Journal of Mathematical Finance and Risk Management
ISSN: 3080-9983 (Print), ISSN: 3080-9991 (Online) | Vol. 1, No. 2, 2026

Driftuard -HCL: Concept-Drift-Aware Continual Heterogeneous raph Contrastive Learning for Evolving Financial Fraud Detection

Kevin Liu

Department of Computer Science, University of Texas at Austin, Austin, TX, USA
kevinliutexas@outlook.com

Abstract: Financial fraud detectors are commonly trained as stationary classifiers even though transaction distributions, merchant populations, and attack typologies evolve. This paper presents DriftGuard-HCL, a continual heterogeneous graph-contrastive framework that combines a scalable typed graph-context encoder, semantic-group contrastive regularization, delayed-label replay, and a relation-support-aware drift gate. Each transaction is modeled on a temporal customer–category–merchant graph; causal customer and merchant neighborhood summaries are fused with a typed relation embedding through a bilinear interaction. A multi-signal detector combines Jensen–Shannon divergence over category and amount distributions, latent feature shift, and novel-relation mass. The detector switches a stability–plasticity ensemble from a frozen warm-up anchor toward a replay-regularized current model only after a calibrated shift. We evaluate the method with a strict prequential protocol on the public BankSim benchmark: 30 six-day snapshots, a one-snapshot label delay, five shared random seeds, and no future-label feature construction. In the native chronological stream, DriftGuard-HCL obtains 0.732 PR-AUC and 0.648 F1. In a controlled emergent-typology stress stream, it obtains 0.747 PR-AUC and 0.668 F1, improving over a continually updated heterogeneous contrastive model by 0.169 PR-AUC and 0.306 F1. The drift detector triggers once at typology restoration and never in the native stream. These results are reproducible from the accompanying code and raw result files; they support the value of drift-gated adaptation while not constituting evidence of production-bank performance.

Keywords: Concept drift; Continual learning; Financial fraud detection; Heterogeneous graphs; Contrastive learning; Delayed labels.

1. Introduction

Financial fraud detection is a nonstationary decision problem. Customer behavior changes seasonally, merchants enter and leave the ecosystem, payment categories expand, and attackers deliberately revise their operating patterns. Yet many graph-based detectors are trained once on a static graph and evaluated on a randomly partitioned sample. Such protocols can overstate deployment performance because future interactions influence representation learning and because delayed case confirmation is ignored. Concept-drift research has long distinguished abrupt, gradual, and recurring changes [4], [5], but those ideas have not been fully integrated with heterogeneous graph contrastive learning under delayed fraud labels.

Graph learning provides a natural representation for coordinated financial behavior. GCN, GraphSAGE, and graph attention models aggregate relational evidence [8]–[10], while relation-specific and heterogeneous architectures preserve edge semantics [11]–[13]. Dynamic graph models add temporal state [14]–[17]. Fraud-specific methods further address camouflage and class imbalance [22], [23]. A recent strong example is Audit-HCL, which positively demonstrates that heterogeneous structural views and temporal contrastive learning can uncover collusive financial fraud under scarce labels [1]. Its design motivates the present work, while leaving a distinct deployment question: how should a detector decide when to adapt, how strongly to retain its earlier decision function, and how to do so when labels arrive after the transaction distribution has already changed?

We answer that question with DriftGuard-HCL. Rather than executing expensive whole-graph message passing for every streaming update, the method constructs causal typed neighborhood summaries from a customer–category–merchant graph. The summaries are inductive: customer and merchant identifiers are not embedded, so the model cannot solve the benchmark through endpoint memorization. A semantic-group contrastive objective masks customer, merchant, and relation views as coherent units instead of perturbing arbitrary scalars. Continual learning uses balanced replay and one-snapshot delayed labels. A multi-signal detector identifies changes in relation support, amount distributions, and feature means, then gates a frozen-anchor/current-model ensemble.

The contribution is fourfold. First, the paper formulates delayed-label fraud detection as prequential learning on an evolving typed transaction graph. Second, it introduces a relation-support-aware drift score that explicitly reacts to newly appearing transaction categories. Third, it couples drift detection to a stability–plasticity ensemble rather than indiscriminately updating prediction weights at every snapshot. Fourth, it reports five-seed, code-generated results on both the unmodified BankSim chronology and a fully disclosed emergent-typology stress stream. All tables and figures in this paper are derived from the supplied CSV files; no experimental value is manually inserted.

