Effective Continual Learning for Text Classification with Lightweight Snapshots

Jue Wang[1,2*], Dajie Dong[1,2*], Lidan Shou[1,2†], Ke Chen[1,2], Gang Chen[1,2]

[1]Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University
[2]College of Computer Science and Technology, Zhejiang University
{zjuwangjue,dajiedong,should,chenk,cg}@zju.edu.cn

Abstract

Continual learning is known for suffering from catastrophic forgetting, a phenomenon where previously learned concepts are forgotten upon learning new tasks. A natural remedy is to use trained models for old tasks as 'teachers' to regularize the update of the current model to prevent such forgetting. However, this requires storing all past models, which is very space-consuming for large models, e.g. BERT, thus impractical in real-world applications. To tackle this issue, we propose to construct snapshots of seen tasks whose key knowledge is captured in lightweight adapters. During continual learning, we transfer knowledge from past snapshots to the current model through knowledge distillation, allowing the current model to review previously learned knowledge while learning new tasks. We also design representation recalibration to better handle the class-incremental setting. Experiments over various task sequences show that our approach effectively mitigates catastrophic forgetting and outperforms all baselines.

1 Introduction

Continual learning (CL) (Ring 1997; Thrun 1998) has become increasingly important in NLP. It aims to continually update the model and accumulate knowledge over a sequence of tasks. A fundamental problem in CL is catastrophic forgetting (McCloskey and Cohen 1989), i.e. knowledge acquired in previous tasks could be forgotten when the model is trained on new tasks. To overcome this problem, various methods have been proposed. Regularization-based methods constrain the model's updates to prevent it from changing too much from its previous state. However, as the number of tasks increases, the model will still gradually deviate from the state of the first few tasks. Exemplar-based methods selectively store seen examples from previous tasks and replay them while learning new tasks, but storing training data further brings data privacy concerns.

Therefore, how to preserve previously learned knowledge without storing training data becomes an important problem. This paper first investigates a straightforward solution that retains past models trained on previous tasks. We consider those models as *snapshots* of previous tasks, a compressed form of task-specific knowledge. So the current model can access old knowledge by learning from them via knowledge distillation (Hinton, Vinyals, and Dean 2015). However, storing all parameters of the past models is often unrealistic in practice due to the high storage cost. For instance, even the base version of BERT model (Devlin et al. 2019) contains 110 million parameters, which means 440 MB for each task. To tackle the storage challenge, we propose to construct snapshots with adapters (Rebuffi, Bilen, and Vedaldi 2018; Houlsby et al. 2019). The adapters make each snapshot only 0.8% the capacity of BERT while maintaining comparable accuracy, thus drastically reducing the space complexity. And since snapshots are no longer updated once created, the task-specific knowledge that they learned can also be preserved.

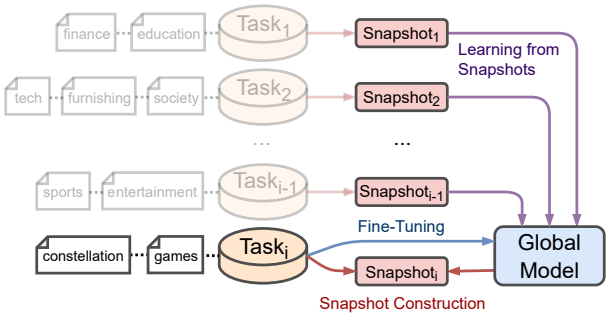


Figure 1: Overview of learning with snapshots. The global model learns from past snapshots when training new tasks.

This paper will focus on class-incremental learning (Class-IL) (Van de Ven and Tolias 2019), a challenging scenario where the task identity (indicating the candidate label set) is unknown at test time. Therefore, the model needs to make predictions on the complete label set for all tasks seen so far. In contrast, task-incremental learning (Task-IL) relaxes this constraint, and assumes that the task identity is given at test time, so the model can use its task-specific components to predict that task. Recent work finds neural networks are often less calibrated, and tend to yield overconfident predictions (Guo et al. 2017). As a result, it becomes difficult to determine the task identity by compar-

*These authors contributed equally.
†Corresponding Author.

ing the prediction scores between different tasks, as they are overconfident especially for recently-learned tasks (Masana et al. 2022). Existing countermeasures (Rebuffi et al. 2017; Castro et al. 2018) usually require storing old data samples to distinguish the task boundary, which however leads to privacy concerns. To address this issue, we design *representation recalibration* to regularize the representations of different tasks orthogonal to each other, so as to avoid activating irrelevant classifiers.

Our contributions can be summarized as follows:

- We propose to mitigate forgetting by learning from snapshots of previous tasks when learning new tasks;
- We propose to construct snapshots with lightweight adapters to capture the task-specific knowledge and reduce the space requirement to 0.8% of the original BERT;
- We design representation recalibration to distinguish the boundary between different tasks to better handle the Class-IL scenario without using any old training data.

