ONLINE CONTINUAL DATASET DISTILLATION FROM EVOLVING LEARNING TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

We study online continual dataset distillation, where a compact standalone dataset is constructed from a single-pass, non-stationary stream and later used to train a fresh model from scratch. Existing dataset distillation methods typically require repeated access to source data or offline expert trajectories, while prior continual-learning approaches mainly use condensed samples as replay for the learner being trained. Although stream data become unavailable after being processed, the learning trajectories they induce remain available as supervision for dataset construction. We exploit this supervision and propose BiTraC, which distills two levels of learning dynamics from a meta-continual learner. Inner trajectories capture current-task acquisition and supervise the synthesis of newly observed tasks. Outer trajectories provide complementary cross-task information to retrospectively refine earlier synthetic data. A progressive trajectory curriculum gradually incorporates later learning states during distillation. Experiments on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet show that the resulting datasets outperform retained real-data and online summarization baselines in most settings, remain effective at intermediate task boundaries, and transfer across downstream model architectures.

1 INTRODUCTION

When data arrive as a non-stationary stream, the full training history cannot be revisited freely. Continual learning therefore focuses on preserving previously acquired knowledge in the parameters of an online learner, often with the help of a bounded replay memory. Dataset distillation takes a different view. It preserves training utility in a compact synthetic dataset from which a model can be trained from scratch (Wang et al., 2020; Lei & Tao, 2024; Yu et al., 2023). We study the intersection of these two settings. Our goal is to construct a standalone distilled dataset from a single-pass stream, so that the knowledge acquired during online learning can later be reused to train a fresh model independently of the original continual learner.

Existing dataset distillation methods construct synthetic data by matching gradients (Zhao et al., 2021), feature distributions (Zhao & Bilen, 2022), or training trajectories (Cazenavette et al., 2022). Most assume repeated access to the source dataset during distillation, while trajectory matching further relies on expert trajectories generated offline from the full training set. These assumptions do not directly fit a single-pass stream. Existing work combining dataset distillation with continual learning has mainly used synthetic samples as replay memory for a particular online learner (Sangermano et al., 2022; Gu et al., 2024), where their value is measured by how well they reduce forgetting in that learner. Another line of work applies offline condensation to sequential tasks and accumulates the resulting synthetic sets (Zhao et al., 2021; Zhao & Bilen, 2022; Zhao et al., 2023; Zhou et al., 2022; Yin et al., 2024; Yang et al., 2023), but still assumes repeated access to the data within each task. In contrast, we study *online continual dataset distillation*, where each example is observed once and the resulting synthetic data form a standalone training set that is evaluated independently by training a new model from random initialization.

The key observation is that although source examples are transient, the parameter transitions they induce are generated as part of the online learning process and can be recorded. These learning trajectories provide supervision for dataset construction without repeatedly revisiting the source data or training a separate bank of offline experts. The setting also differs from conventional trajectory matching in an important way. Offline expert trajectories are generated by repeatedly training on a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fixed dataset, whereas trajectories in continual learning evolve together with the learner as new tasks arrive and past knowledge is revisited through replay. The problem is therefore not simply to replace offline experts with an online trajectory, but to make use of learning dynamics that change throughout the stream.

We exploit these evolving learning dynamics with BiTraC. Meta-continual learning naturally produces two nested forms of parameter evolution. Inner-loop trajectories describe how the learner adapts to the current task and provide supervision for synthesizing newly observed data. Outer-loop trajectories describe how the persistent learner evolves when current-task learning is combined with replayed prior knowledge. We use these outer trajectories as complementary supervision to retrospectively refine historical synthetic images and soft labels using cross-task information that becomes available later in the stream. Since the recorded trajectories themselves evolve over time, BiTraC further uses a progressive chronological curriculum that first exposes earlier trajectories and gradually incorporates later, more context-dependent learning states.

Experiments on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet evaluate the distilled datasets independently by training fresh models from scratch. BiTraC achieves the highest mean accuracy in 6 of 7 dataset–budget settings, including improvements of 4.66 and 6.35 percentage points over the strongest competing retained-data method on Seq-CIFAR100 at Images Per Class IPC$= 1$ and IPC$= 10$, respectively. The accumulated datasets are also useful before the stream ends. On Seq-CIFAR100 with IPC$= 10$, BiTraC achieves the highest mean accuracy at all 10 task boundaries. The resulting datasets further transfer to architectures not used during distillation. Ablation studies show that inner-trajectory matching provides the largest gain, while soft-target learning, outer-trajectory refinement, and the progressive curriculum provide additional improvements.

Our contributions are summarized as follows.

- We study online continual dataset distillation, where a compact standalone dataset is constructed from a single-pass, non-stationary stream and evaluated independently by training a fresh model from scratch. This setting differs from replay-oriented condensation and from continual distillation methods that retain repeated access to each task.
- We propose BiTraC, which uses the evolving learning trajectories generated by a meta-continual learner as supervision for dataset construction. Inner trajectories supervise the synthesis of newly observed tasks, while outer trajectories provide complementary cross-task information for refining historical synthetic data. A progressive curriculum accommodates the changing trajectory states throughout the stream.
- We show on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet that the resulting datasets achieve strong standalone training performance, remain effective at intermediate task boundaries, and transfer across downstream model architectures.

2 Related Work

Dataset Distillation and Trajectory Matching. Dataset distillation constructs a compact synthetic dataset that preserves the training utility of a much larger real dataset (Wang et al., 2020). Existing methods optimize synthetic samples by matching gradients (Zhao et al., 2021; Zhao & Bilen, 2021), feature distributions (Zhao & Bilen, 2022; Zhao et al., 2023), or model training dynamics (Cazenavette et al., 2022; Guo et al., 2024). Matching Training Trajectories (MTT) first trains expert networks offline on the full real dataset and then optimizes synthetic samples so that training on them reproduces segments of the expert parameter trajectories (Cazenavette et al., 2022). DATM (Guo et al., 2024) further selects trajectory segments according to learning difficulty and the available synthetic-data budget. These methods are designed for an offline setting in which the source data can be revisited and expert trajectories can be generated in a separate training stage. BiTraC uses trajectories generated during online continual learning itself, so the available trajectory supervision evolves together with the data stream.

Dataset Distillation for Continual Learning. Most work combining dataset distillation with continual learning uses synthetic or condensed samples as replay memory. Early methods synthesize samples to mitigate catastrophic forgetting (Masarczyk & Tautkute, 2020; Wiewel & Yang, 2021; Rosasco et al., 2021; Sangermano et al., 2022), while later methods improve replay through efficient

condensation (Liao et al., 2025; Nguyen et al., 2026), stream summarization (Gu et al., 2024), or hybrid real–synthetic memories (Kong et al., 2024). The distilled samples are typically evaluated through the performance of the continual learner that uses them for replay. Other work considers compressed data that can be reused independently of the original learner. Remember the Past constructs addressable distilled memories for training new models (Deng & Russakovsky, 2022), while related formulations have been studied in few-shot class-incremental learning (Bao et al., 2026), domain-incremental dataset distillation (Oh & Sim, 2026), streaming image classification (Khatib & Aperstein, 2025), and streaming recommendation (Zhang et al., 2026). These works establish different forms of sequential data condensation, but differ in both how the source data can be accessed and how the resulting condensed data are used.

