weren’t able to re-run this experiment due to limited resources, we suspect that this number is even lower after accounting for noise.

5.3 Curriculum Transfer Experiment

For the baseline with Llama2-7B, the result is as in Figure 10. Just like in the baseline experiment with TinyLlama-1.1B-Chat-v1.0, the experiment figure is consistent with CF phenomenon.

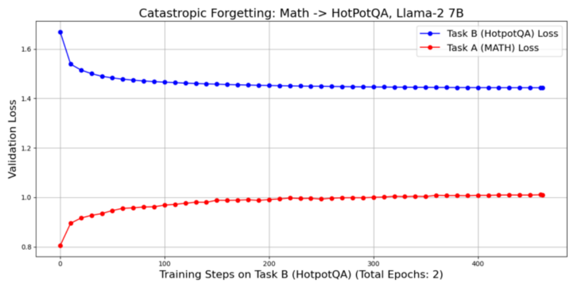

Figure 10: Result of baseline on Llama2-7B

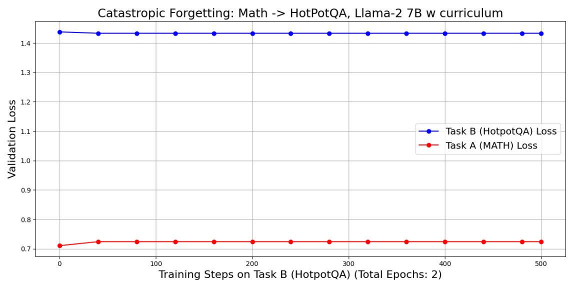

Figure 11: Result of Llama2-7B applied with Curriculum

From the experiment results on the aforementioned strategies, we concluded that the only strategy used for the most optimal curriculum is the ratio-ing method with the ratio of 20%. This is because interleaving, mixturing and ratio-ing can’t be implemented simultaneously, meanwhile loss smoothing and difficulty strategy proved to be unhelpful or did not provide enough benefit to necessitate the additional complexity. The experimental result figure is as in Figure 11, indicating improved loss stability (and thus reduced catastrophic forgetting). It also reveals an unexpected pattern: despite the 7B model being significantly larger than the 1.1B model, its loss on HotpotQA remained relatively unchanged throughout training. This suggests that the limited number of HotpotQA samples may have been insufficient for the 7B model to meaningfully benefit from the curriculum, leading to its consistently high loss. More clarification on this can be referred to F.

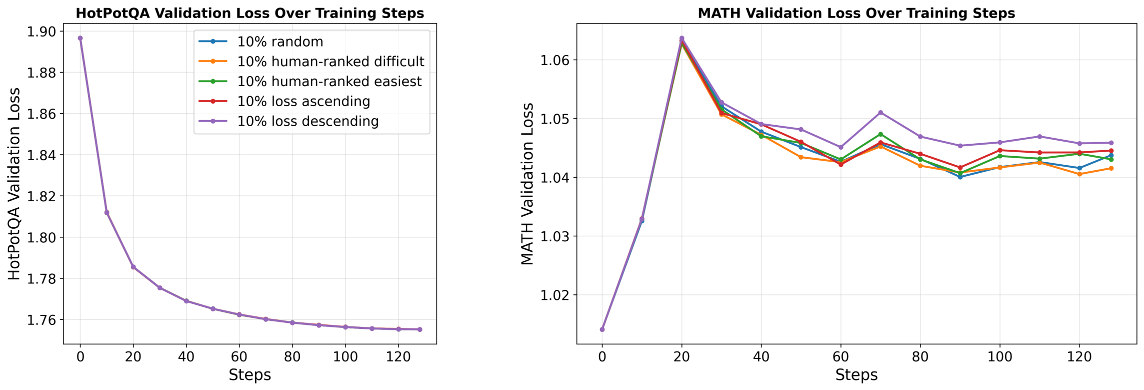

Figure 12: Results of the **Difficulty** experiment; as a reminder, ”human-ranked easiest” uses only ”level 1” examples in MATH and ”human-ranked difficult” uses only ”level 5” examples. The ”loss ascending” uses the examples with the lowest loss and ”loss descending” uses the examples with the highest loss.

