STIM: Predicting Memory Uncorrectable Errors with Spatio-Temporal Transformer

Zhexiong Liu, t-zhexliu@microsoft.com Cris Benge, crbenge@microsoft.com Yash Dagli, ydagli@microsoft.com
Siduo Jiang, stonejiang@microsoft.com

Abstract

Memory Uncorrectable Errors (UEs) have been identified as a leading cause of server crashes in Microsoft Azure datacenters. To mitigate this, leveraging machine learning methods to predict UEs before undertaking preventive maintenance has become an effective measurement to decrease server downtime in large-scale clusters. However, the prediction of UEs presents several challenges: (1) anomaly UEs are exceedingly rare in memory events. This rarity renders machine learning methods highly vulnerable to imbalanced data, leading to high false positives and low recall rates. (2) memory data is inherently noisy and heterogeneous because memory hardware comes from a range of manufacturers (e.g., Samsung, SK-Hynix) and is deployed in diverse operational environments. This necessitates the need for robust machine learning models. (3) Predicting UEs demands the handling of intricate spatial and temporal variability in the memory framework, given the hardware degradation over its lifecycle and the iterative introduction of new software and hardware configurations. To address these issues, we proposed a Spatial-temporal Transformer in Memory (STIM) model that learns spatial and temporal features with self-attention mechanisms across memory cells. Comparative evaluations on our developed Azure Memory Error Dataset (AMED) reveal that our proposed STIM model outperforms existing machine learning methods and strong baselines, which marks the significant application of Language Models (LMs) to address complex UE prediction tasks.

Keywords: Uncorrectable Errors, Memory Failures, Spatio-Temperal, Transformer, Language Models.

1. Introduction

Dynamic Random Access Memory (DRAM) has served as the primary memory for data storage and retrieval in modern computer systems. A typical DRAM chip comprises thousands of capacitors, each either charged or discharged to denote binary bits [1]. However, these stored electric charges are vulnerable to alteration due to various internal and external factors, e.g., unstable data transmission along bitlines, unexpected charge losses before bit restoration, significant temperature fluctuations in the operational environment, etc. Such vulnerabilities often lead to inconsistencies between the bits (data) read from DRAM and the bits (data) originally stored. To mitigate these inconsistencies, Error Correction Codes (ECCs) are employed to identify and correct erroneous bits. One widely used Chipkill ECC [2] is tailored to rectify incorrect bits from a single DRAM chip during memory access at the cache line granularity. These rectifiable errors are termed Correctable Errors (CEs). Nonetheless, ECCs can be ineffective if multiple erroneous bits occur simultaneously, exceeding the ECC's error-correction capacity, or if certain error patterns escape the ECC's detection capabilities. In such cases, Uncorrectable Errors (UEs) occur, potentially leading to critical system disruptions and potential security issues [3, 4, 5]. Figure 1 shows that UEs have been identified as the leading contributors to severe cluster outrages in Microsoft Azure datacenters.

In addressing Uncorrectable Errors (UEs) proactively, employing a method that can predictively diagnose memory errors becomes crucial. This allows for the timely replacement of memory hardware modules, such as the Dual Inline Memory Module (DIMM), preventing potential catastrophic system failures [6]. In literature, traditional approaches mostly learn from historical statistics of CEs to predict UEs. For instance, when a DIMM records an unusually elevated amount of CEs over a predefined period, a proactive replacement is initiated. This practice is based on the assumption that DIMMs with high CE rates are prone to initiating UEs [7, 8].

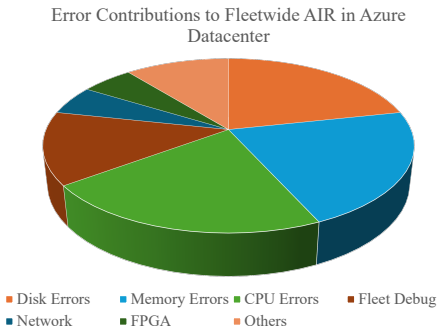

Figure 1. Recent error contributions to Fleetwide AIRs that cause cluster severe outrages in Azure datacenter.

However, such methods often yield false positives, making them less than ideal for large-scale datacenters. To address this issue, [9, 10] leverage micro Correctable Error (CE) information (e.g., channel, rank, bank, row, column, etc.) to develop advanced machine learning methods. Interestingly, these works found that predictors based on column fault identification exhibit superior efficacy in predicting future UEs, which demonstrates the importance of learning spatial information within memory modules. Yet, these works often failed to take into account essential bit features that have strong connections to UE occurrences. To bridge the gap, [11] introduces bit CE features (e.g., DQ-beat error bits) and a set of hierarchical predictors to reveal latent UE patterns, which exhibits the potential to learn spatial information through different levels of features. Despite these advancements, the current methods mainly focus on extracting useful features rather than developing advanced machine learning models to capture spatio-temporal variability. While feature-driven methods excel at leveraging effective predictors, they still struggle to learn interactions across spatio-temporal erroneous bits (e.g., dynamic bit alternation in different timeframes). In contrast, our study presents a Spatio-temporal Transformer in Memory (STiM) model that aims to capture the variability of these essential spatio-temporal dynamics with an end-to-end Transformer-based model [12]. This initiative marks the first endeavor to employ cutting-edge language models for effectively predicting UEs.

