CAN WE ALERT SMART-CONTRACT FRAUD EARLY? UNICAST FOR CROSS-PHASE MULTI-FRAUD FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Smart-contract-related fraud caused over **US$14** billion in recorded financial losses in 2025 alone (Chainalysis, 2026). Early warning is therefore crucial for mitigating such losses. However, such early warning remains significantly challenging since complex frauds typically span different phases of smart contracts, while behavioral evidence is inevitably sparse and inconclusive in the early warning stages. Prior methods either do not account for cross-phase interdependencies or suffer from limited prediction accuracy under sparse behavioral evidence. Meanwhile, they typically target specific fraud mechanisms, e.g. Rug-Pull, limiting their applicability to diverse real-world fraud patterns. To address these challenges, we propose **UniCast**, a unified cross-phase framework for early fraud warning across multiple types of smart-contract frauds, including Rug Pull, Honeypot, Selling Restrictions, Fee Abuse, and Minting Abuse. Specifically, *Cross-Phase Dependency Modeling* first effectively captures and represents static-to-dynamic and dynamic-to-static inter-dependencies through bidirectional contrastive prediction. Then, the *Early-Warning Classifier* novelly addresses decision ambiguity under limited transaction observation by moving beyond single ambiguous decision space to separately parameterized benign and fraudulent representation spaces, each equipped with multiple subspace-specific anchors to characterize diverse behavioral patterns. Extensive experiments on the Multi-Fraud and RPHunter datasets demonstrate that **UniCast** achieves **98.1%** early-warning accuracy, compared with **78.3%** for the strongest baseline, RPHunter, while being **6.47**$\times$ faster in inference. *We will release our code upon acceptance.*

1 INTRODUCTION

Smart contracts, as one of the most important components of blockchain systems, enable a wide range of applications and services, such as token trading and asset management (Ethereum Foundation, 2026). It is a program deployed on a blockchain that defines and automatically executes a predefined set of rules and conditions governing interactions among participating parties. Given the critical role of smart contracts in blockchain-driven applications, fraudsters employ various techniques to exploit contract controls and manipulate trading activities for financial gain (Zhou et al., 2024; Lin et al., 2024). It is recorded at least **US$14 billion** financial loss in smart contract related scams in 2025 alone (Chainalysis, 2026). Hence, early fraud warning is indispensable for effective mitigation of such financial losses. In Figure 1, two well-known scams, Magnate Finance and Swaprum, accumulated substantial locked value before

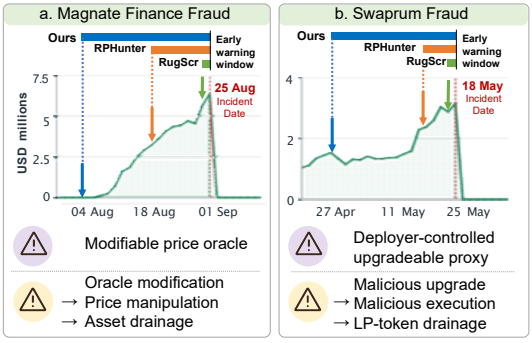

Figure 1: **UniCast** provides a substantially earlier warning window, allowing more time for intervention against potential fraud. Purple and yellow circles at bottom denote risks in pre-trading and during-trading phase, respectively.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

experiencing sharp declines around the reported incidents. The earlier we can identify such fraud, the greater the potential loss we can prevent.

Neverthess, providing early warning is nontrivial. Notic that the full life cycle of a smart contract consist of a *static pre-trading contract creation* phase, where contract logic and rules are designed and deployed, and a *dynamic transaction execution* phase, where user transactions invoke the contract and update its state. To enable effective early fraud alerting,we need to address 3 key challenges. 1). ***Profile Cross-Phase interdependency***. Static Contract code and dynamic transaction behavior provide complementary but fundamentally different views of potential fraud (Lin et al., 2024), while risks introduced during contract creation can shape subsequent behaviors during execution, establishing complex dependencies across the two phases as shown in Figure 1. 2). ***Fraud detection under limited transaction evidence***. Early warning signals may be mixed with normal trading behavior (Wu et al., 2024), while additional transaction evidence needed to disambiguate them may only emerge after substantial losses have occurred. 3). ***Support for diverse fraud types.*** A practically solution should be capable of identifying fraud no matter what type it is within a unified framework, rather than relying on separate, fraud-specific models.

Unfortunately, existing approaches do not adequately address these challenges. 1). Most existing studies consider only a certain phase of the fraud life cycle. They analyze static contract code during the pre-transaction phase (Zhou et al., 2024; Lin et al., 2024) or focus exclusively on dynamic trading behavior during the transaction execution phase (Wu et al., 2024; 2025), overlooking the relationship between what a contract permits and how its capabilities are subsequently exercised through transactions. 2). RPHunter (Wu et al., 2026) takes the first step toward modeling such cross-phase dependencies, but simply concatenating separately encoded features does not effectively model Static-to-Dynamic and Dynamic-to-Static inter-dependencies, and is also limited in making reliable predictions from a small number of observed transactions as shown in Figure. 1. 3). Most prior research, if not all, focuses on a single type of fraud, with a majority on Rug Pull and Ponzi.

Hence, we propose **UniCast**, a novel unified framework for early fraud forecasting that enables more accurate predictions and substantially earlier warning across multiple complex fraud types. Specifically, (1) **Cross-Phase Dependency Modeling (CPDM)** first captures the dependencies between the static properties of a contract and its subsequent dynamic transaction behavior through bidirectional *Static-to-Dynamic* and *Dynamic-to-Static* prediction. Such a *interdependent cross-phase representation* jointly captures static contract characteristics and evolving transaction behavior. (2).**Early-Warning Classifier (EWC)**. Under sparse observations, fraudulent and benign contracts may exhibit similar individual attributes, making their behavioral patterns difficult to distinguish. EWC addresses this ambiguity by learning separate representation subspaces for the two classes, with multiple anchors capturing diverse behavioral patterns within each subspace. By modeling fraudulent and benign behaviors in distinct representation spaces, EWC greatly reduces decision boundary ambiguity to support a substantially extended early-warning window (Fig. 1).

Experiments on two cross-phase fraud detection datasets, RPHunter (Wu et al., 2026) and a newly collected Multi-Fraud Dataset spanning five types of complex cross-phase fraud[1], including Rug Pull, Honeypot, Selling Restrictions, Fee Abuse, and Minting Abuse, demonstrate **UniCast**'s superior performance for early fraud warning. Our main contributions are:

- We propose **UniCast**, the first unified model for detecting multiple types of complex smart-contract fraud by analyzing the full smart contract life cycle, enabling a substantially extended early-warning window.
- CPDM effectively captures interdependent Static-to-Dynamic and Dynamic-to-Static representations, while EWC greatly mitigates decision-boundary ambiguity under limited transaction data. Together, they improve prediction reliability, with EWC also generalizing well across NLP, CV, and Code tasks.
- Extensive evaluation demonstrates our strong capability in early fraud alerting across all tested fraud types. On average, UniCast provides **98.1%** early warning, compared to **78.3%** by the strongest baseline RPHunter, while being **6.47**$\times$ faster in inference.

[1]Ponzi schemes are not included, as they are a simpler fraud type that primarily occurs in the pre-trading phase and can be effectively identified with 95.2% accuracy (Yang et al., 2025) before financial losses arise during transaction execution.

2 Related Work

Early warning for smart-contract fraud. One line of research identifies risks from static contract code: Tokeer (Zhou et al., 2024) analyzes token transfer logic, while CRPWarner (Lin et al., 2024) detects malicious functions that can enable rug pulls, allowing risks to be flagged before these operations are executed. Another line of research examines dynamic observed transactions: TokenScout (Wu et al., 2024) and RugScreener (Wu et al., 2025) use temporal graph learning to characterize fund flows and account interactions and identify evolving suspicious behavior. The above methods fails to model the crucial cross-phase interdependencies. More recently, RPHunter (Wu et al., 2026) have made the first step to fuse static and dynamic information in both phase. but simply concatenating separately encoded features does not effectively model Static-to-Dynamic and Dynamic to Static, resulting in limited prediction accuracy.

Early warning classifier. Recent research improves classification robustness through diverse decision rules: DPE (To et al., 2025) trains prototypical classifiers that focus on different features and samples to mitigate subpopulation shift. Another line of research refines distance-based representations: Roth et al. (Roth et al., 2022) introduce non-isotropy regularization to preserve local structure that sample-to-proxy distance optimization may overlook. These approaches emphasize classifier diversity and representation geometry, but do not directly target the ambiguity arising from limited early observations. To account for differences between class distributions, FeCAM (Goswami et al., 2023) uses class-specific covariance estimates, while Park et al. (Park et al., 2024) combine global and local metric tensors to capture shared and class-dependent geometry. However, adapting distance metrics alone does not specify how to jointly learn class-specific representations and comparable reference structures for early-warning decisions, leaving the distinction between fraudulent and benign contracts under limited transaction evidence an open challenge.

3 Method

Early warning of fraud across the full smart contract life-cycle is challenging due to the complex cross-phase dependencies and the limited transaction evidence available in the early warning window. This presents a fundamental trade-off dilemma: the earlier the alert, the less behavioral information is available for reliable prediction. Hence, **UniCast** (Figure 2) introduces **Cross-Phase Dependency Modeling (CPDM)** to capture the bidirectional dependencies between the static representation in contract-creation phase and dynamic representation in the transaction-execution phase, and **Early-Warning Classifer (EWC)** to improve prediction reliability under limited transaction observations. More model details are in Suppl D.

Problem Formulation. We denote the deployment-time contract code in the Pre-Transaction Phase as static information S, and the ordered events observed within an early warning window in the During-Transaction Phase as dynamic observations $D_{\leq n} = (d_1, \ldots, d_n)$. The static input remains fixed throughout the observation period, whereas the dynamic input grows as new events arrive. We use ERC-20 transfer events as the basic observation unit and represent transaction activity within the early warning window as an ordered sequence. Given a paired input $(S, D_{\leq n})$, the model outputs a fraud score $p_n = f_\theta(S, D_{\leq n})$, where n denotes the number of observed events available at the time of prediction. The contract label $y \in 0, 1$ indicates whether the reviewed contract is benign or fraudulent, respectively. Importantly, y characterizes the contract as a whole and does not require every observed event of a fraudulent contract to be malicious.

3.1 Cross-Phase Dependency Modeling (CPDM)

Contract capabilities defined in the static phase and transaction behavior occurs in the dynamic phases are highly related, but simply concatenating their features as in RPHunter (Wu et al., 2026) does not explicitly encourage the model to learn their correspondence. We therefore require a representation that captures cross-phase correspondence while preserving phase-specific information.

