Dual Contrastive Learning Framework for Incremental Text Classification

Yigong Wang, Zhuoyi Wang, Yu Lin, Jinghui Guo, Sadaf MD Halim, Latifur Khan
The University of Texas at Dallas
{yxw158830, zhuoyi.wang1, yxl163430, jinghui.guo,
sadafmd.halim, lkhan}@utdallas.edu

Abstract

Incremental learning plays a pivotal role in the context of online knowledge discovery, as it encourages large models (LM) to learn and refresh knowledge continuously. Many approaches have been proposed to simultaneously preserve knowledge from previous tasks while learning new concepts in online NLP applications. In this paper, we primarily focus on learning a more generalized embedding space that could be better transferred to various downstream sequence tasks. The key idea is to learn from both task-agnostic and task-specific embedding aspects so that the inherent challenge of catastrophic forgetting that arises in incremental learning scenarios can be addressed with a more generalized solution. We propose a dual contrastive learning (DCL) based framework to foster the transferability of representations across different tasks, it consists of two key components: firstly, we utilize global contrastive learning that intertwines a task-agnostic strategy for promoting a generalized embedding space; secondly, considering the domain shift from unseen distributions can compromise the quality of learned embeddings. We further incorporate a task-specific attention mechanism to enhance the adaptability of task-specific weight for various emerging tasks and ultimately reduce errors in generic representations. Experiments over various text datasets demonstrate that our work achieves superior performance and outperforms the current state-of-the-art methods.

1 Introduction

Fine-tuning a pre-trained large model (Devlin et al., 2019; Dosovitskiy et al., 2021) on extensive datasets has proven highly effective in addressing a wide array of downstream NLP tasks. Despite the significant development of such models in various real-world applications, a prominent challenge remains: adapting trained models to new tasks or datasets while retaining knowledge from prior ones. In large-scale content mining scenarios where the landscape continually evolves, our objective is to update the model perpetually from non-stationary streams of the source data (e.g. the online news summary/search (Geng et al., 2021), catalog from e-commerce (Yang et al., 2019; Wang et al., 2021b)), so that the model parameters/components will keep refreshing with new data continuously involved, which is called incremental, or continual learning. This dynamic adaptation is essential to the web applications that process the constant influx of new tasks, for example, improve the performance of online model tuning for State tracking system (Zhu et al., 2022), sentiment analysis of lifelong user intent in the conversation dialogue (Madotto et al., 2021; Liu and Mazumder, 2021; Geng et al., 2021; Chi et al., 2023).

When we typically fine-tune the pre-trained large language model on the downstream tasks, the model tends to adjust its learned parameters according to the new knowledge or tasks, inadvertently causing the loss of previously acquired concepts, which is called "catastrophic forgetting" (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017). Recent advancements have introduced various strategies to address this challenge, which enable the continual tuning from emerging data while minimizing the rate of forgetting (Aljundi et al., 2018; de Masson D'Autume et al., 2019; Han et al., 2020; Sun et al., 2020; Huang et al., 2021). In continual text mining, these approaches can be broadly categorized into two categories. The first is the experience-replay-based methods((Sun et al., 2020; Huang et al., 2021)), where the data from old tasks are stored or generated to be accessed when the model is training on new tasks. Another type of solution is the regularized-based methods (Aljundi et al., 2018; Han et al., 2020; Wang et al., 2021a), in which constraints are imposed on model parameters or data embeddings to prevent excessive drift while training on new tasks.

Findings of the Association for Computational Linguistics: EMNLP 2023, pages 194–206
December 6-10, 2023

To illustrate, consider a typical incremental learning scenario: suppose we train a model to understand the sentiment of product reviews on an e-commerce website. Later, we wish to analyze the sentiment of movie reviews. We need to classify the topic for a given news article even later. In such situations, having a single model capable of handling all these tasks is preferred over maintaining separate models for each task.

In this work, different from existing solutions that focus on preserving old knowledge from new ones, we draw attention to the fundamental questions: *How to learn a generalizable/transferable global feature space that could achieve task-agnosticism for future tasks? If having newly emerged unknown tasks with domain shift, how to select task-specific features to mitigate the dilution of essential information within the universal embedding?*

Building upon insights from recent works (Prabhu et al., 2020), part of the features can hold significant relevance for the current task (termed "task-specific"), yet maybe less significant to the following ones. Hence, relying solely on learning task-specific features at each stage leads to continual overwriting of the universal embedding space, which optimizes the current task but is sub-optimal for future tasks. On the other hand, if we only depend on task-agnostic embedding spaces for different data distributions from various tasks, the model could not adequately handle the data distribution of unknown, drifted tasks, potentially causing the model performance regression. Therefore, it is amenable to refactor the process into a comprehensive embedding with a task-specific feature selection mechanism, which lifts appropriate features across the universal, transferable embedding space, and maintains discrimination for different emerged tasks on each step.

Taking inspiration from self-supervised learning (SSL) (Chen et al., 2020; Gao et al., 2021; Sang et al., 2022), which can learn transferable representations for down-stream tasks, we introduce a dual contrastive learning framework that aims to learn both task-agnostic and task-specific representation in the generic embedding space continually. This framework comprises two integral components: the global contrastive learning (GCL) module and the task-specific contrastive attention (TCA) module: GCL focuses on acquiring transferable embedding space as task-agnostic, with additional TCA as a feature re-weighting strategy to enhance different components in the generic representation when the model learns from the newly emerged tasks. Specifically, the GCL module leverages the benefits of contrastive learning to discover task-agnostic knowledge from inputs (instead of the task-specific supervisions), so that the learned feature embedding could suffer less from the knowledge forgetting. The TCA part applies the attention mechanism (Kim et al., 2017) as a promising feature selection/re-weighting strategy for various task-specific components. It is used to assign a suitable weight vector for each feature according to its importance in the universal embedding space. Therefore, the model can concentrate on critical components for the task at hand, and also reduce the drift on the major generic representations we have acquired.

