ML-PreP: Machine Learning Based Error Prediction for Phase Change Memory

1st Jake Ekoniak, IEEE Member
Department of Mathematics
University of California
Los Angeles, USA
jekoniak5741@g.ucla.edu

2nd Apoorva Rumale
Computer Science Department
California State University
Northridge, USA
apoorva-sanjay.rumale.462@my.csun.edu

3rd Marjan Asadinia
Computer Science Department
California State University
Northridge, USA
marjan.asadinia@csun.edu

2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC) | 979-8-3315-0769-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/CCWC62904.2025.10903761

***Abstract*—Phase Change Memory (PCM) is a non-volatile memory technology that shows great promise as a potential replacement for DRAM in main memory due to its scalability, low read latency, and potential for high storage capacity. However, as PCM—specifically Multi-Level Cell (MLC) PCM—becomes denser, it becomes increasingly susceptible to errors, a problem that current error correction strategies cannot efficiently address. In response, we have developed a predictive model, ML-PreP, specifically tailored for MLC PCM. This model is trained on MLC PCM data, including write patterns, cell states, and electrical input parameters, to understand the complex relationships between these inputs, output parameters, and error occurrences. Our model employs a multi-layer perceptron network and an AdaBoost regression model to demonstrate high accuracy in forecasting error types. Subsequently, a convolutional neural network estimates the number of errors per line, while an additional detection model pinpoints their locations, enabling the efficient application of standard error correction mechanisms. This approach underscores the potential of machine learning-driven error prediction to significantly enhance the robustness and efficiency of MLC PCM systems.**

***Index Terms*—Multi Level Cell (MLC), Phase Change Memory (PCM), Error Prediction, Accuracy.**

I. Introduction

Phase-Change Memory (PCM) is one of the most promising technologies among emerging non-volatile memories. It stands out for its non-volatility, scalability, and comparable read speeds relative to DRAM [1], [2]. PCM employs cells made of a chalcogenide alloy (GST), which can switch between crystalline and amorphous states, representing binary '1' and '0' respectively. These states are altered using heat and read by measuring their differing electrical resistances [1]–[3].

Recent advancements in PCM include the development of multi-level-cell (MLC) operation, allowing for storing multiple bits per cell. However, current PCM is not without its faults. Due to an asymmetry in the temperature needed for SET (0 to 1, 500°C) and RESET (1 to 0, 650°C) operations, Write Disturbance Errors (WDE) can arise, where heat from a RESET operation dissipates into adjacent cells, potentially triggering a SET operation if those cells are in a '0' state [4], [5]. Additionally, random bit flips, caused by factors like stochastic switching, resistance drift, and thermal variation, necessitate robust Error-Correcting Pointers (ECP) to maintain data integrity [8]. WDEs and Bit Flip errors become more prevalent as PCM cells are densely packed [6]. PCM also has a limitation on the number of write cycles per cell, typically around 10^7 - 10^8 cycles [7], considerably less than the virtually unlimited 10^{15} cycles offered by DRAM. Once a PCM cell reaches its end of life, it becomes stuck in its last state, resulting in 'stuck-at' errors or hard faults. These endurance and reliability issues are exacerbated in multilevel cell operations [8].

Existing techniques aim to reduce the number of writes per cell or use wear-leveling mechanisms to evenly distribute the write pressure across PCM cells, which helps prevent them from wearing out quickly [4], [6], [8], [14], [21], [24]. Although these types of schemes reduce write traffic per cell, they cannot significantly improve memory lifetime. In addition to these approaches, some methods propose low-energy and low-latency error correction schemes for Phase Change Memory (PCM) using a combination of BCH codes and Error-Correcting Pointers (ECPs) to address both transient and permanent errors. Specifically, these methods provide error correction and recovery mechanisms at the line level or page level. However, they introduce complexity in implementation and additional storage overhead [2], [9], [11], [12], [22], [23].

Addressing these challenges requires innovative solutions that can accurately predict errors while enhancing the overall reliability of PCM cells. Our proposed method, ML-PreP, introduces a novel approach by leveraging advanced machine learning techniques to predict error occurrences in PCM cells. This method integrates deep neural networks (such as multi-layer perceptron networks), ensemble learning models (such as AdaBoost regression models), and convolutional neural networks (CNNs) to achieve high prediction accuracy.

The deep neural network component of ML-PreP effectively predicts error occurrences based on key input parameters, including the read-to-write ratio, set energy, reset energy, the percentage of 0s and 1s, and the total read and write counts. While the neural network demonstrated strong predictive performance, it was the implementation of an adaptive algorithm with parameters finely tuned using grid search that achieved near-perfect prediction accuracy. This novel application of machine learning algorithms to PCM error prediction is critical for enhancing the effectiveness of existing error correction mechanisms, thereby significantly improving the reliability

979-8-3315-0769-5/25/$31.00 ©2025 IEEE

and performance of PCM technology. Building on this foundation, the convolutional neural network (CNN) component of ML-PreP adds granularity to error prediction, enabling precise error predictions for each memory line. Additionally, ML-PreP includes an error detection model specifically designed to enhance accuracy and seamlessly integrate with existing error correction algorithms.

The remainder of the paper is organized as follows: Section II details the proposed method, including the integration of NN model for optimizing write parameters in PCM. Section III discusses the evaluation setup, including the environment, parameter generation, initial findings from simulations conducted using NVMain and current status of the research and presents the results of our evaluation, highlighting the effectiveness of our approach in acquiring minimum loss while predicting write process metrics. Section IV concludes the paper by summarizing our contributions and highlighting the potential impact of our proposed methods on the advancement of PCM as a next-generation memory technology. This section also discusses further enhancements to our model and its applications in PCM technology.

II. Proposed Method

In this section, we explain our proposed ML-PreP approach in detail. We begin with a discussion on collecting data about the write process, and read process followed by an explanation of data preprocessing techniques to clean and prepare the data for machine learning model training.

We then describe the development of adaptive learning algorithm for optimizing error prediction by training machine learning models with the collected data.

