sDsCL: A Score Distribution Enhanced Continual Learning Method for Large Language Model

Anonymous ACL submission

Abstract

Catastrophic forgetting poses a significant challenge in the fine-tuning of Large Language Models (LLMs), particularly when adapting them to specific domains that necessitate different knowledge from the inherent knowledge of the LLMs. Efforts have been directed towards continual learning to mitigate this forgetting issue. Among the various strategies, replaying past experiences has proven to be a simple yet effective solution. However, most replay-based methods employ coarse-grained sampling, which only considers data similarity and overlooks changes in task distribution during the fine-tuning process. In this paper, we introduce a score distribution enhanced continual learning method (sDsCL) that incorporates both instruction- and instance-level sampling to capture more representative data for efficient continual learning. Specifically, we assign a score to each old sample based on its performance in previous tasks and align the replay set sampling with this score distribution. Moreover, to address the LLM's focus on potential revisions, we propose a data synthesis technique to augment the data that scored very low, indicating severe failures. This technique aims to correct previously incorrect responses. To evaluate the effectiveness and efficiency of our proposed method, we conducted extensive experiments across a series of tasks. The comprehensive experimental results presented and analyzed in this paper provide strong evidence of the proposed method's efficacy.

1 Introduction

Large language models (LLMs) trained on large-scale corpus data demonstrate superior performance for a wide range of downstream Natural Language Processing (NLP) tasks (Achiam et al., 2023; Touvron et al., 2023; Mann et al., 2020). At the same time, the large language model can also be fine-tuned to adapt to new task patterns through instruction fine-tuning (Peng et al., 2023; Wei et al., 2021). In practical applications, we typically aspire for large language models to simultaneously accommodate a variety of instruction-based tasks, with the flexibility to evolve in response to ongoing demands (Jin et al., 2021; Daruna et al., 2021). However, a major challenge that hinders the effectiveness of LLMs is catastrophic forgetting. This means that LLMs tend to forget previously held information when acquiring new data (Kirkpatrick et al., 2017; Li et al., 2022). Thus Continual Learning (CL) has been proposed to cope with the continuous stream of tasks and to avoid catastrophic forgetting during training (Goodfellow et al., 2013; Kemker et al., 2018).

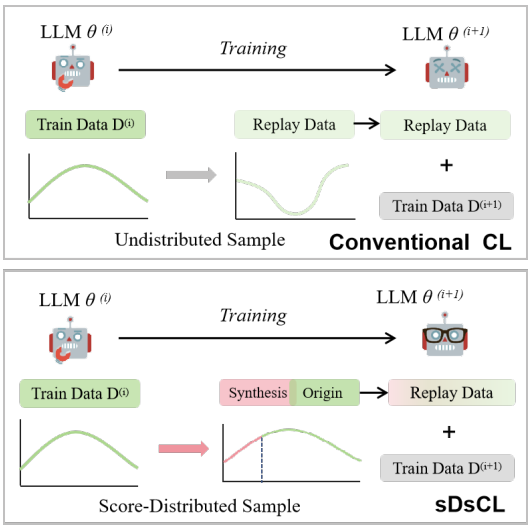

Figure 1: The framework of sDsCL. Unlike conventional Continual Learning methods, we sample replay data from a score distribution perspective while using a portion of synthesis data.

Given the substantial computational costs of training LLMs, the direct application of conventional structure-based or regularization-based continual learning methods is challenging. Replay-based methods are the most resource-friendly and effective approach among all CL methods. Pre-

vious work has followed the traditional approach in continual learning to combat catastrophic forgetting during training by retaining the most representative instances of each task (Zhang et al., 2023, 2022b). Scialom et al. (2022) propose the Continual-T0 to build a data buffer to retain a portion of past data and continuously add data from new tasks in subsequent training. DynaInst (Mok et al., 2023) combines the ideas of regularization and replay to maintain performance by retaining the most representative tasks with corresponding data during training process. However, these tasks are fundamentally grounded in Pre-trained Language Models (PLMs) (Devlin, 2018; Lewis et al., 2019), primarily assessing the model's capacity for continuous learning on conventional natural language processing tasks, such as fundamental text classification.

More and more efforts have been dedicated to applying LLMs to NLP tasks (Sun et al., 2019; Yin et al., 2022). Meanwhile, replay-based CL methods show their potential to relieve forgetting for LLM-based methods. For example, InsCL (Wang et al., 2024) takes into account the instruction dimension, selectively retains a subset of data for each subsequent task, and adaptively modulates the replay data size in accordance with the similarity between instructions. SSR (Huang et al., 2024) uses self-synthesized data to construct the replay set, taking into account that previous task data may not be available in practice. However, these replay-based methods rely on coarse-grained sampling techniques that focus on data similarity, neglecting shifts in task distribution that occur during the fine-tuning process.

