t-DGR: A Trajectory-Based Deep Generative Replay Method for Continual Learning in Decision Making

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Deep generative replay has emerged as a promising approach for continual learning in decision-making tasks. This approach addresses the problem of catastrophic forgetting by leveraging the generation of trajectories from previously encountered tasks to augment the current dataset. However, existing deep generative replay methods for continual learning rely on autoregressive models, which suffer from compounding errors in the generated trajectories. In this paper, we propose a simple, scalable, and non-autoregressive method for continual learning in decision-making tasks using a diffusion model that generates task samples conditioned on the trajectory timestep. We evaluate our method on Continual World benchmarks and find that our approach achieves state-of-the-art performance on the average success rate metric compared to other continual learning methods.

1 Introduction

Continual learning, also known as lifelong learning, is a critical challenge in the advancement of general artificial intelligence, as it enables models to learn from a continuous stream of data encompassing various tasks, rather than having access to all data at once [24]. However, a major challenge in continual learning is the phenomenon of catastrophic forgetting, where previously learned skills are lost when attempting to learn new tasks [18].

To mitigate catastrophic forgetting, replay methods have been proposed, which involve saving data from previous tasks and replaying it to the learner during the learning of future tasks. This approach mimics how humans actively prevent forgetting by reviewing material for tests and replaying memories in dreams. However, storing data from previous tasks requires significant storage space and becomes computationally infeasible as the number of tasks increases.

In the field of cognitive neuroscience, the Complementary Learning Systems theory offers insights into how the human brain manages memory. This theory suggests that the brain employs two complementary learning systems: a fast-learning episodic system and a slow-learning semantic system [17, 14, 16]. The hippocampus serves as the episodic system, responsible for storing specific memories of unique events, while the neocortex functions as the semantic system, extracting general knowledge from episodic memories and organizing it into abstract representations [23].

Drawing inspiration from the human brain, deep generative replay (DGR) addresses the catastrophic forgetting issue in decision-making tasks by using a generative model as the hippocampus to generate trajectories from past tasks and replay them to the learner which acts as the neocortex (Figure 2) [26]. The time-series nature of trajectories in decision-making tasks sets it apart from continual supervised learning, as each timestep of the trajectory requires sufficient replay. In supervised learning, the

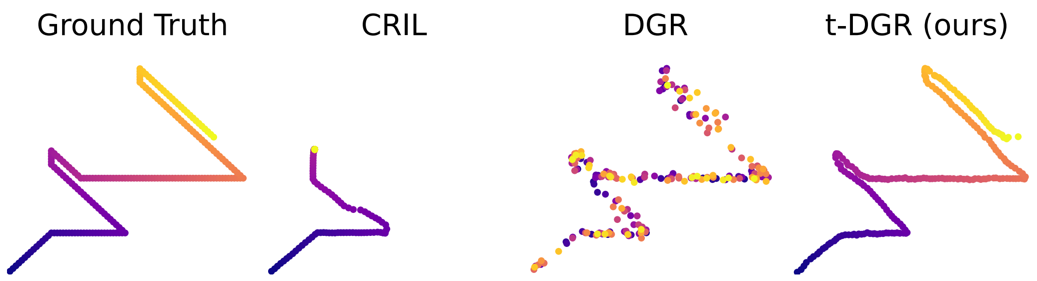

Figure 1: We compare three generative methods for imitating an agent's movement in a continuous 2D plane with Gaussian noise. Our objective is to replicate the ground truth path, which transitions from darker to lighter colors. The autoregressive method (CRIL) encounters a challenge at the first sharp turn as nearby points move in opposing directions. Once the autoregressive method deviates off course, it never recovers and compromises the remaining trajectory. In contrast, sampling individual state observations i.i.d. without considering the temporal nature of trajectories (DGR) leads to a fragmented path with numerous gaps. Our proposed method t-DGR samples individual state observations conditioned on the trajectory timestep. By doing so, t-DGR successfully avoids the pitfalls of CRIL and DGR, ensuring a more accurate replication of the desired trajectory.

learner's performance is not significantly affected if it performs poorly on a small subset of the data. However, in decision-making tasks, poor performance on any part of the trajectory can severely impact the overall performance. Therefore, it is crucial to generate state-action pairs that accurately represent the distribution found in trajectories. Furthermore, the high-dimensional distribution space of trajectories makes it computationally infeasible to generate complete trajectories all at once.