We conduct extensive experiments over several task sequences, including cross-dataset and cross-language task sequences. The results show that our approach can effectively mitigate catastrophic forgetting under different task settings without using any old training data.[1]

2 Related Work

Continual Learning. CL aims to tackle the plasticity-stability dilemma, with plasticity to be the ability to learn new knowledge, and stability to retain old knowledge. Various solutions are proposed in the literature. Among them, *exemplar-based* methods and *regularization-based* methods have been widely applied to enable neural networks to continually acquire new knowledge without forgetting the previously learned knowledge. Exemplar-based methods use data samples from previous tasks (Rebuffi et al. 2017; de Masson D'Autume et al. 2019; Wang et al. 2020) or synthesized with generative models (Shin et al. 2017; Sun, Ho, and Lee 2019; Smith et al. 2021; Choi, El-Khamy, and Lee 2021), so the old knowledge can be replayed to overcome forgetting. Regularization-based methods alleviate forgetting by constraining the model's prediction (Li and Hoiem 2017), hidden states (Rannen et al. 2017; Huang et al. 2021), or parameters (Lopez-Paz and Ranzato 2017; Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017; Aljundi et al. 2018; Chaudhry et al. 2018) from varying too much. Cha et al. (2020) propose classifier-projection regularization that can be applied to regularization-based methods. Besides, Yu et al. (2020) estimate the semantic drift during training and compensate for it. Ke et al. (2021) investigate knowledge transfer across tasks in addition to overcoming catastrophic forgetting. Yin, Li, and Xiong (2022) propose to use textual instruction to guide continual learning. In this paper, we mainly compare with exemplar-free methods under Class-IL scenarios.

Model Fine-Tuning. Large pre-trained language models such as BERT (Devlin et al. 2019), ELMo (Peters et al. 2018), and XLNet (Yang et al. 2019) have been proposed to boost the performance of NLP tasks. These models are pre-trained on large corpora and then *fine-tuned* on task-specific datasets. While fine-tuning per-task is very effective, it also results in a distinct large model for each task, making it costly to store and use, especially when the number of tasks is large.

Recently, parameter-efficient fine-tuning of pre-trained language models (Houlsby et al. 2019) is being actively explored, which only fine-tunes a small subset of parameters while maintaining similar performance as fine-tuning all parameters. Such property is desirable for deploying multiple models in memory-constrained environments, since we only need to keep one base model and multiple sets of task-specific parameters. Among them, Adapter (Houlsby et al. 2019) injects compact task-specific adapter modules between the layers of the pre-trained model. Diff-Pruning (Guo, Rush, and Kim 2020) adds sparse difference-vectors to the original parameters. BitFit (Zaken, Ravfogel, and Goldberg 2021) only fine-tunes the bias term of the model. However, it is non-trivial to directly apply those methods in Class-IL scenarios, since the task identity is not given at test time and thus we do not know which set of task-specific parameters to use in inference.

3 Approach

In this work, we focus on continual learning on a sequence of tasks $\{\mathcal{T}_1, ..., \mathcal{T}_N\}$ arriving in temporal order, where each task $\mathcal{T}_i$ contains a different set of sentence-label pairs. When task $\mathcal{T}_i$ arrives, the data for previous tasks is *no longer* available due to storage constraints and privacy issues.

We instantiate a global model and continuously fine-tune it on new tasks. During training, we use past models of old tasks to regularize the updates of the global model. The intuition behind our approach is that, a model trained on a certain task can be considered as a 'knowledge base' containing necessary knowledge required for that task. Therefore, the global model can access the knowledge of previous tasks from past models via distillation.

Due to the large size of BERT, it is not feasible to store all parameters of past models. To tackle this issue, we propose to leverage lightweight adapters to reduce the number of task-specific parameters so as to reduce the storage cost. Once training for a task is complete, we freeze and store the adapters to build a snapshot of that task. We show that the storage space occupied by such a snapshot is two orders of magnitude smaller than that of the original BERT.

This section will focus on text classification, but we show in Section 4.8 that our approach can be easily adapted to other NLP tasks, e.g. sequence tagging tasks.

3.1 Fine-Tuning Global Model

We first demonstrate the standard procedure of fine-tuning the global model sequentially.

When a new task $\mathcal{T}_i$ arrives, we start from the global model of the previous task $\mathcal{T}_{i-1}$, and then optimize for the new task. Formally, given a sentence-label pair $(\boldsymbol{x}, y)$ sampled from $\mathcal{T}_i$, we aim to minimize the classification loss of

[1] Code available at: https://github.com/LorrinWWW/Snapshot.

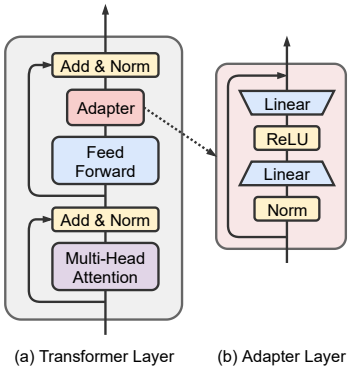


Figure 2: Space-efficient snapshots created with adapters, where we only store the parameters of adapters for each task.

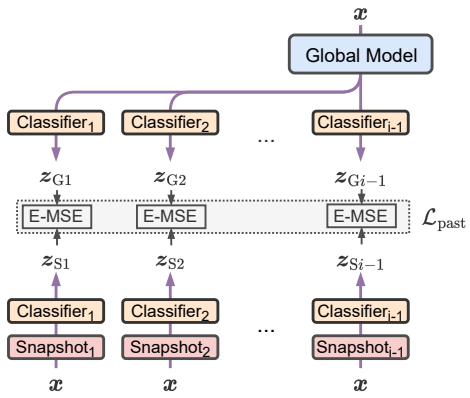


Figure 3: The procedure of learning from snapshots. Past snapshots and classifiers are fixed.

task $\mathcal{T}_i$ defined as follows:

$$\boldsymbol{h}_{\mathrm{G}} = \mathrm{Global}_{\theta}(\boldsymbol{x}), \tag{1}$$

$$\boldsymbol{z}_{\mathrm{G}i} = \boldsymbol{W}_i \boldsymbol{h}_{\mathrm{G}}, \tag{2}$$

$$\mathcal{L}_{\mathrm{cls}} = \mathrm{CE}(\mathrm{softmax}\,(\boldsymbol{z}_{\mathrm{G}i}), y), \tag{3}$$

where $\mathrm{Global}_{\theta}(\cdot)$ is the global model parameterized by θ; $\mathrm{CE}(\cdot)$ is the cross-entropy loss function; $\boldsymbol{W}_i$ is classification weight for task $\mathcal{T}_i$.