2. Related work

A. Graph-Based Fraud Detection

The Elliptic study established a widely used transaction-graph benchmark for anti-money-laundering research [3]. CARE-GNN learns to filter camouflaged neighbors [22],

whereas PC-GNN balances neighborhood sampling under severe label skew [23]. LaundroGraph uses self-supervision to reduce dependence on labels [24]. These methods show that relational structure is valuable, but most evaluations assume that the training distribution is fixed or that all labels required for an update are immediately available. DriftGuard-HCL instead evaluates every test snapshot before learning from its labels.

B. Heterogeneous and Temporal Graph Learning

R-GCN assigns relation-specific transformations [11]; HAN uses hierarchical attention over metapaths [12]; and HGT parameterizes attention by node and edge types [13]. EvolveGCN, DySAT, TGAT, and TGN model graph evolution through recurrent parameters, temporal self-attention, functional time encoding, or node memories [14]–[17]. These architectures are expressive, but repeated whole-graph updates can be costly in high-throughput scoring. The proposed encoder is intentionally lighter: it compresses each endpoint's causal typed neighborhood into fixed-dimensional statistics, then learns nonlinear typed projections. This sacrifices fine-grained multi-hop message passing in exchange for inductive, transaction-level updates and a transparent leakage boundary.

C. Contrastive and Continual Adaptation

Graph contrastive learning derives supervision from agreement across views. GraphCL studies augmentation policies [18], DGI maximizes local–global mutual information [19], MVGRL contrasts diffusion and adjacency views [20], and HeCo aligns network-schema and metapath views in heterogeneous graphs [21]. Continual learning addresses catastrophic forgetting through parameter regularization, exemplars, constrained gradients, or replay [25]–[28]. In data streams, ADWIN and adaptive random forests provide statistically motivated adaptation mechanisms [6], [7]. DriftGuard-HCL combines these lines: typed view perturbations regularize representation learning, replay preserves rare fraud evidence, and a calibrated distributional gate controls the anchor/current mixture.

3. Problem formulation

Let the stream be partitioned into T ordered snapshots. Snapshot t is a heterogeneous bipartite multigraph G_t=(C_t,M_t,E_t,R), where C_t and M_t are customer and merchant nodes, R is the set of transaction categories, and each edge e_i=(c_i,r_i,m_i,x_i,y_i) carries observed covariates x_i and an eventually available binary fraud label y_i. At scoring time, y_i is unknown. Labels from snapshot t become available only before scoring t+1. Feature construction for e_i may use raw attributes of e_i, unlabeled topology within t, and history from snapshots less than t, but never labels or future edges.

The goal is to output probability p_i while maximizing ranking and thresholded detection quality over test snapshots. Because the fraud prevalence is approximately 1.21%, PR-AUC is treated as the primary ranking metric; ROC-AUC, F1, and recall among the top 1% of alerts are also reported. The emphasis on PR-AUC follows the known limitations of ROC curves in strongly imbalanced settings [29].

