Chasing Label Shifters: A Change-Aware Framework for Dynamic Graph Node Classification

Junghoon Kim[1], **Seungyoon Choi**[1], **Hyunsung Kim**[1], **Chanyoung Park**[1]*
[1]KAIST
{jhkim611, csyoon08, hyunsung.kim, cy.park}@kaist.ac.kr

Abstract

Entities in real-world systems often evolve over time: users shift between information consumption patterns, researchers migrate between fields, and firms transition between financial risk states. When such systems are modeled as dynamic graphs, these transitions correspond to nodes whose class labels change between consecutive snapshots, which we call *shifters*. Correctly classifying shifters is often more consequential than classifying stable nodes with persistent labels, as detecting such transitions enables timely intervention in high-stakes settings. Yet, we identify a systematic failure mode across all major dynamic graph neural network architectures: shifter performance consistently lags behind stable nodes, with errors concentrated on the old label. Our structural analysis traces this failure to *embedding inertia*: neighborhood aggregation keeps shifter representations anchored to the old-class centroid, and this effect is strongest precisely where it is hardest to correct, namely for shifters whose neighborhoods remain aligned with the old class. Guided by this finding, we propose **CHASE**, a model-agnostic **CH**ange-**A**ware framework for **S**hifting nod**E**s that wraps any dynamic GNN with targeted components for detecting label shifts and overriding stale neighborhood signals. CHASE consistently improves shifter performance across all tested models and datasets (up to 114.9% and 202.5% improvement on accuracy and F1 score, respectively) while preserving stable-node performance. We additionally contribute three dynamic graph benchmarks with naturally shifting labels, filling a gap in existing resources. The source code is available at `https://github.com/jhkim611/CHASE`.

1 Introduction

In many real-world networks, the entities represented by nodes are not static [13]. Researchers shift their primary area of investigation, social media users alter their information consumption and sharing patterns, listeners migrate across musical genres, and firms transition between financial risk states. These transitions arise both from intrinsic behavioral change and from evolving relational context: a researcher's new collaborations may pull them toward a different subfield, a user's changing social circle may expose them to different content, or a firm's shifting transaction network may signal deteriorating creditworthiness. We refer to nodes whose class label changes between consecutive time steps as ***shifters***, and to those whose labels persist as ***stable nodes***.

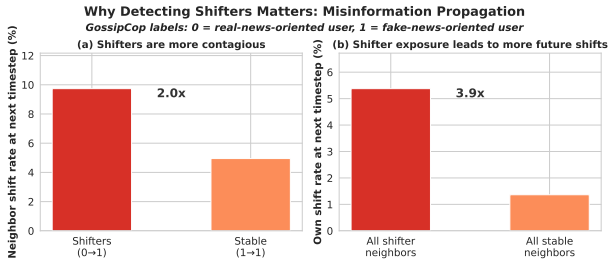


Figure 1: In misinformation propagation, recent *shifters* are more influential than stable entities, making their misclassification much more critical.

*Corresponding Author

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

Such shifters are often disproportionately consequential, making their correct classification more critical than that of stable nodes. In misinformation detection, for instance, users tend to orient toward either credible or unreliable content [1, 29], and the most operationally important nodes are not the stable consumers of fake news, but those who are actively transitioning toward fabricated articles. On the GossipCop dataset [25], where users are connected by co-consumption of news articles and labeled according to whether the majority of their consumed content is fabricated, we find that real-news neighbors of a recent shifter toward fake news are $2.0\times$ more likely to shift themselves than those of a stable fake-news user (See Fig. 1(a)). Moreover, this effect scales with the concentration of recent transitions in the neighborhood: among nodes with fake-news neighbors, those whose fake-news neighbors are predominantly recent shifters are $3.9\times$ more likely to shift than those whose fake-news neighbors are predominantly long-standing consumers (See Fig. 1(b)). In other words, recent shifters are *more influential* than established members of the opposing class, with greater potential to cause future shifts in their neighbors. Hence, accurately detecting these transitions would enable timely intervention, e.g., identifying at-risk neighbors before they shift, a capability that correct classification of stable nodes alone cannot provide.

Dynamic graph neural networks [15, 32] are the natural modeling framework for such settings, as they jointly capture evolving topology and node features over time. These models typically combine a spatial graph encoder with a sequence model to transfer information across discrete snapshots, yet their architectures do not specifically target nodes whose labels change. Further, existing dynamic node classification benchmarks either lack node features [9] or do not exhibit meaningful label change [20, 21], making systematic evaluation of shifter performance difficult.

Therefore, to study label shift in this context, we construct three dynamic graph benchmarks from real-world data, from which we identify a systematic failure mode shared by all major dynamic GNN architectures. Our datasets span academic co-authorship (**DBLP-S**), news co-consumption (**GossipCop**), and music co-listening (**Music**), where node labels, derived from observed behavioral patterns, shift naturally over time. Across all three datasets and seven models, we find that ***shifters are classified with substantially lower accuracy than stable nodes*** (See Fig. 2(a)–(b)). The failure is directional: when models misclassify shifters, they overwhelmingly predict the old label (See Fig. 2(c)). This pattern holds across all architectures examined and is not attributable to any single design choice.

To understand this failure, we conduct a structural analysis of shifter classification across all models (See Section 4). We identify a single architecture-independent cause, which we term ***embedding inertia***: neighborhood aggregation keeps shifter representations anchored to the old-class centroid, and this anchoring is strongest for shifters whose local homophily (the fraction of neighbors sharing the same class) still reflects the old class, i.e., shifters with low current homophily and high previous homophily. Our analysis reveals that correctly classified shifters exhibit positive centroid displacement toward their new class, while incorrectly classified shifters show no such movement.

Guided by these findings, we propose **CHASE** (**CH**ange-**A**ware framework for **S**hifting nod**E**s), a model-agnostic augmentation that wraps any dynamic GNN model without modifying its encoder. CHASE introduces three components, each addressing a specific diagnostic finding: a (a) ***homophily-guided contrastive loss*** that pulls shifter embeddings toward their new-class centroid with strength inversely proportional to current homophily, counteracting embedding inertia where it is most severe; a (b) ***change detection head*** that estimates the probability of a label shift from structural signals; and a (c) ***change-aware classifier*** that receives the change detection estimate as an explicit signal for when to deviate from prior predictions, enabling the model to override stale neighborhood representations.

We evaluate CHASE across seven models and three datasets. Shifter performance improves in all cases (up to 114.9% and 202.5% improvement on accuracy and F1 score, respectively), while stable-node performance is preserved or also enhanced. All components contribute monotonically in ablation. Together, these results establish label shift in dynamic graphs as a distinct, practically grounded, and largely underexplored problem, and demonstrate that it can be addressed through a model-agnostic framework without sacrificing performance on stable nodes.

Our contributions are as follows:

- **The shifter gap on real-world dynamic graph benchmarks.** We construct three dynamic graph datasets from real-world data spanning academic collaboration, misinformation consumption, and music listening, in which node labels, derived from observed behavior, shift naturally across snapshots. Using these benchmarks, we identify and quantify a universal performance disparity on

label-shifting nodes across seven dynamic GNN architectures, and attribute it to an architecture-independent mechanism (which we term *embedding inertia*) in which neighborhood aggregation keeps shifter representations anchored to the old class, with the effect concentrated on shifters whose local homophily still reflects the old class.

- **Analysis-driven framework design.** Each component of CHASE is motivated by a specific diagnostic finding, establishing a direct link between problem analysis and solution design. The contrastive loss corrects embedding inertia, the change detection head addresses *when* to deviate, and the change-aware classifier provides the capacity for *how* to deviate.
- **Model-agnostic augmentation with universal improvement.** CHASE improves shifter classification across all backbone–dataset combinations while preserving stable-node performance. The framework requires no architectural modification to the base encoder, demonstrating that the shifter gap is addressable through targeted change awareness.

2 Related works

2.1 Dynamic Graph Neural Networks

Dynamic graph neural networks extend static GNN architectures to graphs whose topology, node features, or both evolve over time. The dominant paradigm for discrete-time dynamic graphs couples a spatial graph encoder with a sequence model. Early approaches integrate graph convolutions with recurrent units: GCRN [24] applies GCN [17] followed by GRU [5] to learn spatiotemporal representations, while EvolveGCN [21] uses an RNN [5, 12] to evolve the GCN weight matrices themselves across snapshots. Attention-based alternatives replace the recurrent component with self-attention across snapshots: DySAT [23] applies structural GAT [28] attention within each snapshot and transformer-style attention across snapshots, and GCN-SE [9] learns squeeze-and-excitation attention weights over snapshot representations. More recent architectures explore implicit equilibrium formulations in IDGNN [33], time augmentation with deep propagation in TADGNN [26], and biologically inspired spiking activation patterns in SiGNN [3].

While the dynamic graph literature broadly acknowledges that node properties, including labels, may evolve over time [21, 20, 9], existing methods do not explicitly distinguish between *shifters* (nodes whose labels change between consecutive snapshots) and *stable nodes* (those whose labels persist), nor do they design components or evaluation protocols targeting label-shifting nodes. GCN-SE [9] is the most explicit example: it acknowledges time-varying class membership as a property of dynamic graphs and observes on synthetic data that attention weights shift when labels change. However, its real-world evaluation on a dataset with changing labels uses a graph that lacks node features, and the model itself contains no mechanism that detects or adapts to label shifts. To the best of our knowledge, ours is the first work to isolate shifter performance as an evaluation criterion and introduce architectural components specifically designed for this purpose.

At the benchmark level, existing evaluation datasets for dynamic graph learning do not foreground label shift. The Temporal Graph Benchmark (TGB) [14] includes a node-level task, but it is node affinity prediction, i.e., ranking a node's future interactions, which is distinct from classifying nodes whose categorical labels change over time. The benchmark itself notes the scarcity of graph datasets with dynamic node labels. Our work addresses this gap by constructing datasets where node labels are derived from observed behavior and shift naturally across snapshots, and by explicitly evaluating model performance on shifters versus stable nodes.

2.2 Distinction from Distribution Shift on Graphs

A growing body of work addresses distribution shift in graph learning, where training and test distributions diverge due to differences in graph structure, feature distributions, or latent environments [8, 31, 35, 30, 19]. These methods assume a *distributional* mismatch between training and deployment, and seek to learn invariant representations that generalize across contexts. On the other hand, the shifter problem is fundamentally different: it occurs *within* the observed graph across consecutive time steps, where a known node's ground-truth label changes. The failure mode is not unfamiliar structure but *temporal inertia*: the model's own representations and predictions resist updating when a node's class membership shifts. Addressing this requires detecting change and overriding learned continuity, rather than learning environment-invariant features.

Table 1: Dataset statistics. Shifter % is the fraction of nodes whose label differs from the previous snapshot, averaged across snapshots.

Dataset	Domain	# Nodes	# Edges	Snapshots (Split)	Classes	Label meaning	Shifter %
DBLP-S	Academic	34,717	394,385	14 (9/2/3)	2	Research area	2.62%
GossipCop	Social Media	6,091	2,000,390	37 (26/5/6)	2	>50% fake news consumption	4.06%
Music	Music Streaming	878	1,009,457	24 (15/4/5)	5	Dominant genre	18.77%

3 Preliminaries and Datasets

Problem Setting. We consider a discrete-time dynamic graph $\mathcal{G} = \{\mathcal{G}^{(1)}, ..., \mathcal{G}^{(T)}\}$, where each snapshot $\mathcal{G}^{(t)} = (V^{(t)}, E^{(t)}, X^{(t)}, y^{(t)})$ consists of a node set $V^{(t)}$, edge set $E^{(t)}$, node feature matrix $X^{(t)} \in \mathbb{R}^{|V^{(t)}| \times d}$ and label vector $y^{(t)}$. The node set may evolve across snapshots, where new nodes can appear and inactive nodes are retained but have zero degree [21, 33, 3]. For a node v present at consecutive snapshots $\mathcal{G}^{(t-1)}$ and $\mathcal{G}^{(t)}$, we define v as a *shifter* at snapshot $\mathcal{G}^{(t)}$ if $y_v^{(t)} \neq y_v^{(t-1)}$ and as *stable* otherwise. We denote the set of shifters at snapshot $\mathcal{G}^{(t)}$ as $\mathcal{S}^{(t)}$ and stable nodes as $\bar{\mathcal{S}}^{(t)}$, with $n_{\text{shifter}}^{(t)} = |\mathcal{S}^{(t)}|$ and $n_{\text{stable}}^{(t)} = |\bar{\mathcal{S}}^{(t)}|$. The T snapshots are divided into contiguous, non-overlapping training, validation, and test windows in temporal order. During training and validation, models have access to all available information (e.g., graph structure, node features, and labels) within the respective windows, but no test-time information is accessible at any point. At test time, the model predicts labels for all nodes in each test snapshot. Our analysis and method focus specifically on the disparity in classification performance between shifters and stable nodes.

Datasets. We ***construct three dynamic graph benchmarks from real-world data***, spanning distinct domains and label semantics (See Table 1). In each case, node labels are derived from observed behavior and shift naturally across snapshots, i.e., they are not synthetically injected. Detailed construction procedures and feature descriptions are provided in Appendix A.