***Static* and *Dynamic* Representation.** First, a static feature vector $\mathbf{F}_S$ for the contract code and a dynamic feature vector $\mathbf{F}_{D,1\cdots n}$ for the n observed transaction events are extracted by the static encoder $\mathrm{Encoder}_S$ and the dynamic encoder $\mathrm{Encoder}_D$, respectively. The static representation $\mathbf{H}_S$ are then mapped to $\mathbf{F}_S$ via a MLP, and the dynamic feature $\mathbf{F}_{D,1\cdots n}$ together with their temporal information are further processed by a transformer (Vaswani et al., 2017) to obtain the dynamic representation

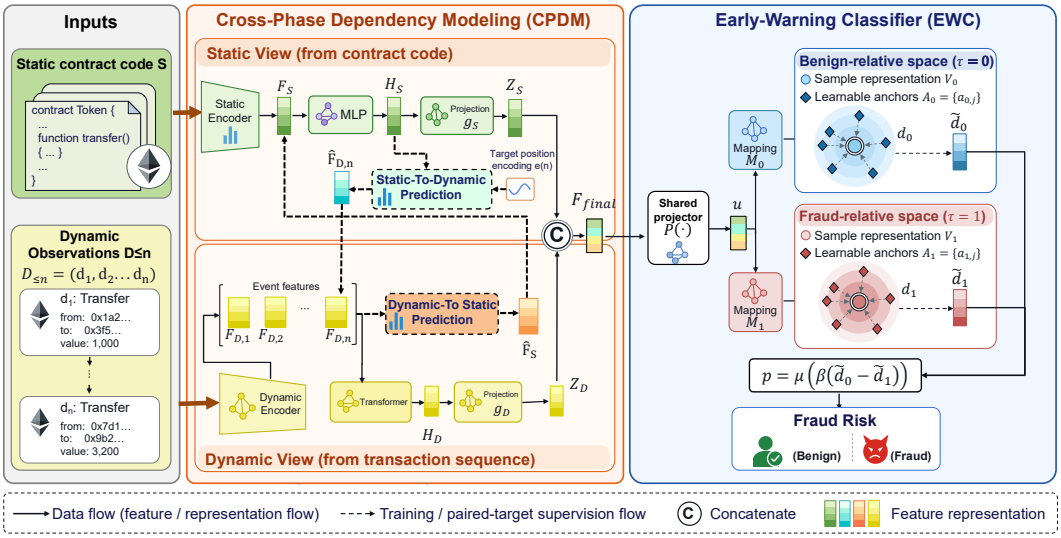

Figure 2: **Overview of UniCast.** CPDM first captures the cross-phase inter-dependencies between static contract properties and dynamic transaction behavior through bidirectional contrastive prediction. Then, EWC reduces the decision boundary ambiguity by learning separate representation subspaces for each class, with multiple anchors capturing diverse behavioral patterns within each subspace. They together improve prediction reliability to support a substantially extended early-warning window.

$\mathbf{H}_D$ for the current observation window. Here, $\mathbf{F}_{D,i}$ describes an individual event, whereas $\mathbf{H}_D$ summarizes the ordered transaction behavior within the window. The feature-construction rules remain fixed throughout training, while the representation networks, which consist of the MLP and Transformer, are optimized jointly.

Static-to-Dynamic and Dynamic-to-Static Interdependency. To learn the interdependency between the two representations, CPDM leverages bidirectional prediction: the static branch use static features to predict the corresponding dynamic transaction features (*Static-to-Dynamic*) while the dynamic branch use dynamic features to predict the corresponding static contract features (*Dynamic-to-Static*). Note that each prediction direction is associated with an observable paired target. For *Static-to-Dynamic* prediction, the fixed feature $\mathbf{F}_{D,n}$ of the last event in the current observation window serves as the target, where n specifies the target position. For *Dynamic-to-Static* prediction, the corresponding contract feature F_S serves as the target. Two prediction heads with independent parameters produce

$$\widehat{\mathbf{F}}_{D,n} = \mathrm{Pred}_{S\to D}(\mathbf{H}_S, n), \qquad \widehat{\mathbf{F}}_S = \mathrm{Pred}_{D\to S}(\mathbf{H}_D). \tag{1}$$

Each prediction head is a two-layer MLP, with the static-to-dynamic head additionally receiving the target-position encoding $e(n)$:

$$\begin{aligned} \mathrm{Pred}_{S\to D}(\mathbf{H}_S, n) &= W_D^{(2)}\phi\Big(W_D^{(1)}[\mathbf{H}_S; e(n)] + b_D^{(1)}\Big) + b_D^{(2)}, \\ \mathrm{Pred}_{D\to S}(\mathbf{H}_D) &= W_S^{(2)}\phi\Big(W_S^{(1)}\mathbf{H}_D + b_S^{(1)}\Big) + b_S^{(2)}. \end{aligned} \tag{2}$$

Here, $[\,;\,]$ denotes concatenation, ϕ is a nonlinear activation, and $W_D^{(l)}, b_D^{(l)}$ and $W_S^{(l)}, b_S^{(l)}$ are the learnable weights and biases of the two heads.

Notice that since training uses observation windows of different lengths, the static branch always predicts the last event of the current observed window rather than the last event in the complete transaction history. The index n identifies the target position without modifying h_S. And the dynamic branch summarizes the entire window, including the target event, which is the last observed event in the current window and provides the feature $F_{D,n}$ for static-to-dynamic prediction.

Aggregation of cross-phase Interdependency and phase-specific representations. While cross-phase correspondence captures shared information between the two phases, phase-specific informa-

tion can offer additional information for more robust classification. Prior multi-modal studies also show that both shared and modality-specific information can contribute to prediction accuracy (Jiang et al., 2023; Dufumier et al., 2025). We define g_S and g_D as learnable projection heads for the static and dynamic representations, respectively. They have independent parameters and map their inputs to a common dimension d. We therefore preserve the two representations through independent projections, $\mathbf{Z}_S = g_S(\mathbf{H}_S)$ and $\mathbf{Z}_D = g_D(\mathbf{H}_D)$, which map them to a common dimension d. The projected representations are then concatenated to form the final CPDM representation:

$$\mathbf{F}_{\text{final}} = [\mathbf{Z}_S; \mathbf{Z}_D] \in \mathbb{R}^{2d}. \tag{3}$$

3.2 Early-Warning Classifier

Equipped with an effective representation of cross-phase dependencies, we still need to address the prediction uncertainty caused by limited transaction observations in the early-warning window. The key challenge is that, with limited observations, samples from different classes may exhibit similar individual attributes, while their class identities may instead depend on different combinations of these attributes. Existing state-of-the-art distance-based classification methods (Snell et al., 2017; Park et al., 2024; Saralajew et al., 2025) typically employ a shared mapping and distance metric, implicitly assuming that samples from different classes can be discriminated within a common representation space. However, when discriminative patterns are class-specific, a shared metric cannot adapt to the distinct attribute relationships that characterize different classes. This limitation becomes particularly problematic under limited observations, where the available evidence is insufficient to clearly separate samples using a single shared representation structure.

Separated Representation Space for Each Class. To address this limitation, the Early-Warning Classifier (EWC) learns a separate representation space for each class, allowing the same sample to be evaluated according to class-specific feature relationships. Specifically, a shared projector $\mathcal{P}(\cdot)$ first maps the final CPDM representation F_{final} into a common latent representation:

$$u = \mathcal{P}(\mathbf{F}_{\text{final}}). \tag{4}$$

For each class τ, a class-specific mapping $\mathcal{M}_\tau(\cdot)$ then transforms u into the corresponding class-specific representation:

$$\mathbf{V}_\alpha = \mathcal{M}_\alpha(u), \qquad \alpha \in \{0 = \text{benign}, 1 = \text{fraud}\}. \tag{5}$$

The two mappings have the same architecture but independent parameters, allowing each representation space to capture the feature relationships that are most characteristic of its corresponding class. Since each class may exhibit multiple behavioral patterns, a single reference point may not adequately characterize its representation space. EWC therefore maintains multiple learnable anchors

$$A_\alpha = \{a_{\alpha,j}\}_{j=1}^{J} \tag{6}$$

for each class, providing multiple reference locations for distance-based comparison. These anchors are learned jointly as reference vectors.

Within-Space Multi-Anchor Distance Aggregation. Multiple anchors provide different reference locations within each class-specific space. Their relationships to a sample therefore need to be aggregated into a single class-level distance score. We first unit-normalize the sample representation and anchors and compute the squared Euclidean distance:

$$\eta_{\alpha,j} = \left|\bar{\mathbf{V}}_\alpha - \bar{a}_{\alpha,j}\right|_2^2, \tag{7}$$

where a bar denotes unit normalization. We then assign distance-based weights to the anchors:

$$\gamma_{\alpha,j} = \text{softmax}_j \frac{-\eta_{\alpha,j}}{\sigma_a}, \tag{8}$$

where $\sigma_a > 0$ controls the concentration of the weights. The resulting weighted distance

$$d_\alpha = \sum_{j=1}^{J} \gamma_{\alpha,j}\eta_{\alpha,j} \tag{9}$$

gives closer anchors greater influence while retaining contributions from other relevant anchors, rather than relying solely on the nearest reference point. This multi-anchor formulation allows each

class-specific space to characterize multiple behavioral patterns rather than forcing all samples of a class toward a single prototype.

Within-Space and Cross-Space Distance Normalization. Since the two class-specific spaces are learned independently, their anchor distributions may exhibit different levels of dispersion, making their raw distances not directly comparable. Although unit normalization controls the magnitude of individual representations, it does not guarantee comparable dispersion across the two anchor sets. To account for this difference, we estimate the characteristic scale $\widetilde{\mathbf{V}}_\tau$ of each class-specific space using the mean pairwise squared distance between its normalized anchors:

$$\widetilde{\mathbf{V}}_\alpha = \frac{1}{J(J-1)} \sum_{j \neq k} |\bar{a}_{\alpha,j} - \bar{a}_{\alpha,k}|_2^2 . \tag{10}$$

We then normalize the class-level distance by the corresponding within-space scale:

$$\widetilde{d}_\alpha = \frac{d_\alpha}{g(\widetilde{\mathbf{V}}_\alpha) + \epsilon}, \qquad \tau \in \{0, 1\}, \tag{11}$$

where $g(\cdot)$ denotes stop-gradient and $\epsilon > 0$ ensures numerical stability. This normalization places distances from the two independently learned spaces on a comparable scale, allowing the relative distance to reflect the sample's fit to each class-specific reference structure rather than differences in the dispersion of the two spaces.

Fraud Score. Finally, EWC converts the relative difference between the normalized class distances into a fraud probability:

$$p = \mu\left(\beta\left(\widetilde{d}_0 - \widetilde{d}_1\right)\right), \tag{12}$$

where $\mu(\cdot)$ is the sigmoid function and $\beta > 0$ is a learnable scale parameter. The output contains no additive bias, so the decision boundary is fixed at $\widetilde{d}_0 - \widetilde{d}_1 = 0$. EWC therefore predicts fraud when the sample is relatively closer to the normalized fraudulent reference structure than to the benign one, i.e., when $\widetilde{d}_0 - \widetilde{d}_1 > 0$.

