Distributionally Robust Memory Evolution With Generalized Divergence for Continual Learning

Zhenyi Wang, Li Shen, Tiehang Duan, Qiuling Suo, Le Fang, Wei Liu, *Fellow, IEEE*, and Mingchen Gao

***Abstract*—Continual learning (CL) aims to learn a non-stationary data distribution and not forget previous knowledge. The effectiveness of existing approaches that rely on memory replay can decrease over time as the model tends to overfit the stored examples. As a result, the model's ability to generalize well is significantly constrained. Additionally, these methods often overlook the inherent uncertainty in the memory data distribution, which differs significantly from the distribution of all previous data examples. To overcome these issues, we propose a principled memory evolution framework that dynamically adjusts the memory data distribution. This evolution is achieved by employing distributionally robust optimization (DRO) to make the memory buffer increasingly difficult to memorize. We consider two types of constraints in DRO: f-divergence and Wasserstein ball constraints. For f-divergence constraint, we derive a family of methods to evolve the memory buffer data in the continuous probability measure space with Wasserstein gradient flow (WGF). For Wasserstein ball constraint, we directly solve it in the euclidean space. Extensive experiments on existing benchmarks demonstrate the effectiveness of the proposed methods for alleviating forgetting. As a by-product of the proposed framework, our method is more robust to adversarial examples than compared CL methods.**

***Index Terms*—Continual learning, distributionally robust optimization, f-divergence, Wasserstein gradient flow.**

I. Introduction

CONTINUAL learning (CL) is to learn on a sequence of tasks without forgetting previous ones. According to [19], the term "task" in CL refers to a distinct training phase that involves learning with a new collection of data. This data typically belongs to a different group of classes, a new data distribution, or involves a different output space compared to previous tasks. In other words, each task in CL represents a separate learning objective that the model needs to adapt to while retaining knowledge from previous tasks. Most CL methods assume knowing task identities and boundaries during training, named task-aware CL. This setting can be further categorized into task/domain/class-incremental CL [53]. Later, CL has been extended to a more general and challenging setup, i.e., task-free CL [2]. This learning scenario does not assume explicit task definition, and data distribution gradually evolves without clear task boundaries, making it applicable to a broader range of real-world problems.

The current widely adopted memory replay approaches, as discussed in references [1], [3], [14], [15], [46], involve the storage of a small portion of previous task data in memory. These stored examples are then replayed alongside new mini-batch data during training. However, this approach has inherent limitations. To begin with, the CL model is prone to overfitting the memory buffer, leading to diminish its effectiveness in mitigating forgetting. Memory overfitting [19], [26] refers to a situation in CL where a model becomes too closely fitted to the memory data, to the extent that it performs poorly on new, unseen data from previous tasks. In other words, the CL model learns to "memorize" the memory data examples instead of generalizing patterns and relationships that can be applied to new unseen data from previous tasks. This is primarily due to two factors. First, the capacity of the memory buffer is typically very limited, which prevents it from accurately representing all previously learned data examples. Second, the CL model repeatedly encounters and learns the data from the memory buffer, resulting in memorization rather than generalization of the knowledge contained within the memory data. This phenomenon is illustrated in Fig. 1(a). Consequently, previously acquired knowledge is at risk of being quickly lost. Moreover, there exists a significant gap between the memory data distribution and the distribution of all previous data examples since the memory data is only *a small subset of the entire data stream*. Most existing approaches [1], [3], [14], [15], [46] tend to overlook the high uncertainty present in the memory data distribution. This arises from the fact that a limited memory buffer cannot accurately reflect the stationary distribution of all examples encountered in the data stream, as depicted in Fig. 2.

To address the above issues, we propose a distributionally robust optimization framework (DRO) to evolve the memory

Manuscript received 19 October 2022; revised 17 June 2023; accepted 14 September 2023. Date of publication 22 September 2023; date of current version 3 November 2023. This work was supported in part by the NSF through under Grants IIS-1910492 and IIS-2239537. Recommended for acceptance by R. Cucchiara. *(Corresponding authors: Zhenyi Wang; Li Shen.)*

Zhenyi Wang is with the Department of Computer Science, University of Maryland, College Park, MD 20742 USA (e-mail: wangzhenyineu@gmail.com).

Li Shen is with JD Explore Academy, Beijing 101111, China (e-mail: mathshenli@gmail.com).

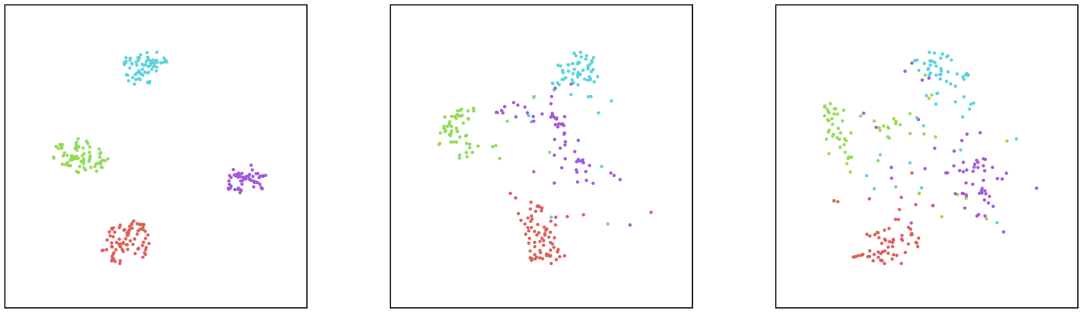
Tiehang Duan is with Meta Inc, Palo Alto, CA 94303 USA (e-mail: tiehang.duan@gmail.com).

Qiuling Suo, Le Fang, and Mingchen Gao are with the Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260 USA (e-mail: qiulings@buffalo.edu; lefang@buffalo.edu; mgao8@buffalo.edu).

Wei Liu is with Tencent Data Platform, Shenzhen 518057, China (e-mail: wl2223@columbia.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3317874, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3317874



(a) $t = t_0$
(easy to memorize and overfit)

(b) $t = t_1$
(diverse and hard)

(c) $t = t_2$
(diverse and harder to classify)

Fig. 1. T-SNE visualization of evolved memory embedded by ResNet18 on CIFAR10 at different CL timestamps. Each dot represents a data point's feature extracted by the last layer of ResNet18. Each color denotes one class of memory data. Initially, the classes are very easy to memorize and overfit. Memory evolution makes the memory more diverse and harder to classify and memorize.

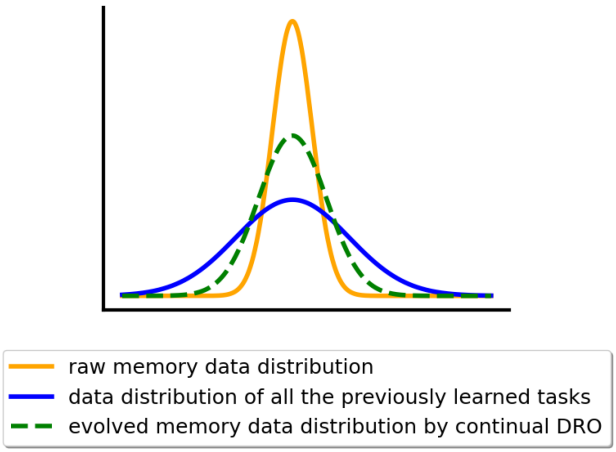


Fig. 2. Motivation of continual DRO. The blue line denotes the stationary distribution of all the previous data, and the orange line denotes the raw memory buffer data distribution. There is a big gap between the raw memory data distribution and the stationary distribution of all the data in the data stream. Traditional experience replay (ER) replays on the raw memory data. In contrast, our continual DRO (green line) fills this gap by evolving the memory data distribution to narrow the gap between the evolved memory data distribution and the distribution of all the previously learned data.

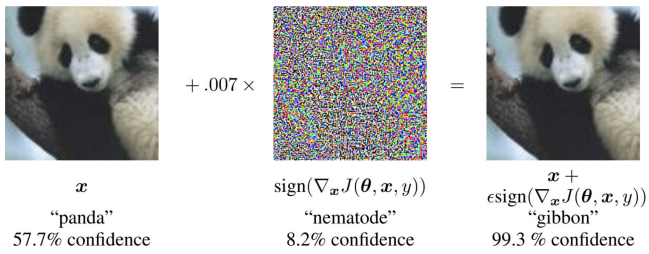


Fig. 3. Illustration of adversarial examples. This example is taken from [22], where the neural network initially correctly predicts the class of the input image as "panda". However, after introducing imperceptible noise to the image, the neural network incorrectly predicts the class of the image as "gibbon".

data distribution dynamically, named *Continual DRO*. This learning objective makes the memory data increasingly harder to memorize and helps the model alleviate memory overfitting [19], [26] and improve generalization, illustrated in Fig. 1(b) and (c). In addition, the proposed DRO framework considers the underlying high uncertainty of memory data distribution. It evolves memory data distribution to fill the gap between the memory data distribution and the ideal stationary distribution of all the previous data, illustrated in Fig. 2. We optimize the model performance under the worst-case evolved memory data distribution since we cannot know the exact distribution of all the previous examples. As a by-product of this worst-case optimization, the model can thus learn substantially more robust features with performance guarantees than previous work. An adversarial example refers to a specially crafted input or data instance that is intentionally designed to deceive or mislead machine learning models. These examples are created by making slight, imperceptible modifications to the input data with the intention of causing the model to make incorrect predictions or classifications [22], illustrated in Fig. 3. Our proposed DRO memory evolution framework is robust to such adversarial examples due to the worst-case performance optimization on the evolved memory data distribution.

We consider two types of distribution constraints that enforces the evolved memory data distribution not deviate from the raw memory data distribution too much in the Continual DRO, including more general f-divergence and Wasserstein ball constraints.

To efficiently and conveniently tackle the problem of *Continual DRO* with *f-divergence constraint*, we develop a systematic equation that converts the *Continual DRO* into a continuous dynamical system called *Dynamic DRO*. This enables a gradual continuous evolution of the initial distribution of raw memory data towards the desired worst-case memory data distribution. Furthermore, the *Dynamic DRO* provides a flexible framework for deriving various methods to facilitate the evolution of memory data distributions. These different methods offer flexibility and versatility in achieving the desired target memory data distribution, allowing for efficient and convenient solutions to the *Continual DRO*. We then introduce three specific memory evolution methods to solve the Dynamic DRO, including Langevin Dynamics [62], Stein Variational Gradient Descent [36], and Hamiltonian flow [34], [38]. The proposed memory evolution

framework is general, flexible, and easily extendable, with many potential extensions for future research.

For the proposed Continual DRO with *Wasserstein ball constraint*, it can model richer family of distributions in the neighbour of original raw memory data distribution than f-divergence, but it is computationally intractable to directly enforce Wasserstein-ball constraint [30] on the memory data distribution perturbation. There are two main reasons for the computational complexity. First, calculating the Wasserstein distance between two probability distributions is computationally expensive and has a complexity of $\mathcal{O}(n^3)$, where n represents the number of data points. Second, there is no closed-form solution available to calculate its gradient since the Wasserstein distance is defined as a constrained optimization problem over the probability distribution space. To address these challenges, we convert the Continual DRO with Wasserstein-ball constraint into a surrogate loss function in the euclidean space. This surrogate loss function approximates the Wasserstein-ball constraint, enabling the continual DRO to be solved in the euclidean space, which is computationally more tractable.

We evaluate the effectiveness of the proposed framework by performing comprehensive experiments on several datasets and comparing them to various strong SOTA baselines. We summarize our contributions as follows:

- We propose the first principled, general, and flexible *memory evolution* framework for both task-aware and task-free CL from the perspective of distributionally robust optimization, named *continual DRO*. Our proposed method is substantially more effective for mitigating forgetting. As a by-product of the proposed DRO framework, our method is more robust to adversarial examples than previous CL methods.
- We instantiate the proposed Continual DRO with both more general f-divergence and Wasserstein ball constraints to ensure the evolved memory data not deviate from the original raw memory data too much, providing great flexibility for specifying the optimization objective for the CL learner.
- We formulate the Continual DRO from a new continuous dynamics perspective and cast it as a gradient flow system, named *Dynamic DRO*. We propose a family of memory evolution methods with different ways to efficiently solve the Dynamic DRO, opening up a new research direction for presenting new strategies to evolve the memory data.
- Extensive experiments on several datasets for both task-aware and task-free CL demonstrate the effectiveness of the proposed method for mitigating forgetting and increasing the robustness to adversarial examples. Our framework is versatile and can be seamlessly integrated with existing memory-replay-based methods to improve their performance.

