Gradient-Guided Epsilon Constraint Method for Online Continual Learning

Song Lai[*1,2], **Changyi Ma**[*2], **Fei Zhu**[2], **Zhe Zhao**[1]
Xi Lin[1], **Gaofeng Meng**[†2,3,4] **Qingfu Zhang**[†1]
[1]Department of Computer Science, City University of Hong Kong
[2]Centre for Artificial Intelligence and Robotics, HKISI, CAS
[3]Institute of Automation, Chinese Academy of Sciences
[4]School of Artificial Intelligence, University of Chinese Academy of Sciences
gfmeng@nlpr.ia.ac.cn, qingfu.zhang@cityu.edu.hk

Abstract

Online Continual Learning (OCL) requires models to learn sequentially from data streams with limited memory. Rehearsal-based methods, particularly Experience Replay (ER), are commonly used in OCL scenarios. This paper revisits ER through the lens of ϵ-constraint optimization, revealing that ER implicitly employs a soft constraint on past task performance, with its weighting parameter post-hoc defining a slack variable. While effective, ER's implicit and fixed slack strategy has limitations: it can inadvertently lead to updates that negatively impact generalization, and its fixed trade-off between plasticity and stability may not optimally balance current streaming with memory retention, potentially overfitting to the memory buffer. To address these shortcomings, we propose the **G**radient-Guided **E**psilon **C**onstraint (**GEC**) method for online continual learning. GEC explicitly formulates the OCL update as an ϵ-constraint optimization problem, which minimize the loss on the current task data and transform the stability objective as constraints and propose a gradient-guided method to dynamically adjusts the update direction based on whether the performance on memory samples violates a predefined slack tolerance $\bar{\varepsilon}$: if forgetting exceeds this tolerance, GEC prioritizes constraint satisfaction; otherwise, it focuses on the current task while controlling the rate of increase in memory loss. Empirical evaluations on standard OCL benchmarks demonstrate GEC's ability to achieve a superior trade-off, leading to improved overall performance. Code is available at https://github.com/laisong-22004009/GEC_OCL.

1 Introduction

Continual Learning (CL) aims to develop systems that can learn sequentially from a stream of tasks, a critical capability for real-world applications where data evolves [De Lange et al., 2021]. Online Continual Learning (OCL) presents a particularly challenging scenario where data arrives in small batches, is typically processed in a single pass, and memory for past experiences is strictly limited [Mai et al., 2022, Lopez-Paz and Ranzato, 2017]. The central challenge in OCL is catastrophic forgetting [McCloskey and Cohen, 1989, French, 1999], where learning new tasks causes a severe degradation of performance on previously learned ones. This has been a focal point of recent research [De Lange et al., 2021, Mai et al., 2022].

[*]Equal Contribution
[†]Corresponding Authors

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Among the various strategies proposed to mitigate forgetting, rehearsal-based methods, which store and replay a small subset of past data,have proven to be particularly effective in the OCL[Rebuffi et al., 2017, Rolnick et al., 2019]. Experience Replay (ER), in its basic form, combines the loss on current data with a weighted loss on replayed memory samples. The potential for overfitting to the limited memory buffer has motivated constraint-based replay methods like Gradient Episodic Memory (GEM) [Lopez-Paz and Ranzato, 2017] and Averaged GEM (A-GEM) [Chaudhry et al., 2018]. These methods use memory samples primarily to constrain the gradient direction of the current task, aiming to prevent an increase in loss on past tasks. However, there is substantial empirical evidence [Zhang et al., 2022] suggesting that rehearsal-based methods like ER, which train directly on memory samples, consistently outperform methods that only use memory for gradient constraints.

In this paper, we investigate the underlying reasons for ER's strong empirical performance by revisiting it from the perspective of ϵ-constraint optimization [Miettinen, 1999] (Section 3). We posit that ER implicitly solves an ϵ-constraint optimization problem, where the constraint on past task performance is soft and the weighting parameter λ for memory samples implicitly defines a slackness. In contrast, methods like GEM and A-GEM enforce hard constraints (near-zero slack). While ER's implicit soft constraints contribute to its success, the fixed nature of λ and the post-hoc, implicit definition of this slack present challenges. Considering the optimization-generalization gap in OCL [Ye and Bors, 2022], it becomes crucial to understand how slack strategies affect the model's test performance.

Our empirical analysis (Section 4) further explores this relationship. We find that ER's approach, which allows for some beneficial conflicts with memory task gradients, can lead to better generalization than strictly adhering to memory constraints. Specifically, mild violations of memory constraints do not always lead to poorer test performance. However, ER is not without its flaws. Firstly, its implicit, post-hoc constraint mechanism means the optimization process can still venture into regions that yield negative generalization gains. Secondly, even when memory constraints are satisfied, a fixed λ can suboptimally manage the plasticity-stability trade-off, potentially leading to overfitting on the memory buffer by excessively minimizing memory loss.

Motivated by these insights, we propose the **G**radient-Guided **E**psilon **C**onstraint (**GEC**) method (Section 5). GEC explicitly adopts a dynamic gradient-based ϵ-constraint optimization approach. At each learning step, GEC formulates the update as a quadratic program. The core idea is to: (1) When the constraint on past task forgetting (defined by a permissible slack $\bar{\varepsilon}$) is violated, GEC prioritizes satisfying this constraint. (2) When the constraint is satisfied, GEC focuses on optimizing the current task and controls the rate at which the memory constraint can be relaxed (i.e., memory loss can increase), preventing overfitting to the memory buffer and preserving generalization capacity. The adjustment mechanism allows GEC to effectively navigate the trade-offs inherent in OCL.

The main contributions of this paper are:

- We revisit Experience Replay from an ϵ-constraint optimization viewpoint, highlighting its implicit soft constraint nature and the role of its weighting parameter as a post-hoc slack definition. We also provide empirical evidence suggesting why ER often outperforms hard constraint-based methods, pointing to the benefits of allowing constraint violations and leveraging positive transfer from memory.
- By identifying key limitations of ER's fixed and implicit constraint strategy, we introduce the Gradient-Guided Epsilon Constraint (GEC) method, a novel OCL algorithm that explicitly enforces and dynamically adjusts forgetting constraints using gradient information, addressing key limitations of ER's fixed and implicit strategy.
- We provide theoretical justification for GEC, analyzing its update dynamics and convergence tendencies within the constrained optimization framework. Comprehensive experiments on standard OCL benchmarks demonstrate that GEC achieves a superior balance in online continual learning, outperforming existing state-of-the-art methods.

2 Preliminary

The ϵ-constraint method [Miettinen, 1999] is a widely used technique for finding Pareto optimal solutions in multi-objective optimization (MOO) [Chankong and Haimes, 2008]. The core idea of

the ϵ-constraint method is to convert a MOO problem into a series of single-objective problems. This is achieved by selecting one of the objective functions to be minimized (or maximized) while transforming the other objective functions into inequality constraints. Specifically, for a problem with K objective functions $f_1(x), f_2(x), \ldots, f_K(x)$ to be minimized, the ϵ-constraint formulation for optimizing $f_s(x)$ (the s-th objective) is:

$$\begin{aligned} \min_{x} \quad & f_s(x) \\ \text{s.t.} \quad & f_j(x) \leq \epsilon_j, \quad \forall j \in \{1, \ldots, K\}, j \neq s \end{aligned} \tag{1}$$

where x is the vector of decision variables and ϵ_j are predefined upper bounds (tolerances) for the constrained objectives $f_j(x)$. By systematically varying the values of ϵ_j, one can generate different pareto optimal solutions [Gunantara, 2018].

In CL, where a model must learn new information while preserving previously acquired knowledge, the objectives of minimizing current task loss and minimizing forgetting of past tasks are often in conflict. The ϵ-constraint framework provides a natural way to formulate and manage this trade-off, as will be explored in subsequent sections. In this paper, objective function of plasticity is selected to be optimized while stability is converted into constraints by setting an upper bound.

3 Revisiting ER from ϵ-Constraint Perspective

3.1 Problem Setup

We consider an OCL setting where a model $f(X;\theta)$ learns from a sequence of tasks $\{(X_1, Y_1), \ldots, (X_T, Y_T)\}$. $\ell_t(f(X_t;\theta), Y_t)$ is the loss for task t. At step k (processing a mini-batch from the current task t), the model has access to a limited memory buffer M containing samples $\{(X_m, Y_m)\}_{m<t}$ from past tasks. The goal is to learn the current task while mitigating forgetting of past tasks. We denote the parameters before the current update as θ_{k-1} and after the update as θ_k.

3.2 Continual Learning as ϵ-Constraint Optimization

As suggested in [Aljundi, 2019], CL can be viewed as attempts to solve a ϵ-constrained optimization problem. The primary objective is to minimize the loss on the current task data, $\ell_c(\theta)$, subject to constraints on the performance degradation of previous tasks, typically measured on samples from the memory buffer M. This can be formulated as:

$$\min_{\theta} \ell_c(f(X_c;\theta), Y_c) \quad \text{s.t. } \Delta\ell_m(\theta) \leq \varepsilon_m(k), \quad \forall m \in M_s \tag{2}$$

where M_s could represent individual past tasks or an aggregate. $\Delta\ell_m(\theta) = \ell_m(f(X_m;\theta), Y_m) - \ell_m(f(X_m;\theta_{k-1}), Y_m)$ is the change in loss for a past task m (or memory sample / average memory loss) relative to its loss before the current update step k. $\varepsilon_m(k)$ is the permissible slack for past task m at step k.

Methods like GEM [Lopez-Paz and Ranzato, 2017] and A-GEM [Chaudhry et al., 2018] explicitly enforce such constraints, typically aiming for $\varepsilon_m(k) \approx 0$ (non-increasing past task loss). For A-GEM, this is often a single constraint on the average memory loss:

$$\min_{\theta} \ell_c(f(X_c;\theta), Y_c) \quad \text{s.t. } \bar{\ell}_M(f(X_M;\theta), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M) \leq 0 \tag{3}$$

where $\bar{\ell}_M$ is the average loss over samples in memory M. Compared with *hard* constraint of GEM and A-GEM. ER optimizes a composite loss:

$$\min_{\theta} \left(\ell_c(f(X_c;\theta), Y_c) + \lambda\bar{\ell}_M(f(X_M;\theta), Y_M)\right) \tag{4}$$

where $\lambda > 0$. Next we will show that ER also implicitly solves an ϵ-constraint problem with a *soft*, data-dependent slack, where λ plays a crucial role in defining this slack post-hoc.

Proposition 1 (ER as ϵ-Constraint Optimization)**.** *Let θ^* be a solution to the Experience Replay optimization problem in Eq.* (4) *using parameters θ_{k-1} as the starting point for the optimization step. Then, θ^* is also a solution to the following ϵ-constraint optimization problem:*

$$\min_{\theta} \ell_c(f(X_c;\theta), Y_c) \quad \textit{s.t. } \bar{\ell}_M(f(X_M;\theta), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M) \leq \varepsilon^*_M(k) \tag{5}$$

where the slack can be chosen as $\varepsilon^*_M(k) = \bar{\ell}_M(f(X_M;\theta^*), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M)$. The magnitude of this effective slack $\varepsilon^*_M(k)$ is influenced by λ.