In summary, EWC reduces decision boundary ambiguity under limited transaction evidence by combining two complementary mechanisms: separate representation spaces allow each class to characterize its own discriminative attribute relationships, while multiple anchors capture distinct behavioral patterns within each class. Together, these mechanisms provide more informative class-specific distance comparisons when the available transaction evidence is insufficient to establish a clear boundary in a shared representation space.

3.3 Model Training

We use bidirectional contrastive prediction to align predicted features with their paired targets through the representation-learning loss $\mathcal{L}_{\mathrm{CPDM}}$, while optimizing the EWC fraud probability with contract-level labels and binary cross-entropy through the classification loss $\mathcal{L}_{\mathrm{EWC}}$. The two objectives are optimized in a unified training objective:

$$\mathcal{L} = \mathcal{L}_{\mathrm{EWC}} + \lambda\, \mathcal{L}_{\mathrm{CPDM}}, \tag{13}$$

where $\lambda \geq 0$ controls the weight of the representation-learning loss. Please refer to Suppl. E for details.

4 Multi-Fraud Dataset

Despite the availability of existing datasets for evaluating fraud detection, we observe a lack of datasets that capture multiple fraud mechanisms and their interdependencies. Although RPHunter (Wu et al., 2026) introduces the first dataset capturing frauds cross-phase inter-dependencies, it focuses exclusively on Rug Pull fraud (Table. 1).

To enable the evaluation of mutiple types of fraud under more diverse and complex scenarios, we collect a multi-fraud dataset (Table. 1 for token and decentralized exchange (DEX) contracts on Ethereum and BNB Smart Chain. It covers five complex cross-phase fraud categories: Rug Pull, Honeypot, Selling Restrictions, Fee Abuse, and Minting Abuse. More details of the proposed multi-fraud dataset are provided in Suppl. A. Note that simpler single-phase scams, such as Ponzi schemes, are excluded, as they can be effectively detected and prevented during the contract creation phase. Notice that although our method primarily targets more complex fraud mechanisms, We show our performance on Ponzi in Suppl. B.

Table 1: Comparison of RpHunter Dataset (Wu et al., 2026) and our Multi-Fraud Dataset.

Fruad Type	RPHunter	**Multi-Fraud**
Rug Pull	645	87
Honeypot	–	142
Selling restrictions	–	65
Fee abuse	–	193
Minting abuse	–	151
Fraud subtotal	645	**638**
Benign	1675	**638**
Total	2320	**1,276**

5 EXPERIMENTS

Datasets. We evaluate performance on RPHunter (Wu et al., 2026), CIFAR-10H(Peterson et al., 2019), CivilComments(Borkan et al., 2019),Devign(Tran et al., 2025) and our Muti-Fraud Dataset. More dataset details are shown in Suppl. A.

Implementation. **UniCast** uses Python 3.12.3, PyTorch 2.5.1, and CUDA 12.4 on a single NVIDIA A800 PCIe GPU with 80 GB of device memory. The operating system is Ubuntu 22.04.4 LTS, the processor is an Intel Xeon Gold 6348 at 2.60 GHz, and the container memory limit is 120 GiB. Computation uses FP32 without automatic mixed precision. More implementation details are shown in Suppl. C.

Baselines. We compare with three categories of approaches: (1) static Contract creation phase-targeted methods (2) Dynamic Contract execution phase-targeted methods, Rug pull detection (Srifa et al., 2025) and RugScreener (Wu et al., 2025); (3) Cross-phase methods, RpHunter (Wu et al., 2026). More baseline details are in Suppl. C.3.

Metrics. We report F1 score(%) in the main paper. Evaluation on metrics Precision and Recall are provided in Suppl. B. All results are averaged over 3 independent runs.

5.1 EVALUATION ON EARLY FRAUD ALERTING

Capability on pre-incident fraud alerting To evaluate our capability to provide warnings before harmful incidents, we replay the transaction histories in the Multi-Fraud dataset chronologically and measure the timing of the first alert relative to the annotated harmful event in terms of pre-event coverage, mean first-alert event, and lead blocks. As shown in Table 2, our method consistently outperforms the SoTA methods across all fraud categories, achieving substantially higher pre-incident coverage and earlier first alerts. On average, in a 50% observed transaction window, our method achieves **98.1%** pre-incident coverage and triggers the first alert after only **27.3%** observed transactions, compared with 78.3% pre-event coverage and 37.3% transactions for the strongest baseline, RPHunter.

Early fraud alerting under different transaction observations. To further evaluate the capability of our method to detect fraud from a limited number of early transactions within a long transaction history. Specifically, we vary the percentage of observed transactions to 5%, 20%, 50%, and evaluate the corresponding fraud detection performance. Higher detection accuracy with fewer observed transactions indicates a stronger capability for early fraud alerting. We observe that although the accuracy of most methods generally increases as the observation window expands, the improvement is not linear. This is mainly because, in the absence of supervision over which transactions provide effective fraud-related signals, additional transaction data does not necessarily provide more useful information. Instead, as the observation window expands, the model may receive an increasing amount of irrelevant or noisy transaction data, resulting in unstable performance. However, as shown in Table 3, **our method consistently and significantly outperforms all baselines** on both datasets, across all five fraud categories and under all observation windows.

Table 2: Pre-incident fraud alerting on the Multi-Fraud dataset with a 50% observed transaction window. Pre-event coverage (PC) is the fraction of cases for which an alert is issued strictly before the fraud event within the 50% observation window. Mean first-alert event (MFE) measures the average transaction position of the first alert. ↑ higher is better, while ↓ lower is better.

Method	Venue	Phase	Rug Pull		Honeypot		Selling Restrictions		Fee Abuse		Minting Abuse		Average	
			PC ↑	MFE ↓	PC ↑	MFE ↓	PC ↑	MFE ↓	PC ↑	MFE ↓	PC ↑	MFE ↓	PC ↑	MFE ↓
Tokeer	ICSE 2024	Contract	76.5%	49.6%	22.9%	78.3%	73.5%	48.9	20.4%	90.6	77.7%	49.5	54.2%	63.4%
CRPWarner	TSE 2024	Contract	75.9%	49.2	78.5%	49.9	30.5%	83.1	78.2%	50.4	78.8%	49.9	68.4%	56.5%
Rug Pull Detection	BRA 2025	Transcation	78.3%	49.4	22.1%	89.2	77.6%	50.1	68.7%	54.9	77.8%	49.0	64.9%	56.7%
RugScreener	TIFS 2025	Transcation	83.9%	46.1	60.7%	49.2	68.2%	53.4	75.2%	48.5	75.0%	48.7	72.6%	49.2%
RPHunter	TSE 2026	Both	93.5%	30.1	69.4%	42.5	81.3%	39.2	70.6%	38.7	76.7%	36.2	78.3%	37.3%
UniCast (Ours)	-	Both	**99.7%**	**21.5%**	**96.3%**	**30.3%**	**97.2%**	**35.1%**	**98.1%**	**22.4%**	**98.3%**	**27.1%**	**98.1%**	**27.3%**

Table 3: Early Fraud Alerting: Fraud Detection under different transaction observations (F1(%)). **Bold** indicates the best performance. Results under metric Precision and Recall are in Suppl. B.1.

Method	RPHunter Dataset			Multi-Fraud Dataset																	
	Rug Pull			Rug Pull			Honeypot			Selling restrictions			Fee abuse			Minting abuse			Average		
	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%
Tokeer.	16.5	21.5	27.6	19.5	39.2	42.4	14.1	36.7	53.1	4.3	20.1	42.9	18.6	21.7	37.8	27.3	43.4	28.6	16.8	32.2	41.0
CRPWarner	36.6	41.5	39.5	24.3	36.0	31.6	36.2	55.8	26.3	15.3	38.4	50.6	18.3	37.0	36.8	38.4	34.4	51.6	26.5	40.3	39.4
RugPull detection	50.4	60.1	62.4	32.9	40.0	63.4	30.4	39.5	49.2	38.2	56.9	63.6	40.3	57.4	61.1	41.2	53.7	54.6	36.6	49.5	58.4
RugScreener	68.5	76.8	79.9	17.7	16.0	70.6	23.8	28.4	71.7	4.4	6.7	60.1	24.9	21.8	63.0	12.0	21.3	70.6	15.6	18.8	61.8
RPHunter	77.1	81.0	88.9	39.8	42.3	79.5	40.5	21.6	57.3	33.5	34.6	60.7	32.1	45.8	32.9	34.6	35.9	41.7	36.1	36.0	54.4
UniCast (ours)	**82.1**	**89.4**	**93.0**	**40.5**	**56.2**	**81.3**	**41.4**	**60.0**	**73.7**	**40.3**	**69.6**	**78.1**	**40.7**	**58.6**	**64.3**	**48.3**	**54.6**	**75.9**	**42.2**	**59.8**	**74.7**

5.2 Evaluation of Early-Warning Classifier

In-domain comparison. To demonstrate the effectiveness of the proposed Early-Warning Classifier, we use t-SNE to visualize the learned representations of the classifier used in RPHunter and our Early-Warning Classifier on the Multi-Fraud Dataset. As shown in Figure 3, RPHunter suffers from severe entanglement between positive and negative samples, whereas UniCast exhibits a clear separation between the two classes. Moreover, as the observation window expands, the class separation becomes increasingly pronounced, indicating that UniCast progressively learns more discriminative representations from the observed transaction data.

Remarks. We note that the limited size of the Multi-Fraud Dataset prevents us from reliably performing multi-class classification across the five fraud types. We therefore acknowledge this as a limitation of the current work and leave comprehensive multi-class fraud classification for future investigation. Further discussion is provided in Suppl. F.

Generalization to other research domain.

Although designed for our target problem, we find that its ability to handle ambiguous decision boundaries generalizes well to other domains, where it consistently outperforms SoTA classifiers. As shown in Table. 4, **our EWC achieves superior performance on classification tasks on datasets from different domains**, including CIFAR-10H (Peterson et al., 2019) in **Compter Vision**, CivilComments (Koh et al., 2021) in **NLP** and Devign (Zhou et al., 2019) in **code classification**. For fair comparison, when a baseline provides a standalone classification head, we attach the head to our representation-learning framework. Otherwise, we evaluate the complete baseline method using

Table 4: Generalization of the Early-Warning Classifier across different domains, reported as F1 (%).

