Efficient Memory Crystallization for Graph Learning under Non-Stationary Distribution Shifts

Yue Hou[1,2], **Ruomei Liu**[1], **Yingke Su**[2], **Junran Wu**[1,*], **Ke Xu**[1],
[1]State Key Laboratory of Complex & Critical Software Environment, Beihang University
[2]Shen Yuan Honors College, Beihang University
{hou_yue, rmliu, suyingke, wu_junran, kexu}@buaa.edu.cn

Abstract

Deep graph learning models deployed in real-world systems often need to cope with non-stationary environments, where the underlying graph distribution drifts continually over time. Prevailing solutions rely on training auxiliary generative modules to synthesize memory graphs for cross-domain adaptation, which incurs substantial computational overhead and scales poorly under prolonged distribution shifts. We argue that a more economical path exists: rather than generating memory, one can crystallize it. To this end, we propose Efficient Memory Crystallization (EMC), a training-free test-time framework that distills each incoming graph domain into a compact, semantically faithful memory through a closed-form solution to a memory-oriented distribution-matching objective, thereby eliminating redundant domain information under continual covariate shifts. To preserve both generalizability and adaptability as the model traverses a long sequence of target domains, EMC further models inter-domain dependencies through state-evolving memories and admits a theoretically grounded, tighter generalization error bound than direct adaptation. Extensive experiments demonstrate the superior performance of EMC over state-of-the-art baselines on graphs under non-stationary distribution shifts, while reducing average runtime by 87.4% and GPU memory consumption by 92.4% relative to the recent competitor, making continual graph adaptation practical at scale.

1 Introduction

Graph-structured data is ubiquitous in real-world applications such as social network analysis, recommender systems, and epidemiological modeling. To effectively learn from such data, Graph Neural Networks (GNNs) [19, 46] have emerged as powerful models that integrate node features and structural information through neighborhood aggregation. While GNNs have demonstrated impressive performance under the assumption that training and testing data follow the same distribution, this assumption is often violated in practice due to distribution shift, that is, a mismatch between the training (source) and testing (target) distributions which poses a major challenge to the generalization ability of deep models. To bridge this gap, Graph Domain Adaptation (GDA) methods aim to transfer knowledge from the source domain to the target domain by mitigating the distributional discrepancy.

In real-world scenarios, the source data is often inaccessible during test time due to privacy or business constraints, making the adaptation problem more challenging. Moreover, the GNN model frequently operates in non-stationary environments, where the target graph distribution drifts continually over time. Transferring knowledge from the source domain to entirely unseen target domains can introduce inductive bias, which exacerbates catastrophic forgetting and degrades the model's generalization ability to future domains, especially when distribution shifts arrive in a continual stream rather than a

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

single jump, as illustrated in Figure 1. This necessitates addressing the problem of *graph learning under non-stationary distribution shifts at test time*.

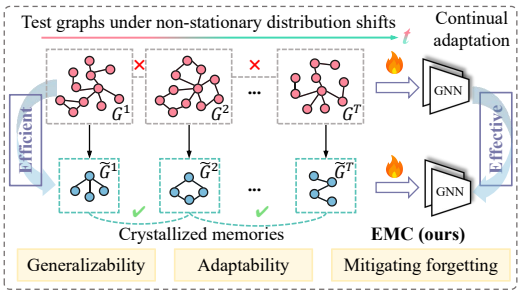


Figure 1: A general paradigm of test-time adaptation under non-stationary distribution shifts. Compared with the first row, which directly adapts the model on the test data without modeling **inter-domain relations** or **intra-domain redundancy**, the second row illustrates our EMC, which crystallizes each domain into a compact memory and explicitly considers these factors to maintain both generalizability and adaptability, thereby ensuring efficiency and effectiveness.

Recently, there has been a growing research trend focusing on single-step domain adaptation, typically employing techniques such as Maximum Mean Discrepancy [8] and adversarial learning [5, 32, 47]. However, these approaches inevitably fall short when the model is exposed to distribution shifts that arrive continually across time or domains. As graph data evolves, models encounter new domains that require ongoing adaptation, while simultaneously losing the ability to perform well on previously seen graphs, which leads to catastrophic forgetting. To mitigate catastrophic forgetting, Continual Learning [37, 38, 48–50] has emerged as a promising direction. These methods typically adopt replay mechanisms, where the model periodically revisits selected or generated historical data to reinforce prior knowledge. Such strategies help balance training cost with long-term knowledge retention. However, most existing approaches rely on labeled data to guide memory selection and replay, making them ill-suited for the test-time setting where labels are unavailable. Although GCAL [31] attempts to improve cross-domain adaptation by training an auxiliary generative module to synthesize memory graphs, its dependence on a separately trained generator introduces substantial computational overhead and limits its practicality under prolonged drift. Furthermore, as shown in Figure 1, current methods under test-time settings with continual drift often ignore the inter-domain relationships and the redundancy within each domain that is irrelevant to the underlying semantic information. We summarize that a desirable framework for adapting graph models under non-stationary distribution shifts at test time should pursue three core objectives: ensuring the model's **generalizability**, maintaining its **adaptability** to novel domains, and **mitigating forgetting** throughout the continual adaptation process, all in a memory- and computation-efficient manner.

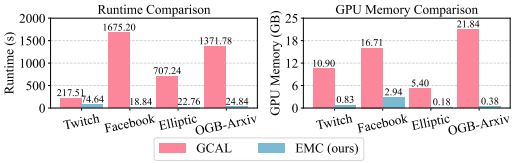


Figure 2: Runtime and GPU memory cost comparison between baseline GCAL and EMC.

In this paper, we argue that a more economical path exists: rather than *generating* memory through a separately trained module, one can *crystallize* it directly from each incoming graph in closed form. We instantiate this idea as **EMC** (**E**fficient **M**emory **C**rystallization), a test-time framework for graph learning under non-stationary distribution shifts. Specifically, we introduce a memory- and computation-efficient procedure that crystallizes each incoming graph domain into a compact, semantically faithful memory through a closed-form solution to a memory-oriented distribution-matching objective, eliminating redundant domain information under continual covariate shifts. To preserve both generalizability and adaptability as the model traverses a long sequence of target domains, we further model inter-domain dependencies through state-evolving memories. We theoretically establish that the generalization error bounds with crystallized memories (from source to target domain) and with state-evolving memories (during continual adaptation on target domains) are both tighter than directly adapting the model on the target domain without such mechanisms, ensuring effectiveness. Extensive experiments on real-world datasets demonstrate the superiority of EMC over state-of-the-art (SOTA) baselines. A comparison of runtime and GPU memory cost between EMC and the recent baseline GCAL [31] across four benchmarks under continual drift is shown in Figure 2. Notably, EMC achieves substantial efficiency gains, requiring on average **87.4% less runtime** and **92.4% less GPU memory** across all datasets. The contributions of this work are as follows:

- We propose an efficient and effective framework, EMC, for graph learning under non-stationary distribution shifts at test time. To the best of our knowledge, this is the first approach to integrate

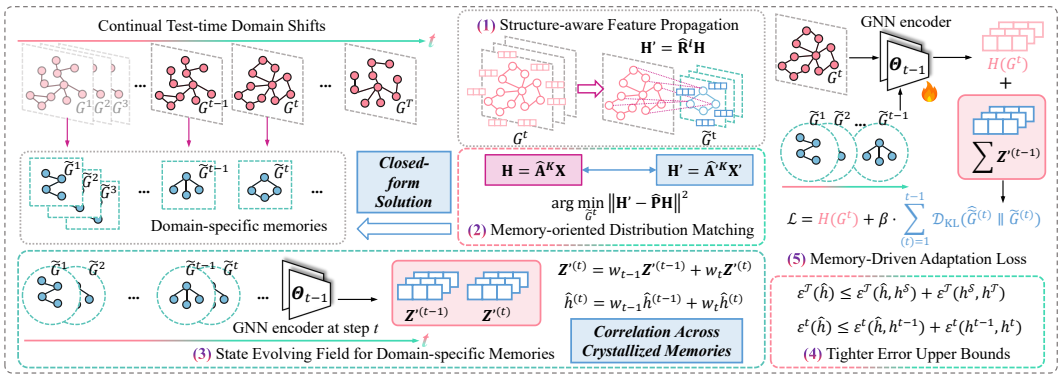


Figure 3: The framework of EMC. (1) Multi-hop feature propagation on the input graph G^t. (2) A closed-form solution to the memory-oriented distribution-matching objective crystallizes the domain-specific memory $\tilde{G}^t$. (3) The state evolving field models the correlation across consecutive domains. (4) Tighter generalization error upper bounds guarantee effectiveness. (5) The adaptation loss updates the parameters $\boldsymbol{\Theta}_{t-1} \to \boldsymbol{\Theta}_t$.

closed-form memory crystallization with state-evolving adaptation without requiring any additional model training at test time.

- We introduce a memory-oriented distribution-matching objective with a closed-form solution for memory construction. We further model inter-domain dependencies through state-evolving memories and derive tighter generalization error bounds.
- Extensive experiments validate the effectiveness and efficiency of EMC, showing superior performance compared to existing SOTA methods, while reducing average runtime by 87.4% and GPU memory consumption by 92.4% relative to the recent competitor.

2 Preliminaries

Problem Setup. We consider the task of test-time adaptation on graphs under non-stationary distribution shifts. Given a GNN model $f(G^{(s)}; \boldsymbol{\Theta}_0) \to \mathcal{Y}$ that is pre-trained on one or multiple source graphs $G^{(s)}$ drawn from the source distribution $\mathbb{P}_{\mathcal{S}}(G, Y)$, where $\boldsymbol{\Theta}_0$ denotes the parameters of the pre-trained GNN encoder and $\{Y\} \in \mathcal{Y}$ is the label set. The target data is represented as a sequence of graphs $\mathcal{T} = \{G^t\}_{t=1}^{T}$ containing T test samples drawn from the marginalized target distribution. Each graph $G^{(t)} = \{\mathbf{A}^{(t)}, \mathbf{X}^{(t)}\} \in \mathbb{P}_{\mathcal{T}}(G)$ corresponds to the t-th target domain, where $\mathbf{A}^{(t)} \in \mathbb{R}^{N_t \times N_t}$ is the adjacency matrix with N_t nodes, and $\mathbf{X}^{(t)} \in \mathbb{R}^{N_t \times d}$ is the node feature with d-dimensional attributes. We use G^t to specifically denote the graph from the t-th target domain in the arrival order, and $G^{(t)}$ as a generic notation when the index (t) may refer to any past or current domain in $\{1, \ldots, T\}$. With the covariate shift assumption, i.e., $\mathbb{P}_{\mathcal{S}}(G) \neq \mathbb{P}_{\mathcal{T}}(G)$ and $\mathbb{P}_{\mathcal{S}}(Y|G) = \mathbb{P}_{\mathcal{T}}(Y|G)$, we aim to continually adapt the model parameters from $\boldsymbol{\Theta}_{t-1}$ to $\boldsymbol{\Theta}_t$ to maintain the generalizability and adaptability[2] on target samples from $\mathbb{P}_{\mathcal{T}}(G)$, while mitigating the risk of catastrophic forgetting on previously encountered domains. We summarize the frequently used notations in Appendix A.

3 Methodology

In this section, we elaborate on the proposed efficient test-time framework EMC with its overall pipeline shown in Figure 3. To efficiently extract low-dimensional representations with reduced redundancy within each domain, we first reformulate a memory-oriented distribution-matching objective and derive a closed-form solution that *crystallizes* each incoming domain into a compact, semantically faithful memory. To enable continual adaptation by capturing inter-domain relationships, we then propose state-evolving memories, which yield a tighter generalization error bound.

3.1 Efficient Memory Crystallization

The original input graphs exhibit covariate shifts across domains and redundancies within each sample, which are unrelated to the underlying semantic information. To retain only the representative

[2]A detailed discussion on the concepts and distinctions between generalizability and adaptability is in Appendix D.3.

characteristics of each domain, we crystallize a domain-specific memory as a low-rank surrogate graph that approximates the original graph in the embedding space.

Structure-aware Feature Propagation. Given an input graph $G^{(t)} = \{\mathbf{A}, \mathbf{X}\}$, the non-parametric feature propagation module is deployed to smooth the node features at the K-th order as $\mathbf{H} = \hat{\mathbf{A}}^K\mathbf{X}$, where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\frac{1}{2}}$ is the normalized adjacency matrix with self-loops, and $\tilde{\mathbf{D}}$ is the degree matrix for $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. This propagation scheme can be replaced with other alternatives, such as SGC [39] and personalized PageRank (PPR) [29]. Higher-order propagation produces increasingly smooth features that capture more global structural information.

To preserve semantic structure, we apply class-wise node partitioning, which divides the $|C_i|$ nodes assigned to class i into $|C_i'|$ non-overlapping sub-classes $\{S_1^i, ..., S_{|C_i'|}^i\}$. At test time, the class assignment is guided by the model's predicted semantics, which serve as *semantic anchors* for grouping nodes with similar semantic content rather than as supervisory targets. Each node k further carries a confidence score r_k that down-weights uncertain contributions during aggregation; a detailed discussion of this design is deferred to Appendix B.3. The node mapping function of this partition is defined as $\pi^i : \{1, \ldots, |C_i|\} \to \{1, \ldots, |C_i'|\}$, and the class-wise aggregation matrix $\mathbf{R}^i \in \mathbb{R}^{|C_i'| \times |C_i|}$ is formulated as

$$\mathbf{R}^i_{j,k} = \begin{cases} r_k/\tau & \text{if } \pi^i(k) = j, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where τ is a temperature parameter. Aggregated embeddings for class i are obtained as $\mathbf{H}_i' = \hat{\mathbf{R}}^i\mathbf{H}_i$, where $\hat{\mathbf{R}}^i$ is the row-normalized version of $\mathbf{R}^i$ and $\mathbf{H}_i$ collects the propagated features of nodes in class i. The full aggregation matrix is constructed by stacking all class-wise aggregation matrices block-diagonally as $\hat{\mathbf{P}} = \text{diag}(\hat{\mathbf{R}}^1, ..., \hat{\mathbf{R}}^C)$, where C is the total number of classes.

Memory-oriented Distribution Matching. We denote the memory crystallized from $G^{(t)}$ as $\tilde{G}^{(t)} = \{\mathbf{A}', \mathbf{X}'\}$. To ensure that the memory faithfully represents the original graph, we reformulate a memory-oriented distribution-matching objective between the representations of the original graph and its memory.

Proposition 3.1. After multi-hop propagation, let $\mathbf{Z} = \mathbf{H}\mathbf{\Psi} = \hat{\mathbf{A}}^K\mathbf{X}\mathbf{\Psi}$ and $\mathbf{Z}' = \mathbf{H}'\mathbf{\Psi} = \hat{\mathbf{A}}'^K\mathbf{X}'\mathbf{\Psi}$ denote the representations of the original graph $G^{(t)}$ and its memory $\tilde{G}^{(t)}$ respectively, where $\mathbf{\Psi}$ is the learnable linear weight of the GNN encoder and Φ is the distribution over its parameter configurations. To directly align the distribution of the input domain with that of its memory, the discrepancy between the propagated features should be minimized as:

$$\min_{\tilde{G}^{(t)}} \mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{P}'\mathbf{H}'\mathbf{\Psi} - \mathbf{P}\mathbf{H}\mathbf{\Psi}\right\|^2. \tag{2}$$

Assuming that nodes in $\tilde{G}^{(t)}$ are organized in ascending order, $\mathbf{P}'$ degenerates into the identity matrix $\mathbf{I}$. Then, the distribution matching objective can be upper-bounded by:

$$\mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{H}'\mathbf{\Psi} - \hat{\mathbf{P}}\mathbf{H}\mathbf{\Psi}\right\|^2 \leq \mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2 \|\mathbf{\Psi}\|^2. \tag{3}$$

Proof for Proposition 3.1 is provided in Appendix C.1. The expectation $\mathbb{E}_{\Psi\sim\Phi}$ is a technical device that yields a bound holding uniformly over the encoder parameter space; it is not a random projection.

Remark. The reformulated objective aligns the essential semantic information between the input graph and its crystallized memory. Since $\mathbf{\Psi}$ is independent of both $\tilde{G}^{(t)}$ and $\hat{\mathbf{P}}$, we can minimize the upper bound by:

$$\arg\min_{\tilde{G}^{(t)}, \hat{\mathbf{P}}} \left\|\mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2. \tag{4}$$

Redundancy-eliminated Memory Construction. According to Proposition 3.1, we expect a symmetric encoding procedure between the original $G^{(t)}$ and its crystallized memory $\tilde{G}^{(t)}$. Therefore, our objective is to construct $\mathbf{A}'$ and $\mathbf{X}'$ satisfying $\hat{\mathbf{A}}'^K\mathbf{X}' = \mathbf{H}'$. To this end, we utilize the pre-defined graph structure and obtain $\mathbf{X}'$ in closed form. Specifically, we construct the structure via a similarity-based thresholding scheme:

$$\mathbf{A}'_{i,j} = \begin{cases} 1 & \text{if } \cos(\mathbf{H}'_i, \mathbf{H}'_j) > \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\cos(\cdot,\cdot)$ measures the cosine similarity and τ is the hyper-parameter for graph sparsification. To ensure that the reconstructed features change smoothly between connected nodes, we introduce the Dirichlet energy constraint [18] into the feature reconstruction loss to quantify the smoothness of graph signals. The optimization objective for $\mathbf{X}'$ is then formulated as:

$$\arg\min_{\mathbf{X}'} \left\| \hat{\mathbf{A}}'^{K}\mathbf{X}' - \mathbf{H}' \right\|^2 + \alpha \operatorname{tr}(\mathbf{X}'^{\top}\mathbf{L}'\mathbf{X}'), \tag{6}$$

where $\mathbf{L}' = \mathbf{D}' - \mathbf{A}'$ is the Laplacian matrix, $\mathbf{D}'$ is the degree matrix, and α controls the smoothness strength.

Proposition 3.2. Assume that $\tilde{G}^{(t)} = \{\mathbf{A}', \mathbf{X}'\}$ is the crystallized memory of an input graph; the closed-form solution of Eq. (6) takes the form $\mathbf{X}' = (\mathbf{Q}^{\top}\mathbf{Q} + \alpha\mathbf{L}')^{-1}\mathbf{Q}^{\top}\mathbf{H}'$, where $\mathbf{Q} = \hat{\mathbf{A}}'^{K}$.

Proof Sketch. Taking the first- and second-order partial derivatives of the objective in Eq. (6) with respect to $\mathbf{X}'$ shows that the objective is positive definite. Therefore, the closed-form solution that satisfies the first-order condition achieves the minimum. A detailed proof is provided in Appendix C.2. The closed-form solution operates on a small $N_t' \times N_t'$ system over the crystallized memory rather than the full $N_t \times N_t$ system on the original graph (with N_t' set to 1% of N_t in our experiments), making memory construction lightweight and naturally scalable. Further empirical scaling analysis is provided in Appendix B.2.

3.2 Memory-Driven Adaptation with Tighter Bound

To capture state transitions and evolutionary patterns across domains, we introduce the State Evolving Field mechanism over crystallized memories with derivation of a tighter generalization error bound.

Proposition 3.3. Given any pair of sequences of samples from the target domain, $\mathcal{T} = \{G^t\}_{t=1}^T$ and the corresponding crystallized memories $\tilde{\mathcal{T}} = \{\tilde{G}^t\}_{t=1}^T$, following the joint distributions $\mathbb{P}_{G^{1:T}}$ and $\mathbb{Q}_{\tilde{G}^{1:T}}$ respectively, with each G^t and $\tilde{G}^t$ following the marginal distributions $\mathbb{P}_{G^t}$ and $\mathbb{Q}_{\tilde{G}^t}$. Then the following relation holds:

$$\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^{1:T}} \,\|\, \mathbb{Q}_{\tilde{G}^{1:T}}) \equiv \sum_{t=1}^{T} \mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t} \,\|\, \mathbb{Q}_{\tilde{G}^t}) \quad \text{iff} \quad \mathbb{P}_{G^{1:T}} = \prod_{t=1}^{T} \mathbb{P}_{G^t},\ \mathbb{Q}_{\tilde{G}^{1:T}} = \prod_{t=1}^{T} \mathbb{Q}_{\tilde{G}^t}, \tag{7}$$

where $\mathcal{D}_{\mathrm{KL}}$ denotes the Kullback–Leibler divergence and $\equiv$ indicates that the two expressions are identically equal.