In this paper, we extend our previous work [61] from both theoretical and empirical aspects. Theoretical aspect: our previous work [61] only considered task-free DRO under the constraint of KL-divergence. However, in this paper, we extend the constraint to encompass more general f-divergence and Wasserstein-ball constraints. Our previous work [61] with KL-divergence can be viewed as a special case of f-divergence. By including these additional constraints, the theoretical aspects of the paper become more comprehensive. This expansion allows for a broader understanding of DRO for CL. Empirical aspect: this paper largely extends the application scenarios of the proposed framework. Our previous work [61] only considered task-free CL, but in this paper, we additionally explore both task-aware and task-free CL. This extension makes the empirical applicability of the paper more general, as it covers a wider range of CL scenarios.

II. Related Work

In this section, we discuss the related CL works [28], [37], [41], [42], [45], [49], [55], [57], [58], [59], [60], [66], which can be categorized into: (1) task-aware CL in Section II-A, where there are clear task definitions and boundaries among the sequentially learned tasks; (2) task-free CL [2], [18], [25], [67] in Section II-B, which focuses on the more general case where the data distribution could change arbitrarily without explicit task splits. We then discuss the related work of distributionally robust optimization (DRO) in Section II-C.

A. Task-Aware CL

Existing works proposed for solving task-aware CL have four major research branches: 1) keeping a memory buffer that stores the data examples from previous tasks and replay those examples during learning new tasks, representative works include [1], [14], [15], [43], [46], [52]; 2) designing dynamic network architectures during CL [21], [47], [65] to retain the previously learned knowledge; 3) adding regularization loss term to avoid forgetting the previous knowledge [28], [55], [66]; and 4) using Bayesian methods to model the parameter update uncertainty and maintain the model parameter distribution unchanged, including [20], [41]. In this paper, we focus on memory-replay-based methods since they often achieve better performance than the methods of other categories. Memory-based CL methods include experience replay [15], which trains the new task with the memory buffer data together. Meta Experience Replay (MER) [46] uses meta-learning to encourage information transfer from previous tasks and minimize interference. Hindsight Anchor Learning (HAL) [13] proposes to use anchor points to mitigate forgetting on previous tasks. GEM [37] and A-GEM [14] use the memory buffer data losses as inequality constraints to constrain their increase but allowing their decrease to mitigate forgetting. DER [9] further integrates experience replay with knowledge distillation. LiDER [7] aims to smooth the decision boundary of replayed samples by restricting the Lipschitz constant of the CL neural network. LiDER is orthogonal to our work since they focuses on model space regularization. In contrast, our method focuses on data space regularization.

Data augmentation is a technique used to generate additional samples from existing data distributions. However, traditional data augmentation methods are typically manually designed and predefined, which limits their adaptability to CL where the data distribution changes with time. Another approach called generative replay [52] requires training a generative model to remember previous data distributions. However, generative

models are challenging and unstable to train, resulting in poorly generated examples. Controlling the quality of generated samples becomes difficult, often deviating from the original raw data distribution when learning new tasks. Additionally, generative models are prone to forgetting previously learned distributions, which hampers their performance compared to memory-based methods [26].

B. Task-Free CL

Existing approaches for task-free CL can be categorized into two classes. The first (majority) one is memory-based methods [1], [3], which store a small number of data from the previous data stream and replay them with the new mini-batch data. The second type is the expansion-based method, such as CN-DPM [31]. However, CN-DPM needs to increase the memory and computation overhead with the network structure's expansion and the increase of the network parameters. Hence, this work focuses on memory-replay-based methods without expanding the network structure since it is simple and effective. Most existing works in this category [14], [15] directly perform replay on the raw data without any adaptation. MIR [1] proposes to replay the samples with which are most interfered. GEN-MIR [1] further uses generative models to synthesize the memory examples. The *heuristic* method, GMED [26], proposes to edit memory examples so that the examples are more likely forgotten and harder to memorize. Our methods share similar motivations with GMED, which *individually* edits the memory data without considering memory data distribution uncertainty and population-level statistics. In contrast, our DRO framework focuses on *population-level and distribution-level evolution*. Orthogonal to our work, Gradient-based Sample Selection (GSS) [3] focuses on *storing diverse examples*. These methods lack theoretical guarantees. In contrast, our framework is principled and focuses on evolving memory data distributions.

C. Distributionally Robust Optimization (DRO)

DRO is an effective optimization framework to handle decision-making under uncertainty [44]. The basic idea of DRO is first to construct a set of probability distributions as an ambiguity set and minimize the worst-case performance in this ambiguity set, thus guaranteeing the model performance. There are various applications of DRO in machine learning problems, including tackling the group-shift [48], subpopulation shift [68], and class imbalances [64].

To our best knowledge, our work is the first principled method with DRO for memory evolution in CL, i.e., continual DRO. It dynamically evolves memory data distributions to avoid forgetting and learn robust features. In addition, we formulate the proposed continual DRO from a new continuous dynamics perspective, making it convenient to handle the evolving memory data distribution with different evolution dynamics. Besides, we propose several ways to evolve the memory data, opening up a new research direction to explore more effective memory evolution methods.

III. Problem Setup

In this section, we provide problem definition for task-aware CL in Section III-A and task-free CL in Section III-B.

A. Task-Aware CL

Task-aware CL aims to solve the case where there are explicit task definitions during CL. The three most common CL scenarios are task/domain/class-incremental learning [53]. We consider the standard CL problem of learning a sequence of N tasks denoted as $\mathcal{D}^{tr} = \{\mathcal{D}_1^{tr}, \mathcal{D}_2^{tr}, \ldots, \mathcal{D}_N^{tr}\}$. The training data of kth task $\mathcal{D}_k^{tr}$ consists of a set of triplets $\{(\boldsymbol{x}_i^k, y_i^k, \mathcal{T}_k)_{i=1}^{n_k}\}$, where $\boldsymbol{x}_i^k$ is the ith task data example, y_i^k is the data label associated with $\boldsymbol{x}_i^k$, and $\mathcal{T}_k$ is the task identifier. The goal is to learn a network function with parameters $\boldsymbol{\theta}$, i.e., $g(\boldsymbol{x}, \boldsymbol{\theta})$, on the training task sequence $\mathcal{D}^{tr}$ so that it performs well on the test set of all the learned tasks $\mathcal{D}^{te} = \{\mathcal{D}_1^{te}, \mathcal{D}_2^{te}, \ldots, \mathcal{D}_N^{te}\}$ without forgetting the knowledge of previous tasks.

B. Task-Free CL

A sequence of mini-batch labeled data $(\boldsymbol{x}_k, y_k, h_k)$ sequentially arrives at each timestamp k and forms a non-stationary data stream. Each data point is associated with a latent task identity h_k, where $\boldsymbol{x}_k$ denotes the mini-batch data received at timestamp k, y_k is the data label associated with $\boldsymbol{x}_k$. During both the training and testing time, the task identity h_k is not available to the learner. According to [2], a more general definition of task-free CL is that data distribution shift could happen at any time without explicit task splits. Our method can be directly applied to those more general cases as well. At the same time, a small memory buffer $\mathcal{M}$ is maintained, and replay the data in $\mathcal{M}$ when learning the new task to avoid forgetting the previously learned knowledge. The memory buffer $\mathcal{M}$ is updated by reservoir sampling (RS), similar to the procedure in [46].

IV. Method

In this section, we present the traditional memory replay in Section IV-A. We propose our general *Continual DRO* framework in Section IV-B. Next, we instantiate the *Continual DRO* framework with general f-divergence constraint in Section IV-C. Finally, we present another variant with Wasserstein-ball constraint for continual DRO in Section IV-D. We summarize the proposed methods in Section IV-E.

A. Traditional Memory Replay for CL

The goal is to learn a model $g(\boldsymbol{x}, \boldsymbol{\theta})$ to perform well on all the tasks seen until timestamp k. Standard memory replay for CL [15] is to optimize an objective under a known probability distribution μ_0. Formally speaking, CL with traditional memory replay can be expressed as

$$\min_{\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}} [\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_k, y_k) + \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu_0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y)], \tag{1}$$

where $\boldsymbol{\theta}$ are model parameters and $\boldsymbol{x}$ is the raw memory buffer data with probability measure (density) μ_0, i.e., $\forall \boldsymbol{x} \in \mathcal{M}, \boldsymbol{x} \sim$

μ_0. $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y)$ is the loss function associated with the data $(\boldsymbol{x}, y)$. In the following, we temporally omit the term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_k, y_k)$ due to the fact that $(\boldsymbol{x}_k, y_k)$ is the mini-batch data arrived at timestamp k and does not depend on the memory data distribution.

B. DRO for General CL

Distributionally Robust Optimization (DRO) [44] is a systematic and elegant framework to model the decision-making with ambiguity in the underlying probability distribution. Traditional memory-replay methods in CL in Section III implicitly assume that μ_0 is known. In contrast, our proposed DRO framework takes that the underlying actual probability distribution of memory data μ is *unknown* and lies in an ambiguity set of probability distributions. Modeling the memory data uncertainty is particularly useful when the memory buffer is small compared to the whole dataset since the memory has limited coverage to approximate the stationary distribution of all examples seen so far, illustrated in Fig. 2. Thus, there is significant uncertainty in modeling the multi-task learning scenarios (the performance upper bound) with only a small memory buffer. The proposed DRO framework optimizes the worst-case performance in the ambiguity set of probability distributions since we cannot access the distribution of all the previous data. Therefore, it helps the model generalize to previous tasks since it can potentially narrow the gap between the memory data distribution and the distribution of all the previous data, illustrated in Fig. 2. As a by-product of this optimization framework, it also helps learn features robust to data distribution perturbations. On the other hand, the memory buffer is updated slowly during CL (e.g., by reservoir sampling), and traditional memory-replay repeatedly trains the memory buffer. As a result, the CL model can easily overfit the memory buffer, as illustrated in Fig. 1(a). Thus, the memory buffer could become less effective for mitigating forgetting. By optimizing the worst-case evolved memory data distribution at each iteration, our proposed DRO can also alleviate the memory overfitting problem by transforming the memory data to make them more difficult to memorize. This is illustrated in Fig. 1(b) and (c). Mathematically speaking, the proposed DRO for general CL can be expressed as

$$\min_{\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{\mu \in \mathcal{P}} \mathbb{E}_{\mu} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \tag{2}$$

$$\text{s.t. } \mathcal{P} = \{\mu : \mathcal{D}(\mu_0, \mu) \leq \mathcal{D}(\mu_0, \pi) \leq \epsilon\}, \tag{3}$$

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu, \boldsymbol{x}' \sim \mu_0} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y) \geq \lambda, \tag{4}$$

where the inner sup optimization is to gradually make the memory data distribution increasingly harder to memorize. $\mathcal{P}$ in (3) denotes the ambiguity set of probability measures (distributions or densities) for the memory data distribution to characterize its uncertainty. We define $\mathcal{P}$ through f- divergence or Wasserstein distance. $\mathcal{D}(\mu_0, \pi)$ denotes the f- divergence or Wasserstein distance between probability measure μ_0 and π, where π is the target worst-case evolved memory data distribution, i.e., $\pi = \arg\max_{\mu \in \mathcal{P}} \mathbb{E}_{\mu} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y)$. Where ϵ in (3) is a constant threshold to characterize the closeness between μ_0 and π to ensure the worst-case evolved memory data distribution π does not deviate from the raw memory data distribution μ_0 too much. In (4) constrains the gradient dot product between the worst-case evolved memory data distribution and raw memory data distribution, i.e., $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y)$, to avoid the evolved memory data deviate from the raw memory data too much. Intuitively, if the gradient dot product is positive, it means the evolved memory data has a similar update direction compared to the raw data. λ is a threshold to determine the constraint magnitude. An illustration of our proposed method is shown in Fig. 4.

