Prototype Conditioned Generative Replay for Continual Learning in NLP

Xi Chen*, Min Zeng*

Hong Kong University of Science and Technology
chenxi.mail.1005@gmail.com, min.zeng.u@gmail.com

Abstract

Generative replay has proven effective in addressing the catastrophic forgetting issue of continual learning (CL) in natural language processing (NLP). However, relying on a single task-specific token or prompt often falls short in generating pseudo-samples that accurately reflect the true data distribution. This leads to issues of semantic inconsistency and scale inconsistency. To tackle these challenges, we propose a Prototype Conditioned Generative Replay (PCGR) method, which enhances generative reply by incorporating task-level statistics through a Prototype Conditioned Variational Autoencoder (PCVAE). Specifically, task-level embedding statistics are stored as prototypes for each old task. When a new task is introduced, PCVAE draws samples from task-specific prototype-based distributions to generate pseudo-samples. By incorporating the prototype, the generated pseudo-samples are both more representative and sufficiently diverse to reflect the real data distribution. Furthermore, as previously stored prototypes may become outdated due to evolving model parameters, we propose a Prototype Shift Estimation (PSE) to adjust for these changes. Experiments on NLP tasks across two different scenarios show that PCGR outperforms previous state-of-the-art (SOTA) methods.

1 Introduction

Continual learning (CL) aims at enabling models to continually acquire, accumulate, and exploit novel knowledge from a non-i.i.d. stream of tasks while retaining previously learned knowledge (Wu et al., 2022; Wang et al., 2024a). In practice, however, when learning new tasks, models often encounter the notorious issue of catastrophic forgetting (CF), which refers to the model updating its parameters for the new task at the cost of forgetting previously acquired knowledge (McCloskey and Cohen, 1989; Parisi et al., 2019). In NLP, pre-trained large language models (LMs) have demonstrated promising performance, leading to an increasing focus on integrating these models into dynamic data distributions, evolving task structures, and shifting user preferences.

Generative replay methods (Sun et al., 2020; Zhao et al., 2022; Zeng et al., 2024) have emerged as promising approaches to tackle the problem of CF in CL. The core idea is to generate authentic pseudo-samples of previous tasks and combine them with the current training data to jointly tune the model. However, existing generative replay approaches typically rely on a single task-specific token (Sun et al., 2020) or prompt (Zhao et al., 2022; Zeng et al., 2024) to generate pseudo-samples. Due to the limited information (Sun et al., 2020) provided by the task-specific tokens or prompts, the generated pseudo-samples often fail to accurately reflect the real data distribution. This leads to two underlying issues: semantic inconsistency, where the pseudo-samples are not semantically aligned with the real data, causing unclear boundaries among tasks and unintended modifications of important parameters; and scale inconsistency, where the pseudo-samples lack the diversity of real data, leading the model to overfit the new task and exacerbating CF.

To tackle these problems, we propose the Prototype Conditioned Generative Rehearsal method (PCGR), which employs Prototype Conditioned Variational Autoencoder (PCVAE) to generate pseudo-samples that more accurately reflect the real data distribution, effectively mitigating catastrophic forgetting. Drawing inspiration from class incremental learning (CIL) in computer vision (CV) (Belouadah et al., 2021), which utilize prototypes (Belouadah and Popescu, 2019; Tan et al., 2024), the previously saved representation statistics, to mitigate CF in feature classification tasks, we introduce the concept of prototypes into CL in

*The two authors contribute equally.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 12754–12770
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

NLP. Prototypes, which carry rich "dark knowledge," can faithfully reflect the real data distribution, making them well-suited to address the issues of semantic and scale inconsistency in CL in NLP. In our context, to tackle the semantic and scale inconsistency issues, we define the prototype as the task-level representation mean and standard deviation. PCVAE then approximates task-level data distribution by conditioning on the prototype, and utilizes samples drawn from the prototype-based Gaussian distribution to guide pseudo generation. This ensures that the pseudo-samples are both representative and sufficiently diverse to accurately reflect the real data distribution. Moreover, as model parameters evolve with the introduction of new tasks, the prototypes for previous tasks shift accordingly. However, due to the unavailability of previous data, recalculating these prototypes is not feasible. Inspired by (Yu et al., 2020; Tan et al., 2024), which estimates representation mean drift using only the current task data, we introduce a Prototype Shift Estimation (PSE) method to compensate for the shift of previous prototypes without requiring access to prior data. In summary, our key contributions include:

- We propose PCGR, which introduces the concept of prototype from CV to CL in NLP. PCGR leverages a PCVAE to generate pseudo-samples that are both representative and diverse, accurately reflecting the real data distribution. This approach effectively mitigates catastrophic forgetting in continual learning.
- We propose PSE, which estimates the shift of prototypes for old tasks without the need for access to previous data.
- Extensive experiments and comprehensive analyses demonstrate the remarkable performance of PCGR and the superior quality of the generated pseudo-samples.

2 Related Work

2.1 Continual Learning

Continual learning approaches can be categorized into three main strategies:

Regularization approaches aim at striking a balance between protecting already learned tasks and granting sufficient flexibility for a new task by incorporating regularization terms into the loss function(Kirkpatrick et al., 2017; Aljundi et al., 2018; Mi et al., 2020a; Mundt et al., 2023). However, adding multiple regularization terms may overly constrain the model, potentially downgrade model performance (Parisi et al., 2019).

Architectural approaches mitigate CF by constructing task-specific parameters rather than incrementally training all tasks with a shared set of parameters (Hu et al., 2019; Zhai et al., 2020; Madotto et al., 2021; Geng et al., 2021; Zhang et al., 2022; Wang et al., 2023; Ke et al., 2023; Zhao et al., 2024). However, architectural approaches may overlook knowledge sharing among tasks and model parameter increases as the number of tasks increases.

Rehearsal approaches, also known as replay-based approaches, address CF by approximating and recovering previous data distribution (Wang et al., 2024a). Depending on whether the replayed data is real data or the pseudo generated data, rehearsal approaches can be categorized into *experience replay* (Rebuffi et al., 2017; Lopez-Paz and Ranzato, 2017) and *generative replay* (Mi et al., 2020b; Sun et al., 2020; Chuang et al., 2020; Kanwatchara et al., 2021; Zhao et al., 2022; Zeng et al., 2024). However, experience replay can be impractical due to memory limitations and the privacy concerns, while generative replay may suffer from limited diversity or poor alignment with the designated task (Zeng et al., 2024).

In this paper, we focus on generative replay due to its impressive performance. Specifically, LAMOL (Sun et al., 2020) leverages the generative capability of language models to produce pseudo-samples using task-specific tokens while learning new tasks. L2KD (Chuang et al., 2020) enhances LAMOL by employing knowledge distillation to retain previously learned knowledge. PCLL (Zhao et al., 2022) utilizes a prompt-conditioned VAE to enhance generative replay. DCL (Zeng et al., 2024) refines PCLL by leveraging the flexibility of the Dirichlet distribution (Zeng et al., 2019) to model the latent variables in VAE. All these previous works rely on a single task-specific token or prompt to generate pseudo-samples, which may not be representative and diverse enough to approximate the true data distribution of previous tasks.