However, we may find that optimizing a single task alone tends to forget the knowledge from previous tasks, thus requiring effective methods to preserve old knowledge.

3.2 Creating Snapshots

To keep previously learned knowledge, we propose to use *adapters* to construct snapshots of previous tasks in a space-efficient way. Houlsby et al. (2019) propose to transfer BERT to down-stream tasks by inserting task-specific adapters. As shown in Fig. 2, adapters are two-layer feed-forward neural networks and typically adopt 'bottleneck' architectures. Each adapter contains a down-projection feed-forward neural network and an up-projection feedforward neural network with activation in between, e.g. ReLU, (Nair and Hinton 2010), followed by a skip connection:

$$\mathrm{A}(\boldsymbol{a}) = \mathrm{FFN}^{\uparrow}(\sigma(\mathrm{FFN}^{\downarrow}(\mathrm{Norm}(\boldsymbol{a})))) + \boldsymbol{a}, \tag{4}$$

where $\mathrm{Norm}(\cdot)$ is layer normalization; $\sigma(\cdot)$ is the activation function; $\mathrm{FFN}(\cdot)$ is a feedforward neural network defined as $\mathrm{FFN}(\boldsymbol{a}) = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{b}$.

For each task, we initialize adapters and insert them to the original BERT model. We keep BERT fixed and only update the parameters of adapters. The trained adapter model is regarded as a snapshot for the current task. We update the adapter parameters by jointly training with the global model. Specifically, for the same sentence-label pair $(\boldsymbol{x}, y)$ in Section 3.1, we define the following loss:

$$\boldsymbol{h}_{\mathrm{S}i} = \mathrm{Snapshot}_{\theta_i}(\boldsymbol{x}), \tag{5}$$

$$\boldsymbol{z}_{\mathrm{S}i} = \boldsymbol{W}_i \boldsymbol{h}_{\mathrm{S}i}, \tag{6}$$

$$\mathcal{L}_{\mathrm{snap}} = \mathrm{CE}(\mathrm{softmax}\,(\boldsymbol{z}_{\mathrm{S}i}), y) + \frac{1}{d}\|\boldsymbol{h}_{\mathrm{G}} - \boldsymbol{h}_{\mathrm{S}i}\|_2^2. \tag{7}$$

where $\mathrm{Snapshot}_{\theta_i}(\cdot)$ is the snapshot for task $\mathcal{T}_i$, parameterized by θ_i. The snapshot and the global model share the same classification weight $\boldsymbol{W}_i$. The loss term $\frac{1}{d}\|\boldsymbol{h}_{\mathrm{G}} - \boldsymbol{h}_{\mathrm{S}i}\|_2^2$ is introduced to encourage representations produced by the current snapshot and the global model to be close to each other. Once the training of current task $\mathcal{T}_i$ is completed, the snapshot and classification weight for task $\mathcal{T}_i$ will be *fixed* and saved. These saved snapshots are taken as a compressed form of previously learned knowledge, and are to be used to prevent the global model from forgetting.

3.3 Learning from Snapshots

To alleviate the forgetting of previous knowledge, we perform knowledge distillation between the global model and past snapshots. In particular, since we cannot access the data of previous tasks, we use past snapshots and classification weights to predict the data of the current task, and use the output as pseudo-labels to train the global model. For the sentence-label pair $(\boldsymbol{x}, y)$ sampled from the current task $\mathcal{T}_i$, we define the distillation loss as follows:

$$\boldsymbol{h}_{\mathrm{G}} = \mathrm{Global}_{\theta}(\boldsymbol{x}), \quad \boldsymbol{z}_{\mathrm{G}j} = \boldsymbol{W}_j \boldsymbol{h}_{\mathrm{G}}, \tag{8}$$

$$\boldsymbol{h}_{\mathrm{S}j} = \mathrm{Snapshot}_{\theta_j}(\boldsymbol{x}), \quad \boldsymbol{z}_{\mathrm{S}j} = \boldsymbol{W}_j \boldsymbol{h}_{\mathrm{S}j}, \tag{9}$$

$$\mathcal{L}_{\mathrm{past}} = \sum_{j=1}^{i-1} \frac{1}{C_j} \|e^{\boldsymbol{z}_{\mathrm{G}j}/T} - e^{\boldsymbol{z}_{\mathrm{S}j}/T}\|_2^2, \tag{10}$$

where T is a temperature parameter to adjust the magnitude of prediction scores; C_j is the number of classes for task $\mathcal{T}_j$.

We use an exponential function to further activate the predictions and pseudo-labels to make the model focus on the large prediction value, which we denote as E-MSE in Fig. 3. Equation (10) will reach its minimum when $\boldsymbol{z}_{\mathrm{G}j} = \boldsymbol{z}_{\mathrm{S}j}$. It should be noted that KL-divergence, adopted by many distillation methods, might not be suitable here as it removes magnitude and bias information and further leads to a lack of calibration between tasks. We empirically evaluate vanilla MSE loss and KL-divergence in Section 4.6, but the performance degrades compared with E-MSE.

To avoid too much training overhead incurred by the growing number of tasks, we set the maximum number of

Algorithm 1: Overall training procedure.