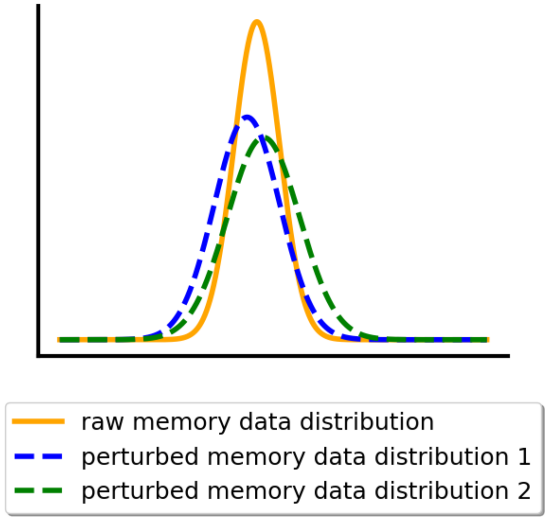


Fig. 4. Illustration of our proposed continual DRO framework. Traditional memory replay only optimizes the loss on the raw memory data distribution (indicated with orange color). Our proposed framework incorporates the uncertainty in the underlying memory data distribution, optimizing the worst-case performance within the neighborhood of the raw memory data distribution (indicated with dotted blue and green color). The evolved memory data covers more data space.

To solve the optimization, i.e., (2)–(4), the worst-case optimization that involves the sup optimization is generally computationally intractable since it involves the optimization over infinitely many probability distributions. To address this problem, by Lagrange duality [8], we convert (2)–(4) into the following unconstrained optimization problem (detailed derivations are put in Appendix A, available online:

$$\min_{\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{\mu} [\mathbb{E}_{\mu} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) - \gamma \mathcal{D}(\mu_0, \mu) + \beta \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu, \boldsymbol{x}' \sim \mu_0} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y)], \tag{5}$$

where γ and β control the magnitude of regularization. The gradient of the third term (gradient dot product term) can be efficiently approximated by finite difference in practice. The optimization (5) is still computationally hard to solve because the inner sup optimization is over probability measure space, which is an infinite-dimensional *function space*. We name (5) as *Continual DRO*.

C. Continual DRO With f-Divergence Constraint

In this section, we propose the solution for Continual DRO with general f-divergence constraint. Specifically, we first present the f-divergence Wasserstein Gradient Flow, which will be used for efficiently solving the Continual DRO in Section IV-C1. We then present the reformulation of *Continual DRO* from a continuous dynamics view as *Dynamic DRO* in Section IV-C2. We next present specific memory evolution methods to solve the Dynamic DRO efficiently in Section IV-C3.

1) f-Divergence Wasserstein Gradient Flow: f-divergence is defined as

$$\mathcal{J}(\mu||\nu) = \int f\left(\frac{\mu(\boldsymbol{x})}{\nu(\boldsymbol{x})}\right)\nu(\boldsymbol{x})d\boldsymbol{x}, \tag{6}$$

where f is a is convex, lower-semicontinuous function with $f(1) = 0$. f-divergence encapsulates many common distribution divergences, including KL divergence when $f(\phi) = \phi\log\phi$, χ^2 divergence when $f(\phi) = (\phi-1)^2$ and α-divergence when $f(\phi) = \frac{4}{1-\alpha^2}(1-\phi^{\frac{1+\alpha}{2}})$.

Below, we define and present the gradient flow of the above defined f-divergence in Wasserstein space. Let $\mathcal{P}_2(\mathbb{R}^d)$ denote the space of probability measures on $\mathbb{R}^d$ with finite second-order moments. Each element $\mu \in \mathcal{P}_2(\mathbb{R}^d)$ is a probability measure represented by its density function $\mu : \mathbb{R}^d \to \mathbb{R}$ with respect to Lebesgue measure dx. The Wasserstein distance between two probability measures $\mu_1, \mu_2 \in \mathcal{P}_2(\mathbb{R}^d)$ is defined as

$$W_2(\mu_1,\mu_2) = \left(\min_{\omega\in\prod(\mu_1,\mu_2)}\int ||\boldsymbol{x}-\boldsymbol{x}'||^2 d\omega(\boldsymbol{x},\boldsymbol{x}'))\right)^{1/2},$$

where $\prod(\mu_1,\mu_2) = \{\omega|\omega(A\times\mathbb{R}^d) = \mu_1(A), \omega(\mathbb{R}^d\times B) = \mu_2(B)\}$. ω is the joint probability measure with marginal measure of μ_1 and μ_2 respectively. Thus, $\mathbb{W}^2 = (\mathcal{P}_2(\mathbb{R}^d), W_2)$ forms a metric space.

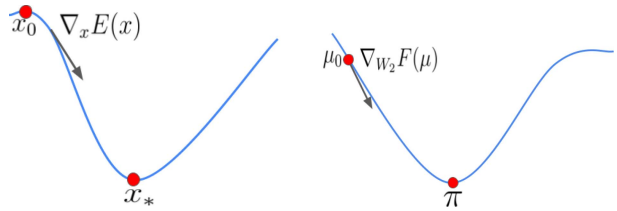
Definition 4.1 (Gradient Flow in euclidean Space):

$$\frac{d\boldsymbol{x}_t}{dt} = -\nabla E(\boldsymbol{x}_t), \quad \text{where} \quad \boldsymbol{x}_0 = \boldsymbol{x}(0). \tag{7}$$

Where the function $E(\boldsymbol{x}_t)$ is the function to be optimized. The gradient flow in euclidean Space is the solution to this differential equation.

The gradient flow in euclidean space basically describes that the data $\boldsymbol{x}_t$ follows the steepest descent direction of the function $E(\boldsymbol{x}_t)$ at any time t, which is the generalization of the discrete gradient descent in continuous limit. We next present Wasserstein gradient flow in probability measure space, which is the generalization of gradient flow in euclidean Space to Wasserstein space.

We first provide the definition of first variation by calculus of variation [40]. The first variation of a *functional* in function space is analogous to the first-order gradient of a *function* in euclidean distance. For more detailed background knowledge of calculus of variations, we recommend the reader refer to [40]. It is defined as below.



(a) Gradient flow in Euclidean space (b) Gradient flow in Wasserstein space

Fig. 5. Illustration of gradient flow in euclidean and Wasserstein space. (a) gradient flow in euclidean space is a continuous curve (blue) where the optimization variable follows the steepest descent direction at each point. (b) gradient flow in Wasserstein space is also a curve where the probability measure follows the steepest descent direction (with Wasserstein gradient) to move towards the target probability measure. The difference is that in euclidean space, each $\boldsymbol{x}$ is a finite dimensional object. By contrast, in Wasseerstein probability measure space, each probability measure μ is an infinite dimensional function.

Definition 4.2 (First Variation): The first variation of a functional $F(\mu)$ is the functional at μ

$$\frac{\delta F}{\delta\mu}(\mu) = \lim_{\epsilon\to 0}\frac{F(\mu+\epsilon\psi)-F(\mu)}{\epsilon}, \tag{8}$$

where ψ is an arbitrary function.

Definition 4.3 (Wasserstein Gradient Flow [4]): Suppose we have a Wasserstein space $\mathbb{W}^2 = (\mathcal{P}_2(\mathbb{R}^d), W_2)$. A curve $(\mu_t)_{t\geq 0}$ of probability measures is a Wasserstein gradient flow for functional F if it satisfies

$$\partial_t\mu_t = \nabla_{W_2}F(\mu_t) := \nabla\cdot\left(\mu_t\nabla\frac{\delta F}{\delta\mu}(\mu_t)\right), \tag{9}$$

where $:=$ means defined as, and $\nabla\cdot(\boldsymbol{r}) := \sum_{i=1}^d \partial_{\boldsymbol{z}^i}\boldsymbol{r}^i(\boldsymbol{z})$ is the divergence operator of a vector-valued function $\boldsymbol{r}:\mathbb{R}^d\to\mathbb{R}^d$, where $\boldsymbol{z}^i$ and $\boldsymbol{r}^i$ are the i th element of $\boldsymbol{z}$ and $\boldsymbol{r}$; ∇ is the gradient of a scalar-valued function. $\nabla_{W_2}F(\mu_t) := \nabla\cdot(\mu_t\nabla\frac{\delta F}{\delta\mu}(\mu_t))$ is the Wasserstein gradient of functional F at μ_t, where $\frac{\delta F}{\delta\mu}(\mu_t)$ is the first variation of F at μ_t.

Intuitively, the WGF describes that the probability measure μ_t follows the steepest curve of the functional $F(\mu)$ in Wasserstein space of probability measures (function space) to gradually move (starting at the initial probability measure μ_0, i.e., the raw memory data) towards the target probability measure π, i.e., the target evolved memory data distribution. To illustrate the WGF, we compare it with gradient flow in euclidean space in Fig. 5.

Theorem 4.4: Given a target memory data distribution π, the gradient flow of f-divergence $\mathcal{J}(\mu||\nu)$ evolves as the following partial differential equation (PDE) in Wasserstein space according to the Definition 4.3

$$\partial_t\mu(\boldsymbol{x},t) = \nabla\cdot\left(f''(\mu)(\boldsymbol{x},t)^{-1}\nabla f'\left(\frac{\mu}{\pi}\right)(\boldsymbol{x},t)\right). \tag{10}$$

In particular, if $f''(\phi) = f(1)\phi^{-\rho}$ (ρ is a constant), the above (10) is equivalent to the following one

$$\partial_t\mu(\boldsymbol{x},t) = \nabla\cdot\left(\pi(\boldsymbol{x})^\rho\nabla\left(\frac{\mu(\boldsymbol{x},t)}{\pi(\boldsymbol{x})}\right)\right). \tag{11}$$

In sample space, the memory data particles satisfy the following stochastic differential equation (SDE):

$$dX_t = \rho\pi(X_t)^{\rho-2}\nabla\pi_t(X_t)dt + \sqrt{2\pi(X_t)^{\rho-1}}dW_t, \quad (12)$$

where $W = (W_t)_{t\geq 0}$ is the standard Brownian motion in $\mathbb{R}^n$ [5].

The proof follows from [32].

In the following, we instantiate our framework by specializing the f-divergence to be KL divergence when $f(\phi) = \phi\log\phi$, χ^2 divergence when $f(\phi) = (\phi-1)^2$ and α-divergence when $f(\phi) = \frac{4}{1-\alpha^2}(1-\phi^{\frac{1+\alpha}{2}})$.

KL-Divergence WGF: When dealing with KL-divergence, we need to instantiate the $f(\phi) = \phi\log\phi$. The first and second order derivative are $f'(\phi) = \log\phi + 1$ and $f''(\phi) = \frac{1}{\phi}$, respectively. $\rho = 1$. According to (11) and (12), the WGF and corresponding diffusion process can be formulated as

$$\partial_t\mu(\boldsymbol{x},t) = \nabla\cdot\left(\pi(\boldsymbol{x})\nabla\left(\frac{\mu(\boldsymbol{x},t)}{\pi(\boldsymbol{x})}\right)\right) \quad (13)$$

$$dX_t = \pi(X_t)^{-1}\nabla\pi(X_t)dt + \sqrt{2}dW_t. \quad (14)$$

In (14) can be further reformulated as

$$dX_t = \nabla\log\pi(X_t)dt + \sqrt{2}dW_t. \quad (15)$$

χ^2-Divergence WGF: When dealing with χ^2-divergence, we need to instantiate the $f(\phi) = (\phi-1)^2$. The first and second order derivative are $f'(\phi) = 2(\phi-1)$ and $f''(\phi) = 2$, respectively. $\rho = 0$. According to (11) and (12), the WGF and corresponding diffusion process can be formulated as

$$\partial_t\mu(\boldsymbol{x},t) = \nabla\cdot\left(\nabla\left(\frac{\mu(\boldsymbol{x},t)}{\pi(\boldsymbol{x})}\right)\right) \quad (16)$$

$$dX_t = \sqrt{2\pi(X_t)^{-1}}dW_t. \quad (17)$$

α-Divergence WGF: In this case, we set the f function to be $f(\phi) = \frac{4}{1-\alpha^2}(1-\phi^{\frac{1+\alpha}{2}})$. According to (11) and (12), the WGF and corresponding diffusion process can be formulated as

$$\partial_t\mu(\boldsymbol{x},t) = \nabla\cdot\left(\pi(\boldsymbol{x})^{\frac{3-\alpha}{2}}\nabla\left(\frac{\mu(\boldsymbol{x},t)}{\pi(\boldsymbol{x})}\right)\right) \quad (18)$$

$$dX_t = \frac{3-\alpha}{2}\pi(X_t)^{\frac{-1-\alpha}{2}}\nabla\pi(X_t)dt + \sqrt{2\pi(X_t)^{\frac{1-\alpha}{2}}}dW_t. \quad (19)$$

Remark: Among the above various divergence instantiation of the f-divergence, KL-divergence is the most widely used and studied divergence metric. χ^2 divergence enjoys faster convergence than the KL-divergence [17]. On the other hand, the nice property of α-divergence is its mass-covering [56] property, i.e., the evolved memory data distribution tends to cover more modes of the target memory data distribution.