Meta-Continual Learning. Meta-continual learning (MCL) organizes online learning around the interaction between adaptation to current data and updates that integrate previously acquired knowledge. MER (Riemer et al., 2019) encourages gradient alignment through repeated meta-updates, while La-MAML (Gupta et al., 2020) formulates online continual learning as bi-level optimization and learns the inner-loop update rates. VR-MCL (Wu et al., 2024) improves the estimation of the corresponding online hypergradients through variance reduction. Our work does not propose a new continual-learning update. Instead, it uses the inner and outer optimization trajectories generated by MCL as supervision for dataset distillation. BiTraC applies to meta-continual learners that expose these two levels of learning dynamics. We instantiate the framework with VR-MCL in our experiments and provide an additional instantiation with La-MAML in Appendix A.5.

3 Preliminaries

3.1 Online Continual Dataset Distillation

We consider a class-incremental stream of T supervised tasks $\{\mathcal{D}_t\}_{t=0}^{T-1}$. Each task introduces a disjoint set of classes $\mathcal{C}_t$, and $\mathcal{C}_{\leq t} = \bigcup_{q=0}^{t} \mathcal{C}_q$ denotes all classes observed through task t. Task boundaries are known during construction, but task identities are unavailable at inference. Each source task is traversed once. Real examples retained in bounded buffers may be reused, including a replay memory $\mathcal{M}$ maintained by the continual learner.

Our goal is to construct a compact synthetic dataset that can train a new model independently of the continual learner. After completing task t, the accumulated dataset is

$$\mathcal{S}^{(t)} = \bigcup_{q=0}^{t} \mathcal{S}_q^{(t)},$$

where $\mathcal{S}_q^{(t)}$ denotes the current version of the synthetic data initially constructed for task q. Each synthetic sample consists of an image $\tilde{x}$ and a soft label $\tilde{z}$. At stage t, the soft label is a probability distribution over $\mathcal{C}_{\leq t}$, including both current and previously observed classes. The superscript allows the images and labels associated with an earlier task to change as later tasks are processed.

At any completed task boundary, we evaluate the performance by freezing $\mathcal{S}^{(t)}$ and training a new model from random initialization using only its synthetic images and stored soft labels. Evaluation is performed over all classes in $\mathcal{C}_{\leq t}$ without task identities. Neither the continual learner's parameters nor its real replay memory is used in this training. The final dataset $\mathcal{S}^{(T-1)}$ is evaluated in the same way over the complete class space.

3.2 Meta-Continual Learning

Meta-continual learning (MCL) combines adaptation to current data with updates informed by previously observed data (Gupta et al., 2020; Wu et al., 2024). Let $\theta_{t,r}$ denote the persistent learner parameters before meta-update r of task t, where $r = 0, \ldots, N_t - 1$ and N_t is the number of meta-updates performed on task t. Starting from $\phi_{t,r}^{(0)} = \theta_{t,r}$, the inner loop performs L adaptation steps on current-task batches $\mathcal{B}_{t,r}^{(l)}$:

$$\phi_{t,r}^{(l+1)} = \phi_{t,r}^{(l)} - \beta \nabla_{\phi_{t,r}^{(l)}} \mathcal{L}\left(\phi_{t,r}^{(l)}; \mathcal{B}_{t,r}^{(l)}\right), \qquad l = 0, \ldots, L-1,$$

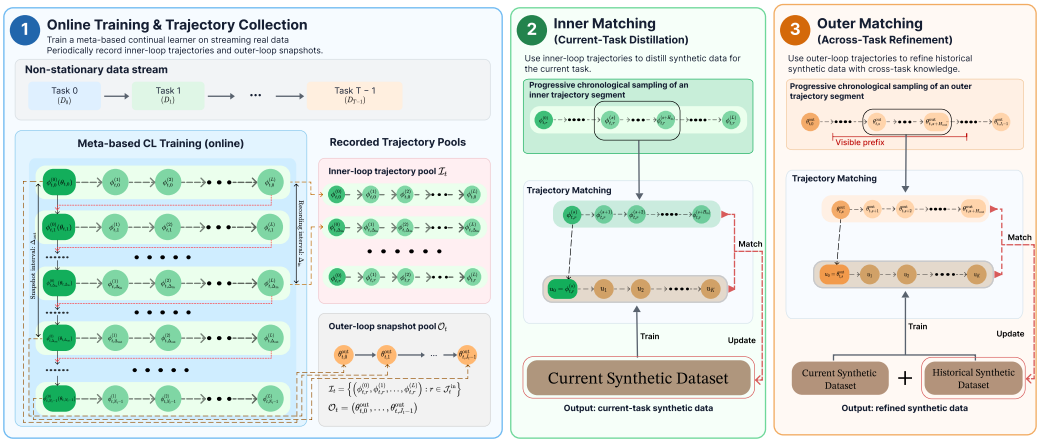


Figure 1: Overview of BiTraC. During the single pass over task t, the MCL learner produces inner adaptation trajectories and a persistent outer trajectory. Inner matching uses the current-task dynamics to construct $\mathcal{S}_t$. After $\mathcal{S}_t$ is fixed, outer matching trains a temporary student on $\mathcal{S}_{\leq t}$ while updating only the historical synthetic data $\mathcal{S}_{<t}$, incorporating cross-task information that becomes available later in the stream.

where $\mathcal{L}$ is the supervised loss on real data and β is the inner-loop learning rate. The adapted parameters are then evaluated on a query batch $\mathcal{Q}_{t,r}$ containing current-task and replay examples. A standard outer update takes the form

$$g_{t,r} = \nabla_{\theta_{t,r}} \mathcal{L}\left(\phi_{t,r}^{(L)}; \mathcal{Q}_{t,r}\right), \qquad \theta_{t,r+1} = \theta_{t,r} - \alpha g_{t,r}, \quad r = 0, \ldots, N_t - 1,$$

where α is the outer-loop learning rate. The hypergradient $g_{t,r}$ accounts for the dependence of the adapted parameters $\phi_{t,r}^{(L)}$ on the starting parameters $\theta_{t,r}$.

These updates produce two levels of learning trajectories. Each meta-update generates a short inner trajectory $(\phi_{t,r}^{(0)}, \ldots, \phi_{t,r}^{(L)})$ that describes current-task adaptation. Across the N_t meta-updates of task t, the persistent parameters form an outer trajectory $(\theta_{t,0}, \theta_{t,1}, \ldots, \theta_{t,N_t-1})$, which captures the evolution of the continual learner under both current-task and replay data. BiTraC applies to MCL learners that expose these two levels of trajectories. We instantiate the framework with VR-MCL (Wu et al., 2024) in our experiments.

4 Method

BiTraC uses the inner and outer trajectories generated during continual learning as supervision for constructing a standalone synthetic dataset. Fig. 1 gives an overview of the construction process. During the single pass over task t, the MCL learner processes real examples and updates its replay memory $\mathcal{M}$. In parallel, we record its learning trajectories and maintain a bounded current-task buffer $\mathcal{R}_t$ for synthetic-image initialization. At the task boundary, inner matching constructs the current-task dataset, after which outer matching refines the historical synthetic data while keeping the current-task data fixed. The current-task buffer and trajectory pools are released after these two stages. Synthetic data never enter the replay memory of the online learner.

For simplicity, we omit version superscripts within each task and write $\mathcal{S}_q = (\widetilde{X}_q, \widetilde{Z}_q)$ for the current synthetic images and soft labels associated with task q. We use $\mathcal{S}_{<t} = \bigcup_{q<t} \mathcal{S}_q$ for historical synthetic data and $\mathcal{S}_{\leq t} = \mathcal{S}_{<t} \cup \mathcal{S}_t$ for the accumulated dataset.

