Improving Task-free Continual Learning by Distributionally Robust Memory Evolution

Zhenyi Wang [1] **Li Shen** [2] **Le Fang** [1] **Qiuling Suo** [1] **Tiehang Duan** [3] **Mingchen Gao** [1]

Abstract

Task-free continual learning (CL) aims to learn a non-stationary data stream without explicit task definitions and not forget previous knowledge. The widely adopted memory replay approach could gradually become less effective for long data streams, as the model may memorize the stored examples and overfit the memory buffer. Second, existing methods overlook the high uncertainty in the memory data distribution since there is a big gap between the memory data distribution and the distribution of all the previous data examples. To address these problems, for the first time, we propose a principled memory evolution framework to dynamically evolve the memory data distribution by making the memory buffer gradually harder to be memorized with distributionally robust optimization (DRO). We then derive a family of methods to evolve the memory buffer data in the continuous probability measure space with Wasserstein gradient flow (WGF). The proposed DRO is w.r.t the worst-case evolved memory data distribution, thus guarantees the model performance and learns significantly more robust features than existing memory-replay-based methods. 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 existing task-free CL methods.

[1]Department of Computer Science and Engineering, University at Buffalo, NY, USA [2]JD Explore Academy, Beijing, China [3]Meta, Seattle, WA, USA. Correspondence to: Zhenyi Wang <zhenyiwa@buffalo.edu>, Li Shen <mathshenli@gmail.com>, Mingchen Gao <mgao8@buffalo.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

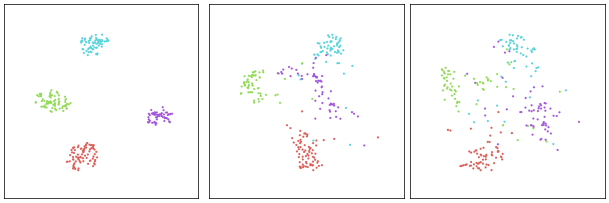
1. Introduction

Continual learning (CL) is to learn on a sequence of tasks without forgetting previous ones. Most CL methods assume knowing task identities and boundaries during training. Instead, this work focuses on a more general and challenging setup, i.e., task-free continual learning (Aljundi et al., 2019b). 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 approach stores a small portion of previous tasks in memory and replays them with the new mini-batch data. The CL model would overfit the memory buffer, and this approach could be gradually less effective for mitigating forgetting as the model repeatedly learns the memory buffer (Jin et al., 2021), illustrated in Figure 1 (a). The model could memorize the memory buffer, and previous knowledge will be quickly lost. Furthermore, there is a big gap between memory data distribution and the distribution of all the previous data examples. These methods ignore the high uncertainty of the memory data distribution since a limited memory buffer cannot accurately reflect the stationary distribution of all examples seen so far in the data stream, illustrated in Figure 2 (a). Most existing CL works ignore this issue.

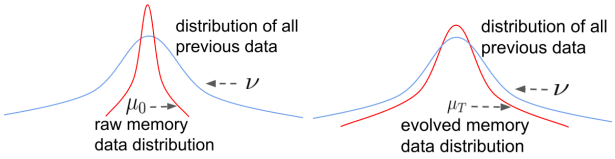
To address the above issues, we propose a distributionally robust optimization framework (DRO) to evolve the memory data distribution dynamically. This learning objective makes the memory data increasingly harder to be memorized and helps the model alleviate memory overfitting and improve generalization, illustrated in Figure 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 Figure 2 (b). 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. The model can thus learn substantially more robust features with performance guarantees than previous work. For the image domain, adversarial examples can have the same appearance as natural images

but have different classification results. 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.



(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)

Figure 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.



(a) Experience replay (raw memory data) (b) Our method (evolved memory data)

Figure 2. The blue line denotes the stationary distribution of all the previous data, and the red line denotes the memory buffer data distribution. (a): Standard experience replay (ER) (left) replays on the raw memory data. There is a big gap between the raw memory data distribution μ_0 and the stationary distribution of all the data in the data stream ν. (b): Our method (right) fills this gap by evolving the memory data distribution to narrow the gap between the evolved memory data distribution μ_T and ν.

Formally, we first design energy functional that measures the degree of forgetting on a specific memory data distribution and the distributional distance between the evolved and raw memory distribution. The goal is to minimize the energy functional $F(\mu)$ for the worse-case performance probability measure (density) π that is within the neighbor probability measures of the raw memory data probability measure μ_0. We name this optimization as *task-free DRO*. However, it is extremely challenging to solve this optimization problem in an infinite-dimensional probability measure space (function space) which contains an infinite number of probability measures. To efficiently solve the task-free DRO, we formulate it from a new continuous dynamics perspective and convert it into a gradient flow system. Specifically, the memory data distribution evolves in probability measure space as Wasserstein gradient flow (WGF), and model parameters evolve in Euclidean space. We name it as *Dynamic DRO*, which makes it convenient to use *function gradient-descent* in probability measure space to solve the task-free DRO. Also, it facilitates the derivation of different methods for memory data distribution evolution. We then introduce three specific memory evolution methods to solve the Dynamic DRO, including Langevin Dynamics (Welling & Teh, 2011), Stein Variational Gradient Descent (Liu & Wang, 2016), and Hamiltonian flow (Ma et al., 2015; Liu et al., 2019). The proposed memory evolution framework is general, flexible, and easily extendable, with many potential extensions for future research. We evaluate the effectiveness of the proposed framework by performing comprehensive experiments on several datasets and comparing them to various strong baselines. We summarize our contributions as follows:

- We propose the first principled, general, and flexible **memory evolution** framework for task-free CL from the perspective of distributionally robust optimization, named *task-free 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 works.
- We formulate the task-free 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 demonstrate the effectiveness of the proposed method for reducing 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.