A. Data Collection

The first step in the process of error prediction was creating workload files for NVMain, a simulator for MLC PCM interactions [18], [19]. NVMain simulator is widely recognized in academic research as a reliable and detailed simulator for emerging non-volatile memory technologies. In our work with NVMain, we use parameters derived from real PCM devices and prior research to ensure that our simulation results closely reflect realistic device behavior. We have also integrated real-world constraints, including temperature effects, process variations, and realistic endurance models, into our simulations. These efforts help bridge the gap between the simulated environment and actual device performance.

Our files, comprising of 100,000 operations each, dictate whether an operation is a Read or a Write, the operation's number, and the data involved. We considered workload files with the percentage of zeroes in the data ranging from 10-90% and with Read operations from 0-100%, the rest being Write operations. We also varied the sequence of operation numbers - random, consecutive, or strided (steps from 1 to 5). Varying workload files is a crucial step to make sure that our methods are robust to error occurrences for all types of MLC PCM applications. After workload files were generated, we injected each one with errors at different percentages. The three types of error we considered were Write Disturbance Errors (WDEs), Bit Flip errors, and stuck-at errors. WDEs may occur when a RESET operation is completed and immediately adjacent cells are in the '0' state, unintentionally setting them to '1'. We assume that the data is written continuously in memory. For each RESET operation, we simulated a WDE for each '0' state neighbor via single *Bernoulli trial* with a given percentage. This percentage was calculated based on set voltage, reset voltage, set pulse duration, and reset pulse duration values. Bit Flip errors occur randomly, thus after inserting the WDEs, Bit Flips were injected at a given percentage with a single *Bernoulli trial* for each bit. This percentage was calculated based on set pulse and reset pulse duration values.

$$Bitflip_{percentage} = \frac{(Pulse_{reset} - ResetPulse_{min})}{(ResetPulse_{max} - ResetPulse_{min})}\alpha_1 + \frac{(Pulse_{set} - SetPulse_{min})}{(SetPulse_{max} - SetPulse_{min})}\beta_1 + \gamma_1 \quad (1)$$

$$WDE_{\text{percentage}} = \frac{\left(\frac{E_{\text{reset}}}{E_{\text{set}}} - ERatio_{\text{min}}\right)}{(ERatio_{\text{max}} - ERatio_{\text{min}})}\alpha_2 + \frac{(Pulse_{\text{reset}} - ResetPulse_{\text{min}})}{(ResetPulse_{\text{max}} - ResetPulse_{\text{min}})}\beta_2 + \gamma_2 \quad (2)$$

In order to keep our Bit Flip error and WDE occurrence percentages within the target ranges of 10-30% and 10-40% respectively, we used the following values for the constants:

The constants can be modified to adjust the error occurrence percentages, allowing for flexible adaptation to various simulation scenarios. The last type of error we considered was stuck-at errors. Stuck-at errors occur due to the limited endurance of PCM, where a cell will die after a certain amount of write cycles, resulting in the cell becoming stuck at its last state. It has been found that counter-based modeling (e.g. after $1e^8$ writes, the cell dies) results in more accurate simulations than probabilistic modeling($1e^8$ chance of cell death at every write) [14], thus we did not explicitly insert stuck-at errors into the workload, but considered them through changing the endurance model in the NVMain simulator. We considered a mean cell endurance of $1e^8$ and variances of $1e^6$, $1e^7$, and $3e^7$ with a normal distribution. The possible error rates, shown in Table I, were designed to include both experimentally observed error rates and exceptionally high error rates to test the robustness of our models.

TABLE I
Percentage Range of Error Types.

Error Type	Percentage Range
Bit Flip	10% - 30%
Write Disturbance	10% - 40%

The parameters α, β, and γ represent scaling factors that influence the relationship between pulse durations and energy ratios with error occurrence rates in the proposed model

(Table II). Their values were determined based on extensive experimental observations in our simulation environment.

In practical applications, these constants should be recalibrated using real-world data from PCM chips to match the specific hardware characteristics. This calibration could be efficiently performed using techniques such as grid search, Bayesian optimization, or random search, depending on the observed error patterns in the system. By tuning these parameters to the unique behavior of different PCM implementations, the accuracy and effectiveness of the proposed model can be maximized, ensuring its viability in real-world environments.

TABLE II
CONSTANT VALUES.

Constant	Value
α_1	0.10
α_2	0.20
β_1	0.10
β_2	0.10
γ_1	0.10
γ_2	0.10

Once the errors were injected into the workloads, each workload was run through the NVMain simulator for 100,000,000 cycles at each of the endurance rates and with varying Set and Reset Energy values. Once completed, the results were scraped and compiled into a CSV file via a python script. For each NVMain simulation, a data entry is created that contains the information used in the creation of the associated workload file, the NVMain input parameters, and the NVMain output parameters. Then, we removed the Set Voltage, Reset Voltage, Set Pulse Duration, Reset Pulse Duration values from each entry, as they are explicitly used when calculating error percentages.

Finally, we trained two supervised multi-output learning models to predict the percentages of Bit Flip and Write Disturbance errors in each workload file. It is notable that trace files were generated based on real application data to closely represent actual workload patterns. By capturing typical data access behaviors and diverse operational characteristics, we ensured the trace files realistically simulate real-world scenarios, enhancing the reliability of our evaluations in NVMain

In Figure 1, we prepared the feature correlation map based on our observations from MLC data in the full system simulator. Key inputs and target outputs are also summarized in this figure. The output features with their respective input features are shown in Table III.