Require: task sequence $[\mathcal{T}_i]_{i=1,2,...,N}$; learning rate η; pre-trained parameters θ for the global model.

initialize $\text{Global}_\theta(\cdot)$
for each task $\mathcal{T}_i, i = 1, 2, ..., N$ **do**
  initialize $\text{Snapshot}_{\theta_i}(\cdot)$ and $\boldsymbol{W}_i$
  for $(\boldsymbol{x}, y) \sim \mathcal{T}_i$ **do**
    // forward pass
    $\boldsymbol{h}_\text{G} \leftarrow \text{Global}_\theta(\boldsymbol{x})$
    $\boldsymbol{h}_{\text{S}j} \leftarrow \text{Snapshot}_{\theta_j}(\boldsymbol{x}), j = 1, ..., i$
    $\boldsymbol{z}_{\text{G}j} \leftarrow \boldsymbol{W}_j \boldsymbol{h}_\text{G}, j = 1, ..., i$
    $\boldsymbol{z}_{\text{S}j} \leftarrow \boldsymbol{W}_j \boldsymbol{h}_{\text{S}j}, j = 1, ..., i$
    $\mathcal{L} \leftarrow \mathcal{L}_\text{cls} + \mathcal{L}_\text{snap} + \mathcal{L}_\text{past} + \mathcal{L}_\text{orth}$
    // update parameters
    $\theta \leftarrow \theta - \eta \partial \mathcal{L} / \partial \theta$
    $\theta_i \leftarrow \theta_i - \eta \partial \mathcal{L} / \partial \theta_i$
    $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \partial \mathcal{L} / \partial \boldsymbol{W}_i$
    $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i / \left\| \boldsymbol{W}_i \right\|_2$
  end for
end for
return $\text{Global}_\theta(\cdot)$ and $[\boldsymbol{W}_i]_{i=1,...,N}$

snapshots M. When $i > M$, we randomly sample M snapshots to compute Eq. (10) for each training step.

3.4 Representation Recalibration

Although learning from snapshots effectively mitigates the forgetting issue, the performance is still unsatisfactory in Class-IL, where the model needs to predict all classes seen in previous tasks without knowing the task identity at inference time. Ideally, we would like the model to produce high-confidence predictions using the correct classification weight, and to be less confident otherwise. However, we often get overconfident predictions from all classification weights, thus causing task boundaries to be ambiguous.

An important observation is that the learned representation spaces for different tasks usually overlap each other. Therefore, an input that activates a relevant classification weight may also activate others. To ensure the representations of different tasks to be distinguishable by clear task boundaries, we construct a regularization term defined as follows:

$$\mathcal{L}_\text{orth} = \sum_{j=1}^{i-1} \left\| \boldsymbol{W}_i \boldsymbol{W}_j^\top \right\|_{1,1}. \tag{11}$$

Moreover, since the classification weights of different tasks are learned individually, they are not properly calibrated and thus the prediction scores are not comparable. To this end, we normalize the classification weight after each training iteration, such that $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i / \left\| \boldsymbol{W}_i \right\|_2$.

3.5 Joint Training

Finally, we sum up all the loss terms and use gradient-based optimization methods to minimize the following loss:

$$\mathcal{L} = \mathcal{L}_\text{cls} + \mathcal{L}_\text{snap} + \mathcal{L}_\text{past} + \mathcal{L}_\text{orth}. \tag{12}$$

Note that for each training iteration, we only update the global model, the current snapshot and classification weight, and keep the previous snapshots and classification weights unchanged. The overall algorithm is shown at Algorithm 1.

Inference. Our approach does not incur additional computation overhead in inference compared with multi-task learning. In Class-IL, we use the global model to obtain the input's representation, predict with all classification weights, and then pick the label with the largest prediction score. Formally, given input $\boldsymbol{x}$, we have:

$$\boldsymbol{h}_\text{G} = \text{Global}_\theta(\boldsymbol{x}), \tag{13}$$

$$\boldsymbol{z}_i = \boldsymbol{W}_i \boldsymbol{h}_\text{G}, i = 1, ..., N \tag{14}$$

$$\hat{y} = \text{argmax}\left([\boldsymbol{z}_1; ...; \boldsymbol{z}_N]\right). \tag{15}$$

Alternatively, in Task-IL, where task identity is given at test time, e.g. $\mathcal{T}_k$, we use its corresponding classification weight to predict $\hat{y} = \text{argmax}\left(\boldsymbol{z}_k\right)$.

4 Experiments

We demonstrate that our approach can effectively mitigate catastrophic forgetting. Specifically, (1) On cross-dataset and cross-lingual task sequences, we show our method can maintain the best overall accuracy, whereas baseline methods often suffer from accuracy drop for previous tasks. (2) By investigating the trade-off between space and accuracy. we show that our method as a lightweight and storage-efficient scheme for handling catastrophic forgetting. (3) We perform ablation study to understand the effectiveness of components and strategies in our approach. (4) We visualize the encoding space during training, which shows that our approach can effectively distinguish the task boundary.

4.1 Data

We compare our approach and other baseline models on the following datasets: THUCNews dataset (Sun et al. 2016), AG's news corpus (Zhang, Zhao, and LeCun 2015), Yelp reviews (Asghar 2016), Amazon reviews (McAuley and Leskovec 2013), and DBPedia dataset (Zhang, Zhao, and LeCun 2015). To balance the number of data samples between these tasks, we prune the datasets to keep them in the same order of magnitude. We merge the label space of Amazon and Yelp considering their similarity.