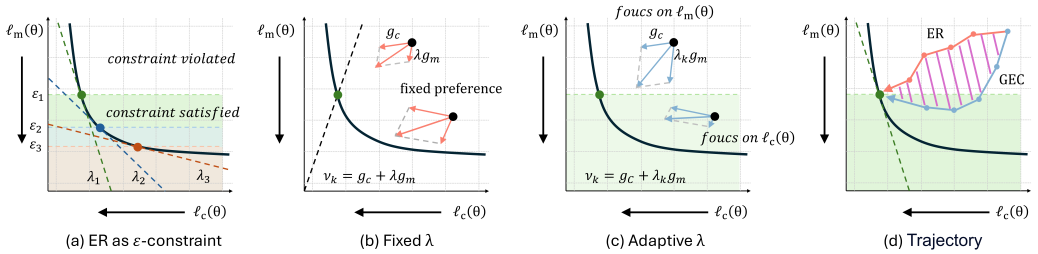

Figure 1: **(a)**: Each choice of ER's linear weighting parameter λ for memory samples corresponds to an implicit slack tolerance ε for the stability constraint. The dashed lines in (a) represent the iso-objective contours for ER's scalarized loss, where the optimal solution is found at the tangent point to the Pareto front; **(b)**: Illustration of a trajectory taken by a fixed weighting strategy like ER. Such strategies potentially leads to suboptimal paths in the objective space; **(c)**: Depiction of dynamic update adjustment by the proposed GEC method. **(d)**: Comparison of moving trajectories in the objective space between ER and GEC.

Proposition 2 (ϵ-Constraint Optimization as ER under Convexity)**.** *Suppose the assumptions stated in Appendix hold, where $\ell_c(\theta)$ and $\bar{\ell}_M(\theta)$ are convex and differentiable. Let θ^* be a solution to:*

$$\min_{\theta} \ell_c(f(X_c;\theta), Y_c) \quad \text{s.t. } \bar{\ell}_M(f(X_M;\theta), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M) \leq \varepsilon_M(k) \tag{6}$$

There exists a weight $\lambda^* \geq 0$ such that θ^* also minimizes $\ell_c(\theta) + \lambda^*(\bar{\ell}_M(\theta) - \bar{\ell}_M(\theta_{k-1}))$. Since $\bar{\ell}_M(\theta_{k-1})$ is constant for the current optimization step, this is equivalent to minimizing $\ell_c(\theta) + \lambda^* \bar{\ell}_M(\theta)$ plus a constant.

Detailed proofs for Propositions 1 and 2 are provided in Appendix A.1. To help understanding, Fig. 1(a) is an intuitive demonstration of the above propositions, which illustrates that ER's weighting parameter λ implicitly defines a constraint slackness $\varepsilon_M^*(k)$. This means ER operates with a flexible, soft constraint. This insight motivates a deeper investigation into how different constraint strategies (different λ values in ER, or the hard constraint of GEM) influence the model's generalization performance.

4 Empirical Evidence for Beneficial Conflicts

To understand why ER often outperforms hard constraint methods like GEM, we analyze the relationship between gradient updates, memory buffer interactions, and generalization. In OCL, the update direction g is derived based on the current task and memory samples. The interaction with memory can be quantified.

Definition 1 (Gradient Cosine Similarity (GCS))**.** *Let g be the gradient vector used for the model update at step k (i.e., $\theta_k = \theta_{k-1} - \eta g$). Let $g_M = \nabla_\theta \bar{\ell}_M(\theta_{k-1})$ be the gradient of the average loss on the memory buffer M, evaluated at θ_{k-1}. The Gradient Cosine Similarity defined as:*

$$GCS = \cos(g, g_M) = \frac{g \cdot g_M}{\|g\| \|g_M\|}. \tag{7}$$

A smaller (more negative) GCS indicates that the update g is more misaligned with the direction that would decrease memory loss. The change in memory loss is approximately $-\eta\langle g, g_M\rangle$ under locally linear assumption. Thus, a smaller GCS (more negative $\langle g, g_M\rangle$) implies a larger increase in memory loss (constraint violation).

Definition 2 (Generalization Gain (GG))**.** *Let θ be the model parameters before an update, and g be the gradient vector for the update. Let $P(\phi)$ be a performance measure (e.g., accuracy) on a held-out test set comprising samples from all encountered tasks. η is the learning rate. The Generalization Gain is:*

$$GG = P(\theta - \eta g) - P(\theta). \tag{8}$$

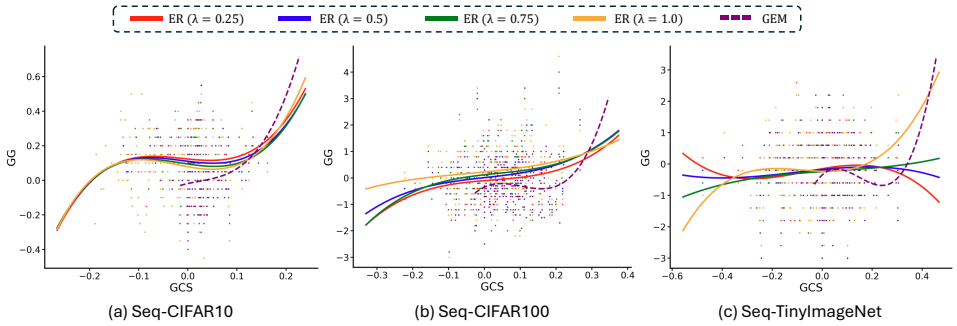

Figure 2: Relationship between GCS and GG for ER and GEM on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet.

By analyzing the relationship between GCS (for different λ in ER, and for GEM) and GG in Fig 2, we observe the following:

Mild gradient conflicts can be tolerable for generalization.

When the model's update direction g moderately conflicts with the memory gradient g_M (i.e., GCS is negative, indicating an increase in memory loss), this does not necessarily lead to a decrease in test set performance. ER, by allowing such conflicts via its soft constraint, can find updates that are beneficial for generalization even if they slightly worsen the loss on the memory buffer. This suggests that strictly preventing any increase in memory loss (as in GEM) might be too restrictive and discard potentially useful updates for generalization. This helps explain ER's empirical advantages over GEM-like methods. However, ER's approach is not without drawbacks. We also observe that:

Implicit post-hoc constraints risk negative generalization.
Fixed λ limits increasing rate of generalization gain when gradients align.

Firstly, ER's constraint is implicit and determined post-hoc by λ. During optimization, the model can still move into regions that yield significant negative generalization gains (left areas). Secondly, when the memory constraint is already satisfied (i.e., memory loss is low or decreasing), a fixed λ in ER continues to allocate a portion of the update to minimizing memory loss. This can lead to overfitting on the memory buffer, as the actual update direction might be driven to further reduce memory loss beyond what is necessary for preserving past knowledge. This over-correction can potentially harm overall generalization by not allowing the model to fully exploit plasticity. And inappropriate λ leads to a strong negative transfer (See Fig 2(c)).

These limitations motivate a method that can: (1) more explicitly manage the constraint on past task forgetting to avoid severe negative generalization, and (2) dynamically adjust its focus between current task learning and memory constraint satisfaction, especially by preventing overfitting to the memory buffer when constraints are met. This dynamic adjustment, illustrated in Fig. 1(b) and (c), ensures that when the forgetting constraint is violated, the update prioritizes constraint satisfaction, and when satisfied, the update focuses on the current task while carefully controlling memory loss increase to maintain generalization.

5 Gradient-Guided Epsilon Constraint Method (GEC)

To address the identified limitations of ER and to explicitly manage the trade-off in OCL, we propose the Gradient-Guided Epsilon Constraint (GEC) method. GEC treats OCL as an ϵ-constraint optimization problem, where we aim to minimize the current task loss while ensuring that the increase in average memory loss does not exceed a predefined slack tolerance $\bar{\varepsilon}$.

Let $\ell_c(\theta_k)$ be the loss on the current mini-batch at step k with parameters θ_k, and $\nabla\ell_c(\theta_k)$ its gradient. Let $\bar{\ell}_m(\theta_k)$ be the average loss on samples from the memory buffer M, and $\nabla\bar{\ell}_m(\theta_k)$ its gradient. We define a constraint function $h(\theta_k)$ that measures the extent of forgetting relative to the

slack $\bar{\varepsilon}$:

$$h(\theta_k) = \bar{\ell}_m(\theta_k) - \bar{\ell}_m(\theta_{k-1}) - \bar{\varepsilon} \tag{9}$$

We desire $h(\theta_k) \leq 0$. $\bar{\ell}_m(\theta_{k-1})$ is the average memory loss before the current update (i.e., using parameters θ_{k-1}). Similar to [Lopez-Paz and Ranzato, 2017], at each update step k, GEC determines the update direction v_k by solving the following quadratic program (QP):

$$v_k = \arg\min_{v \in \mathbb{R}^d} \left\{ \frac{1}{2} \|\nabla \ell_c(\theta_k) - v\|^2 \right\} \quad \text{s.t.} \quad \langle \nabla \bar{\ell}_m(\theta_k), v \rangle \geq \phi_h(\theta_k) \tag{10}$$

The objective of this QP is to find an update direction v that is as close as possible to the current task's gradient $\nabla \ell_c(\theta_k)$. The constraint $\langle \nabla \bar{\ell}_m(\theta_k), v \rangle \geq \phi_h(\theta_k)$ modulates this update based on the state of the memory constraint $h(\theta_k)$. The term $\phi_h(\theta_k)$ is a dynamic barrier function defined as:

$$\phi_h(\theta_k) = \alpha \cdot h(\theta_k) = \alpha \left(\bar{\ell}_m(\theta_k) - \bar{\ell}_m(\theta_{k-1}) - \bar{\varepsilon} \right) \tag{11}$$

where $\alpha > 0$ is a hyperparameter controlling the strength of the barrier.

The behavior of this formulation is as follows:

If $h(\theta_k) > 0$ (Constraint Violated): $\phi_h(\theta_k) > 0$. The QP constraint $\langle \nabla \bar{\ell}_m(\theta_k), v \rangle \geq \phi_h(\theta_k)$ forces the update v_k to have a sufficiently positive projection onto $\nabla \bar{\ell}_m(\theta_k)$. Since the parameter update is $\theta_{k+1} = \theta_k - \eta v_k$, a positive projection $\langle \nabla \bar{\ell}_m(\theta_k), v_k \rangle$ aims to decrease $\bar{\ell}_m$ (or slow its increase significantly), thus working to satisfy $h(\theta_k) \leq 0$. The update prioritizes reducing constraint violation.

If $h(\theta_k) \leq 0$ (Constraint Satisfied): $\phi_h(\theta_k) \leq 0$. The QP constraint $\langle \nabla \bar{\ell}_m(\theta_k), v \rangle \geq \phi_h(\theta_k)$ allows v_k to have a negative projection onto $\nabla \bar{\ell}_m(\theta_k)$ (up to the point dictated by $\phi_h(\theta_k)$), meaning $\bar{\ell}_m$ is allowed to increase. This permits more aggressive optimization of the current task loss $\ell_c(\theta_k)$. The magnitude of $\phi_h(\theta_k)$ controls how much $\bar{\ell}_m$ is allowed to increase, preventing excessive forgetting or overfitting to the memory buffer by trying to decrease $\bar{\ell}_m$ too much.

The solution to the QP in Eq. (10) is:

$$v_k = \nabla \ell_c(\theta_k) + \lambda_k \nabla \bar{\ell}_m(\theta_k) \tag{12}$$

where the coefficient λ_k is:

$$\lambda_k = \max \left(\frac{\phi_h(\theta_k) - \langle \nabla \ell_c(\theta_k), \nabla \bar{\ell}_m(\theta_k) \rangle}{\|\nabla \bar{\ell}_m(\theta_k)\|^2 + \delta}, 0 \right) \tag{13}$$

A small constant $\delta > 0$ is added for numerical stability. The model parameters are then updated: $\theta_{k+1} \leftarrow \theta_k - \eta v_k$, where η is the learning rate. While the formulation above considers a single aggregate memory constraint for clarity, in practice, when task identities are available in the memory buffer, the constraint can be naturally extended to handle per-task memory samples, leading to a multi-constraint optimization that better preserves task-specific knowledge. Since the slack tolerance $\bar{\varepsilon}$ is defined relative to a reference memory loss, we employ a dynamic threshold strategy that calibrates $\bar{\varepsilon}$ based on the observed loss at task boundaries, along with a warm-up phase to establish stable initial estimates. Detailed implementation choices and the full algorithm are provided in Appendix D and Appendix E.