While adapting machine learning models to UE prediction tasks has gained unprecedented interest, it also brings major challenges. For instance, UEs are exceedingly rare in memory events compared to CEs, which renders machine learning methods highly vulnerable to imbalanced data, leading to high false positives and low recall rates. [10] leverages an ensemble learning approach to eliminate the imbalance issue, which increases model complexity and loses model interpretability. It also shows that predictor-based methods perform relatively well on certain datasets and inefficiently on others. Based on this, we developed our first research question (**RQ1**): Can we train an end-to-end model that performs constantly well on multiple datasets? Predicting UEs demands the handling of intricate spatial and temporal variabilities in the memory framework, given the hardware degradation over its lifecycle and the iterative introduction of new software configurations. Prior feature-driven methods [9, 10, 11] have shown potential in learning spatial features in UE predictions but are not sufficient to uncover the complex variability in memory errors. Therefore, we investigate our second research question (**RQ2**): Can we train a model capable of handling spatio-temporal variability in memory errors? Memory data (in both micro and bit) are inherently noisy and heterogeneously distributed because DIMMs come from a range of manufacturers that implement different manufacturing protocols. Also, the DIMMs are deployed in diverse operational environments with various external interventions. This necessitates our third research question (**RQ3**): How to wisely use diverse datasets to train a better model?

To address these questions, we developed the first Azure Memory Error Dataset (AMED) with four variants to examine the effectiveness of various UE prediction methods. We conduct a pilot study on the state-of-the-art machine learning model and extensively evaluate our proposed strong baselines on AMED. We demonstrate the superiority of our proposed STiM model in solving UE prediction tasks. Our contribution is threefold:

- We develop an automated pipeline to collect the first large-scale real-word Azure memory error dataset that contains micro and bit features.
- We proposed the first highly effective Transformer-based language model for UE predictions that achieves excellent performance on AMED.
- We address the spatio-temporal variability in memory using an end-to-end Transformer model, avoiding the complexities of model assembly. This ensures model robustness when handling diverse memory errors.

2. Related Work

2.1. Memory Error Prediction

Recent research in memory error predictions has emphasized the use of CEs to predict UEs. [13] introduces a machine learning model that uses CEs from event and sensor logs to predict future UEs. This work investigates a set of techniques for handling missing data and UE anomalies but still yields relatively high false positives. Furthermore, [9] offers an online learning strategy that leverages micro-level information (e.g., cells, rows, and columns on each DIMM) to train machine learning models on large-scale field data

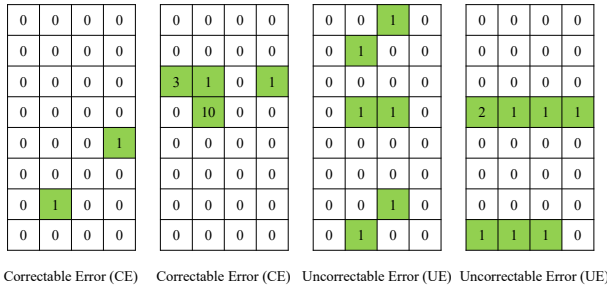

Figure 2. Bitmap examples of CEs and UEs, showing strong spatial (bit) variability across 2D-grid cells. The zero denotes no error and the positive number denotes the number of erroneous bits in a cell in a certain period.

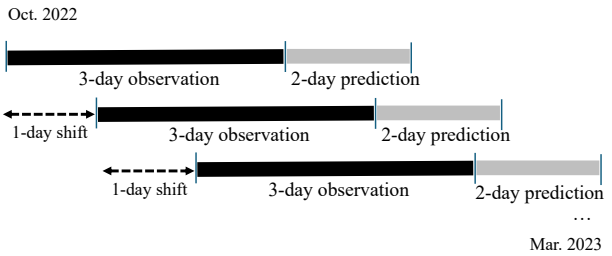

Figure 3. Moving-window-based dataset construction. The observation window collects the micro-level and bit-level features from the data point that has no UEs in the past 3 days. The prediction window collects the memory error labels (CEs/UEs) for the DIMMs presented in the observation window

from more than 30,000 contemporary servers. This method heavily relies on UE predictors (e.g., the daily number of cell errors) thus has low performance when observed data is highly noisy. Later, [10] improves their prior method by using a weighted ensemble approach that gives certain weights to each weak classifier/predictor, which increases the model complexity but misses the opportunity to analyze the important spatio-temporal correlations in erroneous bits. In addition, [14] presents a cost-aware random forest model for UE prediction that uses features from DIMM, sockets, and nodes. This work argues that precision and recall are insufficient to evaluate UE prediction and proposes a cost-benefit analysis to prevent memory error predictors less biased which mitigates the model learning on data from retired DIMMs. This confirms the challenge that memory data has strong temporal variability due to hardware degradation, updates, and iterations. More recently, [1] focuses on erroneous bits and DIMM part numbers to study the relationship between CEs and UEs, which predicts UEs based on different hardware manufacturers and part numbers. [11] further investigates the latent correlations of erroneous bits during DRAM read/write in Data Bus (DQ) and Beat. Additionally, a comprehensive hierarchical framework has proved to be useful for adapting different levels of memory recovery techniques, ranging from the bit-level to the micro-level, to the system-level. In contrast to our study, we further leverage these useful predictors to build a Transformer-based language model to effectively learn the hidden variability across different spatial (bits distribution on 2D grids) and temporal (bits and predictors in different timelines) features. Our study is the first work in the field, aiming to solve the UE prediction task with language models, which opens a new avenue for UE predictions, or even broader hardware error defections.