4.2 Setup

We fine-tune the multilingual cased checkpoint (bert-base-multilingual-cased) for all task settings. We use the Adam optimizer, and set the learning rate to 2e-5 for the global model and 1e-4 for the adapter-based snapshot. We set the training batch size to 32. We tune the hyperparameters by performing a grid search over $d \in [12, 96]$, $T \in [1, 6]$, and $M \in [1, 4]$. Our approach works well with a bottleneck size $d = 48$, temperature $T = 3$, and maximum number of snapshots for each training step $M = 3$. We train each task for one epoch and report the average results over 3 runs with different random seeds.

We measure the average accuracy of all seen tasks. For the current time step i, we compute $\overline{\text{Acc}}_i = \frac{1}{i} \sum_{j=1}^{i} \text{Acc}_{i,j}$, where $\text{Acc}_{i,j}$ is the accuracy of task $\mathcal{T}_j$ after training task $\mathcal{T}_i$.

THU-Seq:	THU_1	$\rightarrow THU_2$	$\rightarrow THU_3$	$\rightarrow THU_4$
MTL	99.0	95.3	95.9	95.7
FT	98.9	61.2	37.1	27.4
LwF	98.9	76.8	53.0	43.8
EWC	98.9	82.5	54.6	51.8
R-Walk	98.9	84.9	81.2	73.5
MAS	98.9	86.4	82.3	60.1
AdapterParallel*	98.7	72.0	71.6	69.4
AdapterStack	98.7	55.6	50.7	33.3
AdapterFusion	98.7	52.7	36.2	26.5
Snapshot	$98.9_{\pm 0.1}$	$\mathbf{87.9}_{\pm 0.2}$	$\mathbf{89.2}_{\pm 0.4}$	$\mathbf{86.3}_{\pm 0.4}$

Table 1: Results on THU-Seq, a task sequence constructed from THUCNews. *Inference overhead is proportional to the number of tasks. Bold numbers indicate the best scores except for MTL.

Mix-Seq:	THU_1	$\rightarrow$ AG	$\rightarrow$ Yelp	$\rightarrow$ Amz.	$\rightarrow$ DBP
MTL	98.9	95.2	88.6	80.8	84.5
FT	98.9	47.7	17.9	23.3	19.6
LwF	98.6	80.0	46.2	34.9	21.5
EWC	98.5	**94.7**	50.4	48.2	60.4
R-Walk	98.6	82.6	68.7	37.6	23.1
MAS	98.9	93.3	67.4	62.8	60.2
AdapterParallel*	98.4	83.2	63.3	57.9	38.4
AdapterStack	98.7	77.3	47.4	29.0	19.8
AdapterFusion	98.7	45.8	25.8	30.1	19.8
Snapshot	$99.0_{\pm 0.1}$	$92.0_{\pm 0.4}$	$\mathbf{69.6}_{\pm 1.2}$	$\mathbf{65.9}_{\pm 1.5}$	$\mathbf{64.1}_{\pm 0.9}$

Table 2: Results on Mix-Seq, a task sequence from THU_1, AG, Yelp, Amazon and DBP. *Inference overhead is proportional to the number of tasks. Bold numbers indicate the best scores except for MTL.

4.3 Baselines

We compare our proposed approach with the following baselines in our experiments:

Fine-Tune (FT): Fine-tune the BERT model sequentially, which is usually considered as a *lower bound* for CL.

Multi-Task Learning (MTL): Train all tasks simultaneously. This can be regarded as an *upper bound* for CL since it can access data from all tasks at the same time.

LwF: Li and Hoiem (2017) use the old model trained on the previous task to predict new task data, and then perform distillation to prevent forgetting. It requires storing the predictions from the old model on new data.

EWC: Kirkpatrick et al. (2017) avoid forgetting old tasks by selectively slowing down learning on the parameters important for those tasks. It maintains "estimated mean" and "estimated fisher" for each parameter.

R-Walk: Chaudhry et al. (2018) propose a generalization of EWC++ and path integral with a theoretically grounded KL-divergence based perspective. This approach needs to store the fisher matrix and the score of each parameter.

MAS: Aljundi et al. (2018) accumulate an importance measure for each parameter based on how sensitive the output is to a change in this parameter. It stores the importance of each parameter.

AdapterParallel: Train each task with an individual set of adapters (Houlsby et al. 2019). At test time, we use *all* models to predict and pick the label with the highest prediction score. Note this baseline is not efficient in practice as its inference overhead grows proportional to the number of tasks.

AdapterStack: Incrementally stack new adapters as new tasks arrive (Pfeiffer et al. 2020). We keep previous adapters fixed and only fine-tune the latest adapter. We add representation recalibration when evaluating this approach.

AdapterFusion: Pfeiffer et al. (2021) leverage existing adapters by learning an adapter fusion module. We add representation recalibration when evaluating this approach.

4.4 Results and Comparison

We first construct a task sequence from THUCNews, denoted as THU-Seq. Specifically, we split THUCNews into 4 non-overlapping subsets, each containing 3 text categories, denoted as $THU_{1,2,3,4}$. Table 1 summarizes the results of different approaches on THU-Seq. The column i shows the average accuracy of seen tasks after training task $\mathcal{T}_i$, and the last column shows the overall accuracy when the training is completed for all tasks. While the naive fine-tuning suffers from severe forgetting problems, LwF, EWC, MAS, and R-Walk show consistent improvements on all tasks, however, their performance is still unsatisfactory. Besides, while Adapter is effective for each individual task, it is not suitable for Class-IL where the task identity is unknown at test time. Even if we use all trained adapter-based models to make predictions (AdapterParallel), which would incur huge computational overhead, the performance is still unsatisfactory since their outputs are not well calibrated. AdapterStack and AdapterFusion will expand parameters upon new tasks arrive, but they still suffer from catastrophic forgetting in the Class-IL scenario. Results of these Adapter-based methods show that the direct application of Adapter *cannot* solve the problem of catastrophic forgetting in continuous learning.