In conclusion, the main contributions of our work could be summarized as:

- We propose an effective contrastive learning-based method that focuses on learning more generalizable and transferable features sequentially;
- We emphasize the task-specific contrastive-based attention mechanism to better learn and distinguish the task-specific knowledge in the universal embedding space;
- Extensive experiments conducted on the text classification datasets demonstrate the effectiveness of our approach, which could achieve better performance than the existing SOTA.

2 Background

2.1 Incremental Learning

Incremental learning aims to learn knowledge from a sequence of new tasks, which is different from multi-task learning which could observe all the tasks during the training time. Under this scenario that only trained on one task at each step (Li and Hoiem, 2017; Aljundi et al., 2018; Wang et al., 2021b; Wu et al., 2021), the models are required to learn new information while retaining previous skills or knowledge (from old tasks). One conventional approach is to instantiate a model trained on the old dataset and then fine-tune the model on the new data (Girshick et al., 2014). However, this method may suffer from decayed performance on the old data, which is called catastrophic forgetting (Li and Hoiem, 2017). Recent

interests in continual learning have resulted in many novel methods for continual learning. In general, they can be grouped into two categories: regularization-based and replay-based approaches. Regularization-based methods would protect significant parameters or data representations and add more penalties to the changes on them as regularization, like CIFDM (Wang et al., 2021a), EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017) and MAS (Aljundi et al., 2018). On the other hand, replay-based methods typically generate/store old samples from previous classes and then apply them to training the new tasks. This includes LAMOL (Sun et al., 2020), IDBR (Huang et al., 2021) and MPBA++ (de Masson D'Autume et al., 2019). Except for these two major categories, there are some sub-category methods, including representation alignment (Wang et al., 2019, 2020) and knowledge distillation (Hinton et al., 2015; Rebuffi et al., 2017). In this paper, we focus on learning transferable and generalized feature embedding with corresponding feature re-weighting, which maintains a more suitable universal task space and utilizes task-specific knowledge simultaneously.

2.2 Contrastive Learning

Contrastive learning is a special case of self-supervised learning, which typically learns with self- supervision (Jing and Tian, 2020; Chen et al., 2020; Gao et al., 2021; Xiang et al., 2022), and has been demonstrated to be effective at learning universal, generalized representations. To be more specific, contrastive learning focuses on constructing positive and negative pairs to concentrate positive samples and push apart negative ones. Except for existing CV applications, in other areas like the NLP, contrastive learning is also widely used for learning better sentence embedding, Logeswaran.et.al (Logeswaran and Lee, 2019) sampled sentences from other documents as negative pairs for better representation learning. Some recent works (Wu et al., 2020; Fang et al., 2020; Yan et al., 2021; Dong et al., 2022) also utilize contrastive learning for training Transformer models with different negative sampling strategies, e.g., back-translation, or various dropout rates. In this paper, except for achieving better transferable feature embedding for newly emerged tasks, we propose contrastive task-specific attention as the feature selection strategy, which aims to extend a further weight constraint in the universal representation for local/specific tasks. We call it dual-contrastive learning. Our work could achieve better task-agnostic and task-specific knowledge acquisition in the sequence learning process.

3 Approach

3.1 Problem Setting

We focus on continual learning for a sequence of text classification tasks, $\mathcal{S}_\mathcal{T} = \{T_1, T_2, \cdots, T_n\}$, each task T_k contains a dataset $(x_i, y_i) \in D_k$ and the collection of classes of this task is $C_k = \{c_k^1, c_k^2, \ldots, c_k^m\}$ (m is the number of classes in task T_k) (Biesialska et al., 2020). We aim to train a discriminative model, $\Phi(\cdot)$, continuously on newly emerging tasks, and encourage the model to achieve an optimal equilibrium for all classes that have been seen, by optimizing Eq. 1.

$$\arg\min_{(\Phi)} \sum_{k=1}^{n} \mathbb{E}_{(x,y)\in D_k}[L(\Phi(x), y)] \qquad (1)$$

where $L(\cdot, \cdot)$ is a customized loss function.

One common assumption is that there is no overlap between the class sets from different tasks, $C_i \cap C_j = \emptyset$ if $i \neq j$. It is not true in general cases, so we assume that the model treats overlapped classes identically. Therefore, the model pays attention to novel classes and takes advantage of previous experiences.

3.2 Overview

The transferable property refers to an embedding's ability to perform effectively not only on specific tasks but also on others. A notable hindrance to obtaining transferable embeddings lies in managing task-specific information. For example, certain aspects of the embeddings may have huge impacts on task T_i, yet are less significant to the following task, T_{i+1}. The conventional neural network tends to shift its capacity to T_{i+1}, leading to catastrophic forgetting in continuous learning. Nevertheless, training the classifier with the raw embeddings is also a sub-optimal solution, since only the task-specific parts of the embeddings are crucial.

To address this problem, we integrate the attention mechanism and SSL into the proposed framework, which we call the Dual Contrastive Learning framework (DCL), to address the previously mentioned challenges. Figure 1 illustrates the architecture and workflow of our proposed DCL framework, which comprises four key components: 1)