- **DBLP-S.** (Source: DBLP-v11[2]) Nodes are researchers and edges represent co-authorship within each yearly snapshot. Each researcher is labeled by their primary research area in that year. Label shifts correspond to researchers whose main field of study changes between consecutive years.
- **GossipCop.** (Source: FakeNewsNet [25]) Nodes are social media users and edges connect users who co-consume the same news article within a weekly snapshot. A user is labeled as a fake-news consumer if more than 50% of the articles they engage with in that week are fabricated. Label shifts correspond to users whose consumption transitions from predominantly real to predominantly fake news, or vice versa.
- **Music.** (Source: LastFM-song-listens [18, 11] and the Million Song Dataset [2]) Nodes are listeners and edges connect users who listen to the same track within a monthly snapshot. Each listener is labeled by their dominant genre (out of five), determined by their listening history in that month. Label shifts correspond to listeners whose primary genre changes between consecutive months.

4 Diagnosing the Shifter Gap

We evaluate seven dynamic GNN models on all three datasets: GCN-GRU [24], EvolveGCN [21], DySAT [23], GCN-SE [9], IDGNN [33], TADGNN [26], and SiGNN [3]. For each node v at snapshot $\mathcal{G}^{(t)}$, we compute the following structural signals. ***Feature drift***: $\text{self_drift}_v^{(t)} = \|\mathbf{x}_v^{(t)} - \mathbf{x}_v^{(t-1)}\|$, the change in v's own features. ***Neighborhood drift***: $\text{nbr_drift}_v^{(t)} = \|\bar{\mathbf{x}}_{\mathcal{N}_v^{(t)}}^{(t)} - \bar{\mathbf{x}}_{\mathcal{N}_v^{(t)}}^{(t-1)}\|$, the change in the mean feature vector of v's neighbors ($\mathcal{N}_v^{(t)}$ is v's one-hop neighborhood at time t). ***Current homophily***: $h_{\text{curr}}(v,t) = |\{u \in \mathcal{N}_v^{(t)} : y_u^{(t)} = y_v^{(t)}\}|/|\mathcal{N}_v^{(t)}|$, the fraction of v's current neighbors sharing its current label. ***Previous homophily***: $h_{\text{prev}}(v,t) = h_{\text{curr}}(v, t-1)$, the fraction of v's previous neighbors sharing its old label. Meanwhile, to measure whether a node's learned representation has moved toward the correct class, we define ***centroid displacement***: $\Delta_c(v,t) = \text{sim}(\mathbf{h}_v^{(t)}, \mathbf{c}_{v,\text{new}}^{(t)}) - \text{sim}(\mathbf{h}_v^{(t)}, \mathbf{c}_{v,\text{old}}^{(t)})$, where sim denotes cosine similarity, $\mathbf{h}_v^{(t)}$ is the learned embedding of node v at time t, and $\mathbf{c}_{v,\text{new}}^{(t)}$ and $\mathbf{c}_{v,\text{old}}^{(t)}$ are the class centroids of the new (at time t) and old (at time $t-1$) labels of v, respectively, computed from stable-node embeddings.

The shifter gap. In Fig. 2, we observe that shifters are classified with substantially lower performance than stable nodes across all seven models and three datasets. More precisely, shifter accuracy and F1

[2] `https://dblp.org/`

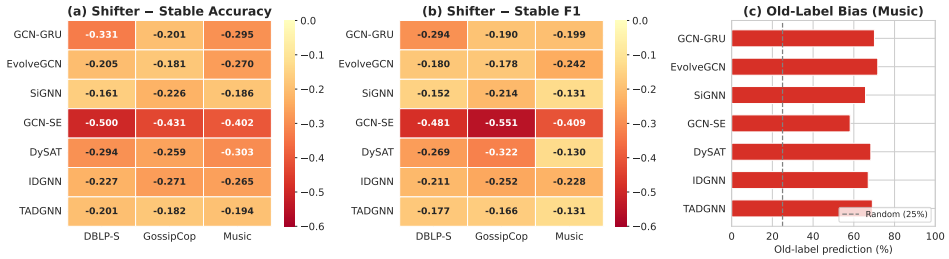


Figure 2: (a), (b) The shifter performance gap is universal across all models and datasets. (c) Shifter errors are predominantly stuck in the previous snapshot's label.

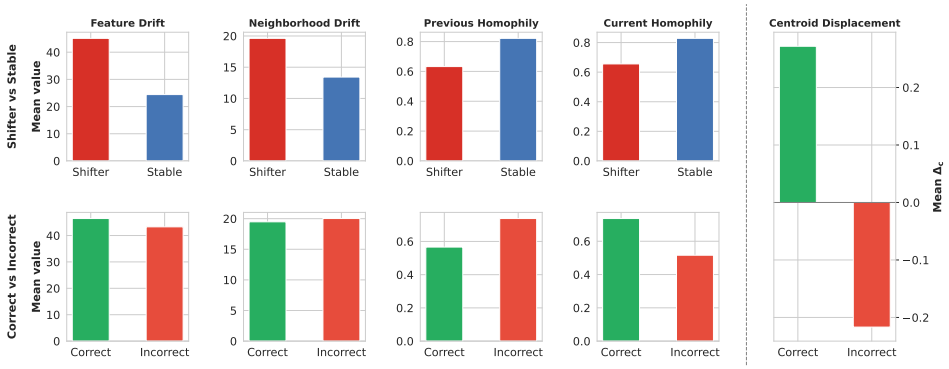


Figure 3: (Left) All four signals distinguish shifters from stable nodes (Top row), but only homophily signals meaningfully separate correctly from incorrectly classified shifters (Bottom row) (Mann-Whitney U test, $p \approx 0$ in all cases). (Right) Correctly classified shifters show positive centroid displacement while incorrectly classified ones do not, confirming embedding inertia. Averaged across models and datasets; per-model and per-dataset breakdowns in Appendix B.

lag behind their stable-node counterparts by up to 50%p and 55.1%p, respectively (See Fig. 2(a)–(b)). Moreover, these errors are not random: when models misclassify shifters, they overwhelmingly predict the *old* label. On the multiclass **Music** dataset (5 classes), up to 76% of shifter errors default to the old class, far exceeding the 25% rate expected under uniform error (See Fig. 2(c)). This old-label bias is consistent across all architectures and is not attributable to any single design choice.

Structural signals distinguish shifters from stable nodes. To understand the sources of this failure, we first investigate which observable signals separate shifters from stable nodes (See Fig. 3(Left), top row). We observe that feature drift and neighborhood drift are both significantly elevated for shifters. Shifters also exhibit lower current homophily and previous homophily than stable nodes. All four signals reliably identify *which* nodes are shifters (rank-biserial $|r|$ from Mann-Whitney U test up to 0.94).

Within shifters, homophily determines correctness. While these signals reliably identify shifters, a separate question is what determines whether a shifter is classified *correctly* (See Fig. 3(Left), bottom row). Here, the drift signals lose their discriminative power: feature drift and neighborhood drift do not meaningfully distinguish correctly from incorrectly classified shifters within the shifter subpopulation ($|r|< 0.10$). On the other hand, it turns out that the signals that *do* distinguish them are homophily measures. Across all models and datasets, incorrectly classified shifters have significantly lower current homophily, i.e., their neighborhoods have not yet adopted the new label, and significantly higher previous homophily, i.e., they were deeply embedded in the old class ($|r|$ up to 0.52).

Embedding inertia. These findings converge on a single failure mechanism. When a node shifts, neighborhood aggregation, the core operation of all GNN architectures, continues to produce representations anchored to the old class, because the local neighborhood still predominantly reflects the old label. This effect is strongest precisely where the homophily pattern is most unfavorable: low current homophily means the neighborhood provides no support for the new class, while high previous homophily means the old-class signal was strong before the shift. We term this mechanism *embedding inertia*. The centroid displacement analysis confirms this directly (See Fig. 3(Right)): correctly classified shifters exhibit positive Δ_c, indicating their embeddings have moved toward the new-class centroid, while incorrectly classified shifters show $\Delta_c \approx 0$ or negative, meaning their embeddings have not moved at all ($p \approx 0$ across all models and datasets). Taking a closer look regarding the old-label bias, on **Music**, each shifter's embedding is most similar to its predicted class

centroid (mean cosine similarity 0.83 vs. 0.65 to other classes). Shifters misclassified as their old class lie closest to that old class (0.84 vs. 0.78 to the true new class and 0.70 to the remaining three), whereas those misclassified as another class lie closest to that class (0.84 vs. 0.76 to the old class). Old-label errors, the majority of shifter errors, thus stem from embedding inertia toward the specific old class rather than from generic confusion.

This analysis yields three actionable insights for method design. First, embedding inertia is the core failure: shifter representations must be actively pulled toward the new class, with stronger correction where homophily is least supportive. Second, our observed structural signals effectively detect shifts, and should inform a dedicated detection mechanism whose output signals the rest of the model. Third, the old-label bias means that models default to the old class even when internal signals indicate change. As such, the classifier must receive an explicit change signal and the capacity to override prior predictions. These insights directly motivate the design of CHASE, described in the following section.

5 CHASE: CHange-Aware Framework for Shifting NodEs

CHASE is a model-agnostic augmentation that can wrap any dynamic GNN backbone without modifying its encoder. It replaces only the classifier head, leaving the base encoder's architecture untouched. At each snapshot $\mathcal{G}^{(t)}$, the backbone model produces embeddings $\mathbf{h}_v^{(t)}$ for each node as usual. Our proposed CHASE augments the classification pipeline with three targeted components (See Fig. 4), each motivated by a specific finding from Section 4.

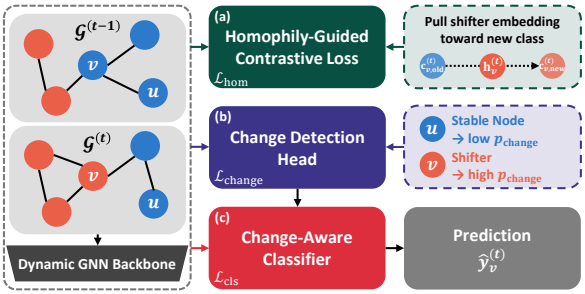


Figure 4: The CHASE architecture. Given snapshots $\mathcal{G}^{(t-1)}, \mathcal{G}^{(t)}$ and a dynamic GNN backbone, the **(a) homophily-guided contrastive loss** pulls shifter embeddings toward their new-class centroid. Then, the **(b) change detection head** estimates the probability p_{change} that a node has shifted. Finally, the **(c) change-aware classifier** receives change signals to guide its prediction.

5.1 Homophily-Guided Contrastive Loss

Embedding inertia keeps shifter representations anchored to the old-class centroid (See Section 4). To counteract this, we introduce a contrastive loss that pulls shifter embeddings toward their new-class centroid and pulls them away from the old. For each shifter v at training time, we compute class centroids $\mathbf{c}_{v,\text{new}}^{(t)}$ and $\mathbf{c}_{v,\text{old}}^{(t)}$ from stable-node embeddings, which serve as reliable references since their labels have not changed. The loss is margin-based and averaged over all shifters across all training snapshots. Let $\mathcal{S} = \{(v,t) : v \in \mathcal{S}^{(t)},\ t \leq \mathcal{T}_{\text{train}}\}$ denote the set of all shifters across training. Then:

$$\mathcal{L}_{\text{hom}} = \frac{1}{|\mathcal{S}|} \sum_{(v,t)\in\mathcal{S}} (1 - h_{\text{curr}}(v,t)) \cdot \max\big(0,\ m - (\text{sim}(\mathbf{h}_v^{(t)}, \mathbf{c}_{v,\text{new}}^{(t)}) - \text{sim}(\mathbf{h}_v^{(t)}, \mathbf{c}_{v,\text{old}}^{(t)}))\big), \quad (1)$$

where sim denotes cosine similarity, and m is a margin hyperparameter (set to 0.5 across all experiments). The weighting by $(1 - h_{\text{curr}}(v,t))$ applies stronger correction to shifters with low current homophily, precisely those for whom embedding inertia is most severe. This loss is applied only during training, where current labels are available.

5.2 Change Detection Head

All four structural signals ($\text{self_drift}_v^{(t)}$, $\text{nbr_drift}_v^{(t)}$, $h_{\text{curr}}(v,t)$ and $h_{\text{prev}}(v,t)$) reliably detect shifters (See Section 4), but this detection is not directly available to the classifier. The change detection head estimates the probability that a node has shifted:

$$p_{\text{change}}(v,t) = \sigma\big(\text{MLP}([\mathbf{h}_v^{(t)};\ \text{self_drift}_v^{(t)};\ \text{nbr_drift}_v^{(t)};\ h_{\text{proxy}}(v,t)])\big), \quad (2)$$

where our signals are concatenated. The term $h_{\text{proxy}}(v,t)$ is a prediction-based approximation of previous homophily: $h_{\text{proxy}}(v,t) = \text{mean}_{u\in\mathcal{N}_v^{(t-1)}} \hat{\mathbf{y}}_u^{(t-1)}[\arg\max \hat{\mathbf{y}}_v^{(t-1)}]$, i.e., the average predicted probability that v's previous neighbors share v's predicted class at $t-1$. As true labels

from the previous snapshot are not available during inference, we use this proxy consistently across training, validation, and test time, ensuring no label leakage. Note that current homophily h_{curr} is not used in the detection head: it requires current labels, which are the prediction target, and its role is already served by the contrastive loss weighting. The head is supervised with weighted binary cross-entropy: $\mathcal{L}_{\text{change}} = -\frac{1}{n}\sum_{v,t}\left[w^{+}s_v^{(t)}\log p_{\text{change}}(v,t) + (1-s_v^{(t)})\log(1-p_{\text{change}}(v,t))\right]$, where $s_v^{(t)} = 1[y_v^{(t)} \neq y_v^{(t-1)}]$ and $w^{+} = n_{\text{stable}}/n_{\text{shifter}}$. The output p_{change} is then passed to the classifier defined below as a soft shift signal. The head itself demonstrates considerable performance in detecting shifters, discussed in Appendix F.1.