6 Conclusion

So far, we have successfully reproduced the catastrophic forgetting phenomenon and verified our proposed strategies. Through the reproduction of CF, we gain some intuition about this phenomenon and accordingly have some thoughts and ideas on mitigation strategies. We elaborate on the motivation and implementation of these strategies. Some of our strategies worked, whereas some of them did not. We also conduct a preliminary investigation into curriculum transfer for LLM continual learning through a case study.

Validation of Distribution-Based Hypotheses (H1, H2): Our experimental results strongly support the hypotheses that altering the data distribution is the most effective for mitigating forgetting. Consistent with H1 (Interleaving), breaking the strict sequential nature of tasks prevents the model from overfitting to the local minima of the most recent task. Furthermore, H2 (Ratio-ing) was validated as a robust strategy, where a simple ”preview” or ”review” mechanism can help to maintain the balance between learning new knowledge and retaining old capabilities.

Refutation of Semantic and Auxiliary Hypotheses (H3, H4, H5): Conversely, our experiments largely refuted the hypotheses that relied on semantic properties or auxiliary regularization alone. H3 (Correlation) and H5 (Difficulty) posited that the semantic relationship or difficulty level of tasks would significantly influence forgetting rates; however, our results showed that neither high semantic similarity nor specific difficulty ordering provided significant gains over random baselines. Similarly, H4 (Loss Smoothing) proved ineffective in this setting, indicating that adding regularization terms to the loss function is insufficient to constrain optimization trajectories.

References

[1] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.

[2] Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[3] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning, 2019.

[4] Soheil Kolouri, Nicholas Ketz, Xinyun Zou, Jeffrey Krichmar, and Praveen Pilly. Attention-based structural-plasticity, 2019.

[5] Samuel J Bell and Neil D Lawrence. The effect of task ordering in continual learning. *arXiv preprint arXiv:2205.13323*, 2022.

[6] Jisu Kim and Juhwan Lee. Strategic data ordering: Enhancing large language model performance through curriculum learning, 2024.

[7] Vinay Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. *arXiv preprint arXiv:2007.07400*, 2020.

[8] Naimul Haque. Catastrophic forgetting in llms: A comparative analysis across language tasks, 2025.

[9] Everton L. Aleixo, Juan G. Colonna, Marco Cristo, and Everlandio Fernandes. Catastrophic forgetting in deep learning: A comprehensive taxonomy, 2023.

[10] Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning, 2024.

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

[12] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019.

[13] Jianshu Zhang, Yankai Fu, Ziheng Peng, Dongyu Yao, and Kun He. Core: Mitigating catastrophic forgetting in continual learning through cognitive replay, 2024.

[14] Arda Sarp Yenicesu, Furkan B. Mutlu, Suleyman S. Kozat, and Ozgur S. Oguz. Cuer: Corrected uniform experience replay for off-policy continuous deep reinforcement learning algorithms, 2024.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[16] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[17] Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. Lora learns less and forgets less, 2024.

[18] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[19] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[20] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

[21] 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, 2021.

[22] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.

[23] Frank F. Xu, Uri Alon, Graham Neubig, and Vincent J. Hellendoorn. A systematic evaluation of large language models of code, 2022.

[24] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, John Schulman, and Ilya Sutskever. Training verifiers to solve math word problems. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021. GSM8K dataset.

[25] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[26] Ming Shen. Rethinking data selection for supervised fine-tuning, 2024.

A Dataset Post-Processing

Since we are doing SFT(Supervised Fine-tuning) [26], we need a prompt to format data into Llama-chat-style prompt for conducting experiments. For fair comparison experiments, we adapt the following formatting functions to pre-process the dataset:

```python
def format_hotpot_qa(example):
    """Formats HotpotQA data into a Llama-chat-style prompt."""
    context = " ".join(["".join(s) for s in example["context"]["sentences"]])
    question = example["question"]
    answer = example["answer"]

    text = (
        f"<s>[INST] You are a helpful assistant. Use the following context to "
        f"answer the question. Context: {context}\n\nQuestion: {question} [/
            INST] "
        f"Answer: {answer}</s>"
    )
    return text
```

```python
def format_math(example):
    """Formats MATH data into a Llama-chat-style prompt."""
    problem = example["problem"]
    solution = example["solution"]

    text = (
        f"<s>[INST] You are a math expert. Solve the following math problem. "
        f"Show your work.\nProblem: {problem} [/INST] "
        f"Solution: {solution}</s>"
    )
    return text
```

And for the CodeParrot dataset, the prompt format is as below:

```python
def format_codeparrot(example):
    """Formats CodeParrot data into a Llama-chat-style prompt."""
    code = example["content"]

    text = (
        f"<s>[INST] You are an expert programmer. Write python code. [/INST] "
        f"{code}</s>"
    )
    return text
```

B Experiment Pseudocode

Here are the pseudocodes for the training process of Baseline, Mixture and Interleaving experiments.