Classifier	Venue	Phase	Domain			
			CV	NLP	CODE	WEB3
			CIFAR-10H	CivilComments	Devign	Multi-Fraud
50-Snapshot	ICLR'25	Uncertainty estimation	**97.1**	63.6	42.5	44.1
MCInfoNCE	ICML'23	Probabilistic	95.2	50.4	31.5	29.7
DPE	ICML'25	General classification	86.9	69.9	46.5	40.1
AFR	ICML'23	General classification	82.1	67.6	49.7	49.1
HGAN4VD	C&S'25	Code vulnerability detection	90.2	65.4	70.4	68.3
SCL-CVD	C&S'24	Code vulnerability detection	87.4	66.3	63.3	69.1
EWC (Ours)	**This work**	**Early warning**	**97.1**	**70.3**	**72.4**	**72.9**

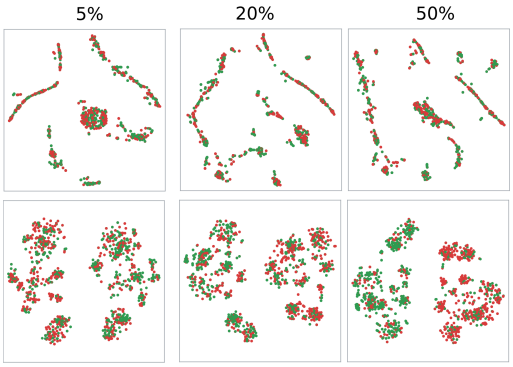

Figure 3: In-domain classifier comparison between RPHunter (Top) and UniCast (Bottom).

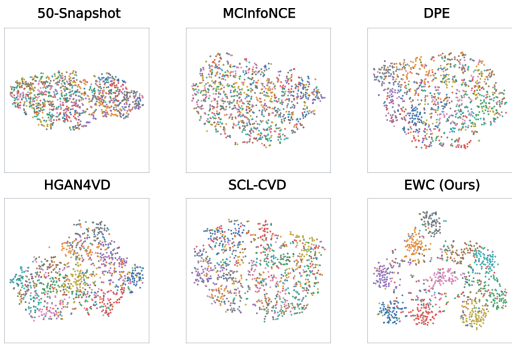

Figure 4: Cross-domain classifier comparison on multi-class classification CV task.

its original classification mechanism. These results demonstrate the effectiveness of the proposed classifier and broader applicability of the proposed classifier beyond the target fraud detection task. Visual results on Computer Vision multi-class classification task under CIFAR-10H (Peterson et al., 2019) is show in Figure 4. More visual comparison in other domains can be found at Suppl. B.

5.3 Ablation Studies

To evaluate the contribution of each component, we conduct an ablation study on Multi-Fraud dataset in Table 5. Removing any component from the representation modeling module or replacing the proposed classifier with alternative classifiers results in a performance drop. Specifically, within the representation modeling module, both prediction directions contribute to the overall performance, with dynamic-to-static prediction having a larger impact. Replacing the proposed classifier with alternative classifiers also leads to a noticeable performance degradation, highlighting the effectiveness of the proposed Early-Warning Classifier.

Table 5: Ablation studies in metric F1 (%).

Module / variant	**5%**	**20%**	**50%**
A. Cross-Phase Dependency Modeling			
w/o bidirectional	36.5	39.4	56.7
w/o static-to-dynamic	39.5	52.4	66.9
w/o dynamic-to-static	37.2	42.6	59.5
B. Early-Warning Classifier			
MLP classifier	16.1	30.5	68.7
Shared space mapping	20.5	39.3	64.9
w/o normalization	36.7	49.4	69.3
Full UniCast(ours)	**42.2**	**59.8**	**74.7**

5.4 Runtime Analysis

Table 6 compares inference latency, parameter counts, and inference memory for the evaluated methods. Although our **UniCast** has more parameters, it uses less inference memory and achieves lower latency under the measured configurations. Its inference is approximately **6.47**× faster than RugScreener under these configurations. At the same time, we report the parameter sensitivity analysis in Suppl B.4.

Table 6: Runtime Analysis on model parameters (thousands), inference memory (GiB), and inference latency (ms) at three observation ratios.

Model	**#Para**	**Mem**	**5%**	**20%**	**50%**
RugScreener	70.95	4.21	23.12	23.47	24.22
RPHunter	**65.38**	1.57	26.57	26.60	26.85
UniCast	629.08	**1.41**	**3.62**	**3.67**	**3.66**

6 Conclusion

To conclude, we present, to the best of our knowledge, the first solution for multi-fraud detection that leverages the full lifecycle of smart contracts, from pre-transaction deployment to transaction-time behavior. Our approach provides a substantially earlier alerting window, enabling timely intervention to mitigate potential financial losses. This capability is enabled by effective modeling of cross-phase dependencies and the robust Early-Warning Classifier. We hope this work can provide insights into the practical deployment of proactive multi-fraud detection in real-world blockchain security systems.

7 STATEMENTS

7.1 REPRODUCIBILITY STATEMENT

We provide detailed information to support the reproducibility of our work. The model architecture and training objectives are described in the main paper, with additional method details, loss formulations experimental settings, including datasets, manipulation generation, degradation settings, and training configurations, are detailed in Suppl C and Suppl E, respectively. Additional quantitative and qualitative results are also provided in the Suppl B. Anonymous source code is also provided in GitHub

7.2 USE OF GENERATIVE AI

We used generative AI tools for grammar checking and language polishing to improve the readability of the paper. We have reviewed all AI-assisted content and experimental artifacts. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI

7.3 ETHICS STATEMENT

The study uses public blockchain data and contract-level labels. It does not attempt to deanonymize natural persons or infer protected personal attributes. Fraud predictions can create reputational and financial harm if treated as definitive accusations; the intended use is risk triage followed by independent verification. We report aggregate results, avoid operational exploit instructions, and frame model output as a screening signal rather than proof of intent.

REFERENCES

Gustaf Ahdritz, Aravind Gollakota, Parikshit Gopalan, Charlotte Peale, and Udi Wieder. Provable uncertainty decomposition via higher-order calibration. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 6244–6285, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/122ea6470232ee5e79a2649243348005-Paper-Conference.pdf.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pp. 491–500, 2019.

Chainalysis. Record $17 billion estimated stolen in crypto scams and fraud in 2025 as impersonation tactics and AI enablement surge. https://www.chainalysis.com/blog/crypto-scams-2026/, January 2026. Published January 13, 2026; accessed September 16, 2026.

Benoit Dufumier, Javiera Castillo Navarro, Devis Tuia, and Jean-Philippe Thiran. What to align in multimodal contrastive learning? In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/108030643e640ac050e0ed5e6aace48f-Abstract-Conference.html.

Ethereum Foundation. Introduction to smart contracts. https://ethereum.org/developers/docs/smart-contracts/, 2026. Accessed September 16, 2026.

Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. FeCAM: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/15294ba2dcfb4521274f7aa1c26f4dd4-Abstract.html.

Qian Jiang, Changyou Chen, Han Zhao, Liqun Chen, Qing Ping, Son Dinh Tran, Yi Xu, Belinda Zeng, and Trishul Chilimbi. Understanding and constructing latent modality structures in multimodal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7661–7671, 2023. URL https://openaccess.thecvf.co

m/content/CVPR2023/html/Jiang_Understanding_and_Constructing_Latent_Modality_Structures_in_Multi-Modal_Representation_Learning_CVPR_2023_paper.html.

Michael Kirchhof, Enkelejda Kasneci, and Seong Joon Oh. Probabilistic contrastive learning recovers the correct aleatoric uncertainty of ambiguous inputs. In *International Conference on Machine Learning*, pp. 17085–17104. PMLR, 2023.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard L. Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664, 2021.

Zewei Lin, Jiachi Chen, Zibin Zheng, Jiajing Wu, Weizhe Zhang, and Yongjuan Wang. CRPWarner: Warning the risk of contract-related rug pull in DeFi smart contracts. *IEEE Transactions on Software Engineering*, 50(6):1534–1547, 2024. doi: 10.1109/TSE.2024.3392451. URL https://doi.org/10.1109/TSE.2024.3392451.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. Codexglue: A machine learning benchmark dataset for code understanding and generation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c16a5320fa475530d9583c34fd356ef5-Abstract-round1.html.

Jae Hyeon Park, Gyoomin Lee, Seunggi Park, and Sung In Cho. Not all classes stand on same embeddings: Calibrating a semantic distance with metric tensor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17722–17731, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Park_Not_All_Classes_Stand_on_Same_Embeddings_Calibrating_a_Semantic_CVPR_2024_paper.html.

Joshua C. Peterson, Ruairidh M. Battleday, Thomas L. Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9617–9626, 2019.

Shikai Qiu, Andres Potapczynski, Pavel Izmailov, and Andrew Gordon Wilson. Simple and fast group robustness by automatic feature reweighting. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28448–28467. PMLR, 2023. URL https://proceedings.mlr.press/v202/qiu23c.html.

Karsten Roth, Oriol Vinyals, and Zeynep Akata. Non-isotropy regularization for proxy-based deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7420–7430, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Roth_Non-Isotropy_Regularization_for_Proxy-Based_Deep_Metric_Learning_CVPR_2022_paper.html.

Sascha Saralajew, Ashish Rana, Thomas Villmann, and Ammar Shaker. A robust prototype-based network with interpretable RBF classifier foundations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20273–20282, 2025. doi: 10.1609/aaai.v39i19.34233. URL https://ojs.aaai.org/index.php/AAAI/article/view/34233.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.

Suparat Srifa, Yury Yanovich, Robert Vasilyev, Tharuka Rupasinghe, and Vladislav Amelin. Rug pull detection on decentralized exchange using transaction data. *Blockchain: Research and Applications*, 6(3):100275, 2025. doi: 10.1016/j.bcra.2025.100275.

Nguyen Nhat Minh To, Paul F R Wilson, Viet Nguyen, Mohamed Harmanani, Michael Cooper, Fahimeh Fooladgar, Purang Abolmaesumi, Parvin Mousavi, and Rahul Krishnan. Diverse prototypical ensembles improve robustness to subpopulation shift. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=qUTiOeM57J.

Hoai-Chau Tran, Anh-Duy Tran, and Kim-Hung Le. Detectvul: A statement-level code vulnerability detection for python. *Future Generation Computer Systems*, 163:107504, 2025. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2024.107504. URL https://www.sciencedirect.com/science/article/pii/S0167739X24004680.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Rongcun Wang, Senlei Xu, Yuan Tian, Xingyu Ji, Xiaobing Sun, and Shujuang Jiang. SCL-CVD: Supervised contrastive learning for code vulnerability detection via GraphCodeBERT. *Computers & Security*, 145:103994, 2024. doi: 10.1016/j.cose.2024.103994. URL https://doi.org/10.1016/j.cose.2024.103994.

Cong Wu, Jing Chen, Ziming Zhao, Kun He, Guowen Xu, Yueming Wu, Haijun Wang, Hongwei Li, Yang Liu, and Yang Xiang. TokenScout: Early detection of Ethereum scam tokens via temporal graph learning. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, pp. 956–970. Association for Computing Machinery, 2024. doi: 10.1145/3658644.3690234. URL https://doi.org/10.1145/3658644.3690234.