2.2. Deep Learning Methods

Recently, deep learning methods have gained unprecedented prevalence in Natural Language Processing (NLP) [15, 16]. For example, Long Short-Term Memory (LSTM) models [17] have been trained to solve multiple downstream NLP tasks [18]. Although LSTMs have bidirectional versions that look forward and backward in the input sequences, it is still insufficient to take advantage of both contexts simultaneously, making its ability to understand sequence content limited. To improve this, [19] proposes the Transformer model which is the first transduction model relying entirely on self-attention to compute representations of inputs and outputs without using sequence-aligned LSTMs or convolutions. Previous work leverages Transformer-based methods to conduct classification in computer vision and NLP [20, 21], and achieved excellent results; however, there is no prior work, to the best of our knowledge, that employs Transformer-based models to hardware-related anomaly detection tasks (e.g., UE predictions) because hardware (memory) data is often domain-specific, complex, and dataset scarcity. Also, the extremely unbalanced CEs and UEs distributions in the UE prediction task, make it challenging to train large-scale deep learning models. To bridge these gaps, we developed the first Azure Memory Error Dataset (AMED) with four variants for UE predictions. Nevertheless, directly applying text-based NLP models to solve UE prediction tasks may not yield good performance as the memory data are all numerical and have no interpretable dependency and relations. Instead, memory bits exhibit strong spatio-temporal variability across different hardware (DIMMs) where the bits (0 or 1) on 2D-grid cells are frequently altered to exhibit different data. Therefore, we initiate the first study, inspired by [12], to use memory bits to train a Spatio-temporal Transformer in Memory (STIM) model for UE predictions.

3. Dataset

Presently, there is no large-scale dataset for UE predictions publically available for two reasons. First, memory data collection directly relies on cloud server providers who own large-scale computing clusters but the others such as aca-

Year-month	UEs	CEs	Total
2022-10	226	12,958	13,184
2022-11	152	13,271	13,423
2022-12	93	14,040	14,133
2023-01	120	14,110	14,230
2023-02	104	12,556	12,660
2023-03	103	13,467	13,570

Table 1. UE and CE distributions in each month

Dataset	Training	Develop	Test
Dataset-A	10/1/22 - 2/15/23	2/16/23 - 2/28/23	3/1/23 - 3/31/23
Dataset-B	11/1/22 - 2/15/23	2/16/23 - 2/28/23	3/1/23 - 3/31/23
Dataset-C	12/1/22 - 2/15/23	2/16/23 - 2/28/23	3/1/23 - 3/31/23
Dataset-D	12/1/22 - 2/28/23	12/1/22 - 2/28/23	3/1/23 - 3/31/23

Table 2. Dataset-A, Dataset-B, Dataset-C, and Dataset-D have different data splits in training and development sets.

demic researchers have no access to data sources. Second, the memory data contains sensitive privacy (e.g., server running time) that has confidential business values and cannot be released to the public. As a result, we curated the first Azure Memory Error Dataset (AMED) pulled from Windows Hardware Error Architecture (WHEA)[1] and Azure Virtual Machine (VM)[2] logs between October 2022 and March 2023. The raw data contains a set of micro-level (e.g., error-count, error-type, failed-pages, devices, banks, locations, running time, etc.) and bit-level (8x4 bitmaps decoded from fail masks obtained by ECC) memory features. Figure 2 shows examples of different bitmap features that have various error-bit layouts. These bits exhibit strong spatial variability across cells on an 2D grid. Furthermore, we simply perform a data aggregate that sums up all the data points (bitmaps and micro-level features) in a one-hour window, given continuously recorded event logs on every DIMM. This aggregation can smooth out anomalies or fluctuations in the raw data. In practice, we collect aggregated data for hundreds of thousands of DIMMs deployed on different clusters for consecutive 6 months. Statistically, there are 798 UEs and 80,402 CEs where UEs take 0.98% and CEs take 99.02% out of total 81,200 memory errors. Table 1 shows the distributions of UEs and CEs in each month in the dataset.

Following [11], we create an observation window and a prediction window (shown in Figure 3) to construct AMED. In particular, we take the past 3 days ($t-3$) as an observation window where we filter out the data points that have no UEs, and then we collect their micro-level and bit-level features. We take the next 2 days ($t+2$) as a prediction window where we collect the memory error labels (CEs/UEs) for corresponding data points presented in the observation window. To this end, we have features and labels for each DIMM at timestamp t. Next, we shift 1 day forward to construct another feature-label pair for each DIMM until the end of the collecting period. In this setting, the model trained on the observation window ($t-3$) at timestamp t can literally predict UEs in the coming two days ($t+2$). In addition, we develop four variants of AMED by differently splitting the dataset into training, developing, and test sets. As shown in Table 2, Dataset-A, Dataset-B, and Dataset-C have different time windows in training sets but the same in development and test sets. This design aims to help train models with the data in different time windows which amplifies temporal variability in AMED. Dataset-D which splits the training and development sets in the same time window is another variant used to (compare to dataset-C) investigate how to optimize training and develop sets splits in order to better fine-tune model parameters.

[1] https://learn.microsoft.com/en-us/windows-hardware/drivers/whea/

[2] https://azure.microsoft.com/en-us/products/virtual-machines

4. Methods

4.1. Preliminary

An overview of the framework is shown in Figure 4. The framework contains a data collection, bitmap and micro features, a Transformer encoder, and an MLP classifier. In the data collection phase, WHEA logs are extracted, transformed, and loaded into hourly aggregated data that are combined with VM logs. Given a very sparse $p \times q$ bitmap S^i at (hourly) timestamp i, where p is the row number and q is the column number, we can flatten $S^{(i)}$ into a 1-dimensional vector $S'^{(i)}$, where $S'^{(i)} \in \mathcal{R}^{pq \times 1}$. In order to leverage micro features $G^{(i)} \in \mathcal{R}^{r \times 1}$ at timestamp i, where r is the dimension of micro features, we concatenate $S'^{(i)}$ and $G^{(i)}$ into a memory feature vector $X^{(i)} = [S'^{(i)}; G^{(i)}]$, where $X^{(i)} \in \mathcal{R}^{m \times 1}$, $m = pq + r$. In an observation window, we take a sequence of the memory features $\mathbf{X} = [X^{(1)}, X^{(2)}, ..., X^{(t)}]$, where t is the length of the feature sequence (e.g., size of the t-hour window), $\mathbf{X} \in \mathcal{R}^{m \times t}$. To this end, the inputs $\mathbf{X}$ contain m features in t-size observation window, where $t = 72$ for $t = 72$ hours (3 days) and $m = 39$ features ($p = 8, q = 4, r = 7$). Therefore, the UE prediction task is formulated as using $\mathbf{X}$ to predict binary labels $\mathbf{y}$ for UE predictions.