Remark. Proposition 3.3 (proof in Appendix C.3) is a *biconditional*: it does not assume that domains are independent. Rather, it characterizes precisely when summing per-domain KL divergences equals the joint KL divergence, namely when and only when the domains (and their memories) are independent. Since real test-time domains are generally correlated under continual drift, this implies that aligning marginal distributions alone is insufficient. This motivates us to model inter-domain relationships explicitly through the state evolving field below.

State Evolving Field over Crystallized Memories. To model the temporal correlation across domains, we define a state evolving field over crystallized memories. At each time step t, the memory representation $\mathbf{Z}'^{(t)}$ evolves from its predecessor $\mathbf{Z}'^{(t-1)}$ through a convex combination:

$$\mathbf{Z}'^{(t)} \leftarrow w_{t-1}\mathbf{Z}'^{(t-1)} + w_t\mathbf{Z}'^{(t)}, \text{where } w_{t-1} + w_t = 1. \tag{8}$$

Here w_{t-1} and w_t are fixed non-negative hyperparameters satisfying the convex-combination constraint and selected by grid search; an adaptive weighting strategy is a natural extension since Theorem 3.6 below holds for any valid choice. During adaptation, $f(G^t; \mathbf{\Theta}_{t-1}) = h \circ g$ is the composition of an encoder $h(\cdot)$ and a fixed classifier $g(\cdot)$, *i.e.*, $\mathbf{Z}'^{(t)} = h(\tilde{G}^t)$ and $\hat{Y}^t = g(\mathbf{Z}'^{(t)})$. At each step t, the mixed memory $\mathbf{Z}'^{(t)} = w_{t-1}\mathbf{Z}'^{(t-1)} + w_t\mathbf{Z}'^{(t)}$ induces a corresponding hypothesis $\hat{h} = w_{t-1}h^{t-1} + w_t h^t$ in the decision space.

Lemma 3.4 (Error Upper Bound for Domain Shift)**.** Let $\tilde{\mathbb{P}}_{\mathcal{S}}$ and $\tilde{\mathbb{P}}_{\mathcal{T}}$ denote the induced distributions over the feature space for $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{T}}$ respectively. The following inequality holds for the risk $\epsilon_0^{\mathcal{T}}(\hat{h})$ with single-step adaptation on the target domain $\mathbb{P}_{\mathcal{T}}$:

$$\epsilon_0^{\mathcal{T}}(\hat{h}) \le \min\{\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}), \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})\} + \epsilon^{\mathcal{S}}(\hat{h}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}), \tag{9}$$

where $d_{\mathcal{H}}$ is the $\mathcal{H}$-divergence.

Theorem 3.5 (Error Upper Bound with Crystallized Memories)**.** Let $\tilde{\mathbb{P}}_{\mathcal{S}}$ and $\tilde{\mathbb{P}}_{\mathcal{T}}$ denote the induced distributions over the feature space for $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{T}}$ respectively. The risk $\epsilon^{\mathcal{T}}(\hat{h})$ with crystallized memories (with mixed weight $w_0 + w_1 = 1$, $w_0 \geq 0$, $w_1 \geq 0$, and $\hat{h} = w_0\hat{h}^{\mathcal{S}} + w_1\hat{h}^{\mathcal{T}}$) is upper bounded by:

$$\epsilon^{\mathcal{T}}(\hat{h}) \leq \epsilon^{\mathcal{T}}(\hat{h}, h^{\mathcal{S}}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}). \tag{10}$$

Theorem 3.6 (Continual Error Upper Bound with State-Evolving Memories)**.** Let $\tilde{\mathbb{P}}_{G^{t-1}}$ and $\tilde{\mathbb{P}}_{G^t}$ denote the induced feature distributions of two consecutive target domains at steps $t-1$ and t. The risk $\epsilon^t(\hat{h})$ with state-evolving memories (with mixed weight $w_{t-1} + w_t = 1$, $w_{t-1}, w_t \geq 0$, and $\hat{h} = w_{t-1}\hat{h}^{t-1} + w_t\hat{h}^t$) is upper bounded by:

$$\epsilon^t(\hat{h}) \leq \epsilon^t(\hat{h}, h^{t-1}) + \epsilon^t(h^{t-1}, h^t). \tag{11}$$

Remark. The bounds on $\epsilon_0^{\mathcal{T}}(\hat{h})$ characterize per-step generalization at each adaptation step t, while the continual aspect across $\{G^t\}_{t=1}^T$ is handled by the state-evolving design ($\triangleright$ Theorem 3.6) with past memories. At each step, the improvement of Theorem 3.5 over Lemma 3.4 comes from eliminating the explicit source–target $\mathcal{H}$-divergence penalty $d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}})$ that appears in Lemma 3.4, which yields a bound that dominates $\epsilon_0^{\mathcal{T}}(\hat{h})$ by a non-negative margin of $d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) \geq 0$ regardless of the relative ordering of $\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}})$ and $\epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})$. We therefore arrive at the following conclusions:

- From Theorem 3.5, the bound on $\epsilon^{\mathcal{T}}(\hat{h})$ obtained with crystallized memories admits a tighter form than the direct source-to-target bound $\epsilon_0^{\mathcal{T}}(\hat{h})$.
-
- By Theorem 3.6, when EMC traverses consecutive domains via state-evolving memories, the per-step bound likewise admits a tighter form than $\epsilon_0^{\mathcal{T}}(\hat{h})$ at each step.

Detailed proofs and comparisons are in Appendix C.3.

Derivation of the Adaptation Loss. Given a test graph from the t-th target domain and the sequence of memories from the previous $t-1$ domains, $\tilde{\mathcal{T}} = \{\tilde{G}^1, \tilde{G}^2, \ldots, \tilde{G}^{t-1}\}$, the GNN model with parameters $\mathbf{\Theta}_{t-1}$ is continually refined to adapt to the new graph while preserving performance on previously seen graphs. The overall optimization objective of EMC is:

$$\mathcal{L} \doteq H(G^t) + \beta \cdot \mathcal{L}_{\text{Adp}}(\tilde{G}^{1:t-1}; \mathbf{\Theta}_{t-1}), \text{ and } \mathcal{L}_{\text{Adp}}(\tilde{G}^{1:t-1}; \mathbf{\Theta}_{t-1}) = \sum_{(t)=1}^{t-1} \mathcal{D}_{\text{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)}), \tag{12}$$

where $H(\cdot)$ denotes the Shannon entropy that measures the uncertainty within the t-th domain, β is a hyperparameter, and $\mathcal{D}_{\text{KL}}(\cdot\|\cdot)$ is the KL divergence between two distributions. Here $\tilde{G}^{(t)}$ denotes the stored memory of the (t)-th domain (frozen anchor), and $\hat{\tilde{G}}^{(t)}$ denotes the predictive distribution induced by the current model with parameters $\mathbf{\Theta}_{t-1}$ evaluated on the same memory $\tilde{G}^{(t)}$. At each step the system first crystallizes $\tilde{G}^t$ and forms the mixed representation, and only then minimizes Eq.(12) to update $\mathbf{\Theta}_{t-1} \to \mathbf{\Theta}_t$. The full pipeline and complexity analysis are provided in Appendix B.

Proposition 3.7 (Stability Bound of the Adaptation Loss)**.** Consider the adaptation step from $\mathbf{\Theta}_{t-1}$ to $\mathbf{\Theta}_t$. For each previously encountered domain indexed by $(t) \in \{1, \ldots, t-1\}$, $\mathcal{D}_{\text{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)})$ can be interpreted as the divergence between the predictive distributions induced by the model before and after adaptation on the same crystallized memory $\tilde{G}^{(t)}$. Moreover, assume that: (i) the predictive distribution is given by a softmax with temperature $\tau > 0$; (ii) all predicted class probabilities are lower-bounded by $\gamma > 0$; and (iii) the encoder $h(\cdot)$ and the classifier $g(\cdot)$ are Lipschitz continuous in the model parameters. Then the following upper bound holds:

$$\mathcal{D}_{\text{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)}) \;\leq\; \frac{L_g^2 L_h^2}{4\,\gamma\,\tau^2} \left\| \mathbf{\Theta}_t - \mathbf{\Theta}_{t-1} \right\|_2^2, \tag{13}$$

where L_h and L_g are the corresponding Lipschitz constants of $h(\cdot)$ and $g(\cdot)$.

Remark. The bound asserts that small parameter updates yield small predictive shifts on stored memories, providing a theoretical justification for the KL regularization in Eq. (12) as a means to mitigate forgetting. Assumption (iii) is mild and is satisfied by standard GNN backbones under common weight regularization. Detailed discussion together with proof is provided in Appendix C.5.

Table 1: Performance comparison (%, mean ± standard deviation for five runs) on different backbones with comprehensive metrics, Runtime (s) and GPU Memory Usage (MiB). We compare EMC with the most recent 2 baselines from two categories of domain adaptation methods. '↑' indicates higher is better, and '↓' indicates lower is better. 'OOM' implies Out-Of-Memory. The best results are highlighted with **bold** and the runner-ups are underlined.

Dataset	Methods	Backbone: GCN						Backbone: SAGE					
		AP (↑)	AF (↑)	TDG (↑)	TDA (↑)	Time (↓)	GPU (↓)	AP (↑)	AF (↑)	TDG (↑)	TDA (↑)	Time (↓)	GPU (↓)
Twitch	LEBED	51.06±0.05	0.24±0.05	51.87±0.06	48.90±0.04	114.97	1440.57	56.37±0.03	0.58±0.04	57.55±0.02	55.08±0.04	252.80	3652.58
	GCAL	55.65±0.09	0.42±0.13	54.78±0.60	53.20±0.62	217.51	11159.8	58.20±0.35	0.79±0.13	59.65±0.39	57.02±0.49	295.11	8580.63
	EMC	**55.95±0.15**	**0.47±0.08**	**55.56±0.13**	**55.83±0.14**	**74.64**	**854.97**	**61.62±0.12**	**0.86±0.04**	**62.02±0.13**	**60.36±0.16**	**138.02**	**2833.18**
Facebook	LEBED	46.90±0.00	-0.25±0.00	47.72±0.00	46.38±0.00	93.59	3481.53	46.92±0.00	-2.20±0.01	48.78±0.00	48.39±0.01	94.81	3718.50
	GCAL	52.72±0.36	0.72±0.19	49.37±0.28	48.96±0.29	1675.2	17107.84	51.87±0.12	0.04±0.11	52.34±0.17	51.55±0.08	1356.07	13633.79
	EMC	**53.10±0.00**	**2.15±0.27**	**53.49±0.27**	**52.30±0.04**	**18.84**	**3006.08**	**53.41±0.07**	**0.63±0.20**	**53.11±0.10**	**52.87±0.25**	**18.61**	**3137.31**
Elliptic	LEBED	49.44±0.19	-4.86±0.20	52.10±0.06	53.82±0.02	114.01	745.36	62.12±0.11	-6.65±0.15	69.30±0.09	67.99±0.08	142.57	1004.31
	GCAL	56.57±0.14	0.88±0.13	53.08±2.54	50.31±2.98	707.24	5533.99	71.35±8.57	-3.40±1.10	63.45±8.82	61.38±9.82	453.87	972.59
	EMC	**56.76±0.46**	**0.96±0.30**	**58.73±0.40**	**55.92±0.31**	**22.76**	**189.36**	**72.28±1.92**	**1.53±1.29**	**74.55±2.50**	**70.22±1.46**	**21.67**	**208.91**
Arxiv	LEBED	19.09±0.04	-0.04±0.05	14.18±0.02	17.44±0.02	227.80	839.39	41.81±0.01	-1.85±0.01	47.00±0.01	42.12±0.02	229.83	1082.70
	GCAL	14.15±6.94	-1.80±2.06	17.90±8.58	13.77±5.57	1362.76	22342.47	45.22±0.17	0.76±0.05	42.40±7.74	41.80±6.59	1371.78	22365.84
	EMC	**19.32±1.57**	**3.38±1.99**	**19.39±1.47**	**20.39±1.71**	**24.95**	**299.18**	**46.07±1.70**	**0.99±0.66**	**47.11±1.80**	**43.52±1.25**	**24.84**	**391.02**

Dataset	Methods	Backbone: GAT						Backbone: GIN					
		AP (↑)	AF (↑)	TDG (↑)	TDA (↑)	Time (↓)	GPU (↓)	AP (↑)	AF (↑)	TDG (↑)	TDA (↑)	Time (↓)	GPU (↓)
Twitch	LEBED	53.55±0.10	-0.84±0.05	54.87±0.07	53.26±0.13	164.51	22461.43	50.29±0.10	-4.05±0.27	52.73±0.14	52.33±0.19	131.26	3542.58
	GCAL	OOM	OOM	OOM	OOM	OOM	OOM	54.13±0.25	-2.21±0.31	54.16±0.31	50.00±0.00	316.05	10226.88
	EMC	**60.40±0.45**	**0.68±0.14**	**59.25±0.48**	**56.31±0.51**	**140.98**	**15007.48**	**57.89±0.27**	**0.35±0.28**	**58.05±0.31**	**55.33±0.27**	**101.24**	**2775.17**
Facebook	LEBED	47.47±0.11	-0.65±0.10	48.41±0.08	47.39±0.05	177.97	21358.57	51.82±0.04	0.14±0.07	50.79±0.02	50.83±0.03	102.06	4108.80
	GCAL	OOM	OOM	OOM	OOM	OOM	OOM	48.26±0.63	-0.83±0.52	48.58±1.36	48.72±1.18	1376.18	13614.52
	EMC	**54.28±1.30**	**0.48±0.67**	**54.21±1.01**	**54.59±1.18**	**66.07**	**11258.32**	**52.42±1.54**	**0.26±0.30**	**51.78±1.43**	**51.88±1.73**	**19.76**	**3220.61**
Elliptic	LEBED	54.53±1.62	-6.87±0.76	58.75±2.53	60.50±2.28	470.84	8095.29	63.58±0.07	-2.00±0.10	67.09±0.09	64.04±0.05	194.11	1461.72
	GCAL	63.53±6.81	-2.81±3.15	51.31±7.09	51.71±9.53	626.61	2037.20	42.23±8.55	-5.87±10.35	45.06±1.59	47.03±1.63	589.75	1011.27
	EMC	**70.51±3.18**	**-2.09±1.48**	**68.16±4.75**	**65.56±3.78**	**27.38**	**939.16**	**66.81±2.36**	**-1.13±2.62**	**68.66±3.52**	**64.86±4.68**	**27.57**	**213.36**
Arxiv	LEBED	25.10±0.06	-3.40±0.05	28.43±0.06	25.29±0.04	312.55	9458.82	24.94±0.03	-2.65±0.04	28.70±0.04	25.58±0.03	236.62	1608.92
	GCAL	20.40±0.00	-6.83±1.72	20.70±0.42	21.25±1.72	1481.93	24770.10	24.15±6.94	-1.80±2.06	27.90±8.58	23.77±5.57	1362.76	22342.47
	EMC	**38.69±1.62**	**2.21±0.58**	**43.43±3.14**	**39.44±2.08**	**26.02**	**3441.89**	**38.78±3.05**	**-0.98±1.22**	**41.13±3.47**	**39.14±2.21**	**25.06**	**508.92**

4 Experiments

In this section, we empirically evaluate the effectiveness and efficiency of the proposed EMC.[3] We organize the experiments around the following research questions:

- **RQ1:** How well does EMC achieve generalizability, adaptability, and forgetting mitigation efficiently with low memory and computational cost under continual drift?
- **RQ2:** How does EMC scale with respect to graph size and the number of domains?
- **RQ3:** What are the impacts of different components on the effectiveness and efficiency of EMC?
- **RQ4:** How sensitive is EMC to its hyperparameters?

4.1 Experimental Setup

Datasets. We evaluate EMC on 4 node classification datasets, following the benchmark in GCAL [31], categorized into two types:

- For **regional shifts**, the Facebook-100 [35] and Twitch-Explicit [33] datasets consist of multiple social network graphs collected from different regions.
- For **temporal shifts**, OGB-Arxiv [16] is a citation network of academic papers, and Elliptic [30] is a Bitcoin transaction network, with both containing graphs spanning different time steps.

In all cases, each graph is treated as a separate domain. We pre-train a GNN model on selected source domains and then perform continual adaptation on the remaining domains. Dataset statistics and further details are in Appendix D.1.

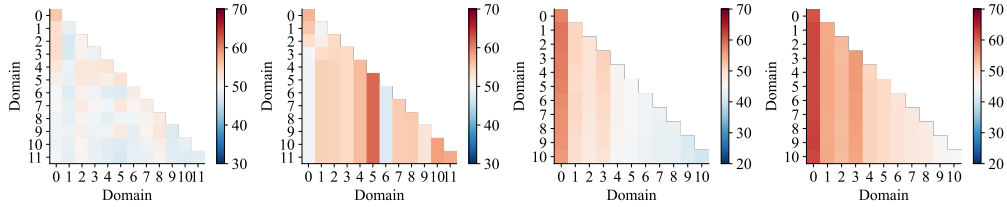
Baselines. We compare EMC against 10 representative baseline methods. "Base Model" refers to the pre-trained model directly applied to target datasets without any adaptation, serving as a lower bound. The baselines are grouped into two categories: (1) *One-step Test-Time Adaptation Methods* [9, 36, 20, 44, 17, 53], and (2) *Continual Test-Time Adaptation Methods* [38, 28, 31].

Evaluation and Implementation. We report the performance matrix $M \in \mathbb{R}^{T\times T}$, a lower triangular matrix where $M_{i,j}$ (for $i \geq j$) denotes the performance on domain j after training on domain i. Following [17, 44], we use ROC-AUC for the Twitch-Explicit dataset, Accuracy for Facebook-100 and OGB-Arxiv, and F1 Score for Elliptic. To summarize performance across all domains, we compute the Average Performance (AP) as $\text{AP} = \frac{1}{T}\sum_{i=1}^{T} M_{T,i}$, which reflects overall adaptation ability, and the Average Forgetting (AF) as $\text{AF} = \frac{1}{T-1}\sum_{i=1}^{T-1}(M_{T,i} - M_{i,i})$, which quantifies forgetting during

[3]The code of EMC is available at: `https://github.com/name-is-what/EMC`.

Table 2: Performance comparison (%, mean ± standard deviation for five runs) across four datasets, evaluated against baseline methods and EMC. The results of baselines are derived from the published works. 'NA' is short for Not Applicable, and 'OOM' implies Out-Of-Memory. The best results are highlighted with **bold** and the runner-ups are underlined.

