Optimizing Lifelong Fine-Tuning for Multiple Tasks via Dataless Distribution Replay

Zhenxing Wang
Peng Cheng Laboratory
zhenxingw086@gmail.com

Abstract

The recent emergence of various large language models, which can be fine-tuned with minimal instruction data, has demonstrated impressive performance across various tasks. However, a phenomenon of forgetting occurs during lifelong fine-tuning because training on new tasks interferes with the previously acquired knowledge. To mitigate catastrophic forgetting, conventional data replay methods achieve high performance, but at the cost of compromising data privacy and security. This paper introduces a dataless distribution replay approach for lifelong fine-tuning. Concretely, the distribution distillation is applied to replay the output distribution of the linear layers at previous task stages. The optimal solution for this distribution replay can be directly computed using the retained inner product matrix of the input data, thereby eliminating the need for previous data. Additionally, Singular Value Decomposition (SVD) and module accumulation are employed to further enhance the performance of dataless distribution replay method. Finally, the evaluation is conducted in a lifelong fine-tuning scenario involving multiple tasks. The experimental results and analysis show that the proposed method achieves significant improvements compared to several strong lifelong fine-tuning methods.

1 Introduction

The recent development of large language models, capable of being fine-tuned with minimal instructional data, has shown remarkable performance across a wide range of tasks. In practical scenarios, due to the difficulty in obtaining all relevant data at once and the high cost of resources, it is necessary to perform lifelong fine-tuning across multiple tasks (Zhuang et al., 2020), rather than training the model on all the data at once. However, a phenomenon known as catastrophic forgetting (Kirkpatrick et al., 2017) occurs during lifelong fine-tuning because training on new tasks interferes with the previously acquired knowledge. Lifelong fine-tuning, also known as continual or incremental fine-tuning, enables a large language model to adapt to new tasks while preserving its knowledge of previous tasks. The Low-Rank Adaptation (LoRA) (Hu et al., 2021), characterized by its efficiency and lightweight nature, has become widely applied in fine-tuning a general large language model into a task-specific model. The experiment in this article is conducted by fine-tuning with LoRA.

Recently, diverse methods have been proposed to overcome the challenge of catastrophic forgetting. Regularization-based methods (Barone et al., 2017; Gu et al., 2022), which train the model on new tasks while minimizing changes to the original parameters, are widely employed. These methods may encounter issues related to either under-constraint or over-constraint. In addition, several studies attempt to freeze the parameters that are relevant to previous tasks and utilize the remaining parameters to train new tasks (Zhuang et al., 2020; Gu et al., 2022). Nevertheless, this method might compromise critical information from previous tasks and struggle to achieve ideal results in lifelong fine-tuning across multiple tasks. Other approaches advance lifelong fine-tuning by implementing data replay training (Lopez-Paz and Ranzato, 2017; Garcia et al., 2021). These strategies require either the retention of old data or the creation of synthetic data to preserve previously acquired knowledge. However, in practical settings, the use of historical or synthetic data for lifelong fine-tuning poses challenges related to data quality and privacy.

This paper introduces a dataless distribution replay approach for lifelong fine-tuning in LoRA. Unlike data replay methods, this newly designed strategy does not require previous data for training while maintaining the high performance of replay algorithms. Specifically, distribution distillation is

Proceedings of the 31st International Conference on Computational Linguistics, pages 11261–11273
January 19–24, 2025.

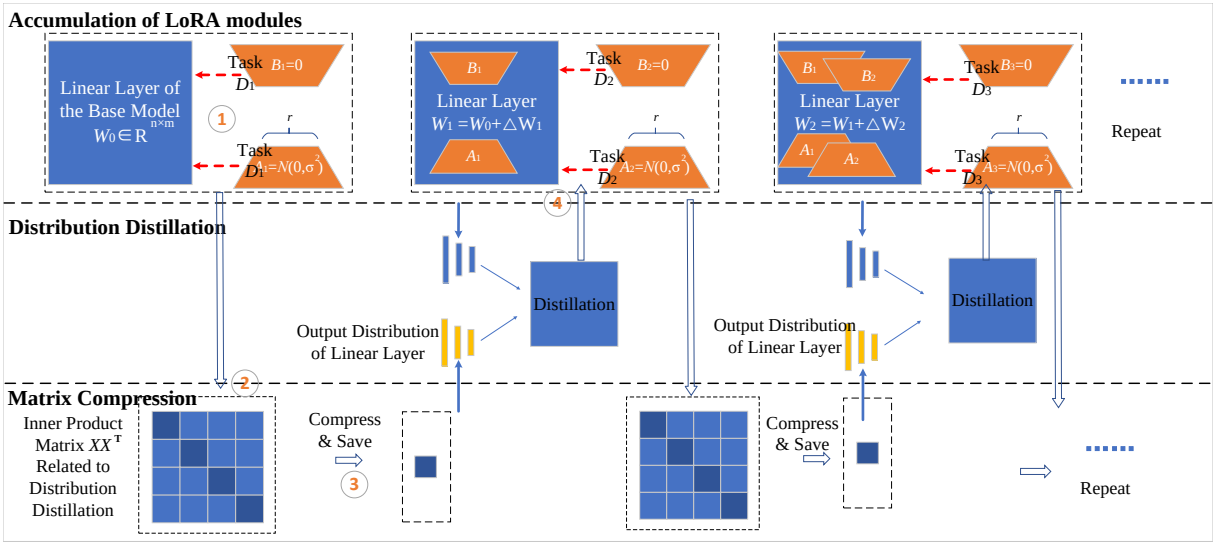


Figure 1: The main architecture of the proposed method. Fisrt, LoRA fine-tuning is conducted on the task D_1 and the LoRA module is integrated into the base model. The red dashed arrows indicate the integration of the LoRA module. Subsequently, the inner product matrix XX^T is calculated for each linear layer's input data X, which is related to distribution distillation. These matrices are then compressed and saved. After that, a new LoRA module is employed to train task D_2 and distribution distillation is performed to replay the output distribution of the linear layers at previous task stages. Similarly, subsequent task stages can utilize the inner product matrices saved at earlier stages to perform distribution distillation.