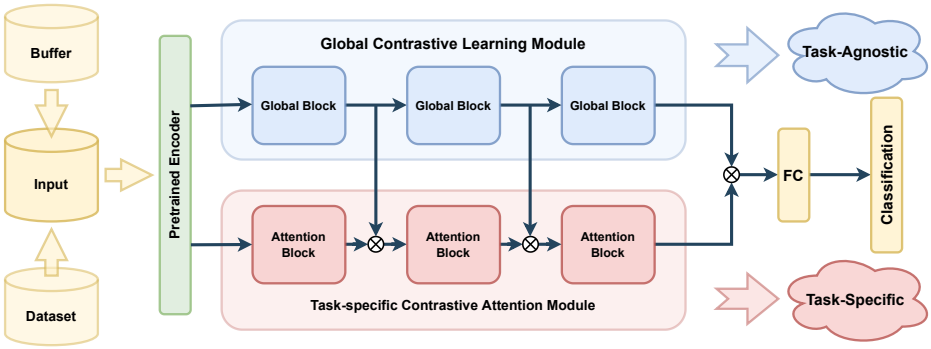


Figure 1: Workflow of DCL. For the current task T_i, we take data from both the memory buffer and task-specific dataset as inputs. We feed them into a pre-trained BERT encoder to obtain the raw embeddings, which will be refined by two collaborative modules: 1) GCL and 2) TCA. GCL produces the refined general embeddings through a sequence of global blocks and TCA progressively adjusts the attention scores based on the outputs of the previous global block and attention block. Finally, we compute the element-wise dot-product of the outputs of these two modules as the final outputs.

a pre-trained encoder (PTE); 2) the task-agnostic global contrastive learning (GCL) module; 3) a task-specific contrastive attention (TCA) module, and 4) a linear classifier. We employ different variants of contrastive learning to train GCL and TCA. The details of each component will be discussed in the following subsections.

3.3 Dual Contrastive learning Mechanism

3.3.1 Contrastive learning

Contrastive learning (Saunshi et al., 2022) is the core of our framework, which aims to learn a discriminative embedding space by pulling semantically-similar instances closer, and pushing apart negative or dissimilar instances. Contrastive learning generally is conducted in a batch-by-batch fashion. Specifically, for a given instance x_i in a mini-batch, contrastive learning finds its most semantically similar neighbor x_i^+, encourages the output embeddings z_i, z_i^+ (of x_i and x_i^+) to have maximized agreement, and considers the rest of the sample pairs in the mini-batch as negative pairs. The cosine similarity ($Sim(a,b) = \frac{a^\top b}{||a||\,||b||}$) is used as the score between instance pairs. Formally, the training objective for a given z_i in a mini-batch with N pairs could be defined as:

$$\ell_i = -\log \frac{e^{sim(z_i, z_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(z_i, z_j^+)/\tau}} \tag{2}$$

where the τ is a temperature parameter.

3.3.2 Global Contrastive Learning Module

Unsupervised contrastive learning has proved its substantial potential to learn a robust and expressive embedding, which helps the model comprehend task-agnostic knowledge. In the unsupervised setting, data augmentation is the de facto technique that generates similar instances, which can be easily extended to the NLP domain, such as substitution and re-ordering. Besides the text-level augmentation, we can apply this method in the embedding space. For example, a recently proposed framework (Gao et al., 2021) generates positive pairs by feeding input into a pre-trained encoder twice, each with a different dropout setting. Later, any embedding of a distinct input is used to construct negative pairs. As shown in Figure 2 a, we define the global embedding by the equations below.

$$\begin{cases} z_i = GCL(PTE(x_i; dp)) \\ z_i^+ = GCL(PTE(x_i; dp')) \end{cases} \tag{3}$$

where $PTE(\cdot)$ denotes the pretrained encoder, dp and dp' are different dropout settings. Then, we substitute them into Eq. 2.

$$\mathcal{L}_{GCL} = \frac{1}{N} \sum_{i=1}^{N} \ell_i \tag{4}$$

The GCL module is a sequence of GCL blocks B_g^l. Each one is implemented by a simple MLP.

3.3.3 Task-specific Contrastive Attention Module

The objective of the TCA module is to produce an attention vector. Recall that the important features are not exactly the same across tasks, we, therefore, adopt this attention vector to help the classifier to focus on the informative parts.

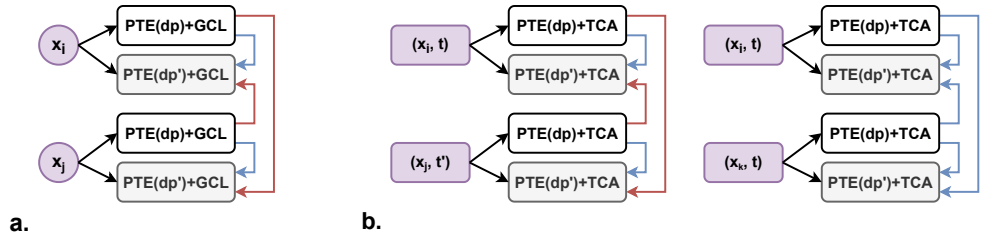


Figure 2: a. Unsupervised contrastive learning. If two instances x_i and x_j, each instance goes through the model twice with different dropout settings. The embedding pair of the same instance is positive, otherwise negative. b. Task-wise contrastive learning. We assume instances x_i and x_k are from the same task, and x_j is from another task. We similarly generate embeddings, but embedding pairs whose instances are from the same task is positive, otherwise negative. Blue arrows indicate positive pairs. Red arrows indicate negative pairs.

We further assume that the distance between two attention vectors should be small if they are from the same task and be large otherwise, shown in Figure 2 b. We guarantee this property through a supervised contrastive learning framework (Khosla et al., 2020).

Compared to the class-wise methods, task-wise supervised contrastive learning is more suitable in this scenario since it captures the differences among multiple tasks. The loss function can be formulated as:

$$\mathcal{L}_{TCA} = \frac{-1}{N}\sum_{i=1}^{N}\left(\log\sum_{t_i=t_k}\frac{e^{sim(a_i,a_k^+)/\tau}}{\sum_j e^{sim(a_i,a_j^+)/\tau}}\right) \tag{5}$$

where a_i is the attention vector and a_i^+ is its positive partner. Similar to the GCL module, the attention vectors are defined as:

$$\begin{cases} a_i = TCA(PTE(x_i; dp), [z_i^l]) \\ a_i^+ = TCA(PTE(x_i; dp'), [z_i^l]) \end{cases} \tag{6}$$

where $[z_i^l]$ is a list of intermediate outputs of GCL blocks.