Existing DGR methods adopt either the generation of individual state observations i.i.d. without considering the temporal nature of trajectories or autoregressive trajectory generation. Autoregressive approaches generate the next state(s) in a trajectory by modeling the conditional probability of the next state(s) given the previously generated state(s). However, autoregressive methods suffer from compounding errors in the generated trajectories. On the other hand, generating individual state observations i.i.d. leads to a higher sample complexity compared to generating entire trajectories, which becomes significant when replay time is limited.

To address the issues in current DGR methods, we propose a simple, scalable, and non-autoregressive trajectory-based DGR method. We define a generated trajectory as temporally coherent if the transitions from one state to the next appear realistic (refer to Section 3.3 for a formal definition). Given that current decision-making methods are trained on state-action pairs, we do not require trajectories to exhibit temporal coherence. Instead, our focus is on ensuring an equal number of samples generated at each timestep of the trajectory to accurately represent the distribution found in trajectories. To achieve equal sample coverage at each timestep, we train our generator to produce state observations conditioned on the trajectory timestep, and then sample from the generator conditioned on each timestep of the trajectory. The intuition behind our method is illustrated in Figure 1.

To evaluate the effectiveness of our proposed method, t-DGR, we conducted experiments on the Continual World benchmarks CW10 and CW20 [29] using imitation learning. Our results indicate that t-DGR achieves state-of-the-art performance in terms of average success rate when compared to other top continual learning methods.

2 Related Work

This section provides an overview of existing continual learning methods within the context of "General Continual Learning", with a particular focus on pseudo-rehearsal methods.

2.1 Continual Learning in the Real World

As the field of continual learning continues to grow, there is an increasing emphasis on developing methods that can be effectively applied in real-world scenarios [28, 3, 4, 10, 27]. The concept of "General Continual Learning" was introduced by Buzzega et al. [5] to address certain properties of the real world that are often overlooked or ignored by existing continual learning methods. Specifically, two important properties, bounded memory and blurry task boundaries, are emphasized in this work. Bounded memory refers to the requirement that the memory footprint of a continual learning method should remain bounded throughout the entire lifespan of the learning agent. This property is crucial to ensure practicality and efficiency in real-world scenarios. Additionally, blurry task boundaries highlight the challenge of training on tasks that are intertwined, without clear delineation of when one task ends and another begins. Many existing methods fail to account for this characteristic, which is common in real-world learning scenarios. While there are other significant properties associated with continual learning in the real world, this study focuses on the often-neglected aspects of bounded memory and blurry task boundaries. By addressing these properties, we aim to develop methods that are more robust and applicable in practical settings.

2.2 Continual Learning Methods

Continual learning methods for decision-making tasks can be categorized into three main categories.

Regularization Regularization methods in continual learning focus on incorporating constraints during model training to promote the retention of past knowledge. One simple approach is to include an L_2 penalty in the loss function. Elastic Weight Consolidation (EWC) builds upon this idea by assigning weights to parameters based on their importance for previous tasks using the Fisher information matrix [13]. MAS measures the sensitivity of parameter changes on the model's output, prioritizing the retention of parameters with a larger effect [2]. VCL leverages variational inference to minimize the Kullback-Leibler divergence between the current and prior parameter distributions [22]. Progress and Compress learns new tasks using a separate model and subsequently distills this knowledge into the main model while safeguarding the previously acquired knowledge [25]. However, it is important to note that regularization methods may struggle with blurry task boundaries as they rely on knowledge of task endpoints to apply regularization techniques effectively. In our experiments, EWC was chosen as the representative regularization method based on its performance in the original Continual World experiments [29].