B. Modeling

Due to the synthetic nature of our data, there were no outliers, thus normalization could be done through Sklearn's StandardScaler. Initially, we explored several foundational models including linear regression, polynomial regression, and random forests to build a baseline understanding of the data's patterns. Building on these insights, we trained two complex models to capture more insights into error occurrence percentages, specifically a neural network and an AdaBoost

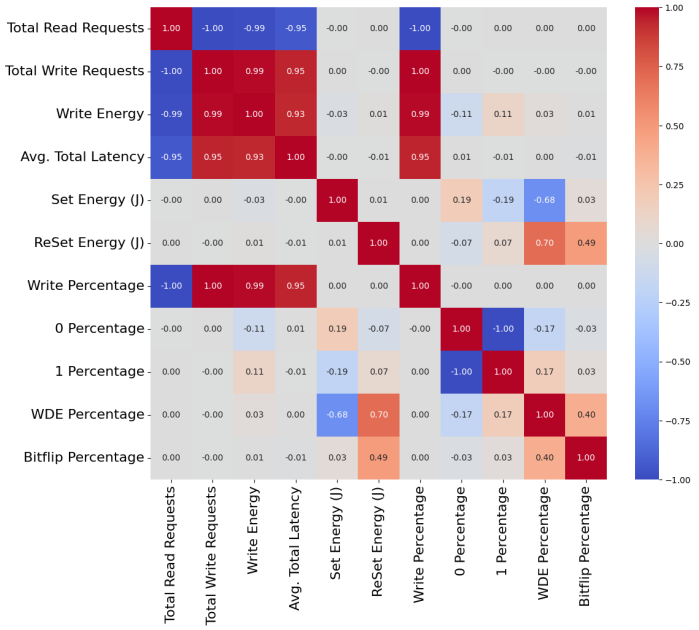

Fig. 1. Feature Correlation Map.

TABLE III
KEY INPUT AND TARGETS OF THE MODEL.

Key Input Features	Targets
Read:Write Ratio	WDE Percentage
Set Energy	Bit Flip Error Percentage
Reset Energy	
0 and 1 Percentages	
Write Energy	
Total Read and Write Counts	

Regression model. All models were trained on a random 80:20 train/test split.

1) Static: The neural network that we trained was a multilayer perceptron (MLP) network with two outputs. The network was constructed with the Keras Sequential API and tuned using the Keras Tuner [20]. To mitigate overfitting, a dropout layer with rate of 0.1 was placed after the final hidden layer and the early stopping callback was used with a patience value of 8 on the validation loss. The tuning of the network's hyperparameters was done through a combination of grid search and 5-fold cross validation. This selected the best combination of activation functions, neurons per layer, and optimizer that would lead to the best MSE performance while remaining generalizable to unseen data. The final structure of the model is detailed below, and was trained with the ADAM optimizer, learning rate of 0.001, and batch size of 10.

A simple diagram of the MLP NN is shown in Figure 2. Table IV also summarizes the layer types and specifications.

2) Adaptive: In addition to the MLP, an AdaBoost Regression model was also trained for the same task as shown in Figure 3. This architecture can be advantageous for MLC PCM systems as it requires less data than the MLP and is flexible

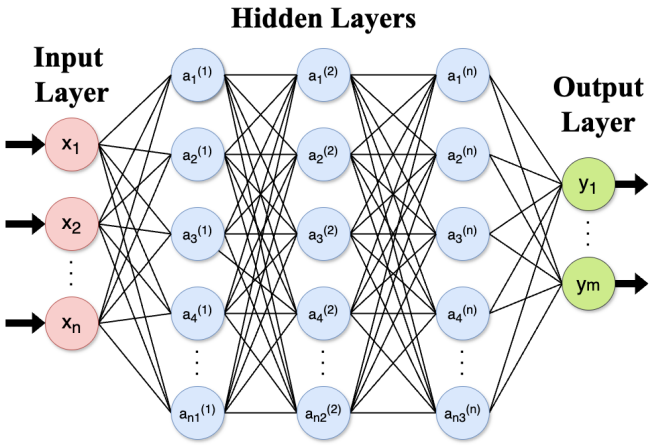

Fig. 2. MLP Network structure with input layers, hidden layers and output layers.

TABLE IV
ARCHITECTURE OF THE MLP.

Layer Type	Specifications
Input Layer	64 neurons, RELU activation, input shape of n features
Hidden Layer	64 neurons, RELU activation
Hidden Layer	64 neurons, RELU activation
Hidden Layer	64 neurons, RELU activation
Hidden Layer	32 neurons, RELU activation
Dropout Layer	Dropout rate of 0.1
Output Layer	2 output dimensions (for each error type)

with regards to model updates. Adaboost works through a combination of multiple weak learners, often decision trees, each considering the mistakes of their predecessors with more weight. The base learner for our AdaBoost model was a decision tree regressor with maximum depth of 6. Then, the model has a maximum of 50 decision trees, each one correcting the errors made by its predecessors. Due to the multi-output nature of the problem, the AdaBoost Regressor was wrapped within Sklearn's MultiOutput Regressor to extend its capability to handle multiple outputs.

Additionally, we employed 5-fold cross validation to confirm the generalizability of the model. The final structure is detailed in Table V.

TABLE V
ARCHITECTURE OF THE ADABOOST REGRESSION MODEL.

Component	Description
Base Estimator	DecisionTreeRegressor, depth of 6
AdaBoost Configuration	Number of estimators is 50
Multioutput Regressor	Extends AdaBoost for multioutput

C. Estimation

The previous models inspect error occurrences at the workload level, providing valuable insights into overall error trends. However, this approach is limited in granularity, as it does not account for the varying number of errors that occur at the line level, influenced by different operations and data patterns. To address this limitation, we developed a third model for more detailed analysis: a 1D convolutional neural network (CNN) that operates at a per-line granularity. This model takes each line of the workload file, along with the associated estimated error percentages, to predict the specific number of each error type present. To adapt the workload file for time-series modeling, we standardized the data section to uniform length and replicated the metadata (operation type, error percentages, 0:1 ratio) as static variables along each step of the series. Each data point was of the dimension (512, 5) representing 512 steps in the time series with 5 static features.