5.1 Theoretical Properties

The GEC method exhibits desirable convergence characteristics within its constrained optimization framework. We analyze its behavior using a continuous-time analogue $d\theta/dt = -v(\theta)$, where $v(\theta)$ is the GEC update direction. Let $\ell_c(\theta)$ be the current task loss and $h(\theta) = \bar{\ell}_m(\theta) - \bar{\varepsilon}$ be the constraint violation measure. The dynamic barrier is $\phi_h(\theta) = \alpha h(\theta)$. The gradients are $\nabla \ell_c(\theta)$ and $\nabla h(\theta) = \nabla \bar{\ell}_m(\theta)$. The update $v(\theta) = \nabla \ell_c(\theta) + \lambda(\theta) \nabla \bar{\ell}_m(\theta)$. Then the properties of GEC are:

Constraint Adherence: When the constraint is violated, i.e., $h(\theta) > 0$, the GEC update actively works to reduce this violation. The measure of constraint violation $[h(\theta)]_+ = \max(h(\theta), 0)$ is non-increasing. Specifically, in continuous time, $\frac{d}{dt}[h(\theta_t)]_+ \leq -[\phi_h(\theta_t)]_+$. Since $\phi_h(\theta_t) = \alpha h(\theta_t) > 0$ when $h(\theta_t) > 0$, this implies $[h(\theta_t)]_+$ decreases, guiding the model towards satisfying the constraint $h(\theta_t) \leq 0$.

Objective Minimization under Satisfied Constraints: When the constraint is satisfied, i.e., $h(\theta) \leq 0$, GEC prioritizes minimizing the current task loss $\ell_c(\theta)$. The current task loss $\ell_c(\theta)$ is non-increasing: $\frac{d}{dt}\ell_c(\theta_t) \leq -\|v(\theta_t)\|^2 - \lambda(\theta_t)[-\phi_h(\theta_t)]_+ \leq 0$. This allows the model to focus on learning the new task while the dynamic barrier ensures that $h(\theta)$ does not grow uncontrollably.

These properties suggest that GEC offers a principled way to dynamically balance learning new information and preserving past knowledge by adaptively managing the ϵ-constraint based on the observed forgetting and the slack tolerance $\bar{\varepsilon}$.

Table 1: Performance comparison (AAA and Acc) with baseline methods on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet using Reduced ResNet-18. Results are averaged over 5 runs.

Method	Seq-CIFAR10 (N=5)				Seq-CIFAR100 (N=20)				Seq-TinyImageNet (N=20)			
	$\|\mathcal{M}\| = 0.6$k		$\|\mathcal{M}\| = 1$k		$\|\mathcal{M}\| = 1$k		$\|\mathcal{M}\| = 5$k		$\|\mathcal{M}\| = 2$k		$\|\mathcal{M}\| = 5$k	
	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc
SGD	34.04	16.68	34.04	16.68	9.67	3.24	9.67	3.24	7.63	2.17	7.63	2.17
ER	54.68	39.43	54.91	42.04	17.86	11.89	20.67	14.87	16.27	10.52	16.10	11.89
Refresh CL	62.34	51.42	65.81	55.57	24.32	17.64	35.61	32.33	18.59	**12.83**	22.68	18.72
DER	49.06	25.80	48.25	23.90	10.96	3.71	10.47	3.68	8.04	2.46	7.65	2.04
DER++	57.17	47.03	61.01	50.31	17.30	8.72	17.08	8.98	12.42	5.57	11.93	5.26
CLSER	61.64	50.36	63.27	53.06	22.58	15.68	23.25	16.42	18.50	10.03	18.88	11.61
CBA	63.41	51.47	65.47	52.08	22.46	15.54	23.07	15.87	18.79	11.43	18.98	10.98
POCL	63.62	53.42	66.23	58.50	26.68	16.54	36.34	33.36	21.56	12.69	25.48	19.40
EWC	36.51	18.37	36.51	18.37	9.87	2.77	9.87	2.77	7.96	2.43	7.96	2.43
GEM	37.78	18.84	37.00	18.73	13.43	6.04	13.71	6.46	10.17	3.70	10.27	3.81
A-GEM	37.67	18.51	37.62	18.03	10.61	3.75	10.80	3.52	7.66	2.33	7.79	2.40
GEC(Ours)	**65.21**	**54.08**	**69.42**	**59.12**	**29.63**	**17.78**	**38.11**	**34.22**	**22.45**	12.77	**27.66**	**20.25**

6 Experiments

6.1 Experiment Setup

Datasets Following [Buzzega et al., 2020, Arani et al., 2022, Lai et al., 2025], we evaluate our method on three commonly used benchmarks for online continual learning: Sequential CIFAR-10 (**Seq-CIFAR10**), Sequential CIFAR-100 (**Seq-CIFAR100**) [Krizhevsky et al., 2009], and Sequential TinyImageNet (**Seq-TinyImageNet**) [Buzzega et al., 2020]. For Seq-CIFAR10, the dataset is split into 5 tasks, each containing 2 classes. We test with memory buffer sizes ($|\mathcal{M}|$) of 0.6k and 1k samples. For Seq-CIFAR100, the dataset is split into 20 tasks, each with 5 classes, using memory buffer sizes of 1k and 5k samples. Seq-TinyImageNet is split into 20 tasks (10 classes per task), with memory buffer sizes of 2k and 5k.

Evaluation Metrics. To comprehensively evaluate performance, let $a_{i,j}$ denote the accuracy of the model on task i after training sequentially up to task j. Let T be the total number of tasks. We use the following standard metrics [Soutif-Cormerais et al., 2023]: **Average Anytime Accuracy (AAA)** measures the average performance over the learning process: $\text{AAA} = \frac{1}{T}\sum_{j=1}^{T}(\frac{1}{j}\sum_{i=1}^{j} a_{i,j})$. **Average Accuracy (Acc)** is average accuracy after training T tasks: $\text{Acc} = \frac{1}{T}\sum_{i=1}^{T} a_{i,T}$. **Forgetting Measure (FM)** quantifies how much the model forgets past tasks: $\text{FM} = \frac{1}{T-1}\sum_{i=1}^{T-1}(a_{i,T} - a_{i,i}^{\max})$, where $a_{i,i}^{\max}$ is the peak accuracy on task i observed after training on task i. A less negative FM indicates less forgetting.

To further dissect the learning dynamics, particularly for visualizing the trade-off managed by GEC, we decompose the performance into: **Plasticity (Acc-P)**: The model's ability to learn the current task, measured as $P = \frac{1}{T}\sum_{k=1}^{T} a_{k,k}$. **Stability (Acc-S)**: The model's ability to retain knowledge of past tasks, measured as $S = \frac{1}{T-1}\sum_{k=1}^{T-1}(a_{T,k} - a_{k,k})$.

Implementation Details. Consistent with [Chrysakis and Moens, 2023], we use the Reduced ResNet-18 [He et al., 2016] as the backbone network for all experiments. Models are trained using

Table 2: Forgetting Measure (FM) comparison with baseline methods on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet using Reduced ResNet-18. Results are averaged over 5 runs.

Method	Seq-CIFAR10 (N=5)		Seq-CIFAR100 (N=20)		Seq-TinyImageNet (N=20)	
	$\|\mathcal{M}\| = 0.6$k	$\|\mathcal{M}\| = 1$k	$\|\mathcal{M}\| = 1$k	$\|\mathcal{M}\| = 5$k	$\|\mathcal{M}\| = 2$k	$\|\mathcal{M}\| = 5$k
SGD	-61.01$\pm$3.30	-61.01$\pm$3.30	-54.24$\pm$1.20	-54.24$\pm$1.20	-43.58$\pm$0.58	-43.58$\pm$0.58
ER	-35.68$\pm$3.20	-32.16$\pm$5.10	-46.25$\pm$0.47	-48.54$\pm$1.10	-37.90$\pm$0.28	-37.43$\pm$0.81
Refresh CL	-25.11$\pm$1.34	-19.63$\pm$0.89	-39.83$\pm$1.05	-20.34$\pm$2.120	-36.63$\pm$1.09	-25.13$\pm$1.27
DER	-54.60$\pm$2.80	-55.22$\pm$1.70	-60.58$\pm$0.38	-59.94$\pm$1.40	-47.11$\pm$0.65	-46.27$\pm$0.85
DER++	-24.00$\pm$1.30	-28.01$\pm$1.31	-58.61$\pm$0.60	-61.23$\pm$0.19	-47.13$\pm$2.00	-47.65$\pm$0.76
CLSER	-29.03$\pm$3.70	-31.38$\pm$1.70	-45.63$\pm$0.62	-50.71$\pm$0.86	-44.45$\pm$0.32	-44.21$\pm$0.14
CBA	-27.15$\pm$4.70	-27.31$\pm$3.70	-44.30$\pm$1.53	-50.16$\pm$2.76	-38.17$\pm$1.56	-40.53$\pm$1.34
POCL	-23.27$\pm$2.50	-16.85$\pm$2.80	-37.61$\pm$0.63	-17.55$\pm$0.36	-31.63$\pm$1.60	-20.22$\pm$1.20
EWC	-64.21$\pm$0.86	-64.21$\pm$0.86	-56.55$\pm$1.30	-56.55$\pm$1.30	-44.30$\pm$1.70	-44.30$\pm$1.70
GEM	-58.09$\pm$3.90	-57.64$\pm$2.70	-45.83$\pm$0.69	-50.61$\pm$1.90	-42.30$\pm$0.11	-42.71$\pm$0.03
A-GEM	-66.17$\pm$1.61	-66.61$\pm$1.60	-60.51$\pm$0.34	-60.89$\pm$0.37	-45.61$\pm$0.04	-45.51$\pm$0.27
GEC(Ours)	**-21.52**$\pm$**2.21**	**-15.69**$\pm$**1.62**	**-34.49**$\pm$**0.89**	**-15.12**$\pm$**1.07**	**-30.13**$\pm$**1.43**	**-19.54**$\pm$**1.89**

Stochastic Gradient Descent (SGD) with a batch size of 32. All experiments are conducted in the online class-incremental learning setting, where each task is trained for a single epoch, and task identity is not available during inference. All reported results are averaged over 5 independent runs. Further details on the experimental setup and baseline methods can be found in Appendix E.

Baseline Methods. We compare GEC with a comprehensive set of baseline methods, including: Simple SGD, ER [Rolnick et al., 2019], regularization-based methods like Elastic Weight Consolidation (EWC) [Huszár, 2018], gradient projection methods like Gradient Episodic Memory (GEM) [Lopez-Paz and Ranzato, 2017] and Averaged GEM (A-GEM) [Chaudhry et al., 2018], and more advanced rehearsal-based methods such as Dark Experience Replay (DER) [Buzzega et al., 2020], DER with logits (DER++) [Buzzega et al., 2020], CLSER [Arani et al., 2022], CBA [Wang et al., 2023], Refresh-CL [Wang et al., 2024], and Pareto Optimized Continual Learning (POCL) [Wu et al., 2024b] . For some of the baseline experimental results, we followed the reported results from [Wu et al., 2024b]. Detailed descriptions of these baselines are provided in Appendix E.