2) Continual DRO. A Continuous Dynamics View: In this section, we instantiate the distance $\mathcal{D}(\mu_0,\mu)$ in (5) to be f-divergence $\mathcal{J}(\mu||\mu_0)$. The optimization goal then becomes the following:

$$\min_{\forall\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sup_{\mu}[\mathbb{E}_\mu\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y) - \gamma\mathcal{J}(\mu||\mu_0) + \beta\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mu,\boldsymbol{x}'\sim\mu_0}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y)\cdot\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x}',y)]. \quad (20)$$

To make it tractable to solve the continual DRO in (20), we formulate it from a new continuous dynamics perspective. This new perspective brings significant advantages over directly solving (20): 1) the optimization of (20) over probability measure space can be converted into a continuous probability distribution evolution, equivalent as a WGF, enabling gradient-based solutions in *probability measure space (function space)*; 2) we can efficiently solve the WGF by various methods, which provide potential derivations of many different memory evolution methods. We then propose a novel solution by decomposing the continual DRO (20) into a gradient flow system. Specifically, we solve the inner sup optimization problem for memory evolution with WGF in Wasserstein space of probability measures and solve the outer optimization problem for the model parameters with gradient flow in euclidean space. We convert the continual DRO into a gradient flow system that alternately updates the memory evolution and model parameters.

Given a memory buffer at timestamp k, the raw memory data is $\mathcal{M} = \{(\boldsymbol{x}_0^1,y^1),(\boldsymbol{x}_0^2,y^2),\ldots,(\boldsymbol{x}_0^N,y^N)\}$, where N is the memory buffer size. We perform a similar memory evolution procedure at each CL timestamp and omit the CL timestamp k for notation clarity. We denote $\boldsymbol{x}_t^i$ as the ith datapoint in the evolved memory buffer after t evolution steps. The raw memory data is assumed to be i.i.d. sampled from the random variable X_0, i.e., $\{\boldsymbol{x}_0^1,\boldsymbol{x}_0^2,\ldots,\boldsymbol{x}_0^N\}\sim X_0$. X_0 follows the probability distribution μ_0, i.e., $X_0\sim\mu_0$. At evolution time t, the memory data $\mathcal{M}$ is evolved as random variable X_t, i.e., $\{\boldsymbol{x}_t^1,\boldsymbol{x}_t^2,\ldots,\boldsymbol{x}_t^N\}\sim X_t$ and probability distribution of random variable X_t follows the probability measure μ_t, i.e., $X_t\sim\mu_t$. The empirical measure on the evolved memory buffer at time t is defined as $\hat{\mu}_t = \frac{1}{N}\sum_{i=1}^{i=N}\delta(\boldsymbol{x}_t^i)$ and δ is the Dirac measure. We model the memory evolution process as continuous WGF in probability measure space, i.e., use $(\mu_t)_{t\geq 0}$ to model the probability distribution evolution of memory data. The evolving $(\mu_t)_{t\geq 0}$ will in turn determine the memory evolution process $(X_t)_{t\geq 0}$ in euclidean space.

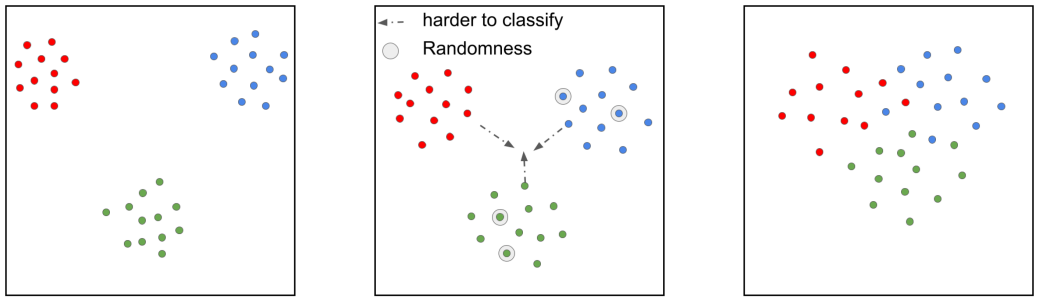
The f-divergence term $\mathcal{J}(\mu||\pi)$ is implicitly handled by WGF, we thus set $\gamma = 1$ throughout this paper. Since the goal is to approach the target distribution π as close as possible, we then define the energy functional $F(\mu)$ for memory evolution as the following:

$$\begin{aligned} F(\mu) &= V(\mu) + \mathcal{J}(\mu||\pi) \\ V(\mu) &= -\mathbb{E}_\mu\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y) \\ &\quad - \beta\mathbb{E}_\mu\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y)\cdot\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x}',y). \end{aligned} \quad (21)$$

By defining such energy functional $F(\mu)$, the (20) can be equivalently solved by the following gradient flow system (22), (23), we name it as *Dynamic DRO*.

$$\begin{cases} \partial_t\mu_t = div\left(\mu_t\nabla\frac{\delta F}{\delta\mu}(\mu_t)\right); & (22) \\ \frac{d\boldsymbol{\theta}}{dt} = -\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mu_t}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y), & (23) \end{cases}$$

where (22) solves the inner sup problem in (20) with WGF in Wasserstein space and (23) solves the outer minimization problem in (20) for parameter update with gradient flow in euclidean space. In the following, we focus on solving (22) and (23).



(a) $t = t_0$ (b) $t = t_1$ (c) $t = t_2$

Fig. 6. Illustration of WGF-LD for memory evolution at different time $t_0 < t_1 < t_2$. Initially ($t = t_0$), the raw memory data is easy to overfit. From $t = t_1$ to $t = t_2$, the memory data becomes harder to classify and overfit. The black arrow (corresponds to the first term ($\nabla_{\boldsymbol{x}} U(\boldsymbol{x}_t^i, \boldsymbol{\theta})$)), drives the memory data to become harder to classify. The white circle (corresponds to the second term (Brownian motion)) serves as a random force such that the memory data becomes more diverse.

3) Training Algorithm for Dynamic DRO With KL-Divergence: In this section, we instantiate the f-divergence as KL-divergence, and other special f-divergence derivations are similar. We propose three different methods for efficiently solving the Dynamic DRO in (22)–(23). The first solution is Langevin dynamics with a diffusion process to evolve the memory data distribution; we name this method WGF-LD. Next, different from WGF-LD, which relies on *randomness* to transform the memory data, we kernelize the WGF in (22) by solving the WGF in reproducing kernel Hilbert space (RKHS). It *deterministically* transforms the memory data; we name this method WGF-SVGD. Furthermore, we generalize the above WGF and improve their flexibility to incorporate prior knowledge or geometry information. One instantiation of this general WGF uses Hamiltonian dynamics; we name it WGF-HMC. More novel memory evolution methods are worth exploring in future work.

Following [63], we define the target distribution π as the energy function $\pi \propto^{-U}$, where the function U is defined as the following:

$$U(\boldsymbol{x}, \boldsymbol{\theta}) = -\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) - \beta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y).$$

Langevin Dynamics for Dynamic DRO: If we directly use the energy functional $F(\mu)$ (21) in (22), solving the SDE (15) corresponds to the Langevin dynamics with the following stochastic differential equation [62] by replacing the π with e^{-U} in (15):

$$dX = -\nabla_X U(X, \boldsymbol{\theta}) dt + \sqrt{2} dW_t, \tag{24}$$

where $X = (X_t)_{t \geq 0}$ is the memory evolution process as previously defined. $W = (W_t)_{t \geq 0}$ is the standard Brownian motion in $\mathbb{R}^n$ [5]. If $X_t \sim \mu_t$ evolves according to the Langevin dynamics (24) in euclidean space, then $\mu_t(\boldsymbol{x})$ evolves according to gradient flow (22) in the space of probability measures [27]. If we discretize the above equation and view each datapoint in memory as one particle, the memory buffer data evolves with Langevin dynamics to obtain diverse memory data by the following updates:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = -h(\nabla_{\boldsymbol{x}} U(\boldsymbol{x}_t^i, \boldsymbol{\theta})) + \sqrt{2\,h}\xi_t. \tag{25}$$

As illustrated in Fig. 6, the first term in (25) drives the particles (memory data) towards the worst-case memory data distribution π to make the memory data gradually harder to memorize by dynamically increasing the energy functional. For the second term, it adds noise to encourage diversity in the transformed memory data, where ξ_t is standard Gaussian noise, and h is step size, i.e., a proper amount of added Gaussian noise tailored to the used step size. We name this memory evolution method as *WGF-LD*.

Kernelized Method for Dynamic DRO: We replace the Wasserstein gradient $\nabla_{W_2} F(\mu_t)$ by the integration transformation $\mathcal{K}_\mu \nabla_{W_2} F(\mu_t) = \int K(\boldsymbol{x}, \boldsymbol{x}') \nabla_{W_2} F(\mu_t)(\boldsymbol{x}') d\mu(\boldsymbol{x}')$; where the RKHS space induced by the kernel K is denoted by $\mathcal{H}$. The probability measure in the kernelized Wasserstein space actually follows the kernelized WGF [35]:

$$\partial_t \mu_t = div\left(\mu_t \mathcal{K}_{\mu_t} \nabla \frac{\delta F}{\delta \mu}(\mu_t)\right). \tag{26}$$

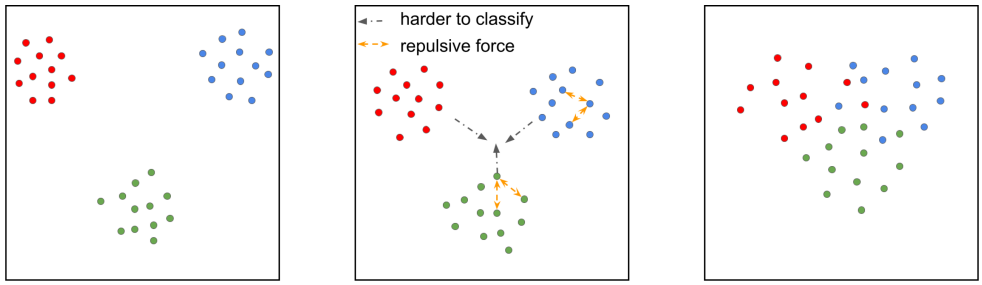
In (26) can be viewed as the WGF (22) in RKHS. It indicates that the random variable X_t which describes the evolved memory data at time t evolves as the following differential equation [35]:

$$\frac{dX}{dt} = -\left[\mathcal{K}_\mu \nabla \frac{\delta F}{\delta \mu}(\mu_t)\right](X). \tag{27}$$

This kernelized version is the deterministic approximation of the WGF in (22) [36]. If we discretize the above equation and view each datapoint in memory as one particle, we can obtain the following memory evolution update equation:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = -\frac{h}{N} \sum_{j=1}^{j=N} [\underbrace{k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j) \nabla_{\boldsymbol{x}_t^j} U(\boldsymbol{x}_t^j, \boldsymbol{\theta})}_{\text{smoothed gradient}} + \underbrace{\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)}_{\text{repulsive term}}], \tag{28}$$

As illustrated in Fig. 7, the first term drives the memory data towards the worst-case memory data distribution by increasing the energy functional. The update is driven by the kernel weighted sum of the gradients from the memory data points, thus smoothing the memory data gradients. The second term serves as a repulsive force that prevents the memory data points from collapsing into a single mode, thus diversifying the memory data population. In this paper, we use Gaussian kernel



(a) $t = t_0$ (b) $t = t_1$ (c) $t = t_2$

Fig. 7. Illustration of WGF-SVGD for memory evolution at different time $t_0 < t_1 < t_2$. Initially ($t = t_0$), the raw memory data is easy to overfit. From $t = t_1$ to $t = t_2$, the memory data becomes harder to classify and overfit. The black arrow (the first term $(k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)\nabla_{\boldsymbol{x}_t^j} U(\boldsymbol{x}_t^j, \boldsymbol{\theta}))$ in (28)) drives the memory data to become harder to classify. The orange arrow (the second term $(\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j))$ in (28)) serves as repulsive force such that the memory data becomes more diverse.