Trajectory-matching objective. Both stages use the same matching mechanism (Cazenavette et al., 2022; Guo et al., 2024). Given a recorded starting state a and target state b, we initialize a temporary student with $u_0 = a$ and train it on synthetic batches $\widetilde{\mathcal{B}}_k$ drawn from a synthetic dataset $\mathcal{S}$:

$$u_{k+1} = u_k - \eta \nabla_{u_k} \ell_{\text{soft}}\left(u_k; \widetilde{\mathcal{B}}_k\right), \qquad k = 0, \ldots, K - 1.$$

Here η is the student learning rate, and ℓ_{soft} denotes supervision using the synthetic images and their soft-label distributions, as in knowledge distillation. The synthetic labels themselves provide the target distributions. After K updates, the normalized endpoint error is

$$\mathcal{E}(\mathcal{S}; a, b, \eta) = \frac{\|u_K - b\|_2^2}{\|a - b\|_2^2}. \tag{1}$$

Gradients through the student updates optimize the synthetic data, while the recorded states remain fixed. These temporary students serve only to construct the dataset. Unlike the evaluation model, they are initialized from recorded parameter states rather than from random initialization.

4.1 Current-Task Distillation

During task t, we record complete inner trajectories at regular intervals, selecting one trajectory every fixed number of meta-updates. If $\mathcal{J}_t^{\text{in}}$ denotes their update indices, the resulting pool is

$$\mathcal{I}_t = \left\{ \left(\phi_{t,r}^{(0)}, \phi_{t,r}^{(1)}, \ldots, \phi_{t,r}^{(L)} \right) : r \in \mathcal{J}_t^{\text{in}} \right\}.$$

Each trajectory records the learner's parameters before and after its successive adaptation steps within a single meta-update. These trajectories provide the reference transitions for synthesizing the current task.

The buffer $\mathcal{R}_t$ maintains up to K_{buf} real examples per current-task class using reservoir sampling. At the task boundary, we select IPC examples per class from this buffer to initialize $\widetilde{X}_t$. Each synthetic image is associated with a trainable soft label in $\widetilde{Z}_t$. Although the images are allocated to current-task classes, their soft labels are defined over all observed classes $\mathcal{C}_{\leq t}$.

At each distillation iteration, we sample an inner trajectory indexed by r and a valid starting position s. The reference segment runs from $\phi_{t,r}^{(s)}$ to $\phi_{t,r}^{(s+H_{\text{in}})}$, where H_{in} is the matching horizon and $s + H_{\text{in}} \leq L$. We initialize the temporary student from the starting state and perform K differentiable updates on $\mathcal{S}_t$. The images, soft labels, and synthetic learning rate η_t are optimized through

$$\min_{\widetilde{X}_t, \widetilde{Z}_t, \eta_t} \mathcal{E}\left(\mathcal{S}_t; \phi_{t,r}^{(s)}, \phi_{t,r}^{(s+H_{\text{in}})}, \eta_t \right). \tag{2}$$

In this stage, each student update uses the current-task synthetic dataset. The student update count K need not equal the reference horizon H_{in}. Repeated matching over the recorded inner trajectories optimizes $\mathcal{S}_t$ to reproduce the current-task adaptation dynamics.

4.2 Historical Refinement

Historical synthetic data are constructed before the current task is observed. The outer updates made while learning task t provide additional cross-task information with which to refine these data. We obtain this supervision from the persistent parameter trajectory of the MCL learner.

We record outer states before the first meta-update, at regular intervals during task t, and after the final meta-update. The stored sequence is

$$\mathcal{O}_t = \left(\theta_{t,0}^{\text{out}}, \ldots, \theta_{t,J_t-1}^{\text{out}} \right), \qquad \theta_{t,j}^{\text{out}} = \theta_{t,r_j},$$

where J_t is the number of snapshots and r_j is the meta-update position corresponding to snapshot j. Thus, j indexes stored snapshots rather than individual meta-updates. These states reflect the persistent learner's evolution under current-task adaptation and replay-informed outer updates.

After current-task distillation, we sample a starting snapshot s and match a segment with horizon H_{out}. The temporary student starts from $\theta_{t,s}^{\text{out}}$ and is trained on batches drawn from $\mathcal{S}_{\leq t}$, which contains both current and historical synthetic data. We optimize only the historical images and labels:

$$\min_{\{\widetilde{X}_q, \widetilde{Z}_q\}_{q<t}} \mathcal{E}\left(\mathcal{S}_{\leq t}; \theta_{t,s}^{\text{out}}, \theta_{t,s+H_{\text{out}}}^{\text{out}}, \eta_{\text{out}} \right), \tag{3}$$

where $s + H_{\text{out}} < J_t$ and η_{out} is a fixed learning-rate hyperparameter. Here H_{out} counts stored snapshot intervals, each of which may span multiple meta-updates.

The current-task dataset $\mathcal{S}_t$ remains fixed throughout historical refinement but still participates in the student's training updates. Gradients through these updates modify the historical images and their soft labels. At this stage, historical labels can be refined over the expanded class set $\mathcal{C}_{\leq t}$, including the newly observed classes. This allows earlier synthetic data to use cross-task supervision that was unavailable when they were first constructed.

4.3 Progressive Trajectory Curriculum

The recorded trajectories originate from learner states that change throughout each task. We progressively expand the available trajectory supervision during matching, beginning with earlier states and gradually including later ones. For matching stage $\ell \in \{\mathrm{in}, \mathrm{out}\}$, the visible fraction at distillation iteration n is

$$\rho_\ell(n) = \rho_{\ell,0} + (1 - \rho_{\ell,0}) \min\left(\frac{n}{N_\ell}, 1\right), \tag{4}$$

where $\rho_{\ell,0} \in (0, 1]$ is the initial fraction and N_ℓ is the number of iterations required to expose the complete trajectory pool. The iteration count starts separately for each matching stage.

For inner matching, we order $\mathcal{I}_t$ by recording time and sample from its first $\lceil \rho_{\mathrm{in}}(n)|\mathcal{I}_t| \rceil$ complete trajectories. Within a selected trajectory, the starting position s is sampled uniformly from the valid positions for horizon H_{in}. The curriculum therefore controls which inner trajectories are available, without imposing an additional progression over positions within a trajectory.

For outer matching, we expose a chronological prefix of $\mathcal{O}_t$. Its length is $\max\{H_{\mathrm{out}}+1, \lceil \rho_{\mathrm{out}}(n) J_t \rceil\}$, ensuring that a valid segment can be selected. We then uniformly sample a starting snapshot whose target lies within this prefix. Both stages eventually use their complete recorded pools. The curriculum changes the temporal coverage of the supervision while leaving the matching objectives unchanged.

Algo. 1 in the appendix provides the complete task-level procedure.

5 Experiments

We evaluate BiTraC from four perspectives: 1) the quality of the final standalone dataset; 2) its availability throughout the task stream; 3) the contribution of individual design choices; and 4) transfer to downstream architectures different from the one used for distillation.