Architecture-based Methods Architecture-based methods aim to maintain distinct sets of parameters for each task, ensuring that future learning does not interfere with the knowledge acquired from previous tasks. Packnet [15], UCL [1], and AGS-CL [11] all safeguard previous task information in a neural network by identifying important parameters and freeing up less important parameters for future learning. Identification of important parameters can be done through iterative pruning (Packnet), parameter uncertainty (UCL), and activation value (AGS-CL). However, a drawback of parameter isolation methods is that each task requires its own set of parameters, which may eventually exhaust the available parameters for new tasks and necessitate a dynamically expanding network without bounded memory [30]. Additionally, parameter isolation methods require training on a single task at a time to prune and isolate parameters, preventing concurrent learning from multiple interwoven tasks. In our experiments, PackNet was selected as the representative architecture-based method based on its performance in the original Continual World experiments [29].

Pseudo-rehearsal Methods Pseudo-rehearsal methods mitigate the forgetting of previous tasks by generating synthetic samples from past tasks and replaying them to the learner. Deep generative replay (DGR) (Figure 2) utilizes a generative model, such as generative adversarial networks [7], variational autoencoders [12], or diffusion models [9], to generate the synthetic samples. Originally, deep generative replay was proposed to address continual supervised learning problems, where the generator only needed to generate single data point samples [26]. However, in decision-making tasks, expert demonstrations consist of trajectories (time-series) with a significantly higher-dimensional distribution space.

One existing DGR method generates individual state observations i.i.d. instead of entire trajectories. However, this approach leads to a higher sample complexity compared to generating entire trajectories. The sample complexity of generating enough individual state observations i.i.d. to cover every portion of the trajectory m times can be described using the Double Dixie Cup problem [20]. For trajectories of length n, it takes an average of $\Theta(n \log n + mn \log \log n)$ i.i.d. samples to ensure at least m samples for each timestep. In scenarios with limited replay time (small m) and long trajectories (large n) the sample complexity can be approximated as $\Theta(n \log n)$ using the Coupon Collector's problem

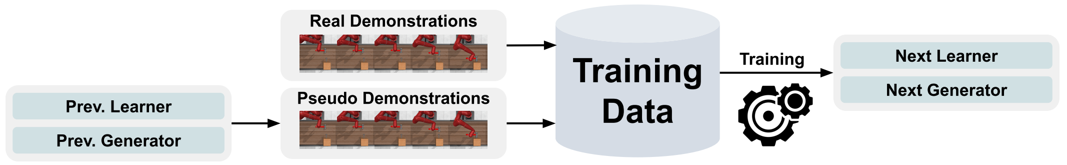

Figure 2: The deep generative replay paradigm. The algorithm learns to generate trajectories from past tasks to augment real trajectories from the current task in order to mitigate catastrophic forgetting. Both the generator and policy model are updated with this augmented dataset.

[19]. The additional $\Theta(\log n)$ factor reduces the likelihood of achieving complete sample coverage of the trajectory when the number of replays or replay time is limited, especially considering the computationally expensive nature of current generative methods. Furthermore, there is a risk that the generator assigns different probabilities to each timestep of the trajectory, leading to a selective focus on certain timesteps rather than equal representation across the trajectory.

Another existing DGR method is autoregressive trajectory generation. In the existing autoregressive method, CRIL, a generator is used to generate samples of the initial state, and a dynamics model predicts the next state based on the current state and action [6]. However, even with a dynamics model accuracy of 99% and a 1% probability of deviating from the desired trajectory, the probability of an autoregressively generated trajectory going off course is $1 - 0.99^n$, where n denotes the trajectory length. With a trajectory length of $n = 200$ (as used in our experiments), the probability of an autoregressively generated trajectory going off course is $1 - 0.99^{200} = 0.87$. This example demonstrates how the issue of compounding error leads to a high probability of failure, even with a highly accurate dynamics model.