The TCA module also is a sequence of blocks, which generates the attention vector in multiple steps. Furthermore, the TCA plays the role of feature selection, so it is valuable for knowing what information the GCL has to adjust itself. The number of blocks of the TCA is identical to the GCL. The $[z_i^l]$ contains information on how the GCL produces the feature embedding. We use B_g^l to be the l^{th} block from GCL and B_a^l to be the l^{th} block of TCA.To have better attention vector, each B_a^l utilizes the outputs from both B_g^{l-1} and B_a^{l-1} to generate output, which means $a_i^l = B_a^l(a_i^{l-1} \odot z_i^{l-1})$.

3.3.4 Final Embedding and Classifier

Once we have the global embedding z_i and the task attention vector a_i for a given input (x_i, y_i), we use the fully connected layer to get the final embedding for classification and use Softmax to get the probability vector: $y_i' = \textbf{Softmax}(FC(a_i \odot z_i))$ where $\odot$ denotes the hadamard product operator. Finally, we train the classification component with the conventional cross-entropy loss:

$$\mathcal{L}_{CLS} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(y_i') \tag{7}$$

3.4 Regularization and Replay Policy

To further prevent forgetting, we use regularization (Li and Hoiem, 2017). For task T_t where $t > 2$, we use the model trained by T_{t-1} to produce both GCL embeddings and TCA attention vectors of T_t's data. We use z_i' and a_i' to denote the GCL and TCA outputs of x_i from the current task T_t. The λ_g and λ_t are weights. The loss functions are:

$$\mathcal{L}_{reg} = -\frac{1}{N}\sum_{i=1}^{N}\left(\lambda_g \left\|z_i - z_i'\right\| + \lambda_t \left\|a_i - a_i'\right\|\right) \tag{8}$$

The regularization loss forces the model to learn the new task without changing data representations much, which may result in catastrophic forgetting.

We also use a replay policy to overcome the forgetting problem. Preserving a small quantity of data is necessary. Not only for replaying, but task-wise contrastive learning also needs data from previous tasks. We select the ratio κ of the whole dataset and update the memory buffer. During the training, for every η step, the model is trained with batches sampled from both current T_t and the buffer. During the replay step, the weight of the

Table 1: Different task sequences for continual text classification.

Order	# Task Sequence
1	ag → yelp → yahoo
2	yelp → yahoo → ag
3	yahoo → ag → yelp
4	ag → yelp → amazon → yahoo → dbpedia
5	yelp → yahoo → amazon → dbpedia → ag
6	dbpedia → yahoo → ag → amazon → yelp
7	yelp → ag → dbpedia → amazon → yahoo

regularization loss will be larger, because the primary goal of replaying is to enhance the knowledge of historical tasks.

3.5 Final Loss Function

Now, we have described every component of DCL. The final loss function is the weighted sum of each loss function mentioned in Eq. 4 5 7 8.

$$\mathcal{L} = \mathcal{L}_{CLS} + \mathcal{L}_{GCL} + \mathcal{L}_{TCA} + \mathcal{L}_{reg} \quad (9)$$

Note, during the replay phase, λ_g and λ_t are larger than the non-replay step. Moreover, $\mathcal{L}_{TCA}$ will not be applied in the non-replay step.

4 Experiment

In this section, we follow the same protocol from IDBR (Huang et al., 2021) to evaluate the model performance in the continual learning scenario.

4.1 Datasets

We choose five web text classification benchmarks (Zhang et al., 2015) (**AGNews**, **Yelp**, **Amazon**, **DBPedia** and **Yahoo**) that cover various domains, such as news classification, sentiment analysis, and Q&A classification. In summary, these datasets have 33 classes (Amazon and Yelp have similar labels) in the continual learning experiments. We use the same dataset setting from our baselines (Sun et al., 2020; Huang et al., 2021), in which each task contains 115,000 training examples and 7,600 test examples. We download the data and preprocessing code from (Huang et al., 2021) for fair comparisons. The original data has been split into training and test sets, so the comparison is on the exact same splits. We utilize Google's pre-trained BERT[1] model to compute the raw embeddings, with the maximum text length as 256 words. Table 1 illustrates all the task sequences that we adopted during the experiments. We examine both length-3 task sequences and length-5 task sequences in various orders, and then execute the task classification on the test set.

Following previous papers setting (Huang et al., 2021), we try two different settings for datasets. (1) We tune all the parameters on a sampled dataset (named **Sampled**) by randomly sampling 2000 training, 2000 validation instances per class, and the full test dataset containing 7600 instances for every task. We use it to evaluate the performances quickly, because of resource limitations. (2) We conduct experiments on the full datasets (called **Full**), which is the same as existing work such as MBPA++ (de Masson D'Autume et al., 2019). We use this setting to compare with SOTA to evaluate our model comprehensively. It uses the same test data mentioned above, which contains 7600 instances for each task.

4.2 Baseline and Evaluation

In our experiments, we compare our method with many popular approaches, including Fine-Tuning (Yogatama et al., 2019), Replay (de Masson D'Autume et al., 2019; Wang et al., 2019), Regularization (Li and Hoiem, 2017). We also compare with some recent SOTA approaches: MBPA++ (de Masson D'Autume et al., 2019), LAMOL (Sun et al., 2020), IDBR (Huang et al., 2021).

During the evaluation, we assess the model performance among all observed classes. For example, at time step k (T_k), the evaluation metrics are computed on $\bigcup_{0<i\leq k} C_i$. We report the average incremental accuracy (de Masson D'Autume et al., 2019) that averages all accuracy obtained after each learning step. It's worth noting that researchers usually adopt task squences in different orders rather than conventional **K-fold validation**, since tasks are not trained independently but in a sequence manner (Huang et al., 2021; Sun et al., 2020).