Benchmarks and backbone. We conduct experiments on three class-incremental benchmarks: Seq-CIFAR10 and Seq-CIFAR100 (Krizhevsky et al., 2009), and Seq-TinyImageNet (Le et al., 2015). Unless otherwise specified, both trajectory collection and downstream evaluation use a CIFAR-style reduced-width ResNet-18 (He et al., 2015). Seq-CIFAR10 contains 5 tasks with 2 classes per task, Seq-CIFAR100 contains 10 tasks with 10 classes per task, and Seq-TinyImageNet contains 20 tasks with 10 classes per task. Each task is traversed once during online learning. Detailed continual-learning and downstream-training protocols are provided in Appendix A.1.

Baselines. We compare BiTraC with representative retained-data and dataset-condensation methods under the same images-per-class (IPC) budget. Random Sampling uniformly retains a class-balanced subset of real examples. GDumb (Prabhu et al., 2020) maintains a class-balanced memory throughout the stream and trains a fresh model on the retained data. SSD (Gu et al., 2024) follows its original online stream-summarization procedure. DM (Zhao & Bilen, 2022) and LoDM (Yang et al., 2023) were originally developed with repeated access to source data, so we adapt them to the single-pass setting by performing task-wise condensation from a bounded current-task reservoir. We freeze each resulting synthetic dataset and evaluate it using the same downstream training protocol. Appendix A.4 provides a sanity check of the online adaptations of DM and LoDM. We instantiate BiTraC with VR-MCL (Wu et al., 2024) in the main experiments. Appendix A.5 additionally evaluates an instantiation with La-MAML (Gupta et al., 2020). We additionally analyze construction-time storage, wall-clock runtime, and peak GPU memory in Appendix A.6.

Table 1: Final Class-IL accuracy (%). Each retained or distilled dataset is independently constructed three times and evaluated with three downstream seeds per construction run. VR-MCL reports the final continual learner and is included only as an uncompressed reference. Ratio denotes the retained images per class as a percentage of the original training examples per class. Best results among retained-data and condensation methods are shown in bold.

Dataset	IPC	Ratio (%)	CL Reference	Retained Real Data			Dataset Condensation		
			VR-MCL	Random	GDumb	SSD	LoDM	DM	BiTraC
Seq-CIFAR10	1	0.02	56.48±1.79	15.31±1.02	18.02±1.07	10.38±0.32	20.76±0.93	**22.34±0.87**	20.91±1.07
	10	0.2		27.09±1.03	30.05±0.93	33.45±1.47	30.38±0.64	31.82±0.58	**36.64±1.73**
	50	1		41.13±1.47	45.31±0.81	43.72±1.89	45.05±0.67	43.90±0.09	**50.11±0.43**
Seq-CIFAR100	1	0.2	19.49±0.69	4.05±0.29	4.51±0.05	4.55±0.27	4.25±0.10	5.17±0.40	**9.83±0.23**
	10	2		12.70±0.61	14.23±0.43	15.52±0.26	13.27±0.23	11.60±0.38	**21.87±0.04**
Seq-TinyImageNet	1	0.2	13.27±0.39	2.41±0.12	2.37±0.05	2.10±0.05	2.54±0.13	2.05±0.09	**4.71±0.24**
	10	2		9.46±0.16	12.02±0.37	10.86±0.80	6.25±0.35	4.03±0.28	**13.02±0.74**

5.1 Final Distilled Dataset Quality

We first evaluate whether the final distilled dataset can train a new model independently of the continual learner that generated its trajectories. We use IPC values of 1, 10, and 50 on Seq-CIFAR10, and IPC values of 1 and 10 on Seq-CIFAR100 and Seq-TinyImageNet. For each setting, we perform three independent dataset-construction runs with different seeds. Each resulting dataset is frozen and used to train three models from random initialization, yielding nine downstream evaluations. We report the mean and standard deviation of the final Class-IL accuracy.

As shown in Tab. 1, BiTraC achieves the highest mean accuracy in six of the seven dataset–budget settings. On Seq-CIFAR10, it reaches 36.64% and 50.11% at IPC=10 and IPC=50, exceeding the strongest competing method by 3.19 and 4.80 percentage points, respectively. The gains are larger on Seq-CIFAR100. BiTraC reaches 9.83% versus 5.17% at IPC=1, and 21.87% versus 15.52% at IPC=10. On Seq-TinyImageNet, the corresponding improvements are 2.17 and 1.00 points. The exception is Seq-CIFAR10 at IPC=1, where DM reaches 22.34% and BiTraC reaches 20.91%. Overall, the results show that the learning trajectories produced during the online run can be converted into a compact dataset that remains useful after the original continual learner is discarded.

5.2 Anytime Availability of Distilled Datasets

A standalone dataset constructed from an online stream should remain usable without knowing in advance when the stream will terminate. We therefore freeze and evaluate the accumulated distilled dataset after every task boundary, using the same dataset-construction and downstream-training protocol as in Sec. 5.1.

Fig. 2 reports Seq-CIFAR100 with IPC=10. The corresponding Seq-CIFAR10 results with IPC=50 are provided in Appendix A.3. BiTraC achieves the highest mean accuracy at all ten Seq-CIFAR100 task boundaries. Its margin over the strongest competing method ranges from 0.86 points after task 2, where BiTraC reaches 34.68% compared with 33.82% for SSD, to 7.77 points after task 7, where the corresponding results are 27.20% and 19.43%. At the final boundary, BiTraC reaches 21.87%, exceeding SSD at 15.52% by 6.35 points. These results show that the accumulated synthetic data remain useful for standalone training throughout the stream rather than only after all tasks have been observed. Because each boundary introduces additional classes, the downward trend across tasks should not be interpreted as forgetting measured on a fixed label space.

5.3 Ablation Studies

Matching components. We first examine the successive components of BiTraC on Seq-CIFAR10 with IPC=50. *Initialization only* retains the real-image initialization without trajectory matching. *Inner only, hard* applies current-task trajectory matching with one-hot labels. *Inner only, learned soft* additionally learns the soft target distributions, while *Full* further enables historical refinement through outer matching. All variants use three independent construction runs and three downstream seeds per run.

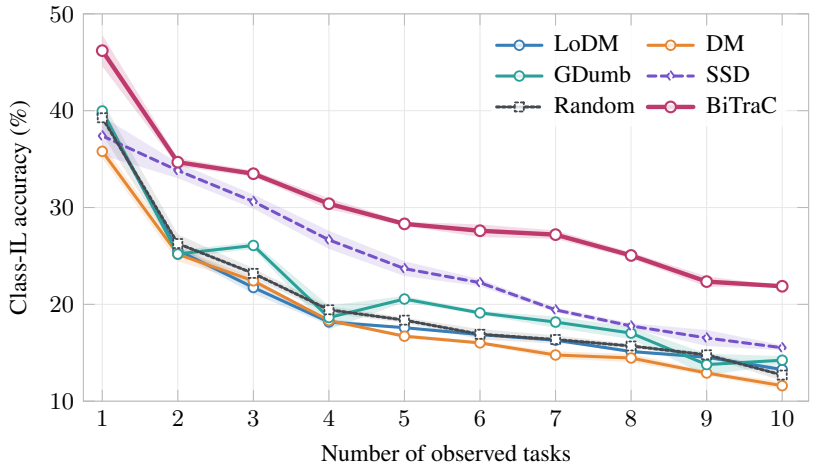


Figure 2: Class-IL accuracy (%) of the standalone datasets available after each task on Seq-CIFAR100 with IPC=10. Points and shaded regions show the mean and one standard deviation over nine downstream evaluations.