Cong Wu, Hangcheng Cao, Jing Chen, Xiyu Yan, Guowen Xu, Ziming Zhao, Yang Liu, and Hongbo Jiang. RugScreener: Leveraging temporal graph neural network for rugpull detection in DeFi. *IEEE Transactions on Information Forensics and Security*, 20:11120–11133, 2025. doi: 10.1109/TIFS.2025.3620669. URL https://doi.org/10.1109/TIFS.2025.3620669.

Hao Wu, Haijun Wang, Shangwang Li, Yin Wu, Ming Fan, Wuxia Jin, and Ting Liu. RPHunter: Unveiling rug pull schemes in crypto token via code-and-transaction fusion analysis. *IEEE Transactions on Software Engineering*, 2026. doi: 10.1109/TSE.2026.3716306. URL https://doi.org/10.1109/TSE.2026.3716306.

Weijia Yang, Tian Lan, Leyuan Liu, Wei Chen, Tianqing Zhu, Sheng Wen, and Xiaosong Zhang. CASPER: Contrastive approach for smart ponzi scheme detecter with more negative samples, 2025. URL https://arxiv.org/abs/2507.16840.

Yucheng Zhang, Xiaolin Ju, Xiang Chen, Misbahul Amin, and Zilong Ren. HGAN4VD: Leveraging heterogeneous graph attention networks for enhanced vulnerability detection. *Computers & Security*, 157:104548, 2025. doi: 10.1016/j.cose.2025.104548. URL https://doi.org/10.1016/j.cose.2025.104548.

Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yuanhang Zhou, Jingxuan Sun, Fuchen Ma, Yuanliang Chen, Zhen Yan, and Yu Jiang. Stop pulling my rug: Exposing rug pull risks in crypto token to investors. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice*, pp. 228–239. Association for Computing Machinery, 2024. doi: 10.1145/3639477.3639722. URL https://doi.org/10.1145/3639477.3639722.

APPENDIX

Our Suppl is structured as follows:

- Multi-Fraud Datasets in Section A
- More Results in Section B
- Implementation Details C
- Model Details in Section D
- Training Details in Section E
- Limitations in Section F

A MULTI-FRAUD DATASETS

A dataset restricted to one fraud mechanism cannot fully reflect the different ways in which contract properties interact with transaction behavior. We therefore construct Multi-Fraud with five categories: Rug Pull, Honeypot, Selling Restrictions, Fee Abuse, and Minting Abuse. The data cover Ethereum and BNB Smart Chain from January 14, 2017 to June 22, 2025. We collect contract bytecode and associated on-chain transaction records, with labels established using audit reports, community annotations, and manual investigations.

The initial collection contains 1,746 candidate contracts, including 811 fraudulent contracts. Following checks for data availability, duplicate records, and labeling errors, the final dataset contains 638 fraudulent and 638 benign contracts, for a total of 1,276 contracts. The fraudulent contracts comprise 87 Rug Pull, 142 Honeypot, 65 Selling Restrictions, 193 Fee Abuse, and 151 Minting Abuse cases. These categories support mechanism-specific evaluation; the primary prediction task remains contract-level fraud classification rather than training a separate detector for each mechanism.

B MORE RESULTS

The main paper evaluates classification, warning time, and computational cost. This section complements the classification tables with Precision and Recall, which distinguish the reliability of positive predictions from the coverage of fraudulent samples. We also report Ponzi results and provide slots for representation and parameter analyses.

B.1 PRECISION AND RECALL FOR THE MAIN-PAPER COMPARISONS

B.1.1 FRAUD DETECTION UNDER DIFFERENT OBSERVATION RATIOS

F1 alone does not reveal whether a performance difference comes from more reliable positive predictions or more complete fraud detection. Tables 7–11 supplement the fraud-detection comparison with Precision and Recall under the same splits, methods, and observation settings. All entries are percentages averaged over three independent runs.

Table 7: Supplementary classification results on the RPHunter dataset (%). P and R denote Precision and Recall. Dashes reserve entries to be completed.

	5%			20%			50%		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Tokeer	14.9	18.4	16.5	25.2	18.7	21.5	31.0	24.9	27.6
CRPWarner	37.3	35.9	36.6	42.2	40.8	41.5	40.2	38.8	39.5
Rug Pull Detection	51.1	49.7	50.4	60.8	59.4	60.1	63.1	61.7	62.4
RugScreener	69.2	67.8	68.5	77.5	76.1	76.8	80.6	79.2	79.9
RPHunter	77.8	76.4	77.1	81.7	80.3	81.0	89.6	88.2	88.9
UniCast(ours)	**82.8**	**81.4**	**82.1**	**90.1**	**88.7**	**89.4**	**93.6**	**92.4**	**93.0**

Table 8: Classification results on Multi-Fraud at the 5% observation ratio (%). Dashes indicate missing Precision and Recall values.

	Rug Pull			Honeypot			Selling restrictions			Fee abuse			Minting abuse		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Tokeer	22.8	17.0	19.5	12.1	16.9	14.1	5.1	3.7	4.3	16.2	21.8	18.6	31.5	24.1	27.3
CRPWarner	21.0	29.0	24.3	32.5	40.9	36.2	18.2	13.2	15.3	21.6	15.9	18.3	33.0	45.9	38.4
Rug Pull Detection	38.5	28.7	32.9	26.1	36.4	30.4	33.0	45.3	38.2	35.8	46.1	40.3	45.2	37.8	41.2
RugScreener	15.1	21.4	17.7	28.2	20.6	23.8	3.8	5.2	4.4	21.2	30.2	24.9	10.2	14.6	12.0
RPHunter	35.0	38.8	39.8	36.1	46.1	40.5	39.2	29.2	33.5	27.2	39.2	32.1	30.2	40.5	34.6
UniCast	**41.2**	**39.8**	**40.5**	**42.0**	**40.8**	**41.4**	**40.9**	**39.7**	**40.3**	**41.5**	**39.9**	**40.7**	**49.1**	**47.5**	**48.3**

Table 9: Classification results on Multi-Fraud at the 20% observation ratio (%). Dashes indicate missing Precision and Recall values.

	Rug Pull			Honeypot			Selling restrictions			Fee abuse			Minting abuse		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Tokeer	33.4	47.4	39.2	38.7	34.9	36.7	28.5	15.5	20.1	30.4	16.9	21.7	39.0	48.9	43.4
CRPWarner	32.4	40.5	36.0	60.6	51.7	55.8	46.5	32.7	38.4	45.4	31.2	37.0	39.9	30.2	34.4
Rug Pull Detection	49.9	33.4	40.0	47.6	33.8	39.5	65.0	50.6	56.9	58.9	56.0	57.4	55.0	52.5	53.7
RugScreener	19.4	13.6	16.0	18.7	59.0	28.4	16.1	4.2	6.7	15.5	36.7	21.8	16.1	31.5	21.3
RPHunter	45.5	39.5	42.3	15.8	34.1	21.6	38.6	31.4	34.6	38.2	57.2	45.8	48.4	28.5	35.9
UniCast	**57.0**	**55.4**	**56.2**	**60.8**	**59.2**	**60.0**	**70.5**	**68.7**	**69.6**	**59.4**	**57.8**	**58.6**	**55.3**	**53.9**	**54.6**

Table 10: Classification results on Multi-Fraud at the 50% observation ratio (%). Dashes indicate missing Precision and Recall values.

	Rug Pull			Honeypot			Selling restrictions			Fee abuse			Minting abuse		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Tokeer	32.3	61.7	42.4	44.3	66.3	53.1	53.4	35.9	42.9	52.5	29.5	37.8	45.6	20.8	28.6
CRPWarner	47.2	23.8	31.6	21.2	34.6	26.3	68.0	40.3	50.6	54.6	27.8	36.8	61.7	44.3	51.6
Rug Pull Detection	65.5	61.4	63.4	55.6	44.1	49.2	76.7	54.3	63.6	63.1	59.2	61.1	65.2	47.0	54.6
RugScreener	75.2	66.5	70.6	74.1	69.5	71.7	62.5	57.9	60.1	65.0	61.1	63.0	76.1	65.8	70.6
RPHunter	82.0	77.1	79.5	69.4	48.8	57.3	71.3	52.8	60.7	30.5	35.7	32.9	59.5	32.1	41.7
UniCast	**82.1**	**80.5**	**81.3**	**74.5**	**72.9**	**73.7**	**78.8**	**77.4**	**78.1**	**65.1**	**63.5**	**64.3**	**76.7**	**75.1**	**75.9**

Table 11: Average classification results across the five Multi-Fraud categories (%). F1 values retain the main-paper entries; Precision and Recall are reserved for completion using the same aggregation protocol.

	5%			20%			50%		
Method	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Tokeer	12.3	26.5	16.8	24.7	46.2	32.2	38.4	44.0	41.0
CRPWarner	36.9	20.7	26.5	45.4	36.2	40.3	32.1	51.0	39.4
Rug Pull Detection	42.4	32.2	36.6	55.0	45.0	49.5	60.6	56.4	58.4
RugScreener	29.4	10.6	15.6	31.1	13.5	18.8	72.5	53.9	61.8
RPHunter	38.6	33.9	36.1	47.6	28.9	36.0	45.1	68.5	54.4
UniCast	**43.1**	**41.3**	**42.2**	**60.9**	**58.7**	**59.8**	**75.8**	**73.6**	**74.7**

B.1.2 CROSS-DOMAIN CLASSIFICATION

Cross-domain comparisons examine whether the classification benefit depends on a particular task or input representation. Table 12 supplements the main-paper comparison with Precision and Recall on CIFAR-10H, CivilComments, Devign, and Multi-Fraud, retaining the same metric-averaging convention. EWC achieves the highest F1 on CivilComments, Devign, and Multi-Fraud and ties with 50-snapshot on CIFAR-10H, supporting its applicability beyond contract fraud.

Table 12: Cross-domain classification results (%). P and R denote Precision and Recall. F1 values are retained from the main paper; dashes reserve missing entries.

	CIFAR-10H			CivilComments			Devign			Multi-Fraud		
Classifier	P	R	F1	P	R	F1	P	R	F1	P	R	F1
50-snapshot	96.7	97.5	**97.1**	61.6	65.7	63.6	34.3	55.9	42.5	57.6	35.7	44.1
MCInfoNCE	97.4	93.1	95.2	48.0	53.1	50.4	25.5	41.2	31.5	27.9	31.7	29.7
DPE	82.6	91.7	86.9	71.4	68.5	69.9	60.3	37.8	46.5	47.0	35.0	40.1
AFR	87.3	77.5	82.1	71.0	64.5	67.6	60.5	42.2	49.7	62.3	40.5	49.1
HGAN4VD	88.5	92.0	90.2	68.8	62.3	65.4	72.8	68.2	70.4	65.7	71.1	68.3
SCL-CVD	92.1	83.2	87.4	67.9	64.8	66.3	58.1	69.5	63.3	73.0	65.6	69.1
EWC	**97.8**	**96.4**	**97.1**	**71.5**	**69.1**	**70.3**	**73.6**	**71.2**	**72.4**	**74.1**	**71.7**	**72.9**

B.2 RESULTS ON PONZI

Ponzi contracts provide a useful setting for distinguishing improved fraud recognition from warning before harm occurs, because their contract rules may already contain strong identifying information. We use contracts from the XBlock Smart Ponzi Dataset and additionally collect the first 100 transactions following each contract's deployment. The supplementary experiment uses contract features extracted with CASPER (Yang et al., 2025), augments them with transaction information, and applies EWC for classification.