Methods / Datasets	Twitch-explicit		Facebook-100		Elliptic		OGB-Arxiv	
	AP@AUC (↑)	AF (↑)	AP@ACC (↑)	AF (↑)	AP@F1 (↑)	AF (↑)	AP@ACC (↑)	AF (↑)
Base Model	53.88±0.00	NA	50.55±0.00	NA	53.97±0.00	NA	42.43±0.00	NA
One-step Test-Time Adaptation Methods								
DANN [9]	51.50±0.39	0.19±0.55	50.63±0.64	0.38±1.13	53.84±0.56	-0.95±0.54	42.62±0.23	-1.28±0.16
Norm [20]	52.65±0.00	-2.30±0.00	47.62±0.00	0.53±0.00	54.67±0.00	0.02±0.00	40.21±0.00	0.11±0.00
TENT [36]	52.84±0.98	-1.83±1.05	46.63±0.00	0.61±0.00	46.54±0.00	0.55±0.00	36.72±0.37	-0.02±0.18
EERM [44]	52.08±0.00	NA	49.70±0.03	NA	46.51±0.00	NA	36.26±0.00	NA
GTrans [17]	53.55±0.29	NA	OOM	NA	54.25±0.02	NA	39.69±0.04	NA
LEBED [53]	51.06±0.05	0.24±0.05	46.90±0.00	-0.25±0.00	49.44±0.19	-4.86±0.20	41.81±0.01	-1.85±0.01
Continual Test-Time Adaptation Methods								
CoTTA [38]	53.94±0.36	0.34±0.41	50.12±0.16	0.59±0.12	54.08±0.05	-1.92±0.06	40.28±0.01	-1.96±0.01
EATA [28]	53.56±0.10	-0.72±0.07	49.02±3.16	-0.57±0.03	50.48±0.07	-0.76±0.06	40.91±0.70	-1.35±0.59
GCAL [31]	55.65±0.09	0.42±0.13	52.72±0.36	0.72±0.19	56.57±0.14	0.88±0.13	45.22±0.17	0.76±0.05
STEM [25]	**56.66±0.05**	-0.07±0.17	NA	NA	NA	NA	31.71±0.14	-4.40±0.19
EMC (ours)	55.95±0.15	**0.47±0.08**	**53.10±0.00**	**2.15±0.27**	**56.76±0.46**	**0.96±0.30**	**46.07±1.70**	**0.99±0.66**



(a) GCAL on Facebook-100 (b) EMC on Facebook-100 (c) GCAL on OGB-Arxiv (d) EMC on OGB-Arxiv

Figure 4: Performance matrices of EMC and GCAL on different datasets.

sequential training. To further assess generalizability and adaptability, we follow [21] to introduce 2 complementary metrics: Target Domain Generalization (TDG) measures the model's performance on each domain before adaptation as TDG $= \frac{1}{T-1}\sum_{i=2}^{T} M_{i-1,i}$. Target Domain Adaptation (TDA) measures performance after adaptation as TDA $= \frac{1}{T}\sum_{i=1}^{T} M_{i,i}$. All reported results are averaged over 5 runs with standard deviations. We perform a grid search to select the key hyperparameters of EMC. More detailed experimental settings are provided in Appendix D.3.

4.2 Overall Performance Comparison (RQ1)

Generalizability and Adaptability Analysis. To answer RQ1, we evaluate the generalizability and adaptability of EMC using the comprehensive metrics in Table 1. We compare EMC with two recent baselines representing different categories of domain adaptation methods. In addition to AP and AF, EMC consistently maintains strong generalizability and adaptability under continual drift. LEBED, not specifically designed for continual settings, shows weaker generalization. Although GCAL is tailored for continual adaptation, it incurs substantial computational overhead. Notably, on the Elliptic dataset with GCN as the backbone, EMC reduces runtime by 96.78% and GPU memory by 96.58%. On OGB-Arxiv with SAGE, it reduces runtime by 98.19% and GPU memory by 98.25%. The results confirm EMC's superior effectiveness and efficiency for graph learning under non-stationary distribution shifts at test time, ensuring generalizability and adaptability while significantly reducing computational overhead.

Extended Comparisons. We further compare EMC with 10 representative baselines following the GCAL setting (▷ Table 2 and Appendix D.5), and conduct extended comparison against the very recent baselines GCAL [31] and STEM [25] on both node-level and graph-level adaptation (▷ Appendix D.6). These results confirm that EMC generalizes across task granularities and adaptation protocols, attaining strong effectiveness without sacrificing memory efficiency.

Visualization of the Performance Matrix. We visualize the overall performance during continual adaptation using the performance matrix. As shown in Figure 4, EMC effectively mitigates forgetting

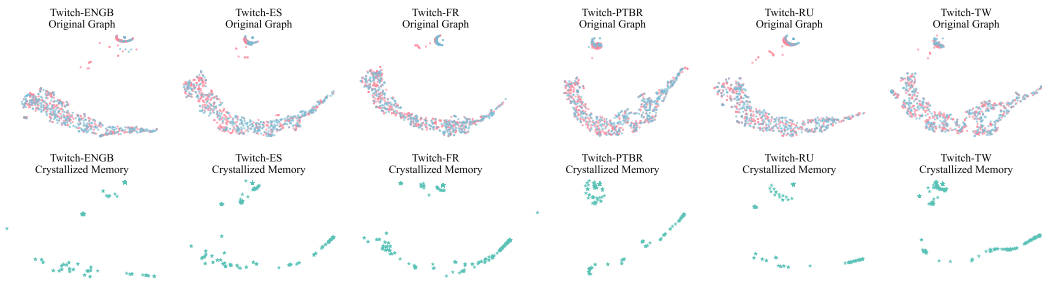


Figure 5: T-SNE visualization of representations of the original graph and its crystallized memory.

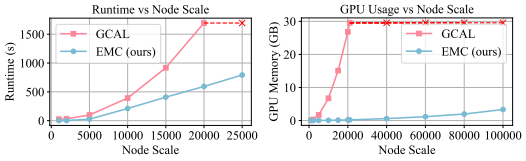


Figure 6: Node scaling efficiency evaluation.

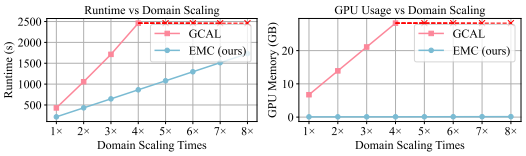


Figure 7: Domain scaling efficiency evaluation.

under continual drift, maintaining strong performance across multiple target domains throughout the adaptation process. Detailed analysis with additional datasets is provided in Appendix D.7.

Visualization of Crystallized Memories. To intuitively examine what the crystallized memories preserve, we provide t-SNE projections of the original and memory embeddings produced by the GNN encoder on the continuous-domain Twitch graphs in Figure 5. The two embedding sets are closely aligned in the representation space, confirming that EMC retains the semantically informative content of each domain in a much more compact form. Additional structural visualizations rendered with NetworkX are provided in Appendix D.8.

4.3 Scaling Efficiency Analysis (RQ2)

To evaluate the efficiency of EMC, we generate synthetic datasets by varying the number of nodes and the number of domains. We plot the runtime and peak GPU memory usage in Figures 6 and 7, where the red cross indicates an Out-Of-Memory (OOM) error. Additional results and detailed settings are provided in Appendix D.11. When the number of nodes exceeds 40k, both GCAL and LEBED are stuck due to OOM, while EMC still operates within GPU limits even at the highest tested scale. We further benchmark the memory crystallization step alone on a single graph up to **200k nodes** (see in Appendix B.2), where the closed-form inversion takes only 0.71s, confirming that the cost on the small $N_t' \times N_t'$ system remains negligible relative to the propagation cost on the original graph.

4.4 Ablation Studies (RQ3)

We conduct ablation studies to analyze the effectiveness of two architectural variants:

- ***w/o DM*:** We remove the Domain-specific Memory crystallization in Section 3.1.
- ***w/o EA*:** We remove the memory-driven adaptation in Section 3.2.

Table 3: Ablation study results of EMC and its variants with runtime (s) and GPU usage (MiB).

Dataset	Method	AP(↑)	TDG(↑)	TDA(↑)	Time(↓)	GPU(↓)
Twitch	*w/o DM*	52.12	53.77	51.35	99.43	1486.81
	w/o EA	51.50	52.31	51.21	60.13	783.14
	EMC	**55.95**	**55.56**	**55.83**	74.64	854.97
Facebook	*w/o DM*	48.78	48.96	48.92	85.45	13481.55
	w/o EA	51.30	50.85	50.34	17.65	2965.72
	EMC	**53.10**	**53.49**	**52.30**	18.84	3006.08
Elliptic	*w/o DM*	47.44	49.20	47.57	97.60	747.93
	w/o EA	47.35	50.55	50.00	20.52	183.57
	EMC	**56.76**	**58.73**	**55.92**	22.76	189.36
Arxiv	*w/o DM*	12.61	16.45	17.58	30.52	1083.73
	w/o EA	20.65	20.50	20.51	13.26	390.12
	EMC	**46.07**	**47.11**	**43.52**	24.84	391.02

Results are shown in Table 3. Overall, EMC outperforms both variants, validating the indispensable contribution of each component. In addition, removing "*DM*" leads to a significant drop in both efficiency and effectiveness, while removing "*EA*" reduces runtime and GPU memory but causes considerable degradation in adaptation performance.

4.5 Parameter Study (RQ4)

Sensitivity Analysis of α. To analyze the sensitivity of α for EMC, we vary it from 0.3 to 5.0. The AP w.r.t. different α values is plotted in Figure 8. The results confirm that EMC is robust to reasonable choices of α, while tuning remains important for optimal performance.

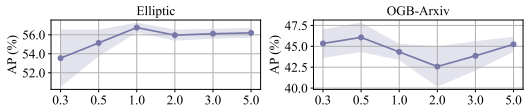


Figure 8: The sensitivity of hyperparameter α.

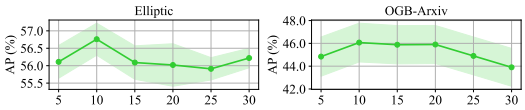


Figure 9: The sensitivity of hyperparameter β.

Sensitivity Analysis of β. We further evaluate the sensitivity of β, which controls the importance of memory replay during adaptation. Results in Figure 9 show that the performance is sensitive to β within a reasonable range. Further analysis on τ and λ is provided in Appendix D.12.

5 Related Work

Graph Continual Learning. Graph continual learning aims to enable graph models to adapt to evolving data distributions while mitigating catastrophic forgetting. Existing approaches are typically categorized into three major classes: regularization methods, parameter isolation methods, and memory replay methods. Regularization-based methods constrain weight updates to preserve important knowledge from previous tasks; for example, TWP [22] incorporates topology-aware weight preservation, and RieGrace [34] leverages gradient-based importance measures. Parameter isolation techniques, such as HPNs [48], allocate task-specific parameters to prevent interference between old and new knowledge. Memory replay methods alleviate forgetting by storing and reusing representative data, as seen in ER-GNN [54] and SSM [49]. While these methods are effective in continual learning scenarios, they generally assume access to labeled training data, limiting their applicability to real-world settings. Inspired by memory replay, our work eliminates the need for additional trainable components by reformulating the distribution matching objective and deriving a closed-form solution for crystallizing each incoming domain into a compact memory at test time. This makes our method well suited for the more realistic and challenging setting of graph learning under non-stationary distribution shifts at test time, where the input distribution drifts continually and no source data or annotations are available.

Graph Domain Adaptation. Graph domain adaptation [26, 43, 45, 6, 23, 17] aims to transfer knowledge learned from labeled data in a source domain to different but related target domains. For example, UDA-GCN [42] and AdaGCN [4] enhance adaptability by exploiting graph topology and reducing source–target discrepancies through local/global consistency or domain adversarial objectives. Continual Test-Time Adaptation (CTTA) is a more practical extension of traditional Test-Time Adaptation (TTA), which tackles the challenge of adapting models to continuously drifting target domains. Recent CTTA approaches, such as CoTTA [38], incorporate strategies like bi-level pseudo-label averaging and stochastic model resets to improve robustness. The most recent graph-specific approach, GCAL [31], trains an auxiliary VGAE to generate memory graphs for replay, but the additional generative training is expensive and scales poorly under prolonged drift. However, existing methods largely overlook the continual nature of distribution shifts at test time and the redundancy within each domain that is irrelevant to the underlying semantics. In this paper, we explicitly model inter-domain relationships under continual drift through state-evolving memories and improve both the efficiency and effectiveness of adaptation by crystallizing each domain in closed form, without requiring any auxiliary trainable component at test time.

6 Conclusion

In this paper, we present **EMC** (**E**fficient **M**emory **C**rystallization), a training-free test-time framework that crystallizes each incoming graph domain into a compact, semantically faithful memory through a closed-form solution, and models inter-domain dependencies via state-evolving memories with provably tighter generalization error bounds. Extensive experiments demonstrate the superiority of EMC, achieving substantial efficiency gains (87.4% less runtime and 92.4% less GPU memory on average over the recent baseline GCAL) while outperforming state-of-the-art baselines on graphs under non-stationary distribution shifts.

Acknowledgements

This work is supported by NSFC (Grant Nos. U25B2070 and T2588101), by the CCSE project (SKLCCSE-2026ZX-07), and by the Academic Excellence Foundation of BUAA for PhD Students.

References

[1] Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] Jinyu Cai, Yunhe Zhang, and Jicong Fan. Self-discriminative modeling for anomalous graph detection. In *Proceedings of the International Conference on Machine Learning*, pages 6387–6408. PMLR, 2025.

[3] Jinyu Cai, Yunhe Zhang, Pengyang Wang, and See-Kiong Ng. Moegad: A mixture-of-experts framework with pseudo-anomaly generation for graph-level anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(4):4473–4489, 2026.

[4] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[5] Jun Dan, Weiming Liu, Chunfeng Xie, Hua Yu, Shunjie Dong, and Yanchao Tan. Tfgda: Exploring topology and feature alignment in semi-supervised graph domain adaptation through robust clustering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[6] Zhengming Ding, Sheng Li, Ming Shao, and Yun Fu. Graph adaptive knowledge transfer for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–52, 2018.

[7] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[8] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[11] Yue Hou, Xueyuan Chen, He Zhu, Ruomei Liu, Bowen Shi, Jiaheng Liu, Junran Wu, and Ke Xu. Nc2d: Novel class discovery for node classification. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 849–859, 2024.

[12] Yue Hou, Ruomei Liu, Yingke Su, Junran Wu, and Ke Xu. Graph out-of-distribution detection via test-time calibration with dual dynamic dictionaries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 21797–21804, 2026.

[13] Yue Hou, Yingke Su, Junran Wu, and Ke Xu. Test-time graph ood detection via dynamic dictionary expansion and ood score calibration. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8845–8853, 2025.

[14] Yue Hou, He Zhu, Ruomei Liu, Yingke Su, Junran Wu, and Ke Xu. Redundancy-aware test-time graph out-of-distribution detection. *Advances in Neural Information Processing Systems*, 38:119899–119942, 2026.

[15] Yue Hou, He Zhu, Ruomei Liu, Yingke Su, Jinxiang Xia, Junran Wu, and Ke Xu. Structural entropy guided unsupervised graph out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17258–17266, 2025.

[16] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[17] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. Empowering graph representation learning with test-time graph transformation. In *The Eleventh International Conference on Learning Representations*, 2022.

[18] Vassilis Kalofolias. How to learn a graph from smooth signals. In *Artificial intelligence and statistics*, pages 920–929. PMLR, 2016.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[20] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[21] Chenxi Liu, Lixu Wang, Lingjuan Lyu, Chen Sun, Xiao Wang, and Qi Zhu. Deja vu: Continual model generalization for unseen domains. In *The Eleventh International Conference on Learning Representations*, 2023.

[22] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8653–8661, 2021.

[23] Shikun Liu, Tianchun Li, Yongbin Feng, Nhan Tran, Han Zhao, Qiang Qiu, and Pan Li. Structural re-weighting improves graph domain adaptation. In *International Conference on Machine Learning*, pages 21778–21793. PMLR, 2023.

[24] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[25] Junyu Luo, Zixuan Ouyang, Xiao Luo, Hourun Li, Zhiping Xiao, Yifan Wang, and Ming Zhang. Space-based parameter evolving with lightweight optimization for graph adaptation to evolving shifts. In *Proceedings of the ACM Web Conference 2026*, pages 1529–1539, 2026.

[26] Xinhong Ma, Tianzhu Zhang, and Changsheng Xu. Gcan: Graph convolutional adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8266–8276, 2019.

[27] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[28] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022.

[29] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[30] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[31] Ziyue Qiao, Qianyi Cai, Hao Dong, Jiawei Gu, Pengyang Wang, Meng Xiao, Xiao Luo, and Hui Xiong. GCAL: Adapting graph models to evolving domain shifts. In *Forty-second International Conference on Machine Learning*, 2025.

[32] Ziyue Qiao, Xiao Luo, Meng Xiao, Hao Dong, Yuanchun Zhou, and Hui Xiong. Semi-supervised domain adaptation in graph transfer learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 2279–2287, 2023.

[33] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[34] Li Sun, Junda Ye, Hao Peng, Feiyang Wang, and S Yu Philip. Self-supervised continual graph learning in adaptive riemannian spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4633–4642, 2023.

[35] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[36] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[37] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[38] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

[39] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6861–6871, 2019.

[40] Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, and Ke Xu. Sega: Structural entropy guided anchor view for graph contrastive learning. In *International Conference on Machine Learning*. PMLR, 2023.

[41] Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. Structural entropy guided graph hierarchical pooling. In *International Conference on Machine Learning*, pages 24017–24030. PMLR, 2022.

[42] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, pages 1457–1467, 2020.

[43] Man Wu, Shirui Pan, and Xingquan Zhu. Attraction and repulsion: Unsupervised domain adaptive graph contrastive learning network. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[44] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *International Conference on Learning Representations*, 2022.

[45] Jiaren Xiao, Quanyu Dai, Xiaochen Xie, Qi Dou, Ka-Wai Kwok, and James Lam. Domain adaptive graph infomax via conditional adversarial networks. *IEEE Transactions on Network Science and Engineering*, 2022.

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[47] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3801–3809, 2018.

[48] Xikun Zhang, Dongjin Song, and Dacheng Tao. Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4622–4636, 2022.

[49] Xikun Zhang, Dongjin Song, and Dacheng Tao. Sparsified subgraph memory for continual graph representation learning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1335–1340. IEEE, 2022.

[50] Xikun Zhang, Dongjin Song, and Dacheng Tao. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[51] Yunhe Zhang, Jinyu Cai, Qi Hao, Pengyang Wang, and See-Kiong Ng. Escaping the homophily trap: A threshold-free graph outlier detection framework via clustering-guided edge reweighting. In *Proceedings of the International Conference on Learning Representations*, 2026.

[52] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International conference on machine learning*, pages 7523–7532. PMLR, 2019.

[53] Xin Zheng, Dongjin Song, Qingsong Wen, Bo Du, and Shirui Pan. Online GNN evaluation under test-time graph distribution shifts. In *The Twelfth International Conference on Learning Representations*, 2024.

[54] Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4714–4722, 2021.

NeurIPS 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: Abstract and introduction accurately reflect the paper's contributions and scope.

   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 limitations regarding the future research direction of the work are discussed in the conclusion section.

   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 full set of assumptions and proofs is provided in the paper.

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: Our paper fully discloses all the information needed to reproduce the main experimental results.

   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: The adopted datasets are public datasets. The code of EMC is available.

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: Our paper specifies all the training and testing details.

   Guidelines:

   - 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: Our paper shows average results over 5 runs with different random seeds and reports the variance for our metrics.

   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: Our paper provides sufficient information on the computer resources.

   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: Our research complies with the NeurIPS Code of Ethics.

   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: The societal impacts of the work are discussed in Appendix E.

    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: [N/A]

    Justification: The paper poses no such risk.

    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: The adopted datasets and compared baselines are properly credited.

    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: [N/A]

    Justification: Our paper does not release new assets.

    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: [N/A]

    Justification: Our 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: [N/A]

    Justification: Our 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.
    - 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: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.

Technical Appendices and Supplementary Material

A Notations **21**

B Algorithm **21**

B.1 Complexity Analysis 22

B.2 Analysis of Scalability on Memory Construction 23

B.3 Semantic-Anchor Aggregation at Test Time 24

C Theoretical Justification **25**

C.1 Reformulation of Distribution Matching 25

C.2 Proof of Proposition C.2 26

C.3 Proof of Proposition 3.3 26

C.4 Proof of Error Upper Bounds 27

C.5 Proof of Proposition 3.7 30

D Additional Experiments **32**

D.1 Dataset Description 32

D.2 Baseline Details 32

D.3 Experimental Setting 33

D.4 Configurations 34

D.5 Performance Comparison under the GCAL Setting 34

D.6 Extended Performance Comparison across Node- and Graph-level Tasks 34

D.7 Additional Performance Matrices 35

D.8 Robustness to Different Domain-Shift Patterns 36

D.9 Memory Construction Alternatives 36

D.10 Additional Visualizations of Crystallized Memories 36

D.11 Additional Scaling Efficiency Analysis 37

D.12 Additional Results of Parameter Study 38

E Broader Impacts **39**

A Notations

As an expansion of the notations in our work, we summarize the frequently used notations in Table 4. We adopt G^t to denote the graph from the t-th target domain in the arrival order, and $G^{(t)}$ as a generic notation when the index (t) may refer to any past or current domain in $\{1, \ldots, T\}$.

