

CE2Graph: A Diagnosis-Oriented Spatio-Temporal Graph Framework for DRAM Failure Prediction

Cong Ma¹, Xinyi He², Jianwang Zhai¹, Kang Zhao^{1,†}

¹Beijing University of Posts and Telecommunications, China

²China Electronics Engineering Design Institute Co., Ltd., China

{mc, zhaijw, zhaokang}@bupt.edu.cn, hexinyi@sdic.com.cn

Abstract—Dynamic random access memory (DRAM) failures, especially uncorrectable error (UE) and correctable-error (CE) storms, represent a major cause of node outages in modern data centers. Existing DRAM failure prediction methods mainly rely on flattened log features or handcrafted spatio-temporal statistics, which are insufficient for capturing the structured propagation of CEs across the hierarchical physical organization of DRAM. This paper proposes CE2Graph, a diagnosis-oriented spatio-temporal graph framework for DIMM-level DRAM failure prediction from correctable-error logs. CE2Graph transforms irregular CE logs into a graph representation, where nodes encode CE events along with physical-location metadata and error-bit patterns (e.g., DQs and Beats), while edges capture temporal adjacency and structural relations induced by shared DRAM components such as rows, columns, and banks. This representation enables explicit modeling of fault locality, temporal evolution, and bursty CE-storm behaviors under weakened error-correcting code (ECC) protection. Based on CE2Graph, we further develop a gated graph neural network (GGNN) with max-based neighborhood aggregation to emphasize severe localized hardfault patterns. Experimental results on large-scale real-world DRAM error and failure logs show that the proposed method consistently outperforms baseline models, including XGBoost and LSTM, in predicting future DRAM failures. These results suggest that graph-based spatio-temporal modeling is a promising direction for memory diagnosis and failure prediction.

Index Terms—DRAM Failure Prediction, GNN, Spatio-Temporal Modeling

I. INTRODUCTION

Dynamic Random Access Memory (DRAM) is a critical component in modern data centers and high-performance computing (HPC) systems. However, DRAM failures have become one of the leading hardware causes of server crashes and node unavailability [1], [2]. To improve memory reliability, modern servers are commonly equipped with Error Correction Code (ECC) mechanisms, such as SEC-DED and Chipkill, to detect and correct early-stage or transient memory errors, referred to as Correctable Errors (CEs) [3]. Nevertheless, as DRAM technology continues to scale, physical wear-out and latent hardware defects can accumulate over time, resulting in persistent hard faults. Once the error pattern exceeds the correction capability of ECC, the fault may escalate into an Uncorrectable Error (UE), leading to system crashes, service

interruptions, and substantial operational losses [4], [5]. Field studies have shown that many DRAM failures are preceded by CE symptoms. In particular, CE storms, i.e., the occurrence of a large number of CEs within a short time interval, are an important signal of impending node unavailability [6]. Therefore, predicting DRAM failures from historical CE logs is an important problem for improving system reliability and enabling proactive maintenance [1].

In recent years, both academia and industry have proposed various machine learning-based methods for DRAM failure prediction, including support vector machines, random forests, XGBoost, and sequential models [7], [8]. However, these methods still have important limitations in practice. Most existing approaches rely on manual feature engineering and flatten multidimensional error logs into simple statistical features, such as the number of CEs observed within a fixed time window [9]. Such flattened representations are often insufficient to capture the structured propagation of failures over the hierarchical physical organization of DRAM. As shown in Fig. 2, DRAM faults exhibit strong spatial locality, for example across rows, columns, and banks that share physical structures, which can lead to CEs [3], [6], [10]. In addition, under weakened ECC protection in modern servers, low-level error-bit patterns, such as DQs and Beats, have been shown to provide important clues for predicting future UEs [4], [11]. These micro-architectural semantics are often ignored in conventional predictors. Prior physics-informed learning studies also suggest that incorporating the physical structure of DRAM into prediction models can be more effective than using purely flattened statistical representations [10].