employed to replay the output distribution of the linear layers at previous task stages. The optimal solution for this distribution replay can be directly computed using the retained inner product matrix of the input data, thereby eliminating the need for previous data. Distribution replay can also reduce the model's overfitting to the current task, which enhances the model's generalization ability for unseen tasks. Additionally, SVD compression is used to reduce the size of the large inner product matrix related to the distribution replay, which maintains the advantage of lightweight file storage in LoRA fine-tuning. Moreover, to further enhance the performance of lifelong fine-tuning in LoRA, the accumulated LoRA modules are combined with the distribution replay method. The accumulated LoRA modules achieve excellent fine-tuning performance in non-lifelong fine-tuning scenarios (Meng et al., 2024). At each task stage, the previous LoRA module, which is no longer active in training, is integrated into the base model. Subsequently, at the new task stage, SVD compression and distribution distillation are performed. The main architecture of the proposed approach is illustrated in Figure 1

This paper's contribution can be summarized as follows:

- To mitigate catastrophic forgetting and enhance the model's zero-shot capabilities for unseen tasks, the dataless distribution distillation approach is utilized to replay the output distribution of the linear layers at previous task stages. This approach does not require training on previous data and helps to reduce the model's overfitting to the current task.
- To further optimize the lifelong fine-tuning approach, SVD compression is employed to maintain the advantage of lightweight file storage in LoRA fine-tuning. In addition, the accumulated LoRA modules are utilized to enhance the effectiveness of the distribution distillation method.
- This paper conducts extensive experiments in the lifelong fine-tuning scenario of multiple tasks and publicly available datasets. Across multiple metrics and various task orders, the method proposed in this paper achieves significant improvements on the continual tasks compared to several strong lifelong fine-tuning methods. In addition, the proposed method demonstrates a distinct advantage in zero-shot capabilities for unseen tasks.

2 Related Work

Recently, numerous researchers have developed various methods to overcome the challenge of catastrophic forgetting.

Regularization-based methods. These approaches are designed to minimize fluctuations in the original parameters while learning new task data (Barone et al., 2017; Gu et al., 2022). Kirkpatrick et al. (2017) use the Fisher information matrix to assess the importance weights of different parameters. Cha et al. (2020) incorporate regularization terms into the classifier to promote wide local minima, thereby maximizing the entropy of the classifier's output distribution. Memory aware synapses (MAS) (Aljundi et al., 2018) determines the importance of parameters based on the gradients of model's output. However, these methods may face issues of insufficient or excessive constraints.

Structure-based methods. These methods Rücklé et al. (2020); Zhuang et al. (2020) assign specific parameters for different tasks or expand the network's parameters to accommodate new tasks. Huang et al. (2023) construct plug-and-play modules with additional parameters tailored for training new data. Madotto et al. (2020) utilize a residual adapter-based approach to construct new parameters. Furthermore, Dabre et al. (2020) suggest initializing newly added parameters with pseudo-data. However, these approaches can cause the model to become increasingly bloated in the context of multi-stage lifelong fine-tuning. Another approach (Zhuang et al., 2020) involves freezing the main parameters of the model and fine-tuning only a subset. Gu et al. (2022) utilize the Hessian matrix to identify high-forgetting risk regions for parameter fixing and define the variation regions for new tasks. However, these methods may encounter challenges due to either excessive or insufficient parameter freezing.

Data-replay-based methods. These methods need to store data from previous stages (de Masson D'Autume et al., 2019; Wang et al., 2020) or generate synthetic data (Kanwatchara et al., 2021), which is used in the new stage. To mitigate the increased loss of previous tasks, the Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) retrieves historical data from memory to compute gradients. Garcia et al. (2021) propose a method that involves dictionary replacement to generate pseudo-data. Another study (Yin et al., 2020) attempts to achieve knowledge distillation by synthesizing data.

Although data replay methods demonstrate superior performance among lifelong fine-tuning approaches, they raise concerns regarding privacy and security. In this paper, distribution distillation is utilized to effectively replay the output distribution of the linear layers at previous task stages. This approach does not require previous data for training and demonstrates a distinct advantage in zero-shot capabilities for unseen tasks compared to data replay methods.

3 Method

In the setting of lifelong fine-tuning, a model f with parameters θ is sequentially trained on a stream of tasks $D = \{D_1, \ldots, D_N\}$. The output of the model f on the task D_t is denoted as $f(D_t, \theta_t)$, where θ_t represents the model parameters at the t-th stage. To mitigate the issue of catastrophic forgetting in lifelong fine-tuning, a dataless distribution replay method is proposed, which involves three main steps: Accumulation of LoRA Modules, Distribution Distillation, Matrix Compression. Accumulation of LoRA Modules and Matrix Compression aim to further optimize the performance of Distribution Distillation.

3.1 Preliminaries

3.1.1 LoRA Fine-Tuning

Training with LoRA involves freezing the parameters of the base model and constructing a trainable LoRA module. The optimization for the entire model is completed through the training of the LoRA module ΔW. The modified forward propagation in linear layer W, related to LoRA module, is $X^T W + X^T \Delta W$, where $X \in \mathbb{R}^n$ denotes the input data of linear layers W, $\Delta W = AB$, $A \in \mathbb{R}^{n \times r}$, $B \in \mathbb{R}^{r \times m}$, and $r << n, m$. Here n and m respectively denote the dimensions of the input and output of the linear layer in the base model. The rank r of the parameter matrix can be selected to be relatively low, resulting in a LoRA module that is substantially smaller than the base model.