4.2. Spatio-Temporal Transformer in Memory

Given that the Transformer [19] uses a constant latent vector with size d throughout all of its layers, we map $\mathbf{X}$ into latent memory features $\tilde{\mathbf{X}} \in \mathcal{R}^{d \times t}$ d dimensions with a trainable linear projection (Eq. 1). The output of the linear projection

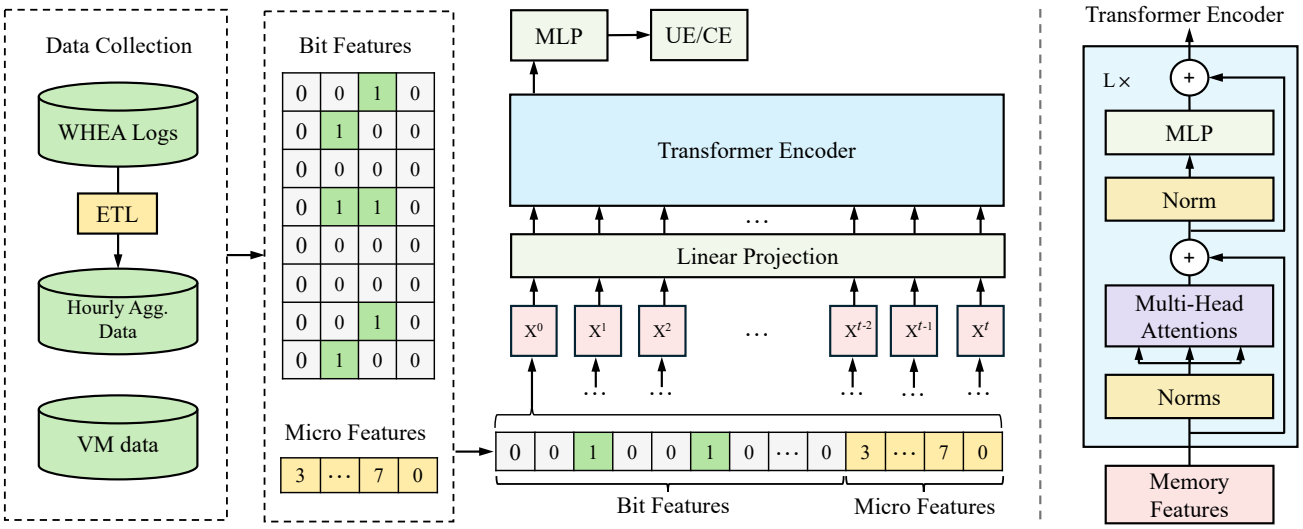

Figure 4. We collect data from WHEA and VM logs, including bit and micro-level features. The 8x4 bitmap is flattened to a 32-bit vector that is concatenated with 7 micro-level features. We feed a sequence of feature vectors (39 dimensions) to a standard Transformer encoder. In order to perform classification, we feed the first feature representation on the last layer of the Transformer encoder to a linear classifier.

$\tilde{\mathbf{X}}$ is fed to the Transformer encoder.

$$\tilde{\mathbf{X}} = \mathbf{X}W. \tag{1}$$

We used a similar Transformer encoder architecture to [12]. In the framework, the encoder is composed of a stack of L identical layers, where each layer has two major blocks.

The first one is a self-attention block (SAB) in Eq. 4.

$$[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \tilde{\mathbf{X}} U_{\mathbf{qkv}}, \tag{2}$$

$$A = \text{softmax}\left(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{D}}\right), \tag{3}$$

$$\text{SAB}(\tilde{\mathbf{X}}) = A\mathbf{v}, \tag{4}$$

where the $U_{\mathbf{qkv}} \in \mathcal{R}^{3d\times t}$ projects each memory feature representation $\tilde{\mathbf{X}}$ to query vector $\mathbf{q} \in \mathcal{R}^{d\times t}$, key vector $\mathbf{k} \in \mathcal{R}^{d\times t}$, and value vector $\mathbf{v} \in \mathcal{R}^{d\times t}$. The dot products performed between queries ($\mathbf{q}$) and keys ($\mathbf{k}$) are scaled by a scalar $\sqrt{d}$, passing to a Softmax function [22] to obtain attention scores A. The attention scores are used to multiply values ($\mathbf{v}$) so the memory feature $\tilde{\mathbf{X}}$ is weighed.