6.2 Experimental Results

To assess the effectiveness of GEC, we perform a comprehensive set of experiments and conduct ablation studies to address the following key questions:

Question 1: How does GEC perform on OCL benchmarks? Table 1 presents the AAA and Acc on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet. The methods in the bottom row are the methods that utilize hard and explicit constraints while the middle and top row are the methods of representative OCL and CL methods, respectively. Our proposed GEC method consistently demonstrates strong performance across all datasets and memory buffer configurations. Notably, GEC surpasses other baseline methods, including strong rehearsal-based techniques CLSER and those designed for specific aspects of online continual learning. This suggests that GEC's dynamic constraint manA-GEMent effectively balances learning new tasks and preserving old knowledge. It is noteworthy that on Seq-CIFAR100 and Seq-TinyImageNet, GEC achieves a significant improvement in AAA compared to most baselines, highlighting its efficacy in more challenging, longer task sequences.

Question 2: How effective is GEC in preventing catastrophic forgetting? Table 2 shows the Forgetting Measure (FM). A higher FM (less negative) indicates less forgetting. GEC demonstrates superior forgetting mitigation compared to many baselines. This underscores GEC's ability to preserve learned knowledge by explicitly constraining the increase in memory loss when forgetting exceeds the tolerance $\bar{\varepsilon}$. The superior performance of GEC across these diverse settings on FM validates our hypothesis that explicitly and dynamically managing the epsilon-constraint on memory loss leads to a more robust and effective OCL strategy. The careful control over constraint relaxation when performance is within tolerance also prevents overfitting to the buffer, which can indirectly aid in better long-term retention.

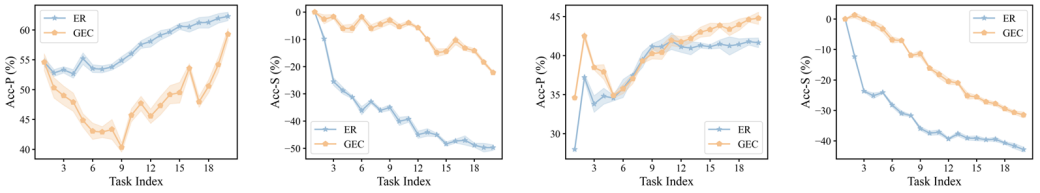

Figure 3: Dynamics of Plasticity (Acc-P) and Stability (Acc-S) for GEC and ER on Seq-CIFAR10 and Seq-TinyImageNet with $|\mathcal{M}| = 1$k.

Question 3: How does the adaptive weighting strategy affect stability-plasticity dynamics? To further investigate how GEC balances learning new information (plasticity) and retaining past knowledge (stability), we plot the dynamics of Acc-P and Acc-S during the training process on Seq-CIFAR100 and Seq-TinyImageNet, comparing GEC with ER (Figure 3). GEC demonstrates a more favorable trajectory in the stability-plasticity space. In the training of the first few tasks when memory constraints are violated (not shown directly, but implied by potential drops in stability for ER), GEC's mechanism to prioritize constraint satisfaction helps maintain better stability (higher Acc-S) without unduly sacrificing plasticity (Acc-P on the current task). In the later stages of training, when the model stability remains stable, GEC's focus on plasticity, while controlling the increase rate of memory loss, allows it to achieve competitive (Seq-CIFAR100) or superior (Seq-TinyImageNet) plasticity compared to ER, which might overfit to the buffer. This dynamic adjustment contributes to GEC's overall better performance.

Question 4: How does the slack tolerance navigates stability-plasticity pareto front? Our GEC method, by design, allows for exploring different trade-offs between current task learning and past task retention through the slack tolerance parameter $\bar{\varepsilon}$. Figure 4 qualitatively illustrates how varying $\bar{\varepsilon}$ can allow GEC to trace out a pareto front in Seq-CIFAR10 with $|\mathcal{M}|$=1k , where each point represents a different balance. A smaller $\bar{\varepsilon}$ would correspond to a stricter constraint on forgetting, potentially leading to higher stability at the cost of plasticity, while a larger $\bar{\varepsilon}$ allows for more flexibility. This controllability can successfully provide a set of well-representative Pareto solutions. Additionally, GEC's solutions dominate all ER solutions.

Question 5: How does GEC algorithm efficiency compare to other OCL algorithms? Balancing computational cost and performance is a practical challenge in continual learning, particularly in online settings [Verwimp et al., 2023]. We analyze the time complexity of GEC by comparing its training time with other representative OCL methods on the Seq-CIFAR10 dataset, as shown in Table. While GEC introduces additional computational steps compared to the simpler ER, it achieves this superior performance with significantly less overhead than other recent advanced methods like POCL and CBA, highlighting a favorable trade-off between performance and efficiency.

Table 3: Training time comparison of different OCL Methods on Seq-CIFAR10.

Method	Time (s)	AAA	Acc	FM
ER	**86.31**	54.91	42.02	-32.16
POCL	474.31	66.23	58.50	-16.85
CBA	274.31	65.47	52.08	-27.31
GEC(Ours)	147.62	**69.42**	**59.12**	**-15.69**

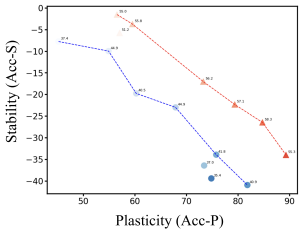

Figure 4: Pareto front achievable by GEC (Red) and ER (Blue).

7 Related Work

Our work is situated within the extensive body of research on Continual Learning, particularly focusing on rehearsal-based strategies for OCL setting. CL methods are broadly categorized into regularization-based, architecture-based, and rehearsal-based approaches [De Lange et al., 2021].

Regularization methods like EWC [Kirkpatrick et al., 2017] and SI [Zenke et al., 2017] penalize changes to important parameters but can be overly conservative.

Rehearsal-based methods, which are highly relevant to our work, maintain a small memory buffer of past examples. The foundational method, ER [Rolnick et al., 2019], co-trains on current and past data, and our work provides a novel ϵ-constraint interpretation of its mechanism. Constraint-based methods like GEM [Lopez-Paz and Ranzato, 2017] and A-GEM [Chaudhry et al., 2018] use the buffer to project gradients, enforcing a hard constraint against increasing past task loss. In contrast, GEC proposes a dynamic, soft-constraint approach, offering a more flexible balance. More recent OCL methods like POCL [Wu et al., 2024b] and CBA [Wang et al., 2023] also tackle the stability-plasticity trade-off, with POCL employing multi-objective optimization perspectives. GEC contributes to this line of work by offering a principled and efficient gradient-guided mechanism rooted in ϵ-constraint optimization. A more comprehensive review of related literature is available in Appendix B.

8 Conclusion

This paper revisited Experience Replay (ER) in Online Continual Learning (OCL) from an ϵ-constraint optimization perspective, highlighting limitations of its implicit and fixed slack strategy. We introduced the Gradient-Guided Epsilon Constraint (GEC) method, which explicitly formulates OCL updates as a dynamically adjusted ϵ-constraint problem solved via a quadratic program. GEC modulates gradient updates based on a predefined slack tolerance $\bar{\varepsilon}$, prioritizing constraint satisfaction when forgetting is high and focusing on current task learning while controlling memory loss otherwise, thereby mitigating catastrophic forgetting and preventing memory overfitting. Comprehensive experiments on standard OCL benchmarks substantiated GEC's superiority in achieving a better stability-plasticity trade-off and outperforming state-of-the-art methods.

Acknowledgements

This work was supported in part by the National Natural Science Foundations of China (Grant No. 62376267), the Beijing Natural Science Foundation (Grant No. L253019), the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF Project No. CityU 11215723), and the innoHK project. We would also like to thank Chengyu Lu of City University of Hong Kong for his valuable discussions and helpful insights.

References

R. Aljundi. Continual learning in neural networks. *arXiv preprint arXiv:1910.02718*, 2019.

R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

E. Arani, F. Sarfraz, and B. Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

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

P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

V. Chankong and Y. Y. Haimes. *Multiobjective decision making: theory and methodology*. Courier Dover Publications, 2008.

A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420, 2018. URL `https://api.semanticscholar.org/CorpusID:54443381`.

A. Chrysakis and M.-F. Moens. Online bias correction for task-free continual learning. *ICLR 2023 at OpenReview*, 2023.

M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4): 128–135, 1999.

C. Gong and X. Liu. Bi-objective trade-off with dynamic barrier gradient descent. *NeurIPS 2021*, 2021.

N. Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018.

G. Gupta, K. Yadav, and L. Paull. Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598, 2020.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

F. Huszár. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(11):E2496–E2497, 2018.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

S. Lai, Z. Zhao, F. Zhu, X. Lin, Q. Zhang, and G. Meng. Pareto continual learning: Preference-conditioned learning and adaption for dynamic stability-plasticity trade-off. In *AAAI Conference on Artificial Intelligence*, 2025. URL `https://api.semanticscholar.org/CorpusID:277452206`.

Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

A. Soutif-Cormerais, A. Carta, A. Cossu, J. Hurtado, V. Lomonaco, J. Van de Weijer, and H. Hemati. A comprehensive empirical evaluation on online continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3518–3528, 2023.

E. Verwimp, R. Aljundi, S. Ben-David, M. Bethge, A. Cossu, A. Gepperth, T. L. Hayes, E. Hüllermeier, C. Kanan, D. Kudithipudi, et al. Continual learning: Applications and the road forward. *arXiv preprint arXiv:2311.11908*, 2023.

Q. Wang, R. Wang, Y. Wu, X. Jia, and D. Meng. Cba: Improving online continual learning via continual bias adaptor. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19082–19092, 2023.

Z. Wang, Y. Li, L. Shen, and H. Huang. A unified and general framework for continual learning. *arXiv preprint arXiv:2403.13249*, 2024.

Y. Wu, L.-K. Huang, R. Wang, D. Meng, and Y. Wei. Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction. In *The Twelfth international conference on learning representations*, volume 2, 2024a.

Y. Wu, H. Wang, P. Zhao, Y. Zheng, Y. Wei, and L.-K. Huang. Mitigating catastrophic forgetting in online continual learning by modeling previous task interrelations via pareto optimization. In *Forty-first International Conference on Machine Learning*, 2024b.

F. Ye and A. G. Bors. Task-free continual learning via online discrepancy distance learning. *Advances in Neural Information Processing Systems*, 35:23675–23688, 2022.

F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

Y. Zhang, B. Pfahringer, E. Frank, A. Bifet, N. J. S. Lim, and Y. Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. *Advances in Neural Information Processing Systems*, 35:14771–14783, 2022.

In this appendix, we mainly provide:

- **Section A.1 and A.2 (Theoretical Analysis):** Detailed proofs for the propositions regarding the ϵ-constraint perspective of ER and the theoretical properties of the GEC method.
- **Section B (Detailed Related Works):** A review of existing literature in Continual Learning, Online Continual Learning, and relevant Multi-Objective Optimization methods.
- **Section C (Additional Results):** Supplementary experimental results.
- **Section D (Algorithm Details):** A pseudo-code implementation of the GEC algorithm.
- **Section E (Experimental Setup Details):** A comprehensive description of the datasets, implementation specifics, and baseline methods used in the empirical evaluation.
- **Section F (Limitations):** A discussion on limitations of the proposed GEC method.
- **Section G (Broader Impact):** A consideration of the potential broader impact and societal implications of this research in Online Continual Learning.

A Theoretical Analysis

Notation Throughout this appendix, we use the following notation: vectors are denoted by lowercase bold letters (e.g., $\mathbf{x} \in \mathbb{R}^n$), matrices by uppercase bold letters (e.g., $\mathbf{X} \in \mathbb{R}^{n \times d}$), and functions by bold letters when they are vector-valued (e.g., $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^k$). Scalars are denoted by regular letters. The transpose of a vector or matrix is denoted by $(\cdot)^\top$.