4. Driftguard-hcl

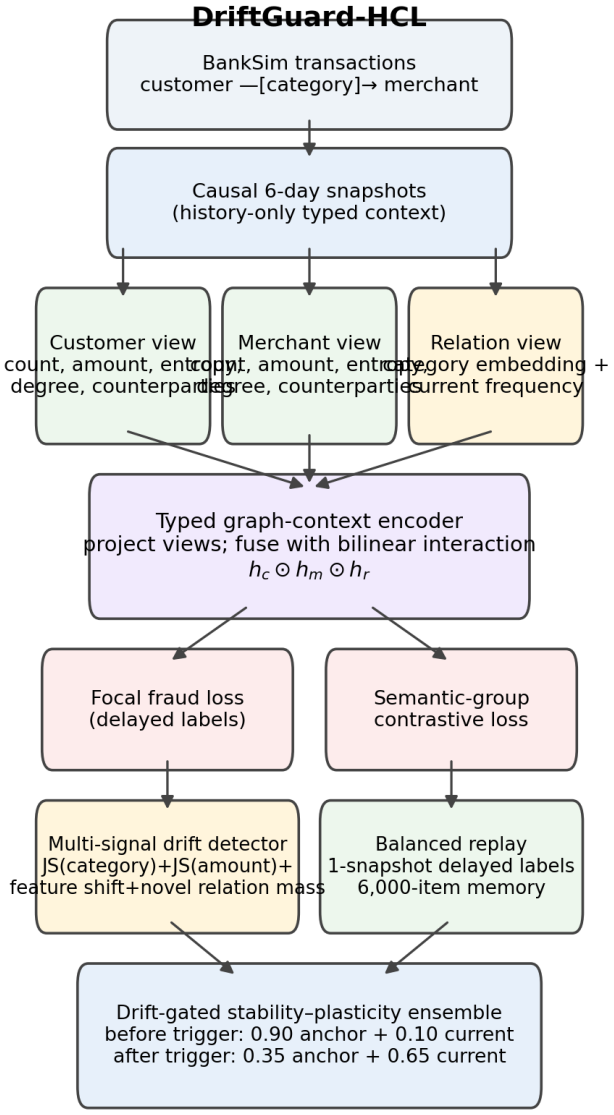

Fig. 1 DriftGuard-HCL processing path. The encoder uses causal typed graph context, contrastive regularization, delayed-label replay, and a drift-gated anchor/current ensemble.

A. Causal Typed Graph Context

For every transaction, the preprocessing stage computes 19 numeric variables. The customer view contains historical transaction count, amount mean and standard deviation, amount deviation, unique-merchant ratio, category entropy, and current-snapshot degree. The merchant view mirrors these quantities with customer diversity. The relation view contains a learned category embedding and the category's unlabeled frequency in the current snapshot. Historical statistics are updated only after all features in a snapshot have been emitted, which prevents within-snapshot label or future-history leakage. Continuous variables are standardized with warm-up statistics only.

B. Typed Encoder and Bilinear Interaction

Let x_i^c, x_i^m, and r_i denote customer, merchant, and relation inputs. Type-specific multilayer perceptrons produce h_i^c and h_i^m; a learned relation projection produces h_i^r. The interaction term is an elementwise three-way product, which acts as a low-cost typed compatibility operator:

$$u_i = h_i^c \odot h_i^m \odot h_i^r \quad (1)$$

The final representation z_i is obtained by concatenating raw transaction context, h_i^c, h_i^m, the category embedding, u_i, age, and gender embeddings, followed by a normalized two-layer encoder. Customer and merchant IDs are deliberately omitted. The graph-context version contains 10,060 trainable parameters; the no-graph variant contains 5,668.

C. Semantic-Group Contrastive Learning

Two stochastic views are generated from the same transaction. Instead of independent scalar masking, the augmenter separately masks raw transaction dimensions, customer context, merchant context, and the relation embedding. This preserves the meaning of each typed view while testing whether the fused representation remains stable when one semantic source is partially unavailable. For a batch B, normalized representations $z_i^{(1)}$ and $z_i^{(2)}$ are optimized with a symmetric InfoNCE objective:

$$L_{con} = -(1/2|B|) \Sigma_i [\log \exp(s_{ii}/\tau)/\Sigma_j \exp(s_{ij}/\tau) + \log \exp(s_{ii}/\tau)/\Sigma_j \exp(s_{ji}/\tau)] \quad (2)$$

where s_{ij} is cosine similarity and τ=0.2. Fraud classification uses focal binary cross-entropy [30], and the total objective is $L=L_{focal}+0.02L_{con}$. AdamW [31] is used for optimization.

D. Multi-Signal Drift Detection

A signature q_t is formed before scoring each snapshot from its category histogram, a ten-bin log-amount histogram, and the mean standardized feature vector. The reference signature is an exponential moving average with coefficient 0.85. In addition, novel-relation mass measures the proportion of current transactions whose category had zero support in the reference histogram. The score is

$$D_t = JS(q_t^r,\bar{q}^r) + 0.7\, JS(q_t^a,\bar{q}^a) + 0.08\, \|\mu_t-\bar{\mu}\|_2/\sqrt{d} + 25\, \nu_t \quad (3)$$

where JS is Jensen–Shannon divergence and ν_t is novel-relation mass. The threshold is the 90th percentile of scores observed across warm-up and validation snapshots; test information is not used for calibration. The large coefficient on ν_t is intentional because a newly supported typed relation is qualitatively different from moderate frequency drift among already observed relations.