Table 2: Component ablation of BiTraC on Seq-CIFAR10 with IPC=50. "Inner" and "Outer" indicate whether the corresponding trajectory-matching stage is enabled.

Variant	Inner	Outer	Soft target	Learned target	Class-IL (%)
Initialization only	–	–	–	–	41.92±0.60
Inner only, hard	✓	–	–	–	48.67±0.75
Inner only, learned soft	✓	–	✓	✓	49.27±0.56
Full	✓	✓	✓	✓	**50.11±0.43**

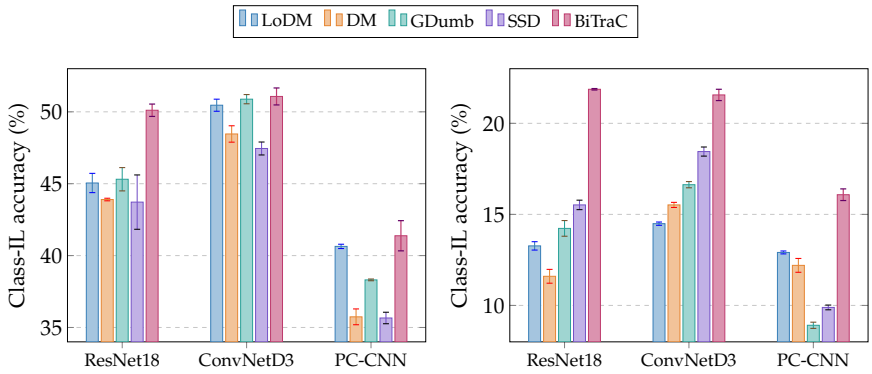
Table 3: Trajectory curriculum ablation of BiTraC on Seq-CIFAR100 with IPC=10. Results are reported as mean ± standard deviation over nine downstream evaluations.

Variant	Inner curriculum	Outer curriculum	Class-IL (%)
No curriculum	–	–	19.71±0.17
Inner only	✓	–	20.68±1.32
Outer only	–	✓	20.31±0.42
Full	✓	✓	**21.87±0.04**

Tab. 2 shows that the full method improves accuracy from 41.92% to 50.11%. Inner matching accounts for most of this gain, reaching 48.67% with hard labels. Learning soft targets further improves the result to 49.27%, and historical refinement through outer matching reaches 50.11%. These results are consistent with the roles of the two trajectory levels: inner trajectories provide the primary supervision for constructing the current task, while outer trajectories provide an additional cross-task refinement signal.

Trajectory curriculum. We next examine the progressive trajectory curriculum by independently enabling it for inner and outer matching. All variants retain both matching stages and learned soft labels, differing only in which trajectory pools are progressively exposed.

As shown in Tab. 3, disabling both curricula gives 19.71%. Applying the curriculum only to inner or outer trajectories improves the accuracy to 20.68% and 20.31%, respectively, while using both reaches 21.87%. The two schedules therefore contribute separately, with their combination providing the largest improvement.



(a) Seq-CIFAR10, IPC=50 (b) Seq-CIFAR100, IPC=10

Figure 3: Cross-architecture Class-IL accuracy (%) on (a) Seq-CIFAR10 with IPC=50 and (b) Seq-CIFAR100 with IPC=10. All distilled datasets are constructed using ResNet-18 trajectories. Error bars indicate the standard deviation over nine downstream evaluations (3 independent construction runs $\times$ 3 downstream seeds).

5.4 Cross-Architecture Generalization

A standalone distilled dataset should remain useful beyond the architecture that generated its matching trajectories. We therefore evaluate the frozen datasets with three downstream architectures: the ResNet-18 backbone used during distillation, ConvNetD3, and PC-CNN. Experiments are conducted on Seq-CIFAR10 with IPC=50 and Seq-CIFAR100 with IPC=10. For each method, we use three independently constructed datasets and train three downstream models from different random initializations for each architecture.

Fig. 3 shows that BiTraC achieves the highest mean accuracy in all six dataset–architecture combinations. On Seq-CIFAR10, its margins over the strongest baseline are 4.80 points on ResNet-18, 0.19 points on ConvNetD3, and 0.74 points on PC-CNN. On Seq-CIFAR100, the corresponding margins are 6.35, 3.11, and 3.17 points. The distilled data therefore retain their training utility when transferred to architectures that were not used to construct them, supporting their use as a standalone dataset rather than a representation tied to a particular continual learner or downstream architecture.

6 Conclusion

We study online continual dataset distillation, where a standalone synthetic dataset is constructed from a single-pass, non-stationary stream and later used to train a fresh model from scratch. We propose BiTraC, which uses the learning trajectories generated during meta-continual learning as supervision for dataset construction. Inner trajectories guide current-task distillation, while outer trajectories provide cross-task information for refining previously constructed synthetic data, with a progressive curriculum over the evolving trajectory states. Experiments on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet show that the resulting datasets achieve strong standalone training performance, remain effective at intermediate task boundaries, and transfer across downstream architectures. An additional instantiation with La-MAML further demonstrates that the same framework can be used with different meta-continual learners.

AI Use Statement

We used generative AI tools to support brainstorming, provide feedback on hypotheses and experimental design, assist with translation and software implementation, and help organize the presentation of experimental results. We did not use generative AI tools to generate synthetic datasets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, clean or reformat datasets, or support qualitative or thematic data analysis. Additionally, we used generative AI tools to assist with scientific figures, supporting code, manuscript editing, literature search and summarization, title and keyword suggestions, and reference formatting.

All AI-assisted work was reviewed by the authors. Code and figures were checked against the intended implementation and reported experiments, and literature summaries and references were checked against the original sources. The authors are responsible for all final text, claims, results, and other materials in this work.

Reproducibility statement

We provide the mathematical formulation of BiTraC, including its inner- and outer-trajectory matching objectives and progressive trajectory curriculum, in Sec. 4. Appendix A.1 documents the datasets, continual-learning protocol, model architectures, memory budgets, optimization hyperparameters, and downstream evaluation procedure. Appendix A.2 provides the complete task-level construction algorithm. Sec. 5.1 reports the numbers of independent dataset-construction runs and downstream evaluation seeds, while Appendix A.4 and Appendix A.5 describe the additional baseline-validation and learner-generalization protocols.

References

Kexin Bao, Daichi Zhang, Hansong Zhang, Yong Li, Yutao Yue, and Shiming Ge. Cdˆ 2: Constrained dataset distillation for few-shot class-incremental learning. *arXiv preprint arXiv:2601.08519*, 2026.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories, 2022. URL https://arxiv.org/abs/2203.11932.

Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks, 2022. URL https://arxiv.org/abs/2206.02916.

Jianyang Gu, Kai Wang, Wei Jiang, and Yang You. Summarizing stream data for memory-constrained online continual learning, 2024. URL https://arxiv.org/abs/2305.16645.

Ziyao Guo, Kai Wang, George Cazenavette, Hui Li, Kaipeng Zhang, and Yang You. Towards lossless dataset distillation via difficulty-aligned trajectory matching, 2024. URL https://arxiv.org/abs/2310.05773.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning, 2020. URL https://arxiv.org/abs/2007.13904.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Rwad Khatib and Yehudit Aperstein. Enhancing classification of streaming data with image distillation, 2025. URL https://arxiv.org/abs/2509.07049.