The configuration without transaction input serves as the static reference, while the other configurations progressively increase the available observations. This experiment evaluates the supplementary Ponzi setting and should be distinguished from evaluation of the complete UniCast framework on Multi-Fraud and RPHunter.

The Ponzi experiment asks whether transaction observations remain useful when contract rules already provide strong identifying information. We compare a static configuration with configurations that progressively add transaction observations, retaining the CASPER-based contract features and the EWC classification mechanism.

Table 13: Ponzi detection with different observation ratios (%). The 0% configuration uses no transaction input.

Observation ratio	Precision	Recall	F1
0%	94.1	96.2	95.2
5%	95.1	96.2	95.7
20%	95.9	95.8	95.9
50%	98.1	97.7	97.9

B.3 REPRESENTATION ANALYSIS

Classification scores summarize prediction quality but do not directly describe the local relationships among sample representations. This analysis examines class overlap, within-class structure, and outliers on CIFAR-10H, CivilComments, and Devign through two-dimensional t-SNE projections. The projections are intended to complement quantitative evaluation rather than substitute for it. Figures 5, 6 show that EWC learn the best representation space among all tasks.



Figure 5: Two-dimensional t-SNE projections of sample representations on CivilComments. Colors denote non-toxic and toxic comments.



Figure 6: Two-dimensional t-SNE projections of sample representations on Devign. Colors denote non-vulnerable and vulnerable functions.

B.4 Parameter Sensitivity Analysis

Parameter analysis examines whether performance depends excessively on one particular configuration. We consider the prediction-loss weight λ, the number of anchors per class J, and the anchor-assignment temperature σ_a. They control the strength of cross-phase supervision, the capacity of the class-specific reference structure, and the concentration of sample-to-anchor weights, respectively. The default configuration is $\lambda = 0.1$, $J = 8$, and $\sigma_a = 0.2$. Figure 7 illustrates the impact of parameter variations on performance and demonstrates that the model achieves optimal performance under our default parameter settings.

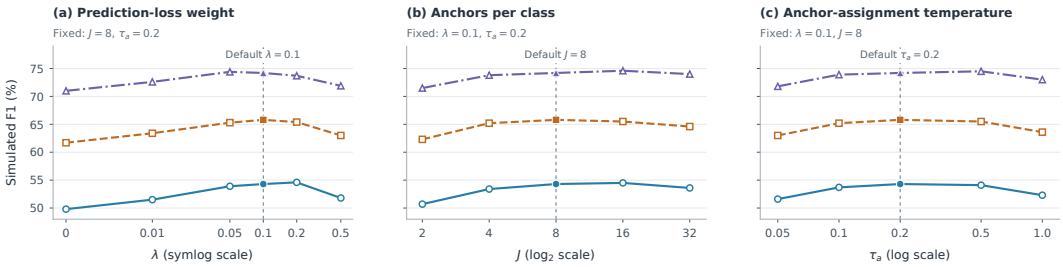

Figure 7: Parameter sensitivity at three observation ratios. Each panel varies one parameter while holding the others fixed; vertical markers indicate the default settings.

C Implementation Details

The main text introduces the roles of CPDM and EWC. A complete implementation also requires specifying how raw inputs become fixed features, how the networks process those features, and how prediction and classification supervision update the model. This section follows that computation from the input to the fraud score. Optimization and model selection are described separately in Section E.

C.1 Computing Environment and Runtime Measurement

Inference timing uses a batch size of one after the model has been loaded. Each contract and observation ratio receives one warm-up prediction and three measured predictions, with CUDA synchronization before and after timing. The measured interval includes deterministic feature construction and model inference but excludes blockchain data collection, disk loading, model loading, training, and logging. Latency is reported in milliseconds.

C.2 Datasets of Experiments

C.2.1 RPHunter

To evaluate detection on an external fraud dataset, we use the public dataset released with RPHunter (Wu et al., 2026). It contains 645 Rug Pull tokens and 1,675 benign tokens, totaling 2,320 samples, and covers Ethereum, BNB Smart Chain, Fantom, Base, Arbitrum, and Polygon. Its construction combines contract-code availability with transaction evidence from real-world token activity.

Unlike Multi-Fraud, this dataset focuses on Rug Pull detection. We use its released data with the splits and observation settings of the present study, rather than treating the original paper's reported results as results obtained under our protocol.

C.2.2 Cross-Domain Classification Data

To examine whether EWC is useful beyond contract fraud, we consider image, text, and code classification tasks. These experiments evaluate the classifier without assuming that the inputs provide paired contract and transaction observations.

CIFAR-10H. CIFAR-10H (Peterson et al., 2019) adds repeated human category judgments to the 10,000 images in the CIFAR-10 test set, covering ten classes. Approximately 50 judgments are available per image. The release provides both category counts and normalized label distributions, reflecting variation in how annotators interpret the same image.

CivilComments. CivilComments (Borkan et al., 2019) contains public comments posted between 2015 and 2017 on approximately 50 English-language news websites. The Jigsaw extension provides toxicity and related annotations. The default public release documented by TensorFlow Datasets contains 1,999,514 comments in total; annotation scores indicate the proportion of annotators assigning the corresponding attribute. We use this dataset for text-toxicity classification.

Devign. Devign (Zhou et al., 2019) is a function-level vulnerability-detection dataset. The commonly used CodeXGLUE release (Lu et al., 2021) contains 27,318 functions, each associated with source code, a sample identifier, and a binary vulnerability label. This task tests whether a classifier can distinguish vulnerable and non-vulnerable code despite similarities in local syntax and semantics.

C.3 Baseline

C.3.1 Contract Fraud Detection Methods

Tokeer. Tokeer (Zhou et al., 2024) is a static verification tool for identifying rug-pull risks in token contracts before potentially malicious operations are executed. It takes runtime bytecode as input and uses a component-configurable transfer model to describe different implementations of token transfer logic. Datalog rules identify predefined risk patterns within this model. Its analysis focuses on dangerous operations permitted by the contract rather than observed transaction behavior, representing a static approach to contract risk detection.

CRPWarner. CRPWarner (Lin et al., 2024) detects rug-pull risks arising from malicious contract functionality, including hidden minting, selling restrictions, and unauthorized token transfers. It decompiles EVM bytecode to recover control-flow and semantic relations, then applies Datalog-based information-flow analysis to characterize balance operations, access permissions, and transfer conditions. Predefined rules identify corresponding risk patterns. Unlike general vulnerability detection tools, CRPWarner focuses on functionality that contract creators may deliberately exploit.

Rug Pull Detection. Srifa et al. (Srifa et al., 2025) investigate the detection of rug-pull tokens from trading activity following token creation. Their method targets Uniswap V3 and constructs statistical features over different observation periods, including holder counts, transaction counts, token prices in USD, and trading volumes in USD. An XGBoost classifier distinguishes legitimate tokens from rug-pull tokens. By examining different observation periods and feature combinations, this work studies how accumulating transaction evidence affects detection performance.

RugScreener. RugScreener (Wu et al., 2025) identifies rug-pull risks in DeFi by modeling the structure and temporal evolution of on-chain transaction networks. It represents account interactions as a dynamic graph and combines memory and attention mechanisms within a temporal graph neural network to encode transaction relationships and historical activity. The original study learns temporal graph representations from unlabeled transactions and evaluates fraud detection using labeled tokens. Its distinguishing feature is the explicit modeling of evolving interactions rather than relying solely on aggregate transaction statistics.

RPHunter. RPHunter (Wu et al., 2026) jointly analyzes contract code and on-chain token flows to detect rug pulls involving code risks and transaction manipulation. Its code branch constructs a semantic risk code graph and uses a relational graph convolutional network to represent risky operations and their dependencies. Its transaction branch constructs a token-flow behavior graph and jointly aggregates node and edge information. The two representations are projected and combined through attention-based fusion before classification, providing a baseline that uses both static contract information and dynamic transaction evidence.

C.3.2 CLASSIFICATION AND UNCERTAINTY MODELING METHODS

50-snapshot. The 50-snapshot model is an experimental configuration of the higher-order calibration framework proposed by Ahdritz et al. (Ahdritz et al., 2025), which distinguishes uncertainty inherent in the data from uncertainty caused by incomplete model knowledge. The framework represents predictions as mixtures over label distributions and performs higher-order calibration using multiple conditionally independent label observations for the same input. Here, "50" denotes the number of label observations in a snapshot, rather than the number of training checkpoints. This method provides a comparison based on calibrated uncertainty estimates for ambiguous inputs.

MCInfoNCE. MCInfoNCE (Kirchhof et al., 2023) is a probabilistic contrastive learning method that represents an input as a distribution in latent space rather than a single deterministic vector. The model learns both the direction and concentration of the representation and uses Monte Carlo sampling to evaluate a probabilistic InfoNCE objective, allowing the representation to capture ambiguity in the underlying latent explanation. Under the theoretical assumptions established in the original study, this objective recovers the latent posterior distribution and its aleatoric uncertainty. MCInfoNCE therefore represents an approach to ambiguous inputs based on probabilistic representations.

DPE. Diverse Prototypical Ensembles (DPE) (To et al., 2025) address subpopulation shifts between training and target data, under which a single classifier may favor particular groups or features. The method replaces a standard linear classification layer with a mixture of prototypical classifiers that make predictions from the relationships between sample representations and class prototypes. Different ensemble members are trained to focus on different features and samples, producing diverse decision rules. DPE represents an approach to robust classification based on prototype structures and classifier diversity.

AFR. Automatic Feature Reweighting (AFR) (Qiu et al., 2023) reduces reliance on spurious features that are predictive in the training distribution but become unreliable under distribution shifts. It first trains a base model through empirical risk minimization, then retains the learned representations and retrains the final classification layer using a weighted loss. The weighting emphasizes examples that the base model predicts poorly, increasing attention to minority groups without requiring group labels for the training examples. AFR represents an approach to group robustness that adjusts how existing features contribute to classification.

HGAN4VD. HGAN4VD (Zhang et al., 2025) is a code vulnerability detection method based on heterogeneous graph attention networks, designed to capture program syntax, semantic relations, and long-range dependencies. It integrates abstract syntax trees, control-flow graphs, data-flow graphs, and program dependence graphs into a heterogeneous code graph, using graph simplification and meta-path construction to organize different relations. Node-level attention aggregates information from related nodes, while semantic-level attention combines representations associated with different meta-paths. The resulting representation supports function-level vulnerability classification.