2.2 Prototype

Prototype (Yu et al., 2020; Zhu et al., 2021a,b; Tan et al., 2024) refers to representation statistics, typically the representation means (Wang et al., 2024a), and is widely used to recover original representation distribution in the deep feature space in CIL in CV. Class incremental learning (Zhou et al., 2024) is a subfield of CL, where tasks arrive sequentially with each task containing a set of new classes. In CIL, a feature extractor and a unified classifier should be learned to classify all classes. Specifically, PASS (Zhu et al., 2021b) memorizes class-level prototypes and arguments the prototypes via Gaussian noise to preserve discrimination between old and new classes. SemanAug (Zhu et al., 2021a) memorizes class-level distribution information as prototypes and uses it to approximate the original feature distribution to avoid CF.

A central challenge with prototypes is the representation shift caused by the sequential updating of the feature extractor, compounded by the unavailability of previous data for recalculating the prototypes. Specifically, SDC (Yu et al., 2020) is an effective method to alleviate the representation mean shift challenge without requiring access to previous data.

In this paper, as mentioned in Section 2.1, pseudo-samples generated by current methods may not accurately approximate the previous data distribution. Therefore, PCGR proposes to incorporate the prototype into the generative replay, as it have proven effective in reflecting real data distribution. However, PCGR also encounters the representation shift issue. Inspired by SDC, we introduce PSE to mitigate this challenge. Furthermore, it is essential to note that we cannot directly adapt prototypes for CL in NLP. Unlike CIL in CV, CL in NLP encompasses not merely classification tasks and faces additional challenges due to the complexity and diversity of natural language (Ke and Liu, 2022; Mehta et al., 2023).

3 Methodology

3.1 Problem Definition

A CL model f_θ in NLP learns a stream of tasks $T = \{T_1, \cdots, T_N\}$ sequentially, where N represents the total task number and can potentially be infinite. For each task T_n, corresponding dataset and CL model are denoted as $D_n = \{x_i^n, y_i^n)\}_{i=1}^{N_n}$ and f_θ^n, respectively, where (x_i^n, y_i^n) represents a general input-output pair and N_n is the sample number. CL aims to develop a model that excels in every task it encounters while minimizing the forgetting of previously learned knowledge.

3.2 Overview

Figure 1(a) depicts the overview of PCGR, comprising three steps: prototype-conditioned pseudo generation, new task learning, and prototype updating. (i) *Prototype-conditioned pseudo generation*: Before learning the new task, different from previous works which only use a single prompt or token to generate pseudo-samples, PCVAE utilizes samples drawn from prototype-based distributions to generate pseudo-samples that are representative and diverse enough to reflect the real data distribution of prior tasks. (ii) *New task learning*: Subsequently, the LM, along with PCVAE, continues training with the generated pseudo-samples and current task samples to learn the new task while minimizing CF. (iii) *Prototype updating*: Once the new task is learned, the prototype for the new task is calculated, and the prototypes for previous tasks are updated using PSE.

3.3 Prototype

Prototype p is defined as the mean and standard deviation (std) of data sample representations, referred to as the prototype's semantic part μ and the prototype's distribution scale part σ, respectively. Denoting $(x_i^n, y_i^n) \in D_n$ as u_i^n, corresponding representation $r_i^n \in \mathbb{R}^d$ in the deep feature space of f_θ^n is calculated as the average self-attention (Vaswani, 2017) over the hidden states of $f_\theta^n(u_i^n)$ in the last layer, where d denotes the dimension of the hidden state. The prototype $p_n^m = (\mu_n^m, \sigma_n^m)$ for task T_n is calculated as follows:

$$\begin{aligned} \mu_n^m &= \frac{1}{N_n} \textstyle\sum_{i=1}^{N_n} r_i^n, \\ (\sigma_n^m)^2 &= \frac{1}{N_n - 1} \textstyle\sum_{i=1}^{N_n} (r_i^n - \mu_n^m)^2. \end{aligned} \tag{1}$$

3.4 Prototype Conditioned VAE

PCVAE employs the prototype-based distribution as the prior distribution and samples from the corresponding distribution to generate authentic pseudo-samples. It is essential to note that the LM shares parameters with the encoder and decoder of PCVAE, facilitating the construction of a unified model for CL. The graphical model of PCVAE is illustrated in Figure 1(b), please refer to Figure 3 in Appendix A for detail model architecture and data flow.

Inferencing Stage For a given prototype $p = (\mu, \sigma)$, the corresponding prototype-based distribution, is defined as the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. During inference, as shown in Figure 1(b), PCVAE samples $s \sim \mathcal{N}(\mu, \sigma^2)$, maps it into the latent variable z using the Prior Network with parameters ϕ, and generates a pseudo utterance $\tilde{u}$

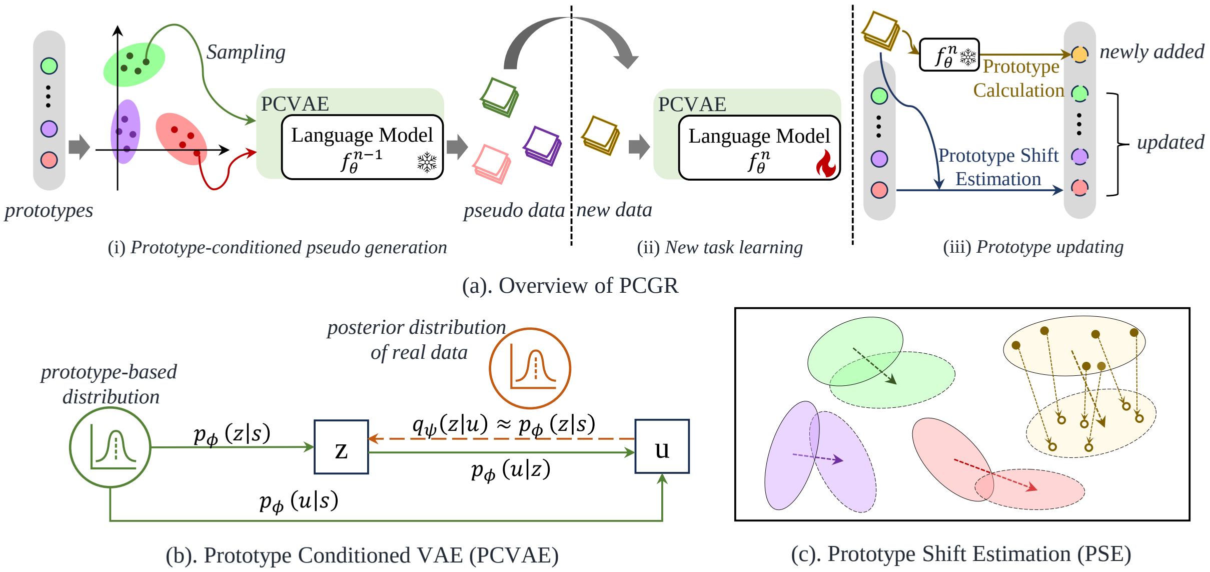


Figure 1: (a) Overview of PCGR. PCGR consists of three steps: (i) Before learning the new task, PCVAE generates pseudo-data by sampling from prototype-based distributions. (ii) The LM and PCVAE are trained with both pseudo-data and new data. (iii) After learning, the prototype for the new task is calculated, and prototypes for previous tasks are updated using PSE. (b) The graphical model of PCVAE. (c) PSE which estimates the unknown shift of previous prototypes based on the observed drift in new data.