In our experiments, t-DGR is evaluated against all existing pseudo-rehearsal methods to assess how well t-DGR addresses the limitations of those methods.

3 Background

This section introduces notation and the formulation of the continual imitation learning problem that we use in this paper.

3.1 Imitation Learning

Imitation learning algorithms aim to learn a policy π_θ parameterized by θ by imitating a set of expert demonstrations $D = \{\tau_i\}_{i=1\ldots M}$. Each trajectory τ_i consists of a sequence of state-action pairs $\{(s_j, a_j)\}_{j=1\ldots|\tau_i|}$ where $|\tau_i|$ is the length of the trajectory. Each trajectory comes from a task $\mathcal{T}$ which is a Markov decision process that can be represented as a tuple $\langle S, A, T, \rho_0\rangle$ with state space S, action space A, transition dynamics $T : S \times A \times S \to [0, 1]$, and initial state distribution ρ_0. Various algorithms exist for imitation learning, including behavioral cloning, GAIL [8], and inverse reinforcement learning [21]. In this work, we use behavioral cloning where the objective can be formulated as minimizing the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a\sim D}\left[\left\|\pi_\theta(s) - a\right\|_2^2\right] \tag{1}$$

where the state and action spaces are continuous.

3.2 Continual Imitation Learning

In the basic formulation most common in the field today, continual imitation learning involves sequentially solving multiple tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$. When solving for task $\mathcal{T}_i$, the learner only gets data from task $\mathcal{T}_i$ and can not access data for any other task. In a more general scenario, certain tasks may have overlapping boundaries, allowing the learner to encounter training data from multiple tasks during certain phases of training. The learner receives a continuous stream of training data in the form of trajectories $\tau_1, \tau_2, \tau_3, \ldots$ from the environment, where each trajectory τ corresponds to one of the N tasks. However, the learner can only access a limited contiguous portion of this stream at any given time.

Let s_i be the success rate of task $\mathcal{T}_i$ after training on all N tasks. The continual imitation learning objective is defined as maximizing the average success rate over all tasks:

$$S = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{2}$$

The primary issue that arises from the continual learning problem formulation is the problem of catastrophic forgetting where previously learned skills are forgotten when training on a new task.

3.3 Notation

Deep generative replay involves training two models: a generator G_β parameterized by β and a learner π_θ parameterized by θ. We define $G_\beta^{(i)}$ as the generator trained on tasks $\mathcal{T}_1 \dots \mathcal{T}_i$ and capable of generating data samples from tasks $\mathcal{T}_1 \dots \mathcal{T}_i$. Similarly, $\pi_\theta^{(i)}$ represents the learner trained on tasks $\mathcal{T}_1 \dots \mathcal{T}_i$ and able to solve tasks $\mathcal{T}_1 \dots \mathcal{T}_i$.

A sequence of state observations $(s_1, s_2, \dots, s_{n-1}, s_n)$ is **temporally coherent** if $\forall 1 \leq i < n, \exists a \in A : T(s_i, a, s_{i+1}) > \varepsilon$, where $0 < \varepsilon < 1$ is a small constant representing a threshold for negligible probabilities.

4 Method

Our proposed method, t-DGR, tackles the challenge of generating long trajectories by training a generator, denoted as $G_\beta(j)$, which is conditioned on the trajectory timestep j to generate state observations. The algorithm begins by initializing the task index, replay ratio, generator model, learner model, and learning rates (Line 1). The replay ratio, denoted as $0 \leq r < 1$, determines the percentage of training samples seen by the learner that are generated. Upon receiving training data from the environment, t-DGR calculates the number of trajectories to generate based on the replay ratio r (Lines 4-5). The variable L (Line 7) represents the maximum length of trajectories observed so far.