Inspired by the above problem, we propose an innovative Score Distribution Enhanced Continual Learning (sDsCL) method to achieve efficient continual learning by combining instruction- and instance-level distribution sampling to obtain more representative data. As shown in Fig. 1, we initiate by assigning a score to each data within the current task dataset, evaluating its performance during the ongoing training phase. This assessment then informs the sampling process for the subsequent replay dataset, guiding it in a targeted and effective manner. Prior work has shown that performance using less but higher quality data can be comparable to complete data (Toneva et al., 2018; Abbas et al., 2023; Tirumala et al., 2024). Taking into account that data with exceedingly low scores could potentially diminish the overall training efficacy, we introduce a data synthesis technique designed to augment the representation of such low-scored data. Leveraging the context generation capabilities of Large Language Models to synthesize similar data aims to mitigate the adverse impact that challenging cases may exert on the training process. Experimental results on a range of tasks show that sDsCL is effective in mitigating catastrophic forgetting during training.

The main contributions of this work are summarized as follows:

- We introduce a novel instance-level score-distribution-based replay data sampling mechanism to mitigate catastrophic forgetting in LLMs.
- Our method takes into account the diversity of all instructions in previous dataset, aligning the replay data distribution as closely as possible to the original data distribution.
- We utilize data synthesis techniques to correct previously incorrect answers, providing potential for performance improvement.
- We validate our method through extensive experiments on eight task streams, demonstrating significant improvements over existing approaches.

2 Related Work

2.1 Conventional Continual Learning

Continuous learning mainly concern with involving algorithms that enable the model to continuously accumulate knowledge based on a constant stream of data, which can be categories, domains, tasks, etc. The current mainstream continual learning methods can be briefly classified into three categories: Regularization-based methods, Structure-based methods, and Replay-based methods. Regularization-based methods (Razdaibiedina et al., 2023; Zhang et al., 2022a; Huang et al., 2021; Cha et al., 2020; Kirkpatrick et al., 2017) generally ensure that some important parameters in the model are not changed by imposing additional constraints on the loss function. Structure-based methods (Madotto et al., 2020; Gu et al., 2020; Rusu et al., 2016), on the other hand, add extra layers or blocks to the original model to learn new knowledge or tasks, and acquire the ability to cope with new tasks by expanding the model, but this method generally leads to an exponential increase in model

parameters. Replay based approaches (de Masson D'Autume et al., 2019; Rolnick et al., 2019; Ke et al., 2021) address the forgetting problem by retaining a portion of each previously trained dataset to be added to the subsequent training process.

2.2 Continual Learning for LLMs

Due to their size and complexity, Large Language Models require a multifaceted approach to continual learning. Llama Pro (Wu et al., 2024) effectively improves the performance on new data and tasks by expanding the transformer block in the Llama structure. O-Lora (Wang et al., 2023) is based on the LoRA (Hu et al., 2021) technique, which sets a corresponding subspace for each task, and optimizes the LoRA blocks of different tasks orthogonally so that different tasks would not affect each other. Bound by the substantial parameters of LLMs, these tasks either demand increased computational resources or face challenges in scaling the number of tasks.

Considering the exorbitant computational resources required for Large Language Models, replay-based methods are the most suitable for large model scenarios. Continual-T0 (Wei et al., 2021) has illustrated that random sample replay can mitigate catastrophic forgetting, achieved by preserving a data buffer throughout the training process. InsCL (Wang et al., 2024) designed a method for dynamically selecting the data to be reenacted at each stage based on the distance of the instruction features in the instruction fine-tuning data. SSR (Huang et al., 2024) took into account the dataset unavailability problem by replacing training set data with synthetic data for data replay. However, these studies uniformly adopt a coarse-grained perspective when assembling replay sets. Our approach diverges from this by transitioning to finer-grained sampling techniques, crafting replay sets with precision at both the instruction level and the instance level.

3 Method

3.1 Task Formulation

The purpose of Continual Learning is to continuously learn a series of new tasks while maintaining the original performance of the model as much as possible. Specifically, for a series of tasks $T = \{T_1, \ldots, T_n\}$ and the corresponding datasets $D = \{D_1, \ldots, D_n\}$, the model switches to the next task after a few rounds of training in task T_i each time. The replay-based continual learning method, on the other hand, keeps a portion of the data subset d_i as the replay data in each stage t. Formally, when training at task t, the dataset with added replay data can be defined as:

$$\hat{D}_t = D_t \bigcup \sum_{i=1}^{t-1} d_i, \tag{1}$$

where $\hat{D}_t$ represents the dataset used for the training of the t-th task and d_i is the corresponding replay sub-dataset for each task.