3.1.2 Null Space

To mitigate the issue of forgetting, Wang et al. (2021) proposed the definition of the null space. In the definition of null space, only the linear layers W in the model f are trained. If the update value ΔW^* for the parameters W lies in the null space of X, i.e.,

$$X^T \Delta W^* = 0 \tag{1}$$

where X denotes the input to the linear layers W. If the update value of all linear layers lies in the null

space, then $f(D_t, \theta_t) = f(D_t, \theta_{t+1})$, where θ_{t+1} represents the model parameters at the $(t+1)$-th stage, as can be readily demonstrated. When only the parameters of the linear layers in the model are updated and the outputs from these linear layers remain identical to those before the update, the output of the entire model remains unchanged. The null space ensures that the model maintains stable outputs on previous tasks while being trained on new tasks.

3.2 Accumulation of LoRA Modules

Initially, a LoRA module is utilized to train task D_1, where each LoRA module ΔW_1 comprises initializing A_1 and B_1 matrices. Then, this LoRA module ΔW_1 is no longer active in training and is integrated into the linear layers W_0 of the base model, resulting in updated linear layers W_1 for the next stage.

$$W_1 = W_0 + \Delta W_1 \tag{2}$$

In the subsequent task, a new LoRA module ΔW_2 is constructed. For the next task, the above process is repeated. At the t-th task stage, the linear layers W_t of the model are denoted as:

$$W_t = W_0 + \Delta W_1 + \Delta W_2 + ... + \Delta W_t \tag{3}$$

3.3 Distribution Distillation

Here, the output distribution of linear layers can be expressed as $X_i^T W_i$, where W_i represents the linear layers at the i-th task stage, and X_i denotes the input of the linear layers W_i. Note that parameters unrelated to the LoRA module are not updated and thus are not discussed in this paper. At the t-th task stage, to replay the output distribution of the linear layers in the previous task stages, the objective function for distribution distillation can be formulated as follows:

$$\min_{\Delta W_t} L_t + \sum_{i=1}^{t-1} ||X_i^T(W_{t-1} + \Delta W_t) - X_i^T W_i||^2 \tag{4}$$

where the first term represents the loss function training the task D_t, and the second term represents the distillation of the output distribution from the linear layers at previous task stages. ΔW_t denotes the LoRA module of the model at the t-th stage, and W_{t-1} denotes the linear layers at the $(t-1)$-th stage, which remain untrained. Direct application of Equation 4 requires retaining the input to the linear layer for each token. Therefore, the null space is utilized to implement a distribution distillation process that does not require training with input data of previous linear layers. At the t-th task stage, the model initially undergoes training on the task D_t. Subsequently, the null space ΔW_t^* of input data X_t is explored while distilling the output distribution of the linear layers at previous task stages. The ideal null space for input data X_t at the t-th task stage is as follows:

$$X_t^T \Delta W_t{}^* = 0 \tag{5}$$

Specifically, the null space ΔW_t^* represents the update value of parameter W_t and also serves as the LoRA value of the model. This indicates that changes in W_t do not affect the output results on the task D_t. Given that it is too strict to guarantee the existence of a null space, an approximate null space is proposed based on the following formula:

$$\min_{\Delta W_t^*} | \, ||X_t^T \Delta W_t^*||^2 \tag{6}$$

Therefore, the new distribution distillation formula is designed as follows:

$$\min_{\Delta W_t^*} ||X_t^T \Delta W_t^*||^2 + \sum_{i=1}^{t-1} ||X_i^T(W_t + \Delta W_t^*) - X_i^T W_i||^2 \tag{7}$$

where the first term in the equation ensures that the output of the linear layers at the t-th stage remains stable, while the second term achieves the distillation of the output distribution from the linear layers at previous task stages. Inspired by the Regression Mean method (RegMean) (Jin et al., 2022) [1], this optimization problem can be directly solved. Furthermore, by combining Equation 3, the final solution is as follows:

$$\Delta W_t^* = (\sum_{i=1}^{t} X_i X_i^T)^{-1} \left[\sum_{i=1}^{t} X_i X_i^T \left(\sum_{j=1}^{i} \Delta W_j \right) \right] + W_0 - W_t \tag{8}$$

The derivation process is detailed in Appendix A. This formula demonstrates that distillation training and storing the input data $X_i \in \mathbb{R}^n$ of the linear layers are unnecessary; instead, the inner product matrix $X_i X_i^T \in \mathbb{R}^{n*n}$ of the input data can be stored for distribution distillation. In addition, it

[1] https://github.com/bloomberg/dataless-model-merging

should be noted that the size of this matrix is significantly larger than that of the original LoRA module $\Delta W = AB$, where $A \in \mathbb{R}^{n*r}$, $B \in \mathbb{R}^{r*m}$ and $r << n$. In addition, since an input data involves multiple tokens, the information of the input data is further protected through cumulative operations.

3.4 Matrix Compression

Here, SVD is employed to compress the matrix XX^T and reduce noise by storing its diagonal values. The compression of matrix XX^T can be represented as:

$$U\Sigma V^T = XX^T \tag{9}$$

where U and V denote orthogonal matrices, and Σ denotes a diagonal matrix. For the purpose of compression, only the first k largest singular values are retained. This truncated diagonal matrix is denoted by Σ_k. U_k and V_k are the corresponding subsets of U and V. The computation to reconstruct the matrix can be expressed as follows:

$$XX^T \approx U_k \Sigma_k {V_k}^T \tag{10}$$

Since XX^T is a symmetric matrix, it is sufficient to store either U_k or V_k.

Then, during the compression and reconstruction process, the diagonal values are stored as a vector d. The complete reconstruction process is as follows:

$$XX^T = \Theta(U_k \Sigma_k {V_k}^T) + diag(d) \tag{11}$$

where Θ denotes the operation of setting the diagonal values of a matrix to zero, while $diag$ refers to the process of transforming a vector d into a diagonal matrix.

4 Experiments

4.1 Dataset and Metric

Datasets. The tasks employed in this study include sentiment analysis (dataset: AGNews) (Zhang et al., 2015), news classification (dataset: Yelp) (Zhang et al., 2015), and question answering (dataset: SQuAD) (Rajpurkar et al., 2016). Furthermore, the widely used continual translation task (dataset: German-English (De-En) translation) (Gu et al., 2022)—is incorporated into the experiments of this paper. For the size of each dataset, the sampling strategy outlined in de Masson D'Autume et al. (2019) is adopted. The specific sizes of the datasets are shown in Appendix B.