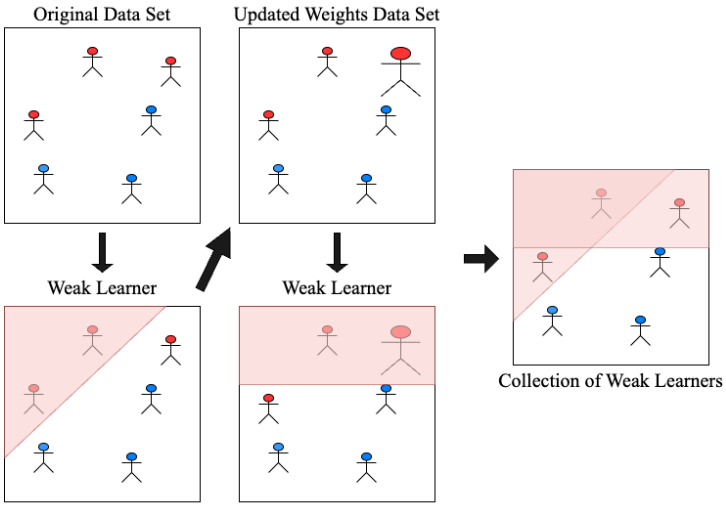

Fig. 3. The overall view of the AdaBoost model.

The CNN architecture consisted of two 1D convolutional layers designed to capture patterns in the sequential data. The first convolutional layer had 32 filters, while the second used 64 filters, both with a kernel size of 3 and ReLU activation functions to introduce non-linearity. These layers progressively extracted higher-level features from the data, capturing intricate relationships between operations and error occurrences. After the convolutional layers, the data was flattened and passed through a fully connected dense layer with 64 neurons and ReLU activation, enabling the model to make final predictions based on the extracted features.

To train the model, we used the ADAM optimizer, known for its efficient handling of sparse gradients, along with Mean Squared Error (MSE) as the loss function. Training was conducted over 10 epochs with a learning rate of 0.001 and a batch size of 32, balancing learning stability with computational efficiency. Table VI summarizes the details of the 1D Convolutional Neural Network parameters.

Figure. 4 illustrates a flow chart summarizing the proposed method's overall architecture and processes.

TABLE VI
ARCHITECTURE OF THE 1D CONVOLUTIONAL NEURAL NETWORK FOR ESTIMATION.

Layer Type	Specifications
Input Layer	Input size of n features
Conv1D Layer	32 filters, Kernel size 3, RELU activation
Conv1D Layer	64 filters, Kernel size 3, RELU activation
Flatten Layer	Flattens the input
Fully-Connected Layer	64 neurons, RELU activation
Output Layer	Dense Layer with 2 neurons

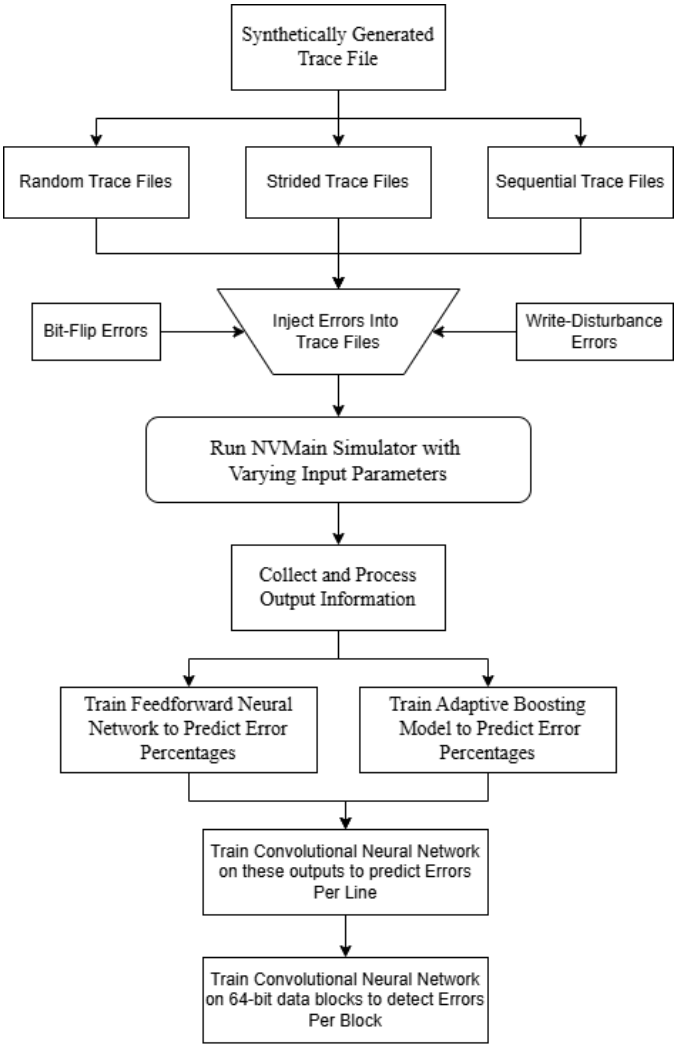

Fig. 4. Data Pipeline of our proposed method.

III. EXPERIMENTAL EVALUATION

This section provides a detailed overview of the simulation environment including its environment, simulator employed with its configuration parameters and workloads utilized.

A. Simulation Environment

We implemented our proposed method in NVMain [18], [19] which is a versatile, cycle-accurate memory simulator that models both conventional DRAM and new non-volatile memory (NVM) technologies like phase change memory (PCM). It offers detailed simulations of memory timing, energy consumption, and specific NVM traits such as limited write endurance and multi-level cells. Additionally, NVMain supports hybrid memory systems, fine-grained bank/subarray-level parallelism, and allows for custom memory controllers and address mapping schemes. For PCM, NVMain effectively models key aspects like asymmetric read/write latencies, write energy, and cell endurance, making it suitable for evaluating PCM optimization strategies [18], [19].

The simulation setup is outlined in Table VII. The setup includes a 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth. It also includes parameters such as clock frequency, bus width, CPU frequency, etc. These parameters were kept unchanged in the default configuration file in NVMain to mimic PCM as accurately as possible. These parameters are essential for accurately simulating and assessing the performance and energy consumption of the memory system under various conditions.

TABLE VII
CONFIGURATION ENVIRONMENT.