2. Related Work

Continual learning (CL) aims to maintain previous knowledge when learning on sequential tasks with data distribution shift. Most existing CL methods (Lopez-Paz & Ranzato, 2017; Nguyen et al., 2018; Kirkpatrick et al., 2017; Zenke et al., 2017; von Oswald et al., 2019; Wang et al., 2021; Saha et al., 2021; Pham et al., 2021; Wang et al., 2022) are only applicable to the task-aware setting, where there are clear task definitions and boundaries among the sequentially learned task sequences. Task-free continual learning (He et al., 2019; Zeno et al., 2019; Aljundi et al., 2019b; Chrysakis & Moens, 2020) instead focuses on the more general case where the data distribution could change

arbitrarily without explicit task splits. This learning scenario has become increasingly important due to the broader application scenarios, and more challenging problem nature (Aljundi et al., 2019b; Lee et al., 2020).

Task-free CL Existing approaches for task-free CL can be categorized into two classes. The first (majority) one is memory-based methods (Aljundi et al., 2019c;a), which store a small number of data from the previous data stream and replay them with the new mini-batch data later. The second type is the expansion-based method, such as CN-DPM (Lee et al., 2020). 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 (Chaudhry et al., 2019b;a) directly perform replay on the raw data without any adaptation. MIR (Aljundi et al., 2019a) proposes to replay the samples with which are most interfered. GEN-MIR (Aljundi et al., 2019a) further uses generative models to synthesize the memory examples. The *heuristic* method, GMED (Jin et al., 2021), proposes to edit memory examples so that the examples are more likely forgotten and harder to be memorized. 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) (Aljundi et al., 2019c) focuses on *storing diverse examples*. These methods lack theoretical guarantees. In contrast, our framework is principled and focuses on evolving memory data distributions.

Distributionally robust optimization (DRO) is an effective optimization framework to handle decision-making under uncertainty (Rahimian & Mehrotra, 2019). 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 (Sagawa et al., 2020), subpopulation shift (Zhai et al., 2021), and class imbalances (Xu et al., 2020).

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

3. Method

In this section, we first present the problem setup in Section 3.1 for the task-free CL setting. Then, we propose our task-free DRO framework in Section 3.2. Next, we view the task-free DRO from a new continuous dynamics perspective and formulate the task-DRO in an equivalent gradient flow system, named Dynamic DRO, in Section 3.3. In Section 3.4, we present three specific memory evolution methods to solve the Dynamic DRO efficiently.

3.1. Problem Setup

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 $\boldsymbol{x}_k$. According to (Aljundi et al., 2019b), 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. During both the training and testing time, the task identity h_k is not available to the learner. 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 (Riemer et al., 2019). The goal is to learn a model $f(\boldsymbol{x}, \boldsymbol{\theta})$ to perform well on all the tasks seen until timestamp k. Standard memory replay for CL (Chaudhry et al., 2019b) is to optimize an objective under a known probability distribution μ_0. Formally speaking, CL with standard 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., $\forall \boldsymbol{x} \in \mathcal{M}, \boldsymbol{x} \sim \mu_0$. $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y)$ 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.

3.2. DRO for Task-free CL

Distributionally Robust Optimization (DRO) (Rahimian & Mehrotra, 2019) is a systematic and elegant framework to model the decision-making with ambiguity in the underlying probability distribution. For task-free CL, different from the standard memory-replay methods in Section 3.1, implicitly assuming 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 Figure 2 (a). 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 Figure 2 (b). 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 standard memory-replay repeatedly trains the memory buffer, and the CL model can easily overfit the memory buffer, as illustrated in Figure 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 Figure 1 (b) and (c). Mathematically speaking, the proposed DRO for task-free CL can be expressed as:

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

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

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

where the inner $\sup$ optimization is to gradually make the memory data distribution increasingly harder to be memorized. $\mathcal{P}$ in Eq. (3) denotes the ambiguity set of probability measures (distributions or densities) for the memory data distribution to characterize its uncertainty. One common choice to define $\mathcal{P}$ is through Kullback-Leibler (KL) divergence. $\mathcal{D}(\mu_0||\pi)$ denotes the KL divergence between probability measure μ_0 and π, where π is the target worst-case evolved memory data distribution, i.e., the probability distribution π that achieves $\sup_{\mu \in \mathcal{P}} \mathbb{E}_{\mu} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y)$. ϵ 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. Eq. (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.

To solve the optimization, i.e., Eq. (2-4), the worst-case optimization that involves the $\sup$ optimization within the KL-divergence ball is generally computationally intractable since it involves the optimization over infinitely many probability distributions. To address this problem, by Lagrange duality (Boyd & Vandenberghe, 2004), we convert Eq. (2-4) into the following unconstrained optimization problem (detailed derivations are put in Appendix B):

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

Where γ and β control the magnitude of regularization. Since the KL-divergence term $\mathcal{D}(\mu||\pi)$ is implicitly handled by gradient flow (discussed in the following sections), we set $\gamma = 1$ for simplicity throughout this paper. The gradient of the third term (gradient dot product term) can be efficiently approximated by finite difference in practice. The optimization Eq. (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 this optimization Eq. (5) as **task-free DRO**.