A.1 Proofs of Propositions

Proof of Proposition 1. Let θ^* be a solution to the Experience Replay (ER) optimization problem given by Eq. (4):

$$\min_{\theta} \left(\ell_c(f(X_c;\theta), Y_c) + \lambda \bar{\ell}_M(f(X_M;\theta), Y_M) \right)$$

where $\lambda > 0$. This means for any θ:

$$\ell_c(f(X_c;\theta^*), Y_c) + \lambda \bar{\ell}_M(f(X_M;\theta^*), Y_M) \leq \ell_c(f(X_c;\theta), Y_c) + \lambda \bar{\ell}_M(f(X_M;\theta), Y_M) \tag{14}$$

Let $f_1(\theta) = \ell_c(f(X_c;\theta), Y_c)$ and $f_2(\theta) = \bar{\ell}_M(f(X_M;\theta), Y_M)$. The ER objective is to minimize $f_1(\theta) + \lambda f_2(\theta)$.

We want to show that θ^* is also a solution to the following ϵ-constraint problem as defined in Eq. (5):

$$\begin{aligned} &\min_{\theta} f_1(\theta) \\ &\text{s.t. } f_2(\theta) - f_2(\theta_{k-1}) \leq \varepsilon_M^*(k) \end{aligned} \tag{15}$$

where the slack is chosen as $\varepsilon_M^*(k) = f_2(\theta^*) - f_2(\theta_{k-1})$. The constraint can be written as $f_2(\theta) \leq f_2(\theta^*)$.

Assume, for the sake of contradiction, that θ^* is not a solution to this ϵ-constraint problem. Then, there must exist some $\hat{\theta}$ such that:

$$\begin{aligned} f_1(\hat{\theta}) &< f_1(\theta^*) \\ f_2(\hat{\theta}) - f_2(\theta_{k-1}) &\leq \varepsilon_M^*(k) \end{aligned} \tag{16}$$

Now, consider the ER objective function for $\hat{\theta}$:

$$\begin{aligned} f_1(\hat{\theta}) + \lambda f_2(\hat{\theta}) &< f_1(\theta^*) + \lambda f_2(\hat{\theta}) \\ &\leq f_1(\theta^*) + \lambda f_2(\theta^*) \end{aligned} \tag{17}$$

This implies $f_1(\hat{\theta}) + \lambda f_2(\hat{\theta}) < f_1(\theta^*) + \lambda f_2(\theta^*)$, which contradicts the assumption that θ^* is a solution to the ER optimization problem (Eq. (14)). Therefore, θ^* must be a solution to the specified ϵ-constraint problem with $\varepsilon_M^*(k) = \bar{\ell}_M(f(X_M;\theta^*), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M)$. □

Assumptions for Proposition 2:

1. The loss function for the current task, $\ell_c(f(X_c;\theta), Y_c)$, denoted as $\ell_c(\theta)$, is a convex and differentiable function of θ.
2. The average loss function on memory samples, $\bar{\ell}_M(f(X_M;\theta), Y_M)$, denoted as $\bar{\ell}_M(\theta)$, is a convex and differentiable function of θ.
3. A suitable constraint qualification (e.g., Slater's condition) holds for the ϵ-constraint optimization problem defined in Eq. (6). Slater's condition implies that there exists a $\tilde{\theta}$ such that $\bar{\ell}_M(f(X_M;\tilde{\theta}), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M) < \varepsilon_M(k)$.

Proof of Proposition 2. Let θ^* be a solution to the ϵ-constraint optimization problem given by Eq. (6):

$$\min_{\theta} \ell_c(f(X_c;\theta), Y_c)$$
$$\text{s.t. } \bar{\ell}_M(f(X_M;\theta), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M) \leq \varepsilon_M(k)$$

Let $f_1(\theta) = \ell_c(f(X_c;\theta), Y_c)$ and $g_1(\theta) = \bar{\ell}_M(f(X_M;\theta), Y_M) - \bar{\ell}_M(f(X_M;\theta_{k-1}), Y_M) - \varepsilon_M(k)$. The problem is $\min_\theta f_1(\theta)$ subject to $g_1(\theta) \leq 0$.

Under the stated assumptions (convexity, differentiability, and constraint qualification), the Karush-Kuhn-Tucker (KKT) [Boyd and Vandenberghe, 2004] conditions are necessary and sufficient for optimality. Thus, there exists a Lagrange multiplier $\lambda^* \geq 0$ such that for θ^*:

1. **Stationarity**: $\nabla f_1(\theta^*) + \lambda^* \nabla g_1(\theta^*) = 0$
2. **Primal feasibility**: $g_1(\theta^*) \leq 0$
3. **Dual feasibility**: $\lambda^* \geq 0$
4. **Complementary slackness**: $\lambda^* g_1(\theta^*) = 0$

The Lagrangian function for this problem is $L(\theta, \lambda^*) = f_1(\theta) + \lambda^* g_1(\theta)$. Substituting the definitions of $f_1(\theta)$ and $g_1(\theta)$:

$$L(\theta, \lambda^*) = \ell_c(\theta) + \lambda^*(\bar{\ell}_M(\theta) - \bar{\ell}_M(\theta_{k-1}) - \varepsilon_M(k))$$

The stationarity condition is $\nabla_\theta L(\theta^*, \lambda^*) = 0$:

$$\nabla \ell_c(\theta^*) + \lambda^* \nabla \bar{\ell}_M(\theta^*) = 0$$

Since $\ell_c(\theta)$ and $\bar{\ell}_M(\theta)$ are convex, and $\lambda^* \geq 0$, the function $L(\theta, \lambda^*)$ is convex in θ. For a convex function, the stationarity condition $\nabla_\theta L(\theta^*, \lambda^*) = 0$ is sufficient for θ^* to be a global minimizer of $L(\theta, \lambda^*)$. Thus, θ^* minimizes $\ell_c(\theta) + \lambda^*(\bar{\ell}_M(\theta) - \bar{\ell}_M(\theta_{k-1}) - \varepsilon_M(k))$. This is equivalent to minimizing $\ell_c(\theta) + \lambda^* \bar{\ell}_M(\theta) - \lambda^* \bar{\ell}_M(\theta_{k-1}) - \lambda^* \varepsilon_M(k)$. Since $\bar{\ell}_M(\theta_{k-1})$ and $\varepsilon_M(k)$ are constants for the current optimization step k, the terms $-\lambda^* \bar{\ell}_M(\theta_{k-1})$ and $-\lambda^* \varepsilon_M(k)$ are also constants. Therefore, θ^* also minimizes the function $\ell_c(\theta) + \lambda^* \bar{\ell}_M(\theta)$, which is the objective function of Experience Replay (Eq. (4)). □

A.2 Properties of the GEC

We analyze the behavior of the GEC update mechanism by considering a continuous-time analogue, where the parameter update follows $\frac{d\theta_t}{dt} = -v_t$, with v_t being the solution to the QP in Eq. (10). The GEC update direction is $v_k = \nabla \ell_c(\theta_k) + \lambda_k \nabla \bar{\ell}_m(\theta_k)$. Let $h(\theta_t) = \bar{\ell}_m(\theta_t) - \bar{\ell}_m(\theta_{k-1}) - \bar{\varepsilon}$ be the constraint function, and $\phi_h(\theta_t) = \alpha h(\theta_t)$ be the dynamic barrier. For brevity, we use $\ell_c(\theta_t)$, $\bar{\ell}_m(\theta_t)$, $\nabla \ell_c$, $\nabla \bar{\ell}_m$, λ_t, and v_t. The KKT conditions for the QP (Eq. (10)) imply that $\lambda_t \geq 0$ and $\lambda_t(\langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle - \phi_h(\theta_t)) = 0$.

Following [Gong and Liu, 2021], we define a penalty function $P_\mu(\theta_t) = \ell_c(\theta_t) + \mu[h(\theta_t)]_+$, where $[x]_+ = \max(x, 0)$ and $\mu \geq 0$. The time derivative of $P_\mu(\theta_t)$ can be shown to be:

$$\frac{d}{dt} P_\mu(\theta_t) \leq -\|v_t\|^2 - (\mu - \lambda_t)\left[\phi_h(\theta_t)\right]_+ - \lambda_t\left[-\phi_h(\theta_t)\right]_+ \tag{18}$$

This can be written as $\frac{d}{dt} P_\mu(\theta_t) \leq -K_{\mu-\lambda_t}(\theta_t, \lambda_t)$, where $K_\nu(\theta_t, \lambda_t) = \|v_t\|^2 + \nu\left[\phi_h(\theta_t)\right]_+ + \lambda_t\left[-\phi_h(\theta_t)\right]_+$ is a KKT-like score function.

Property 1: Constraint Violation Reduction. The thresholded constraint function $[h(\theta_t)]_+$ is non-increasing with time t.

Proof of Constraint Violation Reduction. Consider the derivative of the penalty function $P_\mu(\theta_t) = \ell_c(\theta_t) + \mu[h(\theta_t)]_+$. If $h(\theta_t) > 0$, then $[h(\theta_t)]_+ = h(\theta_t)$ and $\phi_h(\theta_t) = \alpha h(\theta_t) > 0$, so $[\phi_h(\theta_t)]_+ = \phi_h(\theta_t)$ and $[-\phi_h(\theta_t)]_+ = 0$. The derivative is $\frac{d}{dt}P_\mu(\theta_t) = -\langle \nabla \ell_c(\theta_t) + \mu \nabla \bar{\ell}_m(\theta_t), v_t \rangle$. Using $\nabla \ell_c(\theta_t) = v_t - \lambda_t \nabla \bar{\ell}_m(\theta_t)$:

$$\begin{aligned}\frac{d}{dt}P_\mu(\theta_t) &= -\langle v_t - \lambda_t \nabla \bar{\ell}_m(\theta_t) + \mu \nabla \bar{\ell}_m(\theta_t), v_t \rangle \\ &= -\|v_t\|^2 - (\mu - \lambda_t)\langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle\end{aligned}$$

From the QP constraint, $\langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle \geq \phi_h(\theta_t)$. If $\lambda_t > 0$, KKT implies $\langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle = \phi_h(\theta_t)$. If $\lambda_t = 0$, then $\langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle \geq \phi_h(\theta_t)$ still holds. So, $\frac{d}{dt}P_\mu(\theta_t) \leq -\|v_t\|^2 - (\mu - \lambda_t)\phi_h(\theta_t) = -\|v_t\|^2 - (\mu - \lambda_t)[\phi_h(\theta_t)]_+$.

If $h(\theta_t) \leq 0$, then $[h(\theta_t)]_+ = 0$, so $P_\mu(\theta_t) = \ell_c(\theta_t)$. Also, $\phi_h(\theta_t) \leq 0$, so $[\phi_h(\theta_t)]_+ = 0$ and $[-\phi_h(\theta_t)]_+ = -\phi_h(\theta_t)$.

$$\begin{aligned}\frac{d}{dt}P_\mu(\theta_t) = \frac{d}{dt}\ell_c(\theta_t) &= -\langle \nabla \ell_c(\theta_t), v_t \rangle \\ &= -\|v_t\|^2 + \lambda_t \langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle\end{aligned}$$

If $\lambda_t > 0$, $\langle \nabla \bar{\ell}_m(\theta_t), v_t \rangle = \phi_h(\theta_t)$. If $\lambda_t = 0$, $v_t = \nabla \ell_c(\theta_t)$, and $\lambda_t \langle \cdot \rangle = 0 = \lambda_t \phi_h(\theta_t)$. So this holds generally. $\frac{d}{dt}\ell_c(\theta_t) = -\|v_t\|^2 + \lambda_t \phi_h(\theta_t) = -\|v_t\|^2 - \lambda_t[-\phi_h(\theta_t)]_+$.