We also construct a challenging task sequence from a mixture of different datasets, including THU_1, AG, Yelp, Amazon and DBP, denoted as Mix-Seq. These datasets come from different data sources and contain multiple languages (English and Chinese). As shown in Table 2, since the data distribution varies significantly across different tasks, catastrophic forgetting is more serious than THU-Seq. We can observe that the performance of all baselines drops significantly during training. However, our approach maintains stable performance, showing that our approach is robust to significantly different data distributions and even cross-lingual scenarios.

Additional Storage. The baselines in this paper all require to store additional information. Our method stores snapshots for each task and the original BERT. However, since a snapshot only accounts for 0.8% (depending on the hyperparameters, studied in Section 4.5) storage of the

Task Sequences	$\mathcal{T}_1$	$\rightarrow$ $\mathcal{T}_2$	$\rightarrow$ $\mathcal{T}_3$	$\rightarrow$ $\mathcal{T}_4$
$\text{THU}_{1\rightarrow2\rightarrow3\rightarrow4}$	98.9	87.9	89.2	86.3
$\text{THU}_{2\rightarrow1\rightarrow3\rightarrow4}$	98.4	81.4	82.3	76.0
$\text{THU}_{3\rightarrow2\rightarrow1\rightarrow4}$	99.4	95.9	89.9	86.2
$\text{THU}_{3\rightarrow2\rightarrow4\rightarrow1}$	99.4	94.8	88.9	82.1

Table 3: Results with permutations of the task order.

Setting	# Param.	% of BERT	THU-Seq	Mix-Seq
$d = 96$	1.8M	1.6%	85.6	63.0
$d = 48$	0.89M	0.8%	**86.3**	**64.1**
$d = 24$	0.45M	0.4%	81.7	62.3
$d = 12$	0.23M	0.2%	77.6	58.6

Table 4: Impact of bottleneck sizes of snapshots. "# Param." is the number of parameters for each snapshot. "% of BERT" is the ratio of the snapshot to BERT.

global model, our method incurs less storage than EWC and R-walk unless we have 125+ tasks. In typical settings of CL, this number implies our method as a storage-efficient scheme for handling forgetting.

Task Sequence Length. We observe all approaches are prone to suffer from forgetting as the task sequence becomes longer. But our approach still maintains the best result, and the performance margins between our approach and other baselines tend to widen as the number of tasks increases. This indicates the effectiveness of our approach in mitigating forgetting especially for longer task sequences.

Task Sequence Order. We perform experiments with 4 permutations of the task order. Table 3 shows that the performance varies with different task orders, indicating the task order is important to the overall accuracy. Particularly, the performance always drops after training for THU_1 and THU_2. We find some labels in the two tasks to be similar (fashion and entertainment) and thus ambiguous. Therefore, if the two tasks appear one after another, the model is likely to be confused and forget the earlier one. However, even for the task sequence with the worst performance, i.e. $\text{THU}_{2\rightarrow1\rightarrow3\rightarrow4}$, our approach still achieves better average accuracy than other baselines in the default task sequence, which shows the robustness of our approach to different permutations of the task order.

4.5 Trade-off Between Space and Accuracy

To better understand the proposed approach, we conduct a series of experiments with different bottleneck sizes d of adapters. The results are shown in Table 4. Since the number of parameters of adapters depends on the bottleneck size, enlarging bottleneck size allows for a larger capacity of adapters and thus may improve accuracy, but it also leads to greater storage requirements to preserve snapshots. When d is larger than 48, the overall accuracy drops slightly, mainly because larger d tends to overfit new tasks, resulting in lower accuracy of the global model on old tasks.

Setting	THU_1	$\rightarrow\text{THU}_2$	$\rightarrow\text{THU}_3$	$\rightarrow\text{THU}_4$
w/o $\mathcal{L}_{\text{past}}$	98.9	80.2	62.0	38.3
MSE	98.9	87.2	87.6	83.7
KL-Div	98.9	85.9	81.1	76.5
E-MSE	98.9	**87.9**	**89.2**	**86.3**

Table 5: Results of different distillation loss functions.

Setting	$\text{THU}_1 \rightarrow$	THU_2	$\rightarrow$ THU_3	$\rightarrow$ THU_4	$\mathcal{L}$
$T = 1$	98.9 / -	23.9 / **98.5**	21.5 / 79.8	8.5 / 75.9	6.1
$T = 2$	98.9 / -	89.7 / 82.4	91.3 / 85.6	49.8 / **87.4**	22.8
$T = 3$	98.9 / -	89.0 / 86.9	94.7 / 86.6	85.5 / 86.6	19.0
$T = 4$	98.9 / -	83.9 / 80.0	94.5 / **87.1**	87.1 / 85.4	22.4
$T = 6$	98.9 / -	**91.2** / 84.1	**95.6** / 82.0	**90.6** / 82.3	13.0

Table 6: Effect of the temperature parameter in learning from past snapshots. We present the plasticity-stability trade-offs by comparing the accuracy of the new task $\text{Acc}_{i,i}$ and the average accuracy of old tasks $\frac{1}{i-1}\sum_{j=1}^{i-1}\text{Acc}_{i,j}$.