Motivated by these observations, this paper proposes CE2Graph, a diagnosis-oriented spatio-temporal graph framework for DIMM-level DRAM failure prediction. Instead of representing CE logs as flattened feature vectors, CE2Graph converts irregular CE events into a graph representation that jointly models temporal evolution, spatial locality, and DRAM topology [12]. Specifically, nodes encode CE events together with physical-location metadata and low-level error-bit patterns, such as DQs and Beats [11]. Edges capture both temporal adjacency between nearby CE events and structural relations induced by shared DRAM components, such as the same bank, row, or column [3]. In this way, CE2Graph explicitly models localized fault propagation and bursty CE-

This work is supported by the Beijing Natural Science Foundation (No. L2603021), the Fundamental Research Funds for the BUPT (No. 2025A14S20), the Research Initiation Project for Introduced Talents of BUPT (No. 510124058, 510224062). (Corresponding author: Kang Zhao.)

storm behavior over the hierarchical physical organization of DRAM. Based on the constructed graph, we further develop a Gated Graph Neural Network (GGNN) [12] with max-based neighborhood aggregation to emphasize severe localized hard-fault patterns while reducing the influence of sporadic soft-error noise. The main contributions of this paper are as follows:

- We propose CE2Graph, a spatio-temporal graph representation that transforms irregular DIMM-level CE logs into a structured prediction problem over the hierarchical DRAM topology.
- We develop a GGNN-based DRAM failure predictor that jointly models temporal evolution, spatial locality, and micro-level error-bit patterns.
- We conduct preliminary evaluations on a large-scale real-world DRAM failure dataset and compare the proposed method with baselines such as XGBoost and LSTM.

II. RELATED WORK

Large-scale field studies have shown that DRAM errors are not isolated random events, but often exhibit spatial clustering, temporal concentration, and strong correlations between CEs and future failures [5], [13], [14]. As DRAM technology scales, modern memory modules (e.g., DDR4) exhibit increasingly complex failure modes. Recent DDR4 studies further report diverse multi-bit and component-level fault modes [15]. At the feature level, Error-bit patterns such as DQs and Beats, as well as CE storms, have also been identified as important indicators of future UEs [4], [6]. These empirical findings, rooted in low-level hardware characteristics, provide a solid physical rationale for our approach of utilizing graph structures to capture both spatial fault propagation and temporal evolution of CEs.

ML-based DRAM failure prediction has also been widely studied. Giurgiu et al. utilized Random Forests combined with a sliding window technique to predict UEs based on system events and CE logs [16]. Wu et al. and Yu et al. formulated DRAM failure prediction as a multi-class classification problem, extensively evaluating tree-based models on large-scale cloud datasets [8], [17]. To capture evolutionary trends within time series, Du and Li proposed online learning approaches to predict failures at micro-level (e.g., Cell, Row) [7]. Sun et al. observed that traditional RNNs and LSTMs are overly sensitive to specific time delays between faults, prompting them to design Temporal Convolutional Neural Networks (Temporal CNNs) to process irregular sequential log [17]. Despite their efficacy in specific scenarios, these methods heavily rely on manual feature engineering, typically flattening multidimensional error logs into one-dimensional statistical features (e.g., simply counting the total number of errors within a specific time window).

Integrating prior knowledge of a system’s physical topology into machine learning models has been proven to effectively enhance predictive performance. Baseman et al. introduced a physics-informed machine learning approach for DRAM error modeling. By encoding the dependencies among adjacent

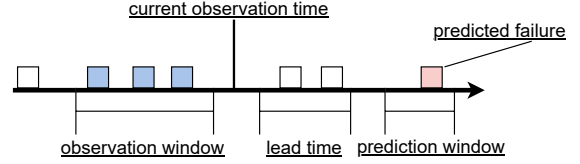


Fig. 1 Memory failure prediction problem formulation.

physical locations, identical rows, and identical columns in DRAM, they demonstrated that their topology-aware model significantly outperforms traditional, “physics-ignorant” statistical models [10]. Gorle et al. proposed the G-MAP framework, G-MAP shows that graph representations can capture spatio-temporal locality in memory access sequences. [12]. Diverging from traditional time-series models, our proposed CE2Graph directly maps discrete CE events, underlying error-bit patterns, physically shared components (Rank / Chip / Bank / Row / Column), and temporal progression into a unified graph structure. This provides a topology-aware angle for DRAM failure prediction that better aligns with the hierarchical physical organization of memory systems.

III. METHODOLOGY

Fig. 2 gives an overview of CE2Graph. Given CE logs within an observation window, CE2Graph first encodes each CE event with physical-location and DQ-Beat error-bit features, then constructs a spatio-temporal graph using topology-aware spatial edges and temporal edges, and finally applies a GG-NN-based predictor to estimate future DRAM failure.