3.2 Instruction Distribution-Based Sampling

Many replay-based continuous learning (CL) methods prioritize the dissimilarity of instructions across different tasks (Gogoulou et al., 2023; Xu et al., 2023; Mi et al., 2020). They aim to maintain the model's performance on the previous task by retaining a subset of data that is most distinct from the new task. However, relying solely on text embedding similarity for replay data selection can lead to a shift in the original data distribution, resulting in unpredictable outcomes. In extreme cases, retaining only a subset of instructions for replay can cause the model to overfit to specific instruction types, thereby affecting the overall task performance.

Intuitively, the instruction distribution in the replay dataset should mirror that of the original dataset as closely as possible to increase the likelihood of reproducing similar performance results. The formula for sampling based on instruction distribution is as follows:

$$R_I = \sum_{i=0}^{N} (K \times \mathcal{D}_I^i); \mathcal{D}_I = \{\frac{I_j}{\sum_{i=0}^{N} I_i}\}_j^N, \tag{2}$$

where K is the number of sampled replay data, R_I denotes the replayed dataset, I and $\mathcal{D}_I$ denote the instruction set and the corresponding distribution, respectively. N is the total number of instructions in the current task dataset.

In this formula, the distribution of instructions in the replay data is constituted by the proportion of each type of instruction in the original dataset.

3.3 Score Distribution-Based Sampling

After sampling instructions based on instruction distribution, we propose a score distribution-based instance-level sampling method within the same

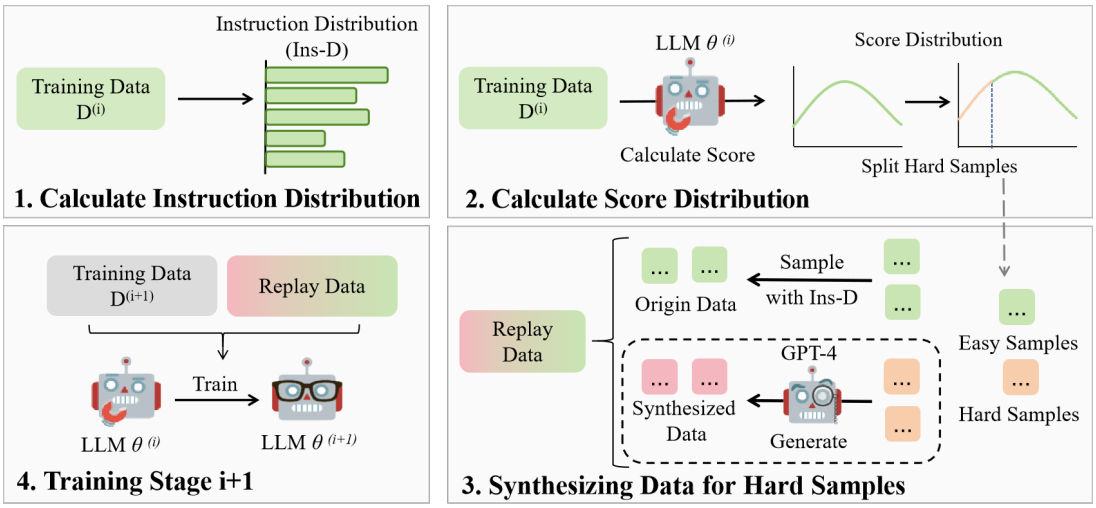

Figure 2: Our sDsCL framework. In order to mitigate catastrophic forgetting from obtaining more representative replay data. We first evaluate the score distribution as well as the instruction distribution of the current task training set, and sample the replay data based on this distribution $\mathcal{D}$. Then, to compensate for the effect of data with low scores, we use the in-context learning capability of LLM to generate similar instances and weightedly fuse them with the original dataset based on the overall distribution to obtain a replay dataset for the next training stage.

instruction. For each instruction, we calculate the score distribution of all instances from the training dataset and sample instances based on this distribution. This score distribution reflects the model's learning outcome from the previous task, with lower scores indicating higher difficulty and higher scores indicating lower difficulty for the model. The formula for the instance-level sampling method based on score distribution is as follows:

$$R_S = \sum_{i=0}^{N} \sum_{j=0}^{M} (K \times \mathcal{D}_I^i \times \mathcal{D}_S^j);$$
$$\mathcal{D}_S = \{\frac{S_j}{\sum_{i=0}^{M} S_i}\}_j^M, \tag{3}$$

where K is the number of sampled replay data, R_S denotes the replayed dataset, S and $\mathcal{D}_S$ denote the score set and the corresponding distribution, respectively. M is the number of score distribution intervals.

In this formula, the composition of the replay data is based on the instruction composition from the previous section. Each instruction's instances are selected based on the score distribution, which is divided into ten parts, and each part is sampled proportionally.