We also calculate the average forgetting measure (Chaudhry et al., 2019) to estimate the forgetting on old tasks. Specifically, for the j-th task, we can calculate the corresponding maximum current forgetting at the time $t = k$ by the formula:

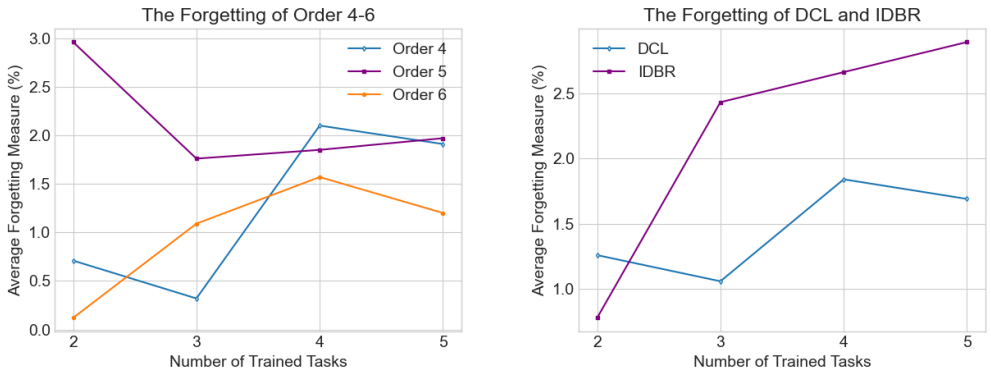
$$\Gamma_j^k = \max_{i\in[j,\ldots,k-1]} (Acc_{i,j} - Acc_{k,j}), \quad j < k \quad (10)$$

where $Acc_{i,j}$ indicates the accuracy of the j-th task when the model is trained after task i. The average

[1] https://github.com/google-research/bert

Table 2: Performance of classification on **sampled** datasets using averaged accuracy over 3 runs, when the model finish training on the last task. All results are average accuracy scores. We make paired t-tests and p-values are less than 0.01, which shows significant differences.

Methods	Length-3 Task Sequences				Length-5 Task Sequences				
	Order-1	Order-2	Order-3	Length-3	Order-4	Order-5	Order-6	Order-7	Length-5
Fine-Tuning	27.58±1.8	38.67±2.11	38.1±2.91	34.79±5.6	31.77±0.6	31.16±1.05	26.36±0.08	20.64±3.53	27.48±4.85
Replay	69.56±0.24	69.06±1.19	71.56±0.25	70.06±1.3	69.42±1.17	70.75±0.23	70.82±0.58	69.7±0.35	70.17±0.92
Regularization	71.43±0.15	71.19±0.55	72.16±0.5	71.59±0.6	72.89±0.34	72.92±0.44	73.17±0.32	72.69±0.09	72.92±0.37
IDBR	71.36±0.31	72.22±0.36	73.08±0.14	72.22±0.76	72.77±0.1	73.82±0.19	73.19±0.33	73.5±0.12	73.32±0.44
Ours	**71.95±0.21**	**73.07±0.16**	**73.31±0.08**	**72.78±0.61**	**73.25±0.18**	**74.34±0.25**	**74.18±0.4**	**73.94±0.09**	**73.93±0.47**
Multi-Task	74.27±0.69				75.16±0.64				



(a) (b)

Figure 3: After finishing one task training, the forgetting measure is calculated. (a) Forgetting measure order 4-6. (b) Compare our method with IDBR on averaged forgetting measure of order 4-6.

forgetting measure is written as the following:

$$\Gamma_k = \frac{1}{k-1}\sum_{j=1}^{k-1}(\Gamma_j^k) \tag{11}$$

which means the average forgetting measures of previous $k-1$ tasks after the kth task has been trained.

4.3 Result and Analysis on Sampled Datasets

In this section, we show the performance of our model that was trained on all tasks, with their average accuracy on test data under the sampled setting. The results are in Table 2. We run each method three times for all orders. We could see that only 1% of stored instances for experience could achieve promising results and help reduce severe knowledge forgetting when fine-tuning. Furthermore, compared with existing works that also work based on experience replay, it is obvious that our proposed approach shows consistent improvements in all orders. The results indicate that contrastive learning under task-agnostic as well as specific feature embedding could further improve performance consistently, under all circumstances. Our model surpasses most of the baselines substantially. For IDBR, we get better results in every task sequence, especially for longer ones. That shows our mechanism could produce more transferable embeddings across different tasks, because of integrating task-agnostic and task-specific information effectively, which is better than others. For further clarification, we use the paired t-test method following the (Sun et al., 2020; Huang et al., 2021). We compare our approach to every baseline for length-3 and length-5, respectively. Each length-3 order experiment involves nine numbers of our method and one baseline, and twelve for each length-5 order. All the p-values are less than 0.01. These results demonstrate that there are significant differences.

4.4 Result and Analysis on Full Datasets

We compare our performance with the previous SOTA methods such as IDBR, MBPA++, and LAMOL (de Masson D'Autume et al., 2019; Sun et al., 2020; Huang et al., 2021), by conducting experiments on the same training and testing sets under the full setting. Table 3 shows the results. Specifically, for MBPA++, there are two different implementation (de Masson D'Autume et al., 2019;

Table 3: Performance of classification on the **full** datasets using averaged accuracy over 2 runs, when the model finished training all tasks. **TT**: whether task-id is available during training. **TI**: whether task-id is available during inference. **LA**: whether need local adaptation during inference. **PM**: pretrained models used for continual learning.

Methods	TT	TI	LA	PM	Length-5 Task Sequences				
					Order-4	Order-5	Order-6	Order-7	Length-5
MBPA++ v1	○	○	•	BERT	70.7	70.2	70.9	70.8	70.7
MBPA++ v2	○	○	•	BERT	74.9	73.1	74.9	74.1	74.3
LAMOL	•	•	○	GPT-2	76.1	76.1	77.2	76.7	76.5
IDBR	•	○	○	BERT	75.9	76.2	76.4	76.7	76.3
Ours	•	○	○	BERT	**77.3**	**77.1**	**77.5**	**77.4**	**77.3**

Sun et al., 2020). Our model significantly outperforms it, although it applies local adaptation during the testing phase. For LAMOL, the model applies the additional task identifier for inference, which is invisible in our experiment setting and makes the prediction much easier because of the extra information. The IDBR adapts specific knowledge to streams of tasks by disentangling the hidden representation via different induction biases. Our model still exceeds the performance of both IDBR and LAMOL with a clear margin. Additionally, compared with the **Sampled** setting, our method gains more improvement versus IDBR in the Full setting. That shows that contrastive learning is highly capable when dealing with large datasets.