B Algorithm

The overall optimization process of EMC is summarized in Algorithm 1. At each step t, we first construct the memory $\tilde{G}^{(t)}$ and form the state-evolved representation $\mathbf{Z}'^{(t)}$ (Steps 1–3), and only then minimize $\mathcal{L}$ to update $\mathbf{\Theta}_{t-1} \to \mathbf{\Theta}_t$ (Steps 4–5). The KL term in Eq. (12) therefore measures the predictive shift on each stored memory induced by the parameter update, serving as an anti-forgetting regularizer with the stability guarantee of Proposition 3.7.

Table 4: Summary of notations and descriptions.

Notations	Descriptions
$\mathcal{T} = \{G^t\}_{t=1}^{T}$	Sequence of T graphs from target domains
$G^{(t)} = \{\mathbf{A}^{(t)}, \mathbf{X}^{(t)}\}$	Graph from the t-th target domain
$\mathbf{A}^{(t)} \in \mathbb{R}^{N_t \times N_t}$	Adjacency matrix of $G^{(t)}$ with N_t nodes
$\mathbf{X}^{(t)} \in \mathbb{R}^{N_t \times d}$	Node feature of $G^{(t)}$ with d-dim features
$\mathbf{\Theta}_t$	Model parameters adapted after t-th domain
$\hat{\mathbf{A}}$	Normalized adjacency matrix with self-loops
$\mathbf{H} = \hat{\mathbf{A}}^K \mathbf{X}$	Embeddings after K-hop propagation
$\mathbf{H}'$	Aggregated memory embeddings
$\tilde{G}^{(t)} = \{\mathbf{A}', \mathbf{X}'\}$	Crystallized memory of $G^{(t)}$
$\mathbf{A}'$	Sparsified memory adjacency matrix
$\mathbf{X}'$	Feature matrix of memory $\tilde{G}^{(t)}$
$\mathbf{L}'$	Laplacian matrix of $\mathbf{A}'$
$\mathbf{R}^i$	Class-wise aggregation matrix for class i
$\hat{\mathbf{P}}$	Aggregation matrix for crystallized memories
$\mathbf{\Psi}$	Learnable linear weight of the GNN encoder
Φ	Distribution over parameter configurations of $\mathbf{\Psi}$
$\mathbf{Z} = \mathbf{H}\mathbf{\Psi}$	Representation of the original graph
$\mathbf{Z}' = \mathbf{H}'\mathbf{\Psi}$	Representation of the crystallized memory
$\mathbf{Z}'^{(t)}$	State-evolved memory representation at step t
$\hat{\tilde{G}}^{(t)}$	Predictive distribution of the current model on $\tilde{G}^{(t)}$
$\epsilon^t(\hat{h})$	Prediction risk on the t-th target domain
$\mathcal{D}_{\mathrm{KL}}(\cdot\|\cdot)$	KL-divergence between two distributions
$d_{\mathcal{H}}(\cdot,\cdot)$	$\mathcal{H}$-divergence
$\mathcal{L}$, $\mathcal{L}_{\text{Adp}}$	Adaptation losses
α, β, τ	Hyperparameters of EMC
w_{t-1}, w_t	Convex-combination weights for state-evolving memory
λ	Memory size scaling factor, $N_t = \lambda^{-1} N_t'$

B.1 Complexity Analysis

The pipeline of EMC consists of the following components: Structure-aware Feature Propagation, Redundancy-eliminated Memory Construction, State Evolving Field Modeling, and Parameter Update. We analyze the time complexity of each module per test-time step t with input G^t.

- **Structure-aware Feature Propagation**: This step performs K-hop neighborhood smoothing over the input graph, with complexity $\mathcal{O}(K|\mathcal{E}_t|d)$, where $|\mathcal{E}_t|$ is the number of edges and d is the node feature dimension. We then use confidence-weighted K-means clustering for partitioning each class. Let $|C_i|$ be the number of nodes in class i and $|C_i'|$ be the number of clusters. The time complexity for class i is $\mathcal{O}(|C_i||C_i'|dn_{\text{iter}})$, where n_{iter} is the number of K-means iterations. This process is repeated for each class.
- **Redundancy-eliminated Memory Construction**: Computing pairwise cosine similarity among N_t' memory nodes takes $\mathcal{O}(N_t'^2 d)$. Solving the regularized least-squares problem involves matrix inversion with complexity $\mathcal{O}(N_t'^3 + N_t'^2 d + (K+1)N_t'|\mathcal{E}_t|' + d|\mathcal{E}_t|')$, where $|\mathcal{E}_t|'$ is the number of edges in the memory graph. The solution is efficient because $N_t' \ll N_t$.
- **State Evolving Field Modeling**: Fusion of memories from previous and current time steps involves linear interpolation of node embeddings and has negligible cost: $\mathcal{O}(N_t'd)$.
- **Parameter Update**: The loss computation includes entropy and KL-divergence between memory predictions. For t total adaptation steps, the cumulative cost of computing $\mathcal{L}_{\text{Adp}}$ scales as $\mathcal{O}(tN_t'd)$ across domains. Parameter updates via gradient descent depend on the model size but are standard and efficient in practice.

The total time complexity per test graph G^t is dominated by class-wise partitioning and memory feature construction. However, since the memory size N_t' is only a small fraction (e.g., 1%) of the original graph size N_t, EMC remains highly efficient. Techniques such as approximate clustering

Algorithm 1: Test-Time Adaptation Process of EMC.

Input: Sequence of test graph samples $\{G^{(t)} = (\mathbf{A}^{(t)}, \mathbf{X}^{(t)})\}_{t=1}^{T}$ from T domains; pre-trained GNN f with parameters $\mathbf{\Theta}_0$.
Output: Adapted parameters $\{\mathbf{\Theta}_t\}_{t=1}^{T}$.

Initialize the model with $\mathbf{\Theta}_0$.
for $t = 1$ *to* T **do**
  `// Step 1: Structure-aware Feature Propagation`
  Compute $\mathbf{H}^{(t)} \leftarrow \hat{\mathbf{A}}^K \mathbf{X}^{(t)}$.
  Compute aggregation matrix $\hat{\mathbf{P}}^{(t)}$ via Eq. (1).
  Obtain aggregated features $\mathbf{H}'^{(t)} \leftarrow \hat{\mathbf{P}}^{(t)} \mathbf{H}^{(t)}$.
  `// Step 2: Redundancy-eliminated Memory Construction`
  Build sparse adjacency $\mathbf{A}'^{(t)}$ via Eq. (5).
  Solve closed-form objective for $\mathbf{X}'^{(t)}$ via Eq. (6).
  Construct domain-specific memory $\tilde{G}^{(t)} = (\mathbf{A}'^{(t)}, \mathbf{X}'^{(t)})$.
  `// Step 3: State Evolving Field`
  $\mathbf{Z}'^{(t)} \leftarrow w_{t-1}\, \mathbf{Z}'^{(t-1)} + w_t\, \mathbf{Z}'^{(t)}$ via Eq. (8).
  `// Step 4: Adaptation Loss Computation`
  Compute $\mathcal{L} \leftarrow H(G^{(t)}) + \beta \cdot \mathcal{L}_{\text{Adp}}(\tilde{G}^{1:t-1})$ via Eq. (12).
  `// Step 5: Parameter Update`
  Update $\mathbf{\Theta}_{t-1} \rightarrow \mathbf{\Theta}_t$ via gradient descent on $\mathcal{L}$.
end
return $\{\mathbf{\Theta}_t\}_{t=1}^{T}$

(e.g., FAISS [7]) and sparse propagation further accelerate the process, making the method suitable for real-time test-time adaptation under continual drift.

Complexity Comparison with the Baseline GCAL. Compared with our efficient closed-form memory crystallization, GCAL [31] introduces substantial additional computational overhead at each adaptation step. Specifically, for the t-th test-time domain $G^t = (\mathbf{A}^t, \mathbf{X}^t)$, the extra complexity mainly comes from the following components:

- **VGAE Encoding:** Encoding the input graph into latent mean and variance representations incurs a cost of $\mathcal{O}(\mathcal{E}_t d + N_t d^2)$, where N_t and $\mathcal{E}_t$ are the number of nodes and edges in G^t, respectively, and d is the node feature dimension.
- **VGAE Decoding:** Reconstructing the synthetic graph $\tilde{G}^t$ through sampling and edge scoring requires $\mathcal{O}(N_t^2 d)$ due to pairwise similarity computations.
- **Dual Forward Pass:** Forwarding both G^t and $\tilde{G}^t$ through the base GNN leads to $\mathcal{O}(\mathcal{E}_t d + N_t d^2 + \bar{\mathcal{E}}_t d + N_t' d^2)$, where N_t' and $\bar{\mathcal{E}}_t$ denote the node and edge count of the synthetic graph $\tilde{G}^t$.
- **Gradient Matching:** Computing real gradients and second-order synthetic gradients with respect to P model parameters has cost $\mathcal{O}(P + P^2)$.
- **Regularization Terms:** The KL divergence and reparameterization loss over latent variables add $\mathcal{O}(N_t d)$ complexity, and the MMD loss comparing node embeddings from G^t and $\tilde{G}^t$ incurs $\mathcal{O}(N_t N_t' d)$.
- **Additional Inner-loop Iterations:** All steps above are repeated for L inner-loop updates, resulting in a total cost of $\mathcal{O}\big(L \cdot (\mathcal{E}_t d + N_t d^2 + N_t^2 d + \bar{\mathcal{E}}_t d + N_t' d^2 + P + P^2 + N_t N_t' d)\big)$.

In each adaptation epoch, GCAL introduces the above-mentioned additional computational overhead, which is further amplified during the parameter update stage of the GNN model. This design leads to significantly higher time and memory costs during continual test-time adaptation, making GCAL less practical for long-horizon or resource-constrained scenarios.

B.2 Analysis of Scalability on Memory Construction

The closed-form construction in EMC is designed for scalability: the linear system $\mathbf{X}' = (\mathbf{Q}^\top \mathbf{Q} + \alpha \mathbf{L}')^{-1} \mathbf{Q}^\top \mathbf{H}'$ is solved on the small memory graph of size N_t' rather than on the full input of size

N_t. With $N'_t = 0.01N_t$ throughout our experiments, the linear system stays compact regardless of the input scale, decoupling the dominant cost of memory construction from the input size. This compact formulation is what makes the method memory-efficient (▷ Section 4.2) and what we now show transfers to large input graphs.

Sparsity from the design. The similarity threshold τ in Eq. (5) keeps only edges between node pairs whose embeddings are sufficiently close, so the memory adjacency $\mathbf{A}'$ is constructed sparse by design rather than densified post hoc. As a result, $\mathbf{Q} = \hat{\mathbf{A}}'^K$ inherits this sparsity, and the resulting linear system remains well-conditioned and efficient to solve in practice.

Table 5: Memory-construction scaling on a single graph up to 200k nodes (memory size $N'_t = 0.01N_t$). The matrix inversion in Eq. (6) stays under 1 second even at $N_t = 200$k.

Metric / #Node	1k	2k	5k	10k	20k	40k	50k	100k	200k
Matrix Inversion Time (s)	0.0077	0.0096	0.0156	0.0234	0.0974	0.1142	0.1224	0.3211	0.7121
Total Processing Time (s)	0.6414	1.0703	2.8145	17.7121	54.2386	184.0252	271.6589	1158.5502	5580.7541
GPU Memory (MiB)	11.38	15.17	29.75	66.17	181.71	564.07	829.68	3358.29	12806.52

Empirical scaling on a single graph. Table 5 reports runtime and GPU memory of the memory-construction step on a single Erdős–Rényi graph as the node count N_t grows from 1k to 200k (memory size fixed at $N'_t = 0.01N_t$). The matrix inversion remains under 1 second even at $N_t = 200$k, while total processing scales near-linearly in the dominant propagation term, confirming that the closed-form solver is not a bottleneck at industrial scale.

B.3 Semantic-Anchor Aggregation at Test Time

We elaborate on how class-wise node partitioning in Section 3.1 is instantiated at test time, where ground-truth labels of target-domain graphs are unavailable, and explain why the resulting memories are robust to noisy partitioning.

Test-Time Setting under Continual Drift. We consider a standard covariate shift setting, where the label space remains unchanged across domains, *i.e.*, $\mathbb{P}_{\mathcal{S}}(Y) = \mathbb{P}_{\mathcal{T}}(Y)$ and $\mathbb{P}_{\mathcal{S}}(G) \neq \mathbb{P}_{\mathcal{T}}(G)$. That is, distribution shifts at test time affect the input graph distributions, while the semantic meanings of classes are consistent with those observed during pre-training. This assumption is critical: under covariate shift the class-conditional decision boundary is preserved across domains, so systematic confident misclassification by the current model is substantially less likely than under label shift.

Semantic Anchors for Class-wise Aggregation. Class-wise partitioning at test time is guided by the model's predicted semantics rather than ground-truth labels. Concretely, given a node v in the incoming graph $G^{(t)}$, the current model with parameters $\mathbf{\Theta}_{t-1}$ produces a predictive distribution $p_{\mathbf{\Theta}_{t-1}}(y|v)$, from which a soft semantic assignment is derived together with a confidence score $r_v = \max_{c\in\mathcal{Y}} p_{\mathbf{\Theta}_{t-1}}(y = c|v)$. Nodes with similar predicted semantics are grouped together, and the confidence score weights each node's contribution to the resulting centroid through the aggregation matrix $\mathbf{R}^i$ in Eq. (1). Crucially, the predicted semantics serve as *semantic anchors* for grouping rather than as supervisory targets, so the procedure does not rely on the correctness of any individual assignment.

Design Choices that Stabilize the Aggregation. Memory crystallization in EMC relies on four complementary design choices that together keep aggregation stable under continual drift:

- **Covariate shift limits systematic errors.** As stated above, the preserved class-conditional decision boundary makes systematic confident misclassification far less likely than under label shift.
- **Anchors are not supervisory targets.** They are used only to partition nodes into sub-clusters for centroid aggregation. Even noisy assignments yield valid centroids through a many-to-one smoothing effect, analogous to the behavior of k-means under individual point mislabeling [24].
- **K-hop propagation denoises before partitioning.** The features $\mathbf{H} = \hat{\mathbf{A}}^K\mathbf{X}$ aggregate neighborhood information, smoothing out local prediction errors before memory construction.
- **Dirichlet regularization provides an independent smoothness constraint.** The term $\alpha \cdot \mathrm{tr}(\mathbf{X}'^\top\mathbf{L}'\mathbf{X}')$ in Eq. (6) enforces feature smoothness on the crystallized memory, further reducing any residual noise from partitioning.

Empirically, EMC consistently outperforms baselines on all four datasets (▷ Section 4.2), including Elliptic (temporal Bitcoin transaction shifts that can be abrupt) and OGB-Arxiv (year-by-year citation shifts), confirming that the four design choices above translate into reliable partitioning under continual drift in practice.

C Theoretical Justification

C.1 Reformulation of Distribution Matching

We provide a complete proof for Proposition 3.1 and retain the original analysis in the remark. We denote the memory crystallized from $G^{(t)}$ as $\tilde{G}^{(t)} = \{\mathbf{A}', \mathbf{X}'\}$. After multi-hop propagation, let $\mathbf{Z} = \mathbf{H}\mathbf{\Psi} = \hat{\mathbf{A}}^K\mathbf{X}\mathbf{\Psi}$ and $\mathbf{Z}' = \mathbf{H}'\mathbf{\Psi} = \hat{\mathbf{A}}'^K\mathbf{X}'\mathbf{\Psi}$ denote the representations of the original graph $G^{(t)}$ and its memory $\tilde{G}^{(t)}$, respectively, where $\mathbf{\Psi}$ is the learnable linear weight of the GNN encoder and Φ is the distribution over its parameter configurations. To directly align the distribution of the input domain with that of its memory, the discrepancy between propagated features is minimized as:

$$\min_{\tilde{G}^{(t)}} \mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{P}'\mathbf{H}'\mathbf{\Psi} - \mathbf{P}\mathbf{H}\mathbf{\Psi}\right\|^2. \tag{14}$$

Proposition C.1 (Restatement of Proposition 3.1)**.** Assuming that nodes in $\tilde{G}^{(t)}$ are organized in ascending order, $\mathbf{P}'$ degenerates into the identity matrix $\mathbf{I}$. Let $\hat{\mathbf{P}}$ be the row-normalized version of $\mathbf{P}$. Then, the objective in Eq. (14) can be upper-bounded by:

$$\mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{H}'\mathbf{\Psi} - \hat{\mathbf{P}}\mathbf{H}\mathbf{\Psi}\right\|^2 \le \mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2 \|\mathbf{\Psi}\|^2. \tag{15}$$

Proof. We start from Eq. (14). Under the canonical ordering of nodes in $\tilde{G}^{(t)}$, we have $\mathbf{P}' = \mathbf{I}$. By the construction of the aggregation matrix in the main text, $\mathbf{P}$ aggregates nodes within each class-wise subclass, and $\hat{\mathbf{P}}$ is its row-normalized version. Thus Eq. (14) reduces to

$$\mathop{\mathbb{E}}_{\mathbf{\Psi}\sim\Phi} \left\|\mathbf{H}'\mathbf{\Psi} - \hat{\mathbf{P}}\mathbf{H}\mathbf{\Psi}\right\|^2. \tag{16}$$

Let $\mathbf{M} \doteq \mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}$. Then

$$\left\|\mathbf{H}'\mathbf{\Psi} - \hat{\mathbf{P}}\mathbf{H}\mathbf{\Psi}\right\|^2 = \|\mathbf{M}\mathbf{\Psi}\|^2. \tag{17}$$

By sub-multiplicativity of matrix norms, for any compatible matrices $\mathbf{M}$ and $\mathbf{\Psi}$,

$$\|\mathbf{M}\mathbf{\Psi}\| \le \|\mathbf{M}\|\,\|\mathbf{\Psi}\|. \tag{18}$$

Squaring both sides yields

$$\|\mathbf{M}\mathbf{\Psi}\|^2 \le \|\mathbf{M}\|^2\,\|\mathbf{\Psi}\|^2. \tag{19}$$

Taking expectation over $\mathbf{\Psi} \sim \Phi$ on both sides gives

$$\mathbb{E}_{\mathbf{\Psi}\sim\Phi}\|\mathbf{M}\mathbf{\Psi}\|^2 \le \mathbb{E}_{\mathbf{\Psi}\sim\Phi}\left(\|\mathbf{M}\|^2\,\|\mathbf{\Psi}\|^2\right) = \|\mathbf{M}\|^2\,\mathbb{E}_{\mathbf{\Psi}\sim\Phi}\|\mathbf{\Psi}\|^2, \tag{20}$$

where we used that $\mathbf{M} = \mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}$ is independent of $\mathbf{\Psi}$. Substituting back $\mathbf{M}$ and rewriting yields Eq. (15), completing the proof. □

Remark. The objective in Eq. (14) is upper-bounded by Eq. (15), where $\mathbf{\Psi}$ is the (random) weight matrix that transforms the K-th order propagated features $\mathbf{H}'$ and $\mathbf{H}$. Here $\hat{\mathbf{A}}' = \tilde{\mathbf{D}}'^{\frac{1}{2}}\tilde{\mathbf{A}}'\tilde{\mathbf{D}}'^{\frac{1}{2}}$ and $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\frac{1}{2}}$ represent the symmetric normalized adjacency matrices, where $\tilde{\mathbf{A}}'$ and $\tilde{\mathbf{A}}$ are adjacency matrices with self-loops, and $\tilde{\mathbf{D}}'$ and $\tilde{\mathbf{D}}$ are degree matrices for $\tilde{\mathbf{A}}'$ and $\tilde{\mathbf{A}}$. Since $\mathbf{\Psi}$ is independent of both $\tilde{G}^{(t)}$ and $\hat{\mathbf{P}}$, minimizing the upper bound reduces to the deterministic matching problem:

$$\arg\min_{\tilde{G}^{(t)},\hat{\mathbf{P}}} \left\|\mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2. \tag{21}$$

According to Proposition C.1, we expect a symmetric encoding procedure between the original $G^{(t)}$ and the crystallized memory $\tilde{G}^{(t)}$. Therefore, our goal is to construct $\mathbf{A}'$ and $\mathbf{X}'$ satisfying

$\hat{\mathbf{A}}'^K\mathbf{X}' = \mathbf{H}'$. To this end, we utilize a pre-defined graph structure and compute $\mathbf{X}'$ in closed form. Specifically, we construct the structure via a similarity-based thresholding scheme:

$$\mathbf{A}'_{i,j} = \begin{cases} 1 & \text{if } \cos(\mathbf{H}'_i, \mathbf{H}'_j) > \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{22}$$

where $\cos(\cdot,\cdot)$ measures the cosine similarity and τ is the hyper-parameter for graph sparsification. To ensure that reconstructed features change smoothly between connected nodes, we introduce the Dirichlet energy constraint [18] into the feature reconstruction loss to quantify the smoothness of graph signals. The optimization objective for $\mathbf{X}'$ is:

$$\arg\min_{\mathbf{X}'} \left\| \hat{\mathbf{A}}'^K\mathbf{X}' - \mathbf{H}' \right\|^2 + \alpha \operatorname{tr}(\mathbf{X}'^\top \mathbf{L}'\mathbf{X}'), \tag{23}$$

where $\mathbf{L}' = \mathbf{D}' - \mathbf{A}'$ is the Laplacian matrix, $\mathbf{D}'$ is the degree matrix, and α controls the smoothness strength.