Parameter	Value
Memory Specifications	
Technology node	20nm
Operating voltage	1.8V
Device capacity	8Gb
Program bandwidth	40MB/s
Interface Specifications	
Clock frequency in MHz	400
Bus width in bits	64
Number of bits per device	8
CPU frequency in MHz	2000
MLC Parameters	
Number of MLC levels	2
Memory Controller Parameters	
Memory controller type	FRFCFS
Address mapping scheme	R:RK:BK:CH
Read queue size	32
Write queue size	32
Endurance Model Parameters	
Endurance model type	BitModel
Endurance distribution type	Normal
Endurance distribution mean	1000000
Endurance distribution variance	100000

B. Experimental Results

Our experimental evaluation seeks to test the effectiveness of the models developed for error prediction in MLC PCM. We began by assessing the performance of the error percentage prediction models. Using a dataset of 11,340 entries, these models were trained and tested with a random 80:20 split. Figures 5 and 6 show the prediction performance for bit flip and WDE. Performance was evaluated through three metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2). MSE offers a standard comparison to potential future models, MAE provides interpretability and contrast to MSE, and R^2 quantifies the predictive strength of our models.

The results of both the MLP and AdaBoost models are highly accurate, as shown in Table VIII. The graphs, however,

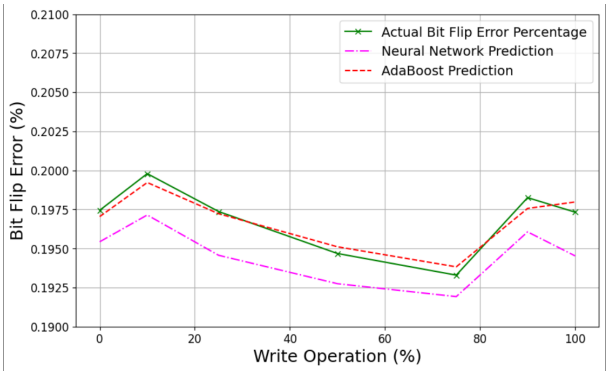

Fig. 5. Bit Flip Prediction Performance.

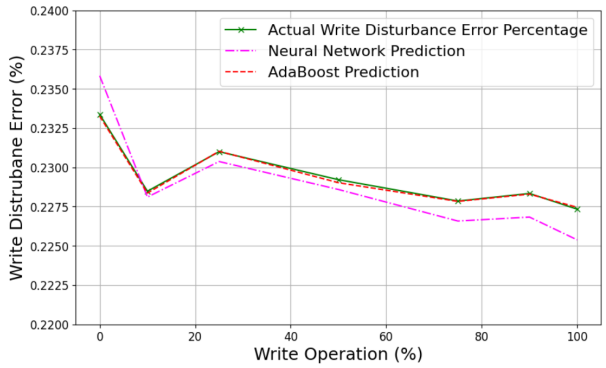

Fig. 6. WDE Prediction Performance.

exhibit slight slopes across the data points. These variations are attributed to the randomness inherent in the selection of testing and training sets. By observing the scale on the y-axis, it is clear that these slopes are largely inconsequential. Should the models predict values across the entire dataset, it is expected the resulting graphs would further approach a perfectly flat line, as Write Operation Percentage has no impact on error percentage. Another important aspect of this data is that in both models, WDEs are more accurately predicted than Bit Flips. This is an optimistic result, as Bit Flips are entirely random. Thus, if we are to extend this study to more PCM errors, which have an inherent pattern to their occurrence similar to WDEs, they would likely be predicted at the same or better rate than Bit Flip errors. It is also of note that the AdaBoost model, despite have a large MSE, has smaller MAE values. This likely indicates a slight over fitting of the MLP, something that the AdaBoost model is more robust to. Another interesting observation is that with our current dataset, the metrics from the MLP and the AdaBoost Regression model are comparable. Yet, when training with less data, the AdaBoost Model consistently outperformed the MLP. This occurred up until around 10,000 data points were gathered where the results converged.

In Figure 7, we assessed the performance of our convolutional neural network, which predicts the total number of errors at a line granularity (512 bits). This model was trained

TABLE VIII
PERFORMANCE METRICS FOR MLP AND ADABOOST REGRESSION.

Metric	MLP Network		AdaBoost Regression	
	WDE	Bit Flip	WDE	Bit Flip
MSE	8.36×10^{-6}	7.44×10^{-5}	6.20×10^{-6}	1.38×10^{-4}
MAE	0.0022	0.0045	0.0019	0.0093
R^2	0.997	0.957	0.997	0.919

and tested with an 80:20 split on 2,520 time data points, each with 512 steps.

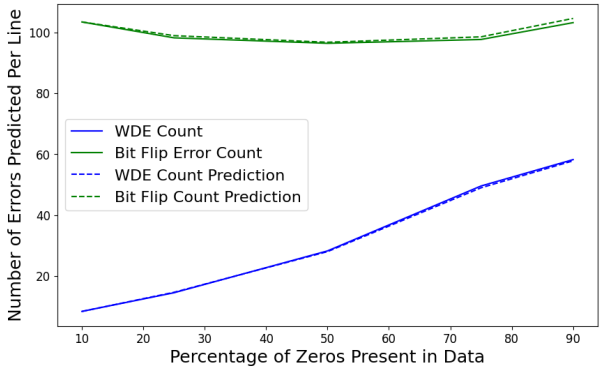

Fig. 7. CNN Error Count Predictions.

TABLE IX
ERROR PREDICTION COUNT METRICS FOR WDE AND BIT FLIP.

Metric	WDE Prediction	Bit Flip Prediction
MSE	0.192	0.378
MAE	0.297	0.496
R^2	0.999	0.999

The results here are highly accurate as well. As seen in the plot, while Bit Flips stay constant as the percentage of 0's changes, as expected, WDEs increase. This is again expected, as the more 0's are written in the data, the more opportunities WDEs have to occur. WDEs are slightly easier to predict than Bit Flips, owing to the structure behind WDE occurrences that the model can learn. Once again, this implies that other PCM error types will have a similar or better error prediction performance than Bit Flips. Additionally, as shown in Figures 8 and 9, as the Write Percentage increases, WDEs increase while Bit Flips remain constant. This is expected, as WDEs occur only during Write operations, so we can expect more WDEs in workload files with a higher number of Write operations.