using the decoder. This generative inference process can be expressed by the following conditional distribution:

$$p_\phi(\tilde{u}|s) = p_\phi(z|s)p_\phi(\tilde{u}|z). \tag{2}$$

Training Stage According to the inference stage depicted in Equation 2, PCVAE aims to approximate $p_\phi(z|s)$ and $p_\phi(\tilde{u}|z)$. As shown in Figure 1(b), PCVAE learns $p_\phi(z|s)$ by approximating it with the posterior distribution of the real data $q_\psi(z|u)$ using a Recognition Network with parameters ψ and an encoder. It acquires the generative capability of $p_\phi(u|z)$ by maximizing the likelihood of the reconstructed samples based on z. Consequently, the loss of PCVAE, $\mathcal{L}_{PCVAE}$, can be expressed as:

$$\begin{aligned}\mathcal{L}_{PCVAE}(\theta,\phi,\psi;u,z,s) = &-\mathbb{E}_{q_\psi(z|u)}[\log p_\phi(u|z)] \\ &+ \lambda D_{KL}(q_\psi(z|u)||p_\phi(z|s)),\end{aligned} \tag{3}$$

where $\mathbb{E}_{q_\psi(z|u)}[\log p_\phi(u|z)]$ denotes the reconstruction loss, D_{KL} represents the non-negative Kullback–Leibler divergence (Kullback and Leibler, 1951), and λ represents corresponding weight. Additionally, since the LM f_θ, along with PCVAE, keeps updating during the training stage, rendering the prototype for the current task inaccessible, we use the batch-level sample representation mean as a proxy for $s \sim \mathcal{N}(\mu, \sigma^2)$.

3.5 Prototype Shift Estimation

Sample representations drift in the deep feature space when the LM is learned in a sequential manner, causing previously memorized prototypes to be outdated. Using these outdated prototypes in pseudo-generation results in a performance drop. When previous data is unavailable, PSE, as shown in Figure 1(c), aims to estimate the unknown shift of previous prototypes based on the known drift of the current task data, before and after learning a new task. PSE consists of two components: Semantic Drift Estimation (SDE), drawn from (Yu et al., 2020), and Distribution Scale Variation Estimation (DSVE), a newly proposed method that focuses on estimating changes in the prototype distribution scale.

Semantic Drift Estimation Given T_k, with $k < t$, we define the prototype semantic part drift from the deep feature space of f_θ^{n-1} to f_θ^n as: $\triangle\mu_k^{n-1\to n} = \mu_k^n - \mu_k^{n-1}$. After the LM learning T_n, $\triangle\mu_k^{n-1\to n}$ is estimated as the weighted average of current data's drifts, where the weights are determined by their respective distance to the previous prototype semantic part μ_k^{n-1}:

$$\begin{aligned}\tilde{\triangle}\mu_k^{n-1\to n} &= \frac{\sum_{i=1}^{N_n}\alpha_i(r_i^n - r_i^{n-1})}{\sum_{i=1}^{N_n}\alpha_i}, \\ \alpha_i &= e^{-\frac{\|r_i^{n-1}-\mu_k^{n-1}\|}{2s^2}},\end{aligned} \tag{4}$$

where s is a hyperparameter, r_i^{n-1} and r_i^n are available before and after the LM learning T_n, respectively. The prototype semantic part can be updated as:

$$\mu_k^n = \mu_k^{n-1} + \tilde{\triangle}\mu_k^{n-1\to n}. \quad (5)$$

Distribution Scale Variation Estimation Given task T_k, the prototype distribution scale variation from the deep feature of f_θ^{n-1} to f_θ^n is defined as:

$$\triangle\sigma_k^{n-1\to n} = \frac{\sigma_k^n}{\sigma_k^{n-1}}. \quad (6)$$

Equation 6 can be transformed from the multiplicative form to the additive form by taking its logarithm:

$$\triangle \log\sigma_k^{n-1\to n} = \log\sigma_k^n - \log\sigma_k^{n-1}. \quad (7)$$

After the LM learns T_n, the variation $\triangle \log\sigma_k^{n-1\to n}$ is estimated as the weighted average of current data's distance change to the corresponding prototype :

$$\begin{aligned} &\tilde{\triangle}\log\sigma_k^{n-1\to n} = \\ &\frac{\sum_{i=1}^{N_n}\beta_i[\log(r_i^n - \mu_n^n) - \log(r_i^{n-1} - \mu_n^{n-1})]}{\sum_{i=1}^{N_n}\beta_i}, \\ &\beta_i = e^{-\frac{\left\|\log(r_i^{n-1}-\mu_n^{n-1})-\log\sigma_k^{n-1}\right\|}{2c^2}}, \end{aligned} \quad (8)$$

where c is a hyperparameter. The prototype distribution scale part can be updated via:

$$\sigma_k^n = e^{(\log\sigma_k^{n-1} + \tilde{\triangle}\log\sigma_k^{n-1\to n})}. \quad (9)$$

3.6 Integrated Objective of PCGR

In the new task learning stage of PCGR, given task T_n, the LM f_θ^n is learned with PCVAE simultaneously by minimizing:

$$\mathcal{L}_{LM} = -\textstyle\sum_{i=1}^{N_n}\log p_\theta(x_i^n, y_i^n) + \log p_\theta(y_i^n|x_i^n). \quad (10)$$

To further mitigate CF, we leverage knowledge distillation (KD) (Hinton, 2015) to retain previously learned knowledge in f_θ^{n-1} by minimizing:

$$\begin{aligned} \mathcal{L}_{KD} =& \textstyle\sum_{i=1}^{N_n}[\gamma\cdot\tau^2\cdot\mathcal{L}_{KL}(l_i^n, l_i^{n-1}) \\ &+ (1-\gamma)\mathcal{L}_{CE}(l_i^n, u_i^n)], \end{aligned} \quad (11)$$

where l_i^n and l_i^{n-1} are logits of f_θ^n and f_θ^{n-1} with temperature τ, respectively, and γ is hyperparameter. Consequently, the integrated learning objective of PCGR is:

$$\mathcal{L} = \mathcal{L}_{PCVAE} + \mathcal{L}_{LM} + \mathcal{L}_{KD}. \quad (12)$$

4 Experiments

4.1 Evaluation Benchmark

Following (Zhang et al., 2022), we evaluate our method under two common scenarios.

CL on similar tasks All tasks in the sequential order share the same task pattern. In this scenario, we select 12 natural language generation tasks.

CL on dissimilar tasks Tasks in the sequential order exhibit various task patterns. The task distributions vary relatively large, posing a bigger challenge for CL methods to maintain previous knowledge while learning new tasks. In this scenario, we select 12 tasks spanning five task types: programming language generation, intent detection, dialogue state tracking, natural language generation, and question answering.

In each scenario, we explore four task sequential orders. For details on datasets and tasks, please refer to Appendix B.1, and for task sequential orders, see Appendix B.2.

4.2 Metrics

Let $a_{i,j}$ denote the testing performance on the j-th task after the CL method has learned the i-th task. The evaluation metrics for CL methods are defined as follows:

Average Performance (AP) (Chaudhry et al., 2018) The average performance of all tasks after the CL model learning the last task, $AP = \frac{1}{N}\sum_{i=1}^{N} a_{N,i}$.

Forward Transfer (FWT) (Lopez-Paz and Ranzato, 2017) FWT measures the influence of learning previous tasks on the new task, $FWT = \frac{1}{N}\sum_{i=1}^{N}(a_{i,i} - a_{0,i})$, where $a_{0,i}$ refers the performance of learning the i-th task individually.

For evaluation metrics on each individual task, please refer to Appendix B.3.