5.3 Change-Aware Classifier

The analysis shows that when models misclassify shifters, they default to the old label even when internal representations contain signals of change. The change-aware classifier is designed to receive these signals explicitly and learn when to override prior predictions. It takes as input:

$$\hat{\mathbf{y}}_v^{(t)} = \text{MLP}\big([\mathbf{h}_v^{(t)};\ \mathbf{z}_{\text{self},v}^{(t)};\ p_{\text{change}}(v,t);\ \hat{\mathbf{y}}_v^{(t-1)}]\big), \tag{3}$$

where $\mathbf{h}_v^{(t)}$ is the base encoder embedding (neighbor-aggregated), $\mathbf{z}_{\text{self},v}^{(t)} = \text{ReLU}(\mathbf{W}_{\text{self}}\mathbf{x}_v^{(t)})$ is an aggregation-free projection of the node's own features, $p_{\text{change}}(v,t)$ is the change detection output, and $\hat{\mathbf{y}}_v^{(t-1)}$ is the model's prediction from the previous snapshot. Each input serves a distinct role. When p_{change} is low (stable node), all signals tend to agree and the classifier can rely on $\mathbf{h}_v^{(t)}$ and $\hat{\mathbf{y}}_v^{(t-1)}$. When p_{change} is high (likely shifter), $\mathbf{h}_v^{(t)}$ and $\hat{\mathbf{y}}_v^{(t-1)}$ may still point to the old class, but $\mathbf{z}_{\text{self},v}^{(t)}$ provides aggregation-free evidence and p_{change} signals the need to deviate. The MLP learns this interaction end-to-end, supervised with cross-entropy: $\mathcal{L}_{\text{cls}} = \frac{1}{n}\sum_{v,t} w_v^{(t)}\,\ell_{\text{CE}}(\hat{\mathbf{y}}_v^{(t)}, y_v^{(t)})$, where $w_v^{(t)} = \sqrt{(n_{\text{shifter}}^{(t)} + n_{\text{stable}}^{(t)})/(2\cdot n_{g(v,t)})}$ with $n_{g(v,t)} \in \{n_{\text{shifter}}^{(t)}, n_{\text{stable}}^{(t)}\}$ applies sqrt-dampened weights to address the rarity of shifters.

5.4 Training Objective

The total loss combines the three terms: $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{hom}}\cdot\mathcal{L}_{\text{hom}} + \lambda_{\text{change}}\cdot\mathcal{L}_{\text{change}}$, where $\lambda_{\text{hom}} = \lambda_{\text{change}} = 0.1$ across all experiments. CHASE is trained end-to-end with the backbone in a single run, with all parameters, including the base encoder, updated jointly under $\mathcal{L}$. During inference, only the change detection head and the change-aware classifier are used, and $\mathcal{L}_{\text{hom}}$ does not apply. All inputs at test time are derived from node features, graph structure, and the model's own previous predictions, *requiring no ground-truth labels from any test snapshot*.

5.5 Model-Agnostic Design

CHASE replaces only the classifier head of the backbone model, i.e., no architectural modification to the base encoder. The baseline configuration maps $\mathbf{h}_v^{(t)} \to \text{MLP} \to \hat{\mathbf{y}}_v^{(t)}$; CHASE augments the input to $[\mathbf{h}_v^{(t)}; \mathbf{z}_{\text{self},v}^{(t)}; p_{\text{change}}(v,t); \hat{\mathbf{y}}_v^{(t-1)}] \to \text{MLP} \to \hat{\mathbf{y}}_v^{(t)}$. Here, "model-agnostic" refers to the freedom to choose the backbone, not wrapping a frozen pre-trained model. All backbones use the same 2-layer MLP classifier, ensuring that performance differences are attributable to the CHASE components rather than classifier capacity. Full pseudocode and a complexity analysis are provided in Appendix C.

6 Experiments

6.1 Setup

We construct three dynamic graph datasets containing label shifts from real-world data spanning academic collaboration (**DBLP-S**), misinformation consumption (**GossipCop**), and music listening (**Music**) (See Table 1 and Appendix A) for evaluating CHASE. We compare against seven dynamic GNN baselines: GCN-GRU [24], EvolveGCN [21], DySAT [23], GCN-SE [9], IDGNN [33], TADGNN [26] and SiGNN [3]. We additionally include two static baselines, i.e., a standard GCN [17] and EERM [30] (a GNN method designed to address distribution shift), in two settings: *collapsed*, where all training snapshots are merged into a single graph, and *sequential*, where the model is trained on each snapshot in order. The sequential variant is additionally augmented with CHASE. For each

Table 2: Main results. Shifter (Sh.) and stable node (St.) accuracy (Acc) and F1 score for each backbone, without and with CHASE augmentation. On shaded rows, **bold** values indicate improvement over the base, with relative change shown as superscripts. All values are 5-seed means in percentage points; standard deviations are reported in Appendix E.

	DBLP-S				GossipCop				Music			
Model	Sh.Acc	Sh.F1	St.Acc	St.F1	Sh.Acc	Sh.F1	St.Acc	St.F1	Sh.Acc	Sh.F1	St.Acc	St.F1
GCN (collapsed)	73.0	76.5	96.0	97.1	48.6	22.1	69.7	40.1	38.1	23.7	64.5	22.9
GCN (seq)	71.8	75.5	96.1	97.1	52.8	25.0	75.6	45.3	46.3	29.0	73.6	45.0
+CHASE	**83.2**$^{+15.9\%}$	**85.9**$^{+13.8\%}$	**96.2**$^{+0.1\%}$	**97.2**$^{+0.1\%}$	**69.6**$^{+31.8\%}$	**60.1**$^{+140.4\%}$	**87.7**$^{+16.0\%}$	**76.9**$^{+69.8\%}$	**80.0**$^{+72.8\%}$	**78.6**$^{+171.0\%}$	**93.8**$^{+27.4\%}$	**89.5**$^{+98.9\%}$
EERM (collapsed)	73.5	77.4	96.2	97.3	47.5	20.0	69.6	44.5	39.2	22.4	64.7	25.6
EERM (seq)	73.6	77.3	96.3	97.3	52.5	23.4	76.1	47.1	44.0	24.6	73.8	45.4
+CHASE	**83.1**$^{+12.9\%}$	**85.6**$^{+10.7\%}$	96.1$^{-0.2\%}$	97.2$^{-0.1\%}$	**70.0**$^{+33.3\%}$	**61.0**$^{+160.7\%}$	**87.7**$^{+15.2\%}$	**76.9**$^{+63.3\%}$	**80.3**$^{+82.5\%}$	**79.3**$^{+222.4\%}$	**93.9**$^{+27.2\%}$	**89.5**$^{+97.1\%}$
GCN-GRU	63.9	68.4	97.0	97.8	58.2	34.1	78.3	53.1	50.1	35.2	79.5	55.1
+CHASE	**82.3**$^{+28.8\%}$	**85.0**$^{+24.3\%}$	96.3$^{-0.8\%}$	97.2$^{-0.6\%}$	**70.1**$^{+20.4\%}$	**64.4**$^{+88.8\%}$	**87.7**$^{+12.0\%}$	**77.5**$^{+46.0\%}$	**80.1**$^{+60.0\%}$	**79.2**$^{+124.8\%}$	**93.7**$^{+17.8\%}$	**89.6**$^{+62.4\%}$
EvolveGCN	75.7	79.2	96.2	97.2	70.0	60.1	88.1	78.0	59.9	52.2	86.9	76.3
+CHASE	**82.4**$^{+8.9\%}$	**85.1**$^{+7.5\%}$	96.2$^{+0.0\%}$	97.2$^{+0.0\%}$	**72.3**$^{+3.4\%}$	**64.4**$^{+7.1\%}$	87.4$^{-0.8\%}$	76.3$^{-2.1\%}$	**81.0**$^{+35.3\%}$	**80.2**$^{+53.7\%}$	**93.6**$^{+7.7\%}$	**88.8**$^{+16.3\%}$
SiGNN	71.3	75.3	87.4	90.6	49.1	28.4	71.7	49.8	71.3	68.0	89.9	81.1
+CHASE	**83.2**$^{+16.7\%}$	**85.8**$^{+13.9\%}$	**95.9**$^{+9.7\%}$	**97.0**$^{+7.1\%}$	**71.3**$^{+45.2\%}$	**65.5**$^{+130.6\%}$	**86.1**$^{+20.1\%}$	**73.9**$^{+48.4\%}$	**79.4**$^{+11.4\%}$	**78.2**$^{+15.0\%}$	**93.8**$^{+4.3\%}$	**89.6**$^{+10.5\%}$
GCN-SE	47.1	49.9	97.2	97.9	41.0	20.1	84.2	75.2	36.8	30.3	76.9	71.2
+CHASE	**79.5**$^{+68.8\%}$	**82.2**$^{+64.7\%}$	96.4$^{-0.8\%}$	97.3$^{-0.6\%}$	**69.3**$^{+69.0\%}$	**60.8**$^{+202.5\%}$	**86.8**$^{+3.1\%}$	**75.7**$^{+0.7\%}$	**79.1**$^{+114.9\%}$	**77.3**$^{+155.1\%}$	**94.0**$^{+22.2\%}$	**89.5**$^{+25.7\%}$
DySAT	61.2	67.1	90.7	94.0	49.8	29.0	75.8	61.2	39.9	34.8	70.2	47.9
+CHASE	**81.2**$^{+32.7\%}$	**84.1**$^{+25.3\%}$	**96.5**$^{+6.4\%}$	**97.4**$^{+3.6\%}$	**69.4**$^{+39.4\%}$	**60.6**$^{+109.0\%}$	**84.7**$^{+11.7\%}$	**70.5**$^{+15.2\%}$	**80.3**$^{+101.3\%}$	**79.3**$^{+127.9\%}$	**93.4**$^{+33.0\%}$	**88.8**$^{+85.4\%}$
IDGNN	73.3	75.9	96.0	97.1	58.0	46.3	85.1	71.4	67.4	66.1	93.9	88.9
+CHASE	**83.3**$^{+13.6\%}$	**86.0**$^{+13.2\%}$	**96.0**$^{+0.0\%}$	**97.1**$^{+0.0\%}$	**66.9**$^{+15.4\%}$	**58.2**$^{+25.8\%}$	**86.1**$^{+1.3\%}$	**73.9**$^{+3.4\%}$	**77.8**$^{+15.4\%}$	**76.0**$^{+15.0\%}$	93.7$^{-0.2\%}$	**89.1**$^{+0.1\%}$
TADGNN	76.6	79.9	96.7	97.6	65.9	50.6	84.1	67.2	69.1	62.1	88.4	75.1
+CHASE	**83.2**$^{+8.7\%}$	**85.9**$^{+7.5\%}$	96.0$^{-0.8\%}$	97.0$^{-0.6\%}$	**69.8**$^{+5.9\%}$	**61.4**$^{+21.3\%}$	**84.6**$^{+0.6\%}$	**70.4**$^{+4.7\%}$	**80.2**$^{+16.1\%}$	**79.0**$^{+27.2\%}$	**93.6**$^{+5.9\%}$	**89.2**$^{+18.7\%}$

backbone, we compare the baseline classifier against the same encoder augmented with CHASE. All models are trained with five random seeds and selected based on validation F1. We report accuracy and macro F1, computed separately for shifters and stable nodes, averaged across test snapshots. Full implementation details, hyperparameters, and hardware specifications are provided in Appendix D.

6.2 Main Results

Table 2 summarizes the main results. CHASE ***improves shifter performance across all 21 dynamic backbone–dataset combinations***, yielding universal improvement regardless of the domain or underlying encoder: up to 114.9% and 202.5% improvement on accuracy and F1 score, respectively. Gains are largest on **GossipCop** and **Music**, where baseline shifter performance is lowest, and even on **DBLP-S**, where baselines already achieve moderate shifter performance, CHASE still yields consistent improvements. Critically, ***these gains do not come at the expense of stable-node performance***: accuracy and F1 score on stable nodes are generally preserved or also improved substantially in all cases.

CHASE also improves the static baselines, confirming that the benefit is not contingent on temporal modeling. Notably, EERM, which targets distribution shift through invariant learning, fails to meaningfully improve upon GCN on its own. This reinforces that the shifter problem is fundamentally different from distributional mismatch (See Section 2.2). Dynamic baselines generally outperform their static counterparts, yet even the sequential GCN and EERM, which lack any cross-snapshot information flow, benefit immensely from CHASE augmentation.

Meanwhile, the shifter–stable gap narrows substantially under CHASE across all dynamic backbones and datasets (See Fig. 5), indicating that our framework ***reduces the performance disparity between the two node types***, a desirable property for real-world deployment where reliable classification of both shifters and stable nodes is important.

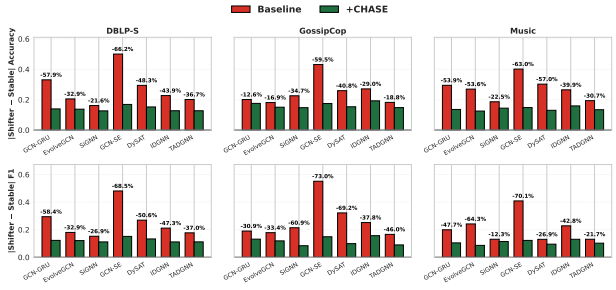


Figure 5: Performance gap between shifters and stable nodes is significantly reduced (up to -73.0%) under CHASE.