Algorithm 1: Baseline

```
Input: MATH train $D_M$, Hotpot train $D_H$, validation sets $V_M$, $V_H$
Initialize LoRA model $f_\theta$;
for epoch = 1 to $E_M$ do
    for batch $B_M$ in $D_M$ do
        Loss $L_M = \text{CE}(f_\theta(B_M))$; backprop; step;
    end
end
Evaluate Phase 1 losses on $V_M$, $V_H$;
for epoch = 1 to $E_H$ do
    for batch $B_H$ in $D_H$ do
        Loss $L_H = \text{CE}(f_\theta(B_H))$; backprop; step;
        if $global\_step \bmod logging\_steps = 0$ then
            Evaluate $L_H$ (learning) on $V_H$, $L_M$ (forgetting) on $V_M$; log;
        end
    end
end
```

Algorithm 2: Mixture

```
Input: Hotpot train set $D_H$, MATH train set $D_M$, validation sets $V_H$, $V_M$
Concatenate: $D = \text{shuffle}(D_H \cup D_M)$;
Initialize LoRA model $f_\theta$;
for epoch = 1 to $E_{joint}$ do
    for batch $B$ in $D$ do
        Compute loss $L = \text{CE}(f_\theta(B))$; Backprop $L$, optimizer step;
        if $global\_step \bmod logging\_steps = 0$ then
            Evaluate $L_H$ on $V_H$, $L_M$ on $V_M$; log;
        end
    end
end
```

Algorithm 3: Interleaving

```
Input: MATH train loader $\mathcal{I}_M$, Hotpot train loader $\mathcal{I}_H$, validation loaders $V_M$, $V_H$
Initialize LoRA model $f_\theta$;
for epoch = 1 to $E_{interleave}$ do
    Reset iterators over $\mathcal{I}_M$, $\mathcal{I}_H$;
    for i = 1 to $\min(|\mathcal{I}_M|, |\mathcal{I}_H|)$ do
        Get batch $B_M$; $L_M = \text{CE}(f_\theta(B_M))$; backprop; step; global_step++;
        if $global\_step \bmod logging\_steps = 0$ then
            Evaluate losses on $V_M$, $V_H$; log;
        end
        Get batch $B_H$; $L_H = \text{CE}(f_\theta(B_H))$; backprop; step; global_step++;
        if $global\_step \bmod logging\_steps = 0$ then
            Evaluate losses on $V_M$, $V_H$; log;
        end
    end
end
```

Algorithm 4: Simplified Loss Smoothing Strategy During Sequential Fine-Tuning

```
Input: Base model $f_\theta$, smoothing coefficient $\alpha \in [0, 1)$,
training batches $\{B_t\}_{t=1}^{T}$ from the Phase 2 dataset
Output: Updated model parameters $\theta$
Initialize previous loss $\tilde{\mathcal{L}}_0 \leftarrow 0$
for $i = 1$ to $N$ do
    Sample batch $B_t = \{(x, y)\}$;
    Compute instantaneous loss:
        $\mathcal{L}_t = \text{CE}\big(f_\theta(x),\, y\big)$
    Compute smoothed loss:
        $\tilde{\mathcal{L}}_t = (1 - \alpha)\, \mathcal{L}_t + \alpha\, \tilde{\mathcal{L}}_{t-1}$
    Backpropagate using smoothed loss:
        $\nabla_\theta \tilde{\mathcal{L}}_t$
    Update model parameters:
        $\theta \leftarrow \theta - \eta\, \nabla_\theta \tilde{\mathcal{L}}_t$
    Set previous smoothed loss:
        $\tilde{\mathcal{L}}_{t-1} \leftarrow \tilde{\mathcal{L}}_t$
end
return $\theta$;
```