C. Detection Analysis

While the estimation model excelled at predicting error counts on individual lines of workload files, the detection model enhances this by working at a finer resolution—evaluating 64-bit data blocks to detect the presence of

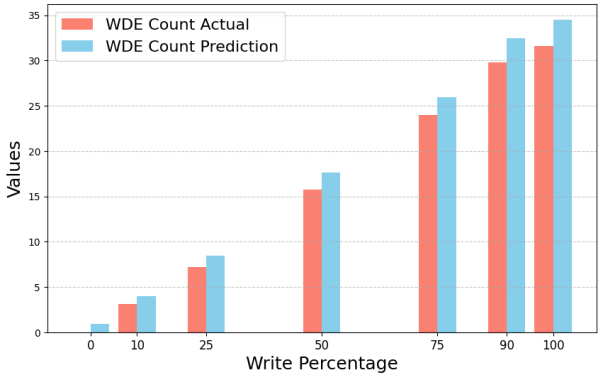

Fig. 8. CNN WDE Predictions Per Write Percentages.

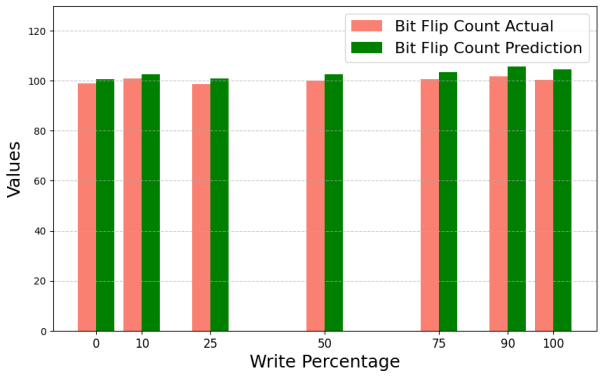

Fig. 9. CNN Bit Flip Predictions Per Write Percentages.

errors. This choice of block size was intentional: 64 bits is large enough to capture meaningful error patterns without imposing significant overhead, allowing efficient processing even at this granular level. Additionally, a 64-bit block size aligns with the standard for many error correction algorithms, making it well-suited for seamless integration with correction algorithms following detection. It is also notable that this method is designed with scalability in mind, making it applicable to larger block sizes beyond the current 64-bit block structure. The model's architecture and detection capabilities are expected to adapt effectively, enabling robust error prediction and detection across larger memory blocks while maintaining high accuracy and efficiency. Detection is achieved through a flip mask, where each block is assigned a mask value of 1 or 0, indicating the presence or absence of an error, respectively. In this context, a mask value of 0 signifies no detected error in the block, while a 1 denotes an error.

For this task, a dataset of 3,200,000 64-bit blocks was processed enabling effective use during training and evaluation. The model architecture is designed for binary classification, identifying whether each 64-bit block contains an error. The architecture consists of a 1D convolutional layer with 64 filters (kernel size of 3) and ReLU activation to detect local patterns indicative of errors, followed by a MaxPooling1D layer with a pool size of 2 to reduce dimensionality and preserve essential features. The flattened output then passes through a dense layer with 128 neurons and ReLU activation, which further refines the features for classification. Finally, a single neuron with a sigmoid activation function in the output layer provides a binary classification, labeling each block as error-free or containing an error.

Table X summarizes the details of the 1D Convolutional Neural Network parameters for Detection model. Compiled with the Adam optimizer this model was trained for 30 epochs with an 80-20 train-validation split, optimizing both performance and generalization. Various thresholds were tested during validation to fine-tune the model's sensitivity, resulting in high accuracy in detecting errors across data blocks, and confirming its efficacy for precise error detection in large-scale workload files. In the context of flip mask prediction for error detection, applying a threshold to the model's probabilistic outputs enables precise classification of each data block as either containing an error or being error-free. The threshold acts as a confidence cutoff, where predictions above this value indicate the likelihood of an error occurrence and are marked in the flip mask as '1' (indicating a detected error), while predictions below the threshold are marked as '0' (indicating no error). This approach is crucial for the effective identification of Errors and allows for systematic differentiation between blocks likely impacted by errors and those unaffected.

For example, if a model predicts that a specific 64-bit block has a probability of 0.75 for containing an error, applying a threshold of 0.9 would classify this block as error-free (flip mask '0'), while a threshold of 0.7 would classify it as containing an error (flip mask '1'). This flexibility in threshold selection is beneficial for tailoring the error detection sensitivity of the model, as higher thresholds may reduce unnecessary flips by only flagging high-confidence predictions as errors, while lower thresholds capture a broader range of potential errors. Through this targeted threshold application, the flip mask prediction process can be adjusted to meet the reliability requirements of the error detection system.

TABLE X
ARCHITECTURE OF THE 1D CONVOLUTIONAL NEURAL NETWORK FOR ERROR DETECTION.

Layer Type	Specifications
Input Layer	Input size of 64 bits per block
Conv1D Layer	64 filters, Kernel size 3, RELU activation
MaxPooling1D Layer	Pool size of 2
Flatten Layer	Flattens the input
Fully-Connected Layer	128 neurons, RELU activation
Output Layer	Dense Layer with 1 neuron, Sigmoid activation

The analysis of Bit Flip accuracy across different thresholds and error rates reveals consistent model performance. In the Figure 10 , we see that accuracy remains around 60% across thresholds ranging from 0.3 to 0.9, indicating that the threshold level has minimal impact on Bit Flip prediction accuracy. This stability suggests that the model's ability to predict Bit Flips is robust to changes in threshold.