6.3 Ablation Study

To isolate the contribution of each CHASE component, we incrementally add components to the GCN-GRU baseline across all three datasets (See Table 3). Starting from the baseline MLP classifier on $[\mathbf{h}_v^{(t)}]$, we first add the contrastive loss $\mathcal{L}_{\text{hom}}$ (+Contr.), and then its $(1 - h_{\text{curr}})$

Table 3: Ablation study on GCN-GRU.

	DBLP-S		GossipCop		Music	
Config	Sh.Acc	Sh.F1	Sh.Acc	Sh.F1	Sh.Acc	Sh.F1
Baseline	63.9$_{\pm 1.0}$	68.4$_{\pm 1.1}$	58.2$_{\pm 4.7}$	34.1$_{\pm 12.4}$	50.1$_{\pm 7.0}$	35.2$_{\pm 13.6}$
+Contr.	65.9$_{\pm 0.6}$	69.9$_{\pm 0.6}$	60.4$_{\pm 4.8}$	45.5$_{\pm 3.4}$	53.6$_{\pm 7.0}$	44.6$_{\pm 7.4}$
+Hom.wt.	68.2$_{\pm 1.0}$	71.9$_{\pm 1.1}$	61.9$_{\pm 4.5}$	48.7$_{\pm 3.7}$	58.1$_{\pm 4.6}$	51.7$_{\pm 4.5}$
+Cls.	70.7$_{\pm 1.1}$	73.3$_{\pm 1.8}$	63.4$_{\pm 3.6}$	51.3$_{\pm 3.6}$	62.2$_{\pm 4.2}$	56.7$_{\pm 3.6}$
+Detect.	76.6$_{\pm 1.8}$	80.2$_{\pm 2.2}$	68.4$_{\pm 1.4}$	59.2$_{\pm 1.4}$	76.5$_{\pm 1.0}$	75.3$_{\pm 1.0}$
Full	**82.3**$_{\pm 0.5}$	**85.0**$_{\pm 0.6}$	**70.1**$_{\pm 0.9}$	**64.4**$_{\pm 1.5}$	**80.1**$_{\pm 1.0}$	**79.2**$_{\pm 1.0}$

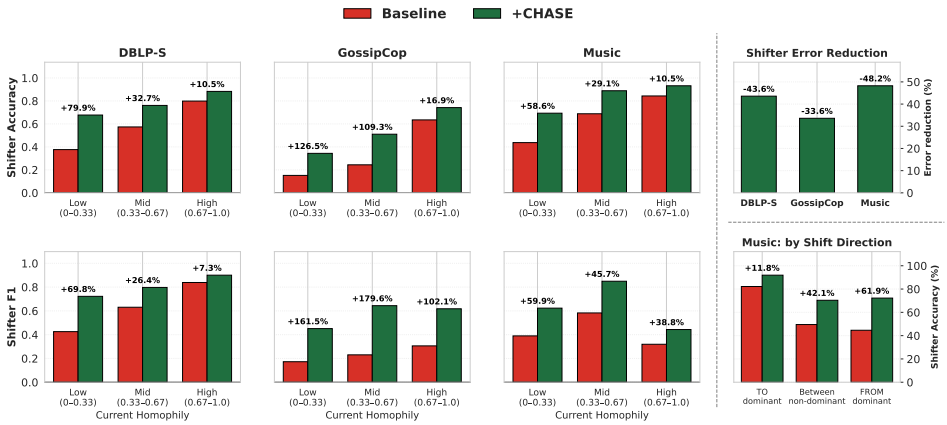


Figure 6: Performance improvements after CHASE augmentation, averaged across all models. (Left) Accuracy and F1 score gains are greater for nodes with lower current homophily. (Top right) Classification error on shifters is substantially reduced. (Bottom right) On the multiclass **Music** dataset, shifts *away from* the dominant class benefit much more than shifts *toward* it.

weighting (+Hom.wt.). We next augment the classifier input with $\mathbf{z}_{\text{self},v}^{(t)}$ and $\hat{\mathbf{y}}_v^{(t-1)}$ (+Cls.), then add the change detection head (and $\mathcal{L}_{\text{change}}$) whose output p_{change} is fed into the classifier (+Detect.), and finally add sqrt-dampened weighting to $\mathcal{L}_{\text{cls}}$ (Full).

Every component contributes monotonically to shifter performance. Particularly, ***the largest single jump occurs at +Detect.***, confirming the analysis in Section 4: the preceding steps give the classifier the *capacity* to deviate from old predictions (via $\mathbf{z}_{\text{self},v}^{(t)}$ and $\hat{\mathbf{y}}_v^{(t-1)}$), but without p_{change} it lacks the *signal* for when to do so. We additionally compare against a rule-based variant in Appendix F.1.

6.4 Further Analysis

CHASE improvement aligns with homophily. The analysis in Section 4 identifies low current homophily as the structural condition under which embedding inertia is most severe. If CHASE correctly addresses this mechanism, its gains should be concentrated more on these nodes. Fig. 6(Left) confirms this: across all backbones and datasets, CHASE's relative improvement on shifter accuracy and F1 is largest for nodes with the lowest current homophily and decreases monotonically as homophily increases. This alignment between the diagnostic finding and the empirical improvement validates the analysis-driven design of CHASE.

Error reduction toward the correct label. A natural concern is whether CHASE simply learns to avoid predicting the old label, rather than actively predicting the correct new one. On the binary datasets, these are equivalent, i.e., any move away from the old label is toward the correct one. On the multiclass **Music** dataset (5 classes), however, a model that merely suppresses the old-label prediction would still misclassify 75% of the time among the remaining classes. Fig. 6(Top right) shows that CHASE achieves substantial shifter error reduction on all three datasets, with greatest improvement on **Music**, confirming that the improvement reflects genuine classification rather than mere bias removal.

Larger gains on shifts away from the dominant class. On the **Music** dataset, class 0 (Rock genre) accounts for the majority of class labels. Under standard cross-entropy training, the loss is dominated by the dominant class, biasing learned representations in its favor. As a result, shifts *toward* the dominant class are naturally easier for baselines, while shifts *away from* it are harder. Fig. 6(Bottom right) confirms this: baseline accuracy is lowest for shifts away from the dominant class, moderate for shifts between non-dominant classes, and highest for shifts toward it. CHASE's performance gain follows the opposite ordering: largest for shifts away, moderate between non-dominant classes, and smallest toward. This indicates that CHASE provides the most benefit where embedding inertia is strongest.

CHASE directly repairs embedding inertia. We further examine embedding inertia across all shifters, beyond specific structural or class conditions. First, we bin shifters by their centroid displacement Δ_c before applying CHASE (See Table 4). Baseline accuracy drops sharply as inertia increases, and CHASE's gain is concentrated on the inertial bins (up to +40.1%p), with the smallest gain in the near-zero bin and none for non-inertial shifters. Second, we compare each shifter's Δ_c

Table 4: Shifter accuracy by baseline inertia (Δ_c), pooled over all backbones and datasets.

Baseline Δ_c	Baseline Acc. (%)	Gain from +CHASE (%p)
< -0.25 (most inertial)	3.0	+40.1
-0.25 to -0.05	15.9	+37.5
-0.05 to 0.05	61.1	+17.9
> 0.05 (non-inertial)	94.0	0.0

before and after adding CHASE to the same model. Shifters corrected by CHASE move from the old-class side to the new-class side (Δ_c: $-0.19 \rightarrow +0.09$), while already-correct shifters remain on the new-class side ($+0.27 \rightarrow +0.19$). Together, these results complement our analysis on embedding inertia driving shifter errors and show that CHASE improves performance by reversing it.

Additional analyses are provided in Appendix F, including further exploration on p_{change}, improvement stratified by number of shifts per node, per-snapshot consistency with comparison against retraining baselines that have access to ground-truth labels from previous test snapshots, robustness to incorrect $\hat{\mathbf{y}}^{(t-1)}$ and input noise, hyperparameter sensitivity, case studies, and computation overhead.

7 Conclusion

In this work, we identified a systematic failure mode shared by all major dynamic GNN architectures: *shifters*, nodes whose labels change between snapshots, are classified with substantially lower performance than stable nodes, with errors concentrated on the old label. We traced this to *embedding inertia*, where neighborhood aggregation anchors representations to the old class, and proposed **CHASE**, a model-agnostic framework that counteracts this effect through a homophily-guided contrastive loss, a change detection head, and a change-aware classifier. CHASE improves shifter performance across all tested backbone–dataset combinations while preserving stable-node performance. We additionally contributed three dynamic graph benchmarks with naturally shifting labels, addressing a gap in existing evaluation resources.

8 Limitations and Future Work

CHASE treats label change as binary: a node either shifted or it did not. In practice, many transitions are gradual, e.g., a user's consumption pattern may drift over several snapshots before crossing the labeling threshold, yet our current setup treats the node as stable until the threshold is crossed. This means the contrastive loss and detection head do not act on nodes approaching the decision boundary, potentially allowing embedding inertia to accumulate before the shift is registered. A *continuous transition intensity score* could allow CHASE to recognize and act on nodes that are mid-transition, rather than ignoring them until the label flips.

Such a formulation would also support an *anticipatory* extension: predicting which nodes will shift before the transition occurs, rather than correcting after the fact. CHASE's structural signals, e.g., declining current homophily and rising neighborhood drift, are plausible early indicators, but the current formulation targets detection at time t rather than prediction at $t+1$. Combining transition intensity scores with forward-looking detection is a natural next step toward early-warning systems for label change in dynamic graphs.

Finally, CHASE currently operates on discrete snapshot sequences, which fundamentally limits both the granularity of transition intensity measurement and the temporal resolution of anticipatory detection. Extending CHASE to continuous-time dynamic graph models such as TGN [22] and DyRep [27], where events are observed as they occur, is a promising next step. In this setting, drift signals could be computed over sliding time windows of recent interactions rather than between fixed snapshots, the homophily proxy could use each neighbor's most recent prediction rather than a snapshot-level aggregate, and the change detection head could output p_{change} per event rather than per snapshot, enabling finer-grained tracking of gradual transitions and more timely shift prediction.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (RS-2024-00406985) and the Ministry of Science and ICT (RS-2022-NR068758), and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2022-II220157).

References

[1] Eytan Bakshy, Solomon Messing, and Lada A Adamic. Exposure to ideologically diverse news and opinion on facebook. *Science*, 348(6239):1130–1132, 2015.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.

[3] Dong Chen, Shuai Zheng, Muhao Xu, Zhenfeng Zhu, and Yao Zhao. Signn: A spike-induced graph neural network for dynamic graph representation learning. *Pattern Recognition*, 158: 111026, 2025.

[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[5] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1724–1734, 2014.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Yingtong Dou, Kai Shu, Congying Xia, Philip S Yu, and Lichao Sun. User preference-aware fake news detection. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2051–2055, 2021.

[8] Shaohua Fan, Xiao Wang, Chuan Shi, Peng Cui, and Bai Wang. Generalizing graph neural networks on out-of-distribution graphs. *IEEE transactions on pattern analysis and machine intelligence*, 46(1):322–337, 2023.

[9] Yucai Fan, Yuhang Yao, and Carlee Joe-Wong. Gcn-se: Attention as explainability for node classification in dynamic graphs. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1060–1065. IEEE, 2021.

[10] Shihong Gao, Yiming Li, Yanyan Shen, Yingxia Shao, and Lei Chen. Etc: Efficient training of temporal graph neural networks over large-scale dynamic graphs. *Proceedings of the VLDB Endowment*, 17(5):1060–1072, 2024.

[11] Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer, 2012.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[13] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[14] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36:2056–2073, 2023.

[15] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1269–1278, 2019.

[19] Shuhan Liu and Kaize Ding. Beyond generalization: A survey of out-of-distribution adaptation on graphs. *arXiv preprint arXiv:2402.11153*, 2024.

[20] Xiaoxu Ma, Chen Zhao, Minglai Shao, and Yujie Lin. Hypergraph-based dynamic graph node classification. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[21] 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.

[22] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[23] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*, pages 519–527, 2020.

[24] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018.

[25] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3):171–188, 2020.

[26] Jiarui Sun, Mengting Gu, Chin-Chia Michael Yeh, Yujie Fan, Girish Chowdhary, and Wei Zhang. Dynamic graph node classification via time augmentation. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 800–805. IEEE, 2022.

[27] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[29] Jiaying Wu and Bryan Hooi. Decor: Degree-corrected social graph refinement for fake news detection. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2582–2593, 2023.

[30] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

[31] Qitian Wu, Fan Nie, Chenxiao Yang, Tianyi Bao, and Junchi Yan. Graph out-of-distribution generalization via causal intervention. In *Proceedings of the ACM Web Conference 2024*, pages 850–860, 2024.

[32] Yanping Zheng, Lu Yi, and Zhewei Wei. A survey of dynamic graph neural networks. *Frontiers of Computer Science*, 19(6):196323, 2025.

[33] Yongjian Zhong, Hieu Vu, Tianbao Yang, and Bijaya Adhikari. Efficient and effective implicit dynamic graph neural network. In *Proceedings of the 30th acm sigkdd conference on knowledge discovery and data mining*, pages 4595–4606, 2024.