4.3 Baselines

We evaluate PCGR using 12 methods, categorized as follows: (2) is a regularization method, (3)-(6) are architectural methods, (7)-(11) are rehearsal methods, and (12) represents the upper bound. Specifically, (1) **Finetune** (Yogatama et al., 2019) sequentially tunes the language model on new tasks; (2) **EWC** (Kirkpatrick et al., 2017) adds regularization to the loss function to avoid updating important parameters for previous tasks; (3) **Adapter** (Madotto et al., 2021) adds task-specific adapters to avoid CF; (4) **ACM** (Zhang et al., 2022) is a modification of Adapter. It reuses previous adapter mod-

Method	Similar		Dissimilar	
	AP ↑	FWT ↑	AP ↑	FWT ↑
Finetune (Yogatama et al., 2019)	8.38±3.02	-9.82±0.34	12.86±1.49	-6.50±0.17
EWC (Kirkpatrick et al., 2017)	22.31±3.93	-10.57±0.57	19.17±1.41	-19.53±13.19
Adapter (Madotto et al., 2020)	33.60±0.00	N/A	45.16±0.00	N/A
ACM (Zhang et al., 2022)	37.12±0.59	-9.97±0.22	41.60±2.70	-8.24±0.75
O-LoRA (Wang et al., 2023)	33.92±2.15	-8.65±0.42	19.32±1.34	-23.59±1.19
SAPT (Zhao et al., 2024)	43.80±0.22	-5.47±0.33	55.40±1.09	-9.43±0.62
InsCL (Wang et al., 2024b)	41.69±2.19	-4.87±0.34	52.35±2.50	-9.77±0.82
LAMOL-g (Sun et al., 2019)	36.97±1.45	-9.53±0.44	43.32±4.01	-6.20±0.73
LAMOL-t (Sun et al., 2019)	37.78±0.52	-9.81±0.64	44.56±1.83	-7.02±0.44
PCLL (Zhao et al., 2022)	33.75±2.76	0.64±0.17	43.94±1.98	-0.24±1.01
DCL (Zeng et al., 2024)	43.75±1.28	0.73±0.20	56.44±1.04	-0.55±0.45
PCGR (Ours)	**46.10±0.86**	**0.99±0.35**	**59.48±0.55**	**0.43±0.44**
Multi (Upper Bound)	49.70	N/A	70.98	N/A

Table 1: Comparison results of PCGR and baselines. The best results are emphasized in bold.

ules based on task similarity to enlarge knowledge sharing; (5) **O-LoRA** (Wang et al., 2023) learns tasks in orthogonal low-rank vector subspaces, minimizing interference while incurring only marginal additional parameter costs; (6) **SAPT** (Zhao et al., 2024) is a novel shared attention framework that aligns task-specific knowledge acquisition with a selection module, effectively addressing CF and enlarging knowledge transfer; (7) **InsCL** (Wang et al., 2024b) dynamically replays previous data based on task similarity, using wasserstein distance and an instruction information metric to enhance replay strategies; (8) **LAMOL-g** (Sun et al., 2019) uses a global generation token to control pseudo generation; (9) **LAMOL-t** (Sun et al., 2019) uses task-specific tokens to control pseudo generation; (10) **PCLL** (Zhao et al., 2022) takes task-specific prompts as input and utilizes a CAVE model to generate pseudo-samples; (11) **DCL** (Zeng et al., 2024), the current SOTA method, is a modification of PCLL. It leverages the flexibility of Dirichlet distribution to improve the generation power of CAVE; (12) **Multi-task learning (Multi)** is commonly regarded as the upper bound for CL, where all tasks are learned concurrently.

4.4 Implementation Details

All methods are trained on a Tesla-V100 GPU, with each sequential task order taking around 12 hours. We utilize Adam optimizer (Kingma, 2014). The batch size, learning rate, and epoch number are set to 8, 5e-5, and 10 separately.

When learning the new task T_n, PCGR generates $ratio \times N_n$ pseudo-samples, where $ratio$ denotes the pseudo-sample ratio. Following previous works (Sun et al., 2020; Zhao et al., 2022; Zeng et al., 2024), we set $ratio$ to 0.2.

In PCVAE, the encoder and decoder share parameters with the LM, specifically GPT-2 (Radford et al., 2019), which is the backbone of this work. The Prior Network and Recognition Network are implemented as 2-layer MLPs (Pinkus, 1999), with the dimension of the latent variable set to 128. Hyperparameters s for SDE and c for DSVE are both set to 4. γ and τ in knowledge distillation are set to 1.0 and 2.0, separately, while λ for training PCVAE is set to 0.5. All hyperparameters in PCGR are selected using grad search.

To foster reproducibility and comparison of PCGR, we make our code publicly available at Github[1].

5 Results and Analysis

5.1 Main Results

Table 1 presents the performance comparison of PCGR and various baselines, with the upper bound included, for both scenarios. For each scenario, we report the average AP and FWT over the four corresponding task sequential orders. The detailed results of each task sequential order are provided in Appendix C.1.

- Compared to all the baseline methods, including regulation, architectural, and rehearsal methods, PCGR achieves SOTA performance in AP and FWT in both similar and dissimilar scenarios. Moreover, PCGR only lags behind the upper

[1] https://github.com/OzymandiasChen/PCGR

bound by 3.80 in the similar scenario, and it is the only method that achieves a positive average FWT in the dissimilar scenario, which implies positive knowledge transfer from previous to new tasks. These superior performances can be attributed to PCGR's considering prototypes to guide pseudo generation. By utilizing task-level statistics from prototypes, PCGR can generate more representative and diverse pseudo-samples, facilitating a better approximation of the data distribution of previous tasks when learning a new task, thereby alleviating CF.
- Compared to other generative replay methods, including LAMOL-g, LMAOL-t, PCLL, and DCL, which rely solely on a single task-specific token or prompt to guide pseudo-generation, PCGR outperforms them all. This superior performance indicates that a single task-specific token or prompt is insufficient to guide pseudo generation. In contrast, prototypes in PCGR serve as a more robust and effective mechanism for guiding pseudo-generation.
- PCGR outperforms InsCL, which replays previous real data, by a significant margin, with 4.49 higher AP in the similar scenario and 7.13 higher AP in the dissimilar scenario. This superior performance highlights the high quality of pseudo-samples in PCGR. Furthermore, this suggests that PCGR is a promising approach for mitigating CF without privacy concerns and memory overhead.

5.2 Ablation Study

We conduct ablation studies to verify the effectiveness of prototypes and Prototype Shift Estimation in pseudo-generation. (1) **w/o PSE** means Prototype Shift Estimation is not performed after the LM learns new tasks. (2) **w/o PSE, w/o Prototype** means the prototype is not considered in pseudo-generation. In this setting, the prior distribution is no longer prototype-based Gaussian distribution but Normal distribution. Corresponding VAE samples from the Normal distribution and utilizes a task-specific prompt to generate pseudo-samples.

The comparison between PCGR and w/o PSE verifies the effectiveness of Prototype Shift Estimation in Section 3.5. As shown in Table 2, Comparing w/o PSE and PCGR, w/o PSE lags behind PCGR by 2.71 on Order similar_0 and 4.21 on Order dissimilar_0. This performance decrease is because previously saved prototypes become outdated after the LM learning new tasks. In other words,

	AP ↑	FWT ↑
	Similar	
PCGR	**46.10±0.86**	**0.99±0.35**
w/o PSE	43.39±1.37	0.53±0.09
w/o PSE, w/o Prototype	41.50±2.11	0.68±0.65
	Dissimilar	
PCGR	**59.48±0.55**	**0.43±0.44**
w/o PSE	55.27±2.01	-0.24±0.71
w/o PSE, w/o Prototype	53.32±6.00	-0.63±0.62

Table 2: Ablation study on the effectiveness of prototype and PSE.

the previously saved prototypes can no longer accurately reflect the previous real data distribution in the updated feature space of LM.