3.3. Task-free DRO: A Continuous Dynamics View

To make it tractable to solve the task-free DRO Eq. (5), we formulate it from a new continuous dynamics perspective. This new perspective brings significant advantages over directly solving Eq. (5): 1) the optimization of Eq. (5) 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 task-free DRO Eq. (5) into a gradient flow system. Specifically, we solve the inner $\sup$ optimization problem for memory evolution with WGF in continuous Wasserstein space of probability measures and solve the outer optimization problem for the model parameters with gradient flow in Euclidean space. We thus convert the task-free 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 $\mathcal{M} = \{(\boldsymbol{x}_0^1, y^1), (\boldsymbol{x}_0^2, y^2), \cdots, (\boldsymbol{x}_0^N, y^N)\}$, 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 i^{th} 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., $\{\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \cdots, \boldsymbol{x}_0^N\} \sim X_0$. 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., $\{\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \cdots, \boldsymbol{x}_t^N\} \sim X_t$ 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.

Below, we define and present the gradient flow 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 3.1 (Wasserstein Gradient Flow)**.** (Ambrosio et al., 2008). Suppose we have a Wasserstein space $\mathbb{W}^2 = (\mathcal{P}_2(\mathbb{R}^d), W_2)$. A curve of $(\mu_t)_{t\geq 0}$ of probability measures is a Wasserstein gradient flow for functional F if it satisfies

$$\partial_t \mu_t = \nabla_{W_2} F(\mu_t) := div\left(\mu_t \nabla \frac{\delta F}{\delta \mu}(\mu_t)\right) \tag{6}$$

where $:=$ means defined as, and $div(\boldsymbol{r}) := \sum_{i=1}^d \partial_{\boldsymbol{z}^i} \boldsymbol{r}^i(\boldsymbol{z})$ 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. $\nabla_{W_2} F(\mu_t) := div(\mu_t \nabla \frac{\delta F}{\delta \mu}(\mu_t))$ 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. The first variation of a *functional* in function space is analogous to the first-order gradient of a *function* in Euclidean distance. We put more detailed definition in Appendix D. 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 towards the target probability measure.

We define the energy functional $F(\mu)$ for memory evolution as the following:

$$F(\mu) = V(\mu) + \mathcal{D}(\mu||\pi) \tag{7}$$

$$V(\mu) = -\mathbb{E}_\mu \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) - \beta \mathbb{E}_\mu \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y). \tag{8}$$

By defining such energy functional $F(\mu)$, the task-free DRO in Eq. (5) can be equivalently solved by the following gradient flow system Eq. (9, 10), we name it as **Dynamic DRO**.

$$\begin{cases} \partial_t \mu_t &= div\left(\mu_t \nabla \frac{\delta F}{\delta \mu}(\mu_t)\right); & (9) \\ \frac{d\boldsymbol{\theta}}{dt} &= -\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mu_t} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y), & (10) \end{cases}$$

where Eq. (9) solves the inner sup problem in Eq. (5) with WGF in Wasserstein space and Eq. (10) solves the outer minimization problem in Eq. (5) for parameter update with gradient flow in Euclidean space. In the following, we focus on solving Eq. (9) and Eq. (10).

3.4. Training Algorithm for Dynamic DRO

We propose three different methods for efficiently solving the Dynamic DRO in Eq. (9)-Eq. (10). 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 Eq. (9) 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.

Before introducing the proposed solutions, we first present some preliminaries for solving the WGF. By calculus of variation (Miersemann, 2012), the first variation for the KL divergence and $V(\mu)$ are (Ambrosio et al., 2008):

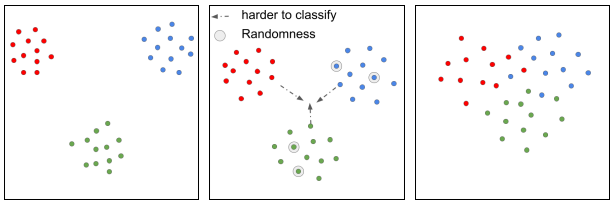
$$\frac{\delta \mathcal{D}(\mu||\pi)}{\delta \mu} = \log \frac{\mu}{\pi} + 1;$$

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

Langevin Dynamics for Dynamic DRO. If we directly use the energy functional $F(\mu)$ (Eq. (7)) in Eq. (9), solving gradient flow Eq. (9) corresponds to the Langevin dynamics with the following stochastic differential equation (Welling & Teh, 2011):

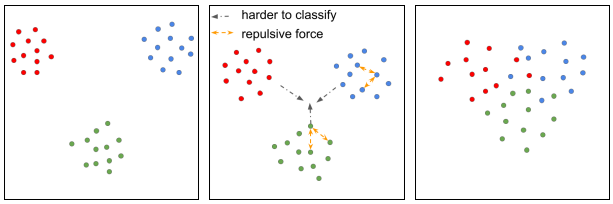
$$dX = -\nabla_X U(X, \boldsymbol{\theta}) dt + \sqrt{2} dW_t, \tag{11}$$

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



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

Figure 3. 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.