E. Delayed Replay and Drift-Gated Ensemble

A balanced replay memory stores at most 6,000 examples, split between fraud and legitimate transactions when possible. Before scoring snapshot t, the current model receives one epoch of updates from snapshot t−1 plus replay. A frozen anchor is copied after warm-up and validation. Prior to any trigger, the probability is 0.90p_anchor+0.10p_current, emphasizing stability. After the first trigger, the mixture changes persistently to 0.35p_anchor+0.65p_current, emphasizing plasticity without discarding the calibrated warm model. We do not perform an unlabeled emergency gradient step at the trigger because pilot runs degraded probability ranking; adaptation waits for the next causally available delayed labels. The implementation records this design choice explicitly.

F. Computational Cost

Preprocessing is linear in transactions plus the maintained endpoint–relation state. Per-batch encoder cost is linear in batch size and representation dimension; InfoNCE is quadratic only in a capped contrastive subset of 96 examples. On the reported CPU environment with four PyTorch threads, the complete post-preprocessing experiment loop for DriftGuard-HCL averages 4.35 s per seed on the native stream and 4.50 s on the emergent stream.

5. Experimental protocol

A. Dataset and Stream Construction

BankSim is an agent-based bank-payment simulator calibrated from aggregated transactional data supplied by a Spanish bank [2]. It is synthetic and contains no production customer records. The downloaded table has 594,643 transactions, 4,112 customers, 50 merchants, 15 categories, and 7,200 fraud transactions (1.2108%). We group steps 0–179 into 30 nonoverlapping six-day snapshots. Each snapshot contains 240 labeled fraud transactions.

Table 1. Dataset and prequential split

Item	Value
Transactions	594,643
Customers / merchants	4,112 / 50
Categories	15
Fraud count / rate	7,200 / 1.2108%
Snapshots	30 × 6 days
Warm / validation	0–14 / 15–17
Test	18–29
Label delay	1 snapshot

The native stream preserves the chronology and all categories. The emergent-typology stream is a controlled stress test, not a naturally observed historical regime change. The categories es_leisure, es_travel, and es_sportsandtoys are removed only from warm-up and validation, then restored at test snapshot 18. They account for 1,795 of 262,051 test transactions (0.685%) and 1,152 of 2,880 test frauds (40.0%). From snapshot 19 onward, their labels become available under the same one-snapshot delay as all other labels. This deliberately severe construction tests relation novelty and adaptation; it must not be interpreted as an estimate of real-world drift frequency.

B. Baselines and Metrics

Six matched models are evaluated. Static MLP uses no graph context, no contrastive loss, and no test-time update. Online MLP adds delayed updates. Static HCL adds typed graph context and contrastive regularization but remains frozen. Continual HCL adds delayed updating without the anchor/current gate. DriftGuard without graph retains replay and gating but removes customer and merchant typed context. DriftGuard-HCL is the full model. The design is an ablation-focused comparison rather than a claim of universal state-of-the-art superiority over every published GNN. Established graph and temporal architectures [8]–[17] are discussed as conceptual baselines, while the released executable comparison isolates the proposed components under one leakage-controlled pipeline.

Metrics are computed on the concatenated test predictions: ROC-AUC, PR-AUC, F1, precision, recall, and recall among the top-scored 1% of transactions within each snapshot. Decision thresholds are selected on validation and then recalibrated only from the previous labeled snapshot. Every method is repeated with shared seeds {11,29,47,73,101}. Means and sample standard deviations are reported. Paired t tests and 20,000-resample seed-level bootstrap intervals compare the full model with selected ablations; five seeds are too few for definitive distributional claims, so the tests are supportive only.

C. Implementation

The released environment uses Python 3.13.5, PyTorch 2.10.0+cpu, scikit-learn 1.8.0, pandas 2.2.3, NumPy 2.3.5, and SciPy 1.17.0. The hidden representation has 32 dimensions. Training uses AdamW with learning rate 6×10^−4 and weight decay 10^−5. Mini-batches are class-balanced with at most four negatives per fraud and 1,024 sampled examples per update. The experiment script writes raw seed-level metrics after each run, so partial progress remains auditable.