The comparison between PCGR and w/o PSE, w/o Prototype verifies the effectiveness of Prototype in Section 3.3. As shown in Table 2, discarding the prototype, the performance of w/o PSE, w/o Prototype drops even further, with around 4.6 drop on Order similar_0 and 6.16 drop on Order dissimilar_0. This performance drop demonstrates that, even with the powerful generative model VAE, a single task-specific prompt is insufficient to generate representative and diverse pseudo-samples that accurately reflect the real data distribution.

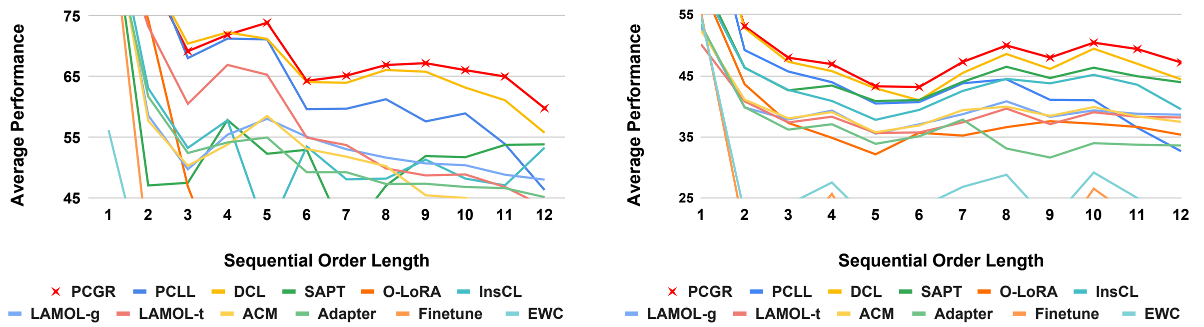
5.3 Analysis

5.3.1 Pseudo-sample Quality Evaluation

Objective Evaluation We utilize Distinction Scores (Dist-n) (Li et al., 2015) to evaluate the pseudo-sample quality of PCGR and other generative replay methods. Dist-n calculates the proportion of unique n-grams in a given corpus, where a higher distinction score indicates more corpus diversity. In generative replay, more diverse pseudo-samples are preferable, as they can better approximate previous data distribution and enhance separability among tasks in the deep feature space of LM, thereby facilitating mitigating CF. Following (Zeng et al., 2024; Zhao et al., 2022), we take $n = 1, 2, 3, 4$ in Dist-n for evaluation.

As shown in Table 3, except for Dist-4 on Order dissimilar_0, which is slightly 0.0027 inferior to the best, PCGR outperforms DCL and PCLL on all distinction scores. This superior performance shows that PCGR can generate more diverse pseudo-samples.

Human Evaluation Please refer to Appendix C.2 for human evaluation of the pseudo-samples.



(a) Similar Scenario. (Any average performance below 35 is excluded.)

(b) Dissimilar Scenario. (Any average performance below 25 is excluded.)

Figure 2: Average performance of PCGR under different sequential order length.

Method	Dist-1	Dist-2	Dist-3	Dist-4
	Order similar_0			
PCLL	0.4597	0.7154	0.8287	0.8908
DCL	0.5018	0.7638	0.8540	0.9015
PCGR	**0.5662**	**0.7928**	**0.8684**	**0.9117**
Real Data	0.6221	0.8725	0.9359	0.9634
	Order dissimilar_0			
PCLL	0.2078	0.4709	0.5916	0.6492
DCL	0.2352	0.5468	0.6887	**0.7482**
PCGR	**0.2480**	**0.5503**	**0.6892**	0.7455
Real Data	0.2463	0.5713	0.6930	0.7396

Table 3: Distinction scores for pseudo-samples of PCGR and the generative replay baselines.

Pseudo-sample Ratio	LAMOL-t	PCLL	DCL	PCGR
	Order similar_0			
0.05	33.863	30.994	38.727	**41.991**
0.1	38.103	32.266	44.136	**44.587**
0.2	38.187	32.665	44.421	**47.192**
0.5	39.305	33.658	45.370	**48.202**
0.8	38.608	37.028	45.826	**48.756**
Upper Bound	49.70			
	Order dissimilar_0			
0.05	36.125	38.563	47.222	**47.252**
0.1	40.366	41.454	**54.810**	54.565
0.2	48.025	46.313	55.743	**59.792**
0.5	40.269	50.633	61.470	**65.077**
0.8	40.751	52.887	63.562	**66.038**
Upper Bound	70.98			

Table 4: Average Performance of different pseudo-sample ratios on PCGR and the generative replay baselines.

5.3.2 Influence of Pseudo-sample Ratio

We conduct experiments of different pseudo-sample ratios on PCGR to evaluate the impact of pseudo-sample number in PCGR. As shown in Table 4, except for being inferior to DCL by 0.16 on the pseudo-sample ratio of 0.1 in Order dissimilar_0, PCGR constantly outperforms other generative replay baselines. This demonstrates PCGR's superior power in generating representative pseudo-samples.

Moreover, the performance of LAMOL-t stops increasing after the pseudo-sample ratio reaches 0.5 in Order similar_0 and 0.2 in Order dissimilar_0. In contrast, the performance of PCGR continues to increase as the pseudo-sample ratio increases. This phenomenon occurs because a single task-specific token cannot fully capture the scale of task-level data distribution. As more low-quality pseudo-samples are included, LAMOL-t's performance may degrade.

5.3.3 Impact of Task Sequential Order Length

We test the performance of CL methods after learning each task to evaluate the impact of task sequential order length. As shown in Figure 2, PCGR outperforms the baselines on different task sequential order lengths. Notably, as the sequential order length increases, PCGR exceeds the previous SOTA method, DCL, by an even greater margin. This superior performance indicates that PCGR remains robust with the introduction of additional new tasks, highlighting its effectiveness in mitigating CF, which is the ultimate goal of CL.

5.3.4 Scalability of PCGR

We conduct experiments on GPT-2 backbones of different sizes to verify PCGR's scalability. As shown in Table 5, PCGR consistently outperforms LAMOL-t, PCLL, and DCL across GPT-2 (124M), GPT-2-medium (355M), and GPT-2-large (774M). It demonstrates the scalability of PCGR and this scalability is particularly desirable in the current era of large language models.

Backbone	LAMOL-t	PCLL	DCL	PCGR
	Order similar_0			
GPT-2 (124M)	38.190	32.670	44.420	**47.190**
GPT-2-medium (355M)	40.955	35.081	45.394	**46.831**
GPT-2-large (774M)	41.743	37.644	44.809	**45.038**
	Order dissimilar_0			
GPT-2 (124M)	43.576	46.313	55.743	**59.792**
GPT-2-medium (355M)	52.865	49.624	62.036	**62.568**
GPT-2-large (774M)	58.200	51.144	60.621	**61.588**

Table 5: Comparison of PCGR and baselines based on different sizes of backbones.

6 Conclusions

This paper proposes PCGR, a rehearsal method that incorporates task-level statistics to enhance generative replay. In PCGR, task-level embedding statistics are stored as prototypes for each old task. When new task comes, PCGR utilizes samples drawn from task-specific prototype-based distributions to generate pseudo-samples that are both representative and diverse enough to reflect the previous real data distribution. Experiments show that PCGR achieves SOTA performance, demonstrates greater robustness to sequential order length compared to other baselines, and can be scaled to larger backbones.