A. Problem Formulation

As shown in Fig. 1, we formulate DRAM failure prediction as a lead-time-aware binary classification task. For a target DIMM d , the input is the set of CE events observed within an observation window:

$$E_d(t) = \{e_i = (t_i, L_i, B_i) \mid t_i \in [t - \Delta t_{\text{obs}}, t]\}_{i=1}^N, \quad (1)$$

where t_i is the timestamp, L_i denotes the physical location in the DRAM hierarchy, and B_i denotes the DQ-Beat error-bit pattern. The target label is defined as

$$y_d(t) = \begin{cases} 1, & \text{if DIMM } d \text{ fails within} \\ & [t + \Delta t_{\text{lead}}, t + \Delta t_{\text{lead}} + \Delta t_{\text{pred}}], \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

CE2Graph constructs a graph $G_d(t)$ from $E_d(t)$ and learns a predictor $\hat{p}_d(t) = f_\theta(G_d(t))$.

B. Node Representation and Error-Bit Feature Encoding

In the CE2Graph framework, the discrete CE event sequence E_d within an observation window is mapped to the node set of a graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_N\}$. Each node $v_i \in V$ corresponds to one CE event e_i in a one-to-one manner. Fig. 2 illustrates the process of constructing feature vector, we construct an node feature vector $x_i \in \mathbb{R}^F$

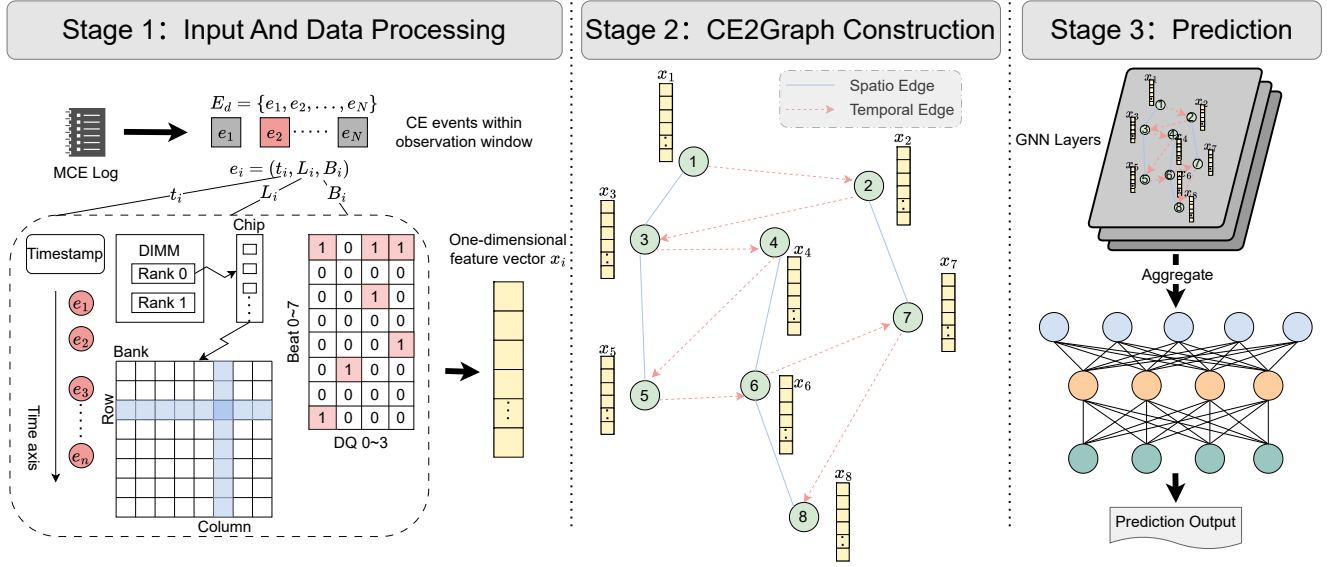


Fig. 2 Spatio-temporal graph memory failure prediction framework.

by jointly encoding the physical location and the micro-level error-bit pattern of each CE event.