Jiangtao Kong, Jiacheng Shi, Ashley Gao, Shaohan Hu, Tianyi Zhou, and Huajie Shao. Hybrid memory replay: Blending real and distilled data for class incremental learning, 2024. URL https://arxiv.org/abs/2410.15372.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Shiye Lei and Dacheng Tao. A comprehensive survey of dataset distillation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):17–32, January 2024. ISSN 1939-3539. doi: 10.1109/tpami.2023.3322540. URL http://dx.doi.org/10.1109/TPAMI.2023.3322540.

Wenyang Liao, Quanziang Wang, Yichen Wu, Renzhen Wang, and Deyu Meng. Data-distill-net: A data distillation approach tailored for reply-based continual learning, 2025. URL https://arxiv.org/abs/2505.20135.

Wojciech Masarczyk and Ivona Tautkute. Reducing catastrophic forgetting with learning on synthetic data, 2020. URL https://arxiv.org/abs/2004.14046.

Minh-Duong Nguyen, Thien-Thanh Dao, Le-Tuan Nguyen, Dung D. Le, and Kok-Seng Wong. Memory-efficient continual learning with prototypical exemplar condensation, 2026. URL https://arxiv.org/abs/2603.13804.

Minyoung Oh and Jae-Young Sim. Asymmetric synthetic data update for domain incremental dataset distillation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XcsaCHaoJh.

Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *The European Conference on Computer Vision (ECCV)*, August 2020.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference, 2019. URL https://arxiv.org/abs/1810.11910.

Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples, 2021. URL https://arxiv.org/abs/2103.15851.

Mattia Sangermano, Antonio Carta, Andrea Cossu, and Davide Bacciu. Sample condensation in online continual learning, 2022. URL https://arxiv.org/abs/2206.11849.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020. URL https://arxiv.org/abs/1811.10959.

Felix Wiewel and Bin Yang. Condensed composite memory continual learning, 2021. URL https://arxiv.org/abs/2102.09890.

Yichen Wu, Long-Kai Huang, Renzhen Wang, Deyu Meng, and Ying Wei. Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TpD2aG1h0D.

Enneng Yang, Li Shen, Zhenyi Wang, Tongliang Liu, and Guibing Guo. An efficient dataset condensation plugin and its application to continual learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Murj6wcjRw.

Zeyuan Yin, Eric Xing, and Zhiqiang Shen. Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective, 2024. URL https://arxiv.org/abs/2306.13092.

Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review, 2023. URL https://arxiv.org/abs/2301.07014.

Jiaqing Zhang, Hao Wang, Mingjia Yin, Bo Chen, Qinglin Jia, Rui Zhou, Ruiming Tang, ChaoYi Ma, and Enhong Chen. Diet: Learning to distill dataset continually for recommender systems, 2026. URL https://arxiv.org/abs/2603.24958.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation, 2021. URL https://arxiv.org/abs/2102.08259.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching, 2022. URL https://arxiv.org/abs/2110.04181.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching, 2021. URL https://arxiv.org/abs/2006.05929.

Ganlong Zhao, Guanbin Li, Yipeng Qin, and Yizhou Yu. Improved distribution matching for dataset condensation, 2023. URL https://arxiv.org/abs/2307.09742.

Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression, 2022. URL https://arxiv.org/abs/2206.00719.

A Supplementary Material

A.1 Implementation Details

A.1.1 Datasets and Continual-Learning Protocol

We evaluate BiTraC on three class-incremental benchmarks. Seq-CIFAR10 divides the 10 classes of CIFAR-10 into 5 tasks with 2 classes per task, Seq-CIFAR100 divides the 100 classes of CIFAR-100 into 10 tasks with 10 classes per task, and Seq-TinyImageNet divides 200 classes into 20 tasks with 10 classes per task. Classes are disjoint across tasks, and each source task is traversed once. Task boundaries are known during dataset construction. At evaluation time, task identities are unavailable and predictions are made over all classes observed up to the current task boundary.

The continual learner maintains a bounded replay memory of real examples. BiTraC additionally maintains a temporary class-balanced reservoir $\mathcal{R}_t$ for the current task. This reservoir is used only to initialize the current-task synthetic images and is released after the corresponding dataset-construction stage. The synthetic data are not used as replay by the continual learner.

A.1.2 Base Learner and Trajectory Collection

The main experiments use VR-MCL (Wu et al., 2024) as the continual learner. We use a CIFAR-style reduced-width ResNet-18 with a base width of 20 channels, giving stage widths of 20, 40, 80, and 160. The network uses a 3×3 input convolution and omits the initial max-pooling layer.

For the current-task reservoir $\mathcal{R}_t$, we use a per-class capacity of $K_{\text{buf}} = 50$ for all datasets and IPC settings. Samples are maintained by standard reservoir sampling within each current-task class. At the task boundary, IPC examples per class are selected from $\mathcal{R}_t$ to initialize the synthetic images.

During online learning, we record the two trajectory sources described in Sec. 3. For the inner trajectories, we store the complete sequence of adaptation states for each recorded meta-update, including the parameters before the first inner update and after every subsequent inner update. For the outer trajectory, we store the persistent learner parameters every two meta-updates. The task-initial and task-final states are always retained. The current-task reservoir and recorded trajectories are released after current-task distillation and historical refinement are completed.

The main results use trajectories generated by VR-MCL. Appendix A.5 reports an additional instantiation with La-MAML (Gupta et al., 2020).

A.1.3 BiTraC Hyperparameters

Unless otherwise specified, we implement BiTraC using PyTorch 2.8.0, torchvision 0.23.0, and CUDA 12.8. Current-task distillation runs for $4{,}000$ matching iterations, while historical refinement runs for $2{,}000$ iterations. Both stages unroll 12 temporary-student updates. Historical refinement matches transitions spanning two stored outer-snapshot intervals.

The progressive curriculum is applied separately to the two trajectory pools. For current-task distillation, the visible fraction of the inner trajectories increases linearly from 0.25 to 1 over the first $2{,}400$ matching iterations. For historical refinement, the visible fraction of the outer trajectory increases from 0.2 to 1 over the first $1{,}200$ iterations.

During current-task distillation, the synthetic images, soft-label parameters, and synthetic student learning rate are jointly optimized using SGD. Their optimizer learning rates are 100, 2, and 10^{-5}, respectively, with corresponding momenta of 0.5, 0.9, and 0.5. The synthetic student learning rate η_t is initialized to 0.005. Each matching iteration uses the complete current-task synthetic dataset.

Each synthetic image is associated with a soft label over all classes observed so far. The label represents a probability distribution, analogous to the soft targets used in knowledge distillation, and is optimized together with the synthetic image. The same soft labels are used to supervise models trained from scratch on the final distilled dataset.

During historical refinement, the current-task synthetic dataset remains fixed while the historical synthetic images and soft labels are optimized. We use a context batch size of 128, an image optimizer learning rate of 25 with momentum 0.5, and a soft-label optimizer learning rate of 2 with momentum

Algorithm 1 BiTraC: Online Continual Dataset Distillation

Require: Task stream $\{\mathcal{D}_t\}_{t=0}^{T-1}$, MCL learner, real replay memory $\mathcal{M}$, images per class IPC, buffer capacity K_{buf}
Ensure: Standalone synthetic dataset $\mathcal{S}$

```
Initialize S ← ∅
for t = 0, ..., T − 1 do
    Initialize current-task buffer R_t and trajectory pools I_t, O_t
    Record the task-initial persistent learner state in O_t
    for each mini-batch B in the single pass over D_t do
        Update the MCL learner using B and replay samples from M
        Record complete inner trajectories and persistent outer states at scheduled intervals
        Update R_t and M using B
    end for
    Record the task-final persistent learner state if it is not already stored
    S_t ← DistillCurrent(R_t, I_t)
    if t > 0 then
        S ← RefineHistory(S, S_t, O_t)        ▷ Update historical synthetic data while keeping S_t fixed
    end if
    S ← S ∪ S_t
    Release R_t, I_t, and O_t
end for
return S
```

0.9. The soft-label optimizer learning rate is reduced to 0.5 on Seq-CIFAR10. The temporary student learning rate η_{out} in Eq. (3) is fixed to the task-specific learning rate η_t learned during inner-loop distillation. Differentiable color, crop, cutout, flip, scale, and rotation augmentations are applied during both dataset construction and downstream training.