SCL-CVD. SCL-CVD (Wang et al., 2024) combines supervised contrastive learning with a pretrained code model to distinguish code fragments that are semantically similar but differ in vulnerability status. It uses GraphCodeBERT to encode code and its data-flow information, with a supervised contrastive objective encouraging closer representations for samples with the same label and greater separation between different classes. R-Drop promotes prediction consistency, while LoRA enables parameter-efficient fine-tuning. A multilayer perceptron performs vulnerability classification from the learned representations.

D MODEL DETAILS

D.1 FIXED FEATURE CONSTRUCTION

Cross-phase modeling first requires establishing the correspondence between static contract information and dynamic observations. Each sample is identified by its blockchain and contract address.

Its static input is the runtime bytecode at deployment, and its dynamic input consists of the associated ERC-20 transfer events within the current observation window. Static features are aggregated over the entire contract without function-level slicing. Dynamic observations are organized at the event level rather than the transaction level, so multiple transfer events within the same transaction are retained separately. Events are ordered by block number, transaction index, and log index, with their ordering checked for consistency.

Contracts vary in code length, making raw byte sequences unsuitable as a uniform statistical input. The static encoder therefore summarizes runtime bytecode using opcode frequencies, indicators of selected operations, and code size. Let O denote the parsed opcode sequence and L the bytecode length. The raw static feature vector is

$$\mathbf{f}_S = \left[\left(\frac{\#\{o \in O : o = v\}}{|O|} \right)_{v=0}^{255} ; (\mathbb{I}[o \in O])_{o \in \mathcal{O}} ; \log(1 + L) \right], \tag{14}$$

where $\mathcal{O}$ contains `SSTORE`, `CALL`, `DELEGATECALL`, `STATICCALL`, `SELFDESTRUCT`, `CREATE`, and `CREATE2`. The three components contribute 256, 7, and 1 dimensions, respectively, yielding a 264-dimensional vector. Immediate bytes following `PUSH1` through `PUSH32` are skipped during parsing to avoid counting instruction arguments as opcodes. The current implementation does not heuristically remove compiler metadata. Opcode indicators describe the operations present in the code rather than assigning them a malicious interpretation in advance; their discriminative value is learned jointly with transaction behavior.

Dynamic features must describe both the current event and the interactions observed by the time it occurs. For the j-th transfer event, the dynamic encoder constructs a 16-dimensional vector. Transfer quantities are represented by the logarithms of the current raw amount and the cumulative raw amount. Five indicators describe whether the sender is the zero address, the receiver is the zero address, the transfer is a self-transfer, the sender is the deployer, or the receiver is the deployer. Additional features indicate whether the sender and receiver have appeared previously, the current sender's cumulative outgoing-event fraction, the current receiver's cumulative incoming-event fraction, and the logarithm of the number of distinct observed addresses. The remaining four dimensions describe block position, inter-event spacing, and event position within the window.

These statistics are computed in event order. Both outgoing- and incoming-event fractions use the cumulative event count i as their denominator, and their numerators include the current event. First-occurrence indicators are determined using the union of previously observed sender and receiver addresses. The deployer address is used only to construct relationship indicators and is not directly encoded as an input feature. When deployer information is unavailable, the corresponding indicators are set to zero. Transfer amounts use the raw integers recorded in the events, with $\log(1+x)$ applied to compress their numerical range; they are not converted using token prices or expressed in US dollars.

To ensure that temporal positions depend only on the currently available observations, let b_d be the deployment block, q the query block, and n the number of events in the current window. Define $D(q) = \max(1, q - b_d)$. The four-dimensional position feature for event i is

$$\mathbf{f}_{\mathrm{pos},i} = \left[\frac{b_i - b_d}{D(q)} ; \frac{b_i - b_{i-1}}{D(q)} ; \frac{i}{\max(n, 1)} ; \log(1 + i) \right], \tag{15}$$

where $b_0 = b_d$ for the first event. Relative positions are recomputed when the query window changes. Thus, fixed feature construction means that the construction rules are not learned; it does not mean that every feature of a historical event remains unchanged across query windows. Harmful-event timestamps, retrospective reports, and fraud labels are excluded from this construction process.

D.1.1 Standardization and CPDM Computation

The input features have different numerical scales: opcode frequencies, transfer amounts, and block intervals cannot be interpreted on the same scale directly. Static and dynamic features are therefore standardized separately, producing the features $\mathbf{F}_S$ and $\mathbf{F}_{D,j}$ used in the main text:

$$\mathbf{F}_S = \frac{\mathbf{f}_S - \boldsymbol{\mu}_S}{\boldsymbol{\sigma}_S^*}, \qquad \mathbf{F}_{D,i} = \frac{\mathbf{f}_{D,i} - \boldsymbol{\mu}_D}{\boldsymbol{\sigma}_D^*}. \tag{16}$$

All operations are coordinate-wise. Means and standard deviations are estimated exclusively from the training data and remain fixed during validation and testing. For dimensions with a standard deviation no greater than 10^{-6}, the scaling factor is replaced by one to avoid amplifying nearly constant features. Dynamic statistics are estimated from real events in the training windows, excluding padding. An event that appears in multiple training windows contributes through each corresponding window. The targets used for bidirectional prediction undergo the same standardization.

The static representation network maps the 264-dimensional input to a 128-dimensional representation. It uses two linear layers with LayerNorm, GELU, and Dropout between them. For an MLP with input, hidden, and output dimensions a, h, and b, respectively, the computation is

$$\mathrm{MLP}_{a,h,b}(\mathbf{x}) = W_2\ \mathrm{Dropout}\left(\mathrm{GELU}\left(\mathrm{LN}(W_1\mathbf{x} + \mathbf{b}_1)\right)\right) + \mathbf{b}_2. \tag{17}$$

The static network uses dimensions $264 \to 128 \to 128$ and a dropout rate of 0.1. The subsequent representation projections, prediction heads, and two-layer MLPs in the classifier follow the same basic structure while maintaining their own parameters.

Dynamic representations must preserve event order and contextual relationships. Each 16-dimensional event vector is first linearly projected to 128 dimensions and augmented with fixed sinusoidal positional encoding. A two-layer Transformer then processes these inputs. Each layer uses four attention heads, a feed-forward dimension of 512, pre-layer normalization, GELU activation, and a dropout rate of 0.1. Self-attention operates over the entire observed window without a causal mask, but no events beyond the query point are supplied.

The window-level dynamic representation is the Transformer output at the last real event. Since this position can attend to other observed events in the window, no additional classification token or mean pooling is used. Variable-length sequences are right-padded within each batch. A mask excludes padding from attention and identifies the final valid output using the real event count. Only real events are standardized; padded positions remain zero.

An empty window contains no event from which to obtain a contextual summary, so a padding vector must not be treated as a real transaction. For such samples, the model bypasses the Transformer and uses a zero-initialized, learnable 128-dimensional empty-context vector as the dynamic representation. This vector still passes through the dynamic projection and participates in classification, allowing the model to use static information together with the absence of dynamic observations.

To preserve information from both sources while learning their relationship, the static and dynamic representations pass through independent $128 \to 128 \to 64$ projection networks and are concatenated into a 128-dimensional fused representation. Their output dimensions are identical, but their values are not constrained to be equal. The classifier receives these projected representations rather than the predicted features produced by the prediction heads.

D.2 Prediction Heads and Supervision Paths

Bidirectional prediction requires specifying which target corresponds to the current representation. Static-to-dynamic prediction uses the fixed feature of the last real event in the current window as its target and distinguishes observation windows through the target position. This position is encoded as the scalar $e(n) = \log(1 + n)$ rather than the raw event count. The two prediction directions are implemented as

$$\begin{aligned} \widehat{\mathbf{F}}_{D,n} &= \mathrm{MLP}_{129,128,16}\left([\mathbf{H}_S; e(n)]\right), \\ \widehat{\mathbf{F}}_{S} &= \mathrm{MLP}_{128,128,264}(\mathbf{H}_D). \end{aligned} \tag{18}$$

The static-to-dynamic head therefore receives the 128-dimensional static representation and a one-dimensional position encoding, producing a 16-dimensional prediction. The dynamic-to-static head produces a 264-dimensional vector matching the static target. The heads have independent parameters, and the dynamic target always belongs to the current window rather than the end of the complete history.

Contrastive matching concerns the correspondence between a contract and its observations, so positive pairs are determined by contract identity rather than fraud class. Each contract contributes one observation window per training batch. Its own target forms the positive pair, while targets from other contracts provide contrastive candidates. Predicted vectors and fixed targets are unit-normalized before cosine similarities are computed, with a contrastive temperature of 0.1. The

matching objective is specified in Section E.1; it is not coordinate-wise mean squared reconstruction error.

Fixed targets provide stable supervision for representation learning and are not updated by the prediction losses. The static-to-dynamic loss updates the static representation network and its prediction head. The dynamic-to-static loss updates the event projection, Transformer, and corresponding prediction head. Classification gradients propagate through EWC and the fusion projections to both representation networks, allowing the final discrimination task to guide cross-phase learning. Since the fusion projections do not lie on the prediction paths, they do not directly receive gradients from the prediction losses.

Empty windows have no last-event target and therefore contribute only to classification supervision. When fewer than two samples in a batch contain dynamic observations, both prediction losses are set to zero, avoiding an uninformative matching task without valid contrastive candidates. These samples continue to contribute to the classification loss.

D.3 EWC Reference Structures and Normalization

To evaluate the same sample against two class-specific reference structures, EWC first transforms the fused representation through a shared $128 \to 128 \to 64$ projection and then applies two $64 \to 128 \to 64$ class mappings. The mappings have the same architecture but independent parameters, and both receive the complete fused information. They do not correspond separately to the static and dynamic branches, and no additional orthogonality or statistical-independence constraint is imposed.

Each class space contains eight learnable 64-dimensional anchors, initialized from a zero-mean Gaussian distribution with a standard deviation of 0.02. The anchors are neither preselected samples nor fixed initializations derived from class means or clustering. Instead, their reference locations are learned under classification supervision. At each forward pass, class representations and anchors are independently unit-normalized before the squared Euclidean distances and distance-based weights defined in the main text are computed. The assignment temperature is 0.2, giving closer anchors greater weight while retaining contributions from other reference locations.

The anchor sets in the two spaces may have different dispersions, so unit normalization alone does not remove the need for distance-scale correction. The implementation estimates the reference scale using the mean squared distance between distinct anchors within each class. Eight anchors produce 28 unordered pairs, excluding self-distances. This calculation is equivalent to the mean over all ordered distinct pairs in the main text. The scale is detached when used to normalize the aggregated distance, and 10^{-8} is added to the denominator. No additional clipping is applied.