Metric. For the four tasks mentioned, the sacreBLEU (Post, 2018) score for the translation task and the Exact Match (EM) score for the remaining tasks were recorded. Additionally, it is important to note that model performance varies across different task stages. For instance, the evaluation results of the first task at the second stage differ from those obtained at the third stage. Therefore, to standardize the evaluation results, a strategy commonly adopted in previous studies (Lopez-Paz and Ranzato, 2017) is utilized to assess the performance of lifelong fine-tuning algorithms. The strategy in Lopez-Paz and Ranzato (2017) includes three metrics: **AS**: The average performance of the model across all tasks at the final stage; **FWT**: The model's generalization ability on unseen tasks; **BWT**: The influence of learning subsequent tasks on earlier tasks. These metrics are detailed in Appendix C.

4.2 Implementation Details

The training framework in this paper uses the Hugging Face Transformers library. For LoRA, the PEFT library in the Hugging Face API [2]is utilized. The rank of LoRA is set to 8 and the scaling factor is set to 16. The LoRA to the self-attention's query and value matrices is applied for training large language model, and the LoRA dropout rate is set to 0.1. For the number k of singular values retained in SVD, considering both the size of the compressed file and the model's performance, k=100 is set, where the original dimensions of the inner product matrix are 2048×2048. More details on model training are provided in Appendix D

Base Model. To maintain consistency with previous work (Scialom et al., 2022), the T0-3B model (Sanh et al., 2022) is employed as base model. The T0 model is a multi-task large model that undergoes pre-training across a broader range of NLP tasks, exhibiting enhanced universality and generalization capabilities[3].

Baselines. All the comparison methods are implemented using LoRA.

Sequential LoRA Fine-tuning (SLFT). The model is sequentially fine-tuned on the same LoRA, one task after another.

L2 (Barone et al., 2017)**.** This method applies L2-norm regularization to the parameters of the previous model.

[2]https://github.com/huggingface/peft

[3]https://huggingface.co/bigscience/T0_3B

	AS	FWT	BWT
SLFT	56.23	0.00	-17.13
L2	59.43	13.08	-6.52
EWC	58.66	0.00	-13.87
LFR-CM	$62.97_{\pm 0.18}$	$10.47_{\pm 0.69}$	$-3.37_{\pm 0.27}$
RandSam	$66.91_{\pm 0.04}$	$0.71_{\pm 0.11}$	$-2.71_{\pm 0.18}$
LAFT	$59.47_{\pm 0.89}$	$0.41_{\pm 0.39}$	$-12.36_{\pm 0.82}$
LADD	$\mathbf{68.22}_{\pm 0.13}$	$\mathbf{19.82}_{\pm 0.92}$	$\mathbf{-0.48}_{\pm 0.10}$
MTL	$68.73_{\pm 0.06}$	/	/

Table 1: Final results of different algorithms on three metrics. The algorithms are trained on the default order: AGNews→De-En→Yelp→SQuAD. Bold indicates the best results for all methods, excluding MTL, under each metric, and details on three metrics are in Appendix C. ± indicates the standard deviation over three runs, which captures the variability in the comparison results of several high-performance methods.

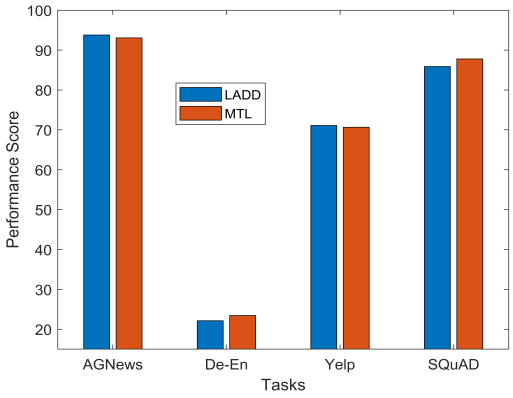


Figure 2: Specific scores of LADD and MTL on continual four tasks (AGNews, De-En, Yelp, SQuAD). The performance of each task is assessed using the model obtained at the final stage of the default order.

EWC (Kirkpatrick et al., 2017)**.** This method involves adding a regularization term to the loss function, which is calculated based on the Fisher Information Matrix. This matrix measures the importance of network parameters with respect to the output of previous tasks.

LFR-CM (Gu et al., 2022)**.** This method identifies high-forgetting risk regions for parameter fixing and defines the variation regions for training a new task.

RandSam (Scialom et al., 2022)**.** This method utilizes random sampling of data from previous tasks to participate in the training of a new task. As indicated by Scialom et al. (2022), using 1% of sampled data can effectively preserve the learned knowledge of the model, so the same sampling rate is set in the experiments.

MTL (de Masson D'Autume et al., 2019)**.** The model is trained on all datasets jointly. **MTL** is often regarded as the upper bound of lifelong fine-tuning.

LoRA Accumulation Fine-tuning (LAFT). This is the method proposed in this paper. The LoRA from previous tasks is merged with the base model to form a new base model, then a new LoRA module is initialized to train a new task.

Distribution Distillation (LADD). In **LADD** method, the output distribution of the linear layers at previous task stages is replayed through distribution distillation, which further mitigates knowledge forgetting.

4.3 Main Results

As shown in Table 1, the performance of different methods on three metrics is compared, which is executed on the default order: AGNews→De-En→Yelp→SQuAD. Given the superior performance of the LADD method proposed in this paper over non-data replay methods, the mean and standard deviation from three runs for several high-performance comparative methods (LFR-CM, RandSam, LAFT, LADD, MTL) are presented to mitigate randomness. Clearly, the LADD method outperforms the competitors in all metrics.