C.2 Proof of Proposition C.2

Proposition C.2. Assume that $\tilde{G}^{(t)} = \{\mathbf{A}', \mathbf{X}'\}$ is the crystallized memory of an input graph; the closed-form solution of Eq. (23) takes the form $\mathbf{X}' = (\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}')^{-1}\mathbf{Q}^\top\mathbf{H}'$, where $\mathbf{Q} = \hat{\mathbf{A}}'^K$.

Proof. Let us denote the objective in Eq. (23) as $\mathcal{I} = \left\| \hat{\mathbf{A}}'^K\mathbf{X}' - \mathbf{H}' \right\|^2 + \alpha \operatorname{tr}(\mathbf{X}'^\top \mathbf{L}'\mathbf{X}')$. To solve the optimization problem in Eq. (23), we compute the first- and second-order derivatives of $\mathcal{I}$ with respect to $\mathbf{X}'$:

$$\begin{aligned} \frac{\partial \mathcal{I}}{\partial \mathbf{X}'} &= 2\mathbf{Q}^\top(\mathbf{Q}\mathbf{X}' - \mathbf{H}') + \alpha(\mathbf{L}' + \mathbf{L}'^\top)\mathbf{X}' = 2(\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}')\mathbf{X}' - 2\mathbf{Q}^\top\mathbf{H}', \\ \frac{\partial^2 \mathcal{I}}{\partial \mathbf{X}'^2} &= 2(\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}'). \end{aligned} \tag{24}$$

According to the definition of $\mathbf{A}'$ in Eq. (22), both $\mathbf{L}'$ and $\mathbf{Q}$ are positive semi-definite matrices. Therefore, the second-order derivative $\frac{\partial^2 \mathcal{I}}{\partial \mathbf{X}'^2}$ is positive semi-definite, and the objective function $\mathcal{I}$ is convex. This implies that the first-order derivative $\frac{\partial \mathcal{I}}{\partial \mathbf{X}'}$ corresponds to a convex function with a unique minimum. Setting the gradient to zero yields the closed-form solution:

$$\mathbf{X}' = (\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}')^{-1}\mathbf{Q}^\top\mathbf{H}', \tag{25}$$

where $\mathbf{Q} = \hat{\mathbf{A}}'^K$. □

Remark. The closed-form solution operates on a compact $N'_t \times N'_t$ system over the crystallized memory rather than on the full $N_t \times N_t$ system over the input graph, with $N'_t \ll N_t$. This decoupling between the construction cost and the input scale is the key reason EMC attains a much smaller memory footprint than methods that train auxiliary modules on the full input graph.

C.3 Proof of Proposition 3.3

The following result identifies the precise condition under which marginal alignment is equivalent to joint alignment, motivating the state-evolving design that follows.

Proposition C.3. Given any pair of sequences of samples from the target domain, $\mathcal{T} = \{G^t\}_{t=1}^T$ and the corresponding crystallized memories $\tilde{\mathcal{T}} = \{\tilde{G}^t\}_{t=1}^T$, following the joint distributions $\mathbb{P}_{G^{1:T}}$ and $\mathbb{Q}_{\tilde{G}^{1:T}}$ respectively, with each G^t and $\tilde{G}^t$ following the marginal distributions $\mathbb{P}_{G^t}$ and $\mathbb{Q}_{\tilde{G}^t}$. Then the following relation holds:

$$\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^{1:T}} \,\|\, \mathbb{Q}_{\tilde{G}^{1:T}}) \equiv \sum_{t=1}^{T} \mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t} \,\|\, \mathbb{Q}_{\tilde{G}^t}) \quad \text{iff} \quad \mathbb{P}_{G^{1:T}} = \prod_{t=1}^{T} \mathbb{P}_{G^t}, \quad \mathbb{Q}_{\tilde{G}^{1:T}} = \prod_{t=1}^{T} \mathbb{Q}_{\tilde{G}^t}, \tag{26}$$

where $\mathcal{D}_{\mathrm{KL}}$ denotes the Kullback–Leibler divergence and $\equiv$ indicates that the two expressions are identically equal.

Proof. First, we prove sufficiency: if the snapshots are temporally independent, then the two expressions coincide. By the definition of KL divergence and its chain rule, the divergence between the joint distributions of $\mathcal{T}$ and $\tilde{\mathcal{T}}$ can be expanded as:

$$\begin{aligned}
&\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^{1:T}} \| \mathbb{Q}_{\tilde{G}^{1:T}}) \\
=&\sum_{G^{1:T}} \mathbb{P}_{G^{1:T}} \log \frac{\mathbb{P}_{G^{1:T}}}{\mathbb{Q}(\tilde{G}^{1:T})} \\
=&\sum_{G^{1:T}} \mathbb{P}_{G^{1:T}} \log \frac{\mathbb{P}(G^1)}{\mathbb{Q}(\tilde{G}^1)} + \sum_{G^{1:T}} \mathbb{P}_{G^{1:T}} \log \frac{\prod_{t=2}^{T} \mathbb{P}(G^t \mid G^{1:t-1})}{\prod_{t=2}^{T} \mathbb{Q}(\tilde{G}^t \mid \tilde{G}^{1:t-1})} \\
=&\sum_{G^1} \mathbb{P}(G^1) \log \frac{\mathbb{P}(G^1)}{\mathbb{Q}(\tilde{G}^1)} + \sum_{t=2}^{T} \sum_{G^{1:t-1}} \mathbb{P}(G^{1:t-1}) \Big[\sum_{g^t} \mathbb{P}(G^t \mid G^{1:t-1}) \log \frac{\mathbb{P}(G^t \mid G^{1:t-1})}{\mathbb{Q}(\tilde{G}^t \mid \tilde{G}^{1:t-1})} \Big] \\
=&\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^1} \| \mathbb{Q}_{\tilde{G}^1}) + \sum_{t=2}^{T} \mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t|G^{1:t-1}} \| \mathbb{Q}_{\tilde{G}^t|\tilde{G}^{1:t-1}}).
\end{aligned} \tag{27}$$

Under the assumption of temporal independence, *i.e.*, $\mathbb{P}_{G^{1:T}} = \prod_{t=1}^{T} \mathbb{P}_{G^t}$ and $\mathbb{Q}_{\tilde{G}^{1:T}} = \prod_{t=1}^{T} \mathbb{Q}_{\tilde{G}^t}$, each conditional distribution $\mathbb{P}_{G^t|G^{1:t-1}}$ reduces to the marginal $\mathbb{P}_{G^t}$, and likewise for $\mathbb{Q}$. As a result, the sum of conditional KL terms collapses to $\sum_{t=2}^{T} \mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t} \| \mathbb{Q}_{\tilde{G}^t})$. Hence the joint KL over the graph–memory divergence collapses to:

$$\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^{1:T}} \| \mathbb{Q}_{\tilde{G}^{1:T}}) = \sum_{t=1}^{T} \mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t} \| \mathbb{Q}_{\tilde{G}^t}). \tag{28}$$

Conversely, if the equality between the joint KL and the sum of marginal KLs holds, then by the chain-rule expansion above, every conditional KL term $\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t|G^{1:t-1}} \| \mathbb{Q}_{\tilde{G}^t|\tilde{G}^{1:t-1}})$ must equal $\mathcal{D}_{\mathrm{KL}}(\mathbb{P}_{G^t} \| \mathbb{Q}_{\tilde{G}^t})$ for all t. By the strict convexity of KL divergence, this forces $\mathbb{P}_{G^t|G^{1:t-1}} = \mathbb{P}_{G^t}$ and $\mathbb{Q}_{\tilde{G}^t|\tilde{G}^{1:t-1}} = \mathbb{Q}_{\tilde{G}^t}$ almost surely, i.e., temporal independence. □

Remark. Proposition C.3 is a *biconditional*: it characterizes precisely when summing per-domain KL divergences equals the joint KL divergence, namely when and only when the domains (and their crystallized memories) are independent. Aligning the marginal distributions of individual graphs $G^{(t)}$ is therefore sufficient to ensure joint distribution alignment over the given sequence of test samples *if and only if* the memories are independent. Since real test-time domains are generally correlated under continual drift, this implies that aligning marginal distributions alone is insufficient, motivating us to model the relationships among continuously arriving test samples under covariate shift through the state evolving field over crystallized memories.

C.4 Proof of Error Upper Bounds

Definition C.4 (Classification error)**.** The classification error of the function $\hat{h}$ under domain $\mathcal{D}_i$ is defined as

$$\epsilon_i(\hat{h}) = \mathbb{E}_{X \sim \mathcal{D}_i} |\hat{h}(X) - h(X)|. \tag{29}$$

For binary classification functions h and $\hat{h}$, we have:

$$\begin{aligned}
\epsilon_i(\hat{h}) = \epsilon_i(\hat{h}, h) &= \mathbb{E}_{X \sim \mathcal{D}_i} |\hat{h}(X) - h(X)| \\
&= \mathbb{P}_{X \sim \mathcal{D}_i}(\hat{h}(X) \neq h(X)).
\end{aligned} \tag{30}$$

Definition C.5 ($\mathcal{H}$-divergence)**.** Given two induced domain feature space distributions $\tilde{\mathcal{D}}_s, \tilde{\mathcal{D}}_t$ and a hypothesis space $\mathcal{H}$, the $\mathcal{H}$-divergence between $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_t$ is defined as:

$$d_{\mathcal{H}}(\tilde{\mathcal{D}}_s, \tilde{\mathcal{D}}_t) = 2 \sup_{h \in \mathcal{H}} \left| \mathbb{E}_{X \sim \tilde{\mathcal{D}}_s}[h(X) = 1] - \mathbb{E}_{X \sim \tilde{\mathcal{D}}_t}[h(X) = 1] \right|. \tag{31}$$

Setting for the bounds. We adopt the standard domain-adaptation framework [52] as the analytical setting, whose bounds are applied to each single adaptation step, where $\mathbb{P}_{\mathcal{S}}$ refers to the source distribution from which $\mathbf{\Theta}_0$ is pre-trained and $\mathbb{P}_{\mathcal{T}}$ refers to the empirical distribution of the current target graph G^t. The bounds therefore characterize per-step generalization behavior; the continual aspect across $\{G^t\}_{t=1}^T$ is handled separately through the state-evolving design in Theorem 3.6 and the stability bound in Proposition 3.7, which together control how predictive behavior on past memories evolves as $\mathbf{\Theta}_{t-1} \to \mathbf{\Theta}_t$.

Lemma C.6 (Error Upper Bound for Domain Shift [52])**.** Let $\tilde{\mathbb{P}}_{\mathcal{S}}$ and $\tilde{\mathbb{P}}_{\mathcal{T}}$ denote the induced distributions over the feature space for $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{T}}$. The following inequality holds for the risk $\epsilon_0^{\mathcal{T}}(\hat{h})$ with single step adaptation on the target domain $\mathbb{P}_{\mathcal{T}}$:

$$\epsilon_0^{\mathcal{T}}(\hat{h}) \leq \min\{\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}), \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})\} + \epsilon^{\mathcal{S}}(\hat{h}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}), \tag{32}$$

where $d_{\mathcal{H}}$ is the $\mathcal{H}$-divergence.

Remark. Note that this bound was first derived by [52]. The first term measures the disagreement between the source and target labeling functions, the second term is the source error, and the third term quantifies the discrepancy between the marginal feature distributions. The bound identifies three key factors for successful domain adaptation: small source risk, close marginal distributions, and consistent labeling functions across domains.

Lemma C.7. Let $\tilde{\mathbb{P}}_{\mathcal{S}}$ and $\tilde{\mathbb{P}}_{\mathcal{T}}$ denote the induced distributions over the feature space for $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{T}}$. Then for any $h^{\mathcal{S}} \in \mathcal{H}, h^{\mathcal{T}} \in \mathcal{H}$, we have

$$|\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) - \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})| \leq d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}). \tag{33}$$

Proof. By definition, for any $h^{\mathcal{S}} \in \mathcal{H}, h^{\mathcal{T}} \in \mathcal{H}$, we have:

$$\begin{aligned} &\left|\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) - \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})\right| \\ =& \sup_{h \in \mathcal{H}} \left|\mathbb{E}_{X \sim \mathbb{P}_{\mathcal{S}}}|h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)| - \mathbb{E}_{X \sim \mathbb{P}_{\mathcal{T}}}|h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)|\right| \\ =& \sup_{h \in \mathcal{H}} \left|\mathbb{E}_{X \sim \mathbb{P}_{\mathcal{S}}}[h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)] - \mathbb{E}_{X \sim \mathbb{P}_{\mathcal{T}}}[h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)]\right|. \end{aligned} \tag{34}$$

Since $\|\cdot\|_\infty \leq 1$ for all $h \in \mathcal{H}$, we have $0 \leq |h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)| \leq 1$ for all $X \in \mathbb{P}_{\mathcal{S}}/\mathbb{P}_{\mathcal{T}}$. Define a hypothesis space $\hat{\mathcal{H}} := \left\{\operatorname{sgn}(|h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)| - z) \mid h^{\mathcal{S}}, h^{\mathcal{T}} \in \mathcal{H}, 0 \leq z \leq 1\right\}$. Then we use Fubini's theorem to bound:

$$\begin{aligned} &\left|\mathbb{E}_{X \sim \mathbb{P}_{\mathcal{S}}}[|h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)|] - \mathbb{E}_{X \sim \mathbb{P}_{\mathcal{T}}}[|h^{\mathcal{S}}(X) - h^{\mathcal{T}}(X)|]\right| \\ =& \left|\int_0^1 (\Pr^{\mathcal{S}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z) - \Pr^{\mathcal{T}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z))dz\right| \\ \leq& \int_0^1 \left|\Pr^{\mathcal{S}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z) - \Pr^{\mathcal{T}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z)\right| dz \\ =& \sup_{z \in [0,1]} \left|\Pr^{\mathcal{S}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z) - \Pr^{\mathcal{T}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z)\right|. \end{aligned} \tag{35}$$

Combining Eq. (34) and (35), and in view of the definition of $\mathcal{H}$-divergence:

$$\begin{aligned} &\sup_{h^{\mathcal{S}}, h^{\mathcal{T}} \in \mathcal{H}} \sup_{z \in [0,1]} \left|\Pr^{\mathcal{S}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z) - \Pr^{\mathcal{T}}(|h^{\mathcal{S}} - h^{\mathcal{T}}| > z)\right| \\ =& 2 \sup_{\hat{h} \in \hat{\mathcal{H}}} \left|\Pr^{\mathcal{S}}(\hat{h}(X) = 1) - \Pr^{\mathcal{T}}(\hat{h}(X) = 1)\right| \\ =& 2 \sup_{\hat{h} \in \hat{\mathcal{H}}} \left|\Pr^{\mathcal{S}}(\hat{h}) - \Pr^{\mathcal{T}}(\hat{h})\right| \\ =& d_{\hat{\mathcal{H}}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) \ \leq \ d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}). \end{aligned} \tag{36}$$

□

Theorem C.8 (Error Upper Bound with Crystallized Memories)**.** Let $\tilde{\mathbb{P}}_{\mathcal{S}}$ and $\tilde{\mathbb{P}}_{\mathcal{T}}$ denote the induced distributions over the feature space for $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{T}}$. The risk $\epsilon^{\mathcal{T}}(\hat{h})$ with crystallized memories (with mixed weight $w_0 + w_1 = 1$, $w_0 \geq 0$, $w_1 \geq 0$, and $\hat{h} = w_0\hat{h}^{\mathcal{S}} + w_1\hat{h}^{\mathcal{T}}$) is upper bounded by:

$$\epsilon^{\mathcal{T}}(\hat{h}) \leq \epsilon^{\mathcal{T}}(\hat{h}, h^{\mathcal{S}}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}). \tag{37}$$

Proof. The proof of the above theorem is as follows:

$$\begin{aligned}
\epsilon^{\mathcal{T}}(\hat{h}) &= \epsilon^{\mathcal{T}}(\hat{h}, h^{\mathcal{T}}) \\
&= \mathbb{E}_{x\sim\mathbb{P}_{\mathcal{T}}}[|\hat{h}(x) - h^{\mathcal{T}}(x)|] \\
&= \mathbb{E}_{x\sim\mathbb{P}_{\mathcal{T}}}[|\hat{h}(x) - h^{\mathcal{S}}(x) + h^{\mathcal{S}}(x) - h^{\mathcal{T}}(x)|] \\
&\leq \mathbb{E}_{x\sim\mathbb{P}_{\mathcal{T}}}[|\hat{h}(x) - h^{\mathcal{S}}(x)|] + \mathbb{E}_{x\sim\mathbb{P}_{\mathcal{T}}}[|h^{\mathcal{S}}(x) - h^{\mathcal{T}}(x)|] \\
&= \epsilon^{\mathcal{T}}(\hat{h}, h^{\mathcal{S}}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}).
\end{aligned} \tag{38}$$

Comparison of Error Upper Bounds. From Lemma C.6, our bound is further bounded by:

$$\begin{aligned}
\epsilon^{\mathcal{T}}(\hat{h}) &\leq \epsilon^{\mathcal{T}}(\hat{h}, h^{\mathcal{S}}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&\leq \epsilon^{\mathcal{S}}(\hat{h}, h^{\mathcal{S}}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&= \epsilon^{\mathcal{S}}(\hat{h}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}).
\end{aligned} \tag{39}$$

We further introduce a density ratio $\frac{\Pr^{\mathcal{T}}(x)}{\Pr^{\mathcal{S}}(x)}$ to represent the divergence between the source and target domains. Concretely, assume that $\Pr^{\mathcal{T}}(x)/\Pr^{\mathcal{S}}(x) \leq M$ for some finite constant M on the support of $\mathbb{P}_{\mathcal{S}}$. This is a standard regularity condition in the domain-adaptation literature [52] and is automatically satisfied in our setting because $\mathbb{P}_{\mathcal{T}}$ is the empirical distribution over a finite target graph G^t with $N_t < \infty$ nodes, so the ratio is bounded by $N_t \cdot \max_x \Pr^{\mathcal{S}}(x)^{-1}$ on the support.

$$\begin{aligned}
\epsilon^{\mathcal{T}}(\hat{h}) &= \int |\hat{h}(x) - h^{\mathcal{S}}(x)| \cdot \Pr^{\mathcal{T}}(x)dx + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&= \int |\hat{h}(x) - h^{\mathcal{S}}(x)| \cdot \frac{\Pr^{\mathcal{T}}(x)}{\Pr^{\mathcal{S}}(x)} \cdot \Pr^{\mathcal{S}}(x)dx + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&= \mathbb{E}_{x\sim\mathbb{P}_{\mathcal{S}}} \frac{\Pr^{\mathcal{T}}(x)}{\Pr^{\mathcal{S}}(x)} \cdot |\hat{h}(x) - h^{\mathcal{S}}(x)| + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&= \frac{\Pr^{\mathcal{T}}(x)}{\Pr^{\mathcal{S}}(x)}\epsilon^{\mathcal{S}}(\hat{h}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&= \epsilon^{\mathcal{S}}(\hat{h}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}),
\end{aligned} \tag{40}$$

where the density ratio $\frac{\Pr^{\mathcal{T}}(x)}{\Pr^{\mathcal{S}}(x)}$ is generally unobservable and is commonly treated as a constant. The structural difference between Theorem C.8 and Lemma C.6 is the absence of the explicit $\mathcal{H}$-divergence penalty $d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}})$ in the prototype-based bound. To verify that this structural difference indeed yields a tighter bound, we examine both possible orderings of $\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}})$ and $\epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})$, since Lemma C.6 takes the minimum of these two terms.