3.4 Synthesizing Data for Hard Samples

Prior research has demonstrated that utilizing synthesized data as replay data can mitigate the models' forgetting issues (Smith et al., 2021; Yin et al., 2020; Van de Ven et al., 2020; Shin et al., 2017). In this work, we propose to replace underperforming data with synthesized data in the replay dataset. Given the model's high sensitivity to data quality, we opt to use GPT-4 (Achiam et al., 2023) for data synthesis instead of Llama-7B.

Specifically, we first identify a subset of data with the lowest scores based on the score distribution and use these as seeds for GPT-4 to generate new, similar questions with equivalent difficulty levels. We then substitute this subset of hard samples with the newly generated data. Experimental results indicate that replacing 25% of the samples with synthesized data yields the highest performance.

3.5 Score Distribution Enhanced Continual Learning Framework

We introduce the Score Distribution Enhanced Continual Learning (sDsCL) paradigm, a novel continual learning framework that emphasizes the preservation of the original dataset's distribution. sDsCL incorporates two key distribution considerations: instruction distribution and score distribution. By sampling the replay dataset based on these distributions, we aim to ensure that the sub-dataset's data distribution aligns as closely as possible with the original dataset.

Recognizing that hard instances may pose learning challenges, sDsCL extends the training process by generating additional data of similar difficulty.

Algorithm 1 sDsCL sampling process at stage t

Input: Trainset D_t, Generated Set D_g, Stage model θ_t,
Replay number K, Synthesis Data Ratio β
Output: Replay-set R_t

```
Obtain Instruction Set and Score Set I_t, S_t from D_t;
Get Distributions D_I^t, D_S^t with model θ_t;
for each I_t^i in I_t do
    k ← K × D_i^t;
    for d_t^j in D_S[β : 1] do
        R_t^d ← sample k × (1 − β) data from d_t^i;
    end for
    R_t^g ← sample k × β data from D_g;
end for
R_t ← R_t^d ∪ R_t^g;
return R_t
```

As detailed in Algorithm 1, for the t-th task in a given task sequence T, we initially train the stage model θ_t on the corresponding dataset D_t. Post-training, we employ the θ_t model to evaluate the current task dataset's difficulty based on Rouge-L scores, thereby obtaining the dataset score distribution $\mathcal{D}_S$. The instruction distribution $\mathcal{D}_I$ is an inherent attribute of each task dataset. We use both $\mathcal{D}_I$ and $\mathcal{D}_S$ to determine the K samples for replay.

Simultaneously, we select a proportion of the lowest scores for each instruction as seeds for GPT-4 to generate new data of similar difficulty. Next, we sample from the generated dataset D_g and use these samples to replace the lowest-scoring β portion of the original K replay data. This comprehensive approach ensures a balanced replay dataset that mirrors the original data distribution while addressing the challenge of hard instances.

4 Experiment

4.1 Datasets

Following previous works (Wang et al., 2024), we conduct our experiments on the SuperNI dataset (Wang et al., 2022), a comprehensive large-scale benchmark dataset for instruction tuning. To simulate a sequence of tasks for continual learning, we manually construct eight tasks from the SuperNI dataset, each with ten different instructions. Each instruction has 1,000 instances for training and 200 instances for evaluation. Specific details of the dataset are provided in the Appendix.

4.2 Implementation

We conduct our experiments using Llama-7B (Wu et al., 2024). We use the Adam optimizer with a learning rate of 1e-5 during training and a global batch size of 64. Each task is trained for one epoch. Additionally, we set the maximum input length to 2048 tokens and retain the checkpoints from the final stage of training for evaluation.

During the synthetic data generation phase, we sampled 10 underperforming instances per instruction from the data, specifically those with the lowest 25% scores, resulting in a total of 100 examples per task. Using pairs of these examples, we prompted GPT-4 to generate similar instances, repeating this process three times to produce 2,700 instances per task. sDsCL then randomly selects an equivalent amount of data with the lowest 25% scores from these generated instances to replace the original data for replay. Consistent with previous works, we set the replay set size to 200 in all experiments.

4.3 Baselines

We leverage Llama-7B as the base model and compare our sDsCL approach with the following strategies:

No-Replay: Incrementally trains Large Language Models (LLMs) without using replay data.

Random-Replay: Randomly samples α instances from each previous task for replay.

InsCL: (Wang et al., 2024) This method dynamically replays previous data based on task similarity and instruction complexity. Parameters used in our experiments remain the same as in the original work.

SSR: (Huang et al., 2024) It generates and refines synthetic instances for rehearsal. For a fair comparison, we use Llama-7B as the base model, while keeping the rest of the experimental setup consistent with SSR.