Extending this analysis, Figure 11 shows Bit Flip accuracy

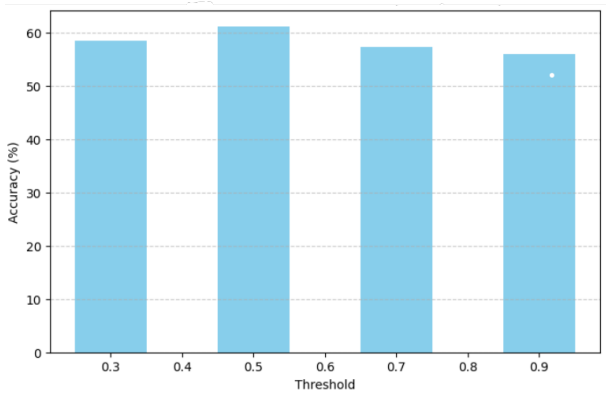

Fig. 10. CNN Bit Flip Detection Accuracy Per 64-bit Block.

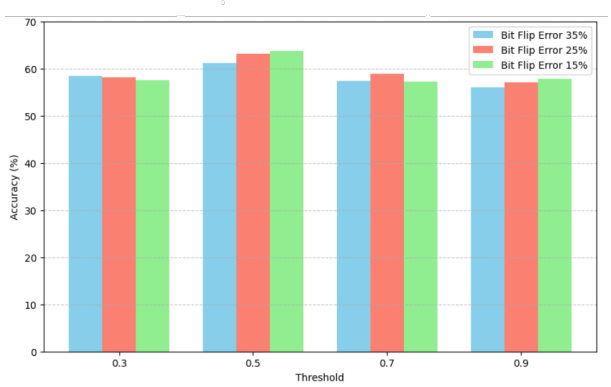

Fig. 11. CNN Bit Flip Detection with varying Bit Flip Error Percentages.

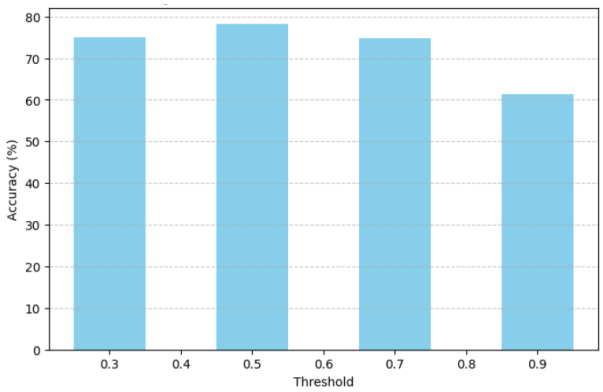

Fig. 12. CNN WDE Detection Accuracy Per 64-bit Block.

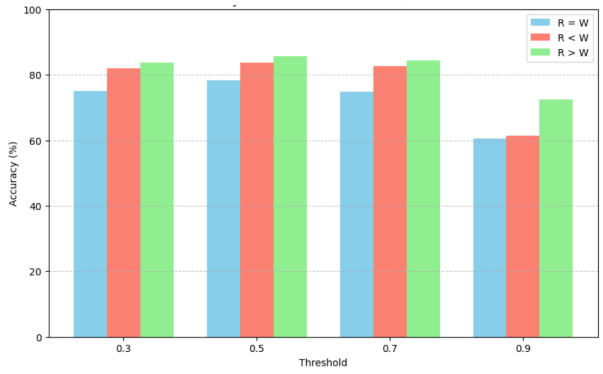

Fig. 13. CNN WDE Detection Accuracy with varying Read/Write Ratios.

at various error rates (15%, 25%, and 35%), with accuracy levels remaining stable regardless of error probability. Although the accuracies across thresholds are similar, a threshold of 0.5 consistently yields the highest accuracy across all scenarios. This stability across threshold levels and error rates underscores the model's robustness in handling Bit Flip errors, delivering reliable predictive performance even as error characteristics vary.

Figure 12 analyzes WDE prediction accuracy across varying thresholds, demonstrating that the model performs consistently well, with stable accuracy levels as thresholds range from 0.3 to 0.9. A threshold of 0.5 notably yields the highest accuracy, reinforcing its optimality for WDE prediction.

Figure 13 examines accuracy under different read:write ratios, showing that scenarios with a higher frequency of write operations ($R < W$) yield the best accuracy, which aligns with the nature of WDEs as they occur exclusively during write operations.

Moreover, Figure 14 displays accuracy at different WDE error probabilities (15%, 25%, and 35%), where the highest error rate (35%) achieves the highest accuracy, especially at lower thresholds. This trend suggests that the model becomes increasingly effective at predicting WDEs as their probability of occurrence rises.

Together, these results suggest that the model becomes increasingly effective at predicting WDEs both in environments dominated by write operations and as the probability of WDE occurrence rises.

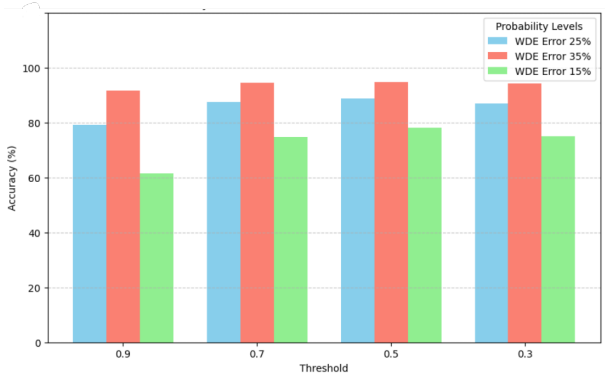

Fig. 14. CNN WDE Detection Accuracy with varying WDE Error Percentage.

IV. Conclusion

Phase-Change Memory (PCM) faces challenges such as high write energy, limited endurance due to repeated phase transitions, and thermal crosstalk between adjacent cells. Addressing these challenges involves optimizing write algorithms to reduce latency, developing error correction mechanisms to

enhance endurance, and refining thermal management techniques to minimize interference. Integrating machine learning models can also improve predictive maintenance and adaptive control, further extending PCM's viability in memory applications. In this study, we introduced a novel machine learning-based approach to error prediction in Multi-Level Cell Phase-Change Memory based on observed patterns. Our models, which include a multi-layer perceptron network, an AdaBoost regression model, and a convolutional neural network, achieve high accuracy in forecasting error information and facilitate efficient error detection. This approach minimizes processing overhead, making it highly suitable for high-performance memory systems. Our results confirm the potential of machine learning in emerging phase-change memory systems, laying the groundwork for error correction mechanisms that enhance both the accuracy and efficiency of these systems.