AS metric. In the AS metric of Table 1, the proposed method LAFT demonstrates a clear advantage over the sequential LoRA fine-tuning approach (SLFT), achieving an AS value that is 3.24 higher. Additionally, when LAFT employs the distribution distillation approach, named LADD, to enhance knowledge preservation, it outperforms all comparative methods. Compared to the data replay algorithm RandSam, LADD shows a 1.31 improvement in the AS metric. Furthermore, it is evident that the LADD method exhibits a distinct advantage in the FWT metric, which is further analyzed in the subsequent content.

In addition, in comparison with the upper bound of the lifelong fine-tuning method, MTL, the performance of LADD is only 0.51 lower. Figure 2 details the values of the AS metric. Specifically, on the Yelp and AG datasets, the LADD method outperforms the upper bound of the lifelong fine-tuning method, MTL, by 0.20 and 0.83 points, respectively. The results highlight the advantages of

	order 1			order 2		
	AS	**FWT**	**BWT**	**AS**	**FWT**	**BWT**
SLFT	60.21	0.35	-8.87	58.95	16.72	-10.35
L2	60.84	23.20	-3.27	59.57	44.42	-6.02
EWC	60.32	0.37	-8.79	58.92	18.26	-10.46
LFR-CM	$60.14_{\pm 0.06}$	$25.97_{\pm 0.80}$	$-1.12_{\pm 0.06}$	$58.73_{\pm 0.38}$	$\mathbf{52.35}_{\pm 0.39}$	$-4.99_{\pm 0.40}$
RandSam	$64.09_{\pm 0.24}$	$11.15_{\pm 1.16}$	$-3.62_{\pm 0.35}$	$63.80_{\pm 0.26}$	$31.74_{\pm 0.81}$	$-4.08_{\pm 0.16}$
LAFT	$62.82_{\pm 0.20}$	$6.32_{\pm 0.90}$	$-5.61_{\pm 0.17}$	$61.89_{\pm 0.22}$	$21.70_{\pm 0.47}$	$-6.97_{\pm 0.42}$
LADD	$\mathbf{66.63}_{\pm 0.02}$	$\mathbf{30.22}_{\pm 0.62}$	$\mathbf{-0.23}_{\pm 0.04}$	$\mathbf{66.56}_{\pm 0.06}$	$51.15_{\pm 0.36}$	$\mathbf{-0.48}_{\pm 0.09}$
MTL	$66.70_{\pm 0.04}$	/	/	$66.70_{\pm 0.04}$	/	/
	order 2			**order 4**		
	AS	**FWT**	**BWT**	**AS**	**FWT**	**BWT**
SLFT	51.38	2.11	-20.53	56.66	23.52	-13.42
L2	61.39	14.60	-3.47	61.35	30.55	-3.73
EWC	55.53	2.68	-15.09	57.69	23.29	-11.99
LFR-CM	$60.60_{\pm 0.03}$	$20.33_{\pm 0.57}$	$-2.35_{\pm 0.11}$	$62.38_{\pm 0.08}$	$30.54_{\pm 0.11}$	$-0.64_{\pm 0.27}$
RandSam	$64.37_{\pm 0.08}$	$14.47_{\pm 1.23}$	$-3.39_{\pm 0.19}$	$61.73_{\pm 0.42}$	$30.18_{\pm 0.86}$	$-6.82_{\pm 0.54}$
LAFT	$59.54_{\pm 0.24}$	$8.22_{\pm 0.96}$	$-10.08_{\pm 0.32}$	$62.50_{\pm 0.08}$	$26.70_{\pm 0.98}$	$-6.00_{\pm 0.11}$
LADD	$\mathbf{66.68}_{\pm 0.05}$	$\mathbf{21.89}_{\pm 0.32}$	$\mathbf{-0.26}_{\pm 0.02}$	$\mathbf{66.47}_{\pm 0.31}$	$\mathbf{31.05}_{\pm 0.06}$	$\mathbf{-0.36}_{\pm 0.22}$
MTL	$66.70_{\pm 0.04}$	/	/	$66.70_{\pm 0.04}$	/	/

Table 2: Final results on continual four tasks under various task orders. Bold indicates the best results for all methods, excluding MTL, under each metric. $\pm$ indicates the standard deviation over three runs, which captures the variability in the comparison results of several high-performance methods. Details on task orders are in Appendix K.

the distribution distillation method. This is mainly attributed to the method's ability to effectively replay the previous output distribution while minimizing its impact on the current task.

FWT metric. In the FWT metric of Table 1, LADD exhibits exceptional performance. Compared to the second-best algorithm L2, LADD's FWT value is 6.74 higher, demonstrating LADD's excellent zero-shot capabilities for unseen tasks. This result is primarily attributed to the mitigation of overfitting to the current task by distribution replay. More detailed experiments and analyses are presented in Appendix E. In addition, a detailed analysis of the **BWT** metric is provided in Appendix G.

Results of different task orders. In multi-task lifelong fine-tuning, the order of tasks can influence the final model's performance. Table 2 presents the results of different algorithms across four task orders. For more details on task orders, refer to Appendix K. Given the computational costs, the dataset for each task was derived by randomly selecting a 10% sample from the original datasets. Table 2 clearly demonstrates that LADD exhibits the best performance across three metrics in four different task orders.

Even when compared with the results from MTL, the AS value of LADD are remarkably close. Table 2 shows that the LADD method achieved an average AS value of 66.58 across four task orders, which is only 0.12 lower than the AS value of MTL. Moreover, compared to the baseline methods, the LADD method demonstrates greater stability across various task orders. Detailed experimental results are presented in Appendix F. These results indicate that LADD significantly outperforms comparative methods. Although the advantage of LADD over the data replay algorithm RandSam in the AS metric diminishes as the proportion of replay increases, it maintains a significantly superior zero-shot capability for unseen tasks. This is further analyzed in subsequent sections based on the results from Figure 4. On the other hand, data replay algorithms may face privacy and security issues, which may restrict access to previous data.