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

Figure 4. 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 (corresponds to the first term ($k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j) \nabla_{\boldsymbol{x}_t^j} U(\boldsymbol{x}_t^j, \boldsymbol{\theta})$)) drives the memory data to become harder to classify. The orange arrow (corresponds to the second term ($\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)$)) serves as repulsive force such that the memory data becomes more diverse.

motion in $\mathbb{R}^n$ (Bass, 2011). If $X_t \sim \mu_t$ evolves according to the Langevin dynamics Eq. (11) in Euclidean space, then $\mu(\boldsymbol{x}, t) = \mu_t(\boldsymbol{x})$ evolves according to gradient flow Eq. (9) in the space of probability measures (Jordan et al., 1998). 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:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = -\alpha(\nabla_{\boldsymbol{x}} U(\boldsymbol{x}_t^i, \boldsymbol{\theta})) + \sqrt{2\alpha}\xi_t. \tag{12}$$

As illustrated in Figure 3, the first term in Eq. (12) drives the particles (memory data) towards the worst-case memory data distribution π to make the memory data gradually harder to be memorized 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 α 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 transformation $\mathcal{K}_\mu \nabla_{W_2} F(\mu_t)$ under the integral operator $\mathcal{K}_\mu f(\boldsymbol{x}) = \int K(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') d\mu(\boldsymbol{x}')$; 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 (Liu, 2017):

$$\partial_t \mu_t = div(\mu_t \mathcal{K}_{\mu_t} \nabla \frac{\delta F}{\delta \mu}(\mu_t)). \tag{13}$$

Eq. (13) can be viewed as the WGF Eq. (9) 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 (Liu, 2017):

$$\frac{dX}{dt} = -[\mathcal{K}_\mu \nabla \frac{\delta F}{\delta \mu}(\mu_t)](X). \tag{14}$$

This kernelized version is the deterministic approximation of the WGF in Eq. (9) (Liu & Wang, 2016). If we discretize the above equation and view each datapoint in memory as one particle, we can obtain the following memory evolution update equation:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = -\frac{\alpha}{N} \sum_{j=1}^{j=N} \underbrace{[k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j) \nabla_{\boldsymbol{x}_t^j} U(\boldsymbol{x}_t^j, \boldsymbol{\theta})}_{\text{smoothed gradient}} + \underbrace{\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)]}_{\text{repulsive term}}, \tag{15}$$

As illustrated in Figure 4, 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 $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^2}{2\sigma^2})$. We name this memory evolution method as **WGF-SVGD**. We put detailed derivations of this evolution equation in Appendix E.

General Memory Evolution for Dynamic DRO. (Ma et al., 2015) 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:

$$\partial_t \mu_t = div(\mu_t (\boldsymbol{D} + \boldsymbol{Q}) \nabla \frac{\delta F}{\delta \mu}(\mu_t)), \tag{16}$$

Where $\boldsymbol{D}$ is a positive semidefinite diffusion matrix, $\boldsymbol{Q}$ is a skew-symmetric curl matrix representing the deterministic traversing effect (Ma et al., 2015). 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 (Chen et al., 2014) and can be solved by the following evolution:

$$\begin{cases} \boldsymbol{x}_{t+1} - \boldsymbol{x}_t = \boldsymbol{v}_t, \\ \boldsymbol{v}_{t+1} - \boldsymbol{v}_t = -\alpha \nabla U(\boldsymbol{x}, \boldsymbol{\theta}) - \tau \boldsymbol{v} + \sqrt{2\tau\alpha}\xi_t, \end{cases} \tag{17}$$

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.

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 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. Note that we do not replace the raw memory data with the evolved memory data for implementation simplicity in the current version. If we replace the raw memory data with the evolved one, we can reduce the number of evolution steps needed at each CL timestamp to improve efficiency since an example can be adequately evolved after accumulating many timestamps. Earlier examples can be evolved less frequently than the later ones. But replacing the raw memory data with the evolved one could slightly reduce 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.

Discrete input. For the application of our methods in the discrete input domain (such as text), text can be embedded in continuous space (e.g., word embedding) and then evolve on embedding with our methods.

Algorithm 1 Distributionally Robust Memory Evolution.

1: **REQUIRE:** model parameters $\boldsymbol{\theta}$, learning rate η, evolution rate (step size) α, 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 WGF-LD (Eq. (12)) or WGF-SVGD (Eq. (15))) or WGF-HMC (Eq. (17)).
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, $\mathcal{M} =$ reservoirsampling$(\mathcal{M}, (\boldsymbol{x}_k, y_k))$
10: **end for**

4. Experiments

To evaluate the effectiveness of our memory evolution methods, we compared a variety of state-of-the-art baseline approaches. We first describe the benchmark datasets and baselines in Section 4.1. Then, we compare various baselines on several datasets in Section 4.2, and evaluate the methods in terms of robustness to adversarial examples in Section 4.3. We perform ablation study in Section 4.4.

4.1. Experiment Setup

CIFAR10, following (Aljundi et al., 2019a), we split the CIFAR-10 dataset into 5 disjoint tasks with the same training, validation, and test sets.