4.6 Ablation Study

Effect of Learning from Past Snapshots. We compare different distillation loss functions in Table 5. Firstly, when we remove $\mathcal{L}_{\text{past}}$, i.e. the global model will not learn anything from past snapshots, the accuracy of old tasks drops significantly, which shows that $\mathcal{L}_{\text{past}}$ is important in mitigating forgetting. Besides, replacing E-MSE with KL-divergence also hurts the performance. KL-divergence requires performing softmax on the prediction scores within a task, which will discard the information of magnitude and average value, thereby leading to the issue of lack of calibration. E-MSE outperforms MSE because exponential activation allows the model to focus more on significant values.

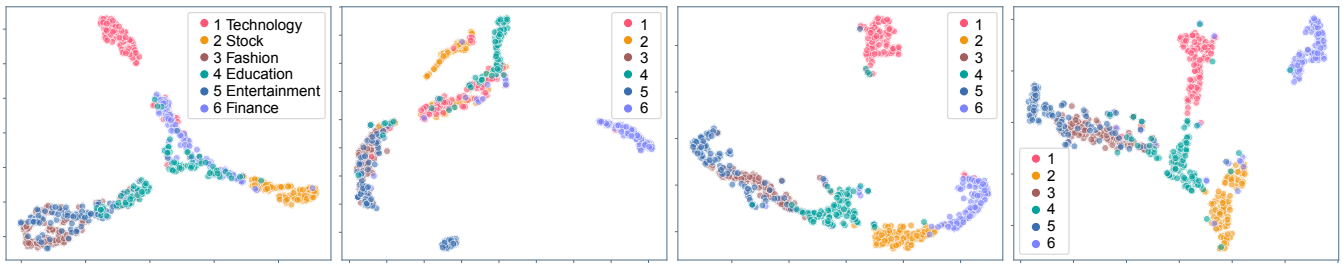
Furthermore, we investigate the effect of the temperature parameter T. We also include forgiveness rate defined by Liu et al. (2020). Table 6 presents the results. Reducing T helps maintain the accuracy of old tasks. However, when T is too small, the global model will struggle in minimizing $\mathcal{L}_{\text{past}}$, and thus it becomes hard for the global model to learn new concepts. Generally, $T = 3$ can achieve a good balance of the plasticity-stability.

Effect of Representation Recalibration. In order to demonstrate the effect of representation recalibration, especially for the Class-IL scenario, we conduct experiments by removing the regularization term $\mathcal{L}_{\text{orth}}$ and removing the weight normalization operation. Table 7 shows that both contribute to the final performance. But the performance gain is more significant when combing them together.

In the task-incremental scenario, where the task identity of the input sample is provided at test time, the effect of representation recalibration is less significant since the model does not have to distinguish the task boundary by itself.

4.7 Visualization

To study the phenomenon that the learned representation spaces for different tasks usually overlap with each other in



(a) FT after $\mathcal{T}_1$ (b) FT after $\mathcal{T}_2$ (c) Snapshot w/o recal. after $\mathcal{T}_2$ (d) Snapshot w/ recal. after $\mathcal{T}_2$

Figure 4: t-SNE visualization of representation of different classes from task $\mathcal{T}_1$ and $\mathcal{T}_2$. $\mathcal{T}_1$ contains class 1, 2, 3, denoted in warm colors; $\mathcal{T}_2$ contains class 4, 5, 6, denoted in cool colors.

Setting	$\text{THU}_1 \rightarrow$	$\text{THU}_2 \rightarrow$	$\text{THU}_3 \rightarrow$	THU_4
Class-IL	98.9	87.9	**89.2**	**86.3**
w/o $\mathcal{L}_{\text{orth}}$	99.0	**88.6**	87.9	79.6
w/o cls. norm.	99.0	87.2	79.7	77.7
w/o both	99.0	86.3	80.7	75.8
Task-IL	98.9	98.3	**98.6**	**98.6**
w/o $\mathcal{L}_{\text{orth}}$	99.0	98.5	98.5	**98.6**
w/o cls. norm.	99.0	**98.6**	98.5	98.5
w/o both	99.0	98.2	98.3	98.5

Table 7: Effect of representation recalibration. "w/o cls. norm." does not normalize the classification weights.

CL, as discussed in Section 3.4, we use t-SNE to visualize the representation produced by the global model. Specifically, we select 6 text classes from THUCNews to construct two tasks, each containing 3 classes, and visualize the representation distribution after learning the first and the second task. We compare our approach and FT that does not employ any mechanism to overcome forgetting.

Figure 4(a) presents the results after learning the first task, and we can see that the representations for the first task are distributed separately into 3 clusters, showing that the model can successfully distinguish the boundary between inter-task classes of the first task. Since the model has not met any data of the second task, it tends to encode these data in the same representation space of the first task. After learning the second task, as shown in Fig. 4(b), the model can now distinguish the class boundary of the second task, but begins to confuse the first 3 classes as they are not available for training now. By learning from past snapshots, as shown in Fig. 4(c), the model prevents forgetting the old knowledge so the representations of the first task are still distributed separately with clear margins. However, the representations of the two tasks sometimes overlap with each other, e.g. representations of class 2 and 6, which means that a representation in the overlapping area may activate both classification weights for the two tasks, and causes ambiguity in inferring the task identity at test time. In comparison, when using representation recalibration, as shown Fig. 4 (d), the model can clearly distinguish the class boundary for most tasks.