Combining both cases ($h(\theta_t) > 0$ and $h(\theta_t) \leq 0$), we obtain the unified inequality in Eq. (18), which holds for all values of θ_t. Now, to show $[h(\theta_t)]_+$ is non-increasing, we can analyze its derivative:

$$\frac{d}{dt}[h(\theta_t)]_+ = \begin{cases} \frac{d}{dt}h(\theta_t) & \text{if } h(\theta_t) > 0 \\ 0 & \text{if } h(\theta_t) < 0 \end{cases} \tag{19}$$

Alternatively, from Eq. (18), dividing by μ and taking $\mu \to +\infty$:

$$\lim_{\mu \to \infty} \frac{1}{\mu}\frac{d}{dt}P_\mu(\theta_t) = \frac{d}{dt}[h(\theta_t)]_+$$

And from the RHS of Eq. (18):

$$\lim_{\mu \to \infty} \frac{1}{\mu}\left(-\|v_t\|^2 - (\mu - \lambda_t)\left[\phi_h(\theta_t)\right]_+ - \lambda_t\left[-\phi_h(\theta_t)\right]_+ \right) = -[\phi_h(\theta_t)]_+$$

Thus,

$$\frac{d}{dt}[h(\theta_t)]_+ \leq -[\phi_h(\theta_t)]_+$$

Since $\phi_h(\theta_t) = \alpha h(\theta_t)$:
If $h(\theta_t) > 0$, then $\phi_h(\theta_t) > 0$, so $[\phi_h(\theta_t)]_+ > 0$. This means $\frac{d}{dt}[h(\theta_t)]_+ < 0$ when $h(\theta_t) > 0$.
If $h(\theta_t) \leq 0$, then $\phi_h(\theta_t) \leq 0$, so $[\phi_h(\theta_t)]_+ = 0$. This means $\frac{d}{dt}[h(\theta_t)]_+ \leq 0$.
Therefore, $[h(\theta_t)]_+$ is always non-increasing. This implies that if the constraint is violated ($h(\theta_t) > 0$), the algorithm works to reduce the violation. If the constraint is satisfied ($h(\theta_t) \leq 0$), it does not become violated. □

Property 2: Current Task Optimization when Constraint is Satisfied. If the constraint is satisfied (i.e., $h(\theta_t) \leq 0$), the current task objective $\ell_c(\theta_t)$ is non-increasing with time t.

Proof of Current Task Optimization when Constraint is Satisfied. When $h(\theta_t) \leq 0$, we have $\phi_h(\theta_t) \leq 0$. This implies $[\phi_h(\theta_t)]_+ = 0$. Consider Eq. (18) with $\mu = 0$. In this case, $P_0(\theta_t) = \ell_c(\theta_t)$.

$$\begin{aligned}\frac{d}{dt}\ell_c(\theta_t) &\leq -\|v_t\|^2 - (0 - \lambda_t)\left[\phi_h(\theta_t)\right]_+ - \lambda_t\left[-\phi_h(\theta_t)\right]_+ \\ &= -\|v_t\|^2 + \lambda_t \cdot 0 - \lambda_t\left[-\phi_h(\theta_t)\right]_+ \\ &= -\|v_t\|^2 - \lambda_t\left[-\phi_h(\theta_t)\right]_+\end{aligned}$$

Since $\lambda_t \geq 0$ by definition (Eq. (13)), $\|v_t\|^2 \geq 0$, and $[-\phi_h(\theta_t)]_+ \geq 0$ (because $\phi_h(\theta_t) \leq 0$), it follows that:

$$\frac{d}{dt}\ell_c(\theta_t) \leq 0$$

Thus, when the memory constraint $h(\theta_t) \leq 0$ is satisfied, the loss on the current task $\ell_c(\theta_t)$ is non-increasing, meaning GEC focuses on optimizing the current task. □

The theoretical properties established above naturally extend to the multi-constraint formulation used in our implementation. When considering per-task constraints $h_t(\theta) = \ell_t(\theta) - \bar{\ell}_t^{\text{ref}} - \bar{\varepsilon}$ for each past task t, the constraint adherence property holds independently for each task, ensuring that $[h_t(\theta)]_+$ is non-increasing for all t. The objective minimization property similarly extends: when all constraints are satisfied (i.e., $h_t(\theta) \leq 0$ for all t), the algorithm prioritizes minimizing the current task loss. The multi-constraint QP solution aggregates the per-task adjustments through the Lagrange multipliers λ_t, providing a principled way to balance multiple competing constraints while maintaining the convergence guarantees of the single-constraint case.

B Detailed Related Works

B.1 Continual Learning

Continual Learning (CL) aims to enable models to learn sequentially from a stream of data without catastrophically forgetting previously acquired knowledge [De Lange et al., 2021]. Various strategies have been proposed to address this challenge, broadly categorized into regularization-based, rehearsal-based, and architecture-based methods. Our work primarily intersects with rehearsal-based methods in the context of Online Continual Learning (OCL).

B.1.1 Regularization-based Methods

Regularization-based approaches introduce additional terms to the loss function to penalize changes in parameters deemed important for previous tasks. Elastic Weight Consolidation (EWC) [Kirkpatrick et al., 2017] computes a Fisher Information Matrix to estimate parameter importance. Synaptic Intelligence (SI) [Zenke et al., 2017] calculates importance weights online based on the sensitivity of the loss function to parameter changes. Learning without Forgetting (LwF) [Li and Hoiem, 2017] uses knowledge distillation, forcing the model to preserve the outputs of the previous model on current task data. While effective in some scenarios, these methods can be conservative and may struggle with long task sequences or significant domain shifts.

B.1.2 Replay-based Methods

Replay-based (or rehearsal-based) methods store a small subset of samples from past tasks in a memory buffer and interleave them with current task data during training [Rolnick et al., 2019]. Experience Replay (ER) [Rolnick et al., 2019] is a foundational technique that simply adds the loss on replayed samples to the current task loss. More advanced methods build upon ER. Dark Experience Replay (DER) [Buzzega et al., 2020] and its variant DER++ enhance ER by also replaying logits. CLSER [Arani et al., 2022] focuses on improving sample selection for the replay buffer.

Constraint-based replay methods, such as Gradient Episodic Memory (GEM) [Lopez-Paz and Ranzato, 2017] and Averaged GEM (A-GEM) [Chaudhry et al., 2018], use memory samples to constrain the gradient updates for the current task, aiming to prevent interference with past task knowledge. These methods typically enforce that the loss on past tasks does not increase. As discussed in our main paper, our proposed GEC method offers a new perspective by explicitly formulating the OCL update as an ϵ-constraint optimization problem, dynamically adjusting the trade-off between current task learning and memory constraint satisfaction, which contrasts with the fixed weighting of ER or the hard constraints of GEM/A-GEM.

B.1.3 Online Continual Learning

Online Continual Learning (OCL) represents a particularly challenging yet realistic CL scenario [Mai et al., 2022, Aljundi et al., 2019]. In OCL, data arrives in a stream, often as small mini-batches, and is typically processed in a single pass. Memory for storing past data is strictly limited.

This setting magnifies the problem of catastrophic forgetting and demands methods that are both computationally efficient and effective at balancing plasticity (learning new tasks) and stability (retaining old knowledge). The significance of OCL lies in its direct applicability to real-world systems where data is inherently sequential and non-stationary, and computational resources are constrained. Our GEC method is specifically designed for the OCL setting.

B.2 Multi-Objective Optimization Methods

Continual learning can be viewed as a multi-objective optimization problem (MOP) where the goals are to minimize the loss on the current task and to minimize forgetting of past tasks. These objectives are often conflicting. Our work leverages concepts from MOP, particularly the ϵ-constraint method.

B.2.1 Preliminaries for Multi-Objective Optimization

A general multi-objective optimization problem (MOP) can be formulated as:

$$\text{minimize} \quad \{\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))^\top\} \quad \text{subject to } \mathbf{x} \in S.$$

Here, $k \geq 2$ is the number of objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$. The vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ consists of n decision variables, belonging to a non-empty feasible region $S \subseteq \mathbb{R}^n$. The set $Z = \mathbf{f}(S) = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in S\} \subseteq \mathbb{R}^k$ is the feasible objective region in the objective space. An element $\mathbf{z} = \mathbf{f}(\mathbf{x}) \in Z$ is an objective vector.

Definition 3 (Dominance)**.** *An objective vector $\mathbf{z}^1 = \mathbf{f}(\mathbf{x}^1)$ is said to dominate another objective vector $\mathbf{z}^2 = \mathbf{f}(\mathbf{x}^2)$ (denoted $\mathbf{z}^1 \prec \mathbf{z}^2$) if $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ for at least one index j.*

Definition 4 (Pareto Optimality)**.** *A decision vector $\mathbf{x}^* \in S$ is Pareto optimal if there is no other decision vector $\mathbf{x} \in S$ such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\mathbf{x}^*)$. Equivalently, $\mathbf{x}^*$ is Pareto optimal if there is no $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one index j. The corresponding objective vector $\mathbf{z}^* = \mathbf{f}(\mathbf{x}^*)$ is a Pareto optimal objective vector. The set of all Pareto optimal objective vectors forms the Pareto front.*

Definition 5 (Ideal Objective Vector)**.** *The ideal objective vector $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^\top$ is formed by components $z_i^\star$, where each $z_i^\star$ is the minimum value of the i-th objective function $f_i(\mathbf{x})$ considered independently over the feasible region S, i.e., $z_i^\star = \min_{\mathbf{x} \in S} f_i(\mathbf{x})$ for $i = 1, \ldots, k$. The ideal objective vector is generally not attainable simultaneously for all objectives.*

Definition 6 (Utopian Objective Vector)**.** *A utopian objective vector $\mathbf{z}^{\star\star}$ is an infeasible vector strictly better than the ideal objective vector. Its components are typically defined as $z_i^{\star\star} = z_i^\star - \delta_i$ for all $i = 1, \ldots, k$, where $\delta_i > 0$ are small positive scalars.*

B.2.2 Scalarization Techniques

Scalarization methods transform a MOP into a single-objective optimization problem (or a series of them). Common techniques include the linear weighting method, the Chebyshev method, and the ϵ-constraint method.

Linear Weighting Method This method combines all objective functions into a single scalar objective by assigning a non-negative scalar weight $w_i \in \mathbb{R}$ to each objective $f_i(\mathbf{x})$. The problem is formulated as:

$$\text{minimize} \quad \sum_{i=1}^{k} w_i f_i(\mathbf{x}) \quad \text{subject to } \mathbf{x} \in S.$$

The weights $w_i \in \mathbb{R}$ are non-negative scalars ($w_i \geq 0$) and are typically normalized such that $\sum_{i=1}^{k} w_i = 1$. Different weight vectors can yield different Pareto optimal solutions. This method is simple but may not find all Pareto optimal solutions if the Pareto front is non-convex. As shown in our paper (Proposition 2), under certain conditions, an ϵ-constraint problem solution can be found by an ER-like weighted sum.

Chebyshev Method The Chebyshev method [Miettinen, 1999] aims to find a solution that minimizes the maximum weighted deviation from a reference point, often the utopian objective vector $\mathbf{z}^{\star\star}$ (or ideal $\mathbf{z}^{\star}$). The problem is:

$$\text{minimize} \quad \max_{i=1,\dots,k} [w_i(f_i(\mathbf{x}) - z_i^{\text{ref}})] \quad \text{subject to } \mathbf{x} \in S.$$

Here, $w_i > 0$ are positive weights, and $\mathbf{z}^{\text{ref}}$ is the reference point (e.g., $\mathbf{z}^{\star\star}$ or $\mathbf{z}^{\star}$). If $\mathbf{z}^{\text{ref}} = \mathbf{z}^{\star\star}$, then $f_i(\mathbf{x}) - z_i^{\star\star}$ is positive for feasible solutions. This can be reformulated as a differentiable problem:

$$\begin{aligned} \text{minimize} \quad & \alpha \\ \text{subject to} \quad & \alpha \geq w_i(f_i(\mathbf{x}) - z_i^{\text{ref}}) \quad \text{for all } i = 1, \dots, k, \\ & \mathbf{x} \in S, \end{aligned}$$

where $\alpha \in \mathbb{R}$ is an auxiliary variable. The Chebyshev method can find any Pareto optimal solution, regardless of the convexity of the Pareto front, by varying weights and the reference point.