Algorithm 5: Loss-Based Replay Selection from *Phase 1* (MATH)

```
Input: Trained Phase 1 model $f_\theta$, tokenizer $T$, MATH train set $D_M = \{(x_i, y_i)\}_{i=1}^{N}$, number of replay
       examples $m$, selection direction $d \in \{\texttt{highest}, \texttt{lowest}\}$
Output: Replay subset $R_M \subset D_M$ with $|R_M| = m$
Initialize empty list of losses $\mathcal{L} \leftarrow [\,]$;
for $i = 1$ to $N$ do
    Construct chat-style prompt $s_i$ from $(x_i, y_i)$ (problem + solution);
    $(\text{input\_ids}_i, \text{labels}_i) \leftarrow T(s_i)$ with labels masked as $-100$ for instruction tokens;
    Compute logits $z_i \leftarrow f_\theta(\text{input\_ids}_i)$;
    Compute token-wise cross-entropy $\ell_{i,t} = \text{CE}(z_{i,t}, \text{labels}_{i,t})$ only for tokens with $\text{labels}_{i,t} \neq -100$;
    Compute per-example loss $\ell_i = \frac{1}{|\{t:\text{labels}_{i,t} \neq -100\}|} \sum_{t:\text{labels}_{i,t} \neq -100} \ell_{i,t}$;
    Append $(i, \ell_i)$ to $\mathcal{L}$;
end
if $d = \texttt{highest}$ then
    Sort $\mathcal{L}$ in descending order by $\ell_i$;
else if $d = \texttt{lowest}$ then
    Sort $\mathcal{L}$ in ascending order by $\ell_i$;
Let $S = \{i_1, \ldots, i_m\}$ be the indices of the first $m$ entries in the sorted list $\mathcal{L}$;
Construct replay set $R_M = \{(x_i, y_i) \in D_M : i \in S\}$;
return $R_M$;
```

C Dataset

The MATH dataset [20] is a comprehensive benchmark designed to evaluate the mathematical reasoning capabilities of models. MATH consists of 12,500 challenging competition-level mathematics problems.

These problems are sourced from many mathematics competitions, including AMC 10, AMC 12, and AIME, etc. This dataset is considered to represent a relatively high standard of complexity that requires model to obtain strong reasoning capabilities. The task is to generate the correct final answer given a problem statement. Performance is typically evaluated based on the accuracy of the final result symbolic verification.

HotpotQA [22] is a question-answering dataset constructed in multi-hop reasoning field. It consists of 113,000 question-answer pairs collected from English Wikipedia. HotpotQA requires systems to reason across multiple supporting documents to infer the correct answer. HotpotQA also support distinguishing difficulty levels through its evaluation settings.

CodeParrot [23] is a code corpus designed to train and evaluate L:anguage Models for code generation tasks. It comprises a massive collection of Python code files scraped from public GitHub repositories. CodeParrot focuses exclusively on Python, serving as a standard benchmark for evaluating a model's proficiency in Python syntax, library usage, and logical implementation. Models trained on CodeParrot are tasked with predicting the next token in a sequence of code.

GSM8K [24] is a dataset consisting of approximately 8.5K high-quality grade school math word problems created by human writers. Unlike simple calculation tasks, these problems require models to perform multi-step reasoning to derive the correct solution. It serves as a standard benchmark for evaluating the reasoning capabilities of Large Language Models.

D Experiment Settings

For the toy expriment, we use DistilBERT-base-uncased as the base model and do full fine-tuning on AG News dataset. We tokenize the offline AG News corpus with the DistilBERT tokenizer truncate to the fixed-length of 128. We adopt AdamW with a learning rate of 2e-5 and a linear scheduler with no warmup and schedulers are parameterized by the number of batches in the current phase. Batch size is fixed at 16.

For the formal Strategy Exploration experiments, overall, we adhered to a unified experimental setup (e.g., hyperparameter selection) as strictly as possible. Generally, we choose TinyLlama-1.1B-Chat-v1.0 and Llama2-7B as language model. In LoRA, we set rank to 8 and scaling coefficient $alpha$ to 16. The LoRA adaptation is specifically applied to the query, key, value, and output projection matrices within each self-attention layer, while all other model parameters remain frozen. We set the max sequence length to 2048, and hence around 10% of data will be filtered. We use 4000 instances for training and 400 instances for evaluation, which is empirically reasonable and sufficient to finetune on a 1.1B and 7B model. Batch size was set to 64 and all experiments were run on a single A100 80B GPU. For loss definition, we adapted to loss masking, i.e., tokens before first occurrence of the closing bracket ']' are set to -100 in labels to avoid penalizing instruction encoding. In Interleaving experiment, since *phase* concept is not adopted, so to ensure the same total training amount, the epoch is set to 4.