6. Results

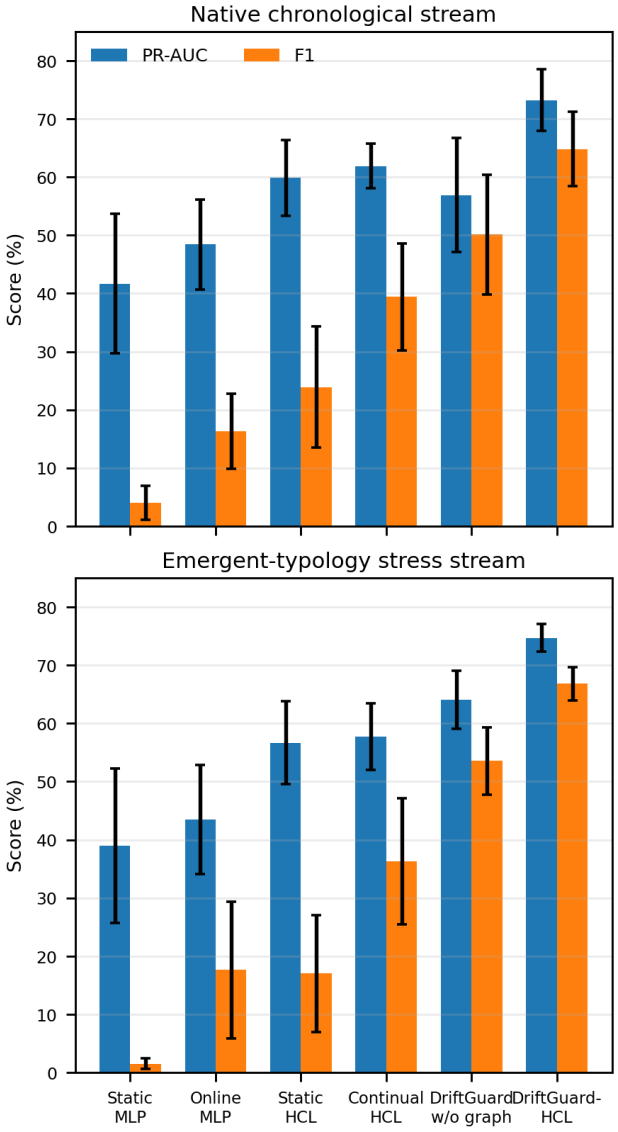

Fig. 2 PR-AUC and F1 on the native and emergent-typology streams. Bars show mean ± standard deviation over five shared seeds.

A. Overall Detection Quality

On the native stream, DriftGuard-HCL reaches 73.20% PR-AUC and 64.78% F1. Relative to Continual HCL, the absolute gains are 11.30 and 25.38 percentage points. The larger F1 gain indicates that the gate and anchored threshold calibration primarily improve operating-point stability, not only global ranking. Relative to the no-graph version, typed graph context contributes 16.31 PR-AUC points and 14.68 F1 points. Recall@1% improves more modestly to 65.51%; the paired difference from Continual HCL is not significant at five seeds.

Under emergent typologies, the full model obtains 74.70% PR-AUC and 66.83% F1. It exceeds Continual HCL by 16.93 PR-AUC points and 30.56 F1 points. The stress-stream score is not directly comparable to the native score as a difficulty ranking: removing three categories from warm-up changes both the training mix and which fraud mechanisms the anchor sees. The valid interpretation is comparative within the same stream. The full model's lower standard deviations and stronger thresholded performance show that drift-aware mixing stabilizes the delayed update sequence.

Table 2. Native stream results (%)

Method	AUC	PR-AUC	F1	R@1%
Static MLP	97.37±0.92	41.67±12.01	4.02±2.90	56.84±15.49
Online MLP	98.06±0.36	48.41±7.76	16.29±6.48	61.22±5.76
Static HCL	98.93±0.19	59.85±6.52	23.87±10.41	62.22±5.28
Continual HCL	99.15±0.14	61.89±3.84	39.40±9.22	64.94±2.22
DG w/o graph	98.49±0.33	56.89±9.79	50.10±10.27	63.21±2.21
DriftGuard-HCL	99.39±0.10	73.20±5.29	64.78±6.39	65.51±5.73

Table 3. Emergent-typology results (%)