ϵ-Constraint Method The ϵ-constraint method [Miettinen, 1999], which is central to our GEC approach, optimizes one objective function $f_\ell(\mathbf{x})$ while treating the other $k-1$ objective functions as constraints, bounded by user-defined values ε_j:

$$\begin{aligned} \text{minimize} \quad & f_\ell(\mathbf{x}) \\ \text{subject to} \quad & f_j(\mathbf{x}) \leq \varepsilon_j \quad \text{for all } j \in \{1, \dots, k\}, j \neq \ell, \\ & \mathbf{x} \in S. \end{aligned}$$

By systematically varying the choice of f_ℓ and the values of ε_j, different Pareto optimal solutions can be generated. This method is also capable of finding Pareto optimal solutions in non-convex problems and offers direct control over the trade-offs via the ε_j values. Our paper (Proposition 1) shows that ER can be interpreted as implicitly solving an ϵ-constraint problem. GEC makes this explicit.

C Additional Results

In this section, we provide supplementary results for Average Anytime Accuracy (AAA) and Average Accuracy (Acc) that include the standard deviations over 5 runs. These tables complement Table 1 in the main paper, presenting the mean ± standard deviation. To further validate the robustness of GEC, we also conduct experiments in more challenging scenarios, including label noise and longer task sequences. The results consistently demonstrate GEC's superior performance.

Table 4: Average Anytime Accuracy (AAA) comparison with baseline methods on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet using Reduced ResNet-18. Results are averaged over 5 runs (mean $\pm$std).

Method	Seq-CIFAR10 (N=5)		Seq-CIFAR100 (N=20)		Seq-TinyImageNet (N=20)	
	$\lvert\mathcal{M}\rvert = 0.6$k	$\lvert\mathcal{M}\rvert = 1$k	$\lvert\mathcal{M}\rvert = 1$k	$\lvert\mathcal{M}\rvert = 5$k	$\lvert\mathcal{M}\rvert = 2$k	$\lvert\mathcal{M}\rvert = 5$k
SGD	34.04$\pm$3.10	34.04$\pm$3.10	9.67$\pm$0.18	9.67$\pm$0.18	7.63$\pm$0.24	7.63$\pm$0.24
ER	54.68$\pm$5.11	54.91$\pm$1.65	17.86$\pm$0.26	20.67$\pm$0.68	16.27$\pm$0.10	16.10$\pm$0.24
Refresh CL	62.34$\pm$1.50	65.81$\pm$1.00	24.32$\pm$0.50	35.61$\pm$1.00	18.59$\pm$0.60	22.68$\pm$0.70
DER	49.06$\pm$1.30	48.25$\pm$1.95	10.96$\pm$0.20	10.47$\pm$0.32	8.04$\pm$0.20	7.65$\pm$0.11
DER++	57.17$\pm$1.08	61.01$\pm$1.55	17.30$\pm$0.30	17.08$\pm$0.79	12.42$\pm$0.34	11.93$\pm$0.34
CLSER	61.64$\pm$1.62	63.27$\pm$0.71	22.58$\pm$0.42	23.25$\pm$1.34	18.50$\pm$0.08	18.88$\pm$0.59
CBA	63.41$\pm$1.67	65.47$\pm$0.77	22.46$\pm$0.53	23.07$\pm$0.58	18.79$\pm$0.05	18.98$\pm$0.85
POCL	63.62$\pm$2.65	66.23$\pm$1.73	26.68$\pm$0.21	36.34$\pm$1.24	21.56$\pm$0.44	25.48$\pm$0.67
EWC	36.51$\pm$0.75	36.51$\pm$0.75	9.87$\pm$0.28	9.87$\pm$0.28	7.96$\pm$0.12	7.96$\pm$0.12
GEM	37.78$\pm$1.98	37.00$\pm$0.47	13.43$\pm$0.04	13.71$\pm$0.23	10.17$\pm$0.28	10.27$\pm$0.07
A-GEM	37.67$\pm$1.76	37.62$\pm$1.53	10.61$\pm$0.09	10.80$\pm$0.15	7.66$\pm$0.35	7.79$\pm$0.03
GEC(Ours)	**65.21**$\pm$2.20	**69.42**$\pm$1.70	**29.63**$\pm$0.90	**38.11**$\pm$1.10	**22.45**$\pm$1.40	**27.66**$\pm$1.80

Table 5: Performance on 40 tasks sequence of Seq-TinyImageNet.

Method	AAA	Acc
SGD	4.95$_{\pm 0.16}$	1.19$_{\pm 0.27}$
ER	13.75$_{\pm 0.12}$	6.82$_{\pm 0.11}$
DER++	9.39$_{\pm 0.07}$	3.76$_{\pm 0.81}$
CLSER	14.93$_{\pm 0.36}$	7.74$_{\pm 0.91}$
POCL	18.41$_{\pm 0.14}$	10.64$_{\pm 0.33}$
GEC (Ours)	**25.37**$_{\pm 0.16}$	**11.89**$_{\pm 0.31}$

Table 6: Performance under various label noise settings on Seq-CIFAR100 ($|\mathcal{M}| = 5k$).

Method	Flip1 (0.05)		Flip1 (0.2)		Flip2 (0.05)		Uniform (0.2)	
	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc
DER++	24.16	15.40	21.80	**12.84**	22.88	14.83	20.94	**12.28**
CLSER	21.69	13.34	19.18	10.69	21.96	10.53	19.62	10.47
POCL	22.38	15.41	18.99	11.41	21.59	11.30	18.82	10.06
GEC (Ours)	**26.76**	**16.50**	**22.11**	11.37	**25.88**	**15.63**	**21.05**	11.53

Table 7: Average Accuracy (Acc) comparison with baseline methods on Seq-CIFAR10, Seq-CIFAR100, and Seq-TinyImageNet using Reduced ResNet-18. Results are averaged over 5 runs (mean $\pm$std).

Method	Seq-CIFAR10 (N=5)		Seq-CIFAR100 (N=20)		Seq-TinyImageNet (N=20)	
	$\|\mathcal{M}\| = 0.6$k	$\|\mathcal{M}\| = 1$k	$\|\mathcal{M}\| = 1$k	$\|\mathcal{M}\| = 5$k	$\|\mathcal{M}\| = 2$k	$\|\mathcal{M}\| = 5$k
SGD	16.68$_{\pm 0.27}$	16.68$_{\pm 0.27}$	3.24$_{\pm 0.12}$	3.24$_{\pm 0.12}$	2.17$_{\pm 0.07}$	2.17$_{\pm 0.07}$
ER	39.43$_{\pm 3.79}$	42.04$_{\pm 4.33}$	11.89$_{\pm 0.27}$	14.87$_{\pm 0.60}$	10.52$_{\pm 0.59}$	11.89$_{\pm 0.63}$
Refresh CL	51.42$_{\pm 2.00}$	55.57$_{\pm 1.50}$	17.64$_{\pm 1.00}$	32.33$_{\pm 2.00}$	**12.83**$_{\pm 0.80}$	18.72$_{\pm 1.00}$
DER	25.80$_{\pm 0.51}$	23.90$_{\pm 0.47}$	3.71$_{\pm 0.11}$	3.68$_{\pm 0.06}$	2.46$_{\pm 0.06}$	2.04$_{\pm 0.15}$
DER++	47.03$_{\pm 1.03}$	50.31$_{\pm 1.50}$	8.72$_{\pm 0.52}$	8.98$_{\pm 1.05}$	5.57$_{\pm 0.11}$	5.26$_{\pm 0.21}$
CLSER	50.36$_{\pm 4.06}$	53.06$_{\pm 1.58}$	15.68$_{\pm 0.62}$	16.42$_{\pm 1.52}$	10.03$_{\pm 0.22}$	11.61$_{\pm 0.19}$
CBA	51.47$_{\pm 1.37}$	52.08$_{\pm 0.15}$	15.54$_{\pm 0.44}$	15.87$_{\pm 0.75}$	11.43$_{\pm 0.67}$	10.98$_{\pm 0.14}$
POCL	53.42$_{\pm 1.17}$	58.50$_{\pm 2.43}$	16.54$_{\pm 0.34}$	33.36$_{\pm 1.92}$	12.69$_{\pm 0.47}$	19.40$_{\pm 0.92}$
EWC	18.37$_{\pm 0.30}$	18.37$_{\pm 0.30}$	2.77$_{\pm 0.27}$	2.77$_{\pm 0.27}$	2.43$_{\pm 0.13}$	2.43$_{\pm 0.13}$
GEM	18.84$_{\pm 1.17}$	18.73$_{\pm 0.76}$	6.04$_{\pm 0.52}$	6.46$_{\pm 0.93}$	3.70$_{\pm 0.44}$	3.81$_{\pm 0.15}$
A-GEM	18.51$_{\pm 0.07}$	18.03$_{\pm 0.43}$	3.75$_{\pm 0.18}$	3.52$_{\pm 0.13}$	2.33$_{\pm 0.18}$	2.40$_{\pm 0.22}$
GEC(Ours)	**54.08**$_{\pm 1.80}$	**59.12**$_{\pm 2.00}$	**17.78**$_{\pm 0.70}$	**34.22**$_{\pm 1.80}$	12.77$_{\pm 1.20}$	**20.25**$_{\pm 1.50}$

D Algorithm Details

This section provides the pseudo-code for the proposed GEC method and a list of key symbols used. Table 8 lists the key symbols used in the GEC algorithm.

Table 8: List of Key Symbols in GEC Pseudo-code.

Symbol	Description
θ	Model parameters.
η	Learning rate.
α	Barrier strength parameter for GEC.
$\bar{\varepsilon}$	Slack tolerance for the memory constraint.
δ	Stability constant for GEC denominator.
M	Memory buffer storing past samples.
T	Total number of tasks.
k	Index for the current update step/iteration.
(X_c, Y_c)	Mini-batch of data from the current task.
(X_m, Y_m)	Mini-batch of data sampled from the memory buffer M.
$\ell(\theta; X, Y)$	Loss function evaluated on data (X, Y) with parameters θ.
$\nabla \ell_c(\theta)$	Gradient of the loss on the current task batch, $\nabla_\theta \ell(\theta; X_c, Y_c)$.
$\bar{\ell}_m(\theta)$	Average loss on the memory batch, AvgLoss$(\theta; X_m, Y_m)$.
$\nabla \bar{\ell}_m(\theta)$	Gradient of the average loss on the memory batch, $\nabla_\theta \bar{\ell}_m(\theta)$.
$\bar{\ell}_m^{\text{ref}}$	Reference average memory loss (value from the start of the previous update step).
h_k	Constraint violation measure at step k.
$\phi_{h,k}$	Dynamic barrier value at step k.
λ_k	Dynamically computed coefficient for the memory gradient at step k.
v_k	Final update direction at step k.
T_m	Task labels stored in memory buffer.
$\mathcal{T}$	Set of unique past tasks in the current memory batch.
$\nabla \ell_t(\theta)$	Gradient of loss for past task t.
$\phi_{h,t}$	Per-task dynamic barrier value.
$\lambda_{k,t}$	Per-task Lagrange multiplier from multi-constraint QP.
N_w	Number of warm-up steps.

Algorithm 1 outlines the training procedure for the GEC method.