MiniImagenet, following (Aljundi et al., 2019a), we split MiniImagenet (Vinyals et al., 2016) dataset with 100 image classes into 20 disjoint tasks. Each task has 5 classes.

CIFAR-100 (Krizhevsky, 2009), contains 100 image classes. We also split it into 20 disjoint tasks.

Baselines. We performed comprehensive experiments by comparing to the following strong baselines, including *Experience Replay (ER) (Chaudhry et al., 2019b)*, *Maximally Interfering Retrieval (MIR) (Aljundi et al., 2019a)*, *AGEM (Chaudhry et al., 2019a)*, Gradient-Based Sample Selection (GSS-Greedy) (Aljundi et al., 2019c) and GMED (Jin et al., 2021). Furthermore, following (Jin et al., 2021), 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$_{aug}$. Following (Aljundi et al., 2019a), we also compare (1) fine-tuning, which trains on each latent task sequentially when new batches of each task arrive without any forgetting mitigation mechanism; (2) iid online: which trains the model with a single-pass through the iid sampled data on the same set of samples; (3) 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. Detailed descriptions of the compared baselines are placed in Appendix A.

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. For illustration, we combine our proposed methods with ER, MIR, and GMED to show the effectiveness. We name the combination 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. We use the Resnet-18 as (Aljundi et al., 2019a). The number of evolution steps T is set to be 5, the evolution rate α 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, $5K$ for CIFAR-100 and $10K$ for MiniImagenet. All other hyperparameters are the same as (Aljundi et al., 2019a). All reported results in our experiments are the average accuracy of 10 runs with standard deviation.

4.2. Comparison to Continual Learning

We compare the proposed methods to various task-free CL baselines. Table 1 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 + WGF methods, our method outperforms baselines by 4.5%, 1.4%, and 2.8% on CIFAR10, CIFAR-100, and Mini-ImageNet, respectively. For MIR and MIR + WGF methods, our method outperforms baselines by 3.8%, 1.6%, and 2.1% on CIFAR10, CIFAR-100, and MiniImageNet, respectively. For GMED and GMED + WGF methods, our method outperforms baselines by 3.6%, 0.9%, and 1.4% on CIFAR10, CIFAR-100, and MiniImageNet, 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.

4.3. Robustness to Adversarial Perturbations

In this section, we evaluate the robustness of the CL model to adversarial perturbed examples. 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.

Table 1. Comparison to continual learning 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	$\mathbf{21.5 \pm 1.3}$	27.3 ± 1.0
ER + WGF-SVGD	36.5 ± 1.4	21.3 ± 1.5	$\mathbf{27.6 \pm 1.3}$
ER + WGF-HMC	$\mathbf{37.8 \pm 1.3}$	21.2 ± 1.4	27.2 ± 1.1
MIR	34.4 ± 2.5	20.0 ± 1.7	25.3 ± 1.7
MIR + WGF-LD	$\mathbf{38.2 \pm 1.2}$	$\mathbf{21.6 \pm 1.2}$	26.9 ± 1.0
MIR + WGF-SVGD	37.0 ± 1.4	21.2 ± 1.5	$\mathbf{27.4 \pm 1.2}$
MIR + WGF-HMC	37.9 ± 1.5	21.3 ± 1.4	27.1 ± 1.3
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
ER_{aug} + ER	46.3 ± 2.7	18.3 ± 1.9	30.8 ± 2.2
ER_{aug} + WGF-LD	47.6 ± 2.4	19.8 ± 2.2	31.9 ± 1.8
ER_{aug} + WGF-SVGD	$\mathbf{47.9 \pm 2.5}$	19.9 ± 2.3	$\mathbf{32.2 \pm 1.5}$
ER_{aug} + WGF-HMC	47.8 ± 2.6	$\mathbf{20.3 \pm 2.1}$	31.7 ± 2.0
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

Table 2. Carlini & 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	$\mathbf{3.0 \pm 0.2}$	$\mathbf{3.1 \pm 0.1}$
ER + WGF-SVGD	4.2 ± 0.1	0.0	2.2 ± 0.2
ER + WGF-HMC	$\mathbf{8.2 \pm 0.3}$	2.5 ± 0.2	3.0 ± 0.1

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 (Madry et al., 2018), which constructs adversarial examples with projected gradient descent and ℓ_∞ norm constraints. The adversarial perturbation magnitude ranges from $[1/255, 2/255, \cdots, 10/255]$ 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 (Carlini & Wagner, 2017). For illustration, we evaluate the model robustness to adversarial examples after training with ER, GMED, ER + WGF-LD, ER + WGF-SVGD, and ER + WGF-HMC, respectively. Figure 5 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 random-

ness 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 2 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.

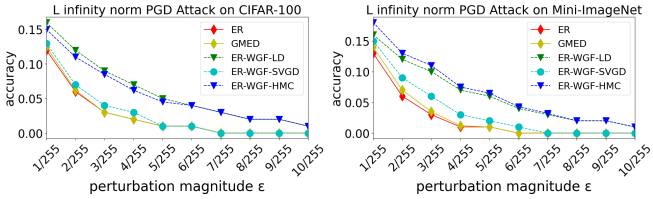


Figure 5. PGD ℓ_∞ attack results on two datasets: CIFAR-100 (left) and Mini-ImageNet (right).