7 Limitations

Our method is the generative replay method, which typically tunes the whole language model. Given the large number of learnable parameters, the time required for generative replay is substantial, as is the case with our PCGR. In the future, we can explore hybridizing our PCGR with architectural methods to reduce training costs.

References

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

Eden Belouadah and Adrian Popescu. 2019. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 583–592.

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. 2021. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Andy Cedilnik, and Kyu-Young Kim. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. *arXiv preprint arXiv:1909.05358*.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547.

Yung-Sung Chuang, Shang-Yu Su, and Yun-Nung Chen. 2020. Lifelong language knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2914–2924.

Binzong Geng, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang. 2021. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 517–523.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao Tao, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International conference on learning representations*.

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

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

Zixuan Ke, Bing Liu, Wenhan Xiong, Asli Celikyilmaz, and Haoran Li. 2023. Sub-network discovery and soft-masking for continual learning of mixed tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15090–15107.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

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

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

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul A Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7452–7467.

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.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. 2023. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50.

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020a. Continual learning for natural language generation in task-oriented dialog systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3461–3474.

Fei Mi, Lingjing Kong, Tao Lin, Kaicheng Yu, and Boi Faltings. 2020b. Generalized class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 240–241.

Martin Mundt, Yongwon Hong, Iuliia Pliushch, and Visvanathan Ramesh. 2023. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.

Allan Pinkus. 1999. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8689–8696.

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.

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

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Yuwen Tan, Qinhao Zhou, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. 2024. Semantically-shifted incremental adapter-tuning is a continual vitransformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23252–23262.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024a. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. 2023. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024b. Inscl: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 663–677.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *arXiv preprint arXiv:1905.08743*.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan Fang Li, Guilin Qi, and Gholamreza Haffari. 2022. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations 2022*.

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*.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. 2020. Semantic drift compensation for class-incremental learning. in 2020 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6980–6989.

Min Zeng, Yisen Wang, and Yuan Luo. 2019. Dirichlet latent variable hierarchical recurrent encoder-decoder in dialogue generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1267–1272.

Min Zeng, Haiqin Yang, Wei Xue, Qifeng Liu, and Yike Guo. 2024. Dirichlet continual learning: Tackling catastrophic forgetting in nlp. In *The 40th Conference on Uncertainty in Artificial Intelligence*.

Mengyao Zhai, Lei Chen, Jiawei He, Megha Nawhal, Frederick Tung, and Greg Mori. 2020. Piggyback gan: Efficient lifelong learning for image conditioned generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 397–413. Springer.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022. Continual sequence generation with adaptive compositional modules. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3653–3667.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11641–11661.

Yingxiu Zhao, Yinhe Zheng, Zhiliang Tian, Chang Gao, Jian Sun, and Nevin L Zhang. 2022. Prompt conditioned vae: Enhancing generative replay for lifelong learning in task-oriented dialogue. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11153–11169.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. 2024. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. 2021a. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34:14306–14318.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. 2021b. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880.

A Additional Methodology Details

We provide detail model architecture and data flow of PCVAE in Figure 3.

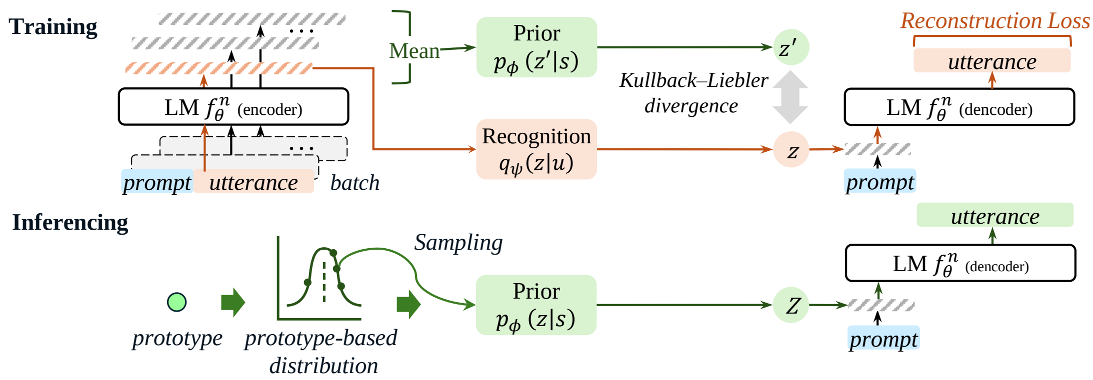


Figure 3: Model architecture and data flow of PCVAE.

B Experiment Details

B.1 Task and Dataset Details

To evaluate the CL methods, we take into consideration of five task patterns:

Programming Language Generation involves automatically generating code snippets or full programs from natural language descriptions. It aims to bridge the gap between human intentions and machine code. For example, in a SQL language generation task, the natural language description is "`the table has columns rank , name , team , goals , appearances , minutes played and key words max , min , count , sum , avg , = , > , < , op , select , where , and , col , table , caption , page , section , op , cond , question , agg , aggops , condops - - what rank has a team of roma ?`", corresponding SQL code is "`select rank from table where team = roma`".

In this paper, we take into consideration of a SQL language generation task of WikiSQL (Zhong et al., 2017).

Intent Detection focuses on identifying the purpose behind a user's input, commonly used in chatbots and virtual assistants. It classifies user queries into predefined categories. For example, given user input "`I believe someone is using my card without my agreement!`", corresponding intent is "`compromised_card`".

In this paper, we task into consideration of three intent detection tasks, including banking (Casanueva et al., 2020), top_split2 (Gupta et al., 2018), and clinic (Larson et al., 2019).

Dialogue State Tracking (DST) involves maintaining the context and state of a conversation in dialog systems. It keeps track of user intents, entities, and any relevant information throughout the interaction. For example, in a restaurant booking system, if the user says "`I want to book a table for two at Italian cuisine`", the DST would update the state to reflect the intent "`booking`", "`number_of_people=two`", and "`cuisine_type=Italian`".

In this paper, we task into consideration of three DST tasks, including dst8 (Rastogi et al., 2020), atis_slot (Hemphill et al., 1990), and mit_movie_eng[2].

Natural Language Generation (NLG) refers to the process of converting structured data into human-readable text. For example, given structured data "`Temperature=75, Condition=Sunny`", an NLG system might produce the output "`The weather today is sunny with a temperature of 75 degrees Fahrenheit.`".

In this paper, we take into consideration of 12 NLG tasks, including tm20_flight_nlg, tm20_sport_nlg, tm20_hotel_nlg, tm20_restaurant_nlg, and tm20_music_nlg from (Byrne et al., 2019), sgd_events_nlg from (Rastogi et al., 2020), tm19_movie_nlg from (Byrne et al., 2019), e2enlg from (Novikova et al., 2017), and rnnlg_restaurant, rnnlg_tv, rnnlg_hotel, and rnnlg_laptop from (Wen et al., 2015).

Question Answering (QA) aims at answering the questions given the context. For example, give context "`the stock pot should be chilled and the solid lump of dripping which settles when chilled should be scraped clean and re-chilled for future use .`" and question "`what settles ?`", the answer should be "`the solid lump of dripping`".