Algorithm 1 Gradient-Guided Epsilon Constraint (GEC)

Require: Learning rate η, strength parameter $\alpha > 0$, tolerance $\bar{\varepsilon}$, constant $\delta > 0$.
Ensure: Final model parameters θ.
Initialize model parameters θ.
Initialize memory buffer $M \leftarrow \emptyset$.
Initialize per-task reference losses $\{\bar{\ell}_t^{\text{ref}}\}_{t=0}^{T-1} \leftarrow \{0\}$.
Initialize step counter $k \leftarrow 0$.
for each task $T_{\text{current}} = 1, 2, \ldots, T$ **do**
  for each incoming batch (X_c, Y_c) from task T_{current} **do**
    $k \leftarrow k + 1$
    if $k \leq N_w$ **or** M is empty **then**
      Compute $\nabla \ell_c(\theta)$ and update $\theta \leftarrow \theta - \eta \nabla \ell_c(\theta)$
    else
      Sample memory batch (X_m, Y_m, T_m) from M
      Identify unique past tasks $\mathcal{T} = \text{unique}(T_m)$
      for each past task $t \in \mathcal{T}$ **do**
        Extract task-t samples: $(X_m^t, Y_m^t) \leftarrow \{(x, y) : T_m = t\}$
        Compute per-task look-ahead gradient $\nabla \ell_t(\theta)$ and loss $\ell_t(\theta)$
        Compute barrier: $\phi_{h,t} \leftarrow \alpha(\ell_t(\theta) - \bar{\ell}_t^{\text{ref}} - \bar{\varepsilon})$
      end for
      Compute current task gradient $\nabla \ell_c(\theta)$
      Solve multi-constraint QP for $\{\lambda_t\}_{t \in \mathcal{T}}$:
      $\min_v \frac{1}{2}\|v - \nabla \ell_c(\theta)\|^2$ s.t. $\langle \nabla \ell_t(\theta), v \rangle \geq \phi_{h,t}, \forall t$
      Compute update: $v \leftarrow \nabla \ell_c(\theta) + \sum_{t \in \mathcal{T}} \lambda_t \nabla \ell_t(\theta)$
      Update parameters: $\theta \leftarrow \theta - \eta v$
      Update reference losses: $\bar{\ell}_t^{\text{ref}} \leftarrow \ell_t(\theta), \forall t \in \mathcal{T}$
    end if
    Add $(X_c, Y_c, T_{\text{current}})$ to memory M via reservoir sampling
  end for
  Record task completion loss for threshold calibration
end for
return θ.

E Experimental Setup Details

E.1 Datasets

We evaluate our GEC method on three standard Online Continual Learning (OCL) benchmarks. The details of these datasets and their configuration for our experiments are summarized in Table 9. For all datasets, standard data augmentation techniques are applied during training.

Table 9: Details of datasets

Dataset	Total Classes	Tasks (N)	Classes per Task	Training Images	Test Images	Resolution
Seq-CIFAR10	10	5	2	50,000	10,000	$32 \times 32 \times 3$
Seq-CIFAR100	100	20	5	50,000	10,000	$32 \times 32 \times 3$
Seq-TinyImageNet	200	20	10	100,000	10,000	$32 \times 32 \times 3$

E.2 Implementation Details

Consistent with recent works [Buzzega et al., 2020, Chrysakis and Moens, 2023], we use a Reduced ResNet-18 [He et al., 2016] as the backbone network for all experiments. The models are trained using SGD. The learning rate is set to 0.03 for GEC and all baseline methods. The streaming batch size and replay batch size are set to 32 for all experiments. The memory buffer M is updated using reservoir sampling.

GEC Hyperparameters. For our GEC method, the barrier strength parameter α is set to 0.5, and the stability constant δ is set to 10^{-6}. We employ a dynamic threshold mode for $\bar{\varepsilon}$: at the completion of each task, the average training loss is recorded as a reference, and the threshold is set to allow a small margin (5%) of degradation from this reference value. This adaptive calibration naturally accommodates the varying loss scales across different tasks.

Multi-Constraint Extension. In our implementation, we extend the single-constraint formulation presented in Section 5 to a multi-constraint setting for improved practical performance. Specifically, when task labels are stored alongside samples in the memory buffer, we compute separate gradients and constraints for each past task $t \in \{0, 1, \ldots, T-1\}$. This leads to a multi-constraint quadratic program:

$$v_k = \arg\min_{v \in \mathbb{R}^d} \left\{ \frac{1}{2} \|\nabla \ell_c(\theta_k) - v\|^2 \right\} \quad \text{s.t.} \quad \langle \nabla \ell_t(\theta_k), v \rangle \geq \phi_{h,t}(\theta_k), \quad \forall t < T_{\text{current}} \tag{20}$$

where $\phi_{h,t}(\theta_k) = \alpha \cdot h_t(\theta_k)$ is the per-task dynamic barrier. This multi-constraint QP is solved efficiently using standard quadratic programming solvers. The solution takes the form $v_k = \nabla \ell_c(\theta_k) + \sum_t \lambda_{k,t} \nabla \ell_t(\theta_k)$, where $\lambda_{k,t} \geq 0$ are the Lagrange multipliers obtained from the QP solution. This extension preserves the theoretical properties discussed in Appendix A.2 while providing finer-grained control over per-task forgetting.

Following [Wu et al., 2024b, Gupta et al., 2020, Wu et al., 2024a], we adapt a look-ahead gradient computation strategy to better assess the effect of updates on memory performance. Specifically, we perform an inner-loop gradient step on task-specific data before computing the outer-loop gradient used for the constraint evaluation, which provides more accurate gradient information for the constraint satisfaction mechanism, particularly in the early stages of learning each task. Since the dynamic threshold mechanism requires a stable reference loss for calibration, we apply a brief warm-up phase using standard experience replay for the first 5 update steps when learning each new task. This allows the memory buffer to accumulate sufficient samples and establishes stable gradient estimates before the full GEC constraint mechanism is activated.

To ensure numerical stability, gradients are clipped before storage and computation. This prevents extreme gradient values from destabilizing the QP solution. All experiments are conducted over 5 independent runs with different random seeds, and we report the mean and standard deviation of evaluation metrics. The experiments were performed on 8 NVIDIA A100 GPUs.

E.3 Baseline Method Descriptions

We compare GEC with a comprehensive set of baseline methods. Key rehearsal-based and constraint-based methods include:

E.3.1 Foundational and Regularization Methods

- **SGD**: Standard Stochastic Gradient Descent training on current task data without any specific mechanism to mitigate forgetting. This serves as a lower-bound baseline.
- **Elastic Weight Consolidation (EWC)** [Kirkpatrick et al., 2017, Huszár, 2018]: EWC is a regularization-based method that penalizes changes to parameters deemed important for previously learned tasks. Importance is estimated using the diagonal of the Fisher Information Matrix.

E.3.2 Replay-based Methods

- **Experience Replay (ER)** [Rolnick et al., 2019]: ER trains on a combination of current task data and data sampled from a memory buffer that stores examples from previous tasks. The model is typically trained on a composite loss, often a weighted sum of the current task loss and the loss on the replayed memory samples.
- **Dark Experience Replay (DER & DER++)** [Buzzega et al., 2020]: DER and its variant DER++ enhance ER by additionally replaying network logits (outputs before the softmax layer) from past tasks. This is usually achieved by incorporating a knowledge distillation loss based on the stored logits, aiming to better preserve the previous model's predictions on past data.

- **CLSER** [Arani et al., 2022]: CLSER proposes a dual memory experience replay (ER) method which maintains short-term and long-term semantic memories that interact with the episodic memory.
- **Refresh CL** [Wang et al., 2024]: Refresh CL proposes a general framework for CL and introduces a refresh learning mechanism (unlearn-relearn) inspired by neuroscience. It aims to shed outdated information to improve retention of crucial knowledge and facilitate new learning, acting as a plug-in for existing CL methods.

E.3.3 Constraint-based Methods

- **Gradient Episodic Memory (GEM)** [Lopez-Paz and Ranzato, 2017]: GEM uses samples from the memory buffer to constrain the gradient updates of the current task. It aims to prevent any increase in the average loss on memory samples by projecting the current task's gradient if it conflicts with the objective of not increasing memory loss (i.e., if the dot product between the current task gradient and memory gradient is negative).
- **Averaged GEM (A-GEM)** [Chaudhry et al., 2018]: A-GEM is a more computationally efficient variant of GEM. It computes a single constraint based on the average loss over all samples in the memory buffer, rather than per-task constraints, and projects the current task gradient if it would increase this average memory loss.

E.3.4 Online Continual Learning Methods

- **Continual Bias Adaptor (CBA)** [Wang et al., 2023]: CBA introduces a module to adapt to distribution shifts during training in online CL. It augments the classifier to handle catastrophic distribution changes and can be removed during testing, incurring no extra computational cost at inference.
- **Pareto Optimized Continual Learning (POCL)** [Wu et al., 2024b]: POCL explicitly treats OCL as a multi-objective optimization problem, aiming to find solutions on the Pareto front that balance plasticity and stability. It often involves techniques to steer the optimization towards desired trade-off regions.

F Limitations

While the GEC method demonstrates strong performance in OCL benchmarks, it has certain limitations. One primary consideration is the computational overhead introduced by solving a QP problem at each training step to determine the update direction v_k. Although the QP is relatively small and has a closed-form solution as shown in Eq. (13), this still involves additional computations compared to simpler methods like ER. Future work will focus on investigating more computationally efficient approximations to achieve dynamic constraint-guided update with reduced overhead, making GEC even more suitable for resource-constrained online learning scenarios.

G Broader Impact

The proposed GEC method contributes to the field of Online Continual Learning, aiming to develop AI systems that can learn sequentially and adapt to evolving data streams more effectively. Success in this area has significant positive implications for real-world applications where data is inherently dynamic and non-stationary. For instance, GEC could enhance the capabilities of autonomous systems (e.g., robotics, self-driving cars) to learn from new experiences in real-time without forgetting previously learned skills. In personalized learning or recommendation systems, it could allow models to adapt to changing user preferences over time. In healthcare, it could enable diagnostic models to incorporate new medical knowledge or patient data continuously. By improving the stability-plasticity trade-off, GEC can lead to more robust, reliable, and adaptable AI. While the direct societal risks of this specific algorithmic improvement are low, as with any advancement in AI, responsible development and deployment are crucial. The goal of this research is to advance AI's learning capabilities, which can be a powerful tool for societal benefit when applied thoughtfully and ethically.

NeurIPS 2025 Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly articulate the core idea of the GEC method and its advantages in OCL.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussion of computational efficiency and strong assumptions of GEC methods are in Appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper presents Propositions 1 and 2 in Section 2, stating that proofs are provided in Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 5.1 details the experimental setup, including datasets, evaluation metrics, backbone network, optimizer and other Key hyperparameters for GEC implementaion.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have provided instructions in the paper on data access and we will release the code implementation.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please kindly refer to Section 6.1.

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results include values with $\pm$, implying statistics over multiple runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please kindly refer to Section 6.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research does not involve human subjects, does not directly handle sensitive personal data (uses public, processed datasets), and has no apparent goal of causing direct societal harm (e.g., weapon development, discriminatory applications).

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have include the discussion of broader impacts in Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The proposed model do not fall into the category of pre-trained language models or image generators with high misuse risk, thus not requiring specific safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have mentioned the datasets used (Seq-CIFAR10, Seq-CIFAR100, Seq-TinyImageNet) and the backbone network (Reduced ResNet-18), referencing relevant works.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The main contribution of the paper is a new algorithm.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The research presented in the paper does not involve human subjects and therefore does not require IRB approval or equivalent review.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs are not a component of our core methodology.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.