1) *Micro-level DQ-Beat Matrix Representation*: DRAM failures in modern data centers are closely related to the spatial distribution of error bits. In this paper, DQ denotes the data I/O lane of a DRAM chip, and Beat denotes the position within a burst transfer. For an x4 DDR4 device, a memory transfer can be abstracted as 4 DQ lanes over 8 burst beats [4], [18]. Based on this hardware organization, we represent the micro-level error-bit pattern associated with a CE event as a two-dimensional binary matrix

$$B_i \in \{0, 1\}^{8 \times 4}, \quad (3)$$

where the rows correspond to the 8 burst beats and the columns correspond to the 4 DQ lanes. Each entry B_i indicates whether an error bit is observed at the corresponding DQ-Beat position. Specifically, $B_i = 1$ means that the CE event contains an error bit at the corresponding Beat-DQ position, and $B_i = 0$ otherwise. This representation preserves the spatial layout of the error-bit manifestation and provides fine-grained clues about the underlying fault behavior. For example, when an error-bit pattern spans multiple DQs or multiple Beats, it is more likely to exceed the correction range of ECC and become a risky CE pattern that may evolve into a future Uncorrectable Error (UE) [11]. Therefore, directly embedding the spatial morphology of B_i into the node representation is important for predicting future DRAM failures.

2) *Node Feature Formulation*: To jointly capture both the error-bit pattern and the macro-level physical topology, we construct a composite feature representation for each node v_i . For the i -th CE event, its physical location is denoted as

$$L_i = (\text{Rank}_i, \text{Chip}_i, \text{Bank}_i, \text{Row}_i, \text{Col}_i). \quad (4)$$

and we flatten the two-dimensional DQ-Beat matrix B_i into a one-dimensional binary vector

$$\text{vec}(B_i) \in \{0, 1\}^{32}, \quad (5)$$

Meanwhile, the physical address tuple L_i is fed into a multi-layer perceptron (MLP) to obtain a learnable nonlinear encoding of the event location. The initial node feature vector is then defined as

$$x_i = \text{vec}(B_i) \oplus \text{MLP}_\theta(L_i), \quad (6)$$

where $\oplus$ denotes vector concatenation and MLP_θ denotes the location-encoding network parameterized by θ .

With each CE event encoded as a high-dimensional node feature that preserves its micro-level error morphology, the next critical step is to construct the topological connections that capture how these isolated events interact and propagate over time and space, which is detailed in Section III-C.

C. Spatio-Temporal Graph Construction

Given the node set V , this section defines the edge set E of CE2Graph to capture both the spatial propagation of DRAM faults over the physical topology and the temporal evolution of CE events. In CE2Graph, we construct a weighted adjacency matrix $A \in \mathbb{R}^{N \times N}$, where each entry represents the relation strength between two CE events. The adjacency weight is defined as the combination of a spatial component and a temporal component:

$$A_{i,j} = A_{i,j}^{\text{spat}} + A_{i,j}^{\text{temp}}. \quad (7)$$

1) *Spatial Edges*: The purpose of spatial edges is to model the propagation of hard faults in the physical layout of the DRAM. To explicitly encode spatio features into the graph, we introduce undirected spatial edges between CE nodes that are structurally related in the DRAM hierarchy. In particular,

for two nodes v_i and v_j ($i \neq j$), the spatial adjacency is defined as

$$A_{i,j}^{\text{spat}} = I(\text{Bank}_i = \text{Bank}_j) \cdot \left[w_{\text{row}} I(\text{Row}_i = \text{Row}_j) + w_{\text{col}} I(\text{Col}_i = \text{Col}_j) \right], \quad (8)$$

where $I(\cdot)$ is the indicator function, and w_{row} and w_{col} are learnable weights that control the relative contributions of row-sharing and column-sharing relations. Under this formulation, two CE events are spatially connected only when they belong to the same bank and share either the same row or the same column. Such spatial edges allow CE2Graph to encode row-fault and column-fault patterns, which are important manifestations of spatio-temporal fault patterns in DRAM.

2) *Temporal Edges*: In addition to spatial locality, DRAM failures also exhibit temporal concentration. Temporal edges are introduced to model the dynamic deterioration process of DRAM faults, especially the bursty phenomenon.

To enable the graph model to capture short-term temporal escalation, we create directed temporal edges between nodes that are close in time [12]. For a node pair (v_i, v_j) , if event e_j occurs after event e_i , i.e., $t_j > t_i$, and their time interval is smaller than a predefined storm-awareness threshold τ_{storm} , then a directed temporal edge from v_i to v_j is added. The temporal adjacency weight is defined as:

$$A_{i,j}^{\text{temp}} = \begin{cases} \exp(-\lambda(t_j - t_i)), & \text{if } 0 < t_j - t_i \leq \tau_{\text{storm}}, \\ 0, & \text{otherwise,} \end{cases} \quad (9)$$

where λ is a parameter controlling the decay rate over time. We set $\tau_{\text{storm}} = 60$ s, and the exponential decay assigns larger weights to events that occur closer in time and progressively discounts older interactions.