The forgetting measures are shown in Figure 3. The results show that our model significantly relieves catastrophic forgetting. Even though longer sequences of tasks generally have a more severe forgetting problem, the absolute value is still reasonable. Here, we shows the forgetting measure over length-5 sequences average score of both ours and IDBR. It reveals that our method alleviates catastrophic forgetting better.

4.5 Ablation Study and Parameter Analysis

4.5.1 Influence of GCL and TCA

We analyze the significance of different components in our approach: the global contrastive learning component (GCL) and the task-specific attention mechanism (TCA). If we entirely remove one of the components, the model will degenerate into a stack of GCL/TCA blocks, because the GCL and TCA work dependently. When we analyze one component, we enable the contrastive loss of the target component and disable the contrastive loss of another one. When all of them are applied together, our approach is conducted. We use a model that is a stack of GCL/TCA blocks without contrastive learning, as the plain model without any components. Here, we provide the results to analyze the effectiveness of each part on both length-3 and length-5 orders for our classification benchmark, with the average accuracy on all the order sequences as the evaluation metrics.

From the results in Table 4, we can observe how each component contributes to the effectiveness of the whole performance. (1). When only considering the global contrastive learning component (GCL), we could observe a remarkable increase compared with the MLP stack model with replay, which shows that contrastive learning could help the model learn more transferable features to reduce the relative knowledge forgetting. (2) When considering re-weighting the different features in the universal embedding space (using attention strategy TCA), it is apparent that the accuracy also increases. The result shows that the attention mechanism for feature selection in different tasks could help to enhance task-specific knowledge. (3) Also, the final combination of all the components demonstrates that the complete model contributes the most to the final experiment result.

4.5.2 Influence of Connecting GCL and TCA

We also evaluate the effectiveness of the connection between GCL and TCA blocks. We take an ablation study that enables and disables the connection between GCL and TCA modules. Figure 4 shows how the connection influences performances. We assume that the TCA produces an attention vector to let the model know which parts of the GCL embeddings are informative for the current task. Based on this assumption, the TCA should get information from the GCL before it produces the final attention vector. Continually getting information from the GCL side helps the TCA adjust its output. Hence, the TCA can generate a more suitable attention vector for both the GCL embedding and the current task. Therefore, we have $a_i^l = B_a^l(a_i^{l-1} \odot z_i^{l-1})$ rather than $a_i^l = B_a^l(a_i^{l-1})$.

Table 4: Ablation study on different modules on Sampled dataset, with results on various task lengths.

	Order-1	Order-2	Order-3	Order-4	Order-5	Order-6	Order-7
No Module	70.26	71.45	71.61	72.08	72.13	72.11	71.84
GCL	71.72	72.52	72.02	72.68	73.88	72.36	72.12
TCA	70.47	71.52	72.1	72.15	72.22	72.76	71.99
Apply All	**71.95**	**73.07**	**73.31**	**73.25**	**74.34**	**74.18**	**73.94**

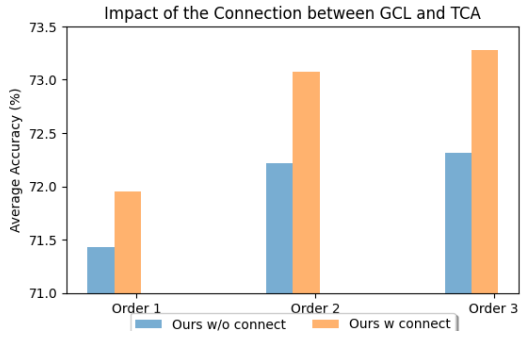


Figure 4: Analyzing GCL and TCA connection.

5 Conclusion

This paper proposes a contrastive learning scheme for incrementally learning embedding under the sequence task scenarios. Our approach leverages the contrastive learning approach (GCL) in the embedding space to obtain better task-agnostic features, further, we propose a contrastive-attention mechanism (TCA) for task-specific feature re-weighting, which could constrain task-specific knowledge in the universal embedding space more appropriate. Extensive experiments demonstrate that our approach could achieve clearly better results on different benchmarks, compared with existing state-of-the-art continual learning approaches. In future work, we hope to further reduce the cost of training on new tasks through the Coreset/active learning (Yoon et al., 2021; Li et al., 2022) optimization.

6 Limitations

The DCL framework uses contrastive learning to learn both task-agnostic and task-specific knowledge, in order to reduce catastrophic forgetting, our framework still needs to store a small part of the historical data as a replay mechanism during the learning period, which is a limitation in some application that need preserve data privacy. We will solve this limitation in future work to replace the original data with the prompt represent.

7 Acknowledgement

The research reported herein was supported in part by NIST Award # 60NANB23D007, NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, and DGE-1906630, ONR awards N00014-17-1-2995 and N00014-20-1-2738.

References

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

Magdalena Biesialska, Katarzyna Biesialska, and Marta R Costa-Jussa. 2020. Continual lifelong learning in natural language processing: A survey. *arXiv preprint arXiv:2012.09823*.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with a-gem. *ICLR*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607.

Sunyi Chi, Bo Dong, Yiming Xu, Zhenyu Shi, and Zheng Du. 2023. Apam: Adaptive pre-training and adaptive meta learning in language model for noisy labels and long-tailed learning. *arXiv preprint arXiv:2302.03488*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Bo Dong, Yiyi Wang, Hanbo Sun, Yunji Wang, Alireza Hashemi, and Zheng Du. 2022. Cml: A contrastive meta learning method to estimate human label confidence scores and reduce data collection cost. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 35–43.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*.