The second block transforms the weighted $\tilde{\mathbf{X}}$ using a Multi-Layer Perceptron (MLP), where the GELU [23] activation function and (0.5) dropout was used. In addition, a layer normalization is applied before the SAB and MLP block, and residual connections after the two blocks [24, 25]. The overall Transformer encoder is formulated as

$$\tilde{\mathbf{X}}'_{l-1} = \text{SAB}\left(\text{Norms}\left(\tilde{\mathbf{X}}_{l-1}\right)\right) + \tilde{\mathbf{X}}_{l-1}, \tag{5}$$

$$\tilde{\mathbf{X}}_{l} = \text{MLP}\left(\text{Norms}\left(\tilde{\mathbf{X}}'_{l-1}\right)\right) + \tilde{\mathbf{X}}'_{l-1}, \tag{6}$$

where $l = 0, 1, 2, ..., L$ is the layer number. Each encoder layer l uses the output of the encoder from the previous layer $\tilde{\mathbf{X}}_{l-1}$. Following [26], the output of the last layer (layer L) of the Transformer encoder is a sequence of memory feature representations in a n-size observation window. We take the first representation $\tilde{\mathbf{X}}^i_L$ at (hourly) timestamp $i = 0$ as the is taken as the input of the MLP classifier. The MLP classifier used to predict UEs and CEs is implemented with a linear layer and a Sigmoid function [27].

$$\hat{\mathbf{y}} = \text{MLP}\left(\tilde{\mathbf{X}}^0_L\right), \tag{7}$$

where $\hat{\mathbf{y}}$ is the prediction. We use binary cross-validation loss to train our model, which is described as

$$Loss(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y}\log(\hat{\mathbf{y}}) - (1-\mathbf{y})\log(1-\hat{\mathbf{y}}) \tag{8}$$

where $\mathbf{y}$ is the ground truth label for UE predictions.

4.3. Proposed Deep Learning Baselines

In the literature, the state-of-the-art UE prediction methods mainly focus on extracting useful predictors and applying boosting-based classifiers with the extracted features [1, 9, 11, 28]. However, these traditional machine learning methods often require complex feature engineering and are also inefficient in handling intricate data complexity. In comparison to traditional machine learning methods, prior works have demonstrated that deep learning methods have great superiority in solving image classification [29], text classification [30], network intrusion detection [31],

Dataset	Models	Precision	Recall	F1-score	AUC	AP	MCC
Dataset-A	BiGRU	0.7612	0.4951	0.6000	0.9349	0.5849	0.6116
	BiGRU-CNN	0.6988	0.5631	0.6237	0.9153	0.6325	0.6248
	BiGRU-ATT	0.6897	**0.5825**	0.6316	0.9483	0.6215	0.6313
	BiGRU-CNN-ATT	0.7432	0.5340	0.6215	0.9302	0.5876	0.6276
	STIM (Ours)	**0.7917**	0.5534	**0.6514**	**0.9425**	**0.6504**	**0.6598**
Dataset-B	BiGRU	0.6105	0.5631	0.5859	0.9089	0.5492	0.5833
	BiGRU-CNN	0.7324	0.5049	0.5977	0.8628	0.5337	0.6057
	BiGRU-ATT	0.6022	0.5437	0.5714	0.9219	0.5579	0.5691
	BiGRU-CNN-ATT	0.7121	0.4563	0.5562	0.8841	0.5235	0.5675
	STIM (Ours)	**0.8133**	**0.5922**	**0.6854**	**0.9543**	**0.6582**	**0.6921**
Dataset-C	BiGRU	0.7361	0.5146	0.6057	0.8413	0.5335	0.6131
	BiGRU-CNN	0.6197	0.4272	0.5057	0.8693	0.5022	0.5115
	BiGRU-ATT	0.7910	0.5146	0.6235	0.8877	0.5545	0.6359
	BiGRU-CNN-ATT	0.6962	0.5340	0.6044	0.9159	0.5651	0.6071
	STIM (Ours)	**0.8750**	**0.5437**	**0.6707**	**0.9257**	**0.6340**	**0.6880**
Dataset-D	BiGRU	**0.9348**	0.4175	0.5772	0.9193	0.6352	0.6230
	BiGRU-CNN	0.8596	0.4757	0.6125	0.8850	0.5949	0.6376
	BiGRU-ATT	0.7714	0.5243	0.6243	0.9126	0.5800	0.6337
	BiGRU-CNN-ATT	0.8182	0.5243	0.6391	**0.9385**	0.6542	0.6529
	STIM (Ours)	0.9077	**0.5728**	**0.7024**	0.9361	**0.6601**	**0.7195**

Table 3. The performance of the proposed strong baselines and STIM models. Note that the precision, recall, and F1-score are the scores for the (minority) positive labels. AUC, AP, and MCC denote ROCAUC, Average Precision (AP), and Matthews Correlation Coefficient, respectively. Our proposed STIM almost outperforms all the baselines across different metrics and different datasets.

cybersecurity [32], hard disk driver failure prediction [33], etc. This is because deep learning methods have a stronger capability of handling complex sequential (temporal) and hierarchical (spatial) data, which are also two key characteristics of memory data. In our pilot study, we employed XGBoost [34] to train a classifier for the UE prediction on AMED (Dataset-A), and obtained 0.19 precision, 0.26 recall, and 0.22 F1 score, respectively, which are very poor. Therefore, our main interest in this work remains in deep learning methods. Because there is no prior work, to the best of our knowledge, utilizing deep learning methods in UE predictions, we propose four strong baselines as follows.