3) *Graph Construction*: Combining the spatial and temporal relations above, as Fig. 2 shows, CE2Graph transforms the irregular CE event sequence into a weighted spatio-temporal graph

$$G = (V, E, A), \quad (10)$$

where V is the set of CE-event nodes, E is the set of spatial and temporal edges, and A is the corresponding weighted adjacency matrix.

Having established the topology of CE2Graph, the raw sequential logs are successfully transformed into a unified spatio-temporal structure. The final stage is to leverage this graph structure to learn predictive fault patterns and generate failure alerts, as formulated in Section III-D.

D. GG-NN-Based Spatio-Temporal Failure Prediction

After constructing the spatio-temporal graph $G_d(t)$, we employ a Gated Graph Sequence Neural Network (GG-NN) to perform feature propagation and state evolution over CE2Graph. Compared with conventional graph convolution layers, GG-NN integrates gated recurrent units (GRUs) into graph message passing, making it well suited for modeling

Algorithm 1: GG-NN-based spatio-temporal learning on CE2Graph

Input: Spatio-temporal graph $G_d(t) = (V, E, A)$; node features $\{x_i\}_{v_i \in V}$; number of layers L ; Model parameters $\Theta = \{W^{(l)}, W_z, U_z, W_r, U_r, W_h, U_h, \phi\}$
Output: Failure probability $\hat{p}_d(t)$ and loss $\mathcal{L}_d$

```

1 foreach  $v_i \in V$  do
2   Initialize node hidden state:  $h_i^{(0)} \leftarrow x_i$ 
3 end
4 for  $l \leftarrow 1$  to  $L$  do
5   foreach  $v_i \in V$  do
6     Aggregate neighborhood messages:
7      $a_i^{(l)} \leftarrow \max_{j \in \mathcal{N}(i)} (A_{j,i} W^{(l)} h_j^{(l-1)})$ 
8     Update gate:
9      $z_i^{(l)} \leftarrow \sigma(W_z a_i^{(l)} + U_z h_i^{(l-1)})$ 
10    Reset gate:
11     $r_i^{(l)} \leftarrow \sigma(W_r a_i^{(l)} + U_r h_i^{(l-1)})$ 
12    Candidate hidden state:
13     $\tilde{h}_i^{(l)} \leftarrow \tanh(W_h a_i^{(l)} + U_h (r_i^{(l)} \odot h_i^{(l-1)}))$ 
14    Node state update:
15     $h_i^{(l)} \leftarrow (1 - z_i^{(l)}) \odot h_i^{(l-1)} + z_i^{(l)} \odot \tilde{h}_i^{(l)}$ 
16  end
17 end
18 Graph readout by global max pooling:
19  $h_d^G \leftarrow \max_{v_i \in V} h_i^{(L)}$ 
20 Failure probability prediction:
21  $\hat{p}_d(t) \leftarrow \sigma(\text{MLP}_\phi(h_d^G))$ 
22 Weighted binary cross-entropy loss:
23  $\mathcal{L}_d \leftarrow -[w_{\text{pos}} y_d(t) \log \hat{p}_d(t) + w_{\text{neg}} (1 - y_d(t)) \log (1 - \hat{p}_d(t))]$ 
24 return  $\hat{p}_d(t), \mathcal{L}_d$ 

```

progressively accumulated error states and bursty temporal escalation patterns [12].

Algorithm 1 shows the complete pipeline of the proposed CE2Graph model. Starting from the graph, the model iteratively propagates spatio-temporal messages through GG-NN layers, aggregates graph-level features, and outputs the probability of future failures.

1) *Neighborhood Message Aggregation*: At each message-passing layer, a node v_i aggregates information from its spatio-temporal neighborhood $\mathcal{N}(i)$. In real systems, DRAM logs often contain a large number of sporadic soft errors caused by transient disturbances, while catastrophic failures are more often associated with concentrated hard-fault patterns [11]. Thus, we adopt max-based neighborhood aggregation to emphasize the most severe local failure evidence rather than averaging it out.