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *EMNLP*.

Binzong Geng, Min Yang, Fajie Yuan, Shupeng Wang, Xiang Ao, and Ruifeng Xu. 2021. Iterative network pruning with uncertainty regularization for lifelong sentiment classification. In *SIGIR*, pages 1229–1238.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *ACL*, pages 6429–6440.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Longlong Jing and Yingli Tian. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *NeurIPS*, 33:18661–18673.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

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*, page 201611835.

Xiaoting Li, Yuhang Wu, Vineeth Rakesh, Yusan Lin, Hao Yang, and Fei Wang. 2022. Smartquery: An active learning framework for graph neural networks through hybrid uncertainty reduction. In *CIKM*, pages 4199–4203.

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

Bing Liu and Sahisnu Mazumder. 2021. Lifelong and continual learning dialogue systems: learning during conversation. *Proceedings of AAAI-2021*.

Lajanugen Logeswaran and Honglak Lee. 2019. An efficient framework for learning sentence representations. *ICLR*.

J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. *EMNLP*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, pages 524–540. Springer.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Mufan Sang, Haoqi Li, Fang Liu, Andrew O Arnold, and Li Wan. 2022. Self-supervised speaker verification with simple siamese network and self-supervised regularization. In *ICASSP*, pages 6127–6131. IEEE.

Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. 2022. Understanding contrastive learning requires incorporating inductive biases. *arXiv preprint arXiv:2202.14037*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. Lamol: Language modeling for lifelong language learning. *ICLR*.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Sentence embedding alignment for lifelong relation extraction. *NAACL*.

Yigong Wang, Zhuoyi Wang, Yu Lin, Latifur Khan, and Dingcheng Li. 2021a. Cifdm: continual and interactive feature distillation for multi-label stream learning. In *SIGIR*, pages 2121–2125.

Zhuoyi Wang, Yuqiao Chen, Chen Zhao, Yu Lin, Xujiang Zhao, Hemeng Tao, Yigong Wang, and Latifur Khan. 2021b. Clear: Contrastive-prototype learning with drift estimation for resource constrained stream mining. In *Proceedings of the Web Conference*, pages 1351–1362.

Zhuoyi Wang, Yigong Wang, Bo Dong, Sahoo Pracheta, Kevin Hamlen, and Latifur Khan. 2020. Adaptive margin based deep adversarial metric learning. In *IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity)*, pages 100–108. IEEE.

Jiapeng Wu, Yishi Xu, Yingxue Zhang, Chen Ma, Mark Coates, and Jackie Chi Kit Cheung. 2021. Tie: A framework for embedding-based incremental temporal knowledge graph completion. In *SIGIR*, pages 428–437.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

Siyuan Xiang, Anbang Yang, Yanfei Xue, Yaoqing Yang, and Chen Feng. 2022. Self-supervised spatial reasoning on multi-view line drawings. In *CVPR*, pages 12745–12754.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. *ACL*.

Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, and Yuan Jiang. 2019. Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In *KDD*, pages 74–82.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. 2021. Online coreset selection for rehearsal-based continual learning. *ICLR*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *NeurIPS*, 28.

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. 2022. Continual prompt tuning for dialog state tracking. *ACL*.

A Algorithm Pseudocode

The final pseudo-code is in Algorithm 1. The model is initialized with an empty memory buffer and uses Eq. 4 to warm up. For the first task, the training process only contains classification and unsupervised learning. For the remaining tasks, the model is warmed up by Eq. 9. During the warming-up process, the GCL has more samples to learn test-agnostic knowledge, and the TCA learns the differences among tasks, especially for novel tasks. After warming up, the training process has two stages, replay or non-replay. In the non-replay stage, the model is trained by Eq. 9 without $\mathcal{L}_{TCA}$. In the replay stage, one batch of historical data is fetched from the memory buffer. The training batch consists of both historical and novel data. The loss function is Eq. 9. Once the current training process finishes, the data selection mechanism selects data from the current dataset and stores it in the memory buffer. Then, the model waits for the next novel task.

B More Experiment Details

Table 5: The description of the **Sampled** training dataset we used for incremental classification. Type means the domain of task classification. Each task has a validation set that has same size with the training set. It is the same setting with (Huang et al., 2021).

Raw	Dataset	# Type	# classes	# Train	# Test
Text	AGNews	News	4	8,000	7600
	Yelp	Sentiment	5	10,000	7600
	Amazon	Sentiment	5	10,000	7600
	DBPedia	Wikipedia	14	28,000	7600
	Yahoo	Q&A	10	20,000	7600

B.1 Implementation

Our model uses the Pytorch framework that is trained on an A100 GPU and sequentially fine-tuned on different (3 or 5-length) tasks. We utilize an AdamW optimizer to train the model and set the output dimension of the BERT encoder as 768. For both GCL and TCA blocks, we adopt MLP followed by a Tanh layer. Because of the information exchange between GCL and TCA, we need at least two blocks for both GCL and TCA. The classifier is simply one linear layer followed by a softmax layer.