4.4. Ablation Study

Effects of different memory size. To investigate the effect of different smaller memory sizes on the model performance, we evaluate the effects on Mini-ImageNet with memory sizes of 3000, 5000, and 10000. We evaluate the effects on CIFAR-100 with the memory sizes of 2000, 3000, and 5000. We show the results in Table 3. In most cases, our WGF memory evolution substantially outperforms the baselines with different memory buffer sizes.

Table 3. 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	$\mathbf{12.9 \pm 1.2}$	17.0 ± 1.1	$\mathbf{21.5 \pm 1.3}$
ER + WGF-SVGD	12.3 ± 1.1	16.0 ± 1.2	21.3 ± 1.5
ER + WGF-HMC	12.7 ± 1.0	$\mathbf{17.2 \pm 1.0}$	21.2 ± 1.4
MIR	11.6 ± 0.8	15.6 ± 1.0	20.0 ± 1.7
MIR + WGF-LD	13.1 ± 0.9	17.3 ± 1.2	$\mathbf{21.6 \pm 1.2}$
MIR + WGF-SVGD	12.7 ± 1.0	16.5 ± 1.3	21.2 ± 1.5
MIR + WGF-HMC	$\mathbf{13.2 \pm 1.2}$	$\mathbf{17.5 \pm 1.1}$	21.3 ± 1.4
Mini-Imagenet			
Memory Size	3000	5000	10000
ER	13.4 ± 1.4	17.9 ± 1.6	24.8 ± 0.9
ER + WGF-LD	$\mathbf{16.2 \pm 1.2}$	20.8 ± 1.2	27.3 ± 1.0
ER + WGF-SVGD	15.7 ± 1.2	$\mathbf{21.3 \pm 1.0}$	$\mathbf{27.6 \pm 1.3}$
ER + WGF-HMC	15.9 ± 1.5	20.6 ± 1.4	27.2 ± 1.1
MIR	12.6 ± 1.5	17.4 ± 1.2	25.3 ± 1.7
MIR + WGF-LD	15.5 ± 1.4	20.5 ± 1.1	26.9 ± 1.0
MIR + WGF-SVGD	15.3 ± 1.2	$\mathbf{20.7 \pm 1.6}$	$\mathbf{27.4 \pm 1.2}$
MIR + WGF-HMC	$\mathbf{15.8 \pm 1.7}$	20.3 ± 1.5	27.1 ± 1.3

Effect of number of evolution steps. To investigate the effect of different evolution steps, we compare 3, 5, and 7 evolution steps, respectively. Table 4 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. For efficiency and sufficiently exploring the input space to evolve harder memory examples, we choose the evolution step of 5.

Table 4. 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

Hyperparameter sensitivity. Due to space limitation, we put hyperparameter sensitivity analysis, including the regularizer weight β and evolution rate α, in Appendix C.

Computation cost. We compare the proposed ER + WGF to ER to evaluate its running efficiency. Table 5 shows the efficiency comparison results. We set the simple baseline ER with a running time unit of 1. Our method has 3-4 times the computational cost compared to a simple ER baseline. In future work, we will improve its computational efficiency.

Table 5. Computation efficiency (relative training time) of the proposed method compared to baseline.

Algorithm	running time
ER	1.0
ER + WGF-LD	3.4
ER + WGF-SVGD	4.1
ER + WGF-HMC	3.5

5. Conclusion

This paper proposes a novel concept of DRO memory evolution for task-free 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 propose a family of memory evolution methods with WGF. 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. Interestingly, our methods are more robust to adversarial examples than compared baselines.

6. Acknowledgement

We thank all the anonymous reviewers for their insightful and thoughtful comments. This research was supported in part by NSF through grant IIS-1910492.

References

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. Online continual learning with maximal interfered retrieval. *Advances in Neural Information Processing Systems 32*, pp. 11849–11860, 2019a.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems 30*, 2019c.

Ambrosio, L., Gigli, N., and Savare, G. Gradient flows: In metric spaces and in the space of probability measures. *(Lectures in Mathematics. ETH)*, 2008.

Bass, R. F. *Stochastic Processes*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2011. doi: 10.1017/CBO9780511997044.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. *Proceedings of the International Conference on Learning Representations*, 2019a.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. Continual learning with tiny episodic memories. *https://arxiv.org/abs/1902.10486*, 2019b.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Chewi, S., Le Gouic, T., Lu, C., Maunu, T., and Rigollet, P. Svgd as a kernelized wasserstein gradient flow of the chi-squared divergence. In *Advances in Neural Information Processing Systems*, 2020.

Chrysakis, A. and Moens, M.-F. Online continual learning from imbalanced data. *Proceedings of the 37th International Conference on Machine Learning*, 119:1952–1961, 2020.

He, X., Sygnowski, J., Galashov, A., Rusu, A. A., Teh, Y. W., and Pascanu, R. Task agnostic continual learning via meta learning. *https://arxiv.org/abs/1906.05201*, 2019.

Jin, X., Sadhu, A., Du, J., and Ren, X. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 2021.

Jordan, R., Kinderlehrer, D., , and Otto., F. The variational formulation of the fokker–planck equation. *SIAM Journal on Mathematical Analysis*, 1998.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Lee, S., Ha, J., Zhang, D., and Kim, G. A neural dirichlet process mixture model for task-free continual learning. In *Proceedings of the 17th International Conference on Machine Learning*, 2020.