Case 1: If $\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \leq \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})$, then the direct adaptation bound from Lemma C.6 becomes:

$$\epsilon_0^{\mathcal{T}}(\hat{h}) \leq \epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) + \epsilon^{\mathcal{S}}(\hat{h}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}). \tag{41}$$

Since $\epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \leq \epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}})$, we obtain:

$$\begin{aligned}
\epsilon^{\mathcal{T}}(\hat{h}) &\leq \epsilon^{\mathcal{S}}(\hat{h}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \\
&\leq \epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) + \epsilon^{\mathcal{S}}(\hat{h}) + d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) = \epsilon_0^{\mathcal{T}}(\hat{h}).
\end{aligned} \tag{42}$$

Case 2: If $\epsilon^{\mathcal{S}}(h^{\mathcal{S}}, h^{\mathcal{T}}) \geq \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})$, then $\min\{\cdot\} = \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}})$, and:

$$\epsilon^{\mathcal{T}}(\hat{h}) \leq \epsilon^{\mathcal{S}}(\hat{h}) + \epsilon^{\mathcal{T}}(h^{\mathcal{S}}, h^{\mathcal{T}}) = \epsilon_0^{\mathcal{T}}(\hat{h}) - d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) \leq \epsilon_0^{\mathcal{T}}(\hat{h}). \tag{43}$$

In both cases, the memory-based bound dominates $\epsilon_0^{\mathcal{T}}(\hat{h})$ by exactly $d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) \geq 0$. The fact that the same dominance margin holds under both orderings confirms that the improvement stems from the structural removal of $d_{\mathcal{H}}$ in Theorem C.8, rather than from the choice of which term Lemma C.6 happens to minimize over. □

Theorem C.9 (Continual Error Upper Bound with State-Evolving Memories)**.** Let $\tilde{\mathbb{P}}_{G^{t-1}}$ and $\tilde{\mathbb{P}}_{G^t}$ denote the induced feature distributions of two consecutive target domains at steps $t-1$ and t. The risk $\epsilon^t(\hat{h})$ with state-evolving memories (with mixed weight $w_{t-1} + w_t = 1$, $w_{t-1}, w_t \geq 0$, and $\hat{h} = w_{t-1}\hat{h}^{t-1} + w_t\hat{h}^t$) is upper bounded by:

$$\epsilon^t(\hat{h}) \leq \epsilon^t(\hat{h}, h^{t-1}) + \epsilon^t(h^{t-1}, h^t). \tag{44}$$

Proof. By substituting $\mathcal{T}$ with t and $\mathcal{S}$ with $t-1$ in Theorem C.8, we obtain the following inequality, which holds throughout the continual adaptation process:

$$\epsilon^t(\hat{h}) \leq \epsilon^t(\hat{h}, h^{t-1}) + \epsilon^t(h^{t-1}, h^t). \tag{45}$$

Comparison of Error Upper Bounds. Theorem C.9 can be regarded as a special case of Theorem C.8, where the t-th domain serves as the initial target domain for adaptation. Therefore, the same ordering of error upper bounds holds, and the bound remains tighter than that of $\epsilon_0^{\mathcal{T}}(\hat{h})$. □

Remark. Lemma C.6 provides a generalization error bound for adaptation using samples from the target domain. Theorem C.8 gives the error bound when adapting a model from the source domain to the target domain with the help of crystallized memories. Theorem C.9 further refines this by providing the error bound when modeling inter-domain relations through state-evolving memories during continual adaptation. We arrive at the following conclusions:

- From Theorem C.8, the error bound $\epsilon^{\mathcal{T}}(\hat{h})$ using crystallized memories is ***tighter*** than the bound $\epsilon_0^{\mathcal{T}}(\hat{h})$ for direct source-to-target adaptation, and the gap is at least $d_{\mathcal{H}}(\tilde{\mathbb{P}}_{\mathcal{S}}, \tilde{\mathbb{P}}_{\mathcal{T}}) \geq 0$.
- According to Theorem C.9, the error bound $\epsilon^t(\hat{h})$ using state-evolving memories is also ***tighter*** than $\epsilon_0^{\mathcal{T}}(\hat{h})$ throughout continual adaptation.

C.5 Proof of Proposition 3.7

Proposition C.10 (Stability Bound of the Adaptation Loss)**.** Consider the adaptation step from $\mathbf{\Theta}_{t-1}$ to $\mathbf{\Theta}_t$. For each previously encountered domain indexed by $(t) \in \{1, \ldots, t-1\}$, $\mathcal{D}_{\mathrm{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)})$ can be interpreted as the divergence between the predictive distributions induced by the model before and after adaptation on the same crystallized memory $\tilde{G}^{(t)}$. Moreover, assume that: (i) the predictive distribution is given by a softmax with temperature $\tau > 0$; (ii) all predicted class probabilities are lower-bounded by $\gamma > 0$; and (iii) the encoder $h(\cdot)$ and the classifier $g(\cdot)$ are Lipschitz continuous *with respect to the model parameters*. Then the following upper bound holds:

$$\mathcal{D}_{\mathrm{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)}) \;\leq\; \frac{L_g^2 L_h^2}{4\,\gamma\,\tau^2} \left\| \mathbf{\Theta}_t - \mathbf{\Theta}_{t-1} \right\|_2^2, \tag{46}$$

where L_h and L_g are the parameter-space Lipschitz constants of $h(\cdot)$ and $g(\cdot)$.

Proof. We derive the stability bound step by step.

Step 1: Rewriting the KL divergence. For each $(t) \in \{1, \ldots, t-1\}$, define the predictive distributions on the crystallized memory $\tilde{G}^{(t)}$ as

$$p_{t-1}^{(t)}(\cdot) = \mathrm{Softmax}\Big(g\Big(h(\tilde{G}^{(t)}; \mathbf{\Theta}_{t-1})\Big)/\tau\Big), \quad p_t^{(t)}(\cdot) = \mathrm{Softmax}\Big(g\Big(h(\tilde{G}^{(t)}; \mathbf{\Theta}_t)\Big)/\tau\Big). \tag{47}$$

By definition, the KL divergence in Eq. (12) can be written as

$$\mathcal{D}_{\mathrm{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)}) = \mathcal{D}_{\mathrm{KL}}\Big(p_{t-1}^{(t)} \,\|\, p_t^{(t)}\Big). \tag{48}$$

Step 2: Upper bounding KL divergence by probability differences. Using the inequality $\log x \leq x - 1$ and the assumption $p_t^{(t)}(y) \geq \gamma$ for all classes y,

$$\mathcal{D}_{\mathrm{KL}}\Big(p_{t-1}^{(t)} \,\|\, p_t^{(t)}\Big) = \sum_y p_{t-1}^{(t)}(y) \log \frac{p_{t-1}^{(t)}(y)}{p_t^{(t)}(y)} \tag{49}$$

$$\leq \sum_y p_{t-1}^{(t)}(y) \left(\frac{p_{t-1}^{(t)}(y)}{p_t^{(t)}(y)} - 1 \right) \leq \frac{1}{\gamma} \|p_{t-1}^{(t)} - p_t^{(t)}\|_2^2. \tag{50}$$

Step 3: Lipschitz continuity of the softmax operator. Let $\mathbf{s}_{t-1}^{(t)} = g(h(\tilde{G}^{(t)}; \mathbf{\Theta}_{t-1}))$ and $\mathbf{s}_t^{(t)} = g(h(\tilde{G}^{(t)}; \mathbf{\Theta}_t))$ denote the corresponding logits. The softmax operator with temperature τ is $1/(2\tau)$-Lipschitz, yielding

$$\big\|p_{t-1}^{(t)} - p_t^{(t)}\big\|_2 \leq \frac{1}{2\tau} \big\|\mathbf{s}_{t-1}^{(t)} - \mathbf{s}_t^{(t)}\big\|_2. \tag{51}$$

Step 4: Lipschitz chain with respect to model parameters. By the parameter-space Lipschitz continuity of $g(\cdot)$ and $h(\cdot)$,

$$\big\|\mathbf{s}_{t-1}^{(t)} - \mathbf{s}_t^{(t)}\big\|_2 \leq L_g L_h \big\|\mathbf{\Theta}_{t-1} - \mathbf{\Theta}_t\big\|_2. \tag{52}$$

Step 5: Combining all bounds. Substituting the above inequalities yields

$$\mathcal{D}_{\mathrm{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)}) \leq \frac{L_g^2 L_h^2}{4\,\gamma\,\tau^2} \big\|\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}\big\|_2^2, \tag{53}$$

which completes the proof. □

Remark on the Lipschitz Assumption. Assumption (iii) is parameter-space Lipschitz continuity, namely that the network output changes smoothly as the parameters $\mathbf{\Theta}$ vary for a fixed input. This is fundamentally different from input-space Lipschitz continuity (which would require explicit spectral normalization). For any bounded-weight network, the parameter-space sensitivity satisfies $\|\partial h/\partial \mathbf{\Theta}\| \leq \prod_{\ell=1}^{L} \|\mathbf{W}^\ell\|_2 \cdot \|x\|$, which is finite on compact inputs and is directly bounded by weight decay through $\|\mathbf{W}^\ell\|_2 \leq \|\mathbf{W}^\ell\|_F$ [1, 27]. Therefore, standard backbones such as GCN and GraphSAGE satisfy Assumption (iii) under the weight decay used in our experiments, without any additional spectral normalization. Importantly, the proposition does not require $L_h L_g$ to be tight: it only asserts that small parameter updates lead to small predictive shifts on stored memories, which is sufficient to justify the KL regularization in Eq. (12) as an anti-forgetting mechanism.

Interpretation of the Two Loss Terms. Proposition C.10 provides an explicit interpretation of the adaptation loss in Eq. (12) and clarifies how the two components in the overall objective serve complementary purposes during adaptation.

- **Adaptation to the current domain.** The entropy minimization term $H(G^t)$ encourages confident predictions on the current test graph. By reducing predictive uncertainty on G^t, this term drives the model to better align with the data distribution of the newly arriving domain. This component focuses exclusively on the current domain.
- **Stability on previously encountered domains.** The KL divergence term $\mathcal{D}_{\mathrm{KL}}(\hat{\tilde{G}}^{(t)} \| \tilde{G}^{(t)})$ does not compare different domains. Instead, for each previously stored memory $\tilde{G}^{(t)}$, it constrains the change in predictive distributions induced by updating the model from $\mathbf{\Theta}_{t-1}$ to $\mathbf{\Theta}_t$. By Proposition C.10, this divergence admits an explicit quadratic upper bound with respect to the parameter update magnitude, implying that small updates lead to bounded changes in predictive behavior on past domains.

Table 6: The statistics of datasets with distribution shifts.

Category	Datasets	#Nodes	#Edges	#Domains
Regional Shifts	Twitch-explicit	1,912 - 9,498	31,299 - 153,138	7
	Facebook-100	769 - 41,554	33,312 - 2,724,458	12
Temporal Shifts	Elliptic	1,089 - 7,880	1,168 - 9,164	41
	OGB-Arxiv	4,427 - 39,711	1,225 - 38,735	11

- **Mitigating catastrophic forgetting through controlled transfer.** Together, the two terms balance adaptation and stability: entropy minimization promotes rapid adaptation to the current domain, while the KL regularization prevents excessive drift on previously encountered domains. The adaptation loss can thus be viewed as a mechanism that regulates cross-domain knowledge transfer during continual adaptation.

Overall, Proposition C.10 formalizes the intuition that test-time adaptation under non-stationary drift should not only fit the current domain but also preserve predictive consistency on past domains.

D Additional Experiments

D.1 Dataset Description

We evaluate EMC on four benchmark node classification datasets, encompassing two distinct types of distribution shifts: **regional shifts** and **temporal shifts**. This setting follows the continual adaptation protocol in GCAL [31], where each graph is treated as an individual domain. The statistical summary of all datasets is provided in Table 6.

Regional Shifts. *Facebook-100* [35] contains 100 ego-centric Facebook friendship networks collected from U.S. universities in 2005. Each network represents a unique university and varies significantly in node count, density, and degree distribution. We select three university graphs (Amherst41, Caltech36, and Johns Hopkins55) for initial training, and perform continual adaptation on eleven other university graphs without labels, following the protocol in [44]. *Twitch-Explicit* [33] includes seven social graphs of Twitch users from different regions (DE, ENGB, ES, FR, PTBR, RU, TW). Each graph contains users as nodes and friendship relations as edges. The DE network is used for pretraining, and the remaining regional graphs are used for continual test-time adaptation, also without labels.

Temporal Shifts. *OGB-Arxiv* [16] is a citation network of Arxiv CS papers across 40 research areas. Nodes represent papers, and edges indicate citation links. We pretrain the model on papers published before 2011 and adapt it sequentially to papers published from 2011 onward, capturing the temporal evolution of scientific collaborations. *Elliptic* [30] consists of 49 graph snapshots of Bitcoin transaction networks. Nodes represent transactions, and edges denote payment flows. Approximately 20% of nodes are labeled as licit or illicit. To avoid extreme class imbalance, we use snapshots 7–9 for pretraining and adapt the model continually using the remaining graphs, reflecting the temporal dynamics of illicit activity detection.

Across all datasets, we follow a consistent strategy: pretrain a GNN on a small set of labeled graphs, then continually adapt to the remaining unlabeled graphs in a test-time online setting. To simplify notation in our tables and figures, we refer to Twitch-Explicit as **Twitch**, Facebook-100 as **Facebook**, and OGB-Arxiv as **Arxiv**.

D.2 Baseline Details

We compare EMC against ten representative baseline methods. The "Base Model" refers to the GNN pre-trained on source domains and directly applied to target domains without any adaptation, serving as a lower-bound reference. These methods are grouped into two main categories: (1) one-step test-time adaptation methods and (2) continual test-time adaptation methods. To ensure fair comparison, all non-graph-specific methods are adapted to GNN-based architectures.

- **One-step test-time adaptation methods** perform adaptation using only the current domain without memory or explicit long-term modeling. DANN [9] is a classical domain adaptation approach that minimizes domain discrepancy via adversarial training using source and target data. Tent [36]

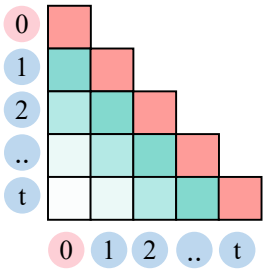


Figure 10: An example of the performance matrix. For the current t-th domain: TDG is computed as the mean of the green elements; TDA is the mean of the pink elements; AP is the average of the elements in the last row; AF is the average of the differences between each element in the last row (excluding the last one) and the corresponding element in the previous row.

minimizes the entropy of predictions on unlabeled target samples to adapt the model. BN Stats Adapt [20] updates network weights and batch normalization statistics based on the current test batch. EERM [44], designed specifically for graphs, maximizes environment risk variance to address domain shift and out-of-distribution challenges. GTRANS [17] refines both graph structures and node features using a contrastive surrogate loss to improve test-time performance.

- **Continual test-time adaptation methods** aim to maintain performance under sequential distribution shifts. CoTTA [38] combines weight-averaged predictions and stochastic neuron restoration to mitigate error accumulation and catastrophic forgetting. EATA [28] improves adaptation efficiency through entropy-based updates and a Fisher-based regularizer for stability. GCAL [31] enhances cross-domain adaptation by training an auxiliary VGAE to generate memory graphs, but its reliance on a separately trained generator limits scalability in long-term scenarios. STEM [25] introduces a controller-worker architecture to dynamically generate adapter parameters for a frozen base GNN.

D.3 Experimental Setting

In this work, we are the first to investigate the generalizability and adaptability of graph models under non-stationary distribution shifts at test time, while also addressing the challenge of catastrophic forgetting.

We distinguish between two key concepts: **generalizability** and **adaptability**. Generalizability refers to the model's inherent ability to perform well on previously seen domains, even while being adapted to new domains. This emphasizes the model's capacity to retain knowledge throughout the continual adaptation process. In contrast, adaptability reflects how effectively a model can enhance its performance on a newly encountered domain through test-time optimization using unlabeled data. In other words, generalizability focuses on performance retention, while adaptability emphasizes performance gain under novel domain shifts. Figure 10 provides a visual comparison of these two metrics, with exact formulations introduced below.

For our experimental configuration, we use a 2-layer GCN as the backbone for three datasets, except for OGB-Arxiv where we adopt GraphSAGE [10]. For pretraining, we split the source graphs into 60% training, 20% validation, and 20% testing. The pretraining phase uses a learning rate of 1e-04 and a weight decay of 5e-04. During adaptation, the learning rate is set to 1e-03 with the same weight decay. Pretraining lasts for 100–200 epochs depending on the dataset, while adaptation uses 1–10 epochs per domain. All hyperparameter choices follow the protocol of GCAL [31]. The number of nodes in each crystallized memory N_t' is set to 1% of the total number of nodes in the input graph. We analyze the sensitivity of hyperparameters α and β in the main paper, with extended analyses on τ and the memory size factor λ in Section D.12. The convex-combination weights w_{t-1}, w_t are fixed hyperparameters satisfying $w_{t-1} + w_t = 1$ and selected by grid search over $w_t \in \{0.3, 0.5, 0.7, 0.9\}$. All results are averaged over five independent runs, with standard deviations reported.

To measure continual adaptation performance, we construct a performance matrix $M \in \mathbb{R}^{T\times T}$, where $M_{i,j}$ (for $i \geq j$) represents the model's performance on domain j after training on domain i. Following [17, 44], we use ROC-AUC for Twitch-Explicit, Accuracy for Facebook-100 and OGB-Arxiv, and F1 Score for Elliptic. We report the following metrics derived from M:

Table 7: Performance comparison (%, mean ± standard deviation for five runs) across node-level (Twitch-explicit, OGB-Arxiv) and graph-level (Mutagenicity, FRANKENSTEIN) datasets under continual distribution shifts. "Avg." is computed uniformly as the mean of TDG and TDA. "↑" indicates higher is better and "↓" indicates lower is better. The best results are highlighted in **bold** and the runner-ups are underlined.