We set the mini-batch size as 8 for the sample datasets and 32 for the full. Our approach has different learning rates for different components:

Algorithm 1 DCL

Require: $\mathcal{S}_\mathcal{T}$ - Data Stream; n - number of tasks;
Ensure: Optimal Models PTE, GCL, TCA, FC;

$\mathcal{B} = \{\}$ - Initialize the memory buffer;
Use T_1 with Eq. 4 to warmup;
for $T_t \in \mathcal{S}_\mathcal{T}, t \in \{1, 2 \cdots n\}$ **do**
  if $t == 1$ **then**
    for $batch \subset T_t$ **do**
      Data augmentation and getting model outputs;
      Optimize $\mathcal{L} = \mathcal{L}_{CLS} + \mathcal{L}_{GCL}$;
    end for
  else
    Store z_i and $a_i, \forall x_i n \in T_t \cup \mathcal{B}$
    Warmup with Eq. 9 with $\{T_t$ batch $\} \cup \mathcal{B}$
    for $batch \subset T_t$ **do**
      if Replay Step **then**
        Sample one batch from $\mathcal{B}$ to be $batch_{past}$
        $batch = batch_{past} \cup batch$
        Data augmentation and getting model outputs;
        Optimize Eq. 9
      else
        Data augmentation and getting model outputs;
        Optimize Eq. 9 without $\mathcal{L}_{TCA}$
      end if
    end for
  end if
  $\mathcal{B}'$ = {Data selection} $\subset T_t$
  $\mathcal{B} = \mathcal{B} \cup \mathcal{B}'$ - Update memory buffer
end for

an initial learning rate $\in$ {5e-5, 6e-5, 8e-5} for parts including GCL, TCA and classifier, and {1e-5, 2e-5, 3e-5} for encoder (BERT). The weight decay for all parameters is 0.01. λ_{gcl}^{reg} is 0.25 and λ_{tca}^{reg} is 0.15 in non-replay step, and 2.5 and 1.5 in replay step respectively.

For the memory volume for experience replay, we store 1% of seen examples in the episodic memory module during the experiment. Besides, we set the replay frequency as 10, which indicates that we execute the experience replay once every ten steps.

B.2 More Baseline Details

Fine-Tuning (Yogatama et al., 2019) simply trains the existing model with tasks that appear sequentially. Replay (de Masson D'Autume et al., 2019; Wang et al., 2019) is a popular method to overcome forgetting by using episodic memory. Regularization (Li and Hoiem, 2017) uses L2 to avoid forgetting. Except for only replay, we run replay + regularization to be one baseline. We also compare with some recent SOTA approaches: MBPA++ (de Masson D'Autume et al., 2019), a lifelong language learning setup with an episodic memory module that learns from a stream of text examples, without any dataset identifier. It mitigates catastrophic forgetting through sparse experience replay and local adaptation. Here we set the number of neighbors K = 32, and the local adaptation steps L = 30. LAMOL (Sun et al., 2020) is a language model for lifelong language learning that simultaneously learns to solve tasks and generate training samples – it prevents catastrophic forgetting by replaying pseudo-samples of previous tasks, with GPT-2 as the generator. Here we choose the GEN token and pseudo sample ratio as 0.2 for old classification tasks. IDBR (Huang et al., 2021) uses the idea of information disentanglement to generate the features independently, and it also recombines the disentanglement features for classifying. Table 5 shows information about Sample data.

Table 6: Ablation study on different selection methods on sampled dataset, with results on various task lengths.

Sample Method	Order 1	Order 2	Order 3	Order 4
Kmeans	**72.16**	72.91	**73.36**	73.12
Random	71.95	**73.07**	73.28	**73.25**

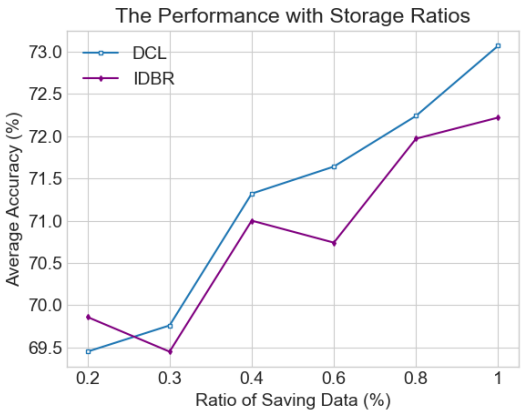


Figure 5: Analyzing the influence of the storage ratio for old data.

Table 7: Supplement results for ablation study analyzing the influences of regularization loss and contrastive learning.

	No Any	Contrastive	L_reg	Both
Order 1	69.42	69.89	70.26	**71.95**
Order 2	68.98	70.48	71.45	**73.07**
Order 3	71.24	71.82	71.61	**73.31**
Order 4	68.83	70.09	72.08	**73.25**
Order 5	70.81	72.21	72.13	**74.34**
Order 6	71.12	72.17	72.11	**74.18**
Order 7	70.38	71.07	71.84	**73.94**

C Extra Ablation Study

C.1 Influence of Data Selection Method

Because of the limited memory space for the storage of historical data, it is preferable to select the more informative data points. We consider two methods: random selection, and KMeans (MacQueen, 1967), to sample data and update the buffer. We use scikit-learn (Pedregosa et al., 2011) to implement KMeans. According to our experiments in Table 6, KMeans does not show a significant competitive advantage for performance over the random selection, especially for large datasets. More importantly, the running time of KMeans is much longer than random selection when the model stores thousands of instances. In many scenarios, the systems that need continual learning are time sensitive, so it is valuable to update the model quickly. Therefore, we decided to adopt the random selection.

C.2 Influence of Storage Ratio

We perform further analysis of the storage ratio (for older data) for both our approach and the current SOTA approach IDBR and show the result on the 3-length tasks in Figure 5. We could see that when the store ratio is low (fewer previous experiences are permitted to access), the difference between our approach and IDBR is not obvious. As we store relatively more instances, our approach could improve the performance better and more stably than IDBR, until our previous setting (1%). Moreover, Our approach grows steadily, but IDBR is not stable. It shows that contrastive learning can learn more generalized embeddings compared with the regular cross-entropy function when there is more previous data.

C.3 Influence of Contrastive learning and regularization loss

We provide more ablation study results to clarify the effectiveness of our framework. Table 7 shows the impacts of regularization loss and contrastive learning. We simply disable the regularization loss when we test contrastive learning in our model. We use the structure of GCL module without contrastive learning loss to test regularization loss. We demonstrate the results of all seven orders for 3-length and 5-length tasks. It is clear that we cannot achieve the best performance if we only apply one technique. Thus, our framework has the best result.