A.1.4 Baselines and Downstream Evaluation

We compare BiTraC with Random Sampling, GDumb (Prabhu et al., 2020), SSD (Gu et al., 2024), LoDM (Yang et al., 2023), and DM (Zhao & Bilen, 2022) under the same IPC budget. Random Sampling and GDumb retain class-balanced real examples. SSD follows its original online memory-update procedure. For LoDM, we use the same-memory configuration. The online adaptations of DM and LoDM are described in Appendix A.4.

We evaluate IPC values of 1, 10, and 50 on Seq-CIFAR10, and IPC values of 1 and 10 on Seq-CIFAR100 and Seq-TinyImageNet. For each main experimental setting, we perform three independent dataset-construction runs. Each resulting dataset is frozen and used to train three downstream models from different random initializations. We report the mean and standard deviation over the resulting nine downstream accuracies.

Unless otherwise specified, downstream evaluation uses the same reduced-width ResNet-18 architecture described above, initialized from scratch. Each model is trained for $1{,}000$ epochs using SGD with a learning rate of 0.01, momentum 0.9, weight decay 5×10^{-4}, a maximum batch size of 64, and no learning-rate decay. The cross-architecture experiments replace the downstream network with ConvNetD3 or PC-CNN while keeping the distilled dataset fixed.

A.2 Full Construction Algorithm

Algorithm 1 gives the complete task-level procedure. DistillCurrent performs current-task distillation using the inner trajectories, while RefineHistory updates the historical synthetic data using the outer trajectory.

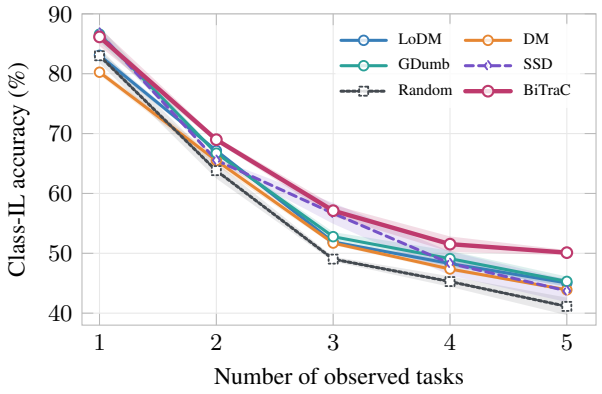


Figure 4: Class-IL accuracy (%) of the standalone datasets available after each task on Seq-CIFAR10 with IPC=50. Points and shaded regions show the mean and one standard deviation over nine downstream evaluations.

A.3 Anytime Availability on Seq-CIFAR10

The main paper reports the task-boundary evaluation on Seq-CIFAR100 with IPC=10. Fig. 4 gives the corresponding results on Seq-CIFAR10 with IPC=50. At each task boundary, the accumulated synthetic dataset is frozen and a new model is trained from random initialization over all classes observed so far.

As shown in Fig. 4, BiTraC achieves the highest mean accuracy after tasks 2–5. After task 1, it reaches 86.15%, compared with 86.80% for SSD. At the final boundary, BiTraC reaches 50.11%, compared with 45.31% for GDumb, the strongest competing method at that point. Together with the Seq-CIFAR100 results in the main paper, the results show that the accumulated synthetic datasets remain effective for standalone training at intermediate task boundaries.

A.4 Online Adaptations of DM and LoDM

DM (Zhao & Bilen, 2022) and LoDM (Yang et al., 2023) were originally developed with repeated access to source data. We adapt them to the single-pass setting by maintaining a class-balanced reservoir for the current task. At each task boundary, the current-task synthetic samples are distilled using only the real examples retained in this reservoir. Previously distilled samples are retained across subsequent tasks.

We use three settings to examine this adaptation:

- **Offline Joint**: the original offline procedure jointly distills all classes using the complete CIFAR-10 training set;
- **Task-wise Full**: each task is distilled separately with unrestricted access to all real examples from the current task;
- **Actual Online**: each task is traversed once and at most 50 real examples per class are retained in the current-task reservoir.

All experiments are conducted on CIFAR-10 with IPC=50 using ConvNetD3. For LoDM, we use the same-memory configuration with rank 4. For each setting, we construct one distilled dataset and train three independently initialized downstream models. Tab. 4 reports the mean and standard deviation over these three downstream runs.

Tab. 4 shows that Task-wise Full gives results close to Offline Joint for both methods. DM changes from 61.22% to 61.92%, while LoDM changes from 63.84% to 64.02%. Under the Actual Online protocol, the corresponding accuracies are 50.03% and 51.53%. These results quantify the effect of applying DM and LoDM under the single-pass data-access constraint used in the main experiments.

Table 4: Online adaptations of DM and LoDM on CIFAR-10 with IPC=50. Each entry reports the mean Class-IL accuracy (%) and standard deviation over three downstream models trained from one constructed dataset.

Setting	**DM**	**LoDM**
Offline Joint	61.22 ± 0.28	63.84 ± 0.11
Task-wise Full	61.92 ± 0.19	64.02 ± 0.18
Actual Online	50.03 ± 1.02	51.53 ± 0.77

Table 5: Instantiation of BiTraC with La-MAML on Seq-CIFAR10 with IPC=50. CL reference reports the final Class-IL accuracy of the corresponding continual learner. Dataset accuracy is obtained by training a fresh reduced-width ResNet-18 solely on the resulting synthetic dataset. Results are reported as mean ± standard deviation over nine downstream evaluations from three independent dataset-construction runs and three downstream seeds per run.

Method	Base learner	CL reference (%)	Dataset accuracy (%)
Random Sampling	–	–	41.13 ± 1.47
BiTraC	La-MAML	33.43 ± 1.21	43.43 ± 0.42
BiTraC	VR-MCL	56.48 ± 1.79	$\mathbf{50.11 \pm 0.43}$

A.5 Instantiation with an Alternative Meta-Continual Learner

The main experiments use VR-MCL (Wu et al., 2024) to generate the learning trajectories. We additionally instantiate BiTraC with La-MAML (Gupta et al., 2020). Like VR-MCL, La-MAML provides inner adaptation trajectories within each meta-update and an outer trajectory of persistent parameters across meta-updates. We apply the same current-task distillation and historical-refinement procedure to these trajectories.

We conduct the experiment on Seq-CIFAR10 with IPC=50. Shared components and hyperparameters of BiTraC are kept unchanged, while learner-specific optimization settings follow the corresponding MCL implementation. As in the main experiments, we perform three independent dataset-construction runs and evaluate each resulting dataset with three independently initialized reduced-width ResNet-18 models. Results are reported as mean and standard deviation over the nine downstream evaluations. Random Sampling under the same IPC budget is included as a reference.