BiGRU: The conventional Recurrent Neural Networks (RNNs) are specifically designed to handle sequence data, making them suitable for tasks like time series forecasting and NLP tasks [17]. The Gated Recurrent Unit [35] is an advanced version of the RNN which is lighter and capable of handling longer sequences. In our implementation, we use Bidirectional GRU (BiGRU) [36] that can capture past (left-side) and future (right-side) information simultaneously for a given point in a sequence. This is particularly useful for tasks where context from both sides is crucial for understanding the current data point. Given the input gate i, forget gate f, and output gate o that determine whether the incoming data should be retained or forgotten, the hidden state $\vec{h}^{(t)}$ at timestamp t for the forward direction is described as:

$$i^{(t)} = \sigma\left(W_{Xi}X^{(t)} + W_{hi}h^{(t-1)} + W_{ci}c^{(t-1)} + b_i\right) \quad (9)$$

$$f^{(t)} = \sigma\left(W_{Xf}X^{(t)} + W_{hf}h^{(t-1)} + W_{cf}c^{(t-1)} + b_f\right) \quad (10)$$

$$c^{(t)} = f_t c^{(t-1)} + i_t \tanh\left(W_{Xc}X^{(t)} + W_{hc}h^{(t-1)} + b_c\right) \quad (11)$$

$$o^{(t)} = \sigma\left(W_{Xo}X^{(t)} + W_{ho}h^{(t-1)} + W_{co}c_t + b_o\right) \quad (12)$$

$$\vec{h}^{(t)} = o^{(t)} \tanh\left(c_t\right) \quad (13)$$

where $X^{(i}$ denotes the memory feature at timestamp i, σ denotes the Sigmoid function, c is the cell state, W is the weight matrix, b is the bias term. At the same time, the hidden state $\overleftarrow{h}^{(i)}$ for the backward direction can also be obtained accordingly. The forward and backward hidden states are then combined $h = \vec{h}\|\overleftarrow{h}$ for every timestamp. The hidden state h is later fed to a linear classifier for UE predictions.

BiGRU-CNN: Although BiGRU is able to learn sequential (temporal) information for UE predictions, the spatial variability (e.g., erroneous bits on the bitmap in Figure 2) is not handled. Thus we use a simple Convolutional Neural Network (CNN) to learn the spatial relations on the bitmap. Inspired by [1] that used indicators at the level of a row and a column to learn the spatial information, we design a 1D (p-dimension) row-kernel and 1D (q-dimension) column-kernel to perform row and convolutions, respectively. The CNN-extracted row and column (convolutional) features are flattened into a 1D vector $S_c^{(i)} \in \mathcal{R}^{k\times 1}$ ($k = 32$). Furthermore, we concatenate the spatial vector extracted from CNN, the original flattened bitmap, and micro features into a new feature vector $X_c^{(i)} = \left[S_c^{(i)}; S'^{(i)}; G^{(i)}\right] \in$

$\mathcal{R}^{(k+m)\times 1}$ at timestamp i in the given observation window. To this end, the spatial features are learned intensively. Afterward, a sequence of spatial-enhanced features $\mathbf{X}_c = \left[\mathbf{X}_c^{(1)}, \mathbf{X}_c^{(2)}, \ldots, \mathbf{X}_c^{(t)}\right] \in \mathcal{R}^{(k+m)\times t}$ are fed to a BiGRU model and the output of the BiGRU is fed to a linear classifier for UE predictions.

BiGRU-ATT: BiGRU can learn the interactions between sequences but cannot learn their importance. Thus the BiGRU may learn unnecessary information but ignore useful information in the input sequence. To address this issue, [37] proposed an Attention-based Bidirectional Long short-term memory (BiLSTM-ATT) that applies an attention layer to the top of the BiLSTM model. In our implementation, we added an attention layer to the BiGRU (BiGRU-ATT) model. In particular, an attention block that contains several dense layers and a Softmax layer is applied to learn the attention α for each hidden state $h^{(t)}$ at time step t

$$\alpha_t = \text{Softmax}\left(W_{att} \cdot h_t + b_{att}\right) \tag{14}$$

where W_{att} and b_{att} are attention weight and bias, respectively. So, the aggregated attention α and hidden state h over all the time steps are fed to a linear classifier for UE predictions.

BiGRU-CNN-ATT: This baseline combines the BiGRU and CNN convolutions and attention layers into one end-to-end model. In particular, the spatially enhanced features extracted from a row-kernel and column-kernel in a CNN model are flattened into $S_c^{(i)}$, and then concatenated with the original flattened bitmap and micro features $X_c^{(i)}$ at timestamp i. Afterward, a sequence of spatial-enhanced features $\mathbf{X}_c$ is fed to a BiGRU model, followed by a linear classifier for UE predictions.

5. Experiments and Analysis

5.1. Implementation Details

In the data preprocessing, we develop a set of rules to clean the WHEA and VM log data, such as removing missing values, etc. We conducted extensive experiments on the sizes of the observation window and prediction window. We found that the UEs have the most correlation to the CEs in the past 3 days and setting the prediction window to 2 days meets our needs. We implement all the models with Pytorch[3]. We train the BiGRU, CNN-aided modules, Transformer that uses one self-attention block, and a binary linear classifier from scratch on an Nvidia V100 GPU. All the models use the same cross-entropy loss, the 64 batch size, and the 0.0003 learning rate with 0.05 decay every 10 out of a total of 60 epochs. We tune the model hyper-parameters on the development set and report the test set performance

[3] https://pytorch.org/

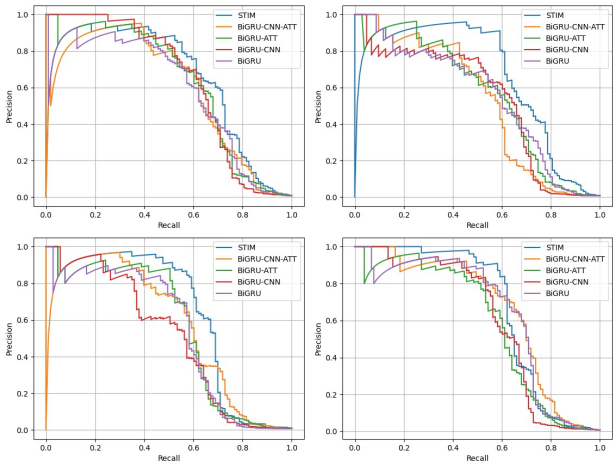

Figure 5. The precision-recall curve for each model on each dataset. The upper left is Dataset-A, the upper right is Dataset-B, the lower-left is Dataset-C, and the lower-right is Dataset-D.

using the best model. We report the (minority) positive-label Precision, Recall, F1-scores, AUC of ROC scores that are often used in anomaly detection tasks [38], the Average Precision (AP) [39] that summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, and Matthews Correlation Coefficient (MCC) [40] that takes into account true and false positives and negatives and is generally used even if the labels are very unbalanced.