Liu, C., Zhuo, J., and Zhu, J. Understanding mcmc dynamics as flows on the wasserstein spac. *Proceedings of the International Conference on Machine Learning*, 2019.

Liu, Q. Stein variational gradient descent as gradient flow. *Advances in Neural Information Processing Systems*, 2017.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in Neural Information Processing Systems*, 2016.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 2017.

Ma, Y.-A., Chen, T., and Fox, E. B. A complete recipe for stochastic gradient mcmc. *Advances in Neural Information Processing Systems*, 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *Proceedings of the International Conference on Learning Representations*, 2018.

Miersemann, E. *Calculus of Variations, Lecture Notes*. Leipzig University, 2012.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. *Proceedings of the International Conference on Learning Representations*, 2018.

Pham, Q., Liu, C., Sahoo, D., and HOI, S. Contextual transformation networks for online continual learning. *Proceedings of the International Conference on Learning Representations*, 2021.

Rahimian, H. and Mehrotra, S. Distributionally robust optimization: A review. 2019.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. *International Conference on Learning Representations*, 2019.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *International Conference on Learning Representations*, 2020.

Saha, G., Garg, I., and Roy, K. Gradient projection memory for continual learning. *Proceedings of the International Conference on Learning Representations*, 2021.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. Matching networks for one shot learning. 29, 2016.

von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. Continual learning with hypernetworks. *https://arxiv.org/abs/1906.00695*, 2019.

Wang, Z., Duan, T., Fang, L., Suo, Q., and Gao, M. Meta learning on a sequence of imbalanced domains with difficulty awareness. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Wang, Z., Shen, L., Duan, T., Zhan, D., Fang, L., and Gao, M. Learning to learn and remember super long multi-domain task sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. *Proceedings of the International Conference on Machine Learning*, 2011.

Xu, Z., Dan, C., Khim, J., and Ravikumar, P. Class-weighted classification: Trade-offs and robust approaches. *Proceedings of the International Conference on Machine Learning*, 2020.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. *https://arxiv.org/abs/1703.04200*, 2017.

Zeno, C., Golan, I., Hoffer, E., and Soudry, D. Task agnostic continual learning using online variational bayes. *https://arxiv.org/abs/1803.10123*, 2019.

Zhai, R., Dan, C., Kolter, J. Z., and Ravikumar, P. Doro: Distributional and outlier robust optimization. *Proceedings of the International Conference on Machine Learning*, 2021.

A. Baselines

The detailed descriptions of baselines are the following:

Experience Replay (ER) (Chaudhry et al., 2019b), which stores a small set of examples from previous tasks with reservoir sampling (Chaudhry et al., 2019b). At later time, we randomly sample a set of examples from the memory buffer to train with the received mini-batch data together to avoid forgetting.

Maximally Interfering Retrieval (MIR) (Aljundi et al., 2019a), the goal of MIR is to select the examples that are easily forgettable for replay. The idea of MIR is not using randomly selected data from the memory buffer, but instead replaying the samples that would be (maximally) interfered by the new received data. Following (Aljundi et al., 2019a), we evaluate the model forgetting on 25 examples for Mini-ImageNet dataset, and 50 examples for other datasets.

AGEM (Chaudhry et al., 2019a), AGEM tries to ensure that at every training step the average episodic memory loss over the previous tasks does not increase. They project the gradient update direction such that they are less interfered with current data by projecting the gradient to the closest in L2 norm gradient that keeps the angle within 90 degree.

Gradient-Based Sample Selection (GSS-Greedy) (Aljundi et al., 2019c) is to encourage diverse examples in the memory buffer. We use GSS-Greedy, which is efficient and performs the best.

GMED (Jin et al., 2021). GMED is the recent memory-replay methods, which edits the memory data so that they are harder to be memorized, shares the similar hypothesis as (Aljundi et al., 2019a).

B. Lagrangian Duality Derivation

The basic idea in Lagrangian duality (Boyd & Vandenberghe, 2004) is to consider the constraints by augmenting the objective function with a weighted sum of the constraint functions. We first convert the optimization Eq. (2), Eq. (3) and Eq. (4) into the following optimization.

$$\min_{\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{\mu \in \mathcal{P}} \mathbb{E}_{\mu} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \tag{18}$$

$$\text{s.t. } \mathcal{P} = \{\mu : \mathcal{D}(\mu || \pi) \leq \mathcal{D}(\mu_0 || \pi) \leq \epsilon\} \tag{19}$$

$$- \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu, \boldsymbol{x}' \sim \mu_0} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y) \leq -\sigma. \tag{20}$$

Then, by Lagrangian duality (Boyd & Vandenberghe, 2004), we can get the equivalent constrained optimization:

$$\begin{aligned} \min_{\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{\mu} [\mathbb{E}_{\mu} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) - \gamma \mathcal{D}(\mu || \pi) + \\ \beta \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu, \boldsymbol{x}' \sim \mu_0} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}', y)]. \end{aligned} \tag{21}$$

where γ and β are the Lagrange multiplier associated with the corresponding inequality constraints.