Formally, at layer l , the aggregated message received by node v_i is defined as

$$a_i^{(l)} = \max_{j \in \mathcal{N}(i)} (A_{j,i} W^{(l)} h_j^{(l-1)}), \quad (11)$$

where $h_j^{(l-1)}$ denotes the hidden representation of node v_j from the previous layer, $W^{(l)}$ is a learnable weight matrix, and $A_{j,i}$ is the spatio-temporal edge weight defined in Section 3.3. Here, the max operator is applied element-wise over the transformed neighbor messages. Compared with sum or mean aggregation, this design is intended to preserve high-risk

localized activation patterns that may otherwise be diluted by a large number of weak or noisy neighbors.

2) *GRU-Based Node State Update*: After obtaining the aggregated neighborhood message $a_i^{(l)}$, the aggregated message is then used to update node states through a standard GRU-style update, following GG-NN. Lines 8 to 15 of Algorithm 1 illustrate the update process.

3) *Graph Readout and Failure Prediction*: After L layers of spatio-temporal message passing, each node representation has incorporated both spatio and temporal context from its local neighborhood. To predict the health status of the target DIMM d , we aggregate node embeddings into a graph-level representation using global max pooling, and the graph representation h_d^G is then fed into a multilayer perceptron (MLP) followed by a sigmoid activation to produce the probability of a future failure:

$$h_d^G = \max_{v_i \in V} h_i^{(L)}, \quad \hat{p}_d(t) = \sigma(\text{MLP}_\phi(h_d^G)). \quad (12)$$

4) *Loss Function for Imbalanced Learning*: In real-world environments, positive failure samples are extremely rare compared with normal CE logs, leading to severe class imbalance [17]. To mitigate this issue, we train the model using a weighted binary cross-entropy loss shown in Algorithm 1, where $y_k \in \{0, 1\}$ is the ground-truth label of sample k , $\hat{p}_k$ is the predicted probability, and w_{pos} and w_{neg} are the class weights for positive and negative samples, respectively.

Summary of the CE2Graph Workflow. The aforementioned steps constitute the complete CE2Graph pipeline. First, discrete CE logs are encoded into node representations capturing physical locations and DQ-Beat patterns (Section III-B). Next, spatial and temporal edges connect these nodes to form a unified spatio-temporal graph (Section III-C). Finally, the GG-NN aggregates spatio and temporal fault evidence from this graph, mapping it to a failure probability $\hat{p}_d(t)$ (Section III-D). This end-to-end workflow effectively bridges low-level hardware symptoms and high-level failure prediction.

IV. EXPERIMENT

A. Experimental Setup

1) *Datasets and Data Splitting*: We evaluate the proposed CE2Graph framework on a real-world DRAM failure dataset collected from a large-scale data center [19]. The dataset contains continuous CE logs, static configurations, and failure tickets that document actual DRAM-related failures. It covers 56,403 server files and 72,641 DIMM slots. The raw logs contain DIMM identity fields (e.g., server, CPU, channel, and DIMM identifiers), hierarchical physical-location fields (rank, device, bank group, bank, row, and column), timestamps, error types, decoded DQ-Beat error-bit information, and static platform configuration fields such as manufacturer, CPU model, DIMM part number, capacity, frequency and so on. CE2Graph mainly uses the DIMM identity, physical-location fields, timestamps, error types, and DQ-Beat information to construct node features and spatio-temporal edges.

We construct DIMM-level prediction samples using a sliding-window strategy. For each reference time t , CE events

TABLE I Performance comparison.

Method	Recall	Precision	F1-score
Tree-based models			
Random Forest	0.3473	0.2131	0.2641
XGBoost	0.2934	0.2973	0.2953
CatBoost	0.3054	0.2566	0.2789
LightGBM	0.5030	0.1742	0.2588
Deep learning models			
LSTM	0.4491	0.0806	0.1366
FT-Transformer	0.5210	0.0452	0.0832
Rule-based and handcrafted-feature predictors			
Cell fault predictor	0.69	0.04	0.08
Row fault predictor	0.43	0.06	0.11
Column fault predictor	0.26	0.08	0.13
Bank fault predictor	0.30	0.12	0.17
CE overflow predictor	0.15	0.38	0.21
CE storm predictor	0.61	0.08	0.13
Multi-DQ predictor	0.77	0.20	0.32
Multi-DQ-Beat predictor	0.79	0.14	0.24
CE2Graph (Ours)	0.3173	0.5940	0.4136

observed on the same DIMM within the 7-day observation window $[t - \Delta t_{\text{obs}}, t]$ are used to construct the input graph $G_d(t)$. Following the dataset protocol, we set $\Delta t_{\text{obs}} = 7$ days and $\Delta t_{\text{pred}} = 30$ days; thus, a sample is labeled positive if the DIMM experiences a DRAM-related failure within the next 30 days.