[34] Hongkuan Zhou, Da Zheng, Israt Nisa, Vasileios Ioannidis, Xiang Song, and George Karypis. Tgl: A general framework for temporal gnn training on billion-scale graphs. *arXiv preprint arXiv:2203.14883*, 2022.

[35] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems*, 34:27965–27977, 2021.

Supplementary Material for

Chasing Label Shifters: A Change-Aware Framework for Dynamic Graph Node Classification

A Dataset Construction Details **15**

B Extended Diagnostic Analysis **16**

C Pseudocode and Computational Complexity **19**

C.1 Pseudocode 19

C.2 Computational Complexity 19

D Additional Experimental Setup **20**

D.1 Baselines 20

D.2 Implementation Details 20

E Full Numerical Results **21**

F Further Analysis **21**

F.1 Analysis on p_{change} 21

F.2 Improvement by Number of Shifts 22

F.3 Per-Snapshot Consistency and Comparison with Retraining Baselines 22

F.4 Hyperparameter Sensitivity 23

F.5 Case Study 23

F.6 Robustness to Incorrect Previous Predictions 23

F.7 Robustness to Input Noise 24

F.8 Computation Overhead 24

A Dataset Construction Details

We describe the construction of each dataset in detail, including data sources, node and edge definitions, label derivation, and feature extraction. Dataset statistics are summarized in Table 1 in the main text.

DBLP-S.

- **Source.** DBLP-v11 citation network dataset[3]. Our dataset is derived from GCN-SE's [9] DBLP-E, which was introduced as a label-changing dataset but lacked node features and used a fixed node pool across snapshots. We extend it with text-based node features and an evolving node pool.
- **Node set.** All authors who have at least one paper corresponding to one of the two label categories in the final snapshot. The node set evolves across snapshots: authors enter the graph when they first publish a relevant paper, and inactive authors are retained but become isolated. The dataset spans 14 yearly snapshots (2005–2018) with 34,717 unique nodes across all snapshots.
- **Edge construction.** An edge (u, v) exists in snapshot $G^{(t)}$ if authors u and v co-authored at least one paper in year t, weighted by the number of shared papers.
- **Label derivation.** Each author is labeled based on the `fos.name` (field of study) metadata of their papers: 0 for Computer Networks and 1 for Machine Learning. An author's label at snapshot $G^{(t)}$ is determined by whichever field accounts for the majority of their publications in that year.
- **Node features.** 768-dimensional feature vectors are obtained by extracting the [CLS] token from a pretrained BERT [6] model (`bert-base-uncased`) applied to the concatenated title and abstract of each author's papers (corresponding to one of the two label categories) in that year.

GossipCop.

- **Source.** FakeNewsNet [25] containing user interactions with real or fabricated news articles on celebrity gossip, constructed by combining the datasets from DECOR [29] and UPFD [7].
- **Node set.** Users who engaged with (retweeted) at least 3 articles in the dataset. The dataset spans 37 weekly snapshots (November 2017 – July 2018) with 6,091 unique nodes across all snapshots.
- **Edge construction.** An edge (u, v) exists in snapshot $G^{(t)}$ if users u and v both engaged with the same article within week t, weighted by the number of co-consumed articles.
- **Label derivation.** A user is labeled 1 (fake-news consumer) at snapshot $G^{(t)}$ if more than 50% of the articles they engaged with in that week are fabricated, and 0 otherwise.
- **Node features.** Each user's 768-dimensional feature vector is the mean of the embeddings of all articles they engaged with in that snapshot, where article embeddings are the [CLS] token from `bert-base-uncased` applied to the article text.

Music.

- **Source.** Constructed from the LastFM-song-listens [18, 11] dataset and the Million Song Dataset [2].
- **Node set.** All listeners active within the collection period. The dataset spans 24 monthly snapshots (January 2007 – December 2008) with 878 unique nodes across all snapshots.
- **Edge construction.** An edge (u, v) exists in snapshot $G^{(t)}$ if users u and v both listened to the same track within month t, weighted by the number of co-listened tracks.
- **Label derivation.** Each listener is labeled by their dominant genre in that month, determined by the most common genre across all tracks listened to in that month. The genre taxonomy consists of 5 classes: Rock (0), Metal (1), Electronic (2), Pop/Urban (3), and Niche/Diverse (4).
- **Node features.** 779-dimensional feature vectors composed of four groups: (i) 768-dimensional semantic features, obtained by averaging the [CLS] token from `bert-base-uncased` over all tracks' user-contributed tags, capturing the mood and style descriptors of the user's monthly listening; (ii) audio properties (duration, loudness, mode, tempo), averaged over all tracks listened to in that month; (iii) popularity signals (artist familiarity, artist hotness, song hotness), averaged over all tracks listened to in that month, capturing mainstream vs. niche listening; and (iv) behavioral features computed from all plays: log play count, artist diversity (unique artist count), repeat ratio (fraction of plays that are re-listens), and temporal spread (standard deviation of listening timestamps).

[3] `https://dblp.org/`

B Extended Diagnostic Analysis

Figs. 7– 10 contain per-model and per-dataset breakdowns of our analysis in Section 4, demonstrating that our findings are universal and not limited to specific cases.

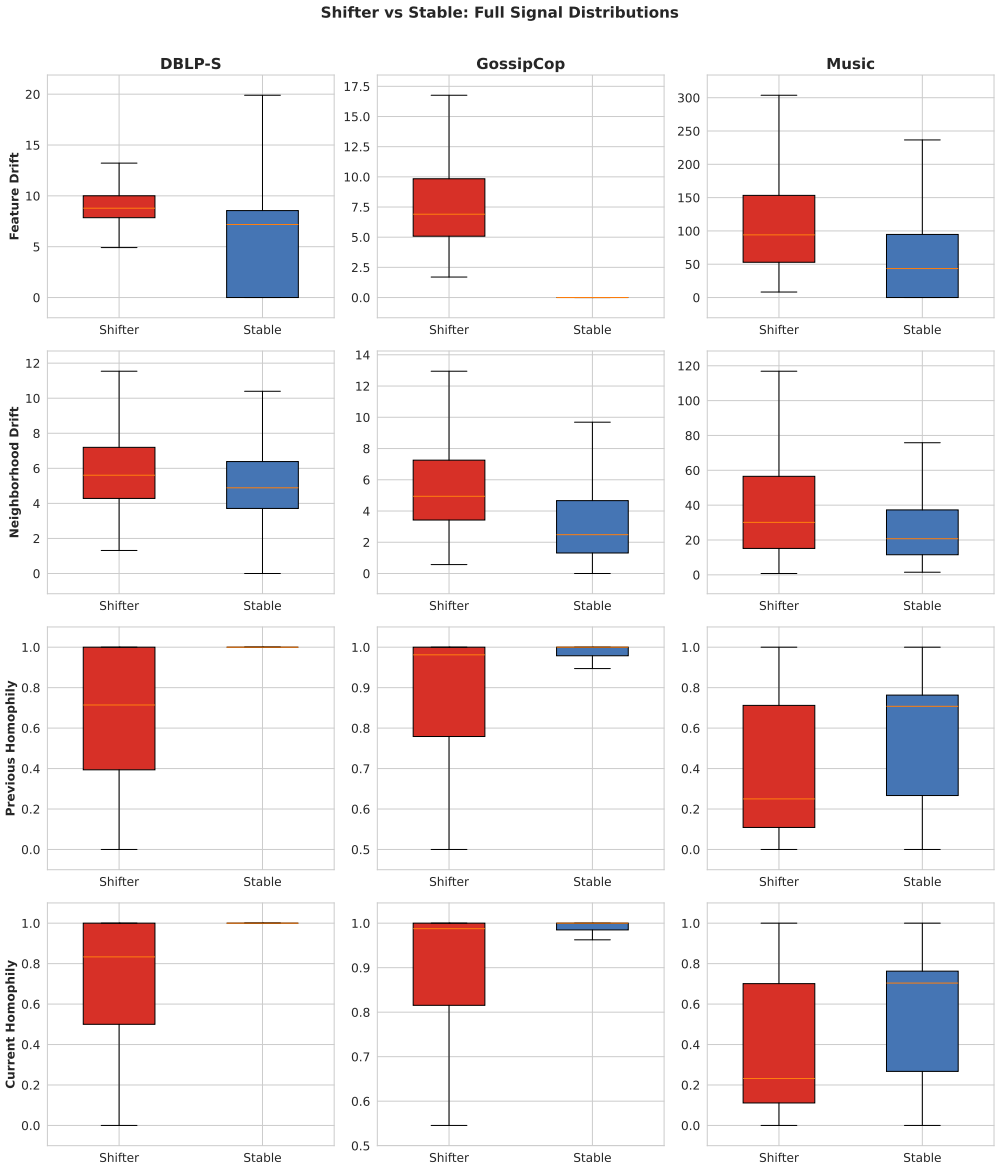


Figure 7: All four signals distinguish shifters from stable nodes, consistently across all three datasets.

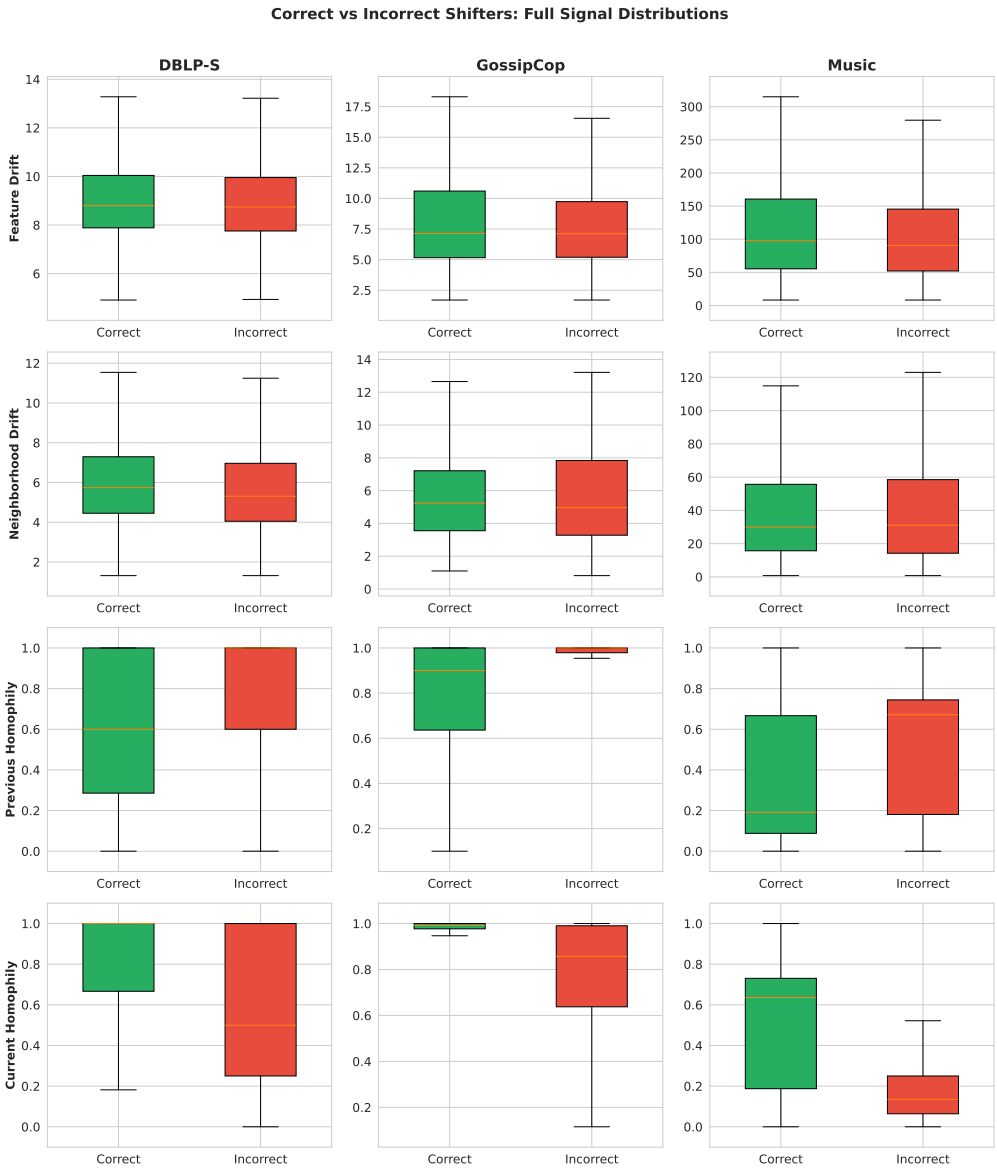


Figure 8: Homophily signals meaningfully separate correctly from incorrectly classified shifters while the drift signals lose their discriminative power, consistently across all three datasets.