C. More Experiments Results

Hyperparameter Sensitivity Analysis

We use ER+WGF-LD as ablation study, similar results for other memory evolution methods. We perform sensitivity analysis for the regularization weight β of gradient dot product between perturbed memory data and raw data. The results are shown in Table 6. In particular, we show the results that $\beta = 0.0$, i.e., without gradient dot product regularization. Also, we perform sensitivity analysis on memory evolution rate α in Table 7, 8 and 9.

Table 6. Sensitivity analysis of regularization weight β.

Hyperparameter	CIFAR-10	CIFAR-100	Mini-Imagenet
$\beta = 0.0$	37.2 ± 1.7	21.2 ± 1.5	27.1 ± 1.4
$\beta = 0.001$	37.3 ± 1.8	21.6 ± 1.2	27.5 ± 1.8
$\beta = 0.003$	37.6 ± 1.5	21.5 ± 1.3	27.3 ± 1.0
$\beta = 0.005$	37.5 ± 1.1	21.3 ± 1.1	27.6 ± 1.4

Table 7. Sensitivity analysis of evolution rate α on CIFAR10.

$\alpha = 0.005$	$\alpha = 0.01$	$\alpha = 0.03$
37.5 ± 1.2	37.6 ± 1.5	37.9 ± 1.6

Table 8. Sensitivity analysis of evolution rate α on CIFAR100.

$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$
21.6 ± 1.2	21.5 ± 1.3	21.3 ± 1.2

Table 9. Sensitivity analysis of evolution rate α on miniImageNet.

$\alpha = 0.0001$	$\alpha = 0.001$	$\alpha = 0.005$
27.5 ± 1.2	27.3 ± 1.0	26.9 ± 1.5

D. Foundations of Calculus of Variations

In calculus of variations, the first variation is defined as following:

Definition D.1 (First Variations)**.** The first variation of a functional $F(\mu)$ is the functional at μ

$$\frac{\delta F}{\delta \mu}(\mu) = \lim_{\epsilon \to 0} \frac{F(\mu + \epsilon\psi) - F(\mu)}{\epsilon}, \tag{22}$$

where ψ is arbitrary function.

For more detailed background knowledge of calculus of variations, we recommend the reader refer to (Miersemann, 2012).

E. Derivations of the memory evolution methods

Specifically, we derive the WGF-SVGD as an example:

$$\frac{\delta \mathcal{D}(\mu||\pi)}{\delta \mu} = \log \frac{\mu}{\pi} + 1;$$
$$\frac{\delta V(u)}{\delta \mu} = U(\boldsymbol{x},\boldsymbol{\theta}) = -\mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y) - \beta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta},\boldsymbol{x},y) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta},\boldsymbol{x}',y).$$

We define the target worst-case evolved memory data distribution as $\pi \propto e^{-U}$ by the energy function $U(\boldsymbol{x},\boldsymbol{\theta})$ defined above.

We replace the Wasserstein gradient $\nabla_{W_2} F(\mu_t)$ by the transformation $\mathcal{K}_\mu \nabla_{W_2} F(\mu_t)$ under the integral operator $\mathcal{K}_\mu f(\boldsymbol{x}) = \int K(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') d\mu(\boldsymbol{x}')$;

The probability measure in the kernelized Wasserstein space actually follows the kernelized WGF (Liu, 2017):

$$\partial_t \mu_t = div(\mu_t \mathcal{K}_{\mu_t} \nabla \frac{\delta F}{\delta \mu}(\mu_t)). \tag{23}$$

Eq. (23) can be viewed as the WGF Eq. (9) 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 (Liu, 2017; Chewi et al., 2020):

$$\frac{dX}{dt} = -[\mathcal{K}_\mu \nabla \frac{\delta F}{\delta \mu}(\mu_t)](X). \tag{24}$$

First, we derive the kernelized Wasserstein gradient of

$$\mathcal{K}_{\mu_t} \nabla \log \frac{\mu_t}{\pi}(\boldsymbol{x}) := \int K(\boldsymbol{x}, \cdot) \nabla \log \frac{\mu_t}{\pi} = \int K(\boldsymbol{x}, \cdot) \nabla U d\mu_t - \int \nabla_2 K(\boldsymbol{x}, \cdot) d\mu_t \tag{25}$$

where, we apply integration by parts in the second identity. ∇_2 means the gradient of the kernel w.r.t. the second argument.

It indicates that the random variable X_t which describes the evolved memory data at time t evolves as the following differential equation (Liu, 2017):

We plug Eq. 25 into Eq. 24, we can obtain the following equations.

$$\frac{dX}{dt} = -\int K(\boldsymbol{x}, \cdot) \nabla U d\mu_t + \int \nabla_2 K(\boldsymbol{x}, \cdot) d\mu_t \tag{26}$$

If we discretize the above Eq. 26 and view each datapoint in memory as one particle, we can obtain the following memory evolution update equation:

$$\boldsymbol{x}_{t+1}^i - \boldsymbol{x}_t^i = -\frac{\alpha}{N} \sum_{j=1}^{j=N} [\underbrace{k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j) \nabla_{\boldsymbol{x}_t^j} U(\boldsymbol{x}_t^j, \boldsymbol{\theta})}_{\text{smoothed gradient}} + \underbrace{\nabla_{\boldsymbol{x}_t^j} k(\boldsymbol{x}_t^i, \boldsymbol{x}_t^j)}_{\text{repulsive term}}], \tag{27}$$