To generate a trajectory τ of length L, t-DGR iterates over each timestep $1 \leq j \leq L$ (Line 9). At each timestep, t-DGR generates the j-th state observation of the trajectory using the previous generator $G_\beta^{(t-1)}$ conditioned on timestep j (Line 10), and then labels it with an action using the previous policy $\pi_\theta^{(t-1)}$ (Line 11). After generating all timesteps in the trajectory τ, t-DGR adds it to the existing training dataset (Line 14). It's important to note that the generated state observations within a trajectory do not have temporal coherence, as each state observation is generated independently of other timesteps. This approach is acceptable since our learner is trained on state-action pairs rather than full trajectories. However, unlike generating state observations i.i.d., our method ensures equal coverage of every timestep during the generative process, significantly reducing sample complexity.

Once t-DGR has augmented the training samples from the environment with our generated training samples, t-DGR employs backpropagation to update both the generator and learner using the augmented dataset (Lines 16-18). The t-DGR algorithm continues this process of generative replay throughout the agent's lifetime, which can be infinite (Line 2). It is worth mentioning that although we perform the generative process of t-DGR at task boundaries for ease of understanding, no part of t-DGR is dependent on clear task boundaries.

5 Experiments

In this section, we outline the experimental setup and performance metrics employed to compare t-DGR with representative methods, followed by an analysis of experimental results across different benchmarks and performance metrics.

5.1 Experimental Setup

We evaluate our method on the Continual World benchmarks CW10 and CW20 [29], along with our own "General Continual Learning" variant of CW10 called GCL10. CW10 consists of a sequence of

Algorithm 1 Trajectory-based Deep Generative Replay (t-DGR)

```
 1: Initialize task index $t = 0$, replay ratio $r$, generator $G_\beta^{(0)}$, learner $\pi_\theta^{(0)}$, and learning rates $\lambda_\beta, \lambda_\theta$.
 2: while new task available do
 3:     $t \leftarrow t + 1$
 4:     Initialize dataset $D$ with trajectories from task $t$.
 5:     $n \leftarrow \frac{r*|D|}{1-r}$                          ▷ number of trajectories to generate
 6:     for $i = 1$ to $n$ do
 7:         $L \leftarrow$ maximum trajectory length
 8:         $\tau \leftarrow \emptyset$                           ▷ initialize trajectory of length $L$
 9:         for $j = 1$ to $L$ do
10:             $S \leftarrow G_\beta^{(t-1)}(j)$                  ▷ generate states
11:             $A \leftarrow \pi_\theta^{(t-1)}(S)$               ▷ label with actions
12:             $\tau_j \leftarrow (S, A)$                        ▷ add to trajectory
13:         end for
14:         $D \leftarrow D \cup \tau$                            ▷ add generated trajectory to $D$
15:     end for
16:     Update generator and learner using $D$
17:     $\beta^{(t)} \leftarrow \beta^{(t-1)} - \lambda_\beta \nabla_\beta \mathcal{L}_{G^{(t-1)}}(\beta^{(t-1)})$
18:     $\theta^{(t)} \leftarrow \theta^{(t-1)} - \lambda_\theta \nabla_\theta \mathcal{L}_{\pi^{(t-1)}}(\theta^{(t-1)})$
19: end while
```

10 Meta-World [31] tasks, where each task involves a Sawyer arm manipulating one or two objects in the Mujuco physics simulator. Notably, the observation and action spaces are continuous and remain consistent across all tasks. CW20 is an extension of CW10 with the tasks repeated twice. To our knowledge, Continual World is the only standard continual learning benchmark for decision-making tasks. GCL10 gives data to the learner in 10 sequential buckets $B_1, \ldots, B_{10}$. Data from task $\mathcal{T}_i$ from CW10 is split evenly between buckets B_{i-1}, B_i, and B_{i+1}, except for the first and last task. Task $\mathcal{T}_1$ is evenly split between buckets B_0 and B_1, and task $\mathcal{T}_{10}$ is evenly split between buckets B_9 and B_{10}.