Method	AUC	PR-AUC	F1	R@1%
Static MLP	97.25±0.85	38.97±13.27	1.51±0.88	53.85±18.48
Online MLP	97.82±0.43	43.48±9.43	17.65±11.75	57.82±10.13
Static HCL	98.83±0.20	56.68±7.17	17.03±10.03	60.23±5.34
Continual HCL	99.01±0.20	57.76±5.72	36.28±10.83	63.80±2.68
DG w/o graph	98.61±0.09	64.05±4.96	53.53±5.77	59.28±5.22
DriftGuard-HCL	99.43±0.06	74.70±2.36	66.83±2.85	66.03±2.25

B. Drift Response

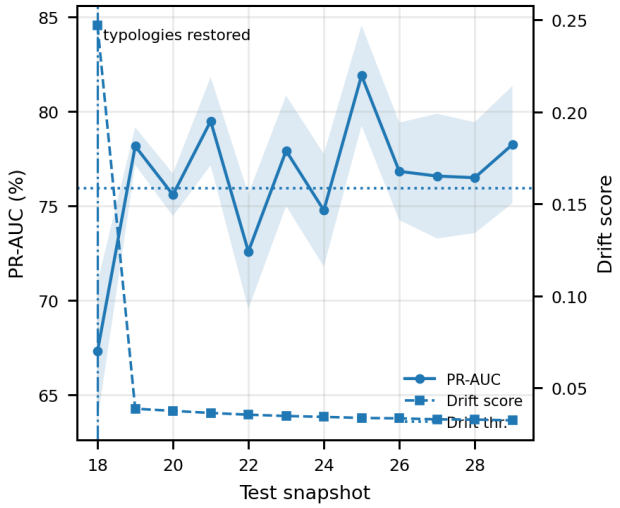

Fig. 3 Emergent-stream response of DriftGuard-HCL. The relation-novelty term produces one trigger when held-out categories are restored at snapshot 18; subsequent snapshots remain below the calibrated threshold while PR-AUC recovers after delayed labels become available.

The detector produces exactly one trigger in every emergent-stream seed and zero triggers in the native stream. At snapshot 18, novel categories create a drift score above the warm-validation 90th-percentile threshold. The ensemble therefore shifts from 10% to 65% current-model weight. Snapshot 18 is still scored before any label from the restored categories is available. Before snapshot 19, labels from snapshot 18 enter the replay update, after which per-snapshot PR-AUC remains high. This sequence demonstrates that the trigger is based on unlabeled distributional evidence while parameter adaptation remains causally delayed.

C. Component Contributions

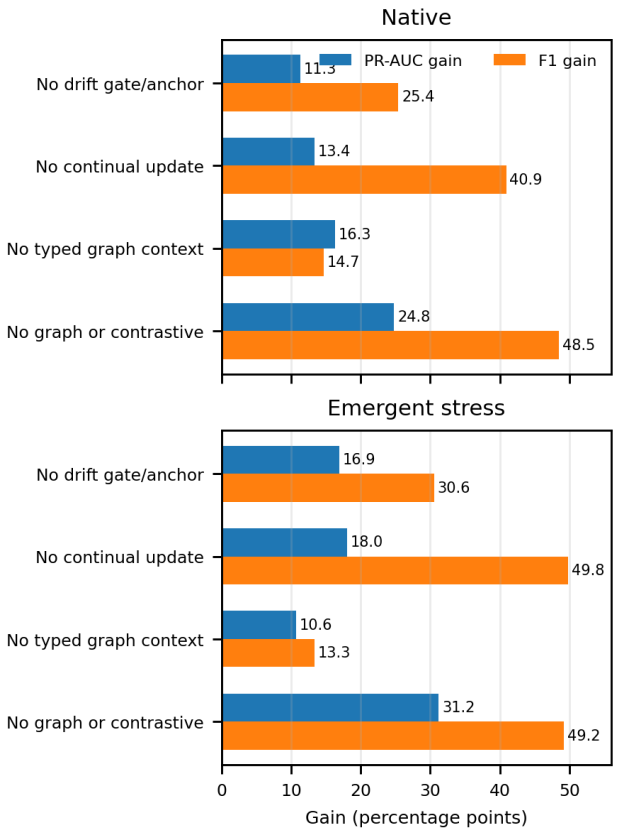

Fig. 4 Absolute gain of the full model over targeted ablations. Typed graph context, continual updating, and the drift-gated anchor each contribute; gains are measured within the same stream.