In this paper, we take into consideration of one

[2]`https://sls.csail.mit.edu/downloads/`

Task Sequential Order	Scenario	Task Sequence
Order similar_0	Similar	rnnlg_tv, tm20_music_nlg, tm20_flight_nlg, tm20_sport_nlg, e2enlg, tm20_restaurant_nlg, rnnlg_hotel, rnnlg_laptop, tm19_movie_nlg, rnnlg_restaurant, sgd_events_nlg, tm20_hotel_nlg
Order similar_1	Similar	tm20_music_nlg, sgd_events_nlg, e2enlg, rnnlg_restaurant, rnnlg_hotel, tm20_sport_nlg, rnnlg_laptop, rnnlg_tv, tm20_restaurant_nlg, tm20_hotel_nlg, tm19_movie_nlg, tm20_flight_nlg
Order similar_2	Similar	rnnlg_laptop, sgd_events_nlg, e2enlg, rnnlg_restaurant, tm20_sport_nlg, rnnlg_hotel, tm20_restaurant_nlg, rnnlg_tv, tm19_movie_nlg, tm20_flight_nlg, tm20_hotel_nlg, tm20_music_nlg
Order similar_3	Similar	tm20_flight_nlg, tm20_sport_nlg, e2enlg, tm20_hotel_nlg, sgd_events_nlg, tm19_movie_nlg, rnnlg_restaurant, tm20_restaurant_nlg, rnnlg_tv, rnnlg_hotel, tm20_music_nlg, rnnlg_laptop
Order dissimilar_0	Dissimilar	top_split2, WikiSQL, tm20_sport_nlg, dstc8, clinc, e2enlg, banking, atis_slot, rnnlg_laptop, mit_movie_eng, srl, sgd_events_nlg
Order dissimilar_1	Dissimilar	dstc8, tm20_sport_nlg, banking, top_split2, e2enlg, atis_slot, mit_movie_eng, sgd_events_nlg, clinc, rnnlg_laptop, WikiSQL, srl
Order dissimilar_2	Dissimilar	atis_slot, banking, sgd_events_nlg, tm20_sport_nlg, e2enlg, WikiSQL, srl, top_split2, clinc, mit_movie_eng, rnnlg_laptop, dstc8
Order dissimilar_3	Dissimilar	WikiSQL, dstc8, srl, atis_slot, top_split2, e2enlg, tm20_sport_nlg, banking, mit_movie_eng, rnnlg_laptop, clinc, sgd_events_nlg

Table 6: The detail of eight task sequential orders across two scenarios.

QA dataset, srl (He et al., 2015).

For each task, following (Wang et al., 2023; Zhao et al., 2024), we randomly select 3,000 samples for training and 256 samples for evaluating and testing.

B.2 Task Sequential Orders

We provide the details of eight task sequential orders across two scenarios in Table 6.

B.3 Metric Details

The evaluation metrics for each task are as follows:
Programming Language Generation we employ the Exact Match (EM), which assesses the percentage of generated code snippets that exactly match the expected output, ensuring precision in code generation.
Intent Detection We utilize the Exact Match metric to determine the accuracy of identifying the user's intent from their input, reflecting the system's ability to understand user queries correctly.
Dialogue State Tracking we adopt the Joint Goal Accuracy (JGA) (Wu et al., 2019), where the intent keyword and the corresponding value should exactly match with the gold. It measures the system's capability to maintain and track multiple dialogue goals simultaneously, thus ensuring a coherent interaction.
Natural Language Generation We use BLEU score (Papineni et al., 2002), which measures the overlap between the generated text and one or more reference texts based on n-grams.
Question Answering We also use the BLEU score.

C More Results and Analysis

C.1 Detail Results on each task sequential order

We provide detailed results of each task sequential order for the similar and dissimilar scenarios. As shown in Table 7, our PCGR outperforms the baselines on AP regardless of the task sequential order.

C.2 Human Evaluation of Pseudo-samples

We present pseudo-samples generated by PCGR and other generative replay baselines, including DCL, PCLL, and LAMOL-t, as shown in Tables

Method	Order similar_0		Order similar_1		Order similar_2		Order similar_3	
	AP	FWT	AP	FWT	AP	FWT	AP	FWT
Finetune (Yogatama et al., 2019)	16.00	-9.41	16.22	-9.96	18.82	-9.71	22.48	-10.19
EWC (Kirkpatrick et al., 2017)	20.13	-10.07	19.77	-10.39	21.21	-10.43	28.13	-11.40
Adapter (Madotto et al., 2020)	33.60	N/A	33.60	N/A	33.60	N/A	33.60	N/A
ACM (Zhang et al., 2022)	37.47	-9.83	37.74	-10.07	36.79	-9.76	36.47	-10.24
O-LoRA (Wang et al., 2023)	35.36	-8.07	31.08	-8.99	33.46	-8.60	35.79	-8.94
SAPT (Zhao et al., 2024)	43.95	-5.62	43.53	-5.78	44.01	-5.02	43.73	-5.44
InsCL (Wang et al., 2024b)	39.53	-4.76	43.80	-5.36	40.10	-4.80	43.35	-4.58
LAMOL-g (Sun et al., 2019)	38.64	-9.06	35.34	-9.25	36.29	-9.91	37.60	-9.91
LAMOL-t (Sun et al., 2019)	38.19	-10.13	37.10	-10.46	38.18	-8.99	37.66	-9.65
PCLL (Zhao et al., 2022)	32.67	0.86	31.75	0.64	37.84	0.60	32.77	0.45
DCL (Zeng et al., 2024)	44.42	0.59	45.14	1.02	43.13	0.61	42.29	0.68
PCGR (Ours)	**47.19**	**1.10**	**46.37**	**1.07**	**45.33**	**1.29**	**45.49**	**0.49**
Multi (Upper Bound)	49.70	N/A	49.70	N/A	49.70	N/A	49.70	N/A

Method	Order dissimilar_0		Order dissimilar_1		Order dissimilar_2		Order dissimilar_3	
	AP	FWT	AP	FWT	AP	FWT	AP	FWT
Finetune (Yogatama et al., 2019)	13.67	-6.49	13.46	-6.73	13.67	-6.33	10.63	-6.45
EWC (Kirkpatrick et al., 2017)	20.69	-31.10	17.49	-9.24	18.62	-7.02	19.9	-30.74
Adapter (Madotto et al., 2020)	45.16	N/A	45.16	N/A	45.16	N/A	45.16	N/A
ACM (Zhang et al., 2022)	41.63	-7.55	39.43	-8.35	39.96	-9.24	45.40	-7.80
SAPT (Zhao et al., 2024)	53.84	-9.43	56.23	-9.12	55.43	-8.88	56.08	-10.30
O-LoRA (Wang et al., 2023)	18.88	-25.31	17.80	-23.44	19.60	-22.72	21.01	-22.90
InsCL (Wang et al., 2024b)	53.29	-9.02	51.22	-10.12	49.57	-10.76	55.32	-9.18
LAMOL-g (Sun et al., 2019)	48.03	-5.18	43.23	-6.68	43.79	-6.17	38.24	-6.79
LAMOL-t (Sun et al., 2019)	43.58	-7.07	46.27	-6.68	45.91	-7.62	42.48	-6.72
PCLL (Zhao et al., 2022)	46.31	**1.19**	42.43	-0.24	42.19	-1	44.83	-0.92
DCL (Zeng et al., 2024)	55.74	-0.50	57.09	-0.46	57.53	-0.07	55.38	-1.15
PCGR (Ours)	**59.79**	0.15	**59.48**	**0.11**	**59.94**	**0.40**	**58.72**	**1.07**
Multi (Upper Bound)	70.98	N/A	70.98	N/A	70.98	N/A	70.98	N/A

Table 7: Comparison results of PCGR and the baselines on on each task sequential order.

