

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 lower-case 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., $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^k$). 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):

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

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

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

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):

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

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:

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

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

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

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 $\varepsilon_M^*(k) = \bar{\ell}_M(f(X_M; \theta^*), Y_M) - \bar{\ell}_M(f(X_M; \theta_{k-1}), Y_M)$. \square

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):

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

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)). \square

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:

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

This can be written as $\frac{d}{dt} P_{\mu}(\theta_t) \leq -K_{\mu-\lambda_t}(\theta_t, \lambda_t)$, where $K_{\nu}(\theta_t, \lambda_t) = \|v_t\|^2 + \nu [\phi_h(\theta_t)]_+ + \lambda_t [-\phi_h(\theta_t)]_+$ 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:

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

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

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

And from the RHS of Eq. (18):

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

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. \square

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, $P_0(\theta_t) = \ell_c(\theta_t)$.

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

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. \square

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:

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

Here, $k \geq 2$ is the number of objective functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$. The vector $\mathbf{x} = (x_1, x_2, \dots, x_n)^\top$ 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 $i = 1, \dots, k$ 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 $i = 1, \dots, k$ 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 $\mathbf{z}^* = (z_1^*, \dots, z_k^*)^\top$ is formed by components z_i^* , where each z_i^* is the minimum value of the i -th objective function $f_i(\mathbf{x})$ considered independently over the feasible region S , i.e., $z_i^* = \min_{\mathbf{x} \in S} f_i(\mathbf{x})$ for $i = 1, \dots, k$. The ideal objective vector is generally not attainable simultaneously for all objectives.

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

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

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}^{**} (or ideal \mathbf{z}^*). The problem is:

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

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

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

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 :

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

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 \pm 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)	
	$ \mathcal{M} = 0.6k$	$ \mathcal{M} = 1k$	$ \mathcal{M} = 1k$	$ \mathcal{M} = 5k$	$ \mathcal{M} = 2k$	$ \mathcal{M} = 5k$
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\pm2.20	69.42\pm1.70	29.63\pm0.90	38.11\pm1.10	22.45\pm1.40	27.66\pm1.80

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 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\pm0.16	11.89\pm0.31

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} = 0.6k$	$ \mathcal{M} = 1k$	$ \mathcal{M} = 1k$	$ \mathcal{M} = 5k$	$ \mathcal{M} = 2k$	$ \mathcal{M} = 5k$
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\pm0.80	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\pm1.80	59.12\pm2.00	17.78\pm0.70	34.22\pm1.80	12.77 \pm 1.20	20.25\pm1.50

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

Symbol	Description
θ	Model parameters.
η	Learning rate.
α	Barrier strength parameter for GEC.
$\bar{\epsilon}$	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, $\text{AvgLoss}(\theta; X_m, Y_m)$.
$\nabla \bar{\ell}_m(\theta)$	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 .

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 .

```
1: Initialize model parameters  $\theta$ .
2: Initialize memory buffer  $M \leftarrow \emptyset$ .
3: Initialize reference memory loss  $\bar{\ell}_m^{\text{ref}} \leftarrow 0$ .
4: for each task  $t = 1, 2, \dots, T$  do
5:   for each incoming batch  $(X_c, Y_c)$  from task  $t$  (denote current step as  $k$ ) do
6:     Compute current task gradient  $\nabla \ell_c(\theta) \leftarrow \nabla_{\theta} \ell(\theta; X_c, Y_c)$ .
7:     if  $M$  is not empty then
8:       Sample memory batch  $(X_m, Y_m)$  from  $M$ .
9:       Compute current average memory loss  $\bar{\ell}_m(\theta) \leftarrow \text{AvgLoss}(\theta; X_m, Y_m)$ .
10:      Compute average memory gradient  $\nabla \bar{\ell}_m(\theta) \leftarrow \nabla_{\theta} \bar{\ell}_m(\theta)$ .
11:      Compute constraint violation  $h_k \leftarrow \bar{\ell}_m(\theta) - \bar{\ell}_m^{\text{ref}} - \bar{\varepsilon}$ . ▷ Refers to Eq. (7)
12:      Compute dynamic barrier  $\phi_{h,k} \leftarrow \alpha \cdot h_k$ . ▷ Refers to Eq. (9)
13:      Compute coefficient  $\lambda_k \leftarrow \max \left( \frac{\phi_{h,k} - \langle \nabla \ell_c(\theta), \nabla \bar{\ell}_m(\theta) \rangle}{\|\nabla \bar{\ell}_m(\theta)\|^2 + \delta}, 0 \right)$ . ▷ Refers to Eq. (11)
14:      Compute update direction  $v_k \leftarrow \nabla \ell_c(\theta) + \lambda_k \nabla \bar{\ell}_m(\theta)$ . ▷ Refers to Eq. (10)
15:      Update reference memory loss for next step:  $\bar{\ell}_m^{\text{ref}} \leftarrow \bar{\ell}_m(\theta)$ .
16:     else
17:        $v_k \leftarrow \nabla \ell_c(\theta)$ .
18:        $\bar{\ell}_m^{\text{ref}} \leftarrow 0$ . ▷ Reset if memory was empty, for the first constrained step
19:     end if
20:     Update parameters  $\theta \leftarrow \theta - \eta v_k$ .
21:     Add samples from  $(X_c, Y_c)$  to memory  $M$  (e.g., using reservoir sampling).
22:   end for
23: end for
24: return  $\theta$ .
```

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. For our GEC method, the key hyperparameters α (barrier strength parameter), $\bar{\varepsilon}$ (slack tolerance), and δ (stability constant in Eq. (13)) were selected based on preliminary experiments. For α and $\bar{\varepsilon}$, we use 0.5 and 0.05 for GEC implementation. 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 a server equipped with 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., 2024]: 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.