The ablations reveal complementary effects. Removing typed graph context but retaining drift gating reduces PR-AUC by 16.31 points on the native stream and 10.65 points on the emergent stream. Replacing the full model with a continually updated HCL model removes the anchor and gating policy, reducing PR-AUC by 11.30 and 16.93 points. Freezing HCL reduces it by 13.35 and 18.01 points. The no-graph/no-contrastive Online MLP trails by 24.78 and 31.21 points. These nonadditive drops are expected because replay, contrastive regularization, and gating interact rather than contribute independently.

D. Paired Evidence and Efficiency

Against Continual HCL, the paired PR-AUC gains are supported by intervals excluding zero in both streams: 6.81–15.79 points natively and 13.74–20.17 points under emergence. F1 intervals also exclude zero. Top-1% recall is different: the native interval crosses zero and the emergent paired t test has p=0.1269 despite a positive bootstrap interval. We therefore make no claim that DriftGuard-HCL decisively improves extreme-budget recall. Runtime remains small because the model is compact: 10,060 parameters and roughly 4.4 s per seed after preprocessing on CPU, compared with 3.0–3.2 s for Continual HCL.

Table 4. Full model minus continual hcl

Stream	Metric	Gain	95% CI	p
Native	PR-AUC	11.30	[6.81,15.79]	0.0113
Native	F1	25.38	[18.38,34.17]	0.0058
Native	R@1%	0.58	[-3.83,4.60]	0.8273
Emergent	PR-AUC	16.93	[13.74,20.17]	0.0009
Emergent	F1	30.56	[24.08,37.03]	0.0013
Emergent	R@1%	2.23	[0.38,4.34]	0.1269

7. Limitations and discussion

The evaluation has four important limits. First, BankSim is synthetic. It was designed to support reproducible fraud research [2], but it cannot reproduce every operational constraint, investigation delay, adversarial response, or privacy restriction of a bank. Recent work on realistic AML simulation similarly emphasizes that synthetic data are complements rather than substitutes for institution-specific validation [32]. Second, the emergent stream is constructed by withholding three categories; it tests typed relation novelty but not every form of gradual behavioral drift. Third, the compact graph-context encoder does not propagate arbitrary multi-hop messages and therefore should not be described as a replacement for HGT, TGN, or collusion-ring subgraph models. It is a scalable inductive alternative for transaction-level adaptation.

Fourth, the comparison is deliberately component controlled. A broader benchmark would tune whole-graph temporal GNNs, tree ensembles, and production tabular models under the identical prequential split. The present results instead establish that, within one executable family, the proposed gate, replay, contrastive objective, and typed context jointly improve performance. A future study should include naturally time-stamped fraud investigations, uncertain label arrival, recurring concepts, cost-sensitive alert allocation, and calibration metrics. It should also test whether an Audit-HCL-style dynamic heterogeneous encoder [1] can be used as the current branch while retaining the drift-gated stability–plasticity policy introduced here.

There is also a governance implication. A trigger should not automatically be treated as proof of attack. It indicates that the observable transaction distribution has departed from the calibration regime. In practice, the score components should be surfaced separately so that analysts can distinguish relation novelty, amount drift, and feature-mean shift. The frozen anchor provides a recoverable reference, and replay records which delayed labels drove adaptation, but additional audit logs and human approval may be required in regulated deployments.

8. Conclusion

DriftGuard-HCL addresses evolving fraud detection as a delayed-label, continual heterogeneous graph problem. The method compresses causal typed neighborhoods into an inductive encoder, applies semantic-group contrastive regularization, detects drift through distribution and relation-support signals, and gates a frozen-anchor/current-model ensemble. On five reproducible BankSim runs, it improves PR-AUC and F1 over matched static, online, continual, and no-graph variants in both the native chronology and a

disclosed emergent-typology stress test. The detector triggers only at the controlled relation-emergence event. These findings justify broader evaluation on real institutional streams, while the released code, seed-level results, figures, and reference-verification file make the current claims directly auditable.

References

[1] Jiao, Y., Fan, H., Yue, X., Ping, W., Sun, T., & Wang, J. (2026). Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud. Scientific Reports, 16, Article 12456. https://doi.org/10.1038/s41598-026-58938-5

[2] Lopez-Rojas, E. A., & Axelsson, S. (2014). BankSim: A bank payments simulator for fraud detection research. In Proceedings of the 26th European Modeling and Simulation Symposium (EMSS) (pp. 144-152).