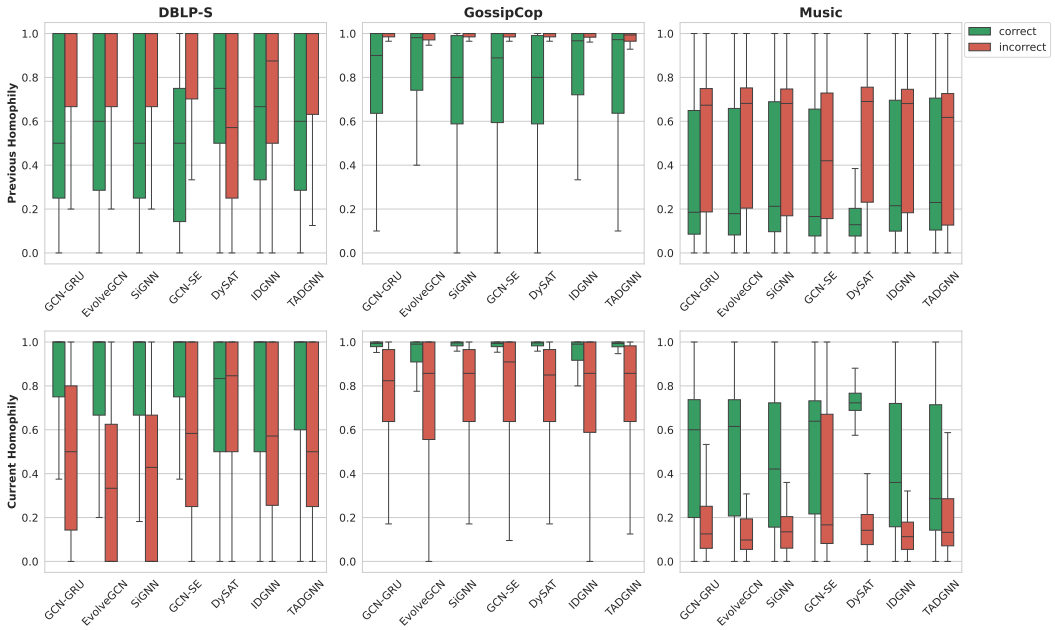


Figure 9: The homophily patterns (high current homophily and low previous homophily for correctly classified shifters, vice versa for incorrectly classified shifters) are consistent across all models and datasets.

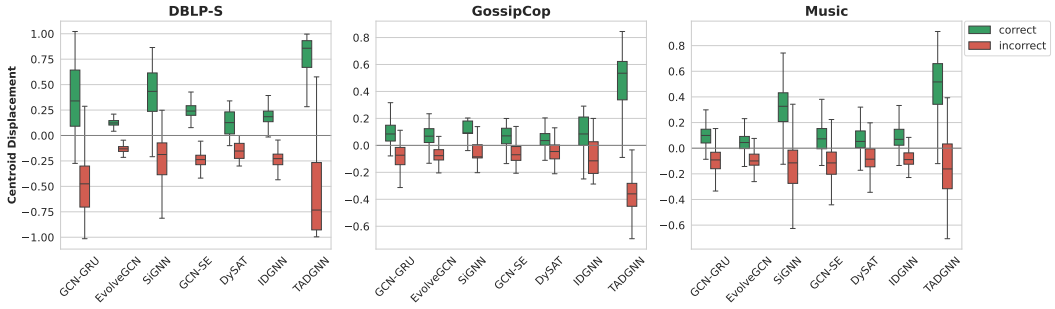


Figure 10: Correctly classified shifters generally show positive centroid displacement and incorrectly classified shifters generally display negative centroid displacement, consistently across all models and datasets.

Algorithm 1 CHASE forward pass at snapshot $G^{(t)}$

Require: Graph $\mathcal{G}^{(t)} = (V^{(t)}, E^{(t)}, X^{(t)}, y^{(t)})$, previous node features $X^{(t-1)}$, previous edges $E^{(t-1)}$, previous hidden state $\mathbf{h}^{(t-1)}$, previous predictions $\hat{\mathbf{y}}^{(t-1)}$

Ensure: Node predictions $\hat{\mathbf{y}}^{(t)}$, updated hidden state $\mathbf{h}^{(t)}$, change probabilities $\mathbf{p}^{(t)}_{\text{change}}$

1: $\mathbf{h}^{(t)} \leftarrow \text{BaseEncoder}(\mathbf{X}^{(t)}, E^{(t)}, \mathbf{h}^{(t-1)})$ {Any dynamic GNN — unmodified}
2: $\mathbf{z}^{(t)}_{\text{self}} \leftarrow \text{ReLU}(\mathbf{W}_{\text{self}}\, \mathbf{X}^{(t)})$ {Aggregation-free self-projection}
3: **for** each node $v \in V^{(t)}$ **do**
4: $\quad \text{self_drift}^{(t)}_v \leftarrow \|\mathbf{x}^{(t)}_v - \mathbf{x}^{(t-1)}_v\|$
5: $\quad \text{nbr_drift}^{(t)}_v \leftarrow \|\bar{\mathbf{x}}^{(t)}_{\mathcal{N}_v} - \bar{\mathbf{x}}^{(t-1)}_{\mathcal{N}_v}\|$
6: $\quad h_{\text{proxy}}(v,t) \leftarrow \text{mean}_{u \in \mathcal{N}^{(t-1)}_v} \hat{y}^{(t-1)}_u[\arg\max \hat{y}^{(t-1)}_v]$
7: **end for**
8: $\mathbf{p}^{(t)}_{\text{change}} \leftarrow \sigma\big(\text{MLP}_{\text{detect}}([\mathbf{h}^{(t)};\, \text{self_drift}^{(t)};\, \text{nbr_drift}^{(t)};\, h^{(t)}_{\text{proxy}}])\big)$
9: $\hat{\mathbf{y}}^{(t)} \leftarrow \text{MLP}_{\text{cls}}([\mathbf{h}^{(t)};\, \mathbf{z}^{(t)}_{\text{self}};\, \mathbf{p}^{(t)}_{\text{change}};\, \hat{\mathbf{y}}^{(t-1)}])$
10: **if** training **then**
11: $\quad$ Compute $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{hom}}$ (over shifters, weighted by $1 - h_{\text{curr}}$), $\mathcal{L}_{\text{change}}$
12: $\quad \mathcal{L} \leftarrow \mathcal{L}_{\text{cls}} + \lambda_{\text{hom}} \cdot \mathcal{L}_{\text{hom}} + \lambda_{\text{change}} \cdot \mathcal{L}_{\text{change}}$
13: **end if**
14: **return** $\hat{\mathbf{y}}^{(t)}, \mathbf{h}^{(t)}, \mathbf{p}^{(t)}_{\text{change}}$

C Pseudocode and Computational Complexity

C.1 Pseudocode

Algorithm 1 summarizes the CHASE forward pass for a single snapshot $G^{(t)}$.

In practice, the feature drift and neighborhood drift signals (lines 4–5) and the current homophily weights used in $\mathcal{L}_{\text{hom}}$ are precomputed once per training run over all snapshots; they are shown inline for clarity. The proxy for previous homophily (line 6) depends on the model's predictions and is computed at each forward pass.

C.2 Computational Complexity

Let $n = |V^{(t)}|$ denote the number of nodes, $e = |E^{(t)}|$ the number of edges, d the feature dimension, C the number of classes, and d_h the hidden dimension size.

Base encoder. The cost of the base encoder is unchanged by CHASE.

Self-projection. A single linear layer: $O(n \cdot d \cdot d_h)$.

Structural signals (precomputed once). Feature drift requires $O(n \cdot d)$ per snapshot. Neighborhood drift requires computing mean neighbor features at t and $t-1$, each $O(e \cdot d)$, followed by an $O(n \cdot d)$ difference. Current homophily weights require $O(e \cdot C)$ per snapshot. These are computed once over all training snapshots before training begins.

Homophily proxy (per forward pass). Requires one gather and a scatter-mean over previous edges: $O(|E^{(t-1)}| \cdot C)$ per snapshot.

Change detection head. An MLP over n nodes with input dimension $d_h + 3$: $O(n \cdot d_h)$.

Change-aware classifier. An MLP over n nodes with input dimension $d_h + d_h + 1 + C$: $O(n \cdot (d_h + C) \cdot d_h)$.

Contrastive loss (training only). Computing centroids from stable nodes: $O(n \cdot d_h)$. Per-shifter cosine similarity: $O(n^{(t)}_{\text{shifter}} \cdot d_h)$. Since $n^{(t)}_{\text{shifter}} \ll n$ in practice, this is negligible.

Overall. The CHASE overhead is $O(n \cdot d \cdot d_h + |E^{(t-1)}| \cdot C)$ per snapshot, dominated by the self-projection and homophily proxy. The one-time precomputation cost is $O(\mathcal{T}_{\text{train}} \cdot (e \cdot d + n \cdot d))$ over all training snapshots. In practice, both are a small fraction of the base encoder cost, as the encoder itself involves multi-layer message passing. We report wall-clock training times in Appendix F.8.

D Additional Experimental Setup

D.1 Baselines

We evaluate nine baseline architectures, grouped into static and dynamic models. For each dynamic backbone (and static backbone with sequential training), we compare the baseline classifier against the same encoder augmented with CHASE.

Static baselines.

- **GCN [17].** A graph convolutional network. We evaluate two training protocols: (i) *collapsed*, where all training snapshots are merged into a single graph by taking the union of edges across snapshots and the node features (and labels) in the last training snapshot, and (ii) *sequential*, where the model is trained on each training snapshot in order, processing them one at a time. The sequential variant is additionally augmented with CHASE.
- **EERM [30].** A GNN method designed to handle distribution shift through environment-based invariant learning. Evaluated with the same two training protocols and CHASE augmentation as GCN.

Dynamic baselines.

- **GCN-GRU.** A GCN model co-trained with a GRU on node embeddings. This is a variant of GCRN [24] adapted in [21].
- **EvolveGCN [21].** Uses an RNN [5, 12] to evolve the GCN weight matrices across snapshots. This allows the model to adapt its graph convolution parameters over time without relying on node-level recurrence.
- **DySAT [23].** Applies structural GAT attention within each snapshot and transformer-style self-attention across the temporal dimension.
- **GCN-SE [9].** Applies GCN independently per snapshot and learns squeeze-and-excitation attention weights to combine snapshot-level representations.
- **IDGNN [33].** An implicit-depth GNN that iterates message passing until convergence via a fixed-point formulation, avoiding oversmoothing by design.
- **TADGNN [26].** Augments the input graph with time-stamped edges and applies GCNII-style [4] deep propagation with initial residual connections.
- **SiGNN [3].** Combines GCN with biologically inspired BLIF spiking neurons in a dual-path temporal activation (sigmoid $\times$ spike), designed to capture both smooth and abrupt temporal dynamics.

D.2 Implementation Details

All models are trained using the Adam [16] optimizer. Training runs for a maximum of 500 epochs with early stopping based on validation F1 (patience = 10 epochs). We temporally split the snapshots into training, validation and test snapshots (See Table 1). For all backbones we follow the originally suggested hyperparameter values. For CHASE, the change detection head is a 2-layer MLP with hidden dimension 32, and the change-aware classifier is a 2-layer MLP with hidden dimension 128. The self-projection $\mathbf{W}_{\text{self}}$ maps from input feature dimension d to hidden dimension 128. We set $\lambda_{\text{hom}} = \lambda_{\text{change}} = 0.1$ and margin $m = 0.5$ across all CHASE-augmented experiments (See Appendix F.4 for a sensitivity analysis). All models (initial backbones and CHASE-augmented) use the same classifier architecture to ensure that performance differences are attributable to the CHASE components rather than classifier capacity. All experiments are executed based on PyTorch 2.2.1 with CUDA 11.8, and training is performed on a single NVIDIA RTX A6000 GPU (48GB) and Intel Xeon Gold 6226R CPU @ 2.90GHz (2 sockets × 16 cores, 64 threads total).

Table 5: Full results with standard deviations (5 seeds) corresponding to Table 2. All values in percentage points.