$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^2}{2\sigma^2})$. We name this memory evolution method as *WGF-SVGD*. We put detailed derivations in Appendix C, available online.

General Memory Evolution for Dynamic DRO: [38] found that any continuous Markov process that provides samples from the target distribution can be written in a very general sampler form. The corresponding general WGF for memory evolution can be written as

$$\partial_t \mu_t = div\left(\mu_t(\boldsymbol{D} + \boldsymbol{Q})\nabla \frac{\delta F}{\delta \mu}(\mu_t)\right), \tag{29}$$

where $\boldsymbol{D}$ is a positive semidefinite diffusion matrix, $\boldsymbol{Q}$ is a skew-symmetric curl matrix representing the deterministic traversing effect [38]. One particular case is

$$\boldsymbol{D} = \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{C} \end{pmatrix}, \boldsymbol{Q} = \begin{pmatrix} 0 & -\boldsymbol{I} \\ \boldsymbol{I} & 0 \end{pmatrix},$$

where $\boldsymbol{C}$ is the friction term, and $\boldsymbol{I}$ is the identity matrix. This WGF corresponds to Hamiltonian dynamics [16] and can be solved by the following evolution:

$$\begin{cases} \boldsymbol{x}_{t+1} - \boldsymbol{x}_t = \boldsymbol{v}_t, \\ \boldsymbol{v}_{t+1} - \boldsymbol{v}_t = -h\nabla U(\boldsymbol{x}, \boldsymbol{\theta}) - \tau \boldsymbol{v} + \sqrt{2\tau h}\xi_t, \end{cases} \tag{30}$$

where $\boldsymbol{v}$ is the momentum variable, and τ is the momentum weight. We name this method as *WGF-HMC*.

The most attractive property of this WGF for memory distribution evolution is that we can freely specify the matrix $\boldsymbol{Q}$ and $\boldsymbol{D}$ tailored to practical requirements. We can consider prior knowledge or geometry information by designing tailored $\boldsymbol{Q}$ and $\boldsymbol{D}$, or develop the kernelized version of this general WGF. These research directions specialized for CL are left as future work to explore. Our framework is quite flexible and general due to the energy functional design and WGF specification.

D. Continual DRO With Wasserstein Ball Constraint

In this section, we present another variant of Continual-DRO by instantiating the distance $\mathcal{D}(\mu_0, \mu)$ in (5) to be Wasserstein distance, $W(\mu_0, \mu)$. Specifically, we formulate the continual DRO as the following optimization:

$$\min_{\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{\mu} [\mathbb{E}_\mu \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) - \gamma W(\mu_0, \mu) + \beta \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu, \boldsymbol{x}' \sim \mu_0} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y)], \tag{31}$$

where (31) is to optimize the memory loss on the worst-case memory data distribution. Different from Section IV-C, which uses f-divergence constraint, we use the Wasserstein ball constraint as an ambiguity set in (31) to ensure the evolved memory distribution μ not deviate from the raw memory distribution μ_0. f-divergence and Wasserstein-ball constraints have different modeling and computation properties for Continual DRO. We can efficiently solve the Continual DRO with f-divergence in function space by calculus of variation. However, using Wasserstein ball constraint can incorporate richer family of distributions in the neighbourhood of raw memory data distribution. On the other hand, it is computationally hard to solve the Wasserstein ball constraint [30] since (1) computing the Wasserstein distance itself is already computationally expensive; (2) we cannot calculate the functional gradient of Wasserstein-ball constraint in a closed form. We thus convert the above optimization problem into the following approximate optimization problem by using the surrogate loss function from [6]:

$$\min_{\boldsymbol{\theta}} \sup_{\boldsymbol{x} \in \mathcal{X}} (-\mathbb{E}_\mu U(\boldsymbol{x}, \boldsymbol{\theta}) - \gamma c(\boldsymbol{x}_0, \boldsymbol{x})), \tag{32}$$

where $\boldsymbol{x}_0$ is the raw memory data, $\boldsymbol{x}$ is the evolved memory data and $c(\boldsymbol{x}_0, \boldsymbol{x}) = ||\boldsymbol{x}_0 - \boldsymbol{x}||_2^2$. We thus can solve this optimization in euclidean space. The memory data evolves by gradient ascent of the inner sup optimization as the following equation:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = h\nabla_{\boldsymbol{x}}(-U(\boldsymbol{x}_t^i, \boldsymbol{\theta}) - \gamma c(\boldsymbol{x}_0, \boldsymbol{x})). \tag{33}$$

We name this method as *WD-continual*.

E. Algorithm Summary

The proposed memory evolution algorithm is shown in Algorithm 1, with the flexibility to use various evolution methods. Line 3-4 describes that a mini-batch data arrives at time k and samples a mini-batch data from the memory buffer. Line 5-7 is to evolve the mini-batch memory data with T steps depending on using which evolution methods. Line 8 updates the model

Algorithm 1: Distributionally Robust Memory Evolution.

1: **REQUIRE:** model parameters $\boldsymbol{\theta}$, learning rate η, evolution rate (step size) h, number of evolution steps T at each iteration, memory buffer $\mathcal{M}$; K is the number of mini-batch data in the data stream.
2: **for** $k = 1$ to K **do**
3: a new mini-batch data $(\boldsymbol{x}_k, y_k)$ arrives.
4: sample mini-batch from memory buffer, i.e., $(\boldsymbol{x}, y) \sim \mathcal{M}$
5: **for** $t = 1$ to T **do**
6: $(\boldsymbol{x}, y) = Evolve((\boldsymbol{x}, y))$ by any of the above mentioned f-divergence memory evolution methods, e.g., through WGF-LD (25) or WGF-SVGD (28) or WGF-HMC (30) or WD-continual (33).
7: **end for**
8: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla_{\boldsymbol{\theta}} [\mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x}, y) + \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x}_k, y_k)]$
9: update memory buffer by reservoir sampling (RS), $\mathcal{M} = \text{RS}(\mathcal{M}, (\boldsymbol{x}_k, y_k))$
10: **end for**

parameters with the evolved memory data and mini-batch data received at time k. Line 9 updates the memory buffer with the mini-batch data received at time k using reservoir sampling.

We do not replace the raw memory data with the evolved memory data in the current version. Replacing the raw memory data with the evolved one could decrease the performance in our experiment. Our method needs to store a mini-batch of evolved memory data. This memory cost is negligible compared to the entire data stream memory buffer. We can view memory evolution from two perspectives. First, *local evolution* evolves the current raw memory data distribution with several adaptation steps. Second, *global evolution* evolves the memory data at different CL timestamps due to different model parameters.

V. Experiments

To evaluate the effectiveness of our memory evolution methods, we compare to various state-of-the-art (SOTA) baseline approaches in the task-aware CL setting on several datasets in Section V-A and compare various methods in the task-free CL setting in Section V-B. We then evaluate the methods in terms of robustness to adversarial examples in Section V-C. We perform ablation study in Section V-D.

Datasets: We compare different methods on the following most commonly used datasets, including:

- *CIFAR10*, which has 10 image classes;
- *MiniImagenet* [54], which has 100 image classes;
- *CIFAR-100* [29], which also contains 100 image classes.

In the following experiments, we mainly focus on f-divergence specification with KL-divergence constraint, while we leave other f-divergence specifications in ablation study in Section V-D.

A. Task-Aware CL

We compare various methods on: (1) task-incremental learning (Task-IL), which provides task identities to the CL learner; and (2) class-incremental learning (Class-IL) [53], which does not provide task identities to the CL learner.

Baseline: We compare to various SOTA CL methods, described in the following:

- Regularization-based methods, including Classifier-Projection Regularization (CPR) [12], Gradient Projection Memory (GPM) [49], oEWC [50], synaptic intelligence (SI) [66], Learning without Forgetting (LwF) [33] and deep Streaming Linear Discriminant Analysis (SLDA) [23].
- Bayesian-based methods, including UCB [20].
- Architecture-based methods, including HAT [51].
- Memory-based CL methods, including ER [15], A-GEM [14], GSS [3], HAL [13], DER++ [9], GMED [26] and ER-ACE [10].

Most baseline implementations are based on [9]. We apply the proposed method on top of the implementation [9]. In addition, our proposed methods are orthogonal to existing memory-replay-based CL methods. Thus, they are versatile, and we can seamlessly combine the proposed methods with them. We name these methods as ER+WGF-LD, DER++WGF-LD, DER++WGF-SVGD, etc.

Evaluation Metrics: We use overall average accuracy and backward transfer at the end of CL training to evaluate the performance of the proposed methods and the baseline methods. We denote $a_{N,k}$ as the test accuracy on task k after learning on task N. Thus, the overall accuracy for all the tasks are $ACC = \frac{1}{N}\sum_{k=1}^{k=N} a_{N,k}$. To measure catastrophic forgetting, we also evaluate backward transfer (BWT). BWT is formally defined as: $BWT = \frac{1}{N-1}\sum_{k=1}^{k=N-1}(a_{N,k} - a_{k,k})$. $BWT < 0$ indicates that there is knowledge forgetting on previous tasks after learning new ones, and $BWT > 0$ indicates that learning new tasks is beneficial to improve the performance of previous tasks.

Implementation Details: Following [9], we use ResNet18 [24] as the CL learner (classifier) for all datasets. Following [9], we split the CIFAR-10 dataset into 5 disjoint tasks, where each task consists of 2 classes. We split MiniImagenet into 10 disjoint tasks, where each task has 10 classes. We also split CIFAR-100 into 10 disjoint tasks, where each task consists of 10 classes. We also follow [9] for other hyperparameter settings. To improve running efficiency, we randomly sample the number of evolution steps from the interval $[1, 5]$ at each CL step. The memory buffer can store 500 data points by default. We provide the performance of baselines and proposed methods with a larger memory size of 2000 in Appendix B.1, available online. We use the average accuracy and standard deviation across ten runs as the evaluation results. The compared methods and our proposed methods are based on the public implementation.[1]

Result: We compare the proposed methods to various CL baselines and combination with our proposed methods in Table I. Due to space limitations, we put the results on backward transfer in Appendix B.2, available online. We can observe that our method outperforms those baselines. In particular, for class-IL, our methods improves over ER by 2.3%, 2.5%, and 3.7%, on MiniImageNet, CIFAR-100 and CIFAR10, respectively. For task-IL, our methods improve over ER by 2.3%,

[1] [Online]. Available: https://github.com/aimagelab/mammoth

TABLE I
TASK-IL AND CLASS-IL RESULTS ON CIFAR10, CIFAR-100 AND MINIIMAGENET, RESPECTIVELY WITH MEMORY SIZE 500. '—' INDICATES NOT APPLICABLE