Stopping gradients only blocks direct backpropagation through the scale denominator; it does not freeze the anchors. They continue to receive gradients through sample-to-anchor distances and their aggregation weights. The underlying anchor parameters are also not permanently constrained to unit length. Instead, they are normalized again during each forward pass.

The output scale must remain positive so that a smaller distance to the fraudulent reference structure consistently corresponds to a higher fraud score. The model therefore parameterizes this scale using a learnable scalar a:

$$\beta = \mathrm{softplus}(a) + \epsilon_\beta, \tag{19}$$

where ϵ_β is a small numerical constant ensuring strict positivity. The parameter a is initialized so that β is approximately one. No additional bias is applied to the final logit, whose sign depends only on the relative normalized distances. This absence of an additive bias concerns the final distance-to-logit transformation; the linear layers within the MLPs retain their bias parameters.

additional ranking, anchor-balance, or orthogonality losses.

D.4 Cross-Domain Adaptation

Cross-domain experiments involve different numbers of classes, so the extension of EWC preserves the principle of learning a separate reference structure for each class. For K classes, the shared projection is followed by K independent mappings and their associated anchor sets. Distance aggregation and scale normalization remain unchanged, and negative normalized distances serve as

class scores:

$$p_c = \frac{\exp(-\beta \widetilde{d}_c)}{\sum_{k=1}^{K} \exp(-\beta \widetilde{d}_k)}. \tag{20}$$

The model is trained using multiclass cross entropy and predicts the class with the highest probability. For two classes, this formulation is equivalent to the distance-difference sigmoid decision in the main text, so the extension does not require changing the underlying distance-comparison principle.

These experiments evaluate the classifier's applicability using feature representations from the corresponding tasks, rather than paired contract and transaction observations. Cross-domain classification therefore does not use the contract-specific bidirectional prediction losses and should be distinguished from joint training of the complete fraud detection model.

For comparisons based on fixed representations, CIFAR-10H uses 512-dimensional ResNet18 global-average-pooling features, CivilComments uses 768-dimensional BERT classification-token features, and Devign uses 768-dimensional GraphCodeBERT classification-token features. The corresponding feature extractors are not further fine-tuned on the target dataset in this setting. Following the main-paper protocol, a standalone classification head is attached to the task representation when it can be separated from the baseline; otherwise, the complete method is evaluated with its classification mechanism. The latter comparison is not an identical-backbone comparison of classification heads.

E Training Details

Joint optimization allows cross-phase correspondence and fraud discrimination to shape the same representations. We train CPDM and EWC together from initialization, using the objectives in Section E.1, without a separate pretraining stage. The deterministic feature encoders and training-set standardization statistics remain fixed, while the representation networks, prediction heads, projections, class mappings, and anchors are learned.

E.1 Loss Functions

Cross-phase correspondence and fraud discrimination require complementary supervision. Predictive matching encourages the representations to retain information relevant to the other phase, while classification supervision guides the fused representation toward the final decision.

Let $\mathcal{B}$ denote a training batch with one observation window per contract, and let $\mathcal{V} \subseteq \mathcal{B}$ contain samples with observed dynamic events. For sample i, n_i is the number of observed events, $\widehat{\mathbf{F}}_{D,i}$ predicts the feature of its last event at position n_i, and $\widehat{\mathbf{F}}_{S,i}$ predicts its contract feature. Its own contract and event targets form the paired targets, while targets from other contracts provide contrastive candidates. The CPDM objective is

$$\begin{aligned}\mathcal{L}_{\mathrm{CPDM}} = -\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \Bigg[& \log \frac{\exp\Big(s(\widehat{\mathbf{F}}_{D,i}, \mathbf{F}_{D,i,n_i})/\tau\Big)}{\sum_{j \in \mathcal{V}} \exp\Big(s(\widehat{\mathbf{F}}_{D,i}, \mathbf{F}_{D,j,n_j})/\tau\Big)} \\ & + \log \frac{\exp\Big(s(\widehat{\mathbf{F}}_{S,i}, \mathbf{F}_{S,i})/\tau\Big)}{\sum_{j \in \mathcal{V}} \exp\Big(s(\widehat{\mathbf{F}}_{S,i}, \mathbf{F}_{S,j})/\tau\Big)} \Bigg],\end{aligned} \tag{21}$$

where s is cosine similarity after unit normalization and $\tau = 0.1$. The two terms equally weight static-to-dynamic and dynamic-to-static matching. Pairing is determined by contract-observation correspondence rather than by the fraud label.

EWC is supervised by contract-level labels using binary cross-entropy:

$$\mathcal{L}_{\mathrm{EWC}} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right]. \tag{22}$$

Here, $y_i \in \{0, 1\}$ is the contract label, with 1 denoting fraud, and p_i is its predicted fraud probability. For numerical stability, the implementation computes this objective directly from logits rather than

explicitly applying sigmoid followed by logarithms. The joint objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{EWC}} + \lambda \mathcal{L}_{\mathrm{CPDM}}, \qquad \lambda = 0.1. \tag{23}$$

Fixed targets remain unchanged. Classification gradients propagate through EWC and the fusion projections into the representation networks, while each predictive objective updates its corresponding representation branch and head. Empty windows contribute only to classification; when fewer than two samples have valid dynamic targets, the CPDM loss is set to zero. The current objective does not introduce

E.2 Data Splits

To separate parameter learning, model selection, and final evaluation, the samples included in each experiment are divided into training, validation, and test sets in a 6:2:2 ratio. For contract data, every observation window associated with the same contract remains in the same split. Feature-standardization statistics are estimated using training data only and remain fixed during validation and testing.

E.3 Observation Windows and Batch Construction

A contract observed with different amounts of evidence should be evaluated using a consistent decision mechanism. We therefore train one model per random seed across the three observation ratios of 5%, 20%, and 50%, rather than assigning a separately trained model to each ratio. During each epoch, every training contract contributes its observation window once at each ratio. Contracts are independently shuffled within each ratio, and each batch contains windows from one ratio without repeated contract identities.

All observation windows from a contract remain in the same dataset partition. Within a batch, variable-length sequences are right-padded to the longest sequence, and the padding mask excludes the artificial positions from attention and last-event selection. Windows without observed events use the learned empty-context representation and remain eligible for classification supervision, but do not supply targets for bidirectional prediction.

E.4 Optimization

To optimize correspondence learning without replacing the classification objective, we use a representation-loss weight of $\lambda = 0.1$, with equal weights for the two prediction directions. AdamW optimizes the learnable parameters with a learning rate of 3×10^{-4}, weight decay of 10^{-4}, and a batch size of 64. Training runs for 30 epochs, with gradient-norm clipping at 1.0 and a dropout rate of 0.1 in the representation and prediction networks.

The learning rate remains constant throughout training; neither early stopping nor a learning-rate scheduler is used. Computation uses FP32 without automatic mixed precision. We run each experiment three times with random seeds 2026, 2027, and 2028 and report the arithmetic mean of the independently computed evaluation metrics. Seeds control parameter initialization and sample shuffling, but do not imply bitwise equality across different hardware or software environments.

E.5 Inference and Warning

At inference time, detection uses only the information available in the current observation window. Static features and observed dynamic events pass through their respective representation networks and projections, after which their concatenation is evaluated by EWC. Let $\widetilde{d}_0$ and $\widetilde{d}_1$ denote the normalized distances to the benign and fraudulent reference structures, respectively. The decision score and fraud probability are

$$r = \widetilde{d}_0 - \widetilde{d}_1, \qquad p = \mathrm{sigmoid}(\beta r). \tag{24}$$

The model predicts fraud when $r > 0$. Applying the rule to the distance difference avoids relying on a numerically saturated sigmoid output. Inference is performed in evaluation mode without gradient computation.

The two prediction heads are not executed during inference: their effect is retained in the learned representations. Predicted dynamic features do not replace observed events, and predicted static features do not replace the contract input. As new events arrive, the dynamic features and contextual representation can be recomputed to update the risk score. The first query satisfying the decision rule is recorded as the first alert. Whether this alert precedes a harmful event is evaluated against an independently established incident time, rather than inferred from the predicted class itself.

F Limitations and Future Work

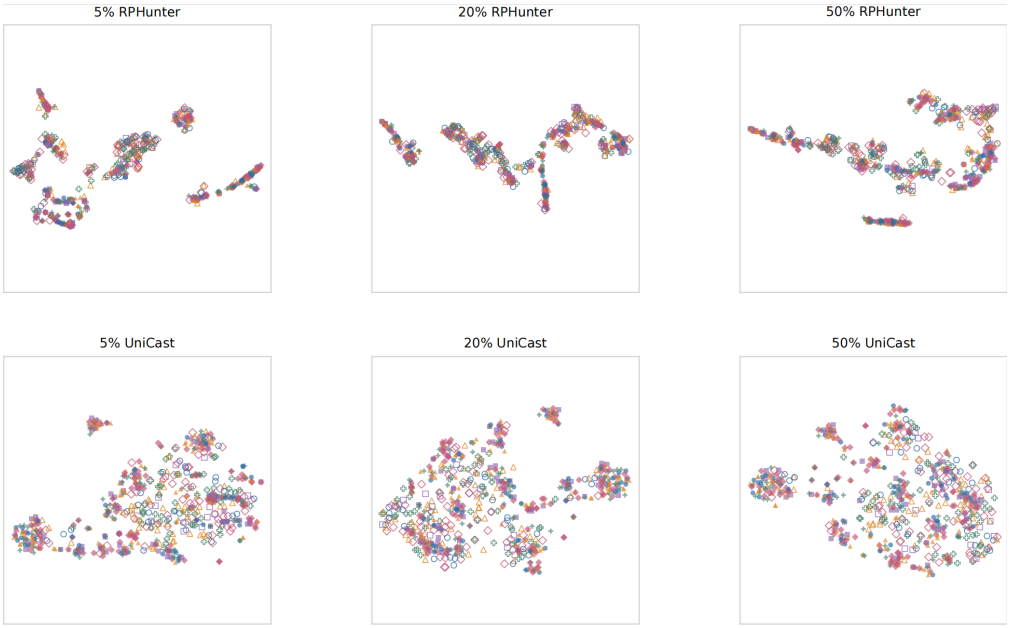

Figure 8: t-SNE plots of feature distributions across multiple categories for RPHunter and UniCast under different time windows

This paper primarily discusses **UniCast**'s capability for early fraud detection, but does not focus mainly on its ability to identify any specific category. The main reason lies in the fact that, as a work based on deep learning, the current dataset (Table 1) size does not allow us to effectively learn significant differences among various sample categories.

We also attempted to validate the potential of our method in multi-class classification. As shown in Figure 8, the five categories are highly confused in the feature space, and there is no relatively sparse region available to separate them. However, compared to RPHunter, our sample space is sparser and the samples are less likely to be confused with each other. This indicates that our method has sufficient potential to accomplish multi-classification tasks, which will also become the focus of our future work.