	DBLP-S				GossipCop				Music			
Model	Sh.Acc	Sh.F1	St.Acc	St.F1	Sh.Acc	Sh.F1	St.Acc	St.F1	Sh.Acc	Sh.F1	St.Acc	St.F1
GCN (collapsed)	$73.0_{\pm 0.2}$	$76.5_{\pm 0.4}$	$96.0_{\pm 0.0}$	$97.1_{\pm 0.0}$	$48.6_{\pm 0.9}$	$22.1_{\pm 4.9}$	$69.7_{\pm 1.3}$	$40.1_{\pm 6.5}$	$38.1_{\pm 0.7}$	$23.7_{\pm 3.2}$	$64.5_{\pm 0.7}$	$22.9_{\pm 0.3}$
GCN (sequential)	$71.8_{\pm 0.7}$	$75.5_{\pm 1.0}$	$96.1_{\pm 0.0}$	$97.1_{\pm 0.0}$	$52.8_{\pm 0.1}$	$25.0_{\pm 1.2}$	$75.6_{\pm 0.1}$	$45.3_{\pm 0.7}$	$46.3_{\pm 2.9}$	$29.0_{\pm 3.5}$	$73.6_{\pm 1.7}$	$45.0_{\pm 3.9}$
+CHASE	$83.2_{\pm 0.6}$	$85.9_{\pm 0.5}$	$96.2_{\pm 0.1}$	$97.2_{\pm 0.1}$	$69.6_{\pm 0.6}$	$60.1_{\pm 3.8}$	$87.7_{\pm 1.2}$	$76.9_{\pm 2.5}$	$80.0_{\pm 0.8}$	$78.6_{\pm 0.9}$	$93.8_{\pm 0.3}$	$89.5_{\pm 0.5}$
EERM (collapsed)	$73.5_{\pm 0.4}$	$77.4_{\pm 0.5}$	$96.2_{\pm 0.0}$	$97.3_{\pm 0.0}$	$47.5_{\pm 2.4}$	$20.0_{\pm 6.5}$	$69.6_{\pm 1.4}$	$44.5_{\pm 8.5}$	$39.2_{\pm 2.3}$	$22.4_{\pm 0.2}$	$64.7_{\pm 0.3}$	$25.6_{\pm 2.4}$
EERM (seq)	$73.6_{\pm 0.5}$	$77.3_{\pm 0.7}$	$96.3_{\pm 0.0}$	$97.3_{\pm 0.0}$	$52.5_{\pm 2.7}$	$23.4_{\pm 9.4}$	$76.1_{\pm 4.9}$	$47.1_{\pm 3.0}$	$44.0_{\pm 3.4}$	$24.6_{\pm 6.8}$	$73.8_{\pm 2.5}$	$45.4_{\pm 8.0}$
+CHASE	$83.1_{\pm 0.9}$	$85.6_{\pm 0.9}$	$96.1_{\pm 0.1}$	$97.2_{\pm 0.1}$	$70.0_{\pm 1.0}$	$61.0_{\pm 2.3}$	$87.7_{\pm 1.0}$	$76.9_{\pm 2.2}$	$80.3_{\pm 0.8}$	$79.3_{\pm 1.1}$	$93.9_{\pm 0.5}$	$89.5_{\pm 1.0}$
GCN-GRU	$63.9_{\pm 1.0}$	$68.4_{\pm 1.1}$	$97.0_{\pm 0.0}$	$97.8_{\pm 0.0}$	$58.2_{\pm 4.7}$	$34.1_{\pm 6.5}$	$78.3_{\pm 3.5}$	$53.1_{\pm 12.1}$	$50.1_{\pm 7.0}$	$35.2_{\pm 7.6}$	$79.5_{\pm 9.8}$	$55.1_{\pm 7.7}$
+CHASE	$82.3_{\pm 0.5}$	$85.0_{\pm 0.6}$	$96.3_{\pm 0.2}$	$97.2_{\pm 0.1}$	$70.1_{\pm 0.9}$	$64.4_{\pm 1.5}$	$87.7_{\pm 0.7}$	$77.5_{\pm 0.9}$	$80.1_{\pm 1.0}$	$79.2_{\pm 1.0}$	$93.7_{\pm 0.4}$	$89.6_{\pm 0.6}$
EvolveGCN	$75.7_{\pm 0.4}$	$79.2_{\pm 0.5}$	$96.2_{\pm 0.0}$	$97.2_{\pm 0.0}$	$70.0_{\pm 0.7}$	$60.1_{\pm 2.8}$	$88.1_{\pm 2.7}$	$78.0_{\pm 4.7}$	$59.9_{\pm 0.7}$	$52.2_{\pm 2.4}$	$86.9_{\pm 0.8}$	$76.3_{\pm 0.7}$
+CHASE	$82.4_{\pm 0.8}$	$85.1_{\pm 0.7}$	$96.2_{\pm 0.1}$	$97.2_{\pm 0.1}$	$72.3_{\pm 1.5}$	$64.4_{\pm 2.6}$	$87.4_{\pm 1.9}$	$76.3_{\pm 3.9}$	$81.0_{\pm 1.3}$	$80.2_{\pm 1.6}$	$93.6_{\pm 0.7}$	$88.8_{\pm 1.3}$
SiGNN	$71.3_{\pm 3.6}$	$75.3_{\pm 4.2}$	$87.4_{\pm 2.4}$	$90.6_{\pm 2.1}$	$49.1_{\pm 0.1}$	$28.4_{\pm 0.7}$	$71.7_{\pm 1.4}$	$49.8_{\pm 7.5}$	$71.3_{\pm 1.8}$	$68.0_{\pm 3.3}$	$89.9_{\pm 0.8}$	$81.1_{\pm 2.5}$
+CHASE	$83.2_{\pm 0.5}$	$85.8_{\pm 0.6}$	$95.9_{\pm 0.3}$	$97.0_{\pm 0.2}$	$71.3_{\pm 1.0}$	$65.5_{\pm 2.8}$	$86.1_{\pm 0.8}$	$73.9_{\pm 1.2}$	$79.4_{\pm 0.7}$	$78.2_{\pm 0.6}$	$93.8_{\pm 0.2}$	$89.6_{\pm 0.5}$
GCN-SE	$47.1_{\pm 0.9}$	$49.9_{\pm 2.0}$	$97.2_{\pm 0.0}$	$97.9_{\pm 0.0}$	$41.0_{\pm 0.8}$	$20.1_{\pm 2.1}$	$84.2_{\pm 0.9}$	$75.2_{\pm 4.7}$	$36.8_{\pm 2.3}$	$30.3_{\pm 2.8}$	$76.9_{\pm 5.8}$	$71.2_{\pm 2.4}$
+CHASE	$79.5_{\pm 1.4}$	$82.2_{\pm 1.2}$	$96.4_{\pm 0.2}$	$97.3_{\pm 0.2}$	$69.3_{\pm 0.9}$	$60.8_{\pm 1.4}$	$86.8_{\pm 1.6}$	$75.7_{\pm 3.3}$	$79.1_{\pm 1.0}$	$77.3_{\pm 1.0}$	$94.0_{\pm 0.1}$	$89.5_{\pm 0.4}$
DySAT	$61.2_{\pm 2.1}$	$67.1_{\pm 5.8}$	$90.7_{\pm 13.0}$	$94.0_{\pm 7.5}$	$49.8_{\pm 0.9}$	$29.0_{\pm 4.0}$	$75.8_{\pm 3.9}$	$61.2_{\pm 7.2}$	$39.9_{\pm 1.7}$	$34.8_{\pm 6.5}$	$70.2_{\pm 5.2}$	$47.9_{\pm 4.9}$
+CHASE	$81.2_{\pm 0.5}$	$84.1_{\pm 0.5}$	$96.5_{\pm 0.0}$	$97.4_{\pm 0.0}$	$69.4_{\pm 1.2}$	$60.6_{\pm 3.7}$	$84.7_{\pm 2.2}$	$70.5_{\pm 6.0}$	$80.3_{\pm 1.1}$	$79.3_{\pm 1.4}$	$93.4_{\pm 0.2}$	$88.8_{\pm 0.2}$
IDGNN	$73.3_{\pm 0.7}$	$75.9_{\pm 0.7}$	$96.0_{\pm 0.0}$	$97.1_{\pm 0.0}$	$58.0_{\pm 1.6}$	$46.3_{\pm 3.4}$	$85.1_{\pm 1.7}$	$71.4_{\pm 4.0}$	$67.4_{\pm 1.0}$	$66.1_{\pm 0.6}$	$93.9_{\pm 0.4}$	$88.9_{\pm 1.0}$
+CHASE	$83.3_{\pm 0.4}$	$86.0_{\pm 0.5}$	$96.0_{\pm 0.0}$	$97.1_{\pm 0.0}$	$66.9_{\pm 1.4}$	$58.2_{\pm 1.4}$	$86.1_{\pm 1.3}$	$73.9_{\pm 2.4}$	$77.8_{\pm 0.9}$	$76.0_{\pm 1.1}$	$93.7_{\pm 0.4}$	$89.1_{\pm 0.9}$
TADGNN	$76.6_{\pm 4.5}$	$79.9_{\pm 4.2}$	$96.7_{\pm 0.3}$	$97.6_{\pm 0.2}$	$65.9_{\pm 2.0}$	$50.6_{\pm 4.5}$	$84.1_{\pm 3.0}$	$67.2_{\pm 7.7}$	$69.1_{\pm 7.3}$	$62.1_{\pm 8.6}$	$88.4_{\pm 7.4}$	$75.1_{\pm 6.2}$
+CHASE	$83.2_{\pm 0.2}$	$85.9_{\pm 0.2}$	$96.0_{\pm 0.0}$	$97.0_{\pm 0.0}$	$69.8_{\pm 3.4}$	$61.4_{\pm 4.9}$	$84.6_{\pm 3.4}$	$70.4_{\pm 8.7}$	$80.2_{\pm 1.0}$	$79.0_{\pm 1.1}$	$93.6_{\pm 0.4}$	$89.2_{\pm 0.9}$

Table 6: AUROC (%) of p_{change} for shifter detection on test snapshots.

Backbone	**DBLP-S**	**GossipCop**	**Music**
GCN-GRU	93.1	97.0	79.5
EvolveGCN	87.2	97.6	77.9
SiGNN	83.7	96.9	79.1
GCN-SE	87.3	96.9	78.8
DySAT	90.4	97.0	79.2
IDGNN	85.6	97.4	79.1
TADGNN	84.3	97.1	81.1

Table 7: Shifter F1: CHASE vs. rule-based variant, averaged across backbones.

Method	**DBLP-S**	**GossipCop**	**Music**
Baseline	70.8	30.1	46.8
Rule-based	74.7	51.4	18.6
+CHASE	**84.9**	**62.2**	**78.4**

E Full Numerical Results

Table 5 reports the standard deviations corresponding to the main results in Table 2. Beyond improving mean performance, CHASE also reduces variance across seeds: baseline standard deviations are consistently higher than their +CHASE counterparts, indicating more stable training.

F Further Analysis

F.1 Analysis on p_{change}

The change detection head's performance. The change detection head is trained to distinguish shifters from stable nodes. Table 6 reports the AUROC of p_{change} for shifter detection on test snapshots across all backbones and datasets. Detection performance is consistently high (up to 97.4%), confirming that the structural signals we analyzed in Section 4 provide a reliable basis for identifying label-shifting nodes.

CHASE vs. rule-based variant. To further highlight the performance of our change-aware classifier, we additionally compare CHASE against a rule-based variant: given p_{change} and the previous prediction, when $p_{\text{change}} > 0.5$, this variant simply flips from the old label to the opposite class (binary datasets) or replaces it with a random non-old-label class (multiclass datasets). On the binary datasets **DBLP-S** and **GossipCop**, the rule-based variant also outperforms the unmodified baselines (See Table 7), further confirming that p_{change} is a meaningful signal for identifying shifters. However, on the multiclass **Music** dataset, it degrades considerably: knowing *when* a node has shifted is insufficient without knowing *what* it has shifted to. CHASE outperforms the rule-based variant in all cases, demonstrating that the change-aware classifier learns not only when to deviate from the previous prediction but how, by integrating p_{change} with other relevant signals, e.g., $\mathbf{z}^{(t)}_{\text{self},v}$.

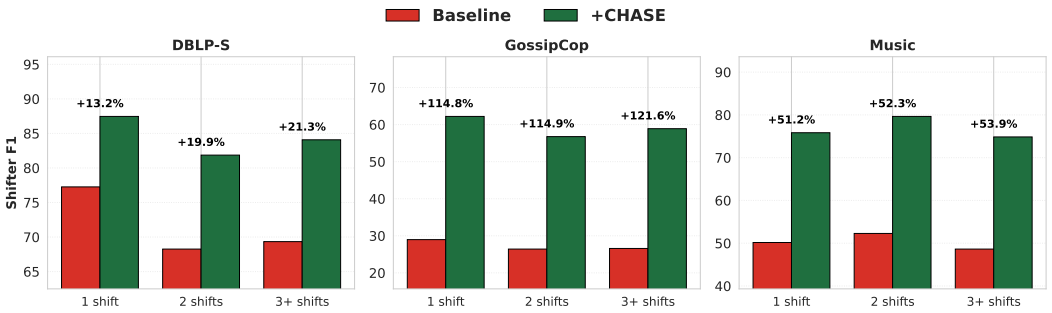


Figure 11: Relative improvement in shifter F1 (+CHASE over baseline), stratified by number of label shifts during test snapshots. Averaged across all seven dynamic backbones. CHASE's benefit increases with shift frequency across all datasets.

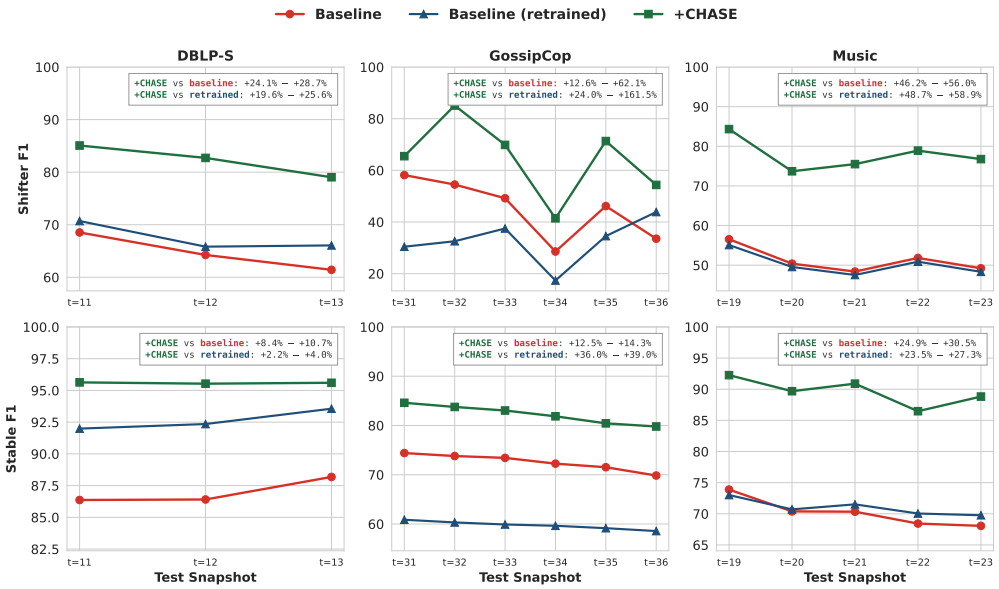


Figure 12: Shifter and stable F1 across test snapshots for baseline, baseline with retraining, and baseline + CHASE. CHASE outperforms both alternatives consistently, despite the retraining baseline having access to ground-truth labels from previous test snapshots. Averaged across all backbones.