Dataset	Methods	Backbone: GCN						Backbone: SAGE					
		AP (↑)	AF (↑)	TDG (↑)	TDA (↑)	Avg. (↑)	GPU (↓)	AP (↑)	AF (↑)	TDG (↑)	TDA (↑)	Avg. (↑)	GPU (↓)
Node-level Adaptation													
Twitch-explicit	GCAL	55.65±0.09	0.42±0.13	54.78±0.60	53.20±0.62	53.99	11159.80	58.20±0.35	0.79±0.13	59.65±0.39	57.02±0.49	58.34	8580.63
	STEM	**56.66±0.05**	-0.07±0.17	**57.35±0.07**	**55.87±0.15**	**56.61**	7393.27	59.49±0.06	-0.45±0.02	**63.56±0.01**	56.33±0.05	59.95	7961.84
	EMC	55.95±0.15	**0.47±0.08**	55.56±0.13	55.83±0.14	55.70	**854.97**	**61.62±0.12**	**0.86±0.04**	62.02±0.13	**60.36±0.16**	**61.19**	**2833.18**
OGB-Arxiv	GCAL	14.15±6.94	-1.80±2.06	17.90±8.58	13.77±5.57	15.83	22342.47	45.22±0.17	0.76±0.05	42.40±7.74	41.80±6.59	42.10	22365.84
	STEM	10.25±0.01	-2.79±0.01	12.75±0.01	10.56±0.01	11.66	7010.74	31.71±0.14	-4.40±0.19	34.19±0.25	32.77±0.18	33.48	7012.99
	EMC	**19.32±1.57**	**3.38±1.99**	**19.39±1.47**	**20.39±1.71**	**19.89**	**299.18**	**46.07±1.70**	**0.99±0.66**	**47.11±1.80**	**43.52±1.25**	**45.31**	**391.02**
Graph-level Adaptation													
Mutagenicity	GCAL	67.96±0.10	-1.55±0.14	67.59±0.06	69.13±0.04	68.36	3805.50	65.49±0.78	-0.70±0.54	64.22±0.63	66.19±0.33	65.20	1221.09
	STEM	64.48±0.26	-1.60±0.22	66.73±0.11	64.72±0.13	65.72	10361.79	**67.58±0.71**	-3.92±0.77	**70.42±0.50**	**70.33±0.15**	**70.38**	10357.06
	EMC	**68.40±1.66**	**-0.85±0.61**	**69.94±0.81**	**69.71±1.22**	**69.82**	**85.14**	67.15±0.25	**-0.66±0.07**	66.34±0.07	66.68±0.21	66.51	**111.61**
FRANKENSTEIN	GCAL	59.26±0.66	-0.22±0.21	59.53±0.33	59.32±0.52	59.42	4950.38	**58.34±0.70**	**0.40±0.64**	**56.87±0.46**	**57.44±0.17**	**57.16**	2085.58
	STEM	61.61±0.41	**1.41±0.25**	60.61±0.38	59.81±0.21	60.21	8322.35	54.36±0.33	-0.96±0.86	53.91±0.67	54.15±0.59	54.03	10461.86
	EMC	**61.86±0.67**	-0.45±0.15	**61.25±0.15**	**61.11±0.75**	**61.18**	**201.65**	56.42±0.10	-0.43±0.08	53.80±0.13	55.38±0.05	54.59	**222.39**

- **Average Performance (AP)**: AP $= \frac{1}{T}\sum_{i=1}^{T} M_{T,i}$, reflecting the model's overall adaptation ability after completing all domain shifts.
- **Average Forgetting (AF)**: AF $= \frac{1}{T-1}\sum_{i=1}^{T-1}(M_{T,i} - M_{i,i})$, quantifying the degradation in performance on past domains.
- **Target Domain Generalization (TDG)** [21]: TDG $= \frac{1}{T-1}\sum_{i=2}^{T} M_{i-1,i}$, measuring model generalization to unseen domains before adaptation.
- **Target Domain Adaptation (TDA)** [21]: TDA $= \frac{1}{T}\sum_{i=1}^{T} M_{i,i}$, measuring performance on each domain after adaptation.

D.4 Configurations

We conduct the experiments with:

- Operating System: Ubuntu 20.04 LTS.
- CPU: Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz, 256GB RAM.
- GPU: Tesla V100 PCIe 32GB GPU.
- Software: Python 3.7, Pytorch 1.8, CUDA 11.0, and Pytorch-Geometric 2.0.1.

D.5 Performance Comparison under the GCAL Setting

We provide the complete comparison summarized in Section 4.2 between EMC and the 10 representative baselines under the GCAL setting in Table 2. We use GCN as the backbone on Twitch-explicit, Facebook-100, and Elliptic, and GraphSAGE on OGB-Arxiv. The baselines span both *one-step test-time adaptation* (DANN, Norm, TENT, EERM, GTrans, LEBED) and *continual test-time adaptation* (CoTTA, EATA, GCAL, STEM). For STEM, we report results reproduced from the official codebase, which only supports Twitch-explicit and OGB-Arxiv, and the remaining two datasets are therefore marked as Not Applicable.

Results show that EMC achieves the best AP on three of the four datasets (Facebook-100, Elliptic, OGB-Arxiv) and the best AF on all four. On Twitch-explicit, STEM attains a slightly higher AP but exhibits a negative AF, indicating that its gain on the latest domain comes at the cost of forgetting earlier ones. By contrast, EMC consistently maintains a positive AF across all datasets, confirming that the closed-form memory crystallization and state-evolving design jointly support adaptability without sacrificing knowledge retention.

D.6 Extended Performance Comparison across Node- and Graph-level Tasks

To further situate EMC against the very recent state-of-the-art STEM [25], we extend the comparison to four datasets covering both node-level adaptation (Twitch-explicit, OGB-Arxiv) and graph-level

adaptation (Mutagenicity, FRANKENSTEIN), evaluated under two backbones (GCN and GraphSAGE). We compare EMC against the recent strong baselines GCAL [31] and STEM [25]. Results are reported in Table 7, where "Avg. " denotes the mean of TDG and TDA and serves as a balanced summary of pre- and post-adaptation behavior.

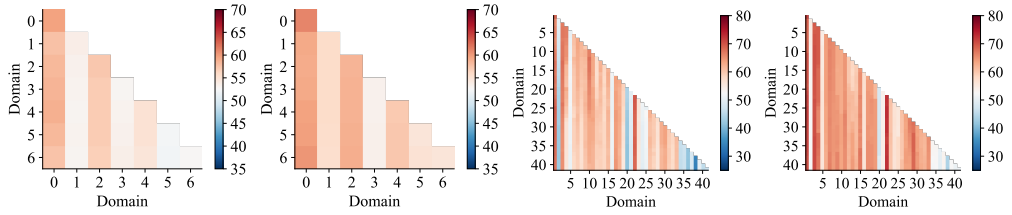
Effectiveness across node-level and graph-level tasks. EMC attains the best Avg. on five out of eight (dataset, backbone) combinations and the runner-up on the remaining three, demonstrating consistent effectiveness beyond the node-level setting that the prior literature on graph continual test-time adaptation has primarily focused on. The pattern of best results also covers both task granularities: EMC leads on three of four node-level entries and on two of four graph-level entries, indicating that the closed-form memory crystallization transfers naturally from node-level to graph-level adaptation without any task-specific modification. We further note that EMC consistently achieves the best AF in most settings, confirming that the state-evolving design effectively mitigates forgetting under continual drift, whereas STEM's stronger adaptation on certain domains is often accompanied by a negative AF (*e.g.*, -3.92 on Mutagenicity-GraphSAGE, -4.40 on Arxiv-GraphSAGE), reflecting a less favorable retention–adaptation trade-off.

Substantial Efficiency Advantage. The most pronounced gap appears in GPU memory consumption, where the closed-form construction in EMC removes the need for any auxiliary trainable component at test time. On node-level tasks, EMC is roughly an order of magnitude more memory-efficient than STEM and over an order of magnitude lighter than GCAL. The advantage widens on graph-level tasks, where EMC operates within a few hundred MiB while STEM exceeds ten thousand MiB on both Mutagenicity and FRANKENSTEIN, a difference of two orders of magnitude. Such a gap is structural rather than incidental: STEM and GCAL both rely on auxiliary modules whose memory cost grows with graph size, whereas the closed-form memory in EMC is solved analytically over a fraction of the input nodes (▷ Section 3.1), yielding a footprint that stays compact regardless of the underlying task granularity. These results confirm that EMC delivers competitive or superior effectiveness while remaining the most memory-efficient option among recent continual test-time adaptation methods, making it well-suited for prolonged adaptation under non-stationary shifts.

D.7 Additional Performance Matrices

We extend the performance-matrix visualization in Figure 4 of the main paper to the Twitch-Explicit and Elliptic datasets in Figure 11. Across both datasets, EMC maintains consistently strong performance across the diagonal and lower-triangular entries, indicating that the model not only adapts to each newly arriving domain but also retains its competence on previously encountered ones. In contrast, GCAL exhibits visibly degraded performance on earlier domains as adaptation proceeds, particularly on Elliptic, where the temporal drift can be abrupt.

We further note an observation on Facebook-100 in Figure 4: the performance of EMC on the 6th target domain is slightly lower than on neighboring domains. We attribute this to a larger covariate shift between this domain and the rest of the sequence, which makes single-pass adaptation under the test-time setting more challenging. Importantly, this localized dip does not propagate: the performance on subsequent domains recovers, and the overall AP and AF on Facebook-100 in Table 1 remain ahead of all baselines, confirming that EMC is robust to occasional outlier domains under continual drift.



(a) GCAL on Twitch-explicit (b) EMC on Twitch-explicit (c) GCAL on Elliptic (d) EMC on Elliptic

Figure 11: Performance matrices of EMC and GCAL on different datasets.

Table 8: Robustness to non-monotonic domain orderings on Twitch-Explicit.

Domain Order	Method	AP	AF	TDG	TDA	Time (s)	GPU (MiB)
ENGB, ES, FR, PTBR, RU, TW	GCAL	55.65	0.42	54.78	53.20	217.51	11159.8
	EMC	**55.95**	**0.47**	**55.56**	**55.83**	**74.64**	**854.97**
RU, FR, ES, ENGB, TW, PTBR	GCAL	53.58	0.18	52.67	53.32	217.11	12225.69
	EMC	**57.04**	**0.36**	**55.69**	**57.46**	**75.92**	**832.58**
FR, PTBR, TW, ENGB, RU, ES	GCAL	53.37	0.12	52.47	53.93	216.35	10304.73
	EMC	**57.12**	**0.46**	**55.68**	**58.20**	**73.75**	**648.62**
FR, PTBR, ES, RU, TW, ENGB	GCAL	53.42	0.02	52.63	54.10	217.56	10677.54
	EMC	**56.97**	**0.30**	**55.67**	**58.32**	**77.85**	**830.88**

Table 9: Robustness to cyclic domain shifts on Twitch-Explicit. Domains repeat to test stability under non-monotonic drift patterns.

Domain Sequence	Method	AP	AF	TDG	TDA	Time (s)	GPU (MiB)
ENGB, ES, FR, ENGB, ES, FR	GCAL	53.78	-0.21	53.27	54.62	209.09	10116.01
	EMC	**54.00**	**0.06**	**53.39**	**54.94**	**56.67**	**605.80**
PTBR, RU, TW, PTBR, RU, TW	GCAL	52.84	-0.06	52.03	54.04	223.54	14332.55
	EMC	**53.02**	-0.17	**52.67**	53.96	**95.79**	**828.53**
RU, FR, ES, RU, FR, ES	GCAL	52.11	0.00	51.25	52.56	218.63	12559.53
	EMC	**52.22**	-0.06	**51.39**	**52.80**	**77.21**	**774.43**
ENGB, TW, PTBR, ENGB, TW, PTBR	GCAL	**54.80**	0.02	**54.24**	54.54	214.80	10802.73
	EMC	54.61	**0.20**	53.84	**55.09**	**82.64**	**768.01**
ENGB, ES, FR, PTBR, RU, TW, ENGB, ES, FR, PTBR, RU, TW	GCAL	52.87	-0.35	52.71	53.27	444.15	20293.89
	EMC	**53.16**	**0.05**	**52.84**	**53.54**	**152.72**	**785.17**

D.8 Robustness to Different Domain-Shift Patterns

The state evolving field is a first-order linear interpolation, which prompts a natural question: does EMC remain effective when distribution shifts arrive in a non-monotonic or cyclic pattern? We note that Theorem 3.6 holds for *any* convex combination satisfying $w_{t-1} + w_t = 1$, so the theoretical tightness does not depend on a stronger linearity assumption. Empirically, the memory at each step is reconstructed from the current domain rather than carried forward, the state evolving field interpolates only encoded representations of adjacent domains, and entropy minimization keeps the model calibrated regardless of the order of arrival.

We further verify this with two stress tests on Twitch-Explicit with GCN backbone: (i) varying the order of the six target domains to break monotonicity, and (ii) cyclic shifts where target domains repeat. Tables 8 and 9 show that EMC outperforms GCAL across all orderings and cyclic patterns, with much lower runtime and GPU memory.

D.9 Memory Construction Alternatives

To further verify that EMC's efficiency advantage stems from *both* eliminating auxiliary generative training *and* producing informative memories, we compare against two simpler training-free memory schemes that replace GCAL's VGAE: *Random subgraph sampling* and *Feature averaging*. As shown in Table 10, both alternatives achieve comparable cost to EMC but suffer substantial performance drops, with feature averaging falling below the base model. This confirms that the closed-form crystallization in EMC produces memories that are not only cheap but also semantically faithful.

D.10 Additional Visualizations of Crystallized Memories

We complement the t-SNE projection in Figure 5 of the main paper with structural visualizations of the original graphs and their crystallized memories on the continuous-domain Twitch graphs, rendered with the NetworkX library from Figure 12 to Figure 22. To keep individual figures legible, the original graph is rendered after a fixed-rate node sampling, while the crystallized memory is rendered in full. Across the visualized domains, the structural layout of each crystallized memory closely traces that of its original graph, while the t-SNE projection in the main paper shows that

Table 10: Comparison with alternative training-free memory construction schemes on Twitch-Explicit with GCN backbone.

Method	AP (↑)	AF (↑)	TDG (↑)	Time (↓)	GPU (↓)
Base Model	53.88	NA	NA	NA	NA
GCAL with VGAE	55.65	0.42	54.78	217.51	11159.8
Random subgraph sampling	51.26	-1.45	51.32	67.81	800.96
Feature averaging	52.91	-0.85	52.19	65.56	783.13
EMC	**55.95**	**0.47**	**55.56**	**74.64**	**854.97**

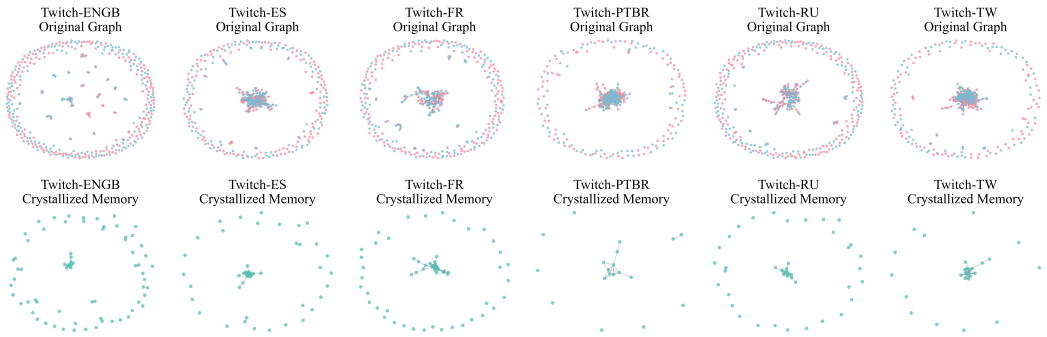


Figure 12: Structural visualization of the original graph and its crystallized memory on continuous-domain Twitch graphs with GCN backbone.

the two also remain aligned in the embedding space. Together, the two views support the design intuition that memory crystallization captures the semantically informative content of each domain in a compact, training-free manner.

D.11 Additional Scaling Efficiency Analysis

To evaluate the efficiency of EMC, we construct datasets by generating Erdős–Rényi graphs with varying node and domain scales, based on the size characteristics of existing datasets. In Figures 6 and 7 of the main paper, we report the runtime and peak GPU memory usage under increasing node and domain scales, where a red cross indicates an Out-Of-Memory (OOM) failure. We provide detailed experimental settings below.

Node Scaling Analysis. For the node scaling analysis, we fix the number of target domains to 10 and gradually increase the number of nodes per graph from 1k to 40k. The node feature dimension is set to 1288, consistent with OGB-Arxiv. The numerical results are reported in Table 11. This experiment analyzes how the efficiency of EMC changes as each test sample increases in size. Specifically, when the number of nodes exceeds 40k, GCAL fails due to OOM, while EMC remains within GPU memory limits even at the largest tested scale.

Domain Scaling Analysis. For the domain scaling analysis, we set the initial number of target domains to 10, with each graph containing 10k nodes and 128-dimensional node features (again aligned with OGB-Arxiv). We then gradually increase the number of target domains by a scaling factor. Detailed results are shown in Table 12. Compared to GCAL, which runs out of memory when the number of target domains reaches 50 or more, EMC maintains stable runtime and GPU usage across all scales.

We also benchmark the memory crystallization step alone on a single graph up to **200k nodes** in Appendix B.2, confirming that the closed-form inversion remains under one second even at this scale and is therefore not the bottleneck of the pipeline.

Storage Cost of Crystallized Memories. We additionally report the storage cost of the crystallized memories as the stream length increases in Table 13. Although the storage cost of crystallized memories grows with the number of domains, it remains small in absolute size. EMC maintains a low GPU memory footprint as the stream grows, requiring only 145.59 MB with 80 domains, while the crystallized memories occupy 39.07 MB. In comparison, GCAL exhibits substantially higher memory growth and runs out of GPU memory when the number of domains reaches 50.

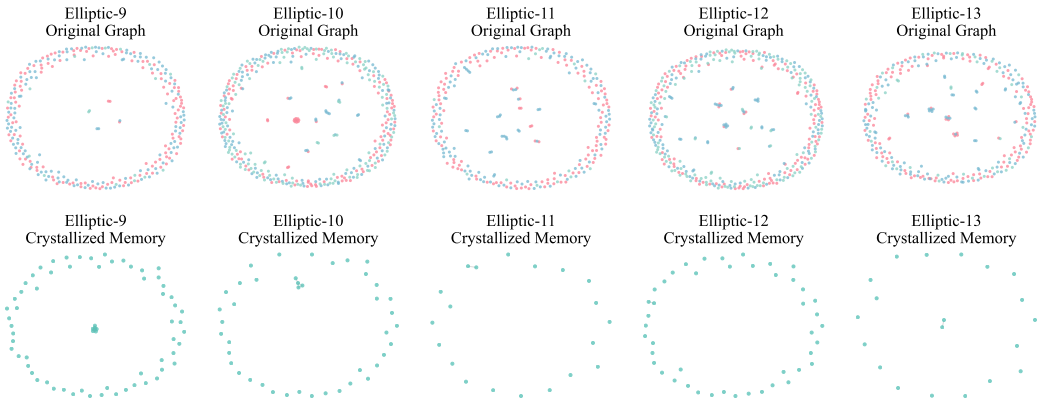


Figure 13: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part I).

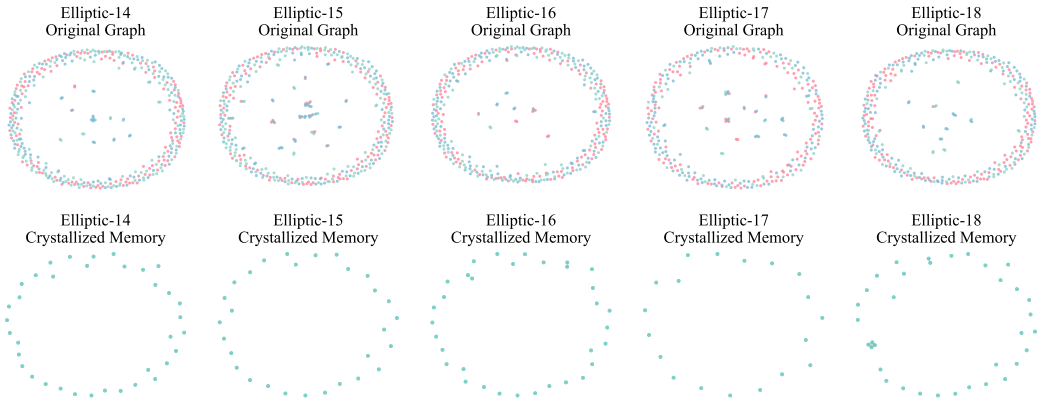


Figure 14: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part II).

Discussion on Memory Budgeting. A natural strategy is to cap the buffer at K memories and manage it as in agent-memory mechanisms, for example keeping the most recent ones in a first-in-first-out manner, retaining those most representative by similarity, or maintaining a weighted moving average of past memories. Because crystallization is a closed-form operation on node features, the oldest memories can also be merged and re-crystallized into a single memory at negligible cost, yielding constant storage, which is not available to generative replay without retraining the generator.

D.12 Additional Results of Parameter Study

We provide additional results on hyperparameter sensitivity in Figure 23 and Figure 24 to further support our analysis.