4.4 Analysis

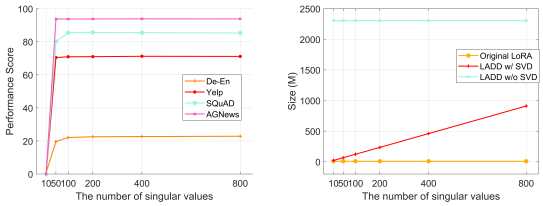
Ablation study of LoRA accumulation and distribution distillation. Firstly, Table 3 illustrates the effect of LoRA accumulation. When combined with distribution distillation, this accumulation can further enhance the model's performance. This can be attributed to the accumulated LoRA modules' capability to preserve prior knowledge to some

extent. Furthermore, Table 3 illustrates the pivotal role of distribution distillation. In the AS and FWT metrics, distribution distillation yields improvements of at least 8.32 and 15.17 points, respectively.

LoRA accum	Dis dist	AS	FWT
×	×	56.23	0.00
×	✓	67.54	15.17
✓	×	59.90	0.16
✓	✓	68.22	19.82

Table 3: Ablation studies on LoRA accumulation (LoRA accum) and distribution distillation (Dis dist) on the default order.

Effect of the number of singular values on performance. In the distribution distillation approach, SVD plays an important role. Figure 3a illustrates how the number of singular values in SVD affects model's performance by determining the extent of matrix compression. When the number of singular values is set to 10, the model's prediction scores across various tasks are zero, which is attributed to low singular values introducing excessive noise. When the number of singular values varies from 100 to 800, the model's performance remains stable. This stability is primarily due to the adequacy of a specific number of singular values in preserving significant information from the original matrix.



(a) Specific results on continual four tasks.

(b) File size required for storage at each stage.

Figure 3: Impact of different number of singular values on the default order.

Effect of the number of singular values on size. Figure 3b illustrates the variations in the size of the retained matrix with varying numbers of singular values. Although the final retained files are relatively larger than the original LoRA files, they are substantially smaller than the uncompressed files. Specifically, when the number of singular values is set to 100, the original matrix can be compressed by a factor of approximately 20.

Comparison of LADD and data replay methods with different sampling ratios. In non-data replay algorithms, the advantages of the LADD algorithm are highly evident. Figure 4 further shows the comparative performance of LADD and replay algorithms with various sampling ratios under task order 1. The other three orders are detailed in Appendix I, and the data size is the same as shown in Table 2. It is clear that when the sampling ratio reaches 32%, the AS value of LADD is only 0.15 lower than that of the replay algorithm. However, LADD's FWT value exceeds that of the replay method by 9.94. Additionally, replay algorithms may encounter privacy and security issues in practical scenarios, potentially rendering previous data inaccessible. These factors all indicate that the LADD method offers distinct advantages.

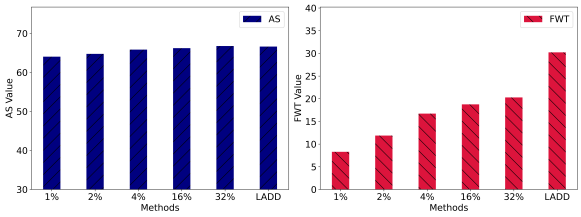


Figure 4: Comparisons of LADD and data replay algorithms with different sampling ratios on task order 1.

Performance on different base models. Given the influence of different base models on lifelong fine-tuning methods, simple experiments were conducted with various base models, including MT0-XL (3.7B) (Muennighoff et al., 2022) and T0pp (13B), as detailed in Appendix H. MT0-XL is a multilingual and multitask model, and T0pp is a larger-scale model. The results in Table 6 and 7 demonstrate that LLAD surpasses the replay algorithm by 1.18 and 1.26 under MT0-XL and T0pp models, respectively, indicating the universal applicability of the LADD method.

5 Conclusion

This paper introduces the dataless distribution replay method to effectively address the issue of catastrophic forgetting during multi-task lifelong fine-tuning in LoRA. Concretely, distribution distillation is employed to replay the output distribution of the linear layers at previous task stages. The optimal solution for this distribution replay can be directly computed, eliminating the need for training. Simultaneously, SVD compression is used to reduce the size of large storage files related to distribution distillation. Moreover, the accumulated

LoRA modules are combined with the distribution distillation method, which further enhances the performance of lifelong fine-tuning in LoRA . The experimental results show that the method proposed in this paper achieves significant improvements compared to several strong baselines.

Limitations

the method proposed in this paper is designed to address the forgetting issue during lifelong fine-tuning in LoRA. For full-parameter fine-tuning, this requires further investigation in the future. Furthermore, in the AS metric, the approach proposed in this paper underperforms compared to data replay methods with high sampling ratios and requires additional storage for the compressed inner product matrix.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. *arXiv preprint arXiv:1707.09920*.

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*.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Computing Surveys (CSUR)*, 53(5):1–38.

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.

Xavier Garcia, Noah Constant, Ankur P Parikh, and Orhan Firat. 2021. Towards continual learning for multilingual machine translation via vocabulary substitution. *arXiv preprint arXiv:2103.06799*.

Shuhao Gu, Bojie Hu, and Yang Feng. 2022. Continual learning of neural machine translation within low forgetting risk regions. *arXiv preprint arXiv:2211.01542*.

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*.

Kaiyu Huang, Peng Li, Jin Ma, Ting Yao, and Yang Liu. 2023. Knowledge transfer in incremental learning for multilingual neural machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15286–15304.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Kasidis Kanwatchara, Thanapapas Horsuwan, Piyawat Lertvittayakumjorn, Boonserm Kijsirikul, and Peerapon Vateekul. 2021. Rational lamol: A rationale-based lifelong learning framework. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2942–2953.

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.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

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*.

Xiangdi Meng, Damai Dai, Weiyao Luo, Zhe Yang, Shaoxiang Wu, Xiaochen Wang, Peiyi Wang, Qingxiu Dong, Liang Chen, and Zhifang Sui. 2024. Periodiclora: Breaking the low-rank bottleneck in lora optimization. *arXiv preprint arXiv:2402.16141*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

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

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

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. 2021. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193.

Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime Carbonell. 2020. Efficient meta lifelong-learning with limited memory. *arXiv preprint arXiv:2010.02500*.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.

A The derivation process of the optimization problem

Inspired by Paper (Jin et al., 2022), the optimization process for the objective function described in Equation 7 is as follows:

First, the objective function is denoted as L, and then compute the gradient of the objective function L w.r.t $\Delta {W_t}^*$. Since L is convex w.r.t. $\Delta {W_t}^*$, the solution for $\Delta {W_t}^*$ can be directly obtained by letting $\frac{\partial L}{\partial \Delta {W_t}^*} = 0$.

$$
\begin{aligned}
&\frac{\partial L}{\partial \Delta {W_t}^*} = X_t {X_t}^T \Delta {W_t}^* + \sum_{i=1}^{t-1} X_i {X_i}^T (W_t \\
&\quad + \Delta {W_t}^*) - \sum_{i=1}^{t-1} X_i {X_i}^T (W_i) = 0 \\
\Rightarrow \quad & X_t {X_t}^T (W_t + \Delta {W_t}^* - W_t) + \sum_{i=1}^{t-1} X_i {X_i}^T (W_t \\
&\quad + \Delta {W_t}^*) - \sum_{i=1}^{t-1} X_i {X_i}^T (W_i) = 0 \\
\Rightarrow \quad & \sum_{i=1}^{t} X_i {X_i}^T (W_t + \Delta {W_t}^*) - \sum_{i=1}^{t} X_i {X_i}^T (W_i) = 0 \\
\Rightarrow \quad & \Delta {W_t}^* = (\sum_{i=1}^{t} X_i {X_i}^T)^{-1} \sum_{i=1}^{t} X_i {X_i}^T (W_i) - W_t
\end{aligned}
\tag{12}
$$

Since $W_i = W_0 + \Delta W_1 + ... + \Delta W_i$, the following result can be derived:

$$
\begin{aligned}
\Delta {W_t}^* = &(\sum_{i=1}^{t} X_i {X_i}^T)^{-1} \left[\sum_{i=1}^{t} X_i {X_i}^T \left(\sum_{j=1}^{i} \Delta W_j \right) \right] \\
&+ W_0 - W_t
\end{aligned}
\tag{13}
$$

B Data Statistics

Here, the specific data sizes for the four datasets are demonstrated (DE-EN, Yelp, AGNews, SQuAD).

Dataset	#Train	#valid	#Test
DE-EN	90000	4000	4000
Yelp	115000	4000	7600
AGNews	115000	4000	7600
SQuAD	83000	4000	10570

Table 4: The sizes of four datasets.

C The three metrics for continual tasks

The specific definitions of the three metrics are as follows: Here, S_i^j denotes the performance score of the model on the task D_i at the j-th stage.

- **Average Score (AS).** It is used to quantify the final average performance of the model across all T tasks at the final stage T, which is defined as follows:

$$
\mathrm{AS} = \frac{1}{T} \sum_{i=1}^{T} S_i^T \tag{14}
$$

where S_i^T denotes the performance score of the model on the task D_i at the final stage T.

- **Forward Transfer (FWT).** This metric assesses the model's generalization ability on unseen tasks. It measures the average zero-shot performance S_i^{i-1} for unseen tasks D_i at every stage $i-1$.

$$\mathrm{FWT} = \frac{1}{T-1}\sum_{i=2}^{T} S_i^{i-1} \tag{15}$$

- **Backward Transfer (BWT).** This metric is employed to measure the influence of learning subsequent tasks on earlier tasks. Except for the ultimate one, it compares the performance S_i^T at the final stage T with the immediate online performance S_i^i at stage i. It denotes the extent to which the model has forgotten previously acquired knowledge.

$$\mathrm{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1} (S_i^T - \mathrm{S}_i^i) \tag{16}$$

D Training details

the AdamW_Torch optimizer with β_1= 0.9 and β_2= 0.999 is utilized, setting the batch size to 8, and the weight decay to 3e-7. A cosine learning rate schedule is utilized, with a warmup ratio of 0.1 and a maximum learning rate of 3e-4. All the systems are trained on 2 A100 GPUs. Following the previous strategy (Sun et al., 2019), the final checkpoint for evaluation is used. For continual matrix compression, two approaches can be used. The first approach is to reduce the size of the matrix by first accumulating the previously stored inner product matrix and the current inner product matrix, and then performing compression. To improve performance, the second approach is to compress the inner product matrix at each stage and save it separately. The first approach is used in the four continuous task experiments in this paper.

E The analysis of zero-shot results for unseen tasks

Table 5 presents the performance of the model on specific tasks at each sequential stage. In stage 1, due to the lack of prior tasks, no distribution distillation occurs. The results indicate that while the LAFT method provides effective results for the current task, AGNews, it fails to achieve satisfactory outcomes for the subsequent task, De-En. This is attributed to overfitting to the current task, resulting in the loss of the model's generalization ability. In stage 2, when the LAFT algorithm does not perform distribution distillation, it results in the model losing its generalization ability for unseen tasks. However, after performing distribution distillation, the model exhibits a slight decrease of 0.25 points in performance on the current task, but demonstrates a significant enhancement of 25.14 points in its generalization ability for unseen tasks. In stage 3, the benefits of distribution distillation in enhancing the generalization ability are similarly evident.

	stage 1	
	AGNews (S_1^1)	De-En (S_2^1)
LAFT		
w/o Dis dist	93.90	0.00
	stage 2	
	De-En (S_2^2)	Yelp (S_3^2)
LAFT		
w/o Dis dist	23.52	0.51
w/ Dis dist	23.27	25.65
	stage 3	
	Yelp (S_3^3)	squad (S_4^3)
LAFT		
w/o Dis dist	71.22	0.00
w/ Dis dist	71.05	33.81

Table 5: Specific effects of the distribution distillation (Dis dist) method on the LAFT method across various stages of the default order. S_i^i represents the performance score of the model on task D_i at stage i, while S_i^{i+1} indicates the performance score on the subsequent task D_{i+1} at stage i. Therefore, S_i^{i+1} illustrates the model's zero-shot capabilities for unseen task.