SNIPS	$\mathcal{T}_1$	$\rightarrow \mathcal{T}_2$	$\rightarrow \mathcal{T}_3$	$\rightarrow \mathcal{T}_4$	$\rightarrow \mathcal{T}_5$	$\rightarrow \mathcal{T}_6$	$\rightarrow \mathcal{T}_7$
FT	99.0	86.3	51.7	56.0	32.0	22.9	19.8
Snapshot	99.0	97.4	98.0	96.8	92.1	85.8	79.4

Table 8: Adaptation to slot filling. We calculate the average F1 score for the tasks seen so far.

4.8 Adaptation to Sequence Tagging Tasks

Our approach can be easily adapted to other task forms. We here take slot filling as an example. It is usually interpreted as a sequence tagging process, during which slot values and their corresponding slot types are annotated. We evaluate our approach on the SNIPS benchmark (Coucke et al. 2018), and construct a sequence of seven tasks based on the user intents: AddToPlaylist $\rightarrow$ BookRestaurant $\rightarrow$ GetWeather $\rightarrow$ PlayMusic $\rightarrow$ RateBook $\rightarrow$ SearchCreativeWork $\rightarrow$ SearchScreeningEvent. We use averaged F1 score of seen tasks as the evaluation metric. The results are shown in Table 8. Specifically, the performance of naive FT degrades significantly as the number of tasks increases, especially for the previous tasks. Actually, when the third task arrives, its overall F1 already drops to 51.7%, which is basically unusable in practice. And the overall F1 becomes less than 20% after training all tasks. In comparison, our approach can maintain acceptable performance, which indicates the effectiveness of our approach in mitigating forgetting for slot filling.

5 Conclusion

This paper presents a novel continual learning approach, which aims to prevent catastrophic forgetting by taking advantage of space-efficient snapshots. We construct the snapshot of seen tasks with adapters. During continual learning, we perform knowledge distillation between the past snapshots and the global model, so the global model can review past knowledge while learning new tasks. To support the Class-IL scenario, we also design representation recalibration. We perform extensive experiments on different task settings, including cross-dataset and cross-language task sequences, which shows the effectiveness and robustness of our approach. In the future, we would like to design effective training methods if it is allowed to replay a small portion of old data from previous tasks.

Acknowledgements

This research was supported by the National Key Research and Development Program of China (No. 2022YFB3304100), and Fundamental Research Funds for the Central Universities.

References

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *ECCV*.

Asghar, N. 2016. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*.

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C.; and Alahari, K. 2018. End-to-end incremental learning. In *ECCV*.

Cha, S.; Hsu, H.; Hwang, T.; Calmon, F.; and Moon, T. 2020. CPR: Classifier-Projection Regularization for Continual Learning. In *ICLR*.

Chaudhry, A.; Dokania, P. K.; Ajanthan, T.; and Torr, P. H. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*.

Choi, Y.; El-Khamy, M.; and Lee, J. 2021. Dual-teacher class-incremental learning with data-free generative replay. In *CVPR*.

Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; Lavril, T.; et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

de Masson D'Autume, C.; Ruder, S.; Kong, L.; and Yogatama, D. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *ICML*.

Guo, D.; Rush, A. M.; and Kim, Y. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *ICML*.

Huang, Y.; Zhang, Y.; Chen, J.; Wang, X.; and Yang, D. 2021. Continual Learning for Text Classification with Information Disentanglement Based Regularization. In *NAACL-HLT*.

Ke, Z.; Liu, B.; Ma, N.; Xu, H.; and Shu, L. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34: 22443–22456.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2935–2947.

Liu, Y.; Su, Y.; Liu, A.-A.; Schiele, B.; and Sun, Q. 2020. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A. D.; and van de Weijer, J. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

McAuley, J.; and Leskovec, J. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*. Elsevier.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*.

Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; and Gurevych, I. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *EACL*.

Pfeiffer, J.; Vulić, I.; Gurevych, I.; and Ruder, S. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *EMNLP*.

Rannen, A.; Aljundi, R.; Blaschko, M. B.; and Tuytelaars, T. 2017. Encoder based lifelong learning. In *ICCV*.

Rebuffi, S.-A.; Bilen, H.; and Vedaldi, A. 2018. Efficient parametrization of multi-domain deep neural networks. In *CVPR*.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *CVPR*.

Ring, M. B. 1997. CHILD: A first step towards continual learning. *Machine Learning*, 28(1): 77–104.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Smith, J.; Hsu, Y.-C.; Balloch, J.; Shen, Y.; Jin, H.; and Kira, Z. 2021. Always be dreaming: A new approach for data-free class-incremental learning. In *CVPR*.

Sun, F.-K.; Ho, C.-H.; and Lee, H.-Y. 2019. LAMOL: LAnguage MOdeling for Lifelong Language Learning. In *ICLR*.

Sun, M.; Li, J.; Guo, Z.; Yu, Z.; Zheng, Y.; Si, X.; and Liu, Z. 2016. Thuctc: an efficient chinese text classifier. *GitHub Repository*.

Thrun, S. 1998. Lifelong Learning Algorithms. *Learning to learn*, 8: 181–209.

Van de Ven, G. M.; and Tolias, A. S. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Wang, Z.; Mehta, S. V.; Póczos, B.; and Carbonell, J. G. 2020. Efficient Meta Lifelong-Learning with Limited Memory. In *EMNLP*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yin, W.; Li, J.; and Xiong, C. 2022. ConTinTin: Continual Learning from Task Instructions. In *ACL*.

Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S.; and Weijer, J. v. d. 2020. Semantic drift compensation for class-incremental learning. In *CVPR*.

Zaken, E. B.; Ravfogel, S.; and Goldberg, Y. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *ICML*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.