In order to ensure bounded memory usage, we adopt a one-hot vector approach to condition the model on the task, rather than maintaining a separate final neural network layer for each individual task. Additionally, we do not allow separate biases for each task, as originally done in EWC [13]. Expert demonstrations for training are acquired by gathering 100 trajectories per task using hand-designed policies from Meta-World, with each trajectory limited to a maximum of 200 steps. Importantly, the learner model remains consistent across different methods and benchmark evaluations. Moreover, we maintain a consistent replay ratio of $r = 0.9$ across all pseudo-rehearsal methods.

We estimated the success rate S of a model by running each task 100 times. The metrics for each method were computed using 5 seeds to create a 90% confidence interval. Further experimental details, such as hyperparameters, model architecture, random seeds, and computational resources, are included in the appendix. This standardization enables a fair and comprehensive comparison of our proposed approach with other existing methods.

5.2 Metrics

We evaluate our models using three metrics proposed by the Continual World benchmark [29], with the average success rate being the primary metric. Although the forward transfer and forgetting metrics are not well-defined in a “General Continual Learning" setting, they are informative within the context of Continual World benchmarks. As a reminder from Section 3.2, let N denote the number of tasks, and s_i represent the success rate of the learner on task $\mathcal{T}_i$. Additionally, let $s_i(t)$ denote the success rate of the learner on task $\mathcal{T}_i$ after training on tasks $\mathcal{T}_1$ to $\mathcal{T}_t$.

Average Success Rate The average success rate, as given by Equation 2, serves as the primary evaluation metric for continual learning methods.

Average Forward Transfer We introduce a slightly modified metric for forward transfer that applies to a broader range of continual learning problems beyond just continual reinforcement learning in the Continual World benchmark. Let s_i^{ref} represent the reference performance of a single-task experiment on task $\mathcal{T}_i$. The forward transfer metric FT_i is computed as follows:

$$FT_i = \frac{D_i - D_i^{\text{ref}}}{1 - D_i^{\text{ref}}} \qquad D_i = \frac{s_i(i) + s_i(i-1)}{2} \qquad D_i^{\text{ref}} = \frac{s_i^{\text{ref}}}{2}$$

The average forward transfer FT is then defined as the mean forward transfer over all tasks, calculated as $FT = \frac{1}{N}\sum_{i=1}^{N} FT_i$.

Average Forgetting We measure forgetting using the metric F_i, which represents the amount of forgetting for task i after all training has concluded. F_i is defined as the difference between the success rate on task $\mathcal{T}_i$ immediately after training and the success rate on task $\mathcal{T}_i$ at the end of training.

$$F_i = s_i(i) - s_i(N)$$

The average forgetting F is then computed as the mean forgetting over all tasks, given by $F = \frac{1}{N}\sum_{i=1}^{N} F_i$.

5.3 Baselines

We compare the following methods on the Continual World benchmark using average success rate as the primary evaluation metric. Representative methods were chosen based on their success in the original Continual World experiments, while DGR-based methods were selected to evaluate whether t-DGR addresses the limitations of existing pseudo-rehearsal methods.

- **Finetune:** The policy is trained only on data from the current task.
- **Multitask:** The policy is trained on data from all tasks simultaneously.
- **oEWC [25]:** A variation of EWC known as online Elastic Weight Consolidation (oEWC) bounds the memory of EWC by employing a single penalty term for the previous model instead of individual penalty terms for each task. This baseline is the representative regularization-based method.
- **PackNet [15]:** This baseline is the representative parameter isolation method. Packnet safeguards previous task information in a neural network by iteratively pruning, freezing, and retraining parts of the network.
- **DGR [26]:** This baseline is a deep generative replay method that only generates individual state observations i.i.d. and not entire trajectories.
- **CRIL [6]:** This baseline is a deep generative replay method that trains a policy along with a start state generator and a dynamics model that predicts the next state given the current state and action. Trajectories are generated by using the dynamics model and policy to autoregressively generate next states from a start state.
- **t-DGR:** Our proposed method.