To simulate realistic online deployment, we do not adopt random cross-validation. Instead, we use a *chronological splitting* strategy that follows the temporal order of the logs [6], [9]. The dataset is split into a training set, a validation set, and a test set. The split dataset contains 127,900 samples, including 20,647 positives and 107,253 negatives, with a positive ratio of 16.14%. The training, validation, and test sets contain 86,904, 20,465, and 20,531 samples.

2) *Baselines*: CE2Graph and all learning-based baselines use the same CE-log information, and CE2Graph differs by organizing CE events as a spatio-temporal graph.

a) *Tree-based models*: We include Random Forest (RF), XGBoost, CatBoost and Lightgbm as classical baselines [20], [21], [22], [23]. These methods operate on flattened tabular features extracted from the observation window and represent the dominant paradigm in prior studies.

b) *Deep learning models*: We include Long Short-Term Memory (LSTM) and FT-Transformer as representative deep learning baselines [24], [25]. LSTM is capable of modeling temporal dependency in CE logs, but it does not explicitly encode the physical topology of DRAM components. FT-Transformer is a strong deep model for tabular data.

c) *Rule-based and handcrafted-feature predictors*: We also compare against conventional rule-based and handcrafted-feature methods, such as Cell fault predictor, Row/Column fault predictor, Bank fault predictor, CE overflow predictor, CE storm predictor, Multi-DQ predictor, and Multi-DQ-Beat predictor [11].

TABLE II Ablation study of CE2Graph.

Variant	Recall	Precision	F1-score
Only Temporal	0.2365	0.7074	0.3545
Temporal + Bank	0.3078	0.6102	0.4092
Temporal + Row	0.2797	0.6507	0.3912
Temporal + Column	0.2578	0.7364	0.3819
Temporal + Cell	0.2722	0.6525	0.3841
Temporal + Bank + Row	0.2457	0.6332	0.3540
Only Spatio	0.2584	0.7640	0.3862
w/o DQ-Beat	0.2656	0.6716	0.3807
CE2Graph	0.3173	0.5940	0.4136

3) *Evaluation Metrics*: We evaluate model performance using *Precision*, *Recall*, and *F1-score*, which are standard metrics for binary failure prediction.

B. Overall Performance Comparison

Table I presents the performance comparison between CE2Graph and three groups of baselines. Overall, CE2Graph achieves the best *F1-score* of 0.4136 among all evaluated methods, with a precision of 0.5940 and a recall of 0.3173. Compared with the strongest tree-based baseline, **XGBoost** (*F1-score* = 0.2953), CE2Graph improves the *F1-score* by **40.1%**. It also outperforms the strongest rule-based baseline, **Multi-DQ predictor** (*F1-score* = 0.32), by **29.3%**. These results show that explicitly modeling the spatio-temporal structure of CE propagation provides a better precision–recall balance than flattened handcrafted features.

The deep learning baselines, including **LSTM** and **FT-Transformer**, obtain relatively high recall but much lower precision, leading to poor *F1-scores*. Rule-based predictors, such as **Multi-DQ** and **Multi-DQ-Beat**, also achieve high recall but suffer from low precision and therefore tend to generate many false positives. In contrast, CE2Graph combines DQ-Beat patterns, physical-location information, and graph-based Spatio-Temporal message passing, resulting in the best overall *F1-score* under the same evaluation setting.

C. Ablation Study

To assess the contribution of each CE2Graph component, we conduct ablation studies by removing or simplifying spatial edges, temporal edges, and DQ-Beat features, while keeping the same raw CE-log fields and training protocol. As shown in Table II, the full CE2Graph achieves the best *F1-score*. Removing spatial relations and keeping only temporal edges reduces the *F1-score* from 0.4136 to 0.3545, while removing temporal edges decreases it to 0.3862. Removing DQ-Beat features also lowers the *F1-score* to 0.3807. These results indicate that topology-aware spatial relations, temporal adjacency, and bit-level DQ-Beat patterns all contribute to the prediction performance.