4.4 Metrics

To effectively evaluate the performance of generative models on open language tasks, we use Rouge-L (Lin, 2004) as the primary evaluation metric. This metric is widely recognized for reflecting the quality of model outputs (Zhang et al., 2022a). Following previous work (Scialom et al., 2022; Jang et al., 2023), we employ the following evaluation metrics base on Rouge-L. e_i^t denotes the Rouge-L performance on task t at training stage i:

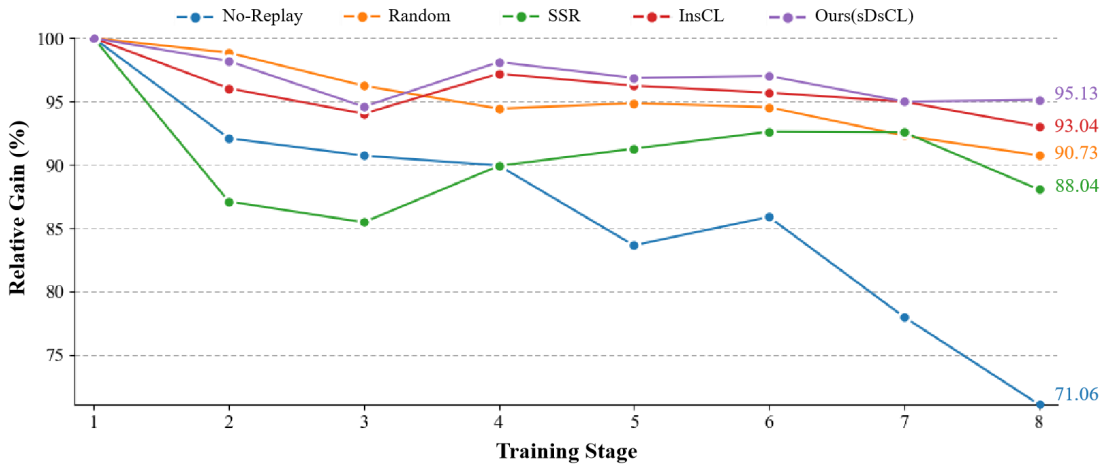

Figure 3: Progressive Relative Gain results for Llama-7B in continual instruction tuning. We set the Relative Gain to 100 for the first task, representing the initial performance without forgetting. For stage t, we plot the Average Relative Gain across all tasks. The closer the Relative Gain is to 100, the better the method is at alleviating catastrophic forgetting and preserving knowledge.

Method	Curriculum			Reverse			Random		
	ARG	BWT	AR	ARG	BWT	AR	ARG	BWT	AR
No-replay	80.63	-11.69	50.85	74.92	-8.95	54.15	70.93	-4.18	54.9
Random-replay	92.64	-3.06	63.95	90.01	-2.24	64.57	91.42	-1.43	65.05
SSR (Huang et al., 2024)	89.70	-3.50	61.77	84.79	-2.55	61.80	86.58	-3.51	61.02
InsCL (Wang et al., 2024)	94.54	-2.19	64.69	91.34	-1.55	65.55	92.24	-1.23	65.37
Ours (sDsCL)	**95.89**	**-1.28**	**65.52**	**93.74**	**-1.35**	**66.33**	**94.25**	**-0.70**	**66.86**

Table 1: Results for different training orders. We evaluated the tasks using Curriculum, Reverse, and Random training orders. A promising method is expected to yield a high metric score, indicating good performance and stability. The best results are highlighted in bold, while the second-best results are underlined.

Average Rouge-L (AR). This metric evaluates the average performance of the model across all tasks after T rounds of training.

$$\mathbf{AR} = \frac{1}{T}\sum_{i=1}^{T} e_i^{(T)}. \tag{4}$$

Backward Transfer (BWT). This metric assesses the impact of learning the current task on the performance of previous tasks. For the current training phase t, it compares the difference in metrics between the current phase and the original training phase for a given task i:

$$\mathbf{BWT} = \frac{1}{t-1}\sum_{i=1}^{t-1}(e_t^{(t)} - e_t^{(i)}). \tag{5}$$

The higher the BWT score the better, indicating that the LLM has not forgotten previously acquired knowledge.

Average Relative Gain (ARG). This metric focuses on the issue of forgetting by comparing performance on a single task to its performance in a continual learning setting. Following the approach of (Wang et al., 2024), the Relative Gain (RG) in training phase t is defined as:

$$\mathbf{RG}_t = \frac{1}{t-1}\sum_{i=1}^{t-1}\frac{R_t^i}{R_{std}} \times 100\%, \tag{6}$$

Additionally, we use the average of all stages of RG to indicate the efficiency of continual learning:

$$\mathbf{ARG} = \frac{1}{T}\sum_{t=1}^{T} RG_t. \tag{7}$$

4.5 Main Result

We employed three distinct training orders for the eight tasks: random, curriculum, and reverse curriculum. As demonstrated in Table 2, our method consistently achieved the highest average relative

gain, backward transfer, and average Rouge-L score across all three training orders. Specifically, for the curriculum training order, our sDsCL achieved an average relative gain of 95.89 and an average Rouge-L score of 65.52. These robust results indicate that our proposed sDsCL method can effectively mitigate the forgetting phenomenon while maintaining an impressive average performance after all tasks have been trained.