Due to the inability of oEWC and PackNet to handle blurry task boundaries, we made several adjustments for CW20 and GCL10. Since PackNet cannot continue training parameters for a task once they have been fixed, we treated the second repetition of tasks in CW20 as distinct from the first iteration, resulting in PackNet being evaluated with $N = 20$, while the other methods were evaluated with $N = 10$. As for GCL10 and its blurry task boundaries, the best approach we could adopt with oEWC and PackNet was to apply their regularization techniques at regular training intervals rather than strictly at task boundaries. During evaluation, all tasks were assessed using the last fixed set of parameters in the case of PackNet.

5.4 Discussion

t-DGR emerges as the leading method, demonstrating the highest success rate on CW10 (Table 1a), CW20 (Table 1c), and GCL10 (Table 1b). Notably, PackNet's performance on the second iteration of tasks in CW20 diminishes, highlighting its limited capacity for continually accommodating new tasks. This limitation underscores the fact that PackNet falls short of being a true lifelong learner, as it necessitates prior knowledge of the task count for appropriate parameter capacity allocation. On the

(a) CW10

Method	Success Rate ↑	FT↑	Forgetting↓
Finetune	16.4 ±6.4	-3.0 ±6.0	78.8 ±7.6
Multitask	97.0 ±1.0	N/A	N/A
oEWC	18.6 ±5.3	-6.3 ±5.7	74.1 ±6.1
PackNet	81.4 ±3.7	-14.8 ±7.8	**-0.1** ±1.2
DGR	75.0 ±5.8	-4.3 ±5.1	17.8 ±4.1
CRIL	28.4 ±10.6	-1.1 ±2.8	68.6 ±10.4
t-DGR	**81.9** ±3.3	**-0.3** ±4.9	14.4 ±2.5

(b) GCL10

Method	Success Rate ↑
Finetune	21.7 ±2.6
Multitask	97.0 ±1.0
oEWC	21.8 ±1.7
PackNet	26.9 ±5.6
DGR	75.3 ±4.4
CRIL	53.5 ±5.5
t-DGR	**81.7** ±4.0

(c) CW20

Method	Success Rate ↑	FT↑	Forgetting↓
Finetune	14.2 ±4.0	-0.5 ±3.0	82.2 ±5.6
Multitask	97.0 ±1.0	N/A	N/A
oEWC	19.4 ±5.3	-2.8 ±4.1	75.2 ±7.5
PackNet	74.1 ±4.1	-20.4 ±3.4	**-0.2** ±0.9
DGR	74.1 ±4.1	18.9 ±2.9	23.3 ±3.3
CRIL	50.8 ±4.4	4.4 ±4.9	46.1 ±5.4
t-DGR	**83.9** ±3.0	**30.6** ±4.5	14.6 ±2.9

(d) Replay Ratio

Ratio	t-DGR	DGR
0.5	**63.2** ±2.6	52.8 ±2.9
0.6	**66.3** ±4.4	56.9 ±4.5
0.7	**70.8** ±4.1	62.5 ±3.6
0.8	**75.0** ±6.9	69.2 ±4.9
0.9	**81.9** ±3.3	75.0 ±5.8

Table 1: Tables (a), (b), and (c) present the results for Continual World 10, General Continual Learning 10, and Continual World 20, respectively. The tables display the average success rate, forward transfer, and forgetting (if applicable) with 90% confidence intervals using 5 random seeds. An up arrow indicates that higher values are better and a down arrow indicates that smaller values are better. Table (d) compares the impact of replay amount on the average success rate of t-DGR and DGR on CW10 with 90% confidence intervals obtained using 5 random seeds. The best results are highlighted in bold.

contrary, pseudo-rehearsal methods, such as t-DGR, exhibit improved performance with the second iteration of tasks in CW20 due to an increased replay time. These findings emphasize the ability of DGR methods to effectively leverage past knowledge, as evidenced by their superior forward transfer in both CW10 and CW20.