The choice of TinyLlama-1.1B and Llama-2-7B is motivated by three key factors. First, both models share the same foundational Llama architecture. Second, given the extensive sub-experiments, TinyLlama-1.1B can be computationally efficient. Lastly, Llama-2-7B represents a widely adopted standard in the open-source community.

For the LoRA configuration, we follows the empirical recommendations from the original LoRA paper [16]. We utilized the AdamW optimizer with a learning rate of 2e-5 and a linear decay scheduler. This conservative learning rate was selected to mitigate the risk of 'catastrophic interference' during fine-tuning; a higher learning rate could aggressively alter the pre-trained weights, thereby exacerbating the forgetting of previous knowledge. In general, given the extensive sub-experiments required for our curriculum strategy exploration, we prioritized a fixed, robust set of hyperparameters over an exhaustive grid search.

E More Experiment Results

The following figures 13 and 14 are the results of Baseline and Interleaving. These two experiments are run three times so that error bands can be plotted. Due to compute resource limit, these two experiments are the only two which are run three times.

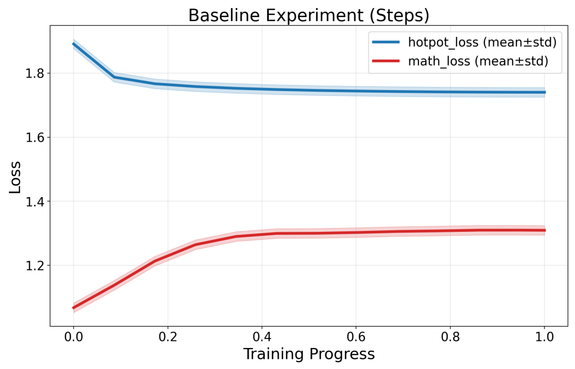

Figure 13: Result of Baseline

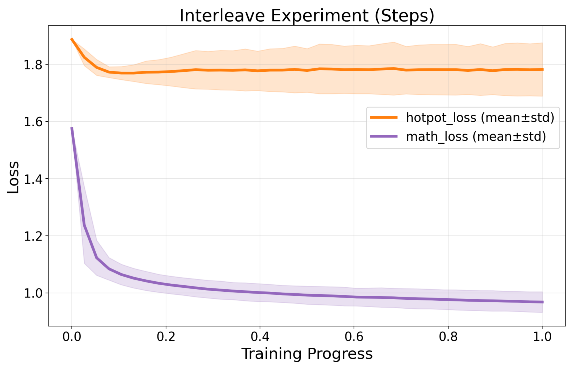

Figure 14: Result of **Interleaving** Strategy Exp.

F Limitation

First, the scope of our experiments is not exhaustive. Due to time and resource constraints, we could not achieve comprehensive coverage in terms of strategy validation or comparative baseline design. Second, given the timeline and the large volume of experiments, we were unable to conduct repeated trials with multiple random seeds for all the sub-experiments, precluding the inclusion of error bars for a more rigorous statistical assessment. Finally, for the experiment of curriculum transfer, it should be noted that the sample size settings used in this study were not tuned, which resulted in accelerated convergence rates and artificially smoothed trajectories for the curriculum strategy. While we recognize this as an experimental oversight, strict time constraints precluded us from re-conducting the experiments to rectify this issue.

G AI tool narrative

During the development of this project, we utilized Gemini 3 Pro for partial writing polishing and code debugging assistance. All AI-generated suggestions were rigorously verified by the authors.

H Incorporation with Feedback

We would like to express our sincere gratitude to professor's suggestion on the poster session and peer reviewers' valuable advice. We make the following specific refinement to address reviewers' concerns and issue-shooting. In D, we supplement the implementation detail on LoRA and the reason for hyperparameter and language model choice. In E, we try our best to run two sub-experiements for multiple time so as to plot figures with error band. In Figure 5, we combine three sub-experiments together in one single plot for space saving, and put their separate figures in Appendix E. We complement the AI tool narrative in Appendix G. In Conclusion section, we elaborate more on the effectiveness and validity of our hypothesis. We have corrected minor notational error throughout the paper writing.