5.2. Qualitative Analysis

In our pilot study (indicated in Sec. 4.3), the state-of-the-art methods focus on predictor extraction and applying boosting-based models to classify UE predictions. Our implemented XG-Boost classifiers achieve 0.19 precision, 0.26 recall, and 0.22 F1-scores, respectively on Dataset-A, which is much poorer than our strong baselines (in Table 3). Nevertheless, our proposed STIM model shows constant superiority over strong baselines on the Dataset-A, Dataset-B, and Dataset-C. Note that if a recall for the (minority) anomaly label is over 0.5, the model will be favored in practice given the extremely unbalanced dataset (0.98% UE labels), thus our proposed methods, including baselines and STIM, are highly effective. In addition, we observe that the STIM has increased performance when training on Dataset-A, Dataset-B, and Dataset-C, respectively, given these datasets only have different training slits but the same development and test sets. This finding indicates that training the model with the most recent (new) data is better than training the model with old datasets that are even larger. It answers our **RQ1** that we can train an end-to-end STIM model that performs well on all the datasets while the best model would be the one trained with the most recent data. It also demonstrates **RQ2** that the model can handle spatio-

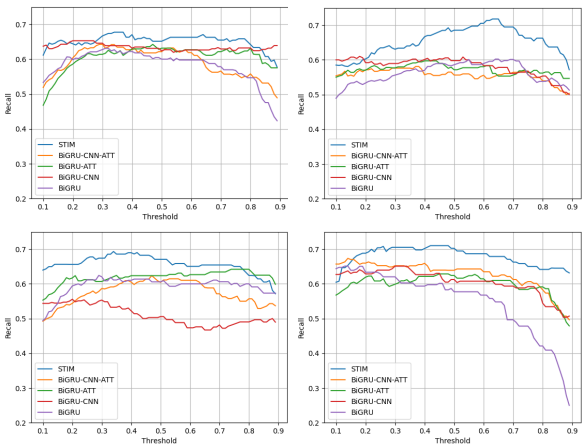

Figure 6. The F1 scores with respect to different thresholds on the Dataset-A (upper left), Dataset-B (upper right), Dataset-C (lower left), and Dataset-D (lower-right).

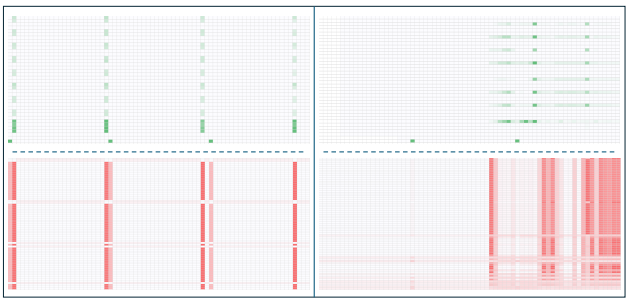

Figure 7. The memory features with high temporal (row) and spatial (column) variability are shown in the upper row, where the non-colors mean the values in the cell are close to 0, the deeper the greens the higher the values in the cell. The lower row is the attention map learned in the STIM model, where the deeper the reds the higher the (normalized) attention scores in corresponding positions. We observe the attention scores match the spatial (column) and temporal (row) locations of erroneous memory features, which suggests the STIM model can learn feature importance well.

temporal variability even training on data collected from different time windows. In comparison between Dataset-C and Dataset-D, the differences are that the training and development set in Dataset-D share the same time window but not in Dataset-C (Table 2). We observe that the performance in Dataset-D for the BiGRU-CNN-ATT and STIM models are generally better than those of Dataset-C, which answers our **RQ3** that having the training and development set sharing the same time window can help the models learn better hyper-parameters. In terms of other metrics, our proposed STIM model is constantly better than all the baselines except the ROCAUC scores on Dataset-D. We argue this is because STIM can not only learn extremely rare UE well but also learn the majority CE well.

5.3. Quantitative Analysis

We plot the precision-recall curve in Figure 5. The performance of most baseline models is close on all the datasets, where STIM is evidently better than the others. There is a significant fluctuation near the zero-recall regions in Dataset-A, Dataset-B, and Dataset-C, which is because these datasets are extremely unbalanced so none of the positive (UEs) are correctly predicted by most of the models. However, this is not observed in the Dataset-D, where the training set and development set have the same time window (Table 2). This finding indicates that the data-splitting strategy in spatio-temporal datasets may also influence the model performance. In Dataset-D, the baselines and STIM models all perform well, where STIM is better than the other baselines when recall is less than 0.6, but not competitive when the recall is greater. The precisions dramatically drop when the recall exceeds 0.6, which means the false positive goes high after a certain threshold. Hence, we further analyze how the thresholds influence the F1 score performance in Figure 6, which shows the F1 score changes with respect to different thresholds in all four datasets. We observe that the F1 scores of the baseline models do not change much when the threshold is in the range between 0.3 to 0.7 but our STIM model has slightly higher sensitivity. In other words, our STIM model can be tuned to achieve even better F1 scores. In terms of attention mechanism in the STIM model, Figure 7 shows the input memory features $\mathbf{X} \in \mathcal{R}^{m \times t}$ and their corresponding learned attention scores. We observe that the attention score is in line with the spatial and temporal locations on the feature map, which demonstrates that the STIM model does learn the importance of the input features.