As shown in Tab. 5, the La-MAML instantiation reaches a dataset accuracy of $43.43 \pm 0.42\%$, compared with $41.13 \pm 1.47\%$ for Random Sampling and $50.11 \pm 0.43\%$ for the VR-MCL instantiation. The corresponding continual learners reach $33.43 \pm 1.21\%$ and $56.48 \pm 1.79\%$ for La-MAML and VR-MCL, respectively. The results show that the same trajectory-based dataset construction can also be instantiated with a different meta-continual learner.

A.6 Resource and Computational Cost Analysis

The IPC budget determines the size of the final distilled dataset, while dataset construction additionally requires temporary buffers and recorded learning trajectories. We report the construction-time storage and computational cost on Seq-CIFAR100 with IPC=10.

A.6.1 Construction-Time Storage

During task t, BiTraC maintains the replay memory $\mathcal{M}$ of the continual learner, the current-task reservoir $\mathcal{R}_t$, the inner-trajectory pool $\mathcal{I}_t$, and the outer-trajectory pool $\mathcal{O}_t$, in addition to the accumulated synthetic dataset. The replay memory is part of the underlying continual learner; the remaining temporary resources are introduced during dataset construction.

Let P_{main} denote the number of parameters in an inner-loop state and b the number of bytes used to store each parameter. If task t retains $|\mathcal{J}_t^{\text{in}}|$ complete inner trajectories, each containing $L+1$ states, the inner-trajectory storage is

$$|\mathcal{J}_t^{\text{in}}|(L+1)P_{\text{main}}b.$$

Table 6: Construction-time storage on Seq-CIFAR100 with IPC=10. Task-specific buffers and trajectories are released after each task.

Resource	Size formulation	Peak storage	BiTraC-specific
Replay memory $\mathcal{M}$	Base-learner memory	12.30 MB	No
Current-task reservoir $\mathcal{R}_t$	$K_{\text{buf}}\lvert\mathcal{C}_t\rvert$	500 images / 6.15 MB	Yes
Inner trajectory pool $\mathcal{I}_t$	$\lvert\mathcal{J}_t^{\text{in}}\rvert(L+1)P_{\text{main}}b$	1.75 GB	Yes
Outer trajectory pool $\mathcal{O}_t$	$J_t P_{\text{outer}} b$	0.70 GB	Yes
Synthetic dataset $\mathcal{S}^{(t)}$	IPC-matched images + soft labels	12.70 MB	Yes
Peak stored-state footprint	–	$\approx$ 2.48 GB	–

Table 7: Construction time and peak GPU memory on Seq-CIFAR100 with IPC=10.

Method	Construction time (min)	Peak GPU memory (GB)
DM	988.8	2.55
LoDM	858.5	1.12
BiTraC	992.7	10.78

Similarly, let P_{outer} denote the number of parameters in an outer state. For J_t recorded outer snapshots, the corresponding storage is

$$J_t P_{\text{outer}} b.$$

The current-task reservoir and both trajectory pools are released after current-task distillation and historical refinement. Their storage therefore does not accumulate across tasks; we report the maximum footprint over individual tasks.

Tab. 6 shows that trajectory recording dominates the stored-state footprint. The inner trajectories require approximately 1.75 GB and the outer trajectory 0.70 GB, while the replay memory, current-task reservoir, and accumulated synthetic dataset each require only tens of megabytes. The resulting peak stored-state footprint is approximately 2.48 GB.

A.6.2 Computational Cost

We further compare construction time and peak GPU memory with DM and LoDM under the same hardware and software environment. Construction time is measured from the beginning of stream processing until the final distilled dataset is obtained. For BiTraC, this includes online continual learning and trajectory collection, current-task distillation, and historical refinement. Downstream training is excluded from this measurement. Peak GPU memory is the maximum allocated GPU memory observed during dataset construction.

As shown in Tab. 7, BiTraC takes 992.7 minutes for the complete construction process, compared with 988.8 minutes for DM and 858.5 minutes for LoDM. The construction time is therefore close to DM and 15.6% higher than LoDM.

The larger difference appears in peak GPU memory. BiTraC reaches 10.78 GB, compared with 2.55 GB for DM and 1.12 GB for LoDM. This peak includes the memory used by differentiable trajectory matching, including model parameters, activations, gradients, optimizer states, and the computation graphs retained across the unrolled student updates. It therefore measures a different quantity from the stored-state footprint in Tab. 6, which accounts for the buffers, recorded trajectories, and accumulated synthetic data retained during construction.

Overall, BiTraC has similar wall-clock construction time to DM in this setting, while requiring substantially more GPU memory during trajectory matching. The recorded trajectory states are task-local and are released after each task, leaving only the IPC-controlled synthetic dataset after construction.

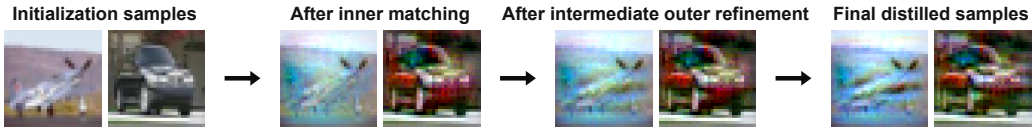


Figure 5: Evolution of representative samples from real-image initialization through current-task distillation and historical refinement.



Figure 6: Final distilled samples on Seq-CIFAR10 with IPC=1.

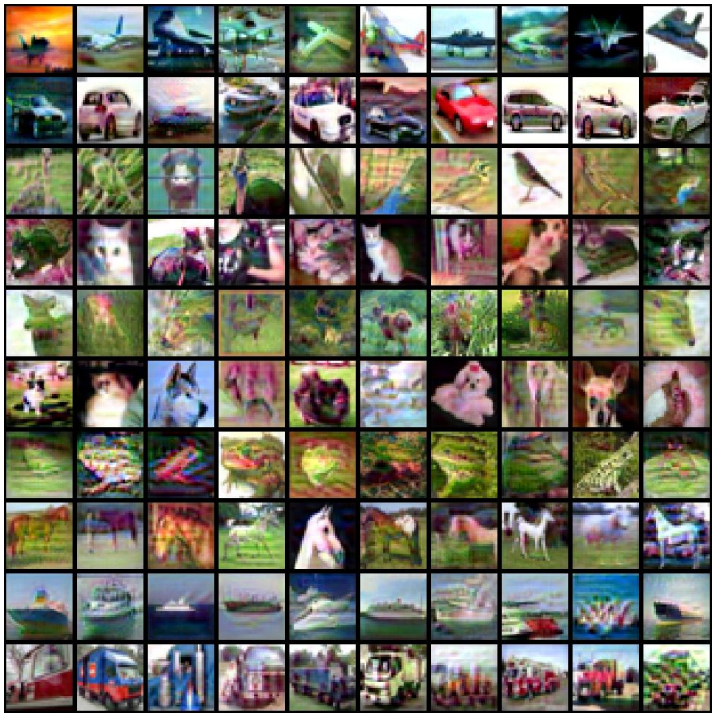


Figure 7: Final distilled samples on Seq-CIFAR10 with IPC=50.

A.7 Visualization of Distilled Samples

Fig. 5 shows representative synthetic images from initialization through current-task distillation, historical refinement, and the final distilled dataset. Figs. 6 and 7 show the final synthetic samples on Seq-CIFAR10 at IPC=1 and IPC=50, respectively.