Sensitivity Analysis of τ. We investigate the impact of τ, the hyperparameter for graph sparsification, by varying it from 0.5 to 1.0. As shown in Figure 25, EMC exhibits robustness to a wide range of τ values, while proper tuning is still beneficial for optimal performance.

Analysis of Memory Size. To study the influence of the memory size on model performance, we define the relationship between the number of memory nodes N_t' and the number of original sample nodes N_t via a scaling factor λ, such that $N_t = \lambda^{-1} N_t'$. In the main paper, we set $\lambda = 0.01$. We further analyze the sensitivity of λ in Table 14. The size of the memory has a minor effect on adaptation performance: the model maintains both generalizability and adaptability regardless of the scaling factor between N_t' and N_t, demonstrating the robustness of EMC. Moreover, different values of λ incur only negligible variation in runtime and GPU memory cost, indicating that constructing crystallized memories introduces minimal and stable computational overhead during adaptation.

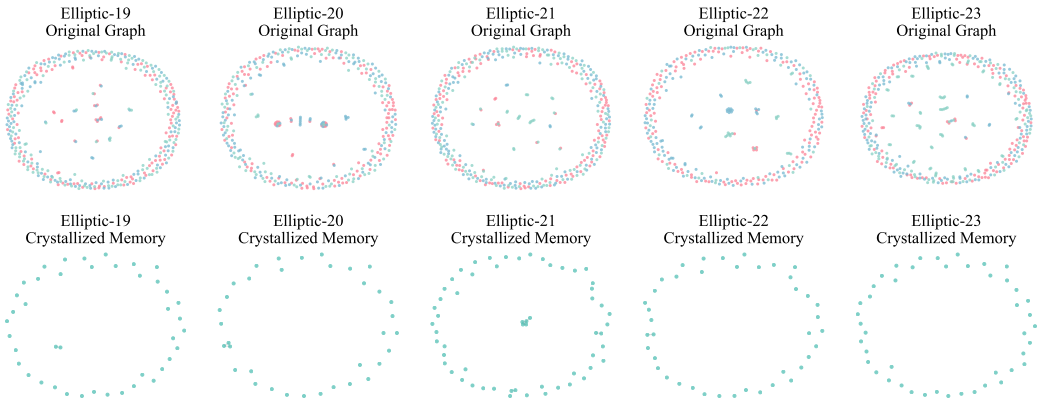


Figure 15: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part III).

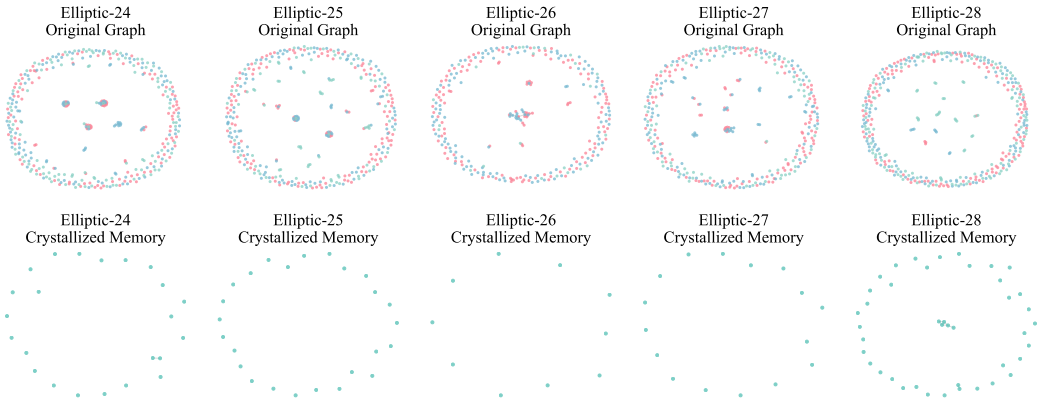


Figure 16: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part IV).

E Broader Impacts

Distribution shifts arise pervasively in real-world graph applications, including social networks, recommendation systems, citation analysis, and financial transaction monitoring, where the underlying data evolves continually over time. By enabling graph models to adapt under such non-stationary shifts in a training-free and memory-efficient manner, EMC contributes to deploying reliable graph learning systems at lower computational and energy cost, which we view as a positive societal outcome. We do not foresee specific negative societal impacts beyond those general to applied machine learning.

Future Directions. A broader direction suggested by our results is to distinguish task-relevant structural evolution from redundant variations under continual distribution shifts. EMC already follows this principle implicitly by crystallizing each incoming domain into a compact memory that preserves predictive information while suppressing redundant domain-specific content. Recent studies on graph OOD detection [15, 14, 13, 12] and anomaly detection [2, 3, 51] provide complementary evidence that explicitly separating essential graph structure from redundancy can improve robustness under distribution shifts. Structural-entropy-based [15, 41, 40] and redundancy-aware approaches [14] aim to preserve informative structural patterns while filtering irrelevant topology, whereas test-time calibration methods [13, 12] dynamically summarize evolving distributional factors through adaptive dictionaries. These perspectives suggest a unified future direction for continual graph adaptation: first extract the task-relevant structural information of each incoming domain, then measure distribution drift within this essential representation, and finally use the estimated drift to control memory construction and adaptation strength. Such a design could allow graph models to respond more strongly to meaningful structural changes while remaining stable against redundant or transient

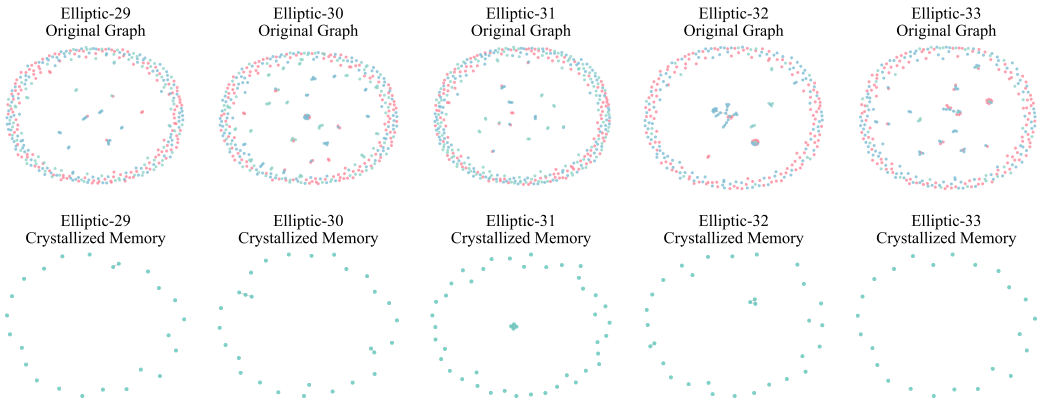


Figure 17: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part V).

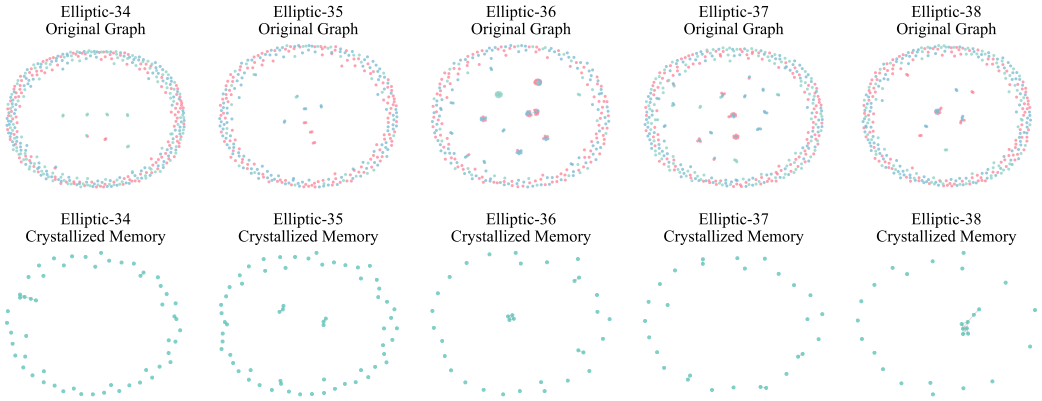


Figure 18: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part VI).

fluctuations. Extending this framework to open-world streams with emerging semantic classes could further connect continual domain adaptation with novel class discovery [11].

Limitations. Despite its practical advantages, EMC has two main limitations:

- **Fixed hyperparameter configuration.** Our current experiments use a single fixed hyperparameter configuration for each dataset. The hyperparameters are not dynamically adjusted for each incoming domain, which may limit adaptation when the characteristics of distribution shifts vary substantially over time.
- **Lack of explicit drift measurement.** EMC adapts sequentially under non-stationary distribution shifts, but does not explicitly estimate the magnitude of domain drift to determine the appropriate adaptation strength. Incorporating explicit drift estimation and drift-aware adaptation is a promising direction for future work.

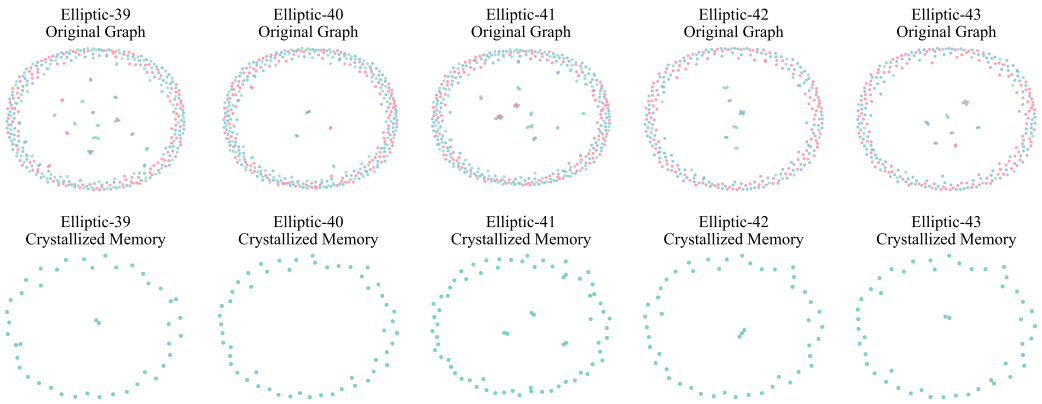


Figure 19: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part VII).

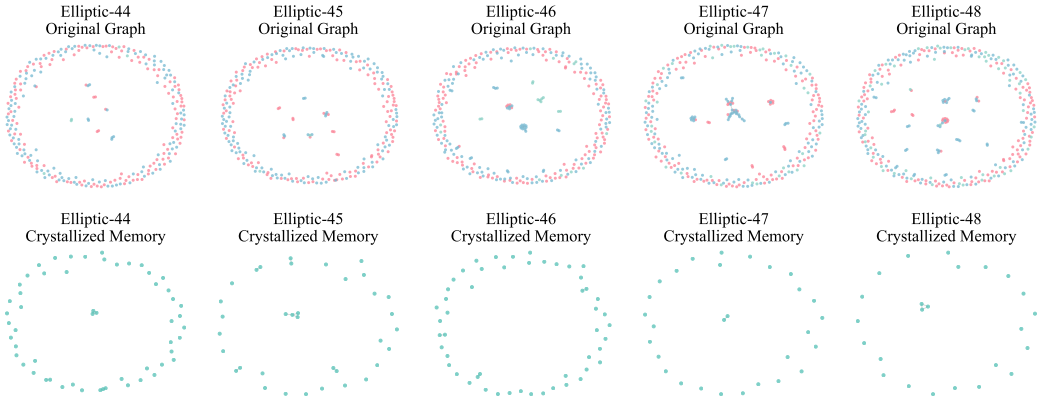


Figure 20: Structural visualization of the original graph and its crystallized memory on continuous-domain Elliptic graphs with GCN backbone (Part VIII).

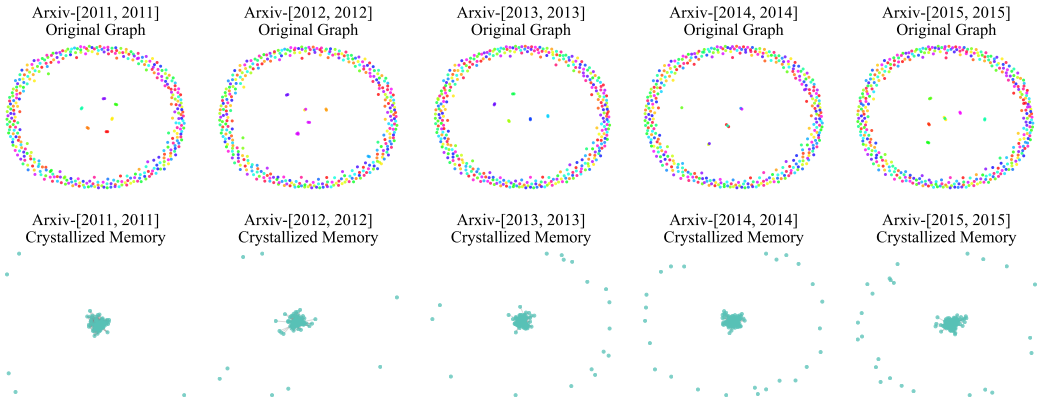


Figure 21: Structural visualization of the original graph and its crystallized memory on continuous-domain OGB-Arxiv graphs with GCN backbone (Part I).

Table 11: Node scaling efficiency evaluation with runtime (s) and GPU memory usage (MiB).

Methods	**Metric / #Node**	1000	2000	5000	10000	15000	20000	25000	30000	40000
GCAL	Runtime (s)	25.41	33	97.77	391.87	918.07	1694.24	OOM	OOM	OOM
	GPU Memory (MiB)	91.11	304.42	1754.81	6892.91	15472.69	27462.27	OOM	OOM	OOM
EMC	Runtime (s)	**1.75**	**2.72**	**25.89**	**210.52**	**407.95**	**593.20**	**792.00**	**1207.47**	**2213.20**
	GPU Memory (MiB)	**20.72**	**25.06**	**42.2**	**81.02**	**131.6**	**200.07**	**278.16**	**366.92**	**597.48**

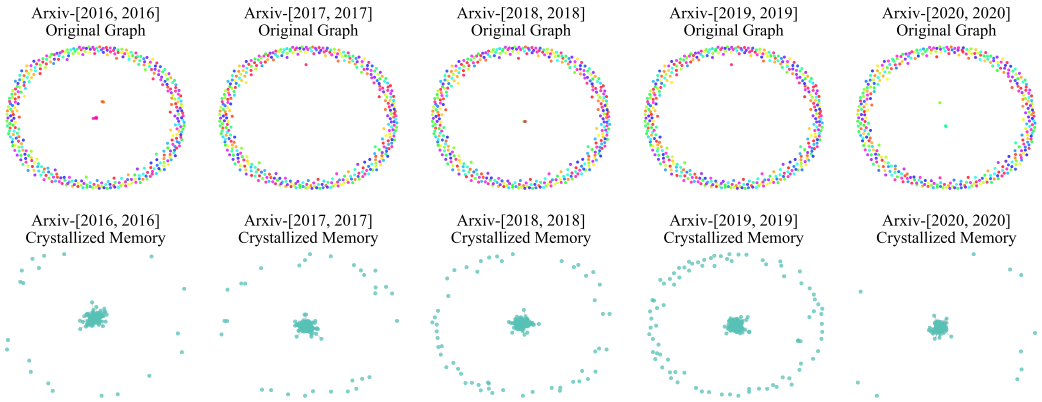


Figure 22: Structural visualization of the original graph and its crystallized memory on continuous-domain OGB-Arxiv graphs with GCN backbone (Part II).

Table 12: Domain scaling efficiency evaluation with runtime (s) and GPU memory usage (MiB).

Methods	**Metric / #Domain**	1×	2×	3×	4×	5×	6×	7×	8×
GCAL	Runtime (s)	430.04	1056.97	1713.34	2460.46	OOM	OOM	OOM	OOM
	GPU Memory (MiB)	6892.91	14239.83	21586.75	28933.67	OOM	OOM	OOM	OOM
EMC	Runtime (s)	**215.35**	**431.27**	**646.81**	**863.62**	**1079.35**	**1296.11**	**1513.47**	**1730.93**
	GPU Memory (MiB)	**81.02**	**85.92**	**90.80**	**95.71**	**104.52**	**118.21**	**131.90**	**145.59**

Table 13: Scaling efficiency of GPU memory consumption and crystallized memory storage.

#Domain	10	20	30	40	50	60	70	80
GCAL GPU (MiB)	6892.91	14239.83	21586.75	28933.67	OOM	OOM	OOM	OOM
EMC GPU (MiB)	81.02	85.92	90.80	95.71	104.52	118.21	131.90	145.59
Crystallized Memory (MiB)	4.88	9.76	14.65	19.53	24.41	29.30	34.18	39.07

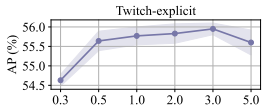


Figure 23: The sensitivity of hyperparameter α.

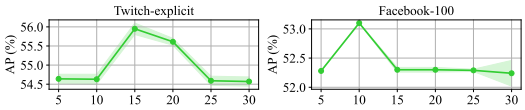


Figure 24: The sensitivity of hyperparameter β.

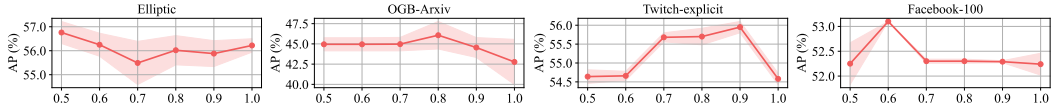


Figure 25: The sensitivity of hyperparameter τ.

Table 14: Performance comparison of EMC under different values of λ across multiple metrics, with corresponding runtime (s) and GPU memory usage (MiB). '↑' indicates higher is better, and '↓' indicates lower is better.

Dataset	λ	AP (↑)	TDG (↑)	TDA (↑)	Time (↓)	GPU (↓)
Twitch	0.01	55.95±0.15	55.56±0.13	55.83±0.14	74.64	854.97
	0.05	54.57±0.15	55.46±0.15	53.90±0.12	74.92	854.98
	0.1	54.57±0.15	55.46±0.15	53.89±0.13	74.90	854.97
	0.2	54.57±0.16	55.46±0.15	53.90±0.12	74.84	854.97
	0.3	54.56±0.16	55.46±0.15	53.88±0.12	75.09	854.11
	0.5	54.59±0.16	55.47±0.15	53.88±0.12	74.67	854.97
Facebook	0.01	53.10±0.00	53.49±0.27	52.30±0.04	18.84	3006.08
	0.05	53.10±0.00	53.45±0.35	52.29±0.02	18.84	3006.08
	0.1	53.10±0.00	53.45±0.35	52.30±0.03	18.92	3006.08
	0.2	53.10±0.00	53.46±0.33	52.30±0.04	18.92	3006.08
	0.3	53.10±0.00	53.36±0.53	52.30±0.03	18.98	3006.08
	0.5	53.10±0.00	53.50±0.24	52.30±0.04	18.93	3006.08

Dataset	λ	AP (↑)	TDG (↑)	TDA (↑)	Time (↓)	GPU (↓)
Elliptic	0.01	56.76±0.46	58.73±0.40	55.92±0.31	22.76	189.36
	0.05	56.18±0.25	58.44±0.24	55.96±0.28	23.38	189.36
	0.1	56.20±0.24	58.48±0.19	55.96±0.27	23.52	189.36
	0.2	56.23±0.24	58.53±0.19	55.98±0.23	23.19	189.36
	0.3	56.22±0.21	58.56±0.20	56.02±0.21	23.18	189.36
	0.5	56.25±0.24	56.03±0.19	58.58±0.23	22.98	189.36
Arxiv	0.01	46.07±1.70	47.11±1.80	43.52±1.25	24.84	391.02
	0.05	46.06±1.54	47.38±1.54	47.38±1.54	25.54	391.05
	0.1	46.00±1.59	47.29±1.62	47.29±1.62	25.73	390.98
	0.2	46.02±1.63	47.20±1.70	43.50±1.23	25.62	391.05
	0.3	43.91±1.65	47.16±1.73	43.50±1.23	26.12	391.05
	0.5	43.85±1.70	47.11±1.79	43.48±1.26	25.70	391.05