6. Conclusion and Future Work

In Microsoft Azure datacenters, server crashes are predominantly caused by Memory Uncorrectable Errors (UEs). To address this, various machine learning methods have been employed to predict UEs in order to facilitate efficient maintenance and reduce server downtimes. However, predicting UEs is rather challenging due to data variability across spatial and temporal dimensions. To tackle this challenge, we propose the first Spatial-temporal Transformer in Memory (STIM) model for UE predictions, leveraging self-attention mechanisms to learn the importance of the input features. The performance evaluations on four variants of our developed Azure Memory Error Dataset (AMED) demonstrate that our proposed STIM model is greatly superior to existing state-of-the-art methods and strong baselines. Our results highlighted the potential of Language Models (LMs) in handling complex UE prediction tasks. In our future work, we intend to collect additional data from the Microsoft System Event Log (SEL) to evaluate the effectiveness of STIM. We

also aim to extend our approach to other hardware-centric tasks (e.g., predicting disk failures and CPU errors) to enhance server performance within the Azure datacenter.

References

[1] C. Li, Y. Zhang, J. Wang, H. Chen, X. Liu, T. Huang, L. Peng, S. Zhou, L. Wang, and S. Ge, "From Correctable Memory Errors to Uncorrectable Memory Errors: What Error Bits Tell," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 01–14, Nov. 2022. ISSN: 2167-4337.

[2] T. J. Dell, "A white paper on the benefits of chipkill-correct ecc for pc server main memory," 1997.

[3] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you?," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, FAST '07, (USA), p. 1–es, USENIX Association, 2007.

[4] M. Rinard, C. Cadar, D. Dumitran, D. M. Roy, and T. Leu, "A dynamic technique for eliminating buffer overflow vulnerabilities (and other memory errors)," in *20th Annual Computer Security Applications Conference*, pp. 82–90, IEEE, 2004.

[5] O. Mutlu, "The rowhammer problem and other issues we may face as memory becomes denser," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 1116–1121, 2017.

[6] Z. Cheng, S. Han, P. P. Lee, X. Li, J. Liu, and Z. Li, "An in-depth correlative study between dram errors and server failures in production data centers," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, pp. 262–272, IEEE, 2022.

[7] S. Levy, K. B. Ferreira, N. DeBardeleben, T. Siddiqua, V. Sridharan, and E. Baseman, "Lessons learned from memory errors observed over the lifetime of cielo," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 554–565, 2018.

[8] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 415–426, 2015.

[9] X. Du, C. Li, S. Zhou, M. Ye, and J. Li, "Predicting uncorrectable memory errors for proactive replacement: An empirical study on large-scale field data," in *2020 16th European Dependable Computing Conference (EDCC)*, pp. 41–46, 2020.

[10] X. Du and C. Li, "Predicting Uncorrectable Memory Errors from the Correctable Error History: No Free

Predictors in the Field," in *The International Symposium on Memory Systems*, (Washington DC DC USA), pp. 1–10, ACM, Sept. 2021.

[11] Q. Yu, W. Zhang, P. Notaro, S. Haeri, J. Cardoso, and O. Kao, "HiMFP: Hierarchical Intelligent Memory Failure Prediction for Cloud Service Reliability," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 216–228, June 2023. ISSN: 2158-3927.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] I. Giurgiu, J. Szabo, D. Wiesmann, and J. Bird, "Predicting dram reliability in the field with machine learning," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track*, Middleware '17, (New York, NY, USA), p. 15–21, Association for Computing Machinery, 2017.

[14] I. Boixaderas, D. Zivanovic, S. Moré, J. Bartolome, D. Vicente, M. Casas, P. M. Carpenter, P. Radojković, and E. Ayguadé, "Cost-aware prediction of uncorrected dram errors in the field," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15, 2020.

[15] M. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text classification using machine learning techniques.," *WSEAS transactions on computers*, vol. 4, no. 8, pp. 966–974, 2005.

[16] A. Parvat, J. Chavan, S. Kadam, S. Dev, and V. Pathak, "A survey of deep-learning frameworks," in *2017 International Conference on Inventive Systems and Control (ICISC)*, pp. 1–7, IEEE, 2017.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[20] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "Ammus: A survey of transformer-based pretrained models in natural language processing," *arXiv preprint arXiv:2108.05542*, 2021.

[21] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 87–110, 2022.

[22] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in neural information processing systems*, vol. 2, 1989.

[23] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[24] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[25] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787*, 2019.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," June 2021. arXiv:2010.11929 [cs].

[27] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[28] X. Li, M. C. Huang, K. Shen, and L. Chu, "An Empirical Study of Memory Hardware Errors in A Server Farm,"

[29] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.

[30] C. N. Kamath, S. S. Bukhari, and A. Dengel, "Comparative study between traditional machine learning and deep learning approaches for text classification," in *Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1–11, 2018.

[31] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *2016 8th IEEE international conference on communication software and networks (ICCSN)*, pp. 581–585, IEEE, 2016.

[32] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning

and deep learning methods for cybersecurity," *Ieee access*, vol. 6, pp. 35365–35381, 2018.

[33] Q. Hai, S. Zhang, C. Liu, and G. Han, "Hard disk drive failure prediction based on gru neural network," in *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 696–701, 2022.

[34] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[36] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[37] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 606–615, 2016.

[38] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32142–32159, 2022.

[39] M. Grobelnik, "Feature selection for unbalanced class distribution and naive bayes," in *ICML '99: Proceedings of the sixteenth international conference on machine learning*, pp. 258–267, Citeseer, 1999.

[40] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.