Moreover, all methods performed better when utilizing the curriculum training order, which progressively increased task difficulty. This aligns with the principles of curriculum learning (Wang et al., 2021; Bengio et al., 2009), where rationalizing the training order can enhance the stability and learning outcomes of each training phase.

Figure 3 illustrates the progressive relative gain over eight training stages using the curriculum training order. Our method achieved a 15.26 higher score compared to the no replay method, and a relative gain improvement of 3.25 compared to the random replay method. This underscores the importance of selecting a portion of data for replay and the careful design of such a selection schema.

Furthermore, sDsCL outperformed InsCL by achieving a 1.36 higher relative gain, by focusing on instance-level instructions rather than solely on instruction level. This improvement may also be attributed to sDsCL's consideration of the diversity of instructions in the replay data, while InsCL only considers the most dissimilar previous instructions, potentially overlooking others. Interestingly, SSR performed worse than random replay. This could be due to SSR's original use of Llama2 to generate synthetic data, while we used Llama in our experiment. This highlights the potential pitfalls of relying solely on a generative approach, where low-quality synthetic data can exacerbate the forgetting phenomenon.

In contrast, sDsCL employs a score-distribution-based replay method that selects replay data at a granular instance level and judiciously uses only a portion of synthesized data targeting previously underperformed instances. This approach maximizes the retention of accuracy for each previous task, demonstrating the effectiveness of our method.

4.6 Analysis

To further investigate the role of each component in sDsCL, we used random-replay as a simple baseline and added each core component of our method separately: using 25% synthetic data and 75% random data, sampling instructions based on the instruction distribution, and sampling based on the task's score distribution. As shown in Table 2, each component individually boosts the ARG, BWT, and AR metrics of random-replay by an average of 1.69, 0.67, and 0.82, respectively. When all these components are combined in sDsCL, the improvements are even more significant, with increases of 3.25, 1.78, and 1.57 in ARG, BWT, and AR, respectively. This demonstrates the importance of these three factors. In the following sections, we will conduct a deeper analysis of each component.

Method	Curriculum		
	ARG	BWT	AR
No-Replay	80.63	-11.69	50.85
Random-Replay	92.64	-3.06	63.95
+ Gen Only	93.99	-2.65	63.88
+ InsD Only	94.34	-2.31	65.32
+ ScoreD Only	94.67	-2.21	65.13
Ours (sDsCL)	**95.89**	**-1.28**	**65.52**

Table 2: Effect of each component of sDsCL. We report the results of using synthetic data (Gen Only), sampling based on the instruction distribution (InsD Only), and sampling based on the score distribution (ScoreD Only). The best results are in bold.

4.6.1 Effect of Instruction Distribution-Based Sampling

Previous research has often focused on selecting replay data that is most dissimilar to the new task, encompassing only certain types of instructions (Yin et al., 2023; Wang et al., 2024). In contrast, our experiments have investigated the use of different proportions (10%, 50%, and 100%) of instruction types. As illustrated in Figure 4 (a), sampling instructions based on their original distribution, which includes all types of instructions, significantly surpasses the performance of partial instruction sampling methods. This underscores that the model's performance is heavily dependent on the diversity of the instruction dataset during fine-tuning. Therefore, it is essential to incorporate all types of instructions during replay, as the omission of any instruction type can potentially compromise the model's performance.

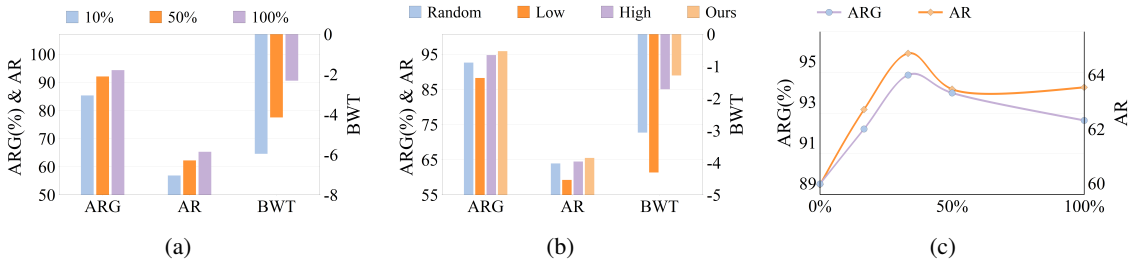

Figure 4: Analysis of the different factors in sampling based on instruction distribution, score distribution, and using synthetic data. (a) Effect of sampling different portions of instructions as replay data. (b) Effect of different sampling schemes from the score distribution: randomly sampled, sampling only from high scores (top 20%), sampling only from low scores (lowest 20%), and our method. (c) Effect of using different ratios of synthetic data.