GCL10 (Table 1b) demonstrates that pseudo-rehearsal methods are mostly unaffected by blurry task boundaries, whereas PackNet's success rate experiences a significant drop-off. This discrepancy arises from the fact that PackNet's regularization technique does not work effectively with less clearly defined task boundaries.

Additionally, it is worth noting the diminishing performance gap between DGR and t-DGR as the replay ratio increases in Table 1d, indicating that a higher replay ratio reduces the likelihood of any portion of the trajectory being insufficiently covered when sampling individual state observations i.i.d., thereby contributing to improved performance. This trend supports the theoretical sample complexity of DGR derived in Section 2.2, as $\Theta(n \log n + mn \log \log n)$ closely approximates the sample complexity of t-DGR, $\Theta(mn)$, when the replay amount $m \to \infty$. However, it is important to emphasize that while DGR can achieve comparable performance to t-DGR with a high replay ratio, the availability of extensive replay time is often limited in many real-world applications.

Overall, t-DGR exhibits promising results, outperforming other methods in terms of success rate in all evaluations. Notably, t-DGR achieves a significant improvement over existing pseudo-rehearsal methods on CW20 using a Welch t-test with a significance level of p-value $= 0.005$. Its ability to handle blurry task boundaries, leverage past knowledge, and make the most of replay opportunities position it as a state-of-the-art method for continual lifelong learning in decision-making.

6 Conclusion

In conclusion, we have introduced t-DGR, a novel method for continual learning in decision-making tasks, which has demonstrated state-of-the-art performance on the Continual World benchmarks. Our approach stands out due to its simplicity, scalability, and non-autoregressive nature, positioning it as a solid foundation for future research in this domain.

Importantly, t-DGR aligns with the concept of “General Continual Learning" by taking into account essential properties of the real world, including bounded memory and blurry task boundaries. These considerations ensure that our method remains applicable and effective in real-world scenarios, enabling its potential integration into practical applications.

Looking ahead, one potential avenue for future research is the refinement of the replay mechanism employed in t-DGR. Rather than assigning equal weight to all past trajectories, a more selective approach could be explored. By prioritizing certain memories over others and strategically determining when to replay memories to the learner, akin to human learning processes, we could potentially enhance the performance and adaptability of our method.

References

[1] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization, 2019.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget, 2018.

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[4] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples, 2021.

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline, 2020.

[6] Chongkai Gao, Haichuan Gao, Shangqi Guo, Tianren Zhang, and Feng Chen. Cril: Continual robot imitation learning via generative and prediction model, 2021.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[8] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[10] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines, 2019.

[11] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization, 2021.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, mar 2017.

[14] Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

[15] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning, 2018.

[16] James Mcclelland, Bruce Mcnaughton, and Randall O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102:419–57, 08 1995.

[17] James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Complementary learning systems within the hippocampus: A neural network modeling approach to understanding episodic memory consolidation. *Psychological Review*, 102(3):419–457, 1995.

[18] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[19] Amy N. Myers and Herbert S. Wilf. Some new aspects of the coupon-collector's problem, 2003.

[20] Donald J. Newman. The double dixie cup problem. *The American Mathematical Monthly*, 67(1):58–61, 1960.

[21] Andrew Y. Ng and Stuart J. Russell. Inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 663–670, 2000.

[22] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning, 2018.

[23] Randall C. O'Reilly and Kenneth A. Norman. Hippocampal and neocortical contributions to memory: Advances in the complementary learning systems framework. *Trends in Cognitive Sciences*, 6(12):505–510, December 2002.

[24] Mark Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, 1994.

[25] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress compress: A scalable framework for continual learning, 2018.

[26] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay, 2017.

[27] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning, 2019.

[28] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2023.

[29] Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning, 2021.

[30] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018.

[31] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021.