Algorithm Method	**CIFAR-10**		**CIFAR-100**		**MiniImagenet**	
	Class-IL	Task-IL	Class-IL	Task-IL	Class-IL	Task-IL
fine-tuning	19.62 ± 0.05	61.02 ± 3.33	9.29 ± 0.33	33.78 ± 0.42	8.59 ± 0.29	27.48 ± 0.38
Joint train	92.20 ± 0.15	98.31 ± 0.12	71.32 ± 0.21	91.31 ± 0.17	65.56 ± 0.18	87.74 ± 0.15
oEWC	19.49 ± 0.12	68.29 ± 3.92	8.24 ± 0.21	21.2 ± 2.08	7.32 ± 0.16	22.54 ± 1.78
SI	19.48 ± 0.17	68.05 ± 5.91	9.41 ± 0.24	31.08 ± 1.65	8.07 ± 0.26	30.16 ± 1.72
LwF	19.61 ± 0.05	63.29 ± 2.35	9.70 ± 0.23	28.07 ± 1.96	7.65 ± 1.31	21.49 ± 2.06
CPR	21.38 ± 0.19	72.37 ± 3.51	10.31 ± 0.24	31.93 ± 2.31	9.85 ± 0.23	31.87 ± 1.91
SLDA	25.70 ± 0.37	73.28 ± 3.78	12.19 ± 0.58	32.38 ± 2.72	10.97 ± 0.53	33.29 ± 1.77
GPM	—	90.68 ± 3.29	—	72.48 ± 0.40	—	60.41 ± 0.61
UCB	—	79.28 ± 1.87	—	57.15 ± 1.67	—	49.36 ± 1.09
HAT	—	92.56 ± 0.78	—	72.06 ± 0.50	—	59.78 ± 0.57
A-GEM	22.67 ± 0.57	89.48 ± 1.45	9.30 ± 0.32	48.06 ± 0.57	7.76 ± 0.12	39.28 ± 0.43
GSS	49.73 ± 4.78	91.02 ± 1.57	13.60 ± 2.98	57.50 ± 1.93	11.22 ± 3.17	50.19 ± 1.78
HAL	41.79 ± 4.46	84.54 ± 2.36	9.05 ± 2.76	42.94 ± 1.80	4.46 ± 1.72	31.97 ± 1.57
ER	57.74 ± 0.27	93.61 ± 0.27	20.98 ± 0.35	73.37 ± 0.43	11.76 ± 0.75	61.58 ± 0.31
ER+GMED	57.89 ± 0.38	93.45 ± 0.39	21.07 ± 0.43	73.42 ± 0.49	11.85 ± 0.81	61.76 ± 0.37
ER+WGF-LD	61.28 ± 0.43	94.72 ± 0.51	23.28 ± 0.56	75.53 ± 0.61	$\mathbf{14.09 \pm 0.76}$	63.67 ± 0.49
ER+WGF-SVGD	$\mathbf{61.42 \pm 0.51}$	94.61 ± 0.56	$\mathbf{23.51 \pm 0.65}$	75.65 ± 0.60	13.87 ± 0.76	63.83 ± 0.58
ER+WGF-HMC	61.32 ± 0.56	94.51 ± 0.64	23.32 ± 0.70	$\mathbf{75.72 \pm 0.66}$	13.93 ± 0.79	$\mathbf{63.91 \pm 0.65}$
ER+WD-continual	61.06 ± 0.61	$\mathbf{94.67 \pm 0.68}$	23.21 ± 0.75	75.37 ± 0.72	13.76 ± 0.81	63.02 ± 0.68
DER++	72.70 ± 1.36	93.88 ± 0.50	36.37 ± 0.85	75.64 ± 0.60	22.09 ± 0.63	61.26 ± 0.57
DER++GMED	72.82 ± 1.79	93.94 ± 0.70	36.25 ± 0.69	75.49 ± 0.64	22.21 ± 0.81	61.42 ± 0.64
DER++WGF-LD	74.03 ± 1.62	94.62 ± 0.66	37.51 ± 0.62	$\mathbf{77.93 \pm 0.73}$	23.82 ± 0.87	63.51 ± 0.77
DER++WGF-SVGD	74.17 ± 1.73	$\mathbf{94.95 \pm 0.71}$	37.89 ± 0.78	77.81 ± 0.76	$\mathbf{24.28 \pm 0.91}$	$\mathbf{63.67 \pm 0.83}$
DER++WGF-HMC	74.28 ± 1.79	94.69 ± 0.78	$\mathbf{37.96 \pm 0.82}$	77.86 ± 0.83	23.93 ± 0.95	63.49 ± 0.87
DER++WD-continual	$\mathbf{74.42 \pm 1.65}$	94.86 ± 0.81	37.83 ± 0.87	77.10 ± 0.86	23.64 ± 0.98	63.21 ± 0.91
ER-ACE + LiDER (w/o pre-train)	72.91 ± 1.25	—	37.56 ± 0.86	—	23.56 ± 0.72	—
+ WGF-LD	$\mathbf{74.64 \pm 1.22}$	—	38.42 ± 0.79	—	24.31 ± 0.75	—
+ WGF-SVGD	74.38 ± 1.09	—	$\mathbf{38.68 \pm 0.82}$	—	$\mathbf{24.71 \pm 0.79}$	—
+ WGF-HMC	74.33 ± 1.16	—	38.16 ± 0.80	—	24.17 ± 0.86	—
+ WD-continual	74.05 ± 1.25	—	37.79 ± 0.83	—	23.90 ± 0.67	—

2.4% and 1.1% on MiniImageNet, CIFAR-100, and CIFAR10, respectively. Furthermore, for class-IL, our methods enhance the performance of DER++ by 2.2%, 1.6% and 1.7% on MiniImageNet, CIFAR-100 and CIFAR10 respectively. For task-IL, our methods increase the performance of DER++ by 2.4%, 2.3%, and 1.0% on MiniImageNet, CIFAR-100 and CIFAR10 respectively. GMED brings little or even worse performance, consistent with the observations of [26]. We believe this is because, in task-aware CL, memory buffer data could be replayed many epochs, and GMED may edit the memory data too much so that they may significantly deviate from the original raw data. Our method outperforms baselines because the transformed memory buffer is more difficult for the CL model to overfit.

B. Task-Free CL

Baselines: We performed comprehensive experiments by comparing to the following strong baselines:

- *Experience Replay (ER)* [15], which stores a subset of examples from previous tasks with reservoir sampling [15]. We randomly replay a subset of examples from the memory buffer at each iteration.
- *Maximally Interfering Retrieval (MIR)* [1], the goal of MIR is to replay the examples in memory buffer that are easily forgettable for replay.
- *AGEM* [14], AGEM tries to ensure that at every training step the average episodic memory loss over the previous tasks does not increase.
- *Gradient-Based Sample Selection (GSS-Greedy)* [3] is to store diverse examples in the memory buffer. We use the efficient GSS-Greedy method.
- *GMED [26]:* GMED is the recent memory-replay method, which edits the memory data so that they are more easily forgettable.
- *LiDER [7]:* LiDER is a Lipschitz regularization method to mitigate forgetting by reducing the progressive deterioration of decision boundaries between different classes.
- *Data augmentation*, following [26], we also compare data augmentation, such as random rotations, scaling, and horizontal flipping applied to memory buffer data in ER and name this baseline as ER_{aug}.
- *Fine-tuning*, which trains on each latent task sequentially when new batches of each task arrive without any forgetting mitigation mechanism.
- *iid online*, which trains the model with a single-pass through the iid sampled data on the same set of samples.
- *iid offline*, (upper-bound) which trains the model with multiple passes through the iid sampled data. We train the model with 5 epochs for this baseline.

We combine our proposed methods with ER, MIR, and GMED to show the effectiveness. We name the combination

TABLE II
COMPARISON TO *TASK-FREE CL* BASELINES ON CIFAR10, CIFAR-100 AND MINIIMAGENET BY COMBING OUR PROPOSED METHOD WITH EXISTING CL METHODS

Algorithm	CIFAR10	CIFAR-100	MiniImagenet
fine-tuning	18.9 ± 0.1	3.1 ± 0.2	2.9 ± 0.5
A-GEM	19.0 ± 0.3	2.4 ± 0.2	3.0 ± 0.4
GSS-Greedy	29.9 ± 1.5	19.5 ± 1.3	17.4 ± 0.9
ER	33.3 ± 2.8	20.1 ± 1.2	24.8 ± 1.0
ER + WGF-LD	37.6 ± 1.5	$\mathbf{21.5 \pm 1.3}$	$\mathbf{28.0 \pm 1.0}$
ER + WGF-SVGD	$\mathbf{37.9 \pm 1.4}$	21.3 ± 1.5	27.8 ± 1.3
ER + WGF-HMC	37.8 ± 1.3	21.2 ± 1.4	27.6 ± 1.1
ER + WD-continual	36.1 ± 1.2	20.8 ± 1.1	27.1 ± 1.2
MIR	34.4 ± 2.5	20.0 ± 1.7	25.3 ± 1.7
MIR + WGF-LD	$\mathbf{38.2 \pm 1.2}$	$\mathbf{21.6 \pm 1.2}$	27.6 ± 1.0
MIR + WGF-SVGD	37.0 ± 1.4	21.2 ± 1.5	$\mathbf{27.9 \pm 1.2}$
MIR + WGF-HMC	37.9 ± 1.5	21.3 ± 1.4	27.5 ± 1.3
MIR + WD-continual	36.3 ± 1.3	20.9 ± 1.5	27.0 ± 1.5
GMED (ER)	34.8 ± 2.2	20.9 ± 1.6	27.3 ± 1.8
GMED + WGF-LD	$\mathbf{38.4 \pm 1.6}$	21.7 ± 1.7	28.3 ± 1.9
GMED + WGF-SVGD	37.6 ± 1.7	$\mathbf{21.8 \pm 1.5}$	$\mathbf{28.7 \pm 1.5}$
GMED + WGF-HMC	37.8 ± 1.2	21.5 ± 1.9	28.4 ± 1.3
GMED + WD-continual	36.5 ± 1.4	21.1 ± 1.7	27.6 ± 1.2
ER_{aug} + ER	46.3 ± 2.7	18.3 ± 1.9	30.8 ± 2.2
ER_{aug} + WGF-LD	48.2 ± 2.1	19.8 ± 2.2	31.9 ± 1.8
ER_{aug} + WGF-SVGD	48.3 ± 2.3	19.9 ± 2.3	$\mathbf{32.2 \pm 1.5}$
ER_{aug} + WGF-HMC	$\mathbf{48.5 \pm 2.2}$	20.3 ± 2.1	31.7 ± 2.0
ER_{aug} + WD-continual	47.6 ± 2.0	$\mathbf{20.6 \pm 1.9}$	31.3 ± 2.3
iid online	60.3 ± 1.4	18.7 ± 1.2	17.7 ± 1.5
iid offline	78.7 ± 1.1	44.9 ± 1.5	39.8 ± 1.4

methods ER+WGF-LD, ER+WGF-SVGD, ER+WGF-HMC, MIR+WGF-LD, GMED+WGF-LD, etc. Combining with other memory-replay-based methods is straightforward.

Implementation Details: For dataset split in task-free CL setting, we follow the splits in [1]. We split the CIFAR-10 dataset into 5 disjoint tasks with the same training, validation, and test sets. We split MiniImagenet [54] dataset into 20 disjoint tasks. Each task has 5 classes. We split CIFAR-100 dataset into 20 disjoint tasks, each with 5 classes. We use the Resnet-18 as [1]. The evolution rate h is set to be 0.01 for CIFAR10, 0.05 for CIFAR100 and 0.001 for MiniImagenet, $\beta = 0.003$ and momentum $\tau = 0.1$. We set the memory buffer size to be 500 for CIFAR-10 dataset, $5\,K$ for CIFAR-100 and $10\,K$ for MiniImagenet. All other hyperparameters are the same as [1]. All reported results in our experiments are the average accuracy across 10 runs with standard deviation.

Result: We compare the proposed methods to various task-free CL baselines. Table II shows the effectiveness of combining the proposed method with existing memory-replay methods, e.g., ER, MIR and GMED. We can observe that our method outperforms these strong baselines. In particular, for ER and ER + DRO methods, our method outperforms baselines by 4.6%, 3.2% and 1.4% on CIFAR10, MiniImageNet and CIFAR-100, respectively. For MIR and MIR + DRO methods, our method outperforms baselines by 3.8%, 2.6% and 1.6% on CIFAR10, MiniImageNet and CIFAR-100, respectively. For GMED and GMED + DRO methods, our method outperforms baselines by 3.6%, 1.4% and 0.9% on CIFAR10, MiniImageNet and CIFAR-100, respectively. Our methods outperform baselines because they dynamically evolve the memory data distribution and sufficiently explore the input space in a principled way. The proposed methods generate more diverse memory data, and the evolved memory becomes more difficult for the CL model to memorize than baseline methods. In addition, WGF-SVGD generally performs better than WGF-SGLD and WGF-HMC. We believe this is because, in RKHS, the evolved memory data is encouraged to be far apart by the kernel repulsive forces; the evolved memory data may better represent the distribution of all the previous data.