4.6.2 Effect of Score Distribution-Based Sampling

To delve deeper into the significance of sampling in line with the model's score distribution, we conducted experiments using random sampling (without considering score distribution), and three score-based sampling methods: sampling exclusively from the top 20% highest scoring data, the bottom 20% lowest scoring data, and sampling proportionally according to the score distribution (i.e., the top 20% highest and bottom 20% lowest scoring data each make up 20% of the replay data). As depicted in Figure 4 (b), utilizing the model's prior performance as a guide for replay data selection markedly outperforms random replay. However, solely selecting low scoring data underperforms compared to random replay. As previously discussed, retraining the model with data it previously struggled with can exacerbate its performance. Furthermore, it is more beneficial to include all data and select according to the score distribution, rather than merely focusing on the data where the model performed well.

4.6.3 Effect of Synthesis Data Ratio

Incorporating synthetic data into model training is a prevalent approach, but over-reliance on such data can potentially degrade model performance (Shumailov et al., 2024; Gibney, 2024; Wenger, 2024). Our experiments, as shown in Figure 4 (c), explored the impact of replacing varying proportions (0%, 10%, 25%, 50%, and 100%) of the lowest-performing data with synthetic data.

The results indicate that a model trained solely on synthetic data underperforms compared to one trained without any synthetic data, highlighting the potential pitfalls of over-reliance on synthetic data. However, when synthetic data is used to supplement a portion of the training data, specifically 10%, 25%, and 50%, we observe an improvement in model performance across all metrics.

This suggests that for data where the model initially underperforms, introducing new, similar synthetic data can be more beneficial than simply retraining on the same data. However, the benefits are not linearly proportional to the amount of synthetic data used. Using a small fraction (10%) of synthetic data doesn't yield significant improvements, possibly due to the persistence of underperforming data in the training set. Conversely, a larger fraction (50%) might introduce noise and negatively impact the data that the model was previously handling well.

Our experiments indicate that replacing 25% of the underperforming data with synthetic data strikes the optimal balance, leading to the highest performance across all metrics.

5 Conclusion

In this work we introduce a novel framework, Score Distribution Enhanced Continual Learning (sDsCL), which mitigates catastrophic forgetting by intelligently constructing replayed data to mirror the original dataset's distribution. sDsCL selects replay data based on both the original instruction distribution and the model's weaknesses, ensuring it is representative of the dataset and addresses areas where the model struggled. Additionally, we boost model accuracy by generating data similar to underperforming examples with LLMs, helping the model encounter and leran from challenging cases. Our extensive experiments on a dataset comprising eight diverse tasks demonstrate the effectiveness of sDsCL in mitigating catastrophic forgetting.

6 Limitations

sDsCL shows good performance with a uniform number of instances of each instruction which is idealized. However, we believe that the standard distribution to some extent also proves the importance of the distribution factor in the process of picking data. In addition, for datasets with a large amount of data, evaluating the whole data is also computationally expensive, and designing a lightweight method for evaluating the data is our subsequent endeavor. Meanwhile, we only simply sampled the generated data for replacing the low-scoring original data, and there should be more potential for synthetic data to be explored in continuous learning.

References

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flavio P Calmon, and Taesup Moon. 2020. Cpr: classifier-projection regularization for continual learning. *arXiv preprint arXiv:2006.07326*.

Angel Daruna, Mehul Gupta, Mohan Sridharan, and Sonia Chernova. 2021. Continual learning of knowledge graph embeddings. *IEEE Robotics and Automation Letters*, 6(2):1128–1135.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elizabeth Gibney. 2024. Ai models fed ai-generated data quickly spew nonsense. *Nature*.

Evangelia Gogoulou, Timothée Lesort, Magnus Boman, and Joakim Nivre. 2023. A study of continual learning under language shift. *arXiv preprint arXiv:2311.01200*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2020. On the transformer growth for progressive bert training. *arXiv preprint arXiv:2010.12562*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. *arXiv preprint arXiv:2403.01244*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. *arXiv preprint arXiv:2104.05489*.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023. Exploring the benefits of training expert language models over instruction tuning. In *International Conference on Machine Learning*, pages 14702–14729. PMLR.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2021. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22443–22456.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. 2022. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5454.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2020. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1.

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020. Continual learning for natural language generation in task-oriented dialog systems. *arXiv preprint arXiv:2010.00910*.

Jisoo Mok, Jaeyoung Do, Sungjin Lee, Tara Taghavi, Seunghak Yu, and Sungroh Yoon. 2023. Large-scale lifelong learning of in-context instructions and how to tackle it. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12573–12589.