[3] Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. arXiv, arXiv:1908.02591. https://doi.org/10.48550/arXiv.1908.02591

[4] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), Article 44. https://doi.org/10.1145/2523813

[5] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346-2363. https://doi.org/10.1109/TKDE.2018.2876857

[6] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the SIAM International Conference on Data Mining (pp. 443-448). SIAM. https://doi.org/10.1137/1.9781611972771.42

[7] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. Machine Learning, 106(9-10), 1469-1495. https://doi.org/10.1007/s10994-017-5642-8

[8] Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations (ICLR).

[9] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems 30 (pp. 1024-1034).

[10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In Proceedings of the International Conference on Learning Representations (ICLR).

[11] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In The Semantic Web: 15th International Conference, ESWC 2018 (LNCS 10843, pp. 593-607). Springer. https://doi.org/10.1007/978-3-319-93417-4_38

[12] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019). Heterogeneous graph attention network. In Proceedings of the World Wide Web Conference (WWW) (pp. 2022-2032). ACM. https://doi.org/10.1145/3308558.3313562

[13] Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020). Heterogeneous graph transformer. In Proceedings of the Web Conference (WWW) (pp. 2704-2710). ACM. https://doi.org/10.1145/3366423.3380027

[14] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., & Leiserson, C. E. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 4, pp. 5363-5370). AAAI Press. https://doi.org/10.1609/aaai.v34i04.5984

[15] Sankar, A., Wu, Y., Gou, L., Zhang, W., & Yang, H. (2020). 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 (WSDM) (pp. 519-527). ACM. https://doi.org/10.1145/3336191.3371813

[16] Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. In Proceedings of the International Conference on Learning Representations (ICLR).

[17] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. arXiv, arXiv:2006.10637. https://doi.org/10.48550/arXiv.2006.10637

[18] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., & Shen, Y. (2020). Graph contrastive learning with augmentations. In Advances in Neural Information Processing Systems 33 (pp. 5812-5823).

[19] Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2019). Deep graph infomax. In Proceedings of the International Conference on Learning Representations (ICLR).

[20] Hassani, K., & Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. In Proceedings of the International Conference on Machine Learning (ICML), PMLR 119 (pp. 4116-4126).

[21] Wang, X., Liu, N., Han, H., & Shi, C. (2021). Self-supervised heterogeneous graph neural network with co-contrastive learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (pp. 1726-1736). ACM. https://doi.org/10.1145/3447548.3467370

[22] Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., & Yu, P. S. (2020). Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM) (pp. 315-324). ACM. https://doi.org/10.1145/3340531.3411903

[23] Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., & He, Q. (2021). Pick and choose: A GNN-based imbalanced learning approach for fraud detection. In Proceedings of the Web Conference (WWW) (pp. 3168-3177). ACM. https://doi.org/10.1145/3442381.3449948

[24] Cardoso, M., Saleiro, P., & Bizarro, P. (2022). LaundroGraph: Self-supervised graph representation learning for anti-money laundering. In Proceedings of the 3rd ACM International Conference on AI in Finance (ICAIF) (pp. 130-138). ACM. https://doi.org/10.1145/3533271.3561777

[25] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13), 3521-3526. https://doi.org/10.1073/pnas.1611835114

[26] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). iCaRL: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2001-2010). IEEE. https://doi.org/10.1109/CVPR.2017.587

[27] Teng, D. (2025). TEAS: Token- and energy-aware autoscaling for cost-efficient LLM serving. AI and Data Science Journal, 6(3), 1-12.

[28] Zhang, F., Guo, Z., Ding, J., Yang, J., & Liu, W. (2026). Adaptive sensor fusion for robust perception in dense fog: A gated vision and LiDAR integration framework. Sensors, 26(8), 2345. https://doi.org/10.3390/s26082345

[29] Zi, B. (2024). Large language models for enterprise workflow automation in financial operations. Innovation and Technology Studies, 1(1), 24-29.

[30] Zi, B. (2024). Cloud-native distributed systems for real-time payment intelligence. AI and Data Science Journal, 5(4), 1-10.

[31] Teng, D. (2025). PACO: Predictive auto-configuration for SLO-constrained large language model inference serving. Innovation and Technology Studies, 2(1), 1-15.

[32] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access, 14, 1-18.