V. CONCLUSION

In this paper, we proposed **CE2Graph**, a graph-based spatio-temporal framework for DRAM failure prediction. By transforming irregular CE logs into structured graphs and jointly modeling error-bit patterns, physical topology, and temporal features, CE2Graph provides an effective way to

capture failure patterns beyond conventional flattened-feature methods. Experimental results, including comparisons with representative baselines and analyses of key design factors, show that CE2Graph outperforms compared methods and achieves the best balance between precision and recall. In future work, we will evaluate the robustness of CE2Graph across different hardware platforms and explore its integration with online mitigation strategies.

VI. ACKNOWLEDGEMENT

The authors would like to thank Limin Zeng (HIAS, UCAS) for his helpful discussions.

REFERENCES

- [1] Z. Cheng *et al.*, “An in-depth correlative study between DRAM errors and server failures in production data centers,” in *Proc. SRDS*, 2022.
- [2] B. Schroeder and G. A. Gibson, “A large-scale study of failures in high-performance computing systems,” *IEEE Trans. Dependable Secure Comput.*, 2009.
- [3] J. Jung and M. Erez, “Predicting future-system reliability with a component-level DRAM fault model,” in *Proc. MICRO*, 2023.
- [4] C. Li *et al.*, “From correctable memory errors to uncorrectable memory errors: What error bits tell,” in *Proc. SC*, 2022.
- [5] B. Schroeder *et al.*, “DRAM errors in the wild: a large-scale field study,” *Commun. ACM*, 2011.
- [6] P. Zhang *et al.*, “Predicting DRAM-caused node unavailability in hyper-scale clouds,” in *Proc. DSN*, 2022.
- [7] X. Du and C. Li, “Memory failure prediction using online learning,” in *Proc. MEMSYS*, 2018.
- [8] Z. Wu *et al.*, “DRAM failure prediction in AIOps: empirical evaluation, challenges and opportunities,” *arXiv:2104.15052*, 2021.
- [9] X. Peng *et al.*, “EXPERT: exploiting DRAM error types to improve the effective forecasting coverage in the field,” in *Proc. DSN-S*, 2023.
- [10] E. Baseman *et al.*, “Physics-informed machine learning for DRAM error modeling,” in *Proc. DFT*, 2018.
- [11] Q. Yu *et al.*, “HiMFP: hierarchical intelligent memory failure prediction for cloud service reliability,” in *Proc. DSN*, 2023.
- [12] A. R. Gorle *et al.*, “G-MAP: a graph neural network-based framework for memory access prediction,” in *Proc. HPEC*, 2023.
- [13] A. A. Hwang *et al.*, “Cosmic rays don’t strike twice: understanding the nature of DRAM errors and the implications for system design,” *SIGPLAN Not.*, 2012.
- [14] V. Sridharan and D. Liberty, “A study of DRAM failures in the field,” in *Proc. SC*, 2012.
- [15] M. V. Beigi *et al.*, “A systematic study of DDR4 DRAM faults in the field,” in *Proc. HPCA*, 2023.
- [16] I. Giurgiu *et al.*, “Predicting DRAM reliability in the field with machine learning,” in *Proc. Middleware Industrial Track*, 2017.
- [17] X. Sun *et al.*, “System-level hardware failure prediction using deep learning,” in *Proc. DAC*, 2019.
- [18] H. Xie *et al.*, “M2-MFP: a multi-scale and multi-level memory failure prediction framework for reliable cloud infrastructure,” in *Proc. KDD*, 2025.
- [19] “Large-scale DRAM logs dataset,” <https://www.codabench.org/competitions/3586/>, 2025.
- [20] L. Breiman, “Random forests,” *Mach. Learn.*, 2001.
- [21] T. Chen and C. Guestrin, “XGBoost: a scalable tree boosting system,” in *Proc. KDD*, 2016.
- [22] L. Prokhorenkova *et al.*, “CatBoost: unbiased boosting with categorical features,” in *Proc. NeurIPS*, 2018.
- [23] G. Ke *et al.*, “LightGBM: a highly efficient gradient boosting decision tree,” in *Proc. NeurIPS*, 2017.
- [24] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, 1997.
- [25] Y. Gorishniy *et al.*, “Revisiting deep learning models for tabular data,” in *Proc. NeurIPS*, 2021.