F.2 Improvement by Number of Shifts

We stratify nodes by the number of label shifts they undergo during the test window (1, 2, or 3+) and measure CHASE's relative improvement on shifter F1, averaged across all seven dynamic backbones. Fig. 11 shows that baseline shifter F1 generally decreases with the number of shifts, confirming that frequently shifting nodes are harder to classify. CHASE's improvement increases accordingly: nodes that shift more frequently benefit the most from CHASE augmentation.

F.3 Per-Snapshot Consistency and Comparison with Retraining Baselines

We additionally compare CHASE against retraining baselines that re-train the backbone before each new test snapshot using all previously observed data (i.e., up to and including the test snapshot immediately before it), including ground-truth labels. This represents a strictly stronger baseline, as CHASE's structural signals rely only on its own predictions from the previous snapshot. Fig. 12 shows that CHASE consistently achieves the highest shifter and stable F1 across test snapshots on all datasets, with improvement distributed evenly across time rather than concentrated in particular snapshots. Notably, retraining does not always benefit baselines, and even when it does, CHASE still outperforms substantially.

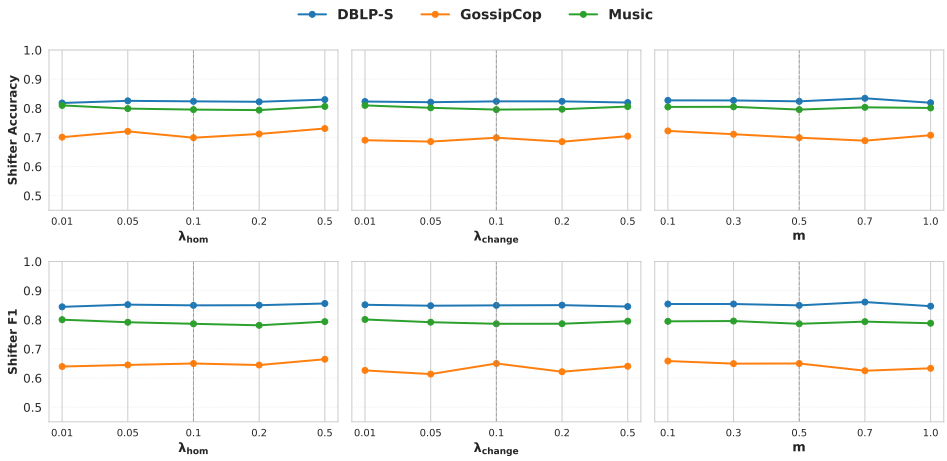


Figure 13: Hyperparameter sensitivity on GCN-GRU. Shifter accuracy and F1 remains stable across all tested values of λ_{hom}, λ_{change}, and m. Dashed lines indicate default values.

Table 8: Comparison of a shifter vs. a stable node.

	shifter (node 46) label: $0 \to 1$	stable (node 688) label: $0 \to 0$
self_drift	91.65	6.78
nbr_drift	121.67	5.70
h_{prev}	0.76	0.77
h_{curr}	0.05	0.80
p_{change}	0.83	0.04

F.4 Hyperparameter Sensitivity

CHASE introduces three hyperparameters: the contrastive loss weight λ_{hom}, the change detection loss weight λ_{change}, and the contrastive margin m (defaults: 0.1, 0.1, 0.5). We vary each independently while holding the others at their defaults, using GCN-GRU as the backbone across all three datasets. Fig. 13 shows that performance is stable across all tested ranges ($\lambda_{\text{hom}}, \lambda_{\text{change}} \in \{0.01, 0.05, 0.1, 0.2, 0.5\}$; $m \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$), indicating that CHASE is minimally sensitive to hyperparameter tuning.

F.5 Case Study

We trace two nodes through CHASE's signals on Music using GCN-GRU as the backbone. Table 8 demonstrates that both nodes instantiate the patterns in Fig. 3. Node 46 shows high drift on both signals together with low current and high previous homophily, exactly the condition under which shifters are misclassified, and its embedding is duly anchored to the old class (negative Δ_c), so the baseline predicts the old label. Based on these signals, CHASE's change detection head fires ($p_{\text{change}} = 0.83$), and the change-aware classifier moves the node across to the new class (Δ_c rises by +0.24, turning positive), predicting correctly. Node 688 matches the stable-node profile, triggers no change signal, and is left correct and untouched.

F.6 Robustness to Incorrect Previous Predictions

At test time, CHASE uses $\hat{\mathbf{y}}^{(t-1)}$ from the model's own predictions, which may be incorrect. We assess the impact by grouping shifters into those with correct vs. incorrect previous predictions and measuring the change in shifter F1 relative to the baseline, averaged across all backbones (See Table 9).

Table 9: Effect of incorrect $\hat{\mathbf{y}}^{(t-1)}$ on shifters: change in F1 (ΔF1, in percentage points) from baseline, mean h_{proxy}, and mean p_{change}. Averaged across backbones.

	DBLP-S			**GossipCop**			**Music**		
$\hat{\mathbf{y}}^{(t-1)}$	ΔF1	h_{proxy}	p_{change}	ΔF1	h_{proxy}	p_{change}	ΔF1	h_{proxy}	p_{change}
Correct	+27.7	0.707	0.750	+40.0	0.910	0.909	+30.9	0.442	0.664
Incorrect	-4.6	0.589	0.562	-8.0	0.711	0.696	+22.5	0.472	0.643

Table 10: Mean drift signal values for shifters (Sh.) and stable nodes (St.) at noise level $\alpha = 0.5$. Separation persists under both perturbation types.

	DBLP-S				**GossipCop**				**Music**			
	self_drift		nbr_drift		self_drift		nbr_drift		self_drift		nbr_drift	
Noise	Sh.	St.	Sh.	St.	Sh.	St.	Sh.	St.	Sh.	St.	Sh.	St.
Feature	10.61	8.47	3.86	3.22	9.42	4.85	1.35	0.16	178.05	149.63	42.32	21.47
Edge	9.25	5.97	4.39	4.01	8.06	0.57	3.33	2.49	118.19	66.68	38.89	31.46

When $\hat{\mathbf{y}}^{(t-1)}$ is correct, CHASE yields substantial improvement across all datasets, as expected. When $\hat{\mathbf{y}}^{(t-1)}$ is incorrect, the effect depends on the label space. On the binary datasets **DBLP-S** and **GossipCop**, performance degrades modestly: an incorrect previous prediction causes h_{proxy} to be computed against the wrong class for the target node, lowering its value and in turn reducing p_{change}, as lower h_{proxy} signals a less likely change. The model therefore shifts less aggressively from the (wrong) previous prediction, which for binary classification often results in a correct current prediction. On the multiclass **Music** dataset, CHASE still yields substantial improvement: with more than two classes, h_{proxy} is less affected by an incorrect previous prediction, so p_{change} remains comparable to the correct-prediction group. The classifier is still signaled to shift, and the additional inputs provide sufficient information to identify the correct new class.

F.7 Robustness to Input Noise

We evaluate CHASE's robustness under two types of input perturbation using GCN-GRU as the backbone. *Feature noise* adds Gaussian noise scaled to each feature dimension: $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t)} + \epsilon \cdot \text{std}(\mathbf{x}^{(t)}) \cdot \alpha$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and α is the noise level. *Edge noise* randomly drops a fraction α of edges per snapshot and adds an equal number of random edges, preserving the total edge count while shuffling neighborhood structure. We test noise levels $\alpha \in \{0.0, 0.1, 0.3, 0.5\}$.

Fig. 14 shows that CHASE is more robust than the baseline under both perturbation types across all datasets. Even at $\alpha = 0.5$, CHASE retains a substantial advantage: the structural drift signals (feature drift, neighborhood drift) remain meaningfully discriminative between shifters and stable nodes even under heavy noise (see Table 10), as they measure *relative change* between snapshots rather than absolute values. This robustness to noisy inputs is encouraging for real-world deployment, where node features and graph structure are often incomplete or imprecise.

F.8 Computation Overhead

CHASE adds a change detection head, a self-projection layer, and a change-aware classifier MLP on top of the backbone. We benchmark the resulting overhead on a single GPU (5 warmup + 20 measured epochs per configuration). Table 11 reports the per-epoch wall-clock and peak GPU memory overhead.

Per-epoch wall-clock. The absolute overhead is bounded at $\sim$0.24–0.50s per epoch across all configurations. Even over 100 training epochs, this translates to under a minute of additional wall-clock in the worst case.

Precomputation. Structural signals and homophily weights are computed once per training run over the training and validation snapshots: a one-time cost of 15.1s on DBLP-S (11 snapshots), 8.1s on GossipCop (31 snapshots), and 1.5s on Music (19 snapshots).

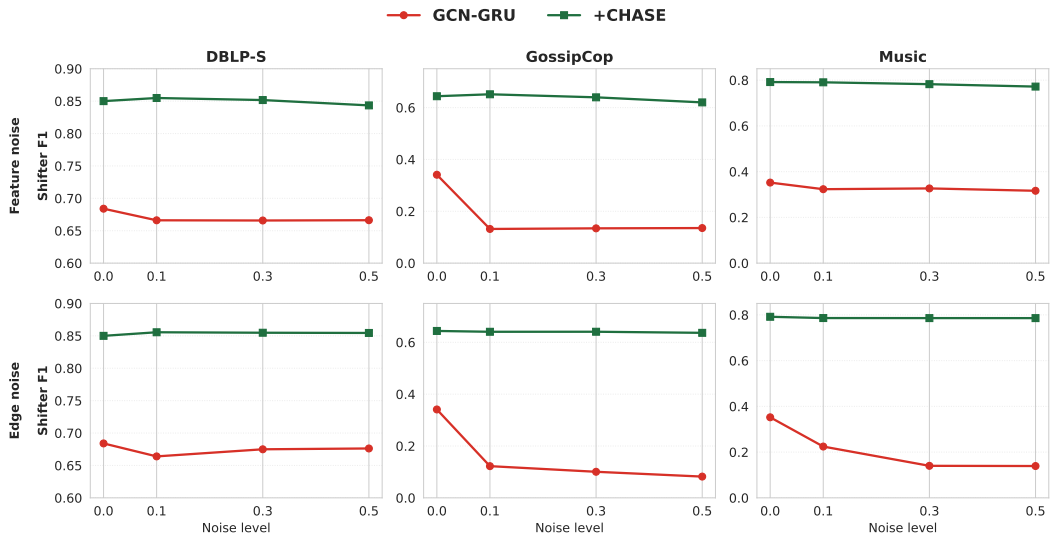


Figure 14: Shifter F1 under increasing feature noise (Top) and edge noise (Bottom) on GCN-GRU. CHASE 's degradation is minimal compared to the baseline across all noise levels and datasets.

Table 11: CHASE overhead: additional wall-clock time (seconds per epoch) and peak GPU memory (MB) over the baseline.

	DBLP-S		**GossipCop**		**Music**	
Backbone	Δs/ep	ΔMB	Δs/ep	ΔMB	Δs/ep	ΔMB
GCN-GRU	+0.24	+25.6	+0.35	+6.8	+0.24	+2.2
EvolveGCN	+0.40	+42.5	+0.50	+12.7	+0.22	+7.4
SiGNN	+0.25	+137.9	+0.37	+10.5	+0.24	+3.1
GCN-SE	+0.30	+42.9	+0.38	+18.4	+0.21	+3.1
DySAT	+0.24	+184.9	+0.43	+85.2	+0.19	+8.8
IDGNN	+0.14	+10.6	+0.29	+3.4	+0.23	+0.5
TADGNN	+0.33	+185.8	+0.36	+88.5	+0.23	+10.5

Scalability on massive graphs. To address whether CHASE scales beyond the three real-world datasets used in the main experiments (35K nodes maximum), we benchmark on a massive synthetic dynamic graph with 1,000,000 nodes, 10 snapshots (split 6/2/2), 2 classes, and average degree $\approx$ 20 per snapshot. Each snapshot's edges are independently sampled from a stochastic block model with 80% within-class assortativity. Node features are 768-dimensional class-conditional Gaussians, matching our initial datasets. On the GCN-GRU backbone, CHASE adds 15.2s per epoch (i.e., $\sim$25 min overhead at 100 epochs) and 635MB to peak GPU memory, while the one-time precomputation cost is 377.3s (8 snapshots). These results show that CHASE's overhead is modest relative to standard large-scale dynamic GNN training [34, 10], where per-epoch training times range from minutes to hours, highlighting its practicality.

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: Yes, the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Yes, we discuss limitations of our work in the appendix.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: The paper does not utilize theoretical results.

Guidelines:

- The answer [N/A] 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: Yes, we discuss implementation details and pseudocode in the appendix.

   Guidelines:

   - The answer [N/A] 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: Yes, we provide access to the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we specify training and test details in the appendix.

   Guidelines:

   - The answer [N/A] 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: Yes, we provide standard deviations for the main tables.

   Guidelines:

   - The answer [N/A] 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: Yes, we discuss such details in the appendix.

   Guidelines:

   - The answer [N/A] 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: Yes, we follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: We do not believe there are potential negative societal impacts of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] 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: Yes, we have adequately cited the relevant works and data sources.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: Yes, our source code is provided and well documented, and we discuss implementation details alongside dataset construction details in the appendix.

    Guidelines:

    - The answer [N/A] 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: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Our model does not involve LLMs.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.