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

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in neural information processing systems*, 32.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. *arXiv preprint arXiv:2205.12393*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631:755 – 759.

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2021. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9374–9384.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. 2024. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024. Inscl: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. *arXiv preprint arXiv:2403.11435*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022.

Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

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

Emily Wenger. 2024. Ai produces gibberish when trained on too much ai-generated data. *Nature*, 631 8022:742–743.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. 2024. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*.

Zihao Xu, Xuan Tang, Yufei Shi, Jianfeng Zhang, Jian Yang, Mingsong Chen, and Xian Wei. 2023. Continual learning via manifold expansion replay. In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 733–738. IEEE.

Da Yin, Xiao Liu, Fan Yin, Ming Zhong, Hritik Bansal, Jiawei Han, and Kai-Wei Chang. 2023. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. *arXiv preprint arXiv:2305.14327*.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8715–8724.

Wenpeng Yin, Jia Li, and Caiming Xiong. 2022. Contintin: Continual learning from task instructions. *arXiv preprint arXiv:2203.08512*.

Han Zhang, Sheng Zhang, Yang Xiang, Bin Liang, Jinsong Su, Zhongjian Miao, Hui Wang, and Ruifeng Xu. 2022a. Clle: A benchmark for continual language learning evaluation in multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 428–443.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022b. Continual sequence generation with adaptive compositional modules. *arXiv preprint arXiv:2203.10652*.

Zihan Zhang, Meng Fang, Ling Chen, and Mohammad-Reza Namazi-Rad. 2023. Citb: A benchmark for continual instruction tuning. *arXiv preprint arXiv:2310.14510*.

A Dataset

A.1 Data Synthesis

In the data synthesis phase, we employ GPT-4 as the generator. We utilize the in-context learning capability of the LLMs, using straightforward instructions to guide GPT-4 in producing synthetic data that aligns with the given instrucitons and maintains a similar level of difficulty with the demonstrations.

Generate a similar question pair of similar difficulty based on the instructions and examples. Outputs only one pair of questions, strictly adhering to the input-output format, without outputting any additional information. {Instruction.} {Examples 1&2.}

Table 3: Designed Synthesis Prompt

A.2 Dataset Composition

We collate eight tasks from the SuperNI dataset (Wang et al., 2022) as the experimental dataset referring to (Wang et al., 2024). The dataset stores each instruction in a separate file, and the file begins with a unique id (e.g., task001_quoref_question_generation.json). Table 4 lists all corresponding task IDS for the eight tasks we selected for our main experiments. In the process of filtering the data, we avoid as much as possible the performance impact due to cutoff during training by limiting the number of data instances corresponding to instructions and the data length.

Also to ensure the robustness of the method, we designed three different training orders. We use the order from easy to hard as the default order according to the curriculum learning method (Bengio et al., 2009). The specific order details are in Table 5.

B Experimental Details

B.1 Hardware

The proposed method performs training and evaluation on eight Nvidia Tesla V100 GPUs and an Intel Xeon Platinum 8255C CPU @ 2.50GHz

B.2 Training process Hyperparameters

We reproduce the experimental results of the methods of previous work (Wang et al., 2024; Huang et al., 2024) on our dataset. To ensure a fair comparison we uniformly set the training epoch to 1 in each training phase, and the optimizer is AdamW (Loshchilov and Hutter, 2017) with the cosine learning rate scheduler and warm-up, and the peak learning rate is set to 1e-5. The bacth size was set to 64.

Category	Task IDs
Classification	'069', '112', '1288', '1505', '1541', '1554', '298', '341', '343', '496'
Detection	'088', '209', '316', '319', '325', '333', '337', '608', '609', '905'
Sentiment Analysis	'1292', '1310', '475', '478', '494', '587', '823', '875', '903', '929'
Extraction	'039', '1447', '281', '292', '456', '578', '613', '620', '645', '683'
Program Execution	'063', '078', '079', '091', '125', '366', '369', '376', '497', '600'
Open QA	'075', '1286', '1419', '1420', '1431', '151', '194', '469', '591', '849'
Misc.	'1595', '1596', '169', '305', '306', '307', '308', '383', '567', '924'
Generation	'067', '1217', '1342', '1660', '453', '500', '574', '592', '649', '847'

Table 4: We integrated eight tasks from the SuperNI dataset. The task IDs contained in each category are reported.

Order	Task Sequence
Curriculum	Classification → Detection → Sentiment Analysis → Extraction → Program Execution → Open QA → Misc. → Generation
Reverse	Generation → Misc. → Open QA → Program Execution → Extraction → Sentiment Analysis → Detection → Classification
Random	Extraction → Detection → Program Execution → Sentiment Analysis → Generation → Open QA → Misc. → Classification

Table 5: Continual Learning Orders in eight tasks.