Analysis: Our finding is that in task-aware CL, WGF-SVGD overally (in majority) performs the best, while in task-free CL, WGF-LD and WGF-SVGD perform similarly (they achieve the best results in equal number of cases). Therefore, overally, WGF-SVGD performs the best. We believe this is because the repulsive term in (28) diversifies the evolved memory data to cover more modes in the data space.

The performance of our approaches exhibits variations across different datasets, primarily due to the varying resolutions and levels of learning difficulties present in each dataset. Specifically, when comparing CIFAR10, CIFAR100, and MiniImagenet, it becomes evident that CIFAR10 is the dataset with the least complexity. This is attributed to its smaller number of classes in comparison to the other two datasets. On the other hand, CIFAR100 and MiniImagenet pose more significant challenges as they consist of 100 classes with different image resolutions.

Furthermore, the memory evolution equations (WGF-LD with (25) and WGF-SVGD with (28)) are dataset dependent. For example, the terms $\nabla_{\boldsymbol{x}} U(\boldsymbol{x}, \boldsymbol{\theta})$, $k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)$ and $\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)$ are all influenced by the dataset being used. Each dataset has its own unique characteristics, such as the distinct feature space, image classes and resolutions present within it. Consequently, these terms exhibit differences when applied to different datasets. The dataset-dependent nature of these terms further emphasizes the impact that dataset choice can have on the properties and performance associated with them.

C. Robustness to Adversarial Perturbations

In this section, we evaluate the robustness of the CL model to adversarial perturbed examples in task-free CL setting. Given a classifier $f(\boldsymbol{x}, \boldsymbol{\theta})$, for an image $\boldsymbol{x}$, the goal is to find another example $\boldsymbol{x}'$ that is close enough to $\boldsymbol{x}$ measured by some distance function $D(\boldsymbol{x}.\boldsymbol{x}') \leq \epsilon$ such that the classifier classifies it into another different class, i.e., $f(\boldsymbol{x}, \boldsymbol{\theta}) \neq f(\boldsymbol{x}', \boldsymbol{\theta})$. In this paper, we focus on the most commonly used ℓ_∞ and ℓ_2 norm as the distance function.

Our memory evolution methods optimize on the worst-case evolved memory data distribution during CL; the model would be thus naturally robust to adversarial perturbations. For ℓ_∞ norm attack, we evaluate the robustness under the strong PGD ℓ_∞ attack [39], which constructs adversarial examples with projected gradient descent and ℓ_∞ norm constraints. The adversarial perturbation magnitude ranges from $[1/255, 2/255, \ldots, 10/255]$ with 20 steps attack and stepsize of $\frac{2}{255}$ on CIFAR-100 and Mini-ImageNet. For ℓ_2 norm attack, we adopt the strong Carlini & Wagner attack [11]. For illustration,

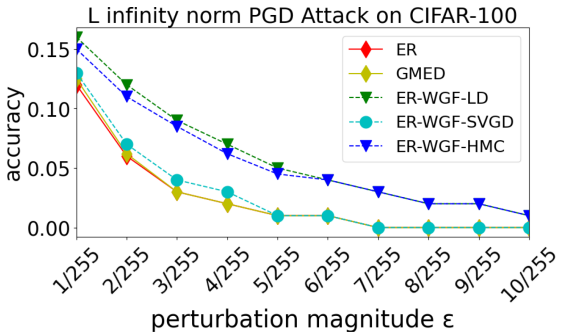


Fig. 8. PGD ℓ_∞ attack results on CIFAR-100.

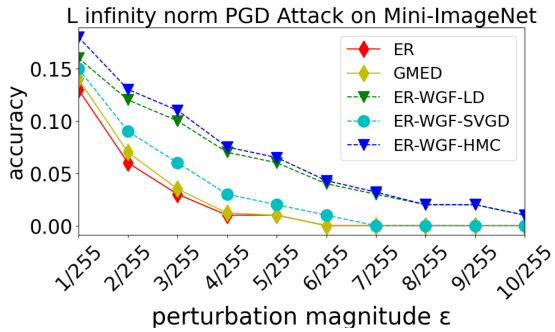


Fig. 9. PGD ℓ_∞ attack results on Mini-ImageNet.

TABLE III
CARLINI AND WAGNER ATTACK MODEL PERFORMANCE FOR THE CIFAR-100 AND MINI-IMAGENET DATASET

Algorithm	CIFAR-10	CIFAR-100	Mini-Imagenet
ER	2.0 ± 0.1	0.0	0.0
GMED	2.1 ± 0.1	0.0	0.0
ER + WGF-LD	8.0 ± 0.2	**3.0 ± 0.2**	**3.1 ± 0.1**
ER + WGF-SVGD	4.2 ± 0.1	0.0	2.2 ± 0.2
ER + WGF-HMC	**8.2 ± 0.3**	2.5 ± 0.2	3.0 ± 0.1

we evaluate the model robustness to adversarial examples after training with ER, GMED, ER + WGF-LD, ER + WGF-SVGD, and ER + WGF-HMC, respectively. Figs. 8 and 9 shows the PGD ℓ_∞ attack results on CIFAR-100 and Mini-ImageNet. Our WGF-HMC and WGF-LD memory evolution significantly outperforms naive ER baseline by 4%–12% depending on the perturbation magnitude. In addition, WGF-HMC and WGF-LD perform more robust than WGF-SVGD. We believe this is because the randomness introduced in WGF-HMC and WGF-LD can better explore the input space and thus can generate harder evolved memory data. WGF-SVGD smooths the function gradient by the kernel function, thus may generate less hard examples. Table III shows the Carlini & Wagner attack results. We can see that under the strong Carlini & Wagner attack, ER baseline accuracy becomes zero, and our methods still outperform baselines ranging from $6.1\%, 3.0\%, 3.1\%$ on CIFAR-10, CIFAR-100, Mini-ImageNet, respectively. Both results demonstrate the robustness of our proposed methods to adversarial examples.

TABLE IV
ABLATION STUDY ON THE EFFECT OF DIFFERENT f-DIVERGENCE ON CIFAR10, CIFAR-100, AND MINIIMAGENET IN TASK-FREE CL SCENARIO

Algorithm	**CIFAR10**	**CIFAR-100**	**MiniImagenet**
ER	33.3 ± 2.8	20.1 ± 1.2	24.8 ± 1.0
ER + WGF-KL	37.6 ± 1.5	21.5 ± 1.3	28.0 ± 1.0
ER + WGF-χ^2	35.2 ± 1.9	20.9 ± 1.1	25.9 ± 1.2
ER + WGF-$\alpha = 5$	**38.8 ± 1.6**	**21.8 ± 1.5**	**28.5 ± 1.1**

TABLE V
EFFECT OF DIFFERENT MEMORY SIZE ON THE MODEL PERFORMANCE FOR THE CIFAR-100 AND MINI-IMAGENET DATASETS

CIFAR-100			
Memory Size	2000	3000	5000
ER	11.2 ± 1.0	15.0 ± 0.9	20.1 ± 1.2
ER + WGF-LD	**12.9 ± 1.2**	17.0 ± 1.1	**21.5 ± 1.3**
ER + WGF-SVGD	12.3 ± 1.1	**17.8 ± 1.2**	21.3 ± 1.5
ER + WGF-HMC	12.7 ± 1.0	17.2 ± 1.0	21.2 ± 1.4
ER + WD-continual	12.2 ± 1.2	16.9 ± 1.1	20.9 ± 1.2
MIR	11.6 ± 0.8	15.6 ± 1.0	20.0 ± 1.7
MIR + WGF-LD	13.1 ± 0.9	17.3 ± 1.2	**21.6 ± 1.2**
MIR + WGF-SVGD	12.7 ± 1.0	17.2 ± 1.3	21.2 ± 1.5
MIR + WGF-HMC	**13.2 ± 1.2**	**17.5 ± 1.1**	21.3 ± 1.4
MIR + WD-continual	13.0 ± 1.2	17.1 ± 1.2	21.0 ± 1.5
Mini-Imagenet			
Memory Size	3000	5000	10000
ER	13.4 ± 1.4	17.9 ± 1.6	24.8 ± 0.9
ER + WGF-LD	**16.2 ± 1.2**	20.8 ± 1.2	28.0 ± 1.0
ER + WGF-SVGD	15.7 ± 1.2	**21.3 ± 1.0**	**27.8 ± 1.3**
ER + WGF-HMC	15.9 ± 1.5	20.6 ± 1.4	27.6 ± 1.1
ER + WD-continual	15.6 ± 1.3	20.1 ± 1.2	27.1 ± 1.2
MIR	12.6 ± 1.5	17.4 ± 1.2	25.3 ± 1.7
MIR + WGF-LD	15.5 ± 1.4	20.5 ± 1.1	27.6 ± 1.0
MIR + WGF-SVGD	15.3 ± 1.2	**20.7 ± 1.6**	**27.9 ± 1.2**
MIR + WGF-HMC	**15.8 ± 1.7**	20.3 ± 1.5	27.5 ± 1.3
MIR + WD-continual	15.1 ± 1.5	20.0 ± 1.7	27.1 ± 1.5

D. Ablation Study

Effect of Different f-Divergence: We evaluate the effect of different f-divergence instances, including KL-divergence, χ^2-divergence and α-divergence in Table IV. We observe that χ^2-divergence generally underperforms due to the absence of gradient information in the memory evolution equation regarding the target probability distribution. On the other hand, the more general α-divergence achieves better performance than KL and χ^2-divergence due to its strong mass-covering property. However, α-divergence does come with additional computation costs due to the involvement of more terms in the calculation process.

Effects of Different Memory Size: To investigate the effect of different smaller memory sizes on the model performance in task-free CL, we evaluate the effects on Mini-ImageNet with memory sizes of 3,000, 5,000, and 10,000. We evaluate the effects on CIFAR-100 with the memory sizes of 2,000, 3,000, and 5,000. We show the results in Table V. In most cases, our WGF memory evolution substantially outperforms the baselines with different memory buffer sizes.

Effect of Number of Evolution Steps: To investigate the effect of different evolution steps, we compare 3, 5, and 7 evolution

TABLE VI
EFFECT OF NUMBER OF EVOLUTION STEPS ON MINI-IMAGENET

Evolution Steps	3	5	7
ER + WGF-LD	27.0 ± 0.9	27.3 ± 1.0	27.5 ± 1.4
ER + WGF-SVGD	27.2 ± 1.2	27.6 ± 1.3	27.2 ± 1.2
ER + WGF-HMC	27.1 ± 1.3	27.2 ± 1.1	27.6 ± 1.0
ER + WD-continual	27.1 ± 1.2	26.7 ± 1.6	26.1 ± 0.8

TABLE VII
SENSITIVITY ANALYSIS OF EVOLUTION RATE h

CIFAR10	h	0.005	0.01	0.03
	Accuracy	37.8 ± 1.2	38.2 ± 1.5	38.1 ± 1.6
CIFAR100	h	0.01	0.05	0.1
	Accuracy	21.6 ± 1.2	21.5 ± 1.3	21.3 ± 1.2
Mini-ImageNet	h	0.0001	0.001	0.005
	Accuracy	28.0 ± 1.1	27.8 ± 1.0	27.7 ± 1.5

TABLE VIII
COMPUTATION EFFICIENCY (RELATIVE TRAINING TIME) OF THE PROPOSED METHOD COMPARED TO BASELINE

Algorithm	running time
ER	1.0
ER + GMED	1.5
ER + WGF-LD	1.9
ER + WGF-SVGD	2.6
ER + WGF-HMC	2.0
ER + WD-continual	1.9

steps, respectively. Table VI shows the performance variation of different number of evolution steps. We can find that the performance improves slightly with an increasing number of evolution steps; the performance becomes worse if increasing the number of evolution steps further. We believe this is because the memory buffer becomes too hard for the CL learner to learn well. For running efficiency and sufficiently exploring the input space to evolve harder memory examples, we choose a moderate number of evolution steps.