F Statistical results for four task orders

Figure 5 illustrates the mean and standard deviation of the AS value for different methods across various orders, revealing that LADD outperforms the others in both metrics. Specifically, the standard deviation of the AS value is 0.08, which is the lowest among all methods. These results demonstrate that LADD maintains high stability while preserving superior performance across various task orders.

G The analysis of the BWT metric

According to the BWT metric in Table 1, LADD improves the BWT value by 2.23 points compared to other methods. Since BWT measures the

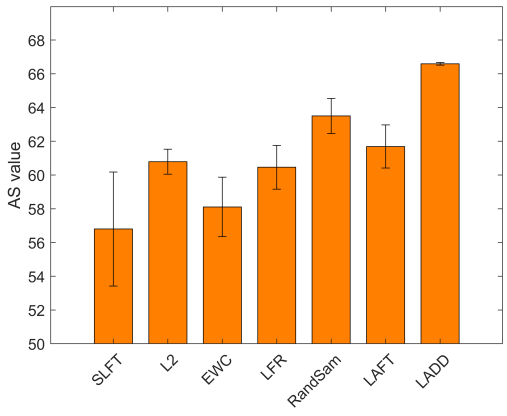


Figure 5: The mean and standard deviation of the AS values of different algorithms under different task orders.

	order1 AS	order2 AS	order3 AS	order4 AS
RandSam	66.16	66.28	66.96	65.57
LADD	**67.46**	**67.54**	**67.47**	**67.24**

Table 6: Results of different algorithms on MT0-XL model. Bold indicates the better results.

model's resistance to forgetting, this result indicates that LADD effectively prevents catastrophic forgetting during lifelong fine-tuning.

H Performance on different base models

In the MT0-XL model experiment (Table 6), the same dataset and task orders as presented in Table 2 are used to conduct experiments across four task orders. Since the LADD method outperforms non-data replay methods, only the data replay method is employed as a comparative approach. In this experiment, the singular value is set to 400. In addition, due to resource limitations, the experiments (Table 7) on task Order 1 in the T0pp model are conducted using the same dataset presented in Table 2. In this experiment, the singular value is set to 100.

I The performance of the LADD method and the data replay methods with different sampling ratios.

Figures 6, 7, and 8 show the performance of LADD compared to replay algorithms with different sampling ratios across three task orders. In the AS metric, LADD's performance is essentially equivalent to that of replay algorithms with high sampling ratios across these task orders. In the FWT metric, LADD's performance for order 4 is essentially equivalent to that of the replay algorithms. However, for order 2 and order 3, LADD significantly outperforms all replay algorithms.

	AGNews	De-En	Yelp	SQuAD
RandSam	**90.43**	20.52	68.47	**87.55**
LADD	90.30	**23.72**	**70.77**	87.24

Table 7: Results of different algorithms on T0pp model. Bold indicates the better results.

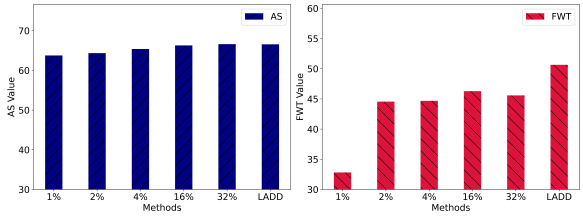


Figure 6: Comparisons of LADD and data replay algorithms on task order 2.

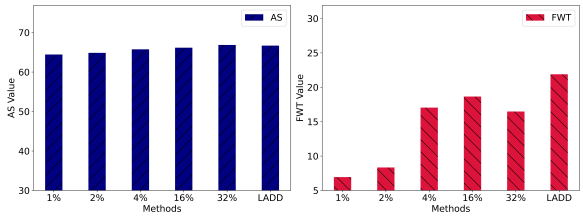


Figure 7: Comparisons of LADD and data replay algorithms on task order 3.

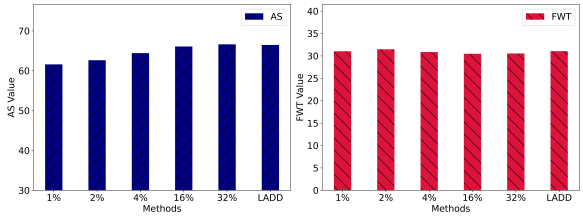


Figure 8: Comparisons of LADD and data replay algorithms on task order 4.

J The experiments on more tasks

Here, four tasks (Cola, Mnli, Mrpc, QQP) from the GLUE dataset are added for continual learning. The task order is (AGNews→De-En→Yelp→SQuAD→Cola→Mnli→Mrpc→QQP). The comparison method we chose is the highest-ranked RandSam algorithm from previous comparison methods. Clearly, our method outperforms the competitor. The experimental results are presented in Figure 9. For continuous matrix compression, the inner product matrix at each stage is compressed and saved separately.

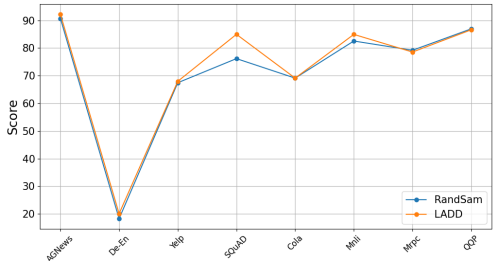


Figure 9: Performance of Different Algorithms on Supplementary Tasks

The performance of each task is assessed using the model obtained at the final stage of the order.

K Different task orders

Experiments were conducted using the following four distinct task orders.
order 1: AGNews→De-En→Yelp→SQuAD.
order 2: De-En→Yelp →AGNews→SQuAD.
order 2: Yelp →SQuAD →De-En→AGNews.
order 4: SQuAD→AGNews→Yelp →De-En.