As future work, we aim to further refine our machine learning-based error prediction for MLC PCM by expanding our research to address additional error types, such as resistance drift and crosstalk errors. Incorporating these error types will enhance the robustness of our predictive models and improve overall efficiency when combined with correction techniques. Additionally, we plan to integrate our predictive models and detection model with new error correction methods. This combined error prediction and correction system will help identify practical challenges for real-world implementation and provide valuable insights into the system's operational efficiency.

V. Acknowledgment

This work was supported by the National Science Foundation (NSF) under Grant Nos. [CNS-2244391] and [2318553]. We gratefully acknowledge the NSF for their financial support through both grants and for providing the resources necessary to conduct this research. The views expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] H. Luo et al., "Write Energy Reduction for PCM via Pumping Efficiency Improvement," ACM Transactions on Storage, vol. 14, no. 3, pp. 1–21, Aug. 2018, doi: https://doi.org/10.1145/3200139.

[2] A. Ehrmann, T. Blachowicz, G. Ehrmann, and T. Grethe, "Recent developments in phase-change memory," Applied Research, Jun. 2022, doi: https://doi.org/10.1002/appl.202200024.

[3] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Throughput Enhancement for Phase Change Memories," IEEE Transactions on Computers, vol. 63, no. 8, pp. 2080–2093, Aug. 2014, doi: https://doi.org/10.1109/tc.2013.76.

[4] Manuel Le Gallo and A. Sebastian, "Phase-change memory," Elsevier eBooks, pp. 63–96, Jan. 2020, doi: https://doi.org/10.1016/b978-0-08-102782-0.00003-4.

[5] J. Jang, W. Shin, J. Choi, Y. Kim, and L.-S. Kim, "Sparse-Insertion Write Cache to Mitigate Write Disturbance Errors in Phase Change Memory," I.E.E.E. transactions on computers/IEEE transactions on computers, vol. 68, no. 5, pp. 752–764, May 2019, doi: https://doi.org/10.1109/tc.2018.2881137.

[6] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," ACM SIGARCH Computer Architecture News, vol. 37, no. 3, pp. 2–13, Jun. 2009, doi: https://doi.org/10.1145/1555815.1555758.

[7] L. Jiang, Y. Zhang, and J. Yang, "Mitigating Write Disturbance in Super-Dense Phase Change Memories," Jun. 2014, doi: https://doi.org/10.1109/dsn.2014.32.

[8] Marjan Asadinia and Christophe Bobda, "Enhancing Lifetime of PCM-Based Main Memory with Efficient Recovery of Stuck-at Faults," Jul. 2018, doi: https://doi.org/10.1109/isvlsi.2018.00072.

[9] X. Zhang, F. Cai, and M. Anantram, "Low-Energy and Low-Latency Error-Correction for Phase Change Memory", 2013.

[10] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Improving phase change memory performance with data content aware access," arXiv (Cornell University), Jun. 2020, doi: https://doi.org/10.1145/3381898.3397210.

[11] C. Zhang, K. Abdelaal, A. Chen, X. Zhao, W. Wen, and X. Guo, "ECC Cache: A Lightweight Error Detection for Phase-Change Memory Stuck-At Faults," doi: https://doi.org/10.1145/3400302.3415650.

[12] L. Mukhanov, K. Tovletoglou, H. Vandierendonck, D. Nikolopoulos, and G. Karakonstantis, "Workload-Aware DRAM Error Prediction using Machine Learning."

[13] E. Baseman et al., "Physics-Informed Machine Learning for DRAM Error Modeling."

[14] M. Kim, H. Lee, H. Kim and H. -J. Lee, "WL-WD: Wear-Leveling Solution to Mitigate Write Disturbance Errors for Phase-Change Memory," in IEEE Access, vol. 10, pp. 11420-11431, 2022, doi: 10.1109/ACCESS.2022.3145986.

[15] H. Rong, Cn, and L. Chang, "Application of an Improved BP-AdaBoost Model in Semiconductor Quality Prediction."

[16] B. Gou, Y. Xu, S. Fang, R. A. Pratama and S. Liu, "Remaining Useful Life Prediction for Lithium-ion Battery Using Ensemble Learning Method," 2019 IEEE Power & Energy Society General Meeting (PESGM), Atlanta, GA, USA, 2019, pp. 1-5.

[17] M. Shi, P. Mo and J. Liu, "Deep Neural Network for Accurate and Efficient Atomistic Modeling of Phase Change Memory," in IEEE Electron Device Letters, vol. 41, no. 3, pp. 365-368, March 2020, doi: 10.1109/LED.2020.2964779.

[18] M. Poremba and Y. Xie, "NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories," in *2012 IEEE Computer Society Annual Symposium on VLSI*, Amherst, MA, 2012, pp. 392-397, doi: 10.1109/ISVLSI.2012.82.

[19] M. Poremba, T. Zhang and Y. Xie, "NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140-143, July-Dec. 2015, doi: 10.1109/LCA.2015.2402435.

[20] Chollet, F., "keras," GitHub, Jul. 07, 2021. https://github.com/fchollet/keras.

[21] M. Asadinia, M. Jalili, and H. Sarbazi-Azad. "Bless: a simple and efficient scheme for prolonging pcm lifetime." In Proceedings of the 53rd Annual Design Automation Conference, page 93. ACM, 2016.

[22] R. Azevedo, J. D. Davis, K. Strauss, P. Gopalan, M. Manasse, and S. Yekhanin. "Zombie memory: Extending memory lifetime by reviving dead blocks". In ISCA, 2013.

[23] J. Fan, S. Jiang, J. Shu, Y. Zhang, and W. Zhen. Aegis: Partitioning data block for efficient recovery of stuck-at-faults in phase change memory. In Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, pages 433–444. ACM, 2013.

[24] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda. Dynamically replicated memory: building reliable systems from nanoscale resistive memories. In ASPLOS, pages 3–14, Mar. 2010.