Invalid Type	Description
$x - y$ mismatch	The output y does not semantically align with the input x. For example, in the seventh pseudo-sample from PCGR, the correct output for the input "`Can you tell me what currencies I can use to top up my account?`" should be "`supported_cards_and_currencies`", not "`op_up_by_cash_or_cheque`".
corpus duplication	The pseudo-sample is identical to previous pseudo-samples, indicating that the generative replay method fails to produce sufficiently diverse data. For example, the fifth and tenth pseudo-samples in LAMOL-t are identical to the third pseudo-sample in LAMOL-t.
corpus incoherent	There are syntax or semantic errors in the input x or output y. For example, in the fifth pseudo-sample of DCL, the input x "`How do I track the card I received?`" is problematic because if someone has already received the card, tracking it is unnecessary, as they already possess it.

Table 8: Invalid pseudo-sample type.

9, 10, 11, and 12 for human evaluation of pseudo-sample quality. All models were trained under the Order dissimilar_0. To ensure a fair comparison, we selected the first ten generated pseudo-samples from the same task: "banking", which is an intent detection task. Additionally, we define three types of invalid pseudo-samples, as shown in Table 8.

When comparing the pseudo-samples generated by PCGR with those from other generative replay baselines, we find that the pseudo-samples generated by PCGR are more representative and diverse, contributing to its state-of-the-art performance:

- **Representativeness of Pseudo-samples**: PCGR has only 2 out of 10 invalid pseudo-samples, while DCL, PCLL, and LAMOL-t have 4 out of 10, 4 out of 10, and 7 out of 10 invalid samples, respectively. This lower ratio of invalid pseudo-samples in PCGR indicates that the prototype-guided pseudo-samples are more representative and better semantically aligned with real data. In contrast, the pseudo-samples from PCLL, LAMOL-t, and DCL exhibit more noise, leading to their inferior performance.
- **Diversity of Pseudo-samples**: (1) Regardless of validity, 3 out of 10 pseudo-samples in PCLL share the same intent type, and 3 out of 10 in LAMOL-t have issues with corpus duplication. In contrast, both PCGR and DCL have only 1 out of 10 pseudo-samples sharing the same intent type. (2) Additionally, 2 out of 10 pseudo-samples in DCL start with "How long," whereas PCGR does not have this issue. Both (1) and (2) indicate that PCGR can generate more diverse pseudo-samples than the other generative replay baselines.

Index	Input x	Output y
1	I'm waiting for my card to arrive.	card_arrival
2	I would like to have a virtual card.	getting_virtual_card
3	***Where's my card PIN?***	***get_physical_card***
4	What's the reason I can't top up?	top_up_failed
5	*There is a charge showing on my account.*	card_payment_fee_charged
6	How do I complete the ID check?	verify_my_identity
7	***Can you tell me what currencies I can use to top up my account?***	***top_up_by_cash_or_cheque***
8	Hi, I would like to make a card payment that is pending.	pending_card_payment
9	What are the charges I see on my statement?	card_payment_fee_charged
10	Please explain the exchange rate.	exchange_rate

Index	Explanation for the Invalid Pseudo-sample
3	Invalid type: $x - y$ mismatch Invalid reason: The input x has nothing to do with 'physical card'.
7	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'supported_cards_and_currencies'.

Table 9: Pseudo-samples from PCGR. Invalid pseudo-samples are highlighted in ***italic bold*** with explanation provided.

Index	Input x	Output y
1	***Why did my payment go through?***	***pending_card_payment***
2	I got my new card, but it's not working.	card_not_working
3	Transfer could not be completed.	failed_transfer
4	***A payment I made to get my card is showing up twice.***	***card_payment_not_recognised***
5	***How do I track the card I received?***	***card_arrival***
6	I was charged twice	transaction_charged_twice
7	***I bought something with my card, but the money didn't go through***	***card_payment_not_recognised***
8	How long is the transfer to show up in my account?	transfer_timing
9	How long does it take to verify my identity?	verify_my_identity
10	For topping up by card I have been charged a fee.	top_up_by_card_charge

Index	Explanation for the Invalid Pseudo-sample
1	Invalid type: $x - y$ mismatch Invalid reason: The input x indicates that the payment was successful and not intended to be processed. Therefore, the payment is confirmed as successful rather than pending. As a result, there is a mismatch between input x and output y.
4	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'transaction_charged_twice'.
5	Invalid type: corpus incoherent Invalid Reason: It is clear that if someone has already received the card, tracking it is unnecessary since they already have possession of it. As a result, there is a semantic error, which makes the pseudo-sample incoherent.
7	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'declined_card_payment'.

Table 10: Pseudo-samples from DCL. Invalid pseudo-samples are highlighted in ***italic bold*** with explanation provided.

Index	Input x	Output y
1	***I have a bank transfer that was approved.***	***transfer_not_received_by_recipient***
2	Can I have a second card issued to another family member?	getting_spare_card
3	I do not recognize this card payment.	card_payment_not_recognised
4	***How do I get my card refund?***	***card_payment_not_recognised***
5	In which countries does your app work?	country_support
6	I can't log in with the passcode being rejected.	passcode_forgotten
7	***How do I get my card renewed?***	***activate_my_card***
8	***I have a charge for using my card.***	***card_payment_not_recognised***
9	Is there any way I can get a second card issued to another family member?	getting_spare_card
10	Can I transfer using a bank transfer?	transfer_into_account

Index	Explanation for the Invalid Pseudo-sample
1	Invalid type: $x - y$ mismatch Invalid Reason: The input does not indicate that the transfer was not received.
4	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'request_refund'.
7	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'card_about_to_expire'.
8	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'card_payment_fee_charged'.

Table 11: Pseudo-samples from PCLL. Invalid pseudo-samples are highlighted in ***italic bold*** with explanation provided.

Index	Input x	Output y
1	Where can I change my pin?	change_pin
2	***I got a refund and I don't remember where I ordered it from***	***Refund_not_showing_up***
3	What currencies can I use to get money?	fiat_currency_support
4	***I got a refund and I don't remember where I ordered it from***	***Refund_not_showing_up***
5	***What currencies can I use to get money?***	***fiat_currency_support***
6	***I see a pending payment on my card. I thought I had withdrawn funds earlier.***	***pending_card_payment***
7	***I got a refund and I don't remember where I ordered it from***	***Refund_not_showing_up***
8	***Why am I getting an error code when I make a transfer?***	***beneficiary_not_allowed***
9	How long does it take to get an actual card?	card_delivery_estimate
10	***What currencies can I use to get money?***	***fiat_currency_support***

Index	Explanation for the Invalid Pseudo-sample
2	Invalid type: $x - y$ mismatch Invalid reason: It indicates an unrecognized refund in the input x .
4	Invalid type: $x - y$ mismatch, corpus duplication Invalid reason: The pseudo-sample is identical to the second pseudo-sample.
5	Invalid type: corpus duplication Invalid reason: The pseudo-sample is identical to the third pseudo-sample.
6	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'dechashlined_cash_withdrawal'.
7	Invalid type: $x - y$ mismatch, corpus duplication Invalid reason: The pseudo-sample is identical to the second pseudo-sample.
8	Invalid type: $x - y$ mismatch Invalid reason: The output y should be 'failed_transfer'.
10	Invalid type: corpus duplication Invalid reason: The pseudo-sample is identical to the third pseudo-sample.

Table 12: Pseudo-samples from LAMOL-t. Invalid pseudo-samples are highlighted in ***italic bold*** with explanation provided.