Effect of Evolution Rate: To evaluate the effect of evolution rate on the performance, we compare the effects of different evolution rate h on the CL model performance in Table VII. We can observe that with larger evolution rate of h, the performance becomes worse. We believe this is because the evolved memory becomes too hard for the CL learner so that it cannot learn effectively.

Hyperparameter Sensitivity: Due to space limitation, we put hyperparameter sensitivity analysis, including the regularizer weight β in Appendix B, available online.

Computation Cost: We compare the proposed ER + DRO to ER to evaluate its running efficiency. Table VIII shows the efficiency comparison results. We set the simple baseline ER with a running time unit of 1. Our method increases 0.9-1.6 times the computational cost compared to a simple ER baseline. The efficiency improvement is due to random sample the number of evolution steps at each CL step.

VI. CONCLUSION

This paper proposes a novel concept of DRO memory evolution for continual learning to dynamically evolve the memory data distribution to mitigate the memory overfitting issue and fill the gap between the memory data distribution and the distribution of all the previous data. We solve two types of constraints in the DRO memory evolution. We propose a family of memory evolution methods for *Continual DRO* with general f-divergence constraint and Wasserstein ball constraint. The proposed principled framework is general, flexible, and easily expandable. Future work includes designing more informative functional and novel gradient flow dynamics to incorporate physical intuitions and geometry constraints. Extensive experiments compared to various state-of-the-art methods demonstrate the effectiveness of the proposed method. Moreover, our methods are more robust to adversarial examples than compared baselines.

REFERENCES

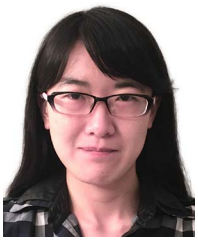
[1] R. Aljundi et al., "Online continual learning with maximal interfered retrieval," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11849–11860.
[2] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11246–11255.
[3] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, Art. no. 1058.
[4] L. Ambrosio, N. Gigli, and G. Savare, "Gradient flows: In metric spaces and in the space of probability measures," in *Lectures in Mathematics*, Zürich, Switzerland: ETH Zürich, 2008.
[5] R. F. Bass, *Stochastic Processes. Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[6] J. Blanchet and K. R. A. Murthy, "Quantifying distributional model risk via optimal transport," 2017. [Online]. Available: https://arxiv.org/abs/1604.01446
[7] L. Bonicelli, M. Boschini, A. Porrello, C. Spampinato, and S. Calderara, "On the effectiveness of Lipschitz-driven rehearsal in continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 31886–31901.
[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[9] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: A strong, simple baseline," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1335.
[10] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky, "New insights on reducing abrupt representation change in online continual learning," in *Proc. Int. Conf. Learn. Representations*, 2022.
[11] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.
[12] S. Cha, H. Hsu, T. Hwang, F. Calmon, and T. Moon, "CPR: Classifier-projection regularization for continual learning," in *Proc. Int. Conf. Learn. Representations*, 2021.
[13] A. Chaudhry et al., "Using hindsight to anchor past knowledge in continual learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 6993–7001.
[14] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *Proc. Int. Conf. Learn. Representations*, 2019.
[15] A. Chaudhry et al., "Continual learning with tiny episodic memories," 2019. [Online]. Available: https://arxiv.org/abs/1902.10486
[16] T. Chen, E. Fox, and C. Guestrin, "Stochastic gradient Hamiltonian Monte Carlo," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1683–1691.
[17] S. Chewi, T. Le Gouic, C. Lu, T. Maunu, and P. Rigollet, "SVGD as a kernelized wasserstein gradient flow of the chi-squared divergence," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 2098–2109.
[18] A. Chrysakis and M.-F. Moens, "Online continual learning from imbalanced data," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1952–1961.

[19] M. Delange et al., "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, Jul. 2022.
[20] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach, "Uncertainty-guided continual learning with Bayesian neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.
[21] C. Fernando et al., "PathNet: Evolution channels gradient descent in super neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1701.08734
[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.
[23] L. T. Hayes and C. Kanan, "Lifelong machine learning with deep streaming linear discriminant analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 220–221.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[25] X. He, J. Sygnowski, A. Galashov, A. A. Rusu, Y. W. Teh, and R. Pascanu, "Task agnostic continual learning via meta learning," 2019. [Online]. Available: https://arxiv.org/abs/1906.05201
[26] X. Jin, A. Sadhu, J. Du, and X. Ren, "Gradient-based editing of memory examples for online task-free continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29193–29205.
[27] R. Jordan, D. Kinderlehrer, and F. Otto, "The variational formulation of the Fokker–Planck equation," *SIAM J. Math. Anal.*, vol. 29, pp. 1–17, 1998.
[28] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, pp. 3521–3526, 2017.
[29] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
[30] D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh, "Wasserstein distributionally robust optimization: Theory and applications in machine learning," 2019, *arXiv:1908.08729*.
[31] S. Lee, J. Ha, D. Zhang, and G. Kim, "A neural Dirichlet process mixture model for task-free continual learning," in *Proc. 17th Int. Conf. Mach. Learn.*, 2020.
[32] W. Li and L. Ying, "Hessian transport gradient flows," *Res. Math. Sci.*, vol. 6, no. 4, p. 34, 2019.
[33] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[34] C. Liu, J. Zhuo, and J. Zhu, "Understanding MCMC dynamics as flows on the Wasserstein space," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4093–4103.
[35] Q. Liu, "Stein variational gradient descent as gradient flow," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3115–3123.
[36] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose Bayesian inference algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2370–2378.
[37] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6467–6476.
[38] Y.-A. Ma, T. Chen, and E. B. Fox, "A complete recipe for stochastic gradient MCMC," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2917–2925.
[39] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[40] E. Miersemann, "Calculus of variations," Lecture Notes, Leipzig University, 2012.
[41] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *Proc. Int. Conf. Learn. Representations*, 2018.
[42] Q. Pham, C. Liu, D. Sahoo, and S. HOI, "Contextual transformation networks for online continual learning," in *Proc. Int. Conf. Learn. Representations*, 2021.
[43] M. PourKeshavarzi, G. Zhao, and M. Sabokrou, "Looking back on learned experiences for class/task incremental learning," in *Proc. Int. Conf. Learn. Representations*, 2022.
[44] H. Rahimian and S. Mehrotra, "Distributionally robust optimization: A review," 2019, *arXiv:1908.05659*.
[45] D. Rao, F. Visin, A. A. Rusu, Y. W. Teh, R. Pascanu, and R. Hadsell, "Continual unsupervised representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7645–7655.
[46] M. Riemer et al., "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *Proc. Int. Conf. Learn. Representations*, 2019.
[47] A. A. Rusu et al., "Progressive neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1606.04671
[48] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," in *Proc. Int. Conf. Learn. Representations*, 2020.
[49] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," in *Proc. Int. Conf. Learn. Representations*, 2021.
[50] J. Schwarz et al., "Progress and compress: A scalable framework for continual learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4535–4544.
[51] J. Serrá, D. Surís, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4555–4564.
[52] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inform. Process. Syst.*, 2017, pp. 2990–2999.
[53] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019, *arXiv:1904.07734*.
[54] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3637–3645.
[55] J. von Oswald, C. Henning, J. Sacramento, and B. F. Grewe, "Continual learning with hypernetworks," 2019. [Online]. Available: https://arxiv.org/abs/1906.00695
[56] D. Wang, H. Liu, and Q. Liu, "Variational inference with tail-adaptive f-divergence," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5742–5752.
[57] L. Wang et al., "Memory replay with data compression for continual learning," in *Proc. Int. Conf. Learn. Representations*, 2022.
[58] Z. Wang, L. Liu, Y. Kong, J. Guo, and D. Tao, "Online continual learning with contrastive vision transformer," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 631–650.
[59] Z. Wang, T. Duan, L. Fang, Q. Suo, and M. Gao, "Meta learning on a sequence of imbalanced domains with difficulty awareness," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 8947–8957.
[60] Z. Wang, L. Shen, T. Duan, D. Zhan, L. Fang, and M. Gao, "Learning to learn and remember super long multi-domain task sequence," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 7982–7992.
[61] Z. Wang, L. Shen, L. Fang, Q. Suo, T. Duan, and M. Gao, "Improving task-free continual learning by distributionally robust memory evolution," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 22985–22998.
[62] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 681–688.
[63] A. Wibisono, "Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem," in *Proc. Conf. Learn. Theory*, PMLR, 2018, pp. 2093–3027.
[64] Z. Xu, C. Dan, J. Khim, and P. Ravikumar, "Class-weighted classification: Trade-offs and robust approaches," in *Proc. Int. Conf. Mach. Learn.*, 2020, Art. no. 977.
[65] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[66] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. 34th Int. Conf.Mach. Learn.*, 2017, pp. 3987–3995.
[67] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task agnostic continual learning using online variational bayes," 2019. [Online]. Available: https://arxiv.org/abs/1803.10123
[68] R. Zhai, C. Dan, J. Z. Kolter, and P. Ravikumar, "Doro: Distributional and outlier robust optimization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12345–12355.

Zhenyi Wang received the bachelor's degree from Northeastern University, China, and the PhD degree from The State University of New York at Buffalo. He is currently a postdoctoral associate with the University of Maryland, College Park. His research interests include meta learning, continual learning, and Bayesian learning.

Li Shen received the PhD degree from the School of Mathematics, South China University of Technology, in 2017. He is currently a research scientist with JD Explore Academy, China. Previously, he was a research scientist with Tencent AI Lab, China. His research interests include theory and algorithms for large scale convex/nonconvex/minimax optimization problems, and their applications in statistical machine learning, deep learning, reinforcement learning, and game theory.

Tiehang Duan received the BS degree in information engineering from Shanghai Jiao Tong University, and the PhD degree in computer science and engineering from University at Buffalo. He is currently a research scientist with Meta Inc. His research interest is mainly on machine learning of sequential data with applications in natural language processing.



Qiuling Suo is currently working toward the PhD degree in computer science with University at Buffalo. She is broadly interested in the areas of machine learning and data mining, including meta learning, metric learning, adversarial learning, and time series analysis. She also has experience on application areas, such as healthcare and traffic prediction.

Le Fang received the PhD degree from the Department of Computer Science and Engineering, SUNY at Buffalo, in 2020. He is currently a research scientist in Meta ads ranking core ML team. His research interests include text generation, advertising click-through rate (ctr) prediction, and meta learning.

Wei Liu (Fellow, IEEE) received the PhD degree in electrical engineering and computer science from Columbia University, in 2012. He is currently a distinguished scientist of Tencent and the director of Ads Multimedia AI with Tencent Data Platform. Prior to that, he has been a research staff member of IBM T. J. Watson Research Center from 2012 to 2015. He has long been devoted to fundamental research and technological development in core fields of AI, including deep learning, machine learning, reinforcement learning, computer vision, information retrieval, Big Data, etc. To date, he has published extensively in these fields with more than 280 peer-reviewed technical papers, and also issued more than 30 US patents. He currently serves on the editorial boards of internationally leading AI journals like *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, and *IEEE Intelligent Systems*. He is an area chair of top-tier computer science and AI conferences, e.g., NeurIPS, ICML, IEEE CVPR, IEEE ICCV, IJCAI, and AAAI. He is a fellow of the IAPR, and IMA, and an elected member of the ISI.

Mingchen Gao received the BEng degree from Southeast University, Nanjing, China, in 2007, and the PhD degree in computer science from Rutgers University, The State University of New Jersey, in 2014. From 2014 to 2017, she was a postdoctoral fellow with the Radiology and Imaging Science Department, National Institutes of Health. She is an assistant professor with the Department of Computer Science and Engineering, University at Buffalo, The State University of New York. Her research interest focuses on medical imaging informatics. She received the NSF CAREER award, in 2023.